

# Ordenagailuen egitura (240306)

Candido Aramburu

2023-10-02

# Edukien taula

I Instruction Repertoire Architecture (ISA): von Neumann ordenagailua, datuak, argibideak, programazioa . . . . .	1
1. Konputagailuen Egituraren Sarrera . . . . .	2
1.1. Sarrera . . . . .	2
1.2. Makina baten arkitektura . . . . .	2
1.3. Arkitektura HW ikuspegitik . . . . .	3
1.4. Programazio Lengoaiak eta Makina Lengoaia . . . . .	7
1.5. Software/Hardware Interfazea . . . . .	8
1.6. Oharrak . . . . .	10
1.7. Ikastaroa . . . . .	10
1.7.1. Oinarrizko Bibliografia . . . . .	hamaika
1.7.2. Bibliografia Osagarria . . . . .	13
1.8. Fakultatea . . . . .	13
1.9. Informatika Gradua . . . . .	13
1.10. Egutegiak . . . . .	14
1.11. Problemen ebazpenaren bidezko ariketak . . . . .	hamabost
1.12. Praktikak . . . . .	hamabost
1.12.1. Gidoiak . . . . .	hamabost
1.12.2. Ebaluazio Ariketak . . . . .	hamabost
1.12.3. Memoriak . . . . .	hamabost
1.13. Baliabide informatikoak . . . . .	hamabost
1.13.1. UPNA . . . . .	hamabost
1.13.2. Lanpostuak: 32 eta 64 bit . . . . .	16
1.13.3. Erregistroak . . . . .	17
1.14. Praktika Taldeak . . . . .	17
1.15. Metodologia . . . . .	22
1.15.1. Kredituen banaketa . . . . .	23
1.15.2. Praktiketarako kredituen banaketa . . . . .	23
1.16. Ebaluazioa . . . . .	23
1.17. Azterketak: Edukiak . . . . .	24
2. Von Neumann Arkitektura . . . . .	25
2.1. Von Neumann Arkitektura . . . . .	25
2.1.1. Ikastaroa . . . . .	25
2.1.2. Testuinguru historikoa . . . . .	25
2.2. Institute Advanced Machine (IAS): Arkitektura . . . . .	26
2.2.1. Erreferentzia . . . . .	26
2.2.2. sum1toN Programaren adibidea . . . . .	26
23. IAS ordenagailuaren egitura . . . . .	28
2.3.1. Moduluak . . . . .	28
2.3.2. Prozesatzeko Unitate Zentrala (CPU) . . . . .	30
2.3.3. Memoriak . . . . .	31
2.3.4. Autobusa . . . . .	32
2.3.5. Sarrera Irteera (I/O) . . . . .	33
2.3.6. Irakaskuntza Zikloaren animazioa . . . . .	33
2.4. ISA: IAS Machine Instruction Repertoire Architecture . . . . .	33
2.4.1. IAS ordenagailuaren datuen formatua eta argibideak . . . . .	33

2.4.2. ISA Errepetorioa . . . . .	35
2.4.3. ISA interfazea. . . . .	37
2.5. Programazioa IAS Asanblea Lengoain. . . . .	38
Lengoain Programa bat Garatzeko Estrategia . . . . .	38
Hamaseimala. . . . .	39
2.5.3. 1. adibidea: sum1toN.ias . . . . .	40
2.5.4. IASSim hizkuntzako programen adibideak . . . . .	46
Makinaren Eragiketa: Datuen Bidea . . . . .	2.6. IAS
Ondorioak . . . . .	47
3. Datuen irudikapena . . . . .	48
Ikastaroa. . . . .	48
3.2. Helburua. . . . .	48
3.3. Datuak eta argibideak: Kodeketa bitarra. . . . .	48
3.4. Bit, Byte, Word. . . . .	48
Zenbaki osoak. . . . .	3.5. Zenbaki hamartarrak . . . . .
3.5.1. Oinarri hamartarra . . . . .	49
3.5.2. Oinarri bitarra . . . . .	49
3.5.3. Bihurketa hamartar-bitarra. . . . .	berrogeita hamar
3.5.4. Oinarri octala . . . . .	51
3.5.5. Kalkulagailua . . . . .	51
3.5.6. Python . . . . .	51
3.5.7. Zenbaki oso sinatutakoak . . . . .	
52 3.6. Zenbaki errealak . . . . .	54
3.6.1. Koma finkoa . . . . .	54
3.6.2. Koma flotagarria . . . . .	55
3.7. Karaktere mota . . . . .	57
3.7.1. ASCII. . . . .	57
3.7.2. Python . . . . .	60
3.7.3. Unicode UTF-8. . . . .	60
3.8. ISO-8859-1. . . . .	62
3.8.1. Programazioa C-n . . . . .	62
Beste batzuk . . . . .	3.8.2. Beste batzuk . . . . .
4. Eragiketa aritmetikoak eta logikoak. . . . .	64
Ikastaroa. . . . .	4.1. Beste batzuk . . . . .
4.2. Helburua. . . . .	64
4.3. Sarrera . . . . .	64
4.4. Aritmetika Binarioa . . . . .	64
4.4.1. 2 modulu batuketa (bitarra) bitar hutsean (NO NATURALA) . . . . .	64
modulo kenketa (bitarra) bitar hutsean . . . . .	4.4.2. 2 modulu batuketa (bitarra) bitar hutsean . . . . .
Kenketa 2 moduluan (bitarra) 2 osagarrian . . . . .	66
batuketa (hamasezimala) . . . . .	4.4.3. Kenketa 2 moduluan (bitarra) 2 osagarrian . . . . .
moduluan (Hamaseitarra) . . . . .	67
motak. . . . .	4.4.4. 16. moduluko batuketa (hamasezimala) . . . . .
logikoak . . . . .	70
Eragileak . . . . .	4.4.5. Kenketa 16. moduluko batuketa (hamasezimala) . . . . .
Biderketa . . . . .	70
Programazioa. . . . .	4.4.6. C -ko aldagai motak . . . . .
funtzio matematikoak . . . . .	71
	4.5. Eragiketa logikoak . . . . .
	71 4.5.1. BITWISE
	72 4.6. Eragileak . . . . .
	72 4.7. Biderketa . . . . .
	73 4.7.1. Programazioa. . . . .
	73

4.7.2. Aplikazio . . . . .	73
4.8. Hardwarea. . . . .	73
4.8.1. Zirkuitu digitalak. . . . .	73
Unitate Logiko Aritmetikoa (ALU) . . . . .	73
EFLAG bandera-erregistroa. . . . .	74
Float Point Unitatea-FPU. . . . .	76
5. Jarraibideen irudikapena. . . . .	77
Ikastaroa. . . . .	77
5.1.1. Bibliografia . . . . .	77
5.2. Helburuak. . . . .	77
5.2.1. Baldintzak . . . . .	77
5.3. Goi-mailako programazio-lengoaiak vs. behe-mailako programazio-lengoaiak. . . . .	77
hizkuntzak. . . . .	77
eta muntaia-lengoaiak. . . . .	78
5.3.1. Goi mailako hizkuntzak. . . . .	77
5.3.2. Makina-lengoaiak . . . . .	77
5.4. Makina baten instrukzio-elementuak. . . . .	78
5.4.1. Operando arkitektura motak: adibideak. . . . .	79
5.5. Argibideak x86 arkitekturako makina-lengoaien. . . . .	80
5.6. Ordenagailuetarako, oro har, muntatzaile-lengoaien (ASM) jarraibideen irudikapena. . . . .	80
5.6.1. Sarrera . . . . .	80
5.6.2. Eragiketa-kodeak. . . . .	81
Eragileak: Helbideratzeko moduak. . . . .	82
versus AT&T hizkuntza. . . . .	85
arkitekturaren muntaia-lengoaiak. . . . .	85
hizkuntzan. . . . .	85
5.7. Intel hizkuntza	85
5.7.1. i386/amd64	85
5.7.2. Argibideen sintaxia INTEL	85
moduak. . . . .	86
Kokapena. . . . .	86
5.8.2. Helbideratzeko moduak. . . . .	86
5.9. Programak ASM hizkuntzan eta Binary hizkuntzan. . . . .	88
6. Mihiztadura Lengoaiaren Programazioa (x86): Goi-mailako lengoaien oinarrizko eraikuntzak. . . . .	89
6.1. Ikastaroa. . . . .	89
6.2. Sarrera . . . . .	89
6.2.1. Helburuak. . . . .	89
6.2.2. Baldintzak . . . . .	89
6.2.3. Erreferentziak. . . . .	89
6.3. Ordenagailuaren egitura Intel x86-64 Arkitekturarekin. . . . .	90
ISA x86 eskuliburuak. . . . .	90
6.3.2. Sarrera. . . . .	90
6.3.3. CPU-Memoria. . . . .	90
6.3.4. x86 CPUaren barneko erregistroak. . . . .	90
6.4. Datuen irudikapena batzar-lengoaien (ASM) i386/amd64 arkitekturarako. . . . .	93
6.4.1. Datu mota . . . . .	93
6.4.2. x86	93
eragigaien tamaina. . . . .	94
6.4.3. Byte-lerroatzea: Big-Little Endian. . . . .	94
Adibidea . . . . .	96
6.5. I386/amd64 arkitekturarako jarraibide multzoa muntatzaile-lengoaien (ASM): Eragiketak. . . . .	96

6.5.1. Adibidea . . . . .	96
6.5.2. Eskuliburu azkarra. . . . .	96
6.5.3. Eskuliburuak eta Taulak. . . . .	96
6.5.4. Eragiketa-kodeen deskribapen mota Intel eskuliburuan . . . . .	96
6.6. Oinarrizko Mnemoteknia (Azalduta) . . . . .	97
6.6.1. Eragiketa aritmetikoak . . . . .	97
6.6.2. Seinalearen luzapena . . . . .	98
6.6.3. Tamaina aldaketa. . . . .	98
6.6.4. Eragiketa boolearrak . . . . .	99
6.6.5. Baldintzapeko tratamendua: CMP,TEST,SETcc. . . . .	99
6.6.6. Jauziak . . . . .	100
6.6.7. Desplazamendua eta biraketa . . . . .	101
6.6.8. Aldatu Endianess . . . . .	101
6.7. Argibide-formatua: ISA Intel x86-64 . . . . .	101
6.8. Azpirrutinak . . . . .	101
6.8.1. Erreferentziak . . . . .	101
6.8.2. Sarrera . . . . .	101
6.8.3. C hizkuntza: Funtzio adierazpena. . . . .	102
6.8.4. Funtzioen habiaratzea . . . . .	104
6.8.5. Pila/Markoa . . . . .	104
6.8.6. Azpirrutinen argumentuak . . . . .	104
6.8.7. Azpirutina deia . . . . .	105
6.8.8. Azpirrutinaren definizioa . . . . .	105
6.8.9. Animazioa: habia-deiak . . . . .	106
6.8.10. Kontserbatzeko Erregistroak . . . . .	106
6.8.11. Azpirrutinaren itzulera. . . . .	107
6.8.12. Bateria egoera . . . . .	107
6.8.13. Programazio ariketa praktikoak. . . . .	109
6.9. Sistema eragilera deiak. . . . .	109
6.9.1. Sarrera . . . . .	109
6.9.2. amd64 arkitektura. . . . .	109
6.9.3. i386 arkitektura . . . . .	110
6.9.4. Dei-kodeak. . . . .	110
6.9.5. Adibidea . . . . .	114
II Oinarrizko Unitateak: Prozesadore Zentrala, Memoria Unitatea, Sarrera/Irteerako Mekanismoak . . . . .	117
7. Prozesadore zentrala . . . . .	118
7.1. Ikastaroa. . . . .	118
7.2. Erref. . . . .	118
7.3. Sarrera . . . . .	118
7.4. Ordenagailua Programatzailaren ikuspuntutik (I) . . . . .	119
7.4.1. Argibide multzoa . . . . .	119
7.4.2. Adibideak: Intel x86, Motorola 68000, MIPS, ARM . . . . .	120
7.5. Mikroarkitektura: Unitate Funtzionalak HW ikuspegitik . . . . .	120
7.5.1. Sarrera . . . . .	120
7.5.2. Instrukzioa gauzatzeko fazeak . . . . .	120
7.5.3. Adibidea: Von-Neumann IAS makina . . . . .	124
7.5.4. Instrukzio-zikloaren ezarpena: CPU. . . . .	124
7.5.5. Mikroprogramatutako Kontrol Unitatea . . . . .	128

7.6. CPU adibideak: IAS eta Intel 4004 . . . . .	129
7.6.1. IAS. . . . .	129
7.6.2. Intel 4004 . . . . .	129
7.7. Ordenagailua programatzalearen ikuspuntutik (II) . . . . .	132
7.7.1. Abstrakzio mailak edo geruzak . . . . .	132
7.7.2. Softwarearen bateragarritasuna . . . . .	135
7.8. Ordenagailu pertsonala: T560 ordenagailu eramangarria. . . . .	135
7.8.1. linux komandoak . . . . .	137
7.8.2. aplikazioak. . . . .	137
7.8.3. Web informazioa . . . . .	137
7.8.4. sistema eragilea . . . . .	138
7.8.5. Plataforma. . . . .	138
7.8.6. CPU. . . . .	139
7.9. Irakaskuntza-mailako paralelismoa (ILP). . . . .	143
7.9.1. Pipeline (Segmentazioa). . . . .	143
7.9.2. VLIW vs Superscalar . . . . .	146
7.10. CISC/RISC arkitekturak . . . . .	148
7.10.1. Sarrera . . . . .	148
7.10.2. CISC . . . . .	149
7.10.3. RISC . . . . .	149
7.10.4. Gaiak . . . . .	149
7.10.5. SW . . . . .	149
7.10.6. Konparazio taula . . . . .	150
7.11. Mikroarkitektura modernoak . . . . .	150
7.12. Ariketak . . . . .	150
8. Sarrera/Irteera Mekanismoak. . . . .	151
8.1. Ikastaroa. . . . .	151
8.2. Bibliografia . . . . .	151
8.3. Periferikoak . . . . .	151
8.3.1. Adibideak . . . . .	151
8.3.2. Eredua . . . . .	151
8.4. Teklatua. . . . .	152
8.5. Konputagailuen Arkitektura. . . . .	152
8.5.1. Von Neumann . . . . .	152
8.5.2. CPU-I/O konexioa . . . . .	153
8.5.3. I/O kontrolagailua . . . . .	153
8.5.4. Helbide espazioa. . . . .	155
8.5.5. Autobusak . . . . .	156
8.5.6. Analisia: Lenovo T520 - Disko gogorra . . . . .	157
8.6. I/O programa . . . . .	158
8.6.1. Iturburu-modulua . . . . .	158
8.7. Gidaria: Sistema Eragilea . . . . .	158
8.7.1. I/O kudeatzailea: hierarkia. . . . .	158
8.7.2. Iturburu kodea . . . . .	159
8.7.3. Kontzeptua . . . . .	159
8.7.4. Gidaria erabiltzea . . . . .	159
8.8. I/O Interfazea Ezartzeko Mekanismoak . . . . .	160
8.8.1. Sarrera . . . . .	160

8.8.2. Inkestaren sinkronizazioa . . . . .	160
8.8.3. Etenaren sinkronizazioa . . . . .	160
8.8.4. Memoria Zuzeneko Sarbidea (DMA) . . . . .	161
8.8.5. I/O kanala . . . . .	162
8.9. Etenaren sinkronizazioa. . . . .	163
8.9.1. Kontzeptua . . . . .	163
8.9.2. Eten-mekanismoa . . . . .	164
8.9.3. Eten kontrolatzalea . . . . .	164
8.9.4. Etenaren kudeatzailea . . . . .	166
8.9.5. Eten motak . . . . .	166
8.9.6. Eten bektoreen taula . . . . .	167
8.9.7. Etenaldiaren arreta prozesua. . . . .	172
8.10. DMA memoriarako sarbide zuzena. . . . .	173
8.10.1. Funtzionalitatea. . . . .	174
8.10.2. Transferentziak. . . . .	174
8.10.3. Sinkronizazioa . . . . .	174
8.10.4. DMA kontrolagailuaren funtzionamendua . . . . .	174
8.10.5. Cache memorian koherentzia arazoak . . . . .	175
8.11. Autobusak . . . . .	175
8.11.1. DA. . . . .	175
8.11.2. PCI . . . . .	177
8.11.3. Ipar-Hego zubia . . . . .	178
8.11.4. x58 chipset-a . . . . .	179
8.12. Sarrera/irteera errutinak programatzea . . . . .	179
8.12.1. Sistema eragilearen software hierarkikoa . . . . .	179
8.12.2. Argibide multzoaArkitektura . . . . .	180
8.12.3. Eten programagarrien kontrolagailuen programazioa . . . . .	181
8.12.4. Teklatuaren kontrolatzalea . . . . .	182
8.12.5. ataka paraleloa. . . . .	182
8.12.6. RS-232 serieko komunikazioa . . . . .	183
8.13. Ariketak . . . . .	184
9. Memoria Unitatea. . . . .	185
9.1. Sarrera . . . . .	185
9.1.1. Ikastaroa. . . . .	185
9.1.2. Liburua: William Stalling . . . . .	185
9.1.3. Erref. . . . .	185
9.1.4. Historia . . . . .	185
9.1.5. Interesa . . . . .	185
9.1.6. Ikuspegiak. . . . .	186
9.1.7. Memoriaren hierarkia. . . . .	187
9.2. Erregistroak . . . . .	188
9.2.1. DA. . . . .	188
9.2.2. amd64 arkitektura. . . . .	188
9.3. Memoria nagusia (Dynamic RAM DRAM). . . . .	191
9.3.1. Erdieroaleen Memoria motak . . . . .	191
9.3.2. Erdieroaleen memoria NAGUSIA . . . . .	192
9.3.3. DRAM memoriaren antolaketa aurreratua . . . . .	210
9.3.4. Informazio osagarria . . . . .	215

9.3.5. Thinkpad T560. . . . .	215
9.4. Cache Memoria . . . . .	216
9.4.1. Bibliografia . . . . .	216
9.4.2. Sarrera . . . . .	216
9.4.3. Oinarrizko Printzipioak . . . . .	216
9.4.4. Cachearen diseinu-elementuak. . . . .	221
9.4.5. Politikak. . . . .	229
9.4.6. Optimizazio Adibidea . . . . .	230
9.4.7. Thinkpad L560. . . . .	230
9.5. Memoria Birtuala . . . . .	231
9.5.1. Sarrera . . . . .	231
9.5.2. Programatzairearen ikuspuntu eta CPU . . . . .	231
9.5.3. Memoria birtualeko helbide-espazioa . . . . .	231
9.5.4. Programa baten antolaketa logikoa makina-lengoaien: Atalak eta Segmentuak . . . . .	231
III Teoria Ariketak. . . . .	232
10. Ariketak . . . . .	233
10.1. Gutxieneko Ariketa Zerrenda . . . . .	233
10.1.1. I. zatia . . . . .	233
10.1.2. II. zatia . . . . .	233
10.2. von Neumann arkitektura . . . . .	233
10.2.1. Ordenagailuak: IAS, ENIAC, . . . . .	233
10.2.2. CPU-Memoria Interkonexioa . . . . .	237
10.3. Datuen irudikapena. . . . .	241
10.4. Eragiketa aritmetikoak . . . . .	245
10.5. Eragiketa logikoak . . . . .	247
10.6. Argibideen irudikapena . . . . .	248
10.7. asm programazioa . . . . .	262
10.7.1. Datuak. . . . .	262
10.7.2. Helbideratzeko moduak. . . . .	263
10.7.3. Aritmetika . . . . .	264
10.7.4. Jauziak . . . . .	266
10.7.5. Bada-Orduan-Bestela . . . . .	267
10.7.6. Do-While Loops . . . . .	268
10.7.7. Azpirutina deia . . . . .	270
10.8. C Programazio Lengoaia . . . . .	272
10.8.1. Erakusleak . . . . .	272
10.9. 4. kapitulua: Cache memoria . . . . .	273
10.10. 5. kapitulua: Memoria sinkrono dinamikoa RAM (SDRAM) . . . . .	279
10.11. 7. kapitulua: Sarrera/Irteera Sistemak . . . . .	281
10.12. 8. kapitulua: Sistema eragilea . . . . .	286
10.13. 12. kapitulua: prozesadorearen egitura eta funtzioa (9thEd-ko 14. kapitulua) . . . . .	293
10.14. 13. kapitulua: Instrukzio multzoa ordenagailua murrizten du (15. kapitulua 9thEd) . . . . .	300
IV Autoebaluazioaren Teoria . . . . .	308
11. Teoria: Galdetegia . . . . .	309
11.1. von Neumann arkitektura . . . . .	309
V Praktika-gidoiak: x86 muttaiaren-programazioa. . . . .	310
12. AT&T x86-32 Asanblea Lengoaiaren Programazioaren Sarrera . . . . .	311
12.1. Sarrera . . . . .	311

12.1.1. Helburuak . . . . .	311
12.1.2. Baldintzak . . . . .	311
12.2. IRAKUR NAIZ . . . . .	311
12.3. Gaiak . . . . .	312
12.4. Lanpostua . . . . .	312
12.5. sum1toN.c programazioa . . . . .	312
12.5.1. Algoritmoa . . . . .	312
12.5.2. Iturburu-moduluua Editatu: sum1toN.c . . . . .	312
12.5.3. Konpilazioa . . . . .	313
12.5.4. Moduluen analisia . . . . .	313
12.5.5. Exekuzioa . . . . .	313
12.5.6. Depurazioa . . . . .	314
12.5.7. Oroigarria: Memoria dokumentua . . . . .	316
12.5.8. Ariketa gehiagorekin jarraitzen dugu. . . . .	316
12.6. sum1toN.s programazioa . . . . .	317
12.6.1. Algoritmoa . . . . .	317
12.6.2. Iturburu-moduluaren edizioa: sum1toN.s . . . . .	317
12.6.3. Konpilazioa . . . . .	318
12.6.4. Exekuzioa . . . . .	318
12.6.5. Iturria moduluaren analisia . . . . .	318
12.6.6. Depurazioa . . . . .	318
12.7. amd64 arkitektura . . . . .	319
13. Datuen irudikapena . . . . .	320
13.1. Sarrera . . . . .	320
13.1.1. Helburuak . . . . .	320
13.1.2. Iturburu-moduluak: ezaugarriak. . . . .	320
13.1.3. Baldintzak . . . . .	320
13.2. IRAKUR NAIZ . . . . .	321
13.3. Aukerako galderak. . . . .	321
13.4. Barne CPU erregistroak . . . . .	321
13.5. Datuen tamaina eta aldagaiak. . . . .	321
13.5.1. Algoritmoa . . . . .	321
13.5.2. Iturburu-moduluaren edizioa: data_size.s . . . . .	321
13.5.3. Konpilazioa . . . . .	322
13.5.4. Exekuzioa . . . . .	322
13.5.5. Iturburu-moduluaren analisia . . . . .	322
13.5.6. GDB: Behaketak . . . . .	323
13.5.7. GDB: urratsez urrats exekuzioa . . . . .	323
13.6. Operandoaren tamaina. . . . .	326
13.6.1. Iturburu-moduluaren edizioa: data_atzizkiak.s . . . . .	326
13.6.2. Konpilazioa . . . . .	327
13.6.3. Exekuzioa . . . . .	328
13.6.4. asm iturburu-moduluaren analisia . . . . .	328
13.6.5. Eragaiaren tamaina asm instrukzio batean ondorioztatzea . . . . .	328
13.6.6. GDB: urratsez urrats exekuzioa . . . . .	328
13.7. Helbideratzeko moduak . . . . .	329
13.7.1. Iturburu-moduluaren edizioa: data_addressing.s . . . . .	329
13.7.2. Konpilazioa . . . . .	331

13.7.3. Exekuzioa . . . . .	331
13.7.4. asm iturburu-moduluaren analisia . . . . .	331
13.7.5. GDB: Urratsez urratseko exekuzioa . . . . .	331
14. Eragiketa aritmetikoak eta logikoak . . . . .	333
14.1. Sarrera . . . . .	333
14.1.1. Helbuuak . . . . .	333
14.1.2. Arkitektura kontzeptuak. . . . .	333
14.1.3. Iturburu moduluak . . . . .	333
14.1.4. Baldintzak. . . . .	333
14.2. IRAKUR NAIZ . . . . .	333
14.3. Gaiak . . . . .	333
14.4. Barne CPU erregistroak. . . . .	334
14.5. Eragiketa aritmetikoak eta logikoak Zenbaki osoekin . . . . .	334
14.5.1. Editatu iturburu-modulu: op_arit_log.s . . . . .	334
14.5.2. Konpilazioa . . . . .	336
14.5.3. Exekuzioa . . . . .	336
14.5.4. Iturburu-moduluaren analisia . . . . .	336
14.5.5. Urratsez urrats exekuzioa . . . . .	336
15. Baldintzapeko jauzien jarraibideak. . . . .	338
15.1. Sarrera . . . . .	338
15.1.1. Helbuuak . . . . .	338
15.1.2. Baldintzak. . . . .	338
15.2. IRAKUR NAIZ . . . . .	338
15.3. Gaiak . . . . .	338
15.4. Baldintzazko jauziak . . . . .	338
15.4.1. Algoritmoa . . . . .	338
15.4.2. Sorburuaren edizioa Modulu: jauziak.s.. . . . .	338
15.4.3. Konpilazioa . . . . .	341
15.4.4. Exekuzioa . . . . .	341
15.4.5. Iturburu-moduluaren analisia . . . . .	342
15.4.6. Urratsez urrats exekuzioa . . . . .	342
15.5. Mnemoteknia erabilia . . . . .	343
16. Sistema eragilera (kernel) deiak . . . . .	344
16.1. Sarrera . . . . .	344
16.1.1. Zer dira sistema-deiak ? . . . . .	344
16.1.2. Sistema deien eskuliburuak . . . . .	344
16.1.3. Dei-kodeak. . . . .	344
16.1.4. Nola pasa argumentuak zuzenean Kernelera . . . . .	344
16.1.5. Nola pasa argumentuak zeharka libc funtzioen bidez . . . . .	3. 4. 5
16.2. IRAKUR NAIZ . . . . .	3. 4. 5
16.3. Gaiak . . . . .	3. 4. 5
16.4. Irten deitik . . . . .	3. 4. 5
16.4.1. Sorburua editatzen Modulu:salida.c/salida.s . . . . .	3. 4. 5
16.5. Deitu C liburutegira muntaketa-kodetik . . 16.5.1. inprimatu.s: inprimatuf . . . . .	346
16.6. Sistema-deiak AMD64 arkitekturen. . . . .	347
17. Azpirrutinak . . . . .	349
17.1. Sarrera . . . . .	349

17.1.1. Golak . . . . .	349
17.2. Iturria Modulua. . . . .	349
17.3. Baldintzak . . . . .	349
17.4. IRAKUR NAIZ. . . . .	349
17.5. Gaiak. . . . .	350
17.6. Datuen tamaina eta aldagaiak. . . . .	350
17.6.1. Algoritmoa. . . . .	350
17.6.2. Iturburu-modulua Editatu: sumMtoN.s . . . . .	350
17.6.3. Konpilazioa. . . . .	352
17.6.4. Exekuzioa . . . . .	352
17.6.5. Iturburu-moduluaren analisia. . . . .	352
18. Irudiak: Bit Map eramangarria. . . . .	354
18.1. Sarrera . . . . .	354
18.2. Aplikazio. . . . .	354
18.2.1. Fitxategiak barne. . . . .	354
18.2.2. Konpilazio automatikoaren adibidea. . . . .	354
18.3. BMP formatua. . . . .	355
18.3.1. Pantaila: pixela. . . . .	355
18.3.2. Kodetzea. . . . .	355
18.3.3. Memoria mapa: Monitorea ý Buffer. . . . .	355
18.3.4. BMP fitxategia. . . . .	356
18.4. Moduluaren iturria bitmap_gen_test.c . . . . .	356
18.4.1. Deskribapena . . . . .	356
18.4.2. Programaren funtzioka. . . . .	356
18.5. Ariketak . . . . .	358
18.5.1. Programazioa C-n. . . . .	358
18.5.2. Programazioa ASMn. . . . .	358
18.5.3. GDB. . . . .	358
VI Erreferentzia Azkar Orriak. . . . .	362
19. AT&T x86 muntatzaileen programazioa. . . . .	363
19.1. x86-32 Programak. . . . .	363
19.1.1. Programa Minimalista. . . . .	363
19.1.2. Oinarrizko Adibidea. . . . .	364
19.2. Muntatzaileen AS Zuzentaraauak. . . . .	366
19.3. Muntatzeko jarraibideen erreperitorioa. . . . .	367
19.3.1. TRANSFERENTZIA. . . . .	367
19.3.2. ARITMETIKA. . . . .	368
19.3.3. LOGIKOA. . . . .	369
19.3.4. DENBERAK. . . . .	370
19.3.5. JAUZIAK (orokorra) . . . . .	370
19.3.6. JAUZTUAK Zeinurik gabe (Kardinal) JAUZTIK Signuarekin (Osokoa) . . . . .	370
19.3.7. BANDERAK (ODITSZAPC) . . . . .	371
19.3.8. Mnemoteknikoaren zerrenda. . . . .	372
19.4. Erregistroak. . . . .	374
19.4.1. Ikuksen osoa. . . . .	374
19.4.2. Programatzaileak ikusgai dauden erregistroak. . . . .	375
19.4.3. Bateragarritasuna 32-64. . . . .	376
19.4.4. Kontrol-bandera-erregistroa. . . . .	376

19.5. GDB . . . . .	379
VII Autoebaluazio Praktikak . . . . .	381
20. Praktikak: Galdetegia . . . . .	382
20.1. 1. praktika: AT&T x86-32 Mihiztadura Lengoaiaren Programazioari buruzko Sarrera. . . . .	382
20.1.1. Gai teorikoak. . . . .	382
20.1.2. Gai praktikoak. . . . .	382
20.2. 2. praktika: Datuen irudikapena . . . . .	383
20.2.1. Moduluaren datuak_tamaina.s . . . . .	383
20.2.2. Moduluaren datuak_atzizkiak.s . . . . .	383
20.2.3. Moduluaren helbideratze_datuak.s . . . . .	383
20.3. 3. praktika: Eragiketa aritmetiko-logikoak eta baldintzapeko jauzi-argibideak . . . . .	383
20.3.1. op_arit_log.s modulua. . . . .	383
20.3.2. Modulu jauziak.s . . . . .	384
20.4. 4. praktika: Sistema Eragilerako deiak . . . . .	384
20.4.1. syscall_write_puts.c modulua . . . . .	384
20.4.2. syscall_write_puts.s modulua . . . . .	386
20.5. 5. praktika: Azpirutina baterako deiak . . . . .	386
20.5.1. sumMtoN_aviso.c modulua . . . . .	386
20.5.2. sumMtoN_aviso.s modulua . . . . .	387
20.6. 6. praktika: Bit mapa eramangarriaren irudia . . . . .	387
20.6.1. Programazioa C-n . . . . .	387
20.6.2. Programazioa ASMn . . . . .	387
20.6.3. GDB . . . . .	387
VIII Eranskinak . . . . .	388
21. Konputagailuen Arkitektura . . . . .	389
21.1. Konputagailuen Egitura . . . . .	389
21.1.1. Testuingurua . . . . .	389
21.1.2. HW arkitektura . . . . .	389
21.1.3. CPU. . . . .	389
21.1.4. Memoria . . . . .	391
21.2. Instruction Set Architecture (ISA) . . . . .	393
21.2.1. Adibideak: Intel x86, Motorola 68000, MIPS, ARM. . . . .	394
21.3. x86 arkitektura duten Intel prozesadoreak . . . . .	394
21.3.1. Nomenklatura. . . . .	394
22. RTL Erregistroa Transferitzeko Hizkuntza . . . . .	396
22.1. RTL hizkuntza . . . . .	396
22.1.1. Sarrera . . . . .	396
22.1.2. Registroak . . . . .	396
22.1.3. Sinboloak. . . . .	398
22.1.4. RTL esaldiak. . . . .	398
22.1.5. RTL adibideak adierazpen aritmetiko-logikoekin . . . . .	399
23. IASSim muntatzaile-programak . . . . .	400
23.1. 1. adibidea: sum1toN.ias . . . . .	400
23.1.1. 2. adibidea: Produktua/Koientzia . . . . .	401
23.1.2. 3. adibidea: Bektoreak . . . . .	405
24. IASSim simulagailua. . . . .	410
24.1. Java JVM makina birtuala. . . . .	410
24.2. IAS simulagailua . . . . .	410

24.3. Simulazioa/Arazketa . . . . .	411
25. Goi-mailako eta behe-mailako programazio-lengoaiak . . . . .	413
25.1. Maila handiko programazio-lengoaiak eta maila baxuko hizkuntzak . . . . .	413
25.2. Sum1toN adibidea programazio hizkuntza ezberdinetan. . . . .	413
26. Mihiztadura-programazio-lengoaiak. . . . .	418
26.1. Erreferentzia eskuliburuak . . . . .	418
26.1.1. Intel hizkuntza . . . . .	418
26.1.2. AT&T hizkuntza. . . . .	418
26.1.3. i386 arkitekturaren ezaugarriak . . . . .	418
26.1.4. Muntatzailea: Zuzentaraauak . . . . .	418
26.1.5. Eztabaida zergatik ASM AT&T. . . . .	418
26.1.6. TRANSFERENTZIA . . . . .	418
26.1.7. ARITMETIKA . . . . .	419
26.1.8. LOGIKOA. . . . .	421
26.1.9. DENBERAK . . . . .	421
26.1.10. JAUZIAK (orokorra). . . . .	421
26.1.11. JAUZTUAK Zeinurik gabe (Kardinal) JAUZTIK Signuarekin (Osokoa) . . . . .	422
26.1.12. BANDERAK (ODITSZAPC) . . . . .	422
26.1.13. Atzizkiak. . . . .	422
26.2. Intel x86-32 / i386. . . . .	423
26.2.1. batuketa1toN.s . . . . .	423
26.2.2. kaixo_mundua.s. . . . .	423
26.2.3. hello_world: Makina Kode Binarioa . . . . .	424
26.3. Intel x86-64 / AMD 64. . . . .	427
26.3.1. batuketa1toN.s . . . . .	427
26.3.2. Kaixo Mundua . . . . .	428
26.3.3. Denetarikoa . . . . .	429
26.3.4. sum1toN.s: intel language . . . . .	430
26.4. ARM arkitektura . . . . .	431
26.4.1. Kaixo Mundua . . . . .	431
26.4.2. DA. . . . .	431
26.5. Motorola 68000 . . . . .	432
26.5.1. Kaixo Mundua . . . . .	432
26.5.2. DA. . . . .	432
26.6. MIPS arkitektura. . . . .	433
26.6.1. DA. . . . .	433
27. Tresna-katea: Tresna-katea konpilazio-prozesuan. . . . .	435
27.1. Tresna-katea . . . . .	435
27.2. C hizkuntzan programa baten konpilazio prozesua . . . . .	435
27.3. Muntaketa prozesuaren itzultzzaileak . . . . .	435
27.4. "Aso" muntatzailea . . . . .	436
27.4.1. Zuzentaraauak . . . . .	436
28. Programazioa hasieratik landu . . . . .	437
28.1. Dokumentazioa: gidoiak, bibliografia, oharrak. . . . .	437
28.2. Garapen Plataforma . . . . .	437
28.2.1. Tresnak . . . . .	437
28.2.2. Sareko programazioa . . . . .	438
28.2.3. Erreferentziak . . . . .	438

28.3. Txostenaren dokumentua: edukia eta formatua . . . . .	438
28.3.1. Edukia . . . . .	438
28.3.2. Formatua . . . . .	439
28.3.3. Memoria Dokumentua entregatzea . . . . .	439
28.4. Ebaluazioa . . . . .	439
28.5. Programazioa . . . . .	440
28.5.1. Metodologia . . . . .	440
28.6. Konpilazioa . . . . .	440
28.6.1. Iturburu-modulua C hizkuntzan . . . . .	440
28.6.2. Programaren sarrera-puntuak: main vs _start . . . . .	440
28.6.3. Konpilazio faseak . . . . .	441
28.6.4. Tresna-katea . . . . .	442
28.6.5. iturburu-modulua mutua hizkuntzan . . . . .	442
28.7. Depurazioa. . . . .	442
28.8. Ohiko akatsak . . . . .	443
28.8.1. gcc. . . . .	443
28.8.2. gdb . . . . .	443
28.9. Programa eta Araztu hutsetik . . . . .	443
28.9.1. Hasteko: ASM . . . . .	443
28.9.2. Hasteko: C . . . . .	446
29. Sistema eragilerako deiak . . . . .	448
29.1. Sarrera . . . . .	448
29.2. Deitu eskuliburuak . . . . .	449
29.3. ZEHARKAKO deia . . . . .	449
29.4. ZUZENEKO DEIA . . . . .	449
29.4.1. Dei zuzeneko argudioak. . . . .	449
29.4.2. Zuzeneko dei-kodeak . . . . .	450
29.5. Adibideak: C hizkuntza . . . . .	450
29.6. Adibideak: ZEHARKAKO ASM . . . . .	450
29.7. Adibideak: ZUZENEKO ASM . . . . .	451
29.8. Komando lerroa . . . . .	451
29.8.1. Procedura . . . . .	451
29.8.2. StackInitialization . . . . .	452
29.8.3. Errutina nagusia Itzuliarekin . . . . .	453
29.8.4. Ariketak: sum_linea_com.s, maximum_linea_com.s . . . . .	454
30. Pila . . . . .	457
30.1. Kontzeptua . . . . .	457
30.2. Zabalera . . . . .	457
30.3. Markoa: marko-erakuslea eta pila-erakuslea . . . . .	458
30.4. Push-Pop muntatzailearen jarraibideak . . . . .	459
30.4.1. Dei habiara . . . . .	460
31. Batzar Hizkuntza Programak: Proposamenak . . . . .	461
31.1. Maila Ertaina . . . . .	461
31.1.1. BMP formatua. . . . .	461
32. Eranskina: CPU Unitatea . . . . .	470
32.1. Mikroarkitektura Modernoak (2000. urtetik aurrera) . . . . .	470
32.2. Intel . . . . .	470
32.3. AMD . . . . .	470

32.4. Skylake-U . . . . .	470
32.5. ARM Cortex-A76/Cortex-A55 . . . . .	472
32.5.1. Sarrera . . . . .	472
32.5.2. Huawei P30 Pro telefonoa . . . . .	472
32.5.3. ISA Cortex-A76/Cortex-A55. . . . .	473
32.5.4. Nukleo anitzekoa. . . . .	473
32.5.5. Cortex-A76 Mikroarkitektura . . . . .	473
32.5.6. Cortex-A55 Mikroarkitektura . . . . .	473
33. Eranskina: DRAM Memoria Unitatea . . . . .	475
33.1. Memoria nagusia: DRAM . . . . .	475
33.1.1. Irakurketa/idazketa eragiketaren denbora . . . . . 33.1.2. latentzia denborak . . . . .	475
33.1.3. Adibidea PC2-6400 (DDR2-800) 5-5-5-16 . . . . .	476
33.2. DRAM memoriaren antolaketa aurreratura . . . . .	477
33.2.1. DRAM asinkronoa . . . . .	477
33.2.2. SDRAM (DRAM sinkronikoa) . . . . .	477
34. Eranskina: Memoria Birtuala. . . . .	479
34.1. Bibliografia . . . . .	479
34.2. Sistema eragileak: Memoriaren kudeaketa . . . . .	479
34.2.1. Prozesu anitzeko sistemak . . . . .	479
34.2.2. Memoria fisikoaren kudeaketa . . . . .	479
34.2.3. Memoria Birtualaren bidezko kudeaketa . . . . .	480
34.3. Memoria birtuala segmentatua . . . . .	482
34.3.1. Segmentazioaren interpretazioa . . . . .	482
34.3.2. Atalak. . . . .	482
34.3.3. Atalen Esteka. . . . .	483
34.3.4. Segmentu logikoak . . . . .	484
34.3.5. Intel memoria bilakaera 8086-80286 . . . . .	486
34.4. Orrikako Memoria Birtuala . . . . .	489
34.4.1. Fundazioa . . . . .	489
34.4.2. Memoria birtual orrikatuaren kontzeptua. . . . .	490
34.4.3. Zatiketa. . . . .	490
34.4.4. MMU . . . . .	490
34.4.5. Memoria birtuala cachean gordeta . . . . .	490
34.4.6. Orrialde taula. . . . .	491
34.4.7. Maila anitzeko orrialdea . . . . .	492
34.4.8. Intel: Memoria birtualaren bilakaera . . . . .	493
34.4.9. Glosarioa . . . . .	494
34.4.10. Itzulpena: helbide birtuala fisikoa. . . . .	495
34.4.11. Itzulpen Lookaside Buffer. . . . .	496
34.4.12. Ariketa . . . . .	497
34.4.13. Intel Core i7 . . . . .	501
34.5. Sistema eragileak: Memoriaren kudeaketa . . . . .	505
34.5.1. Babesa . . . . .	505
34.5.2. Eskaeraren arabera orria. . . . .	505
34.5.3. Ordezkoa . . . . .	505
34.5.4. VMTool . . . . .	505
35. C Programazio Lengoaia . . . . .	506

35.1. Sarrera . . . . .	506
35.2. Galdaketa . . . . .	506
35.2.1. Kontzeptua . . . . .	506
35.2.2. Adibidea. . . . .	506
35.3. Erakuslea . . . . .	506
35.3.1. Erreferentziak . . . . .	506
35.3.2. Sarrera . . . . .	506
35.3.3. Kontzeptua . . . . .	507
35.3.4. Modulu Ilustratzalea . . . . .	511
35.3.5. Adierazpena . . . . .	512
35.3.6. Operadorearen helbidea. . . . .	512
35.3.7. Bideratze- edo deserreferentzia-eragilea . . . . .	513
35.3.8. Adibidea. . . . .	513
35.3.9. Erakusleen aplikazioak . . . . .	513
35.3.10. Katea literala. . . . .	514
35.3.11. Erakuslea Erakuslea . . . . .	515
35.3.12. StringVariable. . . . .	515
35.3.13. Ezaugarriak. . . . .	516
36. x87 FPU . . . . .	517
36.1. x87 FPU . . . . .	517
36.1.1. Laburpena . . . . .	517
36.1.2. Erref. . . . .	518
37. Konputagailuen Egitura 2022: Lehen Teoria Partziala . . . . .	519
37.1. 1-6 gaietarako ariketak. . . . .	519
37.2. Mihiztadura Lengoaiaren Programazioa . . . . .	524
38. Konputagailuen Egitura 2022: Lehen Praktika Partzialak. . . . .	525
38.1. Sarrera . . . . .	525
38.2. Iturburu-modulua . . . . .	525
38.3. Azterketa metodologia . . . . .	526
38.4. Ebaluazioa . . . . .	527
39. Konputagailuen Egitura 2022: Bigarren Praktika Partzialak. . . . .	528
39.1. Sarrera . . . . .	528
39.2. Azterketa metodologia . . . . .	528
39.3. "Lerroak" azterketa. . . . .	529
40. Azterketa-espedienteen izendapena. . . . .	531
41. Aurreko Ikastaroetako azterketak. . . . .	533
41.1. 2018 urtea . . . . .	533
41.1.1. azaroa . . . . .	533
41.2. 2017 urtea . . . . .	535
42. Myaulary: Bideokonferentzia . . . . .	544
42.1. Sarrera . . . . .	544
42.2. Zoomaren instalazioa . . . . .	544
42.3. Zoom erabiltzailearen gida . . . . .	544
42.3.1. Ezarpena. . . . .	544
42.4. Bideokonferentzia saioa. . . . .	544
IX Bibliografia . . . . .	545
X Glosarioa . . . . .	547
XI Kolofoia . . . . .	548



I Instruction Repertoire Architecture (ISA): von Neumann  
ordenagailua, datuak, argibideak, programazioa.

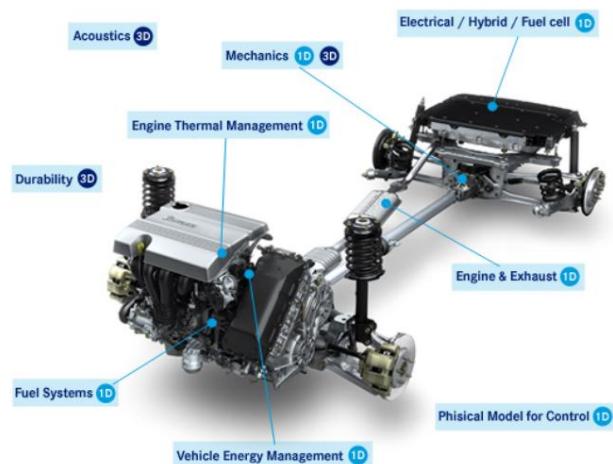
# 1. Kapitula. Konputagailuen Egituraren Sarrera

## 1.1. Sarrera

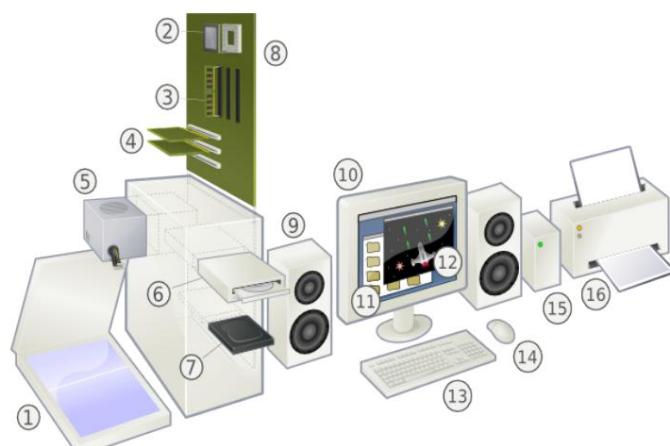
- Nafarroako Unibertsitate Publikoko Informatika Ingeniaritza Graduko Konputagailuen Egitura (240306) irakasgaiaren helburua konputagailuen arkitekturarako hastapeneko unibertsitate-ikastaroa izatea da, bere oinarrizko osagaia (prozesadorea, memoria eta sarrera/irteera modulua) aztertzen dituena. maila baxuko programazioa x86 mihiadura lengoian, konpilatzalea, arazketa, etab. softwarea garatzeko tresnen bitartez.

## 1.2. Makina baten arkitektura

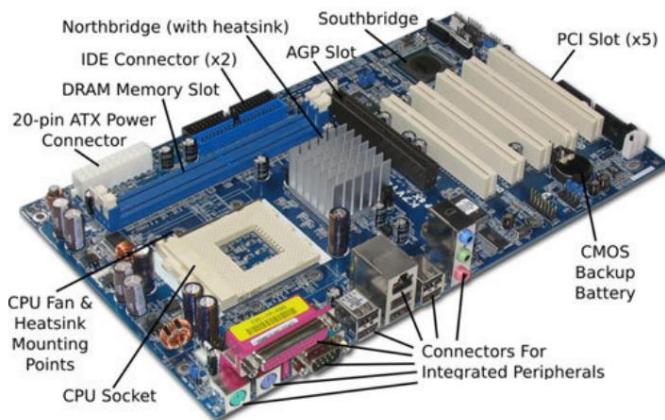
- Arkitektura: Antolaketa, Egitura: Zer, Nola, Ezarpena (teknologia)



1. irudia. Automobilaren egitura



2. Irudia Ordenagailu pertsonala



3. Irudia. Plaka

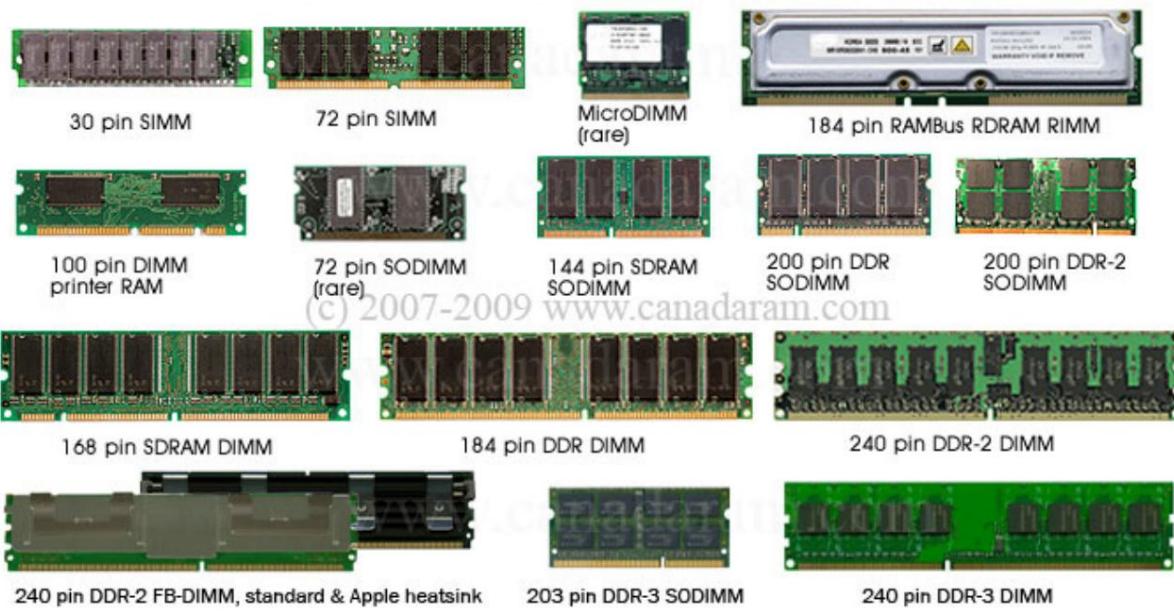
### 1.3. Arkitektura HW ikuspegitik

- 4 Oinarrizko Modulu: CPU-MEMORIA-CONTROLADORA Sarrera/Irteera [Periferikoak]-BUSES

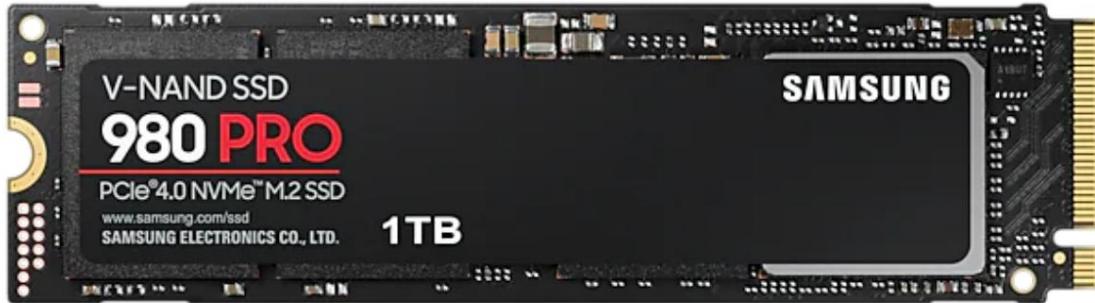


4. irudia. Intel Core i7 4. belaunaldiko CPU

Note, as well as the different number of pins, the different spacing of the slots in the connector-edge



5. Irudia DRAM memoria



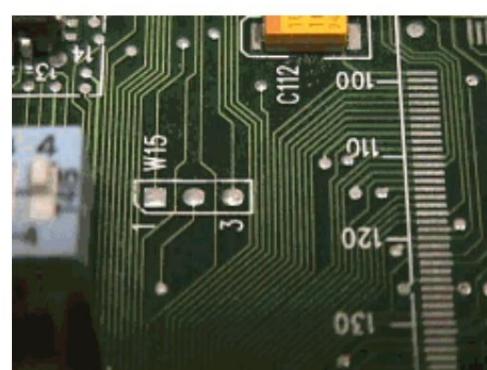
6. Irudia. Periferikoak: Semiconductor Solid State Drive (SSD) memoria



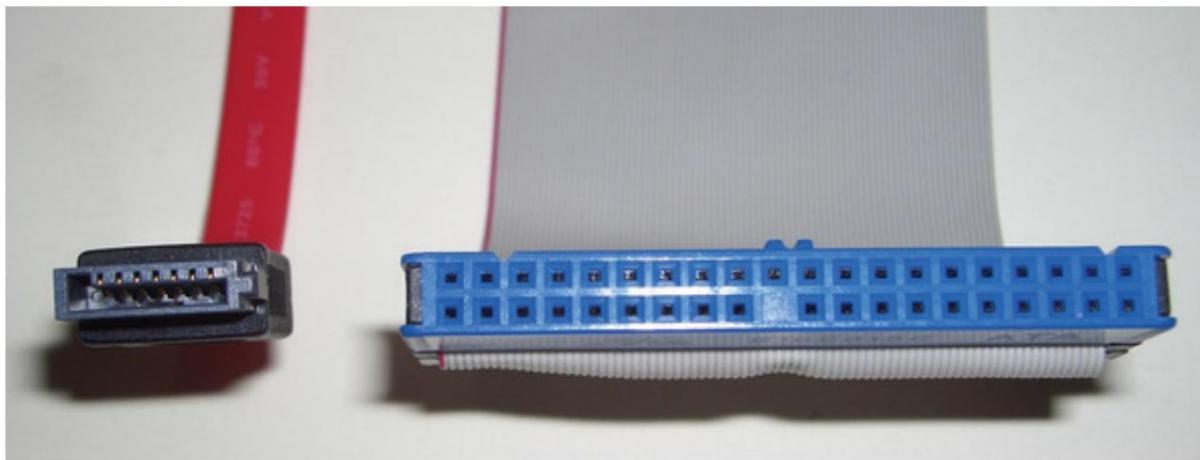
7. Irudia. Periferikoak: Seagate Disko gogorra



8. Irudia. Periferikoak: M2M Memoria



9. Irudia Plaka-Busa



## 10. Irudia Busen kableatzea

## 1.4. Programazio Lengoaiak eta Makina Lengoaia



## 11. Irudia Makina Lengoia: Bitarra

- Pascal hizkuntza

```
programaHello_world;  
    hasi  
idazten('kaixo mundua')  
    amaiera.
```

- L. Makina-Binario Intel x86

```

ÿ 4001c6: 10111111 00000000 00000000 00000000 00000000
ÿ 4001cb: 11101000 01111000 10010110 00000001 00000000
ÿ 4001d0: 11101000 00010011 00111101 00000001 00000000
ÿ 4001d5: 01001000 10001001 11011111
ÿ 4001d8: 11101000 01110011 10010101 00000001 00000000
ÿ 4001dd: 11101000 00000110 00111101 00000001 00000000
ÿ 4001e2: 11101000 10011001 01000010 00000001 00000000
ÿ 4001e7: 01001000 10001011 01011101 11111000
ÿ 4001eb: 10101001
ÿ 4001ec: 10010011

```

- Makina-lengoaia (Kode hamaseimala) vs. ASM x86 Mihiztadura-lengoaia

```

# [2] hasten da
0000000004001a0 <PASCALMAIN>:
ÿ 4001a0: 55          bultzatu %rbp
ÿ 4001a1: 48 89 e5    mov    %rsp,%rbp
ÿ 4001a4: 48 83 eq 10ÿ 4001a8:    azpi   $0x10,%rsp
48 89 5d f8 4001ac: e8 b7 3e    mov    %rbx,-0x8(%rbp)
01 00 # [3] writeln('kaixo mundua')    callq 414068 <FPC_INITIALIZEUNITS>

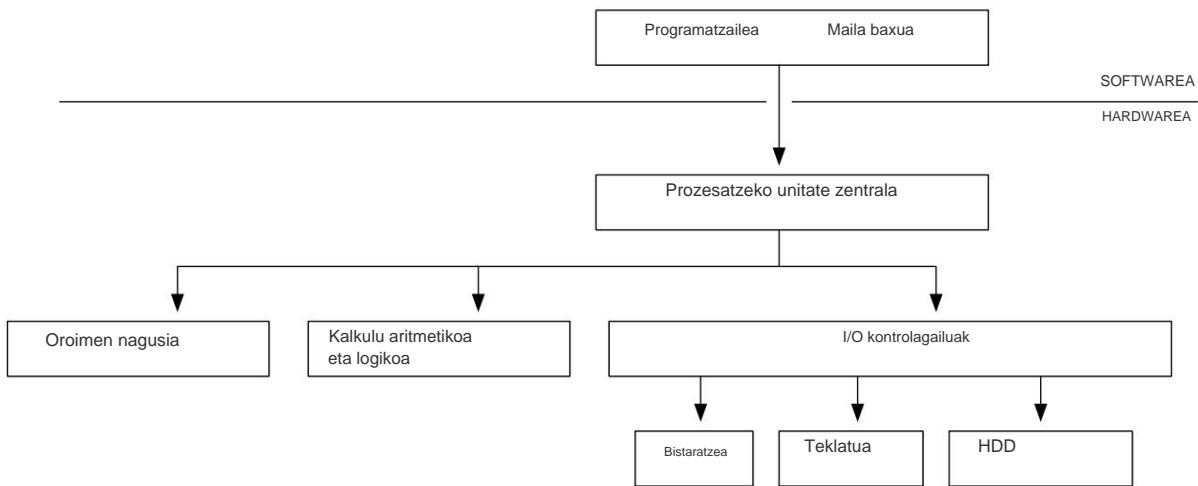
ÿ 4001b1: e8 ca 93 01 00ÿ 4001b6:    callq 419580 <fpc_get_output>
48 89 c3 4001b9: 48 89    mov    %rax,%rbx
                                mov    %rbx,%rsi
ÿ 4001bc: 48 ba c0 f6 61 00 00    movabs $0x61f6c0,%rdx
ÿ 4001c3: 00 00 00
ÿ 4001c6: bf 00 00 00 00ÿ 4001cb:    mov    $0x0,%edi
e8 78 96 01 00    callq 419848 <FPC_WRITE_TEXT_SHORTSTR>
ÿ 4001d0: e8 13 3d 01 00ÿ 4001d5:    callq 413ee8 <FPC_IOCHECK>
48 89 df 4001d8: e8 73 95    mov    %rbx,%rdi
01 00ÿ 4001dd: e8 06 3d 01 00    callq 419750 <fpc writeln_end>
                                callq 413ee8 <FPC_IOCHECK>
ÿ 4001e2: e8 99 42 01 00ÿ 4001e7:    callq 414480 <FPC_DO_EXIT>
48 8b 5d f8 4001eb: c9 4001ec:    mov    -0x8(%rbp),%rbx
c3                  utziq
                                retq

```

- Giza programazio lengoaia (testua) vs Makina programazio lengoaia (bitarra)
- Itzuli ÿ Konpilatu
- Interpretatu
- Programazio-lengoaiak: javascript, ruby, python, java, bash, C, assembly
- ÿ Abstrakzio mailak: gizakia/makina

## 1.5. Software/Hardware Interfazea

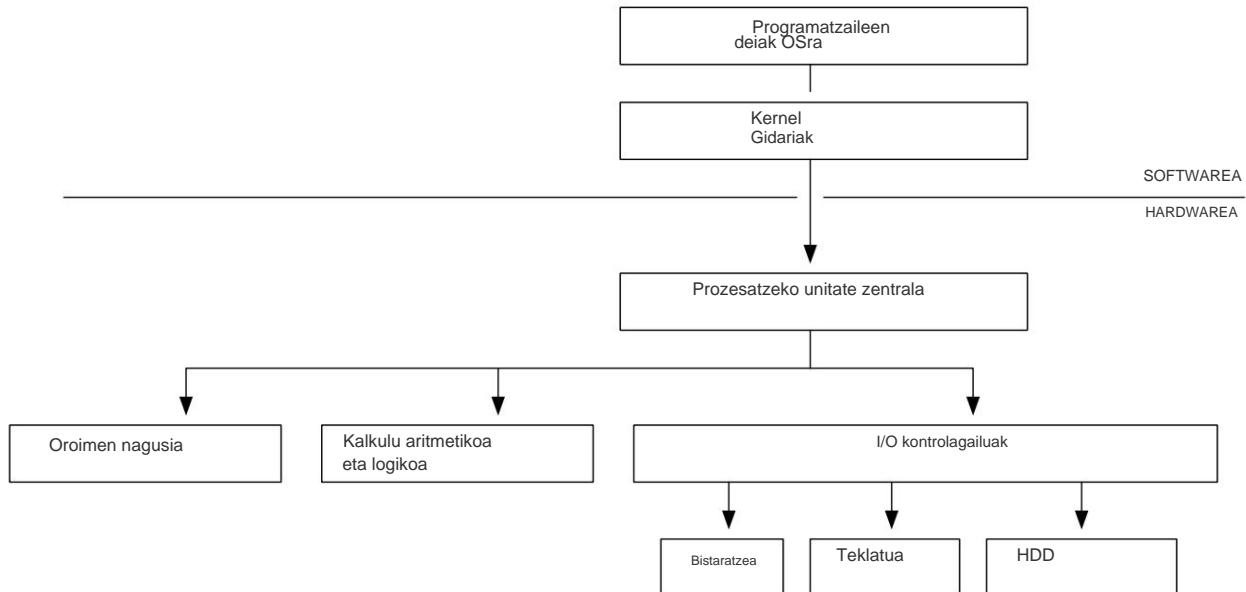
- Programatzaileak Makinaren hardwarearekin zuzeneko interakzioa ÿ Programazioa kostu baxuko hizkuntzan maila (muntatzailea, C)



- Maila baxuko programazioa
  - ÿ Iturburu-modulua: C lengoia edo Mihiztadura lengoia
  - ÿ Objektu-modulua: Makina-lengoia ÿ Iturburu-moduluaren konpilazioa, muntaketa eta lotura.
  - ÿ Bare Metal Sistemak: Ez dago Sistema Eragilerik. Hardwarean zuzenean programatzen dugu.

- Programatzaileak makinaren hardwarearekin duen zeharkako elkarrekintza. Programatzaileak sistema eragilearekin (OS) elkarreragin egiten du

ÿ Sistema eragilearen (OS) Kernelaren (kernelaren, adibidez, Linux) interakzioa makinarekin (adibidez, Intel x86)



- Sistema eragilea duten sistemak (adib. GNU/linux): Programazioan sistema eragilearen bidez hardwarera sartzen diren liburutegiak erabiltzen ditugu. Adibidez, printf() funtzioa, libc liburutegiko, konpilatzean sistema eragilearen write() funtziora itzultzen da, zeinak pantailaren txartel grafikoaren kontrolatzalea (sistema eragilean) deitzen duena. Sistema Eragileak makinaren HW fisikoa "abstraitza" lortzen du eta asko errazten du programazioa, ordenagailuaren funtzionamendu fisikoa ezagutu beharrik ez izateagatik.

## 1.6. Oharrak

- Bost Praktika-gidoiak oharretan daude eskuragarri: PDF dokumentuan eta  
On Line Notes dokumentua, azken hau da eguneratuena

[qr eecc] | ../images/introduction/qr\_eecc.jpg

12. Irudia. Sareko Oharrak

## 1.7. Ikastaroa

- Ordenagailuen Web Egitura

### Ikastaroa

#### 1. Sarrera

##### 2 - Von Neumann Arkitektura

2.1 CPU

2.2 Memoria

2.3 Sarrera/Irteera

##### 3 - Datuen irudikapena

3.1 Bit, Byte eta Word

3.2 Karaktereak, osoak eta errealkak

##### 4 - Aritmetika eta logika

4.1 Eragiketa aritmetikoak eta logikoak zenbaki osoen gainean bitarrean

4.2 Biribilketa eta erroreen hedapena zenbaki errealetan

##### 5 - Argibideen irudikapena

5.1 Makina-lengoaia, mutua-lengoaia eta goi-mailako lengoaiak

5.2 Argibide-formatua

5.3 Argibide motak eta helbideratze moduak

##### 6 - Mihiztatzale-lengoaien programazio-lengoaiaren oinarrizko eraikuntzen programazioa maila altua

6.1 Esleipenen adierazpenak

6.2 Baldintzazko adierazpenak

6.3 Begiztak

6.4 Funtzio edo azpierrutinen deiak eta itzulketak

##### 7 - PUZaren arkitektura eta antolaketa

7.1 Argibide multzoa

7.2 CISC, RISC eta VLIW arkitekturak

7.3 Argibideen exekuzio faseak

7.4 Datuen bidea

##### 8 - Sarrera/irteera sistema

8.1 Inkestaren araberako sinkronizazioa

8.2 Eten sinkronizazioa

8.3 Eten bektorea

8.4 DMA memoriarako sarbide zuzena

8.5 Sarrera/irteera errutinen mutua-lengoaiaren programazioa

##### 9 - Memoriaren antolaketa

9.1 Memoriaren hierarkia

9.2 Latentzia eta banda-zabalera

9.3 Cache memoria

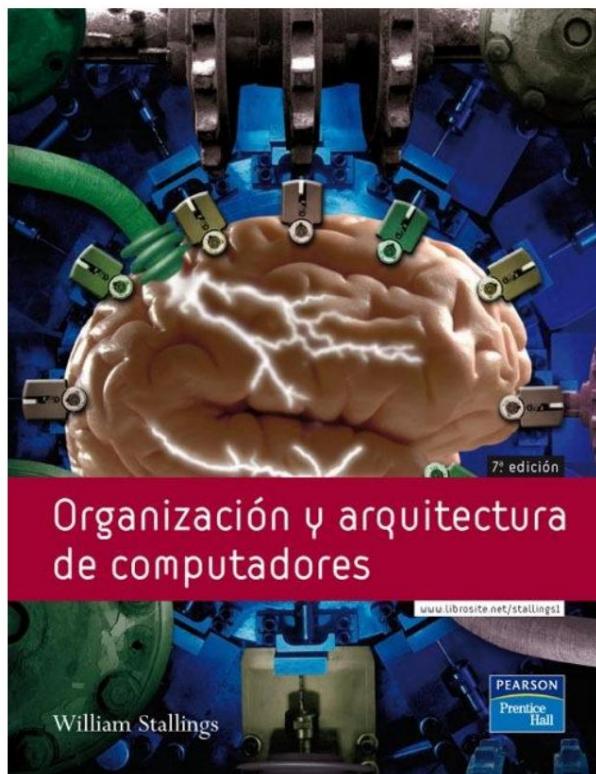
## 9.4 Memoria birtuala

### 1.7.1. Oinarrizko Bibliografia

#### Inprimaketa

- Teoria

↳ William Stallings

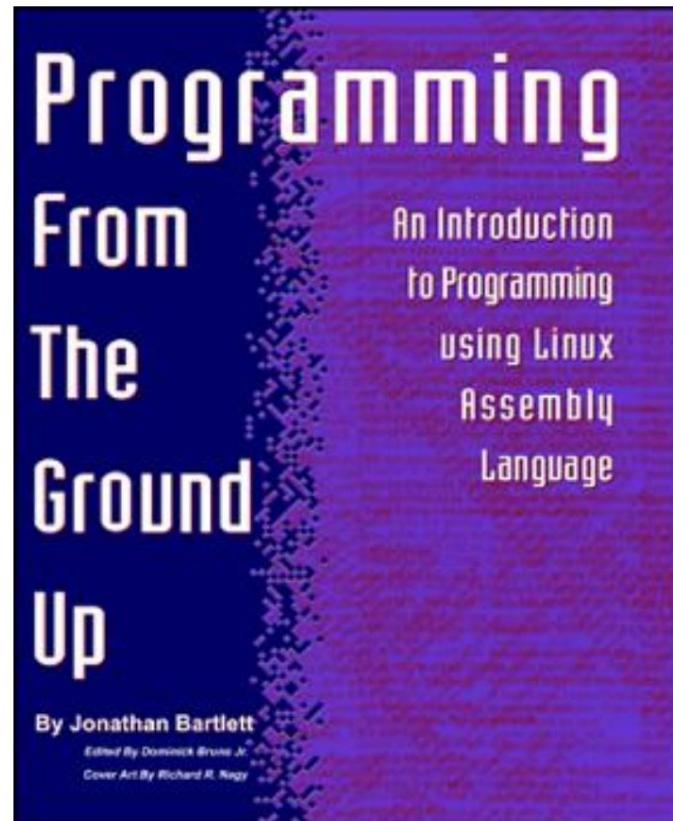


13. Irudia William Stallings Book

Computer Organization and Architecture. William Stallings 7. edizioa, Pearson  
Prentice Hall-en berrargitalpena ISBN 8489660824,  
9788489660823. 2006

Ordenagailuen antolaketa eta arkitektura: errendimendurako diseinua.  
William Stallings 9th  
Ed Upper Saddle River (NJ): Prentice Hall, [2013]  
ISBN 0-273-76919-7. 2012

- COA 7.a
- 2019ko 11. edizioa
- Praktikak:



14. Irudia Programazio Muntaia

Programazioa oinarritik Jonathan Bartlett-

ek zuzendua Dominick Bruno, Jr.

Copyright © 2003 Jonathan Bartlett ISBN 0-9752838-4-7

Bartlett argitaletxeak

argitaratua Broken Arrow, Oklahoma-n

- Programazioa Ground Up Jonathan Bartlett-en eskutik. Programazioa AT&T Asanblea Lengoainean x86 arkitekturarako.

ÿ PGU liburua sarean

ÿ PGU book home

ÿ pdf

- Muntatzailea... baina oso erraza da: IA-32 (i386) (AT&T sintaxia)

ÿ GNU eskuliburuak:

- Eskuliburuen dokumentazioa: GNU eta Linux fundazioaren aplikazio eta software tresna gehienek Eskuliburuak dituzte linean edo lokalean makinan bertan. gcc konpilatzailearen eskuliburua: **\$man gcc**



Informatika-profesionalak eskuratu beharreko trebetasuna da eskuliburu hauek sartzeko eta erabiltzeko, baita lan saioetan sarbide azkarreko erreferentzia-orriak edukitzeko ere.

#### On-line

- <https://diveintosystems.org/book/preface.html>

### 1.7.2. Irakurketa gehiago

- Joan bibliografiako [kapitulura](#).

## 1.8. Fakultatea

- Cándido Aramburu Mayo.

ÿ Telekomunikazio Ingeniaritzan doktorea. Unibertsitateko katedraduna.

ÿ Los Tejos eraikina, 2. solairua. 2028ko bidalketa.

ÿ barneko posta elektronikoa: Miaulario zerbitzariaren bidez.

ÿ [PDI Irakasleak](#)

ÿ Tutoretzak, Ikasgelak eta Ordutegia

Online tutoretza (ordua eskatuta) eta presentziala Lekua:

Los Tejos eraikina 2. solairua, 2028 bulegoa (Cándido Aramburu irakaslea)

Astelehena: 10:00-13:00

Asteazkena 10:00-13:00

G91 -> A223, G1 ->A113, G2->A122, P91->A015, P1->A305, P2->E-ISM, P3-A eta P3-B ->A015

G91(L-15:00), G1(X-17:00), G2(L-17:00), P91(X-19:00), P1(Os-17:00), P2(M-19:00), P3-A eta P3-B (M-17:00)

E-ISM Ikasgela: "Las Encinas" eraikina (Liburutegiaren eta liburutegiaren arteko eskuinaldean Errektoreta) Sotoan, "ISM" Informatika Laborategia

- Andrés Garde Gurpegui

ÿ Gobernuko Informazio Teknologien eta Telekomunikazioen Zuzendaritza Nagusiko goi-mailako teknikaria Nafarroa.

ÿ Irakasle Agregatua (Laborategiko Praktikak) ÿ Los

Tejos eraikina, 2. solairua, ELEKTRIKO INGENIARITZA saileko Elkargoen Aretoa, ELEKTRONIKA ETA KOMUNIKAZIOA.

ÿ andres.garde@unavarra.es

ÿ [PDI Irakasleak](#)

- Carlos Juan de Dios Ursúa

ÿ Ingeniaritza Elektrikoa eta Elektronikoa

ÿ Robotika eta Automatizazioan Masterra

ÿ Irakasle Agregatua (Konputagailuen Egituraren Teoria) ÿ Los

Tejos eraikina, 2. solairua, ELEKTRIKO INGENIARITZA saileko Elkargoen Aretoa, ELEKTRONIKA ETA KOMUNIKAZIOA.

ÿ carlos.juandedios@unavarra.es

## 1.9. Informatika Gradua

- [Informatika Gradua](#)

ÿ 200 - Industria, Informatika eta Telekomunikazio Ingeniaritzako Goi Eskola Teknikoa

ÿ 240 - Informatika Ingeniaritzan lizentziatura Nafarroako Unibertsitate Publikoan

#### 1.10. Egutegiak

- Egutegi administratiboa

ÿ Teoria

ÿ 1 taldea G1 asteazkena 17:00 A113

ÿ 2 taldea G2 astelehena 17:00 A122

ÿ Taldea91 G91 Astelehena 15:00 A223

- Praktikak

ÿ Informatika Ikasgelako egutegia 1. seihilekoa

ÿ Informatika Ikasgelako egutegia 2. seihilekoa

ÿ P3-A: asteartean 17:00etan A-015ean (40 eserleku) ÿ G1 eta G2 talde teorikoen praktikak

ÿ P3-B: asteartean 17:00etan A-015ean (40 eserleku) ÿ G1 eta G2 talde teorikoen praktikak

ÿ P2: asteazkenean 17:00etan E-ISMN (36 plaza) ÿ G2 Teoria Taldeko Praktikak

ÿ P1: astelehenean, 19:00etan, A305ean (36 eserleku) ÿ Teoria Taldeko Praktikak G1

ÿ P91: asteazkena 19:00etan A015ean (20 eserleku) ÿ Euskarazko Teoria Taldeko Praktikak

ÿ E-ISM Ikasgela: "Las Encinas" eraikina (Liburutegiaren eta Errektoregoaren arteko eskuinaldean) Sotoan,  
"ISM" Informatika Laborategia

IRAILA Asteartea asteazkena osteguna

26 P3-A P2 27 P91 28 P1

URRIA

3 P3-B 5 P1

10 P3-A P2 11 P91

17 P3-B

24 P3-A P2 25 P91 26 P1

31 P3-B

AZAROAK

7 P3-A P2 8 P91 9 P1

14 P3-B

21 P3-A P2 22 P91 23 P1

28 P3-B

2022ko udazkeneko seihileko egutegia

- Festak

Urriaren 12a, asteazkena (Espainiako Jai Nazionala).

Azaroak 1, asteartea (Santu Guztien eguna).

Abenduak 3, asteartea (San Frantzisko Xabier, Nafarroaren eguna).

Abenduak 6, osteguna (Konstituzio eguna).

- Ikastaroaren asteak:

ÿ 15 astetik: Ikastaroaren iraupena irailaren 2tik abenduaren 20ra

- azterketen egutegia

ÿ 2023ko udazkena

ÿ Ohiko Deia 2023/12/18 08:00 Ezohiko Deia: 2024/01/18 16:00

## 1.11. Problemen ebatzenaren bidezko ariketak

- Oharretan jasotako programako kapitulu bakoitzean oinarrizko ariketak egitea.
- Azterketa motako ariketak.

## 1.12. Praktikak

### 1.12.1. Gidoiak

- Bost Praktika Gidoiak oharretan daude eskuragarri: eecc\_book.pdf dokumentuan eta dokumentua [https://kandido.github.io/eecc\\_book.html](https://kandido.github.io/eecc_book.html) azken hau egunerautuena izanik

### 1.12.2. Ebaluazio Ariketak

- Praktika-saioko bakoitzaren hasieran, aurreko saioko edukiari buruzko 20 minutuko banakako ariketa bat egingo da, zeinaren kalifikazioa praktikaren nota orokorraren kalkuluan sartzen baita.

### 1.12.3. Oroitzapenak

- Banakako Praktikak.
- Oroitzapenak :

ÿ Postontziaren zerbitzariaren bidez memoria entrega hurrengo saioa baino lehen egingo da irakasleak adierazi eta argitaratutako egunean praktikak.

ÿ Txosten PDF formatuan dokumentu bakarra da.

ÿ Fitxategiaren izena abizena1\_abizena2\_tituloguionpractica.pdf izan behar da

ÿ Memoria edukia.

ÿ Pseudokodean deskribatutako programa

ÿ Iturburu-kodea mutua-lengoian behar bezala komentatuta, koherentiarekin pseudocode programa.

ÿ Txostenaren estiloa ez da baloratzen bere edukia baizik, garrantzitsua baita ohar gisa balio izatea ebaluazio probak.

ÿ • Ezinbestekoak da oħarrak hartzeari laborategiaren barruan eta kanpoan. Goðe lan guztiak pendrive batean edo bidali posta elektronikoz

ÿ • Disko gogorrekoei edukia egunero ezabatzea automatikoki.

## 1.13. Informazio baliabideak

### 1.13.1. UPNA

## informatika zerbitzua

- NUPeko Informatika Zerbitzua
  - ÿ Erreserba Gelak
  - ÿ VDI ikasgelak
  - ÿ VDI Urruneko Sarbiderako Eskuliburua
  - ÿ Mahai birtuala
- Urruneko Informatika Laborategia
  - ÿ sarbide orokorra ÿ ?

## ARM/FPGA urrutiko laborategia

- ARM/FPGA urruneko laborategiaren erabilera ez da ikastaroaren planean aurreikusi, baina erabilgarri dago nahi duenak, eta horretarako sarbide-kontu bat ireki behar da.
- [fpga sarbidea](#) ÿ FPGA-Raspberry urrutiko laborategia
  - ÿ Guacamole saioa ÿ UPNA kredentzialak
  - ÿ Sartzea DE1-SoC (ARM) txartela ÿ saioa hasi ÿ Linux shell kontsola
  - ÿ Sarbidea Raspberry txartela (ARM) ÿ saioa hasi ÿ Linux shell kontsola
  - ÿ Xilinx (ARM) txartelaren sarbidea ÿ saioa hasi ÿ Linux shell kontsola
- [Labsland enpresa kolaboratzailea](#): DE1-SoC instantziak. Web interfazea.
  - ÿ Zure sarbidea nire kontuaren bidez instalatuko da EDA botoia erabiliz.
  - ÿ Gailu digitalak diseinatzeko tresnak

## 1.13.2. Lanpostuak: 32 eta 64 bit

### Arkitektura



Ikaslearen lan-estazio berezia Ubuntu (64 biteko) x86-64 prozesadore batean bada, ez dago arazorik x86-64 arkitektura duten programak exekutatzeko; hala ere, irakasgaian garatu beharreko programek 32 biteko arkitektura erabiltzen dute. programa hauek konpilatu eta exekutatzeko beharrezkoa da

### Zer instalatu

- Beharrezko da arkitektura x86-64 edo amd64 duen CPU bat eta Linux/[https://www.gnu.org/software/\[GNU\] Sistema Eragilea Ubuntu](https://www.gnu.org/software/[GNU] Sistema Eragilea Ubuntu) banaketa erabiltzen duen PC bat edukitzea . 18 baino bertsio handiagoa duen Ubuntu banaketa berria. Ubuntu banaketa gomendagarria da praktikak egiten diren informatika laborategian instalatutako banaketa baita.
- ÿ Sistema eragilea natiboki edo birtualki instalatu daiteke (VMware, Virtualbox, [Virtio](#), etab).
- Praktikak egiteko, instalazio-tresnekin gutxieneko instalazioa behar da.
  - maila baxuko programazioa: **binutils**, **gcc** konpilatzailea , **gdb** debugger , **vim** editorea
- Ubuntun:
  - ÿ instalatu paketeak `sudo apt-get install binutils gcc gcc-multilib gdb vim` komandoa erabiliz

ÿ egiaztu paketeen instalazioa `dpkg -l binutils gcc gcc-multilib gdb vim` komandoa erabiliz programa bakoitzaren egoera ii dela.  
 ÿ egiaztu bertsioak: `gcc --version && as --version && ld --version && gdb --version && vim --bertsioa`

- C eta ASM mutua-lengoaietan programak editatzeko, Vim editoreaz gain, hobetsia den editorea (nano, sublime, gedit, kate, emacs, etab.) eta/edo hobetsia den IDE (Visual Studio Code) erabil daiteke. , eklipsea, etab.)

#### 1.13.3. Check in

miaulario

- Miaulario

Github

- Beharrezkoa da kontu irekia edukitzea Github biltegiko plataforman

Google

- Google kontua
- Google Chrome arakatzailea: hizkuntza hautatzeko aukera ematen du

#### 1.14. Praktika Taldeak

1. Taula. Praktika Taldeak

IKASLEA EECC	EECC ID	EECC TALDE PRAKTIKAK EECC	IKASGELA	IRAKASLE
Alén Urra, Saúl alen.146325		G1	P3-A (asteartea 17-19)	A015 (30) Andres Garde
Alignani Álvarez, alignani.153933		G1 Tomás	P3-A (asteartea 17-19)	
Altamirano Trujillo, Ruth Nazaret	altamirano.1460 80	G1	P3-A (asteartea 17-19)	
Alustiza Sánchez, Aimar	alustiza.15399	G1	P3-A (asteartea 17-19)	
Álvarez Alonso, Markel	alvarez.147717	G1	P3-A (asteartea 17-19)	
Alvira Ballano, Alonso	alvira.153972	G1	P3-A (asteartea 17-19)	
Andueza Del Campo, Eneko	andueza.153944	G1	P3-A (asteartea 17-19)	
Arellano Cameo, Juan	arellano.154190	G1	P3-A (asteartea 17-19)	
Azkona Ekiza, Iker	azkona.153931	G1	P3-A (asteartea 17-19)	
Aznarez Gil, Inigo	aznarez.138165	G1	P3-A (asteartea 17-19)	
Baquedano Simon, Javier	baquedano.1539	G1 45	P3-A (asteartea 17-19)	
Auger Ribas, Fermin	barrena.154004	G1	P3-A (asteartea 17-19)	

IKASLEA EECC	EECC ID	EECC TALDE PRAKTIKAK EECC	IKASGELA	IRAKASLE
Basterra Arteaga, Paula	basterra.128530 G1	P3-A (asteartea 17-19)		
Bellera Alsina, Albert	bellera.142014 G1	P3-A (asteartea 17-19)		
Biurrun Rodríguez, Luna Maria	biurrun.153961 G1	P3-A (asteartea 17-19)		
Blasco Catalán, Diego	blasco.154001 G1	P1 (osteguna 17-19)	A305 (36)	Andres Garde
Boukhlik, Hajar boukhlik.	153919 G1	P1 (osteguna 17-19)		
Bustince Labairu, Lucas	bustince.154005 G1	P1 (osteguna 17-19)		
Ahuntza Jimenez, Natalia Ana	ahuntza.153898 G1	P1 (osteguna 17-19)		
Pascual kalea, Vidal	kalezuloa.149323 G1	P1 (osteguna 17-19)		
Cardenas Curicho, Solange Samantha	cardenas.13947 4	G1	P1 (osteguna 17-19)	
Cerezo Uriz, Iñaki	gereziondoa.147574 G1	P1 (osteguna 17-19)		
Chacón Flores, Malena Paola	chacon.145983 G1	P1 (osteguna 17-19)		
Chacon Gonzalez, Ruben	chacon.154006 G1	P1 (osteguna 17-19)		
Chueca Irisarri, Daniel	chueca.146727 G1	P1 (osteguna 17-19)		
Clemos Gómara, Samuel	clemos.153868 G1	P1 (osteguna 17-19)		
Couceiro Eizaguirre, Javier	couceiro.126729 G1	P1 (osteguna 17-19)		
Crespo Lamelas, crespo.	153999 G1	Markel	P1 (osteguna 17-19)	
Daban Baines, Jorge	dan.153969 G1		P1 (osteguna 17-19)	
Karmenetik Hernandez, Aimer	delcarmen.1538 71	G1	P1 (osteguna 17-19)	
Elizaincin Irungaray, Unax	elizaincin.15396 5	G1	P1 (osteguna 17-19)	

IKASLEA EECC	EECC ID	EECC TALDE PRAKTIKAK EECC	IKASGELA	IRAKASLE
Notarioa Lopez, Miguel	notarioa.154010	G1	P1 (osteguna 17-19)	
Arantza Mila, Ivan	arantzatsu.153952	G1	P1 (osteguna 17-19)	
Ezcurdia Ruiz, Fermin	ezcurdia.153981 G1		P1 (osteguna 17-19)	
Ezpeleta García,	ezpeleta.153901 G1	Laura	P1 (osteguna 17-19)	
Fernandez Picorelli, Marina	fernandez.149320	G1	P1 (osteguna 17-19)	
Fortun Iñurrieta, Lucas Nicolas	fortuna.140006 G1		P1 (osteguna 17-19)	
Garate Moreno, Gaizka	garate.153980 G1		P1 (osteguna 17-19)	
Doodle Larrayoz, Iñaki	garatea.139258 G1		P1 (osteguna 17-19)	
Garcia Manterola, Julen	garcia.153963 G1		P1 (osteguna 17-19)	
Gomez De Seguru Rodriguez, Laura	gomezdesegura.153995	G1	P1 (osteguna 17-19)	
Gonzalez Augusto, Ibai	gonzalez.154119	G1	P1 (osteguna 17-19)	
Goñi Lara, Iker goni.	147794	G1	P1 (osteguna 17-19)	
Goyache Sarasa, Karlos	goyache.153907 G1		P1 (osteguna 17-19)	
Arriazuko Guardia, Jon	guardia.153924 G1		P1 (osteguna 17-19)	
Errementari Lantza, Eva Isabel	errementaria.139723 G1		P1 (osteguna 17-19)	
Hualde Romero, Israel	hualde.146905 G1		P1 (osteguna 17-19)	
Illa Criado, Alvaro	illa.154577	G1	P1 (osteguna 17-19)	
Iribarren Ruíz, Beñat	iribarren.148642 G1		P1 (osteguna 17-19)	
Juárez Jiménez, Adrian	juarez.146292 G1		P1 (osteguna 17-19)	
Larrayoz Díaz, Asier	larrayoz.146585 G2		P2 (asteartea 19- <small>hogela bat)</small>	EISM (34)
Larrea Gallego, Javier	larrea.154156 G2		P2 (asteartea 19- <small>hogela bat)</small>	Inozoa

IKASLEA EECC	EECC ID	EECC TALDE PRAKTIKAK EECC	IKASGELA	IRAKASLE
Latasa Santxa, Ivan	latase.146388 G2		P2 (asteartea 19-21)	
Leiza Dávila, Leyre	leiza.153964 G2		P2 (asteartea 19-21)	
Louvres Korrikalari, Imanol	argiak.14755 1	G2	P2 (asteartea 19-21)	
Maeztu Vitaller, Karlos	maeztu.154202 G2		P2 (asteartea 19-21)	
Manrique Rangel, Angelo Gabriel	manrique.15394 0	G2	P2 (asteartea 19-21)	
Martinez Alberdi, Pablo	martinez.147687 G2		P2 (asteartea 19-21)	
Martínez Arpon, Alan	martinez.146281 G2		P2 (asteartea 19-21)	
Martinez Erronkari, Rodrigo	martinez.153968 G2		P2 (asteartea 19-21)	
Martínez Sesma, Álvaro	martinez.149151 G2		P2 (asteartea 19-21)	
Merino Pinedo, Javier	merino.153915 G2		P2 (asteartea 19-21)	
Molina Puyuelo, Alexandro	errota.147893 G2		P2 (asteartea 19-21)	
Morala Garcia, Haizea	morala.147341 G2		P2 (asteartea 19-21)	
Munoz Villanueva, Asier	munoz.138255 G2		P2 (asteartea 19-21)	
Nagore Irigoyen, Alexandro	nagore.95315 G2		P2 (asteartea 19-21)	
Ochoa Martínez, Angela	octa.153956 G2		P2 (asteartea 19-21)	
Orenes Peñas, Martin	oreinak.153979 G2		P2 (asteartea 19-21)	
Ozcáriz Armendariz, Iñaki	ozcariz.153949 G2		P2 (asteartea 19-21)	
Parrales Gómez, Sarah Alice	mahatsonoak.153967 G2		P2 (asteartea 19-21)	
Percaz Angós, Asier	percaz.154165 G2		P2 (asteartea 19-21)	
Pidlubnyy Lahodyn, Artem	pidlubnyy.15427 2	G2	P2 (asteartea 19-21)	

IKASLEA EECC	EECC ID	EECC TALDE PRAKTIKAK EECC	IKASGELA	IRAKASLE
Pinillos Osteriz, Iñaki	pinillos.153938 G2	P2 (asteartea 19- <small>hogeita bat)</small>		
Pío Aymerich, Mikel	pio.153922	G2	P2 (asteartea 19- <small>hogeita bat)</small>	
Prim Labiano, Ainara	prim.153947	G2	P2 (asteartea 19- <small>hogeita bat)</small>	
Razvant Sofrone, Bogdan	razvant.153957 G2		P2 (asteartea 19- <small>hogeita bat)</small>	
Reimóndez Zuria, Luis Rafael	reimondez.1540 02	G2	P2 (asteartea 19- <small>hogeita bat)</small>	
Resano Eguizabal, Amaia Duo	resano.154007 G2		P2 (asteartea 19- <small>hogeita bat)</small>	
Rodriguez Hidalgo, Junior Albert	rodriguez.15394 1	G2	P2 (asteartea 19- <small>hogeita bat)</small>	
Rojas Urrutia, Aitor	gorria.150213 G2		P2 (asteartea 19- <small>hogeita bat)</small>	
Errameroa Samaniego, Oscar Hugo	errameroa.153951 G2		P2 (asteartea 19- <small>hogeita bat)</small>	
Ros Villanueva, Alvaro	ros.154011	G2	P3-B (asteartea 17-19)	A015 (30) Andres
Ruiz de Gopegui Rubio, Paula	ruizdegopegui.1 48800	G2	P3-B (asteartea 17-19)	
Sáez Zubillaga, Javier	saez.154043 G2		P3-B (asteartea 17-19)	
Sáinz González, Mario	sainz.153904 G2		P3-B (asteartea 17-19)	
Santos Garzon, Jaider Felipe	santuak.138639 G2		P3-B (asteartea 17-19)	
Sotelo Agirre, Martin	sotelo.153910 G2		P3-B (asteartea 17-19)	
Suarez Etulain, Diego	suarez.153960 G2		P3-B (asteartea 17-19)	
Tiraplegui Etulain, Manex	tiraplegui.15401 2	G2	P3-B (asteartea 17-19)	
Vallés Zamora, Migel	haranak.146260 G2		P3-B (asteartea 17-19)	
Vicente Obeso, Mario	vicente.154000 G2		P3-B (asteartea 17-19)	

IKASLEA EECC	EECC ID	EECC TALDE PRAKTIKAK EECC	IKASGELA	IRAKASLE
Vinces Velastegui, José Inazio	vinces.153986 G2	P3-B (asteartea 17-19)		
Zamboran Maldonado, Andrea	zamboran.14623 1	G2	P3-B (asteartea 17-19)	
Zheng, Yushan zheng.	zheng.146049 G2		P3-B (asteartea 17-19)	
Zubisti Aristu, Ruben	zubasti.153989 G2		P3-B (asteartea 17-19)	

## 1.15. Metodologia

- Teoria, Ariketak, Praktika eta Azterketak.

2. taula. metodologia

Metodologia - Jarduera	Pertsonalaren Ordutegia	Kontaktu gabeko orduak
A-1 Master Class	24	
A-2 Ikasketa autonomoa		30
A-3 Saio praktikoak	16	
A-4 Programazioa / esperimentazioa edo bestelako lanak ordenagailuan / laborategian		hogei
A-5 Arazoak ebaaztea, ariketak eta bestelako jarduerak aplikazio		12
A-6 Oinarritutako ikaskuntza arazoak eta/edo kasuak	14	
A-7 Lanak prestatzea eta/edo proiektuak eta memoria idaztea		hamarka
A-8 gidoiak irakurtzea, aurkezpenak prestatzea lanak, proiektuak, etab...		hamabost
A-9 Ebaluazio-jarduerak		6
A-10 Tutoretzak	2	
Guztira	62	88

- Gaia 2 zatitan banatuta dago

ÿ I. zatia: 1-6 gaiak (oinarrizko kontzeptuak)

ÿ II. zatia: 7-9 gaiak (kontzeptu aurreratuak eta kasu praktikoak)

### 1.15.1. Kreditu banaketa

- Kredituen banaketa:

ÿ kreditu guztira: 6 ECTS  
ÿ ordu guztira: 6x25: 150 ordu  
ÿ Harremanetarako ordutegia: 62 ordu  
ÿ eskolak: 24 ordu. Ikastaroaren guztira 15 astetako 12 astez (2 ordu/astean).  
ÿ laborategiko praktikak: 16 ordu. 8 astez (2 ordu/astean)  
ÿ arazoak: 14 ordu. 7 astez (2 ordu/astean)  
ÿ ebaluazioa: 6 ordu. Ikastaroaren guztira 15 astetako 3 astez (2 ordu/astean).  
ÿ tutoretzak: 2 ordu  
ÿ Harremanik gabeko orduak: 88 ordu

### 1.15.2. Praktiken kredituen banaketa

- Ikasle bakoitzak 16 orduko praktika izango ditu 2 orduko saioetan: 12 laborategian eta 4 Oroitzapenak osatzeko.

## 1.16. Ebaluazioa

- Ordenagailuen Web Egitura
- Azken nota: % 15 (asistentzia, klasean eta laborategian jarrera) + % 35 (praktikak) + % 35 (teoria eta bere ariketak)+%15 (paperezko programazioa)
- praktikak: praktika saio bakoitzeko ariketak, txostenak eta azken proba. Azken proba 2023/01/18an (1., 2., 3., 4., 5. eta 6. praktikak), BMP irudiak sortzeko praktikan arreta berezia jarriz.
  - ÿ Praktikak nota orokorraren % 35 dira, gutxienez 4 puntuko kalifikazioa behar da irakasgaia gainditu eta berreskuragarriak dira.
  - ÿ Praktikaren nota hauen batez besteko hiztunak da: praktiketara joatea eta txostenak ematea (% 15), saio bakoitzean laborategian egindako ariketen ebaluazioa (% 20) eta praktikako azken azterketa (% 65).
- teoria: nota orokorraren %35 adierazten du, lehenengo partzialean gutxienez 4 puntuko kalifikazioa behar da. irakasgaia gainditzeo eta berreskuragarriak dira.
  - ÿ 2 proba partzial. Lehen proba partziala urriaren 16an (1etik 6ra bitarteko gaiak). 2023/12/18ko azken deialdi arrunteko bigarren proba partziala (6tik 9ra bitarteko gaiak eta 1etik 6ra lehenengo partzialean ikusi ez zena).
  - ÿ Lehenengo partzialak teoriaren notaren %60ko balioa du eta bigarren partzialak %40.
- Paperezko programazioa
  - ÿ Azken notaren %15 adierazten du
- Ezohiko Deialdia:
  - ÿ Praktikak edo teoria bertan behera utzi badituzu, bata onartu eta bestea bertan behera, Onartutako zatiaren kalifikazioa (teoria edo praktika) gordetzen da.
  - ÿ Derrigorrezko proba praktiko eta teorikoak:
- Laborategiko praktiketara joatea: Derrigorrezko da laborategiko praktiken orduen %87,5era joatea. laborategia.

- Praktika-txostenen entrega: Derrigorrezko da txostenen %100a myaulario zerbitzarian gaituta dagoen egunean ezarritako epean entregatzea. Praktiken edo zereginen txostenak ez dira jasotzen posta zerbitzariak ezarritako epetik kanpo.
- Praktika-orduen %87,5ean derrigorrezko asistentzia eta txostenak entregatzea Ezarritako epea eta erdia ezinbesteko baldintza da irakasgaia gainditu ahal izateko.

#### 1.17. Azterketak: Edukiak

- Azterketa teorikoak klasean ikusitako kontzeptuei buruzko galdera teorikoak eta lehenengo zatian "gutxi gorabeherako" banaketa duten ariketak izango dira: %20 kontzeptu teorikoei buruzko galderak eta %80 ariketak; eta bigarren partzialerako: %60 kontzeptu teorikoei buruzko galderak eta %40 ariketak.
  - Paperezko programazio-azterketa ordenagailurik gabe egingo da eta mihiztadura-lengoaiaren instrukzioen erreferentzia-orriekin, GDB araztailearekin eta C lengoaiaren adierazpenekin, mihiztatzaile-lengoaiaren programazioari buruzko galdelez osatuta dago eta ez GDBri buruzkoa.
  - Laborategiko azterketa praktikoa txosten praktikoekin eta erreferentzia-fitxekin egingo da eta informazio elektronikorik erabili gabe ez urrutitik (Interneterako sarbidea) ez lokalean (USB pendrive-a, etab.).
- Egutegia:

1. teoria partziala: 1,2,3,4,5 eta 6 gaiak: 2022/10/16 2.

teoria partziala: Gai guztiak: 2023/12/18

Praktika azterketa: Ohiko Deialdia 2023/12/18

## 2. kapitula. Von Neumann Arkitektura

### 2.1. Von Neumann Arkitektura

- Kalkulatu batura  $\sum_{i=1}^N i = N(N + 1) / 2$

#### 2.1.1. Ikastaroa

2. Von Neumann Arkitektura:

- to. CPU
- b. Memoria
- c. Sarrera irteera

#### 2.1.2. Testuinguru historikoa

##### Aurrekariak

- 1833: Charles Babbage ѹ 1. ordenagailu mekanikoa diseinatu zuen
- 1890: Herman Hollerith-en tabulazio-makina . AEBetako errolda. IBM (1925)
- 1936: Alan Turing ѹ Algoritmoak eta Turing makinaren kontzeptua. Enigma kode makina.
- **Bigarren Mundu Gerra 1939-1945**
- 1944: AEB, IBM Harvard Mark I ordenagailu elektromekanikoa
- 1944: Colossus (Koloso Mark I eta Colossus Mark 2). Komunikazioak deskodetzea.

##### ENIAC

- 1947: **ENIAC** (Electronic Numerical Integrator And Calculator) eraiki zen Pennsylvaniako Unibertsitatean (artilleriaren ikerketa balistikorako laborategia).
  - 尹 Balistikari buruzko ekuazio diferentzialak (angelua = f (kokapena, buztaneko haizea, haize gurutzatua, airearen dentsitatea, tenperatura, oskolaren pisua, propultsatzailearen karga, ...))
  - 尹 Erabilera orokorreko ordenagailu elektronikoa (ez mekanikoa) .
  - 尹 Memoria: 20 metagailu bakarrik ѹ triodoekin egindako txankletak
    - 尹 18.000 tutu elektroniko edo huts-balbula
  - 尹 Etengailuen eskuzko programazioa
  - 尹 100.000 jarraibide segundoko
  - 尹 300 biderketa segundoko
  - 尹 200 kW
  - 尹 13 tona eta 180 m2

##### EDVAC

- 1951: Pennsylvaniako Unibertsitatean (J. Presper Eckert eta John William Mauchly) **EDVAC** (Electronic Discrete Variable Automatic Computer ) funtzionatzen hasi zen , **John von Neumannek** asmatua , ENIAC ez bezala ez zen hamartar, bitarra baizik, eta zuen. biltegiratzeko diseinatutako lehen **programa** (ez soilik datuak) : GORDETAKO PROGRAMA ORDENAGAILUA ѹ programa datu gisa manipula daiteke.
  - 尹 500.000 \$

ŷ EDVAC-ek fisikoki ia 6.000 balbula termoioniko eta 12.000 kristal-diodo zituen. kontsumitu 56 kilowatt-eko potentzia. 45,5 m<sup>2</sup>-ko azalera zuen eta 7.850 kg pisatzen zuen.

ŷ Arkitektura:

- ŷ Zinta magnetikoko irakurgailu-grabagailu bat
- ŷ osziloskopiodun kontrol-unitatea, kontrol-argibideak jasotzeko unitatea
- ŷ memoria: 2000 hitz biltegiratzea "merkurioko atzerapen-lerroak" ŷ fidagarritasun eskasa
- ŷ koma mugikorreko unite aritmetikoa 1958an.

## IAS

- 1946-1952: **IAS** (Institute Advanced Studies) mainframe:

ŷ EDVAC-en bilakaera: memoria-unitate nagusia eta bigarren mailako danbor magnetikoa.  
ŷ Selectron memoria: biltegiratze kapazitiboa ŷ karga elektrostatikoa

## Beranduago

- 1952: **UNIVAC I** (UNIVersal Automatic Computer I) lehen ordenagailu nagusi komertziala izan zen. AEBetan errolda prozesatzeko aplikatutako Hollerith tabulazio-makinaren bilakaera.
- 1952: IBM 701, garatzen ari zen bitartean "Defentsa kalkulagailua" izenez ezagutzen dena, IBMren lehen ordenagailu zientifiko komertziala izan zen ŷ lehen **MUNTATZAILE lengoaia**.
- 1964: **IBM 360** mainframe ŷ lehen ordenagailua ISA (mikroprogramazioa) ŷ bateragarritasuna  
ŷ siliziozko osagai diskretuen eta beste osagai batzuen arteko teknologia hibridoa ŷ ez "zirkuitu integratuak".  
ŷ Oinarrizko sistema eragilea/360 (BOS/360), Disko sistema eragilea/360 (DOS/360)

## Erdieroaleen Teknologia

- 1947: Bell laborategietan, John Bardeen, Walter H. Brattain eta William Shockley-k asmatu zuten. **transistorea**.
- 1958: Kilby, germaniozko lehen zirkuitu integratua.
- 1957: Robert Norton Noyce, Fairchild Semiconductor-en sortzailekidea, lehen zirkuitu integratu planoa
- 1968: Robert Norton Noyce eta Gordon Moore Intel aurkitu zuten.
- 1971: Intel 4004 ŷ siliziozko CPU integratua ŷ 8 bit

## 2.2. Institute Advanced Machine (IAS): Arkitektura

### 2.2.1. Erreferentzia

- [Von Neumann Makina](#)

### 2.2.2. sum1toN Programaren adibidea

Kode bitarra kalkulatzeko

$$\sum_{i=1}^5 i$$

Address	Data	Comments
0	01 005	loop: S(x)->Ac+ n ;load n into AC
0	0F 002	Cc->S(x) pos ;if AC >= 0, jump to pos
1	00 000	halt ;otherwise done
1	00 000	.empty ;a 20-bit 0
2	05 007	pos: S(x)->Ah+ sum ;add n to the sum
2	11 007	At->S(x) sum ;put total back at sum
3	01 005	S(x)->Ac+ n ;load n into AC
3	06 006	S(x)->Ah- one ;decrement n
4	11 005	At->S(x) n ;store decremented n
4	0D 000	Cu->S(x) loop ;go back and do it again
5	00 000	n: .data 5 ;will loop 6 times total
5	00 005	
6	00 000	one: .data 1 ;constant for decrementing n
6	00 001	
7	00 000	sum: .data 0 ;where the running/final total is kept
7	00 000	
8	00 000	
8	00 000	

15. Irudia IAS makinan sum1toN programaren makina-kodea

#### Programazio inperatiboa

- Paradigma:

ŷ Paradigma inperatiboa edo estrukturala: programa bat garatuz ezartzen da algoritmoa  
makinak exekutatu behar dituen ORDENAK ditu

ŷ Programazio deklaratiboa ez bezala: algoritmoak programak nahi dugun ZER implementatzen du.  
ordenagailua, ez NOLA, ez zuzenean exekutatu beharreko aginduak.

Adibidez, eragiketa  $\sum_{i=1}^5 i$ , Python-en honela deskriba daiteke:

batura (barrutia (5,0,-1))

#### Memoriaren edukia: datuak eta argibideak

- IAS ordenagailua zuzenean makina lengoaien programatu zen, ez zuen hizkuntza sinbolikorik  
muntaia hizkuntza bezala.
- Makina Lengoaia: Kode Bitarra
- Kode bitarra editatzeara txartel zulatuak edo zinta magnetikoak erabiliz kontsola baten bidez.
- Memorian jasotako informazio mota: DATUAK eta JARRAIBIDEAK
  - ŷ Datuen adibidea: zenbaki osoak +3278,+5,-1,-6592,...
  - ŷ Argibide adibideak:
    - ŷ LOAD M(8): kargatu 8. memoria-posizioko edukia metagailu-erregistroan
    - ŷ GEHITU M(3): memoriaren 3. posizioko edukiak gehitzen ditu metagailu-erregistroan
    - ŷ JMP M(100): 100. memoria-posiziora salto egin
    - ŷ etab
- Biltegiratutako programaren kontzeptua: Unitatean gordetako instrukzio bitarrak eta datu bitarrak  
Memoria

- ÿ Von Neumann arkitekturaren nobedade handia izan zen
- ÿ Beharrezko da modulu bitarra ordenagailuaren MEMORIAN KARGATZEA gera dadin gordeta.
- Programazio sekuentziala: jarraibideak sekuentzialki exekutatzen dira bertan gordetzen diren heinean memoria... sekuentzia hausten duen jauzi-instrukzio esplizitua exekutatzen ez den bitartean.

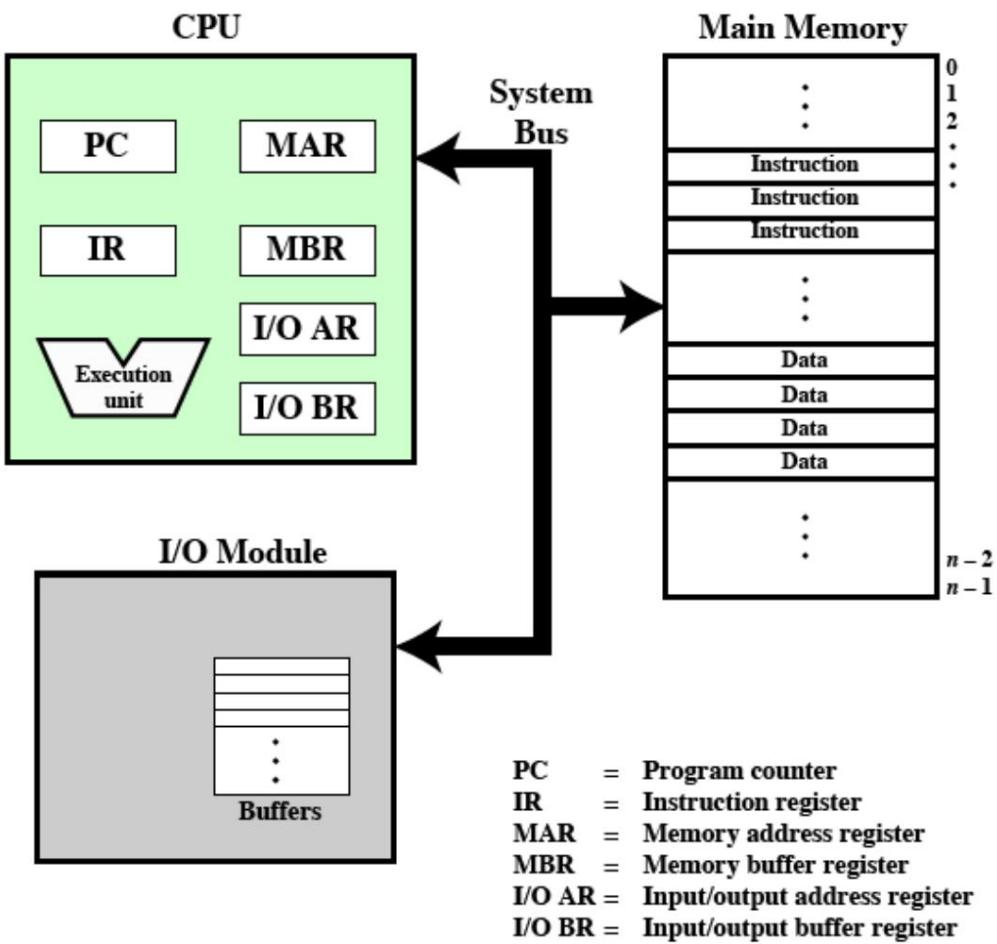
#### Arkitektura: Instruction Set Architecture (ISA)

- Programa aztertzeko beharrezko da makinaren hizkuntza bitarra ezagutzeaz gain bere ARKITEKTURA ezagutzea ere. Ordenagailuaren arkitektura makinaren ZER da, hau da, makina ZEIN instrukzio exekutatzeko gai den, horretarako IRAKASPEN ERREPERTORIOAREN ARKITEKTURA (Instruction Set Architecture ISA) ezagutza beharrezko da:
  - ÿ instrukzio multzoa: eragiketak eta datuetara sartzeko modua
  - ÿ memoria-hierarkia: memoria nagusia eta erregistroak
  - ÿ instrukzioa eta datuen formatua
- ÿ ISA ordenagailuaren hardware fisikoaren lehen abstrakzio maila da.

### 23. IAS ordenagailuaren egitura

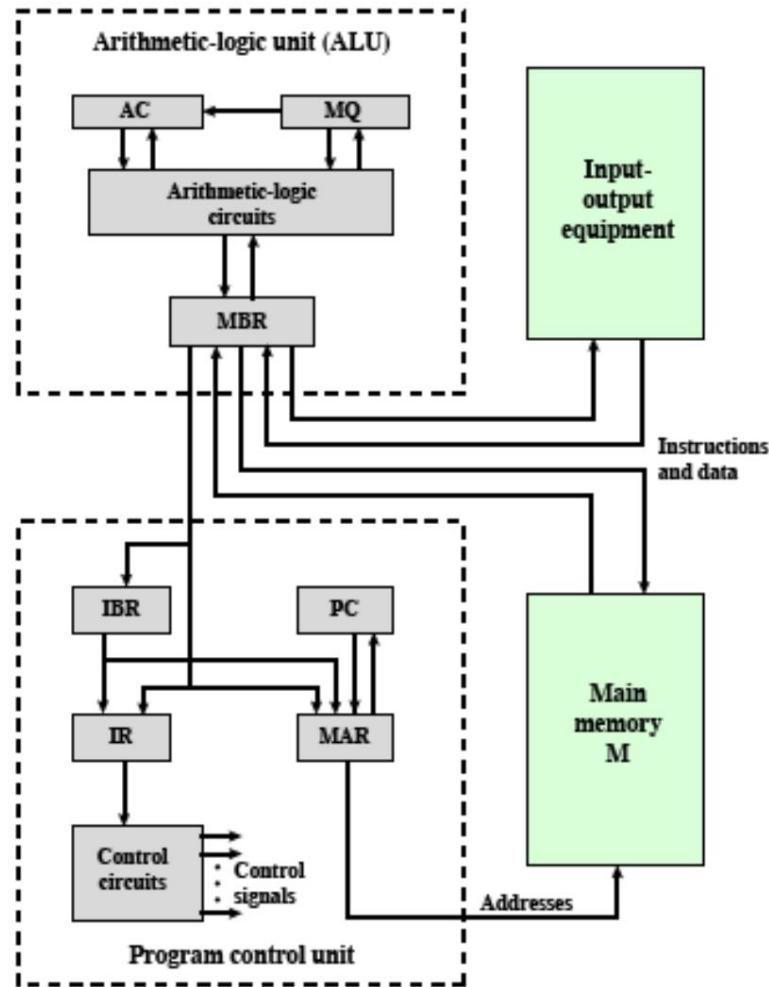
#### 2.3.1. Moduluak

- ÿ Egitura makinaren NOL edo Zerrendako moduluak du gizarteakatu ahal izateko
- Egitura modularra duen hardwarea:
  - ÿ CPU-Memoria-I/O-Bus
  - ÿ Memoriaren hierarkia: 2 maila: Memoria nagusia (PUZaren kantza) eta Erregistroak (barnekoak). CPU



### 16. Irudia IAS makinen arkitektura

- CPUaren Barne Arkitektura: Mikroarkitektura



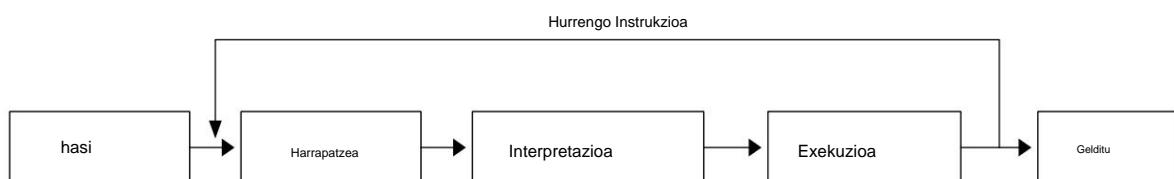
**Figure 2.3 Expanded Structure of IAS Computer**

17. Irudia IAS makinaren egitura

### 2.3.2. Prozesatzeko Unitate Zentrala (CPU)

- CPU:

ŷ CPUaren Funtzionamendua 3 FASEtan banatzen da: Hartu, Interpretatu eta Exekutatu argibideak sekuentzialki. 3 faseen sekuentzia **Instrukzio Zikloa bezala ezagutzen da.**



18. Irudia. Irakaskuntza Zikloa

- Programa baten makina-instrukzio bakoitza PUZak eta horretan harrapatzen, interpretatzen eta exekutatzen du ordena. Sekuentzia hori kontrolatzeko ardura duen zirkuitu elektroniko digitala Unitatea **da PUZan integratutako kontrola**. Kontrol Unitateak seinale elektronikoen bidez mikroaginduak ematen dizkio harrapatzeko azpizirkuitua, interpretearen azpizirkuituari eta exekutatzailaren azpizirkuituari, beraz, guztiak memoria nagusian gordetako programaren instrukzio bakoitzaren instrukzio-zikloaren faseak.

- CPUaren hiru azpimodulu nagusi:

ü Kalkulu-unitatea: Unitate Aritmetiko-Logikoa (ALU)  
 ü Kontrol-unitatea: Instrukzio Zikloa implementatzen duen zirkuitu sekuentziala, bloke ezberdinei (ALU, memoria nagusia, erregistroak, busak, etab.) agindu elektrikoak emanet fase bakoitzean, instrukzio-zikloa amaitu arte.  
 ü Memoria-erregistroak: Datu bat edo bi instrukzio idatzi edo irakurri daitezke erregistro batean.

### 2.3.3. Oroitzapenak

#### Oroimen nagusia

ren espazioa HELBIDEAK	EDUKIA
0x00000000	01010101010101010
0x00000001	01010101010101010
0x00000002	01010101010101010
0x00000009	
0x0000000a	
0x0000000f	

#### 19. Irudia Memoria Nagusiaren edukiak bideratzea

- Exekutatu beharreko programa kode bitar batean gorde behar duzu.
- CPUa memoria nagusirako sarbidea duen modulu bakarra da.
- Programaren argibideak eta datuak sekuentzialki gordetzen dira.

- Programa bi ataletan gordetzen du: Datuen Atala eta Argibideen Atala
- Ausaz eskuragarri dauden Hitzetan antolatuta. Ausazko Sarbide Memoria.
- Dinamismoa: Datuak eta argibideak irakurtzea/idaztea
- IAS makinan memoria-helbideek gorde dezaketen 40 biteko hitzak edo bat seinalatzen dituzte 40 biteko datuak edo 20 biteko bi instrukzio bakoitza.
- Random Access Memory (RAM): memoria-kokapen bakoitza zuzendu daiteke.
- Memoria partekatua: datuen eta argibideen arteko memoria partekatua. Sarbide autobusa ere partekatzen dute memoriara.
- $2^{12}=4K$  hitzetarako gaitasuna hitz bakoitzeko 40 bitrekin.

$$\circlearrowleft 4K \times 40\text{bit} = 4K \times 5\text{byte} = 20\text{KByte}$$

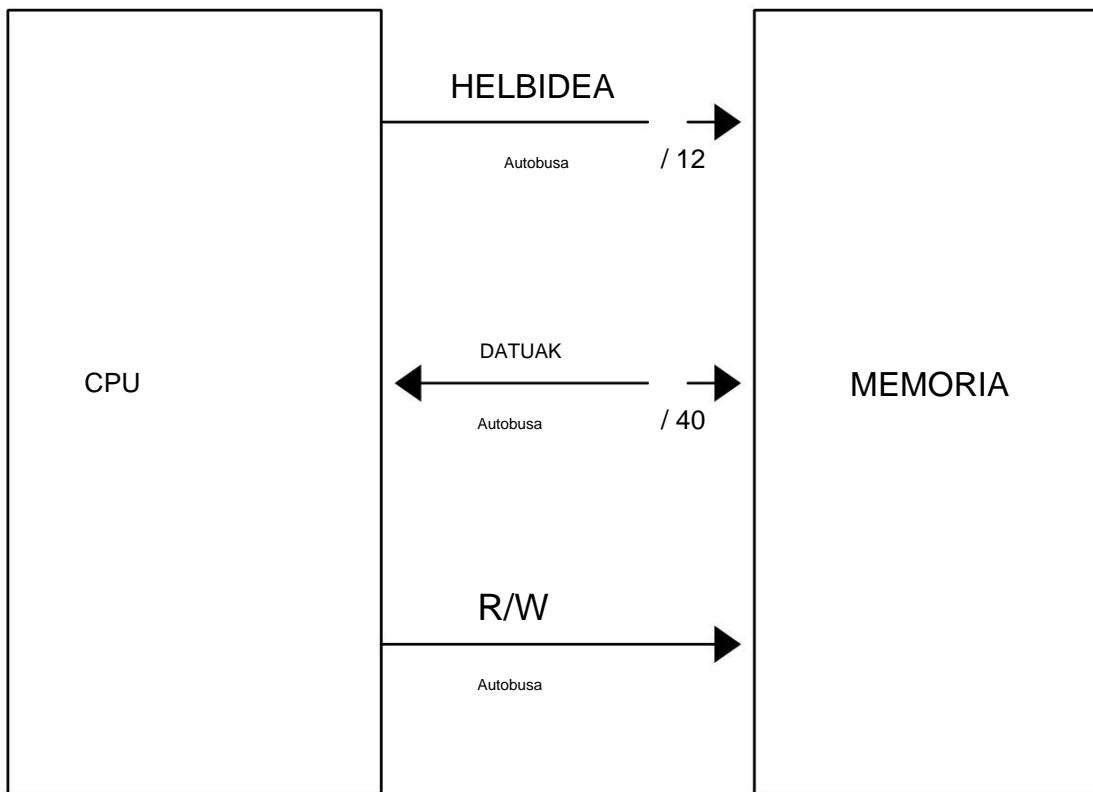
ÿ Bestalde, garai hartan zegoen memoria fisikoa hauxe zen: 40 biteko 1024 hitz = 5 KByte («The Computer from Pascal von Neumann» liburua, Herdman Godstine, pg314, ISBN 0-691-02367-0). Muga teknologikoa.

#### CPU erregistroak

- PUZaren barne memoria: 2 erregistro mota: programatzailailek eskura dezake eta ez dago eskuragarri programatzailailea.
- AC eta AR/MQ: ALU metagailuak. Biderkatzailea/Kotientea. Hauek dira erabiltzaileak eskura ditzakeen erregistro bakarrak programatzailailea.
- Programatzailailek ez ditu eskura daitezkeen erregistroak: Kontrol Unitateko erregistro guztiak: MBR,PC,IR,IBR,MAR
  - ÿ MBR: Selectron Register edo Memory Buffer Register MBR edo Data Buffer Register DBR. 40 biteko tamaina. Instrukzio-zikloaren harrapaketa-fasearen ondoriozko memoriatik irakurritako datuak edo instrukzio-pareak gordetzen ditu edo instrukzio-zikloaren azken faseko memorian idatzi beharreko datuak gordetzen ditu.
  - ÿ PC: Kontrol-kontagailua: Programa-kontagailua (PC) edo Instrukzio-erakuslea (IP). 12 biteko tamaina. Harrapatzeko hurrengo instrukzioa seinalatzen du
  - ÿ IR: Kontrol Erregistroa: Instruction Register IR ere deitua. 20 biteko tamaina. Gorde irakaskuntza-zikloan jasotako instrukzioa
  - ÿ IBR: Instruction Buffer Register: Instrukzio-zikloan jasotako bigarren instrukzioa gordetzen du. 20 biteko tamaina. Kontuan izan horrek esan nahi duela harrapatzeko fasean bi instrukzio aldi berean harrapatzen direla.
  - ÿ MAR: Memory Address Register: uneko Memoria Helbidea. 12 biteko tamaina. Eragileari edo irakaskuntza-zikloaren lehen fasean jaso beharreko irakaskuntza.
- Bere tamainak arkitekturaren "hitzen tamaina" deritzona definitzen du. IAS makinak bat dauka 40 biteko arkitektura edo 40 biteko Word

#### 2.3.4. Autobusa

- Bi gailu elektroniko konektatzeko metalezko hari edo bide paraleloen multzoa. Denek Esku artean USB kable bat izan duzu USB bus bat (Universal Serial Bus).
- Sistema Busa:
  - ÿ CPU-Memoria Nagusiaren interkonexioa: datuen eta argibideen transferentzia.
  - ÿ Datu-Busa (40 hari), Helbide-Busa (12 hari) eta Kontrol-Busa (Irakurri/Idaztea) (hari 1). In Guztira 53 hari edo pista behar dira CPUa eta Memoria Nagusia elkarrekin konektatzeko.



20. Irudia CPU-Memoria Nagusiaren Konexioa

### 2.3.5. Sarrera Irteera (I/O)

- Ordenagailu baten sarrerak eta irteerak beharrezkoak dira haien funtzionatu ahal izateko, bai programatzalearekin, bai beste makinekin. Ordenagailura kanpotik sartzeko, teklatuak, pantailak eta abar bezalako periferikoak beharrezkoak dira.
  - ÿ IAS makinan programa txartel zulatuetan idazten da (Punch Cards). Datu-txartelak eta argibide-txartelak. Beharrezko da datuak memorian kargatzea programa exekutatu aurretik.
  - ÿ txartel zulatuak, kontsola, danbor magnetikoak, zinta magnetikoak, memoria-kargagailua txartel-irakurgailua erabiliz, pantaila huts-hodiak erabiliz, etab. ÿ teknologia zaharkitua.
  - ÿ Ez dugu I/O modulua kontuan hartuko eta CPU-Memoria Nagusiko moduluetan zentratuko gara.

### 2.3.6. Irakaskuntza Zikloaren Animazioa

- [Irakaskuntza-zikloaren animazioa](#)

## 2.4. ISA: IAS Machine Instruction Repertoire Architecture

### 2.4.1. IAS Informatikako Datuen Formatua eta Argibideak

- Memoriaren Arkitektura
  - ÿ Hitza

ÿ 40 bit: 1 datu edo 2 instrukzio

ÿ Datuak

ÿ Zenbaki osoak 2 osagarri formatuan.

ÿ Datuen formatua



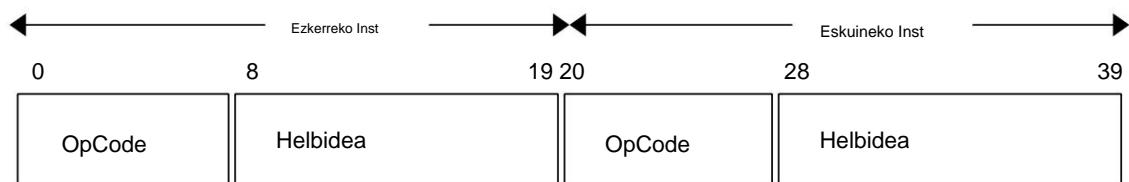
## 21. Irudia Datuen formatua

ÿ Kontuan izan zero zenbakidun bit-a ezkerrekoa dela.

ÿ Argibideak

ÿ 8 biteko eragiketa-kodea eta ondoren 12 biteko eragigai bat (datuen helbidea)

ÿ Argibide-formatua



## 22. Irudia Argibide-formatua

ÿ **Eragiketa bakarra** edo bat ere ez definitzen dugu instrukzio bakoitzean

ÿ Metagailuetan Oinarritutako Arkitektura

ÿ Bi eragiketa behar dituen eragiketa batek implizituki eragiketa bati egiten dio erreferentzia metagailuan gordeta

ÿ Kontuan izan zero zenbakidun bit-a ezkerrekoa dela.

ÿ Ezkerreko instrukzioa (0-19) CPUaren barneko erregistroetan kargatzen da, kodea IR funtzionamendua eta SEAk operazio eremua.

ÿ Eskuineko instrukzioa (20-39) PUZaren barneko erregistroan kargatzen da, IBR.

ÿ Operandoen helbideratzetako modua: Eragileen erreferentzia. Arkitektura hau diseinatu zen "Helbide zuzena" izeneko helbideratzetako modua soilik eremuan dagoen tokian instrukzio-eragigaiak **eragigaiaren memoria-helbidea** zehazten du

- Memoria Edukia

ÿ Memoria-helbideak bikoitz gisa bistaratzen ditugu, lehenengoari 20an aipatzen baitute LSB bitak eta bigarrena 40 biteko memoria-hitz baten 20 MSB bitetara.

ÿ Kontuan izan datuen zutabeen bi atal daudela: argibideen atala eta datuak

ÿ Von Neumann arkitekturan, datuek eta instrukzioak helbide-espazio bera partekatzen dute. memoria.

Address	Data	Comments
0	01 005	loop: S(x)->Ac+ n ;load n into AC
0	0F 002	Cc->S(x) pos ;if AC >= 0, jump to pos
1	00 000	halt ;otherwise done
1	00 000	.empty ;a 20-bit 0
2	05 007	pos: S(x)->Ah+ sum ;add n to the sum
2	11 007	At->S(x) sum ;put total back at sum
3	01 005	S(x)->Ac+ n ;load n into AC
3	06 006	S(x)->Ah- one ;decrement n
4	11 005	At->S(x) n ;store decremented n
4	0D 000	Cu->S(x) loop ;go back and do it again
5	00 000	n: .data 5 ;will loop 6 times total
5	00 005	
6	00 000	one: .data 1 ;constant for decrementing n
6	00 001	
7	00 000	sum: .data 0 ;where the running/final total is kept
7	00 000	
8	00 000	
8	00 000	

23. Irudia IAS makinaren makina kodea sum1toN

## 2.4.2. ISA errepetorioa

### RTL hizkuntza

- **Etranskinean** Erregistro Transferentzia Hizkuntzari (RTL) buruzko informazioa
- Erregistro-transferentzia-lengoaiari esker, programak mikroeragiketen mailan deskriba daitezke, makina-lengoaia bitarrak eta mihiztadura-lengoaia bezala. RTL hizkuntzaren abantaila da bere sintaxia ordenagailuaren arkitekturatik independentea dela, hau da, hizkuntza Unibertsala dela. Beraz, programa bat RTL hizkuntzan deskribatzen badugu, haren transkripzioa amd64, ARM, RISC-V, etab. arkitektura espezifikoa bat duen mutua-lengoaia batera transkripzioa askoz errazagoa da.

### IAS makinaren errepetorioa

- Instruction Set Architecture (ISA): instrukzio multzoaren definizioa eta ezaugarriak. Irakaskuntza Errepetorioaren Arkitektura.
- Jatorrizko bertsioan ez zegoen mutua-koderik, zuzenean makina-lengoaiaren programatu zen.
  - ÿ Erantsitako taulan, bigarren zutabeen, instrukzio-eragiketen **MNEMONIKA** (LOAD, ADD, SUB, etab.) William Stallings-en testuliburuak diseinatutakoekin bat datozi. Lehenengo eta azken zutabetan eragiketak erregistroen arteko transferentzia-hizkuntza baten bidez sinbolizatzen dira.

ÿ Selectron Memoria Nagusirako erabiltzen den teknologiaren izena da.

ÿ S(x) notazioa RTL idazkeran M[x]-ren baliokidea da.

ÿ R.W.Stalling-ek MQ erregistroari deitzen dion AR erregistroa da.

3. Taula. Argibide multzoa

Instrukzioen yam	instrukzioa	Op	Deskribapena	Izena eman Transferentzia Hizkuntza (RTL)
S(x)ÿA C+	KARGA M(X)	1	kopiatu Selectron kokapenean x zenbakia AC-ra	AC ÿ M[x]

Instrukzioa zioa yam	instrukzioa yam	Op Deskribapena	Izena eman Transferentzia Hizkuntza (RTL)
S(x)ÿA c	KARGA -M(X)	2 #1 berdina baina kopiatu zenbakiaren negatiboa	AC ÿ ~M[x]+1
S(x)ÿA cm	ZARGA  M(X)	3 #1 berdina baina kopiatu balio absolutua	AC ÿ  M[x]
S(x)ÿA cm	ZARGA - M(X)	4 #1 berdina baina balio absolutua kendu	AC ÿ AC- M[x]
S(x)ÿA h+	GEHITU M(X)	5 Gehitu Selectron kokapenean x zenbakia AC-n	
S(x)ÿA	AZPI M(X) h	6 kendu Selectron kokapenean x zenbakia ACtik	
S(X)ÿ AhM	GEHITU  M(X)	7 #5 berdina, baina balio absolutua gehitu	
S(X)ÿ Ah-M	AZPIA  M(X)	8 #7 bezala, baina balio absolutua kendu	
S(x)ÿ R	KARGA MQ,M(X)	9 kopiatu Selectron kokapenean x zenbakia AR-n	
RÿA KARGA MQ		Kopiatu zenbakia AR-n AC-ra	
S(x)*R ÿA	MUL M (X)	B Biderkatu Selectron kokapenean x zenbakia zenbakia AR-n. Jarri emaitzaren ezkerreko erdia AC-n eta eskuineko erdia AR-n.	
A/S(x) ÿR	DIV M(X)	C Zatitu AC-ko zenbakia Selectron-eko zenbakiaz kokapena x. Jarri zatidura AR eta gainerakoan AC-n.	
CuÿS( x)	JUMP M(X,0:19) D	Jarraitu exekuzioan ezkerreko instrukzioan parekatu Selectron kokapenean x	
Cu`ÿS (x)	SALTO EGIN M(X,20:39)	E Jarraitu exekuzioan eskuineko instrukzioan parekatu Selectron kokapenean x	
CcÿS( x)	JAUZTU+ M(X,0:19)	F AC-ko zenbakia $\geq 0$ bada, jarraitu #D-n bezala. Bestela, jarraitu normal.	
Cc`ÿS( x)	JAUZTU+ M(X,20:39)	10 AC-ko zenbakia $\geq 0$ bada, jarraitu #E-n bezala. Bestela, jarraitu normal.	
AtÿS(x)	STOR M(X) )	11 Kopiatu AC zenbakia Selectron x kokapenean	
ApÿS( x)		12 Ordezkatu ezkerreko eskuineko 12 bitak instrukzioa Selectron kokapenean x eskuineko 12 AC-aren zatiak	
Ap`ÿS( x)		13 #12ren berdina baina eskuineko instrukzioa aldatzen du	
I	LSH	14 Mugitu AC zenbakia ezkerrera 1 bit (bit berria eskuina 0 da)	
R	RSH	15 Mugitu AC zenbakia eskuinera 1 bit (ezkerreko bit-a da kopiatu)	

Instrukzioainstrukzioa yam yam		Op Deskribapena	Izena eman Transferentzia Hizkuntza (RTL)
gelditu	0	Programa geldiarazi (ikus IAS txostenaren 6.8.5 paragrafoa)	

- Argibide multzoa (William Stalling)

**Table 2.1** The IAS Instruction Set

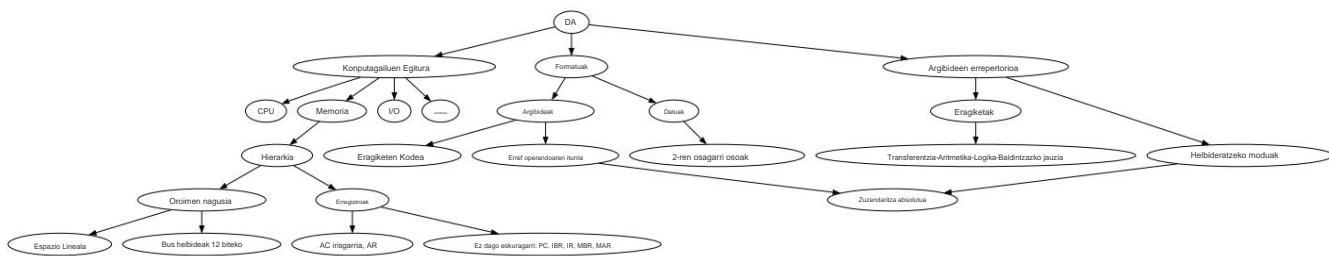
Instruction Type	Opcode	Symbolic Representation	Description
Data transfer	00001010	LOAD MQ	Transfer contents of register MQ to the accumulator AC
	00001001	LOAD MQ,M(X)	Transfer contents of memory location X to MQ
	00100001	STOR M(X)	Transfer contents of accumulator to memory location X
	00000001	LOAD M(X)	Transfer M(X) to the accumulator
	00000010	LOAD -M(X)	Transfer -M(X) to the accumulator
	00000011	LOAD  M(X)	Transfer absolute value of M(X) to the accumulator
	00000100	LOAD - M(X)	Transfer - M(X)  to the accumulator
Unconditional branch	00001101	JUMP M(X,0:19)	Take next instruction from left half of M(X)
	00001110	JUMP M(X,20:39)	Take next instruction from right half of M(X)
Conditional branch	00001111	JUMP + M(X,0:19)	If number in the accumulator is nonnegative, take next instruction from left half of M(X)
	00010000	JUMP + M(X,20:39)	If number in the accumulator is nonnegative, take next instruction from right half of M(X)
Arithmetic	00000101	ADD M(X)	Add M(X) to AC; put the result in AC
	00000111	ADD  M(X)	Add  M(X)  to AC; put the result in AC
	00000110	SUB M(X)	Subtract M(X) from AC; put the result in AC
	00001000	SUB  M(X)	Subtract  M(X)  from AC; put the remainder in AC
	00001011	MUL M(X)	Multiply M(X) by MQ; put most significant bits of result in AC, put least significant bits in MQ
	00001100	DIV M(X)	Divide AC by M(X); put the quotient in MQ and the remainder in AC
	00010100	LSH	Multiply accumulator by 2; that is, shift left one bit position
	00010101	RSH	Divide accumulator by 2; that is, shift right one position
Address modify	00010010	STOR M(X,8:19)	Replace left address field at M(X) by 12 rightmost bits of AC
	00010011	STOR M(X,28:39)	Replace right address field at M(X) by 12 rightmost bits of AC

24. irudia IAS\_Instrukzio\_multzoa

#### 2.4.3. ISA interfazea

- Instruction Set Architecture (ISA) Hardwarearen eta Softwarearen arteko INTERFAZEA definitzen du makinarena
  - ŷ Bi CPU guztiz desberdinak izan ditzakegu, adibidez AMD eta Intel, baina ISA bera badute makina bateragarriak izango dira sistema eragilearen ikuspuntutik.
  - ŷ Familia kontzeptua: jarraibideen erreperitorio bera desberdinek exekutatu dezakete ordenagailuak

- Makina ezberdinen ISA

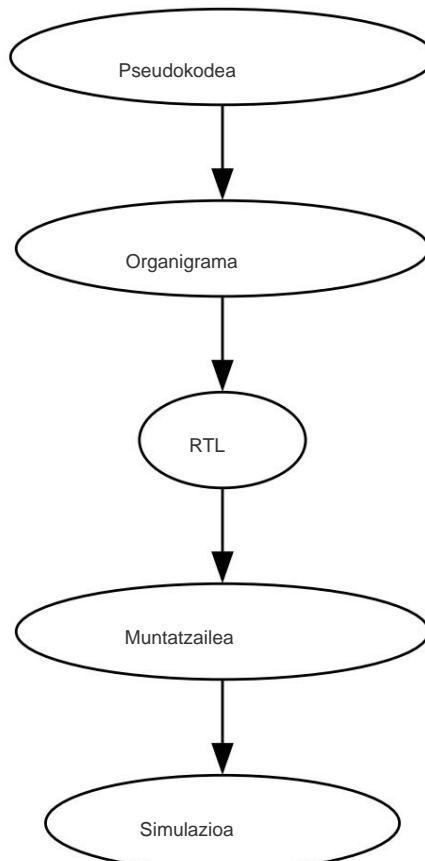


25. Irudia Instruction Set Architecture (ISA)

## 2.5. IAS Mihiztadura Lengoaiaren Programazioa

#### 2.5.1. Muntaia Lengoaian Programa bat garatzeko estrategia

- Programazioaren bidez konpondu beharreko arazoa ulertu ondoren, ez da programatzenean iturburu-modulua arazoaren soluzioa, baizik eta deskribatuz konpontzen da problema eta konponbide algoritmoa hizkuntza ezberdinetan eta fase hauetan:
    - ŷ Algoritmoaren deskribapena "pseudokode" hizkuntzan.
    - ŷ Algoritmoaren deskribapena organigrama edo fluxu-diagrama erabiliz.
    - ŷ Algoritmoaren deskribapena RTL erregistro-transferentzia hizkuntzan.
    - ŷ Algoritmoaren deskribapena ordenagailuaren berezko muntaia-lengoain



## 26. Irudia Programazio faseak

ÿ Goi-mailako hizkuntza batzuk deskribapenetikotako hizkuntza batzuen hizkuntza batzera igorotzen urtean egiten da

makina. Goi-mailako instrukzio bakoitza behe-mailako instrukzioen bloke batera itzuli beharko da.

## 2.5.2. Kodeketa Binario-Hamaseimala

- Exekutatu programa python hizkuntzan:

```
# Hamartar-binario-hamaseitar taula

i tartean ( 256 ):ÿ
inprimatu(str(i)+" 0b"+'{:b}'.format(i).zfill(8))ÿ
inprimatu(str(i)+" 0b"+'{:9_b}'.format(i)+" 0x"+'{:_x}'.format(i))ÿ inprimatu()
```

- Zifra bitarrak 2 dira: 0 eta 1
- zifra hamaseitarrak 16 dira: 0-1-2-3-4-5-6-7-8-9-ABCDEF. 0-1-2-... balioei dagozkienak 15 balioraino.
- 9 balioa bitarrean 1001 gisa adierazten da, non zifra bakoitzaren pisua bere arabera zehazten den.  
posizioa: 23 , <sup>22</sup>, <sub>2+0\*21+1\*20</sub> hogei  
ÿ 1001 zenbakiaren balioa zenbakiaren zifren batura hantzatua da bere pisuarekin:  $1*23+0*2$
- Hamasetarrez
  - ÿ 0xA zenbakiak 10, 0xB 11, 0xC 12 balio du
  - ÿ 0x10 zenbakiak  $1*161+0*160 = 16$  balioa du
  - ÿ 0xFD zenbakiak  $15*161+13*160 = 240+13 = 253$  balioa du
- Erlazio hamaseitar-bitarra
  - ÿ 0xF6 zenbakia bitar bihurtzen da zifra hamaseitar bakoitza 4 zifra bitarreko talde batean bihurtuz. F (15 balioa) 1111 bitarrean eta 6 (6 balioa) 0110 bitarrean. Beraz, 0xF6 zenbaki hamaseitarra 0b11110110 bitarrari dagokio.
- Byte 1eko zenbakien adibideak: hamartar-bitarra-hamaseitarra

```
16. 2 0b0001_0110 0x16 23 0b0001_0111 0x17 24 0b0001_1000 0x18 25 0b0001_1001
0x19 26 0b0001_1010 0b0001_1010
1x10 0x10 0x10 0x10 0x10 0 b0001_1100 0x1c 29 0b0001_1101 0x1d 30 0b0001_1110
0x1e 31 0b0001_1111 0x1f

192 0b1100_0000 0xc0 193 0b1100_0001 0xc1 194 0b1100_0010 0xc2 195 0b1100_0011
0xc3 196 0b1100_1100_0010 0xc2 0xc5 198
0b1100_0110 0xc6 199 0b1100_0111 0xc7 200 0b1100_1000 0xc8 201 0b1100_1001 0xc9
202 0xc9 202 0b10101000_0b10101000 1
0xcb 204 0b1100_1100 0xcc 205 0b1100_1101 0xcd 206 0b1100_1110 0xce 207 0b1100_1111 0xcf
```

## 2.5.3. 1. adibidea: sum1toN.ias

Adierazpena

- Kalkulatu batura  $\sum_{i=1}^N i = N(N + 1) / 2$

Pseudokodea

- Algoritmoaren deskribapena hizkuntza NATURALEko testu-adierazpenen bidez

- ALDAGAIAK:**

ŷ batura aldagaia: emaitza partzialak eta amaierakoak gordetzen ditu

ŷ N aldagaia: sarrerako datuak gordetzen ditu

ŷ i aldagaia: iterazio bakoitzean aldatzen den gehigarria gordetzen du

- Kode inperatiboaren egitura:

ŷ Oinarrizko instrukzioen eraikuntza begizta bat da

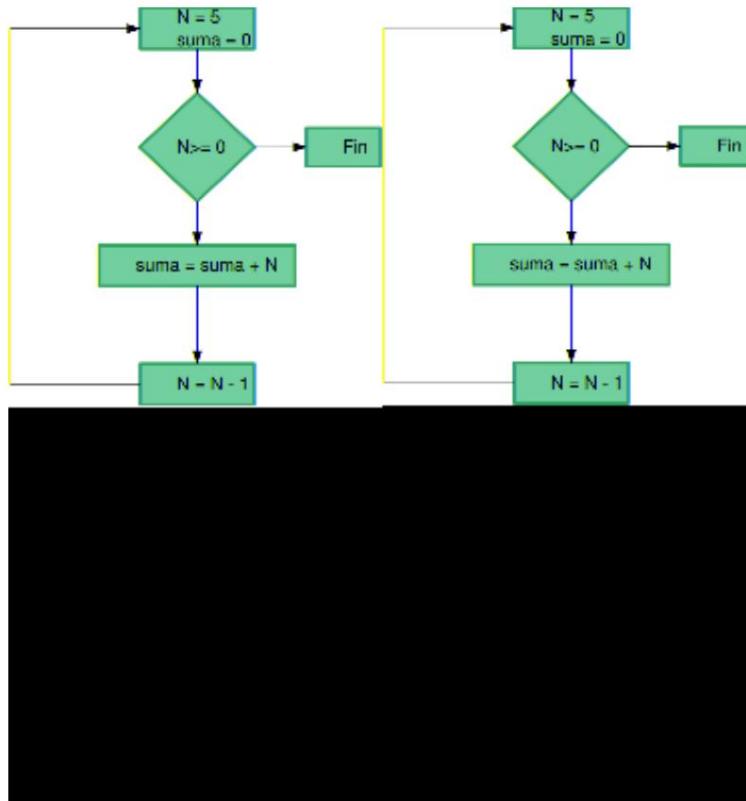
ŷ Begiztak iterazioak beherantz zenbatzen ditu

ŷ Iterazio bakoitzean "i" gehigarri bat sortzen da eta batura=bat+i egiten da

ŷ "i=N hasieratzen da eta iterazio bakoitzean i=i-1

ŷ Begiztak irteten da i=-1 denean

Organigrama



27. Irudia Organigrama: Fluxu-diagrama

RTL

- Metagailura zuzendutako IAS makinaren RTL deskribapena

:BURUKOA

; Sum1toN algoritmoaren deskribapena RTL hizkuntzan

:DATUEN ATALA:

; Etiketa-deklarazioa, kanpoko memoria-erreserba, hasieratzea

; Aldagai arruntak n: M[n] <-

5 ; aldagaien batuketa eta hasierako batura: M[batura] <- 0; batura partziala eta  
amaierako aldagaia

:JARRAIBIDEEN ATALA

;Metagailuetara zuzendutako arkitektura (AC)

;Erregistro eskuragarriak: AC ;

begizta hasiera: begizta gehigarrien batuketa eta sorrera: AC <- M[n];

karga gehitza

AC>=0 : PC <- gehitu ; gehigarria < 0 begiztaren amaiera bada eta > 0 badara jauzi  
Gehitu

; amaierako begizta gelditu;

egin

batuketa batuketa: AC <- AC +

M[batura]

M[batura] <- AC ;

eguneraztzea gehituz

AC <- M[n]

AC <- AC - 1

M[n] <- AC ;

hurrengo iterazioa

PC <- begizta

## IAS Machine WStalling Assembly Language

- sum1toN.ws iturburu-modulua

; OHARRA ;

1. bertsioa: sum1toN\_v1.ias; Zenbaki

osoen segida baten batura kalkulatzen du: batura = 1+2+..+n ; sarrerako datuak: n; irteerako

datuak: batura; Algoritmoa: n

iterazioko begizta

; Gehitzailaileak beheranzko norabidean sortzen dira na 0 Begizta irtengo da gehigarria

; negatiboa bada -> -1; Datuen egiturak: n eta batura aldagaiaik.

Konstante bat.

; Mihiztadura hizkuntza: William Stalling; Von Neumann

IAS makinaren arkitektura

;;;;;; JARRAIBIDEEN ATALA ; Metagailura

zuzendutako arkitektura (AC)

;Erregistro eskuragarriak: AC ;

algoritmoa: n, n-1, n-2,...0,-1 segida sortzen duen begizta n>=0 bada: LOAD M(n)

; AC <- M[n]

```

JMP + ; Gehitu      ; AC >= 0 bada, egin salto gehitzera
begiztaren amaiera
GELDITU          ; gelditu

; batuketa egin
gehitu: GEHITU M (batuketa)      ; AC <- AC+M (batura)
STOR ;      M (batuketa)      ; M[batura] <- AC
eguneraeztea gehituz
      ZARGA      M(n)      ; AC <- M[n]
AZPI        M (bat)      ; AC <- AC-M[bat]
STOR ;      M(n)      ; M[batura] <- AC
egiaztatu begizta egoera
JMP begizta ; baldintzarik gabeko jauzia

```

```

;:::::::::::::DATU ATAL
; Etiketa-deklarazioa, kanpoko memoria-erreserba, hasieratzea.
; Aldagai arruntak
n: .datuak 5 ; aldagaia gehituz
bat:      .datuak 1 ; cte
gain:     .datuak 0 ; batuketa partzialak eta azken emaitza

```

- ÿ Memoria erreserbatzeko datuen atala garatu da.
- ÿ BIZKIZ SIMPLE BAT egin da, begizta algoritmoan beharrezko eraikuntza baita finala.
- ÿ KENDURA eragiketa egin da azken algoritmoan beharrezkoan den eragiketa baita.
- ÿ Kodea IRUZKINATU da

iassim mutuaia-lengoiaia

Iturburu-moduluaren garapena mutuaia-lengoiaian EZ da hasieratik amaierara arte egiten baizik Urratsez burutua, probatuko den ahalik eta kode errazenetik hasita eta garatu aurretik arazketa kode osora iritsi arte

- sum1toN\_A.ias iturburu-modulua:

- ÿ 1. bertsioak begizta bat implementatzen du bere gorputzak datuak batura aldagaian soilik gordetzen dituena. Berak datuak aldatu egiten dira iterazio bakoitzean.
- ÿ Sintaxia ÿ etiketa: eragiketa eragiketa ; iruzkina ÿ 4 zutabe
- ÿ Hitz baten eskuinaldean dauden argibideak seinalatzen dituzten etiketak soilik egon daitezke.
- ÿ Eragiketa adierazteko ikurrak (Adb. S(x)ÿAc+) ez dira mnemoteknikoak
- ÿ Ez erabili azentu-markak iruzkinetan edo etiketetan, ASCII kodea bakarrik onartzen baita. ez luzatu.
- ÿ Instrukzio kopurua bakoitia bada, azken instrukzioko 40 biteko hitza bete behar da pisu txikieneko 20 bitekin zeroan ezarrita programako argibide kopurua lortzeko berdin izan: erabili .hutsik zuzentaraua

ÿ Argibideen atala datuen atalaren aurretik joan behar da

; OHARRA ;

Kode hau ARAZTU behar da; 1. bertsioa:

sum1toN\_A.ias; Zenbaki osoen segida

batuen batura kalkulatzen du: batura =  $1+2+..+n$  ; sarrerako datuak: n; irteerako datuak: batura;

Algoritmoa: n iterazioko begizta

; Gehitzaireak beheranzko norabidean sortzen dira na 0 Begizta irtengo da gehigarria negatiboa bada -> -1; ; Datuen egiturak: n eta batura aldagaiak.

Konstante bat.

; Mihiztadura hizkuntza: IASSim; Von

Neumann IAS makinaren arkitektura

;,:,:,; JARRAIBIDEEN ATALA ; Metagailura

zuzendutako arkitektura (AC)

;Erregistro eskuragarriak: AC ;

algoritmoa: n, n-1, n-2,...0,-1 segida sortzen duen begizta  $n \geq 0$  bada:  $S(x) \rightarrow Ac + n$  ; AC  $\leftarrow M[n]$

$S(x) \rightarrow Ah - bat$  ; AC  $\leftarrow AC - M[bat]$

At  $\rightarrow S(x)$  batura ; M[batura]  $\leftarrow AC$  Cc  $\rightarrow S(x)$  begizta ; AC

$\geq 0$  bada, salto egin begizta; begiztaren amaierako geldialdia

; gelditu

.hutsik

;,:,:,;DATU ATAL ; Etiketa-deklarazioa,

kanpoko memoria-erreserba, hasieratzea.

; Aldagai arruntak n: .datuak 5 ;

aldagaia gehituz .data 1 ; cte .data 0 ; batuketa

bat: partzialak eta azken

gain: emaitza

ÿ Memoria erreserbatzeko datuen atala garatu da.

ÿ BIZKIZ SIMPLE BAT egin da, begizta algoritmoan beharrezko eraikuntza baita finala.

ÿ KENDURA eragiketa egin da azken algoritmoan beharrezkoa den eragiketa baita.

ÿ Kodea IRUZKINATU da

ÿ FALTA:

ÿ Eragigaiak "n" memoria helbidean gordetzeko argibideak gehitu bakoitzean begizta iterazioa

ÿ Gehitu batuketa partziala egiteko argibideak begiztaren iterazio bakoitzean

ÿ Ikusi programaren bertsioa RTL hizkuntzan ():

#### JARRAIBIDEAK ATALA

begizta: AC  $\leftarrow M[n]$  ; karga gehitza

AC  $\geq 0$  : PC  $\leftarrow$  gehitu ; gehigarria  $< 0$  begiztaren amaiera bada eta ez bada salto

```

Gehitu
gelditu
; egin batura
gehitu: AC <- AC + M[batura]
M[batura] <- AC
; egunerasatzea gehituz
AC <- M[n]
AC <- AC - M[bat]
M[n]           <-AC
PC <- begizta ; baldintzarik gabeko jauzia

DATU ATALA
n: 5          ; aldagaiai gehitza eta hasieratza
bat:      1    ; aldagaiai gehitza eta hasieratza
batura: 0     ; batura partziala eta amaierako aldagaia

```

- Bertsio zuzendua

```

; OHARRA
; Kodea zuzendua
; Bertsioa: sum1toN_A.ias
; Zenbaki osoen segida baten batura kalkulatzen du: batura = 1+2+..+n
; sarrerako datuak: n
; irteerako datuak: batuketa
; Algoritmoa: n iterazioko begizta
;
; Gehigarriak na Orik beheranzko norabidean sortzen dira
; Begizta irtengo da gehigarria negatiboa bada -> -1
; Datuen egiturak: n eta batura aldagaia. Konstante bat.
; Mihiztadura hizkuntza: IASSim
; Von Neumann IAS makinen arkitektura

;::::::::::: JARRAIBIDEEN ATALA
;Metagailuetara zuzendutako arkitektura (AC)
;Erregistro eskuragarriak: AC
; algoritmoa: n, n-1, n-2,...0,-1 segida sortzen duen begizta n>=0 bada
begizta: S(x)->Ac+ n ; AC <- M[n]
Cc->S(x) gehitu ; gehigarria < 0 begizta amaiera bada eta ez bada gehitzera salto egin
; begiztaren amaiera
gelditu           ; gelditu
.hutsik
; egin batura
gehitu: S(x)->Ah+ batura At->S(x) ; AC <- AC + M[batura]
batura ; egunerasatzea ; batura <- AC
gehituz
S(x)->Ac+ n S(x) ; AC <- M[n]
>Ah- bat At->S(x) n ; AC <- AC - M[bat]
Cu->S(x) begizta ; n <- AC
; baldintzarik gabeko salto ezkerreko begiztaren instrukziora

;::::::::::: DATU ATAL

```

```
; Etiketa-deklarazioa, kanpoko memoria-erreserba, hasieratzea.
; Aldagai arruntak
n: .datuak 5 ; aldaia gehituz
bat: .datuak 1 ; cte
gain: .datuak 0 ; batuketa partzialak eta azken emaitza
```

- B bertsioa

```
; Bertsioa: sum1toN_B.ias
; Zenbaki osoen segida baten batura kalkulatzen du: batura = 1+2+..+n
begizta: S(x)->Ac+ n ;kargatu n AC sartu
S(x)->Ah+ batura ;n gehitu baturari
At->S(x) batura ;guztira itzuli batura
S(x)->Ac+ n ;kargatu n AC sartu
S(x)->Ah- bat ;gutxitu n
At->S(x) n ;denda dekrementatua n
Cc->S(x) begizta ;AC >= 0 bada, jauzi pos eskuineko instrukziora
gelditu ;bestela eginda
```

```
n: .data 5 ;guztira 6 aldiz biraka egingo du
bat: .datuak 1 ;gutxitzeko konstantea n
batura: .data 0 ;non exekutiboa/azkeneko guztirakoa gordetzen den
```

## Erregistroak

- IAS makinak 7 erregistro ditu: Metagailua, Erregistro aritmetikoa / Kooziente biderkatzailea (AR/MQ), Kontrola Kontagailua, Kontrol Erregistroa, Funtzio Taularen Erregistroa, Memoria Helbideen Erregistroa, Selectron Erregistroa
  - ÿ Metagailua (AC) eta Erregistro aritmetikoa (AR/MQ) programatzaleak ikusgai dauden bi bakarrak dira. erregistroak
  - ÿ Kontrol-kontagailua gaur egun Programa-kontagailua deitzen dioguna da
  - ÿ Kontrol Erregistroak unean exekutatzen ari den IBR instrukzioa gordetzen du. ren instrukzioa bakarrik exekutatzeko eskubidea.
  - ÿ Funtzio-taularen Erregistroak uneko IR opcodea gordetzen du
  - ÿ Memoriaren helbidea Erregistratu uneko memoria helbidea MAR
  - ÿ Selectron Erregistratu memoriatik irakurtzen edo idazten ari den uneko datu-balioa ÿ MBR

## IASSim simulagailua

- Von IAS Machine IASSim Simulator Instalatzeko eta Funtzionatzeko Argibideak Neumann [Franskinean](#)

### Kalifikazioak

- Beharrezkoa da argibide kopurua bikoitia izatea. Bitxia bada, .empty zuzentaraua gehitzen da.
- Etiketa batek ezkerreko instrukzioa adierazi behar du. Eskuinean badago, argibide bat jar daiteke haren aurretik. baldintzarik gabeko salto esandako etiketara.
- Datuen atala kodearen atalaren ondoren badago, kodearen atalak amaitu behar du eskuineko instrukzio batekin eta ez bada .hutsik zuzentarauekin betetzen dugu.

## Akatsa

- Errore bat MAR erregistroaren balioa bistaratzean
  - ÿ Sum1toN.ias-en lehen instrukzioa exekutatzen denean, MAR edukia 28 da, tartea baino handiagoa. programa kargatzen den memoria nagusiaren helbideak.
  - ÿ Errorea Windows 7 eta Ubuntu 17.04-n gertatzen da

#### 2.5.4. IASSim hizkuntzako programen adibideak

- Adibide gehiago Eranskinean .

### 2.6. IAS Makinaren Eragiketa: Datuen Bidea

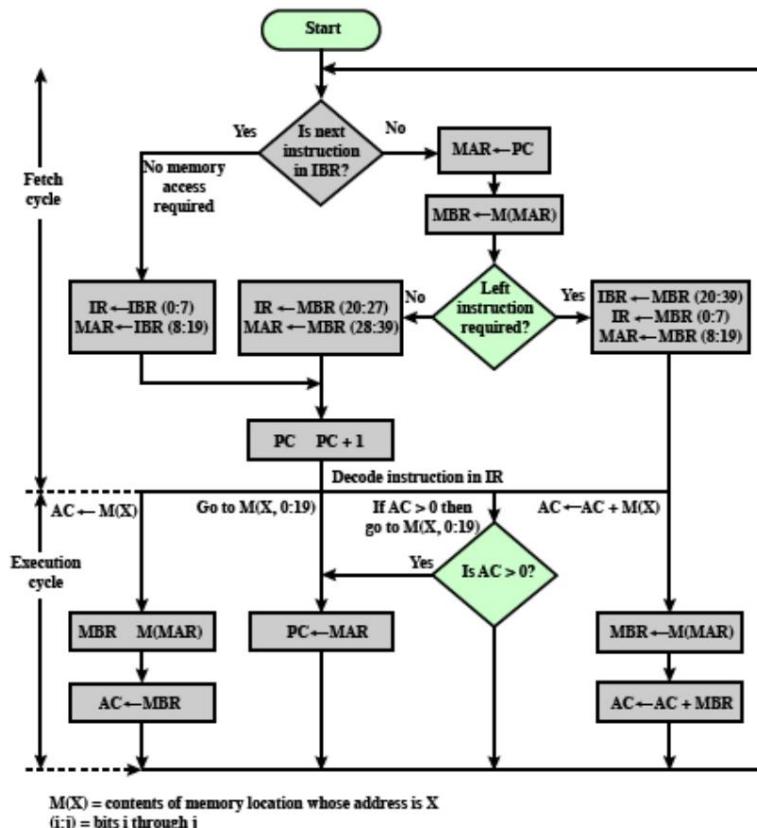


Figure 2.4 Partial Flowchart of IAS Operation

#### 28. Irudia IAS Eragiketa

- IAS makinaren funtzionamendua:
  - ÿ Irakaskuntza-zikloak bi FASE ditu
  - ÿ Lehenengo fasea argibide guztientzat komuna da.
- Argibideen adibideak
  - ÿ X: eragigaien erreferentzia
  - ÿ AC ÿ M(X)
  - ÿ GOTO M(X,0:19): Baldintzarik gabeko saltoa X helbidera. X-k bi argibide seinalatzen ditu. X,0:19 da Ezkerreko Instrukzioaren erreferentzia.
  - ÿ AC>0 M(X,0:19) joan bada: baldintzazko jauzia

ÿ AC ÿ AC+M(x).

## 2.7. Ondorioak

1. Behe-mailako programaziorako ordenagailuaren ISA arkitekturaren ezaugarri nagusiak ezagutzea beharrezkoa da: ordenagailuaren Egitura (memoria, erregistroak, etab.), Datuen formatua (2 osagarriaren formatua, etab.) eta argibideak eta Irakaskuntza Errepertorioa (eragiketak, helbideratze moduak, etab.)
2. Mihiztadura-lengoaian programazioa ez da zuzenean hizkuntza horretan egiten, goitik beherako estrategia bat jarraitzen da, pseudokode hizkuntzan, organigraman, RTL hizkuntzan, etab. deskribapen batekin hasita.
3. Ordenagailuaren ISA argibide multzoaren diseinua da, maila baxuko programazioa errazten edo zaitzen duena. Von Neumann-en IAS makina bezalako errepetorio gehiegi mugatu batek zaila egiten du biderketa eta zatiketa bezain sinpleak diren adierazpen matematikoak egitea. RTL instrukzio-sekuentziak algoritmoaren garapena erraztea kontuan izan behar du.
4. Algoritmoaren programazioak mutua-lengoaian behetik gorako estrategia bat jarraitzen du garatu nahi den programaren bertsio osatugabe eta errazenetik hasita.
5. Mihiztadura-lengoaiaren programaren bertsio garatu bakoitzarazketa eta egiatztu behar da emaitza partzialak aztertzeko urratsez urratseko moduan exekutatzeko aukera ematen duen simulagailu eta arazte baten bidez.

# 3. kapitulua. Datuen irudikapena

## 3.1. Ikastaroa

- 3. Datuen irudikapena
  - a. Bit, Byte eta Word
  - b. Pertsonaiak, zenbaki osoak eta errealkak

## 3.2. Helburua

- Datu alfanumerikoen irudikapena makina-lengoaian, hau da, kode bitarran.
- Testu liburua W.Stalling
  - ÿ 3. zatia, 9. kapitulua: Zenbaki-sistemak

## 3.3. Datuak eta argibideak: Kodeketa bitarra

- Memoria nagusian gordetako programa bat makina-lengoaian irudikatzen da eta datu eta argibidez osatuta dago. Makina-lengoaia 0 eta 1 sinboloeak osatzen duten lengoaia bitarra da. Horregatik, memoria nagusian gordetako programa baten datuak eta argibideak bi ikur hauek erabiliz kodetu eta irudikatu behar dira.
- Datuek zenbakizko balio bat dute, Unitate Logiko Aritmetikoa (ALU) prozesatu dezakeena, batuketa, kenketa eta abar bezalako eragiketa aritmetikoak egiteko edo ez, edo, etab., eragiketa logikoak egiteko.
- Datuak memorian gordetako 0 eta 1eko sekuentziak dira CPUak atzematen dituena batuketa bezalako eragiketa aritmetikoen bidez prozesatu pej.
- Instrukzioak memorian gordetako 0 eta 1eko sekuentziak dira, CPUak interpretatu eta exekutatzeko harrapatzen dituena. Pej instrukzioa "gehitu" bi zenbaki osoak.

## 3.4. Bit, Byte, Word

- BINARY DIGIT (bit): zifra bitarrak 0 eta 1 dira. Biak kodetzeko erabiltzen diren sinboloak dira. jarraibideak memorian gordetako programa-datu gisa.
- Byte: 8 zifra bitarreko segida bat da. Adibidea: 00110101
- Hitza: 8ren multiploko zifra bitarren segida da, hau da, byte baten multiploa. Konputagailuen arkitektura-ingurunean, prozesatzeko unitate zentrala (CPU) memoria nagusiarekin konektatzen duen datu-busaren bit-kopurua bezala definitzen da eta PUZaren barne-memoriaren helburu orokorreko erregistroen zabalera ere izan ohi da.

ÿ Linux +sudo lshw -C sisteman | gehiago + ÿ zabalera: **64 bit**

lur

Deskribapena: Koadernoaren

produktua: 20F1S0H400 (LENOVO\_MT\_20F1\_BU\_Think\_FM\_ThinkPad L560) fabrikatzailea:

LENOVO bertsioa: ThinkPad L560

seriea: MP15YSW7

zabalera: 64 biteko

gaitasunak: smbios-2.8 dmi-2.8 smp vsyscall32

konfigurazioa: **administrator\_password=desgaituta dagoen txasis=koaderno familia =ThinkPad L560 power-on\_password=desgaituta dagoen sku**

=LENOVO\_MT\_20F1\_BU\_Think\_FM\_ThinkPad L560 uuid=4C2F45AA-0A2C-B211-A85C  
B5C56EB5BBAC

## 3.5. Zenbaki osoak

### 3.5.1. Oinarri hamartarra

- Hauetan oinarritutako Posizio Sistema baten bidez adierazten dira:

ŷ Zenbakia oinarri hamartarran

ŷ Hamar zifra konbinatuz irudikatzea: 0,1,2,3,...,9

ŷ Posizioa ŷ indizea

ŷ Posizio bakoitzaren pisuak ŷ 10posizio formako potentziak dira ŷ ..., Ehunak, hamarrak, unitateak

ŷ Adierazitako balioa = posizio pisuarekin hiztatutako zifren batura

ŷ Adibidea: zenbaki hamartar oso baten 1197 irudikapena emanda, kalkulatu haren balioa.

Posizioa				0
Pisua	3 10^3	2 10^2	1 10^1	10^0
	1000	100	10	1
Zifra		1	9	7
Pisatzea	1 1*1000	1*100	9*10	7*1

Merezi  $1*1000+1*100+9*10+7*1=$ mila ehun laurogeita hamazazpi.

Bat-bat-bederatzi-zazpi irudikapenak mila ehun eta laurogeita hamazazpi balioa du.  
zazpi

### 3.5.2. Oinarri bitarra

- 2. oinarri kodetutako zenbakia

ŷ Bi Zifra konbinatuz irudikatzea: 0,1

ŷ Posizioa ŷ indizea

ŷ Posizio bakoitzaren pisuak ŷ 2 posizioko potentzia dira ŷ ... 25,24,23,22,21,20 ...,64,32,16,8,4,2,1

ŷ Adierazitako balioa = beren pisu posizioarekin hiztatutako zifren batura

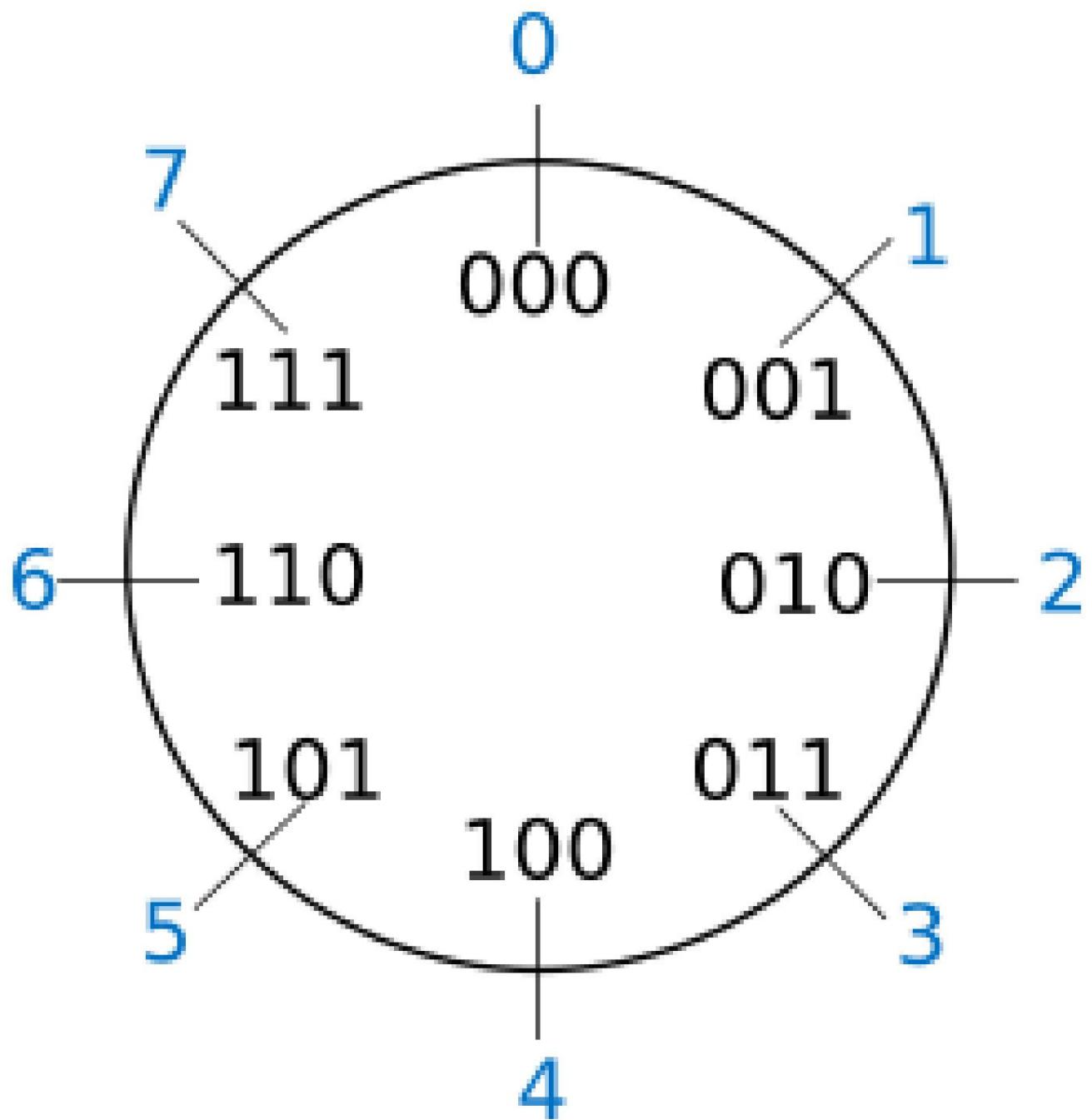
ŷ Adibidea: zenbaki oso baten 1010 irudikapena emanda, kalkulatu haren balioa.

Posizioa	3	2	1	0
Pisua	23	2^2	2^1	2^0
	8			1
Zifrak	1	40	—	0
Pisatzea	1*8	0*4	1*2	0*1

Merezi  $1*8+0*4+1*2+0*1 =$ hamar

Bat-zero-bat-zero irudikapenak hamar balioa du

- Gurpila: 3 biteko zeinurik gabeko zenbakien irudikapena eta balioa.



29. Irudia Sinatu gabeko zenbakien irudikapena

### 3.5.3. Bihurketa hamartar-bitarra

- Ondoz ondoko zatiketak / 2 ÿ Dibidendua<sub>1</sub> = 2\*Kozientzia<sub>1</sub> + hondarra<sub>1</sub>

$$\text{Hondarra}_1 * 2^0 + \bullet \text{Zatidura}_1 = 2 * \text{Zoziente}_2 + \text{Hondarra}_2 \quad \text{Dibide} \# 1 \text{Kozientzia}_2 + \text{Hondarra}_1 = \\ \text{Gainerakoa}_2 * 2^1 + \text{Koziente}_2 * 2^2$$

- hondarra<sub>1</sub> 0 posizioko zifra bitarra da, hondarra<sub>2</sub> 1 posizioko zifra bitarra da, zatidura<sub>2</sub> posizioko zifra bitarra.

- Araua: zifra bitarrak hondar guztiak eta azken zatidura dira.

- Zatiketa zatidura 2z zatigarria ez denean amaitzen da, hau da, zatidura 1 da. Zatidura hau MSB da.

- Adibidea: 1197 balioa ÿ Kalkulatu bere irudikapena kode bitarrean ÿ Soluzioa: 10010101101

4. taula. Bitar-bihurketa hamartar

zenbakia ro	1. Div		2. Div		3. Div		4. Div		5. Div		6. Div	
	Kok	Coc Atseden	Atsedena									
1197 598	1	299	0	149	1	74	1	37	0	18	1	

Zenbakia 7. Div	8. Div			9. Div			10. Div		
	Kok		Atsedena	Kok		Atsedena	Kok		Atsedena
1197	9	0	4	1	2	0	1	0	0

### 3.5.4. Oinarri octala

- 8. oinarria
- Zifrak: 0,1,2,3,4,5,6,7
- Pisuak: 8 posiziora igota
- C-n oinarria 0 ÿ [int 077](#) aurrizkiarekin zehazten da ;
- Octal ÿ Bitar bihurketa eta alderantziz ÿ zortziko zifra bakoitza 3 biteko bitar batean deskonposatzen da
- hamartar 1197 ÿ Kalkulatu bere irudikapena zortziko kodean.
  - ÿ soluzioa a) 10010101101 bitarra ÿ 02255 zortzikoa
  - ÿ soluzioa b) 8 oinarriaren ondoz ondoko zatiketak.

#### Oinarri hamaseitarra

- 16. oinarria
- Zifrak: 0-1-2-3-4-5-6-7-8-9-ABCDEF
- Pisuak: 16 posiziora igota
- C-n oinarria 0x ÿ [int 0xAF](#) aurrizkiarekin zehazten da ;
- Hamasetar ÿ Bitarra eta alderantziz ÿ zifra hamaseimal bakoitza 4 biteko bitar batean deskonposatzen da
- hamartar 1197 ÿ Kalkulatu bere irudikapena kode hamaseimalean
- Soluzioa a) 10010101101 bitarra ÿ 0x4AD
- Soluzioa b) 16 oinarriaren ondoz ondoko zatiketak.

### 3.5.5. Kalkulagailua

- Kalkulagailua Linux sisteman
  - ÿ `candido@lur:~$ echo "obase=2 ; ibase=16; 80AA010F" | bc.`
  - ÿ 100000001010101000000100001111
  - ÿ `echo "obase=10; ibase=16; 80AA010F" | bc` ÿ derrigorrezkoa da formatuaren oinarria lehenik jartzea irten
  - ÿ 2.158.625.039
  - ÿ `$bc` interpretea

### 3.5.6. Python

- <https://docs.python.org/3/tutorial/index.html>

ÿ lagunza (eraikitakoak)

```
bin (1197) -> '0b10010101101'
oct (1197) -> '02255'
hex (1197) -> '0x4ad'
int (0x4ad) -> 1197
```

### 3.5.7. Zenbaki oso sinatutakoak

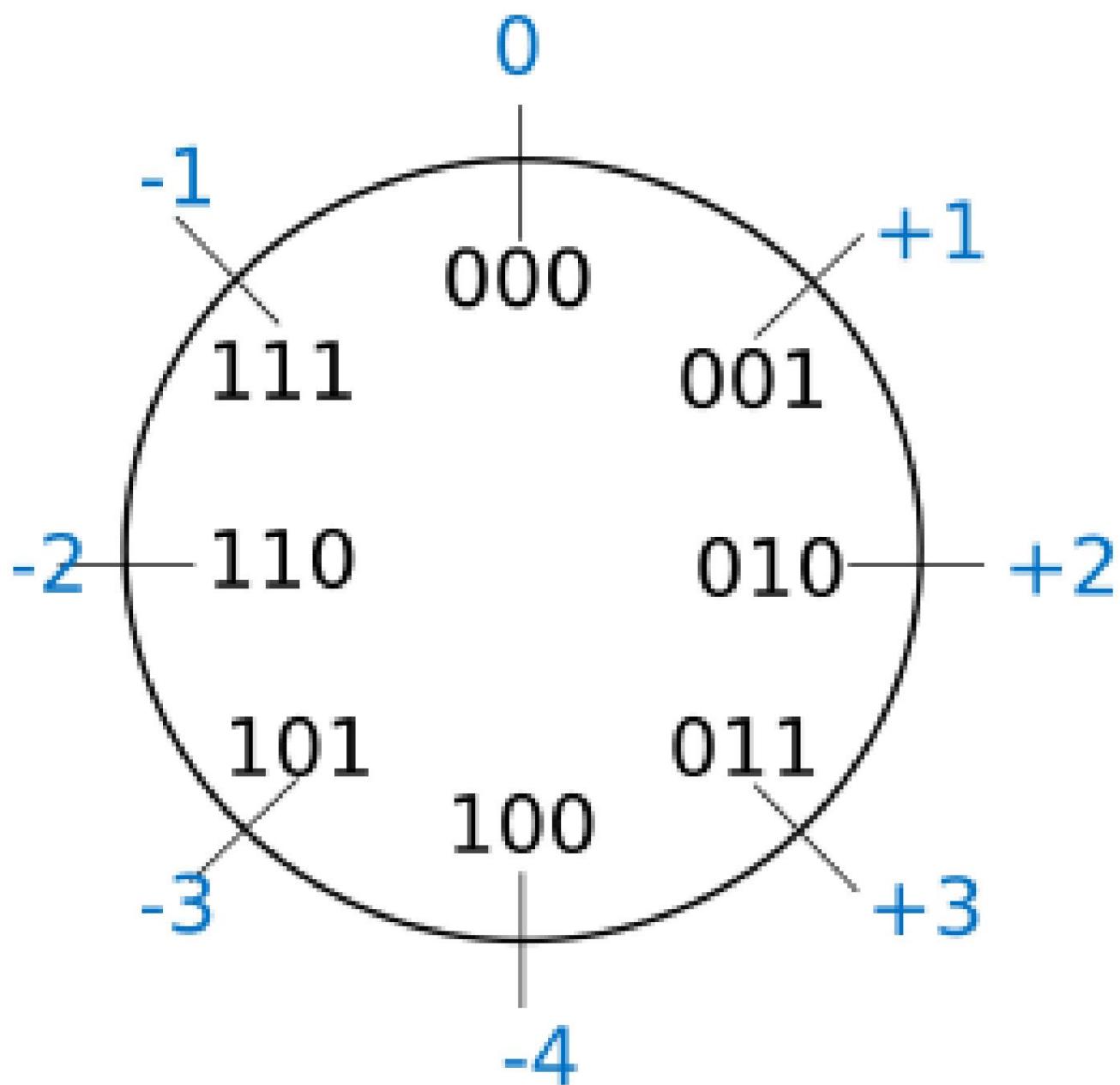
- Bi formatu aztertuko dira: Zeinu-Magnitudea eta 2-ren osagarria, azken hau hedatuena. ordenagailuen arkitekturan.

#### Zeinu-Magnitude

- Zeinu-Magnitude formatua
  - ÿ Bit esanguratsuenak ez du baliorik, zeinua adierazten du: zero zenbaki positiboentzat eta batentzat negatiboak.
  - ÿ Gainontzeko bitek zenbaki osoaren modulua adierazten dute
  - ÿ Adibidea:
    - ÿ Balioa +1197 ÿ 010010101101 irudikapena
    - ÿ Balioa -1197 ÿ 110010101101 irudikapena
    - ÿ Nola adierazten da zero balioa?

#### 2-ren osagarria

- 2-ren osagarri formatua.
  - ÿ Positiboak: zeinu-magnitude formatuaren berdina: MSB Bit= 0. Pisuak: 2posizio- potentzia .
    - ÿ Bit esanguratsuagoa (MSB) ÿ zero ez den balio duen posiziorik altuena.
    - ÿ Bit esanguratsu gutxiago (LSB)
  - ÿ Negatiboak: Magnitude berdina baina positiboa duen zenbakiaren transformazioa funtzioa erabiliz 2ren osagarria.
- Gurpila: 3 biteko zenbaki sinatutakoien irudikapena eta balioa.



30. Irudia 2-ren Osagarrien Adierazpena

ÿ N: zenbakiaren bit kopurua: 3 bit

ÿ Zatitu zirkunferentzia konbinazio bitar posibleen kopurutan:  $2N : 2^3$

ÿ Erlojuaren orratzen noranzkoan konbinazio bitar guztiak sekuentzialki margotzen ditut:  
000.001.010.011,

ÿ Balioak txandaka margotzen ditut: 0, +1, -1, +2, -2, ...

ÿ Ariketa: Adierazi +4 zenbaki positiboa 2 osagarrian

- Ondorioak:

ÿ Eremu positiboaren eta negatiboaren arteko asimetria

ÿ Zerok irudikapen bakarra du

ÿ Zenbaki negatiboak 1etik hasten dira

ÿ -1 balioa 1111111111111111 zifra guztiekin kodetuta dago

ÿ Zeinu luzapena: ezkerreko 1 bat ezkerreko zero bat bezalakoa da positiboetan: ez du balio eta ezkerreko 1 errepikapena ezabatu daiteke, azken 1a esanguratsuena utziz. 11110111 10111ren baliokidea da.

- 2-ren osagarriaren funtzioa: zenbaki oso baten 2-ren osagarria bere zeinua aldatzearen baliokidea da. 2 osagarriko zenbaki oso positibo eta negatiboen arteko bihurketa metodo desberdinak erabiliz egin daiteke.
- X osoko bitarraren 2 osagarria lortzeko adibideak:
  - to. 1. metodoa: egin X-ren eragiketa logikoa osagarria (ezeztapena) eta gehitu 1  $\ddot{y} \sim X+1$ 
    - $\ddot{y} X=0101 + 5$  balioa du 2 osagarrian
    - $\ddot{y} 0101$ aren 2 osagarria?  $\sim 0101 + 1 = 1010 + 1 = 1011 = -5$   $\ddot{y}$  2 osagarriaren balioa zeinua aldatzearen baliokidea da.
    - $\ddot{y} X=1111 - 1$  balioa du 2 osagarrian
    - $\ddot{y} 1111$ ren 2 osagarria?  $\sim 1111 + 1 = 0000 + 1 = 0001 = +1$
    - $\ddot{y} X=0110011100010101010000$   $\ddot{y}$  positiboa bit esanguratsuena (MSB) zero delako
    - $\ddot{y} C2(X)=1001100011101010101111+1=1001100011101010110000$   $\ddot{y}$  negatiboa bit duelako esanguratsuena (MSB) bat
  - b. 2. metodoa: X kodearen 0 posiziotik (bit
    - $\ddot{y} X = 0110011100010101010000$   $\ddot{y}$  guztira 22 bit
    - $\ddot{y} X$ -ren 1 lehen zifra 4 posizioan dago  $\ddot{y} 01100111000101010-10000$   $\ddot{y}$  kopiatu 5 lehenengo zifrak eta alderantzizkatu gainerako 17ak
    - $\ddot{y} C2 (X) = 10011000111010101-10000$
  - c. 3. metodoa: Egin 0-X eragiketa aritmetikoa
    - $\ddot{y} X = 0110011100010101010000$
    - $\ddot{y} 0-X=000000000000000000000000000000 - 0110011100010101010000 = 1001100011101010110000$
- Adibideak
  - $\ddot{y}$  Adierazi -1197 zenbaki oso negatiboa zeinu-magnitudean eta 2 osagarrian
    - $\ddot{y} +1197 = 010010101101$  zeinu-magnitudean eta 2 osagarrian
    - $\ddot{y} -1197 = 101101010011$
  - $\ddot{y}$  Kalkulatu 2 osagarrian 8 bit dituzten zenbaki osoen tarteak
    - $\ddot{y}$  Gehieneko kode positiboa: 01111111  $\ddot{y}$  Balioa = 27 -1
    - $\ddot{y}$  Gutxieneko kode negatiboa: 10000000
    - $\ddot{y} C2(n=10000000) = 01000000 = 27$ , orduan n=10000000 -27 balioa du  $\ddot{y}$  Barrutia [-27,+27 -1]

### 3.6. Zenbaki errealkak

#### 3.6.1. Koma finkoa

- Zenbaki errealkak puntu finkoan:
  - $\ddot{y} 1234.56789$
  - $\ddot{y}$  Posizio-sistema
    - $\ddot{y}$  zatiki zifren posizioa: -1,-2,-3,...  $\ddot{y}$  zatiki zifren
    - pisuak: 10-1, 10-2, 10-3  $\ddot{y}$  pisua 1234,56789 =
 
$$1*10^3+2*10^2+3*10^1+4*10^0+5*10^{-1}+6*10^{-2}+7*10^{-3}+8*10^{-4}+9*10^{-5}$$
  - Oinarri Bitarra

ÿ 1010.101  $\cdot 1^{*} 23 + 0^{*} 22 + 0^{*} 21 + 1^{*} 20 + 1^{*} 2 - 1 + 0^{*} 2 - 1 + 1^{*} 2 - 2$  ÿ 10.625

### 3.6.2. Koma flotagarria

#### Formatua

- Koma flotagarria ÿ Notazio zientifikoa

ÿ -23.4567E-34 edo -23.4567\*10-34

ÿ Mantisa edo **esanahia** potentzia biderkatzen duen zenbakia da ÿ -23,4567

ÿ Mantisa **normalizatua** : mantisak osoko zati gisa dauka beste zifra bateko zenbaki oso bat. zero.ÿ -2,34567\*10-33

ÿ mantisa normalizatuaren zati osoa: 2

ÿ mantisa normalizatuaren zati zatia: 0,34567

ÿ **Berretzailea** potentziaren oinarria altxatzen den zenbaki osoa da. Lekuaren araberakoa da mantisan jan. Kasu honetan -33 da.

ÿ **Oinarria** boterearen oinarria da. Adibide honetan 10 da.

- Kodetze bitarra

ÿ Adibidea: 1234.56789

ÿ Zati osoa: 1234 ÿ 10011010010

ÿ Zati zatia: 0,56789

$$\begin{aligned}
 & 0,56789 * 2 = 1,13578 - 1 + 0,13578 - 1, \text{ posizio bit } -1,13578 * 2 = \\
 & 0,27156 - 2 = 0,27156 - 0,27156 * 2 = 0,54312 \rightarrow 0, \\
 & \text{posizioa -3 bit } 0,54312 * 2 = 1,08624 = 1 + 0,08624 \\
 & \rightarrow 1, \text{ posizio bit } -4
 \end{aligned}$$

ÿ frakzio biribildua 0,1001

ÿ zatiki biribildugabea 0,10010001011000010

ÿ frakzio biribildua 0,100100011

ÿ Kode binarioko puntu finkoa: 10011010010.10010001011000010 ÿ Notazio

cientifikoa: 1.001101001010010001011000010\*2+10 ÿ koma mugikorra ÿ zati osoa beti 1 balio du.

#### Zehaztasuna

- Zifra esanguratsuen kopurua da

- q zenbakia  $p \neq 0$  zenbakiaren hurbilketa dela esaten da, gutxienez **m** zifren zehaztasunarekin. b oinarrian esanguratsua, baldin eta errore erlatiboa  $|pq|/p$  ÿ  $0,5^{\ast}b-m+1$

ÿ m aurreko desberdintasuna betetzen duen zenbaki osorik handiena denean, q gutxi gorabeherakoa dela esaten da. p m zifra esanguratsuekin.

- Adibidea

to.  $p = 1E0$  eta  $q = .9999E0$  ÿ Errorre erlatiboa=0.1E-3<0.5E(-4+1) ÿ 4 zifra esanguratsuren zehaztasuna

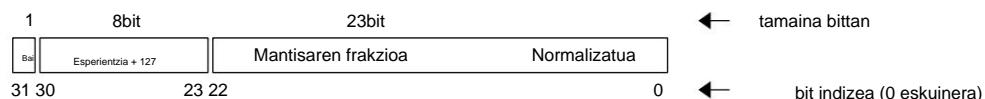
b. Kalkulagailu batek, A, 2. oinarrian funtzionatzen du 22 biteko mantisarekin eta beste batek, B, 16. oinarrian 6rekin. doitasun-digituak (24 bit). Bietatik zein da zehatzagoa?

## IEEE-754 estandarra

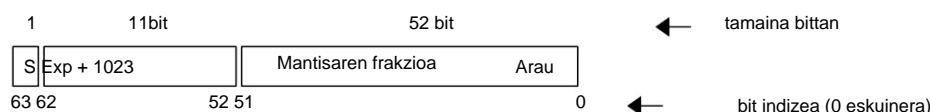
- Flotatzalea

ÿ IEEE-754 estandarra

ÿ Doitasun bakarra ÿ 32 biteko formatua 3 eremutan Zeinua/Berretzailea/Frakzioa  
luzerak 1/8/23 bit



ÿ Doitasun bikoitza ÿ 64 biteko formatua 3 eremutan Zeinua/Berretzailea/Frakzioa  
luzerak 1/11/52 bit



ÿ [eskuzko bihurketa](#)

ÿ [wikia](#)

ÿ Bitarra: Hiru eremu



ÿ Balioa (-1)Zeinua x 1. Mantisa\_Normalizatua\_Frakzioa x 2 Berretzailea

ÿ Zeinua: positiboa -ÿ 0 bit, negatiboa --ÿ 1 bit

ÿ Gehiegizko berretzailea: 127 (zehaztasun bakarra) edo 1023 (zehaztasun simplea) gehitzen zaion berretzailea da.  
bikoitza)

ÿ Mantisa normalizatua: mantisa da, bere zati osoa 1 den

ÿ Mantisa normalizatuaren frakzioa: Mantisa normalizatuaren frakzioa da.

## Kode bihurgailuak

- Sareko bihurgailuak:

ÿ [Bihurgailu bitarra](#): motak char,short,int,float,double

ÿ [ieee754 bihurgailua](#)

ÿ [IEEE 754 doitasun bakarra](#): hamartar ÿ bitarra/hamaseitarra eta alderantziz

## Float Point: Zero, Infinity eta Indeterminate irudikapena

- Berretzaile-eremua zeroak edo batak direnean, ez da zenbaki baten arau orokorra betetzen  
normalizatua

5. taula. Doitasun bakarra

Zenbakiak	Exp	Zatikia
Zeroak	0x00	0

Zenbakiak	Exp	Zatikia
Zenbaki desnortutuak	0x00	0-tik desberdina
Zenbaki normalizatuak	0x01-0xFE	edozein
Infinituak	0xFF	0
NaN (ez da zenbaki bat)	0xFF	0-tik desberdina

- Zero
  - ÿ Zergatik adierazten da zero zehaztasun bakarrean 32 zeroen segida gisa?
  - ÿ Zergatik berretzaile-eremua zero denean potentzia 2-126koa da 2-127koa izan beharrean eta mantisa EZ normalizatutzat hartzen du, hau da, 0.frakzioa 1.frakzioa izan beharrean.

- Maryland oharrak
- Infinitua

## Erreferentzia

- zenbakizko analisia: adibide-programa errazak
- IEEE
- William Kahan
- Yale: C programazioa. karroza.c.
- Bruce Dawson bloga
  - ÿ <https://randomascii.wordpress.com/2012/01/11/tricks-with-the-floating-point-format/>
- wikipedia
  - ÿ <http://www.validlab.com/goldberg/paper.pdf>
    - ÿ Oating-point-en kalkuluak egiazatzeko akatsak
    - ÿ berdin?
- cprogramazioa
- kodea aholkuak
- c berrikuspena: praktikak

## 3.7. Karaktere mota

### 3.7.1. ASCII

- ASCII kodeketa
  - ÿ American Standard Code International Interchange: 7 biteko kodeketa: 0x00-0x7F tartea
  - ÿ Karaktere-kode\_hamaseimala-kode bitarra bihurtzeko taula
    - ÿ gizon ascii
    - ÿ KN King, E eranskina, pg801

Pertsonaia	ASCII hex kontrola (Sekuentzia Ihesketa)
0	0x30

Pertsonaia	ASCII hex kontrola (Sekuentzia Ihesketa)	
1	0x31	
to	0x61	
ERA	0x41	
+	0x2B	
^J	0x0A	ierro berria (\n)
^M	0x0D	itzulera autoa (\r)

ÿ konpondu J eta ^J arteko kode-erlazioa, M eta ^M arteko...

C programaren '\X' ihesak adierazten dira.

Urr Abendu Hex Char

Urr Abendu Hex Char

000 0 001	00	NUL '\0'	100 64 101	40	@
1 002 2	01	SOH (izenburuaren hasiera)	65 102 66	41A	
	02	STX (testuaren hasiera)		42	b
003 3	03	ETX (testuaren amaiera) 103 67 EOT (transmisioaren		43	c
004 4 005	04	amaiera) 104 68 ENQ (kontsulta) 105 69 ACK (aitortu)		44	d
5 006 6	05	106 70 BEL '\a' (txirrin)		45	ETA
	06			46	F
007 7	07		107 71	47	g
010 8 011	08	BS '\b' (atzerantz)	110 72 73	48	h
9 012 10	09	HT '\t' (fitxa horizontala) 111 LF '\n' (ierro berria)		49	Yo
	0A		112 74	4A	J.
013 11	0B	LH '\v' (fitxa bertikala)	113 75	4B	K
014 12 015	0C	FF '\f' (inprimaki-jarioa)	114 76 115	4C	I
13 016 14	0D	CR '\r' (karroza ret)	77 116 78	4D	M
	0E	OS (kanpora)		4E	N
017 15	0F	BAI (aldaketa)	117 79	4F	ATA
020 16 021	10	DLE (datuen estekaren ihesa)	120 80 121	50	G
17 022 18	11	DC1 (gailuaren kontrola 1)	81 122 82	51	G
	12	DC2 (gailuaren kontrola 2)		52	R
023 19 024	13	DC3 (gailuaren kontrola 3)	123 83 124	53	Bai
20 025 21	14	DC4 (gailuaren kontrola 4)	84 125 85	54	T
	15	NAK (erantzun negatiboa)		55	END
026 22	16	SYN (inaktibo sinkronoa)	126 86	56	V
027 23 030	17	ETB (trans. blk amaiera)	127 87 130	57W	
24 031 25	18	CAN (ezeztatu)	88 131 89	58	X
	19	EM (ertainaren amaiera)		59	ETA
032 26	1A	SUB (ordezkoa)	132 90	5A	Z
033 27 034	1 B	ESC (ihesketa)	133 91 134	5B	[
28	1 C	FS (fitxategien bereizlea)	92	5C	\ \W

035 29 036	1D	GS (taldeen bereizlea)	135 93 136	5 D	]
30	1E	RS (erregistroen bereizlea)	94	5E	^
037 31	1F	AEB (unitate-bereizlea)	137 95	5F	-
040 32 041	20	ESPAZIOA!	140 96 141	60	
33	21	"	97	61	to
042 34	22		14298	62	b
043 35 044	23	#	143 99 63 144 100 64	c	
36 045 37	24	\$	145 101 65	d	
	25	%			eta
046 38	26	&	146 102 66	F	
047 39 050	27	'	147 103 67 150 104 68	g	
40 051 41	28		151 105 69	h	
	29	( )			yo
052 42	2A	*	152 106 6A	j	
053 43 054	2B	+	153 107 6B	k	
44 055 45	2C	,	154 108 6C	l	
	2D	-	155 109 6D	m	
056 46	2E	.	156 110 6E	n	
057 47 060	2F	/	157 111 6F	bai	
48 061 49	30	0	160 112 70 161 113 71	or	
	31	1			q
062 50	32	2	162 114 72	r	
063 51 064	33		163 115 73 164 116 74	bai	
52 065 53	34	3..4	165 117 75	t	
	35	5			edo
066 54	36	6	166 118 76	v	
067 55 070	37	7	167 119 77 w		
56 071 57	38	8	170 120 78 171 121 79	x	
	39	9			eta
072 58 073	3A	:	172 122 7A	z	
59 074 60	3B	;	173 123 7B	{	
	3C	<	174 124 7C		
075 61	3D	=	175 125 7D	}	
076 62 077	3E	>	176 126 7E	~	
63	3F	?	177 127 7F	DU	

- ASCII hedatua

ÿ [https://en.wikipedia.org/wiki/Extended\\_ASCII#ISO\\_8859\\_and\\_proprietary\\_adaptations](https://en.wikipedia.org/wiki/Extended_ASCII#ISO_8859_and_proprietary_adaptations)

ÿ man iso\_8859\_1: latin-1: ascii hedatua: 0x80-0xFF

ÿ gizon iso\_8859-1 | grep ÿ

ÿ <http://www.theasciicode.com.ar/ascii-printable-characters/vertical-bar-vbar-vertical-line-vertical-slash-ascii-code-124.html>

ÿ Linux-en, sakatu ctrl-Shift-u-ascii\_code Sartu

ÿ Adibidea: ÿ-ren kodea hedatua 0xF1 da ÿ CSu-f1 Sartu ÿ CSu aldi berean eta u agertzen da  
kodearen zain, F-1-sartu

ÿ ascii kodearen bilatzailea

ÿ 0

ÿ ~

ÿ ~

ÿ ñ

### 3.7.2. Python

- bihurketa adibideak

ÿ pitoia

```
ord('A') hex(ord('A'))

hex(ord('\n')) chr(65) chr(0x41) [hex(ord(c))]

c - rako "Kaixo" [chr (c) c -n [ 0x48, 0x6f,
0x6c, 0x61, 0x20,

0x4d, 0x75, 0x6e, 0x64,
0x6f]] [hex( ord (c)) c for 'ñ' ] [hex(ord(c))] c -rako " \n ']
```

### 3.7.3. Unicode UTF-8

- [Unicode Nagusia](#)
- Unicode transformazio formatua (UTF)
- Unicode: Unicode karaktere-kode ezberdinaren bidez implemenatua daiteke. Unicode estandarrak Unicode Transformation Formats (UTF) definitzen ditu: UTF-8, UTF-16 eta UTF-32, eta beste hainbat kodeketa. Gehien erabiltzen diren kodeketak UTF-8, UTF-16 eta UCS-2 zaharkitua dira (UTF-16ren aitzindaria Unicoderako guztiz onartzen ez duena)
- Unicode kodetua: <https://www.unicode.org/versions/Unicode14.0.0/ch02.pdf#G25564>
  - ÿ U+ aurrizkiarekin eta ondoren zenbaki oso batekin deskribatzen da (0tik 0x10FFFF bitarteko osoak). Kodeari **code point deitzen zaio**.
- UTF-8:
  - ÿ World Wide Web-en eta Unix antzeko sistema eragile gehienetan kodeketa nagusia
  - ÿ Byte bat [1. oharra] (8 bit) erabiltzen du lehen 128 kode puntuetarako, eta gehienez 4 byte beste karaktereetarako. Lehenengo 128 Unicode kode puntuek ASCII karaktereak adierazten dituzte, hau da, edozein ASCII testu bat ere bada.
  - ÿ ñ-ak 0xc3b1 ateratzen du. Terminala Unicode UTF-8 irteerarekin konfiguratuta dago ingurune lokaleko aldagaien oinarrituta. **Local charmap** komandoa erabiliz terminalaren sarrera/irteera dugun kodeketa iraultzen dugu. **Local -m** balizkoak erabiliz . Iso-8859-1 (ascii hedatua) izan zitekeen utf8-ren ordez.
- **localectl egoera** ÿ teklatuaren sarrerako kodeketa

Sistemaren lokalizazioa: LANG=eu\_ES.UTF-8 LANGUAGE=eu\_ES.eu:en\_GB:en VC teklama-pa: n/a X11

Diseinua: es X11 Modeloa: pc105

- [utf8](#):

ÿ 8 biteko Unicode eraldaketa formatua

ÿ Luzera aldakorreko ikurrak erabiltzen ditu (Unicode karaktere bakoitzeko 1 eta 4 byte).

ÿ Byte bateko hitzen transmisiora bideratzen da.

ÿ unicode ñ

ÿ ñ Unicode puntu U+00F1 edo hex\_code\_utf8 0xC3B1 du

ÿ utf-8 wikipediak Unicode puntutik hex kodera nola joan azaltzen du.

ÿ <https://unicode-table.com/es/00F1/>

ÿ Arazoa AEBetako ASCII ez diren karaktereak kopiatzeko Firefox URL barratik: [https://es.wikipedia.org/wiki/Commutaci%C3%B3n\\_de\\_circuitos](https://es.wikipedia.org/wiki/Commutaci%C3%B3n_de_circuitos). ÿ C3B3 ó karakterearen utf-8 kodearen kode hamaseitarra da.

ÿ wikipedia utf-8:

ÿ Kodearen banaketa 1byte, 2byte, 3 byte, 4 byte arabera.

ÿ Unicode kodearen puntu utf-8tik hamaseimalera nola mapatu

ÿ man utf-8

- showkey -a : letra bat sakatzeko itxarongo du eta kodeketan sakatutako letraren kdea bistaratuko du. sistema eragilearen sarrera.

ÿ ASCII karaktere-kode estandarra (7 bit) UTF8rekin bat dator baina ez gainerako ASCII karaktere hedatuekin.

ÿ karaktere bakoitzaren kdea ascii estandarrean eta ñ, á, é, í, ó, ú si karaktereen kdea ezagutzeko erabilgarria. sistema konfiguratuta dagoen kdea (UTF8)

ÿ Ctrl-C, CR, Ctrl-CR, Ctrl-D konbinazioen kontrol-kodea ezagutzeko erabilgarria.

```
\92 0134 0x5c -> ESC tekla: ihes
^J 10 0012 0x0a -> Ctrl-CR teklak: ierro-jauzi
^M 13 0015 0x0d -> CR gakoa: Karga-itzulera
^C      3 0003 0x03 -> Ctrl-c teklak 4 0004 0x04 ->
^D      Ctrl-d teklak
ñ      195 0303 0xc3 -> MSB: Byte esanguratsuagoa. 177 0261 0xb1 ->
LSB: byte esanguratsu gutxiago hex_code_utf8 -> 0xC3B1
```

ÿ UPNA ÿ 0x55-0x50-0x4e-0x41-0x00 non 0x00 katearen amaierako NUL karakterea den.

- HTML dokumentuak

ÿ ñ ÿ &#x00F1 ÿ "unicode point" kdea erabiltzen du

- URL ÿ arakatzaile baten helbide barran jarri kamioia ÿ sartu

ÿ kopiari URLa eta kopiari barra berri batera ÿ <https://www.google.com/search?channel=fs&client=ubuntu&q=cami%C3%B3n>

ÿ %C3%B3 or-en UTF-8 hex\_code da

- Sistema eragilea: ingurune-aldaiaiak

ÿ env | grep LC\_

- Unicode taula

ÿ Erakuslea kategoriaren gainean jartzeak karaktere-multzoaren barruti hamaseimala bistaratzen du

ÿ Sinboiloak Puntuazioa:

ÿ Puntuazioa: ASCII Puntuazioa: [U0000.pdf](#)

ÿ Aurkitu diagrama kode hexadezimalaren arabera: 278a

ÿ Piktogramak: Dingbats: x278a ÿ ÿ [U2700.pdf](#)

ÿ Ikur matematikoak: Eragile matematikoak:

ÿ [wikipedia](#)

ÿ [U2200](#): Eragile matematikoen tartea: 2200–22FF

ÿ U+2228 ÿ hamaseitarra x2228 ÿ ÿ edo hamartar 8744 ÿ

ÿ U+22BC ÿ x22bc ÿ ÿ

ÿ U+22BD ÿ x22bd ÿ ÿ

ÿ U+22A6 ÿ x22a6 ÿ ÿ

ÿ beste batzuk

ÿ x1f60b ÿ ÿ

ÿ U+00F1 ÿ x00f1 ÿ ñ

ÿ 241 ÿ ñ

ÿ x2190 ÿ ÿ

ÿ x2192 ÿ ÿ

• [https://wiki.mozilla.org/Help:Special\\_characters#Unicode](https://wiki.mozilla.org/Help:Special_characters#Unicode)

• Unicode karaktere bati buruzko informazio zehatza: Pej U+0305

• Microsoft Office

• Gain-lerroa edo superlerroa

ÿ LibreOffice-k zuzeneko euskarria du [formatu/karaktere/letra-tipoan](#) gain-lerroaren hainbat estilotarako  
Efektuak" elkarritzeta-koadroa: [Ierrokatuta](#)

## 3.8. ISO-8859-1

• UTF-8ren alternatiboa latindar alfabetorako

• [https://es.wikipedia.org/wiki/ISO/IEC\\_8859-1](https://es.wikipedia.org/wiki/ISO/IEC_8859-1)

ÿ Byte 1 bakarrik erabiltzen du, beraz, ascii hedatuaren baliokidea da.

ÿ ISO/IEC 8859-15 arauak ISO 8859-1 berrikuspen bat izan zuen, eta ikurra sartu zuen.  
Euroa

• [gizon iso\\_8859-1](#)

• "ñ"-ak 0xF1 kodea du

### 3.8.1. C programazioa

• Bihurtu zenbakizko karaktere bat bere balio osora

ÿ Eragiketa aritmetiko baten bidez

• Bihurtu karaktere minuskula bat maiuskula

ÿ Eragiketa aritmetiko baten bidez

### 3.8.2. Beste batzuk

• [C hizkuntza: printf](#)

ÿ locale -a ÿ C.UTF8

ÿ ñ ÿ env printf \u00f1 \n : sartu komatxo bakunak

ÿ printf deitza

## 4. kapitulua. Eragiketa aritmetikoak eta logikoak

### 4.1. Ikastaroa

#### 1. Aritmetika eta logika

- a. Eragiketa aritmetikoak eta logikoak zenbaki osoen gainean bitarrean
- b. Biribilketa eta erroreen hedapena zenbaki errealetan

### 4.2. Helburua

- Kodean adierazten diren zenbaki natural eta osoekin batuketa eta kenketa eragiketa aritmetikoak bitarra.
- Kode bitarrean irudikatutako datuen eragiketa logikoak.
- Testu liburua
  - ÿ 3. zatia, 10. kapitulua: Konputagailuen aritmetika

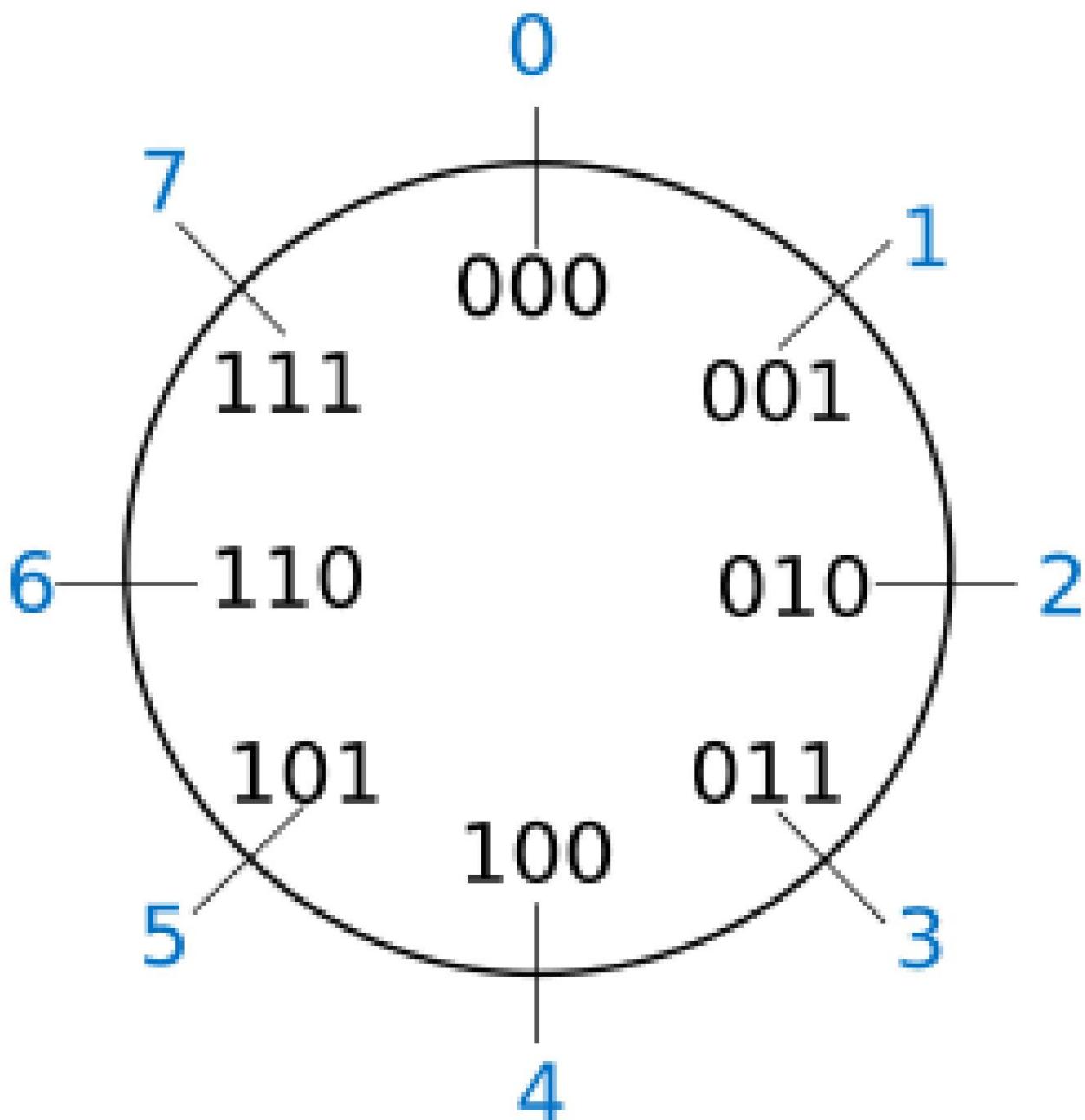
### 4.3. Sarrera

- Unitate Logiko Aritmetikoa (ALU) batuketa eta kenketa bezalako kalkulu aritmetikoko eragiketak egiteaz eta EZ, EDO, ETA eta abar bezalako eragiketa logiko boolearrak egiteaz arduratzan den oinarritzko hardware-unitatea da.

### 4.4. Aritmetika Bitarra

#### 4.4.1. 2 modulu batuketa (bitarra) bitar hutsean (NO NATURALA)

- Zenbaki naturalek ez dute zeinu baten marka denak positiboak baitira: 0,1,2,3....
- Kode bitar hutsean datuak gehitzea
  - ÿ Funtzio modulararen kontzeptua
    - ÿ Adibidea: 100.000 modulu ÿ Interpretazio grafiko modularra zirkunferentzia erabiliz. Hori Autoaren bidaia-kontagailuan gertatzen da 99.999ra iristen garenean.
    - ÿ Adibidea: zenbaki bitarren irudikapena 8. moduloan. Batuketa bitarra 8. moduloan.  
3 biteko zenbaki bitarren adierazpen grafikoa ÿ modulo= 23 = 8



31. Irudia. Sinadurarik gabeko Zenbakien Adierazpena

ŷ Batuketa: grafikoki  $7+1=0$  dela ikus dezakezu. Eragiketa egitean ALUK  $111+001$  gehitzen dio ondorioz 000.  $7+1$ ek 8. moduloan 0 den 8 balioa ematen du.

ŷ Gehitu: Zein litzateke  $7+7$ ? ŷ 14 8 modulo 6 da. Emaitza moduloaren berdina edo handiagoa bada Moduluari behar adina aldiz kendu behar diozu.

ŷ Batura: Zenbat da  $33 \bmod 8$ .  $33 = 8 \cdot 4 + 1$ . 1 (0, 7) tartean dago.

ŷ Kenketa: grafikoki  $0-1=7$  dela ikus dezakezu. Eragiketa egiterakoan ALUK  $000-001$  kentzen dio ondorioz 111

ŷ **Eramatea** (eramatea) 2 baliora iristean edo pasatzean gertatzen da.

ŷ  $1+1=\text{bat gehi bat} = bi >= 2$  ŷ Bi ( $2-2=0$ ) balioari 2 modulo kentzen diot eta bat hartzen dut. 2 balioa bitarrean 1 0 gisa adierazten da, non zero batuketa zifraren posizio berean dagoen irudikapena den eta 1 hurrengo posiziorean eramatzen den.

ŷ  $1+1+1=\text{bat gehi bat gehi bat} = \text{hiru} >= 2$  ŷ Hiru baliotik 2 modulo kentzen dut ( $3-2=1$ ) eta bat hartzen dut. 3 balioa bitarrean 1 1 gisa adierazten da, non eskuinekoa zifra gehitzen duen posizio berean dagoen irudikapena eta ezkerreko 1 hurrengo posiziorean eramandakoa den.

- Ariketa: kalkulatu  $10011011+00011011 = 10110110$  batura

Eramanak -->	1 1 1 1
	1 0 0 1 1 0 1 1 <-gehituz
	+ 0 0 0 1 1 0 1 1 <-gehituz
Batura balioa	1 3 2 1 3 2
	*****
Emaitza -->	1 0 1 1 0 1 1 0 <-batura

### Gainezka

- Batuketak edo kenketak gainezka egin duela esaten da:
  - ÿ Irudikatu nahi den emaitzaren balioa tartetik kanpo dago, zifra kopuruaren muga dela eta.
  - ÿ Eragiketa aritmetikoaren emaitza hitz funtziok onartzen duen tamaina baino handiagoa da. gordeta dagoen memoria edo erregistroa.
  - ÿ Konponbidea datuak adierazten dituzten zifren kopurua handitzea izango litzateke, baina beti ez da posible.
  - ÿ Logikoki, gainezkatzea gertatzen bada, ALUk emandako emaitza ez da zuzena. ALU Bit bat gordetzen duen OF gainezkatzeko bandera edo bandera du. Bitxia bada OF=1 esan nahi du ALUk egindako azken eragiketak gainezka egin duela. Programatzalea OF bandera irakurrita gainezkatze-erroreren bat egon den jakin dezakezu.
- Adibideak:
  - ÿ ALU unitateak "1-byte" bi sarrera-erregistro ditu, AL eta BL, bakoitza non gordetzen duen bi datu: 10011011 eta 10011011. Byte bateko irteerako erregistroa ere badu, CL. Kalkulatu formatu bitar hutsean gehitzearen emaitza: CL ÿ AL+BL eta OF banderaren balioa.

Eramanak -->	1 1 1 1
	1 0 0 1 + 1 0 1 1 <-AL
	1 0 0 1 1 0 1 1 <-BL
Batura balioa	2 1 3 2 1 3 2
	*****
Emaitza --> 1 0 0 1	1 0 1 1 0 <-batura
CL : 0 0 1 1 0 1 1 0	
HONEK: 1	

- ÿ Gaintze-errorea ALUk baturaren emaitza kalkulatu duelako: 00110110
  - ÿ 100110110 emaitza zuzena 8 biteko erregistro baten barrutitik kanpo dago. Baimendutako barrutia 0000000 eta 11111111 arteko zenbakiak izango lirateke, hau da, 0 arteko balioak. eta 255. 100110110 datuak zeinen balioa 310 den. Irtenbidea erregistroekin CPU berri bat diseinatzea litzateke. bere hitzaren tamaina byte 1 baino handiagoa den, adibidez, 2 byte. Beraz, ALUko sarreran bada AX=000000010011011 BX=0000000010011011 dugu CX eragiketaren emaitza
    - ÿ AX+BX 000000100110110 izango litzateke eta OF=0 ÿ ez dago gainezkatze-errorerik.

### 4.4.2. 2 modulo kenketa (bitarra) bitar hutsean

- Bi zenbaki natural (zeinurik gabe) kendu ahal izateko beharrezkoa da minuendoaren balioa handiagoa izatea. subtrahendearena.

ŷ 0-0 = 0

ŷ 1-1 = 0

ŷ 1-0 = 0

ŷ Zer gertatzen da Otik 1 kendu behar badut? Ezin zaio 1 balioa kendu 0 balioari. Zifra bat denean "p" posizioan dagoen minuendoaren azpianduraren "p" posizio bereko zifra baino txikiagoa da, Soluzioa p posizioaren minuendora modulua (2 bitarrean) gehitzea da eta aldi berean ere gehitu balio bera kendura baina "p+1" posizioaren bidez, beraz, gehitzen badugu balio berdina bai minuendoari bai kenkariari, kenketaren emaitzari ez zaio eragiten.

ŷ "p" posizioa: minuendoa 0 - substrahend1 ŷ Minuendoan 0+modulua-1=0+2-1= 1. 2 balioa.

"p" posizioa "p+1" posizioko 1 balioaren baliokidea da. "p+1" posizioan 1 gehituko diogu kendurari.

ŷ "p" posizioa: minuendo 0 - azpitranne 1 - eramana 1 ŷ Minenendoan 0+modulu-1-1=0+2-1-1=0 eta kendutakoaren hurrengo posizioari gehituko diogun 1 hartua.

ŷ "p" posizioa: 1-1-1 ŷ kendura 1+modulua-1-1=1+2-1-1= 1 eta 1 hartu dugu gehituko dugun

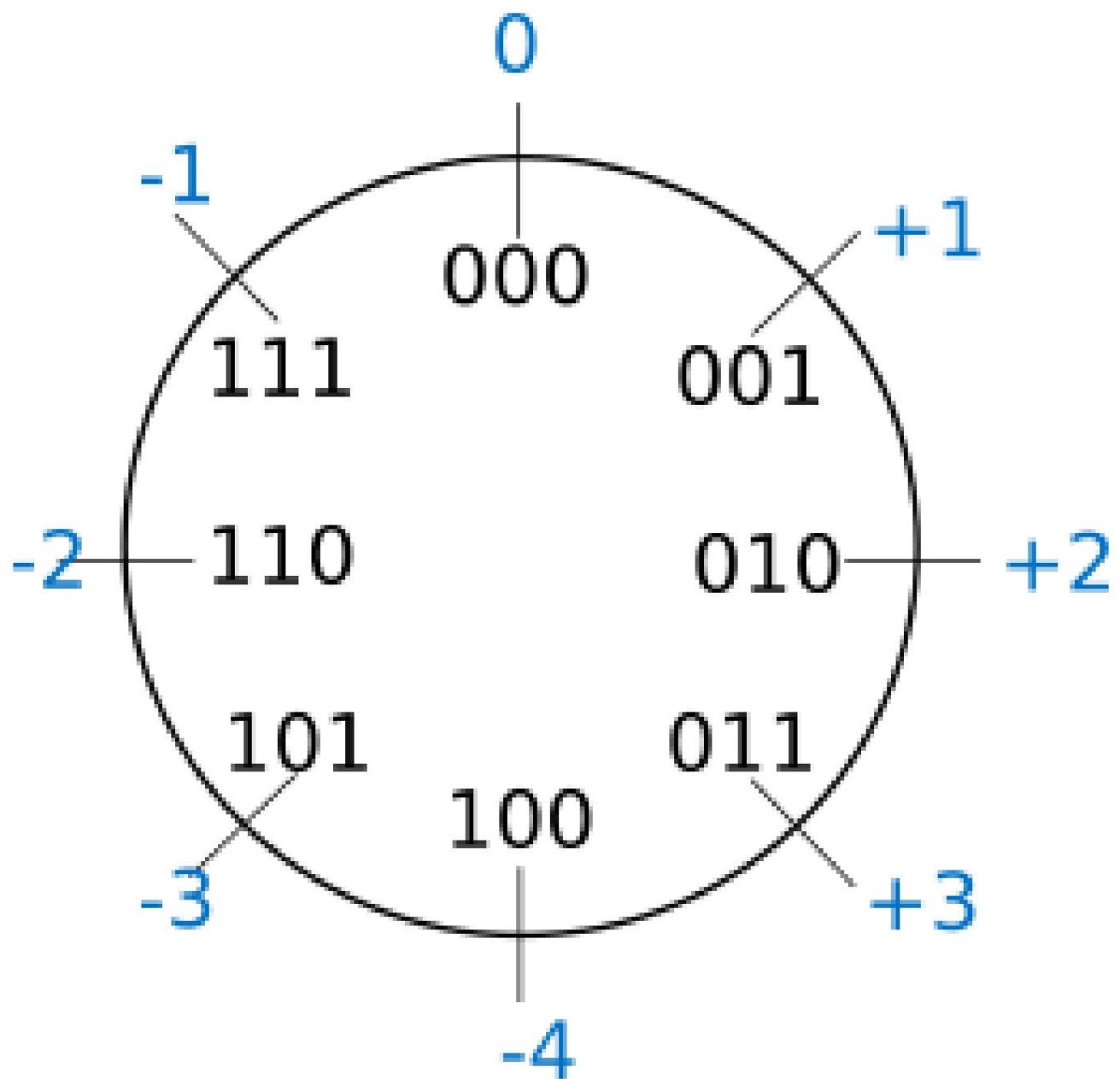
Substraendaren hurrengo posizioa.

ŷ 10110110 - 10011011 = 00011011

Gehitu kreditua minuendari	2 2	2 2
	1 0 1 -	1 0 1 1
	1 0 0 1	0 1
		1 0 <--minuend
		1 <--substrahend
Substraendeari hartutakoa gehitza	1 1	1 1
	*****	*****
Kenketa	0 0 0 1	1 0 1 1

#### 4.4.3. Batuketa/Kenketa 2 modulo (bitarra) 2 osagarrian

- Berrikusi 2-ren osagarri formatua sinatutako zenbaki osoetarako



32. Irudia 2-ren Osagarrien Adierazpena

gainera

- Egin 2-ren osagarrian 00100101 eta 0111 1 byte osoko zenbakien batuketa.
- Bi datuak zeroz hasten dira, gero positiboak dira 2 osagarriaren formatuaren arabera  
ŷ Gehigarriak hedatzen ditut denak tamaina berekoak izan daitezen. 0111 zeinu-bit hedatz 0 da 00000111

Eramanak -->	1 1 1
	0 0 1 0 0 1 0 1 <--AL
	+ 0 0 0 0 0 1      1 1 <--BL
Batura balioa	1 3 2 2
	*****
Emaitza -->	0 0 1 0 1      1 0 0 <--batura

## kenketa

- XY kenketa  $X+(-Y)$  batuketaren baliokidea da. -XY kenketa  $(-X)(-Y)$  batuketaren baliokidea da . Beraz, ALU Kenketa batuketa eragiketa erabiliz eta eragigaien zeinua aldatuz egiten da.

ÿ Adibidea: Egin 27-101 kenketa 2 osagarrian 1 byte-erregistroak erabiliz

lehenik +27 minuendoa eta +101 azpia kodetzen ditut

+27 -> 00011011

+101 -> 01100101

-101 +101 -> 10011011-ren 2 osagarria da

Eragiketa  $(-101)+27 \rightarrow 10011011+00011011$  baturaren baliokidea da

Eramanak --> 1 1 1 1

$$\begin{array}{r} 1001 \quad 101 \quad 1 \\ +0001 \quad 101 \quad 1 \\ \hline \end{array} \text{--AL}$$

Batura balioa 1 3 2 1 3 2  
\*\*\*\*\*

Emaitzza --> 1 0 1 1 0 1 1 0 <--batura

ÿ Zein da emaitzaren balioa?

emaitzak posizio-bit esanguratsuena 1ean ezarrita dauka, beraz  
balioa negatiboa da 2-ren osagarrian. Negatiboa bada ezin dut bere balioa kalkulatu  
batuketa hiztatuen bidez ez baita posiziozko irudikapena. behar dut  
aldatu zeinua positiboa izan dadin eta horrela bere balioa baturaz kalkulatu ahal izateko  
hiztatua.

10110110 -> 01001010 emaitzaren 2 osagarria zeinaren balioa +74 den, beraz  
10110010-ren balioa -74 dela.

ÿ errepikatu eragiketa ordenagailuz aldatuz eta 2 byteko erregistroak erabiliz. -n oinarrituta  
urreko atala.

10011011 zenbaki negatiboaren zeinu-bitu luzatzen dut 16 osatzeko  
bitsak

AX <-- 111111110011011 (-101)

00011011 zenbaki positiboaren zeinu-bitu luzatzen dut 16 osatzeko  
bitsak

BX <-- 0000000000011011 (+27)

10110110 emaitza negatiboaren zeinu-bitu luzatzen dut 16 guztiak osatu arte  
bitsak

CX <-- 111111110110110 (-74)

**Gainezka 2 osagarrian (C2)**

- Gehiegikeria batuketa eta kenketa aritmetiko-eragiketetan gertatzen da eragiketaren emaitza irudikapen posibleen barrutitik kanpoko tamainakoa denean, beraz, ondoriozko balioa baliogabea da eta akatsak eragiten ditu.

ŷ 2 byte-erregistroak erabiliz batuketaren adibidea:  $10000000+10000000 = 00000000$  ŷ Gainezka

ŷ Errorea -128-128 ez baita zero.

ŷ Bi gehigarriak negatiboak badira, emaitza ezin da positiboa izan

Emaitzazuzena izan dadin byte bat baino tamaina handiagoa duten erregistroak erabili beharko genituzke, adibidez 9 bit. Kasu honetan eragiketa berriro egingo dugu, datuak 1 bit gehiago luzatzu:  
 $110000000+110000000 = 100000000$  -> ez dago gainezkarik -> bi zenbakibako batura negatiboa izan da

Eragiketa hamartarrez egiten badugu ->  $(-128)+(-128) = (-256)$

ŷ Bi gehigarriak positiboa badira emaitza ezin da negatiboa izan

- Intelx86-k gainezkatze-errorea aktibatzen du sinatutako eragiketa aritmetiko baten emaitzan MSB bit-a eramateak emaitzaren balioari eragiten dionean.

ŷ

Kontuan izan batuketa eta kenketa eragiketa aritmetikoak egitean emaitza-kodea berdin-berdina dela zeinurik gabeko zenbakietan eta 2-ren osagarrian. Kodea berdina da baina hari lotutako balioa ez.

**4.4.4. 16. batuketa modulua (hamasezimala)**

- 16. modulo batuketa:

ŷ garraioa moduluaren baliora iristean edo gainditzean gertatzen da: 16.

ŷ  $0xF+0x1 = 0x10$

ŷ  $F+1=hamabost$  gehi bat = hamasei  $\geq 16$  ŷ Hamasei ( $16-16=0$ ) emaitzari 16 kentzen diot eta hartzen a.

ŷ  $0x3AF+0xA = 0x3B9$

ŷ  $F+A=hamabost$  gehi  $10 = 25 \geq 16$  ŷ hogeita bost emaitzatik 16 kentzen dut ( $25-16=9$ ) eta lortzen dut

ŷ  $0x3A1F+0xF4E1=0x12F00$

ŷ  $F+1=hamabost$  gehi  $1 = 16 \geq 16$  ŷ Hamasei ( $16-16=0$ ) emaitzari 16 kentzen diot eta bat hartzen dut.

**4.4.5. Kenketa 16. moduluan (Hamaseitarra)**

- Goian zenbaki bitarretan ikusitako guztia beste edozein oinarritan egin daiteke, adibidez. hamaseitar kodetutako zenbakietan.
- 16. modulo kenketa:

ŷ eramatea minuendoaren p posizio bat sustraendoaren p posizio bera baino txikiagoa denean gertatzen da, eta kasu horretan, minuendoari 16. modulua gehitu behar zaio eta eramana azpirendearren hurrengo  $p+1$  posiziora gehitu behar zaio. :

ŷ  $0x4308 - 0x1ABC = 0x$

	0x 4 3 0 8 <-- Minuendoa - 0x 1 ABC <--
Hartua -->	Sustraendoa 1 1 1 *****
	0x 2 8 4 C

ÿ 8-C ÿ 8+modulua\_16-12=8+16-12=12=0xC eta 1 hartu hurrengo posiziora

ÿ 0-B-Taken ÿ 0+module\_16-11-1=0+16-11-1=4=0x4 eta 1 hartu hurrengo posiziora

ÿ 3-A-Carried ÿ 3+module\_16-10-1=3+16-10=8=0x8 eta 1 hartu zuen hurrengo posiziora

ÿ 4-1-Eramatea ÿ 4-1-1=2

#### Oinarri hamaseimaleko gehiketa 2 osagarri formatuan

- 0xEC+0xAB=0x97

ÿ Bitarrean MSB bit 1 da, balioa negatiboa dela esan nahi du

ÿ Bi gehigarriak eta emaitza negatiboak dira

ÿ Bi zenbaki negatiboren batuketak gainezka ematen du emaitza positiboa bada, beraz, ez dago gainezkarik

ÿ C2 0xEC ÿ 0xEC ezeztatuta 0x13 da eta 1 ÿ 0x15 gehituz

ÿ 0xAB-ren C2 ÿ 0x54+1 ÿ 0x55

ÿ C2 0x97 ÿ 0x68+1 ÿ 0x69

#### 8. oinarrian gehitzea (Oktala)

- 8. modulo batuketa. Garraioa 8 zifraren balioa iristen edo pasatzen denean gertatzen da.

ÿ 08+01 = 010

ÿ 0377+06 = 0305

### 4.4.6. C-ko aldagai motak

- Zenbaki osoak

ÿ kar

ÿ laburra

ÿ int

ÿ luzea

- Erreala

ÿ flotatu

ÿ bikoitza

- sizeof() eragilea

- Mota bihurketa

ÿ casting

## 4.5. Eragiketa Logikoak

### 4.5.1. BITWISE Eragileak

- Bitean: bitaren araberako eragiketak
  - ¬ ez, eta, edo, xor

#### C hizkuntza

- [https://www.salesforce.com/us/developer/docs/apexcode/Content/langCon\\_apex\\_expressions\\_operators\\_understanding.htm](https://www.salesforce.com/us/developer/docs/apexcode/Content/langCon_apex_expressions_operators_understanding.htm)
- Algebra booleana
- algebra ikurrak

¬ Bit bidezko eragilea: eta &, edo |, xor ^, ez ~

Shift operadorea: ezkerrera <<, eskuinera sinatuta >>, eskuinera sinatu gabe >>>

Algebra C Eragilea		
EZ	¬	~
EDO	∨	
ETA	∧	&
XOR	⊕	^
NOR	¬∨	
NAND	¬⊕	
Ezkerrera SHIFT		x << m
Eskuineko SHIFT sinatua		x >> m
Eskuineko SHIFT sinatu gabe		x >>> m

#### Egia Taulak

x	eta	z=x¬y	z=x¬y	z=x¬y
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	0

#### Adierazpen Logikoa

- $Z = \neg X \cdot Y + X \cdot \neg Y$ 
  - Egia-taula garatzen badugu XOR eragilearekin duen baliokidetasuna egiaztatzen dugu

## 4.6. Biderketa

- Biderketa 0xFF x 0x6
  - Egin ezazu Bitarrean
    - Kontuan izan 2ko potentziazean biderkatzean biderkaduraren desplazamendu bat dagoela

eskubidea

ÿ biderkatu = gehitu eta aldatu

## 4.7. Programazioa

### 4.7.1. funtzi matematikoak

- <http://bisqwit.iki.fi/story/howto/bitmath/>
  - ÿ Iturburu kodea C hizkuntzan idatzita dago
- C liburutegi estandarra libm.so

### 4.7.2. Aplikazio

- Zenbaki osoak biderkatzen dituen programa bat garatu.

## 4.8. Hardwarea

### 4.8.1. Zirkuitu digitalak

Oinarrizkoak: Ate logikoa

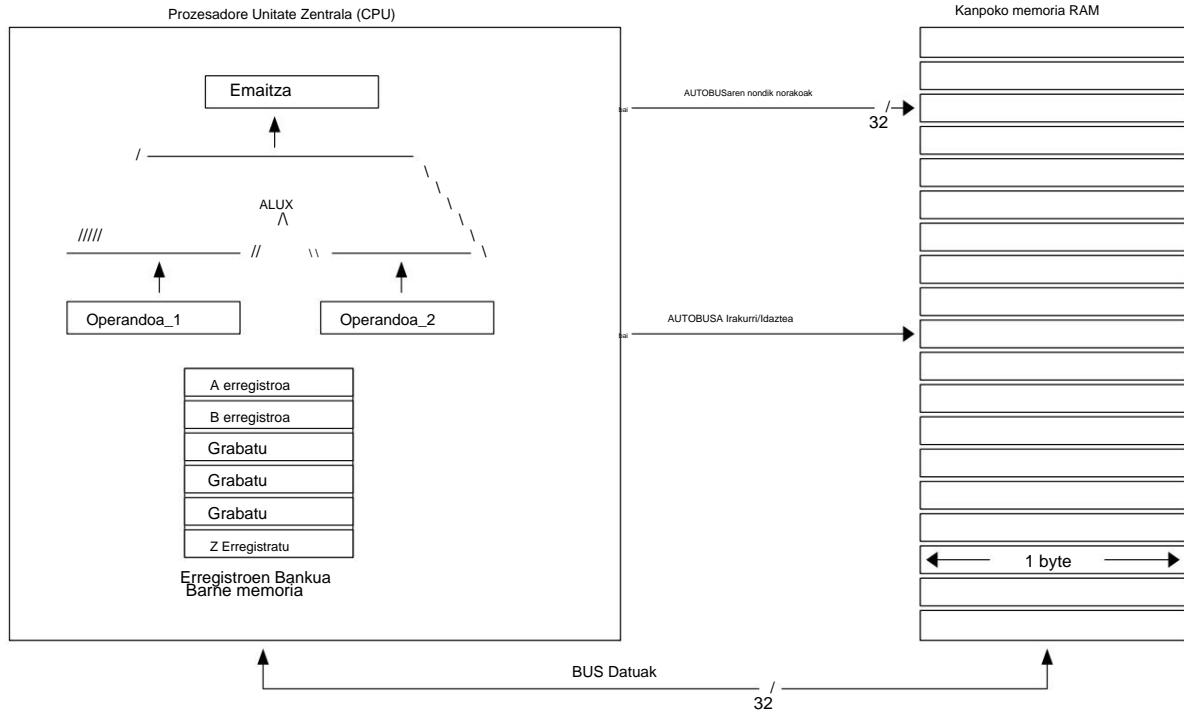
- Ate logikoak
  - ÿ ez, eta, edo, xor

Konplexuak

- batutzaile erdi, batutzaile osoa
- biderkatzailea
  - ÿ ate logikoz osatutako zirkuitu konbinatiboa
  - ÿ metagailu eta aldagailu-erregistroa

### 4.8.2. Unitate Logiko Aritmetikoa (ALU)

- Unitate logiko aritmetikoa (ALU)
- Zirkuitu Digitala
- CPU-DRAM konexioa
  - ÿ Argibideen eta Datuen transferentzia
  - ÿ ALU CPUaren barnekoa da eta erregistroetan gordetako zenbaki osoko datuak prozesatzen ditu. helburu orokorra.



33. Irudia Intel x86 32 biteko arkitektura

#### 4.8.3. EFLAG bandera erregistraztea

- EFLAGS bandera-erregistroa Intel x86 CPUaren barneko memoria-erregistroa da
- 32 biteko erregistroko bit bakoitza, emaitzaren arabera aktibatzen den bandera bat da exekutatutako azken makinaren instrukzioak egindako eragiketa.

6. taula. RFLAG Erregistroa

Bandera	Bit	Yam
C.F.	0	Eraman bandera
PF	2	Parekidetasun bandera
AF	4	bandera egokitu
ZF	6	zero bandera
S.F.	7	Sinatu bandera
OF	hemakoa	Gainetik bandera

- Eraman bandera CF:

ÿ aktibatuta dago led-ak hitzaren tamaina baino bit posizioa handiagoa badu  
ALU sinatu gabeko edo sinatutako osoko eragiketa aritmetiko batean

- Gainetik bandera:

ÿ aktibatuta dago, MSB pisu handiena duen bit-a kontuan hartuta (hitzaren tamaina kanpoan egon arren), adierazten badu  
akatsa eragiketako zeinudun zenbakia osoekin. MSB ez bada kontuan hartzen  
hitzaren tamainatik kanpo, eragiketa zuzena da.

- Parekidetasunaren bandera:

ÿ azken eragiketaren emaitzaren LSB bytearen bit kopurua bikotia den ala ez adierazten du.

- Sinatu bandera:

ÿ aktibatzen da azken eragiketaren emaitza negatiboa izan bada.

- Doitu bandera:

ÿ aktibatu egiten da LSB-n azken eragiketaren emaitza moztuta eramatene bida

- Adibideak:

**ÿ**

Batura zeinu AREKIN eta batura zeinu GABE kasuak bereizi behar ditugu. Lehenengo kasuan errore matematikoa OF banderarekin soilik detektatzen dugu eta bigarren kasuan CF banderarekin soilik detektatzen dugu errore matematikoa.

- Zeinu DUTEN zenbakiak (2 osagarria):

ÿ gainezka dagoen jakiteko, gehitu beti... kenketa bat batuketa bihur daiteke

ÿ CF garraioliaren banderak ez du zentzurik. OF interpretatzen dut operazioan akatsik dagoen jakiteko aritmetika.

$$\begin{array}{r} 11111111 \\ +00000001 \\ \hline \end{array}$$

100000000 -> Batuketa hau EZ da zuzena, izan ere, emaitza 9 bitekin irudikatzeko eragigaiak 9 biteko izan behar dute eta, beraz, 8 biteko eragigaien zeinu-bitia luzatu behar da. 9 bit dituen batura hau izango litzateke:

Eragigaiak eta 9 biteko emaitza: 11111111

$$+00000001$$

000000000 -> Ez dago gainezkatzerik, batuketa-eragileek zeinu desberdinak dituztelako

Eragigaiak eta 8 biteko emaitza: 11111111

$$+00000001$$

000000000 -> Ez dago gainezkatzerik, batuketa-eragileek zeinu desberdinak dituztelako

Ez da inoiz gainezka egingo kontrako zeinuko datuak gehitzen baditugu

AB kendu non

A=11110000

B=00010100

AB=A+(-B) -> kenketa batuketa bihurtzen dut

HORI: 11110000

-B: +11101100

A- 11011100 -> Eramatea dago baina gainezka EZ. Bi daturen batura

B: negatiboek zenbaki negatiboa lortzen dute.CF=1 eta OF=0

A=10000000

B=10000000

A+B

8 bitren ordez 9 zifrekin batuketa egiteko, bi eragigaiak luzatzen ditut 9 zifrak osatu arte 110000000

HORI:

B: +110000000

A+B: 100000000

Behatzen dugu ez dagoela gainezkarik 9 zifra erabiltzen baditugu. Baino ALU 8 biteko erregistroekin funtzionatzen badu, gainezka dago. Bi gehigarriak negatiboak dira (zeinu-bit 7. posizioa) eta emaitzaren zeinu-bitak (bit 7. posizioa) positiboa da, beraz, emaitza okerra da.

- SINATU GABEKO zenbakiak

ŷ OF gainezka-bandak ez du zentzurik. CF interpretatzen dut operazioan akatsik dagoen jakiteko aritmetika.

11111111

+00000001

\_\_\_\_\_  
00000000 -> Bit esanguratsuenean eraman dago CF=1.

Kontzeptualki gainezka dago beraz, ALUk lortutako emaitza, elektronikoki zuzena bada ere, ez da matematikoki zuzena ( $511+1=0$ ). CF=1-rekin gainezka efektua detektatzen dut.

Aurreko eragiketaren eragigaiak 1 bitarekin luzatzen ditut. 01111111

+000000001

\_\_\_\_\_  
100000000 -> Garraiorik ez -> CF=0. Batuketa hau matematikoki zuzena da  $511+1=512$  gerotzik eta, beraz, ez dago gainezkarik -> CF=0

ŷ Hurrengo kapituluau Mihiztadura [Lengoaiaren Programazioa \(x86\)](#)

#### 4.8.4. Float Point Unitatea-FPU

- Kopara mugikorreko datuak prozesatzeko unitatea
- Antzina koprozesadore matematiko izeneko CPUan integratuta ez dagoen unitatea zen
- Erabilitako Erabilera Orokorreko Erregistroetik desberdinak diren SSE izeneko erregistro espezifikoak erabiltzen ditu ALUk zenbaki osoekin eragiketak egiteko.

## 5. kapitula. Argibideen irudikapena

### 5.1. Ikastaroa

#### 1. Argibideen irudikapena

- to. Makina-lengoaia, muntaia-lengoaia eta goi-mailako lengoaiak
- b. Argibide-formatua
- c. Argibide motak eta helbideratze moduak

#### 5.1.1. Bibliografia

- W. Stalling testuliburuan aipatzen den gaia
  - ÿ 10. kapitula: Argibide multzoak: ezaugarriak eta funtziak (datuak, operandoak eta Eragiketak)
  - ÿ 11. kapitula: Instrukzio-multzoak: instrukzio-formatuak eta helbideratze-moduak (Asanblada hizkuntza)
  - ÿ B eranskina: Mihiztadura Lengoaia eta Tresna-katea

### 5.2. Helburuak

- Makinen jarraibideen erreperitorioaren arkitektura aztertzea (argibideen formatua, datuak, eragiketak eta eragiketen helbideratzea) ISA arkitekturak, oro har.

#### 5.2.1. Baldintzak

- Baldintzak:
  - ÿ Von Neumann Architecture: Konputagailuen Arkitektura, IAS Machine.
  - ÿ IAS muntaia-lengoaiaren programazioa
  - ÿ Datuen irudikapena
  - ÿ Eragiketa aritmetikoak eta logikoak

### 5.3. Goi-mailako programazio-lengoaiak vs. behe-mailako programazio-lengoaiak

#### 5.3.1. Goi mailako hizkuntzak

Java, Python, C, etab... maila altuko lengoaiak garatu ziren algoritmoak, datu-egiturak, etab... programatzeko zereginha errazteko, programatzaleentzat erabiltzeko erraza den lengoia bat erabiliz. Bestalde, ordenagailuko CPUek maneiatsen dituzten datuak eta argibideak beste hizkuntza batean daude, BINARY MACHINE lengoia, zeina ordenagailuaren prozesadore motaren araberakoa da (intel, AMD, RISC-V, etab...). Gure ordenagailuko Intel prozesadore baten makina-lengoia desberdina da smartphone bateko ARM prozesadorearen MAKINA-lengoaiatik.

Datuak bezala, instrukzioak ere formatu BITARRA batean kodetu behar dira. Datuez eta instrukzio bitarrez osatutako makina-lengoaiatzko programak memoria nagusiko RAM memorian kargatzeko eta CPUak prozesatzeko prestatuta daude.

- Maila handiko eta beheko Programazio lengoaien adibideak: Eranskina.

### 5.3.2. Makina-lengoaia eta muntaia-lengoaia

- Gai honek makina-lengoaian instrukzioen irudikapena eta interpretazioa lantzen du eta Mihiztadura hizkuntza.
- Argibideak bi hizkuntzatan irudikatu daitezke
  - ÿ Makina-lengoaia formatu bitarrean: 010101010111111000011111
  - ÿ Hizkera bitarrak instrukzio formatua dakar.
  - ÿ Lengoaia sinbolikoa edo mihiztadura testu formatuan: amaiera: GEHITU 0x33, emaitza
  - ÿ Mihiztadura-hizkuntzak sintaxia dakar
- Instrukzioak hizkuntza bitarrean irudikatzeak memoria nagusian gordetzea ahalbidetzen du, baita instrukzio-zikloa erraztu ere, PUZak deskodetu eta exekutatzeko.
- Aginduak lengoaia sinbolikoan irudikatzeak, testua adibidez, programatzaileak argibideak interpretatzeko eta programak muntaia-lengoaian garatzeko duen zereginha nahi du.
- Prozesadore zehatz baten makinen instrukzio formatuen azterketaren parte da Prozesadorearen Arkitekturaren ISA kontzeptuaren ikus eranskina

### 5.4. Makina baten instrukzio-elementuak

- Makina baten instrukzioa hainbat eremutan egituratzen da: eragiketen eremua, operandoa, etab... Eremu kopurua diseinatutako prozesadorearen araberakoa izango da.

Eragiketa-eremu		Operando Eremua		Eremua...	
-----------------	--	-----------------	--	-----------	--

- IAS makinaren kasuan, instrukzio-formatuak bi eremu baino ez ditu: eragiketa-kodea eta eragiketa-eremuia.

- Eragiketen kodea:

ÿ Instrukzioak PUZak zer eragiketa egin behar duen zehaztu behar du. Eragiketak, hala nola, batuketa eta kenketa aritmetika, eragiketa logikoak, esate baterako, eta eta, memoria nagusien kokapenen arteko datuak transferitzeko eragiketak, sarrera eta irteera eragiketak, hala nola, datuak disko gogorretik memoria nagusira transferitzea, etab.

- Iturburuko operandoaren erreferentzia:

ÿ Eragiketa batek datu bat edo gehiago prozesatzea eska dezake. Adibidez eragiketa logikoa NOT eragigai bat behar du, ADD batuketa eragiketak bi eragiketa behar ditu, etab.

- Helburuko operandoaren erreferentzia:

ÿ Batuketa-eragiketa batek bi eragiketa behar ditu, bata iturburuko eragiketa eta bestea. helmuga.

- Emaitza Erreferentzia:

ÿ Gehitze-eragiketa batek eragiketaren emaitza gordetzea eskatzen du.

- Hurrengo argibideen erreferentzia:

ÿ Instrukzioaren exekuzioa amaitutakoan, CPUari non dagoen esan behar da PC Program Counter bidez exekutatuko den hurrengo instrukzioa gorde du.

- Eragiketa-helbide implizituak: instrukzioan esplicituki agertzen ez diren helbideak.

Adibideak:

- ÿ Hurrengo instrukzioa beste erregistro batean gordetako helbidea da: Programa-kontagailua
- ÿ Eragiketaren emaitza beste erregistro batean gordetzen da: Metagailuan
- ÿ etab

#### 5.4.1. Operando arkitektura motak: adibideak

- 3 Mota

ÿ Metagailura zuzendutako arkitektura: Eragiketa bat implizituki pilatzailean dago

ÿ Pilara zuzendutako arkitektura (Pila eranskina):

ÿ Eragileak PUZaren barneko pilatik txertatzen edo kentzen dira

ÿ Pila kontzeptua: push/pop ÿ push/pull ÿ lehen sartu azken atera ÿ Lehen  
Sarrera Azken Irteera

ÿ SP: Pila Erakusleen erregistroa: pilaren goiko aldea (pilaren goiko aldea) seinalatzen duen erregistroa.

ÿ Erregistroari zuzendutako arkitektura:

ÿ Bi mota: Reg/Mem eta Load/Store, amd64 eta arm arkitekturaren kasuan bezala.  
hurrenez hurren.

ÿ Reg/Mem: instrukzioa exekutatu ahal izateko, bi eragigaietako batek a-n egon behar du  
erregistroa

ÿ Load/Store: bi eragigaietako batek bi erregistrotan egon behar dute instrukzio hori exekutatu ahal izateko.  
Korrika egin

- Adibidea: C=A+B eragiketa 4 eragiketa arkitektura ezberdinetan egiteko kodea.

pilatu	Metagailua	Erregistratu/Memoria	Kargatu/Gorde
Bultza A	Karga A	Karga R1,A	Karga R1,A
Bultza B	Gehitu B	Gehitu R3,R1,B	Karga R2,B
Gehitu	C denda	Denda R3,C	Gehitu R3,R1,R2
Pop C			Denda R3,C

ÿ Aldagaien izenak, A, B, C Memoria Nagusiaren erreferentziak dira.

ÿ RTL deskribapena

ÿ Pila: M[SP]ÿM[A],SPÿSP-1; M[SP]ÿM[B],SPÿSP-1;M[SP+1]ÿM[SP]+M[SP+1],SPÿSP+1;

ÿ Gehitu ÿ Ez dago erreferentziarik ez sorburuko operandoari ez helmugako eragigaiari.

ÿ Eragigaiak aldez aurretik pilara kargatu behar dira

ÿ Metagailua: ACÿM[A];ACÿAC+M[B];CÿM[AC]

ÿ Gehitu B ÿ Ez dago HELMUGA eragigaiaren erreferentziarik

ÿ Helmugako Operandoa aldez aurretik kargatu behar da metagailuan.

ÿ Reg/Mem: R1ÿM[A];R3ÿR1+M[B];M[C]ÿR3

ÿ Gehitu R3,R1,B ÿ EZIN DIOZU eragiketa bat baino gehiago erreferentzia MEMORIA

ÿ Eragiketa bat memorian gordetzen bada, gainerakoa aldez aurretik kargatu behar da  
erregistroak.

ÿ Kargatu/Gorde: R1ÿM[A];R2ÿM[B];R3ÿR1+R2;M[C]ÿR3.

ŷ Gehitu R3,R1,R2 ŷ Erreferentziak soilik egiten dira RECORDS, ez dago erreferentziarik.  
memoria

ŷ Jatorrizko eta helmugako eragigaiak aldez aurretik kargatu behar dira erregistroetan

ŷ

x86 arkitektura Reg/Mem-era bideratua dago, beraz, MEMORIAN iturburu-eragigai bat eta MEMORY-n helmuga-eragigaia ere ezin dira erreferentzia egin instrukzio berean, hau da, MEMORY-ra erreferentziatutako bi eragigai.

- Kode-adibidea **(AB)/(DxE+C)** eragiketa egiteko 4 ISA arkitektura ezberdinen arabera: erreferentziatutako 3 eragiketa dituen arkitektura, erreferentziatutako 2 eragiketarekin, erreferentziatutako eragiketa 1 batekin eta erreferentziarik gabeko eragiketarekin.

<b>Instruction</b>	<b>Comment</b>
SUB Y, A, B	$Y \leftarrow A - B$
MPY T, D, E	$T \leftarrow D \times E$
ADD T, T, C	$T \leftarrow T + C$
DIV Y, Y, T	$Y \leftarrow Y \div T$

(a) Three-address instructions

<b>Instruction</b>	<b>Comment</b>
MOVE Y, A	$Y \leftarrow A$
SUB Y, B	$Y \leftarrow Y - B$
MOVE T, D	$T \leftarrow D$
MPY T, E	$T \leftarrow T \times E$
ADD T, C	$T \leftarrow T + C$
DIV Y, T	$Y \leftarrow Y \div T$

<b>Instruction</b>	<b>Comment</b>
LOAD D	$AC \leftarrow D$
MPY E	$AC \leftarrow AC \times E$
ADD C	$AC \leftarrow AC + C$
STOR Y	$Y \leftarrow AC$
LOAD A	$AC \leftarrow A$
SUB B	$AC \leftarrow AC - B$
DIV Y	$AC \leftarrow AC \div Y$
STOR Y	$Y \leftarrow AC$

- 4. kasua: Stack Operando Arkitektura:

ŷ M[SP]ŷM[C];M[SP]ŷM[E];M[SP]ŷM[D];MUL;GEHITU;M[SP]ŷM[B];M[SP]ŷ M[A];SUB;DIV

ŷ bultza C; bultza E; bultza D; mul; gehitu; bultza B; bultzaA; azpia; div;

## 5.5. Argibideak x86 arkitekturako makina-lengoaian

- Ikusi eranskina [Eranskina](#) adibide bat ulertzeko. Ezin da eskuz programatu makina-lengoia ordenagailu moderno batean.

## 5.6. Ordenagailuetarako, oro har, muntatzaile-lengoaian (ASM) jarraibideen irudikapena

### 5.6.1. Sarrera

- Instrukzio-formatuaren bi eremu garrantzitsuenak eta ia bakarrak hauek dira: Eragiketa-kodea eta eragiketa-eremuen helbideratze-moduak. Eragigai kopurua 0,1,2,3 eta abar izan daiteke.

### 5.6.2. Eragiketa-kodeak

- Eragiketa multzoaren kodeketa ISA arkitektura bakoitzaren araberakoa da.
- Kategoriak eragiketa motaren arabera:
  - ÿ Datuen tratamendua: argibide aritmetikoak eta logikoak
  - ÿ Datuak kargatzea/gordetzea: Datuak erregistro eta/edo memoria-kokapenetara edo kanpotik mugitzea
  - ÿ Datuen mugimendua: I/O argibideak
  - ÿ Kontrola: Proba eta Adar-argibideak
  - ÿ Errepertorioa hau izan daiteke: murriztua/zabala, konplexua/simplea.
- Mihiztadura-hizkuntzan, eragiketa mota adierazten duen hitzari, batuketa baterako ADD adibidez, mnemoteknikoa deitzen zaio eta normalean ingelesez dago, zein eragiketa den erraz intuitzeko aukera emanez.
- Mihiztadura-lengoaiaren mnemotekniaz lantzeko modurik onena programazioa da, hurrengo x86-en mihiztadura-lengoaiaren gaian ikusiko dena.

**Table 12.3** Common Instruction Set Operations

Type	Operation Name	Description
Data transfer	Move (transfer)	Transfer word or block from source to destination
	Store	Transfer word from processor to memory
	Load (fetch)	Transfer word from memory to processor
	Exchange	Swap contents of source and destination
	Clear (reset)	Transfer word of 0s to destination
	Set	Transfer word of 1s to destination
	Push	Transfer word from source to top of stack
	Pop	Transfer word from top of stack to destination
Arithmetic	Add	Compute sum of two operands
	Subtract	Compute difference of two operands
	Multiply	Compute product of two operands
	Divide	Compute quotient of two operands
	Absolute	Replace operand by its absolute value
	Negate	Change sign of operand
	Increment	Add 1 to operand
	Decrement	Subtract 1 from operand
Logical	AND	Perform logical AND
	OR	Perform logical OR
	NOT (complement)	Perform logical NOT
	Exclusive-OR	Perform logical XOR
	Test	Test specified condition; set flag(s) based on outcome
	Compare	Make logical or arithmetic comparison of two or more operands; set flag(s) based on outcome
	Set Control Variables	Class of instructions to set controls for protection purposes, interrupt handling, timer control, etc.
	Shift	Left (right) shift operand, introducing constants at end
Transfer of control	Rotate	Left (right) shift operand, with wraparound end
	Jump (branch)	Unconditional transfer; load PC with specified address
	Jump Conditional	Test specified condition; either load PC with specified address or do nothing, based on condition
	Jump to Subroutine	Place current program control information in known location; jump to specified address
	Return	Replace contents of PC and other register from known location
	Execute	Fetch operand from specified location and execute as instruction; do not modify PC
	Skip	Increment PC to skip next instruction
	Skip Conditional	Test specified condition; either skip or do nothing based on condition
Control	Halt	Stop program execution
	Wait (hold)	Stop program execution; test specified condition repeatedly; resume execution when condition is satisfied
	No operation	No operation is performed, but program execution is continued

### 5.6.3. Eragileak: Helbideratzeko moduak

#### Kokapena

- Eragigaien kokapen posibleak.

- ÿ Instrukzioan bertan
- ÿ Barne memoria: CPU erregistroak
- ÿ Memoria Nagusia: DRAM memoria
- ÿ I/O memoria: portu izeneko sarrera/irteera kontrolagailuetan erregistratzen da.

- Instrukzioa nolabait (helbideratzeko modua) erreferentzia egin behar du funtzionatzen.

### Instrukzio-zikloan erreferentziatutako helbideak

- Instrukzio-zikloan zehar honako hauei erreferentzia egin daiteke:
  - ÿ Helbide bat instrukzioari erreferentzia egiteko
  - ÿ Lehenengo eragigaiaren helbidea
  - ÿ Bigarren eragigaiaren helbidea
  - ÿ Emaitzarako helbide bat
  - ÿ Hurrengo instrukzioari erreferentzia egiten dion helbidea
- Argibide motak exekutatzen diren bitartean erreferentziatutako helbide kopuruaren arabera.
  - ÿ Eragigairik gabeko argibideak, eragiketa bakarrarekin, eragiketa anitzekin.
  - ÿ Arkitekturaren araberakoa da: Metagailua (Adib.: IAS makina), Erregistro-Memoria (Adib.: x86 makina), Kargatu/Gorde (Adib.: ARM), Pila (Adib.: JVM makina), Memoria-Memoria
  - ÿ Eragigaiaren erreferentzia inplizituak

### Argibide-formatua: Eremuak



- ISA arkitektura batean hiru eremuko instrukzio-formatu egitura baten **adibide berezia** .
  - ÿ Eragiketa kodea: mugitu, kargatu, batu, kendu, etab.
  - ÿ A kodea: eragigaiaren eremua: eragigaiaren kokapenari egiten dio erreferentzia
  - ÿ Helbidea Mod. Kodea: A eremua interpretatzeko modua adierazten du
- EA: Helbide eraginkorra: Eragigaia kokatzen den helbide eraginkorra
- Op: Operandoa EA helbide eraginkorrean jasotako datuak dira.
- Op eragiketaren datuak honako hauetan gorde daitezke:
  1. Kanpoko RAM memoria
    - a. Datuak dituen memoria helbidea.
    - b. Instrukzio bat duen memoria helbidea. Datuak eremuetako bat dira berez instrukzioa. Berehalako Helbidea.
  2. GPR barne memoria
    - a. Rax, eax,... erregistratzen ditu

## Helbide Motak

- Eragiketa-kokapenaren EA erreferentzia-helbide eraginkorra desberdinaren arabera lortzen da helbideratze moduak.

- Helbideratzeko modua MD eremuan kodetuta dago

- Berehala:
  - ÿ Eragigaia instrukzioaren beraren eremutik lortzen da.
  - ÿ EA= ez da existitzen
  - ÿ Op=A

- Zuzena:
  - ÿ Eragigaia kanpoko memorian dago. Eragigaien eremuak helbide eraginkorra dauka
  - ÿ EA=A
  - ÿ Op=M[EA]

- Grabatu:
  - ÿ Eragigaia barne memorian dago. Eragigaien eremuak Erregistroaren erreferentzia dauka.
  - ÿ EA=A
  - ÿ Op=R

- Zeharkakoa:
  - ÿ Helbide eraginkorra kanpoko edo barneko memoria-kokapen batean gordetzen da.
  - ÿ EA=M[A] edo R
  - ÿ Op=M[M[A]] edo M[R]

- Desplazamendua:
  - ÿ Eragigaiaren helbide eraginkorra oinarrizko helbide baten eta oinarrizko helbidearekiko desplazamendu baten arteko eragiketa aritmetiko baten bidez lortzen da. Oinarrizko helbidea erreferentzia gisa hartzen da eta desplazamendua oinarri-helbidearekiko erlatiboa da.
    - to. Ordenagailuko programa-kontagailuari dagokionez:
      - ÿ Oinarrizko helbidea implizituki programaren kontagailua da eta desplazamendua eragigaien eremuan dago.
      - ÿ EA=PC+A
      - ÿ Op=M[EA]

- b. Oinarriari dagokionez:
  - ÿ Desplazamendua eragiketa eremuan dago eta oinarrizko helbidea erregistroan dago.
  - ÿ EA=R+A
  - ÿ Op=M[EA]

- c. Indexatua:
  - ÿ Desplazamendua erregistroan dago eta oinarrizko helbidea eragigaien eremuan dago.
  - ÿ EA=A+R
  - ÿ Op=M[EA]

- Sorburuko edo helmugako eragiketei erreferentzia egiteko, instrukzio-arkitektura oso malgua da, eragigai horiek bideratzeko modu desberdinak baitaude.

## 5.7. Intel hizkuntza versus AT&T hizkuntza

### 5.7.1. i386/amd64 arkitektura mutua-lengoaiak

- AMD64 arkitekturako instrukzio-multzoaren makina-kode hizkuntza bakarra da, baina ez horrela, aipatutako arkitekturari dagokion mihiztadura.
- "Konputagailuen Egitura" irakasgaien telefono konpainiaren AT&T sintaxia erabiltzen da AT&T estatubatuarran.

### 5.7.2. Argibideen sintaxia INTEL hizkuntzan

- Mihiztadura-lengoaiaren instrukzioen formatua instrukzio-sintaxia bezala ezagutzen da.
- ASM SINTAXA: Etiketa-Eragiketa Kodea- Operandoa1- Operandoa2- Iruzkina
- x86-64
- x86

7. taula. Intel Sintaxia

etiketa:	op_mnemonic	eragile_helmuga	, eragiketa_iturria	#iruzkindu
----------	-------------	-----------------	---------------------	------------

ÿ Adibidea:

ÿ begizta: sub rsp,16 ;RSP ÿ RSP-16. Begizta kenketa eragiketarekin hastea  
 ÿ je loop ;je: salto berdina: salto egin azken eragiketak zero emaitza eman badu  
 ÿ batura: gehitu eax,esi ;EAX ÿ EAX+M[ESI] . Gehitu  
 ÿ mov ax,[emaitza] ;AX ÿ M[emaitza]. Kopiatu emaitza

ÿ Mihiztadura-lengoaiaren sintaxia erabilitako mihiztadura-prozesuaren itzultzalearen (mihiztatzailea) araberakoa da, kasu honetan, NASM mihiztatzailea erabiltzen da. Ikus adibide bat Intel muntatzale-lengoian eta "NetWide Asm" (nasm) mihiztagailuan programa baten eranskinean

## GNU Batzarra (Gas)

- AT&T telefono konpainiak garatutako hizkuntza
- Mihiztagailua (GNU gisa)
  - ÿ arkitekturak: i386, amd64, mips, 68000, etab.
  - ÿ Sintaxia: Etiketa-Eragiketa Kodea- Operandoa1- Operandoa2- Iruzkina

8. taula. AT&T sintaxia

etiketa:	op_mnemonic	eragiketa_iturria	, eragiketa_helmuga	;iruzkindu
----------	-------------	-------------------	---------------------	------------

ÿ Adibidea:

ÿ begizta: subq \$16,%rsp ;RSP ÿ RSP-16. Begizta kenketa eragiketarekin hastea  
 ÿ je loop ;je: salto berdina: salto egin azken eragiketak zero emaitza eman badu  
 ÿ batura: gehi %esi,%eax ;EAX ÿ EAX+M[ESI] . Gehitu  
 ÿ movw %ax,result ;AX ÿ M[emaitza]. Kopiatu emaitza

### • ETIKETA

ÿ Lehenengo zutabeen zehazten da. Atzizkia du:

- ERAGIKETA-KODEA: sinbolo mnemoteknikoak erabiltzen dira intuizioz interpretatzen laguntzen dutenak operazioa. Peg: GEHITU gehitu, MOV mugitu, SUB kendu, ...
- ERAGIKETA ITURRI ETA/EDO HELMUGA
  - ÿ datu alfanumerikoa: irudikapen alfanumerikoa ÿ 16
  - ÿ berehalako helbideratzea: \$ aurritzka
  - ÿ kanpoko memoria helbidea: etiketa ÿ emaitza
  - ÿ zuzeneko helbidea
  - ÿ CPU barneko erregistroak: %rax,%rbx,%rsp,%esi,...
  - ÿ % aurrizkiak izenak erregistro bati erreferentzia egiten diola esan nahi du
  - ÿ Eragilearen datuen tamaina: atzizki mnemoteknikoak: q(quad):8 byte, l(luzea):4 byte, w(hitza):2 byte, b(byte):1 byte.

ÿ Mihiztadura (~~itzikuntza eta sintaxia hasztaketa prozesuaren taulaak eta erantzunak~~) berakoa da

## 5.8. Eragileak: Helbideratzeko moduak

### 5.8.1. Kokapena

- Adibidea:
  - ÿ begizta: SUBQ \$16,%rsp ;begizta hasiera
    - ÿ Iturburuko eragia: \$-k BEREHALA helbideratzea adierazten du Eragigaia iturrian bertan dago. instrukzioa ÿ Operandoa=16
    - ÿ Helmuga eragiketa: % ERREGISTROA adierazten du. Eragigaia RSP erregistroan dago
    - ÿ batura: ADDW (%ESI), emaitza ;eragiketaren amaiera
      - ÿ Iturburuko eragia: () ADIERAZIOA eta % erregistroa adierazten du ESI erregistroak eragigaia dagoen memoria helbidea du.
      - ÿ Helburu-eragigaia: "emaitza" etiketa bat da. ZUZENDARITZA ABSOLUTUA. Eragilea da "emaitza" memoria helbidean.

### 5.8.2. Helbideratzeko moduak

- Muntatzailearen eskuliburua, x86 arkitekturaren menpeko zuzentarauei buruzko atala

ÿ [https://sourceware.org/binutils/docs-2.26/as/i386\\_002dDependent.html#i386\\_002dDependent](https://sourceware.org/binutils/docs-2.26/as/i386_002dDependent.html#i386_002dDependent):

ÿ Lehenengo sei atalak gutxienez irakurtzea GOMENDUTA

- Helbideak:

BEREHALA:	Eragigaiaren balioa instrukzioaren opkodearen ondoren kokatzen da. Iturburuko eragiketa bakarrik zehazten da.
	Sintaxia: Eragigaiaren balioa \$ aurrizkiaren bidez adierazten da. adibidea: movl \$0xabcd1234, %ebx. Iturburu-eragigaia 0xABCD1234 balioa da
GRABATU:	Eragigaiaren balioa CPU erregistro batean dago.

	<p>sintaxia: % aurritzka duen erregistroaren izena.</p> <p>adibidea: movl %eax, %ebx. Iturburuko eragiketa EAX REGISTRO eta helmuga EBX REGISTRO da</p>
ZUZENA:	<p>Memorian gordetako eragiketa seinalatzen duen helbide eraginkorra Nagusia etiketak aipatzen duen helbide absolutua da eragiketa eremuan zehaztuta. Programatzaleak erabiltzen du zuzeneko helbidea baina konpilatzaleak a bihurtzen du programaren kontagailuarekiko helbideratzea. Ikus helbideratza desplazamenduarekin.</p>
	<p>sintaxia: programatzaleak zehaztutako etiketa</p> <p>adibidea: je somePlace . Joan etiketak markatutako helbidera somePlace aurreko eragiketaren emaitzak ZF=1 bandera aktibatzen badu RFLAG erregistroarena.</p>
INDEXATUA:	<p>Eragigaiaren balioa memorian kokatzen da. Norabide eraginkorra Memoria seinalatzen duena base_register balioaren GEHI eskalaren BURUA da indizea_erreregistroko balioa, GEHI desplazamendua.</p> <p>EA=Offset+R_Base+R_index*Scale</p>
	<p>Sintaxia: komaz bereizitako balioen zerrenda parentesi artean (oinarriza_erreregistroa, indizea_erreregistroa, eskala) eta desplazamendu baten aurretik.</p> <p>adibidea: movl \$0x6789cdef, -16(%edx, %eax, 4) . Norabide eraginkorra Helmuga EDX + EAX*4 - 16 da.</p>
ZEHARKOAK:	<p>(base_register)-n indexatzeko modu orokorra zehazten badugu orduan, operandoaren helbidea ez da indexatz lortzen baina helbide eraginkorra rdx-en edukia da eta beraz zeharka eragiketara sartzen da.</p>
	<p>sintaxia: (base_register)</p> <p>adibidea: movl \$0x6789cdef, (%edx) . Helmugaren helbide eraginkorra da EDX. EDX erakuslea da.</p>
ERLATIBOA:	<p>Eragigaiaren balioa memorian kokatzen da. -ren norabide eraginkorra Eragigai oinarrizko erreregistro batean jasotako balioaren batura gehi balio bat da desplazamendu.</p>
	<p>sintaxia: grabatu parentesi artean eta berehalako desplazamendua parentesia.</p> <p>adibidea: movl \$0xaabbccdd, -12(%eax) . -ren norabide eraginkorra helmuga eragiketa EAX-12 da</p>

## Adibideak

9. taula. Eragileen helbideratze moduak

Operandoaren balio-operandoa helbideratza	Moduaren izena
\$0	Zero Balioa
%rax	RAX
begizta_irten	M[loop_exit]
datu_elementuak(%rdi,4)	M[datu_elementua + 4*RDI]
(%rbx)	M[RBX]
(%rbx,%rdi,4)	M[RBX + 4*RDI]

- M[loop\_exit]: zuzena loop\_exit kanpoko memoria-helbide bat baita eta M-k kanpoko memoria adierazten baitu.
- M[RBX]: zeharkakoa, RBX barne-memoriaren helbidea delako eta M-k kanpoko memoria adierazten baitu: To mem. kanpoko mem. bidez sartzen da. barnekoa.

## 5.9. Programak ASM hizkuntzan eta Binary hizkuntzan

- Adibideak Eranskinean

# 6. kapitula. Mihiztadura-lengoaien programazioa (x86): goi-mailako lengoaien oinarrizko eraikuntzak.

## 6.1. Ikastaroa

### 1. Programazioa x86 Asanblea Lengoaian

- to. x86: Datuen irudikapena, argibideen irudikapena, helbideratze moduak.
- b. Esleipenen adierazpenak
- c. Baldintzazko adierazpenak
- d. Begiztak

eta. Funtzio edo azpierrutinen deiak eta itzulketak

## 6.2. Sarrera

### 6.2.1. Helburuak

- x86-64 arkitekturako makinaren instrukzio-errepertorioaren arkitektura (argibide-formatua, datu-formatua, eragiketak eta eragiketen helbideratzea) aztertzea, Gnu\_AS(gas) mihiztatzaile-lengoaian programen garapen praktikoan erabiltzeko.
- x86 (32-bit) arkitekturarako mihiztatzaile-lengoaian programa txikiak garatzeko gaitasuna, bai paper inprimatuaren, bai garapen konputazionalaren ingurunean, bai ordenagailu pertsonalean eta baita Informatika Laborategiko lanestazioetan ere.

### 6.2.2. Baldintzak

- Baldintzak:
  - ÿ Von Neumann Architecture: Konputagailuen Arkitektura, IAS Machine.
  - ÿ IAS muntaia-lengoaiaren programazioa
  - ÿ Datuen irudikapena
  - ÿ Eragiketa aritmetikoak eta logikoak
  - ÿ Jarraibideen irudikapena
    - ÿ [ASM x86 sintaxia eta helbideratzea](#): GnuAS mihiztadura-lengoaian eta helbideratze-moduak x86 arkitekturarako jarraibideen sintaxia.

### 6.2.3. Erreferentziak

- Eskuzko muntatzailea honela: [Eranskina: Toolchain Toolchain "as i386"](#)
- **Irakurri** "Milatzailea gisa" eskuliburuaren lehen zortzi atalak: [Eranskina: "i386 gisa" ezaugarriak](#)
- Eskuzko muntatzailea honela: [eranskina: "i386 gisa" zuzentaraauak](#)
- [Eranskinean](#) eskuliburuuen erreferentziak
- Bibliografia: [Muntaien Programazioa](#).
- Osatu [WikiBook](#) programazio oharrak AT&T hizkuntzan hainbat alderdirekin.

## 6.3. Ordenagailuen egitura Intel Arkitekturarekin x86-64

### 6.3.1. Intel ISA x86 eskuliburuak

- <https://www.intel.es/content/www/es/es/architecture-and-technology/64-ia-32-architectures-software-developer-vol-1-manual.html>

- 8 bolumen: 1,2A,2B,2C,3A,3B,3C,3D

ÿ 1. liburukia: Oinarrizko Arkitektura

ÿ 2A, 2B eta 2C liburukiak: Instruction Set Reference

ÿ 3A, 3B, 3C eta 3D liburukiak: Sistemaren Programazio Gida

### 6.3.2. Sarrera

- GAS x86 mutua-lengoaian modu eraginkorrean programatu ahal izateko, oinarrizkoak izatea beharrezko da. x86 ISA-ren oinarriak.
- 64 biteko amd64 arkitektura aztertuko da, 32 biteko i386 arkitektura ere barne hartzen duena.
- Intel eta AMD konpainiek arkitektura hauek partekatzen dituzte, zeinen merkatuak goi-mailako ordenagailu biak hartzen dituena mahaigaina zerbitzari gisa.

### 6.3.3. CPU-Memoria

- Ikusi [Eranskinean](#) ordenagailu baten egitura orokorra

### 6.3.4. x86 CPUaren barneko erregistroak

#### sarrera

- Programatzaleak amd64 arkitekturan EZ atzi ditzake erregistroak
  - ÿ PC: Programa-kontagailua: x86-k RIP deitzen du: 64 bit
  - ÿ IR: Instrukzio-erregistroa: 64 bit
  - ÿ MBR: Memoria buffer erregistroa: 64 bit ÿ HITZA TAMAINA: 64
  - ÿ MAR: Memoria-helbideen erregistroa: 40 bit
    - ÿ Memoria Edukiera: 240 : 1TB

- I386 arkitekturen kasuan, ordezkatu 64 bit 32 bitemen eta MAR erregistroa ere 32 bitemko da.

#### Programatzaleak ikusgai dauden erregistroak

63-0	31-0	15-0	15-8	7-0
rax	eax	aizkora	oi	du
rbx	ebx	bx	bh	bl
rcx	ecx	cx	ch	cl
rdx	edx	dx	dh	dl
rsi	Hori da	Bai		sil
rdi	zumurubila	eman		dil
rbp	ebp	bp		bpl

rsp	esp	sp		spl
r8	r8d	r8w		r8b
r9	r9d	r9w		r9b
r10	r10d	r10w		r10b
r11	r11d	r11w		r11b
r12	r12d	r12w		r12b
r13	r13d	r13w		r13b
r14	r14d	r14w		r14b
r15	r15d	r15w		r15b

### Bateragarritasuna 32-64

- 64 biteko arkitekturako erregistroen izendapenean, R ordeztu E-rekin eta izendapena lortuko dugu.  
32 biteko arkitekturarena.

64 bit	32 bit
R.I.P.	EIP
RAX	EAX
RFLAG	EFLAG
.....	.....

- Salbuespenak daude

### Kontrol-bandera-erregistroa

- EGOERA erregistroa: Instrukzio baten exekuzioak hori bandera izeneko bit batzuk aktibatzen ditu egindako eragiketaren ondorioak adierazi. Adibidea: gainezkatze bandera: eragiketa dela adierazten du egindako aritmetikak aipatutako eragiketaren emaitza gainezka egin du.
- [wikipedia](#)
- OSZAPC banderei bakarrik begiratzen diegu.

10. taula. RFLAG Erregistroa

Bandera	Bit	Yam
C.F.	0	Eraman bandera
PF	2	Parekidetasun bandera
AF	4	bandera egokitu
ZF	6	zero bandera
S.F.	7	Sinatu bandera
OF	namika	Gainerako bandera

- Eraman bandera:

ÿ aktibatuta dago led-ak hitzaren tamaina baino bit posizioa handiagoa badu  
ALU sinatu gabeko edo sinatutako osoko eragiketa aritmetiko batean

- Gainerako bandera:

ÿ aktibatuta dago, baldin eta, MSB pisu handiena duen bita kontuan hartuta (nahiz eta hitzaren tamaina kanpoan egon), eragiketa aritmetikoan errore bat adierazten badu zenbaki osoekin zeinudun. Hitzaren tamainaz kanpoko MSB ez bada kontuan hartzen, eragiketa zuzena da.

- Parekide dasunaren bandera:

ÿ azken eragiketaren emaitzaren LSB bytearen bit kopurua bikoitia den ala ez adierazten du.

- Sinatu bandera:

ÿ aktibatzen da azken eragiketaren emaitza negatiboa izan bada.

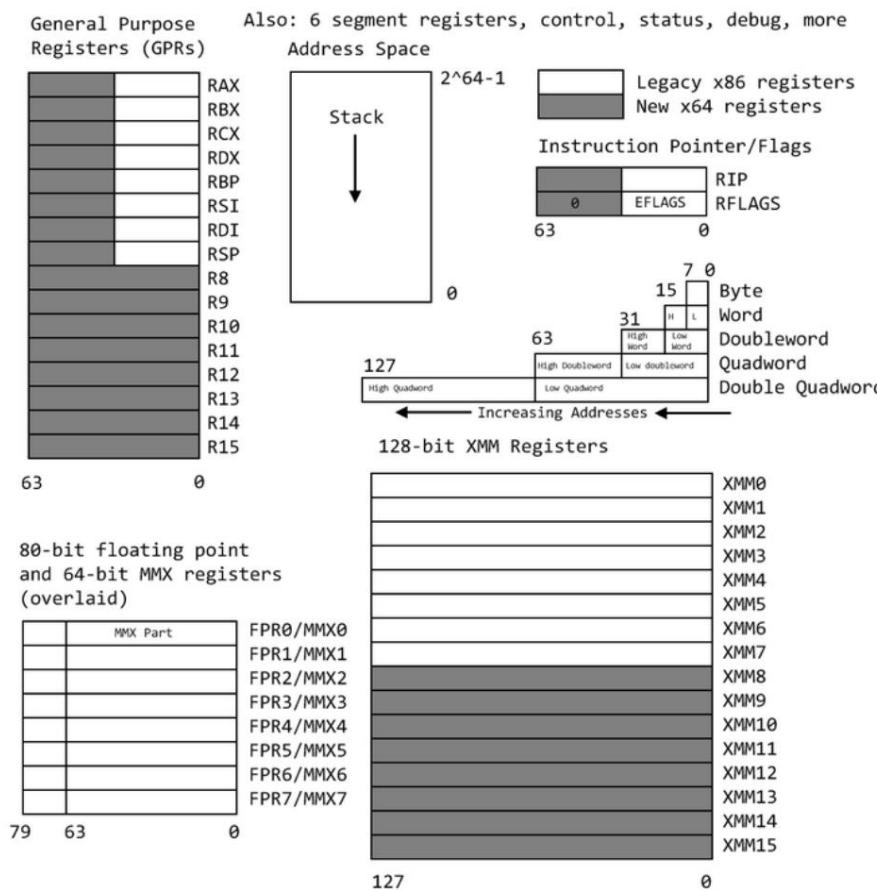
- Doitu bandera:

ÿ aktibatu egiten da LSB-n azken eragiketaren emaitza moztuta eramatzen bada

## Gaintze-kasuak

ÿ Gainezkatzearen ondoriozko errore matematikoen adibideak: [CF eta OF](#) banderak aktibatzea

### x86 CPUaren barneko beste erregistro batzuk



- Segmentu Erregistroak: CS,DS,ES,FS,GS,SS

ÿ [https://en.wikipedia.org/wiki/X86\\_memory\\_segmentation](https://en.wikipedia.org/wiki/X86_memory_segmentation)

ÿ Normalean inplizituki erabiltzen dira: instrukzioak kode-segmentuan daude CS erregistroarekiko helbideetan, datuak segmentu-datueta daude DS erregistroarekiko helbideetan, pila-segmentuan dago SS-ko helbideetan. erregistratu.

ÿ erabilera esplizitua:

ÿ `movl $42, %fs:(%eax); M[fs:eax]j42 ; eax-ek helbidea dauka`

**FS helbidea**

ŷ Sistema Eragileak segmentu-erregistroak erabiltzen ditu memoria birtualean kudeaketan orrialdea eta segmentazio mekanismoak: <https://nixhacker.com/segmentation-in-intel-64-bit/> : The Memoriaren kudeaketa irakasgaiaren bigarren zatiko gaia da.

- fp, mmx eta xmm erregistroak instrukzioak konplexuak exekutatzeko erabiltzen dira, hala nola, hori tangentea Koma mugikorreko zenbaki errealetan edo anitzekin eragiketak egiten dituzten jarraibideetan jardutea osoko datuak (Single Instruction Multiple Data) (adibidez, produktu eskalarra).
- Informazio gehiago [FPU\\_x87 eranskinean](#)

## 6.4. Datuen irudikapena hizkuntzan muntatzailea (ASM) i386/amd64 arkitekturarako

### 6.4.1. Datu mota

#### Zenbakiak eta Pertsonaiak

- Zenbakirik gabeko zenbakia (naturala): kodeketa bitar naturala
- Zenbaki oso sinatuak: **2 osagarri** kodetutako zenbaki osoak
- Zenbaki erreal errealkak zehaztasun bakarreko edo bikoitzeko **IEEE-754** formatuan kodetuta
- Karaktere alfanumerikoak: **ASCII** kodea

#### Datuen Ataleko Zuzentaraauak

- Erreferentzia [Eranskina](#)

11. taula. Oinarritzko zuzentaraauak

Zuzentaraauak	deskribapena
.global edo .globl etiketa	aldagai globalak
.atala .datuak	tokiko aldagai estatikoen atala hasieratu
.atala .testua	argibideak atalean
.atala .bss	hasiera gabeko aldagaien atala
.atala .rodata	irakurtzeko soilik aldagaien atala
deskribapena ,initzizena	aldagai mota, adibidez, @funtzioa
.ohikoa 100	hasierarik gabeko 100 byte gordetzen ditu eta izan daiteke globalki erreferentziatuta
.lcomm begizta, 100	tokiko ikurrarekin erreferentziatutako 100 byte erreservatu begizta. Hasierarik gabekoa.
.espazioa 100	100 byte erreserva zerora hasieratuta
.100 espazioa, 3	100 byte erreservatu hasierako 3
.string "Kaixo"	gehitu 0 bytea katearen amaieran
.asciz "Kaixo"	gehitu 0 bytea katearen amaieran
.ascii "Kaixo"	ez du NULL karakterea gehitzen katearen amaieran
.byte 3,7,-10,0b1010,0xFF,0777	1 byteko tamaina eta formatuak hamartar, hamartar, hamartar, bitarra, hamaseitarra, zortzitarra

Zuzentaraauak	deskribapena
.2byte 3.7,-10,0b1010,0xFF,0777	tamaina 2 byte
.3.7,-10,0b1010,0xFF,0777 hitza	tamaina 2 byte
.laburra 3.7,-10,0b1010,0xFF,0777	2B tamaina
.4byte 3.7,-10,0b1010,0xFF,0777	4B tamaina
.luzea 3.7,-10,0b1010,0xFF,0777	4B tamaina
.int 3.7,-10,0b1010,0xFF,0777	4B tamaina
.8byte 3.7,-10,0b1010,0xFF,0777	8B tamaina
.quad 3.7,-10,0b1010,0xFF,0777	8B tamaina
.octa 3.7,-10,0b1010,0xFF,0777	zortziko formatua
.bikoitza 3.14159, 2 E-6	zehaztasun bikoitza
.karroza 2E-6, 3.14159	zehaztasun bakarra
.bakarra 2E-6	zehaztasun bakarra
.sartu "fitxategia"	fitxategia barne hartzen du. Komatxoak derrigorrezkoak dira.
.equ ARRAKASTA, 0	ARRAKASTA ikurra 0 zenbakiarekin lotzen duen makroa
.macro macname macargs	izeneko makro baten hasiera definitzen du macname eta macargs argumentuak
.amaiera makroa	makro baten amaiera definitzen du
.lerrokatu n	ondorengo argibideak edo datuak orduetan hasiko dira n byteko helbide anitz bat.
.amaiera	muntaketaren amaiera

- Et: Etiketa

#### 6.4.2. x86 eragigaien tamaina

- Eragileen tamaina: atzizki MNEMONIKOAK.

q (laukoa) 8byte  
 l (luzea) 4byte  
 w (hitza) 2 byte  
 b (byte) 1byte

ÿ Adibideak:

```

    ÿ movq %rax,emaitza
    ÿ movl %eax,result
    ÿ movw %ax,result
    ÿ movb %ah,emaitza
  
```

#### 6.4.3. Byte-lerroatzea: Big-LittleEndian

- Byte anitzeko datu baten byteak memorian gorde daitezke MSB-\_- LSB edo MSB-\_ norabidean.  
-LSB

- Little Endian lerroatzea: LSB byte txikiena memoria-kokapen baxuenean gordetzen da

- 0x40000 adibidea: 00 AF BF CF

ÿ 4 byteko datuak 0x40000 memoria nagusian gordetzen dira: 00 AF BF CF

ÿ Byteak goranzko memoria helbidean gordetzen dira. Horizontalki idazten denean, Gorantzeak ezkerretik eskuinera esan nahi du.

ÿ 0x40000 posizioan 00 byte dago ÿ LSB (byte esanguratsu gutxien)

ÿ 0x40001 posizioan AF bytea dago

ÿ 0x40002 posizioan BF bytea dago

ÿ 0x40003 posizioan CF ÿ MSB (byte esanguratsuena) bytea dago

HELBIDEAK

EDUKIA

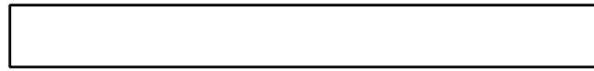
0x00000



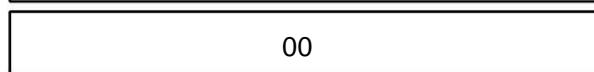
0x00001



0x00002



0x40000



0x40001



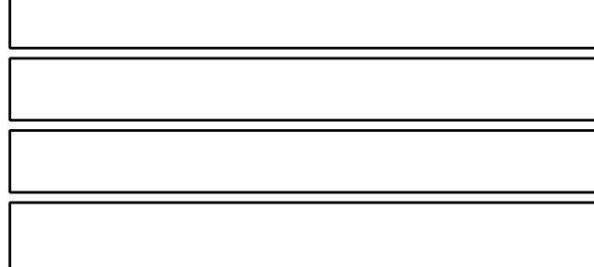
0x40002



0x40003



0xfffff



ÿ Byterik arinena memoria baxueneko kokapenean gordetzen da. Kokapen baxuena lau 0x4000 da, non 00 gordetzen den, beraz, hau da byte baxuena. Datuak little-endian formatuan gordeta 0xCFBFAF00 da.

ÿ i386/amd64 arkitekturak LITTLE ENDIAN erabiltzen du

- Little endian formatua jarraitzen duten informazio motak.

Argibideetarako formatua eremuena araberakoa da, beraz, ez du zentzurik posizioei buruz hitz egiteak

instrukzioaren pisu handiagoa edo txikiagoa, little endian formatua ez jarraitzen.

Kateek ez dute baliorik adierazten eta, beraz, ez dute formatu txikia jarraitzen  
endian

Zenbaki osoak little endian formatuan gordetzen dira.

Zenbaki errealak little endian formatuan gordetzen dira

Memoria-helbideak Little Endian erakundeari jarraituz gordetzen dira.

- BigEndian formatua

Biltegiratz-ordena endian txikiaren alderantzizkoa da, hau da, datuen LSB byte-a gordetzen da  
datuek hartzen duten eskualdeko memoria-helbide handienean.

#### 6.4.4. Adibidea

- Sum1toN att x86-32 programaren kodea aztertu

#### 6.5. Hizkuntzako Argibideen Errepertorioa

muntatzailea (ASM) i386/amd64 arkitekturarako:

Eragiketak

##### 6.5.1. Adibidea

- Jarraian aztertutako atal bakoitzean, sum1toN att x86-32 programaren kodea aztertu  
eragiketa kodeak interpretatzea.

##### 6.5.2. eskuliburu azkarra

- Intel eskuliburu azkarra: gomendagarria

##### 6.5.3. Eskuliburuak eta Taulak

- Eranskinean eskuliburuen erreferentziak

##### 6.5.4. Eragiketa-kodeen deskribapen mota Intel eskuliburuan

MOV eragiketa

- MOV

MOV -- Mugitu datuak

Opcode Instrukzioa	Erlojuak	Deskribapena
88 /r MOV r/m8,r8 89 /r	2/2	Eraman byte-erregistroa r/m bytera
MOV r/m16,r16 89 /r MOV	2/2	Eraman hitz-erregistroa r/m hitzera
r/m32,r32	2/2	Eraman dword erregistroa r/m dword-ra
8A /r MOV r8,r/m8	2/4	Mugitu r/m byte byte erregistrora
8B /r MOV r16,r/m16	2/4	Mugitu r/m hitzetik hitz erregistrora
8B /r MOV r32,r/m32	2/4	Eraman r/m dword dword erregistrora
8C /r MOV r/m16,Sreg	2/2	Mugitu segmentu-erregistroa r/m hitzera
8D /r MOV Sreg,r/m16	2/5,pm=18/19	Mugitu r/m hitza segmentu-erregistrora

A0	MOV AL,moffs8	4	Mugitu bytea (seg: offset) AL-era
A1	MOV AX,moffs16	4	Eraman hitza (seg: offset) AX-ra
A1	MOV EAX,moffs32 4		Eraman dword at (seg:offset) EAXera
A2	MOV moffs8,AL 2		Eraman AL-ra (seg:desplazamendua)
A3	MOV moffs16,AX 2		Eraman AX-ra (seg: offset)
A3	MOV moffs32,EAX 2		Eraman EAX hona (seg: offset)
B0 + rb	MOV reg8,imm8 2		Mugitu berehalako bytea erregistratzeko
B8 + rw	MOV reg16,imm16 2		Mugitu berehalako hitza erregistratzeko
B8 + rd	MOV reg32,imm32 2		Mugitu dword berehalako erregistratzeko
C6	MOV r/m8,imm8 2/2		Mugitu berehalako bytea r/m bytera
C7	MOV r/m16,imm16 2/2		Mugitu berehalako hitza r/m hitzera
C7	MOV r/m32,imm32 2/2		Mugitu berehalako dword r/m dword-ra

ŷ MOV-k EZ du inolako banderarik eragiten

ŷ dword :bikoitza: 32 bit

ŷ r8: 8 biteko erregistroa

ŷ r/m8 - edozein tamainatako edo 8 biteko memoria-kokapeneko erregistroa

ŷ imm8 : 8 biteko berehalako eragiketa

ŷ reg8: 8 biteko erregistroa

ŷ Sreg: segmentu-erregistroak ŷ CS,DS,ES,FS,GS,SS

ŷ Moffs8, moffs16 eta moffs32 eragigaien desplazamendu simple bat zehazten dute segmentuaren oinarriarekiko, non 8, 16 eta 32 datuen tamainari dagozkio. Instrukzioaren helbidea-tamaina atributua desplazamenduaren tamaina zehazten duzu, 16 edo 32 biteko.

<https://www.intel.es/content/www/es/es/architecture-and-technology/64-ia-32-architectures-software-developer-vol-1-manual.html>

### 3.7.4 Segmentu-hautatzailea zehaztea

Segmentu-hautatzailea implizituki edo esplizituki zehaztu daiteke. [...] The prozesadoreak automatikoki aukeratzen du segmentu bat Taulan emandako arauen arabera 3-5.

SS Oinarrizko erregistro gisa ESP edo EBP erregistroa erabiltzen duen edozein memoria erreferentzia.

DS Datu-erreferentzia guztiak, pila edo kate helmugarekin erlazionatuta daudenean izan ezik.

ŷ Eskuliburu teknika & Memoria sistema teknikak eta AT&T teknika Intel ŷ baizik

## 6.6. Oinarrizko mnemoteknia (azalduta)

### 6.6.1. Eragiketa aritmetikoak

- mul: zenbaki naturalen biderketa, zeinurik gabekoa

Lehenengo eragigaiaren (helmugako eragiketa) biderketa sinatu gabea egiten du eta bigarren eragigai (iturburuko eragiketa) eta emaitza helmugan gordetzen du operandoa Helmuga-eragigaien AL, AX erregistroan dagoen eragiketa implizitua da

edo EAX (eragigaiaren tamainaren arabera); iturburu-eragigaia erabilera orokorreko erregistro batean edo memoria-kokapen batean kokatzen da

- imul: zenbaki osodunen biderketa
  - ÿ 1,2 edo 3 eragiketa izan ditzake
  - ÿ imull Label : R[ %edx ]:R[ %eax ] ÿ M[Label] × R[ %eax ]

Bi eragigaiaren biderketa sinatua egiten du. Instrukzio honek hiru forma ditu, eragiketa kopuruaren arabera.

Eragiketa bakarreko forma — Forma hau MUL instrukzioak erabiltzen duenaren berdina da. Hemen, iturburu-eragigaia (erabilera orokorreko erregistroan edo memoria-kokapenean) AL, AX, EAX edo RAX erregistroko balioarekin (eragigaiaren tamainaren arabera) eta produktuarekin (sarrerako eragigaiaren tamainaren bikoitza) biderkatzen da. ) AX, DX:AX, EDX:EAX edo RDX:RAX erregistroetan gordetzen da, hurrenez hurren.

- div: zenbaki naturalen zatiketa, zeinurik gabekoa
- idiv: sinatutako zenbaki osoen zatiketa
  - ÿ 1,2 edo 3 eragiketa izan ditzake

AX, DX:AX edo EDX:EAX (dividendoa) iturburuko eragigaiaren (zatitzale) balioa (sinatua) zatitzen du eta emaitza AX (AH:AL), DX:AX edo EDX:EAX erregistroetan gordetzen du. . Iturburu-eragigaia erabilera orokorreko erregistro bat edo memoria-kokapen bat izan daiteke. Instrukzio honen ekintza eragigaien tamainaren araberakoa da (dibidendua/zatitzalea).

Emaitza ez-integralak 0 aldera mozten dira (txikituta).

ÿ idiv

ÿ EAXÿQuotient{[EDX:EAX]/M[Source\_Op]}, EDXÿRest{[EDX:EAX]/M[Source\_Op]}

#### 6.6.2. Sinadura luzapena

- movsbw src,Reg ÿ Mov Sign Byte Word-era
- movsbl src,Reg ÿ Mov Sign Byte to Long
- movswl rc,Reg ÿ Mov Sign Word to Long

#### 6.6.3. Tamaina aldatu

- movzbw src,Reg ÿ Mov byte Word-era
- movzbl src,Reg ÿ Mugitu byte luzera
- movzwl src,Reg ÿ Mugitu hitza luzera

## 6.6.4. Eragiketa boolearrak

- [EZ](#)

ÿ banderarik ez

- [ETA](#)

ÿ Garbitu CF,OF

ÿ Aldatu SF,ZF,PF

- [EDO](#)

ÿ Garbitu CF,OF

- [XOR](#)

ÿ Garbitu CF,OF

ÿ Aldatu SF,ZF,PF

## 6.6.5. Baldintzapeko tratamendua: CMP,TEST,SETcc

### CMP

- [CMP](#)

• CF,OF,SF,ZF,PF,AF aldatzen ditu

• Jcc, CMOVcc eta SETcc argibideek erabiltzen dituzten baldintza-kodeak baten emaitzetan oinarritzen dira. CMP argibideak

• Lehenengo iturburu-eragigaia bigarren iturburu-eragigaiarekin alderatzen du eta EFLAGS erregistroan egoera-markak ezartzen ditu emaitzen arabera. Konparazioa lehenengo eragiketari bigarren eragiketa kenduz eta, ondoren, egoera-markak ezarriz **SUB instrukzioaren** modu berean egiten da. Berehalako balio bat eragigai gisa erabiltzen denean, zeinua lutzatzen da lehenengo eragigaiaren luzera arte.

ÿ Hau da, SUB kenketaren instrukzioa interpretatzen jakin behar duzu eta, zehazki, sinatutako OF bandera. gainezkatzea eta CF sinatu gabeko gainezkatzearako.

ÿ Ariketak:

ÿ Egin **CMP instrukzio ariketa** eta baita jauziak aztertzen dituen praktika-gidoian ere  
CMP argibideen eskuz interpretatzea.

### AZPI

- [AZPI](#)

• Bigarren eragigaia (iturburuko eragigaia) lehenengo eragigaitik (helmugako eragigaia) kentzen du eta emaitza helmugako eragigaian gordetzen du. Berehalako balio bat eragigai gisa erabiltzen denean, zeinu-hedatu egiten da helmugako eragigai formatuaren luzera arte.

• SUB instrukzioak zenbaki osoen kenketa egiten du.

• Sinatu edo sinatu gabeko eragiketa osoen emaitza ebaluatzen du eta OF eta CF banderak ezartzen ditu sinatu edo sinatu gabeko emaitzan gainezka bat adierazteko, hurrenez hurren. SF banderak sinatutako emaitzaren zeinua adierazten du.

ÿ Hau da, kenketa egiten du bai eragigaiak zenbaki natural eta zeinudun gisa interpretatuz, emaitzaren irudikapen bitarra berdina baita. Gaintze kasuan, erabili OF bandera sinatutako interpretaziorako eta CF bandera sinatu gabeko gainezkatzeko.

**PROBA**• [PROBA](#)

- ÿ Lehenengo eragigaiaren (iturburu 1 eragigai) eta bigarren eragigaiaren bit-moduko **AND** logikoa kalkulatzen du.
- (2. iturria eragiketa)
- ÿ Garbitu CF,OF
- ÿ Aldatu SF,ZF,PF

**SETcc**• [SETcc](#)

- ÿ SETcc eragiketa
- ÿ Ez du banderarik aldatzen. Eragiketa aldatzen du baldintza betetzen bada.

**MOV**• [MOV](#)

- ÿ MOV instrukzioak ez du EFLAG erregistroko banderarik eragiten

**6.6.6. Jauziak****Baldintzak: Jcc**• [Jcc](#)

- ÿ Baldintza egiaztatzen du eta betetzen bada, eremuan erreferentziatutako helbidera jauzia exekutatzen da. funtzionatzen.
- ÿ salto laburra: eragigaiaren balioa PCarekiko erlatiboa da
- ÿ Egiaztatu CF, OF, PF, SF eta ZF banderak.
- ÿ "gutxiago" eta "handiagoa": konparatu sinatutako zenbakiak: jl, jle, jg, jge, etab...
- ÿ "goian" eta "behean": konparatu sinatu gabeko zenbakiak: ja, jae, jb, jbe, etab...

- Baldintzazko jauzi baten aurretik, behar-beharrezko da instrukzio motaren bat exekutatu non eragigaien erlazioa (aritmetikoa, logikoa, etab.) saltoa egiteko baldintza den. Aurreko argibideak hauek izan daitezke: CMP eta TEST.

**Zeharkakoa**

Ikurra \*:

Izartxoaren ikurra jauzietan zeharkakotasuna adierazteko eta helbide erlatibotik bereizteko.

```
jmp begizta -> EI Parekiko jauzia jmp
*begizta
jmp *eax
jmp *(eax)
jmp
*(mem) jmp *taula (%ebx,%esi,4)
```

Bestalde, MOV mugimenduetan ez da izartxoaren ikurra behar, ez baitago helbide erlatiboa duen mugimendurik.

## 6.6.7. Korritzea eta biraketa

- sar,sal : Eskuinera aritmetika desplazatu, ezkerrera aritmetika desplazatu.

ÿ desplazamendu aritmetikoa: ezkerretik edo eskuinetik sartzen den zifra zeinu-bit da.

Lehenengo eragigaiako (hel mugako eragigai) bitak ezkerrera edo eskuinera desplazatzen ditu bigarren eragiaian zehaztutako bit kopuruaren arabera (zenbaketa eragigai). Hel mugako operandoaren mugatik haratago desplazatutako bitak CF banderara desplazatzen dira lehenik, eta gero bantzten dira. Desplazamendu eragiketaren amaieran, CF banderak hel mugako eragiketatik desplazatutako azken bita dauka.

ÿ **sarl \$31, %edx** : 31 biteko eskuineko desplazamendua eta sarrerako bita eragiaiaren zeinu-bit da EDXn.

- shr,shl
- desplazamendu logikoa: zeroak sartzen dira
  - ÿ Biderketa eta zatiketaren adibideak
- ROL,ROR : ezkerreko biraketa eta eskuineko biraketa.
  - ÿ Irtetzen den bita CF-ra kopiatzen da
  - ÿ Aplikazioa: endianess bihurketa

## 6.6.8. Aldatu Endianess

```
## Endianess aldaketa EAX-en. Aurretik gorde jatorrizko EAX eta amaieran leheneratu EAX
```

```
swapbytes: xchg  
    (%ebx), %eax bswap %eax xchg  
    (%ebx), %eax
```

## 6.7. Argibide-formatua: ISA Intel x86-64

- Eranskina [Argibide-formatua](#)

## 6.8. Azpirrutinak

### 6.8.1. Erreferentziak

- [x86-32 ABI](#)
- [MicroSoft Call Convention](#)

### 6.8.2. Sarrera

- Mihiztadura-lengoiaian azpierrutinak mihiztadura-lengoiaian funtzioen baliokideak dira. programazioa C-n, beraz, C hizkuntzan funtzioko kontzeptua berrikustea beharrezkoa da.
- 5. praktika saioan, azpierrutinak programatuko dira.

### 6.8.3. C Hizkuntza: Funtzio adierazpena

#### Sarrera

Funtzioen helburua programa kode moduluetan deskonposatzea da, programa hornitzeko programaren garapena eta mantentzea errazten duen egitura antolatua. Liburutegi estandarra "libc" C lengoain garatutako oinarrizko funtzioen bildumak dira, gehienek berrerabilten dituztenak programen. Horrela programatzailak ez du gurpila asmatu beharrik. Beraz, programa batean erabiltzaileak berak C lengoain garatutako funtzioak eta liburutegiko funtzioak bertan eskuragarri kode bitarra.

#### Adierazpena

- C lengoain funtzio baten **adierazpenari** prototipo deritza . Prototipoaren adibidea: **int sumMtoN(gehiketa laburra1, batuketa laburra2)** non
  - ÿ Izena: funtzioaren izena sumMtoN da
  - ÿ Argudioak: lehen argumentuaren izena suming1 da eta labur motakoa da, 2.aren izena. argumentua summing2 da eta shrot motakoa da.
  - ÿ Itzultzeko balio-mota: itzultzeko balioaren mota int da.

#### Definizioa

- SumMtoN funtzioaren **definizioa** C sententzia erabiliz algoritmoa garatzean datza, hau da hau da, funtzioaren gorputza:

```
int sumMtoN (gehiketa laburra1, gehiketa laburra2) {
    //summing2 > summing1
    i laburra;
    int emaitza=0; // funtzioaren aldagai lokala
    i=gehitu2;
    bitartean (i >= gehituz1) {
        emaitza += i ;
        Yo--;
    }
    printf("\n\t Azpirrutina sumMtoN \n");
    emaitza itzuli;
}
```

ÿ emaitza itzuliko balioa duen aldagai da

#### Deitu eta Itzuli

- Main() funtzioak sumMtoN() funtzioari deitzen dio eta exekutatu ondoren emaitza itzultzen du batura.

```
/*
Programa: sumMtoN.c
Eraikitzea: gcc -g -ggdb3 -o sumMtoN sumMtoN.c
            -ggdb3 : informazioa txertatzen du arazketa-ikurren taulan
makroak
https://gcc.gnu.org/onlinedocs/gcc-12.2.0/gcc/Debugging
```

```

Options.html#Debugging-Options
*/



// Funtzio-prototipoak
#include <stdio.h> // printf() funtzioaren adierazpena
#include <stdlib.h> // Exit() funtzioaren adierazpena


//Makroak
#define ARRAKASTA 0


//Prototipoak: sumMtoN() funtzioaren deklarazioa
int sumMtoN(gehiketa laburra1, gehiketa laburra2);

// Funtzio Nagusiaren definizioa main()
void main(void) {
    //sumMtoN() funtzioaren M eta N argumentuen hasieratza
    laburra M=1, N=1, luzera;
    // printf() eta sumMtoN() funtzioetara deitu
    length=printf("Burketaren emaitza %d \n da", sumMtoN(M,N));
    printf("Goiko inprimatutako katearen karaktere kopurua da
%d\n",luzera);
    // sumMtoN-ren ebaluazioa honako hauek datza: funtzoari deitzea eta harrapatzea
    itzultzeko balioa.
    // Irteera funtziora deitu
    irten(ARRAKASTA);
}

// Funtzioen definizioa
int sumMtoN (gehiketa laburra1, gehiketa laburra2) {
    //summing2 > summing1
    i laburra;
    int emaitza=0; // funtzioaren aldagai lokala
    i=gehitu2;
    bitartean (i >= gehituz1) {
        emaitza += i ;
        Yo--;
    }
    printf("\n\t Azpirrutina sumMtoN \n");
    emaitza itzuli;
}

```

- to. sumMtoN()-ren ebaluazioa funtzioaren exekuzioaren itzulera-balioa lortzean datza  
**sumMtoN()**
- b. printf ý sumMtoN : printf-k sumMtoN-ren , sumMtoN ebaluatzen da eta ebaluazioaren emaitza inprimatzen da sumMtoN() funtzioa deitzen du.
- c. printf() funtzioaren itzulera balioa inprimatutako katearen karaktere kopurua da. Ireak a terminala eta exekutatu "man 3" eskuliburu "ITZULERAKO BALIOA" atala irakurtzeko.

#### 6.8.4. Funtzioa Habiaratzea

- Deien habiaratzea: init() ÿ main() ÿ sumMtoN() ÿ printf() ÿ write()
- GNU/linux sistema eragilearen shell-ak erabiltzailearen programaren main() funtzioari deitzen dio, eta horrek erabiltzailearen era berean, libc sumMtoN() liburutegiko printf() funtzioari eta , funtzioari deitzen dio. sistema eragilearen write() funtzioari dei egiten die.
- Itzulera:



#### 6.8.5. Pila/Markoa

- Ikusi pila kontzeptua Eranskinean.
- Pila memoria nagusian exekutatzen den programaren atal bat da. Datuen atalean eta instrukzioen atalean ez bezala, pila exekuzioan sortzen da, ez memorian kargatzean.
- sumMtoN() funtzioaren M eta N argumentuak main() funtziotik sumMtoN() funtziora pasatzen dira. pilaren bidez.
- Pila zatiketa markoetan: programa-funtzio bakoitzak bere eremu mugatua du pila-segmentuaren barruan. Funtzio bakoitzari lotutako eremu mugatu bakoitzari markoa deitzen zaio. Hori dela eta, pilaren atalean, fotogramak habitatuko dira funtzio bat deitzen den bakoitzean, eta fotogramak desagertu egingo dira funtzio bat itzultzen den bakoitzean.

ÿ main() funtzioak bere markoa sortzen du shellak deitzen duenean eta sumMtoN funtzioak berea sortzen du. markoa main(k) deitzen duenean.

ÿ Fotogramak azpierrutinen deiak habiaratu ahala pilatzen dira. Eta itzultzean desaktibatu egiten dira.

ÿ Dinamismoa: programaren exekuzioaren une jakin batean, sortutako azken fotograma fotograma da. ondasuna.

ÿ Marko aktiboaren beheko zatiari EBP erakusleak erreferentzia egiten dio eta markoaren goiko zatiari (goiko) ESP erakusleak. Baxua eta altua terminoek pila-helbideari egiten diote erreferentzia eta ez memoria-helbideei. Pila baxua eta altua.

#### 6.8.6. Azpirrutinen argumentuak

- Argudioak pila bidez eta deia egin aurretik transferitu behar dira.
- Argumentuak bata bestearen atzetik pilatzen dira azken argumentutik hasi eta amaieran lehen argumentua.
- Push argument instrukzioa erabiliz pilatzen dira, non eragigaia den transferitu beharreko argumentua.

bultzatu N  
bultzatu M

#### 6.8.7. Azpirrutina deia

- Deitzeko errutina nagusiak sumMtoN azpierrutina deitzen du dei bat sumMtoN sententzia erabiliz. Beraz, errutina nagusia eten egiten da sumMtoN azpierrutinaren exekuzioa amaitu arte.
- Deiaren instrukzioa bi fasetan exekutatzen da:
  - to. Itzultzeko helbidea pilatu: errutina nagusian honako adierazpena sumMtoN deitzeko: ESP ÿ ESP-4 eta M[EUSK] ÿ PC
  - b. Saltatu sumMtoN etiketara: PC ÿ sumMtoN
- Funtsean, dei-instrukzioa errutina eten zen helbidera itzultzeko jauzi bat da deitzalea.

```
push N  
push M  
deitu sumMtoN
```

#### 6.8.8. Azpirrutinaren definizioa

- Izena: sumMtoN
- Azpirrutinaren izena azpierrutinaren lehen adierazpena adierazten duen etiketa da.
- Azpirrutina ret instrukzioarekin amaitzen da.
- Azpirrutina 3 zatitan egituratuta dago:

ÿ Hitzaurrea:

Yo. Gorde azpierrutinaren gorputzak aldatuko dituen erregistroak.  
ii. Aktibatu marko berria EBP eta ESP erakusleak hasieratuz.

ÿ Gorputza:

Yo. Hartu argudioak eta prozesatu

ÿ Epilogoa:

Yo. Gorde itzuliko balioa EAX erregistroan  
ii. Berreskuratu Hitzaurrean gordetako erregistroen balioa  
iii. Aktibatu deia egin duen funtziaren markoa, EBP eta ESP beren zaharrekin egunearatuz  
balioak.  
iv. Itzuli deia egin duen funtziora.

- Kodea

```
# SumMtoN azpierrutinaren hasiera:  
  
# Prologue  
push %ebp # gorde deitzeko funtziaren markoaren behealdea hemen  
marko berria deskargatu  
mov %esp,%ebp # ezarri %ebp erakuslea fotograma berriaren behealdera seina dezan  
  
diren          # Beharrezkoa bada: gorde push xxx azpierrutinaren gorputza push xxx-n erabiliko  
erregistroak  
  
# Gorputza
```

```

    mov 8(%ebp),%ebx #kapturatu 1. argumentua
    mov 12(%ebp),%ecx #capture the 2. argument
    xxx xxx
    xxx xxx

# Epilogoa
    mov emaitza,%eax #initialize itzuliko balioa
    pop xxx          #berreskuratu fitxategian gordetako erregistroak
Hitzaurrea
    pop xxx
    mov %ebp,%esp
    pop %ebp
    ret

```

## 6.8.9. Animazia: habia-deiak

- <https://diveintosystems.org/book/C8-IA32/recursion.html>

ÿ Sum1toN planifikazioa programazio errekurtsiboaren bidez: batura azpierrutinak bere buruari deitzen dio.  
ÿ Deialdi eta itzulerekia deialdi eta itzulerekia pilaren hazkuntza eta gutxitzea bistaratzea.

## 6.8.10. Kontserbatzeko Erregistroak

### Deiak egiteko errutina: i386 arkitektura

- Dei-errutina beharrezkoa da ondorengo erregistroak gordetzeko erabiltzen ari zara:

ÿ EAX-ECX-EDX



Hau da, erroregistro hauek libreki erabil ditzake azpierrutina izenekoak. In Azpierrutinak erabiltzen ez baditu, ez litzateke beharrezkoa izango kontserbatzea. Izatearen kasuan errutina nagusiak erabilitako pilara kopiatuko litzateke deia egin **aurretik** azpierrutina eta azpierrutina amaitzean berreskuratuko litzateke.

- Adibidea:

```

    mov $0,%edx      # esleitu zero balioa EDX erroregistroari
    deien azpierrutina
    # EDX AZPIRUTINATIK itzultzean ez du zertan ZERO baliorik izan
    gehitu %edx,%edx # eragiketak ez du zertan 0+0 izan

```

ÿ irtenbidea:

```

    mov $0,%edx # esleitu zero balioa EDX erroregistroari
    push %edx # gorde EDX balioa pilan

    azpierrutina deitu

    pop %edx # berreskuratu EDX balioa pilatik
    gehitu %edx,%edx # egin 0+0 eragiketa

```

ÿ berdin EAX eta ECX erregistroetarako

### Azpirrutina izenekoa: i386 arkitektura

Deitutako azpierrutina (kalaren errutina) beharrezko da erregistro hauek gordetzeko:

- EBX-ESP-EBP-ESI-EDI eta X87CW

Hau da, azpierrutinaren amaieran horrelako erregistroek deiaren aurreko balio bera mantentzen dute. Erabiltzen ez baditzu, ez litzateke beharrezkoa izango kontserbatzea.

### amd64 arkitektura

Deitzaileen errutina: RAX, RCX, RDX, R8, R9, R10, R11 erregistroak lurrunkortzat hartzen dira eta funtzi deietan suntsituzat jo behar dira (salbu eta programa osoaren optimizazioa bezalako segurtasuna frogatzen ez bada).

Callee errutina: RBX, RBP, RDI, RSI, RSP, R12, R13, R14 eta R15 erregistroak lurrunkortzat hartzen dira eta horiek erabiltzen dituen funtzi batek gorde eta berreskuratu behar ditu.

### 6.8.11. Azpirrutinaren itzulera

- Azpirrutinaren azken instrukzioa **RET** da , zeinaren exekuzioa PUZaren Kontrol Unitateak egiten duen agindu hauek:
  - to. **PC** ÿ **M[ESP]** : **CALL** instrukzioak gordetako itzulera-helbidea pilatik ateratzen du eta Programa-kontagailuan kargatzen du, beraz, **sumMtoN** deiaren ondoren instrukzioaren instrukzio-zikloa exekutatuko da.
  - b. Eguneratu pila erakuslea: **ESP** ÿ **ESP + 4**
- Beharrezko da azpierrutinaren epilogoa, RET exekutatu aurretik, pila-erakusleak seinalatzea. Itzultzeko helbidea gordetzen den pilaren helbidea.

### 6.8.12. Bateriaren egoera

#### Analisia

- Pila egitura dinamiko bat da, zeinaren egoera (erakusleen erregistroak EIP, EBP, ESP) aldatzen den azpierrutinen deiak eta itzulerak egiten diren heinean. Arrazoitu aipatutako erregistroen edukia eta egiaztatu GDB araztearekin.

#### Deitik azpierrutinara salto egin aurretik

- Deia **sumMtoN instrukzioa** exekutatu aurretik errutina nagusia exekutatzen duen pilaren egoera :
  - ÿ Pilaren marko aktiboa nagusiari dagokiona da.
  - ÿ Marko nagusian pilatutako azken datuak sumMtoN-ren argumentuak dira

```
push N  
push M  
deitu sumMtoN
```

- EIP, EBP, ESP erregistroen edukia aztertzea:

ÿ EIP: instrukzio erakuslea  
ÿ EBP: pilatu beheko erakuslea

ÿ ESP: pilatu goiko erakuslea ÿ

Deitik azpierrutinara jauzi egin ondoren

- SumMtoN azpierrutina exekutatzen duen pilaren egoera sumMtoN dei-jauzia exekutatu ondoren:

ÿ Pilaren marko aktiboa nagusiari dagokiona da.

ÿ Marko nagusian pilatutako azken datuak sumMtoN-tik nagusira itzultzeko helbidea da

- EIP, EBP, ESP erregistroen edukia aztertzea:

ÿ EIP: instrukzio erakuslea ÿ

ÿ EBP: pilatu beheko erakuslea ÿ

ÿ ESP: pilatu goiko erakuslea ÿ

SumMtoN marko berriaren sorrera

- Pila-egoera exekutatu ondoren:

sumMtoN:

bultzatu %ebp

mugitu %esp,%ebp

- EIP, EBP, ESP erregistroen edukia aztertzea:

ÿ EIP: instrukzio erakuslea ÿ

ÿ EBP: pilatu beheko erakuslea ÿ

ÿ ESP: pilatu goiko erakuslea ÿ

Itzulerako jauzia baino lehen

- Pila egoera sumMtoN azpierrutina exekutatzen ret instrukzioa exekutatu aurretik:

ÿ Pilaren fotograma aktiboa sumMtoN-ri dagokiona da.

ÿ SumMtoN markoaren goiko ESP erakusleak pila helbidea duen pila-helbidera seinalatzen du.  
itzuli

- EIP, EBP, ESP erregistroen edukia aztertzea:

ÿ EIP: instrukzio erakuslea ÿ

ÿ EBP: pilatu beheko erakuslea ÿ

ÿ ESP: pilatu goiko erakuslea ÿ

Itzulera saltoaren ostean

- Pila egoera sumMtoN azpierrutina exekutatzen ret instrukzioa exekutatu ondoren:

ÿ ret exekuzioak eragiketa hauek egin ditu:

ÿ pop %irp

ÿ Pilaren marko aktiboa nagusiari dagokiona da.

- EIP, EBP, ESP erregistroen edukia aztertzea:

ÿ EIP: instrukzio erakuslea ÿ

ÿ EBP: pilatu beheko erakuslea ÿ

ÿ ESP: pilatu goiko erakuslea ÿ

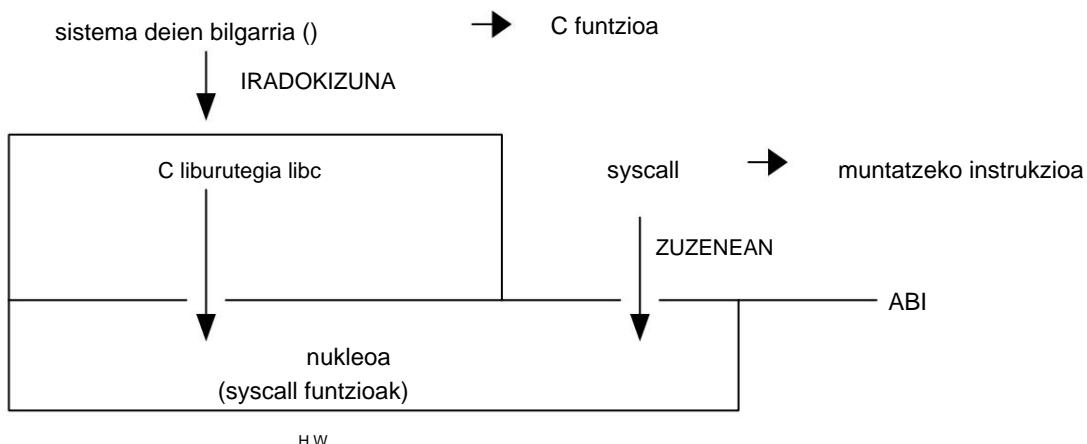
### 6.8.13. Programazio ariketa praktikoak

- Joan Ariketak kapitulura

### 6.9. Sistema eragilera deiak

#### 6.9.1. Sarrera

- Erabiltzaile-programak sistemara egiten dituen deiei sistema-deiak deitzen zaie. Sistema eragilearen Kernelaren azpierrutinak.
- Sistema eragilearen funtziopribilegiatuak betetzeko, hala nola, ordenagailuko I/O gailuetarako sarbidea, beharrezko da erabiltzailearen programek nukleoa deitzea, eragiketa modu seguru eta eraginkor batean egin dezan. Honek aplikazioaren programatzalea hardwarera sartzea eragozten du eta, aldi berean, programazioa errazten du.
- Deien adibideak
  - ÿ irten: nukleoak programaren exekuzioa eten egiten du prozesua hilez
  - ÿ irakurri: nukleoak fitxategi bateko datuak irakurtzen ditu disko gogorrean sartuta
  - ÿ idatzi: nukleoak fitxategi batean idazten du
  - ÿ ireki: nukleoak fitxategi bat irekitzen du
  - ÿ itxi: nukleoak prozesua ixten du
  - ÿ dei-adibide gehiago man 2 syscalls zerrendan
- Syscalls izeneko nukleo-zerbitzuei deia bi modutara egin daiteke: zuzenean edo iradokizuna
  - ÿ Zuzena: ASMtik syscall sententzia erabiliz
  - ÿ Zeharkakoa: C edo ASMtik libc liburutegiko funtziointerfaceak erabiliz: zuzeneko deietarako bilgarriak
- API/ABI



#### 6.9.2. amd64 arkitektura

## Dei-kodeak

- '/usr/include/asm/unistd\_64.h': makro-adierazpena x86 arkitekturako dei-kodearekin 64
- /usr/include/bits/syscall.h : makro zaharrak x86-32 arkitekturan ere balio dute

## Argudioak

- Amd64 arkitektura adibidea
  - ÿ Argumentuak pasatzeko, pila (kanpoko memoria) ez da erabiltzen baina RPG erregistroak (kanpoko memoria) barne CPU)
  - ÿ Lehenengo 6 argumentuak erregistroetatik pasatzen dira: [RAX-RDI-RSI-RDX-R10-R8-R9] syscall instrukzioa baino lehen.

```
* printf() -> write(int fd, const void *buf, size_t count) -> [RAX-RDI-RSI-RDX
R10-R8-R9,syscall] -> kernel syscall idazketa-bilgarri
* API          ->          funtzioa                                ->          ABI
-> kernel syscall
```

## amd64 adibideak

- Ikusi Eranskinean

### 6.9.3. i386 arkitektura

#### Arau

- ABI SystemV i386: Application Binary Interface (ABI)
  - ÿ ABI SystemV i386 estandarra interfaze bitarra deskribatzen duen estandar ofiziala da, hau da, programa baten modulu desberdinak maila bitarrean, makina hizkuntza mailan, nola komunikatzen diren.
  - ÿ Nukleorako deia int \$0x80 instrukzioa erabiliz egiten da, non int etenaren mnemonikoa den (exekutatzen ari den erabiltzailearen programa eten egiten da nukleoaren funtzio bat deitzeko)

### 6.9.4. Dei-kodeak

- /usr/include/asm/unistd\_32.h : makro-adierazpena dei-kodearekin
  - ÿ /usr/include/bits/syscall.h: makro zaharrak ere balio du
- ohiko deiak:
  - ÿ irten-sardexka-irakurtzea-idatzi-ireki-itxi-...

irten - uneko prozesua amaitu		
unean	eax	1
	ebx	itzultzeko kodea
Kanpoan	(Dei hau ez da itzultzen)	
sardexka - sortu haur-prozesua		
unean	ea 2	
Out eax 0 klonean; klonaren ID prozesua edo EAGAIN edo ENOMEM deitzailean		

		irten - uneko prozesua amaitu
		irakurri - irakurri fitxategitik edo gailutik
urtean	ea 3	
		ebx fitxategi deskribatzailea
		Buffer-aren ecx helbidea irakurtzeko
		edx irakurtzeko gehienezko byte kopurua
		Byte-kopuru bakoitzeko gehienez irakurtzen da: EAGAIN : EBADF : EFAULT : EINTR : EINVAL : EIO : EISDIR
		idatzi - idatzi fitxategian edo gailuan
urtean	ea 4	
		ebx fitxategi deskribatzailea
		Idazteko buffer-aren ecx helbidea
		edx idazteko gehienezko byte kopurua
		Benetan bidalitako byte kopurua: EAGAIN : EBADF : EFAULT : EINTR : EINVAL : EIO : ENOSPC : EPIPE
		ireki - ireki, sortu edo moztu fitxategi edo gailu bat
urtean	ea 5	
		zero amaierako bide-izenaren ebx helbidea
		ecx fitxategirako sarbide-bitak
		edx fitxategiarengan baimen modua
		Irekitako fitxategiarengan eax fitxategiarengan deskribatzailea : EACCESS : EEXIST : EFAULT : EISDIR : ELOOP : EMFILE, : ENAMETOOLONG : ENFILE : ENOENT : ENODEV : ENODIR : ENOMEM : ENOSPC : ENXIO : EROFS : ETXTBSY
		itxi: fitxategi edo gailu bat itxi
urtean	ea 6	
		ebx fitxategi deskribatzailea
		Out eax zero arrakasta lortzeko: EBADF
		waitpid - itxaron prozesu bat amaitu arte
urtean	ea 7	
		Itxaron beharreko prozesuaren ebx process id
		ecx 0, edo buffer-aren helbidea irteera-egoera mantentzeko
		edx aukera banderak : 0 : WNOHANG : WUNTRACED
		Amaitutako prozesuaren pid bakoitzeko atera : ECHILD : EINVAL : ERESTART
		ecx amaitutako prozesuaren irteera egoera, ecx-n balio ez nulua sartu bada
		sortu - fitxategi bat sortu
urtean	ea 8	
		zero amaierako bide-izenaren ebx helbidea
		ecx fitxategiarengan baimen modua

<b>irten - uneko prozesua amaitu</b>		
		Irekitako fitxategiaren eax fitxategiaren deskribatzailea : EACCESS : EEXIST :EFAULT : EISDIR : ELOOP : EMFILE, : ENAMETOOLONG : ENFILE : ENOENT : ENODEV : ENODIR : ENOMEM : ENOSPC : ENXIO : EROFS : ETXTBSY
Oharra Dei hau open deiaren berdina da sarbide-bitekin O_CREATE: O_WRONLY: O_TRUNC		
<b>esteka - fitxategi baterako esteka gogor bat sortu</b>		
urtean	ea 9	
		Lehendik dagoen fitxategi-izenaren zero amaierako bide-izena ebx helbidea
		izen berriaren zero amaierako bide-izena ecx helbidea
Out eax 0 : EACCESS : EIO : EPERM : EEXIST :EFAULT : ELOOP : EMLINK : ENAMETOOLONG : ENOENT : ENOMEM : ENOSPC : ENOTDIR : EPERM : EROFS : EXDEV		
<b>unlink - ezabatu izen bat eta kendu fitxategia lanpetuta ez dagoenean</b>		
urtean	eax	10
		Lehendik dagoen fitxategi-izenaren zero amaierako bide-izena ebx helbidea
Out eax 0 : EACCES :EFAULT : EIO : EISDIR : ELOOP : ENAMETOOLONG : ENOENT : ENOMEM : ENOTDIR : EPERM : EROFS		
<b>execve - programa bat exekutatu</b>		
urtean	eax	namika
		Programaren zero amaierako bide-izenaren ebx helbidea
		zero amaierako ecx helbidea zero amaierako helbideen zerrenda argumentu-kateak
		zero amaierako inguruneko kateen helbide zerrendaren edx helbidea
Kanpoan eax		Arrakasta bada, ez da bueltarik programa berriak baliabideak heredatzen dituelako eta deitzailea gainidazten duelako; bestela: E2BIG : EACCES : EINVAL : EIO : EISDIR : ELIBBAD : ELOOP : ENFILE : ENOEXEC : ENOENT : ENOMEM : ENOTDIR :EFAULT : ENAMETOOLONG : EPERM : ETXTBUSY
<b>chdir - aldatu lan-direktorioa</b>		
urtean	eax	12
		Lehendik dagoen direktorioaren zero amaierako bide-izena ebx helbidea
Out eax 0 : EACCES : EBADF :EFAULT : EIO : ELOOP : ENAMETOOLONG : ENOENT : ENOMEM : ENOTDIR		

• Fitxategien deskribatzaileak

ÿ 0 (STDIN): terminaleko gailuaren sarrera estandarra (normalean teklatua). Makroa desberdindu stdin makroaren STDOUT\_FILENO.

ÿ 1 (STDOUT): terminaleko gailuaren irteera estandarra (normalean terminalaren pantaila). Bereizi STDOUT\_FILENO makroa stdout makrotik.

ÿ 2 (STDERR): terminaleko gailuaren errore estandarraren irteera (normalean terminalaren pantaila)

## Argudioak nola pasa

- Argumentuak pasatzeko, pila (kanpo memoria) ez da erabiltzen, RPG erregistroak (barne memoria) baizik. CPUarena)
- Lehenengo 6 argumentuak erregistroetatik pasatzen dira: [EBX-ECX-EDX-ESI-EDI-EBP] deitzeko instrukzioa baino lehen int \$0x80

```
* printf() -> write(int fd, const void *buf, size_t count) -> [EBX-ECX-EDX-ESI-EDI  
EDI,int 0x80] -> kernel syscall idazketa  
*API bilgarri funtzioa  
-> nukleoaren syscall
```

-&gt;

ABI

- Pasatu 1.-2.-3.-4.-5.-6. argumentuak EBX-ECX-EDX-ESI-EDI-EBP erregistroetatik ordenan. C-ko funtziaren prototipoan adierazita:
- Adib.: ASM hizkuntzan iturburu-modulu batetik nukleoko syscall irteerarako deia:
  - ÿ gizon 3 irteera
  - ÿ syscall (irteera\_kodea, int egoera)
  - ÿ asm modulua

```
mov $1,%eax  
mov $status_value,%ebx int  
$0x80
```

- Adib.: nukleoaren syscall idazteko deia iturburu-modulu batetik ASM hizkuntzan:
  - ÿ gizon 2 idatzi
  - ÿ syscall(write\_code,int fd, const void \*buf, size\_t count)
  - ÿ asm modulua

```
mov $4,%eax  
mov $1,%ebx  
mov $buffer_address_label,%ecx mov  
tamaina,%edx int  
$0x80
```

- Itzultzeko balioa

ÿ Itzultzeko balioa EAX erregistrotik pasatzen da

## C liburutegi estandarreko funtziotarako deiak

- ASMrak Linux-en instalatutako C liburutegiaren funtziotara deitu dezakezu: libc
- printf() funtziari deitzeko printf deiaren instrukzioa erabiltzen dugu
- Funtziaren argumentuak deiaren aurretek pasatzen dira.
  - ÿ Argumentuak pilatik pasatzen dira Eskuneko ÿ Ezkerreko norabidean definitutako moduan.
- Objektu modulua libc liburutegiarekin lotzea beharrezkoa da
- Adib.:

ÿ C programazioa

```
planetak = 9;
printf(" Planeta kopurua %d \n da", planetak);
```

ÿ ASM programazioa

```
.atala .datuak
katea:
.asciz " Planeta kopurua % d da \n"
planetak:
.luzea 0
.atala .testua
_hasi:
ÿ movl $9, planetak
bultzatu planetak
push $string
deitu printf
dei irteera
.amaiera
```

## Komando-lerroa

- Prozesuaren Hastapena
- Nukleoan main() funtzioa deklaratzen da: **extern int main (int argc , char\* argv[], char\* envp[]);**
  - ÿ Modulu nagusiaren adierazpena eta definizioa
  - ÿ argc argumentu-zenbaketa ez-negatiboa da;
  - ÿ argv argumentu-kateen array bat da, argv[argc]==0 duena;
  - ÿ envp ingurune-kateen array bat da, erakusle nulu batekin amaitzen dena ere.
- Pila hasieratzea

12. taula. ABI Hitzarmena: Pila

StackReference	Interpretazioa
	argumentu-kateak
4n(%esp)	- kate-zenbakiaren erakuslea
8(%esp)	- 2. argumentuaren erakuslea katea
4(%esp)	- 1. argumentuaren erakuslea katea
0(%esp)	- argc

## 6.9.5. Adibidea

- Sartu sum.sa programaren datuak komando lerroaren bidez

```
### funtzioa: gehitu zifra bateko bi zenbaki oso.  
### gehigarriak komando lerroaren bidez pasatzen dira  
## gcc -m32 -nostartfiles -g -o sum_input sum_input.s  
## korrika 5 7  
## x /a (char**)($esp+4) -> 0xfffffd0a4: 0xfffffd26e  
## x /c *(char**)($esp+4) -> 0xfffffd26e: 47 '/'  
## x /s *(char**)($esp+4) -> 0xfffffd26e:  
"/home/candido/tutoriales/as_tutorial/algoritmos_x86-32/basicos/sum_input"  
## p /s *(char**)($esp+4) -> 0xfffffd26e  
"/home/candido/tutoriales/as_tutorial/algoritmos_x86-32/basicos/sum_input"  
## x /s *(char**)($esp+8) -> 0xfffffd2b7: "5"  
## x /s *(char**)($esp+12) -> 0xfffffd2b9: "7"  
  
.atala .testua  
  
.globl _hasi  
_hasi:  
  
## azalpen argibideak  
  
leial 8(%esp),%eax #eax-ek argv[1] dauka pila-helbidea  
katearen argumentuaren erakuslea dauka  
    movl 8(%esp),%ebx          #ebx-ek pila-edukia = helbidea du  
katea  
    xor %ecx,%ecx  
    movb (%ebx),%cl            #ASCII karakterea  
  
## katearen argumentu erakusleak  
    movl 8(%esp),%eax katea.      #eax-ek pila-edukia du = helbidea  
argv[1]  
    movl 12(%esp),%ebx katea.     #eax-ek pila-edukia du = helbidea  
argv[2]  
    ## lortu katea zeharka  
    ## Ascii zenbakiek balio bihurtu  
    xor %ecx,%ecx  
    xor %edx,%edx  
    movb (%eax), %cl argv        # zeharka erreferentziazko kateari atzitzeko  
[1]  
    movb (%ebx),%dl argv         # zeharka erreferentziazko kateari atzitzeko  
[1]  
    subl $0x30,%ecx  
    subl $0x30,%edx  
  
    bultzatu %ecx  
    bultzatu %edx  
  
deitu batura  
  
## irten  
    movl %eax,%ebx  
    movl $1, %eax               #1 exit() syscall da
```

```
int $0x80
```

### Bi **balioen** arteko maximoa kalkulatzen duen  
**funtzioa.mota** batura ,  
@funtzioa.atala .testua

gain:

ÿ ## prologue  
push %ebp  
movl %esp,%ebp  
subl \$1,%esp #memoria erreserba  
push %ebx  
push %edi  
push %esi ##  
argudioen harrapaketa movl  
8(%ebp),%eax #1st argument movl 12(%ebp),  
%ecx #2. argument ## body addl %ecx,%eax  
## gorde  
emaitza #

ÿ ## emaitza **EAX** -en dagoÿ ##

epilogo pop  
%esi pop  
%edi pop  
%ebx mov  
%ebp,%esp pop # aurreko fotograma  
%ebp  
ret # berreskuratu itzulera helbidea

## II Oinarrizko Unitateak: Prozesadore Zentrala, Memoria Unitatea, Sarrera/Irteerako Mekanismoak.

# 7. kapitula. Prozesadore zentrala

## 7.1. Ikastaroa

7. Von Neumann IAS eta Intel 4004 Ordenagailuen arkitektura eta antolaketa.

to. Berrikuspena

8. PUZaren arkitektura eta antolaketa

to. Argibide multzoa

b. CISC, RISC eta VLIW arkitekturak

c. Argibideen exekuzio faseak

d. datuen bidea

## 7.2. Erref

- Oharrak: 2. gaia: von Neumann arkitektura (kontrol-unitatea)
- Testu liburua: Ordenagailuen egitura eta antolaketa. William Stalling. 12. kapitula.

## 7.3. Sarrera

- Berrikuspen gisa eta gaiaren sarrera gisa, 1951tik gaur arte Von Neumann ordenagailua erreferentziazko arkitektura gisa eta Intel-en lehen prozesadore integratua komertziala, 4004 cputa, 1971n ikusiko dugu. Gaur egungo gainerako PUZak dira. aipatutako prozesadoreen bilakaera.
- PUZaren helburu nagusia instrukzio-zikloaren ezarpena da. Hardwarearen euskarria da programa baten jarraibideek dakarren eragiketa guztiak egiteko gai izatea.
- Prozesatzeko Unitate Zentrala (CPU) edo Prozesadorea.
  - ÿ Izenak: Prozesadorea, mikroprozesadorea (Oinarrizko osagai elektronikoa bere jatorrian mikra ordenako tamaina duen transistorea da), ..
  - ÿ Zentrala ordenagailuak hainbat prozesadore dituelako: Adibidez memoria kontrolatzailea eta gidari periferikoak.
- Von-Neumann Arkitektura.
  - ÿ CPU Von-Neumann arkitektura osatzen duten oinarrizko unitateetako bat da (CPU-MP-IO) eta Autobusak.
- PUZA programatzailaren ikuspuntutik edo prozesadorearen diseinatzailaren ikuspuntutik (mikroarkitektura) ikus daiteke.
  1. Programatzailaren ikuspuntutik, interesgarria da jakitea:
    - ÿ Instruction Repertoire Architecture (ISA). Gaur egun prozesadoreak CISC eta RISC arkitekturetan ISAREN konplexutasunaren edo simpletasunaren arabera sailka daitezke. ISA CISC (Complex Instruction Set Computer) arkitekturaren adibide bat Intel x86 arkitektura da ordenagailu pertsonaletan oso erabilia eta ISA RISC (Complex Instruction Set Computer) arkitekturaren adibide bat telefono mugikorretan oso erabiliak diren ARM prozesadoreak dira.
    - ÿ Erregistroak: programatzailak eskura ditzakeen helburu orokorreko erregistroak (metagailuak, indize-erregistroa, pila-erakusleak, etab.), egoera-erregistroak, koma mugikorreko erregistroak, multimedia-erregistroak, memoria-segmentazio-erregistroak, eskurazinak diren erregistroak, hala nola programa-kontagailua, erregistro-tamaina, etab.

ŷ PUZaren funtzionamendu moduak: supererabiltzaile modua, erabiltzaile modua, eten modua. Funtzionamendu modu ezberdinek CPUren funtzionamendu egokia babesteko aukera ematen dute erabiltzaile, administratzaile, baliabideak partekatzeko eta abar egiteko lanak egiten diren ala ez.

## 2. PUZaren Mikroarkitekturen edo barne antolaketaren ikuspuntutik.

- ŷ PUZaren oinarrizko unitateak hauek dira: Kontrol Unitatea (UC), eta Datuen Bidea (ALU, FPU, MMU, Erregistroak eta bideratz-ezirkuituak, hala nola multiplexadoreak, etengailuak, etab.).
- ŷ PUZaren mikroarkitekturek datu-bide bat baino gehiago implementatu ditzake, horrela instrukzio bat baino gehiago aldi berean exekutatzeko aukera ematen du. Teknologia hori erabiltzen duten prozesadoreei prozesadore supereskalarak deitzen zaie.
- ŷ Instrukzio-zikloa sekuentziala edo segmentatua izan daiteke, eta exekuzio-denboran instrukzio bat baino gehiagoren gainjartzea ahalbidetzen du (paralelismo-teknikak instrukzio-mailan, ILP)

### • Programa baten exekuzioa optimizatzeko HW teknikak (**Performance**)

ŷ PUZaren bertsio berriak diseinatzerakoan oinarrizko helburua bere errendimendua handitzea da, hau da, denbora-unitateko exekutatzeko diren instrukzio kopurua eta, beraz, programen exekuzio-denbora. Konputagailuen mikroarkitektura mailan hainbat teknika garatu dira helburu horrekin, hala nola:

- ŷ **Segmentazioa-Pipelining**: instrukzio-zikloa fase edo segmentutan antolatu eta exekutatu paraleloa.
- ŷ **prozesadore supereskalarra**
- ŷ **Aginduz kanpoko exekuzioa OoO**: exekutatzeko denbora
- ŷ **Grabatu izena aldatu**: Konpilatzalea eta exekuzio denbora
- ŷ **Adarren iragarlea**: exekutatzeko denbora



34. Irudia. Ikuspegiak: SW eta HW

## 7.4. Ordenagailua Programatzailaren ikuspuntutik (I)

### 7.4.1. Argibide multzoa

#### Arkitektura (ISA)

- Irakasgaiaren lehen zatiaren oroigarria:
  - ŷ Gaiak: von Neumann arkitektura, datuen irudikapena, eragiketa aritmetiko-logikoak, jarrabideen eta programazioaren irudikapena mutua-lengoian.
- Instruction Set Architecture (ISA)
  - ŷ Instrukzio-multzoaren arkitekturak definitzen ditu: eragiketa kodeak, eragiketa motak, helbideratze moduak, etab.

ŷ PUZak zuzenean exekuta daitezkeen makinaren instrukzioak dira kode bitarrean: hizkuntza  
makina

ŷ Exekutatu beharreko instrukzioa kode bitarrean gordetzen da Kontrol Unitateko RI erregistroan.

- Argibideen errepertorioa PUZaren programatzailearen eskuliburuan zehazten da:

ŷ Makina lengoian beharrean mutuaia hizkuntzan programatzen dugu

ŷ Eskuliburuak Instrukzio Errepertorioaren Arkitekturaren definizioa jasotzen du.

ŷ mikroprozesadoreak exekuta ditzakeen argibide guztien zerrenda eta deskribapena

ŷ instrukzio kategoriak: transferentzia (mov), kontrola (jmpz, begizta), aritmetika (gehitu),  
logikoa (xor), i/o (sartzea/irtena)

ŷ Opcode mnemoteknia

ŷ Helbideratzeko moduak: berehalakoa, zuzena, zeharkakoa, desplazamendua

ŷ Datu motak: osoak, errealkak, alfanumerikoak

ŷ Formatu bitarrak

ŷ Argibideak: eragiketa-eremuak, eragigaiak, helbideratze-modua

ŷ Datuetatik: 2-ren osagarria, koma mugikorra

## 7.4.2. Adibideak: Intel x86, Motorola 68000, MIPS, ARM

- Ikus [Eranskina Batzar Lengoaiak](#)

# 7.5. Mikroarkitektura: Unitate Funtzionalak HW ikuspegitik

## 7.5.1. Sarrera

- Mikroprozesadorearen barne arkitektura mikroarkitektura izenez ezagutzen da.

ŷ Mikroarkitektura ISAK definitutako instrukzio-multzoaren instrukzio-zikloaren diseinua eta ezarpena da.

ŷ Adibideak

ŷ IAS

ŷ Intel: 4004, 8008, 8051, x86

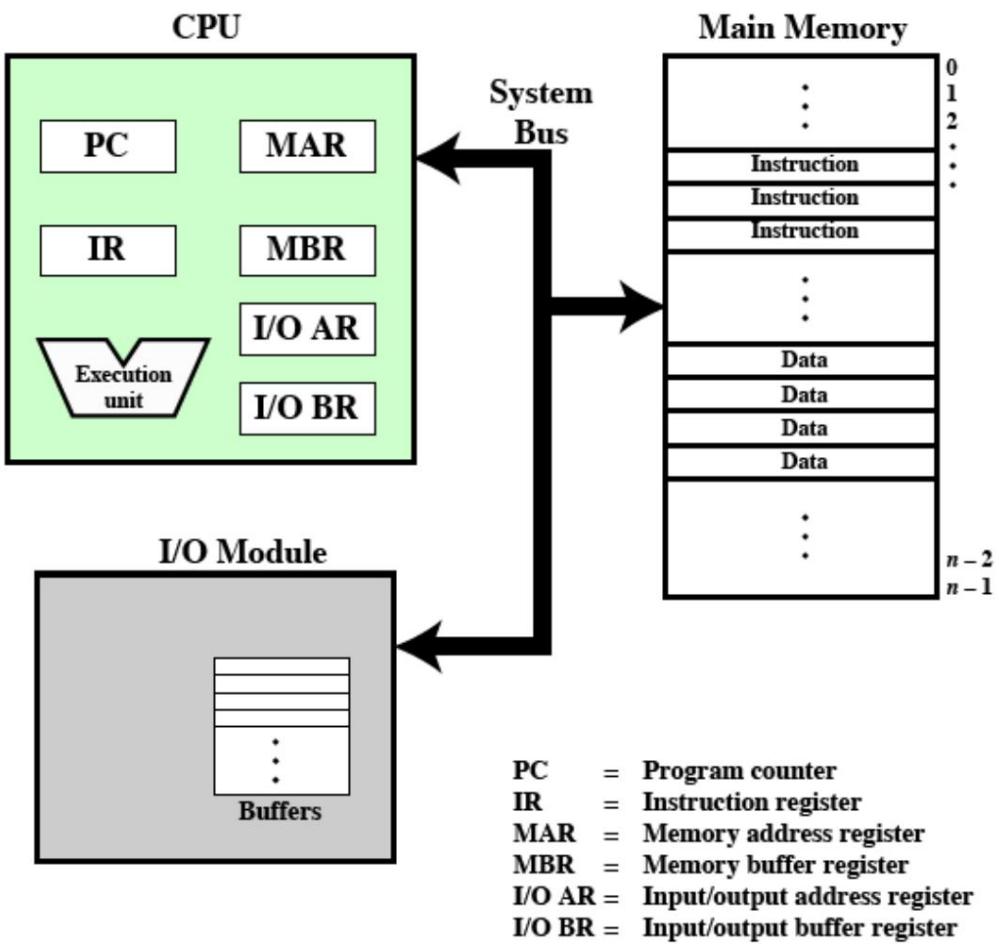
ŷ AMD: x86

ŷ BESOA: Cortex

- Mikroaurrizkia (mikroelektronika, mikroprozesadorea, mikroarkitektura, mikrobusa, mikroprograma, mikrokodea, mikroeragiketa, etab.) lehen zirkuitu edo txip integratuen transistoreen tamainarekin erlazionatuta dago, hamarnaka MIKROMETROren ordenakoak baitziren. Dimentsio horretan informatika-ingurunean gertatzen den guztia mikroa jarri ohi da

## 7.5.2. Instrukzioa gauzatzeko faseak

### Egitura



35. Irudia IAS\_Arkitektura

Zikloa / Diagrama / Fasak

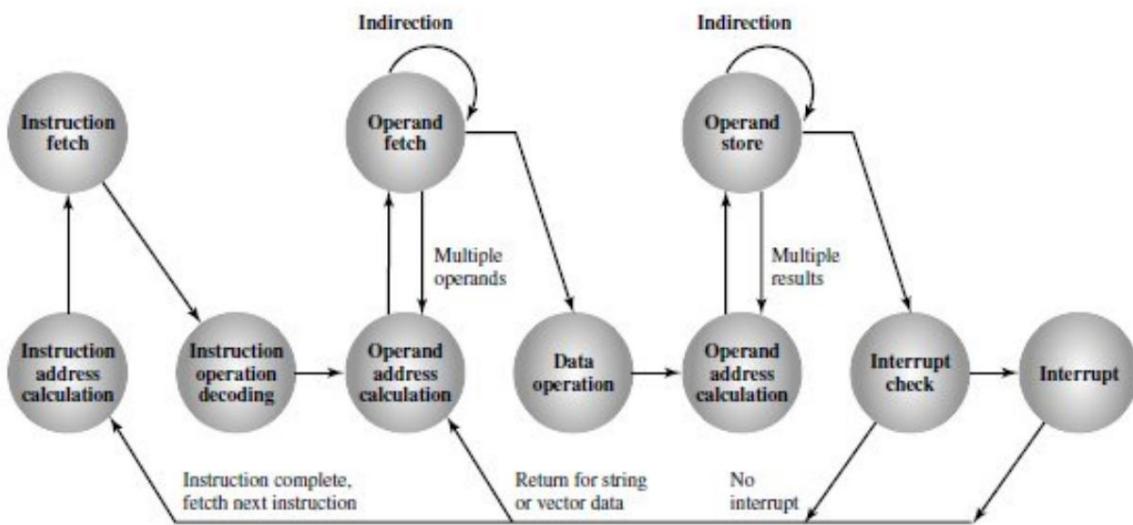


Figure 12.5 Instruction Cycle State Diagram

## 36. Irudia. Egoera-diagrama

- Zikloak: Instrukzio Zikloa Faseen edo Makina Zikloen sekuentzia bat da. Makina-ziklo bakoitza PUZaren erloju-ziklo batean edo gehiagotan exekutatu daiteke.
- Instrukzio-zikloaren edo makinen zikloaren faseak:
  1. Eskuratu Instrukzioa: FI
    - ÿ Hasieran, Kontagailuaren edukia memoria helbide-busera bota behar da.
    - Programa (PC)
    - ÿ Harrapatu instrukzioa
    - ÿ PC ÿ PC+1
  2. Argibideen Deskodeketa: ID
    - ÿ Instrukzioa interpretatzea
  3. Eskuratu operandoa: OF
    - ÿ datuak harrapatzea, eragigaiak harrapatzea
    - ÿ helbide eraginkorra ebatzi
  4. Instrukzioa Exekutatu : EI
    - ÿ instrukzioa daturekin prozesatu
  5. Idatzi operandoa: WO
    - ÿ Emaitza gorde
    - ÿ helbide eraginkorra ebatzi
  6. Etenaldia : II
    - ÿ Programa eten daiteke arreta periferikoko beste programa bat exekutatzeko lehentasunagatik, etab. Etenaldiari erantzuna ematen zaionean, programak hurrengo instrukzio-zikloarekin jarraitzen du.
  7. Hurrengo instrukzioa: NI

- Irakaskuntza-zikloa

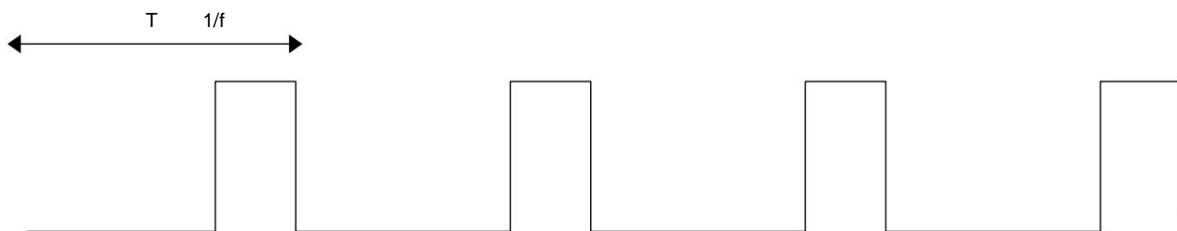
- Instukzioen harrapaketa fasearen (FI), Exekuzio fasearen (EI) edo Operandoaren Helbide Eraginkorra zehaztea eta Eragigaiaren Lorpena (OF)
- Exekuzio-fasearen ondoren etenaldi baten arreta-begizta egon daiteke.

## Mikroeragiketen Diagrama

- Mikroeragiketak: PUZak barnean egiten dituen eragiketak, Makina-Instrukzio bat exekutatzen denean.

Adibideak: MAR erregistroan idatzi, MPrincipal-era irakurri ordena, MBRtik irakurri, -tik interpretatu. IR, PC handitu, etab.

Exekuzio sinkronikoa PUZaren erlojuarekin:



Erlojuaren ertzak: maila aldaketa 0→1 (positibo) edo 1→0 (negatiboa)

IAS ez da sinkronoa: mikroeragiketa bat ez da inolako denbora-eredurekin hasten.

Mikroeragiketen deskribapena: Register Transfer Language (RTL)

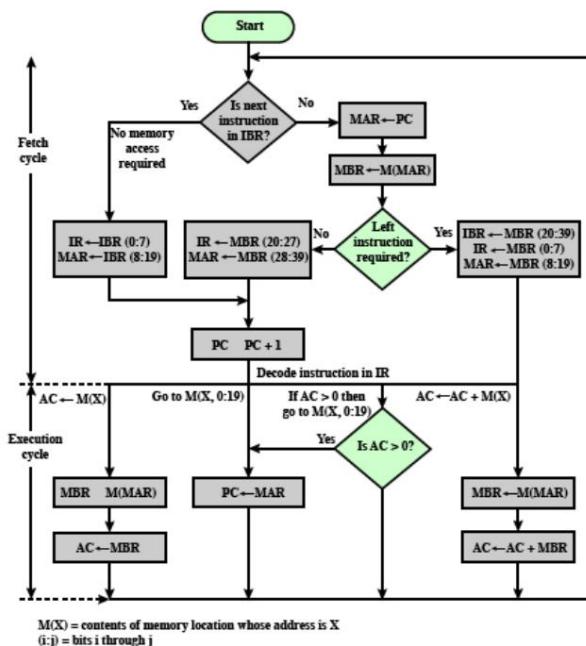


Figure 2.4 Partial Flowchart of IAS Operation

### 37. Irudia IAS Eragiketa

- IAS makinaren funtzionamendua:

Irakaskuntza-zikloak bi FASE ditu

Lehenengo fasea argibide guztientzat komuna da.

- Argibideen adibideak

ÿ X: eragigaien erreferentzia

ÿ AC ÿ M(X)

ÿ GOTO M(X,0:19): Baldintzarik gabeko saltoa X helbidera. X-k bi argibide seinalatzen ditu. X,0:19 da Ezkerreko Instrukzioaren erreferentzia.

ÿ AC>0 M(X,0:19) joan bada: baldintzazko jauzia

ÿ AC ÿ AC+M(x).

#### 7.5.3. Adibidea: Von-Neumann IAS makina

- 2. gaia: Von-Neumann Arkitektura
- IAS ordenagailuaren instrukzio bakoitza fase-segida bat jarraituz exekutatzen da. Sekuentzia hau instrukzio bakoitzerako errepikatzen da eta prozesatzeko unitate zentrala (CPU) instrukzio-ziklo bezala ezagutzen da.
- Kontrol-unitatea instrukzio-zikloaren fase bakoitza implematatzen duen PUZaren unitatea da.
- Kontrol-unitateak CPUaren datu-bidea kontrolatzen du mikrokomandoak erabiliz.
- Barruan mikroordenak sortzeko zirkuitua eta erregistroak osatzen dute: programa-kontagailua eta instrukzio-erregistroa.

#### 7.5.4. Instrukzio-zikloaren ezarpena: CPU

Nola ezarri irakaskuntza-zikloa?

- Zirkuitu Elektroniko Digital sekuentzial baten bidez: FSM egoera finituko makina implematatzen duena egoera diagramaren sekuentzia eta Kontrol Unitatea deritzo.
- Kontrol-unitatea kontrol-zikloaren fase desberdinak burutzen dituen egoera-segida bat da. instrukzioa.
- Instrukzio-zikloaren fase ezberdinek unitate funtzional desberdinak erabiltzen dituzte, hala nola: erregistroak, ALU, etab
- Makina-argibide ezberdinen interpretazioak makinan egoera-segida desberdinak sortuko ditu. Kontrol-unitatea.

CPU egitura

- Oinarrizko hiru baliabide: Kontrol Unitatea, Exekuzio Unitatea eta Erregistroak.
- CPUaren oinarrizko bi bloke
  - ÿ Kontrol Unitatea (UC) eta Datuen Bidea (DataPath).
- Kontrol-unitateaz osatuta dago
  - ÿ instrukzio-zikloa ezartzen duten mikroeragiketen sortzailea
    - ÿ erregistroak: IR instrukzioen erregistroa, ordenagailuko programaren kontagailuaren erregistroa
- Datuen Bidea osatzen dute
  - ÿ EBko Exekuzio Unitatea:
    - ÿ ALU Unitate Logiko Aritmetikoa: zenbaki osoen kalkuluak
    - ÿ Floating Point Unit FPU: zenbaki errealen kalkuluak
  - ÿ Load/Store Unit LSU: helbideen kalkulu eraginkorrik eta memoria nagusirako sarbidea
    - ÿ Memoria Kudeatzeko Unitatea (MMU): MP-aren helbide FISIKO eraginkorra kalkulatzea. CPUak erabiltzen dituen memoria birtualeko helbideak memoria nagusiaren helbide fisikoetara itzultzen ditu.

## ŷ Erregistroak

- ŷ Helburu orokorreko GPR erregistroak programatzaleak eskura ditzake
- ŷ SR egoera-erregistroak
- ŷ Erregistroak Helburua (rax, ~~de~~<sup>mnx, sse, xmm,..</sup>, (erabiltzailea, ~~Orokorra~~<sup>Supererabiltzailea, orria, etenaldia,..</sup>) kontrola eta egoera (rflags, ..)).
- ŷ Kontrol-erregistroak ez ditu erabiltzaileak eskuragarri, sistemak eskura ditzake operatiboa.

- Barne memoria

- ŷ L0 Cache Memoria
- ŷ Memoria Kudeatzeko Unitatea (MMU)

- Sinkronismoa

- ŷ Zereginak sinkronizatzeko erlojua: hardwarearen diseinua errazten du.

Intel

- Intelek PUZa osatzen duten Unitate Elektroniko edo Zirkuitu desberdinak izendatzen ditu haien inplikazioaren arabera irakaskuntza-zikloaren fase bakotzean izen hauekin:

- ŷ Helbide Unitatea (AU): instrukzioaren edo datuen helbide eraginkorra lortzeaz arduratzen da
- ŷ Bus Unitatea (BU): memoria nagusia duten komunikazio-busetara sartzeaz arduratzen da
- ŷ Instrukzio Unitatea (IU): exekutatu beharreko instrukzioa interpretatzeaz eta kudeatzeaz arduratzen da
- ŷ Execution Unit (EB): martxan dauden instrukzioa edo argibideak prozesatzeaz arduratzen da.
- ŷ Kontrol Unitateari dagokio UA-UB-IU-EU unitateak erabiltzeko eta sinkronizatzeko ardura. irakaskuntza-zikloak zuzen.

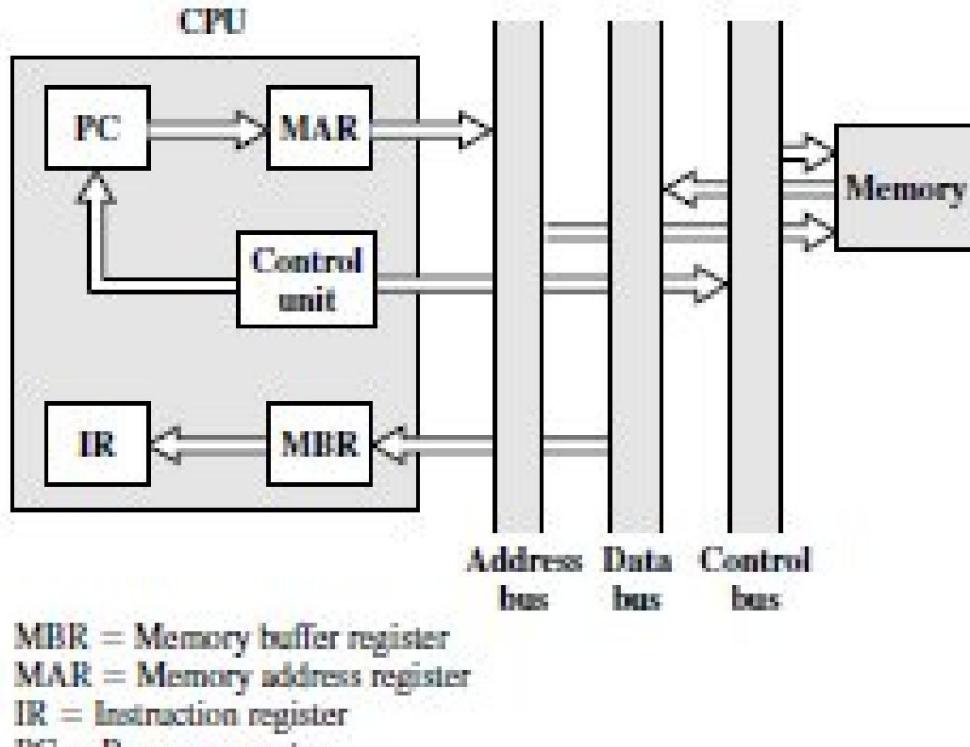
Kontrol-unitatea

- Kontrol-unitatea (batuetan lortu/deskodetzeko unitatea deitzen zaio) banakako instrukzioak memorian dauden tokitik berreskuratzeaz arduratzen da, gero CPUak uler ditzakeen komandoetan itzultzea. Komando hauak makina-hizkuntzako instrukzio gisa deitzen dira normalean, baina batuetan mikro-eragiketak edo UOP deitzen dira. Itzulpena amaitutakoan, kontrol-unitateak exekuzio-unitatera bidaltzen ditu UOPak prozesatzeko.

- UC kontrol-seinaleak

- ŷ Seinale digital bitarrak
- Adibidea: Instrukzio-zikloko Harrapaketa Faseko mikroeragiketak.

- ŷ Instrukzio bat irakurtzen da aktibatzen diren ekintza hauen bidez
- Kontrol unitatea:
  - ŷ Programa-kontagailuak (PC) edo Instrukzio-erakusleak (IP) erreferentzia-helbidea dauka harrapatzeko instrukzioa
  - ŷ Memoriaren Helbideen Erregistroa (MAR) kargatzen da (PC) edukiarekin.
  - ŷ Sistemaren helbide-busa MARren edukiarekin kargatuta dago
  - ŷ Memoriaren I/O Bufferera zuzendutako helbidearen edukia iraultzen da, handik datu-busera horrela, Memoria Buffer Erregistrora (MBR) transferitzen da



**Figure 12.6 Data Flow, Fetch Cycle**

38. Irudia. Datu-fluxua. Harrapaketa Zikloa

- Adibideko mikroaginduen sekuentzia:
  - to. MAR ѕ helbide-busa
  - b. UC ѕ autobus kontrola
  - c. datu-busa ѕ MBR
  - d. MBR ѕ IR eta UC ѕ PC

eta. exekuzioaren amaieran: PC ѕ MAR

#### Exekuzio Unitatea (EB)

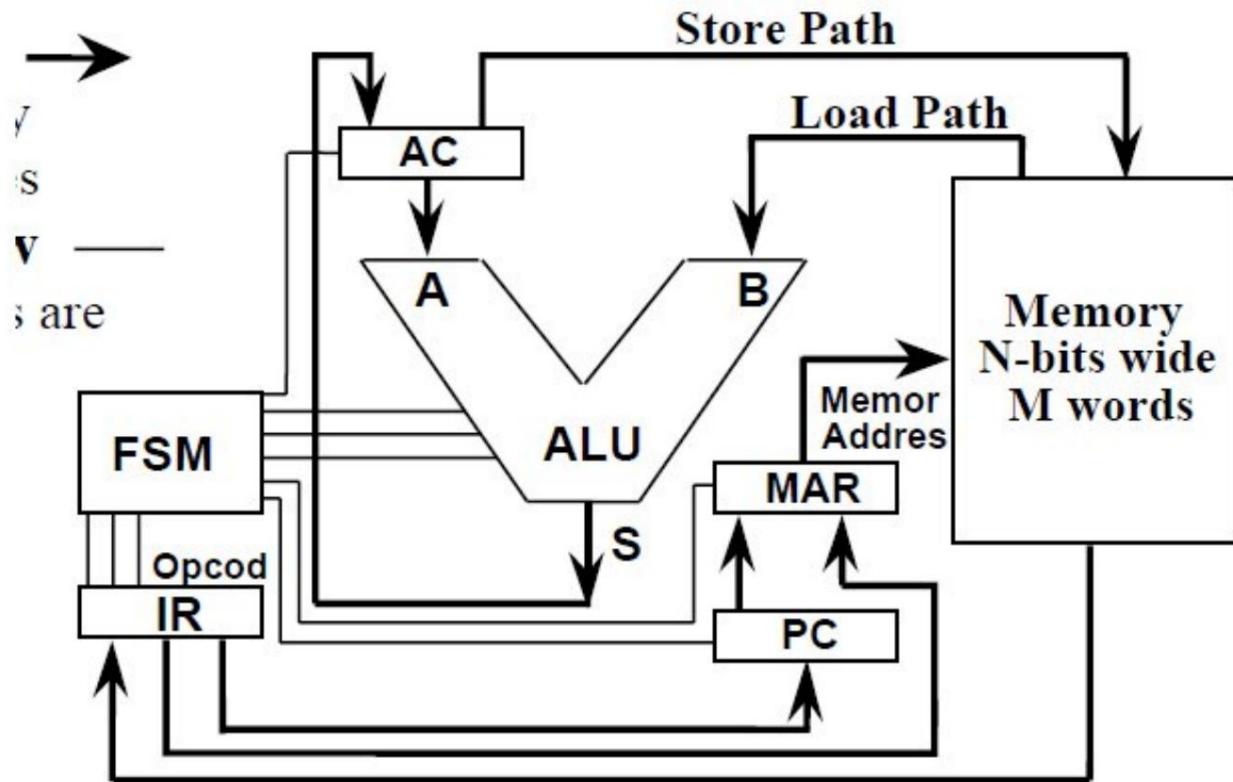
- Exekuzio-unitatea instrukzio-zikloaren hirugarren urratsa egiteaz arduratzen da, hau da, exekuzioaz, edo instrukzioan zehaztutako eragiketa egitea.
- Dakar: ALU+FPU+LSU+RPG  
 ѕ Eragiketak: aritmetika, logika, transferentzia,

#### Datu bidea

- Datuek (argibideak, instrukzio-formatuaren eremuak, eragiketa, helbidea, etab...) prozesadorean zehar, barrutik prozesadorerantz egiten duten ibilbidea da, Kontrol Unitateak zuzenduta.
- Transferitu ahal izateko CPUaren unitate eta azpiunitate desberdinak elkarri konektatzea beharrezkoa da eta haien arteko bit eta bit multzoak prozesatu.
- Seinalean UC mikroaginduek datu horiek garraiatu eta prozesatzen dituzte.  
 ѕ Mikrokomandoen adibideak: ireki atea, konektatu busa, multiplexatu datuak, etab... mikrokomandoak hardware kontrola  
 ѕ Garraio eta prozesamendu hori exekutatzen ari den instrukzioaren interpretazioaren eta exekutatzen denaren araberakoa izango da

mikroarkitektura diseinua.

- Datu Bidearen oinarrizko osagaiak hauek dira:
  - ÿ Garraio-unitateak: BUS, switch, multiplexer, etab.
  - ÿ Memoria unitatea: helbide eraginkorren kalkulua, kanpoko memoriarekin interfazea
  - ÿ Prozesatzeko unitateak: ALU
  - ÿ Biltegiratze-unitateak: erregistroak
- RTL: Erregistratu transferentzia hizkuntza
  - ÿ Garraio, izapide eta biltegiratze ekintzak adierazteko hizkuntza.
  - ÿ AC ÿ [PC]+ M[CS:SP]
- Datu-bideen eskema



39. Irudia Datapath

- ÿ Lerro lodiak: datu-busa
- ÿ Lerro finak: kontrol-busa ÿ txip hautatzea, mikroordena gehitzea, karga-erregistroa, etab.
- Ikusi datu-bideen miniatplikazioa Irudiak atalean
- Datapath diseinua
  - ÿ zehaztu zein mikrounitate diren beharrezkoak
  - ÿ nola konektatu
  - ÿ Zer mikroseinale aktibatu eta noiz mikroeragiketa bakoitzean. Paralelismoa mailan mikro eragiketak
  - ÿ datuen kokapena eta denboralizazioa UC egoera diagramaren sekuentziaren arabera
  - ÿ AC ÿ [PC]+ M[CS:SP] ÿ lotutako mikro-eragiketak eta denbora-diagrama

### 7.5.5. Mikroprogramatutako Kontrol Unitatea

- Kontrol-unitatea mikroprogramatua vs
  - Mikrokablea: kontrol-unitatearen sekuentiazioa edo FSM-ak zuzenean exekutatzen ditu argibideak memoria nagusian gordetako makina-kodea
  - Mikroprogramatua:
    - ÿ Memoria nagusian gordetako makina-argibideak (ISA) eta erabiltzailearen programaren makina-kodea osatzen duten sekuentzia ez ditu UC-k zuzenean exekutatzen. Horren ordez, makina-kodeko instrukzio bakoitza mikroargibideen sekuentzia batean itzultzen da eta mikroagindu bakoitzak instrukzio-zikloa osatzen duten kontrol-unitateko mikroeragiketak edo mikroseinaleak sortzen ditu.
    - ÿ Mikro instrukzio bati lotutako mikroaginduen sekuentziak Kontrol Unitatearen barnean irakurtzeko soilik den memoria batean (Read Only Memory ROM) gordetzen duen mikrokodea osatzen du.
    - ÿ Gure Kontrol Unitateari mikrokodea aldatuz edo gehituz arkitektura berriak lortzen ditugu ISA modu malguagoan kabledun kontrol-unitatearekin baino.
    - ÿ mikrokodea
- ÿ CPU izena oraindik erabiltzen bada ere, gaur egun lehen CPUen funtzioa nukleoek betetzen dute. Gaur egungo CPUek, prozesadore edo nukleo zentralaz gain, beste funtzio mota batzuk ere integratzen dituzte, hala nola cache memoria, I/O kontrolagailuak, etab., beraz, benetan Sistema Integratuak deitu behar zaie (sistema txiparen gainean, etab.).

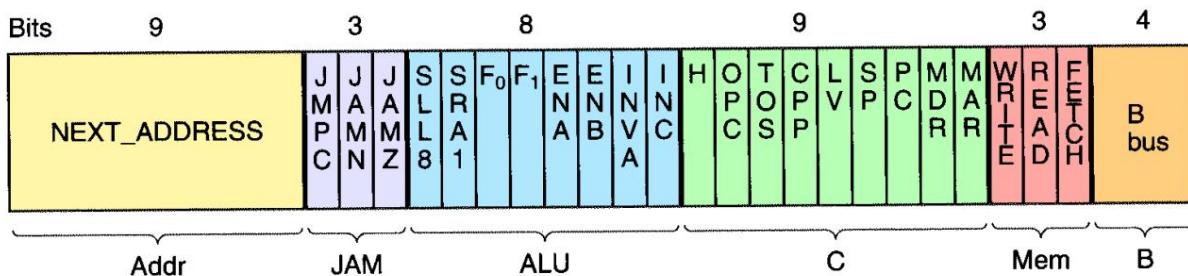
- Mikroprogramatutako Kontrol Unitate bi mota daude: Horizontala eta Bertikala
- Kontrol-unitateko "mikro instrukzio horizontal" baten formatua

ÿ Kontrol mikroseinale adina bit ditu. 256 mikroseinale baino gehiago.

ÿ Mikro instrukzioko bit bakoitzak kontrol-seinale bat aktibatzen du:

ÿ INC bit-ak seinalearen gehikuntza ALUren 1ean aktibatzen du

ÿ READ bit-ak DRAM memoria nagusiaren READ seinalea aktibatzen du



40. irudia. horizontal\_microinstruction

- "Mikro instrukzio bertikal" baten formatua

ÿ Mikroseinale guztiak ez dira aldi berean aktibatzen, beraz, konbinazio posibileen kopurua  $2^{N\_of\_microsignal}$  baino askoz txikiagoa da, adibidez, mikroseinale kopurua 256 bada, konbinazio posibileen kopurua 2256 baino askoz txikiagoa da.

ÿ Konbinazio posibileen kopurua  $64=26$  balitz , orduan, kasu guztiak 256 biten ordez 6 bitekin kodetu litezke, beraz, mikro instrukzioaren luzera 256tik 6ra murritzen da. Eragozpena da mikroinstrukzio bertikal bakoitza deskodetu behar dela.

## 7.6. PUZaren adibideak: IAS eta Intel 4004

### 7.6.1. IAS

- [https://es.wikipedia.org/wiki/IAS\\_machine](https://es.wikipedia.org/wiki/IAS_machine)
- Garapen Programatzileen Konputagailuen Arkitekturaren ikuspegia:

ISA:

- argibideak: operazio-kodeak eta helbideratze-moduak - datu-formatua - instrukzio-formatua
- memoriaren antolaketa - erregistroak

- Mikroarkitektura: HW (elektronikarik gabe, unitate funtzionalak) Konputagailuen arkitekturaren ikuspegia

Irakaskuntza Zikloa

FI-ID-EI

Unitateak

CU

uoperations bus udata

bus

AU

PCa

BU

ITSASOA

MBR

UI

IR, IBR

EB

ALU

A.C., Q.C.

- Plataformaren arkitektura

CPU

MEMORIA memoria BUS: helbidea, datuak, kontrola (R/W)

### 7.6.2. Intel 4004

sarrera

- Intel: Intel 1968ko uztailaren 18an sortu zen Integrated Electronics Corporation gisa
- Lehenengo i4004 mikroprozesadore integratua "4001, 4002, 4003, & 4004 Micro Computer Set" MCS-4 chipset familiakoa zen.
- [https://es.wikipedia.org/wiki/Intel\\_4004](https://es.wikipedia.org/wiki/Intel_4004)

- Denetarikoa

Eguna: 1971ko abendua

Kostua: 60\$

Teknologia: 10 um nodoa

4 biteko arkitektura

## DA

- Garapen Programatzaleen Konputagailuen Arkitekturaren ikuspegia:

ÿ ISA:

- argibideak: operazio-kodeak eta helbideratze moduak
  - Jarraibide multzoa 46 instrukzioz osatuta dago
  - Helbideratzeko moduak: zuzena, zeharkakoa, berehalakoa.
- Datuen formatua
  - 4 bit zabaleko datuak
  - instrukzio formatua
  - 41 instrukzioek 8 bit zabal dute eta 5ek 16 bit.
- Memoriaren antolaketa
  - memoria ezberdinetan banandutako datuak eta argibideak -> Arkitektura Harvard, ez Neumann.
  - erregistroak
    - Helburu orokorreko 16 erregistro
    - 12 biteko 3 erregistro pilaren fotograma desberdinak apuntatzeko

ÿ tresnak

muntaia hizkuntza  
muntatzaile itzultzeko

## Mikroarkitektura

- Mikroarkitektura: HW (elektronikarik gabe, unitate funtzionalak) Konputagailuen arkitekturaren ikuspegia

Arkitektura Tamaina: 4 bit

Irakaskuntza Zikloa

FI-ID-EI

Unitateak

C.U.

uoperazioen autobusa

udata bus -> 4 bit

AU

PC -> 12 bit

pilatzeko erakusleak -> 12 bit

BU

memoria Helbide\_bus/Datu\_bus ->

KONTUZ: helbide eta datuetarako bus bakarra -> bus partekatua,

beraz, denboran ziklo desberdinak beharrezkoak dira. 12 biteko helbideak eta  
4 biteko helbide-busa: (3 zikloko 4 biteko denborazko multiplexazioa helbidea osatzeko)

MAR/MBR : funtzio bikoitza duen erregistro bakarra -> 4 bit 4 memoria banku: Memoria  
Bankua -> kontrol-unitateko memoria kontrolatzeko busa (chip\_select, R/W)

UI

IR, IBR

EB

ALU

A.C., Q.C.

Erregistroak

RPG (Ondorio Orokorreko Erregistroak)  
pilatu

Erlojuaren maitzasuna: 740KHz

MSC-4 chipset

## Memoria

- (Memoria unitateari buruzko hurrengo gaian ikusiko da)
- Memoriaren antolaketa
  - ÿ Datu/helbide-bus partekatua: 4 biteko datu eta helbide-bus bakarra.
  - ÿ Helbideen multiplexazioa ÿ 12 biteko helbidea=4bit/ziklo x 3ziklo, hau da, kontrol-unitateak 3 erloju-ziklo  
behar ditu 12 helbide-bitak osatzeko.
  - ÿ ROM-ko programa-memoriak eta RAM-eko programa-memoriak 212 byte-ko edukiera dute ,  
4KByte
  - ÿ Bestalde, RAM-eko datu-memoriak beste helbide mota bat du eta 5120 biteko ahalmena du.
- **Chipset:** txip konstelazioa.

i4004 mikroprozesadore txipa, instrukzio-sekuentziatzalea, instrukzio-deskodetzailea, azpierrutinen  
pila, ALU eta 4 biteko hamasei datu-erregistro integratzen dituena. Hau izan zen historiako txip bakarreko lehen  
mikroprozesadorea. i4001 ROM txipak 256x8 biteko maskara  
programagarria den ROM bat eta software kontrolagarriak diren lau sarrera/irteera ataka konbinatzen ditu.  
i4002 RAM txipa 320 biteko RAM (4 biteko  
20 hitzeko 4 erregistro) eta softwarez kontrola daitezkeen lau irteera-porturekin: Datuetarako bus partekatua eta  
i4003 irteera-hedagailura zuzentzen da,  
funtsean, 10 biteko serieko paralelo-irteera  
desplazamendua. erregistratu.

- ÿ Gutxieneko MCS4 sistema bat i4004 CPU eta i4001 ROM txip batez osatuta dago (eta kanpoko osagai  
batzuk bi faseko erloju-sorgailua bezalakoak), sistema handiena, berriz, hamasei ROM txip (4 KByte)  
eta hamasei 4002 RAM txip (640 byte). .
- ÿ Egungo arkitekturak ez bezala, txipsetak memoria barne hartzen duela ikusten da eta hori CPUak  
berak kontrolatzen du memoria kudeatzeko jarraibideen bidez, hala nola DCL eta SRC jarraibideen  
bidez, lehen memoria bankua eta ondoren datuen helbidea hautatzeko erabili ziren jarraibideen  
bidez. datu horiekin eragiketak egiteko.
- ÿ RAM memoriaren antolaketa bankuetan/txipetan/erregistroan/hitzetan honako hau da (an ikusiko da

memoria unitateari buruzko gai hau:

- ÿ Memoria Bankuen helburua memoria-ahalmena handitzea da  
Programa-kontagailuaren eta Helbide-Busaren muga.

Gehienez 4 4002 RAM txip konekta daitezke banku bakoitzean (4004 CPUREn CM linea). Sistema 4 edo 8 banku izateko konfigura daiteke.

Beraz, sistemak onartzen duen gehienezko txip kopurua 32 da.  
RAM memoriatik instrukzio edo datu bat hautatzen denean, bankuetako bat hautatzen da, hautatutako bankuarentzat txipetako bat eta hautatutako txiparentzat txiparen 4 erregistroetako baten hitza (instrukzioa edo datuak).

## 7.7. Ordenagailua programatzailearen ikuspuntutik (II)

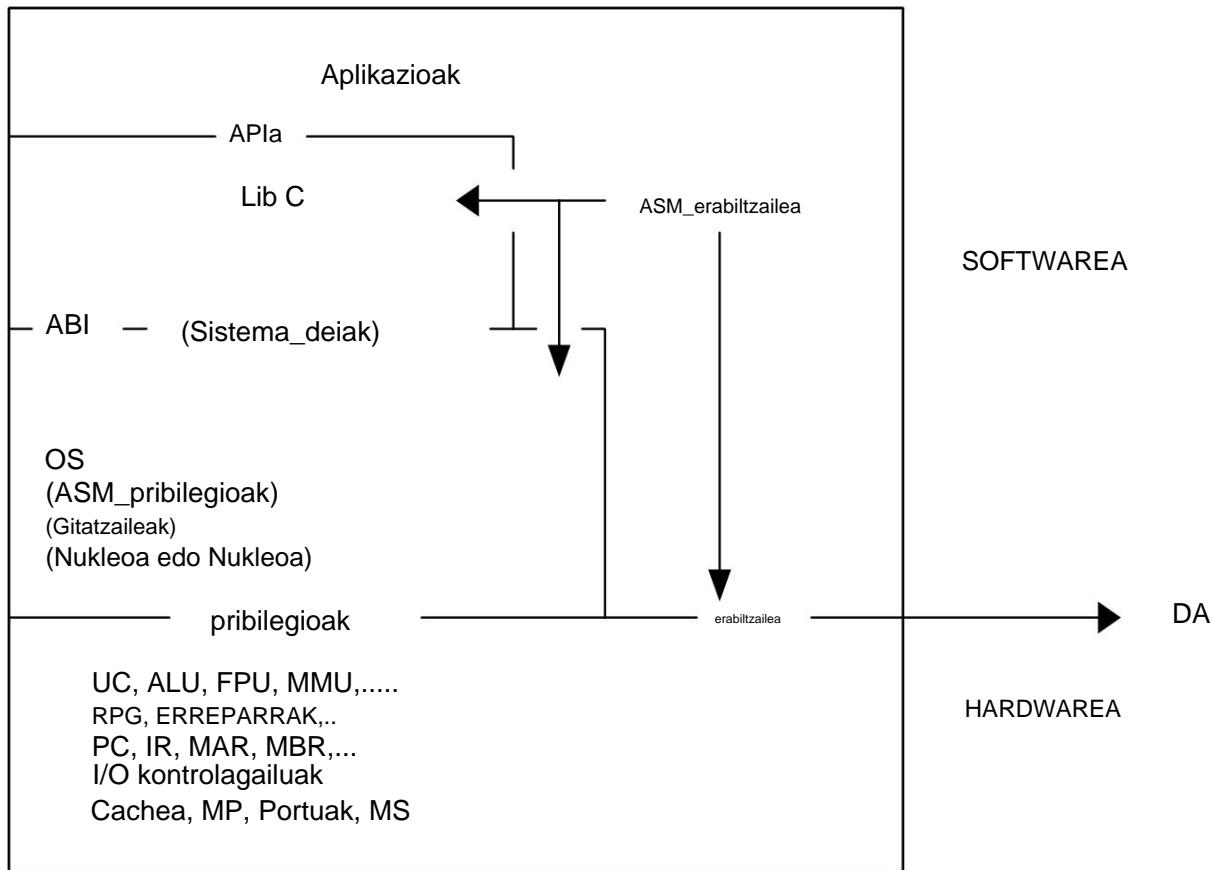
### 7.7.1. Abstrakzio mailak edo geruzak

- Sistemaren programatzailea Hardwarearen implementaziotik abstraitzen da (neurri batean) Sistema Eragilearen Kernelari esker. Programatzaileak Sistema Eragilearekin elkarreagin egiten du ordenagailuaren HW baliabideetara sartzeko.
- Aplikazioak garatzen dituen programatzaileak, hala nola, konpiladoreak, estekatzaileak, nukleo-baliabideak, hala nola kontroladore-kontrolatzailak, nukleorako sarbide-kudeatzaileak, adibidez, CPU eta memoria-konfigurazioa, etab. Hardwarea maila funtzional batean ezagutu behar du eta ez fisiko edo elektronikoan. maila eta hori software mailan dagoen geruza baxuenaren bidez lortzen da, hau da, ordenagailuaren geruza fisiko edo elektronikoa abstraitzea lortzen duen makina-lengoaia.
- Machine Abstrakzioa: ISA argibideak / ABI zehaztapenak erabiliz

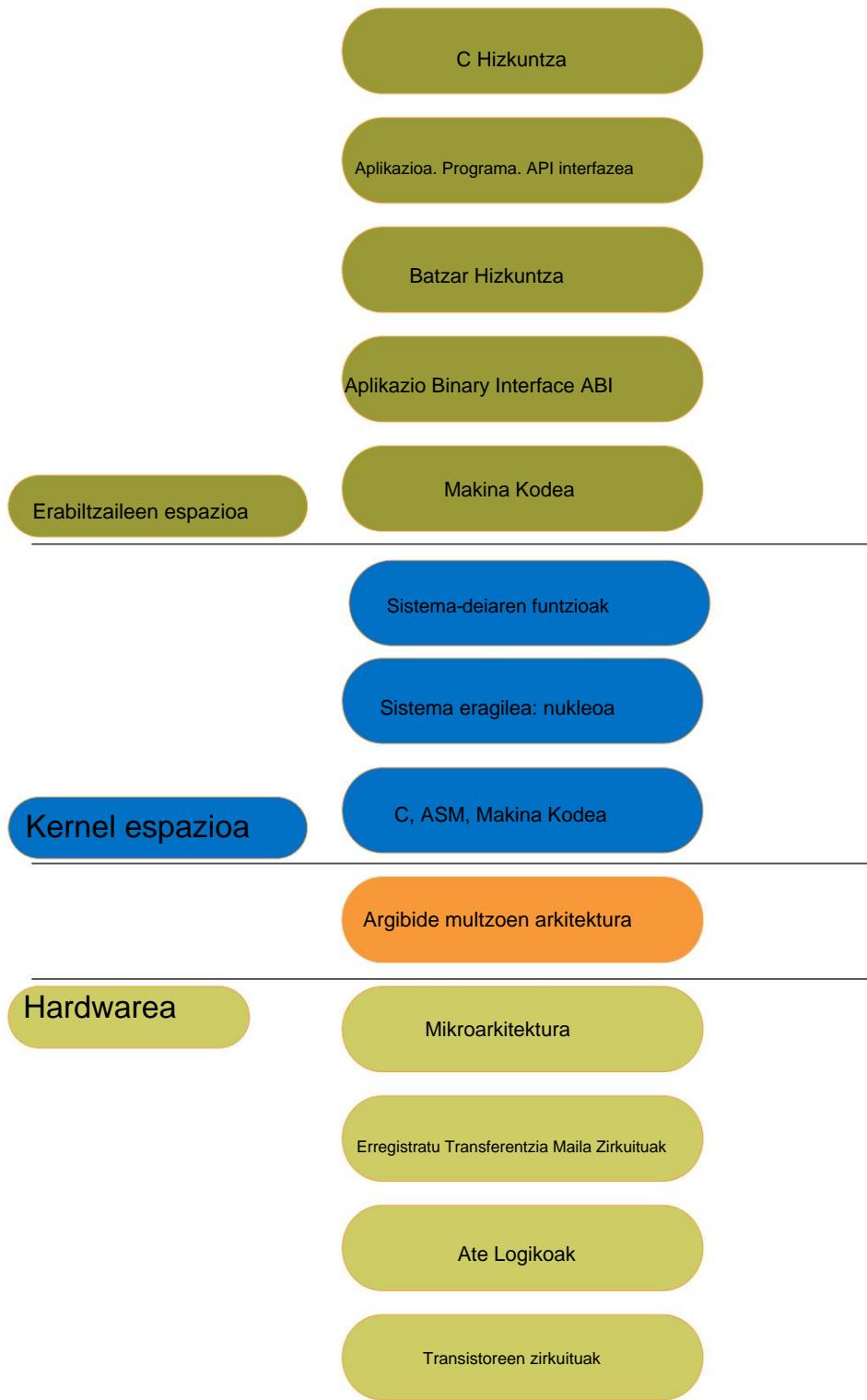
ÿ **ABI** : "Aplikazioaren Interfaze Bitarra". Softwarearen ezaugarri bitarrak zehazten dituen dokumentua da, hau da, softwarearen maila baxuena. Adibidez, azpirrutinen dei-kontenzioa, pila nola egituratzen den, sistema-deiak nola egin, objektu exekutagarriaren moduluaren formatu bitarra, etab... Konpilatzailak, estekatzaileak eta kargatzailak ABI interfazea zehatz-mehatz ezagutu behar du.

ÿ Behe-mailako aplikazio-programatzailearen ikuspuntutik: Makinareniko interfazea da sistema-deiak (ABI) eta "erabiltzaile ISA" erreperiorioa

- Abstrakzioa goragoko mailetan
  - ÿ Liburutegiarekiko interfazea liburutegiko funtzioen prototipoak dira (Aplikazioen Programazioa Interfazea - APIa)
- **Maila baxuko lengoietan** garatutako aplikazioen interfazeen diagrama :



- OS sistema eragilearen ikuspuntutik:
  - ÿ Makinarekiko interfazea ISA da (sistema isa eta erabiltzailea isa)
  - ÿ Programatzailearekiko interfazea ABI da
- Programatzailearen ikuspuntutik
  - ÿ OSrik ez badago, makinaren interfazea OSaren parekoa izango da
  - ÿ OS eta liburutegiak badaude, makinarekiko interfazea:
    - ÿ C hizkuntzan: C-eko API eta ABI espezifikoak
    - ÿ ASM hizkuntzan: API © eta asm-eko ABI espezifikoak
    - ÿ Behe-mailako programazioak makinaren hardwarearen nolabaiteko ezagutza izatea eskatzen du, eta ezinezkoa da haren abstrakzio osoa. Beraz, CPU programatzaileen ikuspuntutik aztertzea beharrezko da.



#### 41. Irudia Abstrakzio-geruzak

- Zein izango litzateke hurrengo abstrakzio-maila altuagoek ikusten duten mailen edo geruzen eskema?
  - ÿ Mahaigaina
  - ÿ Java Programazio Lengoiaia

### 7.7.2. Softwarearen bateragarritasuna

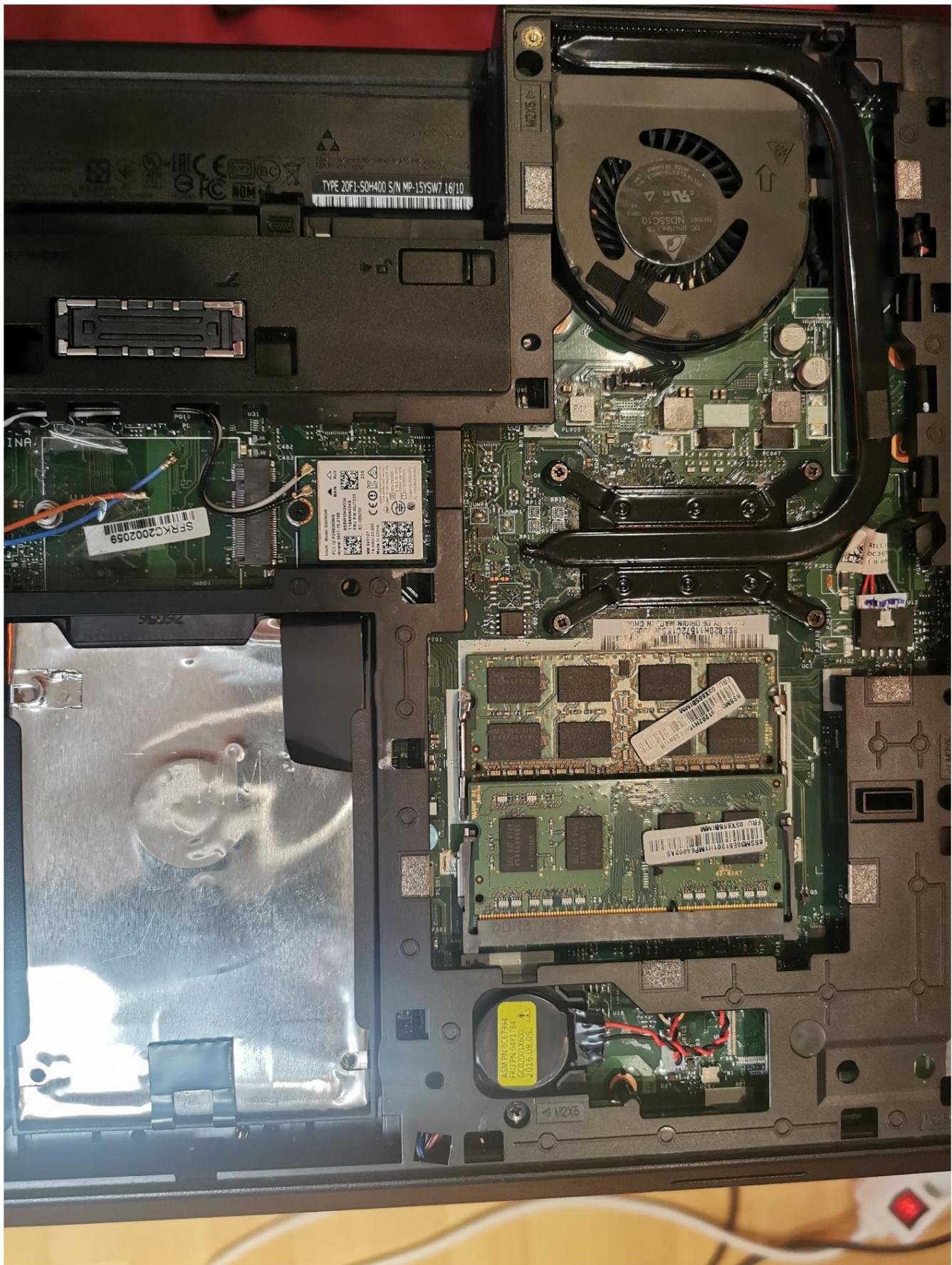
#### Bateragarritasuna

- Prozesadore bakoitzak bere argibideen erreperitorioa du
- Bi prozesadorek instrukzio-multzo bera badute, hau da, arkitektura bera badute, mihiztadura-lengoaiaren iturburu-modulua bateragarria izango da bi prozesadoreentzat nahiz eta PUZaren barne-egitura desberdina izan: Adibidea: Intel IA64 eta AMD64

#### Adibideak

- Programatzaleak ARCH-KERNEL-LIBC hirukotea ezagutu behar du
  - ÿ Arch ordenagailuaren arkitekturari egiten dio erreferentzia ÿ ISA
  - ÿ Kernel: sistema eragilearen muina. Sistema-deiak ezartzea
  - ÿ Libc: aplikazioaren programatzaleentzako liburutegia. Sistema-deiak ezartzea
  - ÿ Bai Kernelak bai Liburutegiak maila altuko interfazeak (API) eta maila baxuko interfazeak dituzte. (ABI)
- Adibideak arch/kernel/libc
  - ÿ amd64-linux-gnu
  - ÿ arm-linux-gnueabi

### 7.8. Ordenagailu pertsonala: T560 ordenagailu eramangarria



42. Iruzia. Laptop\_T560: Barneko ikuspegia



43. irudia. hardisk\_platform\_off

#### 7.8.1. linux komandoak

- HW-ý Kernel-ý Fitxategi sistema (/proc, /sys /dev) -ý linux komandoa -ý erabiltzailearen sistemaren informazioa

• zerrenda

```
sudo lshw --help man lshw
```

```
sudo hwinfo --
```

```
help man hwinfo
```

```
sudo dmidecode --help man
```

```
hwinfo sudo inxi
```

#### 7.8.2. aplikazioak

- cpu-x

#### 7.8.3. Webgunearen informazioa

- wikipedia

- adimena

• <https://ark.intel.com/content/www/us/en/ark.html>

յ <https://www.intel.com/content/www/us/en/developer/articles/technical/intel-sdm.html>

- <https://www.cpu-world.com/>
- <https://agner.org/>

#### 7.8.4. sistema eragilea

- batu nirekin -a

```
Ubuntu/GNU/linux 20.04 kernel  
linux 5.4.0-131-generic
```

#### 7.8.5. Plataforma

##### aginduak

- zerrenda

```
sudo lshw -X  
sudo lshw | gehiago  
sudo lshw > myplataforma.txt sudo  
lshw -html sudo lshw  
-short sudo dmidecode  
-system  
batu nirekin -a  
arku  
sudo cpu-x  
inxi  
inxi -c 0 -ACdGMNSz
```

##### Lenovo Thinkpad T560 txostena

- oinarrizko ezaugarrien zerrenda

```
ordenagailu-eredua: 20F1S0H400 (LENOVO_MT_20F1_BU_Think_FM_ThinkPad L560) PUZaren  
eredua: Intel(R) Core(TM) i5-6300U CPU @ 2.40GHz PUZaren arkitektura:  
x86_64 -> arku komandoa Hitzaren zabalera: 64
```

Memoria Edukiera: 12GiB Sistemaren Memoria -> 4+8 Chipset edo

PCH: Sunrise Point-LP USB 3.0 xHCI

Kontrolatzailea SATA Kontrolatzailea

[AHCI modua]

PCI Express Root Port LPC

kontrolagailua HD

Audio

grafikoak: Skylake GT2 [HD Graphics 520]

Sarea:

Haririk gabeko 8260

Ethernet konexioa I219-LM

Erlojuaren maiztasuna: 2,40 GHz

Cachea

64KiB L1 cachea

64KiB L1 cachea

512 KiB L2 cachea

3 MiB L3 cachea

#### 7.8.6. CPU



44. irudia. cpu\_package

# MOST SCALABLE EVER

## FROM 4.5W TO 91W: DELIVERING INNOVATION FOR EACH SEGMENTS

	2 in 1 Detachables, Tablets and Compute Stick	Thin Light Notebooks, Portable AIO, Minis and Conference Room	Ultimate Mobile Performance, Mobile Workstations	Desktop Performance to Value, AIO and Minis
	Y-SERIES	U-SERIES	H-SERIES	S-SERIES
<b>5 Dies 4 Packages</b>				
<b>Dies</b>	2+2	2+2	2+3e	4+2
<b>Package (mm)</b>	BGA 1515	BGA 1356	BGA 1440	LGA 1151
	20 x 16.5	42 x 24	42 x 28	37.5 x 37.5
<b>TDP (W)</b>	4.5	15	15, 28	45
<b>Chipset</b>	Integrated 6 <sup>th</sup> Gen Intel® Core™ Platform I/O			Intel® 100 Series chipset (23mm x 23mm)

Intel Confidential – UNDER EMBARGO UNTIL September 1, 2015 9:00PM PDT  
All products, dates, and figures specified are preliminary based on current expectations, and are subject to change without notice.



45. irudia skylake-family-table

aginduak

• zerrenda

```
sudo lshw -C cpu lscpu
sudo
dmidecode -t prozesadorea cpuid cat /proc/
cpuinfo
sudo cpu-x
```

web informazioa

- cpu-mundua
- Intel: 6. belaunaldiko core i5 prozesadoreak
  - ↳ Intel: Core i5-6300U

txostena

- oinarrizko ezaugarrrien zerrenda

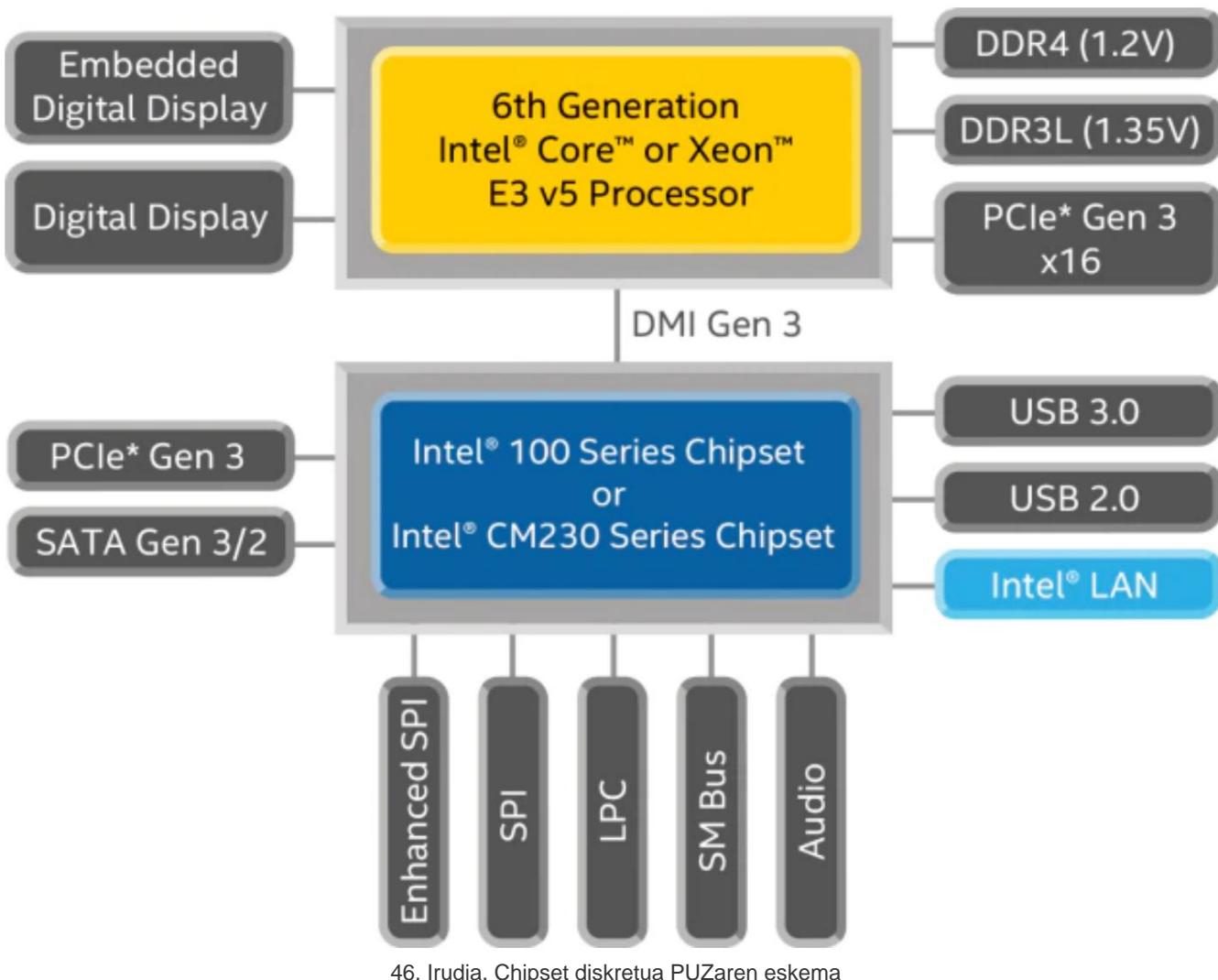
PUZaren eredua: Intel(R) Core(TM) i5-6300U CPU @ 2.40GHz Arkitektura:

x86_64 Socket(k): Byteen	
ordena:	1 -> Soldadu edo
Helbideen	Endian txikia
tamaina: Nukleo	39 biteko fisikoa, 48 biteko birtuala
kopurua: 2 Hari kopurua: 2	
nukleo bakoitzeko Maiztasuna:	

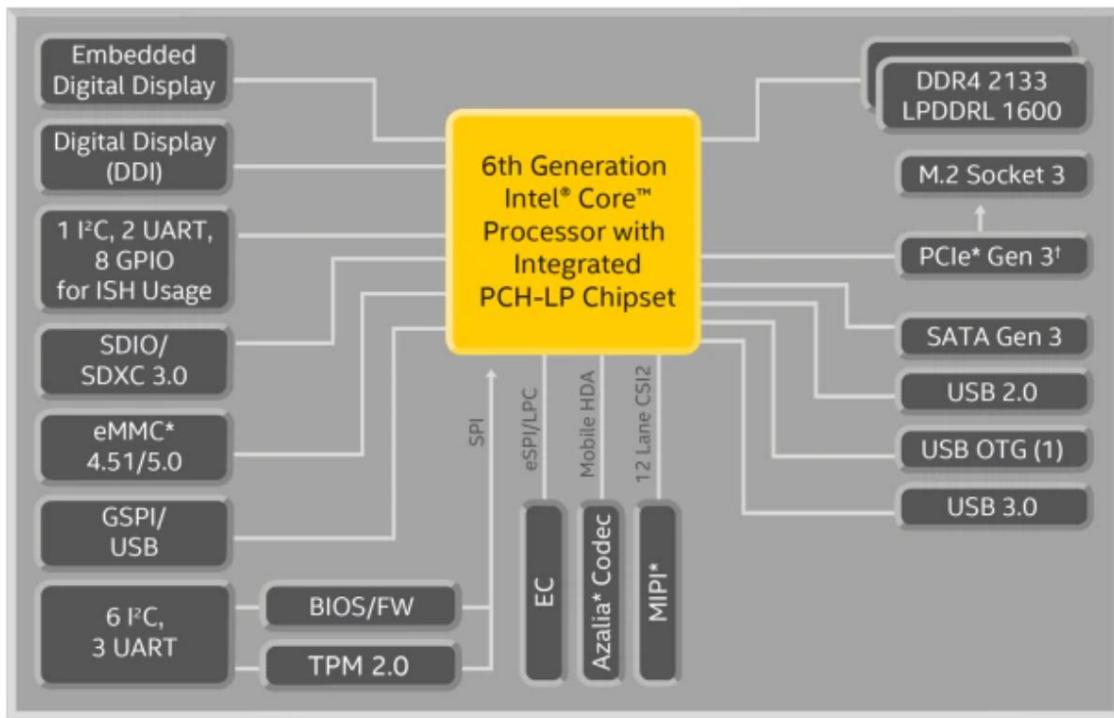
erlojua: 100MHz -> Plaka amaren maiztasuna 2.4GHz ->  
marketina 600 MHz ->  
komandoa exekutatzeko unean cpu frekuentzia Biderkatzzailea: gehienezko balioa 24  
-> freq\_cpu=24xFreq\_motherboard -> biderkatzzailea dinamikoa da, etengabe aldatzen  
da.  
Mikroarkitektura: Skylake 14 nm -> Skylake-U  
Paketea: BGA  
Birtualizazioa: VT-x -> CPU-k Intel birtualizazioa onartzen du Banderak:  
Im:  
modu luzea -> PUZaren funtzionamendua 64 biteko moduan x86\_64:  
arkitektura -> hitzaren zabalera SSE, AVX -  
> Oinarrizko ISA luzapena vmx argibide gehiagorekin: BIOS CPUDek  
birtualizazioa gaituta 406E3 -> PUZaren  
identifikatzialeen kodea -> beharrezkoa da mikrokodea egunearatzeko Gehienezko  
funtzionamendu-tenperatura 100°C Diseinu  
Termikoko Potentzia 15 Watt \_> kontsumoa

- Intel Core™ i5-6300U ѕ merkataritza-kodearen esanahia
  - ÿ Googlen bilatzeko "i5-6300U" izen komertziala erabiltzen dut
  - ÿ Korporazio artekoa
  - ÿ marka: muina
  - ÿ familia: i5
  - ÿ 6. belaunaldia: 6
    - ÿ bertsioa: 6000
    - ÿ koaderno-segmentua (koadernoak): U
- Data: 2015 3. seihilekoa.
- PUZaren maiztasuna instrukzio-zikloko makina-zikloen erloju-zikloa da.
  - ÿ instrukzio-zikloen iraupena eta haien faseak erloju-zikloetan adierazten dira, ez MHz-tan.
- Mikroarkitekturaren kode-izena nabarmenzeko: **Skylake-U**
  - ÿ CPUREn hardware baliabide guztiei buruzko informazioa ematen digu.

#### integrazio maila: cpu-chipset



46. Irudia. Chipset diskretua PUZaren eskema



47. Irudia. Chipset PUZaren eskema integratua, scaledwidth=

- On txip (moduluen integrazioa txip berean): **moduluen integrazioa txip berean da garrantzitsua**

Nukleoak -> 2 x87 nukleo

FPU txipan -> Float Point Unitatea: zenbaki errealak prozesatzen ditu ez bezala.

Zenbaki osoak prozesatzen dituen ALU

Chipset edo PCH (Platform Hub)

ÿ CPUa eta chipset-a txip berean daude baina "troquel" ezberdinan. Bakotzak barneratuta

SM2102KO substratu ezberdin bat.

## Entxufea

- socketa: <https://globalamericaninc.com/types-of-cpu-sockets/>

ÿ cpu-mundua

ÿ Paketea: 1356-ball micro-FCBGA

ÿ Entxufea BGA1356 ÿ BGA ÿ plakara soldatuta ÿ ezin da berritu

ÿ txiparen 1356 pinak txip berean integratutako modulu guztienak dira: 2.  
nukleoak eta chipset kontrolatzaileak.

ÿ sudo hwinfo | grep -i socket ÿ "U3E1" ÿ Ezin al dut informaziorik aurkitu?

ÿ sudo dmidecode -t prozesadorea | grep -i socket > Socket izendapena: "U3E1"

## Datu okerrak

- sudo lshw -C cpu ÿ tamaina: 975MHz
- sudo dmidecode -t prozesadorea | grep -i freq ÿ gehieneko maiztasuna 8GHz dela esaten dit

## 7.9. Irakaskuntza-mailako parallelismoa (ILP)

- [wikipedia](#)

ÿ Instrukzio-mailako parallelismoa (ILP) programa baten zenbat instrukzio aldi berean exekutatu daitezkeen neurria da.  
Instrukzioen exekuzioan gainjartzeari instrukzio-mailako parallelismoa (ILP) deritzo.

ÿ ILPa lortzeko bi mekanismo daude

ÿ Hardwarea

ÿ Softwarea

- ILP gainjartzea bilatzen duten mikroarkitektura diseinu-teknikak

ÿ VLIW

ÿ Supereskalarra

ÿ Pipelining (segmentazioa)

ÿ Ordenantzaz kanpoko exekuzioa

ÿ etab

### 7.9.1. Pipeline (segmentazioa)

- Hodia: kanala edo hodia. Segmentazio seriea.
  - Autoak garbitzeko adibidea
- ÿ Faseak: Hezeta - Xaboa - Eskuila - Garbitu - Lehortu - Leundi

- Makina sekuentziala

ÿ Makinaren aurrean autoen ilara

ÿ Kotxe bat faseren batean badago, hurrengo kotxea ez da sartzen.

ÿ Kotxea irteteko denbora tartea fase guztien batura izango da. Zenbat aldiz irteten da kotxe batek garbigailutik?

ÿ **Throughput (ekoizpena):** denbora-unitate bakotzeko irteerako auto kopurua

- Segmentazioa vs Sekuentziala.

ÿ Fase guztiak egiten dituen makina bat izan beharrean, makina independenteak ditugu fase bakotza burutu.

ÿ Kotxea irteteko denbora tartea fase luzeenaren iraupena izango da.

ÿ Errendimendua, denbora-unitate bakotzeko zerbitzatutako auto kopurua, handitzen da.

- Instrukzio-fluxua 2 etapatan segmentatzen duena

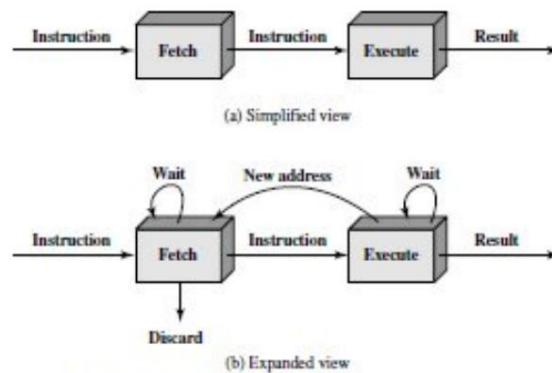


Figure 12.9 Two-Stage Instruction Pipeline

#### 48. Irudia. Segmentazioa 2 etapatan

- Etapa bakotzeko denborak desberdinak badira edo fluxuan jauziengatik zigorra badago, itxaron denborak sortuko ditu.

BERAK

	Time →													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Instruction 1	FI	DI	CO	FO	EI	WO								
Instruction 2		FI	DI	CO	FO	EI	WO							
Instruction 3			FI	DI	CO	FO	EI	WO						
Instruction 4				FI	DI	CO	FO	EI	WO					
Instruction 5					FI	DI	CO	FO	EI	WO				
Instruction 6						FI	DI	CO	FO	EI	WO			
Instruction 7							FI	DI	CO	FO	EI	WO		
Instruction 8								FI	DI	CO	FO	EI	WO	
Instruction 9									FI	DI	CO	FO	EI	WO

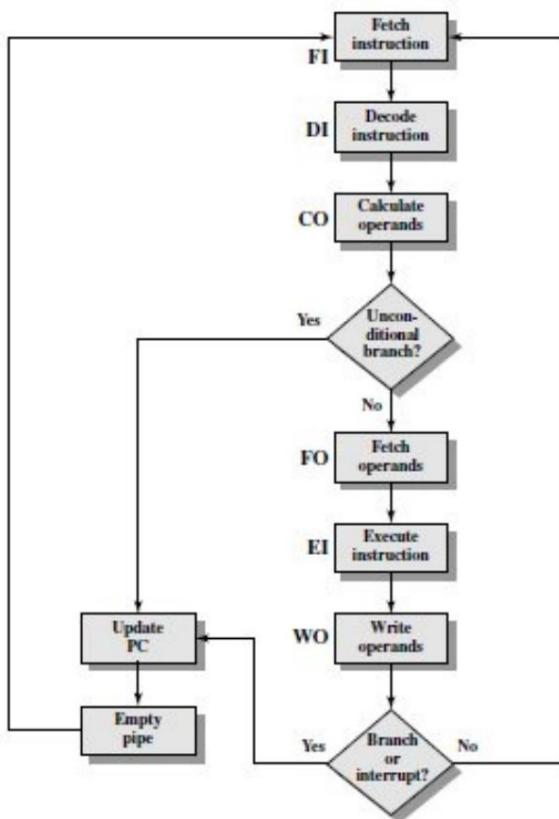
Figure 12.10 Timing Diagram for Instruction Pipeline Operation

49. Irudia. Denbora-diagrama 6 etapako segmentazioa

	Time →													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Instruction 1	FI	DI	CO	FO	EI	WO								
Instruction 2		FI	DI	CO	FO	EI	WO							
Instruction 3			FI	DI	CO	FO	EI	WO						
Instruction 4				FI	DI	CO	FO							
Instruction 5					FI	DI	CO							
Instruction 6						FI	DI							
Instruction 7							FI							
Instruction 15								FI	DI	CO	FO	EI	WO	
Instruction 16									FI	DI	CO	FO	EI	WO

Figure 12.11 The Effect of a Conditional Branch on Instruction Pipeline Operation

50. Irudia. Denbora-diagrama. Baldintzarik gabeko jauzia



51. Irudia 6 etapako kanalizazioa duen instrukzio-fluxua

- Jauzi batek hodia hustera behartzen du ý Kontrol Unitaterako mikroordena berria ý hodia hutsa

### 7.9.2. VLIW vs Superscalar

#### VLIW

- Argibide-hitzak oso luzeak
- CPUak Exekuzio Unitate anitz ditu
- Hitz batek exekuzio-unitate adina instrukzio ditu.

ý Hitzari Instruction Word deitzen zaio, eta horrek hainbat makina jarraibide ditu.

ý Konpilatzaileak Word Argibideak sortzen ditu, horietako bakoitza **esleitzen duten** hainbat argibiderekin beste Exekuzio Unitate bat.

ý Argibide anitz paraleloan

#### Supereskalarra

- Arkitektura supereskalarra esan nahi du CPUak datu-bide anitz dituela (Exekuzio Unitate (UE) eta bideraketa anitzak), ez da nukleo anitzekin nahastu behar, eta CPU bera da exekuzio-denboran UE desberdinak eta beste makina baliabideak esleitzen **dituena** . argibideak.
- Arkitektura honek hainbat instrukzio aldi berean exekutatzeko aukera ematen du.
- **N bideko** CPU supereskalarra esan nahi du aldi berean n instrukzio exekutatu ditzakeela.
- Supereskalarra ez du multicore esan nahi. Nukleo bakarra supereskalarra da.

#### Superskalar-VLIW konparaziao

- Konputagailuen arkitekturako eztabaidea handietako bat estatikoa vs. dinamikoa. **static** esan nahi du normalean "har dezagun gure konpilatzailea arduratzen den hau", eta **dinamikoak** normalean "eraiki dezagun hau zaintzen duen hardwarea" esan nahi du. Alde bakoitzak bere abantailak eta desabantailak ditu. konpiladorearen ikuspegiak denboraren onura du:

konpilatzaileak egun osoa eman dezake kode zati bat aztertzen. Hala ere, konpiladore batek lor ditzakeen ondorioak mugatuak dira, ez baitaki zeintzuk izango diren aldagai guztien balioak programa benetan exekutatzen denean. Imagina dezakezun bezala, hardwarearen ikuspegira joaten bagara, makilaren beste muturra lortuko dugu. hardwarean egin dezakegun analisi kopuruaren muga dago, gure baliabideak askoz mugatuagoak baitira. Bestalde, programa benetan exekutatzen denean azter dezakegu, beraz, programaren aldagai guztien ezagutza osoa dugu.

- VLIW planteamenduak normalean "estatiko" kategorian sartzen dira, non konpilatzaileak lan guztia egiten baitu.
- Ikuspegি supereskalarak normalean "dinamiko" kategorian sartzen dira, non hardware berezian prozesadoreak lan guztia egiten du. kontuan hartu hurrengo kode-sekuentzia:

```
sw $7, 4 ($2) lw  
$1, 8 ($5)
```

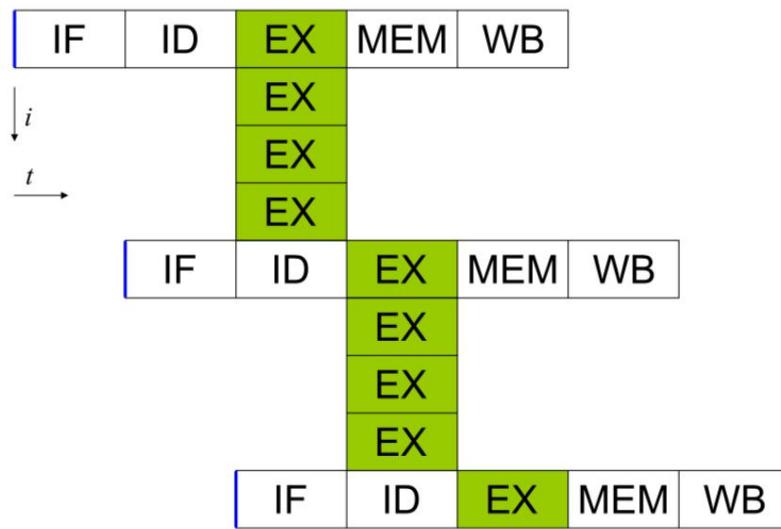
\$ helbideratze zuzena esan nahi du () zeharkako  
helbide indexatua

- demagun bi memoria-eragiketa paraleloan exekutatu ditzakegula [baina menpekotasunik ez badute, noski]. menpekotasunik al dago bi argibide horien artean? beno, \$5 eta \$2 balioen araberakoa da, hau da, 5 eta 2 memoria helbideen edukiaren araberakoa. \$5 helbidearen edukia 0 bada eta \$2 helbidearen edukia 4 bada, orduan bakoitzaren araberakoak dira. beste batzuk: zama baino lehen denda martxan jarri behar dugu eta ezin dira paraleloan exekutatu.

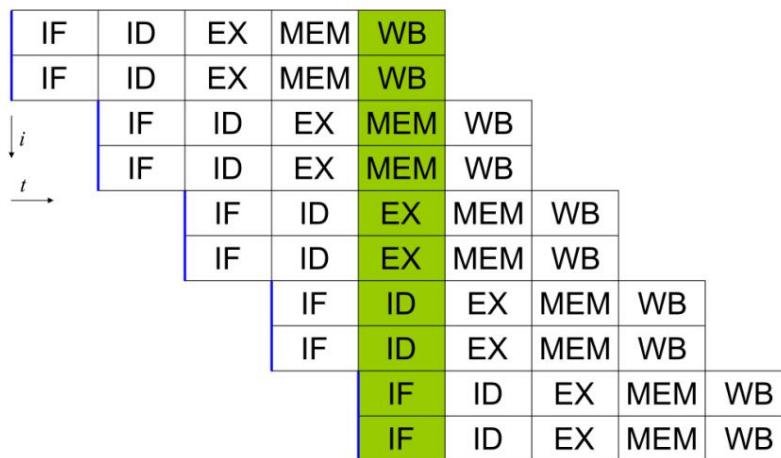
ŷ VLIW hurbilketa batean, gure konpilatzaileak erabakitzentzu dira seguruak paraleloan exekutatzeko. ez dago gure konpilatzaileak ziur esango dizu hemen menpekotasunik dagoen. beraz, alde seguruan geratu behar dugu, eta denda beti zama aurretik ibili behar dela agindu. hau kode zati handiagoa balitz, kodea aztertu eta menpekotasunik ez dagoela erakusten duen froga bat eraikitzen saiatuko ginateke. [Paralelizazio-konpilatzaile modernoek benetan hau egiten dute!]

ŷ Ikuspegি SUPERSCALAR bat erabakitzentzu dira, gure prozesadorean argibideak paraleloan exekutatu ditzakegungo erabakitzentzu duen hardware pieza bat dugu. arazoa errazagoa da, mendekotasun egiaztapen hori gure prozesadorearen hardware batean gertatuko delako, kodea exekutatu ahala. beraz, \$ 2 eta \$ 5 balioak zein diren jakingo dugu. Horrek esan nahi du beti jakingo dugula segurua den bi argibide hauetan paraleloan exekutatzeko. Hori dela eta, batuetan bi argibideak paraleloan exekutatuko dira eta beste batuetan ez.

ŷ Zorionez, inplikatutako konpromezuetako batzuk ikusiko dituzula. ikuspegি dinamikoek programaren informazio gehiago dute eskura, baina analisirako eskuragarri dauden baliabideen kopuru oso mugatua da. Adibidez, gure prozesadore supereskalarak kodean jarraibide independenteak bilatzeko nahi badugu, gauzak oso ilesu bihurtzen hasiko dira. Ikuspegি estatikoek programaren informazio gutxiago dute eskura, baina baliabide asko gasta ditzakete analisian. adibidez, konpilatzaile batentzat nahiko erraza da kodean jarraibide independenteak bilatzeko.



52. Irudia VLIW



53. Irudia Bi norabideko supereskalarra

### VLIW aplikazioak

- VLIW teknikaren ohiko aplikazioa "Digital Processing Signal DSP" prozesadoreak dira, hala nola TIren TMS320C6x eta ADIren Tiger-SHARC, audio, bideo, komunikazio eta abar seinaleak digitalki prozesatzen dituztenak denbora errealean. Ez dira prozesadore generikoak, baina ISA eta mikroarkitektura eragiketa matematikoak abiadura handian eta datu kopuru handiarekin egiten espezializatuak daude. Gaur egun bere merkatu handia telefonia mugikorra da. Adibidez: Cadence® Tensilica® Vision P6 DSP bere 10 nm Kirin 970 mugikorreko aplikazio prozesadorerako, Huawei-ren Mate 10 Serieko telefono mugikor berrietan estreinatu zena. VLIW SIMD arkitektura zabala dela eta, argibide multzo oso optimizatua eta adituen sintonizatutako irudien liburutegia dela eta, DSP plataforma ezin hobea da sortzen ari diren irudien aplikazioetarako, hala nola 3D sentsazioa, gizaki/makina interfazea, AR/VR eta plataforma mugikorrentzako identifikazio biometrikoa.

## 7.10. CISC/RISC arkitekturak

### 7.10.1. Sarrera

- CISC: Instruction Set Computer
- RISC: Instrukzio-multzo murriztua ordenagailua
- CISC eta RISC ordenagailuen diseinuko bi filosofia dira, bi arkitektura.

## 7.10.2. CISC

- Adibideak: Motorola 68k, Intel x86.
- Helburua DRAM memoria gutxi hartzen zuten programak lortzea zen, hasieran memoria garestia baitzen. Eragiketa konplexuak egiten zitzuten argibideak lortu behar ziren, programa oso luzea ez izateko.
- Errepertorioak argibide anitzeko eta ez-uniformeak biltzen ditu.
- HW konplexu bat behar du, espazio asko hartzen duena eta erloju-ziklo asko behar dituena.
- Mihiztadura-lengoaiaren arkitektura goi-mailako lengoia batetik hurbil dago, hala nola C hizkuntzara. horrek lana errazten die kopilatzaileei eta mihiztzaileen programatzaleei.
- Bestalde, CPUa bezalako hardware elementuen diseinua eta ezarpema zaitzen du.

## 7.10.3. RISC

- Adibideak: PowerPC, ARM, MIPS eta SPARC
- Jatorria aurrekontu baxuko mikroprozesadore bat diseinatzea zen, beraz, apustua hardware simpleko mikroarkitektura bat izan zen, argibide simple eta arruntak exekutatzen zituena.
  - ÿ HW simple bat azkarra da eta txip eremu txikia hartzen du. Erraza da etapa kopurua handitzea hoditeria.
  - ÿ Ondorioa gerora haririk gabeko merkatu osoa hartuko zuen kontsumo baxuko mikrofonoa izan zen. telefonia mugikorra.
  - ÿ Instrukzio-errepertorioa eragiketa-kode eta helbide gutxirekin murrizten da simplea, eta horrek makina-argibide ugari dituzten programak sortuko lituzke.
- Desegokia:
  - ÿ Memoria-gaitasun gehiago behar du programa handiak gordetzeko. Utzi a izateari deserosoa DRAM memoria unitateak merkeago bihurtzen badira.
  - ÿ Memoria sarbide ugari argibideak, eragigaiak eta emaitza harrapatzeko. Jada ez da eragozpen bat latentzia baxuko memoriak erabiltzen badira.
  - ÿ Irtenbidea: barne memoria handitu: barneko Erregistro kopurua eta cache memoria. Eremu murriztuaren nukleoaren ondorioz lekua dago
- Eremu murriztuaren nukleoak aukera ematen du:
  - ÿ Barne memoria-ahalmen ultra-azkarra handitu.
  - ÿ arkitektura supereskalarren moduen kopurua handitzea.

## 7.10.4. Gaiak

- Zein arkitekturak optimizatzen du programaren bytearen tamaina?
- Zein arkitekturak optimizatzen duen instrukzio bakotzaren exekuzio-denbora
- Zein arkitekturak optimizatzen dituen CPUaren tamaina eta fabrikazio kostua
- Zer arkitekturak optimizatzen duen kontsumoa
- Zer arkitekturak optimizatzen duen memoria-harrapaketa kopurua. Ba al dago independentzia harrapaketa eta argibideak betetzea?

## 7.10.5. S.W.

- RISC arkitektura muntatzeko programa batek CISC batek baino argibide gehiago ditu
- RISC instrukzio bakotza CISC bat baino denbora gutxiagoan exekutatzen da.

#### 7.10.6. Konparazio taula

- RISC vs CISC

#### 7.11. Mikroarkitektura modernoak

- Eranskinean honako mikroarkitektura hauek azterzen dira:
  - ŷ ThinkPad TL560 ordenagailua: Intel x86-64 CPU Skylake mikroarkitektura
  - ŷ Huawei 30 Pro telefonoa: ARM Cortex-A76 eta ARM Cortex-A55

#### 7.12. Ariketak

- William Stalling testuliburuko 12. kapitulua.
- William Stalling testuliburuko 13. kapitulua
- Ariketak

# 8. Kapitulua. Sarrera/Irteera mekanismoak

## 8.1. Ikastaroa

8. Sarrera/irteera sistema
  - a. Inkestaren sinkronizazioa
  - b. Sinkronizazioa eten
  - c. Eten bektorea
  - d. Zuzeneko DMA memoriarako sarbidea
- eta. Sarrera/irteera errutinen mutaia-lengoia programatzea

## 8.2. Bibliografia

- Testu liburua:
  - ÿ Ordenagailuen Egitura eta Antolamendua. William Stallings: 7. kapitulua

## 8.3. Periferikoak

### 8.3.1. Adibideak

- Teklatua
  - Monitorea
  - HDD
  - Sarea
    - ÿ LAN
    - ÿ Wi-Fi
  - Kanpoko periferikoa
    - ÿ Pen Drive
- Adibide bakotzeko, honi buruzko informazioa:
- ÿ eredua eta ezaugarriekin lotura
  - ÿ interfazea: bus elektrikoa, komunikazio protokoloa
  - ÿ banda zabalera

### 8.3.2. Eredua

- Euskarriak: Magnetikoak (HD), Mekanikoak (Robot), Optikoak (CD), Elektrikoak (pen drive), etab.
  - ÿ Elektronika analogikoa.
- HW gidaria
  - ÿ Komunikabideekiko interfazea:
    - ÿ euskarrien gainean eragiten duten seinaleak mota ezberdinak dira: optikoak (arinak), mekanikoak (pneumatikoak), akustikoak, etab. Seinale hauek seinale elektriko baten eraldaketatik lortzen dira normalean: interfaze elektrikoa/optikoa, elektrikoa/mekanikoa, elektrikoa/akustikoa.
    - ÿ Adib.: Hitzun bat
    - ÿ Adib.: Disko optikoa duen Laser bat.

ŷ Adib.: Laser modulatzaire elektronikoa

- Kontrolagailu periferikoa (**MCU: MicroController Unit**)

ŷ [Disko kontrolagailu baten irudia](#)

ŷ Kontrolatzaileak gidari HW-ri aginduak ematen dizkio

ŷ **Aginduak** (zeregin periferikoetarako hizkuntza espezifikoa) interpretatzen dituen sekuentziatzalea da , zeinen exekuzioak periferikoari berariazko funtzioak beteko dituen.

ŷ Aginte hizkuntza. Komando multzoen arkitektura (CSA) ISA?

ŷ [SCSI hizkuntza](#)

ŷ [ATA / ATAPI hizkuntza](#)

ŷ ATA Komando Multzoa (ACS): IDENTIFY, READ DMA, WRITE DMA eta komandoen adibidea  
URTU CACHE komandoak

ŷ datuak transferitzeko komandoak, kontrol komandoak (eragiketa mekanikoak, hala nola biraka), probako aginduak (periferikoen egoera: konektatuta, deskonektatuta)

ŷ Adib.: diskoko baten kasuan "biratu zenbait biratan" komandoa. Diskoak bere sekuentziadore propioa integratzen du, MCU bat.

- Firmwareea

ŷ Komando periferikoen multzoa MCU kontroladorearen memorian kargatutako softwareak (firmwareak) interpretatzen du. Software hau periferikoen fabrikatzaleak grabatu du. Erabiltzaileak periferikoaren konfigurazio-parametro batzuk bakarrik idatzi ahal izango ditu periferikoan, eta Firmwarea fabrikatzaleak soilik eskura dezake.

## 8.4. Teklatua

- Egitura



- Kodeak



- Gidaria

## 8.5. Konputagailuen Arkitektura

### 8.5.1. Von Neumann

- 3 oinarrizko unitate
- I/O kontrolagailua

ŷ Kontrolagailu laguntzailea, ez zentrala, I/O eragiketetarako berariazko dedikazioa duena.

- Sarrera/Irteera azpisistema: Von Neumann ereduaren 3 osagietako bat da.
- Makinara sartzeko beharrezko da:

ŷ Sartu programa: Biltegiratze euskarri batetik (paperak, diskoa, etab.) programa memorian kargatu behar da.

ŷ Makinak sortutako emaitza ateraz: Memoriatik emaitzak biltegiratze euskarri batean gorde behar dira (diskoa, inprimagailua, etab.), bistaratuz (pantaila, etab.), transferitu (sarea, etab.).

- Gailu Periferikoak

ŷ PUZ-MEMORIA tandemaren zerbitzuak osatu eta zabaltzen dituzten hardware baliabideak dira, programatzalea eta erabiltzailearen zereginak erraztuz.

ŷ Aniztasun handia: teklatua, monitorea, sagua, diskoa, bideo-txartela, sare-txartela,...

ŷ Teklatu baten eta disco gogor baten arteko konplexutasun-aldea

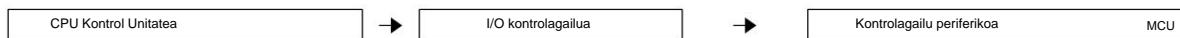
- CPUa normalean programa guztiak eta periferiko guztiak partekatzen duten baliabide bakarra da.

### 8.5.2. CPU-I/O konexioa

- Arkitektura, beraz, bi kontrolagailuz osatuta dago: CPU eta MCU. CPUak kontroladore generalista (CPU) du, periferikoak, berriz, kontroladore oso zehatza (MCU). CPUaren makina-lengoaia orokorra da, eta periferikoaren makina-lengoaia oso zehatza da.

ŷ Periferikoak prozesadore propioa duen **zerbitzari**- makina bat bezala jokatzen du. **Ostalari**- makinaz eta **zerbitzari** -makinaz hitz egin dezakegu . Ostalariaren kontrolatzalea ordenagailuaren CPUa da eta zerbitzariaren kontrolatzalea periferikoaren MCUsa.

ŷ CPUa ez da MCUsarekin zuzenean komunikatzen, baina periferikoen zeregina prozesadore ez zentralei eskuordetzen die, hau da, I/O kontrolagailuei. Ordenagailuaren arkitektura tipikoa Prozesadore Zentral batena eta Intelek **Chipset** deitzen duen I/O kontrolagailuen multzoa da .



ŷ Adibidea: Disko gogorra

ŷ [Seagate Momentus 7200.4 500GB 7.2K 2.5 hazbeteko SATA disco gogorra ST9500420AS](#)

ŷ [besoen mugimenduaren bideoa](#)

ŷ **Disko-buffera:** Unitatearen transferentziaren eta atakako I/O transferentziaren arteko interfaze-memoria.

ŷ [Etxeko Atmel kontrolagailua:](#) Sata diskoa Atmel kontroladorearekin eta Ethernet interfazearekin.

### 8.5.3. I/O kontrolagailua

#### Sarrera

- Urruneko periferikoa:

ŷ Adibidez:PC---ŷSATA---ŷDisko trinkoa

- PC I/O kontrolagailua:

ŷ EZ CPU Kontrol Unitatea

ŷ Von Neumann arkitekturaren oinarritzko 3 elementuetako bat da

ŷ PUZak instrukzioen exekuzioa I/O kontroladore izeneko beste kontrolagailu bati delegatzen dio sarrera/irteera makina.

ŷ Interfaze bat beharrezkoa da CPU eta PERIFERIKOAK

ŷ ISA: CPU sarrera/irteera makinaren argibideak: irakurketa (IN) eta idazketa (OUT).

Komunikazioa bi norabideetan.

ŷ Periferikoko DATU eta KANDOAK transferitzen dira.

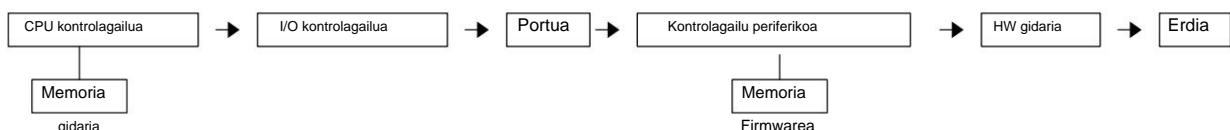
ŷ Williamek I/O kontrolatzailari I/O Modulua deitzen dio

ŷ Kontrolagailu periferikora komandoak eta datuak transferitzen dituen I/O kontrolatzaila da.

ŷ Kontrolagailu periferikoak I/O kontrolagailutik jasotako KOMANDOak interpretatu eta idatziko ditu  
edo/eta DATUAK irakurriko ditu.

ŷ Adibidea: Host Controller Interface aurreratua (AHCI) Intel-ek definitutako estandar tekniko bat da  
Serial ATA (SATA) io kontrolagailuaren funtzionamendua zehazten duena (ostalari-busaren egokitzaila).

- Egitura



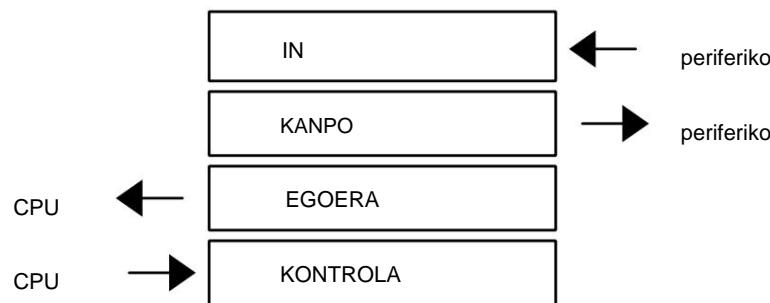
## Portuak

- Portuak i/o kontrolagailuan implementatutako memoria-erregistroak dira.

ŷ Portu bat erregistro mota ezberdinek osatzen dute: datuen sarrera, datuen irteera, egoera  
periferiko, kontrol periferiko

I/O kontrolagailua

portua



- I/O kontrolagailuak bere portuen bidez komunikazioak kontrolatzen eta exekutatzen ditu.

ŷ Adibidez: SATA ataka duen I/O kontrolagailua

- Komunikazio ataka: beste solaskidearen sarbidea (kasu honetan periferikoa)

- I/O kontrolagailu batek hainbat ataka izan ditzake eta hainbat periferikorekin komunikazioak kontrolatu ditzake.

- Ataka bat hainbat periferikok parteka dezakete

- Linux

ŷ cat /proc/ioports

#### 8.5.4. helbide-espazioa

- I/o kontroladorearen atakaren i/o helbideak bi modutara implemenatzen daitezke:
  - ÿ memoria nagusian mapatutako atakak ÿ MMIO
  - ÿ memoria nagusia ez den beste espazio bateko ataka-helbideak: i/o espazioa. IO edo PMIO

##### Memoria-mapatutako I/O (MMIO)

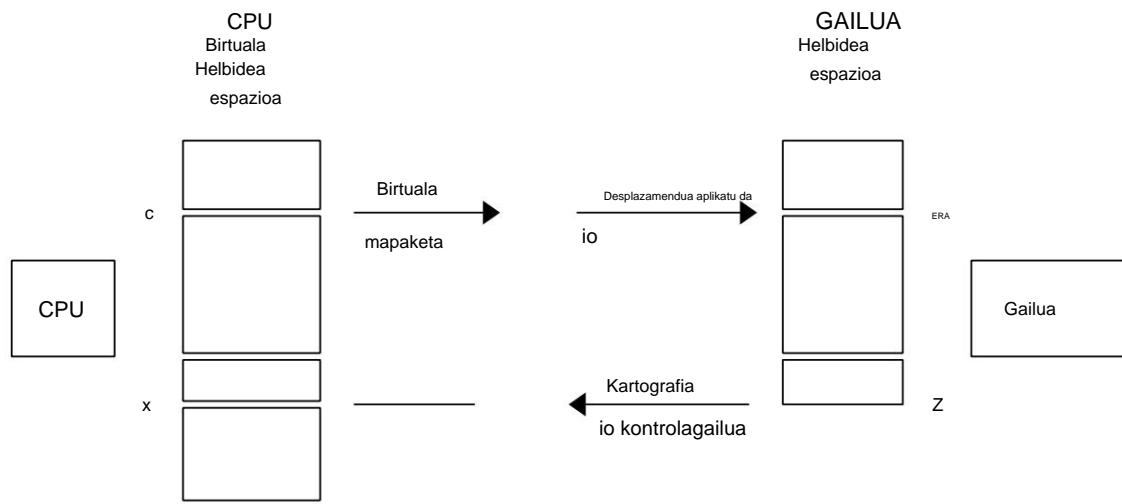
- Memoria Nagusia Helbide Espazioa
- Sistema-busak memoria helbide-espazioa partekatzen du I/O gailuen eta programaren memoriaren artean
- I/O-ren interfazea lehen mailako memoria-kokapenen multzo gisa tratatzen da
- Software kontrolatzaillek gordetako edo berreskuratutako datuen esanahia zehazten dute.
- Memoria espazio bat galtzea (8086 - 300K), I/O interfazeetarako gordeta dagoelako. Garrantzi gutxiagorekin 4 GiB helbide-espazioa.
- Eskuragarri dauden instrukzio-modu guztiak ÿ Instrukzio-errepertorio osoa, ez bakarrik IN,OUT.
- Memoria-busaren sarbidea moteldu dezake.
- Ondoko memoria-barrutia mugatu edo zaildu dezake.
- Jatorrizko x86 arkitekturak 1 MiB-ko muga zuen, I/O 640K-tik gora mapatuta zegoelako.

##### Portu mapatutako I/O (PMIO)

- I/O helbide-espazioa
- PUZak argibide-multzo bereiziak ditu PUZaren pin zehatzetara sartzeko, ataka gisa jarduten dutenak edo a eragiten dutenak demux helbidea eta datuen pin-outak i/o-ra loturiko beste lerro multzo batera konektatzeko.
- I/O gailuei lotuta dagoen bus bereizia. CPU memoria-busa erabil daiteke I/O gailuek erantzuten duten bitartean.
- PUZaren diseinuari konplexutasuna gehitzen dio.
- Sarritan argibide multzo mugatua. Baliteke memorian idatzi behar izatea beste ekintza batzuk egin aurretik.

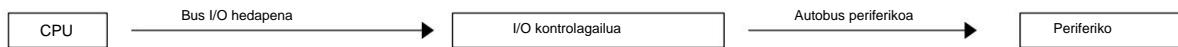
##### Helbide periferikoak

- Gidaria
  - ÿ Driver-programa memoria birtualaren mekanismoa gainontzekoak bezala erabiltzen duen i/o prozesu bat da prozesuak.
  - ÿ Mapa /proc/iomem fitxategian
- Periferiko
  - Periferikoek erabiltzen duten helbide-espazioari gailuen helbidea deitzen zaio, periferikoak ikusgai dauden helbideak direlako, bere helbide-espazioa ordenagailutik independentea da.
  - Beharrezkoa izango da mapeatu, itzuli, gailuen helbideak CPUak ikusgai dauden helbide birtualetara
    - ÿ Mapa /proc/ioports fitxategian



### 8.5.5. Autobusak

- Autobus motak
  - ÿ Prozesadorea: barneko busa CPUna
  - ÿ memoria: memoria kontrolagailuaren eta memoria nagusiaren arteko busa
  - ÿ sistema: CPUtik kanpoko busa, kanpoko gailuak interkonektatzeko, hala nola memoria nagusia eta I/O kontrolagailu periferikoak.
  - ÿ lokala: banda zabalera handia ahalbidetzen duen I/O bus laburra
  - ÿ Hedapena: hainbat txartel konektatzeko aukera ematen duen I/O bus luzea
  - ÿ periferikoa: kanpoko gailuak ordenagailura konektatzeko aukera ematen duen I/O busa



- Tokiko autobus i/o; vlb, PCI, AGP
- Hedapeneko I/O busa: ISA, EISA
  - ÿ I/O txartelaren konexio zuzena plaka amaren hedapen-busarekin zirkituen bidez:
- Bus periferikoa: SCSI, SATA, USB, RS232
  - ÿ Kanpoko konexioa kable bidez
- Io autobusaren arkitekturaz osatuta dago
  - ÿ Interfazea (kablea eta konektorea)
  - ÿ Komunikazio-protokoloa: sistemaren eta i/o gailuen arteko elkarrekintza koherentea izateko arau estandarizatuen multzoa, propietate fisikoak, sarbide-metodoak, datu-formatuak, etab. Autobusak izena ematen dio protokoloari.
  - ÿ Aginte-lengoiaia
- Adibide praktikoak
  - ÿ ISA
    - ÿ Industriako Arkitektura Estandarra
    - ÿ PC/XT 8086 (1983) 8 bit

ÿ 4 DMA kanal

ÿ PC/AT i286 (1984) 16 bit

ÿ 16 MB/s

ÿ 7 DMA kanal

ÿ 11 IRQ linea

ÿ EISA

ÿ Industriaren Arkitektura Estandar Hedatua

ÿ PC Kiona: i386-i486 (1988)

ÿ 32 bit

ÿ IBMren MCA jabedunaren alternatiba klonatzen du zure PS/2n

ÿ 33 MB/s-ko transferentzia-abiadura autobus maisuetarako eta DMA gailuetarako

ÿ 7 DMA kanal

ÿ 15 IRQ linea

ÿ MCA

ÿ Mikrokanalen Arkitektura

ÿ IBM PS/2 (1987)

ÿ 32 bit

ÿ PCI: Osagai Periferikoen Interkonexioa

ÿ PCI Express

ÿ banda-zabaleren zerrenda

- <http://www.karbosguide.com/hardware/module2b2.htm>

ÿ I/O kontrolagailua zeharka sistemaren busera (CPU-MP) konektatuta dago jumperen bidez.  
(zubiak)

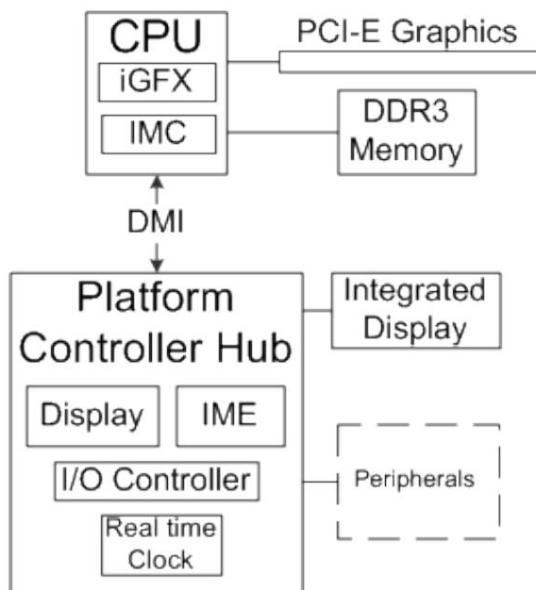
- Intel

ÿ Intelek Hego eta Iparraldeko zubiak dituen ICH zubietatik PCH Zentral Kontzentratzaile batera eboluzionatu du.

ÿ Kontuan izan CPUak memoria-kontrolagailu integratua (IMC) eta bideo-ir/i-ko kontrolagailuak (PCI-E grafikoak) integratzen dituela.

#### 8.5.6. Analisia: Lenovo T520 - Disko gogorra

- Lenovo ordenagailuko disco gogor eta memoria nagusiaren arteko datu-transferentziaren bidea T520
- Diskoa (ATA diskoa, ST9500420AS Seagate) ÿ Kontrolatzaile mekanikoa/elektrikoa/magnetikoa ÿ Disko mikroa (SATA Interfazea, Seagate) ÿ Serial I/O Bus (SATA 6Gb/s) ÿ Ostalariaren egokitzalea (Platform\_Controller\_Hub PCH, ChipSet 200C/6 seriea, SATA AHCI kontrolatzailea) ÿ Pantilarako interfaze malgua (FDI) ÿ CPU (Intel Core i5)
- SATA: Serial Advanced Technology Attachment ordenagailuaren bus interfazea da, ostalari-busaren egokitzaleak (disko kontrolatzailea) biltegiratze masiboko gailuekin (MCU, MicroControllerUnit), hala nola disco gogorrarekin.



54. Irudia Lenovo T520

## 8.6. I/O programa

- I/O programatua (PIO)

ÿ Konsulta I/O mekanismoak erabiliz datu-transferentziak I/O programa batek egiten ditu (PIO) CPUak exekutatuta. PUZa memoria eta periferikoen arteko datu-transferentzia bakoitzean Transferentzia amaitu arte IZAN behar duzu.

### 8.6.1. Iturburu-modulu

- 512 byteko transferentzia 0x380 ataka eta buffer baten artean.

```

mov %bx,buf ; helmuga helbidea. BX buffer baten erakuslea da
mov $512,%bai ; zenbatu. Transferitu beharreko byte kopurua
mov $0x380,%dx ; iturburu-ataka. DX atakaren erakuslea da

begizta:
    %dx,%al-en; lortu byte i/o atakatik. AL<-DX
    mov %al,(%bx) ; gorde M[bx]<-AL buffer batean
    inc %bx ; hurrengo memoria-kokapena buf-en
    abendu %bai ; gutxitu byte geratzen dira
    jnz begizta

```

### DA

- IN: atakako datuak irakurri
- OUT: datuak portuan idatzi

## 8.7. Gidaria: Sistema Eragilea

### 8.7.1. I/O kudeatzailea: hierarkia

- I/O eragiketen kudeaketa Sistema Eragileak egiten du
- Sistema eragilearen I/O kudeatzaile programaren egitura egitura hierarkiko batean oinarritzen da

mailak:

- ÿ Maila baxuena: periferikoko hw i/o kontrolagailuaren sw kontrolatzailea (driver modulua).
- ÿ Maila gorena: Fitxategi sistema birtuala. Aplikazioak periferikoetara sartzen dira hauen abstrakzioa fitxategi birtualetan.

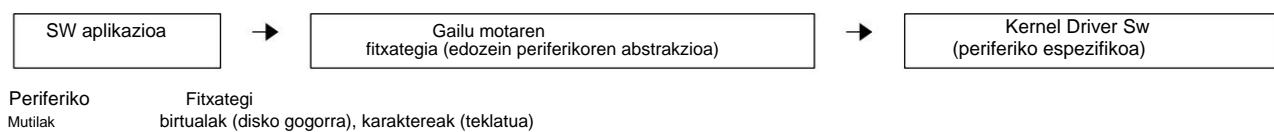
#### 8.7.2. Iturburu kodea

- Sasi-kodea

```
Bitartean (EGOERA ==
LANPETUA) ; // itxaron gailua okupatuta egon ez arte. Portu okupatua.
Idatzi datuak DATA erregistroan // atakatik kanpora transmititzeko datuak Idatzi komandoa COMMAND
erregistroan // kontrola erregistroan // Hori eginez gero, gailua abiarazten da eta komandoa
exekutatzen du While (STATUS == BUSY) ; // itxaron gailua zure eskaerarekin amaitu arte
```

#### 8.7.3. Kontzeptua

- Funtzio periferikoak implementatzen dituen programa DRIVER izeneko nukleoaren modulua da.
  - ÿ Teklatuaren kontrolatzailea, monitorea, disko gogorra,
- SW geruzak:



- Adibidea: fitxategi bat disko gogorrean idaztea

ÿ idatzi ÿ syscall ÿ OUT ÿ HD-ren komando propioa



ÿ Erabiltzaile-espazioa: idaztea (datuak idazteko funtzioa), syscall (datuak kudeatzeko modulurako deia).  
Sistema eragilea I/O)

ÿ OS kernel espazioa: Driver: PUZak interpretatzen duen ordena eta I/O kontrolagailuak exekutatzen du  
datuak (aginduak eta datuak) Memoriaren eta Kontrolagailu Periferikoaren artean transferitzeko

ÿ Espazio periferikoa: Periferikoak (Firmwarea) eta Datu-transferentziak interpretatu beharreko komandoak.

#### 8.7.4. Gidaria erabiliz

- Gidaria Sistema Eragileak babestuta dago. ioctl bezalako funtziok daude erabiltzaileari gidariarekin elkarrelagiteko aukera ematen diotenak.

ÿ Erabiltzailearen eta gidariaren arteko interfazea sistema eragilerako deiak dira.

ÿ Makinaren instrukzioa SYSCALL (x86-64) edo int 0x80 (x86-32) erabiliz zeharka deitzen dugu.  
gidariaren funtziotara sistema eragilearen bidez.

- Adibidea

ÿ Inprimatu pantailara: ireki, idatzi, itxi ÿ ireki eta itxi fitxategi-sistemarekin elkarreragin birtuala.

## 8.8. I/O Interfazea Ezartzeko Mekanismoak

### 8.8.1. Sarrera

- CPUan edo CPU eta memoria primarioaren artean zuzenean egiten ez den datuen manipulazio guztiak I/O dira.
- PIO:Inkesta
- Etenaldia
- DMA: Memoriarako Sarbide Zuzena

### 8.8.2. Inkestaren sinkronizazioa

- Inkesta: inkesta
- Konsulta: inkesta
- Mekanismoa

ÿ Egoera egiaztatzea edo galdeketa

ÿ CPUak periferikoak konektatuta dauden ataka bakoitzaren egoera-erregistroa konsultatzen du.

Egiaztu periferikoren batek CPU zerbitzua behar duen. I/O gailu bakoitzeko erregistro bat erreserbatzen du. Erregistro bakoitza etengabe galdetzen da datuen etorrera detektatzeko.

ÿ Beharrezko da arreta periferikoko programak exekutatu behar dituenean: Sinkronizazioa.

Sinkronizatzeak esan nahi du ostalariaren eta periferikoaren artean transferentzia noiz egingo den zehaztea edo adieraztea.

ÿ Ostalariak i/o kontroladorearen egoera-bitia galdetzen du

ÿ Identifikazioa

ÿ Bezeroaren eskaera onartutakoan (i/o kontrolatzalea)

ÿ Kontrolatzaleak zerbitzua eskatzen duen periferikoa identifikatzen du

ÿ Komunikatu sistema eragileari zein periferiko

- Egitura

ÿ CPU:

ÿ I/O programa exekutatzen du: I/O eragiketa ZUZENEAN kontrolatzen duen programa:  
Programatutako I/O ÿ PIO

ÿ memoria nagusiaren eta I/O kontrolagailuaren arteko transferentziak egiten ditu

ÿ itxaron periferikoa amaitu arte. PUZak I/O eragiketa amaitu arte itxarongo du.

ÿ Memoria nagusia: programaren I/O gordetzen du

ÿ I/O kontrolagailua

ÿ Portua: Portu bat datu, kontrol, proba motako ERREGISTROek osatzen dute

ÿ datuak periferikora transferitzen ditu

### 8.8.3. Etenaren sinkronizazioa

- Etenaldiak gidatutako I/O (Etendura I/O mekanismoa)
- Egitura

ŷ CPU:

- ŷ Etenaldirako arreta programa exekutatzen du. ZUZENEAN kontrolatzen duen programa eragiketak.
- ŷ memoria nagusiaren eta I/O kontrolagailuaren arteko transferentziak egiten ditu
- ŷ ez du itxaron periferikoa amaitu arte. Egin behar den bakoitzean eten egiten da transferentzia bat
- ŷ PUZak exekutatutako instrukzio bakoitzaren instrukzio-zikloaren amaieran, eten-eskaerako seinalea aktibatuta dagoen egiaztatzen du.
- ŷ Memoria nagusia: programaren I/O gordetzen du
- ŷ I/O kontrolagailua
  - ŷ Portuak: datuak, kontrola, proba
  - ŷ datuak periferikora transferitzen ditu

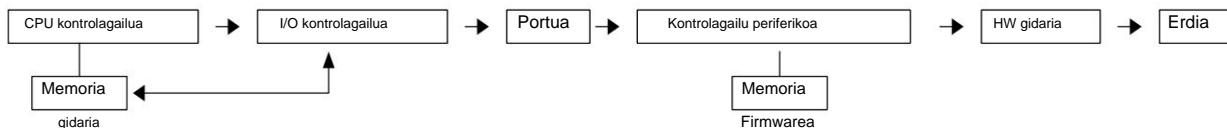
- CPUari beste gauza batzuk egiteko aukera ematen dio I/O eskatu arte

- ŷ Eten eskaera - I/O gidatua (Oraindik PIO - CPUak datuak mugitu behar ditu)
- ŷ I/O gailuek edozein unetan etenaldi batekin PUZaren arreta eska dezakete, baina behar denean bakarrik.
- ŷ CPUak denbora luzea eskaini diezaiokе gailu jakin bat.
- ŷ PUZak ez du arretarik behar ez duen I/O-n egiaztatu beharrik.
- ŷ CPUak I/O eskaerak prozesatzea atzeratu dezake.
- ŷ Sistema berriagoak - CPU-k datuak bigarren mailako kontrolagailura transferitzen ditu, eta horrek zeregina edo arazoa amaitzean bakarrik eteten du CPU-a.
- ŷ Kanpoko zirkuituak behar ditu.
  - ŷ Adib. 8259A eten-kontrola programagarria (PIC). Baliteke CPUak PICarekin komunikatu behar izatea eskaera egiten duen gailua identifikatzeko.

- Programatutako sarrera/irteera (PIO) CPU eta periferiko baten artean datuak transferitzeko metodo bat da, hala nola sare-egokitzalea edo ATA biltegiratze-gailu bat. Oro har, programatutako I/O geritzen da CPUan exekutatzen den softwareak I/O helbide-espazioan sartzen diren I/O argibideak erabiltzen dituenean I/O gailu batera edo gailu batetik datuak transferitzeko. Hau Direct Memory Access (DMA) transferentzien kontrakoa da.
- Programatutako I/O erabiltzen duen ordenagailuko gailu baten adibiderik ezagunena ATA interfazea da;

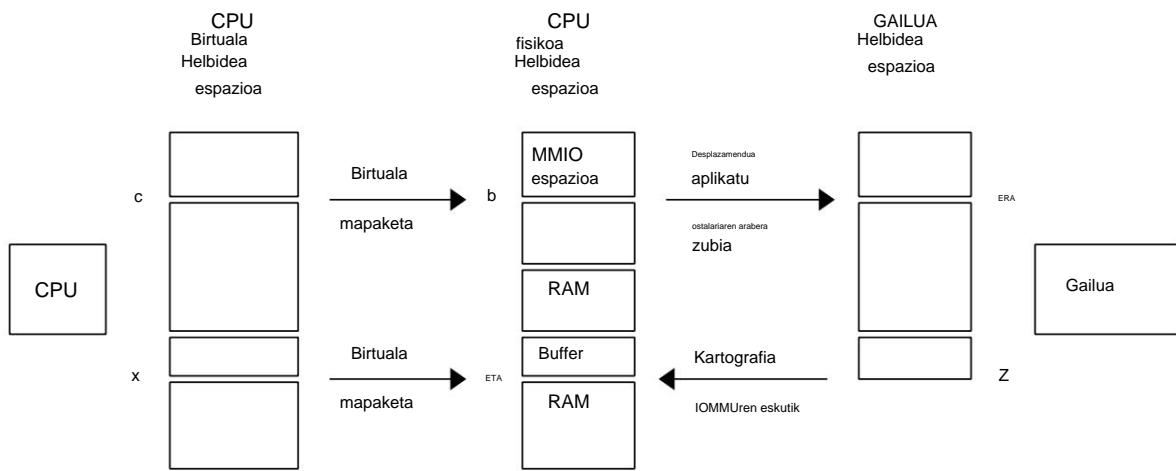
#### 8.8.4. Zuzeneko memoriarako sarbidea (DMA)

- Egitura



- Helbideen mapaketa
- Hardware-unitate bat beharko da DEVICE helbide-espazioa helbideetara itzultzeko.

Memoria Nagusi fisikoa: iommu



- Egitura

ÿ CPU:

ÿ Exekutatu i/o programa. Programak ez du transferentzia kontrolatzen baina hasieratzen du (kopurua transferitzeko byteak, kokapena memoria nagusian, kokapena periferikoan, kontrola akatsak, etab)

ÿ DMA kontrolagailura (DMAC) transferentzien kontrola ematen du, I/O prozesatzea batera deskargatzen du. Xehetasunak zaintzen dituen helburu bereziko txipa. Transferentzia DMAC-ek kontrolatzen eta egiten du Hardwarearen arabera ÿ Ez da PIO bezalako programaren arabera.

ÿ Memoria nagusia: programaren I/O gordetzen du

ÿ I/O kontrolagailua

ÿ DMA kontrolatzalea da

ÿ Portuak: orain portuak ez dira datuak transferitzeko, kontrolerako soilik CPU-DMA

ÿ datuak memoria nagusiaren eta periferikoen artean transferitzen ditu

ÿ Kontrolatzaleak ez du periferikoaren zain geratzen

ÿ Zuzeneko Memoria Sarbide kontrolatzalea.

I/O interakzioa kudeatzen du ondoren CPUaren esku-hartzerek gabe  
hasierakoa

CPU interakzioa. Etenaldi erabiltzen ditu CPUra egoeraren berri eramateko.

Arbitraje-protokolo bereizia behar du - busak CPUarekin partekatzen ditu.

Aurrez definitutako zeregin estandarizatua.

CPU EZ dago okupatuta baina baliteke baliabideak lortzeko lehiatu behar izatea.

### 8.8.5. I/O kanala

- I/O prozesadore dedikatuak erabiltzen ditu

ÿ Kanalaren I/O (Mainframe edo Superordenagailua)

ÿ Egitura: DMA unitatea gehi prozesadore zehatz bat integratzen ditu.

ÿ Programagarria: memoria nagusian gordetako kanal-programa exekutatzen du. (Horekiko aldea DMA).

ÿ Datuak transferitu (Memoria nagusia <ÿ Periferikoa) CPUtik independenteki

#### Memoria partekatua

- Egitura

ÿ CPU

ÿ prozesadoreari edo I/O kanalari transferentzien kontrola ematen du

ÿ Memoria nagusia: programaren I/O gordetzen du

ÿ I/O prozesadorea

ÿ I/o kanala da

ÿ Memoria nagusian gordetako I/O programa exekutatzen du

ÿ Portuak: orain portuak ez dira datuak transferitzeko, kontrolerako soilik  
CPU-DMA

ÿ datuak memoria nagusiaren eta periferikoen artean transferitzen ditu

ÿ Memoria Nagusia

ÿ CPU eta IO\_Channel-en artean partekatua

#### Memoria Independentea

- Egitura

ÿ CPU

ÿ prozesadoreari edo I/O kanalari transferentzien kontrola ematen du

ÿ Memoria nagusia: programaren I/O gordetzen du

ÿ I/O prozesadorea

ÿ I/o kanala da

ÿ Memoria nagusian gordetako I/O programa exekutatzen du

ÿ Portuak: orain portuak ez dira datuak transferitzeko, kontrolerako soilik  
CPU-DMA

ÿ datuak memoria nagusiaren eta periferikoen artean transferitzen ditu

ÿ Memoria Nagusia

ÿ CPUak soilik eskura dezake

ÿ IO memoria

ÿ IO\_kanalaren bidez bakarrik eskura daiteke

### 8.9. Etenaren sinkronizazioa

- I/o-ko etenaldi-interfazearen ezarpenari buruzko aurreko atalaren hedapena.

#### 8.9.1. Kontzeptua

- Polling-en eragozpena da CPUak kontsulta egiten duela periferikoak behar ez badu ere zerbitzuak.
- Periferikoak ekimena hartzen du eta exekutatzen ari den programa ETENA eskatzen du periferikoak eskatzen duen programa exekutatu

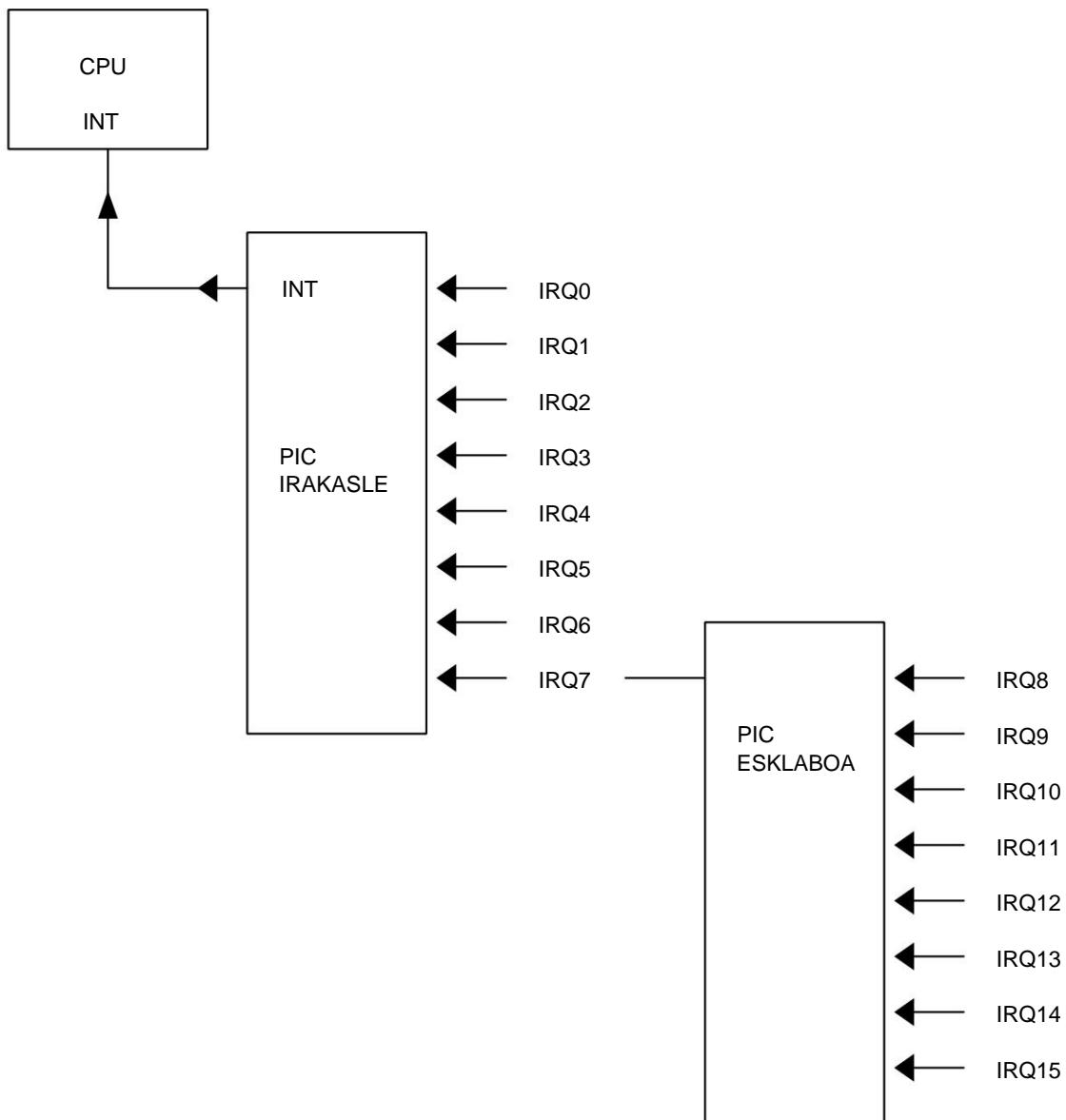
## 8.9.2. Eten-mekanismoa

- Periferikoak, PUZaren sarrerako potentzia-lerro baten bidez, I/O kontrolatzalea eskatzen du zerbitzuak.  
nukleoak
  - ÿ Nukleoa **ETEN** egingo da
- CPUak bi eten lerro ditu:
  - ÿ Eten eskaera (IRQ) Line: maskagarria
  - ÿ Ez Maskable Eten (NMI) Linea
  - ÿ Instrukcio-zikloko CPUak Etenaren Egiaztapenaren fasea du azken fasean
    - ÿ Nukleotik zerbitzu bat eskatzen bazaio, CPUak etenaldien arreta moduan sartzen da eta kontrola Kernelaren Eten **Kudeaketa** modulura pasatzen du.
- Etenaldia eskatzeko linea periferikoei (IRQ) deitzen zaie.

## 8.9.3. Eten kontrolatzalea

### PIC

- Eten-kontrola programagarria
  - ÿ **ARGAZKIA** : Eten-kontrola programagarria
    - ÿ PUZA nukleo bakarra duen arkitekturan erabiltzen da.
    - ÿ Sarrera gisa eten-lerro periferiko guztiak ditu
    - ÿ Irteerak: INT (eten eskaera) eta D0-D7 (kontrola, egoera eta eten-bektorea)
- Adibidea: [PIC 8259](#)
- Daisy-kate konexioa
  - ÿ Maisua - Esklaboa



- PIC kontroladoreak egindako ekintzak

ÿ Egiaztu seinaleren bat aktibatuta dagoen (Monitorizazioa). Bat baino gehiago aktibatzen badira, lehentasuna izango da. maila baxuenera eta irabaziak:

ÿ Aktibatutako IRQn lerroa bektore bihurtzen du (0x00-0xFF)

ÿ Idatzi bektorea PIC I/O atakan. Portua CPUak eskura dezake. Bektorea eten zerbitzuaren errutinarako erakuslea duen taula baten sarrerara seinalatzen da (ISR).

ÿ PUZaren INTR seinalea aktibatzen du

ÿ CPUak balio bektoriala irakurtzen badu, INTR seinalea desaktibatu egingo da

- IRQn lerroa Intel-en arabera lehenespenez n+32 bektoreari dagokio. Mapa hori alda daiteke PIC programatzea.

ÿ IRQ0 lerroa 0x20 bektoreari dagokio

- IRQ lerro bakoitza programaren arabera ezezta daiteke, baina eten hori ez da galtzen.

- cli instrukzioa erabiliz, EFLAGS registroko IF bandera garbitzen da, CPUak alde batera utziz

hardware-etenaldiak.

ÿ **Sli** instrukzioa erabiliz IF multzo bat egiten dugu.

## NMI

- Ez-Maskable Eten (NMI)
- PUZaren sarrerako seinalea da.
- NMI (PIC etenen antzeko hardware-eragindako eten bat da, baina NMI zuzenean CPUsa doa, eta ez PIC kontrolagailuaren bidez. Zerbitzu hau ezin da ezkutatu eta gainidatzi, beraz, kasu kritikoetarako erabiltzen da.
- Aplikazioak
  - ÿ Watchdog temporizadorea
    - ÿ Aldian-aldian berrezarri behar den temporizadorea da. PUZa blokeatuta badago, ezin izango du temporizadorea berrezarri eta NMI eten bat sortuko du, beraz, programaren kontagailua NMI etenaldien arreta errutina seinalatzen duen helbide batekin kargatuko da, zeinaren exekuzioak PUZaren egoera desblokeatuko duen. .

## Intel

- Intel
  - ÿ Etenen 1. belaunaldia (XT-PIC): 15 eten bakarrik onartzen ditu.
  - ÿ 2. belaunaldia (IO-APIC): onartzen diren eten kopurua 24 arte.
    - ÿ **APIC** : Eten-kontrola programagarri aurreratua: arkitektura modernoetan erabiltzen da anitzeko nukleoa.
  - ÿ 3. belaunaldia, MSI: erabilgarri dauden eten kopurua 224ra arte.

### 8.9.4. Etenaldiaren kudeatzailea

- Interrupt Manager sistema eragileak implementatzen du: sarrera.S Linux 2.x bertsioan
- Etenaren eskatzailea identifikatzen du etenaldi horri erantzuteko berariazko errutina gauzatu ahal izateko. etenaldia.
- Lehentasunezkoak ez diren beste gailuek eteteko aukera bertan behera uzten du, IF banderaren bidez , -ren 9. bit CPU rflags kontrolatzeko erregistroa.
- Konsultatu Eten Bektorea (Etenaldien arreta-errutinen erakusleen taula).
- Eten egingo den martxan dagoen programaren ingurunea gordetzen du.
- Eten zerbitzuen errutina (ISR) aktibatzen du.
  - ÿ Baimendu berriro etenaldiak
  - ÿ Errutina hau nukleoaren kontrolatzailearen moduluan ezarriko da.

### 8.9.5. Eten motak

- Intelek bi Eten Seinale mota definitzen ditu
  - ÿ **Sinkronikoa**
    - ÿ PUZak berak sortzen ditu instrukzio-zikloaren amaieran, gertaera bati erantzuteko. Instrukzio-zikloak, beraz, beste fase bat du: etenaldi-fasea
    - ÿ Salbuespenak deitzen dira
      - ÿ Software-etenaldia: x86-64 arkitekturako **syscall** instrukzioak sortua eta

int instrukzioa \$0x80 x86-32 arkitekturan: nukleoa eteten da.  
Sistema Deiak.

ÿ Errore batek sortua: Matxura, Tranpa, Abortua, Segmentu-urraketa.

ÿ Asinkronoak

ÿ CPUa ez den periferikoek edo hardwareek sortua

ÿ Etenaldiak edo hardware-etenaldiak deitzen dira.

ÿ Maskable: IRQ ÿ nahi ez dugun denbora tartean desgaitu daiteke  
eten egiten gaituzte.

ÿ Maskagarria: NMI ÿ ezin da desgaitu.

#### 8.9.6. Eten-bektoreen taula

Modu erreala: IVT taula

- OS duten plataformetan ordenagailua piztean (abio-kargagailuarekin abiaraztean) CPUa funtzionatzen ari da hasieran modu errealean eta OSrik gabeko plataformetan (abiarazte BIOSarekin) CPUak etengabe funtzionatzen du Modu errealean. OS duten plataformetan, abioa modu errealean hasten da eta ordenagailua konfiguratuta dago babestutako moduan sartzeko, sistema eragilea memoria nagusian kargatu aurretik.

ÿ Modu erreala:

ÿ 16 biteko modu sinplista da, x86 prozesadore guzietan dagoena:  
8086 CPU zaharra.

ÿ 8086 CPUak 20 helbide-bit eta 16 datu-bit ditu.

ÿ Modu errealeko erakuslea 16 biteko segmentu-helbide gisa definitzen da eta 16 biteko desplazamendua segmentu horretan.

ÿ Segmentua 20 bitetara zabaltzen da x4 biderkatuz.

ÿ 2 20: Kodeak RAM memoriako lehen Megaaon egon behar du

- BIOS funtziotara sartzeko aukera ematen du.

ÿ BIOS eten taula

ÿ PC teknologia

MOV AH, 0Eh ; Inprimatu karakterea pantailan

MOV AL, '!' ; inprimatu beharreko karakterea

INT 10h ; BIOS bideo funtzioei deitzea

- IVT taula.

Eten helbide mota	Deskribapena
00h	0000:0000h Prozesadorea Zatitu zeroz
01h	0000:0004h Prozesadorea Urrats bakarra
02h	0000:0008h Prozesadorea Eten ez maskagarria (NMI)
03h	0000:000Ch prozesadorearen eten-puntua
04h	0000:0010h Prozesadorea Gaintze aritmetikoa
05h	0000:0014h Softwarea Pantaila inprimatu
06h	0000:0018h Prozesadorea Operazio-kode baliogabea
07h	0000:001Ch Prozesadorea Koprozesadorea ez dago erabilgarri
08h	0000:0020h Hardwarea Sistemaren temporizadorearen zerbitzu errutina

09h	0000:0024h	Hardwarea	Teklatu-gailuaren zerbitzu-errutina	
0Ah	0000:0028h	Hardwarea	2. etendura kontrolagailu programagarritik kaskada	
0Bh	0000:002Ch	Hardwarea	Serieko ataka zerbitzua - COM post 2	
0 Ch	0000:0030h	Hardwarea	Serieko ataka zerbitzua - COM ataka 1	
0Dh	0000:0034h	Hardwarea	Inprimagailu paraleloen zerbitzua - LPT 2	
0eh	0000:0038h	Hardwarea	diskete zerbitzua	
0Fh	0000:003Ch	Hardwarea	Inprimagailu paraleloen zerbitzua - LPT 1	
10h	0000:0040h	Softwarea	Bideo-zerbitzuaren errutina	
11h	0000:0044h	Softwarea	Ekipoen zerrrenda zerbitzuaren errutina	
12h	0000:0048H	Softwarea	Memoria tamaina zerbitzuaren errutina	
13:00	0000:004Ch	Softwarea	Disko gogorreko disco zerbitzua	
14:00	0000:0050h	Softwarea	Serieko komunikazio zerbitzuen errutinak	
15:00	0000:0054h	Softwarea	Sistema-zerbitzuek errutinak onartzen dituzte	
16:00elan	0000:0058h	Softwarea	Teklatuaren laguntza zerbitzuaren errutinak	
17:00elan	0000:005Ch	Softwarea	Inprimagailu paraleloen laguntza-zerbitzuak	
18:00elan	0000:0060h	Softwarea	Kargatu eta exekutatu ROM BASIC	
19:00elan.	0000:0064h	Softwarea	DOS kargatzeko errutina	
1Ah	0000:0068h	Softwarea	<b>Denbora</b> errealako erlojuaren zerbitzu errutinak	
1Bh	0000:006Ch	Softwarea	CRTL - BREAK zerbitzu errutinak	
1 Ch	0000:0070h	Softwarea	Erabiltzaileen temporizadorearen zerbitzu errutina	
1Dh	00000074h	Softwarea	Bideo-kontroleko parametroen taula	
1Eh 0000:0078h	Softwarea		Disketearen parametroen errutina	
1Fh	0000:007Ch	Softwarea	Bideo grafikoaren karaktereen errutina	
20h-3Fh 0000:0080f-0000:00FCh	SW		BI eten puntu	
40h 0000:0100h	Softwarea		Diskete hargailuen errutina	
41h	0000:0104h	Softwarea	C disko gogorra: parametroen taula	
42h	0000:0108h	Softwarea	EGA bideo kontrolatziale lehenetsia	
43h	0000:010Ch	Softwarea	Bideo grafikoaren pertsonaiak	
44h	0000:0110h	Softwarea	Netware API berria	
45h	0000:0114h	Softwarea	Ez da erabiltzen	
46h	0000:0118h	Softwarea	D disko gogorra: parametroen taula	
47h	0000:011 Ch -	Softwarea	Ez da erabiltzen	
48h	Softwarea		Ez da erabiltzen	
49h	0000:0124h	Softwarea	Ez da erabiltzen	
4Ah	0000:0128h	Softwarea	Erabiltzaileen alarma	
4Bh-63h	0000:012 Ch -	Softwarea	Ez da erabiltzen	
64 ordu	Softwarea		Netware IPX berria	
65h-66h	Softwarea		Ez da erabiltzen	
67h	Softwarea		EMS laguntza errutinak	
68h-6Fh	0000:01BCh	Softwarea	Ez da erabiltzen	
70h	0000:01C0h	Hardwarea	<b>denbora</b> errealako erlojua	
71h	0000:01C4h	Hardwarea	Birbideratu eten kaskada	
72h-74h	0000:01C8h - 0000:01D0h		Hardwarea	Erreserbatuta - Ez erabili
75h	0000:01D4h	Hardwarea	Matematika koprozesadorearen salbuespena	
76h	0000:01D8h	Hardwarea	Disko gogorreko euskarria	
77h	0000:01DCh	Hardwarea	Eten eskaera	
78h-79h	0000:01E0h -	Hardwarea	Ez da erabiltzen	
7Ah	Softwarea		Novell Netware APIa	
78h-FFh	0000:03FCh	Softwarea	Ez da erabiltzen	

ÿ Taularen edukia Intel CPUaren sorreraren araberakoa da

ÿ Lehen zutabea: Etenaren bektore-zenbakia. Taula bektorialeko sarreraren zenbakia.

ÿ Bigarren zutabea: etenaldi-bektore-zenbakien taulako desplazamendua

ÿ X zutabea: taulan falta da.

ÿ 4 byteko bektorea: **ISR etenaldiaren arreta errutinaren erakuslea da**

ÿ Norabidea segmentatzen da. Segmentua: Desplazamendua. Bi byte segmentuarentzat eta beste bi  
desplazamendu

ÿ Helbidea:

ÿ IDTR Erregistroak taulako lehen sarrerara seinalatzen du.

ÿ IVT taula normalean 0000:0000H-n kokatzen da, eta 400 H byte-ko tamaina du (**4 byte bakoitzeko 265 etenaldiren etenaldia**).

ÿ Helbide erlatiboa eten-zenbakia biderkatuz lor dezakegula ikusten dugu  
x4.

ÿ 9. bektorea IVT 36 desplazamenduari dagokio, hau da, 0x24 ÿ 0000:0024h forma segmentatuan

ÿ Azken sarreraren desplazamendua = 4 x 0xFF = 0x400-4 =0x3FC izango da

- Eten motak

ÿ Lehenengo 32 bektoreak prozesadorearen barne salbuespenetarako erreserbatuta daude (0x00-0x1F)

ÿ 0x20-0xFF etenaldiak hardware IRQ etenak dira.

ÿ ARGAZKIA

ÿ PIC kontroladoreak IRQ seinalea taulako sarrera-bektore batekin mapatzeaz arduratzen da.

ÿ IRQ0 periferikoa ÿ PIC bektorea 0x20 ÿ IVT taula erakuslea 0000:0080f (RAM) ÿ ISR funtziora deitu IRQ0 periferikoa (RAM) artatzeko

### Babestutako modua: IDT taula

- Sistema eragilea duten plataformetan, eragiketak modu errealean amaitutakoan, abio-kargatzaireak sistema eragilea kargatzen amaitzen du eta CPUa modu babestuan konfiguratzen da, erabiltzaileak ezin izango du: OS moduluak exekutatu, hala nola kontrolatzaireak, memoriaren edozein eskualdetan sartu. fisika, erregistro pribilegiatuak, argibide pribilegiatuak,...
- OSak IDT etenen deskribapen-taula konfiguratzen du IVTren funtzio berdinarekin baina eduki desberdinak.
- [Eten deskribatzaireen taula IDT](#)

RTD Offset	INT #	Deskribapena
0x0000	0x00	Zatitu 0z
0x0004	0x01	Erreserbatuta
0x0008	0x02	NMI Etenaldia
0x000C	0x03	Eten-puntua (INT3)
0x0010	0x04	Gainezka (INTO)
0x0014	0x05	Mugen barrutia gainditu da (BOUND)
0x0018	0x06	Opcode baliogabea (UD2)
0x001C	0x07	Gailua ez dago erabilgarri (WAIT/FWAIT)
0x0020	0x08	Errua bikoitza
0x0024	0x09	Koprozesadorearen segmentua gainditzea
0x0028	0x0A	TSS baliogabea
0x002C	0x0B	Ez dago segmentua
0x0030	0x0C	Pila-segmentu-matxura
0x00030	0x0D	Babes orokorren akatsa
0x00030	0x0E	Orriaren akatsa
0x0003	0x0F	Erreserbatutako
0044	0x10	x87 FPU errorea
0x0048	0x11	Lerrokadura egiaztatzea
0x004C	0x12	Makina egiaztatzea
0x00xx	0x10	SIMD Koma Mugikorreko Salbuespena
0x0xxx	0x12 0x01	Erreserbatuta
0x12 0x01 0x12 0x01	0x01	Erabiltzaileak defini daiteke
		→ IRQ, SW eten egiten du

ÿ Edukia OS kernelaren araberakoa da

ÿ Lehen zutabea: desplazamendua ISR etenaldi alerta errutinarekin

ÿ Bigarren zutabea: eten bektore-zenbakia.

ÿ eten motak

ÿ 0-0x1F: ERROREA eta NMI salbuespenak

ÿ 0x20-0x2F: INT maskagarria: IRQ0-----IRQ15

ÿ 0x30-0xFF: SW salbuespenak

ÿ 0x80 ÿ SISTEMA DEIAK

ÿ isa x86-64: syscall

ÿ isa x86: int \$0x80

ÿ Zein errutina adierazten du 0x0E bektoreak? ÿ Orriaren akatsa

ÿ Sarreren deskribapena

ÿ IDTR: taulako lehen sarrera seinalatzen duen erregistroa

ÿ Sarrera bakoitzak 8 byte ditu Intelek ate deitzen dituena.

ÿ Segmentu-hautatzaile bat dauka, segmentu-deskribatzailea identifikatzen duena  
segmentu-deskribatzaileak (ikusi intel segmentazioa)

## IRQ

- XT-PIC etenek Intel 8259 eten programagarrien kontrolagailu (PIC) pare bat erabiltzen dute.

ÿ Linux kernelak konfiguraturako PIC : [Understanding the Linux Kernel, Marco Cesati, Daniel P. Bovet -en bigarren edizioa](#)

ÿ Adibidea XT-PIC IRQ esleipena, [intelek papera eten egiten du](#): Konfigurazio hau adibide bat da, hau da, OSak **birprogramatu** dezake eta bere konfigurazioa alda dezake.

IRQ      Eten hardware-gailua (taula bektoriala)

- 0 32 Temporizadore
- 1 33 Teklatua
- 2 34 PIC Kaskada
- 3 35 segundo serieko ataka (COM2)
- 4 36 Lehen serieko ataka (COM 1)
- 5 37 <Doan>
- 6 38 Disketea
- 7 39 <Doan>
- 8 40 Sistemako erlojua
- 9 41 <Doan>
- 10 42 Sareko Interfaze Txartela (NIC)
- 11 43 USB ataka eta soinu txartela
- 12 44 Sagua (PS2)
- 13 45 Matematika Koprozesadorea
- 14 46 IDE kanala 1
- 15 47 IDE kanala 2

Oharra: Linux\*ek IRQ 0, 2 eta 13 behar ditu erakusten den moduan egotea.

ÿ Master 8259 (PC bateragarria)

IVT Offset	INT #	IRQ#	Deskribapena
0x0020	0x08	0	PIT
0x0024	0x09		Teklatua
0x0028	0x0A	1	8259A kontroladore esklaboa
0x002C	0x0B	2	COM2/COM4
0x0030	0x0C	3	COM1/COM3
0x0034	0x0D	4	LPT2
0x0038	0x0E	5	Diskete kontrolatzalea
0x003C	0x0F	6 7	LPT1

ÿ Bigarren zutabea: PIC-ko etenaren zenbakia

ÿ Hirugarren zutabea: IRQ etenaren zenbakia

ÿ Lehen zutabea: sarrera horren desplazamendua lehenengo sarrerarekin alderatuta. Bektore-zenbakia.

ÿ Esklaboa 8259

IVT Offset	INT #	IRQ#	Deskribapena
0x01C0	0x70	8	RTC
0x01C4	0x71	9	Esleitu gabe
0x01C8	0x72	10	Esleitu gabe
0x01CC	0x73	11	Esleitu gabe
0x01D0	0x74	12	saguaren kontrolagailua
0x01D4	0x75	13	Matematika koprozesadorea
0x01D8	0x76	14	Disko gogorreko kontrolagailua 1
0x01DC	0x77	15	Disko gogorreko kontrolagailua 2

ÿ Bigarren zutabea: PIC-ko etenaren zenbakia

ÿ Hirugarren zutabea: IRQ etenaren zenbakia

↳ Lehen zutabea: sarrera horren desplazamendua lehenengo sarrerarekin alderatuta. Bektore-zenbakia.

## Linux

- Nukleoak konfiguratutako etenaldiak: cat /proc/interrupts

## Lerro partekatuak

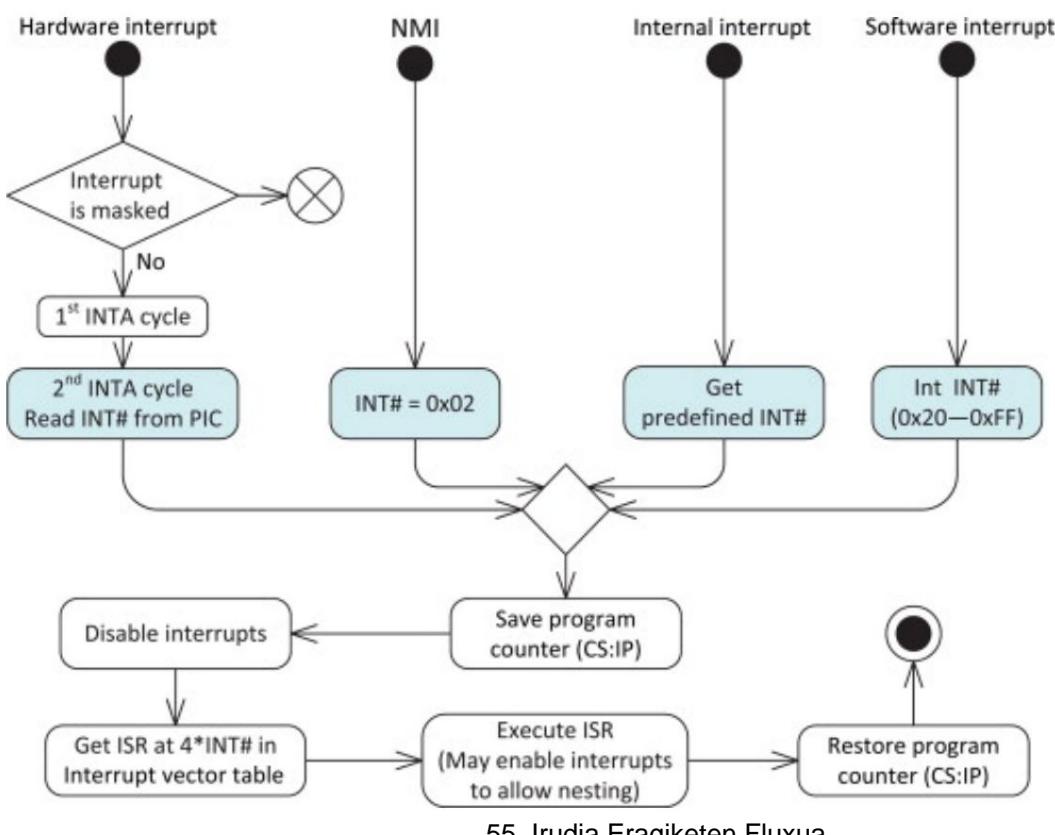
- [https://nptel.ac.in/courses/Webcourse-contents/IIT-%20Guwahati/comp\\_org\\_arc/web/module06\\_io/lect\\_03\\_intr/lect\\_03.htm](https://nptel.ac.in/courses/Webcourse-contents/IIT-%20Guwahati/comp_org_arc/web/module06_io/lect_03_intr/lect_03.htm)
- Eten-lerro bat hainbat gailuk partekatzen badute, gailuetako batek eten-seinalea linea arruntera bidaltzen duenean, CPUak hainbat modutara identifikatu dezake etenaldia eskatzen duen gailua:

↳ lineako kide bakoitzari galdeketa softwarea

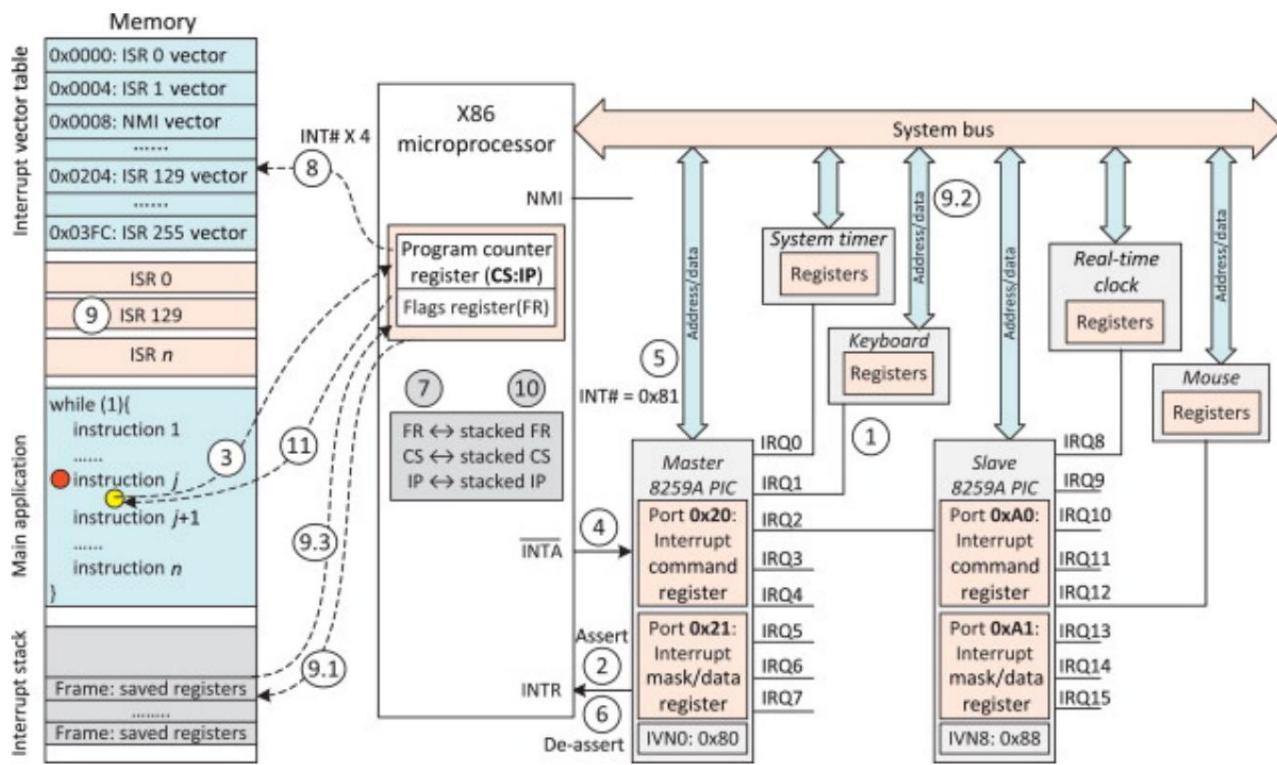
↳ PUZaren esleipen-lerroa margarita-kate moduan jarritako gailuei (banaka-banaka ematen diente diru-laguntza), diru-laguntza eskaera egiten duen kidearengana iristen denean, bere identifikazioa itzultzen du.

### 8.9.7. Etenaldiaren arreta prozesua

#### Eragiketen fluxua



Adibidea: teklatuak eragindako etenaren prozesamendua.



56. Irudia Teklatuaren Etenaldia

1. Tekla bat sakatzen da eta teklatuaren kontrolatzaleak PIC-tik IRQ1 seinalea aktibatzen du.
2. PIC-k aztertzen du IRQ1 eteteko baimenik dagoen ala ez eta lehentasun handiagoko etenik ez dagoen. Kasu honetan, PICak PUZaren INTR sarrera aktibatzen du.
3. PUZak main\_application prozesuaren exekuzioa eteten du (eten egiten du).
4. CPUak egiaztatzen du IF banderak etenaldia baimentzen duela, PIC-ri etenaldia onartzen duela berresten dio eta IRQ1-ekin lotutako eten-bektorea eskatzen du.
5. PICak 129 (0x81) bektorea bidaltzen du helbide-busera.
6. PICak INTR eten eskaera desgaitzen du CPUra, beste IRQ bat kudeatzeko.
7. CPUak testuingurua gordetzen du (Bandera-erregistroa, Kode-segmentua, Instrukzio-erakuslea) Eten-pilean.
8. CPUak IF bandera (IF=0) berrezartzen du, ISR errutinaren testuinguruak eteten duen prozesuaren testuinguru-aldaketan etetea baimendu gabe. Eten-bektore-taulatik 129 eten-bektorea harrapatzen du ý taulatik ateratzen du 129x4 ý 0x81\*4=0x204 sarrera, teklatu periferikoarekin lotutako Eten Zerbitzu Errutinaren ISR helbidea duen.
9. CPUak CS:IP (0x206:0x204) eguneratzen du 129 ISR helbidera salto egiteko eta exekuzioari ekiten dio.
  - to. ISRk "erregistro aldaezinak" gordetzen ditu Eten pila ý Markoa: gordetako erregistroak. Baimena eman etenaldiak IF=1 (Etenaldiak habiatzeko aukera)
  - ISRen amaiera. Etenaldiak (IF=0) ez dira onartzen aurreko testuingurua berreskuratzean.
  - CPUak "erregistro aldaezinak" berreskuratzen ditu Eten pilatik ý Markoa: gordetako erregistroak
10. Aurreko testuingurua berreskuratzen da eta, beraz, itzulera helbidea ere kontagailua eguneratuz. CS:IP programa
11. Eten dagoen programa main\_application exekutatzen jarraitzen du

## 8.10. DMA memoriarako sarbide zuzena

#### 8.10.1. Funtzionalitatea

- Datu-transferentziak egin, horrela CPUa askatuz
- Aplikazioa: disco gogor eta memoria nagusiaren arteko datuak transferitzen dira
- Unitatea: DMAC (DMA kontroladorea)
  - ÿ Hainbat DMA kanal izan ditzakezu: kanal bakoitzak periferiko baten transferentzia kudeatzen du.

#### 8.10.2. Transferentziak

- Burst modua
  - ÿ DMAC-ek sistema-busaren kontrola hartzen duenean, ez du uko egiten bloke osoa osatu da
  - ÿ Sistema-busa DMAC-ek okupatzen duen bitartean, CPUak cache memoriarekin funtziona dezake.
- Zikloaren lapurreta modua
  - ÿ DMAC-ek sistema-busaren kontrola CPUrak itzultzen dio hitz bat transferitzen duen bakoitzean.
  - ÿ Autobusa denboran partekatzen da: erabilgarria denbora errealeko sistema kritikoetan
- Modu gardena
  - ÿ DMAC-k bakarrik hartzen du autobusa libre dagoenean eta CPUak behar ez duenean.

#### 8.10.3. Sinkronizazioa

- PUZak DMA eragiketa bat has dezake irakurtzeko edo idazteko bus-zikloaren mugetan. Beraz, DMA eragiketa bat abiarazi daiteke instrukzio-zikloan zehar.

#### 8.10.4. DMA kontrolagailuaren funtzionamendua

##### Maila handiko urratsen sekuentzia

- Prozesu batek irakurketa-deia egiten duenean, gidariak memoria nagusiko eskualde bat (DMA buffer) esleitzen dio eta hw seinaleak sortzen ditu datuak DMA bufferera transferitzeko eskatzeko. Prozesua lo egoeran jarraitzen du.
- DMAk datuak DMA bufferera transferitzen ditu eta eten-seinalea abiarazten du amaitzean.
- Eten-kudeatzaileak datuak bufferretik azken kokapenera kokatzen ditu, zerbitzu-etenaldi baten berri ematen du eta prozesua esnatzen du, eta orain memoria nagusiko datuak irakur ditzake.

##### Maila baxuko urrats sekuentzia

- Programatzeko hiru parametro:
  - ÿ Transferitu beharreko datu-blokearen MPincipal hasierako helbidea: AR
  - ÿ Transferitu beharreko datu kopurua: Komuna
  - ÿ Transferentzia modua
- Urratsak
  - to. Ordenagailuaren abiaraztean CPUak DMA hasieratzen du parametroak programatuz.
  - b. Kontrolagailu periferikoak zure zerbitzuak eskatzen ditu.
  - c. Periferikoak DMA eskaera bat egiten dio DMAC-ri (DMA Controller): DMA eskaera.
  - d. DMA-ek onarpen-seinale batekin erantzuten du

eta. DMAk DMA eskaera-lerroa aktibatzen du CPUrak: Bus Request

- F. Uneko autobus-zikloaren amaieran, prozesadoreak sistemako bus-lineak impedantzia handian ezartzen ditu eta DMA saioa aktibatzen du: Bus Grant
- g. DMAC sistema-busaren kontrola bere gain hartzen du
- h. I/O gailuak datu-hitz berri bat transmititzen du DMAC-en bitarteko datu-erregistrora (DMACen buffer txiki bat)

Yo. DMAcak memoria idazteko ziklo bat exekutatzen du tarteko erregistroaren edukia transferitzeko M[AR] posiziora.

- j. DMAcak WC gutxitzen du eta AR handitzen du.
- k. DMAC-k autobusa askatu eta DMA eskaera-lerroa desaktibatu egiten du.
- l. DMAC-ek WC 0rekin alderatzen du:
- m. WC > 0 bada, errepikatu 2. urratsetik.
- n. WC = 0 bada, DMAC gelditu egiten da eta eten eskaera bat bidaltzen dio prozesadoreari.

### 8.10.5. Cachearen koherentzia arazoak

- DMA kontrolatzailak, periferikoaren eta memoria nagusiaren artean datuak transferitzean, cache-memoriaren lerroak memoria nagusien blokeen kopiak ez izatea eragiten du. Cache kontrolatzailak cachea egunerau beharko du DMA eragiketa baten ondoren.

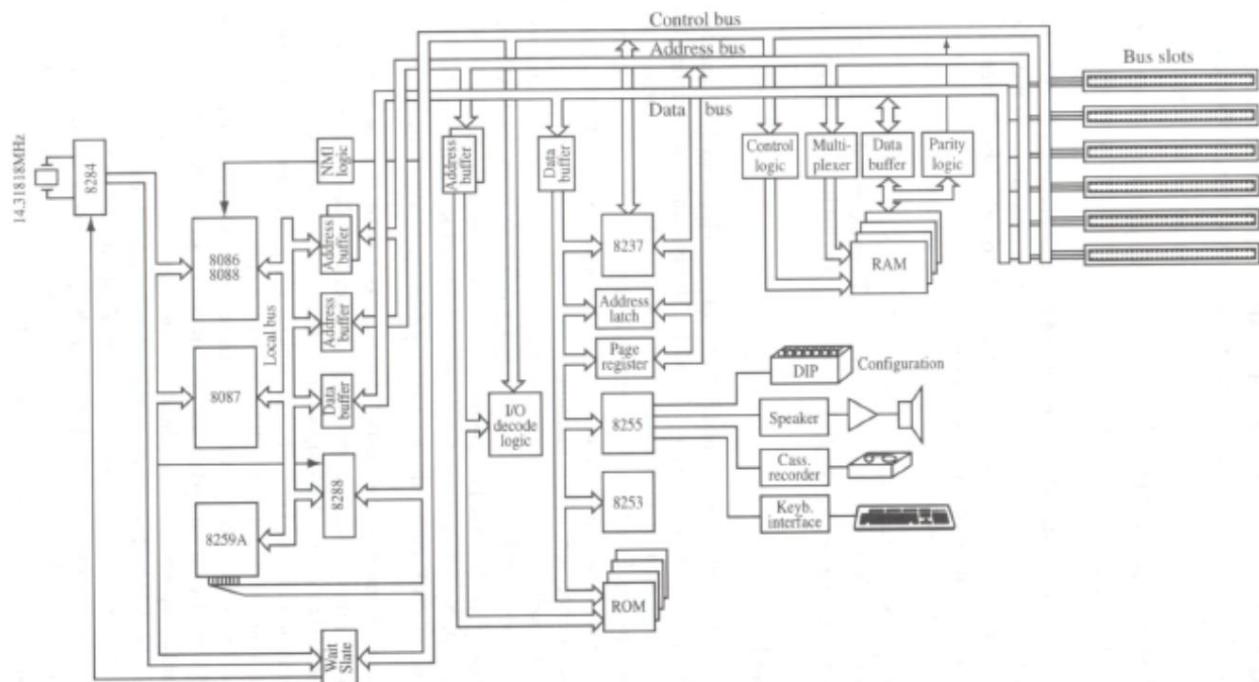
## 8.11. Autobusak

- I/o arkitektura bi norabidetan eboluzionatzen ari da
  - ÿ autobusaren banda zabalera handitza
  - ÿ I/O kontrolagailuak txip bakarrean integratzea

### 8.11.1. DA

# IBM PC/XT Architecture ('82, '83)

(XT is “extended” PC – 4.77 MHz Bus)



57. Irudia ISA Bus Arkitektura

# ISA (8-bit Version)

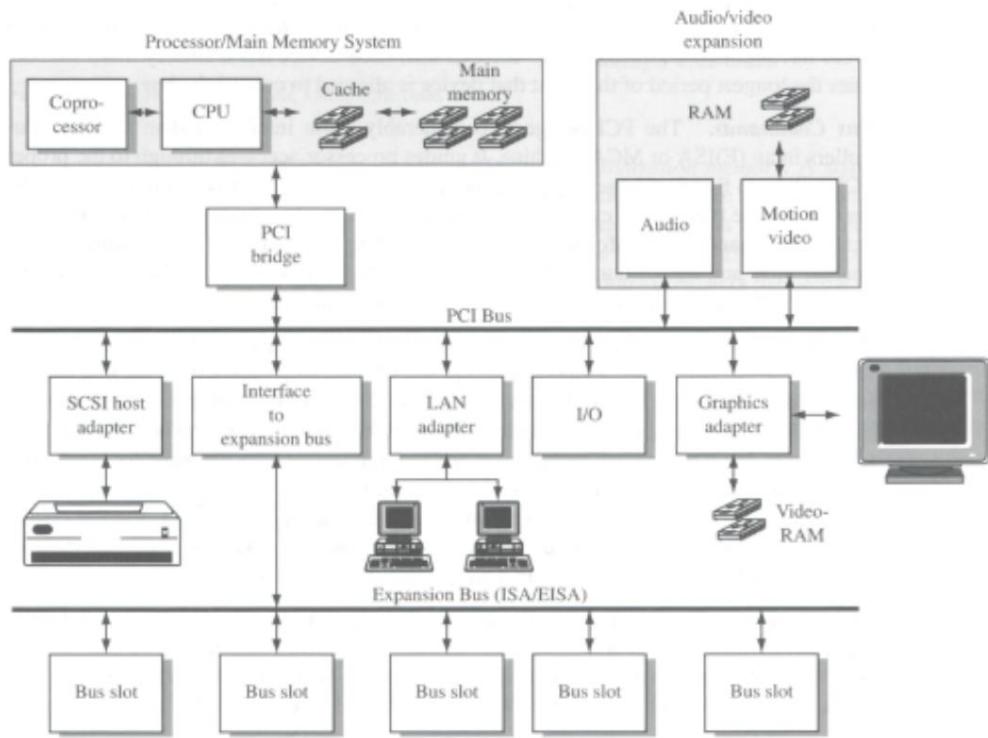
*(Typical 4.77 MHz Bus – IBM PC, IBM XT)*

	Signal	Name	Type*	Description
GND	A0–A19	Address Lines	Output	20-bit address bus
PRESET DRV	D7	AEN	Output	High when DMA controller is controlling the buses.
+5V	D6			
IRQ2	D5	ALE	Output	High when valid address signals are on the bus.
-5V	D4	CLK	Output	In the original PC this was 4.77 MHz
DRQ2	D3	D0–D7	Input/Output	8-bit data bus
-12V	D2	DACK0–DACK3	Output	When low these signals acknowledge a peripheral's DMA request.
res	D1	DRQ1–DRQ3	Input	High to request a DMA transfer. DRQ0 is dedicated to memory refresh and is therefore not available on the bus.
+12V	D0	I/O CH CK	Input	Low to indicate an error condition and generate an NMI.
GND	B10 A10	AEN		
MEMW	A19	A18		
MEMR	A18	DRQ1–DRQ3		
IOW	A17	I/O CH RDY		
TOR	A16	I/O CH CK		
DACK3	A15	IOW		
DRQ3	A14	IRQ2–IRQ7		
DACK1	A13	MEMR		
DRQ1	A12	MEMW		
DACK0	A11	OSC		
CLK	A10	RESET DRV		
IRQ7	A9	T/C		
IRQ6	A8			
IRQ5	A7			
IRQ4	A6			
IRQ3	A5			
DACK2	A4			
TC	A3			
ALE	A2			
+5V	A1			
OSC	A0			
GND	B31 A31			

\*Input to or output from the processor/bus controller.

## 58. Irudia ISA Bus Interfazea

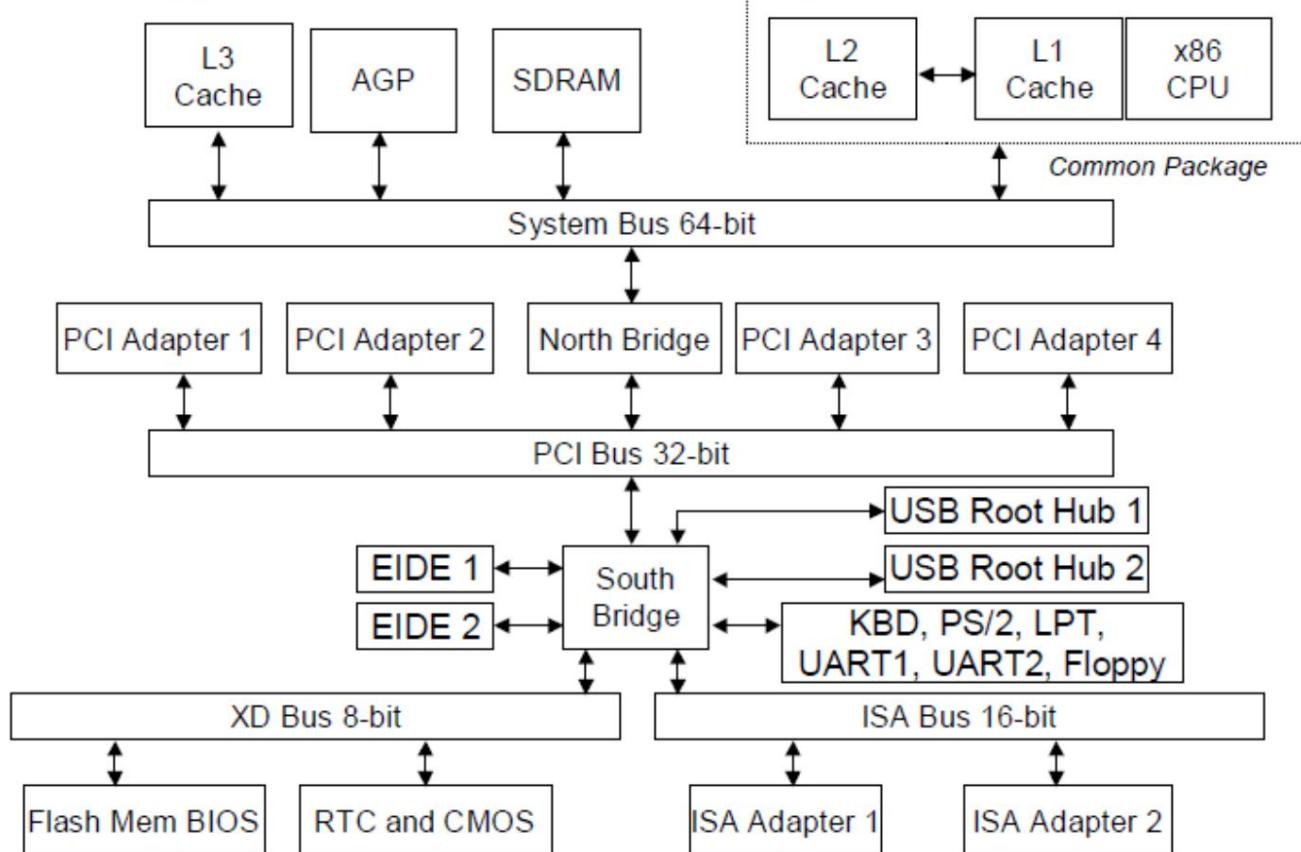
### 8.11.2. PCI



59. Irudia. PCI Bus Arkitektura

## 8.11.3. Ipar-Hego zubia

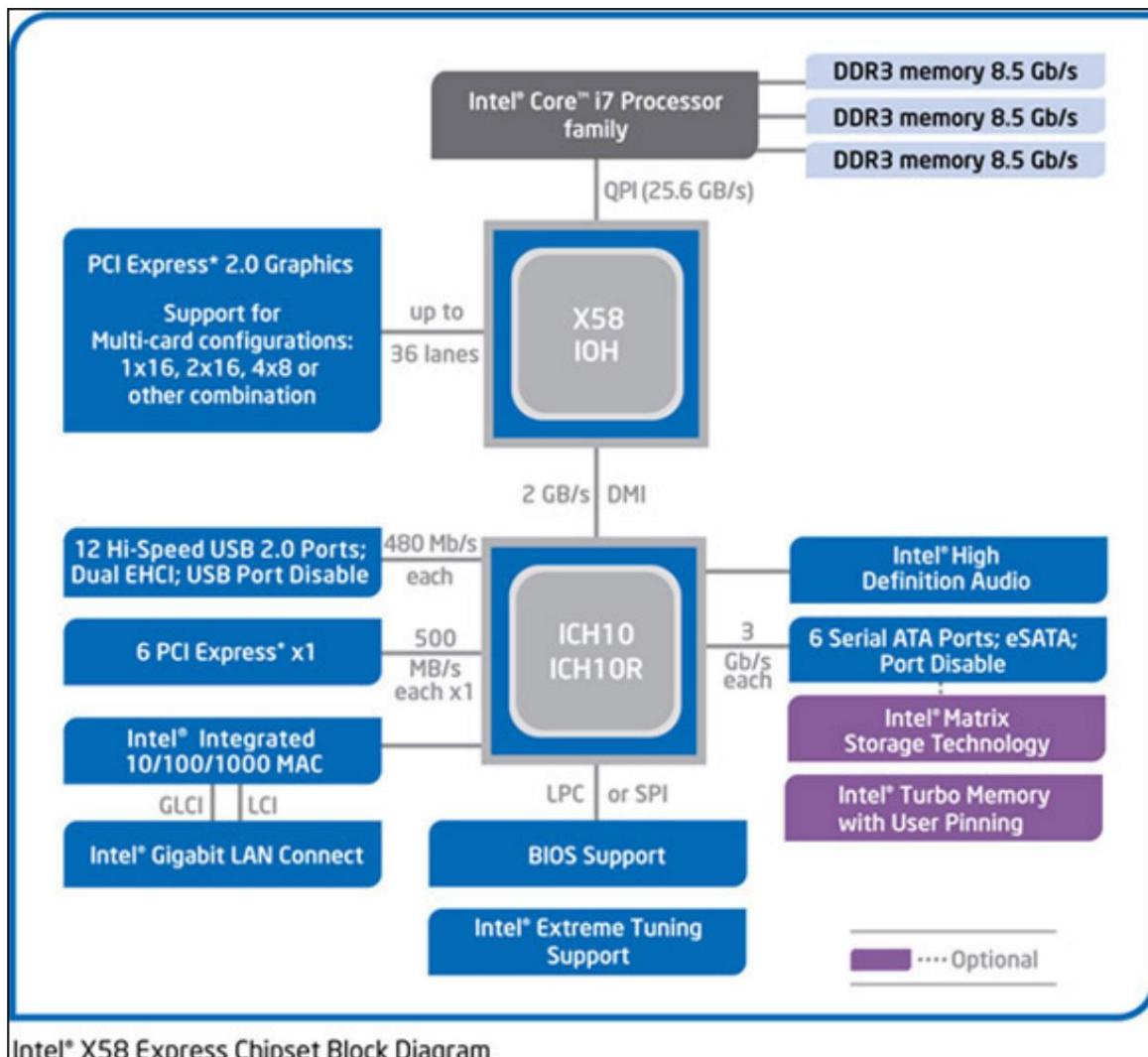
## Typical PCI Based x86 Computer Architecture



60. Irudia Ipar eta Hegualdeko zubiak

#### 8.11.4. x58 chipset

- [https://en.wikipedia.org/wiki/Intel\\_X58](https://en.wikipedia.org/wiki/Intel_X58)
- PCH: Platform Controller Hub
- FSB: Front Side Bus
- BSB: Back Side Bus
- FDI: Pantailarako Interfaze Malgua (kontrolagailu grafikoa integratzen duten CPUetarako)
- DMI: Zuzeneko Media Interfazea
- ICH: i/o Controller Hub
- IOH: i/o Hub



61. irudia corei7 x58 chipset: 2008. urtea

LGA 1366, LGA 2011 eta LGA 2011-v3 CPUak onartzen dituzten chipsetak: X58 (2008), X79 (2011), X99 (2014).

#### 8.12. Sarrera/irteera errutinak programatzea

##### 8.12.1. Sistema eragilearen software hierarkikoa

- Gidaria edo sw kontrolatzalea sw egituraren maila baxuena da: hardwarearen araberakoa da. ordenagailua: C edo mihiztatzale lengoaiaren programatzea.

### 8.12.2. Argibide multzoen arkitektura

- I/O sarbidea

ÿ OUTx : byte bat (edo hitza edo dword) bidaltzen du I/O kokapen batean. Izen tradizionalak outb, outw eta outl dira hurrenez hurren. "a" modifikatzaileak val eax erregistroan jartzea behartzen du asm komandoa igorri baino lehen eta "Nd"-ek byte bateko balio konstanteak konstante gisa muntatzea ahalbidetzen du, edx erregistroa beste kasuetarako askatzu.

```
static inline void
outb( sinatu gabeko ataka laburra, sinatu gabeko char val ) { asm
    volatile( "outb %0, %1"
              : : "a"(val), "Nd"(port) );
}
```

ÿ C iturburu-programak ASM lengoia du: C iturburu-programa inline-asm-ekin.

ÿ %0 "a" lehen aldagaiari dagokio, %i igarren aldagaiari.

ÿ INx : byte bat (edo hitza edo dword) jasotzen du I/O kokapen batetik. Izen tradizionalak inb, inw eta inl dira hurrenez hurren.

```
static inline
unsigned char inb( unsigned short port ) { unsigned
    char ret; asm volatile( "inb
                           %1, %0"
                           : "=a"(ret): "Nd"(ataka) );
    itzuli ret; }
```

- Erregistroko I/O argibideek IN (I/O atakatik sarrera) eta OUT (I/O atakarako irteera) datuak mugitzen dituzte I/O ataka eta EAX erregistroa (32 biteko I/O), AX erregistroa ( 16 biteko I/O), edo AL (8 biteko I/O) erregistroa. I/O atakaren helbidea berehalako balio batekin edo DX erregistroko balio batekin eman daiteke.

#### Intel eskuliburua

- Instrukzio hau prozesadorearen I/O helbide-espazioan dauden I/O portuetara sartzeko bakarrik da erabilgarria. Ikus Intel® 64 eta IA-32 Architectures Software Garatzaileen Eskuliburuaren, 1. liburukian, "Sarrera/Irteera" kapitulua, 16 edo 14. I/O portuetara sartzeari buruzko informazio gehiago lortzeko, I/O helbide-espazioan.
- I/O atakak mapatu daitezke, I/O helbide-espazioan edo memoria fisikoko helbidean ager daitezen. espazioa (memoriaren mapatutako I/O) edo biak.
- memoria-mapatua:
  - ÿ kontrol-bus linea baten bidez, helbidea memoria nagusia edo i/o ataka den zehazten da, prozesadore batzuetan M/IO# pinaren bidez.
  - ÿ Memorian mapatutako I/O erabiltzean, I/O eragiketetarako mapatutako helbide-espazioaren cachean gorde behar da. galarazi
- I/O mapatua
  - ÿ I/O gailuek ez dute memoriarekin talka egiten, beste helbide-espazio bat erabiltzen baitute, helbideetan (atakuetan) balioak irakurtzeko eta idazteko argibide ezberdinak. CPU memoria-I/O bus transakzio-argibideak deskodetu I/O atakak hautatzeko Gailu hauek ezin dira zuzendu makina-kodeen argibideak erabiliz

memoria helburu duena. Gertatzen ari dena da bi seinale ezberdin daudela: MREQ eta IOREQ. The lehenengo memoriako instrukzio guztietan baieztatzen da, bigarrena, I/O instrukzio guztietan. Beraz, hau kodea...

**MOV DX,1234h**

MOV AL,[DX]	;memoriaren helbidea irakurtzen <b>du 1234h</b> (memoriaren helbide-espazioa)
AL,DX	;I/O ataka <b>1234h</b> irakurtzen du (I/O helbide-espazioa)

ÿ 1234h atakan dagoen I/O gailua sistema-busera konektatuta dago, beraz, helbidea helbidea bada soilik gaituta egon dadin. 1234h, RD (Irakurri datuak) aldarrikatzen da eta IOREQ aldarrikatzen da.

ÿ (64K) banaka zuzendu daitezkeen 8 biteko I/O atakak

- Babesa

ÿ Portu mapatua

ÿ Hemen, nukleoa eta gailu-kontrolatzaileek I/O egiteko baimena dute, pribilegiork gabeko gailu-gidariekin, berriz, eta aplikazio-programiei I/O helbide-espaziorako sarbidea ukatzen zaie. Aplikazio-programak behar dira ondoren, sistema eragileari deia egin I/O egiteko.

ÿ Memoria mapatua

ÿ Segmentazio arruntak eta orrien babesak i/o atakarako sarbideari eragiten diote.

### 8.12.3. Eten programagarrien kontrolagailuen programazioa

- irudien programazioa

ÿ PIC mapaketa

```
/* PIC kontrolagailuak gure bektoreekin birmapatu
8 + 70 baino lehen mapatutako moduan */

#definefinitu PIC1          0x20
#definefinitu PIC2          0xA0
#definefinitu PIC1_COMMAND    IRUDIA 1
#definefinitu PIC1_DATA (PIC1+1)
#definefinitu PIC2_COMMAND    PIC2
#definefinitu PIC2_DATA (PIC2+1)
#definefinitu PIC_EOI 0x20

#definefinitu ICW1_ICW4 0x01      /* ICW4 (ez) beharrezko */
#definefinitu ICW1_SINGLE 0x02     /* Bakarra (kaskadaren) modua */
#definefinitu ICW1_INTERVAL4 0x04   /* Deiaren helbide-tartea 4 (8) */
#definefinitu ICW1_LEVEL 0x08      /* Maila abiarazitako (ertza) modua */
#definefinitu ICW1_INIT 0x10       /* Hasieratzea - beharrezko! */

#definefinitu ICW4_8086 0x01      /* 8086/88 (MCS-80/85) modua */
#definefinitu ICW4_AUTO 0x02       /* Auto (normala) EOI */
#definefinitu ICW4_BUF_SLAVE 0x08   /* Buffer modua/esklaboa */
#definefinitu ICW4_BUF_MASTER 0x0C  /* Buffer modua/master */
#definefinitu ICW4_SFNM 0x10       /* Erabat habiaratu berezia (ez) */

void remap_pics (int pic1, int pic2)
```

```
{
    UCHAR a1, a2;

    a1=inb(PIC1_DATUAK);
    a2=inb(PIC2_DATUAK);

    outb(PIC1_COMMAND, ICW1_INIT+ICW1_ICW4);
    io_itxaron();
    outb(PIC2_COMMAND, ICW1_INIT+ICW1_ICW4);
    io_itxaron();
    outb(PIC1_DATA, pic1);
    io_itxaron();
    outb(PIC2_DATA, pic2);
    io_itxaron();
    outb(PIC1_DATUAK, 4);
    io_itxaron();
    outb(PIC2_DATUAK, 2);
    io_itxaron();

    outb(PIC1_DATA, ICW4_8086);
    io_itxaron();
    outb(PIC2_DATA, ICW4_8086);
    io_itxaron();

    outb(PIC1_DATUAK, a1);
    outb(PIC2_DATUAK, a2); }
```

#### 8.12.4. Teklatuaren kontrolatzalea

- Erreparatu nola programatzen den, etenaldiak kudeatzeko mekanismoa kontuan hartuta.
- Bilatu nukleo soil baten iturburu-kodea.

#### 8.12.5. ataka paraleloa

##### Erabiltzaile espaziotik

- Beharrezkoa da sistema-deiak egitea, erabiltzaile-espaziotik ezin baitugu sartu zuzenean HWra
- [Oinarritzko ataka helbideak](#)
- Atzitu Linux-en ataka erabiltzailearen espaziotik

```
/* led_capslock.c: ataka I/O adibide oso simplea *

* Kode honek LED teklatuaren CAP aktibatzen du, ataka idazteko, eten bat,
* eta ataka irakurtzeko. Konpilatu `gcc -O2 -o led_capslock led_capslock.c'-rekin * eta
* exekutatu root gisa `sudo ./led_capslock'-rekin. */
```

**#include <stdio.h>**

```
#include <stdlib.h>
#include <unistd.h>
#include <sys/io.h>

#define BASEPORT 0x0060 /* teklatua */

int main()
{
    /* Lortu portuetarako sarbidea */
    if (ioperm(BASEPORT, 3, 1)) {pererror("ioperm"); irten(1);}

    printf("\n\t\t Portua -> erregistro-egoera: %d\n", inb(BASEPORT + 1));

    /* Ezarri portuaren datu-seinaleak (D0-7) baxuan (0) */
    outb(0xED, BASEPORT);

    /* Lo pixka bat (100 ms) */
    lo egin(1000);
    printf("\n\t\tPiztu BLOKE MAIUSC teklaren LED \n");
    outb(0x07, BASEPORT);

    lo egin(1000);

    /* Ez ditugu portuak gehiago behar */
    if (ioperm(BASEPORT, 3, 0)) {pererror("ioperm"); irten(1);}

    irten(0);
}
```

ÿ <http://tldp.org/HOWTO/IO-Port-Programming-2.html>

ÿ gizon ioperm

ÿ gizon inb

ÿ cat /proc/iports

- <http://opensourceforu.efytimes.com/2011/07/accessing-x86-specific-io-mapped-hardware-in-linux/>

## 8.12.6. RS-232 serieko komunikazioa

- [tutoretza](#)

ÿ Avr Atmega 8bit txartela ÿ Pej Arduino One

ÿ Bi kasu: i/o galdeketa etenek gidatutako i/o

- UART ataka (RS-232)

ÿ Konektore fisikoa

ÿ Bi terminalen arteko komunikazio erdi-duplexa: DTE (PC) eta DCE (Arduino)

ÿ Tx Rx seinaleak

ÿ Portuko erregistroak

ÿ Kontrola

ÿ Egoera

ÿ Rx eta Tx datuak: UDR

- Programazioa
  - ÿ Liburu-denda
  - ÿ Cross toolchain
  - ÿ Algoritmoa: Fluxu-diagrama
    - ÿ Bozketa eta etenaldi kasuak
  - ÿ Egitura modularra: bi modulu
  - ÿ Simboloak
    - ÿ I/O datuen bufferra
    - ÿ Eten-bektorearen izena

#### 8.13. Ariketak

- William Stalling testuliburuko 7. kapitulua.

## 9. kapitulua. Memoria Unitatea

### 9.1. Sarrera

#### 9.1.1. Ikastaroa

##### 9. Memoriaren antolaketa

- a. Memoriaren hierarkia
- b. Latentzia eta banda zabalera
- c. Cachea
- d. Memoria birtuala

#### 9.1.2. Liburua: William Stalling

##### 1. Aurkeztu Williamen kontzeptuak.

- ÿ Williamek kapitulu bat du memoria nagusirako eta beste bat cache memoriarako
- ÿ 4. kapitulua: cachea
  - ÿ Cache memoriaren sarrerak Kontzeptu Orokorrak ditu
- ÿ 5. kapitulua: Barne memoria (DRAM)
- ÿ 6. kapitulua: Kanpoko memoria (biltegiratze periferikoa)
- ÿ 8. kapitulua: Sistema eragileak: Memoriaren kudeaketa

#### 9.1.3. Erref

- Programatzale bakoitzak Memoriari buruz jakin beharko lukeena. Ulrich Drepper. Red Hat, Inc.
  - ÿ bere irakurketa osoak maila aurreratua eskatzen du

#### 9.1.4. Historia

- John Von Neumann ingenieriaren aurrerapen handia IAS ordenagailua garatzea izan zen, zeinetan programak ez zeuden kablerik gabe, baina elektronikoki gordetzen ziren Selectron izeneko memoria-unitate batean. Programak "editatu" ziren kartoi zulatu-txartelak idatzipiz, gero zifra bitarreko sekuentziatan bihurtzen ziren, Selectrón memoria-unitatean kode bitarrean gorde ahal izateko. Modu honetan "gordetutako programa" edo software kontzeptua eta memoria unitateen garapena sortu ziren.
- Memoria ez zen unitate bakar batez osatuta, maila ezberdinietan egituratuta zegoen:
  - ÿ CPU maila: PC, MAR, MBR, IR, IBR, AC eta AR metagailu-erregistroak: egiteko gaitasuna duten erregistroak. instrukzio bat eta datu bat gorde.
  - ÿ Memoria nagusia: Selectron memoria programak gordetzeko ahalmena duena, jarraibideak atal batean taldekatzea eta datuak beste atal batean. 4K hitzak hitz bakoitza 40 bitekin zuzentzeko gaitasuna
  - ÿ Memoria sekundarioa: bilduma bat gordetzeko gaitasuna duten "danbor" magnetikoak programak.

#### 9.1.5. Interesa

- Maila handiko lengoia batean programatzea
  - ÿ Kontuan hartzen al dugu memoriaren kontzeptua? ÿ Kudeaketa mekanismoen abstrakzioa

OS, hardware, etab.

ÿ Memoriaren egitura, antolaketa eta kudeaketa ezagutzeak lagunten du programazioan ÿ Arazte fasea, diseinua, etab.

ÿ Programazioa eta sistemengeniaritza

- Memoriari buruzko aurretiazko ezagutzak: Konputagailuen Egitura irakasgaiko aurreko gaiak.

ÿ aldagai, erakuslea, erregistroak, atalak, helbidea, segfault, estekatzalea,...

- OS ezagutza

ÿ Prozesuen memoriaren kudeaketa, orria, memoria birtuala, TLB, etab.

- Helburua

ÿ Zer: Gorde: datuak eta argibideak ÿ programak ÿ fitxategiak ÿ prozesuak

ÿ Zertarako:

ÿ von-Neumann arkitektura: gordetako programa.

ÿ Instrukzio-zikloa: CPUtik harrapatzea (datuak, argibideak). CPU-RAM bloke-diagrama.

ÿ Nola: nola gordetzen dira, nola harrapatzen dira?

- Fisikoki, memoria da

ÿ chip erdieroalea CPUPa konektatuta.

ÿ PUZaren barneko memoria erdieroalea

ÿ biltegiratze masiboko memoria magnetikoa

- Memoria gaia gainerakoetatik isolatuta dago:

ÿ ordenagailuen arkitektura?:

ÿ CPU? nukleoarena?

ÿ programazioa?

ÿ argibideak eta datuak?

ÿ zer da array bat?

ÿ goto bat?

ÿ muntatzailean zer da .text zuzentaraaua?

ÿ zer da bateria?

## 9.1.6. Ikuspegiak

- Memoriaren kontzeptua ikuspuntu ezberdinatik abiatuta azter daiteke.
- Exekutatzen diren prozesuen memoria kudeatzea sistema eragilearen nukleoak vs Memoriaren Antolakuntza (fitxategiak, atalak, memoria hierarkia,...)
- ISA: memoriarako sarbidea, helbideratze moduak, little endian, etab.
- Softwarea:
  - ÿ Programazioa:
    - ÿ aldagaiak (memoria erreserba eta hasieratzea), erakusleak, memoria dinamikoaren esleipena malloc(), memoria-atalak (testua, datuak, rodala, bss, etab..), pila, ..
  - ÿ Tresnak:
    - ÿ Konpilatzailea, Lokatzailea (helbide lekuadagarriak, helbideen ebatzen, memoria-segmentuak...), Kargatzalea (memoria fisikoa, memoria-mapa, etab.), memoria dump (objdump...)
  - ÿ Sistema eragilea:

- ÿ Memoria birtuala kudeatzailea
- ÿ Fitxategi-sistema birtualak kudeatzea

- Hardwarea:

- ÿ Memoria Kudeatzeko Unitatea (MMU): Memoria-espazio bihurgailua helbide birtualak fisikoan.
- ÿ Memoria Moduluak
  - ÿ txartelak, txipak, autobus konexioa
  - ÿ ezaugarriak: edukiera, abiadura, kontsumoa, teknologia

### 9.1.7. Memoriaren hierarkia

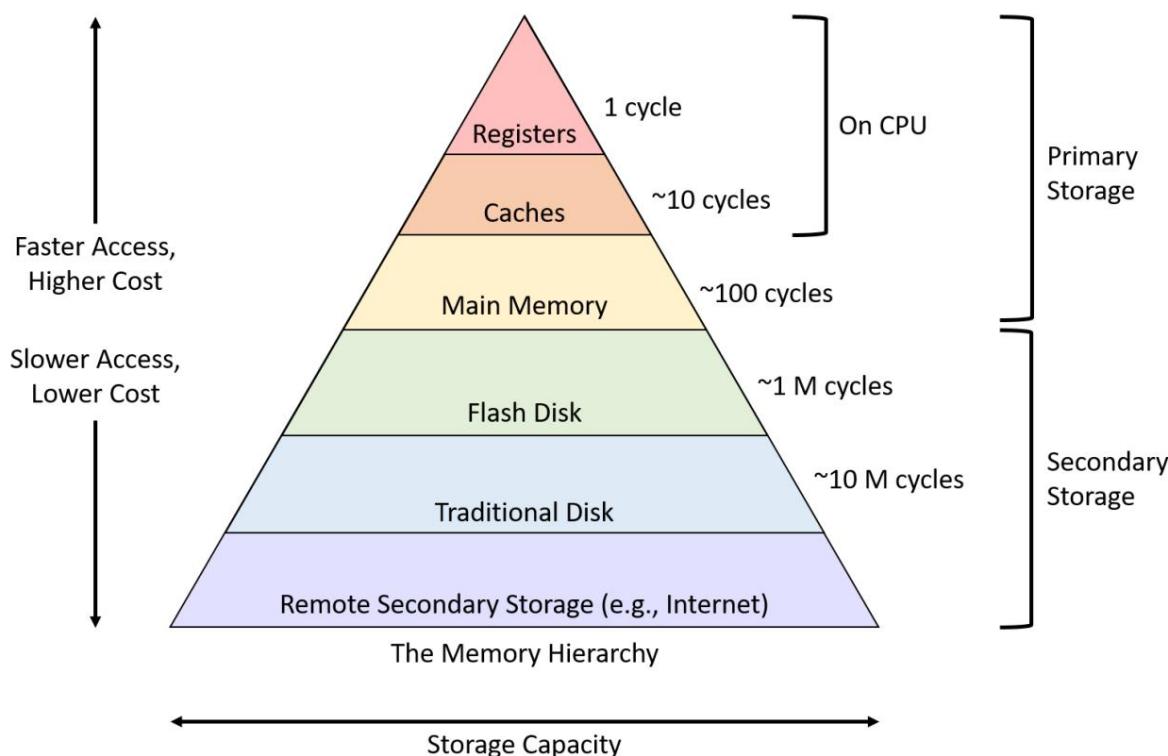
- Gaur egun erdieroaleen teknologiak aurrerapenak lortu ditu ahalmenean eta memorian, mailakako egitura mantenduz hierarkia honetan:

1. Memoriaren hierarkia

- Mailak: L0,L1,L2,L3,L4
- CPU Erregistroak Memoria ÿ L0
- Cache memoria ÿ L1,L2,L3
- Memoria nagusia
- Bigarren mailako memoria: diskoa, pen-drive
- Ezaugarriak

ÿ Goitik beherako goranzko ahalmena

ÿ Goitik beherako memoriaren ziklo-denbora



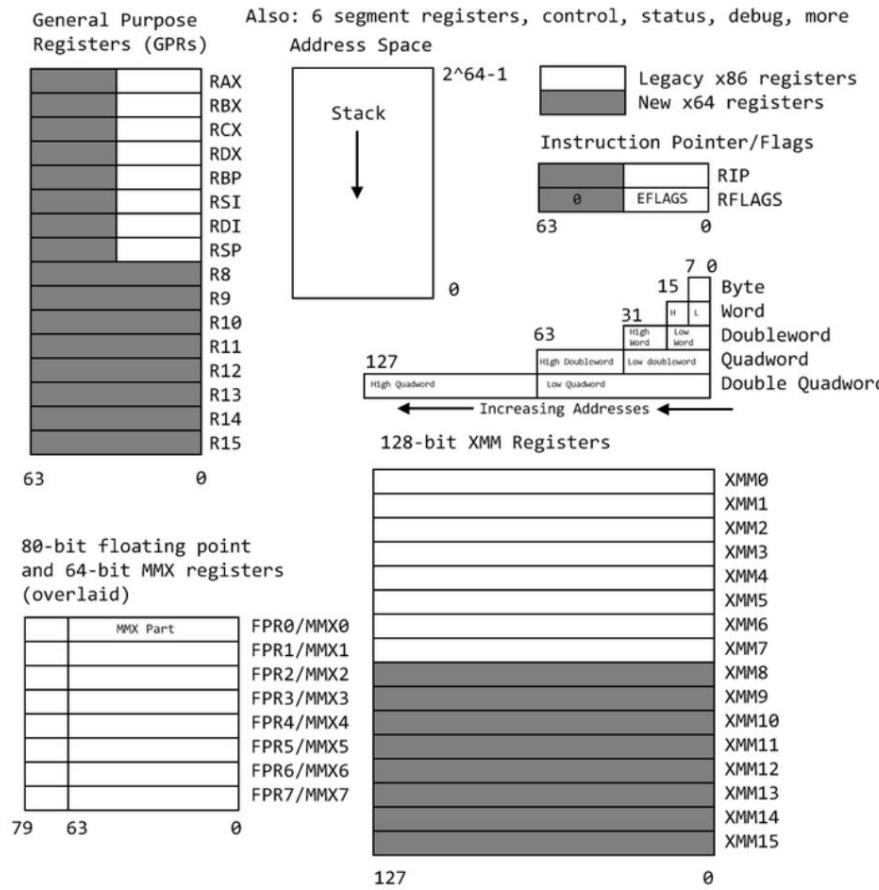
62. Irudia Memoriaren hierarkia

## 9.2. Erregistroak

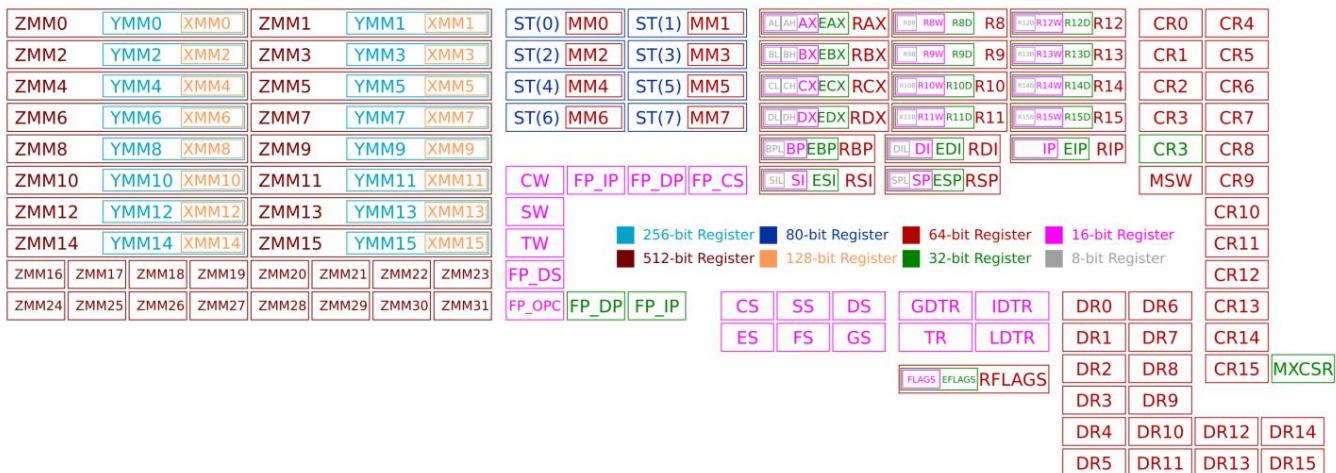
### 9.2.1. DA

- Erregistroak hitz bakarreko edukiera (instrukzioa, datuak, memoria helbidea, aldi baterako emaitza, etab.) eta 1 ns ordenako latentzia duten memoria-unitateak dira. Erregistro hauen izendapen arruntak Programa-kontagailua, Instrukzio Erregistroa, Metagailua, Datu Buffer, etab... dira. Kontrol Unitateak CPU mikroarkitekturaren datu-bidean erabiltzen ditu instrukzio-zikloaren oinarrizko zati gisa. Erregistro hauek ez ditu programatzaileak eskuragarri.
- Programatzaileak eskura ditzakeen erregistroak ordenagailuaren arkitekturaren memoria hierarkiaren L0 maila dira. Instruction Repertoire Architecture (ISA) instrukzio-formatuaren helbide-eremuan aipatzen diren helburu orokorreko Erregistroak ezagutzen dira. X86-32 arkitekturaren kasuan, helburu orokorreko EXtension erregistroak daude: EXA, EXB, EXC, ESP, ESL, etab. (urreko 16 biteko arkitekturaren luzapena AX,BX,CX, etab...).
- Erregistroak CPUaren mikroarkitekturaren parte direnez, atal hau CPUari eskainitako gaiaren barruan ere azter liteke.

### 9.2.2. amd64 arkitektura



63. irudia x86 erregistroak



64. Irudia x86 erregistroak



65. Irudia SIMD argibideen bilakaera

- Kontuan izan rax-ek eax eta eax-ek ax eta ax-ek al barne hartzen dituen bezala, zmm-k ymm e ere barne hartzen dituela.

ymm xmm barne hartzen du

- [x86 wikipedia](#)

- [Erabilera Orokorreko Erregistroak x86](#)

ŷ RAX, etab.

ŷ RFLAGS

ŷ CS-DS-SS-

ŷ ST0-ST7

ŷ Float Point Erregistroak

ŷ FPU erregistroko aliasak

ŷ 80 biteko zabaleko zortzi erregistro: 32, 64 edo 80 biteko koma mugikorrekoa, 16, 32 edo 64 biteko zenbaki oso (bitarra) eta 80 biteko zenbaki oso hamartar paketatua.

## ÿ MMX

ÿ dagoeneko zaharkitua eta XMM-k gaindituta.

ÿ MMX argibideak: SIMD (Single Instruction Multiple Data) integratzea. MMX Intel-ek diseinatutako argibide multzoa da, 1997an aurkeztu zuen bere P5-n oinarritutako Pentium mikroprozesadore-lerroarekin. Ondoren, instrukzio bakar bat 32 biteko bi zenbakiri, 16 bitemko lau zenbakiri edo 8 bitemko zortzi zenbakiri aplika dokieke. hamaika.

ÿ MM0-MM7 (64 bit). Erregistro bakoitzak 64 bitemko zabalera du eta 64 bitemko zenbaki osoak edo hainbat zenbaki oso txikiago "paketatuta" formatuan edukitzeko erabil daiteke.

ÿ paketatutako datu motak: 32 bitemko bi zenbaki oso, 16 bitemko lau zenbaki oso edo 8 bitemko zortzi zenbaki oso aldi berean ÿ zenbaki osoak bakarrik.

ÿ Zenbaki osoekin bakarrik funtzionatzen du baina sistema eragilearen testuinguruaren aldaketetan bateragarritasun arrazoiengatik, MMX eta FPUnren artean alias bat sortu zen, FP eta MMX aplikazio berean ezin erabiltzearen arazoa sortuz baten eragiketak egin zirenetik. besteari eragiten dio.

ÿ x87 FPUnren FP erregistroek pila moduan sarbidea dute, MMX erregistroek sarbidea duten bitartean ausaz.

## ÿ XMM

ÿ Dagoeneko zaharkitua eta YMM-k gaindituta.

ÿ XMM0-XMM15 (128 bit)

ÿ SSE (Streaming SIMD Extensions) instrukzioa. SIMD koma mugikorreko instrukzio-multzoaren luzapena da 1999an Pentium III-rekin aurkeztutako x86 arkitekturara.

ÿ MMXren bilakaera. MMX-ren baliokidea baina koma mugikorreko datuekin.

ÿ FP, MMX eta XMM-ekin batera funtziona dezakezu.

## ÿ YMM

ÿ YMM0-YMM15 (256 bit)

ÿ XMM hedapena 256 bitera

ÿ AVX: Luzapen bektorial aurreratuen argibideak

ÿ Koma mugikorreko datuak soilik: koma mugikorreko SIMD

ÿ AVX-ek hiru eragileko SIMD instrukzio-formatua aurkezten du, non helmuga-erregistroa bi iturburuko eragiketetik bereizten den.

ÿ Bateragarritasuna: AVX argibideek 128 bitemko eta 256 bitemko SIMD onartzen dute

ÿ Intel 2011n hasi zen multzo honekin: Sandy Bridge prozesadorea, 2011ko Q1.

ÿ AVX2

ÿ Haswell mikroarkitektura urtea 2013: Haswell prozesadorea, 2013ko Q2

ÿ AVX2 - Osoko datu-motak 256 bitemko SIMDra zabaldu dira

## ÿ ZMM

ÿ (ZMM0-ZMM31): 512 bit

ÿ Intel AVX-512: 2013ko uztaila

ÿ Programek doitasun bikoitzeko zortzi zenbaki edo doitasun bakarreko hamasei koma mugikorreko zenbaki, edo 64 bitemko zortzi zenbaki oso, edo 32 bitemko hamasei zenbaki oso 512 bitemko bektoreen barruan bildu ditzakete. Honek AVX/AVX2-k instrukzio bakar batekin prozesatu ditzakeen datu-elementuen bikoitza eta SSEren lau aldiz gehiago prozesatzeko aukera ematen du.

ÿ AVX-512 argibideak

• Kontrol Erregistroak x86:CRx

ÿ Kontrolatu, adibidez, memoria-orria.

- Arazte-erregistroak x86:DRx
  - ÿ Etenguneen helbideak ezartzeko erabiltzen dira, adibidez: DR0-DR3
- Egiaztu
  - ÿ Linux-en cpuid instrukzioarekin cpuak luzapenekin duen bateragarritasuna egiaztu dezakegu  
ISA: mmx, sse, avx, etab.
- Kontzeptuak
  - ÿ SIMD: instrukzio bakarreko hainbat datu.
  - ÿ Kodea bektorializatzea: bektoreekin funtzionatzen duten argibideak ÿ datu anitzetan eragiketa bat aldi berean.
  - ÿ DSP: Seinale Digital Prozesadorea
- Erregistro motak
  - ÿ [https://en.wikibooks.org/wiki/Microprocessor\\_Design/Register\\_File](https://en.wikibooks.org/wiki/Microprocessor_Design/Register_File)
  - ÿ Erregistro fitxategia: helbide-busa duten erregistro-sekuentziaz osatutako memoria estatikoa horrek, deskodetzaile baten bidez, erregistroetako bat hautatzen du.
  - ÿ Erregistro Bankua: bi interpretazio posible.
    - ÿ Etendurak kudeatzeko banku-erregistroak: memoria nagusia (pila) erabili beharrean, prozesu baten exekuzioa eteten denean kanpoko eten baten ondorioz erregistroak gordetzeko eta berreskuratzeko, barneko CPU erregistroak erabiltzen ditugu horretarako: etenaldia. Teknika hau ezartzeko modua eteten den errutinak erabiltzen dituen erregistroak etendako errutinari dagokionez berrizendatzea da.
  - ÿ Bankuen arabera taldekatzea: Erregistroen multzoa egon daitezkeen bankuen arabera taldekatzen da aldi berean sartzen da.

### 9.3. Memoria nagusia (RAM dinamikoa DRAM)

#### 9.3.1. Erdieroaleen Memoria motak

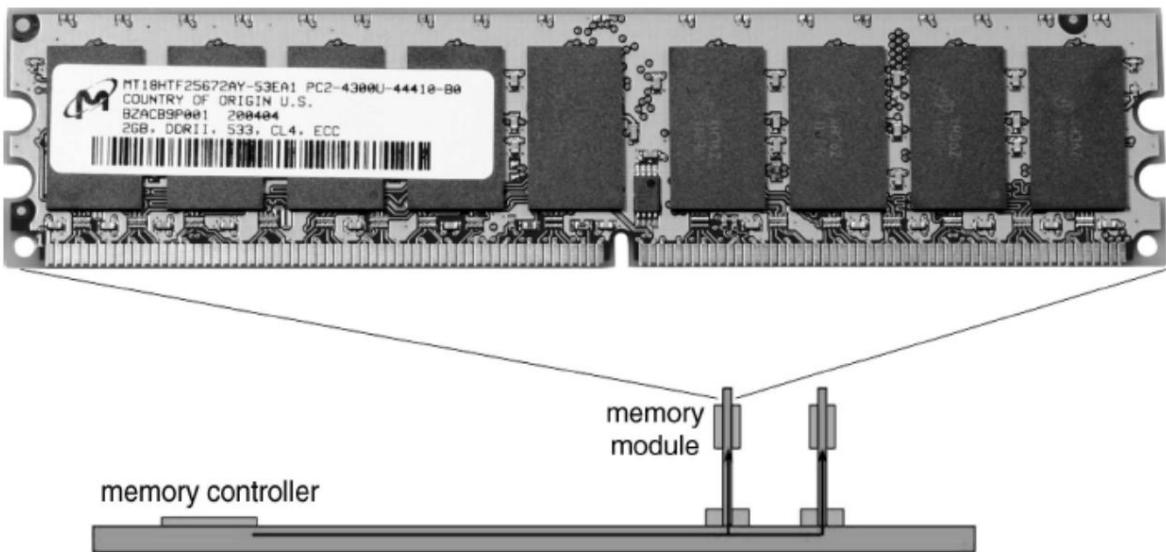
Memoria Mota Kategoria		Ezabatzea	Idatzi Mekanismoa	Lurrunkortasuna
Ausazko sarbideko memoria (RAM)	Irakurri-idatzi memoria	Elektrikoki, byte mailakoa	Elektrikoki	Lurrunkorra
Irakurtzeko soilik memoria (ROM)	Irakurtzeko soilik memoria	Ezin da	Maskarak	Ez lurrunkorra
ROM programagarria (PROM)			Elektrikoki	
PROM ezabagarria (EPROM)	Irakurri-gehiengoa memoria	UV argia, txip maila		
Elektrikoki PROM ezabagarria (EEPROM)		Elektrikoki, byte mailakoa		
flash memoria		Elektrikoki, bloke mailakoa		

- Ausazko Sarbidea: edozein memoria-helbidera AUSAZ sar daiteke, ez sekuentziala (3. posiziora sartzeko ez da beharrezkoa lehen 1. eta 2. posizioa sartzea)
- Lurrunkorra: energia kentzen bada, memoriaren edukia ezabatzen da.

- ROM: Irakurtzeko soilik memoria ÿ Memoriaren edukia fabrikan eta erabiltzaileak bakarrik idazten du irakur dezakezu.
- PROM: ROM programagarria ÿ Erabiltzaile-programek memoria bakarrik irakur dezakete administratzaleak eguneratu dezake. Adibidez, BIOSa.
- Ezabagarria: ROM memoria programagarria ezabatu behar da idatzi aurretik ÿ argiarekin ezabatzen da ultramorea
- EE: elektrikoki ezabagarria . PROM elektrikoki ezabatzen da UV argiaren ordez
- Flasha: sekuentzialki ezabatu beharrean, hitzetik hitzera, hau da, motela, memoria ezabatzen da. bloke handien bidez. Bloke handi bat berehala ezabatuko da (flash efektua)
- SSD: Solide State Drive: Edukiera handiko flash memoria  
ÿ SLC, TLC eta MLC (maila bakarreko, hirukoitzerako eta anitzeko gelaxka)

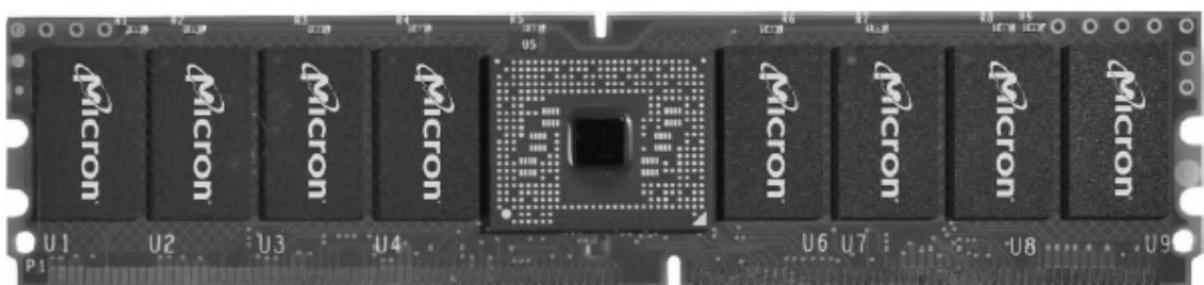
### 9.3.2. Erdieroaleen memoria NAGUSIA

- Bigarren mailako memorian gordetako programa bat (kanpoko diskoa) sistema eragileak Memoria Nagusian kargatu behar du CPUak exekutatu ahal izateko. Kontrol Unitateak memoria nagusian dauden argibideak/datuak harrapatzen ditu.
- CPUak memoria nagusiarekin egiten dituen eragiketak IRAKURTZEA eta IDAZKETA dira (Irakurri/idatzi) datuen eta argibide bitarrenak.
- RAM:
  - ÿ Silizio Erdieroalea: transistoreak.
  - ÿ Ausazko Sarbide Memoria
  - ÿ Irakurri eta idazteko eragiketak
    - ÿ Lurrunkorra
    - ÿ Motak
  - ÿ SRAM: RAM estatikoa.
    - ÿ Elikatzen den bitartean, informazioa ez da galtzen. Ez du ihes egiten, ez da beharrezkoa freskatu informazioa milisegundo gutxitan behin.
    - ÿ 6T Zelula Egitura: sei transistore. Tamaina eta kontsumo handia. Latentzia eta gaitasun murritzua.
    - ÿ Cache memoria.
  - ÿ DRAM: RAM dinamikoa.
    - ÿ Zelula egitura: 1C1T: kondentsadorea eta transistorea. Tamaina eta kontsumo murritzua. Latentzia eta ahalmen handia.
    - ÿ Memoria nagusia: Programak memoria horretan kargatzen dira exekutatzeko CPU.
    - ÿ Bit-zelulek (kondentsadoreek) ihes egiten dute, beraz, aldizka behar dute berridazketa (DINAMISMOA).
    - ÿ DRAM asinkronoa:
    - ÿ SDRAM sinkronikoa: R/W eragiketak markatutako uneetan egiten dira a



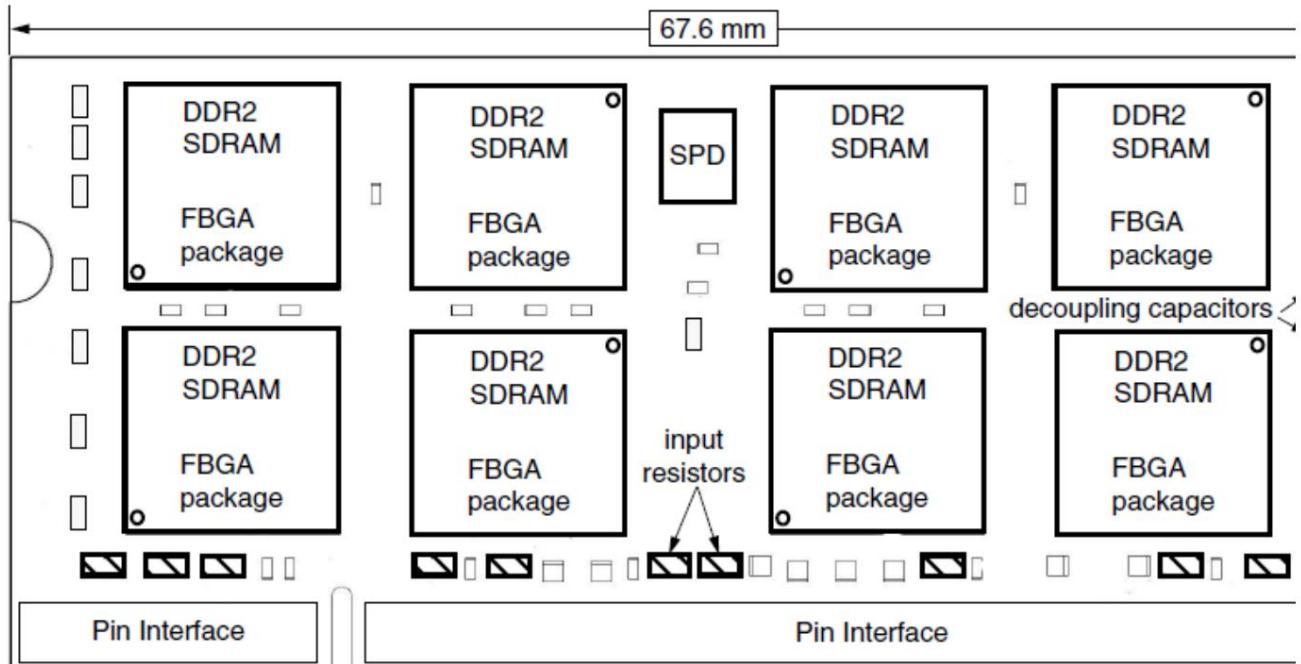
66. Irudia. Lineako memoria-modulu bikoitza (DIMM)

Modulu bakoitza ordenagailuaren plakako zirrikitu batean sartzen da.



67. Irudia. Lineako memoria-modulu bikoitza (DIMM)

Txartela edo Modulua txip multzo bat da. Txipak normalean DIMM moduluaren bi aldeetan soldatzen dira.



68. Irudia. Lineako memoria-modulu bikoitza (DIMM)

#### Antolaketa

- Memoria-zelula:

ŷ bit baten oinarrizko biltegiratze-unitatea da (Binary digit). Bitak balio logiko altua edo baxua da, 1 edo 0

ŷ gelaxkarako sarbidea:

ŷ Helbide-lerroak irakurtzeko edo idazteko gelaxka hautatzen du

ŷ bit-lerroa irakurri edo idatzi beharreko bitaren sarrera/irteera lerroa da.

- Matrizea: memoria-zelulak errenkadaz eta zutabeez osatutako 2D matrize-egitura batean antolatzen dira.

- System Bus: interkonexio-busa MC kontroladorearen (Memory Controller) eta CPUaren artean.

ŷ PUZaren kontrol-unitatea ez da zuzenean komunikatzen Memoria Nagusiarekin. PUZaren Kontrol Unitateak irakurtzeko eta idazteko aginduak ematen ditu, baina Memoria Nagusiaren antolaketa eta teknologiaren ekintza primitibo eta espezifikoagoak baztertzen ditu. DRAM memorien zeregin zehatz horietarako MC kontrolagailua dago

- Memory Bus: MC kontroladorearen eta MPren arteko interkonexio-busa

ŷ Autobusaren jarrainbideak:

ŷ Helbide-busak hautatu beharreko hitzaren kodea transferitzen du

ŷ Helbidea aldi baterako memoria-helbideen bufferean gordetzen da

- ÿ Helbide-busa memoria-helbideen bufferera konektatzen da
- ÿ Helbide-buffer-a memoria-helbideen deskodetzailearen sarrerara konektatuta dago
- ÿ Helbidea deskodetuta dago. Deskodetzailearen irteerak memoria helbidea aktibatzen du irakurtzeko edo idazteko datuak/argibideak

ÿ Datu-busa:

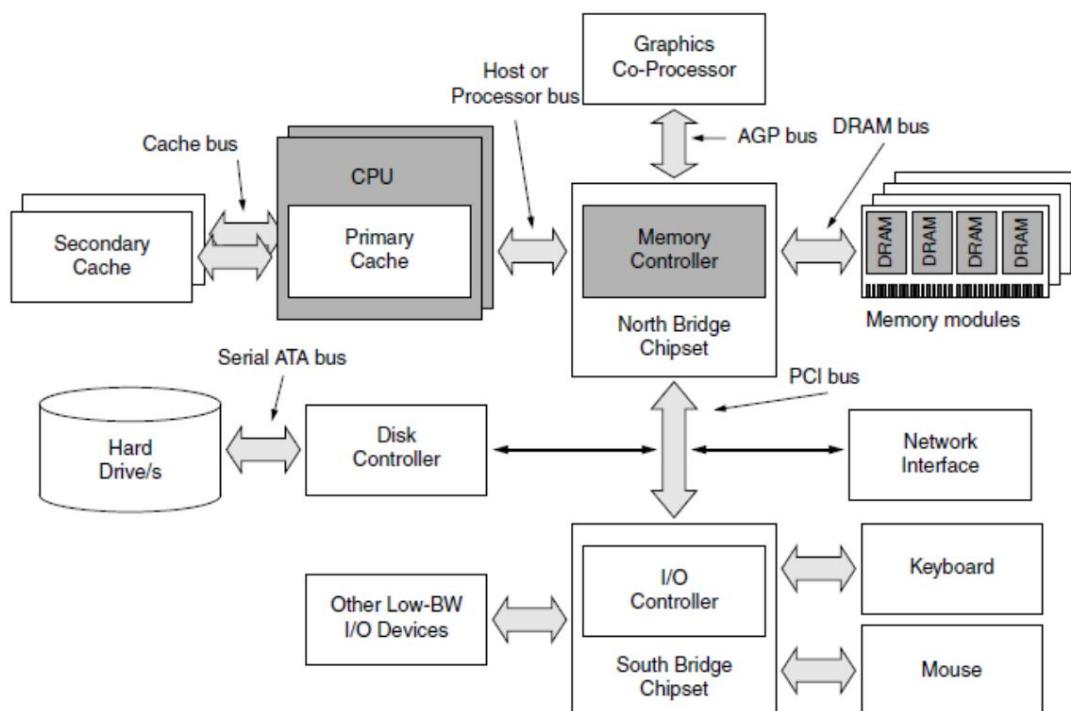
- ÿ Irteerako edo sarrerako datuak aldi baterako gordetzen dira memoriako I/O datuen bufferean
- ÿ Gelaxkak ez dira zuzenean konektatzen memoriako I/O datu-bufferera
- ÿ Hautatutako gelaxken irteerak amplifikatu egiten dira 0 edo 1 gordetzen duten detektatzeko
- ÿ Datu-busa memoria-datuuen bufferera konektatuta dago

ÿ Autobusen kontrola:

- ÿ Memoria volt gutxiko tentsio jarraituarekin (1v) elikatzea beharrezkoa da.
- ÿ PUZA edo I/O kontrolagailua aktibatzen duen seinalea irakurri eta idatzi
- ÿ Sinkronizazio erlojuaren seinalea. MP eta kontroladorearen artean egin beharreko zereginak sinkronizatzen ditu. memoria (MC)

ÿ Autobus txiparen hautaketa:

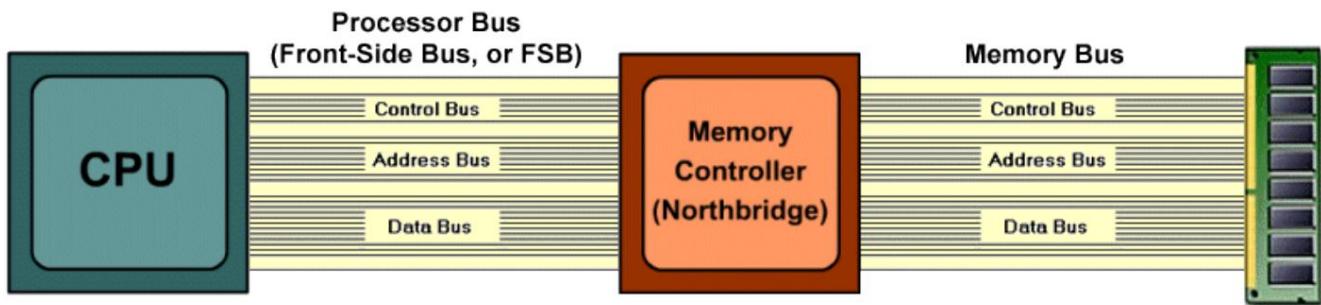
- ÿ Chip Select (CS) seinalea helbide eta datu-busetara konektatzen duen memoria-modulua hautatzeko. CS seinalea aktibo ez badago, memoria modulua busetatik deskonektatuko da.



#### 69. Irudia Sistema Informatikoa

- Memoria kontrolatzailea (MC)
  - ÿ MPa ez da zuzenean CPUra konektatzen. MC kontrolagailuak bitartekari gisa jarduten du.

- ŷ MC kontrolagailua alde batetik CPUrak konektatuta dago eta bestetik MP memoriara.
- ŷ CPUak komandoak bidaltzen dizkio MC kontroladoreari, MP-n jarduteko.
- ŷ MC kontrolagailua barne-egituraren nola jokatzen dakien sekuentziatzalea da memoria:
  - ŷ zein modulu hautatu, zein txip hautatu, zein hitz hautatu.
  - ŷ datuak irakurri eta idatzi
  - ŷ memorian beste ekintza batzuk, hala nola mantentze-lanak, egiaztapenak, akatsak hautematea, etab.



70. Irudia Memoria-kontrolatzalea

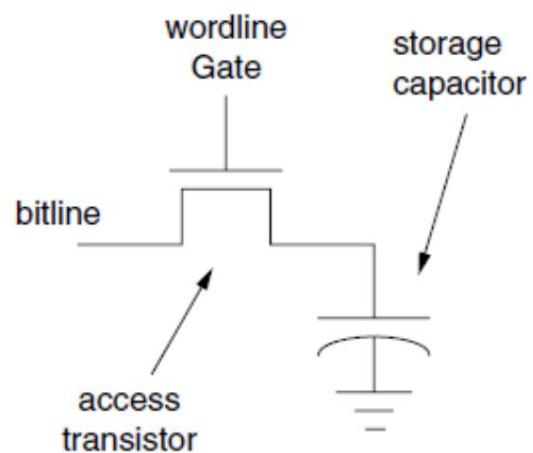
- Memoria Kudeatzeko Unitatea (MMU)

- ŷ CPU Kontrol Unitateak bere helburu orokorreko erregistroetan funtzionatzen duen helbideak, programa-kontagailua, etab., ez dira fisikoak ŷ **helbide birtualak** dira
- ŷ Programatzeko dugunean, programatzaleak, konpilatzaleak, estekatzaleak, desmuntatzaleak, arazleak eta abar espazio birtualean lan egiten dute. ELF exekutagarria den modulua eta prozesuak erreferentziako espazio birtuala.
- ŷ Prozesuek (PUZak exekutatzeko ari diren programak) espazio birtualeko helbideekin funtzionatzen dute ŷ prozesuaren memoria birtuala
- ŷ MMU: HW zirkuitu elektronikoa, espazio birtualeko helbideak (CPU) MPren helbide fisikoetara **itzultzen dituena** eta memoria fisikora sartzeko cache kontroladorearen busera eta MC memoria kontrolagailura transferitzen direnak izango direnak.

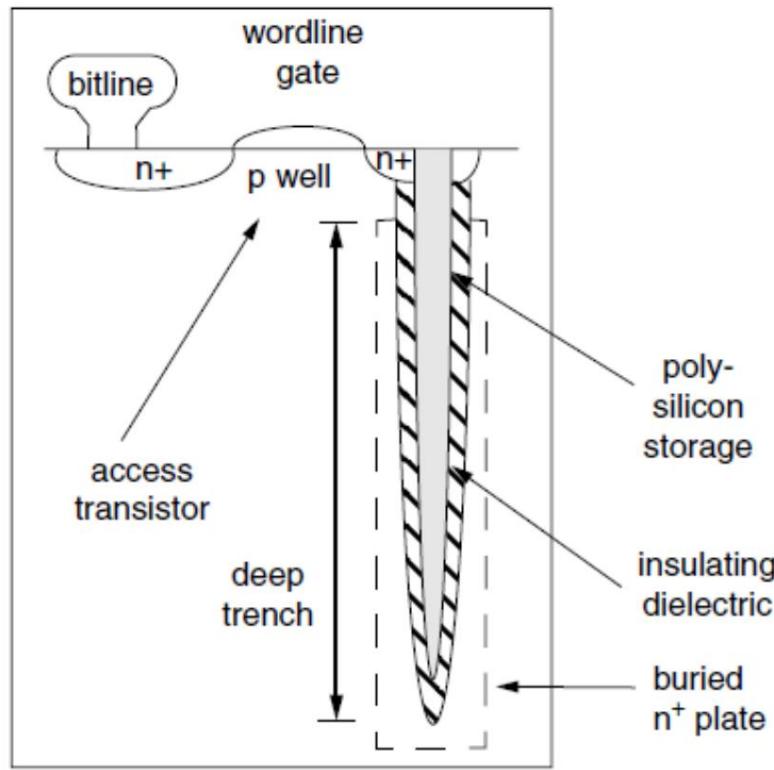
#### **DRAM (Dynamic Random Access Memory)**

- Zelula

- ŷ Egitura fisikoa:
  - ŷ Siliziozko substratu batean fabrikatutako Metal-Dielektriko-Metal (Polysilicon) kondentsadorea da.
  - ŷ Bere ahalmena femto-farad-en ordenakoa da:  $C=10\text{--}(-15)\text{F}$
  - ŷ  $1\text{mv}$ -ko tentsioa aplikatzen badugu, gordetako karga  $Q=CV=1\text{mv}\cdot1\text{fF}=1\cdot10\text{--}18\text{ coulomb}$  dozena bat elektroiren baliokidea dena.
  - ŷ Bere forma substratuan txertatutako zilindro batena da.
  - ŷ Kondentsadorearen sekzioa  $30\text{ nm}$ -ko ordenakoa da 2010ean
    - ŷ **prozesu teknologikoaren bilakaera:**  $3\text{nm}$ -ko nodoa 2023an
  - ŷ Kondentsadoreen dentsitatea gigabyte-ren ordenakoa da ŷ  $10\text{--}9$  kondentsadore.
  - ŷ Beharrezkoa da kondentsadorea helbidera eta bit-ferroetara konektatzea bertara sartzeko. Etengailu gisa jokatzen duen CMOS transistore baten bidez konektatzen da.



71. irudia zelula\_1t1c



72. irudia zelula\_kondentsadorea

DRAM (irakurketa-idazketa-freskatze eragiketak)

- Biltegiratzea:

ŷ kondentsadorea hasiera batean ez dago linea batera konektatuta, bere etengailua irekita dagoelako

ŷ Zirkuitu irekian kondentsadoreak karga gordetzen du elikatuta dagoen bitartean ŷ lurrunkorra

ŷ kondentsadorea ISURTASUNA DA eta substratutik deskargatzen da. Bit-a 64 ms behin berridatzi behar da: DINAMIKOA (gordetzen duen informazioa ezin da estatikoa izan, ALDIKIO FRESKATZEN DA)

ŷ Idazketa:

ŷ Etengailua (helbide-lerroa) ixten dugu kondentsadorea bit-lerroarekin (llerroa datuak)

ŷ Bit-lerroaren bidez kondentsadorea kargatu (H) edo deskargatzen dugu (L).

ŷ Irakurketa:

ŷ Irakurri beharreko gelaxka hautatu ondoren, bere egoera detektatzen duen eta I/O bufferean idazten duen Load Sentsoreari (amplifikadoreari) konektatzen zaio.

ŷ Irakurketa hau suntsitzalea da, kondentsadorea deskargatuta utziz. Beharrezko da amplifikadoreak kondentsadorea bere jatorrizko egoerara itzultzea. I/o buffer-aren idazketa eta kondentsadorearen RE idazketa aldi berean gertatzen dira.

ŷ Freskagarria

ŷ Beharrezko da kondentsadore guztiak irakurri eta berridatzi. Eragiketa hau sentsoreak egiten du zama.

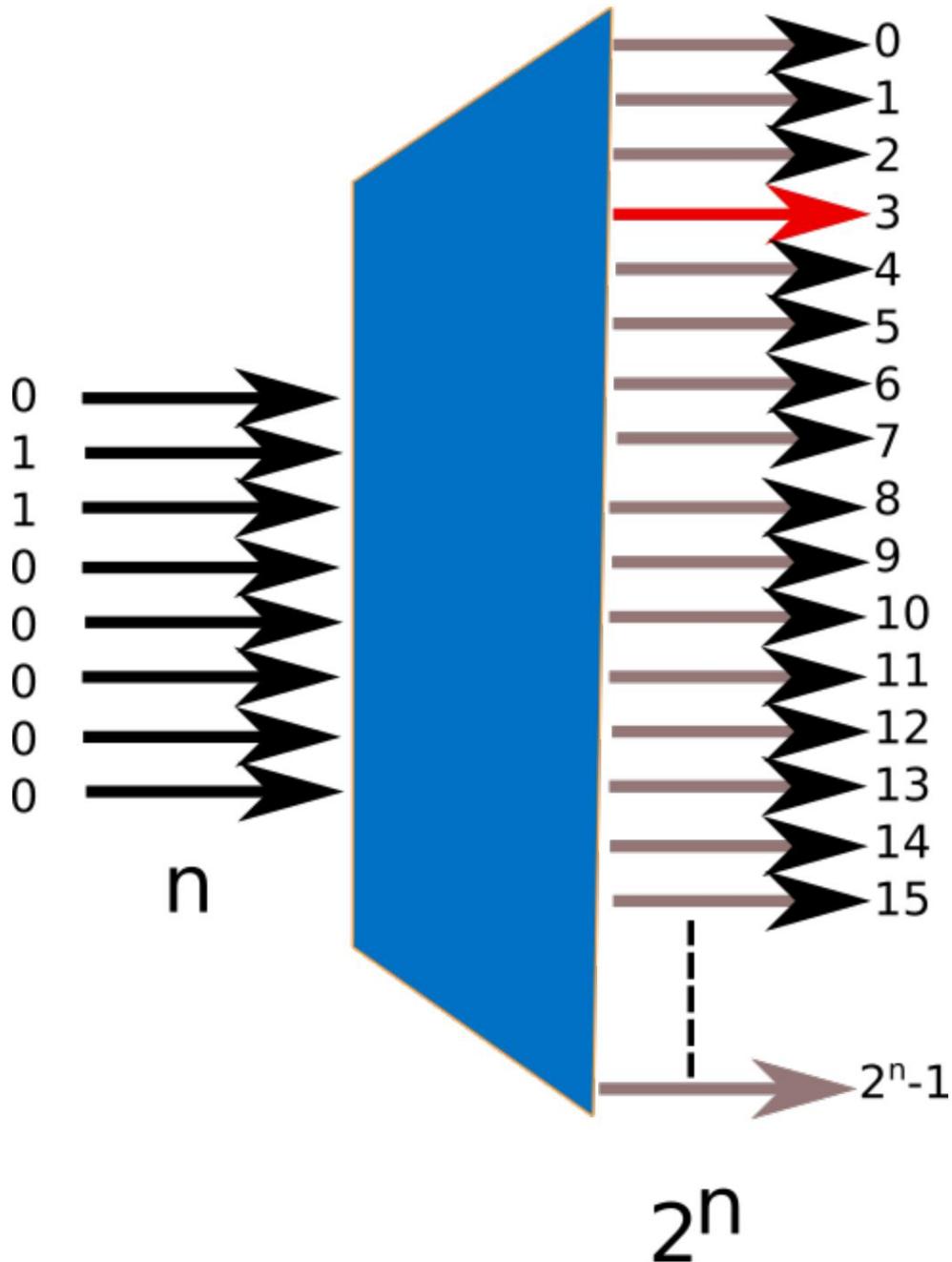
ŷ Beharrezko da gelaxka guztiak berridaztea 64 ms baino gutxiagoko denboran.

## Egitura Adibidea

- <https://www.anandtech.com/print/3851/everything-you-always-wanted-to-know-about-sdram-memory-but-galdetzeko-beldur-ziren>

### DRAM array (2D array)

- Adibidea: 30 lineako helbide-busak 230 ZELULAREN bat zuzen dezake , hautatu beharrekoak. Haietako bat.
- 30 sarrera bitar (helbide-busa) eta 230 irteera (irteera bakarra aktibatuta) dituen DEMULTIPLEXADOR bat erabiliz, memoria gelaxka edo bit bat duen errenkada 1 hauta dezakegu.



73. Irudia Desmultiplexadorea/Deskodetzailea

- 230 irteera dituen demultiplexadorea (1.073.741.824 irteera) ezinezkoa da fabrikatu
- Irtenbidea:

ŷ Antolatu ZELULAK 2D array batean: Errenkadak eta zutabeak: 1 DEMUX edo DESKODETZAILEA errenkadetarako eta 1 DEMUX/MUX edo DESKODETZAILEA zutabeetarako

$\hat{y} 2^{30} = 215*215 =$  Orain demux bakoitzaren irteera kopurua 230etik 215era murriztu da, hau da, 32.768 irteerara murriztu da, erro karratuko faktorera. 32.768 irteera dituen multiplexera fabrikatu daiteke.

ÿ hitz-lerroak gelaxken errenkada bateko (ROW) zutabe guztiak hautatzen ditu  
 ÿ bit-lerroak hautatutako errenkadaren zutabeetako bat (COL) hautatzen du  
 ÿ emaitza ARRAY-tik CELL bat hautatzea eta I/O BUFFER-a kargatzea (IRAKURRI) da.

- Helbide-bus oso trinkoa: 30 linearen adibidea

ÿ Linea kopuru erdia duen autobus bat diseina dezakegu eta kodea bi aldiz multiplexatu.  
 norabideak (errenkada hautatzen duen zatia eta zutabea hautatzen duen zatia).

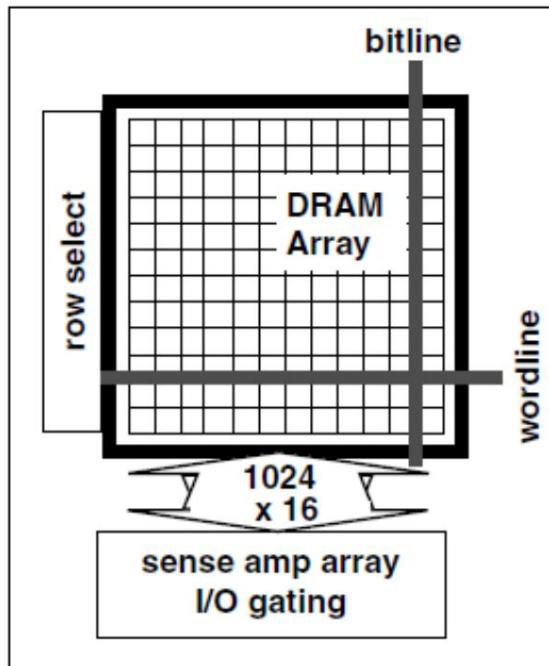
ÿ Errenkada helbidearen eta zutabearren helbidearen denborazko multiplexazioa: MURRIZTU DUGU LERRO KOPURUA AMA PLAKA

ÿ RAS: Row Address Strobe: Helbide-busa balioztatzen duen seinalea kodea dela adierazten duena.  
 hautatu matrizearen ERRENKADA.

ÿ CAS: Row Address Strobe: Helbide-busa balioztatzen duen seinalea kodea dela adierazten duena.  
 matrizeko ZUTAPE hautatzen du.

- Lehertu

ÿ Behin gelaxken errenkada bat hautatuta (OPEN ROW), zutabe bereko ondoz ondoko gelaxkak nahi baditugu erloju-ziklo bakoitzean jarraian irakurri edo idatzi ditzakegu. Cache memoriara/tik transferitzeko hitz blokea. Memoria kontrolatzaleak komando bat bidali behar izan zuen memoriara, eztandako hitz kopurua ezartzeko.



74. Irudia. Gelaxken array

- Gelaxken antolaketa dimentsio bakarrean: 1D: Errenkadak.
- Errenkada deskodetzailearen zirkuitu bat

- 1024 errenkada eta errenkada bakoitzak 16 gelaxka ditu.
- I/O buffer-a 16 bitekoa da.
- Errenkada bat zuzentzean, 16 gelaxkak i/o bufferera edo atebideko i/o-ra konektatuta daude.

ÿ Zelulek (kondentsadoreek) oso karga gutxi gordetzen dute (picocoulombs) eta, beraz, ezin dira zuzenean helbide-busera konektatu, amplifikadorea edo CHARGE sentsore bat beharrezkoa da. Amplifikadore horren irteeran sarrera/irteera I/O erregistroa edo erregistroak daude (BUFFER). Hau da, zelulen eduki bitarra BUFFER I/Ora isurtzen da. I/O buffer-a plakaren DATU bus-lerroak gidatzeko gai den teknologia batean egina dago.

[array2] | images/dram/array2.png 75.

Irudia. 2D zelula-matrizea

- Gelaxken antolaketa bi dimentsiotan: 2D: errenkaden eta zutabeen matrizea
- Bi deskodetzaile: errenkada-deskodetzailea eta zutabe-deskodetzailea.
- Bikote bakoitzak (errenkada, zutabea) gelaxka batera sartzen da.

#### Chip Logic (liburuko 5.3 irudia)

- Osagaiak:

ÿ buffer-ak: helbideak eta zutabeak

ÿ deskodetzaileak: errenkada eta zutabe helbidearen kodea deskodetu eta hautatu (errenkada, zutabea) zelula bat

ÿ karga-sentsoreoa (anplifikadorea): zelula kargatuta edo deskargatuta dagoen detektatzen du eta H bat ateratzen du.edo L i/o buffer-ean.

ÿ 4 kontrol-seinale: RAS, CAS, WE, OE

ÿ kontrol-seinaleen konbinazioa (24) ere erabiltzen da komandoak kodetzeko memoria kontrolatzaila.

ÿ KANDOAK: memoria-moduluetarako aginduak dira, non memoria-ezaugarriak, hala nola, denbora (latentzia-denborak, zikloa, etab.) eta leherketaren luzera (bloke bakoitzeko hitz kopurua, leherren luzera) programagarriak diren eta, beraz, CPUak parametro hauek konfigura ditzake.

ÿ freskagarri zirkuitua:

ÿ Helbide-kontagailua eta temporizadorea

ÿ JEDEC elkartea 64 ms behin freskatze osoa gomendatzen du.

#### Kapsulatua

- Erdieroaleen memoriak babestu behar diren mm2 batzuk hartzen ditu (termikoak eta mekanikoak) eta konexioak sendoak izatea ahalbidetzen dute kanpoko lineetara soldatzeko, horregatik plastikozko kapsulazioa behar du.
- Paketearen terminalei PIN deitzen zaie eta memoria-txartelean soldatzen dira.

ÿ pinak edo terminalak:

ÿ helbide-busa (A0-A29)

ÿ datu-busa (DQ0-DQ7): txipek ez dute 64 datu-pin: 1,2,4,8.

ÿ Vcc potentzia

ÿ masa Vss

ÿ txipa hautatzea /CS

- ÿ idazketa gaitza (/WE): L (idatzi) H (irakurtzea)
- ÿ Irteera gaitu (/OE): L (datu pinak datu-busera konektatzen dira)

### **Irakurketa/idazketa eragiketa denbora**

- **Etranskinak** memoria txartelarekin MC memoria kontrolagailuaren eragiketen denbora kontrolatzen duten seinaleei buruzko informazio zehatza jasotzen du.
- Denbora edo latentzia tRAS, tCAS, etab. **denbora-tarteen kontrol** -seinale desberdinak aktibatzen diren denbora-tarteak kontrolatzen dituzten seinale guztien artean errespetatu beharreko atzerapenak dira , memoria helbideak hautatzeko eta horiek aktibatu arte behar den denbora itxaroteko. memorian irakurtzeko edo idazteko zikloa osatu.
- Sinkronismoa
  - ÿ Memoria kontrolatzeko busak irakurketa eta idazketa eragiketak **kontrolatzen** dituen hari bat du memoria nagusitik. Ez da nahastu behar CPU Kontrol Unitateko erlojuarekin.
  - ÿ **SDRAM** : RAM dinamiko sinkronikoa, **erloju bat** (eta erlojua gitzeko) lerro bat gehituz . Beste seinale guztiak erlojuaren goranzko ertzean jasotzen dira. Ez du ahalik eta azkarren erantzuten, baina goranzko ertzaren zain dago.
- MP memoria modulu programagarria da tCL-tRCD-tRP-tRAS denborak alda ditzakegu eta eztandaren luzera ere (leherketa edo blokea)
- MP memoria moduluak normalean tCL-tRCD-tRP-tRAS sekuentzia adierazten du ziklo-balio tipikoekin memoria-erlojuarena.

### **Taldekatzeak: Moduluak-Kanka-Txipak-Bankua**

- Hierarkia: DRAM memoriaren egitura helbide taldeetan

### **Kanala**

- Kanal:
  - ÿ memoria kontrolagailuak sistemaren busarekin interfazea du.
  - ÿ Kanal bakoitzak bere memoria fisikoaren busa du.
  - ÿ Kontrolagailuak sistema-busera eta memoria-bus bat baino gehiagorako sarbidea du.
  - ÿ Memoria kontrolagailu bereko kanal guztiak memoria fisikoaren espazio osoa osatzen dute, beraz kontrolagailu bati kanal logiko bat esleitzen zaio (memoria-espazio osoa) hainbat fisikok (memoria-espazio desberdinak) osatutakoa.

### **DIMM modulu**

- Lineako memoria-modulu bikoitza (DIMM):
  - ÿ Dual Inline esan nahi du txartelak edo moduluak konektoreak dituela txartelaren bi aldeetan.
  - ÿ Datu-buserako konexio fisikoa (64 biteko hitza), helbide-busa, eta kontrola eta MEMORIA BUSaren txip-aukera (CS) busera.
  - ÿ Memoria kontrolagailuaren (MC) memoria-busera konektatzen den plakaren entxufean sartzen den kapsulatutako memoria txartela da.
  - ÿ PCetarako, memoria moduluen konexioa **DIMM** da eta **SO-DIMM** ordenagailu eramangarrietarako. DIMM kapsulatzeak txartelaren bi aldeetan konektoreak eta txipak (urrealdekoa eta atzoko aldea) ahalbidetzen ditu.
  - ÿ Txartelaren memoria-txip guztiak interkonektatuta daude moduluan.
  - ÿ Kontrolagailu-kanal bat memoria-modulu batera baino gehiagotara konekta daiteke: Adibidez, 4 GBko bi modulu bakoitzak.Kanal batek modulu bat baino gehiago baditu, modulu guztiak BUS bera partekatzen dute.

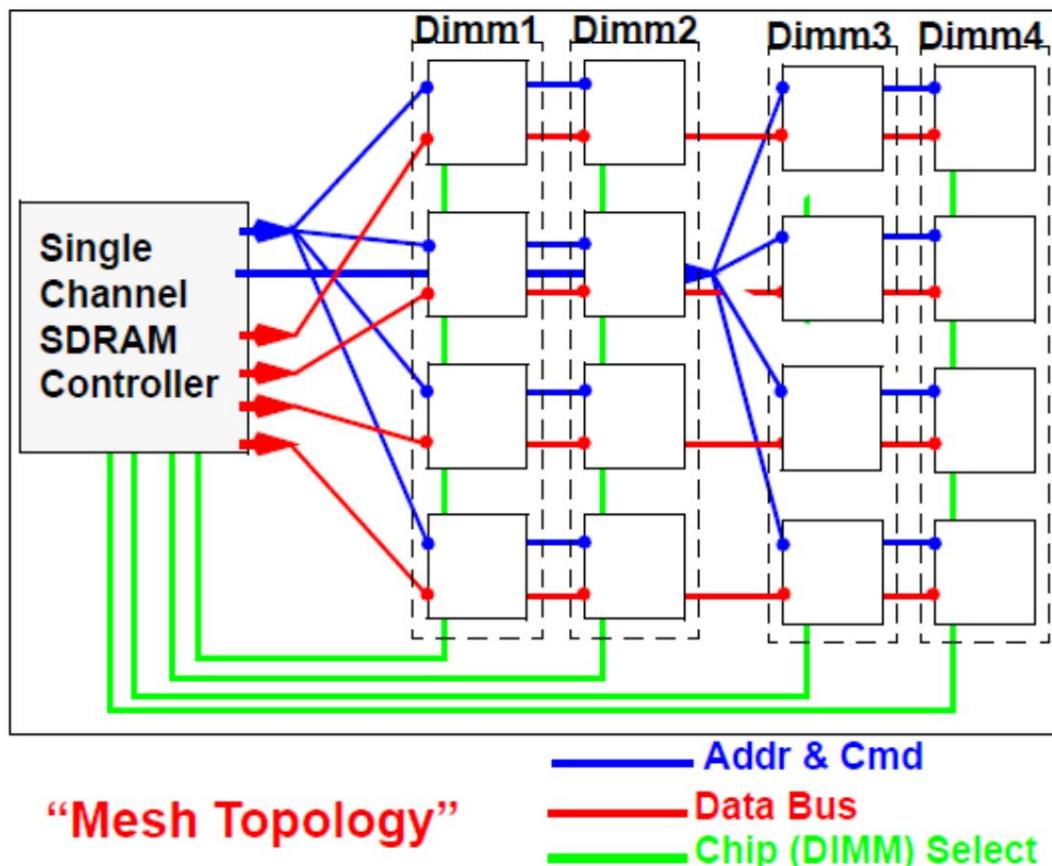
**MEMORIA.** Modulu bakoitzak memoria helbide desberdinak ezartzen ditu.

## Sailkapena

## • Sailkapena:

- ŷ Memoria-sistema osoaren barruan dauden bankuen multzoa edo multzoa da (DIMM modulu guztiak, ez DIMM modulu bakoitza). Non bankuak Txipetan ezartzen diren. Maila bateko bankuak hautatzeko, banku horien txipak konektatu behar dira, non txip-hautaketaren (CS) seinalea guzientzako komuna izan dadin, eta horrela helbide-espazio bera partekatuz.
- ŷ Modu honetan antolatuta, maila bereko txip-banku guztiiek BATEAN erantzun diezaiekete helbide-bus berari (erenkadak, zutabeak). CS seinalea aktibatuz eta Errenkada bat hautatuz gero, maila bereko txip-banku guztiak array guztiak errenkada guztien zutabe guztiak aktibatzen dira. Hau da taldekatzearen helburua.
- ŷ Maila bat gainontzekoekiko independentea da, memoria helbide ezberdinekin, helbide-bus bera partekatzen du eta txip-busa hautatu. Maila bat, INDEPENDENTE izaki, aurretiaz kargatu, freskatu, aktibatu eta abar egin daiteke gainontzeko mailak dituzten aldi berean.

## Memory System Organization

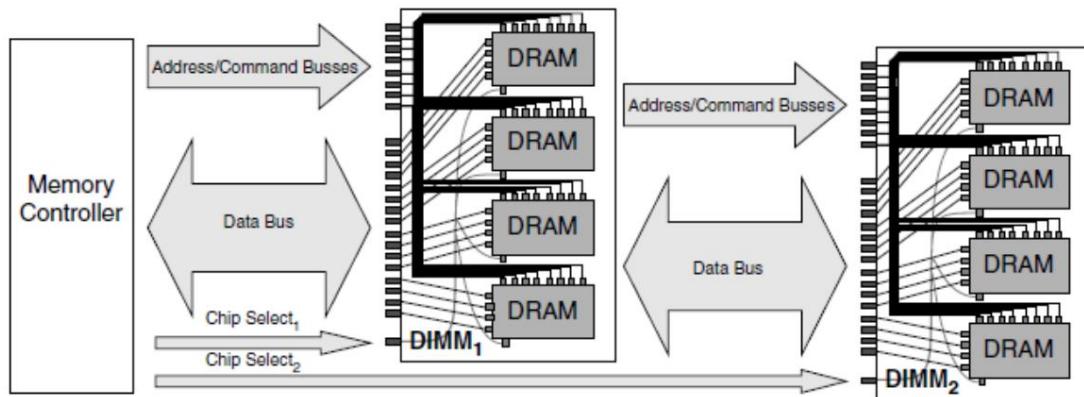


76. Irudia. Sailkapena: DIMM memoria-moduluan txip taldekatzea

## MAILA:

- 4 DIMM modulu bakoitza 4 txiprekin.
- Txip-bankuak memoria-SISTEMAN taldekatzea (memoria-modulu guztiak DIMM).

- Kasu honetan, Rank bakoitzak modulu bereko txip guztiak eta, beraz, banku guztiak multzokatzen ditu txip bakoitza.
- Aukeratutako txip bakoitzak maila ezberdin bat hautatzen du.

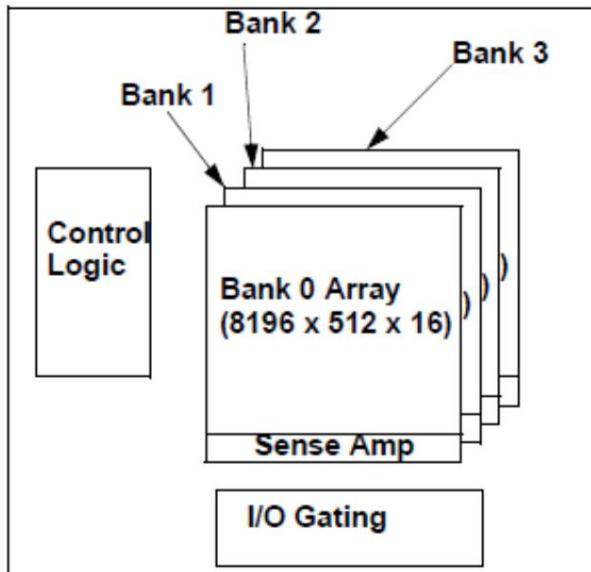


77. Irudia Rank

## Txipa

- Txipa:
  - ŷ Zelulak implementatzen diren erdieroaleen matrizea duen zirkuitu integratua da memoria (kondentsadoreak) eta etengailuak (transistoreak).
  - ŷ Txiparen pin kopurua emandako datuen tamainaren eta edukieraren araberakoa izango da datuak biltegiratzea.
  - ŷ Txipak I/O buffer-aren bidez emandako datuen "N" bit kopurua adierazten da txipa xN dela esanez: x2,x4,x8,x16,x32.
  - ŷ Hainbat banku, I/O buffer, errenkada demux, zutabe demux eta kontrolaren logika.

# Bank



78. Irudia Chipx16

- 4 bankuz eta 16 biteko I/O buffer 1ez osatua dagoen Chipx16.
- Banku bakoitzak 16 2D array dira ý 16.a 3D da, eta 8196x512 2D array bakoitza.
- Array bakoitzak 8192 errenkada eta 512 zutabe ditu. 213 errenkada x 29 zutabe = 222 gelaxka
- 3D banku bakoitzaren edukiera: 222 gelaxka x 16 bit =  $4 \times 220 \times 2 \text{ bytes} = 8 \text{ MBytes}$
- 3D banku bakoitza 8MB helbide-espazio bereizia da.
- Chipx16: 4 banku: 32 MB ý 16M x2byte DRAM ý 16M x16bit DRAM  
ý datu-busa: 16 bit ý 2 byte ý helbide-  
busa: 32 MB-ko edukierarako: 225 ý 25 hari

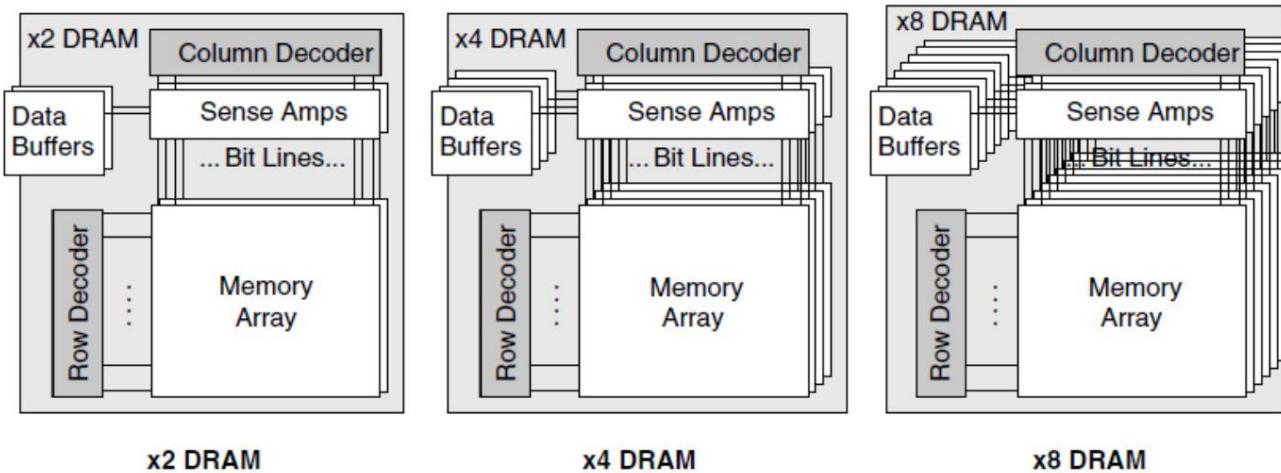
## bankua

- Termino hau nahasia da, testuinguruaren araberakoa baita eta interpretazio desberdinak ere badaude.  
ý Intel 4004 CPUaren kasuan, hau da... (ikus CPU Gaiaren i4004 atala)  
ý Linux ingurunean, "lshw -C memoria" komandoa exekutatzen badugu ezberdinei buruzko informazioa  
bankuak, non ingurune honetan bankua plaka amaren memoria zirrikituarekin lotuta dagoen.
- Bankua:  
ý Txipa bankuetan egituratzen da.  
ý Bankuak logikoki multzokatzen dira ordena edo banku-zenbaki bera lotzen duten Ranketan  
txip desberdinak.

↳ Bankua multzo bat da, 2D matrizeen multzokatzea, 2D matrizeak taldekatuz egitura bat dugu 3D.

↳ 3D egitura ~~marrazten kontuan kopurua~~ (errenkada-zutabea) datu bat dugula.

- Array bakoitzak bit bat laguntzen badu i/o buffer-ari, orduan bankuko array kopurua i/o buffer-eko bit kopuruaren berdina izango da.
- Modu honetan antolatuta, txip ezberdinaren banku-ordena bereko matrizek BATEAN erantzun diezaiekete helbide-bus berari (errenkadak, zutabeak). Bankuko array bakoitzak aukeratutako gelaxkaren bita ematen du, beraz, bankuak emandako datuen bit kopurua bankuko array kopurua izango da.
- Txiparen banku **guztiak** txiparen I/O buffer beraren parte dira .
- "n" bit kopurua , bankuak i/o buffer-aren bidez emandako datuetatik, txipa xn dela esanez adierazten da: x2,x4,x8,x16,x32
- Txip ezberdinaren bankuak elkarrengandik independenteak dira, memoria helbide ezberdinekin, helbide-bus bera partekatzen dute eta txip-bus hautatzen dute. Banku bat, gainontzekoekiko INDEPENDENTE izanik, aurrekargatu, freskatu, aktibatu, etab. maila bereko gainontzeko bankuen aldi berean: PARALELISMOA Banku mailan, DRAM memoriarako sarbide-denborak murrizteko helburuarekin. .



79. Irudia Bankua: DIMM memoria txiparen array-a

Hiru txip. Txip bakoitza BANKU batek osatzen du . Bankua Arrays talde bat da.

Txipa x2 DRAM:

- x2-k 2 biteko I/O buffer bat duen txipa esan nahi du ↳ 2 biteko datu-busa

- 2 array dituen banku bat. Array bakoitzak 1 bit eskaintzen du.

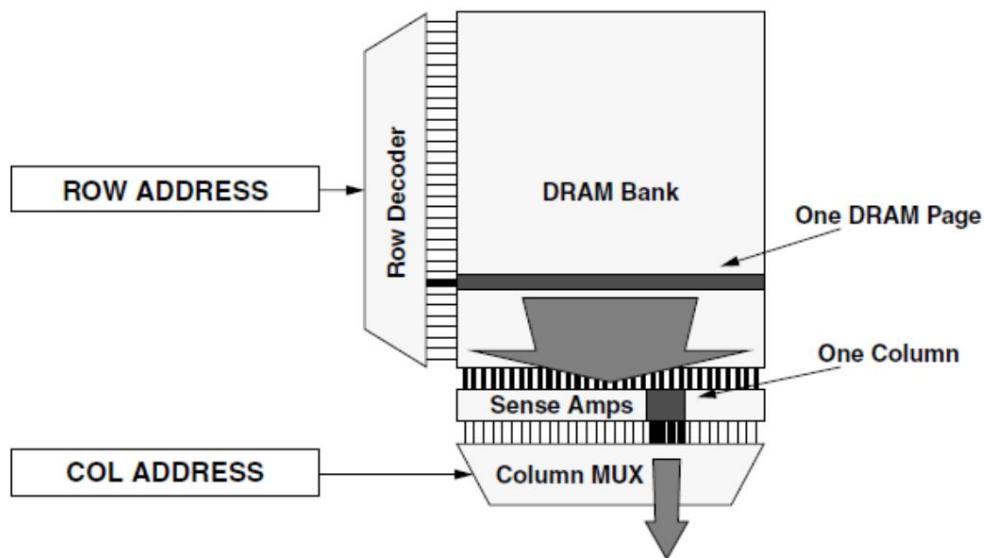
x4 DRAM txipa:

- x4-k 4 biteko I/O buffer bat duen txipa esan nahi du ý 4 biteko datu-busa
- 4 array dituen banku bat. Array bakoitzak 1 bit eskaintzen du.

x8 DRAM txipa:

- x8 esan nahi du 8 biteko I/O buffer txipa ý 8 biteko datu-busa
- 8 array dituen banku bat. Array bakoitzak 1 bit eskaintzen du.

#### Orria/Rankada-Buffer



80. irudia. Orrialdea

- Ez nahastu behar Sistema Eragilearen orrialde birtualekin
- Ordena bereko Banku multzo bakoitzak (adibidez, 3. Bankua) Rank bereko txip ezberdinenean multzo bakoitzak "erregistro birtual" bat dauka, ROW-BUFFER izenekoa. Birtuala edo logikoa da, txip fisiko desberdinak estaltzen baititu.
- Orrialde batek ordena edo banku-zenbaki berdina duten banku ezberdinenean errenkada-helbide bereko (Pej zenbakia 2) datuak (errenkada bera, zutabe bera bankuaren matrize ezberdinien) biltzen ditu (Pej zenbakia 3). Banku guztietako errenkada multzoari **orrialde** deitzen zaio (orri bat errenkadaz osatuta dago). Beraz, ROW-BUFFER-aren edukia orrialde batena da. Behar baino zutabe gehiago deskargatzen dira errenkada-bufferera, modu honetan errenkada-bufferak CACHE (kopia) gisa funtzionatzen du eskatutakoaren ondoan dauden datuak gordetzeko, baina hasieran beharrezkoa ez dena. Ondoren, ordena bereko zutabe guztiak (adibidez, 2. zutabea) errenkada-buffer-eko errenkada bakoitzeko zutabe guztiak hautatu eta i/o bufferean kargatzen dira. Hurrengoan

memoriarako sarbidea, behar diren datuak bufferean egon daitezke dagoeneko, beraz EZ da beharrezkoa matrizeko gelaxketatik datuak deskargatzea eta, beraz, latentzia murrizten da eta datuen irakurketa askoz azkarragoa da. Adibidez, array bat irakurri nahi izanez gero, array orri batean gordeta egon bada, orrialde osoa irakur liteke gelaxketarako sarbide bakar batekin eta, ondoren, buffer-aren edukia leherketatan transferitu.

#### array

- Array:

- ü Ilara eta zutabeetan antolatutako gelaxka multzoak edo multzoak dira.
- ü Memoria-helbide batek (lerroa, zutabea) matrizeko gelaxka bat hautatzen du.

#### Zelula

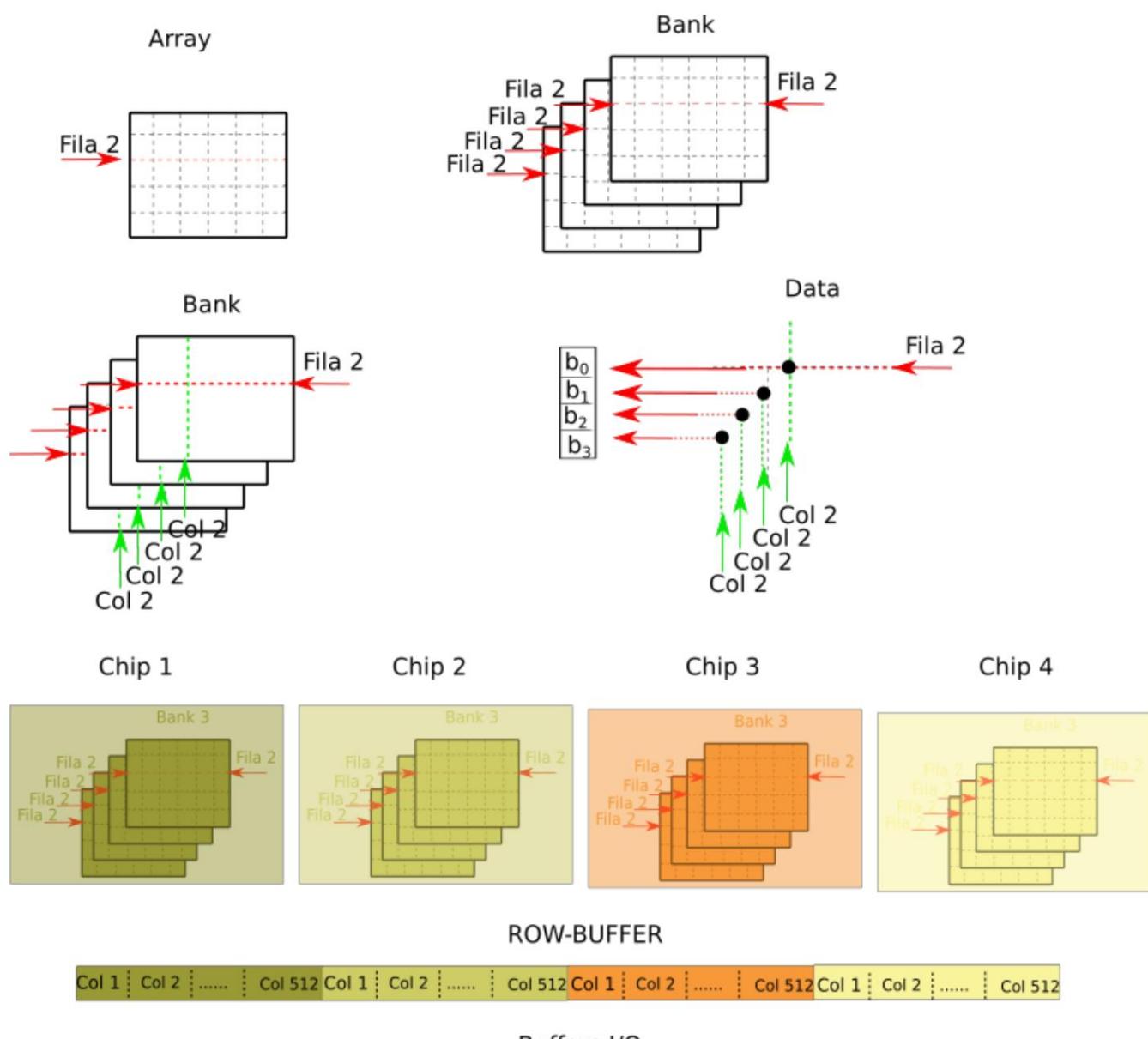
- Zelulak:

- ü Memoria gelaxkak 1 edo gehiagoko (2,4,8,16) biteko informazioa gordetzen du. Hasieran, berriz, ez Bestela zehaztuta, bit bakarra gordeko du.

#### MP memoriatik hitz bat IRAKURTZEA

##### Txipa/Bankua/Errenkadak/Zutabeen hautaketa

- Irakurketa adibidea: 4-chipx4-ko memoria txip bakoitzeko 8 bankurekin eta 1024x512 4 1024x512 matrizeko bankuekin bat. Memoria 32 bankuko 1 mailatan antolatuta dago.
  - to. Memoria kontrolatzaleak helbide fisiko bat jasotzen du espazio linealean eta Rank/Bankua/Errenkada/Zutabe-espazioarekin mapatzen du.
  - b. Maila-zenbaki bat hautatzen da: (adib. 1)
  - c. 1 eta 8 arteko banku-zenbaki bat hautatzen da: (adib. 3)
  - d. 1 eta 1024 arteko errenkada bat hautatzen da: (adib. 2)
  - eta. Maila bereko (adib. 1. zenbakia) banku guztiak (adib. 2. errenkada) (adib. 3. zenbakiaren bankua) aukeratzen dira.
  - F. aukeratutako errenkadaren (2. errenkadaren) (3. bankua) txip (txip) bakoitzaren zutabe guztiak (512 zutabe) datuak (4 bit: b3b2b1b0) ROW-BUFFER-era isurtzen dira.
  - g. Zutabe-zenbaki bat hautatu da (adib. 2)
  - h. 2 ordenako zutabe guztiak 4 i/o bufferretara husten dira (txip bakoitzeko bat). Datuak 4bit/zutabea x 4txip = 16 bit
  - Yo. 4 I/O bufferen irteerak (16 bit) datu-busen lineak (16 hari) aktibatzen ditu.



81. Irudia. Row-Buffer-eko orrialdea: 1. zenbakia/3. bankua/2. errenkada

#### Aurrekargatu/OpenRow

- Faseak:

ÿ CPUak emandako memoria helbidea helbide fisiko bihurtzen du zirkuituak  
MMU

ÿ MP zirkuituak memoria fisikoaren helbidea kodeetan deskonposatu behar du:

ÿ MAILA-BANKUA-ERRENKADA-ZUTABEA

ÿ Kodeak hauekin lotuta daude: memoria moduluaren barruan maila zehatz bat, mailaren barruan banku zehatz bat, bankuaren barruan errenkada zehatz bat eta errenkada barruan zutabe zehatz bat.

ÿ Maila-banku-errranka identifikatu ondoren, banku-bit\_lerroak AURREKARGATUTA daude (zero logiko baten eta bat logiko baten arteko batez besteko tensioarekin polarizatuta daude).

ÿ Bankua aurrez kargatuta dagoenean, errenkada IREKIA da: errenkada irekita geratzen da karga-sentsoreko milaka amplifikadoreek bankuko array guztien aukeratutako errenkadetan dauden milaka zelulen edukia detektatzen dutenean. Orria irekita dago amplifikadorearen irteerek hautemandako balioak berreskuratzen dituztenean eta biltegiratutako datuekin line\_bit aktibatzen dutenean.

ÿ Ekintza hau /RAS seinalea aktibatu eta tRCD denboraren zain hasten da

ŷ tRCD-a igarotakoan, bankuko array guztieng zutabe zehatzak hautatzen dira.

(x4,x8,...) eta I/O buffera hautatutako daturekin kargatzen da.

ŷ Ekintza hau /CAS seinalea aktibatzen hasten da.

### Adibidea

- Sistema batek 16 GB-ko memoria-ahalmen nagusia badu eta 4 modulutan egituratzen badugu, zeinen txipx16 4 mailatan antolatuta dauden 16 txip/rank, 8 banku/txip, 16 array/banku. Kalkulu bit/matrize kopurua.

ŷ  $2^4 \times 230 \times 23$  bit/byte = 22 (maila/kanal)  $\times 24$  (txipak/maila)  $\times 23$  (bankuak/txipa)  $\times 24$  (matriz/bankua)  $\times N$  (bit/matrize) ŷ  $N = 2(4+30)+3-2-4-3-4 = 2^{24}$  bit 2D array batean antolatuta

ŷ 4 modulu osoak 4 mailatan antolatuta daude

- Soluzio posible bat 212 errenkada  $\times$  212 zutabe izango litzateke.
- Memoria-busera datuak transferitzeko I/O buffer-ak  $\times 16$  tamaina du, hau da, 16 bit.

### 9.3.3. DRAM memoriaren antolaketa aurreratua

- [Eranskinean](#) informazio zehatza

#### SDRAM (DRAM sinkronikoa)

##### Sarrera

- Erlojuaren ertza eragiketen hasierako eta amaierako eredu da

- **DDR (Datu-tasa bikoitza)**

ŷ Bitak erlojuaren **beheranzko** eta **gorako** ertzetan transferitzeko aukera ematen du (**ponpaketa bikoitza**)

- I/O buffer maiztasuna

ŷ Memoriaren I/O buffer-a x2, x4 eta x8 maiztasunetara joan daiteke memoriarako sarbide-maiztasunari dagokionez. zelula.

ŷ **Superzelula** edo **Makrozelula**: orain matrize baten hautapen batek (errenkada, zutabea) ez du gelaxka 1 hautatzea, baizik eta matrizeko 2, 4 edo 8 zelula hautatzea.

ŷ DDR1: 21 zelula dituen makrozelula ŷ 2 zelula

ŷ 1. Belaunaldia: 2000. urtea

ŷ DDR2: 22 zelula dituen makrozelula ŷ 4 zelula

ŷ 2. Belaunaldia: 2006. urtea

ŷ DDR3: 23 zelula dituen makrozelula ŷ 8 zelula

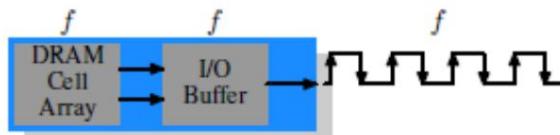
ŷ 3. belaunaldia: 2011. urtea

ŷ DDR4: 24 zelulako makrozelula ŷ 16 zelula

ŷ 4. belaunaldia: 2014. urtea

ŷ DDR5: 25 zelula dituen makrozelula ŷ 32 zelula

ŷ 5. belaunaldia: 2020. urtea



82. Irudia DDR transferentzia-abiadura

Macrocell ѕ I/O Buffer

- Zelulek (kondentsadoreek) oso karga gutxi gordetzen dute (picocoulombs) eta, beraz, ezin dira zuzenean helbide-busera konektatu, amplifikadorea edo CHARGE sentsore bat beharrezko da. Amplifikadore horren irteeran sarrera/irteera I/O erregistroa edo erregistroak daude (BUFFER). Hau da, zelulen eduki bitarra BUFFER I/Ora isurtzen da. I/O buffer-a plakaren DATU bus-lerroak gidatzeko gai den teknologia batean egina dago.
- I/o erregistro bateko bit bakoitzak gelaxka bakar baten edukia gordeko luke. Beraz, makrozelulak irakurtzeko, erregistro bat beharrezko da makrozelularen gelaxka bakoitzeko. Erregistro multzoari BUFFER deitzen zaio. DDR3 memoria batek, beraz, 8 Erregistroko i/o buffer bat izango du eta erregistro bakoitzean hitz bat gordetzen da. Memoria-txipa x16 bada, 16 biteko hitzak gordeko dira erregistro bakoitzean.

ÿ "movw sum,%ax" bezalako instrukzio bat programatzen dugunean, DDR3 memoria baten MC memoria kontrolatzalea ez da 2 byteko batura datuak soilik irakurtzera mugatzen, baina gutxienez 8 hitz jarraian irakurtzen ditu eta aldi berean irakurtzen ditu sarbide batean. datu-busera konektatutako I/O buffer-ean gordailutuz soilik.

- I/o buffer-ak datu orri bat gorde dezake. Buffering-ak aukera ematen du atzemateko datuak dagoeneko memoria-irteeran egotea zeluletarra sartu beharrik gabe. Laburbilduz, buffera harrapatzen ari diren datuetatik hurbil dauden gelaxken cachea (kopia) da. Horrela, memoriarako sarbide-denborak murrizten dira, I/O bufferra soilik sartzen baita benetan.

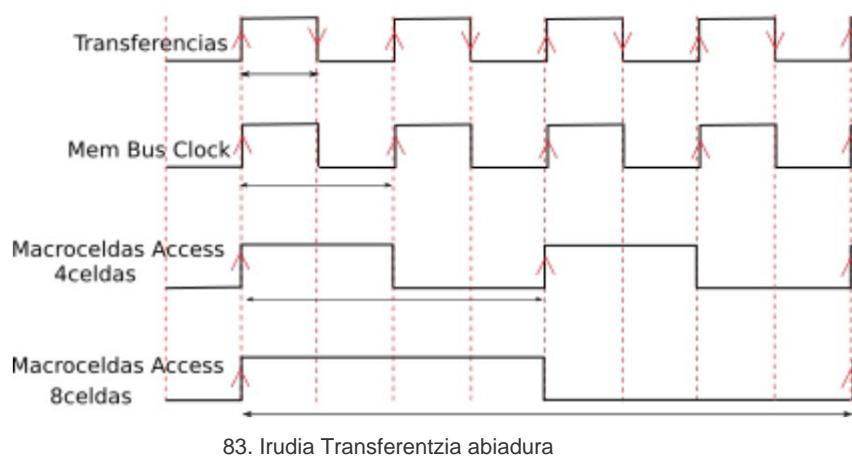
#### Transferentzia abiadura

- BW (bit/s) = BF (ziklo/s)\*CW (bit/kanal)\*TC (transferentzia/ziklo)
  - ÿ FE: maiztasun eraginkorra: transferentziaren maiztasuna
  - ÿ BF: Sistema-busen maiztasuna (1GHz-tik gertu 2000. urtean)
  - ÿ CW: kanaleko datu-busaren bit kopurua. Normalean 64 bit (2000. urtea) ѕ x86-64 arkitektura
  - ÿ TC: sistema-busaren erlojuaren ziklo batean transferentzia kopurua. Normalean 1 transferentzia (gorantz ertzak) edo 2 transferentzia (gorantz eta jaitsiera ertzak) BF autobusaren erloju-ziklo bakoitzeko.

ŷ BW (bit/s) = maiztasun eraginkorra\* datu-busaren zabalera=  $400\text{MHz} \times 2^*64 = 51200 \times 10^6$  bit/s = 51.2Gbps  
 = 6400 MBps ŷ sistema hamartarra (ohikoa)

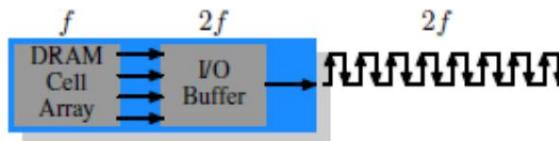
ŷ NC: zelula/makrozelula kopurua

ŷ CA: zelula sartzeko maiztasuna = FE/NC



Memoria-moduluaren adibidea: PC2-6400 (DDR2-800) 5-5-5-16

- PC2-6400 modulua (DDR2-800) 5-5-5-16
- PC2 : Bigarren belaunaldiko SDRAM ŷ Double\_Data\_Rate x2 ŷ 4 zelula makrozelulak.
- 6400 MB/s banda-zabalera edo abiadura ŷ BW= 6400 MB/s
- FE=800MHz sistema eraginkorra bus erloju-zikloa: bus maiztasuna BF ŷ bikoitzu  $800\text{MHz}=2^*\text{BF}$
- ŷ DDR: BF ziklo bakoitzean bi transferentzia egiten dira ŷ TC=2
- ŷ 64 biteko hitz bakoitza 800MHz-ko ziklo batean transferitzen da ŷ CW=800MHz
- ŷ Memoria Bus Erlojuaren Zikloa  $\text{BF}=800\text{MHz}/2$  ŷ  $\text{BF}=400\text{MHz}$
- ŷ Erlojuaren ziklo-denbora =  $1/400\text{MHz} = 2,5\text{ns}$
- 5-5-5-16 tCL-tRCD-tRP-tRAS ŷ 12.5ns denboraren erloju-zikloak dira ( $400\text{MHz} \times 2.5\text{ns}$ ).  
 $12,5\text{ns}-12,5\text{ns}-40\text{ns}$
- DDR2 ŷ NC=22 ŷ 4 zelula makrozelula bakoitzeko
- ŷ CA=FE/NC=800MHz/4=200MHz ŷ makrozeluletarra sartzeko maiztasuna



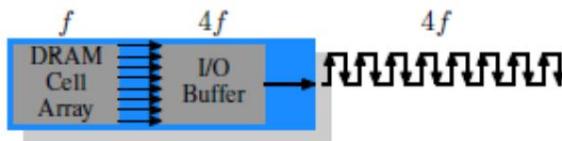
84. Irudia DDR2 transferentzia-abiadura

Memoria moduluaren adibidea: DDR3-800 / PC3-6400 5-5-5

- DDR3-800 edo PC3-6400 memoria modulua
- ŷ denbora 5-5-5
- ŷ 800MHz datu-busaren maiztasun eraginkorra da ŷ 800MT/s
- ŷ 6400 MB/s banda zabalera da
- ŷ DDR ŷ Memoria-busaren maiztasuna maiztasun eraginkorren erdia da =  $800/2 = 400\text{MHz}$ .

$1/400\text{MHz} = 2.5\text{ns}$ -ko erloju-ziklo baten balioidea

ÿ 5-5-5: erloju-zikloak dira, 400MHz-ko busaren maiztasun errealean, denbora-tCL parametroen tRCD-tRP



85. Irudia DDR3 transferentzia-abiadura

13. taula. DDR3

estandardaryam	Memoria-erlojua (MHz)	Ziklo-denbora (ns)	I/O buseko erlojua (MHz z)	Datu-tasa (MT/s)	Moduluayam	Gailurra (transferentzia tasa (MB/s tRP) )	Ordutegiak CL-tRCD	CAS latentzia (ns)
DDR3-800D 100		10	400	800	PC3-6400 6400		5-5-5	12½
DDR3-800E							6-6-6	latency

4. zifra tRAS da (aktibazio eta aurrekargaren arteko gutxieneko atzerapena) ez da eman. Laugarren zutabeak tCL nanosegundotan ematen du.

- Parametroak:

ÿ Memoria-erlojua: 100MHz: hitzak sartzeko maiztasuna. Zelula transferentzia ÿ i/o buffer

ÿ Ziklo-denbora: 10ns: taula honetan memoria-erlojuaren periodoari dagokio eta ez du esanahirik. memoria zikloaren definizioa

ÿ I/O bus-erlojua: 400MHz: memoria-bus-erlojua zeinaren ertzak (positiboak, negatiboak) hitz-transferentziak sinkronizatzen dituen.

ÿ Autobusaren zikloa =  $1/400\text{MHz} = 2.5\text{ns}$  = hau da atzerapenen edo latentziaren denbora faktorea tCL,tRCD, etab

ÿ Datu-tasa: 800MT/s ÿ Transferentziak maiztasun eraginkorrean egiten dira.

ÿ Transferentzia-tasa gailurra: BW banda-zabalera: 6400MB/s

ÿ ordutegiak: I/O bus-erlojuaren ziklo-kopurua eta ekitaldien iraupena: 5-5-5-12½

Ÿ tCL = 5 erloju-ziklo =  $5 \times 2,5 = 12,5\text{ns}$   
 Ÿ tRCD = 5 erloju-ziklo =  $5 \times 2,5 = 12,5\text{ ns}$   
 Ÿ tRP = 5 erloju-ziklo =  $5 \times 2,5 = 12,5\text{ns}$   
 Ÿ tRAS = ez da eman

#### Adibidea PC3-22400 11-14-14-35

- Dominator® Platinum Corsair Link konektorearekin — 1,65 V 16 GB kanal bikoitzeko DDR3 memoria kit (CMD16GX3M4A2800C11):
- Memoria mota: DDR3
- Abiadura-balorazioa: PC3-22400 (2800MHz)
- Probatutako Latentzia: 11-14-14-35
- Gure Prezioa: 80€
- 16 GBko kit (4 x 4 GB)
- Kanal Bikoitza
- Deduzitutako ezaugarriak:

Ÿ Banda zabalera gailurra = 22400MB/s  
 Ÿ Datu-tasa (1data=8bytes) = 2800MT/s  
 Ÿ I/O busaren erloju eraginkorra = 2800MHz. I/O memoria I/O buffer busari egiten dio erreferentzia.  
 Ÿ I/O busaren erlojua = 2800MHz / 2 = 1400MHz  
 Ÿ I/O busaren ziklo-denbora = 1/1400MHz = 710ps  
 Ÿ Latentzia

Ÿ tCL = 11 ziklo =  $11 \times 710\text{ps} = 7,8\text{ns}$   
 Ÿ tRCD = 14 ziklo =  $14 \times 710\text{ps} = 10\text{ns}$   
 Ÿ tRP = 14 ziklo =  $14 \times 710\text{ps} = 10\text{ns}$   
 Ÿ tRAS = 35 ziklo =  $35 \times 710\text{ps} = 24.8\text{ns}$

- Hobetua PC3-22400 vs PC3-6400
- Ÿ I/O busaren ziklo-denbora hobetua = 710ps vs 2.5ns =  $1.79/2.5 = \% 71$

#### PC2-6400 eta PC3-6400 arteko aldea

- Ez dago alderik latentziari dagokionez, bi kasuetan 5-5-5 batek maiztasun bat aipatzen baitu. 400MHz-ko memoria-busa.
- Desberdintasunak daude pin, hornidura tentsio etab.

#### Banda zabalera estandarrak

- DDR1 SDRAM moduluak: PC-3200/PC-2700/PC-2100/PC-1600
- DDR2 SDRAM moduluak: PC2-6400/PC2-5300/PC2-4200/PC2-3200
- DDR3 SDRAM moduluak:
  - Ÿ PC3-22400/PC3-21300/PC3-19200/PC3-17066/PC3-15000/PC3-12800/PC3-10600/PC3-8500/PC3-6400
  - Ÿ DDR3-2800/DDR3-2666/DDR3-2400/DDR3-2133/DDR3-1866/DDR3-1600/DDR3-1325/DDR3-1065/DDR3-800/

## Gaitasuna

### Erregistratutako/Buffered Memoria

- [http://en.wikipedia.org/wiki/Registered\\_memory](http://en.wikipedia.org/wiki/Registered_memory)
- Erregistratua: RDIMM: Memoria kontrolatzalearen eta memoria moduluaren artean kontrol-lerroen informazioa memorizatzen duen erregistro bat dago. Kontrol-agindua transferentzia baino lehen bidaltzen da, autobus-ziklo gehigarri bat gehituz. Modu honetan, komandoak kontrolagailura transferitzeko kontrol-lerroak ezabatzen dira eta horrela memoria-kontrolagailuaren memoria-busaren karga murrizten da eta modulu gehiago konekta daitezke kontroladorearen kanalera, memoria-ahalmena handituz.
- Buffer gabea: UDIMM: Kontrol-lerroaren informazioa ez dago blokeatuta.
- guztiz buffered:
  - ÿ Bai kontrol-seinalearen informazioa eta bai datuak eta helbide-seinaleak memoria-kontrolagailuen kanal-bus guztietan karga nabarmen murriztuz grabatzen dira.
  - ÿ Datuak seriean transferitzen dira paraleloan beharrean lerro kopurua murriztuz eta, beraz kanalera konektatutako memoria moduluen kopurua handituz.

### Banku aldaketa

- [Banku-aldaketa](#)
  - ÿ 8 eta 16 biteko arkitektura muguetan (adibidez, Intel 4004) memoria bankuaren teknika erabiltzen da. memoria gaitasuna handitzeko.
  - ÿ Helbide-busaren zabalera handitu ordez CPUaren eta plaka-busaren hitzaren tamaina handituz, memoria-gailu helbideragarri gehiago gehitzen dira bus bera eta memoria-gailuetako bat (Bankua) hautatzen duen erregistro berri bat erabiliz. **Ez da nahastu** memoria txip-bankuekin edo erregistro-bankuekin.
  - ÿ Banku aldaketak memoria bankuak aldatzea esan nahi du.

### 9.3.4. Informazio Gehigarria

- Informazio osagarria Eranskinean .

### 9.3.5. Thinkpad T560

#### aginduak

- zerrenda

```
sudo lshw -short -C memoria
sudo lshw -class memory
lspci | grep -i mem -> memoria kontrolatzalea -> chipset
sudo inxi -m

sudo cpu-
x sudo dmidecode -t
memoria sudo dmidecode -t 16
```

```
lsmem free cat /
proc/meminfo https://access.redhat.com/solutions/406773
```

## analisia

- `sudo dmidecode -t 17`

```
DDR3
Transferentzia-tasa 1600 MT/
s kanala A
kanala B
banku-lokatzailea
bankua0 banku-lokatzailea bankua2
SODIMM DDR3 sinkronikoa 1600 MHz (0,6 ns)
-bankua:0 -> zirkitua: ChannelA-
DIMM0 -bankua:2 -> zirkitua: ChannelB-DIMM0
```

ÿ zirkitua: memoria zirkitua memoriarako entxufe bakoitza da

ÿ bankua: banku bat, memoria zirkituen multzo bat besterik ez da

ÿ kanala: kanala memoria-bus bat da, datuak CPUtik RAMera bidaiatzeko dedikatzen dena moduluak

- `sudo hwinfo | grep -i mem -A 10`

## 9.4. Cachea

### 9.4.1. Bibliografia

- William Stalling Liburua  
ÿ 4. kapitulua.

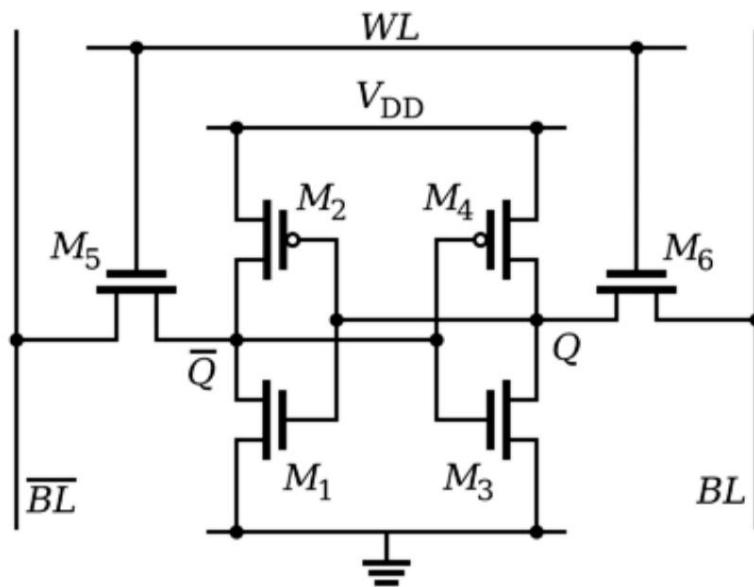
### 9.4.2. Sarrera

- Helburua
  - ÿ Toki-printzipioa: espaziala eta denborazkoa
- Ariketa: Sartzeko denbora (porrot edo arrakasta izateko probabilitatea)
- Teknologia: 6T
- Egitura
  - ÿ Kontrolatzalea: funtzioa
- Helbide-espazioak
  - ÿ memoria nagusia
  - ÿ cache memoria
- Helbide-espazioaren mapa-funtzioak:
  - ÿ Mapa zuzena
  - ÿ Elkarte osoa
  - ÿ Mitzoka elkartzea

### 9.4.3. Oinarrizko printzipioak

## Teknologia

- Zelula: SRAM-6T



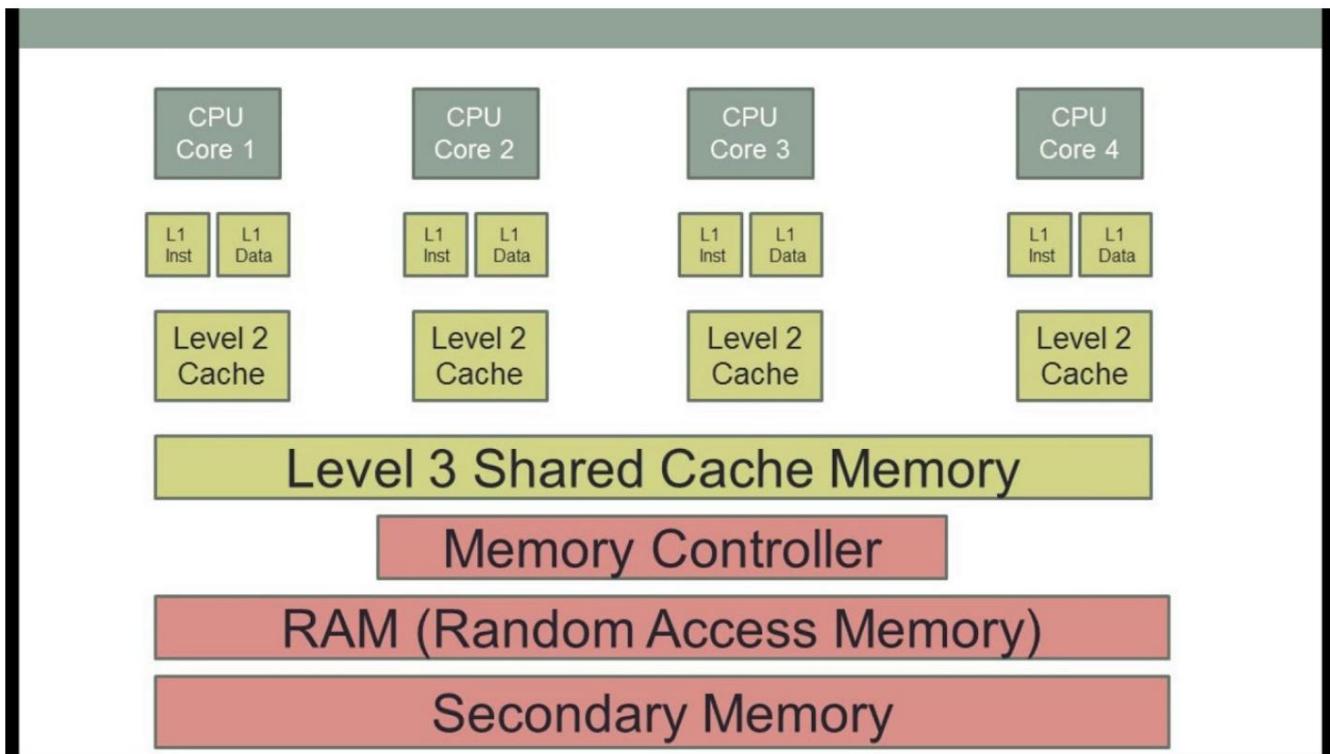
86. Irudia SRAM-zelula

## Funtzionalitatea

- von Neumann Bottleneck: instrukzio-zikloaren latentziaren latentzia baino askoz txikiagoa da.  
memoria nagusirako sarbidea y CPU hildako denborak.
- Cachea:
  - y PUZan integratua DRAM memoria baino magnitude ordena bat txikiagoa duen sarbidearekin.
  - y Memoria nagusiko eskualdeen edo blokeen kopia da.
  - y Hit-Miss: DRAM memoria nagusiaren helbideak seinalatzen dituen instrukzioa/datuak cachean badaude, CPUak instrukzioa edo datuak eskuratzten ditu barneko cache memoriatik, zeinaren sarbide-denbora DRAM kanpoko memoria baino askoz txikiagoa den CPUna.
  - y Hit-Miss
  - y Liburuko 4.1 adibidea
- Tokiko printzipioa
  - y Espaziala: begiztak, azpierrutinak, array y argibideak eta datuak helbideetan gordetzea hurrengoa.
  - y Denborazkoa: historikoa y hurbileko helbideetako datu eta argibideetarako sarbidearen sekuentzia.

## Hierarkia

- Cache-mailak
  - y L1 maila: PUZaren barnekoa: SRAM: instrukzioetarako memoria bereiziak eta datuetarako memoria
  - y L2 maila: PUZaren kanpokoa/barnekoa:
  - y L3 Maila: CPUtik kanpokoa

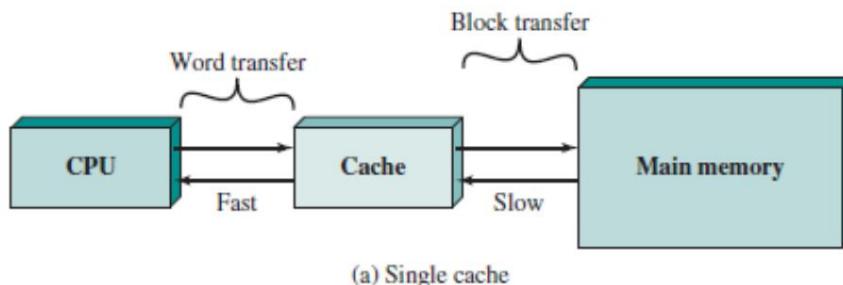


87. Irudia 3 Mailak

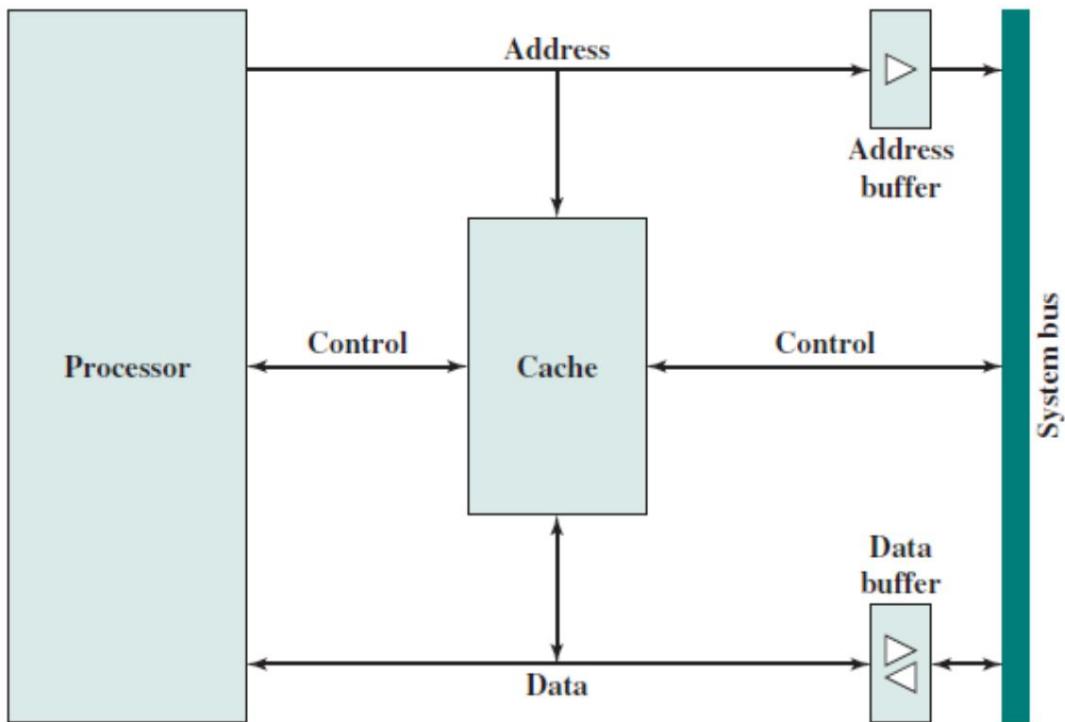
- SRAM cache-memorietako zeluletan transistoreen komutazio-abiadura handitzeko, beharrezko da haitatik zirkulatzen duen korrontea handitu eta, horretarako, transistore horien tamaina handitu behar da, Siliziozko azalera gehiago kontsumituz. Hau da abiadura handiko memoriaren ahalmenaren mugetako bat hamarnaka Kbyte gutxira. Muga hori hobetzeko, maila desberdinak abiadura ezberdinetan sortzen dira, L1 maila abiadura handiena izanik. Abiadura baxuagoko L2 eta L3 mailak transistore txikiagoekin fabrikatzen dira, haien ahalmena Mbyte-ren ordenara igotzeko aukera emanez.

#### Interkonexioa

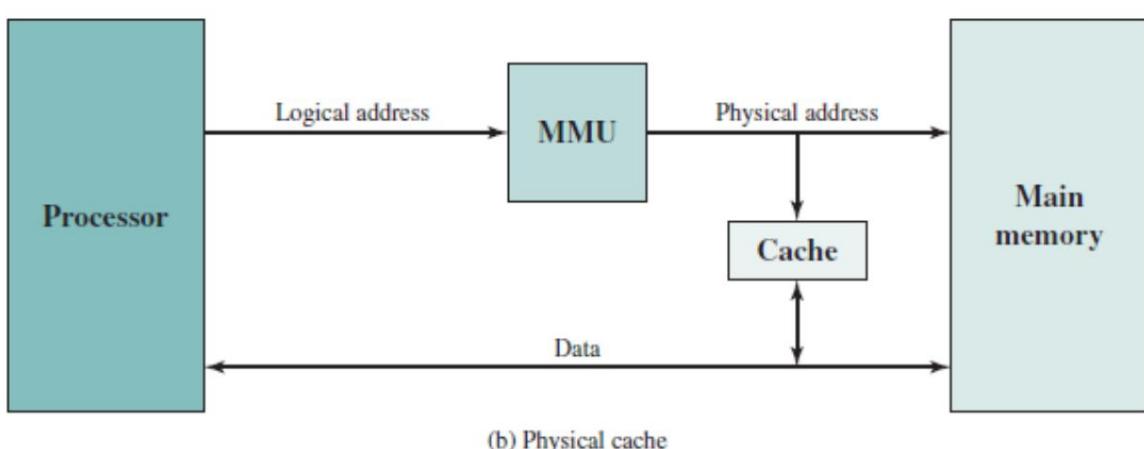
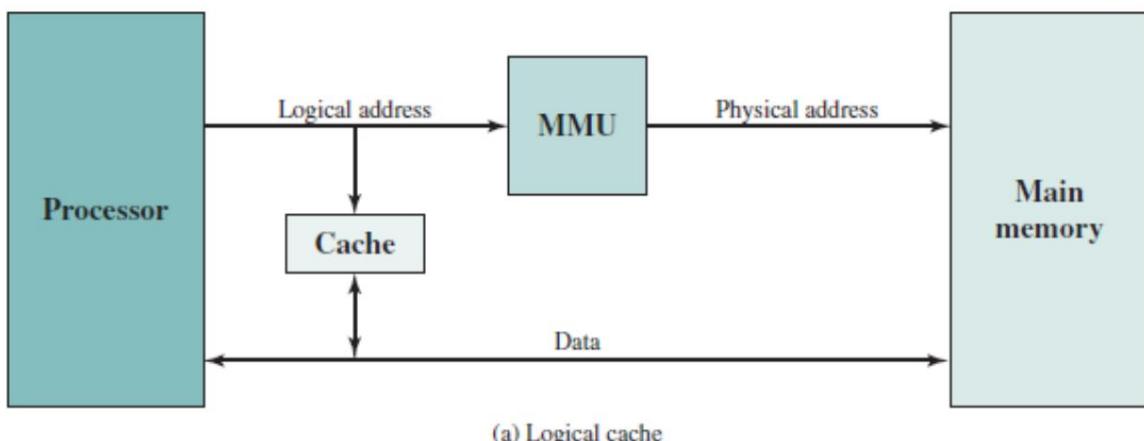
- Seriean CPU → L1 → L2 → SDRAM
- CPU → L : Hitz transferentzia
- L → SDRAM: blokeak



88. Irudia Interkonexioa

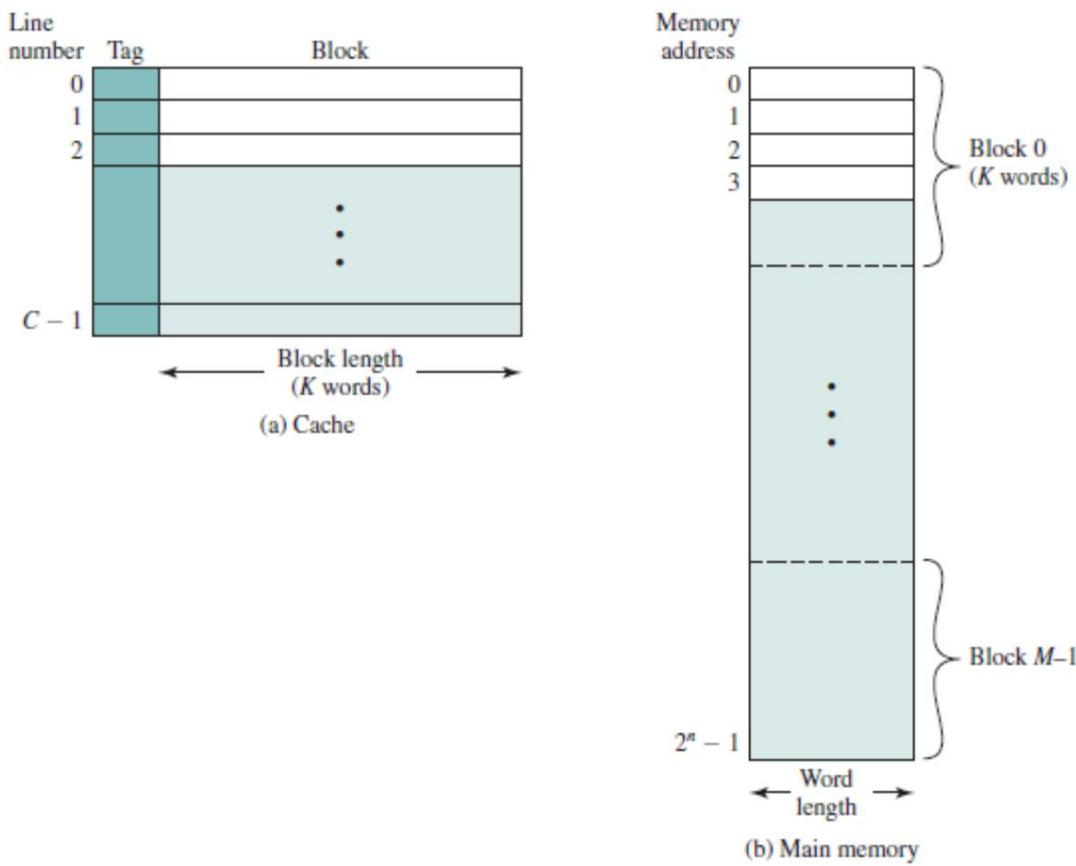
**Figure 4.6** Typical Cache Organization

89. Irudia Interkonexioa

**Figure 4.7** Logical and Physical Caches

90. Irudia MMU

## Cachea/Egitura nagusia

**Figure 4.4** Cache/Main Memory Structure

## 91. Irudia Clustering egitura

- Cachea: byteak ѕ hitzak ѕ lerroak
- MP: byteak ѕ hitzak ѕ blokeak
- Memoria Nagusiko blokeak Cache Memoriako lerroetan kopiatzen dira, beraz, bloke baten eta lerro baten tamaina berdinak dira.

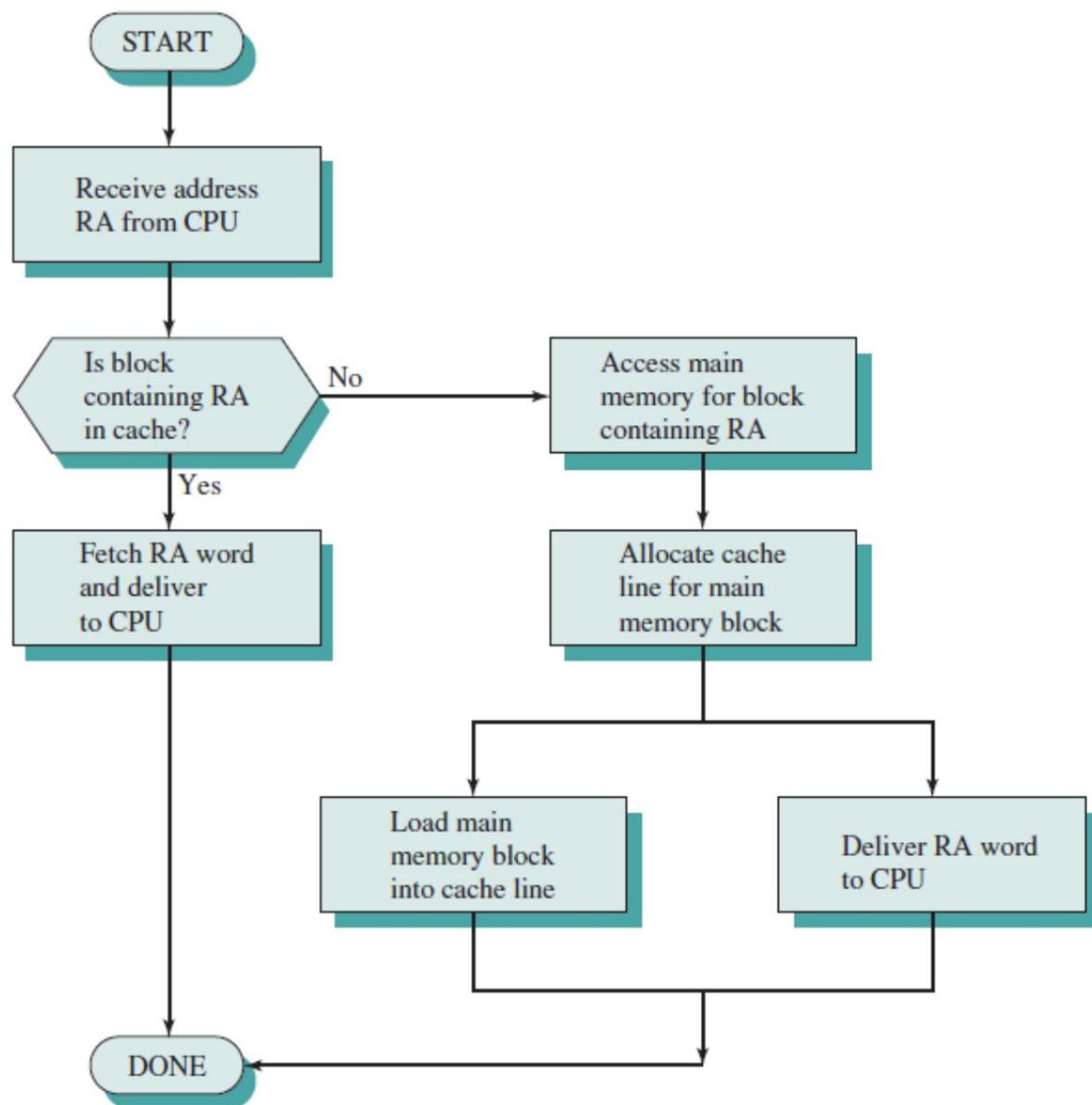
## Nagusia

- Helbidea: n bit: sistemaren helbide-busa ѕ Edukia=2n hitz ѕ OHARRA: byterik ez
- Helbideragarria: hitzak: sistemaren datu-busa
- K hitzen blokeetan antolatuta
- Bloke kopurua: Edukiera hitzetan / K

## Cachea

- Helbidea: Lerro zenbakia eta Hitza
- Helbideragarria: Word
- Edukiera: C Lineak
- K hitzen lerroetan eta etiketa 1etan antolatuta
- Etiketa: Lerro-elkarketa Memoria-Bloke Nagusiarekin

## Irakurri Eragiketa



92. irudia. Irakurketa eragiketa (RA: Irakurri helbidea)

- Diagrama operatiboa
- Cache kontrolatzailea

### 9.4.4. Cachearen diseinu-elementuak

- Elementuak:
  - ÿ Cachearen tamaina, bat etortzeko funtzioa, ordezkapen algoritmoak, idazketa politika, Lerroaren tamaina, cache kopurua

#### Tamaina

- Kontraesana: zenbat eta handiagoa, orduan eta motelagoa eta arrakasta izateko aukera gehiago.
- L1: KB
- L2: MB
- L3: MB

## Main\_Memory/Cache\_Memory Korrespondentzia Funtzioa

- Memoria Nagusiko blokeak Cache Memoriaren lerrokin lotzeko modu desberdinak.
- Blokea <ü Lerro korrespondentzia motak:
  - ÿ Zuzena (Mapea zuzena), Erabat elkartua (Asozialio osoa), Talde elkartua (Ezarpena elkartua)

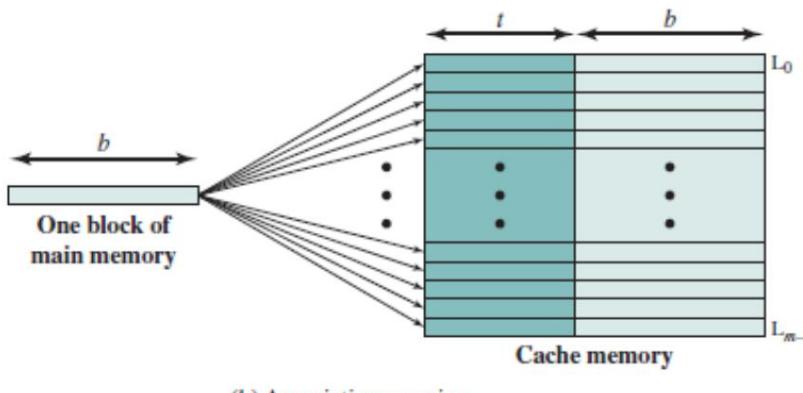
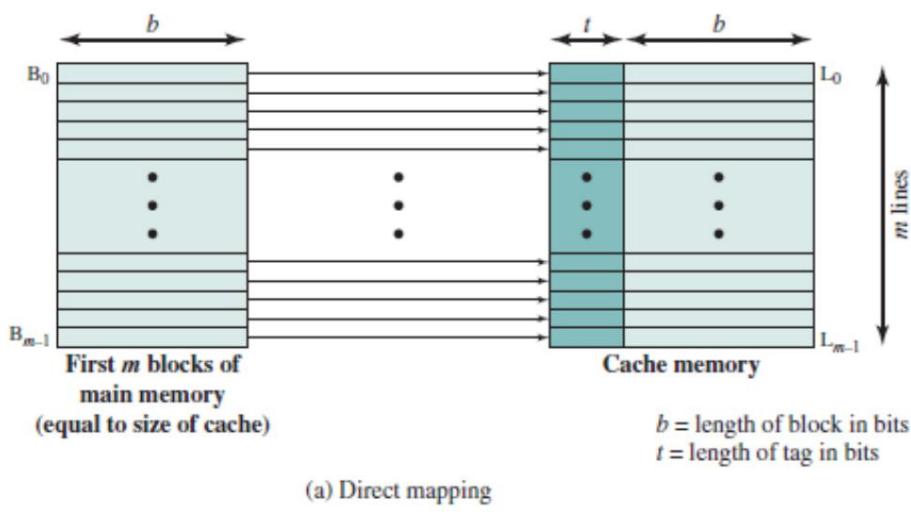
## Adibidea

- W.Stalling liburua. 4. kapitulua. 4.2 adibidea
- 3 kasuetarako adibidea:
  - ÿ m: cache-ahalmena  $64\text{ KB} = 4 * 214\text{ byte}$
  - ÿ MP:
    - ÿ hitzaren tamaina: 1 byte
    - ÿ hitzak/bloke = 4.
    - ÿ edukiera =  $16\text{MB} = 224\text{byte} = 224\text{byte} / (4\text{byte}/\text{bloke}) = 222\text{ bloke} = 4\text{M bloke}$

- Cachearen edukiera  $m = 4 * 214\text{ byte} / (4\text{byte}/\text{lerro}) = 214\text{ lerro} = 16\text{K lerro}$
- MP bloke bat Cache lerro baten baliokidea denez ÿ Cache/MP edukiera erlaziona da  $16\text{K}/4\text{M} = 1/(28)$

## Zuzeneko Elkartea

- Helbideen egitura
  - ÿ Memoria nagusia: hitz blokeak
  - ÿ Cache memoria: hitz lerroak
  - ÿ Memoria-bloke nagusi baten tamaina Cache-lerro baten tamainaren berdina da.
- Cache kontroladorearen korrespondentzia funtzioa:
  - ÿ deterministikoa - ez dago askatasun mailarik bloke bati dagokion lerroa aukeratzerakoan ziurra.
  - ÿ memoria nagusiaren helbide-lerroaren eremuak kontrolagailuak bilatzen duen hitza egon daitekeen Cacheko lerro bakarrari seinalatzen du. Memoria nagusiaren helbidearen etiketa-eremua Cache-lerroaren etiketa-eremuarekin bat badator, esan nahi du bilatutako hitza lerro horretan dagoela eta memoria-bloke nagusiaren kopia bat denez, cache-lerrotik atera daitekeela esan nahi du. "Hit" bat egon dela esan du. Huts egiten bada ("galdu"), Memoria Nagusiko hitza atzman beharko da, beraz, latenzia etiketa egiaztatzen emandako denbora gehi memoria nagusiko hitza sartzen emandako denboraren batura izango da.



**Figure 4.8** Mapping from Main Memory to Cache: Direct and Associative

93. Irudia Mapa Zuzeneko Egitura

- i: cache-lerroaren zenbakia
- j: memoria nagusiaren bloke-zenbakia
- m: cacheko lerro kopurua
- Lotura-funtzioa
  - ÿ  $i = j \text{ modulo } m$
- Cachearen antolaketa
  - ÿ Cache\_Memoria/Cache\_Controller/Main\_Memory

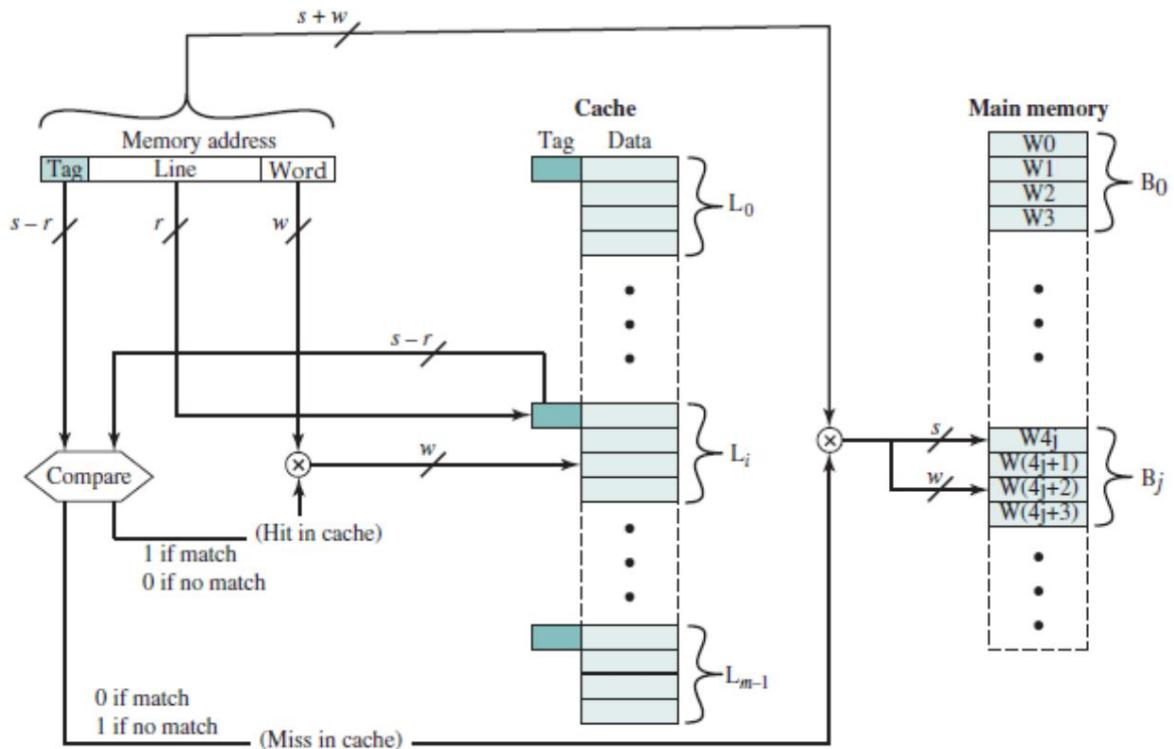


Figure 4.9 Direct-Mapping Cache Organization

94. Irudia Mapa Zuzeneko Kontrolatzalea

- Helbide-formatua
  - ÿ Memoria nagusiaren helbide fisikoa: bloke-hitza
  - ÿ Cache memoriaren helbide fisikoa: tag-line-word
- Bilatu eragiketa hitz bat cachean.
  - ÿ Zehaztu cacheko helbidearen formatuaren etiketa, lerroa eta hitz-eremuak.
  - ÿ Hitz esleitutako lerroan bakarrik egon zitekeen, beraz, konparatu beharra dago helbidearen formatuaren etiketa duen lerro horren etiketa soilik.

#### • 4.2 adibidea

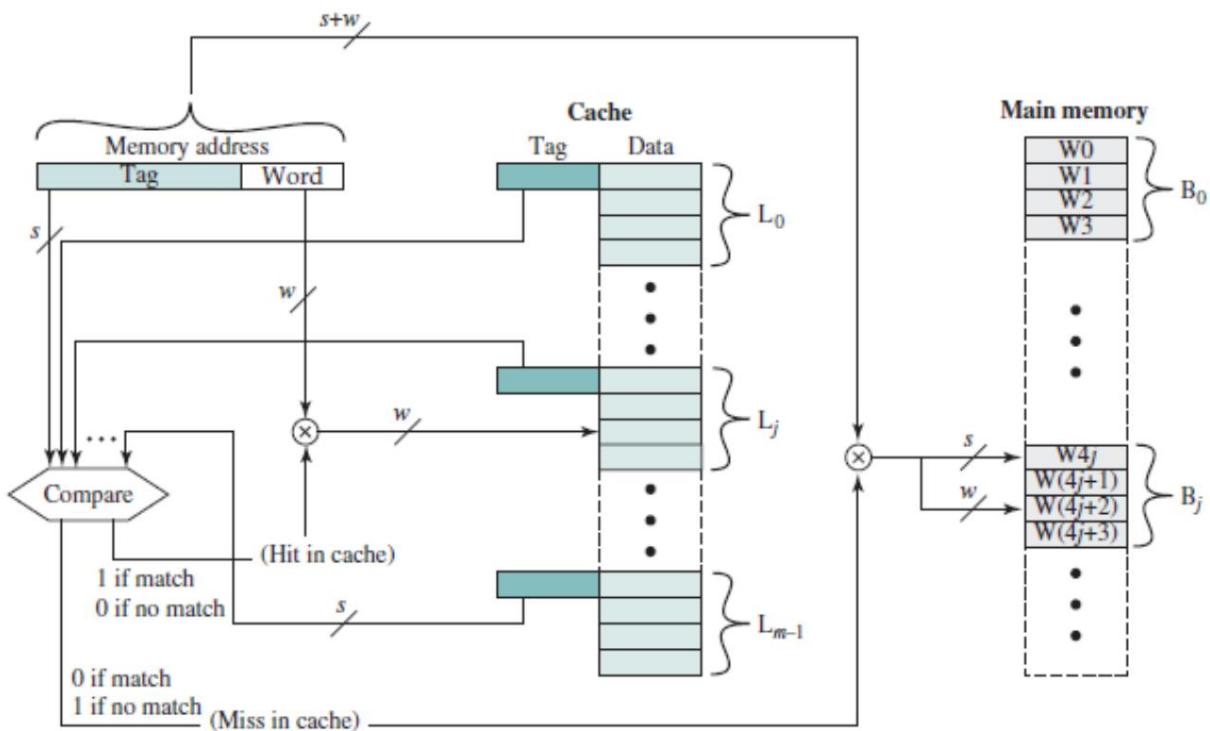
ÿ a) atala

ÿ Memoria-helbidearen formatua eremuetan:

$2^{24}$  hitzak  $\rightarrow$  2 biteko hitzen eremua  $2^{14}$  lerroak  
 $\rightarrow$  14 biteko lerroaren eremua gainerako bitak  
 $(24-14-2)=8 \rightarrow$  8 biteko etiketa eremua

ÿ ...jarraitu

Erabat Elkartzalea



**Figure 4.11** Fully Associative Cache Organization

95. Irudia "Kontrolatzaile elkartu osoa!"

- Korrespondentzia guztiz asoziatiboa esan nahi du ez dagoela korrespondentziarik
  - ÿ EZ DAGO ARAUA ÿ MP BLOKE BAKOITZA EDOZEIN LERROARI ESLEITU DAITEKE KATXEA
    - ÿ DOAKO: MP bloke batek ez du lerro zehatzik esleituta eta cache kontrolatzaileak bloke horri zein lerro esleituko zaion hauta dezake.
- Memoria helbideak bi eremu baino ez ditu
  - ÿ s bits= etiketa: MP blokearen ordena: 1etik 4Mra.
  - ÿ w bit= bloke barruko hitz-ordena: 1etik 4ra.
- cache kontrolatzailea
  - ÿ Cache-lerroen etiketa guztiak aldi berean erreferentziatutako hitzaren etiketarekin alderatzen dira.
    - ÿ Arrakasta denean, s erreferentziakoa hitza duen marra seinalatzen du eta w hitza erreferentziatuta.
    - ÿ hutsegite kasuan, s erreferentziakoa hitza eta w puntuak dituen MP-aren blokera seinalatzen du erreferentziakoa hitzari.
- Helbide-formatua
  - ÿ Memoria nagusiaren helbide fisikoa: bloke-hitzoa
  - ÿ Cache memoriaren helbide fisikoa: etiketa-hitzoa
- Bilatu eragiketa hitz bat cachean.
  - ÿ Zehaztu cacheko helbidearen formatuaren etiketa eta hitz-eremuak.
  - ÿ Hitza edozein lerrotan egon daiteke, beraz, guztien etiketak konparatzea beharrezko da lerroak
- Zuzeneko mapearekiko aldea:

ŷ etiketa eremuak s bit >> sr bit ditu

ŷ etiketa guzien arteko konparazioa (etiketa luzeak) ŷ hardware konplexua ŷ kostua

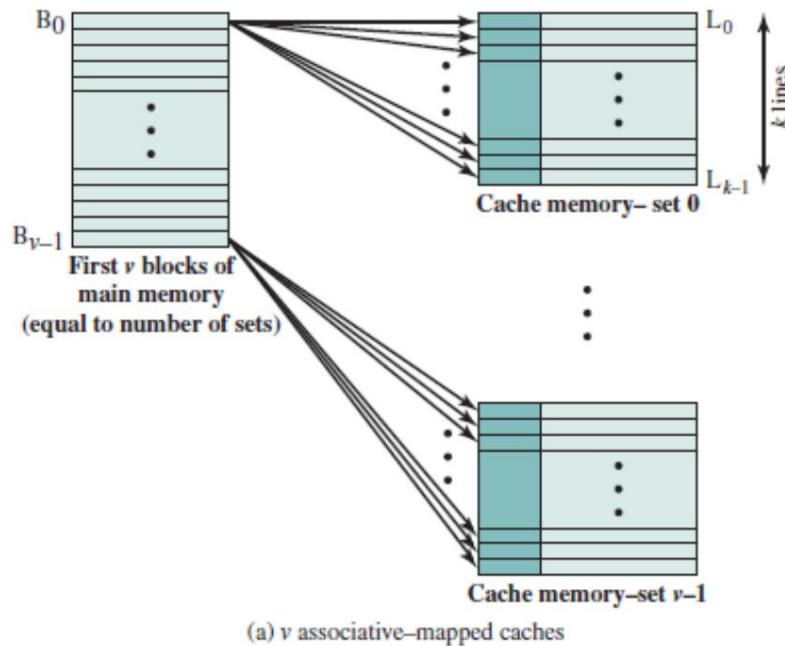
ŷ 4.2 Adibidea: Memoria-helbidearen formatua eremuetan:

$2^{2^k}$  hitzak -> 2 biteko hitzen eremua gainerako bitak

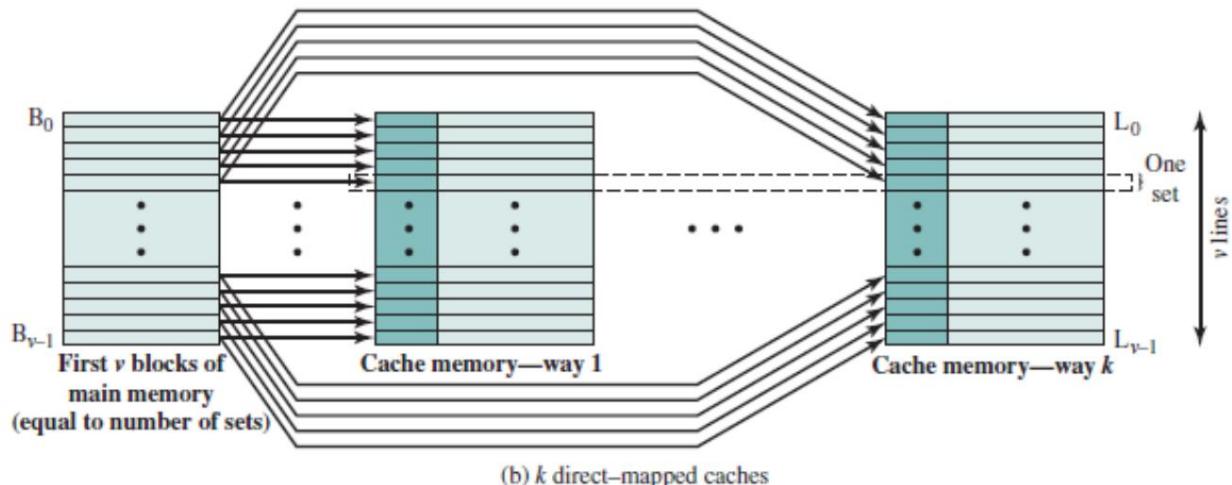
$(24-2)=22$  -> 22 biteko etiketa eremua

ŷ ...jarraitu

Taldeka elkartuta



(a)  $v$  associative-mapped caches



(b)  $k$  direct-mapped caches

**Figure 4.13** Mapping from Main Memory to Cache:  $k$ -Way Set Associative

- zuzeneko korrespondentziaren zorroztasunaren eta erabateko malgutasunaren arteko konpromisoa elkartua.

ŷ Memoria helbideak 3 eremu ditu: TAG-SET-WORD ŷ (sdw)/d/w

ŷ  $k$  lerro multzo bakoitzeko.

ÿ w bit: hitz ordena. 2w hitzekin bloke bat osatzen dut

ÿ d bit:

ÿ Bloke MULTZOA edo bloke MULTZOA edo SUPERblokea edo SUPERierroa.

ÿ d < l: cachea v superblokeetan banatzen dugu. ÿ k lerro/

blokeko v=2d superblokeekin cache memoria osatzen dut.

ÿ Superbloke bakoitzeko k lerro kopuruari VIA (WAY) deitzen zaio.

ÿ MP-aren egitura: bloke eta superblokeetan banatuta dago.

ÿ 2 <sup>hi</sup> MPren bloke-kopurua da, zeina k bloke-multzotan multzokatzen baditut MP 2t taldekatze bakoitzak 2d multzoko multzoka izango ditudan ÿ 2 s \*2w = 2t \*2d \*k\*2w ÿ 2 s=2t \* 2d \* k ÿ MP = bloke kopurua  
Talde kopurua X\*Multzo kopurua Cachean\*Multzoko lerroak

ÿ sd bits: etiketaren bit kopurua

ÿ parekatzeko funtzioa

ÿ i = j modulua v ÿ ARAU ERDI ZIGORRA: BLOKE BAKOITZAK MULTZO BAT ESLEITATU DU  
LERROAK ZEHATZATZEN DITUT BAINA LERROA EZ DAGO MULTZOAREN BARRUAN ESLEITZEN

ÿ non v superbloke-kopurua den, j bloke-zenbakia MP-n eta ei-ren kopurua  
superblokea cachean.

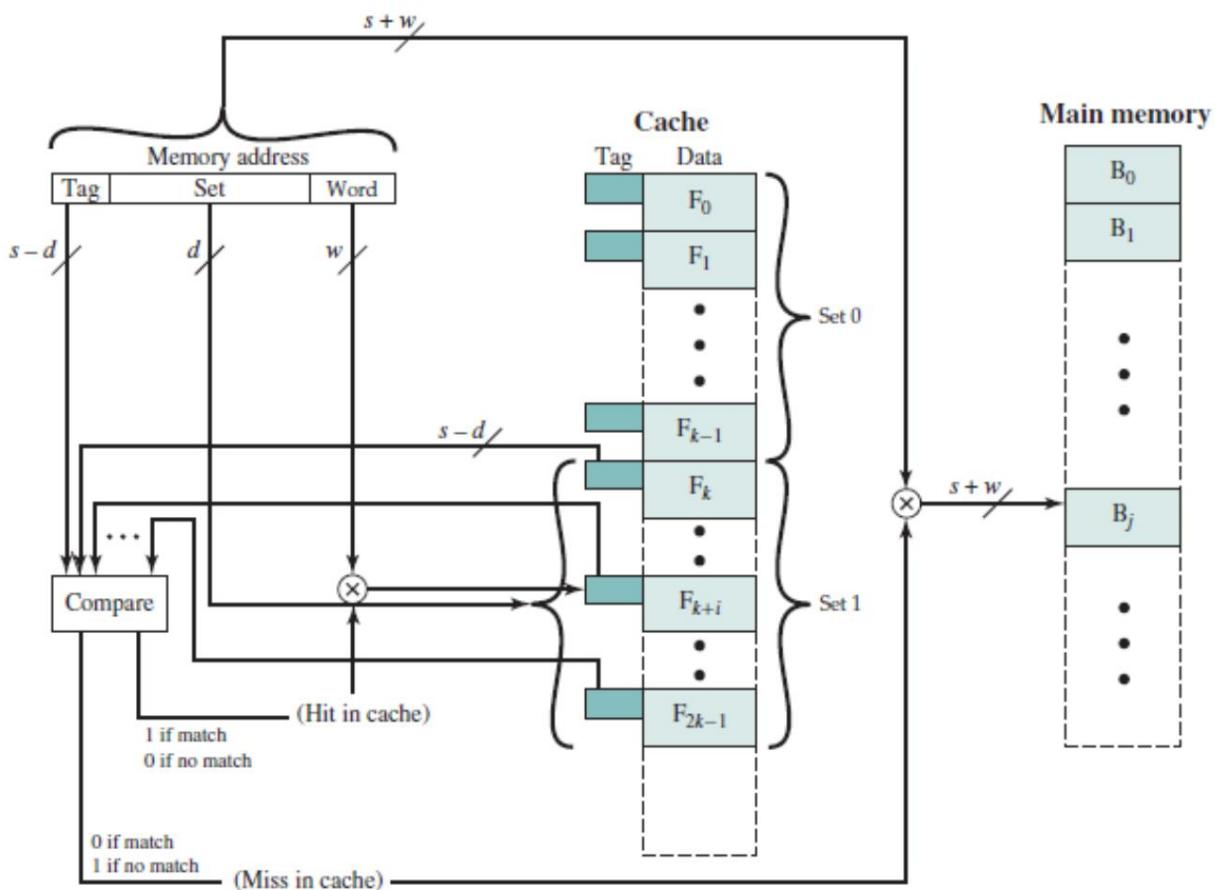
ÿ Superblokearen barruan malgutasuna dago barruko lerro edo moduren bat esleitzeko  
superblokea.

• modu kopurua

ÿ bide-kopurua 1 balitz ez legoke esleitzeko askatasunik eta zuzeneko korrespondentziaren kasuan egongo ginateke

ÿ Pista kopurua 2 balitz askatasun falta izango litzateke ordezkatu beharreko lerro bat aukeratu beharko litzatekeelako  
bi lerro posible.

ÿ Modu kopurua cachearen ahalmena balitz, askatasun maila maximoa izango litzateke, bat ordezkatzen  
m posiblearen lerroa.



**Figure 4.14** K-Way Set Associative Cache Organization

96. Irudia Ezarri Kontrolatzale Elkartzailea

- Helbide-formatua
  - ÿ Memoria nagusiaren helbide fisikoa: bloke-hitza
  - ÿ Cache memoriaren helbide fisikoa: tag-set-word
- Bilatu eragiketa hitz bat cachean.
  - ÿ Cache helbidearen formatuaren etiketa eta ezarri eremuak zehaztu.
  - ÿ Hitza edozein lerrotan egon daiteke baina esleitutako multzoan bakarrik, beraz, beharrezko da alderatu aipatutako multzoko lerroen etiketak soilik.
- 4.2 adibidea: 2 moduko multzoak
  - ÿ Memoria-helbidearen formatua eremuetan:

$2^2$  hitzak  $\rightarrow$  2 biteko hitz-eremuak  $k=2$

Cache-ko multzoen kopurua  $\rightarrow m / k = 2^{14} / 2 = 2^{13}$  multzoen cachean  $\rightarrow d=13$  bit

Etiketa: gainerako bitak =  $(24-13-2)= 9$  bit

ÿ ...jarraitu

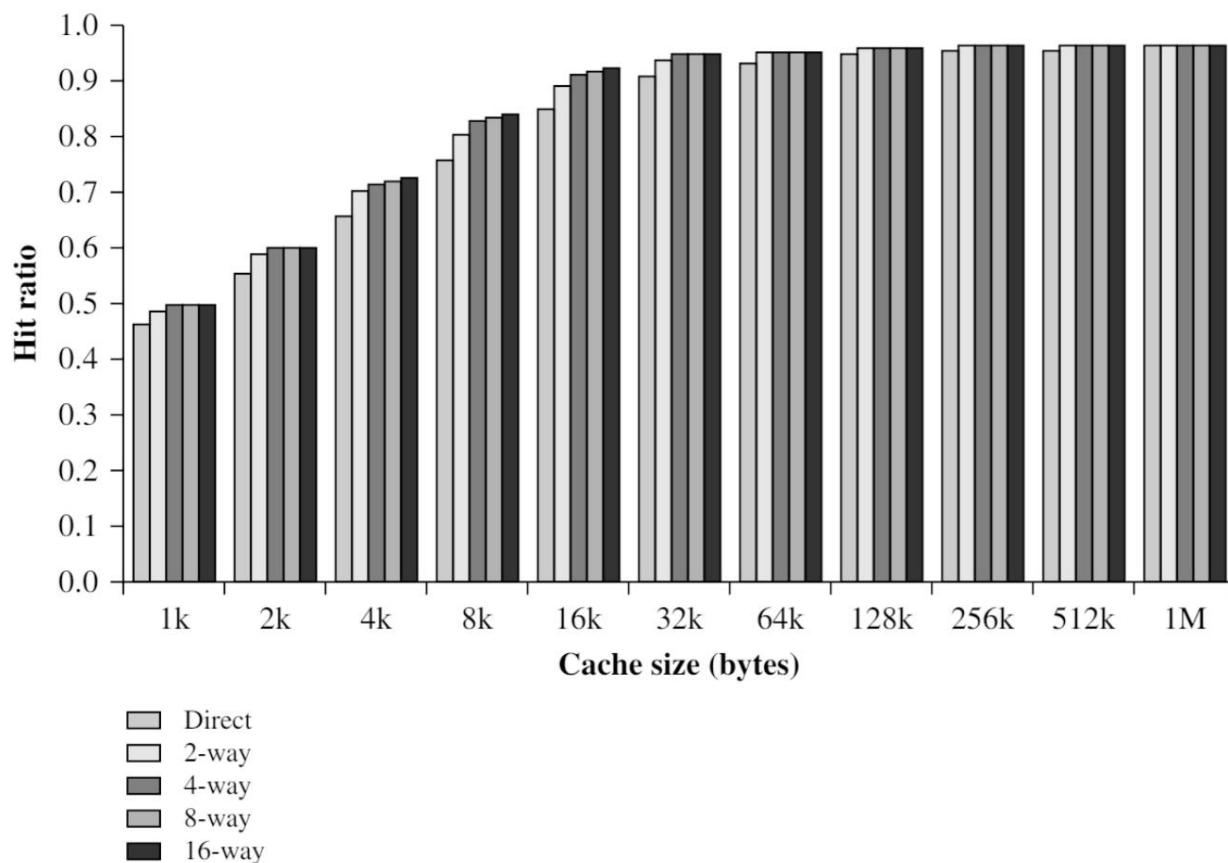
3 funtzi motak alderatzea

- Mapa zuzeneko bat etortze-funtzioan ez dago askatasunik aukeratzerakoan

cache-lerroa, korrespondentzia funtziok zehazten du. Funtzio asoziatibo oso baten kasuan, askatasuna erabateko da, irizpide estatistikoei jarraituz okupatu beharreko lerroa aukeratu ahal izatea, etab. Funtzio-multzo-asoziatiboaren kasuan, funtziok zehaztutako zuzenen multzoaren esleipenean ez dago askatasunik, baina ez dago askatasunik funtziok esleitutako multzoaren barruan lerroa aukeratzerakoan.

- Lehenengo bi kasuak, mapa zuzena eta asoziazio totala, by asoziatiboaren muturreko kasuak dira multzoak:

- ÿ Lerro bateko multzoa duten multzoen bidez asoziatiboak mapa zuzenaren kasua izango litzateke
- ÿ Cache-lerro guztien multzo batekin elkartua multzoa asoziatibo osoaren kasua litzateke



97. Irudia. Arrakasta-probabilitatea tamainaren eta elkartze-funtzioaren arabera

#### 9.4.5. Politikak

##### Ordezkoa

- Cache-kontrolatzailearen ordezko politika da "galdu" bat gertatzen denean, cache-lerro bat hautatu behar dela cachetik kendu eta memoria nagusitik harrapatutako blokeari lekua egiteko. Politika ezagunena gutxien erabilitako (LRU) politika da, non gutxien erabilitako elementuak lehenik bazterzen diren.

##### Idaztea

- Reading-arekin gertatzen den bezala, Writing-ekin CPUak Cachean idazten du, beraz, idazketa egin den lerroaren edukia ez dator bat Memoria Nagusian lotutako blokearekin. Write politika cache-lerroaren kopia eta eduki ezberdinekin lotutako memoria blokeak noiz egiten diren zehaztean datza. Write-Back politika eguneratzea edo kopia egitean datza lerro hori baztertu egingo den unean Ordezkaren politikaren arabera.

## 9.4.6. Optimizazio Adibidea

### Matrix produktua

- <https://johnnysswlab.com/make-your-programs-run-faster-by-better-using-the-data-cache/>

ÿ Arrays-en atzitzen duen begizta baten bidez implementatutako C hizkuntzan produktu matrizea datuak Cache memoriaren antolaketa nola kontuan hartu behar den adibidea da programatu algoritmoa. Adibideak ordena aldatzeko "loop truke" teknika erakusten du arrayetako elementuak harrapatzea.

## 9.4.7. Thinkpad L560

- cpuid | gehiago

```

--- cache 0 ---

cache mota = datuen cachea (1)
cache maila = 0x1 (1)

asoziatibotasun moduak = 0x8 (8)
multzo kopurua = 0x40 (64)

(tamaina sintetizadorea) = 32768 (32KB)
lerroaren tamaina? 32KB=2^15^bytes= 64multzo x 8 lerro/multzo x

n_bytes/line=512*n=2^9*n -> n=2^6^=64bytes
Modua: 0x01 (Idatzi itzuli)

--- cache 1 ---

cache-mota cache- = instrukzioen cachea (2)
maila elkartutasun- = 0x1 (1)

moduak multzo-kopurua (tamaina) = 0x8 (8)
sintetizadorea) = 0x40 (64)

= 32768 (32KB)
Modua: 0x01 (Idatzi itzuli)

--- cache 2 ---

cache-mota cache- = cache bateratua (3)
maila elkartutasun- = 0x2 (2)

moduak multzo-kopurua = 0x4 (4)
= 0x400 (1024)

(size synth) line size? = 262144 (256KB)
x n_byte/lerroa=2^12*n ->
256KB=2^18^byte= 1024 multzo x 4 lerro/multzo

n=2^6^=64bytes

--- cache 3 ---

cache-mota cache- = cache bateratua (3)
maila elkartutasun- = 0x3 (3)

moduak multzo-kopurua = 0xc (12)
= 0x1000 (4096)

(tamaina sintetizadorea) = 3145728 (3MB)

```

- getconf LEVEL1\_DCACHE\_LINESIZE
- getconf LEVELTABTAB
- sudo hwinfo | grep -i cachea -A 10
- sudo dmidecode -t 17

ÿ cache ÿ idazteko instrukzioa

## 9.5. Memoria birtuala

### 9.5.1. Sarrera

- [Eranskinean](#) informazio zehatza aurki daiteke
- Memoria birtuala zehatz-mehatz aztertzen da Sistema Eragileen irakasgaian, beraz, hemen Oinarritzko ideia pare bat baino ez dira gogoratuko.

### 9.5.2. Programatzailearen ikuspuntu eta CPU

- Programatzaileak mutua-lengoaien programatzen duenean, araztearekin lan egiten duenean, etab. eta memoria helbideak bistaratzen direnean, memoria-helbide hauek helbide-espazio BIRTUALAri dagozkio, ez helbide fisikoari.
- Era berean, CPUak helbide birtualekin funtzionatzen du. Adibidez programaren kontagailuak helbide birtualekin funtzionatzen du  
ÿ Beraz, hardware-zirkuitu bat beharrezkoa da, helbide birtualak helbide fisikoetara itzultzen dituena eta MC memoria-kontrolatzaileari entregatzen dituena. Hardware itzultaile honi Memory Management Unit (MMU) deitzen zaio.

### 9.5.3. Memoria birtualeko helbide-espazioa

- Konpilatzaileak, estekatzaileak eta abar goi-mailako programazio-lengoaiak (adibidez C lengoia) edo behe-mailako programazio-lengoaiak (adibidez, ASM lengoia, etab.) itzultzen dituztenean eta makina-lengoia bihurtzean, esleitu behar dute. memoriak argibideei, aldagaietik, datu-egiturei, erakusleei eta abarrei zuzendutakoak dira. Esleipen hau erabat libreki egiten da, ordenagailuaren memoria fisikoa kontuan hartu gabe. Helbideak abstraktuki esleitzen dira, hau da, taula baten sarrerak balira bezala, non taularen edukiak argibideak eta datuak diren.

### 9.5.4. Programa baten antolaketa logikoa makina-lengoaien: Atalak eta Segmentuak

- Atalei (datuak, testua, rodata...) eta segmentuei (kodea, datuak, pila, pila) buruzko informazioa.  
[eranskina](#)
- Programa-moduluak atalaletan eta programa/prozesu bat segmentutan antolatzeak errazten du itzulpena (konpilatzailea, estekatzailea, mihiztatzailea) eta baita prozesuari memoria birtuala esleitzea ere.

### III Teoria Ariketak

# 10. kapitulua. Ariketak

## 10.1. Gutxieneko Ariketa Zerrenda

### 10.1.1. I. zatia

- Ondorengo zerrendak egin beharreko gutxieneko ariketak hautatzen ditu:

1. Von Neumann Arkitektura: 1.1,1.2,1.3,1.4,1.6,2.1,2.2,2.3  
2. Datuen irudikapena: 2,3,4,5  
3. Eragiketa aritmetikoak: 2,5,8,10  
4. Eragiketa logikoak: 1,2,3  
5. Argibideen irudikapena: 1,2,5,7,9,15,16  
6. Muntaia Programazioa: 1.1,2.1,2,2

### 10.1.2. II. zatia

- Ondorengo zerrendak egin beharreko gutxieneko ariketak hautatzen ditu:

1. Von Neumann Arkitektura: 2.5,2.6  
7. CPU: 12.4,12.9,12.11,13.4  
8. Memoria: DRAM (Teoria gaiaren adibideak), Cachea (1,2,3,4,5,  
9. Sarrera/Irteera:

## 10.2. von Neumann arkitektura

### 10.2.1. Ordenagailuak: IAS, ENIAC, ..

1. IAS programa bat idatzi behar duzu hurrengo ekuazioaren emaitzak kalkulatzeko.  $Y = \text{Batura}(X) \quad X=1\text{etik } X=\text{Nrako}$ . Demagun kalkuluaren emaitzak gainezka egiten ez duela eta  $X$ ,  $Y$  eta  $N$  zenbaki oso positiboak direla  $N > 1$ . Oharra: IAS-ek ez zuen mihiatzadura hizkuntza soilik makina-lengoaiarik.

1. Erabili  $\text{Sum}(Y) = N(N+1)/2$  ekuazioa IAS programa idaztean. Kontzeptuak komentatu: datuak, argibideak, memoria nagusia, erregistroak, atalak, eragiketa, eragiketa, eragiketa-eremua, eragiketa-eremua, memoria-helbidea, memoria-edukia, eragigaiaren erreferentzia, eragiketa inplizitua, eragigaiaren helbideratzea, zuzeneko helbideratzea. eragigaiarena, kode bitarra, kode hamaseitarra, eragiketa-kodea, eragiketa-eremuaren kodea. Exekutatu iturburu-programa IASSIm emuladorearekin.

ÿ Garapena:

; Lehenengo  $N$  zenbaki osoen batura.  $Y=N(N+1)/2$ ; CPU  
IAS ;  
muntaia-lengoaia: IASSim simulagailua;  
William Stalling-en, Structure of Computers liburuko 2.1 ariketa

; Jarraitzen dugun algoritmoa hau da: lehenengo batuketa  $N+1$ , gero  $N(N+1)$  biderketa eta azkenik zatiketa. Emaitza bakoitza bertan gordetzen da

memoria nagusia behar izanez gero.

; Programa 4 hizkuntzatan deskribatzen dugu: iassim mutaia-lengoia,  
makina-lengoia bitarra, RTL hizkuntza eta WStalling mutaia-lengoia

; JARRAIBIDEEN ATALA

S(x)->Ac+ n 01 n ; ;	A.C. <- M[n]	:KARGA M(n)
S(x)->Ah+ bat ; 05 bat ;	A.C. <- AC+1	:GEHITU M (bat)
At->S(x) y 11 y ; ;	M[y] <- AC	:DENDA M(y)
S(x)->R y 09 y ; ;	A.R. <- M[y]	:KARGA MQ,M(y)
S(x)*R->A n ; ;	0B n ; ;	AC:AR <- AR*M[n] ;MUL M(n)
R->A ; ;	0A ; ;	A.C. <- AR :KARGATU MQ
A/S(x)->R bi;	0C 2 ; ;	A.R. <- AC/2 :DIV M (bi)
R->A ; ;	0A ; ;	A.C. <- AR :KARGATU MQ
At->S(x) eta ; ;	11 eta ; ;	M[y] <- AC :DENDA eta
geldialdia		
; Argibide kopurua bikoitia denez, zuzentaraua ez da beharrezko .hutsik		

; DATU ATALA

; Aldagaien adierazpena eta hasieratzea

eta: .datuak 0 ;emaiza

; Konstanteen Adierazpena

n: .datuak 5 ;N parametroa

bat: .datuak 1

bi: .datuak 2

2.

Egin "bide gogorra" (a) ataleko ekuazioa erabili gabe:

$$\sum_{i=1}^5 i$$

ÿ Garapena:

; n...+3+2+1(+0) balioak batzen ditu begizta batean eta gordetzen ditu  
; memorian dagoen batura "batura" etiketatutako tokian

begizta: S(x)->Ac+ n ;kargatu n AC-ra

Cc->S(x) pos ;AC >= 0 bada, jauzi pos  
gelditu ;bestela eginda

.hutsik ; 20 biteko 0

pos: S(x)->Ah+ batura ;n gehitu baturari

At->S(x) batura ;guztira itzuli batura

S(x)->Ac+ n ;kargatu n AC sartu

S(x)->Ah- bat ;gutxitu n

At->S(x) n ;denda dekrementatua n

Cu->S(x) begizta ;atzera egin eta berriro egin

n: .data 5 ;guztira 6 aldiz biraka egingo du

bat: .datua 1 ;gutxitzeko konstantea n

batura: .datuak 0 ;non exekutiboa/azkeneko guztirakoa gordetzen den

2. IAS-n, nolakoa izango litzateke makina-kodearen instrukzioa 2. memoria helbidearen edukia kargatzeko metagailura? Memoriara zenbat bidaia egin behar ditu CPUak instrukzio hau osatzeko irakaskuntza-zikloan zehar?

ŷ Garapena:

ŷ 0x01002: 01 eragiketa kodea eta 002 eragigaiaren erreferentzia

ŷ Memoria nagusirako bi sarbide: instrukzioen atzematea eta eragigaien atzematea

3. IAS-en, deskriba ezazu ingelessez CPUak memoriatik balio bat irakurtzeko egin behar duen prozesua eta memorian balio bat idaztea MAR, MBR, helbide-busa, datu-busa, eta jartzen denaren arabera autobus kontrola.

ŷ Garapena:

#### Irakurketa

ITSASOA                    <- helbidea

Helbidea Autobusa <- MAR

Autobus Kontrola <- Irakurri

Datuen Busa            <- Datuak

MBR                    <- Datu-busa

#### Idaztea

ITSASOA                    <- helbidea

Datuen Busa            <- MBR

Kontrol Busa <- Idatzi

ŷ Informazio-fluxuaren deskribapen honi CPU datu-bide deritza.

4. Behean agertzen den IAS ordenagailuaren memoria edukia kontuan hartuta,

#### Helbide Edukiak

08A 010FA110FB

08B 010FA0F08D

08C 020FA110FB

ŷ erakutsi programaren mihiztadura-lengoaiaren kodea, 08A helbidetik hasita. Azaldu zer hau programak egiten du.

ŷ Garapena:

Helbidea	Edukiak	RTL	Argibideak	
08A	010FA110FB AC $\ddot{y}$ M[0FA] M[0FB] $\ddot{y}$ AC LOAD M[0FA] DENDA			M[0FB]
08B	010FA0F08D AC $\ddot{y}$ M[0FA] AC>0:PC Ÿ 0x08D (0:19)		KARGATU M[0FA] JMP +M[08D(0:19)]	
08C	020FA110FB AC $\ddot{y}$ -M[0FA] M[0FB] $\ddot{y}$ AC LOAD -M[0FA] DENDA			M[0FB]

ŷ Edukia: ezkerreko instrukzioa 010FA (Cod Op 01 Helbidea 0FA) eta eskuineko instrukzioa 110FB -

ÿ Alderantzizko muntaketa prozesu honi (mihiztadura-lengoaia ÿ kode bitarra) deitzen zaio  
desmuntatu (kode bitarra ÿ mihiztadura hizkuntza)

ÿ Programak funtziotan hau betetzen du:

ÿ 0x0FAren edukia positiboa bada, 0x0FA posizioko memoria edukia 0xFB posiziora kopiatzen du eta 0x08D posiziora jauzi egiten du, 0xFAREN edukia metagailuan utziz. 0x0FAren edukia negatiboa bada, 0x0FA posizioko memoriarren edukia 0xFB posiziora kopiatzen du zeinu aldaketarekin eta 0x08D posiziora jauzi egiten du, 0xFAREN edukia positiboa utziz metagailuan, hau da, moduluan.

5. Adierazi IAS mikroarkitekturaren datu-bide bakoitzaren zabalera, bitetan (adibidez, AC eta ALU artekoan).

ÿ Garapena:

ÿ AC, AR eta MBR 40 bit

ÿ IBR 20 bit

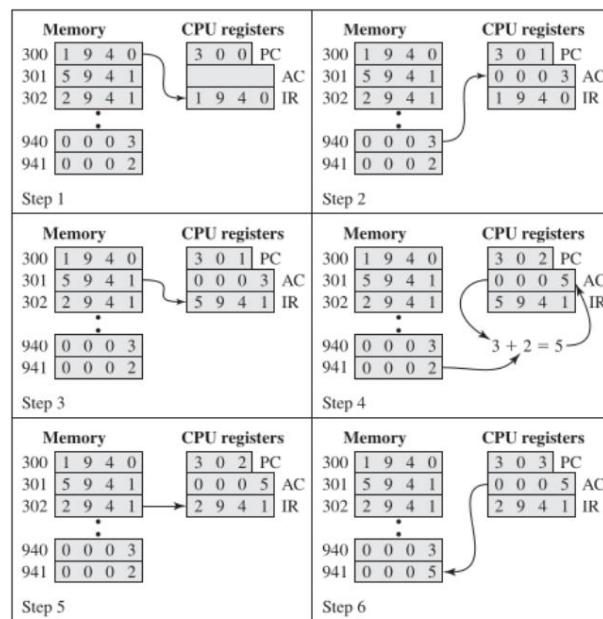
ÿ MAR eta PC 12 bit

ÿ IR 8 bit

6. Demagun jarraibide hauek dituen ordenagailu bat:

Eragiketa Kodea	Deskribapena
0001	Kargatu AC memoriatik
0010	Gorde AC memorian
0101	Gehitu ACra memoriatik

to. PUZaren instrukzio-zikloen fazeak deskribatzea programa hori exekutatzeko orduan kontuan hartuta  
Kontuan hartu hurrengo irudia:



98. Irudia. Irakaskuntza Zikloak

b. Zein da instrukzio-formatua? Zein da programa-kontagailu-erregistroen eta instrukzio-erregistroen tamaina? Zein da datu-busaren eta helbide-busaren tamaina?

7. ENIAC makina hamartar bat zen, non erregistro bat 10 huts-hodiko eratzun batek adierazten zuen. Edozein unetan, huts-hodi bakarra zegoen ON egoeran, 10 zifren bat adierazten duena. ENIAC-ek aldi berean ON eta OFF egoeran hainbat huts-hodi izateko gaitasuna zuela suposatuz, zergatik da irudikapen hau "alferrik" eta zein balio osoen sorta irudikatu genezake 10 huts-hodiak erabiliz?

ÿ Garapena: 10 hodirekin 0-9 zifrak soilik irudikatu ditzakegu

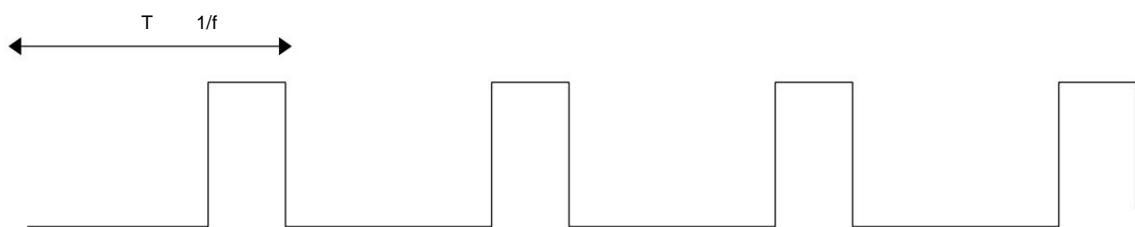
8. Erreferentiazko programa bat 40 MHz-ko prozesadore batean exekutatzen da. Exekutaturiko programa 100.000ek osatzen dute instrukzioen exekuzioak, instrukzio-nahasketa eta erloju-ziklo-zenbaketa honekin:

Instrukzio_Mota	Instrukzio_zenbaketa	Zikloak_Instrukzioko
Aritmetika_osokoa	45.000	1
datu_transferentzia	32.000	2
koma_mugikorra	15.000	2
Kontrol_transferentzia	8000	2

ÿ Programa honen KPI, MIPS tasa eta exekuzio denbora eraginkorra zehaztu.

ÿ Garapena:

ÿ CPU erlojua



$\ddot{y}$  T=1/f= 25ns: PUZaren erloju-zikloa: mikroeragiketa baten gutxieneko iraupena.

$$\text{ý KPI: zikloak} \quad \text{arabera} \quad \text{argibidea:} \quad \text{merezi} \quad \text{erdia} = \\ 1*(45/100)+2*(32/100)+2*(15/100)+2*(8/100)=0,45+0,64+0,30+0,16=1,55 \text{ cpi}$$

ŷ MIPS: Milioika Inst. segundoko:  $(1/\text{CPI})(\text{inst}/\text{ziklo}) * F_{\text{clock}}(\text{ziklo}/\text{seg}) * 10^{-6} = (1/1,55) * 40 * 10^6 * 10^{-6} = 25,8 \text{ mips}$

$$\hat{y} T = (1/\text{MIPS})(\text{seq/milior argibide}) * 100.000 * 10^{-6} = 3,87 \text{ ms}$$

### 10.2.2. CPU-Memoria Interkonexioa

1. Demagun 16 biteko helbidea sortzen duen mikroprozesadore hipotetiko bat (adibidez, demagun programa-kontagailua eta helbide-erregistroak 16 biteko zabalera dute) eta 16 biteko datu-busa dute.
    1. Zein da prozesadoreak zuzenean atzi dezakeen memoria-helbide-espazio maximoa konektatuta badago "16 biteko memoria"ra?
    2. Zein da prozesadoreak zuzenean atzi dezakeen memoria-helbide-espazio maximoa konektatuta badago "8 biteko memoria" batera?
    3. Zein ezaugarri arkitektonikok mikroprozesadore honi "I/O espazio" bereizi batera sartzeko aukera emango diote? asko 8-bit I/O atakak opar ditzake mikroprozesadoreak? Zenbat 16 biteko I/O ataka? Azaldu.

v Gogooan izan:

ÿ Helbide-espazioa: helbide-bus bereko helbide multzoa. The  
Abalmena BYTEtan adierazten da.

ü Memoria Nausia eta I/O Controller Space espazio desberdinak dira

Sistemaren bus helbide-bus bera partekatzen dute baina kontrol seinale bat dago memoria nagusiarekin edo I/O kontrolagailurekin konexioa aktibatzen duena.

“16 biteko memoria”: 16 biteko hitzaren tamaina. 16 biteko datu-busa

“8 biteko memoria”: 8 biteko hitzaren tamaina. 16 biteko datu-busa

ÿ I/O atakaren zenbakia: I/O kontrolagailuko ataken kopurua. Ataka bakoitzaz periferiko baterako.  
Sarrerako ataka irteerako atakatik bereizten da. 8 biteko I/O ataka 8 biteko datu-buffer bat duen ataka  
da eta 16 biteko I/O ataka 16 biteko datu-buffer bat duena.

ÿ Garapena:

ÿ Marraztu ikusten duzun autobusaren diagrama:

ÿ periferikoen, ataken, I/O kontroladorearen, memoria nagusia, CPUaren, busen arteko  
interkonexioak, bus bakoitzarekin funtzionatzeko modua adieraziz.



.....

ÿ a) 216 byte. 64KB. Bi byteko datuak datu-busean transferitzen dira.

ÿ b) 216 byte. 64KB. Byte bateko datuak datu-busean transferitzen dira.

ÿ c) Sistema-busean kontrol-seinalea behar da: I/O seinalea

ÿ d) 28= 256 sarrerako ataka eta 256 irteera ataka kontrolagailuan helbideratzea  
I/O I/O buffer-aren tamaina edozein dela ere 8 biteko edo 16 biteko.

2. Demagun 32 biteko mikroprozesadore bat, 16 biteko kanpoko datu-busa duena, 8 MHz-eko sarrerako erloju batek gidatua. Demagun mikroprozesadore honek bus-ziklo bat duela, zeinaren gutxieneko iraupena lau sarrerako erloju-ziklo berdina duen. Zein da mikroprozesadore honek jasan dezakeen gehieneko datu-transferentzia-tasa autobusean, byte/s-tan? Bere errendimendua areagotzeko, hobe al litzateke kanpoko datu-busa 32 bit egitea edo mikroprozesadoreari ematen zaion kanpoko erloju-maiztasuna bikoiztea? Adierazi egiten dituzun beste hipotesi batzuk, eta azaldu. Aholkua: zehaztu autobus-ziklo bakoitzeko transferitu daitezkeen byte kopurua.

ÿ Garapena:

ÿ 32 biteko CPU: barneko CPU erregistroen tamaina. CPU tokiko (barneko) datu-busa

ÿ 16 biteko kanpoko datu-busa: sistemaren datu-busa

ÿ CPU sarrerako erlojua: 8MHz

ÿ bus-zikloa: sistema-bus-zikloa: iraupena CPUarena baino 4 aldiz handiagoa da: 2 MHz.

ÿ a) Datuen transferentzia-tasa: teorikoki bus-ziklo bakoitzeko transferentzia baten sekuentzia jarraituan datu kopurua segundo 1ean: 2MTransfer/s. Transferentzia bakoitzak datu-busak 16 bit transferitzen ditu, hau da, 2 byte = 2M/s \* 2B = 4MB/s

ÿ b) Datu-busaren zabalera bikoitzuz, banda zabalera bikoitzu ÿ 8MB/s

ÿ c) Erlojuaren maiztasuna bikoizteak proportzionalki autobusaren ziklo murrizten du eta autobusaren zabalera bikoitzu egiten du. banda ÿ 8MB/s

3. Demagun bi mikroprozesadore 8 eta 16 biteko zabalerako kanpoko datu-busak, hurrenez hurren. Biak Bestela, prozesadoreak berdinak dira eta haien autobus-zikloak bezain luzeak dira.

1. Demagun instrukzio eta eragiketa guztiak bi byteko luzera dutela. Zein faktoreren arabera egiten dute datu maximoek transferentzia-tasak desberdinak dira?

2. Errepikatu operandoen eta argibideen erdiak byte bateko luzera dutela suposatuz.

ŷ Garapena:

ŷ a) 8 biteko CPU1ek banda zabalera erdia du (%50) 16 biteko CPU2rekin alderatuta.

ŷ b1) CPU1: 2 byteen % 50 2 bus-ziklotan 2 byte bakoitzeko eta beste % 50 byte 1 batean.

$$\text{byte bakoitzeko ziklo} = 2 \text{ziklo} * \%50 + 1 \text{ziklo} * \%50 = 1,5 \text{ziklo}$$

ŷ b2) CPU2: 2 byteen % 50 bus ziklo batean 2 byte bakoitzeko eta beste % 50 bytearen % 50 bus ziklo batean (instrukzio edo datu bakarra atzi daiteke bus ziklo bakoitzean) = 1 ziklo \* % 50 + 1 ziklo \* %50 = 0,5 + 0,5 = 1 ziklo

ŷ b) b1 eta b2-ren arabera, CPU1ek CPU2 baino %150eko banda zabalera du, hau da,

PUZaren %66,62.

4. Mikroprozesadore batek gehikuntza-memoria zuzeneko instrukzioa du, memoria-kokapen bateko balioari 1 gehitzen diona.

Instrukzioak bost fase ditu: lortu opcode (buseko lau erloju-ziklo), lortu eragigaien helbidea (hiru ziklo), eskuratu eragigaia (hiru ziklo), 1 eragigai gehitu (hiru ziklo) eta gorde eragigaia (hiru ziklo).

1. Zein kopurutan (ehunekotan) handituko da instrukzioaren iraupena bi autobus sartu behar baditugu itxaron egoerak memoria irakurtzeko eta memoria idazteko eragiketa bakoitzean?

2. Errepikatu gehikuntza-eragiketak 3 zikloren ordez 13 ziklo behar dituela suposatuz.

ŷ Garapena:

ŷ irakaskuntza-zikloa:  $4+3+3+3+3= 16 \text{ ziklo}$

ŷ a) memoria sarbideak 3 bilatze faseetan eta biltegiratze fasean ŷ  $2^4$  zikloko gehikuntza itxaron ŷ  $8/16$ ko igoera ŷ %50eko igoera

ŷ b) irakaskuntza-zikloa:  $4+3+3+13+3= 26 \text{ ziklo}$  ŷ irakaskuntza-zikloa  $8/26$  handitu ŷ %34 handitu

5. Intel 8088 mikroprozesadoreak 3.19 irudikoaren antzeko irakurketa-busaren denbora du, baina lau prozesadorearen erloju-ziklo behar ditu. Baliozko datuak autobusean daude, laugarren prozesadorearen erlojuaren ziklora luzatzen den denbora tarte batez. Demagun prozesadorearen erloju-abiadura 8 MHz-ekoa.

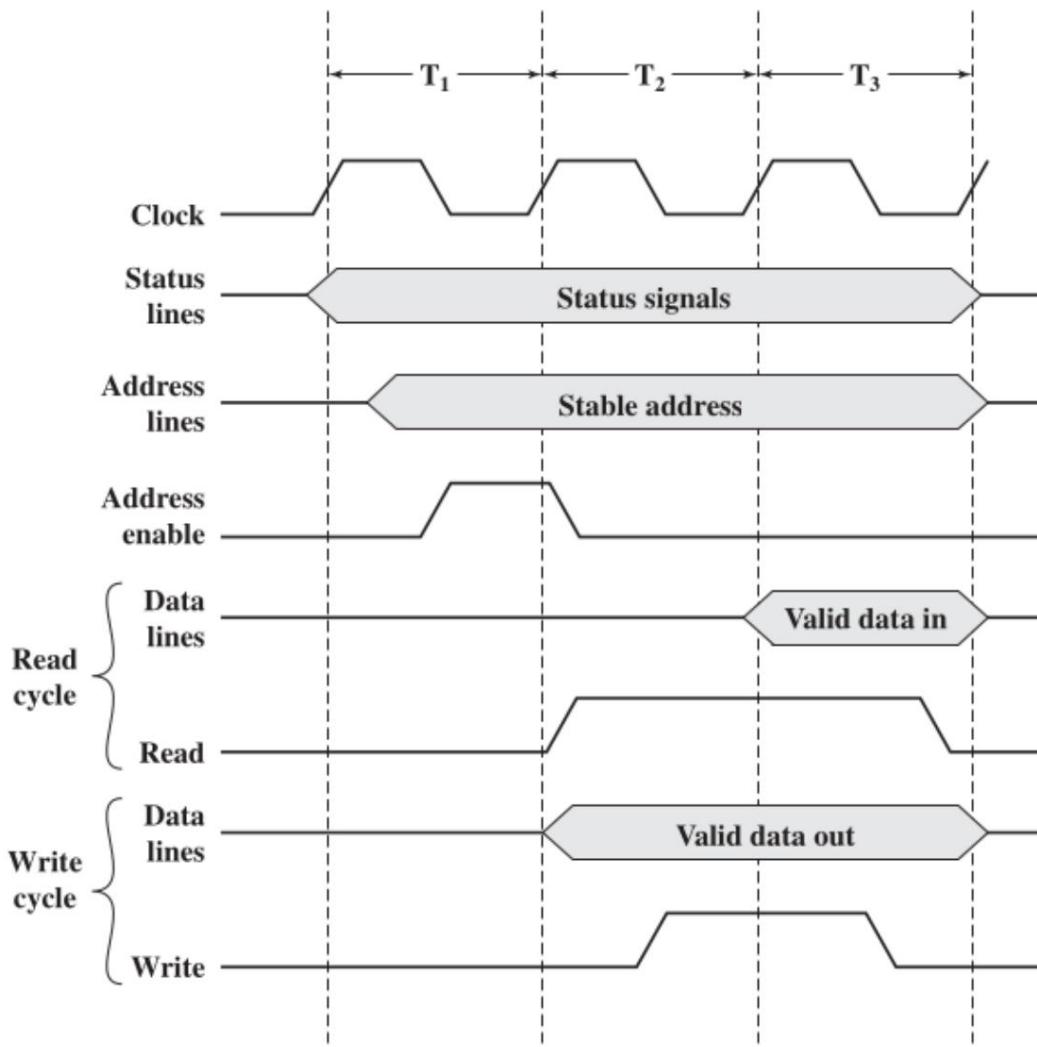


Figure 3.19 Timing of Synchronous Bus Operations

99. Irudia Memoria Nagusiaren Irakurtzea/Idazketa baten egutegia

1. Zein da datu-transferentzia-tasa maximoa?

2. Errepikatu baina transferitutako byte bakoitzeko itxaron-egoera bat txertatu beharra dagoela suposatu.

ŷ Garapena:

ŷ 8088: datu-busa: 1 byte

ŷ irakurtzeko denbora: 4 CPU ziklo

ŷ datuak baliozkoak: 1 prozesadorearen erloju-ziklo. Irakurtzeko denboraren laugarren zikloa.

ŷ CPU erloju: 8MHz

ŷ a) 4 ziklo transferentzia bakoitzeko.  $8\text{MHz}/4\text{ziklo} = 2\text{MT/s} = 1 \text{ byte transferentzia bakoitzeko}$  ŷ 2MB/s

ŷ b) transferentzia bakoitza ziklo bat da transferitu gabe (4,1) ŷ errrendimendua = gehienezkoaren  $4/5$  ŷ  $(4/5)*2\text{MB/s} \approx 1.6\text{MB/s}$

6. Intel 8086 16 biteko prozesadorea da 8088-ren modu askotan antzekoa. 8086-ak 16 biteko bus bat erabiltzen du, aldi berean 2 byte transferi ditzakeena, beti ere beheko byteak helbide berdina badu. . Hala ere, 8086-k hitz-eragigai lerrokatu biak zein bakoitiak onartzen ditu. Lerrokatutako hitz bakoiti bat erreferentzia egiten bazaio, bi memoria-ziklo behar dira, bakoitza lau bus-ziklo osatua, hitza transferitzeko. Demagun 8086-ri buruzko instrukzio bat, 16 biteko bi eragiketarekin. Zenbat denbora behar da eragigaiak eskuratzeko?

Eman erantzun posibleen sorta. Demagun 4 MHz-ko erloju-tasa eta itxaron-egoerarik ez

ŷ Garapena:

ŷ 8086: datu-busa: 2 byte

- ÿ intel : little endian: LSB bytea helbide txikian gordetzen da eta MSB bytea helbidean goi-mailakoa.
- ÿ Datu pareak lerrokatzeak 1 memoria-ziklo behar du.
- ÿ lerrokatuta dauden hitz bakoitiek 2 memoria-ziklo behar dituzte. Memoria-ziklo bakoitzak 4 ziklo ditu autobusa
- ÿ 2 biteko 2 eragiketa dituen instrukzioa. 4MHz PUZaren erlojua ÿ 0,250 mikrosegundo ÿ 250 ns
  - ÿ a) bi eragigaien lerrokadura bikoitia dute
    - ÿ 1 memoria-ziklo eragigai bakoitza: 2 memoria-ziklo: 8 bus-ziklo ÿ 2 mikrosegundo
  - ÿ b) eragigai batek lerrokadura bikoitia du eta besteak bakoitiak
    - ÿ 1 memoria-ziklo zenbaki bikoitiarentzat eta 2 ziklo bakoitiarentzat: 3 memoria-ziklo: 12 bus-ziklo ÿ 3 mikrosegundo
  - ÿ c) bi eragigaien lerrokadura bakoitia dute
    - ÿ 2 ziklo eragigai bakoitza: 4 memoria ziklo: 16 bus ziklo ÿ 4 mikrosegundo.

7. Demagun 32 biteko mikroprozesadore bat, zeinaren bus-zikloa 16 biteko mikroprozesadorearen iraupen bera duen. Demagun, batez beste, eragiketen eta argibideen %20ak 32 bit-eko luzera duela, %40ak 16 bit-eko luzera duela eta %40ak 8 bit-eko luzera besterik ez duela. Kalkulatu 32 biteko mikroprozesadorearekin instrukzioak eta eragigaiak eskuratzean lortutako hobekuntza.

#### ÿ Garapena

- ÿ Bus-zikloaren ertz positibo bakoitzean, memoria eta CPUren arteko transferentzia egiten da. 16 biteko CPUak 2 byte edo gutxiagoko transferentzia egiten du eta 32 biteko CPUak 4 byte edo gutxiagoko transferentzia egiten du.
- ÿ Ziklo erdiak (16 biteko CPU) =  $0,2 \times (2 \text{ ziklo bakoitzeko } 2 \text{ byteko bi transferentziatarako } a) + 0,4 \times 1 + 0,4 \times 1 = 1,2$  batez besteko ziklo
- ÿ Batez besteko zikloak (32 biteko CPU) =  $0,2 \times 1 + 0,4 \times 1 + 0,4 \times 1 = 1$  batez besteko ziklo
- ÿ  $(1,2 - 1) / 1,2 = \% 17$

### 10.3. Datuen irudikapena

1. Adierazi 1197 zenbaki hamartarra oinarriean:

to. Hamaseitarra

Ondoz ondoko zatiketak 16 0x4AD

b. zortzikoa:

Ondoz ondoko zatiketak 8 0o2255

c. bitarra:

Ondoz ondoko zatiketak 2  
0b10010101101

- d. Adierazi 0x4AD zenbakia oinarri bitarrean eta oinarri octalean bihurketa zuzen baten bidez, gabe kalkulatu bere balioa.
2. Adierazi -1197 zenbakia oinarri bitarrean eta hamaseitarren eta formatuan:
- to. Zeinu-magnitude: 0b110010101101 ѕ 0xCAD
  - b. 2 osagarria: 0b101101010011 ѕ 0xB53
3. Kalkulatu 2 osagarrian 8 biteko zenbaki osoen barrutia (27 -1,-27)
4. Erabili hamaseitar idazkera:
1. Adierazi 23 balioa BCD formatuan.
    - ÿ Binary Code Decimal (BCD) formatuan, zifra hamartar bakoitza modu independentean zabaltzen da bere 4 biteko kode bitarrean.
    - ÿ 2ÿ0010; 3ÿ0011; 23 ѕ 0010-0011
  2. ASCII karaktereak 23
    - ÿ 0x32-0x33 ѕ 0011-0010-0011-0011
5. Idatzi á azentua duen a karakterea UTF-8 kode unibertsalean (hamasimalean), Unicode kodean (U+XXXX), ascii hedatuan eta iso-8859-1 UTF-8 estekan kontsultatuz, eskuliburuak: man ascii, man iso-8859-1 eta showkey komandoa -a ѕ idatzi á
6. Paketatutako zenbaki hamartar bakoitzerako, erakutsi balio hamartarra
1. 0111 0011 0000 1001
    - ÿ 7309
  2. 0101 1000 0010
    - ÿ 582
  3. 0100 1010 0110
    - ÿ Ezin da 1010 10 balioari dagokiarik, ez baitu zifra hamartar bat bi baizik.
7. (EZ) Batuetan aurkitzen den zenbaki oso bitaren beste irudikapen bat baten osagarria da.
- Zenbaki oso positiboak zeinu-magnitudearen modu berean adierazten dira. Zenbaki oso negatibo bat dagokion zenbaki positiboaren bit bakoitzaren osagarri boolearra hartuz adierazten da. Oharra: osagarri aritmetika bat hardwaretik desagertu zen 1960ko hamarkadan, baina oraindik ere Internet Protokoaren (IP) eta Transmisiaren Kontrolerako Protokoaren (TCP) kontrol-sumaren kalkuluetatik irauten du.
1. Eman uneko osagarien zenbakien definizioa biten batura hztatua erabiliz.
    - ÿ Eman n=3 bit-eko bihurketa-adibideak
      - ÿ 000ÿ111 (gero zeroak bi irudikapen ditu), +1: 001ÿ -1:110, +2:010ÿ'-2':101,  
+1:011ÿ'-3':100
    - positiboa n bitekinÿ  

$$\sum_{i=0}^{n-1} b_i 2^i$$
    - ÿ n biteko negatiboak
      - ÿ Kontuan hartzen dugu bi osagarria = a\_osagarria + 1
      - ÿ X-ren n biteko bi osagarria 2n kenketa bitar gisa kalkula daiteke (bitarrean) -
        - X: adibidez 3 bitekin +1 bi osagarria 1000-1=111 da
      - ÿ 1-en osagarria 2-ren osagarria ken 1 ѕ 2 n -X -1 da. Adibidez, 3 bitekin  
+1-ren osagarria 1000-1-1 = 110
  2. Zein da n biteko osagarietan irudika daitezkeen zenbakien tarte?
- ÿ Maximo positiboa ѕ 011\_\_1 : 2n -1
  - ÿ Gehienezko negatiboa ѕ 100\_\_0 : -(2n -1)

8. Irudikatu 0,56789 bitarrean ondoz ondoko biderketak erabiliz

$$\begin{aligned} 0,56789 * 2 &= 1.13578 - 1 + 0,13578 - 1, \text{ posizio bit } -1,13578 * 2 = 0,27156 - 2 \\ &= 0,27156 - 0,27156 * 2 = 0,54312 \rightarrow 0, \text{ posizioa } -3 \text{ bit } 0,54312 \\ * 2 &= 1.08624 = 1 + 0,08624 \rightarrow 1, \text{ posizio bit } -4 \end{aligned}$$

9. Irudikatu 0,0625 bitarrean, ondoz ondoko biderketak erabili gabe.

ÿ 0,0625 = M\*2<sup>E</sup>, hala nola, E zenbaki oso bat izan dadin

ÿ log\_dos (0,0625) = log\_dos (M)+E

ÿ -4 = log\_dos(M)+E ÿ E=-4 eta log\_dos(M)=0 ÿ M=1

10. Adierazi 1234.56789 zenbaki erreala oinarri bitarrean:

ÿ Puntu finkoko formatuan

Zati osoa: 1234: 10011010010 Zatikia:

0,56789: 0,100100010110000101 Zenbakia 1234,56789:

10011010010,1001000101100001

ÿ Idazketa zientifikoan: 1,0011010010100100010110000101\*2<sup>-10</sup>

ÿ Doitasun bakarreko koma mugikorrean:

Zeinu eremua: + : 0

Berretzaile-eremua (8 bit): 10+127 = 137 = 10001001

Mantisa frakzioaren eremua (23 bit) = 00110100101001000101100

ÿ Doitasun bikotzeako koma mugikorrean:

Zeinu eremua: + : 0

Berretzaile-eremua (11 bit): 10+1023 = 1033 = 10000001001

Mantisa frakzioaren eremua (52 bit) = 00110100101001000101100001010\_0

11. Kodetu 3 zenbaki osoa doitasun bakarreko FPn

ÿ 3 = 11 = 1,1 21

ÿ S=0, E=1+127=128, Mn=0,1

ÿ 0-1000-0000-1000-0000-zeroak

ÿ Ez da biribildu behar

ÿ Emaitza= 0x40400000

12. Irudikatu 123456789 zenbaki naturala doitasun bakarreko Koma Mugikorrean (IEEE-754)

ÿ 123456789 = 0x075BCD15 = 111-0101-1011-1100-1101-0001-0101  
1.1101011011100110100010101\*2+26 =

ÿ Biribila = 1,1101011011100110100011\*2+26

ÿ Eremuaren zeinua= 0

ÿ Eremu Exp=  $26+127=153=10011001$

✓ Mantisa frakzioaren eremua = 0.11010110111100110100011

ÿ 0x4CEB79A3 emaitza

### 13. Float Point:

1. Demagun puntu finkoko irudikapen bat zifra hamartarrak erabiliz, zeinetan inplizitutako erroda-puntua edozein posiziotan egon daitekeen (adibidez, zifra esanguratsuenaren eskuinean, zifra esanguratsuenaren eskuinean, eta abar). Zenbat zifra hamartar behar diren Planck-en ( $6,63 \times 10^{-27}$ ) konstantearen zein Avogadroren zenbakieren ( $6,02 \times 10^{23}$ ) hurbilketak adierazteko ) Erdi-puntua inplizitua posizio berean egon behar da bi zenbakientzat.

ÿ Planck zenbakieri dagokionez, koma 27 posizioak ezkerrera mugitu behar dituzu gehi bi zifrak  
eskuinera (63) ÿ 27 zifrako zatikia

ÿ avogrado zenbakien kasuan, koma 23 posizioak eskuinera mugitu behar dira gehi zifra. ezkerrean (7) ÿ 24 zifrako osoko zatia

ŷ  $29+24=53$  zifra bietarako

2. Kontuan hartu orain koma mugikorreko formatu hamartar bat, berretzailea 50eko alborapeneko irudikapen alboratuan gordeta duena. Adierazpen normalizatua suposatzen da. Zenbat digitu hamartar behar dira konstante hauek koma mugikorreko formatu honetan irudikatzeko?

ÿ planck ÿ 0.63x 10-26 ÿ 0.63x 10-26+50 ÿ 0.63x 10+24 ÿ

avogrado ÿ 0.602 x 1024 ÿ 0.602 x 1024 +50 ÿ 0.602 x 1074

ŷ bietarako behar duzu = 3 zifra zatikiarentzat eta 2 zifra berretzailearentzat.

14. Ordenagailu batean erabiltzen den koma mugikorreko edozein errepresentazio zenbaki erreal jakin batzuk soilik irudika ditzake; beste guztiak gutxi gorabeherakoak izan behar dira. Ap A balio erreala hurbiltzen duen gordetako balioa bada, orduan errore erlatiboa,  $r = (A - Ap)/A$  gisa adierazten da. Adierazi  $+0,4$  kantitate hamartarra hurrengo koma mugikorrean: oinarria=2 berretzailea. : alboraturik, 4 bit; esanahia, 7 bit. Zein da errore erlatiboa?

✓ Puntu finkoko bitarrean

$$\ddot{y} \ 0,4 \times 2 = 0,8 \ \ddot{y} 0$$

$$\checkmark 0,8 \times 2 = 1,6 \checkmark 1$$

$$\checkmark 0,6 \times 2 = 1,2 \checkmark 1$$

$$\checkmark 0.2 \times 2 = 0.4 \checkmark 0$$

0.4x2 berriro 0110-orduan zenbakia periodikoa da 0.0110-0110-0110-epetik 1.10-0110-0110-etc.

normalizatua x 2-1 y 10-0110-0110- zatikia irudikatzen da

✓ 7 biteko frakzioarekin 1001100

✓ Zenbakiaren gutxi gorabeherako balioa  $1.1001100 \times 2^{-1} = 110011 \times 2^{-6} = (32+16+2+1) \times 2^{-6} = 51/64 = .796875$

$$\text{Error} = (0.8 - 0.796875) / 0.8 = 0.0390625 = 0.4\%$$

4 bit baino gehiago. Zenbakia gaindituta, tartea (0,15) da. Gehiegizkoa erdia da kohinazioak -1-16/-2-1-7 orduan (-7,8) berretzaileak irudika daitezke.

15. Irudikatu Pi zenbakia doitasun bakarrean eta bikoitzean IEEE koma mugikorrean

✓ SOLUZIONI

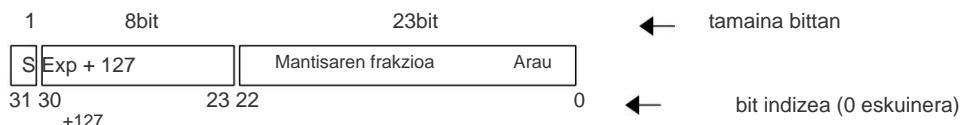
Ü Hamartar formatua: 3 1415926535897932384626433832795028841968

Ü Bitarra Jan

11.00100100001111110110101000100010000101101000110000100011010011000100110001100111000101010101010101

ÿ Puntu finkoa hamaseitarra: 0011.0010-0100-0011-1111-0110-1010-1000-1000-0101-1010-0011-0000-1000-1101-0011-0001-0011-0001-1001-1000-1010-0010-1110-0000-0011-0111

ÿ 3.243F6A8885A308D313198A2E03E



100. Irudia IEEE-754 doitasun bakarra (32 bit)

$$\text{ÿ } v = s \times 2^e \times m$$

ÿ Notazio zientifikoan mantisa normalizatuarekin eta bere zatiki zatiarekin 23 bitera moztuta eta biribildua:

$$\text{ÿ } + (1 + 0,10010010000111111011011) * 2^{+1}$$

ÿ Eremuak:

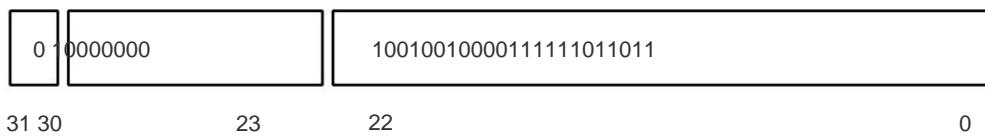
ÿ Zeinua: positiboa --ÿ 0 --ÿ 1 bit

ÿ Berretzailea:+1

ÿ Berretzaile desplazatua  $+ 127 = +1+127 = 128$  ÿ 10000000 ÿ 8 bit

ÿ Mantisa normalizatua:  $1 + 0,10010010000111111011011$

ÿ Mantisa normalizatuaren zatikia: 10010010000111111011011 ÿ 23 bit



101. Irudia IEEE-754 doitasun bakarreko formatua

$$\text{ÿ Emaitza= } 0x40490fdb$$

#### 10.4. Eragiketa aritmetikoak

##### 1. Gehitu bitar hutsean

$$\text{ÿ } 10011011+00011011 \text{ SOLUZIOA: } = 10110110$$

$$\text{ÿ } 0x3A1F+0xF4E1 \text{ SOLUZIOA: } = 0x12f00$$

$$\text{ÿ } 10011011+10011011 \text{ SOLUZIOA: } = 100110110$$

##### 2. Gehitu 2 osagarrian: 50+23

$$\text{ÿ Egin eragiketak kode bitarrean EBAZPENA: } = 0110010+0010111 = 01001001$$

$$\text{ÿ Egin eragiketak kode hamaseimalean EBAZPENA: } = 0x32+0x17 = 0x49$$

##### 3. Adierazi -66 balioa 2 osagarrian

$$\text{ÿ SOLUZIOA: } +66 = 01000010 \text{ ÿ } -66 = 10111101+1 = 10111110$$

##### 4. Kenketa 2 osagarrian: 33-66

$$\text{ÿ Egin eragiketak kode bitarrean: SOLUZIOA: } 0100001+1011110=1011111$$

$$\text{ÿ Egin eragiketak kode hamaseimalean: SOLUZIOA: } 0x21+0xBE=0xDF$$

5. Adierazi hamaseitarrez 2 osagarrian adieraz daitekeen zenbakirik handiena  
16 bitekin

ŷ SOLUZIOA: 1000-0000-0000-0000 = 0x8000

6. Gehitu 2 osagarria 16 bitekin 0x8000+0x8000

ŷ SOLUZIOA: 0x8000+0x8000=0x0000

7. 0110010 - 0010111 bitar hutsez kendu

ŷ SOLUZIOA:

0110010 <- minuend

0010111 <- kendu

1111 <- eraman

\*\*\*\*\*

0011011

8. 0x32-0x17 hamaseitarrez kendu

0x32 <- minuend

0x17 <- kendu

1 <- eraman

\*\*\*\*\*

0x1B

9. Biderketa

ŷ Zein da 2n 2ren potentzia positibo batez biderkatzearen baliokidea termino bitarretan? :

ŷ SOLUZIOA:

ŷ Zenbaki erreala batean mugitu koma posizioak eskuinera

ŷ zenbaki oso batean gehitu n zero eskuinera

ŷ Erregistro batean bitak n posizioak ezkerrera mugitza eta n zeroak eskuinera sartzea da

ŷ Zein da 2 2-n-ko potentzia negatibo batez biderkatzearen baliokidea termino bitarretan? :

ŷ SOLUZIOA:

ŷ Zenbaki erreala batean koma posizioak ezkerrera eraman

ŷ zenbaki oso batean lehen zero zero gehitu

ŷ Erregistro batean bitak n posizioak eskuinera mugitza eta ezkerretik n zeroak sartzea da

10. Egin zenbaki natural hauen biderketa:

ŷ 1010\*1010

ŷ 1010\*1111

ŷ SOLUZIOA:

$\begin{array}{r} 1010 \\ \times 1010 \\ \hline \end{array}$	$\begin{array}{r} 1010 \\ \times 1111 \\ \hline \end{array}$
*****	
$\begin{array}{r} 0000 \\ 1010 \\ \hline \end{array}$	$\begin{array}{r} 1010 \\ 1010 \\ \hline \end{array}$

0000	1010
1010	1010
*****	*****
1100100	10010110

## 10.5. Eragiketa Logikoak

1. Egin  $\tilde{A}$ ,  $\tilde{A}+1$ ,  $A+B$ ,  $A \cdot B$ ,  $A \circ B$  eragiketa logikoak  $A=10101010$  eta  $B=11110000$ -ekin

ÿ SOLUZIOA:

ÿ  $A+B$  ÿ  $A|B$  ÿ  $AVB$

ÿ  $A \cdot B$  ÿ  $A \& B$  ÿ  $A \circ B$

ÿ  $\tilde{A} = 01010101$

ÿ  $\tilde{A}+1 = 01010110$

ÿ  $A+B = 11111010$

ÿ  $A \cdot B = 10100000$

ÿ  $A \circ B = 01011010$

2. 20 biteko eragigai emanda, adierazi zer eragiketa logikoa egin behar den: (eragiketa eragigaiarekin kode hamaseimalean adierazi)

ÿ Ezarri 7. bit (7. posizioa)

ÿ SOLUZIOA: 20 bit 5 zifra hex dira ÿ Operandoa | 0x000080

ÿ Garbitu 15. bit

ÿ SOLUZIOA: Operandoa & 0xF7FFF

ÿ Toogle bit 19

ÿ SOLUZIOA: Funcionamendua ÿ 0x8000

ÿ Ezarri hitz osoa

ÿ SOLUZIOA: Operandoa | 0xFFFF

ÿ Garbitu hitz guztiak

ÿ SOLUZIOA: Operandoa ÿ Operandoa edo Operandoa & 0x000000

3. SAR X,n (desplazamendu aritmetikoa eskuinera X, n), SLR X,n (desplazamendu logikoa eskuinera desplazamendu logikoa ezkerrera X, n) eragiketa logikoak emanda, non X eragiketa den, xxR eskuinera esan nahi du, xxL ezkerrera eta n da. aldatu beharreko postu kopurua. Egin eragiketa hauek  $A=10101010$  eragigaiarekin: SAR A,4, SLR A,4, SAL A, 4 eta SLL A,4 eskuz, C hizkuntzako programa bat erabiliz eta GDB araztailea ere erabiliz.

ÿ SAR A,4 = 11111010

ÿ SLR A,4 = 00001010

ÿ GATZA A,4 = 10100000

ÿ SLL A,4 = 10100000

4. Egin eskuz A-22 eta A-2-2 biderketa non  $A=10101010$  lehenengo aritmetikoki eta ondoren, eragiketa logikoen bidez.

ÿ  $A-22 = 1010101000$

ÿ  $A-2-2 = 101010,10$

ÿ Eragiketa logikoekin, A-22 eragiketa

Yo. Bikoitzu A ý D:A ý 0000000001010101

ii. mugitu SLL D:A,2 ý D:A ý 0000000101010100

## 10.6. Jarraibideen irudikapena

- Izan bedi 32 biteko hitzak dituen ordenagailu bat. CPUak 64 eragiketa bakarreko instrukzio (eragiketa) ditu, 32 biteko erabilera orokorreko erregistroak eta erregistrarra zuzeneko helbideratze-aukera (eragindu-eremua zuzenean erregistroa da) edo zeharkako helbideratza oinarrizko erregistrarra mugimenduarekin. a) Ordenagailu honen instrukzio-formatua diseinatu. Memoria-helbideetarako erregistro bat eta desplazamendu-balio bat zehaztu behar dira, helbideratze-moduaz eta opcodeaz gain. b) Zein da desplazamenduaren balio maximoa (desplazamendua C2ko zenbaki bat da)?

ý SOLUZIOA

to. Formatua

32 biteko Word ý 32 biteko Erabilera Orokorreko Erregistroak eta 32 biteko IR Instrukzio Erregistroa

Argibide-formatua 4 eremutan egitura duena: eragiketa-kodea, helbideratze-modua, operando-eremua (erregistroa edo txandakako erregistroa)

1. eremua: eragiketa kodea: 64 argibideak: 26 ý 6 bit
2. Eremua: helbideratze modua: 2 mota: 21 ý 1 bit
3. Eremua: erregistroa: 32 erregistro: 25 ý 5 bit
4. eremua: Osoko desplazamendua: (32-(6+1+5))bit ý 20 bit

Kod. Op.

Operandoa (erregistroa)

Kod. Op.

Operandoa (Oinarrizko Erregistroa)

Operandoa (Desplazamendu-erregistroa)

b. 20 biteko desplazamendua

ý 2 osagarrian: 2 19-1 balioaren 0111-1111-1111-1111-1111-1111 balio positibo maximoa 2 19-1 eta zeinua aldatu duen gutxieneko balio negatiboa 1000-0000-0000 0-1000- 0000 da. 0000-0000 balioa +219, beraz, barrutia [+219-1,-219] ý [+524287,- 524288] da

2. 24 biteko hitzak dituen ordenagailu batek 16 eragiketa bakarreko instrukzio ezberdin, helburu orokorreko 8 erregistro eta 3 helbideratze modu ditu (zuzeneko erregistrarako, zeharkako erregistroarekin eta zeharkako erregistro-oinarrirako desplazamenduarekin).

to. Diseina ezazu ordenagailu honetarako argibide-formatu bat. Eragiketa kodea zehaztu behar duzu helbideratze modua, erregistro bat eta desplazamendu bat.

ý SOLUZIOA:

Word Size = 24 ý 24 biteko helburu orokorreko erregistroak eta 24 biteko IR instrukzioen erregistroa

4 eremu instrukzio formatuan:

Kod. Op.	Helbideratzeko modua	Grabatu	Desplazamendua
----------	----------------------	---------	----------------

1. eremua: eragiketa kodea: 16 argibideak: 24 ÿ 4 bit
2. eremua: helbideratze modua: 3 mota: 22 ÿ 2 bit 3 motetarako (Direct Reg, Zeharkako Reg eta Zeharkako RegyDesp)
3. Eremua: erregistroa: 8 erregistro: 23 ÿ 3 bit
4. eremua: Osoko desplazamendua:  $(24-(4+2+3))$ bit ÿ 15 bit

b. Zein da desplazamendu-balioen tarte magnitudean? Eta C2-n?

ÿ Magnitude formatua: Gutxieneko zero eta maximoa  $111-1111-1111-1111 = 215-1 = 32768$  ÿ 2 osagarria:

Gehienezko positiboa 0111-1111-1111-1111 2 balioarekin 14-1 eta gutxieneko negatiboa 0000-0-0000 Zeinua aldatu den -0000 0100-0000-0000-0000 da +214 balioarekin, beraz, barrutia  $[+214-1, -214]$  ÿ  $[+16383, -16384]$  da.

3. Ordenagailu batek 11 biteko instrukzio formatua du, non eragigaien eremua 4 bitekoa den.

Possible al da formatu honetan kodetzea 5 instrukzio bi eragiketarekin, 45 eragiketa batekin eta 48 eragigai gabe? Justifikatu erantzuna.

ÿ SOLUZIOA

ÿ 3 formatu mota

ÿ 1. mota: eremu mota - Op. Kodea - Op1 - Op2

ÿ 2 bit - x bit - 4 bit - 4 bit ÿ  $x=11-(2+4+4)=1$  bit ÿ Gehienez 2 instrukzio < 5 instrukzio ÿ Ezin da

ÿ 2. mota: mota eremua - Op. Code - Op

ÿ 2 bit - y bit - 4 bit ÿ  $y=11-(2+4)=5$  bit ÿ Gehienez 32 instrukzio < 45 instrukzio

ÿ Ezinezkoa

ÿ 3. mota: eremu mota - Op. Kodea.

ÿ 2 bit - z bit ÿ 48 argibide

ÿ Alternatiba: 3 IR Erregistro , mota bakoitzeko bat

ÿ Mota eremurik gabe:  $5+45+48=98$  argibide ÿ 2  
 $7+4+4=15$  bit > 11 ÿ Ezinezkoa

<sup>7</sup> ÿ 7 bit ÿ 1 motako instrukzioak okupatuko luke

4. 16 bit-ko hitz-zabalera duen ordenagailu bat (argibideak, memoria-hitza, erregistroak) eta 8. erregistroak, jarraibideen errepertorio hau du:

ÿ 14 operando bakarreko erreferentzia-instrukzio memorian, zuzeneko helbideratzeekin eta memoria zeharkakoa

ÿ Zuzeneko eta zeharkako helbideratze-moduekin bi eragiketadun 31 instrukzio erregistroa.

ÿ Eragile esplizitrik gabeko 32 instrukzio.

to. Zehaztu argibideen kodenak.

b. Zehaztu helbideratze-mota bakoitzean eskura daitekeen memoria-eremua eta tarte posiblea  
Eragileen balioak ( $C'2-n$ ).

## ŷ SOLUZIOA

- ŷ Hitzaren Tamaina = 16 ŷ 16 biteko Erabilera Orokorreko Erregistroak
- ŷ 3 formatu motako errepertorioa
- ŷ 1. Mota: Mota-OpCode-Modua Helbidea-Op1 ŷ 14 instrukzio (24 ), Bits:2-4-1-x ŷ x=16-  
(2+4+1)=9bit
- ŷ 2. Mota: Mota-OpCode-Modua Helbidea1-Op1-Modua Helbidea2-Op2 ŷ 31 instrukzio (25 ) Bit:2-5-1-x 1-x ŷ x=(16-  
(2+5+2 ))/ 2=3bit
- ŷ 3. Mota: Type-Op.Code. ŷ 32 argibide (25 ) Bit: 2-5 ŷ 16tik 7 bit okupatuta.

5. Motorola M68000 prozesadorean oinarritutako ordenagailu batek honako edukia aurkezten du  
erregistroa eta memoria:

ERREKOAK		MEMORIA	
Grabatu	Edukitua	Helbidea	Edukitua
A1	100	99	104
A2	2	100	108
		101	106
		102	107
...	...	...	...
		199	100
		200	3. 4
		201	96
		202	201

ŷ Exekutatzen den instrukzioaren desplazamendu-edukia after=99 bada, zein izango litzateke balioa?  
operandoa (byte-tamaina) helbideratze-modu hauekin?

- to. Zuzeneko memoriatik edo absolututik (helbidea = desplazamendua).
- b. Zuzeneko erregistroa A1-rekin.
- c. Zeharkako erregistroa A1-ekin.
- d. Zeharkakoa A1 oinarrizko erregistroarekin desplazamenduarekin
- eta. Zeharkako desplazamenduarekin oinarri-erregistroarekin A2.
- F. Zeharkakoa desplazamenduarekin A1 oinarri-erregistroarekin eta A2rekin indexatua.
- g. Zeharkako inskripzioa A1-rekin aurre-dekretuarekin.

ŷ SOLUZIOA:

Zuzeneko memoriatik edo absolututik (helbidea = offset) -> M[99]=104  
**OHARRA:** Lokatzaileak eragigaiaren helbide absolutua ebatzen du a zehatzuz  
desplazamendua segmentuaren hasiera adierazten duen erregistro baten aldean edo  
argibideak atalean. Modu honetan eragigaien eremua baino laburragoa da  
helbide absolutua ezartzea.

Zuzeneko erregistroa A1-rekin. -> R[A1]=100  
Zeharkako erregistroa A1-ekin. -> M[A1]=M[100]=108  
Zeharkako desplazamenduarekin oinarrizko erregistroarekin A1 -> M[A1+99]=M[199]=100

Zeharkako desplazamenduarekin oinarrizko erregistroarekin A2.-> M[A2+99]=M[101]=106  
 Zeharkako desplazamenduarekin A1 oinarri-erregistroarekin eta A2.->rekin indexatua  
 $M[A1+99+A2]=M[100+99+2]=M[201]=96$   
 Zeharkako erregistroa A1.-> M[A1-1]=M[100-1]=104-rekin aurre-gutxitzearekin

6. Ordenagailu batek erregistro eta memoria eduki hauetan ditu:

ERREKOAK		MEMORIA	
Grabatu	Edukia	Helbidea	Edukia
R1	99	96	100
R2	6	97	102
		98	101
		99	104
		100	108
		101	106
		102	107
		103	109
		104	110

ÿ Exekutatzen den instrukzioaren desplazamendu-edukia 96 bada, zein izango litzatekeen balioa honako helbide hauetan funtzionatzen?

- a)Memoria zuzena (dir = desp).
- b)Zeharkako memoria (memory dir = desp).
- c)R1eko zuzeneko erregistroa.
- d)Zeharkako erregistroa R1n.
- e)Zeharkako desplazamenduarekin R2 oinarri-erregistroarekin

7. 32 biteko hitz-zabalera duen ordenagailua eta 32 erregistroko erregistro-bankua daukazu.

32 bit. Ordenagailuak 64 argibide ezberdin eta helbideratze modu hauetan ditu:

Instrukzio formatu bat memoria zuzeneko helbideraketarekin eta zeharkako memoria helbideratzearekin eta bestea zuzenbide zeharkako formatua oinarri-erregistrarra aldatzearekin.

to. Diseina ezazu bi eragiketen instrukzioen bi formatuak beti eragiketa bat dela jakinda memoria dago eta beste bat erregistroan.

ÿ SOLUZIOA:

ÿ CodOp/Mode/Source\_Memory/Destination\_Register

ÿ CodOp=26=64 argibideak

ÿ Modua: zuzena edo zeharkakoa:21

ÿ Memoria=x helbide-bit

ÿ Reg=25=32 erregistro ÿ 5 bit

ÿ IR totala =32 bit ÿ Memoria=32-(6+1+5)=32-12=20 bit

ÿ CodOp/Desplazamiento-Iturria\_Erregistroa/Helmuga\_Erregistroa

ÿ CodOp=26=64 argibideak

- ÿ Desplazamendua = x helbide-bit
- ÿ Reg\_base=25=32 erregistro ÿ 5 bit
- ÿ Erregistroa=25=32 erregistro ÿ 5 bit
- ÿ Guztira=32 bit ÿ Desplazamendua=32-(6+5+5)=16 bit

b. Memoria-helbide bakoitzak byte bat erreferentzia egiten badio, zer memoria-eremurekin sar daiteke helbideratze-modu bakoitza?

ÿ SOLUZIOA

8. Ikus ditzagun lau prozesadore-arkitektura: metagailua, pila, memoria-memoria eta erregistro-erregistroa. 16 erregistrorekin. Lau arkitekturarentzat datu komun hauek daude:

- ÿ Kodea beti bytekoa da
- ÿ Memoria-helbide guztiak 2 bytekoak dira
- ÿ Datu guztiak 4 bytekoak dira
- ÿ Argibide guztiak byte kopuru oso baten berdina dute
  - to. Idatzi orokorean programak mihiztatzaile hizkuntzan arkitekturak honako eragiketa hau egiteko; A=B+C. Programa bakoitzerako, kalkulatu trafikoa memoria eta kodearen tamainarekin. Zein da eraginkorragoa?
  - b. Idatzi A=B+C eragiketa-segida honetarako lau muntaketa-programak; B=A+C ; D=AB Kalkulatu memoria-trafikoa eta kodearen tamaina. Zein da eraginkorragoa?

9. Kontuan izanda prozesadore batean instrukzio bakoitzak bilatzen duela eta eragiketa baterako sarbide bakoitzak ziklo bat kontsumitu eta 3 programatarako kontuan hartuz hurrengo datuak milioika erreferentziak.

	TEX	Espezia	C
RR Arkitektura			
Datu-erreferentziak	5.4	4.9	1.4
Instrukzio hitzak	14	18.9	3.9
MM arkitektura			
Datu-erreferentziak	12.4	10.5	4.1
Instrukzio hitzak	7.5	8.4	2.4

to. Kalkulatu hiruko argibideak bilatzeko egiten diren memoria sarbideen ehunekoa  
RR arkitekturarako eta MM arkitekturarako programak.

b. Zein da guztizko sarbideen erlazioa bi arkitekturen artean?

10. Motorola M68000 32 biteko arkitekturarako, bistaratu erregistro guztien edukia eta Instrukzio bakoitzaren exekuzioak eragindako memoria posizioak (PC barne), beti hasierako baldintza hauetatik abiatzen garela suposatuz:

Argibideak:	
to. CLR.L	-(A1)
b. CLR.W	D2
c. MUGITU.W	\$ 1204, D1
d. MUGITU.W	## 1204,D1
eta. MUGITU.B	(A2)+, \$1200

Argibideak:	
F. MUGITU.L	D1,-(A2)
g. MUGITU.L	(A1)+,D2

Hasierako baldintzak:	
ERREKOAK	MEMORIA
A1:00001202	0011FE:7777
A2:00001204	001200:1111
D1:01020304	001202:2222
D2:F0F1F2F3	001204:3388
	001206:4444
	001208:5555
	00120A:6666

ÿ Atzizkiak ÿ Luzea=4 byte, Hitza=2byte, Byte=1byte

ÿ Helbide eraginkorren gehikuntza edo gutxitzea eragigaiaren tamainarekin eskalatzen da

ÿ SOLUZIOA

CLR.L -(A1):

Garbitu eragiketa luzea

A1 erregistroaren aurrekaltetzea eta jarraian zeharkakoa

A1<-A1-4 ;(A1-4=0x1202-0x4=0x11FE) A1:000011FE

M[A1]<-0,M[A1+1]<-0,M[A1+2]<-0,M[A1+3]<-0 M[0011FE]:0000 M[001200]:0000

CLR.W D2:

Hitz eragile argia

D2(15:0)<-0 ; D2:F0F10000

MOVE.W \$ 1204,D1

Kopiatu 2 byte Source\_Op-tik (zuzeneko helbidea) Destination\_Op-era (erregistroa)

D1(15:0)<-M[0x1204] ;(M[0x1204]=3388); D1:01023388

MUGITU.W #\$1204,D1

Kopiatu 2 byte Source\_Op-tik (Berehalako helbidea) Destination\_Op-era (erregistroa)

D1(15:0)<-0x1204 ; (D1:01021204)

MOVE.B (A2)+, 1200 \$

Kopiatu 1 byte Source\_Op (zeharkakoa osteko gehikuntzarekin), Destination\_Op (Zuzenean)

0x1200<-M[A2][LSB] ;(M[A2][LSB]=M[A2+1]=M[1205]=88) ; M[1200]:1188

A2<-A2+1 ;(A2+1=0x1204+0x1=0x1205) ; A2:00001205

MUG.L D1,-(A2)

Kopiatu 4byte Op\_source (Erregistratu) Op\_destination (zeharkakoarekin dekretua)

A2<-A2-4 ;(A2-4=0x1204-0x4=0x1200) A2:00001200

M[A2+3]<-D1(7:0),M[A2+2]<-D1(15:8),M[A2+1]<-D1(23:16),M[A2]<-D1(31:24)

; M[001200]:01020304

MUGI.L (A1)+,D2

Kopiatu 4byte-ko Op\_source (Zeharkakoa postincrementarekin) hona Target\_op(Erregistratu)

D2(7:0)<-M[A1+3] ; D2(15:8)<-M[A1+2]; D2(23:16)<-M[A1+1]; D2 (31:24)<

M[A1] ; D2:11112222  
 A1<-A1+4; (A1=0x1202+0x4=0x1206) A1:00001206

11. Erakutsi kaltetutako erregistro eta memoria-kokapen guztiak edukia (PC barne).  
 argibide bakoitzaren exekuzioa, beti baldintzetatik abiatzen garela suposatuz  
 zehaztutako inizialak:

Argibideak:	to. MUGI.W -(A1),A3
b. CLR.B -11(A2)	c. MUG.W (A4)+,-100 (A1,D5.W)

Hasierako baldintzak:	
ERREKOAK	MEMORIA
A1:00001504	001500:1234
A2:00001510	001502:5678
A3:11122233	001504:9ABC
A4:00001506	001506:EF11
D5:FA000064	001508:2233
	00150A:4455

12. Motorola-ren M68000-32 arkitekturarako, hasierako baldintza hauek betetzen direla suposatuz,  
 Eragindako erregistro eta memoria-kokapen guztiak edukia bistaratuz (PC barne).  
 instrukzio bakoitzaren exekuzioa. Demagun, gainera, argibideak posizioetan daudela  
 memoria jarraian exekutatzen da, \$ 2000 helbidetik hasita, eta sekuentzian exekutatzen da.

Hasierako baldintzak:	
ERREKOAK	MEMORIA
A1:00001504	001500:1234
A2:00001510	001502:5678
A4:00001506	001504:9ABC
D3:11122233	001506:EF11
D5:FA000070	001508:2233
D6:AB00FF9B	00150A:4455

Argibideak:	CLR.B -(A4)
MUG.L -124(A2, D5.W), -(A1)	MUGITU.W \$64(A4,D6.W), D3

13. Konparatu 1, 2 eta 3 bideko ordenagailuak adierazpena kalkulatzeko programak idatziz  
 $X = (A+B*C)/(DE^F)$  horietako bakoitzaren erreperitorioak honako hauek izanik:

0 Helbidea	1 Helbidea	2 Helbidea	3 Helbidea
BULTZATU M	KARGA M	MUGITU X,Y ;(X<-Y)	MUGITU X,Y ;(X<-Y)
POP M	DENDA M	GEHITU X,Y ;(X <- X+Y)	GEHITU X,Y ;(X <- Y+Z)
A.D.D.	GEHITU M	AZPI X,Y ;(X <- XY)	AZPI X,Y ;(X <- YZ)
AZPI	AZPI M	MUL X,Y ;(X <- X*Y)	MUL X,Y ;(X <- Y*Z)
MUL	MUL M	DIV X,Y ;(X <- X/Y)	DIV X,Y ;(X <- X/Y)

DIV

DIV M

ÿ SOLUZIOA:

BUKATU A	ZARGA E	MOV R0, E	MUL R0, E, F
BULTZATU B	MUL F	MUL R0, F	AZPIA R0, D, R0
BULTZATU C	DENDA T	MOV R1, D	MUL R1, B, C
MUL	ZARGA D	AZPIA R1, R0	GEHITU R1, A, R1
A.D.D.	AZPI T	MOV R0, B	DIV X, R0, R1
BULTZATU D	DENDA T	MOV R0, C	
PUSH E	ZARGA B	GEHITU R0, A	
BULTZATU F	MUL C	DIV R0, R1	
MUL	GEHITU A	MOV X, R0	
AZPI	DIV T		
DIV	S.T.O. X		
POP X			

14. 8 biteko hitza duen ordenagailu batean egindako azken eragiketa biak dituen gehiketa izan balitz  
eragigaiak 00000010 eta 00000011 ziren, zein izango lizateke honako bandera hauen balioa?

ÿ Eraman

ÿZero

ÿ Gainezka

ÿ Sinatu

ÿ Parekidetasuna

ÿ Erdi-Eramatea

ÿ SOLUZIOA:

ÿ 0010+0011=0101 ÿ MSBn ez dago baliorik, emaitza ez da zero, ez dago gainezka egiten geroztik  
ez dago led, positibo, bikoitien kopururik, ez dago ledrik 3. posizioan. Beraz  
Bandera guztia desgaituta, parekotasun bandera izan ezik. Parekidetasunaren bandera 1ean izango da.

15. x86 Compare instrukzioak (CMP) iturburuko eragigaia helmugako eragigaitik kentzen du; Elementua  
egoera-markak eguneratzen ditu (C, P, A, Z, S, O) baina ez ditu eragigaietako bat ere aldatzen. CMP argibideak  
helmuga eragigaia iturria baino handiagoa, berdina edo txikiagoa den zehazteko erabil daiteke  
operandoa. CMP exekutatzeari bi gainezka-marka daude EFLAGS erregistroan: CF banderak ez du egiten  
Carrier Flag funtzioa baina gainezka-bandera funtzioa sinatu gabeko zenbakiak alderatzen badira. Konparatzen badute  
sinatutako zenbakiak overflow bandera OF da. Irakurri baldintzapeko argibideei buruzko oharrak

to. Demagun bi eragigaiak zeinu gabeko zenbaki oso gisa tratatzen direla. Erakutsi zein egoera-markari dagozkion garrantzitsuak  
zehaztu bi zenbaki osoen tamaina erlatiboa eta banderen zer balio duten baino handiagoak diren,  
berdina edo txikiagoa.

b. Demagun bi eragigaiak biko osoko osagarri gisa tratatzen direla. Erakutsi zein egoera-marka  
garrantzitsuak dira bi zenbaki osoen tamaina erlatiboa eta banderen zer balio duten zehazteko  
baino handiagoa, berdina edo txikiagoa.

c. Nola aldatzen dira SF zeinua eta OF gainezkatzea 1 byte sinatutako eragigaietarako: eragigaia bada  
helmuga 0x80 da eta iturburu-eragilea 0x7F da; helmuga eragigaia 0x7F bada eta iturburua  
0x81.

ÿ SOLUZIOA:

ÿ Ikus Mihiztadura Lengoaiaren Programazioa (x86),

ÿ CMP ÿ Dest - Iturria

ÿ a. Sinatu gabeko zenbaki osoak

14. taula. CMP/EFLAFS

Kasua	C.F.	PF	AF	ZF	S.F.	OF
Dest>Iturria	0	x	x	0	x	x
Dest=Iturria	0	x	x	1	x	x
Dest<Iturria	1	x	x	0	x	x

ÿ CF gainezkatze-marka da kasu honetan.

ÿ Sinatu gabe ÿ Helmuga > Iturria kasurako, Helmuga - Iturria kenketak CF=0 eta OF sortzen ditu.  
sinatutako kenketaren arabera. Seinalearen banderak ez du SF=X ezartzen.

ÿ Sinatu gabe ÿ Helmuga < Iturria kasurako, Helmuga - Iturburua kenketak CF=1 eta OF sortzen ditu.  
sinatutako kenketaren arabera. Seinalearen bandera ez da aldatzen SF=X.

ÿ b. Zenbaki oso sinatuak

15. taula. CMP/EFLAFS

Kasua	C.F.	PF	AF	ZF	S.F.	OF
Dest>Iturria	x	x	x	0	0/1	0/1
Dest=Iturria	0	x	x	1	0	0
Dest<Iturria	1	x	x	0	0/1	1/0

ÿ SF eta OF-ren arteko harremana

ÿ 1 byte datuak: A > B adibidea

1 Byte: gehienezko balioa +127, gutxieneko balioa -128 eta eragiketak  
 $2^8=256$  moduluko ordenagailua  
A=+127 eta B= -8 -> A > B bada  
CMP instrukzioa -> AB=+127-(-8)=+135  
+135 > +127 -> OF=1  
Ordenagailuak +135 256 modulo kalkulatzen du, hau da, A emaitza  
B=+135-modulua=+135-256=-121 -> zeinu negatiboa -> SF=1  
OF eta SFren arteko erlazioa OF=SF da beti A > B kasurako

ÿ 1 byte datuak: A < B adibidea

1 Byte: gehienezko balioa +127, gutxieneko balioa -128 eta eragiketak  
 $2^8=256$  moduluko ordenagailua  
A=-128 eta B= +8 -> A<B bada  
CMP instrukzioa -> AB=-128-(+8)=-136  
-136 < -128 -> OF=1  
Ordenagailuak -136 256 modulo kalkulatzen du, hau da, AB=- emaitza.  
136+modulua=-136+256=+120 -> zeinu positiboa -> SF=0  
OF eta SFren arteko erlazioa OF<>SF da beti A < B kasurako

ŷ interpretazio grafikoa: deskribatu aurreko bi adibideak, A eta B bektoreak lerro zuzen batean edo zirkulu batean kokatuz eta balio maximo, minimo eta moduluen balioa markatuz.

ŷ OF gainezkako bandera da.

ŷ Zeinuarekin ŷ Helmuga > Iturria kasurako, Helmuga - Iturburua kenketak CF-a sortzen du.  
zeinurik gabeko kenketa. SF=0 edo 1. OF=SF

ŷ Zeinuarekin ŷ Helmuga < Iturria kasurako, Helmuga - Iturburua kenketak CF-a sortzen du.  
zeinurik gabeko kenketa. SF=0 edo 1. OF<>SF

ŷ ONDORIOA

CMP-k bi kasuak aldi berean egiten ditu (sinatutakoak eta sinatu gabeak), beraz, CF eta OF gainezkatze-banderei eragiten die Helmuga > Iturburu kasurako, Helmuga -  
Iturria kenketak sortzen du.  
SF=OF eta  
CF=0 Helmuga < Iturria kasurako, Helmuga - Iturria kenketak sortzen du.  
SF<>OF eta CF=1

16. CF eta OF banderen interpretazioa ADD batuketa aritmetikoko instrukzioarekin. SUB-arekin gertatzen den bezala, CF banderak sinatu gabeko batuketa aritmetikoaren gainezkatzeari erantzuten dio eta OF banderak sinatutako batuketa aritmetikoaren gainezkaari erantzuten dio. Egiaztatu programarekin:

```
#### CF-k eta OF-k gainezkadurarekin duten erlazioaren azalpena #### ADD
eta SUB instrukzioek ERAGIKIN eta GABE IKUSKIZUNAREN ALDIBERAKO interpretazioa egiten dute #### CF
eragiketaren
gainezkaketaren emaitza da (gehitu ,sub,etc) WITHOUT ikurra #### OF eragiketaren gainezkatzearen
emaitza da (gehitu,sub,etc) WITH zeinuarekin #### Zein eragiketa aritmetikotan CF EZ da sinatu gabeko
gainezkatzea eta BAI CARRY hutsa da ?
```

```
.section .data oper1: .byte
0x80 # sinatuta -128 eta sinatu gabe 256 oper2: .byte 0x7F # sinatuta
+127 eta sinatu gabe 255
```

```
oper3: .byte 0x00 # sinatuta 0 eta sinatu gabe 0 oper4: .byte
0xFF # sinatuta -1 eta sinatu gabe 511
```

```
.atala .testua
.global main main:
push %ebp mov %esp,
%ebp
```

```
mov oper1,%al mov
oper2,%bl mov
oper3,%cl mov oper4,%dl
```

```
subb $0x01,%al # egiaztatu zeinurik gabe ez dagoela gainezkarik (CF=0) eta zeinuarekin gainezka dagoela (OF=1)
addb $0x01,%bl #
egiaztatu zeinurik gabe ez dagoela gainezkarik (CF=0 ) eta bai zeinuarekin
```

```
ginezka dago (OF=1)
```

```
subb $0x01,%cl # egiazta zeinurik gabe ginezka dagoela (CF=1) eta zeinuarekin ez dagoela ginezkarik (OF=0) addb  
$0x01,%dl # egiazta  
zeinurik gabe ginezka dagoela (CF=1) eta zeinuarekin ez dago ginezkarik (OF=0)
```

```
mov %esp,%ebp pop  
%ebp  
ret  
.amaiera
```

17. Mikroprozesadorearen instrukzio-multzo askok baldintza bat probatzen duen eta helmuga-eragile bat ezartzen du baldintza egia bada. Adibideen artean, SETcc x86 prozesadorean, Scc Motorola MC68000 prozesadorean eta Scond National NS32000 prozesadorean daude. [Intel eskuliburu azkarra](#): interpretatu SETcc instrukzioaren nmemonikoak eta eragigaiak

to. Argibide hauen artean desberdintasun batzuk daude:

- ÿ SETcc eta Scc byte batean bakarrik funtzionatzen dute, Scond bytean, hitzean eta hitz bikoitzean funtzionatzen dute. eragigaiak.
- ÿ SETcc eta Scond-ek eragigai **bat osoko moduan ezartzen dute egia bada** eta **zerora gezurra bada, hau** da, logika positiboa. Scc-k bitar guztiak ezartzen ditu egiazkoak badira eta zero guztiak faltsuak badira. Zeintzuk dira desberdintasun horien abantaila eta desabantaila erlatiboak?
- b. Instrukzio hauetako batek ere ez du baldintza-kodeen markarik ezartzen, eta, beraz, instrukzioaren emaitzaren proba esplizitua behar da haren balioa zehazteko. Eztabaideitu instrukzio (proba) honen ondorioz baldintza kodeak ezarri behar diren. [Intel eskuliburu azkarra](#): TEST instrukzioa interpretatu.
- c. IF b > a THEN bezalako IF adierazpen simple bat implementatu daitete zenbakizko irudikapen-metodo bat erabiliz, hau da, balio boolearra agerraraziz (memorian aldagai boolearra erabiliz), kontrol-metodo fluxu baten aurka, balioa adierazten duena. Adierazpen boolearra programan lortutako puntu baten bidez (baldintzazko jauziekin). (Lehenengo transkribatu adierazpenaren ASM hizkuntza RTL hizkuntzara). Irakurri [baldintzapeko argibideei](#) buruzko oharrak. Interpretatu argibideak: [Intel quick manual](#). Konpiladore batek IF b > a THEN hurrengo **x86 kode** honekin implementatu dezake kontrol-metodoa ezartzeko:

; Intel Sintaxia: Operation DestinationOp, SourceOp

**SUB CX, CX ;ezarri** CX erregistroa 0

**MOV AX, B ;mugatu** B kokapenaren edukia AX erregistratzeko

**CMP AX, A ;** konparatu AX erregistroko edukiak eta A kokapena

**JLE TEST ;jauzi** Target\_Op < Source\_Op edo Target\_Op=Source\_Op -> B<A bada  
edo B=A

**INC CX ;gehitu** 1 erregistroaren edukiari CX

**PROBA: JCXZ OUT ;jauzi** egin CXren edukia 0 berdina bada

**GERO: XXXXX ;** B>A

**IRTEERA: XXXXX ;** B<A edo B=A

ÿ Garatu if.\_ba.s programaren iturburu-modulu: editatu kode berria eta iruzkinak, konpilatu iturburu-modulu eta exekutatu modulu bitarra urratsez urrats araztearekin programak behar bezala funtzionatzen duen arte.

ÿ Idatzi programa hau bera memoria eta programazio denbora aurrezten duen SETcc instrukzioa erabiliz. exekuzioa.

d. Orain kontuan hartu goi-mailako hizkuntza adierazpena, (lehenik transkribatu adierazpenaren ASM hizkuntza RTL hizkuntzara):

ÿ A:=(B > C) EDO (D == F) non := esleipena esan nahi du

ÿ Konpiladore batek kode hau sor dezake:

```

ÿ MOV EAX, B ;mugatu B kokapenaren edukia EAX erregistratzeko
ÿ CMP EAX, C ; konparatu EAX erregistroko edukia eta C kokapena
ÿ MOV BL, 0 ;0-k faltsua adierazten du -> MOV instrukzioak EZ die erregistroei eragiten

ÿ JLE N1 ;jauzi egin B<C edo B=C bada
ÿ MOV BL, 1 ;1 egia adierazten du
N1 MOV EAX, D
ÿ CMP EAX, F
ÿ MOV BH, 0 ;0k faltsua adierazten du
ÿ JNE N2 ;jauzi F<>D bada
ÿ MOV BH, 1 ;1 egia adierazten du
N2 EDO BL, BH
```

ÿ Garatu if\_doble\_condicion.s programaren iturburu-modulu: Editatu kode berria eta iruzkinak, konpilatu iturburu-modulu eta exekutatu modulu bitarra urratsez urrats araztearekin programak behar bezala funtzionatzen duen arte.

ÿ Zein da BLren balioa gezurrezko eta egiazko kasuetarako?

ÿ Idatzi programa hau bera memoria eta denbora aurrezten dituen SETcc instrukzioa erabiliz exekuzioaren.

ÿ SOLUZIOA:

ÿ a. **JLE TargetOp**

ÿ Jauzi egiten du azken emaitzak ZF=1 bandera aktibatzen badu edo SF eta OF banderak desberdinak badira (SF<>OF)

ÿ **CMP Destination\_Op\_B, Source\_Op\_A** ÿ BA ÿ sinatutako eragiketak bi osagarrian

ÿ ZF=1 ÿ B==A

ÿ SF <> OF

ÿ SF=1,OF=0 ÿ B<A

ÿ SF=0,OF=1 ÿ B>A

ÿ Ordeztu **JLE Op\_destination** saltoa **SETcc**- rekin

```

SUB CX, CX ;ezarri CX erregistroa 0
MOV AX, B ;mugatu B kokapenaren edukia AX erregistratzeko
CMP AX, A ; konparatu AX erregistroko edukiak eta A kokapena
```

**SETGE CX ;CX** = (b GE a) logika positiboan

PROBA: **JCXZ OUT** ;jauzi egin CXren edukia 0 berdina bada; egia bada

GERO: **XXXXXX**

IRTEERA: **XXXXXX ;bai** gezurra

ÿ Garatu if\_doble\_condicion.s programaren iturburu-modulu: Editatu kode berria eta iruzkinak, konpilatu iturburu-modulu eta exekutatu modulu bitarra urratsez urrats araztearekin programak behar bezala funtzionatzen duen arte.

ÿ b.

```
MOV EAX, B ; mugitu B kokapenetik EAX erregistratzera
CMP EAX, C
SETG BL ; Setcc SETGreater ;BL = 0/1 emaitzaren arabera B>C
MOV EAX, D
CMP EAX, F
MOV BH, 0
SETE BH ; Setcc SETEqual ;BH= 0 edo 1 emaitzaren arabera D=F
EDO BL, BH
```

ÿ BL edo BH egia bada, ebaluatu beharreko baieztapena (B > C) EDO (D == F) egia da.

ÿ BL: A aldagai logikoa logika positiboa duena

ÿ Garatu if\_doble\_condicion.s programaren iturburu-modulu: Editatu kode berria eta iruzkinak, konpilatu iturburu-modulu eta exekutatu modulu bitarra urratsez urrats araztearekin programak behar bezala funtzionatzen duen arte.

18. Hurrengo datu-egituraren, marraztu little-endian memoria-diseinua, kontuan hartuz konpilatzaileak datuak 4ren multiploko helbideekin lerrokatzen dituela, hutsuneak zeroz betez, eta, horrela, memoriaren eta CPUaren arteko transferentzia kopurua minimizatu. datuen harketan.

ÿ Adierazpena:

```
#include <stdio.h>
void main (void)

{ struct{ÿ
int a; int
pad; //ÿ b bikoitza ;
int* c; char
d[7];ÿ e
laburra ; int f;
char q[4]; }

s={.a=0x11121314,.pad=0,.b=0x2122232425262728,d={'A','B','C','D','E','F'
,'G'},.e=0x5152,f=0x61626364,q="abc"}; sc=&s.e; }
```

ÿ Little Endian SOLUZIOA:

```
00: 14 13 12 11
04: xx xx xx xx 08:
rr rr rr rr
0C: rr rr rr rr
10: aa aa aa aa
```

14:41 42 43 44  
 18:45 46 47 orr.  
 1C: 52 51 orr.  
 20: 64 63 62 61  
 24: 61 62 63 00 orr.

ÿ xx: zehaztugabea

ÿ rr: IEEE-754 doitasun bikoitzeko kodea

ÿ aa: c erakuslea hasieratzen duen e aldagaiaren helbidea.

ÿ char q[4] = "abc": 4 karaktere-elementuko array. Honen baliokidea da: char q[4] = {a, b, c, NULL} non NULL karakterea 00 den.

19. Endian arkitektura txiki baterako memoria-helbideen mapa beheko irudian agertzen dena da. Lotu s1 eta s2 egituren deklarazioa C lengoainan eta haien hasieratzea egituren elementuen memoria helbideak adierazten dituen helbide-maparekin.

Little-endian address mapping								Byte address
07	06	05	04	03	02	01	00	
21	22	23	24	25	26	27	28	
0F	0E	0D	0C	0B	0A	09	08	
'D'	'C'	'B'	'A'	31	32	33	34	
17	16	15	14	13	12	11	10	
		51	52		'G'	'F'	'E'	
1F	1E	1D	1C	1B	1A	19	18	
				61	62	63	64	
				23	22	21	20	

102. Irudia Little Endian

to. s1 aldagaiaren deklarazioa: egitura mota

```
struct
{ i bikoitza ; //0x1112131415161718 ; 8
byte } s1;
```

ÿ SOLUZIOA:

ÿ MSB(i):0x11 0x03 helbidean eta LSB(i):18 MSB-7 helbidean

b. s2 aldagaiaren deklarazioa: egitura mota

```
struct
{ int i; //0x11121314 ; 4 byte int
j; //0x15161718 } s2;
```

ÿ SOLUZIOA:

ÿ i:

ÿ MSB(i):0x11 0x03 helbidean eta LSB(i) 0x00 helbidean

ÿ j:

ÿ j sekuentzialki erreserbatuko da i ondoren, beraz, LSB(j) helbidean egongo da  
0x04 eta MSB(j) 0x07 helbidean

ÿ 0x04: 18 17 16 15

20. Idatzi programa txiki bat makinaren endiantasuna zehazteko eta emaitzen berri eman. Exekutatu programa  
zure eskura dagoen ordenagailu bat eta aktibatu irteera.

```
#include <stdio.h> main()
{
    int
    integer; /*4 byte*/ char *p; osokoa =
    0x30313233; /
    * ASCII '0', '1', '2', '3' karaktereetarako */ p = (char *)&zenbaki osoa if (*p=='0' && *(p+1)=='1' &&
    *(p+2)=='2' && *(p+3)=='3')

    printf("Hau big endian makina bat da.\n"); bestela bada (*p=='3'
    && *(p+1)=='2' && *(p+2)=='1' && *(p+3)=='0') printf(" Hau endian makina txiki bat da.\n");

}
else
printf("Errorea logikan makinaren endian-tasuna zehazteko.\n");
```

ÿ SOLUZIOA:

ÿ p zenbaki osoko aldagaiaren lehen byterea seinalatzen du, (p+1) hurrengo byterea eta abar.

ÿ p 0 karakterea adierazten badu, MSB osoko helbide baxuenean gordeta dagoela esan nahi du ÿ  
big endian

ÿ p-k 3. karakterea adierazten badu, MSB osoko helbiderik altuenean gordeta dagoela esan nahi du ÿ  
endian txikia

## 10.7. asm programazioa

- Kode mnemotekniko oinarrizkoenen eragiketen azalpen laburra. Informazio gehiagorako  
zehatza joan INTEL Instrukzio Erreperitorio Eskuliburura

### 10.7.1. Datuak

#### 1. Interpretatu x86-64 mihiztadura-lengoaiaren programa baten jarraibide hauek deskribatuz

RTL hizkuntzan:

to. mugitu da1,da4

b. mov \$0xFF00FF00FF00FF00,%rax

c. mugitu \$0xFF,%rsi

d. mov \$da1,%rsp

eta. irakurri da1,%rsp

F. mov da4,%ebx

g. movb da4,%ebx

h. movl da4,%ax

Yo. movw %ebx,da4

j. movw %ebx,da1

ŷ datuen atalean honako kode hau aurkezten dela kontuan izanda:

```
.datuak
```

```
da1: .byte 0x0A da2:
```

```
.2byte 0xA0B da4: .4byte
```

```
0xA0B0C0D agurra: .ascii "kaixo"
```

```
zerrenda: .int 1,2,3,4,5
```

ŷ SOLUZIOA

```
mov da1,da4
mov $0xFF00FF00FF00FF00,%rax
mov $0xFF,%rsi
mov $da1,%rsp
lea da1,%rsp
mov da4,%ebx
movb da4,%ebx
movl da4,%ax
movw %ebx,da4
movq %ebx ,da4
```

#### 10.7.2. Helbideratzeko moduak

1. Deduzitu eragigaiaren helbide eraginkorra adierazpen hauetan:

to. \$0

b. %rax

c. begizta\_irten

d. datu\_elementuak(%rdi,4)

eta. (%rbx)

F. (%rbx,%rdi,4)

ŷ SOLUZIOAK

ŷ \$0: berehalakoa: eragiketa instrukzioan bertan dago, 0 da.

ŷ %rax: zuzeneko erregistroa. Eragiketa erregistroan dago. R operandoa[rax]

ŷ loop\_exit: zuzeneko memoria. Etiketa memorian dagoen operandoaren helbide eraginkorra da.

M operandoa[loop\_exit]

ŷ data\_items(%rdi,4) : indexatzea eta berehalako mugitzea. Helbide eraginkorra =

datu\_elementua+4\*RDI. M operandoa[datu\_elementua+4\*RDI]

ŷ (%rbx): Erregistratzeko zeharkakoa. Helbide eraginkorra=RBX . Negoiazioa M[RBX]

ÿ (%rbx,%rdi,4) : oinarritzko erregistroan indexatzea eta desplazatzea. Helbide eraginkorra = RBX+4\*RDI  
.M[RBX+4\*RDI] funtzionatzen

2. Deskribatu kodea RTL hizkuntzan:

```
irakurri buffer,%eax
mov da2,(%eax)
mov da2,%bx
mov %bx, (%eax)
incw da2
irakurri da2,%ebx
incw 2(%ebx)
mov $3,%esi
mov da2(%esi, 2),%ebx
```

ÿ SOLUZIOA:

```
irakurri buffer,%eax
mov da2,(%eax)
mov da2,%bx
mov %bx, (%eax)
incw da2
irakurri da2,%ebx
incw 2(%ebx)
inc 2(%ebx)
mov $3,%esi
mov da2(%esi,2),%ebx
```

### 10.7.3. Aritmetika

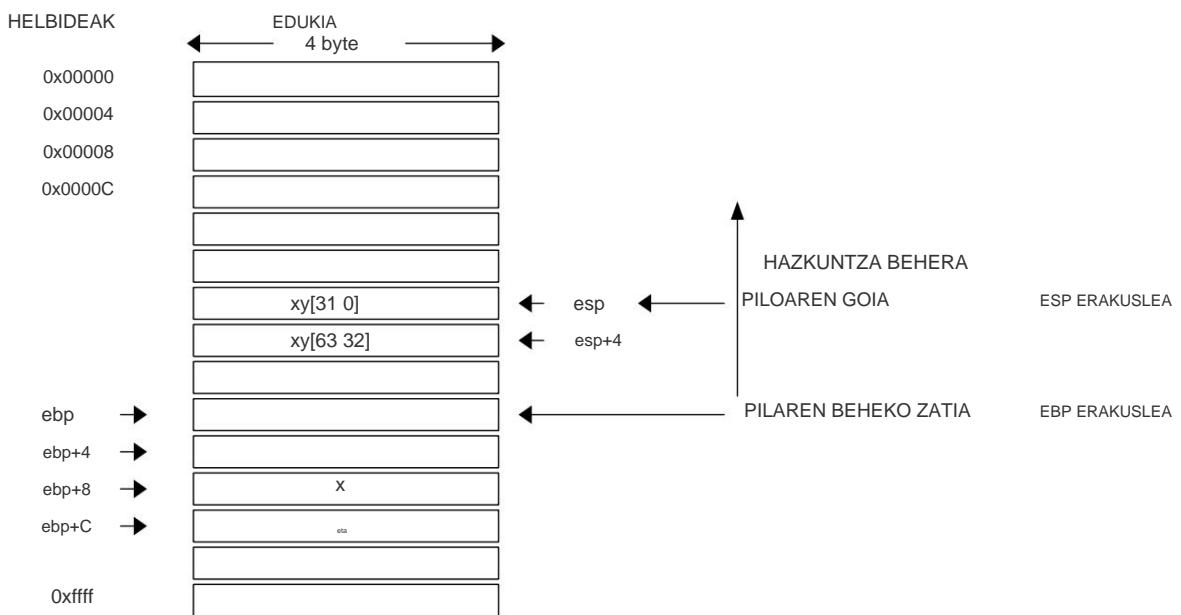
1. Demagun x eta y sinatutako zenbaki osoak %ebp erregistroarekiko 8 eta 12 posizioetan gordeta daudela eta pilaren goiko aldean  $x \times y$  8 byteko produktua gorde nahi duzula, erregistroa %esp izanik ( pila-erakuslea) pilaren goiko erakuslea. a) Garatu i386 arkitekturako gas-multzo-kodea eta b) marraztu pilaren edukia jakinda pilaren zabalera hitz batekoa dela i386 arkitekturan eta ebp erregistroa eta esp pila-erakuslea 4 hitz bereizten direla suposatz.

ÿ SOLUZIOA: Asm modulua:

```
x %ebp+8-n, y %ebp+12-n

1 movl 12(%ebp), %eax 2
imull 8(%ebp) ; EDX:EAX<-x*y[63-32:31-0] ; imul: -ren biderketa
sinatutako zenbaki
osoak 3 movl %eax, ; pila <-x*y[31:0] ; pila
(%esp) 4 movl %edx, 4(%esp) <-x*y[63:32]
```

ÿ Pila: i386 arkitektura ÿ x86-32 ÿ hitza=4bytes. Pilaren hazkundea memoria-helbide baxuagoetara doa.



ŷ EBP: oinarrizko erakusle-erregistroa: pilaren behealdera seinalatzen du

ŷ ESP: pila-erakusleen erregistroa: pilaren goiko alderatzen du

ŷ Pila okupatuta dago behetik goiko helbidera, non goiko helbidea < behetko norabidea.

2. Demagun x eta y zenbaki oso zeinudunak 8 eta 12 posizioetan gordeta daudela.

erregistratu %ebp, eta x/y produktua eta x mod y ere pilaren goiko aldean gorde nahi dituzu, erregistratu %esp (pila-erakuslea) pilaren goiko erakuslea. Garatu gasa muntatzeko kodea i386 arkitektura.

ŷ Soluzioa: asm modulua

```

x ebp+8-n, y ebp+12-n

1 movl 8(%ebp), %edx ; edx<-x
2 movl %edx, %eax ; kopiatu x eax
3 sarl $31, dibidenduaren      ; edx x-ren zeinu-bitarekin betetzen dut, parte denez
                                ; %edx.
                                ; edx:eax <- x
4 idivl 12(%ebp) 5 movl      ; EAX<-Kotientea{x/y}, EDX<-Hondarra{x/y}
%eax, 4(%esp) 6 movl %edx,    ; pila -- zatidura{x/y}
(%esp)                      ; pila <- x%y ; x%y = hondarra{x/y}

```

ŷ Aurreko adierazpen bera baina cltd zeinua luzapena instrukzioa erabiliz

ŷ asm modulua

```

x ebp+8-n, y ebp+12-n

1 movl 8(%ebp), %edx ; edx<-x
2 movl %edx, %eax ; kopiatu x eax
3 cltd ; eax-en eragigaiaren zeinua edx-era hedatzen du
4 idivl 12(%ebp)      ; EAX<-Kotientea{x/y}, EDX<-Hondarra{x/y}

```

```
5 movl %eax, 4(%esp) 6 ; pila <- Koziente{x/y} ; pila <-
movl %edx, (%esp) x%y ; x%y = hondarra{x/y}
```

## 3. Iturburu-modulutik C hizkuntzan:

- ÿ Deduzitu num\_t mota store\_prod funtzioaren prototipoaren argumentutik
- ÿ ASM modulua RTL hizkuntzan interpretatzea

```
void denda_prod(num_t *dest, sinatu gabeko x, num_t y) { *dest = x*y; }
```

dest ebp+8-n, x ebp+12-n, y ebp+16-n

```
1 movl 12(%ebp), %eax 2
movl 20(%ebp), %ecx 3 imull
%eax, %ecx 4 mull
16(%ebp) 5 leial
(%ecx,%edx), %edx 6 movl
8(%ebp), %ecx 7 movl
%eax, (%ecx) 8 movl
%edx, 4(%ecx)
```

## ÿ SOLUZIOA:

- ÿ Dest argumentua ebp+8 helbidean ezartzen da
- ÿ dest num\_t motako objektu baten erakuslea da
- ÿ 6. lerroa: kargatu ecx dest balioarekin
- ÿ 7. lerroa: kargatu eax dest-ek adierazitako memoria helbidean
- ÿ 1. lerroa: kargatu eax zeinurik gabeko motako x aldagaiarekin, orduan num\_t sinatu gabekoa da.

## 10.7.4. Jauziak

- Kalkulatu salto-helbideak makina-kodeean hurrengo mutaia-kode-blokean:

## ÿ iturburu-modulua:

```
1 jle .L2 bada <, joan dest2 2 .L5:
dest1: 3 movl
%edx, %eax 4 sarl %eax

5 subl %eax, %edx 6
leial (%edx,%edx,2), %edx 7 testl
%edx, %edx 8 jg .L5
bada >, joan dest1 9 .L2: dest2: 10
movl %edx, %eax
```

ŷ Birkoka daitekeen objektu-moduluak: memoria-kokapenak erreferentzia-helbide "tonto" arekiko erlatiboak dira (Iekuz alda daitekeen moduluaren zero helbidea)

```

1 8: 7e 0d jle 17 <silly+0x17> Helburua = dest2 2 a:
89 d0 mov %edx,%eax dest1: 3 c:
d1 f8 sar %eax
4 e: 29 c2 azpi %eax,%edx 5
10: 8d 14 52 irakurri (%edx,%edx,2),%edx
6 13: 85 d2 proba %edx,%edx
7 15: 7f f3 jg a <tontoa + 0xa> Helburua = dest1 8
17:89 d0 mov %edx,%eax dest2:

```

ŷ Objektu exekutatu-moduluak: Lokatzaileak objektu-moduluaren memoria-kokapen erlatiboak ebatzi ditu Iekuz alda daitezke memoria absolutu-helbideetan bihurtuz.

```

1 804839c: 7e 0d jle 80483ab <silly+0x17> 2
804839e: 89 d0 mov %edx,%eax
3 80483a0: d1 f8 sar %eax
4 80483a2: 29 c2 azpi %eax,
%edx 5 80483a4: 8d 14 52 irakurri (%edx,
%edx,2),%edx 6 80483a7: 85 d2
proba %edx,%edx 7 80483a9: 80483a9 jg
704 +0xa> 8 80483ab: 89 d0 mov %edx,%eax

```

ŷ SOLUZIOA:

ŷ Etenaldiak kodearen 1. eta 7. lerroetan daude.

ŷ 1. lerroko eten operandoa dest2 etiketatutako 80483ab helbide absolutua da

ŷ 1. lerroa exekutatzenean, PCak edo RIPak 2. lerrora seinalatzen du, hau da, 804839e.

ŷ Saltoa 80483ab - 804839e = 0D kenketa izango da

ŷ 7. lerroko salto-eragilea 804839e helbidea dest1 etiketatua da

ŷ Jauzia 804839e - 80483ab kenketa izango da

ŷ Kenketa negatiboa izango denez, eragigaiak alderantzizatzen ditut eta ondoren, ikurra aldatzen dut emaitza

ŷ 80483ab - 804839e = 0D

ŷ 0D-ren 1 byteko tamainako 2 osagarria F2+1 = F3 da

#### 10.7.5. Bada-Orduan-Bestela

1. Erlazionatu C lengoainan (Ref. Randal194) programa bat eta mihiztatzaileen programa baliokidea behe-mailako iturburu-moduluko iruzkinen bidez:

ŷ C moduluak:

```

(a) Jatorrizko C
kodea 1 int absdiff(int x, int y)
{ 2 if (x < y)
3 return y - x;

```

**4****bestela 5 itzuli**x - y; **6 }**(b) **Goto** baliokidea **1.** bertsioaint **gotodiff(int x, int y) { 2 int**emaitza; **3 if**(x >= y) **4 goto**x\_ge\_y; **5**emaitza = y - x; **6**eginda **joan** ;**7 x\_ge\_y:****8 emaitza = x - y;****9 eginda:****10 emaitza itzuli ;****11 }**

ŷ ASM modulua:

(c) Sortutako x muntaketa-

kodea %ebp+8-n, y %ebp+12

**-n 1** movl 8(%ebp), %edxLortu x **2** movl 12(%ebp), %eaxLortu y **3** cmpl %eax, %edxKonparatu x:y **4** jge .L2 **if** >=**goto** x\_ge\_y **5** subl %edx, %eax Kalkulatuemaitza = yx **6**

jmp .L3 Goto

done **7 .L2:** x\_ge\_y: **8** subl %eax, %edxKalkulatu emaitza = xy **9** movl % edx, %eax Ezarriemaitza **itzulera-** balio gisa **10 .L3:** eginda: osatzeko kodea hasi

## 10.7.6. Do-While Loops

1. Erlazionatu C lengoainan (Ref. Randal199) programa bat eta mihiztatzaileen programa baliokidea behe-mailako iturburu-moduluko iruzkinen bidez:

ŷ C modulua:

**1 int dw\_loop(int x, int y, int n) { 2 do**{ **3 x**+= n; **4**

eta \*= n;

**5 n--;****6 } bitartean ((n > 0) && (y < n)); 7****itzuli x; 8 }**

ŷ ASM modulua:

x %ebp+8-n, y %ebp+12-n, n %ebp+16-n

```

1 movl 8(%ebp), %eax
2 movl 12(%ebp), %ecx
3 movl 16(%ebp), %edx
4 .L2:
5 addl %edx, %eax
6 imull %edx, %ecx
7 subl $1, %edx
8 testl %edx, %edx
9 jle .L5
10 cmpl %edx, %ecx
11 jl .L2
12 .L5:
```

ÿ Idatzi funtziaren goto bertsio bat (C-n) muntaketa-kode programaren funtzionamendua imitatzen duena.

2. Erlazionatu C lengoaian (Ref. Randal201) programa bat eta mihiztatzaileen programa baliokidea behe-mailako iturburu-moduluko iruzkinen bidez:

ÿ C modulua:

```

1 int loop_while(int a, int b) 2 { 3
int
emaitza = 1; 4
bitartean (a < b)
{ 5 emaitza *= (a+b);
6 a++;
7 }
8 emaitza itzuli ;
9 }
```

ÿ Asanblada: Asanblada-kodea sortzean, gcc-k eraldaketa interesgarri bat egiten du eta, hain zuen ere, programaren aldagai berri bat sartzen du. %edx erregistroa 6. lerroan hasieratzen da eta 11. lerroko begiztaren barruan eguneratzenten da.  
Deskribatu nola erlazionatzen den C kodean aldagaiekin. Sortu funtzi honetarako erregistro-erabileraren taula.

a %ebp+8-n, b %ebp+12-n

```

1 movl 8(%ebp), %ecx
2 movl 12(%ebp), %ebx
3 movl $1, %eax
4 cmpl %ebx, %ecx
5 jge .L11
6 leial (%ebx,%ecx), %edx
7 movl $1, %eax
8 .L12:
9 imull %edx, %eax
10 gehi $1, %ecx
11 gehi $1, %edx
```

```
12 cmpl %ecx, %ebx 13
jg .L12 14 .L11:
```

3. Erlazionatu C lengoainan (Ref. Randal204) programa bat eta mihiztatzaileen programa baliokidea behe-mailako iturburu-moduluko iruzkinen bidez:

ÿ C modulua:

```
1 int fact_for_goto(int n) 2 { 3 int i =
2;
4 int emaitza =
1; 5 baldin (!i <= n)) 6
eginda goto; 7 begizta:
8 emaitza *= i; 9
i++; 10 if
(i <= n) 11 joan
begizta;
12 eginda:

13 itzuli emaitza; 14}
```

ÿ asm modulua

Argudioa: n %ebp+8-n  
 Erregistroak: n %ecx-en, i %edx-en, emaitza %eax-en

```
1 movl 8(%ebp), %ecx Lortu n 2 movl
$2, %edx Ezarri i 2 gisa (init) 3 movl $1, %eax
Ezarri emaitza 1 4 cmpl $1, %ecx Konparatu
n:1 (ltest) 5 jle .L14 If <=, goto done 6 .L17: begizta:
7 imull %edx, %eax Kalkulatu emaitza
*= i (gorputza) 8
addl $1, %edx Increment i (eguneratzea) 9 cmpl %edx, %ecx
Konparatu n :i (proba) 10 jge .L17 >= bada, joan 11.
begiztara .L14: eginda:
```

#### 10.7.7. Azpirrutina deia

##### 1. Aztertu sumMtoN.s azpierrutina

```
##Suroutine: sumMtoN
##Deskribapena: 1. gehigarritik 2. gehigarrira segidan zenbaki osoen batura kalkulatzen du

## Sarrerako argumentuak: 1. gehitzeko eta 2. gehitzeko argumentuak pasatzen ditu
```

errutina nagusia pilaren bidez: ## 1. azken argumentua pilatzen da eta azkenik 1. argumentua pilatzen da.

## Irteera argumentua: batuketaren emaitza da eta errutina nagusira pasatzen da EAX erregistroaren bidez.

## Aldagai lokalak: aldagai lokal bat implementatzen da pilan baina ez da erabiltzen

```
.type sumMtoN, @function # deklaratu sumMtoN etiketa

batuketa:

## Hitzurrea: Sortu pila-marko berria pushl %ebp
#gorde zaharra marko-erakuslea movl %esp, %ebp
#eguneratu marko-erakuslea
## Erreserbatu hitz bat pilean aldagai lokal gisa
## Aldagai lokala kanpoko memorian: sum subl $4, %esp

## Capturing arguments movl
8(%ebp), %ebx #1. argumentua %ebx movl 12(%ebp) kopiatura, %ecx #2. argumentua %ecx-ra kopiatura

## 1oarg balioaren eta 2oarg balioaren arteko sekuentzia gehitzen du ## 1o arg < 2oarg ## EDX aldagai lokal gisa erabiltzen dut aldagai lokalaren kanpoko erreserbaren ordez: abiadura optimizatzen du

## Tokiko aldagai batuketa movl $0, %edx hasieratzen dut

## Mugimendu kopurua %ecx,
%eax azpi %ebx,
%eax

begizta:
gehitu %ebx, %edx
inc %ebx sub
$1, %eax jns loop

## Gorde batura emaitza movl %edx, %eax itzulera-balio gisa

## Epilogoa: berreskuratu marko zaharra movl %ebp, %esp #restore the stack pointer popl %ebp #restore the frame pointer

## Itzuli errutina nagusira ret .end
```

2. Programatu errutina nagusi bat sumMtoN azpierrutinari dei batekin. Irten nagusitik irten sistema-deiarekin. Exekutatu programa GDB araztearekin eta egiaztatu "hurrengo" komandoa sumMtoN azpierrutinan **sartzen dela**. Azpierrutina barruan egonda, interpretatu jakinarazten duen "backtrace" **bt** komandoa

pilako fotograma kopuruari eta egoerari buruz.

3. Programatu errutina nagusi bat sumMtoN azpierrutinari dei batekin. Main sartu bezain laster, editatu hitzaurre nagusia (marko berri baten sorrera) eta amaitu nagusia epilogarekin (markoaren amaiera) eta "ret" instrukzioarekin. Azpirrutina barruan egonda, interpretatu pilako fotogramen kopurua eta egoeraren berri ematen duen "backtrace" bt komandoa.

```
## Hitzaurrea: Sortu pila-marko berria pushl %ebp
#gorde zaharra marko-erakuslea movl %esp, %ebp
#eguneraatu marko-erakuslea
```

```
## Epilogoa: berreskuratu marko zaharra movl
%ebp, %esp #restore the stack pointer popl %ebp
#restore the frame pointer
```

## 10.8. C Programazio Lengoaia

### 10.8.1. Erakusleak

1. Editatu eta exekutatu hurrengo programa C hizkuntzan emaitza interpretatuz zerrenda hori elementu-zerrenda[0] seinalatzen duen "erakusle-aldagai" dela jakinda.

```
#include <stdio.h>

void main(void) {

    int zerrenda[]={1,2,3,4,5};

    printf ("\n *****ARRAY LIST*****"); printf ("\n
*****zerrenda ERAKUSKETA ALDAGAIA da*****"); printf ("\n
**aldagaien zerrendak zerrendaren helbidea dauka[0]*****\n\n"); printf("\n zerrenda[0] = %d
zerrendako 1. elementua da \n", zerrenda[0]); printf("\n zerrenda = %p 1.
elementuaren helbidea da \n", zerrenda); printf("\n &list[0] = %p 1. elementuaren
helbidea da \n", &list[0]); printf("\n *zerrenda = %d zerrenda berdina [0] \n", *zerrenda);
printf("\n *&list = %p zerrenda berdina \n", *&list); printf("\n
*&zerrenda = %d zerrenda berdina [0] \n", **&zerrenda);

    printf ("\n\n *****POINTERO ARITMETIKA*****"); printf("\n
*(zerrenda+1) = %d zerrenda [1] da \n", *(zerrenda+1)); printf("\n *(zerrenda+4)
= %d zerrenda [4] da \n", *(zerrenda+4));

    printf ("\n\n *****CASTING*****"); printf("\n
(int *)list = %p \n", (int *)list); printf("\n *(int *)list = %d \n",
*(int *)list); printf("\n *(int)list+1 = %d \n", *(int *)list+1); printf("\n
*(int)list+4 = %d \n", *(int *)list+4);
```

```
printf("\n *(((labur *)zerrenda+1) = %d . Orain 2 elementu dituen zerrenda deklaratzen dut
byte \n", *(((labur *)zerrenda+1));
}
```

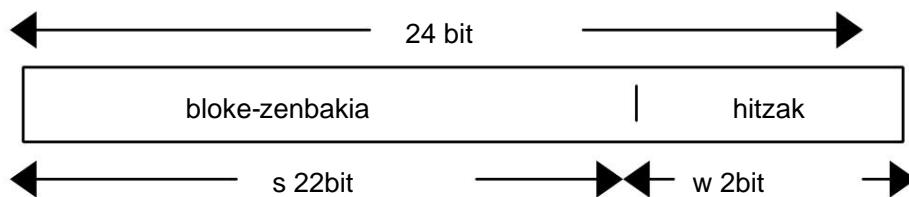
2. GDB araztearekin arazketan muntaketa-lengoaian idatzitako programa bat non zerrenda zuzentaraua: .int 1,2,3,4,5, ondorioztatu ondorengo adierazpenak zerrenda etiketa muntatuta badago 0x55bf0000 helbidea gisa eta, beraz, ez da erakusle-aldaigai bat, baina erakusle bera da, da Hau da, zerrenda-matrizeko lehen elementuaren helbidea. Adierazi helbide-maparen arteko erlazioa memoria eta zerrenda-marizeko elementuen posizioen mapa.

```
&zerrenda: 0 posizioan dagoen elementuaren "array aldagai" zerrendaren helbidea
: 0x55bf0000
&list+1           : 0x55bf0004 -> 1. posizioan dagoen elementuaren helbidea.
(void *)&list+1 (int   : 0x55bf0001
*&list+1 zerrenda  : 0x55bf0004
                  : 1 -> berdinaren zerrenda[0]
(int)list (int     : 1
*&list (int       : 0x55bf0000
[5])list *(int    : {1,2,3,4,5}
*&list+1          : 2
```

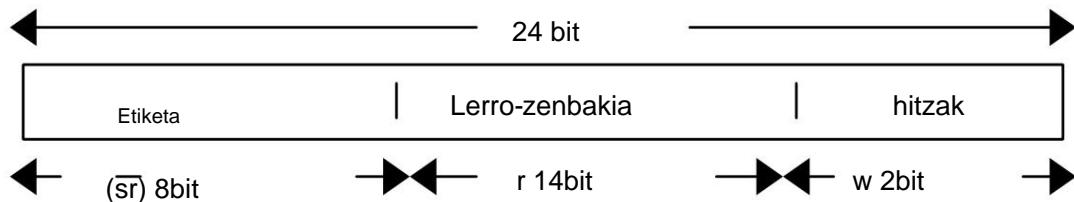
#### 10.9. 4. kapitula: Cache memoria

1. Liburua William Stalling 4.2a Adibidea 125. orrialdea. Kalkulatu Cache kontrolagailu baterako Association of Zuzeneko mapaketa, zein cache-lerroa (etiketa/lerroa/hitza) 0x02ABCD helbidearekin lotuta dagoen memoria nagusitik.

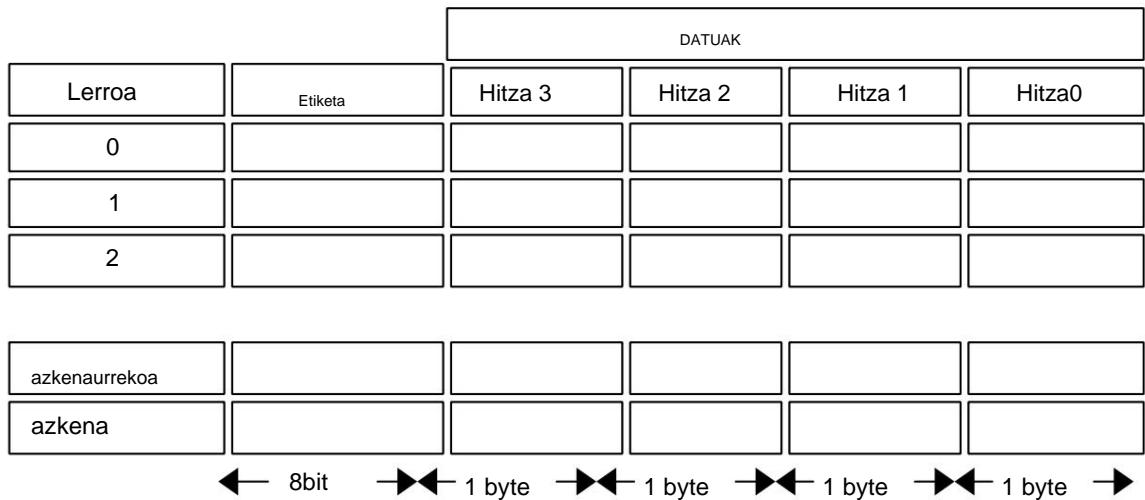
ÿ Memoria nagusiaren helbide-formatua (MPrincipal/Word Block)



ÿ Memoria nagusiaren helbide formatua (Tag/Cache Line/Word)



ÿ Cache Memoria



ÿ 0x02ABCD = 0000-0010-1010-1011-1100-1101

ÿ Blokea/Hitza ÿ 0000001010101011110011/01 ÿ Blokea=0000001010101011110011 Hitza=01 ÿ blokea=0x0AAF3 hitza=01

ÿ Etiketa/Lerroa/Hitza ÿ 00000010/10101011110011/01 ÿ Etiketa=00000010 Lerroa=10101011110011 Hitza=01 ÿ etiketa=02 lerroa=0x2AF3 hitza=01

ÿ Etiketa 0x02 lerroa 0x2AF3 1. hitza

ÿ PUZaren arazoak 0x02ABCD helbidean, Cache kontrolatzalea 0x2AF3 lerrora joango da eta Lerro horren Etiketa 0x02 bada, lerro horren datuen edukia (4 byte) bat dela esan nahi du. 0x0AAF3 blokeko datuen edukiaren kopia (4 byte) eta, beraz, horren 01 hitza memoria bloke nagusia cache-lerro horretan dago ÿ sakatu. Lerro horretan badezpada etiketa ez da 02, lerro horretako datuak ez dira blokearen eta cache kontrolatzalearen kopia bat memoria-kontrolatzale nagusiari cacherako blokearen kopia bat eskatu beharko diozu eta CPUrako beste kopia bat.

2. Errepikatu aurreko ariketa biderketa eta zatiketa bitarrak erabiliz, honako hauei erantzunez atalak:

to. osatu taula Cache Memoriako lerroak MPrincipal-eko helbideekin lotuz (lehenik bloke bakotzeko hitza)

Lerroa	MP helbidea	MP helbidea	MP helbidea	MP helbidea
0	0x000000	0x010000		
1	0x000004			
$2^{14} - 1$				

ÿ lerroa - MP helbidea erlazioa (etiketa izan ezik) x4 biderketa bat da, lerro eta bloke bakotza 4 byte dituzte.

ÿ azken lerrorako:  $2^{14} - 1 = 0x4000 - 1 = 0x3FFF$  ÿ  $0x3FFF \times 4 = 14$  eta ondoren 2\_zeroak(x4)= 0xFFFFC

ÿ Helbidea 0x010000 ÿ 4 zifra baxuenak (16 bit), hau da, guziak etiketa izan ezik, Cache helbide-busaren baliokideak dira, baina ez lerro helbideak, byte helbideak baizik, alegia cachearen edukiera bytetan.

b. kalkulatu MP-aren 0x0000B helbide lineala Block/Word helbidea dagokio:

ÿ bloke/hitz helbidea = MP helbidea 4z zatituta ÿ zatidura bloke-zenbakia eta gainerako hitzaren zenbakia

ÿ  $0x0000B \div 4$  ÿ  $11=2\times4+3$  ÿ 2. zenbakia eta hitza 3. zenbakia

c. kalkulatu 0x02ABCD MP helbiderako zein Bloke/Hitz helbideri dagokion:

ÿ  $0x02ABCD \div 4$  ÿ Koma bi posizio ezkerrera mugitzen dut ( $1/4=2-2$ ) ÿ zati osoa Bloke zk. 0xAF3 eta zatiki zatia Hitz N°01

d. kalkulatu 0xAF3 bloke zenbakia zein lerro dagokion:

ÿ  $i=j \bmod m$ , non  $m=214$  ÿ zuzena 0xAF3 214z zatiztearen hondarra den ÿ zatikiaren zatia da. mugitu koma 14 leku ezkerrera ÿ 0xAF3

eta. kalkulatu MP helbiderako 0x02ABCD zein byte eta cache-lerroari dagokion:

ÿ byte ÿ 0xABCD

ÿ lerroa ÿ  $0xABCD \div 4$  ÿ koma bi bit ezkerrera mugitzen dut eta osoko zatia hartzen dut ÿ 0xAF3

F. kalkulatu 0x02ABCD MP helbiderako zein cache-hiri dagokion

ÿ hitza ÿ 0xABCD 4 moduloan ÿ koma bi bit ezkerrera mugitzen dut eta zatiki zatia hartzen dut ÿ 01

3. William Stalling Adibidea 4.2a Pg125: Sistemak 64KB-ko Cache memoria eta 16MB-ko Memoria Nagusia ditu byte bateko hitz-tamainarekin eta lau hitz-bloke-tamainarekin. Mapa zuzeneko korrespondentzia-funtzioa duen cache-kontrolatzalea baterako, bilatu memoria-bloke nagusiak korrespondentzia cache-memoriaren 0x0CE7 zenbaki-lerroari zuzentzen dio.

ÿ Garapena:

ÿ Memoria nagusia: 16MB, 1word=1byte, 1bloke=4 byte.

ÿ  $16MB \div 2^{24}$  ÿ 24 biteko helbide-busa

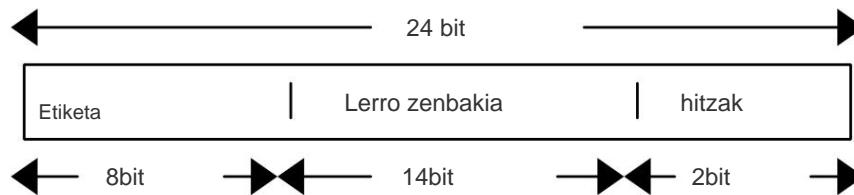
ÿ Cache memoria: 64KB, 4 byte lerroa, 16K lerroa.

ÿ  $16K \div 2^{14}$  ÿ 14 biteko lerro-eremua

ÿ Mapa zuzeneko korrespondentzia funtzioa: 0x0CE7 cache-lerroa

ÿ  $i=j \bmod m$  non i lerro-zenbakia den, j bloke-zenbakia eta m lerro-kopurua cachea.

ÿ 24 biteko helbidea 3 eremutan deskonposatuta dago: etiketa-llerro-hitza



ÿ Etiketa bakoitzak 16K bloke taldekatzen ditu ÿ  $(16K \text{ bloke}/\text{etiketa}) \times (4\text{byte}/\text{bloke}) \times (\text{etiketen kopurua} N) = \text{CapacityMemPrincipal} = 16MB \div 2^{14} \times 2^{12} \times N = 224\text{byte}$  ÿ  $N=28$  Etiketa ÿ Etiketa talde bakoitzaren baliokidea denez Cachearen edukiera ÿ MPrincipal 256 cacheren baliokidea da.

ÿ 0CE7 lineako helbidea: 14 biteko lineako helbidea: 00-1100-1110-0111. Helbideak bilatzen ditugu lerro horrekin lotutako memoria.

ÿ MemPrincipal Helbidea = 0 etiketa, 0CE7 lerroa, 0 hitza ÿ 0000-0000-00-1100-1110-0111-00 = 0000-0000-0011-0011-1001-1100 ÿ 00339C

ÿ MemPrincipal Helbidea = Etiketa 1, 0CE7 lerroa, 0 hitza ÿ aldatu lehen eremua 01 ÿ  
01339C

ÿ MemPrincipal Helbidea = Tag 255, Line 0CE7, Word 0 ÿ aldatu lehenengo eremua FF-ra  
ÿ FF339C

Ý Bloke baten memoria helbideak Memoria Nagusiko blokearen lehen hitzaren helbidea dira.

4. Multzo-asoziaziozko cacheak 64 lerro edo zirrikitu ditu, lau lerroko multzotan banatuta. Memoria nagusiak 128 hitzeko 4K bloke ditu. Erakutsi memoria nagusien helbideen formatura. Kalkulatu 0x7FAFA memoria helbidea kode hamaseimalean erakutsi cache formatua. Nola jakin cache kontrolatzalea 0x7FAFA sartuta badago cache memoria.

ÿ Garapena:

Cachea: 128 hitzeko 64 lerro bakoitzak 4 lerroko multzotan bilduta

ÿ 128 hitz ÿ 7 bit lerro barruan hitza zuzentzeko

ŷ 16 multzo ŷ 2<sup>4</sup> ŷ 4 bit cachearen barruan dauden multzoak zuzentzeko

ÿ Memoria nagusia: 27 hitzeko 4Kbloke

ÿ 12 bit bloke bat zuzentzeko

ÿ 2<sup>19</sup> hitzak ÿ 512Khz ÿ 19 bit hitz bat zuzentzeko ÿ busaren zabalera  
helbide

ŷ multzo elkartua ŷ  $i = j \bmod v$  non  $v$  multzo kopurua den,  $j$  blokea eta ei multzoa

ŷ Etiketa Ÿ kodea multzo berera doazen blokeak bereizteko. Etiketa bitak=bit guztira - Ezarri bitak - Word bits=19-4-7=8 bit.

ÿ Etiketa/Multzoa/Hitza ÿ 19 helbide-bit 8/4/7 biteko 3 eremutan banatuta

y Eguzkia:

ÿ Etiketa/Multzoa/Hitza: 8/4/7

ÿ 0x7FAFA ÿ Etiketa 0xFF, Set 0x6, Word 0x7A. Etiketa 0xFF 0x6 multzoko lau lerroetako batean badago, orduan "hit" bat gertatu da.

5. 111111, 666666, BBBBBB memoria nagusi hamaseitar helbideetarako, erakutsi informazio hau, formatu hamaseimalean:

1. Zuzeneko mapatutako cache baterako etiketa, lerroa eta hitza balioak, 4.10 irudiko formatua erabiliz
  2. Etiketa eta Word balioak cache asoziatibo baterako, 4.12 irudiko formatua erabiliz
  3. Etiketa, multzo eta Word balioak bi norabideko multzo-asoziazio cache baterako, 4.15 irudiko formatua erabiliz

ÿ Garapena:

ÿ a) Zuzeneko mapatutako Etiketa/Lerroa/Hitza ÿ 24 helbide-bit 3 eremuetan banatuta  
8/14/2 bit

ÿ 111111 = 0001-0001-0001-0001-0001-0001 = 00010001-00010001000100-01=0001-0001-00-0100-0100-0100-01=11-0444-1 ÿ 0 ez dago ezkerraldean hexadegiroan idatzia

ÿ b) Cache asoziatibo osoa Etiketa/Hitza ÿ 24 helbide-bit 2 eremuetan banatuta  
22/2 bit

ŷ 111111 = 0001-0001-0001-0001-0001 = 0001000100010001000100-01=00-0100-  
0100-0100-0100-0100-01=044444-1. ŷ 0 ez dago ezkerraldean hexadegiroan idatzita.

ŷ c) Ezarri cache asoziatiboa Tag/Set/Word ŷ 24 helbide-bit 3 eremutan banatuta  
9/12/2012 kurra

Ü Equizka:

## 16. taula Helbideak

Helbidea	111111	666666	BBBBBB
to.	11/444/1	66/1999/2	BB/2EEE/3
Etiketa/lerroa/hitza			
b. Etiketa/Hitza	44444/1	199999/2	2EEEEEE/3
c. Etiketa/Multzoa/Hitza	22/444/1	CC/1999/2	177/EEE/3

6. Demagun 32 biteko mikroprozesadore bat, txip-an 16 KByteko lau bideko multzo-asoziazioko cachea duena. Demagun cacheak 32 biteko lau hitzeko lerro-tamaina duela.

1. Marraztu cache honen bloke-diagrama, bere antolaketa eta helbide-eremu desberdinak nola dauden erakutsiz cache-ren arrakasta/ezintasuna zehazteko erabiltzen da.

2. Cachean non dagoen ABCDE8F8 memoria-kokapeneko hitza mapatuta

ŷ Garapena:

ŷ Memoria nagusia

ŷ Kanpoko autobusari buruz ez du ezer esaten, gehienez 32 bit ŷ 2 32 byte ŷ 4GB

ŷ Txiparen cachea: bus lokala: 32 biteko datu-busa eta helbide-busa: multzo bakoitzeko 4 lerroko multzo elkartua.

ŷ Lerro bakoitzeko 4 byteko 4 hitzek guztira 16 byte osatzen dute lerro bakoitzeko (4 bit hitz eremuan). 4 biteko kodeak hitz bakoitzaren lehen bytea zuzentzen du (0x0 hitza 0, 0x4 hitza 1, 0x8 hitza 2, 0xC hitza 3)

ŷ Multzo kopurua guztizko edukiera/byte da multzo bakoitzeko = 16KB / **(4 lerro/multzo) (4 hitz/llerro) (4 byte/hitz)** = 16KB/64B = 28=256 multzo ŷ 8bit

ŷ bloke kopurua cachean dago edukiera/llerro bakoitzeko\_bytes= 16KB/(4hitz/llerro)\*(4byte/hitz)=1Kbloke

ŷ 1kblokeak 4 lerroko multzotan lotzen dira.

ŷ helbide-busa=etiketa bits+set bits+hitza bits ŷ 32=etiketa bits+8+4 ŷ tag\_bits=32-8-4=20 bit.

Etiketa eremuak multzo bereko blokeak bereizten ditu.

ŷ Zein bloke doaz multzo berera? i = j mod v, non i j blokea doan multzo-zenbakia den, v multzo-kopurua den.

Hau da,  $2^{20}$  bloke multzo berdinarekin lotuta daude, beraz, 4 lerro partekatu behar dituzte ŷ 4 20rako.

ŷ Etiketa/Multzoa/Hitza ŷ 32 helbide-bit 20/8/4 biteko 3 eremutan banatuta

ŷ a) Etiketa/multzoa/hitz deskonposizioaren ondoren, helbideratutako multzoa hautatzen da eta 4 lerroetako etiketak helbide-etiketa absolutuarekin alderatzen dira. 4 konparagailu daude, bat errei bakoitzeko. Memoria nagusia banatzen dugun multzo bakoitzeko lehen lerroa Konparatzalea\_1era joango da. 232 byte multzo bakoitzeko 16 hitzeko multzotan multzokatzen ditut ŷ memoria nagusia 28 multzotan banatzen da

ŷ b) 20/8/4 helbidearen deskonposizioa ABCD8F8ŷ ABCD/8F/8 ŷ 8F 143 ezarri da eta 8 bytea 2 hitz-zenbakia da.

ŷ Sol: a... Deskonposizioa: Etiketa/Ezarpena/Offset. 4 konparatzale: 1 Multzoaren modu bakoitzeko. b...143. multzoa, edozein lerro, hitz bikoitza 2 zenbakia.

7. Intel 80486-k txip-eko cache bateratua du. 8 KByte ditu eta lau noranzko multzo-asoziazo-antolaketa eta 32 biteko lau hitzen bloke-luzera ditu. Cachea 128 multzotan antolatuta dago. "Lerro baliozko bit" bakarra eta hiru bit, B0, B1 eta B2 ("LRU" bitak), lerro bakoitzeko daude. Cache hutsean, 80486-k memoria nagusiko 16 byteko lerro bat irakurtzen du bus-memoriaren irakurketa-leherketa batean.

1. Marraztu cachearen diagrama simplifikatu bat

2. helbidearen eremu desberdinak nola interpretatzen diren erakutsi.

ŷ Garapena:

ÿ Intel 80486 (38,47,130 orrialdeak) 32 biteko memoria-busa du ÿ 32 biteko helbide-busa, byte 1 zuzentzen duena.

ÿ Cachea: 8KB, 4 bide-multzo elkartua, lerro bakoitzean 4 byteko 4 hitz (16 byte 4 biteko), eta 128 multzo (7 bit)

ÿ Lerro bakoitzeko 4 byteko 4 hitzek guztira 16 byte osatzen dute lerro bakoitzeko (4 bit hitz eremuan). 4 biteko kodeak hitz bakoitzaren lehen bytea zuzentzen du (0x0 hitza 0, 0x4 hitza 1, 0x8 hitza 2, 0xC hitza 3)

ÿ 32 biten deskonposizioa: Tag/Set/Offset ÿ 7/21/4

ÿ 32 bitez gain, gehitu behar da:

ÿ 3 ERABILI bit lau lerroetatik zein den gutxien erabili den adierazteko, ahalik eta 8 baliorik txikiena duena: 000-001-010-011-101-110-111

ÿ Lerroa gainidatzi aurretik MP-n egunerautu behar dela 1 balioarekin adierazten duen baliozkotze-bit 1 ÿ idazketa osteko teknika.

ÿ Lerroa: Baliozko/LRU/Etiketa/Ezarpena/Desplazamendua ÿ 1/3/21/7/4

ÿ Soluzioa: a.. Ezarri eskema elkartua b.. Baliozko/LRU/Etiketa/Ezarri/Desplazamendua ÿ 1/3/21/7/4

8. Kontuan izan hurrengo kodea:

```
for (i=0; i=20; i++) for (j=0;
j=10; j++) a[i] = a[i] * j ;
```

1. Eman tokiko espazioaren adibide bat kodean.

2. Eman tokiko denborazko adibide bat kodean.

ÿ Garapena

ÿ barruko begiztan instrukzio bera errepikatzen da beti, beti berdin sartzen zara instrukzioa dagoen helbidea ÿ kokapen espaziala

ÿ barneko begiztan beti errepikatzen da instrukzio bera, etorkizuna orainaldia da ÿ lokaltasuna aldi baterako

ÿ j=0 duen barruko begiztan a[0] eragiketara sartzen zara eta hurrengo iterazioan prozesua errepikatzen da eragiketa bera a[0] ÿ leku espaziala eta denborazkoa.

9. Demagun 16 byteko 4 lerroko cache bat. Memoria nagusia 16 byteko blokeetan banatzen da. Hau da, 0 blokeak 0tik 15era bitarteko helbideak dituzten byteak ditu, eta abar. Orain kontuan hartu memoria hurrengo helbide-sekuentzian sartzen den programa bat:

Hamaika: 63tik 70era

Begiztatu hamar aldiz: 15etik 32ra; 80tik 95era

1. Demagun cachea mapa zuzen gisa antolatuta dagoela. 0, 4 eta abar memoria blokeak 1. lerroari esleitzen zaizkio; 1, 5 eta abar blokeak 2. lerrora; eta abar. Kalkulatu hit ratioa.

2. Demagun cachea bi norabideko multzo elkartu gisa antolatuta dagoela, bakoitza bi lerroko bi multzorekin. Zenbaki bikoitiak 0 multzoari esleitzen zaizkio eta bakoitiak 1. multzoari. Kalkulatu bi norabideko multzo-asoziazioko cachearen arrakasta-ratioa gutxien erabili den ordezkapen-eskema erabiliz.

10. Demagun maila bakarreko cache bat 2,5 ns-ko sarbide-denbora duena, 64 byte-ko lerroaren tamaina eta  $H = 0,95$ -ko arrakasta-erlazioa. Memoria nagusiak bloke-transferentzia-gaitasuna erabiltzen du, lehen hitza (4 byte) 50 ns-eko sarbide-denbora duena eta ondoren hitz bakoitzeko 5 ns-eko sarbide-denbora duena.

1. Zein da sarbide-denbora cache hutsa dagoenean? Demagun cachea lerroak egin arte itxaroten duela memoria nagusitik eskuratu eta gero berriro exekutatzen da hit baterako.
  2. Demagun lerroaren tamaina 128 byteraino handitzeak H 0,97ra igotzen duela. Horrek memoriarako batez besteko sarbide-denbora murrizten al du?
11. Motorola 68020 mikroprozesadorean, cache-rako sarbideak bi erloju-ziklo behar ditu. Memoria nagusitik datuak busaren bidez prozesadorera sartzeko hiru erloju-ziklo behar dira itxaron egoera txertatzerik ez dagoenean; datuak prozesadorera bidaltzen dira cachera entregatzen denarekin batera.
1. Kalkulatu memoria-ziklo baten luzera eraginkorra 0,9ko arrakasta-erlazioa eta 16,67-ko erloju-tasa emanda. MHz.
  2. Errepikatu kalkuluak memoria bakoitzeko ziklo bateko bi itxaron-egoera txertatuz zikloa. Zer ondorio atera dezakezu emaitzetatik?
12. Cache-maila bi dituen sistema baterako, definitu Tc1 lehen mailako cache-rako sarbide-denbora; Tc2 bigarren mailako cacherako sarbide-denbora; Tm memoria sarbide-denbora; H1 lehen mailako cache-ren arrakasta-erlazioa; H2 lehen/bigarren mailako cache-erlazioa konbinatuta. Eman Ta ekuazio bat irakurtzeko eragiketa baterako.

## 10.10. 5. kapitula: Memoria sinkrono dinamikoa (SDRAM)

1. Intel ordenagailu baten arkitekturak 100MHz-ko erloju-maiztasuna duen sistema-busa du, datu-busaren zabalera 64 bit-ekoia da eta plakaren helbide-busaren zabalera 48 bit-koa.
  1. Kalkulatu autobusaren banda-zabalera transferentzia/s eta byte/s-tan
  2. Kalkulatu memoria-ahalmena
  3. Kalkulatu memoria-zikloa kontuan hartuta DRAM memoriaren latentzia 10ns dela.
 

ÿ Garapena

Yo.  $100 \times 106 \text{ ziklo/seg} \times 1 \text{ Transferentzia/ziklo} = 100 \text{ MT/s}$

ii. helbide-busa ÿ 2 petabyte       $^{48} \text{ Hitzak} = 28 \times 240 = 256 \text{ BIHitzak} = 248 \times 23 \text{ Byte} = 2 \times 250 = 2$

iii. memoria-ziklo ideal (bus multiplexatu gabe, aurrekargarik gabe, etab) = 1 Transfer = memoria\_latentzia + transfer\_bus\_latency =  $10\text{ns} + 1/(108) = 10\text{ns} + 10\text{ns} = 20\text{ ns}$
2. Demagun RAM dinamiko bat, ms bakoitzeko 64 aldiz freskatze-ziklo bat eman behar zaiona. Freskatzeko eragiketa bakoitzak 150 ns behar ditu; memoria-ziklo batek 250 ns behar ditu Memoriaren funtzionamendu-denboraren zenbat ehuneko eman behar da freskatzeko?
 

ÿ Garapena

ÿ 1 ms-tan 150 ns-ko 64 freskatze ÿ 9600 ns freskatuz

ÿ  $9600\text{ns}/1\text{ms} = 0,0096 = \%1$

ÿ Eguzkia:

% hamaika
3. 5.16 irudiak DRAM irakurketa-eragiketa baten denbora-diagrama sinplifikatu bat erakusten du autobus baten gainean. Sarbide-denbora t1-tik t2-ra irauten dela jotzen da. Ondoren, kargatzeko denbora bat dago, t2-tik t3-ra irauten duena, eta horretan DRAM txipak kargatu beharko dira prozesadoreak berriro sartu aurretik.
  1. Demagun sarbide-denbora 60 ns dela eta kargatzeko denbora 40 ns. Zein da memoria-ziklo-denbora? Zein da DRAM honek jasan dezakeen datu-tasa maximoa, 1-bit irteera suposatuz?
  2. Txip hauek erabiliz 32 biteko zabalerako memoria sistema eraikitzeak zer datu transferentzia-tasa ematen du?
 

ÿ Garapena:

ÿ t1ÿt2 : helbidea

ŷ t2yt3

ŷ datuetarako sarbidea

ŷ helbide-busa birkargatzen du 0 eta 1 artean

ŷ 60 ns latenzia eta 40 ns aurrekarga = 100 ns memoria zikloa 2 irakurketa jarraian

ŷ Aurrekargan zehar, leherketa egingo litzateke, hau da, handiagoa, txikiagoa edo berdina izan daiteke. aurrekarga

ŷ 100 ns-ko autobus-zikloa  $1/100\text{ns} = 10\text{MHz}$  da. Bus ziklo bakoitzeko bit bat transferitzen badugu= 10 Mbps

ŷ Paraleloan 32 lerro erabiltzen baditugu = 32 bit/transferentzia  $\times 10\text{MT/s} = 320\text{Mbps} = 40 \text{ MB/s}$

ŷ Eguzkia:

Yo. tzikloa = 100ns. BW=10Mbps

ii. 40 MB/s

4. 5.6 irudiak 1 MByte gorde dezakeen txip-modulu bat nola eraiki adierazten du 256 Kbyte-ko lau txip talde batean oinarrituta. Demagun txip-modulu hau 1 Mbyte-ko txip bakar gisa paketatuta dagoela, non hitzaren tamaina 1 byte den. Eman goi-mailako txip-eskema bat nola eraiki 8 Mbyte-ko ordenagailuko memoria bat 1-Mbyte-ko zortzi txip erabiliz. Ziurtatu zure diagraman helbide-lerroak eta helbide-lerroak zertarako erabiltzen diren erakusten duzula.

ŷ Garapena:

ŷ 256Kbit 4 txipekin 1Mb txip modulua sortzen dut

ŷ 1Mb-ko 8 txipekin 8Mb-ko modulu bat sortzen dut 1Mb-ko txip bakoitzari aukeratutako txip desberdinak ekarriz.

8tik 1 txip hautatzeko 3 helbide-bit behar ditut, adibidez, 3 posizio altuena. 1Mb-ko txip bati heltzeko 20 biteko helbide-busa behar dut. Guztira  $20+3=23$  biteko autobusa behar dut.

ŷ Eguzkia:

Yo. 1M edukiera duten 8 x1 txip non txip-aukeratutako sarrera bakoitza 3 biteko deskodetzaile baten irteera den norabide-lerroak

5. Intel 8086-n oinarritutako sistema tipiko batean, sistema-busaren bidez DRAM memoriara konektatuta, irakurketa eragiketa baterako, RAS Helbidea Gaitu seinalearen azken ertzean aktibatzen da (3.19. Irudia). Hala ere, hedapena eta beste atzerapen batzuk direla eta, RAS ez da aktibo egongo Helbidea Gaitu ondoren 50 ns arte. Demagun azken hori T1 egoeraren bigarren erdiaren erdialdean gertatzen dela (3.19 irudian baino zertxobait lehenago). Datuak prozesadoreak irakurtzen ditu T3 amaieran. Prozesadoreari garaiz aurkezteko, ordea, datuak 60 ns lehenago eman behar dira memoriaz. Tarte honek datu-bideetan zehar (memoriatik prozesadorera) eta prozesadorearen datuen denbora-eskakizunen hedapen-atzerapenak hartzten ditu. Demagun 10 MHz-eko erloju-abiadura.

Yo. Zein azkar (sarbide-denbora) izan beharko lukete DRAMek itxaron-egoerarik txertatu behar ez bada?

- ii. Zenbat itxaron-egoera txertatu behar ditugu memoria irakurtzeko eragiketa bakoitzean sarbide-denbora bada DRAMak 150 ns dira?

ŷ Garapena:

ŷ Irakurketa zikloa

ŷ Kargatu helbidea - Helbidea gaituta (EA) - Helbide komandoa - Sarbide datuak

ŷ Azken ertza = erortze ertza = ertz negatiboa

ŷ AE huts egitea = T1 zikloaren bigarren erdian. Berehalako 75 ns

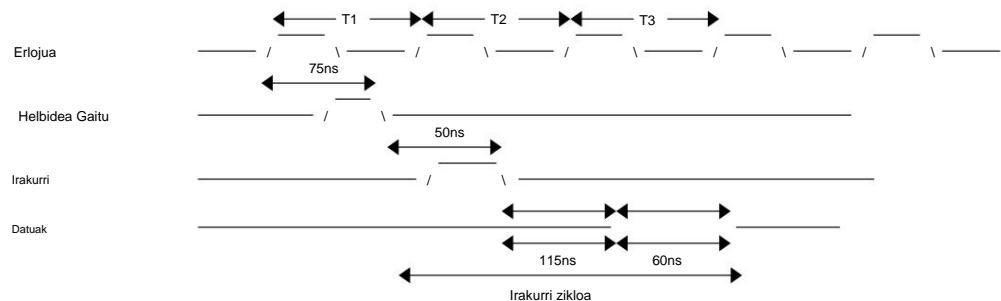
ŷ RAS = Irakurri komandoa: 50ns atzerapena AE hutsegitearekiko. Instant  $75+50=125\text{ns}$

ŷ Datuen aurkezpena autobusean karga baino 60 ns egin behar da

CPUko datuen T3 zikloaren amaiera (300ns), hau da,  $300\text{ns}-60\text{ns}=240\text{ns}$

ŷ Sistema-busaren erlojua = 10 MHz = 100 ns.

- A. Sarbide-denbora (irakurketa-ordenatik datuak memorian kargatu arte) gabe  
 wait states = Ibilbidearen atzerapenek ezarritako gutxieneko sarbide-denbora  
 datuak (CPU eta busa): 240ns - 125ns = 115ns



- B. DRAM memoriak 150 ns-ko sarbide-denbora badu, bus-ziklo bat baino handiagoa,  
 Irakurketa-ordenatik PUZak bi erloju-ziklo itxaron behar ditu, bata  
 irakurri eskaera-zikloa eta beste ziklo gehigarri bat edo WAIT ziklo bat. Beraz, 200ns da  
 nahikoa 150ns sarbide-denbora gainditzeke. Itxaron zikloa bada  
 115ns ondoren hasten da, 215ns dugu, 150ns gainditzen duena.

ŷ Eguzkia:

iii. 115ns

iv. 1

## 10.11. 7. kapitula: Sarrera/Irteera Sistemak

- 7.1 Mikroprozesadore tipiko batean, I/O helbide desberdin bat erabiltzen da I/O datu-erregistroei erreferentzia egiteko. gailu jakin baterako I/O kontrolagailu bateko kontrol eta egoera erregistroetarako helbide ezberdina. Horrelako erregistroak portu gisa aipatzen dira. Intel 8088-n, bi I/O instrukzio formatu erabiltzen dira. Formatu batean, 8 biteko opcode-k I/O eragiketa bat zehazten du; honen atzetik 8 biteko ataka-helbide bat dago. Beste I/O opcode batzuek hori adierazten dute ataka helbidea 16 biteko DX erregistroan dago. Zenbat ataka dezake 8088 I/O helbideratze bakoitzean Modu? .

ŷ Garapena:

ŷ Memoria mapatutako I/O: RAM helbideak I/Orako gordeta daude

ŷ I/O kontrolatzalea: datuak, egoera eta kontrol erregistroak: ataka

ŷ 2 formatu

ŷ CodOP/Helbidea (zuzeneko helbidea): 8bit/8bit

ŷ CodOP/DX erregistroa (Zeharkako helbidea): 8bit/8bit ŷ DX:16bit

ŷ Portu kopurua: Zuzena ŷ  $2^8$  eta Zeharkako ŷ  $2^{16}$  ŷ guztira=  $256+65536=65792$  ataka

ŷ Eguzkia:

ŷ 65792 portu

- 7.2 Zilog Z8000 mikroprozesadore familial antzeko instrukzio-formatu bat erabiltzen da. Kasu honetan, bat dago zuzeneko ataka helbideratzeko gaitasuna, zeinetan 16 biteko ataka-helbide bat instrukzioaren parte den, eta zeharkakoa ataka helbideratzeko gaitasuna, zeinetan instrukzioak 16 biteko helburu orokorreko erregistroetako bati erreferentzia egiten dio, portuaren helbidea duen. Zenbat ataka hel dezake Z8000-k I/O helbideratze-modu bakoitzean?

ŷ Garapena

ŷ Zuzeneko modua:  $2^{16} = 64K = 65536$  ataka

ŷ Zeharkako modua:  $2^{16} = 64 K = 65536$  ataka

ÿ Eguzkia

ÿ 128K=131072 ataka

- 7.5 Sistema bat 8 biteko mikroprozesadore batean oinarritzen da eta bi I/O gailu ditu. Sistema honen I/O kontrolagailuek kontrol eta egoera erregistro bereiziak erabiltzen dituzte. Bi gailuek byte bateko datuak kudeatzen dituzte. Lehenengo gailuak bi egoera-lerro eta hiru kontrol-lerro ditu. Bigarren gailuak hiru egoera-lerro eta lau kontrol-lerro ditu.

to. Zenbat 8 biteko I/O kontrol modulu erregistro behar ditugu bakoitzaren egoera irakurtzeko eta kontrolatzeko gailua?

b. Zein da behar diren kontrol-modulu-erregistroen guztizko kopurua, lehenengo gailua irteera bat dela kontuan hartuta gailu bakarra?

c. Zenbat helbide desberdin behar dira bi gailuak kontrolatzeko?

ÿ eredu: periferikoen I/O kontrolagailuak periferikoarekiko interfaze diren atakak ezarri ditu. Portuak helbideragarriak dira eta erregistro-banku batez osatuta daude: datu-erregistroa, kontrol-erregistroa, egoera-erregistroa.

- Garapena:

ÿ 8 biteko datu-buffer: 2 datu-portu (sarrera, irteera) periferiko bakoitzeko.

ÿ 1 irakurketa-egoera-buffer (egoera-erregistroa) eta 1 idazketa-kontrol-buffer (kontrol-erregistroa) bakoitzeko periferiko bakoitza

ÿ egoera eta kontrol-lerroak ez dira helbide-lerroak, dagozkien portuetara konektatutako lineak baizik. Egoera-lerroak egoera-erregistro 1era eta kontrol-lerroak kontrol-erregistro 1era.

to. : 1 kontrol-erregistroa eta 1 egoera-erregistroa periferiko bakoitzeko

b. : A periferikoa (1Datu+1Egoera+1Kontrol) eta B periferikoa (2Datu+1Egoera+1Kontrol) = 7 erregistro

c. : adina helbide erregistro = 7 helbide

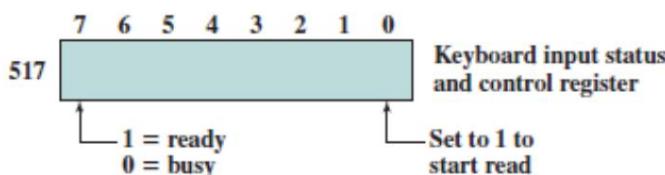
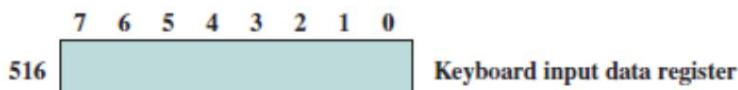
- Eguzkia:

to. 1 kontrol-erregistro eta 1 egoera-erreg

b. 7 erregistro

c. 7 norabide

- 7.6 Programatutako I/Orako, 7.5 irudiak adierazten du prozesadorea itxaron-begizta batean itsatsita dagoela I/O gailu baten egoera egiaztatzen. Eraginkortasuna areagotzeko, I/O softwarea idatzi liteke, prozesadoreak aldian-aldian gailuaren egoera egiazatzeko. Gailua prest ez badago, prozesadoreak beste zeregin batzuetara joan daiteke. Denbora tarte baten ondoren, prozesadorea berriro iztultzen da egoera egiazatzeko.



ADDRESS	INSTRUCTION	OPERAND	COMMENT
200	Load AC	"1"	Load accumulator
	Store AC	517	Initiate keyboard read
202	Load AC	517	Get status byte
	Branch if Sign = 0	202	Loop until ready
	Load AC	516	Load data byte

(a) Memory-mapped I/O

ADDRESS	INSTRUCTION	OPERAND	COMMENT
200	Load I/O	5	Initiate keyboard read
201	Test I/O	5	Check for completion
	Branch Not Ready	201	Loop until complete
	In	5	Load data byte

(b) Isolated I/O

**Figure 7.5** Memory-Mapped and Isolated I/O

to. Demagun goiko eskema 10ean funtzionatzen duen inprimagailu batera datuak karaktere bat ateratzeko segundoko karaktereak (cps). Zer gertatuko da bere egoera 200 ms behin eskaneatzen bada?

b. Ondoren, kontuan hartu karaktere bakarreko buffer bat duen teklatua. Batez beste, karaktereak 10 fps-ko abiaduran sartzen dira. Dena den, ondoz ondoko bi tekla sakatzearen arteko denbora tarteak 60 ms-koa izan daiteke. Zein maiztasunarekin eskaneatu behar du teklatua I/O programak?

- Garapena:

to. 10cps ý Periferikoak karaktere 1 transmititu behar ditu 100ms behin eta datuak portuan idazten ditu. PUZak idazketa bakoitzaren aurretik periferikoak idatzitako datuak gordetzen ez baditu, datuak galdu egiten dira. CPUak 200ms behin kontsultak egiten baditu eta periferikoak 100ms behin idazten badu, bigarren datu bakoitza galtzen da. Irtenbidea datu-buffera bi karakteretara handitzea edo kontsulta-maiztasuna 100 ms-ko epeetara handitzea izango litzateke.

b. Batez besteko abiadura 10 fps da baina gehieneko maiztasuna 60 ms. Eskaneatzeko maiztasuna CPUak gutxienez 1/60ms ý 16.66Hz izan behar du

- 7.10 Demagun etendurak gidatutako I/O erabiltzen duen sistema bat datuak transferitzen dituen gailu jakin baterako. 8 KB/s-ko batez bestekoa etengabe.

1. Demagun eten prozesamenduak 100 us inguru behar dituela (hau da, eten zerbitzuaren errutinara (ISR) jauzi egiteko, exekutatzeko eta programa nagusira itzultzeko denbora). Zehaztu I/O gailu honek prozesadorearen denboraren zein zati kontsumitzen duen byte bakoitzean eteten bada.
2. Orain demagun gailuak 16 byteko bi buffer dituela eta prozesadorea eteten duela bufferetako bat beteta dagoenean. Berez, etenen prozesamenduak denbora gehiago behar du, ISRk 16 byte transferitu behar dituelako. ISR exekutatzentz ari den bitartean, prozesadoreak 8 us inguru hartzen ditu byte bakoitzaren transferentziarako. Zehaztu I/O gailu honek prozesadorearen denboraren zein zati kontsumitzen duen kasu honetan.
3. Orain demagun prozesadorea bloke-transferentziako I/O instrukzio batekin hornituta dagoela, esaterako, Z8000-n aurkitutakoa. Honi esker, lotutako ISR-ak bloke baten byte bakoitza 2 us-tan transferi dezake. Zehaztu I/O gailu honek prozesadorearen denboraren zein zati kontsumitzen duen kasu honetan.

- Garapena:

- to. 8kB/s ý T\_int\_rq=1/8KB/s=125us ý frakzio=100us/125us=%80
- b. T\_etena\_zerbitzua=100us(ISR+1byte)+15bytesx8us=220us ý T\_16=16x125us=2ms ý zatikia=220us/2000us=0,11=%11
- c. T\_int\_serv=100us(ISR+1byte)+15bytesx2us=130us ý frakzio=130us/2000us=%6,5
- 7.11 DMA moduluak dituzten sistema ia guztietan, DMA memoria nagusirako sarbideari lehentasun handiagoa ematen zaio CPU memoria nagusirako sarbideari baino. Zergatik?
 

ý DMAC datuen buffer-a bete eta irakurtzen ez bada, datuak galduko lirateke.
  - 7.12 DMA modulu bat karaktereak memoriara transferitzen ari da ziklo-lapurreta erabiliz, 9600 bps-an transmititzen duen gailu batetik. Prozesadorea segundoko milioi bat instrukzioko (1 MIPS) abiadura jasotzen ari da. Demagun CPUak jarraibideak jasotzen dituela etengabe (ez da datuak eskuratzeko). Zenbat motelduko da prozesadorea DMA jardueraren ondorioz?
- ý 1MIPS ý 1 instrukzio mikrosegundo bakoitzean. PUZak jarraibideak etengabe eskuratzeko dituenez, autobusa okupatuta izango du mikrosegundo batean instrukzio bakoitza jasotzea eta instrukzio-zikloa osatzeko, beraz, sistema-bus-zikloa 1us da.
- ý 1 karaktere = 8 bit
- ý 9600 bps ý 1200 byte/s ý 1/1200 seg/byte =833us/byte ý byte bakoitza ziklo lapurretaz transferitzen da. Sistema-busa 833us behin lapurtzen da, hau da, 833 sistema-bus ziklo bakoitzean.
- ý DMAcik sistema-busa du ziklo baterako, hau da, 1 us.
- ý 833 ziklo bakoitzean DMAcik 1 ý 1/833 ý % 0,12 lapurtzen du
- ý 1MIPSx(1-0,0012)=998800 jarraibide segundoko.
- 7.13 Demagun autobus-zikloak 500 ns hartzen dituen sistema bat. Autobusaren kontrola bi norabidetan transferitzea, prozesadoretik I/O gailura edo alderantziz, 250 ns behar dira. I/O gailuetako batek 50 KB/s-ko datu-transferentzia-tasa du eta DMA erabiltzen du. Datuak byte batean transferitzen dira.
    1. Demagun DMA leherketa moduan erabiltzen dugula. Hau da, DMA interfazeak blokeen transferentzia hasi aurretik autobusaren maisutasuna lortzen du eta autobusaren kontrola mantentzen du bloke osoa transferitu arte. Zenbat denboraz lotuko luke gailuak autobusa 128 byteko bloke bat transferitzean?
    2. Errepikatu kalkulua zikloak lapurtzeko modurako.
  - Garapena:
 

ý 500ns sistemaren autobusaren ziklo-denbora

ý Ttx=1/50KB=20us/B. DMA-k datuak periferiotik jasotzen dituenez, memoria nagusira transferitzen ditu, eta sistema-busaren bidez datuak transferitzen ditu periferikoaren abiadura berean. Abiadura handiko periferikoetan bakarrik du zentzua.

    1. Burst modua:  $T=\text{access\_bus}+\text{tbus\_io\_transfer\_block}+\text{trelease\_bus}=250\text{ns}+128\times20\text{us}+250\text{ns}=2560\text{us}$
    2. Lapurreta zikloaren askapendibusa)=128x  $T=128\times(\text{bus\_access}+\text{tbus\_io\_byte\_transfer}+(250\text{ns}+20\text{us}+250\text{ns}))=128\times20.5\text{us}=2624\text{us}$
  - 7.16 DMA kontrolagailu batek jasotzeko soilik diren lau telekomunikazio esteka (bat DMA kanal bakoitzeko) 64 Kbps-ko abiadura dute.
    1. Kontrolagailua lehertzeko moduan edo ziklo-lapurtzeko moduan funtzionatuko al zenuke?
    2. Zein lehentasun-eskema erabiliko zenuke DMA kanalen zerbitzurako?
  - Garapena:
 

to. Orain DMA-ek lau datu-buffer izango ditu eta autobusera sartu ahal izango da bakarrarekin. Telekomunikazio-loturek kanala etengabe okupatzen dutenez (ahotsa edo datuak), komunikazioak irauten duen denbora osoan, leherketa moduak autobusa okupatuko luke denboraren %100ean.

Beraz, ziklo lapurreta aukeratu genuen.

b. 4 bezeroren arteko lehentasuna: lehentasun bera abiadura berdina baitute. Desberdinak balira abiadurak, azkarrenak, trafiko gehien duenak, izango luke abiadura handiena.

- 7.17 32 biteko ordenagailu batek bi hautatzaile kanal eta multiplexer kanal bat ditu. Kanal-hautatzaile bakoitzak bi disko magnetiko eta bi zinta magnetiko unitate onartzen ditu. Multiplexagailuaren kanalak bi lineako inprimagailu, bi txartel irakurgailu eta 10 VDT terminal ditu. Demagun transferentzia-tasa hauek:

ÿ Disko unitatea 800 KByte/s  
 ÿ Zinta magnetikoa 200 KByte/s  
 ÿ Lerro-inprimagailua 6,6 KByte/s  
 ÿ Txartel irakurgailua 1,2 KByte/s  
 ÿ VDT 1 KByte/s  
 ÿ Kalkulatu sistema honetako I/O transferentzia-tasa aggregatua.

to. Bi hautatzaile-kanalek periferiko berdinak dituzte. Hautatzaile-kanal bat bere periferikoei behin betiko esleituta dago eta esleitutako periferikoetako bat bakarrik zerbitza dezake. Multiplexatzaleak, berriz, guztientzako balio du ÿ  
 $Tasa = 800 + 800 + 2 \times 6,6 + 2 \times 1,2 + 10 \times 1 = 1625,6 \text{ KB/s}$

- 7.18 Ordenagailu bat prozesadore batek eta D I/O gailu batek M memoria nagusira konektatuta dago, hitz bateko datu-busaren zabalera duen bus partekatu baten bidez. Prozesadoreak segundoko 106 instrukzio exekutatu ditzake gehienez . Batez besteko instrukzio batek bost makina-ziklo behar ditu, eta horietako hiruk memoria-busa erabiltzen dute. Memoria irakurtzeko edo idazteko eragiketa batek makina-ziklo bat erabiltzen du. Demagun prozesadorea etengabe exekutatzen ari dela bere instrukzioen exekuzio tasaren %95a baina ezedozein I/O instrukzioa behar duten "atzeko planoan" programak, hau da, %5a I/O instrukzioak direla programa bakoitzeko I/O mekanismoa erabiltzen baduzu. Demagun prozesadore-ziklo bat bus-ziklo bat dela. Orain demagun I/O gailua M eta D arteko datu-bloke handiak transferitzeko erabili behar dela.

1. Programatutako I/O erabiltzen bada eta hitz bakarreko I/O transferentzia bakoitzak prozesadoreak bi instrukzio exekutatu behar baditu, kalkulatu ahal den I/O datu-transferentzia-tasa maximoa, segundoko hitzetan, D bidez.

2. Estimatu tasa bera DMA erabiltzen bada.

- Garapena:

to. Programaren I/O mekanismoa

Yo. Transferentzia programaz egiten da eta CPUak egiten du. Hitz 1 transferentzia bi instrukzio betetzea eskatzen du.

ii. 1 instrukzio = 3 makina-ziklo memoria-busarekin

iii. Bus-zikloa makina-ziklo baten baliokidea denez ÿ 3 bus-ziklo memoria-busarekin ÿ Hitz baten transferentziak 3 bus-ziklo behar ditu ÿ Bus-ziklo bakoitzean, hitzaren herena transferitzen da.

iv. Atzeko planoko programek PUZaren instrukzioen % 95 behar dute, eta I/Orako argibideen % 5 geratzen da

v. I/o argibideen %5etik, %2,5 transferentziak dira, bi i/o instrukzio behar baitira transferentzia bidez.

zerra.  $T_{transfer} (\text{hitz 1}) = 0.025 \times 106 \text{ instrukzio io/seg} = 25000 \text{ hitz/seg}$

b. DMA:

Yo. PUZak sistema-busa erabiltzen ez duen denbora = argibideen % 5 (5 ziklo bakoitzeko instrukzioa) GEHI argibideen % 95 (2 ziklo instrukzio bakoitzeko).

ii. PUZaren % 5eko exekuzioan, PUZA doakoa da:  $106 \text{ (inst/sec)} \times 0.05 \times 5 \text{ (ziklo/instr)} = \text{DMAk erabiltzen duen prozesadorearen } 250000 \text{ ziklo/seg} = \text{DMAk erabiltzen dituen I/O-ko } 250000 \text{ ziklo/seg}$

iii. PUZaren % 95eko exekuzioa, DMA-k sistema-busa partekatzen du = 5eko 106 x 0,95x2 ziklo libre zikloak= 1900000 ziklo/seg CPU= 1900000 ziklo/seg i/o

iv. Guztira=1900000+250000=2.150.000 ziklo/seg bus i/o

v. Ziklo bakoitzean transferentzia bat egin bidaiteke, hori izango litzateke abiadura maximoa. CPUak ez du memoria sartzeko eragiketa egiten du, memoria kontrolagailuak egiten du.

## 10.12. 8. kapitulua: Sistema eragilea

- 8.3 Programa batek  $C_i = \sum[a_{ij}]$  errenkaden batuketak kalkulatzen ditu  $100 \times 100$  den A array baten  $j=1,n$ . Demagun ordenagailuak 1000 hitzeko orrialde-tamaina duen eskaera orria erabiltzen duela eta Datuetarako esleitutako memoria nagusia bost orrialdeko fotograma da. Desberdintasunik al dago orriaren akatsen tasan A memoria birtualean errenkada edo zutabeen bidez gordeta balego? Azaldu.

ŷ A matrizea =  $100 \times 100$  hitz = 10.000 hitz

ŷ Memoria: 5 orrialdeko markoak: 5000 hitz

ŷ Prozesua: 10000 hitz  $10000/1000=10$  orrialdetan banatuko dira

ŷ Errenkadak biltegiratzea

ŷ 1. orrialdea:  $a_{1\_1}, a_{1\_2}, \dots, a_{1\_100}, a_{2\_1}, \dots, a_{2\_100}, \dots, a_{10\_1}, \dots, a_{10\_100}$  ŷ hamar ilara

ŷ 5. orrialdea:  $a_{41\_1}, \dots, a_{50\_100}$

ŷ 10. orrialdea:  $a_{91\_1}, \dots, a_{100\_100}$

ŷ xgarren orrialdea:  $a_{10}*(x-1)+1$ etik  $a_{10}^*x\_1$  arte ŷ hamar errenkada

ŷ Lehenengo errenkadaren exekuzioa:  $C_1 = \sum[a_{1j}] j=1..100$

ŷ Eskariaren orria:

ŷ MP hutsik dago, ez dago orri markorik hasieratu, orrialde guztiak diskoan, ez dago kopiarik  
MPincipal esparruan.

ŷ  $a_{11}$  atzematea ŷ FAULT (ez dago MPan, diskoan dago) ŷ kopiatu 1. orrialdea ŷ lortu C1

ŷ C2 exekuzioa ŷ a21 lehenengo orrialdean bada ŷ C2 lortzen du

ŷ C3, ..., C10 exekuzioa ŷ akatsik ez, lehen orrialdean daudelako

ŷ Execución C11 ŷ FAUT ŷ kopiatu 2. orrialdea

ŷ Execution C21 ŷ FAUT ŷ kopiatu 3. orrialdea

ŷ FAUTAS: C1, C11, C21, ..., C91

ŷ 100 Ci errenkadak exekutatzen diren bakoitzean, 10 hutsegite gertatzen dira

ŷ Emaitza berdina izango zen 5 orrialderen ordez orrialde bat edukiko balu pribatutasun politika  
ordezkao FIFO da

ŷ 4 orrialdeko markoak erabil ditzakezu datu berdinak eta ordezkapenak bertan egin  
markoa.

- Zutabeen biltegiratzea:

ŷ 1. orrialdea:  $a_{1\_1}, a_{2\_1}, \dots, a_{100\_1}, a_{1\_2}, \dots, a_{100\_2}, \dots, a_{1\_10}, \dots, a_{100\_10}$  ŷ hamar zutabe

ŷ xgarren orrialdea:  $a_{1\_10}*(x-1)+1$ etik  $a_{100\_10}^*x$  arte ŷ hamar zutabe

ŷ 1. errenkadako exekuzioa:  $C_1 = \sum[a_{1j}] j=1..100$

ŷ 1. errenkadaren 100 zutabeak kargatu behar ditut ŷ 10 orrialde behar ditut orrialde bakoitzeko hamar zutabe ŷ  
10 AKATSE

ŷ Exekuzio-zenbakien errenkada: 100 zutabe behar dira, 10etik 10eko orrialdeetan banatuta.  
zutabeak. ŷ 10 orrialde behar dira ŷ 10 AKATSE

ŷ 100 Ci errenkadak exekutatzen diren bakoitzean: 10 akats errenkada bakoitzeko ŷ 1000 akatsak

- 8.4 Demagun partizio-eskema finko bat, 216 byteko tamaina berdinak partizioak eta guztira 224 byteko memoria nagusiaren tamaina dituena. Prozesu-taula bat mantentzen da, prozesu egoiliar bakoitzeko partizio baten erakuslea barne hartzen duena. Zenbat bit behar dira erakuslearentzat?

ŷ Deskribatzaire-taulak indizeak eta edukiak osatzen dituzte, kasu honetan a  
erakuslea.

ÿ 2 24/216=28 memoria-partizio ÿ 216-ren

tamaina ÿ 0000z hamaseimalez amaitzen diren helbideak ÿ k\*216ÿ0xnn0000 helbideak

ÿ Erakuslea: nn zifrak altuenak bakarrik gorde behar dira ÿ 8 bit

ÿ gero 16 bit ezkerrera desplazatzen dira oinarrizko helbidea izateko

- 8.6 Demagun prozesadorean une honetan exekutatzen ari den prozesuaren orrialde-taulak honako itxura duela.
- Zenbaki guztiak hamartarrak dira, dena zenbakituta dago zerotik hasita eta helbide guztiak memoria byte-helbideak dira.  
Orrialdearen tamaina 1024 byte da.

<b>Virtual page number</b>	<b>Valid bit</b>	<b>Reference bit</b>	<b>Modify bit</b>	<b>Page frame number</b>
0	1	1	0	4
1	1	1	1	7
2	0	0	0	—
3	1	0	0	2
4	0	0	0	—
5	1	0	1	0

103. Irudia VM

•

1. Deskribatu zehatz-mehatz nola, oro har, PUZak sortutako helbide birtual bat memoria nagusi fisikoko helbide batera itzultzen den.

ÿ Helbide birtuala eremuek (VPN,VPO) ÿ (oinarria, desplazamendua) osatzen dute. Orrialde birtualeko taula erabiliz VPN PPNra itzultzen dugu. Helbide fisikoa parea (PPN,PPO) da non desplazamendua PPO=VPO

ÿ Zergatik dago orrialde birtuala cachean orrialde birtualeko taulan? Balidazio-bitak badu  
1 balio du.

2. Zein helbide fisikori dagokio, halakorik balego, honako helbide birtual hauetako bakoitza? (Ez egin  
saiatu orrialdeko akatsak kudeatzen, halakorik balego.)

to. 1052

ÿ VPN=Modua{1052/1024}=1 ÿ Baliozko bit=1 ÿ PPN=7

ÿ VPO=Atseden{1052/1024}=28

ÿ Helbide fisikoa = 7\*1024+28=7196

b. 2221

ÿ VPN=Modua{2221/1024}=2 ÿ Baliozko bit=0

ÿ Orrialde horren kopiarik ez dago, beraz, ezin da itzulpena egin

c. 5499

ÿ VPN=Modua{5499/1024}=5 ÿ Baliozko Bit=1 ÿ PPN=0

ÿ VPO=Atseden{1052/1024}=379

ÿ Helbide fisikoa = 0+379=379

- 8.8 Prozesu batek bost orrialde aipatzen ditu, A, B, C, D eta E, hurrenkera honetan: A; B; C; D; TO; B; ETA; TO; B; C; D;  
E .Demagun ordezkapen-algoritmoa lehena-lehena irteten dela eta aurkitu zenbat orrialde-transferentzia-kopurua erreferentzia-sekuentzia honetan hiru orrialdeko marko dituen memoria nagusi huts batekin hasita. Errepikatu lau orrialdeko markoetarako.

1. MP ÿ 3 orrialdeko markoak; FIFO politika

ÿ v/v/v; Aÿ A/v/v ; Bÿ A/B/v ; Cÿ A/B/C; Dÿ D/B/C; Aÿ D/A/C; Bÿ D/A/B; Eÿ E/A/B ; Aÿ

E/A/B; B/y E/A/B; C/y E/C/B; D/y E/C/D; E/y E/C/D

ŷ 10 hutsegite

## 2. MP ŷ 4 orrialdeko markoak; FIFO politika

ŷ v/v/v/v; A/y A/v/v/v ; B/y A/B/v/v ; C/y A/B/C/v; D/y A/B/C/D; A/y A/B/C/D; B/y A/B/C/D;  
E/y E/B/C/D; A/y E/A/C/D; B/y E/A/B/D; C/y E/A/B/C; D/y D/A/B/C; E/y D/E/B/C

- 8.9 Orrialde birtualen zenbaki sekuentzia hau exekutatzen ari den bitartean aurkitzen da memoria birtuala duen ordenagailua: 3 4 2 6 4 7 1 3 2 6 3 5 1 2 3 Demagun gutxienez duela gutxi erabilitako orria ordezko politika onartzen da. Marraztu orrialdearen arrakasta-ratioaren grafiko bat (orriaren erreferentziaren zatia memoria nagusian dago) n 1 ŷ n/y8-rako memoria nagusiko orrialde-ahalmenaren arabera. Demagun nagusi hori memoria hutsik dago hasieran.

fotograma zk	Zatikien hits			
1	0	3	4	2
2	0	4	2	6
3	2/15	4	4	7
4	3/15	4	4	1
5	4/15	4	4	3
6	7/15	4	2	3
7	8/15	3	4	2
8	8/15	3	4	2
		2	6	3
		3	5	1
		1	2	3
		3	2	3
		1	2	3
		1	2	3

- 8.11 Demagun programaren adierazpena

izan ere (i = 1; i <= n; i++)

a[i] = b[i] + c[i];

- 1000 hitzeko orrialde-tamaina duen memoria batean exekutatzen da. Izan bedi n = 1000. betea duen makina bat erabiliz Erregistrok erregistroko argibide sorta eta indize-erregistroak erabiltzen ditu, programa hipotetiko bat idatzi aurreko adierazpena gauzatu. Ondoren, erakutsi orriaren erreferentziaren sekuentzia exekuzioan zehar.

ŷ Kodea zein datuak memoria birtualean egongo dira

ŷ 1000 hitz-markoak.

ŷ Kargatu/gorde arkitektura-programa (helbide-espazio birtuala)

```

KODEA ATALA
Ri <- 1
Ra <- n

loop_start: R1 <- b[Ri]
            R2 <- c[Ri]
            R3 <- R1+R2
            a[Ri] <- R3
            Banderak <- Ri<Ra
            Banderak: PC <- loop_start
            CPU <- gelditu
            HASIERAKO DATUEN ATALA

bat:      1
n:       1000
          array a[1000]
b:       b array [1000]

```

C: array **c[1000]**

ÿ Espazio birtuala esleitzea

ÿ PV1 orrialdeko kodea

ÿ A matrizeak orrialde bat hartzen du ÿ PV2

ÿ B arrayak orrialde bat hartzen du ÿ PV3

ÿ C arrayak orrialde bat hartzen du ÿ PV4

ÿ bat eta n datu motako orrialde batean ÿ PV5

ÿ Exekuzioa

ÿ 1515(131411211)100011

- 8.13 Demagun sistema informatiko bat segmentazioa eta orrialdea dituena. Segmentu bat memorian dagoenean, hitz batzuk alferrik galtzen dira azken orrialdean. Horrez gain, s segmentu tamainarako eta p orrialderako tamainarako, s/p orrialde-taularen sarrerak daude. Zenbat eta txikiagoa izan orriaren tamaina, orduan eta hondakin gutxiago segmentuaren azken orrialdean, baina orduan eta handiagoa da orrialde-taula. Zein orriaren tamainak gutxitzen du gainkostu osoa?

ÿ Garapena:

1. Segmentu bakoitzeko orrialde kopurua: segmentuaren tamaina/orriaren tamaina = s/p
2. Segmentu bakoitzak bere orri-taula du
3. Orrialdearen tamaina murrizten badugu, barne zatiketa murrizten da baina orrialde-taularen sarrera kopurua handitzen da.
4. Guztira alferrik galduen hitzak (w) segmentu bakoitzaren azken orriei gehi orri-taularen tamainari eragiten dien hondakina da. Segmentu guztien barne-zatiketaren batez besteko balioa p/2 da eta orri-taularen tamaina s/p ÿ w=p/2+s/p ÿ dw/dp= 1/2 taula-sarrera kopuruarekiko proportzionala da. -s/p2=0 ÿ p 2=2s

- 8.14 Ordenagailu batek cache bat, memoria nagusia eta memoria birtualerako erabiltzen den disko bat ditu. Erreferentziazko hitz bat cachean badago, 20 ns behar dira bertara sartzeko. Memoria nagusian badago baina ez cachean, 60 ns behar dira cachean kargatzeko, eta orduan erreferentzia berriro hasten da. Hitza memoria nagusian ez badago, 12 ms behar dira hitza diskotik ateratzeko, 60 ns ondoren cachean kopiatzeko, eta, ondoren, erreferentzia berriro hasten da. Cache-ko arrakasta-erlazioa 0,9koa da eta memoria nagusiko sakatze-ratioa 0,6koa. Zein da sistema honetan erreferentziatutako hitz batera sartzeko behar den batez besteko denbora ns-tan?

ÿ  $T = \text{hit\_cache} * t_{\text{acc\_ca}} + (1 - \text{hit\_cache}) \text{hit\_main}(t_{\text{main\_cache}} + t_{\text{acc\_ca}}) + (1 - \text{hit\_cache})(1 - \text{hit\_main})(t_{\text{acc\_disk\_main}} + t_{\text{main\_cache}} + t_{\text{acc\_ca}}) = 0.9 * 20 + 0.1 * 0.6 * (60 + 20) + 0.1 * 0.4 * (12000000 + 60 + 20) = 480us$

- 8.15 Demagun ataza bat tamaina berdinako lau segmentutan banatuta dagoela eta sistemak segmentu bakoitzeko zortzi sarrerako orrialde deskribatzaile-taula bat eraikitzen duela. Horrela, sistemak segmentazioa eta orrialdea konbinatzen ditu. Era berean, demagun orrialdearen tamaina 2 Kbytekoa dela.

1. Zein da segmentu bakoitzaren gehieneko tamaina?

2. Zein da zereginaren gehienezko helbide-espazio logikoa?

3. Demagun ataza honen bidez 00021ABC kokapen fisikoko elementu batera sartzen dela. Zein da atazak horretarako sortzen duen helbide logikoaren formatua? Zein da sistemaren helbide fisikoaren gehienezko espazioa?

ÿ Garapena:

1. PAGE taula: 8 sarrera: 2KB-ko 8 orrialde birtual ÿ Segmentua: 8 \* 2KB = 16KB.

to. Ez du ezer esaten baina... SEGMENTUAK taulak sarrera bat izango du segmentu bakoitzeko. Segmentu-sarrera bakoitzak orrialde-taula ezberdin batera adieraziko du. Orrialde bateko taula segmentu bakoitzeko.

2. Prozesua: 4 segmentu ÿ 4 \* 16KB = 64KB

3. Helbide logikoa ѕ Formatua (Segmentua, Orria, VPO) ѕ (4seg,8page,2KB) ѕ  
(2bit,3bit,11bit) ѕ 16bit helbide logikoa
    - to. Helbide fisikoa ѕ 00021ABC ѕ 8 zifra hex ѕ 32 bit ѕ 4GB b. Orrialde-markoak ѕ  
4GB/2KB ѕ 2\*220 markoak
    - c. 00021ABC ѕ 0000-0000-0000-0010-0001-1010-1011-1100 ѕ markoa/desplazamendua ѕ 21/11 ѕ  
00000000000000001000011 / 010101111700 ѕ markoa desplazamendua
  4. Itzulpena: Desplazamendu birtual eta fisiko berdina (11bit) ѕ Segmentu logikoak (2bit) orrialde-taula bat seinalatzen du. Orri birtuala (3 bit) orriaren taularen desplazamendua da. Orrialde-taularen sarrera bakoitzera memoria nagusiko fotograma baten erakuslea da (21 biteko oinarrizko helbidea). Segmentuen deskribapen-taula eta segmentu bakoitzaren lau orrialde-taulak asmatzeko galdera gehituko da. Ariketa honetan helbide logikoak 01010111100 desplazamendua izango du eta seg/orria bikotearen 5 bitak ezin dira ezagutu 67 fotogramako erakuslea zein taula eta posiziota dagoen jakin beharko litzatekeelako.
- 8.16 Demagun mikroprozesadore bat memoria nagusi fisikoaren 232 byteko sartzea gai dena. 231 byteko gehienezko tamainako helbide logiko segmentatu bat ezartzen du. Instrukzio bakoitzak bi zatitako helbide osoa duka. Kanpoko memoria kudeatzeko unitateak (MMU) erabiltzen dira, zeinen kudeaketa-eskemak 222 byte -ko tamaina finkoko memoria fisikoaren ondoko blokeak esleitzenten dizkie segmentuei. Segmentu baten hasierako helbide fisikoa 1024z zatigarria da beti. Erakutsi helbide logikoak helbide fisikoetara bihurtzen dituen kanpoko mapa-mekanismoaren interkonexio zehatz MMU kopuru egokia erabiliz, eta erakutsi MMU baten barne-egitura zehatz (MMU bakoitzak dela suposatuz). zuzenean mapatutako 128 sarrerako segmentu deskribatzaileen cachea duka) eta MMU bakoitza nola hautatzen den.
- ѕ 231 byteko espazio birtual segmentatua : ez da segmentu bakoitzaren espazio birtuala baizik  
segmentu guztiak.
- ѕ bi zati dituen helbide logikoa ѕ (segmentua, desplazamendua)
- ѕ MP: 232 byte-ko espazioa 222- byte-ko bloke ondokoekin segmentu bakoitzeko  
 ѕ 22 byteko desplazamendua
- ѕ 2 31/222 = 29 segmentu espazio birtualean ѕ 9 bit helbide birtualeko segmentuarentzat eta bat  
512 sarrera dituen segmentu-taula
- ѕ 31 byteko helbide logikoa (9,22) ѕ (seg, desplazamendua)
- ѕ MP: 1K-ren multiploetan lerrokatutako segmentuak  
 ѕ segmentu fisikoa: pisu txikiena duten 10 bitak zero dira eta pisu handiena duten 22ak taulan.  
segmentuak.
- ѕ Helbide fisikoa: segmentua+desplazamendua
- ѕ MMU: 128 sarrerako segmentu-taula (27 )
1. Ez dugu 512 sarrera dituen taula bat, 128ko lau taula baizik.
  2. MMU kopurua: espazio birtualean 29 segmentu baditugu eta MMUk 27ko taula badu  
4 MMU beharko ditugu.
  3. Itzulpena: espazio logikoa (9,22)(seg,offset) 32 byteko segmentu batean+offset helbide batean.
    - to. 9 segmentu birtualen bitetik, bi bitek MMU hautatuko dute eta beste zazpi bitek sarrera hautatuko dute.  
segmentu-taulatik. (2,7,22)
    - b. 9 segmentu logikoko bitak segmentu fisiko baten 22 bit altuak dituen segmentu-taularen indizea dira.
    - c. Desplazamendu fisikoak ere 22 byteko tamaina du
    - d. helbide fisikoa: oinarrizko helbidearen multiploa 1K gehi 22 byteko desplazamendua.
- 8.17 Demagun orrialdedun helbide-espazio logiko bat (2 Kbyte-ko 32 orrialdez osatua) 1- batean mapatutakoa.  
Mbyte memoria fisikoko espazioa.

1. Zein da prozesadorearen helbide logikoaren formatua?
2. Zein da orrialde-taularen luzera eta zabalera ("sarbide-eskubideak" bitak alde batera utzita)?
3. Zer eragin du orrialde-taulan memoria fisikoaren espazioa erdira murrizten bada?

ŷ Garapena:

ŷ MP: 1MB 2KB orriekin ŷ  $2^{20}/2^{11} = 2^9$  orrialde-markoak

1. VPN/OFFSET ŷ VPN: 32 orrialde 25 dira , 5 bit; OFFSET: 2KB 211, 11bit esan nahi du

2. Orri-taula: orrialde birtualen kopuruaren luzera = 32 eta zabalera erakuslearen berdina  
29 fotogrametako batera , hau da, 9 biteko.

3. MP erdira murrizten bada, fotograma kopurua ere erdira murrizten da ŷ 2  
orrialde-markoak ŷ 8 biteko zabalera.

8

- Randal 9. kapitulua: 9.19 irudiak helbide birtualen eta fisikoen formatuak erakusten ditu. Orrialde bakoitzak  $26 = 64$  byte denez, helbide birtualen eta fisikoen ordena baxuko 6 bitek VPO eta PPO gisa balio dute hurrenez hurren. Helbide birtualeko maila altuko 8 bitek VPN gisa balio dute. Helbide fisikoaren maila handiko 6 bitek PPN gisa balio dute. 9.20 irudiak gure memoria sistema txikiaren argazki bat erakusten du, TLB barne (9.20(a) irudia), orrialde-taularen zati bat (9.20(b) irudia) eta L1 cachea (9.20(c) irudia). TLB eta cachearen zifren gainetik, helbide birtualen eta fisikoen bitak hardwareak nola partitzen dituen ere erakutsi dugu gailu hauetara sartzen den heinean.

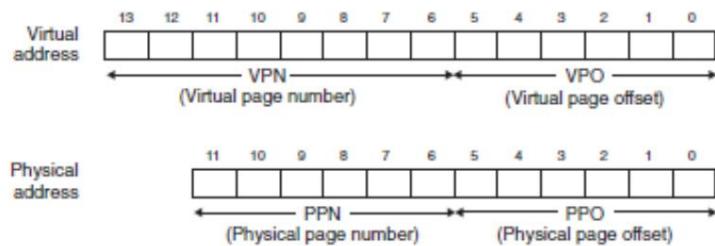
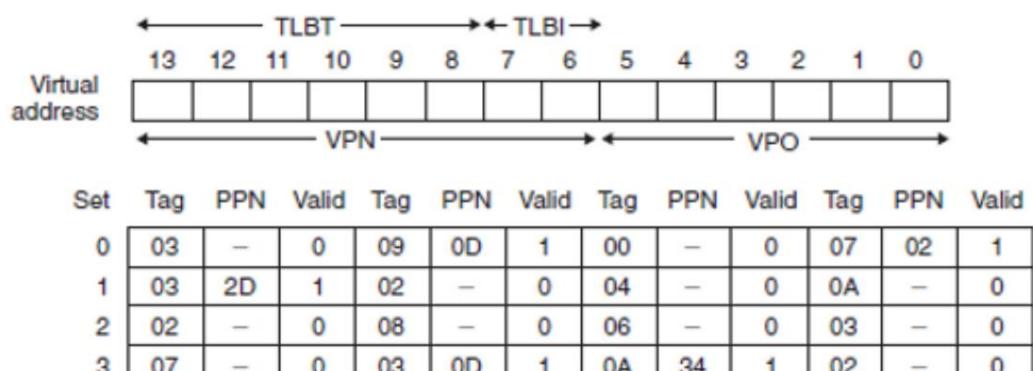


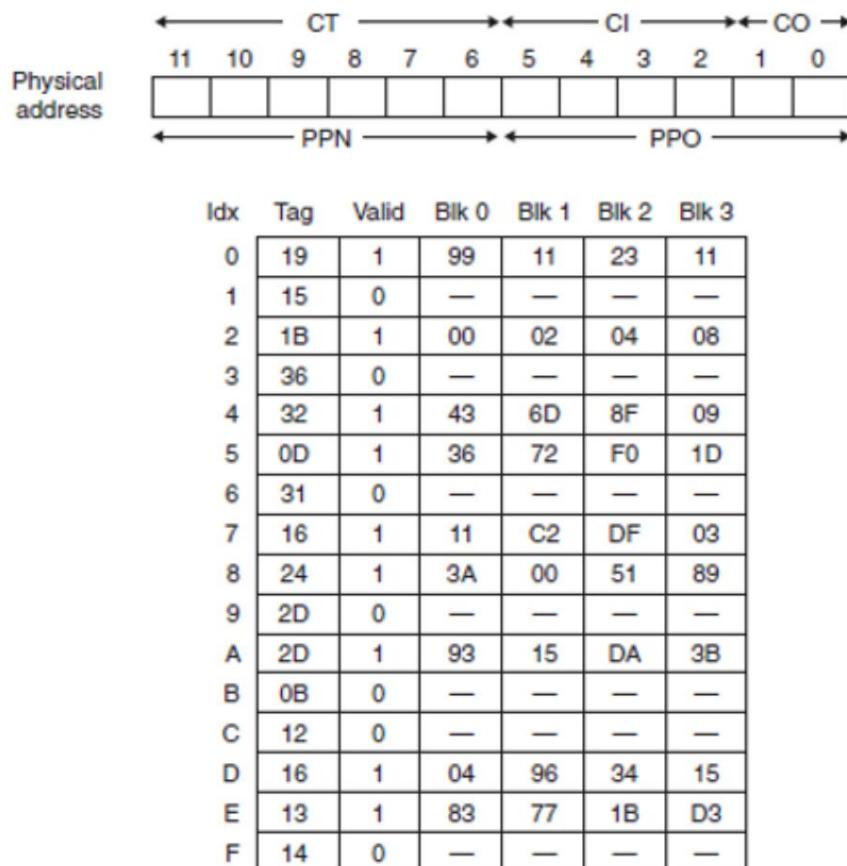
Figure 9.19 Addressing for small memory system. Assume 14-bit virtual addresses ( $n = 14$ ), 12-bit physical addresses ( $m = 12$ ), and 64-byte pages ( $P = 64$ ).



(a) TLB: Four sets, 16 entries, four-way set associative

VPN	PPN	Valid	VPN	PPN	Valid
00	28	1	08	13	1
01	-	0	09	17	1
02	33	1	0A	09	1
03	02	1	0B	-	0
04	-	0	0C	-	0
05	16	1	0D	2D	1
06	-	0	0E	11	1
07	-	0	0F	0D	1

(b) Page table: Only the first 16 PTEs are shown



(c) Cache: Sixteen sets, 4-byte blocks, direct mapped

**Figure 9.20** TLB, page table, and cache for small memory system. All values in the TLB, page table, and cache are in hexadecimal notation.

ÿ Hasierako konfigurazio hau ikusita, ikus dezagun zer gertatzen den PUZak irakurtzen duen karga-instrukuzio bat exekutatzenean.  
0x03d4 helbidean dagoen bytea

ÿ Irtenbidea

ÿ LLLT:0x03

ÿ TLBT:0x3

ÿ VPN:0x0f

ÿ VPO:0x14

ÿ PPN=0x0D

ÿ helbide fisikoa=0x354

ÿ CO=0x0

ÿ CI=0x5

ÿ CT=0x0D

ÿ Datuak=0x36

#### 10.13. 12. kapitulua: prozesadorearen egitura eta funtzioa (9. edizioko 14. kapitulua)

- 12.1 a. 8 biteko hitza duen ordenagailu batean egindako azken eragiketa bat izan bada, zeinetan bi eragigaiak 00000010 eta 00000011 ziren, zein izango litzateke honako bandera hauen balioa?

ÿ Eraman

ÿZero

ÿ Gainetik ÿ Sinatutako zebakia

ÿ Sinatu

ÿ Parekidetasuna ÿ Parekidetasuna

ÿ Erdi-Eramatea

ÿ Garapena

ÿ 0010+0011=0101 ÿ MSBn ez dago garraiorik, emaitza ez da zero, ez dago gainezkarik, positiborik, BIKI-kopururik, ez dago 3. posizioan eramanik. Beraz, bandera guztiak desgaituta daude, izan ezik. parekidetasun berdinakoa. Parekidetasunaren bandera 1ean izango da.

- 12.3 Mikroprozesadore bat 5 GHz-ko abiaduran dago.

1. Zenbat irauten du erloju-ziklo batek?

2. Zein da hiru erloju-ziklo osatutako makina-instrukuzio mota jakin baten iraupena?

ÿ Garapena

ÿ a.  $T = 1/f = 1/(5 \cdot 10^9) = 0,2\text{ns}$

ÿ b.  $3T = 3 \cdot 0,2 = 0,6\text{ns}$

- 12.4 Mikroprozesadore batek memoriaren eremu batetik bestera byte kate bat mugitzeko gai den instrukzio bat eskaintzen du. Instrukzioa lortzeko eta hasierako deskodetzeak 10 erloju-ziklo behar ditu. Hortik aurrera, 15 erloju-ziklo behar dira byte bakoitzak transferitzeko. Mikroprozesadorea 10 GHz-ko abiaduran dago.

1. Zehaztu instrukzio-zikloaren luzera 64 byteko kate baten kasuan.

2. Zein da etenaldi bat aitortzeko kasurik txarrena, instrukzioa etenik ez bada?

3. Errepikatu (b) zatia byte transferentzia bakoitzaren hasieran instrukzioa eten daitekeela suposatuz

ÿ Garapena

ÿ a) IC=Irakaskuntza Zikloa= FI+DI+CO+FO+EI+WO; IF+DI=10T ; CO+FO= 0 ; EI= 15T/byte ;  
 WO=0 ; T=1/(10\*109 ))=0,1 ns; IC=(10+15\*64)\*T=970\*0.1= 97ns

ÿ b) Instrukzioa hasi bezain laster, ziklo osoa geratuko litzateke bertaratu ahal izateko.  
 etenaldia: 97 ns.

ÿ c) Lehen transferentzia baino lehen eteten bada 10T beharko luke gehienez, eta transferentzietaan  
 eteten bada 15T izango litzateke. Beraz, kasurik txarrena 15T=15\*0.1=1.5ns izango litzateke

- 12.5 Intel 8088 bus-interfaze-unitate batek (BIU) eta exekuzio-unitateak (EU) osatzen dute, 2 etapako kanalizazioa osatzen dutenak. BIUk 4 byteko instrukzio-ilara batean jasotzen ditu argibideak. BIUk helbideen kalkuluetan ere parte hartzen du, eragigaiak eskuratzeko ditu eta emaitzak memorian idazten ditu EBk eskuatu bezala. Halako eskaerarik ez badago eta autobusa doakoa bada, BIUk argibide-ilaran dauden hutsuneak betetzen ditu. EBk instrukzio bat exekutatzen amaitzen duenean, edozein emaitza BIUra pasatzen du (memoriari edo I/Ora zuzendua) eta hurrengo instrukziora jarraitzen du. [wikipedia](#): 8088ak **8 biteko** kanpoko datu-busa zuen

1. Demagun BIUk eta EBk egiten dituzten zereginak denbora berdina hartzen dutela. Zein faktorerekin hobetzen du kanalizazioak 8088ren errrendimendua? Ez ikusi adar-argibideen eragina.
2. Errepikatu kalkulua EBk BIUren bikoitza hartzen duela suposatuz.

ÿ Garapena

ÿ 0. 8088 mikroak 8 biteko datu-busa du. Exekuzio unitatea ALU eta Erregistroak osatzen dute. BIUk EBrekin batera bi unitate dituen CPU segmentatu bat osatzen dute. Prefetch instrukzio-ilara exekutatu beharreko hurrengo instrukzioa gordetzen duen buffer-a da.

ÿ a. Etapa batek x hartzen du eta hurrengoak ere x. Lehenengo instrukzioa 2x behar da exekutatzeko eta tarte bikoitz bakoitzean berri bat igortzen da.

ÿ b.  $x+2x=3x$ . Bigarren instrukziotik 2x hartzen dute. 3x instrukzio-ziklo batean, argibideak  $2x \div (3x)$  ziklo-denbora) / (2x denbora/instrukzioa) = 1,5 aldiz jarraibide gehiago ziklo bakoitzeko ematen dira.

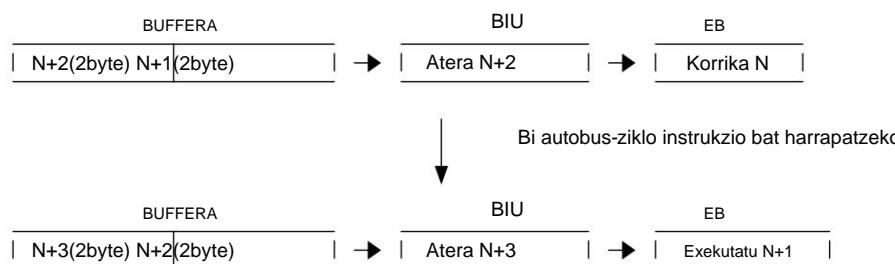
- 12.6 Demagun 8088 bat programa bat jauzi egiteko probabilitatea 0,1ekoak den programa bat exekutatzen ari dela. Simpletasunerako, demagun argibide guztiak 2 byte direla. Eskuratu aurretiko instrukzio-ilara hutsik badago, EBk hurrengo instrukzio-bytea eskuratu eta ilaren goialdera eraman arte itxarongo du. EBk adar edo salto-instrukzio bat exekutatzen duenean, kontrola beste instrukzio sekuentzial multzo bati dagokion kokapen batera transferitzen du. Hori gertatzen den bakoitzean, BIU-k automatikoki berrezartzen du ilara eta, ondoren, kokapen berri horretatik argibideak jasotzen hasten da ilara berriz betetzeko.

1. Zer zati alferrik galtzen da autobus-zikloak lortzeko?
2. Errepikatu instrukzio-ilarak 8 byteko luzera badu.

ÿ Buffer (4 byte bi instrukzioetarako) ÿ BIU ÿ EU: instrukzio bat irakurtzeko, **BI bus ziklo** behar dira , datu-bus bat (byte 1) bus ziklo bakoitzeko.

ÿ Garapena

ÿ 0. N instrukzioaren exekuzioan jauzirik ez badago, N+1 instrukzioa atzematen da.



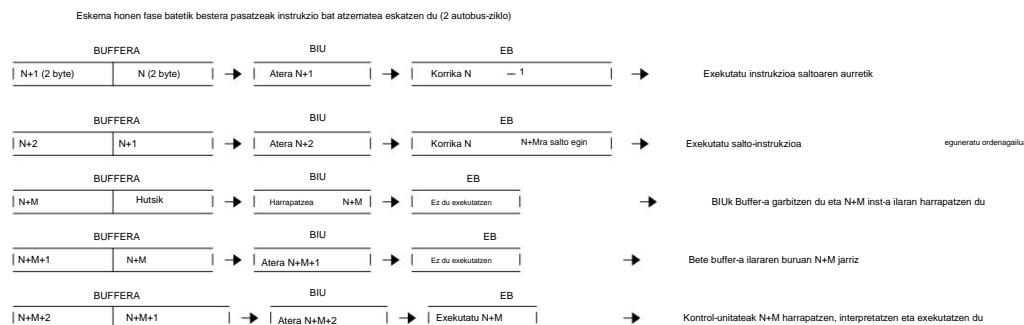
ÿ 1. N instrukzioaren exekuzioak M instrukzioen saltoa badakar, instrukzioa ez da exekutatuko.

Bufferean itxaroten den N+1 instrukzioa, baina N+M instrukzioa exekutatu behar da.

N-ren exekuzioan, ez da ezer harrapatzzen, kontagailua eguneratzen da.

Programatu N+M-ra eta hurrengo instrukzio-zikloan N+M harrapatuko da. Beraz, badago salto egin, harrapatzeko zikloa gutxi erabiliko da eta harrapatzeko buffera garbitu beharko da. argibideak.

### ÿ Interpretazioa



ÿ a. BIUk N+1 memoria nagusitik lortzen duenean eta N atzean ilaran jartzen duenean, instrukzioa harrapatzeko behar diren bi autobus zikloak alferrik galduz Exekutatuko ez den M+1. Gauza bera gertatzen da N+2 memoria nagusitik harrapatzearekin. Beraz, N+1 eta N+2-ren sistema-bus-zikloak alferrik galtzen dira, hau da, 4 ziklo autobusa.

ÿ Instrukzio-buffer-a 4 bytekoa da aurreko arketaren arabera, beraz, beharrezko da "bulkatu" N+1 eta N+2 4 byteak N+M instrukzio berria pasatzen uzteko. memoria nagusitik ateratzen da.

ÿ No\_jump instrukzioa harrapatzeko, ondo erabilitako 2 autobus-ziklo behar dira, instrukzioa Buffer-aren buruan egongo da CPUak exekutatu behar duenean. Jauzi-instrukzio batena Memoria nagusitik harrapatzeko 2 ziklo ondo erabilita esan nahi du, baina 4 ziklo gaizki buffera hustu eta N+M instrukzioa ilaratik burura eramateko erabiltzen da buffer-aren, CPUa zain dagoen bitartean. 100 instrukzio bakoitzeko 100 izango ditugu argibideakx2ziklo/instr ondo erabilita eta 10 argibidex4ziklo/instrukzio gaizki erabilita erabilia ÿ guztira 240 ziklo ÿ azpierabilera frakzioa = 40/240 = 0,166 = % 17 Autobusa bilatze lanetan erabiltzen ez den bitartean, BIU busa garbitzen ari da. buffer.

ÿ b. 8 byteko ilararekin ÿ Guztira=100x2+10\*8=280ÿ eraginkortasun eza=80/280=0,285=%28,5

- 12.7 Demagun 12.10 irudien denbora-diagrama. Demagun bi etapako kanalizazioa soilik dagoela (eskuratu, exekutatu). Marraztu berriro diagrama lau instrukzioetarako zenbat denbora-unitate behar diren erakusteko.

ÿ Garapena

	1	2	3	4	5	
I1	FI	EI				
I2		FI	EI			
I3			FI	EI		
I4				FI	EI	

ÿ 5 Denbora-unitate behar dira

- 12.9 Prozesadore kanalizatu batek 2,5 GHz-ko erloju-abiadura du eta 1,5 milioi instrukzio dituen programa bat exekutatzen du.

Kanalizazioak bost etapa ditu, eta argibideak erloju-ziklo bakoitzeko bat ematen dira.

Ez ikusi adar-argibideen eta sekuentziaz kanpoko exekuzioen ondoriozko zigorrak.

1. Zein da prozesadore honek programa honetarako duen bzikortzea kanalizatu gabeko prozesadore batekin alderatuta, 12.4 atalean erabilitako hipotesi berdinak eginez?

2. Zein da kanalizatutako prozesadorearen errendimendua (MIPS eta?)?

ÿ Garapena

ÿ a. Iraupena Programa N instrukzioekin, t iraupeneko k etaparen segmentazioa bakoitza = 1. instrukzioa gehi gainerakoa =  $k*t + (N-1)t = t(N+k-1)$  = for  $N >> k = t^* (N-1)$ . Sin\_sec/con\_sec =  $N*k*t / t(N+k-1) = Nk/(N+k-1)$  erlazioa. N infiniturako joera badu ÿ  $Nk/N=k=5$

ÿ b. Errendimendua = programaren argibideak/programaren iraupena =  $N/\{t^*(N+k-1)\}=1/t$

- 12.11 Demagun n luzera duen instrukzio-sekuentzia bat, instrukzio-tubidean zehar igortzen dena. Izan bedi p baldintzapeko edo baldintzarik gabeko adar-instrukzio bat topatzeko probabilitatea, eta q izan bedi l adar-instrukzio baten exekuzioak segidan ez den helbide batera jauzia eragiteko probabilitatea. Demagun horrelako jauzi bakoitzak kanalizazioa garbitu behar duela, etengabeko instrukzio-prozesamendu guztiak suntsituz, azken fasetik ateratzen naizenean. Berrikusi (12.1) eta (12.2) ekuazioak probabilitate horiek kontuan hartzeko.

ÿ Garapena

ÿ Exekuzioa segidakoa ez den jauzi bat duten argibideak: pqn

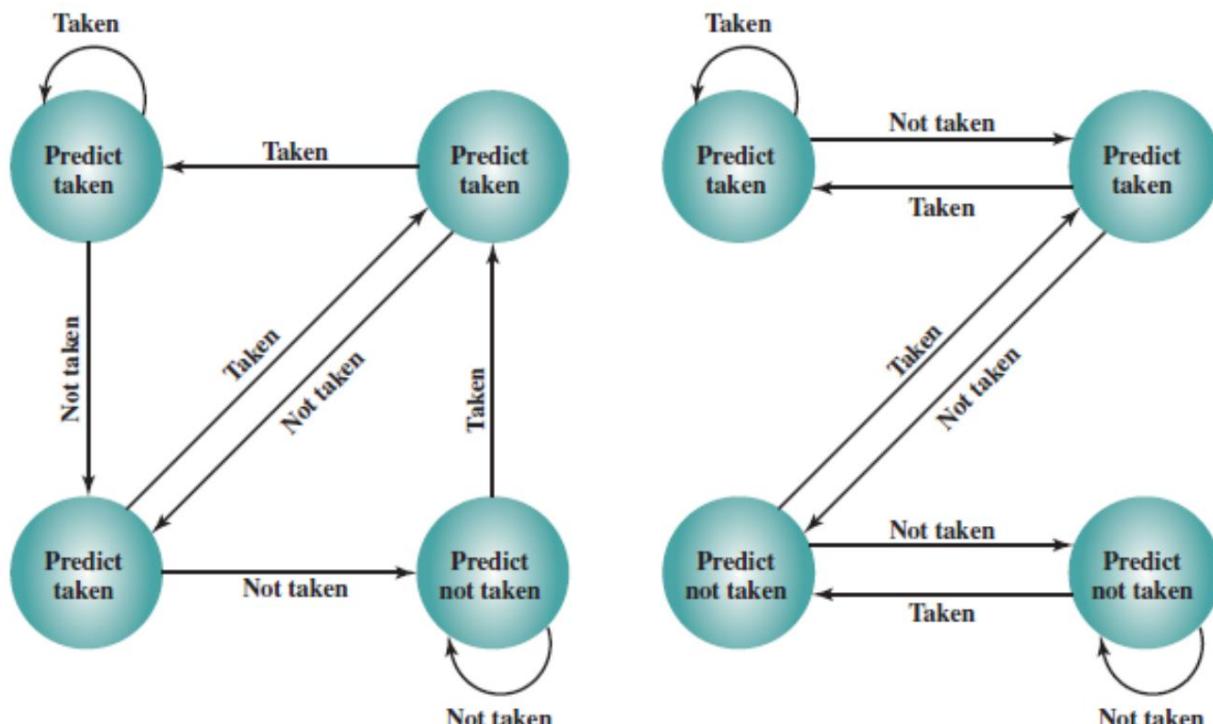
ÿ Exekuzioak ez-jauzi bat dakarren argibideak: (1-pq)n

ÿ  $T_{\text{program}} = T_{\text{inst\_salto}} + T_{\text{inst\_nosalto}} = \{pq*nkt\} + \{(1-pq)*(k+n-1)t\}$

- 12.13 Demagun 12.28 irudiko egoera-diagramak.

1. Deskribatu bakoitzaren portaera.

2. Konparatu hauek 12.4 ataleko adar-iragarpen-egoeraren diagramarekin. Eztabaidatu adar-iragarpenaren hiru ikuspegietako bakoitzaren meritu erlatiboak.



**Figure 14.28** Two Branch Prediction State Diagrams

- Predict taken: BAI saltoaren iragarprena.

- A diagrama:

ÿ Baiezko iragарpenetik negatiborako aldaketa:

ÿ Saltoen iragарpenetik hasi BAI ÿ Bi "EZ" jarraian iragarprena EZ aldatzeko

ÿ Iragarpen negatibotik baiezko aldatzea:

ÿ Jauzi iragарpenetik abiatuta EZ ÿ "BAI" bat iragarprena BAI aldatzeko

- B diagrama:

ÿ Baiezko iragарpenetik negatiborako aldaketa:

ÿ Saltoen iragарpenetik hasi BAI ÿ Bi "EZ" jarraian iragarprena EZ aldatzeko

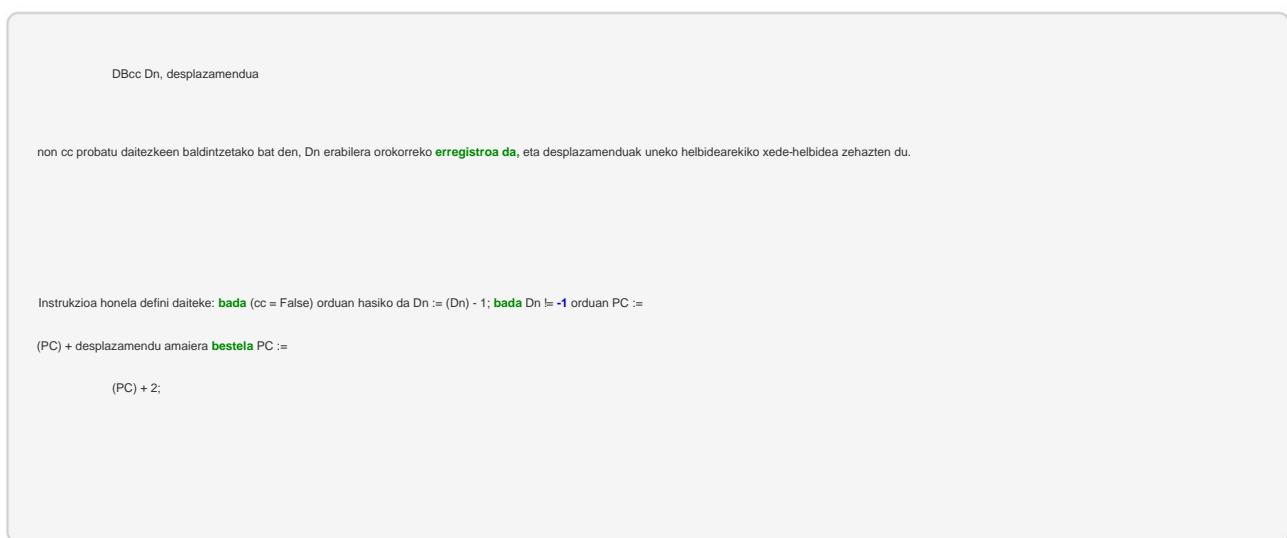
ÿ Iragarpen negatibotik baiezko aldatzea:

ÿ Jauzi iragарpenetik hasita EZ ÿ "BAI" bat aurreikuspena BAI aldatzeko, aurretik egin badu.  
bi "JAUZTI EZ" jarraian egon dira

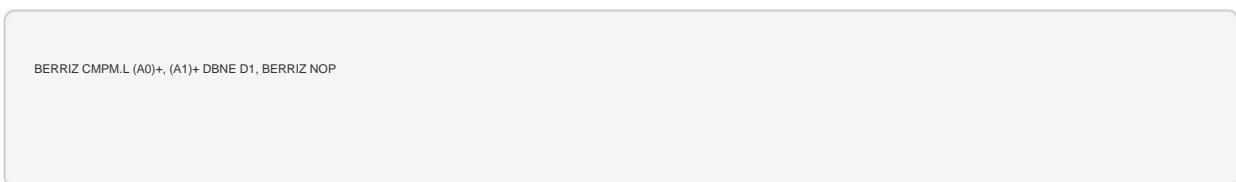
ÿ Jauzi iragарpenetik hasita EZ ÿ Bi "BAI" iragarprena BAI aldatzeko gehiago egon bada  
bi "JAUZTI EZ" jarraian

- 12.14 Motorola 680x0 makinek Dekretu eta Adarraren araberako instrukzioa dakin.

Baldintza, forma hau duena:



ÿ Instrukzioa exekutatzen denean, baldintza probatzen da, begiztaren amaiera-baldintza betetzen den zehazteko. Hala bada, ez da eragiketarik egiten eta exekuzioa sekuentziako hurrengo instrukzioan jarraitzen du. Baldintza faltsua bada, zehaztutako datu-erregistroa gutxitzen da eta zero baino txikiagoa den egiaztatzen da. Zero baino txikiagoa bada, begizta amaitzen da eta exekuzioa sekuentziako hurrengo instrukzioan jarraitzen du. Bestela, programa zehaztutako kokapenera adarkatzen da. Orain kontuan hartu hurrengo mutaria-hizkuntzako programa zati hau:



ÿ A0 eta A1 bidez zuzendutako bi kate konparatzen dira berdintasunerako; kate-erakusleak gehitzen dira erreferentzia bakoitzarekin. D1 hasieran konparatu beharreko hitz luzeen kopurua (4 byte) dauka.

- Erregistroen hasierako edukiak A0 = \$00004000, A1 = \$00005000 eta D1 = \$000000FF dira (\$-k idazkera hamaseitarra adierazten du). \$4000 eta \$6000 arteko memoria \$AAAA hitzekin kargatuta dago. Aurreko programa exekutatzen bada, zehaztu zenbat aldiz exekutatzen den DBNE begizta eta hiru erregistroen edukia NOP instrukziora iristen denean.

2. Errepikatu (a), baina orain suposatu \$4000 eta \$4FEE arteko memoria \$0000rekin kargatuta dagoela eta \$5000 eta \$6000 artean \$AAArekin kargatuta dagoela.

ÿ Garapena:

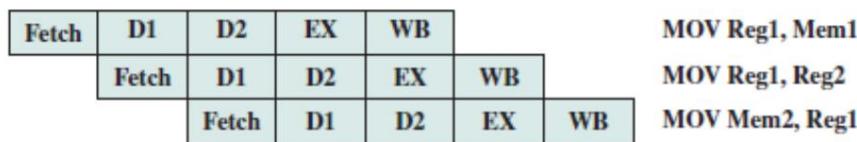
ÿ DBcc instrukzioa begiztak kontrolatzeko erabiltzen da iterazio bakoitza amaitutakoan. cc baldintzak DBcc instrukzioaren aurreko azken eragiketari egiten dio erreferentzia, kasu honetan CMPM.L. Baldintza egia bada ÿ Amaitu begizta eta jarraitu programaren sekuentzia. Baldintza faltsua bada, iterazio-kontagailua gutxitzen du eta begiztaren hasierara jauzi egiten du. L motako hitzak (handia) 4 byteko. D1=0xFF. D1-1=0xFFFFFFFF. A0 erakuslea kateari ÿ (A0) zeharkakotasuna ÿ (A0)+ erakuslea hitz bat handitzen du exekuzio bakoitzean.

1. Bi erakusleek \$AAAA duen edukia memorira seinalatzen dute, beraz, konparaketak BERDINA izango du. DBcc sententziaren baldintza NE da, ez berdina, beraz GEZURRA da eta begizta exekutatzen da. 0xFF+1 aldiz exekutatzen da D1=-1 kontagailura iritsi arte.

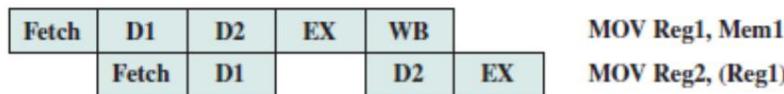
Azken helbidea ren erakuslea A0  
 $0x4000 + 0xFF \text{words} + 1 \text{word} = 0x4000 + 4x(0xFF) \text{bytes} + 4 \text{bytes}$  ÿ2 desplazatu 0xFF bi bit  $x(0xFF)$ -ren balioakidea da ezkerrera =  $0x3FC$  ÿ  $0x4000 + 0x3FC + 4 = 0x4400$ . A1 erakuslea ÿ  $0x5000 + 0x3FC + 4 = 0x5400$ .

2. Konparazio guziek zero ez den beste emaitzak lortzen dituzte ý NE ý beraz EGIA ý Iterazio bakarra exekutatzen da.  
D1=0xFF-1=0xFE; A0=A0+1hitza=0x4004 eta A1=0x5004

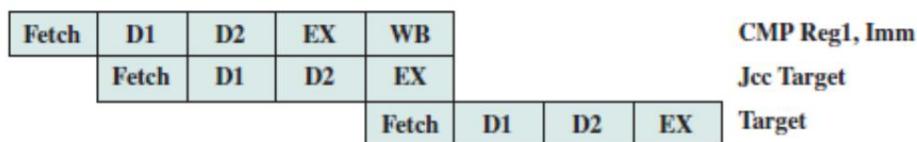
- 12.15 Birmarraztu 12.19c (14.21c) irudiak, baldintzapeko adarra ez dela hartzen suposatuzz



(a) No data load delay in the pipeline



### (b) Pointer load delay



(c) Branch instruction timing

**Figure 14.21** 80486 Instruction Pipeline Examples

ÿ Garapena:

ÿ 80486: 32 bit

✓ Instrukzio-hodietako etapak: FE-D1-D2-Ex-WB.

✓ Eskuratu: instrukzioen harrapaketa

ŷ D1: OpCode deskodetzeko eta helbideratzeko modua

ŷ D2: Operandoaren Helbide Eraginkorra kalkulatzeko eraiketak

© ADIB.: ALU eraqiketak eta eraqiqaietarako sarbidea

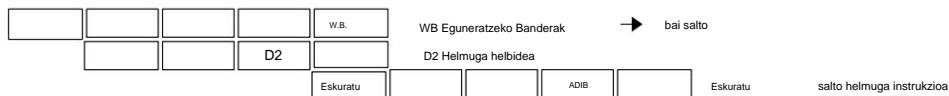
✓ WB: EFLAGS, emaitzak Req eta Mem (Cache eta MP)

ŷ 12.19 irudia b) 1. ins. EXn eragaiaren memoriatik irakurtzen du eta D2ko 2.ak memorian sartzen du eragaiaren helbidea erakusleto irakurtzeko

ŷ 12.19 Irudia c) CMP instrukzioak erreg-markak eguneratzen ditu. D2ko Jcc instrukzioak dagoeneko badu jauzi norabidea EX-n Programa-kontagailua eguneratzen duen arren. 3. inst D2 ondoren Jcc orain harrapatu daiteke.

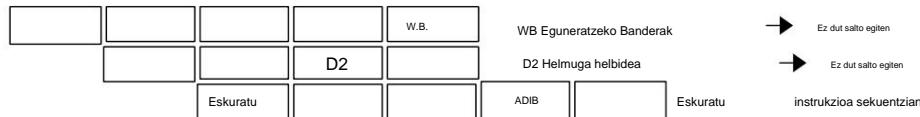
ŷ Bai salto egiten dut:

ŷ CPUak 3. instrukzioa (Fetch) harrapatzea egiten du harrapatu eta berehala bigarrenarena baina esandako harrapaketa okerra da hurrengoa segidan harrapatu baitu 2.a eta ez xede instrukzioa. Helmugako instrukzioa harrapatzea noiz egin behar da argibide hori non dagoen helbidea ezagutzen da.



ŷ Ez dut salto egiten:

ŷ CPUak 3 instrukzioak sekuentzian jasotzen ditu eta ez du akatsik egiten hirugarrenean, ez baitago. jauzi.



- 12.16 14.5 taulak [MACD84]-ren estatistikak laburbiltzen ditu hainbat klasetako adarren portaerari buruz. aplikazioak. 1 motako adarraren portaera izan ezik, ez dago alde nabarmenik aplikazio klaseak. Zehaztu sukursalaren xede-helbidera doan adar guztien zatia ingurune zientifikoak. Errepikatu merkataritza-inguruneetarako eta sistema-inguruneetarako.

**Table 14.5 Branch Behavior in Sample Applications**

<b>Occurrence of branch classes:</b>			
Type 1: Branch	72.5%		
Type 2: Loop control	9.8%		
Type 3: Procedure call, return	17.7%		
<b>Type 1 branch: where it goes</b>			
Unconditional—100% go to target	20%	40%	35%
Conditional—went to target	43.2%	24.3%	32.5%
Conditional—did not go to target (inline)	36.8%	35.7%	32.5%
<b>Type 2 branch (all environments)</b>			
That go to target	91%		
That go inline	9%		
<b>Type 3 branch</b>			
100% go to target			

ŷ Garapena:

ŷ mota1=72,5 ; mota2=9,8; mota3=17,7

ŷ 1 mota: 1 motaren barruan 3 kasu daude (1/3 jauzi baldintzarik gabe, 1/3 jauzi baldintzapean, 1/3 ez du jauzi)

ŷ 2 mota: % 91k jauzi egiten du, % 9k ez dute jauzi

3 mota: salto guztiak

þ Helmugara salto= type1x[(0.2+0.4+0.35)x100/100+(43.2+24.3+32.5)x1/3]+ type2x0.91+  
mota 3x100/100 =

ÿ Aplikazioen arabera etiketara jauzi egiten du

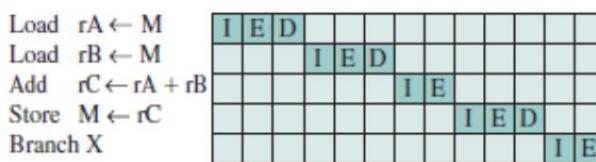
ÿ zientifikoa=mota1x[(0,2 )x100/100+(43,2)x1/3]+ mota2x0,91+mota3x100/100=0,724ÿ % 72 aplikazio zientifiko baten jauziak helmugara dira.

$$\text{ŷ komertziala} = \text{mota1x}[0,4\text{)}x100/100 + (24,3)x1/3] + \text{mota2x}0,91 + \text{mota3x}100/100 = 0,732$$

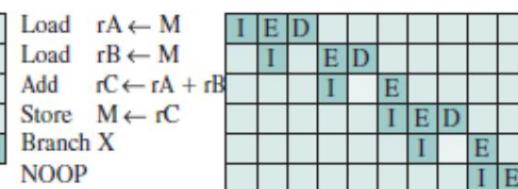
$$\text{y sistema} = \text{mota1x}[0,35 \times 100 / 100 + (32,5) \times 1 / 3] + \text{mota2} \times 0,91 + \text{mota3} \times 100 / 100 = 0,756$$

10.14. 13. kapitulua: Instrukzio-multzoaren ordenagailua murritzten du (15. kapitulua 9th Ed-n)

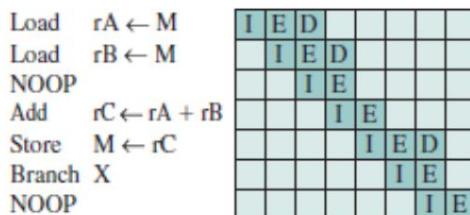
- 13.3 Programa jakin baterako exekuzio-denbora zehaztu nahi dugu 13.5 atalean aztertutako kanalizazio-eskema desberdinak erabiliz. Izan bedi  $N$  = exekutatutako instrukzioen kopurua,  $J$  = jauziko jarraibideen kopurua,  $D$  = memoriako sarbide kopurua. RISC arkitektura baterako eskema sekuentzial sinplerako (13.6a irudia), exekuzio denbora  $2N+D$  etapakoa da. Eratorri bi faseko, hiru faseko eta lau faseko kanalizaziorako formulak.



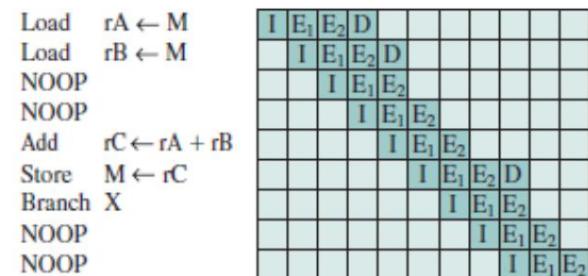
### (a) Sequential execution



### (b) Two-stage pipelined timing



### (c) Three-stage pipelined timing



(d) Four-stage pipelined timing

Ü. Coronene:

Ý KANAL SEGMENTUA: I Ý instrukzioen harrapaketa. Eý ALU eragiketak Reg. u  
belarria erreginkorra lotzera. Dú Mem. transferentzia, ú Reg.

ŷ Instrukzio bakoitzak gutxienez bi etapa ditu: E eta I. Hala ere, instrukzio guztiekin ez dute D etapa (erreg eta mem  
artearen kerretutik eta garde bakoitzak).

- Irudiaren atala  $\ddot{y}T = Nx(te+ti) + tdxD$ ; baldin  $ti=te=td=t$   $\ddot{y} T = [2N+D]t$
  - Irudiaren b. atala  $\ddot{y}$  Kanal segmentatua  $\ddot{y} k=2$   $\ddot{y}$ . lauzi-argibiderik gabe  $\ddot{y} Tk n=[k+(n-1)]t$   $\ddot{y} T2 n=[2+(N-1)]t$

to. Etapa bakoitzean memoria sarbide bakarra da posible

b. I etapa bat da E eta D etapa bakarra osatzen dute

c. E eta I gainiartzen dira ñ N etapa Ell

- d. D ez da gainjartzen ý D etapak

eta. Branch instrukzioaren E fasean jauzi helbidea kalkulatzen da, beraz, I fasea Helmugako instrukzioa ezin da aipatutako E fasearekin bat eterri. Instrukzio batekin ebazten da NOOP ez den eragiketa.

F. Jauziek NOOP bat eragiten dute ý atzerapen-urrats bat gehitzeko

g.  $T = (N + D + J)t$

3. Irudiaren sekzioa c ý k=3 etapa

to. Bi memoria sarbide posible dira etapa batean.

b. 2. kargatzeko instrukzioak datuak D etapako erregistroan kargatzen ditu, beraz, ezin du batura instrukzioaren exekuzioarekin bat datozi.

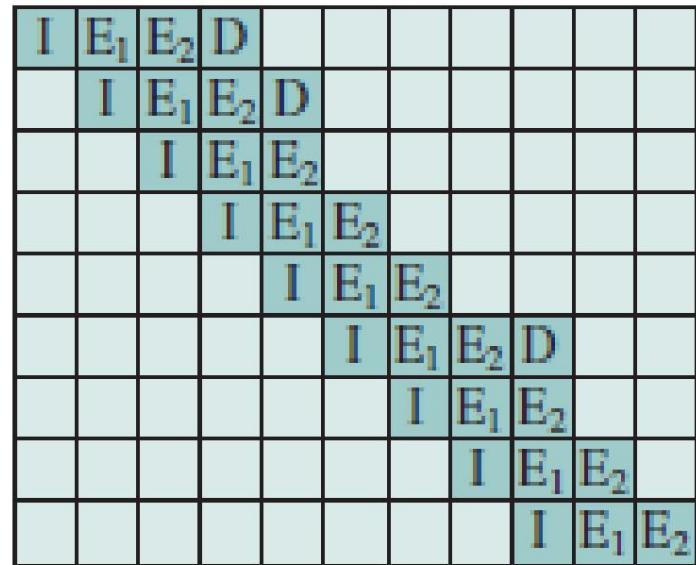
c. D, E eta I gainjartzen dira datuen mendekotasunik ez badago

d. Mendekotasunak badaude D ez da gainjartzen, beraz, D instrukzioen zati bat dago hori gehitu behar da.

- eta.  $T = (N + \alpha * D + J)t$  y J noops  
4. Irudiaren atala d y k=4 etapa  
to. E banatzetan dugu E1 (RPG irakurketa) eta E2 (AI U eta RPG idazketa)

- b. D-ren gainjartzeak datuen mendekotasunarekin atzerapen bat eta J-k beste bi sartzen ditu irudia.
- c.  $T = (N + \alpha * D + 2J)t$
- 13.4 Berrantolatu 13.6d irudiko kode-sekuentzia NOOP kopurua murrizteko. ariketa irudia  
13.3 d).

Load     $rA \leftarrow M$   
 Load     $rB \leftarrow M$   
 NOOP  
 NOOP  
 Add     $rC \leftarrow rA + rB$   
 Store     $M \leftarrow rC$   
 Branch    X  
 NOOP  
 NOOP



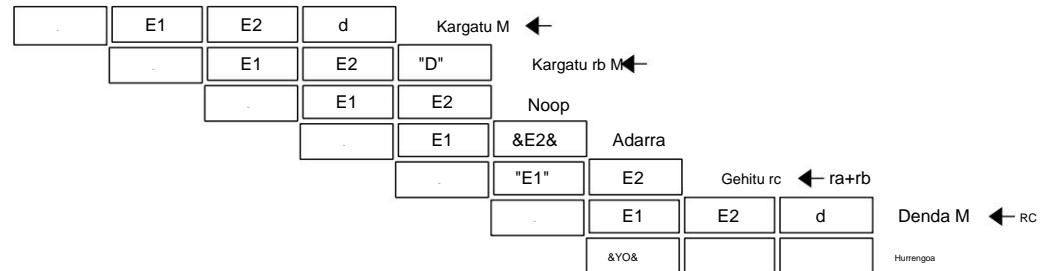
#### (d) Four-stage pipelined timing

ÿ Branch instrukzioa 2. NOOP-aren posizioan exekutatzen dugu

ÿ Saltoaren aurreko bi argibideak (Gehitu eta Gorde) saltoaren ondorengo 2 NOOPen ordez Adarra.

```
Karga ra<-M
Kargatu rb<-M
Noop
Adarra
Gehitu rc<-ra+rb
Gorde M<-rc
Hurrengoa
```

ÿ Gehitu eta Gorde exekutatzen diren bitartean, X helbidea kalkulatzen da



ÿ Mendekotasunak: rb "D" fasean kargatuta dagoenean, orain rb irakur dezakezu "E1"-n

ŷ Mendekotasunak: X &E2&n kalkulatu ondoren, Hurrengo instrukzioa atzman daiteke  
&I& bitartean

- 13.5 Demagun kode zati hau goi-mailako hizkuntza batean:

I-n **1 ...100** begizta S ŷ S +  
Q(i).VAL amaierako begizta;

- Demagun Q 32 byteko erregistroen array bat dela eta VAL eremua erregistro bakoitzaren lehen 4 byteetan dagoela. Erabiliz x86 kodea, programa zati hau honela konpila dezakegu:

```
MOV ECX,1 ;erabil ezazu ECX erregistroa I eusteko
LP: IMUL EAX, ECX, 32 ; lortu konpentsazioa EAX-en
MOV EBX, Q[EAX] ;kargatu VAL eremua
GEHITU S, EBX; gehitu S-ra
INC ECX; gehikuntza I
CMP ECX, 101 : alderatu 101ekin
JNE LP ;begiztatu I = 100 arte
```

- Programa honek IMUL instrukzioa erabiltzen du, bigarren eragia hirugarren eragiaiko berehalako balioarekin biderkatzen du eta emaitza lehenengo eragiaian jartzen du (ikus 10.13 problema). RISC defendatzale batek frogatu nahiko luke konpilatzale burutsu batek alferrikako instrukzio konplexuak ezaba ditzakeela, hala nola IMUL. Eman erakustaldia goiko x86 programa berridatziz IMUL instrukzioa erabili gabe.

to. Array Q: 32 byteko bakoitzeko 100 erregistro (egitura).

b. VAL eremua: erregistroaren lehen 4 byteak.

```
typedef struct {int VAL;....} Datuak; Datu
Q[100];
```

c. Q(i).VAL : Q(i) erregistro bakoitzaren VAL eremua

ŷ **## Begiztak 100 erregistroen VAL eremuak batu ditu**

MOV ECX,1 ;erabil ezazu **erregistroa** ECX I #Array erregistro-indizea edukitzeko  
LP: IMUL EAX, ECX, **32** ;**lortu** desplazamendua EAX-n #EAX: Q matrizeko ECX indizeak adierazitako  
erregistroko VAL eremuaren helbide erlatiboa

#32 byte bakoitzean MOV EBX

**erregistro** bat, Q[EAX] ;kargatu VAL eremua #Q mihiztagailuan bytetan egituratuta dago.

100recordsx32bytes/record=3200records ADD S, EBX ;S to gehitu

#Val eremuko **4** byteen batura ;increment I #next record INC ECX CMP ECX, **101** ; **101** #azken+1  
erregistroarekin alderatu ?

JNE LP ;loop I = **100** arte #Hurrengo elkarrekintza azkena ez bada

d. 2x biderkatzeak x biteko desplazamendu bat da ezkerrera

ŷ Biderkatu 32z ŷ x25 ŷ mugitu 5 bit ezkerrera: shl \$5,%ecx

- 13.6 Demagun begizta hau:

```

S := 0;
K-rako := 1etik 100era do
  S := S - K;

```

- Honen itzulpen soil bat muntaketa-lengoaia generiko batera honelako itxura izango luke:

```

LD R1, 0           ;mantendu S-ren balioa R1n
LD R2,1 ;K-ren balioa R2n mantendu
LP SUB R1, R1, R2 ;S := S - K
BEQ R2, 100, EXIT ;eginda K = 100 bada;bestela      #BranchEqual
GEHITU R2, R2, 1   K gehitzea
JMP LP; itzuli begizta hasierara

```

- RISC makina batentzako konpiladore batek atzerapen zirrikituak sartuko ditu kode honetan, prozesadoreak erabil dezan atzeratutako adar mekanismoa. JMP instrukzioa erraza da aurre egiteko, instrukzio hau bai beti SUB instrukzioa jarraituz; Hori dela eta, SUB instrukzioaren kopia bat jar dezakegu JMPren ondoren atzerapen-zirrikitua. BEQ-k zaitasun bat aurkezten du. Ezin dugu kodea dagoen bezala utzi, izan ere ADD instrukzioa exekutatu egingo litzateke orduan gehiegitan. Beraz, NOP instrukzioa behar da. Erakutsi ondoriozko kodea.

- Garapena:

to. Adar atzeratua

	1	2	3	4	5	6	7
<b>100 LOAD X, rA</b>	I	E	D				
<b>101 ADD 1, rA</b>		I	E				
<b>102 JUMP 105</b>			I	E			
<b>103 ADD rA, rB</b>				I	E		
<b>105 STORE rA, Z</b>					I	E	D

(a) Traditional pipeline

<b>100 LOAD X, rA</b>	I	E	D				
<b>101 ADD 1, rA</b>		I	E				
<b>102 JUMP 106</b>			I	E			
<b>103 NOOP</b>				I	E		
<b>106 STORE rA, Z</b>					I	E	D

(b) RISC pipeline with inserted NOOP

<b>100 LOAD X, Ar</b>	I	E	D				
<b>101 JUMP 105</b>		I	E				
<b>102 ADD 1, rA</b>			I	E			
<b>105 STORE rA, Z</b>				I	E	D	

(c) Reversed instructions

**Figure 15.7** Use of the Delayed Branch

ŷ Lehenengo grafikoa jauzi normala da eta beste biak atzeratuta daude. Azken grafikoak a adierazten du instrukzioa gutxiago

- b. Programak atzeratutako bi jauzi jarraibide ditu: BEQ eta JMP
- c. JMP LP

ŷ Baldintzarik gabeko jauzia. Jatorrizko programan jauzia **SUB R1, R1, R2** ondoren exekutatuko da . Irtenbidea:

JMP LP	; itzuli begizta hasierara
GEHITU R2, R2, <b>1</b>	<b>;bestela</b> gehitu K

ŷ Modu honetan , **ADD R2, R2, 1-en** ondoren jauzi egiten da

- d. BEQ R2, 100, IRLEN

ŷ Baldintzazko jauzia

ŷ Bere horretan uzten badugu, jauzia ADD R2, R2, 1 ondoren izango da. Irtenbidea:

BEQ R2, <b>100</b> , EXIT ;eginda K = <b>100 bada</b>	#BranchEqual
LP SUB R1, R1, R2 ;S := S - K	

ŷ Orain baldintzapeko jauzia SUB kenketaren ondoren egingo da

- eta. 1. irtenbidea:

LD R1, <b>0 :mantendu</b> S-ren balioa R1n	
LD R2,1 ;K-ren balioa R2n mantendu	
LP BEQ R2, <b>100</b> , EXIT ;eginda K = <b>100 bada</b>	#BranchEqual
AZPIA R1, R1, R2 ;S := S - K	
JMP LP; itzuli begizta hasierara	
GEHITU R2, R2, <b>1</b> ; <b>bestela</b> gehitu K	

ŷ R2=100 bada, SUB ere exekutatzen den akatsa du, R1 eta R2ren azken balioa aldatuz.

- F. Azken irtenbidea:

LD R1, <b>0</b> ;mantendu S-ren balioa R1n	
LD R2,1 ;K-ren balioa R2n mantendu	
LP BEQ R2, <b>100</b> , EXIT ;eginda K = <b>100 bada</b>	#BranchEqual
EZ	
GEHITU R2, R2, <b>1</b> ; <b>bestela</b> gehitu K	
JMP LP ; itzuli begizta hasierara	
AZPIA R1, R1, R2 ;S := S - K	

- 13.7 RISC makina batek erregistro sinbolikoen mapaketa bat egin dezake benetako erregistroekin eta kanalizazioaren eraginkortasunerako argibideak berrantolatzea. Galdera interesgarri bat sortzen da ordenaren inguruan bi eragiketa hauek egin beharko lirateke. Kontuan izan honako programa zati hau:

LD SR1,A	;kargatu A <b>erregistro</b> sinbolikoan <b>1</b>
LD SR2, B	;kargatu B <b>erregistro</b> sinbolikoan <b>2</b>

```
GEHITU SR3, SR1, SR2; gehitu SR1 eta SR2-ren edukia eta gorde SR3-n
LD SR4, C
LD SR5,D
GEHITU SR6, SR4, SR5
```

1. Lehenik eta behin, egin erregistro-mapaketa eta, ondoren, posible den instrukzioen berrantolaketa. Zenbat makina erregistroak erabiltzen dira? Hobekuntzarik egon al da hoditeria?
  2. Jatorrizko programatik hasita, orain instrukzioen berrantolaketa egin eta gero mapa posible guztiak. Nola makina-erregistro asko erabiltzen dira? Hobekuntzarik egon al da hoditeria?
- Garapena: .
  - 13.9 Kasu askotan, MIPS instrukzio-multzoaren zati gisa zerrendatzen ez diren makina-argibide arruntak MIPS instrukzio bakar batekin sintetizatu daiteke. Erakutsi hau honako hauetarako:
    1. Erregistratu-erregistratzeko mugimendua
    2. Gehitu, gutxitu
    3. Osagarria

4. Ukatu zeure buruari
  - 5. Garbi
- 13.11 SPARC-ek CISC makinetan normalean aurkitzen diren hainbat argibide falta ditu. Horietako batzuk erraz simulatzen dira R0 erregistroa, beti 0-n ezarrita dagoena, edo eragiketa konstante bat erabiliz. Simulatutako instrukzio hauei pseudoinstrukzio deitzen zaie eta SPARC konpilatzaileak ezagutzen ditu. Erakutsi ondorengo pseudo-instrukzio hauek nola simulatu, bakoitza SPARC instrukzio bakar batekin. Horietan guzietan, src eta dst erregistroak aipatzen dira. (Aholkua: R0-ra dagoen denda batek ez du eragini.)
1. MOV src, dst
  2. KONPARATU src1, src2
  3. PROBA src1
  4. EZ dst
  5. NEG dst
  6. INC dst
  7. ABENDUA dst
  8. CLR dst
  9. EZ
- 13.12 Demagun kode zati hau:

K > **10** bada

L := K + **1**

**bestela**

L := K - 1;

- Adierazpen honen itzulpen soil bat SPARC mihiztagailura honako forma hau har liteke:

```
sethi %hi(K), %r8          ;kargatu ordena handiko 22 bit kokapenaren helbidea
                                ;K r8 erregistroan
ld [%r8 + %lo(K)], %r8 ;kargatu K kokapenaren edukia r8 cmp %r8, 10
                                ; alderatu r8-ren edukia 10arekin
```

```
ble L1 ;adarra (r8) <= 10 bada
nop sethi
%hi(K), %r9 ld [%r9 +
%lo(K)], %r9 ;kargatu K kokapenaren edukia r9-n inc %r9 ;gehitu 1 (r9)

sethi %hi(L), %r10 st %r9,
[%r10 + %lo(L)] ;gorde (r9) L b L2 kokapenean

ezetz
L1: sethi %hi(K), %r11 ld
[%r11 + %lo(K)], %r12 ;kargatu K kokapenaren edukia r12 dec %r12 sethi %hi(L), %r13 st
%r12, [ %r13 +
;kendu 1 (r12)
%lo(L)] ;gorde (r12) L
kokapenean
L2:
```

- Kodeak nop bat dauka adar-agindu bakoitzaren ondoren, adarretako funtzionamendu atzeratua ahalbidetzen.

1. RISC makinekin zerikusirik ez duten konpilatzairen optimizazio estandarrak, oro har, eraginkorrik dira aurreko kodean bi transformazio egin ahal izateko. Kontuan izan bi karga ez direla beharrezkoak eta bi dendak bateratu daitezkeela denda kodean beste leku batera eramatzen badira. Erakutsi programa bi aldaketa hauetan egin ondoren.
2. Orain SPARC-en berezko optimizazio batzuk egin daitezke. blearen ondorengo nop-a ordezkatu daiteke beste instrukzio bat atzerapen-zirriki horretara eramanez eta ble instrukzioan anula-bit ezarriz (ble,a L1 gisa adierazita). Erakutsi programa aldaketa honen ondoren.
3. Orain beharrezkoak ez diren bi argibide daude. Kendu hauetan eta erakutsi ondoriozko programa

## IV Autoebaluazioaren Teoria

## 11. kapitula. Teoria: Galdetegia

### 11.1. von Neumann arkitektura

1. Zein funtio du muntaketa programak?
2. Zein hizkunza mota den makina-lengoaia
3. Zein urtetan garatu zen IAS makina?
4. Zein da IAS makinaren instrukzio-formatua
5. IAS makinaren bi argibide adierazten ditu bi hizkunza ezberdinetan.
6. Zein da emuladorearen, IAS makinaren, muntaia-lengoaia erakusteko programa?  
IASSim.
7. Zeintzuk dira emulatzaile demo programaren makina-lengoaiaren lehen bi argibideak?  
IASSim.
8. Zertan datza ISA arkitektura?
9. Erabili abstrakzio hitza irakasgaiaren zenbait kontzeptutan
10. Bi mikroordena adierazten ditu
11. Zein unitate sortzen dituen mikroordenak
12. Zein elementutara zuzentzen diren mikroordenak
13. Zer esan nahi du 1MB
14. Zeintzuk dira IAS UCk erabiltzen dituen erregistroak?
15. Zeintzuk dira IAS ALUk erabiltzen dituen erregistroak?
16. Zeintzuk dira irakaskuntza-zikloaren faseak
17. Zein unitate edo unitatek ezartzen dute instrukzio-zikloa.
18. Zein memoria kategoriatik irakurtzen ditu CPUak programak?
19. Zer da prozesu bat
20. Definitu CPU arkitektura gutxienez termino hauek erabiliz:
  - ÿ instrukzio-zikloa
  - ÿ makinen instrukzioa
  - ÿ biltegiratzea
  - ÿ interpretazioa
  - ÿ datu-bidea
  - ÿ erregistroa
  - ÿ mikroaginduak

# **V Praktika-gidoiak: x86 Muntatzaileen Programazioa**

## 12. kapitula. AT&T x86-32 Mihiztadura Lengoaiaren Programazioaren Sarrera

### 12.1. Sarrera

#### 12.1.1. Helburuak

- Maila baxuko programaziorako sarrera C eta AT&T mihiztatzaile lengoaiak erabiliz Intel 32 biteko x86 arkitektura.
- Maila baxuko softwarea garatzeko tresnak erabiltzea, hala nola toolchain (konpilatzailea, muntatzailea, estekatzailea) eta GDB araztailea, GNU oinarritik askea.
- Lan-estazio baten garapena garapen-tresnak ingurune batean instalatuz GNU/linux/x86 ordenagailu pertsonal batean.
- Ordenagailuaren funtzionamendua maila baxuko programatzaile baten ikuspuntutik ulertzea. maila.
- Maila handiko hizkuntza baten propietateak maila baxuko hizkuntza batekin erlazionatu.
- Mihiztadura-lengoaia ordenagailuen arkitektura eta funtzionamendua azterzeko tresna gisa erabil daiteke. Ez da irakasgai honen helburu nagusia maila baxuko programazio lengoaietan edo algoritmoen garapenean aditua izatea, oso oinarrizko maila den arren.

#### 12.1.2. Baldintzak

##### Teoriak

- ISA arkitektura baten oinarrizko ezagutzak: Von Neumann eredu-arkitektura (CPU mikroarkitektura, Main Memory arkitektura, instrukzio-zikloa), datuen irudikapena eta eragiketa aritmetikoak, instrukzio-formatua eta oinarrizko programazioa mutuaia-lengoaia eta makina-lengoaia IAS makinarekin John von Neumann. , AT&T x86 mutuaia-lengoaiaren sintaxia eta Intel-en x86-64 eta x86-32 prozesadorearen oinarrizko arkitektura.

##### Praktikoa

- GNU/linux(AMD64)/x86\_64(Intel edo AMD) "Development Platform" tresna egokiekin konfiguratuta eduki.
- C lengoian ezinbesteko programazioa eta "Transfer Language arteko RTL Records, GNU/linux ingurunearren oinarrizko kudeaketa eta edizio tresna bat.
- Gaiaren gunean erregistratuta egotea myaulary zerbitzari batean.
- Programa baten exekuzioa simulatu izana mutuaia-lengoaiaren eta makina-lengoaiaren. Adibidez John von Neumann-en IAS makinan IASSim Web emuladorea erabiliz

ÿ

Mihiztadura-lengoaiaren programa simple baten sintaxia eta egitura aztertu izana AT&T x86, programaren exekuzioa pauso pauso konpilatu, exekutatu eta aztertu zuen GDB araztailea erabiliz Eranskinako informazioa erabiliz:  
Programazioa hasieratik eta tutorial txikia Eranskina: ASM abiaraztea eta Eranskina: Hasteko C

### 12.2. IRAKUR NAIZ

- Praktikaren gidoia eta Programazioaren 1. eta 2. kapituluak Ground-Up Book-etik irakurtzea.

- Oharrak eta Testu liburua

ŷ

Memoria denbora errealean egiten da praktikan zehar eta azterketa egunean erabiliko den inprimatutako dokumentazio bakarra da, beraz, ondo dokumentatuta egon behar da. Txosten MyAulario/TASKS bidez txostenia osatu eta egun gutxira entregatzen da Eranskinean zehaztutako argibideak jarraituz: Txostenaren edukia eta formatua eta MyAulario/TASKS-en adierazitako egunean.

- Ireki testu-dokumentu bat ordenagailuko mahaigainean, memoria izango dena non praktika-saio osoan zehar lortutako informazio guztia gehituko du
- Ebaluazioa: ebaluazio sistema
- Programazioa: metodologia

### 12.3. Gaiak

- Aukerako “Praktikaren Autoebaluazioa”: Praktikak: Galdetegia

### 12.4. Lanpostua

- Idatzi Garapen Plataformaren ezaugarriak Memoriaren Dokumentuan.

### 12.5. sum1toN.c programazioa

#### 12.5.1. Algoritmoa

- Garatu C lengoian i386 arkitekturarekin batuketak egiten dituen programabatzasieneko deskribapen programazio metodoa erabiliz pseudokode lengoian eta organigraman.

#### 12.5.2. Editatu iturburu-modulua: sum1toN.c

- Editatu programa myaulary-tik "sum1toN.c" iturburu-modulua deskargatzetik eta iruzkinak gehituz egokia.
- Informazio osagarria duen goiburua:

```
/*
Programa: sum1toN.c
Deskribapena: 1,2,3,...N serieen batura egiten du Von
Neumann IAS makinaren sum1toN.ias-en baliokide den C hizkuntzako programa da.

eta sum1toN.s programaren baliokidea AT&T mihiztadura hizkuntzan
Hizkuntza: C99
Deskribapena: Lehenengo 5 zenbaki naturalen batura
Sarrera: Aldagai batean definitua
Irteera: Irteera periferikorik ez<<<s
Eraikitzea: gcc -m32 -o sum1toN sum1toN.c
OS: GNU/linux 5.4.0-128-generic ubuntu 20.04 x86-64 /
Liburutegia: usr/lib/x86_64-linux-gnu/libc.so
PCa: ThinkPad L560 produktua: 20F1S0H400 seriea: MP15YSW7
```

CPU: Intel(R) Core(TM) i5-6300U CPU @ 2.40GHz zabalera: 64 bit gcc bertsioa  
 Konpilatzailea: 9.4.0 GNU muntatzaile  
 Muntatzailea: bertsioa 2.34  
 Lokatzailea/Kargatzalea: GNU Id (GNU Binutils Ubunturako) 2.34 Gaia:  
 Data: Informatika Egitura 2022/09/25  
 Egilea: Candido Aramburu  
 \*/

- Programa:

```
// Modulu nagusia void
main (void) {
//Aldagai lokalen deklarazioa eta begizta parametroen hasieratzea int sum=0,n=5;

// Gehigarriak sortzen dituen eta gehiketa egin duen begizta while(n>0)
{
    //Begizta iteera baldintza gehigarria negatiboa denean
    batura+=n;
    n--; }} //Gehigarria eguneratzen

//Instrukzioak etenik edo hutsegiterik gabe exekutatu badira main()-ek zero balioa itzultzen dio sistema
eragileari.
```

#### 12.5.3. Konpilazioa

- Saio guztietan ohikoak diren bilketa-prozesuaren urratsak jarraitu.

ÿ gcc -m32 -g -o sum1toN sum1toN.c  
 ÿ zerrendatu fitxategiak: ls -l sum1toN\*. Zer adierazten du horietako bakoitzak?  
 ÿ gcc --save-temps -m32 -g -o sum1toN sum1toN.c  
 ÿ zerrendatu fitxategiak: ls -l sum1toN\*. Zer adierazten dute fitxategi berriek?

#### 12.5.4. Moduluaren azterketa

- Modulu desberdinak egiaztatzeko analisia:

ÿ sum1toN.c fitxategia  
 ÿ fitxategia sum1toN.i  
 ÿ sum1toN.s fitxategia  
 ÿ fitxategia sum1toN.o  
 ÿ sum1toN fitxategia

#### 12.5.5. Exekuzioa

- ./sum1toN : modulu bitar exekutagarria deia
- oihartzuna \$? : sum1toN programak linux sistema eragileari itzulitako balioaren bistaratzea.  
 ÿ Zero balioa programa inongo motarik gabe exekutatu izanaren adierazle gisa erabiltzen da atzerakada.

## 12.5.6. Depurazioa

sarrera

- Programaren exekuzioa urratsez urrats, instrukzioz instrukzioa, programaren exekuzioaren behe-mailako azterketa zehatza egiteko aukera ematen du, programa geldiarazi eta aldagaien balioa memoria nagusian, CPUaren egoera iraultzeko gai izanik. erregistroak, etab.
- GDB Debugger (GNU DeBugger) urratsez urrats exekuzioa eta memoria aztertzea ahalbidetzen du arazketarako berariazko komandoen erreperiorio baten bidez.

Sinboloen taula sortzea

- gcc -m32 -g -o sum1toN sum1toN.c
  - ÿ -g aukera: Sartu "Symbol Table" modulu bitar exekutagarrian.
- Komando-lerroan, erabili TAB TAB izenak osatzeko: gcc -m32 -g -o batuketa1TAB suTAB

gdb

- Ireak arazketa: gdb arazketa saioa hasteko.
  - ÿ Probintzia duen leihoa (gdb): araztearen komando-lerroa interpretatuko duena. GDB.

ireki leihoa berri bat

- (gdb) layout src edo Control-x Control-a
- nabigatu bi leihoen artean: Control-x edo y baina foku src eta foku cmd ere bai
- bideratzen lagundu

Erregistroa

Erregistroa bi leihoa erekita konfiguratu behar dira: komandoen leihoa eta iturburu-kodearen leihoa. Komando-leihoaarekin bakarrik egiten bada, leihoa gehiago erekitzen dituzunean saioa egiteari uzten dio.

- Gorde arazketa-saio osoa sum1toN\_gdb\_c.txt fitxategian
  - ÿ Sarrerak ÿ (gdb) aktibatu arrasto-komandoak,
  - ÿ Irteerak ÿ (gdb) eazarri erregistro-fitxategia sum1toN\_gdb\_c.txt
  - ÿ Aktibazioa ÿ eazarri saioa hasteko

Irteera sum1toN\_gdb\_c.txt-era kopiatzen.  
Arazte-irteera sum1toN\_gdb\_c.txt-era kopiatzen.

- Praktika saio berean araztagailuan irten eta sartzen zaren bakoitzean eta sum1toN\_gdb\_c.txt historia-fitxategi bera erabiltzen duzun bakoitzean, aurreko konfigurazioa egin behar duzu bi leihoa erekita.

Linux komandoak gdb komando lerrokatik

- shell ls -l sum1toN\_gdb\_c.txt
  - ÿ shell data

ÿ shell pwd

ÿ shell ls

ÿ shTAB : Erabili TAB TAB izenak osatzeko.

ÿ daTAB shell

## EGIAZTU arazketa-saioen historia

- Ireki kontsola bat eta laneko karpetan egiaztatu sum1toN\_gdb\_c.txt fitxategia badagoela eta aurrez exekutatutako gdb komandoak ditu.

ÿ Oso etsigaita gauza orduko lanaren ondoren ez dela gordeta konturatzea

## Leihoak

- Diseinua: **Cx a** ÿ lehenespenez bi leiho: iturburu-modulua eta komando-lerroa.
- Windows arakatzailea: **Cx edo**
- Arakatu komandoen historia
  - ÿ Komandoen historiako erregistroa: Aktibatu GDB komandoen leihoa. Nabigatu teklen bidez gora/behera gezia.

## Laguntzak

- help shell** edo **h shell**

## Kargatu moduluaren objektu exekutagarria

- Kargatu Sinboloen Taula duen objektu bitar modulua: **sum1toN** fitxategia
- iturburu-modulua Sinboloen Taulari lotutako sinboloekin: **informazio iturriak**

ÿ Kontuan izanak zituztenean sinboloen taularen existentzia berresten duela, eta hori ezinbestekoa da

## Urratsez urrats exekuzioa

- Egitarauaren sarreran eten-puntu: **break main**
- Exekuzioa: **exekutatu** lerroa exekutatu EZ duen eten-puntura arte ÿ Iturburu-kodea agertzen da.
- Hurrengo iturburu-lerroa: **hurrengo, n**
- Hurrengo 5 iturri-lerroak: **n 5, inprimatu batura, p sum**
- Hasi berriro hasieratik: **korrika** edo **hasi**
- Jarraitu hurrengo etengunera: **jarraitu, c**
- korrika, n, itzulera, RET, RET, p batura**

## Begizta

- nola irten ehunka edo milaka iterazioko begizta batetik hurrengo adierazpenera begiztatik kanpo?
- exekutatu, arte, RET,RET,RET...** begiztatik irten arte..., **p batura, c**

## DRAM memoria nagusiaren azterketa

- aldagaien edukiak eta haien helbideak memoria nagusian inprimatu
- inprimatu n, pn, p /tn, p /xn, ptype n, zer da n,p &n
- inprimatu ikurra : ikurra aldagaiaren izena da, ez helbidea.
- p \$eax
- p \$ebx
- p \$ecx
- informazio erregistroak

## Desmuntatu

- Desmuntatu: alderantzizko ingeniaritza. Bihurtu kode bitarra batzar-kode  
    ÿ zatitu diseinua  
    ÿ Hurrengo makinaren instrukzioa: ni, RET, RET, RET, RET, until, RET,... begiztatik irten arte  
        ÿ Makinaren argibideak exekutatzen ditu (ikus leihoa mutua kodearekin)

## Atera

- irten  
    ÿ Egiaztatu sum1toN\_gdb\_C.txt fitxategiaren edukia zuzena dela.

## 12.5.7. Oroigarria: Memoria dokumentua

- Gorde lana eta egin ahala komentatu.
- Kontsolan sum1toN\_gdb\_C.txt fitxategia irekitzen dugu, berearekin erabiltzen diren komando guztiak biltzen dituena. botata.
- Gorde sum1toN\_gdb\_C.txt-en edukia Memoria Dokumentuan iruzkinak gehitzu beharrezkoak.

## 12.5.8. Ariketa gehiagorekin jarraitzen dugu

### Jolastu iturri-moduluarekin

- Sorburuko Moduluaren aldaketak C hizkuntzan.  
    ÿ Aldatu datuak mota hauetako batekin:  
        ÿ char, labur, int, luze  
    ÿ Aldatu zenbakien formatua oinarri hauetako batekin:  
        ÿ hamartar, hamaseitarra, zortzikoa, bitarra ÿ aurizkiak 0x, 0, 0b ÿ 0x5, 05, 0b5  
    ÿ Konpilatu eta exekutatu. Adierazi memorian errorerik dagoen edo ez

### Jolastu araztearekin

- GDB  
    ÿ Aldatu aldagaien tamaina char eta iturburu-moduluko batura+=n adierazpena adierazpenera.  
        batura-=n .  
    ÿ Konpilatu iturburu-modulua sinboloen taula txertatzeko aukerarekin

- ÿ Ireak arazketa eta kargatu modulu bitarra
- ÿ Exekutatu urratsez urrats batura partzialak behatuz komando hauetan:
  - ÿ  $x /1db +, x /1tb +, x /1ob +, x /1xb +$
  - ÿ adierazi emaitza komandoari
  - ÿ  $x$  laguntzarekin azaldu  $1db, 1tb, 1ob, 1xb$ -en esanahia.

## 12.6. sum1toN.s programazioa

### 12.6.1. Algoritmoa

- Garatu programa bat AT&T mihiztzaile lengoaien i386 arkitekturarekin, zeinaren emaitza hasierako deskribapen  $\sum_{i=1}^N i$  programazio-metodoa  $N(N+1)/2$  erabiltzen duen batura egiten duena. pseudokodea eta organigrama.

### 12.6.2. Editatu iturburu-modulua: sum1toN.s

- Deskargatu "sum1toN.s" iturburu-modulua myaulariotik eta gehitu komentario egokiak.
- x86 Intel-en 32 biteko arkitektura da
- i386 linux-en esan nahi du: x86-32 arkitektura
- GNU AT&T i386 arkitekturarako mihiztadura-lengoia ÿ GNU hizkuntza gisa ÿ gas-lengoia

```
### Programa: sum1toN.s
### Deskribapena: 1,2,3,...N serieen batura egiten du. Sarreran definitzen da
programa propioa eta irteera OSra pasatzen da
### Hizkuntza: i386 arkitekturarako GNU mutua-lengoia -> GNU gisa ->
gasa -> AT&T
### IAS makinaren sum.ias-en baliokide den AT&T i386 hizkuntzako programa da.
von Neumann
### gcc -m32 -g -nostartfiles -o sum1toN sum1toN.s
### Asanblada gisa --32 --gstabs sum1toN.s -o sum1toN.o
### estekatzalea -> ld -melf_i386 -o sum1toN sum1toN.o

## Aldagaien adierazpena
## DATU ATALA
.atala .datuak

n: .int 5

.global _hasiera

## Kodearen hasiera
## JARRAIBIDEEN ATALA
.atala .testua
._hasi:
    mov $0,%ecx # ECX-k batura aldaagaia ezartzen du
    mov n,%edx
begizta:
    gehitu %edx,%ecx
```

```
azpi $1,%edx
jnz begizta

mov %ecx, %ebx # irteerako argumentua OSra EBX bidez ABI i386 konbentzioaren arabera

## irteera
mov $1, %eax # sistema eragilearen dei-kodea: subroutine exit int $0x80 # deitu sistema eragileari
azpirrutina exekutatzeko EAX-en balioan oinarrituta

.amaiera
```

### 12.6.3. Konpilazioa

- Modulu bat mutua-lengoaian osatzeko pausoak jarraitu.

ÿ gcc -m32 -g -o sum1toN sum1toN.s

### 12.6.4. Exekuzioa

- ./sum1toN
- oihartzuna \$?

### 12.6.5. Iturria moduluaren analisia

- Irakurri Minimalist Example Programa erreferentzia azkarreko orriean

### 12.6.6. Depurazioa

Hasi

- GDB Debugger: GNU Debugger.
- gcc -m32 -g -o sum1toN sum1toN.s
- gdb
- Cx a
- Cx edo
- aktibatu arrasto-komandoak
- ezarri sum1toN\_gdb\_asm.txt erreregistro-fitxategia
- saioa hasteko ezarri
- shell ls -l sum1toN\_gdb\_asm.txt

Hasi programa

- sum1toN fitxategia
- informazio iturriak
- b \_hasi

- korrika

### Analisi ikurrak memorian

- pmota n
- p.n.
  - x helbidea : memoria nagusiaren helbidea aztertu . Memoria helbidearen edukia itzultzen du ý &sinboloa non ikurra aldagaiaren izena den.
- x &n, x n, x /1bw &n, x /1xw &n, x /4xw &n
- b begizta
- c
- hasi
- c
- n

### Erregistroak

- p \$ecx
- p \$edx
- arte
- p \$ecx
- p \$edx
- informazio erregistroak

### Makinaren argibideak

- zatitu diseinua
- hasi
- c

Amalera

- irten
- Konsolan erabilitako komando guztiak biltzen dituen [gedit sum1toN\\_gdb\\_asm.txt](#) fitxategia irekitzen dugu. beren zabortegiekin.
- Gorde sum1toN\_gdb\_asm.txt-en edukia Memoria Dokumentuan iruzkinak gehituz beharrezkoak.

## 12.7. amd64 arkitektura

- Adibidea eranskinean: [sum1toN.s](#)

Ý Kodea aztertu, konpilatu, exekutatu eta araztagailua erabiliz egiaztatzea erregistroak benetan 64 biteko direla.

# 13. kapitulua. Datuen irudikapena

## 13.1. Sarrera

### 13.1.1. Helburuak

- Programazioa:

- ÿ Datu mota desberdinak gorde eta prozesatzen dituzten programak garatzea, hala nola sinatutako zenbaki osoak, karaktereak, matrizeak, tamaina ezberdinak kateak, hala nola 1 byte, 2 byte, 4 byte, etab.
- ÿ Atzizki mnemoteknikoen erabilera: operandoen tamaina aztertu mnemoteknikoaren atzizkiaren arabera. erabilitako mnemoteknia, registroaren eragigaiaren tamaina eta memorian dagoen eragigai mota.
- ÿ Erabili eragigaietara sartzeko helbideratze-modu desberdinak (berehalako, zuzena, zeharkakoa, indexatua).

- ÿ Makroak zuzentarauen bidez erabiltzea.

- ÿ Sistema eragileari deitzeko kontzeptua.

- Analisia:

- ÿ Memorian gordetako little endian lerrokatzeko-mota egiaztatzea

- ÿ Analizatu memoriaren edukia sinatutako zenbaki, karaktere, matrize eta kate gisa: motak eta zenbakizko eragigaien eta alfanumeriko eragiketen tamainak

- ÿ Egin desmuntatze eragiketa memoria nagusian kargatutako modulu exekutagarriaren makina-lengoaia egiazatzeko.

### 13.1.2. Iturburu-moduluak: ezaugarriak

- **datu\_tamaina.s**

- ÿ Eragileen tamainaren adierazpena:

- ÿ Zuzentarauen bidez (.byte, .2byte, .short, etab...)

- ÿ Zenbakizko datu- **matrizeen** deklarazioa zuzentarauak erabiliz (.short,.int, etab..)

- ÿ Datu **alfanumerikoen** adierazpena :

- ÿ Muntatzaileen zuzentarauak (.ascii, .asciiz, .string, etab.)

- ÿ Makroen erabilera

- **atzizki\_datuak.s**

- ÿ Eragigaietarako sarbidea (b,w,l,q) **atzizkiak** dituzten argibideen bidez

- **helbideratze\_datuak.s**

- ÿ Eragigaien helbideratze modu desberdinak: berehalako, zuzena, zeharkakoa, indexatua

### 13.1.3. Baldintzak

- Teoria: datuen irudikapena, instrukzio-formatua eta X86 arkitekturaren ISA errepetorioa.

- ÿ Little endian byte- **lerrotutako** biltegiratzea

- Aurreko praktika: Programaziorako sarrera AT&T x86-32 mihiztadura-lengoaien • C **programazio-lengoaiaren** kontzeptuak:

- ÿ Erakusleak, array, katea eta casting-eragiketa.

## 13.2. IRAKUR NAIZ

- Praktikaren gidoia eta Programazioaren 3. kapituluua Ground-Up Booketik irakurtzea.
- Oharrak eta Testu liburua
- Txostenaren dokumentazioa: Txostenaren edukia eta formatua
- Ebaluazioa: ebaluazio sistema
- Garapen Plataforma: ordenagailu pertsonalaren konfiguraziaoa
- Programazioa: metodologia

## 13.3. Aukerako galderak

- Aukerako “Praktikaren Autoebaluazioa”: Praktikak: Galdetegia
  - ÿ Aukerako ariketek 4 puntu gehitzen dizkiote txostenen kalifikazioari.

## 13.4. CPU barneko erregistroak

- Amd64 arkitekturak honako hauek ditu:
  - ÿ 64 biteko 16 helburu orokorreko erregistro (RPG) bakoitza: rax, rbx, rcx, rdx, rsi, rdi, rsp, etab.
  - ÿ 1 64 biteko egoera-erregistroa: rflags
- Helburu orokorreko erregistroetarako sarbidea partziala izan daiteke:
  - ÿ RAX erregistroa: 64 biteko erregistroa da
  - ÿ EAX erregistroa: RAXen 32 bit txikienak dira
  - ÿ AX erregistroa: RAXen 16 bit txikienak dira
  - ÿ AL erregistroa: RAXen 8 bit txikienak dira
  - ÿ Erregistratu AH: RAXen 8:15 posizioetan dauden bitak dituen byta da.
- "Erreferentzia azkarreko orriean" talde ezberdinien izen guztiak irudikatzen dira helburu orokorreko erregistro bakoitzeko bit kopurua.

## 13.5. Datuen tamaina eta aldagaiak

### 13.5.1. Algoritmoa

- Argibideen atalean bi erakusle hasieratzen dituen algoritmo bat dago.

### 13.5.2. Iturburu-modulua editatzea:data\_size.s

- Deskargatu "datos\_size.s" iturburu-modulua myaulariotik eta gehitu komentario egokiak.

```
### Programa: data_size.s ###  
Deskribapena: deklaratu eta atzitu eragigaien tamaina desberdinak ### Konpilazioa:  
gcc -m32 -g -o data_size data_size.s  
  
## MAKROAK  
.equ SYS_EXIT, 1 .equ  
SUCCESS, 0
```

```
ÿ ## TOKICO ALDAGAIAKÿ .datuak
```

```
da1:     .byte 0x0A
da2:     .2byte 0x0A0B
da4:     .4byte 0x0A0B0C0D .ascii
        "kaixo" menu1:
zerrenda: .int 1,2,3,4,5
```

```
ÿ ## JARRAIBIDEAK
```

```
ÿ .global _hasieraÿ .testua
```

```
_hasi:ÿ
mov $da4,%eaxÿ
irakurri da4,%ebxÿ
mov (%eax),%ecxÿ mov
(%ebx),%edx irteera:ÿ
mugitu
$SYS_EXIT, %eaxÿ mov
$SUCCESS, %ebxÿ int $0x80
```

```
ÿ .amaiera
```

### 13.5.3. Konpilazioa

- Saio guzietan ohikoak diren [bilketa-](#) prozesuaren urratsak jarraitu .

ÿ gcc -nostartfiles -m32 -g -o data\_size data\_size.s

### 13.5.4. Exekuzioa

- ./datuen\_tamaina
- oihartzuna \$?

### 13.5.5. Iturburu-moduluaren azterketa

- Irakurri [Minimalist Example Programa](#) erreferentzia azkarreko orriean

#### Egitura ataletan: muntaia

- Programaren egitura elementu hauek osatzen dute:
  - ÿ Goiburua
  - ÿ Makroen definizioa
  - ÿ Datuen Atala
  - ÿ Argibideen Atala

#### Makroen definizioa

- Makroa:
  - ÿ Makro eraikuntza iturburu-programan erabiltzen da programan erabiltzen diren datuak ordezkatzen

iturburua iturburu-kodea erraz irakurtzen duten testu-ikurren bidez.

ÿ Horretarako sintaxia duen "EQU" zuzentaraua erabiltzen dugu: .EQU **SYMBOL**, datuak

ÿ Konpilazioaren lehen faseko aurreprozesadoreak datu eta argibideen atalean zehar agertzen den SINBOLO testua erlazionatutako datuekin ordezkatuko du.

- Erabilitako makroak

ÿ SYS\_EXIT: sistema-deiaren kodea programa amaitzeko eta Sistema Eragileari kontrola itzultzeko. i386 arkitekturan bere balioa 1 da.

ÿ ARRAKASTA: programek exekuzioa egin dela adierazteko erabiltzen duten kodea normala. Bere balioa 0 da.

## Datuaren Atala

- Datuen memoria erreserbatzeko eta hasieratzeko etiketak eta zuzentaraauak interpretatzea [zuzentaraau-taula erabiliz](#): aldagai arruntak, kateak eta arrayak identifikatu.



Memoria objektu bat objektuaren tamaina baino zifra gutxiagorekin adierazten den zenbaki oso batekin hasieratzen bada, zifra handienek zero balioa izango dute.

Adibidez: **.4byte 0xFF .4byte 0x000000FF** -ren baliokidea da

## Argibideen Atala

- Programarako sarrerako instrukzioa zehaztu.
- Programaren irteera blokea zehaztu.

## 13.5.6. GDB: Behaketak

- Araztailea erregistroen edukia ikustean:

ÿ Eragileen tamainako byte kopurua bakarrik bistaratzen du... "rx" erregistroak 64 biteko diren arren (rax, rbx, etab.)

ÿ Zenbaki osoekin, ez du pisu handieneko zerorik erakusten, hau da, ez zeinua ez zenbaki positiboen zeinu luzapena.

## 13.5.7. GDB: urratsez urrats exekuzioa

### Hasieratzea

- Konpilatu programa araztaileak eskatzen duen sinboloen taula sortzeko aukerarekin eta sortu modulu bitar exekutagarria:

ÿ **gcc -nostartfiles -m32 -g -o data\_size data\_size.s**

- Ireaki GDB aratzketa, kargatu modulu bitar exekutagarria eta egiaztatu taula kargatuta dagoela. modulu bitar exekutagarriaren ondoan dauden sinbololoak.

ÿ **gdb**

ÿ **fitxategiaren datuak\_tamaina**

ÿ **informazio iturriak**

- Ireaki iturburu-moduluaren leihoa

ÿ **layout src** edo **Control-x Control-a**

- Komandoen erregistro historikorako fitxategia konfiguratu.

ÿ **aktibatu arrasto-komandoak**

ÿ ezarri erregistro-fitxategia **data\_size\_gdb\_asm.txt**

ÿ ezarri saioa hasteko

ÿ shell ls -l **data\_size\_gdb\_asm.txt**

- Aktibatu eten puntu bat programa sartzeko instrukzioan.

ÿ **b \_hasi**

- Exekutatu programa programaren lehen adierazpenean geldituz.

ÿ **korrika**

### Komandoak eta eragileak: x, p, disas, casting, &, \*, @

- eXaminar x komandoa: memoria- **helbide** baten edukia iraultzen du

ÿ **x /nvt helbidea**

ÿ /nvt formatua: "t" memoriako aldagaiaren tamaina da , "v" bistaratu nahi den memoria-edukiaren balioaren kodeketa eta "n" byte-taldeak sekuentzialki iraultzeko "t" tamainako memoriara helbide-helbidetik hasita.

ÿ **lagunza x** : d (hamartarra), x (hamasetarra), t (bitarra), o (oktala), c (karaktere), a (helbidea), i formatuak (argibideak), etab.

ÿ Adibideak:

ÿ x /1d4 helbidea (exekutatu azterketa komandoa kode hamartarrean behin, botaz a 4 byte helbidea helbidean kokatuta)

ÿ x /2t4 helbidea (exekutatu aztertzeko komandoa bi aldiz kode bitarrean: lehen aldian helbide-helbidean kokatutako 4 byte-ko objektu bat bota eta bigarrenean helbide-helbidean kokatutako 4-byte-ko objektua +4)

ÿ x /100x1 helbidea: 100 byte-ko datuak iraultzen ditu helbidetik kode hamaseimalean helbidea.

ÿ Aztertu komandoaren argumentuaren sintaxia programazio-lengoaian dagoen bera **da** c.

ÿ laneko hizkuntza: **erakutsi hizkuntza** ÿ GDB esamoldeen hizkuntza ASM dela adierazten du (adibidez, \$eax) baina C hizkuntza ere onartzen duela (&aldagaia)

ÿ Laneko hizkuntzak onartzen: **ezarri hizkuntza**

- & operadorea : etiketa baten aurritzka jartzeko erabiltzen da, aipatzen duen memoria helbidea ebalutzeko erreferentzia etiketa bat

- \* operadorea memoria-kokapen baten edukia erakusleen zeharkakotasunaren bidez ebalutzeko erabiltzen da

- galdaleta eragiketa :

ÿ **C Eranskina Lengoaien Programazioa**

ÿ Casting aldagai mota definitzean edo birdefinitzean datza. Birdefinitu beharreko aldagaiaren aurritzki gisa erabiltzen da eta parentesi artean sartzen da.

ÿ "zerrenda" etiketa datuen atalean definitzen da ".int" zuzentaraua erabiliz. Zuzentarau honek memoria gordetzen du helbidetik &zerrendako datuak hasieratzeko baina EZ da mota-adierazpena, beraz, arazketa-ak ez du informaziorik zerrenda-matrizeko elementu motari buruz eta, beraz, beharrezkoa da deklarazioak casting moduan egitea.

ÿ Adib. (char \*): **char \*** mota 1 byte osoko zenbaki baten erakuslea da.

- Print p komandoa : komandoaren argumentua ebaluatzen du eta ondoriozko balioa pantailan inprimatzen du

ÿ Aztertu komandoaren argumentuaren sintaxia C programazio-lengoaian bezalakoa da.

ÿ Adib. p /a &zerrenda: emaitzaren balioa "a" (helbidea) formatuan inprimatuta dagoen &zerrenda ebaluatzen du  
 ÿ inprimatzeko formatuak: eXaminar bezalakoak: laguntza x  
 ÿ operadorea @: helbidea@n: matrize artifiziala: "helbidea" adierazpena ebaluatzen du (@-ren ezkerraldean) eta  
 memoria-helbide bat izan behar du. "n" luzerako (@ operadorearen eskuinean dagoen parametroaren balioa) byte-ko array artifizial bat sortzen du.

- disas komandoa : kode bitarra desmuntatzen du muntaketa-kode bihurtuz.

## Analisia

- Memoria nagusiko edukien analisia GDB araztailea erabiliz:

```
//Datu baten byteen barne lerroatzea x /tb &da1 x /xh &da2
```

```
x /xw &da4
x /5xb &da4 -> Endian lerrokadura txikia
```

```
//Kate baten byteen lerroatzea x /5cb &men1
-> Lerroatzea x /5xb &men1
```

sekuentzian

```
//Kate baten iraulketa p /s
(char *)&men1 -> karaktere-kate bat inprimatzen du lehen helbidetik NULL karakterea (0x00) aurkitu arte.
```

```
//Matrize baten iraulketa x /
5xw &zerrenda          -> 5 zerrendako elementuen edukia
p /a &zerrenda -> array-zerrendaren helbidea p /a &zerrenda+1
-> araztaileak igorpen motaren bat (deklarazio dinamikoa) egitea beharrezko dela jakinarazten du p /a
( void *) &list+1 -> 1 bytez gehitzen da p /
a (int *)&list+1 -> eskalatzea: 1*4 bytez gehitzen da array-ko bigarren
elementura p list casting p (int)list -> p arrayaren lehen elementua (int *)&list -> array zerrendaren helbidea
p *((int *)&list+1) -> zerrendaren
bigarren                -> araztatzaleak a egiteko beharrezko dela jakinarazten du
elementua
x /dw (int *)&list+1 -> zerrendaren bigarren elementua p (int [ 5])list p
*(int *)&list@5 helbidea &zerrenda.
```

```
-> zerrendako bost elementuren edukia
-> int motako 5 elementuren array artifiziala
```

```
//muntaia-agindu baten iraulketa p
&_start x /i
&_start.           -> desmuntatu: makina kodea kodea bihurtu
```

```
//Desmuntatu: makina kodea desmuntatzale bihurtzea /r _start
```

```
zatitu diseinua
```

```
// Erakusleen b irteerako
azterketa
c
p /a &da4
x /x4 &da4
p /x (int)da4 p /
x $eax p /
x *(int *)$eax
```

## 13.6. Operandoaren tamaina

### 13.6.1. Iturburu-moduluaren edizioa: data\_atzizkiak.s

- Jaitsi "datos\_suffijos.s" iturburu-moduluua myaulariotik eta gehitu komentario egokiak.

```
### Programa: data_suffixes.s ####
Deskribapena: mnemoteknikoa egiteko atzizki desberdinak erabili operandoen tamaina desberdinak
adierazten dituzten ####
Konpilazioa: gcc -nostartfiles -m32 -g -o data_suffixes data_suffixes.s

## MAKROS .equ
SYS_EXIT, 1 .equ
SUCCESS, 0

## TOKIKO ALDAGAIAK .datuak

da1:     .byte 0x0A
da2: .2byte 0x0A0B da4:
        .4byte 0x0A0B0C0D
agurra: .ascii "kaixo"
zerrenda: .int      1,2,3,4,5

## JARRAIBIDEAK .global
_hasi .testua

_hasi:

## Berrezarri Erregistroak xor
%eax,%eax xor %ebx,
%ebx xor %ecx,%ecx
xor %edx,%edx

## Kargatu datuak ##
mov da1,da4 bi           ERROREA: bi helbide eraginkorren erreferentziagatik
eragiketa memoria nagusian mov da4,%eax
```

```
movl da4,%ebx movw  
da4,%cx movb da4,%dl
```

```
## Berrezarri Erregistroak xor  
%eax,%eax xor %ebx,  
%ebx xor %ecx,%ecx  
xor %edx,%edx
```

```
## Kargatu datuak mov  
da4,%al ## movw #aplikatu AL tamaina  
da4,%al movb da4,%ebx ERROREA: -w eta ALren arteko inkoherentzia  
atzizkia #OHARRA, EZ errorea: BL erregistroaren eta koherentziaren artean
```

```
mov da1,%ecx mov  
da4,%dx
```

```
## Berrezarri Erregistroak xor  
%eax,%eax xor %ebx,  
%ebx xor %ecx,%ecx  
xor %edx,%edx
```

## Datuak kargatzen

```
mov da1,%al
```

```
## inc da1 ERROREA: eragaiaren helbide eraginkorra memoria nagusiaren  
erreferentzia denez, ez du eragaiaren tamaina mugatzen. Atzizkirik ere ez zehaztuz, mihiztatzaleak  
ez du eragaiaren tamaina ezagutzen. incb da1 incw da2
```

barne da4

```
## irten  
$SYS_EXIT mugimendua,  
%eax mugimendua $SUCCESS,  
%ebx int $0x80
```

.amaiera

### 13.6.2. Konpilazioa

- Modulu bat muntaia-lengoaian osatzeko pausoak jarraitu.
  - ÿ gcc -nostartfiles -m32 -g -o suffix\_data suffix\_data.s
  - ÿ OHARRA: Abisua: %bl erabiltzea %ebx-ren ordez b atzizki gisa erabiltzeagatik

ŷ Movb da4,%ebx instrukzioaren sintaxiari buruzko abisua da , EZ da errorea.

### 13.6.3. Exekuzioa

- ./datu\_atzizkiak
- oihartzuna \$?

### 13.6.4. asm iturri-moduluaren azterketa

- Eragaien tamaina desberdinak adierazten dituzten atzizki mnemoteknikoak: b,w,l

ŷ **movw da4,%cx** : 2 byteko "w" atzizkia eta bi byteko CX helmuga erregistroa.

ŷ **movw da4,%al** : "w" atzizkiak 2 byte-ko transferentzia behartzen du 1-byteko AL helburu-erregistro batera ŷ muntaketa-errorea.

ŷ **movb da4,%ebx** - "b" atzizkia ez dator bat 4 byteko EBX helburuko erregistroarekin eta muntaketarekin  
BLrekin ekoizten da.

- Atzizkirk gabe:

ŷ **xor %eax,%eax** : 4 byteko EAX iturburu eta helmuga eragaiak

ŷ **mov da4,%al** : AL helburuko erregistroak byte 1era mugatzen du transferentzia eta ez dago atzizkiarekin kontraesanik ez dagoenez.

ŷ **mov da1,%ecx** : bi eragiketen artean, erregistroa eta memoria, erregistroa da tamaina lehenesten duena.  
transferentzia.

ŷ **inc da1** : Eragaiaren helbide eraginkorra memoria nagusiaren erreferentzia denez, ez du eragaiaren tamaina mugatzen. Atzizkirk  
ere ez zehaztuz, mihiztzaileak ez du eragaiaren tamaina ezagutzen ŷ muntaketa-errorea

### 13.6.5. Eragaiaren tamaina asm instrukzio batean ondorioztatzea

1. Memorian edo erregistroan eragigai baten erreferentziaren arteko aldea

to. Memoria-helbide batek erreferentziatutako eragigai batek ez du tamaina zehatzik  
muntatzailea.

b. Bestalde, erregistro baten izena eragaien tamaina batekin lotzen du muntatzaileak.

2. Memorian eragiketa bakarra duen instrukzio batean tamaina atzizki mnemoteknikoari esker ondorioztatzen du muntatzaileak, beraz, kasu  
honetan mnemoteknikoak atzizkirk ez badu mihiztzaileak ez du instrukzioa itzuliko.

3. Bi eragai dituen instrukzio batean, bata memorian eta bestea erregistroan, erregistroko eragigaia edo atzizkia da bi eragai iturri eta  
helmugaren tamaina zehazten duena:

to. Mnemoteknikoak atzizkia badu, iturburuko eragiketen tamaina zehazten duen atzizkia esaten da eta  
helmuga.

b. Mnemoteknikoak atzizkirk ez badu, erregistroaren tamaina da eragaien tamaina zehazten duena.  
iturria eta helmuga.

4. Errore kasuak

to. Mnemoteknikoak helmuga-erregistroaren tamaina baino atzizkia handiagoa badu.

b. Mnemoteknikoak atzizkirk ez duen eta iturburu-erregistroaren tamaina baino handiagoa den kasuetan  
helmuga-erregistroa.

### 13.6.6. GDB: urratsez urrats exekuzioa

## Hasieratzea

- Konpilatu programa araztaileak eskatzen duen sinbolean taula sortzeko aukerarekin eta sortu modulu bitar exekutagarria:
  - ÿ gcc -nostartfiles -m32 -g -o suffix\_data suffix\_data.s non source\_module konpilatu nahi duzun fitxategiaren izenarekin ordezkatzen den.
- Ireki GDB arazketa, kargatu modulu bitar exekutagarria eta egiaztatu taula kargatuta dagoela. modulu bitar exekutagarriaren ondoan dauden sinbolean.
- ÿ gdb
  - ÿ exekutagarri\_modulu fitxategia
  - ÿ informazio iturriak
- Ireki iturburu-moduluaren leihoa
  - ÿ layout src edo Control-x Control-a
- Komandoen erregistro historikorako fitxategia konfiguratu.
  - ÿ aktibatu arrasto-komandoak
  - ÿ ezarri erregistro-fitxategia data\_suffix\_gdb\_asm.txt
  - ÿ ezarri saioa hasteko
  - ÿ shell ls -l data\_suffix\_gdb\_asm.txt
- Aktibatu eten puntu bat programa sartzeko instrukzioan.
  - ÿ b \_hasi
- Exekutatu programa programaren lehen adierazpenean geldituz.
  - ÿ korrika
- ireki erregistroak ikusteko leihoa
  - ÿ diseinu-erreg
- Memoria nagusiko edukien azterketa GDB araztailea erabiliz.
  - ÿ Programa exekutatu urratsez urrats instrukzio bakoitzaren exekuzioaren emaitza aztertz
  - ÿ n
  - ÿ RET, RET, RET, ....

## 13.7. Helbideratzeko moduak

## 13.7.1. Iturburu-moduluaren edizioa: data\_addressing.s

- Deskargatu "data\_addressing.s" iturburu-modulu myaulariotik eta gehitu komentario egokiak.

```

### Programa: helbideratze_datuak.s
### Deskribapena: Erabili datu-egiturak helbide ezberdinekin ### Konpilazioa: gcc -m32 -g -o
addressing_data addressing_data.s hasierako fitxategien aukerarik gabe ### etiketarekin erreferentziatutako
sarrera-puntua erabiltzean

## MAKROAK
.equ SYS_EXIT, 1 .equ
SUCCESS, 0

```

```

ÿ ## TOKIKO ALDAGAIAK
ÿ .datuak

ÿ .lerrokatu 4 4 # MP- ekin lerrokatzeak multiploak dituen helbideak

da2: .2byte 0x0A0B,0b000011101011100,-21.0xFFFF # Array da2 2 elementuz

byteÿ .lerrokatu
4 1,2,3,4,5 # 2 byteko elementuen array zerrenda

zerrenda: .hitzaÿ .lerrokatu # 100 byteko buffer array
8 buffer: .space

100ÿ .lerrokatu 2 agurra:ÿ .string #Koixoi bateko elementuen agurra karaktereak direlako

ÿ ## JARRAIBIDEAK
ÿ .global nagusiÿ .text
main:

ÿ ## Berrezarri

ÿ xor %eax,%eaxÿ xor
%ebx,%ebxÿ xor %ecx,
%ecxÿ xor %edx,%edxÿ
xor %esi,%esi xor %edi,
%edi

ÿ ## sum1toN ALGORITMOA

ÿ ## Berehalako helbideaÿ mugitu $4,%bai

ÿ ## Helbideratze- begizta indexatua: gehitu
zerrenda(%esi,2),%di
ÿ ## Izena emateko helbideaÿ esan %bai

ÿ ## PC - helbide erlatiboaÿ jns begizta

ÿ ## ZUZENDARI BURUZKO ARIKETAK

ÿ ## Zeharkako helbideratzeaÿ irakurri buffer,
%eaxÿ ## mov da2,(%eax) #hasi EAX erakuslea
ERROREA: bi eragiketen helbide eraginkorra memoria nagusiari dagokioÿ mov da2,%bxÿ mugimendua %bx,
(%eax)

ÿ ## Zuzeneko helbidea

```

```

incw da2
## Helbide indexatua
irakurri da2,%ebx ##
inc 2(%ebx) ERRORREA: helbide eraginkorra memorian eta incw 2 atzizkirk ez (%ebx)

mov $3,%esi mov
da2(%esi,2),%ebx

## IRLEN

$SYS_EXIT mugimendua,
%eax mugimendua
$SUCCESS, %ebx int $0x80

.amaiera

```

### 13.7.2. Konpilazioa

- Modulu bat mutua-lengoaian osatzeko pausoak jarraitu.
  - ÿ Sarrera-puntuaz ez da "\_start".
  - ÿ gcc -m32 -g -o helbideratze\_datuak helbideratze\_datuak.s

### 13.7.3. Exekuzioa

- ./helbideratze\_datuak
- oiartzuna \$?

### 13.7.4. asm iturri-moduluaren azterketa

- Datuak .align n zuzentaraua erabiliz lerrokatzeak n-ren memoria-helbide anitz bat esleitzen dio hurrengo deklaratutako datuei.

ÿ EZ da onartzen bi eragiketa dituen instrukzio baten kasuan, biak memoria nagusian egotea.  
Eragigai bat edo biak helburu orokorreko erregistroetan egon behar dira.

### 13.7.5. GDB: urratsez urrats exekuzioa

#### hasieratzea

- Konpilatu programa araztaileak eskatzen duen sinboloen taula sortzeko aukerarekin eta sortu modulu bitar exekutagarria:
  - ÿ gcc -m32 -g -o addressing\_data helbide\_datuak.s non iturri\_modulua den konpilatu nahi duzun fitxategiaren izenaren ordez.
- Ireki GDB arazketa, kargatu modulu bitar exekutagarria eta egiaztatu taula kargatuta dagoela. modulu bitar exekutagarriaren ondoan dauden sinboloak.
  - ÿ gdb
  - ÿ fitxategi helbideratze\_datuak

ÿ informazio iturriak

- Ireki iturburu-moduluaren leihoa
  - ÿ layout src edo Control-x Control-a
- Komandoen registro historikorako fitxategia konfiguratu.

ÿ aktibatu arrasto-komandoak

ÿ ezarri gdb\_asm\_address\_data.txt registro-fitxategia

ÿ ezarri saioa hasteko

ÿ shell ls -l address\_data\_gdb\_asm.txt

- Aktibatu eten puntu bat programa sartzeko instrukzioan.

ÿ b \_hasi

- Exekutatu programa programaren lehen adierazpenean geldituz.

ÿ  
Komika egin

## Analisia

- Array da2

ÿ Inprimatu da2 matrizearen memoria helbidea eta lehen elementuaren edukia: x /xh &da2

ÿ da2 arrayaren 2 byteko 4 elementu: x /4xh &da2

ÿ p /x (laburra[4])da2

- Zerrenda array

ÿ ptype zerrenda

ÿ p (labur[5])zerrenda

- Array buffer

ÿ ptype buffer

ÿ Inprimatu buffer arrayaren memoria helbidea eta egiaztatu haren lerrokatzea: p &buffer

- Katea

ÿ ptype greeting: ez dago arazketa informaziorik ÿ ez du onartzen array-elementuen erreferentziaren agur-esamoldea[n]

ÿ p /c (char[5])agurra :casting array

ÿ x /5c (char \*)&agurra: igortzeko erakuslea

ÿ p /c \*(char \*)&agurra: igortza erakuslea eta zeharka

ÿ p /s (char \*)&agurra: igortzeko erakuslea eta kate formatua

# 14. kapitula. Eragiketa aritmetikoak eta logikoak

## 14.1. Sarrera

### 14.1.1. Helburuak

- Programazioa:
  - ÿ Zenbaki osoekin eragiketa aritmetikoak egitea (baketa, kenketa, biderketa eta zatiketa).
- Analisia:
  - ÿ Egiaztu eragiketa logiko eta aritmetikoek egoera-erregistroaren banderak nola eragiten dituzten I RENDERAK
  - ÿ Analizatu memoriaren edukia sinatutako zenbaki, karaktere, matrize eta kate gisa: motak eta zenbakizko eragigaien eta alfanumeriko eragiketen tamainak
  - ÿ Egin desmuntatze eragiketa memoria nagusian kargatutako modulu exekutarriaren makina-lengoaia egiazatzeko.

### 14.1.2. Arkitektura kontzeptuak

- ALU Unitate Logiko Aritmetikoak helburu orokorreko erregistroetan gordetako zenbaki osoekin bakarrik funtzionatzen du. Zenbaki errealekin funtzionatzeko, Float Process Unit FPU beharrezkoa da koma mugikorreko zenbakietarako erregistro espezifikoetan gordetako eragigaietan.

### 14.1.3. Iturburu-moduluak

- [op\\_arit\\_log.s](#)

### 14.1.4. Baldintzak

- Teoria: datuen irudikapena, eragiketa aritmetikoak eta logikoak, instrukzio-formatua eta errepertorioa X86 arkitekturaren ISA.
- Aurreko praktikak:
  - ÿ Programaziorako sarrera AT&T x86-32 mutua-lengoaian
  - ÿ Datuen irudikapena

## 14.2. IRAKURRI

- Praktikaren gidoia eta Programazioaren 3. kapitula Ground-Up Booketik irakurtzea.
- [Oharrak eta Testu liburua](#)
- [Txostenaren dokumentazioa](#): Txostenaren edukia eta formatua
- [Ebaluazioa](#): ebaluazio sistema
- [Garapen Plataforma](#) : ordenagailu pertsonalaren konfigurazioa
- [Programazioa](#) : metodologia

## 14.3. Gaiak

- Aukerako "[Praktikatu Autoebaluazioa](#)" : Praktikak: Galdetegia

#### 14.4. CPU barneko erregistroak

- Amd64 arkitekturak honako hauek ditu:
  - ÿ 64 biteko 16 helburu orokorreko erregistro (RPG) bakoitza: rax, rbx, rcx, rdx, rsi, rdi, rsp, etab.
  - ÿ 1 64 biteko egoera-erregistroa: rflags
- Helburu orokorreko erregistroetarako sarbidea partziala izan daiteke:
  - ÿ RAX erregistroa: 64 biteko erregistroa da
  - ÿ EAX erregistroa: RAXen 32 bit txikienak dira
  - ÿ AX erregistroa: RAXen 16 bit txikienak dira
  - ÿ AL erregistroa: RAXen 8 bit txikienak dira
  - ÿ Erregistroa AH: RAXen 8:15 posizioetan dauden bitak dituen byta da.
- "Erreferentzia azkarreko orriean" talde ezberdinaren izen guztiak irudikatzen dira helburu orokorreko erregistro bakoitzeko bit kopurua.

#### 14.5. Eragiketa aritmetikoak eta logikoak Zenbaki osoekin

##### 14.5.1. Iturburu-modulua editatzea: op\_arit\_log.s

- Deskargatu "op\_arit\_log.s" iturburu-modulua myaulariotik eta gehitu komentario egokiak.

```
### Programa:      op_arit_log.s
### Deskribapena: Eragiketa logiko eta aritmetiko ezberdinekin datu-egiturak erabili.

### Eraikitza: gcc -m32 -g -o op_arit_log op_arit_log.s

##
MAKROS .equ
SYS_EXIT, 1 .equ
SUCCESS, 0 .equ N, 5

## TOKIKO ALDAGAIAK

## JARRAIBIDEAK
.global main .text
main:

## Berrezarri

xor %eax,%eax xor
%ebx,%ebx xor %ecx,
%ecx xor %edx,%edx
xor %esi,%esi xor
%edi,%edi
```

```
## ERAGIKETA ARITMETIKOAK ZENBAKI OSOekin
```

```
## gehitu: batura
mugitu $ 5,%eax
mugitu $10,%ebx
gehitu %ebx,%eax
```

```
## azpi: kenketa ÿ
mugitu $ 5,%bez
mugitu $10,%ebx
azpi %ebx,%eax
```

```
## imul: "sinatua" zenbaki osoen biderketa: AX<- BL*AL ÿ
```

```
    movb $-3,%bl
movb $5,%al
imulb %bl
```

```
    ## idiv: "sinatua" zatiketa. (AL=kotientea, AH=hondarra) <- AX/(byte barne
erregistroa edo memoria) ÿ
movw $5,%ax #dividend
movb $3,%bl #zatitzalea
idivb %bl # 5/3 = 1*3 + 2
```

```
## 2-ren osagarria: ezezko zeinua aldatzearen baliokidea
negb %bl
```

```
## N*(N+1)/2 adierazpena
```

```
movw $N,%bx
movw $(N+1),%ax
imulw %bx #imulw Op ; Op=hitza ; DX:AX<- AX*Op
movw $2,%bx
## Emaitza AX da eta gainerakoa DX=0 ÿ
idivw %bx           #idivw Op ; Op=hitza ; AX<-(DX:AX)/Op ; DX:=Atseden
```

```
## ERAGIKETA LOGIKOAK
```

```
mov $0xFFFF1F, %eax
    mov $0x0000F1, %ebx
ez %eax          # inbertsioa
eta %ebx,%eax # produktu logikoa
edo %ebx,%eax # batura logikoa
```

```
## 2-ren osagarria not() + 1 eragiketa logikoa erabiliz
mov %ebx,%eax
ez %eax
inc %eax
```

```
## Bit desplazamendua ÿ shr $4,%eax
    #desplazamendu logikoa: sartu beharreko bitak -> 0.. sar $4,%eax
#desplazamendu aritmetikoa: sartu beharreko bits -> zeinu-luzapena

## IRTEN

$SYS_EXIT mugimendua,
%eax mugimendua
$SUCCESS, %ebx int $0x80

.amaiera
```

ÿ Autoebaluazioko galderetan aipatzen diren argibideak

ÿ ..  
ÿ ..  
ÿ ..  
ÿ ..  
ÿ ..

#### 14.5.2. Konpilazioa

- Modulu bat mutuaia-lengoaian osatzeko pausoak jarraitu.
  - ÿ Sarrera-puntua ez da "\_start".
  - ÿ gcc -m32 -g -o op\_arit\_log op\_arit\_log.s

#### 14.5.3. Exekuzioa

- ./op\_arit\_log
- oihartzuna \$?

#### 14.5.4. Iturburu-moduluaren azterketa

- Add, sub, imul (zenbaki osoen biderketa), idiv (zenbakien zatiketa), neg, not, eta, or, xor, shr, sar instrukzioak interpretatzeko, konsultatu eragiketen taula erreferentzia azkarreko orriean.
- Eragiketek atzizki mnemoteknikoak adierazten duten bit kopurua soilik prozesatzen dute... nahiz eta "rx" erregistroak 64 biteko dira.

#### 14.5.5. Urratsez urrats exekuzioa

##### Behaketak

ÿ Araztailea erregistroen edukia ikustean: Eragileen tamainako byte kopurua bakarrik bistaratzen du... "rx" erregistroak 64 biteko diren arren. Zenbaki osoekin, ez ditu pisu handieneko zeroak bistaratzen, hau da, ez zeinua ez zenbaki positiboen zeinu-luzapena.

##### Eragiketak

- Konpilatu programa araztaileak eskatzen duen sinboloen taula sortzeko aukerarekin eta

sortu modulu bitar exekutagarria:

ÿ gcc -m32 -g -o op\_arit\_log op\_arit\_log.s non source\_module konpilatu nahi duzun fitxategiaren izenarekin ordezkatzen den.

- Ireki GDB arazketa, kargatu modulu bitar exekutagarria eta egiaztu taula kargatuta dagoela. modulu bitar exekutagarriaren ondoan dauden sinboloak.

ÿ gdb

ÿ op\_arit\_log fitxategia

ÿ informazio iturriak

- Komandoen erregeristro historikorako fitxategia konfiguratu.

ÿ aktibatu arrasto-komandoak

ÿ eazarri erregeristro-fitxategia op\_arit\_log\_gdb\_asm.txt

ÿ eazarri saioa hasteko

ÿ shell ls -l op\_arit\_log\_gdb\_asm.txt

- Aktibatu eten puntu bat programa sartzeko instrukzioan.

ÿ b \_hasi

- Exekutatu programa programaren lehen adierazpenean geldituz.

ÿ korrika

### Eragiketa aritmetikoak

- Batuketa, kenketa, biderketa, zatiketa eta eragiketa aritmetikoen emaitzak egiaztatzea sinatutako zenbaki osoen ezeztapena

### Eragiketa logikoak

- Egiaztu ezeztapen, biderketa, batuketa edo bitarteko eragiketa logikoen emaitzak esklusiboak eta desplazamenduak.

# 15. kapitula. Baldintzapeko jauzi-argibideak

## 15.1. Sarrera

### 15.1.1. Helburuak

- Banderen erregistroaren kudeaketa, konparazio jarraibideak eta baldintzapeko jauziak aplikatzeko goi-mailako hizkuntza adierazpenak, hala nola if, for, while, switch-case.
- GDB arazketa
  - ÿ [Watch](#) komandoa erabiliz

### 15.1.2. Baldintzak

- Teoria: datuen irudikapena, eragiketa aritmetikoak eta logikoak, instrukzio-formatua eta errepetorioa X86 arkitekturaren ISA.
- Aurreko praktika: AT&T x86-32 mihiztadura-lengoaien programatzeko hastapena
- C programazio-lengoaiaren kontzeptuak: if, for, while, switch-case

## 15.2. IRAKURRI

- Praktikaren gidoia eta Programazioaren 3. kapitula Ground-Up Booketik irakurtzea.
- [Oharak eta Testu liburua](#)
- [Txostenaren dokumentazioa](#): Txostenaren edukia eta formatua
- [Ebaluazioa](#): ebaluazio sistema
- [Garapen Plataforma](#) : ordenagailu pertsonalaren konfigurazioa
- [Programazioa](#) : metodologia

## 15.3. Gaiak

- Aukerako "[Praktikatu Autoebaluazioa](#)" : Praktikak: Galdetegia

## 15.4. Baldintzazko jauziak

### 15.4.1. Algoritmoa

- Ez da algoritmorik garatzen. Baldintzazko jauziak lantzeko jarraibideak dira.

### 15.4.2. Sorburuaren edizioa Modulua: jauziak.s

- Deskargatu "saltos.s" iturburu-modulua myaulariotik eta gehitu komentario egokiak.

```
/*
```

```
Programa: data_jumps.s
```

```
Deskribapena: Erabili helbide ezberdinako datu-egiturak ### Konpilazioa: gcc  
-m32 -g -o data_jumps data_jumps.s
```

```
*/
## MAKROAK
.equ SYS_EXIT, 1 .equ
SUCCESS, 0

## TOKIKO ALDAGAIAK .datuak

## JARRAIBIDEAK .global
main
.testua
nagusia:

## Berrezarri

xor %eax,%eax xor
%ebx,%ebx xor %ecx,
%ecx xor %edx,%edx
xor %esi,%esi xor %edi,
%edi

## BALDINTZA GABEKO JAIZIAK

## Helbide erlatiboa jmp hop1 #hop
erlatiboa programaren kontagailua pc -> eip xor %esi,%esi

## BANDERA ERREGISTROA BANDERAK ERAGIKETAK /
*
Eragiketa aritmetikoak, logikoak eta abar egiten direnean banderak aktibatu egiten dira, aipatutako eragiketa horren emaitzaren arabera CF:
Eragiketaren emaitza helmugako MSB bitak eramatzen du OF: El resultado de la señal gainezka egiten duen eragiketa, bere tamaina onartzen den tamaina gainditzen du.

ZF: eragiketaren emaitzak zero balioa du SF: eragiketaren emaitzak balio negatiboa du PF: eragiketaren emaitzak LSB byte du bit bikoitiekin */

salto1:
xor %eax,%eax          # zero emaitza -> ZF eta PF aktibatzen ditu baina desaktibatu egiten ditu
CF,OF,SF
inc %eax                # ZF eta PF
neg %eax 2-              desaktibatzen ditu # SF,PF eta CF aktibatzen ditu: definición de la máscara de campo
ren osagarria :(0-N) shr # Shift Ecuadorean: desplazamiento logico: n bits desplazados
$1,%eax helmuga-       # Shift Ecuadorean: desplazamiento logico: n bits desplazados
eragilea.
/* Ecuadorean bitak ateratzen dira eta ezkerrean zeroak sartzen dira.
Azken bit-eko irteera CF-n geratzen da.
```

SF=0 MSB INTEL MANUALean zero bat sartu denez: <http://www.cs.nyu.edu/~mwalfish/classes/ut/s13-cs439/ref/i386/SAL.htm> SHR-rako, OF da ezarri jatorrizko eragigaiaren  
ordena handiko bitean.  
OF=MSB=1  
OF bandera 1 biteko aldaketetan bakarrik eragiten du.  
 $2^n$  zatitzearren baliokidea da eskuinera mugitzen banaiz eta  $2^n$  ezkerrera biderkatzea (baliteke gainezka egitea).  
\*/ shl \$1,%eax clc xor

```
%eax,%eax CF,OF,SF
movw          # garbitu CF -> CF=0
$0xFFFF,%ax addw      # zero emaitza -> ZF eta PF aktibatzen ditu baina desaktibatu egiten ditu
$0xFFFF,
%ax clc movw $0x7FFF,%ax      # MOV-k EZ du inolako banderarik eragiten
addw $1,%ax #aktibatu OF      # aktibatu SF eta CF baina ez OF
baina ez
CF, OF-k gehikuntzan
izandako erroreaz ohartarazten du eta SF aktibatu dela ikus dezakezu
```

#### ## KONPARAZIOKO JARRAIBIDEAK: TEST, CMP

## Egiaztatu 5. posizioan dagoen bita zero den 0x0010 maskararekin posizio hori isolatzen duen ## probak EFLAGS banderei eragiten  
dien AND eragiketa egiten du, baina ez du emaitza helmuga operandoan gordetzen movw \$0xABFF, %ax movw \$0 x0BCF, %bx proba \$0x0010, %ax # AX^0x0010=0x0010=positibo  
-> SF=0, byte baxua=0x10  
bakoitia ->  
PF=0,  
# Eskuliburuak dio -> OF eta CF banderak garbitu dira  
proba \$0xFFFF, %ax # SF=1 AX^0xFFFF=AX= negatiboa delako, byte baxua=AL= berdina  
-> PF=1  
proba \$0b0000000000010000, %bx # SF=0 AX^0x0010=positibo delako, ZF=1 delako  
BX[5] zero da, PF=0

## Egiaztatu aldagai baten balioa 0x00FF balioa baino handiagoa, txikiagoa edo berdina den ## cmp-k SUB eragiketa  
egiten du EFLAGS banderak eragiten dituena, baina ez du emaitza helmuga-eragilean gordetzen

## SUB: Eragiketa oso sinatuen eta sinatu gabekoena emaitza ebaluatzen du eta OF eta CF banderak ezartzen ditu  
## sinatutako edo sinatu gabeko  
ernaitzan gainezka bat adierazteko, hurrenez hurren movw \$0x01FF, %ax movw \$0x0001, %bx movw \$0.  
x00FF, %cx cmp \$0x00FF,  
%ax # AX-0x00FF=0x0100 >  
0 -> ZF=0 eta SF=0, byte  
baxua=00 ->  
PF=1  
cmp \$0x00FF, %bx # BX-0x00FF=0x0001+0xFF01=0xFF02 < 0 -> SF=1, 0x02 bakoitia  
-> PF=0,

```

# sinatu gabeko gainezkatzea -> CF=1, sinatuta ez gainezkatzea OF=0
cmp $0x00FF, %cx # CX-0x00FF=0 -> ZF=1, SF=0, 0xFF par PF=1, CF=0, OF=0

## BALDINTZAKO JAUZIAK

movw $0x01FF, %ax
movw $0x0001, %bx
movw $0x00FF, %cx
cmp $0x00FF, %ax # AX-0x00FF=0x0100 > 0, gero ZF=0 eta SF=0, 0x00 for ->
PF=1
jg jauzi4           # salto handia -> sinatutako zenbakien kenketa -> SF=0 eta jauzi
ezetz
jump4: cmp $0x00FF, %bx # BX-0x00FF=0x0001+0xFF01=0xFF02 < 0, gero ZF=0
SF=1,
          # CF=1 baino gehiago sinatu gabe eta OF=0 baino gehiago
jl jump5 # jauzi gutxiago -> sinatutako zenbakien kenketa -> SF=1 eta salto
ezetz
jump5: movw $0x8000, %ax # 0x8000 -32768 sinatuta eta 32768 sinatu gabe
cmp $0x0001, %ax # Sinatuta ->0x8000 - 0x1 = 0x8000+0xFFFF=0x7FFF >0 ->
SF=0,
          # OF=1 bi negatiboren batura positiboa izan baita
          # CF=0 0x01FF bitar hutsean geroztik
0x00001=0x01FE, gainezkarik gabe
          # 0xFF bikoitia da -> PF=1

jauzia6           # gainetik salto -> kenketa sinatu gabeko zenbakien -> 32768-1>0
ezetz
jump6: cmp $0x00FF, %cx # CX-0x00FF = 0, gero ZF=1 eta SF=0
jeje salto7 # jauzi berdina
ezetz

## IRTEN

jump7: mugitu $SYS_EXIT, %eax
mov $SUCCESS, %ebx
int $0x80

.amaiera

```

### 15.4.3. Konpilazioa

- Saio guzietan ohikoak diren **bilketa-** prozesuaren urratsak jarraitu .
  - ÿ gcc -m32 -g -o jauziak jauziak.s

### 15.4.4. Exekuzioa

- ./jauziak
- oihartzuna \$?

### 15.4.5. Iturburu-moduluaren azterketa

- Irakurri [Minimalist Example Programa](#) erreferentzia azkarreko orriean

#### Egitura

- Programaren egitura elementu hauek osatzen dute:
  - ÿ Goiburua
  - ÿ Makroen definizioa
  - ÿ Datuen Atala
  - ÿ Argibideen Atala

### 15.4.6. Urratsez urrats exekuzioa

#### Hasierako Eragiketak

- Konpilatu programa araztaileak eskatzen duen sinboloen taula sortzeko aukerarekin eta sortu modulu bitar exekutagarria:
- ÿ `gcc -m32 -g -o jauziak jauziak.s`
- Ireki GDB arazketa, kargatu modulu bitar exekutagarria eta egiaztatu taula kargatuta dagoela. modulu bitar exekutagarriaren ondoan dauden sinboloak.

ÿ `gdb`  
 ÿ `fitxategi-hausteak`  
 ÿ `informazio iturriak`

- Komandoen registro historikorako fitxategia konfiguratu.

ÿ `aktibatu arrasto-komandoak`  
 ÿ `ezarri registro-fitxategia Saltos_gdb_asm.txt`  
 ÿ `ezarri saioa hasteko`  
 ÿ `shell ls -l salts_gdb_asm.txt`

- Aktibatu eten puntu bat programa sartzeko instrukzioan.

ÿ `b _hasi`

- Exekutatu programa programaren lehen adierazpenean geldituz.

ÿ `korrika`

#### EFLAGS Erregistroa

- `p $eflags` ÿ bandera aktiboen izenak inprimatzeko
- egiaztatu banderaren egoera eragiketa logiko, aritmetiko eta TEST instrukzioekin eta CMP.
- Banderak pauso moduan aztertzeko, erabili GDB araztailearen `watch` komandoa

(gdb) `watch $eflags` - exekuzioa eteten du eta EFLAGS erregistroaren edukia erakusten du bere balioa aldatzen den bakoitzean.  
 (gdb) `info watch`: definitutako erlojuak bistaratzen (gdb)  
 eten-puntuak ezabatu: etenaldiak, erlojuak, etab.

## Jauziak

- Egiaztatu jauzia EFLAGS erregistro-banderen egoera eta interpretazioarekin exekutatzen den ala ez. FLAGS interpretatzeko ariketa bat da ý Jauzi instrukzioa exekutatzen denean, PUZak erabaki behar du jauzi egin edo ez FLAGS interpretatzu. Jakingo al genuke nola ERLAZIOATU ja baldintza (GOIAGO baldin bada salto) BANDERAK irakurri eta GOIKO baldintzan konparatzen diren eragigaien balioak irakurri gabe?

## 15.5. Mnemoteknia erabilia

- Ikusi kapitulua "Muntaketa-programazioa: Oinarritzko Mnemoteknia (Azalduta)"

# 16. kapitulua. Sistema eragilerako deiak (kernel)

## 16.1. Sarrera

### 16.1.1. Zer dira sistema-deiak

- HWa OS Kernel-ek babestuta dago eta, beraz, muntatzaile-programatzaleak HWra zeharka sartzen du "Sistema-deien" bidez, Sistema Eragileari sarrera/irteera eragiketak eskatuz. Beraz, teklatura eta monitoreoa sartu nahi baditugu, beharrezkoa izango da nuklearako deiak egitea.
- Programatzalearen eta OS kernelaren arteko interfazearen definizioa: System V Application Binary Interface: SysV-ABI
  - ÿ Mihiztadura hizkuntzak C hizkuntzarako ABI estandarra jarraitzen du.
- Script honetan i386 arkitekturaren sistema-deiak lantzen dira.
- Sistema-deiak mutaia-kodetik:
  - ÿ zuzenean `int $0x80` instrukzioarekin
  - ÿ zeharka libc liburutegi estandarraren funtzioen bidez dei- `instrukzioarekin`
  - ÿ Asanblada-kodean argumentuak pasa behar dira deia exekutatu aurretik `deitu`

### 16.1.2. Sistema deien eskuliburuak

- Zerrenda sistema-deien izenekin: `man syscalls`
  - ÿ Sistema-deiaren irteera: `man 3 irteera`
    - ÿ sistema deien funtzioa deskribatzen du
    - ÿ C hizkuntzan konpilatzeko beharrezko den liburutegiko goiburuan izena zehazten du.
    - ÿ funtzioak behar dituen parametroak eta horiek pasatzen diren ordena zehazten ditu.
  - ÿ Sistema-deia idazteko: `gizon 2 idazteko`

### 16.1.3. Dei-kodeak

- Sistema deien kodeak x86-32 arkitekturan:
  - ÿ `/usr/include/asm/unistd_32.h`
  - ÿ `/usr/include/x86-64-linux-gnu/asm/unistd_32.h`
  - ÿ deitik irten ÿ 1. kdea
  - ÿ irakurri ÿ 3
  - ÿ idatzi ÿ 4
- Dei-kdea EAX erregistrotik pasatzen da.

### 16.1.4. Nola pasa argumentuak zuzenean Kernelera

- Pilatik behera pasatzen diren erabiltzaileen azpirrutinen deien argumentuek ez bezala, sistema eragilearen azpierrutinen deien argumentuek erregistroak erabiltzen dituzte argumentuak pasatzeko memoria gisa.

- Lehenengotik seigarrenera bitarteko parametroak erregistroei dagozkie: EBX, ECX, EDX, ESI, EDI, EBP
- Itzultzeko balioa: EAX

#### 16.1.5. Nola pasa argumentuak zeharka libc funtzioen bidez

- ASMko iturburu-modulu batetik
- Parametroak pilaren bidez libc funtzioetara pasatzen dira eta, beraz, libc bilgarrietara ere pasatzen dira. C liburutegia.
- Itzultzeko balioa: EAX

#### 16.2. IRAKUR NAIZ

- Praktikaren gidoia eta Programazioaren 3. kapitulua Ground-Up Booketik irakurtzea.
- Oharrak eta Testu liburua
- Txostenaren dokumentazioa: Txostenaren edukia eta formatua
- Ebaluazioa: ebaluazio sistema
- Garapen Plataforma: ordenagailu pertsonalaren konfigurazioa
- Programazioa: metodologia

#### 16.3. Gaiak

- Aukerako “Praktikaren Autoebaluazioa”: Praktikak: Galdetegia

#### 16.4. Deitu Irteera

##### 16.4.1. Iturburu-modulua editatzea:salida.c/salida.s

- gcc -m32 -g -o irteera irteera.c

```
#include <stdlib.h>
void main (void)

{ irten (0xFF); }
```

- gcc -m32 -o irteera irteera.c

```
/* C-ren sistema-deia
Prototipoa:           int syscall (int zenbakia, ...);
man syscall */
```

```
#define _GNU_SOURCE
#include <unistd.h>
#include <sys/syscall.h>

void main (huts)
```

```
{
    syscall(__NR_irten,0xFF);
}
```

- gcc -m32 -g -nostartfiles -o irteera irteera.s

```
.global _hasiera
.atala .testua

_hasi:
    push    $0xFF      #itzulerakodea
    deia    irteera    #libc liburutegia
    .amaiera
```

- gcc -m32 -g -nostartfiles -o irteera irteera.s

```
.global _hasiera
.atala .testua

_hasi:
    push    $0xFF #itzulerakodea
    push    $1 #irteteko syscall kodea
    call    syscall #libc liburutegia
    .amaiera
```

- gcc -m32 -g -nostartfiles -o irteera irteera.s

```
.global _hasiera
.atala .testua

_hasi:
    mov     $1,%eax    #irten
    mov     $0xFF,%ebx   #argudioa
    int    $0x80        #sistema-deia
    .amaiera
```

## 16.5. Deitu C liburutegira muntaketa-kodetik

### 16.5.1. inprimatu.s: inprimatuf

- inprimatu.s

```
.atala .datuak

planeta:
    ý .luzea 9          # planeta aldakorra

    .atala .rodata

mezua:
    .asciz "Planeta kopurua %d da \n" #kate formatua
```

```

printf funtzioa

.global _hasiera
    .atala .testua

_hasi:
    ## pantailara inprimatu
    push planet # printf funtziaren 2. argumentua
    push $message # printf funtziaren 1. argumentua : katearen helbidea
    deitu printf
    ## Saioa amaitu

bultzatu $0
dei irteera

```

ŷ Gcc-rekin konpilatzea: ez da beharrezko libc objektu-moduluua estekatzaiareni adieraztea, honen bidez lotzen baitu. akatsa.

- ŷ `gcc -m32 -g -nostartfiles -o print print.s`
- ŷ As eta ld-ekin konpilazioa
  - ŷ `as --32 -gstabs -o print.o print.s`
  - ŷ `ld -melf_i386 -dynamic-linker /lib32/ld-linux.so.2 -o print print.o -lc :`  
estekatu libc objektu moduluarekin

## 16.6. Sistema-deiak AMD64 arkitekturan

- Libc liburutegi estandarraren funtziotara deiak.
- ŷ libc funtzioen prototipoen eskuliburua GNUn eskuragarri dago man komandoarekin. Aupa **gizona idatzi**
- ŷ Deia exekutatu aurretik argumentuak pasa behar dira instrukzioa erabiliz deitu.
- ŷ Parametroak diren %rdi, %rsi, %rdx, %rcx, %r8 eta %r9 erregistroetatik pasatzen dira. libc funtziaren argumentuekin lotzen da ezker-eskuin norabidean.
- Gorde erregistroak: %rbp, %rbx eta %r12tik %r15era
- Itzulteko balioa: %rax, %rdx sekuentziako bi erregistro libreetako bat.
- Adibidea:

```
#include <stdlib.h>
```

```
irten (0xFF)
```

xor %rax	#berrezarri RAX
mov \$0xFF, %rdi	#itzuleria kodea
deitu irteera	#libc liburutegia

mov 60 \$, %rax	#irten
mov \$0xFF, %rdi	#itzuleria kodea

syscall

# 17. kapitulua. Azpirrutinak

## 17.1. Sarrera

### 17.1.1. Helburuak

#### Programazioa

- Azpirrutinaren kontzeptua AT&T x86-32 Mihiztadura Lengoaian
- Deitu eta itzultzeko argibideak: deitu eta atzera
- Argudioak: pilaren erabilera
- Pilatzeko argibideak: push eta pop
- Pila-egitura: pilaren beheko eta goialdeko erakusleak: EBP eta ESP erregistroak.
- Deien habia: pila "Frame" segmentuetan banatzea.
- Deitzeko konbentzioa: Pasatzeko parametroak, Itzultzeko balioa, Itzultzeko helbidea, Pila, Markoa pila, Erakusleak pilatzeko Markoa, Epiloga, Prologoa
- Zuzentaraauak: .mota sumMtoN, @funtzioa

#### Analisia

- Pila-analisia GDB araztailea erabiliz
  - ÿ behatu fotograma berri baten sorrera
  - ÿ Erakusle-erregistroen bidez markoaren mugak identifikatzea.
  - ÿ bota azpirrutinaren argumentuak, itzultzeko helbidea eta itzultzeko balioa pilara.

## 17.2. Iturria Modulua

- SumMtoN.s iturburuko moduluak \_start errutina nagusitik sumMtoN azpierrutinara egiten du deia. bi argumentu pasatuz eta baturaren emaitza jasoz.
- SumMtoN azpierrutinak M osotik N osoko batuketa egiten du non N>M.

## 17.3. Baldintzak

- Ordenagailuaren egituraren oinarrizko kontzeptuak.
- Intel x86-32 oinarrizko arkitektura.
- AT&T mihiztataile-lengoaian programatzea: datuekin praktikatu, helbideratze moduak eta oinarrizko transferentzia, aritmetika eta jauzi-argibideak.

## 17.4. IRAKURRI

- Praktikaren gidoia eta Programazioaren 3. kapitulua Ground-Up Booketik irakurtzea.
- [Oharrak eta Testu liburua](#)
- [Txostenaren dokumentazioa:](#) Txostenaren edukia eta formatua
- [Ebaluazioa:](#) ebaluazio sistema
- [Garapen Plataforma :](#) ordenagailu pertsonalaren konfigurazioa

- [Programazioa](#) : metodologia

## 17.5. Gaiak

- Aukerako "Praktikatu Autoebaluazioa" : Praktikak: Galdetegia

## 17.6. Datuen tamaina eta aldagaiak

### 17.6.1. Algoritmoa

- Garatu programa bat i386 arkitekturako mihiztadura-lengoaian, zeinaren emaitza hasierako deskribapen [programazio-metodoa](#) pseudokode-lengoaian eta organigraman erabiltzen duen batura egiten duena. Programak bi modulu izan behar ditu: \_start izenarekin erreferentziatutako modulu nagusi bat eta batura egiten duen sumMtoN izeneko azpierrutina. Programa nagusiak M eta N parametroak azpierrutinara pasatzen ditu, batuketa egin ondoren, batuketaren emaitza azpierrutinaren itzulera balio gisa itzultzeko.

### 17.6.2. Iturburu-moduluaren edizioa: sumMtoN.s

- Deskargatu "sumMtoN.s" iturburu-modulua myaulariotik eta gehitu komentario egokiak.

```
/*
Programa: sumMtoN.s
Deskribapena: M,M+1,M+2,M+3,...N serieko zenbaki osoen batuketa egiten du: sumMtoN(1. arg=M, 2. arg=N) non M < N
Exekuzioa: editatu M eta N balioak eta kopiatatu programa.
Execute $./sumMtoN
Batuketaren emaitza sistema eragilek ateratzen da linux komandoarekin: echo $?
```

```
gcc -nostartfiles -m32 -g -o sumMtoN sumMtoN.s Asanblada gisa
--32 --gstabs sumMtoN.s -o sumMtoN.o estekatzalea -> ld -melf_i386 */
-o sumMtoN sumMtoN.o
```

```
## MAKROAK
1 .equ ##      SYS_EXIT,
DATU
.atala .datuak

## JARRAIBIDEAK .atala .testua

.globl _start _start: ##
M eta N
bi argumentuak azpierrutinara pasatzen ditut pila pushl $10 #push bigarren argumentua -> N
pushl $5 #push lehen argumentua -> M

## Deitu sum1toN azpierrutinara deitu sumMtoN
```

```
## Sum11toN-ren irteera argumentuari pasatzen diot exit() sistema-deia mov %eax, %ebx # (%ebx itzultzen  
da)  
## OS deien kodea movl $SYS_EXIT, %eax # irten deia ## Eten  
OS int $0x80
```

```
/*  
Azpirrutina: sumMtoN
```

Deskribapena: 1. gehigarritik 2. gehigarria segidan zenbaki osoen batura kalkulatzen du Sarrerako argumentuak:  
1. gehigarria eta 2.

gehigarria argumentuak errutina nagusiak pasatzen ditu pilatik: 1.a  
azken argumentua pilatzen da eta azkenik 1. argumentua pilatuta dago.

Irteerako argumentua: batuketaren emaitza da eta errutina nagusira pasatzen da EAX erregistroaren bidez.

Aldagai lokalak: aldagai lokal bat implementatzen da pilan baina ez da erabiltzen \*/ .type sumMtoN, @function  
#

sumMtoN etiketa sumMtoN deklaratzen du:

```
## Hitzaurrea: Sortu pila-marko berria pushl %ebp #gorde  
zaharra marko-erakuslea movl %esp, %ebp #eguneratu marko-erakuslea
```

```
## Erreserbatu hitz bat pilean aldagai lokal gisa  
## Aldagai lokala kanpoko memorian: sum subl $4, %esp
```

```
## Argumentuen harrapaketa movl  
8(%ebp), %ebx #1. argumentua %ebx movl 12(%ebp), %ecx #2. argumentua  
%ecx-ra kopiatu da
```

```
## 1. argaren balioaren eta 2. argaren balioaren arteko sekuentzia gehitzen du ## 1. arg < 2. arg  
## EDX aldagai lokal gisa  
erabiltzen dut aldagai lokal baterako kanpoko erreserbaren ordez: abiadura optimizatzen du
```

```
## Tokiko aldagai batuketa movl $0,%edx hasieratzen dut
```

```
## Iterazio kopurua  
mov %ecx,%eax azpian  
%ebx,%eax
```

begizta:  
gehitu %ebx,%edx inc  
%ebx sub  
\$1,%eax jns loop

```
## Gorde batura emaitza itzultzeko balio gisa
```

```
    movl %edx, %eax

## Epilogoa: %ebp, %esp popl %ebp, fotograma zaharra
berreskuratu          #berreskuratu pila-erakuslea
                      #berreskuratu marko-erakuslea

## Itzuli errutina nagusira ret .end
```

#### 17.6.3. Konpilazioa

- Saio guztietan ohikoak diren bilketa-prozesuaren urratsak jarraitu.

ÿ gcc -nostartfiles -m32 -g -o sumMtoN sumMtoN.s

#### 17.6.4. Exekuzioa

- ./sumMtoN
- oihartzuna \$?
- Egiaztatu behar bezala funtzionatzen duela parametroen balioak aldatuz: batuketaren 1. balioa eta Baturaren 2. balioa.

#### 17.6.5. Iturburu-moduluaren azterketa

- Irakurri Minimalist Example Programa erreferentzia azkarreko orriean

#### Egitura

- Programaren egitura elementu hauek osatzen dute:
  - ÿ Goiburua
  - ÿ Makroen definizioa
  - ÿ Datuen Atala
  - ÿ Argibideen Atala

#### Urratsez pauso exekuzioa GDB arazketa erabiliz

- Konpilatu programa araztaileak eskatzen duen sinboloen taula sortzeko aukerarekin eta sortu modulu bitar exekutagarria:

ÿ gcc -nostartfiles -m32 -g -o sumMtoN sumMtoN.s
- Ireaki GDB arazketa, kargatu modulu bitar exekutagarria eta egiaztatu taula kargatuta dagoela. modulu bitar exekutagarriaren ondoan dauden sinboloak.

ÿ gdb

ÿ exekutagarri\_modulu fitxategia

ÿ informazio iturriak
- Komandoen registro historikorako fitxategia konfiguratu.

ÿ aktibatu arrasto-komandoak

ÿ ezauri sumMtoN\_gdb\_asm.txt registro-fitxategia

ÿ ezarri saioa hasteko

ÿ shell ls -l sumMtoN\_gdb\_asm.txt

- Aktibatu eten puntu bat programa sartzeko instrukzioan.

ÿ b \_hasi

- Exekutatu programa programaren lehen adierazpenean geldituz.

ÿ korrika

- Programaren instrukziorik exekutatu gabe

ÿ Bateriaren egoera

ÿ Pilaren goiko aldea: x \$esp edo x \$sp: pila-erakuslea

ÿ Markoaren behealdea: x \$ebp edo x \$fp: marko-erakuslea

ÿ Pilaren goiko edukia (sp helbidea): argc: exekutatzen ari den komando-lerroaren kate argumentu kopurua.

ÿ x /xw \$sp

ÿ Edukia pilaren goiko aldean dagoen posizio bat (helbidea sp+4): argv[0]: 1. katearen helbidea  
exekutatzen ari den komando-lerrotik

ÿ p /s \*(char \*\*)(\\$sp+4)

- Exekutatu beharrezko lerroak azpierrutinan sartu arte:

ÿ Step komandoa: s n komandoak ez baitu azpierrutinan sartzen, baizik eta guztiz exekutatzen du.

ÿ Nora jartzen du sp-erakusleak? Zein informazio dauka sp-ak nora?

ÿ x /i \*(int \*)\$sp : zer instrukzio da?

- Azpierrutinen prologoa exekutatu

ÿ Marko berria

ÿ Fotograma erakuslearen balio berria: p \$fp

ÿ Pilatzeko erakuslearen balioa: p \$sp

ÿ Itzulpen-helbiderako sarbidea marko-erakuslea erreferentzia gisa hartuta: x /i \*(int  
\*)(\$fp+4)

- Exekutatu azpierrutina itzulera-balioa lortu arte.

ÿ Inprimatu itzuleraren balioa: p \$eax

- Azpierrutinaren epilogoa exekutatu

ÿ Markoaren erakuslearen balioa: p \$fp

ÿ Pilatzeko erakuslearen balioa: p \$sp

ÿ Itzultzeko helbidea: x \*(int \*)\$sp

- Exekutatu itzulera adierazpena

ÿ Pila-erakuslearen helbidea: p \$sp

ÿ Zergatik aldatu da pila-erakuslearen helbidea?

# 18. kapitulua. Irudiak: Bit-mapa eramangarria

## 18.1. Sarrera

ÿ Azken aztertuan ~~erakundearen bidezko banaka~~, **banaka** egitea komeni da

- Praktikaren helburua funtzi baten baliokide den muntatzaile-lengoaian azpierrutina bat garatzea da C-ren BMP formatuan irudiak sortzeko aplikazio baten barruan.
- C hizkuntzako lehen lau ariketak era gidatuan egingo dira irakaslearekin eta gainerako ikasleekin. banaka .

## 18.2. Aplikazio

### 18.2.1. Fitxategiak barne

- Deskargatu bmp\_practica6.zip fitxategia eta atera fitxategiak.
- Gidoiak:
  - ÿ **comp\_ejec\_vis.sh** : konpilatu, exekutatu eta bistaratzeko atazak automatizatzen dituen scripta deitzu Makefile gidoia.
  - ÿ **Makefile\_C** - C iturburu-programaren konpilazio-ataza automatizatzen duen scripta.
  - ÿ **Makefile\_pixels\_as**: C eta asm iturburuko moduluak muntatzeko, konpilatzeko eta lotzeko zeregina automatizatzen duen scripta.
  - ÿ **IRAKURRI.txt** : iturburu-programa desberdinak C eta mihiztzaile-lengoaian editatzeko jarraibideak eta programa horiek kopiatzeko izen egokiekin comp\_ejec\_vis.sh **scripta** exekutatu aurretik.
- Iturburu-moduluak:
  - ÿ bitmap\_gen\_test.c: 512x512 bitmapako irudi bat sortzen du BMP formatuan eta fitxategian gordetzen du. proba.bmp.
  - ÿ square\_128x128.c: DIMENSIONxDIMENSION bitmap irudi bat sortzen du BMP formatuan eta test.bmp fitxategian gordeta.
  - ÿ squares\_4.c: BMP formatuan habiaraturiko lau bit-mapa laukizuzen sortzen ditu eta irudia gordetzen du. test.bmp fitxategia.
  - ÿ bmp\_funcion.c: bitmap\_gen\_test.c-tik hasita, pixel-sorgailuaren begizta pixels\_generator(xcoor,ycoor,top,buffer) funtziok definitzen du.
  - ÿ bmp\_as.c: Garatu beharreko modulua sartu gabe.
  - ÿ pixels.s: Garatu beharreko modulua sartu gabe.
- Fitxategia
  - ÿ proba.bmp

### 18.2.2. Konpilazio automatikoaren adibidea

- Kopiatu **cp Makefile\_C Makefile**
- Kopiatu **cp squares\_4.c bmp\_imagen.c** eta exekutatu **comp\_ejec\_vis.sh**
- Kopiatu **cp bitmap\_gen\_test.c bmp\_imagen.c** eta exekutatu **comp\_ejec\_vis.sh**
- **Comp\_ejec\_vis.sh** eta **Makefile** script-ak interpretatu

### 18.3. BMP formatua

#### 18.3.1. Pantaila: pixela

- Ordenagailuaren pantaila bi dimentsioko pixelez osatuta dago, non pixel bakoitza pantaila programagarriko puntu diskretu batean kokatuta dagoen. Fisikoki, ordenagailuko monitorea hainbat motatakoa izan daiteke: CRT, LCD, LED, OLED, Plasma, etab. Ikupegi funtzional batetik, demagun LED motako monitore bat non pixel bakoitza 3 LED diodoz osatuta dagoen: LED urdin bat, beste LED berde bat eta hirugarren LED gorria. Oinarritzko 3 koloreen konbinazioak kolore posible guztiak sortzen ditu.  
Pixel baten dimentsioak, hau da, 3 LED, ehunka mikrometrokoak dira. 15,6" pantaila duen ordenagailu eramangarri batek 15" zabal eta 12" altu ditu. xdpinfo | grep dimentsioak: txostenak: 1366x768 pixel (361x203 milimetro=73283 mm<sup>2</sup>) ÿ73283mm<sup>2</sup> /(1366x768 pixel)=0,0707mm<sup>2</sup> / pixel = µm /pixel , hau da, 265µm-ko karratu bat alde bakoitzean.

#### 18.3.2. Kodetzea

- BitMapPortable (BPM) formatua irudi-formatu eskalar bat da, hau da, pixel bakoitzaren datuak biltzen ditu RGB kolore-osagaien intentsitatea kodetuz pantailan agertuko diren moduan.
- BPM formatuan, pantailaren pixel-matrizea bideo-txartelaren memorian gordetako errenkadak (ardatz horizontala) eta zutabeak (ardatz bertikala) 2D bi dimentsioko matrize motako datu-egitura bati lotuta daude. Arrayaren koordenatuen jatorria beheko ezkerreko izkina da. 2D arrayaren bikote bakoitza (x,y) pixel baten koloreari dagokio.
- True Color: matrizeko elementu bakoitzak 3 eremuz osatutako datuak ditu, non eremu bakoitzak kolore bat adierazten duen (Urdina-Berde-Gorria) eta byte bat hartzen du. RGB kolore-osagai bakoitza byte batekin kodetzen da, eta horrek adierazten duen byte batekin kodetzen da. kolorea . Adibideak:
  - ÿ RGB: 0xFF-0x00-0x00 ÿ % 100 pixel gorria eta intentsitate maximoa.
  - ÿ RGB: 0xFF-0x00-0xFF ÿ pixel % 50 gorria eta % 50 urdina ÿ kolore morea.
  - ÿ RGB: 0x00-0x00-0x00 ÿ kolorerik eza ÿ kolore beltza
  - ÿ RGB: 0xFF-0xFF-0xFF ÿ kolore primarioen proportzio bera ÿ kolore zuria
  - ÿ RGB: 0x7F-0x7F-0x7F ÿ kolore primarioen proportzio bera ÿ grisena eskala beltza (00-00-00) eta zuriaren (FF-FF-FF) artean
- 512 x 512 pixeleko tamainako irudi batek 512 x 512 pixel x 3ko array bat sortuko du  
byte/pixel = 786432 byte = 768KB

#### 18.3.3. Memoria mapa: Monitorea ÿ Buffer

- MxN2D matrizearen koloreak memoria linealean idatziz non memoria helbide bakoitza a den byte, datu-egitura edo buffer-ak bere osagaiak honela banatuta ditu:
- ÿ F0C0BGR-F0C1BGR-...-F0C\_(N-1)BGR-F1C0BGR-...-F1C\_(N-1)BGR-...-F\_(M-1)C0BGR-F\_(M-1)-2-3-4-5-...-(MxNx3-1) posizio erlatiboei dagozkien 1)C1BGR-...-F\_(M-1)C\_(N-1)BGR. Non F0 0 errenkada den, C0 0 zutabea eta BGR 3 byteko Urdin-Berde-Gorria den sekuentzia.

ÿ memoria buffer-aren luzera osoa: MxNx3 byte

ÿ F0C0BGRren interpretazioa:

\*\*\* Row Zero Column Zero-ko BGR pixela \*\*\* BGR: 3 byte  
urdin-berde-gorri ordenan.

\*\*\* Byte urdinak 0 posizio erlatiboa hartuko du buffer barruan, byte berdeak 1. posizioa eta byte gorriak 2. Posizioak bufferaren lehen bytearen helbidearekiko.

ŷ F0C1BGR:

byte urdina buffer barruan 3. posizioan dago byte berdea 4. posizioan  
dago buffer barruan byte gorria 5. posizioan dago buffer barruan

ŷ Monitorearen zero errenkada: F0C\_(N-1)BGR: Bufferean, byte urdina  $3*(N-1)$  posizioan dago, berdea  $3*(N-1)+1$ ean eta gorria  $3 * (N-1)+2$ .

ŷ Monitorearen lehen errenkada: F1C0BGR: byte urdina ŷ  $3*N$  posizioa

ŷ Monitorearen lehen errenkada: F1C\_(N-1)BGR: byte urdina ŷ posizioa  $3*N+3*(N-1)$

ŷ F\_iC\_jBGR:

byte urdina -> posizioa  $3*N*i+3*j$  non  $0 < i < M$  eta  $0 < j < N$  byte berdea ->  
posizioa  $(3*N*i+3*j)+1$  non  $0 < i < M$  eta  $0 < j < N$  byte gorria -> posizioa  $(3*N*i+3*j)$   
+2 non  $0 < i < M$  eta  $0 < j < N$

#### 18.3.4. BMP fitxategia

- BMP formatuko irudiak "\*.bmp" lizapena duten fitxategietan gordetzen dira, esate baterako, "test.bmp"
- BMP fitxategiak, datuen bufferaz gain, testuinguru honetan azaltzeko egokia ez den metainformazioa duen goiburu bat dauka.

### 18.4. Moduluaren iturria bitmap\_gen\_test.c

#### 18.4.1. Deskribapena

- bitmap\_gen\_test programak 2D-ko pixel-matrize bat sortzen du eta test.bmp fitxategian gordetzen du.  
ŷ Arrayaren dimentsioak bitmap\_gen\_test.c iturburu-programan definitzen dira

#### 18.4.2. Programaren Ezaugarriak

nagusia ()

- Funtzio nagusia
- Blokearen deskribapena:

RGB\_data buffer[512][512]: eginda dagoen aldagai lokala deklaratzen da eta 2D "buffer" pixel-matrizea gordetzen du non pixel bakotza RGB\_data motakoa den.

RGB\_data aldagai mota: 3 byte jarraian non lehena urdinaren intentsitatea den, bigarrena berdea eta hirugarrena gorria. Intentsitateak sinatu gabeko zenbaki osoak dira. Adibidez: 0x7F-0xFF-0x00 % 50eko intentsitate urdina, % 100eko intentsitate berdea eta % 0ko intentsitate gorria adierazten du.

memset(buffer, 0, sizeof(buffer))

- Liburutegi estandarraren libc ŷ man memset ŷ string.h goiburuan deklaratzen da.

- "Buffer" pixelen 2D matrizea 0-ra hasten du.
- Ikus ezazu adibide gisa bitmap\_gen\_test.c moduluko memset() funtziaren deia

bmp\_generator("./test.bmp", 512, 512, (BYTE\*)buffer)

- Sortu test.bmp fitxategia eta idatzi fitxategi horretan izendun pixelen 2D arrayaren edukia. buffer.
- Buffer argumentua pasatzeko beharrezkoa da casting-a egitea (BYTE\*). Ikusi BYTE motaren deklarazioa bitmap\_gen\_test.c moduluan

RGB\_datuen egitura

- Ikusi RGB\_data motaren deklarazioa bitmap\_gen\_test.c moduluan.

begizta bikoitza

- for begizta:
  - ÿ i aldagaia errenkada-indizea da eta j aldagaia zutabe-indizea.
  - ÿ buffer[i][j].b: pixelaren byte urdina (i,j) posizioan
  - ÿ buffer[i][j].g: pixelaren byte berdea (i,j) posizioan
  - ÿ buffer[i][j].r: posizioko pixelaren sare bytea (i,j)
  - ÿ Array-buffer-aren elementu bakotza[i][j]-k deklaratutako RGB\_data datu-egitura bat da.  
b,g eta r elementuekin programatzalea.

## 18.5. Ariketak

ÿ Mihiztaduraphantazkotzepenatutinaz osatutako azken ariketa izan ohi da oinarria

### 18.5.1. C programazioa

- Irakurri programazio prozedura LEEME.txt fitxategian
- Helburua bitmap\_gen\_test.c jatorrizko programaren main() funtzi nagusia aldatzea da.  
bata bestearengandik independente diren programa ezberdinetara.
  1. - Bitmap\_gen\_test.c programa osatu eta exekutatu
  2. - bistaratu test.bmp fitxategiaren irudia: \$display test.bmp
  3. - Modulu karratua\_128x128.c: Aldatu irudiaren neurriak 128 pixel x 128 pixelera makro DIMENSION=128 editatzu bere balio maximoaren %50eko intentsitatea duen grisa.
  4. - Karratuak\_4.c modulua: Sortu 4 karratu, bata bestearen barruan simetrikoki, non karratu beltz handiena 512x512koa den eta gainerakoak 1/8 bakoitzean murrizten diren. Ez erabili ctes C adierazpenetan, erabili makroak x\_coor, y\_coor, top for-en hasierako balioa eta errenkada eta zutabeen gehienezko posizioa (goian) adierazteko. Kolore karratuak: atzoko planoa (00-00-00)/(FF 00-FF)/(00-FF-FF)/(FF-FF-00)/
  5. - bmp\_funcion.c modulua: main() funtzioaren barruan, ordezkatu ezazu bihurgune karratuaren pixelak hasieratzeko begizta egiten duen kode blokea ezaugarri hauetan funtzi batekin:
    - ÿ prototipoa: void pixels\_generator (unsigned int x, unsigned int y, unsigned int gehienez, RGB\_data reg\_mem[][goian])
    - ÿ x eta y karratuaren koordenatu-jatorria dira (beheko ezkerreko izkina)
    - ÿ maximoa karratuaren koordenatu handiena da.
    - ÿ funtzi-deia: pixels\_generator(xcoor,ycoor,top,buffer);
    - ÿ xcoor=top/8, ycoor=top/8 eta top=512 argumentuek makroak erabiliz definitzen dituzte.

### 18.5.2. Programazioa ASMn

1. - bmp\_as.c modulua: void pixels\_generator (unsigned int maximum, RGB\_data reg\_mem[][top]) funtzia implementatu pixels\_generator azpierrutina mihiztadura hizkuntzan garatuz array\_pixel.s fitxategi berrian. Mihiztadura-hizkuntzako fitxategiak azpierrutina baino ez du edukiko.
  - ÿ Azpierrutinak begizta bikoitza implementatzen du.
  - ÿ Inplizituki, azpierrutinan bertan, x=y=0 argumentuak hartuko ditugu kontuan karratuaren jatorri koordenatua.
  - ÿ Karratuaren pixel guztiak kolore berdina dute, zeinaren intentsitatea aldagai urdinekin definitzen den, berdea eta gorria.

### 18.5.3. GDB

1. bmp\_funcion.c modulurako, adierazi pilaren posizioa non itzultzeko helbidea pixels\_generator azpierrutina, baita fotograma-erakuslearen eta pila-erakuslearen edukia ere.
2. Aurreko atalean bmp\_as.c moduluarekin pixels\_generator azpierrutinarako
3. Interpretatu gdb komandoak kasu honetan:

+fitxategia bmp\_image

bmp\_image-tik ikurrak irakurtzen... eginda. +b nagusia

1. eten-puntu 0x851-n: bmp\_imagen.c fitxategia, 128. lerroa.

+korrika

Hasierako programa: /home/candido/Dropbox/apuntes/apuntes\_Estr\_Computadores/upna /practicas/x86/BMP/bmp\_imagen

1. eten-puntu, nagusia (argc=1, argv=0xffffcb54) bmp\_imagen.c:128 helbidean

+ n

+ n

+s

pixel\_generator (x=64, y=64, maxim=512, reg\_mem=0xffff3ca9c) bmp\_imagen.c:112

+ n

+ n

+ n

+ n

+ n

+ptype buffer

"buffer" ikurra ez dago uneko testuinguruan. +ptype reg\_mem type =

struct { BYTE b;

BYTE g; BYTE r; } (\*)

[512] +amaitu

Exekutatu #0

pixel\_generator-

etik irtetera

(x=64,

y=64, maximum=512, reg\_mem =0xffff3ca9c) at bmp\_imagen.c:114 0x5655588a main (argc=1, argv= 0xffffcb54)

bmp\_imagen.c:139 helbidean

+ n

+ptype buffer

mota = struct {

BYTE b;

BYTE g;

BYTE r; } [512]

[512] +x /xb

&buffer 0xffff3ca9c:

0x00 +p /a &buffer \$1

= 0xffff3ca9c +x /

x &buffer

0xffff3ca9c: 0x00 +x /

x (\* RGB\_data)&buffer Sintaxi-

erreba bat adierazpenean, ")&buffer-aren ondoan". +x /x (RGB\_data

\*)&buffer 0xffff3ca9c: 0x00 +x /3xb

(RGB\_data \*)&buffer

0xffff3ca9c: 0x00 0x00

0x00

+x /3xb &buffer

0xffff3ca9c: 0x00 0x00 +x /3xb 0x00

&buffer[0][0]

```

0xffff3ca9c: 0x00          0x00      0x00
+p /x buffer
balioak 786432 byte behar ditu, hau da, gehienezko balio-tamaina +p /x
buffer[0][0] $2 = {b =
0x0, g = 0x0, r = 0x0} +p /x buffer[0][0].r
$3 = 0x0 +p /x *(char
*)buffer[0]
[0]
Ezin da memoria sartu 0x0 +p /x *(char *)&buffer[0][0] $4 = 0x0
+p /a (char *)&buffer[0][0] $5 =
0xffff3ca9c
+p /a (char *) )&buffer[0][1] $6 =
0xffff3ca9f +p /a
&buffer[0][1] $7 = 0xffff3ca9f +p /
a &buffer[0][1].b
$8 = 0xffff3ca9f +p /a
&buffer[0][1].1.g
$9 = 0xffff3caa0 +p /a
&buffer[0][1].r $10
= 0xffff3caa1 +p /a
&buffer[64][64].b
$11 = 0xffff54b5c +p /a
buffer[ 64][64] .b
$12 = 0xff +p buffer[64][64].b
$13 = 255 '\377'
+p /x buffer[64][64].b $14
= 0xff +p /x
*( (char *)&buffer
+64*512+64) $15
= 0x0 +p /x buffer[512]
[512].b $16
= 0x3d +p /x buffer[511][511].b $17 = 0x0
+p / x
buffer[511] [511] $18 = {b =
0x0, g =
0x0, r = 0x0} +p /a buffer[64]
[64] $19 =
{b = 0xff, g = 0x0, r = 0xff} +
p /x *( (char *)&buffer+64*512+64) $20
= 0x0 +p /x *((char
*)&buffer+64*512+63) $21 = 0x0 +p /x
*((char * )&buffer+64* 512+65) $22 = 0x0
+xx /3xb
((char *)&buffer+64*512+64) 0xffff44adc:
0x00 0x00
+p /a &buffer[64][64] $23 = 0xffff54b5c

```

0x00

```
+p 0xffff54b5c - 0xffff44adc
24 $ = 65664
+x /3xb ((char *)&buffer+3*(64*512+64))
0xffff54b5c: 0xff 0x00 0xff

+(gdb) x /3xb ((char *)&buffer+3*(64*512+64))
ordena mugagabea: "". Saitu "laguntza"
++x /3xb ((char *)&buffer+3*(64*512+64))
ordena mugagabea: «+x». Saitu "laguntza"
+0xffff54b5c: 0xff          0x00      0xff

zehaztu gabeko ordena: "0xffff54b5c". Saitu "laguntza"
+(gdb) x /3xb ((char *)&buffer+3*(64*512+64))
ordena mugagabea: "". Saitu "laguntza"
++x /3xb ((char *)&buffer+3*(64*512+64))
ordena mugagabea: «+x». Saitu "laguntza"
+0xffff54b5c: 0xff 0x00 zehaztu gabeko      0xff

ordena: "0xffff54b5c". Saitu "laguntza"
+p /a (char *)&buffer[0][1]
25 $ = 0xffff3ca9f
+x /3xb (RGB_data *)&buffer+1
0xffff3ca9f: 0x00 0x00 0x00

+irten
```

AZPIRUTINAREN BARRUAN mainan adierazitako motak erabil ditzaket

```
(gdb) ptype RGB_data
mota = egitura {
BYTE b;
BYTE g;
BYTE r;
}
(gdb) p *(RGB_data *)RGB_ptr
$29 = {b = 0 '\000', g = 0 '\000', r = 255 '\377'}
(gdb) p (*RGB_data *)RGB_ptr.r
28 $ = 255 '\377'
```

# VI Erreferentzia Azkar Orriak

# 19. kapitulua. AT&T x86 muntatzaileen programazioa

## 19.1. x86-32 programak

### 19.1.1. Programa Minimalista

- minimalista.s

```
### Programa: minimalist.s ### gcc -m32 -g -o minimalist minimalist.s

.global atal nagusia .testua

nagusia:

    mov $1, %eax # sistema eragilearen dei kodea: azpierrutina irten int $0x80 # sistema eragilearen deia .end
```

#### Programaren egitura

- Goiburu bat eta bi atal:
  - Goiburua iruzkinekin.
  - Datuen Atala: hasierako aldagaiak ezartzeko erreserba egiten da
    - Zuzentaraua .atala .datuak edo soilik .datuak
  - Argibideen Atala: Mihiztagailuaren jarraibideen sekuentzia
    - Zuzentaraua .atala .testua edo soilik .testua

#### Oheburua

- Goiburua honi buruzko iruzkinekin:
  - Programaren izena, programazio-lengoaia.
  - Programaren deskribapena: programaren sarrerak, irteerak, programaren funtzioa.
  - Programazio-ingurunea: sistema eragilea, erabilitako muntatzailea, konpilazio komandoak, mutaia, lokailu.
  - Egileari, datari eta abarri buruzko iruzkinak.

#### Datuen Atala

- .section .data direktiba : datuen atalaren hasiera adierazten du
- Label n: :memoria erreserba n helbide sinbolikoan
- .int zuzentaraua : 4 byteko erreserba n helbidean hasita: n,n+1,n+2,n+3 helbideak
- 5 literal : memoria-erreserbaren hasierako balioa

Argibideak Atala: sarrera-puntu eta irteera blokea: sistema-deia eta sistema-deia

- Zuzentaraua .section .text : argibideen atalaren hasiera adierazten du.
- Argibide-sintaxia AT&T hizkuntzan:
  - ÿ etiketa: eragiketa iturburu\_operandoa, helmuga\_operandoa #comment
- Sistema eragiletik programara sartzeko puntuatua:

ÿ SISTEMA Eragileak programari edo aplikazioari dei egiten dio.

ÿ \_hasi etiketa: . Seinalatu programaren lehen instrukzioa.

ÿ .global direktiba: \_start etiketak sum1toN programatik kanpo "ikusgai" egon behar du, estekatzaileak Linux sistema eragilearekin lotu dezan sarrera-puntu gisa, hau da, gainerako programetarako ikur globala izan behar du eta ez. tokiko bat. sum1toN programara. Ld estekatzaile lehenetsiak sarrera-puntuaren etiketa gisa erabiltzen den ikurra \_hasiera dela suposatzen du.

- Programatik sistema eragilerako irteera puntuatua:

ÿ Beharrezko da sum1toN programaren prozesua hiltzea eta prozesu horrek erabiltzen dituen baliabide guztiak askatzea. Prozesuaren amaierako zeregin hori sistema eragileak edo Linux kernelak egin behar du.

ÿ Programak edo aplikazioak SISTEMA Eragilea deitzen du.

ÿ Sum1toN programak sistema eragileari dei egiten dio prozesuaren amaierako eragiketa egiteko, irteera (argumentua) funtzioa exekutatzuz. Sistema eragileak deituz gero exekutatzen dituen funtzio posibleen zerrenda du. Horrelako funtzio bat irten (argumentua) da.

ÿ Deitu sistema eragilera muntaia hizkuntzan:

ÿ EAX Erregistroa: Sistema Eragileak exekutatuko duen funtzioaren kodea gordetzen du. ren kodea irteera funtzioa 1 da.

ÿ EBX erregistroa: irteera (argumentu) funtzioaren argumentu-kodea gordetzen du. 0 balioa programaren exekuzio zuzen gisa interpretatzten da.

ÿ Instruction int \$0x80: Instrukzio honek sistema eragileari dei egiten dio, sistema eragilea ETEN DU EAX erregistroa gordetako kodearekin lotutako funtzioa exekutatzeko.

ÿ Programazioa oinarritik

Muntaiaren amaiera

- .amaiera zuzentaraua

#### 19.1.2. Oinarrizko Adibidea

- Iturburu-modulu: sum1toN.s

```
### Programa: sum1toN.s ###
```

Deskribapena: serieen batura egiten du 1,2,3,...N ### von IAS makinaren sum.ias-en baliokide den AT&T i386 hizkuntzako programa da. Neumann

```
### gcc -m32 -g -nostartfiles -o sum1toN sum1toN.s ### Asanblada
gisu --32 --gstabs sum1toN.s -o sum1toN.o ### linker -> ld -melf_i386 -o
sum1toN sum1toN.o
```

## Aldagaien adierazpena .atala .datuak

n: .int 5

.global \_hasiera

```

## Kodearen hasiera
.atala .testua

_hasi:
    mov $0,%ecx # ECX-k batura aldagaia ezartzen du
    mov n,%edx

begizta:
    gehitu %edx,%ecx
    azpian $1, %edx
    jnz begizta
    mov %ecx, %ebx # irteerako argumentua OSra EBX bidez

akordioa
    ## irten

    mov $1, %eax # sistema eragilearen deien kodea: irten azpierrutina
    int $0x80 # OS deia

.amaiera

```

- Konpilazioa: gcc -nostartfiles -m32 -g -o sum1toN sum1toN.s
- Muntaketa: as --32 --gstabs sum1toN.s -o sum1toN.o
- estekatzailea: ld -melf\_i386 -o sum1toN sum1toN.o
- Asanblada-itzultzailen zuzentaraauak: .section, .data, .text, .byte, .end, etab... bezalako puntu batekin hasten dira. aurritzka

etiketa: op_mnemonic	eragiketa_iturria	,	eragile_helmuga	;iruzkindu
----------------------	-------------------	---	-----------------	------------

- Etiketak atzizkia dute:
- \_start etiketa: programarako sarrera-puntua da. Nahitaezkoa. Lokatzaileak erabiltzen du.
- Mnemoteknikoaren atzizkiak
  - ÿ b ÿ 1byte ÿ Adib.: movb  
ÿ w ÿ hitza ÿ 2 byte ÿ Adib.: movw . Testuinguru honetan hitza 2 byte da arrazoi historikoengatik.  
ÿ l ÿ luzea ÿ 4bytes ÿ Adib.: movl . Balio lehenetsia.  
ÿ q ÿ quad ÿ 8bytes ÿ Adib.: movq
- Eragigaien helbidea:

ÿ Instrukzio berean iturburu eta helmuga eragigaien ezin dute biek erreferentzia egin memoria nagusia.  
 ÿ berehalakoa: \$ operandoaren aurritzka  
 ÿ erregistroa: % aurritzka erregistratu  
 ÿ zuzena: eragigaien memoria nagusira seinalatzen duen etiketa da  
 ÿ zeharkakoa: eragigaien etiketa edo erregistro bat da: parentesiak erabiltzen ditu. (etiketa) edo (%record). Ikusi indexatua.  
 ÿ Etiketak memoriaaren kokapen bati erreferentzia egiten dio, eta horrek, aldi berean, helbide bat dauka eragiketa seinalatzen duen memoria nagusia.  
 ÿ Erregistroak eragigaien memoria-helbide nagusia dauka.  
 ÿ indexatua  
 ÿ helbide eraginkorra: base + index\*scale + disp ÿ sintaxia hau da: disp(base,index,scale)  
 ÿ foo(%ebp,%esi,4) ÿ helbide eraginkorra= EBP + 4\*ESI + foo

ÿ (%edi) ÿ helbide eraginkorra= EDI ÿ zeharkako helbideratza

ÿ -4(%ebp) ÿ norabide eraginkorra= EBP - 4

ÿ foo(%eax,4) ÿ helbide eraginkorra= 4\*EAX + foo

ÿ foo(,1) ÿ helbide eraginkorra= foo

ÿ Memoria nagusian eragigai baten erreferentzia duen eta arik ez duen edozein instrukzio erregistroko erreferentzia, eragigaiaren tamaina (byte, hitza, luzea edo laukoitza) zehaztu behar duzu atzizkia hartzan duen instrukzioa ('b', 'w', 'l' edo 'q', hurrenez hurren).

## 19.2. Muntatzaileen AS Zuzentaraauak

- Eskuliburua

ÿ <https://sourceware.org/binutils/docs/as/>

17. taula. Oinarrizko zuzentaraauak

.global edo .globl	aldagai globalak
.atala .datuak	tokiko aldagai estatikoen atala hasieratu
.atala .testua	argibideak atalean
.atala .bss	hasiera gabeko aldagaien atala
.atala .rodata	irakurtzeko soilik aldagaien atala
deskribapena .ridatzena	aldagai mota, adibidez, @funtzioa
.ohikoa 100	hasierarik gabeko 100 byte gordetzen ditu eta izan daiteke globalki erreferentziatuta
.lcomm begizta, 100	tokiko ikurrarekin erreferentziatutako 100 byte erreserbatu begizta. Hasierarik gabekoa.
.espazioa 100	100 byte erreserba zerora hasieratuta
.espazioa 100, 3	100 byte erreserbatu hasierako 3
.string "Kaixo"	gehitu 0 bytea katearen amaieran
.asciz "Kaixo"	gehitu 0 bytea katearen amaieran
.ascii "Kaixo"	ez du NULL karakterea gehitzen katearen amaieran
.byte 3,7,-10,0b1010,0xFF,0777	1 byteko tamaina eta formatuak hamartar, hamartar, hamartar, bitarra, hamaseitarra, zortzitarra
.2byte 3,7,-10,0b1010,0xFF,0777	tamaina 2 byte
.3,7,-10,0b1010,0xFF,0777 hitza	tamaina 2 byte
.laburra 3,7,-10,0b1010,0xFF,0777	2B tamaina
.4byte 3,7,-10,0b1010,0xFF,0777	4B tamaina
.luzea 3,7,-10,0b1010,0xFF,0777	4B tamaina
.int 3,7,-10,0b1010,0xFF,0777	4B tamaina
.8byte 3,7,-10,0b1010,0xFF,0777	8B tamaina
.quad 3,7,-10,0b1010,0xFF,0777	8B tamaina
.octa 3,7,-10,0b1010,0xFF,0777	zortzikoa formatua

.global edo .globl	aldagai globalak
.bikoitza 3.14159, 2 E-6	zehaztasun bikoitza
.karroza 2E-6, 3.14159	zehaztasun bakarra
.bakarra 2E-6	zehaztasun bakarra
.sartu "fixategia"	fixategia barne hartzen du. Komatxoak derrigorrezkoak dira.
.equ ARRAKASTA, 0	ARRAKASTA ikurra 0 zenbakiarekin lotzen duen makroa
.macro macname macargs	izeneko makro baten hasiera definitzen du macname eta macargs argumentuak
.amaiera makroa	makro baten amaiera definitzen du
.lerrokatu n	ondorengo argibideak edo datuak orduetan hasiko dira n byteko helbide anitz bat.
.amaiera	muntaketaren amaiera

- Little Endian Alignment: byte arinena, LSB, memoria-kokapen handienean gordetzen da. baxua.

ÿ .int OxAABBCCDD ÿ 0xDD helbide baxuenean gordetzen da lehenik, gainerako byteak. goranzko ordenan gordetzen da 0xCC, 0xBB, 0xAA ordenan

### 19.3. Muntatzeko jarraibideen erreperitorioa

- Instrukzio Erreperiorioko Eskuliburuak
- AT&T batzar hizkuntza

#### 19.3.1. TRANSFERENTZIA

Izena eta	Iruzkina	Kodea	Operazioa	ODITSZAPC
MOV Mugitu (kopiatu) MOV Source,Dest Dest:=Iturria				
XCHG Trukea	XCHG Op1,Op2 Op1:=Op2, Op2:=Op1			
STC Ezarri eramatea (Eraman = 1)	STC	CF:=1		1
CLC Garbitu Eramatea = 0)	CLC	CF:=0		0
CMC Osagarria Eraman	CMC	CF:=Ø		±
STD Ezarri STD helbidea		DF:=1(goitik kateak interpretatzen ditu behera)		1
CLD Garbitu CLD helbidea		DF:=0 (beheko kateak interpretatzen ditu gora)		0
STI	Bandera 1ean atsedenaldia	STI	BADA:=1	1
CLI	Bandera Etenaldia 0-n	CLI	BADA:=0	0
PUSH Pila pila PUSH Iturria		DEC SP, [SP]:=Iturria		

Izena eta	Iruzkina	Kodea	Operazioa	ODITSZAPC
PUSHF	Pilatu banderak	PUSHF	O, D, I, T, S, Z, A, P, C 286+: Baita NT, IOPL	
PUSHA	Pilatu erregistroak orokorra	PUSHA	AX, CX, DX, BX, SP, BP, SI, DI	
POP	Despilatu pilatik POP Dest		Helmuga:=[SP], INC SP	
POPF-k	despilatzen du banderak	POPF	O,D,I,T,S,Z,A,P,C 286+: Baita NT, IOPL	± ± ± ± ± ± ± ±
POPA	Erregistroak despilatzen ditu. orokorra.	STERN	DI, SI, BP, SP, BX, DX, CX, AX	
CBW	Bihurtu byte-ra hitzak	C.B.W.	AX:=AL (sinatua)	
CWD	Bihurtu Word-era Bikoitza	CWD	DX:AX:=AX (sinatua)	
CWDE	konb. hitza a Luzapen bikoitza	CWDE 386	EAX:=AX (ikurra duena)	
IN	Sarrera	Dest, Portuan	AL/AX/EAX := byte/hitza/bikoitza portu berezia	
OUT	Irteera		OUT ataka, iturburua byte/hitza/portuaren bikoitza := AL/AX/EAX	

- Banderak: ± =Instrukzio honek eraginda, ? =Instrukzio honen ondoren definitu gabe

### 19.3.2. ARITMETIKA

Izena eta	Iruzkina	Kodea	Operazioa	ODITSZAPC
GEHITU	Batura	GEHITU Iturburua,Dest Dest:=Dest+ Ituria		± ± ± ± ± ±
ADC	Gehitu carry ADC Ituria, Dest Dest:=Dest+ Ituria +CF			± ± ± ± ± ±
SUB	Kenketa	SUB Source,Dest Dest:=Dest- Ituria		± ± ± ± ± ±
SBB	kenketa eramangarriarekin	SBB Ituria,Dest Dest:=Dest-(Ituria +CF)		± ± ± ± ± ±
DIV	Zatiketa (gabe seinalea)	DIV Op	Op=byte: AL:=AX / Op AH:=Atseden	???????
DIV	Zatiketa (gabe seinalea)	DIV Op	Op=hitza: AX:=DX:AX / Op DX:=Atseden ??????	
DIV	386 Dibisioa (gabe seinalea)	DIV Op	Op=doblew.: EAX:=EDX:EAX / Op EDX:=Atseden	???????
IDIV	Zenbaki osoen zatiketa zeinuarekin	IDIV Op	Op=byte: AL:=AX / Op AH:=Atseden	???????
IDIV	Zenbaki osoen zatiketa zeinuarekin	IDIV Op	Op=hitza: AX:=DX:AX / Op DX:=Atseden ??????	
IDIV	386 Dibisioa zeinudun osokoa	IDIV Op	Op=doblew.: EAX:=EDX:EAX / Op EDX:=Atseden	???????

Izena eta	Iruzkina	Kodea	Operazioa	ODITSZAPC
MUL biderketa (gabe seinaldea)	MUL Op		Op=byte: AX:=AL*Op AH=0 bada #	± ? ? ? ±
MUL biderketa (gabe seinaldea)	MUL Op		Op=hitza: DX:AX:=AX*Op baldin DX=0 # ± ? ? ? ±	
MUL 386 Biderketa (seinalerik ez)	MUL Op		Op=bikoitza: EDX:EAX:=EAX*Op bai EDX=0 #	± ? ? ? ? ±
IMUL i Biderketa osorik zeinuarekin	IMUL Op		Op=byte: AX:=AL*Op AL bada nahikoa #	± ? ? ? ? ±
IMUL Multip. osorik zeinuarekin	IMUL Op		Op=hitza: DX:AX:=AX*Op AX bada nahikoa #	± ? ? ? ? ±
IMUL 386 Biderketa zeinudun osokoa	IMUL Op		Op=bikoitza: EDX:EAX:=EAX*Op bai EAX sufia da. #	± ? ? ? ? ±
INC	Areagotu	INC Op	Op:=Op+1 (The Carry ez da kaltetua!)	± ± ± ± ±
DEK Dekretu		ABENDUA Op	Op:=Op-1 (Eramatea ez da emaitzarik ematen kaltetua!)	± ± ± ± ±
CMP Konparatu		CMP	Helmuga-Iturria	± ± ± ± ± ±
GATZA	Desplazamendua aritmetika. ezkerretara	GATZA	Op, Kantitatea	i ± ± ? ± ±
SAR Desplazamendua aritmetika. eskuinera		BERAK	Op, Kantitatea	i ± ± ? ± ±
RCL	Biratu ezkerrera eraman/eraman	RCL Op,Kantitatea		i ±
RCR Biratu eskuinera eraman/eraman		RCR Op, Kantitatea		i ±
ROLA Biratu hona ezker		ROLA Op,Kantitatea		i ±

- i: informazio gehiago lortzeko, ikusi argibideen zehaztapenak,

- #: gero CF:=0, OF:=0 bestela CF:=1, OF:=1

### 19.3.3. LOGIKOA

Izena eta	Iruzkina	Kodea	Operazioa	ODITSZAPC
NEG Ezeztapena (2 osagarria)	NEG Op		Op:=0-Op Op=0 bada CF:=0 bestela CF:=1	± ± ± ± ±
EZ alderantzikatu bit bakoitza EZ Op			Op:=Ø~Op (bit bakoitza alderantzikatzen du)	
AND AND (And) logikoa AND Source,Dest Dest:=Dest ^ Source				0 ± ± ? ± 0
OR OR (Edo) logikoa	OR Source,Dest Dest:=Dest v Iturria			0 ± ± ? ± 0
XOR O (Or) XOR iturri esklusiboa,Dest Dest:=Dest (xor) Iturria				0 ± ± ? ± 0

Izena eta	Iruzkina	Kodea	Operazioa	ODITSZAPC
SHL Desplazamendua logikoa geratzen da	SHL Op, Kantitatea			i ± ± ? ± ±
SHR Desplazamendua logikoa kantitatea.	SHR Op, Eskuineko			i ± ± ? ± ±

## 19.3.4. DENBERAK

Izena eta	Iruzkina	Kodea	Operazioa	ODITSZAPC
EZ Ez egin ezer	EZ		Ez du inolako eragiketarik egiten	
IRAKURRI	Kargatu helbidea Eraginkorra	IRAKURRI Iturburua, Dest Dest := iturburu helbidea		
INT	Etenaldia	INT Zenb	Uneko prozesua eten eta bertara jauzi egiten du bektorea Zenb	0 0

## 19.3.5. JAUZIAK (orokorra)

- wiki x86 muntaia

Izena Iruzkina	Kodea	Operazioa
DEITU Azpirrutina deia	DEIALDIA Proc	
JMP Hop	JMP Dest	
J.E. Saltatu berdina bada	JE Dest	(=JZ)
J.Z. Saltatu zero bada	JZ Dest	(=JE)
JCXZ Saltatu CX zero bada	JCXZ Dest	
J.P. Saltatu parekotasuna badago	JP Dest	(= JPE)
JPE Saltatu parekotasuna ere badago	JPE Dest	(=JP)
J.P.O. Saltatu parekotasun bakoitia badago	JPO Dest	(= JNP)
JNE Saltatu berdina ez bada	JNE Dest	(=JNZ)
JNZ Saltatu zero ez bada	JNZ Dest	(= JNE)
JECXZ Jauzi ECX zero bada	JECXZ Dest 386	
JNP Saltatu parekotasunik ez bada	JNP Dest	(=JPO)
RET Azpirrutinaren itzulera	RET	

## 19.3.6. JUSTOAK Zeinurik gabe (kardinala) JUSTOAK zeinuarekin (Osokoa)

Izena Iruzkina JA Saltatu	Kodea	Operazioa
altuagoa bada Saltatu handiagoa edo	JA Dest	(=JNBE)
JAE berdina bada JAE Dest Saltatu baxuagoa bada Jauzi		(= JNB = JNC)
J.B. txikiagoa bada edo berdina	JB Dest	(= JNAE = JC)
JBE JBE Dest Saltatu handiagoa ez bada JNA Dest Saltatu		(= JNA)
JNA super ez bada. edo berdina JNAE Dest Saltatu txikiagoa		(= JBE)
JNAE ez bada Saltatu txikiagoa ez bada. edo berdin JNBE Dest		(= JB = JC)
J.N.B. Skip JC Dest eramanez gero	JNB Dest	(= JAE = JNC)
JNBE		(=HA)
J.C.	JO Dest	Salto gainezka badago

JNC	Jauzi egin garaiatzea ez badago	JNC Dest	
JNO	Saltatu gainezkarik ez badago	JNO Dest	
J.S.	Saltatu zeinu bat badago (=negatiboa) JS Dest		
J.G.	Saltatu zaharragoa bada	JG Dest	(=JNLE)
JGE	Saltatu handiagoa edo berdina bada	JGE Dest	(=JNL)
J.L.	Saltatu gutxiago bada	JL Dest	(=JNGE)
JLE	Saltatu baino txikiagoa edo berdina bada	JLE Dest	(=JNG)
JNG	Saltatu zaharragoa ez bada	JNG Dest	(=JLE)
JNGE	Saltatu JNGE Dest baino handiagoa edo berdina ez bada		(=JL)
J.N.L.	Saltatu beherago ez bada	JNL Dest	(= JGE)
JNLE	Saltatu JNLE Dest baino txikiagoa edo berdina ez bada		(=JG)

### 19.3.7. BANDERAK (ODITSZAPC)

O: gainezka funtzionamenduaren emaitza. seinalerik gabe handiegia edo txikiagia da.

D: Helbidea

I: Etenaldia Etenaldia gerta daitezkeen edo ez adierazten du.

T: Trap Urratsez urrats arazketarako

S: Zeinua emaitzaren seinale. Zenbaki osoentzat bakarrik arrazoizkoa. 1=ez. 0=pos.

Z: Zero Eragiketaren emaitza zero da. 1=Zero

A: Carru Aux. Carry-ren antzekoa, baina nibble baxurako soilik mugatuta

P: Parekidetasuna 1 = emaitzak batean bit bikoitiak ditu

C: Eragiketaren emaitza eraman. sinatu gabekoa oso handia da edo zero baino txikiagoa da

- Opcode atzizki mnemoteknikoak:

ÿ q : laukoa: 8 byteko eragigaia: lauko hitza

ÿ l : luzea: 4 byteko eragigaia: hitz bikoitza

ÿ w : hitza: 2 byteko eragigaia: hitza

ÿ b : byte: 1 byteko eragigaia

- Eragiketa mnemoteknikoak atzizkirkirik ez badu, eragigaiaren tamaina lehenetsia luzea da

### 19.3.8. Mnemoteknikoaren zerrenda

Quick x86 muntaketa mnemoteknikoa NUP-UPNA

A.D.D.		NEG ;Ezezkoa (2 bete)
ADC	;Gehiketa ;Gehiketa	EZ; Inbertitu bit bakoitza
AZPI		ETA ;'Y' (Eta) logikoa
SBB		EDO ;'O' (Edo) logikoa
DIV	eramatearekin ;Kenketa ;Kenketarekin eramanean	XCHG ;Desplazamendu logikoa ezkerrera
DIV	;Zatiketa (sinatu gabe)	SHR; Desplazamendu logikoa eskuinera
DIV	;386 Dibisioa (sinatu gabe)	MOV ;Mugitu (kopiatu)
IDIV	;Sinatutako Zenbakien	XCHG ;Trukea
IDIV	Zatiketa ;Sinatutako Zenbakien	STC ;Ezarri eramatea (Eraman=1)
IDIV	Zatiketa ;386 Sinatuko Zenbakien Zatiketa CLC ;Garbitu Eramatea (Eramatea=0)	
MUL	; Biderketa (sinatu gabe)	CMC ;Complement Carry
MUL	; Biderketa (sinatu gabe)	STD ;Ezarri helbidea
MUL	;386 Biderketa (sinatu gabe) CLD ;Garbitu helbidea	
IMUL	;i Biderkatu. zenbaki osoa STI ikurra duena; Eten bandera 1ean	
IMUL	; Biderkatu sinatutako osoko CLI ;Etenen bandera 0an ezarri da	
IMUL	;386 Biderkatu integer signPUSH ;Pila pila batean	
INC		PUSHF; Pilotu banderak
AB		PUSHA; Pilatu arau orokorrak
CMP		POP ;Despilatu pilatik
GATZA	;Increment ;Decrement ;Konparatu ;Shift aritm. EZA ;Despilatu	INCA ;Despilatu banderak
BERAK	;Shift aritmética. eskuinera	AFT ;Despilatu erregistroak. orokorra.
RCL	;Biratu ezkerrera w/carry ;Biratu	CBW ;Bihurketa Byte hitzetik
RCR	eskuinera w/carry CWD ;Conv. Hitza Bikoitzu	
ERROLA	;Biratu ezkerrera ;Biratu	CWDE ;konb. Hitz bikoitzeko luzapena
R.O.R.	eskuinera ;Ezer ez	IN ;Sarrera
EZ	egin ;Kargatu	OUT ;Irteera
IRAKURRI	helbidea Eraginkorra ;Eten Azpirrutina	Zuzentaraauak
INT	deia Saltatu	.global edo .globl etiketa
DEITU		.atala .datuak
JMP		.atala .testua
J.E.	Jauzi berdina bada Salto	.atala .bss
J.Z.	zero bada Jauzi CX zero	.atala .rodata
JCXZ	bada Jauzi parekidetasuna	.mota izena, deskribapena
J.P.	bada Salto parekidetasun	.tamaina arrunta
JPE	bikoitia bada .lcomm Jauzi parekidetasun bakoitia	etiketa, tamaina
J.P.O.	bada .space Jauzi berdina ez bada .end	tamaina
JNE		
JNZ	Saltatu zero ez bada	.katea
JECXZ	Jauzi ECX zero bada	.asviz
JNP	Saltatu parekotasunik ez bada	.ascii
RET	Azpirrutinaren itzulera	.byte
JA	Saltatu .2 byte baino handiagoa bada	
JAE	Saltatu .word baino handiagoa edo berdina bada	
J.B.	Saltatu txikiagoa bada .labur	
JBE	Saltatu .4byte baino txikiagoa edo berdina bada	
JNA	Saltatu gorago ez bada .luze	

JNAE	Saltatu super ez bada. edo berdin .int	
J.N.B.	Saltatu .8byte baino gutxiago ez bada	
JNBE	Saltatu ondorioztatu ez bada. edo berdin .quad	
J.C.	Saltatu carry JC Dest .octa badago	
JNC	Saltatu ez bada eraman .bikoitza	
JNO	Saltatu gainezkarik ez badago Saltatu	.flotatu
J.S.	zeinua badago (=negatiboa).bakarra	
J.G.	Saltatu handiagoa bada .include "fitxategia"	
JGE	Saltatu NAME, balioa baino handiagoa edo berdinazada	bada
J.L.	Jauzi txikiagoa bada Salto	.macro macname m
JLE	baino txikiagoa edo berdina bada Salto	.amaiera makroa
JNG	handiagoa ez bada Salto	.zenbakia lerrokatu
JNGE	handiagoa edo berdina ez bada	
J.N.L.	Saltatu beherago ez bada	
JNLE	Saltatu baino txikiagoa edo berdina ez bada	

|63-0| 31-0 | 15-0| 15-8| 7-0

|rax | eax | aizkora | oi | du

|rbx | ebx | bx | bh | bl

|rcx | ecx | cx | ch | cl

|rdx | edx | dx | dh | dl

|rsi | esi | bai || sil

|rdi | edi | esan || dil

|rbp | ebp | bp || bpl

|rsp | esp | sp || spl

|r8 | r8d | r8w || r8b

|r9 | r9d | r9w || r9b

|r10 | r10d | r10w| r10b

|r11 | r11d | r11w| r11b

|r12 | r12d | r12w| r12b

|r13 | r13d | r13w| r13b

|r14 | r14d | r14w| r14b

|r15 | r15d | r15w| r15b

## 19.4. Erregistroak

### 19.4.1. Ikuspegি osoa

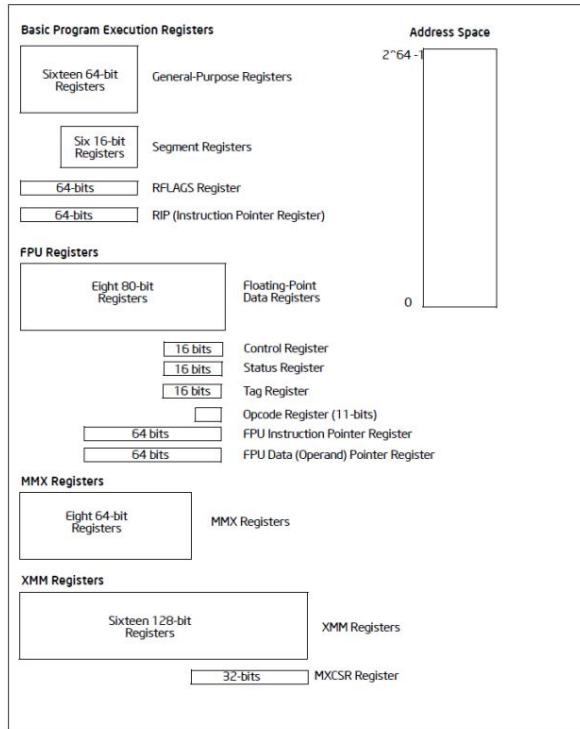


Figure 3-2. 64-Bit Mode Execution Environment

- 32 biteko RPG helburu orokorreko erregistroak hauetakoak dira:

ü '%eax' (metagailua), '%ebx', '%ecx', '%edx', '%edi', '%esi', '%ebp' (marko erakuslea) eta '%esp' (pila-erakuslea).

register encoding	not modified for 8-bit operands				low 8-bit	16-bit	32-bit	64-bit
	not modified for 16-bit operands		zero-extended for 32-bit operands					
0			AH*	AL		AX	EAX	RAX
3			BH*	BL		BX	EBX	RBX
1			CH*	CL		CX	ECX	RCX
2			DH*	DL		DX	EDX	RDX
6				SIL**		SI	ESI	RSI
7				DIL**		DI	EDI	RDI
5				BPL**		BP	EBP	RBP
4				SPL**		SP	ESP	RSP
8				R8B		R8W	R8D	R8
9				R9B		R9W	R9D	R9
10				R10B		R10W	R10D	R10
11				R11B		R11W	R11D	R11
12				R12B		R12W	R12D	R12
13				R13B		R13W	R13D	R13
14				R14B		R14W	R14D	R14
15				R15B		R15W	R15D	R15

**Figure 2-3.** General Registers in 64-Bit Mode

#### **19.4.2. Programatzailak ikusgai dauden erregistroak**

63-0	31-0	15-0	15-8	7-0
rax	eax	aizkora	oi	du
rbx	ebx	bx	bh	bl
rcx	ecx	cx	ch	cl
rdx	edx	dx	dh	dl
rsi	Hori da	Bai		sil
rdi	zumurbiloa	erman		dil

RFLAGS

513-309.eps

RIP

- \* Not addressable when a REX prefix is used.

**\*\* Only addressable when a REX prefix is used.**

rbp	ebp	bp		bpl
rsp	esp	sp		spl
r8	r8d	r8w		r8b
r9	r9d	r9w		r9b
r10	r10d	r10w		r10b
r11	r11d	r11w		r11b
r12	r12d	r12w		r12b
r13	r13d	r13w		r13b
r14	r14d	r14w		r14b
r15	r15d	r15w		r15b

#### 19.4.3. Bateragarritasuna 32-64

- 64 biteko arkitekturako erregistroen izendapenean, R ordeztu E-rekin eta izendapena lortuko dugu.  
32 biteko arkitekturarena.

64 bit	32 bit
R.I.P.	EIP
RAX	EAX
RFLAG	EFLAG
.....	.....

- Salbuespenak daude

#### 19.4.4. Kontrol-bandera-erregistroa

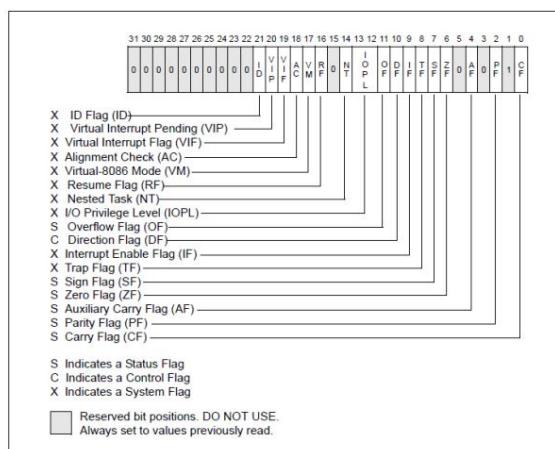


Figure 3-8. EFLAGS Register

- EGOERA erregistroa: Instrukzio baten exekuzioak hori bandera izeneko bit batzuk aktibatzen ditu egindako eragiketaren ondorioak adierazi. Adibidea: gainezkatze bandera: eragiketa dela adierazten du egindako aritmetikak aipatutako eragiketaren emaitza gainezka egin du.
- wikipedia
- OSZAPC banderei bakarrik begiratzen diegu.

18. taula. RFLAG Erregistroa

Bandera	Bit	Yam
C.F.	0	Eraman bandera
PF	2	Parekidetasun bandera
AF	4	bandera egokitu
ZF	6	zero bandera
S.F.	7	Sinatu bandera
OF	8 tamaina	Gainetik bandera

- Eraman bandera:

ÿ aktibatuta dago zenbaki-eragiketa aritmetiko batean liburu-liburuak ALUren hitzaren zabalera gainditzen badu sinatu gabeko edo sinatu gabeko zenbaki osoak

- Gainerako bandera:

ÿ aktibatuta dago, MSB pisu handiena duen bit-a kontuan hartuta (tamainatik kanpo egon arren) errore bat adierazten badu. eragiketa aritmetikoan zeinudun zenbaki osoekin.

- Parekidetasunaren bandera:

ÿ azken eragiketaren emaitzaren LSB bytearen bit kopurua bikoitia den ala ez adierazten du.

- Sinatu bandera:

ÿ aktibatzen da azken eragiketaren emaitza negatiboa izan bada.

- Doitu bandera:

ÿ aktibatu egiten da LSB-n azken eragiketaren emaitza moztuta eramatzen bada

**GDB QUICK REFERENCE** GDB Version 5**Essential Commands**

gdb program [core] debug program [using coredump core]  
 b [file:]line set breakpoint at function [in file]  
 run [arglist] start your program [with arglist]  
 bt backtrace: display program stack  
 p expr display the value of an expression  
 c continue running your program  
 n next line, stepping over function calls  
 s next line, stepping into function calls

**Starting GDB**

gdb start GDB, with no debugging files  
 gdb program begin debugging program  
 gdb program core debug core dump core produced by  
 program  
 gdb --help describe command line options

**Stopping GDB**

quit exit GDB; also q or EOF (eg C-d)  
 INTERRUPT (eg C-c) terminate current command, or  
 send to running process

**Getting Help**

help list classes of commands  
 help class one-line descriptions for commands in  
 class  
 help command describe command

**Executing your Program**

run arglist start your program with arglist  
 run start your program with current arguments  
 run ... <inf>>outf start your program with input, output  
 redirected  
 kill kill running program  
 tty dev use dev as stdin and stdout for next run  
 set args arglist specify arglist for next run  
 set args specify empty argument list  
 show args display arguments list  
 show env show all environment variables  
 show env var show value of environment variable var  
 set env var string set environment variable var  
 unset env var remove var from environment

**Shell Commands**

cd dir change working directory to dir  
 pwd Print working directory  
 make ... call "make"  
 shell cmd execute arbitrary shell command string

[ ] surround optional arguments ... show one or more arguments

©1998, 2000, 2010 Free Software Foundation, Inc. Permissions on back

**Breakpoints and Watchpoints**

break [file:]line set breakpoints at line number [in file]  
 b [file:]line eg: break main.c:37  
 break [file:]func set breakpoints at func [in file]  
 break \*offset set break at offset lines from current stop  
 break \*addr set breakpoints as address addr  
 break set breakpoints as next instruction  
 break ... if expr break conditionally on nonzero expr  
 cond n [expr] now conditional expression on breakpoint  
 n; make unconditional if no expr  
 tbreak ... temporary break; disable when reached  
 rbreak [file:]reger break on all functions matching reger [in  
 file]  
 watch expr set a watchpoints for expression expr  
 catch event break at event, which may be catch,  
 throw, exec, fork, vfork, load, or  
 unload.  
 info break show defined breakpoints  
 info watch show defined watchpoints  
 clear delete breakpoints at next instruction  
 clear [file:]fun delete breakpoints at entry to fun()  
 clear [file:]line delete breakpoints on source line  
 delete [n] delete breakpoints [or breakpoint n]  
 disable [n] disable breakpoints [or breakpoints n]  
 enable [n] enable breakpoints [or breakpoints n]  
 enable once [n] enable breakpoints [or breakpoints n];  
 disable again when reached  
 enable del [n] enable breakpoints [or breakpoints n];  
 delete when reached  
 ignore n count ignore breakpoint n, count times  
 commands n execute CDB command-list every time  
 [silent] breakpoint n is reached. [silent  
 command-list suppresses default display]  
 end end of command-list

**Program Stack**

backtrace [n] print trace of all frames in stack; or of n  
 frames—innermost if n>0, outermost if  
 n<0  
 bt [n] select frame number n or frame at address  
 frame [n] n; if no n, display current frame  
 up n select frame n frames up  
 down n select frame n frames down  
 info frame [addr] describe selected frame, or frame at addr  
 info args arguments of selected frame  
 info locals local variables of selected frame  
 info reg [rn]... register values [for reg: rn] in selected  
 frame; all-reg includes floating point  
 info all-reg [rn]

**Execution Control**

continue [count] continue running; if count specified, ignore  
 this breakpoint next count times  
 step [count] execute until another line reached; repeat  
 count times if specified  
 s [count] step by machine instructions rather than  
 source lines  
 next [count] execute next line, including any function  
 calls  
 n [count] next machine instruction rather than  
 source line  
 until [location] run until next instruction (or location)  
 finish run until selected stack frame returns  
 return [expr] pop selected stack frame without  
 executing [setting return value]  
 signal num resume execution with signal s (none if 0)  
 jump line resume execution at specified line number  
 or address  
 set var=expr evaluate expr without displaying it; use  
 for altering program variables

**Display**

print [/f] [expr]	show value of expr [or last value \$] according to formats f:
p [/f] [expr]	x hexadecim d signed decimal u unsigned decimal o octal t binary a address, absolute and relative c character f floating point
call [/f] expr	like print but does not display void
x [/Nuf] expr	examine memory at address expr; optional format spec follows slash count of how many units to display
N	unit size: one of
u	b individual bytes h halfwords (two bytes) w words (four bytes) g giant words (eight bytes)
f	printing formats. Any print format, or a null-terminated string i machine instructions
disassm [addr]	display memory as machine instructions

**Automatic Display**

display [/f] expr show value of expr each time program  
 stops [according to formats f]  
 display display all enabled expressions on its  
 undisplay n remove number(s) n from list of  
 automatically displayed expressions  
 disable disp n disable display for expression(n)  
 enable disp n enable display for expression(s) number n  
 info display numbered list of display expressions

Expressions	Controlling GDB	Source Files
<code>expr</code>	an expression in C, C++, or Modula-2 (including function calls), or:	<code>dir names</code> add directory names to front of source path
<code>addr@len</code>	an array of len elements beginning at addr	<code>dir</code> clear source path
<code>file::nm</code>	a variable or function nm defined in file	<code>show dir</code> show current source path
<code>{type} addr</code>	read memory at addr as specified type	<code>list</code> show next ten lines of source
<code>\$</code>	most recent displayed value	<code>list -</code> show previous ten lines
<code>\$n</code>	nth displayed value	<code>list lines</code> display source surrounding lines, specified as:
<code>\$\$</code>	displayed value previous to \$	<code>[file:]num</code> line number [in named file]
<code>\$\$n</code>	nth displayed value back from \$	<code>[file:]function</code> beginning of function [in named file]
<code>\$_</code>	last address examined with x	<code>*off</code> off lines after last printed
<code>\$var</code>	value at address \$_	<code>-off</code> off lines previous to last printed
<code>show values [n]</code>	convenience variable; assign any value	<code>*address</code> line containing address
<code>show conv</code>	show last 10 values [or surrounding \$n]	<code>list f,l</code> from line f to line l
	display all convenience variables	<code>info line num</code> show starting, ending addresses of compiled code for source line num
Symbol Table		<code>info source</code> show name of current source file
<code>info address s</code>	show where symbol s is stored	<code>info sources</code> list all source files in use
<code>info func [regez]</code>	show names, types of defined functions (all, or matching regez)	<code>forw regex</code> search following source lines for regex
<code>info var [regez]</code>	show names, types of global variables (all, or matching regez)	<code>rev regex</code> search preceding source lines for regex
<code>whatis [expr]</code>	show data type of expr [or \$] without evaluating; ptype gives more detail	
<code>ptype [expr]</code>	describe type, struct, union, or enum	
<code>ptype type</code>		
GDB Scripts		
<code>source script</code>	read, execute GDB commands from file script	
<code>define cmd</code>	create new GDB command cmd; execute	
<code>command-list</code>	scripts defined by command-list	
<code>end</code>	end of command-list	
<code>document cmd</code>	create online documentation for new GDB command cmd	
<code>help-text</code>	help-text	
<code>and</code>	end of help-text	
Signals		
<code>handle signal act</code>	specify GDB actions for signal:	
<code>print</code>	announce signal	
<code>noprint</code>	be silent for signal	
<code>stop</code>	halt execution on signal	
<code>nostop</code>	do not halt execution	
<code>pass</code>	allow your program to handle signal	
<code>nopass</code>	do not allow your program to see signal	
<code>info signals</code>	show table of signals, GDB action for each	
Debugging Targets		
<code>target type param</code>	connect to target machine, process, or file	
<code>help target</code>	display available targets	
<code>attach param</code>	connect to another process	
<code>detach</code>	release target from GDB control	
Controlling GDB		
<code>set param value</code>	set one of GDB's internal parameters	
<code>show param</code>	display current setting of parameter	
<code>Parameters understood by set and show:</code>		
<code>complaint limit</code>	number of messages on unusual symbols	
<code>confirm on/off</code>	enable or disable cautionary queries	
<code>editing on/off</code>	control readline command-line editing	
<code>height lpp</code>	number of lines before pause in display	
<code>language lang</code>	Language for GDB expressions (auto, c or modula-2)	
<code>listsize n</code>	number of lines shown by list	
<code>prompt str</code>	use str as GDB prompt	
<code>radix base</code>	octal, decimal, or hex number representation	
<code>verbose on/off</code>	control messages when loading symbols	
<code>width cpl</code>	number of characters before line folded	
<code>write on/off</code>	Allow or forbid patching binary, core files (when reopened with exec or core)	
<code>history ...</code>	groups with the following options:	
<code>b ...</code>		
<code>b exp off/on</code>	disable/enable reading history expansion	
<code>b file filename</code>	file for recording GDB command history	
<code>b size size</code>	number of commands kept in history list	
<code>b save off/on</code>	control use of external file for command history	
<code>print ...</code>	groups with the following options:	
<code>p ...</code>		
<code>p address on/off</code>	print memory addresses in stabs, values	
<code>p array off/on</code>	compact or attractive format for arrays	
<code>p demangle on/off</code>	sources (demangled) or internal form for C++ symbols	
<code>p asm-dec on/off</code>	demangle C++ symbols in machine-instruction output	
<code>p elements limit</code>	number of array elements to display	
<code>p object on/off</code>	prints C++ derived types for objects	
<code>p pretty off/on</code>	struct display: compact or indented	
<code>p union on/off</code>	display of union members	
<code>p vtbl off/on</code>	display of C++ virtual function tables	
<code>show commands</code>	show last 10 commands	
<code>show commands n</code>	show 10 commands around number n	
<code>show commands +</code>	show next 10 commands	
Working Files		
<code>file [file]</code>	use file for both symbols and executable; with no arg, discard both	
<code>core [file]</code>	read file as coredump; or discard	
<code>exec [file]</code>	use file as executable only; or discard	
<code>symbol [file]</code>	use symbol table from file; or discard	
<code>load file</code>	dynamically link file and add its symbols	
<code>add-sym file addr</code>	read additional symbols from file, dynamically loaded at addr	
<code>info files</code>	display working files and targets in use	
<code>path dirs</code>	add dirs to front of path searched for executable and symbol files	
<code>show path</code>	display executable and symbol file path	
<code>info share</code>	list names of shared libraries currently loaded	
Source Files		
<code>dir names</code>	add directory names to front of source path	
<code>dir</code>	clear source path	
<code>show dir</code>	show current source path	
<code>list</code>	show next ten lines of source	
<code>list -</code>	show previous ten lines	
<code>list lines</code>	display source surrounding lines, specified as:	
<code>[file:]num</code>	line number [in named file]	
<code>[file:]function</code>	beginning of function [in named file]	
<code>*off</code>	off lines after last printed	
<code>-off</code>	off lines previous to last printed	
<code>*address</code>	line containing address	
<code>list f,l</code>	from line f to line l	
<code>info line num</code>	show starting, ending addresses of compiled code for source line num	
<code>info source</code>	show name of current source file	
<code>info sources</code>	list all source files in use	
<code>forw regex</code>	search following source lines for regex	
<code>rev regex</code>	search preceding source lines for regex	
GDB under GNU Emacs		
<code>M-x gdb</code>	run GDB under Emacs	
<code>C-h m</code>	describe GDB mode	
<code>M-s</code>	step one line (step)	
<code>M-n</code>	next line (next)	
<code>M-i</code>	step one instruction (stop)	
<code>C-c C-f</code>	finish current stack frame (finish)	
<code>M-c</code>	continue (cont)	
<code>M-u</code>	up arg frames (up)	
<code>M-d</code>	down arg frames (down)	
<code>C-x k</code>	copy number from point, insert as end	
<code>C-x SPC</code>	(in source file) set break at point	
GDB License		
<code>show copying</code>	Display GNU General Public License	
<code>show warranty</code>	There is NO WARRANTY for GDB. Display full no-warranty statement.	
Copyright ©1991, 1992, 1993, 1998, 2000, 2010 Free Software Foundation, Inc. Author: Roland H. Poesch		
The author assumes no responsibility for any errors on this card.		
This card may be freely distributed under the terms of the GNU General Public License.		
Please contribute to development of this card by annotating it.		
Improvements can be sent to bug-gdb@gnu.org.		
GDB itself is free software; you are welcome to distribute copies of it under the terms of the GNU General Public License. There is absolutely no warranty for GDB.		

## 19.5. GDB

- Oinarritzko komandoak: exekutatu `gdb` komandoa

```

shell data
shell pwd
shell ls
shTAB shell
Cx-tik
Cx-ra

edo historiako komandoak: nabigatu geziekin ezarri arrasto-
komandoak ezarri erregistro-
fitxategian gdb_salida.txt ezarri saioa shell-
en ls -l gdb_salida.txt
fitxategia module_bin informazio
ituriak informazio
iturria

hautsi nagusia
b _hasi
informazioa eten
puntuak info reTAB

```

Konika egin

```

hurrengoa , n , n 5
urratsa, bai
ITZULI
jarraitu, c
hasi
arte, itzulera, itzulera ... hurrengo
instrukzioa, ni, RET, RET, RET, RET, arte, RET,..begizta urratsetik irten arte ,s bai

```

```

ptype aldagaia
whatis aldagaia
inprimatu aldagaia, p aldagaia, p /t aldagaia, p /xnp &n p $rax p
$eax
p $ax p
$ah p
$rflags
p
$eflags p /t
$eflags p
$rip x helbidea
x
&aldagaia, x /
1bw &aldagaia, +x /1xw &aldagaia, x /4xw &aldagaia zain? n layout split h layout layout src
focus src :
nabigatu fokua
cmd :
nabigatu
informazioa erregistroen
informazioa reTAB disas /
r _hasi disas /m
_hasi

q

```

# VII Autoebaluazio Praktikak

## 20. kapitula. Praktikak: Galdetegia

### 20.1. Praktika 1: Mihiztadura Lengoaiaren Programaziorako Sarrera AT&T x86-32

#### 20.1.1. Gai teorikoak

1. Zein da AT&T-ren muntaia-lengoaia eta Intel-en arteko desberdintasun nagusia.
2. Zein fase ditu tresna-kateak?
3. Jardunaldietan erabiliko diren garapen-tresnak zerrendatu erabilitako bi prozesadoreak erabiliz.
4. Liburua: Programazioa oinarritik
  - a. Zer da GNU/Linux
  - b. Zer da GNU
  - c. Zer da gcc
  - d. Nukleoak kudeatzen duena
  - e. Argibideak eta datuak aldi berean sar daitezke? Zergatik?
  - f. Zein da PC erregistroaren funtzioa
  - g. Zeintzuk dira CPU-erregistro bi motak
  - h. Zer esan nahi du hitzaren tamainak?
  - Yo. Zer da erakuslea aldagai bat.
  - j. Zerrendatu lau eragiketa modu desberdinan eragiketa bat zuzentzeko.

#### 20.1.2. Gai praktikoak

1. Taula barne hartzen duen gcc frontend-a erabiliz muntaketa-iturburu-programa konpilatzeko komandoa arazketarako sinboloak
2. Lekualdagari dagoen objektu moduluaren estekatzailearen komandoa.
3. C hizkuntzan n byte bateko zeinu osoko aldagai motaren adierazpena.
4. Batuketa bat egiten duen sum1toN.s programaren mihiztadura-hizkuntzaren instrukzioa.
5. gdb debugger komandoak n aldagaiaren edukia inprimatzeko
6. Exekutatu sum1toN, sum1toN.s-etik konpilatuta, urratsez urrats GDB arazketa erabiliz behar diren komandoak:
  - to. n aldagaiaren edukia eta bere helbidea inprimatu memoria nagusian
  - b. inprimatu begizta etiketa helbidea
  - c. inprimatu ECX erregistroaren edukia begiztatik irtetean
7. Aldatu gehiketa-eragigaien tamaina 2 byte-ra
  - to. Eragigaiaren tamaina aldatu n ÿ n: .hitza 5
8. Aldatu add %edx,%ecx instrukzioa addw %dx,%ec instrukziora
9. GDBn zein komando erabili behar den programaren begiztaren iterazio guztiak exekutatzeko etengabe.
10. Iturburu-moduluen C eta ASM hizkuntza-bertsioak alderatuz, zergatik exekutatzen den GDB araztailearen hasta sententzia C iturburu-moduluaren kasuan, while sententzian zehar.

11. Iturburu-modulua editatzeak muntaketa-errore bat eragiten du, ez baitago koherentziarik n etiketak erreferentziatutako eragigaiaren tamainaren deklarazioaren eta gehitzeko instrukzioaren eragigaiaren tamainaren deklarazioaren artean.

## 20.2. 2. praktika: Datuen irudikapena

### 20.2.1. Moduluaren datuak\_tamaina.s

- Zein ordenatan goredetzen dira "kaixo" kateko karaktereak?
- Zein da o karakterearen ASCII kodea?
- Zein da "kaixo" katea goredetzen den memoria nagusia?
- Zein da zerrenda-matrizea goredetzen den memoria-helbide nagusia?
  - ÿ Zein da goranzko norabidean aurreko helbidetik hasita lehen 4 byteen edukia?

### 20.2.2. Moduluaren datuak\_atzizkiak.s

- Zein ordenatan goredetzen dira da4 datuen byteak?
- Zein da `mov da1,%ecx` exekutatzeko emaitza ?

### 20.2.3. Moduluaren helbideratze\_datuak.s

- Araztearekin, exekutatu programa urratsez urrats moduan, honako eragiketa hauetan eginez.
  - Array da2
    - ÿ Inprimatu da2 matrizearen memoria helbidea eta lehen elementuaren edukia: `x /xh &da2`
    - ÿ da2 arrayaren 2 byteko 4 elementu: `x /4xh &da2`
    - ÿ **ptype da2:** arazte-informaziorik ez: araztaileak elementuen tamainari buruzko informaziorik ez duenez, Ondorengo komandoetan esplizituki adierazi behar dira.
      - ÿ **Casting** bat egin behar da : 2 byte-ko 4 elementuko array: `p /x (labur[4])da2`
      - ÿ Egiaztatu eXaminar komandoarekin emaitza **casting** bat egiten dugun ala ez (laburra \*): `x /4xh (laburra *)&da2`
      - ÿ Datuen tamaina eta mota komandoaren argumentuak ezartzen du: `/4xh`
    - ÿ Egiaztatu little endian biltegiratze estandarra, memoria-helbide bakoitzak a identifikatuz byte bere edukiarekin.
    - ÿ Sartu da2 arrayaren -21 balioa duen elementuaren memoria helbidea:
      - ÿ `p da2[2]` -n array-elementuaren argumentua baliogabea da araztailea falta baita informazioa

- Desmuntatu
  - ÿ `disas salto1`
  - ÿ `dis /r salto1`

## 20.3. 3. praktika: Eragiketa aritmetiko-logikoak eta baldintzapeko jauzi-argibideak

### 20.3.1. modulua op\_arit\_log.s

- [Op\\_arit\\_log iturburu-modulua](#)

- Adierazi gehigarrien balioa OPE1 eta OPE2 makroei nola lotu.

- Eragileen balioa aldatu gabe:

ÿ adierazi kenketaren balioa **1. adierazpenean**  
 ÿ adierazi biderketaren balioa **2. adierazpenean**  
 ÿ adierazi zatiketaren balioa **3. adierazpenean**  
 ÿ adierazi zatiketaren balioa **4. adierazpenean**  
 ÿ adierazi **5. adierazpeneko eragiketa logikoen** balioa

### 20.3.2. Modulu jauziak.s

- Banderaren erregistroa

ÿ Editatu, konpilatu eta exekutatu hurrengo adierazpen-blokea erregistroaren edukia adierazteko  
 EAX eta CF,ZF,SF,PF,OF banderen egoera instrukzio bakoitza exekutatu ondoren:

```
mugitu $0xFFFFFFFF,%eax shr  

$1,%eax gehitu %eax,  

%eax probab $0xFF,%eax  
  

cmpi $0xFFFFFFFF,%eax
```

- Jauziak

ÿ Editatu, konpilatu eta exekutatu jarraibide-bloke hau CF,ZF,SF,PF,OF jauziaren instrukzioa exekutatu baino lehen  
 banderen egoera adierazteko eta jauzia gertatzen den edo ez adierazteko.

```
mov $0x00AA, %ax mov  

$0xFF00, %bx cmp %bx,%ax  

eta jump1 jg jump2  

jump1: mugitu $0xFF,  

%ebx jump2: mugitu  

$1,%eax int $0x80
```

## 20.4. 4. praktika: Sistema Eragilerako deiak

### 20.4.1. modulua `syscall_write_puts.c`

- "Kaixo" ongietorri-mezua pantailan inprimatzten duen C hizkuntzan garatu programa bat.  
 Erabili C liburutegi estandarreko puts(), write() eta syscall() funtziokoak.

ÿ gizona 2 jartzen du

ÿ man 2 write : funtzioaren prototipoa

WRITE(2) Linux Programatzalearen Eskuliburua

YAM

idatzi - idatzi fitxategi deskribatzaile batean

## SINOPSIS

```
#include <unistd.h>

ssize_t write(int fd, const void *buf, size_t count);
```

ŷ fd : fitxategiaren deskribatzailea: monitorea 1. zenbakia duen deskribatzailea duen fitxategi birtuala da.

ŷ void \*buf: buffer inprimatu nahi den karaktere katera seinalatzen duen erakuslea da.

ŷ count: inprimatu beharreko karaktere-katearen gehienezko tamaina

ŷ Funtzio honek zeharka deitzen dio sistema eragileari syscall() deiaren bidez.

```
/*
Programa syscall_write_puts.c Deskribapena:
Sistema eragileari deia egiten dio pantailan inprimatzeko Deia hiru modu ezberdinetan egiten du: puts,
write,
syscall.

Konpilazioa: gcc -m32 -g -o puts_gets puts_gets.c */
```

```
// Liburutegiko goiburuak #include
<stdio.h> // puts() funtzioaren prototipoa #include <unistd.h> //makroen
deklaraziona STDOUT_FILENO, STDIN_FILENO #include <syscall.h> //syscall funtzioaren deklaraziona
#include <sys/syscall.h> // __NR_write eta __NR_exit makro adierazpena
#include <stdlib.h> //exit() adierazpena
```

```
// Makroak
#define LON_BUF 5 // Katearen tamaina
```

```
void main(void) {
```

```
char buffer[LON_BUF] = "Kaixo\n";
```

```
puts("\n***** Praktika: SISTEMA DEIAK *****\n"); // libc
liburutegiko puts() funtzioa puts("\n***** write() funtzioa erabiliz ongietorri
mezua inprimatzen dut: ");
```

```
idatzi(STDOUT_FILENO, buffer, LON_BUF); // idazketa sistema-deiaren bilgarria.
```

```
// write() a barne hartzen baitu
syscall(), zeharka deitzen dio sistemari
```

```
puts("\n***** Ongietorri mezua inprimatzen dut
```

```

    syscall() sistema-deia: ");
    syscall(__NR_write, STDOUT_FILENO, buffer, LON_BUF); // zuzeneko sistema deia
    syscall funtzioa. exit(0xAA); //

    Iten sistematik 0xAA kodea bidaliz. Ez da berdina
    itzuli

}

```

#### 20.4.2. modulua syscall\_write\_puts.s

- Garatu syscall\_write\_puts.s syscall\_write\_puts.c programaren baliokidea den x86-32 mihiztadura-lengoia programa puts() eta write() funtzioei deituz. syscall() funtzioa erabili beharrean, egin sistema-deia zuzenean int 0x80 instrukzioarekin. Sistema eragilearen deiaren argumentuak erregistroen bidez pasatzen dira:
  - ÿ 1. argumentua: EAX bidez: int mota: balioa \_\_NR\_write: 4 balioa
  - ÿ 2. argumentua: EBX: type int fd: balioa STDOUT\_FILENO: balioa 1
  - ÿ 3. argumentua: ECX: idatzi const void \*buf: inprimatu beharreko katearen erakuslea, buffer
  - ÿ 4. argumentua: EDX: mota size\_t count: balioa LON\_BUF: balioa 5
  - ÿ RTL deskribapena

```

# deitu libc liburutegiko puts funtzioari. Beharrezko da libc-rekin lotzea. stack <-argument
call-ek # deia
jartzen dio
libc liburutegiko idazketa funtzioari. Beharrezko da libc-rekin lotzea. pila <- 3. argumentu pila
<- 2. argumentu pila <-
1. argumentu deia idatzi
# Deitu sistema
eragileari
idazketa eragiketa exekutatzeko EAX<-4

EBX<-1
ECX<-etiketa, inprimatu nahi den katea seinalatzen duena
EDX<-5
deitu sistema_erafilea
# Deitu sistema eragileari irteera eragiketa exekutatzeko
EAX<-1
EBX<-0
deitu sistema_erafilea

```

#### 20.5. 5. praktika: Azpirrutina baterako deiak

##### 20.5.1. modulua sumMtoN\_aviso.c

- SumMtoN\_aviso.c programa sumMtoN\_aviso.c sumMtoN.s moduluaren baliokidea den asm hizkuntza-moduluaren baliokidea garatzea eta errorea gertatuz gero abisu-mezu bat gehituz, 1. gehikuntzaren eta 2. gehiketaren parametroen arteko erlazio zuzena adieraziz.

## 20.5.2. modulua sumMtoN\_aviso.s

- Gehitu sumMtoN.s iturburu-programari abisu-mezu bat erlazioa adierazten duen akatsen kasuan zuzendu 1. gehigarria eta 2. gehigarria parametroen artean.

# 20.6. 6. praktika: Bit-maparen irudi eramangarria

## 20.6.1. C programazioa

- Irakurri programazio prozedura **LEEME.txt** fitxategian
- Helburua **bitmap\_gen\_test.c** jatorrizko programaren **main()** funtzi nagusia aldatzea da. bata bestearengandik independente diren programa ezberdinetara.
  - Konpilatu eta exekutatu bitmap\_gen\_test.c programa 2. -  
bistaratu test.bmp fitxategiaren irudia: **\$display test.bmp**
  - **Square\_128x128.c** modulua: Aldatu irudiaren neurriak 128 pixel x 128 pixelera, makro DIMENSIONA=128 definituz eta pixel bakoitzeko kolore gris bat definituz bere balio maximoaren %50eko intentsitatea duena.
  - **Karratuak\_4.c** modulua : Sortu 4 karratu, bata bestearen barruan simetriko, non karratu beltz handiena 512x512koa den eta gainerakoak 1/8tan murrizten diren bakoitzean. Ez erabili ctes C adierazpenetan, erabili makroak **x\_coor**, **y\_coor**, **top for** -en hasierako balioa eta errenkada eta zutabeen gehienezko posizioa (goian) adierazteko . Kolore karratuak: atzoko planoa (00-00-00)/(FF 00-FF)/(00-FF-FF)/(FF-FF-00)/
  - **bmp\_funcion.c** modulua: begizta egiten duen kode blokea pixelak hasieratzeko. karratua funtzi bihurtu:
    - ÿ prototipoa: void pixels\_generator (unsigned int x, unsigned int y, unsigned int gehienez, RGB\_data reg\_mem[][goian])
    - ÿ x eta y karratuaren koordenatuen jatorria dira
    - ÿ maximoa karratuaren koordenatu handiena da
    - ÿ funtzi-deia: pixels\_generator(xcoor,ycoor,top,buffer);
    - ÿ xcoor=top/8, ycoor=top/8 eta top=512 argumentuek makroak erabiliz definitzen dituzte.

## 20.6.2. Programazioa ASMn

- **bmp\_as.c** modulua: void pixels\_generator (unsigned int maximum, RGB\_data reg\_mem[][top]) funtzioa implementatu pixels\_generator azpierrutina garatuz array\_pixel.s fitxategi berrian **muntaia hizkuntzan**. Mihiztadura-hizkuntzako fitxategiak azpierrutina baino ez du edukiko.
  - ÿ Azpierrutinak begizta bikoitza implementatzen du.
  - ÿ Inplizituki azpierrutinan bertan x=y=0 argumentuak hartuko ditugu kontuan.
  - ÿ Urdina, gorria eta berdea karratuko pixel guztien intentsitateak dira.

## 20.6.3. GDB

- bmp\_funcion.c** programan adierazi pilaren posizioa non itzultzeko helbidea pixels\_generator **azpierrutina**, baita fotograma-erakuslearen eta pila-erakuslearen edukia ere.
- Aurreko atalean **bmp\_as.c** programarekin pixels\_generator azpierrutinarako

# VIII Eranskinak

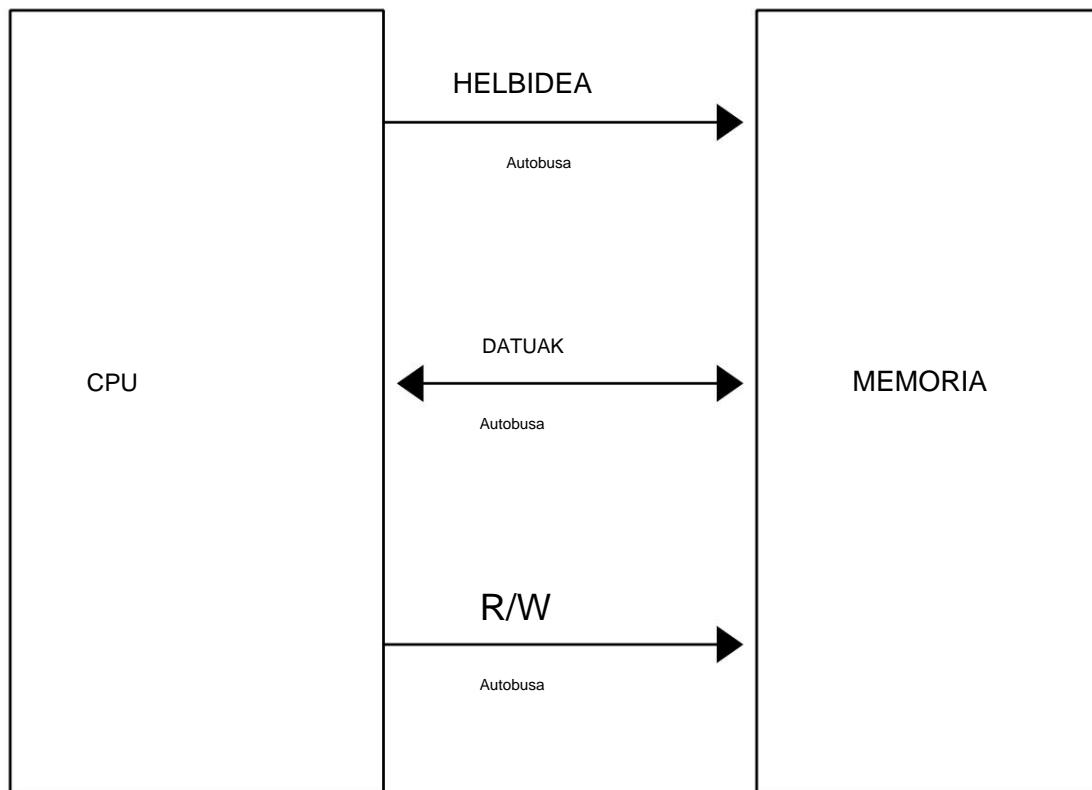
# 21. kapitula. Konputagailuen arkitektura

## 21.1. Konputagailuen Egitura

### 21.1.1. Testuingurua

- Ordenagailu batean arkitektura terminoaren esanahia testuinguruaren araberakoa da.
  - ÿ Lehenengo dibisio nagusia hau izango litzateke: Hardware Arkitektura eta Software Arkitektura
  - ÿ Konputagailuen arkitektura: bloke handietan dagoen antolaketa da, esaterako CPU, Memoria, kontrolagailu periferikoak, busak, etab.
  - ÿ Prozesadore baten arkitektura: PUZA interpretatzeko eta exekutatzeko gai den instrukzio eta datuen erreperiorioa da. Barne egitura ezberdin batekin fabrikatutako bi prozesadore egon daitezke baina instrukzio eta datu berdinak prozesatzen dituztenak, hau da, makina-lengoaia bera prozesatzen dute eta, beraz, arkitektura bera dute. Testuinguru honetan ISA (Instruction Set Architecture) terminoa erabiliko dugu. AMD64 prozesadore baten eta Intel x86-64 prozesadore baten ISA berdina da, makina-lengoaia berdinarekin funtzionatzen dute. AMD64-ko programa bitar bat Intel x86-64-rekin bateragarria da.
  - ÿ Mikroprozesadore baten mikroarkitektura: prozesadore baten barne antolaketa da.

### 21.1.2. HW arkitektura



### 21.1.3. CPU

- CPUaren oinarritzko osagaiak
  - ÿ ALU

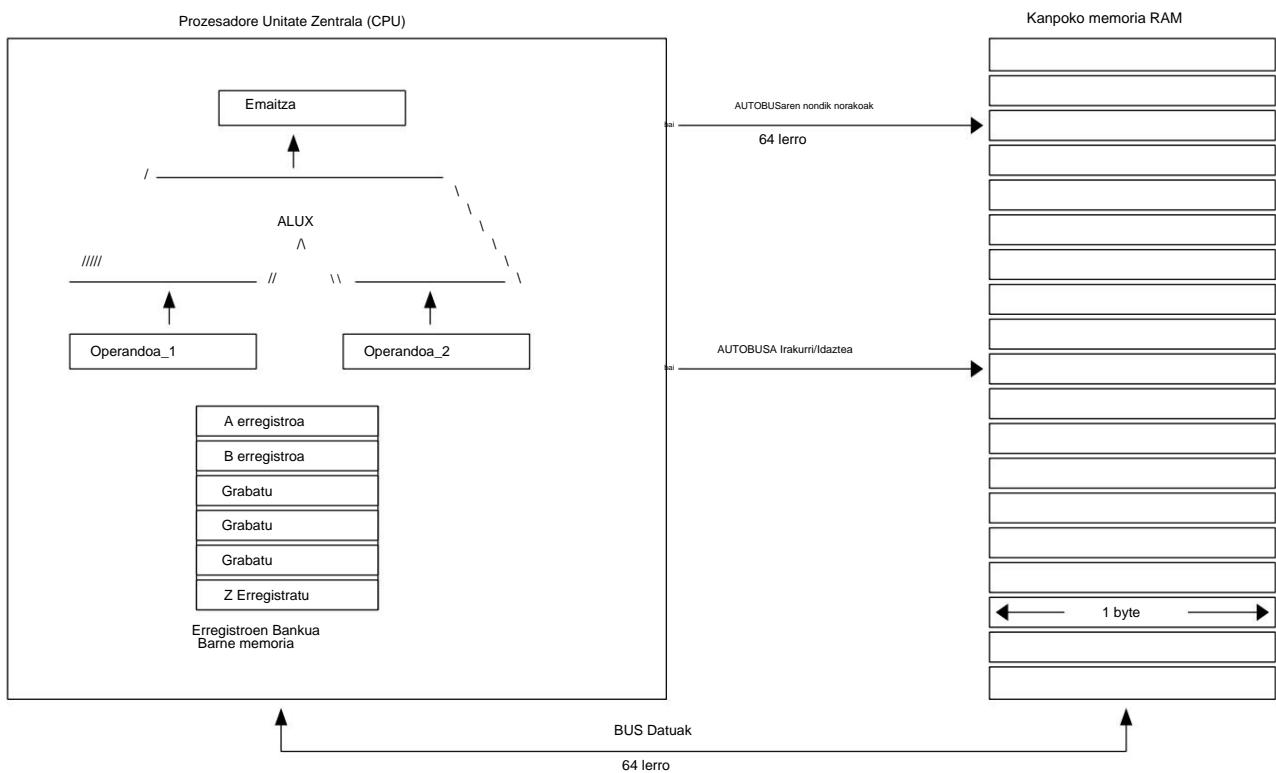
ÿ FPU

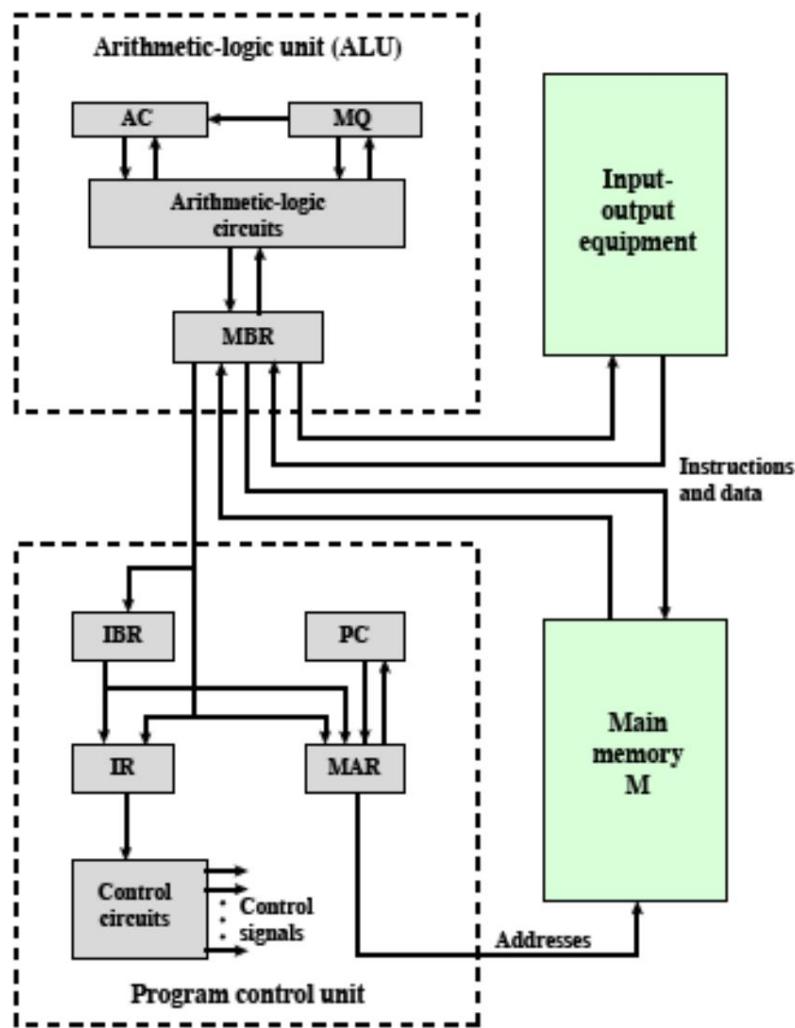
ÿ Kontrol Unitatea

ÿ Barne-erregistroak

- CPU funtzioa

ÿ Memoria nagusian gordetako jarraibideen instrukzio-zikloa egitea





**Figure 2.3 Expanded Structure of IAS Computer**

104. Irudia IAS Egitura

#### 21.1.4. Memoria

- Memoriaren hierarkia: PUZaren barneko eta PUZaren kanpoko memoria nagusia (DRAM) erregistratzen ditu

##### Oroimen nagusia

- DRAM Memoria: Ausazko Sarbide Dinamikoaren Memoria
- Makina bitarren jarraibideen eta datuen sekuentzia formatu bitarrean gordetzen du.
- Helbide-espazio lineala: idazkera hamaseimala
- Helbidea: byteak: idazkera hamaseitarra

## HELBIDEAK

## EDUKIA

0x00000000

1 0 1 0 1 0 1 0

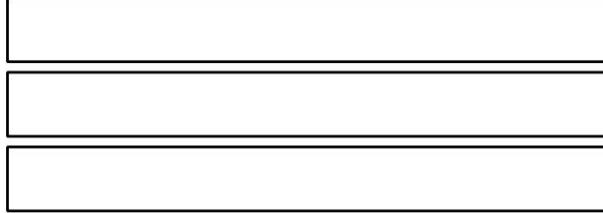
0x00000001

1 0 1 0 1 0 1 0

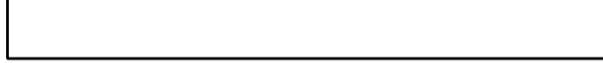
0x00000002

1 0 1 0 1 0 1 0

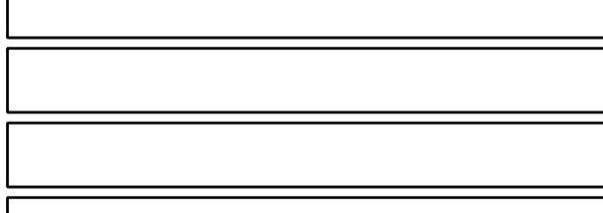
1 1 1 1 1 1 1 1



0x00000009



0x0000000a



0x0000000f



## CPUaren barneko erregistroak

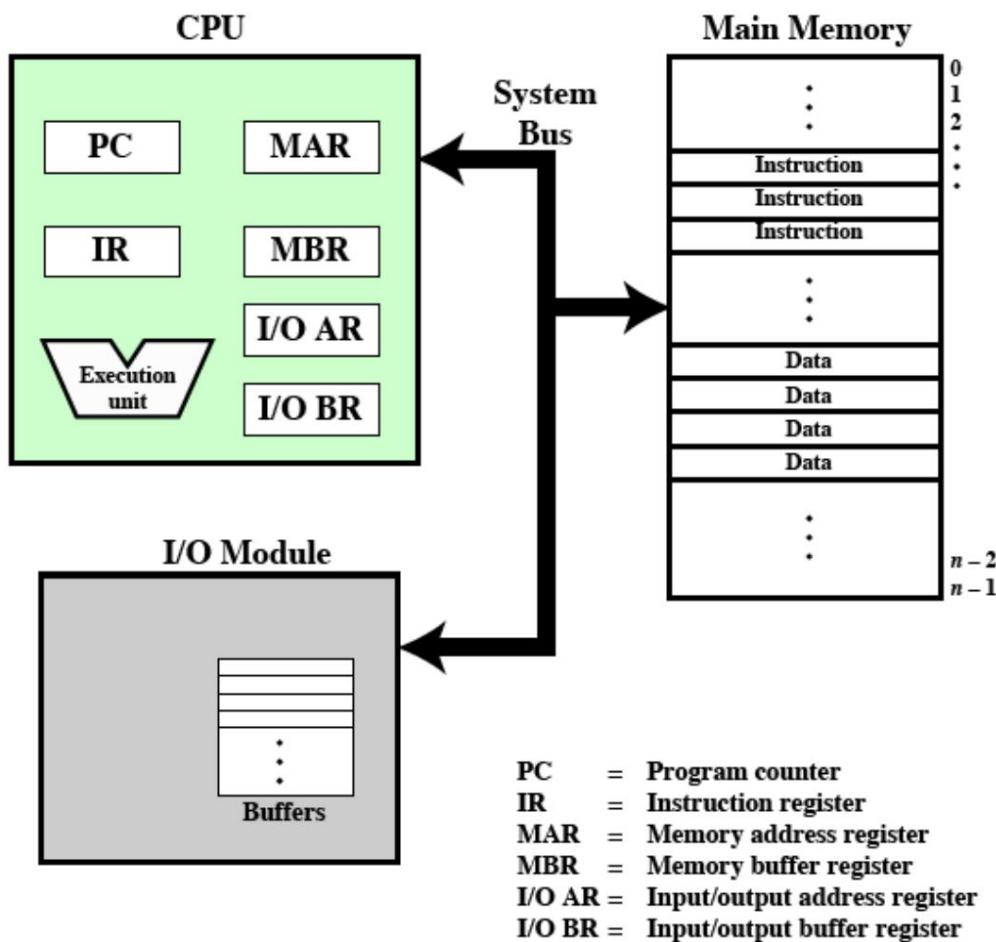
## Erregistroak EZ dira programatzailaik ikusgai

- Programatzailaik amd64 arkitekturan EZ atzi ditzake erregistroak
  - ÿ PC: Programa-kontagailua: x86-k RIP deitzen du: 64 bit
  - ÿ IR: Instrukzio-erregistroa: 64 bit
  - ÿ MBR: Memoria buffer erregistroa: 64 bit ÿ HITZA TAMAINA: 64
  - ÿ MAR: Memoria-helbideen erregistroa: 40 bit ÿ Memoriaren edukiera: 240 : 1TB
- I386 arkitekturaren kasuan, ordezkatu 64 bit 32 bitemkin eta MAR erregistroa ere 32 bitemkoa da.

## Programatzailaik ikusgai dauden erregistroak

- Programatzailaik erregistro hauek erabiltzen ditu datuak gordetzeko (idatzi eta irakurri).

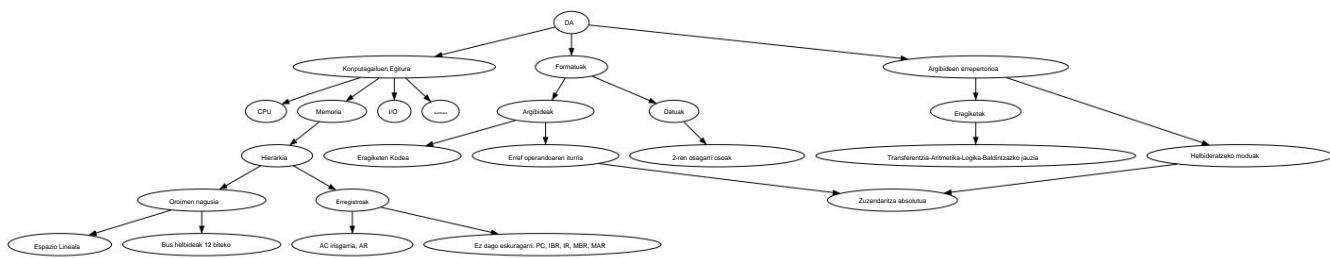
- Badaude eragiketak (baketa, kenketa, etab...) non eragigaiak erregistroetan egon daitezkeen.
- CPU erregistrorako sarbidea memoria-kokapen batera sartzea baino askoz AZKARAGOA da RAM nagusia.



105. Irudia IAS\_Arkitektura

## 21.2. Argibide multzoen arkitektura (ISA)

- Ordenagailu baten instrukzio-multzoen arkitekturak (ISA) definitzen ditu nagusiki:
  - ÿ Ordenagailuaren egitura: CPU-Memoria-Bus-I/O
  - ÿ Datuen irudikapena eta formatua.
  - ÿ Argibideen irudikapena eta formatua.
  - ÿ Instrukzio-errepertorioa: Ordenagailuak interpretatu eta exekutatu behar dituen eragiketak eta helbideratze-moduak.
- ISA arkitekturak konputagailuaren CPUaren potentziala definitzen du.
- ISA arkitekturaren diseinuak ordenagailuaren errendimenduan eragingo du.
  - ÿ Iturburu-programaren kompilazioaren ondoriozko programa bitarra.
  - ÿ Memoriaren okupazioa
  - ÿ Implementazioa (zaitasuna) eta CPUaren errendimendua.



## 21.2.1. Adibideak: Intel x86, Motorola 68000, MIPS, ARM

- Ikus [Eranskina Batzar Lengoaiak](#)

# 21.3. x86 arkitektura duten Intel prozesadoreak

## 21.3.1. Nomenklatura

### Orokorra

- Intel prozesadoreek izen ezagunak dituzte: Pentium II, Pentium III, Corei3, Corei5, Corei7, etab.
- Prozesadore hauek erabiltzen dituzten ordenagailuen arkitekturak arkitektura bati erantzuten dio ohikoa
  - ÿ x86 arkitektura 32 biteko arkitekturaren kasuan
  - ÿ x86-64 arkitektura 64 biteko arkitekturaren kasuan.
- 32 biteko x86 arkitektura duten prozesadoreak

\*\*\* 1978 eta 1979 Intel 8086 eta 8088. Lehenengo arkitekturako mikroprozesadoreak x86.

1980 Intel 8087. x86 arkitekturako lehen zenbakizko koprozesadorea, hasieran. x87 seriea.

1980ko NEC V20 eta V30. 8088 eta 8086 prozesadoreen klonak, hurrenez hurren, NEC-ek fabrikatua.

1982 Intel 80186 eta 80188. Hobekuntzak 8086 eta 8088.

1982 Intel 80286. Modu babestua agertzen da, zeregin anitzeko gaitasunak ditu.

\*\*\* 1985 Intel 80386. Lehenengo 32 biteko x86 mikroprozesadorea.

1989 Intel 80486. Zenbakizko koprozesadorea txertatzen du zirkuituan bertan integratua.

1993 Intel Pentium. Errendimendu hobea, arkitektura supereskalarra.

1995 Pentium Pro. Ordenantzaz kanpoko exekuzioa eta exekuzio espekulatiboa  
1996 Amd k5. Intel Pentium-en arerio zuzena.

1997 Intel Pentium II. 16 biteko kodean abiadura hobetzen du, MMX barne hartzen du

1998 AMD K6-2. Intel Pentium II-ren lehiakide zuzena, 3DNow!

1999 Intel Pentium III. SSE argibideen sarrera

2000 Intel Pentium 4. NetBurst. SSE argibideak hobetuak

2005 Intel Pentium D. EM64T. Bit NX, Intel Viiv

- 64 biteko x86-64 arkitektura duten prozesadoreak

\*\*\*2003 AMD Opteron. Lehenengo 64 biteko x86 mikroprozesadorea, multzoarekin

AMD64 argibideak)

2003 AMD Athlon.

2006 Intel Core 2. Intel P8 Mikroarkitektura Sarrera. Kontsumo txikiagoa, nukleo anitz, SSSE3 hardware birtualizazio euskarria.

x86-64 barne eta

2008 Core i7

2009 Core i5

2010 Core i3

- [Intel Core serieko prozesadoreen kode-izenen esanahia](https://www.intel.com/content/www/us/en/products/processors/core/view-all.html): Kodearen lehen digituak belaunaldia adierazten du (8. belaunaldia 2017an kaleratu zen)

- Intel Core seriea:

    ŷ <https://www.intel.com/content/www/us/en/products/processors/core/view-all.html>

- Intel Core X seriea: i9, i7, i5 familiak

    ŷ Intel seriea X:

        ŷ <https://ark.intel.com/products/series/123588/Intel-Core-X-series-Processors>

- [https://es.wikipedia.org/wiki/Anexo:AMD\\_Processors](https://es.wikipedia.org/wiki/Anexo:AMD_Processors)

- INTEL-AMD LEHIAKETA urtea 2018 mahaigaineeko ordenagailuetan.

    ŷ AMD Ryzen 2. belaunaldia - INTEL Core i7-8086K

## 22. kapitulua. RTL Erregistroaren transferentzia hizkuntza

### 22.1. RTL hizkuntza

#### 22.1.1. Sarrera

- JARRAIBIDEAK deskribapen hizkuntza: Erregistro transferentzia hizkuntza (RTL)
- RTL hizkuntzak CPUak exekutatzen dituen instrukzioak adierazteko gai izan nahi du, hala nola gehitu (GEHITU), kendu (SUB), mugitu (MOV), etab. Deskribapena PUZaren barne-erregistroen artean edo barne-erregistroen eta kanpoko memoriaren arteko datu-transferentzia mailan egiten da.
- RTL hizkuntza , Programatzailak interpreta daitezkeen ikurrak erabiliz, hardware mailan bere portaera deskribatzeko aukera ematen du eta, horrela, makina baten arkitekturaren diseinua definitzeko aukera ematen du.
- Erregistroak memoriaren oinarrizko elementua dira ordenagailuaren oinarrizko unitate ezberdinen arteko datuen eta argibideen ibilbidean. Erregistroa datuak gordetzen, memorizatzen eta gordetzen dituen zirkuitu digitala da.
- Datuen bidea busek eta elementuek osatzen dute (erregistroak, multiplexadoreak, etengailuak, metroak, etab.) autobusen bidez konektatuta daudenak
  - ÿ Adibidea: memoria nagusiko kokapen batetik eragiketa erregistroetara datu baten bidea ALUren.
  - ÿ Buffer kontzeptua: datuen bideko memoriaren tarteko etapa.
- PUZak exekutatutako instrukzioak exekutatzeak datuen transferentzia dakin datu-bideen erregistroak.

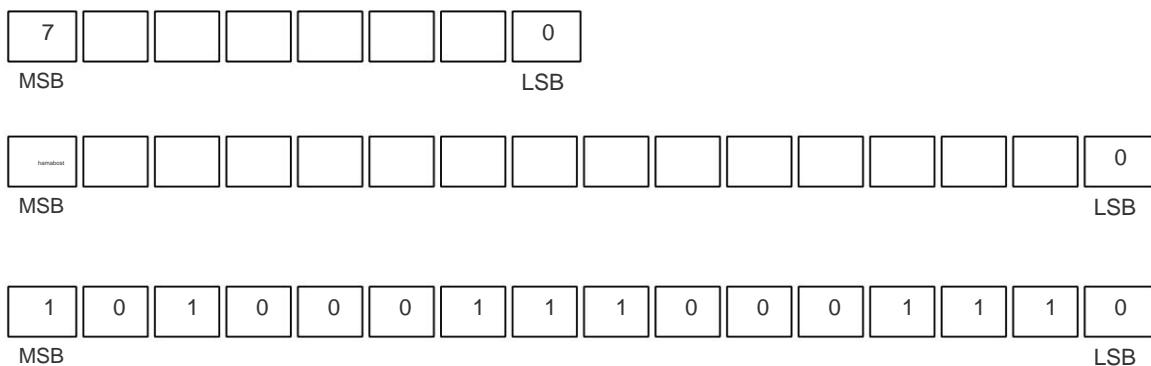


Ez nahastu RTL (Register Transfer Language) eta RTL (Register Transfer Level). Erregistro transferentzia maila HARDWARE deskribapen lengoaietan erabiltzen dute (Hardware deskribapen hizkuntza HDL)

#### 22.1.2. Erregistroak

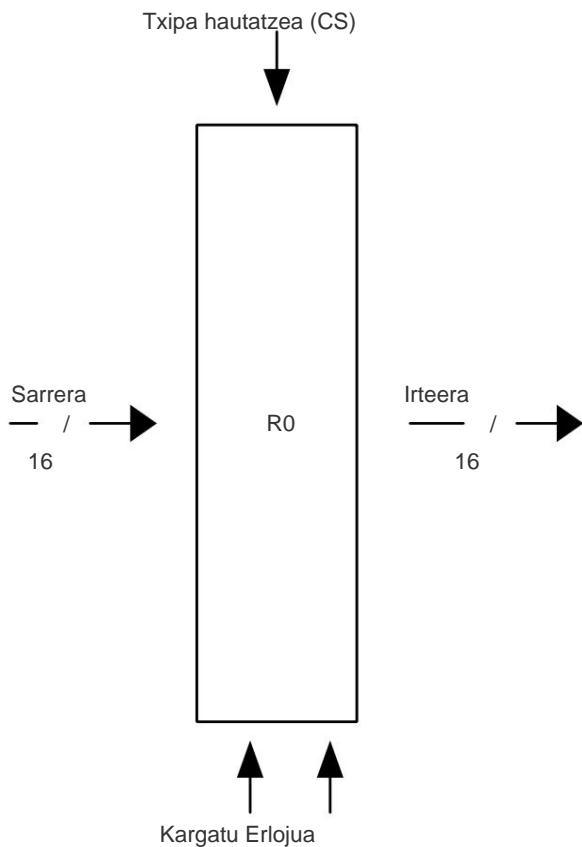
##### Arkitektura

- Erregistro baten arkitekturak bere funtzionaltasuna barne hartzen du eta bere ezarpena egituratzen du.
- Erregistroak:
  - ÿ bit-segida batez osatutako hitza gorde.
  - ÿ dimentsio bateko gelaxka-matrize bat dira, non gelaxka bakoitzak bit bat gordetzen duen.
- Bere tamaina 8 byteko multiploa izan ohi da eta izena ematen zaio.
  - ÿ 8 bit: 1 byte
  - ÿ 16 bit: Word. Arrazoi historikoengatik (gogoratu hitz baten tamaina beste testuinguru batean kasuan kasuko makinaren araberakoa da)
  - ÿ 32 bit: hitz bikoitza
  - ÿ 64 bit: hitz laukoa
- Gelaxkak zerotik hasita zenbatzen dira.
  - ÿ LSB: Least Significant Bit pisu gutxien duen bit da
  - ÿ MSB: Bit esanguratsuena pisu handiena duen bit da



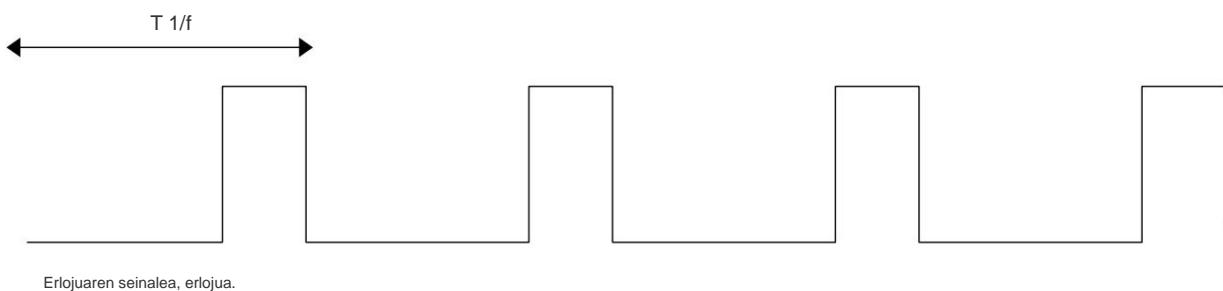
### Egitura

- Erregistroaren egitura bere funtzionalitatearen ezarpena da
- Grabatu beharreko datuen bit bakoitza memoria ahalmena duen gelaxka batean gordetzen da. A-ren zelulak erregistroa flip-flop izeneko zirkuitu digital batekin ezartzen dira. Flip-flop bakoitzak bit bat gordetzen du.



- Erregistroa kanpoko mundura konektatzen da busen bidez: sarrera-busa eta irteera-busa
- CS: Txipa hautatzea: R0 erregistroaren barne irteera irteerako busera konektatzen du ÿ Irakurri eragiketa erregistroa
- Kargatu: seinalea aktibatuta badago, sarrerako busaren balioa R0 erregistroan kargatzeko agindua ematen da, eta grabatzen da. sarrerako datuak. Idatzi eragiketa erregistroan.

- Erlojua: aldizkako seinale digital bitarra.
- Karga erloju-seinalearen CLK erlojuarekin sinkronoa da. Sinkronismoa ertz positiboetan gertatzen da  $\_|\backslash$  edo negatiboa



Erlojuaren seinalea, erlojua.

#### 22.1.3. Sinboloak

- Erregistroaren izenak letra larriz adierazten dira
  - ÿ PC: Programa-kontagailua
  - ÿ IR: Instrukzio Erregistroa
  - ÿ R2: 2. erregistroa
- Erregistro baten atalak
  - ÿ PC(L): programa-kontagailuaren erregistroko byte baxuena
  - ÿ PC(H): programa-kontagailuaren erregistroko byte altuena
  - ÿ PC(7:0): Bit-sekuentzia zero posiziotik kontagailu-erregistroko zazpi posiziora. programa.

#### 22.1.4. RTL perpausak

##### RTL eragiketak eta esaldiak

- RTL hizkuntzan esaldiz ulertzen dugu eragiketak egiten dituen esamolde bat erregistroak.
- RTL eragiketak:
  - ÿ Erregistroen arteko transferentziak, bi erregistroren edukia gehitzea, baten edukia alderantziz. erregistroa, etab.

##### Mikrooperazioa

- MIKROeragiketak: MIKROprozesadoreak barnean egiten dituen eragiketak, a exekutatzen denean Makinen Instrukzioa.
  - ÿ Adibideak: erregistro batean idaztea, M Nagusira irakurtzeko ordena, erregistro batetik irakurtzea, instrukzio bat deskodetzea, kontagailu bat gehitzea, gehitza (mikroordenatuak batutzaile-zirkuituari), erregistro baten bitak aldatzea, ETA logika, etab.
- Erregistro batean idazteko edo erregistro batetik CPUrak irakurtzeko eragiketa mikroeragiketa bat da.

##### Erregistroen arteko transferentzia

- Transferentzia operadorea ÿ
- Transferentzia adierazpena: R2ÿR1

• R1 iturburu-erregistroa deitzen da eta R2-ri helmugako erregistroa

• R1 erregistroko edukia R2 erregistrora kopiatzen dugu

### Baldintza perpausa

- (K1=1) bada R2  
• K1:R2

• Transferentzia edo kopia K1 egia bada bakarrik egiten da, hau da, K1 1 balio logikoa da.

### Aldibereko Perpausa

- Koma eragilea
- K3:R2,R3  
• K3 egia bada, R1-en edukiak R2 eta R3-n kopiatzen dira

### Memoria Nagusiaren erreferentzia

- Kortxeteak eta M ikurra erabiltzen dira.
- M[0x80000] : memoria-kokapenaren edukia 0x8000
- AC ÿ M[0x80000] : kopiatu 0x8000 posizioko memoriaren edukia AC erregistrora
- AC ÿ M[AC]: AC erregistroak adierazitako memoria-kokapenaren edukia kopiatu erregistrora.  
A.C.
- M[0x8000] ÿ AC: kopiatu AC erregistroko edukia Memoria posiziora 0x8000  
• M[0x8000] ÿ R[AC]: kopiatu AC erregistroaren edukia Memoria kokapenean 0x8000

### Ezker-eskuin balioa

- Kontzeptu hau C hizkuntzan erabiltzen da = sententzia-esleipena definitzerakoan
- M[0x1000] ÿ M[0x2000]
  - 0x2000 posizioaren edukia 0x1000 posizioan kopiatzen da
  - ÿ eragilearen eskuinean dagoena ebaluatzen da eta BALIO bat lortzen da
  - Eragilearen ezkerrean dagoena ÿ Memoriaren HELBIDEA edo ERREFERENTZIA da (Nagusia edo Grabatu)

## 22.1.5. RTL adibideak adierazpen aritmetiko-logikoekin

- AC ÿ R1 v R2
  - EDO eragiketa logikoa
- (K1+K2):R1 ÿ R2+R3,R4  
• + sinboloak bi esanahi ditu: boolearra edo aritmetikoa.
  - K1+k2-n esanahi boolearra du: edo. Seinale logikoen batuketa aritmetikoak ez du zentzurik hemen. Zentzuzkoa da seinaleak aktibo dauden edo ez ebaluatzea.
  - R2+R3-n esanahi aritmetikoa du.
- Esapide batean lehentasuna adierazteko parentesiak erabiliko ditugu.

# 23. kapitulua. IASSim muntatzaile-programak

## 23.1. 1. adibidea: sum1toN.ias

- **Demo bertsioa** tutorial.ias: simulagailuaren deskarga fitxategian sartutako demo bertsioa.

```

begizta: S(x)->Ac+ n           ;kargatu n AC-ra
          Cc->S(x) pos ; AC >= 0 bada, jauzi pos
          gelditu ;bestela eginda
          .hutsik           ; 20 biteko 0
pos:    S(x)->Ah+ batura ;n gehitu baturari
          At->S(x) batura ;guztira itzuli batura
          S(x)->Ac+ n ;kargatu n AC sartu
          S(x)->Ah- bat ;gutxitu n
          At->S(x) n ;denda dekrementatua n
          Cu->S(x) begizta ;atzera egin eta berriro egin
n:      .data 5 ; guztira 6 aldiz biraka egingo du
bat:   .datua 1 ; gutxitzeko konstantea n
batura: .datuak 0 ;non exekutiboa/azkeneko guztirakoa gordetzen den

```

- **Demo Version** tutorial.ias -arekin adibidea :

ÿ aldatu iturburu-moduluaren izena: sum1toN.ias  
 ÿ programa berriro editatu etiketa eta iruzkinekin gaztelaniaz.  
 ÿ kodearen iruzkinak garatzeko aurreko faseetan deskribatutako moduluetako informazioarekin  
 programa

```

;::::::::::::: OHARRA
; iturburu-modulu sum1toN.ias
; Zenbakia osoen segida baten batura kalkulatzen du: batura = 1+2+..+n
; sarrerako datuak: N eta irteerako datuak: batura
; Algoritmoa: N iterazioen begizta
;               Gehigarriak na -1etik beheranzko norabidean sortzen dira
;               Gehitzalea negatiboa bada -> -1 , gehitzea ez da egiten eta amaitzen da
; begizta
; Datuen egiturak: n eta batura aldagaiak . Konstante bat.
; Mihiztadura hizkuntza: IASSim
; ISA: Von Neumann IAS Makinen Arkitektura

;::::::::::::: JARRAIBIDEEN ATALA
;Metagailuetara zuzendutako arkitektura (AC)
;Erregistro eskuragarriak: AC
; algoritmoa: n, n-1, .... -1 gehigarriak sortzen dituen begizta
;               eta egin batura = n + batura eragiketa n>=0 bada

; loop start: gehigarrien batuketa eta sorrera
begizta: S(x)->Ac+ n ;karga gehitzea
          Cc->S(x) gehitu; gehigarria < 0 begiztaren amaiera bada
; begiztaren amaiera

```

```

; gelditu
;20 biteko Ora, argibide kopurua bikoitia izan dadin.

gelditu .hutsik ; egin batura
gehitu: S(x)->Ah+ batura ;
At->S(x) batura ;
; eguneraitzea gehituz
S(x)->Ac+ n ;
S(x)->Ah- bat ;
At->S(x) n ;           ;
hurrengo iterazioa
Cu->S(x) begizta ;

:::::::::::::DATU ATAL
; Etiketa-deklarazioa, kanpoko memoria-erreserba, hasieratzea.
; Aldagai arruntak
n: .datuak 5 ; gehitza eta hasieratza
batura: .datuak 0 ; batuketa partziala eta amaierakoa
; konstanteak
bat: .datuak 1           ;

```

### 23.1.1. 2. adibidea: Produktua/Kozientzia

#### Adierazpena

- Eragiketa egiten duen programa garatu  $N(N + 1) / 2$ -ren emaitza lortzearen baliokidea

$$\text{Tutorialaren batura} \sum_{i=1}^N i = N(N + 1) / 2.$$

ÿ Algoritmoaren pseudokodea

ÿ Algoritmoaren fluxu-diagrama

ÿ Programa RTL hizkuntzanÿ Komentatu behar bezala programari (goiburuarekin metainformazioa, egitura-atalak, bloke funtzionalak).

ÿ Mihiztadura-lengoaiaaren programa

ÿ Programa exekutatu pausoz pauso biderketak egiterakoan erregistroen balioa aztertuz eta zatiketa.

- RTL deskribapenean KONTUAN HARTU:

ÿ M zifrako bi zenbakiren biderkadurak 2M zifrako zenbakia sortzen du, hau da, the biderkatzaileen bikoitza. Horrek zaildu egiten ditu biderketaren ondoren eragiketa aritmetikoak adierazpen matematikoa. Horregatik biderketa eragiketa azkenerako utziko dugu, lehentasuna emanez batuketa eta zatiketa

$$N(N + 1) / 2 = ((N + 1) / 2) * N$$

ÿ Zenbaki oso baten 2z zatitzeak 1 edo 0 hondarra izan dezake, dibidetu bikoitia edo bikoitia denaren arabera. bakoitia

ÿ N bakoitia bada ÿ (N+1)/2 hondarra 0 ÿ du  $((N + 1) / 2) * N$  non N+1 bikoitia den

ÿ N bikoitia bada ÿ (N+1)/2 hondarra 1 badu ÿ (N+1) = zatidura\*2+hondarra ÿ non  $N(N + 1) / 2 = N * C + N / 2$  N bikoitia den

ÿ 21 bezalako 2 potentziatz zatitzea eragiketa logiko baten bidez egiten da: desplazatu 1 bit utzi zuen dibidendua. Desplazatu beharreko bit kopurua berretzailearen balioa da.

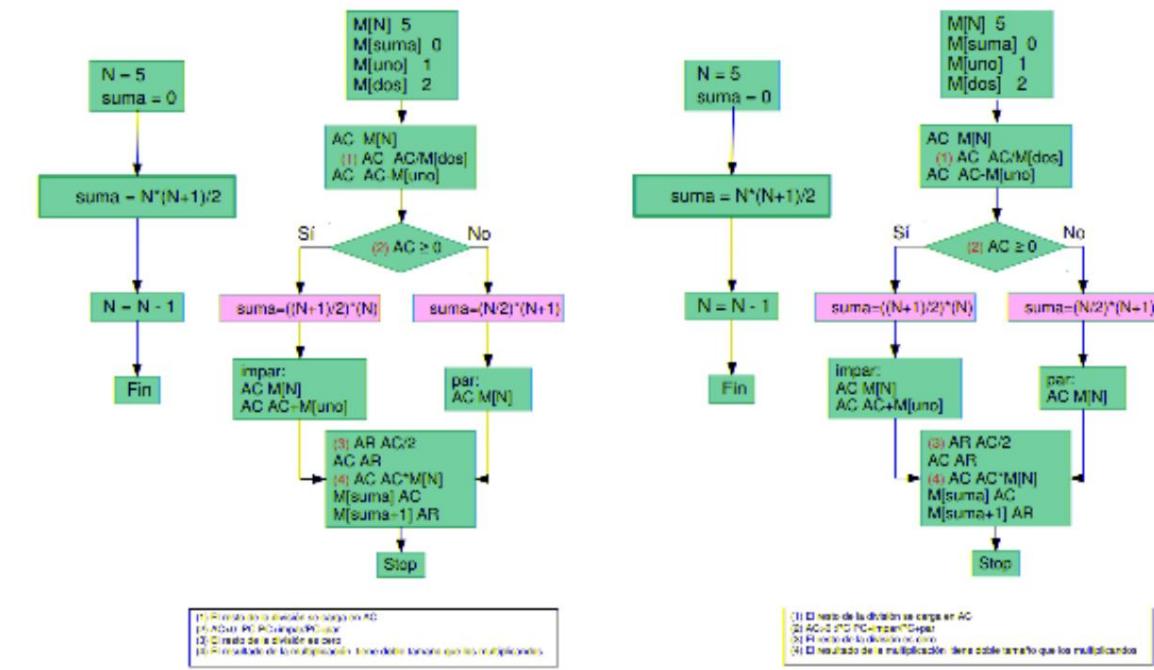
ÿ deskribapena RTL AC ÿ AC<<1

### Pseudokodea

- Algoritmoaren deskribapena hizkuntza NATURALEko testu-adierazpenen bidez
- ALDAGAIAK:
  - ÿ batura aldagaia: emaitza partzialak eta amaierakoak gordetzen ditu
  - ÿ N aldagaia: sarrerako datuak gordetzen ditu
- Kode inperatiboaren egitura:
  - ÿ Oinarrizko instrukzioaren eraikuntza zeregina da
    - $suma = N(N + 1) / 2$

### Organigramak: Goi Mailako eta RTL

- Algoritmoaren deskribapen grafikoa:
  - ÿ Goi Maila: hizkuntza naturala ezinbesteko
  - ÿ RTL: ordenagailuaren ISA kontuan hartzen duen maila baxuko lengoaia



106. Irudia. Fluxu-diagrama: Pseudokodea eta RTL

### IAS batzar hizkuntza

- sum1toN\_mul\_A.ias bertsioa

; Lehenengo N zenbaki osoen batura. Y=N(N+1)/2; IAS  
CPU

; mutua-lengoia: IASSim simulagailua  
; William Stalling-en, Structure of Computers liburuko 2.1 ariketa

; JARRAIBIDEEN ATALA  
; N berdina al da? -> Gainontzeko N/2  
S(x)->Ac+ n ;01 n ;AC <- M[n]  
.  
.  
; 1. kasua: zenbaki bikoitia  
.  
.  
; 2. kasua: N bakoitia  
.  
.  
; Biderketa N(N+1)/2  
.  
.  
; DATU ATALA  
; Aldagaien adierazpena eta hasieratzea  
eta: .datuak 0 ;emaitza  
.  
; Konstanteen Adierazpena  
n: .datuak 5 ;N parametroa  
bat: .datuak 1  
bi: .datuak 2

- bertsio sinplifikatua sum1toM\_mul.ias: N(N+1) biderketa egiten dut lehenengo eta emaitza beti da bikotea. Ondoren, 2rekin zatitzen dut.

; Lehenengo N zenbaki osoen batura. Y=N(N+1)/2  
; IAS CPU  
; mutua-lengoia: IASSim simulagailua  
; William Stalling-en, Structure of Computers liburuko 2.1 ariketa

; JARRAIBIDEEN ATALA  
S(x)->Ac+ n ;01 n ;AC <- M[n]  
S(x)->Ah+ bat ;05 bat ;AC <- AC+1  
At->S(x) y ;11 y ;M[y] <- AC  
S(x)->R y <- M[y] ;09 y ;AR  
S(x)\*R->A n ;0B n ;AC:AR <- AR\*M[n]  
; AC=0 kasu berezia  
R->A ;0A A/S(x)->R ;AC <- AR  
bi ;0C 2 ;Dibidendua ;AR <- AC/2  
bikoitia denez, hondarra zero da  
R->A ;0A ;AC <- AR  
At->S(x) y ;11 y ;M[y] <- AC  
gelditu  
; Argibide kopurua bikoitia denez, ez da beharrezkoa .empty zuzentaraaua.

```

; DATU ATALA;
Aldagaien adierazpena eta hasieratzea .datuak
eta: 0 ;emaitza

; Konstanteen Adierazpena .data
n:      5 ;parametroa N bat: .data
1 bi: .data 2

```

## simulazioa

- simulazioa IASSim emuladorearekin

### 23.1.2. 3. Adibidea: Bektoreak

#### Adierazpena

- Egin batura  $C = A + B$  10 elementutako A eta B bi bektoreen batura, biak hasierakoarekin  
1etik 10era bitarteko balioak.



Bektore baten elementu bakotzean sartzeko beharrezko da eragigaiaren memoria-helbide absolutua handitu bektorearen elementuetara sartzen den instrukzioan, beraz, beharrezko da instrukzioaren eragigaien eremua aldatzea. Eragiketa eremuko 12 bitak AC erregistroko 12 bit txikienetara transferitzeko instrukzio bat dago, hau da, AC(28:39)  $\ddot{y}$  M[operandoa](8:19). Eta alderantzizko transferentzia egiten duen beste instrukzio bat M[operandoa (8:19)]  $\ddot{y}$  AC(28:39). Modu honetan, eragiketa aritmetikoak eta logikoak egin daitezke instrukzio baten operando-eremuaren 12 bitetan.

- Algoritmoaren pseudokodea
- Algoritmoaren fluxu-diagrama
- Programa RTL hizkuntzan  $\ddot{y}$  Komentatu behar den programari (goiburuarekin metainformazioa, egitura-atalak, bloke funtzionalak).
- Programa muntatzailean: Programa zuzenean osorik ez egitea komeni da faseka, bertsio simple batetik hasi eta azken bertsioan programa osatu arte. Adibidez:

- $\ddot{y}$  1. bertsioa:  $A[i]=i$  bektorea hasieratu
- $\ddot{y}$  2. bertsioa:  $A[i]=i$ ,  $B[i]=i$ ,  $C[i]=i$  bektoreak hasieratu
- $\ddot{y}$  3. bertsioa:  $C[i]=A[i]+B[i]$

$\ddot{y}$  Aldagai posibleak: len: bektorearen luzera, A0: A bektorearen lehen elementuaren helbidea, i: bektorearen indizea.  
bektorea, etab.

- Exekutatu programa urratsez urrats programaren bertsio desberdinak arazketan.

#### Pseudokodea

- Algoritmoaren deskribapena hizkuntza NATURALEko testu-adierazpenen bidez
- ALDAGAIAK:

$\ddot{y}$  A,B,C aldagai bektorialak: Adierazi eta hasieratu  $A[i]=i$ ,  $B[i]=i$ ,  $C[i]=0$   
 $\ddot{y}$  len aldagai: bektoreen tamaina gordetzen du

ÿ A0 aldagaia – A bektorearen lehen elementuaren helbidea gordetzen du

ÿ i aldagaia: edozein bektoreren i posizio-elementuaren indizea

- Kode inperatiboaren egitura:

ÿ Oinarrizko instrukzioen eraikuntza begizta bat da

ÿ Begiztak iterazioak beherantz zenbatzen ditu

ÿ "i"=len-1 indizea hasieratzen da eta

ÿ Iterazio bakoitzean  $A[i]=i$  esleitzen da,  $B[i]=i$ ,  $C[i]=A[i]+B[i]$

ÿ Iterazio bakoitzean  $i=i-1$  indizea eguneratzen da

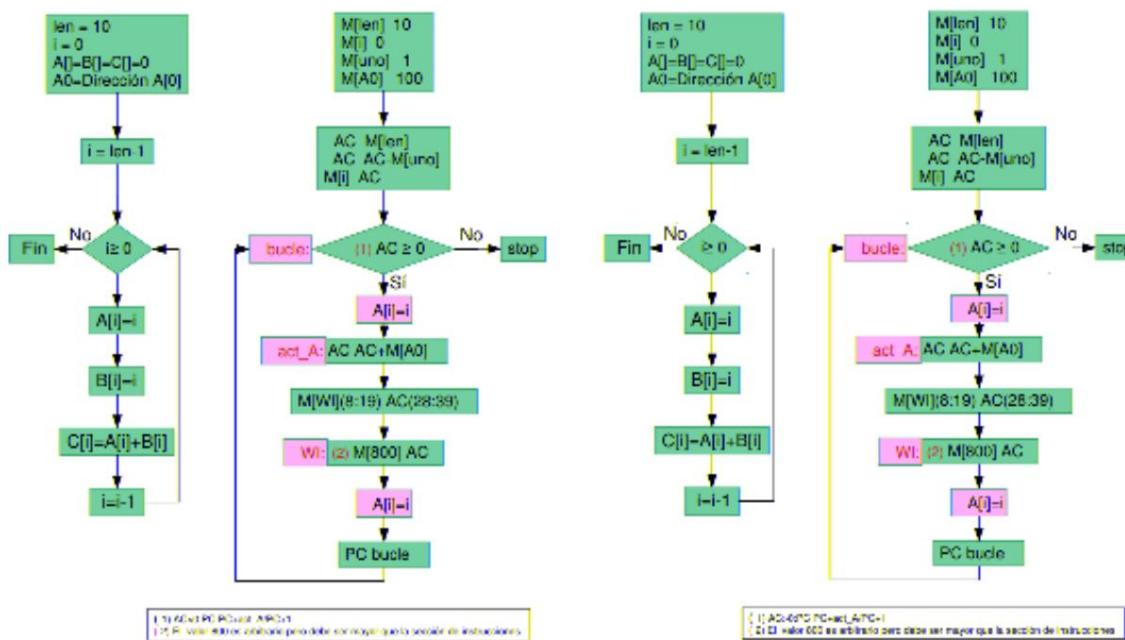
ÿ Begiztak irteten da  $i=-1$  denean

Organigramak (1. bertsioa): Goi Mailako eta RTL

- Algoritmoaren deskribapen grafikoa:

ÿ Goi Maila: hizkuntza naturala ezinbestekoa

ÿ RTL: ordenagailuaren ISA kontuan hartzen duen maila baxuko lengoia



107. Irudia. Fluxu-diagrama: Pseudokodea eta RTL

### Organigrama (2. bertsioa): RTL

- Aukera bat: A[], B[] eta C[] 3 bektoreak hasieratu

### Organigrama (3. bertsioa): RTL

- Behin betiko bertsioa: C[] = A[]+B[] bektorea

## IAS batzar hizkuntza (1. bertsioa)

- bektore\_hasiera\_A.ias

; vector\_start\_A.ias  
; Hasieratu A bektorea  
; A sekuentzian gordetzen den "len" tamainako bektorea da. Norabidea  
A-ren lehen elementuaren A0 aldagaian gorde zen  
; A[i]=i bektorea hasieratzen dugu  
; Arrayko elementuetarako sarbidea eremuan idatziz egiten da  
irakurtzeko/idazteko instrukzioaren helbideak.  
; Ezkerreko argibideek bakarrik izan ditzakete etiketak  
instrukzioa Cu'->S(x) etiketa erabili beharko da [Jauzi instrukziora  
eskuineko posizio-etiketan] eta Cu'->S(x) etiketa [Jauzi instrukziora  
etiketaren posizioan utzita] etiketa guztiak LERROKATZEKO  
argibideak utzi.  
; Begizta, batura eta C helbideen argibideak ote diren jakin behar da  
memoria hitzaren ezkerrera edo eskuinera.  
; Argibide kopurua bikoitia izan behar da. Erabili .hutsa kasu bakoitietan  
; azenturik ez iruzkinetan  
; Online laguntza: erreferentzia eskuliburua -> datu motak  
; Ikusi -> Hobespenak -> Selectron Memory Capacity

.....;JARRAIBIDEEN ATALA

```
;;;;;
; i = len - 1 indizea hasieratzen dut
eskuin1: Cu'->S(x) eskuin1      ; eskuineko eskuinera jauzi
S(x)->Ac+ len      ;
S(x)->Ah- bat      ;
At->S(x) i      ;
```

;;;;;; start while : baldintza elementua > 0  
begizta: Cc->S(x) actu\_A ;AC >= 0 bada, joan Actu\_Ara  
Cu->S(x) amaiera

; ; ; ; ; A[i]=i bektorea eguneratzen dut  
; Erakuslea A[i]-ra eguneratzen dut  
actu\_A: S(x)->Ac+ zero S(x)- ;  
>Ah+ A0 S(x)->Ah+ ;hasi erakuslea A[0]-rekin  
i ;erakuslea A[0]+i-rekin hasieratu  
Ap->S(x) wa ;Eguneratu LEFT instrukzioaren helbidearen eremua  
"wa"-n kokatuta M[wa](8:19) <- AC (28:39)  
; A[i]=i eguneratzen dut  
S(x)->Ac+ i Cu- ;  
>S(x) wa wa: At- ; jauzi wa-ren ezkerrera  
>S(x) 100 exekuzioa. ;MI[100]<-AC. Helbideratu aldi berean 100 aldaketa

..... Hurrengo iterazioa

```

S(x)->Ac+ i S(x)-          ;
>Ah- bat At->S(x) i        ;
Cc->S(x) begizta          ;

.hutsik
amaiera: gelditu
.hutsik

;:::::::::::::DATU ATAL

;;;;;; aldagai arruntak
len: .datuak 10           ; luzera-bektoreak A[], B[] eta C[]
      .datuak 30           ; helbidea A[0]
A0: i:      .datuak 0       ; array-indizea

;;;;;; konstanteak
bat:           .datuak 1       ;
zero:          .datuak 0

```

### **Simulazioa (1. bertsioa)**

- Kode bitarra IASsim emuladorearekin exekutatu

### **IAS batzar hizkuntza (2. bertsioa)**

- Vector\_iniciar\_A\_B\_C.ias programaren iturburu kodea garatu:

### **Simulazioa (2. bertsioa)**

- Kode bitarra IASsim emuladorearekin exekutatu

### **IAS batzar hizkuntza (3. bertsioa)**

- VectorA+B.ias programaren iturburu kodea garatu:

### **Simulazioa (3. bertsioa)**

- Kode bitarra IASsim emuladorearekin exekutatu

# 24. kapitula. IASSim simulagailua

## 24.1. Java makina birtuala JVM

- Instalatu Java Garapen Kita ([Java Development Kit-JDK](#)) ubuntu sisteman

ÿ [openjdk-11-jdk](#) Linux/GNU Ubuntu 18.0 banaketa bionikoan.

ÿ Egiaztu paketerako sarbidea duzula: [apt-cache search openjdk-11-jdk](#)

ÿ Instalatu paketea: [sudo apt-get install openjdk-11-jdk](#)

ÿ Egiaztu paketea instalatuta dagoela: [dpkg -l openjdk-11-jdk](#)

ÿ Egiaztu instalatutako java bertsioa: [java --version](#)

- Instalazio datuak Ubuntu 17n

Eguna: 2017ko irailaren 15a.

Emulatzailearen bertsioa: IASSim2.0.4

Emulatzailearen komandoa: `java -cp IASSim2.0.4.jar:jhall.jar:IASSimHelp2.0.jar iassim.Main -m IAS.cpu Sistema eragilea: GNU/linux`

Banatzailearen IDa: Ubuntu Deskribapena:

Ubuntu 17.04 bertsioa : 17.04 Kode-

izena: zesty Java bertsioa: openjdk bertsioa  
"1.8.0\_131"

OpenJDK Runtime Environment (1.8.0\_131-8u131-b11- eraikitzea)

2ubuntu1.17.04.3-b11)

OpenJDK 64 biteko zerbitzariaren VM (25.131-b11 eraikitza, modu mistoa)

IASSim Java JVM makina birtualean exekutatzen da, beraz, beharrezkoa da JVM makina birtuala instalatuta edukitzea. Makina bat birtualizazioa edozein Sistema Eragileren gainean (Linux, MacOS, Windows) SW geruza bat instalatzean datza, birtualizazio geruzan instalatutako edozein aplikazio (Adibidez IASSim) sistema eragilearen menpe egon ez dadin eta horrela. Iortzen da.aplikazioa (adibidez IASSim) Sistema Eragile ezberdinatik independentea izatea.

## 24.2. IAS simulagailua

- IASSim: Von Neumann IAS ordenagailu bidezko simulazio-tresna erabilgarria da programa baten instrukzioen exekuzioa urratsez urrats makina-kodean simulatzeko. Instrukzio-ziklo bakoitzen amaineran Selectron memoria nagusiaren eta PUZaren erregistroen edukia ikusteko aukera ematen du.

- [Colby IASSim Web](#) : Klik eginez IASSim simulator biltegira konektatzen gara.

ÿ Deskargatu IASSim2.0.4 Simulator: zip fitxategia

ÿ Deskonprimitu IASSim2.0.4.zip fitxategia.

- Ireki simulagailua komandoa erabiliz:

```
1. ./IASSim2.0.4$ java -cp IASSim2.0.4.jar:jhall.jar:IASSimHelp2.0.jar iassim.Main -m IAS.cpu
```

ÿ Windows-en .bat lizapena duen batch fitxategian klik bikoitza egin dezakezu.

## 24.3. Simulazioa/Arazketa

- Simulazioaren helburuak bi dira:
  - a. Interpretatu instrukzio bakoitzaren exekuzioa memoria eta erregistroak nola aldatzen diren ikusiz
  - b. Programaren garapenean egon daitezkeen akatsak araztea.
- <https://www.linuxvoice.com/john-von-neumann/>
- Zenbaki osoen kodeketa hamaseitarra eta kode bihurtzea ezagutu behar da bitarra.
- Emuladorearekin berarekin datorren **Demo tutorial.ias** programari sum1toN.ias deituko diogu
  1. Deskargatutako zip fitxategiak deskonprimituta egon behar du: begiratu ateratako fitxategiak, horietako bat Hauek dira emuladorea irekitzeko argibideak.
  2. Ireki emuladorea:

ÿ Lerroan: **IASSim2.0.4.jar** **java -jar IASSim2.0.jar** **IASSim.Main -m IAS.cpu** Java -cp

ÿ Windows-en: Egin klik bikoitza \*.bat luzapena duen batch fitxategian
  3. Laguntza: **Laguntza** ÿ **IASSim Laguntza orokorra** ÿ **Mihiztadura hizkuntza** ÿ **Sintaxia eta argibide arruntak**: muntaia hizkuntzaren eskuliburua
  4. Ezabatu barneko zein kanpoko memoriaren edukia. **Exekutatu** ÿ **Garbitu guztia**
  5. Desgaitu arazketa modua: **Exekutatu** ÿ **Arazte modua EZ** hautatuta
  6. Kargatu sum1toN.ias programa mihiztadura hizkuntzan: **Fitxategia** ÿ **Ireki** ÿ **sum1toN.ias**

ÿ Mihiztadura hizkuntza: IASSim aplikazioaren egileek sortua.
  7. RAM Selectrons leihoa: helbideak eta edukia kode hamaseitar, hamartar, bitar... Zabalera memoria: 20 edo 40 bit.
  8. Hautatu iturburu-kodea duen leihoa muntaia hizkuntzan.
  9. Muntatu eta kargatu modulu exekutagarria memorian: **Execute** ÿ **Muntatu & Kargatu**
  10. Memoria-mapa aztertzea: instrukzio-atala eta datu-atala
  11. Aktibatu arazketa modua: **Execute** ÿ **Araztu modua**
  12. Instrukzio bakoitzaren exekuzioa urratsez urrats: **Urratsez urrats**
- Memoria Edukia

ÿ Lehenengo instrukzioa memoria-kokapenaren ezkerreko 20 bitetan gordetzen da eta eskuineko bigarren instrukzioa.

Address	Data	Comments
0	01 005	loop: S(x)->Ac+ n ;load n into AC
0	0F 002	Cc->S(x) pos ;if AC >= 0, jump to pos
1	00 000	halt ;otherwise done
1	00 000	.empty ;a 20-bit 0
2	05 007	pos: S(x)->Ah+ sum ;add n to the sum
2	11 007	At->S(x) sum ;put total back at sum
3	01 005	S(x)->Ac+ n ;load n into AC
3	06 006	S(x)->Ah- one ;decrement n
4	11 005	At->S(x) n ;store decremented n
4	0D 000	Cu->S(x) loop ;go back and do it again
5	00 000	n: .data 5 ;will loop 6 times total
5	00 005	
6	00 000	one: .data 1 ;constant for decrementing n
6	00 001	
7	00 000	sum: .data 0 ;where the running/total is kept
7	00 000	
8	00 000	
8	00 000	

108. Irudia IAS Makina Kodea

- Erregistroen edukia:

Registers		
Base: <b>Hexadecimal</b>		
name	width	value
Accumulator (AC)	40	00 0000 0000
Arithmetic Register (AR)	40	00 0000 0000
Control Counter (CC)	12	000
Control Register (CR)	20	0 0000
Function Table Register (FR)	8	00
Memory Address Register (MAR)	12	000
Selectron Register (SR)	40	00 0000 0000

109. Irudia IAS Erregistroak

- Ariketa:

- ÿ Instrukzio bakoitzak exekutatu aurretik, interpreta ezazu: interpretatu instrukzioa makina-lengoan.
- ÿ memoriako datuen ataleko eduki berria eskaintza
- ÿ CPU erregistroen eduki berria aurreikustea.
- ÿ exekutatuko den hurrengo instrukzioa aurreikusi
- ÿ Programaren organigrama ondorioztatu.

## 25. kapitulua. Goi-mailako eta behe-mailako programazio-lengoaiak

### 25.1. Maila handiko programazio-lengoaiak eta maila baxuko hizkuntzak

- 1960ko hamarkadan programazioa makina-lengoaia eta muntaia-lengoaian egiten bazen ere, hala izan zen oso ez-produktiboa, esfortzu eta denbora asko eskatuko lukeelako.
- Konponbidea ordenagailuaren funtzioak abstrakzioa egitea izan zen goi-mailako lengoaiak erabiliz:
  - ÿ <https://www.tiobe.com/tiobe-index/>
- Programatzaleentzako goi-mailako hizkuntza eraginkorrik erabiltzeak itzultzale bat behar du. ordenagailuko makina-lengoaia.

### 25.2. Sum1toN adibidea programazio hizkuntza ezberdinetan

- sum1toN algoritmoa ÿ  $\sum_{i=1}^N i = N(N + 1) / 2$
- Erref
  - ÿ <http://wiki.c2.com/?ArraySumInManyProgrammingLanguages>
  - ÿ [https://www.rosettacode.org/wiki/Sum\\_and\\_product\\_of\\_an\\_array#With\\_explicit\\_conversion](https://www.rosettacode.org/wiki/Sum_and_product_of_an_array#With_explicit_conversion)
- Sum1toN algoritmoa garatu: Lisp, Python, Java, C, Pascal, ...
- elisp

```
(setq array [1 2 3 4 5])
(aplikatu '+ (erantsi array nil))
(aplikatu (erants array nil))
```

- Python

```
>>> batura (barrutia (5,0,-1))
```

- Java

```
/* Iturburu-programa: sum1toN.java

eraiki: javac sum1toN.java -> sortu BYTECODE sum1toN.class run -> java -cp .
batuketa1toN ; *.class bytecode-a behar du eta klasaren nagusia exekutatuko du

*/
public class sum1toN { //
klase klasen kapsulatutako metodo nagusia, estatikoa, mainak atribituak alda ez ditzan, publikoa
eskuragarri izateko.
```

```

ÿ public static void main(String[] args) {
    System.out.println("Zenbaki osoen batura");
    ÿ int x=5, batura=0;

    bitartean (x >= 0 ) {
        Sistema.out.print( x );
        System.out.print(",");
        batuketa=batura+x;
        x--;
    }
    System.out.print("\n");
    System.out.print("sum="+sum);
    System.out.print("\n");
}
}

```

- C

```

/*
Programa: batuketa1toN.c
Deskribapena: egin 1,2,3,...N serieen batura
                Sum1toN.ias-en baliokide den C hizkuntzako programa da
von Neumann IAS makina
Hizkuntza: C99
Deskribapena: Lehenengo 5 zenbaki naturalen batura
Sarrera: Aldagai batean definitua
Irteera: Irteerarik ez
Konpilazioa: gcc -m32 -g -o sum1toN sum1toN.c -> -g: arazteko modulu bitarra
                -> -m: modulu bitarra

x86-32 biteko arkitektura
OS: GNU/linux 4.10 ubuntu 17.04 x86-64
Liburutegia: /usr/lib/x86_64-linux-gnu/libc.so
CPU: Intel(R) Core(TM) i5-6300U CPU @ 3.0GHz
Konpilatzailea: gcc 6.3 bertsioa
Muntatzailea: GNU muntatzailea 2.28 bertsioa
Lokatzailea/Kargatzailea: GNU ld (GNU Binutils Ubunturako) 2.28
Gaia: Data: Egilea: Konputagailuen Egitura
                2017/09/20
                Candido Aramburu
*/

```

```

#include <stdio.h> // printf() funtzio liburutegiaren goiburua

// programa sartzeko funtzioa
void main (huts)
{
    ÿ // Tokiko aldagaien adierazpena
    ÿ kararen batura=0;
    ÿ charn =0b101;
    ÿ // begizta

```

```

bitartean(n>0){
    batura+=n;
    n--;
}
printf("\n batura = %d \n",batura);
}

```

- ATT mutua-lengoia x86-32 arkitekturako

```

### Programa: sum.s
### Deskribapena: 1,2,3,...N serieen batura egiten du
### gcc -m32 -g -nostartfiles -o sum1toN sum1toN.s
### Asanblada gisa --32 --gstabs source.s -o object.o
### estekatzalea -> ld -melf_i386 -l/lib/i386-linux-gnu/ld-linux.so.2 -o exekutagarria
objektua.o -lc

## Aldagaien adierazpena
.atala .datuak

n: .int 5

.global _hasiera

## Kodearen hasiera
.atala .testua

_hasi:
    mov $0,%ecx # ECX-k batura aldaagaia ezartzen du
    mov n,%edx
begizta:
    gehitu %edx,%ecx
    azpian $1, %edx
    jnz begizta

    mov %ecx, %ebx # irteerako argumentua OSra EBX bidez
akordioa

## irten
    mov $1, %eax # sistema eragilearen deien kodea: irten azpierrutina
    int $0x80 # OS deia

.amaiera

```

- AT&T mutua-lengoia x86-64 arkitekturako

```

##### Programa: sum1toN.s
### Deskribapena: 1,2,3,...N serieen batura egiten du. Sarrera definituta dago
programan bertan eta irteera OSra pasatzen da
### Hizkuntza: GNU mutua-lengoia AMD64 arkitekturarako

```

```

ÿ ### gcc -no-pie -g -nostartfiles -o sum1toN sum1toN.s
ÿ ### Muntaketa gisa --gstabs sum1toN.s -o sum1toN.o
    ### estekatzalea -> ld          -o sum1toN sum1toN.o
ÿ ## Aldagaien adierazpena
ÿ ## DATU ATALA
    .atala .datuak

n:      .quad 5

.global _hasiera

## Kodearen hasiera_
## JARRAIBIDEEN ATALA

.atala .testua

._hasi:
    movq $0,%rdi # RDI-k batura aldagai ezartzen du
    movq n,%rdx
begizta:
    gehitu %rdx,%rdi
    azpiko $1,%rdx
    jnz begizta

## RDI bidez OSra irteera argumentua AMD64 ABI konbentzioaren arabera
ÿ ## irteera
    mov $60,%rax # sistema eragilearen dei -kdea : irten azpierrutina
    syscall # dei sistema eragileari azpierrutina exekutatzeko
RAX balioa

.amaiera

```

- BESOA

```

/* Programa: sum1toN.s
Deskribapena: 1,2,3,...N serieen batura egiten du
IAS makinaren sum1toN.ias -en baliokidea den ARM hizkuntza-programa da
von Neumann

gcc -g -nostartfiles -o sum1toN sum1toN.s
Muntaketa gisa --gstabs sum1toN.s -o sum1toN.o
lokailua -> ld          -o sum1toN sum1toN.o
*/
@ Aldagaien adierazpena
.atala .datuak
n:      .int 5

.global _hasiera

@ Kodearen hasiera
.atala .testua

```

\_hasi:

```
    mov r0,#0 ldr          @R0-k batura aldagaia implementatzen du  
    r2,=n ldr r1,[r2]      @R1-ek n aldagaia implementatzen du zeharka
```

/\* Helbide zuzena:

```
    mov r1,n-k errore bat ematen du, mov-ek ez duelako zuzeneko memoria-helbideratzea onartzen.  
    mov-ek berehalako helbideratza onartzen du 32 biteko literalak ez badu
```

ezkerrean eta eskuinean zeroen errepikapena du

8 biteko literal bihurtzeko eta ondoren

desplazamenduak

```
    ldr r1,n Errorea: barne_lokalizazioa (OFFSET_IMM mota) ez dago osatuta
```

Errore bat 32ko literalak (n helbidea) kodetzen sariatzean

bitsak.

\*/

begizta:

```
    gehitu r0,r1  
    azpiak r1,#1  
    bne begizta
```

@r0 EBX bidez OSra iristeko argumentua da konbentzioaren arabera

```
/* irten syscall */  
    mov r7, #1  
    swi #0
```

.amaiera

## 26. kapitula. Mihiztadura-programazio-lengoaiak

### 26.1. Erreferentzia eskuliburuak

#### 26.1.1. Intel hizkuntza

- Eskuliburu ofizialak

- ÿ Intel: 2. liburukia

- ÿ AMD: Eskuliburuen atala: 3. liburukia

- ÿ AMD64 Arkitektura Programatzalearen eskuliburua 3. liburukia: helburu orokorra eta sistemaren argibideak

- Eskuliburu ez ofizialak:

- ÿ Intel eskuliburu azkarra: gomendagarria

- ÿ Intel deskribatzalea i386

- ÿ ISA Erreperitorioa eta Instrukzio Formatua

- ÿ kluge

- ÿ Baldintzazko jauziak

#### Netwide ASM (NASM) Intel hizkuntzarako

- Sum1toN.asm Intel muntatzaile-lengoaian eta "NetWide Asm" (nasm) mihiztagailuko programa baten adibidea.
- NASM tutoretza tutoretza puntu osoa
- Bristoleko Unibertsitateko oharrak
- Paul Carter eta Paul Carter doktorearen oharrak

#### 26.1.2. AT&T hizkuntza

- Oracle Solaris ASM

- ÿ Dokumentu honetan AT&T-k "Oracle Solaris" izena du sintaxia.

- ÿ AT&T Solaris eskuliburua amd64-i386: montaia-lengoaia eta itzultzalea.

#### 26.1.3. i386 arkitekturaren ezaugarriak

- Muntatzailea: x86 arkitekturaren menpeko ezaugarriak

#### 26.1.4. Muntatzailea: Zuzentaraauak

- Muntatzaileen Itzultzzaileen Zuzentaraauak

#### 26.1.5. Eztabaida zergatik ASM AT&T

- <http://es.tldp.org/Presentaciones/200002hispalinux/conf-28/28.ps.gz>

#### 26.1.6. TRANSFERENTZIA

Izena eta	Iruzkina	Kodea	Operazioa	ODITSZAPC
MOV Mugitu (kopiatu) MOV Source,Dest Dest:=Iturria				
XCHG Trukea		XCHG Op1,Op2 Op1:=Op2, Op2:=Op1		
STC Ezarri eramatea (Eraman = 1)		STC	CF:=1	1
CLC Garbitu Eramatea = 0)		CLC	CF:=0	0
CMC Osagarria Eraman		CMC	CF:=Ø	±
STD Ezarri STD helbidea			DF:=1(goitik kateak interpretatzen ditu behera)	1
CLD Garbitu CLD helbidea			DF:=0 (beheko kateak interpretatzen ditu gora)	0
STI	Bandera 1ean atsedenaldia	STI	BADA:=1	1
CLI	Bandera Etenaldia 0-n	CLI	BADA:=0	0
PUSH Pila pila PUSH Iturria			DEC SP, [SP]:=Iturria	
PUSHF Pilatu banderak		PUSHF	O, D, I, T, S, Z, A, P, C 286+: Baita NT, IOPL	
PUSHA Pilatu erregistroak orokorra		PUSHA	AX, CX, DX, BX, SP, BP, SI, DI	
POP Despilatu pilatik POP Dest			Helmuga:=[SP], INC SP	
POPF-k despilatzen du banderak		POPF	O,D,I,T,S,Z,A,P,C 286+: Baita NT, IOPL	± ± ± ± ± ± ± ±
POPA Erregistroak despilatzen ditu. orokorra.		STERN	DI, SI, BP, SP, BX, DX, CX, AX	
CBW Bihurtu byte-ra hitzak		C.B.W.	AX:=AL (sinatua)	
CWD Bihurtu Word-era Bikoitza		CWD	DX:AX:=AX (sinatua)	
CWDE konb. hitza a Luzapen bikoitza		CWDE 386	EAX:=AX (ikurra duena)	
IN	Sarrera	Dest, Portuan	AL/AX/EAX := byte/hitza/bikoitza portu berezia	
OUT Irteera		OUT ataka, iturburua byte/hitza/portuaren bikoitza := AL/AX/EAX		

- Banderak: ± =Instrukzio honek eraginda, ? =Instrukzio honen ondoren definitu gabe

#### 26.1.7. ARITMETIKA

Izena eta	Iruzkina	Kodea	Operazioa	ODITSZAPC
GEHITU Batura	GEHITU Iturburua,Dest	Dest:=Dest+ Iturria		± ± ± ± ±
ADC Gehitu carry ADC Iturria, Dest	Dest:=Dest+ Iturria +CF			± ± ± ± ±
SUB Kenketa	SUB Source,Dest	Dest:=Dest- Iturria		± ± ± ± ±
SBB kenketa eramangarriarekin	SBB Iturria,Dest	Dest:=Dest-(Iturria +CF)		± ± ± ± ±
DIV	Zatiketa (gabe seinaldea)	DIV Op	Op=byte: AL:=AX / Op AH:=Atseden	? ? ? ? ?
DIV	Zatiketa (gabe seinaldea)	DIV Op	Op=hitza: AX:=DX:AX / Op DX:=Atseden ? ? ? ? ?	
DIV	386 Dibisioa (gabe seinaldea)	DIV Op	Op=doblew.: EAX:=EDX:EAX / Op EDX:=Atseden	? ? ? ? ?
IDIV	Zenbaki osoen zatiketa zeinuarekin	IDIV Op	Op=byte: AL:=AX / Op AH:=Atseden	? ? ? ? ?
IDIV	Zenbaki osoen zatiketa zeinuarekin	IDIV Op	Op=hitza: AX:=DX:AX / Op DX:=Atseden ? ? ? ? ?	
IDIV	386 Dibisioa zeinudun osokoa	IDIV Op	Op=doblew.: EAX:=EDX:EAX / Op EDX:=Atseden	? ? ? ? ?
MUL biderketa (gabe seinaldea)	MUL Op		Op=byte: AX:=AL*Op AH=0 bada #	± ? ? ? ? ±
MUL biderketa (gabe seinaldea)	MUL Op		Op=hitza: DX:AX:=AX*Op baldin DX=0 # ± ? ? ? ±	
MUL 386 Biderketa (seinalerik ez)	MUL Op		Op=bikoitza: EDX:EAX:=EAX*Op bai EDX=0 #	± ? ? ? ? ±
IMUL i Biderketa osorik zeinuarekin	IMUL Op		Op=byte: AX:=AL*Op AL bada nahikoa #	± ? ? ? ? ±
IMUL Multip. osorik zeinuarekin	IMUL Op		Op=hitza: DX:AX:=AX*Op AX bada nahikoa #	± ? ? ? ? ±
IMUL 386 Biderketa zeinudun osokoa	IMUL Op		Op=bikoitza: EDX:EAX:=EAX*Op bai EAX sufia da. #	± ? ? ? ? ±
INC	Areagotu	INC Op	Op:=Op+1 (The Carry ez da kaltetua!)	± ± ± ± ±
DEK Dekretu		ABENDUA Op	Op:=Op-1 (Eramatea ez da emaitzarik ematen kaltetua!)	± ± ± ± ±
CMP Konparatu	CMP Iturria, helmuga		Helmuga-Iturria	± ± ± ± ±
GATZA	Desplazamendua aritmetika. ezkerretara	GATZA	Op, Kantitatea	i ± ± ? ± ±
SAR Desplazamendua aritmetika. eskuinera		BERAK	Op, Kantitatea	i ± ± ? ± ±
RCL	Biratu ezkerrera eraman/eraman	RCL Op,Kantitatea		i ±
RCR Biratu eskuinera eraman/eraman		RCR Op, Kantitatea		i ±

Izena eta	Iruzkina	Kodea	Operazioa	ODITSZAPC
ROLA	Biratu hona ezker	ROLA Op,Kantitatea		i ±

- i: informazio gehiago lortzeko, ikusi argibideen zehaztapenak,
- #: gero CF:=0, OF:=0 bestela CF:=1, OF:=1

## 26.1.8. LOGIKOA

Izena eta	Iruzkina	Kodea	Operazioa	ODITSZAPC
NEG	Ezeztapena (2 osagarria)	NEG Op	Op:=0-Op Op=0 bada CF:=0 bestela CF:=1	± ± ± ± ±
EZ	alderantzikatu bit bakoitza EZ	Op	Op:=Ø~Op (bit bakoitza alderantzikatzen du)	
AND	AND (And) logikoa AND Source,Dest Dest:=Dest ^ Source			0 ± ± ? ±0
OR	OR OR (Edo) logikoa	OR Source,Dest Dest:=Dest v Iturria		0 ± ± ? ±0
XOR	O (Or) XOR iturri esklusiboa,Dest Dest:=Dest (xor)	Iturria		0 ± ± ? ±0
SHL	Desplazamendua logikoa geratzen da	SHL Op, Kantitatea		i ± ± ? ± ±
SHR	Desplazamendua logikoa kantitatea.	SHR Op, Eskuineko		i ± ± ? ± ±

## 26.1.9. DENBERAK

Izena eta	Iruzkina	Kodea	Operazioa	ODITSZAPC
EZ	Ez egin ezer	EZ	Ez du inolako eragiketarik egiten	
IRAKURRI	Kargatu helbidea Eraginkorra	IRAKURRI	Iturburua, Dest Dest := iturburu helbidea	
INT	Etenaldia	INT Zenb	Uneko prozesua eten eta bertara jauzi egiten du bektorea Zenb	0 0

## 26.1.10. JAUZIAK (orokorra)

- [wiki x86 mutaria](#)

Izena Iruzkina	Kodea	Operazioa
DEITU	Azpirrutina deia	DEIALDIA Proc
JMP	Hop	JMP Dest
J.E.	Saltatu berdina bada	JE Dest (=JZ)
J.Z.	Saltatu zero bada	JZ Dest (=JE)
JCXZ	Saltatu CX zero bada	JCXZ Dest
J.P.	Saltatu parekotasuna badago	JP Dest (= JPE)
JPE	Saltatu parekotasuna ere badago	JPE Dest (=JP)
J.P.O.	Saltatu parekotasun bakoitia badago	JPO Dest (= JNP)
JNE	Saltatu berdina ez bada	JNE Dest (=JNZ)

JNZ	Saltatu zero ez bada	JNZ Dest	(= JNE)
JECXZ	Jauzi ECX zero bada	JECXZ Dest	386
JNP	Saltatu parekotasunik ez bada	JNP Dest	(=JPO)
RET	Azpirrutinaren itzulera	RET	

### 26.1.11. JUSTOAK Zeinurik gabe (kardinala) JUSTOAK zeinuarekin (Osokoa)

Izena Iruzkina	Kodea	Operazioa
JA Jauzi gorago bada Salto	JA Dest	(=JNBE)
JAE handiagoa edo berdina bada JAE Dest Jauzi baxuagoa bada		(= JNB = JNC)
J.B. Jauzi txikiagoa bada edo	JB Dest	(= JNAE = JC)
JBE berdina JBE Dest Jauzi handiagoa ez bada JNA Dest Jauzi		(= JNA)
JNA super ez bada. edo berdin JNAE Dest Skip txikiagoa ez bada		(= JBE)
JNAE JNB Dest Skip txikiagoa ez bada. edo berdina JNBE Dest		(= JB = JC)
J.N.B. Saltatu garraioa badago JC Dest Saltatu garraiorik ez badago		(= JAE = JNC)
JNBE Saltatu gainezkariak ez badago Saltatu seinalea badago		(=HA)
J.C. (=negatiboa) JS Dest	JO Dest	Salto gainezka badago
JNC	JNC Dest	
JNO	JNO Dest	
J.S.		
J.G. Jauzi handiagoa bada JG Dest Jauzi handiagoa bada edo		(=JNLE)
JGE berdina bada Salto txikiagoa bada	JGE Dest	(=JNL)
J.L.	JL Dest	(=JNGE)
JLE Saltatu baino txikiagoa edo berdina bada	JLE Dest	(=JNG)
JNG Saltatu zaharragoa ez bada	JNG Dest	(=JLE)
JNGE Saltatu JNGE Dest baino handiagoa edo berdina ez bada		(=JL)
J.N.L. Saltatu beherago ez bada	JNL Dest	(= JGE)
JNLE Saltatu JNLE Dest baino txikiagoa edo berdina ez bada		(=JG)

### 26.1.12. BANDERAK (ODITSZAPC)

O: gainezka funtzionamenduaren emaitza. seinalerik gabe handiegia edo txikiegia da.

D: Helbidea

I: Etenaldia Etenaldiak gerta daitezkeen edo ez adierazten du.

T: Trap Urratsez urrats arazketarako

S: Zeinua emaitzaren seinale. Zenbaki osoentzat bakarrik arrazoizkoa. 1=ez. 0=pos.

Z: Zero Eragiketaren emaitza zero da. 1=Zero

A: Carru Aux. Carry-ren antzekoa, baina nibble baxurako soilik mugatuta

P: Parekidetasuna 1 = emaitzak batean bit bikoitiak ditu

C: Eragiketaren emaitza eraman. sinatu gabekoa oso handia da edo zero baino txikiagoa da

### 26.1.13. Atzizkiak

- Opcode atzizki mnemoteknikoak:

ÿ q : laukoak: 8 byteko eragigaia: lauko hitza

ÿ l : luzea: 4 byteko eragigaia: hitz bikoitza

ÿ w : hitza: 2 byteko eragigaia: hitza

ÿ b : byte: 1 byteko eragigai

- Eragiketa mnemoteknikoak atzizkirik ez badu, eragigaiaren tamaina lehenetsia luzea da

## 26.2. Intel x86-32 / i386

### 26.2.1. batuketa1toN.s

- ATT mutua-lengoaia x86-32 arkitekturarako

```
### Programa: sum1toN.s
### Deskribapena: 1,2,3,...N serieen batura egiten du
### gcc -m32 -g -nostartfiles -o sum1toN sum1toN.s
### Asanblada gisa --32 --gstabs source.s -o object.o
### estekatzailea -> ld -melf_i386 -l/lib/i386-linux-gnu/ld-linux.so.2 -o exekutagarria
objektua.o -lc

## Aldagaien adierazpena
.atala .datuak

n: .int 5

.global _hasiera

## Kodearen hasiera
.atala .testua

_hasi:
    mov $0,%ecx # ECX-k batura aldagaia ezartzen du
    mov n,%edx
begizta:
    gehitu %edx,%ecx
    azpian $1, %edx
    jnz begizta

    mov %ecx, %ebx # irteerako argumentua OSra EBX bidez
akordioa

## irten
    mov $1, %eax # sistema eragilearen deien kodea: irten azpierrutina
    int $0x80 # OS deia

.amaiera
```

## 26.2.2. kaixo\_mundua.s

- Konpilatu programa hello\_world.s mihiztatzale lengoain eta bota objektu bitar modulu.

ÿ Iturburu-modulu: hello\_world.s.

###

```

#### kaixo_mundua.s
####
### Softwarea garatzeko hasierako programa simplea
x86-32 AT&T muntatzalea.
####
### Konpilazioa:
### muntatu: as --32 hello_world.s -o hello_world.o
### esteka erabiliz: ld -melf-i386 hello_world.o -o hello_world
### gcc kontrolatzalea: gcc -m32 -nostartfiles hello_world.s -o
Kaixo Mundua
####
### eguneratua: 2022ko iraila -- Linux-en i386 inguruneko
###
###

.att_sintaxia

## Kanpoko sinboloen aitorpena
.global _hasiera          # sarrera-puntu ikusgai

## Memoria erreserba datu aldakorretarako
.atala           .datuak

mezua:      .ascii "Kaixo mundua\n"
luzera:     .2byte . - mezua      # tamaina mezu-katearen bytetan

## Batzar Hizkuntza Irakaskuntza Kodearen Atala
.atala .testua

._hasi:

mov        $4, %eax    # SYS_WRITE
mov        $1, %ebx # gailuaren ID-zenbakia
mov        $message, %ecx # mezuaren helbidea
mov        $luzera, %edx # mezuaren luzera
int        $0x80 # sartu nukleoan

## amaitu programa hau
mov        $1, %eax    # SYS_EXIT
mov        $0, %ebx # itzuleraren balioa
int        $0x80 # sartu nukleoan

.amaiera      # gehiago ez muntatzeko

```

### 26.2.3. hello\_world: Makina Kode Binarioa

- Datuen Atala

```
08049ff4 <mezua>: 48 6f 6c 61 20 4d 75 63 64 6f 0a 08049fff <luzera>:  
0b 00
```

Kaixo SP mundua /n

ÿ Maila handiko hizkuntza batean aldagaien deklarazioa eta hasieratzea izango litzateke.

ÿ Etiketa: memoria erreferentzia

ÿ Karaktere bakoitzak byte bat hartzen du (ASCII kodeketa). Ez interpretatu katea osorik (ez da gutxi endian) zenbaki osoak eta zenbaki errealkak ez bezala.

ÿ Luzera etiketak aipatzen dituen datuak little endian formatuan daude ÿ 00 0b

- Argibideen Atala

#### ## Batzar Hizkuntza Irakaskuntza Kodearen Atala

```
08048190 <_hasi>:  
8048190: b8 04 00 00 00 8048195:      mov    $ 0x4, %ax  
bb 01 00 00 00                           mov    $0x1,%ebx  
804819a: b9 f4 9f 04 08 804819f:      mov    $0x8049ff4,%ecx  
8b 15 ff 9f 04 08 80481a5: cd 80       mov    0x8049fff,%edx  
                                         int    $0x80
```

#### ## amaitu programa hau

```
80481a7:b8 01 00 00 00      mov    $ 0x1, %ax  
80481ac: bb 00 00 00 00      mov    $0x0,%ebx  
80481b1: cd 80              int    $0x80
```

Jarri programa bitar bat

- Komandoa erabiliz: objdump -d hello\_world, non hello\_world modulu bitar exekutagarria den.

Programa bitarra Memoria Nagusian gordetzea

- Sorburu-moduluaren itzulpen-prozesua mutua hizkuntzan amaitutakoan, a Disko gogorrean fitxategi moduan gordetzen den lengoia bitarreko objektu-modulua.
- Objektu exekutagarriaren modulu hizkuntza bitarrean duen fitxategia kargatu behar da memoria nagusia. Zeregin hau sistema eragilearen kargatzaleak egiten du.
- Memoria-helbide bakoitzak byte 1era seinalatzen du.
- Helbide baxuenak objektu osora seinalatzen du: instrukzioa edo datuak.
- Adibidea:

ÿ amd64 arkitekturako makinen instrukzioa.

ÿ 4001a4: 48 83 ec 10 ÿ subq \$16,%rsp

ÿ 4001a4: 48 83 eq 10

ÿ 0x4001A4 posizioan 48 bytea dago

ÿ 0x4001A4+1 posizioan 83 byte dago

ÿ 0x4001A4+2 posizioan EC byta dago

ÿ 0x4001A4+3 posizioan 10 byte dago

ÿ 4 Byte-ko instrukzioa 0x4001A4 memoria nagusian gordetzen da

Makina kodean instrukzio bat interpretatzea: Intel x86-64 ISA Instruction Format

- Adibidea:

ÿ amd64 arkitekturako makinen instrukzioa.

ÿ 4001a4: 48 83 ec 10 ÿ subq \$16,%rsp

ÿ Programatzailaren interpretazioa:

ÿ AT&T x86 arkitekturen mihiztadura-lengoaia.

ÿ Irakaskuntzaren deskribapena RTL hizkuntzan: RSP ÿ RSP - 16

ÿ \$16,%rsp subq instrukzioa 0x4001A4 memoria nagusian gordetzen da

ÿ subq-k kenketa eragiketa adierazten du 64 biteko osoko datuekin (q atzizkia). Eragigaiaren kenketa helmuga iturri-eragigaia.

ÿ Iturburuko eragigaia 16 balio hamartar bat du, 0x10 hamaseimalean eta eragigai honen helbideratza berehalakoa da, hau da, bere balioa 16 da eta instrukzioan bertan kokatzen da.

ÿ Helmuga eragigaia RSP izeneko CPUaren barneko erregistroan gordetzen da

ÿ Hurrengo instrukzioaren erreferentzia ez da instrukzioa egiten, baizik eta PUZak egiten duena eragiketa PCþPC+aginduen tamaina bytetan.

ÿ Nola interpretatu makina bat instrukzio bat hizkuntza bitarrean? Beharrezko da x86 makinaren ISA Arkitektura Erreferentzia Eskuliburu kontsultatzea eta helbideratze moduen ezagutza izatea.

ÿ Intel x86 edo x86-64rako eskuliburu ofiziala: kontuz ibili Intel sintaxiarekin.

ÿ kontsultatu 2B liburukia (4. kapitulua, 394. orrialdea) SUB instrukziorako. Kontuan hartu behar duzu eragigaien tamaina eta helbideratze moduak.

ÿ SUBQ eragiketaren q atzizkiak 64 biteko eragigai bat adierazten du. \$16 iturburuko eragigaia berehalako helbideratzearekin aipatzen da eta 8 bitekin kodetu daiteke eta helmugako %RSP eragigaia 64 biteko erregistroa da. Beraz, Intel deskribapena eskuliburan SUB r64, imm8 REX.W + 83 /5 ib eragiketa kodeari dagokiona izango da.

ÿ Intelek funtzionamendu-kodearen deskribapena ez da erraza eta beharrezko da 3. liburukiko Argibideen Interpretazioa kontsultatzea 3.1 JARRAIBIDEEN ERREFERENTZIA ORRIAK INTERPRETATZEA eta 2A liburukiko Argibideen Formatua (2. kapitulua Argibideen Formatua)

ÿ 2.1 Irudia Intel 64 eta IA-32 Arkitekturek instrukzio-formatua

ÿ Instrukzio formatuak eremu hauek ditu: REXprefix-CodOp-ModRB gure kasuan 48-83-EC balio dute

ÿ REXaurrizkiaren eremuaren interpretazioa: REX.W: Eskulibura ÿ REX aurrizkia 64 biteko eragigaietarako erabiltzen da, berehalako edo/eta GlobalPurposeRegister(rax,rbx, etab.) erregistratzen dituena, 2.2.1.2 REX aurrizkiaren eremuie buruz gehiago

ÿ Lehenengo bytea 48 ÿ 01001000 da, non 3. posizioko bit-a aktibatuta dagoen, beraz, "2-4. Taula. REX Aurrizkiaren Eremuak [BITS: 0100WRXB]" taularen arabera, eragigaia 64 bitekoa dela esan nahi du.

ÿ /5 : instrukzioaren ModR/M byteak r/m (erregistroa edo memoria) eragigaia soilik erabiltzen du. In kasu honetan erregistroa.Ikus behean R/M azpieremua.

ÿ ib : 1 byte (ib) berehalako eragiketa.

ÿ Lehen mailako Opcode eremua: bigarren bytea 83 ÿ SUB kenketa eragiketa berdina da

ÿ ModRB eremua: hirugarren bytea EC ÿ 1110-1100 da eta RSP erregistroari egiten dio erreferentzia.

- ÿ 2.1.3 ModR/M eta SIB byteak: memoriako eragiketa bati erreferentzia egiten dioten jarraibide asko.  
(memoria nagusia edo barneko PUZaren erregistroa) helbideratze-formaren zehazlearen byte bat ikusi du (deitutakoa ModR/M...). Eremu hau azpieremuetan banatzen da: **Mod-Reg/Opcode-R/M**
- ÿ Mod azpieremua: **11** : mod eremua r/m eremuarekin konbinatzen da 32 balio posible eratzeko:  
zortzi erregistro (rax, rbx, rcx, rdx, rsi, rdi, rsp, rbp) eta 24 helbideratze modu
- ÿ Reg/Opcode azpieremua: kasu honetan ez da Bigarren mailako Opcode, baizik eta Reg: rrr= **101**
- ÿ R/M azpieremua: kasu honetan R: bbb= **100** r/m eremuak erregistro bat zehaztu dezake  
operandoa edo mod eremuarekin konbina daiteke helbideratze modu bat kodetzeko. **Honetan**  
**kasua 4 kodearekin funtzionatzen duen erregistroa da.** 3.1 taulan **hitz lauko** kodea  
**erregistratu** Reg Eremuaren balioa **4 RSP** erregistroa da
- ÿ 2-5 irudia. Erregistratu-Erregistratu Helbideratzea (Memoria Operandorik gabe); REX.X Ez da erabiltzen ÿ  
Mod=11 ÿ Rrrr =0101 ÿ Bbbb=0100
- ÿ Laugarren byteak hamaseitarrez **10** balio du, hau da, berehalako balioari dagokion 16.  
hamartar eta 64 bitera zabaldu behar da.

## 26.3. Intel x86-64 / AMD 64

### 26.3.1. batuketa1toN.s

```
#### Programa: sum1toN.s
### Deskribapena: 1,2,3,...N serieen batura egiten du. Sarrera urtean definitzen da
programa bera eta irteera OSra pasatzen da
### Hizkuntza: GNU mutaia-lengoaia AMD64 arkitekturarako
### gcc -no-pie -g -nostartfiles -o sum1toN sum1toN.s
### Muntaketa gisa --gstabs sum1toN.s -o sum1toN.o
        ### estekatzalea -> ld          -o sum1toN sum1toN.o
## Aldagaien adierazpena
## DATU ATALA
        .atala .datuak

n:      .quad 5

        .global _hasiera

        ## Kodearen hasiera
        ## JARRAIBIDEEN ATALA

        .atala .testua

_hasi:
        movq $0,%rdi # RDI-k batura aldaagaia ezartzen du
        movq n,%rdx
begizta:
        gehitu %rdx,%rdi
        azpiko $1,%rdx
        jnz begizta

        ## RDI bidez OSra irteera argumentua AMD64 ABI konbentzioaren arabera
## irten
        mov $60, %rax # sistema eragilearen dei-kodea: irten azpirrutina
```

```
    syscall # dei sistema eragileari azpierrutina exekutatzeko
RAX balioa
```

.amaiera

### 26.3.2. Kaixo Mundua

- Iturburu-modulua hola\_mundo.s mutuaia hizkuntzan.

```
### -----
### kaixo_x86-64_att.s
###
### Softwarea garatzeko hasierako programa simplea
x86-64 AT&T muntatzailea .
###
Fitxategi osagarriak : macros_x86-64_gas.h
###
###
Konpilazioa:
muntatu erabiliz: as hello_intel_gas.s -o hello_intel_gas.o
esteka erabiliz: ld hello_intel_gas.o -o hello_intel_gas
gcc kontrolatzalea : gcc -nostartfiles hello_intel_gas.s -o
kaixo_intel_gas
###
eguneratua : 2015eko OTSAILA -- Linuxs x86_64 ingurunerako
###
### -----
ÿ .att_sintaxia

ÿ ## Sartu fitxategia makroekin
ÿ .sartu "macros_x86-64_gas.h"

ÿ ## Kanpoko sinboloen aitorpena
ÿ .global _hasiera          # sarrera-puntu ikusgai

ÿ ## Memoria erreserba datu aldakorretarako
ÿ .atala           .datuak

msg0:   .ascii "Kaixo mundua\n"
len0:   .quad . - msg0      #tamaina msg0 katearen bytetan

ÿ ## Batzar Hizkuntza Irakaskuntza Kodearen Atala
ÿ .atala .testua

_hasi:

ÿ ## Programaren gonbita : mezua inprimatu
```

```

ÿ ## Deitu nukleora pantailara sartzeko eta inprimatzeko.
ÿ mov      $SYS_WRITE, %rax # zerbitzuaren ID-zenbakia
ÿ mov      $STDOUT_ID, %rdi # gailuaren ID-zenbakia
ÿ mov      $msg0, %rsi # mezu helbidea
ÿ mov      len0, %rdx # mezuaren luzera
ÿ syscall

```

```

ÿ ## amaitu programa hau
ÿ mov      $SYS_EXIT, %eax # zerbitzuaren ID-zenbakia
ÿ mov      $0, %rdi # konfigurazio irteera-kodea
ÿ syscall # sartu nukleoa

ÿ .amaiera          # gehiago ez muntatzeko

```

#### ## Makroak macros\_x86-64\_gas.h fitxategian

## Sistema -deiak _	
STDIN_ID, 0	# sarrera-gailu (teklatua)
STDOUT_ID, 1	# irteera-gailu (pantaila)
.equ .equ .SYS_READ, 0	# "irakurtzeko" ID zenbakia
.equ     SYS_WRITE, 1	# "idatzi" ren ID zenbakia
SYS_OPEN, 2	# "ireki" -ren ID zenbakia
SYS_CLOSE, 3	# "Itxi" -ren ID zenbakia
.equ .equ .SYS_EXIT, 60	# Irteerarako ID zenbakia

### 26.3.3. Denetarikoa

#### Datu mota

- Datu mota:

ÿ Memoriaren helbidea edo memoria-erreferentzia: izenaren etiketaren luzera

ÿ mov \$length,%edx ÿ mov 0x8049fff,%edx ÿ goi-mailako hizkuntzan baten hasierako erakuslea

ÿ sinatutako zenbaki oso: 2-ren osagarri formatua.

ÿ mugitu \$0x4,%eax

ÿ 0x4 operandoa instrukzioan bertan dago, eragiketa eremuan. 0x4 datuak da "Little Endian"-en gordeta ÿ Operando eremua: hitz bikoitza: 32 bit 0x00000004 ÿ Ingorkako memoria: 8048191 helbidea: 04 00 00 00

ÿ karakterea: ASCII kodeketa

ÿ 08049ff4 <mezua>: 48 6f 6c 61 20 ÿ Kaixo SP

#### Zenbaki errealkak

- kluge

ÿ Zenbaki errealekin egindako eragiketen adibide interesgarriak

#### Irakaskuntza Zikloa

- PUZaren esku-hartzea 4001a4 instrukzioan: 48 83 ec 10 ÿ subq \$16,%rsp
  - ÿ PUZak instrukzio-zikloan zehar (harrapatze-fasea-deskodetze-fasea-exekuzio fasea) bat egiten du. ataza-sekuentzia.
  - ÿ PUZak instrukzio-zikloan zehar egin beharreko atazen sekuentzia hizkuntzan deskribatzen dugu RTL.
  - ÿ MBR ÿM[0x4001a4]
  - ÿ JOAN ÿMBR
  - ÿ AC ÿ RSP
  - ÿ AC ÿ AC-16 ; (ALU kenketa)
  - ÿ RSP ÿ AC
  - ÿ PC ÿ PC+1
  - ÿ MAR ÿ PC

#### 26.3.4. sum1toN.s: intel language

- INTEL asanblea hizkuntza eta nasm asanblada

```
;; Programa: sum1toN.asm
;; Deskribapena: 1,2,3,...N serieen batura egiten du
;; INTEL hizkuntza
;; NASM Muntatzailea

;; nasm -hf -> f aukeraren laguntza
;; Asanblada nasm -g -f elf sum1toN.asm -o sum1toN.o
;; estekatzalea -> ld -m elf_i386           -o sum1toN sum1toN.o

BITS 32 ; CPU MODUA
; Aldagaien adierazpena
atala .datuak

n: dd 5          ; 4 byte

global _hasiera

; Kodearen hasiera
atala .testua

_hasi:
    mov ecx,0 ; ECX-k batura aldagaiia implementatzen du
    mov edx,[n] ; EDX implements n aldagaiaren alias bat da
begizta:
    gehitu ecx,edx
    azpi edx,1
    jnz begizta

    mov ebx, ecx ; EBX bidez OSra irteera argumentua arabera
```

akordioa

```
;; irten
mugitu eax,1; sistema eragilearen dei-kodea: irten azpierrutina
int 0x80 ; sistema eragilearen deia
```

## 26.4. ARM arkitektura

### 26.4.1. Kaixo Mundua

```
/*
```

AT&T ARM prozesadorerako muntaia-lengoaia programa

Iturburu-programa: hello\_world.s

Muntatzailea: arm-linux-gnueabi-as -o hello\_world.o hello\_world.s

Lokatzailea: arm-linux-gnueabi-ld -o hello\_world hello\_world.o

```
*/
```

.datuak

mezua:

.ascii "Kaixo, ARM Mundua!\n"

len = . - mezua

.testua

```
.globl _hasi
_hasi:
/* idatzi syscall */
mugitu %r0, $1
ldr %r1, =msg
ldr %r2, =len
mugimendua %r7, $4
swi $ 0
```

```
/* irten syscall */
mugitu %r0, $0
mugimendua %r7, $1
swi $ 0
```

### 26.4.2. DA

- ARM: RISC Makina aurreratua

ÿ Garatzaleen gidak

## 26.5. Motorola 68000

### 26.5.1. Kaixo Mundua

;CISC Sharp X68000 (Human68K): Motorola 68000 **ilarra** (katea)

**dc.w \$FF09**

; push string helbidea pilara ; deitu DOS

**addq.l #4,a7**

"inprimatu" salbuespen bat eraginez; leheneratu pila erakuslea

**dc.w \$FF00**

; deitu DOS "irten"

katea:

**dc.b "Kaixo, mundua!",13,10,0**

### 26.5.2. DA

- Erreferentziak

ÿ [Oinarrizko Argibide multzoa](#)

ÿ [Wikibook](#)

ÿ [Erreferentzia eskuliburua](#)

ÿ [Motorola 68K edo M68000](#)

ÿ m68k 1991 arte

ÿ ppc (powerpc) 1991tik Apple eta IBMrekin ÿ iMac (1996-2006)

- arkitektura orokorra

2 bertsio: 16 biteko edo 32 biteko prozesadorea Gutxi gorabehera.

90 makinaren argibide 12 helbideratze modu

9 instrukzio formatu ezberdin hitz batetik bostera bitarteko tamainakoak

Datu-busaren zabalera: 16 bit edo 32 bit

Gutxieneko tamaina: 1 byte

Helbide-busaren zabalera: 24 bit ( $2^{24}$  byte = 16 Mbyte helbideragarria den memoria)

- Erregistroak:

ÿ Xede orokorreko 8 Datu Erregistroak (16/32): D0-D7

ÿ 7 Erabilera Orokorreko Instrukzio Erregistroak (16/32): A0-A6

- helbideratze moduak

ÿ #: berehalakoa

ÿ Esan: erregistro zuzena

ÿ (Ai): inskripzioa zeharkakoa

ÿ +(Ai): inkrementaren ondorengo erregistroa zeharkako eragiketa-tamaina eskalatuz (1,2,4 byte)

ÿ (Ai)+: inkrementaren ondorengo erregistroa zeharkako eragiketa-tamaina eskalatuz

ÿ -(Ai): zeharkako erregistroa aurredekretuarekin eragiketa-tamaina eskalatuz

- ÿ (Ai)-: zeharkako erregistroa aurreinkrementarekin eragiketa-tamaina eskalatuz
- ÿ D(Ai): zeharkako erregistroa D desplazamenduarekin
- ÿ D(Ai,Ri,X): zeharkako erregistro indexatua D desplazamenduarekin Ai Ri
- ÿ D(PC): D desplazamendua duen PCarekiko
- ÿ D(PC,Ri,X): D offset duen Ri indexatutako PCarekiko erlatiboa

- Datuak

- ÿ Zenbaki osoak 2 osagarrian.
  - ÿ Eragiketa-atzizkiak: B byte (1 byte), W hitza (2 byte), L luzea (4 byte)
  - ÿ Datu-aurrizkiak: \$ hamaseitarra

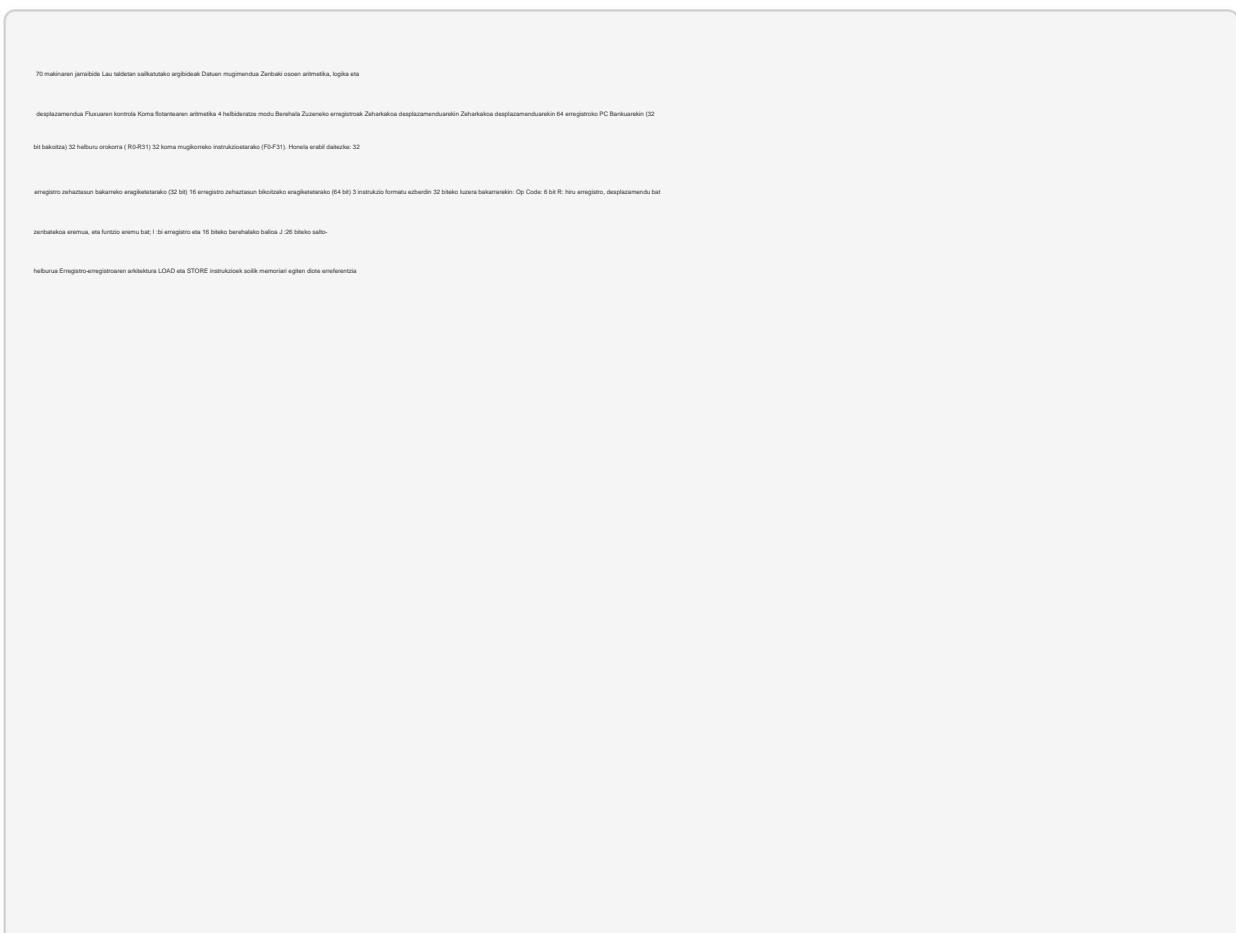
- Memoria

- ÿ Big Endian: LSB noranzko altuenean eta MSB noranzko baxuenean

## 26.6. MIPS arkitektura

### 26.6.1. DA

- 32 biteko arkitektura duen prozesadorea
- Interlocked Pipeline Stages (MIPS) Arkitekturariik gabeko mikroprozesadorea
- MIPS arkitektura bertsioak:
  - ÿ MIPS I (R2000 CPU), II (R6000), III (R4000), IV (R8000, R5000, R10000) eta V (inoiz ez ezarri);
  - ÿ MIPS32/64: MIPS32 MIPS II-n oinarritzen da, MIPS III, MIPS IV eta MIPS III-ren ezaugarri gehigarri batzuekin. MIPS V; MIPS64 MIPS V-en oinarritzen da



Gainontzeko argibideek erregistroetan funtzionatzen dute Hiru  
eragiketa dituzten instrukzioak: 2 iturburu eta 1 helmuga.

Batzar-notazioa: op x, y, z                             $x <- (y) op(z)$

Datuak:

2-ren osagarri osoak: byte (1B), hitz erdia (2B), hitza (4B)

Benetako zenbakia: IEEE-754 doitasun bakarra eta bikoitza

- [MIPS arkitektura](#)
- [ISA MIPS bertsioak](#)
- [MIPS arkitektura duten prozesadoreak](#): R2000, etab.
- [tutorial azkarra](#)
- [MIPS lineako emuladorea](#)

# 27. kapitulua. Tresna-katea: Tresna-katea konpilazio-prozesuan

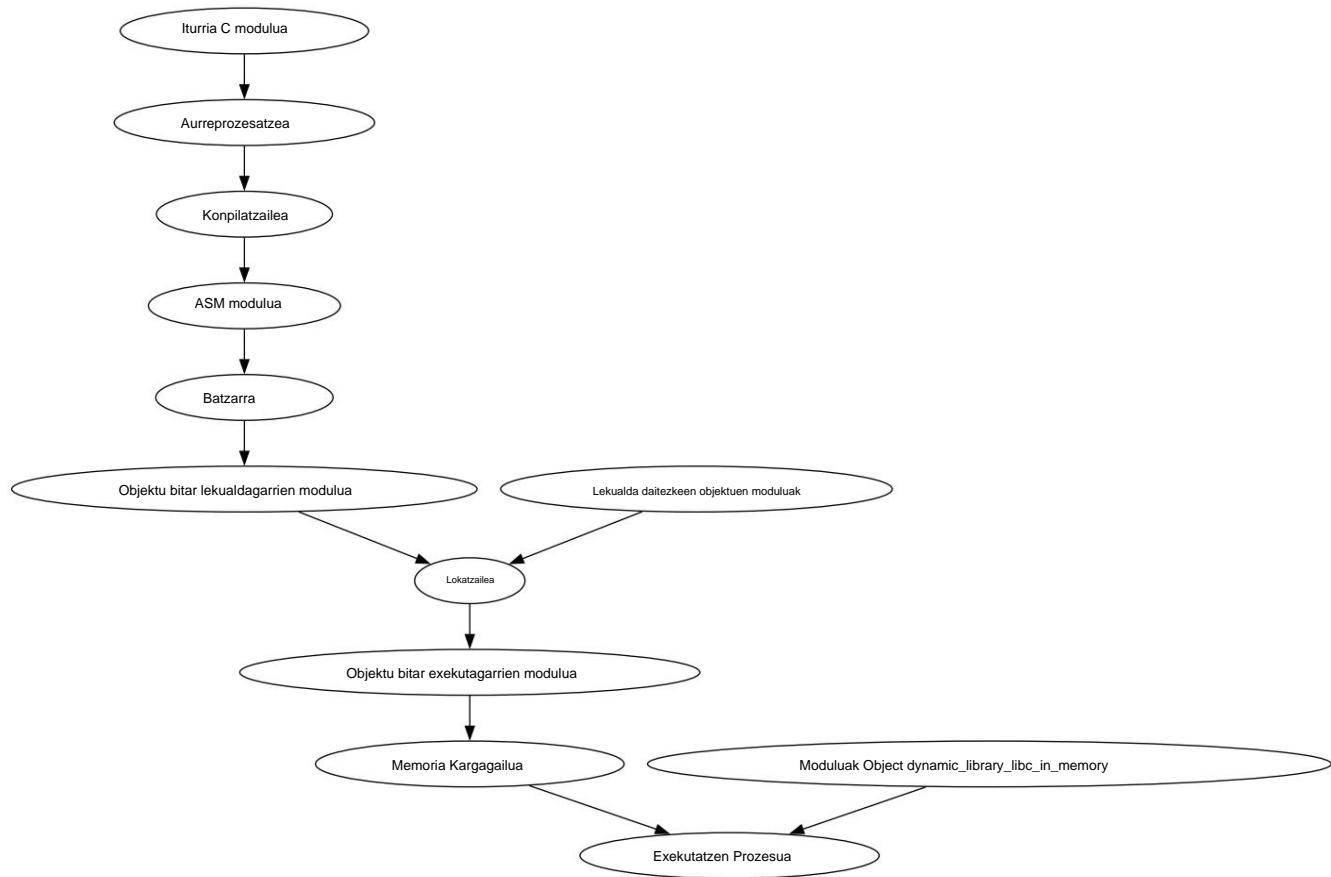
## 27.1. Tresna-katea

- Toolchain hizkuntza sinbolikoetako moduluetatik makina-lengoaian aplikazio bat lortzeko esku hartzen duten programa edo tresna desberdinak bezala ezagutzen dira, hala nola C, muntatzailea eta liburutegiko moduluak.
- Itzulpen prozesuan 3 tresna nagusiak hauek dira: konpilatzzailea, muntatzailea eta lokailua. Programak hizkuntza bitarrean aztertzeko tresnak ere badaude (Pej desmuntatzailea).

ÿ binutilak:

- ÿ **gisa: itzultzzailea muntatzailea:** komando-lerroko aukerak, zuzentaraauak, datu motak, etab.
- ÿ **ld: lokailu**
- ÿ **objdump, readelf, ....**

## 27.2. Programa baten konpilazio prozesua C lengoaian



## 27.3. Muntaketa prozesuko itzultzzaileak

- i386/amd64 ISArako bi mihiztatzaile-lengoaia daude, bi sintaxi ezberdin, makina-lengoaia berdinean modulu bitar bat ekoizten duten muntaia-itzultzzaile ezberdinek itzuli behar dituztenak.

- "NASM" (Netwide Asm), "FASM", "MASM", "TASM" eta "YASM" itzultzalee mihiztatzaileek **Intel** mihiztadura-lengoaiaren sintaxia erabiltzen duten iturburu-moduluak i386/amd64 arkitekturarako modulu bitar bihurtzen dituzte .
- GNU Fundazioaren "ace" itzultzaleak GNU hizkuntzaren sintaxiak erabiltzen dituzten iturburu-moduluak itzultzen ditu. **AT&T** muntatzalea i386/amd64 arkitekturarako modulu bitarretarako
- Disko gogorrean fitxategi batean gordetako iturburu-moduluaren irudikapenaren eta makina-lengoaiaren dagoen programaren irudikapenaren arteko itzulpena muntaketa-prozesuan gertatzen da :

ÿ Iturburu-modulua mutua-lengoaien ÿ Mihiztatzaile itzultzalea ÿ Tokiz alda daitekeen objektuaren modulua ÿ  
Lokatzalea ÿ Objektu exekutagarri bitar modulua ÿ Kargatzalea ÿ Prozesua

ÿ Adibidea: hello\_world.s ÿ **as** ÿ hello\_world.o ÿ **ld** ÿ hello\_world (ELF formatua) ÿ **loader** ÿ hello\_world (memoria nagusia)  
ÿ prozesuaren sorerra (hello\_world martxan). "as", "ld" eta "loader" dira mihiztadura-lengoaiaren modulutik prozesu bat  
sortzeko beharrezkoak diren GNU tresnak.

ÿ **ld** estekatzaleak objektuaren modulua sistema eragilearen inguruneko moduluekin eta moduluekin nahasten du  
liburu-dendetatik.

## 27.4. "Aso" muntatzalea

### 27.4.1. Zuzentaraauak

- GNU/linux sistemak hizkuntzarako erabiltzen duen "as" mutua-itzultzalearen zuzentaraauak AT&T muntatzalea.

ÿ GNU mihiztatzailea "gas" bezala ere ezagutzen da.

ÿ [binutilak](#) ÿ [muntatzale gisa](#): eskuliburu ofiziala

ÿ [gas erreferentzi txartela](#)

ÿ .word-ek 2 byte gordetzen ditu amd64-n.

ÿ [orakulua](#)

ÿ [x86 eta x86-64 aukerak](#)

#### Eskuliburua

- [binutils gisa](#): eskuliburu ofiziala
- [Linux Assembly nola](#): muntatzale ezberdinaren erreferentziak
- [M.I.T.](#)

# 28. kapitulua. Programazioa hasieratik lantzea

## 28.1. Dokumentazioa: gidoiak, bibliografia, oharrak

- Meowary-n eskuragarri:

ÿ Gidoiak, erreferentzia-orriak, eranskinak, autoebaluazio-ariketak eta teoria biltzen dituzten eecc\_book.pdf oharrak.

ÿ Iturburu kodea .s duten moduluak (nire liburua/Baliabideak/praktikak/iturburu\_kodea.zip) [esteka G1](#) praktika guzietan erabiltzen dira NUPeko myaulario zerbitzarian eskuragarri: Baliabideak/praktikak

ÿ Mihiztadura lantzeko gidoiak oinarri dituen testuliburu: [Programazioa Lurretik](#).

ÿ Mihiztadura-lengoaiaren programazio liburuaren sarrera: 'ASM programatzen... baina oso da erraza

## 28.2. Garapen Plataforma

### 28.2.1. Tresnak

- Editoreak

ÿ [Editoreak](#): gedit, emacs, vim, sublime, kate,

ÿ [Edizio, konpilazio eta arazketa tresna integratuak](#): eclipse CDT, netbeans, code::blocks, codelite, Microsoft en Visual Studio Code Editor, jetbrains clion, jeany, ajunta, [GNAT Programazioa](#) Studio, emacs, kdevelop, codestudio, etab.

- Deiturak

ÿ [i386](#) : x86-32 arkitekturarako Linux izena

ÿ [amd64](#): x86-64 arkitekturaren Linux izendatzea

ÿ [IA32](#): Intel izena x86-32 arkitekturarentzat

ÿ [IA64](#): Intel izena x86-64 arkitekturarentzat

- GNU/linux Sistema Eragilea: Ubuntu Banaketa: 2014tik aurrera edozein bertsio: 14.04, 14.08,...,17.04, 17.08

ÿ [lsb\\_release -a](#): askatu

ÿ [uname -o](#) : SO

ÿ [uname -r](#) : nukleoa

ÿ [uname -a](#) : prozesadorea

- Gcc, as, ld tresnak 32 [i386](#) arkitekturan funtzionatzeko beharrezkoak diren liburutegiak bitsak.

ÿ [dpkg -l gcc-multilib](#) :

Desiratua=ezezaguna(U)/Instalatu/KenduR/Purge/Erretain(H) |

Egoera=Ez/Inst/Files-Conf/Unpacked/Half-ConF/Half-Inst(H)/Abiaraztearen zain (W)/Abiaraztearen zain

```

// Err?=(bat ere ez)/requires-Reinst (Egoera,Err: caps=txarra)
// Izena Arkitektura Deskribapena
=====
ii gcc-multilib 4:7.3.0-3ubu amd64                               GNU C konpilatzalea (multilib fitxategiak)

```

ÿ "Desiratu/Egoera" lehenengo bi zutabeek **ii** esaten ez badute , esan nahi du liburu-dendak.

ÿ Internet bidez eskura daitekeen biltegian daudela egiaztatzen dut:

- ÿ [apt-cache show gcc-multilib](#) : biltegia
- ÿ [sudo apt-get install gcc-multilib](#) : deskargatu eta instalatu baduzu bakarrik administratzaile eskubideak

- Tresna-katea

ÿ [as --version & ld --version & gcc --version](#) : ohartu bertsioak

## 28.2.2. Sareko programazioa

- [ias assembler unicamp online](#)
- [gdb sarean](#)

## 28.2.3. Erreferentziak

- [GNU](#) Baliabideak :

ÿ IDE garatzeko tresna integratua (Emacs, Eclipse, Vim, etab...) edo Editore bat (Geany, Kate, Gedit, Sublime, etab...)

ÿ [gisa](#) : AT&T hizkuntza-muntatzailea

ÿ [ld](#) : lokailu

ÿ [cc](#) – C konpilatzalea

ÿ [GCC](#) : tresna-kate automatikoaren frontend-a: Gnu Compiler Collection. Iturburu-fitxategiaren luzapenaren arabera hizkuntza desberdinatarako kontrolatzalea.

ÿ [gizon gcc](#)

ÿ [GDB](#) : arazketa.

ÿ [gizon gdb](#)

## 28.3. Txostenaren dokumentua: edukia eta formatua

### 28.3.1. Edukia

- Praktikaren garapenean:
  - to. Garatutako programen iturburu-kodea berriro editatu behar da iruzkinekin.
  - b. Konpilatu iturburu-modulua lineako komandoak erabiliz
  - c. Aztertu iturburua eta kode bitarra arazketa erabiliz: araztearekin egin beharreko eragiketak Beharrezko da fitxategi batean gordetzea.
- Praktikan zehar Testu Editore bat irekita edukitzea beharrezkoa da memoria praktikaren burutzapenarekin batera.
- Memoria-dokumentuak eduki behar du:

- ÿ Praktikaren izenburua eta datu pertsonalak dituen azala.
- ÿ Aurkibide gisa aurkibide bat duen lehen orrialdea, ez da beharrezkoa orri-zenbakia adieraztea.
- ÿ Iruzkinen iturri-moduluak,
- ÿ Konpilazio eta analisi komandoak.
- ÿ GDB komandoen historia eta haien irteerak, praktikan zehar erabilitakoak.
- ÿ Ondorioen atal bat praktikan ikasitakoarekin.
- ÿ Ebatzi gabeko zalantzen atala.
- ÿ Memorian zehar agertzen diren galdera esplizituak, halakorik balego.
- ÿ **AUKERAZ** Praktiken Autoebaluazio galdetegiko galderak eta erantzunak. Ikusi Ebaluazio atala.
- ÿ Beharrezko informazio pertsonal mota guztiak azterketan erabiltzeko ohar gisa.

### 28.3.2. Formatua

- Memoriaren barne egitura librea da.
- Memoria-formatuak **PDF izan behar du**, eta ez Microsoft Word edo beste formatu ezberdin bat.
- Memoria-fitxategiaren izenak **N-XXX-abizena1\_abizena2.pdf** izan behar du
  - ÿ Fitxategiaren izenak ez du azenturik edo zuriunerik izango
  - ÿ XXX praktiken taldea esan nahi du: P1 edo P2 edo P1P2 edo P91
  - ÿ N praktika saioaren zenbakia esan nahi du: 1,2,3,4 edo 5.

### 28.3.3. Memoria Dokumentua entregatzea

- Bidali Memoria Dokumentua Myaulary Server **Tasks** aplikazioaren bidez. Epea irakasleak esleipen egutegiaren bidez adieraziko du. Txostenak epez kanpo aurkezteak ohiko deialdian praktika hori egiteko aztertu behar izatea dakin.

## 28.4. Ebaluazioa

- Baloratuko dira:
  - ÿ Ezarritako kanalaren bidez memoria entregatzea puntu 1eko zigorrarekin egun bakoitzeko atzerapena.
  - ÿ Txostenaren egitura eta formatua datu pertsonalekin, aurkibidea, sarrera, garapena, ondorioak eta pdf formatua dagokion izenarekin.
  - ÿ Iturburu-moduluan zehaztutako goi-mailako iruzkinak (pseudokodea) bai instrukzio-blokeen mailan, bai interpretatzeko konplikatuak diren edo kodea ulertzeko garrantzitsutzat jotzen diren argibideak.
  - ÿ **Praktika Autoebaluaziorako aukerako galdetegia**. Praktikaren autoebaluazioko ariketak egiten ez badira eta txostenei gehitzen bazaizkie, txostenen gehienezko nota **6 puntukoa izango da**.

ÿ

Irakasleak etengabe evaluatuko ditu ikaslearen jarrera eta laborategian egindako lana, eta ikaslea azterketa egiteatik askatu ahal izango du, egindako ezagutzak eta eginkizunek hala erakusten badute.

## 28.5. Programazioa

### 28.5.1. Metodologia

- Garatu beharreko programaren adierazpena irakurri.
- Editatu algoritmoaren deskribapena Pseudokode gisa:
  - ÿ Algoritmoa garatu datu-egiturak eta instrukzio-egiturak zehatzuz.
  - ÿ konstanteak, aldagaiak, matrizeak, erakusleak, hasieraketak, begiztak, hautaketa-adierazpenak, funtzieak eta parametroak, programaren sarrera eta irteera, etab.
- Goi-mailako Organigrama marraztu
  - ÿ Maila handiko hizkuntza baterako (Pascal, C...), pseudokodean oinarrituta.
- Maila baxuko organigrama marraztu
  - ÿ Algoritmoa garatzea RTL hizkuntzan x86 arkitekturan oinarrituta. Itzuli organigrama goi-mailatik behen mailara. Atalak, aldagaiak, matrizeak, erakusleak, hasieraketak, begiztak, hautaketa-adierazpenak, azpierrutinak eta parametroak, programaren sarrera eta irteera, etab.
- Bihurtu RTL kodea x86 arkitekturarako AT&T mutua-lengoiaia.
- Gcc-rekin edo tresna-katearen bidez biltzea: as-ld
  - ÿ Araztu sintesi-erroreak.
- Exekuzioa: Araztu erroreak urratsez urrats moduan GDB arazketa erabiliz

## 28.6. Konpilazioa

### 28.6.1. Iturburu-modulua C hizkuntzan

- Konpilazioa
  - ÿ gcc -m32 -o sum1toN sum1toN.c
    - ÿ m32: 32 biteko arkitekturako makina
    - ÿ sum1toN.c: iturri-modulua C hizkuntzan
    - ÿ -o : irteera
    - ÿ sum1toN luzapenik gabe: objektu exekutagarriaren modulua, nahiz eta zehatzagoa litzatekeen kargagarria esatea memoria nagusia.
    - ÿ memoria nagusian kargatu
      - ÿ Sistema eragileak automatikoki egiten du programa exekutagarria terminal batetik edo mahaigainetik deitzean.
    - ÿ gcc -m32 -g -o sum1toN sum1toN.c
      - ÿ -g: GDB arazketarako sum1toN.c iturburu-programatik ikur-taula sortzea eta sum1toN modulu exekutagarrian txertatzea zehazten du. Horrela, kode bitarra, etiketa batena adibidez, bere sinboloarekin (testu hizkuntza) lotzen da.

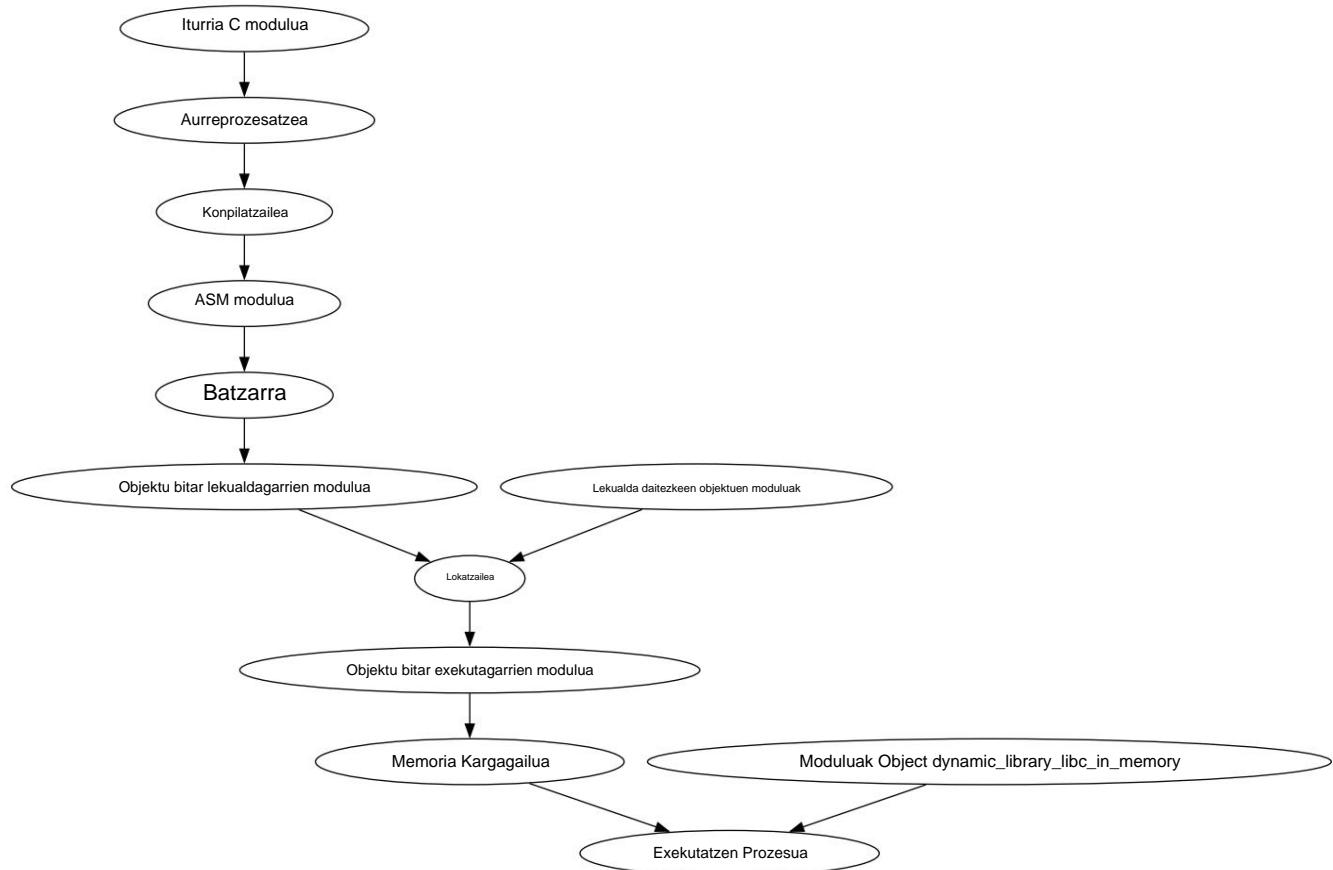
### 28.6.2. Programaren sarrera-puntuak: main vs \_start

- Programarako sarrera-puntuak \_start edo etiketa nagusia erabiliz izendatu behar da.
  - ÿ Zuzentara globala: sarrera-puntuak iturburu-moduluan definitzen da eta global deklaratzen da, hau da, beste programek eskura izan behar dute. Sarrera-puntuak kanpoan definitutako ikur gisa deklaratzen duen estekatzaleak eskura izan behar du.
  - ÿ Gcc konpilatzailearen -nostartfiles aukera zehazten ez bada, sarrerako etiketa

muntaia programa nagusia izan behar da. -nostartfiles aukera zehazten bada, programaren sarrera \_start izango da

- Konpilazioa sarrera-puntu \_hasiera bada: gcc -nostartfiles -g -m32 -o sum1toN sum1toN.s
- Konpilazioa sarrera-puntu nagusia bada: gcc -g -m32 -o sum1toN sum1toN.s

#### 28.6.3. Konpilazio faseak



111. Irudia Konpilazio-prozesua

- Gelditu konpilazioa 1. fasean: aurreprozesatzea: gcc -E sum1toN.c -o sum1toN.i
  - ÿ \*.i: Aurreprozesadorearen irteera: kodea ez den informazioa kentzen du (iruzkinak, etab.)
- Gelditu konpilazioa 2. fasean: itzuli C mutua: gcc -S sum1toN.c -o sum1toN.s
  - ÿ .s: mutua-iturburu-hizkuntzako modulu.s
- Konpilazioa gelditu 3. fasean: Sortu lekuz alda daitekeen objektuaren modulu: gcc -c sum1toN.c -o sum1toN.o
  - ÿ \*.o: lekuz alda daitekeen objektu-modulu: estekatzalearen bidez sistema eragleko beste objektu-modulu batzuekin, C libc liburutegiarekin edo programatzalearen beste modulu batzuekin lotu aurretik kode bitarra.
- Egin 4 faseak: Objektu exekutagarriaren modulu sortu: gcc -c sum1toN.c -o sum1toN
  - ÿ luzapenik gabeko fitxategia: moduluaren objektu exekutagarria: modulu bitarra fitxategian kargatzeko konfiguratura memoria nagusia eta CPUak exekutatua.
- gcc -m32 --save-temps -o sum1toN sum1toN.c
  - ÿ --save-temps: gcc -k .,.,s konpilazio prozesu osoaren 3 fitxategi partzialak (gordetzen ditu) sortzen ditu ,\*,edo .
- ÿ Egiaztu guztira 5 fitxategi ditugula: .c, .i, .s, .o eta luzapenik gabeko exekutagarria.

## 28.6.4. Tresna-katea

- Konpilazio-fase desberdinak exekutatzen dituen **gcc** kontrolatzalearekin komando bakarra erabiliz konpilazioa egitearen alternatiba gisa , konpilazio-prozesua fase desberdinako bat egiten duten tresnak kateatuz egin daiteke.
- Tresna-katearen tresnak:
  - ÿ Itzulpena C-tik Muntatzailera: ez du tresna propiorik: **gcc -S sum1toN.c -o sum1toN.s**
  - ÿ **as**: Muntatzeko tresna edo muntatzailea: **as --32 --gstabs -o sum1toN.o sum1toN.s**
    - ÿ --32: 32 biteko arkitektura
    - ÿ --gstabs: sinboloen taula sortzen du
    - ÿ -o: irteera fitxategia: lekuz alda daitekeen modulu bitar \*.o
  - ÿ **ld**: Lotzeko tresna: **ld -melf\_i386 -o sum1toN sum1toN.o**
    - ÿ -melf\_i386: 32 biteko arkitektura
    - ÿ -o: irteera fitxategia: modulu bitar exekutagarria

## 28.6.5. iturburu-modulua mihitzadura-lengoain

- Iturburu-programari buruzko iruzkinak modu funtzional/operatibo abstraktuan eta ez RTL modu literalean.
- Eskuzko tresna-katea:
  - ÿ **as --32 --gstabs -o sum1toN.o sum1toN.s** : muntaia
    - ÿ \*.s: iturburu-modulua asm hizkuntzan
    - ÿ \*.o : lekuz alda daitekeen objektu-modulua
    - ÿ --gstabs: sinboloen taula sortzea eta modulu exekutagarrian txertatzea.
    - ÿ --32: 32 biteko ISArako iturburu eta objektu moduluak
  - ÿ **ld -melf\_i386 -o sum1toN sum1toN.o**
    - ÿ -melf\_i386: 32 biteko ISArako objektu moduluak
- Tresna-kate automatikoa
  - ÿ **gcc -m32 -nostartfiles -g -o sum1toN sum1toN.s**
    - ÿ -m32: i386 arkitekturarako iturburu eta objektu moduluak.
    - ÿ -nostartfiles : sarrera-puntuak ez dela nagusia zehazten du, \_hasiera baizik.

ÿ

Sarrera-puntuak **nagusia** bada , estekatzaileari jakinarazi behar zaio sarrera-puntuak nagusia dela: **gcc -e main -m32 -nostartfiles -g -o sum1toN sum1toN.s** eta **ld -e main -melf\_i386 -o sum1toN sum1toN.o**

- g: iturburu-programaren sinboloen taula sum1toN.s GDB arazketarako sortuko dela zehazten du eta sum1toN modulu exekutagarrian txertatzen da

## 28.7. Depurazioa

- Erabiltzen dugun arazketa GBU gdb programa da: Gnu DeBugger
- [GDB](#) eskuliburua
- Programa bat araztu ahal izateko, **-g** arazketa aukerarekin konpilatu behar da: **gcc -g -m32 -o sum1toN sum1toN.s**
  - ÿ egiaztatu arazketarako ikurrak dituen **sum1toN fitxategiarekin**

- Exekutatu gdb ÿ araztearen komando-leihoa irekitzen da. Gonbidapena (gdb) da

```
(gdb) sum1toN fitxategia -> programa bitarreko sinbolooak irakurtzea
(gdb) breakpoint main -> gehitu eten puntu bat non exekuzioa geldituko den (gdb) exekutatu ->
exekuzioa (gdb) layout
src edo Control-x Control-a -> leihoa iturburu kodea (gdb) focus src edo Control-x gehitzen da
edo -> leihoa aldatzen da, non teklatuko geziak aktibatzen diren nabigatu ahal izateko.
```

- Komandoek erabiltzen dituzten esamoldeak C programazio-lengoaian daude

```
inprimatu &sum -> batura aldagaiaren helbidea inprimatzen
du inprimatu batura -> batura aldagaiaren edukia inprimatzen
du *sum_pointer -> sum_pointer erakusleak adierazitako objektuaren edukia inprimatzen du
```

## 28.8. Ohiko akatsak

### 28.8.1. gcc

- Ubuntu 18.0-n amd64rako konpilatzen baduzu (gcc -nostartfiles -g -o sum1to64 sum1to64.s) konpilazioa errore-mezuarekin gelditzen da:

```
/usr/bin/x86_64-linux-gnu-ld: /tmp/ccbhD6Vr.o: ` .data ' ren aurkako R_X86_64_32S
Ilekualdatzea ezin da erabili PIE objektu bat egiterakoan; birkonpilatu -fPIC-rekin /
/usr/bin/x86_64-linux-gnu-ld: azken esteka huts egin du: atala ezin da irteeran errendatu

collect2: errorea: ld-k 1 irteera-egoera itzuli du
```

ÿ kausa: lehenespenez -pie aukerarekin aktibatzen da eta desaktibatu egin behar da

ÿ irtenbidea: (gcc -no-pie -nostartfiles -g -o sum1to64 sum1to64.s)

### 28.8.2. gdb

- Fitxategi batean gordetzeko gdb komandoen erregistro historikoa desgaituta dago

## 28.9. Programa eta Araztu hutsetik

### 28.9.1. Hasteko: ASM

tresnak

- gdb --bertsioa
- gcc --bertsioa

## ASM iturri-modulua

- batuketa1toN.s

```
### Programa: sum1toN.s
### Deskribapena: 1,2,3,...5 serieen batura egiten du
### gcc -m32 -g -nostartfiles -o sum1toN sum1toN.s
### Asanblada gisa --32 --gstabs source.s -o object.o
### estekatzalea -> ld -melf_i386 -l/lib/i386-linux-gnu/ld-linux.so.2 -o exekutagarria
objektua.o -lc
```

```
## Aldagaien adierazpena
.atala .datuak
```

n: .int 5

```
.global _hasiera
```

```
## Kodearen hasiera
.atala .testua
```

\_hasi:

```
mov $0,%ecx # ECX-k batura aldaagaia ezartzen du
mov n,%edx
```

begizta:

```
gehitu %edx,%ecx
azpian $1, %edx
jnz begizta
```

```
mov %ecx, %ebx # irteerako argumentua OSra EBX bidez
```

akordioa

```
## irten
mov $1, %eax # sistema eragilearen deien kodea: irten azpierrutina
int $0x80 # OS deia
```

```
.amaiera
```

## Konpilazioa

- \_hasierarekin: gcc -nostartfiles -g -m32 -o sum1toN sum1toN.s
- Nagusiekin: gcc -g -m32 -o sum1toN sum1toN.s

## Modulu bitarren propietateak

- Modulu bitarra: sum1toN fitxategia

## Exekuzioa

- ./sum1toN
- oihartzuna \$?

## Arazketa

- **gdb arazketa**

```
fitxategia
sum1toN layout src edo Control-x Control-a
break main edo b main
run edo
r rTAB -> autoosatzeko laguntza
r -> r exekutatu al da?
inprimatu n edo
pn inprimatu /
xnp /
tnp /on
hurrengo
edo
nn 6 jarraitu edo c
hasi edo s
n 6
p $ecx

ezarri $ecx=-1
p $ecx
info regs
layout regs
layout src

ezarri var n=10
ezarri var &n=10
ezarri var {int}&n=10
ezarri var {int *}&n=10

ezarri var {char[10]}&n="Kaixo" p /sn

ezarri $pc=&main

inten
```

```
info iturriak shell
ls
help layout
layout src
layout asm
layout split
laguntza
p inprimatzeko formatuak: /t /x /o /s /a
```

## 28.9.2. Hasteke: C

- batuketa1toN.c

```
/*
Programa: sum1toN.c

gcc -g -m32 -o sum1toN sum1toN.c fitxategia
sum1toN .
sum1toN
echo $?

*/
// Exit() funtziaren adierazpena #include
<stdlib.h>

// Modulu nagusia int main
(void) {
    // Aldagai lokalen deklarazioa eta begiztaren parametroen hasieratzea int batura=0,n=5;

    // Gehigarriak sortzen dituen eta batuketa egiten duen begizta
    while(n>0){ // Begizta irteera baldintza gehigarria negatiboa denean
        batura+=n;
        n--; } irten          // Gehigarria eguneratzentzat

    (batura); } //irten:
programaren exekuzioa amaitzen du eta batura argumentua itzultzen du S.op.
```

- Konpilazioa: `gcc -g -m32 -o sum1toN sum1toN.c`
- Modulu bitarra: `sum1toN fitxategia`
- `gdb` arazketa

```
sum1toN fitxategia
layout src
break main

Korrika egin

hurrengoa jarraitu
hasi
hurrengo 6
inprimatu
batura irten
```

sum1toN fitxategia -> sinboloen taula berrespena diseinua  
src eten puntu  
nagusia edo b main

r edo

exekutatu hl edo

lagunza zerrenda n edo hurrengo -> exekutatu iturburu-moduluaren hurrengo

lerroa n 6 -> exekutatu 6 aldiz nc edo

jarraitu

bai hasi

zer kendu duzu

pmota batura

zer da batura

p /d sum p /

x sum p /t

sum p /o

sum p /a

&sum

p sum

p &sum

set var sum=22

p

batura multzoa var {int}

&sum=-3 p /x batura

+

informazio iturriak

shell ls

lagunza

diseinua src

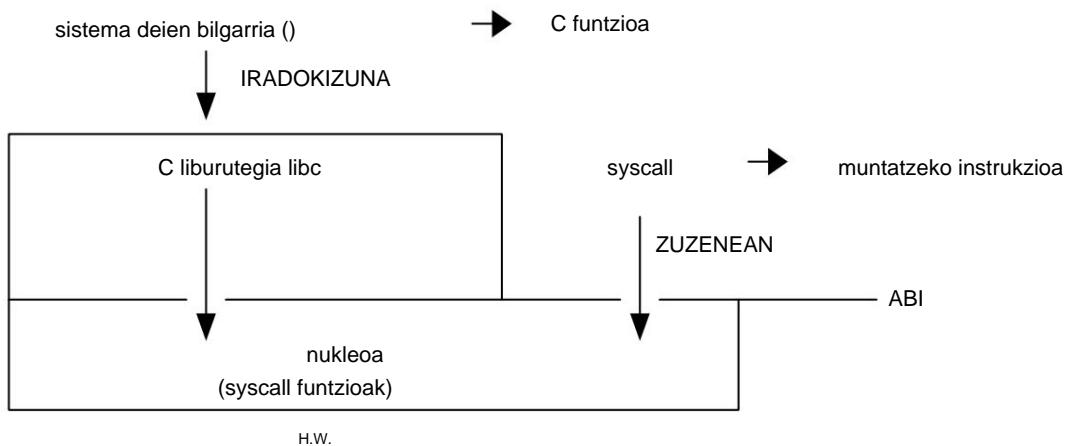
diseinua asm

diseinua zatitu

# 29. kapitulua. Sistema eragilerako deiak

## 29.1. Sarrera

- Erabiltzaile-programak sistemara egiten dituen deiei sistema-deiak deitzen zaie. Sistema eragilearen Kernelaren azpierrutinak.
- Sistema eragilearen funtziotako pribilegiatuak betetzeko, hala nola, ordenagailuko I/O gailuetarako sarbidea, beharrezko da erabiltzailearen programek nukleoa deitzea, eragiketa modu seguru eta eraginkor batean egin dezan. Honek aplikazioaren programatzalea hardwarera sartzea eragozten du eta, aldi berean, programazioa errazten du.
- Deien adibideak
  - ÿ **irten** : nukleoak programaren exekuzioa eten egiten du prozesua hilez
  - ÿ **irakurri** : nukleoak fitxategi bateko datuak irakurtzen ditu disko gogorran sartuta
  - ÿ **idatzi**: nukleoak fitxategi batean idazten du
  - ÿ **ireki** : nukleoak fitxategi bat irekitzen du
  - ÿ **itxi**: nukleoak prozesua ixten du
  - ÿ dei-adibide gehiago man **2 syscalls** zerrendan
- Syscalls izeneko nukleo-zerbitzuei deia bi modutara egin daiteke: **zuzenean** edo **iradokizuna**
  - ÿ Zuzena: ASMtik **syscall** sententzia erabiliz
  - ÿ Zeharkakoa: C edo ASMtik **libc liburutegiko funtzioak erabiliz**: zuzeneko deietarako bilgarriak
- API/ABI



- Adibidea

```
* printf() -> write(int fd, const void *buf, size_t count) -> [RAX-RDI-RSI-RDX-R10-R8-R9,syscall] -> kernel syscall idazketa
* API bilgarri funtzioa -> kernel syscall
```

-> ABI

## 29.2. Deitu eskuliburuak

- Syscall-ak libc liburutegien bilgarrien eskuliburueta deskribatzen dira
- syscall-en zerrenda

ÿ info syscalls edo **man syscalls**

- sistema deiak:

ÿ irten ÿ **gizon 3 irten**

ÿ irakurri ÿ **gizon 2 irakurri**

ÿ idatzi ÿ **gizon 2 idatzi**

ÿ ireki ÿ **gizon 2 ireki**

ÿ itxi ÿ **gizon 2 itxi**

ÿ etab.

- Sistema-deiaren argumentuak libc liburutegiko wrapper funtziorekin lotutakoak dira.

ÿ Sistema-deiaren 1. argumentua libc-ko funtziaren EZKERRA argumentua da eta azkena.  
ESKUBIDEA.

## 29.3. ZEHARKAKO deia

- C: C aplikazio-programatzaleak GNU libc liburutegiko interfaze-funtzioak erabiltzen ditu  
**zeharka** sartu nukleoa bilgarrien bidez.

ÿ system calls wrapper: ASM hizkuntzan implementatutako deien C hizkuntzara egokitzea.

## 29.4. Deitu zuzenean

- **ASM:** ASM hizkuntza-aplikazioen programatzaleak sistema-deiak erabiltzen ditu sartzeko  
**zuzenean** nukleora

ÿ Deia x86-64-n **syscall** muntaketa-instrukzioa eta **int \$0x80** x86-32-n erabiliz egiten da.

ÿ Deiaren argumentuak GPR helburu orokorreko erregistroen bidez pasatzen dira

ÿ Dei mota zenbaki oso baten bidez zehazten da eta **RAX** bidez pasatzen da

ÿ "Sistemaren dei-zenbakia" kodeak /usr/include/asm/unistd\_32.h fitxategian eskuragarri

### **## Makroak macros\_x86-64\_gas.h fitxategian**

#### **## Sistema-deiak**

STDIN_ID, 0	# sarrera-gailu (teklatua)
STDOUT_ID, 1	# irteera-gailu (pantaila)
.equ .equ .SYSCALL_READ, 0	# "irakurtzeko" ID -zenbakia
.equ SYS_WRITE, 1	# "idatzi" ren ID zenbakia
SYS_OPEN, 2	# "ireki" -ren ID-zenbakia
SYS_CLOSE, 3	# Ixteko ID zenbakia
.equ .equ .SYSCALL_EXIT, 60	# Irteerarako ID -zenbakia

### 29.4.1. Dei zuzeneko argudioak

- Deiaren konbentzia ABI estandarrean deskribatzen da

- x86-64

ÿ Deiaren lehenengo 6 argumentuak erregistroetatik pasatzen dira sekuentziari jarraituz:  
RDI-RSI-RDX-R10-R8-R9

ÿ Deia itzultzeko balioa RAX erregistrotik pasatzen da

- x86-32

ÿ Lehenengo 6 argumentuak sekuentziari jarraituz erregistroetatik pasatzen dira: EBX-ECX  
EDX-ESI-EDI-EBP

ÿ Deia itzultzeko balioa EAX erregistrotik pasatzen da

- libc eskuliburua: deiaren argumentuak zein direnei buruzko informazioa

#### 29.4.2. Zuzeneko dei-kodeak

- Dei-kodea nukleoak exekutatu behar duen funtziorekin lotutako zenbaki oso bat da.
- Dei-kodea nukleora pasatzen da RAX bidez
- Kodeak:

ÿ /usr/include/asm/unistd\_64.h: arkitekturako dei-kodearekin makro-adierazpena  
x86-64

ÿ irten ÿ 60, irakurri ÿ 0, idatzi ÿ 1, ireki ÿ 2, itxi ÿ 3, etab.

ÿ /usr/include/asm/unistd\_32.h: arkitekturako dei-kodearen makro-adierazpena  
x86-32

ÿ /usr/include/bits/syscall.h : makro zaharrak x86-32 arkitekturan ere balio dute

#### 29.5. Adibideak: C hizkuntza

- irten (egoera\_balioa) eta syscall (irteera\_kodea, egoera\_balioa)
  - ÿ irten (0xFF) eta syscall (60,0xFF)
- idatzi (int fd, const void \*buf, size\_t count) eta syscall (write\_code,int fd, const void \*buf, size\_t zenbaketa)
  - ÿ idatzi (0,buffer,80) eta syscall (1,1,buffer,80)

#### 29.6. Adibideak: ZEHARKAKO ASM

- ASM lengoian programatzea libc liburutegiko wrapper-ei dei diezaiekegu.
- irten(egoera\_balioa)

```
mov $status_value,%rdi
deiaren irteera
```

- syscall (irteera\_kodea, egoera\_balioa)

```
mov $60,%rax
mov $status_value,%rdi call
syscall
```

- idatzi (int fd, const void \*buf, size\_t count)

```
mov fd,%rdi #fd idatziko den fitxategiaren erreferentzia da mov $buffer_address_label,
%rsi #memoria helbidea fitxategian idatziko den mov tamainan,%rdx dei idazketa

#memoria buffer-aren #tamaina idatziko den #idazketa
ordena nukleoan libc liburutegiaren bidez
```

- `syscall(write_code,int fd, const void *buf, size_t count)`

```
mov $1,%rax
mov $1,%rdi          # 1 pantailako fitxategiaren kodea da . Unix-en gailuak
fitxategiak dira.
mov $buffer_address_label,%rsi
mov size,%rdx
call syscall
```

## 29.7. Adibideak: ASM ZUZENEAN

- irten

```
mov $60,%rax
mov $status_value,%rdi
syscall
```

- idatzi

```
mov $1,%rax
mov $1,%rdi          # 1 pantailako fitxategiaren kodea da . Unix-en gailuak
fitxategiak dira.
mov $buffer_address_label,%rsi
mov tamaina,
%rdx syscall
```

## 29.8. Komando-lerroa

### 29.8.1. Procedura

- Prozesuaren Hastapena

ÿ Komando edo programa bat shell komando-lerroan idazten dugunean, sistema eragileak kate sekuentzia gisa interpretatzen ditu. Adibidez `$sum 2 3` komando lerroko hiru argumentu dira:

- ÿ Kodeketa bere karaktereen sekuentzia ASCII kodean da eta amaieran.  
0x00 kdea duen NULL karakterea
- ÿ "batura" ktea: 5 ASCII karaktere: 0x73,0x75,0x6d,0x61,0x00
- ÿ "2" ktea: 2 ASCII karaktere: 0x32,0x00
- ÿ "3" ktea: 2 ASCII karaktere: 0x33,0x00

ŷ Lerroan 3 argumentu daudenez, argument counter argc parametroa 3 izango da.

ŷ Hiru komando-lerroko kateak, "sum"- "2"- "3", kateen array aldagaiari esleitzen zaizkio argv

ŷ argv[0] "sum" katera seinalatzen du

ŷ argv[1] "2" katera seinalatzen du

ŷ argv[2] "3" katera seinalatzen du

ŷ argv[argc] NULL karakterea seinalatzen du

ŷ argv erakusleen array bat da, beraz, (char \*\*)argv motakoa da

- nukleoa

ŷ Nukleoak extern int main prototipoa deklaratzen du (int argc , char\* argv[] , char\* envp[]) ;

ŷ modulu nagusiaren adierazpena eta definizioa

ŷ Funtzio nagusia nukleoak global deklaratzen du eta erabiltzaileak definitzen du.

ŷ argc argumentu-zenbaketa ez-negatiboa da;

ŷ argv argumentu-kateen array bat da, argv[argc]==0 duena;

ŷ envp ingurune-kateen array bat da, erakusle nulu batekin amaitzen dena ere.

```
#include <stdio.h>
#include <stdlib.h>

/*
 * Komando-lerroan programa eta argumentu bat sartzen ditugu * Argumentuak zuriuneak baditu,
komatxo bakunen artean sartu: 'Kaixo Mundua' * gcc -g -o komando-lerroa komando-lerroa.c * ./programa
'Kaixo Mundua' */

int main (int parc, char *parv[]) { if (parc==1)

{ printf("Sartu edozein
mezu komando lerroan\n\n"); irten (EXIT_FAILURE); } printf("%s\n",parv[1]); itzuli EXIT_SUCCESS; }
```

### 29.8.2. Pila hasieratzea

- Main() funtzioa edo \_start instrukzioa exekutatzen hasten denean, pilaren egoera hau da:
- Pila hasieratzea

ŷ Nukleoak argc eta argv argumentuak main funtzioko globalari pasatzen dizkio STACK bidez. Funtzioa nagusia deitzen den funtzioa da.

19. taula. ABI Hitzarmena: Pila

StackReference	Interpretazioa
	argumentu-kateak
	0
1 hitz aldagai bakoitzean	Ingurugiroaren erakusleak
8+8*argc(%rsp)	0
8*argc(%rsp)	- argc <sup>0</sup> katearen erakuslea
.....	.....
16(%rsp)	- 2. argumentuaren erakuslea katea ý argv[1]
8(%rsp)	- 1. argumentuaren erakuslea katea ý katea argv[0]
0(%rsp)	- argumentu kopurua ý argc

### 29.8.3. Errutina nagusia Itzuliarekin

- Errutina nagusia irteera -deiarekin amaitzen ez bada eta ret instrukzioarekin amaitzen bada , dei-konbentzia funtziodeia da, beraz, argc eta argv parametroak **RDI** erregistroetatik pasatzen dira  
**RSI-RDX-RCX-R8-R9**
- Adibidea: print\_arg.s

```
### gcc print_arg.s
### ./a.out 'Kaixo mundua'
###
###

.equ STDOUT,1
.equ SYSWRITE,1
.equ EXIT_SUCCESS,0xFF
.equ ARGV1,8

mezua:
.ascii "Idatzi mezu bat programaren argumentu gisa. Mezua bada
zuriuneak ditu, jarri mezua komatxo bakarren artean "\n"
.equ LON,. - mezua           #mezuanen luzera

.atala .testua
.global nagusi

nagusia:
push %rsi #argv argumentua izan ezik
## egiaztatu komando lerroak bi argumentu dituela
cmp $2,%rdi
jeje imp_arg
## programa argumenturik gabe bakarrik badut: inprimatu pantailan
mov $SYSWRITE,%rax
mugitu $STDOUT,%rdi          #fd egongo den fitxategiaren erreferentzia da
```

```

idatzi
    mov $message, %rsi
    joango denaren #memoria helbidea
fitxategian idatzi
    mov $LON,%rdx idatzi
    #memoria buffer-aren tamaina

    syscall jmp
    #idazteko ordena nukleorako

irteera

imp_arg:
    pop %rsi ->
    #pilaren erakusleak gordetako %rsira seinalatzen du eta berreskuratzen dut
    argv -> argv[0]
    gehitu $ARGV1,
    #if lehen erakuslea seinalatzen du, 8 gehitzen baditut seinalatzen du
    %rsi bigarren erakuslea
    mov (%rsi), %rdi call puts
    #zeharkako bidez bigarren erakuslea dut

irteera:
    ret

.amaiera

```

## 29.8.4. Ariketak: sum\_linea\_com.s, maximum\_linea\_com.s

## 1. sum\_linea\_com.s

ÿ Sartu sum\_linea\_com.s programako datuak (bi gehigarriren batura) lerroaren bidez  
aginduak

```

### funtzioa: gehitu zifra bateko bi zenbaki oso.
### gehigarriak komando lerroaren bidez pasatzen dira
## X86-64 arkitekturan konpilazioa
## gcc -nostartfiles -g -o sum_input sum_input.s
## korrika 5 7
## x /x %rsp           ->3          argc:zenbakia
argumentuak
## x /a (char**)(%rsp+8) -> 0xfffffd0a4: 0xfffffd26e
## x /c *(char**)(%rsp+8) -> 0xfffffd26e: 47 '/'
## x /s *(char**)(%rsp+8) -> 0xfffffd26e:
"/home/candido/tutoriales/as_tutorial/algoritmos_x86-32/basicos/sum_input"
## p /s *(char**)(%rsp+8) -> 0xfffffd26e
"/home/candido/tutoriales/as_tutorial/algoritmos_x86-32/basicos/sum_input"
## x /s *(char**)(%rsp+16) -> 0xfffffd2b7: "5"
## x /s *(char**)(%rsp+24) -> 0xfffffd2b9: "7"

.atala .testua

.globl _hasi
_hasi:

## azalpen argibideak

irakurri 8(%rsp),%rax #eax-ek argv[1] duen pila helbidea dauka
katearen argumentuaren erakuslea dauka

```

```

        mov 8(%rsp),%rbx          #ebx-ek pila-edukia = helbidea du
katea
        xor %rcx,%rcx
        movb (%rbx),%cl          #ASCII karakterea

## katearen argumentu erakusleak
        mov 16(%rsp),             #eax pila edukia = helbidea du
katearen%rax. argv[2]
        mov 24(%rsp), katearen   #eax pila edukia = helbidea du
%rbx. argv[3]
        ## lortu katea zeharka
        ## Ascii zenbakiak balio bihurtu
        xor %rcx,%rcx
        xor %rdx,%rdx
        movb (%rax), %cl          # zeharkako katea sartzeko
argv[1] aipatzen du
        movb (%rbx), %dl-k        # zeharkako katea sartzeko
argv[1] aipatzen du
        azpi $0x30,%rcx
        azpitik $0x30,%rdx

        mov %rcx,%rsi
        mov %rdx,%rdi

        deitu batura

## irten
        mov %rax,%rdi
        mov $60, %rax syscall      #1 exit() syscall da

#### Bi balioen arteko batura kalkulatzen duen funtzioa
.mota batura, @funtzioa
.atala .testua
gain:
## Hitzaurrea
bultzatu %bp
mov %rsp,%rbp
azpian $8,%rsp              #memoria erreserba

##argudioen harrapaketa
mov %rdi,%rax mov %rsi,      #1. argumentua
%rcx ## gorputza           #2. argumentua

addl %ecx,%eax ## gorde    #
emaitza
## emaitza EAX-en dago
jauzi:
## epilogoa

```

```
mov %rbp,%rsp pop          # aurreko fotograma  
%rbp  
ret                         # berreskuratu itzulera helbidea
```

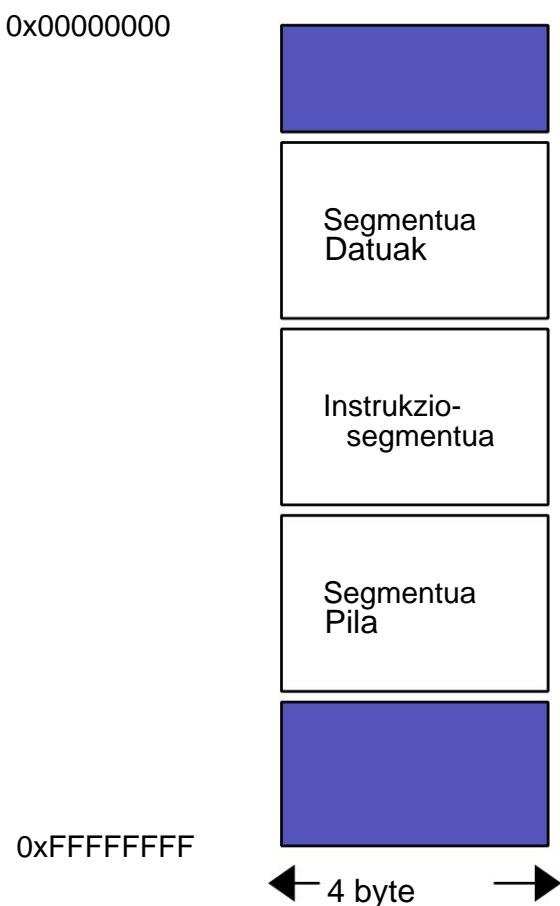
## 2. maximum\_linea\_com.s

ü Sartu maximum\_linea\_com.sa programaren datuak komando lerroaren bidez

# 30. kapitulua. Pila

## 30.1. Kontzeptua

- Pila:
  - ÿ Last Input First Output (LIFO) Datuen Egitura
- Kanpoko memoria
- Pilatzeko helbidea: Memoria-helbideen norabidean.
- Programa bat segmentutan egituratuta dago: Datuen segmentua, Instrukzio segmentua, Pila segmentua, ...
- ÿ Memoria nagusia segmentatua:



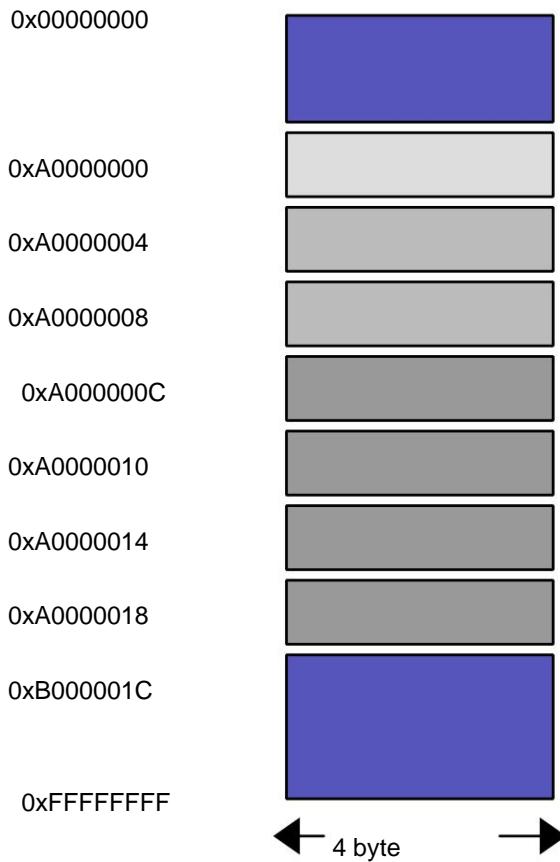
## 30.2. Zabalera

- Pila zabalera ÿ Hitzaren tamaina:
  - ÿ x86-64rako: 64 biteko zabalera
  - ÿ i386 arkitekturan 32 bitekoa da
- Pila memoria lerrokatzea ÿ hitzen tamainaren multiploak
  - ÿ x86-64-ren kasuan: 8 byteko (64 biteko) multiploak ÿ 0 eta amaitzen diren helbide hamaseitarrak

8an.

ÿ Pilatu beharreko datuak pilaren zabalera baino txikiagoak badira, luzatu egin beharko da. Luzapen mota datu motaren araberakoa izango da (zenbaki osoarekin, etab.)

- i386 Arkitektura Pila Segmentua:



### 30.3. Markoa: marko-erakuslea eta pila-erakuslea

- Markoa: pila atalaren zatiketa

ÿ Deitzen den funtziobakoitzak fotograma bat sortzen du

ÿ Marko aktiboaren mugak bi erakuslerekin adierazten dira:

ÿ beheko muga: marko-erakuslea, pilatutako lehen elementuaren kokapena adierazten du.

ÿ goiko muga: pila-erakuslea, pilatutako azken elementuaren kokapena adierazten du.

- Pila erakuslea (sp)

ÿ Markoaren TOP elementura seinalatzen duen erakuslea: azken elementua dagoen pilaren muga altua pilatuta.

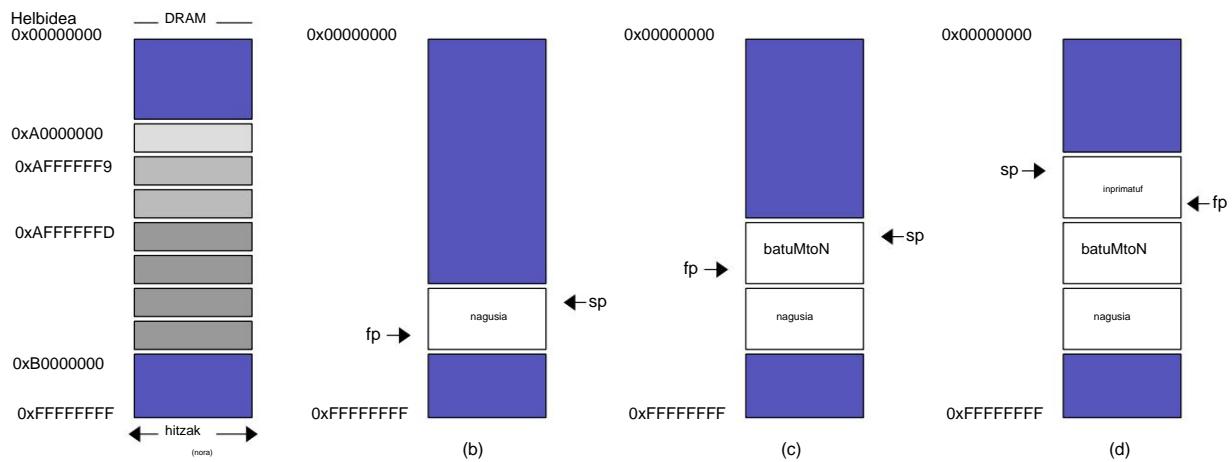
ÿ Intel x86-n RSP erregistroa da

- Fotograma erakuslea (fp)

ÿ Markoaren BEHEKO elementua seinalatzen duen erakuslea: lehenengo dagoen pilaren muga baxua. pilatutako elementua.

ÿ Intel x86-n RBP erregistroa da

- Pila atala (markoaren partizioa)

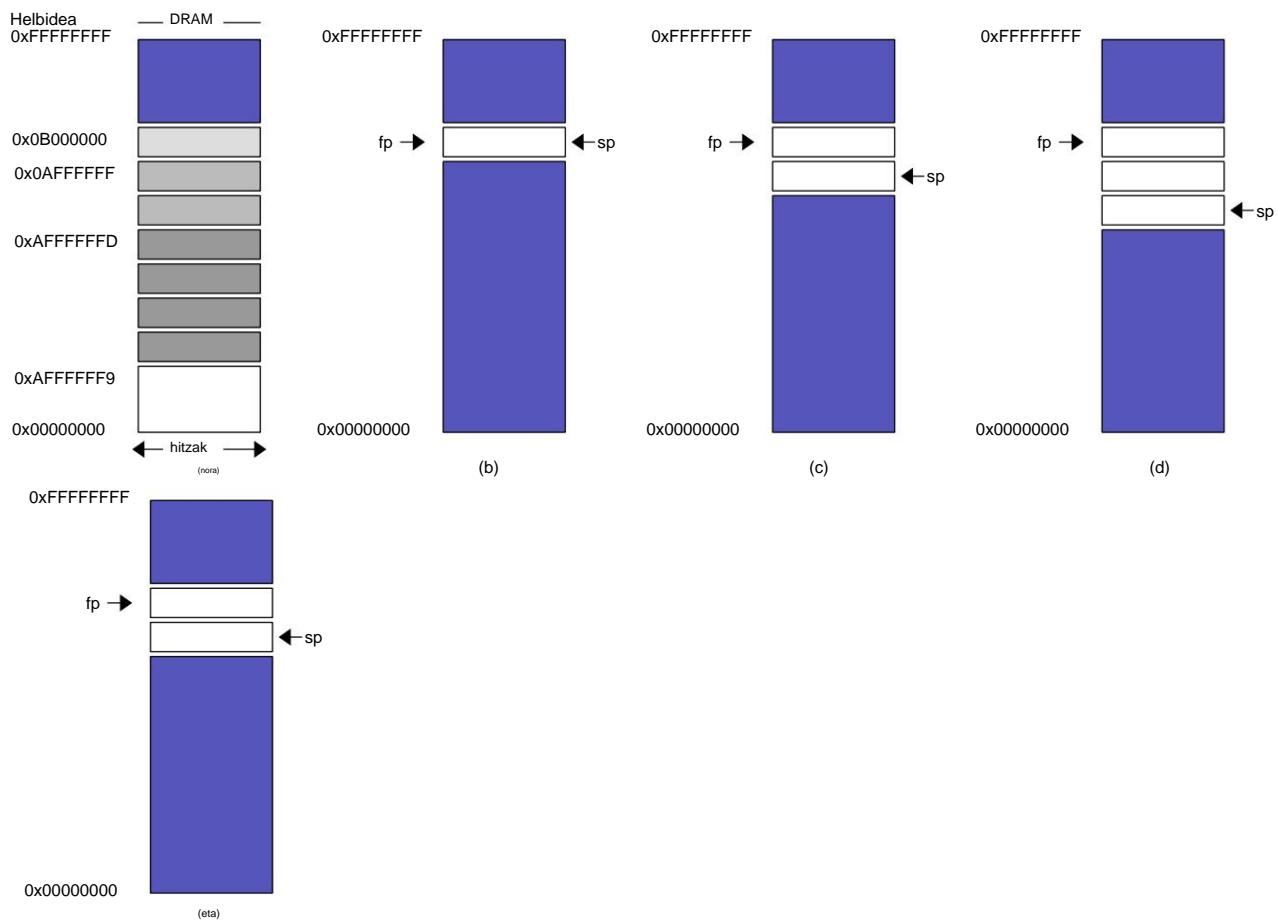


- (a) Pila ez dago osatuta
- (b) dei nagusira: marko nagusia eratzen da. Markoa hazi eta txikiagotu egiten da pilatu eta atera ahala
- (c) deia nagusitik sumMtoNra: sumMtoN markoa nagusitik aurrekoaren gainean eratzen da: erakusle berriak FP eta SP.
- (d) sumMtoN-tik printf-ra deitu: printf markoa aurrekoaren gainean eratzen da sumMtoN-tik: erakusle berriak FP eta SP.

ÿ Pila a-ren deia nagusitik sumMtoNra eratzen da: erakusle berriak FP eta SP.

#### 30.4. Push-Pop muntatzailearen jarraibideak

- Push-Pop Instrukzioa: pilaketa-tira
  - ÿ Push Op\_source
    - ÿ Eragiketa: datuak txertatu.
    - ÿ Helmuga-eragigaia: pila.
    - ÿ Pila-erakuslea hitz batez TXINTZEN DA. SP ÿ SP-1\*WordSize eta gero sartu iturburuko eragigaia helmugan.
  - ÿ Pop Op\_dest
    - ÿ Eragiketa: datuak atera.
    - ÿ Iturburu-eragilea: pilatutako azken objektua.
    - ÿ Lehenik pila-erakusleak erreferentziatutako objektua ateratzen da. Ondoren pila erakuslea HANDITZEA hitz bakarrean. SP ÿ SP+1\*WordSize



- (a) Pila ez dago osatuta
- (b) Pila pila-erakusleak hasieratuz osatzen da: fotograma-erakuslea (fp) eta pila-erakuslea (sp)
- (c) Push exekuzioa
- (d) Push exekuzioa
- (e) Pop emanaldia

#### 30.4.1. Dei habia egitea

- GUZTIAK

## 31. kapitulua. Batzar-hizkuntzako programak: proposamenak

### 31.1. Maila ertaina

#### 31.1.1. BMP formatua

##### Zirkulu

- Erdiko puntuaren algoritmoa

x: abszisa-ardatza -> positiboa eskuinera y:  
 ordenatuen ardatza -> positiboa gorako zirkulu-  
 ekuazioa: zentroa (0,0) eta lehen zortziduna ( $x_{\text{positivo}}$ ,  $y_{\text{positivo}}$ ,  $y < x$ ). Zortzitzaleen ordenak erlojuaren  
 noranzkoan doaz ekuazio diskretua yn e xn zenbaki osoak: xn eta yn lehen zortzidunera mugatuta:  
 lehen puntu:  $x_1=r$   $y_1=0$  bigarren puntu:  $y_2=1 \rightarrow y_2=y_1+1$   
 handitzen da

:  $x_2-k$  bi alternatiba baino ez ditu: aurreko  $x_1$ -aren berdina edo  
 unitate bat handitu  
 : lortu  $x_2$  zirkulu-ekuazioaren arabera ->  $x^{2-n+1} = x^{2-n}$   
 $-2^y n \sim -1$

$2^y n \sim -1 = x^{2-n} - 2^y n \sim -1$ : zenbaki errealekin ebatzi daiteke baina interesgarria da zenbaki osoekin aritmetika egitea eta  
 horrela mihiadura hizkuntzan programatu ahal izatea.

ÿ Zenbaki osoetan oinarritutako aritmetika duen aldaera ÿ Bresenham-en lerro-algoritmoa: aritmetikarekin garatua  
 osorik

- Bresenham zirkulua:

xn eta yn zenbaki osoen  
 koordenatuak lehen zortzikote jatorriko  
 zirkuluan (0,0)  $x_{\text{start}}=r$   
 $y_{\text{start}}=0$

$E_{\text{hasiera}}=3-2r$

elkarrekintza bakoitzean:  
 $E_{\text{korrontea}} < 0 \rightarrow E_{\text{next}}=E_{\text{cur}} + 2*(3+2^y c) ; x_{\text{next}}=x_{\text{unelea}} ; y_{\text{next}}=y_{\text{current}}+1$   
 $E_{\text{korrontea}} > 0 \rightarrow E_{\text{next}}=E_{\text{cur}} + 2*(5+2^y c-2^x c) ; x_{\text{next}}=x_{\text{current}}-1 ; y_{\text{next}}=y_{\text{current}}+1$

ondoko elkarrekintza:

$E_{\text{current}}=E_{\text{next}} ; y_{\text{current}}=y_{\text{next}} ; x_{\text{current}}=x_{\text{next}}$

- Iturburu kodea C-n

```
/*
```

Programa: pixel\_circulo.c

Deskribapena: Bresenham algoritmoa. [https://en.wikipedia.org/wiki/Midpoint\\_circle\\_algorithm](https://en.wikipedia.org/wiki/Midpoint_circle_algorithm) OHARRA: ekuazio matematikoak

OKTANT baterako garatzen dira.

zentroa ( $y=0, x=0$ ) (beheko ezkerreko izkina). Ardatz bertikala  $\rightarrow x$ , Ardatz horizontala  $\rightarrow y$  OHARRA: horregatik ekuazio matematikoek octantea hasten dute.

$(xc=r, yc=0)$  (eskuinean behean) eta octantea gora doan margotzen da ( $yc+1$ ) eta ezkerrerantz doa ( $xc$  ez da aldatzen edo  $xc=xc-1$ )  $x=0$  zentrorantz.

OHARRA: zortzidunen ordena bi modutara deitu daiteke. Bertan

artikulua lehen octantea  $x_-, y$  positiboa eta  $y < x$  positiboa dira eta hurrengo octanteak erlojuaren orratzen norantzan doaz. Octanteak erloju baten arabera ordenatu ditut, non lehen octantea 12:00etatik bi orratzetara doan 1 eta bi artean, erlojuaren orratzen noranzkoak zerrendatz. Beraz, lehen octantea 12:00etan hasten da lehen puntueta ez 3:00etan ekuazio matematikoetan bezala, nik egiten dudana  $x-a$  mapatzea da, hau da, abszisa matematikoa, ez bmp-ko  $j$  zutabe gisa, i errenkada gisa baizik. , hau da,  $x>i$   $y>j$ . Zutabeetarako CHANGE errenkadak lehen zortzikoa 12:00etan hasten da.

OHARRA: BPM formatuak (0,0) du (beheko ezkerreko izkinan). Ardatz bertikala  $\rightarrow i$  errenkadak, Ardatz horizontala  $\rightarrow$  zutabeak  $j$

pixels\_generator(): BMP formatuko irudi batean RGB koloreak definitzeko kanpoko funtzioa

Programa nagusia main(): bmp\_funcion\_arguDimension.c

gcc -m32 -g -c -o pixel\_circulo.o pixel\_circulo.c -c aukerarekin objektu-modulua modulu exekutagarria sortu gabe sortzen da

KONTUZ: MATRIZEAREN DIMENTSIOA deklaratzeko da bai mod\_funtziorako deia duen fitxategian bai funtzioaren definizioa duen fitxategian. Saiatu naiz DIMENSIONaren balio desberdinak jartzen bi fitxategietan (deia eta definizioa) eta ez dago konpilazio edo exekuzio akatsik baina exekuzioaren emaitza aldatzen da.

Lehenengo oktantea erdigunea (0,0) jarrita kalkulatzen da eta gero itzuli egiten da.

Bigarren octantea eta zortzigarren arte lehenengo zortzidunaren simetria zuzena eta translazioa eginez kalkulatzen dira

\*/

**#define 512 DIMENTSIOA // OHARRA: derrigorrezkoa da dimentsio anitzeko matrizeek gutxienez dimentsio GUZTIAK izatea definitutako bat izan ezik**

//

**typedef unsigned char BYTE;**

```
// pixel bakoitzaren motaren definizioa. Pixel bakoitzak hiru byte ditu. Byte bakoitza BYTE  
motakoa da.  
typedef struct {  
    BYTE     b;  
    BYTE     g;  
    BYTE     r;  
} RGB_datuak;  
  
void paint(int i, int j, BYTE urdina, BYTE berdea, BYTE gorria, RGB_data reg_mem  
[][DIMENSION]);  
  
void pixels_generator_circle (sinatutako int origin_x, sinatu gabeko int origin_y, sinatu  
gabeko int erradioa, sinatu gabeko int proportzioa, sinatu gabeko int dimentsioa, RGB_data  
reg_mem[][]DIMENSION])  
{  
  
int Ecur,Enext,xn,xc,yn,yc,i,j;  
  
// ekuazio matematikoak zentroa (0,0) duen zirkulu baterako garatzen dira  
Ecur=3-2*erradioa;  
xc=erradioa;  
yc=0;  
  
bitartean (yc<xc) { // lehen zortzidun (x,y) -> (x eta y positiboa y<x-rekin) Ardatz bertikala  
-> x Ardatz horizontala -> y => (abzisa, ordenatua)...(x ,y )  
bada (Ecur < 0) {  
    xn=xc;  
    yn=yc+1;  
    Enext = Ecur + 2*(3+2*yc);  
  
}  
bestela  
{ xn=xc-1;  
yn=yc+1;  
Enext = Ecur + 2*(5+2*yc-2*xc); //printf  
("i=%dx=%dy=%d Ecur=%d Enext=%d \n",i,x[i],y[i],Ecur,Enext);  
  
}  
  
// zentroko itzulpena (0,0) (x_jatorria,y_jatorria) i=jatorria_x+xc; //  
x ROW gisa j=jatorria_y+yc; // j-a  
ZUTABIDE gisa -> gero j abzisa da -> (j,i). Espazio matematikoaren aldaketa hau (y,x) da  
  
// lehen zortzidun koordenatuak (y,x)  
paint(i,j,0,0,0xFF,reg_mem);  
  
// SIMETRIA bigarren zortzidun lehenarekiko -> (x,y)
```

```

i=jatorria_y+yc;
j=jatorria_x+xc;
paint(i,j,0,0xFF,0,reg_mem);

// SIMETRIA hirugarren zortzidun lehenengoarekiko -> (x,-y) -> ordenatua
negatiboa da ->
i i=jatorria_y-yc;
j=jatorria_x+xc;
paint(i,j,0xFF,0,0,reg_mem);

// SIMETRIA laugarren zortzikoa lehenengoarekiko -> (y,-x) -> ordenatua
negatiboa da ->
i i=jatorria_x-xc;
j=jatorria_y+yc;
paint(i,j,0xFF,0xFF,0XFF,reg_mem);

// SIMETRIA bosgarren zortzidun lehen -> (-y,-x) i=jatorria_x-xc;
j=jatorria_y-yc;

paint(i,j,0,0,0XFF,reg_mem);

// SIMETRIA seigarren zortzikoa lehenengo -> (-x,-y)
i=jatorria_y-yc;
j=jatorria_x-xc;
paint(i,j,0,0xFF,0,reg_mem);

// SIMETRIA zazpigaren zortzidun lehen -> (-x,y) i=jatorria_y+yc;
j=jatorria_x-xc;

paint(i,j,0xFF,0,0,reg_mem);

// SIMETRIA zortzigarren zortzigarren lehen -> (-y,x)
i=jatorria_x+xc;
j=jatorria_y-yc;
paint(i,j,0xFF,0xFF,0XFF,reg_mem);

Ecur=Enext;
yc=yn;
xc=xn;

}

}

void paint(int i, int j, BYTE urdina, BYTE berdea, BYTE gorria, RGB_data reg_mem
 $\text{[][][DIMENTSIOA]} \{$ 

    // red intensity reg_mem[i]
     $j].r = \text{red};$  // intentsitate
    berdea

```

```

reg_mem[i][j].g = berdea; // blue
intensity reg_mem[i][j].b = blue;

}

```

## Lerroa

- Bresenham-en\_lerroaren\_algoritmoa

• x: abszisen ardatza • eskuinera positiboa  
 • y: ordenatuen ardatza • positiboa beherantz  
 • (0,0): goiko, ezkerreko izkina  
 • (x0,y0) eta (x1,y1) lerroaren amaierak dira  
 • zuzenak malda < 1 du  
 • Zuzenaren ekuazioa • f(x,y)=0  
 • f(x,y) = A\*X+B\*y+C = 0  
 • Iturburu kodea C-n

```

#define DIMENSION 1024
typedef unsigned char BYTE; typedef
struct {

    BYTE      b;
    BYTE      g;
    BYTE      r;
} RGB_datuak;

void paint(int i, int j, BYTE urdina, BYTE berdea, BYTE gorria, RGB_data reg_mem
[[[DIMENTSIOA]]];

void pixels_generator_linea (unsigned int x1, unsigned int y1, unsigned int x2, unsigned int y2, unsigned int
dimentsioa, RGB_data reg_mem[[DIMENSION]]) {

    int i1,i2,dx,dy,d,x,y,xmax; // ekuazio
    matematikoak malda zuzen baterako garatzen dira
    1 baino
    TXIKIA // (x1,y1) eta (x2,y2) lerroaren amaierak dira LEHEN KOADRANTEA

    dx=x2-x1;
    dy = y2-y1;
    i1=2*dy;
    i2=2*(dy-dx); d=i1-
    dx;

    bada (dx<0)
    { x=x2;
    y=y2;
    xmax=x1;

```

```

} bestela
{ x=x1;
y=y1;
xmax=x2; }
bitartean (x < xmax)
{ baldin
(d<0) d=d+i1; bestela
{ d=d+i2; y=y+1; }
x=x+1; // x zutabeak
-> j // y
errenkadak -> iÿ

paint(y,x,0,0,0xFF,reg_mem); }}

void paint(int i, int j, BYTE urdina, BYTE berdea, BYTE gorria, RGB_data reg_mem
[[DIMENTSIOA)]{

// red intensity reg_mem[i][j].r = red; //
intentsitate berdea reg_mem[i][j].g =
berdea; // blue intensity reg_mem[i][j].b =
blue;

}

```

- Kodea 1 baino txikiagoa eta handiagoa den eskaeretarako

```

/*
Programa: pixel_linea.c Deskribapena:
Bresenham-en algoritmoa: https://en.wikipedia.org/wiki/Bresenham-line\_algorithm -> kontzeptua
: https://www.javatpoint.com/computer-graphics
bresenhams-line-algorithm -> algoritmoa
:https://iq.opengenus.org/bresenham-line-drawing-algorithm/ ->
malda >1
ardatzak: BMP formatuko irudi batean Programa nagusia main(): bmp_funcion_arguDimension.c

```

```
gcc -m32 -g -c -o pixel_linea.o pixel_linea.c -c aukerarekin  
objektu-modulua modulu exekutagarria sortu gabe sortzen da
```

KONTUZ: MATRIZAREN DIMENTSIOA espedientean deklaratzentz da deiarekin batera  
funcion\_mod funtziaren definizioa duen fitxategian bezala. Bi fitxategietan DIMENSIONari balio desberdinak  
jartzen saiatu naiz (deia eta definizioa) eta ez dago konpilazio edo exekuzio errorerik baina exekuzioaren emaitza  
aldatzen da.

```
*/
```

```
#define 1024 DIMENTSIOA // OHARRA: derrigorrezkoa da dimentsio anitzeko matrizeek gutxienez dimentsio GUZTIAK  
definituta izatea, bat izan ezik
```

```
//
```

```
typedef unsigned char BYTE;
```

```
// pixel bakoitzaren motaren definizioa. Pixel bakoitzak hiru byte ditu. Byte bakoitza BYTE motakoa da.
```

```
typedef struct {
```

```
    BYTE      b;  
    BYTE      g;  
    BYTE      r;
```

```
} RGB_datuak;
```

```
void paint(int i, int j, BYTE urdina, BYTE berdea, BYTE gorria, RGB_data reg_mem[][][  
DIMENTSIOA]);
```

```
void pixels_generator_linea (unsigned int x1, unsigned int y1, unsigned int x2, unsigned int y2, unsigned int  
dimentsioa, RGB_data reg_mem[][DIMENSION]) {
```

```
int i1,i2,i3,dx,dy,d,x,y,xmax; float p; // //  
(x1,y1)  
eta (x2,y2) lerroaren amaierak dira LEHENENGO KUADRANTEan
```

```
dx=x2-x1;  
dy = y2-y1;  
i1=2*dy;  
i2=2*(dy-dx);  
i3=2*dx;  
p=(float)dy/dx;//malda
```

```
// hasierako amaiera (ezkerrean) (x1,y1) edo (x2,y2) izan daiteke (dx<0) { x=x2;  
y=y2; xmax=x1;
```

```

}

ÿ bestela {
x=x1;
y=y1;
xmax=x2;
}

ÿ baldin (p<1,0) { // 1 baino malda txikiagoa bada
    d=2*dy-dx;
    bitartean (x < xmax){
        bada (d<0)
            d=d+i1;
        else { // 1 baino malda handiagoa
            d=d+i2;
            y=y+1;
        }
        x=x+1;
        paint(y,x,0,0,0xFF,reg_mem);
    }
}
ÿ bestela {
ÿ // p>1
    d=2*dx-dy;
    bitartean (x < xmax){
        bada (d<0)
            d=d+i3;
        bestela {
            d=d-i2;
            x=x+1;
        }
        y=y+1;
        paint(y,x,0,0,0xFF,reg_mem);
    }
}
// x zutabeak -> j
// eta errenkadak -> i

}

void paint(int i, int j, BYTE urdina, BYTE berdea, BYTE gorria, RGB_data reg_mem[][],  

DIMENTSIOA]){
    // intentsitate gorria
    reg_mem[i][j].r = sare;
    ÿ // intentsitate berdea
    reg_mem[i][j].g = berdea;
    ÿ // intentsitate urdina
    reg_mem[i][j].b = urdina;
}

```

}

## 32. kapitula. Eranskina: CPU Unitatea

### 32.1. Mikroarkitektura modernoak (2000tik aurrera)

- [Medimu Bloga: mitterandekole](#)

- adar iragarpena: PUZak adar-argibideak aurreikusteko eta adarraren norabidea aurreikusteko gaitasuna du. Modu honetan, hutsegiteak ez dira gertatzen "pipeline" exekuzioan eta posible da "aurreikusi" zein instrukzio exekutatu behar diren beren instrukzio-ziklo elkartuak hasi aurretik.
- Exekuzio espekulatiboa: instrukzio-zikloa behin betiko exekutatuko diren ala ez dakien argibideekin hastea da.
- Frontend-a: Kontrol Unitateko Front-end blokea zein instrukzio exekutatu behar diren iragartzean datza, Cache memoriatik atera eta mikroeragiketan deskodetzen, txanda heldu baino lehen ÿ Pre-Fetch. Aurrez harrapatzen dira eta Front-End bufferean gordetzen dira. Modu honetan, instrukzio-zikloa hasten denean, aurretik harrapatutako instrukzioak dagoeneko deskodetuta daude, hau da, instrukzio-zikloaren lehen faseak eginda dituzte.

### 32.2. Intel

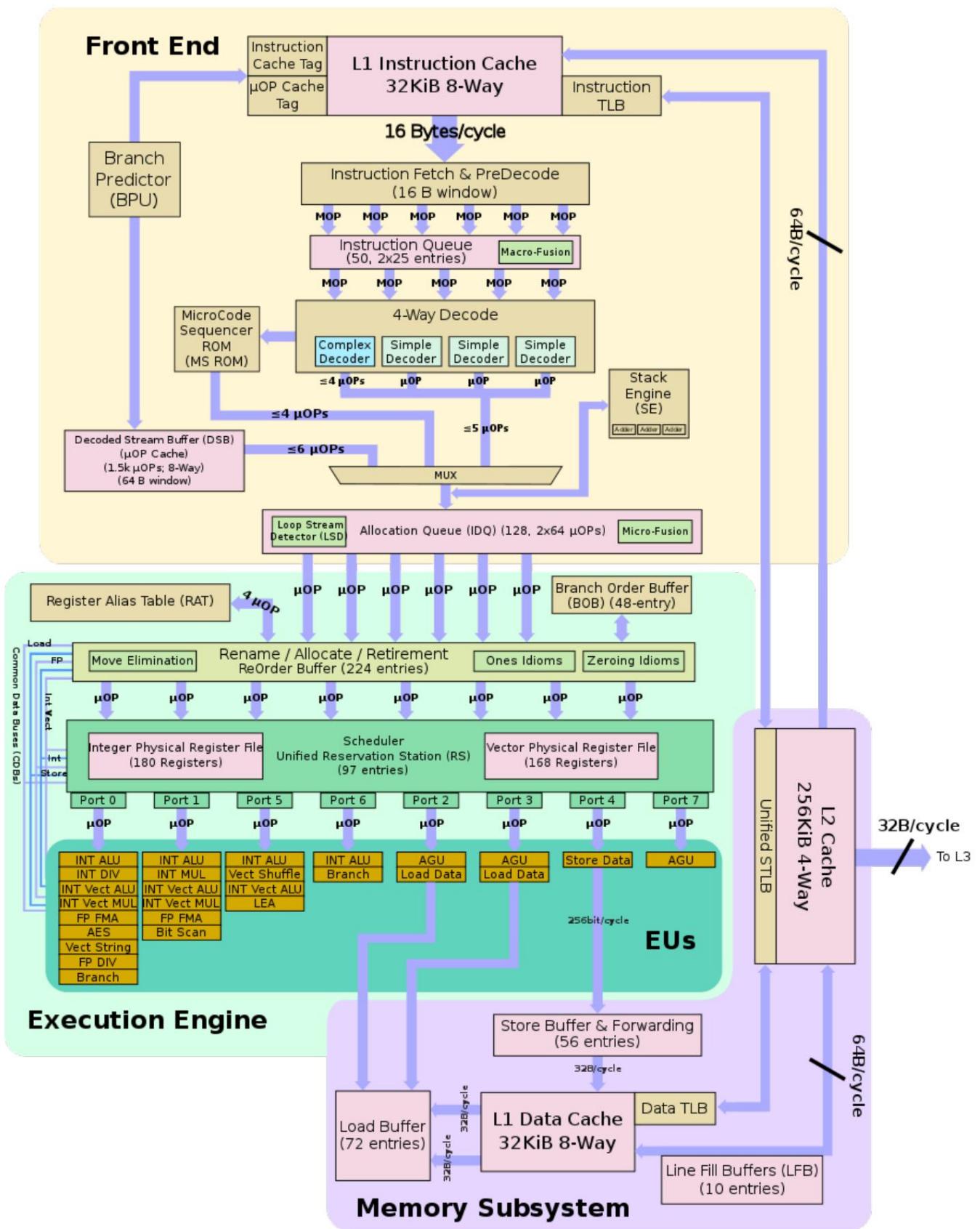
- [Youtube Coffee Lake \(2017\), Ice Lake \(2019\)](#)

### 32.3. amd

- : [Zen4](#)

### 32.4. Skylake-U

- Intel prozesadorea: Core i5-6300U



112. Irudia Skylake mikroarkitektura

- Front End ȳ Harrapaketa eta deskodetze fazeak aurreratzen dituen UC blokea.
- Ordenantzaz kanpoko exekuzioaren (OoO) eta mikrooperazioaren kontzeptua (uOP, ez nahastu behar mikrokodearen kontrol unitatea):

Ordenaz kanpoko exekuzioa duten mikroprozesadoreek instrukzio guztiak mikroeragiketa bihurtzen ari dira - μops edo uops laburtuta. ADD bezalako instrukzio simple bat

EAX, EBX-k pop bakarra sortzen du, eta ADD EAX,[MEM1] bezalako instrukzio batek, berriz, bi sor ditzake: bat memoriatik irakurtzeko aldi baterako erregistro batean (izenik gabekoa) eta aldi baterako erregistroaren edukia EAXra gehitzeko. ADD [MEM1], EAX instrukzioak hiru pops sor ditzake: bat memoriatik irakurtzeko, bat gehitzeko eta bestea emaitza memorian idazteko. Honen abantaila da pop-ak ordenaz kanpo exekutatu daitezkeela.

Adibidea: ; Adibidea 2.1. Ordez kanpo prozesatzen

```
mov eax, [mem1] imul  
eax, 5 add eax,  
[mem2] mov [mem3],  
eax Hemen, ADD EAX,
```

[MEM2] instrukzioa bi popsetan banatzen da. Honen abantaila da mikroprozesadoreak [MEM2]-ren balioa lor dezakeela biderketa egiten ari den aldi berean. Datuetako bat ere ez badago cachean, orduan mikroprozesadorea [MEM2] lortzen hasiko da [MEM1] lortzen hasi eta berehala, eta biderketa hasi baino askoz lehenago.

- Behatu bi eragiketa bide guztiz desberdinak: MOP eta uOP.
  - ÿ MOP: makro instrukzioak: CISC motako Intel ISA instrukzioak dira
  - ÿ uOP: CISC motako ISA instrukzioak RISC motako argibide simpleagoetan banatzen dira.

## 32.5. ARM Cortex-A76/Cortex-A55

### 32.5.1. Sarrera

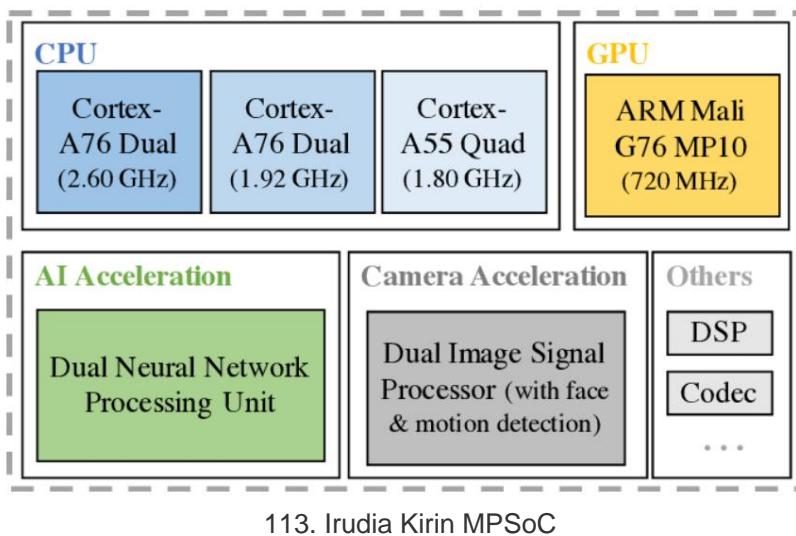
- [https://en.wikipedia.org/wiki/Arm\\_\(empresa\)](https://en.wikipedia.org/wiki/Arm_(empresa))
- ARM : Ezkurra (ezkurra) RISC Makina ÿ RISC Makina Aurreratuak
- RISC arkitektura.

### 32.5.2. Huawei P30 Pro telefonoa

- Huawei P30 Pro VOG-L29 telefono modeloak MultiProcessor System On Chip (MPSoC) barne hartzen du. Huawei **Kirin 980**. SoC-ak nukleoak eta chipset-a integratzen ditu.

2018ko abuztuaren

31an Kirin 980 HiSilicon-ek diseinatutako eta 2018aren amaieran aurkeztutako 64 biteko errendimendu handiko ARM LTE SoC mugikor bat da. TSMC-ren 7 nm-ko prozesuan egina, 980-k lau Cortex-A76 nukleo handi ditu 2,6 GHz-ra funtzionatzen dutenak. 1,8 GHz-ra arte funtzionatzen duten lau Cortex A55 nukleo txikiiek. SoC honek 1,4 Gbps-ko deskarga onartzen duen LTE modem bat du (Cat21), ARM Mali-G76 bat dauka eta LPDDR4X-4266 memoria onartzen du.



### 32.5.3. ISA Cortex-A76/Cortex-A55

- ISA ARMv8.2-A arkitektura

64 biteko

ikusi ARM konpainiaren ISA ARMv8.2-A eskuliburua

- Mikroarkitektura

ÿ [https://en.wikipedia.org/wiki/Comparison\\_of\\_Armv8-A\\_processors](https://en.wikipedia.org/wiki/Comparison_of_Armv8-A_processors)

### 32.5.4. Nukleo anitzekoa

- 8 nukleo
- handia.TXIKIA arkitektura heterogeneoa: 2 nukleo mota > 4+4
  - ÿ nukleo handiak: errendimendu handia eta kontsumo moderatua ÿ Cortex-A76
  - ÿ LITTLEcores: errendimendu moderatua eta potentzia baxua ÿ Cortex-A55

### 32.5.5. Cortex-A76 Mikroarkitektura

- [https://en.wikipedia.org/wiki/Comparison\\_of\\_Armv8-A\\_processors](https://en.wikipedia.org/wiki/Comparison_of_Armv8-A_processors)
- [https://en.wikipedia.org/wiki/ARM\\_Cortex-A76](https://en.wikipedia.org/wiki/ARM_Cortex-A76)

Cortex-A76 frontend-a 4 zabaleko deskodeketa ordenaz kanpoko diseinu supereskalarra da. Ziklo bakoitzeko 4 argibide lor ditzake. Eta [argitu behar da] izena aldatu eta bidali 4 fregona eta 8 pops ziklo bakoitzeko. Ordenaz kanpoko leihoa tamaina 128 sarrerakoa da. Backend-a 8 exekuzio ataka da, 13 etapako kanalizazio-sakonera eta 11 etapako exekuzio latentzia dutenak.

ÿ supereskalarra - 4 bideko datu-bidea

ÿ kanalizazioa: 13 etapa

### 32.5.6. Cortex-A55 Mikroarkitektura

- [https://en.wikipedia.org/wiki/ARM\\_Cortex-A55](https://en.wikipedia.org/wiki/ARM_Cortex-A55)

- [https://en.wikipedia.org/wiki/Comparison\\_of\\_Armv8-A\\_processors](https://en.wikipedia.org/wiki/Comparison_of_Armv8-A_processors)

ÿ supereskalarra: bi norabideko datu-bidea

ÿ kanalizazioa: 8 etapa

### 33. kapitula. Eranskina: Memoria Unitatea

#### DRAM

##### 33.1. Memoria nagusia: DRAM

###### 33.1.1. Irakurketa/idazketa eragiketa denbora

20. taula. SDRAM pin esleipena

Seinaleak	Deskribapena
A0-A29	Gelaxka helbidearen sarrera
CLK	Autobusaren erlojuaren sarrera memoria
/CS	Txip aukeraketa
/RAS	Errenkadaren norabidea hautatzea
/CAS	Helbidea hautatzea zutabea
/GU	Idazteko gaitza
DQ0-DQ7	Datu sarrera/irteera

Barra / seinale ukatua esan nahi du: logika negatiboa: maila baxuan aktibatzen da (L).

Wikipediak hartutako adibidea:

21. taula. DRAM asinkronoa: Denbora

	"50 ns" "60 ns" Deskribapena	
tRC	84 ns	104 ns Ausazko irakurketa edo idazketa zikloaren denbora (/RAS ziklo oso batetik beste bat)
tRAC	50 ns	60 ns Sarbide-denbora: /RAS baxua datuak baliozko ateratzeko
tRCD	11 ns	14 ns /RAS low to /CAS low time
ondoren	50 ns	60 ns /RAS pultsuaren zabalera (gutxieneko /RAS denbora baxua)
tRP	30 ns	40 ns /RAS aurrekargatzeko denbora (gutxieneko /RAS denbora altua)
tPC	20 ns	25 ns Orri-modua irakurtzeko edo idazteko ziklo-denbora (/CAS to /CAS)
tAA	25 ns	30 ns Sarbide-denbora: zutabe-helbidea baliozko datuak ateratzeko balio du (barne helbidea konfiguratzeko ordua baino lehen /CAS baxua)
tCAC	13 ns	15 ns Sarbide-denbora: /CAS baxua baliozko datuak ateratzeko
tCAS	8 ns	10 ns /CAS pultsu-zabalera baxua gutxienez

- Sinkronismoa

ŷ DRAM : asinkronoa: aldaketei ahalik eta azkarren erantzuten die

ŷ SDRAM-ek memoria asinkronoaren interfazea nabarmen berrikusten du, erloju bat (eta erloju bat) gehituz. gaitu) lerroa. Beste seinale guztiek erlojuaren goranzko ertzean jasotzen dira. Ez du hain azkar erantzuten ahalik eta, baina goranzko ertzaren zain dago.

- EZ dugu DRAM memoria asinkronorik ikusten, kontzeptua soilik.

## 22. Taula. DRAM sinkronoa: Denbora

	PC-3200 (DDR-400) PC2-6400 (DDR2-800) PC3-12800 (DDR3-1600)										Deskribapena		
	Tipikoa		azkar		Tipikoa		azkar		Tipikoa		azkar		
	zikloa da	denbora-zikloa da	denbora-zikloa da	denbora-zikloa da	denbora-zikloa da	denbora-zikloa da	denbora-zikloa da	denbora					
tCL	3 15ns	2 10ns	5 12.5		ns	4 10ns	9 11,2	5ns	8	10 ns	/CAS baxua da baliozko datuak atera (tCAC-ren baliokidea)		
tRCD	4 20ns	2 10ns	5 12.5		ns	4 10ns	9 11.2		5ns	8 10ns	/RAS baxutik /CAS baxutik denbora		
tRP	4 20ns	2 10ns	5 12.5		ns	4 10ns	9 11.2		5ns	8 10ns	/ RAS aurrekarga-denbora (gutxieneko aurrekarga denbora aktiborako)		
8 ondoren	40ns	5 25ns	16 40ns	12 30ns	27 33,7	5ns				24 30 ns	Errenkadako denbora aktiboa (gutxieneko aktiboa aurrez kargatzeko denbora)		

- Memoria sinkronoa deskribatzean, denbora-tartea marratxozen bereizita memoria-busaren erloju-zikloen arabera deskribatzen da. Zenbaki hauek tCL-tRCD-tRP-tRAS adierazten dute DRAM erlojuaren ziklo-denboraren multiploetan

## 33.1.2. Latentzia-denborak

- Testuinguru honetan latentzia atzerapenaren sinonimo da. Memoria Latentziaren kontzeptu ezberdina, hau da, denbora sarbidearena.
- tCL :Cas Latentzia . Atzerapena CAS seinalistik datuak I/O bufferean lortu arte
- tRCD :Ras Cas Atzerapena. RAS seinalistik CAS seinalerako atzerapena
- tRP :Ras Aurrekarga. Aurrekarga eta aktibazio arteko gutxieneko atzerapena
- ondoren :Row Active Time. Ilara aktibatzeko eta ilararen hasierarako igaro behar den gutxieneko denbora aurrekarga.
- Sarbidea: tCL+tRCD: bus helbidea baliozkotzen denetik datuak I/O bufferean lortu arte. erreferentziatuta.
- Autobusaren Tzikloa (irakurketa edo idazketa): tCL+tRCD+tRP+tBURST edo tCL+tRCD+tRAS (bat transferitu badugu). diputatuari agindua):
  - ÿ Sarbidea gehi CPUna transferitzeko atzerapena. Bi irakurketa edo bi idazketaren arteko denbora jarraian,
- tBURST: hitz bloke bat transferitzeko behar den denbora: BURST: ez da transferentziarik egiten 1 byte: 2,4,8,16, ..
- MP modulu programagarria da, tCL-tRCD-tRP-tRAS denborak alda ditzakegu eta baita ere leherketaren luzera (leherketa edo blokea)
- MP moduluak normalean tCL-tRCD-tRP-tRAS sekuentzia adierazten du erloju-zikloen balio tipikoekin

## 33.1.3. Adibidea PC2-6400 (DDR2-800) 5-5-5-16

- PC2-6400 modulu (DDR2-800) 5-5-5-16
- PC2 : Bigarren belaunaldiko SDRAM ÿ Double\_Data\_Rate x2
- 6400 MB/s-ko banda-zabalera edo abiadura

- 800MHz sistema eraginkorra bus erloju zikloa

↳ Hitz bakoitza 800MHz-ko ziklo batean transferitzen da.

↳ Memoria Bus Erlojuaren Zikloa 400MHz

↳ Erlojuaren ziklo-denbora =  $1/400\text{MHz} = 2,5\text{ns}$

- 5-5-5-16 tCL-tRCD-tRP-tRAS ↳ 12.5ns denboraren erloju-zikloak dira (400MHz ↳ 2.5ns).

12,5ns-12,5ns-40ns

23. taula. Glosarioa

denbora Deskribapena		denbora Deskribapena	
tCL	CAS latentzia	tRRD	RAStik RASrako atzerapena
tCR	Agindu-tasa	tRTP	Irakurri aurrez kargatzeko atzerapena
tPTP	aldez aurretik kargatu atzerapena	tRTR	Irakurri irakurtzeko atzerapena
ondoren	RAS denbora aktiboa	tRTW	Irakurri idazteko atzerapena
tRCD	RAStik CASrako atzerapena	tWR	Idatzi berreskuratzeko denbora
tREF	Freskatzeko epea	tWTP	Idatzi aurrez kargatzeko atzerapena
tRFC	Errenkadak freskatzeko zikloaren denbora	tWTR	Idatzi irakurtzeko atzerapena

- memoria-denborizazioa

- Testu liburuko 5.13 irudia: SDRAM irakurketa (leherren luzera=4, CL=2)

### 33.2. DRAM memoriaren antolaketa aurreratua

#### 33.2.1. DRAM asinkronoa

- Memoria asinkronoan egindako ekintzak kontrolagailuaren eta memoriaren arteko elkarriketaren araberakoak dira.
- Memoria sinkronikoak erlojuaren goranzko edo beherako ertzean ekintzak hasi eta amaitzen ditu, erraztuz. zirkuitu digital elektronikoaren diseinua eta autobusean abiadura handiagoak ahalbidetzea.
- RAM asinkronoa den arren, seinaleak normalean erlojudun memoria kontrolagailu batek sortzen ditu, eta horrek kontrolagailuaren erloju-zikloaren multiploetara mugatzen du haien denbora.
- DRAM

#### 33.2.2. SDRAM (DRAM sinkronikoa)

##### Erreferentziak

- SDRAM
- DDR\_SDRAM
- DDR2\_SDRAM
- DDR3\_SDRAM
- jedec estandarra
- [http://www.freescale.com/webapp/sps/site/overview.jsp?code=784\\_LPBB\\_DDR](http://www.freescale.com/webapp/sps/site/overview.jsp?code=784_LPBB_DDR)
- memoria-denborizazioa
- [http://en.wikipedia.org/wiki/Memory\\_timings](http://en.wikipedia.org/wiki/Memory_timings)

- [http://en.wikipedia.org/wiki/SDRAM\\_latency](http://en.wikipedia.org/wiki/SDRAM_latency)
- [http://en.wikipedia.org/wiki/CAS\\_latency](http://en.wikipedia.org/wiki/CAS_latency)

## Sarrera

- Erlojuaren ertza eragiketen hasierako eta amaierako eredua da
- **DDR (Datu-tasa bikoitza)**
  - ÿ Bitak erlojuaren beheranzko eta gorako ertzetan transferitzeko aukera ematen du (**ponpaketa bikoitza**)
  - I/O buffer maiztasuna
    - ÿ Memoriaren I/O buffer-a x2, x4 eta x8 maiztasunetara joan daiteke memoriarako sarbide-maiztasunari dagokionez. zelula.
    - ÿ **Superzelula:** orain matrize baten hautapenak (errenkada, zutabea) ez du gelaxka 1 hautatzea, baizik eta arrayko 2, 4 edo 8 ZELULAK.
    - ÿ DDR1: 21 zelula dituen makrozelula ÿ 2 zelula
    - ÿ DDR2: 22 zelula dituen makrozelula ÿ 4 zelula
    - ÿ DDR1: 23 zelula dituen makrozelula ÿ 8 zelula
    - ÿ DDR1: 24 zelula makro-zelula ÿ 16 zelula
    - ÿ DDR1: 25 zelula dituen makrozelula ÿ 32 zelula
- Fabrikatzaileak: Samsung, Hitachi, NEC, IBM, Siemens.

24. taula. PCrako DDR moduluak: ezaugarriak

	<b>DDR1</b>	<b>DDR2</b>	<b>DDR3</b>
bit i/o: zelulak/bus_zikloa	x2	x4	x8
autobus maiztasuna	F	2f	4f
gutxieneko leherketa	2	4	8
DIMM pinak	184	240	240
SO-DIMM pinak	200	200	144/200/ 204
elikadura (v)	2.5	1.8	1.5

# 34. kapitula. Eranskina: Memoria Birtuala

## 34.1. Bibliografia

- Ordenagailuen Antolakuntza eta Arkitektura: Errendimendurako Diseinua. William Stallings, Cap8 Systems Operatiboa: 8.3 Memoriaren kudeaketa
  - ÿ Sistema eragilea: Memoriaren kudeaketa
- Computer Systems A Programmer's Perspective, Randal E. Bryant. 9. kapitula. Memoria birtuala

## 34.2. Sistema eragileak: Memoriaren kudeaketa

### 34.2.1. Prozesu anitzeko sistemak

- Ataza anitzeko sistema batean prozesu bat baino gehiago dago martxan eta memoria nagusian bizi da.
- Memoria nagusia prozesu guztiak partekatzen duten baliabidea da. Hari anitzeko ingurune batean, beharrezkoa da partekatutako baliabidea kudeatzea prozesu bakoitzari memoria fisikoaren eskualdeak esleitzeko, prozesu ezberdinaren arteko memoria espazioak babesteko, etab.
- Historikoki, memoria nagusia oso mugatua zen programen tamainarekin alderatuta.
- Memoria kudeatzeko teknikak sistema eragileetan: trukatzea, partizioa, memoria birtuala, segmentazio, orriketa. .

### 34.2.2. Memoria fisikoaren kudeaketa

#### Oroimen nagusia

- Memoria nagusia CPUtik kanpoko memoria fisikoa da eta SDRAM teknologiarekin implementatutakoa 2010ean 4GBko ohiko edukiera.

#### Trukatzea

- Trukea esan nahi du.
- Programak disko gogorrean gordetzen dira objektu exekutagarrien modulu gisa.
- Exekutatu beharreko moduluak memorian kargatu behar dira, prozesu bihurtuz.
- Memoria mugatua da, beraz, ezin ditu erabiltzaileak eskatzen dituen prozesu guztiak gorde.
- Soluzio bat prozesuek memoria nagusia zein disko gogorra erabiltzea da. Une jakin batean exekutatzen diren prozesuak memorian daude eta prozesu horietako batek CPUrik behar ez duenean (I/O gertaera baten zain dago) (ez dago prest egoeran) disko gogorrekin trukatzen da. egiten duen prozesuak CPUa behar du (prest egoeran dago). Prozesua disko nagusira transferitzea swap-in deitzen da eta disko gogorrera transferitzea swap-out.
- Prozesu **osoa** trukatzen da.
- Memoriaren eta disko gogorren artean prozesuen truke osoa dago. Hau da, Solaris bezalako sistema eragile batzuen truke kontzeptua eta definizioz hartzen dena. Linuxen beste esanahi bat du.
- Prozesu bat memoriatik disko gogorrera eramatean, zulo bat sortzen **da memoria**. ren existentzia memorian zehar sakabanatuta dauden hainbat zulo deritzo kanpoko zatiketa.
- Eragozpena da trukatzeak I/O eragiketa bat behar duela disko gogorrean, motelduz ordenagailuaren errendimendua.

## Zatiketa (zatiketa)

- **Gakoa:**

- ÿ Memoria birtuala gabe, memoria fisikoan dauden prozesuen kodeak **ondokoa izan behar du.**
- ÿ Memoria birtuala erabiltzen ez badugu, partizio dinamikoak (prozesuak behar duen memoria-eskualdea besterik ez du okupatzen) kanpoko zatiketa sortzen du partizioak ezabatzean eta partizio finkoak (prozesuak erreserbatutako eskualdea baino gutxiago okupatzen du) barne zatiketa sortzen du.

- **Konponbidea**, prozesu batek memoria **FISIKOAN KONTU GAINeko** partizioak okupa ditzan: memoria birtuala.

- ÿ Aldakoak ez diren partizio finko txikiak esleitu ahal izateko, kanpoko partizioa desagertzen da eta barnekoa gehienez (partizioaren tamaina baino txikiagoa) murrizten da

- **Partizioan** dauden prozesu ezberdinei memoria nagusia esleitzeko teknika bat da ordenagailuan aldi berean exekutatzen ari da.

- Memoria tamaina ez-uniformeko hainbat eskualde edo partiziotan banatzen da.

- Prozesu bati tamaina bereko edo handiagoko partizio bat esleitzen zaio.

- Ataza anitzeko sistemetan erabiltzen da, non memoria nagusia prozesu anitzek partekatzen duten. Modu honetan prozesu ezberdinaren arteko memoria partekatzea, babes, baimenak, supererabiltaile eta abar kudeatu ditzakezu.

- Prozesua prest ez dagoenean, disko gogorrarekin trukea egiten da.

- Bi alternatiba: partizioen egitura denboran zehar finkoa edo aldakorra izan daiteke. Bi kasuetan Prozesu batek partizio bat behar du, hau da, ondoko memoria-kokapenen eskualde bat .

- Partizio **FINKOAK :**

- ÿ Memoria nagusia prozesuen exekuzioan zehar tamaina aldatzen ez den eskualdeetan banatzen da.

- ÿ Tamaina berdineko eskualdeekin zatiketa finkoa edo desberdinako eskualdeekin partizio finkoa tamaina.

- ÿ Prozesuei behar baino partizio-tamaina handiagoa esleitzen zaie. Honek **barne zatiketa sortzen du**, partizioaren eremu bat ez baita inongo prozesuk erabiltzen.

- Zatiketa aldakorra edo **DINAMIKOA** :

- ÿ Prozesu bakoitzari behar duen memoria besterik ez zaio esleitzen. Ez dago barne zatiketarik.

- ÿ Partizioen tamainak dinamikoki aldatzen dira prozesuak diskoarekin trukatzen diren heinean zaila da haien tamainara egokitzea.

- ÿ **Kanpo zatiketa** nabarmena da. Sakabanatuta dauden hutsuneak trinkotuz murriztu liteke, eta horretarako beharrezko da memoria nagusiko prozesuak mugitu edo lekuz aldatzea. Prozesuak lekualdatzeak esan nahi du helbide fisiko guztiak berriro ebaletxe exekuzio garaian ÿ baliteke behar den denbora bideragarria ez izatea.

- ÿ Memoria nagusia desfragmentatzen duten programen iragarkiak daude, geroztik ezinezkoa da OS soilik daki prozesu baten helbide fisikoak.

## Alternatiba

- Memoria fisikoaren zatiketa-teknikaren ondorioz zatiketaren irtenbidea hau da memoria birtuala segmentatua edo orrialdekatua edo biak.

## 34.2.3. Memoria Birtualaren bidez kudeatzea

### Alternatiba

- Prozesuen memoria kudeatzeko arazoak prozesuak zuzenean esleituz a helbide fisikoak memoria birtualaren mekanismoaren bidez ebaletzen dira.

## Helbide birtuala

- Programatzaleak, konpilatzaleak, estekatzaileak eta prozesuek ez dute helbide fisikoekin funtzionatzen.
- Programatzaleak iturburu-moduluaren memoriarri erreferentzia egiten dio ikurrekin (etiketak, aldagaia, izen-izenak). funtzioak, etab)
- Konpilatzaleak eta lokailuak irudimenezko memoria lineal eta ondoko baten helbideetara itzultzen dituzte sinboloak memoria birtuala deritzona.
- Helbide fisikoekiko independentzia horrek asko errazten du memoriaren kudeaketa.

## Adibidez: irten programa

- Diskoan bizi den objektu exekutagarriaren modulua desmuntatzea

ÿ objdump -S irten

irteera: fitxategi-formatua elf64-x86-64

.testua atalaren desmuntatzea:

```
0000000000400078 <.text>:  
400078: 48 c7 c0 3c 00 00 00 40007f: 48 c7      mov    $0x3c,%rax  
c7 ff 00 00 00                                mov    $0xff,%rdi  
400086: 0f 05                                syscall
```

ÿ Helbideak 0x400078,... espazio birtualeko helbideak dira. Helbide-espazio birtuala lineala, ondokoa eta bakarra da.

ÿ 0x0000000000400078 helbide birtualak 16 zifra hamaseimal ditu, hau da, espazioa.

Irteera-prozesuaren helbide birtualak 264 byteko edukiera du.

## Memoria: partekatutako baliabidea

- Prozesu bakoitzak bere helbide birtuala du.
- Prozesu guztiak memoria fisiko bera partekatu behar dute.
- Espazio birtual guztiak espazio fisiko berdinera itzuli behar dira.

## Itzulpen birtuala ÿ fisikoa

- PUZak memorian sartzen den bakoitzean instrukzioak eta datuak lortzeko edo emaitzak idazteko, hala izango da Helbide logikoa helbide fisiko batera itzultzea beharrezkoa da. Hau da, helbideak
- Itzulpen hau hardware-kudeaketako memoria-unitateak (MMU) egiten du.

## Helbide logikoa

- Memoria eta disko gogorraren artean trukatzean, datuak eta argibideak kargatzen diren memoria fisikoaren helbideak alda daitezke. Hori dela eta, helbide fisiko absolutuak erabiltzen dituen helbideratze-eredu bat ez da bideragarria.
- Prozesu-helbideak oinarrizko helbide batekin erlatiboki adierazten dira. Bikotea oinarrizko helbidea eta desplazamendua helbide logikoa deritzo.
- Helbide logiko hau helbide birtuala da, ez helbide fisikoa.

## Bi mota: Segmentazioa eta orria

- Memoria birtualaren espazioa bi mekanismo edo horien konbinazio baten bidez kudeatu daiteke:
  - ÿ Segmentazioa
    - ÿ Prozesu baten memoria birtuala izeneko unitate logiko zatiezinetan banatzen da segmentuak
  - ÿ Paginazioa
    - ÿ Prozesu baten memoria birtuala eta ordenagailuaren memoria fisikoa orrialde izeneko unitateetan banatzen dira (memoria birtualean logikoa eta memoria nagusian fisikoa).

### 34.3. Memoria birtuala segmentatua

#### 34.3.1. Segmentazioaren interpretazioa

- Segmentazioa helbide fisikoan zein helbide fisikoan aplika daiteke birtuala.
  - to. Helbide-espazio birtualeko segmentazioa
    - ÿ Programa bat (prozesua) unitate logikoetan zatitzea: kodea, hasieratutako aldagaiak, hasieratu gabeko aldagaiak, irakurtzeko soilik datuak, etab. Prozesu baten memoria birtuala zatitza ondoko memoria eremuetan, zeinen tamaina dinamikoki alda daitekeen. Segmentu logikoak ezin dira zatitu.
    - ÿ Konpilatzailaren, loturaren, partekatzeko eta abarren lana errazten du.
    - ÿ Prozesu guztiek osatzen duten memoria birtualaren espazio osoa segmentuaren oinarrizko helbideak eta programa-kontagailuaren erregistroaren desplazamenduak osatuko luke.
    - ÿ Memoria birtual segmentatua erabili da CPUetan: 80286, 80386, 80486 eta Pentium
  - b. Helbide fisikoaren espazioaren segmentazioa.
    - ÿ Intel 8086 arkitekturan 16 biteko helbide-busetik 20 biteko batera pasatzeko erabiltzen zen. erregistroen tamaina 16 bitekin mantenduz.
    - ÿ Helbide fisikoaren espazioa handitu segmentu-erregistroa gehituz eta programa-kontagailu-erregistroaren tamaina handitu gabe. Esate baterako, segmentaziorik gabeko 16 biteko Intel mikroprozesadore batek  $2^{16} = 64KB$ -ra mugatzen du espazio fisikoa. Mikro berarekin eta 16 biteko RS segmentu-erregistro gehigarri batekin RS erregistroa ordenagailuko programaren kontagailuarekin kateatu dezakegu 32 biteko helbide fisikoak osatuz, eta horiekin  $2^{32} = 4GB$ ko helbide fisikoa izango genuke.

#### 34.3.2. Atalak

- Iekuz alda daitekeen objektu-modulu bakoitza ataletan egituratuta dago
- atal bat zatiketa logikoa da, ez fisikoa.
- atalaren egitura iturburu-moduluan zehazten da
- Atal nagusiak
  - ÿ testua: argibideak
  - ÿ datuak: hasieratutako aldagaiak
  - ÿ rodata: irakurtzeko soilik aldagai
  - ÿ bss: hasieratu gabeko aldagaiak
- readelf -S gehienez

16 atal goiburu daude, 0x448 desplazamendutik hasita:

Atalaren goiburuak:

[Nr] Izena	Mota	Helbidea	Desplazamendua		
Tamaina	EntSize				
[0]	NULL	Banderak Estekaren informazioa Lerrokatu 0000000000000000 00000000	0	0	0
	0000000000000000 0000000000000000 [ 1].interp				
0000000000000001c	PROGBIAK	0000000000400158 00000158			
	0000000000000000 A		0	0	1
[2].hash	HASH	0000000000400178 00000178			
	000000000000000c 0000000000000004 A [ 3].dynsym		3	0	8
0000000000000000	DINSIMIA	0000000000400188 00000188			
	0000000000000018 A		4	1	8
[4].dynstr	STRTAB	0000000000400188 00000188			
	00000000000000b 0000000000000000 A		0	0	1
[5].testua	PROGBIAK	0000000000400193 00000193			
	000000000000037 0000000000000000 AX		0	0	1
[6].eh_markoa	PROGBIAK	00000000004001d0 000001d0			
	0000000000000000 0000000000000000 A [ 7].dinamikoa		0	0	8
00000000000000d0	DINAMIKOA	00000000006001d0 000001d0			
	0000000000000010 WA		4	0	8
[8].datuak	PROGBIAK	00000000006002a0 000002a0			
	00000000000000e 0000000000000000 WA		0	0	1
[9].arazte_aranges	PROGBIAK	0000000000000000 000002b0			
	000000000000030 0000000000000000		0	0	16
[10].arazte_informazioa	PROGBIAK	0000000000000000 000002e0			
	000000000000078 0000000000000000		0	0	1
[11].debug_abrev	PROGBIAK	0000000000000000 00000358			
	000000000000014 0000000000000000		0	0	1
[12].arazketa_lerroa	PROGBIAK	0000000000000000 0000036c			
	00000000000004a 0000000000000000 [13] .shstrtab		0	0	1
000000000000008d	STRTAB	0000000000000000 000003b6			
	0000000000000000 [14] .syntab		0	0	1
	SYMTAB	0000000000000000 00000848			
0000000000000000 15] .strtab			hamabot	hogei	8
0000000000000006f	STRTAB	0000000000000000 00000a88			
	0000000000000000		0	0	1

Banderen giltza:

- W (idatzi), A (alloc), X (exekutatu), M (batzea), S (kateak), I (handia)
- I (informazioa), L (esteken ordena), G (taldea), T (TLS), E (baztertu), x (ezzaguna)
- O (OS prozesatzea beharrezko da) o (OS espezifiko), p (prozesadorearen espezifikoa)

### 34.3.3. Atala Esteka

- Lokatzaileak modu antolatuan nahasten ditu leku aldaketa daitezkeen objektuen modulu guztien atal mota bakoitza objektu exekutagarri bakarreko modulua sortzea
- Tokiz alda daitezkeen hiru objektu-moduluren adibidea:
  - ÿ p1.c, p2.c, p3.c hiru iturburu-moduluak p1.o, p2.o eta p3.o sortzen dira.
  - exekutagarria sorrarazten duen esteka p

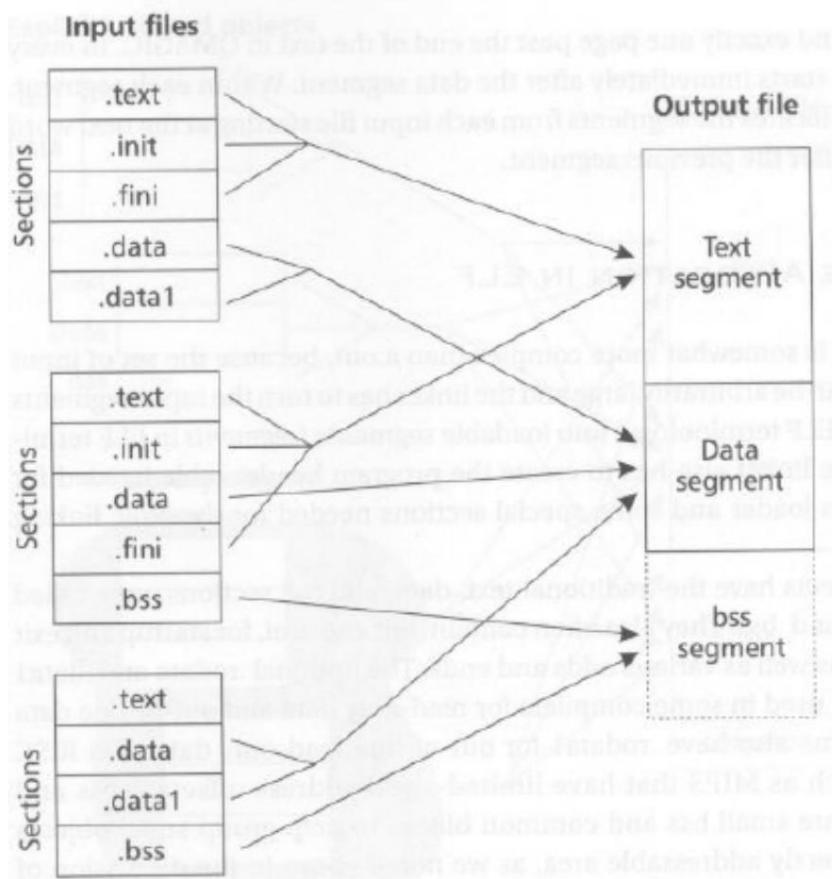
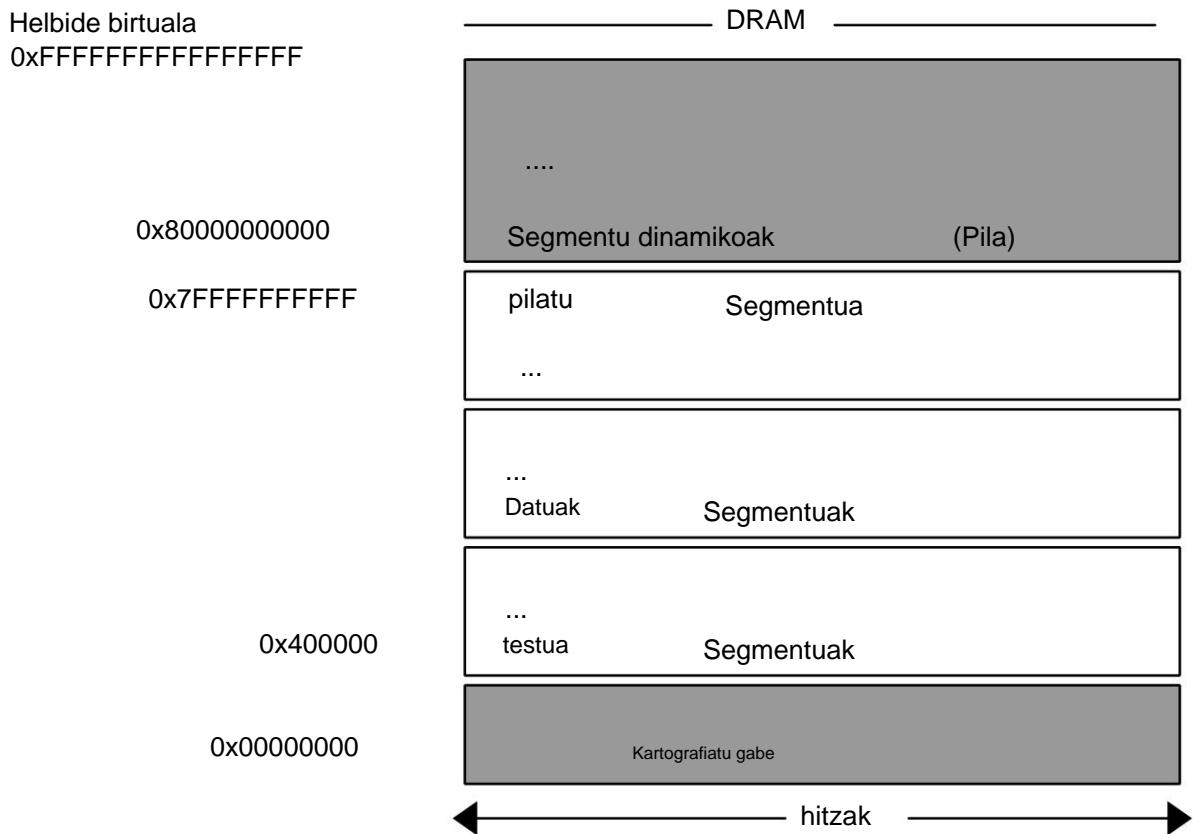


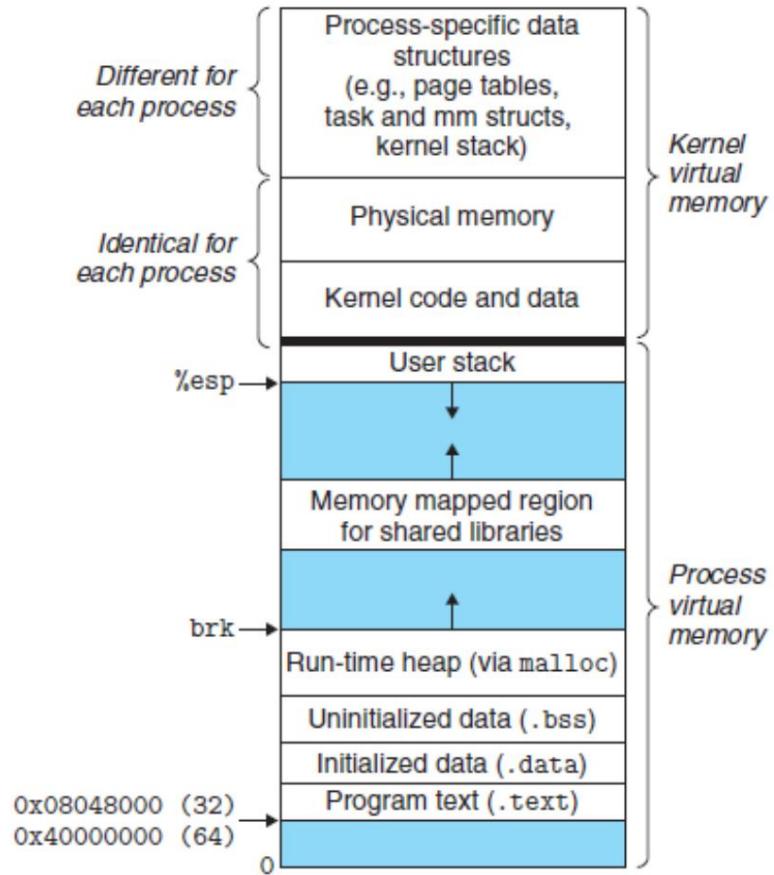
FIGURE 4.9 • ELF linking.

### 34.3.4. Segmentu logikoak

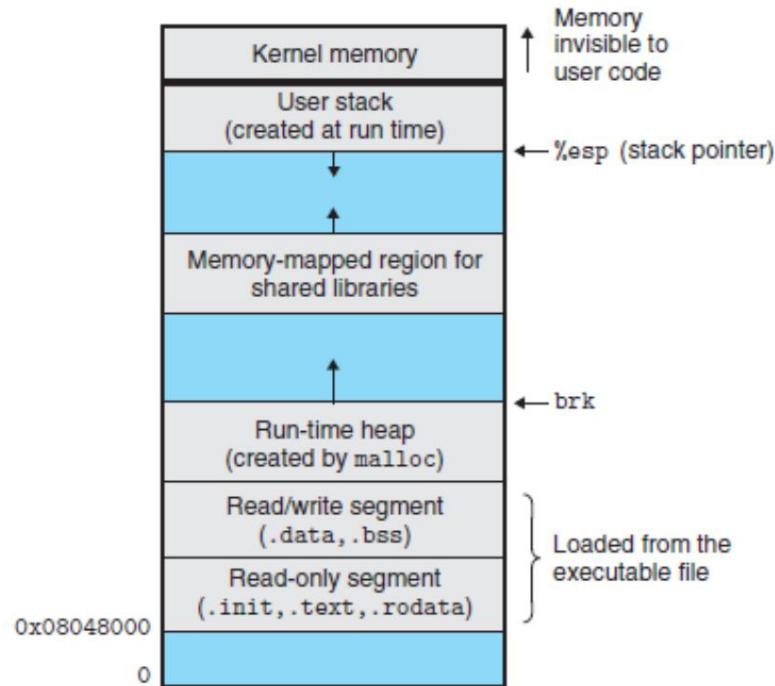
- Modulu exekutagarria segmentutan egituratuta dago
  - ÿ testua
    - ÿ Exekutatu beharreko jarraibideen kodea
    - ÿ datuak
      - ÿ datu-kodea: hasieratu gabeko aldagaiak
      - ÿ pilatu
        - ÿ pilatu
        - ÿ pila
        - ÿ tumulua
        - ÿ exekuzio-denborako memoria-esleipena da
        - ÿ C-n malloc() funtzioa : memoria-esleipena: `void *malloc(tamaina_t tamaina)`
        - ÿ tamaina: esleitu beharreko memoriaren bytetan dagoen tamaina
        - ÿ Erakuslea itzultzen du esleitutako memoria-eskualdera
  - martxan dagoen programaren memoria-mapa



**Figure 9.26**  
The virtual memory of a Linux process.

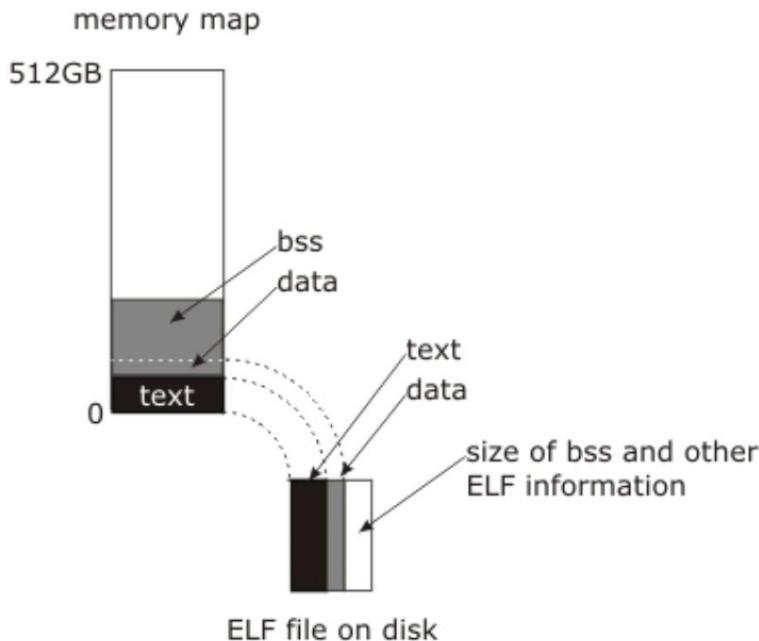


**Figure 7.13**  
Linux run-time memory image.



115. irudia linux\_vm\_map\_2

- Prozesu bakoitzak bere memoria birtuala du gainerako prozesuetatik independentea
- Segmentuek tamaina dinamikoki alda dezakete exekuzioan.
  - ÿ Objektu exekutagarriaren modulu kargatzea
- Kargatzaileak ez du modulu exekutagarria DRAMen kargatzen, baizik eta fitxategia memoria birtualean mapatzen du, sortuz orrialde-taula.
- Kargatze eraginkorra eskaeraren arabera egiten da.



### 34.3.5. Intel 8086-80286 memoriaren bilakaera

#### 8086

- 80x86 ÿ (helbide-busaren bitak, datu-busaren bitak)

- 8086 ÿ (20,16) ÿ 2 20=1MB memoria fisikoÿ Modu erreala

ÿ Segmentazioa

ÿ Helbide logikoa bi balioko tupla batek osatzen du: oinarrizko helbidea eta offset.

ÿ Helbide logikotik helbide fisikora bihurtzea:

ÿ 16 biteko programa-kontagailuarekin, 64KB bideratu daitezke. 16 biteko segmentu-erregistro gehigarri bat gehitzen badiogu zeinaren edukia 4 bit ezkerrera desplazatzen dugun ( 24z **biderkatzearen** baliokidea ) 20 biteko oinarrizko helbide bat izango genuke eta horri 16 biteko PC offset-a gehituko genioke, 20 bat lortuz. -bit helbide fisikoa. -> 1 MB espacio fisikoa.

ÿ Memoria modu honi modu erreala deitzen zitzaion: memoriaren helbide-espazio segmentatua 20 bit.

## 80286

- 80286 ÿ (24,16) ÿ 2 24=16MB memoria fisikoÿ Modu errealkak eta babestuak.

ÿ Memoria birtualaren kontzeptua: konpilatzaleak eta prozesuek exekutatzen dutenean sortutako memoria

ÿ Kasu honetan 4 byte daude memoria birtual ÿ 2 byte altuenak segmentu-hautatzailea dira eta bi byte baxuenak desplazamendua.

ÿ Memoria birtualaren edukiera  $2^{32} = 4 \text{ GB}$

ÿ Prozesuen memoria birtuala segmentutan banatzen da.

ÿ Segmentazioa

ÿ Datu, kode eta pila-segmentu desberdinak osatuz eta gainjartzea saihestuz

ÿ Segmentu bakoitzak 64KB soilik zuzendu ditzake, Programa-kontagailua 16 delako bitsak

ÿ Memoria logikoa memoria fisikora bihurtzea:

ÿ CS,DS,ES,SS 4 segmentu-erregistroetako bat erabiltzen da: 64 bit dira: 16 bit ikusgai eta 6 ezkutuko byte

ÿ Memoria nagusian bizi den segmentuen deskribapen-taula erabiltzen da: taulako sarrera bakoitza 8 bytekoa da, eta horietatik 3 byte dira segmentatutako helbide birtualarekin lotutako oinarrizko helbide fisikoa.

ÿ Helbide birtualeko 2 byte altuenak segmentu-erregistroaren zati ikusgaien kargatzen dira (segmentu-hautatzailea)

ÿ Segmentu-hautatzaileak segmentuaren deskribapenaren hautapen-taularen sarreretako batera seinalatzen du eta taulako 6 byte kargatzen ditu segmentu-erregistroaren eremu ezkutuan, honako hauek dituena: oinarrizko helbide fisikoa (3 byte), segmentuaren tamaina (2 byte) eta segmentuaren propietateak (byte 1)

ÿ Helbide fisikoa: oinarrizko helbide (3byte) gehi desplazamendua (2byte): 3 byterekin ÿ 2 24=16MB espacio fisikoa

ÿ Segmentu bateko 4GB memoria birtualaren helbide-espazioa 16MB-ko helbide fisikora itzul daiteke, baina 64KB baino ezin du atzitu.

ÿ Espazio fisiko osoa: 16 MB posibletatik, segmentu batek 64KB baino ez ditu zuzentzen eta guk dugun bezala 4 segmentu ÿ  $4 * 64\text{KB} = 256\text{KB}$  guztira.

ÿ Multiataza, memoria kudeaketa (txipa MMU), babestutako memoria ÿ babestutako modua: memoria espazioa 24 biteko segmentatutako memoria helbideak.

**80386**

- 80386 ȳ (32.32) ȳ Espazio fisikoa: 232 = 4GB

ȳ 286ren arkitektura bera, baina datuen bidea 16 bitetik 32 bitera handitzen du, bi gehitzen ditu segmentu-erregistro gehiago (FS,GS) eta orri-teknika gehitzen du.

ȳ Memoria birtuala: 6 byte: 248 = 64TB. 2 byte altuak erregistro-hautatzailea dira eta 4 byte baxuak desplazamendua

ȳ 64TB posibletatik, 6 segmentuek 4GB hel ditzakete bakotza une bakoitzean ȳ  
6\*4GB=24GB

ȳ Segmentazioa

ȳ Segmentu-hautatzailea = 2 byte 286-n bezala ȳ deskribatzaile-taularen sarrerako erakuslea  
segmentua

ȳ Segmentu deskribatzailea = Oinarrizko helbide fisikoaren 4 byte ditu

ȳ Helbide logikotik helbide fisikorako bihurtzea, segmentazioarekin soilik

ȳ Helbide birtualaren desplazamendua (4 byte) oinarrizko helbide fisikoari gehitzen zaio (4 byte) ȳ  
32 biteko helbide fisikoa.

ȳ Kasu honetan, 80286an ez bezala, segmentu bakoitzaren 4GB memoria birtuala 4GB espazio fisikora itzul daiteke.

ȳ argibideak

ȳ `movl $42,%fs:(%eax)`

ȳ inplizituki

ȳ push, pop ȳ SS,DS

ȳ Ikus 80386 orrialdea

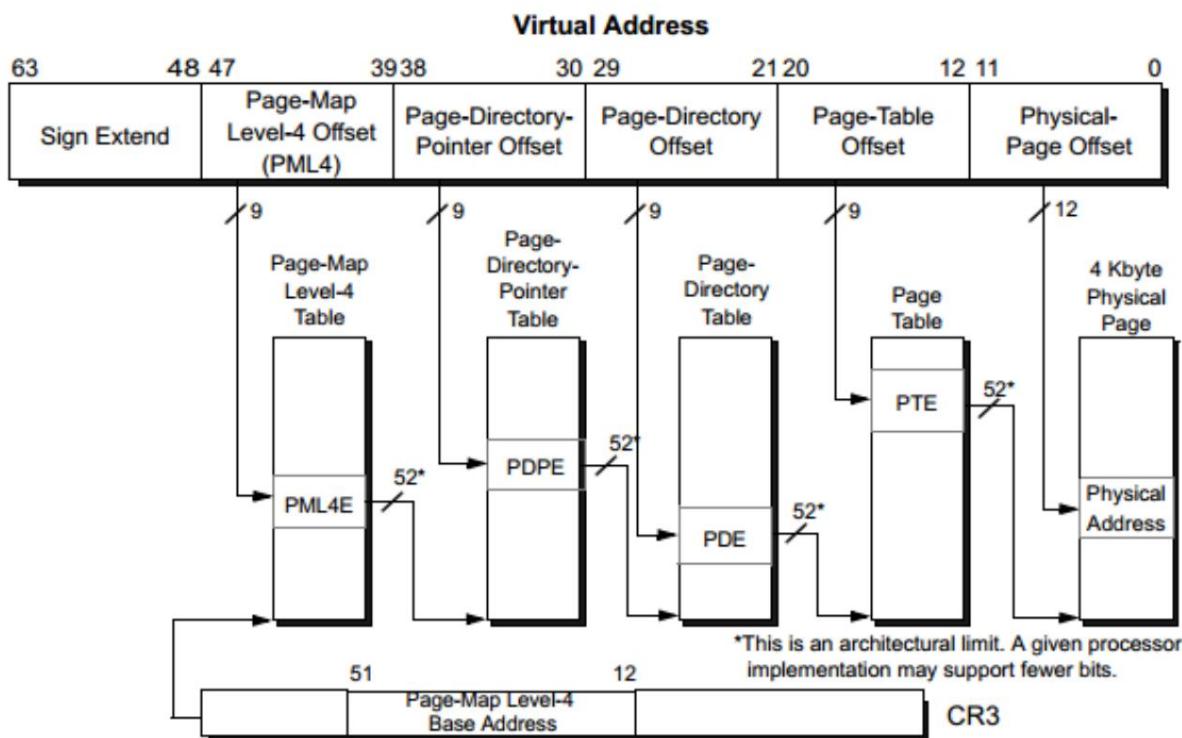
**amd64**

- amd64 ȳ (52.64) ȳ Espazio fisikoa: 252 = 4PetaByte eta Espazio Birtuala 248 = 256TB

ȳ [https://en.wikipedia.org/wiki/X86-64#Physical\\_address\\_space\\_details](https://en.wikipedia.org/wiki/X86-64#Physical_address_space_details)

ȳ **Ez du** helbide birtualeko espazioaren segmentazio logikorik erabiltzen, 256TBko memoria birtuala nahikoa delako prozesu guztietarako. Prozesu baten segmentu logikoak (testua, datuak, pila, pila, etab.) prozesu horri esleitutako espazio birtual berean biltegiratzen dira paging teknika erabiliz.

ȳ Kontuan izan behar dugu orrialde birtualen taularen muga, orrialde birtualen kopuruaren eta orri fisiko baten helbidearen araberakoa. Okupatutako memoria nagusiaren eremua esponentzialki hazten da mahaiaren tamainarekin eta beharrik gabe.



116. irudia amd64 helbide-formatua

## 34.4. Orrikako Memoria Birtuala

### 34.4.1. Oinarria

- Paging prozesuen memoria fisikoa eta memoria birtuala banatzean datza orrialdeak izeneko pieza txikiak.
- Memoria fisikoko zatiei orrialde-markoak deitzen zaie eta memoria birtualeko zatiei orri-markoak. prozesuari orriak deitzen dira
- Kasu honetan orrialde bakotza orri-marko desberdin bat esleitzen zaio, prozesu-zatiak utziz memoria- eremu **EZ JOAN EZKOTAN** sakabanatuta .
- Horrela, barne-zatiketa murrizten da, gutxiegi erabiltzen den memoria beti orrialde baten tamaina baino txikiagoa izango baita.
- Sistema eragileak prozesu bakotzerako orriak markoekin mapatzen dituen orri-taula sortzen du.
  - ÿ PTE: Orrialde Taularen Sarrera ÿ (index,PhysPageNumber)
- Helbide logikoa
  - ÿ Helbide logiko bakotza orriaren oinarrizko helbideak eta desplazamenduak osatuko du orrialdea. Helbide **ez-lineala** , (oinarrizko helbidea, desplazamendua)
- Helbide logikotik fisikorako itzulpena.
  - ÿ Helbide-espazio fisikoa bakarra eta ondokoa da, hau da, lineala.
  - ÿ Prozesu-orriaren oinarrizko helbidea markoaren oinarrizko helbidearekin lotuta dago: taula orrialdeak.
  - ÿ Markoaren barruko desplazamendua orriaren barruko desplazamenduaren berdina izango da.
  - ÿ Orrialdeen kudeaketa MMUk egiten du

### 34.4.2. Memoria birtual orrikatuaren kontzeptua

- Ez delako beharrezko prozesuaren orrialde guztiak kargatzea y memoria-espazioa  
**PROZESA memoria fisikoa baino handiagoa** izan daiteke y **memoria birtualaren** kontzeptua
- Memoria birtuala bakarra da, ondokoa, hau da, LINEALA. Abstrakzio bat da, horren menpe ez egoteko helbide fisikoak.
- Memoria fisikoan tokikotasunaren printzipioa dela eta, momentu jakin batean dinamikoki beharrezkoak diren orrialde birtualen kopia baino ez dago.

#### y cache kontzeptua

- Memoria nagusia bigarren mailako memoriaren cachea da (fitxategiak diskoan edo pendrivean)
- SDRAM cachea

### 34.4.3. Zatiketa

- Beharrezkoak ez diren memoria-eremuen desalojoan, memoria zatikatzen duten hutsuneak sortzen dira. memoria fisikoa erabilitako memoria-eremuen eta erabili gabeko eremuen segida batean
- Barne zatiketa txikiagoa izango da orrialdeak zenbat eta txikiagoak izan.
  - ÿ Orrigintzan, orriek erabili gabeko byteak izan ditzakete, orrien barruko hutsuneak dira.
- Kanpoko zatiketa murrizten da, prozesu bati orri-markoak eslei dakizkioelako prozesuaren tamaina edozein dela ere.
  - ÿ Prozesu batek ondokoak izan behar ez duten orrialdeek utzitako hutsuneak beteko dituzte. Prozesuaren tamainak behar diren orrialde kopuruari eragingo dio prozesu osoa memoria nagusian egon nahi badugu.

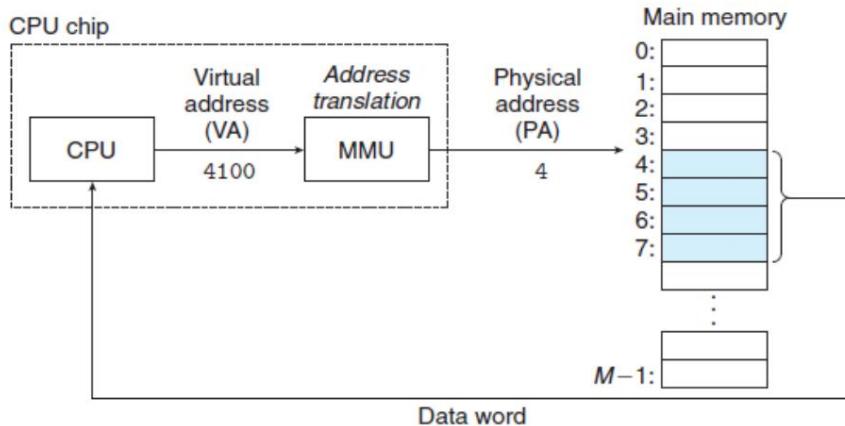
### 34.4.4. MMU

- PUZaren barneko hardware-unitatea
- Bere sarrera helbide birtualaren busa da eta bere irteera helbide fisikoaren busa da.
- MMUak segmentuaren deskribapen-taula eta orrialde-taulara sartzen ditu eta helbide birtuala helbide fisikorako itzulpena egiten du
- MMU orri-taula espazio birtuala eta fisikoa mapatzen duen mapea-funtzioa da.

### 34.4.5. Memoria Birtuala Cachea

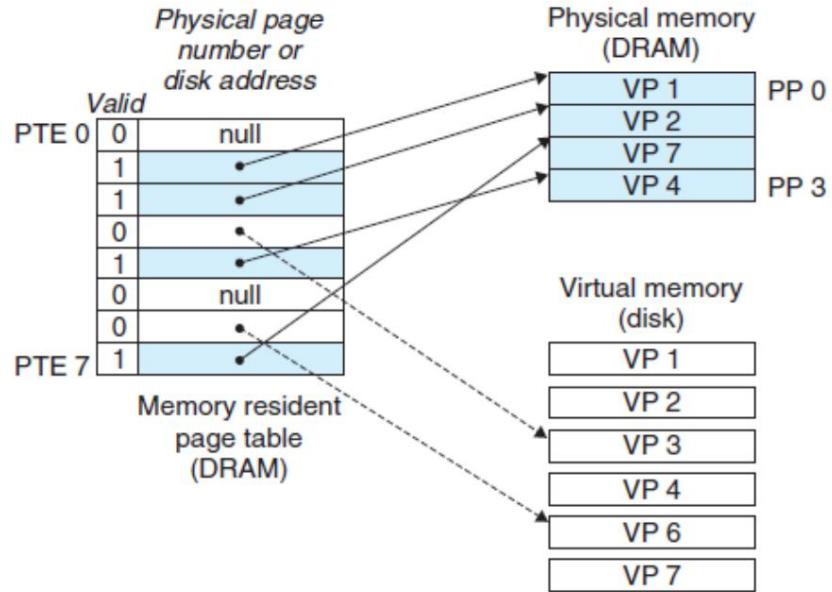
- Orrialde-taula guztiz asoziatiboa den mapa-funtzioarekin (VPorriak edozeinekin lotuta daude orriaren markoa)
- Balioztatze-bit
  - ÿ 1 y cachean gordetako orria
  - ÿ edo y cached gabeko orria: esleitua edo esleitu gabea

**Figure 9.2**  
A system that uses virtual addressing.



117. Irudia MMU

**Figure 9.4**  
Page table.



118. Irudia MMU

### 34.4.6. Orrialde taula

- Orrialde-taula SDRAM memoria nagusian dago.
- Taularen sarrerak orri fisikoen markoen erakuslea dira
- Orri birtualak bezainbeste sarrera
- Orrialde birtualaren zenbakia taularen aurkibidea da.
- MMU-k orrialde-taulan sartzen da eta helbide birtuala helbide fisikora itzultzen du
- Nukleoak orri-taula eguneratzen du eta transferentziak aktibatzen ditu

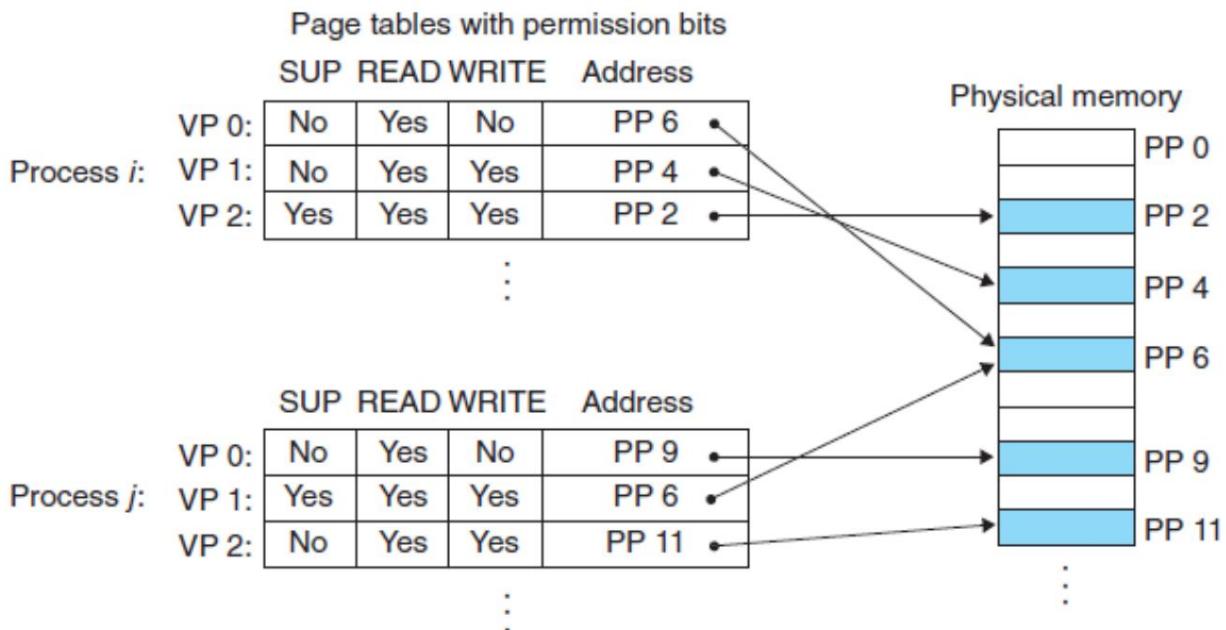


Figure 9.10 Using VM to provide page-level memory protection.

119. Irudia Babesa

- SUP: SUPERvisor: kernelak bakarrik du sarbidea
- Idatzi Ez: irakurtzeko soilik.

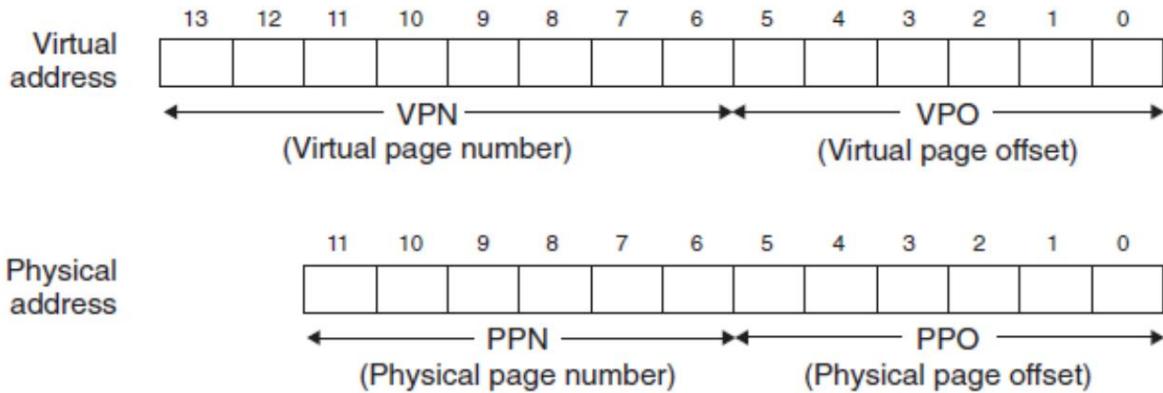


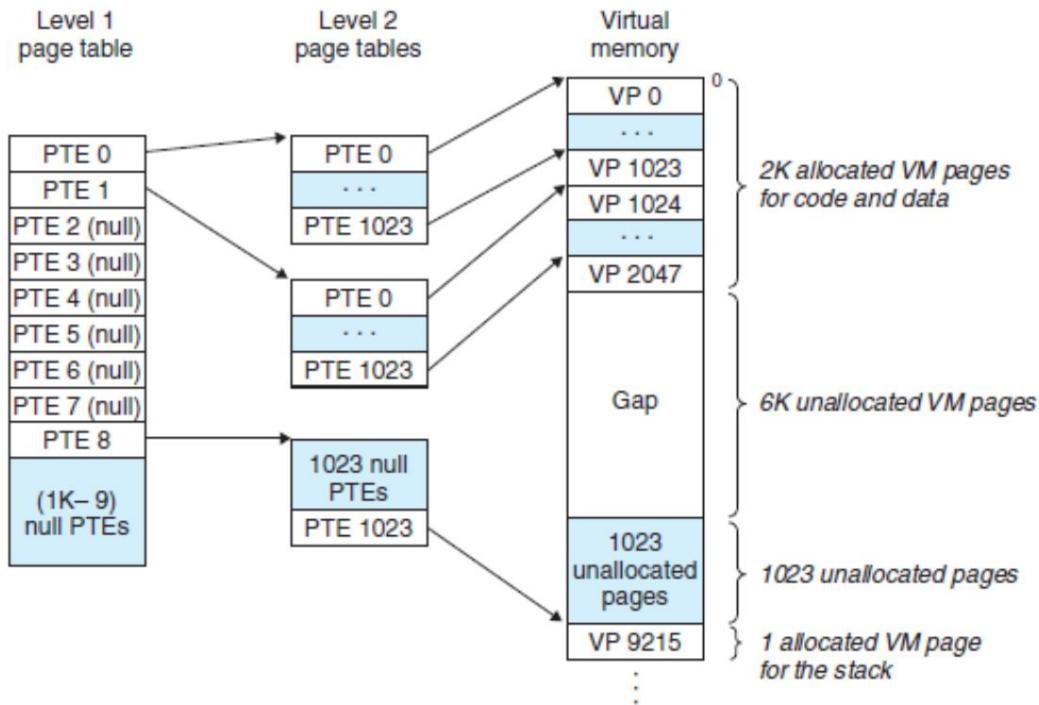
Figure 9.19 Addressing for small memory system. Assume 14-bit virtual addresses ( $n = 14$ ), 12-bit physical addresses ( $m = 12$ ), and 64-byte pages ( $P = 64$ ).

120. Irudia Helbide-formatua

#### 34.4.7. Maila anitzeko orrialdea

- Orrialde-taula handia izan daitekeenez, orrialde-taulak kontsumitzen duen ram-aren eremua murriztea jotzen da. taula hainbat taularen antolaketa hierarkikoa erabiliz.
- Memoria hierarkikoki banatu daiteke orrialde multzoetan. Orriak taldekatzen dituzten superorriak, superorriak taldekatzen dituzten hiperorriak.
- Adibidea: 3 mailako orrialdea: 16 MB zatien 1. maila, 2 MB zatien 2. maila, 2. maila 4KB orriak.
- Taldekatze-maila bakoitzak maila hori deskribatzen duen taula bat du. Orrialde-taula taula anitzeko hierarkia bihurtzen da.

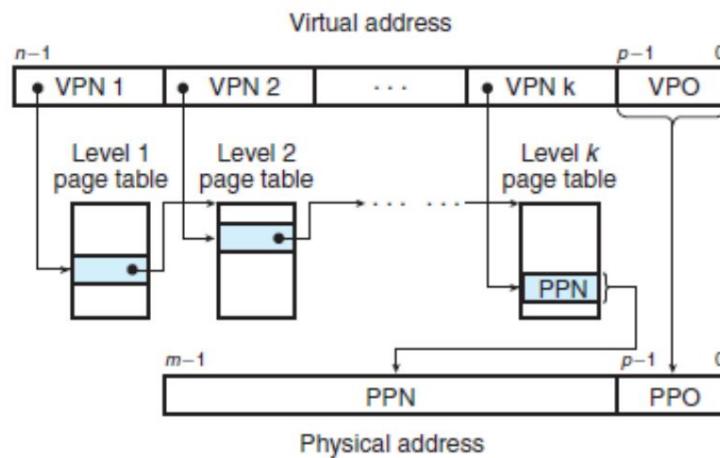
- MMU itzulpen prozesuan helbide birtuala hainbat eremutan deskonposatzen da. Bakoitzak eremua erlazionatutako taula bakoitzaren indizea izango da, maila adina orrialde lotuz.
- Maila-taulak NULL edukia badu, taulako goi-mailako i+1, i+2, etab.-taulak ez dira existituko. lotura-katea.
- Hainbat taula sartzeak ez du helbideen itzulpena moteltzen taulak MMUren barneko cachean implementatzen badira. Desberdina izango litzateke orrialde hauek DRAM memorian egongo balira.



**Figure 9.17** A two-level page table hierarchy. Notice that addresses increase from top to bottom.

121. Irudia. Bi mailatako taula

**Figure 9.18**  
Address translation with a  $k$ -level page table.



122. Irudia K mailen taula

#### 34.4.8. Intel: Memoria birtualaren bilakaera

80386

- Paginazioa lehen aldiz erabiltzen da.
- Memoria birtuala memoria fisikora itzultzeak bi fase hartzen ditu: lehenik eta behin segmentazioa eta gero

orrialdea (aukerakoa)

- Ikusi segmentazio mekanismoa.
- Segmentazioak espazio birtuala 32 biteko espazio lineal batean itzultzen du eremuekin: dir(10 bit)-p(10)-desplazamendua (12)
- Orrialde-taulen bi maila implementatzen dira: dir orri-taulen erakusleen taula da (orriaren direktorioa)
  - ÿ 10 bitekin 210 erakusle lortzen dituzu tauletarako ÿ 1K taulak
- p orri-taularen indizea da
  - ÿ 10 bitekin 210 taulako sarrera lortzen dituzu ÿ 1K orrialde fisikoekin lotutako 1K orri birtualak
  - ÿ Orrialde fisikoaren helbidea 32 bitekoa da
- 12 bit desplazamenduarekin orriaren tamaina  $2^{12}=4\text{KB}$  da
- Taula bakoitzak 1K orrialde dituen 1K taulak guztira 1M orrialde dira eta 4KB orrialde bakoitzak bat ematen du. guztira 4 GB memoria fisiko helbide.
- Beraz, memoria birtual posiblaren 64TBtik une bakoitzean 24GBra itzul dezakegu memoria segmentatua eta segmentu bakoitza 4GB linealetik 4GB memoria fisikora.

#### amd64

- amd64 ÿ 64 bit ÿ Espazio Birtual Teorikoa =  $2^{64} = 16 \text{ ExaBytes}$
- ÿ Paginazioa eta **ez segmentazioa**.
  - ÿ Espazio birtuala = 256 TeraByte, CPUak helbide birtualerako 48 bit baino ez dituelako erabiltzen, egungo aplikazioetarako memoria nahikoa delako, 64 bit erabiltzeak orrialde-taulak handiak eragingo lituzke, sistemaren errendimendua alferrik jaitsiz. Ez dago bigarren mailako memoriarik ere hainbeste memoria birtualarentzat.

### 34.4.9. Glosarioa

- Espazioak: Logika (segmentazioa) ÿ Logika Lineala (birtuala, orria) ÿ Lineala Fisikoa
- LH: Orrialde Birtuala
- VA: Helbide birtuala
- PP: Orrialde fisikoa
- PA: Helbide fisikoa
- VPO: VP offset
- VPN: VP zenbakia
- TLB: Translation lookaside Buffer: orrialde-taularen lookaside buffer (cachea). Bizilaguna MMU.
- PTE: Orrialde Taularen Sarrera ÿ (indizea/edukia) ÿ (VPN/PPN)
- PTBR: CPU kontrol-erregistroa: orrialde-taularen oinarri-erregistroa: TLBrako erakuslea
- TLBI: TLB indizea ÿ cache-multzoaren eremua
- TLBT: TLB etiketa
- PPO: PP offset
- PPN: PP zenbakia
- CO: cache-ko desplazamendua superblokean
- CT: Cache etiketa

- CI: Cache-indizea edo lerroa

#### 34.4.10. Itzulpena: helbide birtuala fisikora

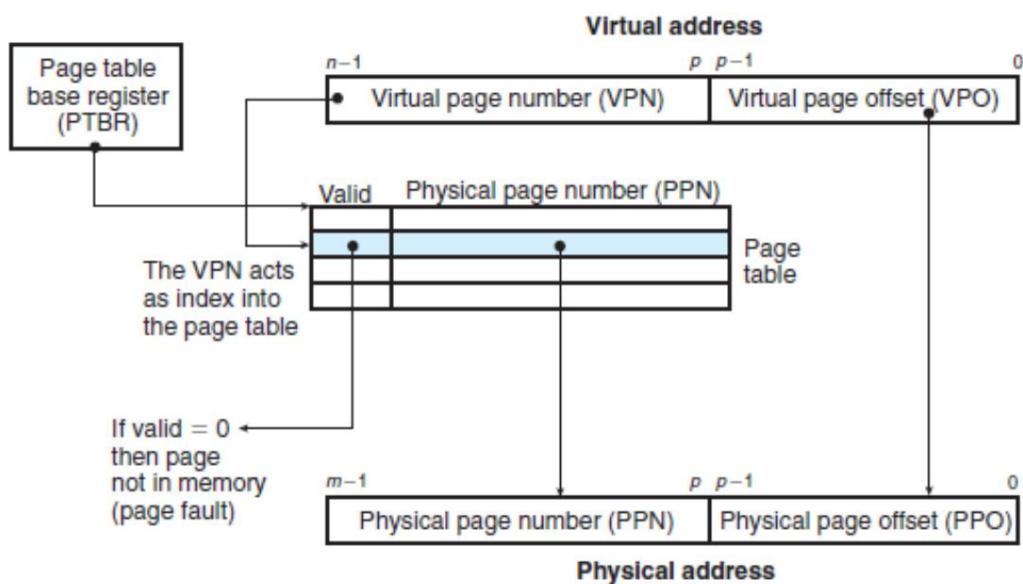
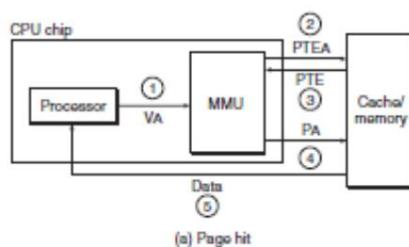


Figure 9.12 Address translation with a page table.

#### 123. Irudia Birtuala ѕ Itzulpen fisikoa



124. Irudia Emaitza arrakastatsua

1. CPU: deskargatu memoria birtualaren helbidea
2. MMU: memoria nagusian dagoen orrialde-taularen sarrera seinalatzen du
3. Memoria nagusia: taulako sarreraren edukia itzultzen du. MMU: Helbide logikotik helbide fisikoa lortzen duzu.
4. MMU: helbide fisikoa sistema-busaren helbide-busera botatzen du.
5. Erreferentiatutako datuak cache memorian edo memoria nagusian egon daitezke.

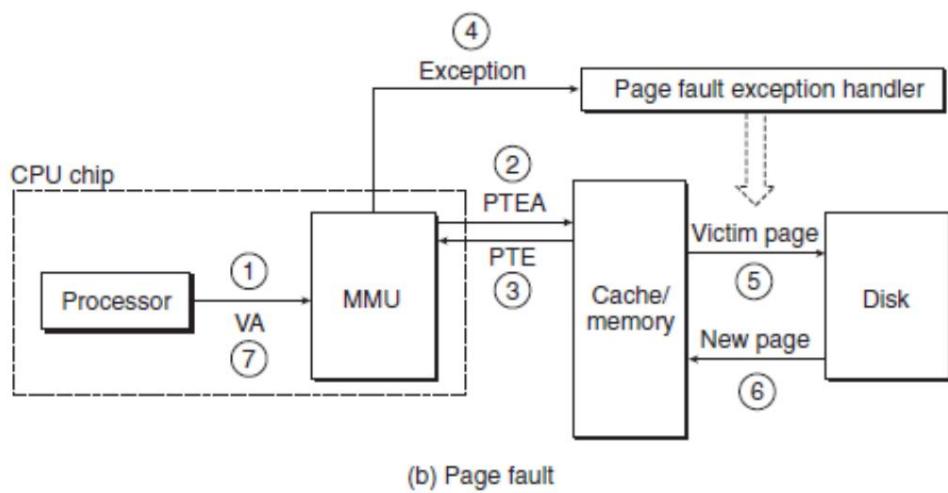
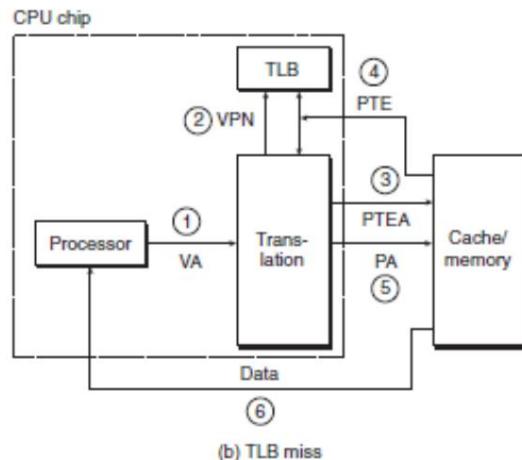
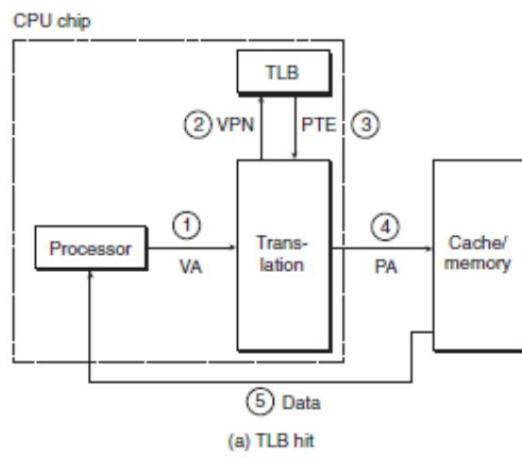


Figure 9.13 Operational view of page hits and page faults. VA: virtual address. PTEA: page table entry address. PTE: page table entry. PA: physical address.

125. Irudia. Porrotaren emaitza

#### 34.4.11. Itzulpen Lookaside Buffer

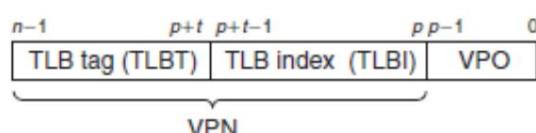
- TLB
- POS orrialde birtualeko taularen cachea da. Memoria nagusian dagoen orrialde-taulaz gain, taula horren kopia partziala dago MMUren barne memoria-unitate batean. Helburua: Taulen sarbidearen abiadura handitzea, maila hierarkikoko taula anitz ebazteko, kanpoko memoria nagusirako sarbide anitz behar baita.
- Helbide birtualaren formatua TLB mapaketa funtzio asoziatiboa duen cachea bada  
    ü Indizea cachearen multzoko eremua edo superbloke tipikoa da



**Figure 9.16** Operational view of a TLB hit and miss.

126. Irudia TLBrekin funtzionamendua

**Figure 9.15**  
Components of a virtual address that are used to access the TLB.

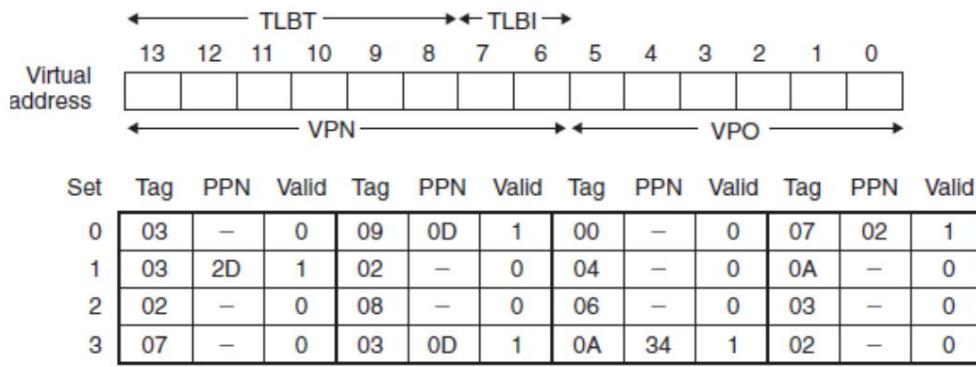


127. Irudia Formatu birtuala TLBrekin

- TLBTag
- TLBIndex

#### 34.4.12. Ariketa

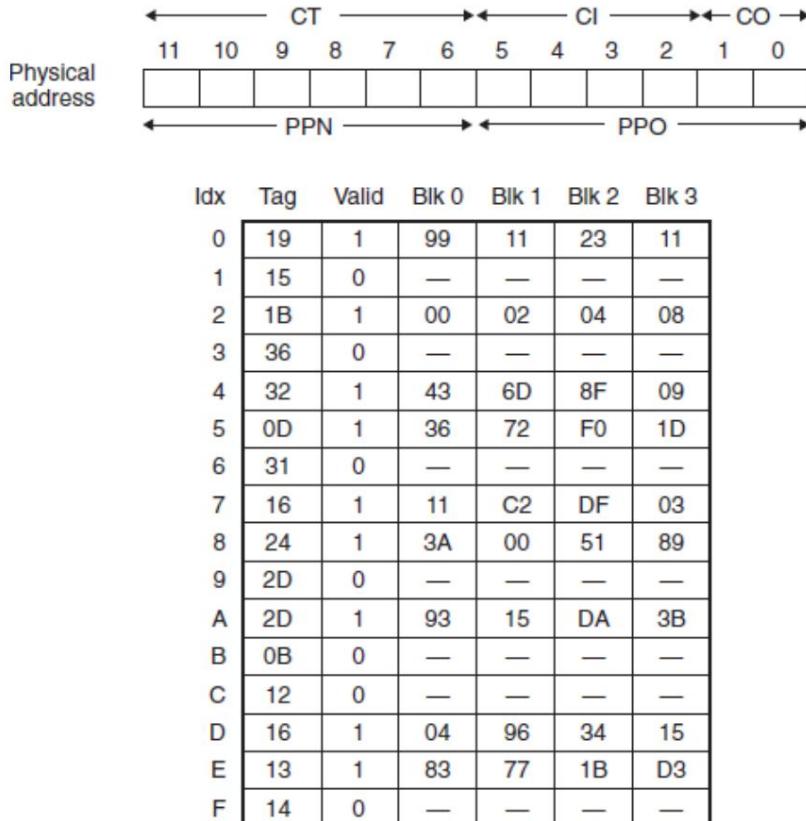
- Ordenagailu baten arkitekturak TLB eta L1 d-Cache ditu. Memoria bytetik helbideragarria da byte eta byte 1eko hitzak ditu.
- MMU-k TLB Taula (Translation Lookup Buffer) eta d-Cache memoria ditu irudien arabera.



(a) TLB: Four sets, 16 entries, four-way set associative

VPN	PPN	Valid	VPN	PPN	Valid
00	28	1	08	13	1
01	-	0	09	17	1
02	33	1	0A	09	1
03	02	1	0B	-	0
04	-	0	0C	-	0
05	16	1	0D	2D	1
06	-	0	0E	11	1
07	-	0	0F	0D	1

(b) Page table: Only the first 16 PTEs are shown



(c) Cache: Sixteen sets, 4-byte blocks, direct mapped

**Figure 9.20** TLB, page table, and cache for small memory system. All values in the TLB, page table, and cache are in hexadecimal notation.

- ÿ Helbide birtualek 14 biteko zabalera dute ( $n = 14$ ).
- ÿ Helbide fisikoek 12 biteko zabalera dute ( $m = 12$ ).
- ÿ Orriaren tamaina 64 bytekoa da ( $P = 64$ ).
- ÿ TLB lau norabideko multzo elkartua da, guztira 16 sarrerarekin.
- ÿ L1 d-cache-a fisikoki zuzenduta dago eta zuzenean mapatzen da, 4 byteko lerro-tamainarekin eta guztira 16 multzorekin.

- Kalkulatu HELBIDE BIRTUALAREN 0x03d4 helbide fisikoa

to. Helbide formatua

- Yo. VPO dimentsioa
- ii. PPO dimentsioa
- iii. VPN dimentsioa
- iv. PPN dimentsioa

b. Memoria nagusian eta TLB cachean orrialde-taularen sarrera kopurua

c. TLB

- Yo. TLB multzo bakoitzeko lerroak
- ii. TLB multzoak
- iii. LLLT tamaina
- iv. TLBT tamaina
- v. Hitz bakoitzeko bits
- zerra. TLB lerro bakoitzeko hitzak

vii. TLBI-TLBT Balioak

d. PPN TLBn al dago?

eta. PPN balioa

F. BP balioa

g. d-Cachea

- Yo. Memoria Cachea: Mota
- ii. multzoak
- iii. Lerroak/Multzoa
- iv. Hitzak/Lerroa
- v. Byte/Hitza

h. Helbide fisikoaren formatua

- Yo. CO
- ii. IC
- iii. C.T.
- iv. CT/CI/CO ÿ PA balioak

Yo. PA cachean al dago?

j. PA edukia

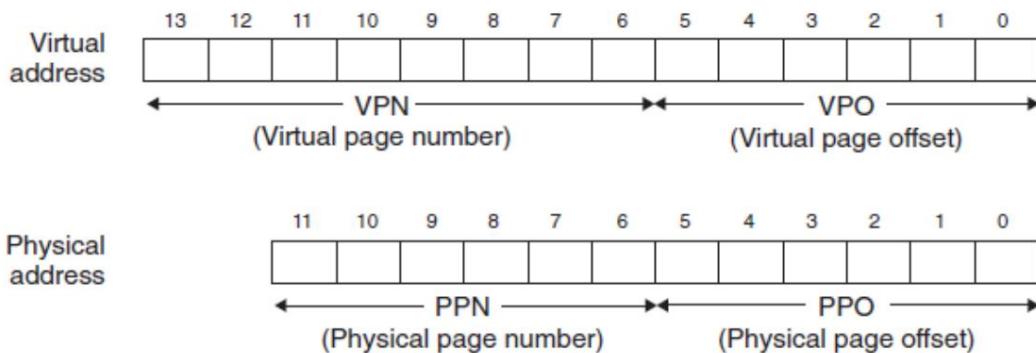
k. Azken emaitzaren laburpena

Garapena

- Erantzunak

to. Helbidearen formatua da

- Yo. VO eta PO ÿ orriaren tamaina: 64 byte ÿ 2<sup>6</sup> = 64 bit offset, birtuala zein fisikoa
- ii. VPN bitak= VA-VPO=14-6=8 bit ÿ 2<sup>8</sup> = 256 orrialde birtual
- iii. PPN bit= PA-PPO=12-6=6 bit ÿ 2<sup>6</sup> = 64 orrialdeko markoak



**Figure 9.19 Addressing for small memory system.** Assume 14-bit virtual addresses ( $n = 14$ ), 12-bit physical addresses ( $m = 12$ ), and 64-byte pages ( $P = 64$ ).

129. Irudia Helbide-formatua

b. VPN eta VPO ezarpenak

- ÿ 0x03D4 14 biteko VA helbide birtuala bitar moduan kodetuta dago: 00-0011-1101-0100 ÿ 00001111-010100 ÿ VPN-VPO
- ÿ VPO=PPO=010100=01x4
- ÿ VPN=00001111=0x0F

c. Orrialde taula

- ÿ 256 sarrera puntu. Sarrera bakoitzak 64 orrialdeko markoetako baten helbidea dauka.
- Hau da, 256 helbidez eta 6 biteko hitzez osatutako taula gehi balioztatze, babes eta abar bit.

- ÿ MP-ko taulak 256 sarrera ditu, eta cachean gordetako TLBak, berriz, 16 sarrera ditu ÿ helbideragarria.
- 4 bitekin.

d. TLB

- Yo. TLB multzo bakoitzeko lerroak: 4 modu ÿ 4 lerro/multzo
- ii. TLB multzoak: 16 sarrera 16 lerro dira guztira 4 lerro/multzotan multzokatuta = 4 multzo
- iii. TLBI tamaina: 2 bit behar dira 4 multzotarako
- iv. TLBT tamaina:

ÿ 256 sarreretarako beharrezkoak diren 8 bitetatik, 2 TLBI indizerako badira, 6 TLBT etiketarako izango dira.

v. Hitz bakoitzeko bitak: 1 byte hitz bakoitzeko, adierazpenaren arabera

zerra. TLB lerro bakoitzeko hitzak

ÿ Taularen marrazkiari erreparatzen badiot, lerro bakoitzak PPN+etiketa bakarra dauka, hau da, bat hitza.

vii. TLBI-TLBT Balioak

ÿ VPN RAM-eko orrialde-taularen helbidea da. Cache kontrolatzaleak TLBT-TLBI-n deskonposatzen du

ÿ VPN=00001111=000011-11=TLBT-TLBI=0x3-0x3

eta. PPN TLBn al dago?

ŷ 0x3 TLB multzoan begiratzen dut lerroren batek 0x3 TLBT etiketa badu eta bigarrenak badu.  
lerroa.

ŷ 3. multzoko bigarren lerroak baliozkotze-bit 1ean ezarrita dauka, beraz, orri birtuala memoria nagusian eta/edo d-cachean dago.

F. PPN balioa

ŷ 3. multzoko bigarren lerroak PPN=0x0D edukia du

g. BP balioa

ŷ 12 bit dira

ŷ PPN(6)-PPO(6) kateamendua: 001101-010100=001101010100=0011-0101-0100= **0x354**  
=PA

h. d-Cachea

Yo. Memoria Cachea: Mota: mapa zuzena

ŷ Zuzeneko mapak direnez, multzoak lerro 1 dira, beraz berdina da multzoa lerro gisa esatea.

ii. multzoak

ŷ 16 lerro

iii. Lerroak/Multzoa: 1

iv. Hitzak/lerroa: 4

v. Byte/Hitza: 1

Yo. Helbide fisikoaren formatua

Yo. CO: 4 hitz zuzentzeko, 2 bit behar dira

ii. CI: 16 lerro zuzentzeko, 4 bit behar dira

iii. CT: PA helbide fisikoa 12 bit da ŷ CT=PA-DI-CO=12-4-2=6 bit

iv. CT/CI/CO balioak ŷ PA=001101010100=001101-0101-00

ŷ 0005 linea; 00 hitza: 001101 etiketa=0x0D

j. PA d-Cachean dago?

ŷ 5. lerroan etiketa 0D da ŷ helbide fisikoaren etiketarekin bat dator ŷ arrakasta ŷ datuak sartuta daude  
d-cachea

ŷ Balidazio-bit 1 da, beraz, bere edukia eguneratuta dago eta, beraz, baliozkoa da.

k. PA edukia:

ŷ d-cacheko 5. lerroko 0 hitzaren edukia **0x36** bytea da

l. Azken emaitzaren laburpena.

ŷ **0x03d4** helbide birtuala **0x354** helbide fisikoari dagokio, zeinaren edukia **0x36** den.

### 34.4.13. Intel Core i7

## Processor package

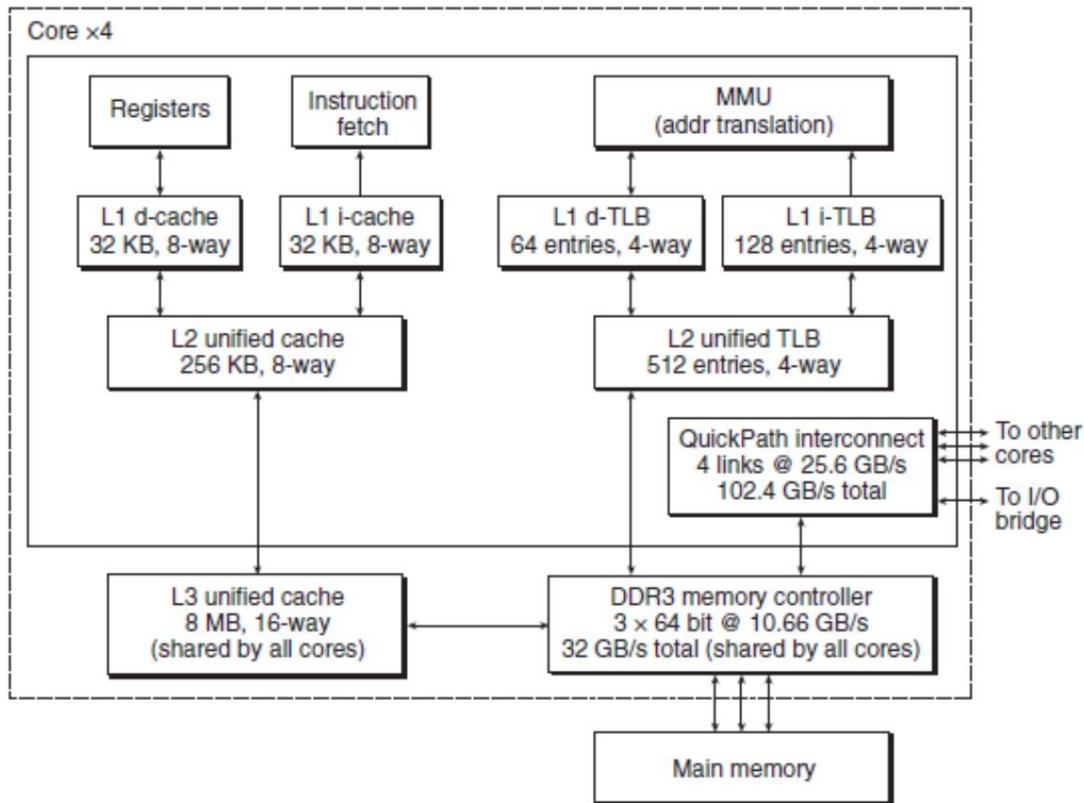
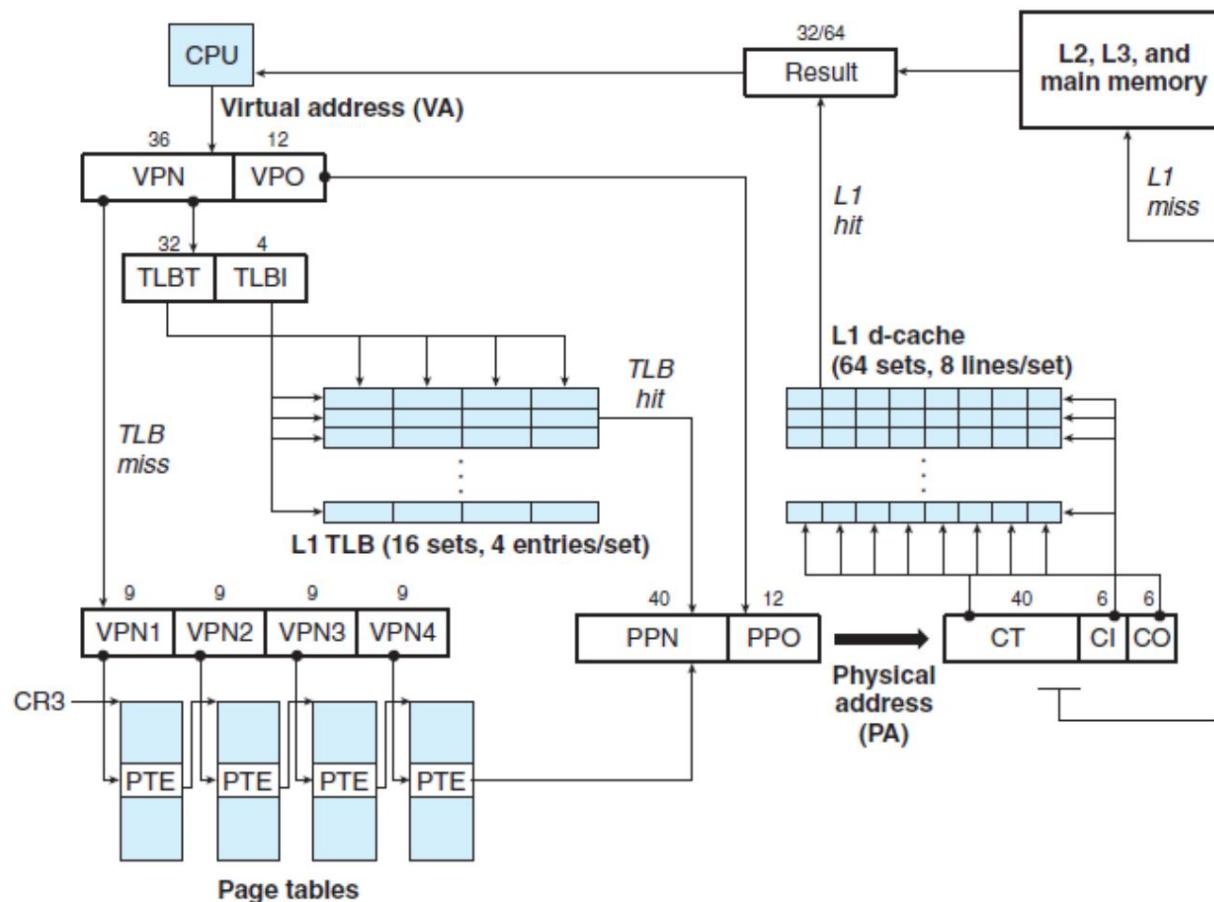


Figure 9.21 The Core i7 memory system.

130. Irudia Core i7 memoria



**Figure 9.22 Summary of Core i7 address translation.** For simplicity, the i-caches, i-TLB, and L2 unified TLB are not shown.

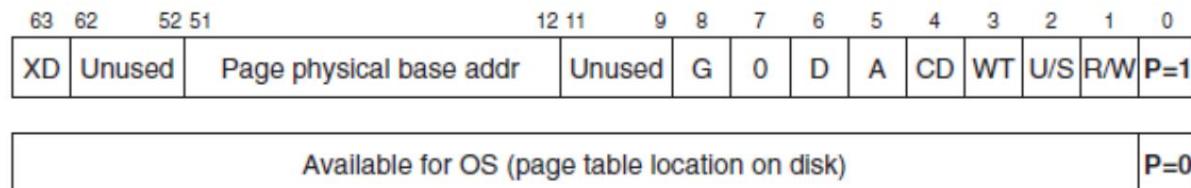
131. Irudia MMU Funtzionamendua

63	62	52	51	12	11	9	8	7	6	5	4	3	2	1	0	
XD	Unused	Page table physical base addr		Unused	G	PS		A	CD	WT	U/S	R/W	P=1			
Available for OS (page table location on disk)															P=0	

Field	Description
P	Child page table present in physical memory (1) or not (0).
R/W	Read-only or read-write access permission for all reachable pages.
U/S	User or supervisor (kernel) mode access permission for all reachable pages.
WT	Write-through or write-back cache policy for the child page table.
CD	Caching disabled or enabled for the child page table.
A	Reference bit (set by MMU on reads and writes, cleared by software).
PS	Page size either 4 KB or 4 MB (defined for Level 1 PTEs only).
Base addr	40 most significant bits of physical base address of child page table.
XD	Disable or enable instruction fetches from all pages reachable from this PTE.

**Figure 9.23 Format of level 1, level 2, and level 3 page table entries.** Each entry references a 4 KB child page table.

132. Irudia. Lehenengo hiru mailetako taulen formatua



Field	Description
P	Child page present in physical memory (1) or not (0).
R/W	Read-only or read/write access permission for child page.
U/S	User or supervisor mode (kernel mode) access permission for child page.
WT	Write-through or write-back cache policy for the child page.
CD	Cache disabled or enabled.
A	Reference bit (set by MMU on reads and writes, cleared by software).
D	Dirty bit (set by MMU on writes, cleared by software).
G	Global page (don't evict from TLB on task switch).
Base addr	40 most significant bits of physical base address of child page.
XD	Disable or enable instruction fetches from the child page.

Figure 9.24 Format of level 4 page table entries. Each entry references a 4 KB child page.

133. Irudia 4. mailako taularen formatua

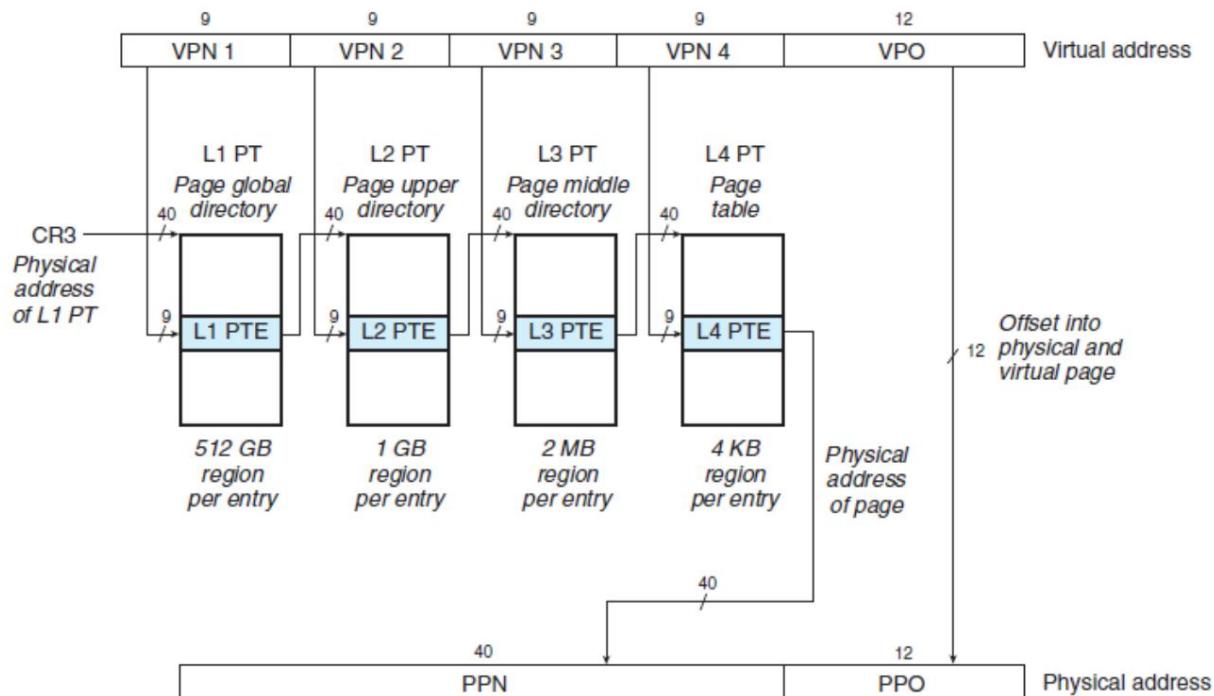


Figure 9.25 Core i7 page table translation. Legend: PT: page table, PTE: page table entry, VPN: virtual page number, VPO: virtual page offset, PPN: physical page number, PPO: physical page offset. The Linux names for the four levels of page tables are also shown.

134. Irudia Linux: 4 maila

### 34.5. Sistema eragileak: Memoriaren kudeaketa

#### 34.5.1. Babes

- Orrialde mailako babes. HW isolamendua. MMUk kudeatzen dituen orrialdeek ekintzarako, sarbideak, etab. baimenak adierazten dituzten kontrol-bitak dituzte.
- Segmentazio akatsa edo Babes akatsa

#### 34.5.2. Eskaeraren orria

- Eskariaren arabera orrikatzea
  - ÿ Prozesu baten orrialdeak eskatzen direnean soilik kargatzen dira memorian.  
Prozesu baten orrialde guztiak kargatzen dituzte.
  - ÿ Orri bat behar denean eta memorian ez dagoenean, orrialdearen akatsa sortzen da  
MMU eta OS arduratuko dira beharrezko orria kargatzeaz.
- Tokikotasunaren printzipioa
  - ÿ Une jakin batean prozesu batek erabiltzen ari diren orrialdeak baino ez ditu memorian edo erabiltzeko probabilitate handia dutenekin.
  - ÿ Gainera, denbora aurrezten da erabiltzen ez diren orriak ez direlako memorian eta memoriatik kanpo trukatzen, pieza hori berehala eska baitaiteke.

#### 34.5.3. Ordezkoa

- Orri bat ordezkatu
  - ÿ Dilema: zer orrialde ateratzen dut memoriatik? Ordezko algoritmoak ÿ Gutxien erabilitako LRU, lehen sarrerako irteera (FIFO)
  - ÿ Ordezko politika OSak kudeatzen du.

#### 34.5.4. VMTool

- Memoria birtuala tresna bat da:
  - ÿ Lehen M. disko-cachea dela
  - ÿ Kudeatu Memoria
    - ÿ Lotura sinplifikatzea: azken helbide fisikotik independente diren helbideekin nahastea
    - ÿ Kargatza erraztea: eskatutako orrialdeak eskaeraren arabera kargatzen dira
    - ÿ Partekatza sinplifikatzea: partekatzen diren prozesuak (liburutegiak).
  - ÿ Memoriaren esleipena sinplifikatza: Memoria birtualean OS edo konpilatzaleak segmentuak elkarren ondoan banatzen ditu eta gero malgutasuna dago memoria fisikoan arbitriarioki jartzeko.
  - ÿ Segmentuak babestu: kontrolatu suparekin (begiralea), irakurri, idatzi bitak

## 35. kapitulua. C Programazio Lengoaia

### 35.1. Sarrera

- Hau ez da C lengoaiaren programazioaren tutoriala, kapitulu honen helburua Konputagailuen Egitura irakasgaien erabiltzen diren C lengoaiaren programazioaren alderdi zehatzak komentatzea da.

### 35.2. Galdaketa

#### 35.2.1. Kontzeptua

- Sintaxia

(mota\_izena) adierazpena

- ÿ Bihurketa esplizitua operadore unitarioa erabiliz ( ).
- ÿ Eragile unitarioek lehentasun handiagoa dute eragile bitarrak baino.

#### 35.2.2. Adibidea

- Zenbaki osoen zatiketaren adibidea

```
int i=8,j=5;  
flotatu  
x; x = i/j;  
x = (float)i/j;
```

- Aldagai arrunta int mota gisa deklaratzten da hasieran  
ÿ i/j ÿ 8/5 eragiketak 1 zenbaki osoa izango luke
- Galdaketa (float) i aldagaian egiten badugu i aldagaia float motakoa da eta ez int, beraz bere balioa 8.000 zenbaki erreala izango da eta ez 8 osokoa.  
ÿ (flotatzalea)i/j = 8,0000/5 = 1,6000

### 35.3. Erakuslea

#### 35.3.1. Erreferentziak

KN King Testbook: 11. kapitulua. Erakusleak. 241. orrialdea

#### 35.3.2. Sarrera

Erakusle kontzeptua oinarrizkoa da maila baxuko programazio inperatiboan, datu-egitura simpleak edo konplexuak barne hartzen dituzten algoritmoak programatzeko kodea errazten baitu.

ÿ

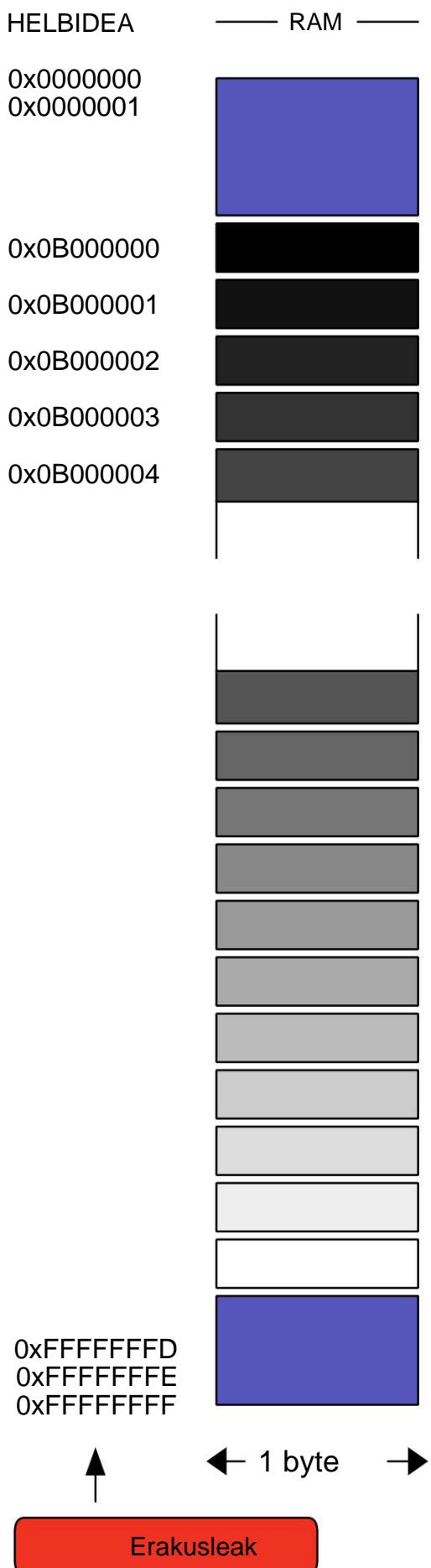
Ikasteko: erakusleei lotutako kontzeptuak, haien sintaxia, haien aplikazioa, etab..., beharrezko da programak URRATEZ URRATSA moduan exekutatu memorian dauden objektuen edukiak eta erreferentziak ikusi ahal izateko. Araztailea erabiliko dugu

GDB.

### 35.3.3. Kontzeptua

#### Memoria

- Memoria nagusia RAM Byte bideragarrietan antolatuta dago.
- Helbide-barrutia makinaren arkitekturaren araberakoa da.
- Adibidez: 48 lineaoko Helbide-Bus batek  $248 = 28 \times 240 = 256$  TB helbidera dezake
- Objektu bat memoria-eskualde bat da (byte anitz) datu oso bat, karaktere-datu bat, datu flotatzaile-matrizeari, instrukzio-blokeari, etab. Memoria-testuinguru honetan, objektu-kontzeptua objektuetara zuzendutako programazioaren objektu-kontzeptutik desberdina da.
- RAM memorian kokatzen diren memoria-helbideek erreferentzia egiten duten objektuak ezartzen dira. Erreferentzia byte anitzeko objektua gordetzen den lehenengo bytearen helbidea da.
- Memoria mapa:



## Erakuslea

Erakuslea memoria helbide baten baliokidea da

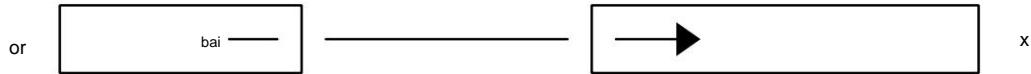
Horren ordez, ERAKUSKETA ALDAGAIA:

- Memoriaren helbidea adierazten duen datuak gordetzen dituen aldaagaia da.
- Erakusleen aldagaien erakusleak gordetzen dituzte.
- Beren balioak memoria helbideen balioetara mugatzen dituzte. Ezin da inoiz balio negatiboa izan edo benetakoak, etab.
- Objektuak seinalatzen dituzte
- Objektuak aipatzen dituzte



KNKing liburuak "erakuslearen aldaagaia" eta erakuslea bereizten ditu. Literatura orokorrean, erakusleei buruz hitz egiten dugunean, erakusle-aldagaietan ari gara, eta kasu horretan erakuslearen edukiari erreferentzia edo helbidea deitzen zaio erreferentziatutako objektuari.

"Erakuslearen aldagaiaren" irudikapen grafikoa or

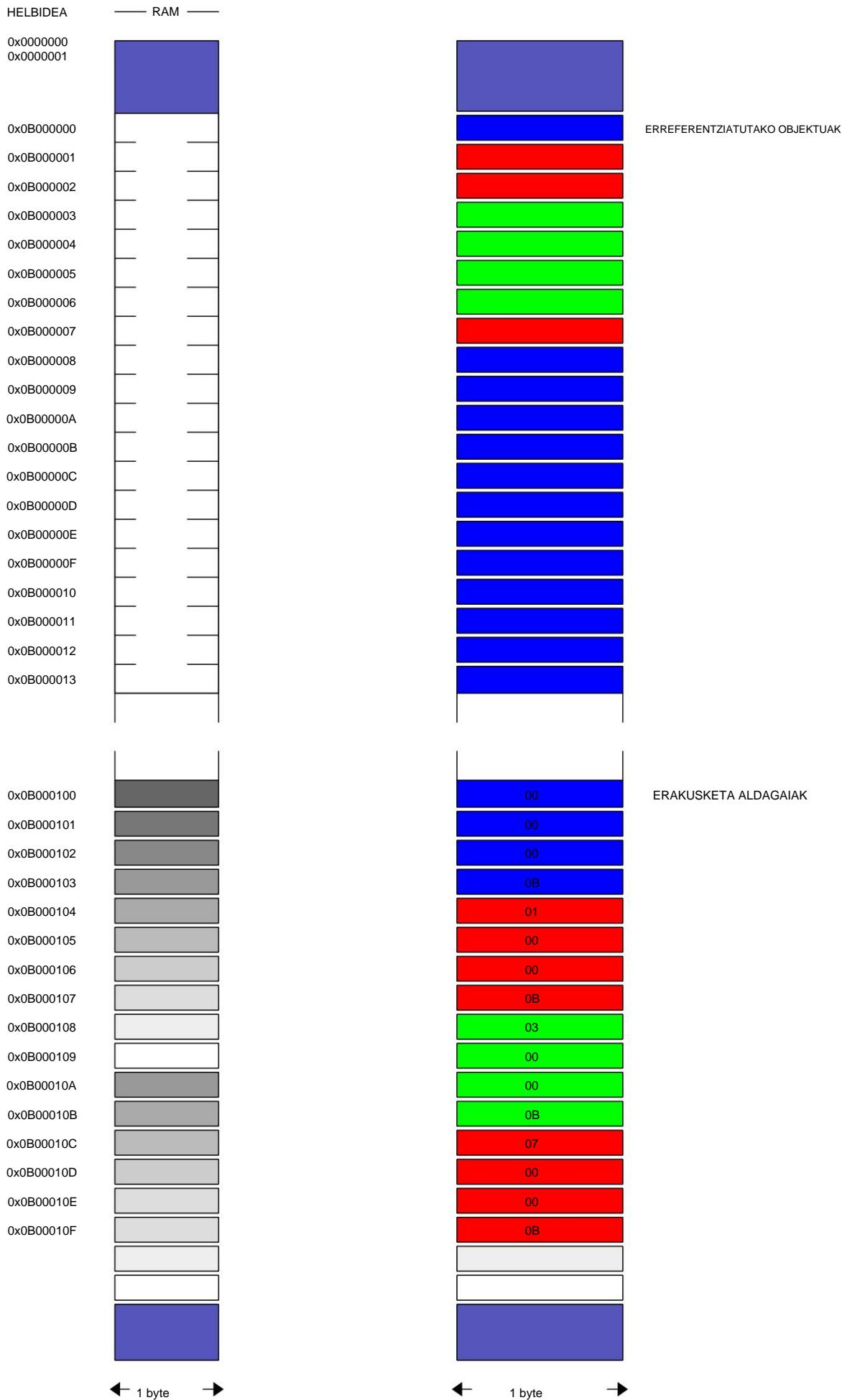


p: erakuslearen aldagaiaren identifikatzalea

x: erreferentziatutako objektuaren identifikatzalea, adibidez aldagai arrunt bat.

gezia: x objektura seinalatzen duen p erakuslea aldagaiaren hasieratzea

Erakusleen, objektuen eta erakusleen aldagaien adibideak



- 0x0B000100 helbidean (+0,+1,+2,+3 byteak) 0x0B000000 helbidea duka, 1 byte-ko objektu batera seinalatzen duena.
- 0x0B000104 helbideko erakuslearen aldagaia (+0,+1,+2,+3 byteak) 0x0B000001 helbidea duka, 2 byteko objektu batera seinalatzen duena.
- 0x0B000108 helbideko erakuslearen aldagaia (+0,+1,+2,+3 byteak) 0x0B000003 helbidea duka, 4 byteko objektu batera seinalatzen duena.

**EzkerrekoBalioa-EskuinekoBalioa**

- Esleipen-operadore batean (=) erreferentziatutako aldagai arrunt batek beste interpretazio bat du esleipen-operadorearen ezkerrean edo eskuinaldean badago:

ÿ x=y

ÿ x: ezkerreko aldagai arrunta x-ren memoria-helbide gisa interpretatzen da: x-ren ezkerreko balioa

ÿ y : eskuineko aldagai arrunta y : rightvalue-ren memoriako eduki gisa interpretatzen da  
eta

- Objektuaren edukia RightValue da
- Objektuaren erreferentzia LeftValue da
- Erakuslearen aldagai baten edukia erreferentziatutako objektuaren LeftValue da.

**35.3.4. Modulu Ilustratzailea**

```
/* Erakusleen hastapena.*/
```

```
#include <stdio.h>
#include <stdlib.h>
```

```
void main (void)
{
```

ÿ /\* Kontzeptua \*/

ÿ /\* Operadorearen helbidea \*/

```
ÿ int i, k, *p, *q; float
x, y, *r, *s; char c, d, *u,
*v; i = 10; k = 100; x =
3E-10f; y
= 3,1416;
c = 'A'; d = '@';
```

```
p = &i; q
= &k; r =
&x; s = &y;
```

```

u = &c; v
= &d;

printf("Sartu karaktere bat \n"); scanf("%c",&c);
printf("Irakurritako
karakterea %c \n da", c);

/*Zeharkako Eragilea*/

printf("Irakurritako karakterea %c \n da", *u); printf("i
aldagaiaren balioa %d edo %d \n", i, *p); printf("Plren balioa %f edo %f \n", y,
*s);

/*String aldagaia*/
/*Array*/
char string[]="Kaixo";

/*Erakuslea*/
char *agurra="Kaixo";
char **pt_agurra;

pt_greeting = &agurra;

irten(0); }
```

### 35.3.5. Adierazpena

Sintaxia: idatzi \*erakuslea\_aldagaia

```

int i, k, *p, *q; flotatu
x, y, *r, *s; char c, d,
*u, *v; i = 10; k = 100;
x =
3E-10f;
y = 3,1416;
c = 'A'; d =
'@';
```

\*p, \*q, etab... erakusle-aldagaien adierazpenak dira. Izartxoak EZ du eragiketarik egiten aldagaian, erakuslea MOTA adierazteko aurritzka baino ez da.

### 35.3.6. Operadorearen helbidea

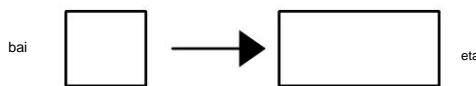
Ikurra &

```

p = &i;
q = &k;
```

```
r = &x;
s = &y;
u = &c;
v = &d;
scanf(&c);
scanf(u);
```

& operadoreak aldagaiaren LeftValue lortzen du eta erakusleak hasieratzeko erabiltzen da.



### 35.3.7. Bideratze- edo deserreferentzia-eragilea

Ikurra \*

Erakuslearen aldagai baten aurrikia: erreferentziatutako objektura sartzen da

```
printf(" i aldagaiaren balioa %d edo %d \n", i, *p); printf(" Piren
balioa %f edo %f \n", y, *s);
```

### 35.3.8. Adibidea

- Mota ezberdinako objektuak deklaratzea: osokoa, flotatzailea, kar
- Erakusle motako objektuak deklaratu eta hasieratu aurreko objektuekin
- Erakusleak grafikoki irudikatzea
  - ÿ Maila baxua: RAM memoria
  - ÿ Goi Maila: geziekin seinalatzen duten laukiak dituzten diagramak.

### 35.3.9. Erakusleen aplikazioak

- Array
  - ÿ Array Erakuslea
  - ÿ Erakusleen Aritmetika
- Katea literala
- Erakuslea Erakuslerantz
- String-era sarbidea
  - ÿ Array izena

- Ÿ Erakuslearen aldagaia
- Datuen egitura
  - Ÿ Izenen zerrenda (kateen erakusleen array)
- Ezaugarriak
  - Ÿ Argumentuak erreferentziaz pasa
  - Ÿ Erreferentzia bidez itzultzea.
- Linux shell-en lerroko komandoen argumentuak.

### Array Erakuslea

- Kontzeptua
  - Ÿ Array bat erakuslea eta elementuen zerrenda bat da. Erakusleak zerrendako lehen elementura seinalatzen du.
  - Ÿ Array bat sortzen denean, bi objektu sortzen dira
    - Ÿ Memoriaren esleipena ondokoa duten array-ko elementuak
    - Ÿ Arrayko lehen elementura seinalatzen duen erakuslea
- Adibidea
  - Ÿ Zenbakien array: `data_items: 3,67,34,222,45,75,54,34,44,33,22,11,66,0`
  - Ÿ Adierazi eta hasieratu
  - Ÿ Irakurketa
  - Ÿ Idaztea
- CONSTANT erakuslea
  - Ÿ EZIN DUZUE ERAKUSKLEAREN BALIOA ALDATU
  - Ÿ Aldatu erakuslea
- Adibidea:
  - Ÿ Character Array: `katea: H,o,I,a,\0`
  - Ÿ Deklaratu eta hasieratu `char string[]={H,o,I,a,\0};`
  - Ÿ Irakurketa
  - Ÿ Idaztea

### Erakusleen Aritmetika

- Indexatzea: lehen elementua GEHI i elementuaren posizioa
  - Ÿ `datu_elementuak + i`
- Erakusleen adierazpen aritmetikoen bidez matrizeko elementuen erreferentzia-adierazpenak aldatzea

### 35.3.10. Katea literala

- Bi faseko kontzeptua
  - Ÿ "Kaixo" izeneko array, zeinaren elementuak karaktere motakoak diren.



ÿ Hasieratu Array katearekin Kaixo



- Adibidea

ÿ Karaktere-matriz bat deklaratu eta hasieratu "Kaixo" kate literal batekin

ÿ char string[]="Kaixo";

ÿ Kate literala:

ÿ Karaktere katea

ÿ Komatxo bikoitzak

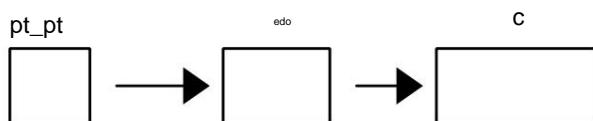
- Arrayak

ÿ Sartu deklaratutako array-ra: irakurtzen eta idazten

ÿ Sartu hasierako matrizean: irakurtzen eta idazten

ÿ Matrizeak kopiatu katea1=kate2 esleituta?

### 35.3.11. Erakuslea Erakuslea



- Adibidea

ÿ u pertsonaia seinalatzen du c

ÿ pt\_pt au seinalatzen du

### 35.3.12. Katearen aldagai

#### Array izena

- Karaktere array katea deklaratzen dut eta Kaixo katearekin hasieratzen dut: char string[]="Kaixo";

#### Erakuslearen aldagaia

- Agurra aldagaia deklaratzen dut eta **kate-** erakuslearekin hasieratzen dut  
    ÿ char \*\*agurra

#### 35.3.13. Ezaugarriak

- Argumentuak **Erreferentziaren** arabera pasa  
    ÿ Funtzio-parametroak erakusle-aldagai gisa deklaratu.
- **Erreferentzia bidez itzultzea.**  
    ÿ Adierazi itzuleraren balioa erakusle gisa.

## 36. kapitula. x87 FPU

### 36.1. x87 FPU

#### 36.1.1. Laburpen

- x87 arkitektura:

ÿ 1980

ÿ zenbaki errealekin eragiketa matematiko konplexuak egiten dituen instrukzio multzoa da nola kalkulatu tangentea, etab.

- x87 koprozesadorea edo x87 FPU (Float Point Unit):

ÿ x86 CPUtik independentea den prozesadorea da, x87 arkitektureko argibideak exekutatzeko.

- x87 erregistroak:

ÿ FPUnren barne-erregistroak dira. 8-mailako pila-egitura sakona ez zorrotza, ST (0) eta ST (7) bitartekoak.

Ez dira zuzenean irisgarriak, baina push, pop edo desplazamendu erlatiboarekin pilaren goiko alderatzen dira.

- FPU: koma mugikorreko eragiketak kalkulatzen espezializatutako prozesatzeko unitate zentralaren osagaia da, ALU-a RPG erregistroetan gordetako zenbaki osoekin egiten den modu berean.

- Datuen formatua:

ÿ doitasun bakarra, doitasun bikoitza eta 80 biteko doitasun bikoitz hedatua koma mugikorreko aritmetika bitarra IEEE 754 arauaren arabera

ÿ edo zenbaki oso bat baino gehiago 8, 16 edo 32 biteko erregistro berean.

- FP: Float Point: FPU pila-erregistroak, ST erregistroentzako izen berria.

- MMX: SIMD (Single Instruction Multiple Data) instrukzio multzoa Intelek diseinatu eta 1997an bere Pentium MMX mikropkopzesadoreetan sartu zen.

ÿ MMX-k FPUaren lehendik dauden zortzi FPR erregistroak berrerabilten ditu, beraz, ezin da erabili mms instrukzioekin eta fpu instrukzioekin aldi berean. 64 biteko MMX erregistroak zuzenean eskura daitezke pila arkitektura duten FPRak ez bezala.

ÿ Hala ere, x87 argibideak GCCren lehenetsiak dira IA32 koma mugikorreko kodea sortzen denean.

- SSE: Streaming SIMD Extensions (SSE) x87 azpimultzoa ez den SIMD argibide-multzoa da, Intelek seinale digitala MMX x86 arkitektura prozesatzeko, prozesatzeko eta grafikoetarako diseinatua. aplikazioetarako.

ÿ 1999an Pentium III-rekin hasi zen.

ÿ 128 biteko 16 erregistro berri gehitzen ditu XMM0-XMM15

ÿ XMM: SSE koma mugikorreko instrukzioek erregistro-multzo independente berri batean funtzionatzen dute (XMM erregistroak), eta MMX erregistroetan lan egiten duten zenbaki osoko instrukzio batzuk gehitzen ditu.

ÿ SSE2 Pentium 4-n (2000).

- AVX: Iuzapen bektorial aurreratuak

ÿ 256 biteko 16 erregistro gehitzen ditu: YMM0-YMM15

ÿ Lehen 128 biteko XMMrekin funtzionatzen zuten argibideek orain 128 biteko behekoekin funtzionatzen dute YMM pisua.

### 36.1.2. Erref

- Programazioa x87 Floating-Point Unitatearekin: Intel 1. liburukia 8-1
- Computer Systems: A Programmer's Perspective, 2/E (CS:APP2e) Randal E. Bryant eta David R. O'Hallaron, Carnegie Mellon Unibertsitatea

# 37. Kapitulua. Konputagailuen Egitura 2022: Lehen Teoria Partziala

• Abizenak:

• Izena:

• Taldea:

## 37.1. 1-6 gaietarako ariketak

1. Von Neumann arkitektura duen ordenagailu bat, 8 hariko datu-busa eta helbide-busa emanda  
4 hari.

to. Marraztu ordenagailuaren arkitekturaren diagrama, klasean ikusitako elementu nagusiak erakutsiz (0,5 puntu)

b. Honako erregistro hauetako bakoitzak izan behar dituen bitak definitzen ditu (0,4 pt):

i. Yo. PC ſ

ii. JOAN ſ

iii. ITSASO ſ

iv. MBR ſ

c. Kalkulatu memoria biltegiratzeko ahalmena Bytetan (0,5 pt)

d. Helbide-busa 8 harietara hedatuz eta memoria-taula honekin:

25. taula. Memoria

...	...
0x0A	KARGA 0x27
...	...
0x26	0x16
0x27	0x4A
0x28	0xF3



ŷ Aurreko ataletako erregistroak, gehi sistema-busa eta metagailua erabiliz, deskribatu PC=0x0A duen IAS makinan instrukzio-ziklo baten funtzionamenduaren faseak eta erregistro eta busen edukia. Horretarako, bete hurrengo taula behar adina zutabe erabiliz: (1,6 puntu)

26. Taula. Irakaskuntza Zikloa

	1	2	3	4
Deskribapena				
PCa				
JOAN				
ITSASOA				
MBR				
Datuen Busa				
Autobusen Kontrola				
Autobusaren norabidea				
Metagailua				

27. Taula. Irakaskuntza Zikloa

	5	6	7	8
Deskribapena				
Faseak				
PCa				
JOAN				
ITSASOA				
MBR				
Datuen Busa				
Autobusen Kontrola				
Autobusaren norabidea				
Metagailua				

2. Egin batuketa hamaseitarrez hiru zifrekin, liburu-liburuak erakutsiz, zeinurik gabeko zenbaki oso hauen batuketa eta emaitza hamaseimalean eta bitarrean erakutsi: 0xF7+0x2A (0,5 pt)

Eramanak -&gt;

Gehituz -&gt;

Gehituz -&gt;

Gehitzea -&gt;

3. Egin 1010010 – 110110 zeinurik gabeko zenbakien kenketa hau (0,5 pt)

minuend      ->

Subtrahend ->

Eramanak      ->

Kenketa      ->

4. Zifra kopuru **minimoarekin**, emaitza matematikoa zuzena izan dadin, egin eragiketak emandako zenbaki hauek: A: 0101011 eta B: 11011

to. Zeinu gabeko zenbaki osoak izatea: (1 pt)

Yo. Egin batura  $C = A + B$

.

.

.

.

.

.

ii. A, B eta C hamaseitar eta hamartarrez adierazten ditu

.

.

.

.

b. Zenbaki osoak 2 osagarrian (1 pt)

Yo. Egin batura  $C = A + B$

.

.

.

.

.

ii. A, B eta C hamaseitar eta hamartarrez adierazten ditu

.

.

.

.

c. Zenbaki osoak izatea zeinu-magnitudean (1 pt)

Yo. Arrazoia  $C = A + B$  baturaren emaitza lortzeko

.

.

.

.

.

.

ii. A, B eta C hamaseitar eta hamartarrez adierazten ditu

.

.

.

.

.

5. Egin  $0x26$  eta  $0x3C$  zenbaki naturalen biderketa bitarra (0,4 pt)

.

.

.

.

.

.

.

6. Ordenagailu batek balio hauek ditu gordeta:

28. taula. Memoria

ERREKOAK		MEMORIA	
Grabatu	Edukia	Helbidea	Edukia
EAX	87	87	01
EBX	02	88	07
ECX	8C	89	03
		8A	02
		8B	08
		8C	0F
		8D	24
		...	...
		94	32
		95	00

ÿ Instrukzio bakotzaren helbideratze-modua adierazten du eta horietako baterako EDX-n sartutako eragigaiaren balioa zehazten du helbideratze-modu hauekin: (0,3 pt bakotza)

	<b>Era</b>	<b>Merezi</b> Funtzionamendua
movb \$0x89, %edx		
movb %eax, %edx		
movb(%ecx,%ebx,4),%edx		
movb (%eax), %edx		
movb 0x88, %edx		
movb -3(%ecx), %edx		

7. 8 biteko Intel ordenagailu batean egindako azken eragiketa 2 osagarriko zenbaki hauen batura bida: 10001011 eta 10101101, **arra佐itu** zein den honako bandera hauen balioa (0,2 pt bakotza)

a. Gainetik bandera:

b. Eraman bandera:

c. Zero Bandera:

d. Seinalarearen bandera:

## 37.2. Mihiztadura Lengoaiaren Programazioa

1. Garatu programa **nagusia** osoa Intel x86 arkitekturako AT&T mihiztadura-lengoaien, hasieran  $n = 5$  balioarekin definitutako **n** (2 byte tamaina) aldagai oso bat 4 gehitzen diona eta emaitza beste **batura** aldagai batean gordetzen duena (4 byte). eta emaitza sistema eragilera itzuli. Gehitu oinarritzotzat jotzen dituzun 5 iruzkin programari.

# 38. kapitulua. Konputagailuen Egitura 2022: Lehen Praktika Partzialak

- Abizenak:
- Izena:
- Taldea:

## 38.1. Sarrera

- Azterketa eguna: 2022ko urriaren 28a, ostirala 08:00-10:00
- Tokia: Ikasgelak A307 (euskarra eta asteleheneko taldeak 19:00-21:00)-A308 (asteazkeneko taldea 15:00-17:00)-A015 (asteazkeneko taldea 17:00-19:00) :
- Azterketa programa bat garatzea eta GDB araztailea erabiltzean izango da, P1 eta P2 praktiketan ikusitako eta x86 mihiztadura lengoian programatzeko teoriako 6. gaia kontuan hartuta, azpierrutinak barne hartzen ez dituena. non ezinbesteko baldintza den 15 (CMP instrukzioa), 16 (CF/OF banderak) eta 17 (JLE, SETcc instrukzioak) ariketak eskuz eta haien programazioa muntaketa kodean, konpilazioan, exekuzioan eta arazketan egitea . Ariketaren helburua maila baxuko programak garatzeko eta horien arazketarako trebetasunak erakustea da.

## 38.2. Iturburu-modulu

- Hizkuntzan garatu beharreko programaren adierazpena **izan daitekeenaren** adibidea edo prototipoa muntatzailea
- Jarraian, iturburu-kodearen **zati bat** dago osatu eta arazketa egin behar duena ikasleak osatu gabe dagoelako eta akats edo akats batzuk izan ditzakeelako.
- Iruzkinak errespetatu behar dira, ondoren zer programatu behar den eta zer den adierazten baitute ebaluazioaren arrazoia.
- Sorburuko moduluak goiburu hau eduki behar du:

### Ordenagailuaren **Egitura** . 2022ko kurtsoa **1.** azterketa partziala .  
### AT&T mutua-lengoian **programatzea** x86 **32** biteko arkitekturarako **Programa**:  
**Abizena1\_Abizena2.s**, adibidez, Alen\_Urra.s **Programaren 1.** atala  
abizena1\_abizena2.s ( **3 puntu** ) **Adierazi** mezu-aldaagaia "m"  
karakterearekin hasieratuz  
### **Bihurtu** "m" karakterea "M" letra larri **letra** larriez letra  
xeheak baino **0x20** ASCII kodea txikiagoa dutela **jakinda** . **a" = 0x61** eta "**A" = 0x41** ASCII kodea  
### **GDB**  
araztaileari buruzko **galderak (4 puntu)**: **erantzunek**  
GDB komandoak eta **komando horiek** exekutatzearen  
emaitza izan behar dute . **a** inprimatu edukiak ..... print komandoarekin -> p  
**komandoarekin** (gdb): emaitza:  
###  
###  
###  
###  
###  
b) inprimatu edukiak.... kode hamaseimalean aztertu komandoarekin  
> X

```

#### komandoa (gdb):
#### emaitza:
####

#### c) inprimatu .....-ren helbidea kode hamaseimalean p komandoarekin
#### komandoa (gdb):
#### emaitza:
##

####

#### Programaren 2. atala abizena1_abizena2.s (3 puntu):
#### Deskribapena: programatu honako baldintza-algoritmoa
#### Logika positiboa: Z=EGIA=1 edo Z=GEZURRA=0
#### Z aldagaia DL erregistroarekin ezarriko da
#### Z aldagaiaren balioa itzultzen dio Sistema Eragileari
#### 1 byte-ko tamaina duten X eta Y aldagaien balioa -> X=-3 eta Y=-1
#### Z=EGIA XY produktu logikoak zenbaki bikoitia badu
#### Erabili algoritmoaren emaitzaren berri ematen duten mezu guztiek
bere exekuzioan. EZ aldatu edo gehitu mezurik.

# rodata section: irakurri soilik datuak -> irakurtzeko soilik aldagaia -> konstanteak
.atala .rodata
agurra: .string "Informatika Egitura ikastaroa 2022. UPNA urriak 28"
emaitza: .katea

# inprimatu agurra
    push $agurra
    dei egiten du

# Bihurtu letra minuskula maiuskulaz

# Biderketa eragiketa logikoa Z=XY

Inprimatu emaitza
dei egiten du

# Pasatu emaitza sistema eragilera

# Amaiera

```

### 38.3. Azterketaren metodologia

- Ezin da elektronikoki konektatu ez NUPeko sarera ez Unibertsitatetik kanpo.
- Edozein motatako informazioa duen pendrive bat erabil dezakezu: lantzeko gidoiak, oroitzapenak praktiken, oharren, erreferentzia azkarreko fitxak.
- Edozein material inprimatua erabil daiteke: oharak, oroitzapenak, liburuak, etab.
- Behin ordenagailua martxan jarrita, /home/ikaslea/examen\_parte\_I karpeta sortu beharko du ikasleak
- Ikasleak 10 minutu izango ditu azterketaren hasieran azterketaren laburpena deskargatzeko

MyAulariotik ȳ Baliabideak ȳ Praktikak ȳ **exam\_1partial.s** karpetan [/home/alumno/examen\\_parte\\_I](/home/alumno/examen_parte_I)

ȳ Ondoren, **exam\_1partial.s** fitxategiaren izena aldatuko duzu entregatu beharreko fitxategiaren izenaren adierazpenarekin, behean adierazten den moduan.

- Azterketa **3 ariketak osatzen dute.**
- Entregatuko den iturburu-programa BAT da eta programa bitarra ere BAT, baina programa bera bi aldiz.
  - ȳ 1. eta 2. ariketei dagokien lehen entrega **09:15ak** baino lehen egin behar da. myaulario bidez TASK **azterketa\_I zatia**
  - ȳ 1., 2. eta 3. ariketei dagokien bigarren entrega (lehen zatiaren jarraipena) 10:00ak baino lehen entregatu behar da **nire** postontzikit **TASK azterketa\_II. zatia**.
- Bi zatiak ordenagailuko karpeta ezberdinetan garatuko dira. Lehenengo zatia [/home/alumno/examen\\_parte\\_I](/home/alumno/examen_parte_I) karpetan garatuko da eta bigarren zatia [/home/alumno/examen\\_parte\\_II](/home/alumno/examen_parte_II) karpetan.
- Azterketaren hasieratik bi ataza irekiko dira azterketaren emaitza formatu digitalean entregatzeko denbora muga batekin ZIP formatuan fitxategi **bakarrean** ataza bakoitzean. Zeregin bakoitzean entregatu beharreko fitxategiak **IZENA BERDINA izango du**. Zeregin bakoitza amaitzeko 10 eta 5 minutu geratzen direnean jakinaraziko zaizu.
- Bi zeginetako bakoitzaren ZIP fitxategiak honako hauek izango ditu:
  - ȳ iturri-modulua: Abizena1\_Apellido2.s, pej, **Alen\_Urra.s**, ikasle bakoitzak bere abizenekin.
  - ȳ iturburu kodea eta galdera/erantzunak iruzkinetan.
  - ȳ modulu bitarra: Abizena1\_Apellido2, adibidez, **Alen\_Urra**, ikasle bakoitzak bere abizenekin.

ȳ iturburu kodea ~~etorriko azterketa moduluan~~ azterketa moduluan behar da ~~zatiak~~ zatiak.

- Iturburu-modulua
  - ȳ osatu eta arazketa egin behar da
  - ȳ programaren irakurketan eta interpretazioan lagungarritzat jotzen diren iruzkinak gehitu
  - ȳ erantzun iturburu-moduluan GDB araztaileari buruzko galderai.

## 38.4. Ebaluazioa

- Ebaluazio zati bat proban bertan egingo da, irakaslek izango dira ikasle bakoitzari lehenengo ataza **azterketa\_I zatia\_I** azterketaren aurretik eta ondoren galdetuko diotenak emaitzari buruz, iturburu-programan zein programa bitarraren exekuzioan sartuz. Ikaslek 10:15ak baino lehen lehen zatia amaitu badu, irakasleari deitu beharko dio ebaluatu dezan.

# 39. kapitulua. Konputagailuen Egitura 2022: Bigarren Praktika Partzialak

- Abizenak:
- Izena:
- Taldea:

## 39.1. Sarrera

- Azterketa eguna: 2023ko urtarrilaren 12a, osteguna 08:00-12:00

- Lekua:

## 39.2. Azterketaren metodologia

- Aukeratu bi ariketetatik bat eta bakarra: "azpirutinak" ariketa edo "lerroak" ariketa
  - "Azpi-errutinak" ariketa: Gehienezko nota 6 puntu. Errutina eta azpirrutina simple bat garatzea. Galderak araztearekin pilaren edukiaren gainean. Azterketa burutzeko denbora 45 minitu.
  - "Lerroak" ariketa: Gehienezko puntuazioa 10 puntu. Garatu programa bat x86 mihiztatzaile-lengoainan, BMP formatuan marra bat marrazten duena C hizkuntzan duen soluzioan oinarrituta eta konponbide osatugabeen mutua-kodean garatu beharreko kodeari buruzko iruzkinekin, beraz, irtenbidea pseudokodean oinarritzen da. Ezinbesteko baldintza: BMP praktikaren ezagutza zehatza, pilara sartzeko kodearekin jariotasuna (epilogoa, hitzurrea, testuingurua gordetzea, aldagai lokalak implementatzeari, etab.) eta akatsak konpilatzeko eta exekutatzeko garaian antzemateko gaitasuna. Adibide gisa, bmp\_linea.c eta pixel\_linea.c modulua eskuragarri dago nire ikasgelako praktika karpelan, BMP formatuan lerro zuzen baten irudia sortzen duena.
  - Behin ordenagailua martxan jarrita, ikasleak **/home/student/azpirutines** karpeta sortu behar du edo **/etxea/ikaslea/lerroak**
  - Ikasleak 10 minuto izango ditu azterketaren hasieran azterketaren laburpena deskargatzeko MiAulariotik ÿ Baliabideak ÿ Praktikak ÿ
    - ÿ "azpirutinas" motako azterketaren kasuan, soilik deskargatu **subroutines.c** fitxategia **/home/ikaslea/azpirutinas** karpeta
      - ÿ Ondoren, adierazpena duen **subroutines.c** fitxategiaren izena aldatuko da, behean adierazten den moduan entregatuko den fitxategi pertsonalizatu bakarraren izenarekin. s luzapena duen fitxategi BAT bakarrik entregatuko da.
    - ÿ "Lerroak" motako azterketaren kasuan, deskargatu **pixel\_linea.s**, **pixel\_linea.c** eta fitxategiak **bmp\_linea.c**
      - ÿ Ondoren, adierazpena duen **pixel\_linea.s** fitxategiaren izena aldatuko da behean adierazten den moduan entregatu beharreko fitxategi pertsonalizatu bakarraren izenarekin. s luzapena duen fitxategi BAT bakarrik entregatuko da.
- Azterketaren hasieratik bi ataza irekiko dira (aukera bakoitzeko bat) azterketaren emaitza formatu digitalean fitxategi bakarrean entregatzeko, **ZIP** formatuan. Zeregin bat **azpierrutina\_azterketa** izango da eta bestea **lerro\_azterketa**. Zeregin bakoitzean entregatu beharreko fitxategiak **IZENA BERDINA IZANGO DU**. Zeregin bakoitza amaitzeko 10 eta 5 minutu falta direnean jakinaraziko zaizu.
- Aurreko ataletan "lerroak" aukera emulatzen dute adierazpen batekin, baldin eta iturri-moduluak hauek izan ziren: bmp\_linea.c eta pixel\_linea.s. Azterketan moduluak desberdinak izango dira: bmp\_circulo.c eta pixel\_circulo.c. Beraz, bigarren aukera lerro zuzenaren ordez zirkulu bat marraztean izango da.
- **s** fitxategiak asm kodean iturburu-modulu edukiko du aukeratutako aukeraren arabera:
  - ÿ iturri-modulu: Abizena1\_Apellido2.s, pej, **Alen\_Urra.s**, ikasle bakoitzak bere abizenekin.

ŷ iturburu kodea eta GDB galdera/erantzunak iruzkinetan.

ŷ

Iturburu-kodea duen fitxategiaren izena oharren "Azterketa-fitxategien izendapena" eranskinean ageri diren izenetako bat izan **behar du**. IZEN GENERIKOAK EZ DA BALIOA AZTERKETA.s, pertsonalizatu egin behar da bildu beharreko 90 azterketetatik bakoitza bereizteko.

- Muntatzaileen iturburu-modulua:

ŷ osatu eta arazketa egin behar da.  
 ŷ Egokitzat jotzen diren iruzkinak gehitzeak programaren irakurketa eta interpretazioan laguntzen du  
 ŷ **erantzun** iturburu-moduluan GDB araztaileari buruzko galderai.

- Ezin da elektronikoki konektatu ez NUPeko sarera ez Unibertsitatetik kanpo.
- Edozein motatako informazioa duen pendrive bat erabil dezakezu: praktika-gidoiak, praktika-txostenak, oharrak, erreferentzia azkarreko orriak.
- Edozein material inprimatua erabil daiteke: oharrak, oroitzapenak, liburuak, etab.

### 39.3. "Lerroak" azterketa

- Azterketaren bigarren aukera egitea aukeratzen duten ikasleentzat, bmp grafikoetako "lerroak" izan dira myaulario/recursos/practicas-era bi iturburu-modulu igota: bmp\_linea.c eta pixel\_linea.c.  
 ŷ bmp\_linea.c-k pixel\_linea.c-en definitutako pixel\_generator\_linea() funtziari deitzen dio.
- Pixel\_linea.s mutua moduluak pixel\_linea.c moduluaren isla fidela izan behar du. -ren iruzkinetan mutua-modulua, C lengoaiaaren adierazpenak agertu behar dira, adibidez:

```
#dx=x2-x1;
neg x1 gehitu
x2,x1

#dy = y2-y1; neg
y1 gehitu
y2,y1

#i1=2*dy;

#i2=2*(dy-dx);

# while (x < xmax){ while: mov
dx,$eax

# bada (d<0)
mov dx,$eax

#bestela
bestela:
mov dx,$eax
```

```
endif:  
    mov dx,$eax
```

```
while_exit:  
    mov dx,$eax
```

- Pixels\_generator\_linea() funtziaren aldagai lokalak, hau da, i1,i2,dx,dy,d,x,y,xmax mihiztatzailean ere lokalak izan daitezten eta, beraz, pixels\_generator\_linea( funtziaren) markoa definitu behar dira eta ez fitxategiaren funtzio guztietan komuna den "ataleko datuak".
- Biltzar moduluak galdera/erantzun iruzkin hauek izan behar ditu:

```
# pixel_linea.o objektu-modulu sortzeko komandoa # response_1:
```

```
# bmp_linea modulu exekutagarria sortzeko komandoa #  
response_2:
```

```
# open debug session komandoa # answer_3: gdb --  
readnow
```

```
# (gdb) programa nagusira itzultzeko helbidea harrapatu bmp_linea # command_4: #  
response_4:
```

```
# (gdb) harrapatu pilaren lehen elementua pixel_line prologoaren ondoren # command_5: #  
response_5:
```

## 40. kapitula. Espedienteak izendatzea azterketa

Alen\_Urra  
Alonso\_Gomez  
Altamirano\_Trujillo  
Alvarez\_Alonso  
Andreu\_Mangado  
Apestegua\_Guillen  
Apestegua\_Vazquez  
Arrastio\_Peris  
Arriazu\_Muñoz  
Ayechu\_Garriz  
Azcona\_Furtado  
Aznarez\_Gil  
Baigorrotegui\_Gil  
Bayona\_Restrepo  
Bellera\_Alsina  
Calleja\_Pascual  
Cascan\_Ustarroz  
Cerezo\_Uriz  
Chacon\_Flores  
Cordido\_Perez  
Couceiro\_Eizaguirre  
Cuello\_Tejero  
de\_la\_Ossa\_Goñi  
Diaz\_Ochoa  
Dobromirova\_Karailieva  
Echenique\_Arizaleta  
Eguillor\_Pinillos  
Elcid\_Beperet  
Etxabe\_Gil  
Etxarri\_Martin\_de\_Vidales  
Ezponda\_Igea  
Fernandez\_Illera  
Fernandez\_Picorelli  
Flamarique\_Arellano  
Fortun\_Iñurrieta  
Gallo\_Anza  
Garatea\_Larrayoz  
Garcia\_Vilas  
Gil\_Gil  
Goicoechea\_Elio  
Goikoetxea\_Macua  
Gomez\_Ciganda  
Goñi\_Lara  
Granda\_Saritama  
Hualde\_Romero  
Huarte\_Urriza

Iribarren\_Ruiz  
Isturitz\_Sesma  
Jimenez\_Nadales  
Juanotena\_Ezkurra  
Juarez\_Jimenez  
Labat\_Garcia  
Labiano\_Garcia  
Lalana\_Morales  
Larrayoz\_Diaz  
Larrayoz\_Urra  
Latasa\_Sancha  
Liebana\_Revuelta  
Longas\_Saragüeta  
Lumbresas\_Corridor  
Martinez\_Arpon  
Martinez\_de\_Goñi  
Martinez\_de\_Morentin\_Beaumont  
Martinez\_Sesma  
Melendez\_Uriz  
Mellado\_Ilundain  
Molina\_Puyuelo  
Monreal\_Ayanz  
Moral\_Garcia  
Oroz\_Azcarate  
Orradre\_Berdusan  
Pascal\_Alegria  
Perez\_Alvarez  
Portuondo\_Varona  
Redrejo\_Fernandez  
Ripoll\_Baños  
Romero\_Sadaba  
Ruiz\_de\_Gopegui\_Rubio  
Saiz\_Larraz  
Santos\_Garzon  
Sanz\_Sanz  
Sola\_Alba  
Sola\_Bienzobas  
Solaegui\_Garralda  
Taberna\_Maceira  
Tellechea\_Zamanillo  
Turrillas\_Remiro  
Urroz\_Velasco  
Vadillo\_Navarro  
Yaniz\_Ibañez  
Yarhui\_Sarate  
Zamboran\_Maldonado  
Zheng  
Zulaika\_Irigoien

# 41. kapitula. Aurreko Ikastaroetako azterketak

## 41.1. 2018 urtea

### 41.1.1. Azaroa

1. Proba Partziala. 2018ko azaroaren 10a.

Informatika Gradua 2. maila. Konputagailuen Egitura.

Nafarroako Unibertsitate Publikoa.

Iraupena: 90 minutu.

Abizena:

ÿ

Idatzizko informazio mota guztiak erabil daitezke, hala nola praktiken txostenak, oharrak, erreferentzia-orriak, etab. Ezin da erabili gailu elektronikorik, hala nola, kalkulagailuak, telefonoak, ordenagailuak... Erantzunetan emaitzara heltzeko beharrezkoak diren garapen mota guztiak jaso behar dira.

1. Computer Institute Advanced Studies (IAS) von Neumann-ek:

to. (1 pt) 0x00-0xFF kenketa egiten duen eta emaitza "kenketa" izeneko aldagaien gordetzen duen programa bat garatu. Zein izango da "kenketa" aldagaiaren memoria-kokapenaren edukia?

b. (1 pt) Zein erlazio dago MAR, MBR eta PC hiru osagaien artean?

2. (1 pt) Zein da "Kaixo \n" sei karaktereko katearen kode digitala

3. (1 pt) 0123 eta 0777 zenbakiak zeinurik gabeko zenbakiak dira oinarri zortzitalean. Egin 0123+0777 batura zuzenean oinarri oktalean.

4. (1 pt) 0xABCD eta 0xEFED zenbakiak bi osagarrien zenbakiak dira. Egin kenketa 0xABCD-0xEFED zuzenean. Kalkulatu emaitzaren balioa.

5. (1 pt) Adierazi 6,25 zenbaki hamartarra IEEE-754 doitasun bikoitzeko formatuan.

6. Argibide-formatua:

ÿ Ordenagailu batek 256K hitz-ko memoria-unitatea du, 32 biteko bakoitza, bytez byte helbideragarria.

Instrukzio bat memoriako hitzetako batean gordetzen da. Instrukzioak lau eremuko formatua du: zeharkako bit bat, opkode bat, 64 erregistroetako bat zuzentzeako eragiketa-eremu bat eta memoria helbideak dituen eragigai-eremu bat.

to. (2 pt) Zenbat bit osatzen dute opcode eremua? Eta erregistro-eremuak? Eta helbide-eremuak?

b. (2 pt) Zenbat bit dira disko unitatearen helbide-busaren eta datu-busaren parte? Orioimena?

7. (2 pt) Azpírrutina batean, adierazi zer erlazio dagoen fotogramaren "marko-erakuslearen" artean. Azpírrutina eta itzulera helbidea gordetzen den memoria helbidea.

8. (3 pt) Osatu programaren iturburu-kodea erantsitako mutaia-lengoaian, garatutako algoritmoak zenbaki hamartar bat kode bitar bihurtzen duen hurrengo iturburu-moduluan erantsitako iruzkinak kontuan hartuta:

### Programa: convert\_decbins.s

### Deskribapena: 15 zenbaki natural hamartarra bitar bihurtzen du, ondoz ondoko zatiketaz 2z

```

### Kode bitarrak 32 biteko tamaina du
### gcc -m32 -g -nostartfiles -o convert_decbin convert_decbin.s
### --32 bezalako muntaia --gstabs convert_decbin.s -o convert_decbin.o
### estekatzalea -> Id -melf_i386 -o convert_decbin convert_decbin.o

## MAKROAK

## DATUAK

        .          # hamartar (4 byteko tamaina) kode bitar bihurtzeko
abendua: 32 bit

## bin-ek kodea alderantziz gordetzen du, bin[0]-k bit txikia gordetzen du
pisua.
bin:      .space 32 # 32 byteko array: kodearen bit bat gordetzen du byte bakoitzean
32 biteko bitarra.

zatitzalea: . # zatitzalea (1 byteko tamainakoa)

## JARRAIBIDEAK

## ECX hasieratzen dut zatitzailaren balioarekin

## hasieratu bin arrayaren indizea

## Dibidendua EAX-en kargatzen dut
# eax <-x

## EDXn dibidenduen zeinu-bitaz luzatzen dut
# Dibidendua beti da positiboa

## Ondoz ondoko zatiketak 2z zatidura 0 izan arte
begizta:
## idivl : [EDX:EAX] / source_operand
# EAX<-Kotientea{x/y}, EDX<-Honddarra{x/y}
# gorde gainerakoa (byte 1 tamainakoa) bin array-n
## Ezaugarri bit-a luzatzen dut edx-n
# Dibidendua beti da positiboa

## eguneratu matrizearen indizea

## Zatidura zerora iritsi den egiaztatzen dut begiztatik irteteko

## Kode bitarren bit kopurua itzultzen dut EBX-n

```

## Sistema eragilearen dei-kodea

## Errutina eten eta OSra deitzen dut

ÿ GDB araztearen komandoak erabiltzea

to. (2 pt) inprimatu "bin" array-aren edukia bi adierazpen ezberdinekin, hau erabiliz "arakatu" eta/edo "inprimatu" komandoak.

b. (2 pt) inprimatu bin arrayko lehen elementuaren edukia

c. (2 pt) inprimatu bin arrayaren azken elementuaren edukia

9. (2 pt) Sistema-deiak

ÿ Osatu "convert\_decbins" programa a inprimatzeko beharrezko kodearekin ongietorri mezua zuzeneko dei idazketa erabiliz.

41.2. 2017 urtea

Proba Partziala. 2017ko irailaren 22a.

Informatika Gradua 2. maila. Konputagailuen Egitura.

Nafarroako Unibertsitate Publikoa.

Iraupena: 30 minutu.

Abizena:

1. Von Neumann ereduan, zein da Kontrol Unitatearen funtzioa.

2. Zeintzuk dira Von Neumann makinen instrukzio-zikloaren fase desberdinak.

3. Bihurtu 291 zenbaki hamartarra oinarri zortzitalera.

4. Egin -18-21 eragiketa 2 osagarrian.

5. Zertan datza abstrakzio kontzeptua ordenagailu baten antolaketan.

6. Sum.ias programa implementatzen duen IAS makinarako mihiztadura-lengoaia garatu  $s=1+2$  algoritmoa.

Proba Partziala. 2017ko urriaren 10a.

Informatika Gradua 2. maila. Konputagailuen Egitura.

Nafarroako Unibertsitate Publikoa.

Iraupena: 30 minutu.

Abizena:

ÿ Idatzizko informazio mota guztiek erabil daitezke, hala nola praktiken txostenak, oharrak,

1. Osatu exa\_2017.s iturburu-modulua AT&T x86-32 mihiztadura hizkuntzan (6 puntu)

```

### Informatika Egitura 2017-18 ikasturtea. Ebaluazio proba 2017ko urriaren 10ean
###

### Helburuak:
###      Kudeatu sinatutako zenbaki osoen datuen kodeketa
###      Datu-egiturak: erakuslea eta array
###      Zeharkako eta indexatutako helbideratze moduak
###      asm hizkuntza x86-32

### Algoritmoa: zerrenda-matrizeak tamainako bost zenbaki oso negatibo ditu
bi byte,
### -5etik -1era, -5 zero posizioaren balioa izanik.
###      Kopiatu zerrenda-matrizearen edukia bufferera.
###      Bufferra zeharka sartzen da EAX erakusle aldagaiaren bidez
###      Sistema eragilera bidalitako irteera argumentuak izan behar du
###      zerrenda-matrizearen lehen balioa.

## MAKROAK

.equ LEN,      SYS_EXIT, 1 # Sistema eragilearen dei-kodea
.equ ## 5 # Array eta buffer-aren luzera

ALDAGAIAK: zerrenda eta buffer
.datuak

zerrenda: # Array hasieratutako datuekin HEXADECIMAL-en irudikatuta

-----

buffer: # Erreserbatu memoria hasieratu gabeko buffererako.

-----

## JARRAIBIDEAK
## Sarrera puntuak

-----

_hasi:
## hasieratu irteerako argumentua zero balioarekin

-----

## EAX erakuslearen aldagaia hasieratzen dut

-----

## Begizta hasieratzen dut iterazio kopuruarekin. Erabili makroak.

mov ,%esi
begizta:

-----
-----
```

```
----  
dec %esi jns loop  
## irteera  
  
mov    _ _ _,%eax  
  
int    ----  
.amaiera
```

ŷ Arazoak:

ŷ gdb komandoa bufferaren edukia bistaratzeko kopia amaitutakoan (2 pt):

ŷ (gdb)

ŷ Prest etiketak 0x00555438 helbidea adierazten badu, adierazi helbideen edukia (2 pt):

ŷ 0x0055543C:

ŷ 0x0055543D:

Proba arrunta. 2018ko abenduaren 7a.

Informatika Gradua 2. maila. Konputagailuen Egitura.  
Nafarroako Unibertsitate Publikoa.

Iraupena: 45 minuti.

#### 1. ZATIA (10 puntu)

- Iraupena: 20 minuti
- Titulazioa:

1. (3 pt) Zenbaki sinatu gabeko kenketa: 0x8000 - 0x7AFF ŷ eragiketak urtean egin behar dira  
HEXADECIMAL kodea esklusiboki

2. (3 pt) Sinatutako zenbakien kenketa: 0x8000 - 0x7AFF ŷ eragiketak urtean egin behar dira  
HEXADECIMAL kodea esklusiboki

3. (3 pt) Lotu esaldi bakarrean kontzeptuak: programa-kontagailua, datu-bidea, instrukzio-zikloa, sekuentziatzalea,  
mikroaginduak, unitate logiko aritmetikoa, mikroarkitektura, instrukzio-harrapaketa.

## 2. ZATIA (10 puntu)

- Iraupena: 25 minutu
- Titulazioa:

1. (6 puntu) Garatu iturburu-modulua string\_length.s AT&T x86-32 mihiztadura-lengoaian.

```
/*
```

Programa: kalkulatu iturburu-programan bertan "Kaixo" esaldiarekin hasieratutako karaktere-kate baten tamaina.

Algoritmoa: Implementatu begizta bat katearen amaierako karakterea aurkitu arte: '\0' Etiketak: Katearen erreferentzia katearen ikurra erabiliz egingo da.

Iruzkinak: iturburu-modulua goi-mailako lengoaietan soilik zentzua duten kode-blokeen bidez komentatu behar da, ez makina-instrukzio bat deskribatzen duten kode-lerroen bidez. \*/

```
## MACROS definizioa .equ SUCCESS,  
0 .equ SYS_EXIT, 1 .equ  
FIN_CAR, '\0'
```

- Galderak: (4 puntu)

ü Bi gdb komandoa kate helbidean gordetako objektuaren edukia bistaratzeko

ü (gdb)

ü (gdb)

ü gdb komandoa katearen amaierako karakterea soilik bistaratzeko.

ü (gdb)

ü Adierazi tresna-kate bat erabiliz aurreko iturburu-programa konpilatzeko beharrezkoak diren bi komandoak eskuliburua, gcc frontend-a erabili gabe.



KLUSTERAK:

ABIZENAK:

IZENA:

Proba arrunta. 2018ko abenduaren 7a.

Informatika Gradua 2. maila. Konputagailuen Egitura.

Nafarroako Unibertsitate Publikoa.

Iraupena: 50 minutu.

### 3. ZATIA (10 puntu)

- Iraupena: 50 minutu
- Titulazioa:

1. (2 pt) 6 argumentu eta aldagai lokal bat duen azpierrutina baterako deia batean, **CALL azpierrutinen** instrukzioaren instrukzio-zikloaren amaieran , pila-erakusleak 0xFFFFA0C helbidera seinalatzen du.

Kalkulatu:

ÿ Itzultzeko helbidea gordetzen den memoria helbidea

ÿ Aldagai lokalaren memoria helbidea

ÿ Azpirrutinaren 1. argumentuaren memoria helbidea

1. (4 pt) Tamaina duen ordenagailu baten CPUaren mikroarkitekturaren bloke-diagrama 16 biteko hitza erantsitako fitxako irudikoari dagokio. Aipatutako ordenagailuaren ISA AT&T x86-32 mnemotekniari eta sintaxiari dagokion mutua-lengoaia du. Makina kodea memoria nagusian kargatzen da, argibideen atalari dagokion iturburu-modulu, honakoa:

```
movw $0xF000,R0  
mugitu R0,R1  
gehitu R1,R0  
subw R1,R0
```

ÿ Kontrol-unitateko sekuentziatzalea 4 egoerako makina T0,T1,T2 eta T3 , Adierazi erantsitako taulan instrukzio-zikloaren egoera bakoitzean exekutatu beharreko mikroaginduak programako instrukzio bakoitzeko.

	T0	T1	T2	T3
<b>movw \$0xF000,R0</b>				
<b>mugitu R0,R1</b>				
<b>gehitu R1,R0</b>				
<b>subw R1,R0</b>				

## 1. (4 puntu) Memoria hierarkiko baten antolaketa

- ŷ Programa bat osatzeko prozesuan, hasierako edizio-fasetik programa memoria nagusiko prozesu batean kargatzera arte, tresna-kateak memoria-espazio desberdinak sortzen ditu itzulpen-prozesuaren modulu ezberdinetan.  
 Bete erantsitako taula sortutako espazio bakoitzaren ezaugarriekin.

Tresna	Programa	ren egitura ren espazioa <b>Memoria eta Helbideratzea</b> bai	Helbide motak	Kodearen kokapena
Edizioa	Iturria Modulua	Atalak eta Etiketak	Birtuala, Ez-lineala	Bigarren mailako memoria: Disko gogorra

- ŷ Cache kontrolatzailak memoria memoria hierarkiaren barruan nola egituratzen duen cachea eta RAM dinamikoa.

- ŷ Fisikoki, zertan datzan RAM memoria dinamikoko zelula bat.

- ŷ Datu-tasa bikoitzeko DDR aram memoria dinamikoak nola sinkronizatzen dituen datu-transferentziak sistema-busaren bidez.

## 1. (3 pt) I/O eragiketen mekanismoak

ÿ Marraztu teklatuaren tekla baten eta ordenagailuaren von Neumann arkitekturako oinarrizko unitateen artean beharrezkoa den HW-aren bloke-diagrama eten mekanismoa erabiliz datu-transferentzia egiteko.

ÿ Marraztu teklatuaren tekla baten eta ordenagailuaren von Neumann arkitekturako oinarrizko unitateen artean beharrezkoa den SWaren bloke-diagrama eten mekanismoa erabiliz datu-transferentzia egiteko.

## 42. kapitula. Mybook: Bideokonferentzia

### 42.1. Sarrera

- Informazioa

ÿ [https://miaulario.unavarra.es/access/content/group/b15fc60b-3c66-452d-a7d9-42ec8cdab3a1/CSIE\\_WEB/site\\_csie/zoom-gida/zoom-gida\\_es.html](https://miaulario.unavarra.es/access/content/group/b15fc60b-3c66-452d-a7d9-42ec8cdab3a1/CSIE_WEB/site_csie/zoom-gida/zoom-gida_es.html)

ÿ csie@unavarra.es posta elektronikoa

### 42.2. Zoomaren instalazioa

- Hizlariak ZOOM aplikazioa bere oinarrizko bertsioan instalatu behar du (doan)
- <https://zoom.us/>
- Izena ematea beharrezkoa da
- Ubuntu 18.0 bertsioan, bezeroa ezin da abiarazi Firefox arakatzailletik
- Deskargatu zoom\_amd64.deb paketea
- instalazio komandoa: dpkg -i zoom\_amd64.deb
- Ireki bezeroa: ./zoom

### 42.3. Zoom erabiltzailearen gida

#### 42.3.1. Ezarpena

- Probatu audioa

### 42.4. Bideokonferenzia saioa

- Nire kontutik ÿ saioa hasi ÿ gaia ÿ bideokonferenzia
- [https://miaulario.unavarra.es/access/content/group/b15fc60b-3c66-452d-a7d9-42ec8cdab3a1/CSIE\\_WEB/site\\_csie/zoom-gida/zoom-gida\\_es.html](https://miaulario.unavarra.es/access/content/group/b15fc60b-3c66-452d-a7d9-42ec8cdab3a1/CSIE_WEB/site_csie/zoom-gida/zoom-gida_es.html)

## IX Bibliografia

## Konputagailuen Arkitektura

- ÿ [WS\_es] William Stallings. Ordenagailuen antolaketa eta arkitektura. 7. edizioa, berrargitaratua Pearson Prentice Hall ISBN 8489660824, 9788489660823. 2006
- ÿ [WS\_eu] William Stallings. Ordenagailuen Antolakuntza eta Arkitektura: Performance for Designing. 9th Ed Upper Saddle River (NJ): Prentice Hall, 2013 ISBN 0-273-76919-7. 2012
- ÿ [Randal\_Bryant] Randal E. Bryant, David R. O'Hallaron. Sistema informatikoak: Programatzale batenak Perspektiba. Addison-Wesley. 2. Edizioa. 2010.
- ÿ [Patt\_Henn] David A. Patterson, John L. Hennessy. Ordenagailuen Antolaketa eta Diseinua. Hardware/Software Interfazea. Morgan Kaufmann. 2009. Unibertsitate gehienena Standard Book.
- ÿ [MoMano] Sistema informatikoen arkitektura, Morris Mano.

## x86

- ÿ [BANDERAK] [https://en.wikipedia.org/wiki/FLAGS\\_register](https://en.wikipedia.org/wiki/FLAGS_register)

## Muntaiaaren Programazioa

- ÿ [PGU] Programazioa oinarritik Jonathan Bartlett-ek editatua Dominick Bruno, Jr. Copyright © 2003 Jonathan Bartlett-ek argitaratua Bartlett argitaletxeak Broken Arrow, Oklahoma ISBN 0-9752838-4- 7

- ÿ [UPC] Milatzalea... baina oso erraza da: IA-32 (i386) (AT&T sintaxia)

- ÿ [ATT] Oracle AT&T hizkuntza

- ÿ [WikiBook] WikiBook Oharrak:x86 Muntaia: AT&T

- ÿ [pc\_asm] Paul Carter. PC Mihiztadura Lengoia. Doako sarbidea. 2006.

- ÿ [asm\_linux] Jeff Duntemann. Mihiztadura-lengoia urratsez urrats: Linux-ekin programatzea. Wiley Ed. 3. Edizioa. 2009.

- ÿ [NASM\_tuto] NASM tutorial tutorials point

- ÿ [NASM\_bristol] Oharrak Bristol Community College: NASM Programa

- ÿ [i386] i386 (32 biteko)

- ÿ [A64] amd64 (64 biteko): gomendagarria

- ÿ [IAL] Intel asm hizkuntza: Intel Vol 1 Oinarrizko Arkitektura Ofiziala

- ÿ [AMD] AMD ofiziala. 3. liburukia. 2.3 Erregistroen eta Datu Moten Laburpena

- ÿ [paul\_carter] Paul Carter PC-en muntaia-lengoia. Doako sarbidea. 2006.: Netwide Assembler NASM, Intel hizkuntza

- ÿ [j\_dunte] Jeff Duntemann. Mihiztadura-lengoia urratsez urrats: Linux-ekin programatzea. Wiley Ed. 3 Edizioa. 2009.

- ÿ [irvine] Kip R. Irvine. x86 prozesadoreetarako muntaia-lengoia. Pearson. 6. Edizioa. 2014.

## Behe-mailako programazio dokumentuak

- ÿ [ABI\_i386] ABI i386

- ÿ [ms\_call] MicroSoft-en dei-konbentzioa

C Programazio Lengoaia

ÿ [c\_king] KNKing C programazioa, ikuspegi moderno bat WW Norton 2. Ed. 2008.

Programak garatzeko tresnak

ÿ [I386]AS i386: sintaxia, mnemoteknia, erregistroa

ÿ [GNU] GNU Software Garapena

ÿ [GAS] GNU Assembler

ÿ [GDB] GDB arazketa

ÿ [GCC] GCC konpilatzailea

ÿ [CPP] Aurreprozesadorea cpp

ÿ [BiU] GNU tresnak binutils:as,ld,objdump,...

ÿ [VIM] Vim

ÿ [EMACS] Emacs

Artikuluak

ÿ [jvn] linuxvoice

ÿ [w\_uni] wisconsin unibertsitateko RTL

# X Glosarioa

## Lehen planoa

Dagokion definizioa (koskatuta).

## Bigarren terminoa

Dagokion definizioa (koskatua).

# Kolofoia XI

Liburu baten amaierako testua, bere ekoizpenari buruzko gertaerak deskribatzen dituena.