

Ordenagailuen egitura (240306)

Candido Aramburu

2022-09-05

Edukien taula

I Instruction Set Architecture (ISA): von Neumann ordenagailua, datuak, argibideak, programazioa	1
1. Konputagailuen Egituraren Sarrera	bi
1.1. Sarrera	bi
1.2. makina baten arkitektura	bi
1.3. Arkitektura HW ikuspegitik	3
1.4. Programazio Lengoaiak eta Makina Lengoaia	7
1.5. Software/Hardware interfazea	9
1.6. Agenda	10
1.6.1. Oinarrizko Bibliografia	hamakia
1.6.2. Bibliografia Osagarria	13
1.7. Irakasleak	13
1.8. Informatika Gradua	14
1.9. Egutegiak	14
1.10. Problemak ebazteko ariketak	16
1.11. Praktikak	17
1.11.1. Oroitzapenak	17
1.12. Baliabide informatikoak	17
1.12.1. UPNA	17
1.12.2. Lanpostuak: 32 eta 64 bit	18
1.12.3. Izena eman	19
1.13. Praktika Taldeak	19
1.14. Metodologia	hogei
1.14.1. Kredituen banaketa	hogea bat
1.14.2. Praktiken kredituen banaketa	hogea bat
1.15. Ebaluazioa	hogea bat
1.16. Azterketak	22
2. Von Neumann arkitektura	23
2.1. Von Neumann Arkitektura	23
2.1.1. Ikastaroa	23
2.1.2. Testuinguru historikoa	23
2.2. Institute Advanced Machine (IAS): Arkitektura	25
2.2.1. Erreferentzia	25
2.2.2. Sum1toN Programaren adibidea	25
23. IAS ordenagailuaren egitura	26
2.3.1. Moduluak	26
2.3.2. Prozesatzeko Unitate Zentrala (CPU)	28
2.3.3. Oroitzapenak	29

2.3.4. Autobusa	31
2.3.5. Sarrera Irteera (I/O).	32
2.3.6. Irakaskuntza Zikloaren Animazioa	32
2.4. ISA: IAS Machine Instruction Set Architecture.	33
2.4.1. IAS Ordenagailuaren Argibideak eta Datuen Formatua	33
2.4.2. ISA errepertorioa	3. 4
2.4.3. ISA interfazea	37
2.5. Programazioa IAS Batzar Lengoain	38
2.5.1. Batzar Lengoain Programa bat Garatzeko Estrategia	38
2.5.2. 1. adibidea: sum1toN.ias.	39
2.5.3. IASSim hizkuntzako programen adibideak	44
2.6. IAS Makinaren Eragiketa: Datuen Bidea	44
2.7. Ondorioak	Lau. Bost
3. Datuen irudikapena	46
3.1. Agenda	46
3.2. Helburua	46
3.3. Datuak eta argibideak: kode bitarra	46
3.4. Bit, Byte, Word	46
3.5. Zenbaki Osoak	47
3.5.1. Oinarri hamartarra	47
3.5.2. Oinarri bitarra	47
3.5.3. Bihurketa hamartar-bitarra	48
3.5.4. Oinarri octala	48
3.5.5. Kalkulagailua	49
3.5.6. pitoia	49
3.5.7. Zenbaki oso sinatutakoak.	49
3.6. Zenbaki errealkak	52
3.6.1. Koma finkoa	52
3.6.2. Koma flotagarria	53
3.7. Karaktere mota	56
3.7.1. ASCII	56
3.7.2. pitoia	58
3.7.3. Unicode UTF-8	58
3.8. ISO-8859-1	61
3.8.1. Programazioa C.-n	61
3.8.2. Beste batzuk	61
4. Eragiketa aritmetikoak eta logikoak	62
4.1. Agenda	62
4.2. Helburua	62
4.3. Sarrera	62
4.4. Aritmetika Binarioa	62

4.4.1. 2. moduluko batuketa (bitarra) bitar hutsean (ZENBAKIA NATURALA)	62
4.4.2. 2 modulo (bitarra) kendu bitar hutsean	64
4.4.3. Batuketa/Kenketa 2. moduluan (bitarra) 2ren osagarrian 4.4.4.	64
Batketa 16. moduluan (hamasezimala)	66
4.4.5. Kenketa 16. moduluan (Hamaseitarra)	67
4.4.6. C -ko aldagai motak	68
4.5. Eragiketa logikoak	68
4.5.1. BITWISE operadoreak	68
4.6. Biderketa.	69
4.7. Programazioa	69
4.7.1. funtzio matematikoak	69
4.7.2. Aplikazioa.	70
4.8. Hardwarea	70
4.8.1. Zirkuitu digitalak.	70
4.8.2. Unitate Logiko Aritmetikoa (ALU).	70
4.8.3. EFLAG bandera erregistroa	71
4.8.4. Float Point Unitatea-FPU.	73
5. Argibideen irudikapena.	74
5.1. Agenda	74
5.1.1. Bibliografia.	74
5.2. Helburuak.	74
5.2.1. Baldintzak	74
5.3. Sarrera	74
5.3.1. Makina-lengoaia eta formatu bitarra.	74
5.3.2. makina-lengoaia eta muntaia-lengoaia	75
5.3.3. Goi Mailako Hizkuntzak	75
5.4. Arkitektura	76
5.4.1. Testuingurua	76
5.4.2. Instruction Set Architecture (ISA)	77
5.5. x86 arkitektura duten Intel prozesadoreak	77
5.5.1. Nomenklatura	77
5.6. Konputagailuen Egitura.	79
5.6.1. Arkitektura	79
5.6.2. CPUak.	79
5.6.3. Memoria	81
5.7. Makina baten instrukzio-elementuak.	83
5.7.1. Helbide implizituak	84
5.7.2. Operando arkitektura motak: adibideak	84
5.8. Argibideen irudikapena muntaia-lengoaian (ASM) egiteko ordenagailuak oro har	86
5.8.1. Makina Lengoaia Bitarra.	86

5.9. Eragileak: Helbideratzeko moduak	86
5.9.1. Kokapena.	86
5.9.2. Instrukzio-zikloan erreferentziatutako helbideak	87
5.9.3. Hizkuntza Orokor baterako Helbidea	87
5.10. Eragiketak.	89
5.10.1. Eragiketa-kodeak	89
5.10.2. Eragiketa motak	89
6. Programazioa Mihiztadura Lengoainan (x86): Mihiztatzaile lengoaien oinarrizko eraikuntzak maila altua.	91
6.1. Agenda	91
6.2. Sarrera	91
6.2.1. Helburuak.	91
6.2.2. Baldintzak	91
6.3. Konputagailuen Egitura.	91
6.3.1. CPUak.	91
6.3.2. Memoria	92
6.3.3. Memoria Nagusia	92
6.3.4. CPUaren barneko erregistroak	92
6.4. Intel hizkuntza versus AT&T hizkuntza	95
6.4.1. i386/amd64 arkitekturaren mutuaia-lengoaiak	95
6.4.2. INTEL hizkuntzan jarraibideen sintaxia 6.4.3. Muntaketa- prozesuen itzultzialeak	97
6.4.4. Makina Kodea	98
6.4.5. "Aso" muntatzalea.	99
6.5. Datuen irudikapena mutuaia-lengoainan (ASM) arkitekturarako i386/amd64	100
6.5.1. Datu motak	100
6.5.2. Operandoaren tamaina x86	101
6.5.3. Byte-lerrokatzea: Big-Little Endian.	102
6.6. Eragileak: Helbideratzeko moduak.	104
6.6.1. Kokapena.	104
6.6.2. Helbideratzeko moduak	104
6.7. Argibide multzoa: Eragiketak.	106
6.7.1. Erreferentzia eskuliburuak	106
6.7.2. TRANSFERENTZIA	106
6.7.3. ARITMETIKA	108
6.7.4. LOGIKOA	109
6.7.5. DENBERAK	109
6.7.6. JAUZIAK (orokorra)	110
6.7.7. JAUZTUAK Zeinurik gabe (Kardinal) JAUZTIK Signuarekin (Osokoa).	110
6.7.8. BANDERAK (ODITSZAPC).	111

6.7.9. Atzizkiak	111
6.7.10. Eragiketa-kodeak	111
6.8. Oinarrizko Mnemoteknia (Azalduta)	112
6.8.1. Eragiketa aritmetikoak	112
6.8.2. Baldintzapeko tratamendua	113
6.8.3. jauziak	114
6.8.4. Desplazamendua eta biraketa	114
6.8.5. Aldatu Endianess	114
6.9. Argibide-formatua: ISA Intel x86-64	115
6.10. Azpirrutinak	115
6.10.1. Sarrera	115
6.10.2. C hizkuntza: adierazpen-funtzioa	115
6.10.3. Funtzioa Habiaratzea	117
6.10.4. Pila/Markoa	117
6.10.5. Azpirrutinaren definizioa	117
6.10.6. Kontserbatzeko Erregistroak	118
6.10.7. Azpirrutilen argumentuak	119
6.10.8. Deitu azpierrutinara	119
6.10.9. Azpirrutilatik itzulera	120
6.10.10. Bateria egoera	120
6.11. Sistema eragilera deiak	121
6.11.1. Sarrera	122
6.11.2. Adibideak	123
6.12. Bibliografiak	123
VIII Eranskinak	124
7. IASSim muntatzaile programak	125
7.1. 1. adibidea: sum1toN.ias	125
7.1.1. iassim mutaia hizkuntza	125
7.1.2. 2. adibidea: Produktua/Kozientzia	127
7.1.3. 3. adibidea: Bektoreak	130
8. IASSim simulagailua	135
8.1. Java Makina Birtuala JVM	135
8.2. IAS simulagailua	135
8.3. Simulazioa/Arazketa	136
9. Mihiztadura hizkuntzak	138
9.1. Intel x86 / AMD 64	138
9.1.1. Kaixo Mundua	138
9.1.2. Muntatzaileen Programazioa	139
9.1.3. Zenbaki errealkak	139
9.1.4. Eztabaida zergatik ASM AT&T	140
9.1.5. Denetarikoa	140

9.2. Motorola 68000	141
9.2.1. Kaixo Mundua	141
9.2.2. ISA	141
9.3. MIPS	143
9.3.1. ISA	143
9.4. BESOA	144
9.4.1. Kaixo Mundua	144
9.4.2. ISA	144
10. Sum1toN programazio-lengoaiak	145
10.1. Sum1toN-rako beste hizkuntza batzuk	145
11.RTL Erregistro Transferentzia Hizkuntza	151
11.1. RTL hizkuntza	151
11.1.1. Sarrera	151
11.1.2. Erregistroak	151
11.1.3. Sinboloak	153
11.1.4. RTL esaldiak	154
11.1.5. RTL adibideak adierazpen aritmetiko-logikoenkin	155
12. Argibide-formatua: ISA Intel x86-64	156
12.1. Argibide-formatua: ISA Intel x86-64	156
12.1.1. Adibidea subq \$16,%rsp	156
12.1.2. Beste x86-32	157
13. x87 FPUak	158
13.1. x87 FPUak	158
13.1.1. Laburpen	158
13.1.2. Erref	159
14. Pila	160
14.1. Kontzeptua	160
14.2. Zabalera	160
14.3. Markoa: marko-erakuslea eta pila-erakuslea	161
14.4. Push-Pop muntatzailearen jarraibideak	162
14.4.1. Dei habiara	163
15. Sistema eragilera deiak	164
15.1. Sarrera	164
15.2. Deitu eskuliburuak	165
15.3. ZEHARKAKO deia	165
15.4. ZUZENEKO deia	165
15.4.1. Dei zuzeneko argudioak	166
15.4.2. Zuzeneko dei-kodeak	166
15.5. Adibideak: C hizkuntza	166
15.6. Adibideak: ZEHARKAKO ASM	167
15.7. Adibideak: ZUZENEKO ASM	167

15.8. Komando lerroa	168
15.8.1. Prozesua	168
15.8.2. StackInitialization	169
15.8.3. Errutina nagusia Itzuliarekin	170
15.8.4. Ariketak: sum_line_com.s ,maximum_line_com.s	171
16. C Programazio Lengoaia	173
16.1. Sarrera	173
16.2. casting	173
16.2.1. Kontzeptua	173
16.2.2. Adibidea	173
16.3. Erakuslea	173
16.3.1. Erreferentziak	173
16.3.2. Sarrera	173
16.3.3. Kontzeptua	174
16.3.4. Modulu Ilustratzailea	178
16.3.5. Adierazpena	179
16.3.6. Helbide-eragilea	180
16.3.7. Orientazio edo Deserreferentzia operadorea	180
16.3.8. Adibidea	180
16.3.9. Erakusleen aplikazioak	181
3.16.10. Katea Literal	182
3.16.11. Erakuslea Erakuslea	182
3.16.12. StringVariable	183
3.16.13. Funtzioak	183
17. Praktikak eranskina	184
17.1. Praktikak	184
17.1.1. Dokumentazioa: gidoiak, bibliografia, oharrak	184
17.1.2. Garapen Plataforma	184
17.1.3. Txostenaren dokumentua: edukia eta formatua	186
17.1.4. Ebaluazioa	187
17.1.5. Programazioa	187
17.1.6. Konpilazioa	188
17.1.7. Ohiko akatsak	191
18. amd64 arkitektura	192
18.1. Iturburu-modulua: sum1toN.s	192
19. Aurreko Ikastaroetako azterketak	193
19.1. 2018 urtea	193
19.1.1. azaroa	193
19.2. 2017 urtea	195
20. Miaulario: Bideokonferentzia	205
20.1. Sarrera	205

20.2. Zoomaren instalazioa	205
20.3. Zoomaren erabiltzailearen gida.	205
20.3.1. Ezarpenak	205
20.4. Bideokonferentzia saioa.	205
IX Bibliografia	206
X Glosarioa.	208
Kolofoia XI.	209

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

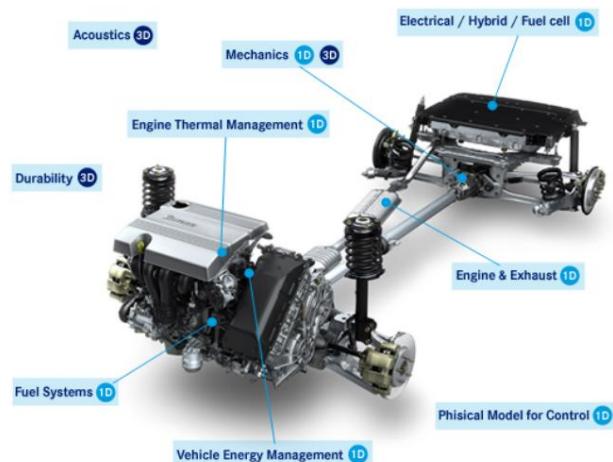
1. Kapitulua. Konputagailuen Egituraren Sarrera

1.1. Sarrera

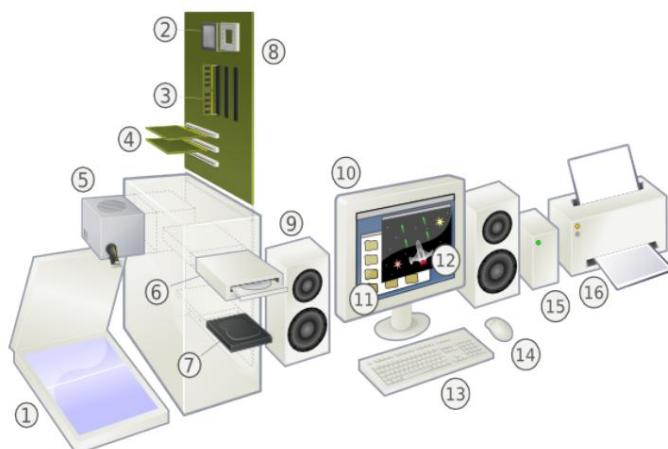
- **Informatika Ingeniaritzako** Graduko Konputagailuen Egitura (240306) irakasgaiaren helburua. Nafarroako Unibertsitate Publikoaren [<http://www.unavarra.es/ets-industrialesytelecos/estudios/grado/grado-en-ingeneria-informatica/presentacion?submenu=yes>] arkitekturaren hastapeneko unibertsitate-ikastaroa izango da. ordenagailuak, haien oinarrizko osagaiak (prozesadorea, memoria eta sarrera/irteera modulua) eta maila baxuko programazioa x86 mihiatzatzaile lengoian aztertuz, software garatzeko tresnak erabiliz, hala nola konpilatzailea, arazketa, etab.

1.2. makina baten arkitektura

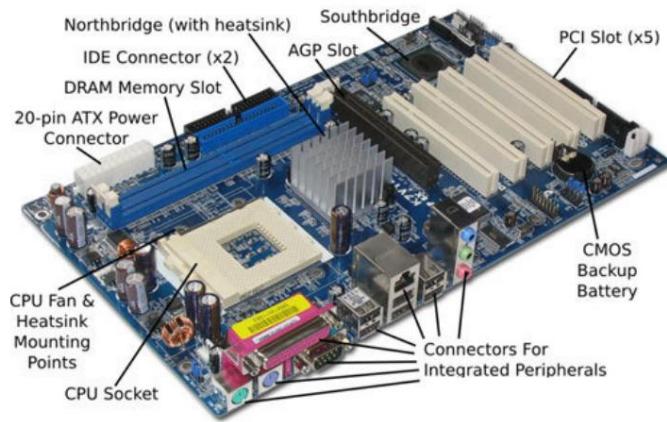
- Arkitektura: Antolaketa, Egitura: Zer, Nola, Ezarpena (teknologia)



1. Irudia Autoaren Egitura



2. Irudia. Ordenagailu pertsonala



3. Irudia. Plaka

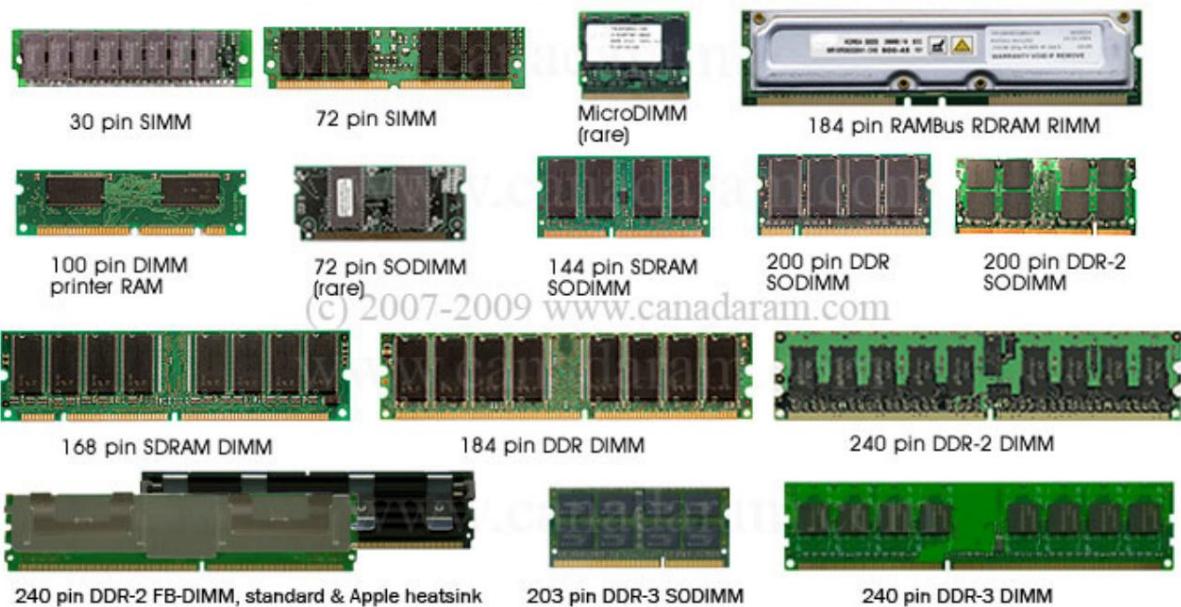
1.3. Arkitektura HW ikuspegitik

- 4 Oinarrizko Modulu: CPU-MEMORIA-I/O KONTROLATZAILEAK [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



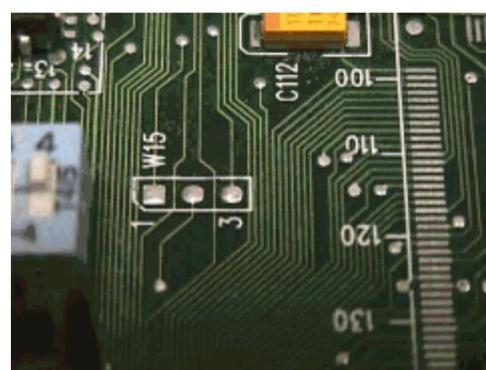
6. Irudia. Periferikoak: Semiconductor Solid State Drive (SSD) memoria



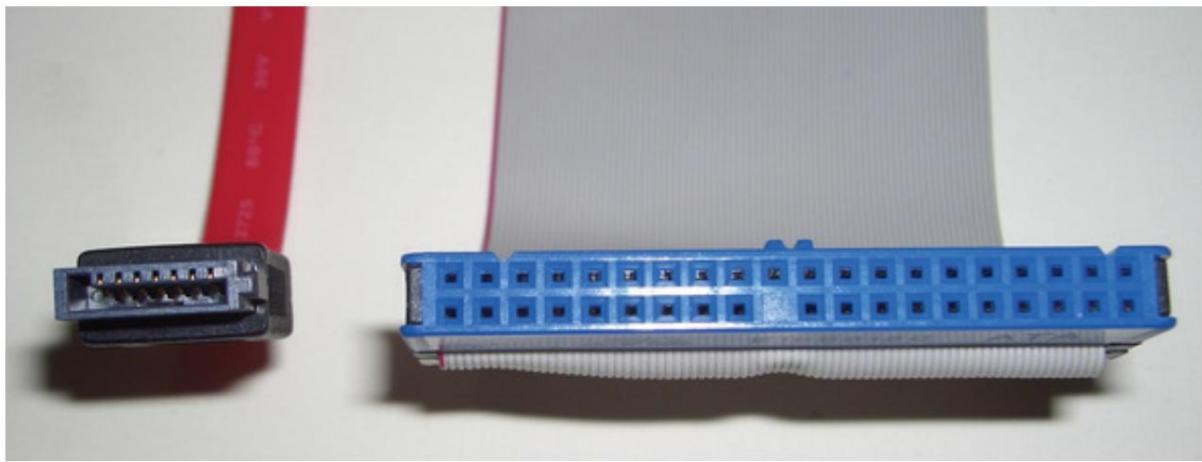
7. Irudia. Periferikoak: Seagate Disko gogorra



8. Irudia. Periferikoak: M2M Memoria

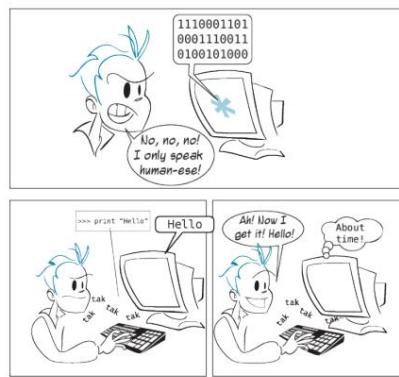


9. Irudia Plaka-Busa



10. Irudia Bus kableatua

1.4. Programazio Lengoaiak eta Makina Lengoaia



11. Irudia Makina Lengoaia: Bitarra

- Pascal hizkuntza

```
Hello_world programma ;  
hasiÿ writeln('Kaixo  
mundua')  
amaiera.
```

- L. Machine-Intel x86 Binary

```
# [2] hasi  
000000000004001a0 <PASCALMAIN>:  
4001a0: 01010101  
ÿ 4001a1: 01001000 10001001 11100101  
ÿ 4001a4: 01001000 10000011 11101100 00010000ÿ 4001a8:  
01001000 10001001 01011101 11111000ÿ 4001ac: 11101000  
10110111 00111110 00000001 00000000 # [3] writeln('kaixo mundua')ÿ  
4001b1: 11101000 11001010 10010011 00000001 00000000ÿ 4001b6:  
01001000 10001001 10100011ÿ 4001b9: 01001000 10001001 11011110  
  
ÿ 4001bc: 01001000 10111010 11000000 11110110 01100001 00000000 00000000
```

```

ÿ 4001c3: 00000000 00000000 00000000
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
00000000004001a0 <PASCALMAIN>:
ÿ 4001a0: 55ÿ                                bultzatu %rbp
4001a1: 48 89 e5 4001a4:                   mov    %rsp,%rbp
48 83 ec 10                                azpi   $0x10,%rsp
ÿ 4001a8: 48 89 5d f8                      mov    %rbx,-0x8(%rbp)
ÿ 4001ac: e8 b7 3e 01 00 # [3]             callq 414068 <FPC_INITIALIZEUNITS>
writeln('kaixo mundua')
ÿ 4001b1: e8 ca 93 01 00                  callq 419580 <fpc_get_output>
ÿ 4001b6: 48 89 c3                        mov    %rax,%rbx
ÿ 4001b9: 48 89 4001bc :                  mov    %rbx,%rsi
48 ba c0 f6 61 00 00ÿ 4001c3: 00 00 00      movabs $0x61f6c0,%rdx

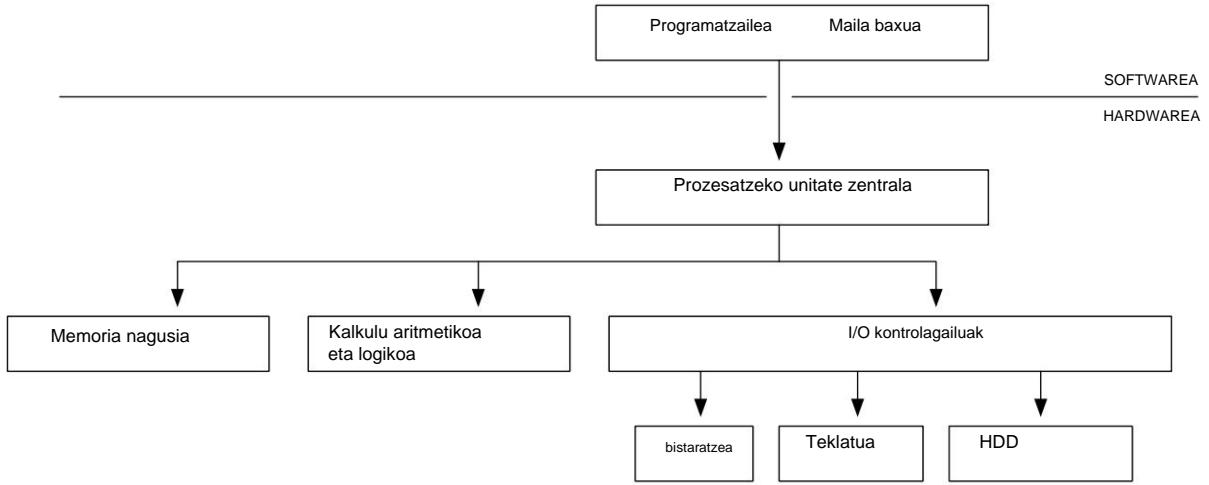
ÿ 4001c6: bf 00 00 00 00ÿ 4001cb:          mov    $0x0,%edi
e8 78 96 01 00ÿ 4001d0: e8 13 3d          callq 419848 <FPC_WRITE_TEXT_SHORTSTR>
01 00ÿ 4001d5: 48 89 df                    callq 413ee8 <FPC_IOCHECK>
                                         mov    %rbx,%rdi
ÿ 4001d8: e8 73 95 01 00ÿ 4001dd:          callq 419750 <fpc writeln_end>
e8 06 3d 01 00ÿ 4001e2: e8 99 42          callq 413ee8 <FPC_IOCHECK>
01 00ÿ 4001e7: 48 8b 5d f8                  callq 414480 <FPC_DO_EXIT>
                                         mov    -0x8(%rbp),%rbx
ÿ 4001eb: c9                                utzi q
4001ec : c3                                retq

```

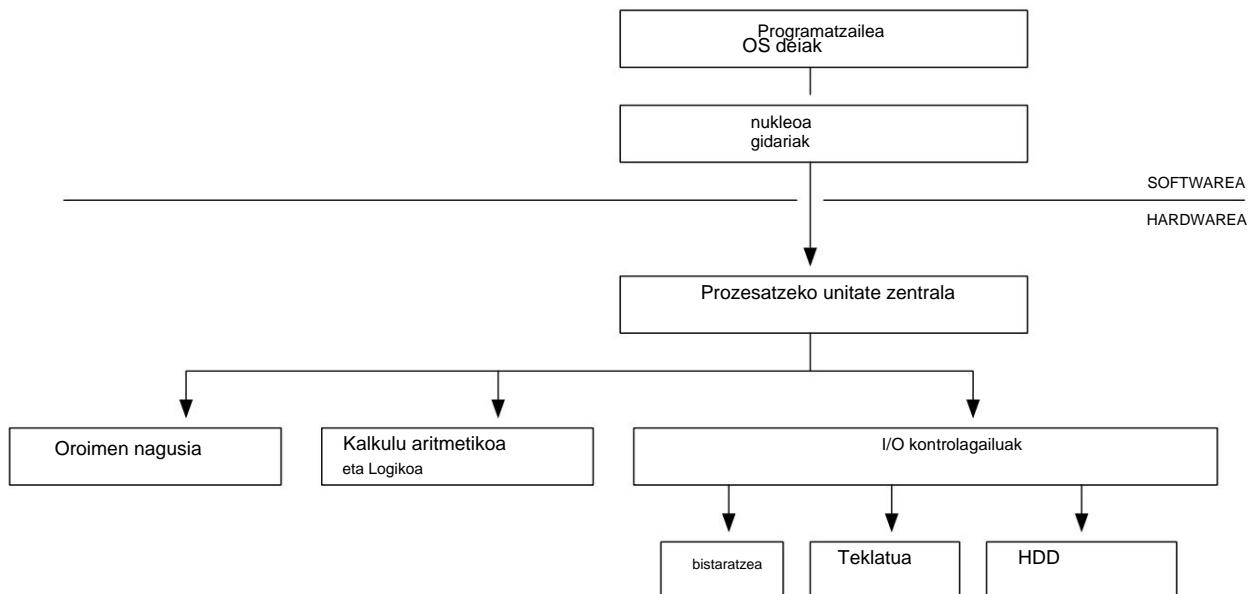
- Giza programazio lengoaia (testua) vs Makina programazio lengoaia (bitarra)
- Itzuli ÿ Konpilatu
- Interpretatu
- Programazio-lengoaiak: javascript, ruby, python, java, bash, C, assembly
 - ÿ Abstrakzio mailak: gizakia/makina

1.5. Software/Hardware interfazea

- Programatzailak hardware-makinarekin duen interakzio zuzena ѕ Programazioa maila baxuko lengoia batean (muntatzailea, C)



- Maila baxuko programazioa
 - ÿ Iturburu-modulua: C lengoia edo Mihiztadura lengoia
 - ÿ Objektu-modulua: Makina-lengoia ѕ Moduluaren konpilazioa, muntaketa eta lotura Letra-tipoa.
 - ÿ Bare Metal Sistemak: Ez dago Sistema Eragilerik. Zuzenean programatzen dugu Hardwarea.
- Programatzailak makinaren hardwarearekin duen zeharkako elkarrekintza. Programatzaila Sistema Eragilearekin (OS) elkarreragin egiten du
 - ÿ Sistema Eragilearen (OS) Kernelaren (nukleoa, adibidez linux) makinarekin (adib. Intel x86)



- Sistema Eragilea duten sistemak (adib. GNU/linux): Programazioan liburutegiak erabiltzen ditugu sistema eragilearen bidez hardwarera sartzen direnak. Adibidez printf() funtzioa, liburutegitik libc, konpilatzen denean, kontrolaztalea deitzen duen sistema eragilearen write() funtziora itzultzen da (sistema eragilean) pantailako txartel grafikoa. Sistema eragilea lortzen da "abstraktu" makinaren HW fisikoa eta asko errazten du programazioa behar ez izateagatik ordenagailuaren funtzionamendu fisikoa ezagutzea.

1.6. Ikastaroa

- Ordenagailuen Web Egitura [<http://www.unavarra.es/ficha-asignaturaDOA?languageId=100000&codPlan=240&codAsig=240306>]

Ikastaroa

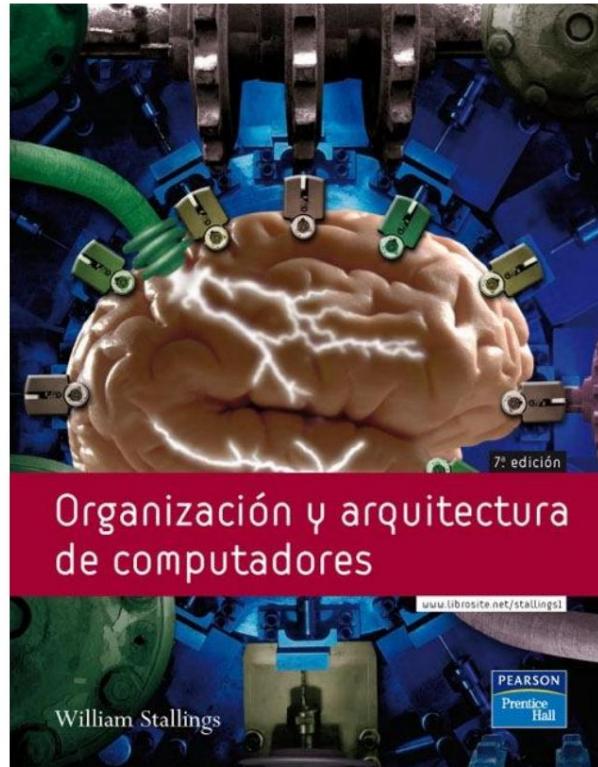
1. Sarrera
- 2 - Von Neumann arkitektura
 - 2.1 CPUak
 - 2.2 Memoria
 - 2.3 Sarrera / Irteera
- 3 - Datuen irudikapena
 - 3.1 Bit, Byte eta Word
 - 3.2 Karaktereak, osoak eta errealkak
- 4 - Aritmetika eta logika
 - 4.1 Eragiketa aritmetikoak eta logikoak zenbaki osoen gainean bitarrean
 - 4.2 Biribilketa 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 - Programazio lengoaien oinarrizko eraikuntzen mutua-lengoaien programatzea maila altaua

- 6.1 Esleipenen adierazpenak
- 6.2 Baldintzazko adierazpenak
- 6.3 Begiztak
- 6.4 Deiak eta funtzioaren edo azpierrutinaren itzulera
- 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 Memoria Zuzeneko Sarbide DMA
 - 8.5 Sarrera/irteera errutinen mutuaia-lengoaiaaren programazioa
- 9 - Memoriaren antolaketa
 - 9.1 Memoriaren hierarkia
 - 9.2 Latentzia eta banda-zabalera
 - 9.3 Cache memoria
 - 9.4 Memoria birtuala

1.6.1. Oinarritzko Bibliografia

- Teoria

William Stalling [<http://williamstallings.com/ComputerOrganization/>]



12. Irudia William Stalling Book

Ordenagailuen Antolaketa eta Arkitektura .William Stallings

7. edizioa, Pearson Prentice Hall berrargitalpena
ISBN 8489660824, 9788489660823. 2006

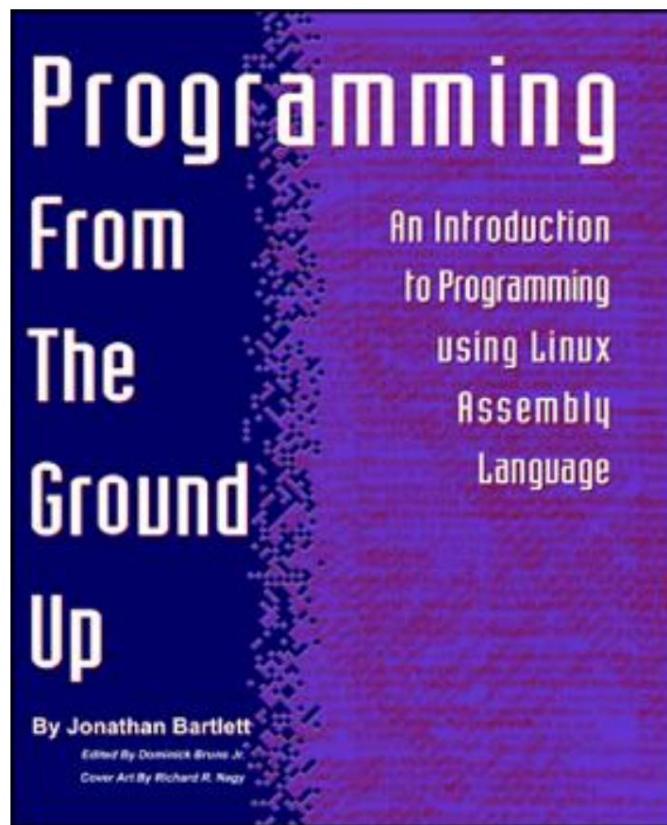
Ordenagailuen antolaketa eta arkitektura: errendimendurako diseinua.

William Stallings 9th

Ed Upper Saddle River (NJ): Prentice Hall, [2013]

ISBN 0-273-76919-7. 2012

- [7. COA](http://williamstallings.com/COA/COA7e.html) [<http://williamstallings.com/COA/COA7e.html>]
- [2019ko 11. edizioa](https://www.pearson.com/us/higher-education/program/Stallings-Pearson-e-Text-for-Computer-Antolakuntza-eta-Arkitektura-Sarbide-Kodea-Txartela-11. Edizioa/PGM2043621.html) [<https://www.pearson.com/us/higher-education/program/Stallings-Pearson-e-Text-for-Computer-Antolakuntza-eta-Arkitektura-Sarbide-Kodea-Txartela-11. Edizioa/PGM2043621.html>]
- Praktikak:



13. 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, Oklahomen

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

↳ [PGU liburua sarean](http://programminggroundup.blogspot.com.es/2007/01/programming-from-ground-gora.html) [<http://programminggroundup.blogspot.com.es/2007/01/programming-from-ground-gora.html>]

- ÿ PGU book home [<http://savannah.nongnu.org/projects/pgubook/>]
- ÿpdf _ [<http://download.savannah.gnu.org/releases/pgubook/>]
- Muntatzailea... baina oso erraza da [<https://upcommons.upc.edu/handle/2117/115067>]: 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 konpilatzailaren eskuliburu: **\$man gcc**



Eskuliburu hauek sartzea eta erabiltzea, lan saioetan azkar sartzeko erreferentzia-orriak izateaz gain, informatikako profesionalak bereganatu beharreko trebetasuna da.

1.6.2. Irakurketa gehiago

- Joan bibliografiako kapitulura.

1.7. Fakultatea

- Candido Aramburu Mayoz.
 - ÿ Telekomunikazio Ingeniaritzako Doktorea. Unibertsitateko katedraduna.
 - ÿ Los Tejos eraikina, 2. solairua. Office 2028.
 - ÿ Udazkeneko seihilekoko tutoretzak: ostiralean 8:00etatik 14:00etara.
 - ÿ barneko posta elektronikoa: Miaulario zerbitzariaren bidez.
 - ÿ PDI Irakasleak [<http://www.unavarra.es/ficha-docente?rangoLetras=az&uid=364>]
- Andres Garde Gurpegui
 - ÿ Jaurlaritzako Informatika eta Telekomunikazio Zuzendaritza Nagusiko goi-mailako teknikaria Nafarroakoa.
 - ÿ Irakasle elkartua (Laborategiko Praktikak)
 - ÿ Los Tejos eraikina, 2. solairua, INGENIARITZA saileko Elkarteen Aretoa ELEKTRIKOA, ELEKTRONIKOA ETA KOMUNIKAZIOA.
 - ÿ andres.garde@unavarra.es
 - ÿ PDI Irakasleak [<http://www.unavarra.es/pdi?uid=6578>]
- Carlos Juan de Dios Ursua
 - ÿ Ingeniaritza Elektrikoa eta Elektronikoa
 - ÿ Robotika eta Automatizazioan Masterra
 - ÿ Irakasle elkartua (Konputagailuen Egituraren Teoria)
 - ÿ Los Tejos eraikina, 2. solairua, INGENIARITZA ELEKTRIKO, ELEKTRONIKO ETA KOMUNIKAZIOKO INGENIARITZA Saileko Elkargokideen Aretoa.

ÿ carlos.juandediosATunavarra.es

1.8. Informatika lizentziatura

- [Informatika lizentziatura](https://www.unavarra.es/sites/grados/informatica-y-telecomunicacion/ingenieria_informatika/irakasgaiak-eta-irakaslea.html) [https://www.unavarra.es/sites/grados/informatica-y-telecomunicacion/ingenieria_informatika/irakasgaiak-eta-irakaslea.html]

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

ÿ 240 - Informatika Ingeniaritzan lizentziatura den Unibertsitate Publikoan
Nafarroa

1.9. egutegiak

- [Egutegi administratiboa](https://www.unavarra.es/sites/estudios/calendars.html) [https://www.unavarra.es/sites/estudios/calendars.html]

ÿ Teoria: Taldea 1 GG Osteguna 15:00 A318 ; 2 taldea GG Ostirala 17:00 A207 ; 91 taldea GG asteluhena 19:00 A135

- Praktikak

ÿ Informatika Ikasgelako egutegia 1. seihilekoa	[https://www2.unavarra.es/gesadi/zerbitzuaOrdenagailua/erabiltzailea/ikasgelak/Schedule1C.htm]
ÿ Informatika Ikasgelako egutegia 2. seihilekoa	[https://www2.unavarra.es/gesadi/zerbitzuaOrdenagailua/erabiltzailea/ikasgelak/Schedule2C.htm]

- Festak

Urriak 12 Asteazkena (Espainiako Jai Nazionala).
Azaroak 1 Asteartea (Santu Guztien Jaiak).
Abenduak 3 Asteartea (San Frantzisko Javier, Nafarroaren Eguna).
Abenduak 6 osteguna (Konstituzio eguna).

- [Ordutegiak eta Ikasgelak](https://www.unavarra.es/sites/grados/informatica-y-telecomunicacion/ingenieria_informatica/horarios-y-aulas.html#cCentralUPNA) [https://www.unavarra.es/sites/grados/informatica-y-telecomunicacion/ingenieria_informatica/horarios-y-aulas.html#cCentralUPNA]

ÿ Teoria

	asteluhena	asteartea	asteazkena osteguna	ostirala
15:00	T91-A135		PG1G2-A303 T1-A125	
erratsaldeko biostean	T91-A327		PG2-A306	T2-A234
19:00	PG1-A336			

ÿ Praktikak

A336 PG1 17:00 A303 PG1G2 15:00	
P327 P91 19:00 A306 PG2 17:00	
3/10	5/10
10/17	10/19

10/24	
	2/11
7/11	11/16
11/21	11/23
11/28	11/30

* P91: 3/10.17/10.24/10.7/11.21/11.28/11

*PG1: 3/10.17/10.24/10.7/11.21/11.28/11

*PG2: 5/10,19/10,2/11,16/11,23/11,30/11

* PG1G2: 5/10,19/10,2/11,16/11,23/11,30/11

* T1: A125 aretoa

* T2: A234 gela

* T91: A135 aretoa

* PG1G2: asteazkenean 15:00etan E-ISM/A303n (40 eserleku) -> Praktikak

G1 eta G2 talde teorikoak

* PG2: asteazkenean 17:00etan A306an (36 plaza) -> G2 Teoria Taldeko Praktikak

* PG1: astelehenean 19:00etan A336an (36 eserleku) -> Teoria Taldeko Praktikak

G1

* P91: astelehenean 17:00etan A327n (20 eserleku) -> Teoria Taldeko Praktikak
euskarra

* Andrés Garde irakaslea: 3 talde gaztelaniaz P1,P2,P3

* Carlos Juan de Dios irakaslea: Gaztelako taldea T2

* Cándido Aramburu irakaslea: T91-P91 euskara taldea eta T1 gaztelera taldea

* E-ISM Ikasgela: 34 eserleku

	A	B	C	D	E	F
1	2022 / 2023	LUNES	MARTES	MIÉRCOLES	JUEVES	VIERNES
2		29	30	31	1	2
3	SEPTIEMBRE	5	6	7	Sesión 1 (T1, T2*) G1 (A125) 15:00 G2 (A234) 17:00	8
4		12	13	14	Sesión 2 (T2) 15	16
5		19	20	19	Sesión 3 (T3*) G1 (? 15:00 G2 (? 17:00	22
6		26	27	26	Sesión 4 (T3/T4*) Sesión 5 (T5) G1 (? 15:00 G2 (? 17:00	29
7		Práctica 1 PG1 (A336) 19:00 3	4	Práctica 1 PG1G2 (A303) 15:00 PG2 (A306) 17:00 5	Sesión 7 (T6) 6	7
8		10	11	12	Sesión 8 (T6) 13	Examen parcial 1 (T1-T5) + Programación papel 14
9		Práctica 2 17	18	Práctica 2 19	Sesión 9 (T6) 20	21
10		Práctica 3 24	25	26	Sesión 10 (T6) 27	Examen de prácticas 1 (P1-P2) 28
11		31	1	Práctica 3 2	Sesión 11 (T6) 3	4
12		Práctica 4 7	8	9	Sesión 12 (T6) 10	11
13	NOVIEMBRE	14	15	Práctica 4 16	Sesión 13 (T6) 17	18
14		Práctica 5 21	22	Práctica 5 23	Sesión 14 (T6) 24	25

14. irudia. Laburpen-egutegia

- Ikastaro asteak:

ÿ 15 aste: ikastaroaren iraupena irailaren 2tik abenduaren 17ra

- azterketen ordutegia [<https://www.unavarra.es/sites/grados/informatica-y-telecomunicacion/ingenieria-computing/evaluation.html#cCentralUPNA>]

ÿ 2023ko udazkena [http://www2.unavarra.es/gesadj/CyD1/ETSIIT/Examenes/Grados/Otono/240_otono.pdf]

1.10. Problemak ebazteko ariketak

- Oinarrizko ariketak egitea apunteetan jasotako programako kapitulu bakoitzean.
- Azterketa moduko ariketak.

1.11.1. Oroitzapenak

- Bost gidoiak: ikasgela birtuala
- Banakako praktikak.
- Oroitzapenak :
 - ÿ Memoriaren entrega myclassroom zerbitzariaren bidez egingo da hurrengo praktika saioa irakasleak adierazi eta argitaratutako egunean.
 - ÿ Txostenetan PDF formatuan dokumentu bakarra da.
 - ÿ Fitxategiaren izenak abizena1_abizena2_tituluaguionpractica.pdf izan behar du
 - ÿ Memoria edukia.
 - ÿ Pseudokodean deskribatutako programa
 - ÿ Iturburu-kodea mutua-lengoaien behar bezala komentatua
programa pseudokodean.
 - ÿ Txostenaren estiloa ez da baloratzen, bere edukia baizik, interesgarria baita balio izatea.
ebaluazio-probetarako oharrak.
- ÿ
 - Ezinbestekoa da oharrak hartza laborategiaren barruan eta kanpoan
 - Gorde lan guztiak pendrive batean edo bidali postaz
- ÿ • Disko gogorreko edukia egunero automatikoki ezabatzea.

1.12.1. UPNA

informatika zerbitzua

- UPNA [https://www.unavarra.es/servicio-informatico/servicios/pdi?languageld=100000]
- ÿ [Erreserba Gelak](http://www2.unavarra.es/gesadj/servicioInformatico/usuario/aulas/GERES.pdf) [http://www2.unavarra.es/gesadj/servicioInformatico/usuario/aulas/GERES.pdf]
 - ÿ [VDI ikasgelak](https://www.unavarra.es/servicio-informatico/servicios/pdi?opcion=23) [https://www.unavarra.es/servicio-informatico/servicios/pdi?opcion=23]
 - ÿ [VDI Urruneko Sarbiderako Eskuliburua](https://www2.unavarra.es/gesadj/servicioInformatico/usuario/aulas/RemoteAccessVDI.pdf) [https://www2.unavarra.es/gesadj/servicioInformatico/usuario/aulas/RemoteAccessVDI.pdf]
 - ÿ [Mahai birtuala](https://vdibroker.unavarra.es) [https://vdibroker.unavarra.es]
 - Urruneko Informatika Laborategia
 - ÿ [sarbide orokorra](https://laboratoriosvirtuales.unavarra.es/) [https://laboratoriosvirtuales.unavarra.es/] ÿ ¿?

ARM/FPGA urruneko laborategia

- ARM/FPGA urruneko laborategiaren erabilera ez da ikastaroaren planean aurreikusi, baina erabilgarri dago erabili nahi duenarentzat, eta horretarako sarbide-kontu bat ireki behar da.
- [fpga sarbidea](https://laboratoriosvirtuales.unavarra.es/fpga/#/) [<https://laboratoriosvirtuales.unavarra.es/fpga/#/>] ÿ FPGA-Raspberry urrutiko laborategia
 - ÿ Guacamole saioa ÿ UPNA kredentzialak
 - ÿ Sartea DE1-SoC (ARM) txartela ÿ saioa hasi ÿ Linux shell kontsola
 - ÿ Sartu Raspberry (ARM) txartela ÿ saioa hasi ÿ Linux shell kontsola
 - ÿ Xilinx (ARM) txartelaren sarbidea ÿ saioa hasi ÿ linux shell kontsola
- [Enpresa kolaboratzailea Labsland](https://labsland.com/es) [<https://labsland.com/es>]: DE1-SoC instantziak. Web interfazea.
 - ÿ Zure sarbidea nire ikasgelan instalatuko da EDA botoia erabiliz.
 - ÿ Gailu digitalak diseinatzeko tresnak

1.12.2. Lanpostuak: 32 eta 64 bit

Arkitektura



Ikaslearen lan-estazio partikularra Ubuntu (64 bit) x86-64 prozesadore batean bada, ez dago arazorik x86-64 arkitektura duten programak exekutatzeko, aldiz, ikastaroan garatu beharreko programek 32 biteko arkitektura erabiltzen dute. programa hauek konpilatu eta exekutatzeko beharrezkoa dela

zer instalatu

- Beharrezko da arkitektura x86-64 edo amd64 duen CPU bat eta [Linux](https://linuxfoundation.org/) Sistema Eragilea duen PC bat edukitzea. [<https://linuxfoundation.org/>]/[[https://www.gnu.org/software/\[GNU\]](https://www.gnu.org/software/[GNU])] banaketa [Ubuntu](https://ubuntu.com/download) banaketa erabiliz [<https://ubuntu.com/download>]. 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, [VirtualBox](https://libvirt.org/) [<https://libvirt.org/>], etab).
- Praktikak egiteko, gutxieneko instalazioa beharrezko da tresnekin maila baxuko programazioa: **binutils**, **gcc** konpilatzailea , **gdb** **araztailea**, **vim** editorea
- Ubuntun:
 - ÿ instalatu paketeak `sudo apt-get install binutils` komandoa erabiliz
`gcc gcc-multilib gdb vim`
 - ÿ egiaztatu paketeen instalazioa `dpkg -l binutils gcc` komandoa erabiliz
`multilib gdb vim` programa bakoitzaren egoera *ii* dela
 - ÿ egiaztatu bertsioak: `gcc --version && as --version && ld --version && gdb --version && vim --bertsioa`

- C eta ASM mutua-lengoaietan programak editatzeko, erabil dezakezu
Vim editoreaz gain, hobetsitako editorea (nano, sublime, gedit, kate, emacs, etab..) eta/edo IDE
hobetsi (Visual Studio Code, Eclipse, etab..)

1.12.3. Check in

nire ikasgela

- [Myaulary](https://miaulario.unavarra.es/portal) [<https://miaulario.unavarra.es/portal>]

github

- Beharrezkoa da kontu bat irekita edukitzea [Github](https://github.com/) biltegiko plataforman [<https://github.com/>]

Google

- [Google kontuak](https://support.google.com/accounts/answer/27441) [<https://support.google.com/accounts/answer/27441>]
- [Google Chrome arakatzailea](https://www.google.com/chrome/) [<https://www.google.com/chrome/>]: hizkuntza hautatzeko aukera ematen du

1.13. Praktika Taldeak

- 3 praktika talde gaztelaniaz: PG1, PG2, PG1G2

PG1 PG1G2	PG2	
Alen Urra, Saul Goikoetxea Macua, Jon Iñaki Beperet, Iker		
Alonso Gomez, Ana Goñi Lara, Iker Fernandez Illera, Iker		
Altamirano Trujillo, Ruth N. Fernández Picorelli, Marina Ithurritz Sesma, Eneko		
Alvarez Alonso, Markel	Flamarique Arellano, Aritz Orradre Berdusán,	
Joaquin		
Andreu Mangado, Alvaro Fortun Iñurrieta, Lucas N. Pascal Alegria, Xabier		
Apesteguia Vazquez, Alejandro Gallo Ansa, Borja Portuondo Varona, Helen		
Arriazu Muñoz, Jon Garatea Larrayoz, Iñaki Ripoll Baños, Izar		
Ayechu Garriz, Santiago Garcia Vilas, Jorge Daniel Romero Sadaba, Irene		
Azcona Furtado, Lucia Gil Gil, Santiago Paula	Ruiz de Gopegui Rubio,	
Aznarez Gil, Iñigo	Gomez Ciganda, Iker Javier Sola Biezobas, Fermin	
Baigorrotegui Gil, Oier Beñat	Granda Saritama, Anthony D. Solaegui Garralda,	
Bayona Restrepo, Karol Julian Hualde Romero, Israel Bellera		Taberna Maceira, Alain
Alsina, Alberto Iribarren Ruíz, Beñat Andrea		Zamborán Maldonado,
Pascual kalea, Vidal	Juanotena Ezkurra, Joseba	Martinez Sesma, Alvaro
Cascan Ustarroz, Eduardo	Juarez Jimenez, Adrian	Melendez Uriz,
Alexandro		
Cerezo Uriz, Inaki	Labat Garcia, Pablo	Mellado Ilundain, Inaki
Chacon Flores, Malena Paola	Labiano Garcia, Martin J.	Molina Puyuelo,
Alexandro		
Cordido Perez, Elena	Larrayoz Diaz, Asier	Monreal Ayanz, Ander
Couceiro Eizaguirre, Javier	Larrayoz Urra, Carlota	Moral Garcia, Haizea

Neck Tejero, Jorge de la Ossa Goñi, Miguel Diaz Ochoa, Alex Misael	Latasa Sancha, Ivan Liébana Revuelta, Diego Longás Saragüeta, Endika J. Redrejo Fernández,	Oroz Azcarate, Angel Perez Alvarez, Angel
Dobromirova Karailieva, Z. Ezponda Igea, Eduardo Felipe Goicoechea Elio, Maria	Lumbreras Korridorea, Imanol Saiz Larraz, Eder Martinez Arpon, Alain	Santos Garzón, Jaider
Sergio	Martinez de Goñi, Unai	Sanz Sanz, Ivan Solo Alba, Alexander Tellechea Zamanillo,
Kaleb		Turrillas Remiro, Ethan Urroz Velasco, Ibai Vadillo Navarro, Asier Yarhui Sarate, Yerson
		zheng, Yushan

1.14. Metodologia

- Teoria, Ariketak, Praktika eta Azterketak.

1. taula. metodologia

Metodologia - Jarduera	Harremanetarako Ordutegia	ordu ez presentzialak
A-1 Master Class	24	
A-2 Ikasketa autonomoa		30
A-3 Saio praktikoak	16	
A-4 Programazioa / esperimentazioa edo beste ordenagailu lana / laborategia		hogei
A-5 Arazoak konpontzea, ariketak eta bestelako jarduerak aplikazioa		12
A-6 Oinarritutako ikaskuntza arazoak eta/edo kasuak	14	
A-7 Obrak prestatzea eta/edo proiektuak eta idazketa oroitzapenak		hamarka
A-8 gidoiaren irakurketa, aurkezpenak prestatzea lanak, proiektuak, etab...		hamabost
A-9 Ebaluazio-jarduerak		6

Metodologia - Jarduera	Harremanetarako Ordutegia	ordu ez presentzialak
A-10 Tutoretzak	bi	
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)

1.14.1. Kreditu banaketa

- Kredituen banaketa:

ÿ kreditu guztira: 6 ECTS
 ÿ ordu guztira: 6x25 : 150 ordu
 ÿ ordu presentzialak: 62 ordu
 ÿ eskolak: 24 ordu. Ikastaroaren guztira 15 asteetako 12 astetan (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 asteetako 3 astez (2 ordu/astean).
 ÿ tutoretzak: 2 ordu
 ÿ Harremanik gabeko orduak: 88 ordu

1.14.2. Praktiken kredituen banaketa

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

1.15. Ebaluazioa

- [Ordenagailuen Web Egitura](http://www.unavarra.es/ficha-asignaturaDOA?languageld=100000&codPlan=240&codAsig=240306) [http://www.unavarra.es/ficha-asignaturaDOA?languageld=100000&codPlan=240&codAsig=240306]

ÿ praktikak: 2 proba partzial. Lehen proba partziala urriaren 28an (praktika 1. eta 2.).
 Bigarren proba partziala 2023/12/01ean (3., 4. eta 5. praktikak). Gutxieneko nota bakoitzean 4 bi proba partzialen artean.
 ÿ teoria: 2 proba partzial. Lehen proba partziala urriaren 14an. Bigarren proba 2023/12/01eko azken deialdi arruntean parte. Gutxieneko nota 4 puntu bakoitzean bi proba partzial.
 ÿ Ebaluazio-proba bakoitzaren balioaren banaketa: % 15 (asistentzia eta jarrera klasea Y laborategia)+%35(ariketa praktikoak)+15%(programazioa paperean) Y lanpostuak)+%35(kontzeptuak Y

ÿ Derrigorrezko proba praktiko eta teorikoak:

- Laborategiko praktiketara joatea: **derrigorrezkoa** da orduen %87,5era joatea praktikak laborategian.
- Praktiken txostenak aurkeztea: **derrigorrezkoa** da txostenen % 100 aurkeztea, ezarritako epean ikasgelako zerbitzarian gaituta dagoen egunean. Praktiken edo zereginen txostenak ez dira my classroom zerbitzariak ezarritako epea igaro ondoren jasotzen.
- Praktiken orduen % 87,5ra joatea derrigorrezkoia izatea eta ezarritako epean eta erdian txostenak ematea ezinbesteko baldintza da irakasgaia gainditu ahal izateko.

1.16. azterketak

- 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 zatirako: %60 kontzeptu teorikoei buruzko galderak eta %40 ariketak.
- Paperean oinarritutako programazio-azterketa ordenagailurik gabe egingo da eta muntaia-lengoaiaren jarraibideen, GDB araztailearen eta C lengoaiaren adierazpenen erreferentzia-orriekin.
- Laborategiko azterketa praktikoa txosten praktikoekin eta erreferentzia-fitxekin egingo da eta informazio elektronikorik erabili gabe ez urrutitik (Interneterako sarbidea) ez lokalean (USB pendrivea, etab).
- Egutegia:

1. teoria partziala: 1,2,3,4,5 eta 6 gaiak: 2022/10/14
praktiko partziala: 1 eta 2 gidoiak: 2022/10/28 deialdi arrunta: 2. partziala (7,8 eta 9 gaiak): 2023/01/12 08:00 berreskuratzeko
deialdia: (1,2,3,4,5,6,7,8 eta 9. gaiak eta praktikak): 2023/01/28 08:00

2. kapitulua. 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:

- CPUak
- Memoria
- Sarrera irteera

2.1.2. Testuinguru historikoa

Aurrekariak

- 1833: Charles Babbage ѕ 1. ordenagailu mekanikoa diseinatu zuen
- 1890: Herman Hollerith-en **tabulazio-makina**. AEBetako errolda. IBM (1925)
- 1936: Alan Turing ѕ Algoritmoa 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: Pennsylvaniako Unibertsitatean (balistikako ikerketa laborategian artilleria) **ENIAC** (Electronic Numerical Integrator And Calculator) eraikitzen da
 - ї Balistikari buruzko ekuazio diferentzialak ($angelua = f$ (kokapena, buztaneko haizea, haize gurutzatua, airearen dentsitatea, tenperatura, oskolaren pisua, karga propultsatzalea,...))
 - ї Erabilera orokorreko ordenagailu elektronikoa (ez mekanikoa).
 - ї Memoria: 20 metagailu bakarrik ѕ triodoekin egindako txankletak
 - ї 18.000 tutu elektroniko edo huts-balbula
 - ї Etengailuen eskuzko programazioa
 - ї 100.000 jarraibide segundoko
 - ї 300 biderketa segundoko
 - ї 200kW
 - ї 13 tona eta 180 m2

EDVAC

- 1951: Pennsviriaiko Unibertsitatean (J. Presper Eckert eta John William Mauchly) **EDVAC** (Electronic Discrete Variable Automatic Computer) funtzionatzen hasi zen, **John von Neumannek** asmatua, zeina, ENIAC ez bezala, ez zen hamartarra, bitarra baizik, eta lehen **programa** (ez soilik datuak) gordetzeko diseinatuta zeukan : BILTEGIA PROGRAMA ORDENAGAILUA y programa datu gisa manipula daiteke.

y 500.000 \$

y EDVACek fisikoki ia 6.000 balbula termioniko eta 12.000 kristal-diodo zituen.

56 kilowatt-ko potentzia kontsumitzen zuen. 45,5 m²-ko azalera hartzen zuen eta 7850 kg-ko pisua zuen.

y Arkitektura:

y Zinta magnetikoko irakurgailu-grabagailu bat

y osziloskopioa duen kontrol-unitatea, argibideak jasotzeko unitatea
kontrola

y memoria: 2000 hitz biltegiratza "merkurioko atzerapen-erroak" y fidagarritasun eskasa

y koma mugikorreko unitate aritmetikoa 1958an.

IAS

- 1946-1952 : **IAS** (Ikasketa Aurreratuen Institutua) mainframe :

y EDVAC bilakaera: danbor magnetikoko memoria-unitate nagusia eta bigarren mailakoa.

y Selectron memoria: biltegiratze kapazitiboa y karga elektrostatikoa

Geroago

- 1952: **UNIVAC I** (UNIVersal Automatic Computer I) lehen ordenagailu nagusi komertziala izan zen. Hollerith tabulatzeko makinaren bilakaera errolda prozesatzeko aplikatutakoa
ERABILERAK.
- 1952: IBM 701, garatu zen bitartean "Defentsa kalkulagailua" bezala ezagutzen dena, IBMren lehen ordenagailu zientifiko komertziala izan zen y lehen **MUNTATZAILE lengoiaia**.
- 1964: mainframe (ordenagailu zentrala) **IBM 360** y ISA duen lehen ordenagailua (firmware) y bateragarritasuna

y siliziozko osagai diskretuen eta beste osagai batzuen arteko teknologia hibridoa y
ez "zirkuitu" integratuak.

y 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 integratua planoa

- 1968: Robert Norton Noyce eta Gordon Moore Intel aurkitu zuten.
- 1971: Intel 4004 8 bit CPU silizioan integratua 8 bit

2.2. Institute Advanced Machine (IAS): Arkitektura

2.2.1. Erreferentzia

- [Von Neumann Makina](https://es.wikipedia.org/wiki/M%C3%A1quina_de_von_Neumann) [https://es.wikipedia.org/wiki/M%C3%A1quina_de_von_Neumann]

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: makinak exekutatu behar dituen ORDENAK dituen programa bat garatuz implementatzen da algoritmoa.

ŷ Programazio deklaratiboa ez bezala: algoritmoak ZER implementatzen du nahi dugun ordenagailuak, ez NOLA, ez zuzenean exekutatu behar dituen 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 hizkuntzarik mihiatzadura bezalako sinbolikoa.
- Makina Lengoia: Kode Bitarra
- Kode bitarraren edizioa txartel zulatuen edo zinta magnetikoen bidez a kontsola.
- Memorian jasotako informazio mota: DATUAK eta JARRAIBIDEAK
 - ÿ Datuen adibidea: zenbaki osoak +3278,+5,-1,-6592,...
 - ÿ Argibide adibideak:
 - ÿ LOAD M(8) : metagailuaren erregistroan kargatu 8. memoriaren posizioaren edukia
 - ÿ GEHITU M(3) : 3. memoria-posizioaren edukia gehitzeko metagailu-erregistroan
 - ÿ JMP M(100): 100. memoria-posiziora salto egin
 - ÿ etab.
- **Biltegiratutako** programaren kontzeptua : Argibide bitarrak eta datu bitarrak
Memoria Unitatea
 - ÿ Von Neumanns arkitekturaren nobedade handia izan zen
 - ÿ Beharrezko da modulu bitarra ordenagailuaren MEMORIAN KARGATU behar da horrela gordeta.
- Programazio sekuentziala: Instrukzioak sekuentzialki exekutatzen dira memorian gordetzen diren heinean... betiere sekuentzia hausten duen jauzi-instrukzio esplizitua exekutatzen ez den bitartean.

Arkitektura: Instruction Set Architecture (ISA)

- Programa aztertu ahal izateko beharrezko da makinaren hizkuntza bitarra ezagutzeaz gain bere ARKITEKTURA ezagutzea ere. Ordenagailu baten arkitektura makinaren ZER da, hau da, makina ZEIN instrukzio exekutatzeko gai den, horretarako INSTRUCCION MULTO ARKITEKTURAREN ARKITEKTURA (Instruction Set Architecture **ISA**) ezagutzea beharrezkoa da:
 - ÿ argibide-multzoa: eragiketak eta datuak 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

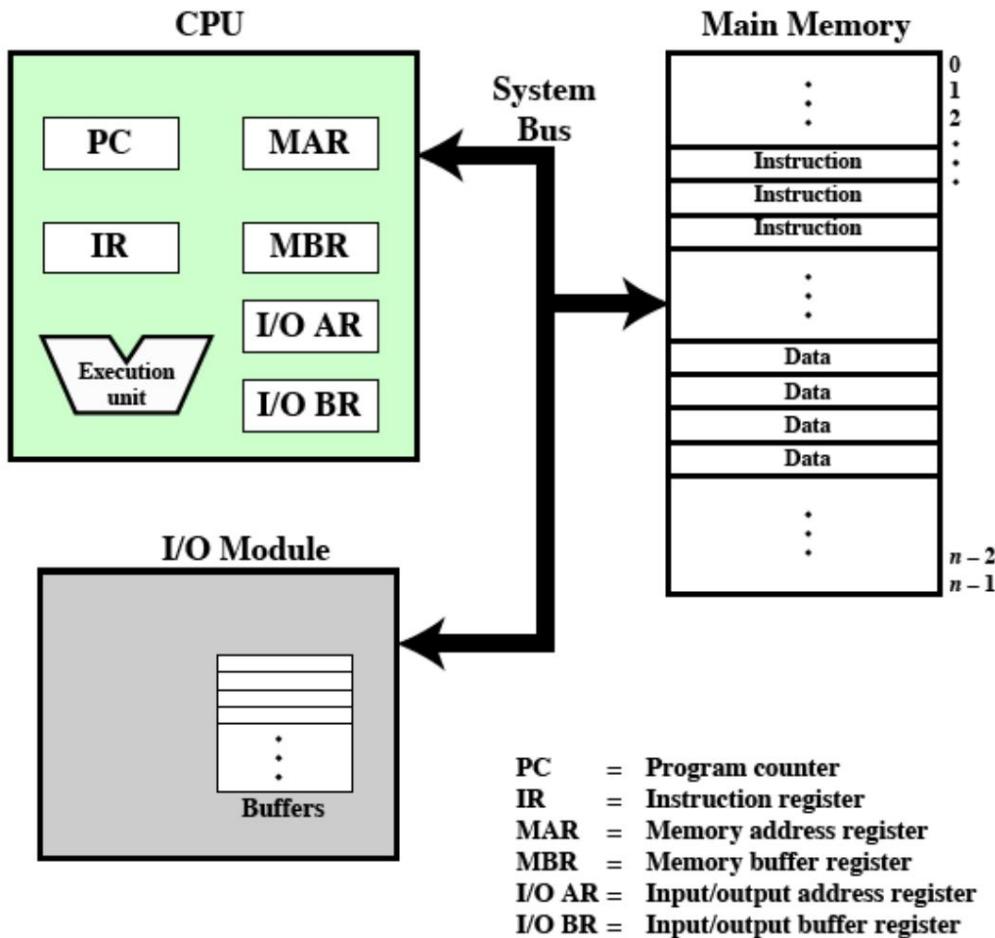
ÿ Egitura makinaren NOLA da. Zer hardware egin behar dugu

arkitekturak definitutako makinaren instrukzioak exekutatu.

- **Egitura modularra** duen hardwarea :

↳ CPU-Memoria-I/O-Bus

↳ Memoriaren hierarkia: 2 maila: Memoria Nagusia (PUZaren kanpoko) eta Erregistroak (PUZaren barne)



16. Irudia IAS makinen arkitektura

- CPU Barne Arkitektura: Mikroarkitektura

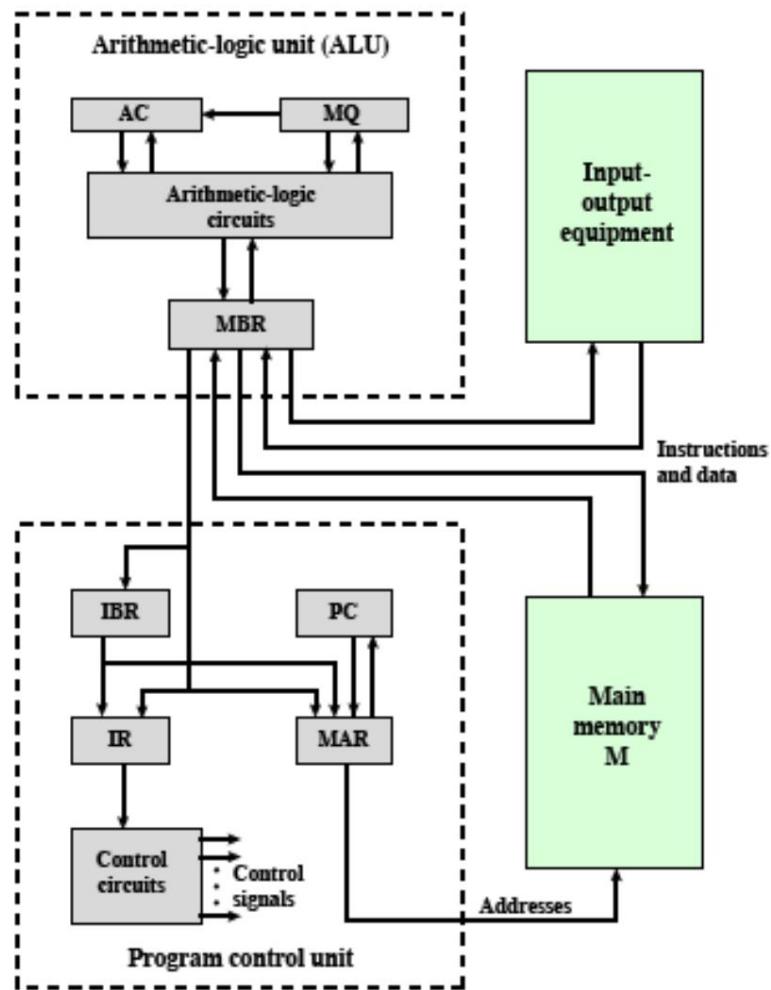


Figure 2.3 Expanded Structure of IAS Computer

17. Irudia IAS makinaren egitura

2.3.2. Prozesatzeko Unitate Zentrala (CPU)

- CPUak:

ÿ PUZaren funtzionamendua 3 FASEtan banatzen da: Harrapatu, Interpretatu eta Exekutatu jarraibideak sekuentzialki. 3 faseen sekuentzia izenarekin ezagutzen da

Irakaskuntza Zikloa.



18. Irudia. Irakaskuntza Zikloa

- Programa baten makina-instrukzio bakoitza CPUak harrapatzen, interpretatzen eta exekutatzen du eta ordena horretan. Hori kontrolatzeaz arduratzen den zirkuitu elektroniko digitala

sekuentzia CPUan integratutako **Kontrol Unitatea** da. Kontrol Unitateak seinale elektronikoen bidez mikroaginduak ematen dizkie kaptura-azpizirkuituari, interprete-azpizirkuituari eta exekutatzailera-azpizirkuituari, memoria nagusian gordetako programaren instrukzio bakoitzaren instrukzio-zikloaren fase guztiak burutu daitezen.

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

2.3.3. Oroitzapenak

Oroimen nagusia

ren espazioa
HELBIDEAK

EDUKIAK

0x00000000	0101010101010101010
0x00000001	0101010101010101010
0x00000002	0101010101010101010
0x00000009	
0x0000000a	
0x0000000f	

19. Irudia Memoria Nagusiaren Edukiaren Helbideratzea

- 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
- Ausazko Irisgarrizko Hitzetan antolatuta. Ausazko Sarbide Memoria.
- Dinamismoa: datuak eta argibideak irakurri/idatzi
- IAS makinan, memoria-helbideek izan daitezkeen 40 biteko hitzetara seinalatzen dute gorde 40 biteko datu bat edo 20 biteko bi instrukzio.
- Random Access Memory (RAM): memoria-kokapen bakoitza zuzendu daiteke.
- Memoria partekatua: datuen eta argibideen arteko memoria partekatua. Autobusa ere partekatzen dute memoriarako sarbidea.

- 212=4K hitzetarako gaitasuna hitz bakoitzeko 40 bitrekin.

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

ŷ Horren ordez, garai hartan zegoen memoria fisikoa hauxe zen: 40 biteko 1024 hitz = 5 KByte («The Computer from Pascal to 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 programatzailailek.
- AC eta AR/MQ: ALUren metagailuak. Biderkatzaila/Kozientea .Erregistro eskuragarri bakarrak dira programatzailailek.
- Programatzailailek eskura ditzakeen erregistroak: Kontrol Unitateko erregistro guztiak: MBR,PC,IR,IBR,SEA

ŷ MBR: Selectron Register edo Memory Buffer Register *MBR* edo Data Buffer Register *DBR*. 40 biteko tamaina. Instrukzio-zikloaren bilatze-fasearen ondorioz memoriatik irakurritako datuak edo instrukzio-pareak gordetzen ditu edo instrukzio-zikloaren azken fasearen ondorioz idatzi beharreko datuak memorian gordetzen ditu.

ŷ PC: Kontrol-kontagailua: Programa-kontagailua (PC) edo Instrukzio-erakuslea (IP). 12 biteko tamaina. Eskuratutako instrukzioa seinalatzen du

ŷ IR: Kontrol-erregistroa: Instrukzio-erregistroa *IR* ere *deitzen zaio*. 20 biteko tamaina. Instrukzio-zikloan eskuratutako instrukzioa gordetzen du

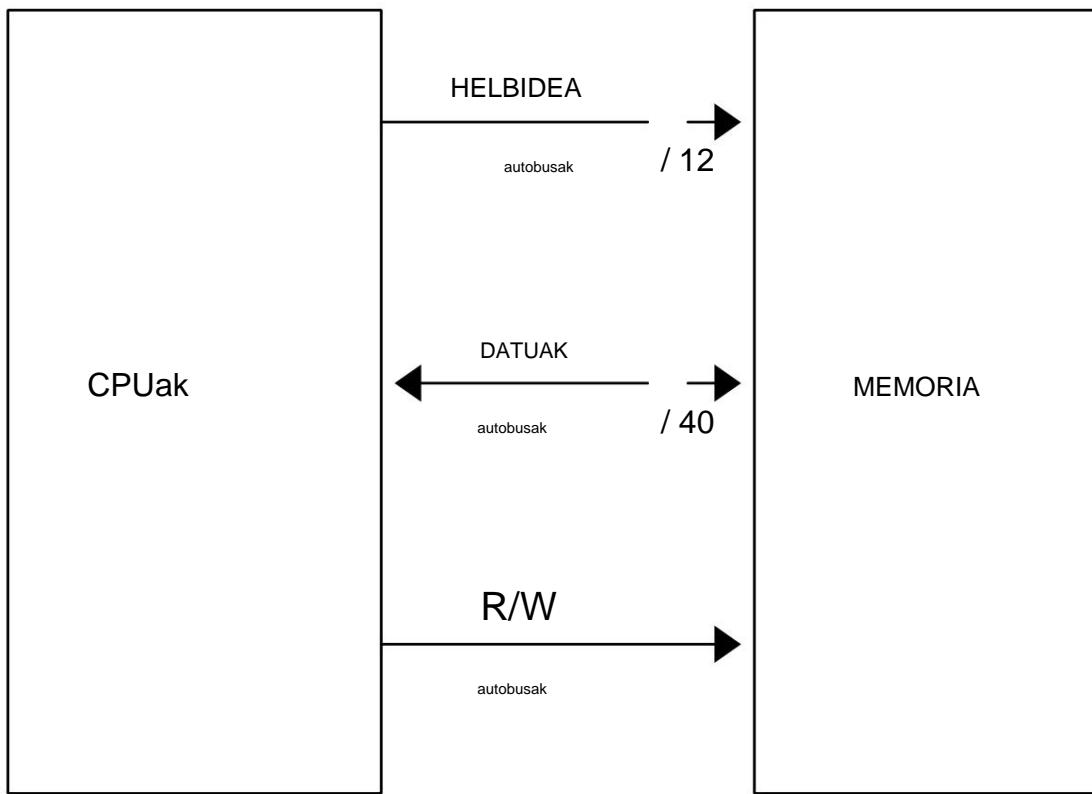
ŷ IBR: Instruction Buffer Register: Instrukzio-zikloan lortutako bigarren instrukzioa gordetzen du. 20 biteko tamaina. Kontuan izan horrek esan nahi duela eskuratzeari bi instrukzio aldi berean lortzen direla.

ŷ MAR: Memory Address Register: uneko Memoria Helbidea. 12 biteko tamaina. Instrukzio-zikloaren lehen fasean eskuratu beharreko eragiketa edo instrukzioa adierazten du.

- Bere tamainak arkitekturaren "hitzen tamaina" deritzona definitzen du. IAS makinak 40 biteko arkitektura edo 40 biteko Word bat du

2.3.4. autobusak

- Bi gailu elektroniko konektatzeko hari edo bide paralelo metalikoen multzoa. Denek izan dute esku artean USB bus bat (Universal Serial Bus) duen USB kable bat.
- Sistema Busa:
 - ŷ CPU-Memoria Nagusiaren interkonexioa: datuen eta argibideen transferentzia.
 - ŷ Datu-Busa (40 hari), Helbide-Busa (12 hari) eta Kontrol-Busa (Irakurri/Idaztea) (hari 1). 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 programatzairearekin, bai beste makinekin. Ordenagailura kanpotik sartzeko, teklatuak, pantailak eta abar bezalako periferikoak beharrezkoak dira.
 - ÿ IAS makinan programa txartel perforatuetan idazten da. Datuetarako txartelak eta argibideetarako txartelak. Beharrezko da datuak memorian kargatzea programa exekutatu aurretik.
 - ÿ txartel zulatuak, kontsola, danbor magnetikoak, zinta magnetikoak, txartel-irakurgailu baten bidezko memoria-kargagailua, huts-hodien bidez bistaratzea, etab. ÿ teknologia zaharkitua.
 - ÿ Ez dugu I/O modulua kontuan hartuko eta CPU-Memoria moduluetan zentratuko gara Nagusia.

2.3.6. Irakaskuntza Zikloaren Animazioa

- [Irakaskuntza-zikloaren animazioa](https://www.youtube.com/watch?v=04UGopESS6A) [https://www.youtube.com/watch?v=04UGopESS6A]

2.4. ISA: Instruction Set Architecture IAS makinarena

2.4.1. Datuen formatua eta IAS Informatikako Argibideak

- Memoriaren Arkitektura

ÿ Hitzak

ÿ 40 bit: 1 datu edo 2 instrukzio

ÿ Datuak

ÿ Zenbaki osoak 2ren osagarri formatuan.

ÿ *Datuen formatua*



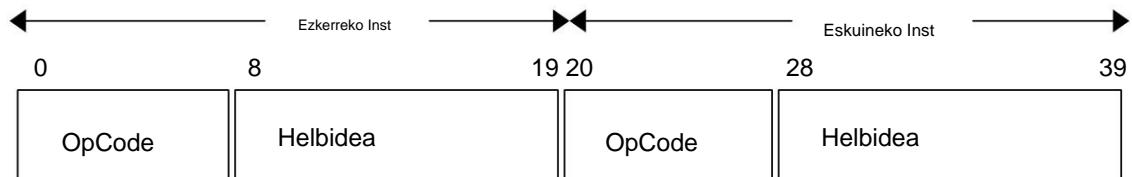
21. Irudia Datuen formatua

ÿ Behatu 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 Argibideen formatua

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

ÿ *Metagailuetan Oinarritutako Arkitektura*

ÿ Bi eragiketa behar dituen eragiketa batek inplizituki aipatzen du

metagailuan gordetako eragiketa

ÿ Behatu zero zenbakidun bit-a ezkerrekoa dela.

ÿ Ezkerreko instrukzioa (0-19) erregistroetan kargatzen da, CPUaren barnean, kodean.

IR funtzionamendua eta SEAko operazio eremua

ÿ Eskuineko instrukzioa (20-39) erregistroan kargatzen da, CPUREN barnean, IBR

ŷ Eragilearen helbideratze modua: Operandoaren erreferentzia. Arkitektura hau "Direct Addressing" izeneko helbideratze modu bakarrarekin diseinatu zen non instrukzioaren eragigaien eremuan eragigaiaren **memoria helbidea** zehazten den

- Memoria Edukia

- ŷ Memoria-helbideak bikoitz gisa bistaratzen dira, lehenengoari 20 biteko LSBri eta bigarrena 40 biteko memoria-hitzaren 20 biteko MSBri egiten diotelako erreferentzia.
- ŷ Kontuan izan datuen zutabeen bi atal daudela: argibideen atala eta atala datuen
- ŷ Von Neumann arkitekturan, datuek eta instrukzioak memoria-espazio bera partekatzen dute. memoria helbideak.

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

- Erregistro-transferentzia hizkuntzari buruzko informazioa (RTL) [Eranskinian](#)

IAS makinaren errepetorioa

- Instruction Set Architecture (ISA): instrukzio multzoaren definizioa eta ezaugarriak. Argibide multzoen arkitektura.
- Jatorrizko bertsioan ez zegoen mutaria-koderik, zuzenean hizkuntzan programatu zen makina.
- ŷ Erantsitako taulan, bigarren zutabeen, jarraibideen eragiketen **MNEMONIKA** (KARGA, GEHITU, SUB, etab.) William Stalling-en testuliburuak diseinatutakoekin bat datoz. 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.

Taula 2.Instrukzio multzoa

Instrukzioa ekintza yam	argibideak	Op Deskribapena	RTL
S(x)ÿ Ac+	LOADM(X)	1 kopiatu zenbakia Selectron kokapenean x AC AC ÿ M[x]	
S(x)ÿ ak	KARGA -M(X)	2 #1 bezala baina kopiatu AC ÿ ~M[x]+1 zenbakiaren negatiboa	
S(x)ÿ akm	ZARGA M(X)	3 #1 berdina baina kopiatu balio absolutua	AC ÿ M[x]
S(x)ÿ Ac-M	ZARGA - M(X)	4 #1 berdina baina balio absolutua kendu	AC ÿ AC- M[x]
S(x)ÿ oh+	GEHITU M(X)	5 Gehitu Selectron kokapenean x zenbakia AC-n	
S(x)ÿ oi	AZPI M (X)	6 kendu Selectron kokapenean x zenbakia AC-tik	
S(X)ÿ hmm	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)ÿR ZARGA 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 in-ko zenbakiaz Aukeratu kokapena x. Jarri zatidura AR-n eta gainerakoa AC-n.	
CuÿS(x)	JUMP M(X,0:19)	D Jarraitu ezkerreko instrukzioan exekutatzen bikotearen Selectron kokapenean x	
Cu`ÿS (x)	SALTO EGIN M(X,20:39)	E Jarraitu eskuineko instrukzioan exekutatzen bikotearen Selectron kokapenean x	
CcÿS(x)	JAUZTU+ M(X,0:19)	F AC-ko zenbakia >= 0 bada, jarraitu #D-n bezala. Bestela, jarraitu normal.	
Cc`ÿS (x)	JAUZTU+ M(X,20:39)	10 AC-ko zenbakia >= 0 bada, jarraitu #E-n bezala. Bestela, jarraitu normal.	

Instrukzioa	argibideak	Op Deskribapena	RTL
ekintza yam	yam		
AtÿS(x)	STOR M(X)	11 Kopiatu AC zenbakia Selectron x kokapenean	
ApÿS(x)		12 Ordezkatu ezkerreko eskuineko 12 bitak Selectron kokapenean x instrukzioa AC-aren eskuineko 12 bitetan	
Ap`ÿS (x)		13 #12ren berdina baina eskuineko eskua aldatzen du argibideak	
L	slh	14 Mugitu AC zenbakia ezkerrera bit 1 (bit berria eskuinean 0 da)	
R	HSR	15 Mugitu zenbakia AC-n eskuinera 1 bit (ezkerrekoena bit kopiatzen da)	
gelditu		0 Programa geldiarazi (ikus IASren 6.8.5 paragrafoa txostena)	

- Argibide multzoa (William Stallings)

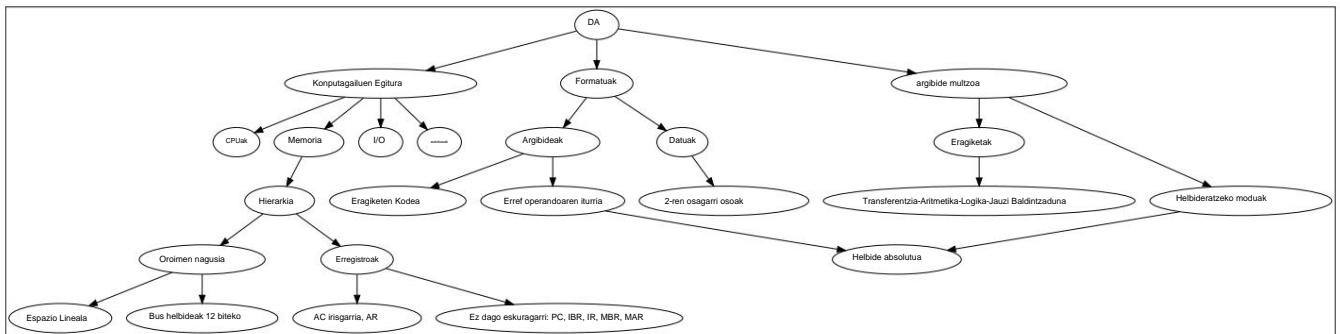
Table 2.1 The IAS Instruction Set

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 **Interfazea** definitzen du makinen softwarea
 - ŷ Bi CPU guztiz desberdinak izan ditzakegu, adibidez AMD eta Intel, baina ISA bera badute makina bateragariak izango dira sistema eragilearen ikuspuntutik.
 - ŷ Familia kontzeptua: argibide-multzo bera desberdinek exekutatu dezakete ordenagailuak
- Makina ezberdinen ISA [http://en.wikipedia.org/wiki/List_of_instruction_sets]

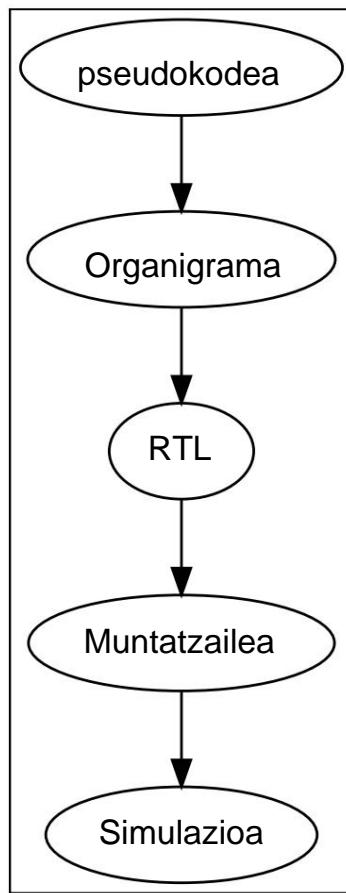


25. Irudia Instruction Set Architecture (ISA)

2.5. Programazioa IAS Asanblea Lengoain

2.5.1. Batzar Hizkuntzako Programa baten Garapen Estrategia

- Programazioaren bidez konpondu beharreko arazoa ulertu ondoren, ez da horrela zuzenean programatu arazoaren iturburu-moduluaren soluzioa baina badoa arazoa eta konponbide-algoritmoa hainbat hizkuntzatan eta hizkuntzatan deskribatuz ebatzea fase hauek:
 - ÿ Algoritmoaren deskribapena "pseudokode" hizkuntzan.
 - ÿ Algoritmoaren deskribapena organigrama edo fluxu-diagrama erabiliz.
 - ÿ RTL erregistroen arteko transferentzia hizkuntzan algoritmoaren deskribapena.
 - ÿ Algoritmoaren deskribapena ordenagailuaren berezko muntaia-lengoaiaren



26. Irudia Programazio faseak

y

Maila handiko lengoia bateko deskribapenetik maila baxurako igarotzea RTL lengoian egiten da makina lengoia exekutatuko den ordenagailuaren arkitektura kontuan hartuta. Goi-mailako instrukzio bakoitza behemailako instrukzioen bloke batean itzuli behar da.

2.5.2. 1. adibidea: sum1toN.ias

adierazpena

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

pseudokodea

- Algoritmoaren deskribapena NATURAL hizkuntzan testu-moduko esamoldeak erabiliz.
- ALDAGAIAK:

ÿ batura aldakorra : emaitza partzialak eta amaierakoak gordetzen ditu

ÿ N aldagaia : sarrerako datuak gordetzen ditu

ÿ i aldagaia : iterazio bakoitzean aldatzen den gehigarria gordetzen du

- Kode inperatiboaren egitura:

ÿ Oinarrizko adierazpenen eraikuntza begizta bat da

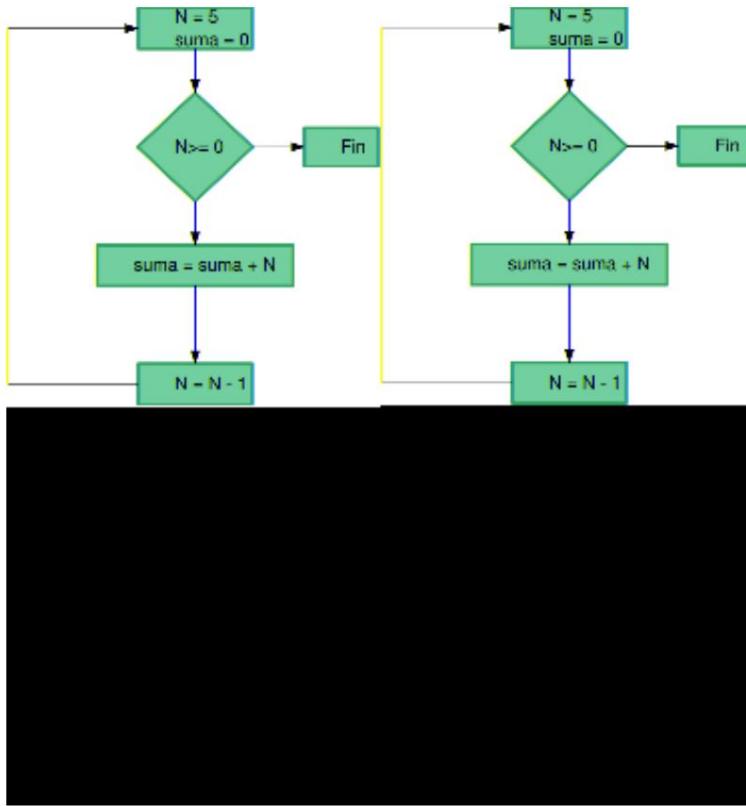
ÿ Begiztak atzerako iterazioak zenbatzen ditu

ŷ Iterazio bakoitzean "i" batuketa bat sortzen da eta sum=sum+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
;RTL sum1toN algoritmoaren deskribapena

;DATUEN ATALA:
; Etiketa-deklarazioa, kanpoko memoria esleitzea, hasieratzea
; Aldagai arruntak n: M[n]
<- 5 ; aldagaieetako batuketa: M[batura] <- 0 ; batura
partziala eta amaierako aldagaia

;ATALAREN JARRAIBIDEAK
;Metagailuei zuzendutako arkitektura (AC)
;Erregistro eskuragarriak :
AC ; start loop : gehigarrien batuketa eta sorrera begizta:
AC <- M[n] ; karga gehitzea
AC>=0 : PC <- gehitu ; sumand < 0 begizta amaieran ; geldialdi-
begizta amaiera
  
```

```
; egin batura
gehitu: AC <- AC + M[batura]
M[batura] <- AC
; eguneratzea gehituz
AC <- M[n]
CA <- CA - 1
M[n]           <- AC
; hurrengo iterazioa
PC <- begizta
```

Mihiztadura-lengoia IAS makinaren Wstallazioa

- iturburu-modulua sum1toN.ws

```
; OHARRA
; 1. bertsioa: sum1toN_v1.ias
; Kalkulatu zenbaki osoen segida baten batura: batura = 1+2+..+n
; sarrerako datuak: n
; irteerako datuak: batura
; Algoritmoa: n iterazioko begizta
;               Gehigarriak na 0tik beheranzko ordenan sortzen dira
;               Begizta irtengo da sumandoa negatiboa bada -> -1
; Datuen egiturak: n eta batura aldagaiak. Konstante bat.
; Mihiztadura hizkuntza: William Stalling
; Von Neumann IAS makinaren arkitektura
```

```
;;;;;; JARRAIBIDEEN ATALA
;Metagailuei zuzendutako arkitektura (AC)
;Erregistro eskuragarriak : AC
; algoritmoa: n, n-1, n-2,...0,-1 segida sortzen duen begizta n>=0 bada
begizta: LOAD M(n) ; AC <- M[n]
SUB M (bat) ; AC <- AC-M[bat]
DENDA      M (batuketa) ; M[batura] <- AC
JMP+ begizta; begiztaren ; AC >= 0 bada, begizta jauzia
amaiera
GELDITU ; gelditu
```

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

- ÿ Memoria erreserbatzeko datuen atala garatu da.
- ÿ BEZKIZ SIMPLE BAT egin da begizta beharrezkoa den eraikuntza delako azken algoritmoa.

ÿ SUB eragiketa egin da, azken algoritmoan beharrezkoan den eragiketa baita.

ÿ Kodea IRUZKINATU da

iassim mutuaia-lengoaia

Iturburu-moduluaren garapena mutuaia-lengoaian EZ da hasieratik amaierara arte egiten,
URRATSUTAN baizik, ahalik eta kode errazenetik hasiz, garatu aurretik kode osoa lortu arte probatu
 eta arazketatuko dena.

- **1. bertsioa:** *sum1toN_v1.ias* iturburu-modulua :

- ÿ 1. bertsioak begizta bat implementatzen du, zeinaren gorputzak datu bat soilik gordetzen duen batura aldagaia. Datuak aldatu egiten dira iterazio bakoitzean.
- ÿ Sintaxia ÿ etiketa: eragiketa eragiketa ; iruzkina ÿ 4 zutabe
- ÿ Eragiketa adierazteko ikurrak (adibidez, $S(x) \rightarrow Ac+$) ez dira mnemoteknikoak
- ÿ Ez erabili azentu-markak iruzkinetan edo etiketetan, hori bakarrik onartzen baita hedatu gabeko ASCII kodea.
- ÿ Instrukzio kopurua bakoitia bada, azken instrukzioko 40 biteko hitza bete behar da. 20 bit esanguratsuenak zerora dituen instrukzioa.
- ÿ Argibideen atala datuen atalaren aurretik joan behar da

; 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

; Batuketak na-tik beherantz sortzen dira 0 Begizta irtengo da sumandoa Konstante bat. negatiboa bada $\rightarrow -1$; ; Datuen egiturak: n eta batura aldagaiak.

; Mihiztadura hizkuntza: IASSim ;

Von Neumann IAS makinen arkitektura

;,:,:,; JARRAIBIDEEN ATALA ;Metagailuei

Orientatutako 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 ≥ 0 bada, salto begiztara; geldialdiaren
 amaiera .begizta hutsa

; gelditu

;,:,:,; DATU ATAL

; Etiketa-deklarazioa, kanpoko memoria-erreserba, hasieratzea.
; Aldagai arruntak n: .datuak 5 ;
aldagai batuketa .datuak 1 ; cte .data 0 ;
bat: batuketa partzialaketa azken
gain:

ŷ Memoria erreserbatzeko datuen atala garatu da.

ŷ BEZKIZ SIMPLE BAT egin da begizta beharrezko den eraikuntza delako
azken algoritmoa.

ŷ SUB eragiketa egin da, azken algoritmoan beharrezko den eragiketa baita.

ŷ Kodea IRUZKINATU da

Erregistroak

- IAS makinak 7 erregistro ditu: Metagailua, Erregistro Aritmetikoa / Biderkatzailea-Kozientea (AR/MQ), Kontrol-kontagailua, Kontrol-erregistroa, Funtzio-taula-erregistroa, Memoria-helbide-erregistroa, Selectron Erregistroa

ŷ Metagailua (AC) eta erregistro aritmetikoak (AR/MQ) bi programatzale bakarrak dira
erregistro ikusgaiak

ŷ Kontrol-kontagailua gaur egun Programa-kontagailua deitzen *dioguna* da

ŷ Kontrol Erregistroak unean exekutatzen ari den *IBR instrukzioa gordetzen du*. Exekutatu beharreko eskubideari buruzko adierazpena bakarrik.

ŷ Funtzio-taularen erregistroak uneko *IR* kodea dauka

ŷ Memoriaren helbidea Erregistratu uneko memoria helbidea *MAR*

ŷ Selectron Erregistratu memoriatik irakurtzen edo idazten ari den uneko datu-balioa ŷ *MBR*

IASSim simulagailua

- IAS makinaren IASSim simulagailua instalatzeko eta funtzionatzeko jarraibideak
Von Neumann eranskinean

Kalifikazioak

- Beharrezko da argibide kopurua bikoitia izatea. Bitxia bada, *.empty zuzentaraua gehitzen da*.
- Etiketa batek ezkerreko instrukzioa adierazi behar du. Eskuinean badago a jar dezakezu etiketa horretara baldintzarik gabeko jauzi-instrukzioa.
- Datuen atalak, kodearen atala jarraitzen badu, datuen atala amaitu behar du.
kodea eskuineko instrukzio batekin eta ez bada *.hutsik zuzentara uarekin betetzen dugu*.

Akatsa

- Errore bat MAR erregistroaren balioa erakustean

ŷ Sum1toN.ias-en lehen instrukzioa exekutatzen denean, MAR edukia 28 da, hau baino handiagoa.
Programa kargatzen den memoria nagusiko helbide sorta.

ÿ Errorea Windows 7 eta Ubuntu 17.04n gertatzen da

2.5.3. IASSim hizkuntzako programen adibideak

- Adibide gehiago [Eranskinean](#).

2.6. IAS Makinaren Eragiketa: Datuen Bidea

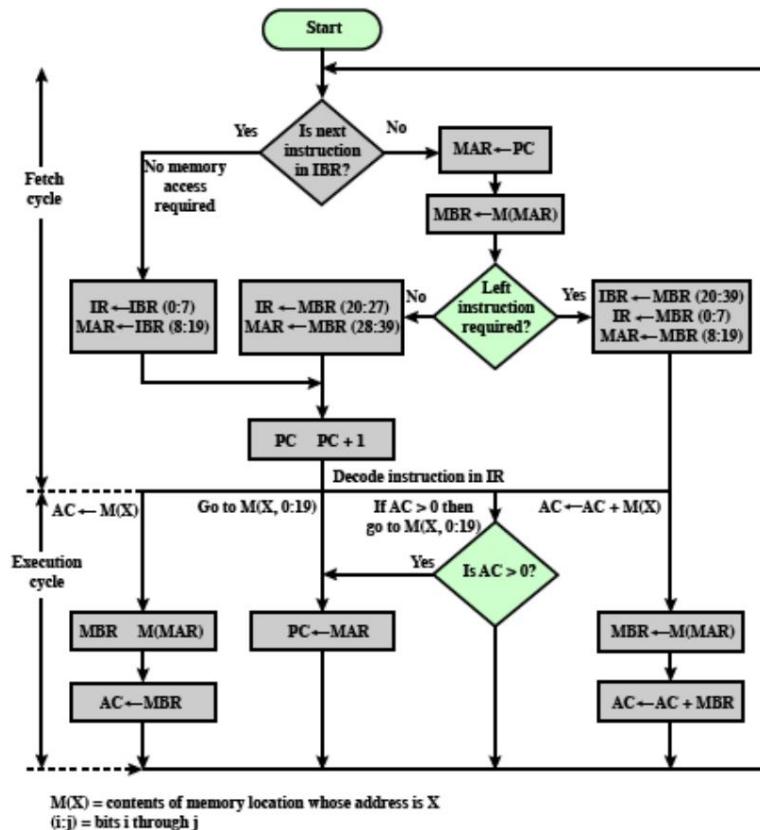


Figure 2.4 Partial Flowchart of IAS Operation

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 beharrezko da: ordenagailuaren egitura (memoria, erregistroak, etab.), datuen formatua (2 osagarriaren formatua, etab.) eta argibideak eta Instrukzioa. Ezarri (eragiketak, helbideratze moduak, etab...)
2. Mihiztadura lengoian ez da zuzenean egiten hizkuntza horretan, baina goitik beherako estrategia bat jarraitzen da, pseudokode hizkuntzan, organigraman, RTL hizkuntzan, etab. deskribapenarekin hasita.
3. Ordenagailuaren ISA argibide multzoaren diseinua da, maila baxuko programazioa erraz edo zaitzen duena. Von Neumann IAS makina bezalako errepertorio gehiegi mugatu batek zaila egiten du biderketa eta zatiketa bezain simpleak diren adierazpen matematikoak egitea. RTL instrukzioen sekuentzia kontuan hartu behar da algoritmoaren garapena erraztea.
4. Algoritmoaren programazioak mutua-lengoian goranzko estrategia bat jarraitzen du garatu nahi den programaren bertsio osatugabe eta errazenetik hasita.
5. Mihiztadura-lengoaiaren programaren bertsio garatu bakoitza arazketa eta egiaztatu 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 errealak

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. Makinaren lengoaia 0 eta 1 sinboloek osatzen duten lengoaia bitarra da. Horregatik, memoria nagusian gordetako programa baten datuak eta argibideak bi ikur horien bidez kodetu eta irudikatu behar dira.
- Datuek Unitate Logiko Aritmetikoa (ALU) prozesatu dezakeen zenbakizko balio bat dute, batuketa, kenketa, etab. edo not,or, etab bezalako eragiketa aritmetikoak egiteko.
- Datuak memorian gordetako 0 eta 1eko sekuentziak dira, CPUak prozesatzeko, adibidez, batuketa bezalako eragiketa aritmetikoen bidez jasotzen dituena.
- Instrukzioak memorian gordetako 0 eta 1eko sekuentziak dira, CPUak atzematen dituena interpretatu eta exekutatzeko. Adibidez, "gehitu" bi zenbaki agindua
zenbaki osoak.

3.4. Bit, Byte, Word

- BINARY DIGIT (bit): zifra bitarrak 0 eta 1 dira. Memorian gordetako programa baten instrukzioak zein datuak kodetzeko erabiltzen diren sinboloak dira.
- 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**

Iur

deskribapena: Koaderno
 produktua: 20F1S0H400 (LENOVO_MT_20F1_BU_Think_FM_ThinkPad L560)
 Fabrikatzailea: LENOVO
 bertsioa: ThinkPad L560
 seriea: MP15YSW7
 zabalera: 64 bit
 gaitasunak: smbios-2.8 dmi-2.8 smp vsyscall32
 konfigurazioa: **administrator_password=desgaitutako** xasis=koaderno **familia**
=ThinkPad L560 power-on_password= sku desgaituta
=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 hamartaran

ŷ Hamar zifren konbinazioaren bidezko irudikapena: 0,1,2,3,...,9

ŷ Posizioa ŷ indizea

ŷ Posizio bakoitzaren pisuak ŷ 10 posizio formako potentzia 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	3	bi	1	0
Pisua	10^3	10^2	10^1	10^0
	1000	100	10	1
Zifra	1	1	9	7
Pisatzea	$1 \cdot 1000$	$1 \cdot 100$	$9 \cdot 10$	$7 \cdot 1$

Merezi $1 \cdot 1000 + 1 \cdot 100 + 9 \cdot 10 + 7 \cdot 1 =$ mila ehun eta laurogeita hamazazpi.

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

3.5.2. Oinarri bitarra

- 2. oinarrian kodetutako zenbakia

ŷ Bi Zifra konbinatuz irudikatzea: 0,1

ŷ Posizioa ŷ indizea

ŷ Posizio bakoitzaren pisuak ŷ 2 formako potentzia dira
 $,64,32,16,8,4,2,1$

posizioa ŷ ... 25,24,23,22,21,20 ... _ _

ŷ Irudikaturiko balioa = beren posizio pisuarekin hiztatutako zifren batura

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

Posizioa	3	bi	1	0
Pisua	2^3	2^2	2^1	2^0
	8	4	2	1
zifrak		0	1	0
Pisatzea	1^8	0^4	1^2	0^1
Merezi	$1^8 + 0^4 + 1^2 + 0^1 = \text{hamar}$			
Bat-zero-bat-zero irudikapenak hamar balioa du				

3.5.3. Bihurketa hamartar-bitarra

- Ondoz ondoko zatiketak / 2 ŷ Dibidendua1 = $2^*Koientzia1 + hondarra1$
- Koientzia1 = $2^*Koientzia2 + hondarra2$ ŷ Dibidendua1 = 2 * $(2^*Koientzia2 + hondarra2) + hondarra1 = hondarra1*20$
+ Gainontzko2 * 21 + Koientzia * 22
- Gainerakoak 0 posizioko zifra bitarra da, hondarra2 1. posizioko zifra bitarra da, zatidura
2. posizioan dagoen zifra bitarra da.
- Araua: zifra bitarrak hondar guztia eta azken zatidura dira.
- Zatiketa amaitzen da zatidura bat 2z zatigarria ez denean, hau da, zatidura 1 denean.
zatidura MSB da.
- Adibidea: 1197 balioa ŷ Kalkulatu bere irudikapena kode bitarrean ŷ Soluzioa: 10010101101

3. Taula Bitar Bitarteko Bihurketa

zenbakia	1. Maila	2. Maila	3. Maila	4. Maila	5. Maila	6. Maila
ibaia						
	sukaldea	Atsedena Sukaldea				
1197 598	1	299	0	149	1	74

Zenbakia 7. Div	8. Maila	9. Maila	10. Maila
	sukaldea	Atsedena	sukaldea
1197	9	0	4

3.5.4. Oinarri octala

- 8. oinarria
- Zifrak: 0,1,2,3,4,5,6,7
- Pisuak: 8 posiziora igota
- C-n oinarria 0 ŷ int 077 aurritzarekin zehazten da ;
- Bihurketa octal ŷ Binarioa eta alderantziz ŷ zifra octal bakotza 3ko bitar batean deskonposatzen da

bitsak

- 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 Hex

- 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 bitar batean deskonposatzen da 4 bit
- 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 derrigorrezko da formatuaren oinarria lehenik jartzea
irteera
ÿ 2.158.625.039
ÿ $bc interpretea
```

3.5.6. pitoia

- <https://docs.python.org/3/tutorial/index.html>
 - ÿ laguntza (eraikitakoak)

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

3.5.7. Zenbaki oso sinatutakoak

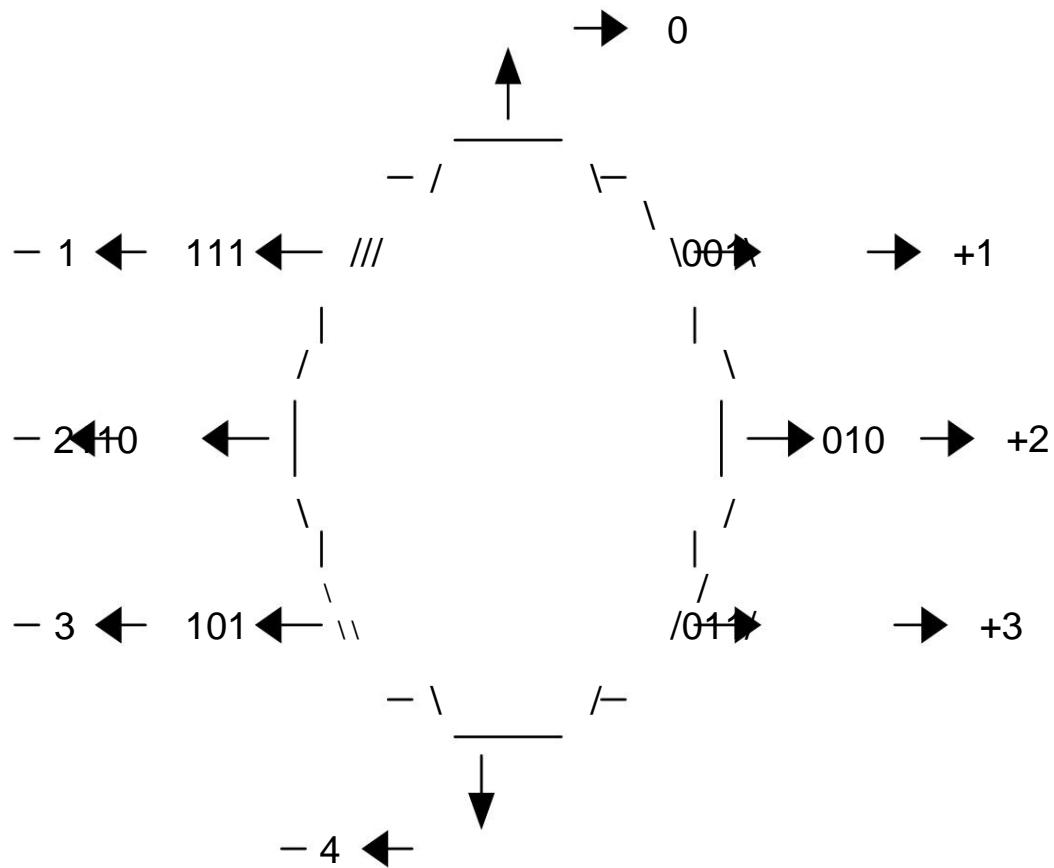
- Bi formatu aztertuko dira: Zeinu-Magnitudea eta 2-ren osagarria, azken hau gehien konputagailuen arkitekturan oso hedatua.

Zeinu-Magnitude

- Zeinu-Magnitude formatua
 - ÿ Bit esanguratsuenak ez du baliorik, zeinua adierazten du: zero zenbaki positiboentzat eta bat zenbaki negatiboentzat.
 - ÿ Gainerako 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?

2ren osagarria

- 2-ren osagarri formatua.
 - ÿ Positiboak: zeinu-magnitude formatuaren berdina: Bit MSB= 0. Pisuak: potentzia 2posizioa.
 - ÿ Bit esanguratsuagoa (MSB) ÿ zero ez den balio duen posiziorik altuena.
 - ÿ Bit esanguratsu gutxiago (LSB)
 - ÿ Negatiboa: Magnitude bereko baina positiboa duen zenbakiaren transformazioa
 - 2-ren osagarri funtzioa.
- Gurpila



29. Irudia 2-ren osagarria. Luzera 3 bit.

ÿ N: zenbakian dagoen bit kopurua: 3 bit

ÿ Zatitu zirkunferentzia konbinazio bitar posibleen kopurutan: $2N : 2^3 = 8$

ÿ Konbinazio bitar guztiak erlojuaren orratzen noranzkoan 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 Sorta positiboa eta negatiboaren arteko asimetria

ÿ Zerok irudikapen berezia du

ÿ Zenbaki negatiboak 1etik hasten dira

ÿ -1 balioa 1111111111111111 zifra guztiekin kodetuta dago

ÿ Zeinu luzapena: ezkerretik 1 bat positiboetan ezkerretik zero bat bezalakoa da: ez du balioa eta ezkerretik 1-ren errepikapena ezabatu dezakezu, gehieneko azken 1a utziz esanguratsua. 11110111 10111ren baliokidea da.

- 2-ren **osagarriaren funtzioa**: zenbaki oso baten 2-ren osagarria aldatzearen baliokidea da. sinatu. 2 osagarrian zenbaki oso positibo eta negatiboen arteko bihurketa izan daiteke metodo ezberdinekin egin.

- X osoko bitarraren 2 osagarria lortzeko adibideak:

a. 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} -ren balioa
2-ren osagarria 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

$\ddot{y} C2(X)=10011000111010101111+1=1001100011101010110000$ \ddot{y} negatiboa, bit esanguratsuena (MSB)
bat delako

b. 2. metodoa: X kodea 0 posizioan hasita (X0 bit) kopiatu zifra guztiak lehen zifrara arte eta hortik ezeztatu zifra
guztiak bit esanguratsuenera (MSB).

$\ddot{y} X = 0110011100010101010000$ \ddot{y} guztira 22 bit

$\ddot{y} X$ -ren 1 lehenengo zifra 4. posizioan dago $\ddot{y} 01100111000101010-10000$ \ddot{y} Lehenengo 5 zifrak kopiatzen
ditut eta gainerako 17ak alderantzizatzen ditut

$\ddot{y} C2 (X) = 10011000111010101-10000$

c. 3. metodoa: Egin 0-X eragiketa aritmetikoa

$\ddot{y} X = 0110011100010101010000$

$\ddot{y} 0-X=0000000000000000000000000000$ - $0110011100010101010000 = 0110011100010101010000$

- 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-ren osagarriaren 8 biteko zenbaki osoen barrutia

\ddot{y} Gehieneko kode positiboa: 0111111 \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} Tartea [-27 ,+27 -1]

3.6. Zenbaki errealkak

3.6.1. Koma finkoa

- Zenbaki errealkak puntu finkoan:

$\ddot{y} 1234.56789$

\ddot{y} Posizio-sistema

ŷ zatiki zifren posizioa: -1,-2,-3,...

ŷ zatiki zifren pisuak: 10-1, 10-2, 10-3 ŷ 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

ŷ 1010.101 ŷ 1*23 + 0*22 + 0*21 + 1*20 + 1*2-1 + 0*2-2 + 1*2-2 ŷ 10.625

3.6.2. koma flotagarria

Formatua

- Koma flotagarria ŷ Notazio zientifikoa

ŷ -23.4567E-34 edo -23.4567*10-34

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

ŷ Mantisa **normalizatua** : mantisak osoko zati gisa zifra bateko zenbaki oso bat du zero ez dena.ŷ -2,34567*10-33

ŷ mantisa normalizatuaren zati osoa: 2

ŷ mantisaren zati normalizatuaren zatia: 0,34567

ŷ **Berretzailea** potentziaren oinarria altzatzen den zenbaki osoa da. Mantisan komak duen lekuaren araberakoa da. Kasu honetan -33 da.

ŷ Oinarria potentziaren oinarria da . Adibide honetan 10 da.

- Kodetze bitarra

ŷ Adibidea: 1234.56789

ŷ Osoko zatia: 1234 ŷ 10011010010

ŷ Zatiki zatia: 0,56789

0,56789 * 2 = 1,13578 = 1 + 0,13578 -> 1, bit **-1** posizioan **0,13578**
*** 2 = 0,27156** -> 0, bit **-2** posizioan **0,27156 * 2 = 0,54312** -> **3** * posizioan bit **1,08624 = 1 + 0,08624** -> 1, bit **-4** posizioan

ŷ frakzio biribildua 0,1001

ŷ zatiki biribildugabea 0,10010001011000010

ŷ frakzio biribildua 0,100100011

ŷ Kode finko bitarra: 10011010010.10010001011000010

ŷ Idazkera zientifikoa: 1,001101001010010001011000010*2+10 ŷ koma higikorra ŷ zati osoaren berdina da beti.

Zehaztasuna

- Zifra esanguratsuen kopurua da

- q zenbakia $p \neq 0$ zenbakien hurbilketa dela esaten da, gutxienez, zehaztasunarekin b oinarriko **m** zifra esanguratsuak, baldin eta errore erlatiboa $|pq|/p \geq 0,5^*b-m+1$
 - ŷ m aurreko inberdintasuna betetzen duen zenbaki osorik handiena denean, q dela esaten dugu p gutxi gorabehera m zifra esanguratsuekin.
- Adibidea
 - $p = 1E0$ eta $q = .9999E0$ ŷ Errore erlatiboa $= 0.1E-3 < 0.5E(-4+1)$ ŷ 4 zifren doitasuna esanguratsua
 - Kalkulagailu batek, A, 2. oinarrian funtzionatzen du 22 biteko mantisarekin eta beste batek, B, 16. oinarrian lan egiten du. Zehaztasuneko 6 digitu (24 bit). Bietatik zein da zehatzagoa?

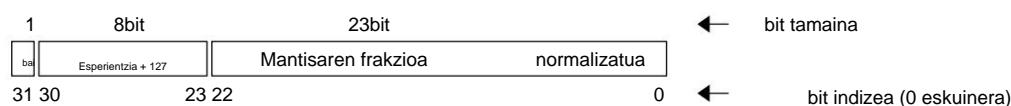
IEEE-754 estandarra

- flotatu

ŷ IEEE-754 estandarra

ŷ Doitasun bakarra ŷ 32 biteko formatua 3 eremutan Zeinua/Berretzailea/Frakzioa

1/8/23 biteko luzerak



ŷ Doitasun bikoitza ŷ 64 biteko formatua 3 eremutan Zeinua /Berretzailea/Frakzioa

luzerak 1/11/52 bit



ŷ [eskuzko bihurketa](http://sandbox.mc.edu/~bennet/cs110/flt/dtov.html) [<http://sandbox.mc.edu/~bennet/cs110/flt/dtov.html>]

ŷ [wiki](http://en.wikipedia.org/wiki/Floating_point) [http://en.wikipedia.org/wiki/Floating_point]

ŷ Bitarra: Hiru eremu

Sinatu	barruan erakuslea Gehiegizkoa	Mantisaren frakzio normalizatua
--------	----------------------------------	---------------------------------

ŷ Balioa (-1)Zeinua x 1.Frakzioa_Mantisa_Normalizatua x 2 Berretzailea

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

ŷ Gehiegizko berretzailea: 127 (zehaztasun bakarra) edo 1023 gehitzen zaion Berretzailea da.
(zehaztasun bikoitza)

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

ŷ Mantisa normalizatuaren frakzioa: Mantisa normalizatuaren frakzioa da.

Kode bihurgailuak

- Sareko bihurgailuak:

ÿbihurgailu [bitarra](http://www.binaryconvert.com/index.html) [<http://www.binaryconvert.com/index.html>]: motak char,short,int,float,double

ÿ [ieee754 bihurgailua](http://www.zator.com/Cpp/E2_2_4a1.htm) [http://www.zator.com/Cpp/E2_2_4a1.htm]

ÿIEEE [754 doitasun bakarrekoa](http://www.h-schmidt.net/FloatConverter/IEEE754.html) [<http://www.h-schmidt.net/FloatConverter/IEEE754.html>]: hamartar ÿ bitarra/hamaseitarra eta alderantziz

Float Point: Zero, Infinity eta Indeterminate irudikapena

- Berretzaile-eremua zero edo bat direnean, a-ren arau orokorra zenbaki normalizatua

4. taula. Doitasun bakarra

Zenbakiak	exp	Zatikia
Zeroak	0x00	0
desnormalizatutako zenbakiak	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 den zero doitasun bakarrean 32 zeroren segida gisa

ÿ Zergatik berretzaile-eremua zero denean potentzia 2-126koa da 2-127 izan beharrean eta mantisa EZ normalizatutzat hartzen da, hau da, 0.frakzioa 1.frakzioa izan beharrean.

- [Maryland oharrak](http://www.cs.umd.edu/class/sum2003/cmsc311/Notes/Data/float.html) [<http://www.cs.umd.edu/class/sum2003/cmsc311/Notes/Data/float.html>]

- Infinitua

Erreferentzia

- [zenbakizko analisia](http://people.ds.cam.ac.uk/nmm1/arithmetic/na1.pdf) [<http://people.ds.cam.ac.uk/nmm1/arithmetic/na1.pdf>]: adibide simpleko programak
- [IEEE](http://grouper.ieee.org/groups/754/) [<http://grouper.ieee.org/groups/754/>]
- [William Kahan](http://www.cs.berkeley.edu/~wkahan/) [<http://www.cs.berkeley.edu/~wkahan/>]
- [Yale](http://www.cs.yale.edu/homes/aspnes/pinewiki/C%282f%29FloatingPoint.html) [<http://www.cs.yale.edu/homes/aspnes/pinewiki/C%282f%29FloatingPoint.html>]: C programazioa. karroza.c.
- [Bruce Dawson bloga](https://randomascii.wordpress.com/category/floating-point/) [<https://randomascii.wordpress.com/category/floating-point/>]
 - ÿ <https://randomascii.wordpress.com/2012/01/11/tricks-with-the-floating-point-format/>
- [Wikipedia](https://en.wikipedia.org/wiki/IEEE_floating_point) [https://en.wikipedia.org/wiki/IEEE_floating_point]
 - ÿ <http://www.validlab.com/goldberg/paper.pdf>
 - ÿ [Oating-point-en kalkuluak egiazatzeko akatsak](https://hal.archives-ouvertes.fr/hal-00128124v5/dokumentua) [<https://hal.archives-ouvertes.fr/hal-00128124v5/dokumentua>]

ŷ berdin? [<http://arxiv.org/pdf/cs/0701192.pdf>]

- [programazioa ulertzen_puntu_flotante_errepresentazioa.html](#) [http://www.cprogramming.com/tutorial/floating_point/]
- [kodea aholkuak](#) [http://www.codeproject.com/Articles/29637/Five-Tips-for-Floating-Point-Programming]
- [c berrikuspena](#) [https://www.cs.princeton.edu/courses/archive/fall09/cos323/precepts/precept2.html]: praktikak

3.7. Karaktere mota

3.7.1. ASCII

- ASCII kodeketa

ŷ American Standard Code International Interchange: 7 biteko kodeketa: 0x00-0x7F tartea

ŷ Karaktere-kode_hamaseimala-kode bitarra bihurtzeko taula

ŷ [gizon ascii](#)

ŷ KN King,E eranskina, pg801

Pertsonaia	ASCII hex kontrola (Sekuentzia Ihesketa)	
0	0x30	
1	0x31	
a	0x61	
A	0x41	
+	0x2B	
^J	0x0A	lerro berria (\n)
^M	0x0D	itzulera autoa (\r)

ŷ konpondu J eta ^J arteko kodearen erlazioa, M eta ^M artean...

C programaren '\X' ihesak adierazten dira.

Urr Abendu Hex Char

Urr Abendu Hex Char

000 0	00	NULL '\0'	100 64	40	@
001 1	01	SOH (izenburuaren hasiera)	101 65	41A	
002 2	02	STX (testuaren hasiera)	102 66	42	B.
003 3	03	ETX (testuaren amaiera)	103 67	43	C
004 4	04	EOT (transmisioaren amaiera)	104 68	44	D

005 5 006	05	ENQ	105 69 106	45	
6	06	ACK (onarpena)	70	46	F
007 7	07	BEL '\a' (kanpaina)	107 71	47	G
010 8 011	08	BS '\b' (atzerantz)	110 72 73	48	H
9	09	HT '\t' (fitxa horizontala) 111 LF '\n' (lerro berria)	49		yo
012 10	0A		112 74	4A	J
013 11 014	0B	LH '\v' (fitxa bertikala)	113 75 114	4B	K
12 015 13	0C	FF '\f' (inprimaki-jarioa)	76 115 77	4C	L
	0D	CR '\r' (karroza ret)		4D	M
016 14	0E	SO (kanpora)	116 78	4E	N
017 15 020	0F	BAI (aldaketa)	117 79 120	4F	
16 021 17	10	DLE (datuen estekaren ihesa)	80 121 81	50	P
	11	DC1 (gailuaren kontrola 1)		51	G
022 18	12	DC2 (gailuaren kontrola 2)	122 82	52	R
023 19 024	13	DC3 (gailuaren kontrola 3)	123 83 124	53	
20 025 21	14	DC4 (gailuaren kontrola 4)	84 125 85	54	T
	15	NAK (erantzun negatiboa)		55	eo
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	Y
032 26	1A	SUB (ordezkoa)	132 90	5A	Z
033 27 034	1 B	ESC (ihesketa)	133 91 134	5B	[
28 035 29	1 C	fs (fitxategien bereizlea)	92 135 93	5C	\\"\\'
	1D	GS (taldeen bereizlea)		5 D]
036 30	1E	RS (erregistroen bereizlea)	136 94	5E	^
037 31 040	1F	AEB (unitate-bereizlea)	137 95 140	5F	-
32 041 33	20	ESPAZIO!	96 141 97	60	
	21	"		61	a
042 34 043	22		142 98 62 143 99 63		b
35 044 36	23	#	144 100 64		c
	24	\$			d
045 37	25	%	145 101	65	
046 38 047	26	&	146 102 66 147 103		F
39 050 40	27		67 150 104 68		g
	28				h
051 41	29	()	151 105 69		yo
052 42 053	2A	*	152 106 6A		j
43 054 44	2B	+	153 107 6B		k
	2 C	,	154 108 6C		baik
055 45	2D	-	155 109 6D		m
056 46 057	2E	.	156 110 6E 157 111		n
47 060 48	2F	/		6F	
	30	0	160 112 70		or
061 49	31	1	161 113 71		zer
062 50 063	32		162 114 72 163 115		r
51 064 52	33	2.	73 164 116 74		s
	34	3. 4			zuk
065 53	35	5	165 117 75		ado
066 54 067	36	6	166 118 76 167 119		v
55	37	7	77W		

070 56	38	8	170 120 78 171	x
071 57	39	9	121 79	Y
072 58	3A	:	172 122 7A	z
073 59	3B	;	173 123 7B	{
074 60	3C	<	174 124 7C	
075 61	3D	=	175 125 7D	}
076 62	3E	>	176 126 7E	~
077 63	3F	?	177 127 7F	OF

- ASCII hedatua

ÿ https://en.wikipedia.org/wiki/Extended_ASCII#ISO_8859_and_proprietary_adaptations

ÿ man iso_8859_1: latin-1: ascii hedatua: 0x80-0xFF

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

ÿ Linux-ek ctrl-Shift-u-ascii_code sakatu Sartu

ÿ Adibidea: ñ -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 [<http://www.mauvecloud.net/charsets/CharCodeFinder.html>]

ÿ0

ÿ-

ÿ-

ÿñ

3.7.2. pitoia

- bihurketa adibideak

ÿpython _

```
ordena('A')
hex (ord('A'))
chr (65)
chr(0x41)
[hex(ord(c)) c - rako "Kaixo"]
[chr(c) c - rako [0x48, 0x6f, 0x6c, 0x61, 0x20, 0x4d, 0x75, 0x6e, 0x64, 0x6f]]
```

3.7.3. Unicode UTF-8

- [UnicodeMain](https://www.unicode.org/main.html) [<https://www.unicode.org/main.html>]
- Unicode: Unicode karaktere-kode ezberdinaren bidez implementa daiteke. Unicode estandarra
Unicode transformazio formatuak (UTF) definitzen ditu: UTF-8, UTF-16 eta UTF-32, eta beste hainbat
kodeketak. Gehien erabiltzen diren kodeketak UTF-8, UTF-16 eta UCS-2 zaharkitua (a
UTF-16ren aitzindaria, Unicoderako laguntza osoa gabe)

- Unicode kodetua: <https://www.unicode.org/versions/Unicode14.0.0/ch02.pdf#G25564>

ÿ U+ aurizkiarekin eta ondoren zenbaki oso batekin deskribatzen da (0tik 0xFFFF bitarteko osoak). Kodeari **code point** deitzen zaoi "

- 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 puntuarako, eta gehienez 4 byte beste karaktereetarako.

Lehenengo 128 Unicode kode puntuak ASCII karaktereak adierazten dituzte, hau da, edozein ASCII testu UTF-8 testu bat ere bada.

ÿ ñ - ak 0xc3b1 ateratzen du. Terminala Unicode UTF8 irteerarekin konfiguratuta dago ingurune lokaleko aldagaien oinarrituta. **Local charmap** komandoaren bidez terminalaren sarrera/irteera zer kodeketarekin iraultzen dugu. **Local -m** bidez posibleak. Iso-8859-1 izan zitekeen utf8-ren ordez.

- `localectl --status` ÿ teklatuaren sarrerako kodeketa

```
Sistemaren lokalizazioa: LANG=eu_ES.UTF-8
                           LANGUAGE=eu_ES:eu:en_GB:en
VC tekla-mapa: n/a X11 Diseinua: es X11 Modeloa:
pc105
```

- [utf8](http://www.utf8-chartable.de/unicode-utf8-table.pl?number=1024) [http://www.utf8-chartable.de/unicode-utf8-table.pl?number=1024]:

ÿ 8 biteko Unicode eraldaketa formatua

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

ÿ Byte bateko hitzen transmisiora bideratzen da.

ÿ [unicode](http://www.fileformat.info/info/unicode/char/f1/index.htm) [http://www.fileformat.info/info/unicode/char/f1/index.htm]

ÿ ñ Unicode puntu *U+00F1* edo hex_code_utf8 *0xC3B1* du

ÿ wikipedian utf-8-k 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. ÿ C3B3 ó karakterearen utf-8 kodearen kode hamaseitarra da.

ÿ Wikipedia utf-8:

ÿ Kodeen 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 sakatu arte itxarongo du eta sakatutako letraren kodea bistaratuko du sistema eragilearen sarrerako kodifikazioan.

ÿ ASCII karaktere estandarren kodea (7bit) bat dator UTF8rekin baina ez gainerako ASCII karaktere hedatuak.

ÿ erabilgarria karaktere bakoitzaren kodea ascii estandarrean eta karaktere kodea ñ, á, é, í, ó, ezagutzeko.

ú sistema konfiguratuta dagoen kodean bada (UTF8)

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

```
\92 0134 0x5c -> ESC tekla: ihes
^J 10 0012 0x0a -> Ctrl-CR teklak: lerro-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: Esanahi gutxiagoko byte
```

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

- HTML dokumentuak

ÿ ñ ÿ ñ ÿ erabili "unicode point" kodea

- Sistema eragilea: ingurune-aldaagaiak

ÿ [bidali | grep LC_](#)

- [Unicodechart](#) [<http://www.unicode.org/charts/>]

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

ÿ Sinboiloak Puntuazioa:

ÿ Puntuazioa: ASCII Puntuazioa: [U0000.pdf](#) [<http://www.unicode.org/charts/PDF/U0000.pdf>]

ÿ Aurkitu diagrama kode hexadesimalaren arabera: 278a

ÿ Piktogramak: Dingbats: ÿ ÿ [U2700.pdf](#) [<http://www.unicode.org/charts/PDF/U2700.pdf>]

ÿ Ikur matematikoak: Eragile matematikoak:

ÿ [Wikipedia](#) [https://en.wikipedia.org/wiki/Mathematical_operators_and_symbols_in_Unicode]

ÿ [U2200](#) [<http://www.unicode.org/charts/PDF/U2200.pdf>]

ÿ x2228 hamaseitarra ÿ ÿ edo 8744 hamartarrez ÿ

ÿ ÿ

ÿ ÿ

ÿ ÿ

ÿ beste batzuk

ÿÿ

ÿ ñ

ÿ ñ

ÿÿ

ÿÿ

- https://wiki.mozilla.org/Help:Special_characters#Unicode

- [Unicode karaktere bat buruzko informazio zehatza](http://www.fileformat.info/info/unicode/char/305/index.htm) [http://www.fileformat.info/info/unicode/char/305/index.htm]: Peg U+0305
- [Microsoft Office](https://support.office.com/en-us/article/Insert-ASCII-or-Unicode-Latin-based-symbols-and-characters-d13f58d3-7bcb-44a7-a4d5-972ee12e50e0) [https://support.office.com/en-us/article/Insert-ASCII-or-Unicode-Latin-based-symbols-and-characters-d13f58d3-7bcb-44a7-a4d5-972ee12e50e0]
- [Gainetik edo azpimarratuta](https://en.wikipedia.org/wiki/Overline) [https://en.wikipedia.org/wiki/Overline]
 - ÿ LibreOffice-k zuzeneko euskarria du **formatu/karaktere/letra** -tipoan gain-lerroaren hainbat estilotarako
Efektuak" elkarrizketa-koadroa: azpimarratuta

3.8. ISO-8859-1

- Latin alfabetorako UTF-8ren alternatiba
- 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 euroaren ikurra
- [gizon iso_8859-1](#)
- "ñ"-ak 0xF1 kodea du

3.8.1. C programazioa

- Bihurtu zenbakizko karaktere bat bere balio osora
 - ÿ Eragiketa aritmetiko bat erabiliz
- Bihurtu karaktere minuskula bat maiuskula
 - ÿ Eragiketa aritmetiko bat erabiliz

3.8.2. Beste batzuk

- [C hizkuntza: printf](https://www.gnu.org/software/coreutils/manual/html_node/printf-invocation.html) [https://www.gnu.org/software/coreutils/manual/html_node/printf-invocation.html]
 - ÿ **locale -a** ÿ C.UTF8
 - ÿ ñ ÿ **env printf \u00f1 \n** : komatxo bakunak sartu
 - ÿ **printf deitza** [http://www.gnu.org/software/coreutils/manual/html_node/printf-invocation.html#printf-deialdia]

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

- Zenbaki natural eta osoekin batuketa eta kenketa eragiketa aritmetikoak kode bitarra.
- Kode bitarrean irudikatutako datu logikoen eragiketak.
- Testu liburua
 ÿ 3. zatia, 10. kapitulua: Konputagailuen aritmetika

4.3. Sarrera

- Unitate Logiko Aritmetikoa (ALU) batuketa eta kenketa bezalako kalkulu-eragiketa aritmetikoak egiteaz eta EZ, EDO, ETA eta abar bezalako boolean motako eragiketa logikoak egiteaz arduratzen den oinarrizko hardware-unitatea da.

4.4. Aritmetika Bitarra

4.4.1. 2. moduluko gehiketa (bitarra) bitar hutsean (ZENBAKIA NATURALA)

- Zenbaki naturalek ez dute zeinu baten marka, denak positiboak baitira: 0,1,2,3....
- Kode bitar hutsean datuak gehitzea
 ÿ Funtzionamendu kontzeptu **modularra**
 ÿ Adibidea: 100.000 modulua ÿ Zirkunferentzia bidezko interpretazio grafiko modularra.
 Zer gertatzen da autoaren bidaia-kontagailuan 99.999ra iristen garenean.
 ÿ 2. moduluko batuketa bitarra. Eramatea (eramatea) 2 baliora iristen edo gainditzen denean gertatzen da.
 ÿ $1+1=bat$ gehi bat = bi ≥ 2 ÿ bi balioari 2 modulua kentzen diot ($2-2=0$) eta bat hartzen dut. 2 balioa bitarrean 1 0 gisa adierazten da, non zero batuketa zifraren posizio berean dagoen irudikapena den eta 1 hurrengo posiziora eramatzen den.
 ÿ $1+1+1=bat$ gehi bat gehi bat = hiru ≥ 2 ÿ Hiru balioetik 2 modulua kentzen dut ($3-2=1$) eta bat hartzen dut. 3 balioa bitarrean 1 1 gisa adierazten da, non eskuineko 1 batuketa zifraren posizio berean dagoen irudikapena den eta ezkerreko 1 hurrengo posiziora 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 *****
Emaitzza -->	1 0 1 1 0 1 1 0 <-batura

Gainezka edo gainezka

- Batuketak edo kenketak gainezka egin duela esaten da:

- ŷ Adierazi beharreko emaitzaren balioa tartetik kanpo dago kopuruaren muga dela eta zifren
- ŷ Eragiketa aritmetikoaren emaitza hitzak onartzen duen tamaina baino handiagoa da gordeta dagoen memoria edo erregistroa.
- ŷ Konponbidea datuak adierazten dituzten zifren kopurua handitzea izango litzateke, baina ez beti Ahal da.
- ŷ Logikoki, gainezkatzea gertatzen bada, ALUk emandako emaitza ez da Eskuin. ALU-k gainezkatze bandera edo banderin bat du OF (gaietik bandera) hori pixka bat gordetzen du. OF=1 bitak esan nahi du ALUk egindako azken eragiketak duela gainezka eginda. Programatzailak gainezkatze-erroren bat egon den esan dezake irakurri DE banderola.

- Adibideak:

- ŷ ALU unitateak "1 byte" bi sarrera-erregistro ditu, AL eta BL, bakoitza non bi datu gordetzen ditu: 10011011 eta 10011011. 1eko irteera-erregistroa ere badu byte, CL. Kalkulatu baturaren emaitza formatu bitar hutsean: CL ŷ AL+BL eta balioa OF banderirraren.

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	bi 1 3 2 1 3 2 *****
Emaitzza --> 1 0 0 1	1 0 1 1 0 <-batura
CL : 0 0 1 1 0 1 1 0	
HONEK: 1	

- ŷ Gaintze-errorea ALUk baturaren emaitza kalkulatu duelako: 00110110

ŷ Arrakastaren emaitza 100110110 8 biteko erregistro baten barrutitik kanpo dago. Sorta onartzen dira 0000000 eta 1111111 arteko zenbakiak, hau da, balioak 0 eta 255 artean. 310 balioa duen 100110110 datuak. Irtenbidea diseinatzea litzateke. CPU berri bat, hitzaren tamaina byte 1 baino handiagoa den erregistroak dituena, adibidez 2 byte. Beraz, ALUren sarreran AX=0000000010011011 BX=0000000010011011 badugu , CX ŷAX+BX eragiketaren emaitza 0000000100110110 izango litzateke eta OF=0 ŷ errorerik ez gainezka.

4.4.2. 2 modulo (bitarra) kendu bitar hutsean

- Bi zenbaki natural (zeinurik gabe) kendu ahal izateko beharrezko da minuendoaren balioa izatea. sustraentearena baino handiagoa.

$$\hat{y}0-0 = 0$$

$$\hat{y} 1-1 = 0$$

$$\hat{y} 1-0 = 0$$

ŷ Zer gertatzen da 0tik 1 kendu behar badut? 0 balioa ezin zaio kendu 1 balioari. Noiz "p" posizioan dagoen minuendaren zifra bat "p" posizio bereko zifra baino txikiagoa da kenkariaren, soluzioa p posizioaren minuendoari modulu gehitzea da (2 bitarrean) eta, aldi berean, balio bera gehitzen dio kenkariari baina bidez "p+1" posizioa, eta horrekin balio bera gehitzen badiogu bai minuendoari bai minuendoari kenketaren emaitza kentzeak ez du eragiten.

ŷ "p" posizioa: minuendoa 0 - substrahend1 ŷ Minenendoan 0+modulua-1=0+2-1= 1. 2 balioa "p" posizioan "p+1" posizioan dagoen 1 balioaren baliokidea da. "p+1" posizioan 1 gehituko dugu kendura.

ŷ "p" posizioa: minuendoa 0 - 1 kentza - 1 eramana ŷ 0 minuendoan + modulua-1-1=0+2-1-1=0 eta kendutako hurrengo posizioari gehituko diogun 1 hartu.

ŷ "p" posizioa: 1-1-1 ŷ kendura 1+modulua-1-1=1+2-1-1= 1 eta gehituko dugun 1 eraman. substrahendaren hurrengo posizioa.

$$\hat{y} 10110110 - 10011011 = 00011011$$

Gehitu kreditua minuendari		2 2	2 2
	1 0 1 -	1 0 1 1 0 <--minuend	
	1 0 0 1	1 0 1	1 <--substrahend
Gehitu kendutako kendura		1 1 1 1	
	*****	*****	
Kenketa	0 0 0 1	1 0 1	1

4.4.3. Gehitu/Kendu 2 modulo (bitarra) 2 osagarrian

- Berrikusi 2-ren osagarri formatua sinatutako zenbaki osoetarako

gainera

- Egin 00100101 eta 0111 byte osoko 2 osagarrien gehiketa.
- Bi datuak zerorekin hasten dira, gero positiboak dira 2 osagarriaren formatuaren arabera
 - ÿ Gehigarriak hedatzen ditut denak tamaina berekoak izan daitezten. 0111 0 zeinu-bit hedatuz 00000111 da

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-k kenketak egiten ditu batuketa eragiketaren bidez eta eragigaien zeinua aldatuz.
 - ÿ Adibidea: 2-ko osagarrian 27-101 kendu byte bateko erregistroak erabiliz

lehenik +27 minuendoa eta +101 azpia kodenzen ditut	
+27 -> 00011011	
+101 -> 01100101	
-101 +101 -> 10011011-ren 2 osagarria da	
Eragiketa (-101)+27 -> 10011011+00011011 baturaren baliokidea da	
Eramanak -->	1 1 1 1
	1 0 0 1 1 0 1 1 <--AL
	+ 0 0 0 1 1 0 1 1 <--BL
batura balioa	1 3 2 1 3 2

Emaitza -->	1 0 1 1 0 1 1 0 <--batura

ÿ Zein da emaitzaren balioa?

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

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

ŷ errepikatu eragiketa ordenagailuak aldatuz eta 2 byteko erregistroak erabiliz. Eraiki aurreko atalean.

10011011 zenbaki negatiboaren zeinu-bitak 16 bit arte luzatzen dut

AX <- 111111110011011 (-101)

00011011 zenbaki positiboaren zeinu-bitak 16 bit arte luzatzen dut

BX <- 0000000000011011 (+27)

10110110 emaitza negatiboaren zeinu-bitak 16 bit arte luzatzen dut 16 bit-ak osatu arte

CX <- 111111110110110 (-74)

Gainezka 2 osagarrian (C2)

- Gainezkatzea edo gainezkatzea batuketa eta kenketa aritmetika eragiketetan gertatzen da eragiketaren emaitza irudikapen posibleen barrutik kanpoko tamainakoa denean, ondorioz, ondoriozko balioa ez da baliozkoa eta akatsak eragiten ditu.

ŷ Gehitzeko adibidea 2 byteko erregistroak erabiliz: $10000000+10000000 = 00000000$ ŷ
Gainezka

ŷ Huts egin du -128-128 ez baita zero.

ŷ Bi gehigarriak negatiboak badira, emaitza ezin da positiboa izan.

Emaitza zuzena izan dadin, byte bat baino tamaina handiagoa duten erregistroak erabili beharko genitzuke, adibidez 9 bit. Kasu honetan, eragiketa berriro egiten dugu, datuak 1 bit gehiago luzatuz: $110000000+110000000 = 100000000$ -> ez dago gainezkarik -> bi zenbaki negatiboren batura negatiboa izan da

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

ŷ Bi gehigarriak positiboak badira, emaitza ezin da negatiboa izan.

- Intelx86-k gainezkatze-errorea eragiten du eragiketa aritmetiko baten emaitzarekin MSB bitaren garraiatzeak emaitzaren balioari eragiten dio.

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. Batuketa 16. moduluaren (hamasezimala)

- 16. moduluaren gehitzea:

ÿ Eramatea modulo-balioa lortzen edo gainditzen denean gertatzen da: 16.

ÿ 0xF+0x1 = 0x10

ÿ F+1=hamabost gehi bat = hamasei ≥ 16 ÿ hamasei ($16-16=0$) emaitzari 16 kendu eta lortu bat daramat.

ÿ 0x3AF+0xA = 0x3B9

ÿ F+A=hamabost gehi $10 = 25 \geq 16$ ÿ emaitzari hogeita bost 16 kentzen diot ($25-16=9$) eta hartzen dut a

ÿ 0x3A1F+0xF4E1=0x12F00

ÿ F+1=hamabost gehi $1 = 16 \geq 16$ ÿ hamasei emaitzari 16 kendu ($16-16=0$) eta hartu bat.

4.4.5. Kenketa 16. moduluan (Hamaseitarra)

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

- 16. moduluko kenketa:

ÿ eramatea gertatzen da minuendoko p posizio bat sustraendoaren p posizio bera baino txikiagoa denean, eta kasu horretan, beharrezkoa da minuendoari 16 moduloa gehitu eta azpirendearren hurrengo p+1 posiziora eraman. :

ÿ 0x4308 - 0x1ABC = 0x

<table style="width: 100%; border-collapse: collapse;"> <tbody> <tr> <td style="text-align: right; padding-right: 10px;">0x 4 3 0 8 <- Minuendoa</td><td></td></tr> <tr> <td style="text-align: right; padding-right: 10px;">- 0x 1 ABC <- Subrahend</td><td></td></tr> <tr> <td style="text-align: right; padding-right: 10px;">jantzita --></td><td style="text-align: center; padding: 0 10px;"> 1 1 1 **** </td></tr> <tr> <td></td><td style="text-align: center; padding: 0 10px;">0x 2 8 4C</td></tr> </tbody> </table>		0x 4 3 0 8 <- Minuendoa		- 0x 1 ABC <- Subrahend		jantzita -->	1 1 1 ****		0x 2 8 4C
0x 4 3 0 8 <- Minuendoa									
- 0x 1 ABC <- Subrahend									
jantzita -->	1 1 1 ****								
	0x 2 8 4C								

ÿ 8-C ÿ 8+modulo_16-12=8+16-12=12=0xC eta eraman 1 hurrengo posiziora

ÿ 0-B-LCarried ÿ 0+module_16-11-1=0+16-11-1=4=0x4 eta 1 eraman zuen hurrengo posiziora

ÿ 3-A-Carried ÿ 3+modulo_16-10-1=3+16-10=8=0x8 eta 1 eraman zuen hurrengo posiziora

ÿ 4-1-Korrika ÿ 4-1-1=2

Oinarri hamaseimaleko gehiketa 2 osagarri formatuan

- 0xEC+0xAB=0x97

ÿ Bitarrean MSB bitak 1 izatea esan nahi du balioa negatiboa dela

ÿ Bi gehigarriak eta emaitza negatiboak dira

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

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

ÿ C2 0xAB-tik ÿ 0x54+1 ÿ 0x55

ÿ C2 0x97 ÿ 0x68+1 ÿ 0x69

8. oinarria gehitzea (oktala)

- 8. moduluko batuketa. Eramanketa 8 zifraren baliora iristean edo pasatzean gertatzen da .
 - ÿ 08+01 = 010
 - ÿ 0377+06 = 0305

4.4.6. C-ko aldagai motak

- Zenbaki osoak

ÿchar

ÿ laburra

ÿint

ÿ luzea

- Erreala

ÿ flotatu

ÿbikoitza

- sizeof() eragilea

- Mota bihurketa

ÿ casting

4.5. Eragiketa Logikoak

4.5.1. BITWISE operadoreak

- Bitean: bitaren araberako eragiketak

ÿ ez, eta, edo, xor

C hizkuntza

- https://www.salesforce.com/us/developer/docs/apexcode/Content/langCon_apex_expressions_operators_understanding.htm
- [Aljebra booleanarra](http://en.wikipedia.org/wiki/Boolean_algebra) [http://en.wikipedia.org/wiki/Boolean_algebra]
- [aljebra ikurrak](http://en.wikipedia.org/wiki/List_of_logic_symbols) [http://en.wikipedia.org/wiki/List_of_logic_symbols]

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

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

Aljebra operadorea		C
EZ	~	~
EDO	ÿ	/
ETA	ÿ	&

Aljebra C operadorea		
XOR	$\bar{x} \bar{y}$	\wedge
NOR	\bar{y}	
NAND	\bar{y}	
Eskuineko SHIFT		$x << m$
Ezkerrera SHIFT sinatua		$x >> m$
Ezkerrera SHIFT sinatu gabe		$x >>> m$

Egia Taulak

X	Y	$z = x \bar{y} y z = x \bar{y} y z = x \bar{y} y$		
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	0

Adierazpen Logikoa

- $Z = \neg x \ y + x \ \neg y$

ŷ Egia-taula garatzen badugu XOR eragilearekin duen baliokidetasuna egiaztatzen dugu

4.6. Biderketa

- Biderketa 0xFF x 0x6

ŷ Egin ezazu Binary-n

ŷ Kontuan izan 2ko potentziarik biderkatzean biderkaduraren desplazamendu bat dagoela eskuinera

ŷ biderketa = gehitu eta aldatu

4.7. Programazioa

4.7.1. funtzio matematikoak

- <http://bisqwit.iki.fi/story/howto/bitmath/>

ŷ Iturburu kodea C hizkuntzan idatzita dago

- C estandarraren libm.so liburutegia

4.7.2. Aplikazioa

- Zenbaki osoak biderkatzen dituen programa bat garatu.

4.8. Hardwarea

4.8.1. Zirkuitu digitalak

Oinarriak: Ate logikoa

- [Ate logikoak](http://en.wikipedia.org/wiki/Logic_gate) [http://en.wikipedia.org/wiki/Logic_gate]

ÿ ez, eta, edo, xor

konplexuak

- [batutzaile erdi, batutzaile osoa](http://en.wikipedia.org/wiki/Adder_%28electronics%29#Full_adder) [http://en.wikipedia.org/wiki/Adder_%28electronics%29#Full_adder]

- [biderkatzaila](http://en.wikipedia.org/wiki/Binary_multiplier) [http://en.wikipedia.org/wiki/Binary_multiplier]

ÿ ate logikoz osatutako zirkuitu konbinatiboa

ÿ metagailua eta aldaketa-erregistroa

4.8.2. Unitate Logiko Aritmetikoa (ALU)

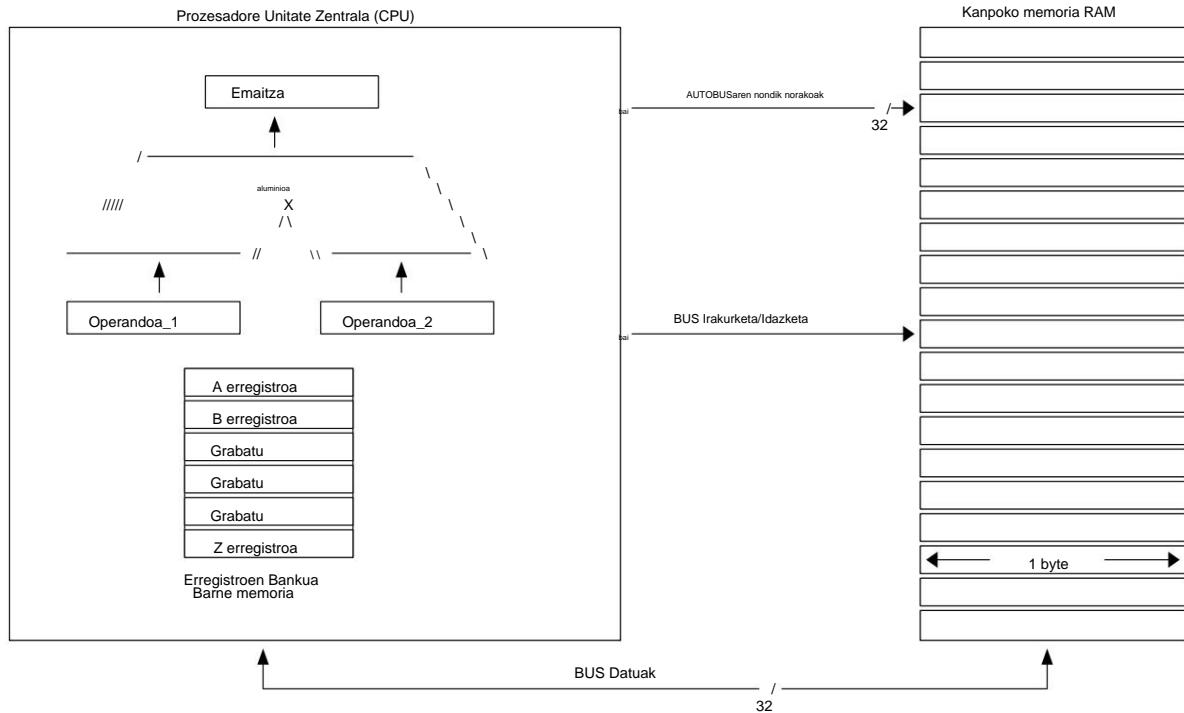
- Unitate logiko aritmetikoa (ALU)

- Zirkuitu Digitala

- CPU-DRAM konexioa

ÿ Argibideen eta Datuen transferentzia

ÿ ALU CPUaren barnekoa da eta helburu orokorreko erregistroetan gordetako zenbaki osoko datuak prozesatzen ditu.



30. Irudia Intel x86 32 biteko arkitektura

4.8.3. EFLAG bandera erregistroa

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

5. taula. RFLAG Erregistroa

pankarta	Bit	Yam
CF	0	eraman-bandera
AF	bi 4	parekidetasun-bandera Egokitu bandera
ZF	6	zero-bandera
SF	7	seinale-bandera
OF	hamaita	Gainetik bandera

- Eraman bandera CF:
ÿ garaiatzeak hitzaren zabalera baino bit posizioan eragiten badu (hitza size) ALUaren **sinatu gabeko edo sinatutako osoko** eragiketa aritmetiko batean

- Gainetik bandera:
ÿ aktibatzen da MSB pisu handiena duen bit-a kontuan hartuz gero (nahiz eta hitzaren tamaina kanpoan egon) adierazi eragiketa aritmetikoan errorea **zenbaki osoekin**. Ez baduzu sartu zenbatu MSB hitzaren tamainatik kanpo, eragiketa zuzena da.

- Parekidetasunaren bandera:

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

- Seinale-bandera:

ÿ aktibatzen da azken eragiketaren emaitza negatiboa izan bada.

- Doitu bandera:

ÿ Suteak LSBn azken eragiketaren emaitzaren zirriborroa egiten bada

- Adibideak

11111111

+ 00000001

100000000 -> Badago garraioa eta BAI gainezkatzea dago? -> Batuketa hau EZ da zuzena, izan ere 9 bitekin eragiketa egiteko eragigaien 9 biteko izan behar dute eta, beraz, zeinu-bitaz luzatu behar da. 9 bit dituen batura hau izango litzateke:

111111111

+ 000000001

000000000 -> Eramatea dago eta ez dago gainezkari. CF=1 eta OF=0.

Ez da inoiz gainezka egingo kontrako zeinuko datuak gehitzen baditugu

A=11110000

B=00010100

AB=A+(-B)

A: 11110000

-B : +11101100

A— 11011100 -> Eramatea dago baina gainezka EZ. Bi datu negatiboren batura

B: zenbaki negatiboa ere ematen du CF=1 eta OF=0

A=10000000

B=10000000

A+B

8 bit beharrean 9 zifrekin batuketak egiteko, bi eragigaiak luzatzen ditut 9 zifrak osatu arte

Norainoko: 110000000

B: +110000000

A+B: 100000000

Kontuan hartzen dugu ez dagoela gainezkari 9 zifra erabiltzen baditugu. Baino ALU 8 biteko erregistroekin funtzionatzen badu, gainezka dago. Bi gehigarriak negatiboak dira (7. zeinu-bit posizioa) eta emaitzaren zeinu-bit (7. bit-posizioa) positiboa da, beraz, emaitza okerra da. 7. posizioan ere carry -> CF=OF=1 dago

- Hurrengo kapituluan berriro ikusia [Programazioa muntatzaile hizkuntzan \(x86\)](#)

4.8.4. Float Point Unitatea-FPU

- Kopara mugikorreko datuak prozesatzeko unitatea
- Antzina CPU ez zen unitate bat zen matematika-koprozesadore izenekoa
- Erabilera Orokoretako Erregistroetatik desberdinak diren SSE izeneko erregistro espezifikoak erabiltzen ditu ALUk erabiltzen du zenbaki osoekin eragiketak egiteko.

5. kapitulua. Argibideen irudikapena

5.1. Ikastaroa

1. Argibideen irudikapena
 - a. Makina-lengoiaia, mutua-lengoiaia eta goi-mailako lengoaiak
 - b. instrukzio formatua
 - c. Argibide motak eta helbideratze moduak

5.1.1. Bibliografia

- W. Stalling testuliburuan aipatzen den gaia
 - ÿ 12. kapitulua: Argibide multzoak: Ezaugarriak eta funtziak (datuak, operandoak eta eragiketak)
 - ÿ 13. kapitulua: Instrukzio-multzoak: instrukzio-formatuak eta instrukzio-moduak
Helbideratzea (Hizkuntza mihiatzatzailea)
 - ÿ B eranskina: Mihiztadura Lengoiaia eta Tresna-katea

5.2. Helburuak

- Oro har, ISA arkitekturen makina-argibideen erreperitorioaren arkitektura (argibide-formatua, datu-formatua, eragiketak eta eragigaien helbideratzea) aztertzea.

5.2.1. Baldintzak

- Baldintzak:
 - ÿ Von Neumann Architecture: Architecture of a Computer, IAS Machine.
 - ÿ Programazioa IAS mihiztadura lengoaien
 - ÿ Datuen irudikapena
 - ÿ Eragiketa aritmetikoak eta logikoak

5.3. Sarrera

5.3.1. Makina-lengoiaia eta formatu bitarra

Java, Python, C, etab. maila altuko lengoaiak garatu ziren, algoritmoak, datu-egiturak eta abar programatzeko zeregina errazteko, programatzaleek erabiltzeko erraza den lengoia bat erabiliz. Bestalde, ordenagailuen CPUek maneiatzten dituzten datuak eta argibideak beste hizkuntza batean daude, MACHINE lengoiaia, prozesadore motaren araberakoa (intel, AMD, RISC).

V, etab...) ordenagailutik. Gure ordenagailuko intel prozesadore baten makina-lengoaia desberdina da smartphone bateko beso-prozesadorearen MAKINA-lengoaia.

Datuak bezala, instrukzioak ere formatu BITARRA batean kodetu behar dira.

Datu bitarrez eta instrukzioz osatutako makina-lengoaiazko programak memoria nagusiko RAM RAMera kargatzeko eta CPUak prozesatzeko prest daude.

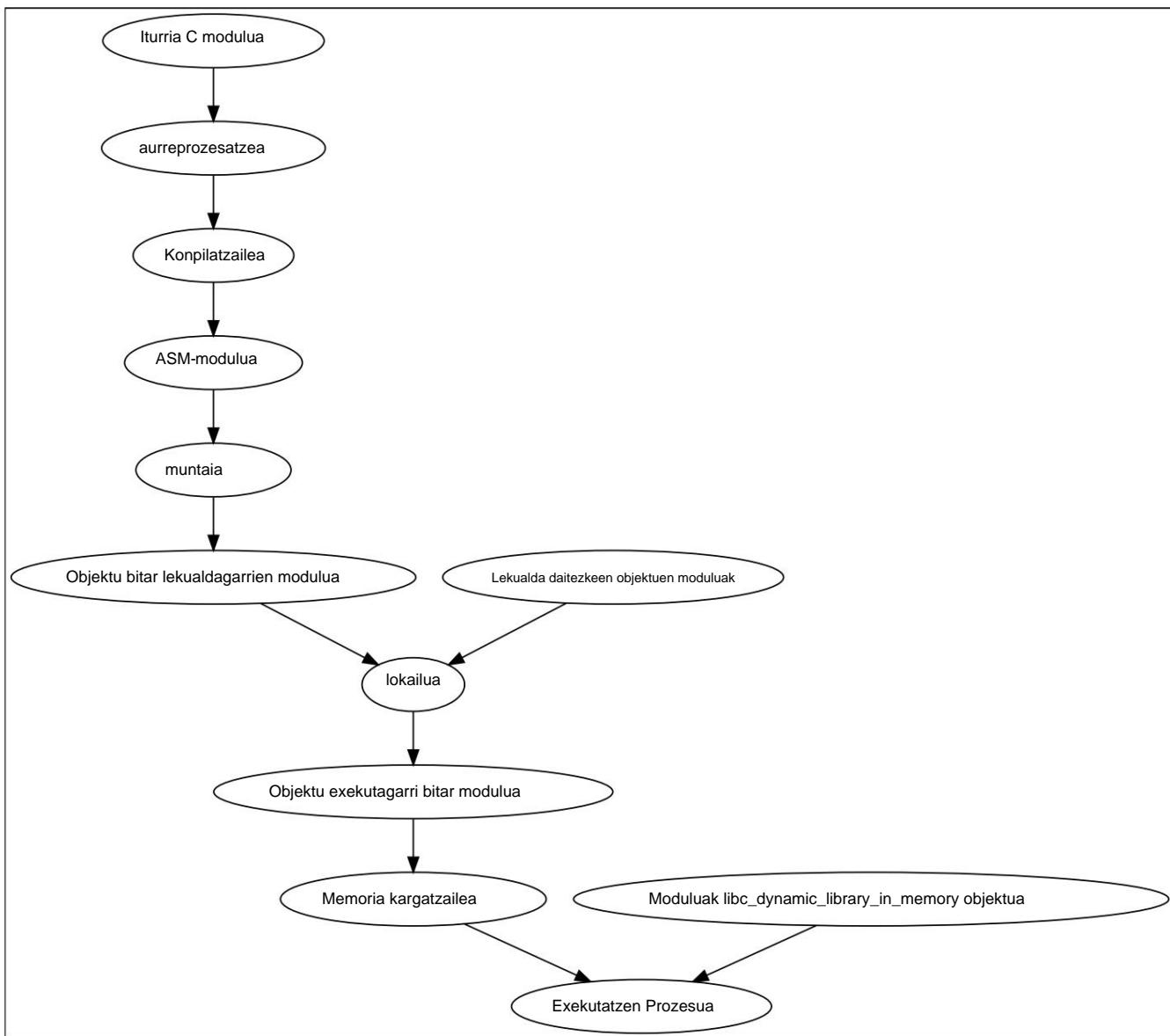
5.3.2. makina-lengoaia eta mutua-lengoaia

Argibideen irudikapena

- Gai honek argibideen irudikapena eta interpretazioa lantzen ditu.
- Argibideak bi hizkuntzatan irudikatu daitezke
 - ÿ Makina-lengoaia formatu bitarrean: 010101010111111000011111
 - ÿ Hizkera bitarrak *irakaskuntza formatua* dakar.
 - ÿ Lengoaia sinbolikoa edo mihitzadura testu formatuan: *amaiera: GEHITU 0x33, emaitza*
 - ÿ Mihitzadura-hizkuntzak *sintaxia* dakar
 - Aginduak hizkuntza bitarrean irudikatzeak memoria nagusian gordetzeko aukera ematen du, baita instrukzio-zikloa erraztu ere, CPUak deskodetu eta exekutatzeko.
 - Aginduak lengoaia sinbolikoan irudikatzeak programatzaileari jarraibideak interpretatzeko eta programak mutua-lengoaian garatzeko duen zeregina erraztu nahi du.

5.3.3. Goi Mailako Hizkuntzak

- 1960ko hamarkadan makina-lengoaia eta mutua-lengoaian programatu bazen ere, hau oso ez-produktiboa zen, esfortzu eta denbora asko eskatzen baitzituen.
- Konponbidea ordenagailuaren funtzioak abstrakzioa egitea izan zen goi-mailako lengoaiak erabiliz:
 - ÿ <https://www.tiobe.com/tiobe-index/>
- Programatzaileentzako goi-mailako hizkuntza eraginkorrik erabiltzeko itzultzale bat behar da ordenagailuaren makina hizkuntzara.



31. Irudia Konpilazio-prozesua

5.4. Arkitektura

5.4.1. Testuingurua

- Ordenagailu batean arkitektura terminoaren esanahia testuinguruaren araberakoa da.
 - ÿ Ordenagailu baten arkitektura: bloke handietan dagoen antolaketa da CPU, Memoria, kontrolagailu periferikoak, busak, etab.
 - ÿ Prozesadore baten arkitektura: CPUa interpretatzeko eta exekutatzeko gai den instrukzio eta datuen erreperitorioa da. Barne egitura ezberdinarekin fabrikatutako bi prozesadore egon daitezke baina instrukzio eta datu berdinak prozesatzen dituztenak, hau da, makina-lengoaia bera prozesatzen dutenak eta, beraz, arkitektura bera dutenak. 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 Prozesadore baten mikroarkitektura: prozesadore baten barne antolaketa da.

5.4.2. Argibide multzoen arkitektura (ISA)

- A-ren instrukzio-multzoen arkitektura (**ISA**: instruction set architecture).

Ordenagailuak, batez ere, honako definizioak ditu:

ÿ Ordenagailuaren egitura: CPU-Memoria-Bus-I/O

ÿ **Datuaren irudikapena eta formatua :**

ÿ **Argibideen irudikapena eta formatua**

ÿ Instrukzio multzoa: izan beharreko eragiketak eta helbideratze moduak ordenagailua interpretatu eta exekutatu.

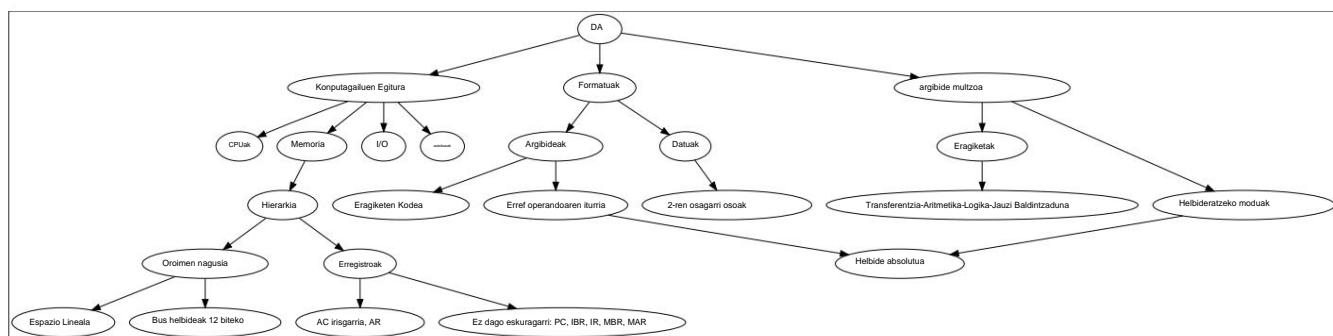
- ISA arkitekturak konputagailuaren CPUaren potentziala definitzen du.

- ISA arkitekturaren diseinuak ordenagailuaren errendimenduan eragingo du.

ÿ Iturburu-programaren konpilazioaren ondoriozko programa bitarra.

ÿ Memoriaren Okupazioa

ÿ Implementazioa (zaitasuna) eta CPUaren errendimendua.



Adibideak: Intel x86, Motorola 68000, MIPS, ARM

- Ikus [Eranskina Mihiztatzaileak](#)

5.5. x86 arkitektura duten Intel prozesadoreak

5.5.1. Nomenklatura

Orokorra

- Intel prozesadoreei izen ezagunak ematen zaizkie: Pentium II, Pentium III, Corei3, Corei5, Corei7, etab.
- Prozesadore hauek erabiltzen dituzten ordenagailuen arkitekturak a erantzuten du arkitektura komuna
 - ÿ x86 arkitektura 32 biteko arkitekturaren kasuan
 - ÿ x86-64 arkitektura 64 biteko arkitekturaren kasuan.
- 32 biteko x86 arkitekturako prozesadoreak

*** 1978 eta 1979 Intel 8086 eta 8088. Lehenengo arkitekturako mikroprozesadoreak

x86.

1980 Intel 8087. x86 arkitekturako lehen zenbakizko koprozesadorea, x87 seriearen hasiera.

1980ko NEC V20 eta V30. 8088 eta 8086 prozesadoreen klonak, hurrenez hurren, NECek fabrikatuak.

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 sartzen du zirkuitu integratuan bertan.

1993 Intel Pentium. Errendimendu hobea, arkitektura supereskalarra.

1995 Pentium Pro. Exekuzioa ordenaz kanpo eta exekuzio espekulatiboa 1996 Amd k5. Intel Pentium-en arerio zuzena.

1997 Intel Pentium II. 16 biteko kodean abiadura hobetzen du, MMX 1998 AMD K6-2 barne hartzen du. Intel Pentium II-ren lehiakide zuzena, 3DNow!

1999 Intel Pentium III. SSE 2000 argibideen sarrera Intel Pentium 4.

NetBurst. SSE argibide hobetuak 2005 Intel Pentium D. EM64T. BitNX, Intel Viiv

- 64 biteko x86-64 arkitekturako prozesadoreak

*** 2003 AMD Opteron. 64 biteko x86 lehen mikroprozesadorea, AMD64 argibide multzoarekin)

2003 AMD Athlon.

2006 Intel Core 2. Intel P8 mikroarkitekturaren sarrera. Kontsumo txikiagoa, nukleo anitz, SSSE3 hardware birtualizazio euskarria. x86-64 barne eta

2008 Core i7

2009 Core i5

2010 Core i3

- [Intel core serie prozesadoreen izenen kodearen esanahia](https://computerhoy.com/noticias/hardware/que-meaning-numbers-letters-processors-intel-56812) [<https://computerhoy.com/noticias/hardware/que-meaning-numbers-letters-processors-intel-56812>]: kodearen lehen zifrak belaunaldia adierazten du (8. belaunaldia 2017an atera zen)

- Intel Core seriea:

ÿ <https://www.intel.com/content/www/us/en/products/processors/core/view-all.html>

- Intel Core X seriea: i9, i7, i5 familiak

ÿ [Intel X seriea](https://www.intel.com/content/www/us/en/products/processors/core/x-series.html) [<https://www.intel.com/content/www/us/en/products/processors/core/x-series.html>]:

ÿ <https://ark.intel.com/products/series/123588/Intel-Core-X-series-Processors>

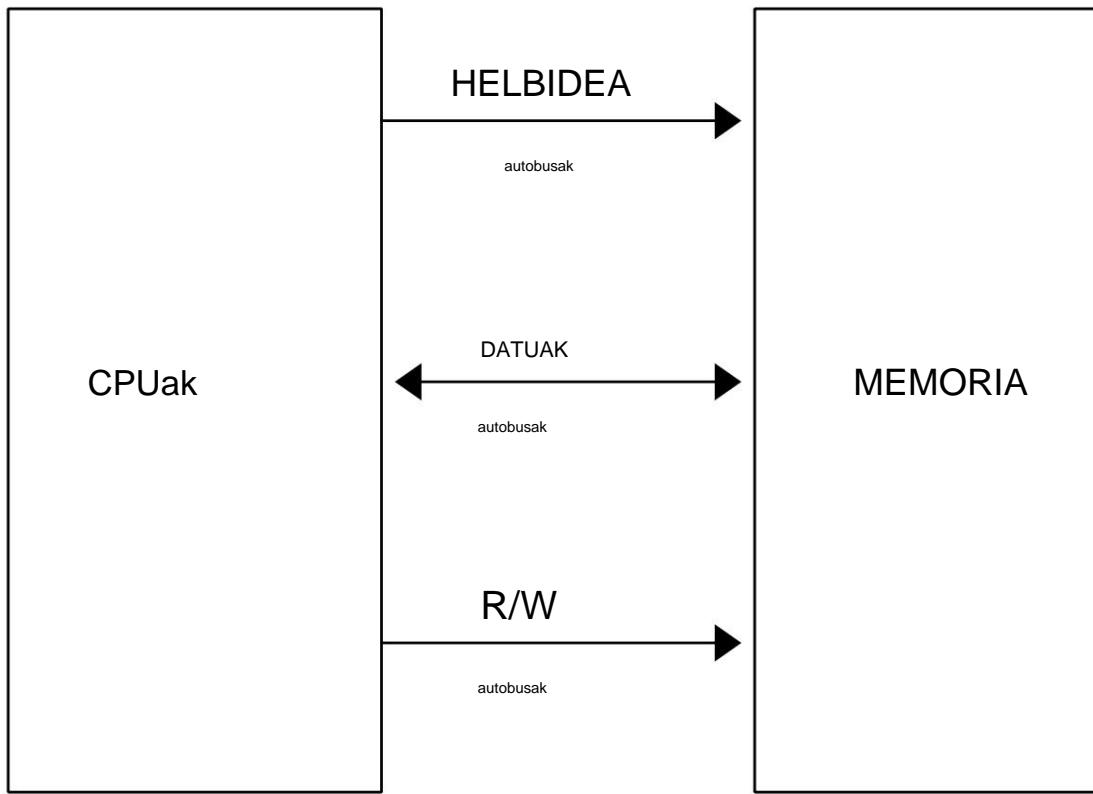
- https://es.wikipedia.org/wiki/Anexo:Procesadores_AMD

- INTEL-AMD LEHIAKETA urtea 2018 mahaigaineko ordenagailuetan.

ÿ [AMD Ryzen 2. belaunaldia - INTEL Core i7-8086K](https://www.techradar.com/news/intel-coffee-aintzira-argitalpen-data-albisteak-eta-zurrumurruak) [<https://www.techradar.com/news/intel-coffee-aintzira-argitalpen-data-albisteak-eta-zurrumurruak>]

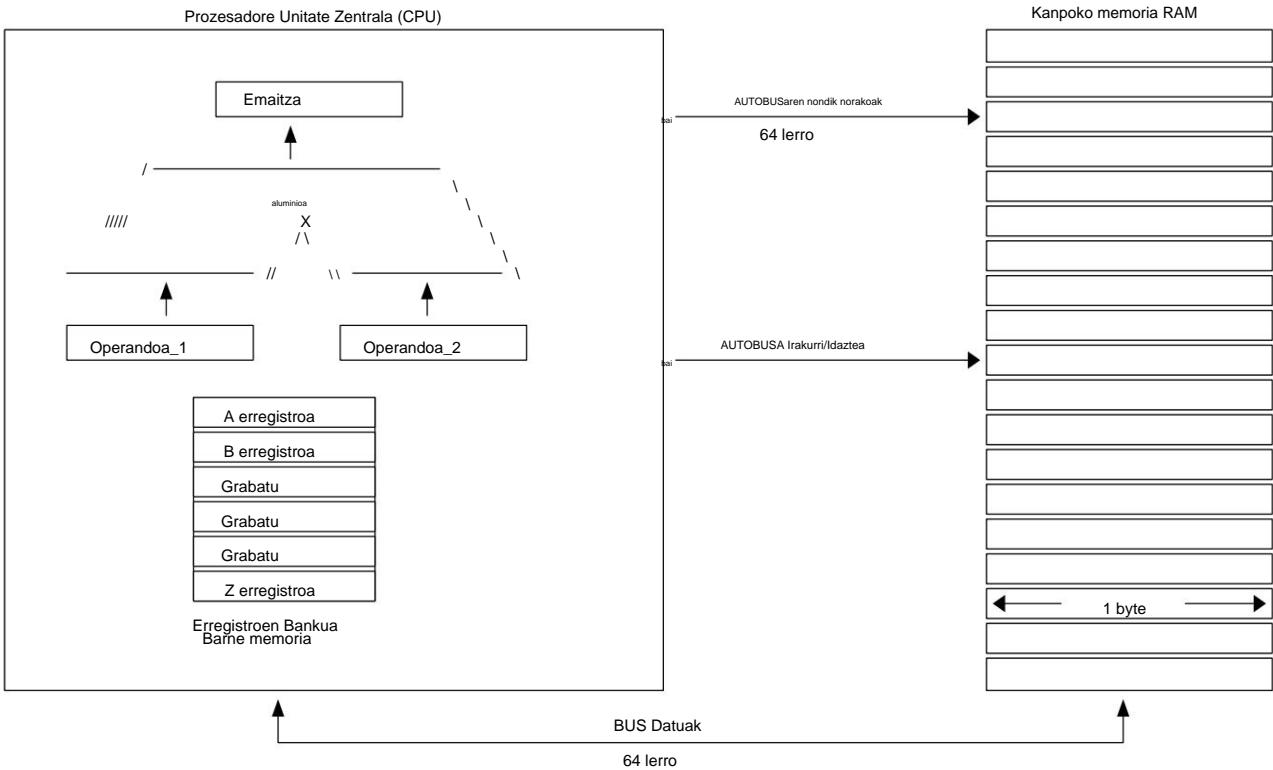
5.6. Konputagailuen Egitura

5.6.1. Arkitektura



5.6.2. CPUak

- CPUaren oinarrizko osagaiaiak
 - ÿ ALU
 - ÿ FPUak
 - ÿ Kontrol Unitatea
 - ÿ Barne erregistroak
- CPU funtzioa
 - ÿ Memorian gordetako jarraibideen instrukzio-zikloa egitea
 - nagusi



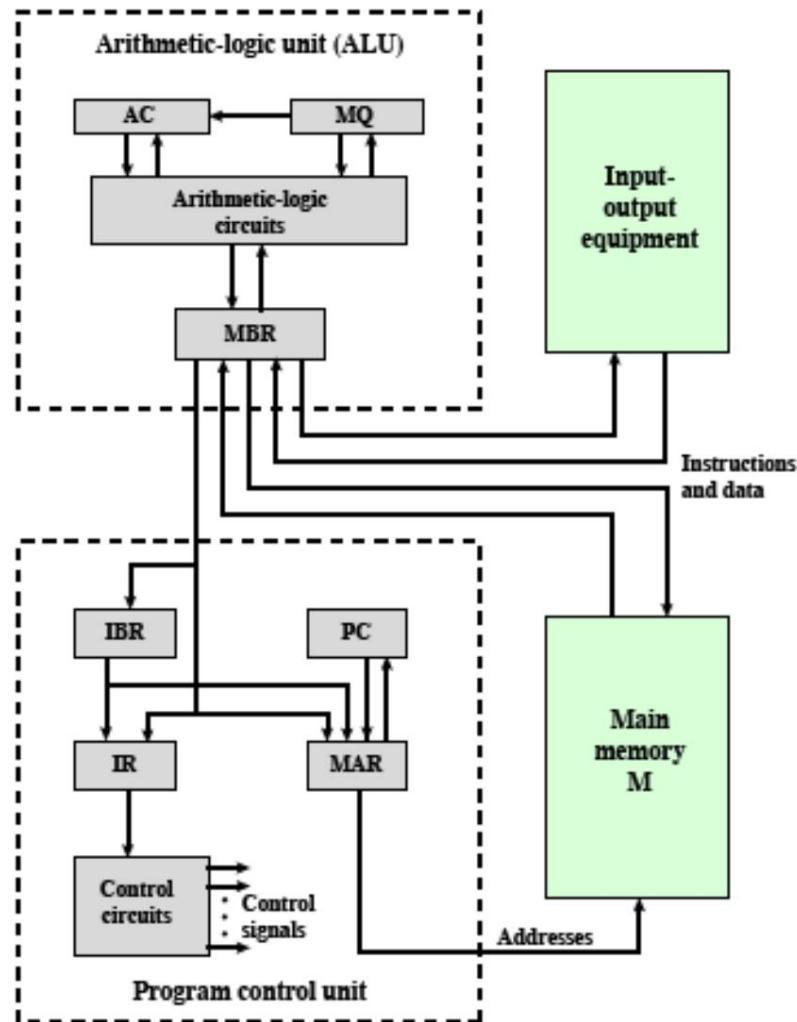


Figure 2.3 Expanded Structure of IAS Computer

32. Irudia IAS Egitura

5.6.3. Memoria

- Memoriaren hierarkia: CPUaren barneko erregistroak eta CPUaren kanpoko Memoria Nagusia (DRAM) erregistratzen ditu. CPUak

Oroimen nagusia

- DRAM Memoria: Ausazko Sarbide Dinamikoaren Memoria
- Makinaren jarraibide bitarren sekuentzia eta datuak formatu bitarrean gordetzen ditu.
- Helbide-espazio lineala: notazio hamaseitarra
- Helbidea: byteak: idazkera hamaseitarra

HELBIDEAK

0x00000000

0x00000001

0x00000002

0x00000009

0x0000000a

0x0000000f

EDUKIAK

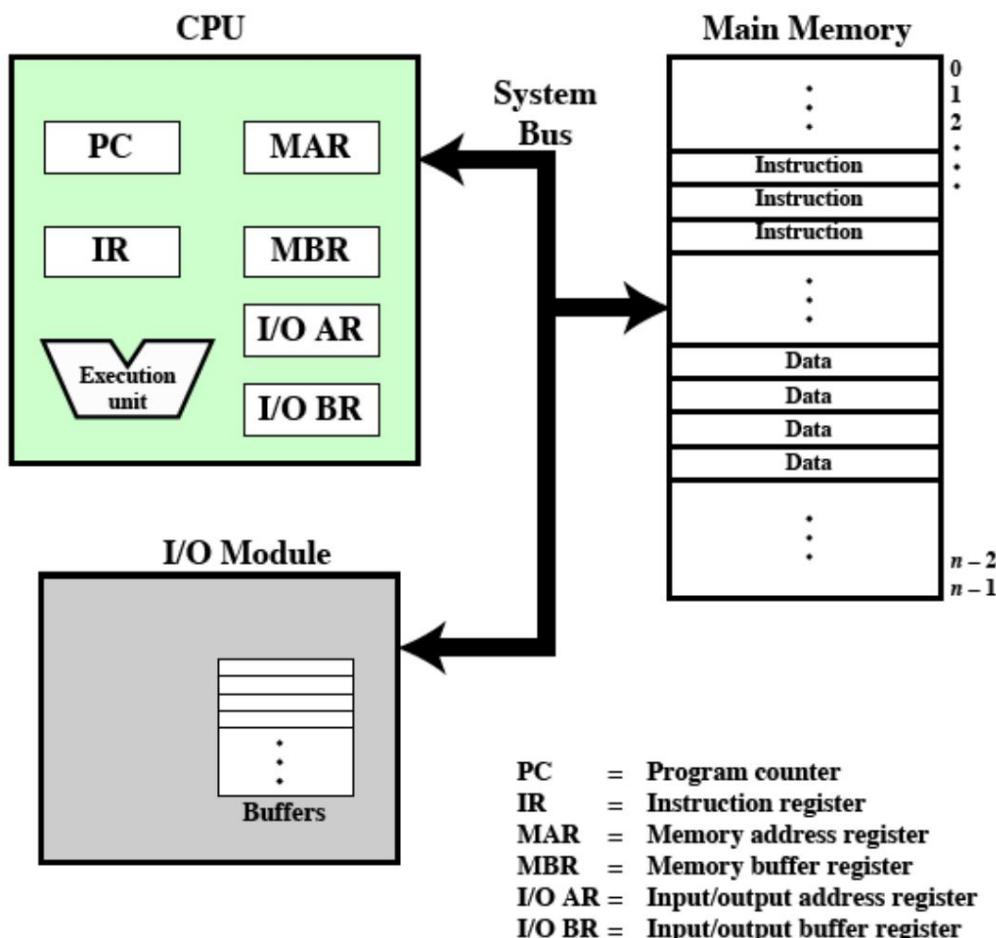
1 0 1 0 1 0 1 0
1 0 1 0 1 0 1 0
1 0 1 0 1 0 1 0
1 1 1 1 1 1 1 1

PUZaren barneko erregistroak**Erregistroak EZ dira ikusgai programatzaileak**

- Programatzaileak amd64 arkitekturan sar daitezkeen 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 arkitekturaren kasuan, ordezkatu 64 bit 32 bit eta MAR erregistroa ere bada. 32 bit.

Erregistroak programatzaleak ikusgai

- Programatzaleak erregistro hauek erabiltzen ditu datuak gordetzeko (idatzi eta irakurri).
- Badaude eragiketak (baketa, kenketa, etab...) non eragigaiak erregistroetan egon daitezkeen.
- CPU erregistrorako sarbidea askoz AZKARAGOA da posizio baterako sarbidea baino RAM memoria nagusia.



33. Irudia IAS_Arkitektura

5.7. Makina baten instrukzio-elementuak

- Eragiketa-kodea:

ÿ Instrukzioak PUZak zer eragiketa egin behar duen zehaztu behar du. Batuketa eta kenketa aritmetika bezalako eragiketak, ez eta eta bezalako ~~datuak keta logikoak, eragiketa logikoak, eragiketa nagusira datuak transferitzeak, esate baterako, disko gogoretik memoria nagusira datuak transferitzea, etab.~~

- Iturburuko operandoaren erreferentzia:

ÿ Eragiketa batek datu bat edo gehiago prozesatzea eska dezake. Adibidez, eragiketa logikoak ez du eragiketarik behar.

- Helburuko operandoaren erreferentzia:

ÿ Gehitze-eragiketa batek bi eragiketa behar ditu, bat iturburuko eragiketa eta bestea iturburuko eragiketa.
helmuga operatiboa.

- Emaitzia Erreferentzia:

ÿ Gehitze-eragiketa batek eragiketaren emaitza gordetzea eskatzen du.

- Hurrengo argibideen erreferentzia:

ÿ Instrukzioaren exekuzioa amaitutakoan, CPUari non dagoen adierazi behar zaio.

PC Program Counter bidez exekutatuko den hurrengo instrukzioa gorde du.

5.7.1. Helbide Implizituak

- Instrukzioan esplizituki agertzen ez diren helbideak. Adibideak:

ÿ Hurrengo instrukzioa Program Counter-en

ÿ Metagailuan emaitza

ÿ etab.

5.7.2. Operando arkitektura motak: adibideak

- 3 Mota

ÿ Metagailura zuzendutako arkitektura: Eragiketa bat implizituki pilatzailean dago

ÿ Pila Bideratutako Arkitektura ([Pila Eranskina](#)):

ÿ Eragileak PUZaren barneko pilatik bultzatzen edo ateratzen dira

ÿ Pila kontzeptua: push/pop ÿ push/pull ÿ lehen sartu azken atera
irten ÿ Lehen sarrera Azken irteera

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

ÿ Erregistroetara zuzendutako arkitektura:

ÿ Bi mota: Reg/Mem eta Load/Store, amd64 eta arm arkitekturarekin gertatzen den bezala.
hurrenez hurren.

ÿ Reg/Mem : instrukzioa exekutatu ahal izateko, bi eragigaietako bat egon behar da
erregistro bat

ÿ Load/Store: Bi eragigaiiek bi erregistrotan egon behar dute, horrela esanda
instrukzioa exekutatzen da

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

Pila	Metagailuen Erregistroa/Memoria Karga/Bildetzea		
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.

ÿRTL 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 pilara kargatu behar dira lehenik

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

ÿ **Gehitu B** ÿ ez dago HELMUGA eragigaiaren erreferentziarik

ÿ Aurretik metagailuan kargatu behar den xede-operandoa.

ÿ Reg/Mem: R1ÿM[A];R3ÿR1+M[B];M[C]ÿR3

ÿ **Gehitu R3,R1,B** ÿ EZ DA posiblea MEMORIAn eragiketa bat baino gehiago aipatu

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.

ÿ **Gehitu R3,R1,R2** ÿ Erregistroei erreferentziak bakarrik egiten zaizkie, ez dago erreferentziarik memoria

ÿ Jatorrizko eta helmugako eragigaiak aldez aurretik kargatu behar dira erregistroetan

ÿ

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

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

<u>Instruction</u>	<u>Comment</u>
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

<u>Instruction</u>	<u>Comment</u>
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	$Y \leftarrow Y \div T$

<u>Instruction</u>	<u>Comment</u>
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:

ÿ M[SP]ÿM[C];M[SP]ÿM[E];M[SP]ÿM[D];MUL;GEHITU;M[SP]ÿM[B];M[SP]ÿ M[A];SUB;DIV
ÿbulkatu C; bultza E; bultza D; mul; gehitu; bultza B; bultza A; azpia; div;

5.8. Ordenagailuetarako, oro har, muntatzaile-lengoain (ASM) jarraibideen irudikapena

5.8.1. Makina Lengoaia Bitarra

adibideak

- [objdump -d modulu_bitarra](#)
- Ikus [Eranskina Mihiztatzaileak](#)

Memoria Biltegiratzea

- Sorburu-moduluaren itzulpen-prozesua mihiztatzaile-lengoain egin ondoren, objektu-modulu bat sortzen da hizkuntza bitarrean, eta disko gogorrean fitxategi moduan gordetzen dena.
- Objektu exekutagarriaren modulua 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 **4001a4: 48 83 ec 10**
 - ÿ **0x4001A4** posizioan **48** bytea dago
 - ÿ **0x4001A4+1** posizioan **83** byte da
 - ÿ **0x4001A4+2** posizioan **EC** bytea dago
 - ÿ **0x4001A4+3** posizioan **10** byte da
 - ÿ **0x4001A4** memoria -kokapen nagusian, 4 byteko instrukzioa gordetzen da.

5.9. Eragileak: Helbideratzeko moduak

5.9.1. Kokapena

- Eragigaien kokapen posibleak.
 - ÿ Adierazpenean bertan
 - ÿ Barne memoria: CPU erregistroak
 - ÿ Memoria Nagusia: DRAM memoria
 - ÿ Memoria i/o: portu izeneko sarrera/irteera kontrolagailuetan erregistratzen da.

5.9.2. Instrukzio-zikloan erreferentziatutako helbideak

- Instrukzio-zikloan zehar 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, eragigai anitzekin.
 - ÿ Arkitekturaren araberakoa da: Metagailua (Adib.: IAS makina), Erregistro-Memoria (Adib.: IAS makina x86), Kargatu/Gorde (Adib.: ARM), Pila (Adib.: JVM makina), Memoria-Memoria
 - ÿ Eragigaiaren erreferentzia implizituak

5.9.3. Hizkuntza orokor baterako helbidea

Argibide-formatua: Eremuak

Op kodea	Mod. Helbideratzea	A
----------	--------------------	---

- Instrukzio-formatuaren egitura baten **adibide berezia** a-n hiru eremutan ISA arkitektura.
 - ÿ Eragiketa kodea: mugitu, kargatu, batu, kendu, etab.
 - ÿ A kodea: eragigaiaren eremua: eragigaiaren kokapenari egiten dio erreferentzia
 - ÿ Kode Mod. Helbidea: A eremua interpretatzeko modua adierazten du
- EA: Helbide eraginkorra : Eragigaia kokatzen den helbide eraginkorra.
- Op: Funtzionamenduan EA helbide eraginkorrean jasotako datuak dira.
- Datu *operatiboak* honako hauetan gorde daitezke:
 1. RAM kanpoko memoria
 - a. Datuak dituen memoria helbidea.
 - b. Instrukzio bat duen memoria helbidea. Datuak arloetako bat da irakaskuntzaren jabe izateko. Berehalako helbidea.
 2. GPR barne memoria
 - a. Rax, eax,... erregistratzen ditu

Helbide Motak

- Eragaiaren kokapenaren EA erreferentzia- *helbide* eraginkorra ren arabera lortzen da helbideratze modu desberdinak.

- Helbideratzeko modua MD eremuan kodetuta dago

- Berehala:

ÿ Eragaia instrukzioaren beraren eremutik lortzen da.

ÿ EA= ez da existitzen

ÿ Op=A

- Zuzena:

ÿ Eragaia kanpoko memorian dago. Eragaien eremuak helbidea dauka
eraginkorra

ÿ AE=A

ÿ Op=M[EA]

- Grabatu:

ÿ Eragaia barne memorian dago. Eragaien eremuak erreferentzia dauka
Grabatu.

ÿ AE=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 Eragaiaren helbide eraginkorra oinarrizko helbide baten eta oinarrizko helbidearekiko desplazamendu baten arteko eragiketa aritmetikoa eginez lortzen da. Oinarrizko helbidea erreferentzia gisa hartzen da eta desplazamendua oinarri-helbidearekiko erlatiboa da.

a. Ordenagailuko programa-kontagailuari dagokionez:

ÿ Oinarrizko helbidea implizituki programaren kontagailua eta desplazamendua da
funtzionamendu-eremuan dago.

ÿ EA=CP+A

ÿ Op=M[EA]

b. Oinarriari dagokionez:

ÿ Desplazamendua eragaien eremuan dago eta oinarrizko helbidea eragaien eremuan.
Izen ematea.

ÿ EA=R+A

ÿ Op=M[EA]

c. Indexatua:

ÿ Desplazamendua erregistroan dago eta oinarrizko helbidea ren eremuan dago funtzionatzen.

ÿ EA=A+R

ÿ Op=M[EA]

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

5.10. Eragiketak

5.10.1. Eragiketa Kodeak

- Eragiketa multzoaren kodeketa ISA arkitektura bakoitzaren araberakoa da.

5.10.2. Eragiketa motak

- Kategoriak:

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

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
• Errepetorioa hau izan daiteke: murriztua/zabala, konplexua/simplea.		

6. Kapitulua. Programazioa Muntatzaile Lengoaian (x86): Goi-mailako lengoaien oinarrizko eraikuntzak.

6.1. Ikastaroa

1. Programazioa Mihiztatzaile lengoaian x86

- a. x86 :Datuen irudikapena, argibideen irudikapena, helbideratze moduak.
 - b. Esleipenen adierazpenak
 - c. baldintzazko adierazpenak
 - d. begiztak
- eta. Deiak eta funtzioaren edo azpierrutinaren itzulera

6.2. Sarrera

6.2.1. Helburuak

- x86-64 arkitekturako makinen instrukzioen erreperitorioaren arkitektura (argibide-formatua, datu-formatua, eragiketak eta eragiketen helbideratzea) aztertzea, GNU_asm_x86 (gasa) mihiztadura-lengoaiaren programen garapen praktikoan erabiltzeko.
- Programa txikiak x86 mihiztatzaile lengoaian garatzeko gaitasuna, funtzio bat izanik laborategiko praktiken ezarpena.

6.2.2. Baldintzak

- Baldintzak:
 - ÿ Von Neumann Architecture: Architecture of a Computer, IAS Machine.
 - ÿ Programazioa IAS mihiztadura lengoaian
 - ÿ Datuen irudikapena
 - ÿ Eragiketa aritmetikoak eta logikoak
 - ÿ Instrukzioen ordezkaritza

6.3. Konputagailuen Egitura

6.3.1. CPUak

- CPUaren oinarrizko osagaiak
 - ÿ ALU

- Kontrol Unitatea
- Barne erregistroak
- CPU funtzioa
 - Memorian gordetako jarraibideen instrukzio-zikloa egitea nagusi

6.3.2. Memoria

- Memoriaren hierarkia: CPUaren barneko erregistroak eta CPUaren kanpoko Memoria Nagusia (DRAM) erregistratzen ditu.
- CPUak

6.3.3. Oroimen nagusia

- DRAM Memoria: Ausazko Sarbide Dinamikoaren Memoria
- Makinaren jarraibide bitarren sekuentzia eta datuak formatu bitarrean gordetzen ditu.
- Helbide-espazio lineala: notazio hamaseitarra
- Helbidea: byteak: idazkera hamaseitarra

6.3.4. PUZaren barneko erregistroak

sarrera

- Programatzailaileak amd64 arkitekturan sar daitezkeen erregistroak
 - PC: Programa-kontagailua: x86-k RIP deitzen du: 64-bit
 - IR: Instrukzio Erregistroa: 64 bit
 - MBR: Memoria buffer erregistroa: 64 bit • HITZA TAMAINA: 64
 - MAR: Memoria-helbideen erregistroa: 40 bit
 - Memoria Edukiera: 240 : 1TB
- i386 arkitekturaren kasuan, ordezkatu 64 bit 32 bit eta MAR erregistroa ere bada. 32 bit.

Erregistroak programatzailaileak ikusgai

63-0	31-0	15-0	15-8	7-0
rax	eax	aizkora	oi	du
rbx	ebx	bx	bh	bl
rcx	ecx	c x	ch	cl
rdx	edx	d x	dh	dl
rsi	Hori da	Bai		sil
rdi	editatu	eman		esan
rbp	ebp	bp		gpl

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 ordez E-rekin eta lortuko dugu 32 biteko arkitektura izendapena.

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

- Salbuespenak daude

Kontrol-bandera-erregistroa

- EGOERA erregistroa: Instrukzio baten exekuzioak bandera izeneko bit batzuk aktibatzen ditu egindako eragiketaren ondorioak adierazten dituztenak. Adibidea: gainezkatze bandera: adierazten du egindako eragiketa aritmetikoak emaitzaren gainezkatzea eragin duela esandako operazioa.
- [Wikipedia](http://en.wikipedia.org/wiki/FLAGS_register_(informatika)) [http://en.wikipedia.org/wiki/FLAGS_register_(informatika)]
- OSZAPC banderei bakarrik begiratzen diegu.

6. taula. RFLAG Erregistroa

pankarta	Bit	Yam
CF	0	eraman-bandera
.....	bi	parekidetasun-bandera
AF	4	Egokitu bandera
ZF	6	zero-bandera
SF	7	seinalde-bandera
OF	hamaita	Gainetik bandera

- Eraman bandera:

ÿ sinatu gabeko edo sinatu gabeko zenbaki osoko eragiketa aritmetiko batean ALUren hitzaren tamaina baino bit posizioa eragiten badu eramanak ezartzen du.

- Gaintze-bandera:

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

- Parekidentasunaren bandera:

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

- Seinale-bandera:

ÿ aktibatzen da azken eragiketaren emaitza negatiboa izan bada.

- Doitu bandera:

ÿ Suteak LSBn azken eragiketaren emaitzaren zirriborroa egiten bada

- Adibideak

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

100000000 -> garraiatze bat dago eta gainezkatzea ere bada 8. posizioko MSB=1 kontuan hartzen badugu emaitza okerra izango litzateke, hau da, zeinu luzapena egingo bagenu emaitza okerriko eta beraz zifra barrutitik kanpo baztertu egin behar da eragiketa zuzena izan dadin. Zeinu-bit "hitzaren tamaina"ren posizio altuenean dagoena da (7. posizioa)

Nora: 11110000

-B : +11101100

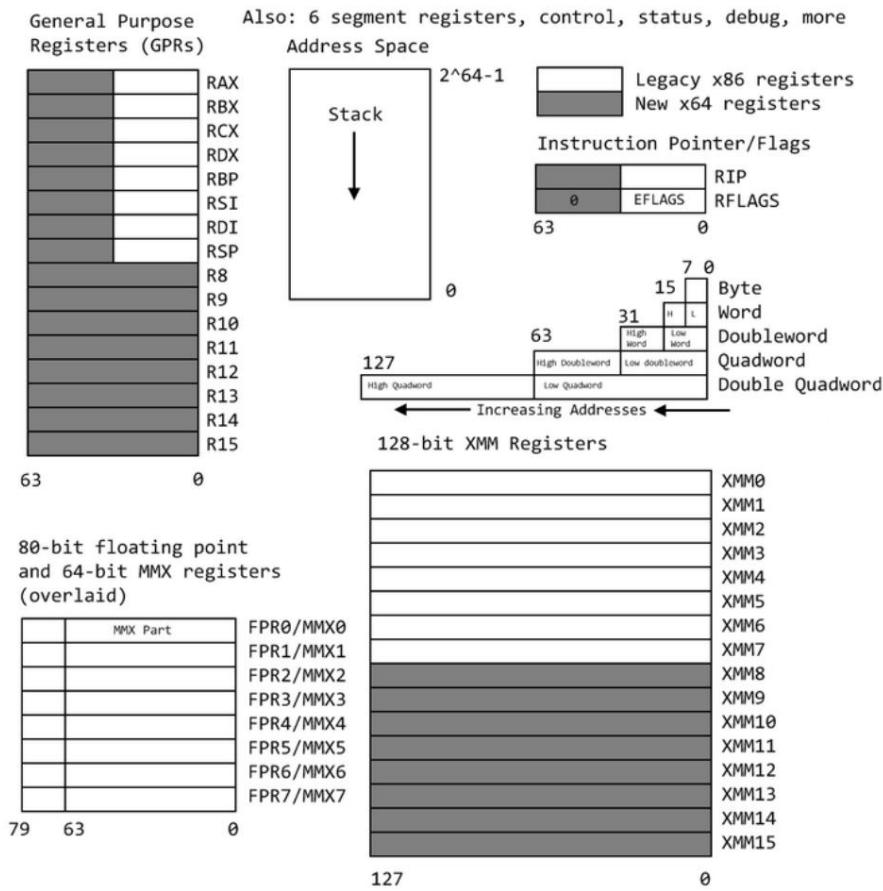
A-B: 111011100 -> garraioa dago baina gainezkarik ez, MSB=1 "hitzaren tamaina"tik kanpo dagoen posizioan ez baitu emaitzaren zeinuan eragiten, izan ere, posizioak MSB (8.) eta aurreko (7.) zeinu-bitaren balio bera dute 1. zifra

A: 10000000

-B: +10000000

A-B: 100000000 -> eraman bat dago. Gainezka ere badago, 1 balioko MSB bit-a 0 balioaren aurreko posizioaren (7., zeinu-bit) desberdina baita. Bi gehigarriak negatiboak dira (seinu-bit-posizioa) eta emaitzaren zeinu-bit (bit-aren 7. posizioa) positiboa da orduan emaitza okerra da.

Beste Erregistroak



- fp, mmx eta xmm erregistroak instrukzio konplexuak exekutatzeko erabiltzen dira, esate baterako, zenbaki errealekin koma mugikorrean funtzionatzen duten tangentea edo osoko datu anitzetan eragiketak egiten dituzten instrukzioak (Instrukzio bakarreko datu anitzak) (adibidez, produktu eskalarra).
- Informazio gehiago [FPU_x87 eranskinean](#)

6.4. Intel Language vs AT&T Language

6.4.1. i386/amd64 arkitektura muntaia-lengoaiak

- AMD64 arkitekturako instrukzio multzoko makina-kode-lengoia da bakarra baina ez aipatutako arkitekturari dagokion mihiztadura.
- "Konputagailuen Egitura" irakasgaian enpresaren **AT&T** sintaxia erabiltzen da. American telefono AT&T.

6.4.2. Argibideen sintaxia INTEL hizkuntzan

- Mihiztadura-lengoiaian jarraibideen formatua argibideen *sintaxia* bezala ezagutzen da . argibideak.
- 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 ;begizta hasiera
 ÿ batuketa: gehitu eax,esi ;sum
 ÿ + mov ax,[emaitza] ;gorde emaitza+

ÿ Mihiztadura, hizkuntzaren sintaxia muntaketa prozesuaren itzultzailearen arabera koa da. Mihiztatzailea (muntatzalea) erabiltzen da, kasu honetan, NASM mihiztatzailea erabiltzen da.

- Intel asanblea hizkuntzan eta "NetWide Asm" mihiztatzaileko programa baten sum1toN.asm [adibidea](#) (nasma).
- Tutoretza osoa [NASM tutorialspoint](#)
- [Bristoleko Unibertsitateko](#) oharrak
- [Paul Carter doktorearen](#) eta Paul Carter doktorearen oharrak

GNU Batzarra (Gas)

- AT&T telefono konpainiak garatutako hizkuntza
- Gas 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 ;begizta hasiera
 ÿ batura: gehigarri %esi,%eax ;gehitu
 ÿ + movw %ax,result ;gorde emaitza+

• ETIKETA

ÿ Lehenengo zutabeen zehaztuta. Atzizkia du :

- ERAGIKETA-KODEA: ikur mnemoteknikoak erabiltzen dira eragiketa intuitiboki interpretatzen laguntzeko. Adib.: GEHITU gehitu, MOV mugitu, SUB kendu, ...
- ERAGIKETA ITURRI ETA/EDO HELMUGA

ÿ datu alfanumerikoa: irudikapen alfanumerikoa ÿ 16

ÿ berehalako helbidea: \$ aurritzka

ÿ kanpoko memoria helbidea: etiketa ÿ emaitza

ÿ zuzeneko helbidea

ÿ CPU barneko erregistroak: %rax,%rbx,%rsp,%esi,.. ÿ %rsp,%esi

ÿ % aurizkiak izenak erregistro bati erreferentzia egiten diola esan nahi du

ÿ Eragileen datuen tamaina: atzizki mnemoteknikoak: q(quad):8 byte, l(luzea):4 byte,
w(hitza):2 byte, b(byte):1 byte.

ÿ Mihiztadura-hizkuntzaren sintaxia muntaketa-prozesuaren itzultzalearen arabera koa da.
erabiltzen den muntala (muntatzarea), Kasu honetan, GAS muntatzarea erabiltzen da.

6.4.3. Muntaketa-prozesuen itzultzaleak

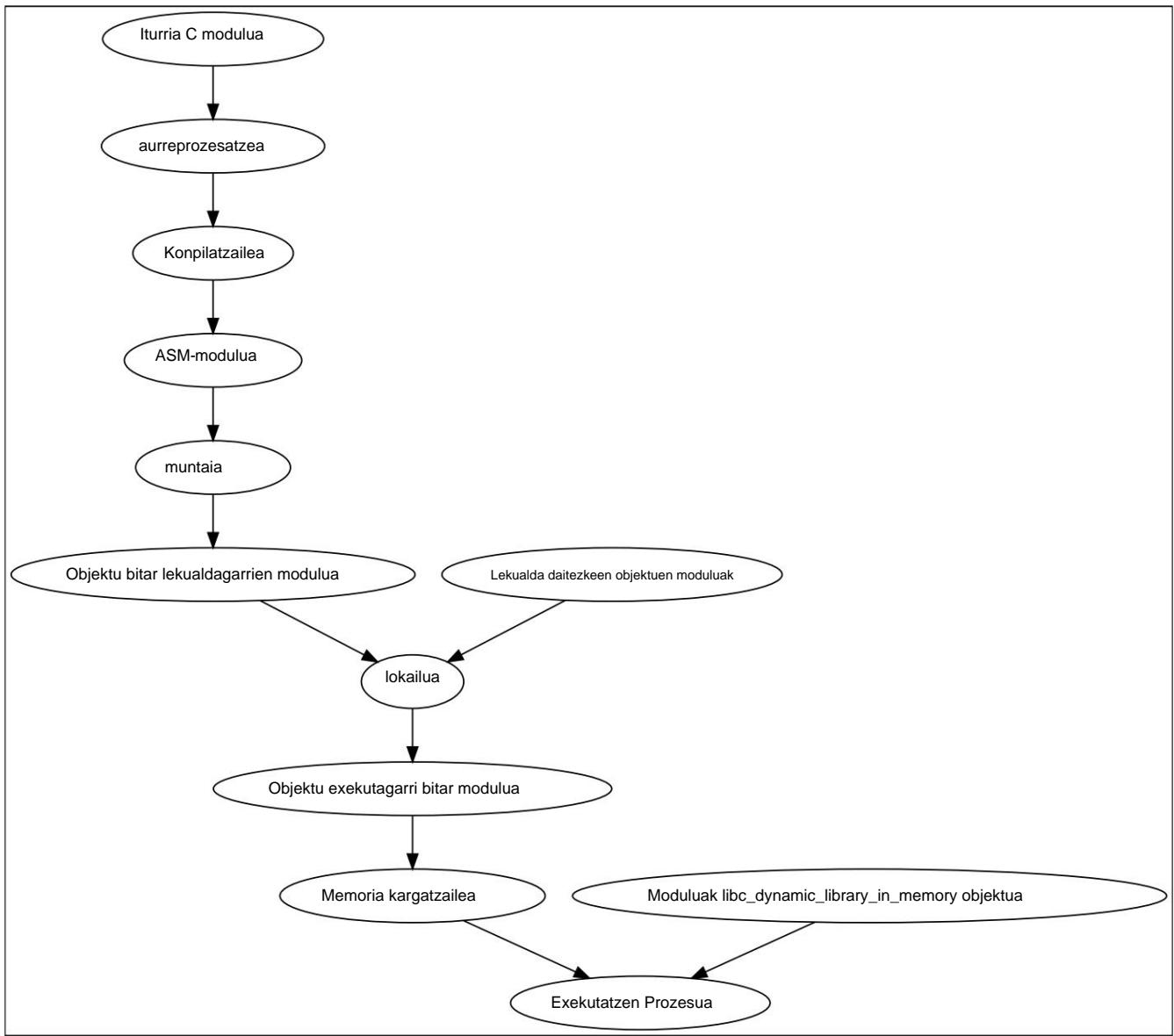
- i386/amd64 ISArako bi mihiztadura-lengoaia daude, bi sintaxi ezberdin, mihiztatzaile-itzultzale ezberdinek itzuli behar dituztenak, makina-lengoaia berean modulu bitar bat ekoizten dutenak.
- "NASM" (Netwide Asm), "FASM", "MASM", "TASM" eta " **YASM** " itzultzale mihiztatzaileek i386/amd64 arkitekturarako i386/amd64 arkitekturarako iturburu-moduluak itzultzen dituzte iturburuko moduluak.
- GNU Foundation "as" itzultzaleak iturburu-moduluak **AT&T** mihiztadura-lengoaiaren sintaxiak erabiliz itzultzen ditu i386/amd64 arkitekturako modulu bitaretara.
- Disko gogorrean fitxategi batean gordetako iturburu-moduluaren irudikapenaren eta makina-lengoaiaren irudikapenaren arteko itzulpena [muntaketa-prozesuan gertatzen da](#):

ÿ Mihiztadura-hizkuntzaren iturburu-moduluak ÿ Mihiztatzaile itzultzaleak ÿ Objektu-modulua

Lekualdagarría ÿ Lokatzailea ÿ Objektu exekutagarri bitar moduluak ÿ Kargatzailea 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 sorrera (hello_world martxan). "as", "ld" eta "loader" dira mihiztadura-
lengoaiaren modulutik prozesu bat sortzeko beharrezkoak diren GNU tresnak.

ÿ **ld** linker -ek objektu-modulu sistema eragilearen inguruneko moduluekin eta liburutegietako moduluekin
nahasten du.



34. Irudia. Konpilazio-prozesua

6.4.4. Makina kodea

Memoria Biltegiratzea

- Sorburu-moduluaren itzulpen-prozesua mihiztatzale-lengoaian egin ondoren, objektu-modulu bat sortzen da hizkuntza bitarrean, eta disko gogorrean fitxategi moduan gordetzen dena.
- Objektu exekutagarriaren modulua hizkuntza bitarrean duen fitxategia kargatu behar da memoria nagusia. Zeregin hau sistema eragilearen **kargatzialeak** egiten du.
- Memoria-helbide bakoitzak byte 1era seinalatzen du.
- Helbide baxuenak objektu osora seinalatzen du: instrukzioa edo datuak.
- Adibidea:

ÿ 4001a4: 48 83 ec 10

ÿ 0x4001A4 posizioan 48 bytea dago

ÿ 0x4001A4+1 posizioan 83 byte da

- ÿ **0x4001A4+2** posizioan **EC** bytea dago
- ÿ **0x4001A4+3** posizioan **10** byte da
- ÿ **0x4001A4** memoria -kokapen nagusian, 4 byteko instrukzioa gordetzen da.
48 83 ec 10.

Makina Kodearen Interpretazioa

- Adibidea:

ÿ amd64 arkitekturako makinen instrukzioa.

ÿ **4001a4: 48 83 ec 10** ÿ **subq \$16,%rsp**

ÿ Programatzailearen interpretazioa:

ÿ AT&T x86 arkitekturaren mihiztadura-lengoaia.

ÿ Argibideen deskribapena **RTL hizkuntzan**: *RSP* ÿ *RSP - 16*

ÿ 0x4001A4 memoria-kokapen nagusian **subq** instrukzioa gordetzen da

\$ 16,% rsp

ÿ subq-k kenketa eragiketa adierazten du 64 biteko osoko datuekin (q atzizkia). kendu
helmuga eragiketa iturburuko eragigaia.

ÿ Sorburuko eragigaiak 16 balio hamartar bat du, 0x10 hamaseimalean eta eragigai honen helbideratzea
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 instrukzioari erreferentzia egiten ez zaio instrukzioari, baizik eta egiten ari den CPUak
eragiketa PCÿPC+aginduen tamaina bytetan.

ÿ Nola interpretatu makina bat instrukzio bat hizkuntza bitarrean? **x86 makinarako ISA Arkitektura Erreferentzia Eskuliburu** kontsultatu eta helbideratze moduak ulertu behar dituzu. Adibide hau Gai honen amaieran garatzen
da 8. atalean) *ISA x86 arkitekturaren instrukzio-formatua*

6.4.5. "Aso" muntatzailea

Zuzentaraauak

- GNU/linux sistemak hizkuntzarako erabiltzen duen "as" itzultaile mihiztatzailearen zuzentaraauak
AT&T muntatzailea.

ÿ GNU mihiztatzailea "gas" bezala ere ezagutzen da.

ÿ [binutilak](https://www.gnu.org/software/binutils/) [<https://www.gnu.org/software/binutils/>] ÿ [mihiztatzaile gisa](https://sourceware.org/binutils/docs/2.26/as/index.html) [<https://sourceware.org/binutils/docs/2.26/as/index.html>]: eskuliburu ofiziala

ÿgas [erreferentzi txartela](http://www.coranac.com/files/gba/re-ejected-gasref.pdf) [<http://www.coranac.com/files/gba/re-ejected-gasref.pdf>]

ÿ .word-ek 2 byte gordetzen ditu amd64-n.

ÿ [Orakulua](https://docs.oracle.com/cd/E26502_01/html/E28388/eoiyg.html) [https://docs.oracle.com/cd/E26502_01/html/E28388/eoiyg.html]

ÿ [x86 eta x86-64 aukerak](https://sourceware.org/binutils/docs-2.26/as/i386_002dDependent.html#) [https://sourceware.org/binutils/docs-2.26/as/i386_002dDependent.html#]

i386_002dMendekoa]

Eskuliburua

- [binutils gisa](https://sourceware.org/binutils/docs/as/) [https://sourceware.org/binutils/docs/as/]: eskuliburu ofiziala
- [Linux Assembly nola egin](http://asm.sourceforge.net/howto/Assembly-HOWTO.html) [http://asm.sourceforge.net/howto/Assembly-HOWTO.html]: erreferentziak muntatzaile desberdinak
- [MIT](http://web.mit.edu/gnu/doc/html/as_1.html) [http://web.mit.edu/gnu/doc/html/as_1.html]

6.5. Datuen irudikapena hizkuntzan Muntatzailea (ASM) i386/amd64 arkitekturarako

6.5.1. Datu mota

Zenbakiak eta Pertsonaiak

- Zenbakirik gabeko zenbakia (naturala): kodeketa bitar naturala
- Zenbaki oso sinatuak: 2-ren osagarria kodenutako zenbaki osoa
- Zenbaki errealk errealkak IEEE-754 doitasun bakarreko edo bikoitzeko formatuan kodenutako
- Karaktere alfanumerikoak: ASCII kodea

Datuen Ataleko Zuzentaraauak



Lehenengo sei atalak gutxienez eta batez ere irakurtzea **gomendatzen** da [x86 arkitekturaren menpeko zuzentarauei](https://sourceware.org/binutils/docs-2.26/as/i386_002dDependent.html#i386_002dDependent) buruz [https://sourceware.org/binutils/docs-2.26/as/i386_002dDependent.html#i386_002dDependent:]

9. taula. Oinarrizko 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, .idata izena	aldagai mota, adibidez, @funtzioa
.ohikoa 100	hasierarik gabeko 100 byte gordetzen ditu eta izan daiteke globalki erreferentziatuta
.lcomm begizta, 100	ikurrarekin erreferentziatutako 100 byte gordetzen ditu tokiko begizta. Hasieratu gabe.
.espazioa 100	100 byte erreserva zerora hasieratuta

Zuzentaraauak	deskribapena
.100 espazioa, 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, oc <small>halakoak</small>
.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 zenbakiarekin lotzen duen makroa 0
.macro macname macargs	izeneko makro baten hasiera definitzen du macname eta macargs argumentuak
.amaiera makroa	makro baten amaiera definitzen du
.lerrokatu	ondorengo argibideak edo datuak hasiko dira n byteko helbide anitzetan.
.amaiera	muntaketaren amaiera

- Et: Etiketa

6.5.2. x86 eragigaien tamaina

- Eragigaiaren tamaina: MNEMONIKAREN atzizkiak.

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
```

6.5.3. Byte-lerroatzea: Big-Little Endian

- Byte anitzeko datu baten byteak memorian gorde daitezke MSB-_-LSB norabidean edo MSB_-LSB
- Little Endian Alignment : LSB byte esanguratsu gutxien memorian gordetzen da baxuagoa
- **0x40000 adibidea: 00 AF BF CF**

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

ÿ Byteak goranzko memoria helbidean gordetzen dira. idazten duzunean horizontala, goranzkoa ezkerretik eskuinera esan nahi du.

ÿ **0x40000** posizioan 00 byte da 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	EDUKIAK	
0x00000		
0x00001		
0x00002		
0x40000	00	LSB
0x40001	AF	
0x40002	BF	
0x40003	CF	MSB
0xfffff		

ŷ Byte esanguratsu txikiena memoria-kokapen baxuenean gordetzen da. postua gehiago Lauetatik baxua 0x4000 da non 00 gordetzen den, orduan hau pisu txikiagoko bytea da. Little-endian formatuan gordetako datuak **0xCFBFAF00** dira.

ŷ i386/amd64 arkitekturak LITTLE ENDIAN erabiltzen du

- Little endian formatua jarraitzen duten informazio motak.

ŷ Argibideetarako, formatua eremuен araberakoa da, beraz, ez du zentzurik buruz hitz egiteak instrukzioaren pisu handiagoa edo txikiagoa duten posizioak, formatu txikiari jarrai ez dezaten endian.

ŷ Kateek ez dute baliorik adierazten eta, beraz, ez dute jarraitzen little-endian formatua.

ŷ Zenbaki osoak little endian formatuan gordetzen dira.

ŷ Zenbaki errealak little endian formatuan gordetzen dira

ŷ Memoria-helbideak Little Endian erakundeari jarraituz gordetzen dira.

- Big Endian formatua

ÿ Biltegiratze-ordena endian txikiaren alderantzizkoa da, hau da, datuen LSB byte-a da.
datuak hartzen dituen eskualdeko memoria helbiderik handienean gordetzen du.

6.6. Eragileak: Helbideratzeko moduak

6.6.1. Kokapena

- Adibidea:

ÿ begizta: SUBQ \$16,%rsp ;begizta hasiera

ÿ Iturburuko eragigaia: \$-k BEREHALA helbideratza adierazten du Eragigaia instrukzioan bertan dago
Operandoa=16

ÿ Helmuga-eragigaia: %-ek REGISTRO adierazten du. Eragigaia RSP erregistroan dago

ÿ batura: ADDW (%ESI),emaitza ;eragiketaren amaiera

ÿ Iturburu-eragigaia: () ZEHAR eta % erregistroa adierazten du ESI erregistroak
eragigaia dagoen memoria helbidea

ÿ Helmuga-eragigaia: "emaitza" etiketa bat da. ZUZENDARITZA ABSOLUTUA. The
eragiketa memoria helbidean dago "emaitza".

6.6.2. Helbideratzeko moduak

- Muntatzailearen eskuliburua, x86 arkitekturaren menpeko zuzentarauen atala.

ÿ 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 \$ aurritzarekin adierazten da . adibidea: movl \$0xabcd1234, %ebx. Iturburu-eragigaia 0xABCD1234 balioa da
IZEN-EMATEA:	Eragigaiaren balioa CPUko registro batean dago.
	sintaxia: % aurritzia duen erregistroaren izena . adibidea: movl %eax, %ebx. Iturburuko eragiketa EAX REGISTRO da eta helmuga EBX REGISTRO

ZUZENA:	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 helbideratza. Iku bideratza desplazamenduarekin.
	sintaxia: programatzaleak definitutako etiketa adibidea: heh somePlace . Joan etiketak markatutako helbidera somePlace aurreko eragiketaren emaitzak ZF=1 bandera aktibatzen badu RFLAG erregistroarena.
INDEXATUA:	Eragigaiaren balioa memorian kokatzen da. helbide eraginkorra Memoria seinalatuz, base_register PLUS eskalaren balioaren BURUA da register_index-en balioa, GEHI desplazamendua. <i>EA=Offset+R_Base+R_index*Scale</i>
	sintaxia: komaz bereizitako balioen zerrenda parentesi artean (oinarria_erregistroa, indizea_erregistroa, eskala) eta desplazamendu baten aurretik. adibidea: movl \$0x6789cdef, -16(%edx, %eax, 4) . helbide eraginkorra helmuga EDX + EAX*4 - 16 da.
ZEHARKOAK:	Indexatzeko modu orokorra (base_register) atalean zehazten bada orduan, operandoaren helbidea ez da indexatuz lortzen baina helbide eraginkorra rdx-en edukia da eta, beraz, halaxe da zeharka eragiketara sartzen da.
	sintaxia: (base_register) adibidea: movl \$0x6789cdef, (%edx) . Helmugaren helbide eraginkorra da EDX. EDX erakuslea da.
ERLATIBOA: oinarritzko erregistro plus desplazamendu bat:	Eragigaiaren balioa memorian kokatzen da. -ren benetako helbidea Eragiga oinarritzko erregistro batean jasotako balioaren batura gehi balio bat da desplazamenduarena.
	sintaxia: grabatu parentesi artean eta berehalako desplazamendua parentesia. adibidea: movl \$0xaabbccdd, -12(%eax) . -ren benetako helbidea helmuga eragiketa EAX-12 da

adibideak**10. taula. Eragileen helbideratze moduak**

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

- M[loop_exit]: zuzena loop_exit kanpoko memoria-helbide bat baita eta M-k adierazten baitu
kanpoko memoria.

- M[RBX]: zeharkakoa RBX barne-memoriaren helbidea delako eta M-k memoria adierazten duelako kanpoko: To mem. kanpoko mem. bidez sartzen da. barnekoa.

6.7. Argibide multzoa: Eragiketak

6.7.1. erreferentzia eskuliburuak

Intel hizkuntza

- Eskuliburu ofizialak

ÿ [Intel](http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html) [http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html]: 2. liburukia

ÿ [AMD](http://developer.amd.com/resources/documentation-articles/developer-guides-manuals/) [http://developer.amd.com/resources/documentation-articles/developer-guides-manuals/]: atala Eskuliburuak: 3. liburukia

ÿ [binutilak](https://sourceware.org/binutils/docs-2.25/) [https://sourceware.org/binutils/docs-2.25/]:

ÿ [gisa: muntatzaile itzultzalea](https://sourceware.org/binutils/docs-2.25/as/index.html) [https://sourceware.org/binutils/docs-2.25/as/index.html]: Aukerak komando-lerroa, zuzentaraauak, datu-motak, etab.

ÿ [ld:linker](https://sourceware.org/binutils/docs-2.25/ld/index.html) [https://sourceware.org/binutils/docs-2.25/ld/index.html]

ÿ [objdump, readelf,](https://sourceware.org/binutils/docs-2.25/binutils/index.html) [https://sourceware.org/binutils/docs-2.25/binutils/index.html]

- Eskuliburu ez ofizialak:

ÿ [Intel eskuliburu azkarra](http://www.felixcloutier.com/x86/) [http://www.felixcloutier.com/x86/]: **gomendagarria**

ÿ [Intel deskribatzalea i386](http://www.cs.nyu.edu/~mwalfish/classes/ut/s13-cs439/ref/i386/toc.htm) [http://www.cs.nyu.edu/~mwalfish/classes/ut/s13-cs439/ref/i386/toc.htm]

ÿ [ISA Errepertorioa eta Instrukzio Formatura](http://www.cs.nyu.edu/~mwalfish/classes/ut/s13-cs439/ref/i386/c17.htm) [http://www.cs.nyu.edu/~mwalfish/classes/ut/s13-cs439/ref/i386/c17.htm]

AT &T Solaris eskuliburu amd64-i386 [http://docs.oracle.com/cd/E19253-01/817-5477/817-5477.pdf]:
muntaia hizkuntza eta itzultzalea.

ÿ [Baldintzazko jauziak](http://www.unixwiz.net/techtips/x86-jumps.html) [http://www.unixwiz.net/techtips/x86-jumps.html]

AT&T hizkuntza

- [Oracle Solaris ASM](https://docs.oracle.com/cd/E53394_01/html/E54851/eqbsu.html) [https://docs.oracle.com/cd/E53394_01/html/E54851/eqbsu.html]

ÿ AT&T-k dokumentu honetako sintaxia "Oracle Solaris" gisa aipatzen du.

6.7.2. TRANSFERENTZIA

izen eta	Iruzkina	Kodea	Operazioa	ODITSZAPC
	MOV Mugitu (kopiatu) MOV	Iturria, Dest	Dest:=Iturria	
	XCHG Exchange XCHG Op1,Op2 Op1:=Op2, Op2:=Op1			

izen eta	Iruzkina	Kodea	Operazioa	ODITSZAPC
STC	Ezarri eramatea (eraman = 1)	STC	CF:=1	1
CLC	Garbitu Eramatea (eraman = 0)	CLC	CF:=0	0
CMC Osagarria Eraman		CMC	CF:=∅	±
ETsak	Ezarri STD helbidea		DF:=1(interpretatu goiko kateak behera)	1
CCD Garbitu helbidea		UNCCD	DF:=0 (interpretatu beheko kateak gora)	0
ITS	ren bandera sartu 1	ITS	BADA:=1	1
CLI	ren bandera sartu 0	CLI	BADA:=0	0
PUSH Push pila PUSH Iturria			DEC SP, [SP]:=Iturria	
PUSHF PUSHF banderak pilatu			O, D, I, T, S, Z, A, P, C 286+: Baita NT, IOPL	
PUSHA Pilatu erregistroak orokorra		BULTZATU	AX, CX, DX, BX, SP, BP, SI, DI	
POP Despilaketak pilatu		POP Helmuga	Helmuga:=[SP], INC SP	
POPF despilatzen du banderak		POPF	O,D,I,T,S,Z,A,P,C 286+: Baita NT, IOPL	± ± ± ± ± ± ± ±
POPA Despilatzen du erreg. orokorra.		STERN	DI, SI, BP, SP, BX, DX, CX, AX	
CBW Bihurtu byte-ra Hitza		CBW	AX:=AL (sinatua)	
CWD Bihurtu Word Bikoiztera		CWD	DX:AX:=AX (sinatua)	
CWDE konb. hitzari Luzapen bikoitza		CWDE 386	EAX:=AX (sinatua)	
IN	Sarrera	IN Dest,Port AL/AX/EAX := byte/hitza/doble del esp ataka		
OUT Irteera		OUT Puer, Iturburu	Byte/hitza/ataka bikoitza := AL/AX/EAX	

- Banderak: ± =Instrukzio honek eraginda, ? =Definitu gabea adierazpen honen ondoren

6.7.3. ARITMETIKA

izena eta	Iruzkina	Kodea	Operazioa	ODITSZAPC
GEHITU Batura		GEHITU Iturburua,Dest Dest:=Dest+ Ituria		± ± ± ± ± ±
ADC Sum-ekin garraioa		ADC Ituria,Dest Dest:=Dest+ Ituria +CF		± ± ± ± ± ±
AZPI	Kenketa	SUB Source,Dest Dest:=Dest- Ituria		± ± ± ± ± ±
SBB	Kenketarekin garraioa	SBB Ituria,Dest Dest:=Dest-(Ituria +CF)		± ± ± ± ± ±
DIV	zatiketa (gabe seinaldea)	DIV Op	Op=byte: AL:=AX / Op AH:=Hondarra? ? ? ? ?	
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:=Gainera	? ? ? ? ? ?
IVDI	osoko zatiketa zeinuarekin	IVDI Op	Op=byte: AL:=AX / Op AH:=Hondarra? ? ? ? ?	
IVDI	osoko zatiketa zeinuarekin	IVDI Op	Op=hitza: AX:=DX:AX / Op DX:=Atseden	? ? ? ? ? ?
IVDI	386 Dibisioa sinatutako zenbaki oso	IVDI Op	Op=doblew.: EAX:=EDX:EAX / Op EDX:=Gainera	? ? ? ? ? ?
MUL Biderketa (ez dago seinalerik)		MUL Op	Op=byte: AX:=AL*Op AH=0 bada #	± ? ? ? ? ±
MUL Biderketa (ez dago seinalerik)		MUL Op	Op=hitza: DX:AX:=AX*Op baldin DX=0 # ± ? ? ? ? ±	
MUL 386 Biderketa (ez dago seinalerik)		MUL Op	Op=bikoitza: EDX:EAX:=EAX*Op bada EDX=0 #	± ? ? ? ? ±
IMUL i Multiplic. sinatutako zenbaki oso		IMUL Op	Op=byte: AX:=AL*Op AL bada nahikoa #	± ? ? ? ? ±
IMUL biderkatzailea osorik zeinuarekin		IMUL Op	Op=hitza: DX:AX:=AX*Op AX bada nahikoa #	± ? ? ? ? ±
IMUL 386 Multip. sinatutako zenbaki oso		IMUL Op	Op=bikoitza: EDX:EAX:=EAX*Op bada EAX sufia da. #	± ? ? ? ? ±
INC	Handitu	INC Op	Op:=Op+1 (The Carry ez da kaltetua!)	± ± ± ± ±

izena eta	Iruzkina	Kodea	Operazioa	ODITSZAPC
	DEC Decrement DEC Op		Op:=Op-1 (The Carry ez da kaltetua!)	± ± ± ± ±
	CMP Konparatu	CMP iturria, helmuga	Helmuga-Iturria	± ± ± ± ± ±
GATZA	Desplazamendua aritmetika. ezkerretara	GATZA	Op, Kantitatea	i ± ± ? ± ±
	Desplazatutako SAR aritmetika. eskuinera	BERAK	Op, Kantitatea	i ± ± ? ± ±
LCR	biratu ezkerrera eraman/eraman	RCL Op,Kantitatea		i ±
	RCR Biratu eskubidea eraman/eraman	RCR Op, Kantitatea		i ±
	ROLA Biratu hona ezker	ROLA Op,Kantitatea		i ±

- i: informazio gehiagorako, ikusi argibideen zehaztapenak,
- #: gero CF:=0, OF:=0 bestela CF:=1, OF:=1

6.7.4. LOGIKOA

izena eta	Iruzkina	Kodea	Operazioa	ODITSZAPC
	NEG Ezeztapena (osagarria bi)	NEG Op	Op:=0-Op Op=0 bada CF:=0 bestela CF:=1	± ± ± ± ± ±
	EZ alderantzikatu bit bakoitzza EZ Op		Op:=Ø-Op (bit bakoitzza alderantzikatzen du)	
	AND Logikoa ETA Iturria,Dest Dest:=Dest ^ Iturria			0 ± ± ? ± 0
EDO	Logikoa EDO EDO Iturria,Dest Dest:=Dest v Iturria			0 ± ± ? ± 0
	XOR Exclusive OR (Or) XOR Source,Dest Dest:=Dest (xor) Iturria			0 ± ± ? ± 0
	SHL Offset ezkerrera logikoa	SHL Op, Kantitatea		i ± ± ? ± ±
	SHR Offset eskuinera logikoa	SHR Op, Kantitatea		i ± ± ? ± ±

6.7.5. DENBERAK

izena eta	Iruzkina	Kodea	Operazioa	ODITSZAPC
NOP	Ez egin ezer	NOP	ez du inolako eragiketarik egiten	
LEA	Kargatu helbidea eraginkorra	IRAKURRI Iturburua, Dest Dest := iturburu helbidea		
INT	Etenaldia	INT Zenb	Uneko prozesua eteten du eta salto Num bektorera	0 0

6.7.6. JUSTOAK (orokorra)

- [wiki x86 muntaia](https://en.wikibooks.org/wiki/X86_Assembly/Control_Flow#Jump_if_Lesser) [https://en.wikibooks.org/wiki/X86_Assembly/Control_Flow#Jump_if_Lesser]

Izena Iruzkina	Kodea	Operazioa
deitu	Azpirrutina deia	DEIALDIA Proc
JMP	Saltatu	JMP Helmuga
je	Saltatu berdina bada	JE Dest (=JZ)
JZ	salto zero bada	JZ Dest (= JE)
JCXZ	Saltatu CX zero bada	JCXZ Dest
JP	Saltatu parekotasuna badago	JP Dest (= JPE)
JPE	Saltatu parekotasuna ere badago	JPE Dest (=JP)
JPO	Saltatu parekotasun bakoitia badago	JPO Dest (=JNP)
JNE	Saltatu berdina ez bada	JNE Dest (=JNZ)
JNZ	Salatu zero ez bada	JNZ Dest (=JNE)
JECXZ	Jauzi ECX zero bada	JECXZ Dest 386
JNP	Saltatu parekotasunik ez bada	JNP Helmuga (= JPO)
RET	azpierrutinaren itzulera	RET

6.7.7. JUMPS Sinatu gabe (kardinala) JUMPS Sinatuta (Osokoa)

Izena Iruzkina JA Saltatu handiagoa bada Saltatu	Kodea	Operazioa
JAE Dest Handiagoa edo	JA Dest	(= JNBE)
JBE berdina bada Saltatu baino txikiagoa bada Saltatu JBE Dest		(=JNB=JNC)
JB txikiagoa bada edo berdina bada Saltatu handiagoa Destada	Destada	(= JNAE = JC)
JNAE JNA Dest Saltatu super ez bada. edo berdina JNAE Dest		(=JNA)
JNA Saltatu txikiagoa ez bada Saltatu txikiagoa ez badeada	Destada	(= JBE)
JNAE JNBE Dest Saltatu eramanez gero JC Dest Saltatu eraman		(=JB=JC)
JNB ezean Saltatu gainezka egin ezean Saltatu seinala Destada	Helmuga	(=JAE=JNC)
JNBE (=negatiboa) JS Dest		(= HA)
JC	JO Dest	Salto gainezka badago
JNC	JNC Dest	
JNO	JNO Dest	
JS		
JG Saltatu handiagoa bada JG Dest Saltatu baino handiagoa		(= JNLE)
JGE edo berdina bada Saltatu txikiagoa bada Saltatu txikiagoa	JGE Dest	(=JNL)
JL bada edo berdina bada Saltatu handiagoa ez badala L. Dest		(= JNGE)
JLE	JLE Dest	(=JNG)
JNG	JNG Dest	(= JLE)

JNGE	Saltatu JNGE Dest baino handiagoa edo berdina ez bada	(=JL)
JNL	Jauzi behera ez bada	JNL Dest (= JGE)
JNLE	Saltatu JNLE Dest baino txikiagoa edo berdina ez bada	(=JG)

6.7.8. BANDERAK (ODITSZAPC)

O: gainezka funtzionamenduaren emaitza. sinatu gabekoa oso handia edo txikia da.

D: helbidea

I: Etenaldia Etenaldiak gerta daitezkeen edo ez adierazten du.

T: Trap Step, arazketarako urratsez urrats

S: Zeinua emaitzaren seinale. Zenbaki osoentzat bakarrik arrazoizkoa. 1=ez. 0=pos.

Z: Zero Eragiketaren emaitza zero da. 1=Zero

Nori: Carru Aux. Carry-ren antzekoa, baina nibble baxurako soilik mugatuta

P: Parekidasuna 1 = emaitzak batean bit kopuru bikoitia du

C: Eragiketaren emaitza eraman. sinatu gabekoa oso handia da edo zero baino txikiagoa da

6.7.9. atzizkiak

- Opcode mnemoteknikoaren atzizkiak:

ÿ q : quad: 8 byteko eragigaia: quadword

ÿ l : luzea: 4 byteko eragigaia: hitz bikoitza

ÿ w : hitza: 2 byteko eragigaia: hitza

ÿ b : byte: 1 byteko eragigaia

- Eragiketa mnemoteknikoak atzizkirik ez badu, eragigaiaren tamaina lehenetsia /luzea da.

6.7.10. Eragiketa Kodeak

- [MOV \[http://www.cs.nyu.edu/~mwalfish/classes/ut/s13-cs439/ref/i386/MOV.htm\]](http://www.cs.nyu.edu/~mwalfish/classes/ut/s13-cs439/ref/i386/MOV.htm)

MOV -- Mugitu datuak

Opcode Instrukzioa	Erlojuak	Deskribapena
88 /r MOV r/m8,r8 89 /r MOV r/m16,r16 89 /r MOV r/m32,r32	2/2 2/2 2/2	Eraman byte-erregistroa r/m bytera Eraman hitz-erregistroa r/m hitzera Eraman dword erregistroa r/m dword-ra
8A /r MOV r8,r/m8	2/4	Mugitu r/m byte byte erregistrarra
8B /r MOV r16,r/m16	2/4	Mugitu r/m hitzetik hitz erregistrarra
8B /r MOV r32,r/m32	2/4	Eraman r/m dword dword erregistrarra
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-erregistrarra
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-ra (seg: offset) EAXera
A2 MOV moffs8,AL 2		Eraman AL-ra (seg:desplazamendua)

A3	MOV moffs16,AX	bi	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

ÿ dword :bikoitza: 32 biteko

ÿ ASM hizkuntzaren sintaxia ez da AT&T Intel ü mnemoteknia baizik
Helmuga_operando, iturburuko eragiketa

Programa Adibideak

6.8. Oinarrizko mnemoteknia (azalduta)

6.8.1. Eragiketa aritmetikoak

- **mul:** zenbaki naturalen biderketa, zeinurik gabe

Lehenengo eragigaiaren (helmugako eragiketa) biderketa sinatu gabea egiten du eta bigarren eragigai (iturburuko eragiketa) eta emaitza helmugan gordetzen du operandoa. Helmuga-eragigai AL, AX erregistroan dagoen eragiketa inplizitua da edo EAX (eragigaiaren tamainaren arabera); iturburu-eragilea a-n kokatzen da erabilera orokorreko erregistroa edo memoria-kokapen bat

- **mul:** zenbaki osodunen biderketa

ÿ 1,2 edo 3 eragiketa izan ditzake

Bi eragigaien biderketa sinatua egiten du. Argibide honek hiru ditu formak, eragiketa kopuruaren arabera.

Eragiketa bakarreko forma — Forma hau MULek erabiltzen duenaren berdina da instrukzioa. Hemen, iturburuko eragigai (erabilera orokorreko erregistro edo memoria batean kokapena) AL, AX, EAX edo RAX erregistroko balioarekin biderkatzen da (eragilearen tamainaren arabera) eta produktua (sarreraren tamainaren bikoitza). eragiketa) AX, DX:AX, EDX:EAX edo RDX:RAX erregistroetan gordetzen da, hurrenez hurren.

ÿ imull Label : R[%edx]:R[%eax] ÿ M[Label] × R[%eax]

- **div:** zenbaki naturalen zatiketa, zeinurik gabe
- **idiv:**

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 **bizia**

ÿ EAXÿQuotient{[EDX:EAX]/M[Op_source]}, EDXÿHondarra{[EDX:EAX]/M[Op_source]}

- Sinadura luzapena

ÿ movsbw src,Reg ÿ Mov Sign Byte Word-era

ÿ movsbl src,Reg ÿ Mov Sign Byte to Long

ÿ movswl rc,Reg ÿ Mov Sign Word to Long

- Tamaina aldatu

ÿ movzbw src,Reg ÿ Mov byte Word-era

ÿ movzbl src,Reg ÿ Mugitu byte luzera

ÿ movzwl src,Reg ÿ Mugitu byte luzera

6.8.2. Baldintzapeko tratamendua

Boolearra eta konparazioa

- EZ

ÿ banderarik ez

- ETA

ÿ Garbitu CF,OF

ÿ SF,ZF,PF aldatzen ditu

- EDO

ÿ Garbitu CF,OF

ÿ SF,ZF,PF aldatzen ditu

- X-OR

ÿ Garbitu CF,OF

ÿ SF,ZF,PF aldatzen ditu

- PROBA

ÿ Garbitu CF,OF

ÿ SF,ZF,PF aldatzen ditu

- WPC

ÿ CF,OF,SF,ZF,PF,AF aldatzen ditu

6.8.3. jauziak

zeharkakoa

- Jauzietan zeharkakotasuna adierazteko eta helbide erlatibotik bereizteko ikurra. Bestalde, MOV mugimenduetan, ikurra ez da beharrezkoa, ez baitago jauzi erlatiborik.

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

6.8.4. Korritu eta Biratu

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

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

Lehenengo eragigaiko (helmugako eragigai) bitak ezkerrera edo eskuinera desplazatzen ditu bigarren eragigaian zehaztutako bit kopuruaren arabera (zenbaketa eragigaia). Helmugako operandoaren mugatik haratago desplazatutako bitak CF banderara desplazatzen dira lehenik, eta gero baztertzen dira. Desplazamendu-eragiketaren amaieran, CF banderak helmugako eragiketatik desplazatutako azken bita dauka.

ÿ **sarl \$31, %edx :** mugitu 31 bit eskuinera eta sarrerako bita zeinua bit da

EDX-n negoziatzen.

- shr,shl
- desplazamendu logikoa: zeroak sartu
 - ÿ Biderketa eta zatiketa adibideak
- ROLA, ROR
 - ÿ Irtetzen den bita CF-ra kopiatzen da
 - ÿ Aplikazioa: endianess bihurketa

6.8.5. Aldatu Endianess

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

swapbytes:

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

```
xchg (%ebx), %eax
```

6.9. Argibide-formatua: ISA Intel x86-64

- Eranskina [Argibide-formatua](#)

6.10. Azpirrutinak

6.10.1. Sarrera

- Mihiztadura-lengoaian azpierrutinak C programazio-lengoaian funtzioen baliokideak dira, beraz, C lengoaian funtzio baten kontzeptua berrikustea beharrezkoa da.
- Azpirrutinen erreferentziak Eranskineko 5. praktika eta 14. kapitulua dira.

6.10.2. C hizkuntza: adierazpen-funtzioa

Sarrera

Funtzioen helburua programa kode-moduluetan banatzea da, programari programaren garapena eta mantentze-lana errazten duen egitura antolatu bat emateko. "libc" liburutegi estandarrak aplikazioan garatutako oinarrizko funtzioen bildumak dira

Programa gehienek berrerabiltzen duten C lengoaia. Horrela programatzaleak ez du gurpila asmatu beharrik. Beraz, erabiltzaileak C lengoaian garatutako funtzioak eta kode bitarrean eskura daitezkeen liburutegiko funtzioak elkarrekin bizi dira programa batean.

Adierazpena

- C hizkuntzan funtzi baten adierazpenari **prototipo deitzen zaio**. Prototipoaren adibidea: `int sumMtoN(gehigarri laburra1, gehigarri laburra2) non`
 - ÿ Name: funtziaren izena *sumMtoN* da
 - ÿ Argumentuak: lehen argumentuaren izena *addend1* da eta labur motakoa da, izena
2. argumentuaren *summand2* da eta shrot motakoa da.
 - ÿ Itzultzeko balio-mota: itzulera-balioaren mota int da.

Definizioa

- *SumMtoN* funtziaren definizioa enuntziatuen bidez algoritmoa garatzean datza
C-ren, hau da, funtziaren gorputza:

```
int sumMtoN(gehigarri laburra1, gehigarri laburra2) { //
    add2 > gehigarri1 laburra i; int emaitza=0; //
    aldagai lokala i=gehitzea2 funtziolan; bitartean (i
    >= gehigarri1) { emaitza += i ;
```

```

    Yo--;
}
printf("\n\t sumMtoN azpierrutina \n");
emaitza itzuli;
}

```

ÿ emaitza itzuliko balioa duen aldagai da

Deitu eta Itzuli

- *Main()* funtzioko *sumMtoN()* funtziari deitzen dio eta, exekutatu ondoren, itzultzen du batura emaitza.

```

/*
Programa: sumMtoN.c
Konpilazioa: gcc -g -ggdb3 -o sumMtoN sumMtoN.c
            -ggdb3 : txertatu informazioa arazketa-ikurren taulan
makroak
*/

// Funtzio-prototipoak
#include <stdio.h> // Printf() funtziaren deklarazioa
#include <stdlib.h> // Exit() funtziaren adierazpena

//Makroa
#define ARRAKASTA 0

//Prototipoak: sumMtoN() funtziaren deklarazioa
int sumMtoN(gehigarri laburra1, gehigarri laburra2);

// Funtzio nagusiaren definizioa main()
void main(void) {
    //sumMtoN() funtziaren M eta N argumentuen hastapena
    laburra M=1;
    laburraN=1;
    // Funtzio-deia
    printf("Burketaren emaitza %d \n da", sumMtoN(M,N));
    // sumMtoN-ren ebaluazioa honako hauek datza: funtziari deitzea eta harrapatzea
    itzultzeko balioa.
    irten(ARRAKASTA);
}

// Funtzioen definizioa
int sumMtoN(gehigarri laburra1, gehigarri laburra2) {
    //gehitzenean > gehitzenean1
    i laburra;
    int emaitza=0; // funtziaren tokiko aldagai
    i=gehitzenean2;
    bitartean (i >= gehigarri1) {
        emaitza += i ;
    }
}

```

```

    Yo--;
}
printf("\n\t sumMtoN azpierrutina \n");
emaitza itzuli;
}

```

ÿ printf ÿ sumMtoN : printf- k *sumMtoN()* funtziaren **ebaluazioaren** emaitza inprimatzen du . Ebaluazioa *sumMtoN()* funtziaren exekuzioaren **itzulera-balioa** lortzean datza.

6.10.3. Funtzia Habiaratzea

- Deitu: init() ÿ main() ÿ sumMtoN() ÿ printf() ÿ write()
- Sistema eragileak programatzalearen funtzi nagusiari deitzen dio, eta honek, aldi berean, programatzalearen sumMtoN() funtziari deitzen dio, eta horrek libc liburutegiko printf() funtziari dei egiten dio, ~~sistemaek~~ eragileari dei egiten dio sistemaren write() funtzia exekutatzeko.
- Itzuli: write() ÿ printf() ÿ sumMtoN() ÿ main () ÿ exit()

6.10.4. Pila/Markoa

- Ikusi bateriaren kontzeptua [Eranskinean](#).
- Pila memoria nagusian dagoen programaren *atal* bat da datuen atala eta bezalakoa argibideak atalean.
- *SumMtoN* azpierrutinaren M eta N argumentuak pilara pasatzen dira.
- Pila zatiketa fotogrametan: errutina eta azpierrutina bakoitzak bere pila - *segmentua* du, alegia deiak **markoa**.
 - ÿ Errutina *nagusiak* bere markoa du eta *sumMtoN* azpierrutinak bere markoa.
 - ÿ Fotogramak azpierrutinen deiak habiaratu ahala pilatzen dira.
 - ÿ Dinamismoa: Programaren exekuzioaren une jakin batean, sortutako azken fotograma da marko aktiboa.
 - ÿ Marko aktiboaren behealdea EBP erakusleak eta markoaren goiko aldea aipatzen dute ESP erakuslearen bidez.

6.10.5. Azpirrutinaren definizioa

- Izena: *sumMtoN*

• Azpirrutinaren izena azpirrutinaren lehen instrukzioa seinalatzen duen etiketa da.

• Azpirrutina ret instrukzioarekin amaitzen da.

- Azpirrutina 3 zatitan egituratuta dago:

ÿ Hitzurrea:

Yo. Gorde azpirrutinaren gorputzak aldatuko dituen erregistroak.

ii. Aktibatu marko berria **EBP** eta **ESP** erakusleak hasieratuz.

ÿ Gorputza:

Yo. Hartu argudioak eta prozesatu

ŷ Epilogoa:

Yo. Gorde itzuliko balioa **EAX** erregistroan

ii. Berreskuratu Prolog-en gordetako erregistroen balioa

iii. Aktibatu deia egin duen funtziaren markoa **EBP** eta **ESP** egunerautz
balio zaharrak.

IV. Itzuli deia egin duen funtziora.

- Kodea

```
# Azpirrutinaren hasiera
batuketa:
# Hitzaurrea
    push %ebp # gorde deitzen duen funtziaren markoaren behealdea zatian
marko berria deskargatu
    mov %esp,%ebp # ezarri %ebp erakuslea behealdean
marko berria
    push xxx          # Beharrezko bida: gorde erabiliko diren erregistroak
Gorputz azpierrutina
    bultzatu xxx
# Gorputza
    mov 8(%ebp),%ebx #harrapatu 1. argumentua
    mov 16(%ebp),%ecx #harrapatu 2. argumentua
    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
```

6.10.6. Kontserbatzeko Erregistroak

Erref

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

errutina deituz

Deitzeko errutina (deitzailearen errutina) beharrezko da honako erregistro hauek gordetzeko baldin erabiltzen ari zara:

- EAX-ECX-EDX

Hau da, erregistro hauek libreki erabil ditzake azpierrutina izenekoak. Azpierrutinak erabiltzen ez baditu, ez litzateke beharrezkoia izango kontserbatzea. Erabiliz gero, pilara kopiaturako lirateke azpierrutina deitu aurretik eta azpierrutinaren amaieran berreskuratuko lirateke.

azpierrutina izenekoak

Deitutako azpierrutina (kalearen errutina) beharrezkoia da erregistro hauek gordetzeko:

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

Hau da, azpierrutinaren amaierako erregistro hauek deiaren aurreko balio bera mantentzen dute. Erabili ezean, ez litzateke beharrezkoia izango kontserbatzea.

amd64 arkitektura

Deitzaileen errutina: RAX, RCX, RDX, R8, R9, R10, R11 erregistroak lurrunkortzat jotzen dira eta funtziodeietan suntsituztak jo behar dira (salbu eta analisiaren bidez segurtasuna frogatzen ez bada, hala nola, programa osoaren optimizazioa).

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

6.10.7. Azpierrutinen argumentuak

- Argudioak pilatik pasa behar dira eta deia egin aurretik.
- Argumentuak bata bestearen atzetik pilatzen dira azken argumentutik hasita eta lehen argumentuarekin amaituz.
- **Argumentu push** instrukzioaren bidez bultzatzen dira, non eragigaia argumentua den transferitzeko.

bultzatu N
bultzatu M

6.10.8. Deitu azpierrutinara

- Dei-errutina nagusiak *sumMtoN azpierrutina* deitzen du **deia sumMtoN sententzia erabiliz**. Beraz, errutina *nagusia* eten egiten da *sumMtoN azpierrutinaren execuzioa amaitu arte*.
- **Deiaren** instrukzioa bi fasetan exekutatzen da:
 - a. Pila itzultzeko helbidea: errutina *nagusian sumMtoN deitzeko* instrukzioa jarraituz : **ESP** ÿ **ESP-4** eta **M[ESP]** ÿ **PC**
 - b. Saltatu *sumMtoN* etiketara: **PC** ÿ **sumMtoN**
- Funtsean, dei-instrukzioa eten zen helbidera itzultzen den salto bat da deitzeko errutina.

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

6.10.9. Azpirrutinatik itzultzea

- Azpirrutinaren azken instrukzioa **RET** da, zeinaren exekuzioak PUZaren Kontrol Unitateak agindu hauek betetzen dituen:
 - a. **PC ѕ M[ESP]** : CALL instrukzioak gordetako itzulera-helbidea pilatik ateratzen du eta Programa-kontagailuan kargatzen du, beraz, **sumMtoN deia** ondoren instrukzioaren instrukzio-zikloa exekutatuko da.
 - b. Eguneratu pila erakuslea: **ESP ѕ ESP + 4**
- Beharrezko da azpierrutinaren epilogoa, RET exekutatu aurretik, pila-erakuslea. Seinalatu itzultzeko helbidea gordetzen den pila-helbidera.

6.10.10. bateriaren egoera

Deitik azpierrutinara salto egin baino lehen

- 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:

ÿRIPS:

ÿ EBP:

ÿ ESP:

Deitik azpierrutinara jauzi egin ondoren

- *SumMtoN* azpierrutina exekutatzen duen pilaren egoera, salto- **deia** exekutatu ondoren batuketa:
 - ÿ Pilaren marko aktiboa nagusiari dagokiona da.
 - ÿ *Marko nagusian* pilatutako azken datuak *sumMtoN*-tik nagusira *itzultzeko helbidea* da
- EIP, EBP, ESP erregistroen edukia:

ÿRIPS:

ÿ EBP:

ÿ ESP:

SumMtoN marko berriaren sorrera

- Bateriaren egoera exekutatu ondoren:

```
sumMtoN:  
bultzatu %ebp mov %esp,%ebp
```

- EIP, EBP, ESP erregistroen edukia:

ÿ PID:

ÿ EBP:

ÿ ESP:

Itzulerako jauzia baino lehen

- *SumMtoN azpierrutina* exekutatzen duen pilaren egoera **ret** instrukzioa exekutatu aurretik:

ÿ Pilaren fotograma aktiboa sumMtoN-ri dagokiona da.

ÿ SumMtoN markoaren goiko *ESP* erakusleak hau duen pila-helbidera seinalatzen du
itzultzeko helbidea

- EIP, EBP, ESP erregistroen edukia:

ÿ PID:

ÿ EBP:

ÿ ESP:

Itzulera osteko jauzia

- *SumMtoN azpierrutina* exekutatzen duen pilaren egoera **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:

ÿ PID:

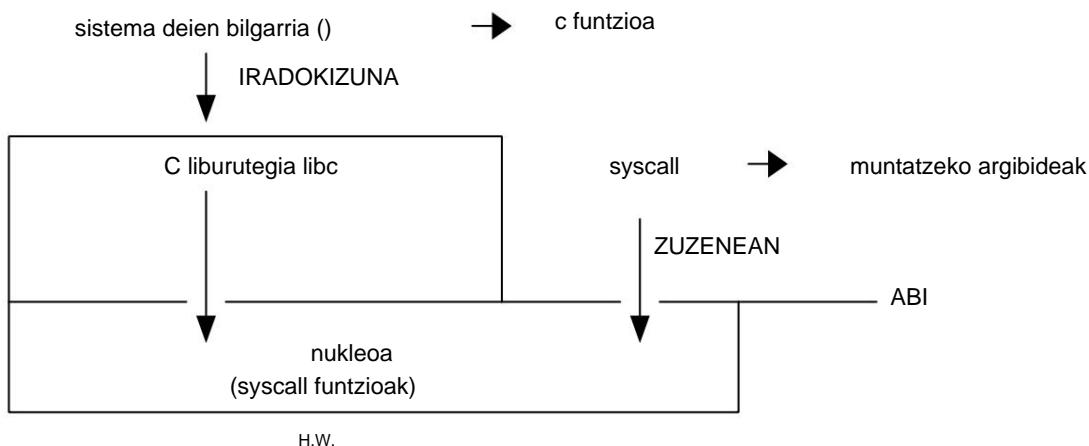
ÿ EBP:

ÿ ESP:

6.11. Sistema eragilera deiak

6.11.1. Sarrera

- Programak egiten dituen deiak *sistema* deien izenarekin ezagutzen dira . erabiltzailea Sistema Eragilearen Kernelaren azpierrutinetara.
- Sistema eragilearen funtziopriilegiatuak burutzeko, hala nola ordenagailuko I/O gailuetarako sarbidea, beharrezko da erabiltzailearen programek nukleoa deitzea, eragiketa seguru eta eraginkortasunez burutzen duen nukleoa izan dadin. Honek aplikazioaren programatzalea hardwarea 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 kernel zerbitzuei deitzea bi modutara egin daiteke:
 - zuzena** edo **zeharkakoa**
 - ÿ Zuzena: [ASMtik syscall](#) instrukzioa erabiliz
 - ÿ Zeharkakoa: C edo ASMtik [libc](#) liburutegiko funtziok erabiliz: dei-bilgarriak zuzena
- API/ABI



- Adibidea i386 arkitektura

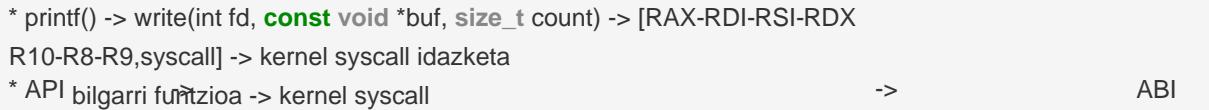
ÿ Lehenengo 6 argumentuak erregistroetatik pasatzen dira: [EBX-ECX-EDX-ESI-EDI-EBP]
[int 0x80](#) instrukzioa deitu aurretik

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



- Adibidea amd64 arkitektura

ÿ Lehenengo 6 argumentuak erregistroetatik pasatzen dira: [RAX-RDI-RSI-RDX-R10-R8-R9]
syscall instrukzioaren aurretik



6.11.2. adibideak

- Ikusi [Eranskinean](#)

6.12. Bibliografiak

- [Muntatzaileen Programazio Zerrenda](#) .
 ÿ Osatu [WikiBookeko](#) oharrak AT&T hizkuntza-programazioari buruzko aniztasunarekin
 alderdiak.

VIII Eranskinak

7. kapitula. IASSim Muntaketa Programak

7.1. 1. adibidea: sum1toN.ias

7.1.1. iassim muntaia-lengoaia

- **2. bertsioa:** *sum1toN_v2.ias* iturburu-modulua :

ŷ 2. bertsioak begizta bat implemenatzen du zeinen gorputzak batuketa partziala egiten duen laburpenak aldatu egiten dira iterazio bakoitzean.

```
;;;;;; OHARRA
; 2. bertsioa: sum1toN_v2.ias
; ;;; JARRAIBIDEEN ATALA
;Metagailuei zuzendutako arkitektura (AC)
;Erregistro eskuragarriak : AC
; algoritmoa: batura = n + batura eragiketa egiten duen begizta n>=0 bada
; hasi begizta
begizta: S(x)->Ac+ n          ; AC <- M[n].Karga gehigarria
      Cc->S(x) gehitu ; AC >= 0 bada, PC <- gehigarria bada < 0 amaieran
begizta
      gelditu           ; gelditu
      .hutsik ; 20 biteko 0 bat, jarraibide kopurua bikoitia izan dadin.
      ; egin batura
gehitu: S(x)->Ah+ gehitu ; AC <- AC + M[batura]
      At->S(x) batura ; M[batura] <- AC

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

ŷ Datuen atala garatu da.

ŷ BEIZKETA bat egin da SUM eragiketarekin begiztaren gorputzean eta STOP batekin.
begiztak atera.

ŷ Kodea IRUZKINATU da

- **Demo bertsioa** *tutorial.ias*: deskargatzeko fitxategian sartzen den demo bertsioa simulagailua.

```
begizta: S(x)->Ac+ n          ;kargatu n AC-ra
      Cc->S(x) pos ; AC >= 0 bada, jauzi pos
      gelditu ;bestela eginda
      .hutsik ;a 20 biteko 0
mezu:      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 izena iturburu-moduluari: *sum1toN.ias*

ÿ programa berrargitaratu etiketa eta iruzkinekin gaztelaniaz.

ÿ kodea komentatu aurreko faseetan deskribatutako moduluen informazioarekin
Programaren garapena

```

;;;;;; OHARRA
; iturburu-modulu sum1toN.ias
; Kalkulatu zenbaki osoen segida baten batura: batura = 1+2+..+n
; sarrerako datuak: N eta irteerako datuak: batura
; Algoritmoa: N iterazioen begizta
; Gehitzaleak na -1 beheranzko ordenan sortzen dira
; Gehitzalea negatiboa bada -> -1 , batura 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
;Metagailuei zuzendutako arkitektura (AC)
;Erregistro eskuragarriak : AC
; algoritmoa: n, n-1, .... -1 gehigarriak sortzen dituen begizta
; eta batura = n + batura eragiketa egiten du n>=0 bada

; start loop : gehigarrien batuketa eta sorrera
begizta: S(x)->Ac+ n ;karga gehigarria
Cc->S(x) gehitu ;bada gehigarria < 0 begizta amaieran
; begiztaren amaiera
; gelditu
;20 biteko 0a, jarraibide kopurua bikoitia izan dadin.

gelditu .hutsik ; egin batura
gehitu: S(x)->Ah+ batura ;
At->S(x) batura ;
; eguneratzea 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 .datuak
n: 5 ; gehitzea eta hasieratzea
batura: .datuak 0 ; batuketa partziala eta
amaierakoa; konstanteak bat: .datuak 1
;

7.1.2. 2. adibidea: Produktua/Kozientzia

adierazpena

- Eragiketa egiten duen programa garatu $N(N + 1) / 2$ emaitza lortzeko baliokidea

$$\text{Tutoretza1 baturatik} \quad \sum_{i=1}^N i = N(N + 1) / 2$$

- ŷ Algoritmoaren pseudokodea
- ŷ Algoritmoen fluxu-diagrama
- ŷ Programa RTL hizkuntzan ŷ Komentatu programa behar bezala (goiburua metainformazioa, egitura-atalak, bloke funtzionalak).
- ŷ Mihiztadura-lengoaiaren programa
- ŷ Programa exekutatu urratsez urrats, biderketa eta zatiketa gauzatzean erregistroen balioa aztertuz.

- RTL deskribapenean KONTUAN HARTU:

ŷ M zifrako bi zenbakiren biderkadurak 2M zifrako zenbakia sortzen du, hau da, biderkatzaileen bikoitza. Horrek zaildu egiten ditu biderketa osteko eragiketa aritmetikoak matematika-adierazpenean. Hori dela eta, biderketa eragiketa amaierarako utziko dugu, batuketari eta zatiketari lehentasuna emanez.

$$\rightarrow N(N + 1) / 2 = ((N + 1) / 2) * N$$

ŷ Zatiketak 1 edo 0 hondarra izan dezake dibidendua bikoitia edo bakoitia denaren arabera

$$\text{ŷ N bakoitia bada } \hat{y} (N+1)/2 \text{ hondarra 0 } \hat{y} \text{ du } ((N + 1) / 2) * N \text{ non } N+1 \text{ bikoitia den}$$

$$\text{ŷ N bikoitia bada } \hat{y} (N+1)/2 \text{ hondarra 1 badu } \hat{y} (N+1) = \text{zatidura}*2+\text{hondarra } \hat{y} \\ N(N + 1) / 2 = N * C + N / 2 \text{ non } N \text{ bikoitia den}$$

ŷ 21 bezalako 2 potentziatz zatitzea eragiketa logiko baten bidez egiten da: dibidendua 1 bit ezkerrera eraman. Desplazatu beharreko bit kopurua berretzailearen balioa da.

ŷ deskribapena RTL AC \hat{y} AC<<1

pseudokodea

- Algoritmoaren deskribapena NATURAL hizkuntzan testu-moduko esamoldeak erabiliz.
- ALDAGAIAK:

ŷ batura aldakorra : emaitza partzialak eta amaierakoak gordetzen ditu

ŷ N aldagaia : sarrerako datuak gordetzen ditu

- Kode inperatiboaren egitura:

ŷ Oinarrizko adierazpenen eraikuntza mapaketa bat da

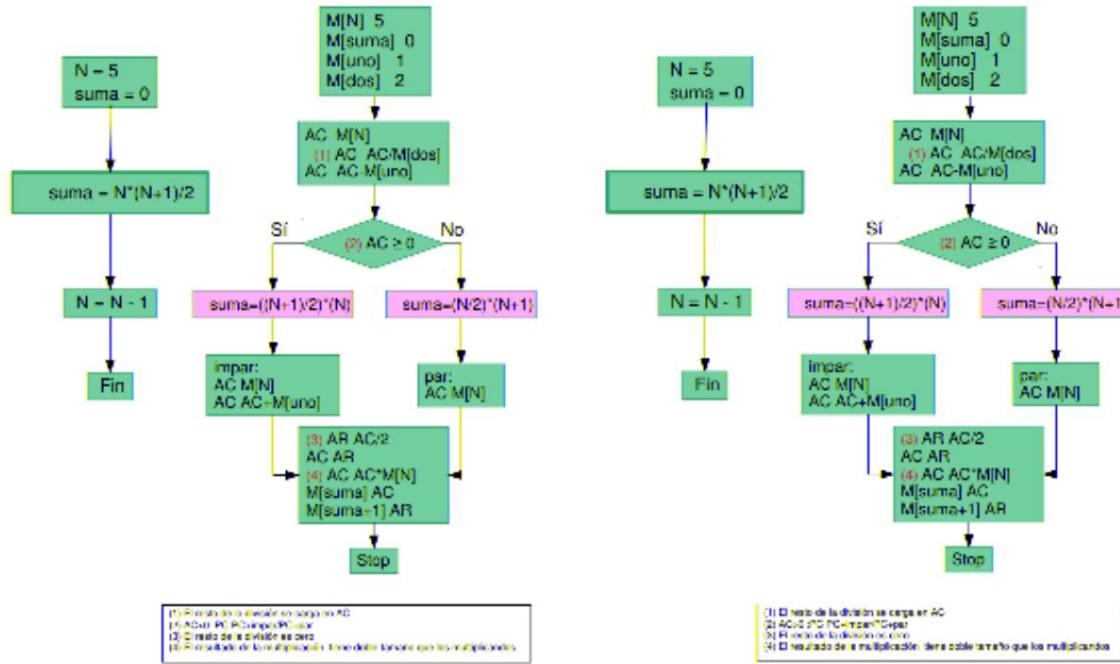
$$\text{sum} = N(N + 1) / 2$$

Organigramak: Goi Mailako eta RTL

- Algoritmoaren deskribapen grafikoa:

ŷ Goi Maila: hizkuntza naturala ezinbestekoa

ŷ RTL: ordenagailuaren ISA kontuan hartzen duen maila baxuko lengoaia



35. Irudia. Fluxu-diagrama: Pseudokodea eta RTL

IAS batzar hizkuntza

- sum1toN_mul.ias

; Lehenengo N zenbaki osoen batura. $Y=N(N+1)/2$

```

; CPU
IAS ; montaia-lengoaia: IASSim emuladorea ;
William Stallings-en, Structure of Computers liburuko 2.1 ariketa

; JARRAIBIDEEN ATALA; N
berdina al da? -> N/2 S(x)->Ac+
n hondarra      :01n      ;AC      <- M[n]
.
.
;
1. kasua: N bikoitia
.
.
;
2. kasua: Ez bakoitia
.
.
;
Biderketa N(N+1)/2
.

;
DATU ATALA ;
Aldagaien deklarazioa eta hasieratzea .data
    0 ;emaitza eta:

;
Konstanteen Adierazpena .data
n:      5 ;parametroa N bat: .data 1

bi: .datuak 2

```

simulazioa

- simulazioa IASSim emuladorearekin

7.1.3. 3. adibidea: Bektoreak

adierazpena

- Egin batura $C = A + B$ 10 elementutako A eta B bi bektoreren batura, biak hasieratuta. 1etik 10era bitarteko balioekin.

Bektore baten elementu bakoitzean sartzeko beharrezkoa da handitu
 Eragigaiaren memoria-helbide absolutua matrizeko elementuetara sartzen den instrukzioan, beraz,
 beharrezkoa da instrukzioaren eragigaien eremua aldatzea. Eragiketa eremuko 12 bitak AC erregistroko
 12 bit esanguratsuenetara transferitzeko instrukzio bat dago, hau da, AC(28:39) y M[operando](8:19).

Eta alderantzizko transferentzia egiten duen beste instrukzio bat M[operandoa (8:19)] y AC(28:39). Modu honetan, eragiketa aritmetikoak eta logikoak egin daitezke instrukzio baten operando-eremuko 12 bitetan.

- Algoritmoaren pseudokodea

- Algoritmoaren fluxu-diagrama
- Programa RTL hizkuntzan ÿ Egoki komentatu programa (goiburua meta informazioarekin, egitura-atalak, bloke funtzionalak).
- Programa muntatzailea: komeni da programa zuzenean ez egitea bere osotasunean, faseka baizik, bertsio simple batetik hasi eta azken bertsioan programa osatu arte. Adibidez:
 - ÿ 1. bertsioa : $A[i]=i$ bektorea hasieratu
 - ÿ 2. bertsioa: $A[i]=i$, $B[i]=i$, $C[i]=i$ bektoreak hasieratu
 - ÿ 3. bertsioa: $C[i]=A[i]+B[i]$
- ÿ Aldagai posibileak: len: luzera bektoriala, A0: A bektorearen lehen elementuaren helbidea, i: indize bektoriala, etab.

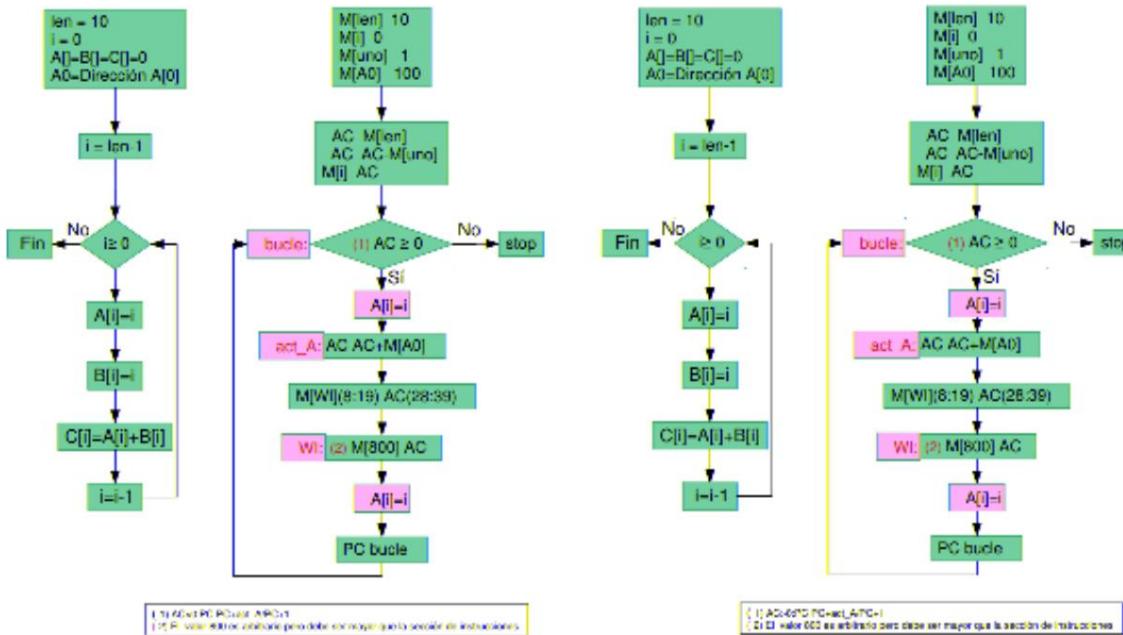
- Programa exekutatu urratsez urrats programaren bertsio desberdinak arazketan.

pseudokodea

- Algoritmoaren deskribapena NATURAL hizkuntzan testu-moduko esamoldeak erabiliz.
- ALDAGAIAK:
 - ÿ A,B,C aldagai bektorialak : Adierazi eta hasieratu $A[i]=i$, $B[i]=i$, $C[i]=0$
 - ÿ len aldakorra : bektoreen tamaina gordetzen du
 - ÿ A0 aldagaia — A bektorearen lehen elementuaren helbidea gordetzen du
 - ÿ i aldagaia : edozein bektoreren i posizioan dagoen elementuaren indizea
- Kode inperatiboaren egitura:
 - ÿ Oinarrizko adierazpenen eraikuntza begizta bat da
 - ÿ Begiztak atzerako iterazioak 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 egunерatzen da
 - ÿ Begiztatik irteten da $i=-1$ denean

Organigramak (1. bertsioa): Goi Mailako eta RTL

- Algoritmoaren deskribapen grafikoa:
 - ÿ Goi Maila: hizkuntza naturala ezinbesteko
 - ÿ RTL: ordenagailuaren ISA kontuan hartzen duen maila baxuko lengoaia



36. 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

```
; hasiera_bektore_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 of eremuan idatziz egiten da
irakurtzeko/idazteko instrukzioaren helbideak.
; Ezkerreko argibideek bakarrik izan ditzakete etiketak, beraz
Cu'->S(x) instrukzioak etiketa erabili beharko da [Jauzi instrukziora
eskuineko etiketaren posizioan] eta Cu'->S(x) etiketa [Jauzi instrukziora
etiketaren posizioan utzita] etiketa guztiak LERROKATZEKO
utzi argibideak.
; Beharrezkoa da jakin behar da helbideen begiztetan, gehitzen eta C-n dauden argibideak
memoria hitzaren ezkerrera edo eskuinera.
; Argibide kopurua bikoitia izan behar da. Erabili .hutsa kasu bakoitietan.
; azenturik ez iruzkinetan
; Laguntza linean: erreferentzia eskuliburua -> datu motak
; Ikusi -> Hobespenak -> Selectron Memory Capacity
```

;;;;;;JARRAIBIDEEN ATALA

```
;;;;;; Hasieratu indizea i = len - 1
eskuin1: Cu'->S(x) eskuin1 ; eskuinetik eskuinetik salto egin1
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 egunерatzen dut
; egunерatu erakuslea A[i]
oraingo_A: S(x)->Ac+ zero ;
S(x)->Ah+ A0 ;hasi erakuslea A[0]-rekin
S(x)->Ah+ i ;erakuslea A[0]+i-rekin hasieratu
Ap->S(x) wa ; LEFT instrukzioaren helbide eremua egunерatzen dut
"wa"-n kokatuta M[wa](8:19) <- AC (28:39)
; egunерatu A[i]=i
S(x)->Ac+ i ;
Cu->S(x) wa wa: ; jauzi wa-ren ezkerrera
At->S(x) 100 ;M[100]<-AC. 100 helbidea aldatzen da
```

execuzioa.

```
;;;;;;;; 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 ;  
zero:          1 .datuak 0
```

Simulazioa (1. bertsioa)

- Kode bitarraren execuzioa egin IASsim emuladorearekin.

IAS batzar hizkuntza (2. bertsioa)

- Vector_iniciar_A_B_C.ias programaren iturburu kodea garatu:

Simulazioa (2. bertsioa)

- Kode bitarraren execuzioa egin IASsim emuladorearekin.

IAS batzar hizkuntza (3. bertsioa)

- VectorA+B.ias programaren iturburu kodea garatu:

Simulazioa (3. bertsioa)

- Kode bitarraren execuzioa egin IASsim emuladorearekin.

8. kapitulua. IASSim simulagailua

8.1. Java makina birtuala JVM

- Instalatu [Java Garapen Kita \(Java Development Kit-JDK\)](http://openjdk.java.net/) [<http://openjdk.java.net/>] sisteman ubuntu
 - ÿ [openjdk-11-jdk](https://packages.ubuntu.com/bionic/openjdk-11-jdk) [<https://packages.ubuntu.com/bionic/openjdk-11-jdk>] linux/GNU banaketan ubuntu 18.0 bionic.
 - ÿ Egiaztatu paketerako sarbidea duzula: `apt-cache search openjdk-11-jdk`
 - ÿ Instalatu paketea: `sudo apt-get install openjdk-11-jdk`
 - ÿ Egiaztatu paketea instalatuta dagoela: `dpkg -l openjdk-11-jdk`
 - ÿ Egiaztatu instalatutako javaren 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 eraikitzea, modu mistoa)

IASSim Java JVMn exekutatzen da, beraz, beharrezko da JVM instalatuta edukitzea. Makina bat birtualizazioa edozein Sistema Eragileren gainean (Linux, MacOS, Windows) SW geruza bat instalatzean datza, horrela birtualizazio geruzan instalatutako edozein aplikazio (adibidez IASSim) sistema eragilearen menpe egon ez dadin. horrela lortzen da.aplikazioa (adibidez IASSim) Sistema Eragile ezberdinatik independentea izatea.

8.2. IAS simulagailua

- IASSim : Von Neumann-en IAS ordenagailu bidezko simulazio-tresna erabilgarria da programa baten instrukzioen exekuzioa urratsez urrats makina kodean simulatzeko. Instrukzio-ziklo bakoitzaren amaieran Selectron memoria nagusiaren eta PUZaren erregistroen edukia ikusteko aukera ematen du.
- [IASSim Web](http://www.cs.colby.edu/djskrien/IASSim/) [<http://www.cs.colby.edu/djskrien/IASSim/>] : klik eginez biltegira konektatzen gara

IASSim simulagailuarena.

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 lutzapena duen batch fitxategian klik bikoitza egin dezakezu .

8.3. Simulazioa/Arazketa

- Simulazioaren helburuak bi dira:

- a. Instrukzio bakoitzaren exekuzioa interpretatu memoria nola aldatzen den ikusiz eta erregistroak

- b. Programaren garapenean egon daitezkeen akatsak araztea.

- <https://www.linuxvoice.com/john-von-neumann/>

- Zenbaki osoen kodeketa hamaseimala eta haien bihurketa ezagutzea beharrezkoa da kode bitarra.

- Emuladorearekin berarekin datorren **Demo tutorial.ias** programari *sum1toN.ias* deituko diogu

1. Deskargatutako zip fitxategia deskonprimituta egon behar da: begiratu ateratako fitxategiak, horietako bat emuladorea irekitzeko argibideak dira.

2. Ireki emuladorea:

ÿ In	Linux	bidez	du	agindua	urtean	Ierroa:	Java	-cp
IASSim2.0.4.jar:jhall.jar:IASSimHelp2.0.jar iassim.Main -m IAS.cpu								

ÿ Leihotan: Egin klik bikoitza *.bat lutzapena duen batch fitxategian

3. Lagunza: **Lagunza** ÿ IASSim Lagunza orokorra ÿ Muntaketa-lengoaia ÿ Sintaxia eta argibide arruntak : muntaia hizkuntzaren eskuliburua

4. Ezabatu barneko zein kanpoko memoriaren edukia. **Exekutatu** ÿ Garbitu guztiak

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 hamaseitar, hamartar, bitar... Memoriaren zabalera: 20 edo 40 bit.

8. Hautatu mihiztadura hizkuntzaren iturburu-kodea duen leihoa.

9. Muntatu eta kargatu modulu exekutagarria memorian: **Execute** ÿ **Muntatu & Kargatu**

10. Memoria-mapa aztertzea: instrukzio-atala eta datu-atala

11. Aktibatu arazketa modua: **Exekutatu** ÿ **Arazte modua**

12. Instrukzio bakoitzaren exekuzioa urratsez urrats : **Urratsez urrats**

- Memoria Edukia

ŷ Lehenengo instrukzioa memoria-kokapenaren ezkerreko 20 bitetan gordetzen da eta eskuineko bigarren adierazpena.

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	

37. Irudia IAS Makina Kodea

- Erregistroen edukia:

Registers		
Base: Hexadecimal		
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

38. Irudia IAS Erregistroak

- Ariketa:

- ŷ Instrukzio bakoitzaren exekuzioa baino lehen interpreta ezazu: interpretatu instrukzioa in makina-lengoaia.
- ŷ memoriaren datuen atalaren eduki berria aurreikustea
- ŷ CPU erregistroen eduki berria aurreikustea.
- ŷ exekutatuko den hurrengo instrukzioa aurreikusi
- ŷ Programaren organigrama ondorioztatu.

9. kapitulua. Mihiztadura-hizkuntzak

9.1. Intel x86 / AMD 64

9.1.1. Kaixo Mundua

- Konputagailuen Arkitektura bakoitzak bere mutuaia-lengoia du.
- Iturburu-modulua hello_world.s mutuaia hizkuntzan.

ÿ x86-64

```
### -----
###      kaixo_x86-64_att.s
###
###      Programak garatzeko hasierako programa simplea
Muntatzailea x86-64 AT&T.
###
###      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 kontrolatzaleak:     gcc -nostartfiles hello_intel_gas.s -o
kaixo_intel_gas
###
###      eguneratua: 2015eko OTSAILA -- Linuxs x86_64 ingurunerako
###
### -----
.att_sintaxia

ÿ ## Sartu fitxategia makroekin
.sartu "macros_x86-64_gas.h"

ÿ ## Kanpoko simboloen aitorpena
.global _start # sarrera-puntuak ikusgai

ÿ ## Memoria-erreserba datu aldakorretarako
.atala          .datuak

msg0:    .ascii "Kaixo mundua\n"
len0:    .quad -msg0          #tamaina msg0 katearen bytetan

ÿ ## Batzar Hizkuntza Jarraibideen Kodearen Atala
.atala .testua

_hasi:
```

ÿ ## Programaren gonbita: inprimatu mezua

ÿ ## Deitu nukleora pantailara sartzeko eta inprimatzeko.

```
mov    $SYS_WRITE, %rax # zerbitzuaren ID-zenbakia
mov    $STDOUT_ID, %rdi # gailuaren ID-zenbakia
mov    rdi $msg0, %rsi # mezu helbidea
mov    len0, %rdx # mezuanen luzera
syscall
```

ÿ ## amaitu programa hau

```
mov    $SYS_EXIT, %eax # zerbitzuaren ID-zenbakia
mov    $0, %rdi # konfigurazio irteera-kodea
syscall # sartu nukleoan
.amaiera # gehiago ez muntatzeko
```

Makroak macros_x86-64_gas.h fitxategian

Sistema-deiak

.equ STDIN_ID, 0	# sarrera-gailu (teklatua)
.equ STDOUT_ID, 1	# irteera-gailu (pantaila)
.equ SYS_READ, 0	# "irakurtzeko" ID- zenbakia
.equ SYS_WRITE, 1	# " idatzi "ren ID zenbakia
.equ .equ .equ SYS_OPEN, 2	# " ireki "-ren ID-zenbakia
.equ SYS_CLOSE, 3	# Ixteko ID zenbakia
.equ SYS_EXIT, 60	# Irteerarako ID- zenbakia

9.1.2. Muntatzaileen Programazioa

- [Felix Cloutier](http://www.felixcloutier.com/x86/) [<http://www.felixcloutier.com/x86/>]
- [kluge](http://kluge.in-chemnitz.de/docs/notes/assembly.php) [<http://kluge.in-chemnitz.de/docs/notes/assembly.php>]
- [AI64](http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html) [<http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html>]
- [AMD64](http://developer.amd.com/resources/documentation-articles/developer-guides-manuals/) [<http://developer.amd.com/resources/documentation-articles/developer-guides-manuals/>]

ÿ AMD64 Arkitektura Programatzailearen eskuliburua 3. liburukia: helburu orokorra eta

[http://developer.amd.com/wordpress/media/2008/10/24594_APM_v3.pdf] Sistemaren argibideak

9.1.3. Zenbaki errealek

- [kluge](http://kluge.in-chemnitz.de/docs/notes/assembly.php) [<http://kluge.in-chemnitz.de/docs/notes/assembly.php>]
- ÿ Zenbaki errealekin egindako eragiketen adibide interesgarriak

9.1.4. Eztabaida zergatik ASM AT&T

- <http://es.tldp.org/Presentations/200002hispalinux/conf-28/28.ps.gz>

9.1.5. Denetarikoa

Datu mota

- Datu mota:

ÿ Memoriaren helbidea edo memoria erreferentzia: etiketaren luzera

ÿ `mov $length,%edx` ÿ `mov 0x8049fff,%edx` ÿ maila handiko hizkuntzan hasieratzea da
erakusle batena

ÿ zenbaki oso sinadura: 2-ren osagarri formatua.

ÿ `mugitu $0x4,%eax`

ÿ 0x4 eragia instrukzioan bertan dago, eragaien eremuan. 0x4 datuak "little endian"-en gordetzen dira ÿ Operando
eremua: hitz bikoitza: 32 bit 0x00000004 ÿ Goranzko memorian: 8048191 helbidea: 04 00 00 00

ÿ karakterea: ASCII kodeketa

ÿ `08049ff4 <mezua>`: 48 6f 6c 61 20 ÿ Kaixo SP

Irakaskuntza Zikloa

- PUZaren esku-hartza **4001a4 instrukzioan: 48 83 ec 10 ÿ subq \$16,%rsp**

ÿ PUZa instrukzio-zikloan zehar (harrapatze-fasea deskodetzea-exekutatu fasea)
zereginen sekuentzia bat egitea.

ÿ PUZak instrukzio-zikloan egin beharreko atazen sekuentzia atalean deskribatzen da
RTL hizkuntza.

ÿ MBR ÿM[0x4001a4]

ÿ IR ÿMBR

ÿ AC ÿ RSP

ÿ AC ÿ AC-16 ; (ALU geratzen da)

ÿ RSP ÿ AC

ÿ CP ÿ CP+1

ÿ ITSASOA ÿ PC

kaixo_mundua.s

- Konpilatu `hello_world.s` mihiztadura-lengoaiaren programa eta bota objektu-modulua
bitarra.

ÿ Iturburu-modulua: `hello_world.s`.

ÿ Makina kodea- Muntatzaire kodea

ŷ Datuen atala

08049ff4 <mezua>: 48 6f 6c **61 20** 4d **75 63 64** 6f 0a egin /n

Kaixo SP mun

08049fff <luzera>: 0b **00**

ŷ Maila handiko hizkuntza batean aldagaien deklarazioa eta hasieratzea izango litzateke.

ŷ Etiketa: memoria erreferentzia

ŷ Karaktere bakoitzak byte bat hartzen du (ASCII kodeketa). Ez interpretatu katea a gisa guztiak (ez little endian) osoak eta zenbaki errealak ez bezala.

ŷ Luzera etiketak aipatzen dituen datuak *little endian* formatuan daude ŷ 00 0b

ŷ Argibideen atala

08048190 <_hasi>:

8048190: b8 04 00 00 00 8048195:	mov	\$ 0x4, %ax
bb 01 00 00 00 804819a: b9 f4 9f	mov	\$0x1,%ebx
04 08	mov	\$0x8049ff4,%ecx
804819f: 8b 15 fr 9f 04 08	mov	0x8049fff,%edx
80481a5: cd 80	int	\$0x80
80481a7: b8 01 00 00 00 80481ac:	mov	\$ 0x1, %ax
bb 00 00 00 00	mov	\$0x0,%ebx
80481b1: cd 80	int	\$0x80

9.2. Motorola 68000

9.2.1. Kaixo Mundua

;CISC Sharp **X68000** (Human68K): Motorola **68000**

```
ilarra (katea)      ; push string helbidea pilara
dc.w $FF09          ; deitu DOS "inprimatu" salbuespen bat eraginez
addq.l #4,a7        ; leheneratu pila erakuslea

dc.w $FF00          ; deitu DOS "irten"
```

katea:

```
dc.b "Kaixo, mundua!",13,10,0
```

9.2.2. DA

- Erreferentziak

ŷ Oinarritzko argibide multzoa [http://lux.dmcsp.pl/pn/assembler_68000/asm.html]

ŷ Wikibook [https://en.wikibooks.org/wiki/68000_Assembly#Zeharkako_helbideratzea_postincrementoarekin]

ŷ Erreferentzia eskuliburua [https://www.nxp.com/files-static/archives/doc/ref_manual/M68000PRM.pdf]

ÿ Motorola 68K edo M68000 [http://www.freescale.com/files/archives/doc/ref_manual/M68000PRM.pdf]

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 makina-argibide 12
helbideratze-modu 9 instrukzio-formatu ezberdin

hitz batetik bostera bitarteko tamainak Datu-busaren zabalera: 16 bit edo 32 bit
Gutxieneko helbiderako 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 Di : izen-estate zuzena

ÿ (Ai): zeharkako erregistroa

ÿ +(Ai)—erregistratu zeharkakotasuna osteko gehikuntzarekin eragiketa-tamainaren eskalarekin (1,2,4
byteak)

ÿ (Ai)+—Erregistratu zeharkakotasuna osteko gehikuntzarekin eragiketa-tamainaren eskalarekin

ÿ -(Ai)—erregistratu zeharkakotasuna, operando-tamainaren eskalaren aurreko dekretuarekin

ÿ (Ai)—Zeharkako erregistroa aurreinkrementarekin eragiketa-tamainaren eskalarekin

ÿ D(Ai): zeharkakoa erregistratu D desplazamenduarekin

ÿ D(Ai,Ri,X): zeharkako erregistroa Ai indexatua D desplazamenduarekin Ri

ÿ D(PC): D desplazamendua duen PCarekiko

ÿ D(PC,Ri,X) — D offset duen Ri indexatutako PCarekiko erlatiboa

- Datuak

ÿ 2-ren osagarri osoak .

ÿ Atzizkiak Eragiketa: B byte (1 byte), W hitza (2 byte), L luzea (4 byte)

ÿ Datu-aurrizkiak: \$ hamaseitarra

- Memoria

ÿ Big Endian: LSB helbiderik altuenean eta MSB helbide baxuenean

9.3. MIPS

9.3.1. DA

- 32 biteko arkitektura duen prozesadorea
- Interlocked Pipeline Stages (MIPS) Arkitekturak gabeko mikroprozesadorea
- MIPS arkitektura bertsioak:
 - ŷ MIPS I (R2000 CPU), II (R6000), III (R4000), IV (R8000, R5000, R10000) eta V (inoiz ez ezarri);
 - ŷ MIPS32/64: MIPS32 MIPS II-n oinarritzen da, MIPS III, MIPS IV, MIPS III, MIPS IV eta beste ezaugarri batzuk dituena. eta MIPS V; MIPS64 MIPS V-en oinarritzen da

70 makinaren jarraibide Lau
 taldean sailkatutako argibideak Datuen mugimendua
 Zenbaki osoak, logikoak eta desplazamendu-aritmetika
 Fluxu-kontrola Koma mugikorreko aritmetika 4 helbideratze-
 modu Berehala Zuzeneko erregistroak Zeharkakoa
 desplazamenduarekin Zeharkakoa desplazamenduarekin
 64 erregistroko PC Bankuarekin (32 bit bakoitzak) 32
 helburu orokorra (R0-R31) 32 koma mugikorreko
 instrukzioetarako (F0-F31). Honela erabil daitezke: 32
 erregistro zehaztasun bakarreko eragiketetarako (32 bit)
 16 erregistro zehaztasun bikoitzeko eragiketetarako (64 bit) 3
 instrukzio-formatu ezberdin 32 biteko luzera bakarrarekin: Op
 Code: 6 bit R :hiru erregistro, a shift zenbatekoa eremua, eta
 funtzio eremu bat; I :bi erregistro eta 16 biteko berehalako balioa J :26 biteko salto-helburua
 Erregistro-erregistro arkitektura LOAD eta STORE instrukzioek bakarrik egiten dute erreferentzia
 memoria Gainontzeko instrukzioek erregistroetan funtzionatzen dute Hiru eragiketa dituzten
 instrukzioak: 2 op.source eta 1 op.Patua

Mihiztatzailearen notazioa: op x, y, z $x \leftarrow (y)op(z)$
 Datuak:

2-ren osagarri osoak: byte (1B), hitz erdia (2B), hitza (4B)
 Zenbaki errealak: IEEE-754 doitasun bakarra eta bikoitza

- [MIPS arkitektura](https://en.wikipedia.org/wiki/MIPS_architecture) [https://en.wikipedia.org/wiki/MIPS_architecture]
- [ISA MIPSen bertsioak](https://en.wikipedia.org/wiki/List_of_MIPS_architecture_processors) [https://en.wikipedia.org/wiki/List_of_MIPS_architecture_processors]
- [MIPS arkitektura duten prozesadoreak](https://en.wikipedia.org/wiki/MIPS_architecture_processors) [https://en.wikipedia.org/wiki/MIPS_architecture_processors]:

R2000 etab.

- [tutorial azkarra](http://logos.cs.uic.edu/366/notes/mips%20quick%20tutorial.htm) [http://logos.cs.uic.edu/366/notes/mips%20quick%20tutorial.htm]
- [MIPS Online emuladorea](https://rivoire.cs.sonoma.edu/cs351/wemips/) [https://rivoire.cs.sonoma.edu/cs351/wemips/]

9.4. MRA

9.4.1. Kaixo Mundua

```
/*
```

AT&T ARM prozesadorerako mutaia-lengoaia programa

Iturburu-programa: hello_world.s

Muntatzailea: Lokatzailera: linux-gnueabi-solo-hello_world.hello_world.o

```
*/
```

.datuak

msg: .ascii

"Kaixo, ARM Mundua!\n" len =

· - mezua

.testua

.globl _start _start: /

* idatzi syscall */

mov %r0, \$1 ldr %r1, =msg

ldr %r2, =len mov %r7, \$4

swi \$0

ÿ /* irten syscall */ mov

%r0, \$0 mov %r7, \$1 swi

\$0

9.4.2. DA

- [MRA](http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc_subset.architecture.reference/index.html) [http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc_subset.architecture.reference/index.html]: RISC makina aurreratua

ÿ Garatzailea _

Gidak

[http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc_subset.architecture.reference/index.html]

10. kapitulua. Programazio lengoaiak batuketa1toN

10.1. Sum1toN-rako beste hizkuntza batzuk

- Erref
 - ÿ <http://wiki.c2.com/?ArraySumInManyProgrammingLanguages>
 - ÿ 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 nul))
(aplikatu (erantsi array nil))
```

- Python

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

- Java

```
/* Iturburu-programa: sum1toN.java

konpilatu: javac sum1toN.java -> sortu BYTECODE sum1toN.class
exekutatu -> java -cp . batuketa1toN ; bytecode *.class behar du eta exekutatu egingo du
klaseko nagusia

*/
```

```
klase publikoa sum1toN {
  // metodo nagusia klaseko klasean kapsulatua, estatikoa, nagusia aldatu ezin dadin
  atributuak, publikoak 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 );
      Sistema.out.print(" ");
      batura=batura+x;
      x--;
    }
    Sistema.out.print("\n");
    Sistema.out.print("sum="+sum);
```

```

        System.out.print("\n");
    }
}

```

- C

```

/*
Programa: batuketa1toN.c
Deskribapena: 1,2,3,...N serieen batura egiten du
C lengoain dagoen programa da sum1toN.ias-en baliokidea
von Neumann IAS makina
Hizkuntza: C99
Deskribapena: Lehenengo 5 zenbaki naturalen batura
Sarrera: Irteera: Aldagai batean definitua
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
Liburudenda: /usr/lib/x86_64-linux-gnu/libc.so
CPUak: Intel(R) Core(TM) i5-6300U CPU @ 3.0GHz
Konpilatzalea: gcc 6.3 bertsioa
Muntatzailea: GNU muntatzailea 2.28 bertsioa
Lokatzailea/Kargatzalea: GNU Id (GNU Binutils Ubunturako) 2.28
Gaia: Data: Egilea: Konputagailuen Egitura
2017/09/20
Candido Aramburu
*/

```

#include <stdio.h> // printf() funtzioaren liburutegiko goiburua

```

// programan sartu funtzioa
void main (huts)
{
    // Tokiko aldagaien adierazpena
    kararen batura=0;
    charn =0b101;
    // begizta
    bitartean(n>0){
        batura+=n;
        n--;
    }
    printf("\n batura = %d \n",batura);
}

```

- INTEL mutaria-lengoia eta nasm mihizagailua

```

::: Programa: sum1toN.asm
::: Deskribapena: 1,2,3,...N serieen batura egiten du

```

```
;; Intel hizkuntza
;; NASM muntatzailea

;; nasm -hf -> f aukeraren lagunza
;; Asanblada nasm -g -f elf sum1toN.asm -o sum1toN.o
;; estekatzalea -> ld -m elf_i386           -o sum1toN sum1toN.o
```

ÿ **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 aldagaia implemenatzen du
mov edx,[n] ; EDX implements n aldagaiaaren alias bat da

begizta:

gehitu ecx,edx
azpiedx ,1
jnz begizta

mov ebx, ecx ; irteerako argumentua OSra EBX bidez
akordioa

;; irten
mugitu **eax ,1** ; sistema eragilearen dei-kodea: irten azpierrutina
int 0x80 ; sistema eragilearen deia

- ATT mutaia-lengoaia x86-32 arkitekturarako

```
### 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
azpi $1,% edx
jnz begizta
```

akordioa

```
## irten
mov $1, %eax # OS dei - kodea : irten azpierrutina
int $0x80 # sistema eragilera deitu
```

.amaiera

- AT&T x86-64 arkitekturako mihiztadura-lengoaia

ÿ ##### Programa: sum1toN.s

ÿ ### Deskribapena: 1,2,3,...N serieen batura egiten du . Sarrera definituta dago programan bertan eta irteera OSra pasatzen da

ÿ ### Hizkuntza : AMD64 arkitekturako GNU muntaia-lengoaia

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 aldaagaia ezartzen du
movq n,%rdx
```

begizta:

```
gehitu %rdx,%rdi
azpian $1,%rdx
jnz begizta
```

RDI bidez OSra irteera argumentua ABI AMD64 konbentzioaren arabera

ÿ ## irten

```
mov $60, %rax # OS dei - kodea : irten azpierrutina
```

```
syscall # dei sistema eragileari , azpierrutina exekutatu dezan
RAX balioa
```

.amaiera

- MRA

```
/* Programa: sum1toN.s
```

Deskribapena: 1,2,3,...N serieen batura egiten du

IAS makinaren sum1toN.ias -en baliokidea den ARM hizkuntza-programa da
von Neumannen eskuistik

```
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 @ R0 batura aldagaia implementatzen du
```

```
ldr r2,=n ldr @ R1-k n aldagaia implementatzen du zeharka
```

r1,[r2]

/* Zuzeneko helbidera :

mov r1,n huts egiten du mov-ek ez duelako zuzeneko memoria-helbideratzea onartzan.

mov -ek berehalako helbideratzea onartzan du 32 biteko literalak ez badu

zeroen errepikapena du ezkerrera eta eskuinera

8 biteko literal bihurtzeko eta ondoren

joan-itorriak

ldr r1,n Errorea: barne_lokalizazioa (OFFSET_IMM mota) ez dago osatuta

Errore bat 32ko literalak (n helbidea) kodetzen saiatzean

bitsak.

```
*/
```

begizta:

gehitu r0,r1

azpiak r1,#1

bne begizta

@r0 OSrako irteera argumentua da EBX bidez konbentzioz

/* irten syscall */

mov r7, #1

swi #0

.amaiera

11. kapitula. RTL Erregistroaren transferentzia hizkuntza

11.1. RTL hizkuntza

11.1.1. Sarrera

- ARGIBIDEEN 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 , Programatzaleak interpreta ditzakeen sinboloen bidez, bere deskribapena ahalbidetzen du hardware mailan jokabidea eta, horrela, makina baten arkitekturaren diseinua definitzea.
- *Erregistroak* memoriaren oinarrizko elementua dira ordenagailuaren oinarrizko unitate ezberdinaren arteko datuen eta argibideen ibilbidean. Erregistroa datuak gordetzen, memorizatzen eta gordetzen dituen zirkuitu digitala da.
- Datuen ibilbidea busek eta autobusen bidez konektatzen diren elementuek (*erregistroak*, multiplexadoreak, etengailuak, kontagailuak, etab.) osatzen dute.

ÿ Adibidea: datu baten bidea memoria-kokapen nagusi batetik ALUko eragiketa-erregistroetara.

ÿ Buffer kontzeptua: datuen bideko memoriaren tarteko etapa.

- PUZak exekutatzen dituen instrukzioak exekuzioak datuen bidez transferitzea dakin datu-bideen erregistroen.



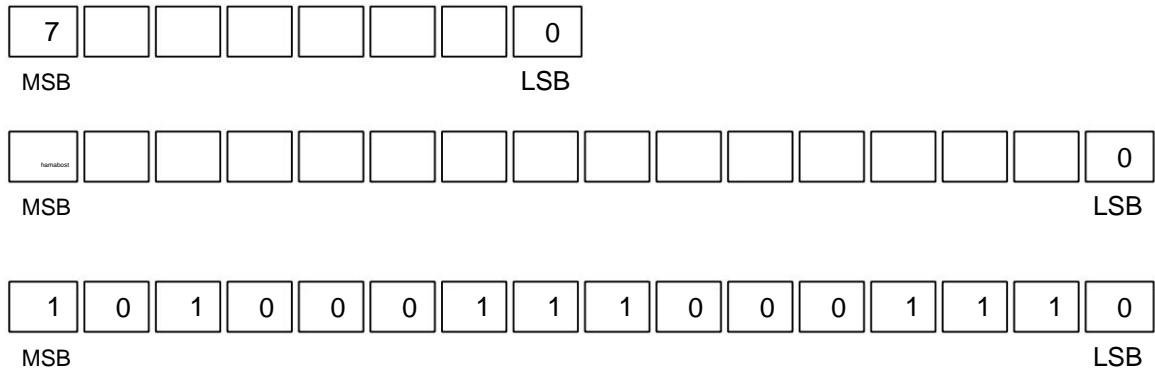
Ez nahastu RTL (Register Transfer Language) eta RTL (Register Transfer Level). Erregistro transferentzia maila HARDWARE deskribapen lengoaien erabiltzen dute (Hardware deskribapen hizkuntza HDL)

11.1.2. Erregistroak

Arkitektura

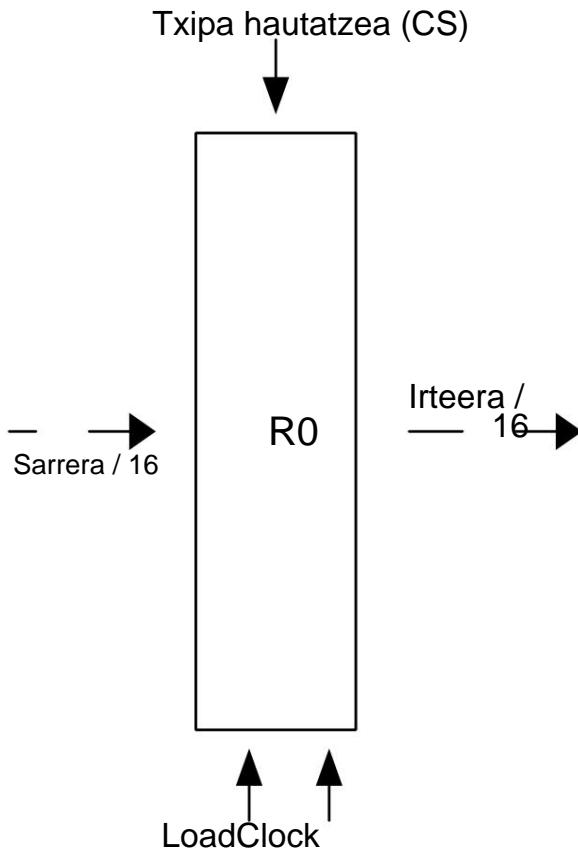
- Erregistro baten arkitekturak bere funtzionaltasuna eta ezarpenaren egitura barne hartzen ditu.
- Erregistroak:
 - ÿ bit-segida batez osatutako hitza *gorde* .
 - ÿ dimentsio bateko gelaxka-matrize bat dira, non gelaxka bakoitzak bit bat gordetzen duen.
- Bere tamaina 8 byteko multiploa izan ohi da eta izen bat jasotzen du.
 - ÿ 8 bit: 1 byte
 - ÿ 16 biteko: Word. Arrazoi historikoengatik. (gogoratu hitz baten tamaina beste batean testuingurua kasuan kasuko makinaren araberakoa da)
 - ÿ 32 biteko: hitz bikoitza
 - ÿ 64 biteko: hitz laukoa

- Gelaxkak zerotik hasita zenbatzen dira.
 - ÿ LSB: Bit esanguratsu gutxien bit esanguratsuena da
 - ÿ MSB: Bit esanguratsuena pisu handiena duen bit da

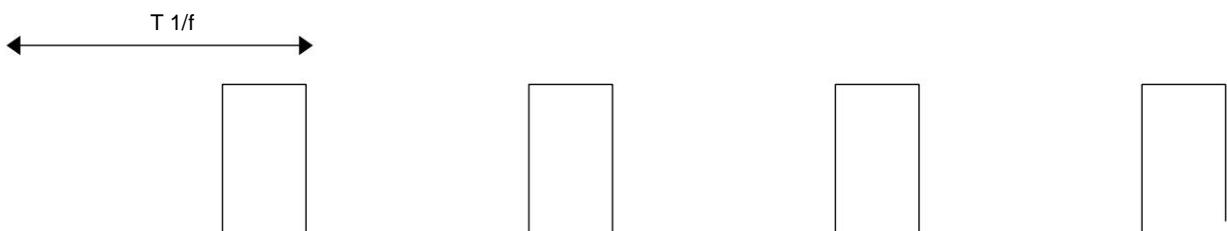


Egitura

- Erregistroaren egitura bere funtzionalitatearen ezarpena da
 - Grabatu beharreko datu-bit bakoitza memoria-ahalmena duen gelaxka batean gordetzen da. -ren zelulak erregistro bat flip-flop izeneko zirkuitu digital batekin ezartzen dira. txankleta bakoitza pixka bat gordetzen du.



- Erregistroa kanpoko mundura konektatzen da busen bidez: sarrera-busa eta irteera-busa
- CS: Txipa hautatza: R0 erregistroaren barne irteera irteerako busera konektatzen du ÿ Irakurri eragiketa erregistrotik
- Kargatu: seinalea aktibo badago, R0 erregistroko sarrera-busaren balioaren karga agintzen da, sarrerako datuak erregistratzen dira. Idatzi eragiketa erregistroan.
- Erlojua: aldizkako seinale digital bitarra.
- Karga sinkronoa da erloju-seinale CLK erlojuarekin. Sinkronismoa ertz positiboetan gertatzen da _/ edo negatiboa



Erlojuaren seinalea, erlojua.

11.1.3. sinboloak

- Erregistroaren izenak maiuskulaz 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 programaren.

11.1.4. RTL esaldiak

Eragiketak eta RTL Perpausak

- RTL hizkuntzan, enuntziatuz ulertzen dugu eragiketak egitea inplikatzen duen esamolde bat erregistroak.
- RTL eragiketak:
 - ÿ Erregistroen arteko transferentziak, bi erregistroren edukien batura, edukiak alderantziku erregistro bat, etab.

mikro eragiketa

- MIKROeragiketak: MIKROprozesadoreak barnean egiten dituen eragiketak, exekutatzean Makinaren Instrukzio bat.
 - ÿ Adibideak: erregistro batean idatzi, M.Main-era irakurtzeko ordena, erregistro batetik irakurri, instrukzio bat deskodetu, kontagailu bat gehitzea, gehitzea (mikroordenatuak batutzailearen zirkuituan), erregistro baten bitak aldatzea, ETA logika, etab.
- Erregistro batean idazteko eragiketa edo CPUrako erregistro baten irakurketa mikroeragiketa bat da.

Erregistroen arteko transferentzia

- Transferentzia operadorea ÿ
- Transferentzia adierazpena: R2ÿR1
 - ÿ R1 iturburu-erregistroa deitzen da eta R2 helmuga-erregistroa
 - ÿ Kopiatu R1 erregistroaren edukia R2 erregistroan

Baldintza perpausa

- (K1=1) bada R2ÿR1
 - ÿ K1:R2ÿR1
 - ÿ Transferentzia edo kopia K1 egia bada bakarrik egiten da, hau da, K1 balioa balio badu. logikoa 1.

Bateko Epaia

- Koma eragilea
- K3:R2;R1,R3;R1
 - ÿ K3 egia bada, R1-en edukia R2 eta R3-n kopiatzen da

Memoria Nagusiaren erreferentzia

- Kortxete eta M ikurra erabiltzen dira.
- M[0x80000] : memoria-kokapenaren edukia 0x8000
- AC ÿ M[0x80000]: kopiatu 0x8000 posizioko memoriaren edukia AC erregistratzeko.
- AC ÿ M[AC]: AC erregistroak **adierazitako** memoria-kokapenaren edukia kopiatu hemen ac erregistroa
- M[0x8000] ÿ AC: kopiatu AC erregistroko edukia Memoria kokapenean 0x8000
 - ÿ M[0x8000] ÿ R[AC]: kopiatu AC erregistroko edukia Memoria kokapenera 0x8000

Ezker-eskuin balioa

- Kontzeptu hau C hizkuntzan erabiltzen da = sententzia-esleipena definitzerakoan
- M[0x1000] ÿ M[0x2000]
 - ÿ 0x2000 posizioaren edukia 0x1000 posizioan kopiatzen da
 - ÿ ÿ operadorearen eskuinean dagoena ebaluatzen da eta BALIO bat itzultzen da
 - ÿ ÿ eragilearen ezkerrean dagoena Memoriaren HELBIDEA edo ERREFERENTZIA da (Nagusia edo Erregistroa)

11.1.5. RTL adibideak adierazpen aritmetiko-logikoekin

- AC ÿ R1 v R2
 - ÿ EDO eragiketa logikoa
- (K1+K2):R1 ÿ R2+R3,R4;R5;R6
 - ÿ + sinboloak bi esanahi ditu: boolearra edo aritmetikoa.
 - ÿ K1+k2-n esanahi boolearra du: edo. Hona hemen-ren batura aritmetikoa seinale logikoak. Zentzuzkoa da seinaleak aktibo dauden edo ez ebaluatzea.
 - ÿ R2+R3-n esanahi aritmetikoa du.
- Esapide batean lehentasuna adierazteko parentesiak erabiliko ditugu.

12. kapitulua. Argibide-formatua: ISA Intel x86-64

12.1. Argibide-formatua: ISA Intel x86-64

12.1.1. Adibidea subq \$16,%rsp

- Adibidea:

ÿ Intel x86 makinaren instrukzioa.

ÿ **4001a4: 48 83 ec 10** ÿ subq \$16,%rsp

ÿ Nola interpretatu makinaren instrukzioa **4883EC10? x86 makinarako ISA Arkitektura Erreferentzia Eskuliburua** kontsultatu eta helbideratze moduak ulertu behar dituzu.

ÿ [intel x86 edo x86-64 eskuliburu ofiziala](http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html) [<http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html>]: adi adi sintaxiari.

ÿ kontsultatu 2B liburukia (4. kapitulua, 394. orrialdea) *SUB instrukziorako*. Behar dute Eragigaien tamaina eta helbideratze moduak aurkitzen ditu.

ÿ **SUBQ** eragiketaren q atzizkiak 64 biteko eragigai bat adierazten du. **\$16** iturburuko eragigaia berehalako zuzendu daiteke erreferentzia eta 8 bitekin kodetu daiteke, eta **%RSP** helmuga eragigaia 64 biteko erregistroa da. Beraz, eskuliburuko intel deskribapena **SUB r64 izango da, imm8 REX.W + 83 /5 ib opcodeeari** dagokiona .

ÿ Intelek funtzionamendu-kodearen deskribapena ez da erraza eta beharrezkoa 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 Arkitekturak instrukzio-formatua

ÿ Instrukzio formatuak eremu hauek ditu: **REXprefix-CodOp-ModRB** hori gurean kasuak **48-83-EC** balio du

ÿ REXprefix eremuaren interpretazioa: **REX.W:** Eskuliburua ÿ REX aurizkia berehalako 64 biteko eragigaietarako eta/edo GlobalPurposeRegister(rax,rbx, etab.) erregistroetarako erabiltzen da, 2.2.1.2 REX aurizkiaren eremuei buruz gehiago

ÿ Lehenengo bytea **48** ÿ **01001000** da , non 3. posizioko bit-a aktibatuta dagoen, beraz, "2-4. Taula. REX Aurizkiaren 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. **Kasu honetan erregistratu .** Ikus behean R/M azpieremuia.

ÿ **ib :** 1 byte (ib) berehalako eragiketa.

ÿ Lehen mailako Opcode eremua: bigarren bytea **83** da ÿ **SUB** kenketa eragiketa

ÿ ModRB eremua: hirugarren byteak **EC** balio du ÿ **1110-1100** RSP erregistroari dagokio.

ŷ 2.1.3 ModR/M eta SIB byteak: eragiketa bati erreferentzia egiten dioten argibide asko. memoriak (memoria nagusia edo CPU barneko erregistroa) helbideratze-forma bat du byte zehatzalea (ModR/M deitzen zaio...). Eremu hau azpieremuetan banatzen da: **Mod Reg/Opcode-R/M**

ŷ Mod azpieremua: **11** : mod eremua r/m eremuarekin konbinatzen da 32 osatzeko balio posibleak: zortzi erregistro (rax, rbx, rcx, rdx, rsi, rdi, rsp, rbp) eta 24 helbideratze moduak

ŷ Reg/Opcode Azpieremua: Kasu honetan ez da Bigarren mailako Opcode, baizik eta Reg: **rrr= 101**

ŷ 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. **Kasu honetan 4.** kodearekin funtzionatzen duen erregistroa da. 3.1 taulan kodearen kdea **Lauko hitzen erregistroa** Reg **Field 4** balioa duen **RSP** erregistroa da

ŷ 2-5 irudia. Erregistratu-Erregistratu Helbideratza (Memoria Operandorik gabe); REX.X Ez da erabiltzen
ŷ Mod=11 ŷ Rrrr =0101 ŷ Bbbb=0100

ŷ Laugarren byteak **10** balio du hamaseimalean , hau da, berehalako 16 balioari dagokiona. hamartarrez eta 64 bitera zabaldu behar da.

12.1.2. Beste x86-32

- http://www.c-jump.com/CIS77/CPU/x86/lecture.html#X77_0010_real_encoding

ŷ GEHITU cl,al ŷ 02C8

ŷ GEHITU EAX, [ESI + disp8] ŷ 0346XX

- [jauzi-argibideak](http://en.wikibooks.org/wiki/X86_Assembly/Control_Flow#Jump_Instructions) [http://en.wikibooks.org/wiki/X86_Assembly/Control_Flow#Jump_Instructions]
- Kaixo mundua x86

08048190 <_hasi>:	
8048190: b8 04 00 00 00	mov \$ 0x4, %ax
8048195: bb 01 00 00 00	mov \$0x1,%ebx
804819a: b9 f4 9f 04 08	mov \$0x8049ff4,%ecx
804819f: 8b 15ff 9f 04 08 80481a5:	mov 0x8049fff,%edx
cd 80 80481a7 : b8 01 00 00 00	int \$0x80
	mov \$ 0x1, %ax
80481ac: bb 00 00 00 00	mov \$0x0,%ebx
80481b1: cd 80	int \$0x80

- **b8 04 00 00 00** ŷ mov \$0x4,%eax

ŷ Eskuliburua: *B8 + rd MOV reg32,imm32 2 Mugitu berehala dword erregistratzeko:*

ŷ Eragiketaren kdea eremua: **B8**

ŷ Operando eremua: hitz bikoitza: 32 bit: **04 00 00 00** ŷ little endian ŷ datuak da 0x00000004

13. kapitula. x87 FPUak

13.1. x87 FPUak

13.1.1. Laburpen

- x87 arkitektura:

ÿ 1980

ÿ zenbaki errealekin eragiketa matematiko konplexuak egiten dituen instrukzio multzoa da, hala nola tangentea kalkulatzea, etab.

- x87 koprozesadorea edo x87 FPU (Float Point Unit):

ÿ x86 CPUtik independentea den prozesadorea da, arkitekturako argibideak exekutatzeko x87.

- x87 erregistroak:

ÿ FPUsen barneko erregistroak dira. 8-mailako *pila* -egitura sakona ez zorrotza , ST (0) eta ST (7) bitarteko. Ez dira zuzenean irisgarriak, baina pilaren goiko aldera bultzatuz, lehertuz edo mugituz atzitzen dira.

- FPU : koma mugikorreko eragiketen kalkuluan espezializatutako prozesatzeko unitate zentralaren osagaia da, ALU-a RPG erregistroetan gordetako zenbaki osoekin egiten den modu berean.

- Datuen formatua:

ÿ doitasun bakarra, zehaztasun bikoitza eta 80 biteko doitasun bikoitz hedatua flotatzaile bitarra puntu aritmetika 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 mikroprozesadoreetan sartu zen.

ÿ MMX-k FPUsen 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) x86 ez-x87 arkitekturarako MMX azpimultzoko SIMD lutzapenaren instrukzio multzo bat da, Intelek seinale digitala prozesatzeko eta grafikoen prozesatzeko aplikazioetarako diseinatua.

ÿ 1999an Pentium III-rekin hasi zen.

ÿ Gehitu 128 biteko 16 erregistro berri XMM0-XMM15

ÿ XMM: SSE koma mugikorreko instrukzioek erregistro-multzo independente berri batean funtzionatzen dute (XMM erregistroak), eta MMX erregistroetan funtzionatzen duten osoko instrukzio batzuk gehitzen ditu.

ÿ SSE2 Pentium 4-n (2000).

- AVX: Iuzapen bektorial aurreratuak

ÿ Gehitu 256 biteko 16 erregistro: YMM0-YMM15

ÿ Lehen 128 biteko XMM-rekin funtzionatzen zuten argibideek orain 128 bitekin funtzionatzen dute.

YMMren pisu txikiagoa.

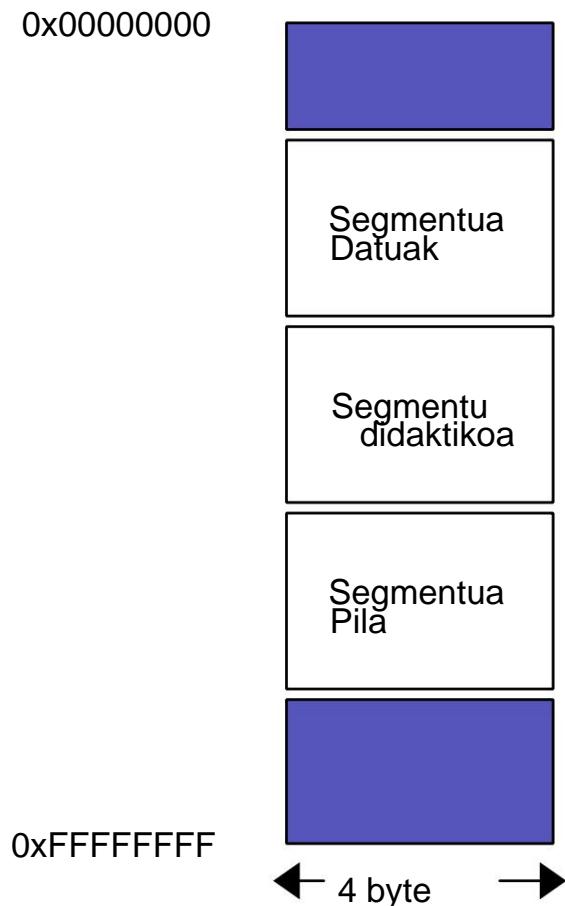
13.1.2. Erref

- [Programazioa x87 Koma Mugikorreko Unitatearekin](http://home.agh.edu.pl/~amrozek/x87.pdf) [http://home.agh.edu.pl/~amrozek/x87.pdf]: Intel Vol.1 8-1
- [Computer Systems: A Programmer's Perspective, 2/E \(CS:APP2e\)](http://csapp.cs.cmu.edu/2e/waside.html) Randal E. Bryant eta David R. O'Hallaron, Carnegie Mellon Unibertsitatea [http://csapp.cs.cmu.edu/2e/waside.html]

14. kapitula. Pila

14.1. Kontzeptua

- Pilatu edo pilatu:
 - 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:



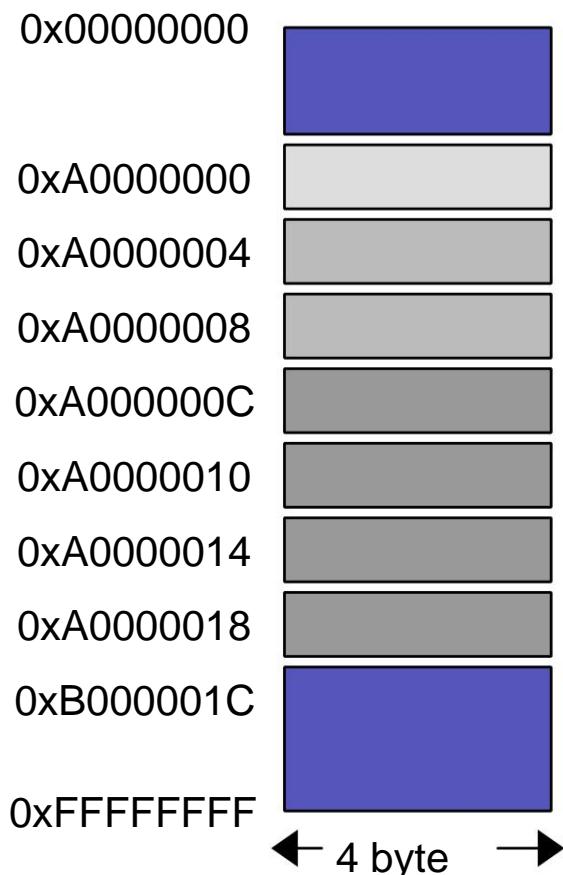
14.2. Zabalera

- Pila zabalera ÿ Hitzaren tamaina:
 - ÿ x86-64 kasuan: 64 biteko zabalera
 - ÿ i386 arkitekturan 32 bitekoa da

- Pila memoria lerrokatzea ÿ hitzen tamainaren multiploak
 - ÿ x86-64 kasuan : 8 byteko (64 bit) multiploak ÿ 0 eta 8z amaitzen diren helbide hamaseitarrak.

ÿ Pilatu beharreko datuak pilaren zabalera baino txikiagoak badira, luzatu egin beharko da. Luzapen mota datu-motaren araberakoa izango da (zenbaki osoarekin, etab.)

- i386 arkitekturaren pila-segmentua:



14.3. Markoa: marko-erakuslea eta pila-erakuslea

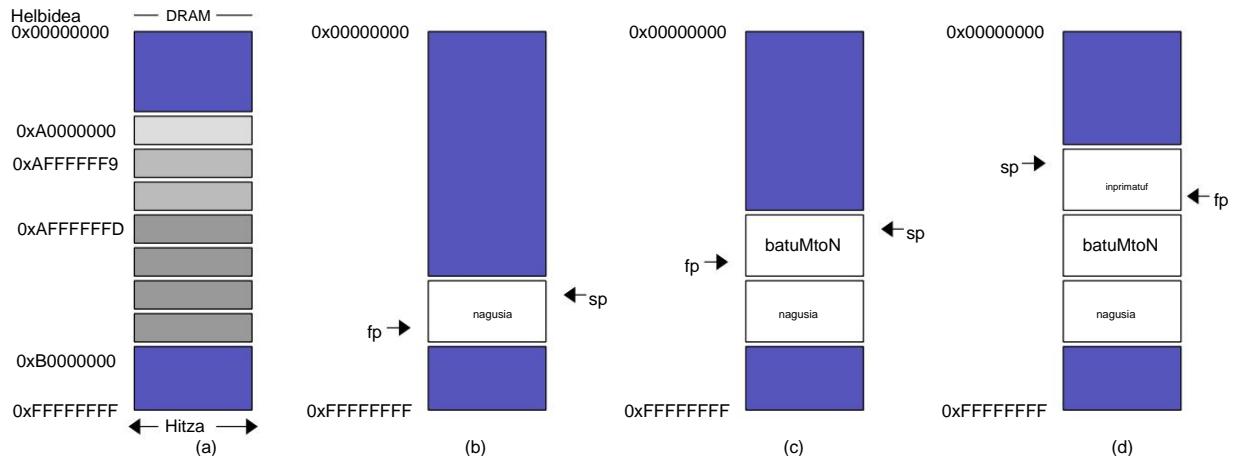
- Markoa: pila atalaren zatiketa
 - ÿ Deitzen den funtziobakoitzak fotograma bat sortzen du
 - ÿ **Marko aktiboaren** mugak bi erakuslez adierazten dira:
 - ÿ beheko muga: marko-erakuslea, pilatutako *lehen* elementuaren kokapena adierazten du .
 - ÿ goiko muga: pila-erakuslea, pilatutako *azken* elementuaren kokapena adierazten du .
- Pila erakuslea (**sp**)
 - ÿ Markoaren TOP elementura seinalatzen duen erakuslea: elementua dagoen pilaren muga alta.
 - ÿ pilatutako azken elementua.
 - ÿ Intel x86-n RSP erregistroa da

- Markoaren erakuslea (**fp**)

ÿ Erakuslea markoaren BEHEKO elementura seinalatzen duena : elementua dagoen pilaren muga baxua pilatutako lehen elementua.

ÿ Intel x86-n RBP erregistroa da

- Pila atala (markoetan banatu)



- (a) Pila ez dago osatuta
- (b) dei nagusira: marko nagusia eratzen da. Markoa hazi eta uzkurtu egiten da pilatu ahala eta ateratzen dugu
- (c) deia nagusitik sumMtoNra: sumMtoN markoa nagusitik aurrekoaren gainean eratzen da: berria FP eta SP erakusleak.
- (d) sumMtoN-tik printf-ra deia: printf markoa aurreko sumMtoN-ren gainean eratzen da: FP eta SP erakusle berriak.

ÿ Pila deiaren hitza sortzen den egitura dinamikoa da
funtzio bat eta funtzioaren hitzakarekin desagertzen da

14.4. Push-Pop muntatzailearen jarraibideak

- Push-Pop Instrukzioa: Stack-Pop

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 iturburuko eragiketa helmugan txertatzen du.

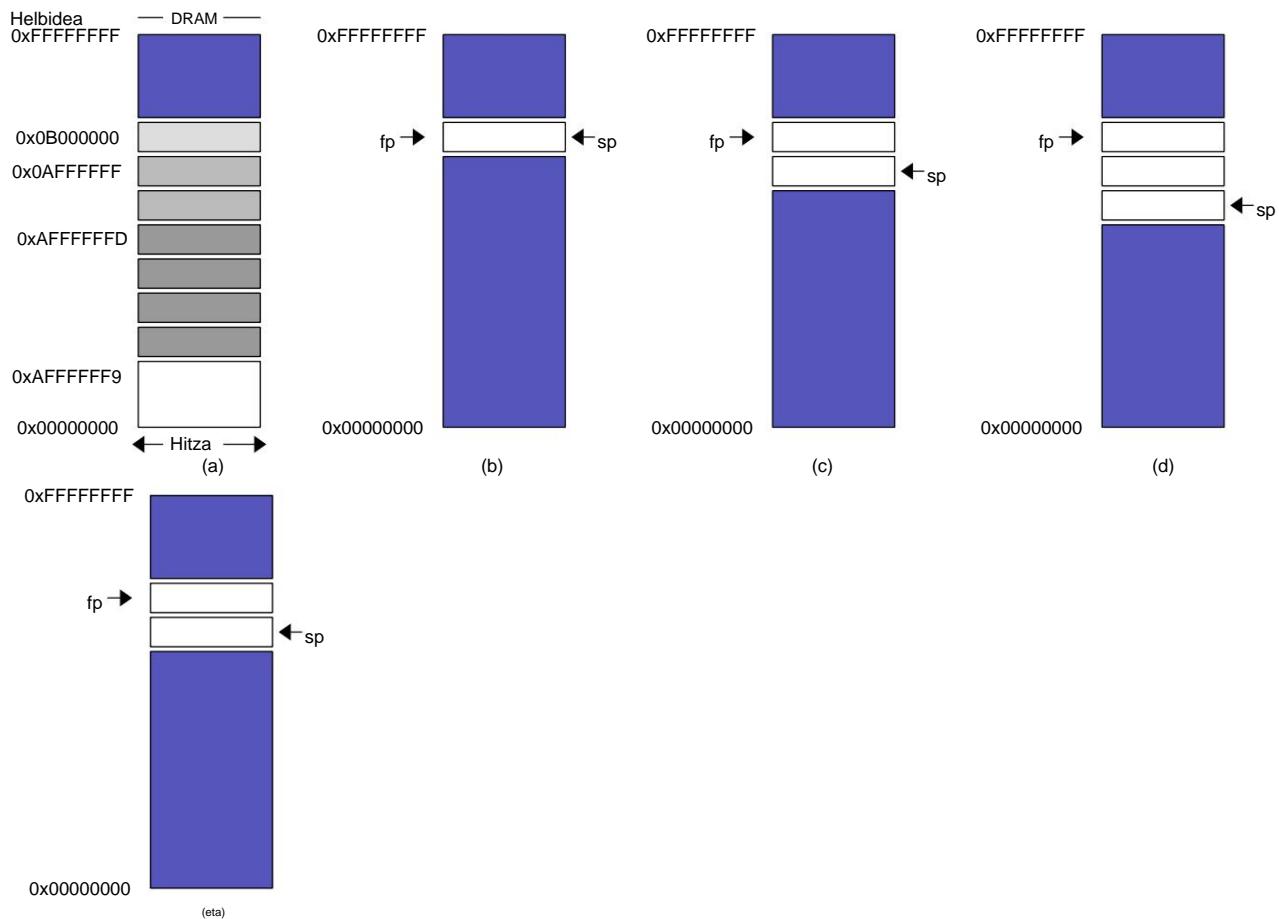
ÿ Pop Op_dest

ÿ Eragiketa: datuak atera.

ÿ Iturburu-eragilea: pilatutako azken objektua.

ÿ Lehenik pila-erakusleak erreferentziatutako objektua ateratzen da. pilaren azpian

Erakuslea hitz batez GEHITZEN DA. 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) Bultzada gauzatzea
- (d) Bultzada gauzatzea
- (e) Pop Exekuzioa

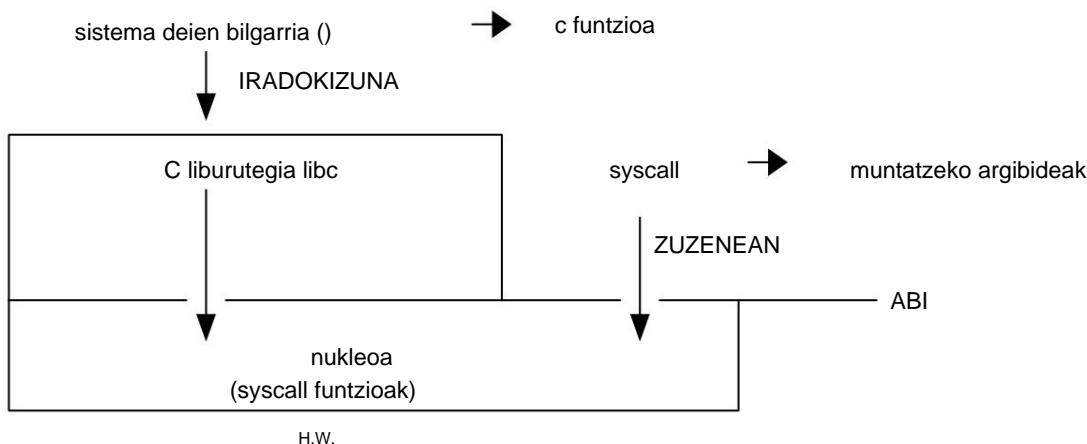
14.4.1. Dei habia egitea

- GUZTIAK

15. kapitulua. Sistema eragilearen deiak

15.1. Sarrera

- Programak egiten dituen deiak *sistema* deien izenarekin ezagutzen dira . erabiltzailea Sistema Eragilearen Kernelaren azpierrutinetara.
- Sistema eragilearen funtziotako pribilegiatuak burutzeko, hala nola, ordenagailuko saihesteko gailuetarako sarbidea, beharrezko da erabiltzailearen programek nukleoa deitzea, eragiketa seguru eta eraginkortasunez burutzen duen nukleoa izan dadin. Honek aplikazioaren programatzailea 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 kernel zerbitzuei deitzea bi modutara egin daiteke:
 - zuzena** edo **zeharkakoa**
 - ÿ Zuzena: **ASMtik syscall** instrukzioa erabiliz
 - ÿ Zeharkakoa: C edo ASMtik **libc** liburutegiko funtziok erabiliz: dei-bilgarriak zuzena
- 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
 -> nukleoaren syscall

->

ABI

15.2. 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 liburutegiaren wrapper funtzoarekin lotutakoak dira libc.

ÿ Sistema-deiaren 1. argumentua funtzoaren EZKER argumentua libc-n eta azkena ESKUINEAN.

15.3. ZEHARKAKO deia

- C: C aplikazio-programatzaleak GNU *libc* liburutegiko interfaze-funtzioak erabiltzen ditu . kernelera **zeharka** sartzeko *edukiontzien bidez* (*bilgarri*).

ÿ system calls wrapper: hizkuntzan implementatutako deien C hizkuntzara egokitzea ASM

15.4. Deitu zuzenean

- **ASM:** ASM hizkuntzan aplikazio-programatzaleak *sistema-deiak* erabiltzen ditu zuzenean kernelera **sartu**

ÿ Deia x86-64-n **syscall** muntaketa-instrukzioa eta **int 0x80** -n egiten da.
 x86-32

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)
    .equ .equ STDOUT_ID, 1 # irteera-gailu (pantaila)
    .equ      SYS_READ, 0   # "irakurtzeko" ID- zenbakia
    .equ      SYS_WRITE, 1  # " idatzi "ren ID zenbakia
    .equ      SYS_OPEN, 2   # " ireki "-ren ID-zenbakia
    .equ .equ SYS_CLOSE, 3  # Ixteko ID zenbakia
    .equ      SYS_EXIT, 60  # Irteerarako ID- zenbakia
```

15.4.1. Dei zuzeneko argudioak

- Deiaren konbentzioa ABI estandarrean deskribatzen da

- x86-64

ÿ Deialdiaren lehen 6 argumentuak erregistroetatik pasatzen dira
sekuentzia: **RDI-RSI-RDX-R10-R8-R9**

ÿ Deiaren itzulera balioa **RAX** erregistrotik pasatzen da

- x86-32

ÿ Lehenengo 6 argumentuak erregistroetatik pasatzen dira hurrengo sekuentzian:
EBX-ECX-EDX-ESI-EDI-EBP

ÿ Deiaren itzulera balioa **EAX** erregistrotik pasatzen da

- libc eskuliburua: deien argudioei buruzko informazioa

15.4.2. Zuzeneko dei-kodeak

- Dei-kodea nukleoak exekutatu beharreko funtzioarekin lotutako zenbaki oso bat da

- Dei-kodea **RAX** bidez kernelera pasatzen da

- Kodeak:

ÿ */usr/include/asm/unistd_64.h*: makro-adierazpena deiaren kodearekin
x86-64 arkitektura

ÿ irten ÿ 60, irakurri ÿ 0, idatzi ÿ 1, ireki ÿ 2, itxi ÿ 3, etab.

ÿ */usr/include/asm/unistd_32.h* : makro-adierazpena deiaren kodearekin
x86-32 arkitektura

ÿ */usr/include/bits/syscall.h* : makro zaharrak x86-32 arkitekturan ere balio dute

15.5. 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)

15.6. Adibideak: ZEHARKAKO ASM

- ASM lengoaian 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 $buffer_address_label mov-ra idatzi beharreko fitxategiaren erreferentzia
da , %rsi #memory helbidea fitxategian zer idatzi behar den mov tamainan,%rdx call write

#memoria buffer-aren tamaina #idazteko 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 gailuetan
fitxategiak dira.
mov $buffer_address_label,%rsi mov
size,%rdx call syscall
```

15.7. 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 gailuetan
```

fitxategiak dira.

```
mov $buffer_address_label,%rsi
mov tamaina,%rdx
syscall
```

15.8. Komando-lerroa

15.8.1. Prozesua

- Prozesuaren Hastapena

ÿ Komando edo programa bat shell komando-lerroan idazten dugunean, sistema eragileak kateen sekuentzia gisa interpretatzen du. Adibidez **\$sum 2 3** komando lerroko hiru argumentu dira:

ÿ Kate baten kodeketa bere karaktereen sekuentzia ASCII kodean eta
0x00 kdea duen NULL karakterearekin amaitu da

ÿ "batuketa" katea: 5 ASCII karaktere: 0x73,0x75,0x6d,0x61,0x00

ÿ "2" katea: 2 ASCII karaktere: 0x32,0x00

ÿ "3" katea: 2 ASCII karaktere: 0x33,0x00

ÿ Lerroko 3 argumentu daudenez, argumentuaren kontagailua **argc** parametroa 3ren berdina izango da.

ÿ Hiru komando-lerroko kateak, "sum"- "2"- "3", aldagai-matrizeari esleitzen zaizkio.

argv kateak

ÿ argv[0] "sum" katera seinalatzen du

ÿ argv[1] "2" katera seinalatzen du

ÿ argv[2] "3" katera seinalatzen du

ÿ argv[argc] NULL karakterea seinalatzen du

ÿ argv erakusleen array bat da, beraz (char **) argv motakoa da

- kernel

ÿ Nukleoak extern int main prototipoa deklaratzentzu (**int argc , char* argv[] , char* envp[]**) ;

ÿ modulu nagusiaren adierazpena eta definizioa **nagusia**

ÿ Funtzio **nagusia** global gisa deklaratzentzu nukleoak eta erabiltzaileak definitzen du.

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.

```
#include <stdio.h>
#include <stdlib.h>

/*
 * Komando lerroan programa eta argumentu bat aurkezten ditugu
```

```
* Argumentuak zuriuneak baditu, sartu komatxo bakarren artean: '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; }
```

15.8.2. Pila hasieratzea

- *Main()* funtzioa edo *_start* instrukzioa exekutatzen hasten denean , pilaren egoera da.

Hurrengoa:

- Pila hasieratzea

ÿ Nukleoak **argc** eta **argv** argumentuak main funtzioko globalari pasatzen dizkio STACK bidez. The funtzioko nagusia deitutako funtzioa da.

11. taula. ABI Hitzarmena: Pila

StackReference	interpretazioa
	argumentu-kateak
	0
1 hitz aldagai bakoitzean	Ingurumena erakusleak
8+8*argc(%rsp)	0
8*argc(%rsp)	- argc ⁰ katearen erakuslea
.....
16(%rsp)	- 2. argumentu katearen erakuslea ÿ argv[1]
8(%rsp)	- 1. argumentu katearen erakuslea ÿ katea argv[0]
0(%rsp)	- argumentu kopurua ÿ argc

15.8.3. Errutina nagusia Itzuliarekin

- Errutina nagusia **irteera** -deiarekin amaitzen ez bada eta **ret** adierazpenarekin amaitzen bada
Dei-konbentzioa funtziodei batena da, beraz, **argc** eta **argv** parametroak pasatzen dira
RDI-RSI-RDX-RCX-R8-R9 erregistroen bidez
- 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      #mezuaren luzera

.atala .testua
.global nagusi

nagusia:
push %rsi #save argument argv
## egiaztu komando lerroak bi argumentu dituela
cmp $2,%rdi
jeje imp_arg
## programa argumenturik gabe badut: inprimatu pantailan
mov $$SYSWRITE,%rax
mov $$STDOUT,%rdi      #fd egongo den fitxategiaren erreferentzia da

idatzi
    mov $$message, %rsi      joango denaren #memoria helbidea
fitxategian idatzi
    mov $$LON,%rdx      #memoria buffer- aren tamaina

idatzi
    syscall      #idatzi komandoa nukleoan
    jmp irteera

imp_arg:
    pop %rsi ->      #pilaren erakusleak gordetako %rsira seinalatzen du eta berreskuratzen dut
    argv -> argv[0]
        gehitu $$ARGV1, %rsi      #rsi lehen erakuslea seinalatzen du, 8 gehitzen baditut seinalatzen du
bigarren erakuslea
    mov (%rsi), %rdi call      #zeharkako bidez bigarren erakuslea lortzen dut
    puts

irteera:
    ret

```

.amaiera

15.8.4. Ariketak: sum_line_com.s ,maximum_line_com.s

1. sum_line_com.s

ÿ Sartu *sum_line_com.s* (bi gehigarriren batura) programaren datuak.
Komando-lerroa

```
### funtzioa: gehitu zifra bateko bi zenbaki oso.
### gehigarriak komando lerrotik pasatzen dira
## Konpilatu x86-64 arkitekturan
## gcc -nostartfiles -g -o sum_input sum_input.s
## korrika 5 7
## x /x %rsp           ->3                   argc: kopurua
argumentuak
## x /a (char**)(%rsp+8) -> 0xfffffd0a4: 0xfffffd26e
## x /c *(char**)(%rsp+8) -> 0xfffffd26e: 47 '/'
## x /s *(char**)(%rsp+8) -> 0xfffffd26e:
"/home/candido/tutorials/as_tutorial/algorithms_x86-32/basic/sum_input"
## p /s *(char**)(%rsp+8) -> 0xfffffd26e
"/home/candido/tutorials/as_tutorial/algorithms_x86-32/basic/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] duka pilaren helbidea
katearen argumentuaren erakuslea duka
    mov 8(%rsp),%rbx          #ebx-en pila= helbidea du edukia
katea
    xor %rcx,%rcx
    movb (%rbx),%cl          #ASCII karakterea

## katearen argumentu erakusleak
    mov 16(%rsp),%rax         #eax-ek pila= helbidearen edukia du
katetik. argv[2]
    mov 24(%rsp),%rbx         #eax-ek pila= helbidearen edukia du
katetik. argv[3]
    ## lortu katea zeharka
    ## Ascii zenbakiak balio bihurtu
    xor %rcx,%rcx
    xor %rdx,%rdx
    movb (%rax),%cl          # zeharkako katea sartzeko
erreferentziatzat argv[1]
```

```

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

ÿ ## 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:
ÿ ## Hitzurrea
bultzatu %rbp
mov %rsp,%rbp
azpian \$8,%rsp #memoria erreserba

ÿ ## argumentu harrapatzea
mov %rdi,%rax mov #1 argumentua
%rsi,%rcx ## gorputza #2 . argumentua

addl %ecx,%eax ## #

gorde emaitza

ÿ ## emaitza EAX-n dago

salto egin:

ÿ ## epilogoa

mov %rbp,%rsp pop # aurreko fotograma
%rbp
ret # berreskuratu itzulera helbidea

2. gehienez_lerro_com.s

ÿ Sartu programaren datuak *maximum_linea_com.s* komando-lerroaren bidez

16. kapitula. C Programazio Lengoaia

16.1. Sarrera

- Hau ez da C lengoaiaren programazioaren tutoriala, kapitulu honen helburua Konputagailuen Egitura irakasgaian erabiltzen diren C lengoaiaren programazioaren alderdi zehatzak komentatzea da.

16.2. galdaKeta

16.2.1. Kontzeptua

- Sintaxia

(mota_izena) adierazpena

ÿ Bihurketa esplizitua operadore unarioa erabiliz ().

ÿ Eragile unarioek lehentasun handiagoa dute eragile bitarrak baino.

16.2.2. Adibidea

- Zenbaki osoen zatiketaren adibidea

```
int i=8,j=5;
flotatu x; x =
i / j; x =
(flotatzalea) i / j;
```

- Aldagai arrunta *int* mota gisa deklaratzen da hasieran

ÿ i/j ÿ 8/5 eragiketak 1 zenbaki osoa izango luke

- Galdaketa (**float**) *i* aldagaiaren gainean egiten badugu , orduan *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

16.3. Erakuslea

16.3.1. Erreferentziak

KN King Testbook: 11. kapitula. Erakusleak. 241. orrialdea

16.3.2. Sarrera

Erakusle kontzeptua oinarrizkoa da maila baxuko programazio inperatiboan, sinplifikatzen duelako

datu-egitura simple edo konplexuak implikatzen dituzten algoritmoak programatzeko kodea.

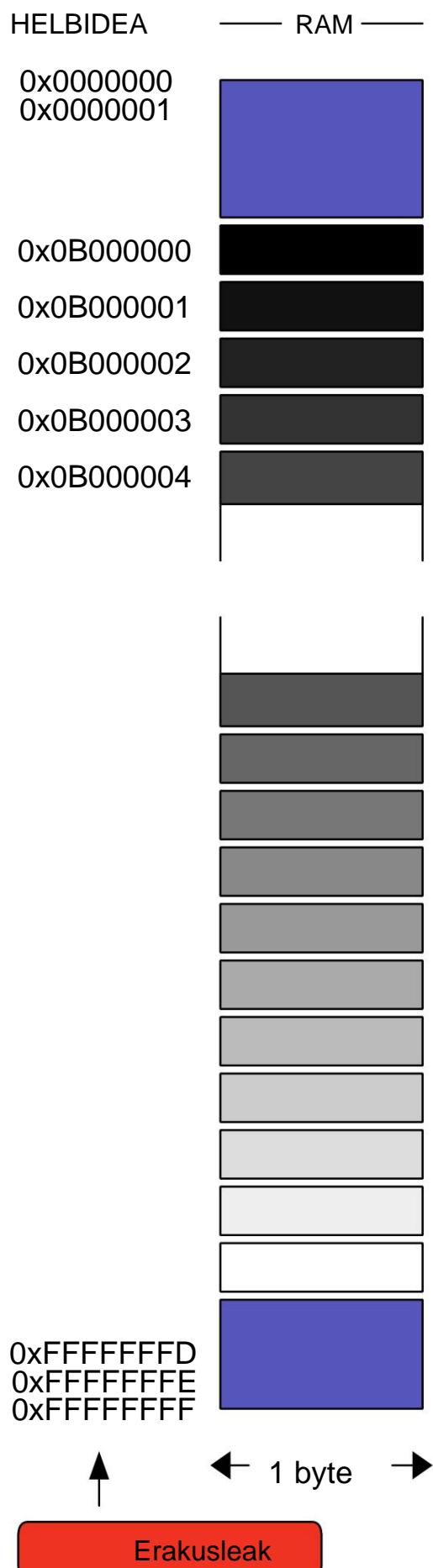
ÿ

Ikasteko: erakusleei lotutako kontzeptuak, haien sintaxia, haien aplikazioa, etab..., beharrezko da programak URRATZ-URRATSO moduan exekutatu memorian dauden objektuen edukiak eta erreferentziak ikusi ahal izateko. GDB araztailea erabiliko dugu.

16.3.3. Kontzeptua

Memoria

- RAM memoria nagusia byte bideragarrieta antolatuta dago.
- Helbide sorta makinaren arkitekturaren araberakoa da.
- Adibidez: 48 lineako 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, float data array, instrukzio-blokeari, etab. Memoria-testuinguru honetan objektu-kontzeptua objektuetara zuzendutako programazioaren objektu-kontzeptutik desberdina da.
- Kokatuta dauden memoria-helbideek erreferentzia egiten duten *objektuak* RAM memorian implementatzen dira. Erreferentzia byte anitzeko objektua gordetzen den lehenengo bytearen helbidea da.
- Memoria mapa:



Erakuslea

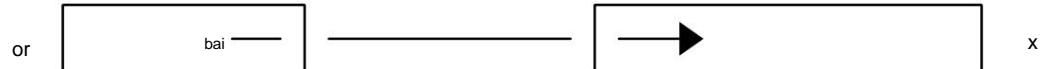
Erakuslea memoria helbide baten baliokidea da.

Erakuslea aldagai bat ordez:

- Memoriaren helbidea adierazten duen datuak gordetzen dituen aldagai da.
- Erakusleen aldagaiak erakusleak gordetzen dituzte.
- Beren balioak memoria helbideen balioetara mugatzen dituzte. ezin du inoiz balio negatiboa edo erreala, etab.
- Objektuak seinalatu
- Objektuak aipatzen dituzte

KNKing liburuak "erakuslearen aldagai" eta erakuslea bereizten ditu. Literaturan, oro har, erakusleei buruz hitz egiten dugunean, erakusle-aldagaietarik ari gara, eta kasu horretan erakuslearen edukiari erreferentzia edo helbidea deitzen zaio erreferentziaozko objektuari.

"Erakuslearen aldagaiaren" irudikapen grafikoa or

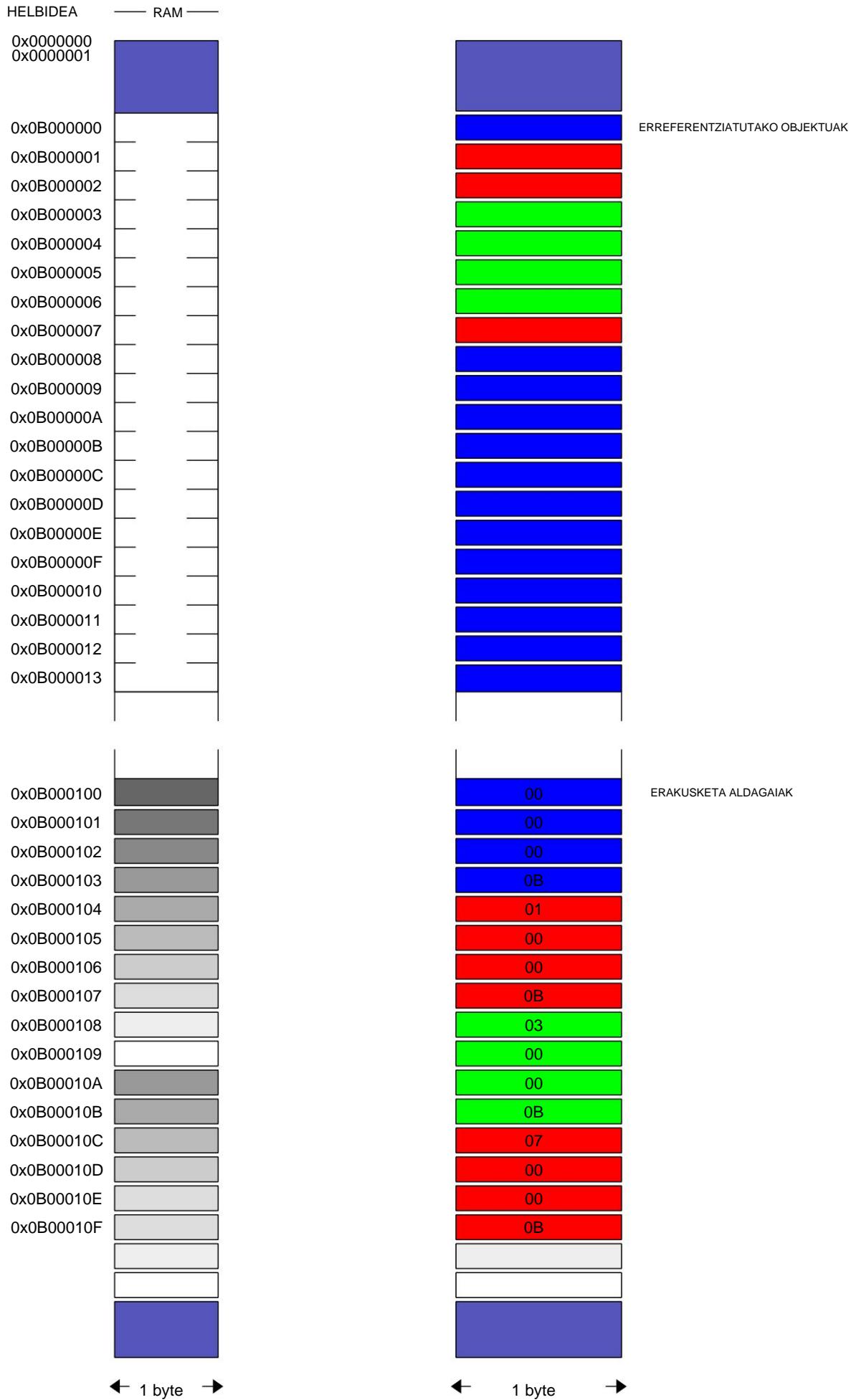


p : erakuslea aldagai-identifikatzalea

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 1 byte-ko objektu batera seinalatzen duen 0x0B000000 helbidea du.
- 0x0B000104 helbidean (+0,+1,+2,+3 byteak) 0x0B000001 helbidea 2 byteko objektu batera seinalatzen duen erakuslearen aldagaiak.
- 0x0B000108 helbidean (+0,+1,+2,+3 byteak) 0x0B000003 helbidea 4 byteko objektu batera seinalatzen duen erakuslearen aldagaiak.

EzkerrekoBalioa-EskuinekoBalioa

- Esleipen-operadore batean (=) erreferentziatutako aldagai arrunt batek beste interpretazio bat du esleipen-operadorearen ezkerrean edo eskuinaldean badago:

$\ddot{y}x=y$

$\ddot{y} x$: ezkerreko aldagai arrunta x-ren memoria helbide gisa interpretatzen da :
x-ren ezkerreko balioa

$\ddot{y} y$: eskuineko aldagai arrunta y-ren memoria eduki gisa interpretatzen da :
y-ren eskubide-balioa

- Objektuaren edukia RightValue da
- Objektuaren erreferentzia LeftValue da
- Erakuslearen aldagai baten edukia erreferentziatutako objektuaren LeftValue da.

16.3.4. Modulu Ilustratzailea

```
/* Erakusleetarako hastapena.*/
```

```
#include <stdio.h>
#include <stdlib.h>
```

```
void main (void) {
```

\ddot{y} /* Kontzeptua */

\ddot{y} /*Operadorearen helbidea*/

```
 $\ddot{y}$  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 = &i; q
= &k; r =
&x; s =
&y; u =
&c; v = &d;

ÿ printf("Sartu karaktere bat \n");ÿ
scanf("%c",&c); printf(" Irakurritako
karakterea %c \n da", c);

ÿ /*Zeharkako operadorea*/
printf(" Irakurritako karakterea %c \n da", *u); printf(
i aldagaiaaren balioa %d baita %d \n", i, *p); printf("Plren balioa %f edo %f ere
\n", y, *s);

ÿ /*StringVariable*/
ÿ /*Matrizea*/ÿ
char string[]{"Kaixo"};

ÿ /*Erakuslea*/
char *agurra="Kaixo";ÿ
char **pt_kaixo;

pt_greeting = &agurra;
irten (0); }

```

16.3.5. 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 =
'@';

```

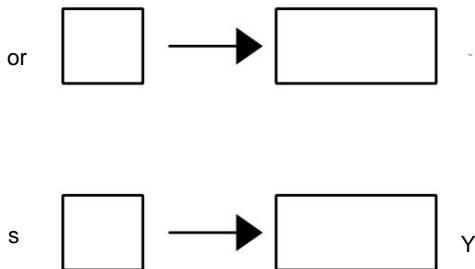
*p, *q, etab... erakusle-aldagaien adierazpenak dira. Izartxoak EZ du eragiketarik egiten aldagaian, erakuslea MOTA adierazteko aurrizkia baino ez da.

16.3.6. Operadorearen helbidea

ampersand _

```
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.



16.3.7. Eragilearen norabidea edo deserreferentzia

Ikurra *

Erakuslearen aldagai baten aurrizkia: erreferentziatutako objektura sartzen da

```
printf(" i aldagaiaren balioa %do %d \n", i, *p); printf("Plren
balioa %f edo %f \n", y, *s);
```

16.3.8. Adibidea

- Mota ezberdinak objektua deklaratzea: osokoa, flotatzalea, kar
- Erakusle motako objektua deklaratu eta hasieratu aurreko objektuekin
- Erakusleak grafikoki irudikatzea
 - ÿ Maila baxua: RAM memoria
 - ÿ Goi Maila: geziez adierazitako laukiak dituzten diagramak.

16.3.9. Erakusleen aplikazioak

- array
 - ÿ Array Erakuslea
 - ÿ Erakusleen Aritmetika
- Katea literala
- Erakuslea Erakuslerantz
- Kateen sarbidea
 - ÿ Array izena
 - ÿ Erakuslearen aldagaien
- Datuen egitura
 - ÿ Izenen zerrenda (kateen erakusleen array)
- Ezaugarriak
 - ÿ Argudioa Erreferentziaz pasatzea
 - ÿ Erreferentzia bidez itzultzea.
- Linux shell komando-lerroko argumentuak.

Array Erakuslea

- Kontzeptua
 - ÿ Array bat erakuslea eta elementuen zerrenda bat da. Erakusleak lehen elementua seinalatzen du prest.
 - 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 matrizea: `datuak_elementuak: 3,67,34,222,45,75,54,34,44,33,22,11,66,0`
 - ÿ Adierazi eta hasieratu
 - ÿ Irakurketa
 - ÿ Idatzi
- CONSTANT erakuslea
 - ÿ EZIN DA ERAKUSLAREN BALIOA ALDATU
 - ÿ Aldatu erakuslea
- Adibidea:
 - ÿ Character Array: `string: H,o,I,a,\0`
 - ÿ Deklaratu eta hasieratu `char string[]={H,o,I,a,\0};`
 - ÿ Irakurketa

ÿ Idatzi

Erakusleen Aritmetika

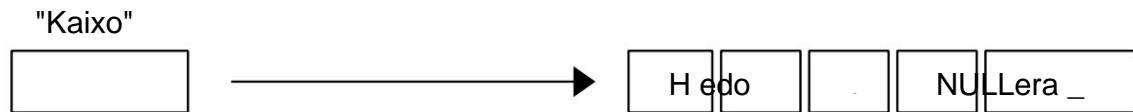
- Indexatzea: lehen elementua GEHI posizio-elementua i
ÿ $\text{datu_elementuak} + i$
- Aldatu matrizeko elementuen erreferentzia-adierazpenak of aritmetiko-adierazpenen bidez erakusleak

3.16.10. Katea literala

- Bi etapako kontzeptua
ÿ "Kaixo" izeneko array, zeinaren elementuak karaktere motakoak diren.

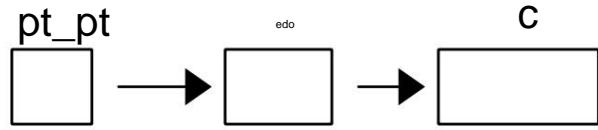


ÿ Hasieratu Array String *Hello* -rekin



- 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: irakurri eta idatzi
 - ÿ Sartu hasierako matrizean: irakurri eta idatzi
 - ÿ Array kopia `kate1=kate2` esleipenaren arabera ?

3.16.11. Erakuslea Erakuslea



- Adibidea

ŷ *u* pertsonaia seinalatzen du *c*

ŷ *pt_pt u -ra* seinalatzen du

3.16.12. StringVariable

Array izena

- **Katearen** karaktere-matrizea deklaratzen dut eta Kaixo katearekin hasieratzen dut : `char string[]{"Kaixo";`

Erakusle aldakorra

- **Agurra** aldagai deklaratzen dut eta **kate** -erakuslearekin hasieratzen dut
ŷ `char **kaixo`

3.16.13. Funtzioak

- Argudioa **Erreferentziaz** pasatzea
 - ŷ Funtzio-parametroak erakusle-aldagai gisa deklaratu.
- **Erreferentzia bidez itzultzea.**
 - ŷ Adierazi itzuleraren balioa erakusle gisa.

17. kapitula Praktikak Eranskina

17.1. Praktikak

17.1.1. Dokumentazioa: gidoiak, bibliografia, oharrak

- Nire ikasgelan eskuragarri:

ÿ Oharrak `eecc_book.pdf`, gidoiak, tranpa-orriak, eranskinak, praktika-ariketak eta autoebaluazioa eta teoria.

ÿ Iturburu kodea .s duten moduluak (nire ikasgela/Baliabideak/praktikak/iturburu_kodea.zip) [esteka G1](#) [https://miaulario.unavarra.es/portal/site/2018_0_240306_1/tool/d61518aa-72cb-45df-a7f8-b71f90e7907e?panel=Nagusia] praktika guztiak erabilitako *NUPeko ikasgelako zerbitzarian eskuragarri daude: Baliabideak/praktikak*

ÿ Mihiztadura-hizkuntzako laborategiko gidoiak oinarritzen diren testuliburuak:
[Programazioa oinarritik](#).

17.1.2. Garapen Plataforma

Tresnak

- Argitaletxeak

ÿ [Editoreak](#) [<https://www.tecmint.com/best-open-source-linux-text-editors/>]: gedit, emacs, vim, sublime, Kate,

ÿ [Edizio, konpilazio, arazketa tresna integratuak](#) [<https://www.tecmint.com/best-linux-ide-editors-source-code-editors/>]: eclipse CDT, netbeans, code::blocks, codelite, Microsoft-en Visual Studio Code Editor, jetbrains clion, jeany, ajunta , GNAT Programming Studio, emacs, kdevelop, codestudio, etab.

- Deiturak

ÿ [i386](#) : x86-32 arkitekturarako linux izena
 ÿ [amd64](#): linux izendatzea x86-64 arkitekturari
 ÿ [IA32](#) —Intel-en izendapena x86-32 arkitekturarako
 ÿ [IA64](#) —Intel-en izendapena x86-64 arkitekturarako

- 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](#) : errendatzalea

- Beharrezko liburutegiak, gcc, as, ld tresnak arkitekturan funtzionatzeko [i386](#) 32 bitekoa.

ÿ `dpkg -l gcc-multilib :`

```
Desiratua=ezezaguna (U)/Instalatu/KenduR/Purge/Eutsi (H) | Egoera=Ez/Inst/
Conf-fitxategiak/unpacked/half-conF/half-inst(H)/wait-trigger(W)/pending-trigger
|/ Err?=(bat ere ez)/require-Reinst (Egoera,Err: maiuskula.=txarra)
```

||/ Izena Arkitektura Deskribapena Bertsioa

+++-----ii gcc-multilib 4:7.3.0-3ubu amd64-----

GNU C konpilatzalea (multilib fitxategiak)

ÿ Lehenengo bi zutabetan "Desirada/Egoera" ez badu **ii**, esan nahi du ez direla instalatutako liburutegiak.

ÿ Interneteko sarearen bidez eskuragarri dagoen biltegian daudela egiaztatzen dut:

ÿ `apt-cache show gcc-multilib` : biltegia

ÿ `sudo apt-get install gcc-multilib` : deskargatu eta instalatu administratzaile-eskubideak badituzu soilik

- Erreminta-katea

ÿ `gisa --version & Id --version & gcc --version` : ohartu bertsioak

lineako programazioa

- [ias assembler unicamp online](https://www.ic.unicamp.br/en/~edson/disciplines/mc404/2017-2s/abef/IAS_Assembler/assembler.html) [https://www.ic.unicamp.br/en/~edson/disciplines/mc404/2017-2s/abef/IAS_Assembler/assembler.html]
- [gdb-online](https://www.onlinegdb.com/classroom) [<https://www.onlinegdb.com/classroom>]

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-asanblada

ÿ `Id` : lokailu

ÿ `cc` : C konpilatzalea

ÿ [GCC](#) : automatikoko tresna-katearen frontend-a: Gnu Compiler Collection. ezberdinak gidariak
hizkuntzak iturburu-fitxategiaren luzapenaren arabera.

ÿ [gizon gcc](#)

ÿ [GDB](#) : arazketa.

ÿ [gizon gdb](#)

17.1.3. Txostenaren dokumentua: edukia eta formatua

Edukiak

- Praktikaren garapenean:
 - a. *Iruzkinen* garatutako programen iturburu-kodea berriro editatu behar da .
 - b. Konpilatu iturburu-modulua *lineako komandoak* erabiliz
 - c. Aztertu iturburua eta kode bitarra *arazketa erabiliz*: egin beharreko eragiketak arazketa beharrezkoa da fitxategi batean gordetzea.
- Praktikan zehar beharrezko da Testu Editorea irekita edukitzea joateko memoria gauzatzea praktikaren burutzapenarekin batera.
- Memoria-dokumentuak eduki behar du:
 - ÿ Praktikaren izenburua eta datu pertsonalak dituen azal bat.
 - ÿ Aurkibide gisa aurkibide bat duen lehen fitxa, ez da beharrezko Zenbakia adierazi orrialdea.
 - ÿ Iturburu-modulu komentatuak,
 - ÿ Konpilatu eta azterzeko komandoak.
 - ÿ GDB komandoen historia eta haien irteerak, laborategian erabilitakoak.
 - ÿ Ondorioen atal bat praktikan ikasitakoarekin.
 - ÿ Ebatzi gabeko zalantzen atala.
 - ÿ Txostenean zehar agertzen diren galdera esplizituak, halakorik balego.
 - ÿ **AUKERAZ** Autoebaluazio galdetegiko galderak eta erantzunak Praktikak. Ikus Ebaluazioa atala.
 - ÿ Era guzietako informazio pertsonala beharrezko da azterketan erabiltzeko ohar gisa.

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 titularraren izenak ez du azenturik, zeinurik edo zuriunerik izango
 - ÿ XX praktika-taldea esan nahi du: P1 edo P2 edo P3
 - ÿ N praktika saioaren zenbakia esan nahi du: 1,2,3,4 edo 5.

Memoria Dokumentua entregatzea

- Bidali Memoria Dokumentua Miaulario Server **Tasks** aplikazioaren bidez. Epea irakasleak etxerako lanen egutegiaren bidez adieraziko du. Txostenak epez kanpo bidaltzeak ohiko deialdian praktika hori egiteko aztertu behar izatea dakar.

17.1.4. Ebaluazioa

- Baloratuko dira:
 - ÿ Oroimenaren entrega ezarritako kanaletik puntu 1eko zigorrarekin bakoitzagatik berandu eguna.
 - ÿ memoriaren 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 zailak diren edo kodea ulertzeko garrantzitsutzat jotzen diren argibideak.

ÿ Aukerako **Praktiken Autoebaluaziorako** Galdetegia

ÿ

Irakasleak etengabe ebaluatuko ditu ikaslearen jarrera eta laborategian egindako lana, ikaslea azterketatik askatu ahal izateko, egindako ezagutzak eta eginkizunek hala frogatzen badute.

ÿ

Ikastaroko oharren V. kapituluan aurki daitezkeen **Praktiken Autoebaluazio** galdetegiko galderak eta erantzunak praktiketako orduetatik kanpo egiten dira modu pertsonalean. Autoebaluazia egiten ez bada, txostenaren gehienezko puntuazioa **6 puntukoa** izango da eta egiten bada, gehienezko puntuazioa **10 puntukoa izango da**.

17.1.5. Programazioa

Metodologia

- Garatu beharreko programaren adierazpena irakurri.
- Editatu algoritmoaren deskribapena Pseudokode gisa:
 - ÿ Datu-egiturak eta instrukzio-egiturak definitzen dituen algoritmoa garatu.
 - ÿ konstanteak, aldagaiak, matrizeak, erakusleak, hasieraketak, begiztak, hautaketa-adierazpenak, funtziak 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 behemailara. Atalak, aldagaiak, arrayak, erakusleak, hasieraketak, begiztak, hautaketa-adierazpenak, azpierrutinak eta parametroak, programaren sarrera eta irteera eta abar itzultzea.
- Bihurtu RTL kodea **AT&T** batzar hizkuntzara x86 arkitekturarako.
- **Gcc** -rekin edo tresna katearen bidez konpilatzea: as-ld
 - ÿ Araztu sintesi-erroreak.
- Exekuzioa: erroreak araztu urratsez urrats moduan **GDB** arazketa erabiliz

17.1.6. Konpilazioa

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 esatea zehatzagoa izango litzatekeen arren memoria nagusian karga daiteke.

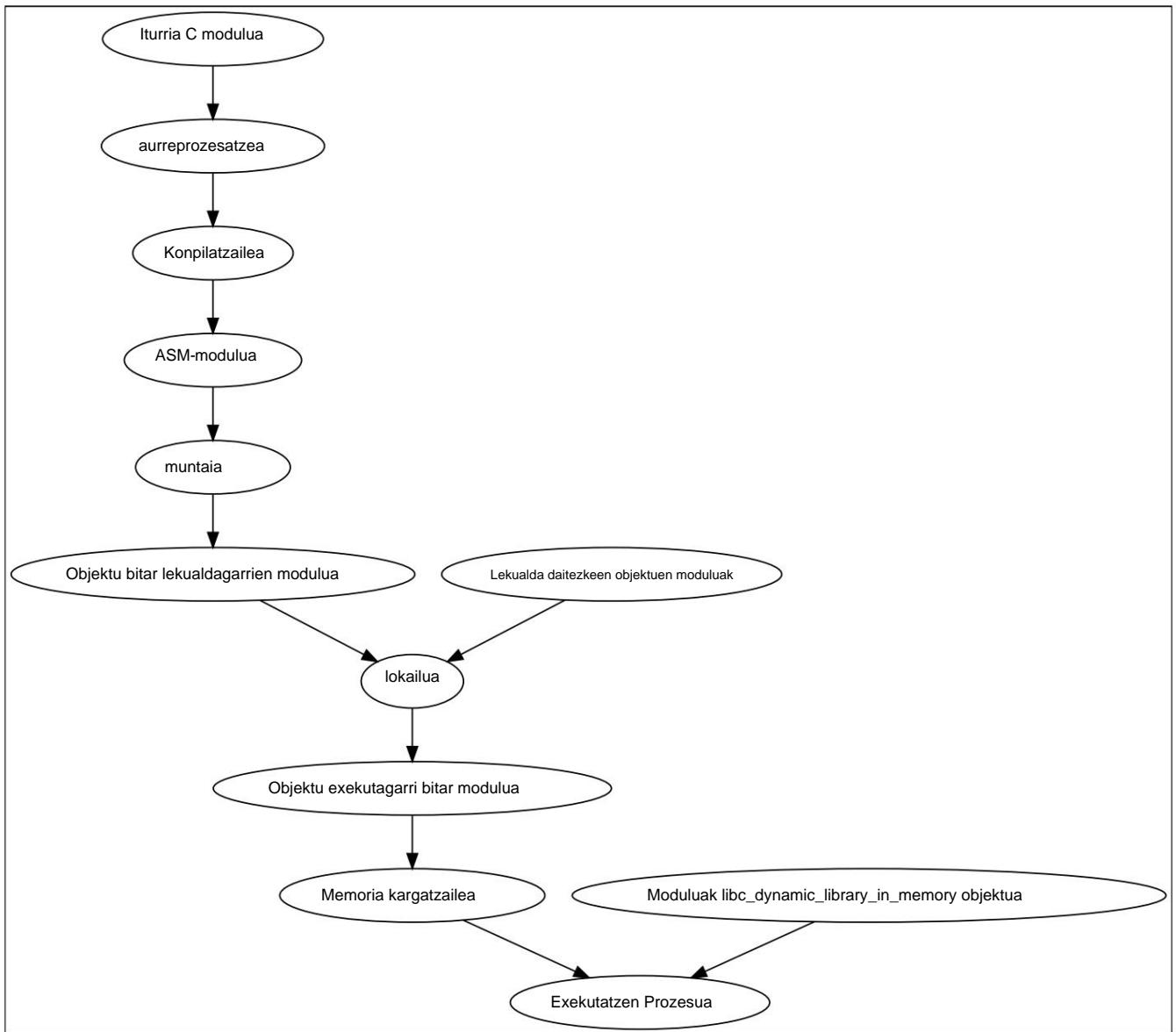
ÿ memoria nagusian kargatu

ÿ Sistema eragileak automatikoki egiten du programa exekutagarria terminal batetik deitzean edo mahaia.

ÿ `gcc -m32 -g -o sum1toN sum1toN.c`

ÿ *-g* - *sum1toN.c* iturburu-programaren sinboloen taula GDB arazketarako sortu eta *sum1toN* modulu exekutagarrian txertatuko dela zehazten du. Horrela, kode bitarra, etiketa batena adibidez, bere sinboloarekin (testu hizkuntza) lotzen da.

Eraikitzeko faseak



39. Irudia Konpilazio-prozesua

- Gelditu konpilazioa 1. fasean:urreprozesatza: `gcc -E sum1toN.c -o sum1toN.i`
 ÿ *.:i: Aurreprozesadorearen irteera: kendu kodea ez den informazioa (iruzkinak, etab.)
- Gelditu konpilazioa 2. fasean: itzuli C mihiztatzailea: `gcc -S sum1toN.c -o sum1toN.s`
 ÿ .s— **Mihiztagailuaren sorburu-hizkuntzaren moduluas**
- Konpilazioa gelditu 3. fasean: Sortu lekuz alda daitekeen objektuaren modulu: `gcc -c sum1toN.c -o batuketa1toN.o`
 ÿ *.o: lekuz alda daitekeen objektu-modulu : Lokatzaileak sistema eragileko beste objektu-modulu batzuekin, C *libc* liburutegiarekin edo beste programatzaille-modulu batzuekin lotu aurretik kode bitarra .
- 4 faseak burtu: Objektu exekutagarriaren modulu sortu: `gcc -c sum1toN.c -o sum1toN`
 ÿ luzapenik gabeko fitxategia: objektu exekutagarria modulu: izateko konfiguratutako modulu bitarra memoria nagusian kargatu eta CPUak exekutatzen du.
- `gcc -m32 --save-temps -o sum1toN sum1toN.c`
 ÿ --save-temps: gcc-k prozesu osoaren 3 fitxategi partzialak (gorde) sortzen ditu (gorde)

eraiki .i,.s,*.o .

ÿ Egiaztu guztira 5 fitxategi ditugula: .c,.i,.s,.o eta luzapenik gabeko exekutagarria.

tresna-katea

- Konpilazio-fase desberdinak exekutatzen dituen **gcc** kontrolatzailarekin 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 bere tresna propioa: **gcc -S sum1toN.c -o batuketa1toN.s**

ÿ **as**: Muntatzeko tresna edo muntatzailea: **as --32 --gstabs -o sum1toN.o sum1toN.s**

ÿ --32: 32 biteko arkitektura

ÿ --gstabs: sinboloen taula sortu

ÿ -o : irteera fitxategia : lekuz alda daitekeen modulu bitar *.o

ÿ **ld**: Lotura tresna: **ld -melf_i386 -o sum1toN sum1toN.o**

ÿ -melf_i386 : 32 biteko arkitektura

ÿ -o: irteera fitxategia: modulu bitar exekutagarria

muntaia-lengoaiaren iturburu-moduluia

- Iturburu-programa modu funtzional/operatibo abstraktuan komentatu eta ez RTL modu literalean.
- Eskuzko erreminta katea:

ÿ **as --32 --gstabs -o sum1toN.o sum1toN.s** : muntaia

ÿ *.s : iturburu-moduluia asm hizkuntzan

ÿ *.o : lekuz alda daitekeen objektu-moduluia

ÿ --stabs: 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

- Erreminta-kate automatikoa

ÿ **gcc -m32 -nostartfiles -g -o sum1toN sum1toN.s**

ÿ -m32: i386 arkitekturarako iturburu eta objektu moduluak.

ÿ -nostartfiles - Sarrera-puntu nagusia ez dela zehazten du, _hasiera baizik .



Sarrera-puntu nagusia bada, estekatzailarei jakinarazi behar zaio sarrera-puntu nagusia dela:
gcc -e main -m32 -nostartfiles -g -o sum1toN sum1toN.s y **ld -e main -melf_i386 -o sum1toN sum1toN.o**

- g: *sum1toN.s* iturburu-programaren ikur-taula sortuko dela zehazten du arazketarako

GDB eta *sum1toN* modulu exekutagarrian txertatu

17.1.7. Ohiko akatsak

gcc

- Ubuntu 18.0-n amd64-rako konpilatuta badago (gcc -nostartfiles -g -o sum1to64 sum1to64.s) eraikuntza gelditzen da errore-mezuarekin:

```
/usr/bin/x86_64-linux-gnu-ld: /tmp/ccbhD6Vr.o: `._data'ren aurkako R_X86_64_32S lekualdatzea 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: -foot aukera lehenespenez aktibatuta dago eta desaktibatu egin behar da

ÿ irtenbidea: (gcc **-no-pie** -nostartfiles -g -o sum1to64 sum1to64.s)

gdb

- Fitxategi batean gordetzeko gdb komandoen erregistro historikoa desgaituta dago.

18. kapitulu. amd64 arkitektura

18.1. Iturburu-modulua: sum1toN.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: AMD64 arkitekturako GNU montaia-lengoaia
### 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 aldagai ezartzen du
    movq n,%rdx
begizta:
    gehitu %rdx,%rdi
    azpi $1, %rdx
    jnz begizta

## RDI bidez OSra irteera argumentua ABI AMD64 konbentzioaren arabera
## irten
    mov $60, %rax # OS dei-kodea: irten azpierrutina
    syscall # dei sistema eragileari, azpierrutina exekutatu dezan
RAX balioa

.amaiera
```

19. kapitula. Aurreko Ikastaroetako azterketak

19.1. 2018 urtea

19.1.1. Azaroa

1. Proba Partziala. 2018ko azaroaren 10a.

Informatikan Gradua 2. maila. Konputagailuen Egitura.
Nafarroako Unibertsitate Publikoa.

Iraupena: 90 minutu.

Abizena:

ÿ

Idatzizko informazio mota oro erabil daiteke, hala nola praktiken txostenak, oharrak, erreferentzia-orriak, etab. Ezin da erabili gailu elektronikorik, hala nola kalkulagailuak, telefonoak, ordenagailuak, etab... Emaitza lortzeko beharrezkoak diren garapen mota guztiak sartu behar dira erantzunetan.

1. Von Neumann Institute Advanced Studies (IAS) Ordenagailua:
 - a. (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 puntu) Zein da MAR, MBR eta PC hiru osagaien arteko erlaziona?
2. (1 pt) Zein da "Kaixo \n" sei karaktere-katearen kode digitala
3. (1 pt) 0123 eta 0777 zenbakiak zeinurik gabeko zenbakiak dira oinarri zortzitalean. batuketa egin 0123+0777 zuzenean zortziko oinarrian.
4. (1 pt) 0xABCD eta 0xEFED zenbakiak bi osagarri-zenbakiak dira. Egin 0xABCD-0xEFED kendu zuzenean. Kalkulatu emaitzaren balioa.
5. (1 pt) Adierazi 6,25 zenbaki hamartarra IEEE-754 doitasun bikoitzeko formatuan.
6. Argibide-formatua:

ÿ Ordenagailu batek 256K hitzeko memoria-unitatea du 32 biteko bideragarri bakoitza bytez byte. Instrukzio bat memoriako hitzetako batean gordetzen da. Instrukzioak lau eremuko formatua du: zeharkako bit bat, operazio-kode bat, 64 erregistroetako bat zuzentzeako eragigai-eremu bat eta memoria helbideak dituen eragigai-eremu bat.

 - a. (2 pt) Zenbat bit osatzen dute eragiketa-kodearen eremua? Eta erregistro-eremuak? Eta helbide-eremuak?
 - b. (2 pt) Zenbat bit dira unitatearen helbide-busaren eta datu-busaren parte? Orioimena?
7. (2 pt) Azpirrutina batean adierazi zer erlazio dagoen markoaren "marko-erakuslea" erakuslearen artean azpiprogramaren eta itzulera helbidea gordetzen den memoria helbidea.

8. (3 pt) Osatu erantsitako mihiztatzaile-lengoaiaren programaren iturburu-kodea kontuan hartuta zenbatu hurrengo iturburu-moduluau erantsitako iruzkinak non algoritmoa garatuak zenbaki hamartar bat kode bitar bihurtzen du:

```
### Programa: convert_decbins
### Deskribapena: 15 zenbaki natural hamartarra bitar bihurtzen du erabiliz
ondoz ondoko zatiketak 2z
### Kode bitarrak 32 biteko luzera du
### gcc -m32 -g -nostartfiles -o convert_decbins convert_decbins.s
### Muntaketa gisa --32 --gstabs convert_decbins.s -o convert_decbins.o
### estekatzalea -> ld -melf_i386 -o convert_decbins convert_decbins.o

## MAKRO
```

DATUAK

```
esan: .          # hamartar (4 byte tamaina) kode bitar bihurtzeko
32 bit
## bin-ek alderantziz gordetzen du kodea, bin[0]-k bit baxuena gordetzen du
pisua.
bin: .space 32 # 32 byte array: kodearen bit bat gordetzen du byte bakoitzean
32 biteko bitarra.
zatitzalea: . # zatitzalea (1 byteko tamainakoa)
```

JARRAIBIDEAK

```
## hasieratu ECX zatitzalearen balioarekin
```

```
## hasieratu bin array indizea
```

```
## EAXn kobratzen dut dibidendua
# eax <-x

## hedatu EDX-n dibidenduaren zeinu-bitia
# Dibidendua beti da positiboa

## Ondoz ondoko zatiketak 2z zatidura 0 berdina izan arte
begizta:
## idivl : [EDX:EAX] / Operand_source
# EAX<-Kotientea{x/y}, EDX<-Hondarra{x/y}
# gorde gainerakoa (1 byte-koia) bin array-n
## Iruzkiak zeinu-bit edx-en
# Dibidendua beti da positiboa
```

```
## egunera matrizearen indizea  
  
## egiaztu zatidura zerora iritsi den begiztatik irteteko  
  
## Itzuli kode bitarren bit kopurua EBX-n  
  
## OS dei-kodea  
  
## Eten errutina eta deitu OS
```

ÿ GDB araztearen komandoak erabiltzea

- a. (2 pt) inprimatu "bin" array-aren edukia bi adierazpen ezberdinekin "arakatu" eta/edo "inprimatu" komandoak erabiliz.
- 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_decbin.s" programa behar den kodearekin ongietorri-mezu bat pantailan inprimatzeko zuzeneko idazketa-deia erabiliz.

19.2. 2017 urtea

Proba Partziala. 2017ko irailaren 22a.
Informatikan Gradua 2. maila. Konputagailuen Egitura.
Nafarroako Unibertsitate Publikoa.

Iraupena: 30 minutu.

Abizenak:

Izena:

1. Von Neumann ereduan, zein da Kontrol Unitatearen funtzioa?
2. Zeintzuk dira Von Neumann makinaren instrukzio-zikloaren fase desberdinak.
3. Bihurtu 291 zenbaki hamartarra oinarri zortzitalera.
4. Egin -18-21 eragiketa 2 osagarrian.
5. Zein da abstrakzio kontzeptua ordenagailuaren antolaketan.
6. Implementatzen duen IAS makinaren sum.ias mihiztadura-lengoaia programa garatu
 $s=1+2$ algoritmoa.

Proba Partziala. 2017ko urriaren 10a.
Informatikan Gradua 2. maila. Konputagailuen Egitura.

Nafarroako Unibertsitate Publikoa.

Iraupena: 30 minutu.

Abizenak:

Izena:

Y

Idatzizko informazio mota guztiak erabil daitezke, hala nola praktiken txostenak, oharrak, erreferentzia-orriak, etab.

1. Osatu exa_2017.s iturburu-modulua AT&T x86-32 mihiztadura hizkuntzan. (6 puntu)

```
### Informatiken Egitura ikastaroa 2017-18. 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-matrizea tamainako bost zenbaki oso negatibo ditu
```

```
bi byte,
```

```
### -5etik -1era, non -5 zero posizioaren balioa den.
```

```
### Kopiatu zerrenda-matrizearren edukia bufferera.
```

```
### Bufferra zeharka sartzen da erakusle-aldagaiaren bidez EAX
```

```
### Sistema eragilera bidalitako irteera argumentuak izan behar du
```

```
### zerrenda-matrizearren lehen balioa.
```

```
## MAKRO
```

```
.equ      SYS_EXIT, 1 # Sistema eragilearen deiaren kodea
```

```
LEN, .equ ##           5 # Array eta buffer luzera
```

```
ALDAGAIAK: zerrenda eta buffer
```

```
.datuak
```

```
zerrenda: # Array hasieratutako datuekin HEXADECIMAL-en irudikatuta
```

```
-----  
buffer: # Esleitu memoria hasieratu gabeko bufferrako.
```

```
## JARRAIBIDEAK
```

```
## Sarrera puntuak
```

```
-----  
_hasi:
```

```
## hasieratu irteerako argumentua zerora
```

```
-----  
## hasieratu erakuslea aldagai EAX
```

```
-----  
-----  
## hasieratu begizta iterazio kopuruarekin. Erabili makroak.  
mov ,%esi  
begizta:  
-----  
-----  
-----  
-----  
dec %esi  
jns begizta  
## irten  
  
mov      _ _ _ _,%eax  
  
int .end -----
```

ÿ Arazoak:

ÿ gdb komandoa bufferaren edukia bistaratzeko kopia amaitutakoan (2 pt):

ÿ .
ÿ (gdb)

ÿ Zerrendako etiketak 0x00555438 helbidea seinalatzen badu, adierazi zein den.
helbideak (2 pt):

ÿ .
ÿ 0x0055543C:

ÿ .
ÿ 0x0055543D:

Proba arrunta. 2018ko abenduaren 7a.
Informatikan Gradua 2. maila. Konputagailuen Egitura.
Nafarroako Unibertsitate Publikoa.
Iraupena: 45 minutu.

1. ZATIA (10 puntu)

- Iraupena: 20 minuto

- Titulazioa:

- (3 pt) Zenbaki sinatu gabeko kenketa: 0x8000 - 0x7AFF ÿ eragiketak urtean egin behar dira
HEX kodea esklusiboki

2. (3 pt) Zenbaki sinatutakoen kenketa: 0x8000 - 0x7AFF ѕ eragiketak kode HEXADECIMAL hutsean egin behar dira

3. (3 pt) Lotu esaldi bakarrean kontzeptuak: programa-kontagailua, datu-bidea, instrukzio-zikloa, sekuentziatzalea, mikroordenak, unitate logiko aritmetikoa, mikroarkitektura, instrukzio-harrapaketa.

2. ZATIA (10 puntu)

- Iraupena: 25 minutu
- Titulazioa:

1. (6 pt) Garatu iturburu-modulua *string_length.s* mihiztadura hizkuntzan AT&T x86-32.

```
/*
```

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 zentzua duten kode-blokeen bidez komentatu behar da, ez makina-instrukzio bat deskribatzen duten kode-lerroen bidez. */

```
## MAKRO definizioa
.equ SUCCESS,
0 .equ SYS_EXIT,
1 .equ END_CHAR, '\0'
```

- Galderak: (4 puntu)

ÿ Dos gdb komandoa helbide-katean gordetako objektuaren edukia bistaratzeko

ÿ (gdb)

ÿ (gdb)

ÿ gdb komandoa katearen amaierako karakterea soilik bistaratzeko.

ÿ (gdb)

ÿ Adierazi goiko iturburu-programa erabiliz konpilatzeko behar diren bi komandoak
eskuzko tresna-kate bat, gcc frontend-a erabili gabe.



KLUSTERAK:

ABIZENAK:

IZENA:

Proba arrunta. 2018ko abenduaren 7a.

Informatikan 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 lokaleko 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) Ordenagailuko CPU baten mikroarkitekturaren bloke-diagrama a
 16 biteko hitzaren tamaina erantsitako fitxako irudiarenarekin bat dator. -ren ISA
 Ordenagailu honek mihiztadura-lengoaia bat du, horri dagokion
 mnemoteknia eta AT&T x86-32 sintaxia. Makina kodea memoria nagusian kargatzen da.
 iturburu-moduluaren jarraibideen atalari dagokiona, honako hau:

```
movw $0xF000,R0
movw R0,R1
gehitu R1,R0
subw R1,R0
```

ÿ Kontrol-unitatearen sekuentziatzalea 4 egoerako makina gisa diseinatuta badago
 T0,T1,T2 eta , erantsitako taulan adierazi zein den egoera bakoitzean exekutatu beharreko mikroaginduak
 T3 irakaskuntza-zikloa programako instrukzio bakoitzeko.

	T0	T1	T2	T3
mugitu \$0xF000,R0				
movw R0,R1				
gehitu R1,R0				
subw R1,R0				

1. (4 pt) Memoria hierarkikoaren antolaketa

ÿ Programa bat osatzeko prozesuan, hasierako edizio-fasetik programa memoria nagusiko prozesu batean kargatzera arte, "toolchain"-ak memoria-espazio desberdinak sortzen ditu kodearen itzulpen-prozesuaren modulu ezberdinetan. Bete erantsitako taula sortutako espazio bakoitzaren ezaugarriekin.

Programa tresna		ren egitura ren espazioa memoria eta Helbideratzea to	helbide mota	Kode lokalizazioa
Edizioa	Iturria Modulua	Atalak eta etiketak	Birtuala, Ez-lineala	Bigarren mailako memoria: Disko gogorra

ÿ Cache kontrolatzaileak memoria-hierarkiaren barruan cachea eta RAM dinamikoa nola egituratzen dituen.

ÿ Fisikoki, zertan datza RAM memoria dinamikoko zelula bat?

ÿ Nola sinkronizatzen dituen datu-transferentziak sistema-busaren bidez, ahari bat
Datu-tasa bikotza dinamikoa DDR.

1. (3 pt) I/O eragiketen mekanismoak

ÿ Marraztu teklatuaren tekla baten eta ordenagailuaren von Neumann arkitekturako oinarrizko unitateen arteko beharrezko HWren bloke-diagrama eten mekanismoaren bidez datu-transferentzia egiteko.

ÿ Marraztu teklatuaren tekla baten eta ordenagailuaren von Neumann arkitekturako oinarrizko unitateen arteko beharrezko SWaren bloke-diagrama eten mekanismoaren bidez datu-transferentzia egiteko.

20. kapitulua. Miaulario: Bideokonferentzia

20.1. Sarrera

- Informazioa
 - ÿ https://miaulario.unavarra.es/access/content/group/b15fc60b-3c66-452d-a7d9-42ec8cdab3a1/CSIE_WEB/site_csie/zoom-gida/zoom-gida_en.html
 - ÿ csie@unavarra.es posta elektronikoa [mailto:csie@unavarra.es]

20.2. Zoomaren instalazioa

- Hizlariak ZOOM aplikazioa bere oinarrizko bertsioan instalatu behar du (doan).
- <https://zoom.us/>
- Izena ematea beharrezkoa da
- Firefox arakatzailearen Ubuntu 18.0 bertsioan ezin da bezeroa abiarazi
- Deskargatu zoom_amd64.deb paketea
- instalazio komandoa: `dpkg -i zoom_amd64.deb`
- Ireki bezeroa: `./zoom`

20.3. Zoomaren erabiltzailearen gida

20.3.1. Ezarpena

- Probatu audioa

20.4. bideokonferentzia saioa

- Nire ikasgelatik ÿ saioa hasi ÿ gaia ÿ bideokonferentzia
- https://miaulario.unavarra.es/access/content/group/b15fc60b-3c66-452d-a7d9-42ec8cdab3a1/CSIE_WEB/site_csie/zoom-gida/zoom-gida_en.html

IX Bibliografia

Konputagailuen Arkitektura

ÿ [WS_eu] William Stallings. Ordenagailuen antolaketa eta arkitektura. 7. edizioa, Pearson Prentice Hall berrargitalpena 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. du Hardware/Software interfazea. Morgan Kaufmann. 2009. Unibertsitate gehienetan Standard Book.

ÿ [MoMano] Sistema informatikoen arkitektura, Morris Mano.

x86

ÿ [BANDERAK] https://en.wikipedia.org/wiki/FLAGS_register

Muntatzaileen Programazioa

ÿ [PGU] [Programazioa oinarritik](http://programminggroundup.blogspot.com.es/2007/01/programazioa-from-up-up.html) [<http://programminggroundup.blogspot.com.es/2007/01/programazioa-from-up-up.html>] Jonathan Bartlett-ek editatua Dominick Bruno, Jr. Copyright © 2003 Jonathan Bartlett-ek argitaratua Bartlett argitaletxeak Broken Arrow, Oklahomen ISBN 0-9752838-4-7

ÿ [ATT] [Oracle AT&T hizkuntza](https://docs.oracle.com/cd/E26502_01/html/E28388/eoiyg.html) [https://docs.oracle.com/cd/E26502_01/html/E28388/eoiyg.html]

ÿ [WikiBook] [Oharrak WikiBook:x86 Asanblada: AT&T](https://en.wikibooks.org/wiki/X86_Assembly) [https://en.wikibooks.org/wiki/X86_Assembly]

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 tutorialspoint](https://www.tutorialspoint.com/assembly_programming/aurkibidea.htm) [https://www.tutorialspoint.com/assembly_programming/aurkibidea.htm]

ÿ [NASM_bristol] [Oharrak Bristol Community College: NASM Prog.](http://www.c-jump.com/CIS77/CIS77syllabus.htm) [<http://www.c-jump.com/CIS77/CIS77syllabus.htm>]

ÿ [i386] [i386 \(32 biteko\)](http://www.cs.nyu.edu/~mwalfish/classes/ut/s13-cs439/ref/i386/toc.htm) [<http://www.cs.nyu.edu/~mwalfish/classes/ut/s13-cs439/ref/i386/toc.htm>]

ÿ [A64] [amd64 \(64 bit\)](http://www.felixcloutier.com/x86/) [<http://www.felixcloutier.com/x86/>]

ÿ [IAL] [Intel asm hizkuntza](https://software.intel.com/en-us/articles/introduction-to-x64-assembly) [<https://software.intel.com/en-us/articles/introduction-to-x64-assembly>]: Intel Oinarrizko Arkitektura Ofiziala

ÿ [AMD] [AMD ofiziala](http://support.amd.com/TechDocs/24593.pdf) [<http://support.amd.com/TechDocs/24593.pdf>]. 3. liburukia. 2.3 Erregistroen laburpena eta Datu motak

ÿ [paul_carter] [Paul Carter PC-en mutua-lengoia. Doako sarbidea. 2006.](http://pacman128.github.io/pcasm/) [<http://pacman128.github.io/pcasm/>]: Netwide Assembler NASM, Intel hizkuntza

ÿ [j_dunte] [Jeff Duntemann](http://www.duntemann.com/) [<http://www.duntemann.com/>]. Batzar hizkuntza urratsez urrats: Linux-ekin programatzea. Wiley Ed. 3. Edizioa. 2009.

ÿ [irvine] [Kip R. Irvine](http://kiprvine.com/) [http://kiprvine.com/]. x86 prozesadoreetarako muntaia-lengoaia. Pearson. 6 Edizioa. 2014.

Behe-mailako programazio dokumentuak

ÿ [ABI_i386] [ABI i386](https://www.uclibc.org/docs/psABI-i386.pdf) [https://www.uclibc.org/docs/psABI-i386.pdf]

ÿ [ms_call] MicroSoft-en [dei-konbentzioa](https://docs.microsoft.com/es-es/cpp/build/x64-software-konbentzioak?view=vs-2017) [https://docs.microsoft.com/es-es/cpp/build/x64-software-konbentzioak?view=vs-2017]

C Programazio Lengoaia

ÿ [c_king] [KNKing](http://knking.com/books/c/) [http://knking.com/books/c/] C programazioa, ikuspegi moderno bat WW Norton 2. Ed. 2008.

Programak garatzeko tresnak

ÿ [i386] [AS i386](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]: sintaxia, mnemoteknia, erregistroa

ÿ [GNU] [GNU Software Garapena](https://www.gnu.org/manual/manual.html) [https://www.gnu.org/manual/manual.html]

ÿ [GAS] [GNU Assembler](http://sourceware.org/binutils/docs-2.31/as/index.html) [http://sourceware.org/binutils/docs-2.31/as/index.html]

ÿ [GDB] [GDB arazketa](https://www.gnu.org/software/gdb/documentation/) [https://www.gnu.org/software/gdb/documentation/]

ÿ [GCC] [GCC konpilatzailea](https://gcc.gnu.org/onlinedocs/gcc/) [https://gcc.gnu.org/onlinedocs/gcc/]

ÿ [CPP] [Aurreprozesadorea cpp](https://gcc.gnu.org/onlinedocs/cpp/) [https://gcc.gnu.org/onlinedocs/cpp/]

ÿ [BiU] [GNU binutils tresnak](http://sourceware.org/binutils/docs-2.31/binutils/index.html) [http://sourceware.org/binutils/docs-2.31/binutils/index.html]:as,ld,objdump,...

ÿ [VIM] [Vim](https://www.vim.org/docs.php) [https://www.vim.org/docs.php]

ÿ [EMACS] [Emacs](http://www.gnu.org/software/emacs/#Manuals) [http://www.gnu.org/software/emacs/#Manuals]

Artikuluak

ÿ [jvn] [linuxvoice](https://www.linuxvoice.com/john-von-neumann/) [https://www.linuxvoice.com/john-von-neumann/]

ÿ [w_uni] [wisconsin unibertsitateko RTL](http://www.cs.uwm.edu/classes/cs458/Lecture/HTML/ch04.html) [http://www.cs.uwm.edu/classes/cs458/Lecture/HTML/ch04.html]

X Glosarioa

Lehen planoa

Dagokion definizioa (koskatuta).

Bigarren agintaldia

Dagokion definizioa (koskatuta).

Kolofoia XI

Liburu baten amaierako testua, bere ekoizpenari buruzko gertaerak deskribatzen dituena.