



Konputagailuen Arkitektura (KA)

2015–2016 ikasturteko programa

Taldea: euskara

Irakaslea: Natxo Martín

Aurreikusitako egutegia (1.1 gela):

Astelehena 10:45 - 12:15	Asteartea 12:30 – 14:00	Ostirala 9:00 - 12:15
Irailak 7: 1. gaia: CM (teoria)	Irailak 8: 1. gaia: Teoria	Irailak 11: 1. gaia: Teoria
Irailak 14: 1. gaia: Teoria	Irailak 15: 1. gaia: Ariketak	Irailak 18: 1. gaia: Ariketak
Irailak 21: 1. gaia: Ariketak	Irailak 22: 2. gaia: PS (Teoria)	Irailak 25: 1. gaia: Ariketak
Irailak 28: 2. gaia: Teoria	Irailak 29: Azterketa: 1 gaia	Urriak 2: 2. gaia: Teoria
Ordutegi trinkoko astea. Eskola: urriak 8, osteguna C: laborategia		
Urriak 12: 2 gaia: Ariketak	Urriak 13: 2. gaia: Teoria	Urriak 16: 2. gaia: Ariketak
Urriak 19: 2 gaia: Ariketak	Urriak 20: 2. gaia: Ariketak	Urriak 23: 2 gaia: Ariketak
Urriak 26: 2 gaia: Teoria	Urriak 27: C: jarraipen testa 2 gaia: Ariketak	Urriak 30: 2. gaia: Ariketak
Azaroak 2: 2. gaia: Ariketak	Azaroak 3: 2. gaia: Ariketak	Azaroak 6: 3 gaia: Paralelismo (POI)
Azaroak 9: 3. gaia: OpenMP	Azaroak 10: 3. gaia: Puzzle	Azaroak 13: 3 gaia: Puzzle (adituak)
Ordutegi trinkoko astea. Eskola: azaroak 19, osteguna Azterketa: 2 gaia		
Azaroak 23: 3 gaia: Puzzle (taldea)	Azaroak 24: 3. gaia: Puzzle (taldea) 3. gaia: aplikazioaren aurkezpena	Azaroak 27: 3. gaia: Puzzle (aurkezpena + debate)
Azaroak 30: 3. gaia: Puzzle (ariketak)	Abenduak 1: 3. gaia: lan-pertsonala	Abenduak 4: 3. gaia: lan-pertsonala
Abenduak 9 (astelehena): 3 gaia: jarraipen testa	Abenduak 8: Jai	Abenduak 11: 3. gaia: lan-pertsonala
		Abenduak 18: 3 gaia: txostenaren jasotzea

Ebaluazioa.

Ikasgaiak bi ebaluazio mota izango ditu: ebaluazio jarraitua eta ebaluazio globala (bukacean). Ebaluazio landutako gaitasun guztiak ebaluatzeko. Azterketaren traupena 3,5 ordu inguru izango da eta atal hauek izango dira: cache memoria (2,5 puntu), prozesadore segmentatuak (3,5 puntu) eta parallelismoa (4 puntu). Azterketak urtarilaren 21ean eta ekainaren 17an izango dira. Azterketa gain, ikasgia gainditzezko, ikasleak praktika bat egin beharko du 3. gauan (parallelismoa). Praktika azterketa baino lehen entregatu beharko da, irakasgaien eGela-n arginatuko den datanak.

Ebaluazio jarraituan, ikasgia ebaluatzeko jarduera hauek kontuan hartuko dira (batzuk individualak eta beste batzuk kooperatiboak):

a) Ebaluazio jarraituari dagozkion proba idatzia (notaren %70a).

Proba hauetik hiru motakoa izango dira: azterketa partzialak, jarraipen probak eta arketen ebaezpenak. Alde batetik, bi azterketa partzial egingo dira: irailaren 29an, lehenengo gaiari buruz (cache memoria), eta azaroaren 19an, bigarren gaiari, buruz (prozesadore segmentatuak). Lehenengo azterketak 2 puntu balioko du, bigarrenak, betri, 3 puntu. Beste aldetik, bi jarraipen-kontrol egingo dira. Batetik, C lengoia ikasteko egindo den laborategia azterketa batzen bidez ebaluatuuko da eta, bestetik, hirugarren gaiari kontrol bat egingo da proiektuarekin lotuta. Lehenengoa urtaren 27an izango da eta 0,5 puntu balioko du. Bigarrena abenduaren 9an izango da eta puntu bat balioko du. Bigarren horretan, ikasleak nota minimo bat artea beharko du proiektua gaintzeari: %63a alegia. Azkenik, bigarren gaiari ikasleek arketeta baten ebaezpenea entregatu beharko dute. Arketeta horren ebaezpenak 0,5 puntu balioko du.

b) Lan idatzia (notaren %20a)

Ikasleek proiektu bat egin beharko dute, hiriko taldean, azkeneko gaiari (parallelismo). Proiektu honetan aplikazio paralelo simple bat programatu beharko dute OpenMP erabiliz. Proiektuaren kostoen teknikak eta proiektuaren karpetak 1,5 puntu balioko dute guztiak. Txostenen abenduaren 18an entregatu beharko da; karpetan, berriñ, urtarilaren 21ean jasoko da. Beste aldetik, ikasleek jarduera bat egin beharko dute lehenengo gaiari (cache memoria). Jarduera hori taldeka egingo da eta 0,5 puntu balioko du.

c) Ikaslea jardueretara etorzea eta parte hartzea (notaren %10a).

Ikasleek proiektuan egindako lana aurkezten: puzzle bukatzerakoan eta proiektu guzta bukatzerakoan. Aurkezpenak azaroaren 27an eta urtarrilaren 21ean egingo dira, hurrenez hurren. Aurkezpenek puntu bat balioko dute guztiak.

Ikasleak aukerantxo du interesatzen zaien ebaluazio mota, ebaluazio jarraitua ikaslearen parte hartzeko aktiboa eskatzen duela kontuan hartua (ikastek eskolaera eta labortegietara etorri beharko du, hainbat parte hartz, eta proposatutako jarduerak egin beharko ditu, salbuesperek salbuesperekin). Ebaluazio jarraitua ikastaroaren lehenengo hilera aukeratu ahal izango da, eta behin betiko bihurtuko da bigarren azterketa partziala ebaliatu eta gero, irakasleak ikaslearen errendimendua egiazatu ondoren. Ebaluatzeko aukera behin konfirmatuta, ikaslek ezin izango du ebaluazio globalera pasa. Epe horretan ikasleak ez badu eskaerrik egin, ebaluazio jarraitua iku egitzen dia la ulertzeko da.

► Edozein ebaluazio mota aukeratuta, irakasgaien gainditzezko ikasleak nota minimo bat atera beharko du irakasgiaren hiru zatietan (cache memoria, segmentazioa eta parallelismoa): atal bakoitzean %30a atera beharko du.

Gai bakoitzeko dedikazioa eta atazak.

Gai	Ekintzak	Eskolak	Lan pertsonala	Puntuazioa
Cache memoria	Teoria + arirketak Ebaluazioa	1,5 h	21,5 h	2,5
Segmentazioa	Teoria + arirketak Ebaluazioa	3 h	31 h	3,5
Parallelismoa	Teoria + arirketak I Laborategia (C) Ebaluazioa	13,5 h 5 h 4 h	38 h	3,5 0,5
Guztia	59,5 h	90,5 h	10	

Bestelako informazioa.

1.- Ikasgaiaren web oria:

<https://cgela1516.chu.es>

2.- Irakasleei buruzko informazioa:

Bulegoak:
Nabko Martín: 218
Olaz Arregi: 252
Olaz Arbelaitz: 219

Hebilde elektronikoak:
jmartin@ehu.eus
olatz.arregi@ehu.eus
olatz.arbelaitz@ehu.eus

1. Iauhileko tutorelta-orlutzegia: ikus ikasgaiaren web oria



Konputagailuen Arkitektura

Informatika Ingeniaritzako Gradua

2015-2016 ikasturteko programa

Irakasleak: Nekane Martín (euskara), Olatz Arregi (gaztelera) eta Olatz Arbelaitz (ingelesez)

6 kreditu = 4 T + 2 P

Edukia.

Konputagailuen Egitura ikasgaien von Neumann motako konputagailuen oinarrizko egitura aztertu egin da. Hordik abiatuta, ikasgi honean, konputagailuen prozesatzeko-abiadura hobetzeko dauen teknika nagusiak azalduko ditugu.
Lehen zatian, cache memoria, hau da, datuak eta aginduak azkarraigo azitzeko erabilten den memoria izango dugu aztertza. Bigarrenean, prozesadoren arkitektura segmentatuaz aztertuko dugu, hala hardwarea nola programak eraginkorrik exekutatzeko erabilten diren oinarritzko konplazio-teknikak. Azkenik, hirugarren zatian, prozesadore asko erabilten dituzten sistema paraleloen ezagunari nagusiak azalduko dirugu.

Gaitasunak.

Ikasgaiaren gainditzu ahala izateko, ikasleak guiciskin lortutako hainbat gaitasun espezifiko lortu beharko ditu; hau da, honakoak egiteko gai izan beharko du:
1. Cache memoriaien diseinuanparametroak ulertu.
2. Programen exekuzioan cache memoriai duen eragina aztertu.
3. Prozesadore segmentatuuen funtzionamendua ulertu. Bloke funtzionalekin egindako disiniu bat ulertzeko eta jarraitzezko aldatuak izan.
4. Prozesadoren eraginkorrasuna handitzeko crabitzen diren oinarritzko optimizazioak ulertu.
5. Behar- eta goi-mailako paralelismoa ezagatu.
6. Parallelismoa erabili aplikazio simple batzuk programatzeko, datu-dependientziak, sinkronizazioa era arazen banaketa kontuan hartuta.

Horietaz gain, ikasgaien dagozioin gaitasun partekatuak landuko dira. Hauek, ikasgaiaren GAIUR web gunean konsultatu daitezke.

Irakaskuntza metodologia.

Ikasgaiaren lehenengo bi gaiak bi jarduera moten bidez irakasten dira: teoria estolak eta artiketa eskolak. Irakaskuntza prozesuan metodologia aktiboak eta ikasketa kooperatiboa erabiliko dira: talde-lanak egongo dira, artiketen cbazpenak denon artean eztabaeditako dira, etab. Helburua ikasleen motibazioa eta parte hartzea handitzea da.

Beste alde batetik, hirugarren gela Projektuaren Oinarriutako Irakuntza metodologia erabiliko da. Gaiaren edukiak projektu baten inguruas landuko da. Horretarako ikasleak 3 kidetxo taldeetan artikulo dira. Praktiken emaitzak softwareko aplikazio bat izango da, dagokion dokumentazioa ere kontuan hartuta.

Gaitegia:

1. CACHE MEMORIA.
 - 1.1. Sarrera. Memoria-hierarkia.
 - 1.2. Ezagutu orokorak: tamaina, edukia eta blokea.
 - 1.3. Diseinu-parametroak: egitura, ordezkapen-algoritmoa eta idazketa-politika..
2. PROZESAORE SEGMENTATUAK.
 - 2.1. Motibazioa: prozesadorearen eraginkortasuna handitu.
 - 2.2. Oinarritzko prozesadore segmentatu batzen disiniua: DLX prozesadorea.
 - 2.3. Datu- eta kontrol-dependentziak.
 - 2.4. Ziklo anitzeko segmentazioa eta prozesadore supereskalatarrak.
 - 2.5. Konplazio-teknikak prozesadore segmentatuera.
3. KONPUTAZIO PARALELOKO SISTEMEN OINARRIAK.
 - 3.1. Flynn-en sailkapena. Eraginkortasuna.
 - 3.2. Partekatuko memoria koplaztutako paraleloak: prozesuen sinkronizazioa eta arazen banaketa.
 - 3.4. Multiprozesadoren programazioa: OpenMP.

Bibliografia:

1. Hennessy J.L., Patterson D.A.: *Computer Architecture: A Quantitative Approach* (4. ed.). Morgan Kaufmann, 2007. [Konputagailuen arkitektura. Hurbilkera kuantitatibo bat, UPV/EHU, 2009]
2. Patterson D.A., Hennessy J.L.: *Computer Organization and Design: The Hardware/Software Interface* (4. ed.). Morgan Kaufmann, 2008.
3. Ortega J., Anguita M., Prieto A.: *Arquitectura de Computadores*. Thomson, 2005.
4. Hennessy J.L., Patterson D.A.: *Arquitectura de Computadores: un Enfoque Cuantitativo* (1. ed.). McGraw-Hill, 1993
5. Patterson D.A., Hennessy J.L.: *Estructura y Diseño de Computadores*. Reverté, 2000.
6. Stallings W.: *Organización y Arquitectura de Computadores* (7. ed.). Prentice Hall, 2006.
7. Almeida F., Giménez D., Mantas J.M., Vidal A. M.: *Introducción a la programación paralela*. Paratinfo, 2008.
8. Culler D.E., Singh J.P.: *Parallel Computer Architecture. A Hardware/Software Approach*. Morgan Kaufmann, 1999.
9. Chandra R., et al.: *Parallel Programming in OpenMP*. Morgan Kaufmann, 2001.
10. Aldizkariak eta fabrikatzileen web orriak: IEEE Computer, IEEE Micro, BYTE, www.top500.org, ...

Karte Memorie: Arithmetik

Kaufkognitiven
Arithmetik

1.1

8K_b+8K_b-elko Cache-memoriell hartsen bedingung

Agrindu-cache, 90% -lo hutsigite-hesa do, etc Datv.-cacheell %6-Uva.

Ordea 16K_b-elko hartsa bellerre hartsa 86%-lo hutsigite-hesa izango

Dv.

c. Banatvarren hutsigite-hasaren batzbestelloa: $0'6 * 6 + 0'4 * 7 = \%6'4$

C. Memoria \rightarrow 8 hitz edukiere

$\left\{ \begin{array}{l} 4 \text{ hitz} \rightarrow 2 \text{ bloke} \\ 2 \text{ hitz} \rightarrow 4 \text{ bloke} \end{array} \right.$

a) A[0,0] A[0,1] ... A[0,3], A[1,0] ...
_{hitz} _{asun}

a1) Hutsigite-hesa:

$$h = \%50$$

asun-hesa

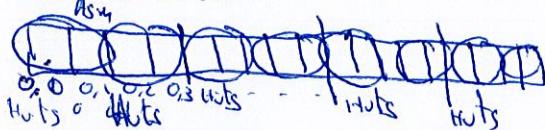
hutsigite-egitean bloke osoa partible diogu: 2 hitz

$$a_2) h = \%25$$

hutsigite-egitean bloke osoa partille da: 4 hitz

b) A[0,0] A[1,0] A[2,0] A[3,0] A[0,1]

$$b1) h = \%50$$



$$b2) h = \%25$$

82 bloke gorde aldi direnes, gorde behar ditugun lehendak geltzegi direneka beti huts egungo du.

1.3

a)

Hütscigite-hose Händler die Doku-bloke tomima bettili aurrez normale
de ingurutakoa apurte egiten baitz C.Mem-er c2 kitz deko händler

b)

$$T = h \times t_{cm} + (1-h) \times t_{wde}$$

$$1 \text{ hitze} \rightarrow T_a = 0.5 \times 1 + 0.5 \times 11 = 6.8$$

$$2 \text{ hitze} \rightarrow T_a = 0.34 \times 1 + 0.36 \times 12 = 4.968$$

$$4 \text{ hitze} \rightarrow T_a = 4.512$$

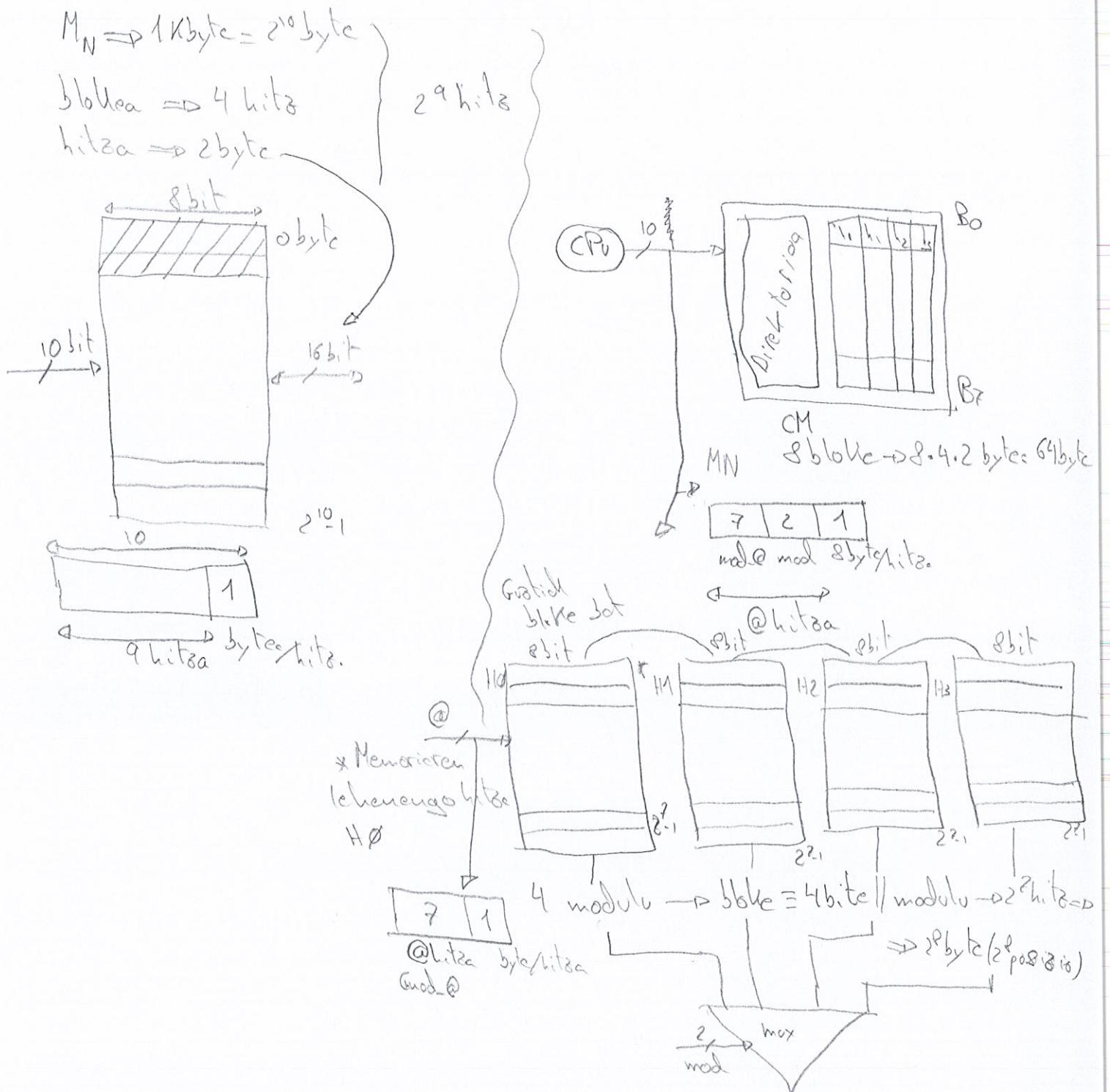
$$8 \text{ hitze} \rightarrow T_a = 4.742$$

$$16 \text{ hitze} \rightarrow T_a = 6.2$$

;

adioroz, 4 hitzeko bloke de ~~ganz~~ oso leku

Kaputagaiuen Arkitektura



Aldi berean bloke batzko hitzak giztioak sailkarriz behar dira.

Cache Memory → Bf.

00	10		2
01	10		6
10	10		10
11	10		14
@Mark,			

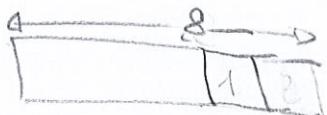
1. Gaia's Cache Memory Architecture

Komputagailuen
Arkitetutura

1.9

a)

$M_n \rightarrow \text{Tasainne} = 256 \text{ MB} = 2^8 \cdot 2^{20} = 2^{28} \rightarrow 28 \text{ bit helbidea}$



1. Gaia: Arithmetik

Komputagaien Architektur

1.7

a)

$M_{main} \rightarrow$ Memoria magisca \rightarrow Tamaño = 1 Mb = 2^{20} byte \rightarrow 20 bit adreßable



@Märke bit bytes

hitze

M_0

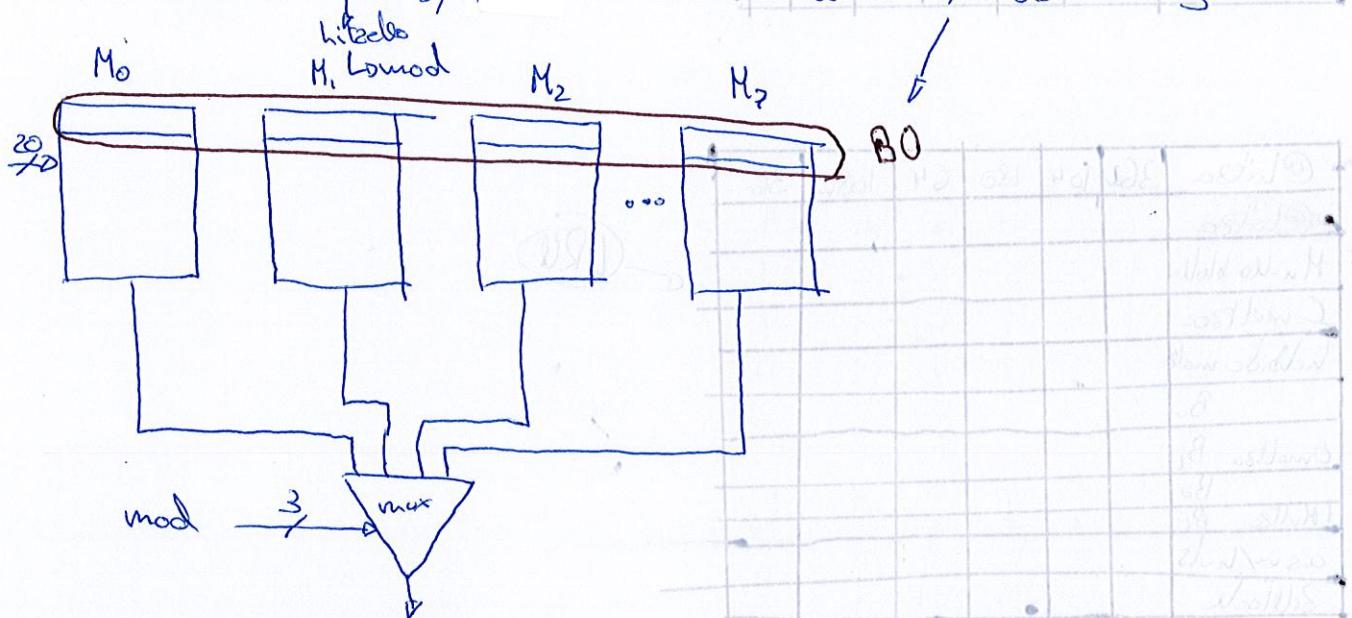
M_1 Lomod

M_2

M_3

B0

8 bitsAdresse, 8 Blöcke je 800 Bytes



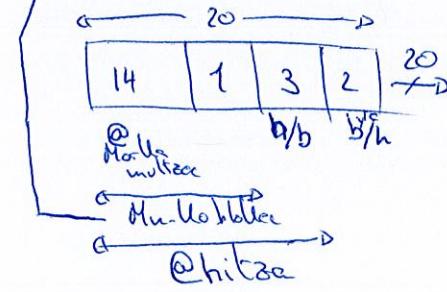
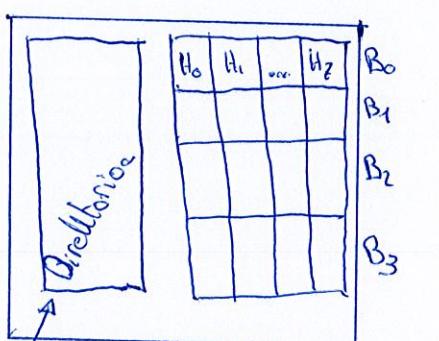
$$T_{Blöcke} = 10_2 + 1 + \dots + 1 = 17_2$$

Lebenszeit Blocktakt

modulwahl 10 2:100 Bet.

bitlo

$M_C \rightarrow$ Cache Memoria \rightarrow Tamaño = 4 bloques = $2^2 \cdot 2^3$ bytes = 2^5 bytes = 2^5 bytes $\cdot 2^2$ = $= 2^7$ bytes \rightarrow 7 bit.



b)

	36w	164	180	64	168w	36
@bytem	9	26	45	16	27	9
@hitza	1	3	5	2	3	1
Multiblock	1	1	1	0	1	1
C multiblock	1	1	1	0	1	1
helvide-werk	0	1	2	1	1	0
B ₀				2	2	2
Omultiblock B ₁						
B ₀	1	1	5	5	5	5
Imultiblock B ₁		3	3	3	3	1
asm/hvts	H	H	H	H	A	A
Zillblock	1+17	1+17	1+17	1+17	1	1+17

FIFO

S.J

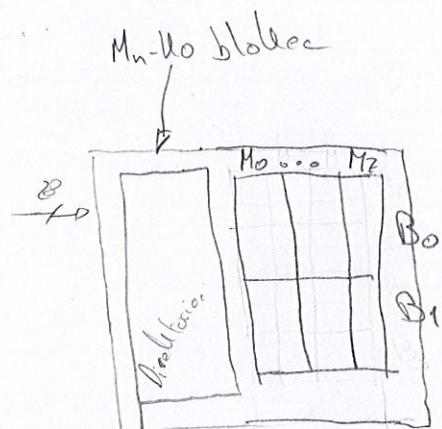
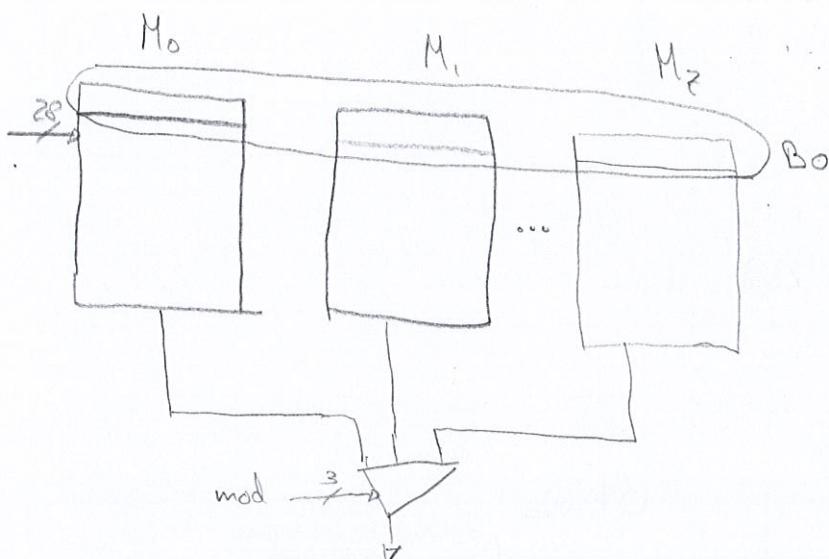
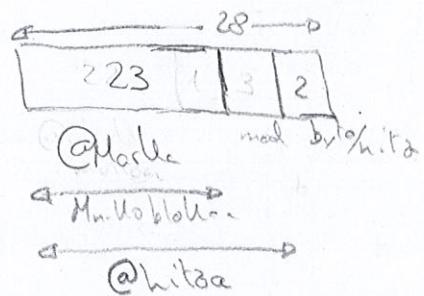
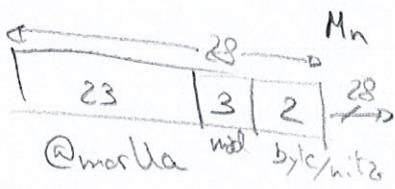
	36w	164	180	64	168w	36
@bytem	9	26	45	16	27	9
@hitza	1	3	5	2	3	1
Multiblock	1	1	1	0	1	1
C multiblock	1	1	1	0	1	1
helvide-werk	0	1	2	1	1	0
B ₀				2	2	2
Omultiblock B ₁						
B ₀	1	1	5	5	1	1
Imultiblock B ₁		3	3	3	3	1
asm/hvts	H	H	H	H	A	A
Zillblock	1+17	1+17	1+17	1+17	1	1

LRU

$$\text{S1} + \overbrace{\text{F} + \dots + \text{F}}^{\text{1 row}} + \text{S0} = \text{overall}$$

↓
1 row
1 row of 6 segments
1 row of 6 off-blocks
1 row of 6 on-blocks

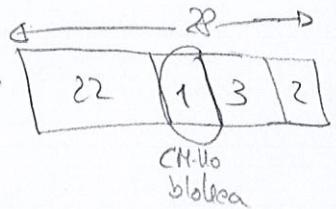
19



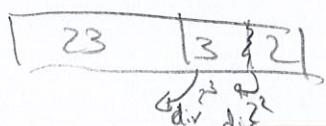
Cache-Memoria sare tamaina 32 bytekoak da, eta Mn-ko Ein-rentamaike berri da. Ordioriaz, tamaina = 32 byte eta hitzak 4 byte

$$\text{Blokkak} = \frac{32 \text{ byte}}{4 \text{ byte}} = 8 \text{ hitz}$$

Cu 8 blokkak bada adatu egingo da hau cache memoria.

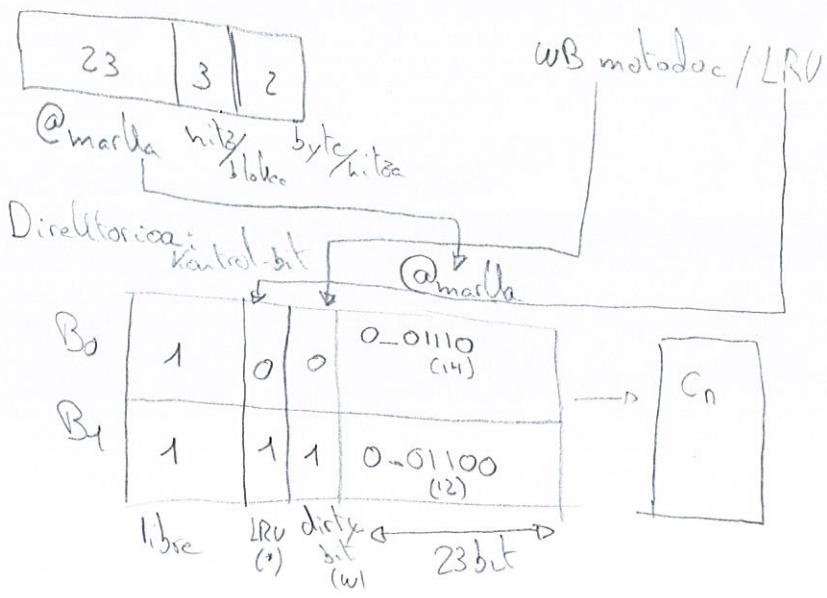


Hitzak hitz lotzeko dir eragikete egiten da.

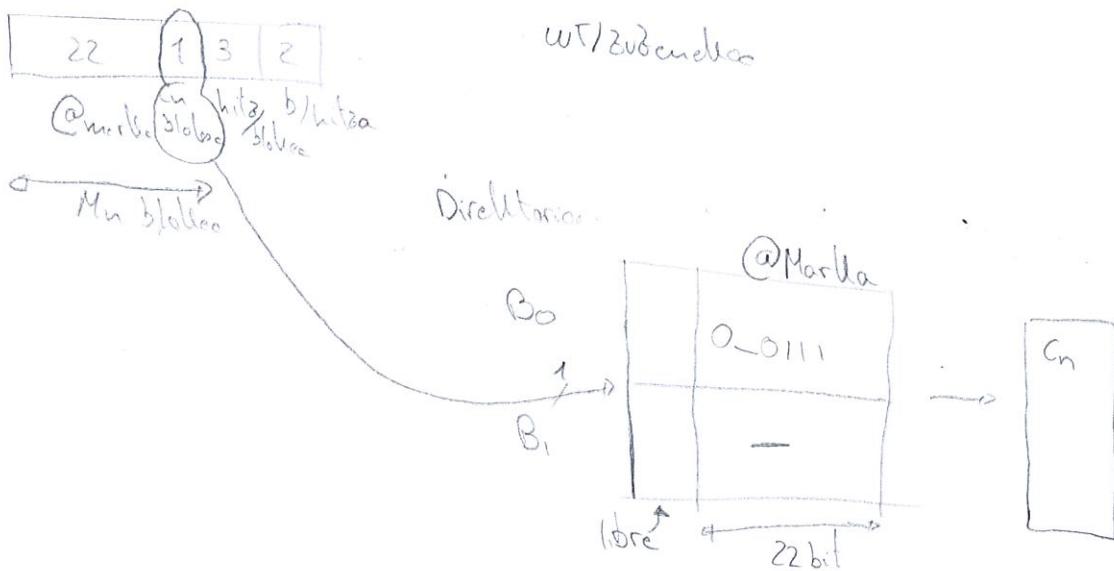


Kosu honetan @marka edo Mn-blokkak betetze da.

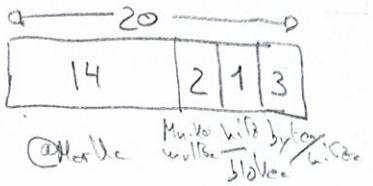
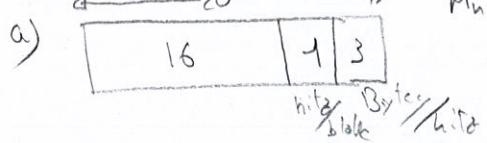
9.a



b)



1.10



b)

$$M_d \text{ blokken} = 3 \cdot 2^2 = 12 \Rightarrow 1 \cdot 2^2 = 4; 7 \cdot 2^2 = 28; 4 \cdot 2^2 = 16$$

+ Händler + Händler + Händler

c) libredirty @Händler

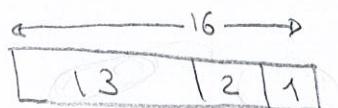
B ₀	1	1	0...0011
B ₁	1	0	0...001
B ₂	1	0	0...0111
B ₃	1	0	0...1100

d)

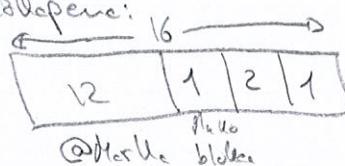
$$T_a = 0'98 \times 1 + (0'02) \times (0'75 \cdot 16 + 0'25 \cdot 31) = 1'375 \text{ Millio}$$

1.2

- M_d , modulu tastatavetan autolatua:



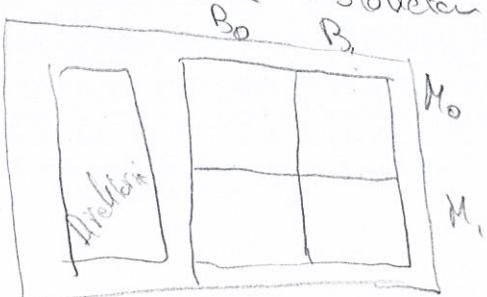
- CM, 4 blokke, write-back idablate, eto LRU ordedlopene:



e)

Memoria Nagusia 4 blokketan dego bantut. eto 64KB -eko tamaina du,

Cache Memoria 4 blokketan dego ere bantut.



Habideretze-unitzeta: bylee dc.

Bitzen tamaina: 2 byte

Blokketamain.: 4 hitz

5)

@bxtea	6w	28	48	8w	30	36	38w	0	42	30	8	16
@hit8a	8	14	24	9	15	18	19w	0	21	15	4	8
MHka Slab	2	30	6	2w	3	4	4w	0	5	3	1	2
Cochella multico	0	1	0	0	1	0	0	0	1	1	1	0
Velarde- nerville	1	1	3	8	1	2	2	0	2	1	0	1
O: multico	2w	2w	2w	2w	2w	2w	3w	0	0	0	0	0
				6	6	6	4	4w	4w	4w	4w	2
L multico	B	3	3	3	3	3	3	3	3	3	3	3
				2					5	5	1	1
ASW/Luts	H	H	H	A	A	H	A	H	H	A	H	H
2WB	1+12	1+12	1+12	1	1	1+12	1	1+12	1+12	1	1+12	1+12
fill load (wt)	1+12	1+12	1+12	1+	1	1+12	1+12	+	1+12	1+12	1	1+12

$$T_{WB} = 132 \text{ kN}$$

$$T_{WT} = 144 \text{ kN}$$

1.4

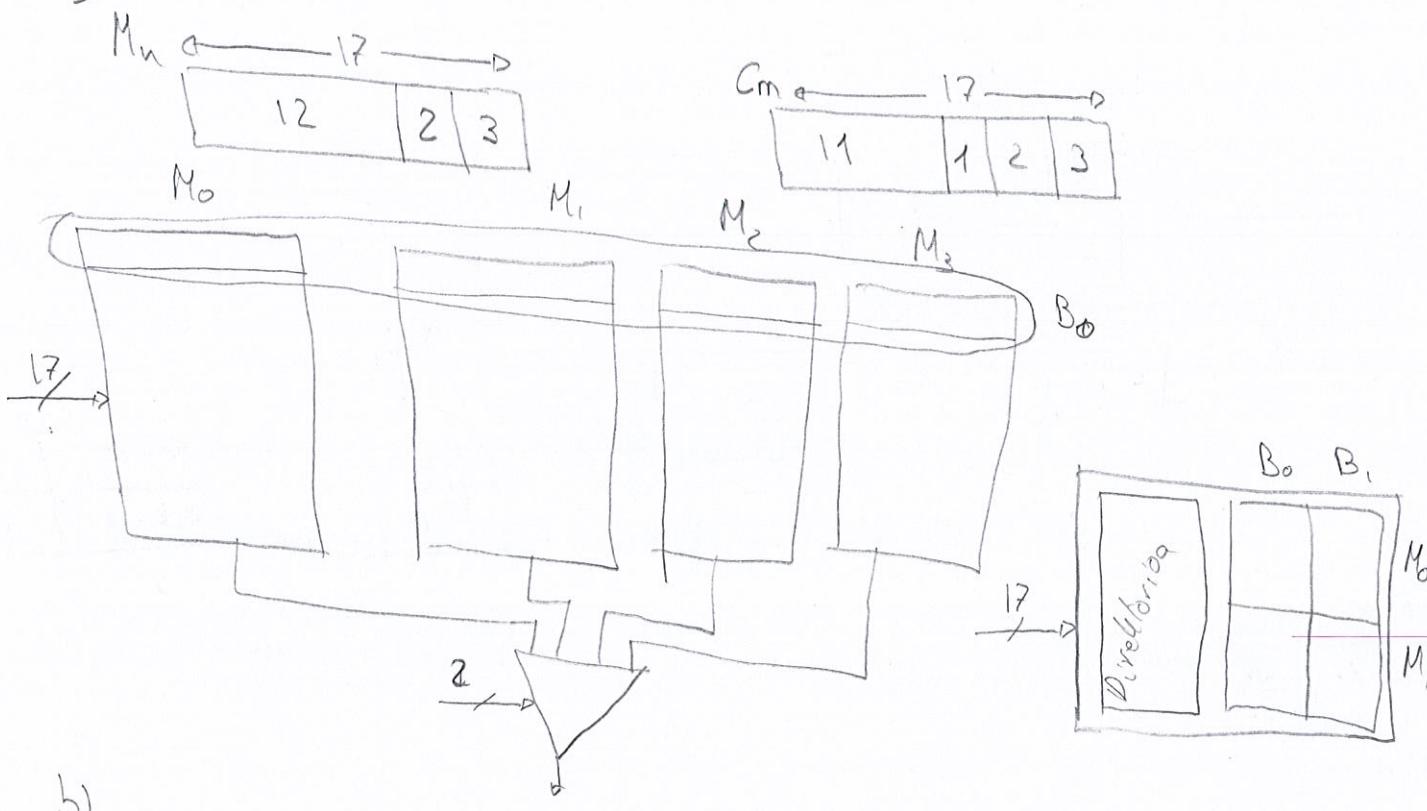
Hebbideratze-unitatea : bytea

Hitōale: 83 yte

Bloke: 4 hstz

$$\text{Tarnende } M_n = 128 \text{ kB} = 2^{17} \Rightarrow 17 \text{ bit} \quad \text{Atziffer - Jederbit} = 8 \text{ kB}$$

a)



@byfca

c)

$$T_{WB} = 121 \text{ g/Mo}$$

$$T_{WT} = 132 \text{ g/Mo}$$

16

Hellbide fisilico: 8bit

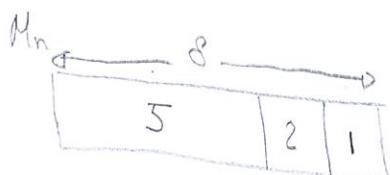
Hellbideratoe - Undatoea: bytca

Hitzale: 2bytes

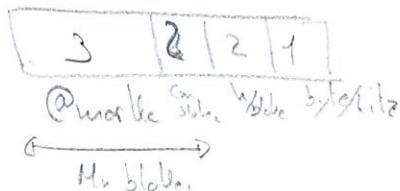
Blakale: 4bits

Datu - cachea: 25 bloketao

a)



Cm



b)

@bytca	118
@hitca	59
Mn Multao	14
@marke	3
	1

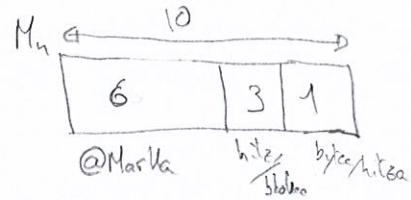
$$59 = 00111011$$

Ondorioz hitseguiles izango da cachean ez baita aldi bereko hellbiderik

Aritetza

1.8

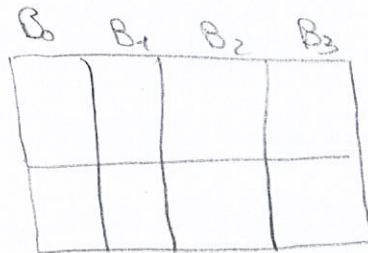
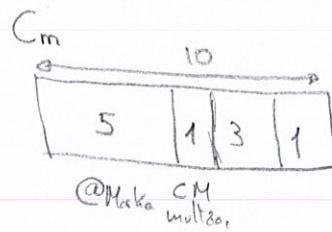
$$M_n \text{ tamaina} = 1024 \text{ blokeak} = 2^{10}$$



libre aldatua LRU tag

No	1	0	01 10	00101
	1	0	00 01	00110
	1	0	10 X	10001
	1	1	11 00	10101
	1	-	--	-----
M_1	1	0 X 1	10 00	00101
	1	1	01 10	00010
	1	1	00 01	00001

hebitide-motako multzoa
 $\rightarrow 0010110^2 = 001010 = 10_{10}$
 $\downarrow \text{div } 2^1 = 00101$



c) $t_{ewb} = 1_3 // t_{cut} = 1 + 10_2 //$

182) Albitzak
 91) @Marka
 11) Muho blokeak
 5) @Marka

Muho blokeak bedugu cache Memoriaan. = 11,

Dirry bit-a aldatua da 1 eko bit jarriz.

Eta LRU ere, Azkena erabiltzen aldatu egungo detalloa.

d)

684
 342
 42
 21

Ez da agertzen cache Memoriaan

$$42 \bmod 2^1 = 0$$

Andorioak, 0 blokeen idatzia behar da.

e)

Ideale - politika

Atpen - denbarak:

Iralurketaak

$$\text{Asmetzaa: } t = t_{cm} \quad \text{Hutsegitea: } t = t_{cm} + t_{bt}$$

$$\text{Hutsegitea + ordelepene + WB(dirty): } t = t_{cm} + t_{bt} + t_{bt}$$

Idealeta

WB

$$\text{Asmetzaa: } t = t_{cm} \quad \text{Hutsegitea: } t = t_{cm} + t_{bt}$$

$$\text{Hutsegitea + Ordelepene: } t = t_{cm} + t_{bt} + t_{bt}$$

WT

Mu eguneratua

$$\text{Asmetzaa: } t = t_{cm} + t_{mu} \quad \text{Hutsegitea: } t = t_{cm} + t_{mu}$$

Mu de cm eguneratua

$$\text{Asmetzaa: } t = t_{cm} + t_{mu} \quad \text{Hutsegitea: } t = t_{cm} + t_{mu} + t_{bt}$$

1.10

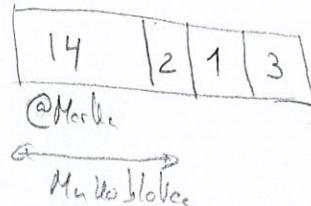
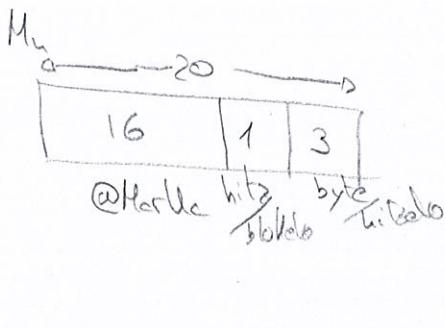
a) Memoria naugusia \Rightarrow tamaña = 1 MB = 2^{20}

Habideratze-unitatea = bytea

bitza = 64 bit = 8 byte

Blocka = 2 bits

Cache memoria \Rightarrow 64 byte \Rightarrow 8 bytes



b)

3, 1, 2 etc 4

$$3 \Rightarrow 3 \cdot 2^2 = 12 \text{ Multiblock}$$

$$1 \Rightarrow 1 \cdot 2^2 = 4 + 1 = 5$$

$$7 \Rightarrow 2 \cdot 4 = 28 + 2 = 30$$

$$4 \Rightarrow 4 \cdot 4 = 16 + 3 = 19$$

@bytea	-	200w	508w	1023	1022
@hitza	-	25	63	127	127
Multiblock	-	12w	31w	63	63
Efectuatz	-	3	7	31	31
B ₀	12	12w	12w	12	12
B ₁	5	5	5	5	5
B ₂	30	30	31w	63	63
B ₃	19	19	19	19	19
Asy/Its	-	A	H	H	A
Liklikas(w)	-	1+B	1+B+15	1+15	H1
Zliklikas(wB)	-	W	1+B	(H+B)+(B)	1

1.9

$\text{Memoria negusis} \Rightarrow \text{tamaña} = 256 \text{ Mb} = 2^{38} \text{ bit}$

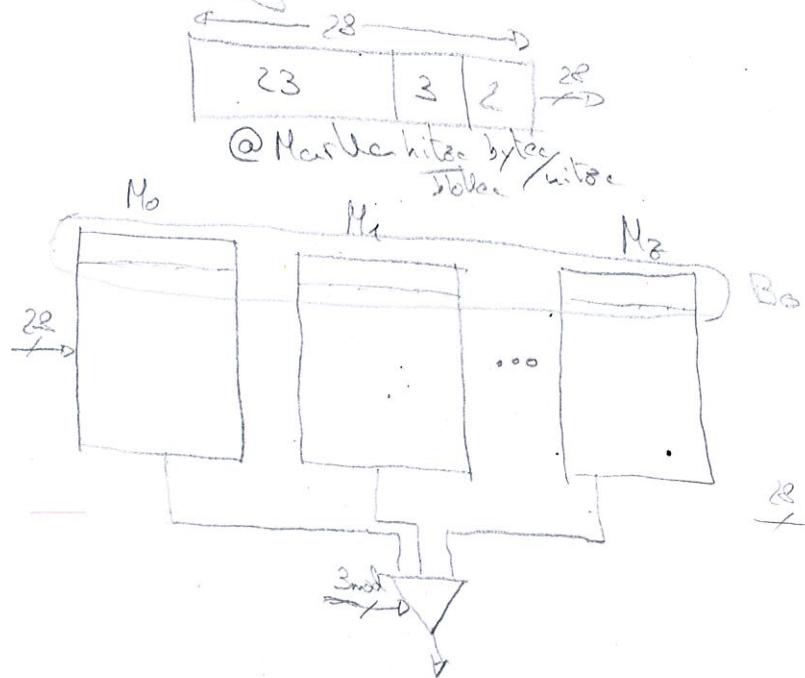
Helioderatae - unitatea = bytea

Hitzahl = 4 byte

$$\text{Data-cache} \Rightarrow 32 \text{ bytes}, 2 \text{ blocks} \Rightarrow \text{Block Length} = \frac{32 \text{ byte}}{4 \text{ byte}} = 8 \text{ bits}$$

a)

Mn - capture



b) WB, qualitatively allergenic

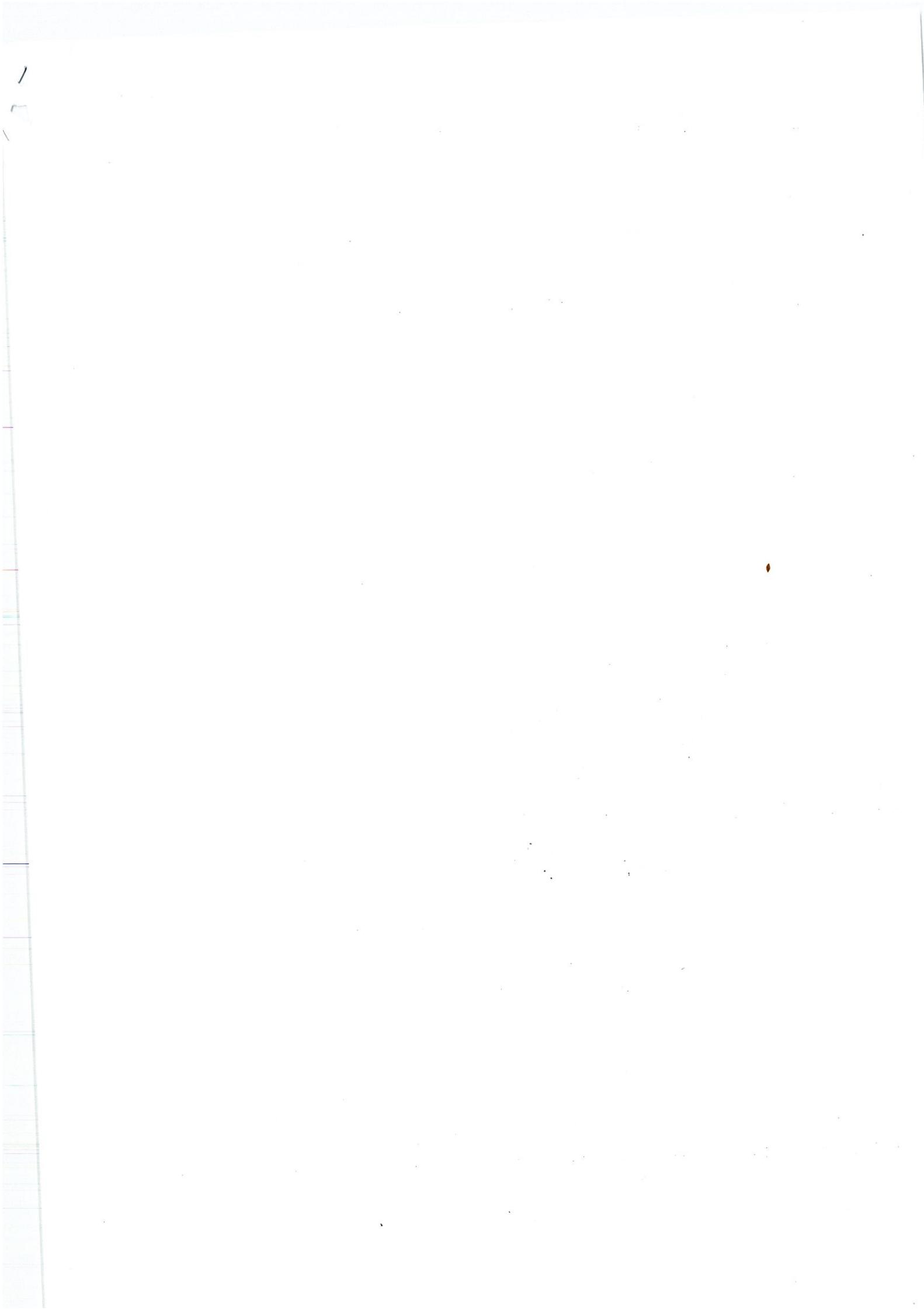
wrote through Mr. Ignatius, Bishop of

@bytee	300w	400	404w	464
@hitzza	200w	100	101w	116
@Markie	25	12	12w	14
Mn Slovec	25w	12	12w	14
B ₀	25w	25w	25w	14
B ₁		12	12w	12w
Asymtots	H	H	A	H
2. Blanks	2+15+ ?	2+15+ ?	2	2+22+
				22

	400w	200	204w	464
	100w	50	51w	121
	6w	3	3	7
	18w	6	6w	15
	6	-		
A.	.	A	H	
2+15	2+15+	2+15	2+22	

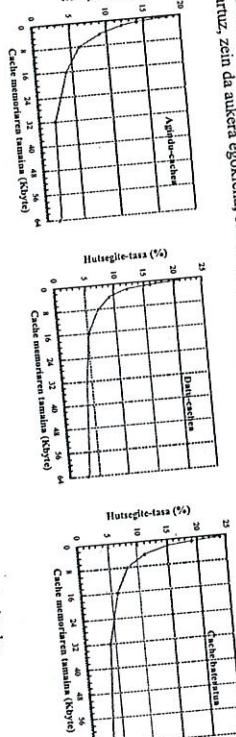
Highest density = 96 zillio

Alziger-Sempore = 828 M.





- 1.1.** Irdiko grafiketan, konputagailu jakin baten cacheko azipeneko hutsigite-tasa ageri da, cachearen edukien arabera, bi kasu hauekarako: (a) agindu-cachea eta datu-cachea banatuta dade; eta (b) cache bakarra dago, danustarko eta agindu-detrako. Atzipenetan %60 aginduak eskaratzeko izan dira. Guzitza 16 kB-ko cache memoria bat diseinatu behar da konputagailu horretarako. Grafioko datuak kontuan hartuz, zein da aukera egokiena, cacheak banatuta edo bateratuta izatea? Zergatik?



- 1.2.** Konputagailu baten memoria-helbideak 16 bitekoak dira, eta eremu hauean banatuta dade:

[13 bit – 2 bit – 1 bit]

- MN, modulu tarteakutuetan antolatua:

- CM, 4 bloke, write-back-idaiezka, eta LRU ordezkapena: [12 bit – 1 bit – 2 bit – 1 bit]

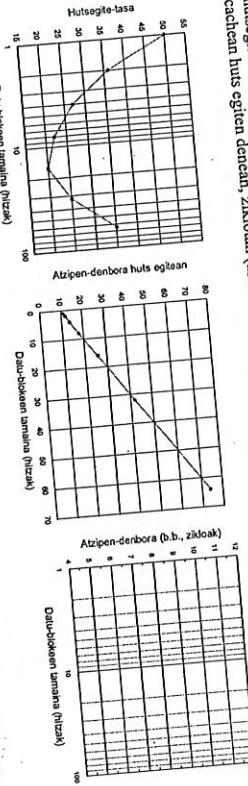
- Atzipen-denborak honako hauek dira: MN, 9 ziklo (ziklo bat tarteakutze-bufferretik). Aita, cache memoria? Zein da helbideratze-unitatea? Eta a. Zein da memoria nagusiaaren egitura? Eta cache memoria?

- b. Kontuan hartu erreferentzia-segida hau, eta kalkula ezazu atzipen-denbora (osoak): 16w, 28, 48, 18w, 30, 36, 38w, 0, 42, 30, 8, 16 (w = write, idazketa).

- c. Idaiezka-politika write-through batziz, zer aldatuko litzateke lehenengo erreferentziaren exekuzioan?

- Kalkula ezazu erreferentzia horri dagokion atzipen-denbora berria.

- 1.3.** Cache memoria baten portarea analizatu egin da, eta honako bi grafiko hauek lortu dira. Lehenengoa, hutsigite-tasa blokearen tamainaren arabera agentzen dena, azipen-denbora ziklo bat da. hutsigite-tasa huts egiten denean, ziklotan (astunetan), cachean huts egiten denean, ziklotan (astunetan),



- a. Aztertu bi grafiokoak eta justifikatu ageri den portarea.
b. Bi grafiokoak kontuan hartuz, marratz ezazu memoria-sistema azitizko batez besteko denbora blokaren tamainaren arabera. Ondorioz, kalkula ezazu bloke-tamainariak egokiena.

Aurreko bi grafiokoan datuak

Datu bloken tamaina: 1 2 4 8 16 32 64
Hutsigite-tasa (b): 50 36 27 22 20 25 35

Multz. ekarg. M1 B1

Hutsigite-tasa (b): 11 12 14 18 26 42 74 (ziklo)

Batez besteko azipen-denbora:

- 1.4.** Prozesadore baten memoria-sisteman honako eragari hauek ditu:
Datu-blokeak 4 hitzekoak dira, hizkak 8 bytekoak dira, eta helbideratze-unitatea bytesa da. Memoriaren azipen-denbora 7 ziklo da (ziklo bat tarteakutze-bufferretik). Memoria nagusia 128 kB-koa da, eta tarteakutze-bufferretik. Cache memoria 128 bytekoak da, multzokako elkarlaria, 2 multzokoa. Ordezkapen-algoritmoa LRU da, eta idazketa-politika write-back. Atzipen-denbora ziklo bat da.

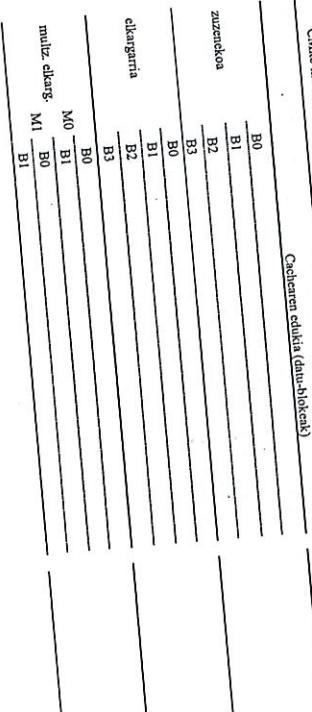
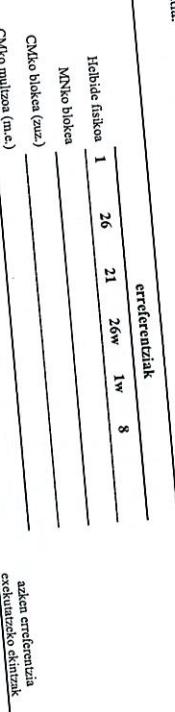
- a. Azal ezazu MNko eta CMko helbideen egitura (eremu guztiak). Kalkula ezazu memoria-azipenentako sejita honen exekuzio-denbora: 544, 10256, 10364, 528w, 10264, 520w, 1312, 532w, 1696, 896, 1312, 1738w. b. Kalkula ezazu memoria-azipenentako sejita honen exekuzio-denbora: 520w, 1312, 532w, 1696, 896, 1312, 1738w. c. Denagun idazketa-politika write-through dela. Kalkula ezazu berriro aurreko segidaren atzipen-denbora.



- 1.5.** Cache memoria batek 2 hitzeko 4 bloke ditu (hitza = byte). Memoria-helbideak 12 bitekoak dira eta helbideratze-unitatea bytesa da. Ordezkapen-algoritmoa LRU da.

- a. Adierazi helbide fisikoaren eremuak, hiru cache-egitura hauekarako: (1) zuzeneko, (2) guziz elkarlaria, eta (3) multzotako elkarlaria (elkarlaria fisikoa): 1, 26, 21, 26w, 1w, 8. Adierazi cachean kargatuko eta idazketa-politika write-back-idaiezka bat. Azken erreferentziatzarako, adierazi zer egun betar den eragiketa betezeko diren datu-blokeak eragiketa horriez exekuziatzen direnean, aurkezten egiturak kontuan harturik (betezko taula edo antezeko bat). Azken erreferentziatzarako, cachean zein memoria nagusia idazketa-politika (1) write-back edo (2) write-through (asmazean, cachean zein memoria nagusia idazketa-politika (1) write-back edo (2) write-through) batez besteko denbora dela idazketa (da) denean.

- b. Izan bedi erreferentzia-segida hau, eta kalkula ezazu exekuziak direnean, aurkezten egiturak kontuan harturik (betezko taula edo antezeko bat). Azken erreferentziatzarako, adierazi zer egun betar den eragiketa betezeko eta, cache memoria honetarako hobe dela write-back idazketa-politika write-through baino. Azal ezazu idazketa-politika (1) write-back edo (2) write-through (asmazean, cachean zein memoria nagusia idazketa-politika (1) write-back edo (2) write-through) batez besteko denbora dela idazketa (da) denean.
- c. Idatz ezazu erreferentzia-segida motz bat (10 erreferentzia edo) non ikusiko batza, exekuzio-denbora dela idazketa-politika write-through baino. Azal ezazu cache memoria honetarako hobe dela write-back idazketa-politika write-through baino.



Mariette

```
int berdinch(FILE *fd1, FILE *fd2)
{ Bootez Berdinchle FILE fd1, fd2 &
  while (!feof(fd1) && !feof(fd2) & & Berdinch)
  {
    if (tscant(fd1, 'S', lesson) != tscant(fd2, 'S', lesson))
      Berdinch = 0, 0;
    if (Bool = true)
      printf(XBi fitxetegich dira);
    else
      printf(Im ez dira berdinch);
```

P. Arifletat

a)

int Max = 20;

char X[21];

b)

long X[100][203];

c) int X[200];

d)

float (*f)(float *b)

10. Arifletat

Void faktoriell(int zan, long *cuc)

```

    {
        int n=0;
        int bat=1;
        while (n<=zan)
        {
            bat = bat * n;
            n++;
        }
    }

```

cuc = (long) bat;

Main (int argc, char *argv[])

{ &long faktorielle;

int zanb;

if (argc != 2)

printf ("Parametriköch fehle! Disk. 2);

exit (-1);

zanb = atoi(argv[1]);

faktoriell(zanb,faktorielle);

C Reference Card (ANSI)

Standard Utility Functions <stdlib.h>

```

Input/Output <stdio.h>
Standard I/O
standard input stream
standard output stream
standard error stream
end of file
get a character
print a character
print formatted data
print to string s
read formatted data
read from string s
read line to string s (< max chars)
print string s
File I/O
declare file pointer
pointer to named file
modes: r (read), w (write), a (append)
get a character
write to file
read from file
close file
non-zero if error
non-zero if EOF
read line to string s (< max chars)
write string s
Codes for Formatted I/O: "%-> o w.pnC"
+ left justify
+ print with sign
space pad with leading zeros
0 min field width
w precision
m conversion character:
c conversion character:
d,i integer
c single char
f double
o octal
p pointer
g,e same as f or e, E depending on exponent
g,e same as f or e, E depending on exponent

```

Variable Argument Lists <stdarg.h>

```

declaration of pointer to arguments
initialization of argument pointer
lastarg is last named parameter of the function
access next unnamed arg, update pointer
call before exiting function
va_list name;
va_start(name, lastarg)
va_arg(name, type)
va_end(name)

```

Integer Type Limits <limits.h>

The numbers given in parentheses are typical values for the constants on a 32-bit Unix system.

abs(n)	(8)
labs(n)	(127 or 255)
div(n,d)	(-128 or 0)
CHAR_MAX	max value of char
CHAR_MIN	min value of char
INT_MAX	max value of int
INT_MIN	min value of int
LONG_MAX	max value of long
LONG_MIN	min value of long
SCHAR_MAX	max value of signed char
SCHAR_MIN	min value of signed char
SHRT_MAX	max value of short
SHRT_MIN	min value of short
UCHAR_MAX	max value of unsigned char
UINT_MAX	max value of unsigned int
ULONG_MAX	max value of unsigned long
USHT_MAX	max value of unsigned short

Floating Type Limits <float.h> (2)

FLT_RADIX	radix of exponent rep
FLT_ROUNDS	floating point rounding mode
FLT_DIG	decimal digits of precision
FLT_EPSILON	smallest x so $1.0 + x \neq 1.0$
FLT_MANT_DIG	number of digits in mantissa
FLT_MAX	maximum floating point number
FLT_MAX_EXP	maximum exponent
FLT_MIN	minimum floating point number
FLT_MIN_EXP	minimum exponent
DBL_DIG	decimal digits of precision
DBL_EPSILON	smallest x so $1.0 + x \neq 1.0$
DBL_MANT_DIG	number of digits in mantissa
DBL_MAX	max double floating point number
DBL_MAX_EXP	maximum exponent
DBL_MIN	min double floating point number
DBL_MIN_EXP	minimum exponent

Mathematical Functions <math.h>

Arguments and returned values are double	
trig functions	$\sin(x)$, $\cos(x)$, $\tan(x)$
inverse trig functions	$\sin^{-1}(x)$, $\cos^{-1}(x)$, $\tan^{-1}(x)$
hyperbolic trig functions	$\sinh(x)$, $\cosh(x)$, $\tanh(x)$
exponentials & logs	e^x , $\log(x)$, $\lg10(x)$
exponentials & logs (2 power)	$\exp(x)$, $\exp_2(x)$, $\lg2(x)$
division & remainder	$\lfloor x \rfloor$, $\lceil x \rceil$, $\mod(x, y)$, $\frac{x}{y}$
powers	x^y , \sqrt{x} , $\sqrt[3]{x}$
rounding	$\text{ceil}(x)$, $\text{floor}(x)$, $\text{fabs}(x)$

May 1999 v1.3. Copyright © 1999 Joseph H. Silverman
Permission is granted to make and distribute copies of this card provided the copyright notice and this permission notice are preserved on all copies.

Send comments and corrections to J.H. Silverman, Math. Dept., Brown Univ., Providence, RI 02912 USA. (jhs@math.brown.edu)

Program Structure/Functions

```

type func(type1,...)
    function declarations
type name
main() {
    external variable declarations
    declarations
    statements
}
local variable declarations

type func(argc,...) {
    function definition
    local variable declarations
}

/* */
statements
return value;

main(int argc, char *argv[])
    main with args
exit(argv)
    terminate execution

C Preprocessor
include library file
replacement text
replacement macro
Example. #define max(A,B) ((A)>(B) ? (A) : (B))
undefined
quoted string in replace
concatenate args and rescan
conditional execution
name defined, not defined?
name defined?
line continuation char
\

Data Types/Declarations
character (1 byte)
integer
float (single precision)
float (double precision)
short (16 bit integer)
long (32 bit integer)
positive and negative
only positive
pointer to int, float, ...
enumeration constant
constant (unchanging) value
declare external variable
register variable
local to source file
no value
structure
create name by data type
size of an object (type is size_t)
size of a data type (type is size_t)

Initialization
initialize variable
initialize array
initialize char string

```

Constants

```

long (suffix)
float (suffix)
exponential form
octal (prefix zero)
hexadecimal (prefix zero-ex)
character constant (char, octal, hex)
newline, cr, tab, backspace
special characters
string constant (ends with '\0')
"abc...de"
\n, \r, \t, \b
\\, \?, \\", \

```

```

function definition
local variable declarations

```

```

declarations
statements

```

```

return value;

```

Pointers, Arrays & Structures

```

declare pointer to type
declare function returning pointer to type type *f()
declare pointer to function returning type type (*pf)()
generic pointer type
null pointer
object pointed to by pointer
address of object name
array
multi-dim array
Structures
    struct tag {
        declarations
    };
    declaration of members
create structure
member of structure from template
member of pointed to structure
Example. (*p).x and p->x are the same
single value, multiple type structure
bit field with b bits
    member : b
union

```

```

Data Types/Declarations
char
int
float
double
short
long
signed
unsigned
*int, *float, ...
enum
const
const
extern
register
static
void
struct
typedef typename
sizeof object
sizeof(type name)
logical or

```

```

Operators (grouped by precedence)
structure member operator
    name.member
structure pointer
    pointer->member
increment, decrement
plus, minus, logical not, bitwise not
indirection via pointer, address of object
cast expression to type
size of an object
multiply, divide, modulus (remainder)
add, subtract
left, right shift [bit ops]
comparisons
comparisons
bitwise and
bitwise exclusive or
bitwise or (incl)
logical and
logical or

```

```

ANSI Standard Libraries
<assert.h>
<cctype.h>
<errno.h>
<float.h>
<locale.h>
<math.h>
<setjmp.h>
<signal.h>
<stddef.h>
<stdio.h>
<stdlib.h>
<string.h>
<time.h>

```

```

Character Class Tests
<cctype.h>
isalnum(c)
isalpha(c)
isctrl(c)
isdigit(c)
isgraph(c)
islower(c)
isprint(c)
ispunct(c)
isspace(c)
isupper(c)
isxdigit(c)
tolower(c)
toupper(c)

```

```

String Operations <string.h>
s,t are strings, cs,ct are constant strings
copy ct to s
length of s
strlenc(s)
strcpy(s,ct)
strncpy(s,ct,n)
strcat(s,ct)
strncat(s,ct,n)
strcmp(cs,ct)
strncmp(cs,ct,n)
strchr(cs,c)
strrchr(cs,c)
memcp(s,ct,n)
memmove(s,ct,n)
memcmp(cs,ct,n)
memchr(cs,c,n)
memset(s,c,n)

```

Flow of Control

```

statement terminator
block delimiters
exit from switch, while, do, for
next iteration of while, do, for
go to
label
label:
    ; 
    { }
    break
    continue
    goto label
return expr

```

```

Flow Constructors
if statement
    if (expr) statement
else if (expr) statement
else statement
while (expr)
    for statement
        for (expr1; expr2; expr3)
            statement
do statement
    do statement
        do statement
        while(expr);
            switch (expr) {
                case const1: statement1 break;
                case const2: statement2 break;
                default: statement
            }

```

```

switch (expr);
    case const1: statement1 break;
    case const2: statement2 break;
    default: statement

```

```

    return expr

```

```

    unary operators, conditional expression and assignment operators group right to left; all others group left to right.

```

1.- Idatz ezazu programa bat 1 eta 100 zenbakien arteko zenbaki bakotien karratua pantailaratzeko. Aldatu programa nagusia funtziola hau erabilitzeko: int karratua (int n).

2.- Kode atal hau exekutatu ondoren, zein izango da ik adagaiaaren amaiarako balioa?

```
int x=12, j=5, k=0;
if (0==x)
    for (j=0; j<10; j=j+4) k=k+j;
else
    for (j=0; j<10; j=j+2) k=k+j;
```

3.- Programa zati bat kontuan hartuz,

```
int a=0, b=5;
int *c = NULL, *d=NULL;
c=&a;
d=&c;
*c = *c + (*d);
```

erantzun itzazu galdera hauek:

- c eta d aldagaiak memoriaiko posizio berdinnean gordetzen al dira?
- c eta d aldagaiak memoriaiko posizio berdinari apuntatzen al diote?
- *c=4 esleipenak, a aldagaiaren balioa aldatzen al diri? eta b-renai?
- zein da a aldagaiaren balioa exekuzioa bukatzean? eta b-renai?

4.- Ondorengo erazagupenak eta jarraian egiten diren esleipenak emanda, eta b) bektorearen osagaien helbideak zeintzuk diren kontuan izanda,

```
int b[3] = { 8, 3, 7 };
        /* &b[0] = 4   &b[1] = 6   &b[2] = 8 */
int *p;
p=&b[2];
```

dierazi ondorengo espresso baktoitzaren balioa zein izango den.

- (a) b (b) *b (c) p (d) *p

5.- Adierazi ondorengo aginduen exekuzioak ze aldaketa eragiten dituen t, x, y eta *y aldagaietan. Horretarako kontuan izan t bektorearen hasierako helbidea 0x4BC dela. Aginduak azaltzen diren ordenan exekutatzen dira.

```
t=7 int t[] = {437, 5, 46};
int x;
int *y;
```

```
(1) y=t[0];      (4) x=t[0];
(2) x=t[0]; x=t[1]; (5) t[1]=x;
(3) *y=69;       (6) t[2]=*y;
```

6.- Idatz ezazu string bat bestetean kopiatzen duten funtziorik erabili gabe. Bi string-ak funtziori pasatzen zaizkion parametroak dira.

7.- Idatz ezazu parametro bat funtzioa, programak teklatutik jasoko du bektorearen osagaiak (gelienez 5), eta M[2x3] (float) matrizearen osagaiak. Programak M matrizearen osagai bakoitzari kopurua (n).

8.- Idatz ezazu bektore batzen osagaien batez bestetean kalkulatzuen duten funtzioa: float bb_kalkulatu (int *b, int n). Funtziok bi parametro du: bektorea (b) eta osagai

1.- bektorearen batez bestetean batuko dio. Bukacran, pantailatik aterako du matrizearen azkeneko baliotak.

2.- Kodetu ondorengo erazagupenak:

- a) x = 20 osagaiatko string bat da
b) x = 100 letero eta 20 zutabe dituen matriz bat da. Matrizearen osagaiak 1 long motakoak

dira (long) ~~x[00000000]~~

c) x = 200 osagaiatko bektore bat da. Bektorearen osagaiak int motako datuen erakusleak

dira (long) ~~x[00000000]~~

e) x ondorengo eremuak dituen datu egitura bat da: 20 osagaiatko string bat, koma

higitxorioko zenbakia bat, 30 zenbakia osotako bektore bat, eta karakteren erakusle bat

dira (long) ~~x[00000000]~~

f2 ezer itzultzen ez duen funtziola bat da zenbakia errealean bektore bat jasotzen du

10.- Idatz faktto funtziola. Funtziola honetako parametro bezala jasotzen duen zen zenbakien faktoriala kalkulatuko du eta emaitza ema parametroan gordeko du. Funtziaren burua ondorengoa da:

```
void faktto (int zen, Long *ema) {
    Idatzi programa nagusi bat funtziola hori probatzeko.
    Programa nagusiaik parametro bezala zenbakia bat jaso dezake bere faktoriala kalkulatzeko. Ez badu parameterriok jasotzen, teklatutik zenbakia oso bat sartzeko eskanuko du.
```

Zenbakien faktoriala kalkulatu ondoren bere balioa aterako du pantailatik. Jasotako zenbakia negatiboa bada, errore mezu bat aterako du.

11.- Bi fixategi berdinak oie diren azterten duen funtziola idatzi. Funtziola hori probatzeko programa nagusi bat idatzi. Fixategien izenak programa nagusien argumentuak bezala pasatzen dira. Programa nagusia funtziari deia egiteaz gain, fixategiak irekiko eta ixiko diru, erroreak kontrollatuko, eta. Funtziaren prototipoa hau da:

```
int berdinak (FILE *fd1, FILE *fd2)
```

Funtziola 1 itzuliko du fixategiak berdinak badira eta 0 bestela.
12.- (a) Definitu bazkide datu egitura, bazkide batu buruzko ondorengo informazioa gordeteko duena: izena (40 karaktere), adina eta bazkide zenbakia. Ondoren, definitu bazkide_zerrenda aldagaiak: bazkide motako egituren 2000 osagai gordeko dituen bektorea.

(b) Idatzi programa bat hau egindo duena: (a) sarrerako fixategi batetik 2000 bazkideren datuak jaso eta bazkide_zerrenda bektorean gorde; (b) jarraitzen, bazkide jubilatuak (65 urteik gorakoak) jubilatuak. dat: fixategian imprimatu; eta (c) bukaeraan, programaren exekuzio-denborra pantailatatu. Sarrerako fixategia (bazkideak.dat) parametro bezala jasoko du.

13.- Erzagupen hauek kontuan edukita,

```
struct egitura {
    double x;
    double y;
};
```

```
struct egitura *p1;
struct egitura osagaiak[100];
```

Sententzia hauen artean zein izango da zuzen idatzita dagoena? Erraztuna arrazoitu.

- (a) osagaiak[5]=p1.x;

- (b) osagaiak.x[5]=p1->x;

- (c) osagaiak[5].y=p1->x;

- (d) osagaiak[5].x=p1.x;

String-alg est une processus bivalve, avec tellement que "lo" marqué bivalve pour deux fois.

int b[3] = {8, 3, 2};

int *p;

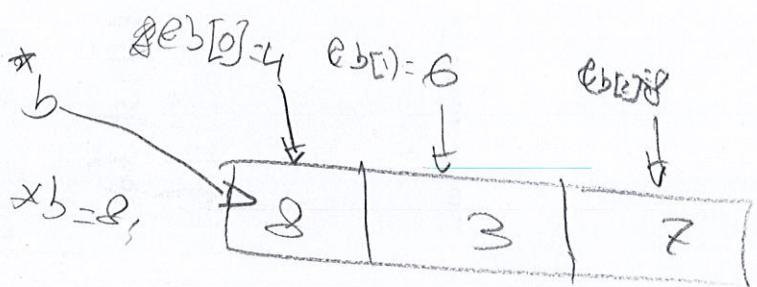
p = &b[2];

a) $b = \cancel{4}$ ~~Bren belbiac~~

b) $\star b = 8;$

c) $p = \cancel{8} \text{ Bren belbiac } + 8$

d) $\star p = \cancel{8} 2;$



$b = \&b[0] = 4;$

Konputagailuen Arkitektura

Puzzlearen enuntziatuak

> Atazten banaketa

Aplikazio bat parallelizatzen denean, ebatzi behar den arazo bat atazten banaketa da (*scheduling*). Banaketak orekutta izan behar du, prozesu guztien artean banatzten diren atazak denborara berean buka dattezen.

Jarduera honen bidez, banaketaren arazoa eta komponibideak ikasi eta ulertu beharko dituzu. Horretarako, erreferentzia batzuk ematen dizkizugu. Irakurri informazioa eta prestatu aurkezen bat zure taldekidetzentzat, arazoa eta irtenbideak azaltzeko.

Horretaz gain, eskolan banatutako ariketa multzoko 7. eta 8. ariketak egin beharko dituzu. Bukaizeko, ataza.c programaren bertsio paraleloa egin beharko duzu. OpenMP-k eskaintzen dituen planifikazio-aukerak probatu eta lortzen den eraginkortasuna analizatu beharko duzu hari kopuru ezberdinetarako, planifikazio-aukera egokiena proposatzeko. Dokumentu labur batean (1 edo 2 orri) azaldu beharko duzu proposatu duzun programa paraleloa, egin dituzun probak eta lortu dituzun emaitzak (exekuzio-demborak) eta lortzen diren azelerazio-faktoreak eraginikortasunak hari kopuru ezberdinetarako) taulak edo grafikoak erabiliz.

ataza.c fitxategia u010415.gi.ehu.es makinen templates/puzzlea direktorioan dago.

Bibliografia orokorra

- <http://www.sc.ehu.es/acwarfra/arpar/AP/apnagusia.htm>. 2. kapituluko 4. atala.
- Chandra R., et al.: *Parallel Programming in OpenMP*. Morgan Kaufmann, 2001.
- Pacheco P., An Introduction to Parallel Programming. Morgan Kaufmann, 2011. 2.6 atala eta 5. kapitulua
- <http://www.sc.ehu.es/acwarfra/arpar/AP/apnagusia.htm>. 4. kapituluko 1, 2, 3 eta 4. atala.
- Ortega J., Anguita M., Prieto A.: *Arquitectura de Computadores*. Thomson, 2005. 10. kapituluko 3. atala.
- Chandra R., et al.: *Parallel Programming in OpenMP*. Morgan Kaufmann, 2001. 5 kapitulua.

Atazten banaketai buruzko bibliografia

- <http://www.sc.ehu.es/acwarfra/arpar/AP/apnagusia.htm>. 8. kapituluko 5. atala.
- Almeida F., Giménez D., Mantas J.M., Vidal A. M.: *Introducción a la programación paralela*. Paranimfo, 2008. 5. kapituluko 3. atala.

> Sinkronizazioa

Aplikazio bat parallelizatzen denean, prozesuen arteko sinkronizazioa kontuan hartu behar da, aldagai partekatuak erabiltzen baitira.

Jarduera honen bidez, sinkronizazioaren arazoa eta komponibideak ikasi eta ulertu beharko dituzu. Horretarako, erreferentzia batzuk ematen dizkizugu. Irakurri informazioa eta prestatu aurkezen bat zure taldekidetzentzat, arazoa eta irtenbideak azaltzeko

Horretaz gain, eskolan banatutako ariketa multzoko 11. ariketa egin beharko. Horretaz gain, bukatzen diren aldagai partekatuak egoki erazagutuz. Dokumentu labur batean (1 edo 2 orri) azaldu beharko duzu proposatu duzun programa paraleloa, egin dituzun azelerazio-faktoreak eta lortu dituzun emaitzak (exekuzio-demborak eta lortzen diren azelerazio-faktoreak eta eraginkortasunak hari kopuru ezberdinetarako) taulak edo grafikoak erabiliz.

bektore.c fitxategia u010415.gi.ehu.es makinen templates/puzzlea direktorioan dago.

Bibliografia orokorra

- <http://www.sc.ehu.es/acwarfra/arpar/AP/apnagusia.htm>. 2. kapituluko 4. atala.
- Chandra R., et al.: *Parallel Programming in OpenMP*. Morgan Kaufmann, 2001.
- Pacheco P., An Introduction to Parallel Programming. Morgan Kaufmann, 2011. 2.6 atala eta 5. kapitulua

Sinkronizazioari buruzko bibliografia

- <http://www.sc.ehu.es/acwarfra/arpar/AP/apnagusia.htm>. 4. kapituluko 1, 2, 3 eta 4. atala.
- Pacheco P., An Introduction to Parallel Programming. Morgan Kaufmann, 2011. 2.6 atala eta 5. kapitulua
- <http://www.sc.ehu.es/acwarfra/arpar/AP/apnagusia.htm>. 8. kapituluko 5. atala.
- Chandra R., et al.: *Parallel Programming in OpenMP*. Morgan Kaufmann, 2001. 5 kapitulua.

> Eraginkortasuna

Aplikazio bat paralelizatzen denean, oso garantziusua da abalik eta eraginkortasun handiena lortzea. Aplikazio paralelo baten eraginkortasuna, besspeak beste, paralelizatu daitetean kode-frakzioaren menpe dago, Amdahl-en legeak adierazten duen moduan.

Jarduera honen bidez, eraginkortasunari buruzko kontzeptuak ikasi eta ulertu beharko dituzu; hauen artean, ezaguna den Amdahl-en legea. Horretarako, erreferentzia batzuk ematen dizkizugu. Irakurri informazioa eta prestatu aurkezen bat zure taldekieentzat, eraginkortasunaren gaiaz azaltzeko

Horretaz gain, eskolan bantutako ariketa multzoko 4., 5. eta 6. arketak egin beharko dituzu. Bukazeko, erag.c programaren bertso paraleloa egin beharko duzu. Horretarako aztertu beharko dituzu begizta guztiak, paralleliza daitetean ala ez erabakitzeko (erabilitzen diren aldagaien erabilera kontuan hartu). Dokumentu labur batetan (1 edo 2 orri) azaldu beharko duzu proposatu duzun programa paraleloa, egin dituzun probak eta lortu dituzun emaitzak (exekuzio-denborak eta lortzen diren azelerauzio-faktoreak eta eraginkortasunak hari kopuru ezberdinetarako) taulak edo grafitoak erabiliz.

erag.c fitxategia u010415.gi.ehu.es makinaren templates/puzzle direktorioan dago.

Bibliografia orokorra

- <http://www.sc.ehu.es/acwarfia/apar/AP/cpnagusia.htm>. 2. kapituluko 4. atala.
- Chandra R., et al.: *Parallel Programming in OpenMP*. Morgan Kaufmann, 2001.
- Pacheco P., An Introduction to Parallel Programming. Morgan Kaufmann, 2011. 2.6 atala eta 5. kapitula

Eraginkortasunari buruzko bibliografía

- <http://www.sc.ehu.es/acwarfia/apar/AP/apnagusia.htm>. 2. kapituluko 5. atala.
- Ortega J., Anguita M., Prieto A.: *Arquitectura de Computadores*. Thomson, 2005. 7. kapituluko 5.2. atala.

Konputagailuen Arkitektura

Karpetaren egiturari eta edukiari buruzko **zenbait iradokizun orokor**

Karpetaren edukia ondo antolatuta mantendu behar da, ezin da dokumentu multzo soila izan. Zaindu edukiaren egitura eta formatua, dokumentu guztiak itxura homogeneoa izan dezaten. Horrez gain, jasotzen duzuen informazioa bere testuinguruan kokatu (adibidez, taula bat sartzen baduzue, taulak adierazten duen informazioa azaldu behar duzue, bere analisia eginaz).

Beste aldetik, karpetaren edukia dinamikoa da, hau da, hobetu daiteke. Adibidez, eskolan landu eta gero, puzzleko ariketak zuzendu ditzakezue une honetan. Horrela, proiektuaren bukaeran, karpetaren edukia zuzena eta osoa izatea izango da helburu. Hori dela eta, karpetaren hurrengo bertsiorako komenigarria izango litzateke adieraztea orri batean zer hobekuntza egin dituzuen aurretik entregatu duzuen bertsioarekiko.

Bukatzeko, karpetaren edukirako proposamen bat egin nahi dizuegu. Hala ere, zuek erantsi dezakezue egokia iruditzen zaizuen guztia. Karpetak bi formatu izango ditu. Alde batetik, paperezko formatua, eta bestetik formatu digitala. Azken hau, programen iturburu kodeek, gardenkiek, aplikazioaren memoriak, etab. osatuko dute. Hori bai, beste dokumentu batzuk, adibidez, egin dituzuen bileren aktak ez dira jarri behar bertsio digitalean.

Karpetaren azken bertsioa **urtarrilaren 21ean** entregatu beharko duzue, proiektuaren aurkezpenarekin batera. Karpetaren azken bertsioa entregar beharko diozue irakasleari, eta bestetik eGela-ren bidez bertsio digitala igo beharko duzue.

Proposatzen dugun egitura minimoa

>> Proiektuaren izenburua eta taldearen identifikazioa (kideak eta kontu-zenbakia)

>> Karpetaren edukiaren aurkibidea

>> Proiektuaren enuntziatura eta konsultatu den informazioa

Atal honetan proiektuaren enuntziatura, konsultatu den informazioa eta bibliografia jarriko da
(konsultatu den materialaren erreferentziak soilik izan daitezke, edo informazio guztia)

>> Proiektauan egin diren jarduerak:

- Posterra

- Puzzlea

Planteatu diren arazoak aurkezpena eta bibliografian aurkitu diren soluzio bideen azalpenak

Ariketak ebatzia

Programen bertsio paraleloa: iturburu kodea komentatua eta lortu diren emaitzen analisia.

Adibidez:

- Eraginkortasunaren kasuan, lortu diren denboren laburpena eta azalpena, eta Amdahl-en legea kontuan hartuta aurreikusi zitezkeen denborak
- Atazen banaketaren kasuan, lortu diren exekuzio-denboren analisia erabili diren planifikazio-aukeren arabera
- Sinkronizazioaren kasuan, ebazpenaren zuzentasunaren analisia

- Aplikazioa

Aplikazioaren memoria (pasa dizuegun iradokizun-dokumentua kontuan hartuta)

>> 1. Eranskina

Aktak: taldea osatzeko agiria eta taldearen lan-saioko agiriak

LAN-TALDEAREN LAN-SAIKO AGIRIA

Donostian, 20....koaren(e)an

.....-etan, Konputagailuen Arkitektura irakasgaiko proiektua elkarrekin lantzeko osatutako taldeko kideak elkartu egin dira bileran.

BILERARA BERTARATUTAKO TALDEKIDEAK:

- 1)
- 2)
- 3)

JORRATUTAKO GAIAK ETA HARTUTAKO ERABAKIAK:

- 1)
- 2)
- 3)
- 4)
- 5)

HURRENGO BILERA BATERAKO GERATU DIREN GAIAK ETA EGIN BEHARREKO ATAZEN BANAKETA:

- 1)
- 2)
- 3)
- 4)
- 5)

Bilera-etan amaitu da, eta bertan hitz egindakoa eta adostutakoa agerian uzteko xedezi, honako hau izenpetu dugu taldekide guztiok.

Izena:.....

Izena:.....

Izena:.....

Konputagailuen Arkitektura (2013/2014)

3. gaian (Konputazio Paraleloko Sistemen Oinarriak) eskolatik kanpo
egindako lanen dedikazioa

Jarduera	Dedikazioa (ordu/min.)
Aurreko lanak C programazio-lengoaia ikastea Proiektuaren edukia finkatu eta oinarrizko kontzeptuak errepasatu OpenMP ikastea	Aurreko lanak guztira
Puzzlea Irakurketa, informazioaren bilaketa eta ulermena Ariketen ebaezpena Programa paraleloen garapena Informazioa lantzeko talde-lana Aurkezpenaren prestaketa	Puzzlea guztira
Aplikazioa Irakurketa, informazioaren bilaketa eta ulermena Garapena: diseinua eta programazioa, aukera desberdinaren analisia, exekuzio-denboren analisia, etab. Memoriaren idazketa Aurkezpenaren prestaketa	Aplikazioa guztira
Azterketa	Azterketa prestatzeko dedikazioa
	Guztira (dedikazio guztien batura)

Esker anitz galdetegi hau betetzeagatik



PROIEKTUETAN OINARRITUTAKO IKASKUNTZA KONPUTAZIO PARALELOKO SISTEMEN OINARRIAK

1.- Galdera eragilea: Ba al dago bizitza Marten?



2.- Aurreko jokalekuak

NASA aztertzen ari da Marten bizitza ote dagoen ala ez. Horretarako, espazio-zunda bat bidali nahi du planetaren argazkiak zehaztasun handiagoarekin ateratzeko eta modu horretan bizitza izatearen aukera hobeto analizatzeko. Informazio gehiago web helbide hauetan aurki dezakezu:

http://www.nasa.gov/mission_pages/msl/index.html
http://en.wikipedia.org/wiki/Mars_Science_Laboratory

Azterketa hauetan argazki asko analizatzen dira. Hori dela eta, oso importantea da irudi baten prozesu-denbora ahalik eta txikiena izatea, minutuko azter daitekeen argazki kopurua handitu ahal izateko. NASAk dionez, 10 MB-ko argazki bat prozesatzeko, denbora maximoa 350 milisegundo izan beharko luke.

Aipatutako beharrak ikusita, zein iruditzen zaizu irtenbide egokiena arazoari aurre egiteko?

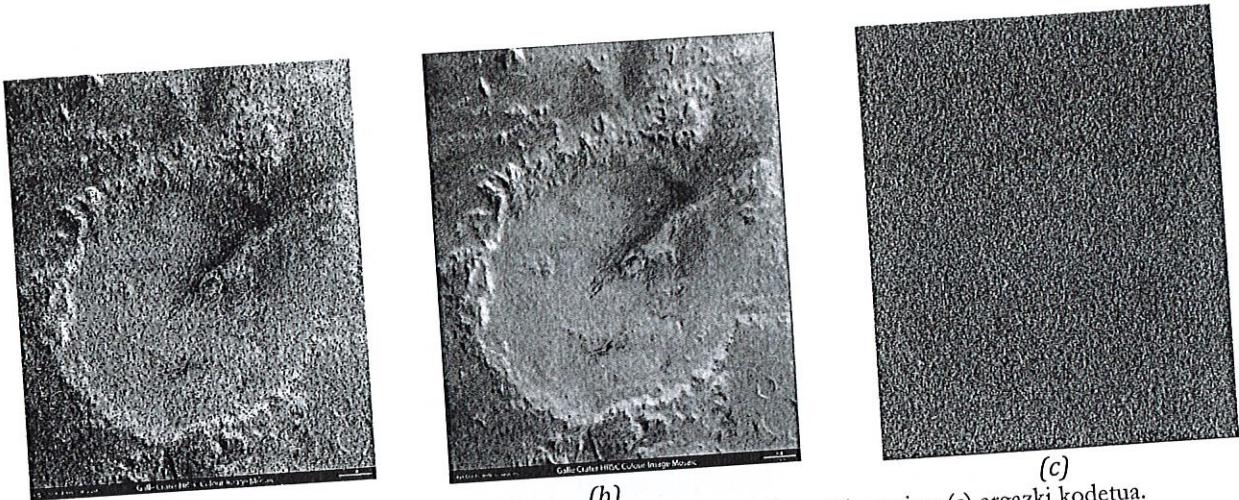
3.- Jokalekuak

Aurreko jokalekuak planteatu den arazoari aurre egiteko, NASAk multiprozesadore bat jarriko du bidaliko duen zundan argazkiak prozesatzeko eta NASAren instalazioetara bidaltzeko.

Argazkiak prozesatzeko eta bidaltzeko algoritmo paralelo bat diseinatzea eskatu digu NASAk. Alde batetik, irudiari iragazki bat aplikatuko zaio zundak lortu duen argazkiaren zarata minimizatzeko, eta bestetik argazkia kodetuko da argazkiaren edukia modu sekretuan bidaltzeko. Bukatzeko, NASAko instalazioetara bidaltzeko argazkiaren transmisioa prestatuko da.

Prozesu-denbora mugetan mantentzeko, zundan jarri den multiprozesadorea aprobetxatu egin nahi da, informazioa paraleloz prozesatzeko ahalmena baitu. Multiprozesadorean algoritmo paralelo bat programatuko dugu. Argazkia ahalik eta azkarren prozesatzea izango da programa paraleloaren helburua. Hala, NASAk ahal den argazki kopuru handiena minutuko analizatzea espero du.

Adibide gisa, 1. Irudian egin nahi den prozesua azaltzen da. Lehenengo argazkia (1a. Irudia) zundak hartutakoa izango litzateke, 1b. Irudian agertzen dena iragazkia aplikatu eta geroko, eta 1c. Irudian agertzen dena kodetu eta gero lortuko dugun irudia. 1. Taulan argazkia prozesatzeko lortzen den eraginkortasunaren adibide bat ageri da, erabili den prozesadore kopuruaren arabera. Jatorrizko irudiaren tamaina 10 MB da.



1. Irudia.- 10 MB-ko irudi baten prozesua: (a) jatorrizko irudia, (b) irudi iragazia y (c) argazki kodetua.
Irudiaren iturria: ESA/DLR/FU Berlin (G. Neukum)

	Exekuzio-denbora (ms)	speed-up
1 prozesu	5979	0,9
2 prozesu	3064	1,7
4 prozesu	1605	3,2
8 prozesu	875	5,8
16 prozesu	494	10,3
32 prozesu	312	16,4

1.Taula.-10 MB-ko irudi prozesatzean lortzen den eraginkortasuna prozesadore kopuruaren arabera.

4.- Ikaste-prozesuaren emaitzak

Ataza hau amaitzean, honako hauek egiteko gai izan beharko lukete ikasleek:

1. Azaltza behe- eta goi-mailako paralelismoko kontzeptuak.
2. Definitza estrategia egokiena aplikazio bat paralelizatzeko, aplikazioan agertzen diren dependentziak analizatuta.
3. Erabiltza sinkronizaziorako mekanismo egokiak programa paraleloen funtzionamendua zuzena izan dadin.
4. Planifikatza prozesuen arteko lan-kargaren banaketa orekatua.
5. Analizatza aplikazio baten implementazio paraleloaren eraginkortasuna.
6. OpenMP erabiltza aplikazioak paralelizatzeko.

Irakasgaiaren berezko gaitasunez gain, <http://www.ehu.es/documents/340468/516505/Gaitasunak.pdf> dokumentuko T4, T8 eta T9 gaitasun orokorrak, eta informatika adarreko IA1 eta IA9 gaitasunak landuko dira.

5.- Entregatzekoak

E1 entregatzekoak

Taldea osatzeko agiria (E1.1), taldean aritzeko hartutako konpromisoen dokumentua eta proiektau aurrera eramateko egin diren bileren aktak.

E2 entregatzekoak

Jokaleku aztertzeko egingo den posterra (E2). Entregatzeko multzo hau puzzlearekin dago lotuta. Ikasle bakoitzak egingo duen lanaren bilketa izango da: txostena (E3.1), aurkezpena (E3.2) eta ariketa ebatziak (E3.3). Horretaz gain, taldekkide bakoitzak bere atala ikasteko erabili duen material berria (hau da, irakasleak eman ez duena) sartuko da entregatzeko honetan.

E3 entregatzekoak

Berдинen arteko aurkezpenen evaluazioen txostenek osatzen dute entregatzeko hau. Alde batetik, puzzlearena (E4.1) eta, bestetik, aplikazioaren aurkezpenarena (E4.2).

E4 entregatzekoak

- E5 entregatzekoak** Gutxieneko ezagutzak eskuratu direla egiaztatzeko egingo diren azterketek osatzen dute entregatzeko hau. Alde batetik, C lengoaiari buruzko laborategiko txantiloia (E5.1), ariketak (E5.3) eta azterketa (E5.2); eta bestetik paralelismoari buruzko azterketa (E5.4).
- E6 entregatzekoak** Jokalekua ebazteko garatu den softwarearen txosten teknikoa (E6.1), planteatu den ariketa nola ebatzi den azalduz, eta ebazenaren aurkezpena prestatzeko erabili den materiala (E6.2).
- E7 entregatzekoak** Portafolio edo karpeta (E7.1), non proiektua egiteko erabilitako material guztia bilduko den, entregatzeko guztiak kontuan hartuta. Hau da, taldekideek egokitzat jotzen duten material guztia, egin duten proiektua ondo ulertzeko. Azkeneko bertsioa proiektuaren bukaeran entregatuko bada ere, talde bakoitzak lehenengo bertsio bat entregatuko du puzzlea egin eta gero. Karpetaren edukia errebisatu eta gero berriro bueltatuko da. Horrela, edukia hobetzeko aukera izango du taldeak.

Ikaslearen ebaluazioarekin lotutako entregatzekoak (E5.1, E5.2, E5.3 eta E5.4) indibidualak izango dira. Beste guztiak, berriz, taldeko entregatzekoak, kooperatiboak izango dira.

6.- Ebaluazio-sistema

Proiektu honen hazta irakasgaiaren kalifikazio osoan %40 da (4 puntu, alegia). Ebaluazioan honako hauek hartuko ditugu kontuan:

- Aurkezpen indibiduala. Proiektuan bi aurkezen izango dira. Bata, puzzleari buruz; bigarrena, proiektuan landu den ebazenari buruz. Talde bakoitzak horietariko bat egingo du, baina taldekide batek egingo du, momentuan ausaz aukeratua. Jarduera hau kooperatiboa denez, ikasleak aurkezpenean lortutako nota talde osoaren nota izango da. Aurkezpenen ebaluazioetan berdinente arteko ebaluazioak kontuan hartuko dira, eta notaren %10a balioko du.
- Gutxieneko ezagutzak egiaztatzeko azterketak. Bi azterketa izango dira: bata, C lengoaiako ezagutzak ebaluatzeko, eta bestea paralelismoko ezagutzak ebaluatzeko. Bigarren kasuan, ikasleak nota minimo bat atera beharko du, azterketaren %30a. Bi azterketa hauek notaren %15a balioko dute.
- Garatu den softwarea eta garapena azaltzen duen txosten-teknikoa. Jarduera honek notaren %13a balioko du.
- Proiektuaren behin betiko karpeta. Honen balorazioa notaren %2 izango da.

Horretaz gain, kontrolgune (KG) motako jarduerak gainditu beharko ditu ikasleak: C-ko laborategiko txantiloia, C-ko ariketak eta puzzleko ariketak. Kontrolgune motako jarduera GAI / EZ GAI bezala ebaluatuko da, proiektuaren notan eragina izan gabe. Kontrolgune motako jardueraren bat edo paralelismoko azterketan nota minimoa lortzen ez bada, ikasleak ezin izango du jarraitu irakaskuntza-metodologia honetan.

Ebaluazio-prozeduran “extra bonus” sistema bat aurreikusten da. Hau da, ebaluazio-sisteman proposatzen diren puntuazioez gain, jardueraren bat bereziki ondo egin bada, gehigarritzko puntuak emango dira.

	Ebaluatzalea		Nota
	Irakaslea	Ikaslea	
Banaka	C: txantiloia (E5.1) C: azterketa (E5.2) C: jarraipen ariketak (E5.3) Paralelismo: azterketa (E5.4)	KG %5 KG %10	%15
Taldeka	Puzzle/ebazpena: aurkezpena (E3.2/E6.2) %5 Puzzle: ariketak (E3.3) Txosten teknikoa eta garapena (E6.1) Proiektuaren karpeta (E7.1)	KG %13 %2	Puzzle/ebazpena: aurkezpena (E4.1/E4.2) %5 %25
Nota		%35	%5 %40

2.Taula.-Ebaluazio-sistema.

IKASLEAREN LANAREN PLANIFIKAZIOA

Eskola data	Saioa	Eskolako ekintzak	Eskolatik kantoko ekintzak	Entregatzekoak	Dedikazioa	Ebaluatutako ehunekoa (%40)
Urriak 8 (E7 labo.)	0.1	Laborategiko praktikak: C lengoiaia	C-n programatzen ikasi	E5.1: laborategiko txantiloia [EK, 4 ordutu]	5 ordutu [EK, 4 ordutu]	E5.1: GAI / EZ GAI
Urriak 27 (gela)	0.2	C-ko ezagutzak egiaztatzeko azterketa	C-ri buruzko jarrapen-arketak	E5.2: azterketa	30 minutu [EK, 2 ordutu]	%5 (E5.2)
Azaroak 6 (gela)	1	Galdera eragilea + aurreko jokalekuak Projektuaren enuntziatura banatu Eztabaida taldeka, posterra Oinarrizko kontzeptuen aurkezpena	Projektuaren edukia finkatu eta oinarrizko kontzeptuak errepasatu	E1.1: taldea osatzeko agiria E2: posterra (din-A4) E5.3: C-ko ariketak	20 minutu 25 minutu 25 minutu 20 minutu [EK, 1 ordutu]	E5.3: GAI / EZ GAI
Azaroak 9 (E7 labo.)	2	Laborategiko praktikak: OpenMP	OpenMP-n programatzen ikasi		90 minutu [EK, 1 ordutu]	
Azaroak 10 (0.1 labo.)	3	Puzzlearen atazak definitu Puzzlean esleitutako ataza landu	Puzzlean esleitutako ataza landu		60 minutu 30 minutu [EK, 1 ordutu]	
Azaroak 13 (gela)	4	Puzzlea: adinuen bilera	Puzzleari buruzko txostena eta aurkezpena prestatu		90 minuta [EK, 2 ordutu]	
Azaroak 23	5		Puzzlearen txostena / aurkezpena		[EK, 2 ordutu]	
Azaroak 24 (gela)	6	Puzzlea: taldearen bilera Aplikazioaren (algoritmoak)	Puzzleari buruzko txostena eta aurkezpena prestatu		60 minutu 30 minutu [EK, 3 ordutu]	
Azaroak 27 (gela)	7	Puzzlea: aurkezpena Puzzlea: eztabaidea	Aplikazioa garatu, dokumentazioa landu, ezagutza lortu	E4.1: berdinan arteko ebaluazioaren txostena	60 minutu 30 minutu [EK, 1 ordutu]	[%10 (E3.2+E4.1)]
Azaroak 30 (gela)	8	Puzzlea: ariketen aurkezpena eta eztabaida	Aplikazioa garatu, dokumentazioa landu, ezagutza lortu	E3.1/E3.2/E3.3: puzzleari buruzko txostena E7.1: karpeta	90 minutu [EK, 2 ordutu]	E3.3: GAI / EZ GAI
Abenduak 1 (E7 labo.)	9	Taldeko lana: aplikazioa garatu, dokumentazioa eta ezagutza partekatu	Aplikazioa garatu, landu, ezagutza lortu		90 minutu [EK, 3 ordutu]	
Abenduak 4 (E7 labo.)	10	Taldeko lana	Aplikazioa garatu, landu, ezagutza lortu		90 minutu [EK, 1 ordutu]	
Abenduak 9 (gela)	11	Gutxienerko ezagutzak egiaztatzeko azterketa	Aplikazioa garatu, landu	E5.4: azterketa	90 minutu [EK, 4 ordutu]	%10 (E5.4) (minimoa: %30a)
Abenduak 11 (E7 labo.)	12	Taldeko lana	Proiektxua garatu, memoria, aurkezpena		90 minutu [EK, 6 ordutu]	
Abenduak 18	13		Aurkezpena prestatu	E6.1: txosten teknikoa eta garatutako softwarea	[EK, 3 ordutu]	%13 (E6.1)
2016/01/21	14	Projektuaren aurkezpena		E6.2: aurkezpenaren materiala E4.2: berdinan arteko ebaluazioaren txostena E7.1: karpeta	2 ordutu	[%10 (E6.2+E4.2)] %2 (E7.1)

Konputagailuen Arkitektura

OpenMP: adibideak

```
*****  
keixo.c (OPENMP)  
Hariak sortu OMP_NUM_THREADS ingurune-aldegaia ezabiliz  
export OMP_NUM_THREADS=4  
*****
```

```
#include <omp.h>  
  
#include <stdio.h>  
  
#define N 24  
  
int A[N];
```

```
main ()  
{  
    int i, tid, nthr;  
    for (i=0; i<N; i++) A[i]=0;  
    #pragma omp parallel  
    {  
        nthr = omp_get_num_threads();  
        tid = omp_get_thread_num();  
        printf ("%d (%d) haria exekuzioan\n", tid, nthr);  
        A[tid] = tid + 100;  
        printf ("%d haria bukatuta\n", tid);  
    }  
  
    for (i=1, i<=N; i++)  
    {  
        printf ("A[%d]=%d ", i-1, A[i-1]);  
        if ((i%8) printf ("\n");  
    }  
}
```

wait 123

Makinaren helbidea u010415.gi.ehu.es da. Fakultateko edozein puntutik atzigarri dago, baina unibertsitateko saretek atatzeko VPN tresna erabili behar da.

Laborategiko saiak linux-en egingo ditugu eta u010415.gi.ehu.es makinara konektatzeko ssh komando hau erabili dugu terminal batetik:

```
> ssh erabiltailea@u010415.gi.ehu.es
```

Laborategian erabiliako ditugun adibideak templates/adibideak direktorioan aurki ditzakezue. Karpeta berri bat ireki dezakezue, adibideak kopiatu eta kapeta horretan lan egiteko. Horretarako komando hauek erabili ditzakezue:

```
> mkdir adibideak  
> cp templates/adibideak/*.c adibideak  
> cd adibideak
```

> Konplazioa eta programen exekuzioa

Programa bat komplatzeko gcc (GNU) kompliladorea erabiliako dugu.

prog.c izeneko programa komplatzeko, honako aukera hauek izango ditugu. Modu paraleloan, OpenMP-ko sasiagindua eta funtzioka erabilitzeko, behar den liburuategia adierazi behar da.

```
> gcc -o prog prog.c          (serian)  
> gcc -fopenmp -o prog prog.c (paraleloan)
```

Akatsik ez badago, dagokion exekutagarria sortuko da. Exekuzioan erabili nahi den hari kopuria kontrolatzeko, beste aukera batzuk badade ere, OMP_NUM_THREADS ingurune-aldegaia erabiliko dugu:

```
> export OMP_NUM_THREADS=xx      xx = hari kopuria
```

```

***** speedup.c (SERIE)
Bari kopuruaren eragina azterzeko adibidea
***** */

#include <stro.h>
#include <sys/time.h>

#define N 10000000
#define ITERAZIO 10

struct timeval t0, t1;
int i;
int C[N], J[N];
int itera_kop;

// ERRTUTINAK

void Exe_Denbora(char * pTestu, struct timeval *pt0, struct timeval *pt1)
{
    double tej;
    double tej = (pt1->tv_sec - pt0->tv_sec) + (pt1->tv_usec - pt0->tv_usec) / 1e6;
    printf("ts = %10.3f ms\n", pTestu, tej*1000);
}

void Bek_Pant(int *V, int Ll, int L2)
{
    int i;
    for (i=Ll; i<L2; i++)
    {
        if(i%5==0) printf("\n");
        printf("%d ", V[i]);
    }
    printf("\n");
}

// PROGRAMA NAGUSIA
main ()
{
    for (itera_kop=0; itera_kop<ITERAZIO; itera_kop++)
    {
        for(i=0; i<N; i++)
        {
            C[i] = i;
            J[i] = 6;
        }

        gettimeofday(&t0, 0);
        for(i=0; i<N; i++)
        {
            C[i] = ((C[i]+1) / J[i])+1;
            J[i] = (C[i] * J[i] + 1) / C[i] * J[i];
        }

        gettimeofday(&t1, 0);
        Exe_Denbora("Time: ", &t0, &t1);
        printf("\n\nC-> ");
        Bek_Pant (C, 0, 10);
        Bek_Pant (C, N-10, N);
        printf("\nD-> ");
        Bek_Pant (D, 0, 10);
        Bek_Pant (D, N-10, N);
    }
}

```

OpenMP 2.0 (Sintaxiaren laburpena, C/C++)

Sasiaginduaren sintaxia

> simtaxi orokorra

> baliditzaapeko konpilazioa

#endif

#ifdef _OPENMP

#pragma omp sasiagindua [klauzula]

(simkronizazio-hesi bat bukarean)

Atzenen banaketa atal Paralelo baten barruan (work sharing) (simkronizazio-hesi bat bukarean)

#pragma omp parallel for [klauzula]

> for begiztak

> sekiyo parallelok
(pragma omp section bana sekiyotan)

#pragma omp section
{ ... }

#pragma omp sections [klauzula]
{ ... }

> sekiyo (....)

#pragma omp parallel for [klauzula]
{ ... }

Atal parallelo + work sharing

> sekiyo (....)

#pragma omp parallel for [klauzula]
{ ... }

> sekiyo (....)

#pragma omp parallel sections [klauzula]
{ ... }

> sekiyozak baino ez dituenetan

#pragma omp parallel sections [klauzula]
{ ... }

> sekiyo kritikoa

#pragma omp atomic
{ ... }

> eragikete atomika

#pragma omp barrier
(hesi bat atal paralleloren eta ataza-banaketaiko sasiaginduen bukarean).

> hesia

#pragma omp flush [klauzula]
(flush implizitua honako sasiagindu hauean: barrier, parallel, critical eta ordered (sartean eta itzieran); eta for, section eta stringe (itzieran), nowait klauzula erabili ez badala).

> kontsistentezka sekuentziala

#pragma omp ordered
(ordenatuta sekuentziala eta ordenatu master (klauzula erabili ez badala)).

> ordena exekuzioa

#pragma omp master
{ ... }

> aldagai pribatu beretik

#pragma omp threadprivate (klauzula)
{ ... }

Aldagaien esparra

int	omp_get_thread_num (void)	get thread number
int	omp_get_num_procs (void)	get number of processes
int	omp_set_num_threads (int)	set number of threads
int	omp_get_max_threads (void)	get maximum threads
void	omp_set_dynamic (int)	set dynamic scheduling
int	omp_get_dynamic (void)	get dynamic scheduling
void	omp_set_nested (int)	set nested parallelism
int	omp_get_nested (void)	get nested parallelism
void	omp_set_nested_level (int)	set nested parallelism level
int	omp_in_parallel (void)	check if in parallel region
double	omp_get_time (void)	get time in seconds
double	omp_get_wtime (void)	get wall time in seconds

- **Gainerako funtzioak** (haridak kudeatzeko eta denborra neutrako)

```

void omp_init_lock(omp_lock_t*)
    memory erreserba tu eta sartaila hasiera tu
void omp_destroy_lock(omp_lock_t*)
    memoria askatu
void omp_inite_lock(omp_lock_t*)
    memoria erreserba tu eta sartaila hasiera tu
void omp_unset_lock(omp_lock_t*)
    ixarion sartaila ireki arte, eta, gero, itxi (LOCK)
void omp_set_lock(omp_lock_t*)
    ixarion sartaila ireki arte, eta, gero, itxi (LOCK)
void omp_useset_lock(omp_lock_t*)
    ireki sartaila (unlock)
void omp_unset_lock(omp_lock_t*)
    aztertu sartailaren balioa; iztuli TRUE edo FALSE
int omp_test_lock(omp_lock_t*)

```

Sarrailak erabilizteko funtziok

OMP_NUM_THREADS	8	8	OMP_SCHEDULE	"dynamic,2"	planifikaazio molta runt i me kasurako (adibide bat)	hari kopuruaren kontrol dimamikoa	akribatua / desakribatu habiratutako parallelismoa	TRUE / FALSE	TRUE / FALSE	omp_dynamic	omp_nested
-----------------	---	---	--------------	-------------	--	-----------------------------------	--	--------------	--------------	-------------	------------

• **ligrurune-aldagaiak** (export xx=...)

• Sastagindu bakotzeke klausulak

1. Multiprozesadore batetik 16 prozesadore ditu. Executatu egin behar den programa analizatu eta gero, honako datu hauek lortu dira: programaren zati bat % 50a, 16 prozesadoreetan executatu direkte paraleloan; beste zati bat, % 25a, 8 prozesadoreetan bakiarrak executatu direkte paraleloan; gainetikoa, prozesadore bakar batean executatu behar da. Kalkula ezazu lortuko den $speed_{avg}$.

2. 8 prozesadoreko makina paralelo bat erabilizten da aplikazio jakin bat executatzeko. Execuzioan zehat, 8 prozesadoreak batera 50 segundutan erabilizten dira, 4 prozesadore 20 segundutan, eta prozesadore bakarra 30 gainikarga bat ($overhead$) izan da, executazio-denboraren % 10a han zuzen ere. Kalkula ezazu lortu den azelerazio-faktorea ($speed_{avg}$) eta erginikortasuna.

3.

- Aplikazio paralelo jakin batean zati bat azorilik gabeko parcelitzat daretze, baina beste bat seriean executatu behar da 64 prozesadoreko makina bat erabilizten da aplikazioa executatzeko.

- a. Aplikazioen portera Andahl-en legearen arabera da. Kalkula itzazu, alde batetik, seriean executatutu behar den programaren zati maximoa, baldin eta erginikortasunak gutxientz % 50a izan behar badu, eta, bestetik, lortzeko den azelerazio-faktorea $= 0.9$ batza.

- b. Aplikazioen portera Gustafson-en legearenak bat datoz. Kalkula ezazu lortuko den azelerazio-faktorea $= 0.9$ denean.

4. Aplikazio jakin bat 32 prozesadoreko multiprozesadore batean executatu da. Kodaren % 10a 32 prozesadoreetan executatu da, eta prozesuen arteko komunikazioen kostua zehaztu ahal izan da: aztertzen kontuan harri gabe, kalkula itzazu lortu den azelerazio-faktorea ($speed_{avg}$) eta erginikortasuna

5. Aurreko aplikazioa sakonago aztertu da, eta prozesuen arteko komunikazioen kostua zehaztu ahal izan da: exekuzio-denboraren % 20 gehingoa 32 prozesadoreetan executatu den zatiari, eta % 10 gehingoa 16 prozesadoreetan executatu den zatiari. Kalkula itzazu lortu den azelerazio-faktorea ($speed_{avg}$) eta erginikortasuna.

6. Aplikazio jakin bat portarek bi zati ditu. Alde batetik, seriean executatu behar den S/I -ko zati bat, eta zati exekuziarreko 100 s behar diri, eta iterazio bakoizaren exekuzio-denbora 50 s da. Kalkula ezazu lortzen legotekin dator bat programaren portera? Makinak 16 prozesadore dituela suposatu.

7. 4 prozesadore artean begiria jakin baten 100 iterazio executatzeko paraleloan. Bat izan ezik, iterazio bakoitzaren exekuzio-denbora 1 s da. Beste iterazioaren exekuzio-denbora 20 s da. Kalkula itzazu lortzen den azelerazio-faktorea ($speed_{avg}$) bi kasu hauean: (a) banaketa estatikoa da, 2 atza prozesadoreko, ("static,2"); (b) banaketa dinamikoa da ("dynamic").

8. SMP motako makina batetik 4 prozesadore ditu. Makina horretan 8 atza independente executatu nahi dira. Atza bakoitzaren exekuzio-denbora (segunduko) honako hauek dira: 5, 10, 6, 15, 8, 3, 2 eta 1. Kalkulatu serieko exekuzio-denbora. Atzak paralleloan executatzeko direnean, kalkula ezazu lortzen den azelerazio-faktorea ($speed_{avg}$) bi kasu hauean: (a) banaketa estatikoa da, 2 atza prozesadoreko, ("static,2"); (b) banaketa dinamikoa da ("dynamic").

9.

Multiprozesadore batean honako kode zati hau executatu nahi da. Idatzi kode paraleloa OpenMP erabiliz.

```
min = INT_MAX;
for (i=0; i<N; i++) if (A[i] < min) min = A[i];
for (i=0; i<N; i++) A[i]=A[i]-min;
```

10. SMP motako multiprozesadore batean honako kode hau executatu nahi da:
- ```
printf("Sartu X-ren balioa");
scanf("%d",&x);
for (i=0; i<N; i++) A[i] = A[i] * X;
min = INT_MAX;
for (i=0; i<N; i++) if (B[i] < min) min = B[i];
for (i=0; i<N; i++) B[i] = B[i] - min;
for (i=0; i < N; i++) printf ("%d\n",B[i]);
```

Idatzi kode paraleloa OpenMP erabiliz. Begizta bat paraleloan executatzeko bidea, atazen banaketak dinamikoen izan behar du. Azaldu aldagaien esprun.

11.

Multiprozesadore batean honako kode zati hau executatu nahi da. Idatzi kode paraleloa OpenMP erabiliz.

```
batura = 0.0;
for (i=0; i<N; i++) batura = batura + A[i]*A[i];
bb = batura / N;
for (i=0; i<N; i++) A[i] = sqrt(A[i]+bb)+pow(bb,4);
```

12.

Honako programa honek irudi baten histograma kalkulatzeko. OpenMP erabiliz, idatzi dagokion kode paraleloa SMP motako makina batean executatzeko.

```
#define N 2048
#define M 2048
typedef struct {
 int i,j,histo[256];
} image;
image;
unsigned char im[N][M],
```

main () {  
 int i,j,histo[256];  
 struct image im,image;  
 for (i=0; i<256; i++) histo[i]=0;  
 for (i=0; i<in.image.h; i++)  
 for (j=0; j<in.image.w; j++)  
 histo[in.image.im[i][j]]++;  
}



## Sisteme paralleloen eraginkortasunak

Sisteme paralleloen helburua, programak P aldiak azkeneko exekutazioa da non, P erabiliko diren prozesadore kopuruak den. Serieloak de Paraleloak diren programak bi parametroren bidez komparatzen dira: azelerazio-faktorea (speed-up) eta eraginkortasuna (efficiency).

Azelerazio-faktorea: Gerbeak aldiak azkeneko exekutazioa, t<sub>s</sub> t<sub>sp</sub>-rekin

$$af = \frac{t_s}{t_{sp}}$$

Eraginkortasuna: Baloitako azelerazio-faktore maximoko eraginkortasuna  
erag = af/P

~~Maximotako~~ idealak horrela hauetako ibaikoak dira:

$$af = \frac{t_s}{t_{sp}} = P$$

$$erag = af/P = 1 (\%)$$

Baina hori normalean ez da lortzen programeak osin baitira %100-en paralleloa. Ondorioz, hala zati bat, f, ~~ez~~ paralleloak dantzean de, 1-f osin dantzean ibaikoak de. Ondorioz, exekuzioak denborak t<sub>sp</sub> korrek geratzen dira.

$$t_{sp} = f \cdot t_p + (1-f) \cdot t_s$$

Eta azelerazio-faktorea hori ibaikoak da:

$$af = \frac{t_s}{t_{sp}} = P / [f + P(1-f)]$$

Aandahl-en legearren arabera seriean exekutatzen den  
programmen eragine oso altua da. Baino horren arabera programme  
peroleak os zuten eskerako beldio.

Baino Gustafson-en legearren arabera programmek terminue  
kreditoen bide eta sareko satie konstante mantendu. I hazi  
ezindegia de auditoria.

$$t_s = (1-f)t_s + f t_{sp}$$

$$t_p = (1-f)t_s + f t_s = t_s \rightarrow af = t_s t_p = (1-f) + fP$$

$Z = \text{'Ergebnisse zuverlässig'}$

$B = \text{'Ergebnisse bereit'}$

$$n_1 = 350 \rightarrow 210 \text{ erglei } Z$$

$$n = 380 \rightarrow 240 \text{ erglei } B$$

$$\begin{cases} n_1 = 350 \\ X_1 = 210 \end{cases} \Rightarrow \hat{p} = 0.6$$

$$\begin{cases} n = 380 \\ X_2 = 240 \end{cases} \Rightarrow \hat{p}_2 = 0.63$$

$$\begin{cases} H_0: p_1 \geq p_2 \\ H_1: p_1 < p_2 \end{cases} \Leftrightarrow \begin{cases} H_0: p_1 - p_2 \geq 0 \\ H_1: p_1 - p_2 < 0 \end{cases}$$

$$1. n_1 \hat{p}_1 \geq 5 \Rightarrow 350 \cdot 0.6 \geq 5$$

$$n_2 \hat{p}_2 \geq 5 \Rightarrow 380 \cdot 0.63 \geq 5$$

? Biologische einschätzliche auffordern

$$\hat{p}_0 = \frac{n_1}{n_1 + n_2} \cdot \hat{p}_1 + \frac{n_2}{n_1 + n_2} \cdot \hat{p}_2 = 0.616$$

$$Z = \frac{\hat{p} - \hat{p}_0}{\sqrt{\hat{p}_0 \hat{p}_0 / (n_1 + n_2)}} \approx N(0, 1)$$

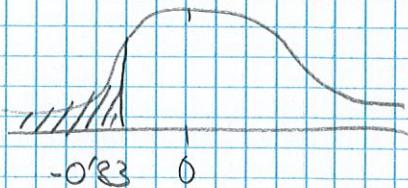
$$Z = \frac{0.6 - 0.63}{\sqrt{0.616 \cdot 0.384 \left( \frac{1}{350} + \frac{1}{380} \right)}} = -0.83$$

$Z_{\text{berechnet}}$

$$P = P(Z < Z_{\text{berechnet}}) = P(Z < -0.83) = \Phi(-0.83) =$$

$$= 1 - 0.7967 = 0.2032$$

Ho etwas cajingo da



152 2017/2018 2018

152 2017/2018 2018

152 2017/2018 2018

152 2017/2018 2018

## 6. Hypothesi-testen erabilesarcello edibiledi

$X_1 = \text{'Evaluierung soldati'}$

$X_2 = \text{'Gizoneeloen soldaten'}$

$X_1 \sim ?$

$X_2 \sim ?$

$$E_{X_1} = \mu_1$$

$$E_{X_2} = \mu_2$$

$$\text{VAR}(X_1) = \sigma^2$$

$$\text{VAR}(X_2) = \sigma^2$$

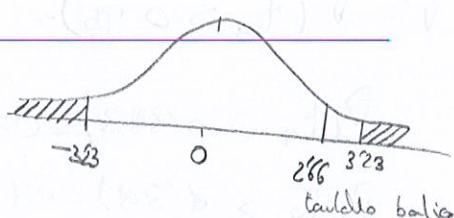
$$\left\{ \begin{array}{l} H_0: \mu_1 = \mu_2 \\ H_1: \mu_1 \neq \mu_2 \end{array} \right. \quad \left\{ \begin{array}{l} H_0: \mu_1 - \mu_2 = 0 \\ H_1: \mu_1 - \mu_2 \neq 0 \end{array} \right.$$

$$X_1, X_2 \sim N(,)$$

$$\sigma^2 = \sigma^2$$

logisch arbeiten

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{S_p^2}{n_1 + n_2 - 2}}} \sim t_{n_1 + n_2 - 2} \quad H_0 \text{ per ergänzen}$$



$$t_{\text{beob}} = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{S_p^2}{n_1 + n_2}}} = \frac{255 - 288}{\sqrt{\frac{1}{25} + \frac{1}{30}}} = -3.23$$

p\_berec

$$P = 2F(t_{53} > 3.23) < 0.005 \quad P(t_{60} > 2.66)$$

$$< 0.005 = 0.01$$

$$0.005$$

Adierazgarritasun-maila  $\alpha = 0.05$

Ondorioz,  $H_0$  batetutu. Beras, Bai, 95.25%-ko adierazgarritasun maila, desberdinak diren soldatuek.

## 5. Adibidene

$$X = \text{'durchs' } \quad EX = \mu$$

$$X \sim N(\mu, \sigma^2)$$

$$H_0: \mu \leq 20$$

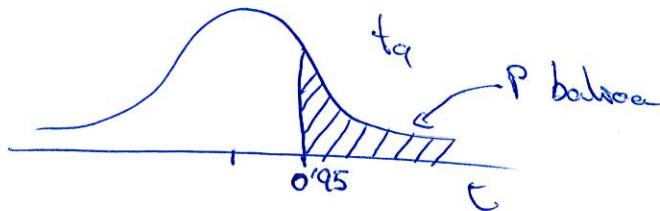
$$H_1: \mu > 20$$

$$t = \frac{\bar{X} - \mu_{20}}{S/\sqrt{n}} \sim t_{n-1}$$

$$\begin{aligned} S^2 &= \frac{1}{n} \sum_{i=1}^n x_i \\ S^2 &= \frac{n}{n-1} S_n^2 \\ S_n^2 &= \frac{1}{n} \sum_{i=1}^n x_i^2 - \bar{x}^2 \end{aligned}$$

$$\begin{cases} \sum x_i = 216 \\ \sum x_i^2 = 481 \end{cases}$$

$$t_{\text{theor}} = \frac{\bar{x} - \mu_0}{S/\sqrt{n}} = \frac{21.6 - 20}{5.68/\sqrt{10}} = 0.949$$



P balance

$$P = P(t_a \geq 0.95) =$$

$$0.1 < p < 0.20 \rightarrow \alpha = 0.05 / H_0 \text{ annehm}$$

$$P(t_a \geq 0.883) = 0.20$$

$$P(T_a \geq 1.38) = 0.1$$

## 8. Adibidene

$$X = \text{'Geraus'}$$

$$X \sim \text{Bernoulli}(p)$$

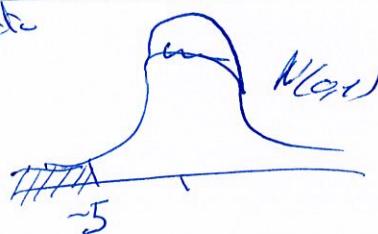
$$p = \text{geraus bei!}$$

Lage

$$\begin{aligned} \alpha &= 0.01 \\ \beta &= 0.8 \end{aligned}$$

$$\begin{aligned} H_0: \mu = \mu_0 &= 0.9 \\ H_1: \mu \neq \mu_0 &< 0.9 \\ \alpha = 0.01 & \leftarrow H_0 \text{ erreichbar} \\ Z = \frac{\mu - \mu_0}{\sqrt{\frac{p(1-p)}{n}}} &\sim N(0,1) \end{aligned}$$

$$\begin{aligned} P(Z < -5) &= P(Z > 5) = \\ P(Z > 2.5) &= 0.45 \xrightarrow{Z = 2.52 \cdot 10^{-3}} \end{aligned}$$



$$\frac{0.8 - 1}{\sqrt{\frac{0.8 \cdot 0.2}{10}}} = \cancel{-0.5} \cancel{-5}$$

Bigaten transpone

$$H_0: p = 0.5$$

$$H_1: p \neq 0.5$$

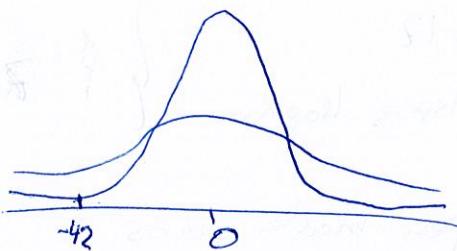
Laginetto informazion:

$$n = 100$$

$$\hat{p} = 0.29$$

$\alpha = 0.05$  fiktivische Dagu

$$z = \frac{\hat{p} - 0.5}{\sqrt{\frac{0.5}{100}}} = N(0,1)$$



$$z_{\text{beob}} = \frac{0.29 - 0.5}{\sqrt{\frac{0.5}{100}}} = -4.2$$

P beobach

$$P = 2P(z \geq 4.2) = 3 \cdot 10^{-5}$$

$\alpha = 0.05 \rightarrow P_{\text{beob}}$

$$\hat{p} = 3 \cdot 10^{-5}$$

Endfazete

## Log Transfome

$$\begin{cases} H_0: p=0.5 \\ H_1: p \neq 0.5 \end{cases}$$

Lagardelle Information

$$n = 12$$

Anpassig Klappe

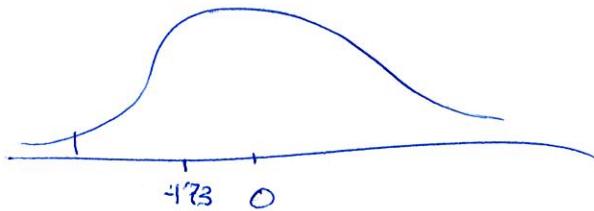
$$\hat{p} = \frac{5}{12} = 0.25$$

Adierospen maß = 0.65

$$\hat{p} \approx N(0.5, \frac{0.5 \cdot 0.5}{12}) \text{ Ho prau}$$

$$(n\hat{p}, n\hat{q} \geq 5)$$

$$z = \frac{\hat{p} - 0.5}{0.5/\sqrt{12}} \approx N(0, 1)$$



$$z_{\text{belieb}} = \frac{0.25 - 0.5}{0.5/\sqrt{12}} = -1.23$$

Pbelieb:

$$P = 2P(Z \geq 1.23) = 0.089$$

Etablierter Kritzaeho:

Pbelieb &

$$0.089 > 0.05 \Rightarrow H_0 \text{ ez bestent}$$

# HIPOTESI-TESTAK

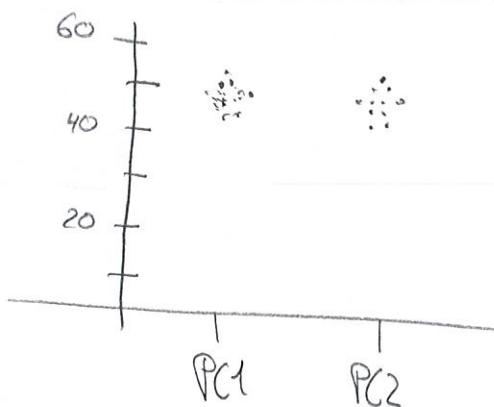
## Adibidea

Software berdina bi PC desberdinietan jarri da martxan. Bi PCtan exekuzio denborak berdinak diren ala ez neurtzeko 10 programa exekutatu dira PC bakoitzean. Hala, ondorengo taulan behatutako exekuzio denborak(s) eta beraien arteko diferentzia bildu dira.

| Programa | PC 1 | PC 2 | Diferentzia |
|----------|------|------|-------------|
| 1        | 49   | 48   | 1           |
| 2        | 52   | 49   | 3           |
| 3        | 51   | 49   | 2           |
| 4        | 47   | 50   | -3          |
| 5        | 51   | 48   | 3           |
| 6        | 49   | 51   | -2          |
| 7        | 50   | 49   | 1           |
| 8        | 49   | 51   | -2          |
| 9        | 49   | 51   | -2          |
| 10       | 52   | 52   | 0           |

Datu horiek eskuetan, bi PCtako exekuzio denboraren diferentzia batazbeste ez dela 0 esango al zenuke?

### Estadistika desribatzailea



$$\bar{X}_1 = 49.9 \quad M_{e_1} = 49.5 \quad S_1 = 1.59$$

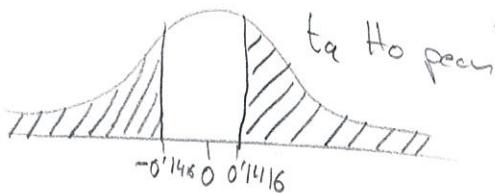
$$\bar{X}_2 = 49.8 \quad M_{e_2} = 49.5 \quad S_2 = 1.40$$

$$X_1 \rightarrow E_{X_1} = \mu$$

$$X_2 \rightarrow E_{X_2} = \mu_2 \quad \mu_1 - \mu_2 = 0?$$

$$\text{Eraiki } D = X_1 - X_2$$

$$ED = \mu_D = \mu_1 - \mu_2$$



$$\begin{cases} H_0: \mu_D = 0 \\ H_1: \mu_D \neq 0 \end{cases} \quad n = 10 \quad \sum d_i = 1 \quad \bar{D} = 0.1$$

Suposatzeko: Dibundutako normala jarraitzen

$$t = \frac{\bar{D} - 0}{\frac{s_D}{\sqrt{n}}} \sim t_9 \quad H_0 \text{ peman}$$

$$P = 2P(t_9 > 0.1416) \Rightarrow 0.8 < P < 0.9$$

$\Downarrow \alpha = 0.05$   
 $H_0$  onartu

$$t_{\text{bekt}} = \frac{0.1 - 0}{\frac{1.40}{\sqrt{10}}} = 0.1416$$

