Euskal Herriko Unibertsitatea
Informatika Fakultatea

Konputagailuen Arkitektura eta Teknologia Saila

Konputagailuen Arkitektura I

Memoria-sistema 1 (ebazpena): Orriztaketa + modulu tartekatuak

Konputagailu baten memoria-sistemak 4 byteko hitzak erabiltzen ditu eta bere helbideratze-unitatea bytea da. Memoria-sistema honen ezaugarriak honako hauek dira:

- **Alegiazko memoria:** 2 MB (megabyte)-ko memoria orriztatua, 256 byteko orriekin. Helbideen itzulpena egiteko TLBa erabiltzen da. Honen atzipen-denbora 20 ziklokoa da hutsegitean eta ziklo batekoa asmatzean. Hasieran, TLBa hutsik dago.
- **Memoria nagusia:** 256 kB-koa da eta 4 modulu tartekatuz osatua dago. Atzipen denbora 10 ziklokoa da (ziklo bat tartekatze-bufferretik).

Konputagailu honetan honako programa hau exekutatzen da:

for (i=0;i<252;i++)

B[i]=(A[i]*A[i+4])+C[0];

```
                movi r1,#1520
                movi r2,#0
                movi r3,#251
begizta:        load r5,A[r2]
                load r6,A[r2+16]
                mul r5,r5,r6
                load r10,[r1]
                add r6,r6,r10
                store r6,B[r2]
                addi r2,r2,#4
                subi r3,r3,#1
                bge r3,begizta
```

Programa 320 helbide logikotik aurrera metatua dago. A bektorea 2048 helbide logikoan hasten da, B bektorea 512 helbide logikoan eta C bektorea 1520 helbide logikoan. Bai aginduak eta bai bektoreen osagaiak hitz batekoak dira.

a) Marraz ezazu memoria-sistemaren egitura, garbi azalduz helbidearen eremu desberdinak zertarako erabiltzen diren, bai alegiazko memorian, baita memoria nagusian ere. Adierazi orri-taularen sarrera-kopurua eta sarrera bakoitzaren neurria. TLBa kontuan hartuta, zein da sarrera baten tamaina?

b) Itzuli programak begiztaren lehen pasaldian sortzen dituen memoriako helbide guztiak eta kalkulatu helbide bakoitzerako memoria-sistema atzitzeko behar den denbora. Erabili ezazu taula bat egiten dituzun pausu guztiak bertan garbi azalduz.

c) Kalkula ezazu programa osoaren exekuzioan memoria-hierarkia atzitzeko behar den denbora (helbideak itzultzeko behar den denbora + memoria nagusia atzitzeko behar den denbora).

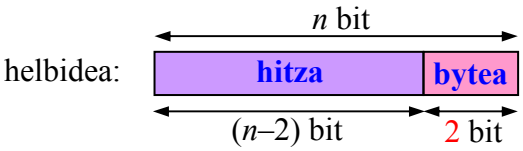
Orri-taulako informazioa:

Orri logikoa:	1	8	5	2	9	3	10	4	11	....
Orri fisikoa:	2	0	5	1	15	20	17	10	3	....

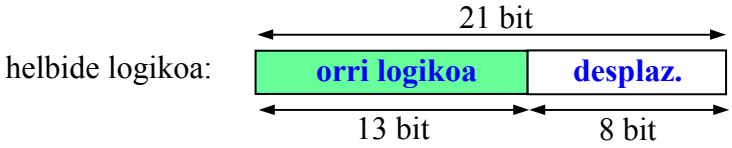
Ebazpena

Lehenik, memoria-sistemaren ezaugarriak ongi aztertu behar dira enuntziatuan, behar den informazio guztia ondorioztatzeko.

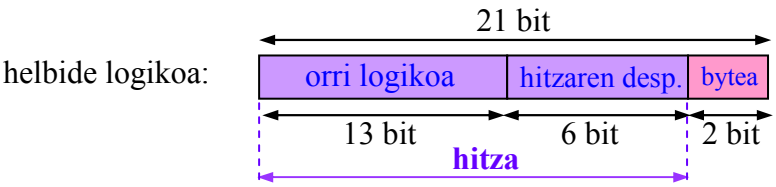
Hala, kontuan hartu beharko dugu memoria-sistema honek **4 byteko hitzak** erabiltzen dituela, eta **helbideratze-unitatea bytea** dela, hau da, posible dela hitz baten byteak banan-banan eskuratzea edo bereiztea. Hori dela eta, helbidearen bitak bi eremutan banatuta egongo dira: alde batetik, atzitu nahi den **hitza** adierazten duten bitak, eta, beste aldetik, hitz horren barruan atzitu nahi den **byte zehatza** adierazten dutenak. Hitz baten barruan 4 byte daudenez, 2 bit beharko dira byte bakoitza adierazteko ($2^2 = 4$ delako). Enuntziatuan aurrera irakurtzen jarraitu beharko dugu, jakin ahal izateko zein den memoriaren tamaina, eta zenbat bit behar diren helbideak adierazteko, baina, oraingoz, suposatzen badugu n bit behar direla, honelaxe geratuko da helbidearen eskema:



Jarrai dezagun enuntziatua irakurtzen. **Alegiazko memoria**ren ezaugarriak aipatzen dira aurrekoaren ondoren. Alde batetik, memoria birtuala 2 MB-koa (megabytekoa) da, eta orriztatua dagoela esaten digute, orrien tamaina 256 bytekoa izanik. Informazio horretatik gauza asko ondorioztatu ditzakegu **helbide logiko**en egiturari buruz. Alde batetik, byteen helbide logikoak emateko 21 bit behar direla, $2 \text{ MB} = 2 \times 2^{20} = 2^{21}$ delako. Beste aldetik, alegiazko memoria orriztatua dagoenez, helbide logikoak bi eremu izango ditu: **orri logikoa** adierazten duten bitak, eta atzitu nahi dugun byteak orri logikoaren barruan duen **desplazamendua** adierazten dutenak. Eta orri logikoak 256 bytekoak direnez, 8 bit behar dira byteen desplazamendua adierazteko ($256 = 2^8$). Ondorioz, $21 - 8 = 13$ bit izango ditugu orri logikoa adierazteko (hortaz, programa batek $2^{13} = 8192$ orri izan ditzake). Honelaxe geratuko da helbide logikoaren eskema:

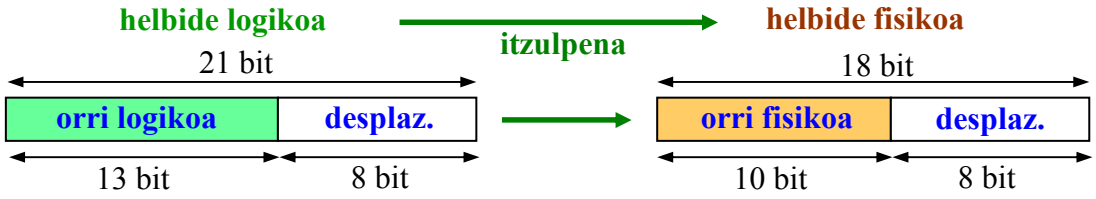


Ezer esan ez badugu ere, aurreko helbide logiko hori byterako helbidea da, hori delako helbideratze-unitatea. Hortaz, hasieran esandakoa kontuan hartzen badugu, alegia, hitzaren barruko byte zehatza adierazteko 2 bit behar direla eta besteek hitza adierazten dutela, kasu honetan hiru eremu hauek ere bereizi ahal izango ditugu: orri logikoa, hitzaren desplazamendua orriaren barruan (6 bit), eta bytea (2 bit). Honelaxe:



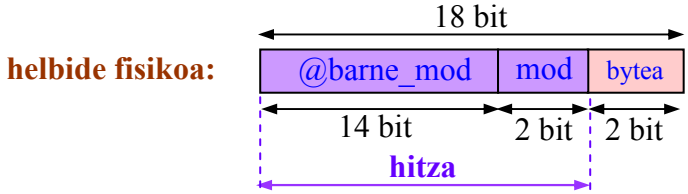
Dena den, ariketak egiteko azken aukera hori baztertuko dugu, helbide logikoaren itzulpena egitean bytea eta hitza bereizteak ez duelako gehitzen informazio berezirik.

Memoria nagusiari dagokionez, jakin badakigu 256 kB-koa (kilobytekoa) dela. Hala, **helbide fisikoa** adierazteko 18 bit behar dira (256 kB = $2^8 \times 2^{10} = 2^{18}$). Hortaz, helbide logikoa helbide fisiko bihurtzeko prozesuan, 21 bit 18 bihurtuko dira. Orri logikoak eta orri fisikoak tamaina berdinekoak direnez, bietan desplazamendua berdina izango da, eta laburragoa izango dena orri fisikoa adierazteko erabilitako bit-kopurua izango da, 10 bitekoa (18– 8) izango delako (hortaz, memoria nagusiak 2^{10} = 1024 orri fisiko izango ditu). Honelaxe egingo da itzulpena:

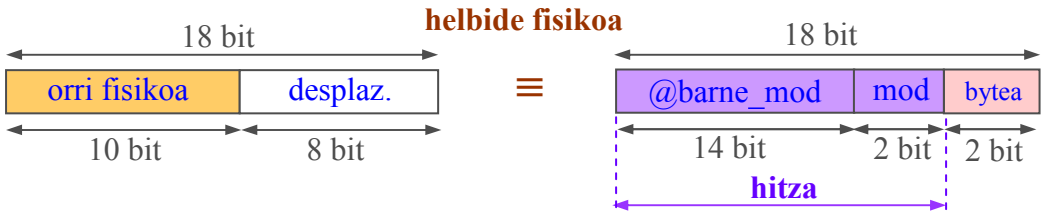


Badakigu itzulpen hori egiteko TLBa daukala sistemak, eta aurreraxeago irudikatuko dugu itzulpena egiteko behar den hardwarearen eskema, (a) atalean, hain zuzen ere.

Itzulpen-prozesuaren ondorioz lortzen den helbide fisiko hori memoria nagusira doa, nahi den hitza eskuratzeko. Eta hemen bai, hitza eta bytea bereiziko ditugu, bereizketa horri esker kalkulu batzuk errazago egingo baititugu. Beraz, hasieran esan bezala, 18 biteko helbide fisikoko pisu txikieneko 2 bitek adieraziko dute hitzaren barruko byte zehatza, eta, ondorioz, beste 16 bitek adieraziko dute hitza. Beste aldetik, hitza adierazten duen eremuan beste bi eremu bereizi behar ditugu: alde batetik, memoria nagusia 4 modulu tartekatuk osatzen dutenez, bilatzen ari garen hitza zein modulutan dagoen adierazten duen eremua beharko dugu (**mod**, 2 bitekoa, 2^2 = 4 delako, 4 izanik modulu kopurua, hain zuzen), eta, beste aldetik, modulu horren barruan hitza zein helbide zehatzetan dagoen ere (**@barne_mod**, 16 – 2 = 14 bitekoa). Honelaxe geratuko da helbide fisikoaren egitura:



Argibide-oharra: Aurreko bi irudiak ikusita, badirudi bi helbide fisiko desberdin ditugula. Alde batetik, itzulpen-prozesutik ateratzen dena, eta, beste aldetik, memoria nagusira doana. Baina, esan bezala, biak gauza bera dira, hau da, itzulpen-prozesutik ateratzen diren 18 bitak dira memoria nagusira doazenak; aldatzen den gauza bakarra zera da: bitei ematen zaien interpretazioa, itzulpenaren ikuspuntutik ala memoria nagusiaren ikuspuntutik, eremu batekoak ala bestekoak izan, besterik gabe. Beraz, biak guztiz baliokideak dira.



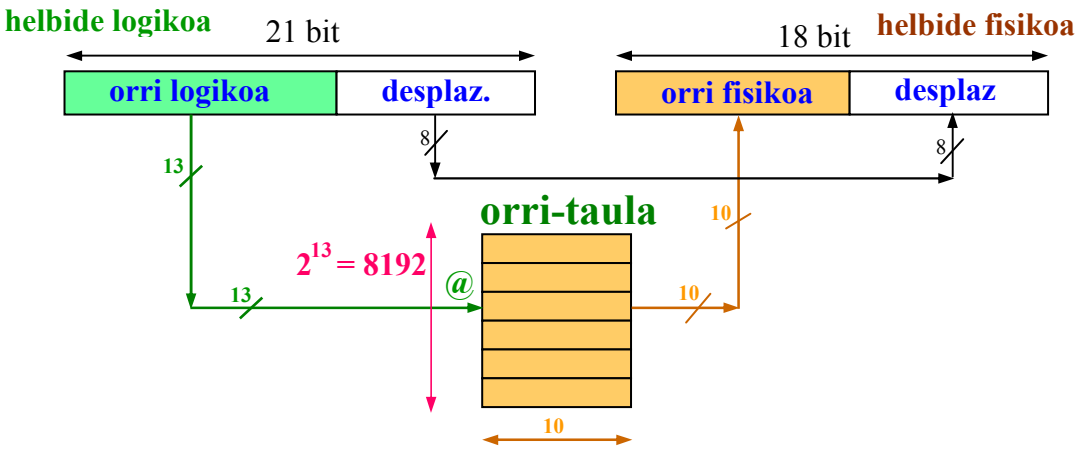
Enuntziatuaren hasierako lerroetatik informazio hori guztia erauzi ondoren, onena dugu hemendik aurrera atalez atal joatea.

(a) Memoria-sistemaren egitura irudikatu behar dugu, agerian uzteko nola erabiltzen diren helbide logikoaren eta helbide fisikoaren eremuetako bitak kasu bakoitzean.

Itzulpenari dagokionez, badakigu orri logikoa adierazten duten bitak direla orri-taulako helbide-sarrerara bideratu behar direnak, bertan begiratu ahal izateko zein orri fisikotan kokatu duen sistemak orri logiko hori memoria nagusian kargatu duenean. Desplazamendua, berriz, orri logikoaren barruan zein orri fisikoaren barruan berdina denez, berau adierazten duten bitak berdinak izango dira helbide logikoan zein fisikoan.

Orri-taula memoria arrunta denez gero, helbide logikotik jasotzen dituen 13 bitak helbide gisa erabiltzen dira bilaketarako; hortaz, orri-taulak $2^{13} = 8192$ posizio edo sarrera izango ditu (orri logiko adina, hain zuzen ere, orri logiko bakoitzari orri-taulako sarrera bana dagokiolako). Beste aldetik, orri-taulako posizio edo sarrera bakoitzean orri fisikoa adierazten duten 10 bitak daude.

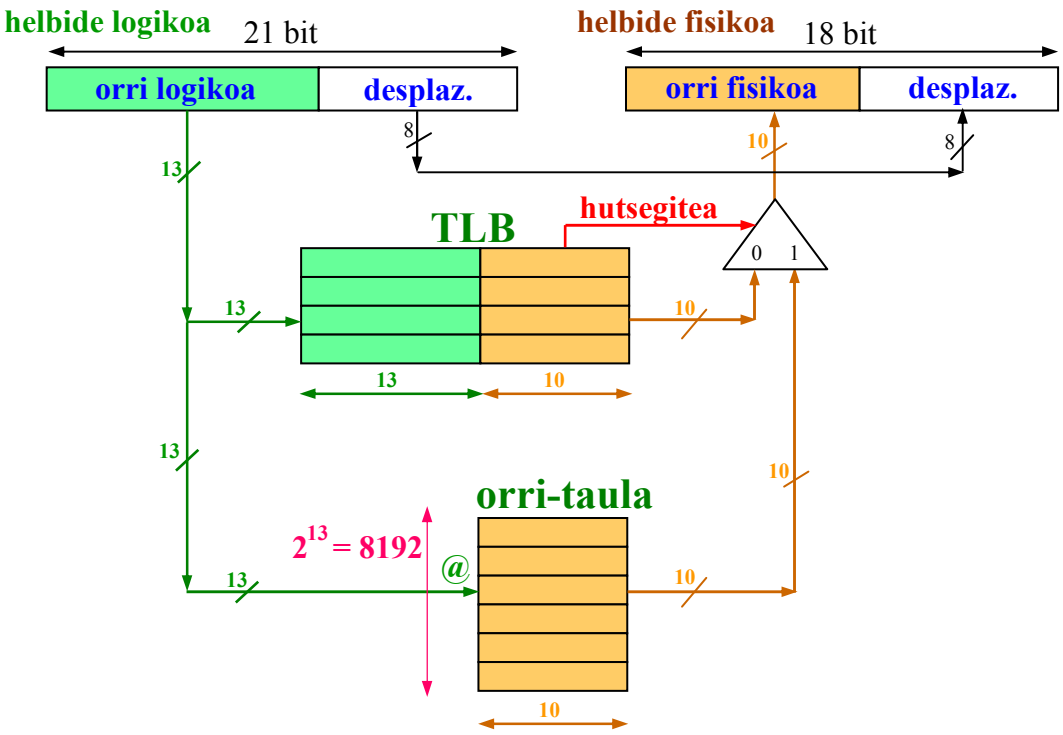
Hala, honelaxe geratzen da "helbide logiko → helbide fisiko" itzulpena egiteaz arduratzen den hardwarearen eskema:



Baina enuntziatuan esandakoaren arabera, sistema honetan, itzulpenak azkartzeko, TLBa dago, haren ezaugarriei esker azkarrago irakurtzen baita TLBan orri-taulan baino. Hala, helbide logiko bat itzuli behar denean, dagokion helbide fisikoa lortzeko, lehenbizi TLBan egiten da bilaketa: helbide logiko horri dagokion orri logikoaren zenbakia lehenengo aldiz agertzen bada prozesadoreak sortutako helbide-segidan, orduan TLBan ez dago horri buruzko informaziorik, eta horregatik hutsegitea dela erabakiko du; orduan, orri logiko hori zein orri fisikotan dagoen kargatuta jakin ahal izateko orri-taulan irakurri beharko du sistemak, eta, horrekin batera TLBan idatziko du orri logiko horri dagokion orri fisikoaren zenbakia. Hala, hurrengo bilaketa batean, orri logiko horren erreferentzia jadanik egongo da TLBan; ondorioz, honek, bilaketaren emaitza asmatzea dela ikusiko du eta haren irteeran orri logiko horri dagokion orri fisikoaren zenbakia adierazten dituzten bitak emango ditu, oso denbora laburrean.

TLBa memoria asoziatiboa denez, edukiaren bitartez egiten ditu bilaketak. Hori dela eta, bilatzeko jasotzen dituen 13 bitek ez dute helbidea osatzen, orri-taulan gertatzen den bezala, baizik eta edukiaren zati bat. Hala, TLBaren posizio bakoitzeko edukiaren zati bat 13 bitekoa izango da, bilatu beharreko orri logikoaren zenbakiari dagokiona, hain zuzen; beste zatia, berriz, 10 bitekoa izango da, bilatzen ari garen orri logikoari dagokion orri fisikoaren zenbakia adierazten duena, eta bigarren zati hau da TLBaren irteera gisa helbide fisikora joango dena. Hortaz, TLBaren posizio edo sarrera bakoitzaren tamaina 23 bitekoa da (13 + 10). Baina TLBaren sarrera-kopuruak ez du zerikusirik bilatzeko jasotzen duen bit-kopuruarekin; printzipioz, TLBak edozein sarrera-kopuru izan dezake, eta hori sistemaren diseinatzaileak erabakitzen du.

Honelaxe geratzen da TLBa duen hardwarearen eskema:



Behin helbide fisikoa lortuta, memoria nagusiaren egitura aztertu beharko dugu.

Memoria nagusia 256 kB-koa izanik (2^{18} byte), eta hitza 4 (2^2) bytekoa denez gero, esan dezakegu memoria nagusian 65536 (2^{16}) hitz sartzen direla:

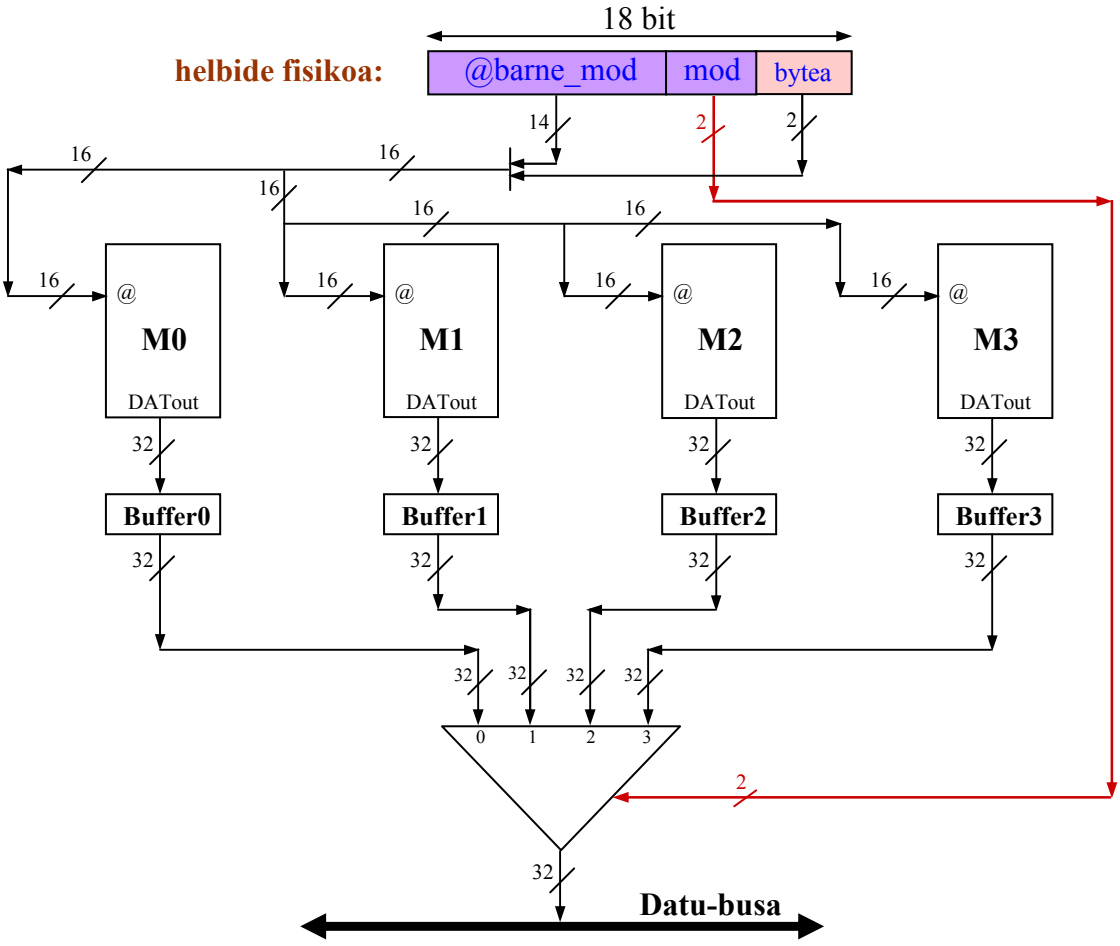
$$\frac{256\ \text{kB}}{4\ \text{byte/hitz}} = \frac{2^{18}\ \text{byte}}{2^2\ \text{byte/hitz}} = 2^{16}\ \text{hitz} = 65536\ \text{hitz}$$

Hala berean, 4 (2^2) moduluk osatzen dutenez memoria nagusia, modulu bakoitza 64 kB-koa da, edo 16384 (2^{14}) hitzekoa:

$$\frac{256\ \text{kB}}{4\ \text{modulu}} = 64\ \text{kB/modulu}$$

$$\frac{2^{16}\ \text{hitz}}{2^2\ \text{modulu}} = 2^{14}\ \text{hitz/modulu} = 16384\ \text{hitz/modulu}$$

Honelaxe geratzen da memoria nagusiaren hardwarearen eskema:



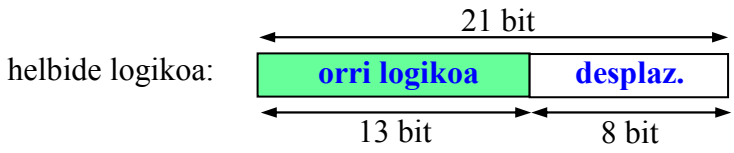
Memoriaren irudiko egiturari erreparatuz, modulu tartekatuen funtzionamendua ondorioztatzen da: modulu guztietan aldi berean irakurtzen dira barne-helbide bera duten hitzak, eta tartekatze-bufferretan metatzen dira. Horri esker, hurrengo atzipenean hitz horietako bat behar izanez gero, ez da beharrezkoa moduluetan irakurtzea, jadanik bufferretan dagoelako informazioa, eta atzipen-denbora asko murrizten da, hurrengo atalean ikusiko dugun legez.

(b) Ematen diguten programa exekutatzean, begiztaren lehenengo pasaldian sortzen diren helbideak nola itzultzen diren, zenbat denbora behar den itzulpena egiteko, eta zenbat denbora behar den memoria nagusiko posizioak atzitzeko kalkulatu behar dugu. Hori guztia modu ordenatuan egiteko, pausoz pauso joango gara eta taula batean laburbilduko ditugu pausoetako emaitzak.

Horretarako, kontuan hartu behar dugu zein diren aginduen eta datuen helbide logikoak, horiek baitira prozesadoreak helbide-busean kokatuko dituen balioak unean uneko agindua edo datua atzitzeko. Hala, enuntziatuan esandakoaren arabera, programa (aginduak, alegia) 320 helbide logikotik aurrera dago. A bektorea, 2048 helbide logikoan hasten da (hau da, helbide horretan A[0] osagaia dago, eta hortik aurrera beste osagai guztiak, bakoitzak 4 posizio okupatuz, osagaien tamaina hitz batekoa delako, 4 bytekoa, beraz); B bektorea, 512 helbide logikoan; eta C bektorea, 1520 helbide logikoan.

Azter ditzagun orain programaren aginduak banan-banan, ikusteko zein helbide logiko sortuko duen prozesadoreak une bakoitzean eta nola burutuko den prozesu guztia memoria nagusia atzitu eta agindua edo datua eskuratu arte.

Lehenengo agindua, `movi r1,#1520`, esan digutenaren arabera, 320 helbide logikoan dago, beraz, hori da prozesadoreak helbide busean kokatuko duen balioa agindu hori atzitu nahi duenean. Badakigu memoria nagusia atzitu ahal izateko lehenengo urratsa dela hardwareak helbide logiko horren itzulpena egitea, dagokion helbide fisikoa lortzeko. Horretarako, hardwarean, helbide logiko horren bitak bi eremutan daude banatuta: orri logikoa eta desplazamendua.



Hortaz, 320 zenbakia bitarrez adieraziko bagenu, pisu handieneko 13 bitek orri logikoa adieraziko lukete eta pisu txikieneko 8 bitek, berriz, desplazamendua. Baina zenbaki hamartarrak bitarrez adierazten ibiltzea nekeza denez, kalkulu guztiak hamartarrez egingo ditugu. Horretarako, arazo honen irtenbidea topatu behar dugu: nola bereizi, zenbaki hamartarrean, bitek osatutako eremuak? Erantzuna sinplea da, nahikoa da gogoratzea zenbaki bitar bat posizio bat eskuinerantz desplazatzean, birekin zatitzen dela, hau da, desplazatutako posizio bakoitzarekin "zati 2" eragiketa egiten dela, eta eskuinetatik galtzen diren bitek zatiketaren hondarra osatzen dutela.

Adibidez, demagun 38 zenbaki hamartarra bitarrez daukagula adierazita, 6 bitetan, eta bertan bi eremu bereizi behar ditugula; alde batetik, pisu handieneko 2 bitek gauza bat adierazten dute (une honetan berdin zaigu zer adierazten duten: orri logikoa izan zitekeen, edo hitza, edo modulua…), eta pisu txikieneko 4 bitek beste zerbait adierazten dute (desplazamendua, esaterako):

$$\text{Zenbakia} = 38_{(10} = \texttt{100110}$$

Demagun 4 posizio desplazatzen dugula eskuinerantz (ezkerretatik 0koak sartuz, 6 biteko adierazpidea mantentzeko). Hau da lortzen den emaitza: `000010`, eta eskuinetatik galdu diren bitak: `0110`, Bi emaitza hauek hamartarrez adierazita 2 eta 6 zenbakiak dira, hurrenez hurren.

Hasierako zenbakia eskuinerantz 4 aldiz desplazatu dugunez, "zati 2" egin dugu 4 aldiz, hau da (((38/2)/2)/2)/2 egin dugu, hots, $38/16 = 38/2^4$. Zatiketa hori egiten badugu, ikusiko dugu ez dela zehatza ateratzen: 38/16 = 2,375 baita.

Zatidura osoa hartzen badugu 2 ateratzen da, eta falta dena, 0,375 × 16 = 6, hondarra da. Eta bi horiek dira 38 zenbakia bitarrez desplazatzean lortu ditugun balioak, hain zuzen ere.

Matematikoki, honelaxe adierazten da:

Pisu handieneko bi bitek osatzen duten eremuaren balioa: $38 \texttt{ div } 2^4 = 2$

Pisu txikieneko 4 bitek osatzen duten eremuaren balioa: $38 \texttt{ mod } 2^4 = 6$

Hortaz, 320 zenbakia bitarrez adieraziko bagenu, orri logikoaren zenbakia kalkulatzeko 8 aldiz desplazatu beharko genuke eskuinerantz. Hau da, 320 `div` 2^8 egin behar dugu orri logikoaren zenbakia kalkulatzeko, eta desplazamendua hondarra izango da: 320 `mod` 2^8.

@logikoa = 320 → orri logikoa = 320 `div` 2^8 = 320 `div` 256 = 1

desplazamendua = 320 `mod` 2^8 = 320 `mod` 256 = 64

Ekuazio matematiko orokorrak ondorioztatu daitezke:

orri logikoa = @logikoa `div` orriaren tamaina bytetan

desplazamendua = @logikoa `mod` orriaren tamaina bytetan

Eta prozesadoreak sortuko dituen helbide logiko guztiei aplikatuko dizkiegu.

Ariketaren ebazpena modu ordenatuan egiteko, esan dugu taula batean bilduko ditugula pauso guztien emaitza guztiak. Has gaitezen oraintxe bertan:

Agindua	Helbide logikoa *@logikoa*	orri logikoa *o.l.*	desplazamendua *d*
`movi r1,#1520`	320	1	64

Jarrai dezagun orain lehenengo aginduaren itzulpen-prozesuarekin. Behin helbide logikoaren eremuak lortu ondoren, orain ikus dezakegu TLBak orri logiko horren zenbakia (1) bilatuko duela bere edukien artean. Aurkituz gero, esaten da asmatzea izan dela, eta orduan hurrengo zikloan TLBaren irteeran orri logiko horri dagokion orri fisikoaren zenbakia izango dugu. Ez badu aurkitzen, berriz, hutsegitea dela esaten da, eta horrek esan nahi du bilatzen ari den orri logiko hori ez dela oraindik erreferentziatu, hau da, hori dela orri logiko horren lehenengo agerpena prozesadoreak sortzen dituen helbide logikoen artean; horregatik, 20 ziklo beharko dira orri-taulan bilatzeko zein orri fisikotan dagoen kargatuta orri logiko hori. Lehenengo aginduaren kasuan, bilaketaren emaitza hutsegitea izango da, programaren hasieran TLBa hutsik baitago. Hortaz, 20 ziklo behar dira lehenengo aginduaren itzulpena burutzeko ($t_{itzulp.}$). Orri-taulan begiratuko dugu jakiteko orri logiko hori zein orri fisikotan kargatu duen sistemak. Kasu honetan, 1 orri logikoa 2 orri fisikoan (*o.f.*) dago kokatuta.

Agindua	*@l*	*o.l.*	*d*	$t_{itzulp.}$	*o.f.*	*d*
`movi r1,#1520`	320	1	64	20	2	64

Orri fisikoaren zenbakia ezaguna denean, helbide fisikoa lortu behar dugu. Horretarako, badakigu zer egiten den helbide fisikoa bitarrez osatzeko: orri fisikoa adierazten duten bitak ezkerraldean kokatu, eta eskuinaldean desplazamenduaren bitak elkartu. Baina, nola egin hamartarrez? Orri fisikoa adierazten duten bitak desplazamendua adierazten duten 8 biten ezkerraldean kokatzeak zera esan nahi du: ezkerretara desplazatu ditugula 8 posizio; beraz, orri fisikoa 2^8rekin biderkatu behar dugu. Desplazamendua adierazten duten bitak aurrekoen eskuinaldean elkartzeak zera esan nahi du: batu behar dugula adierazten duten balioa.

Badaezpada, zalantzarik ez izateko, gogora dezagun 38 zenbakiaren adibidea, baina alderantzizko ikuspuntutik. Hau da: $38 = 2 \times 2^4 + 6$.

Beraz, honelaxe kalkulatuko dugu 320 helbide logikoari dagokion helbide fisikoa (@*f*):

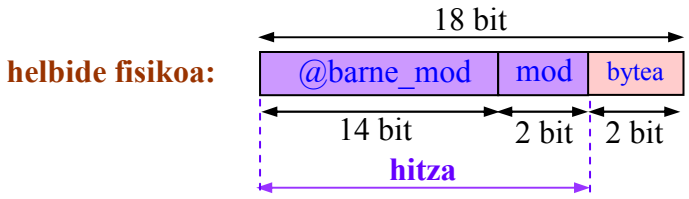
$$@\text{fisikoa} = 2 \times 2^8 + 64 = 576$$

Eta ekuazio orokorra ere ondorioztatzen dugu:

@fisikoa = orri fisikoa × orriaren tamaina bytetan + desplazamendua

Agindua	@*l*	*o.l.*	*d*	$t_{itzulp.}$	*o.f.*	*d*	@*f*
`movi r1,#1520`	320	1	64	20	2	64	576

Behin helbide fisikoa ezaguna denean, iritsi da unea memoria nagusia atzitzeko. Eta horretarako kontuan hartu beharko dugu zein modulutan dagoen helbide fisiko hori eta zein barne-helbidetan moduluaren barruan. Horretarako, gogora dezagun zein diren helbide fisikoaren bit-eremuak:



Kasu honetan, hiru eremu daudenez, banan-banan egingo ditugu eremuen arteko bereizketak. Lehenik, eskuineko eremua (hitzaren barruko byte zehatza adierazten duten 2 bitak) "kenduko" dugu; hau da, ezer baino lehen, helbide fisiko horrek zein hitz adierazten duen kalkulatuko dugu. Badakigu nola egin hori bitarrez: helbide fisikoa 2 posizio desplazatuz eskuinerantz; eta badakigu baita ere nola egin hamartarrez: hitza = helbide fisikoa `div` 2^2 eginez. Hondarra (helbide fisikoa `mod` 2^2) hitzaren barruan bilatu nahi dugun byte zehatza izango da. Oro har, hitz osoak bilatzen direnean, hondarra 0 izango da.

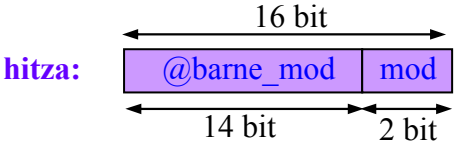
Kasu honetan, beraz: hitza = 576 div 2^2 = 576 div 4 = 144

Lehen bezala, ekuazio orokorra ondorioztatzen dugu:

hitza = @fisikoa `div` hitzaren tamaina bytetan

Agindua	@*l*	*o.l.*	*d*	$t_{itzulp.}$	*o.f.*	*d*	@*f*	hitza
`movi r1,#1520`	320	1	64	20	2	64	576	144

Behin hitza kalkulatuta dagoela, orain berari dagozkion bi eremuak bereizi behar ditugu: modulua eta moduluaren barneko helbidea (@barne-mod) hain zuzen ere:



Desplazamenduen bidezko zatiketak eginez, hau da ondorioztatzen duguna:

@barne_mod = hitza `div` 2^2 = hitza `div` 4 = 144 `div` 4 = 36

mod = hitza `mod` 2^2 = hitza `mod` 4 = 144 `mod` 4 = 0

Hemen ere, ekuazio orokorra ondorioztatzen dugu:

@barne_mod = hitza `div` modulu-kopurua
mod = hitza `mod` modulu-kopurua

Agindua	@ℓ	o.ℓ.	d	$t_{itzulp.}$	o.f.	d	@f	hitza	@barne_mod	mod
`movi r1,#1520`	320	1	64	20	2	64	576	144	36	0

Prozesu osoari amaiera emateko, gauza bakarra falta da: adierazi behar dugu zenbat denbora behar den hitz hori eskuratzeko memoria nagusitik ($t_{atzip.}$). Horretarako, kontuan hartu behar dugu memoriaren atzipen-denbora 10 ziklokoa dela, baina eskuratu nahi dugun hitza jadanik tartekatze-bufferrean baldin badago, orduan ziklo bakarra baino ez da behar. Eta noiz egongo da hitza tartekatze-bufferrean? Erantzuna oso logikoa da: aldez aurretik moduluaren barruko helbide bera irakurri denean, modulu guztietan aldi berean irakurtzen delako. Lehenengo aginduaren kasuan, beraz, 10 zikloko atzipena izango da, programaren exekuzioan hori delako atzitzen den lehenengo helbidea. Hortaz, lehenengo aginduari dagokionez, honelaxe geratzen da taula:

@ℓ	o.ℓ.	d	$t_{itzulp.}$	o.f.	d	@f	hitza	@barne_mod	mod	$t_{atzip.}$
320	1	64	20	2	64	576	144	36	0	10

Agindu horren exekuzioa amaitzean, prozesadoreak `movi r2,#0` aginduaren helbidea kokatuko du helbide busean; aginduak hitz batekoak direnez, agindu bakoitzak 4 byte betetzen ditu; beste aldetik, helbideratze-unitatea bytea denez, bigarren aginduari 324 helbide logikoa dagokio (aurreko aginduaren helbide logikoa (320) gehi 4 posizio, aurreko aginduak betetzen dituenak, alegia). Lehen egindako kalkulu guztiak errepikatuz, honelaxe geratzen da bigarren aginduari dagokion lerroa taulan:

@ℓ	o.ℓ.	d	$t_{itzulp.}$	o.f.	d	@f	hitza	@barne_mod	mod	$t_{atzip.}$
320	1	64	20	2	64	576	144	36	0	10
324	1	68	**1**	2	68	580	145	36	1	**1**

Kasu honetan, bi gauza azpimarratu behar ditugu: alde batetik, itzulpen-denbora ziklo bakar batekoa dela 1 orri logikoa jadanik atzitu delako aurreko erreferentzian; horregatik, TLBak bilaketa egin duenean, topatu du eta asmatzea izan da. Beste aldetik, memoria nagusia atzitu denean, atzipen-denbora ere ziklo bakar batekoa izan da, moduluaren barruko 36 helbideari dagozkion hitzak jadanik zeudelako tartekatze-buferretan, aurreko aginduan horixe izan baita irakurritako helbidea.

Horren ondoren, 3. agindua `movi r3,#251` da, eta 328 helbide logikoan dago (aurrekoa + 4). Beraz, honelaxe geratuko da taulako 3. lerroa:

@ℓ	o.ℓ.	d	$t_{itzulp.}$	o.f.	d	@f	hitza	@barne_mod	mod	$t_{atzip.}$
320	1	64	20	2	64	576	144	36	0	10
324	1	68	1	2	68	580	145	36	1	1
328	1	72	1	2	72	584	146	36	2	1

4. agindua, `load r5,A[r2]`, begiztako lehenengoa da eta 332 posizioan dago; baina agindu honetan aurreko hiruretan gertatu ez den zerbait gertatuko da: aurreko hiru aginduek berehalako datua zeramaten kodean bertan (1520, 0 eta 251 balioak, hain zuzen); 4. aginduan, berriz, datua memorian dago, eta helbideratze indizeduna erabiltzen du aginduak datu hori eskuratzeko. Hau da, aginduan `A` bektorearen oinarri-helbidea (2048) dator kodetuta, eta indize-erregistro bat (`r2`) erabiltzen da datuaren benetako helbidea kalkulatzeko —begiztaren pasaldietan unean uneko `A[i]` osagaia atzitzen joateko, `r2` erregistroan `i` indizearen balioa metatuta izanik—. Helbideratze mota horretan, bi parametroen batuketa egin behar da; kasu honetan, begiztaren lehenengo pasaldian `r2` = 0 da (2. aginduaren exekuzioaren ondorioz), beraz: 2048 + 0 = 2048 helbide logikoa sortuko du prozesadoreak datua atzitzeko —`A[0]` osagaia, hain zuzen, begiztaren lehenengo pasaldian—. Hortaz, 4. aginduari bi lerro dagozkio taulan, bat aginduari berari, eta beste bat datuari, honelaxe:

$@\ell$	$o.\ell.$	d	$t_{itzulp.}$	$o.f.$	d	$@f$	hitza	@barne_mod	mod	$t_{atzip.}$
320	1	64	20	2	64	576	144	36	0	10
324	1	68	1	2	68	580	145	36	1	1
328	1	72	1	2	72	584	146	36	2	1
332	1	76	1	2	76	588	147	36	3	1
2048	8	0	20	0	0	0	0	0	0	10

Datuaren helbideari dagokionez, itzulpen-denbora 20 ziklokoa da, une horretan atzitzen delako lehenengo aldiz 8 orri logikoa; horregatik, TLBak bilaketa egin duenean, ez du topatu, eta bilaketaren emaitza hutsegitea izan da; ondorioz, orri-taulan bilatu du. Beste aldetik, memoria nagusia atzitu denean, atzipen-denbora 10 ziklokoa izan da, moduluaren barruko 0 helbidea lehenengo aldiz atzitzen delako.

5. aginduari, `load r6,A[r2+16]`, 336 helbide logikoa dagokio, baina datuari dagokionez, `A[4]` osagaia atzitu behar du programak begiztaren lehenengo pasaldian; bektoreen osagaiak hitz batekoak direnez, bakoitzak 4 byte betetzen ditu; hori dela eta, `A[4]` osagaia `A[0]` baino 16 posizio aurrerago dago, hau da: 2048 + 16 = 2064 da horren helbide logikoa.

Eta horrela jarraitu dezakegu agindu guztiak exekutatu arte.

Baina, eskuarki, ez dugu taula beteko orain arte egin dugun bezala, horizontalki, lerroz lerro, prozesua errealitatean burutzen den bezala. Aitzitik, bertikalki betetzea, zutabez zutabe, erosoagoa da.

Laburbilduz, lehenik programa hori exekutatzean prozesadoreak zein helbide logiko sortuko dituen bilatuko dugu, agindu-mota eta helbideratze-modua kontuan hartuz. Hau da, taulako ezkerreko zutabeak beteko ditugu lehenik, prozesadoreak begiztaren lehenengo pasaldian sortuko dituen helbide guztiekin, aginduenak zein datuenak.

	Agindua / datua	Helbide logikoa
	`movi r1,#1520`	320
	`movi r2,#0`	324
	`movi r3,#251`	328
`begizta:`	`load r5,A[r2]`	332 / 2048
	`load r6,A[r2+16]`	336 / 2064
	`mul r5,r5,r6`	340
	`load r10,[r1]`	344 / 1520
	`add r6,r6,r10`	348
	`store r6,B[r2]`	352 / 512
	`addi r2,r2,#4`	356
	`subi r3,r3,#1`	360
	`bge r3,begizta`	364

Eta hortik abiatuta taula osoa beteko dugu:

@ℓ	*o.ℓ.*	*d*	$t_{itzulp.}$	*o.f.*	*d*	*@f*	hitza	@barne_mod	mod	$t_{atzip.}$
320	1	64	20	2	64	576	144	36	0	10
324	1	68	1	2	68	580	145	36	1	1
328	1	72	1	2	72	584	146	36	2	1
332	1	76	1	2	76	588	147	36	3	1
2048	8	0	20	0	0	0	0	0	0	10
336	1	80	1	2	80	592	148	37	0	10
2064	8	16	1	0	16	16	4	1	0	10
340	1	84	1	2	84	596	149	37	1	$10^{(1)}$
344	1	88	1	2	88	600	150	37	2	1
1520	5	240	20	5	240	1520	380	95	0	10
348	1	92	1	2	92	604	151	37	3	$10^{(2)}$
352	1	96	1	2	96	608	152	38	0	10
512	2	0	20	1	20	276	69	17	1	10
356	1	100	1	2	100	612	153	38	1	$10^{(3)}$
360	1	104	1	2	104	616	154	38	2	1
364	1	108	1	2	108	620	155	38	3	1

Azalpenak:

(1) Atzipen-denbora 10 ziklokoa da tartekatze-bufferretan aldez aurretik zeuden hitzak, 37 barne-helbideari dagozkionak, desagertu direlako 1 barne-helbidea atzitu denean. Horregatik, berriro 37 barne-helbidea behar denean, berriz irakurri behar da memorian, tartekatze-bufferretan 1 helbideari dagozkion hitzak sartu direlako.

(2) Aurreko kasuan bezala, baina 95 barne-helbidea da tartean sartu dena.

(3) Aurreko bi kasuetan bezala, baina oraingo honetan 17 barne-helbideko hitzak dira tartekatze-bufferretan sartu direnak 38 barne-helbideko bi irakurketen artean.

Laburbilduz, programak begiztaren lehenengo pasaldian sortzen dituen helbideak itzultzeko 92 ziklo behar dira (4 hutsegite × 20 ziklo + 12 asmatze × 1 ziklo), eta behar den informazioa memoria nagusian eskuratzeko, 106 ziklo behar dira (10 irakurketa memorian × 10 ziklo + 6 irakurketa tartekatze-bufferretan × 1 ziklo).

(c) Aurreko atalean lehenengo pasaldian behar den denbora kalkulatu dugu. Oraingo honetan, berriz, programa osoa exekutatzean prozesadoreak sortzen dituen helbide logiko guztiak itzultzeko eta memoria nagusia atzitzeko behar den denbora kalkulatu behar dugu.

Taula betetzea luzeegia litzateke. Horregatik, pentsatzeari ekingo diogu, eta ondorio orokorrak ateraz gero, denbora kalkulatu ahal izango dugu.

Helbide logikoak itzultzeko behar den denborari buruz nahikoa dugu aurreko atalean idatzi dugun ekuazioa orokortzea:

92 ziklo = 4 hutsegite × 20 ziklo + 12 asmatze × 1 ziklo

Hau da, jakin behar dugu guztira zenbat hutsegite gertatzen diren TLBan, eta zenbat asmatze. Esan dugun legez, hutsegiteak gertatzen dira orri logiko bat lehenengo aldiz atzitzen denean (suposatuko dugu orri logikoak memoria nagusian kargatu ondoren bertan jarraituko dutela programaren exekuzioa amaitu arte, hau da, memoria nagusian nahikoa tokia dagoela programa honetako orri logiko guztiak sartzeko, eta sistemak ez dituela beste batzuengatik ordezkatuko toki faltagatik).

Hala, nahikoa da jakitea zenbat orri logiko betetzen dituen programak (aginduek eta datuek). Azter ditzagun banan-banan.

Taulan ikusi dugun legez, agindu guztiak 1 orri logikoan daude. Hortaz, hutsegite bakarra sortuko da aginduen helbideen ondorioz.

A bektoreari dagokionez, taulan ikusi dugu lehenengo osagaiak (A[0] eta A[4] ageri dira taula horretan) 8 orri logikoan daudela. Begiztaren adierazpena aztertzen badugu, ikusiko dugu azken pasaldian (`i` = 251 denean) A[251] eta A[255] osagaiak atzituko direla. Beraz, A bektoreak 256 osagai ditu, A[0] eta A[255] artekoak. Osagai bakoitzak 4 byte okupatzen dituenez, A bektoreak guztira 1024 byte okupatuko ditu. Orri bakoitzean 256 byte sartzen direnez, ondorioztatzen da A bektoreak 4 orri beteko dituela. Edozein kasutan, egiaztatu behar da hori zehatza den ala ez, gerta daitekeelako 5 orri okupatzea, baldin eta A[0] osagaia ez badago orri baten hasieran. Ez da hau kasua, jadanik ikusi dugulako A[0] osagaia 8 orri logikoan dagoela, 0 desplazamenduarekin. Baina beti merezi du egiaztatzea. Horretarako, azken osagaiaren helbide logikoa kalkulatuko dugu, eta gero hortik ondorioztatu zein orri logikotan dagoen. Azken osagaiaren helbide logikoa kalkulatzeko kontuan hartuko dugu zein den bektorearen hasierako helbidea, zenbat osagai dituen, eta zenbat posizio okupatzen dituen osagai bakoitzak. Honelaxe:

$$@A[255] = 2048 + 255 \times 4 = 3068$$

Eta helbide horri dagozkion orri logikoa eta desplazamendua:

@logikoa = 3068 → orri logikoa = 3068 `div` 2^8 = 3068 `div` 256 = 11

desplazamendua = 3068 `mod` 2^8 = 3068 `mod` 256 = 252

Hau da, A bektoreak okupatzen dituen orri logikoak hauek dira: 8, 9, 10, 11.

B bektoreari dagokionez, B[0] osagaia, taulan ikusi dugunez, 2 orri logikoan dago. Begiztaren adierazpena aztertuz, ikusiko dugu azken pasaldian B[251] osagaia atzituko dela. Beraz, azken osagaiaren helbide logikoa kalkulatu dezakegu:

$$@B[251] = 512 + 251 \times 4 = 1516$$

Eta helbide horri dagozkion orri logikoa eta desplazamendua:

@logikoa = 1516 → orri logikoa = 1516 `div` 256 = 5

desplazamendua = 1516 `mod` 256 = 236

Hau da, B bektoreak okupatzen dituen orri logikoak hauek dira: 2, 3, 4, 5.

Azkenik, C bektoreari dagokionez, C[0] osagaia da atzitzen den bakarra, 1520 helbide logikoan dago, eta, taulan ikusi dugunez, 5 orri logikoa dagokio, hau da, 5 orri logikoa B eta C bektoreen artean konpartituta dago.

Taulan ikusi dugun bezala, hauek dira helbide horri dagozkion orri logikoa eta desplazamendua:

@logikoa = 1520 → orri logikoa = 1520 `div` 256 = 5

desplazamendua = 1520 `mod` 256 = 240

Laburbilduz, aginduen eta bektoreen artean, hauek dira atzitzen diren orri logikoak: 1, 2, 3, 4, 5, 8, 9, 10, 11. Hau da, 7 orri logiko, 7 hutsegite.

Orain asmatze-kopurua zein den jakiteko, atzipen-kopuru osoa kalkulatu behar dugu. Horretarako, programa aztertuko dugu. Begiztatik kanpo, hiru agindu daude, eta hauek behin bakarrik atzitzen dira, programaren exekuzioa hasten denean, hain zuzen. Begiztaren barruan, berriz, 9 agindu daude eta bektoreen 4 osagai atzitzen dira pasaldi bakoitzean Hau da, 13 atzipen pasaldi bakoitzean; begizta 252 aldiz errepikatzen denez, atzipen-kopurua, guztira, hau da:

$$\text{atzipen-kopurua} = 3 + (9 + 4) \times 252 = 1020$$

Atzipen horietatik, 7 hutsegiteak dira. Ondorioz (1020 – 7) = 1013 asmatze izango dira. Hala, itzulpen-denbora osoa:

$$t_{itzulp.} = 7 \text{ hutsegite} \times 20 \text{ ziklo} + 1013 \text{ asmatze} \times 1 \text{ ziklo} = 1153 \text{ ziklo}$$

Atzipen-denborari dagokionez, berriz, nahikoa dugu taula begiratzea, bertan atzipenen portaera islatzen baita. Taulan ikusi dugu datuen helbideak aginduen helbideen artean sartzen direla, eta, horren ondorioz, nahiz eta agindu batzuk barne-helbide bertsuetan egon, tartekatze-bufferretatik desagertzen direla datuen eraginez, eta hurrengo atzipenean berriro kargatu behar direla. Portaera hori

begiztaren pasaldi guztietan gertatuko da, atzipenak modu berean egiten direlako. Baina salbuespen nabaria dago, eta hori begiztaren lehenengo aginduari dagokio. Izan ere, begiztaren lehenengo pasaldian, taulan islatzen den moduan, 1 ziklo behar da agindu hori atzitzeko, 36 barne-helbidean dagoelako, eta lehenengo agindua irakurri denean, tartekatze-buferretan kargatu dira 144, 145, 146 eta 147 hitzak, 36 barne-helbidean, 0, 1, 2 eta 3 moduluetan. Baina bigarren pasalditik aurrerako guztietan gauzak aldatzen dira, agindu horren aurretik azkena exekutatu delako, 155 hitza hain zuzen ere, eta hori 38 barne-helbidean dago. Hortaz, 147 hitza behar denean (begiztaren lehenengo agindua, alegia), 10 ziklo beharko dira, beste barne-helbidean dagoelako. Hala, atzipen-denbora kalkulatzeko begiztaren lehenengo pasaldia bereizi behar dugu hurrengo pasaldi guztietatik.

Begiztatik kanpo: $10 + 1 + 1 = 12$ ziklo

Begiztaren lehenengo pasaldian: $9 \times 10 + 4 \times 1 = 94$ ziklo

Begiztaren hurrengo pasaldietan: $10 \times 10 + 3 \times 1 = 103$ ziklo

Begizta, guztira, 252 aldiz errepikatzen denez, lehenengo pasaldia kenduta 251 pasaldi gehiago geratzen dira. Hala, guztira:

$$t_{atzip.} = 12 + 94 + 103 \times 251 = 25959 \text{ ziklo}$$

behar dira memoriako atzipen guztiak egiteko.

Hortaz, guztira, $(1153 + 25959) = 27112$ ziklo behar ditu programak memoriako erreferentzia guztiak itzultzeko eta atzitzeko.