CACHE MEMORIEN OINARRIAK ORDENADORETAN

Olatz Arregi & Clemente Rodriguez

ABSTRACT

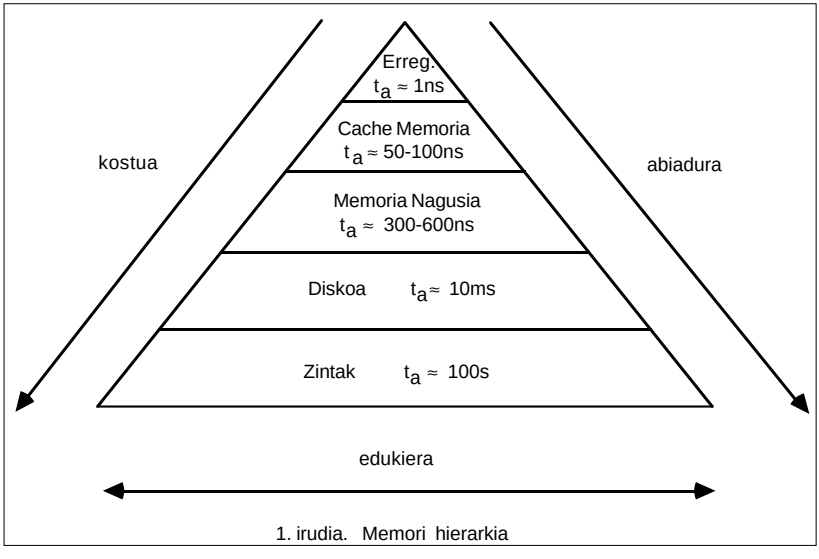
Cache memories are small, high–speed buffer memories, inserted betweed the processors and main memory, used in modern computer systems to hold temporarily those portions of the contents of main memory which are (beliefed to be) currently in use.

The success of cache memories can be attribuited to its property of locality of references. This property has two aspects, temporal an spatial.

The operation of a cache memory is very simple. When a read request is received from the CPU, the contes of a block of memory words containing the location spedified are transferred into the cache. When any of the locations in this block is referenced by the program, its contens are read directly from the cache. The correspondence between the main memory blocks and those in the cache is spedified by means of a mapping function. When the cache is full and a memory vord is referenced that is not in the cache, a decision must bi made as to which block should be removed to create space for the new block that contains the referenced word (replacement algorithm).

SARRERA

Von Neumann motako ordenadoreen eraginkortasuna, denbora–unitate batean memoriatik/ra bidal daitekeen informazio–kopuruak (banda–zabalerak) mugatzen du. 32 biteko mikroprozesadorea duten sistemetan, arazo hau latzagoa gerta daiteke abiaduraren desberdintasuna dela eta, mikroprozesadoreen abiadurak memorien erantzun–ahalmena gainditzen bait du. Irtenbide klasikoa, memori hierarkia bat ezartzean datza. Hierarkiaren goi–maila, memoria azkar eta garestiei dagokie (beraz, txikiak izango dira), behekoa berriz, memoria motel eta merkeei dagokielarik (edukiera handikoak izan daitezke) (ikus 1. irudia). Hierarkia honen barnean, cache memoria (CM)



1. irudia. Memori hierarkia

goi–mailakoen artean aurkitzen da eta prozesatzaile eta memoria nagusiaren (MN) artean kokatzen da.

Cache memoriak prozesatzaileak gehien erabiliko dituen memoria nagusiko zatiak gordeko dituzten memoria azkarrak dira (RAM memoriak). Honela, informazioa denbora–tarte motz batean atzi daiteke eta sistemaren eraginkortasuna hobetu egiten da.

Cache memorien arrakasta programa gehienek betetzen duten berezkotasun batean datza. Esan dugunez, CMetan, etengabe erabiltzen ari diren MNko zatiak gordetzen dira. Zati hauek zeintzuk diren sumatzeko, aurrean aipatutako propietatea hartuko da kontutan. Propietate honi lokaltasun edo ingurutasun deritzo eta bi aldaera bereizten zaizkio:

1.– Denborazko ingurutasuna

Askotan, hurrengo unean erabili behar den informazioa, une honetan erabiltzen ari den berbera da. Horren arrazoia sinplea da: kontuan izan behar da, programetan bigiztak oso erabiliak direla, eta beraz, bai datuak bai aginduak askotan berrerreferentziatu egiten direla.

2.– Lekuzko ingurutasuna

Askotan, une honetan erabiltzen ari denaren inguruko informazioa da hurrengo unean erabili behar dena. Portaera hau oso arrunta da bai programa askotan aginduak sekuentzialki egikaritzen direlako eta baita, datuak (aldagaiak, arrayak) gehienetan batabestearen ondoren gordetzen direlako ere.

Erreferentziatzen diren informazio–azpimultzo hauek gehienetan txikiak izaten dira eta horri esker, nahiz eta CMk oso handiak izan ez, hauek gordetzen duten informazioa gehienetan erabilgarria da.

CM eta MNen arteko desberdintasun nabarienak tamaina eta abiadura dira. Adibide gisa ordenadore batzuen atzipen–denborak aipatuko ditugu. Abiadura handiko ordenadore batean (Amdahl 470V/7, IBM 3081, 3090) MNko atzipen–denbora 300–600 ns bitartekoa da, CMko atzipen–denbora 50–100 ns–koa izanik. Ordenadore hauen egikaritzapen–denbora, jadanik CMren erantzun–denborak mugatzen du. Beraz, CM bat ezartzea beharrezkoa da gaur egun. CM erabiltzea hain onuragarria denez, irtenbidea tamaina handiko cache bat jartzea dela pentsa daiteke, eraginkortasuna hobetzeko alegia, baina kostuak irtenbide hau ezinezko bihurtzen du; ordenadore txikietan batez ere.

CMren kontzeptu teorikoa 1965. urtean azaldu zuen Wilkes jaunak, baina 1968. urtera arte ez zen CM zuen ordenadorerik eraiki, hau IBM 360/85a izan zelarik. CMk sortu zirenetik, hauei dagokien atzipen–denbora asko jaitsi da, gaur egun 30–100 ns–koa delarik.

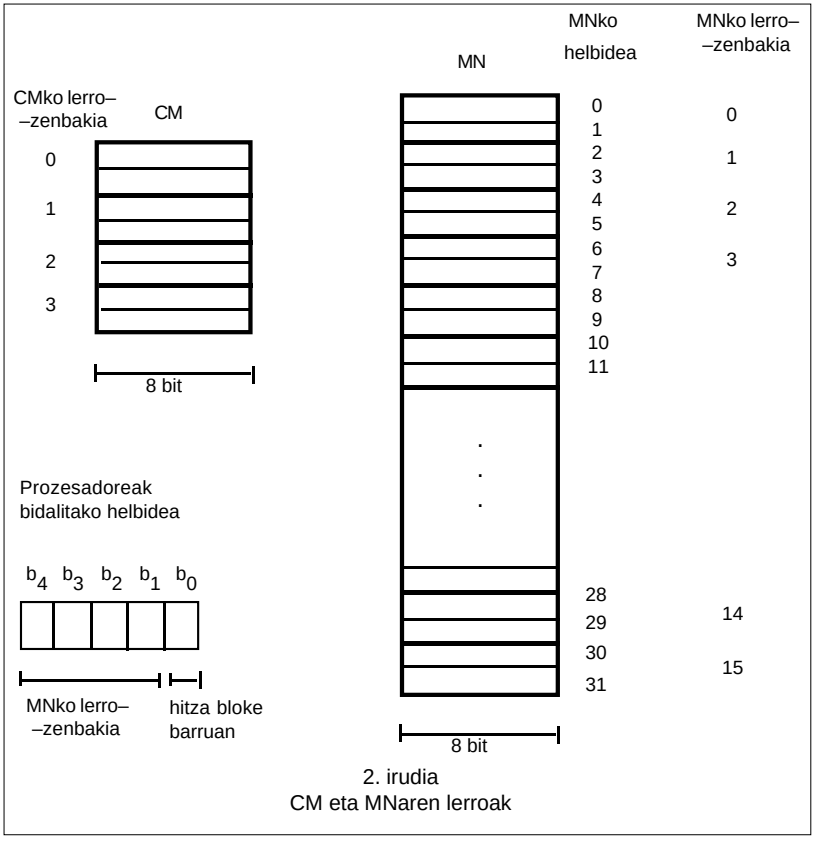
Segmentazio–teknikak dituzten prozesatzaileak erabiltzeak, CMen beharra adierazten du. Teknika hauek, agindu–fluxua, agindu bat ziklo bakoitzeko izatea dute helburutzat (RISC, Reduced Instruction Set Computer, motako ordenadoreen helburua hain zuzen ere). Hori lortzeko, memorien erantzun–denbora eurrez murriztu behar da. Bestalde prozesatzailearen abiadura handitzeak, CM gabeko sistema baliaezina izatea dakar berarekin (kontuan izan Galio Artseniuroko teknologietan oinarritutako sistemetan, ate logiko baten konmutazio–denbora pikosegundotakoa dela).

Gaur egungo CMen tamaina oso aldakorra da. Sistema handi batek, NEC ACOS 9000–k adibidez, 128kbyteko CM izan dezake. Bestalde, 32 biteko mikroprozesadore batzuek, 256 byte besterik ez dute (Texas Instruments–eko TMS34010–k adibidez). Tartean, 64kb (IBM 3033, AMDHAL 470/V8), 16kb (IBM 370/168 3, AMDHAL 470/V7), 16kb (AMDHAL 470/V6, VAX 8600) eta 512 bytekoak (Motorola 68030 mikroprozesadorea) aurki daitezke.

CACHE MEMORIEN FUNTZIONAMENDUA

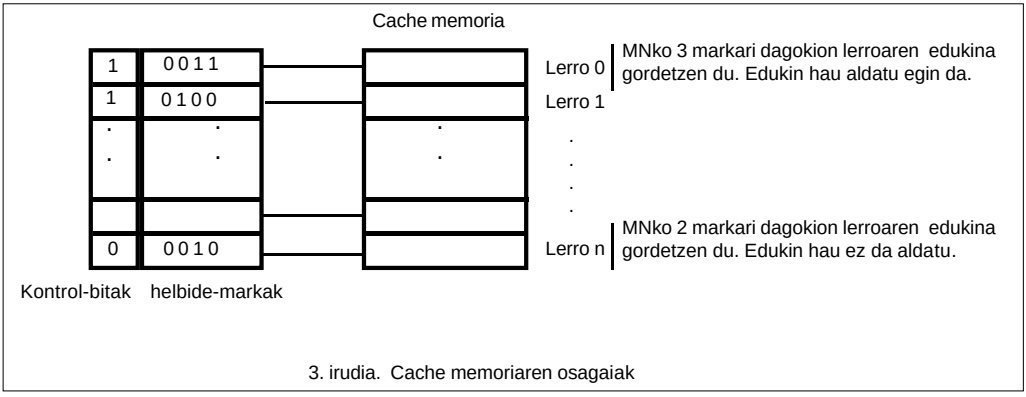
Esan dugunez, CMetan, etengabe erabiltzen ari diren MNko zatiak gordetzen dira. Bi memori mota hauen artean informazio–trukea beharrezkoa da eta transferentzi unitateari lerro deitzen zaio. Lerroa alboko hitz–multzo batez osaturik dago. Bai CM eta bai MN lerro–multzo batez eratzen dira, CMko lerroak beti MNkoen kopiak direlarik. 2. irudian, 4 lerroko CM bat eta 16 lerroko MN baten eskema daukagu. Lerroa bi hitzekoa baldin bada, MNan 32 hitz izango ditugu; beraz, prozesatzaileak bidaliko dituen helbideak 5 bitekoak ($b_4b_3b_2b_1b_0$) izango dira ($2^5 = 32$). Bost biteko helbide hau bi eremutan bana daiteke, non batek, $b_4b_3b_2b_1$ bitez osaturik dagoenak hain zuzen, lerro–zenbakia adieraziko duen ($2^4 = 16$) eta besteak, b_0 bitez osaturik dagoenak, lerro barruan zein hitz helbideratzea nahi den, ($2^1 = 2$).

CM duen sistema baten funtzionamendua ondorengoa da: Prozesatzailearen Unitate Zentralak (CPU) MNko informazioren bat behar duenean, informazioa, gordetzen duen gelaska edo posizioari dagokion helbidea bidaltzen du. Helbide hau egokituz, CMra jotzen da eta bilaketan bertan hasten da. Behar den informazioa CMn baldin badago, atzipena burutu egiten da denbora asko aurreratuz. Kasu honetan asmatze (aurkitze) bat egon dela esaten da eta MNa ez da ezertarako erabiltzen. Erreferentzia CMn aurkitzen ez bada, hutsegitea gertatu da, MNa atzitu beharko da eta MNan bakarrik bilatu behar izan bagenu baino denbora gehiago erabiltzen da. Beraz CM ondo diseinatu behar da gehienetan asmatzea suerta dadin. Hau ez da oso zaila ingurutasun propietateari esker.



2. irudia
CM eta MNaren lerroak

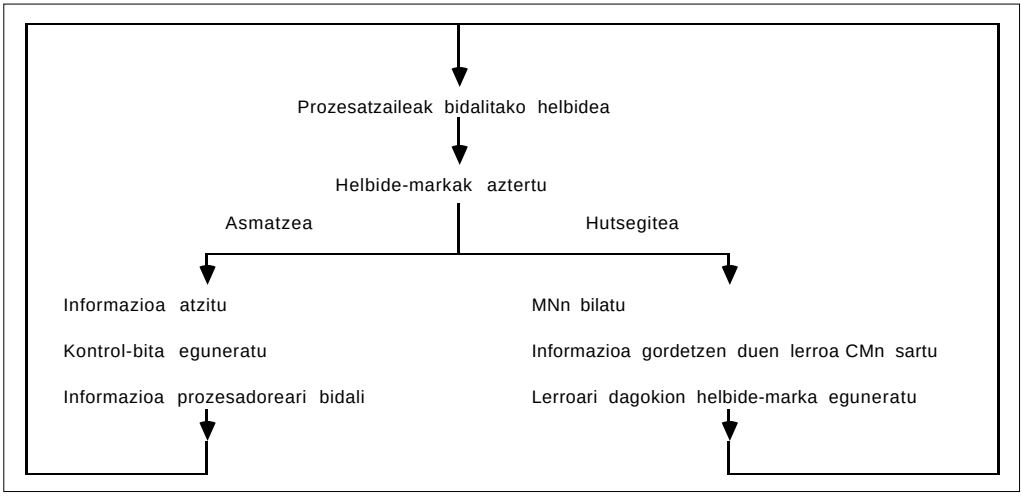
CM batean, lerro bakoitzari kontrolerako bit bat edo batzuk eta helbide–marka bat egokitzen zaizkio. Helbide–marka hauek, CMn MNko zein



3. irudia. Cache memoriaren osagaiak

lerro dauden adierazten dute. Bestalde, kontrol–bitak, lerroan dagoen informazioa aldatu den ala ez adieraziko du. (ikus 3. irudia).

CMen funtzionamenduaren algoritmo orokorra honako hau da:



CACHE MEMORIEN DISEINUA

Sistema batean CM ezartzen denean, sistemaren eraginkortasuna hobetzea nahi da. Hau lortzeko ondorengo helburuak bete behar dira: **a)** erreferentzia bat egitean, eskuratzea nahi den informazioa CMn aurkitzeko probabilitatea handitzea, hau da, asmatze–tasa (aurkitze–tasa) (h) hobetzea, **b)** CMko atzipen–denbora (t_a) murriztea eta **c)** erreferentziatutakoa CMn ez badago, informazioa atzitzeko behar den denbora murriztea. Helburu hauek betetzeko CM diseinatzerakoan erabiliko diren parametroak ondo aukeratu behar dira. Hona hemen CMn diseinurako kontuan izan behar diren parametro batzuk:

1.– Bilaketa–algoritmoa

Bilaketa–algoritmoak informazioa CMra noiz eraman erabakitzen du, hau da, MNko lerro bat CMn noiz kopiatu behar den.

Sistema gehienetan lerroa beharrezkoa suertatzen denean kopiatzen da, hau da, prozesatzaileak erreferentzia bat egin ondoren CMn aurkitzen ez bada, MNra jotzen da eta lerroa atzitu ondoren CMra eramaten da. Metodo honi "bilaketa eskatzerakoan" edo "eskatutakoan bilatu" deritzo. Bilaketa hauek zeharo baztertzea ezinezkoa da, baina murriz daitezke eskaera egin aurretik epe labur batean erabiliko diren lerroak CMra eramaten badira. Lerro hauek aukeratzeko ingurutasun propietatea izango da kontuan. Gaur egun erabiltzen den era bat, lerro bat erreferentziatzean, lerro hau eta hurrengoa CMra eramatea da. Metodo honen antzekoa izango litzateke hurrengo lerroa, erreferentziatzen dena CMn ez dagoenean bakarrik aurreratzea, eta honela lerro bakarra mugitu beharrean, bi mugitzen dira. Hurrengo lerroa hautatzen da bai aginduak bai datuak askotan sekuentzialki erabili behar izaten direla jakin badakigulako.

Lerro bat erreferentziatutakoan huts egitea suertatzen bada, bi eratara konpon daiteke: bata, behar den lerroa CMra pasatu eta bilaketa berriz hastea, eta bestea, eraginkorragoa, behar diren hitzak MNtik prozesatzailera zuzenean bidaltzea, une berean CMn lerroa kopiatuz. Horrela ez da berriz eragiketa burutzen hasi behar. Metodo hau 470/7, 470/8 eta IBM 3033an erabiltzen da.

2.– Kokapen–algoritmoa

Kokapen–algoritmoak MNko lerroak CMko zein lerro beteko duen erabakitzen du, hau da, informazioa CMko zein lerrotan gordetzen den. Erabaki hau hartzeko diseinatzaileak aukeratu duen egokitasun–mota kontuan izan behar da.

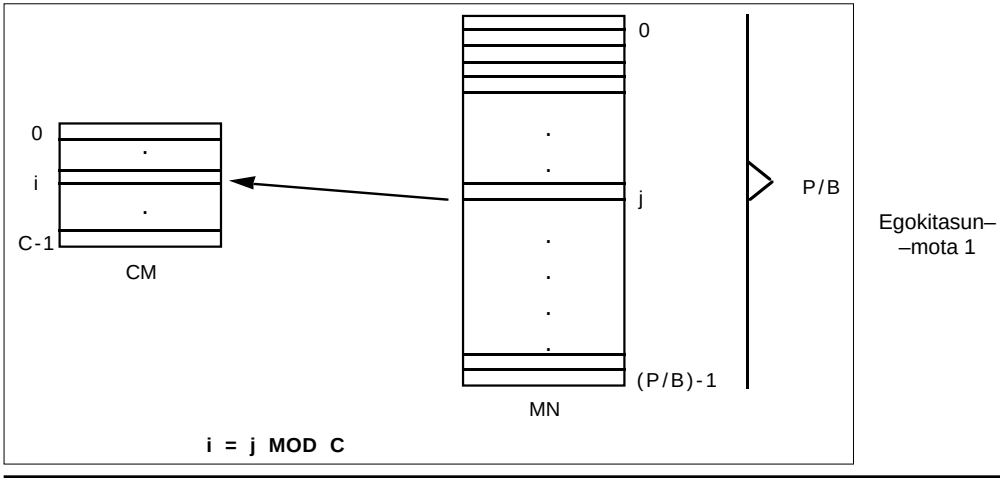
Egokitasun–motak CMko lerro bakoitzak MNko zenbat eta zein lerro gorde dezakeen adierazten du. Hau dela eta, huts egite bat gertatzen denean, CMko lerro batzuk bakarrik aztertu beharko dira; egokitasun–motak adierazten dituenak hain zuzen.

Ondoren, erabiltzen diren egokitasun–mota desberdinak azalduko dira:

a) Egokitasun zuzena

MNko lerro bakoitza CMko lerro bakar batean gorde daiteke nahiz eta honek, MNko lerro bat baino gehiago gordetzeko balio izan. Beraz, CMn bilatzean, helbide–marka bakar bati begiratu beharko zaio. Lerroak aldez aurretik egokituta daude.

Ondorengo formulari jarraituz, MNko lerro bati CMko zein lerro dagokion ezagutu daiteke:



L(CM) = **L**(MN) mod **C**, *non **L** lerro–zenbakia eta **C** CMko lerro–kopurua bait dira.* Beraz, CMko lerro bakoitzean, MNko P/(B*C) lerro gorde daitezke, non P prozesatzailearen helbideratze–lekunea (hitz–kopurua) eta B lerro bakoitzeko hitz–kopurua bait dira.

Egokitasun–mota honen sinpletasunak, eraiketa–kostuaren murriztea dakar. Gainera, bilatzen ari garen lerroa CMn dagoenean, informazioa irakurtzeko behar den atzipen–denbora murriztu egiten da. Bestalde, eragozpen bezala, asmatze–tasaren beherapena daukagu.

b) Zeharo elkargarria den egokitasuna

MNko lerro bakoitza, CMko edozein lerrotan gorde daiteke, beraz CMko lerro bakoitzean MNko P/B lerro gorde daitezke.

Prozesatzaileak bidalitako erreferentzia bat CMn dagoen ala ez jakiteko CMko helbide–marka guztiak aztertu beharko lirateke. Atzipen–denbora murrizteko, helbide–marken azterketa aldi berean egiten da.

Metodo honen abantaila nagusiena, asmatze–tasa handitzea da. Eragozpen bezala aldiz, atzipen–denbora luzatzea aipa daiteke.

d) Multzoka elkargarria den egokitasuna

Zeharo elkargarri den egokitasunaren eta zuzenaren arteko irtenbide bat da. CM tamaina bereko M multzotan banatzen da eta MNko lerro bakoitzari multzo jakin bat egokituko zaio, nahiz eta multzo barruan edozein lerrotan kokatzeko aukera izan. Multzo bakoitzean C/M lerro izango dira.

Ondorengo formulari jarraituz, MNko lerro bati CMko zein multzo dagokion jakin daiteke:

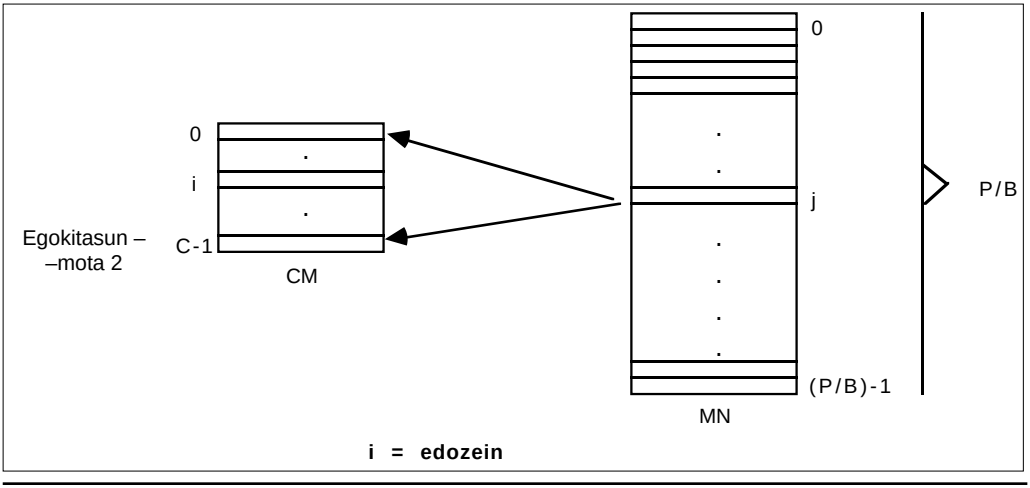
Multzoa(CM) = **L**(MN) mod **M,** *non **M** multzo–kopurua bait da.*

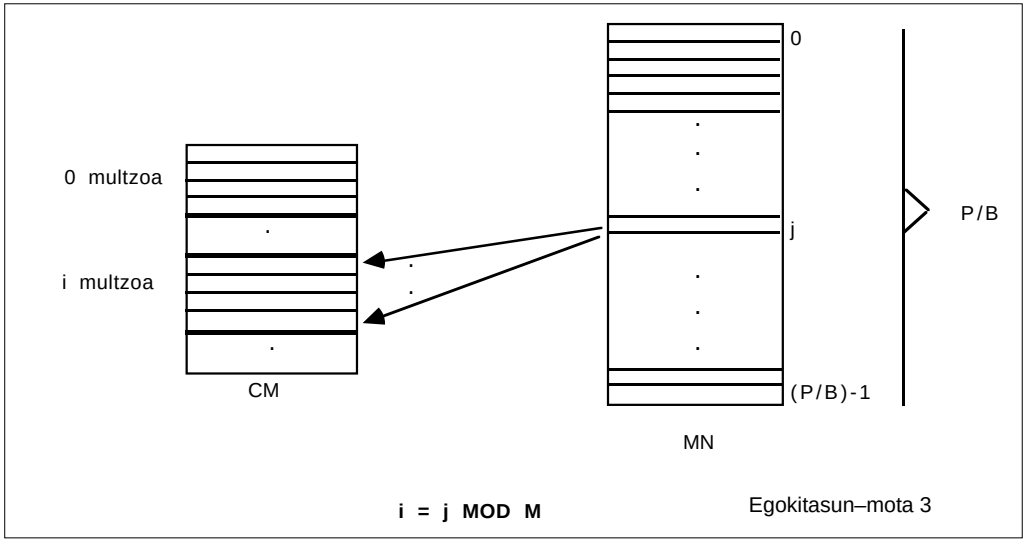
Metodo honen abantaila atzipen–denbora azkarra lortzea da.

3.– Ordezkapen–algoritmoa

Prozesatzaileak MNko lerro bat behar duenean eta hau CMn ez dagoenean, egokitasun–mota kontutan izanda, MNko lerroa CMn gorde behar da. CMn lekurik ez balego, hemengo lerro bat hautatu beharko litzateke ordezka dadin.

Kasu batzuetan, hautaketa aldez aurretik emana dago. Egokitasun zuzena daukagunean adibidez, lerro bakoitzaren posizioa zehaztuta dago. Beraz, hemen ez dago ezer aukeratzerik. Beste bi kasutan berriz, ordezkaketa





egiteko metodoren bati jarraitu beharko gatzaizkio.

Gaur egun erabiltzen diren algoritmoak ondorengoak dira:

RAND (**RAND**om): ordezkatu behar den lerroaren hautaketa ausazkoa da. Metodo hau inplementatzea ez da batere zaila, baina ez da irtenbide aproposena, programaren portaera ezertarako kontutan hartzen ez delako.

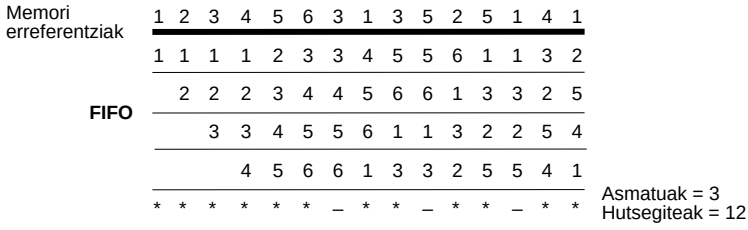
FIFO (**F**irst **I**n **F**irst **O**ut): lehenengo ordezkatuko dena, CMn lehenengo sartu dena izango da. Metodo honek, CM barruan denbora gehien daraman lerroa aterako du. Hau ere askotan ez da oso egokia, maiz erabiltzen ari den lerro bati barruan denbora asko eramateagatik ateratzea tokatuko zaiolako.

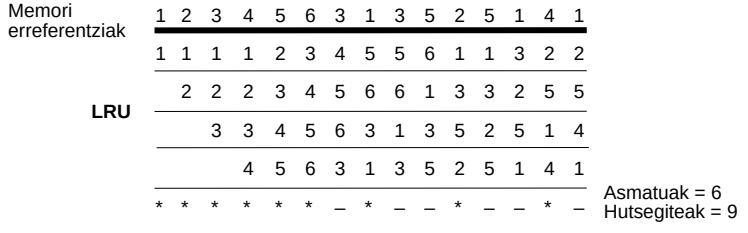
LRU (**L**east **R**ecently **U**sed): ordezkatuko den lerroa, CMn erabili gabe denbora gehien daramana izango da. Metodo hau da egokiena programaren portaera kontuan izaten duelako. Dena dela algoritmo honen inplementazioa konplexuena da, eta sistema txikietan normalean ez du merezi ezartzerik. Sistema handietarako oso egokia da.

4. irudian, zeharo elkargarri den egokitasuna duen sistema batean, FIFO eta LRU metodoen eragina ikusiko dugu. Horretarako, CMk 4 lerroko edukiera duela suposatuko dugu eta memoriak erreferentziatzen duen lerro–sekuentzia ondorengoa dela: 1, 2, 3, 4, 5, 6, 3, 1,3,5,2,5,1,4,1.

4.– Idazketa

Prozesatzaileak memoriako hitz baten edukina aldatu nahi duenean, hau da, idazketa bat egin nahi duenean, erreferentzia bidaltzen du eragiketa burutu dadin. Aldatu nahi den hitza CMn baldin badago, eta CMko lerroak MNkoen kopiak direnez, aldaketa MNan ere azaldu beharko litzateke. MNaren eguneratzea





4. irudia. FIFO eta LRU ordezkapen–metodoak.

bi eratara egin daiteke: **a)** CMn idazten den bakoitzean, MNn ere idatzi (write through) eta **b)** MN, CMn aldatu den lerroa ordezkatu behar denean eguneratzen da (copy–back).

Sistema batean prozesatzaile bakarra baldin badago, copy–back metodoa eraginkorragoa da memori trafikoa murrizten duelako. Kontuan izan, MNko lerro bat CMn dagoen bitartean askotan aldatzen bada, behin bakarrik eguneratuko dugula MNn; ordezkatzen denean hain zuzen. Write through metodoaz, aldatzen den bakoitzean eguneratu beharko genuke.

Copy–back metodoa erabiliz, koherentzi arazoak sortzen dira CMn eta MNan informazio desberdina gordetzen delako. Bestalde, metodo hau erabiltzen denean, bi irtenbide daude: **a)** CMko lerro bat ordezkatzen den bakoitzean, MN eguneratzea eta **b)** lerro bakoitzari bit bat egokitu, lerroa aldatu den ala ez adierazteko. Horrela ordezkapena egiterakoan, MN CMko lerroa aldatua izan bada bakarrik eguneratuko da.

ONDORIOAK

Artikulu honetan, CMen diseinurako kontuan izan behar diren aukera garrantzitsuenak azaldu ditugu. Dena den, badira ahaztu ezineko baina artikulu honetarako gai ez diren beste ezaugarri batzuk. Gaia sakon aztertuko bagenu, lerroen tamaina, cachean gordetzen den informazio–mota, koherentzi arazoa, alegiazko helbideen itzulpena eta abar kontuan izan beharko genituzke.

BIBLIOGRAFIA

A.J.Smith; “Cache Memories”, Computing Surveys, Bol. 14. zk. 3, iraila 1982.

V. Viñals, C. Rodriguez eta A. Olivé; “Memorias Cache”, Mundo Electronico, 167. zk. 1986.

O. Arregi eta C. Rodriguez; “Sistemas con memoria cache”, Automática e instrumentación, 198. zk. urtarrila 1990.

ERANSKINA

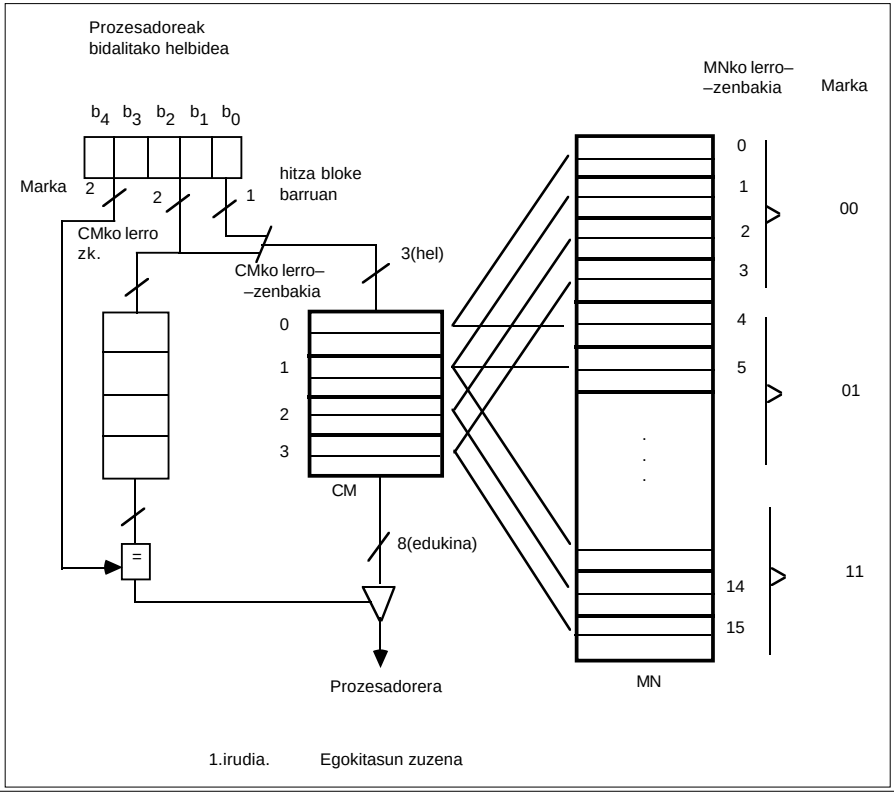
Eranskin honetan, CM duen sistema batean informazioa bilatzeko jarraitu beharreko urratsak azalduko ditugu. Egokitasun–mota bakoitzak bilaketa era batera edo bestera egitera behartuko gaitu. Beraz, kasu bakoitzari dagokion eskema adieraziko dugu. Hobeto ulertzeko, adibideetan memoria–sistema konkretu bat erabiliko dugu, honi dagozkion ezaugarriak ondorengoak direlarik:

hitza : 8 bit
lerroa : 2 hitz
MNren tamaina : 32 hitz (16 lerro)
CMren tamaina : 8 hitz (4 lerro)

MNk 32 (25) hitz dituenez, prozesatzaileak 5 biteko helbideak bidaliko ditu. Bestalde, CMk 8 (23) hitz izanik, 3 biteko helbideak beharko ditu atzipena burutzeko. Beraz, helbideen itzulpena egin beharko da.

a) Egokitasun zuzena.

Prozesatzaileak bidaltzen duen helbidea ($b_4\ b_3\ b_2\ b_1\ b_0$), hiru eremutan banatzen da. Eskuin muturreko bitak (b_0), lerroaren barruan zein hitz bilatu nahi den adieraziko du. Tarteko bi bitek ($b_2\ b_1$), informazioa CMko zein lerrotan egon daitekeen azalduko dute, eta azkenik ezker muturreko



1.irudia. Egokitasun zuzena

bitek (b_4 b_3) bilatzen ari garen informazioa CMn dagoen ala ez esango digute, horretarako azken bi bit hauek eta helbide–markak konparatu behar direlarik. Helbide–markek, CMra eramaten diren hitzen helbideen pisu handieneko ezker muturreko bi bitak gordetzen dituzte.

Egokitasun–mota hau erabiltzen denean, helbide–marken direktorioa eta edukinen memoria (RAM) aldi berean edo paraleloz atzi daitezke. Honi esker, atzipen--denbora murriztea lor daiteke (ikus 1. irudia).

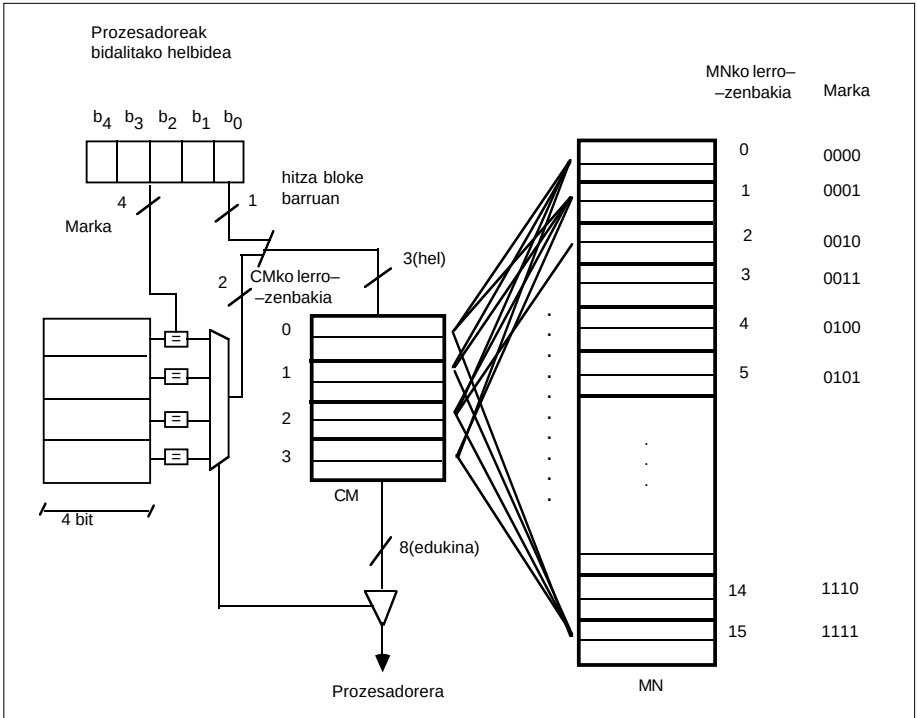
b) Zeharo elkargarria den egokitasuna.

Prozesatzaileak bidalitako helbidea, bi eremutan banatzen da. Alde batetik pisu gutxieneko bita (b_0) izango dugu, zeinak lerroaren barruan zein hitz erreferentziatzen den adieraziko duen, eta bestalde, ezker muturreko lau bitak (b_4 b_3 b_2 b_1), helbide--marka erakutsiko dutenak. Lehenengo urratsa, lau bit hauek CMko marka guztiekin konparatzea izango da. Bilatzen ari dena aurkitzen badu, kodetzailetik ateratzen diren bi bitek informazioa CMko zein lerrotan dagoen adieraziko dute. Ondoren, CM atzi daiteke eta eskatzen zen hitza eskuratu (ikus 2. irudia).

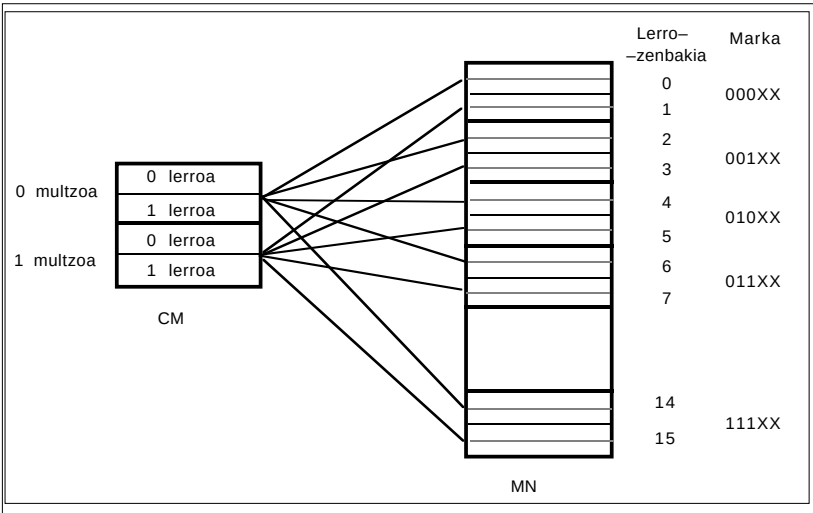
d) Multzoka elkargarria den egokitasuna.

Kasu honetarako, CM bi multzotan zatitzen dela suposatuko dugu.

Prozesatzaileak bidalitako helbidea, hiru

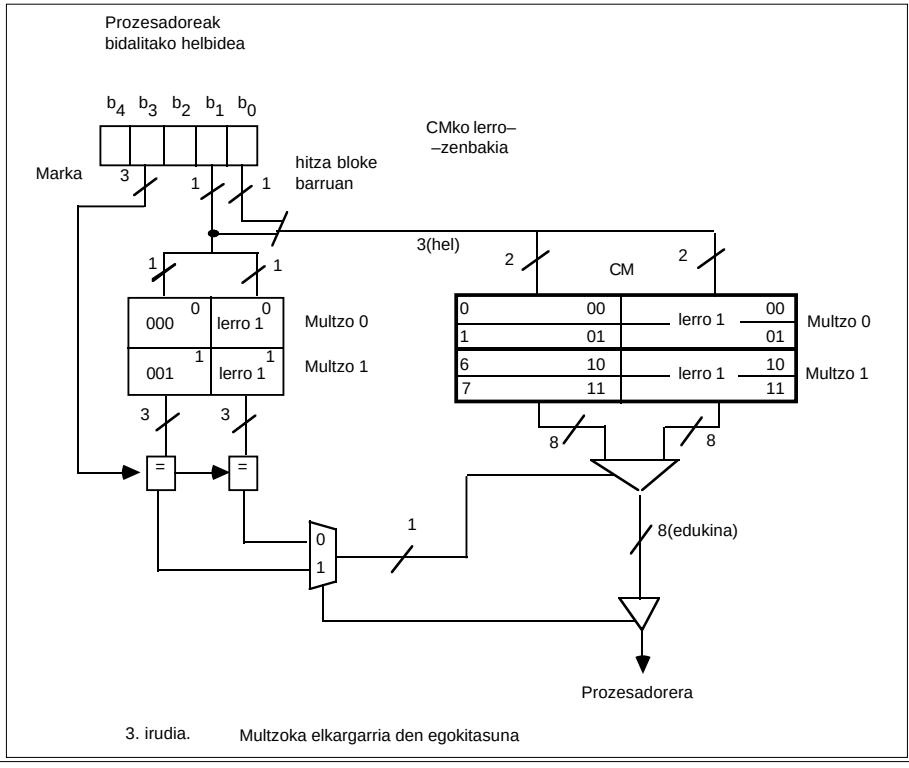


2. irudia. Zeharo elkargarria den egokitasuna



ezkerrean dagoena (b_1) multzoaren ezaugarria izango da eta ezker muturreko beste hiruek (b_4 b_3 b_2) helbide–marka adieraziko dute. Kasu honetan, marken konparaketa eta edukinen bilaketa aldi berean egiten da. Dena den, azkenean, multiplexoreak kodetzailearen emaitza jaso behar du edukina kanporatzeko (ikus 3. irudia).

eremutan banatzen da. Aurreneko kasutan bezala, pisu gutxieneko bitak (b_0) lerro barruan hitza zein den adieraziko du. Honen



3. irudia. Multzoka elkargarria den egokitasuna