eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

BILBOKO INGENIARIEN GOI ESKOLA TEKNIKOA

KONPUTAGAILUEN PROGRAMAZIOA
TURBO PASCAL BITARTEZ

I

EGILEA: Jesus-Mari Romo Uriarte

(hirugarren zirriborroa) 2000-9-20

HITZAURREA

Esku artean duzun liburu honek konputagailuen programazioaren oinarriak aurkezten ditu, helburu nagusi horrekin batera informatikaren hastapeneko kontzeptuak ere lantzen ditu.

Liburu osoan zehar ariketa praktikoei garrantzia handia ematen zaie, batez ere laugarren kapitulutik aurrera, hots, programazioaren gaia hasten denetik aurrera. Esan daiteke kontzeptu bakoitzeko programa bat idatzi dela, programen erabilpena errazago izan dadin kapituluka bildu dira eta kapitulu bakoitzaren amaieran euren identifikazio-taula tartekatu egin da. Programok diskete batean banatzen ditugu horrela eskatuz gero, iturburu-programak direnez exekutatu ahal izateko konpiladoreren bat beharko du ikasleak (guk Borland etxeko Turbo Pascal 7.0 konpiladorea erabili dugu).

Liburuak 14 kapitulu ditu, baina bigarren zirriborro hau amaitzen dugun une honetan lehen hamabiak idatzirik ditugu, gainerakoak hurrengo baterako utzi ditugularik.

14 kapituluen kontzeptuak jasotzeko ikasleak 12 kreditu beharko lituzke, euretatik 4 kreditu gutxienez konputagailuaren aurrean era praktiko batean kurtsatuko lituzke. 12 kreditu horiek bi ikasturtetan banaturik egonez gero, hona hemen proposatzen dugun jokabidea:

Ikasturtea	***Kredituak***	***Kapituluak***
1	6	1, 2, 3, 4, 5, 6, 8, 9, 10, 11
2	6	7, 12, 13, 14

Bilboko Ingeniarien Goi Eskola Teknikoan, Industri Ingeniarien titulazioan erabiltzen da liburu hau eta bertan 9 kreditu ditugu une honetan konputagailuei buruzko kontzeptu guztiak eman ahal izateko. Gauzak horrela, egun prestaturik ditugun lehen hamabi kapituluak jorratzeko denborarik ez dugulako zazpigarrena, unitateak lantzen dituena alegia, sakontasunik gabe gainbegiratzen dugu.

JM Romo Uriarte

NON ZER

1. ATALA: INFORMATIKARAKO SARRERA

AURKIBIDEA

1. ATALA: INFORMATIKARAKO SARRERA *1*

1.1 SARRERA

Informazioa transmititu eta prozesatu beharra betidanik izan du gizakiak, horregatik *informatika* bezala egun ezagutzen dugun zientziak aspaldiko aurretikoak ditu. Lehenengo kapitulu honetan zenbait definizio emango dugu, hurrengo kapituluen oinarriak direlako argi geratu beharko dira, edozein kasutan liburu honen planteamendu didaktikoa praktikoa izanik erabat argi geratuko ez diren kontzeptuak aurrerago ikusiko diren kapituluetan landuko direnez ez da zertan orain dena %100an "ulertu" behar.

Has gaitezen bada informatika hitzak zer esanahi duen, UZEI hiztegitik hartuta honela definitzen da informatika: *ordenadoreen diseinu eta erabilerari eta informazioaren tratamendu automatikoari dagozkion alderdi guztiak biltzen dituen jakintza-arloa*.

Informatika zientziaren definizioan elementu pare bat azpimarratuko genituzke. Batetik, Informatikak darabilen gaia informazioa dela (bigarren kapituluan zehaztuko dugu informazioa zer den), eta informazioaren prozesaketa edo tratamendua du helburu. Bestetik informazioaren tratamendua automatikoa dela, hots, informazioaren tratamendua makinaz egingo dela ordenadoreez[1] alegia.

Zientzia bezala Informatikaren[2] ikergaiak, besteak beste, hauek lirateke:

- Algoritmoak eta datu-egiturak.
- Ordenadoreen arkitektura
- Adimen artifiziala eta Robotika
- Datu-baseak
- Gizaki-makina arteko komunikazioa
- Kalkulua
- Sistema Eragileak
- Programazio-lengoaiak
- Software ingeniaritza

Nabaria denez sarrerako kurtso honetan ezin dira aipatu diren arlo guztiak arakatu, gure helburua ordenadore batek nola egiten duen lan ikastea da, hori bai, oso modu praktikoan irakatsiko dugu ordenadoreen funtzionamendua. Ondorioz ikasleak goi-mailako lengoaia batean programatzen ikasiko du, proposaturiko arazoa era formal batean formulatzen ohituko da eta haren ebazpidea programatzen gai izango da.

1.2 ALGORITMOA

Konputazio Zientziaren (Informatikaren) oinarrizko kontzeptua *algoritmoa* da. Hura formalki definitu aurretik intuitiboki zer den azalduko dugu, edo behintzat saiatuko gara esplikatzen.

Ordenadoreen programazioa zeregin berria da, baina "programazioa" funtsean arazoen soluzioa bilatzeko metodoa da eta esan genezake arazoak bezain zaharrak direla haien

[1] Guri "ordenadore" terminoa baino "konputagailu" gehiago gustatzen zaigu. Logikoagoa iruditzen zaigu bi arrazoiagatik angloxaxoniarren artean ordenadore hitza ezagutu ere ez delako egiten, eta, makina algoritmikoa den ordenadorea definitzean "konputagailu" hobeto egokitzen da eta ez "ordenadore" (ordenadore batek lan egiten duenean beti konputatuko du, baina ez du halabeharrez ordenatu behar)

[2] Informatika berba Frantzian sortua da 1962an bi hitz elkartuz **Infor**mazio Auto**matikoa**, guretzat horren ezaguna den hitza erabat arrotza da gainerako munduan (Amerika eta Europa barne).

ebazpideak. Planteaturiko arazo bat soluzionatzeko lau urrats betetzen dira, ordena-doreen programazioan beteko ditugunak:

1. *Arazoaren ulermena*. Problema zehaztasunez deskribatu beharra dago, hau da, zer egin behar den prezisio osoz jakitea halabeharrezkoa da. Urrats hau ez da beti behar bezala betetzen eta oso ondorio txarrak ekartzen ditu.
2. *Arazoaren ebazteko plana egin*. Zer egin behar den ziurtasunez dakigunean bestelako galderak erantzun genitzake, arazoari nola ekingo diogun bidea erakusten digutenak: Zer baliabide daukagu? Zer nolako tresneria? Nolakoak dira langileak? Zer epetan amaitu behar da? Erosgairik behar da? Errekurtso edo baliabide guztiak nola konbinatuko dira?.
3. *Plana gauzatu*. Zer egin behar den dakigula nola egingo den hausnartu ondoren lanean hasiko gara, helburu zehatz eta hura lortzeko plangintza egokiarekin.
4. *Soluzioa ebaluatu*. Gure produktuak hasierako espezifikazioak betetzen ditu?.

Soluzioa bilatzeko lau urrats horietan ez da esan arazoa Informatikoa izan behar denik, Informatikaren mundura jauzi eginez problema bati erantzuten dion programa lortzeko lau urratsak hauek dira:

1. *Arazoaren definizioa*
2. *Algoritmoa asmatu*
3. *Algoritmoa programa bezala idatzi*
4. *Soluzioa ebaluatu*

Beraz programa bat idatzi aurretik algoritmo bat sortu beharra daukagu. Baina algoritmoa zer da?. Intuitibori[3] erraz definitzen da: lan bat exekutatzeko behar diren instrukzioen multzoa da algoritmoa.

Algoritmoen adibiderik klasikoenak errezetak dira, jarraian Pedro Subijana maisu famatuaren “Denok Sukaldari” liburutik hartutako algoritmoa ematen da:

PORRUSALDA-KREMA BAKAILAOAREKIN

Lau lagunentzat gaiak

porruak: 375 gr
tipula: 180 gr
patata: 470 gr
bakailaoa: 200 gr
bi baratxuri-atal
olioa

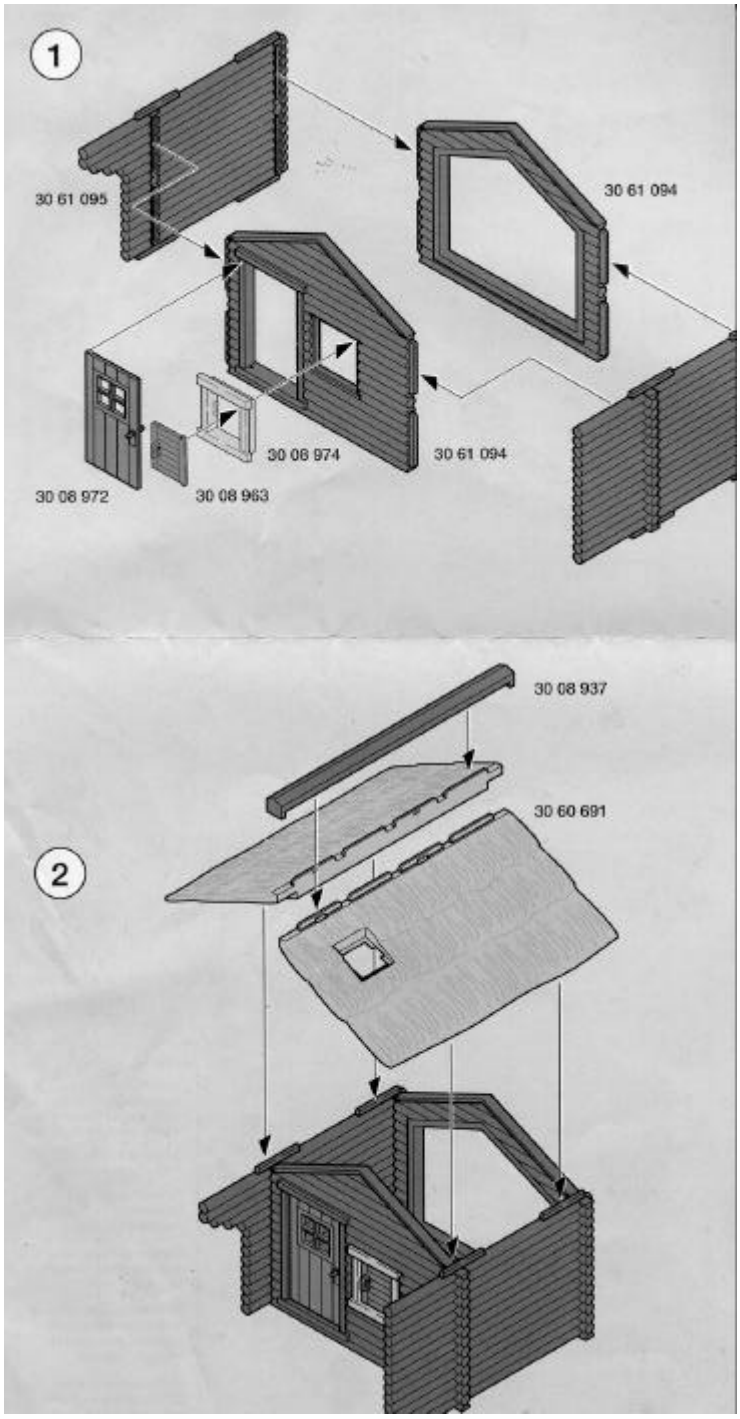
Nola egin

1. Porruak txikitu eta zatirik samurrenak azkenerako utzi hornigaitarako erabiltzeko
2. Gainerakoak kazola batean jarri egosten
3. Gehitu tipula txikitua eta patata zatituak, olio zorrotada bat bota eta su motelean utzi irakiten, 40 minutu edo.

[3] Algoritmo instrukzioen multzo finitua da, instrukzioak exekutagarriak dira eta euren arteko anbiguotasunik (zalantzagarritasunik) ez dago, eta, instrukzioek gidatzen duten zereginak amaiera du.

4. Bitartean, gezatutako bakailaoa prestatu eta mamitu. Zartaginean pasa baratxuri txikitu eta olio pixkarekin, eta lurrintzen utzi botatako ur guztia.
5. Porrusalda egin eta gero, xehatu eta iragazi (fin-fin gera dadila), gehitu bakailao-mamiak eta nahastu.
6. Azkenerako utzitako porru-zatiak erdi egosi zartaginean eta krema gainean bota.

Algoritmoak ideiak transmititzeko tresnak direnez grafikoak izan daitezke, jostailuak muntatzeko orrietan agertzen diren bezalakoak:



1.2.1 Algoritmoen erabilpena

Algoritmoak betidanik erabili dira, badira algoritmoak janariak prestatzeko errezetak deitzen ditugunak, musika jotzeko partiturak bezala ezagutzen ditugunak, etxeko landareak zaintzeko prozedurak algoritmoak izan daitezke, magia egiteko liburuak algoritmoen liburuak dira (amarrua urratsez urrats enumeratzen baita), mendi liburuetako ibilbideak algoritmikoki deskribatu ohi dira.

Antzinako Mesopotamian algoritmoak erabiltzen ziren, Grezia zaharrean ere Euklides matematikariak gure egunetara iritsi den algoritmoa asmatu zuen. Euklides-en algoritmoak bi zenbakien z.k.h. (zatitzaile komunetako handiena) aurkitzeko balio du eta honela dio:

Arazo orokorra:

Bi zenbakien zatitzaile komunetako handiena lortu.

Sarrerako datuak:

Osoak eta positiboak diren bi kopuru.

Prozesua:

Algoritmoak hiru urrats ditu lehenengoa hasieraketa da eta beste biak behin baino gehiagotan errepika daitezke.

1. Datuetan kopuru handienari M deituko diogu, kopuru txikiena N izango da.
2. M eta N arteko zatiketa osoa lortu hondarrari H deituz.
3. H zero ez denean, N balioa M-ri erantsi (esleitu)
   H balioa N-ri erantsi
   bigarren urratsera itzuli;
   bestela, z.k.h. N-ren uneko balioa da

Adibidea:

Datuak 36 eta 16

M	N	H	Zatidura
36	16	4	2
16	4	0	4

z.k.h. **4**

Behin algoritmoa asmatu eta sortu ondoren bi zenbaki osoen z.k.h. lortzea hura aplikatzea aski da, ez da ulertu behar zer egiten dugun urratsak egoki betetzea besterik ez zaigu gelditzen. Esaterako 7 eta 11 arteko zatitzaile komunetako handiena kalkulatu nahi badugu, taula hau beteko litzateke larregirik pentsatu gabe:

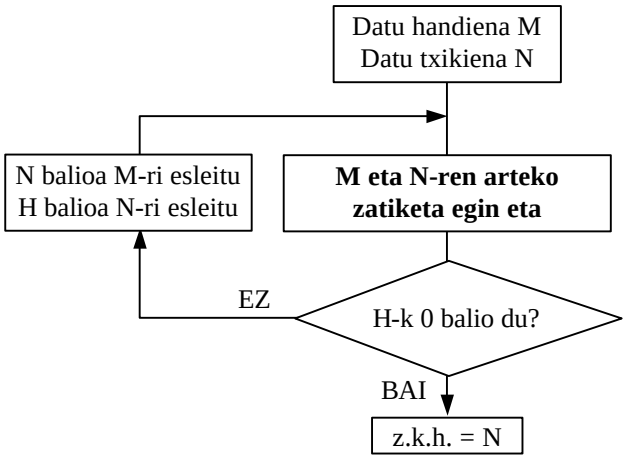
Datuak 11 eta 7

M	N	H	Zatidura
11	7	4	1
7	4	3	1
4	3	1	1
1	1	0	1

z.k.h. **1**

Nolabait esateko z.k.h. kalkulatzeko behar den adimena algoritmoan txertatuta dago eta horren erabiltzailea ez da oso burutsua izan behar, areago, makina automata bat izan daiteke taula eraikiz soluzioa eskura dezakeena. Alderantziz formulatuta, arazo bat emanik hura ebazteko algoritmorik aurkitzen ez badugu, ezinezkoa izango zaio ordenadore bati arazo horren soluzioa bilatzea.

Algoritmoak hobeto ulertzeko urratsen zerrenda baten ordez fluxu-eskemen bitartez adieraz daitezke, lantzen ari garen adibiderako eskema hauxe litzateke:



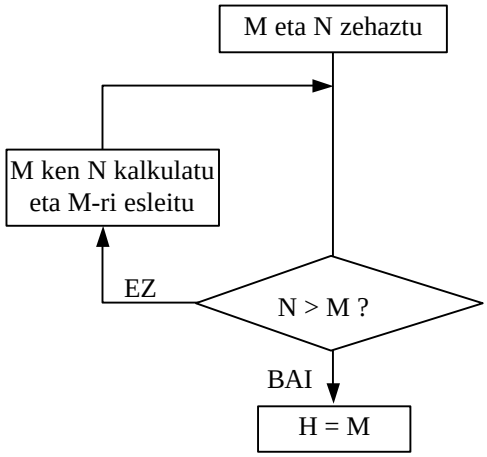
1.2.2 Algoritmoen mailaketa

Euklides-en algoritmoarekin jarraituz, eta azken puntuan emandako fluxu-eskemari begiratuz, zatiketak nola egiten diren ezagutzen dela suposatu dugu. Edo porrusalda-kremaren prestaketan deskribatzen den bosgarren urratsan:

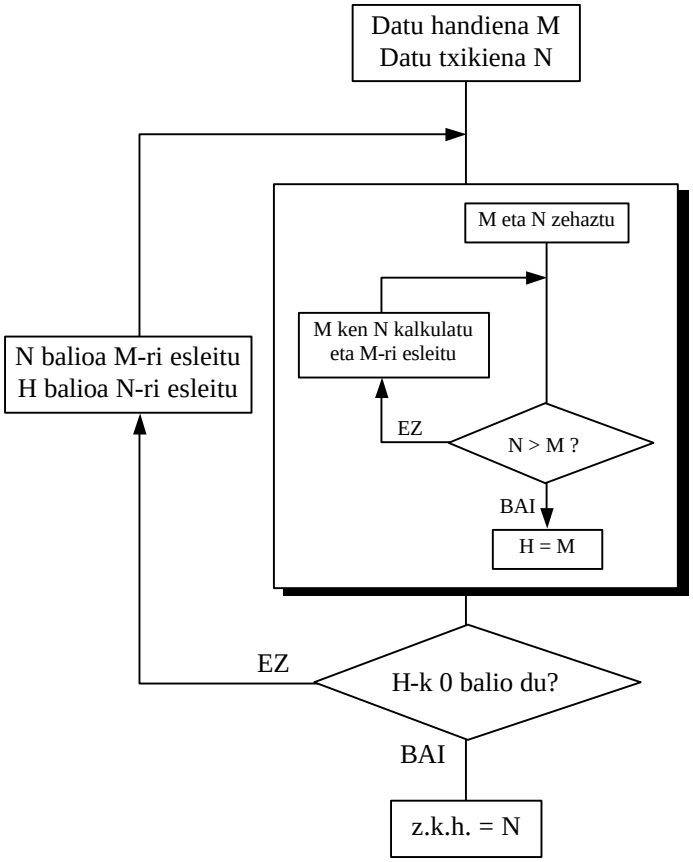
> 5. Porrusalda egin eta gero, xehatu eta iragazi (fin-fin gera dadila), gehitu bakailao-mamiak eta nahastu.

Ez denez ezer gehiagorik zehaztu, porrusalda fin-fin gera dadin irakurleak nola egin behar duen badakiela suposatu da errezeta idatzi denean.

Honek esan nahi du algoritmo bat formulatzean abstrakzio maila bat aukeratu behar dela, eta horren arabera algoritmoak zatika adieraz daitezkeela. Esate baterako Euklides-en algoritmoan zatiketak nola egiten diren zehaztu nahi izanez gero, txikiak ginenean erakutsi ziguten zatiketa operazioari dagokion algoritmoa bilatu beharko genuke (honek behartzen gaitu biderketa definitzera). Edo bestela kontutan izan dezakegu zatiketa operazioa kenketa multzo bat dela, horrela jokatuz hona hemen zatiketa definitzen duen algoritmoari dagokion eskema:



Kenketa nola kalkulatzen den suposatuz, Euklides-en algoritmoari dagokion eskema osatua hau litzateke:



Algoritmoa formulatuta dagoen bezala ordenadoreak ezingo du ulertu, ordenadoreak interpreta ditzakeen aginduak oso elementalak baitira. Algoritmoa osatzen duten instrukzioetatik makinak onartzen dituen aginduetara jauzi egin beharra dago. Azken honi programa esaten zaio eta jarraian garatzen den puntuan horren ideia hartuko dugu.

1.2.3 Algoritmotik programara

Demagun ordenadorearen bitartez joko bat antolatu nahi dugula.

Programaren zeregina honela deskribatzen da: erabiltzaileak, guk, 0 eta 100 artean dagoen zenbaki oso bat pentsatuko dugu eta ordenadoreak zein den asmatu beharko du. Konputagailuak saiakera ezberdinak izango ditu, eta bakoitzean 0 eta 100 arteko zenbaki bat aukeratuko du eta guk pentsatutako kopurutzat harturik soluzio bezala aurkeztuko digu erabiltzailearekiko honelako elkarrizketa izanik:

```
1. saiakeran ordenadoreak aukeratutako zenbakia: 50
zurearekin konparatuta, nolakoa da?
            B Berdina
            H Handiagoa
            T Txikiagoa
Erantzuna: T
```

1. saiakeran erabiltzaileak, teklatuz, dagokion erantzuna emango dio ordenadoreari, horretarako erabiltzaileak pentsatutako zenbakia eta ordenadoreak aukeratu duena kontutan edukiko ditu. Eta, horren arabera hauek izan daitezke konputagailuaren ekintzak:

- Erantzuna `B` izatean, zenbakia asmatu du eta programa amaituko da
- Erantzuna `H` edo `T` izatean beste soluzio berri bat proposatuko du eta aurreko menuaren bitartez erakutsiko digu

Esaterako, lehenengo saiakeran 50 kopurua gure zenbakia baino txikiagoa delako `T` erantzun da, baina bigarren txandan ordenadoreak hau erakutsiko digunean gure erantzuna `H` izango da guk asmatutako zenbakia 50 eta 75 artean baitago:

```
2. saiakeran ordenadoreak aukeratutako zenbakia: 75
zurearekin konparatuta, nolakoa da?
            B Berdina
            H Handiagoa
            T Txikiagoa
Erantzuna: H
```

Azken informazio honekin ordenadoreak soluziotzat `62` hartuko luke. Hau erakutsi eta programa bukatu `B` erantzun baitzaio:

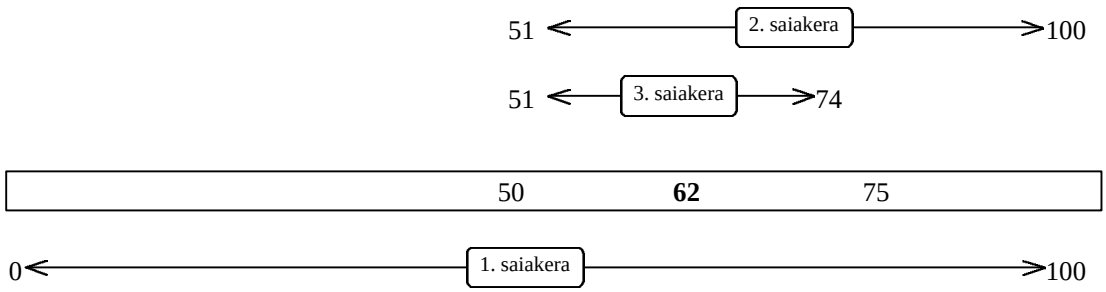
```
3. saiakeran ordenadoreak aukeratutako zenbakia: 62
zurearekin konparatuta, nolakoa da?
            B Berdina
            H Handiagoa
            T Txikiagoa
Erantzuna: B
Pentsatu duzun zenbakia 62 da
```

Falta zaigu azaltzea ordenadoreak zergatik aukeratu dituen `50` , `75` eta `62` zenbakiak, nola “dakien” guk pentsatutako zenbakia mugatzen? ordenadoreak zein estrategia darabilen?. Konputagailuak, hasieran baliagarria den 0 eta 100 arteko esparrua gure erantzunen arabera aldatu egiten du, eta saiakera bakoitzean ordenadoreak proposatzen digun soluzioa uneko esparruaren erdian aurkitzen den zenbakia da.

Algoritmoa laburbiltzen duen taula:

	Uneko esparrua		
Saiakera	**Behemuga**	**Goimuga**	**Erdia**
1.	0	100	(0+100) / 2 = 50
2.	51	100	(51+100) / 2 =75
3.	51	74	(51+74) / 2 =62

Algoritmoa grafikoki adierazita, non gezien muturrek saiakera bakoitzari dagozkion behemuga eta goimuga azaltzen duten:



Zenbakia asmatzen duen algoritmoak hiru zati nagusi ditu: jokoa hasi, saiakerak errepikatu eta jokoaren bukaera. Hona hemen algoritmoa:

Jokoa hasi:

Esparruaren behemuga 0 izan dadila

Esparruaren goimuga 100 izan dadila

Esparruaren erdia kalkulatu: (behemuga + goimuga) / 2

Saiakerak zenbatzen dituen kontagailua 1 izan dadila

Saiakerak errepikatu:

Uneko esparruaren erdia soluziotzat hartu eta pantailaratu

Erabiltzailearen erantzuna jaso

Proposatutako zenbakia guk pentsatutakoa baino handiagoa bada (erantzuna H) esparrua aldatuko da goimuga txikiagoa jarriz. Hurrengo saiakerarako behemuga dagoen bezala mantendu eta goimuga uneko esparruaren erdia ken 1 izan dadila

Proposatutako zenbakia guk pentsatutakoa baino txikiagoa bada (erantzuna T) esparrua aldatuko da behemuga handituz. Hurrengo saiakerarako goimuga ez da aldatzen baina behemuga uneko esparruaren erdia gehi 1 izan dadila

Behemuga edo goimuga berriekin, hasieraketan bezala, hurrengo saiakerari dagokion esparruaren erdia kalkulatu: (behemuga + goimuga) / 2

Saiakerak zenbatzen dituen kontagailua unitate batean inkrementatu

Saiakera berri bat behar den ala ez erabaki. Ordenadoreak proposatutako zenbakia eta guk pentsatutakoa berdinak izan badira (erantzuna B) algoritmoaren hirugarren zatira joan, bestela algoritmoaren bigarren zatiarekin hasieratik jarraitu. Errepikapenak eteteko bigarren arrazoi bat dago, hots, saiakerak 7 baino gehiago izan badira erabiltzaileak ez du erantzunetan zuzen jokatu eta algoritmoaren hirugarren zatira joan.

Jokoaren bukaera:

Algoritmoaren bigarren zatitik irtetea ziurtaturik dago (erantzuna B izan delako edo saiakera kopurua 8raino heldu delako) eta prozesu errepikakorra nola eten den aztertuz soluzioa erakutsiko zaio erabiltzaileari. Erantzuna B izan denean guk pentsatutako zenbakia azken esparruaren erdia da, bestela erabiltzaileak ez du zuzentasunez erantzun

Zenbakia asmatzen digun algoritmoa hizkuntza naturalean formulatu dugu, algoritmo horrek asmaketa lortzeko zein bide jorratu behar den argi eta garbi azaltzen digu, nahiz eta ondo ulertu ez helburua erdiesten dela froga dezake ikasleak. Beraz, zenbakia asmatzeko prozedura hau edozeinek aplika dezake era mekaniko batean, makina batek zenbakia asma dezan algoritmoa ezarri behar zaio. Tamalez hizkuntza naturalean formulaturiko algoritmo hori uler dezakeen ordenadorerik ez dago oraindik, hizkuntza naturala oso aberatsa da baina oso konplexua ere eta ideiak adierazteko anbiguatasun handiak egon daitezke gaizki-ulertze eta erroreak eragiten dituztenak.

Esandakoagatik ordenadoreei algoritmoak ematean ez da hizkuntza naturala erabiltzen *programazio-lengoaia* bat baizik. Programazio-lengoaia bat lengoaia naturalaren azpimultzo bat da (lengoaiaren hiztegia txikia da eta bere arauak oso zehatzak), ondorioz programazio-lengoaia batek edukiko dituen ezaugarriak argitasuna, sinpletasuna eta prezisioa izango dira.

Zenbakiaren asmaketa jokoa konputagailu batek exekuta dezan, algoritmoa hizkuntza naturaletik programazio-lengoaia batetara igaroko dugu. Lengoaia formal batean adierazitako algoritmoari **programa** deritzo. Adibidez Turbo Pascal izenez ezagutzen den lengoaiara igaro ondoren hona hemen zenbakia asmatzen digun programa:

```
PROGRAM ZenbakiaAsmatzen ;                      { \TP70\01\ASMATU.PAS }

                           { ------------- Zenbakiak memorian gordetzeko }
                           { ------------- aldagaiak definitzen dira     }
VAR
  Soluzioa, Saiakera, Behemuga, Goimuga : Byte ;
  Erantzuna: Char ;

BEGIN
                            { ------------- Algoritmoaren lehen zatia }
   Behemuga := 0 ;          { ------------- Jokoa hasi                }
   Goimuga := 100 ;

   Soluzioa := (Goimuga + Behemuga) DIV 2 ;
   Saiakera := 1 ;


                            { ------------- Algoritmoaren bigarren zatia }
   REPEAT                   { ------------- Saiakerak errepikatu         }
      WriteLn ;
      Write (Saiakera, ', saiakeran ordenadoreak aukeratutako zenbakia: ') ;
      WriteLn (Soluzioa) ;
      WriteLn ('zurearekin konparatuta, nolakoa da?') ;
      WriteLn ('            B  Berdina' ) ;
      WriteLn ('            H  Handiagoa' ) ;
      WriteLn ('            T  Txikiagoa' ) ;
      Write ('Erantzuna: ') ;

      Readln (Erantzuna) ;
      Erantzuna := UpCase (Erantzuna) ;

      IF Erantzuna='H' THEN
        Goimuga := Soluzioa - 1 ;

      IF Erantzuna='T' THEN
        Behemuga := Soluzioa + 1 ;

      Soluzioa := (Goimuga + Behemuga) DIV 2 ;
      Saiakera := Saiakera + 1 ;
   UNTIL (Erantzuna = 'B') OR (Saiakera > 7) ;


                            {-------------- Algoritmoaren hirugarren zatia }
                            {-------------- Emaitza erakutsiz jokoa bukatu }
   IF Erantzuna='B' THEN
     WriteLn ('Pentsatu duzun zenbakia ', Soluzioa, ' da')
   ELSE
     WriteLn ('Erantzunetan okertu zara ala gezurti hutsa zara') ;
END.
```

Irakurleak ez ditu `ZenbakiaAsmatzen` programaren sententziak ulertuko baina bai ikus dezake lengoaia naturalean idatziko algoritmoarekiko duen lotura estua, izan ere liburu eta kurtso honen asmoetatik bat algoritmoak Turbo Pascal lengoaia formalean idazten ikastea izango da. Onar dezagun ere algoritmoa programa bezala adierazita dagoenean ordenadore batek hura exekutatzeko aukera duela, eta algoritmoari guk barneratu diogun “adimena” bereganatzen duela.

Hurrengo puntuan makinei buruz arituko gara, algoritmoak exekutatzeko ahalmena duten makinak sailkatu eta aurkeztuko ditugu.

1.2.4 Makina algoritmikoak

Algoritmoa exekuta dezakeen gailuari makina algoritmiko esango diogu. Ordena-doreei zer egin behar duten programa baten bitartez zehazki esan behar zaienez algoritmoren bat erabiliko dute euren funtzionamenduan, horregatik ordenadoreak makina algoritmikoak dira.

Algoritmo kontzeptua Informatika baino zaharrago denez, makina algoritmikoak ere aspaldiko asmakizunak dira (makinaren bat algoritmikoa izan dadin bere funtzionamenduan algoritmoren bat jarraitzen duela kontsideratuz). Makina algoritmikoen bilakaera historikoa ***1.3 MAKINA ALGORITMIKOEN MUGARRI HISTORIKOAK*** puntuan garatuko dugu, baina horren aurretik makinen sailkapen bat egin dezagun.

1.2.4.1 Makina algoritmikoen sailkapena

Makina bat lan bat egin edo funtzioren bat betetzen duen aparatu edo aparatu-multzoa da, makinak bere kabuz funtziona dezake edo erabiltzaileak maneiatuta ibil daiteke. Makina nola kontrolatzen den aintzat harturik makinen sailkapen berezi hau egin daiteke:

1. Makina ez-automatikoak
2. Makina automatikoak
3. Makina programagarriak

Banan-banan azal ditzagun:

1. Makina ez-automatikoak Erabiltzaile batek uneoro maneiatu behar duen aparatua da. Lana burutu dadin erabiltzailearen partehartzea ezinbestekoa da, nahiz eta makina ez-automatikoak bere kabuz zenbait funtzio bete izan. Adibidez, idazmakinak letrak paperean inprimatzen ditu idazleak teklak sakatzen dituen heinean, baina mekano-grafiatzeari utziz gero idazmakina gelditzen da.

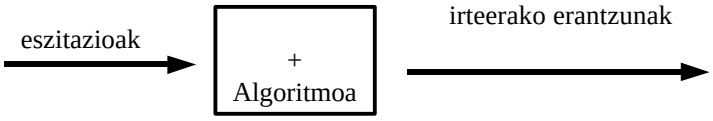
2. Makina automatikoak Makina automatiko batean operadorea ez da uneoro aparatua kontrolatzen ariko. Automatikoak kontsidera daitezkeen makinek duten funtzioa betetzeko, euren kabuz ibiltzen dira kanpotik aginduak hartu eta gero. Esate baterako modernoagoak diren idazmakinekin lan egitean, lerro muturrera iristean orga atzeratu eta papera igotzen du automatikoki.

Automatizazioa gradu bat da eta makina automatikoen adibideak amai ezinak lirateke. Denek onartzen dute erabiltzailearengandik nolabaiteko agindua eta funtzionamendua nahikoa autonomoa izaten da. Adibidez, gaur eguneko igogailuek teklatu txiki baten bitartez hartzen dituzte erabiltzailearen eskakizunak eta bere kabuz jaitsi eta igotzeko mugimenduak gobernatzen ditu (igo, jaitsi eta abar luzeko eginkizunak: abiadura eta azelerazioaren kontrola, ateen irekitzea eta ixtea, argitxo adierazgarriak piztea, bidaiarien eta kargaren pisua neurtzea, ...).

Makina automatiko bat eraikitzen denean etorkizunean izango duen funtzionamendua erabakitzen da, eta bere osagaietan era finko batean ezartzen da. Horrela, zirkuitu elektriko edo eta pneumatiko baten bitartez gobernatzen dira igogailuaren ateak,

eta sistema ongi dabilen bitartean ateak beti ireki eta itxi egingo dira modu berean. Baina ateak astiroago ixtea nahi izango bagenu ez litzateke erraza izango, igogailuaren barne egituretan jantziak izateaz gain fisikoki zaila delako (igogailua desmuntatu beharko litzatekeelako).

Igogailu batek, makina automatikoa den aldetik, algoritmo bat dauka bere zirkuituetan ezarririk, eta funtzionatzean eskema honi erantzuten dio beti:



Makina automatikoa

Makina automatikoaren algoritmoa aldatzea zaila da, ondorioz funtzionamendu autonomoa baina zurruna dauka.

3. Makina programagarriak

Makina programagarriak makina automatiko sofistikatuak dira. Izan ere, erabiltzailearen eskakizunak jasoko ditu baina makina automatikoak ez bezala ez dute portaera finko bat. Esan nahi da makinaren funtzionamendua unez une egokitzeko erraztasunak nabariak daudela makina programagarrietan.

Igogailuaren adibidera atxikiz, horrek duen funtzionamendua automatikoa baina zurruna da, botoi bat sakatuz ondoriotzen diren sekuentziak beti izango dira berdinak. Baina programa-garria den makina batean eszitazio bat jasoz izan daitekeen portaera desberdina litzateke, orain azalduko den arrazoiagatik.

Makina programagarria bi zatiz osaturik dago. Batetik *oinarrizko makina* deituko dugun zatia (aldaezina dela onar daiteke), eta, bestetik makinaren osagaia den *programa* (bigarren zati hau erraz aldatu eta moldatzeko diseinaturik dago):

Makina programagarriaren osagaiak

Makina progr

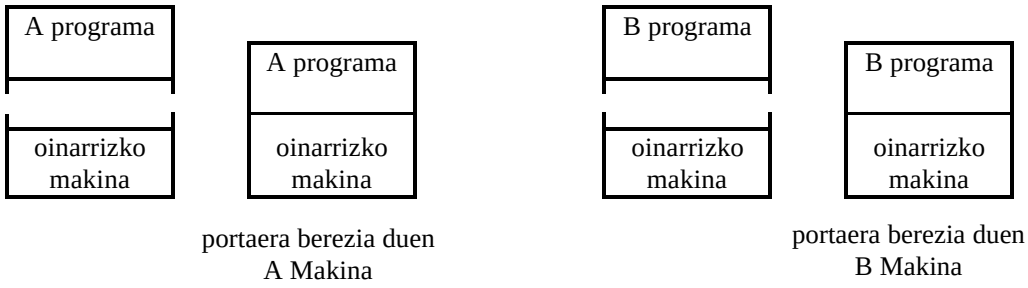
Makina programagarriaren funtzionamendua bi osagaien parte-bestera aldatzea erraza dela kontsideratuz makina programagarriaren funtzionamendua

Egindako sailkapenarekin gogoratuz:

1. Pianoa: -automatikoa, pianojoleak teklak sakatzen dituen bitartean musika entzungo da.
2. Musika kutxa:

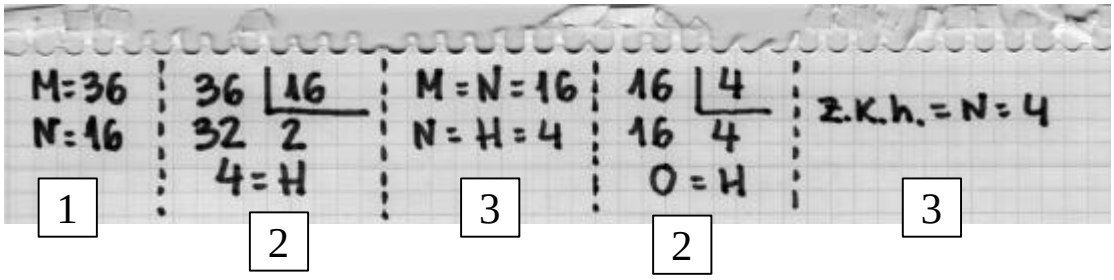
3. Disko-jogailua: Makina programagarria, diskoa kenduz portaera ezberdinak izan ditzakeelako (kontzertu barrokoa izatetik folk jaia izatera pasa daiteke, modu oso errazean). Soinua automatikoki jotzeaz gain apatuaren portaera unez une egokitzea berehalakoa da.

Ordenadore batek makina programagarria dela esango dugu, bi osagai baititu. Oinarrizko dispositiboekin batera (Informatikaren munduan *hardware* esaten zaio sailkapena egitean *oinarrizko makina* izendatu duguna), erraz karga daitekeen programa agertuko da ordenadorearen partaide bezala. Honez gero ordenadore batek duen funtzionamendua erabat alda daiteke exekutatzen duen programaren arabera: inprimagailuarekin batera idazmakina izan daiteke baldin eta testu prozesadore bat kargatzen badiogu, planoak marrazteko lanpostua izan daiteke CAD programa (marrazketa teknikorako programa) kargatzen badiogu, musika sor daiteke, etxeko kontabilitatea, irudi digitalak tratatu, sistemen ekuazioak ebatzi, zenbakizko datuak eman eta grafikoak marraztu, ...



1.2.4.2 Makina algoritmikoen arkitektura

Bi zenbakien zatitzaile komunetako handiena kalkulatzeari ekingo bagenio, eta eskuz egin beharko bagenu paper orria eta arkatza baliabide fisikoz lagundurik eta Euklides-en algoritmoa baliabide intelektualaz, hau egingo genuke:



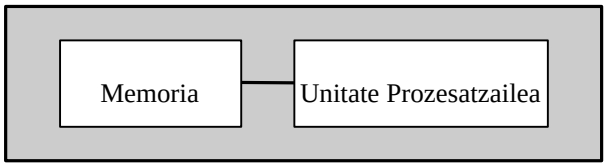
Euklides-en algoritmoaren prozesua gogoratuz:

1. Datuetan kopuru handienari M deituko diogu, kopuru txikiena N izango da.
2. M eta N arteko zatiketa osoa lortu hondarrari H deituz.
3. H zero ez denean, N balioa M-ri esleitu, H balioa N-ri erantsi eta bigarren urratsera itzuli; bestela, z.k.h. N-ren uneko balioa da

Beraz, datuak 36 eta 16 izanik z.k.h. eskuz kalkulatzeko betetzen ditugun zereginak bi dira. Batetik balio bat nonbaiten “gorde” behar dugu, geroago balio hori “eskuratu” ahal izateko,

adibidean 36 datua M deituriko aldagaian[4], zeregin hau *informazioaren biltegitzea* izango da. Bete beharreko bigarren zeregina *informazioaren prozesaketa* izango da, informazioa prozesatzean eragiketa matematiko eta eragiketa logikoak burutzen dira; adibidez H-ri dagokion balioa lortzeko M eta N-ren arteko eragiketa matematikoa den zatiketa egin behar da, prozesu errepikakorra eteteko H-k zero balio duen frogatzen da (eragiketa logikoa).

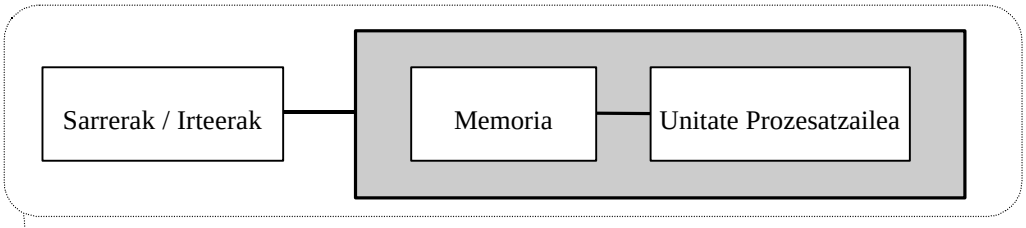
36 eta 16 datuen z.k.h. eskuz kalkulatu ordez makina batek egin beharko balu, honek ere *informazioa biltegitzeko* (gordetzeko) eta *informazioa prozesatzeko* (eragiketak) ahalmena izan beharko luke. Ordenadoreetan atal berezi bi daude funtzio horietaz arduratzen direnak: memoria eta unitate prozesatzailea:



Baina makina automatiko eta programagarriei eszitazioak ematen dizkio erabiltzaileak martxan jarri daitezen. Ordenadoreari ere, algoritmo programatua eta datuak eman beharko zaizkio eta programa hori exekuta dezan agindu beharko zaio. Horrez gain ordenadoreak kalkulatu duen emaitza erabiltzaileari erakutsiko dio (informazioa ordenadoretik kanporatuz).

Beraz, ordenadore baten zereginak lau izango dira, aurreko biak eta sarrera/irteera nozioekin loturik daudenak. Hona hemen makina algoritmiko baten zereginen zerrenda eta eskemarik sinpleena:

1. *Sarrerak onartu* Erabiltzaileak informazioa ordenadoreri emango dio, informazioa diogunean datuak eta programari buruz ari gara.
2. *Irteerak sortu* Informazioa ordenadoretik kanporatzen denean.
3. *Informazioa biltegitu* Informazioa ordenadorearen memorian pilatzen da, informazio hori memoriatik ordenadorearen beste ataletara mugi daiteke, eta, ordenadorearen gainerako barne-osagaietatik memoriara mugi daiteke ere.
4. *Informazioa prozesatu* Informazioa prozesatzeak alderdi bi izango ditu. Eragiketa aritmetikoak eta logikoak burutu behar ditu ordenadoreak, eta, programaren instrukzioak “ulertu” eta exekutatuko ditu.



Elementu hauen konbinazioak ordenadorearen hardwarea osatzen du eta makina programagarriaren aldaezina den zatia litzateke, hots, *oinarrizko makina* deitutakoa. Zati finko horri aldagarria den atala gehituko zaio makina algoritmiko osatua lortzeko, lehen aurreratu dugulako badakigu erraz alda daitekeen zatiari *programa* deitzen zaiola.

[4] M-ri aldagaia esaten zaio berak adierazten duen balioa aldagarria delako prozesuaren zehar, hasieran M-k 36 sarrerako datua ordezkatzen du baina aurrerago M-ren balioa 16 da.

1.3 MAKINA ALGORITMIKOEN MUGARRI HISTORIKOAK

Gizakion artean “lan minimoaren legea” edonon eta edonoiz aplikatu den araua izan da, da, eta izango da. Lan astunak eta errepikakorrak ekidin nahirik gabiltza mundu honetan; eta kalkuluak modu atsegin batean burutu ahal izateko asmatu dira hainbat tresna fisiko eta prozedura adimendutsu, horien artean ordenadoreak dira kate luze baten azken katemailak.

Egungo ordenadoreetara iristeko bide luze bat jorratu da, makina algoritmikoak nola bilakatu diren ikus dezagun. Mugarri historikoak multzoka taldekatu ditugu, nahiz eta euren artean jarraipen estua egon.

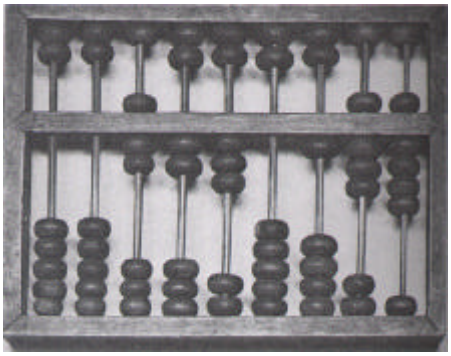
1.3.1 Ordenadoreen aurretikoak

Ordenadoreen aurretikoak hiru etapetan azalduko ditugu: aritmetikaren hastapenak zeintzuk ziren, kalkulugintza nola hasi zen automatizatzen eta historiako lehen makina programagarriek erakarri zituzten aurrerakuntzak.

1.3.1.1 Aritmetikaren hastapenak

Hasiera batean zenbaketak egiteko gizakiak atzamarrak erabiliko zituen, atzamarrak eta harrien bitartez lortuko zuen kopuruen adierazpidea. Kalkulurako ezagutzen dugun tresnarik zaharrena hortik ondoriotuko zen, dakigunez K.a. 3500 inguruan abakoa Mesopotamian ibiltzen zen eta handik Grezia eta Txinara zabaldu omen zen.

Abako batek hainbat ardatz ditu eta bakoitzean egurrezko, harrizko edo zeramikazko bolak mugi daitezke gora eta behera, bola horiek dituzten posizioak behaturik jakin daiteke zein zenbaki adierazten duten. Esate baterako 7230189 kopurua abako batean “idatzi” ondoren honelako egoera izango genuke:



0 0 7 2 3 0 1 8 9

Abakoa eskuz maneiatzen den tresna da zenbaki bat adieraz dezan. Behin kopuruaren

matematikoak burutzeko arauak definitu ziren (ikus kapitulu
aldizkariko artikuluak).

Matematikari hinduek asmatutako errepresentazio sistema hamartarra arabiarren

eta kulturetan. Baina, abakoa
herri askotan.

(1550 1617) matematikari eskoziarrak logaritmo nepertarrak asmatu zituen. Napier matema tikariaren lan teorikoak aurrerago eman zuen bere fruitu praktikoa, 1650ean eraiki zen lehen - dena bulego,

-

gurpil horzdunetan oinarritzen zen (autoetan kilometroak kontatzeko dagoen dispositiboaren ickard eta bere familia osoa eritasunak jota hil zen eta bere

- -

eta filosofoak - asmatu zuen. Schic -

gurpilez osaturik zegoen eta gurpil batean bira ematen zen bitartean zenbakiak 0 tik 9 ra - -

ezkerrean zegoen hurrengo gurpilean posizio

bezala, egungo autoen kilometro kontagailuaren printzipio bera da (azken finean erlojuen

-

hama

Leibnitz (1646 1716) pentsalari alemaniarrak, biderketak eta zatiketak ere egin zitzakeen beste - zuzena).

Pascal eta Leibnitz en diseinuak hobetuz kalkulagailu mekanikoak asko hedatu ziren

eta dendetan erregistrakutxa erraldoiak?.

Pascal eta Leibnitz en -

1822. urtean, - diseinatu eta eraiki zuen. - ren helburua zenbait

matematikoa z

[3] funtzioari dagokion taula lortu nahi dela. Bost zutabeko taula eraiki beharko litzateke, non zutabe bakoitzaren deskribapena hau litzatekeen: lehenengo zutabea x, bigarren zutabea y, hirugarren zutabea y_n eta y_{n+1} arteko kendura (lehenengo diferentzia), laugarren zutabea u_n eta u_{n+1} arteko kendura (bigarren diferentzia), eta, bosgarren zutabea konstantea den v_n eta v_{n+1} arteko kendura (hirugarren diferentzia):

		lehenengo diferentzia	*bigarren diferentzia*	*hirugarren diferentzia*
x	$y = x^3$	*u*	*v*	*w*
1	1			
		7		
2	8		12	
		19		6
3	27		18	
		37		6
4	64		24	
		61		6
5	125		30	
		91		
6	216			
7	**343**			

Ikus daitekeenez, lehenengo diferentziari dagokion zutabea honelaxe kalkulatzen da:

$u_n = y_{n+1} - y_n$

```
7 = 8 - 1
19 = 27 - 8
37 = 64 - 27
61 = 125 - 64
91 = 216 - 125
...
```

Bigarren diferentziari dagokion zutabea:

$v_n = u_{n+1} - u_n$

```
12 = 19 - 7
18 = 37 - 19
24 = 61 - 37
30 = 91 - 61
...
```

Hirugarren diferentziari dagokion zutabea:

$w_n = v_{n+1} - v_n$

```
6 = 18 - 12
6 = 24 - 18
6 = 30 - 24
...
```

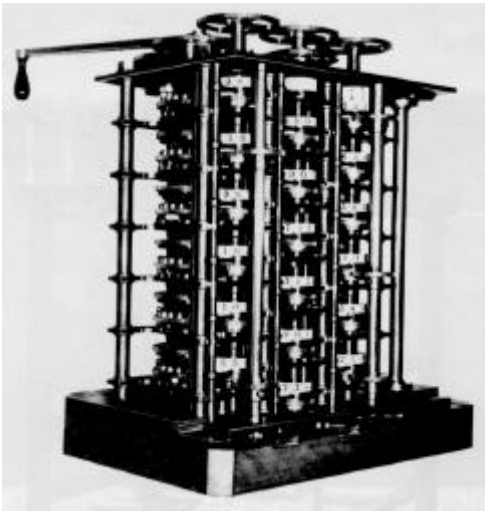
$y=x^3$ funtzioa hirugarren ordenakoa delako, hirugarren diferentzia konstantea izango da beti, hau da hain zuzen ere diferentzia finituen printzipio matematikoa eta edozein n ordenako polinomiotan ere betetzen da (n-garren diferentzia konstantea izango da).

Demagun $y=x^3$ funtzioaren balioa ezagutu nahi dela x=7 denean, 7^3 egingo bagenu 343 aterako litzaiguke. Emaitza bera eskura genezake diferentzia finituen metodoari esker, behar diren datuak aurreko taulan daude, hots, x=1 eta x=6 arteko taula izanez gero, ez dugu 7-ren kuboa kalkulatu behar nahikoa baita gezi beltzak adierazten duen zenbakiak batzea, euren batura 343 izango da, 7^3 alegia.

$7^3 = 343 = 6 + 30 + 91 + 216$

w (x=6) v (x=6) u (x=6) $y=x^3$ (x=6)

Babbagen *Kenketa-makina* aurreko operazioak egiteko gai zen, engranaiak erabilita diferentzia finituen algoritmoa “programatzea” lortu zuen Babbagek. Jarraian erakusten den irudian Babbagen *Kenketa-makina* agertzen da, ikus daitekeenez teknologia aldetik Leibnitz, Pascal eta Schickard makinen familiakoa da:

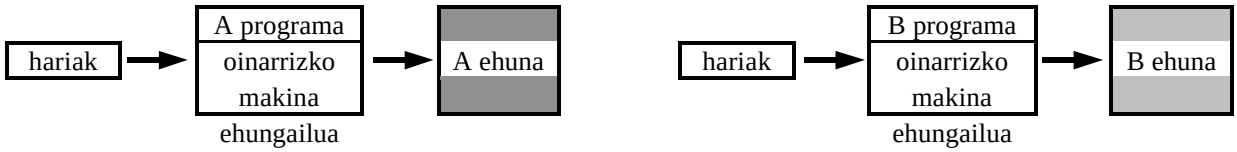


1.3.1.3 Programaren kokapena memorian

Abakotik hasita kutxa erregistratzaileraino ezin genitzake horiek automatikotzat har, eragiketa guztiak burutzeko, operatzailearen presentzia aktiboa ezinbestekoa baita. Aritmetika automatizatu ondoren, hurrengo aurrerapen kontzeptuala kalkuluak automatikoki kontrolatzea izan zen.

Historian martxan jarri zen lehen makina automatikoa arropak ekoizteko ehungailua izan zen. Ehuna egiteko eta ehunaren marrazkiak lortzeko txartel metalikoen bitartez egitea lortu zuen Joseph Marie Jaquard (1752-1834) asmatzaileak.

Txartel metalikoen multzo batek programa bat osatzen zuen, ehungailuaren funtzionamendua gidatzen zuen programa alegia. Beraz, Jaquard-en ehungailuak goimailako automatizazioa erakusten duen makina da, sortu zen lehen makina programagarritzat jo daiteke.



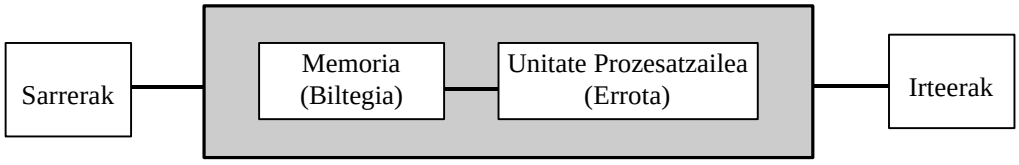
Jaquard-ek 1801. urtean Parisen aurkeztu zuen ehungailua, hamar urte beranduago Frantziako fabriketan horrelako hamar mila ehungailu baino gehiago martxan ari ziren. Beraz, Jaquard-en makinak arrakasta handia eduki zuen eta ehungintzarako aurrerapen galanta izan zen, baina automatizazioak sortu zuen langabeziak arazo sozialak erakarri zituen. Gaur egun Jaquard-en ehungailua erabiltzen da ere, baina txartel metalikoak baztertu dira eta makina kontrolatzeko programaren euskarria zinta magnetikoa da.

1.3.2 Ordenadore mekanikoak

Aritmetikaren automatizazioa eta makina programagarriaren kontzeptu biak Charles Babbage (1792-1871) matematikariak batu zituen. Babbage Informatikaren aitzindaritzat daukagu, izan ere berak aurresan zituen gaur egungo ordenadoreen barne egitura eta diseinua, 1834. urtean, *Makina Analitikoa* asmatu zuenean.

Lehenago azaldu dugun Babbagen *Kenketa-makina*k funtzio matematikoen taulak kalkulatzeko balio zuen, eta erabilitako diferentzia finituen algoritmoa modu finkoan jarrita zegoen makinan. Beraz, makina automatikoa litzateke **1.2.4.1** puntuan emandako sailkapenaren arabera, baina ez makina programagarria.

Kenketa-makina xede bakarreko tresna zen, baina Babbagek Jaquard-en ehungailuaren berri izan zuenean helburu orokorreko automatismo bat eraikitzea erabaki zuen, horrela jaio zen *Makina Analitikoa* deituriko proiektu berria, Makina Analitikoa xede orokorreko aparatua zen eta ondoko eskemari jarraitzen zion, ikusten denez Makina Analitikoaren osagaiak (Biltegia, Errota, Sarrera-sekzioa, Irteera-sekzioa) egungo ordenadoreen berberak dira:



Diseinuz, Makina Analitikoak lau atal zituen:

Memoria	Babbagek biltegia deitu zuen eta informazioa gordetzeko balio zuen. Informazioa gordetzeko Jaquard-en txartel zulatuen ideia bereganatu zuen, informazioz makinaren programa eta programak behar zituen datuak eta bitarteko emaitzak ulertzen da.
Errota	Programaren instrukzioen exekuzio-sekuentzia erregulatuko zuen, eta, beharko liratekeen eragiketak burutuko zituen.
Sarrerak	Txartel zulatuen irakurgailua, informazioa makinaren memorian sartzeko unitatea.
Irteerak	Programak lortutako emaitzak makinatik kanporatuko zuen unitatea, diseinuz inprimagailu bat zen Makina Analitikoaren irteera unitatea.

Makina Analitikoaren ezaugarririk aipagarriena xede orokorreko makina izatean zetzan, hau da, Makina Analitikoa automatikoa izateaz gain programagarria bazen ere. Bere funtzionamenduan txartel zulatuetako instrukzioak irakurri eta exekutatu egiten zituen; agindu baten adibidea hau litzateke “biltegitik bi zenbaki hartu, errotara eraman, hauekin batuketa burutu eta emaitza biltegira bidali”.

Makina Analitikoak kalkulu desberdinak egiteko gauza zen Sarrera-sekzioan txartel zulatu aproposak jarriz, txartel zulatu multzo jakin batek programa bat osatzen zuelako garai hartan hasi ziren lehendabiziko software ekimenak.

Beraz, Makina Analitikoa automata programagarri bat zen. Baina zoritxarrez Babaggen ideiak ez ziren erabat gauzatu Makina Analitikoa ez zelako bukatu, hala ere bere ekarpena funtsezkoa izan zen eta gaur eguneko gure ordenadoreek duten diseinua Makina Analitikoaren diseinuan oinarritzen da, nolabait esateko gure ordenadore modernoak Makina Analitikoaren bertsio gaurkotuak dira.

1.3.3 Ordenadore elektronikoak

Ordenadore elektronikoak mende honetan asmatu ziren eta bigarren gerrate mundialean erabili ziren. Ordenadore elektronikoek zirkuitu logikoz osaturik daude, eta zirkuitu logikoak diseinatzeko Boole-ren Algebra erabiltzen[5] da. XIX. mendearen erdi aldean, 1847. urtean, George Boole (1815-1864) matematikariak bere izena hartu zuen teoria aurkeztu zuen (laugarren kapituluan hitz egingo dugu Boole-ren Algebrari buruz).

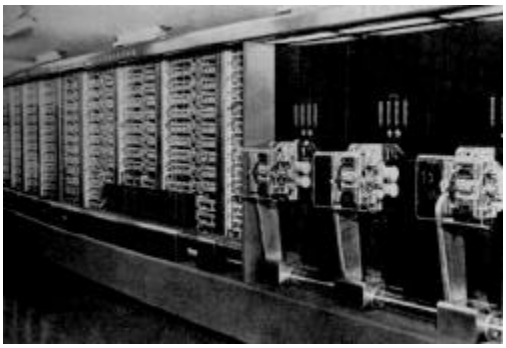
1939. urtean Konrad Zuse ingeniari alemaniar gazteak Z1 deituriko ordenadore programagarria eraiki zuen ingeniaritza kalkuluak errazteko balio zuena, Z1 konputadorea errelez (konektore elektromagnetikoak) egina zegoen eta xede orokorreko makina zen. 1944an aliatuek Berlin bonbardatzean Zuseren ordenadore guztiak suntsituak izan zirenez bere lanek ez zuten ondorengo makinetan eragin zuzenik eduki.

1943. urtean Alan Touring matematikariaren diseinatutako eta Gobernu britaniarrak bultzatutako Colossus izeneko ordenadorea martxan jarri zen. Colossus ez zen xede orokorreko makina izan, bere helburua militar alemaniarren komunikaziorako kode sekretuak

[5] Boole izena ezaguna egiten zaio irakurleari Multzoen Algebra ikasi duelako, Multzoen teoria Logika zientziarekin loturik dago. Logikan esaldiak ebaluatzen dira, aritmetikan espresio matematikoak ebaluatzen diren bezala, horretarako Logika zientziak bi balio erabiltzen ditu `EGIA` eta `GEZURRA`, ordenadore digitalek ere bi balio bakarrekin lan egiten dutenez euren barne osagaiak diseinatzeko zirkuitu logikoak oso aproposak dira.

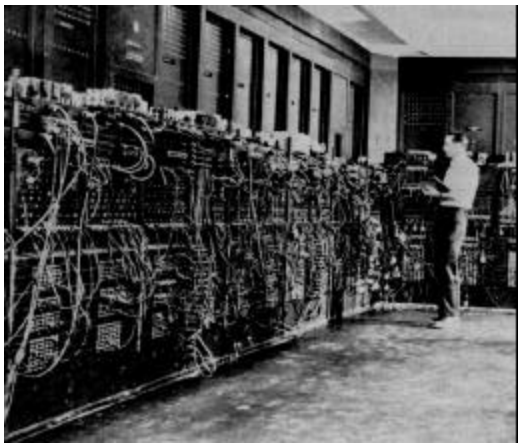
deskodetzea[6] zen, bai eta lortu ere. Colossus-en teknologian huts-balbulak erabili zirelako ordenadore elektronikoa kontsidera daiteke, aipatzeko da lan honek ez zuela jarraipenik izan Gobernu britaniarrak proiektua sekretu militar gisa sailkatu zuelako.

Aurreko biak baino eragin gehiago izan zuen Mark I kalkulagailuak, 1944. urtean hasi zen lanean eta Babaggen diseinuetan oinarrituta zegoen baina teknologia mekanikoa izan ordez elektromagnetikoa zen erreleak erabiltzen baitzituen. Mark I kalkulu-makina benetako erraldoia zen jarraian erakusten den argazki parean ikus daitekeenez:



Bigarren Mundu Gerra ordenadoreen garapenerako bultzagarria izan zela ukaezina da, armada amerikarrak eskaturik kanoiek jaurtitako balen ibilbidea kalkulatzeko ENIAC[7] eraikitzen hasi ziren 1943an. Egia esan ENIAC-ek ezin zuen era praktikoan bere ahalmenak frogatu, eraiki zen ordurako gerra amaiturik baitzegoen, baina proiektua egingarria zela eta komertzialki onuragarria zela aski ongi egiaztaturik geratu zen.

ENIAC konputagailua 30 tonako beste munstro bat zen eta huts-balbulen teknologia elektronikoari esker Mark I baino askoz bizkorragoa zen. Hura programatzeko barne egitura ezagutu beharra zegoen, makinaren zeregina gidatzen zuen programa aldatzeko, argazkian ikus daitezkeen bezalako konexiorako kable, entxufe eta posizio anitzeko 6000 etengailu antolatu behar ziren:

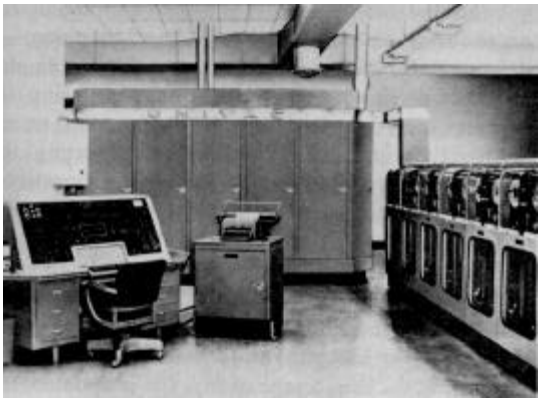


[6] Bigarren Mundu Gerraren hasieran urpekuntzi alemaniarrek kalte handiak sortzen zituzten britaniar armadan. Berlineko almirantegoak irratiz bidaltzen zituzten aginduak urpekuntziei, mezuak, nola ez, enkriptatuta zeuden, ENIGMA zeritzan gailu baten bidez kodetuta zeuden mezuak. Gerraren hasieratik britaniarrek ENIGMA makina bat zeukaten (poloniarrek lapurtutakoa), nahiz eta kodetzeko gailua eduki kodetutako mezu bat deskodetzeko kalkulu asko egin behar zen eta denborak beste zailtasun bat jartzen zion eginkizunari, eskuz ezin zitekeena egin Colossus-ek lortu zuen.

[7] Electronic Numerical Integrator and Computer (Konputagailu eta Zenbakizko Integragailu Elektronikoa).

ENIAC konputagailuaren kable eta etengailuen posizioak eskuz jarriz programa lortzen zen, konputagailua programatzea lan nekeza, astuna, luzea eta malgutasunik gabekoa zen. Benetako makina programagarriak lortzeko, lehenago aipaturiko kontzeptua gogora ekarriz, programa memorian kokatzea ezinbestekoa zen, horretarako programak digitalki errepresentatu behar ziren makinak elektronikoak baitziren (makina elektronikoek bi egoera ezagutzen dituzte 0 eta 1 deitzen direnak). Beste hobekuntza bat, zenbakien aritmetikarekin loturik zegoen, ordurarte ordenadoreek gizakiontzat arrunta zaigun aritmetika hamartarra zerabiltzaten, horren ordez aritmetika bitarra erabiltzeak makina efizienteagoak erdiestea ekarri zuen. Ezaugarria hauek guztiak biltzen zituen lehen ordenadorea 1949. urtean eraiki zen eta EDSAC (Electronic Delay Storage Automatic Calculator) deitu zioten, bere arkitektura gaur eguneko ordenadoreena delako **1.3.4** puntuan zehaztuko dugu.

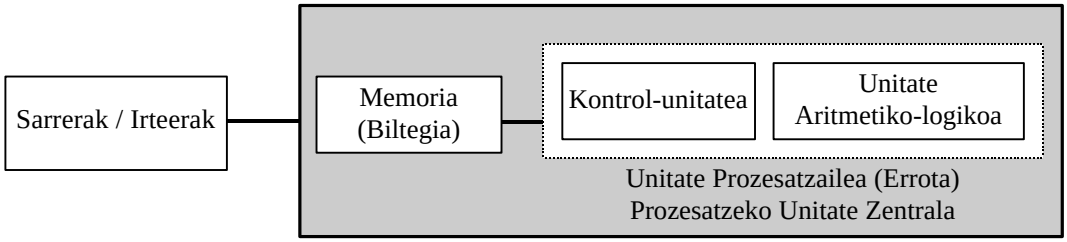
Ordenadore modernoen garapenari bukaera emateko, laborategi prototipo izatetik kontsumo-gaia izatera heldu zen lehen konputagailua aipa dezagun. Merkatuan saldutako aurreneko konputagailua UNIVAC (Universal Automatic Computer) izan zen eta 1951.ean Estatu Batuetako gobernuaren Errolda Bulegoan instalatu zen, UNIVAC konputagailuak zuen itxura hau da:



1.3.4 Gaur egungo makina algoritmikoen arkitektura

Historiak diosku Charles Babbage izan zela konputagailu modernoen osagaiak finkatu zituena, beraz ordenadore arkitektura XIX. mendearen hasieran ezagutzen zela onar daiteke. Baina Charles Babbagen aparatuei begirada bat zuzenduz, eta denontzat ezagunak diren gaur egungo ordenadoreekin konparatzean, zaila da euren arteko baliokidetasunak identifikatzea. Puntu honetan ordenadoreen barne osagaien deskribapen arin bat egingo dugu, ordenadorea algoritmoak exekutatzen dituen makina delarik zeintzuk diren bere atal nagusiak alegia.

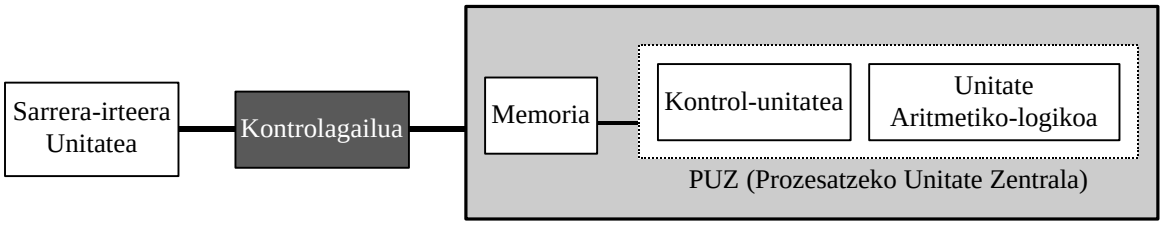
Berreskura dezagun Babbagek erabilitako arkitektura, non Sarrera eta Irteera sekzio biak elkarrekin jartzen ditugun eta Babbagek errota deitu zuen Unitate Prozesatzailea[8] bi zatitan banatu ditugun Kontrol-unitatea eta Unitate Aritmetiko-logikoa:



[8] Ordenadoreen unitate prozesatzaileari CPU (Central Processing Unit) esaten zaio, eta euskaraz PUZ (Prozesatzeko Unitate Zentrala) akronimoa erabiltzen da.

Ordenadorearen atal nagusien azalpen laburra egin dezagun. Une honetan atalen zeregina zein den definituko da, baina bakoitzak bere lana nola betetzen duen ez da azalduko.

Sarrerak / Irteerak Erabiltzailearekiko komunikazioa gauzatzen du Sarrera-irteera Unitatea delako elementu honek. Sarrera-irteera Unitatea edo Sarrera-irteerako Unitatea (teklatua, sagua, monitorea, inprimagailua, disko unitateak, bozgorailuak, ...) periferikoen multzoa da, eta ez da Memoria-PUZ blokera zuzenean konektatzen, lotura hori kontrolagailu baten bitartez erdiesten da:



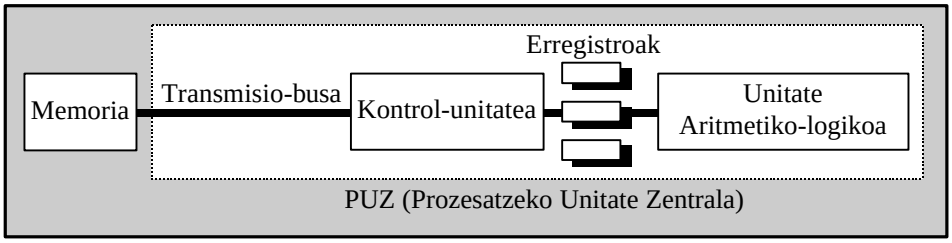
Sarrera-irteera Unitate bakoitzari bere kontrolagailua dagokio eta diseinu estrategia honi esker ordenadorearen PUZ ez da periferikoetaz arduratzen eta aske geratzen da programa definitzen duen algoritmoa lantzeko. Kontrolagailua, definizioz, periferikoen funtzionamendua kontrolatzen duen aparatua da, eta driver deituriko programa bat jarraituz kontrolatzen da periferikoa (horregatik inprimagailu berri bat konektatzen denean, gehienetan driver bat ere instalatu[9] beharko dugu ordenadorean).

Memoria Babbagen makinetan informazioa errepresentatu eta gorde ahal izateko engranai multzoen bitartez egiten zen, engranaien posizioak zenbaki bat ematen zuen. Egungo ordenadoreetan engranaiak ordezkatu dira zirkuitu elektronikoak jarriz, horietako zirkuitu elektroniko batek digitu bakarra gordetzeko ahalmena du, hots, memoria osatzen duen zirkuitu elemental banaren edukia 0 ala 1 izango da.

Memorian gordeak izan daitezen datuak sistema bitarrean jarriko dira (gai hau 2. kapituluan ikasiko dugu). Ordenadorearen funtzionamendua gidatzen duen programa memorian kokatzen dela dakigu, beraz programa ere kode bitarrean idatzirik gordeko da memorian.

Prozesatzeko Unitate Zentrala Programaren partaideak diren aginduak interpretatu eta exekutatzeko ahalmena duen osagaia.

Prozesatzeko Unitate Zentrala ordenadore batean Unitate Aritmetiko-logikoa, Kontrol-unitatea, Transmisio-busa eta Erregistroak biltzen dituen unitate funtzionala da.



[9] Driver horrek periferikoa kontrolatuko du eta fisikoki programa edo instrukzio-multzo bat da, hura instalatzean instrukzio-multzoa Sistema Eragilean integratzen da.

Erregistroak informazioa gordetzeko ahalmena dute, eta horri begira memoriaren zirkuitu elementalak bezalakoak lirateke, baina bizkorragoak izateaz gain ordenadorearen atal desberdinetan sakabanaturik aurkitzen dira (Memoria aldiz toki bakar batean ditu kontzentraturik bere osagaiak). Erregistroak Transmisio-busera konektaturik daude eta ordenadoreko atal desberdinen artean informazioa trukatu ahal izateko balio dute.

Transmisio-busa. Paraleloan antolatutako hainbat eroalen multzoa da, busak hiru dira: Datu-busa, Helbide-busa eta Kontrol-busa.

Kontrol-unitatea. Kontrol-unitateak kalkulu sistema osoaren funtzionamendua kontrolatzen du. Programaren edozein agindu memoriatik eskuratu ondoren interpretatu eta gero, instrukzioa exekutatu ahal izateko behar diren seinaleak beste unitateetako zirkuituetara bidaltzen ditu. Esan daiteke beraz, Kontrol-unitatea sistema osoa gidatzen duen unitatea dela.

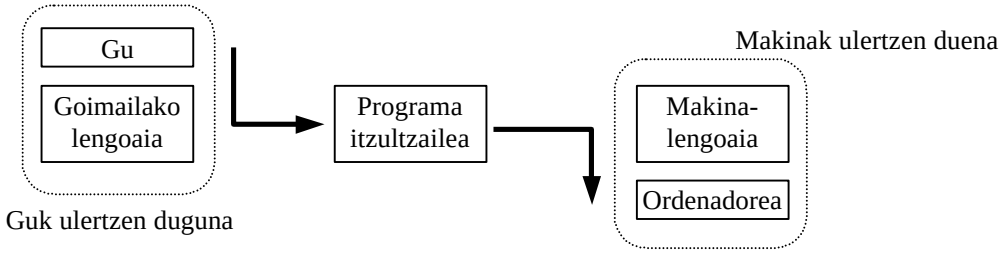
Unitate Aritmetiko-logikoa eragiketak burutzen dituen atala da. Eragiketak bi motatakoak izan daitezke: aritmetikoak (batuketa, kenketa, ...) eta logikoak (NOT, AND, OR, ...)

1.4 PROGRAMAZIO-LENGOAIAK

Algoritmo bat zer den ikasi dugu, ikusi dugu ere algoritmo bat errepresentatzeko era asko dagoela (hizkuntza naturala, diagramak, irudiak, ikurrak, ...). Konputagailuaren lehen aurkezpena egitean, makina algoritmikoa den aldetik, instrukzio zerrenda bat exekutatzeko ahalmena duela esan dugu; eta dakigunez, ordenadoreari instrukzio multzoa programa bezala ematen zaio.

Dagoenekoz ordenadore programa bat ikusi dugu (pentsatutako zenbakia asmatzen diguna) eta Turbo Pascal lengoaian idatzi izan da. Lengoaia da, hain zuzen ere, algoritmo eta ordenadorearen arteko lotura: *lengoaia bat sinbolo multzoa eta sinboloak erabiltzeko arau multzoa da, bere helburua komunikazioa lortzea delarik*. Lengoaia batek dituen ezaugarriak hiru lirateke, batetik erabil daitezkeen sinboloak, bestetik lengoaiaren *sintaxia* gauzak nola esan behar diren (sinboloak nola konbinatu behar diren aginduak osatzeko), eta, hirugarren eta azkenik lengoaiaren *semantika* (sintaktikoki zuzena den agindu baten esanahia zein den).

Ordenadorearen zirkuituek instrukzioak interpretatu eta exekutatzen dituztela jakina da, baina gehiago zehaztuz, ordenadore batek “ulertzen duen” lengoaiari *makina-lengoaia* esaten zaio. Programadoreek ez dute, gehienetan, makina-lengoaian programak idazten goimailako lengoaia batean baizik. Honez gero, algoritmo bat ordenadoreari ematean dagokion makina-lengoaian kodifikatu beharra dago, ikus irudia:



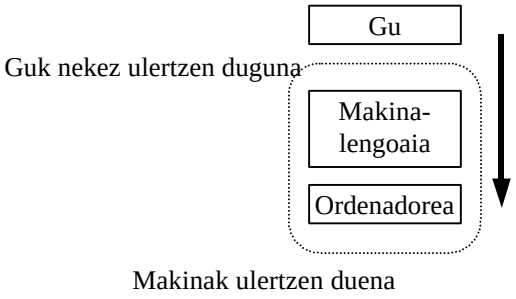
Irudi hau jarraiko puntuetan azalduko dugu.

1.4.1 Makina-lengoaia

Konputagailuak osagai elektroniko digitalez osaturik daude, horregatik konputagailuek bitak (0 eta 1 egoerak) prozesatzeko ahalmena daukate eta ez dute besterik ulertzen. Hau da, digitu bi horiek erabilita eman beharko zaizkio datuak, eta, digitu bi horiek erabilita programatu beharko zaio datuak prozesatzen dituen algoritmoa.

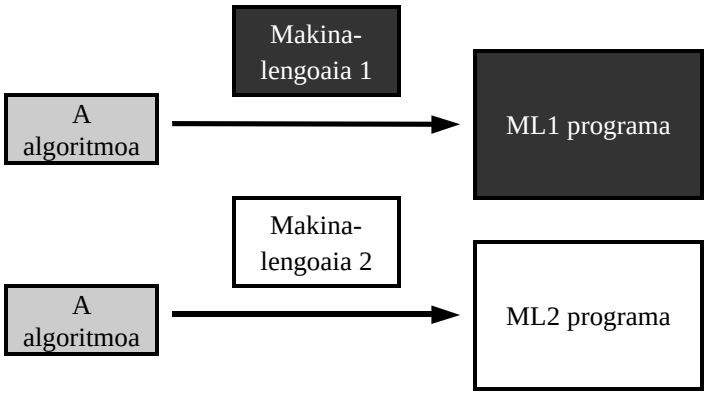
Merkatuan dauden konputagailuen Prozesatzeko Unitate Zentralak, orokorrean, desberdinak dira, horregatik PUZ bakoitzak berarentzat eginiko makina-lengoaia du. Hortaz, *makina-lengoaia* honela definituko dugu: *konputagailuak berez ulertzen duen lengoaia bakarra, kode bitarrez adierazita dago eta makinaren diseinuaren menpekoa da*.

Ordenadore jakin batetarako programak egitea makina-lengoaia erabiliz aurreko irudiaren eskuin aldea jorratzea litzateke. Hau da, programadoreak makina-instrukzioen[10] bitartez programatuko luke algoritmoa:



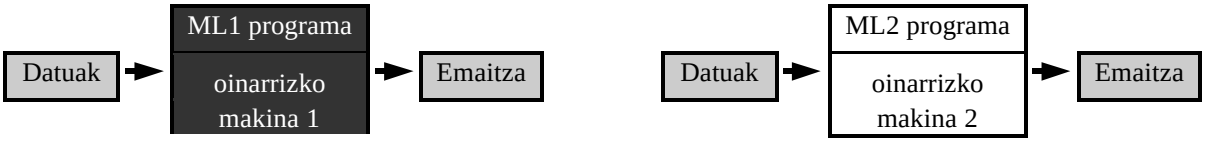
Baina gizakiarentzat, beste kapitulu batean esperimentatuko dugun bezala, oso zaila da makina-lengoaian programatzea (erroreak erraz sortzen direlako, gainera programaren errakuntzak bilatzea benetan zaila delako, programa ondo dabilela hobekuntza bat egitea lan astuna delako, ...). Guzti honi beste desabantaila handi bat gehitu beharko litzaioke, ordenadore jakin batetarako bere makina-lengoaian idatzitako eta frogatutako programa, PUZ desberdina duen beste ordenadore batean ez dabilela.

Beraz, makina-lengoaiak erabiliz programatu beharko bagenu, ordenadoreak diseinuz desberdinak direlako lan bera errepikatu beharko genuke. Algoritmo bera daukagu (diagrama baten bitartez errepresentatuta adibidez), eta horren kodifikazioa eginez lortuko genuke ordenadore bati dagokion programa, baina makina-lengoaia desberdina duen ordenadore batetarako algoritmo bera berkodetuz beste programa bat lortuko genuke:



[10] Konputagailu zehatz batek zuzen-zuzenean ulertu eta exekuta dezakeen agindua, makina-instrukzioen multzoa konputagailu horri dagokion Makina-lengoaia litzateke.

Funtzionalki ML1 eta ML2 programak baliokideak dira hurrengo irudian adierazten den bezala, baina kodeak konparatuz guztiz desberdinak izan daitezke.



1.4.2 Itzultzaileak

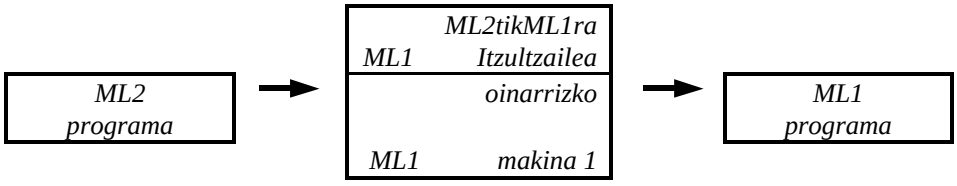
Aurreko iruditik ondorio bi ateratzen dira.

Bat, makina programagarri baten oinarrizko atalak (hardwareak) markatzen du programa nolakoa izan behar duen. Ordenadore bati dagokion PUZ atalari *oinarrizko makina 1* deituz, horrek onartzen duen programa *makina-lengoaia 1* erabiliz lortu da eta beste programek ez dute balioko. Beste irudi honetan ordenadore pare bat adierazi nahi izan dugu *ML1* eta *ML2* makina-lengoaiak ulertzen dituztenak hurrenez hurren.

Bi, ordenadore baten programa bere makina-lengoaian idatzirik ez badago itzuli beharko da interpreta eta exekuta dezan. Itzulpen lana pertsona batek egin dezake, edo zergatik ez, programa batek egin dezake ere. Pentsa dezagun *ML1* makina-lengoaiaren bidez *ML2tikML1ra Itzultzailea* deituriko programa itzultzailea idatzi dela[11], duen helburua *ML2* lengoaian dagoen programatik abiatuta bere ordezkoa *ML1* lengoaian lortzea da. Eta, demagun *ML1 oinarrizko makina 1* batean daukagun *ML2 programa* exekutatu nahi dugula, argi dagoenez bateragarriak ez direnez ezingo dugu elkarrekin lanean uztartu. Soluzioak urrats bi ditu:

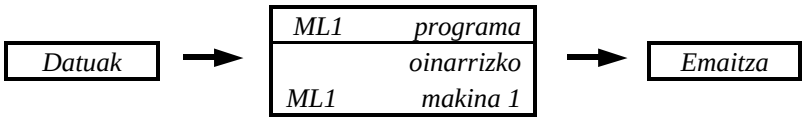
1. *ML2tikML1ra Itzultzailea* programa erabiliz, *ML2 programa* izena duen programari dagokion ordaina lortu (*ML1 programa* deituko duguna)
2. *ML1 programa* deituriko programa, daukagun *ML1 oinarrizko makina* ordenadorean kargatu eta datuak emanez arazorik gabe egikaritu

Aurreneko urratsean ordenadorea gidatuko duen programa *ML2tikML1ra Itzultzailea* izango da, itzultzailea *ML1* makina-lengoaiaren bitartez idatzita dagoenez ez du arazorik sortuko. Datutzat itzuli behar duen *ML2 programa* onartuko du eta *ML1 programa* eskainiko digu emaitza bezala:



[11] Daukagun ordenadoreak ulertzen duen lengoaia *ML1* da, eta programa itzultzailea ordenadore horretan exekuta daiteke *ML1* lengoaian idatzita dagoelako.

Bigarren urratsean *ML2 programa* exekutatuko dugu *ML1* ordenadorean, egia esan *ML2 programa*-ren itzulpena den *ML1 programa* da exekutatuko duguna. Horretarako kargatuta dagoen programa itzultzailea ordenadoretik kendu eta *ML1 programa* jarriko dugu bere memorian:



Emaitza bera izango genuke *ML2* modeloko ordenadorea edukiko izan bagenu eta bertan hasieratik genuen *ML2 programa* datu berdinekin exekutatuko bagenu.

1.4.2.1 Mihiztadura-lengoaia

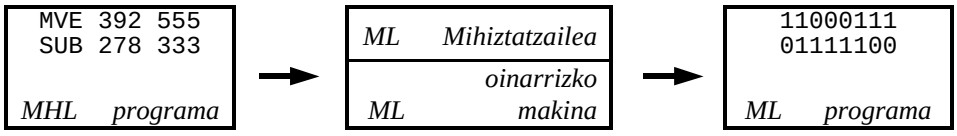
Makina-lengoaia ordenadorearen diseinuaren araberakoa da eta dituen sinboloak bi direla gogoratzen dugu. Lengoaia mota honek onartzen dituen sinbolo bakarrak zeroak eta batekoak direlako guretzat zaila da programatzea, nahiz eta makinak ulertzen duen bakarra izan. Gurengandik urrun dagoelako eta makinaren beharretatik oso hurbil dagoelako makina-lengoaiari behemailako[12] programazio-lengoaia dela esaten da.

Mihiztadura-lengoaia behemailako programazio-lengoaia da ere, zaila gizakiontzat eta makina-lengoaiaren antzekoa. Mihiztadura-lengoaia batean sinbolo nemonikoak erabiltzen dira makina-lengoaiaren instrukzioak idazterakoan, adibidez mihiztadura-lengoaian instrukzio bat `SUB` izan daiteke, ordenadoreak ez du `SUB` agindua ezagutzen berak `10110111` bezalako instrukzioak onartzen dituelako. Mihiztadura-lengoaian eragiketen eragigaiak idaztean `0` eta `1` digituen segidak jarri beharrean sinbolo nemonikoak erabiltzen dira ere.

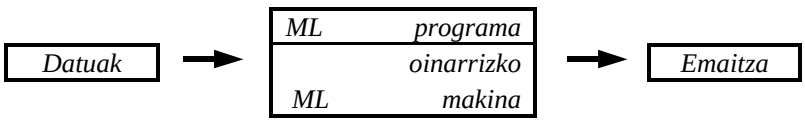
Mihiztadura-lengoaian idatzitako programa bat makina-lengoaia zehatz batera igarotzeko itzultzaile egoki bat behar da, *mihiztatzaile* deitzen dena. Mihiztadura-lengoaiak ez du makinarekiko dependentzia gainditzen, honez gero aurreko programa hori beste makina desberdin batean exekuta dadin behar den itzultzailea, mihiztatzilea, desberdina izango da ere.

Mihiztatzaile aproposa dugularik bi dira eman beharreko urratsak programa bat exekutatzean. Lehenengoan, ordenadoreak mihiztadura-lengoaian idatzitako programa (*MHL programa* deitu duguna) itzuli egiten du makina-lengoaian dagoen *ML programa* lortuz. Bigarrenean, datuak eman eta emaitzak jaso egiten da.

Lehenengo urratsa:

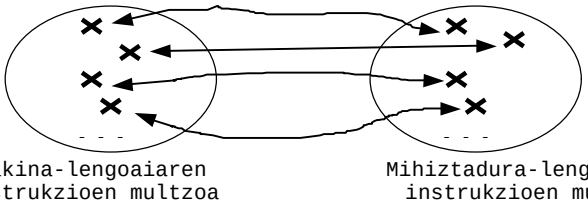


Bigarren urratsa:



[12] Behemailako programazio-lengoaia goimailako programazio-lengoaiari kontrajartzen zaio.

Amaitzeko, ez dezagun ahaz ordenadore zehatz bati dagokion mihiztadura-lengoaiak dituen instrukzioen kopurua, eta, ordenadore horri dagokion makina-lengoaiak dituen instrukzioen kopurua berbera dela. Izan ere, mihiztadura-lengoaia zeroak eta batekoak “ezkutatzen” dituen makina-lengoaia kontsidera daiteke. Ondorioz, mihiztadura-lengoaiatik makina-lengoaiara itzulpena egin behar duen mihiztatzaile programa erraza izango da, beste modu batean esanik, azken finean instrukzioen bi multzoen arteko erlazioa biunibokoa da.



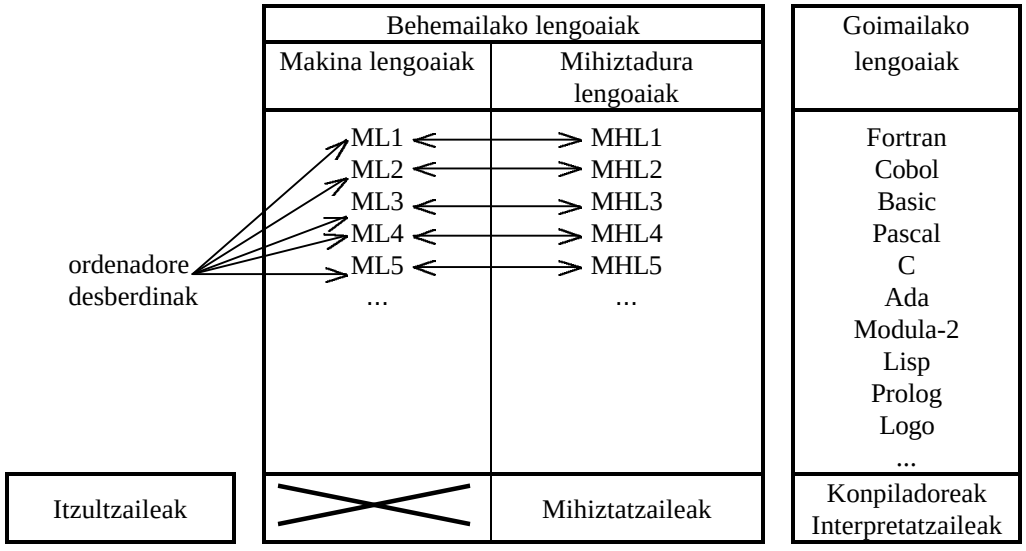
1.4.2.2 Konpiladoreak eta interpretatzaileak

Aipatu ditugun behemailako programazio-lengoaiak gizakiarentzat ulertzeko zailak direlako arazo larriak sortarazten dituzte, horregatik 1950. hamarkadan goimailako lengoaiak asmatu ziren. Goimailako programazio-lengoaia bat, gure lengoaia natural aberatsaren eta ordenadoreen lengoaia zehatzaren artean dago.

Goimailako programazio-lengoaia batek algoritmoak errepresentatzeko balioko du lengoaia formala delako, lengoaia formala honelaxe definitzen digu hiztegiak: komunikazioa batere anbiguotasunik gabe egiteko aukera ematen duten termino eta arau sintaktikoak.

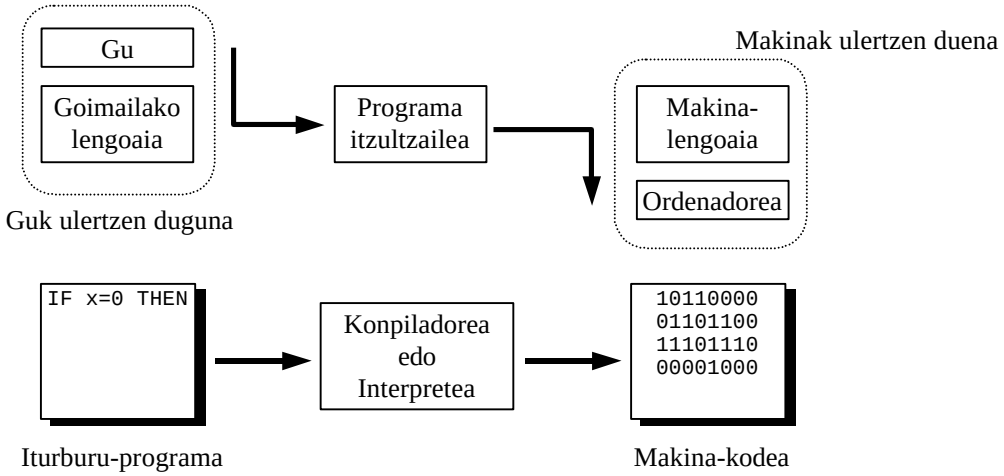
Goimailako lengoaiak gizakion lengoaia naturaletik hurbil dagoenez algoritmoak errazago idatz daitezke behemailako programazio-lengoaiekin baino, gainera goimailako lengoaiaz lortutako programa makinarekiko dependentziarik ez du izango (kontzeptu hau oso garrantzitsua delako aurrerago ere errepikatuko dugu bere ondorioak azpimarratuz).

Goimailako programazio-lengoaia garrantzitsuenei buruzko deskribapen laburrak **1.4.3** puntuan bilduko ditugu, baina hori baino lehen behemailako eta goimailako lengoaiak konpara ditzagun taula bat eginez:



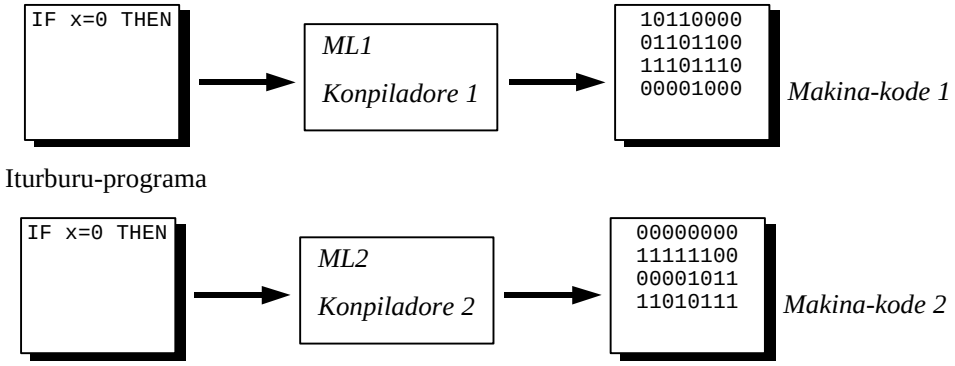
Goimailako programazio-lengoaiak ordenadorearen makina-lengoaiara itzuli behar dira, eta itzulpen lana nola egiten den arabera programa-itzultzaileak bi motatakoak izan daitezke: konpiladoreak eta interpretatzaileak.

Beraz, algoritmo bat ordenadoreari ematean gure hizkuntza naturaletik hurbil dagoen goimailako programazio-lengoaiaren bat aukeratuko dugu, ordenadoreari dagokion makina-lengoaian kodifikatu beharra dagoenez itzultzaile egokia hartuko da makinak uler dezakeen instrukzio multzoa lortzeko, ikus berriro irudi hau:



Goimailako lengoaia baten bitartez algoritmo bat programatzen dugunean lortzen dugun aginduen testuari *iturburu-programa* edo *iturburu-kode* esaten zaio (zenbakia asmatzen duen programa gogoratu). Konpiladore edo interpretatzaile bati esker daukagun iturburu-programa, ordenadorearekin bat datorren, instrukzio bitarrak lortzen dira, makina-lengoaian idatzirik dagoelako exekutagarria den instrukzio multzo honi *makina-kode* esaten zaio.

Demagun dugun iturburu-programa bat, ordenadore desberdinetan exekutatu beharra daukagula, demagun ere ordenadore bakoitzari dagokion konpiladore biak ditugula. Hona hemen irudikatuta, iturburu-programa bakarretik ondoriotzen diren makina-kode desberdinak:

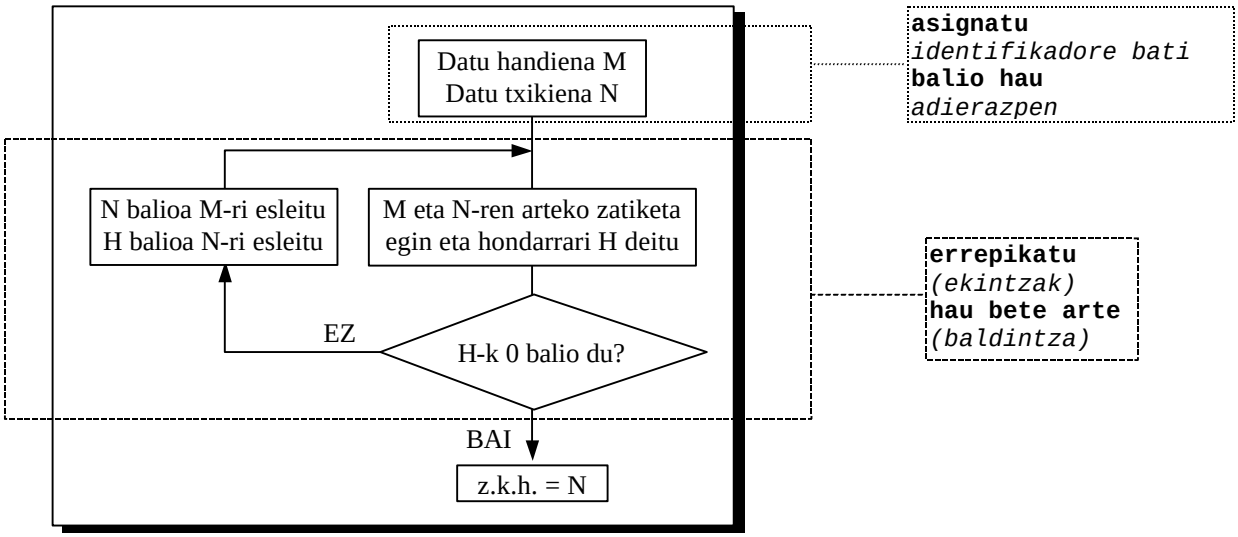


Ordenadore bakoitzari dagokion makina-kodea desberdinak dira duten zero eta batekoen segidak aztertzen baditugu, baina, eta oso garrantzitsua dena, makina-kodeak desberdinak izan arren funtzionalki erabat baliokideak dira. Hau da, *Makina-kode 1* programa hartu eta *Ordenadore 1* konputagailuan datu batzuekin exekutatzen badugu, eta modu beretsuan, *Makina-kode 2* programa dagokion *Ordenadore 2* konputagailuan datu berdinekin exekutatzen badugu, lortuko diren emaitzak berdin-berdinak izango dira.

Kontura gaitezen ordenadore desberdinetan iturburu-programa bera exekuta daitekeela itzultzaile egokiaren laguntzarekin, hau ezinezkoa zen mihiztadura lengoaiekin lan egitean. Horregatik behemailako programazio-lengoaiek makinarekiko dependentzia dutela esaten da, bestalde goimailako lengoaiak makinagandik independenteak direla esaten da.

Goimailako programazio-lengoaiak sortzeko erabili zen estrategia oso sinplea da, gizakiaren pentsamoldetik hurbil zeuden primitibo sorta bat identifikatzea aski zen. Esate

baterako, eta berriz ere, Euklides-en algoritmoa gogoratuz bi dira gutxienez aurkitzen diren primitiboak; batetik ordenadorearen memorian balio bat gordetzea eta bestetik zatiketaren hondarrak zero balio arte aginduak errepikatzea:



Horrek esan nahi du programa bat sortu nahi denean programadoreak goimailako primitiboak erabiliko dituela, aintzat hartu gabe programa hori zein makinetan exekutatuko den, ondorioz iturburu-programa makinagandik independentea da. 36 eta 16 datuekin Euklides-en algoritmoa goimalako lengoaien primitiboaz adierazita hauxe litzateke:

```
asignatu M balio hau 36
asignatu N balio hau 16

errepikatu
|       asignatu H balio hau M eta N arteko hondarra
|       baldin eta H <> 0 orduan
|       |       asignatu M balio hau N
|       |_____  asignatu N balio hau H
hau bete arte H = 0

erakutsi N
```

Euklides-en algoritmoa pseudokodean idatzirik lau primitibo agertzen dira, goimailako lengoaia batera igaroz iturburu-programa hau geratzen da:

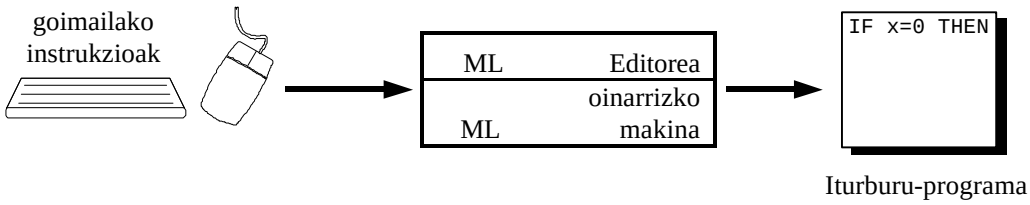
```
PROGRAM Euklides ;                          { \TP70\01\EUKLIDES.PAS }
VAR
   M, N, H : Integer ;
BEGIN
   M := 36 ;                                { asignatu ... }
   N := 16 ;                                { asignatu ... }
   REPEAT                                   { errepikatu ... }
      H := M MOD N ;                        { hondarra kalkulatu eta asignatu}
      IF H <> 0 THEN                        { baldin eta ... }
      BEGIN
         M := N ;
         N := H ;
      END ;
   UNTIL H = 0 ;
   WriteLn ('z.k.h. = ', N) ;               { erakutsi ... }
END.
```

Dakigunez makinak ezin dezake iturburu-programa exekutatu, eta programa itzultzaile[13] baten bitartez, makinari dagokion, makina-kodea lortu behar da. Baina edozein makina izanik `Euklides` deitu dugun aurreko programa hori, behin idatzi ondoren, ez dago zertan aldatu behar 36 eta 16 bi zenbaki osoen zatitzaile komunetan handiena kalkulatzeko.

1.4.2.2.1 Konpiladoreak

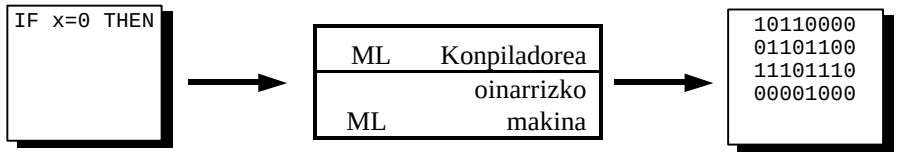
Konpiladore bat programa bat da eta bere helburua, iturburu-programa bati dagokion makina-kodea lortzea da. Ikusi dugun bezala diseinuz goimailako lengoaia batek onartzen dituen egiturak gizakiaren ulermenetik hurbil daude (**`asignatu`**, **`errepikatu`**, **`baldin eta`**, **`erakutsi`**, ...), eta mihiztadura-lengoaiaren kasuan bezala ordenadore konkretu baterako itzuli beharko da. Goimailako lengoaiaz idatzitako programa bat itzultzeko *konpiladorea*[14] edo *interpretatzailea*[15] deituriko programak erabiltzen dira, kontzeptualki mihiztadura-lengoaiarako egiten den itzulpenaren parekoa litzateke konpiladoreak eta interpretatzaileak egiten dutena, baina azken bi hauek mihiztatzaile batekin alderatuz askoz ere programa konplexuagoak dira.

Goimailako lengoaian idatzitako programa, irakurgarria izango zaigun testu bat izango da (Pascal lengoaian `BEGIN`, `IF`, `REPEAT`, `FOR`, ... bezalako hitzak agertuko dira). Testu horri iturburu-programa esaten zaio, eta editore baten bidez lortu ahalko dugu:



Iturburu-programa

Behin iturburu-programa daukagula, editorea baztertu eta programa konpiladorea martxan jarriko dugu. Konpiladoreari esker iturburu-programa itzuli eta objektu-programa lortzen dugu, objektu-programa ordenadorearen makina-kodean dago kodeturik baina ez da exekutagarria errutinak faltatzen zaizkiolako:



Iturburu-programa Objektu-programa

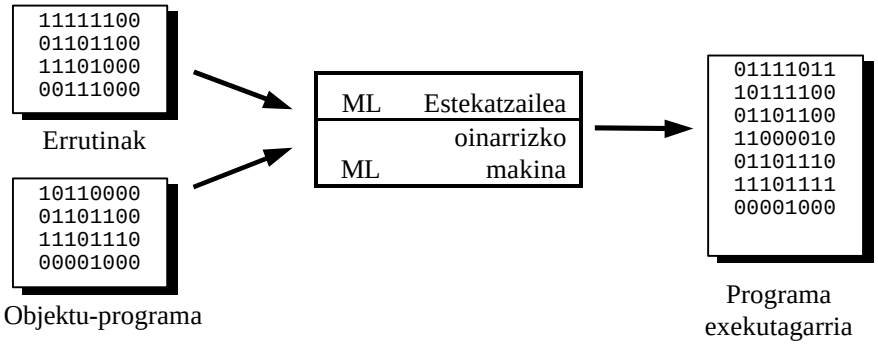
Konpiladore ondoren *linker*[16] programak lan egiten du. Gure objektu-programa eta *programategi* batean metatutako dauden errutinak elkartu eta programa exekutagarria lortzen da estekatzailearen bitartez:

[13] Konpiladore edo interpretatzaile.

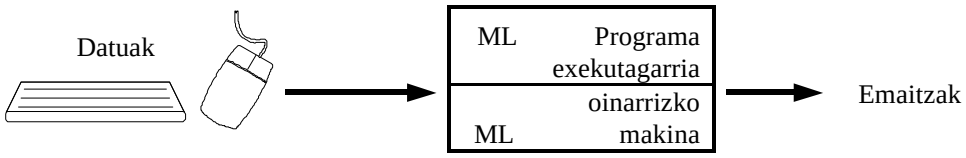
[14] Konpiladore baten ideia hartzeko liburu bat itzultzen den analogia azter dezagun. Itzulpena hasi eta bukatu ondoren liburuaren bi bertsio izaten ditugu, jatorrizko hizkuntzakoa eta helburuko hizkuntzakoa.

[15] Interpretea ulertzeko itzuli beharreko liburuaren adibideak ez digu laguntzen, kongresu eta nazio arteko bileretan unez uneko itzulpen lana da interpretatzaile batek egiten duena.

[16] Linker programari *estekatzaile* esaten zaio, hona hemen bere definizioa: independenteki idatzi eta indenpendenteki konpilatu edo mihiztatutako programa edo moduluak elkarrekin lotu eta programa exekutagarri koherente bat eratzen duen programa.

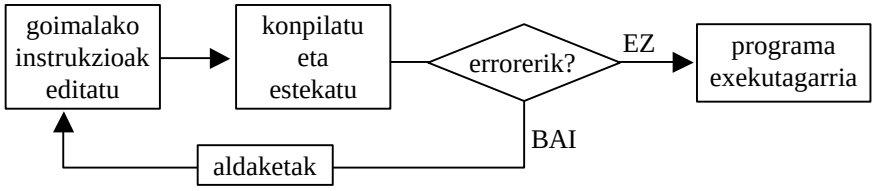


Programa exekutagarria dugularik datuak eman eta emaitzak eskuratuko genituzke:



Ikusi dugun bezala iturburu-programa dugunetik programa exekutagarria izan arte bi urrats ematen dira, lehenengoan konpiladorearen bitartez itzulpena burutzen da eta bigarrenean estekatzaileari esker programa exekutagarria erdiesten da. Gehienetan urrats biak bata bestearen jarraian ematen dira eta bereizketa egitea zaila izaten da.

Programadorearen ikuspegitik, programa exekutagarria lortzea ondoren erakusten den fluxu diagrama egoki betetzea da:

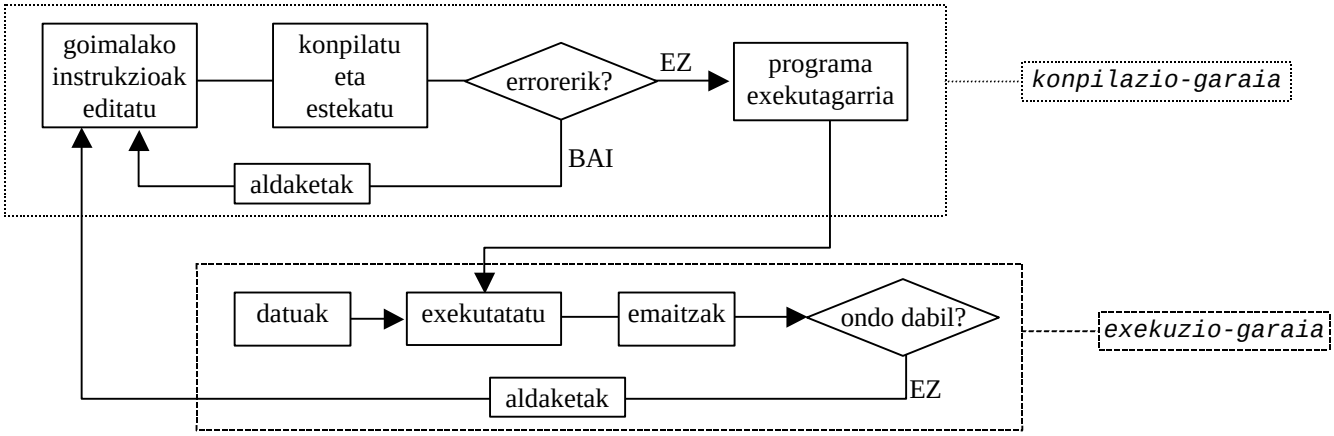


Programatzean beti gertatzen dira hutsegiteak, horregatik goimailako instrukzioak editatu ondoren, iturburu-programa editatu ondoren, erroreak egon daitezke eta ondorioz ezingo litzateke programa exekutagarria eraiki. Konpilazio-garaian detektatzen diren erroreak sailkatzean hiru multzo egingo genituzke:

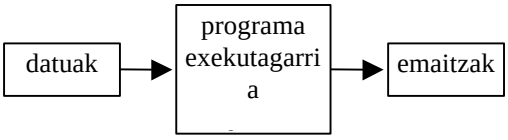
1. *Errore sintaktikoak*, lengoaiaren arabera legez kanpo dagoen zerbaitek eragiten duen errorea. Esate baterako, `FOR` hitz erreserbatua idatzi ordez `F0R` idatzi delako, edo ikurren bat falta izatea edo lekuz kanpo kokatuta egotea.
2. *Mota-erroreak*, lengoaia gehienek datu-motak erabiltzeko definiturik dituzten arauak oso zorrotz kontrolatzen dituzte, eta horiek ez betetzeagatik gertatzen dira aurrerago ikusiko ditugun mota-erroreak. Adibiderako, programa bateko aldagai bat izenak gordetzeko definitu bada ezin izango da aldagai horrekin eragiketa aritmetikorik burutu.
3. *Erazagupen-erroreak*, lengoaia gehienetan aldagai edo bestelako entitate bat erabili aurretik hura deklaratu (erazagutu) behar da. Izen bat gordeko duen aldagaia erazagutuko da berari identifikadore bat emanez, esate baterako `Ikasle`, programaren sententzien batean `Ikasle` idatzi beharrean konturatu gabe `Ikaslea` idatziko balitz konpiladoreak hutsegite hori aurkituko luke. Beraz, erazagupen-erroreak deklaratu gabe dagoen identifikadore bat erabiltzen denean, edo, identifikadore bati erazagupen bikoitza ezartzen zaionean sortuko dira.

Konpilazio-garaian aurkitutako errore guztiak zuzendu arte ezin izango da programa exekutagarria lortu eta iturburu-programaren itzulpena burutu gabe geratzen da, izan ere erroreak egonez gero konpiladoreak programaren esanahia ezin baitu ondorioztatu.

Gerta liteke, lengoaiaren arauak betetzen dituen iturburu-programa izatea eta behar bezala konpilatu eta estekatu ondoren dagokion programa edukitzea, baina programa hori exekutatzean espero zen moduan ez ibiltzea. Adibidez, baliteke programa zeroarekin zatiketak egiten saiatzea, edo, existitzen ez den fitxategi batean irakurketa bat egiten saiatzea. Ondorioa exekuzio-garaian sortutako errorea litzateke, beraz aurreko fluxu diagrama osatuko dugu adieraziz exekuzio garaian sortutako erroreek eragiten dituzten programaren hobekuntzak aintzat hartu direla:



Emaitzak lortzeko prozesua luzeegia dela ematen du, kontutan izan dezagun garatze prozesuan etapa biak (konpilazio-garaia eta exekuzio-garaia) bete behar direla, baina programa exekutagarri fidagarria erdietsi ondoren programa exekutagarri hori nahi den bestetan egikaritzeko aukera izango dugula. Hortaz, programadorearen ikuspegia alde batera utzirik programaren erabiltzailearen eskema honako hau litzateke:



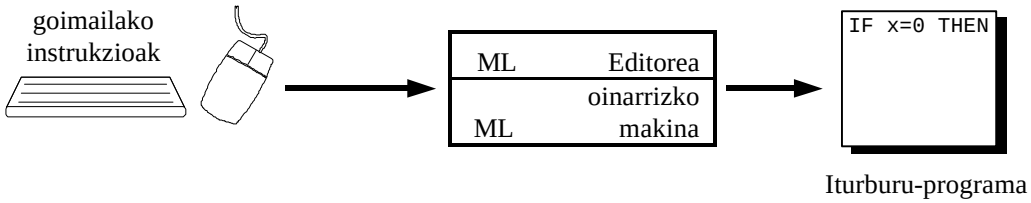
Programa exekutagarria egikaritzeak ez du denbora askorik behar makina-lengoaian dagoelako, eta gainera beste arrazoi garrantzitsu bat dago lengoaia konpilatuek horren arrakasta handia izan dutela justifikatzeko, hots, programaren sortzaileak ez dio programaren erabiltzaileari iturburu-programa salduko bertsio konpilatua baizik, horrela programadorearen jabego intelektuala babesturik geratzen da eta erabiltzailearentzat programa erabiltzea errazago izango da itzultzailerik ez baitu beharko.

1.4.2.2.2 Interpretatzaileak

Interpretatzailea programa itzultzaile bat da honela definitzen dena: *programazio-lengoaia batetik makina-lengoaiarako itzulpena egiten duen programa, baina modu elkarreragilean eta objektu-programarik sortu gabe, interpretatzaile batek iturburu-programa hartu eta aginduz agindu itzuli eta exekutatu egiten du*.

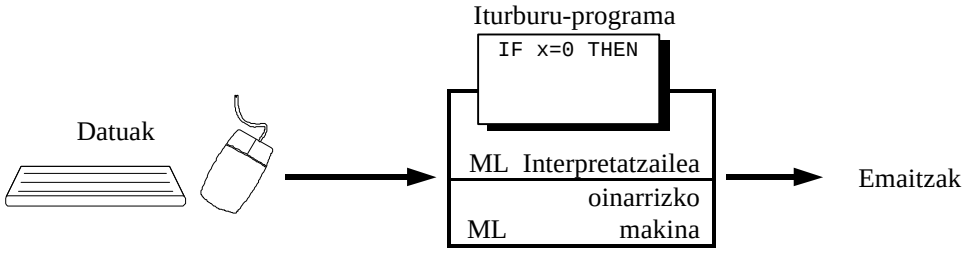
Interpretatzaile batek iturburu-programaren operazio bakoitza interpretatu eta egikaritua izan dadin dagozkion makina-lengoaiaren instrukzioak lortu behar ditu. Beraz,

interpretatzaile batek bere baitan gordetzen ditu goimailako operazioei dagozkien behemailako ordainak. Lengoaia interpretatua denean konpilatua denean bezala lehen urratsa editore baten bitartez iturburu-programa idazten da, baina horren ondoren ez da programa exekutagarririk sortuko:



Iturburu-programa

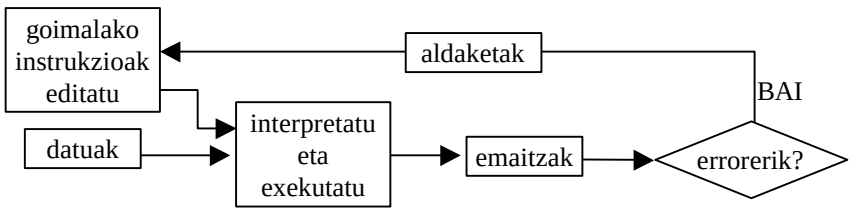
Behin iturburu-programa daukagula, editorea baztertu eta programa interpretatzailea martxan jarriko dugu memorian kargatuz. Interpretatzailearekin batera iturburu-programa ere kargatzen da memorian eta besterik gabe programa exekuta daiteke, horretarako interpretatzaileak iturburu-programaren lehen agindua hartu itzuli (aginduari dagozkion behemailako instrukzioak lortu) eta exekutarazi egiten du, errorerik gertatzen ez bada iturburu-programaren bigarren aginduarekin jarraitzen du interpretatzaileak hura itzuliz eta exekutaraziz. Baldin eta uneko aginduak daturen bat behar izanez gero kanpotik emango zaio exekuzio garaian ohi bezala. Ikus jarraian agertzen irudia, non itzulpena eta egikaritzapena urrats batean ematen dela azaltzen den:



Laburbilduz, interpretatzaileak aginduka lan egiten du (itzuli eta egikariarazi), eta konpiladore/estekatzaile bikoteak berriz programa osoa tratatzen du (iturburu-programa itzuliz programa exekutagarria lortu, ondoren programa exekutagarria egikaritzeko). Oro har esan daiteke interpretatzailearen lan egiteko prozesua errazago dela konpiladorearenarekin alderatzen badugu baina gabezi bi azpimarratuko genituzke, batetik programa bat exekutatu nahi denean iturburu-programa behar dela, eta bestetik, aurrekoa desabantaila handia izateaz gain beste ondorio bat dakar, hots exekuzio garaian iturburu-programa itzuli beharra dagoenez bere exekuzioa askoz geldoagoa izango da.

Lengoaia interpretatuetan itzulpena eta egikaritzapena batera ematen denez erroreak bilatzeko erraza delako (errore bat gertatzen denean itzuli berri eta exekutatzen ari den sententzian dagoela ziurtasuna baitago), askotan erabili dira programazio-lengoaiak ikasteko.

Lengoaia interpretatuetan programadorearen ikuspegia eta programa erabiltzen duenaren ikuspegia bat denez, hauxe litzateke eginkizunen diagrama:



Hurrengo puntuan, informatikaren historia laburrean sortu eta erabili diren goimailako lengoaia garrantzitsuenak aurkeztuko ditugu.

1.4.3 Goi-mailako lengoaia garrantzitsuenak

Goimailako lengoaiak 50ren hamarkadan sortu ziren makina-lengoaien bidezko programazioa saihestu nahirik, ordutik hona ehundaka lengoaia asmatu dira eta hemen garrantzitsuen laburpentxo bat egingo dugu.

1.4.3.1 FORTRAN

Lehen goimailako lengoaia FORTRAN izan zen, IBM-k diseinatu zuen eta 1954. urtean plazaratu zen. FORTRAN akronimoa ***FOR***mula ***TRAN***slation (formulen itzulpena) bi hitzetatik dator eta izenak berak lengoaiaren xedea adierazten du, hots, formula eta espresio matematikoak programatzeko egina baitago FORTRAN lengoaia.

1954. urtetik geroztik zenbakizko kalkulu zientifikoa, nagusiki, FORTRAN lengoaia erabiliz egin zen, esate baterako eguraldiaren iragarpena, aireuntzien diseinua, satelite eta astrountzien ibilbideen kalkuluak eta abar luze bat. Gaur egun ere, zientzilari askok Fortran lengoaiaren bertsio modernoak erabiltzen dituzte.

1.4.3.2 COBOL

Zenbakizko kalkulugintzaren eremua Fortran bitartez ondo beterik baldin bazegoen, ezin gauza bera esan enpresa eta ekonomia arloko eremuari buruz. Horregatik 1960. urtean Estatu Batuko gobernuak bere negozioak kudeatzeko beste lengoaia bat eskatu zuen, eta hortik COBOL lengoaia sortu zen, izenak ere lengoiaren xedea adierazten du COBOL hitzaren esanahia ***CO***mmon ***B***usiness ***O***riented ***L***anguage (enpresara zuzendiriko lengoaia aman-komuna) baita.

Zalantzarik gabe, gaur egun ere, gestio eta enpresa aplikazioetan gehien erabiltzen den lengoaia da Cobol, batez ere ordenadore handietarako aspaldi batean idatzi ziren programei esker mantendu da horren indartsu lengoaia hau.

1.4.3.3 BASIC

Fortran eta Cobol aproposak ziren bakoitza bere aplikazio eremuan, kalkulu zientifikoan Fortran eta merkataritza munduan Cobol. Baina esparru horietatik kanpo ikusi 60. hamarkadan ikusi zen lengoaia zailak zirela biak, Horregatik 60. hamarkadaren erdi aldera BASIC asmatu zen, konputagailu pertsonalekin batera asko hedatu zen lengoaia hau eta esan daiteke xede orokorreko lehenengo lengoaia izan zela, nahiz eta denborarekin beste lengoaia batzuk ordezkatu duten neurri handi batean.

BASIC (***B***eginner´s ***A***ll-purpouse ***S***ymbolic ***I***nstruction ***C***ode[17]) lengoaia John Kemeny eta Thomas Kurtz irakasleek sortu zuten, konputagailuen ikasleak trebatzeko. Izan ere, ordurarte programazioa ikastean ikasle batek bere programa txartel zulatuetan uzten zuen kalkulu-zentruan eta konpiladorearen emaitzak itxaroten geratzen zen, kasurik onenean ordu pare bat itxaron behar izaten zuen. BASIC lengoaiak berriz, interpretatua zelako, programak

[17] Hasiberrientzat xede orokorrerako instrukzio sinbolikoen kodea.

idaztea eta zuzenean frogatzen ahalbidetzen zuen, ondorioz BASIC asko zabaldu zen unibertsitatean eta handik industria eta enpresetara.

Egun erabiltzen den Basic lengoaia interpretatua izan ordez konpilatua da.

1.4.3.4 PASCAL

Hasierako Basic lengoaiaz idatziko programak GoTo sententziaz beterik zeuden, GoTo sententziaren bitartez programaren kontrola pasatzen zen zati batetik beste zati batetara. Programa handiak direnean GoTo sententzia erabiltzeak nahas-pila bat eragiten du eta erroreak bilatzea oso zaila da, horregatik programatzeko paradigma berri bat asmatu behar izan zen: Programazio Egituratua.

Program egitura batek ez du GoTo sententzia erabiltzen programaren fluxua kontrolatzeko, programa modulu edo bloke txikietan banatzen da eta bere fluxu logikoa modulu txikiak piztuz lortzen da. Programa bloketan banatzeko azterketa zehatz bat egin behar da izan daitezkeen moduluak identifikatu ahal izateko, bloke bakoitzaren helburua guztiz definiturik egongo da eta bere sarrerak zein irteerak zeintzuk diren ezagunak izango dira. Programadoreak moduluak konbinatzeko ez du GoTo sententziaren beharrik ondoko hiru kontrol-egiturak baizik:

1. Sekuentzia
2. Hautaketa
3. Errepikapena

1971. urtean Nicklaus Wirth irakasleak programazio egituratuaren abiapuntua izango zen lengoaia asmatu zuen, lengoaia honi PASCAL izena jarri zion (kapitulu honetan lehenago aipatu dugun Blaise Pascal XVII. mendeko matematikariaren omenez). Pascal lengoaia unibertsitatean sortu zen ere, eta bere hasierako helburua programazio egituratua irakastea zen; horretarako, programak garatzean, ikaslea behartuta dago idazkera sistematiko eta ohitura disziplinatuak onartzera. Hala ere, ikasbide lengoaia izateaz gain Pascal lengoaia era guztiko aplikazioak egiteko erabiltzen da, eta horren arrazoia elkarrekiko loturik dauden bi puntu nagusi hauetan oinarritzen da:

Argitasuna. Pascalez idatziriko programak ulertzeko errazak izan ohi dira, programazio-lengoaiak horrela izan dadin behartzen baitu. Ondorioz, programadore batek egindako programa (edo norberak egindakoa denbora luzea igaro eta gero) beste batek irakurtzea eta aditzea posible da. Bestalde, argitasunak beste ondorio garrantzitsu bat dakar, hots, programa garatzen ari garenean gerta daitezkeen erroreak bilatzea erraza dela.

Modularitatea. Pascal lengoaia ondo erabiltzen denean, programa baten baitan dauden eginkizunak modulutan[18] banatzen dira, eta, azpiprograma txikiak direnez euren garapena ez da zaila izaten.

Pascal lengoaia *Programazio Egituratuaren* aintzindaria izan bazen modernoagoa den *Objektuei Orientatutako Programazioa* paradigma onartzeko bertsioak idatzi dira azken urte hauetan. Objektuei Zuzendutako Programazio lengoaia batek datu abstraktuak[19] erabiltzeko ahalmena du, eta honek programak garatzeko metodologia berri bat erakarri du.

[18] Pascal lengoaiak dituen moduluak *funtzio* eta *prozedurak* dira eta seigarren kapituluan ikasi ondoren gainerakotan etengabe erabiliko ditugu.

[19] Datu Abstraktu batek datuak era errutinak software elementu bakar batean integratzen ditu.

1.4.3.5 C

Egun zabalduen dagoen lengoaia da. Bell Labs laborategietan asmatu zuten D. Ritchie eta Ken Thompson-ek UNIX sistema eragilea programatu ahal izateko. C lengoaia ikasteko zaila da eta programadoreek arreta handiz jokatu behar dute, baina C-k dituen ezaugarriegatik bere aplikazio-esparrua oso zabala da. C-ren arrakasta jarraian enumeratzen diren puntuek esplikatzen dute:

Ahalmena. C-ren sententziak gutxi izanik mota guztietako programak egin daitezke, liburutegi asko garatu baita.

Eraginkortasuna. Konpilatu ondoren C-z idatzitako programak oso eraginkorrak gertatzen dira, trinkoak direlako (memori zatia txikia hartuz) eta azkarrak egikaritzen direlako.

Trukagarritasuna. Makina jakin baterako idatzitako iturburu-programa, beste makina batean aldaketarik gabe konpilatu ondoren arazorik gabe exekuta daiteke.

Behemaila. C lengoaian baliabide fisikoetatik hurbil dauden kontzeptuak (erregistroak, bitak, sarrera/irteerako gailuak, ...) erabil daitezkeelako, goimailakoa izan arren behemailako mihiztadura lengoaia ordezka dezake aplikazio askotan. Esan daiteke behe- eta goimailako programazio-lengoaietako ezaugarriak biltzen dituen lengoaia dela .

Programazio Egituratuaren paradigma hobetzen duen *Objektuei Orientatutako Programazioa* sortu da azken urteetan. C lengoaiak bide horretatik jo nahirik C++ deituriko bertsio berriak atera izan dira.

1.4.3.6 ADA

1976an EEBBetako Defentsa-Departamentuaren inkesta baten ondoren 450 lengoaia desberdin erabiltzen ari zirela konturatu ziren, eta txarrena dena hauetako ezein ez zitzaien egokia iruditu Defentsa-Departamentuko softwarea idazteko estandar gisa. Horren ondorioz Defentsa-Departamentuak proiektu handi bat agindu zuen, lengoaiak diseinatzeko ideiarik modernoenak erabiliz goimailako xede orokorreko programazio-lengoaia bat sortzeko.

Lengoaia berria 1979an osatu zen eta 1980-82 artean Jean Ichbiah ikertzaileak zuzenduriko taldeak berrikusi egin zuen. Lengoaia berriaren egileek Pascal aintzat hartu zuen neurri handi batean eta ADA izena[20] jarri zioten. Tamalez esan daiteke ADA-k izan duen hedapena ez da izan hasieran iragarri zitzaion bezain zabala.

1.4.3.7 MODULA-2

Pascal-en beste ondorengo bat da MODULA-2 lengoaia, duen garrantzia mugatua da.

[20] Augusta Ada Byron, Lovaleceko Kondesa (1815-1852) izan zenaren omenez. Augusta Ada Byron lehenago aipatu dugun Charles Babbage-ren laguntzailea izan zen eta onarturik dago munduko lehen programatzailea izan zela.

1.4.3.8 LISP

1957an John McCarthy-k asmatua zenbakizko datuak ez direnak prozesatzeko. Zehatzago LISP lengoaia karaktereak, hitzak eta beste sinbolo batzuk prozesatzeko egokia da, bere izena ***LIS***t ***P***rocessing (zerrenden prozesaketa) laburdura da.

Sinboloen prozesaketa adimenaren ezaugarria delako, Adimen Artifiziala deituriko zientziaren esparruan asko erabiltzen da LISP programazio-lengoaia.

1.4.3.9 PROLOG

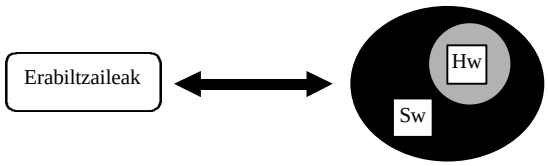
Adimen Artifizialaren esparruko beste lengoaia bat da PROLOG, Marsellako Unibertsitatean sortua ***PRO***graming ***LOG***ic (logikaren programazioa) hitzen laburdura da. PROLOG guk mintzatzen dugun lengoaia naturala ikertzeko asmatu zen, baina gero Adimen Artifizialaren gainerako aplikaziotara zabaldu da ere.

1.4.3.10 LOGO

LOGO lengoaia LISP-en dialektoa da, haurrei zuzendutako programazio-lengoaia da LOGO. Seymour Papert-ek lengoaia bat sortu zuen haurrak programatzen ikas zezaten, oso hitz errazak erabiliz ume batek LOGO bitartez “aginduak” emango dizkio ordenadorearen pantailan agertzen den “dortokari” (kurtsoreari); dortokak egiten duen ibilbidea markaturik gera daiteleelako geometria eta oinarrizko eragiketa matematikoak ikasteko aproposa da.

1.5 KONPUTAZIO SISTEMA BATEN MAILAKETA

Ezaguna denez konputagailu batek bi zati nagusi ditu, batetik hardware[21] hitzaz ezagutzen dena, eta bestetik software[22] hitzaz izendatzen dena. Beraz, konputagailu bat azaltzeko ondoan ematen den eskema erabil daiteke, non hardwarea eta softwarea elkarrekin daudenean erabiltzaileak ordenadoreaz balia daitekeen:

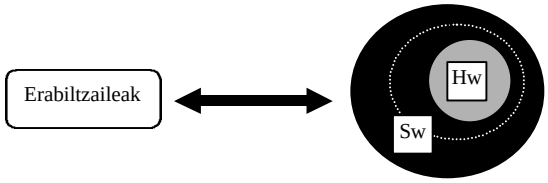


Hardwareari buruzko lehen aipamenak kapitulu honetan egin ditugu eta hirugarren kapituluan gehiago sakonduko dugu, orain horren gainean dagoen bigarren geruza, softwarea alegia, xehetasun gehiagoz aztertuko dugu.

[21] Hardwarea edo euskarri fisikoa honela definitzen da: konputagailu bat osatzen duen elementu fisikoen multzoa (pantaila, teklatua, inprimagailua, zirkuituak, sagua, txipak, etab.).

[22] Softwarea hardwareari kontrajartzen zaio sistema logikoa (programeria) delako, eta honela definitzen da: konputagailu jakin batean erabil daitezkeen programa guztiei erreferentzia egiteko erabiltzen den terminoa.

Softwarea, irudiko geruza beltza, bi mailetan bana daiteke. Hardwaretik hurbilen dagoena (Sistemaren Softwarea) eta erabiltzailetik gertuago dagoena (Aplikazio Programak).

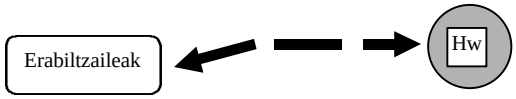


1.5.1 Sistema Eragilea

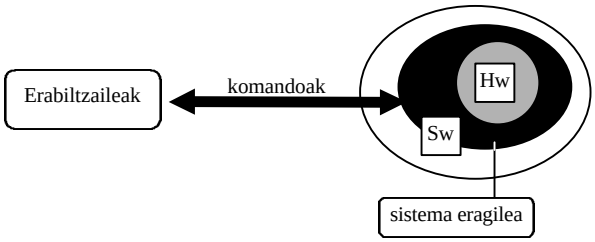
Sistema Eragilea konputagailua erotasunez erabil ahal izateko programa multzoa da, bere azalpena hiru puntutan banaturik landuko dugu: ordenadorearen erabiltzaileak duen ikuspegia, Sistema Eragilearen funtzioak eta Sistema Eragileen motak.

1.5.1.1 Sistema Eragilea eta erabiltzailea

Programarik memorian kargaturik ez duen ordenadorea piztean ez du inolako funtziorik betetzen, gehien jota mezuren bat agertuko du pantaila, mezu horrek Sistema Eragilea falta zaiola adieraziko du, hau da, hardwareaz gain beste zerbait behar duela erabilgarria izateko.



Aurreko irudiak erabiltzailearen eta hardwarearen arteko komunikazio zuzenik ez dagoela adierazten du. *Sistema Eragilea* programa bat da (beraz softwarearen atala) eta horren bitartez ordenadoreari oinarrizko aginduak ematen zaizkio, Sistema Eragileak onartzen dituen aginduei komando deritze. Sistema Eragilea eta erabiltzailearen arteko harremana eskematizatuz:



1.5.1.2 Sistema Eragilearen funtzioak

Aurreko irudiaren arabera, konputagailuaren Sistema Eragilea erabiltzailearekiko komunikazio bidea da, baina Sistema Eragilearen betebeharrak anitzak dira, izan ere makinak eskaintzen dituen zerbitzu guztiak eskuratzea ziurtatzen du. Hala nola:

Periferikoeekiko komunikazioa — Sarrera-irteera Unitate orori (teklatua, sagua, monitorea, inprimagailua, disko unitateak, etab.) bere kontrolagailua dagokio. Dakigunez, kontrolagailua periferikoen funtzionamendua kontrolatzen duen aparatua da, eta periferikoaren kontrola gauzatzeko kontrolagailu bakoitzak driver deituriko

programa bat dauka. Inprimagailu edo beste periferikoren bat lehen aldiz konektatzen denean, gehienetan driver bat ere instalatu beharko dugu ordenadorean (driver hori programa bat da Sistema Eragilean txertatzen, integratzen, dena).

Ondorioz, periferikoa instalatu eta gero Sistema Eragilearen ardura izango da konputagialuaren PUZa eta periferikoen arteko harremana bermatzea (sarrera-irterako eragiketak zuzentazunez bete daitezela).

Konkurrentziaren koordinazioa

Konputagailu batean programa edo lan bat baino egikari daiteke aldiberean, baldin eta Sistema Eragilea lan konkurrenteen arteko koordinazioa aurrera eramateko gai den.

Memoriaren asignazioa

Memoria eta gainerako baliabideen administrazioa Sistema Eragilearen funtziorik zailena izan daiteke. Sistema Eragileak erabakitzen du periferikoen arteko lehentasunak zeintzuk diren eskakizunen ilarak sortuz, suposa daitekeenez baliabideak erabiltzeko lehentasunak erabakitzen dituen algoritmoak konplexuagoak dira Sistema Eragileak prozesu konkurrenteak onartzen baditu.

Fitxategien kudeaketa

Informazioa gorde edo biltegitu behar denean sortzen den lehenengo kontzeptua *fitxategia* da. Fitxategi edo artxibo bat informazio-multzoa da eta, izen bakar baten identifikaziopean, memoria lagungarrian (memoria masiboan) biltegitzen da.

Sistema Eragilearen ardura izango da memoria lagungarrian (zinta edo diskoan) tokia bilatzea fitxategietarako eta informazioaren transferentzia arazorik gabe gerta dadila.

Programen exekuzioen kontrola

Konputagailuak egikaratzen dituen programak memoria lagungarrian aurkitzen dira eta exekutatuak izan daitezen memoria nagusira eraman behar dira. Funtzio hori Sistema Eragilearen scheduler izeneko moduluak betetzen du.

Sistema osoaren abia

Sistema informatiko (hardwarea+softwarea) martxan jartzeko Sistema Eragilearen bitartez egiten da. Txikiak diren zenbait ordenadorek Sistema Eragilea ROM (read-only memory: memoria hila[23]) memorian grabaturik daukate, eta ordenadorea piztean Sistema Eragilea konputagailuaren memoriaran igarotzen da.

Baina ordenadore gehienetan ROM memorian grabaturik dagoena ez da Sistema Eragile osoa, *sistemaren kargatzaile* deitzen den bere zati txiki bat baizik. Horrela, konputagailua piztean Sistema Eragilearen zati hori memoriara igarotzen da eta Sistema Eragile osoa kargatzeko prozesua hasten du.

[23] ROM memorian edo memoria hilan fabrikatzen den momentuan idazten zaio informazioa eta irakurketak bakarrik onartuko ditu. ROM memoria RAM memoriari (memoria biziari) kontrajartzen zaio, honek bertako datuak irakurtzea eta idaztea uzten du.

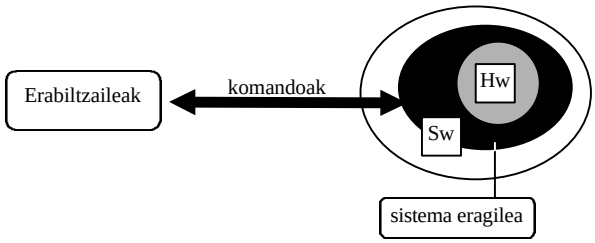
1.5.1.3 Sistema Eragilearen motak

Sistema Eragileak sailkatzeko bi ezaugarri kontutan izango ditugu. Alde batetik erabiltzaile bakarra / erabiltzaile anitzeko sistema izatea, eta bestetik monoataza / multiataza izatea. Hona hemen taula bezala jarririk Sistema Eragileen sailkapena:

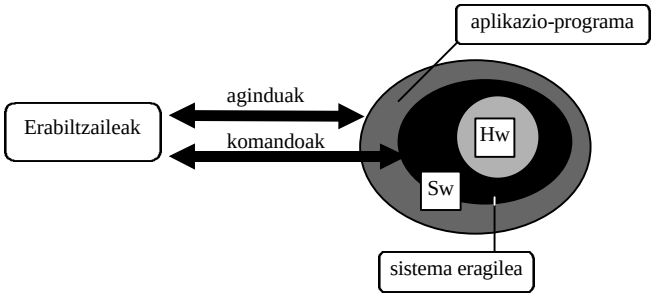
	Monoataza	**Multiataza**
Erabiltzaile bakarra	Une bakoitzeko, erabiltzaile bakar bat onartzen du eta honek lan bakarra burutu ahal du. Adibidez: MS-DOS	Ordenadorearen erabiltzaile bakar batentzat diseinaturik dago, baina honek lan saio bat baino gehiago izan ditzake irekita uneoro. Adibidez: OS/2, Windows
Erabiltzaile anitzekoa	Konputagailua eta bere periferikoak erabiltzaile bat baino gehiagoen artean banaturik dago, eta bakoitzak programa bakarra egikarituko du. Adibidez: PICK	Baliabideak erabiltzaile anitzen artean banaturik, eta bakoitzak lan edo ataza asko irekita izan ditzake. Adibidez: UNIX, XENIX, AIX, Linux, Windows/NT Server,

1.5.2 Aplikazio Programak

Esan den bezala sistema eragile batek erabiltzailearen komandoak onartzen ditu, esate baterako programa bat egikariarazi egiten duen komandoa edo fitxategi baten izena aldatzen duena. Eskematikoki:



Baina sistema eragileak ez du onartzen zenbait lan espezifiko burutzea. Adibidez gutun bat idaztea edo irudi bat marraztea ezinezkoa da sistema eragilearen bitartez. Lan horiek gauzatzeko *aplikazio-programak* daude, aplikazio-programak softwarearen beste maila bat dira eta sistema eragilearen beharra daukate. Erabiltzaileak aplikazio-programarekin komunikatzeko duen era, aplikazio-programaren araberakoa izaten da (aplikazio-programak onar ditzakeen aginduak batzutan teklatuz, beste batzutan saguz ematen dira).



Jarraian aplikazio-programa batzuen deskribapen laburra ematen da, era askotako aplikazio-programak daudelako gutxi batzuk aukeratu izan ditugu, guztien aipamena egin nahi izanez gero zerrenda amai ezina baitzen.

1.5.2.1 Testu-prozesadoreak eta Editoreak

Testu-prozesadore bat bereziki testua prozesatzeko ordenadore-programa litzateke. Testu-prozesadore batek konputagailua idazmakina bihurtzen du, egiatan *testu-prozesadore / ordenadorea* bikotea idazmakina arrunta baino askoz bat ahaltsuagoa da. Hauek lirateke testu-prozesadore batekin lan egitean ematen diren urratsak:

Testua idatzi — Gehienetan idazlana tekleatu egingo da, baina zuzenean ere sar daiteke testu bat ordenadorean; horretarako scanner bat eta OCR (optical character recognition: karaktere-ezagutze optiko) programa bat beharko lirateke.

Testua editatu — Testua idatzi ondoren (edo testua idazten ari garenean) memorian dagoelako aldaketak egitea erraza da, esaterako formatua aukera daiteke letrak eta letra multzoak aldatuz, zuzenketak egin daitezke testu-blokeak tokiz aldatuz, ortografia azter daiteke, irudiak eta grafikoak txertatu, etab..

Testua gorde — Memorian dagoen testua, edozein unetan, memoria masiboan (bigarren mailako memorian edo memoria lagungarrian: diskoa) graba daiteke fitxategi bat sortuz.

Testua inprimatu — Testu-prozesadoreak inprimagailuarekin batera lan egiten duenez idazlana inprimatzeko aukera ematen du.

Testu-prozesadoreek fitxategian letrak gordetzeaz gain kontrolerako karaktere bereziak ere gordetzen dituzte, horien bitartez orrialde amaierak, marjinak, paragrafoak, hizkuntza, etab. markatzen dira. Asko dira merkatuan dauden testu-prozesadoreak, hemen bat aipatzearren Microsoft etxearen *Word* izeneko testu-prozesadorea azpimarratuko genuke, izan ere esku artean duzun idazkia horren bitartez idatzi eta moldatu baita.

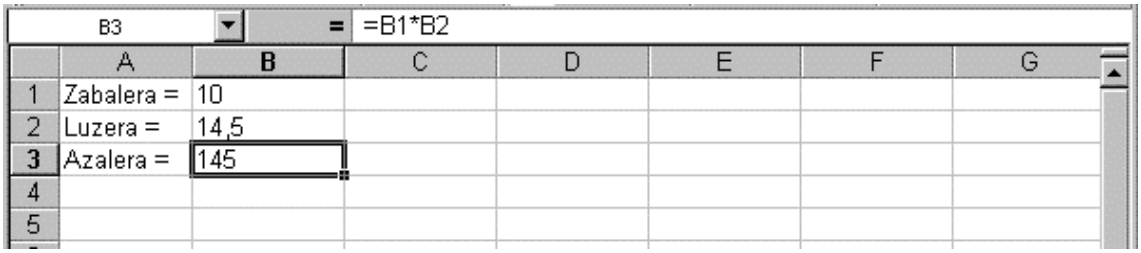
Editoreek testuen fitxategiak sortzen dituzte ere, baina ez dute inolako kontrolerako karaktererik sartzen. Testu hutsa da editore batek fitxategian gordetzen duena, idazki motzak eta programak idazteko erabiltzen dira editoreak. Ezagunenak *DOS* sistema eragileko *EDIT* eta *UNIX* sistema eragileko *vi* lirateke.

1.5.2.2 Kalkulu-orriak

Kalkulu-orriaren kontzeptua eraza da benetan, bi dimentsioko taula bat du bere oinarrian. Taularen zutabeak letraz izendatzen dira eta lerroak beriz zenbakiz desberdintzen dira elkarrekiko, zenbaki/letra bikoteak taularen gelaska bat definitzen du eta bertan informazioa kokatzen da. Kalkulu-orri batek gelasketan onartzen duen informazio mota, nagusiki, hiru motatakoa izango da:

1. Zenbakiak. Zenbakien esanahia aplikazioaren araberakoa izango da, esaterako: soldatak, salmentak, notak, neurri fisikoak, irabaziak, eta abar luze bat. Izan ere, konputatu daitekeen edozer izan daiteke gelasketan gorde ahal izango dena.
2. Etiketak. Kalkulu-orriaren xedea hobeto ulertzeko idazten diren literalak dira, adibidez: “Soldatak”, “Salmentak”, “1. partziala”, “Zabalera”, “Irabaziak”, ...
3. Formulak. Kalkulu-orriak modelatzen duen errealitatea zenbakiz ematen bada, zalantzarik ez dago zenbaki batzuen artean erlazioak daudela. Adibidez, pieza bat fabrikatzeko behar ditugun txaparen neurriak “Zabalera” eta “Luzera” etiketa ondoko gelasketan zenbakiak teklatuz sartzen baditugu, ez dugu “Azalera” ondoko gelaskan

prozesua errepikatuko, hots, azalera adierazten duen zenbakia teklatuz sartu ordez dagokion formula matematikoa idatziko genuke. Ikus Microsoft etxearen Excel kalkulu-orriaren adibidea:



Kalkulu-orri baten bitartez oso erraza da honelako galderen emaitzak ebaluatzea: *Zer gertatuko litzateke baldin eta _____ aldatzen badut?*. Aurreko adibidearekin piezaren zabalera %10an inkrementatzean txaparen azalerak nozitzen duen aldaketa erraz ikus daiteke (zabalerak 10 izatetik 11 izatera pasatuko balitz, azalerako B1*B2 formula bera aplikatuz kalkulu-orriak automatikoki 159,5 emango luke). Bestalde, zenbakien multzoak grafikoki errepresentatzeko baliabideak ematen dituzte kalkulu-orri modernoek.

Merkatuan izan diren eta diren kalkulu-orri ezagunenak hauek lirateke: VisiCalc, LOTUS 1-2-3, Multiplan, Excel, Quattro, etab..

1.5.2.3 Simuladoreak

Ikusi dugunez kalkulu-orri batean dauden zenbakiek munduaren errealitate baten “eredua” osatzen dutela onar daiteke (auto edo beste elementu baten diseinua kalkulu-orri batean daukagunean, errealitate horren hurbilpena egiten ari gara). Zerbaiten eredua estatikoa izan ordez dinamikoa denean simulazioa kontzeptuarekin topo egiten dugu.

Konputagailuen bitarteko simulazioek berebiziko garrantzia dute zientza fisiko, biologiko, sozial eta ekonomikoetan, ingenieritzan ere simulazioak bere tokia hatuta dauka aspalditik. Hauek lirateke simulazioa erabiltzeko bultzatzen duten arrazoi nagusiak:

Ekonomia Auto fabrikatzaile bati merkago gertatzen zaio autoaren zenbakizko eredu digital bat lortzea, eta horren gainean konputagailuen bitarteko frogak egitea altzairuzko benetazko prototipoak egitea baino. Sistemen portaera dinamikoa eta automatikoa ikasten duen zientziaren jakintza arloari Automatika esaten zaio, eta oso arrunta da sistema baten eredua lortu ondoren bere aldaketak ordenadorez simulatzea, adibidez Simulink izeneko programarekin.

Segurantza Pilotuak eta astronautak trebatzeko hasi ziren erabiltzen simuladoreak, egoera seguru eta kontrolatu batean gizonen erantzunak ebaluatu ahal ziren espaziora irten gabe. Sute, uholde, lurrikara, ekaitza eta gainerako natura arrisku, zein gizakiak sortutakoak (istripu nuklear, kutsadura, gosea, gerra, etab.) ikertzeko aproposak dira programa simuladoreak.

Prospektiba Konputagailuen laguntzarik gabe biologoek ez lukete jakin ahal izango laku batean espezie berri bat sartzeak nolako eraginak lerkarkeen, simuladorerik gabe esperimentua egitean gehinetan urteak beharko liratekeelako ekosistemaren aldaketan neurtu ahal izateko. Biologia, ekonomia, soziologia eta ingeniaritzan simuladore baten bitartez izan daitezkeen “etorkizun” desberdinak aurreikus daitezke.

1.5.2.4 Datu-baseak

Datuak gorde eta bereskuratu ahal izateko kalkulu-orriak eta fitxategiak erabil daitezke baldin eta datu gutxi badira. Baina informazioa asko ugaritzen denean bereziki datu-bildumak maneiatzeko programak behar dira, datu-baseak behar dira.

Datu-base bat fitxategi multzo bat da, datu-baseak mota ezberdinekoak izan daitezke: datu-base erlazionalak, datu-base hierarkikoak, datu-base banatuak eta objektuen oinarritutako datu-baseak.

Datu-baseek zertarako balio dute?

Informazioaren biltegitzea 10 edo 20 disko badituzu, ez da beharrezkoa izango datu-base batean erregistratzea, paper batean idatziriko zerrenda aski izango da disko horien gestioa lortzeko. Baina diskoen kopurua 50tik gorakoa bada, ziur izan datu-base konputarizatu batek denbora aurreztuko dizula.

Informazioaren berreskurapena Diskoei buruzko informazioa datu-base batean gordeta badago, eta kontsulta bat egin nahi bada erantzuna berehalakoa izango da. Disko asko ez badira, eta paperean idatzitako zerrenda kantarien arabera egin bada, hura aztertuz ez da denbora askorik behar izango bere abestirik laburrena zein den jakiteko, baina bilaketarako gakoa egokia ez bada (adibidez, zein abestik irauten du 2 minutu 30 segundo?) eskuz eginiko zerrenda goitik behera arakatu beharko da. Datu-base konputarizatuan aldiz informazioa berreskuratzea berehalakoa da.

Informazioaren antolaketa Aurreko adibidearekin jarraituz, eskuz antolatzen den zerrenda batean arreta handiz erabaki behar da zerrendaren sarrera zein izango den. Zerrenda kantarien izenen arabera alfabetikoki antolatu nahi bada, ez da diskoen argitaratze dataren arabera antolaturik egongo. Ondorioz bilaketak eta kontsultak egiteko orduan zerrendaren diseinuzko antolaketak berebiziko garrantzia izango du. Datu-baseetan informazioaren arteko erlazioak zaindu ahal dira eta bilaketak erabat malguak dira.

Informazioaren banaketa Informazio datu-baseetan dagoenean modu desberdinetan bana daiteke. Esate baterako, testu-prozesadore batekin batera lan eginez ehundaka gutun “pertsonalizatu” eta bakoitzari dagokion gutunazalerako etiketak inprima daitezke.

1.5.2.5 CAD-CAM-CAE

Has gaitezen akronimo horien definizioak ematen:

CAD (Computer Aided Design). Konputagailua langunduriko diseinua, hau da produktu industrial bat ekoiztu behar denean planoak marraztu behar dira eta lan hori egiteko programa informatikoa garatu dira. Baina kontutan izan CAD programa batek ez du soilik marrazketarako erraztasunak ematen, produktuaren eredu parametrikoa lortzea ahalbidetzen du eta horrekin batera kalkuluak eta analisiak egiteko tresnak ditu. Adibidez, CAD programa baten bitartez pieza mekaniko bat diseinatzean, planoaren errepresentazio grafikoaz gain piezak jasan ditzakeen tentsioen eta esfortzuen analisia egin daiteke ere.

CAM (Computer Aided Manufacturing). Konputagailua langunduriko fabrikazioa, CAD programa batetik lorturiko datuak CAM programa baten sarrera izaten dira gehienetan, eta honek makina herraminta multzo baten aginduak sortuko ditu diseinatutako pieza ekoiztu ahal izateko.

CAE (Computer Aided Engineering). Konputagailua langunduriko ingenierutza, CAD eta CAM kontzeptuak biltzeari CAE esaten zaio.

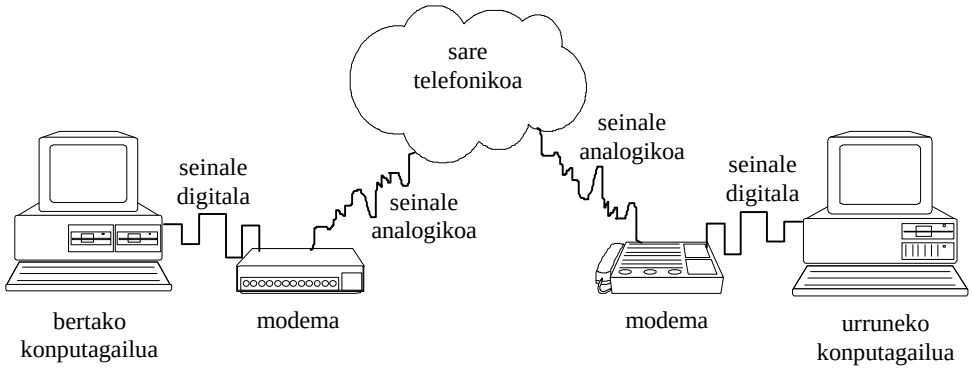
Amaitzeko CAD/CAM kontzeptua esaldi batean laburtzearren zera esan dezakegu: ideiak produktu bihurtzea.

1.5.2.6 Telekomunikazioak

Konputagailuek zenbait periferikoekin konektatu ahal izateko ataka edo portu izeneko konexioak dituzte. Ataka bat, definizioz, aparatu informatiko bateko konexioa da, kable baten bidez beste aparatu batekin konektatzeko bidea ematen duena. Adibidez ordenadore eta inprimagailuaren arteko harremana ataka paraleloaren bitartez gauzatzen da gehinetan.

Baina konputagailuen arteko komunikazioa lortzeko atakaz gain konputagailu biek komunikazioarko txartel bana eta komunikaziorako softwarea beharko dituzte. Konputagailuak hurbil daudenean euren artean kableak tartekatzea ez da arazoa izaten, baina konputagailuen arteko distantzia kilometrotan neurtzen denean ezinezkoa da puntutik punturako konexioa gauzatu, horregatik ordenadoreen arteko puntutik punturako konexioak egiten hasi ziren une beretik ingeniariak telefono hariari begirada maltzurrak botatzen zizkioten. Azken finean, telefono hariak mundu osora hedaturik dagoen sarea da, eta ahotsa bidaiatzen duen antzera bitak ere bidaia zezaketen.

Urruneko ordenadoreak sare telefonikoz konektatzeko zailtasuna, teknika aldetik, seinale motan datza. Izan ere, ahotsa kable telefonikoz transmititzen denean onda jarraia da, hots, seinale analogikoa. Baina ordenadorean dagoen informazioa digitala izatean, telefonia sarean sartu aurretik transformatu (modulatu) egin behar da eta sorburutik helbururaino bidaia egin ondoren, berriro ere transformatu beharko da (seinale analogikotik seinale digitalera igarotzeari demodulatu esaten zaio). Modulazio/demodulazio lanak betetzen dituen aparatua *modem* deitzen da eta honelaxe definitzen da: transmisio-linea baten sarrerako eta irteerako tresna da modema, sarrerako funtzioa betetzen duenean seinalea modulatu egiten du, eta irteerako funtzioa betetzean seinalea demodulatu egiten du.



Ordenadore hurbilen arteko puntutik punturako konexioa dela edo urruneko konexioa dela, esan dugun eran konputagailuek nolabaiteko komunikazio-softwarea behar izaten dute. Askotan komunikazioa softwarea Sistema Eragilean integratzen den zerbitzu-programak dira.

1.6 PROGRAMAK

Hona hemen 1. kapituluaren programak orrialdeen arabera sailkaturik:

Izena	*Programaren identifikadorea*	*ORRI.*	*Ikasgaia*
EUKLIDES.PAS	Euklides	1-07	Algoritmoak
ASMATU.PAS	ZenbakiaAsmatzen	1-11	Algoritmotik programara
DIF_FIN1.PAS	HasieraketaTaula3O	1-17	Algoritmoak
DIF_FIN2.PAS	DiferentziaFinituak3O	1-18	Algoritmoak
DIF_FIN3.PAS	HasieraketaTaula3E	1-17	Algoritmoak
DIF_FIN4.PAS	DiferentziaFinituak3E	1-18	Algoritmoak

1.7 BIBLIOGRAFIA

- Alcalde, E., García, M., Peñuelas, S., *Informática Básica*, McGraw-Hill, Madrid, 1988
- Beekman, G., *Computación & Informática Hoy*, Addison-Wesley Iberoamericana, Wilmington, 1995
- Brookshear, J.G., *Introducción a las Ciencias de la Computación*, Addison-Wesley Iberoamericana, Wilmington, 1995
- Penrose, R., *The Emperor's New Mind*, Oxford University Press, 1989
- Tanenbaum, A., *Konputagailuen Antolaketa Egituratua*, EHU/UPV-ko Argitarapen Zerbitzua, 1994

ERANSKINAK

E1 Abakoa erabiltzeko arauak

E2 Adimena duten makinak

E3 Telekomunikazioen iraultza

E1 Abakoa erabiltzeko arauak

E2 Adimena duten makinak

E3 Telekomunikazioen iraultza

2. ATALA: INFORMAZIOA ETA BERE ADIERAZPIDEA

AURKIBIDEA

2.1 SARRERA

Lehenengo kapituluan esan dugunez, arrazoi teknikoak direla eta, ordenadoreek daukaten funtzionamendua seinale elektriko digitaletan oinarritzen da. Hots, makinen barneko zirkuituek seinale digitalak (tentsio elektrikoa izatea ala tentsiorik ez izatea) lantzen dituztenez, ordenadoreei eman beharko zaien informazioa sistema bitarrean kodifikaturik egotea nahitaezkoa da.

Zirkuitu elektronikoek erabiltzen duten logika, diseinuz, aukeragarri eta arbitrarioa izaten da. Horrela, zirkuituaren punturen batean masarekiko tentsioa izatearen egoera 1 batez adierazten denean (tentsiorik ez izatea 0 batez adieraziko litzateke) logika positiboa erabiltzen dela esaten da. Alderantzizko kasuan (zirkuituan tentsioa izateari 0 esleitzen zaionean, eta ez izateari 1 esleitzen zaionean) ordenadoreak logika negatiboa darabilela esaten da. Edozein kasutan erabiltzaileak kanpotik begiraturik, ordenadoreak zer logika darabilen ez dauka jakiterik eta orokorrean hitz eginez horren ardurarik ez luke izan beharko.

2.2 INFORMAZIOA NEURTZEKO UNITATEAK

Informazioa neurtzeko dauden unitateaz hitz egin aurretik, **informazioa** hitzak duen esanahia zehaztu beharko genuke. Bizitza arruntan informazio hitzaren esanahia *aprobetxagarria delakoan komunikatzen den zerbait* izango litzateke, baina informatika eta konputagailuen munduan komunikatzen den oro informazioa kontsidera daiteke. Beraz, orri honetan agertzen diren letrak eta irudiak informazioa lirateke, eta esaldiren bat azpimarratuko balitz edo iradokizunen bat gehituko balitz informazio gehigarriaz osatuko litzateke.

Ordenadoreen munduan informazioa digitala da (unitate diskretuz osaturik dator) eta pusketan zati daiteke. Ezezaguna edo zaila zaigun hitza silabaka ahoskatzen dugun bezala, ordenadoreak ere zenbakiak, alfabetoko letrak, irudiak, soinuak edo beste mota bateko informazioak unitate txikiagotan zatitzen ditu. Izan ere, ordenadoreak informazioa ulertzeko eta lantzeko bit elementaletan banaturik eman behar zaio.

Bit (***b**inary dig**it***, digitu bitarra) informazio unitaterik txikiena izango litzateke eta balio bi onar ditzakeenez, informazio desberdin bi gorde ditzake: 1 ala 0, bai ala ez, zulatua ala zulatu gabe, beltza ala zuria, tentsio elektrikorik ala tentsioaren eza, ...

Bit batean bi informazio desberdin sar daitezkeela onarturik, zenbat informazio desberdin adieraz daiteke 2 biten bitartez?. Bit bakoitzaren egoera biak kontutan izanik 00 01 10 eta 11 konbinazioak lortzen direnez lau dira adieraz daitezkeen informazioak. Orokorrean *n* bit bitartez *2* ber *n* informazio desberdin adierazi ahal izango da.

Bit bat unitate txikiegia gertatzen denez, byte izeneko unitatea gehiago erabiltzen da. Bytea karaktere bat adierazteko bit-multzoa da, eta zortzi bitez osatzen denez zortzikote deitzen zaio ere. Ondokoak dira gehien erabiltzen diren byte unitatearen anizkoitzak:

Anizkoitzak	Balioa		Ikurra
Kilobyte	2^{10}=1.024	$\cong 10^3$	KB
Megabyte	2^{20}=1.048.576	$\cong 10^6$	MB
Gigabyte	2^{30}=1.073.741.824	$\cong 10^9$	GB

Honezkero, gizakiok dugun informazioa edozein ordenadoretan gordetzean kode bitarrera igaro beharko dela onartu beharra dago, zeregin horri informazioaren kodeketa deritzo. Gure informazioa, laburbilduz, bi eratakoa izan daiteke sinboloak batetik eta kopuruen balioak bestetik (hitzak eta zenbatekoak).

2.3 SINBOLOEN ADIERAZPIDEA: KONPUTAGAILU-KODEAK

Gure bizitzan agertzen zaizkigun kontzeptuak hitzez ordezkatzen ditugu, eta hizkuntza mintzatutik hizkuntza idatzira jauzia egiteko, alfabetoko letrak eta irakurketa erraztea helburua duten puntuazio-zeinuak erabiltzen dira.

Ordenadoreetan informazio testuala gordetzeko biderik zuzenena sinbolo bakoitzari kode bitarrean dagokion ordezkoa asmatzea litzateke, horrela konputagailu-kodeak deituriko zenbait kode agertu izan da, adibidez BCD, EBCDIC eta abar luze bat. Gaur egun gehien erabiltzen den konputagailu-kodea ASCII (*American Standard Code for Information Interchange*: Informazioaren Trukaketarako Kode Estandar Amerikarra) da.

2.3.1 ASCII kodea

Kode honen bitartez kodetu behar diren letrak, digituak eta puntuazio-zeinu zein karaktere bereziak 7 biteko txantiloi batez egiten da, eta zortzigarren bitak besteak kontuan izanik balio jakin bat hartuko du. Txantiloia zazpi biteko katea denez, (eta bit bakoitzak onar ditzakeen balioak 0 eta 1 izan daitezkeenez) guztira 2^7=128 karaktere desberdinen kodetzea lor daiteke.

Hala ere, eta karaktere gehiago adierazteko ahal izateko hasierako ASCII kodearen beste bertsio bat atera izan da[1], zeinek zortzi bit erabiltzen dituen. Beraz ASCII horren bitartez 2^8=256 karaktere desberdin kodetu ahal izango da.

Adibidez, ***Zure izena?*** esaldia ASCII kodean adierazita ondoko bit zerrenda izango litzateke:

1011010	1110101	1110010	1100101	1000000	1101001	1111010	1100101	1101110	1100001	0111111
Z	***u***	***r***	***e***		***i***	***z***	***e***	***n***	***a***	***?***

Zortzi biteko ASCII taularen hasiera eta amaiera honelakoa da:

ASCII-8				
Hamar.	**8tar**	**16tar**	**Bitar**	**Ikurra**
0	000	00	0000 0000	
1	001	01	0000 0001	☺
2	002	02	0000 0010	●
3	003	03	0000 0011	♥
4	004	04	0000 0100	♦
5	005	05	0000 0101	♣
6	006	06	0000 0110	♠
7	007	07	0000 0111	•
8	010	08	0000 1000	□
9	011	09	0000 1001	○
10	012	0A	0000 1010	■
251	373	FB	1111 1011	√
252	374	FC	1111 1100	n
253	375	FD	1111 1101	2
254	376	FE	1111 1110	▮
255	377	FF	1111 1111	

[1] ASCII kode biren bereizketa erdiesteko jatorrizkoari ASCII-7 esaten zaio eta besteari berriz ASCII-8.

2.3.2 BCD kodea

ASCII-ren akronimoak dionez norma batetik ondoriotzen den kodea da, baina horren aurretik IBM enpresak BCD (*Binary Coded Decimal*, Hamartarra Bitarrean Kodeturik) izeneko kodea ezarri zuen bere makinetan. BCD kodea laburregia izan zen sinboloak kodetzeko 6 biteko txantiloiaren bitartez egiten zelako, beraz ordenadoreak gehienez 2^6=64 karaktere desberdin ezagutzen zituen.

2.3.3 EBCDIC kodea

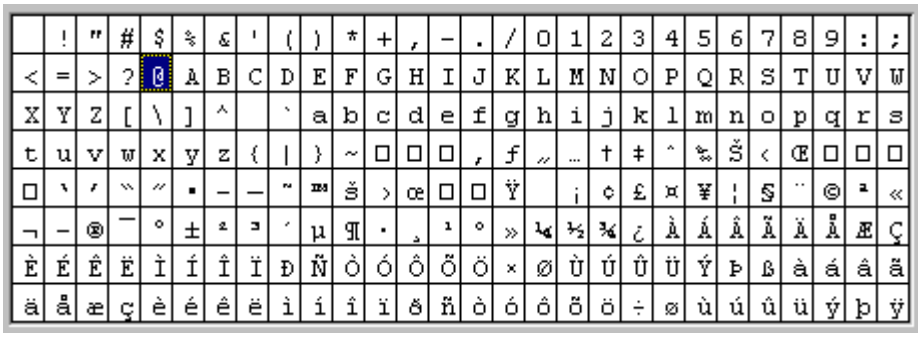
BCD kodeak zuen muga gainditu nahirik IBM-k kode horren hedapena plazaratu zuen EBCDIC izenez ezagutzen dena (*Extended EBC Interchange Code*, Trukaketarako EBC Kode Hedatua). EBCDIC kodea biten zortzikotez baliatzen denez, guztira onar daitezkeen karaktere desberdinak 256 dira.

2.3.4 OEM kodea

DOS sistema eragileak ASCII kode batekin lan egiten du OEM (Original Equipment Manufacturer) deitzen dena. OEM kodean 0tik hasita 127 bitarteko bit konbinazioak ASCII kodearen berdinak dira, baina 128tik 255 bitartekoak bereziak dira unez une kargatu nahi den hizkuntzaren arabera.

2.3.5 ANSI kodea

Estandar bat da eta Windows sistema eragilearen oinarrizko kodea da. Windowsek onartzen du ANSI kode batetik bestera aldatu ahal izatea, ANSI kodeak aukeratzeko letra-tipoa hautatuz egiten da. Adibidez, jarraian erakusten dena ANSI-Courier karaktere-multzoa da:



2.3.6 UNICODE kodea

Ekialde urruneko kulturetan milako karaktere ezberdin behar izaten dira; hori dela eta, bit zortzikotea laburregi geratzen da eta 16 biteko kodifikaziora jo izan da UNICODE bitartez. UNICODE deituriko konputagailu-kodeak 2 ber 16 (65536) karaktere adieraz ditzake, non

hasierako 7 bitek dituzten moldeak ASCII kodearekin alderatuz esanahi bera duten. Ezaguna den Windows NT sistema eragileak UNICODE kodea darabil.

2.4 KOPURUEN ADIERAZPIDEA

Karaktereak kodetuz informazioa gordetzea posiblea izanik, erabat inefizientea suertatzen da informazioak datu numerikoak direnean. Esate baterako, 28 zenbakia EBCDIC kodearen bitartez adierazten bada, ondoko hamasei bitak beharko lirateke 11110010 11111000, eta alderantziz, hamasei biten bitartez adieraz daitekeen zenbakirik handiena 99 litzateke.

Karaktereak ez bezala kopuruen adierazpidea, zenbakien balioak aintzat harturik egiten da, horretarako kopuruaren balioa oinarri bitarrean jarri beharko da. Ikus ditzagun 2 oinarriko zenbaketa sistemaren xehetasunak. Horretarako kopuruak nola zenbatzen ditugun, adibide batez, gogoratu dugu.

Automobil baten kilometro kontagailua biragai diren disko desberdinaz osaturik dago, eta disko bakoitzak zerotik bederatziraino dauden hamar digituak marrazturik dauzka. Automobila berria denean, disko guztiek 0 digitua agerian dute bakoitzari dagokion leihotxotik:

0	0	0	0	0	0	0

Automobila erabiltzen den neurrian diskoak biratzen hasten dira, higitzen den lehenengoa eskuinekoa da eta bederatzigarren kilometroan dagoenean honelako zerbait ikusi ahal izango da:

0	0	0	0	0	0	9

Eskuineko gurpiltxoak biratzen jarraituz gero, hurrengo diskoa mugiarazi egiten du hamargarren kilometroan gaudela adieraziz:

0	0	0	0	0	1	0

Automobilak bidaian jarraitzen badu eskuineko gurpiltxoak, lehen bezala, biratu egingo du 0-tik 9-raino. Orduan, hurrengo kilometroa hasiko denean bigarren diskoa 1-tik 2-ra automatikoki higituko da, honelako aldaketa gertatuz:

0	0	0	0	0	1	9

$\rightarrow$

0	0	0	0	2	0

Hau izango litzateke sistema hamartarrean zenbatzeko darabilgun teknika. Sistema bitarrean era berdinean egiten da baina onargarriak diren digituak bi dira 0 eta 1. Kilometroak egin ahala honelako mezuak agertuko lirateke:

0	0	0	0	0	0	0	0 (berria izatean)
0	0	0	0	0	0	1	1. kilometroan
0	0	0	0	0	1	0	2. kilometroan
0	0	0	0	0	1	1	3. kilometroan
0	0	0	0	1	0	0	4. kilometroan
0	0	0	0	1	0	1	5. kilometroan
0	0	0	0	1	1	0	6. kilometroan
0	0	0	0	1	1	1	7. kilometroan

0-tik 7 bitartera sistema bitarrean zenbatzean aurreko taulan ematen diren sinboloak agertzen dira. Baina gure hasierako arazoa kopuruak modu egoki batean ordenadorean

gordetzea zen, konputagailu-kodeak erabili beharrean kopuruen adierazpidea notazio edo idazkera bitarrean oinarrituko dugu.

Notazio bitarra erabiliz byte baten bitartez 0 eta 255 arteko zenbaki osoak adieraz daitezke (00000000 eta 11111111 arteko konbinazioak). Eta bi byten bidez adieraz daitezkeen zenbaki osoak 0-tik 65.535 bitartera, beraz bi byten horien bitartez EBCDIC edo ASCII-8 kodeak erabiliz lor daitekeen kopururik handiena 99 zela gogoratuz idazkera bitarraren komenigarritasuna nabarmentzen da.

Horregatik kopuruak makinetan gordetzeko idazkera bitarraren bat erabiliko da, idazkera bitarraren bat diogu azaldutako sistema bitarra kopuruak adierazteko teknika desberdinen oinarria delako, jarraian teknika horieratiko hauek ikusiko ditugu: modulua eta zeinua, 1rako osagarria, 2rako osagarria, bitarra gainditua eta koma higikorreko formatu bitarra.

Baina aurrena zenbaketa sistemen arteko erlazioak azter ditzagun.

2.4.1 Zenbaketaren Oinarrizko Teorema

Teorema honek edozein oinarritan emaniko kopurua sistema hamartarrarekin erlazionatzen du. Sistema hamartarraren sinboloak (edo digituak) 0, 1, 2, 3, 4, 5, 6, 7, 8 eta 9 dira eta ***K*** kopuru jakin bat polinomio batez adieraz daiteke:

$$K = \sum_{i=-s}^{z} (digitu)_i \times (oinarri)^i \qquad (1)$$
$$0 \le digitu < oinarri$$

Non:

- *oinarri* = 10
- *i* = komarekiko posizioa
- *s* = komatik eskuinera dauden digituen kopurua
- z = komatik ezkerrera dauden digituen kopurua - 1
- *digitu* = zenbakia osatzen duten sinboloak

Era trinkoan emaniko (1) formula garatuz honelako polinomioa lortzen da:

$$...+X_3*10^3 + X_2*10^2 + X_1*10^1 + X_0*10^0 + X_{-1}*10^{-1} + X_{-2}*10^{-2}... \qquad (2)$$

Adibidez, 1997 eta 34,25 kopuruak adierazteko:

$$1997 = 1*10^3 + 9*10^2 + 9*10^1 + 7*10^0$$
$$34,25 = 3*10^1 + 4*10^0 + 2*10^{-1} + 5*10^{-2}$$

2.4.1.1 n oinarritik hamartarrerako eta hamartarretik n oinarrirako bihurketak

Aurreko puntuan esandakoa kontutan izanik, zenbaki n-tar bat 10 oinarrirako bihurketa lortzeko polinomioa aplikatzea aski da. Horrela:

$$X_p ... X_2 X_1 X_0 X_{-1} X_{-2} ... X_{-q} \lfloor n$$

Hamartar bihurtzeko ondoko eragiketak burutuko lirateke:

$$X_p * n^p + ... + X_2 * n^2 + X_1 * n^1 + X_0 * n^0 + X_{-1} * n^{-1} + X_{-2} * n^{-2} + ... + X_{-q} * n^{-q} \lfloor 10$$

Adibidez:

$234{,}301 \lfloor 5 = ???{,}??? \lfloor 10$

$234{,}301 \lfloor 5 = 2x5^2 + 3x5^1 + 4x5^0 + 3x5^{-1} + 0x5^{-2} + 1x5^{-3} \lfloor 10$

$= 50 + 15 + 4 + 0{,}6 + 0 + 0{,}008 = 69{,}608 \lfloor 10$

$234{,}301 \lfloor 5 = 69{,}608 \lfloor 10$

Demagun orain, $X_1 X_2 X_3 ... X_m$ zenbaki hamartar bat daukagula eta n-tar bihurtu nahi dugula. Zatiketak egingo dira zatidura 0 izan arte, eta zenbaki n-tarra osatzeko hondarrak bukaeratik hasierarantz hartuko dira:

$$\begin{array}{l} X_1 X_2 X_3 ... X_m \mid n \\ \quad h_1 \quad z_1 \mid n \\ \quad\quad h_2 \quad z_2 \mid n \\ \quad\quad\quad ... \\ \quad\quad\quad\quad z_{k-1} \mid n \\ \quad\quad\quad\quad\quad h_k \end{array}$$

$$X_1 X_2 X_3 ... X_m \lfloor 10 = h_k h_{k-1} ... h_2 h_1 \lfloor n$$

Zenbaki hamartarra osoa izan beharrean erreala bada, bere bihurketa n idazkera jakin batera bi urratsetan egiten da. Lehenik arestian azaldu dugun zatiketaren metodoaren bitartez alde osoa bihurtzen da, eta alde zatikiarrari dagokion bihurketa biderkaketaz lortzen da.

Demagun beraz, $0, X_1 X_2 X_3 ... X_m$ zatikia sistema hamartarrean emanik datorrela eta n-tar bihurtu nahi dugula. Helburua den n oinarria aintzat harturik biderkaketak egingo dira eta alde osoak antolatuz bihurketaren emaitza eskuratuko da. Hona hemen eraiki beharreko taula:

$0, X_1 X_2 X_3 ... X_m \times n = b_1 , Y_1 Y_2 Y_3 ... Y_n$

$0 , Y_1 Y_2 Y_3 ... Y_n \times n = b_2 , Z_1 Z_2 Z_3 ... Z_{\tilde{n}}$

$0 , Z_1 Z_2 Z_3 ... Z_{\tilde{n}} \times n = b_3 , ...$

$$0, X_1 X_2 X_3 ... X_m \lfloor 10 = 0, b_1 b_2 b_3 ... \lfloor n$$

Adibidez:

$69{,}608_{\lfloor 10} = ???{,}???_{\lfloor 5}$

$69{,}608_{\lfloor 10} = ???_{\lfloor 5} + 0{,}???_{\lfloor 5}$ (bihurketa bi urratsetan)

	zatidura	hondarra	
69 : 5 =	13	**4**	0,608 x 5 = **3**,04
13 : 5 =	2	**3** ↑	0,04 x 5 = ⌈ **0**,2
2 : 5 =	0	**2** ⌋	0,2 x 5 = ↓ **1**,0

$69{,}608_{\lfloor 10} = 234_{\lfloor 5} + 0{,}301_{\lfloor 5} = 234{,}301_{\lfloor 5}$

2.4.1.2 n oinarritik m oinarrirako bihurketa

Zenbaki n-tar bat m-tar bihurtzeko sistema hamartarraren bitartez egiten da. Zenbaki n-tarra hamartar bihurtzen da dagokion polinomioa aplikatuz, eta sortutako zenbaki hamartarra zatiketa-biderkaketa metodoaren bidez m-tar bihurtzen da.

$$X_{\lfloor n} \xrightarrow{polinomioa} Y_{\lfloor 10} \xrightarrow{zatiketa-biderkaketa} Z_{\lfloor m}$$

2.4.1.3 Bitar-hamartar eta hamartar-bitar bihurketak

Bihurketa hauek aurrerago ikusi ditugun bihurketen kasu partikularrak dira. Horregatik metodo polinomialan zein zatiketa-biderkaketa metodoan n jarri beharrean 2 jarri beharko da, zenbaki bitarretan ager daitezkeen digituak (zifrak edo sinboloak) 0 eta 1 baitira.

Bitar-hamartar bihurketaren adibidea:

$101{,}11_{\lfloor 2} = ???{,}???_{\lfloor 10}$

$101{,}11_{\lfloor 2} = 1x2^2 + 0x2^1 + 1x2^0 + 1x2^{-1} + 1x2^{-2}{}_{\lfloor 10}$

$= 4 + 0 + 1 + 0{,}5 + 0{,}25 = 5{,}75_{\lfloor 10}$

$101{,}11_{\lfloor 2} = 5{,}75_{\lfloor 10}$

Hamartar-bitar bihurketaren adibidea:

$14{,}625_{\lfloor 10} = ???{,}???_{\lfloor 2}$

$14{,}625_{\lfloor 10} = ???_{\lfloor 2} + 0{,}???_{\lfloor 2}$ (bihurketa bi urratsetan)

	zatidura	hondarra		
14 : 2 =	7	**0**	0,625 x 2 =	**1**,25
7 : 2 =	3	**1**	0,25 x 2 = ⌈	**0**,5
3 : 2 =	1	**1** ↑	0,5 x 2 = ↓	**1**,0
1 : 2 =	0	**1** ⌋	0,0 x 2 =	**0**,0

$14{,}608_{\lfloor 10} = 1110_{\lfloor 2} + 0{,}1010_{\lfloor 2} = 1110{,}101_{\lfloor 2}$

2.4.1.4 Bitar-zortzitar eta zortzitar-bitar bihurketak

Kopuruak sistema bitarrean adierazten direnean ateratzen diren zenbaki bitarrak oso luzeak dira, horregatik bitak paperean idazteko dagokien sistema bitarra erabili beharrean sistema zortzitarrean edo hamaseitarrean adierazten dira.

Hurrengo taulan sistema zortzitarreko sinboloak eta dagozkien bitarraren bihurketak ematen dira:

Zortzitar	Bitar
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Bitar-zortzitar bihurketak lortzeko biten hirukoteak sortzen dira, eta taulari so eginez hirukote bakoitzaren zortzitar ordezkoa aurkitzea nahikoa da. Adibidez:

$11100111{,}0010101_{\lfloor 2} = ???{,}???_{\lfloor 8}$

*0*11 100 111 , 001 010 1*00* (hirukoteak sortu)

$3_{\lfloor 8}\ 4_{\lfloor 8}\ 7_{\lfloor 8}\ ,\ 1_{\lfloor 8}\ 2_{\lfloor 8}\ 4_{\lfloor 8}$

$11100111{,}0010101_{\lfloor 2} = 347{,}124_{\lfloor 8}$

Alderantzizko bidea berdin egiten da, sinbolo zortzitar bakoitzari taularen araberako dagokion bitarra bilatu eta kopuru osoa konposatu egingo da. Hona hemen adibide bat:

$103{,}742_{\lfloor 8} = ???{,}???_{\lfloor 2}$

$001_{\lfloor 2}\ 000_{\lfloor 2}\ 011_{\lfloor 2}\ ,\ 111_{\lfloor 2}\ 100_{\lfloor 2}\ 010_{\lfloor 2}$

$103{,}742_{\lfloor 8} = 1000011{,}11110001_{\lfloor 2}$

2.4.1.5 Bitar-hamaseitar eta hamaseitar-bitar bihurketak

Hiru digitu bitar bilduta sinbolo zortzitar bat sortzen den bezala (2^3=8), lau digitu bitar bilduta sinbolo hamaseitar bat lortuko da, izan ere ondoko berdinketatik ondoriotzen da esandako hori: 2^4=16.

Baina hamaseitar-bitar baliokideen taula ikusi aurretik, gauza bat argitu beharra dago. Sistema hamaseitarraren sinboloak aukeratzerakoan honako hauek hautatu eta erabili egiten dira:

0 1 2 3 4 5 6 7 8 9 A B C D E F

Beraz, zenbaki hamaseitar batean zifraz gain zenbait letra ere ager daiteke. Hona hemen taula:

Hamaseitar	Bitar
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Taula hau bitar-zortzitarraren jarraipena litzateke eta bere eraiketarako, Zenbaketaren Oinarrizko Teoremaren bitartez polinomioa aplikatzea aski da, adibidez D sinboloari dagokion balio absolutua 13 izanik bere balio bitarretik emaitza berdina lor daitekeela froga dezagun:

$D \rightarrow 13x16^0 = 13$

$1101 \rightarrow 1x2^3 + 1x2^2 + 0x2^1 + 1x2^0 = 13$

Sinbolo hamaseitar bakoitzaren bere baliokide bitarra erdiesteko, **2.4.1.2** puntuan azaldu bezala idazkera hamartarretik igaro beharko litzateke baina ez da derrigorrezkoa[2] eta metodo zuzena egokiagoa eta bizkorragoa da.

Bitar-hamaseitar bihurketak burutzeko, bitar-zortzitar kasuaren hirukoteak sortu beharrean laukoteak egingo dira. Metodoa berdina izango da eta hiru urrats hauek eman behar dira:

1. Koma abiapuntuz harturik, zenbaki bitarraren alde osoaren zifrak eskuinetatik ezkerretara launaka bilduz taldetxoak egiten dira. Ezkerretako taldetxoa zifraren baten faltan badago, behar diren zeroak gehituko zaizkio ezkerretan azken taldetxo hori laukote bezala osatu arte.
2. Koma abiapuntuz harturik, zenbaki bitarraren alde zatikiarraren zifrak ezkerretatik eskuinetara launaka bilduz taldetxoak egiten dira. Eskuinetako taldetxoari zifraren bat falta bazaio, behar diren zeroak gehituko zaizkio eskuinean azken taldetxo hori laukote bezala osatu arte.
3. Zenbaki osoaren bihurketa lortzeko, laukote bakoitzaren ordezkoa taulan bilatu eta posizio erlatiboa gordez zenbaki hamaseitarra osatuko da.

Bitar-hamaseitar bihurketaren adibidea:

$1011100111{,}0010101_{\lfloor 2} = ???{,}???_{\lfloor 16}$

*00*10 1110 0111 , 0010 101*0* (laukoteak sortu)

$2_{\lfloor 16}\ E_{\lfloor 16}\ 7_{\lfloor 16}\ ,\ 2_{\lfloor 16}\ A_{\lfloor 16}$

$1011100111{,}0010101_{\lfloor 2} = 2E7{,}2A_{\lfloor 16}$

Hamaseitar-bitar bihurketaren adibidea:

$F03{,}B4D_{\lfloor 16} = ???{,}???_{\lfloor 2}$

$1111_{\lfloor 2}\ 0000_{\lfloor 2}\ 0011_{\lfloor 2}\ ,\ 1011_{\lfloor 2}\ 0100_{\lfloor 2}\ 1101_{\lfloor 2}$

$F03{,}B4D_{\lfloor 16} = 111100000011{,}101101001101_{\lfloor 2}$

2.4.2 Zenbaki osoen adierazpidea

Zenbaki osoen adierazpide erraz eta eraginkor bat aukeratzeko **2.4** puntuan aipatutako sistema bitar garbia erabil daiteke. Baina askotan kopuru negatiboak biltegitu behar direnez,

[2] Izan ere, zortzitar eta hamaseitar notazioak bereziak direlako bitarrerako bihurketa zuzenak dauzkate.

sistema bitar garbia zuzenean erabili beharrean bere oinarria duten teknikak garatu dira. Osoak ordenadoreetan gordetzeko lau teknika hauek ikasiko ditugu:

- Modulua eta zeinua (MZ)
- Baterako osagarria (1-O)
- Birako osagarria (2-O)
- Bitarra gainditua (BG)

Sistema fisiko oro finitua eta mugatua delako, lau adierazpide teknika horietan, eta beste edozeinetan, gainezkada deituriko arazoa azalduko da.

2.4.2.1 Modulua eta zeinua

Zenbaki edo kopuru osoen adierazpiderako *modulua eta zeinua* deituriko metodo honetan ezkerragotara[3] dagoen bitak zenbakiaren zeinua markatzen du, 0 denean plus zeinua adieraziz eta 1 denean minus zeinua. Gainerako N-1 bitek zenbakiaren modulua zehaztuko dute.

Demagun zenbakiak lantzeko daukagun sistema informatikoak zortzikoteak onartzen dituela (N=8). Idazkera hamartarrean emaniko *31* eta *-31* zenbakiak adibidez, *modulua eta zeinua* metodoaren bitartez ordenadore horretan adierazteko:

		N-1 = 7	N-1 = 7
31		0 0 0 1 1 1 1 1	0 + 0 + 16 + 8 + 4 + 2 +1
-31		1 0 0 1 1 1 1 1	
	zeinua ↵	modulua	

Adierazpide metodo jakin batean maila edo neurriari H heina esaten zaio, eta adieraz daitezkeen zenbakien kopurua litzateke. Modulu eta zeinu teknikaren heina Nren menpekoa da:

$$1-2^{N-1} \leq H \leq 2^{N-1}-1 \qquad (3)$$

Zortzi biteko kasurako daukagun heina $-127 \leq H \leq 127$ denez ordenadorean gorde daitezkeen zenbakiak muga horietara atzekituko dira. 127 baino handiago bat (edo -127 baino txikiago bat) gorde nahi izanez gero gainezkada[4] suertatzen da.

Hamasei biteko konputazio-sistema baten heina $-32767 \leq H \leq 32767$ mugen artean egongo da. 32 bit erabiltzean adieraz daitekeen edozein zenbaki -2147483647 eta 2147483647 artean aurkituko da.

Modulua eta zeinua adierazpide honek daukan abantailetatik bat heina simetrikoa izatea litzateke, eta akatsen artean zeroarentzat bi adierazpide desberdin edukitzea izango litzateke. Horra hor zeroa zortzi biteko sistemetan:

0 0 0 0 0 0 0 0 (+zero) eta 1 0 0 0 0 0 0 0 (-zero)

[3] Ezker muturrean dagoen bitari (bit esanguratsuena) zeinu-bit esaten zaio.
[4] Ingelesez overflow.

2.4.2.2 1rako osagarria

Zenbaki edo kopuru osoen adierazpiderako metodo honetan ezkerragotara dagoen bit esanguratsuak zenbakiaren zeinua markatzen du ere, 0 denean plus zeinua adieraziz eta 1 denean minus zeinua. Zenbaki positiboetarako, gainerako N-1 bitek zenbakiaren modulua zehaztuko dute (ikusitako *modulua eta zeinua* adierazpidea bezala). Baina, zenbaki negatiboak adierazteko bere simetriko positiboari digituak konplementatu[5] egin behar zaizkio.

Demagun zenbakiak lantzeko daukagun sistema informatikoak zortzikoteak onartzen dituela (N=8). Adibidez, *31* eta *-31* zenbaki hamartarrak *1rako osagarria* (baterako osagarria irakur)metodoaren bitartez ordenadore horretan adierazteko:

N-1 = 7

31	0	0	0	1	1	1	1	1	(lehen bezala)
-31	1	1	1	0	0	0	0	0	(konplementatuz)

zeinua ↵

*1rako osagarria*ren adierazpideak izango duen heina Nren menpekoa da eta (3) formulan emandako berdina izango litzateke:

$$1-2^{N-1} \leq H \leq 2^{N-1}-1 \qquad \text{(3) edo (4)}$$

Metodo honek ere heina simetrikoa du (ikus jarraian emaniko taula), eta zeroarentzat dauzkan bi adierazpideak ondokoak lirateke zortzi biteko sistemetan:

0 0 0 0 0 0 0 0 (+zero) eta 1 1 1 1 1 1 1 1 (-zero)

	*1rako osagarria*ren heina	
	Behemuga	Goimuga
8 bit	-127	$2^7 - 1 = 127$
16 bit	-32767	$2^{15} - 1 = 32767$
32 bit	-2147483647	$2^{31} - 1 = 2147483647$

2.4.2.3 2rako osagarria

Zenbaki edo kopuru osoen adierazpiderako *birako osagarria* deituriko metodo honetan ere, ezkerragotara dagoen bit esanguratsuak zenbakiaren zeinua markatzen du, 0 denean plus zeinua adieraziz eta 1 denean minus zeinua. Zenbaki positiboetarako, gainerako N-1 bitek zenbakiaren modulua zehaztuko dute (ikusitako *modulua eta zeinua* eta *1rako osagarria* adierazpideetan bezala). Baina, zenbaki negatiboak adierazteko dagokion positibotik abiatuz bi urratsetan egiten da:

- Lehenengo urratsa: bere simetriko positiboari digituak konplementatu egiten zaizkio, hau da, zenbaki positiboaren 1rako osagarria lortzen da
- Bigarren urratsa: aurreko urratseko emaitzari 1 gehitzen zaio batuketa bitarra erabiliz. Azkeneko burukorik egonez gero ez da kontutan izan behar

[5] Zeroen ordez batekoak jarriz, eta batekoak dauden posizioetan zeroak idatziz.

Zenbakiak lantzeko daukagun sistema informatikoak zortzikoteak onartzen dituela (N=8) suposatuz, ikus dezagun *31* eta *-31* zenbaki hamartarrak *2rako osagarria* metodoaren bitartez nola adieraziko liratekeen:

	N-1 = 7								
31	0	0	0	1	1	1	1	1	(lehen bezala)
	1	1	1	0	0	0	0	0	(lehenengo urratsa)
							+	1	(bigarren urratsa)
-31	1	1	1	0	0	0	0	1	

*2rako osagarria*ren adierazpideak duen heina, Nren menpekoa izanik, ez da simetrikoa izango eta (5) formularen bitartez emanik dator:

$$-2^{N-1} \leq H \leq 2^{N-1} - 1 \qquad (5)$$

	*2rako osagarria*ren heina	
	Behemuga	Goimuga
4 bit	-8	$2^3 - 1 = 7$
8 bit	-128	$2^7 - 1 = 127$
16 bit	-32768	$2^{15} - 1 = 32767$
32 bit	-2147483648	$2^{31} - 1 = 2147483647$

Zeroarentzat berriz, *2rako osagarria* metodoan adierazpide bakarra dago. Horretarako (+zero) eta (-zero) konparatzea aski da.

	N-1 = 7								
(+zero)	0	0	0	0	0	0	0	0	
(-zero)	1	1	1	1	1	1	1	1	(lehenengo urratsa)
							+	1	(bigarren urratsa)
	1	0	0	0	0	0	0	0	0
ez da aintzat hartzen ↵									

Ikusten denez (+zero) eta (-zero) zenbakiak berdin adierazten dira, azkeneko burukoa ez delako aintzat hartu behar.

Egungo ordenadoreetan zenbaki osoak adierazteko gehien erabiltzen den metodoa *2rako osagarria* deiturikoa denez, aurrekoetan baino gehiago sakonduko dugu batuketak, kenketak, biderkaketak eta zatiketak nola egiten diren ikasiz. Gure azalpenak ahalik eta argien izan daitezen eta helburu didaktiko batekin N=4 biteko hitz-luzeradun ordenadore bat dugula suposatuko dugu.

Demagun beraz 4 biteko ordenadorea daukagula, heinaren goiko taula gogoratuz zenbakirik handiena 7 litzateke eta txikiena -8. Ordenadoreak darabilen barne kodeketa *2rako osagarria*ren bitartez jarraian ematen den taulan erakusten da:

2rako osagarria (N=4)	
Adierazitako kopurua	**Bit segida**
7	0111
6	0110
5	0101
4	0100
3	0011
2	0010
1	0001
0	0000
-1	1111
-2	1110
-3	1101
-4	1100
-5	1011
-6	1010
-7	1001
-8	1000

Taula hori eraikitzeko zenbaki positiboetatik hasiko gara, ahaztu gabe bit esanguratsuena zeinurako erreserbaturik dagoela. Kopuru positibo bat, *3* adibidez, kode bitarrean *11* litzatekeenez ($1x2^1+1x2^0 = 3$) eta txantiloia laukoa delako 0 bat tartekatu beharko da.

3 0 0 *1* *1*

zeinua ↵

Kopuru negatiboei dagozkien kodeak lortzeko biderik errazena (ez bakarra) positiboa konplementatzea eta 1 gehitzea da.

3 0 0 1 1

1 1 0 0 konplementatuz

-3 1 1 0 1 (+1)

zeinua ↵ 5 kode bitarrean

Horregatik *2rako osagarria*ren bitartez adierazitako zenbaki negatibo batean lehenengo bitak zeinua markatzen du (beti 1 izango da) eta gainerako N-1 bitak ez dira kode bitar garbiaren arabera interpretatu behar.

Egoerarik ohizkoena positiboa ezagutzea izaten da, hala ere kontutan izan abiapuntua zenbaki negatiboa izan daitekeela eta bere simetriko positiboa lortzeko berdin egiten da[6]. Hots, alderantziko bidea berbera da, negatiboa konplementatu eta 1 gehitu. Adibidez:

-4 1 1 0 0

0 0 1 1 konplementatuz

4 0 1 0 0 (+1)

zeinua ↵ 4 kode bitarrean

[6] Zenbaki negatibo txikienarentzat izan ezik. Izan ere, zenbaki negatibo txikienak ez du positiboetan simetrikorik.

Batuketa

Kopuru osoen adierazpiderako gaur egun gehien erabiltzen den metodoa *2rako osagarria*rena da, eta horren arrazoirik garrantzitsuena sistema honetan batuketak burutzeko dagoen erraztasunean datza. Izan ere, *2rako osagarria* deituriko adierazpidean emandako bi zenbakien arteko batuketa burutzeko sistema bitar garbian egingo balitz bezala egiten da, baina bai batugaiak zein emaitza N luzera berdinekoak izango direla zainduz.

3	0	0	1	1	(-3)	1	1	0	1
+ 2	0	0	1	0	+ (-2)	1	1	1	0
5	0	1	0	1	(-5)	**1**	1	0	1	1

ez da aintzat hartzen ↵

Beraz, batugai biak positiboak edo batugai biak negatiboak izanik batuketa modu berean egiten da. Horrek ez luke harridurarik sortu beharko, azken finean biak batuketak baitira.

Batugai bat positiboa eta bestea negatiboa batu nahi ditugunean, txikitan eskolan eragiketa hori ikasi genuenean kenketa deitzen dela esan ziguten, eta idazkera hamartarrean jokoan jarri behar diren arauak batuketa eragiketaren arauekin alderatuz desberdinak dira. *2rako osagarria*ren adierazpenean berriz, kenketak batuketak balira burutzen dira. Adibidez:

3	0	0	1	1	(-3)	1	1	0	1
+ (-4)	1	1	0	0	+ 7	0	1	1	1
(-1)	1	1	1	1	4	**1**	0	1	0	0

ez da aintzat hartzen ↵

Ondorioz, *2rako osagarria* adierazpidearen bitartez edozein zeinu duten zenbaki bi batzeko prozedura berdina da. Horrela, *7 - 5* kenketa burutzeko makinak batuketa honen bitartez egin dezake *7 + (- 5)* zenbaki positibo bati dagokion negatiboa lortzen duen zirkuitua diseinatuz, eta edozein zeinuko zenbakiak batzen dituen zirkuitua asmatuz oinarrizko eragiketa biak (batuketa eta kenketa) izango genituzke.

Kenketa

Aurrerago esan bezala *2rako osagarria*ren metodoan batuketa bat izango litzateke. Adibidez, *7 - 5* kenketa burutu ahal izateko (*7*ri *5* kentzea), lehenik *5* zenbakia (makinan 0101 bezala gordeta dagoena) negatibo bihurtuko luke *(-5)* zenbakia (makinan 1011 bezala adierazita dagoena), eta ondoren 0111**+**1011 batuketa burutuko litzateke 0010 lortzeko (idazkera hamartarrean *2*) zein hasierako kenketaren emaitza den.

Biderkaketa

Osoen arteko biderkaketa errepikatzen den batuketa bezala uler daiteke. Hortaz, biderkaketa eragiketa *2rako osagarria*ren adierazpidea duten ordenadoreetan batuketaren zirkuituaz egin ahal izango da, errepikapen kopurua kontrolatzen duen beste zirkuitu kontatzaile baten laguntzarekin batera.

Zatiketa

Zenbaki osoen arteko definitzen den zatiketa eragiketa, zatiketa osoa deitzen da eta duen emaitza beste zenbaki oso bat izango da (gerta daiteke hondarraren bat, beste kopuru oso bat, geratzea ere).

Osoen arteko zatiketa errepikatzen den kenketa bat bezala uler daiteke. Esate baterako, *6 : 2* zatiketa osoaren emaitza *6*ri zenbat aldiz *2* ken diezaiokegun

izango litzateke. Kontutan izan *7 : 2* zatiketa osoaren emaitza aurrekoaren bezala *3* dela. Bietan errepikatzen diren kenketak hiru direlako eta laugarrena ezin daiteke egin geratzen den hondarra (*0* lehenengo kasuan eta *1* bigarrenean) zatitzailea baino txikiagoa delako.

Ondorioz, zatiketa osoa eragiketa makina digitaletan ezartzeko kenketaren zirkuitua (funtsean batuketaren zirkuitua dena) erabil daiteke, kenketen errepikapen kopurua kontrolatzen duen zirkuitu gehigarri bat tartekatuz.

Atal hau amaitu aurretik gainezkadaren arazoa gogora dezagun, eta *2rako osagarria* deituriko adierazpidean overflow edo gainezkada nola detekta daitekeen ikus dezagun. Darabilkigun N=4 biteko konputagailuan *10* zenbakirako ez da biten segidarik eta horregatik ezin da balio hori (eta *7*ren gainetik dauden besteak) gorde, gauza berdina esan daiteke *-9* zenbakirako.

Beraz, *4 + 6* bezalako eragiketaren emaitza ezin daiteke lortu. Batuketaren zirkuituan lortzen duen emaitza 0100 + 0110 = 1010 izango litzateke (idazkera hamartarrean *-6*), emaitza guztiz desegokia dela bistan dago batugai biak positiboak izanik batura negatiboa delako.

Antzeko zerbait esan dezakegu gorde nahi dugun zenbakia txikiegia denean, adibidez *(-3) + (- 6)* bezalako batuketaren emaitza lortu nahi dugunean, gorago egin den eran, *2rako osagarria* adierazpidean 1101 + 1010 = 0111 izango litzateke. Eta batuketaren emaitza idazkera hamartarrean positiboa denez (0111 moldea *(+7)*ri dagokiona da) batuketa horren ezintasunaren adierazlea da, batugaiak negatiboak baitziren.

Laburbilduz, batuketa batean gainezkada gertatu den jakiteko emaitzaren zeinua aztertu behar da; jakinik batugai biak positiboak direla emaitza ezin daitekeela negatiboa izan, eta alderantziz, batugai biak negatiboak direnean emaitza positiborik onartezina dela.

> Batugai bat positiboa denean eta bestea negatiboa denean, *2rako osagarria* delako adierazpidean gainezkadarik gerta daiteke?

Argi dago N=4 heina erabiltzean gainezkada berehala suertatzen dela, mugak -8 eta 7 baitira. Benetako sistema informatikoetan heina handiagoa izaten denez behemuga eta goimugaren artean dagoen esparrua zabalagoa izango da eta gainezkada alditan gertatzen da. Baina ordenadorerik ahaltsuenean ere, zenbakien adierazpiderako txantiloia finitua delako gainezkadaren arriskua izango du ere. Gainezkadaren arazoa ekiditeko biderik zuzenena kopuruek ordezkatzen dituzten kontzeptuen unitateak aldatzea litzateke; horrela, programaren batean milimetroetan edo segundotan kopuruak gorde beharrean (zenbaki handiegiak beharko lirateke) kilometroetan edo orduetan sartuko genituzke.

2.4.2.4 Bitarra gainditua

Bitarra gainditua deituriko adierazpidea zenbaki errealen zati bat adierazteko erabiltzen da, nahiz eta kopuru osoen adierazpidea izan.

Aurreko adierazpideak bezala, zenbaki oso bat *bitarra gainditua* metodoaren bitartez makinan gordetzeko luzera konstantea duen txantiloi bat aukeratu beharko da, baina lehenago aurkeztutako adierazpideetan ez bezala zeinu-bitik ez da ezagutzen. Nahiz eta *bitarra gainditua*n zeinu-bitik ez egon, kopuru positiboak zein negatiboak adierazi ahal izango dira.

Orain arteko N=4 biteko ordenadore didaktikoarekin jarraituko dugu. Ikus dezagun *bitarra gainditua*ren adierazpidean emandako bit konbinazioak zeintzuk diren.

N luzera finkatu ondoren *bitarra gainditua* deituriko adierazpideari dagokion taula eraikitzeko sistema bitar garbian bezala zenbatzen dugu, konbinaziorik garaiena 1111 izanik eta baxuena berriz 0000. Konbinazio garaienari heinaren goimuga *7* egokitzen zaio, eta konbinazio baxuenari behemuga esleitzen zaio, hots *(-8)*. Tarteko zenbakiek ordenez elkartzen dira sistema bitarreko konbinazioekin ondoko taula sortuz:

bitarra gainditua (N=4)	
Adierazitako kopurua	**Bit segida**
7	1111
6	1110
5	1101
4	1100
3	1011
2	1010
1	1001
0	1000
-1	0111
-2	0110
-3	0101
-4	0100
-5	0011
-6	0010
-7	0001
-8	0000

Azter ditzagun taulako *6* eta *(-6)* zenbakiei dagozkien konbinazioak:

	N = 4						
6	1	1	1	0	⟶	$1x2^3 + 1x2^2 + 1x2^1 + 0x2^0 = 14$	14 - 6 = 8
-6	0	0	1	0	⟶	$0x2^3 + 0x2^2 + 1x2^1 + 0x2^0 = 2$	2 - (-6) = 8

Aurreko taulan ematen den *bitarra gainditua* sistemaren edozein bit segida ohizko sistema bitarrarekin alderatuz, beti 8ko diferentzia bat dagoela konturatuko gara. Esate baterako, 1110 biten konbinazioa sistema bitarrean *14* bihurtzen da ($1x2^3+1x2^2+1x2^1 = 14$), eta segida horrek *6* adierazten du *bitarra gainditua* sisteman. Sistema bitar garbian 0000 konbinazioak 0 adierazten du, eta *bitarra gainditua* sisteman berriz *(-8)*.

Aukeratutako heina N=4 izan beharrean bost izango balitz, *bitarra gainditua* eta sistema bitarraren arteko diferentzia konstantea izango litzateke ere; baina aldea 8 izan ordez 16 izango litzateke. Bestalde, N=3 denean ohizko bitarra eta *bitarra gainditua*ren artean dagoen aldea 4koa da (ikus **2.4.3.1** puntuan ematen den taula). Orokorrean, konputazio sistemak N luzerako biten taldeak onartzen baditu, *bitarra gainditua* eta ohizko sistema bitar garbiaren arteko aldea 2^{N-1} konstantea izango da.

Kopuru oso bat, *5* adibidez, *bitarra gainditua* metodoan kodetzeko N=4 denean. Honela egingo da: lehenik 8 gehituko diogu *13* lortzeko eta ostean ohizko sistema bitarrera bihurtuko dugu 1101 erdietsiz.

Zenbaki negatiboekin, *(-4)* adibidez, bide berdina urratu behar da. Hasteko, metodo horrek 8ko gaindiketa bat duenez *(-4)* gehi 8 egiten da 4 sortuz, eta 4 kode bitarrean 100 da. Ez

da ahaztu behar N=4 delako txantiloiak lau bitekoak direla, horregatik lortutako emaitzari zeroak erantsiko zaizkio aurretik laukotea osatu arte. Beraz *(-4)*ri dagokion bit segida 0100 da.

*Bitarra gainditua*ren adierazpideak duen heina, Nren menpekoa izanik, ez da simetrikoa izango eta (6) formularen bitartez emanik dator:

$$-2^{N-1} \leq H \leq 2^{N-1} - 1 \qquad \text{(5) edo (6)}$$

	*bitarra gainditua*ren heina	
	Behemuga	Goimuga
3 bit	-4	$2^2 - 1 = 3$
4 bit	-8	$2^3 - 1 = 7$
8 bit	-128	$2^7 - 1 = 127$
16 bit	-32768	$2^{15} - 1 = 32767$
32 bit	-2147483648	$2^{31} - 1 = 2147483647$

2.4.3 Zenbaki errealen adierazpidea

Zenbaki errealek dituzten ezaugarririk aipagarriena infinitu zenbaki daudela, nahiz eta barruri finitu bat hartu (adibidez 0.0 eta 1.0 mugek definitzen duten barrutia) goi eta behemuga horien artean egon daitezkeen zenbaki errealen kopurua infinitua da.

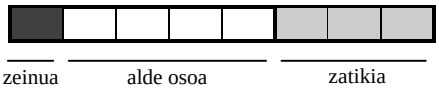
Zenbaki errealak propietate fisikoak neurtzean eta beste hainbat kontzeptuak lantzean agertzen zaizkigu. Adibidez, eta 0.0 eta 1.0 mugek definitzen duten barrutiarekin jarraituz, gertaera bat suertatu dadin probabilitatea 0.5 bitartez adieraztean zenbaki erreal bat darabilkigu eta horren esanahia %50a dela argi dago.

Lehenago esan dugun bezala, ordenadoreetan zenbakizko bat adieraztean digitu-kopuru mugatu batez balia gaitezke eta horrek prezisio galtzea eragiten du zenbaitetan. Zenbaki errealen adierazpidea bi modutan egin daiteke:

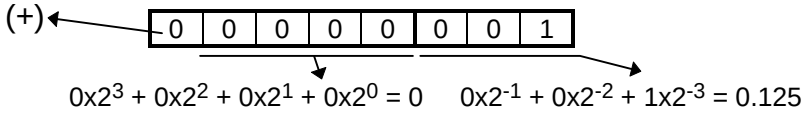
- Koma finkoko adierazpidea
- Koma higikorreko (edo mugikorreko) adierazpidea

2.4.3.1 Koma finkoa

Demagun zortzi biteko ordenadore hipotetiko bat daukagula eta koma bosgarren eta seigarren biten artean dagoela beti. Suposa dezagun ere komatik ezkerretara dagoen eremua (zenbaki errealaren osoa den zatia) modulua eta zeinua teknikaz adierazten dela (lehen bita zeinurako gainerakoak modulurako); komatik eskuinera dauden hiru bitek zenbaki errealaren alde zatikiarraren zifrak direla:

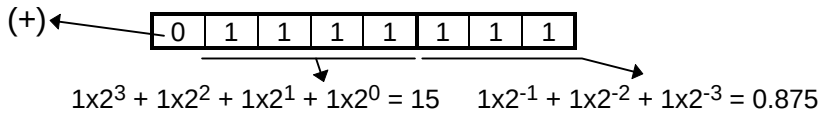


Alegizko ordenadore horrek izan dezakeen zenbaki positiborik txikiena hau litzateke:



Ezagutzen dugun Zenbaketaren Oinarrizko Teorematik ondoriotzen den polinomioa aplikatuz gero, 0 0000 001 zenbaki bitar txikienari sistema hamartarrean dagokion ordaina +0.125 da. Eta hurrengo zenbaki erreala (bitarrean 0 0000 010 kodeturik datorrena) sistema hamartarrean +0.25 litzateke. Ulertzen denez +0.125 eta +0.25 balioen artean egon daitezkeen zenbatekoak hiru bitekin ezin dira errepresentatu, honi *errepresentazio-errorea* esaten zaio eta digitalizazioaren eragina da.

Ikusten denez zenbakirik handiena ordenadore horretan +15.875 litzateke:



Koma finkoko adierazpidea, azken finean, zenbaki osoen adierazpidea bezalakoa da. Ordenadore hipotetikoaren zenbaki erreal positiboen segida idatziz gero, honelako zerrenda geratzen zaigu:

0.0	0.125	0.25	0.375	0.5	0.625	0.75	0.875
1.0	1.125	1.25	1.375	1.5	1.625	1.75	1.875
2.0	2.125	2.25	2.375	2.5	2.625	2.75	2.875
...							
14.0	14.125	14.25	14.375	14.5	14.625	14.75	14.875
15.0	15.125	15.25	15.375	15.5	15.625	15.75	15.875

2.4.3.2 Koma higikorra

Kopuru errealak adierazterakoan balioa gordetzeaz gain komaren kokapena biltegitu ahal da. Hau lortu nahirik idazketa zientifikoa erabil daiteke, zein *koma higikorreko formatu bitarra*, edo laburrago, *koma higikorreko* adierazpidea deitzen den.

Koma higikorreko edo *koma mugikorreko* adierazpidean, zatikia den ***K*** kopuru bat errepresentatzeko ondoko formula aplikatzen da:

$$K = (+)\ (mantisa) \times (oinarri)^{berretzaile} \qquad (7)$$

Koma higikorreko adierazpidean ***K*** kopuruak izan ditzakeen adierazpideak infinituak direlako, idazkera zientifiko normaldua erabiltzen da: mantisak alde osorik ez du, komaren eskuinetara dagoen lehenengo digitua ez da 0 izango (zero zenbakia adieraztekoan izan ezik).

Horrela, 0,0046 eta -128,3 zenbakien adierazpideak hauek dira oinarria 10 denean:

$0{,}0046 = 0{,}???? \times 10^{???}$	$0{,}0046 = 0{,}46 \times 10^{-2}$
$-128{,}3 = 0{,}???? \times 10^{???}$	$-128{,}3 = -0{,}1283 \times 10^{3}$

Idazkera zientifiko normalduan mantisaren alde osoa 0 denez eta modulua 0 ez den balio batez hasten delako, mantisa zenbaki oso baten bitartez ordezka daiteke. Eta ondorioz erreala den ***K*** kopurua mantisa/berretzaile zenbaki osoen bikoteari esker adierazi ahal izango da.

Koma higikorreko adierazpidean ***K*** kopurua zehaztean (7) formulari so eginez, hiru elementu aintzat hartu behar dira: mantisaren zeinua, mantisaren balioa edo modulua eta

berretzailea. Izan ere, makina digitaletan oinarria inplizitoki definiturik dator 2 delarik, gainerako beste hiru elementu horiek kode bitarrean[7] emanik izango dira.

Beraz, *koma higikorreko* adierazpidean ondoko hiru eremuak ezagutuko dira:

zeinua	berretzailea	mantisa

Demagun zenbaki errealak lantzeko daukagun sistema informatikoak, zatikiak byte batez gordetzen dituela. Eta suposa dezagun zenbateko erreala definitzeko aipatutako hiru elementu horiek jarraian agertzen diren eremuetan banatzen direla:

zeinua	berretzailea			mantisa			

Eremu horietan zenbaki errealak nola kodetzen diren ikasteko adibide bat ikus dezagun. Demagun 01101001 konbinazioa gorderik dagoela, eremuka banatuz 0 110 1001.

zeinua	berretzailea			mantisa			
0	1	1	0	1	0	0	1

Zenbakia deskodetzeko atzetik aurrera joko dugu hiru urratsetan:

1. <u>Mantisa</u>

Hirugarren eremuan datorrena mantisa da, bit segida hori hartu eta hasieran koma ezarriko dugu:

, 1 0 0 1

2. <u>Berretzailea</u>

Bigarren eremuko edukia aztertzen da, 3 biteko *bitarra gainditua* bezala interpretatuz. Ondoko taulan 1 1 0 konbinazioari +2 dagokiola ikus daiteke.

bitarra gainditua (N=3)	
Adierazitako kopurua	**Bit segida**
3	111
2	110
1	101
0	100
-1	011
-2	010
-3	001
-4	000

Sistema Bitarra	
111	7
110	6
101	5
100	4
011	3
010	2
001	1
000	0

Horregatik mantisatik lortutako emaitzan koma bi posizioz eskuinetara higituko dugu:

1 0 **,** 0 1

[7] Kopuru osoak direnez, ikasi ditugun kode bitarraren bat aukeratzen da. Berretzailerako, gehienetan, bitarra gainditua metodoa erabiltzen da. Mantisa, berriz, 1rako edo 2rako osagarria deituriko metodoek zehazten dute. Zeinurako bit bakarra erreserbatzen da.

Berretzailea negatiboa izan balitz, 0 0 1 adibidez (bitarra gaindituan -3). Orduan mantisako koma ezkerretara hiru jauzi eginez higituko litzateke zeroak tartekatuz:

, 0 0 0 1 0 0 1

3. Zeinua

Zeinua markatzen duen bitak 0 balio duenez zenbaki erreala ez da negatiboa izango:

(+) 1 0 **,** 0 1

Dena bilduz:

(+) 1 0 , 0 1 $\longrightarrow$ $1x2^1 + 0x2^0 + 0x2^{-1} + 1x2^{-2} = 2 + 0 + 0 + {}^1/_4$

zeinua	berretzailea			mantisa			
0	1	1	0	1	0	0	1

(+) 1 0 , 0 1 $\longrightarrow$ $+2\,{}^1/_4$

Koma higikorreko adierazpideak duen arazoa doitasunarena da. Zenbait kopuru zehatzentzat doitasunaren arazoa zenbaketa sistema guztietan ematen den fenomenoa da, gogoratu bestela Π zenbakia errepresentatzeko infinitu zifra hamartar behar direla eta 3,1415 balioak 3,14 balioak baino doitasun handiagoa ematen duela[8]. Zenbat eta mantisaren eremurako bit-kopuru gehiago izan zenbakien doitasuna handiagoa izango da.

Doitasunaren arazoa beste iturri batetik etor daiteke, sistema hamartarretik bitarrera bihurketa egitean sor daitekeena hain zuzen ere. Ikus dezagun adibidez zer gertatzen den sistema hamartarrean guztiz definiturik dagoen -3,625 zenbakia (beste modu batez $-3\,{}^5/_8$) koma higikorreko formatuan gordetzerakoan:

$-3{,}625 = -3\,{}^5/_8 \longrightarrow$ (-) 1 1 , 1 0 1

$1x2^1 + 1x2^0 + 1x2^{-1} + 0x2^{-2} + 1x2^{-3} = 2 + 1 + {}^1/_2 + 0 + {}^1/_8 = 3\,{}^5/_8$

zeinua	berretzailea			mantisa			
1	?	?	?	?	?	?	?
1	1	1	0	?	?	?	?
1	1	1	0	1	1	1	0

(-) 1 1 , 1 0 1 $\longleftarrow$ $-3\,{}^5/_8$

0 , 1 1 1 0 1 $\longleftarrow$ bi jauzi

0 , 1 1 1 0 1 $\longleftarrow$ mantisa

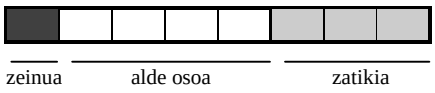
galdutako informazioa mantisaren eremuan tokirik ez delako geratzen ↵

Mantisaren eremuak duen luzera finitua delako doitasun arazoak direla medio, azkenekoan gorde den zenbakia $-3\,{}^5/_8$ izan beharrean $-3\,{}^1/_2$ izan da. Gogoratu fenomeno honi *errepresentazio-errorea* esaten diogula; honekin batera zenbakien artean eragiketak burutzean *konputazio-erroreak* gerta daitezke bitarteko emaitzekin ordenadoreak mozketak edo hurbilketak egiten baldin baditu, adibidez nahiz eta 1.0 eta 3.0 zenbakiek errepresentazio errorerik ez izan, euren arteko zatiketa egiten badugu eta ondoren 3.0-az bidekatzen badugu azken emaitza ez da uste genezakeen bezala 1.0 izango 0.9999 baizik.

[8] Sistema hamartarrean zenbait zenbaki osoki adieraztea ezinezkoa bada (infinitu zifra behar direlako), kode bitarrean arazoa areagotzen da amaitzen ez diren zenbakiak askoz gehiago direlako.

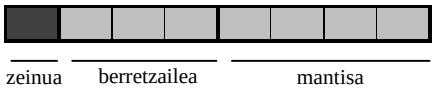
Koma higikorreko adierazpidearen funtzionamendua ulertu eta gero koma finkoko baliokidea izan daitekeen sistemarekin konpara dezagun. Zortzi biteko ordenadore hipotetikoa kontutan izanik koma finkoa hiru eremuz (eta orden honetan) ematen zen:

koma finkoko adierazpidea

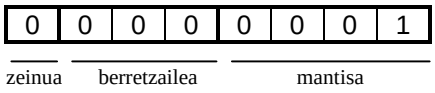


Koma higikorreko adierazpiderako zortzi bitak honela banatu ditugu:

koma mugikorreko adierazpidea



Gogoratu koma finkoko positibo txikiena +0.125 zela, kalkula dezagun koma higikorra delako adierazpideari dagokion zenbaki erreal positibo txikiena. Hona hemen:



0 000 0001 zenbakia deskodetzeko hiru urratsak, hurrenez hurren aplikatuz dagokion zenbaki hamartarra lortuko dugu.

1. <u>Mantisa</u>

Hirugarren eremuan datorrena mantisaren bit segida hartu eta hasieran koma ezarriko dugu:

, 0 0 0 1

2. <u>Berretzailea</u>

Bigarren eremuko edukia aztertzen da, 3 biteko *bitarra gainditua* bezala interpretatuz. Aurreko taulan ikus daiteke berretzailearen 0 0 0 konbinazioari -4 dagokiola.

Negatiboa delako mantisako koma ezkerretara higituko dugu lau jauzi eginez, horretarako zeroak tartekatzen dira:

, *0 0 0 0* 0 0 0 1

3. <u>Zeinua</u>

Zeinuaren bita 0 denez bilatzen dugun zenbaki erreala positiboa izango da:

(+) **,** 0 0 0 0 0 0 0 1

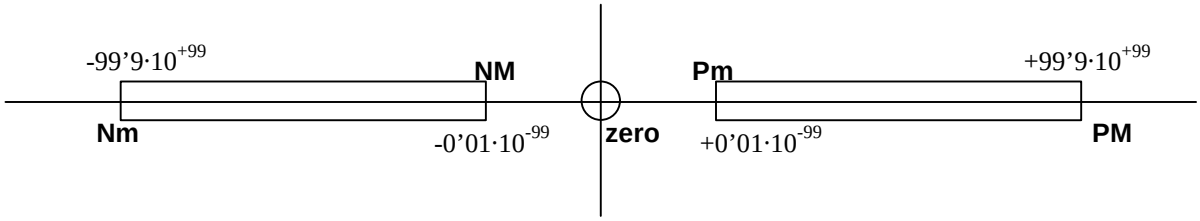
Hiru urratsak bilduz eta hamartarrera igaroz:

$$(+)\ ,00000001 \longrightarrow 0x2^{-1} + 0x2^{-2} + 0x2^{-3} + 0x2^{-4} + 0x2^{-5} + 0x2^{-6} + 0x2^{-7} + 1x2^{-8} = {}^{1}/_{256} = 0{,}00390625$$

Zortzikote bat darabilen makinaren koma finkoko adierazpidean positibo txikiena +0.125 bazen, koma mugikorreko adierazpidean txikiena +0.00390625 da, eta txikitik handienerantz dagokion sekuentzia aztertuz prezisioa handiagoa da koma higikorreko adierazpidean:

0.0 0. 00390625 0. 0078125 0.0117875 0.015625 0.01953125 ...

Koma higikorreko adierazpideak duen heina ez da lineala zenbaki osoekin gertatzen zen bezala. Heina ezagutzeko, kasu honetan, berretzaile eta mantisaren eremuen luzerak ezagutu behar dira. Heina grafikoki jarraian ematen den irudian azaltzen da:



Non:

- **Nm** (Negatibo minimoa) zenbaki negatibo txikiena den

$$\mathrm{Nm} = (-)\ (mantisa\ maximoa) \times (oinarri)^{berretzaile\ maximoa}$$

- **NM** (Negatibo Maximoa) zenbaki negatibo handiena den

$$\mathrm{NM} = (-)\ (mantisa\ minimoa) \times (oinarri)^{(-)\,berretzaile\ maximoa}$$

- **zero** (0 zenbakia) beti definiturik dago

- **Pm** (Positibo minimoa) zenbaki positiborik txikiena den

$$\mathrm{Pm} = (+)\ (mantisa\ minimoa) \times (oinarri)^{(-)\,berretzaile\ maximoa}$$

- **PM** (Positibo maximoa) zenbaki positibo handiena den

$$\mathrm{PM} = (+)\ (mantisa\ maximoa) \times (oinarri)^{berretzaile\ maximoa}$$

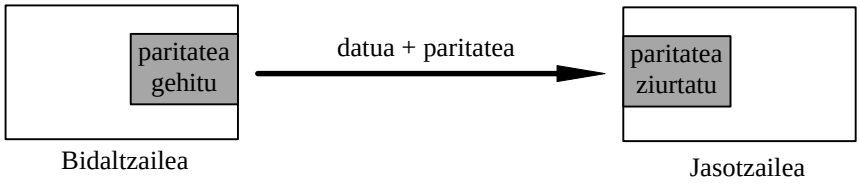
Ikusten denez lau zonaldeko zenbakiak ezin daitezke adierazi, zeroaren inguruan (alde positiboan), zeroaren inguruan (alde negatiboan), gainezkada positiboko zonaldea (eskuin muturra) eta gainezkada negatiboko zonaldea (ezker muturra).

Hala ere, heina edo barrutiaren barnean badira zenbait kopuru adiera ezin dena (Π konstantea gogoratu, edo -3,625 zenbakia zortzi biteko ordenadorearen kasuan -3,5 bezala gordetzen dela biribilketa burutzen delako).

2.4.4 Erroreak detektatzeko kodeak

Ordenadore batean, informazioa bere barneko ataleen artean transmititzean erroreak gerta daitezke (hautsa diskoetan dagoelako, hezadura delako, eremu magnetikoak direlako, ...). Sarreran datu bati dagokion kodea *gaizki* eraikitzen bada, lortuko den emaitza desegokia izango da.

Hona hemen paritate-kontrola nola egiten den:



Kontzeptuak:

- Paritate-kontrola
- Paritate-errorea
- Paritate-bita
- Paritate bakoiti
- Paritate bikoiti

2.5 ARIKETAK

1. Hona hemen ASCII kodez emaniko mezu bat. Zer dio?

1000101 1100100 1100101 1110010 0100000 1100010 1100001
1110011 1101111 1100001 1101110 0100000 1101000 1100001
1110010 1101001 1110100 1111010 1100001 0100000 1101000
1100001 1110010 1100001 1101110 0100000 1100001 1101100
1100100 1100101 1100001 1101110 0100000 1100111 1100001
1110010 1101001 1110100 1111010 1100001 0111111 0101110

2. ASCII kodeko letra bat eta bere majuskula aztertuz, zein da kode biren arteko erlazioa?. Adibidez aurreko galderako lehen eta hirugarren karaktereek dauzkaten kodeen arteko harremana.

3. Hurrengo esaldia ASCII kodez errepresentatu.

Jatorrizko ASCII kodeak 7 biteko txantiloi bat darabil

4. Hiru byte izanik, zein da ASCII bitartez kode daitekeen zenbatekorik handiena? Eta hiru byte horien bitak sistema bitarraren digituak balira erabiliz, zein litzateke orduan zenbakirik handiena?

5. Esku bateko atzamarrak elementu bitarrak kontsideratuz (luzaturik egotea, ala bildurik egotea), zenbat konbinazio desberdin adierazi ahal izango dira?

6. Zenbaketaren Oinarrizko Teorema enuntzia ezazu.

7. Sistema bitarrean emandako hurrengo zenbakiak hamartarrera bihurtu.

1 10 101,01 111,001 1001,111 0,0111 0,001 1010101,01

8. Sistema hamartarretik bitarrera bihur itzazu:

20,5 31,25 72,125 0,3125 16,6235 10,01 894,46875 44,75

9. Sistema bitarreko batuketa erabiliz, ondoko eragiketak burutu:

10101,01	*111,001*	*1001,111*	*0,0111*	*0,001*	*101000,01*
+111,11	*+1,011*	*+11,001*	*+0,0111*	*+11,01*	*+1101,011*

10. Zenbaketa sistemen arteko baliokidetzak bilatuz jarraian ematen den taula osa ezazu:

BITAR	***ZORTZITAR***	***HAMARTAR***	***HAMASEITAR***
100111010011			
	4376		
		4005	
			7C13
10001,011			
	41,7		
		34,625	
			4D,F2

11. Jarraian ematen diren kopuru osoak, 8 biteko makina bateko modulua eta zeinua idazkerara bihurtu:

127 101 63 33 1 0 -1 -33 -63 -101 -127

12. Jarraian ematen diren kopuruak, 5 biteko makina bateko modulua eta zeinua idazkeran datoz. Zeintzuk dira hamartarreko baliokideak?

10101 00111 11011 11001 01110 11000 11110 00011

13. Jarraian ematen diren kopuru osoak, 8 biteko makina bateko 1rako osagarria idazkerara bihurtu:

127 101 63 33 1 0 -1 -33 -63 -101 -127

14. Jarraian ematen diren kopuruak, 5 biteko makina bateko 1rako osagarria idazkeran datoz. Zeintzuk dira hamartarreko baliokideak?

10101 00111 11011 11001 01110 11000 11110 00011

15. Jarraian ematen diren kopuruak, 8 biteko makina bateko 2rako osagarria idazkerara bihurtu:

127 101 63 33 1 0 -1 -33 -63 -101 -127

16. Jarraian ematen diren kopuruak, 5 biteko makina bateko 2rako osagarria idazkeran datoz. Zeintzuk dira hamartarreko baliokideak?

10101 00111 11011 11001 01110 11000 11110 00011

17. Hona hemen 4 biteko makina batean 2rako osagarrian emaniko batugaiak. Baturak lor itzazu, eta zuzentasuna konprobatu batugaiak zein emaitza sistema hamartarrera igaroz:

1011	*0011*	*1010*	*1000*	*0110*	*0000*	*1000*	*0111*
+0011	*+0001*	*+1010*	*+1100*	*+0110*	*+1010*	*+1110*	*+0001*

18. Hona hemen 4 biteko makina batean 2rako osagarrian emaniko zenbakiak. Makinak kenketak burutzeko, zirkuitu batutzaile eta zirkuitu ukatzaile banaren bitartez egiten baditu. Ordenadorearen funtzionamendu simulatu kendurak lortuz eta emaitzen zuzentasuna konprobatu sistema hamartarraren bidez:

1011	*0011*	*1010*	*1000*	*0110*	*0000*	*1000*	*0111*
-0011	*-0001*	*-1010*	*-1100*	*-0110*	*-1010*	*-1110*	*-0001*

19. Aurreko galdera bietan gainezkadak suertatzen diren kasuak aztertu eta azaldu.

20. Hurrengo eragiketak 2rako osagarrirako formatuan idatz (4 biteko makina dela suposa daiteke). Ondoren emaitzak lortu, makinak zirkuitu kentzailerik ezarririk ez duela jakinda (baina bai izango ditu zirkuitu batutzailea eta zirkuitu ukaitzailea):

6	*1*	*(-2)*	*1*	*0*	*0*	*3*	*7*	*(-6)*	*4*	*(-5)*
-1	*+4*	*+3*	*+5*	*-6*	*+4*	*+3*	*-4*	*+4*	*+3*	*-4*

21. Testuan ikusitako koma higikorreko formatua aintzat harturik, ondoko bit konbinazioak deskodetu:

00010101	*11000111*	*11000011*	*11001001*	*01110000*
11000001	*00011111*	*10000101*	*11001011*	*00000001*

22. Testuan ikusitako koma higikorreko formatua aintzat harturik, ondoko zenbakiak kodetu. Doitasun arazoa noiz gertatzen denean azaldu errorea kuantifikatuz.

$-3^{3}/_{4}$ $4^{1}/_{4}$ $-2^{1}/_{2}$ $^{3}/_{4}$ $5^{3}/_{8}$ $-5^{5}/_{8}$ $1^{7}/_{8}$ $-4^{1}/_{16}$ $2^{3}/_{16}$

23. Testuan ikusitako koma higikorreko formatua aintzat harturik, heina ahalik eta zehatzen definitu.

24. Testuan ikusitako koma higikorreko formatua aintzat harturik, ondoko bit konbinazio biren artean zeinek adierazten du zenbateko handiena?. Horrelako konbinazioak jasota eta > konparaketa bat eginez zenbateko handiena zein den jakiteko prozedura erraz bat asma ezazu.

25. Koma higikorrarako ordenadore batek darabilen formatua hau da:

- 23 eta 30 bitarteko bitek berretzailea adierazten dute, bitarra gaindituaren bitartez (gaindiketa 128koa izango da)
- 0 posiziotik 22ra dauden bitek mantisa normaldua adieraziko dute. Horretarako 1rako osagarria metodoa aukeratu izan da
- Bit esanguratsuenak (31. posizioan dagoenak) zeinua markatzen du. 0 balioa zenbaki positiboetarako
- Berretzailearen oinarria 2 da
- Zero zenbaki hamartarra adierazteko bit guztiak 0 izango dira

Ondoko zenbakiak kodetu:

-12 12 136,55 -33,333 1996 -8520,0125 -69,87 996,453

26. Azkeneko galderaren ordenadoreak zenbaki errealak adierazteko duen 32 biteko formatuaren heina kalkula ezazu.

27. Testuaren **2.3.1** puntuan aipatutako ASCII kodeari buruz informazioa bilatu eta txosten bat osatu.

28. Testuaren **2.3.2** puntuan aipatutako BCD kodeari buruz informazioa bilatu eta txosten bat osatu.

29. Testuaren **2.3.3** puntuan aipatutako EBCDIC kodeari buruz informazioa bilatu eta txosten bat osatu.

30. Testuaren **2.3.4** puntuan aipatutako OEM kodeari buruz informazioa bilatu eta txosten bat osatu.

31. Testuaren **2.3.5** puntuan aipatutako ANSI kodeari buruz informazioa bilatu eta txosten bat osatu.

32. Testuaren **2.3.6** puntuan aipatutako UNICODE kodeari buruz informazioa bilatu eta txosten bat osatu.

2.6 PROGRAMAK

Hona hemen 2. kapituluaren programak orrialdeen arabera sailkaturik:

Izena	*Programaren identifikadorea*	*ORRI.*	*Ikasgaia*
ASCII-7.PAS	ASCII_7_Taula	2-04	Konputagailu-kode
BIT.PAS	ZenbakiOsoenBarneAdierazpidea	2-14	2rako osagarria

2.7 BIBLIOGRAFIA

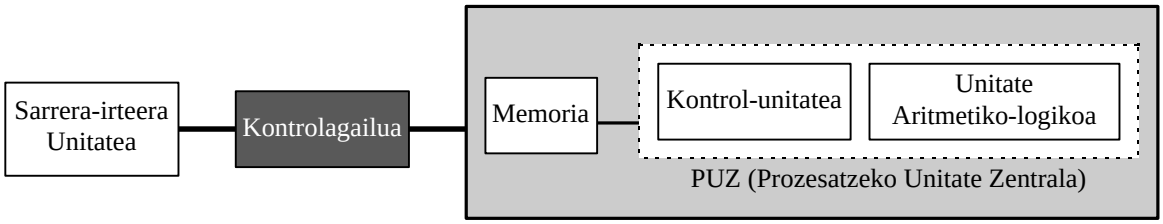
- Brookshear, J.G., *Introducción a las Ciencias de la Computación*, Addison-Wesley Iberoamericana, Wilmington, 1995
- Elhuyar Informatika-taldea, *Informatika Oinarrizko kontzeptuak*, Elkar-Elhuyar, Estella, 1984
- Tanenbaum, A., *Konputagailuen Antolaketa Egituratua*, EHU/UPV-ko Argitarapen Zerbitzua, 1994

3. ATALA: KONPUTAGAILUAREN BARNE OSAGAIAK

AURKIBIDEA

3.1 SARRERA

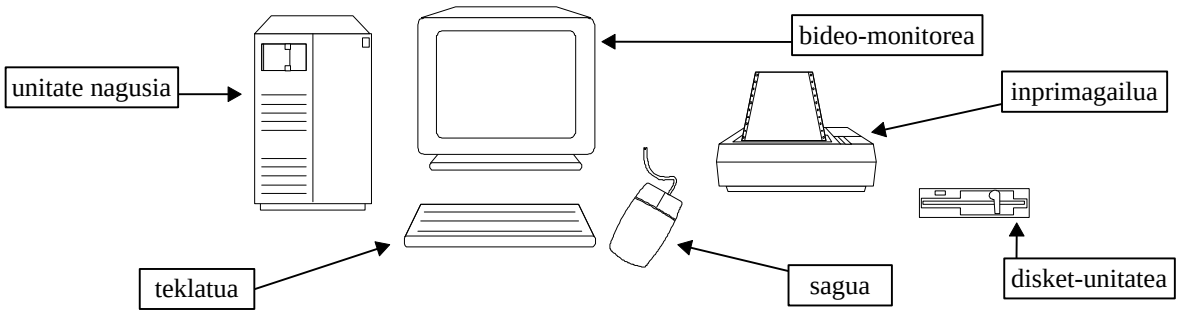
1. kapituluan makina algoritmikoei buruz aritu ginen eta 2. kapituluan informazioa nola errepresentatzen den ikusi genuen, aurreko bien jarraipena dela esan daiteke orain hasten dugun 3. kapitulu hau. Izan ere, konputagailuaren barne osagai funtsezkoenak eskema hau erakutsiz **1.3.4 Gaur egungo makina algoritmikoen arkitektura** puntuan aipatu zen:



3. kapitulu honen zioa eskema horretako elementuei buruzko informazio zehatzagoa ematea da, eta nola ez, konputagailuaren funtzionamendu orokorra azaltzea ere. Has gaitezen gaur egungo makina algoritmikoek zer nolako itxura daukaten kanpotik begitaraturik, kanpotik lehenik eta gero barnera egingo diogu begirada.

3.1.1 Makina algoritmikoa kanpotik, hurbilpena

Denok ikusi dugu inoiz konputagailu edo (modu farfailtsu batean deituta) sistema informatiko bat, eta badakigu subsistema batzuen batura dela konputagailua. Askotan agertzen diren subsistemak ondoko hauek dira: unitate nagusia, teklatua, bideo-monitorea (edo pantaila), sagua, disko unitateak, bozgoragailuak eta inprimagailua.



Unitate nagusia izan ezik aurreko zerrendako subsistema guztiak periferikoak dira, eta izenek adierazten dutenez unitate nagusia edo zentrala sistema informatikoaren oinarria da eta gainerako periferikoak aparatu osagarriak. Honela definitzen dira priferikoak: periferiko bat unitate nagusitik independentea den organo osagarria da, unitate zentraletik independentea izan arren beronekin batera funtzionatzeko prestatuta dagoenez sistema globalaren posibilitateak gehitzen ditu.

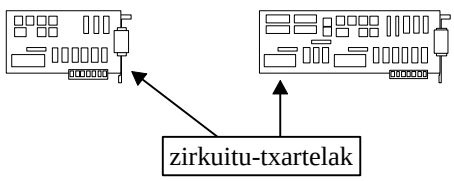
Unitate nagusia zenbait zirrikitu eta argitxo dituen kutxa metalikoa da, konputagailuaren organo garrantzitsuenak babesten dituen kutxa da. Bere aurreko zirrikutei disko-unitate esaten zaie eta honela definitzen dira: disko-unitatea periferiko bat da eta bere helburua, datuak irakurri/idatziz informazioa ordenadoretik diskoetara (eta alderantziz, diskoetatik ordenadoreako memoriara) igarotzea da.

Unitate nagusiaren atzeko aldetik konektoreak aurki genitzake, batzuk okupaturik egongo dira (argindarraren kableena, monitorea, sagua, teklatua eta inprimagailuarena) eta beste batzuk hutsik egongo dira. Atzean daude ere unitate zentrala ireki ahal izateko torlojuak, bihurkin bat hartu eta aska ditzagun torlojutxoak eta unitate nagusiaren estalki metalikoa ken dezagun.

3.1.2 Makina algoritmikoa barrutik, hurbilpena

Konputagailu baten unitate nagusiko estalkia kentzean, besteak beste, honako hauek agertuko dira:

- Argindarra elikatzen duen iturria. Metalezko kutxa poliedriko honek sare elektrikotik 220 voltako korronte alternoa hartu eta helburu bi beteko ditu; batetik konputagailuaren zenbait osagai elikatuko du (tentsio horretan lan egiten baitute), eta bestetik korronte alternoa errektifikatu eta 5 voltako korronte jarraia lortzen du ordenadoren zirkuitu integratuek 5 voltako seinalea behar baitute.
- Kanpoko zirrikitua erreferentziarekin disko-unitateak identifika daizteke, bai disko magnetikoak zein disko optikoekin lan egiten duten disko-unitateak oinarrizko plakari konektaturik daude. Ikusi ere korronte elektrikoa nondik heltzen zaien.
- Konputagailuaren disko gogorra ikustea ezinezkoa da babestuta dagoelako. Disko malguak (disketeak) baino askoz edukiera handiagoa du, atzipen-denbora ere disko malguena baino askoz laburragoa da.
- Ordenadoreak egiten dituen soinu sinpleenak entzun ahal izateko, unitate nagusiaren aurre aldean bozgoragailu txiki bat kokaturik dagoela ikus daiteke.
- Oinarrizko plaka. Unitate zentralaren azalera osorik hartzen duen zirkitu-txartel berdeari oinarrizko plaka edo oinarrizko zirkuitu esaten zaio, bertan kokatzen dira txip beltzak diren zirkuitu integratuak. Zirkuitu integratuetan bat bereizten da, tamainan handiena delako eta gehiegirik berotu ez dadin bere gainean aireztailu bat duelako, mikroprozesadorea da; konputagailuaren txip garrantzitsuena mikroprozesadorea[9] da. Identifikatzeko erraza den beste txip multzo bat, oinarrizko plakan kokatzen den ordenadorearen memoria da. Mikroprozeadorea, edo mikroa, eta gainerako txipen arteko lotura erdiesteko oinarrizko plakan zirkitu metalikoak inprimaturik daudela ikus daiteke, mikroa eta gainerako txipak lotzen dituzten zirkuitu inprimatuei bus deritze.
- Zirkuitu-plakak edo zirkuitu-txartelak. Oinarrizko plakak egiten duen bezala, zirkuitu-txartelek osagai elektronikoak konektatzen dituzten zirkuitu inprimatuak dira. Zirkuitu-txartelak oinarrizko plaka baino txikiagoak dira eta honek propio dituen erretenetan txertatzen dira elkarrekin konektatzeko. Zirkuitu-txartel bakoitzak periferiko bat kontrolatzeko balio duela ikasiko dugu aurrerago, hona hemen oinarrizko plakan txertatu gabeko zirkuitu-txartel pare baten irudia:



Konputagailuaren osagai fisikoak arinki deskribatu ondoren, hemendik aurrera euren funtzionamendua azaltzen saiatuko gara.

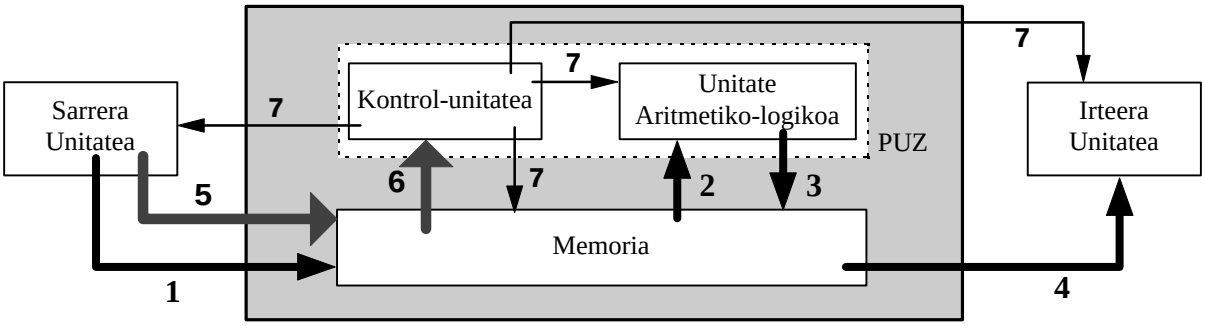
[9] Konputagailu txikietan, mikroprozesadore (edo mikro) txipak 1. kapituluan aipatu genituen elementu hauek biltzen ditu: UAL (Unitate Aritmetiko-logikoa), KU (Kontrol-unitatea) eta erregistroak.

3.2 GAUR EGUNGO MAKINA ALGORITMIKOEN ARKITEKTURA

Aipatutako subsistema guztiak (unitate nagusia, monitorea, teklatua, sagua, ...) biltzean konputagailu hitzaz ezagutzen dugun makina eskuratzen dugu, gailu honek duen ezaugarririk nabarmenena makina algoritmiko bat dela da, bere arkitektura nolakoa den ikus dezagun.

Mikroprozesadore deituriko txipak, besteak beste, Unitate Aritmetiko-logikoa eta Kontrol-unitatea barneratzen ditu (bien artean Prozesatzeko Unitate Zentrala osatzen dute). Makina algoritmikoaren Memoria oinarrizko plakan itsatsita dago, eta oinarrizko txartel honen gainerako txipak periferikoen kontrolagailuak dira.

Jarraian erakusten den eskeman konputagailuaren unitate nagusia jarri da erdian eta periferikoak inguruan, sarrerako periferikoak ezkerrean eta irteerakoak eskuinean. Marraztu diren geziak[10] konputagailuaren barrurantz eta konputagailutik kanporantz dauden fluxuak dira.



Fluxuak hiru dira: datuak, instrukzioak eta kontrola. Hona hemen banan-banan.

Datuen fluxua

Datuak sarrerako periferiko batetik ematen zaizkio unitate nagusiari. Esate baterako, egikaritzen ari den programa batek gure izena sar dezagula eska diezaguke, eta guk sarrera unitatea den teklatuaren bidez eman diezaiokegu, edo bestela, beste programaren batek behar dituen datuak diskete batean irakurriz lor ditzake.

Datuak kanpotik datozenean eta ordenadorearen barrura heltzen direnean Memorian kokatzen dira (**1** gezia).

Datuekin eragiketaren bat egin behar bada, memoriatik Unitate Aritmetiko-logikora eramango da (**2** gezia). Unitate Aritmetiko-logikoak operazioa burutu ondoren emaitza berriro itzuliko da Memoriara (**3** gezia).

Konputagailuaren jarduerako emaitzak guri aurkezteko irteerako periferiko bat beharko da. Esate baterako, programaren emaitzak monitorean ager daitezke, edo bestela inprimagailura bidaltzea nahi izango dugu.

Konputagailuak egindako prozesaketaren emaitzak Memorian aurkitzen dira eta periferikoetara kanporatu ahal izango da (**4** gezia).

Instrukzioen fluxua

Konputagailuak programaren bat egikaritu behar duenean (programak diskoetan gorderik egoten dira) bete behar duen lehen urratsa programa hori Memoriara eramatea da (**5** gezia).

[10] Geziek norantza jakin bat dutelako, fluxua nondik nora ematen den adierazten dute.

Programa bat instrukzioen zerrenda bat da, eta programa exekutatua izan dadin instrukzioak banan-banan Kontrol-unitatera eramango dira Memoriatik (**6** gezia).

Kontrol-fluxua

Kontrol-unitateak gainerako organo guztiak kontrolatzen ditu (**7** geziak), horregatik kontrol-fluxua Kontrol-unitatetik atera eta beste osagaietara iristen diren geziaz errepresentatzen da.

Konputagailu baten barne antolaketa laburbiltzen duen aurreko orrialdeko eskema hori gogoan izatea gomendatzen da, bertan agertzen baitira ordenadorearen funtzionamenduan parte hartzen duten funtsezko elementuak eta hura nola dabilen ulertu ahal izateko gakoa da. Hurrengo puntuetan konputagailuaren barne osagaiak zehaztasunez deskribatuko ditugu, errazago izan dadin sei puntuko banaketa hau egitea erabaki dugu:

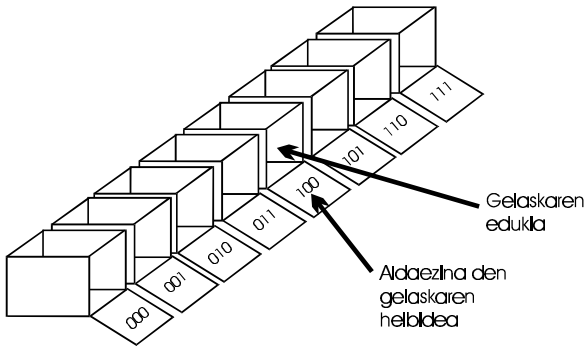
3.2.1 Memoria
3.2.2 Bus konektoreak
3.2.3 Unitate Aritmetiko-logikoa
3.2.4 Kontrol-unitatea
3.2.5 Periferikoak
3.2.6 Memoria masiboa

3.2.1 Memoria

Memorian informazioa biltegitzen da, geroago erabilia izan dadin. Informazioa biltzen dela esatean, bi motatako informazioa izan daitekeela azpimarratu behar da, batetik ordenadoreak landuko dituen datuak eta berak eskaintzen dituen emaitzak, eta bestetik, ordenadorearen zeregina gidatzen duen programa ere memorian gordetzen da.

Ordenadorearen memoriak, bit askoren egoerak pilatzeko zirkuitu elektronikoak[11] ditu (milioika bit gordetzeko ahalmena du). Baina memoriaren ahalmena hobeto kudeatzeko, bit guzti horiek taldetxotan antolatzen dira, zelula edo bit-multzo independente horiek gelaska izenez ezagutzen dira.

Gelaska guztiek alde bi izango dituzte, batetik bere luzera (zenbat biten egoerak gorde ditzakeen neurria) ordenadore jakin batean gelaska guztiek luzera berdina izango dute; eta bestetik gelaska izendatzeko helbidea (gelaska bakoitzak bere helbidea izango du, eta aldaezina da). Ikus irudi hau, non zortzi gelaska dituen memoria agertzen den:



[11] E3 eranskinean memori bit bat nola eraikitzen den azaltzen da. Bit bat ataka logikoetan oinarritzen den zirkuitu sekuentziala da.

Kontura gaitezen memoriaren gelaskak helbideen arabera ordenaturik daudela, eta ukaezina da memoriak buzoi edo posta-kutxatila multzo baten antza duela. Memoriari buruzko sarrera egin ondoren azal dezagun orain bere osagai funtzionalak zeintzuk diren.

3.2.1.1 Memori taula

Memoriaren gelaska guztien multzoari memori taula deritzo. Aurreko irudiak erakutsi digunez memoriaren gelaskak bulego edo etxe bateko posta-kutxatilak bezala antolatzen dira, non gordetako informazioa posta-kutxatilaren edukia den eta posta-kutxatilaren helbideak bere kokapena adierazten duen.

Lehenago esan dugunez, memoriaren oinarrizko elementua bi egoera onartzen dituen zirkuitu elektronikoa da eta bit bat biltegitzeko ahalmena duela, baina bit bat unitate txikiegia delako taldetan antolatzen dira. Konputagailu batek helbidera dezakeen memoria zati txikiena horietako bit talde bat izango da eta horrek *hitz* izen berezia dauka informatikarien artean.

Ordenadorearen biten taldetxoeek hitzak osatzen dituzte, baina ez dugu esan zenbat bit sartzen diren hitz batean. Hitz baten bit bopuruari *hitz-luzera* deritzo eta 2ren potentzia izaten da (8, 16 edo 32), eta datu-busak dituen hari-kopuruarekin linealki erlazionaturik dago. Nolabait esateko eta posta-kutxatilen irudiari atzikiz, gelaska batek duen kapazitatea litzateke hitz-luzera. Gelasketan gordetzen den informazioa, nola ez, kode bitarrean adierazita dagoela ez ahaztu, eta ordenadorearen jardueran gelaska batek gordetzen duena alda daiteke

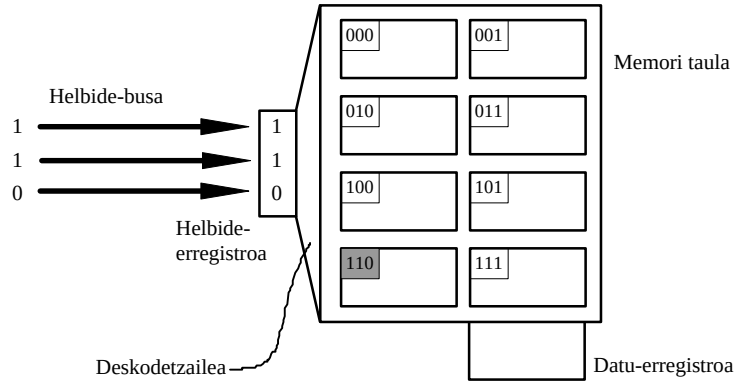
Hitz-luzerako informazio bat gordetzeko gaitasuna duen gelaska batek bigarren ezaugarri bat ere badu: gelaska hori, besteetatik bereiziz, identifikatuko duen helbidea. Helbide bat indize bat besterik ez da, konputagailuan informazio bati dagokion kokalekuaren adierazpena dena, eta dagoenekoz ezaguna zaigun kode bitarrean emanik dago. Informazio bat memoriatik eskuratzeko, dagokion helbidea zehaztu egin behar da. Lehenago esan dugunez, informazioa kode bitarrean gordeko da eta helbideak ere kode bitarrean daude. Denboran zehar gelaska batean gordetzen den datu kodetua desberdina izan daiteke, baina gelaska horren (eta edozeinaren) helbidea iraunkorra eta aldaezina da.

3.2.1.2 Helbide-erregistroa

Konputagailuaren oinarrizko plakaren zehar barreiaturik erregistroak deitzen diren memoria bereziak daude, ordenadoreko unitate desberdinen artean transferentziak egiteko direnez busetara loturik daude. Erregistroak kontzeptuelki, ezagutzen ditugun memori taulako gelaskak bezala dira (informazioa biltegitzeko edukiera batekin eta erregistroa identifikatuko duen helbide batekin), erregistroak memori gelaskak baino bizkorragoak dira baina haiek baino askoz gutxiago dira garestiak direlako.

Memoriako helbide-erregistroa **3.2.2.2** puntuan azalduko dugun datu-erregistrora konektaturik dago. Helbide-erregistroak gordetzen duen informazioak memorian egingo den hurrengo eragiketa zein gelaskatan gertatuko den adierazten du.

Helbide-erregistroaren luzera, hots, barnera dezakeen bit-kopuru maximoa, memori taulan dagoen gelaska-kopuruarekin exponentzilaki erlazionaturik dago. Hurrengo orrialdeko irudian eskala txikiko konputagailu baten memoria agertzen da eta bertan ikus daitekeenez, memori taula osatzen duten gelaskak 8 direlako eta helbideak kode bitarrean adierazten direlako 3 bit beharko dira gelaska guztiak independenteki erreferentziatzeko ($8=2^3$). Helbide-erregistroaren eta memori taularen arteko erlazioa ondokoa da: helbide-erregistroak n bit baditu helbidera daitekeen gelaska-kopuru maximoa 2^n da.

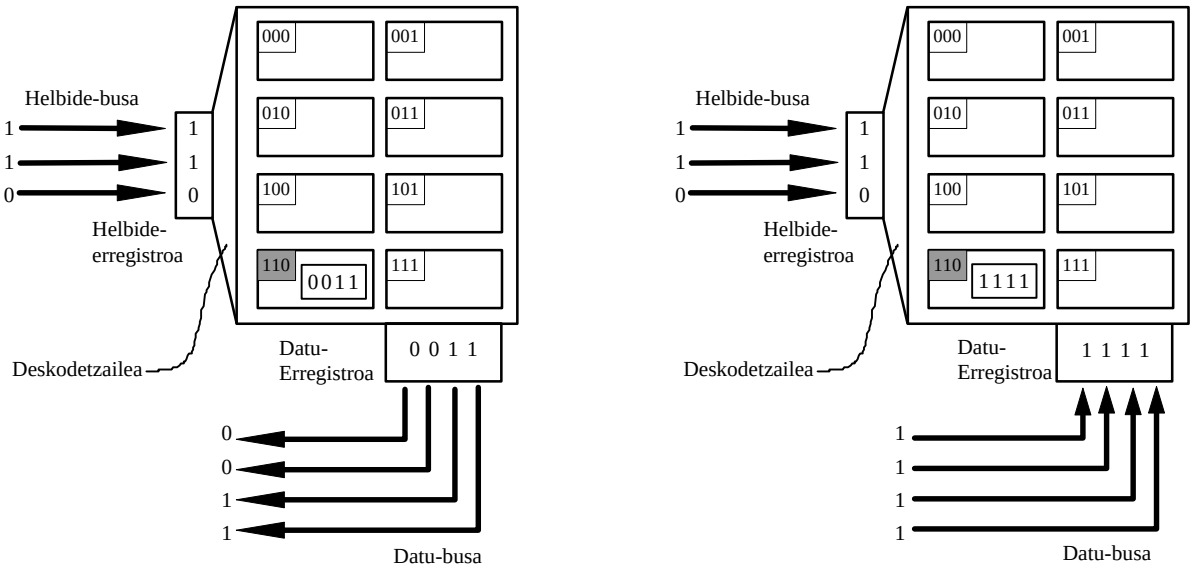


Irudi honetan, memoriaren beste osagai batzuk agertu arren, azpimarratuko dugun kontzeptua helbide-erregistroaren eta memori taularen artean dagoen erlazioa izango da. Ikusten denez helbide-busak 3 hari dituela suposatu dugu, horien bitartez kode bitarrean adieraz daitezkeen gelaska-kopuru maximoa $8=2^3$ dira, bakoitzari dagokion helbidea hau delarik: 000, 001, 010, 011, 100, 101, 110 eta 111.

Helbide-busa erabiliz konputagailuaren Kontrol-unitateak helbide bat bidal dezake memorirantz (adibidez 110 helbidea iristen da helbide-erregistrora), esan dugunez helbide-erregistroak gordetzen duen informazio horrek memorian egingo den hurrengo eragiketa zein gelaskatan gertatuko den adierazten du, beraz hurrengo irakurketa/idazketa operazioan 110 helbidea duen gelaskak hartuko du parte.

3.2.1.3 Datu-erregistroa

Datu-busera konektaturik dago eta memoriako atea da datu-erregistroa, hemendik iragaten da memorian idatziko den datua eta memoriatek irakurtzen dena bertatik iragaten da ere. Datu-erregistroak duen bit-kopurua gelaska batena da, hau da ordenadorearen memori hitzak duen luzeraren berdina izango da datu-erregistroaren bit-kopurua. Behean agertzen diren irudietan datu-erregistroak 4 bit gordetzeko gaitasuna izango du.



Memoriako eragiketak ikasiko ditugunean berriro errepikatuko dugu eta hobeto ulertuko da, baina goazen aurreko irudi bi horien gainean zerbait esatera.

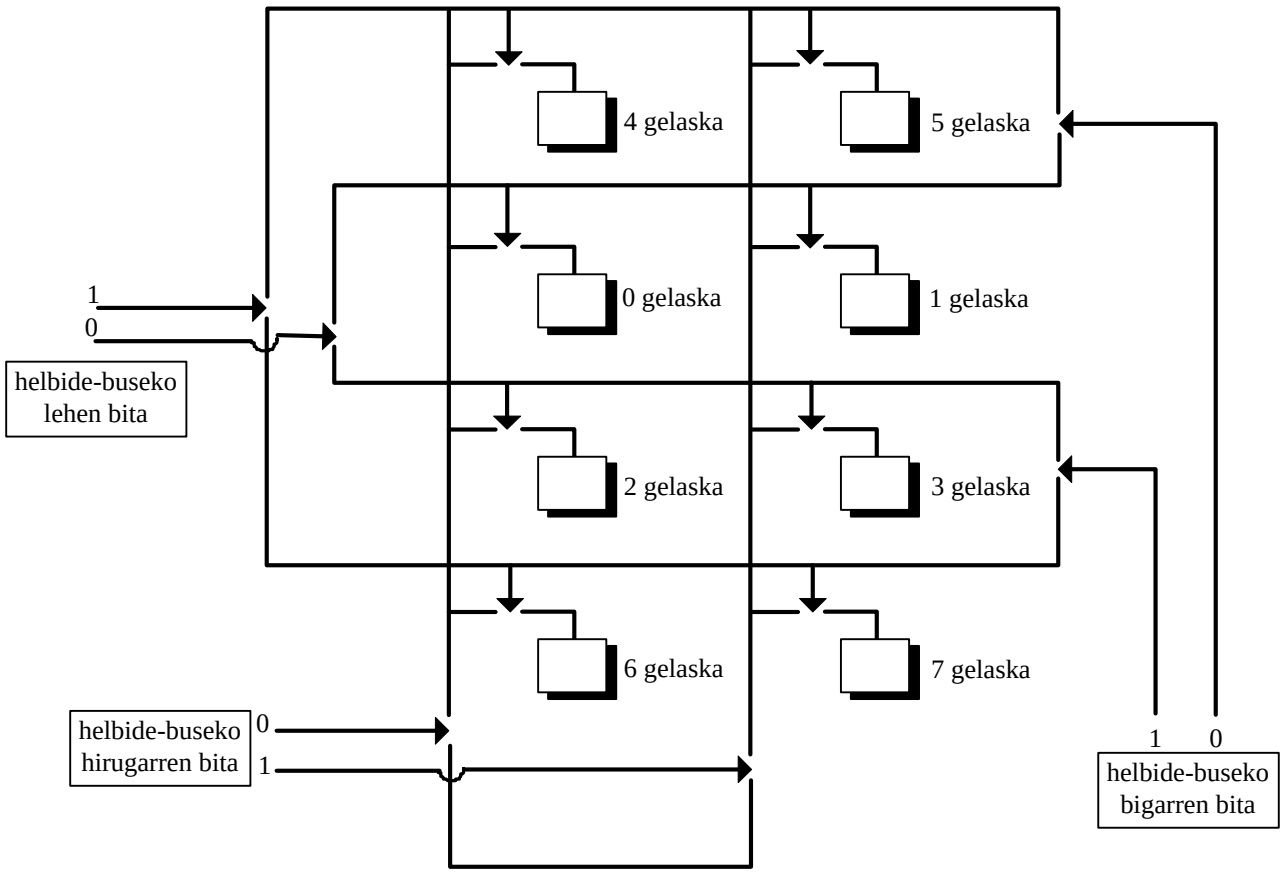
Ezkerreko irudian, helbide-erregistroak duen 110 edukia aintzat harturik, irakurketa bat egin da, laster esplikatuko dugun *Deskodetzaileak* helbide horretako gelaska aktibatzen du. Ondoren gelaskak duen 0011 datua, datu-erregistroaren zehar iraganez, datu-busean jar daiteke eta hemendik PUZeko beste atal batera abia daiteke.

Eskuineko irudian idazketa operazioa burutu da 110 gelaskan. Kasu honetan memorian gorde behar den 1111 informazioa datu-busetik dator datu-erregistroraino eta hortik aktibaturik dagoen gelaskara joango da.

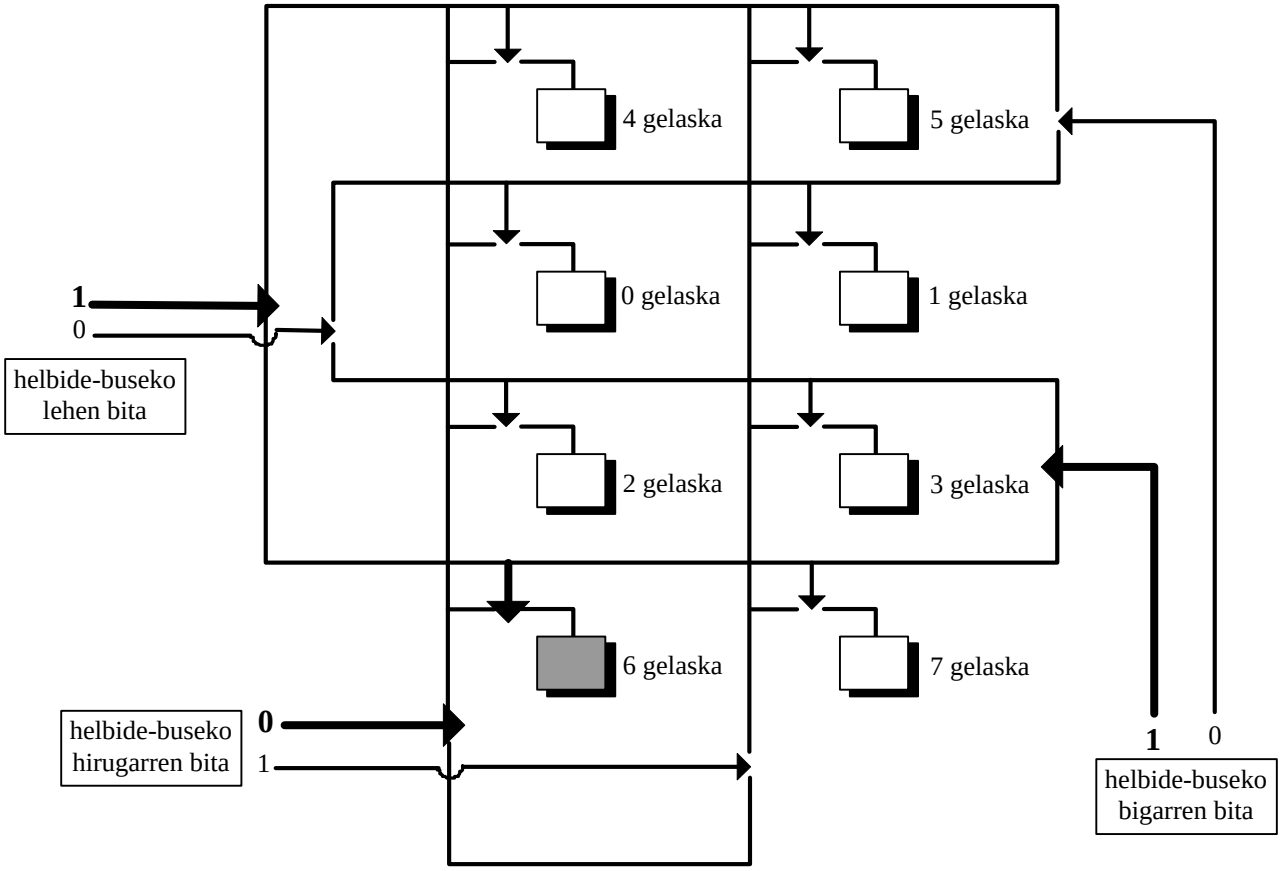
3.2.1.4 Deskodetzailea

Memoriako deskodetzailearen lana helbide-erregistroaren edukiaren araberako memori taulako gelaska aktibatzea da, horretarako kode bitarrean dagoen helbidetik abiatuta korronte elektriko egokiak sortuko ditu.

Hiru hariko helbide-busetik heltzen den informazioa honelaxe kudeatzen du deskodetzaileak. Helbide-busaren hari bakoitzak seinale bakarra eraman dezake, seinalearen balioa “garaia” denean 1koa, eta seinalearen balioa “baxua” denean 0a. Lehen bitak duen balioaren arabera goi-ezkerreko bi etengailuetatik bat itxita egongo da bestea irekita dagoen bitartean; bigarren bitaren balioak eskuineko bi etengailuen egoera emango digu, eta amaitzeko, hirugarren bitaren balioaren arabera behe-ezkerreko bi etengailuetatik bat itxi eta bestea ireki egiten da:



Esate baterako 110 helbiderako, deskodeitzailearen konexioak hurrengo irudian ikusten dira. Lehen 1-ak aktibatzen dituen gelaskak 4, 5, 6 eta 7 dira; helbideko bigarren bitak 1 balio du eta 2, 3, 6 eta 7 gelaskak pizten ditu, eta azkeneko 0-ari esker 4, 0, 2 eta 6 gelaskak aktibatuko dira. 4-5-6-7, 2-3-6-7 eta 4-0-2-6 taldeetan, guztietan, dagoen gelaska bakarra 6-a denez horixe izango da aktibatuko dena:



3.2.1.5 Memoriako eragiketak

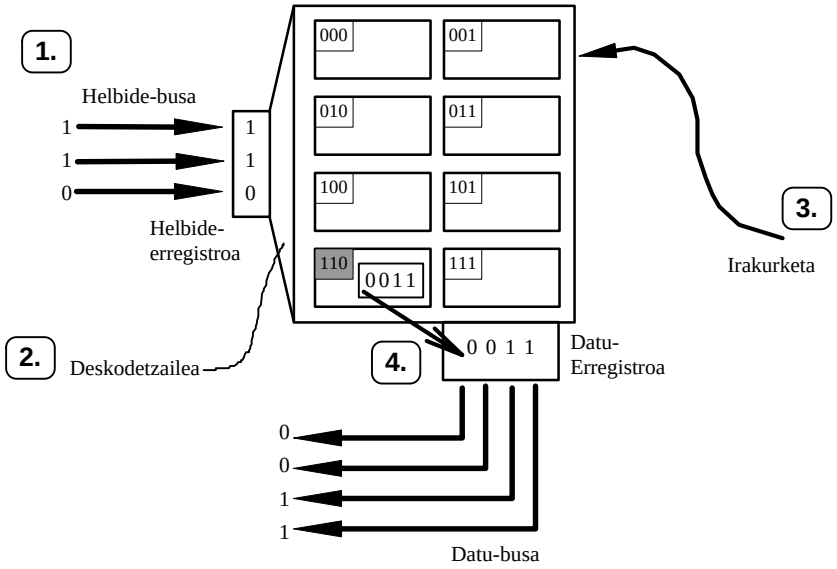
Ordenadorearen memoriak bi eragiketa onartzen ditu: irakurketa (memoriako gelaska bateko informazioa eskuratzea) eta idazketa (memoriako gelaska batean datu bat gordetzea). Bi eragiketak burutzeko, irakurri edo idatzi nahi den datuaz gain erabiliko den gelaska zein izango den adierazi beharra dago (gelaska bere helbidearen bitartez erreferentziatzen da).

3.2.1.5.1 Irakurketa

Demagun zortzi gelaskako memorian irakurketa bat egin behar dela, 110 helbidea duen gelaskatik hain zuzen ere. Irakurketa lau urratsetan betetzen da:

1. Helbide-erregistroan memoriako hurrengo eragiketa zein gelaskarekin egingo den gordeta dago (gelaskaren helbidea biltegitu baita), helbide-erregistrora konektaturik dagoen helbide-busetik zehar iritsi da informazio hori. Irudian 110 helbidea.

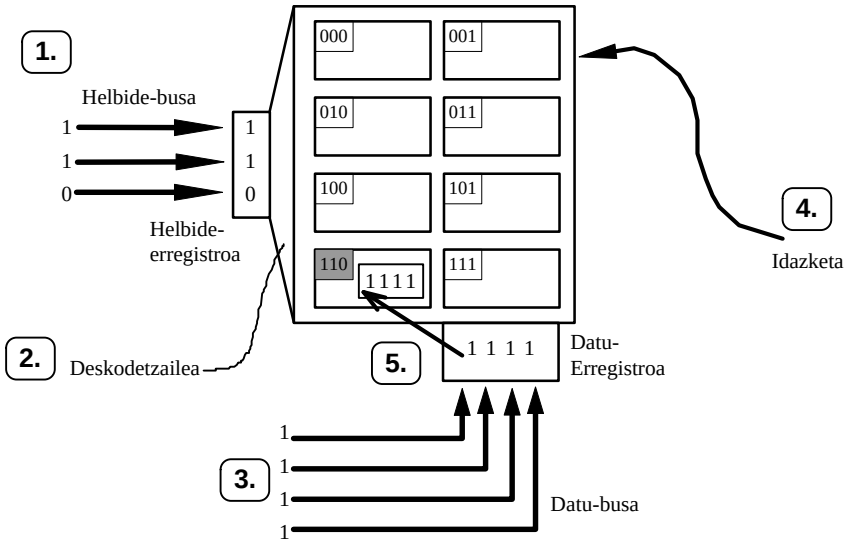
2. Helbide-erregistroko edukiaren arabera deskodetzaileak gelaska bakarra aktibatuko du elektrikoki. Irudian 110 helbidea duen behe-ezkerreko gelaska.
3. Ordenadorearen Kontrol-unitateak irakurketa bat burutu behar dela adieraziko dio memoriari, horretarako kontrol-busa erabiliko du.
4. Operazioa irakurketa delako, aktibaturik dagoen gelaskako edukiaren kopia bat datu-erregistrora transferitzen da. Datu-erregistroa datu-busera konektaturik dagoenez, bere edukia ordenadoreko beste edozein tokitara joateko prest egongo da. Irudian 0011 informazioa kanporatzen da memoriatik.



Irakurketa egin eta gero memoriako gelaskan zegoena aldaketarik gabe mantentzen da, eta informazio hori datu-busaren zehar konputagailuaren beste unitatetara joan daiteke.

3.2.1.5.2 Idazketa

Idazketa egitean, memoriatik kanpo dagoen informazioren bat memoriako gelaska batean gordeko da. Idazketa operazioan gelaskan zegoena galdu egiten da informazio berriagatik ordezkaturik geratzen delako, urratsak bost dira:



1. Helbide-erregistroan memoriako hurrengo eragiketa zein gelaskarekin egingo den gordeta dago (gelaskaren helbidea biltegitu baita), helbide-erregistrora konektaturik dagoen helbide-busetik zehar iritsi da informazio hori. Irudian 110 helbidea.
2. Helbide-erregistroko edukiaren arabera deskodetzaileak gelaska bakarra aktibatuko du elektrikoki. Irudian 110 helbidea duen behe-ezkerreko gelaska.
3. Memorian gorde nahi den informazioa datu-erregistrora iristen da, datu-busaren bidez. Irudian 1111 informazioa.
4. Ordenadorearen Kontrol-unitateak idazketa bat burutu behar dela adieraziko dio memoriari, horretarako kontrol-busa erabiliko du.
5. Operazioa idazketa delako, datu-erregistroan dagoena aktibatu den gelaskara transferitzen da. Idazketa aurretik 110 helbidea duen gelaskan 0011 datua zegoen, eta idazketa ondoren datu-erregistrotik datorrenak ordezkatzen du, beraz, edukin zaharra galdu egiten da idazketa operazioan. Irudian 1111 informazioak 0011 datua ordezkatzen du 110 helbidea duen behe-ezkerreko gelaskan.

3.2.1.6 Memoria motak

Memoriak materiale erdieroaletan fabrikatzen dira eta zirkuitu integratu bezala saltzen dira, bi motako memoriak daude: RAM memoria edo memoria bizi eta ROM memoria edo memoria hil.

3.2.1.6.1 RAM memoria

RAM memoriari (*Random Access Memory*, Ausazko Atzipeneko Memoria[12]) memoria bizi esaten zaio eta bertan idatz eta irakur daiteke programa baten egikaritzapenaren bitartean. Memoria hau exekutatzen ari den programaren instrukzioak biltegitzeko erabiltzen da, eta programak behar dituen datuak eta emaitzak biltegitzen dira ere.

RAM memoria iheskorra denez duen informazioa mantentzeko konputagailua piztuta egongo da, horregatik behar izaten da bigarren mailako memoria edo memoria lagungarria (**3.2.6 Memoria masiboa** izeneko puntuan garatuko dugu).

3.2.1.6.2 ROM memoria

ROM memoria, nolabait esateko RAM memoriari kontrajartzen zaio eta horregatik memori hil esaten zaio ere. ROM memoria (*Read-only Memory*) izenak adierazten duenez, ezin daiteke erabili gure programak eta datuak gordetzeko, mikroprozesadorearen instrukzioak eta konputagailuaren oinarrizko eginkizunak betetzeko programak grabatzen dira ROM memorian.

Atzipena metodoari begira ROM memoriak, RAM memoriak bezala ausazko atzipena du, baina ROM memoriaren ezaugarririk garrantzitsuena iraunkorra dela da, hots, bere informazioa ez da galtzen konputagailua elikatzen duen korronte elektrikoa joatean; ROM memoria fabrikatzen den momentuan idazten zaio informazioa.

[12] Atzipen aleatorioa edo ausazko atzipena datuak idatzi edo irakurtzeko metodoa da, datuak helbide edo gako baten bidez zuzenean aurkitzen dira, aurreko datuak irakurri beharrik gabe. Ausazko atzipena atzipen sekuentzialari kontrajartzen zaio zeinek informazio bat lortzeko bere aurretik dauden guztiak irakurri behar dituen.

3.2.2 Bus konektoreak

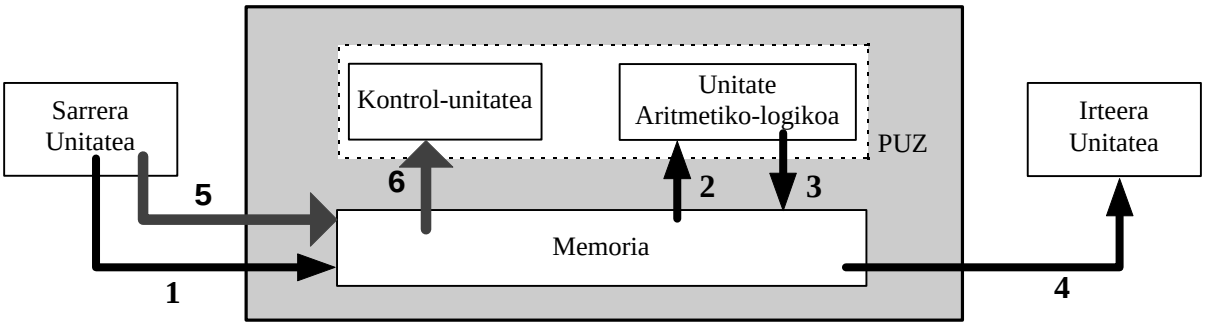
Konputagailuaren mikroprozesadorea eta gainerako txipen arteko lotura, oinarrizko plakan inprimaturik dauden zirkitu metalikoen bitartez lortzen da. Zirkuitu horiek paraleloan antolatuta dauden hari eroaleak dira eta *bus* izenez ezagutzen dira. Hari bakoitzak seinale digital bakar bat garraia dezake une jakin batean, hau da hari batek korrontea garraiatzean bere balioa muga batetik gora ala behera ote dagoen interesatzen zaio ordenadoreari, horrela egoera bi egongo dira: seinale elektrikoa mugatik behera dagoenean (0 logikoa izango da), ala seinale elektrikoa mugatik gora denean (1 logikoa izango da).

Dakigunez busak hiru dira eta horietatik transmititzen den informazioaren arabera sailkatzen dira:

1. Datu-bus
2. Helbide-bus
3. Kontrol-bus

3.2.2.1 Datu-busa

Ordenadore baten osagai desberdinen artean *datuak transmititzen* dituen busa. Une honetan *datuak* transmititzen direla esaten dugunean, bi kontzeptutaz ari gara. Sarrera-unitate batetik memoriara doan informazioa datua da, edo memoriatik ALU-ra eta ALU-tik memoriara doana, edo memoriatik irteera-unitatera doana (**3.2** puntuko eskeman **1**, **2**, **3** eta **4** geziak). Baina sarrera-unitate batetik memoriara doan instrukzio multzoa informazioa da ere (**3.2** puntuko eskeman **5** gezia), eta memoriatik Kontrol-unitatera doan instrukzioa ere datu-busaren bitartez transmititzen da (**3.2** puntuko eskeman **6** gezia).



Datu-busaren hari-kopurua garrantzi handiko parametroa da mikroprozesadore baten diseinua egiten denerako, konputagailuaren *hitza* izango da datu-busaren hari-kopurua eta memori gelaska bakoitzaren bit-kopuruarekin bat dator. Hasierako mikroak 8 bitekoak ziren (Motorola etxeko MP6800, INTEL etxeko MP8085, ZILOG etxeko Z80, etab.), geroago 16 biteko mikroen belaunaldia etorri zen (Motoralako MP68000 edo INTEL-eko MP8086), eta beranduago 32 biteko mikroak plazaratu ziren (Motorolako MP80386 eta INTEL-eko MP80386 eta MP80486).

Memoria erreferentzia bezala hartuta datu-busak norabide bikoitza duela esan daiteke. Izan ere, informazio bat memoria barrura joan daiteke (idazketa operazioa gogoratu nola datu-busa errepresentatzen zuten geziek memoriarantz zuzentzen ziren); baina modu beretsuan informazioa memoriatik irten daiteke ere (irakurketa operazioan berriz, datu-busa errepresentatzen zuten gezien norantza memoriatik kanporakoa zen).

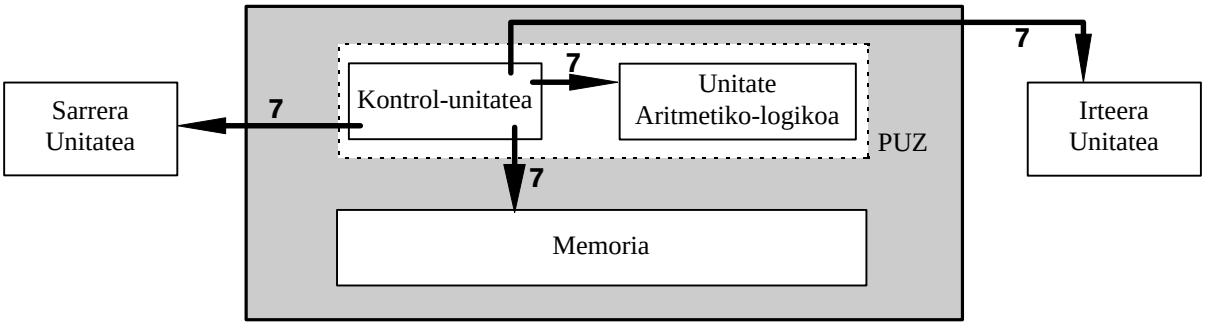
3.2.2.2 Helbide-busa

Helbide-busa helbideak transmititzeko erabiltzen den busa da. Helbide-busak duen hari-kopurua memoriaren tamainarekin lotuta dago formula honen bitartez: $G=2^H$. Non G memori taularen gelaska-kopurua den eta h helbide-busak dituen hari-kopurua den.

Datu-busa azaldu dugun bezala, memoria erreferentzia izanik helbide-busak norabide bakarra duela esan daiteke. Izan ere, idazketa zein irakurketan memoriako helbide-erregistrora iristen dira helbideak eta ondorioz helbide-busa errepresentatzen duten geziek memoriarantz zuzentzen dira eragiketa bietan. Beraz, baiezta daiteke helbide busetik transmititzen den informazioaren norantza bikoitza izan ezik bakarra dela.

3.2.2.3 Kontrol-busa

Kontrol-unitateak konputagailuaren gainerako organo guztiak gainbegiratzen ditu (**3.2** puntuko eskeman **7** geziak), eta horretarako kontrol-busa darabil.



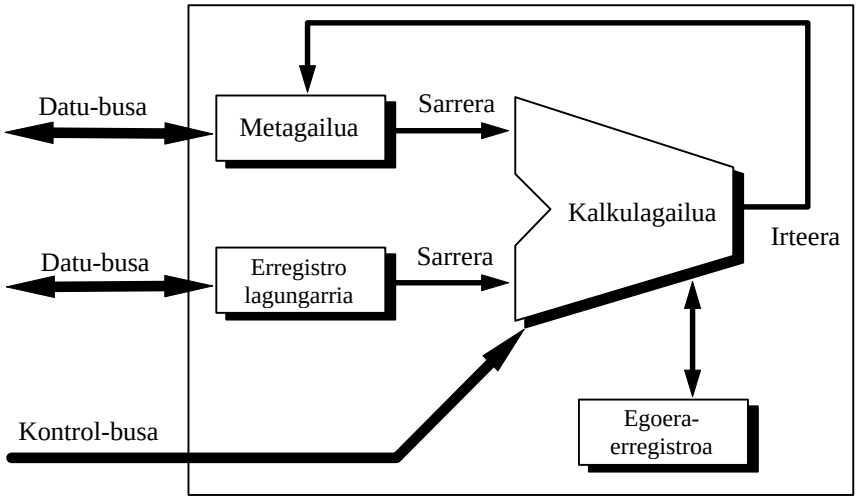
Datu-busa eta helbide-busa hari elektriko multzo paraleloak baldin badira kontrol-busak ez du itxura hori, dituen hariak dedikatuak dira eta guztiek kontrol-seinaleak transmititu arren bakoitzaren zeregina espezifikoa da. Adibidez, memorian burutuko den hurrengo operazioa idazketa ala irakurketa ote den jakiteko kontrol-busaren hari berezien bitartez garraiatzen da seinalea.

3.2.3 Unitate Aritmetiko-logikoa

Nolabait esateko, kalkuluak burutzen dituen unitatea unitate operatzailea da. Izan ere, datuak manipulatzeko zirkuitu elektronikoak dira eta ordenadoreak egin ditzakeen operazioak aritmetikoak (batuketak eta kenketak) eta logikoak (konparaketak) direnez, kalkuluak burutzen dituen elementu honi *Unitate Aritmetiko-logikoa* esaten zaio.

Esan den bezala, zirkuitu elektronikoak transistorean oinarritzen dira (ikus kapitulu honen bukaerako 1. eranskina), eta duten funtzionamendua ondokoa da: zirkuitu elektroniko batek informazio bat (korronte elektrikoa bezala) jasotzeko gaitasuna du, eta seinale elektrikoa aldatuz edo aldatu gabe beste zirkuitu bati transmititzen dio. Sistema digitaletan zirkuitu elektronikoek transmititzen dituzten seinale elektrikoak tentsio altuak ala baxuak dira (zeroak ala batekoak) eta zirkuitu digitalen bitartez aritmetika bitarra nola egiten den 2. eranskinean ematen da.

Mikroprozesadorearen kalkulagailua den UAL unitateari dagokion eskema jarraian erakusten da. Bertan zirkuitu kalkulagailu bat ikusten da (batuketa eta kenketa zirkuitu digitalez nola egiten den kapitulu honetako 2. eranskinean ikus daitekeela gogoratzen dugu), eta zenbait erregistrotara lotuta dago:



Zirkuitu kalkulagailuak bi sarrera izanik irteera bat du, denak erregistroetara loturik. Sarreratik heltzen zaizkion datuak datu-busaren bidez etorriko dira, eragiketa burutu aurretik erregistro horietan aurkitzen diren datuak eragiketaren eragigaiak dira. Eragiketaren emaitza behin lortu ondoren sarrerako erregistro batean gordetzen da, antolaketa honek eragiketa kateatuak egitea ahalbidetzen du (emaitza biltzen duen erregistroari *metagailu* deritzo, ikus `\tp70\03\BIDERKA.PAS` programa).

Unitate Aritmetiko-logikoarekin lotuta *Egoera-erregistroa* dago, honek burutu den azken eragiketari buruzko informazioa jasotzten du (gainezkatzea, zeroa, negatiboa, ...). Egoera-erregistroko bitak letraz identifikatzen dira mikro gehienetan, esate baterako azken eragiketaren emaitza metagailuaren kapazitatea baino handiago bada C bitak 1 balioko du, edo azken eragiketaren emaitza zero izan bada Z bitak 1 balioko du, edo azken emaitza negatiboa izan bada N bita 1 izango da.

3.2.4 Kontrol-unitatea

Konputagailu txikietan mikroprozesadore deitzen den txip bakar batean integraturik daude Unitate Aritmetiko-logikoa eta Kontrol-unitatea. Unitate Aritmetiko-logikoak datuak manipulatzen dituen zirkuituak dituela ikusi dugu; Kontrol-unitateak berriz ordenadorearen akzioak koordinatzen dituen zirkuitu erraldoia da, horretarako kontrol-seinaleak sortuko ditu.

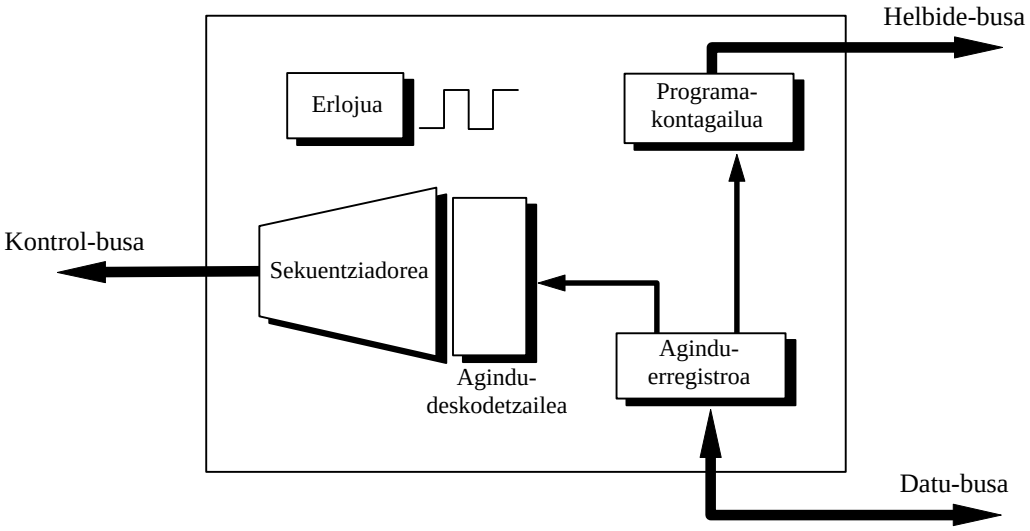
Kontrol-unitateak dituen helburu nagusiak hiru dira, hona hemen:

1. Programaren instrukzioak banan-banan memoriatik jaso, interpretatu eta exekutarazi.
2. PUZ eta memoria, eta, PUZ eta periferikoen artean dagoen datuen transferentzia kontrolatu.
3. Periferikoek igorritako zerbitzu-mezuak erantzun.

Ordenadore batek “uler” ditzakeen aginduak gutxi dira eta agindu bakoitza makinaren zirkuitu zehatz batekin loturik dagoela onar daiteke. Ordenadoreak ezarririk dituen zirkuituen arabera, makinak berezko lengoaia definiturik izango du. Era digitalean emanik dagoen lengoaia hori konputagailuaren *makina-lengoaia* deitzen da (**1.4.1** puntua gogoratu).

3.2.4.1 Kontrol-unitatearen osagaiak

Kontrol-unitatearen eskeman oinarrituta bere osagaiak deskribatuko ditugu orain, eta euren arteko erlazioa hurrengo puntuan (**3.2.4.2 Instrukzio baten exekuzioa** izenekoan) aipatuko dugu.



Programa-kontagailua: Programa-kontagailua edo Programaren Kontagailua erregistro bat da. Programa bat aginduz osaturik dago eta egikaritzen denean aginduak banaka exekutatuz egiten da, programa-kontagailu delako erregistro honek gordetzen duena egikarituko den hurrengo aginduaren helbidea da.

Programa-kontagailuak helbide-busera konektaturik dago eta dituen bit-kopurua helbide-busaren hari-kopurua bera da.

Agindu-erregistroa: Programa bat aginduz osaturik dagoela esan dugu eta memorian aurkitzen dela dakigu. Programa egikaritzen denean aginduak banaka memoriatik Kontrol-unitatera ekartzen dira datu-busaren bidez, memoriatik datorren agindua Agindu-erregistroan kokatzen da eta bertan mantentzen da exekutatua izan arte.

Agindu-erregistroak exekutatzen ari den agindua gordetzen du.

Agindu-deskodetzailea: Exekutatu behar den agindua ezagutu eta zehazten du deskodetzaile honek. Konputagailuak definiturik dituen agindu sortatik zein den identifikatu eta informazio hori sekuentziadoreari pasatzen dio.

Sekuentziadorea: Zirkuitu honek kontrol-seinaleak sortzen ditu, eta erloju-ziklo bakoitzeko exekutatu behar den aginduarekin loturik dauden erregistro eta zirkuituei bidaltzen dizkie, horretarako kontrol-busa erabiltzen du.

Erlojua: Sistema osoaren seinaleen kointzidentzia arautzeko, sinkronismo seinalea sortzen duen gailua da erlojua. Erloju baten seinalea pultsu errepikakorrak dira, eta seinalearen periodoari erloju-ziklo esaten zaio. Erloju-ziklo batean eginkizun asko burutzen ditu ordenadore batek eta segundo batean miloika[13] ziklo dagoenez konputagailuen bizkortasuna neurtzeko parametroa da erlojuaren maiztasuna.

[13] Adibidez 33 Mhz-etako ordenadore batek segundo batean 33.000.000 erloju ziklo ditu.

3.2.4.2 Instrukzio baten exekuzioa, Bilaketa-fasea eta Exekuzio-fasea

Programaren instrukzio edo agindu baten exekuzioa bi fasetan egiten da: Bilaketa-fasea eta Exekuzio-fasea.

Bilaketa-fasearen helburu nagusia instrukzioa memoriatik Kontrol-unitateko agindu-erregistrora ekartzea da. Exekuzio-fasean berriz, agindua exekutatua izan dadin kontrol-seinaleak sortzen dira.

Demagun programa baten agindu bat egikaritu behar duela mikroak, hona hemen urratsak non lehen hiru urratsek bilaketa-fasea osatzen duten eta gainerakoek exekuzio-fasea:

1. Egikaritu behar den instrukzioa, programa osoarekin batera, memoria nagusian aurkitzen da, eta agindu horri dagokion helbidea programa-kontagailu erregistroan gordeta dagoela esan dugu. Beraz, egikaritu behar den instrukzioaren helbidea programa-kontagailutik memoriako helbide-erregistrora transferitzen da helbide-busari esker.

2. Egikaritu behar den instrukzioa gordetzen duen gelaska aktibatuta dagoela Kontrol-unitateak irakurketa bat egitea eskatzen du. Ondorioz, egikaritu beharreko agindua memoriako datu-erregistroan kopiatzen da eta hortik, datu-busaren bitartez, Kontrol-unitateko agindu-erregistrora iritsiko da instrukzioa.

3. Instrukzioa agindu-erregistrora heldu ondoren, programa-kontagailuak gordetzen duen helbidea inkrementatzen da unitate batean; horrela hurrengo[14] instrukzioaren helbidea gordeko du. Dagoenekoz bilaketa-fasea bukatutzat eman daiteke eta exekuzio-faseari hasiera ematen zaio

Aurreko hiru urrats horiek programaren agindu guztiekin ematen dira, hau da, aginduak banan-banan memoriatik Kontrol-unitatera joan behar direnez, bilaketa-fasea agindu guztietan gertatzen da eta hiru urrats horien arabera gertatzen da. Baina exekuzio-fasean betetzen diren urratsak instrukzioaren araberakoak dira eta ez dira beti berdinak izango, hemen eragiketa aritmetiko baten adibidea deskribatzen da, dakusagun:

4. Agindu-erregistroan dagoen instrukzioa agindu-deskodetzailera igarotzen da eta hemen identifikatu egiten da sekuentziadoreari dagokion informazioa emanez. Instrukzioak zein agindu den adierazten du eta horrekin batera behar duen eragigaia non aurkitzen den eta emaitza non gordeko den.

5. Agindua zehaztu ondoren gerta daiteke agindua egikaritzeko operandoren bat behar izatea. Operandoa memorian aurkitzen bada memoriatik irakurriko du eta Unitate Aritmetiko-logikora bidaltzeko, eragigaia erregistroren bat baldin badago bertatik transferituko du Unitate Aritmetiko-logikora.

6. Unitate Aritmetiko-logikoak eragiketa burutzen du emaitza lortuz.

7. Erdietsi den emaitza instrukzioak adierazten duen tokian gorde egiten da. Batzutan memorian biltegitzen da eta beste zenbaitetan erregistro jakin batean gordetzen da. Exekuzio-fasearen azken urratsa betetzean bilaketa-faseari ekiten zaio berriro, izan ere programa-kontagailu erregistroan hurrengo aginduaren helbidea dago eta horri esker programaren sekuentziari jarraitu ahalko zaio.

Bilaketa-fasea eta exekuzio-fasea ulertu eta bereizteko, bi atal idatzi ditugu hirugarren kapitulu honetan ***3.3 KONPUTAGAILU DIDAKTIKO BATEN DISEINUA*** eta ***3.4 PROGRAMEN EXEKUZIOA*** izenekoak.

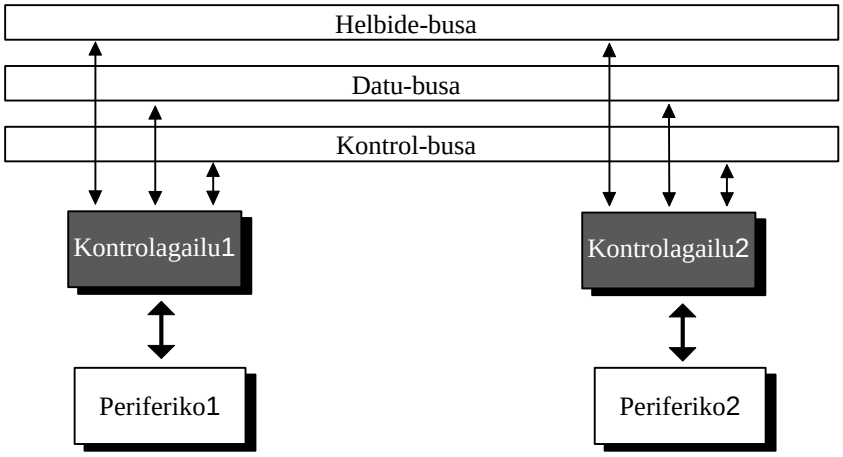
[14] Horrek esan nahi du programa osatzen duten instrukzioak elkar ondoan gordetzen direla memorian.

3.2.5 Periferikoak

Orain arte Prozesatzeko Unitate Zentralak behar dituen datuak eta programak memorian kokaturik daudela suposatu dugu, horra nola iristen diren zehaztu gabe. Baina informazioa kanpotik eman behar zaio ordenadoreari, eta ordenadoreak emaitzak kanporatuko ditu erabiliak izan daitezen.

Programa bat konputagailuan kargatu ondoren programa egikaritzeko bere datu espezifikoak emango ditu erabiltzaileak. Argi dagoenez, erabiltzaileak konputagailuarekin komunikatu behar duela eta horretarako *periferiko* deitzen diren elementuak daudela dakigu. Sarrerako periferikoen bitartez programa eta datuak ordenadorearen memoriara eramaten dira (teklatua, sagua, disko-unitatea, ...); irteerako periferikoei esker prozesaketaren emaitzak jasotzen dira (monitorea, inprimagailua, disko-unitatea, ...).

Sarrera eta irteerako periferikoek sarritan osagai mekanikoak dituzte, eta ondorioz lan egitean Prozesatzeko Unitate Zentralak darabilen abiadura ezin dute jarraitu. Horregatik sarrera-irteera periferikoen eta Prozesatzeko Unitate Zentralaren artean zirkuitu moldatzaileak ezartzen dira, zirkuitu horiek *interfaze zirkuituak* dira eta kontrolatzaileak esaten zaie. Periferiko guztiak sistemaren busetara konektatzen dira kontrolatzaile egokiaren bitartez:



Periferikoek lan egiteko duten abiadura txikia dela eta, PUZ-arekin sinkronizatu gabe daude. Ondorioz, PUZ eta periferikoen artean seinaleak bidaltzen dituzte elkar, horrela sarrera-irteera eragiketaren baten hasiera edo bukaera adierazten dira.

Esandakoaren haritik jarraituta, teklatuaren teklak sakatzean informazioa memoriara iritsi aurretik datuok kontrolagailuan dagoen erregistro batean mantentzen dira eta Kontrol-busaren bitartez datu bat interfazean dagoela adierazten zaio Prozesatzeko Unitate Zentralari. Ordenadoreak memoriatik informazioa irakurtzeko instrukzioak dituen bezala kontrolagailuaren erregistroan dagoena irakurri eta memoriara eraman dezake.

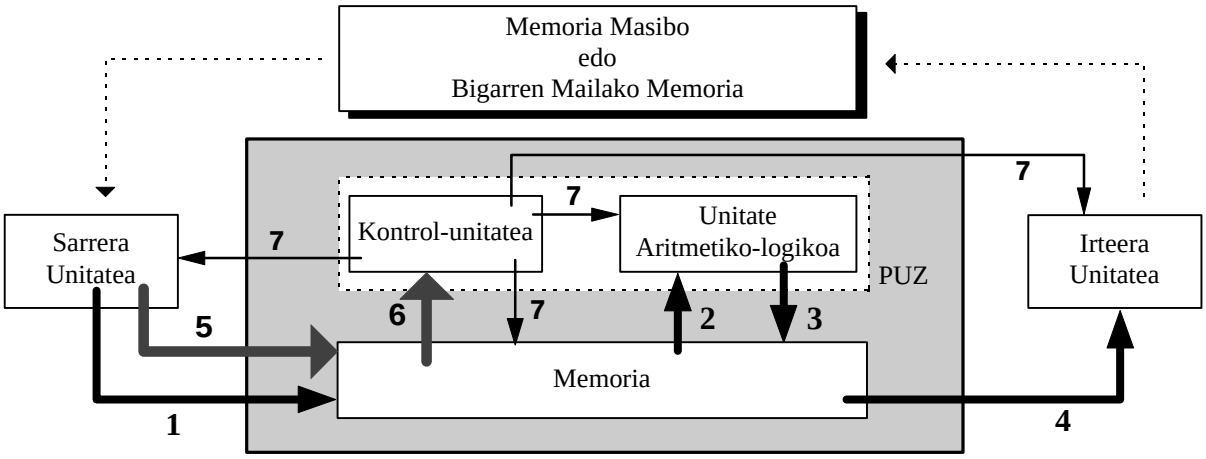
Beraz, periferiko guztiek ordenadorera lotzeko kontrolagailuak behar dituzte. Kontrolagailu bat interfaze zirkuitua da eta duen zeregina bikoitza dela azaldu dugu: periferiko eta ordenadorearen busaren artean tarteko[15] memoriaren lana betetzen du, eta datua sarrerako periferikotik iritsi dela eta PUZ-era sartzeko prest dagoela (edo irteerako periferikora irten dela eta beste datu bat jaso dezakeela) ordenadorearen Kontrol-unitateari adierazi.

[15] Tarteko memoriak buffer bezala ezagutzen dira eta ezaugarri desberdinak dituzten bi unitateren artean datuak iragankorki gordetzeko balio dute. Memoria honen eginkizuna, labur esanda, bi unitateen arteko abiadura-diferentzia konpentsatzea da.

3.2.6 Memoria masiboa

Konputagailuaren RAM memoriak duen eragozpen nabarmenena iheskorra dela da, ordenadorea itzaltzean duen informazioa desagertzen dela alegia. Memoria nagusia hegazkorra delako memoria iraunkorrak erabiltzen dira ere, korrontea kendutakoan informazioa galtzen ez duen memoria izateaz gain memoria nagusiak baino askoz informazio gehiago gorde dezake, horregatik memoria masibo esaten zaio.

Memoria masiboak duen informazioa (programak eta datuak) konputagailuaren kanpoan kokatzen da (disko eta zintetan) eta konputagailuan erabiliak izan daitezen sarrera-irteera unitateen beharra dago. Ikus ezagutzen dugun eskema osatua, non memoria nagusiari kontrajartzen zaion bigarren mailako memoria agertzen den:

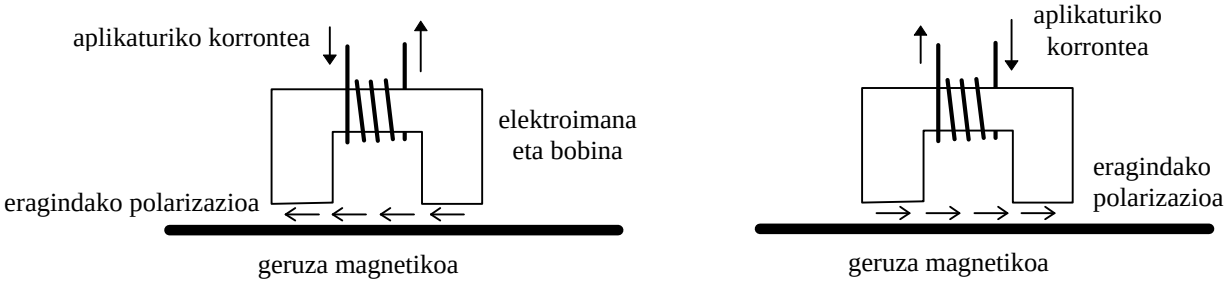


Programak eta datuak iraunkorki gordetzeko zein propietate fisikotan oinarritzen den aintzat harturik, memoria masiboa honela sailka daiteke: memoria magnetikoak eta memoria optikoak.

3.2.6.1 Memoria magnetikoak

Memoria magnetiko arruntenak disketeak eta disko gogorrak dira, baina lehen erabili zirenak zintak izan ziren. Disko eta zinta magnetikoak plastikozko euskarri baten gainean metalezko oxido geruza bat daukate, informazioa biltegitzeko gaitasuna duena.

Disko edo zinta batean informazioa grabatzeko dagokion periferikoa behar da, gailu honek *irakur-idazketarako burua* deituriko elementua du. Irakur-idazketarako burua elektroiman bat da eta korronte elektriko bat aplikatzen zaio idazketa bat egin nahi denean. Korrontea zentzu batean doanean buru azpian dauden molekula magnetikoak zentzu jakin batean polarizatzen (orientatzen) dira, baina elektroimanaren korrontea beste zentzuan doanean polarizazio magnetikoa ere beste zentzuan gertatzen da.



Disko eta zintetan grabaturik dagoen informazioa irakurtzeko gailua eta idazteko erabiltzen dena periferiko bera da baina prozesua, nolabait esateko, aurkakoa da. Informazioa biltegitzen duen euskarria mugitu behar da bertatik datuak edo programak irakurri ahal izateko, dakigunez iman bat mugitzen denean korronte elektrikoa induzitzen da.

Irakurketa bat lortzeko irakur-idazketarako burua disko edo zintaren gainetik pasatzen da, eta molekula magnetikoak iman txikien portaera dutenez korronte elektrikoa sortzen da elektroimanaren bobinan. Molekula magnetikoen polarizazioa alde batekoa bada induzitzen den korronteak zentzu bat izango du, eta molekula magnetikoen polarizazioa aurkakoa bada sortzen den korrontearen zentzua ere aurkakoa izango da.

Hurrengo taulan memoria masibo magnetikoen ezaugarriak erakusten dira:

Zinta	Atzipen sekuentzialeko biltegiak direlako atzipen-denbora handia dute. Merkeak dira eta segurtasun kopiak egiteko erabiltzen dira
Disketea	Disketeak edo disko malguak edukiera txikia dute baina eramangarriak direnez, informazioa konputagailu batetik bestera garraiatzeko balio dute
Disko gogorra	Disko malguak baino askoz edukiera handiagoa eta atzipen-denbora laburragoa du

3.2.6.2 Memoria optikoak

Memoria optikoak[16] magnetikoak baino modernoagoak dira, disko optikoak musikako CD-audio disko bezalakoak dira baina bere informazioa analogikoa izan beharrean digitala da. Disko optikoa CD-ROM izenez ezagutzen da (*Compact Disk Read Only Memory*, Disko Trinkoa Memoria Irakurgarria Bakarrik) bere diametroa 120 mm-koa da, erdian ardatzaren 15 mm-ko zuloa du eta lodiera 1,2 mm-koa da.

Disko magnetikoetan informazioa pista eta sektoreetan antolatzen da baina CD-ROM batean informazioa 5 kilometroko pista espiral bakarrean grabaturik dago zulo txiki bezala. Zuloen sakonera 0,12 mikrakoa da eta guztira 5 Km-ko espiralean 2.000 milioi zulo egon daiteke.

Audio diskoak merkaturatu zirenean fabrikatzaileen artean formatu amankomuna onartu zuten eztabaida gogorrak izan ondoren, horren arabera informazioa irakurtzeko laser-izpia abiadura lineala konstantearekin lan egingo du. Ondorioz, disko trinkoaren biraketa-abiadura aldakorra izango da, irakurtzeko burua zentrutik hurbil dagoenean azkarrago biratuko du abiadura lineala konstantea mantendu behar badu; aldiz irakurtzeko burua diskoaren kanpoko ertzaren ondoan dagoenean biraketa-abiadura txikiagoa izango da.

CD-ROM bateko espiralean grabaturik dagoen zulotxoak informazio digitalaren egoera bat adieraziko du, eta zulotxoaren gabeziak informazio digitalaren beste egoera (batek 0-a eta besteak 1-a). Irakurtzeko buruak laser-izpi bat igortzen du espiralera, argi-izpiaren erreflexioa desberdina da laserren talka zulo batean ala zulorik ez den azalean gertatu bada, sentsore optiko batek jasotzen du laser-izpiak isladatutako argi-kantitatea eta hura neurtuz seinale elektriko egokiak bidaliko ditu irakurketa gauzatuz.

Teknologia magnetikoarekiko CD-ROM disko batek dituen abantaila nagusiak hiru dira:

1. Diskoa fisikoki txikia izanik informazioa biltegitzeko ahalmen handia.
2. Informazioaren iraunkortasuna eremu elektromagnetikoek eragiten ez diotelako.
3. Metatutako informazioaren kostu txikia biteko.

[16] Ikus kapitulu honen bukaerako 6. eranskina (Konputagailuen periferiko optikoei buruzko artikuluak).

3.3 KONPUTAGAILU DIDAKTIKO BATEN DISEINUA

Demagun ordenadore baten diseinua egin behar dugula, horrelako makina bat lortzeko hartu behar diren erabakiak bi mailetan zehazten dira. Lehenik ordenadorearen barne osagaien ezaugarriak zehazten dira, eta ondoren makina horrek “ulertuko” duen lengoaia asmatzen da.

3.3.1 Arkitektura

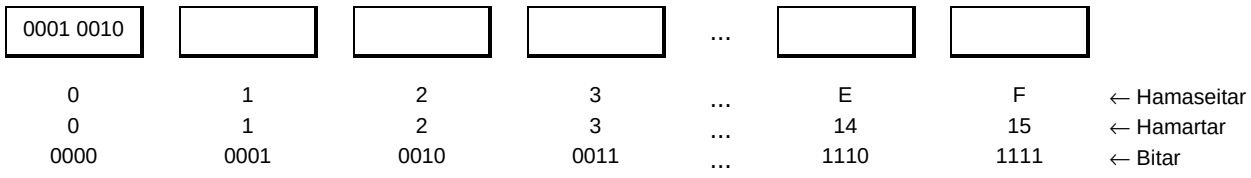
Makinak, erabilpen orokorrerako 16 erregistro izango ditu. Erregistroak identifikatzeko zenbakiak erabil daitezke, adibidez sistema hamartarrean 16 erregistroak identifikatzeko *0*, *1*, *2*, … , *15* zenbakiak aski dira, sistema hamaseitarrean beharko liratekeen zenbakiak berriz ondokoak dira *0*, *1*, *2*, … , *F*.

Makina digitala denez, eta aurreko diseinuaren ondorioz, agindu batean erregistro jakin bat identifikatzeko lau bit beharko dira, nahiz eta guk gutxiago idazteko sistema hamaseitarra erabili:

Erregistroak	Aginduetan	Idaztean
0	0000	0
1	0001	1
...	...	...
15	1111	F

Demagun erregistro bakoitzaren luzera zortzikote bat dela, hona hemen erregistroen irudia eta bakoitzari dagokion identifikadorea (sistema hamaseitar, hamartar eta bitarrean):

Helburu orokorreko 16 erregistro:

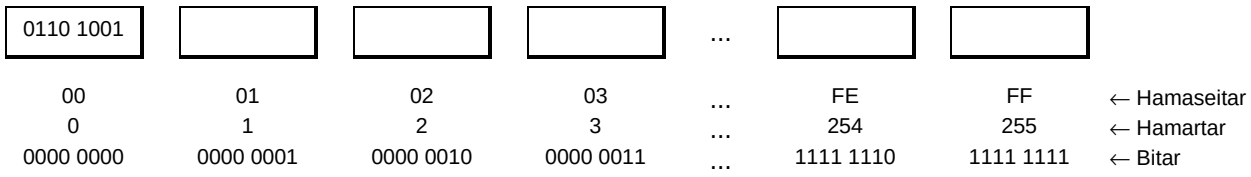


Memoria nagusiaren gelaska-kopurua 256 da, eta gelaska bakoitzaren luzera (erregistroak bezala) zortzi bitekoa da. Gelaskeen helbideak makinan sistema bitarrean emanik egongo dira, eta 256 gelaskak helbideratzeko zortzi biteko txantiloiak beharko dira (euren balioak 0000 0000 eta 1111 1111 artean aldatzen direlarik), helbide horiek identifikatzeko sistema hamartarrean 0 eta 255 artean dauden zenbakiak erabil daitezke, eta sistema hamaseitarraz baliatu ezkero helbideak 00 eta FF artean dauden zenbakien bitartez adieraziko lirateke.

Gelaskak	Aginduetan	Idaztean
0	0000 0000	00
1	0000 0001	01
...	...	...
255	1111 1111	FF

Erregistroetarako lehen egin dugun bezala, gelasken irudi bat egin ezkero honelako zerbait aterako litzaiguke (non, gelaska bakoitzari dagokion helbidea sistema hamaseitar, hamartar eta bitarrean adierazita agertzen den):

Memori taula (256 gelaska):



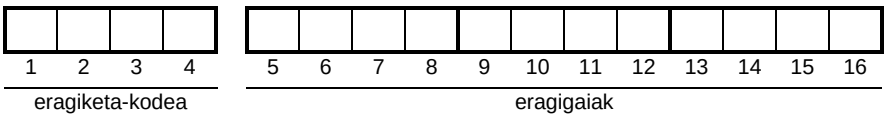
Memorian zenbaki osoak biltegitu ahal izateko 2rako osagarria deituriko adierazpidea erabiltzen dela suposatuko da. Beraz, 0 erregistroak duen edukia 0001 0010 izanik eta kode horrek kopuru oso bat dela jakinik, sistema hamartarrean zenbaki hori 18 izango litzateke[17].

Demagun ere, kopuru errealak biltegitzeko koma higikorra adierazpidean emanik izango direla. Eskema honen arabera:

Horregatik 0 helbidedun gelaskak duen edukia 0110 1001 kopuru erreala dela onarturik, sistema hamartarrean 2'25 zenbakia izango litzateke[18].

3.3.2 Lengoaia

Makina-lengoaiak izango dituen instrukzio bakoitzaren luzera 2 bytekoa izango da. Instruzioaren 16 bitak bi zatitan banaturik daudela uler daitezke, bit horien betebeharra ondokoa delarik: ezkerratik hasita instrukzioaren lehenengo 4 bitek eragiketa-kodea adierazteko balio dute, eta hurrengo 12 bitek eragigaien zatia osatzen dute (ez da helbideratze modua aintzat hartzen).



Instrukzio edo agindu bat harturik, bere eragiketa-kodea ezagutzean hurrengo 12 biten esanahia finkaturik dago (zenbaitetan helbideak izango dira, beste batzuetan, erregistroen identifikadoreak, eta gainerakoetan datuak izango dira).

Konputagailu didaktikoaren berezko aginduak adierazteko eragiketa-kodea 4 bytekoa izanik, konputagailuak gehienez $16=2^4$ instrukzio izan ditzake; baina lengoaia definitzean 12 dira konputagailuaren makina-lengoaia osatzen duten instrukzioak.

Hona hemen makina didaktikoak ezagutzen dituen instrukzioen zerrenda (aginduetan erregistro bat agertzen denean *R*, *S* edo *T* letraz ordezkatzen dugu, erregistroak ez diren kode hamaseitarrak ordezkatzeko *X* eta *Y* letrak erabiltzen[19] dira):

[17] Gogoratu **2.4.2.3 2rako osagarria** puntua, non 0001 0010 = (+)001 0010 = 16+2 = 18

[18] Gogoratu **2.4.3.1 Koma higikorra** puntua, non 0110 1001 = (+)10'01 = 2+0+0+0'25 = 2'25

[19] Instrukzioetan agertuko diren kode hamaseitarrak erregistro identifikadoreak ez direnean, memoriako helbideak edo bestela balioak izango dira.

Eragiketa-kodea	Eragigaia	Deskribapena eta adibidea
0	*000*	Programaren exekuzioa GELDITU.
1	*RXY*	KARGATU *R* erregistroan, memoriako *XY* helbidean dagoen bit multzoa. Adibidez *13A1* instrukzioaren bitartez, memoriaren *A1* helbideko gelaskan dagoena *3* erregistroan jarriko litzateke.
2	*RXY*	KARGATU *R* erregistroan, *XY* bitartez adierazten diren bitak. Adibidez *23A1* instrukzioak, *A1* balioa *3* erregistroan jarriko luke.
3	*RXY*	*R* erregistroan dagoen bit multzoa, memoriako *XY* helbidea dagokion gelaskan BILTEGITU. Adibidez *33A1* instrukzioaren bitartez, memoriaren *A1* helbideko gelaskan *3* erregistroan dagoena gordeko litzateke.
4	*0RS*	*R* erregistroan dagoen bit multzoa *S* erregistrora MUGITU. Adibidez *40F1* instrukzioak, *F* erregistroan dagoenaren kopia bat *1* erregistroan burutuko luke (honek lehenago duena zapalduz).
5	*RST*	*S* eta *T* erregistroetan dauden bit multzoak BATU eta emaitza *R* erregistroan kokatu (bit multzoak 2rako osagarria delako adierazpidean emanik daude). Adibidez *50FE* instrukzioak, *F* eta *E* erregistroetan dauden bit multzoen batuketa osoa lortu eta batura *0* erregistroan gordeko luke.
6	*RST*	*S* eta *T* erregistroetan dauden bit multzoak BATU eta emaitza *R* erregistroan kokatu (bit multzoak koma higikorra delako adierazpidean emanik daudela kontsideratu). Adibidez *6210* instrukzioak, *1* eta *0* erregistroetan dauden bit multzoen batuketa erreala lortu eta batura *2* erregistroan gordeko luke.
7	*RST*	*S* eta *T* erregistroetan dauden bit multzoei OR eragiketa aplikatu eta emaitza *R* erregistroan kokatu. Adibidez *7BAC* instrukzioaren bitartez, *A* eta *C* erregistroetan dauden bit multzoen artean batuketa logikoa burutu eta emaitza *B* erregistroan gordeko litzateke.
8	*RST*	*S* eta *T* erregistroetan dauden bit multzoei AND eragiketa aplikatu eta emaitza *R* erregistroan kokatu. Adibidez *8BDC* instrukzioaren bitartez, *D* eta *C* erregistroetan dauden bit multzoen artean bider-kaketa logikoa burutu eta emaitza *B* erregistroan gordeko litzateke.
9	*RST*	*S* eta *T* erregistroetan dauden bit multzoei XOR eragiketa aplikatu eta emaitza *R* erregistroan kokatu. Adibidez *9BFE* instrukzioaren bitartez, *F* eta *E* erregistroetan dauden bit multzoen artean disjuntzio elkarbaztertzailea burutu eta emaitza *B*-n gordeko litzateke.
A	*R0X*	*R* erregistroan dagoen bit multzoa *X* aldiz eskuinera BIRATU, biraketa bakoitzean bit multzoan ezkerragora dagoen bita eskuinean jartzen da eta besteak posizio bat irabazten dute ezkerretara mugituz. Adibidez *AB05* instrukzioaren bitartez, *B* erregistroan dagoen bit multzoa bost aldiz eskuinera biratuko litzateke.
B	*RXY*	Memoriako *XY* helbidean dagoen instrukziora JAUZI egin, baldin eta *R* erregistroan dagoen bit multzoa eta *0* erregistroan dagoena berdinak badira. Adibidez *B91C* instrukzioak, *9* eta *0* erregistroetan dagoena konparatu eta bit multzo biak berdinak badira exekutatuko den hurrengo instrukzioa memoriako *1C* helbidean dagoena izango da; alderatutako bit multzoak berdinak ez badira exekutatuko den hurrengo instrukzioa txanda sekuentzialki dagokionari izango da.

3.4 PROGRAMEN EXEKUZIOA

Ikus dezagun orain makina didaktikoan programak nola egikarituko liratekeen. Ondoko puntuetan programa labur bi ematen dira, lehenengoaren helburua zenbaki bikote biren baturak lortzea litzateke, eta bigarrenaren zeregina bigizta errepikakor bat antolatzea da.

3.4.1 Zenbakiak batzen

Demagun programa bat ondoko instrukzioz osaturik dagoela, (programaren aginduak, ikus daitekeenez, sistema hamaseitarrean adierazita daude):

Agindua	Azalpena
254C	*4C* zenbakia (0100 1100 sistema bitarrean edo $(+)1x2^6+1x2^3+1x2^2$ = 76 sistema hamartarrean) *5* erregistroan kargatu.
260D	*0D* zenbakia (0000 1101 sistema bitarrean edo $(+)1x2^3+1x2^2+1x2^0$ = 13 sistema hamartarrean) *6* erregistroan kargatu.
5056	*5* eta *6* erregistroan dagoena zenbaki osoak balira batu (76+13=89) eta emaitza *0* erregistroan gorde: 1 1 76 0100 1100 13 0000 1101 89 0101 1001 = $1x2^6+1x2^4+1x2^3+1x2^0$ = = 64 + 16 + 8 + 1 = 89 sistema hamartarrean
306E	*6E* helbidea duen gelaskan, *0* erregistroan dagoena (0101 1001 bitarrean edo 89 hamartarrean) biltegitu.
1A6E	*6E* helbidea duen gelaskan dagoen 0101 1001 bytea hartu eta *A* erregistroan kargatu.
5AA6	*A* eta *6* erregistroan dagoena zenbaki osoak balira batu (89+13=102) eta emaitza *A* erregistroan gorde (zeukan 89 balioa zapalduz): 1 1 1 89 0101 1001 13 0000 1101 102 0110 0110 = $1x2^6+1x2^5+1x2^2+1x2^1$ = = 64 + 32 + 4 + 2 = 102 sistema hamartarrean
3A6E	*6E* helbidea duen gelaskan, *A* erregistroan dagoena (0110 0110 bitarrean edo 102 hamartarrean) biltegitu, ondorioz aurretik zegoen 0101 1001 bytea (89 sistema hamartarrean) galduko da.
0000	Programa gelditu.

Programa aztertzean, batuketa bi egiten direla kontura gaitezke. Lehenengo batuketa erregistro bi datuz kargatu ondoren lortzen da, emaitza gero memoriara eramaten delarik. Bigarren batuketan, eragigaiak erregistro baten eta memoriako gelaska baten edukiak dira. Amaitu aurretik azken emaitza memorian biltegitzen da.

Azal dezagun orain programa hori memorian nola dagoen kokaturik, horretarako jarraian ematen diren taulei so egin. Taula biak, funtsean, berdinak dira batean gelaskeen edukiak eta helbideak sistema hamaseitarrean adierazten dira, eta bestean berriz sistema bitarrean. Memoriaren hitz-luzera 8 bitekoa dela gogoraturik, instrukzio bakoitzeko gelaska bi beharko dira. Suposatuz gure programa 00 helbidetik aurrera biltegiturik dagoela hona hemen memoriaren irudia:

Helbideak	Edukia
00	25
01	4C
02	26
03	0D
04	50
05	56
06	30
07	6E
08	1A
09	6E
0A	5A
0B	A6
0C	3A
0D	6E
0E	00
0F	00

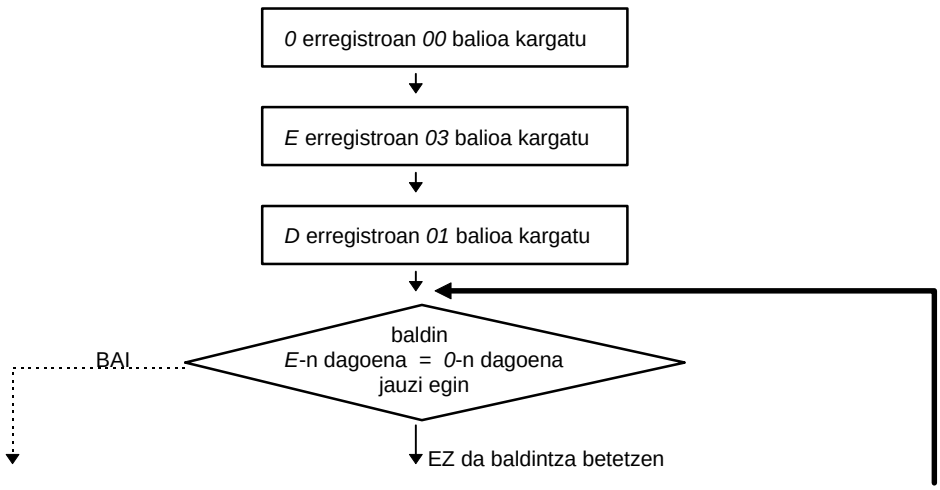
Helbideak	Edukia
0000 0000	0010 0101
0000 0001	0100 1100
0000 0010	0010 0110
0000 0011	0000 1101
0000 0100	0101 0000
0000 0101	0101 0110
0000 0110	0011 0000
0000 0111	0110 1110
0000 1000	0001 1010
0000 1001	0110 1110
0000 1010	0101 1010
0000 1011	1010 0110
0000 1100	0011 1010
0000 1101	0110 1110
0000 1110	0000 0000
0000 1111	0000 0000

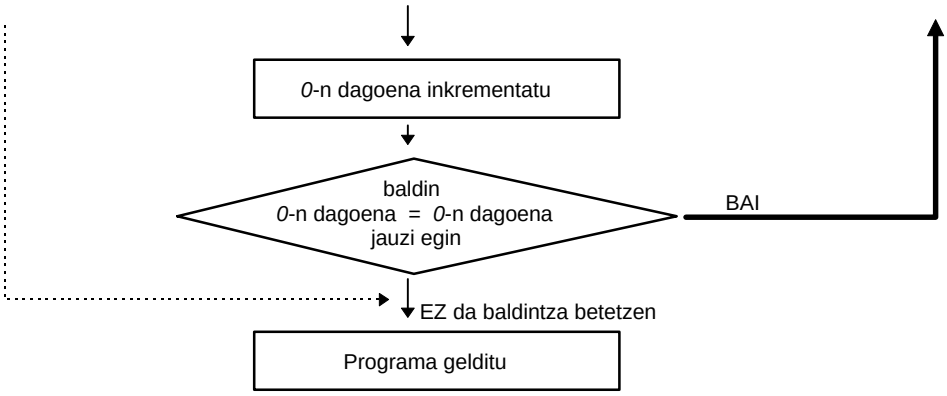
Beraz, *kontrol-unitatean* dagoen *programa-kontagailua* izeneko erregistroan gure programaren lehenengo aginduaren helbidea jarri ezkero (0000 0000 kode bitarrean) instrukzioen exekuzioa hasiko litzateke. Helbide-busaren bitartez, *programa-kontagailuan* aurkitzen den 0000 0000 bytea memoriaren *helbide-erregistrora* garraiatuko litzateke eta irakurketa bat egin ondoren lehenengo instrukzioaren erdia *kontrol-unitatetean* dagoen 16 biteko *Agindu-erregistrora* eramango litzateke (instrukzioaren bigarren zatia modu berean memoriatik hartu eta *Agindu-erregistroaren* informazioa osatuko litzateke). Instrukzioa exekutatzen den bitartean, *programa-kontagailua* delako erregistroan hurrengo aginduaren helbidea edukiko luke (0000 0010 kode bitarrean). Prozesuak aurrera jarraituko luke 0000 instrukzioa *kontrol-unitatera* iritsi eta exekutatu arte.

Ondorioz, adibide-programa hau erabat sekuentziala dela esan dezakegu eta dituen zortzi instrukzioak behin bakarrik (eta bat hurrengoaren ondoan) exekutatzen dira.

3.4.2 Errepikapenak burutzen

Zenbaitetan, **3.4.1** puntuan ikusitako adibide-programan oso nabarmen azaltzen den sekuentzia apur daiteke. Esate baterako, jarraian erakusten den programaren eskeman egiten den bezala:





Programaren eskema ikusi bezain laster sekuentzia haustura bi daudela kontura gaitezke, lehenengo sekuentzia haustura (3-27 orrialdean aurkitzen dena) baldintzatutako jauzi bat da, eta bigarrenean (3-28 orrialdekoan) benetan baldintzarik ez dago eta jauzia beti "gorantz" burutuko litzateke errepikapen[20] prozesu bat eraginez. Goimailako lengoaietan ez dira horrelako jauziak egiten, programak behar duen logika gauzatzeko askoz boteretsuagoak diren `IF`, `WHILE` eta antzeko egiturak daude.

Hauexek lirateke bigarren adibide programari dagozkion instrukzioak:

Helb.	Agindua	Azalpena
A0	*2000*	*0* erregistroan 00 balioa (0 sistema hamartarrean) kargatu.
A2	*2E03*	*E* erregistroan 03 balioa (3 sistema hamartarrean) kargatu.
A4	*2D01*	*D* erregistroan 01 balioa (1 sistema hamartarrean) kargatu.
A6	*BEAC*	*E* erregistroan dagoena *0* erregistroan dagoenarekin konparatu, eta berdinak badira memoriako *AC* helbidean aurkitzen den instrukzioa exekutatuz jarraitu, *E* eta *0* erregistroen edukia ezberdinak badira sekuentzia normalarekin aurrera egin.
A8	*500D*	*0* eta *D* erregistroetan dauden bit multzoak zenbaki osoak balira batu, eta emaitza *0* erregistroan gorde.
AA	*B0A6*	*0* erregistroan dagoena *0* erregistroan dagoenarikin konparatu. Hau beti egia izango denez agindu honek *A6* helbidean aurkitzen den instrukziora jauzia eragiten du.
AC	*0000*	Programa gelditu.

Programaren exekuzioa simulatu nahi izanez gero honelako zerbait izango litzateke, non laukiak *0*, *E* eta *D* erregistroak diren eta barnean dituzten zenbakiak bakoitzaren edukia[21] den:

0	E	D		
0			*2000*	*0* erregistroan 0 balioa kargatu ondorengo egoera
0	3		*2E03*	*E* erregistroan 3 balioa kargatu ondoren
0	3	1	*2D01*	*D* erregistroan 1 balioa kargatu ondoren

[20] Errepikapenak programatzen direnean amaiera dutela bermatu behar da.

[21] Edukiak egitan sistema bitarrean agertu beharko lirateke, baina irakurgarritasunagatik sistema hamartarrean jarri ditugu.

0	3	1	*BEAC*	0 eta 3 balioak desberdinak direnez sekuentzia jarraitu
0	E	D		
1	3	1	*500D*	0 eta 1 balioak batu eta emaitza *0* erregistroan gorde ondoren
0	E	D		
1	3	1	*B0A6*	1 eta 1 balioak berdinak direnez **A6**-ra jauzi egin
0	E	D		
1	3	1	*BEAC*	1 eta 3 balioak desberdinak direnez sekuentzia jarraitu
0	E	D		
2	3	1	*500D*	1 eta 1 balioak batu eta emaitza *0* erregistroan gorde ondoren
0	E	D		
2	3	1	*B0A6*	2 eta 2 balioak berdinak direnez **A6**-ra jauzi egin
0	E	D		
2	3	1	*BEAC*	2 eta 3 balioak desberdinak direnez sekuentzia jarraitu
0	E	D		
3	3	1	*500D*	2 eta 1 balioak batu eta emaitza *0* erregistroan gorde ondoren
0	E	D		
3	3	1	*B0A6*	3 eta 3 balioak berdinak direnez **A6**-ra jauzi egin
0	E	D		
3	3	1	*BEAC*	3 eta 3 balioak berdinak direnez **AC**-ra jauzi egin
0	E	D		
3	3	1	*0000*	Programa gelditu
0	E	D		

3.4.3 Bilaketa-fasea eta Exekuzio-fasea

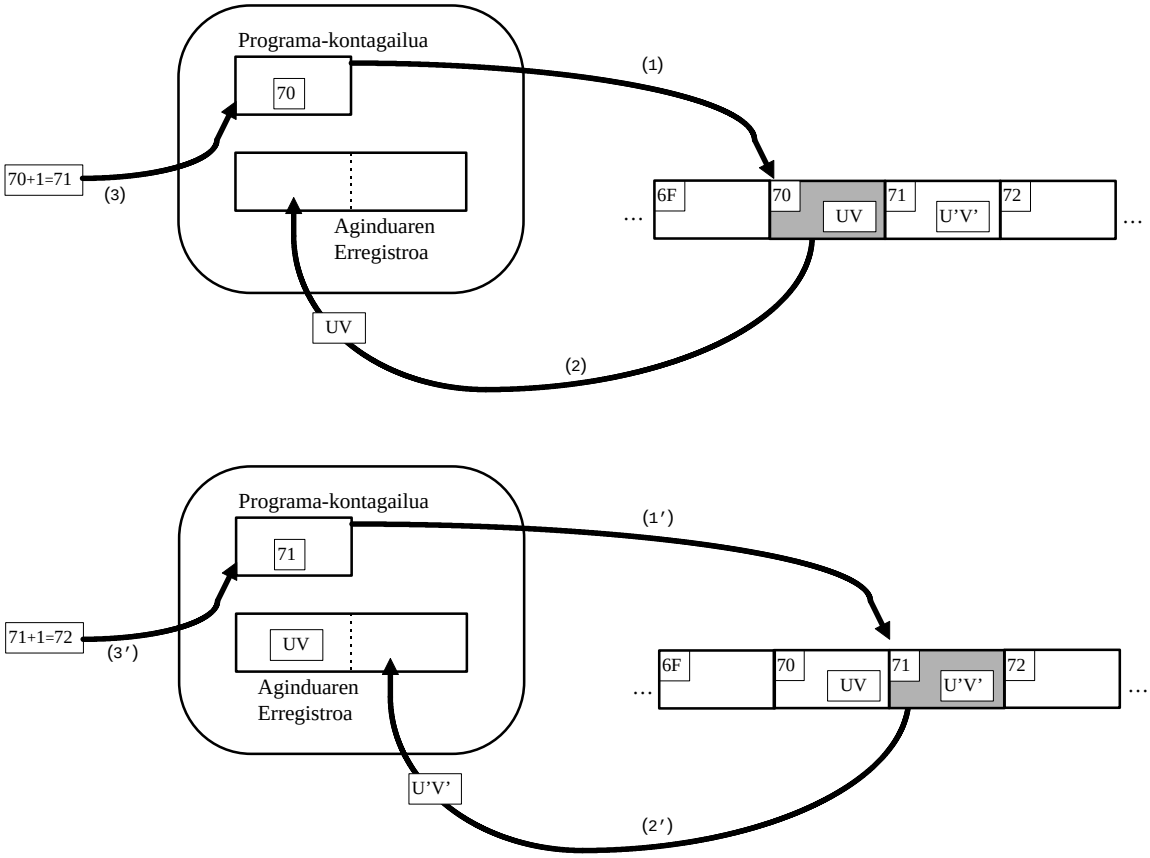
Ondoko orrialdean makina didaktikoaren irudi bat agertzen da. Erdian *Kontrol-unitatea* marraztu izan da, eta bertan osagairik garrantzitsuenak daude agerian, batetik 8 biteko *Programa-kontagailua* izeneko erregistroa, eta bestetik 16 biteko *Agindu-erregistroa*. Ezkerrean *Unitate Aritmetiko-logikoa* dago, non 8 biteko luzera daukaten 16 erregistroak dauden eta horiekin batera instrukzio aritmetiko-logikoak burutzeko zirkuituak ere (*Egoera-erregistroa* multzo honetan marraztu dugu). Irudiaren eskuinean *Memoria* aurkitzen da, eta honen atal nagusiak nabarmenki ikusten dira: 8 biteko *Helbide-erregistroa*, 8 biteko 256 gelaskak (sistema hamaseitarrean 00 eta FF bitarteko helbideak dituztenak), eta, 8 biteko *Datu-erregistroa*.

Bilaketa-fasea: Bilaketa-fasean memorian irakurketa bi egiten dira eta *Agindu-erregistro* osoa instrukzio batez betetzen da, fase hori burutu ondoren *Programa-kontagailuak* izeneko erregistroak zeukan edukia inkrementatuz hurrengo instrukzioaren helbidea gordeko du.

Bilaketa-fasean ematen diren urratsak ondokoak dira: bilaketa-fasea hasten denean (1) oraintxe[22] eskuratuko den instrukzioaren helbidea *Programa-kontagailu* erregistroan aurkitzen da. helbide-busaren bitartez *Programa-kontagailuren* edukia memoriako *Helbide-erregistrora* eramaten da eta helbide hori deskodetuz dagokion gelaska aktibatzen da, kontrol-busaren bitartez irakurketa bat burutuko dela adierazten da eta hurrengo instrukzioaren aurreko 8 bitak *Datu-erregistrora* eramaten dira. Datu-busaren zehar mugituz 8 bit

[22] Exekutatu berri den instrukzioa *Agindu-erregistroan* dago oraindik, eta *Programa-kontagailuan* hurrengo instrukzioaren helbidea dago (behar dugunaren helbidea).

horiek (UV bytea) *Agindu-erregistroaren* aurreko aldean kokatu eta gero (2), *Programa-kontagailuaren* edukia inkrementatu egiten da instrukzioaren bigarren bytearen helbidea gordez (3). Bigarren irakurketa egin ondoren - (1′) (2′) (3′) urratsak - bilaketa fasea amaitutzat eman daiteke.



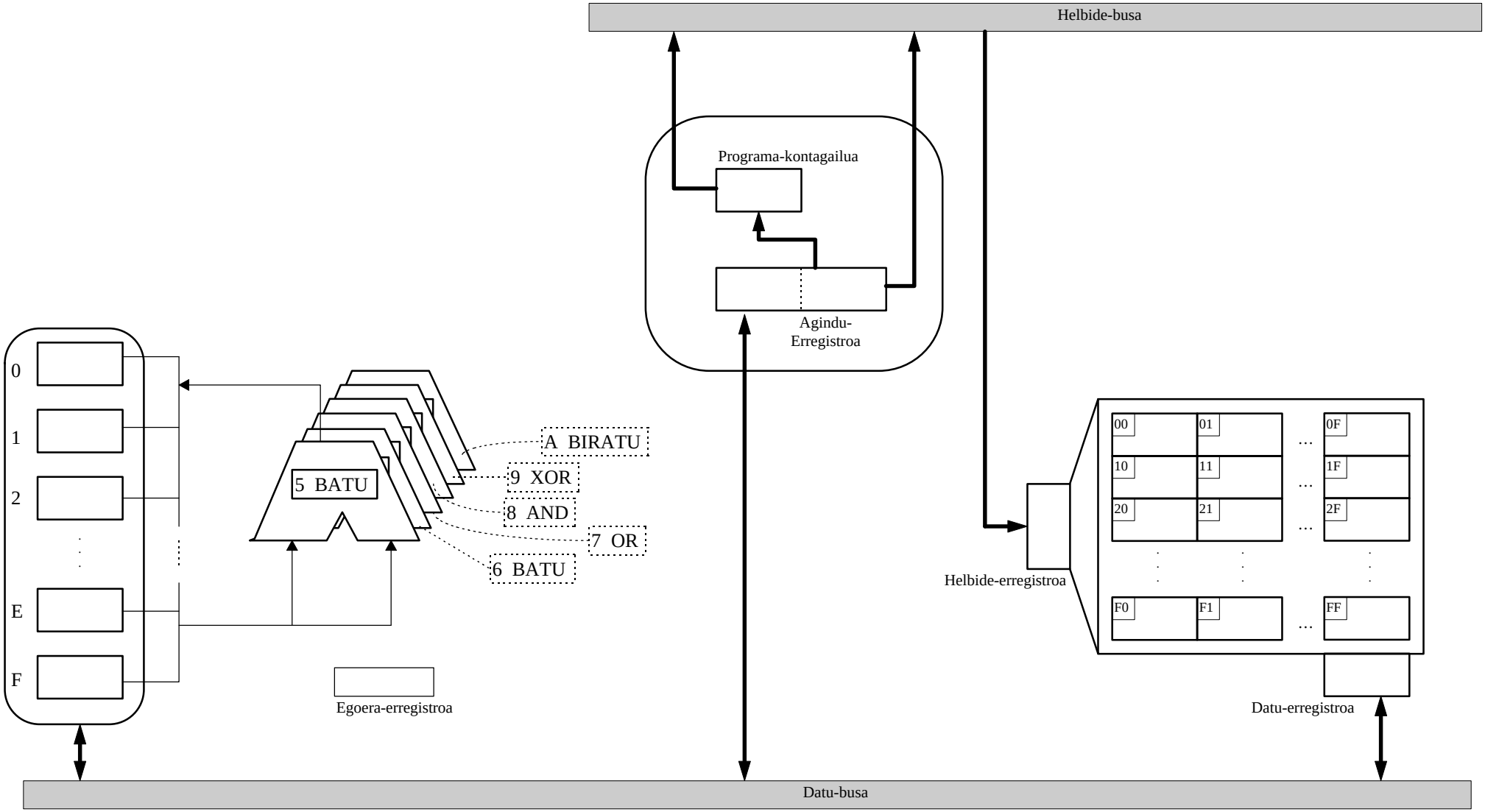
Exekuzio-fasea: Instrukzioak banan-banan memoriatik *Agindu-erregistrora* igaro behar direnez, bilaketa-fasea instrukzio guztietan ematen den zerbait da. Goiko irudian adierazi den bezala, bilaketa-fasea instrukzio guztietarako berdina da.

Baina exekuzio-faseetan betetzen diren urratsak instrukzioen araberakoak dira. Zenbaitetan *Unitate Aritmetiko-logikoaren* erregistroak edo/eta zirkuituak erabiliko dira, beste batzuetan *Programa-kontagailuaren* edukia aldatuko da instrukzioen sekuentzia hautsiz, ...

Exekuzio-fasea amaitzaean, prozesua berriro hasten da bilaketa-fasearekin, izan ere programaren hurrengo aginduaren helbidea programa-kontagailu erregistroan baitago. Bilaketa-fasean programa-kontagailuak zeukan helbideari (zenbaki bat da) inkrementu bat egiten zaio hurrengo aginduaren helbidea gorde dezan. Kasu berezia JAUZI instrukzioarena da, adibidez B4FF instrukzioa exekutatzean 4 eta 0 erregistroen edukiak konparatzen dira eta berdinak izatean hurrengo agindua FF helbidean dagoena izango denez, FF helbide hori programa-kontagailuan gordetzen da eta aginduen sekuentzia apurturik suertatzen da.

Programak bukaerarik badu dagokion sententzia bilaketa-fasean agindu-erregistrora eramango da, eta jarraian datorren exekuzio-fasean programa amaiarazi egingo da.

3.5 KONPUTAGAILU DIDAKTIKOAREN ESKEMA



3.6 ARIKETA

Jarraian azterketa batean jarritako galdera sorta bat erakusten dugu:

3.6.1 Ordenadore batek definiturik dituen oinarrizko instrukzioetatik ondokoak deskribatzen dira (hirugarren zutabeak garrantzirik ez du azterketa honetan):

Instrukzioaren Eragiketa-kodea 1 byte *(hamaseitarrean)*	***Instrukzio OSOAK duen luzera (eragiketa-kodea gehi datuak)***	Esanahaia ***(azterketarako ez du garrantzirik)***
0 0	**1** byte	NOP
0 1	**3** byte	LXI B, Helbide_16
0 2	**1** byte	STAX B
0 6	**2** byte	MVI B, Datu_8
...	**...**	**...**
1 E	**2** byte	MVI E, Datu_8
1 F	**1** byte	RAR
...	**...**	**...**
3 0	**1** byte	SIM
3 1	**3** byte	LXI SP, Helbide_16
3 2	**3** byte	STA Helbide_16
...	**...**	**...**
3 E	**2** byte	MVI A, Datu_8
...	**...**	**...**
7 5	**1** byte	MOV M, L
7 6	**1** byte	HLT
7 7	**1** byte	MOV M, A
7 8	**1** byte	MOV A, B
...	**...**	**...**
8 0	**1** byte	ADD B
...	**...**	**...**
C 6	**2** byte	ADI, Datu_8
...	**...**	**...**

Aurreko taula aintzat harturik, **gehien jota zenbat oinarrizko instrukzio** onar dezake ordenadore horrek? (arrazoindu zure erantzuna)

Instrukzioen kopuru maximoa: *Erantzunaren zergatia:*

Aurreko taula eta jarraian datorren taularen informazioa gogoan izanik (**3.6.1** eta **3.6.2** taulak), zenbat bit behar dira *Programa-kontagailu Erregistroa* egoki dimentsionatzeko?

Erantzunaren zergatia:

3.6.2 Aurreko ordenadorean exekutatzen ari den programa bat hau izanik:

Memori helbidea *(hamaseitarrean)*	***Memori posizioaren edukia 1 byte*** *(hamaseitarrean)*	***Instrukzioa?*** *(Eragiketa?*	***Operandoa?*** *Eragigaia?)*
1 0 0 0	31		
1 0 0 1	77		
1 0 0 2	1F		
1 0 0 3	3E		
1 0 0 4	32		
1 0 0 5	06		
1 0 0 6	30		
1 0 0 7	80		
1 0 0 8	32		
1 0 0 9	00		
1 0 0 A	1E		
1 0 0 B	C6		
1 0 0 C	75		
1 0 0 D	32		
1 0 0 E	02		
1 0 0 F	1E		
1 0 1 0	76		

a) Memori posizio bakoitzak zer gordetzen duen adieraziz **aurreko taularen hirrugarren zutabea bete ezazu**. Hau da, memori gelaska bakoitzak instrukzioaren Eragiketa-kodea ala instrukzioaren Datua (datua edo datuaren helbidea) biltegitzen duen asmatu behar da.

b) Programa zenbat instrukzioz osaturik dago, instrukzio guztien Eragiketa-kodeak sistema hamaseitarrean idatz itzazu.

Programaren instrukzio kopurua: *Programaren instrukzioen Eragiketa-kodeak:*

c) Programa hori exekutatzean *Programa-kontagailu Erregistroa*ren edukia aldatuz doa. Programa exekutatzean erregistro horrek, hurrenez hurren, gordetzen duena idatz ezazu.

Programa-kontagailu Erregistroaren ondoz ondoko edukia:

3.7 PROGRAMAK

Hona hemen 3. kapituluaren programak orrialdeen arabera sailkaturik:

Izena	*Programaren identifikadorea*	*ORRI.*	*Ikasgaia*
BIDERKA.PAS	ZenbakiBiBiderkatzen	3-17	Unitate Aritmetiko-logikoa

3.8 BIBLIOGRAFIA

- Alcalde, E., García, M., Peñuelas, S., *Informática Básica*, McGraw-Hill, Madrid, 1988
- Brookshear, J.G., *Introducción a las Ciencias de la Computación*, Addison-Wesley Iberoamericana, Wilmington, 1995
- Elhuyar Informatika-taldea, *Informatika Oinarrizko kontzeptuak*, Elkar-Elhuyar, Estella, 1984
- Tanenbaum, A., *Konputagailuen Antolaketa Egituratua*, EHU/UPV-ko Argitarapen Zerbitzua, 1994
- Tucker, A., *Fundamentos de Informática. Lógica, resolución de problemas, programas y computadoras*, McGraw-Hill, 1994

ERANSKINAK

E1 Transistorea

E2 Datuak lantzeko zirkuituak

E3 Datuak biltegitzeko zirkuituak

E4 Konputagailuen memoriari buruzko artikulua

E5 Mikroprozesadorei buruzko artikuluak

E6 Konputagailuen periferiko optikoei buruzko artikuluak

KONTZEPTUEN INDIZE ALFABETIKOA

—E—

—F—

—G—

—H—

—I—

—K—

—L—

—M—

—N—

—O—

—P—

—R—

—S—

—T—

—U—

—V—

—W—

—X—

—Z—