

# **Logika digitala eta mikroprogramagarria**

**Egileak: Patxi Arruti, Josu Errasti, J. Carlos Lizarbe  
Iurretako GLHB eta Errenteriako Zamalbideko irakasleak  
Argitaratzailea: ELHUYAR**

**2001eko abendua**

**Maila: Lanbide Heziketako Goi-mailako Zikloa**

**Familia: Elektrizitatea eta Elektronika**

**Zikloa: Produktu elektronikoen garapena**

**Modulua: Logika digitala eta mikroprogramagarria**

**Onespena: 2001-11-23**

## **PROGRAMAZIOA**

Goi mailako Lan-prestakuntzako Heziketa Zikloetara zuzendutako curriculum-materiala da. Hain zuzen ere, ‘Elektrizitatea eta Elektronika’ familiako ‘Produktu elektronikoen garapena’ zikloko modulua da, Lanbide Heziketako goi-mailan ematen dena.

Heziketa Ziklo bateko modulua izanik, dudarik ez dago oinarrizkoa dela ziklo hori osatu ahal izateko. Hala ere, familia bereko, eta beste zenbait familia edo ziklotako material osagarri egokia izan daiteke: informatika, automatizazio eta kontrola...

## **HELBURU ESPEZIFIKOAK**

Material hau landu eta gero, ikasleak ondoko gaitasunak bereganatuak beharko ditu:

- 1.- Elektronikako logika digitalaren oinarrizko funtzioak ezagutzea.
- 2.- Logika digitalean oinarrizkoak diren funtzioak ezagutzea.
- 3.- Funtzio boolearraren oinarrizko elementu eta eragiketak identifikatu eta erabiltzen jakitea.
- 4.- Logika digitalaren talde logikoak ezagutzea.

- 5.- Zirkuitu sekuentzialen definizioa, ezaugarriak eta osaketaz jabetzea.
- 6.- Bihurgailuen behar eta erabileraz jabetzea.
- 7.- Memorien ezaugarri nagusiak eta edukiak ezagutu eta erabiltzen jakitea.
- 8.- Gailu logiko programagarriak erabiltzeko oinarriak ezagutzea.
- 9.- Mikroprozesadoreen ezaugarriak, erabilera eta berezitasunak bereiztea.
- 10.- Mikrokontroladore desberdinen ezagutzaz jabetzea.

## MATERIALAREN EGITURA

*Logika digitala eta mikroprogramagarria* moduluak hamabi unitate didaktiko ditu. Unitate bakoitzean azalpen teorikoa da nagusi, baina azalpen horren osagarri, hainbat taula, grafiko eta eskema daude. Horiez gain, adibideak eta unitate bakoitzean ikasitakoaz ikasleak dakiena erakusteko ariketak ere badaude.

## AURKIBIDEA

Arestian aipatutako edukiak landuz, hona prestaturiko aurkibidea:

<b>1. KAPITULUA. BOOLE-REN ALJEBRA .....</b>
1.1. Sarrera .....
1.2. Aldagaien aurkezpena .....
1.3. Oinarrizko eragiketak eta propietateak .....
1.3.1. Oinarrizko eragiketak .....
1.3.2. Oinarrizko propietateak .....
1.4. Booleren aljebraren beste propietateak .....
1.5. Funtzio boolearraren forma kanonikoa .....
1.6. Egia-taula abiapuntutzat hartuta, funtzio logikoa lortzeko era .....
<b>2. KAPITULUA. OINARRIZKO FUNTZIO BOOLEARRAK .....</b>
2.1. Sarrera .....
2.2. Funtzioak edo ate logikoak .....
2.2.1. “EZ” funtzioa --- NOT atea .....
2.2.2. “EDO” funtzioa --- OR atea .....
2.2.3. “ETA” funtzioa --- AND atea .....
2.2.4. “EZ EDO” funtzioa --- NOR atea .....
2.2.5. “EZ ETA” funtzioa --- NAND atea .....
2.2.6. “ALA” funtzioa --- EX-OR atea .....
2.3. Ate logikoen ikurrak .....
2.4. Zirkuitu integratu digitalen sailkapena .....
<b>3. KAPITULUA. FUNTZIOEN LABURPENA .....</b>

3.1. Sarrera.....
3.1.1. Metodo aljebraikoa.....
3.1.2. Karnaughen metodo grafikoa.....
3.2. Funtzioen osaketa NAND ateen bidez.....
3.3. Funtzioen osaketa NOR ateen bidez.....
<b>4. KAPITULUA. TALDE LOGIKOAK.....</b>
4.1. Sarrera.....
4.2. Ate integratuen ezaugarri nagusiak .....
4.3. TTL talde logikoa.....
4.4. CMOS talde logikoa .....
4.5. TTLren eta CMOSen arteko konparazioa .....
<b>5. KAPITULUA. KONBINAZIOZKO ZIRKUITUAK.....</b>
5.1. Sarrera.....
5.2. Kodeketa eta deskodeketa.....
5.2.1. Sistema bitarra .....
5.2.2. Sistema zortzitarra.....
5.2.3. Sistema hamaseitarra .....
5.2.4. Kodeak.....
5.3. Kodetzaileak.....
5.4. Deskodetzaileak.....
5.4.1. Deskodetzaile hamaseitarra .....
5.4.2. Deskodetzaile hamartarra .....
5.4.3. Funtzio bat gauzatzea deskodetzaileak erabiliz .....
5.5. Multiplexadoreak .....
5.6. Demultiplexadorea .....
5.7. Multiplexadoreen eta demultiplexadoreen zirkuitu integratuak .....
5.8. Konparadoreak .....
5.8.1. Konparadoreen zirkuitu integratuak .....
5.9. Zirkuitu eragileak .....
5.9.1. Batuketa bitarra.....
5.9.2. Kenketa bitarra.....
5.10. Unitate aritmetiko eta logikoa (UAL).....
<b>6. KAPITULUA. ZIRKUITU SEKUENTZIALAK.....</b>
6.1. Zirkuitu sekuentzialen definizioa, ezaugarriak eta osaketa.....
6.1.1. Biegonkorrik .....
6.2. Kontagailuak .....
6.2.1. Kontagailu asinkrono hamartarra (SN7490).....
6.2.2. Kontagailu sinkronoa (SN74160).....
6.2.3. UP/DOWN kontagailua (SN74190).....
6.3. Desplazamendu-erregistroak .....
6.3.1. Serieko sarrera – serieko irteera erregistroa .....
<b>7. KAPITULUA. AD/DA BIHURGAILUAK .....</b>
7.1. Seinale analogikoak eta digitalak .....
7.2. Laginketa eta atxikipena (SAMPLE/HOLD).....
7.3. Bihurtzaile digital/analogikoa .....

7.3.1.	Erresistentzia haztatuak dituztenak.....
7.3.2.	R/2R eskailera dutenak.....
7.4.	Bihurgailu digital/analogikoen ezaugarriak.....
7.5.	Bihurgailu analogiko/digitalak.....
7.5.1.	Konparadoreduna (FLASH).....
7.5.2.	Erreferentzia-tentsioa eskaineran duena edo kontagailuduna.....
7.5.3.	Jarraipen bidezkoa.....
7.5.4.	Malda bakuneko bihurgailu analogiko/digitala.....
7.5.5.	Ondoz ondoko hurbilketa bidezkoa .....

## 8. KAPITULUA. MEMORIAK.....

8.1.	Oinarritzko jakingarriak .....
8.2.	Memorien ezaugarri nagusiak .....
8.3.	Memoria baten edukiera .....
8.4.	Memoria-motak.....
8.5.	Memoria baten kanpoko egitura .....
8.6.	Memoria baten barne-antolamendua.....
8.7.	Hiru egoeratako ateak eta konexio-busak.....
8.8.	Memoria handitzea.....
8.8.1.	Hitz-luzera handitzea .....
8.8.2.	Edukiera edo posizio-kopurua handitzea .....
8.9.	Memoria komertzialen azterketa .....
8.9.1.	SRAM memoria baten azterketa (SRAM 2112).....
8.9.2.	DRAM memoria baten azterketa (MCM514256A) .....
8.9.3.	EPROM memoria baten azterketa (27C64A) .....
8.10.	Memoria bereziak .....

## 9. KAPITULUA. PLD GAILU LOGIKO PROGRAMAGARRIAK (*PROGRAMABLE LOGIC DEVICES*).....

9.1.	Sarrera.....
9.1.1.	ASIC .....
9.1.2.	PLDak .....
9.2.	PLD zirkuituen egitura .....
9.2.1.	PLD motak .....
9.2.2.	PAL motak .....
9.3.	PLD zirkuituak programatzeko softwarea.....
9.3.1.	PLDak programatzeko urratsak .....
9.3.2.	ORCAD/PLD softwarearekin adibide bat .....
9.4.	Ekuazio boolearrak .....
9.4.1.	Kodea eta oharrak .....
9.4.2.	Seinaleen izenak .....
9.4.3.	Zenbakien indexazioa .....
9.4.4.	Eragileak .....
9.4.5.	Ekuazioak .....
9.4.6.	Pinen esleipena .....
9.5.	Egia-taulak .....
9.6.	<i>Stream</i> -ak .....
9.7.	<i>Map</i> -ak .....
9.8.	Zirkuitu sekuentzialak .....

9.8.1. Adibide bat <i>prozedurak</i> erabiliz (bi datu).....
9.9. Simulazio logikoa .....
9.9.1. Simulaziorako komandoak.....
9.10. Konpiladorearen hitz bereziak eta funtzoak.....
9.11. Eskemen bidez logika definitu.....
9.12. Prozesua .....

**10. KAPITULUA. MIKROPROZESADOREAK .....**

10.1. Zirkuitu programagarriak .....
10.1.1. Logika kableatua .....
10.1.2. Zirkuitu programagarriak .....
10.2. Mikroprozesadoreak .....
10.2.1. Funtsezko egitura .....
10.2.2. Memoria .....
10.2.3. Busak .....
10.2.4. Programak .....
10.2.5. Sistema osoa .....
10.3. Mikroprozesadore-familiak .....
10.4. Zortzi biteko mikroprozesadorearen egitura .....
10.5. 8085aren terminalen izenak .....
10.6. 8085 mikroprozesadorearen egitura .....
10.7. Helbideratzeko erak .....
10.8. Instrukzio-motak .....
10.9. Instrukzioen formatua .....
10.10. Instrukzio-zikloa eta makina-zikloa .....
10.11. Sintesi-adibidea .....
10.12. 8085aren instrukzioak .....
10.12.1. Datuen transferentzia .....
10.12.2. Eragiketa aritmetikoak .....
10.12.3. Eragiketa logikoak .....
10.12.4. Adarkatzeak .....
10.12.5. Pila, S/I eta makina-kontrola .....
10.12.6. Sinbologia eta laburdurak .....
10.13. UP 2000 plaka (sistema garatua) .....
10.13.1. Hardwarea .....
10.13.2. Softwarea .....
10.13.3. Monitorearen teklatu-komandoak (aginduak) .....
10.13.4. Asm85 mihiztatzailea .....
10.13.5. Mihiztatzailearen sasiagindu artezkariak .....
10.13.6. DDT 85 .....
10.14. Programazioa .....
10.14.1. Azpiprogramak edo azpirrutinak .....
10.15. Periferikoak: 8255 PPI ( <i>Programmable Peripheral Interface</i> ) .....

**11. KAPITULUA. 8051/52/552 MIKROKONTROLADOREAK .....**

11.1. Sistema osoa .....
11.2. Mikrokontroladorea .....
11.3. Aplikazioak .....
11.4. Aukeraketa .....

11.5.	51 familia .....
11.6.	8051/8052 mikrokontroladoreen ezaugarriak .....
11.7.	8051/52 mikrokontroladoreen <i>pin</i> -ak .....
11.8.	Memoriaren antolaketa 51/52 mikrokontroladoreetan .....
11.8.1.	Datu-memoria eta programa-memoria .....
11.8.2.	Programa-memoria .....
11.8.3.	Datu-memoria .....
11.9.	Helbideratze-motak .....
11.10.	Makina-zikloa .....
11.11.	Instrukzioak .....
11.12.	<i>Timer</i> -ak/kontagailuak .....
11.12.1.	Sarrera .....
11.12.2.	<i>Timer0</i> (T0) eta <i>Timer1</i> (T1) .....
11.12.3.	<i>Timer</i> -en lan egiteko moduak .....
11.13.	Etendurak .....
11.13.1.	Sarrera .....
11.13.2.	Etendura-motak .....
11.13.3.	Etenduren prozesua .....
11.13.4.	Etenduren lehentasun-mailak .....
11.14.	Serieko linea .....
11.14.1.	SCON serieko linearen kontrol-erregistroa .....
11.14.2.	PCON kontsumoa kontrolatzeko erregistroa .....
11.14.3.	Komunikazioaren abiadura .....
11.15.	C51 konpiladorea .....
11.15.1.	C51-k jasaten dituen datu-motak .....
11.15.2.	Memoria-motak .....
11.15.3.	Memoria-ereduak .....
11.15.4.	Erakusleak .....
11.15.5.	Etendurak .....
11.15.6.	Konpilatzeko era .....
11.15.7.	C51-ren liburutegi-funtzioak .....
<b>12.</b>	<b>KAPITULUA. PIC MIKROKONTROLADOREAK .....</b>
12.1.	Mikrokontroladoreen arkitektura orokorrak .....
12.2.	Mikrokontroladoreen barneko egitura .....
12.3.	PICetan ROM memoria-mota ezberdinak .....
12.4.	PIC mikrokontroladoreen ezaugarri nagusiak .....
12.5.	<i>Microchip</i> etxeko lau gama ezberdin .....
12.5.1.	Gama oso baxua: PIC12CXXX (8 pinekoa) .....
12.5.2.	Gama baxua: PIC16C5X (18-28 pinekoa) .....
12.5.3.	Gama ertaina: PIC16CXXX (18-40 pinekoa) .....
12.5.4.	Gama alta: PIC17CXXX (40-68 pinekoa) .....
12.6.	PICen arkitektura .....
12.6.1.	Programa-memoria .....
12.6.2.	Datu-memoria. Berariazko erregistroak .....
12.6.3.	Erlojua eta instrukzio-zikloa .....
12.7.	PIC16X84 txiparen hankatxoen deskribapena .....
12.8.	PIC16X84 mikroaren laburpen-taula .....
12.9.	Temporizadoreak .....

12.9.1. OPTION .....
12.10. Sarrera/irteera atea .....
12.11. EEPROM memoria .....
12.12. Etendurak .....
12.13. Bihurgailu analogiko/digitala .....
12.13.1. ADCON0 .....
12.13.2. ADCON1 .....
12.14. Atseden-egoera (SLEEP) .....
12.15. Instrukzioak .....
12.16. Softwarea .....
12.16.1. Programazioa .....
12.17. µPIC trainer arkitektura .....
12.17.1. Elikatze-iturria .....
12.17.2. Mikrokontroladorea .....
12.17.3. Sarrera digitalak .....
12.17.4. Sarrera analogikoak .....
12.17.5. Irteera digitalak .....
12.17.6. LCD modulua .....
12.17.7. Zirkuitu grabatzailea .....
12.17.8. Hedapen-konektorea .....
12.18. CCS etxeko PCM C konpiladorea .....
12.18.1. Aurreprozesadorearen komandoak .....
12.18.2. Datu-definizioa .....
12.18.3. Konpiladorearen funtzioak .....

# L OGIIKA DIGITALA ETA MIKROPROGRAMAGARRIA



**PATXI ARRUTI  
JOSU ERRASTI  
J. CARLOS LIZARBE**

**HEZIKETA-  
- ZIKLOAK**

IKASLEAREN LIBURUA



# LOGIKA DIGITALA ETA

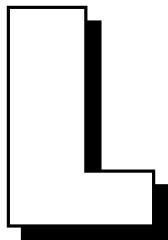
# MIKROPROGRAMAGARRIA



HEZIKETA-  
-ZIKLOAK

Eduardo

Elhuyar



LOGIKA DIGITALA  
ETA

MIKROPROGRAMAGARRIA

**Egileak:** Patxi Arruti  
**Josu Errasti**  
**J. Carlos Lizarbe**

Hezkuntza, Unibertsitate eta Ikerketa Sailak onetsia: 2001-11-23

Azalaren diseinua: M. Karmen Urdangarin

Maketa: Ainara Sarasketa, M. Karmen Urdangarin

Hizkuntz zuzenketak: Elhuyar

© Edizio honena: ELHUYAR, K.E. Asteasuain 14. Txikierdi. 20170 USURBIL (Gip.) (2001)  
E-mail: elhuyar@elhuyar.com - <http://www.elhuyar.com>

© Patxi Arruti, Josu Errasti, J. Carlos Lizarbe

# AURKIBIDEA

---

<b>1. KAPITULUA. BOOLE-REN ALJEBRA</b>	<b>11</b>
1.1. Sarrera	11
1.2. Aldagaien aurkezpena	12
1.3. Oinarrizko eragiketak eta propietateak	12
1.3.1. Oinarrizko eragiketak	12
1.3.2. Oinarrizko propietateak	12
1.4. Booleren aljebraren beste propietateak	13
1.5. Funtzio boolearraren forma kanonikoa	14
1.6. Egia-taula abiapuntutzat hartuta, funtzio logikoa lortzeko era	15
<b>2. KAPITULUA. OINARRIZKO FUNTZIO BOOLEARRAK</b>	<b>17</b>
2.1. Sarrera	17
2.2. Funtzioak edo ate logikoak	17
2.2.1. “EZ” funtzioa --- NOT atea	17
2.2.2. “EDO” funtzioa --- OR atea	18
2.2.3. “ETA” funtzioa --- AND atea	19
2.2.4. “EZ EDO” funtzioa --- NOR atea	19
2.2.5. “EZ ETA” funtzioa --- NAND atea	20
2.2.6. “ALA” funtzioa --- EX-OR atea	21
2.3. Ate logikoen ikurrik	22
2.4. Zirkuitu integratu digitalen sailkapena	23
<b>3. KAPITULUA. FUNTZIOEN LABURPENA</b>	<b>26</b>
3.1. Sarrera	26
3.1.1. Metodo algebraikoa	26
3.1.2. Karnaughen metodo grafikoa	27
3.2. Funtzioen osaketa NAND ateen bidez	31
3.3. Funtzioen osaketa NOR ateen bidez	32
<b>4. KAPITULUA. TALDE LOGIKOAK</b>	<b>37</b>
4.1. Sarrera	37
4.2. Ate integratuen ezaugarri nagusiak	37
4.3. TTL talde logikoa	38
4.4. CMOS talde logikoa	39
4.5. TTLren eta CMOSen arteko konparazioa	40
<b>5. KAPITULUA. KONBINAZIOZKO ZIRKUITUAK</b>	<b>45</b>
5.1. Sarrera	45
5.2. Kodeketa eta deskodeketa	46
5.2.1. Sistema bitarra	46
5.2.2. Sistema zortzitarra	47

5.2.3. Sistema hamaseitarra .....	48
5.2.4. Kodeak .....	50
<b>5.3. Kodetzaileak .....</b>	<b>54</b>
<b>5.4. Deskodetzaileak .....</b>	<b>57</b>
5.4.1. Deskodetzaile hamaseitarra .....	59
5.4.2. Deskodetzaile hamartarra .....	59
5.4.3. Funtzio bat gauzatzea deskodetzaileak erabiliz .....	60
<b>5.5. Multiplexadoreak .....</b>	<b>60</b>
<b>5.6. Demultiplexadorea .....</b>	<b>61</b>
<b>5.7. Multiplexadoreen eta demultiplexadoreen zirkuitu integratuak .....</b>	<b>63</b>
<b>5.8. Konparadoreak .....</b>	<b>64</b>
5.8.1. Konparadoreen zirkuitu integratuak .....	65
<b>5.9. Zirkuitu eragileak .....</b>	<b>66</b>
5.9.1. Batuketa bitarra .....	66
5.9.2. Kenketa bitarra .....	70
<b>5.10. Unitate aritmetiko eta logikoa (UAL) .....</b>	<b>72</b>
 <b>6. KAPITULUA. ZIRKUITU SEKUENTZIALAK .....</b>	<b>79</b>
<b>6.1. Zirkuitu sekuentzialen definizioa, ezaugarriak eta osaketa .....</b>	<b>79</b>
6.1.1. Biegonkorrak .....	81
<b>6.2. Kontagailuak .....</b>	<b>95</b>
6.2.1. Kontagailu asinkrono hamartarra (SN7490) .....	96
6.2.2. Kontagailu sinkronoa (SN74160) .....	96
6.2.3. UP/DOWN kontagailua (SN74190) .....	97
<b>6.3. Desplazamendu-erregistroak .....</b>	<b>99</b>
6.3.1. Serieko sarrera – serieko irteera erregistroa .....	99
 <b>7. KAPITULUA. AD/DA BIHURGAILUAK .....</b>	<b>105</b>
<b>7.1. Seinale analogikoak eta digitalak .....</b>	<b>105</b>
<b>7.2. Laginketa eta atxikipena (SAMPLE/HOLD) .....</b>	<b>108</b>
<b>7.3. Bihurtzaile digital/analogikoak .....</b>	<b>109</b>
7.3.1. Erresistentzia haztatuak dituztenak .....	109
7.3.2. R/2R eskailera dutenak .....	110
<b>7.4. Bihurgailu digital/analogikoen ezaugarriak .....</b>	<b>112</b>
<b>7.5. Bihurgailu analogiko/digitalak .....</b>	<b>113</b>
7.5.1. Konparadoreduna (FLASH) .....	113
7.5.2. Erreferentzia-tentsioa eskaileran duena edo kontagailuduna .....	115
7.5.3. Jarraipen bidezkoa .....	116
7.5.4. Malda bakuneko bihurgailu analogiko/digitala .....	116
7.5.5. Ondoz ondoko hurbilketa bidezkoa .....	125
 <b>8. KAPITULUA. MEMORIAK .....</b>	<b>127</b>
<b>8.1. Oinarrizko jakingarriak .....</b>	<b>127</b>
<b>8.2. Memoriien ezaugarri nagusiak .....</b>	<b>128</b>
<b>8.3. Memoria baten edukiera .....</b>	<b>129</b>
<b>8.4. Memoria-motak .....</b>	<b>130</b>
<b>8.5. Memoria baten kanpoko egitura .....</b>	<b>132</b>

<b>8.6. Memoria baten barne -antolamendua .....</b>	<b>133</b>
<b>8.7. Hiru egoeratako ateak eta konexio-busak .....</b>	<b>134</b>
<b>8.8. Memoria handitzea.....</b>	<b>136</b>
8.8.1. Hitz-luzera handitzea.....	136
8.8.2. Edukiera edo posizio-kopurua handitzea.....	138
<b>8.9. Memoria komertzialen azterketa .....</b>	<b>140</b>
8.9.1. SRAM memoria baten azterketa (SRAM 2112).....	140
8.9.2. DRAM memoria baten azterketa (MCM514256A).....	141
8.9.3. EPROM memoria baten azterketa (27C64A) .....	142
<b>8.10. Memoria bereziak.....</b>	<b>143</b>
 <b>9. KAPITULUA. PLD GAILU LOGIKO PROGRAMAGARRIAK</b> <i>(PROGRAMABLE LOGIC DEVICES)</i> .....	<b>149</b>
<b>9.1. Sarrera .....</b>	<b>149</b>
9.1.1. ASIC.....	149
9.1.2. PLDak.....	151
<b>9.2. PLD zirkuituen egitura.....</b>	<b>152</b>
9.2.1. PLD motak.....	152
9.2.2. PAL motak.....	157
<b>9.3. PLD zirkuituak programatzeko softwarea.....</b>	<b>171</b>
9.3.1. PLDak programatzeko urratsak.....	171
9.3.2. ORCAD/PLD softwarearekin adibide bat.....	172
<b>9.4. Ekuazio boolearrak.....</b>	<b>177</b>
9.4.1. Kodea eta oharrak.....	177
9.4.2. Seinaleen izenak.....	178
9.4.3. Zenbakien indexaziao.....	178
9.4.4. Eragileak.....	178
9.4.5. Ekuazioak.....	179
9.4.6. Pinen esleipena.....	180
<b>9.5. Egia-taulak.....</b>	<b>183</b>
<b>9.6. Stream-ak .....</b>	<b>184</b>
<b>9.7. Map-ak .....</b>	<b>184</b>
<b>9.8. Zirkuitu sekuentzialak.....</b>	<b>187</b>
9.8.1. Adibide bat <i>prozedurak</i> erabiliz (bi datu).....	190
<b>9.9. Simulazio logikoa.....</b>	<b>192</b>
9.9.1. Simulaziorako komandoak.....	192
<b>9.10. Konpiladorearen hitz bereziak eta funtzioak .....</b>	<b>194</b>
<b>9.11. Eskemen bidez logika definitu .....</b>	<b>195</b>
<b>9.12. Prozesua .....</b>	<b>195</b>
 <b>10. KAPITULUA. MIKROPROZESADOREAK .....</b>	<b>209</b>
<b>10.1. Zirkuitu programagarriak .....</b>	<b>209</b>
10.1.1. Logika kableatua .....	209
10.1.2. Zirkuitu programagarriak .....	210
<b>10.2. Mikroprozesadoreak .....</b>	<b>211</b>
10.2.1. Funtsezko egitura .....	211
10.2.2. Memoria .....	211

10.2.3. Busak.....	212
10.2.4. Programak.....	213
10.2.5. Sistema osoa.....	214
<b>10.3. Mikroprozesadore -familiak.....</b>	<b>215</b>
<b>10.4. Zortzi biteko mikroprozesadorearen egitura .....</b>	<b>215</b>
<b>10.5. 8085aren terminalen izenak.....</b>	<b>216</b>
<b>10.6. 8085 mikroprozesadorearen egitura .....</b>	<b>218</b>
<b>10.7. Helbideratzeko erak.....</b>	<b>221</b>
<b>10.8. Instrukzio-motak.....</b>	<b>222</b>
<b>10.9. Instrukzioen formatua .....</b>	<b>223</b>
<b>10.10. Instrukzio-zikloa eta makina-zikloa.....</b>	<b>223</b>
<b>10.11. Sintesi-adibidea.....</b>	<b>224</b>
<b>10.12. 8085aren instrukzioak.....</b>	<b>226</b>
10.12.1. Datuen transferentzia.....	226
10.12.2. Eragiketa aritmetikoak.....	227
10.12.3. Eragiketa logikoak.....	228
10.12.4. Adarkatzeak.....	229
10.12.5. Pila, S/I eta makina-kontrola.....	230
10.12.6. Sinbologia eta laburdurak.....	231
<b>10.13. UP 2000 plaka (sistema garatua).....</b>	<b>231</b>
10.13.1. Hardwarea.....	231
10.13.2. Softwarea.....	232
10.13.3. Monitorearen teklatu-komandoak (aginduak).....	232
10.13.4. Asm85 mihiztatzalea.....	233
10.13.5. Mihiztatzalearen sasiagindu artezkariak.....	233
10.13.6. DDT 85.....	234
<b>10.14. Programazioa.....</b>	<b>234</b>
10.14.1. Azpiprogramak edo azpirrutinak.....	235
<b>10.15. Periferikoak: 8255 PPI (<i>Programmable Peripheral Interface</i>).....</b>	<b>236</b>
 <b>11. KAPITULUA. 8051/52/552 MIKROKONTROLADOREAK .....</b>	<b>241</b>
<b>11.1. Sistema osoa.....</b>	<b>241</b>
<b>11.2. Mikrokontroladorea.....</b>	<b>241</b>
<b>11.3. Aplikazioak.....</b>	<b>242</b>
<b>11.4. Aukeraketa.....</b>	<b>242</b>
<b>11.5. 51 familia.....</b>	<b>243</b>
<b>11.6. 8051/8052 mikrokontroladoreen ezaugarriak.....</b>	<b>244</b>
<b>11.7. 8051/52 mikrokontroladoreen pin-ak.....</b>	<b>245</b>
<b>11.8. Memoriaren antolaketa 51/52 mikrokontroladoreetan.....</b>	<b>247</b>
11.8.1. Datu-memoria eta programa-memoria.....	247
11.8.2. Programa-memoria .....	248
11.8.3. Datu-memoria .....	249
<b>11.9. Helbideratze-motak.....</b>	<b>253</b>
<b>11.10. Makina-zikloa .....</b>	<b>254</b>
<b>11.11. Instrukzioak.....</b>	<b>255</b>
<b>11.12. Timer-ak/kontagailuak.....</b>	<b>260</b>
11.12.1. Sarrera.....	260

11.12.2. <i>Timer0</i> (T0) eta <i>Timer1</i> (T1).....	260
11.12.3. <i>Timer</i> -en lan egiteko moduak.....	263
<b>11.13. Etendurak.....</b>	<b>265</b>
11.13.1. Sarrera.....	265
11.13.2. Etendura-motak.....	266
11.13.3. Etenduren prozesua.....	268
11.13.4. Etenduren lehentasun-mailak.....	269
<b>11.14. Serieko linea.....</b>	<b>270</b>
11.14.1. SCON serieko linearen kontrol-erregistroa.....	272
11.14.2. PCON kontsumoa kontrolatzeko erregistroa.....	273
11.14.3. Komunikazioaren abiadura.....	273
<b>11.15. C51 konpiladorea.....</b>	<b>279</b>
11.15.1. C51-k jasaten dituen datu-motak.....	279
11.15.2. Memoria-motak.....	280
11.15.3. Memoria-ereduak.....	281
11.15.4. Erakusleak.....	281
11.15.5. Etendurak.....	282
11.15.6. Konpilatzeko era.....	283
11.15.7. C51-ren liburutegi-funtzioak.....	283
<b>12. KAPITULUA. PIC MIKROKONTROLADOREAK.....</b>	<b>297</b>
<b>12.1. Mikrokontroladoreen arkitektura orokorrak.....</b>	<b>297</b>
<b>12.2. Mikrokontroladoreen barneko egitura.....</b>	<b>297</b>
<b>12.3. PICetan ROM memoria-mota ezberdinak.....</b>	<b>298</b>
<b>12.4. PIC mikrokontroladoreen ezaugarri nagusiak.....</b>	<b>299</b>
<b>12.5. Microchip etxeko lau gama ezberdin.....</b>	<b>300</b>
12.5.1. Gama oso baxua: PIC12CXXX (8 pinekoa).....	301
12.5.2. Gama baxua: PIC16C5X (18-28 pinekoa).....	301
12.5.3. Gama ertaina: PIC16CXXX (18-40 pinekoa).....	302
12.5.4. Gama altua: PIC17CXXX (40-68 pinekoa).....	304
<b>12.6. PICen arkitektura.....</b>	<b>304</b>
12.6.1. Programa-memoria.....	305
12.6.2. Datu-memoria. Berariazko erregistroak.....	305
12.6.3. Erlojua eta instrukzio-zikloa.....	309
<b>12.7. PIC16X84 txiparen hankatxoen deskribapena.....</b>	<b>310</b>
<b>12.8. PIC16X84 mikroaren laburpen-taula.....</b>	<b>311</b>
<b>12.9. Temporizadoreak.....</b>	<b>312</b>
12.9.1. OPTION.....	312
<b>12.10. Sarrera/irteera atea.....</b>	<b>313</b>
<b>12.11. EEPROM memoria.....</b>	<b>314</b>
<b>12.12. Etendurak.....</b>	<b>315</b>
<b>12.13. Bihurgailu analogiko/digitala.....</b>	<b>315</b>
12.13.1. ADCON0.....	315
12.13.2. ADCON1.....	316
<b>12.14. Atseden-egoera (SLEEP).....</b>	<b>317</b>
<b>12.15. Instrukzioak.....</b>	<b>318</b>
<b>12.16. Softwarea.....</b>	<b>319</b>

12.16.1. Programazioa.....	319
<b>12.17. mPIC trainer arkitektura .....</b>	<b>320</b>
12.17.1. Elikatze-iturria.....	321
12.17.2. Mikrokontroladorea.....	321
12.17.3. Sarrera digitalak.....	321
12.17.4. Sarrera analogikoak.....	321
12.17.5. Irteera digitalak.....	322
12.17.6. LCD modulua.....	322
12.17.7. Zirkuitu grabatzalea.....	322
12.17.8. Hedapen-konektorea.....	323
<b>12.18. CCS etxeko PCM C konpiladorea.....</b>	<b>333</b>
12.18.1. Aurreprozesadorearen komandoak.....	335
12.18.2. Datu-definizioa.....	337
12.18.3. Konpiladorearen funtzioak .....	338

# 1. KAPITULUA

## Boole-ren aljebra

---

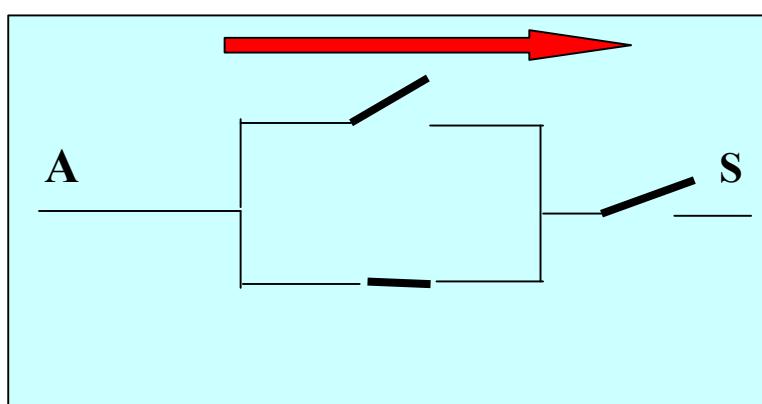
### 1.1. SARRERA

---

Booleren aljebra edo aljebra logikoaren hastapenak 1847. urtekoak dira eta George Boole matematikari britainiarren lanetan oinarritu ziren.

Gure munduan, zenbait problematan erabiltzen diren datu guztiak bi balioatik bat bakarrik har dezakete; eta bat hori hartzean bestea baztertu egiten da, eta alderantziz. Esate baterako, proposizio batzuk ondo eginak badaude, gezurra edo egia izango dira; beraz, proposizio horietako batzuk erlazionatuz lortzen den ondorioa gezurra edo egia izango da. Hona hemen beste adibide bat. Etengailu elektriko batek bi posizio dauzka: itxita, eta korronte elektrikoa iragan egingo da; edo irekita, korronte elektrikoa ez da iragango. Etengailuak elkarri nola lotzen diren, korronteak bidea izango du edo ez du izango.

Adibidea:



Irudian agertzen den zirkuituan, Atik Sra joateko bidea etenda dago, eta korrontea ez da igaroko.

## 1.2. ALDAGAIEN AURKEZPENA

---

Booleren aljebran, ekuazioetan agertzen diren aldagaiaiak letra larriz edo xehez idatzi ahal izaten dira.

**A, B, D, E, MORGAN..... a, b, d, e, f,.....**

## 1.3. OINARRIZKO ERAGIKETAK ETA PROPIETATEAK

---

### 1.3.1. OINARRIZKO ERAGIKETAK

Booleren aljebran bi eragiketa besterik ez dago:

ERAGIKETA	AURKEZTEKO ERA
BATUKETA	$F = A + B$
BIDERKETA	$F = A \cdot B$

Bi eragiketa logiko horietaz gain, beste funtzio hau ere aipatu behar da:

$$\text{EZEZTAPENA} \quad F = \overline{A}, \quad F = \overline{A \cdot B}$$

### 1.3.2. OINARRIZKO PROPIETATEAK

Booleren aljebrako eragiketek betetzen dituzten oinarrizko propietateak hauek dira:

- a) Eragiketak **trukakorrak** dira. Hau da, **a** eta **b** aljebrako elementuak izanik.

$$a + b = b + a \quad \text{eta} \quad a \cdot b = b \cdot a$$

b) Eragiketa bakoitza **banakorra** da bestearekiko.

$$a \cdot (b + c) = a \cdot b + a \cdot c \quad \text{eta} \quad a + b \cdot c = (a + b) \cdot (a + c)$$

c) Eragiketak **elkarkorrak** dira.

$$(a + b) + c = a + (b + c) = a + b + c \quad \text{eta} \quad (a \cdot b) \cdot c = a \cdot (b \cdot c) = a \cdot b \cdot c$$

d) Batuketaren eta biderketaren elementu neutroak 0 eta 1 dira, hurrenez hurren.

$$a + 0 = a \quad \text{eta} \quad a \cdot 1 = a$$

e) Edozein **a**-rentzat existitzen da  $\bar{a}$

$$a + \bar{a} = 1 \quad \text{eta} \quad a \cdot \bar{a} = 0$$

$\bar{a}$  elementuari a-ren **osagarria** deritzo.

f) Aljibrako **a** elementu bakoitzarentzat honakoa egiaztatzen da:

$$a + 1 = 1 \quad \text{eta} \quad a \cdot 0 = 0$$

## 1.4. BOOLEREN ALJEBRAKO TEOREMAK

---

1. **Idenpotentzia-proprietatea.** Aljibrako **a** elementu bakoitzarentzat honakoa betetzen da:

$$a + a = a \quad \text{eta} \quad a \cdot a = a$$

2. **Xurgatze**-proprietatea. Aljibrako **a** eta **b** elementuz osatutako bikote bakoitzarentzat:

$$a + (a \cdot b) = a \quad \text{eta} \quad a \cdot (a + b) = a$$

3. **Inboluzio-proprietatea.** Booleren aljebran **a** elementu guztientzat:

$$\overline{\overline{a}} = a$$

Ondorengo zenbait orrialdetan propietate hau erabiltzean, alderanzketa edo ezeptapen bikoitza egin dugula esango dugu.

4. Booleren aljebran, **Morganen legeak** hauek dira:

$$\overline{a + b} = \overline{a} \cdot \overline{b}$$

$$\overline{a \cdot b} = \overline{a} + \overline{b}$$

## 1.5. FUNTZIO BOOLEARRAREN FORMA KANONIKOA

---

Funtzio boolearraren era kanonikoa edozein biderkaduraren batuketa edo baturaren biderketa gisa adieraz daiteke, baldin eta adierazpidea eratzen duten gai bakoitzean, zuzenean zein era osagarrian, aldagai guztiak agertzen badira. Forma kanonikoen adibideak dira honako funtzio hauek (kontu izan bi forma kanoniko horiek ez dutela zerikusirik batak bestearekin):

$$S_1 = a \cdot b \cdot \overline{c} + a \cdot \overline{b} \cdot \overline{c} + \overline{a} \cdot \overline{b} \cdot \overline{c}$$

$$S_0 = (a + b + c) \cdot (a + \overline{b} + \overline{c}) \cdot (\overline{a} + b + c)$$

**S<sub>1</sub>** bezalako funtzioei lehen forma kanonikoko funtzioko edo **MINTERM** deritzegu.

**S<sub>0</sub>** bezalakoei, ostera, bigarren forma kanonikoko funtzioko edo **MAXTERM**

Funtzio logiko oro, hartarako behar diren aldaketak eginda, forma kanonikoan ere adieraz daiteke.

Hemendik aurrera, funtziobatean **+** eta **.** eragiketak parentesirik gabe nahasirik agertzen badira, lehendabizi **.** eragiketak eta gero **+** eragiketak egin behar direla ulertuko da.

## 1.6. EGIA-TAULA ABIAPUNTUTZAT HARTUTA, FUNTZIO LOGIKOA LORTZEKO ERA

Diseinuaren prozesu logikoak, gehienetan, egia-taula irudikatzea hartzen du abiapuntutzat. Funtzioaren arabera, sarrerako aldagaien konbinazio bitar bakoitzarentzat gertatzen den irteera-balioa jartzen da egia-taulan. Egia-taula oinarrizzat hartuta, forma kanonikoaren bi eratako edozeinetan lor daiteke funtzioa:

- **Lehen forma kanonikoa** funtzioari “1” balioa ematen dioten aldagaien biderkadura kanoniko guztiak hartuta lortzen da.
- **Bigarren forma kanonikoa** funtzioari “0” balioa ematen dioten aldagaien batura kanoniko guztiak biderkatuta lortzen da.

Bi esaldi horiek argiago ikusteko, 1.6. irudian dagoen egia-taulatik lehen eta bigarren forma kanonikoak aterako ditugu.

A	B	C	S
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

1.1. irudia. Funtzio logiko edo boolean baten egia-taula.

$$S_1 = \bar{a} \cdot \bar{b} \cdot \bar{c} + a \cdot \bar{b} \cdot c + a \cdot b \cdot \bar{c}$$

$$S_0 = (a + b + \bar{c}) \cdot (a + \bar{b} + c) \cdot (a + \bar{b} + \bar{c}) \cdot (\bar{a} + b + c) \cdot (\bar{a} + \bar{b} + \bar{c})$$

## Ariketak

- Ondoan agertzen diren ekuazioak laburtu behar dituzu, Booleren aljebraaren propietateak erabiliz.

$$F = (a \cdot 0) \cdot (b + b) + (b + \bar{b}) \cdot (a \cdot a) + (b + 1) \cdot (c \cdot \bar{c})$$

$$T = (b + 1) \cdot a \cdot \bar{a} + a + c \cdot \bar{c} + b \cdot 0 + c$$

- Ekuazio hauei Morganen legeak aplikatu behar dizkiezu<sup>1</sup>.

$$S = \bar{a} \cdot \bar{b} \cdot c + a \cdot d + \bar{b} \cdot c \cdot \bar{d}$$

$$R = (a + b + c) \cdot (\bar{a} + b + \bar{c}) \cdot (a + \bar{b} + c)$$

- Egia-taula hauetatik funtzio kanonikoak lortu behar dituzue, minterm eta maxterm deritzegun gaiak adieraziz.

A	B	C	S
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

X	Y	Z	F
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

---

<sup>1</sup> Morganen legeak betetzen dira bi aldagai baino gehiagorekin ere

$$\begin{aligned} a + b + c + d &= \bar{a} \cdot \bar{b} \cdot \bar{c} \cdot \bar{d} \\ a \cdot b \cdot c \cdot d &= \bar{a} + \bar{b} + \bar{c} + \bar{d} \end{aligned}$$

## 2. KAPITULUA

# Oinarrizko funtzio boolearrak

### 2.1. SARRERA

Elektronikak azken urteotan izan duen aurrerapenari esker, tresna batzuk agertu dira logika aplikatzeko. Tresna horiei zirkuitu integratu digital (digital IC) deritze. Normalki, txip-eran eraikitzen dira eta funtsean funtzio logiko batzuk dituzte. Funtzio logiko horiek elkartuz, edozein sistema digital eraikitzeko ahalmena izango dugu.

Funtzio logiko horiek ate deituriko osagai elektronikoen bidez lortzen dira; beraz, gai honetan ate horiek zein diren, horien ezaugarri logikoak eta nola eraikitzen diren ikasiko dugu.

### 2.2. FUNTZIOAK EDO ATE LOGIKOAK

Jarraian aipatzen direnak dira oinarrizkoenak:

#### 2.2.1. “EZ” FUNTZIOA --- NOT ATEA

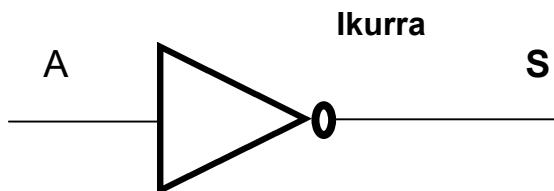
“EZ” funtzioa edo alderanzketa-funtzioa asko erabiltzen den funtzio logikoa dugu, eta honela definitzen da:

$$S = \overline{A}$$

(aldagaiaaren gaineko zeinuak ezeztapena adierazten du)

Horrek esan nahi du, funtzio horren bidez, aldagaia ukatu egiten dela.

Funtzio hori **NOT** ateen bidez lortzen da.



A	S
0	1
1	0

### **2.2.2. “EDO” FUNTZIOA --- OR ATEA**

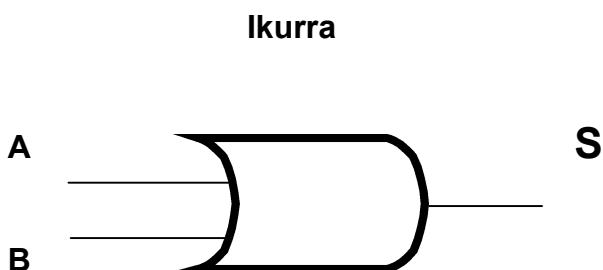
Funtzio hau honela definitzen da:

$$\mathbf{S} = \mathbf{A} + \mathbf{B} + \mathbf{C} + \dots$$

**“EDO” funtzioak honako ezaugarriak izango ditu:**

- Irteera “1” mailan egon dadin (**S = 1**), beharrezko da aldagaietako bat “1” mailan egotea.
  - Aldagai bat baino gehiago badira “1” mailan daudenak, irteera “1” mailan egongo da.
  - **S = 0** izan dadin, aldagai guztiekin “0” mailakoak izan beharko dute.

Normalean, fabrikatzaileek bi aldagaiko “**EDO**” funtzioa egiten dute. Funtzio hori **OR** ateen bidez lortzen da.



Egia-taula		
A	B	S
0	0	0
0	1	1
1	0	1
1	1	1

$$\mathbf{S} = \mathbf{A} + \mathbf{B}$$

--- AND ATEA

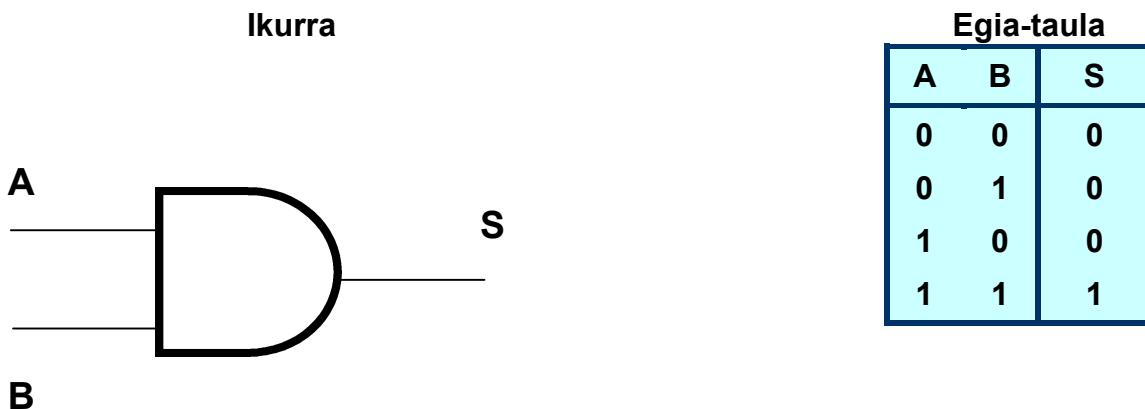
Funtzio hau honela definitzen da:

$$S = A \bullet B \bullet C \bullet \dots$$

**“ETA” funtziok honako ezaugarriak izango ditu:**

- S irteerak “1” maila ukantzen dezakoa da aldagai guztiak “1” mailan egotea.
  - Aldagai batek edo gehiagok “0” balioa hartuko balu, irteera “0” mailan egongo litzateke.

Normalean, fabrikatzaileek 2, 3 eta 4 aldagaiko “ETA” funtziokoak egiten dituzte. Funtzio horiek AND ateeng bidez lortzen dira.



$$\mathbf{S} = \mathbf{A} \bullet \mathbf{B}$$

#### 2.2.4. "EZ EDO" FUNTZIOA --- NOR ATEA

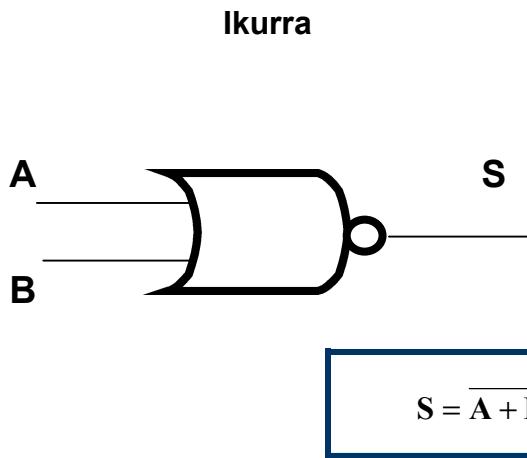
Funtzio hau **EDO** funtziaren alderantzizkoa da, eta honako ekuazio hau du:

$$\mathbf{S} = \overline{\mathbf{A} + \mathbf{B} + \mathbf{C} + \dots}$$

Aldagaiaren balioaren arabera:

- Aldagai guztiak “0” balioa dutenean, “1” maila izango du funtzioak.
- Aldagai batek edo gehiagok “1” balioa dutenean, irteera “0” izango da.

Fabrikatzaileek hainbat sarrera-kopurutakoak egiten dituzte. Funtzio hori **NOR** ateen bidez lortzen da.



**Egia-taula**

A	B	S
0	0	1
0	1	0
1	0	0
1	1	0

## 2.2.5. “EZ ETA” FUNTZIOA --- NAND ATEA

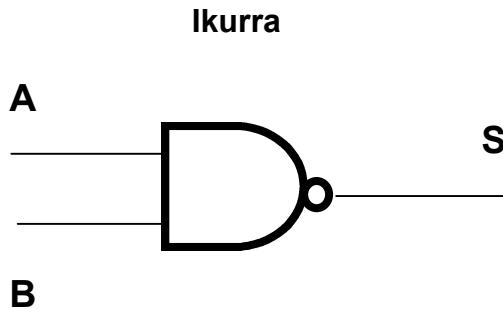
Funtzio hau **ETA** funtzioaren alderantzizkoa da, eta honelako ekuazioa betetzen du:

$$S = \overline{A \cdot B \cdot C \cdot \dots}$$

Aldagaiaren balioaren arabera:

- Funtzioak “1” maila izateko, aldagai batek edo gehiagok “0” mailan izan beharko dute.
- Aldagai guztiak “1” maila izango balute, funtzioak “0” maila hartuko luke.

Fabrikatzaileek hainbat sarrera-kopurutakoak egiten dituzte. Funtzio hori **NAND** ateen bidez lortzen da.



**Egia-taula**

A	B	S
0	0	1
0	1	1
1	0	1
1	1	0

$$S = \overline{A} \cdot B + A \cdot \overline{B}$$

## 2.2.6. “ALA” FUNTZIOA --- EX-OR ATEA

Erabiltzen den beste funtzio bat “ALA” funtzioa da. EXCLUSIVOR ere deitzen zaio, eta honela definitzen da:

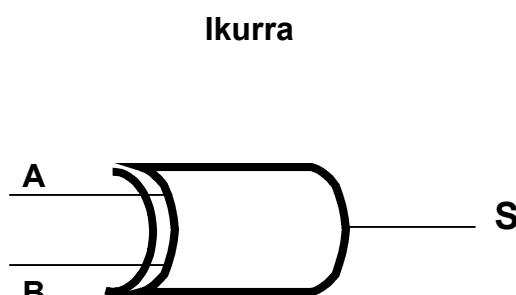
$$S = \overline{A} \cdot B + A \cdot \overline{B}$$

$$S = A \Delta B$$

Aldagaiaren balioaren arabera:

- Funtzioak “1” maila izateko, **A** eta **B** aldagaiiek desberdinak izan behar dute.
- Aldagaiak berdinak badira, irteera “0” izango da.

Fabrikatzaileek bi sarreratakoak egiten dituzte. Funtzio hori **EX-OR** ateen bidez lortzen da.

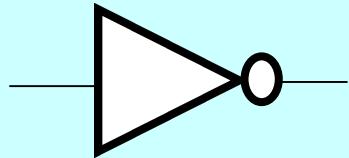
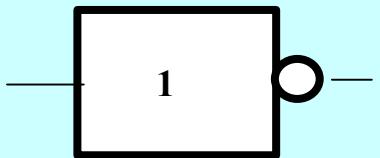
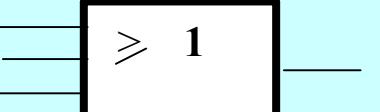
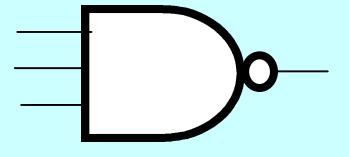
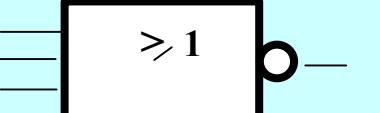
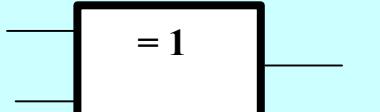


**Egia-taula**

A	B	S
0	0	0
0	1	1
1	0	1
1	1	0

$$S = \overline{A} \cdot B + A \cdot \overline{B}$$

## 2.3. ATE LOGIKOEN IKURRAK

Funtzioa	Ikurrik hedatuena	Nazioarteko Elektroteknika Batzordearen ikurra
EZ (NOT)		
ETA (AND)		
EDO (OR)		
EZ ETA (NAND)		
EZ EDO (NOR)		
ALA (EX - OR)		

## 2.4. ZIRKUITU INTEGRATU DIGITALEN SAILKAPENA

---

Zirkuituen handitasunaren arabera (ez handitasun fisikoa, konplexutasuna baizik) honela sailkatzen dira:

### S.S.I. (Small Scale Integration)

Zirkuitu integratu hauek 10 ate baino gutxiago dituzte.

### M.S.I. (Medium Scale Integration)

Zirkuitu integratu hauek 10 atetik 100era dituzte.

### L.S.I. (Large Scale Integration)

Zirkuitu integratu hauek 100 atetik 1.000ra dituzte.

### V.L.S.I. (Very Large Scale Integration)

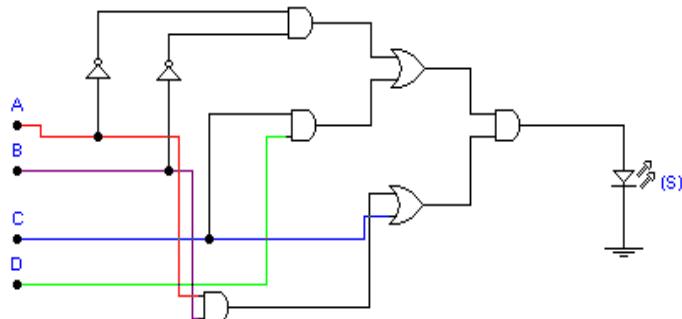
Zirkuitu integratu hauek 1.000 ate logiko baino gehiago dituzte.

## Ariketak

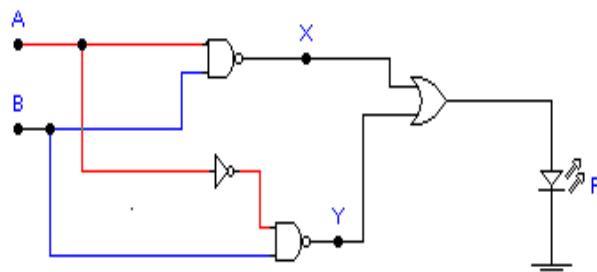
1. Ondorengo funtzio hauek gauzatu behar dituzu ate logikoak erabiliz.

- $S = (a \cdot b + c \cdot d) \cdot \bar{c}$
- $F = (\bar{a} \cdot \bar{b} + c \cdot d) \cdot (a \cdot b + c)$
- $S = \overline{(a \cdot b + a \cdot \bar{b})} \cdot c \cdot \overline{(a \cdot c + a \cdot \bar{c})}$
- $R = [(a + b) \cdot (\bar{a} + \bar{b}) \cdot (\bar{b} + c)]$

2. Irudian agertzen den zirkuituan irteerako ekuazioa (S) lor ezazue.



3. Irudian agertzen den zirkuitua aztertu behar duzu, eta izendatutako puntu bakoitzari dagokion ekuazioa lortu.



B	A	X	Y	F
0	0			
0	1			
1	0			
1	1			

# 3. KAPITULUA

## Funtzioen laburpena

---

### 3.1. SARRERA

---

Zirkuitu digitalen diseinua gauzatzean, egia-taulatik ateratako funtzioak laburtzea oso garrantzitsua da, zeren eta zenbat eta gehiago laburtu beharrezko diren osagaiai gutxiago izango baitira.

Funtzioak laburtzeko erarik ezagunenak hauxek dira:

- Metodo aljebraikoa
- Karnaughen metodo grafikoa

#### 3.1.1. METODO ALJEBRAIKOA

Metodo honetan, ez dago laburpena egiteko araurik. Ahal den neurrian, Booleren aljebraren propietateak erabiliz, funtzia laburtzea lortu behar dugu.

Adibidez:

Demagun ondorengo ekuazioa daukaguila:

$$F = a + b \cdot \left( \overline{\overline{a} \cdot c} \right)$$


Markatutako gaiari Morganen legeak aplikatzen badizkiogu:

$$\left( \overline{\overline{a} \cdot c} \right) \longrightarrow a + c$$


Arestian aipatu dugunarekin jarraituz:

Ordezkatze-gaia ekuazioan sartzen badugu:

$$F = a + b \cdot (a + c)$$

Banatze-, trukatze- eta elkartze-proprietateak hurrenez hurren aplikatuz :

$$F = a + [(b \cdot a) + (b \cdot c)] = a + [(a \cdot b) + (b \cdot c)] = a + (a \cdot b) + (b \cdot c)$$

Xurgatze-proprietateak esaten du:

$$a + (a \cdot b) = a$$

Beraz, ekuazioa honela geratuko da :

$$F = a + (b \cdot c)$$

Eta 14. orrialdean azaldu den irizpideari jarraituz:

$$F = a + b \cdot c$$

### 3.1.2. KARNAUGHEN METODO GRAFIKOA

Karnaugh-en metodo grafikoa gehienez bost aldagai dituzten funtzioak laburtzeko modu erraza da. Sei eta zazpi aldagai dituztenentzat ere balio duen arren, kasu horietan ez du merezi metodo hau erabiltzea, zeren eta aurrerago ikusiko dugunez, dituen zaitasunak eskaintzen dituen abantailak baino ugariagoak baitira. Beraz, bost aldagai baino gehiago dituzten funtzioetarako hobe da zenbakien metodoa erabiltzea (QUINE- McCLUSKEYrena).

Metodo hau erabiltzeko,  $2^n$  oinarrizko laukitzotan banatuko dugun laukia egin behar dugu, n berretzailea funtzioaren aldagai-kopurua izanik.

3.1. irudian dira, hurrenez hurren, bi, hiru eta lau aldagaiko funtzioak laburtzeko grafikoak. Aldagai-kopuru horiexetarako merezi du metodoak.

a  
b

0	0	1
1		

a)

a b  
c

00	01	11	10
1			

b)

c)

**3.1. irudia.** Funtzioak laburtzeko Karnaughen mapak  
a) Bikoa    b) Hirukoa    c) Laukoa

Irudian ikusten denez (**3.1. irudian**), goialdean eta ezkerraldean ipintzen dira aldagai bitarren bi balioak emanda egin daitezkeen konbinazio guztiak. Konbinazioen kokapena ordena batetik hurrengora aldagai bakar baten balioa aldatzeko moduan jarri behar da. Hala eginez gero, auzokidetasun aljebraikoa dagoela esaten da. Beraz, bi, hiru eta lau aldagaiko kasuetan erabat bat datozen auzokidetasun aljebraikoa eta grafikoa. Lehenengo laukitxoa, horizontalki zein bertikalki auzokidea da azkenarekiko.

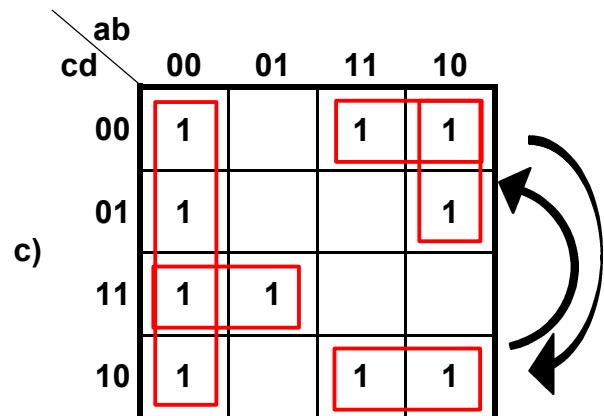
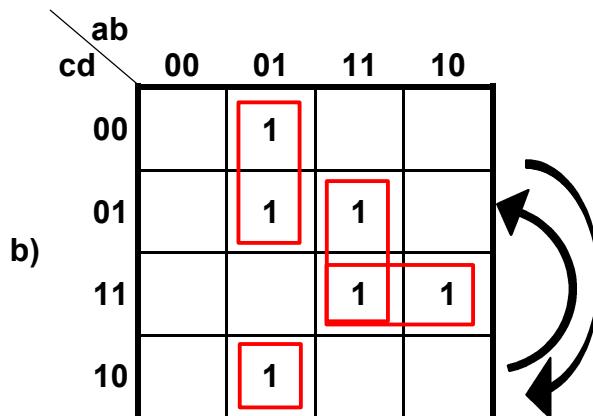
Funtzio bat laburtzeko egin behar diren urratsak zehatz-mehatz azaltzeko kasu jakin bat hartuko dugu. Esate baterako, demagun, problema baten planteamenduari jarraiki, **3.2. irudiko** egia-taula atera dugula.

$$S_1 = \bar{a} \cdot b \cdot \bar{c} \cdot \bar{d} + \bar{a} \cdot b \cdot \bar{c} \cdot d + \bar{a} \cdot b \cdot c \cdot \bar{d} + a \cdot \bar{b} \cdot c \cdot d + a \cdot b \cdot \bar{c} \cdot d + a \cdot b \cdot c \cdot d$$

$$S_0 = (a + b + c + d) \cdot (a + b + c + \bar{d}) \cdot (a + b + \bar{c} + d) \cdot (a + b + \bar{c} + \bar{d}) \cdot \\ \cdot (a + \bar{b} + \bar{c} + \bar{d}) \cdot (\bar{a} + b + c + d) \cdot (\bar{a} + b + c + \bar{d}) \cdot (\bar{a} + b + \bar{c} + d) \cdot \\ \cdot (\bar{a} + \bar{b} + c + d)$$

a	b	c	d	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

a)



3.2. irudia. a) Egia-taula; b)  $S_1$ -en irudikapena; c)  $S_0$ -ren irudikapena  
OHARRA: ↗ Berrantolaketarako ordena-aldaketak

Eta hona hemen funtzioa laburtzeko eman behar diren arauak:

- Lehenengo eta behin, bateko bana ipintzen da funtzioa dagoen laukitxo bakoitzean 3.2. irudian ikusten den bezala, lehen zein bigarren forma kanonikoan.
- Gero batekoak hartu eta **2, 4, 8 eta 16** laukitxoko taldeetan biltzen dira. Talde horiek osatzeko beharrezkoa da batekoak laukitxo auzokideetan egotea.

**Hainbat talde ezberdin egin daitezke. Bateko berbera hainbat taldetakoa izan daiteke. Ahalik eta talde-kopururik txikiiena egiten eta horietako bakoitzean ahalik eta bateko kopururik handiena biltzen ahalegindu behar dugu.**

Gure kasuan, 3.2. irudian erakusten diren taldeak hautatu ordez, 3.3 irudikoak hautatu dira, berrantolaketa bat dela medio, horretara talde-kopuru txikiiena lortzen baita.

a b	00	01	11	10
c d	00	1		
10		1		
11			1 1	
01		1 1		

Diagram showing minterms highlighted in red boxes:  $\bar{a} \cdot b \cdot \bar{d}$ ,  $b \cdot \bar{c} \cdot d$ , and  $a \cdot c \cdot d$ .

a b	00	01	11	10
c d	00	1	1	1
10	1		1	1
11	1	1		
01	1			1

Diagram showing minterms highlighted in red boxes:  $a + b$ ,  $a + \bar{c} + \bar{d}$ ,  $\bar{a} + d$ , and  $b + c + \bar{d}$ .

3.3 irudia. a)  $S_1$ -en irudikapen berrantolatua b)  $S_0$ -en irudikapen berrantolatua

- Bateko talde bakoitzari funtzio murriztuaren formularen gai bat dagokio. Talde bakoitzean kendu egiten dira beren bi balioekin (**0** eta **1**) parte hartzen duten aldagaiak.
- Funtzioaren  $F_1$  adierazpen murriztua lortzeko, aldagaiaren balio osagarria idatziko dugu dagokion balioa “**0**” denean; balioa “**1**” denean, ostera, era zuzenean agertuko da aldagai.

Funtzioaren  $F_0$  adierazpen murriztua lortzeko, aldiz, aurrekoaren aurkako irizpideari jarraituko zaio.

Emaitzak hauek dira:

$$\boxed{\begin{aligned} F_1 &= \bar{b} \times \bar{c} \times d + a \times c \times \bar{d} + \bar{a} \times b \times \bar{d} \\ F_0 &= (\bar{a} + b) \times (\bar{a} + d) \times (\bar{a} + \bar{c} + \bar{d}) \times (b + c + \bar{d}) \end{aligned}}$$

### 3.2. FUNTZIOEN OSAKETA NAND ATEEN BIDEZ

Hona hemen edozein funtzio-mota **NAND** ateez bakarrik osaturiko adierazpen aljebraiko bihurtzeko egin behar den prozesua:

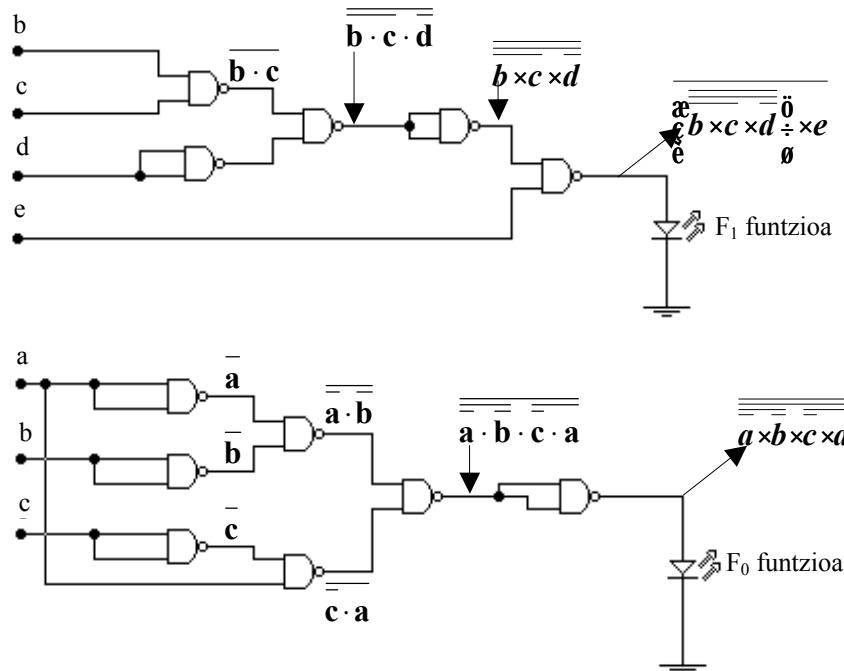
- Lehenengo eta behin, adierazpen osoari alderanzketa bikoitza egin behar zaio, alderanzketa edo ezeztapen horietako batekin Morganen legeak aplikatu ahal izan ditzagun, halakorik behar izanez gero (batuketaren kasuan gertatuko da behar hori).
- Morganen legeak aldi bakoitzean bi aldagairi baino ez zaizkio aplikatuko, bi sarrerako NAND atea erabiliko baititugu, eta ondorioz, adierazpena beti bi zatitan banatu behar da, transformazioa egin ahal izateko.
- Morganen legeak aplikatzu egiten diren transformazioen helburua biderketa ezeztatuak (NAND) lortzea denez, 1) biderketak ezeztatuak agertzen badira, alderanzketa bikoitza ezarriko zaie, eta 2) batuketak agertzen badira, biderketa ezeztatu bihurtuko dira, aipatutako legeak aplikatzu.

Adibideak:

$$1. \quad F_1 = b \times c + d + \bar{e} = \overline{\overline{b} \times c + d + e} = \overline{(\overline{b} \times c + d) \times e} = \overline{(\overline{b} \times c \times \bar{d}) \times e} = \overline{\overline{\overline{\overline{b}} \times \overline{c} \times \overline{d}} \times \overline{e}}$$

$$2. \quad S_2 = (\bar{a} + b) \cdot (\bar{c} + \bar{a}) = \overline{(\bar{a} + b) \cdot (\bar{c} + \bar{a})} = \overline{(\overline{\bar{a} + b})} \cdot \overline{(\overline{\bar{c} + \bar{a}})} \\ F_0 = \overline{\overline{\overline{a}} \cdot \overline{b} \cdot \overline{c} \cdot \overline{a}}$$

Ondorengo irudian erakusten dira funtzioen eskema logikoak eraldatu ondoren.



### 3.3. FUNTZIOEN OSAKETA NOR ATEEN BIDEZ

Segitu behar den prozesua aurreko atalean azaldutakoaren antzekoa da.

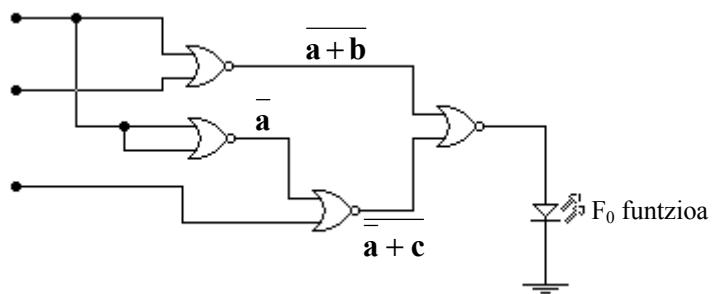
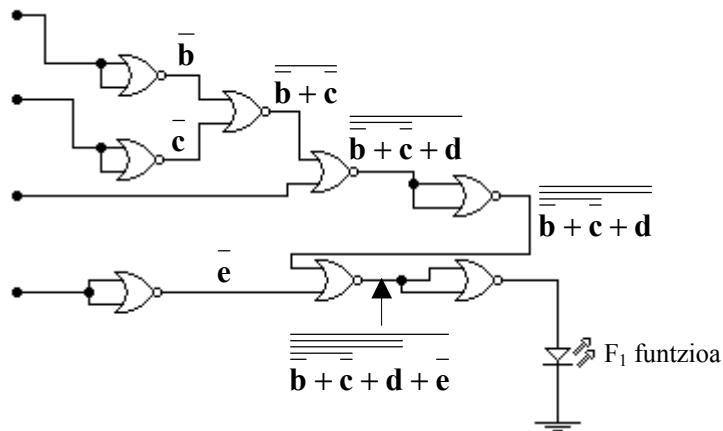
- Lehenengo eta behin, adierazpen osoari alderanzketa bikoitza egin behar zaio, alderanzketa edo ezeztapen horietako batekin Morganen legeak aplikatu ahal izan ditzagun, halakorik behar izanez gero (biderketaren kasuan gertatuko da behar hori).
- Morganen legeak aldi bakoitzean bi aldagairi baino ez zaizkio aplikatuko, bi sarrerako NOR ateak erabiliko baititugu, eta ondorioz, adierazpena beti bi zatitan banatu behar da, transformazioa egin ahal izateko.
- Morganen legeak aplikatz egiten diren transformazioen helburua **batuketa ezeztatuak** (NOR) lortzea denez, 1) batuketak ezeztatu gabe agertzen badira, alderanzketa bikoitza ezarriko zaie, eta 2) biderketak agertzen badira, batuketa ezeztatu bihurtuko dira, aipatutako legeak aplikatz.

Adibideak:

$$1. \quad F_1 = b \cdot c + d + \bar{e} = \overline{\overline{b \cdot c + d + \bar{e}}} = \overline{\overline{b \cdot c + d}} + \bar{e} = \overline{\overline{(b \cdot c + d)}} + \bar{e} = \\ = \overline{\overline{\overline{b + c}} + d} + \bar{e}$$

$$2. \quad F_0 = (a + b) \cdot (\bar{a} + c) = \overline{(a + b)} \cdot \overline{(\bar{a} + c)} = \\ = \overline{(a + b)} + \overline{(\bar{a} + c)}$$

Irudian erakusten dira funtzioko horiei dagozkien zirkuituak :



## Ariketak

1. Eralda itzazu hurrengo funtzioak, NAND ateet eraikiak izan daitezene.

$$F_1 = \overline{a \cdot \bar{b} \cdot c \cdot d + a \cdot b \cdot c}$$

$$F_2 = \overline{(a + b)} \cdot \overline{(a + \bar{b} + c)}$$

2. Eralda itzazu hurrengo funtzioak NOR ateet bidez eraikiak izan daitezene.

$$X_1 = \overline{\overline{a \cdot \bar{b} \cdot c} + \overline{a \cdot c}}$$

$$X_2 = \overline{(a + b)} \cdot \overline{(a + b + c)}$$

3. Labur itzazu ondoren agertzen diren funtzioak, Karnaughen metodoa erabiliz

$$P_1 = \bar{a} \cdot b \cdot c + a \cdot b \cdot c + \bar{a} \cdot \bar{b} \cdot \bar{c} + a \cdot \bar{b} \cdot c$$

$$P_2 = a \cdot b + b \cdot c + \bar{a} \cdot b + a \cdot \bar{b} \cdot c$$

$$P_3 = a \cdot b \cdot c + c \cdot d + a \cdot b \cdot c \cdot d + a \cdot b \cdot c$$

4. Hurrengo espezifikazio teknikoak betetzen dituen gela bateko klimatizazio-sistema kontrolatzeko konbinaziozko zirkuitua diseinatu.

- Barneko tenperatura 25 °C baino handiagoa denean, hozkailua pizten da, eta 15 °C baino txikiagoa denean, berogailua jarriko da martxan.
- Lehendabiziko kasuan kanpoko tenperatura barrukoa baino txikia-goa baldin bada (hozketa), edo bigarren kasuan handiagoa bada (beroketa), dagokion sistemaz aparte (hozkailua/berogailua), haizagailuak jarriko dira martxan.
- Jaiegunetan sistema geldirik egongo da edozein baldintzatan.

Zirkuitua NAND ateet bidez eraiki.

5. Prentsa hidrauliko bat martxan jartzeko, hiru pultsadore (P1, P2 eta P3) aktibatu behar dira aldi berean. Bi bakarrik sakatzen badira ere martxan jartzen da, baina erabilera okerra adierazten duen lanpara ere piztu egiten da. Bat bakarrik sakatzen denean, lanpara pizten da eta prentsa ez da martxan jartzen.

Diseina ezazue kontroleko zirkuitua, bi sarrerako NAND ateak erabiliz.

6. Lau sarrera (A1, A2, A3 eta A4) eta hiru irteera (X1, X2 eta X3) dituen zirkuitu bat eraiki nahi da. Zirkuitu horrek zein sarrera dagoen aktibatua adieraziko digu, sarreren azpindizearen baliokide bitarra irteeran emanez. Sarrera bat baino gehiago daudenean aktibaturik, azpindize txikienak du lehentasuna.



# 4. KAPITULUA

## Talde logikoak

### 4.1. SARRERA

Teknologia ugari erabiltzen da zirkuitu digital integratuak egiteko. Hots, RTL talde logikoa, DTL talde logikoa, TTL talde logikoa, PMOS talde logikoak, NMOS talde logikoa, CMOS talde logikoa eta HCMOS talde logikoa, besteak beste. Hala ere, bi garrantzitsuenak baino ez ditugu jorratuko: TTL eta CMOS. Horiexek dira, hain zuen ere, erabilienak eta aukera zabalenekoak.

Talde logikoak aztertu baino lehen, zirkuitu integratuetan egileek ematen dituzten ezaugarri nagusiak zein diren jakitea komeni zaigu.

### 4.2. ATE INTEGRATUEN EZAUGARRI NAGUSIAK

Fabrikatzaileek ezaugarri-katalogoetan adierazten dizkiguten balio edo parametro garrantzitsuenak, honakoak dira:

- a) **Elikatze-tentsioa eta horren perdoia**
- b) **Laneko temperatura**
- c) **Fan-out edo irteera-kopurua**
- d) **Fan-in edo sarrera-kopurua**
- e) **Sarrerako eta irteerako maila**
- f) **Zarataren tartea 1 eta 0 maila logikoetan**
- g) **Batez besteko hedapen-denbora**
- h) **Potentzia-disipazioa**
- i) **Elikatze-intentsitatearen kontsumoa**

Talde logiko bakoitzak (TTL, CMOS...) abantailak eta eragozpenak ditu bestekiko. Hori dela eta, diseinuaren arabera egokituko dugu aukera.

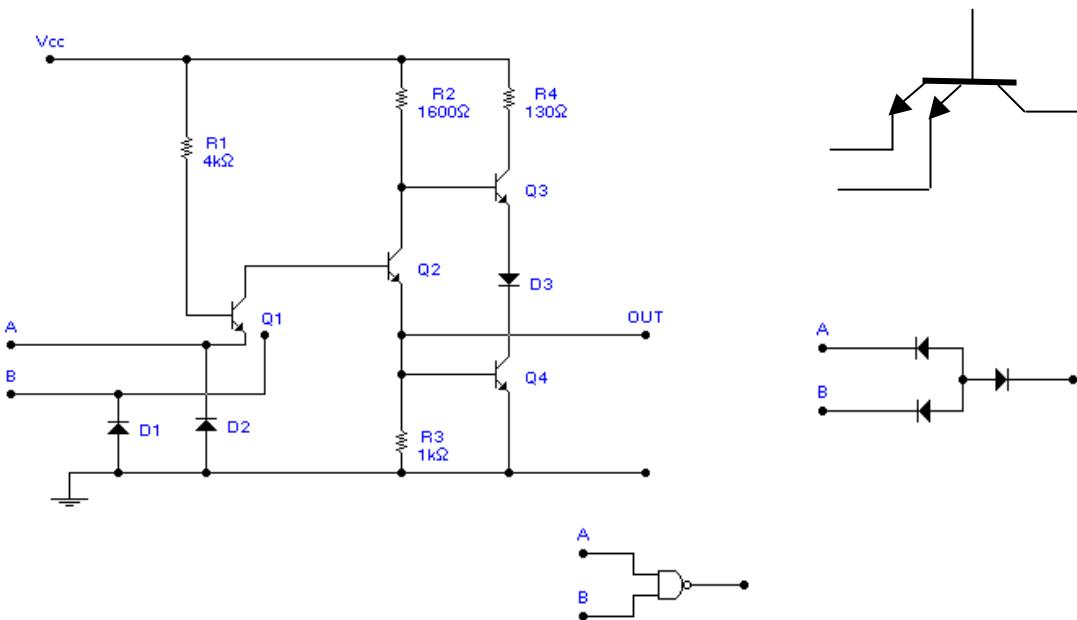
Talde logiko integratuan ezaugarri benetan bikainak honako hauek lirateke:

- **Integrazio-dentsitate handia**

- Hedapen-denbora txikia
- Immunitatea zaratarekiko eta temperaturarekiko
- Bateragarritasuna beste talde logikoekiko
- Kostu txikia, merkea izatea

### 4.3. TTL TALDE LOGIKOA

TTL izena ingelesezko *Transistor-Transistor Logic* gaitik datorkigu, eta atea erresistentziaz, diodoz eta transistorez osatuta daude



4.3. irudia. NAND funtziaren barneko egitura.

**TTL taldea** hainbat seriez osaturik dago, non bigarren serieak lehenengoaren ezaugarriak hobetuta izaten baititu, eta horrela seriea osatu arte. Lehenengo seriea eta horren ezaugari garrantzitsuenak hauxek dira:

* Elikatze-tentsioa:	5 volt
* Laneko temperatura:	0- 70 °C
* Fan-out :	10
* Zarata-tartea:	0,4 volt
* Batez besteko hedapen-denbora:	10 ns
* Potentzia-disipazioa:	10 mW ateko

54 seriearen laneko temperatura -55 °C eta +125 °C bitartekoa da, eta lan berezietan erabiltzen da, 74 seriea baino garestiagoa baita.

**54/74 L (low-power).** Potentzia-disipazioa txikiagoa (1 mW) baina hedapen-denbora handiagoa (33 ns).

**54/74 S (Schottky).** Hedapen-denbora txikiagoa (3 ns) baina potentzia-disipazioa handia (19 mW).

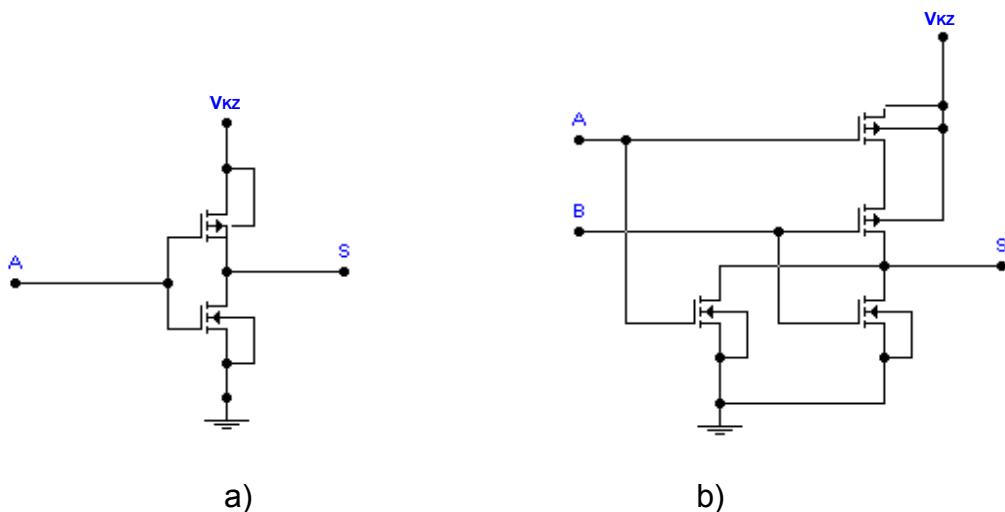
**54/74 LS (Low-power gehi Schottky).** Potentzia-disipazioa 2 mW eta hedapen-denbora 5 ns.

**54/74 F (Fast).** Potentzia-disipazioa 4 mW eta hedapen-denbora 3 ns.

## 4.4. CMOS TALDE LOGIKOA

Bere oinarrizko osagaia MOS (*Metal-Oxide-Semiconductor*) izeneko transistorea delako deitzen zaio horrela.

CMOS taldeko zirkuitu integratuek lehia gogorra egiten diete TTL teknologiaz egindakoei, hainbat aldetatik begiratuta dituzten abantailak direla eta. Abantaila nagusia honakoa da: funtzionario, potentzia-disipazio txikiagoa dute eta, ondorioz, integracio-dentsitate handiagoa dute. Horrez gainera, talde honek TTLk baino immunitate handiagoa dauka zarata elektrikoarekiko. Aitzitik, hedapen-denborak handiagoak dira.



4.4. irudia. a) NOT atea; b) NOR atea

Hona hemen familia honen ezaugarri nagusiak:

- \* **Elikatze-tentsioa aldakorra:**
- \* **Laneko temperatura:**
- \* **Fan-out :**
- \* **Zaratarekiko immunitate handia**

- 3 eta 18 V bitartekoak**
- 40 °C-tik+ 85 °C-ra**
- bitartekoak**
- 50etik gorakoa**

- \* Hedapen-denbora elikaduraren arabera  $V_{KZ} = 18 \text{ V} \text{ --- } 45 \text{ ns}$   
 $V_{KZ} = 3 \text{ V} \text{ --- } 125 \text{ ns}$
- \* Potentzia-disipazioa: 10 nW ateko

CMOS taldea 4000 serie bezala agertzen da fabrikatzaileen katalogoetan. Badaude hemen ere geroztik egindako hobekuntzak; horien ondorioz, **High - Speed CMOS** da aipagarriena, zeren bere elikatze-tentsioa 4,5 volt izanik bere hedapen-denbora 7 ns baita.

#### 4.5. TTLren ETA CMOSen ARTEKO KONPARAZIOA

Dagoeneko konturatu gara txipen fabrikatzaileak zertaz arduratu diren: potentzia-disipazioa gutxitzeaz, hau da, integracio-dentsitatea handitzeaz. Ahalik eta zirkuitu integral txikienak lortzeko, barnean osagai-kopuru handia ezartzen da.

Hedapen-denbora laburtuz, hau da abiadura handituz, eragiketa-denbora laburragoa izan dadin.

Seriea	Potentzia-disipazioa	Hedapen-denbora
TTL	10 mW	10 ns
TTL (L)	1 mW	33 ns
TTL (S)	19 mW	3 ns
TTL (LS)	2 mW	5 ns
TTL (F)	4 mW	3 ns
CMOS	10 mW	125 ns
H-S CMOS	---	7 ns

CMOSen abantailak:

- Elikatze-tentsioaren tarte handia
- Laneko temperaturaren tarte handia
- Fan-out handia
- Zaratarekiko inmunitatea handia
- Potentzia-disipazioa txikia

TTLren abantailak:

- Hedapen-denbora txikia
- Bloke- edo zirkuitu-kopuru handia
- Kostu txikia

Signetics

7402, LS02, S02  
GatesQuad Two-Input NOR Gate  
Product Specification

Logic Products

TYPE	TYPICAL PROPAGATION DELAY	TYPICAL SUPPLY CURRENT (TOTAL)
7402	10ns	11mA
74LS02	10ns	2.2mA
74S02	3.5ns	22mA

## ORDERING CODE

PACKAGES	COMMERCIAL RANGE $V_{CC} = 5V \pm 5\%$ ; $T_A = 0^\circ C$ to $+70^\circ C$
Plastic DIP	N7402N, N74LS02N, N74S02N
Plastic SO	N74LS02D, N74S02D

## NOTE:

For information regarding devices processed to Military Specifications, see the Signetics Military Products Data Manual.

## FUNCTION TABLE

INPUTS		OUTPUT
A	B	Y
L	L	H
L	H	L
H	L	L
H	H	L

H = HIGH voltage level  
L = LOW voltage level

## INPUT AND OUTPUT LOADING AND FAN-OUT TABLE

PINS	DESCRIPTION	74	74S	74LS
A, B	Inputs	1 <ul style="list-style-type: none"></ul>	1 <ul style="list-style-type: none"></ul>	1 <ul style="list-style-type: none"></ul> S <ul style="list-style-type: none"></ul>
Y	Output	10 <ul style="list-style-type: none"></ul>	10 <ul style="list-style-type: none"></ul> S <ul style="list-style-type: none"></ul>	10LS <ul style="list-style-type: none"></ul>

## NOTE:

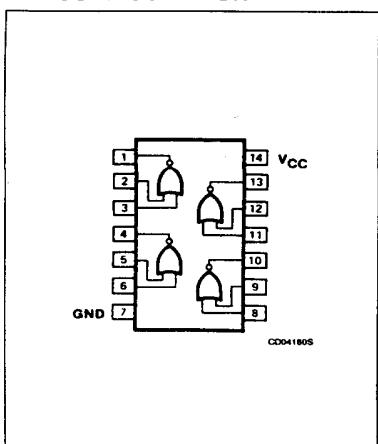
Where a 74 unit load (


) is understood to be  $40\mu A I_{IH}$  and  $-1.6mA I_{IL}$ , a 74S unit load (


S) is  $50\mu A I_{IH}$  and  $-2.0mA I_{IL}$ , and 74LS unit load (

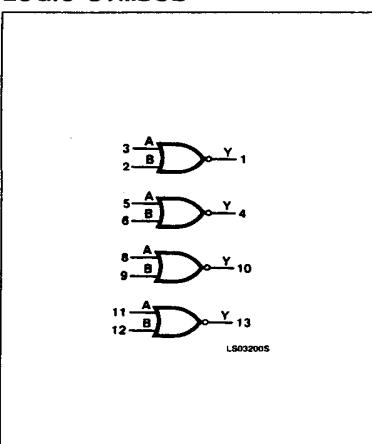

LS) is  $20\mu A I_{IH}$  and  $-0.4mA I_{IL}$ .

## PIN CONFIGURATION



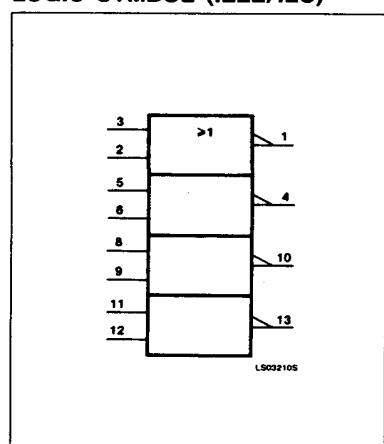
December 4, 1985

## LOGIC SYMBOL



5-9

## LOGIC SYMBOL (IEEE/IEC)



853-0502-81501

## QUADRUPLE 2-INPUT NOR GATE

The HEF4001UB is a quadruple 2-input NOR gate. This unbuffered single stage version provides a direct implementation of the NOR function. The output impedance and output transition time depends on the input voltage and input rise and fall times applied.

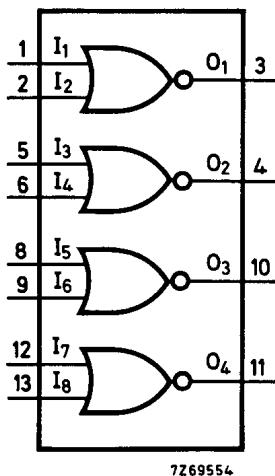


Fig. 1 Functional diagram.

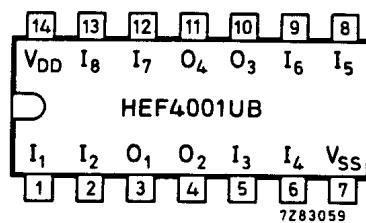


Fig. 2 Pinning diagram.

HEF4001UBP(N): 14-lead DIL; plastic  
(SOT27-1)

HEF4001UBD(F): 14-lead DIL; ceramic (cerdip)  
(SOT73)

HEF4001UBT(D): 14-lead SO; plastic  
(SOT108-1)

( ): Package Designator North America

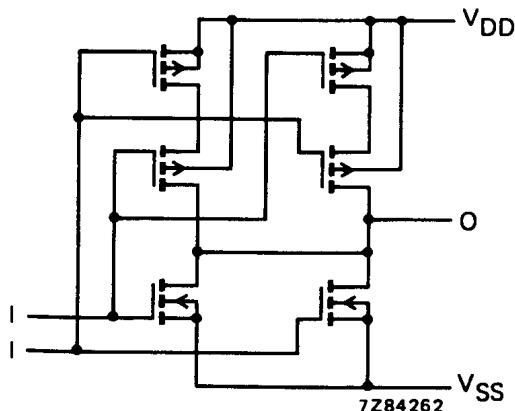


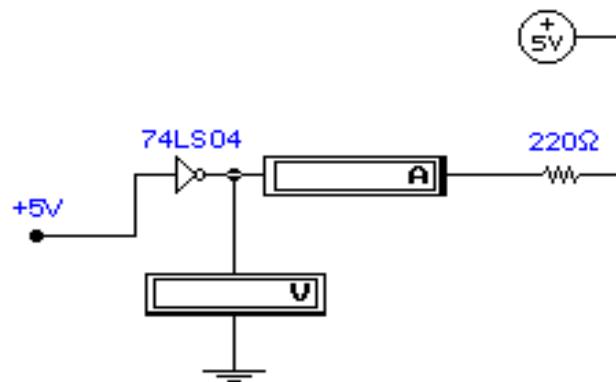
Fig. 3 Schematic diagram (one gate). The splitting-up of the p-transistors provide identical inputs.

**FAMILY DATA**
**I<sub>DD</sub> LIMITS category GATES**

see Family Specifications for V<sub>IH</sub>/V<sub>IL</sub> unbuffered stages

*Ariketak*

1. Osa ezazue taula irudian agertzen den zirkuituan behar dituzuen kalkuluak egin ondoren.



## TTL talde logikoa

Erresistentzia (U)	Irteerako tentsioa	Irteerako korrontea
220		
180		
150		
120		
100		

# 5. KAPITULUA

## Konbinaziozko zirkuituak

### 5.1. SARRERA

Edozein zirkitu logikotan, irteera-egoera sarrera-egoeraren funtzioa denean eta sarreren aurretiko egoerak eraginik ez duenean, zirkitu horri **konbinaziozko zirkitu** deitzen zaio.

Oinarrizko konbinaziozko zirkuituak atea dira, baina zirkitu integratuen bidez egindako beste zirkitu-mota batzuk badira. Hemendik aurrera, hainbat gaitan zehar, zirkitu-mota hauek ikasiko ditugu.

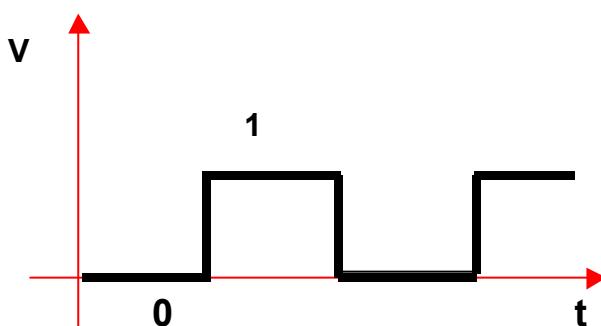
Konbinaziozko zirkuituen sailkapena honakoa da:

- **Komunikazioa lortzeko zirkuituak:**
  - a) **Deskodetzaileak**
    - Ez kitzikatzaileak
    - Kitzikatzaileak
  - b) **Kodetzaileak**
    - Lehentasunik gabeak
    - Lehentasundunak
  - c) **Kode-bihurgailuak**
  - d) **Multiplexadoreak**
  - e) **Demultiplexadoreak**
  - f) **Paritate-bitaren sorgailu/detektoreak**
- **Zirkitu eragileak:**
  - a) **Konparadoreak**
  - b) **Batutzaileak**
  - c) **Kentzaileak**
  - d) **UAL unitatea**

## 5.2. KODEKETA ETA DESKODEKETA

Zirkuitu digital guztien funtzionamendua beren sarreretan seinale digitalak ezartzea eta irteeran seinale digitalak ateratzea izaten da.

Jakina denez, seinale digitalak bi tentsio-mailaz osatuta daude (maila altua eta maila baxua), **5.1. irudian** agertzen den bezala. Bi maila horiek, berriz, atea eta gainerako bloke integratuak osatzen dituzten oinarrizko osagai elektroteknikoen bi egoera egonkorrei (ebakidura eta asetasuna) dagozkie.



**5.1. irudia.** Seinale digitala.

Sistema digitalen bidez tratatu, prozesatu eta metatu nahi den edozein informazio, lehenengo, lengoia-mota apropos batera “**itzuli**” **edo kodetu** beharko da, eta hori egiteko modu egokia edozein zenbaki, hitz, zeinu, agindu edo eragiketa seinale elektriko digitaleko multzo bihurtzea da. Kasuaren arabera, multzoa ezberdina izango da.

Era berean, zirkuitu digital bateko irteerako seinalea ulertzeko ere, emaitza **berritzuli edo deskodetu** egin beharko dugu; horretarako zeroak eta batekoak ulertzeko moduko datu bihurtuko ditugu.

Kodeketa eta deskodeketa ezinbesteko eragiketak dira informazioaz diharduten sistema digitaletan eta datuak edo emaitzak aurkeztu beharra duten prozesu industrialetan. Zenbait sistema zibernetikotan edo kontrolekotan, ostera, ez dira behar izaten eragiketa horiek eta nahikoa izaten da “**bihurgailuen**” bidez seinale digitalak ezartzea eta zirkuituaren irteerak, lanpara, motorra eta horrelako potentzia-gailuetara konektatzea.

### 5.2.1. SISTEMA BITARRA

Bi oinarri duen sistemak 0 eta 1 balioak besterik ez ditu erabiltzen. Zenbaki bitar baten zifra edo digitu bakoitzari bit deitzen zaio, zeina *Binary Digit*-en akronimoa baita.

Zenbaki bitarra era polinomikoan ere adieraz daiteke:

$$N = A_n B^n + A_{n-1} B^{n-1} + \dots + A_1 B^1 + A_0 B^0 + A_{-1} B^{-1} + A_2 B^{-2} \dots$$

Koefizienteek zenbakiaren zifrak adierazten dituzte, hurrenez hurren, eta **B** zenbaki-sistemaren oinarria, kasu honetan, **B = 2** da.

Gaiak 10 oinarrian adierazten badira eta denak batzen badira, adierazten duen zenbaki bitarraren baliokide hamartarra lor daiteke.

Adibideak:

a) Bi oinarria duen **101101** zenbakia hamartarretan eman:

$$1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 45$$

$$101101_2 = 45_{10}$$

b) Bi oinarria duen **0,1001** zenbakia hamartarretan eman:

$$1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 0 \cdot 2^{-3} + 1 \cdot 2^{-4} = 0.5625$$

$$0,1001_2 = 0,5625_{10}$$

c) Bi oinarria duen **10101,011** zenbakia hamartarretan eman:

$$10101,011_2 = ?_{10}$$

### 5.2.2. SISTEMA ZORTZITARRA

Sistema oktala sistema zortzitarra dugu, beraz, kasu honetan **B = 8** eta koefizienteak **0**tik **7**ra bitartekoak izango dira. Sistema honen garrantzia honetan datza: oso erraza da sistema honetatik bitarrera pasatzea, eta honela egin ohi da:

Sistema zortzitarra	Sistema bitarra
0	0 0 0
1	0 0 1
2	0 1 0
3	0 1 1
4	1 0 0
5	1 0 1
6	1 1 0
7	1 1 1

**347,6<sub>8</sub>** zenbakia bitarrean ezarri nahi badugu, zenbaki bakoitza sistema bitarrean ipini beharko dugu hiru zifrako zenbakien bidez. Hau da:

$$\begin{array}{lcl}
 3 & = & 0 1 1 \\
 4 & = & 1 0 0 \\
 7 & = & 1 1 1 \\
 6 & = & 1 1 0
 \end{array}
 \quad \xrightarrow{\text{beraz,}}$$

$$347,6_8 = 011100111,110_2$$

Alderantziz nahi izanez gero, berdin egiten da:

$$110111,1111_2$$

$$\begin{array}{lcl}
 110 & = & 6 \\
 111 & = & 7 \\
 111 & = & 7 \\
 100 & = & 4
 \end{array}
 \quad \xrightarrow{\text{beraz,}}$$

$$110111,1111_2 = 677,4_8$$

### 5.2.3. SISTEMA HAMASEITARRA

Sistema hexadezimala sistema hamaseitarra da, beraz, **b = 16** eta horregatik 16 koefiziente behar ditu. Koefiziente horiek zenbaki batez adierazten direnez, eta hamasei zenbaki ez ditugunez, letrak hartzen dira. Sistema horretako koefizienteak honakoak dira:

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

Taulan agertzen dira sistema bitarreko eta hamaseitarreko kodeen baliokide-tasunak:

Sistema bitarra	Sistema hamaseitarra
000	0
0001	1
10010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Sistema hamaseitarretik bitarrera pasatzeko, zortzitarrean egindakoa egiten da, baina, kasu honetan, kontuan hartu behar dugu taldeak laukoak izan beharko dutela.

$$13A,FA_{16} = ?_2$$

1	=	0001
3	=	0011
A	=	1010
F	=	1111
A	=	1010



beraz,

$$13A,FA_{16} = 000100111010,11111010_2$$

### 5.2.4. KODEAK

Oro har, kodea, informazioaren unitatea-multzoa da, zeinu eta ikur batzuekin era sistematiko eta biunibokoan erlazionatzen dena, aurrez finkaturiko itzulpen-araau jakin batzuen arabera. Sistema dигaletako kodeak bitarrak dira, hau da, zeroaren eta bataren konbinazioak.

Kode daitekeen informazioa, zenbakizko kopuruei buruzkoa ez ezik, zeinu operatibo, letra, agindu eta abarri buruzkoa ere izan daiteke.

Lehen, **9** zenbaki hamartarra sistema bitarrean **1001** zenbakiaz adierazten dela esan dugunean, honakoa esan nahi genuen: **9** zenbakia horrela kodetzen dela informazioa zirkitu logikoren bati ezartzeko. Horrelako erlazioa egiten dugunean, 0,1,2,3,4 eta abarren, eta adierazitakoen moduko konbinazio bitarren multzoaren artean kode posibileetako bat sortzen dugu.

Hauexek dira koderik arruntenak:

- a) Bitar naturala
- b) BCD (kode bitarrean kodetutako kode hamartarra)
  - Naturala
  - Gehi hirukoa
  - Aiken
- c) Kode progresiboak
- d) Detektatzeko kodeak eta errore-zuzentzaileak
- e) Kode alfanumerikoak

Sistema hamartarra	BCD naturala	BCD gehi hirukoa	BCD Aiken
0	0000	0011	0000
1	0001	0100	0001
2	0010	0101	0010
3	0011	0110	0011
4	0100	0111	0100
5	0101	1000	1011
6	0110	1001	1100
7	0111	1010	1101
8	1000	1011	1110
9	1001	1100	1111

**5.2. irudia.** Sistema hamartarraren eta BCD familiako kodeen arteko baliokidetasunak.

Lehen aipatutakoarekin jarraituz:

- **BCD naturalean** batetik hasita lehenengo hamarrak hartzen dira.
- **BCD gehi hirukoan** ez dira hartzen, ez lehen hirurak, ezta azken hirurak ere.
- **BCD Aiken** sistemek lehen bostak nahiz azken bostak hartzen dira

Aurreko 5.2. irudian erakusten dira hiru familiok eta sistema hamartarrarekin duten erlazioa.

### c) Kode progresiboak

Kode hauen ezaugarri nagusia honakoa da: konbinazio bakoitzak bere aurrekoarekin eta ostekoarekin bit bakarreko aldea izatea. Prozesu industrialetan erabiltzen dira magnitude analogikoak digital bihurtzeko Honelako koderik erabilienak **Gray** kodeak dira. Kode horien arteko ezberdintasuna konbinazioen bit-kopurua izaten da. Esate baterako, doitasun handia behar denean, konbinazio luzeagoak erabiltzen dira. 5.3. irudian duzue lau biteko Gray kodea.

Gray kodea			
0	0000	8	1100
1	0001	9	1101
2	0011	10	1111
3	0010	11	1110
4	0110	12	1010
5	0111	13	1011
6	0101	14	1001
7	0100	15	1000

5.3. irudia. Lau biteko Gray kodea.

### d) Detektatzeko kodeak eta errore-zuzentzaileak

Bada zenbait kode landuagoak izanik sare batek emaniko informazioan okerrik datorren detektatu eta inoiz zuzendu ere egiten duenik. Baino okerra konbinazioko bit bakar batean agertzen denean bakarrik detektatzen edo zuzentzen da. Dena dela, okerra bi bitetan batera agertzea arraro samarra da.

Kode horiek, akats bat detektatzeko, gutxienez bost bit izan behar dituzte. Detektatzeko kode ezagunenak paritatedunak, bi zati bost kodea eta kode bibostarra dira.

Azken biak, hurrenez hurren, bost eta zazpi biteko konbinazioz osatzen dira, eta bietan bi bateko logiko agertzen dira konbinazio bakoitzean.

Adierazitako kodeak 5.4. irudian azter ditzakegu:

2 zati bost kodea	Kode bibostarra
0	11000
1	00011
2	00101
3	00110
4	01001
5	01010
6	01100
7	10001
8	10010
9	10100

5.4. irudia. Detektatzeko kodeak.

Errore-zuzentzaile diren kodeek errorea non dagoen esaten digute. Zirkuitu egokia erabiliz, jasotako informazioan automatikoki detekta daiteke akatsa. Kode horiek prozesu industrialetan erabiltzen dira.

Gehien erabiltzen den errore-zuzentzailea **Hamming**-en kodea da.

<b>HAMMINGen KODEA</b>							
	b7	b6	b5	b4	b3	b2	b1
0	0	0	0	0	0	0	0
1	0	0	0	0	1	1	1
2	0	0	1	1	0	0	1
3	0	0	1	1	1	1	0
4	0	1	0	1	0	1	0
5	0	1	0	1	1	0	1
6	0	1	1	0	0	1	1
7	0	1	1	0	1	0	0
8	1	0	0	1	0	1	1
9	1	0	0	1	1	0	0

5.5. irudia. Hamming-en kodea.

### e) Kode alfanumerikoak

Orain arte ikusiriko kodeetan zenbakiak besterik ez ziren agertzen. Konputagailuetan, letrak eta ikurrak ere kodetu behar izaten dira. Horretarako gehien erabiltzen dena **6 edo 7** biteko **ASCII** kodea dugu.

5.6. irudiak 7 biteko ASCII kodea adierazten du, eta b7 da pisu gehien daukan bita.

Konbinazio bakoitzari ondorengo formatua dagokio:

P	b7	b6	b5	b4	b3	b2	b1

Irudi horretan ASCII kodea ikus daiteke:

	b7	0	0	0	0	1	1	1	1			
	b6	0	0	1	1	1	1	1	1			
	b5	0	1	0	1	0	1	0	1			
b4	b3	b2	b1		0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DEL	SP	0	@	P	'	p
0	0	0	0	1	SOH	DC1	!	1	a	Q	a	q
0	0	1	0	2	STX	DC2	"	2	b	R	b	r
0	0	1	1	3	ETX	DC3	#	3	c	S	c	s
0	1	0	0	4	EOT	DC4	s	4	d	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	e	U	e	u
0	1	1	0	6	ACK	SYN	&	6	f	V	f	v
0	1	1	1	7	BEL	ETB	'	7	g	W	g	w
1	0	0	0	8	BS	CAN	(	8	h	X	h	x
1	0	0	1	9	HT	EM	)	9	i	Y	i	y
1	0	1	0	10	LF	SUB	*	:	j	Z	j	z
1	0	1	1	11	VT	ESC	+	;	k	[	k	{
1	1	0	0	12	FF	FS	.	<	l	\		
1	1	0	1	13	CR	GS	-	=	m	]	M	}
1	1	1	0	14	SO	RS	.	>	N	^	N	~
1	1	1	1	15	SI	US	/	?	O	o	DEL	

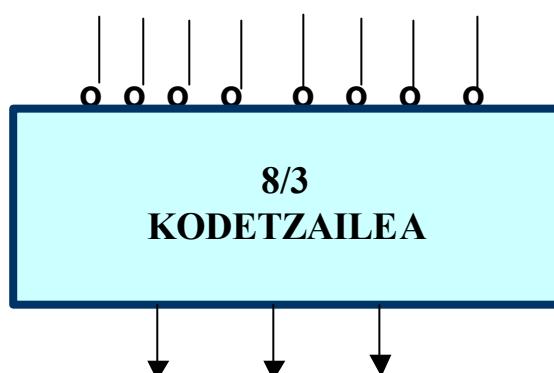
5.6. irudia. ASCII (*American Standard Code for Information Interchange*).

### 5.3. KODETZAILEAK

Kodetzalea konbinazio-zirkuitua da: hainbat sarrera ( $2^n$ ) ditu eta  $n$  irteera. Aktibatutako sarrera zein den, irteeran kode bitarrean adierazten du. Beraz, zirkuitu hauetan sarrera-kopurua irteera-kopurua baino handiagoa izango da. Adibidez, 4 irteera edukiz gero, 16 sarrera eduki ditzake.

5.7. irudian agertzen da  $2^3 = 8$  sarrera eta hiru irteera dituen kodetzale baten egia-taula. Balio aktiboa zero da.

SARRERAK								IRTEERAK		
A7	A6	A5	A4	A3	A2	A1	A0	S2	S1	S0
1	1	1	1	1	1	1	0	0	0	0
1	1	1	1	1	1	0	1	0	0	1
1	1	1	1	1	0	1	1	0	1	0
1	1	1	1	0	1	1	1	0	1	1
1	1	1	0	1	1	1	1	1	0	0
1	1	0	1	1	1	1	1	1	0	1
1	0	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	1	1



5.7. irudia. Kodetzailea.

Bi eratako kodetzaileak ditugu: **lehentasundunak eta lehentasunik gabeak**. Lehenengoetan, bi sarrera batera aktibatuz, zenbaki handiena duena aktibo bezala hartzen da eta besteari ez zaio jaramonik egiten. Egia esanda, merkatuko MSI teknologiako kodetzaileak lehentasundunak dira.

Ildo beretik jarraituz, adibide gisa, lau sarrerako (eta ondorioz, bi irteerako) kodetzaile sinplearen logigrama egingo dugu. Hasteko, egia-taula egin beharko genuke.

SARRERAK				IRTEERAK	
A3	A2	A1	A0	S2	S0
0	0	0	1	0	0
0	0	1	X	0	1
0	1	X	X	1	0
1	X	X	X	1	1

**X** duten sarreren balioei eraginik gabeko gai deritze. Hau da, sarrerak “1” balioa zein “0” balioa eduki, irteerak balio berbera eman beharko du. S0 eta S1 funtziok lortzeko, eraginik gabeko gaiak “0” eta “1” balioatzat hartu beharko dira.

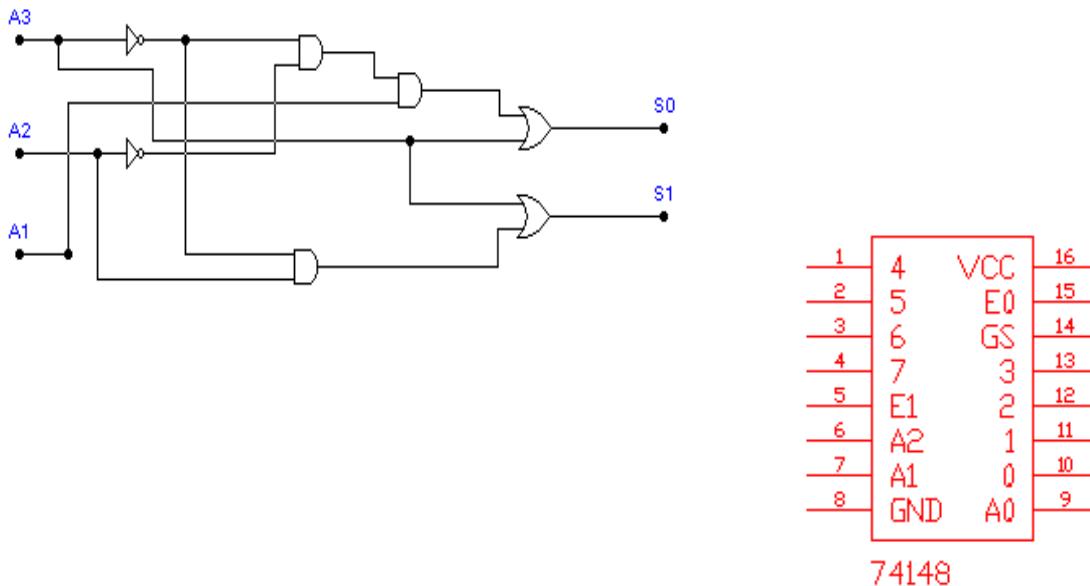
Eraginik gabeko gaiak irteeren zutabeetan agertzen badira, “0” zein “1” balioa har dezakete, komeni denaren arabera.

Funtziok lortu ondoren eta **Karnaughen** metodoaren bidez laburtuta:

$$S0 = \overline{A3} \times \overline{A2} \times A1 + A3$$

$$S1 = A2 \times \overline{A3} + A3$$

Eta horren logigrama ondorengo irudikoa bezalakoa izango da:



## 74LS148 kodetzailea

TTL teknologiaz eginiko zirkitu hau **zortzi sarbide eta hiru irtenbide** dituen kodetzailea da. Teklatu batetik agindutako datuetatik kode bitarra lortzeko erabiltzen da nagusiki.

Kodetzaile honetan **maila aktiboa “0” da**.

**El eragozpen-sarrerak**, gailuak datuak kode ditzan, **“0” balioan egon behar du**.

**EO irteerak** maila baxu baten bidez sarrera guztiak maila altuan daudela adierazten du.

**GS irteera** maila baxura igarotzen da datuen sarreretako bat aktiboa denean, hau da, zirkuitu kodetua dagoenean.

## 5.4. DESKODETZAILEAK

Kodetzaileek betetzen duten funtziaren aurkakoa betetzen dute. Deskodetzaileak, sarreran dagoen konbinazio bitarraren arabera, irteera bat aukeratzen du.

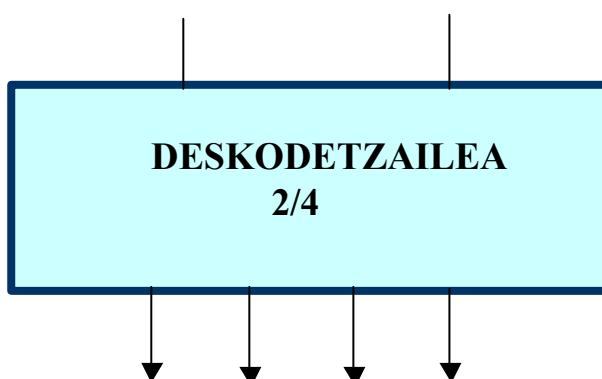
Ezaugarriak:

- \* **Sarrera-kopurua:** N
- \* **Irteera-kopurua:** M
- \* **Sarreretako egoera zehatz bakoitzean irteera bat besterik ez da egongo egoera logiko batean (1 edo 0).**

$$M = 2^N$$

5.7. irudian agertzen den zirkuitua **2/4 deskodetzailea** da.

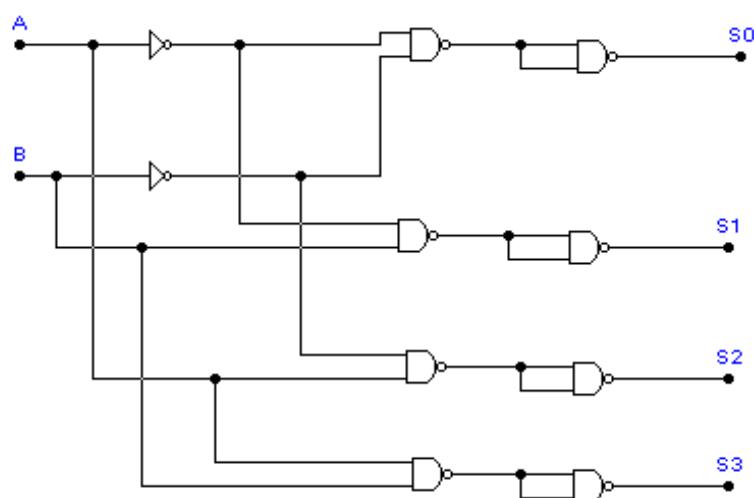
SARRERAK		IRTEERAK			
B	A	S0	S1	S2	S3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



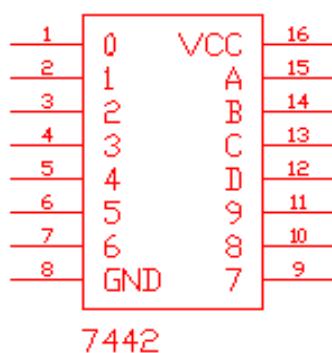
5.7. irudia. 2/4 deskodetzailea.

Egia-taulatik ondorengo ekuazioak lortuko ditugu:

$$\begin{aligned} S_0 &= \overline{B} \times \overline{A} ; \quad S_0 = \overline{\overline{B}} \times \overline{\overline{A}} \\ S_1 &= \overline{B} \times A ; \quad S_1 = \overline{\overline{B}} \times A \\ S_2 &= B \times \overline{A} ; \quad S_2 = \overline{B} \times \overline{A} \\ S_3 &= B \times A ; \quad S_3 = \overline{B} \times A \end{aligned}$$

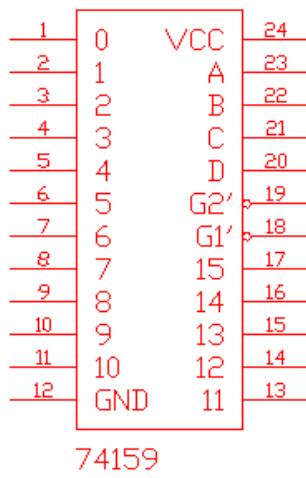


- 74LS42 deskodetzailea**
- \* Lau sarbide eta hamar irtenbide dauzka
    - Sarrerari BCD konbinazioa ezarrita aktibatzen du dagokion irtenbidea
    - Irteerako maila aktiboa “0” da
  - \* TTL teknologiaz eginiko zirkuitua da



### 5.4.1. DESKODETZAILE HAMASEITARRA

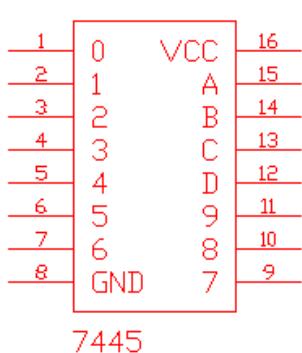
Deskodetzaile honetan lau sarrerari hamasei irteera dagozkie. Deskodetzaile hamartarrerako esaniko guztiak balio du kasu honetan. Gertatzen dena da beste bi sarrera agertzen zaizkigula eta sarrera horiek "0" egoeran egon behar dutela, bestela zirkuituak ez baitu funtzionatzen.



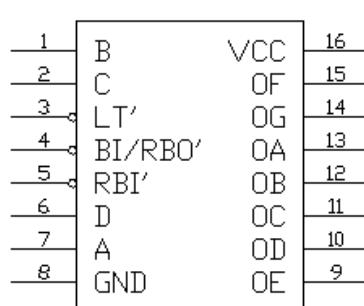
74159

### 5.4.2. DESKODETZAILE HAMARTARRA

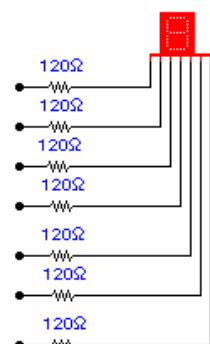
Irudian agertzen den bezala, 4 sarrera ditu eta hamar irteera. Horren ondorioz, ikusi bezain azkar nabaritzen zaion erabilera BCD kodetik hamartarrera pasatzea da, zeren eta BCD kodeko konbinazio bat ezarriz gero, irteeretako batean bakarrik agertzen baita "0" balio logikoa.



7445

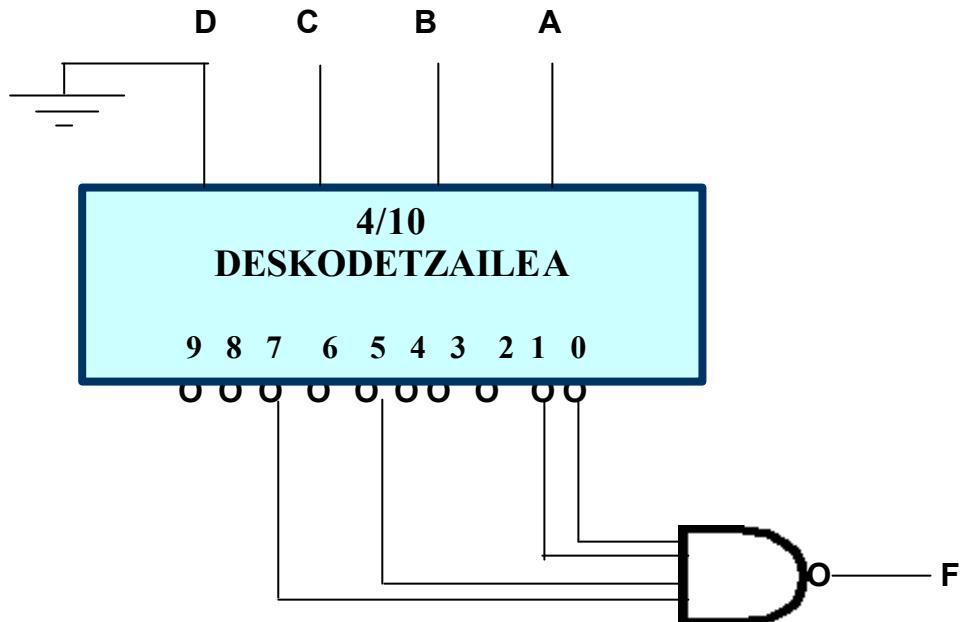


7448



### 5.4.3. FUNTZIO BAT GAUZATU DESKODETZAILEAK ERABILIZ

Deskodetzaile hamartar batek eta ate logikoen bidez, 3 aldagaiako *minterm* eran ipinitako funtziobat gauza daiteke.



5.8. irudia. Zirkuitua.

Adibide bezala, gauza dezagun ondoren agertzen den funtzioa:

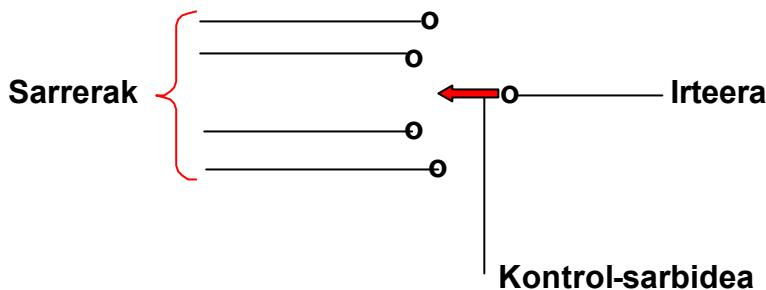
$$F = \bar{a}_3(0,1,5,7)$$

Adierazitako funtzioa gauzatzeko eman behar diren pausoak hauexek dira:

- Funtzioaren hiru aldagaiak (A, B, C) deskodetzailearen A0, A1 eta A2 sarrerei konektatu, eta A3-ri “0” bat ipini
- Deskodetzailearen 0, 1, 5 eta 7 irteerak NAND ate baten irteeretara eraman.
- Funtzioa NAND atearen irteera dugu.

### 5.5. MULTIPLEXADOREAK

Multiplexatzea, beharren arabera, hainbat sarreratan dagoen informazioen bat irteera bakar batetik bidaltzea da. Eginkizun hori betetzen duen gailurik simpleena **komutadorea** da.

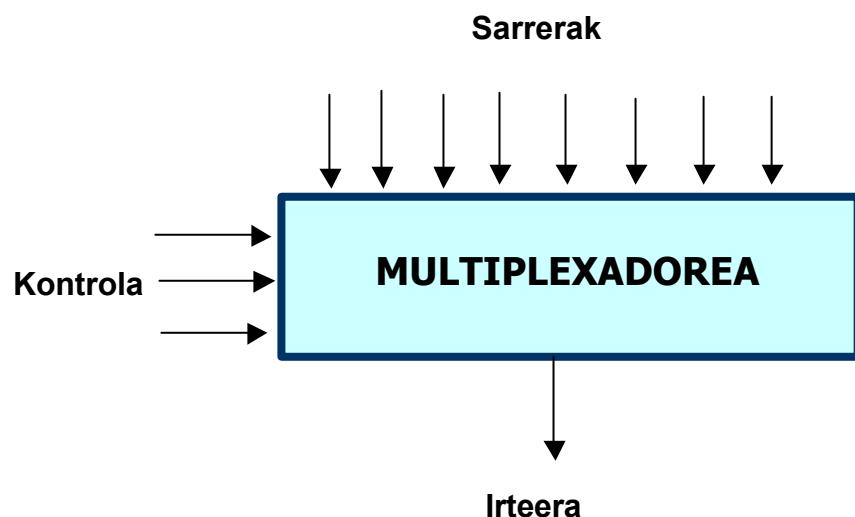


Eginkizun hori betetzen duen konbinaziozko zirkuituari **multiplexadore** esaten zaio, eta **N** informazio-sarreraz, irteera batez eta **n** kontrol-sarbidez osatuta dago. N informazio-sarreren eta kontrol-sarbideen arteko erlazioa hau da:

$$N = 2^n$$

Ondorengo irudian erakusten da sinbolikoki zortzi informazio-sarrera eta bi kontrol-sarbide dituen multiplexadorea.

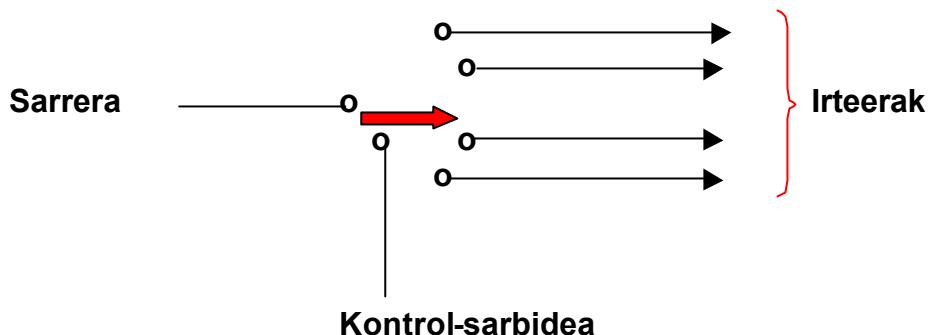
Kontrol-sarbideak			
a	b	c	S
0	0	0	D0
0	0	1	D1
0	1	0	D2
0	1	1	D3
1	0	0	D4
1	0	1	D5
1	1	0	D6
1	1	1	D7



## 5.6. DEMULTIPLEXADOREA

Demultiplexadoreek sarrera bat besterik ez dute, **N** irteera eta **n** kontrol-sarbide. Sarrerako informazioa kontrol-sarbideak adierazitako irteerara doa.

## Konbinaziozko zirkuituak

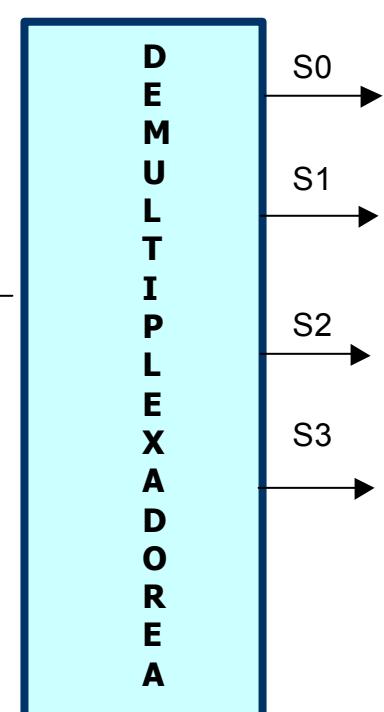


Ondorengo irudia lau irteerako demultiplexadoreari dagokio.

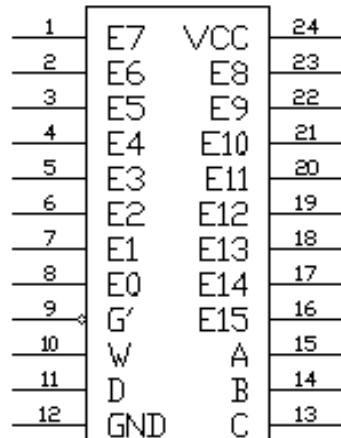
Kontrol-sarbideak		Irteerak			
a	b	S0	S1	S2	S3
0	0	D	0	0	0
0	1	0	D	0	0
1	0	0	0	D	0
1	1	0	0	0	D

Kontrol-sarbideak

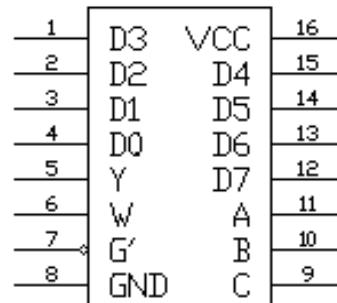
Sarrera (D)



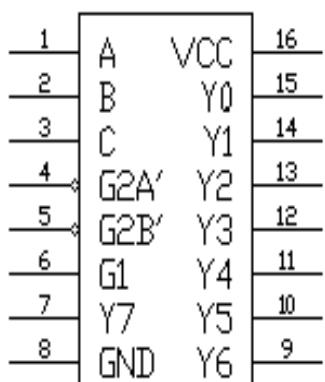
## 5.7. MULTIPLEXADOREEN ETA DEMULTIPLEXADOREEN ZIRKUITU INTEGRATUAK



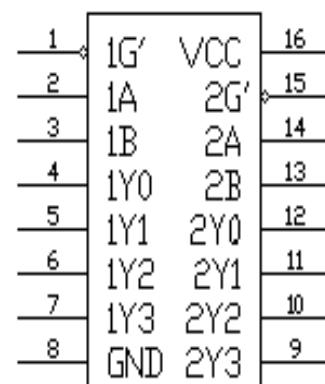
74150



74151



74138



74139

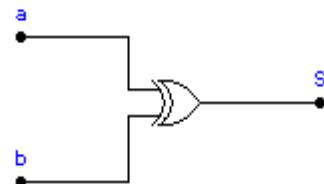
Goiko irudian agertzen diren integratuak TTL taldean daudenak dira.

## 5.8. KONPARADOREAK

Konparadoreak konbinaziozko zirkuituak dira. Horien sarreretan n biteko bi hitz agertzen direnean, bi hitz horiek berdinak diren ala ez (eta ez badira, zein den handiena edo txikiena) detektatzen dute. Hitza aurrez finkaturiko egitura batean pilaturiko multzoa da (4, 6, 8, 16, 32 bit). Kode bat eratzen da, eta sistema digitaletan harremana errazteko erabiltzen da.

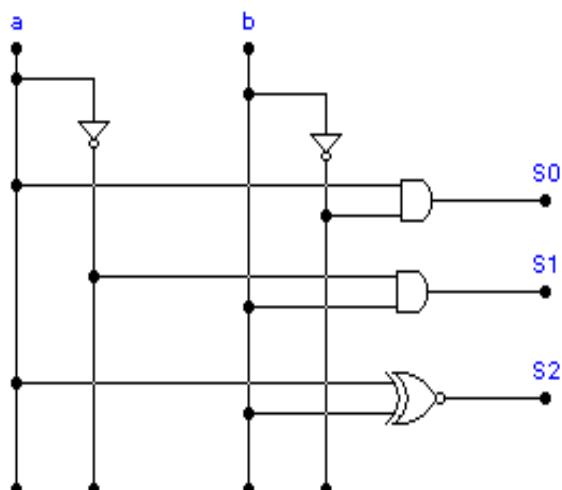
XOR ate baten bidez, oinarrizko zelula konparatzaila egin daiteke.

Sarrerak		Irteera
a	b	S
0	0	0
0	1	1
1	0	1
1	1	0



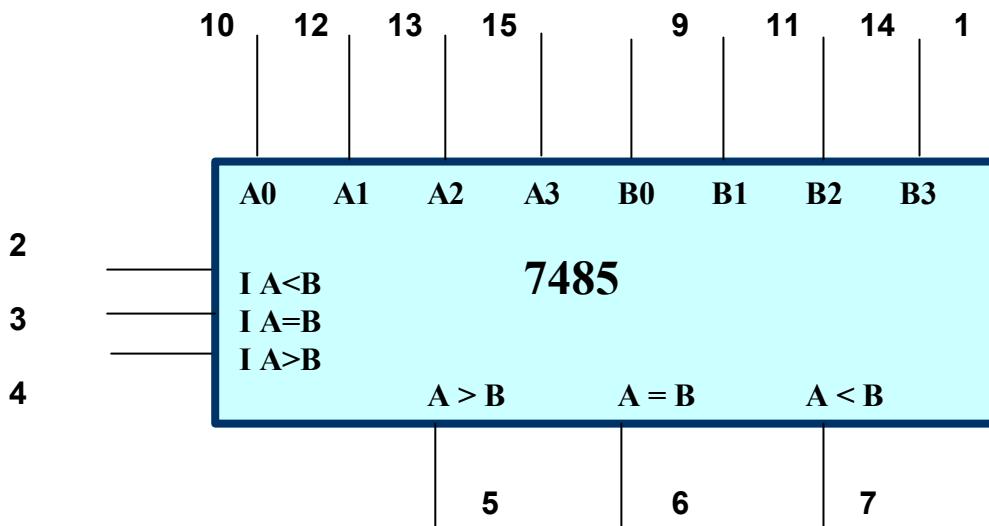
Konparadore baten diseinuaren prozesua lehenengo kasuetakoena modukoa da. Beheko irudian erakusten da bit banako bi hitzetako konparadorearen egia-taula. Diseinu hori egiteko egindako urratsek balio dute baita hitzeko bit-kopuru handiagoa duten konparadoreetarako ere.

Sarrerak		Irteerak		
a	b	S0	S1	S2
0	0	0	0	1
0	1	0	1	0
1	0	1	0	0
1	1	0	0	1



### 5.8.1. KONPARADOREEN ZIRKUITU INTEGRATUAK

#### **zirkuitu integratuaren ikurra**



Launa bit dauzkaten hitzez osaturiko konparadorea da. Gailu hori paraleloan konekta daiteke antzeko beste gailu batzuekin, bit gehiagoko hitzak konparatu nahi badira. Esate baterako, gailu baten **A > B, A = B eta A < B irteerak** beste gailu baten izen bereko sarrerei ezarriz gero, 8 biteko konparadorea lortuko dugu.

SARRERAK				SARRERAK			IRTEERAK		
A3 B3	A2 b2	A1 B1	A0 B0	A>B	A<B	A=B	A>B	A<B	A=B
A3>B3	X	X	X	X	X	X	H	L	L
A3<B3	X	X	X	X	X	X	L	H	L
A3=B3	A2>B2	X	X	X	X	X	H	L	L
A3=B3	A2<B2	X	X	X	X	X	L	H	L
A3=B3	A2=B2	A1>B1	X	X	X	X	H	L	L
A3=B3	A2=B2	A1<B1	X	X	X	X	L	H	L
A3=B3	A2=B2	A1=B1	A0>B0	X	X	X	H	L	L
A3=B3	A2=B2	A1=B1	A0<B0	X	X	X	L	H	L
A3=B3	A2=B2	A1=B1	A0=B0	H	L	L	H	L	L
A3=B3	A2=B2	A1=B1	A0=B0	L	H	L	L	H	L
A3=B3	A2=B2	A1=B1	A0=B0	L	L	H	L	L	H
A3=B3 A2=B2 A1=B1 A0=B0				X	X	H	L	L	H
A3=B3 A2=B2 A1=B1 A0=B0				H	H	L	L	L	L
A3=B3 A2=B2 A1=B1 A0=B0				L	L	L	H	H	L

## 5.9. ZIRKUITU ERAGILEAK

Zirkuitu digitalekin eragiketa matematikoak ere egin daitezke. Horretarako zirkuituak ere, aurreko gaikoak bezala, konbinaziozko zirkuituak dira, baina beren berezitasunak direla eta, gai oso bat aparte eskaintza pentsatu dugu.

Zirkuitu digitalen bidez eragiketa matematikoak egiteko, lehenik datuak eraldatu egin behar dira kodeturiko adierazpenetan, eta gero egin eragiketa aljebra bitarraren algoritmoen bidez.

Biderketaren eta zatiketaren eragiketak hainbat batuketa eta kenketen bidez egiten dira, aurrerago zirkuitu programagarriean ikusiko dugun bezala.

### 5.9.1. BATUKETA BITARRA

Bi biteko batuketa bitarra oso erraza da, eragiketaren emaitzak, hau da, baturak, bi balio soilik (“1” eta “0”) izan ditzake eta.

Bi bitekin eragin ondorengo **bururakoa** (ingelesez *carry*) kontuan hartu beharko du batutzaileak.

5.9. irudian bildu ditugu biten artean egin daitezkeen batuketa guztiak.

A	B	BATUKETA	BURURAKOA
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

5.9. irudia

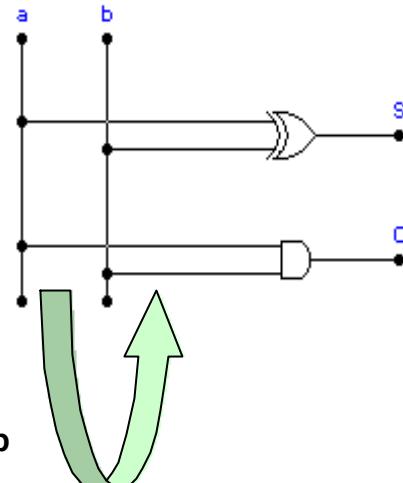
4 eta 5 zenbakien batuketa:

$$\begin{array}{r}
 & 4 \\
 + & 5 \\
 \hline
 9
 \end{array}
 \qquad
 \begin{array}{r}
 0100 \\
 +
 0101 \\
 \hline
 1001
 \end{array}$$

#### 5.9.1.1. Batutzaileak

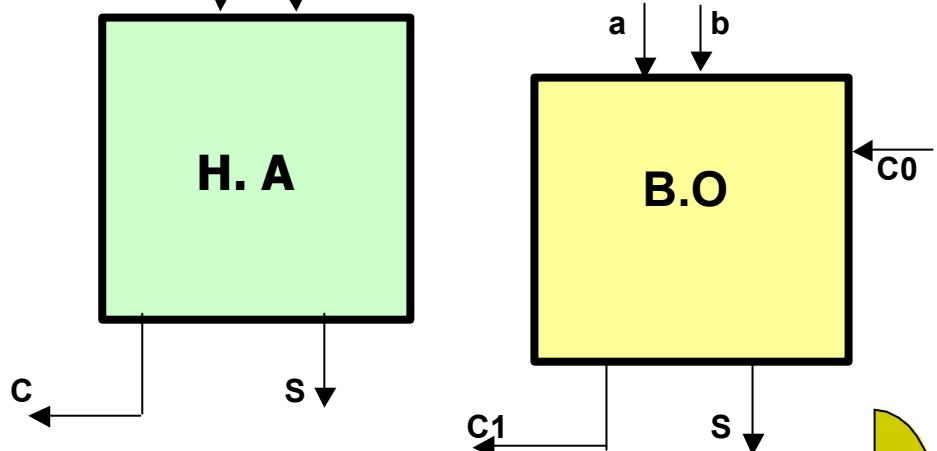
**Erdibatutzailea.** Aurreko bururakoa (*carry*) kontuan hartzen ez duen zirkuitua da.

a	b	S	C
0	0	0	0
1	0º	1	0
0	1	1	0
1	1	0	1



$$S = a \cdot \bar{b} + \bar{a} \cdot b$$

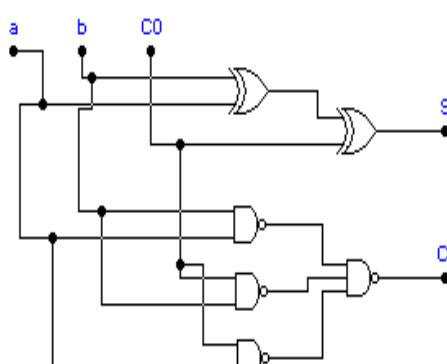
$$C = a \cdot b$$



**Batutzaile osoa.** Aurreko bururakoa (carry) kontuan hartzen duen zirkuitua da.

a	b	C0	S	C1
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

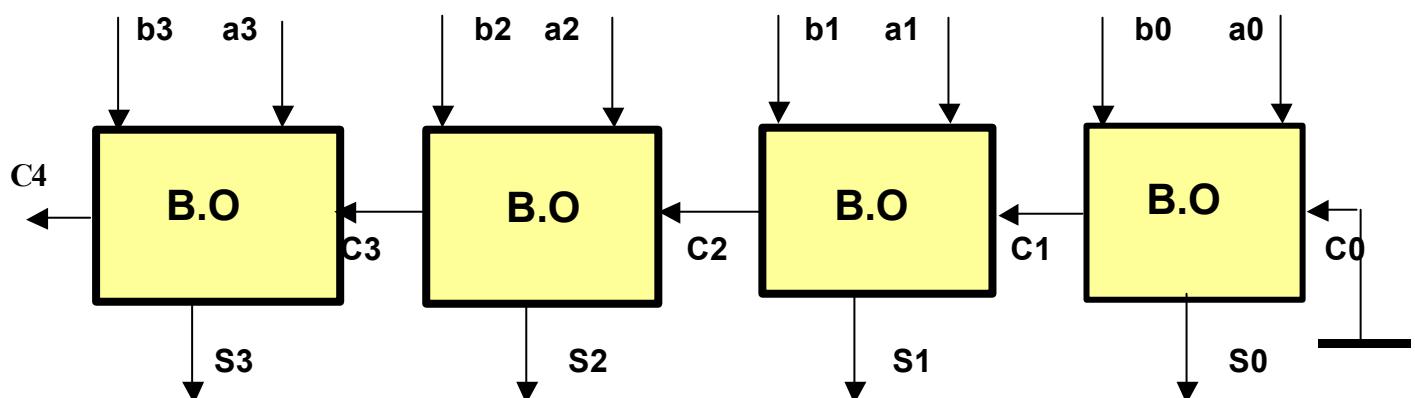
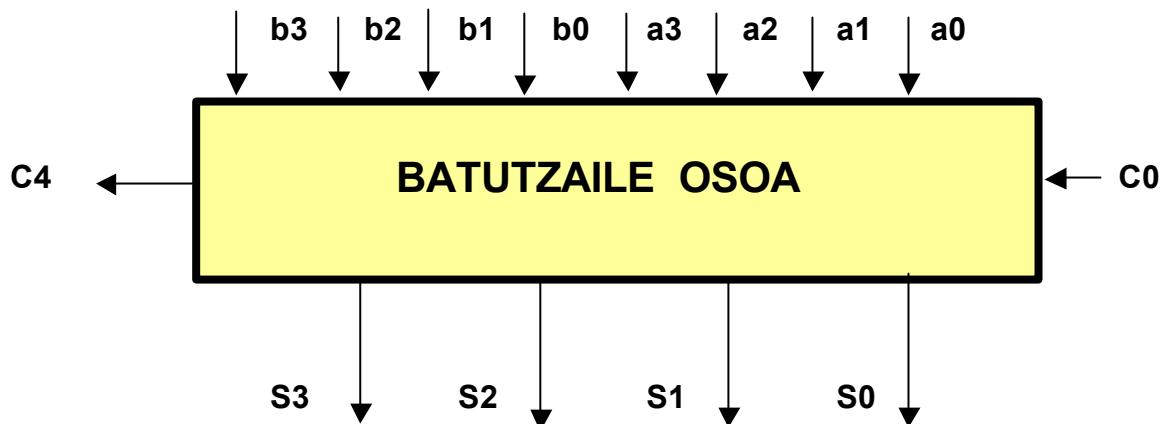
$$S = C0 + (a + b)$$



$$C = \overline{b} \overline{C0} \times a \overline{C0} \times a b$$

**Batutzaile oso laukoitza.** Bit batuetako bi zenbaki batu nahi izanez gero, zirkuitu konplexuagoa behar izaten da.

Nahiz eta eremu zabala izan (bit-kopuruarekiko), gehien erabiltzen dena eta oinarrizkotzat hartzen dena lau bitekoa da.



TTL taldean funtzio hori egin ohi duen zirkuitu integratua hau da:

**4 – BIT FULL ADDER 7483**

CMOS taldean funtzio hori egiten duen zirkuitu integratua honako hau da:

**4 – BIT BINARY FULL ADDER HEF 4008B**

### 5.9.2. KENKETA BITARRA

Kenketa zirkuitu digitalen bidez egin daitekeen arren, oso konplexua litzateke zirkuitu bat batuketentzat eta beste bat kenketentzat erabiltzea. Hori dela eta, kenketa batuketa bihurtzeko trikimailu bat erabiltzen da.

Demagun ondorengo kenketa daukagula:

$$\begin{array}{r} 14 \quad (\text{kenkizuna}) \\ - \\ 8 \quad (\text{kentzailea}) \\ \hline 6 \quad (\text{kendura}) \end{array}$$

Adierazitako kenketa beste era batera jar daiteke:

$$\begin{array}{r} 14 \quad (\text{kenkizuna}) \\ + \\ (-) 8 \quad (\text{kentzailea}) \\ \hline 6 \quad (\text{kendura}) \end{array}$$

Ikus daitekeenez, **kentzaileari zeinua** aldatu besterik ez zaio egin behar kenketa batuketa bihurtzeko.

Hiru era ezberdin dago zenbaki negatibo bat adierazteko. Hauexek dira:

- **Zeinu-bitia**
- **Baterako osagarria**
- **Birako osagarria**

### Zeinu-bitak.

Kasu hauetan, zenbakia positiboa bada, aurretik “0” jarriko diogu. Negatiboei, aldiz, “1” jarriko diegu.

Adibidez:

Zenbakia	ZB
13	0 1101
- 7	1 0111

### Baterako osagarria erabiliz.

Zenbaki baten baterako osagarria lortzeko, zenbaki horren bitak alderanzten dira ( $1 \rightarrow 0$ ) eta ( $0 \rightarrow 1$ ).

Adibidez:

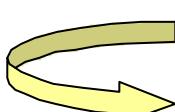
Zenbakia	Baterako osagarria
(8) 1000	(-8) 0111
(13) 1101	(-13) 0010

### Birako osagarria erabiliz.

Zenbaki baten birako osagarria lortzeko, zenbaki horren bitak alderanzten dira eta gero bat gehitu. Eskuinetik hasten da kopiatzen lehenengo batekoa aurkitu arte. Handik aurrera alderanztu egiten dira.

Adibidez:

Zenbakia	Birako osagarria
(14) 1110	(-14) 0001 (baterako osagarria) + 1 ----- 0010



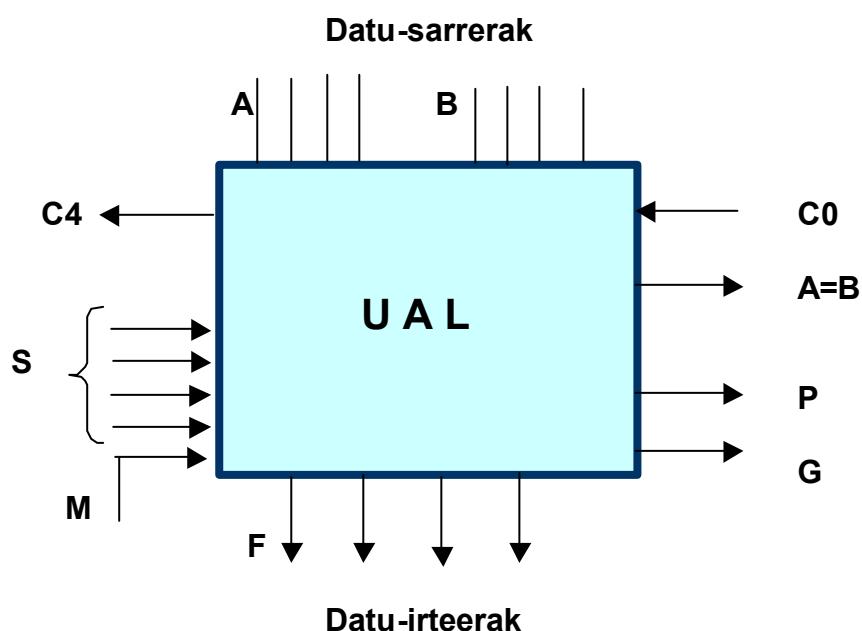
## 5.10. UNITATE ARITMETIKO ETA LOGIKOA (UAL)

Askotan beharrezkoa da eragiketa aritmetikoak eta logikoak egin ditzakeen zirkuitua erabiltzea. Beharra ikusirik, ekoizleek MSI eskala ertaineko integracioko zirkuitua gauzatu zuten, eta UAL izena jarri (Unitate Aritmetiko Logikoa).

UAlez egin daitezkeen eragiketa garrantzitsuenetariko batzuk ondorengoak izan litezke:

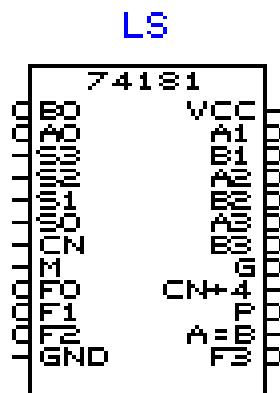
Aritmetikoak			Logikoak			
A	gehi	B		A	+	B
A	gehi	B	gehi 1	A	.	B
A	gehi	1		A	+	B
A	ken	B		A	.	B
A	ken	B	ken 1	A	0	B
Aren birako osagarria				A	0	B
A	ken	1		A	+	B

UALEN ikur logikoa hau da:



**S = Hautapen-sarreren bidez, lau biteko bi biten arteko hainbat eragiketa egin ohi ditu.**

Adierazitako funtzioa egiten duen zirkitu integratua, TTL taldean, hau da:



**Datu-sarrerak:** **A taldea** (A3A2A1A0) **B taldea** (B3B2B1B0)

**Datu-irteerak:** **F taldea** (F3F2F1F0)

**Hautapen-sarrerak:**  $\begin{cases} M = 0 & \text{eragiketa ARITMETIKOAK} \\ M = 1 & \text{eragiketa LOGIKOAK} \end{cases}$

**Hautapen-sarrerak:** **S taldea** (S3S2S1S0)

**G eta P irteerak:** bururako sorgailuari konektatzeko

**C 4 =** hurrengo bururakoa

**C0 =** aurreko bururakoa

Zirkitu integratu honen bidez, lau biteko zenbakien arteko eragiketak egin ditzakegu. Baina noizbehinka, lau bit baino gehiagoko zenbakiekin eragiketak egiteko beharra izan dezakegu. Orduan, beharraren arabera, unitate batzuk konektatu beharko ditugu.

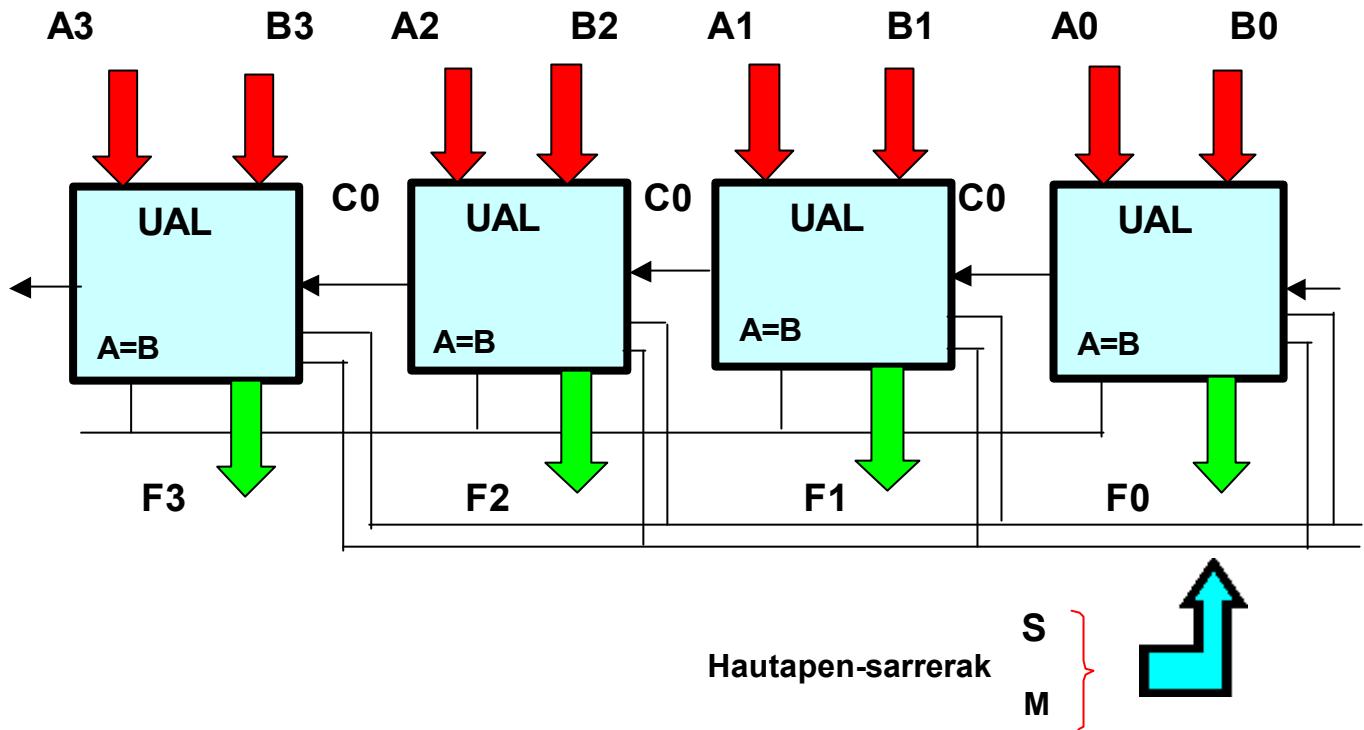
UALen arteko loturak bi motatakoak dira:

- **Serieko hedapena**
- **Paraleloko hedapena**

### a) Serieko hedapena:

Unitateak ematen duen bururakoa (**C4**) bakarrik erabiliko da. Serieko hedapena paraleloko baino askoz errazago gauzatzen da.

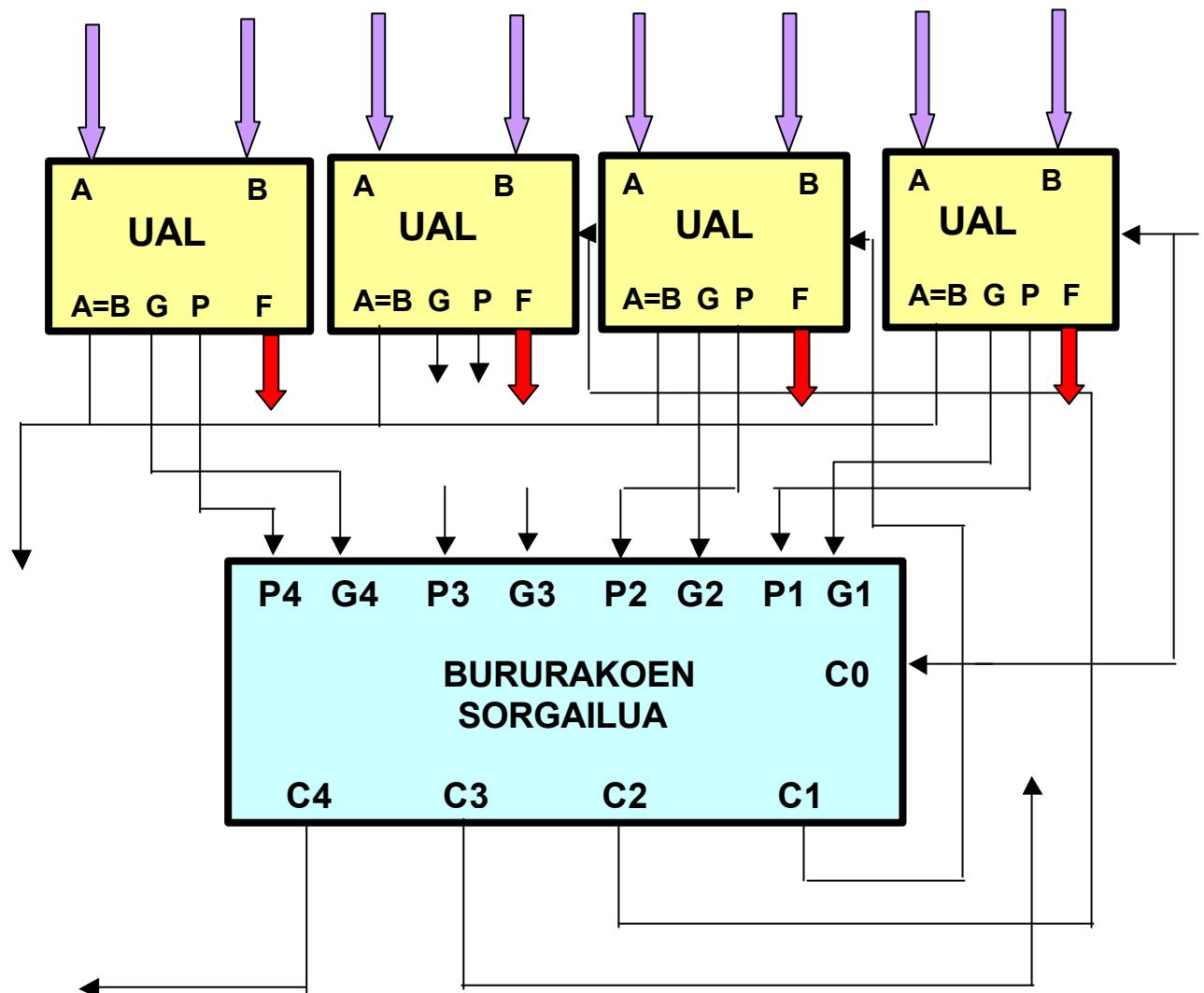
**(16 biteko UALa).**



### b) Paraleloko hedapena:

Kasu honetan, **bururako sorgailua** erabili beharko dugu. Paraleloko hedapena azkarragoa da seriekoa baino.

Ondorengo irudian, 16 biteko eragiketak egin ahal izateko, bururako sorgailua nola konektatzen den ikus daiteke.



(16 biteko UALa - Bururakoaren paraleloko hedapena)

## Ariketak

1. Osa itzazue hurrengo taulak (zenbakiak falta diren zenbaki-sisteman ipini):

HAMASEITARRA	HAMARTARRA	ZORTZITARRA	BITARRA
7DB			
	421		
		1634	
			1000111001

HAMASEITARRA	HAMARTARRA	ZORTZITARRA	BITARRA
A2B			
	384,25		
		7246	
			11101010011

**2. Adieraz itzazue sistema bitarrean eta zeinu-bitez ondorengo zenbaki hamartarrak:**

- a) + 29                  b) - 85                  c) + 100                  d) - 123

**3. Bateko osagarrian adierazitako hurrengo zenbaki bitarren balio hamartarra aurkitu:**

- a) 10011001                  b) 01110100                  c) 10111111                  d) 11001100

**4. Egin hurrengo batuketak sistema bitarrean, zeinu-bitak erabiliz:**

- a) 33 + 15                  b) 56 + (-27)                  c) -46 + 25                  d) -110 + (-84)

**5. Egin hurrengo batuketak; emaitzak sistema hamartarrean egiaztatu:**

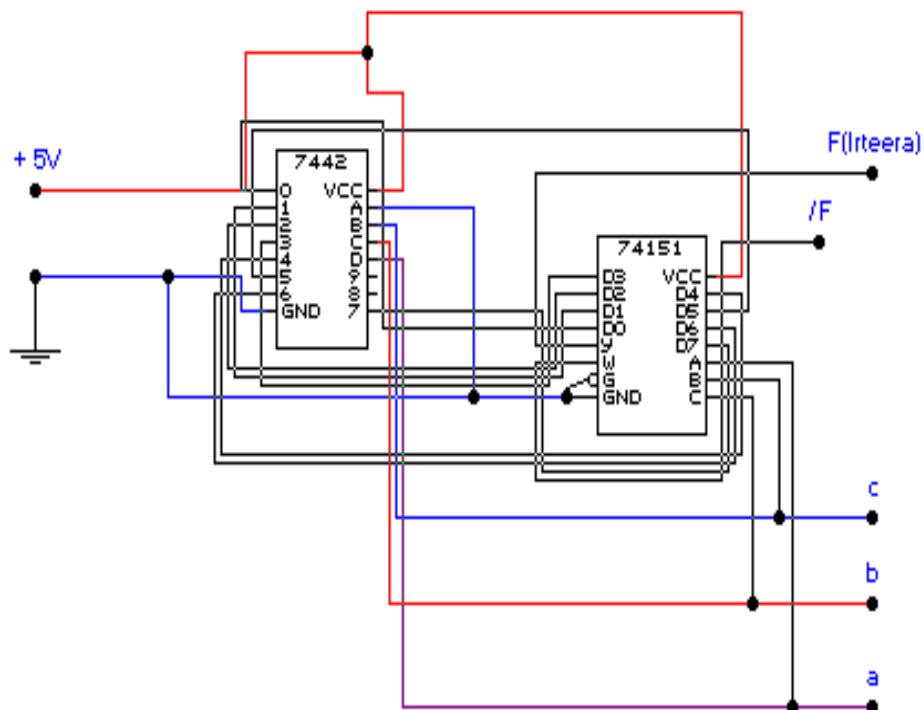
- a) 100010100 + 00111001                  b) 11011001 + 11100111

**6. Implementa ezazue hurrengo funtzioa 74LS42 deskodetzaile komertziala erabiliz:**

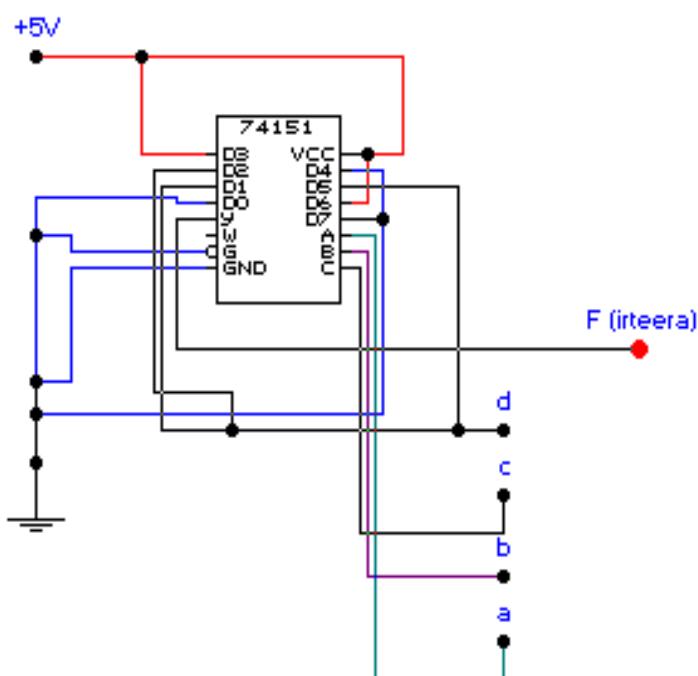
$$F = \bar{a} \times \bar{b} + \bar{a} \times d + \bar{b} \times \bar{c} \times d$$

**7. Diseina ezazue bitik lau lerrorako deskodetzailea, ate logikoak erabiliz, sarrera bitar naturalean eta maila aktiboa altua izanik.**

8. Irudiko eskeman (BCD deskodetzailea 7442 motakoa, eta zortzi kanaleko 74151 multiplexadorea) agertzen den zirkuituaren egia-taula lor ezazue, "a" MSB eta "c" LSB direla jakinik.



9. Irudiko eskeman agertzen den integratua zein da? Lor ezazue implementatzaren duen funtzioaren egia-taula.



# 6. KAPITULUA

## Zirkuitu sekuentzialak

### 6.1. ZIRKUITU SEKUENTZIALEN DEFINIZIOA, EZAUGARRIAK ETA OSAKETA

Konbinaziozko zirkuituak bezala, zirkuitu sekuentzialak ere ate logikoz osatuta daude, eta, ate logikoekin gertatzen den moduan, katalogoetan eskuragarri dauden zirkuitu sekuentzial gehienen integrazio-eskala MSI da. Hala ere, oso ezaugarri bereziak dauzkate. Hona hemen bereizgarriak:

Konbinaziozko zirkuituetan ez bezala, sekuentzialetan, une jakin bateko irteerako balioak ez daude une horretan sarreran ezarritako balioen menpe bakarrik, aurretik agertu diren balioen menpe ere badaude eta.

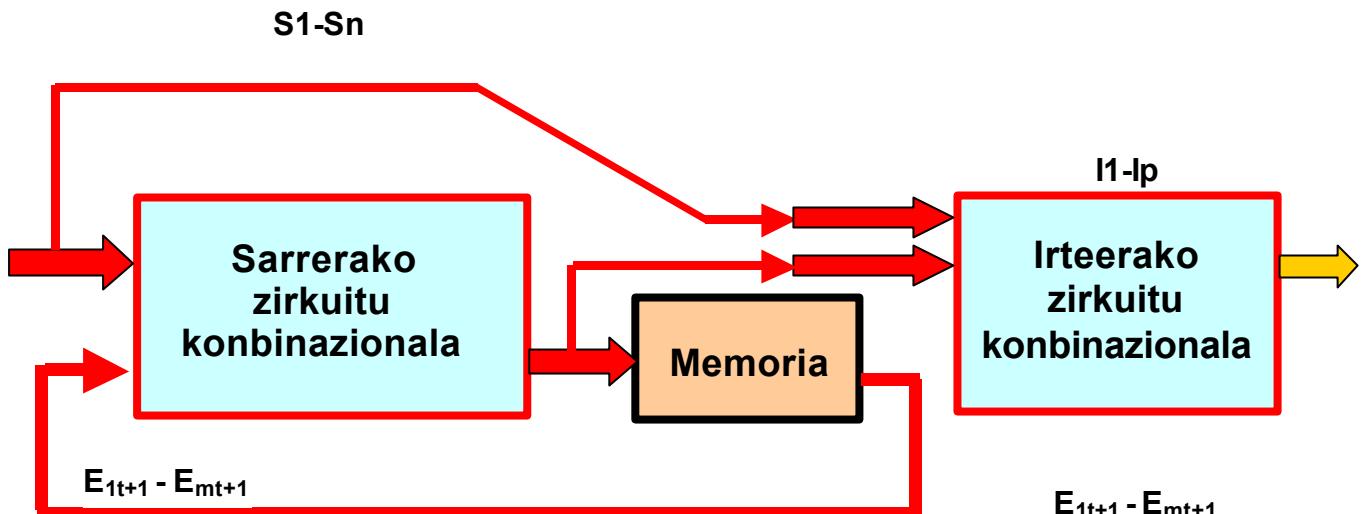
Beraz, gerta daiteke, unearen arabera, sarreretan balio berberak sartzea eta irteeran egoera ezberdinak lortzea.

Nolakoa den horrelako zirkuitu baten osaketa eta eraketa, halakoa izango da erantzuna sarreretan zirkuitu sekuentzial bat ezarriz gero.

Zirkuitu sekuentzialek sarreretako aldagaien balioak “gogoratzeko” edo metatzeko gaitasuna daukate. Eginkizun hori behar-beharrezkoa da zirkuitu digitalekin osaturiko sistema automatikoetan, batez ere programagarriak direnetan.

Zirkuituaren atean gertatzen den informazio-metatze edo -gogoratze hori barne-egoerakoak deritzen aldagai batzuk edukitzeari esker egiten da. Aldagai horien balioa sarreran ezarritako konbinazio bitarrean egindako aldaketen arabera aldatuko da.

Zirkuitu sekuentzialik osoena sarreran konbinaziozko zirkuitu batez, irteeran beste batez eta memoria-unitate multzo batez (barne-egoeraren aldagai bakoitzeko bana) osatuta dagoena da. Halako xe eraketa bat erakusten da, era sinbolikoan, 6.1. irudian. Egitura horri **Mealy**-ren automata deitzen zaio.



6.1. irudia. Zirkuitu sekuentziala – Mealyren automata.

6.1 irudiko bloke-diagraman ikus daitekeenez, irteerako zirkuitu konbinazionalak  $I_{1-lp}$  p irteera-aldagai ematen ditu, sarrerako  $S1-Sn$  n aldagaien eta barne-egoera adierazten duten  $E_{1t}-E_{mt}$  m aldagaien konbinazioen arabera. Barne-egoera islatzen duten  $E_{1t}-E_{mt}$  aldagaiak, halaber, sarrerako zirkuitu konbinazionalak sortzen ditu, kontuan hartuz, batetik,  $S1-Sn$  sarrera-aldagaiak, eta bestetik, Memoria deritzan unitatearen bidez aurreko egoeraren berri ematen duten  $E_{1t+1}-E_{mt+1}$  aldagaiak.

Zirkuitu sekuentzialak sinkronoak eta asinkronoak izan daitezke. Lehenek kontroleko seinalea behar dute, zirkuitutik kanpo dagoen sorgailu batek emana, zeinak giltza balitzan baitihardu. Kontroleko seinaleari erlojukoa (*clock* ingelesez) ere deitzen zaio.

Sistema sekuentzial asinkronoek, beriz, ez daukate erloju-sarrerarik, eta barne-egoeraren aldagaietako eta irteerako balioetako aldaketak, besterik gabe, zirkuituko sarreretako balioak aldatzean gauzatzen dira.

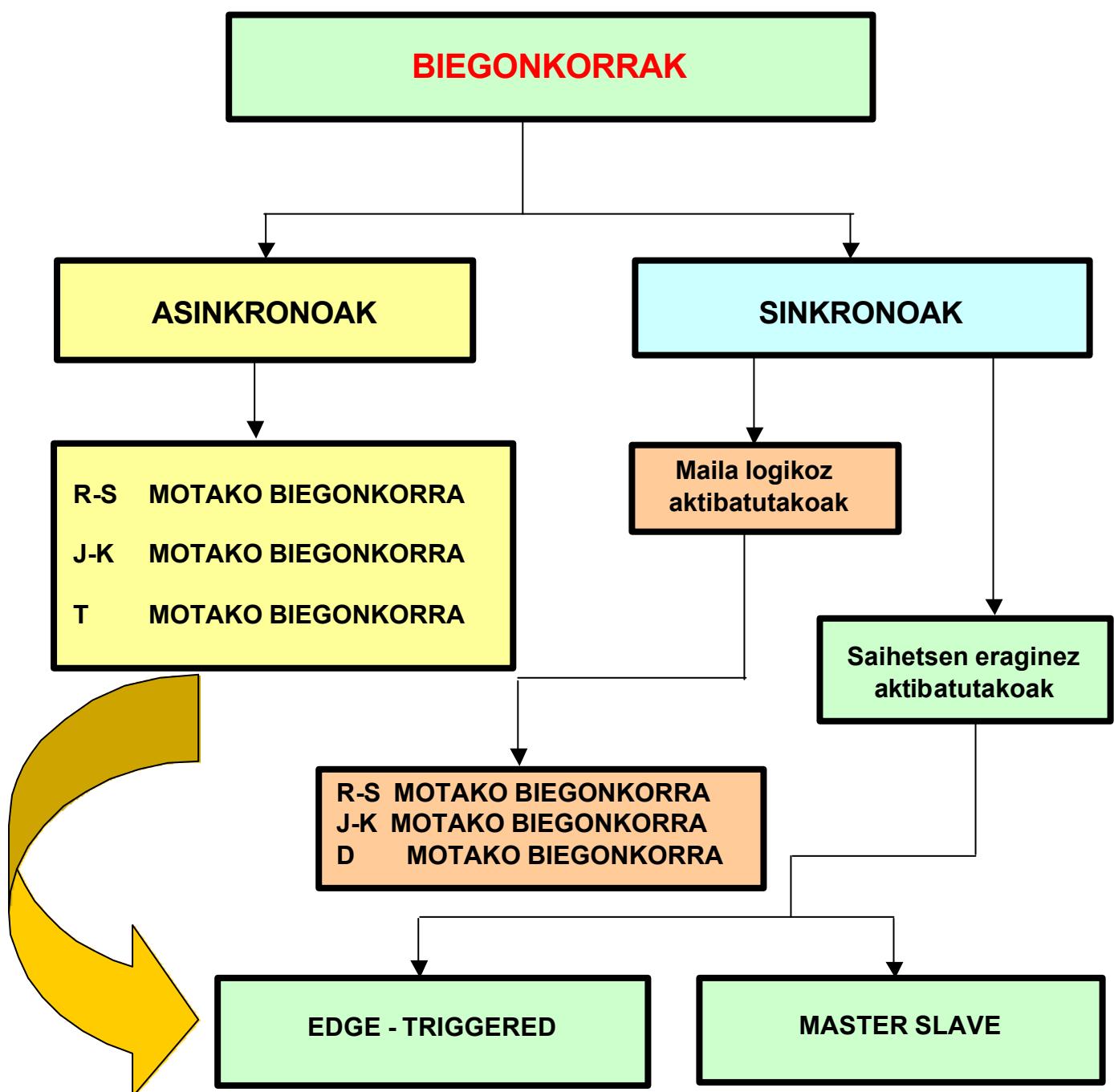
Beraz, gai honetan jorratuko ditugun zirkuitu sekuentzialak hauek izango dira:

- **Biegonkorra**k
- **Kontagailu**ak
- **Desplazamendu-erregistro**ak
- **Maiztasun-zatitzaleak**

### 6.1.1. BIEGONKORRAK

Elementu hauek bit bakarreko informazioa metatzeko ate logikoz osaturiko zirkuitu sekuentzialak dira. Sinkronoak eta asinkronoak izan daitezke, erloju-seinalearen beharraren arabera. Bestalde, haien aktibazioa maila logikoz edo erlojuaren saihets positiboz edo negatiboz egiten da.

Biegonkorren sailkapena honakoa da:



Sistema biegonkor asinkrono guztiak barne-egoerako aldagai bakarra dute, eta horrek memorizatu egiten du zein sarrera egon diren une jakin batean.  $t$  unean sistemak emango duen irteera barne-egoerako aldagaiak aurreko  $t-1$  unean hartu duen balioaren menpe dago. Hau da,  $t$  uneko irteera-balioak,  $t$  uneko sarrera-balioez gain, aurreko  $t-1$  unean egon diren sarreren menpe ere badaude.

#### 6.1.1.1. R-S Asinkronoa

Biegonkor honek bi sarrera dauzka, R (reset) eta S (set), eta bi irteera elkarren osagarriak, Q<sub>1</sub> eta Q<sub>2</sub>.

Ondoren azaltzen den egia-taulak adierazten digu biegonkor honen funtzionamendua.

$\phi_{t-1} = t-1$  uneko R-S sarrerek barne-egoerako aldagaiari eragiten dioten balioa.

$\phi_t = t$  uneko R-S sarrerek eta aurreko  $\phi_{t-1}$  balioek barne-egoerako aldagaiari eragiten dioten balio berria.

$Q_1 = R-S$  sarreren zein  $\phi_t$ -ren balioaren menpe dagoen  $t$  uneko irteeraren balioa.

Biegonkor honen kasuan  $Q_1 = \phi_t$  gertatzen da.

R	S	$j_{t-1}$	$j_t$	$Q_1$
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	1
1	0	0	0	0
1	0	1	0	0
1	1	X	X	X

Biegonkor hauetako erabiltzean, R eta S sarreren balioak aldi berean 1 inoiz ez izatea eskatzen da, eta horregatik ez da emaitzarik ematen kasu horretan.

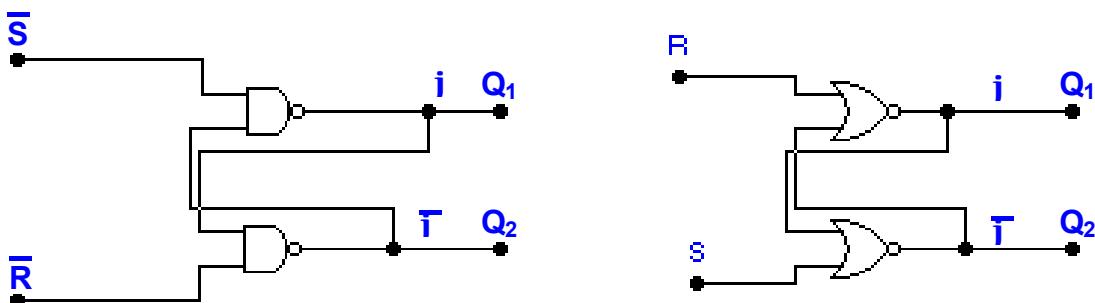
Egia-taulan ikus daitekeen bezala:

- **S sarrerari “1” logikoa aplikatuz,  $\phi_t = Q_1 = 1$ , aurreko balioan edozein izanda ere.**

- R sarrerarekin gauza bera eginez gero,  $Q_t = Q_1$ .
- R eta S “0” direnean, barne-egoeraren aldagaia eta irteerek aurreko balioa mantentzen dute; beraz,  $Q_{t-1} = Q_t = Q_1$
- Sarrera biak “1” balioa dutenean, irteerak indeterminatuak izaten dira.

Sarrera biek “1” balioa hartuko balute, **NOR ateez** eraikitakoan, irteerak ‘0’ balioko luke, eta **NAND ateez** egindakoan, “1” balioa.

Lehendabizikoari **ezabatzeko lehentasunduna** deitzen zaio.  
Bigarrengoa **inskribatzeko lehentasunduna** deitzen zaio.



### 6.1.1.2. J-K Asinkronoa

Biegonkor honek bi sarrera ditu, **J (set)** eta **K (reset)** eta bi irteera elkarren osagarriak,  **$Q_1$  eta  $Q_2$** .

Aurreko R-S biegonkorrean bezala, hemen ere sistema  $Q_1 = \phi_t$  gertatzeko moduan eraikitzen denez, egia-taula simplifikatu egingo dugu, barne-egoerako eta irteerako aldagaia errepikarazi gabe, berberak baitira. Hortaz:

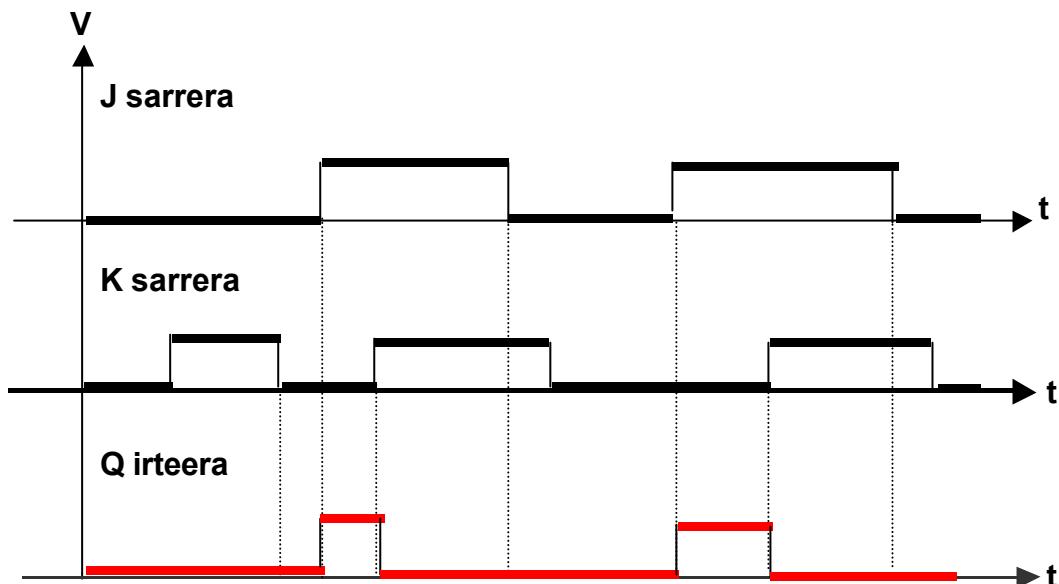
$Q_{t-1} = t-1$  uneko K-K sarrerek eragiten duten  $t-1$  uneko irteera-balioa.  
 $Q_t = t$  uneko JK sarrerek eta  $Q_{t-1}$  irteera-balioak  $t$  unean eragiten duten irteera-balioa.

J	K	$Q_{t-1}$	$Q_t$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0



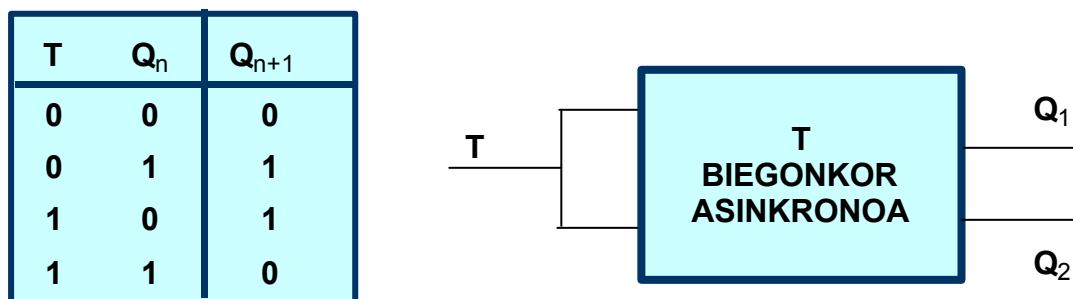
Egia-taulan ikus daitekeen bezala:

- **J = 0 eta K = 0** direnean, egoera ez da aldatzen.
- **J = 0 eta K = 1** direnean, irteera “0” izango da
- **J = 1 eta K = 0** direnean, irteera “1” izango da
- **J = 1 eta K = 1** direnean, irteerako egoera aldatzen da. Lehen “0” bazeen, orain “1”, eta alderantziz.



#### 6.1.1.3. T Asinkronoa

Biegonkor honek sarrera bat besterik ez dauka, **T (TOGGLE)** hain zuzen ere, eta bi irteera elkarren osagarriak, **Q<sub>1</sub>** eta **Q<sub>2</sub>**.



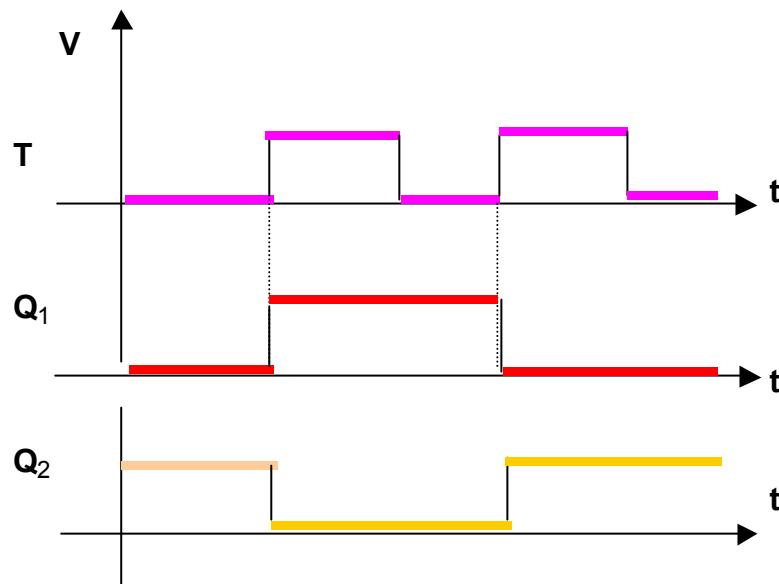
Egia-taula ikus daitekeen bezala:

- **T = 0**      denean, biegonkorren egoera ez da aldatzen

“0” bazein, orain “1”

izango da, eta alderantziz.

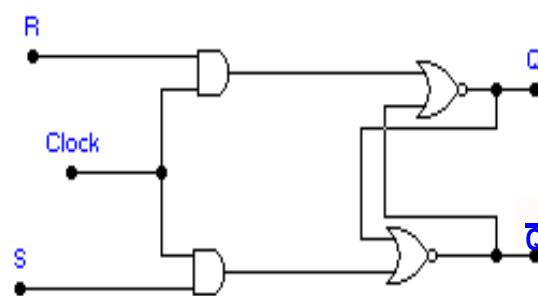
**Denbora-diagrama aztertuz gero:**

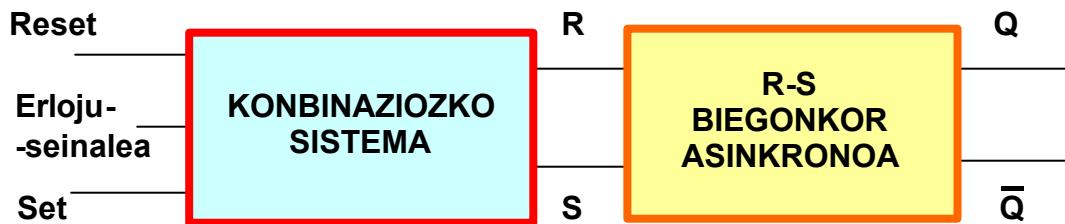
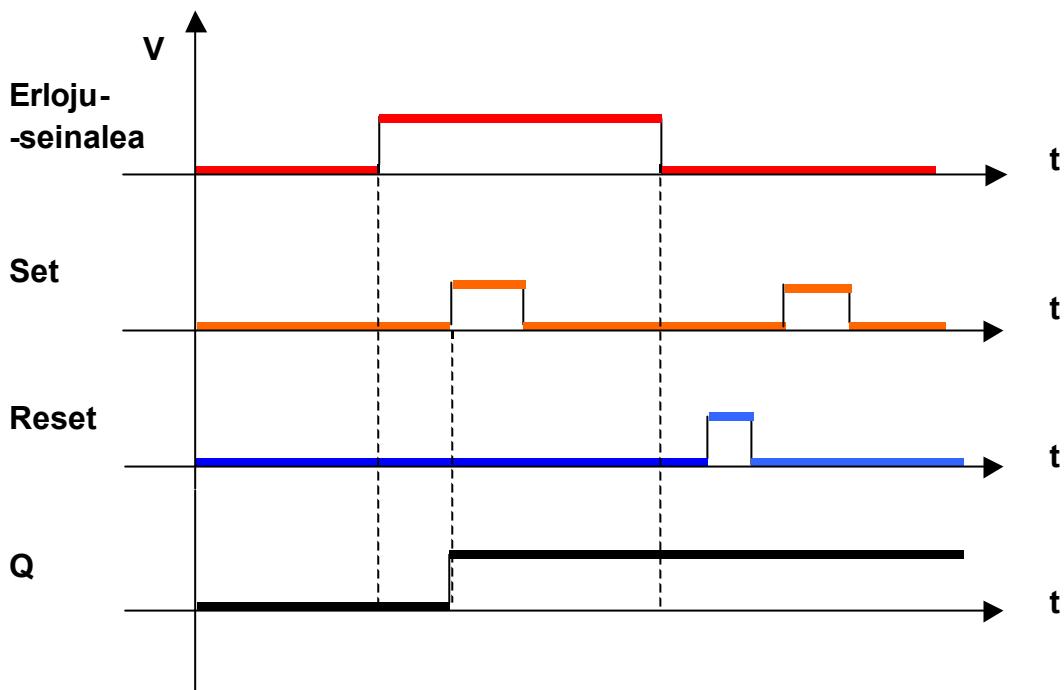


Denbora-diagraman ikusten dugunez, irteeran lortutako **seinalearen maiatz-suna sarrerakoaren erdia** da.

#### 6.1.1.4. Mailaz aktibaturiko R-S biegonkor sinkronoa

Mailaz aktibaturiko biegonkorretan, sarrerako seinaleak irteeran eragin ahal izan dezan, erloju-seinaleak (*clock*) maila altuan egon behar du. Denbora-diagraman ikusten denez, erloju-seinalea maila baxuan badago, sarreretako balioek (R eta S) ez dute eraginik irteeran. Kasu honetan, erloju-seinalea informazioa igarotzea kontrolatzen duen giltza da.



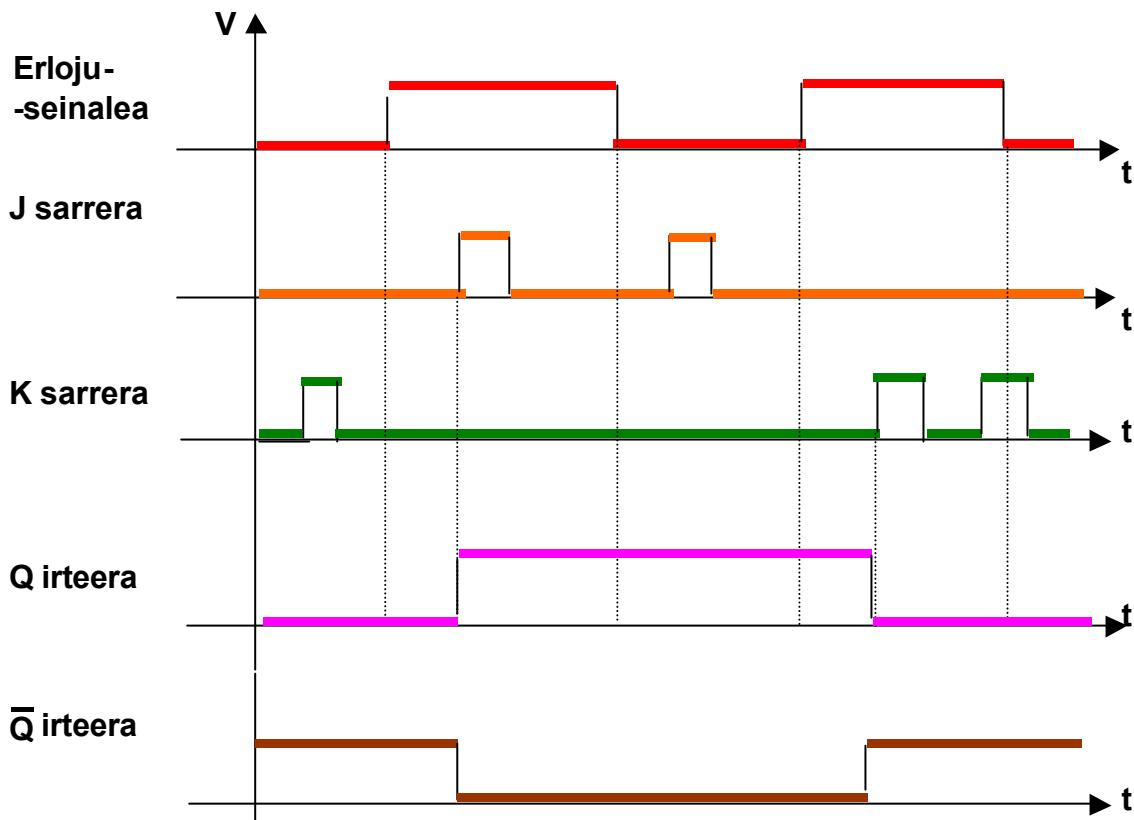


#### 6.1.1.5. Mailaz aktibaturiko J-K biegonkor sinkronoa

Irudian ikus daitekeenez, J-K biegonkor sinkronoak, R-S motakoak bezala, sarreran AND ate logiko pare bat jarriz lortzen dira.



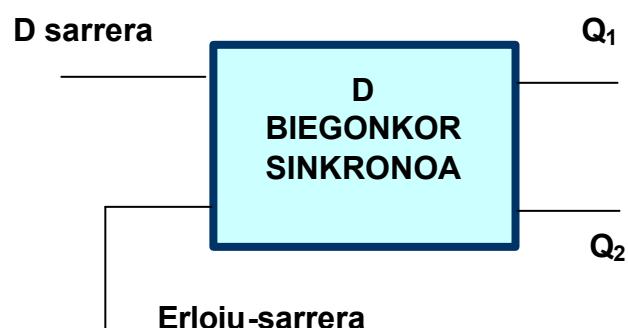
Denbora-diagrama aztertuz gero, horrela izango da:

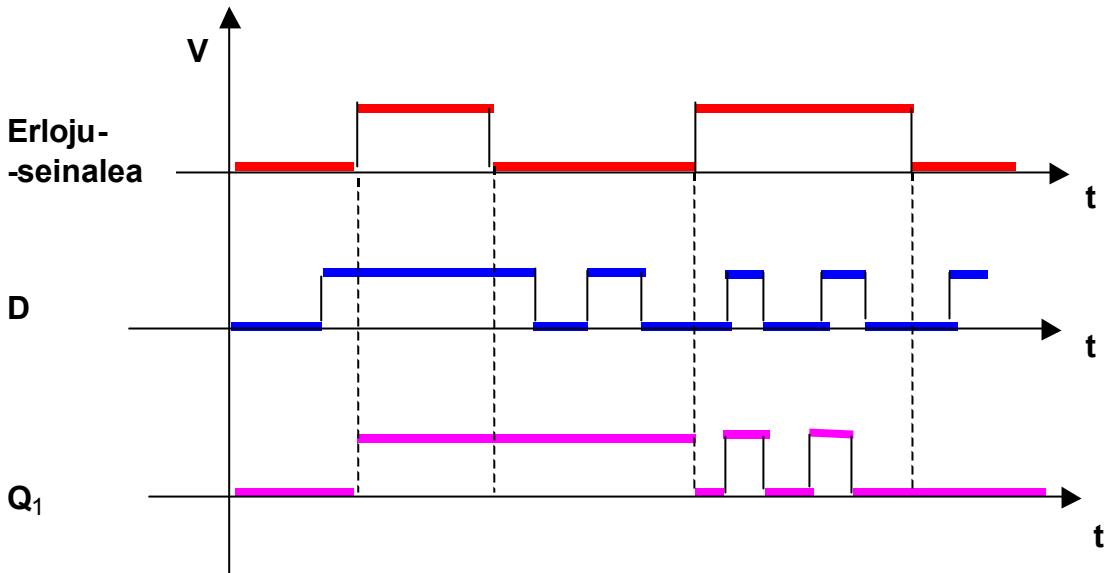


#### 6.1.1.6. Mailaz aktibaturiko D biegonkor sinkronoa

Osagai honek D datu-sarrera, erloju-sarrera eta, betiko moduan, bi irteera ditu, Q eta bere osagarria. Q irteeraren balioa, betiere, D sarreraren balioaren berdina izango da, erloju-seinalea altua bada, behinik behin. Erloju-seinalea eteten bada, biegonkorra uneko informazioa mantenduko du irteeran, irudian ikus daitekeen bezala.

D	C	$Q_n$	$Q_{n+1}$
0	0	0	$0(Q_n)$
0	0	1	$1(Q_n)$
0	1	0	0
0	1	1	0
1	0	0	$0(Q_n)$
1	0	1	$1(Q_n)$
1	1	0	1
1	1	1	1





#### 6.1.1.7. Saihetzen eraginez aktibatutako biegonkor sinkronoak

Ziurtatu ahal izan dugunez, mailaz aktibatutako biegonkorretan, sarreran egindako aldaketak irteeran nabarmenduko dira, clock- seinalean maila altua baldin badago. Funtzionatzeko era horrek, ordea, arazoak sor ditzake sarrerako uhinen komutazioa maiztasun handikoa bada. Erloju-seinaleko maila altuaren edo aktiboaren iraupena laburtzea ez da irtenbidea kasu hauetan. Erloju-seinaleko aldaketaren eragina azkartasunez nabaritzeko, bi prozedura erabil ohi dituzte egungo ekoizleek integratuetan:

- a) **Edge-Triggered konfigurazioa.** Horrek saihets bidezko desarra adierazi nahi du, hau da, erloju-seinaleko sarrerak goranzko saihetsa edota beheranzko saihetsa jasotzen dituenean, biegonkorren informazioak eragina du irteeran. Normalki, erloju-sarrera horrek goranzko saihetsa nabaritzen duenean aktibatzen du biegonkorra.
- b) **Master-Slave konfigurazioa.** Bi etapaz osatuta dago, lehenari **Maisu** izena ematen zaio eta bigarrenari **Morroi**. Etapa horietako bakoitza biegonkor asinkrono bat izaten da, konbinaziozko zirkuitu baten bitartez lotuta.

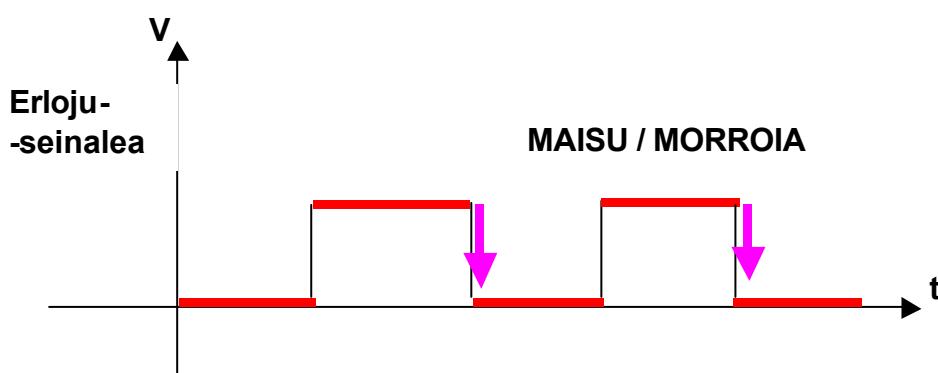
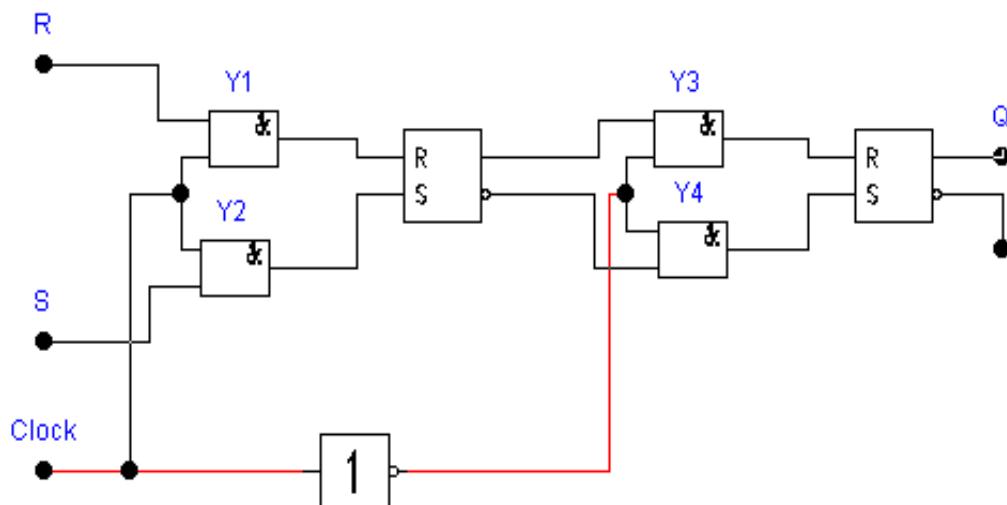
Ondoren ikusiko ditugun biegonkor sinkronoak lehen aipatutako arazoei aurre egiten dieten zirkuituak dira. Bi aukera edo konfigurazioak ikustearren, batzuk Edge-Triggered (saihetsez) aktibatutakoak eta beste batzuk Master-Slave (Maisu/Morroi) eran aktibatuak izango dira.

- **R-S biegonkorra** maisu/morroi erakoa.
- **J-K biegonkorra** saihetsez aktibatua
- **D biegonkorra** saihetsez aktibatua
- **T biegonkorra** saihetsez aktibatua

#### 6.1.1.7.1. R-S Maisu/Morroi biegonkorra

Irudian ikus daitekeenez, erloju-seinalea 0 mailatik 1 mailara pasatzean, R eta S sarreretako informazioa, lehen biegonkor asinkronoan, Maisu izanekoentz sartzen da  $Y_1$  eta  $Y_2$  AND ateen bidez. Bigarren biegonkorren sarreran, ordea, ez dugu informaziorik oraindik,  $Y_3$  eta  $Y_4$  ateek mugatzen baitute.

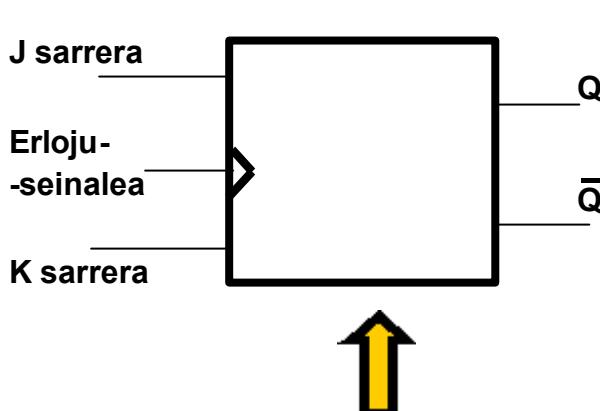
Erloju-seinalea 0 bada, **Maisuan** gordetako informazioa **Morroira** pasatzen da, orain  $Y_3$  eta  $Y_4$  ateek eroaten baitute. Une berean, **Misuak** ez du informaziorik jasoko,  $Y_1$  eta  $Y_2$  ateek mugatzen dutelako.



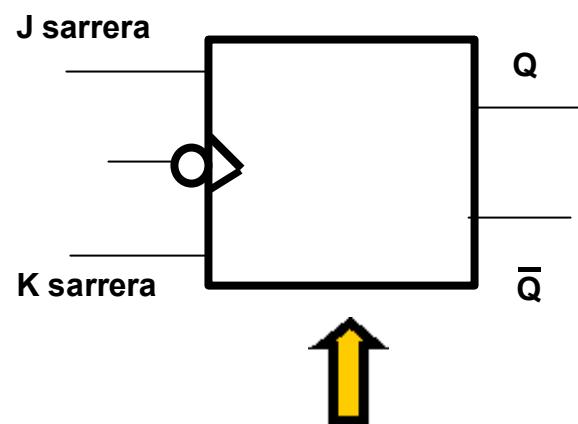
#### 6.1.1.7.2. J-K biegonkorra saihetsez aktibatua

Saihetsez aktibatutako J-K biegonkor garrantzitsuenak SN7470, SN 7473 eta SN 7476 dira.

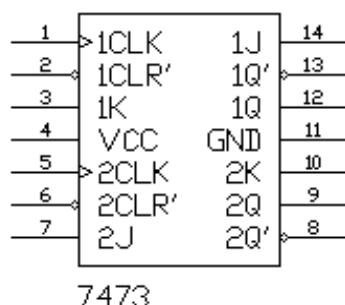
Batera edo bestera eraikitakoa izan, J-K biegonkorra ondorengo ikurraz ezagutuko dugu:



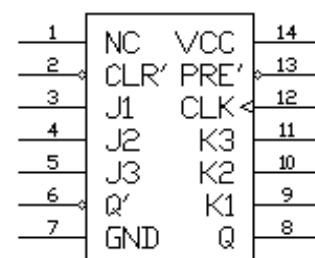
Desarra goranzko saihetsean



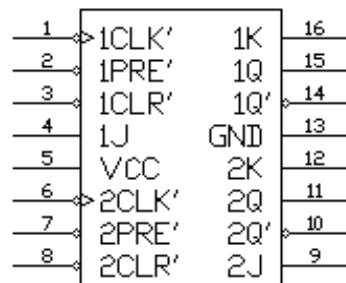
Desarra beheranzko saihetsean



7473



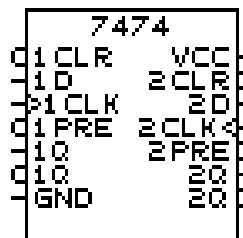
7472



7476

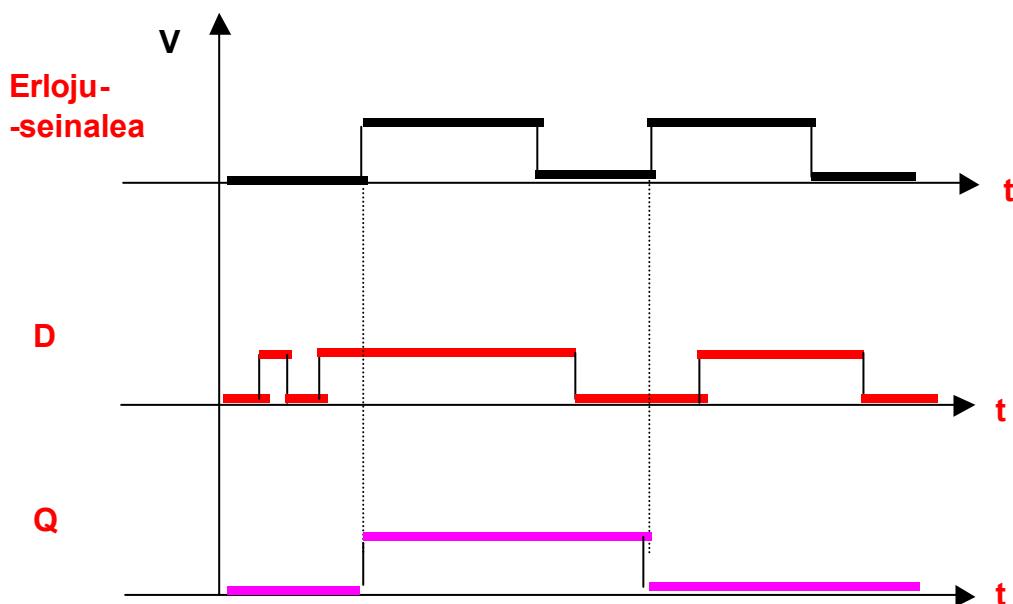
#### 6.1.1.7.3. D biegonkorra saihetsez aktibatua

Osagai hau D motako FLIP - FLOP gisa landuko dugu aurreratzean, nolabait D motako LATCH zirkuitutik bereiztearren. Biegonkor-mota hau lantzeko, zirkuitu komertzialetan topa daitekeen SN7474 erabiliko dugu, hain zuzen ere, goranzko saihetsean aktibatzen denetarikoa, hauetako gehienak bezalaxe.



Irudian ikus daitekeenez, zirkuitu hau datu-sarrera den **D** sarreraz gain, erloju-seinalerako sarrera bat eta bi irteera osagarriez osatuta dago oinarrian. Baditu, ordea, integratu gehienetan topa daitezkeen beste bi sarrera. Bata **CLEAR** (RESET ere deiturikoa), eta **SET** bestea. Lehena aktibatuz irteera "0" mailara eramango dugu eta bigarrena aktibatuz "1" mailara, **D** sarrerako informazioa kontuan hartu gabe.

Ondorengo irudian denbora-diagrama ikus daiteke. Ikus daitekeenez, goranzko saihetsean bideratzen du informazio-irteera; ondoren, nahiz eta erloju-seinaleko sarrera-maila altua izan aldaketa berriak nabarmentzeko, Q irteera aurreko egoeran mantentzen da.

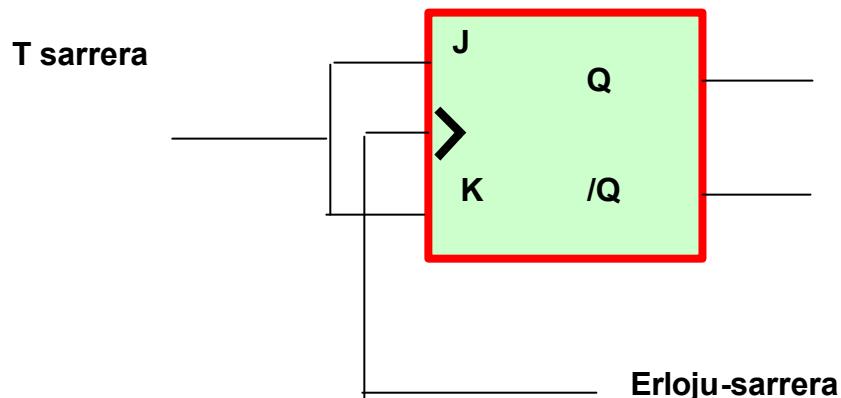


#### 6.1.1.7.4. T biegonkorra saihetsez aktibatua

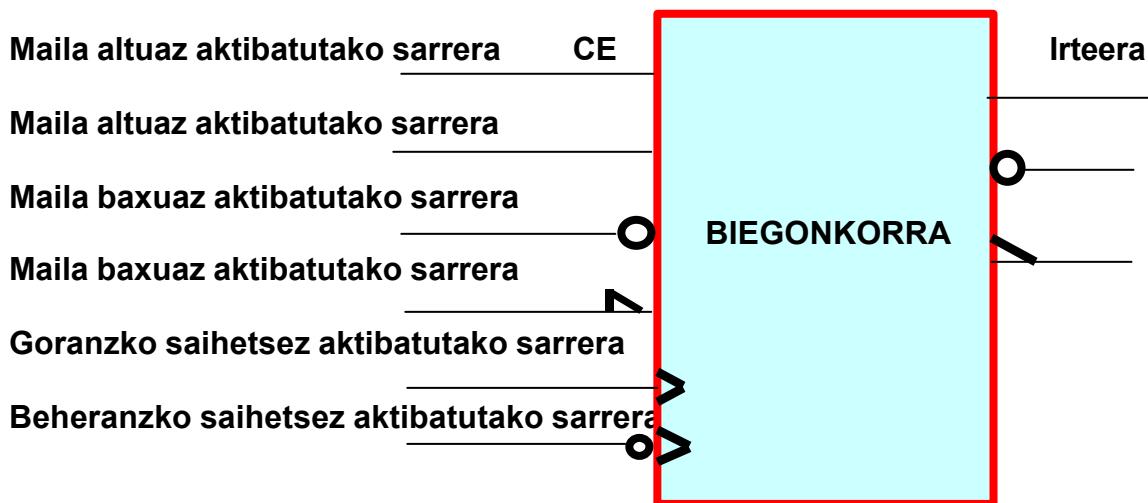
T biegonkorra ez dira zirkuitu integratuetan eraikitzen, hala ere oso erabiliak dira batik bat kontagailuetan eta desplazamendu-erregistroetan.

Saihetsez aktibatutako T biegonkor baten eraikuntza oso erraza da, J-K biegonkor bat edukiz gero. Bi datu-sarrerak lotu besterik ez dugu, eta lotura horri ematen zaio T izena.

Ondoan agertzen den irudian, Saihetsez aktibatutako T biegonkorriari dagokion ikurra ikus daiteke



Hurrengo puntuak honelako zirkuituetan erabili ohi den sinbologia ikusiko dugu



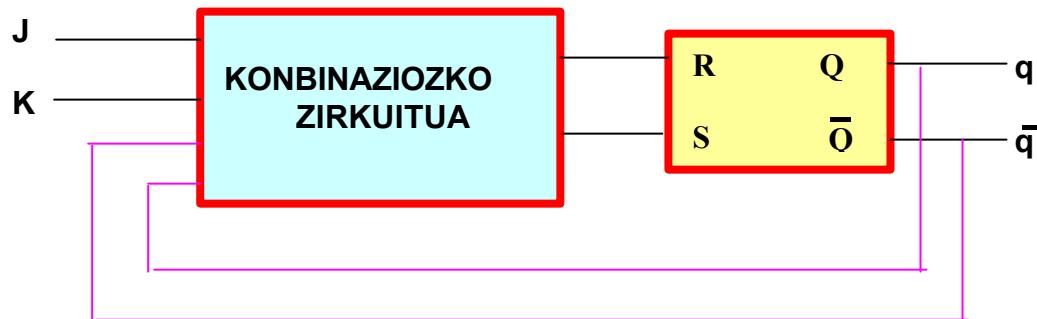
#### 6.1.1.7.5. Biegonkorren arteko eraldaketak

Puntu honetan, biegonkorren arteko eraldaketak nola egiten diren jakiteko, adibide bat besterik ez dugu ikusiko.

**NOR ateez eraikitako R-S biegonkorretik abiatuz, J-K biegonkorra lortu.**

Horretarako, konbinaziozko zirkuitua diseinatu beharko dugu, R-S biegonkorrek elkarlanean J-K biegonkorren funtzioa bete dezaten.

Lehendabizi egia-taula egin behar da, 6.2. irudiari begiratuz: zirkuituaren sarbideak J, K, q (Q) eta  $\bar{q}$  ( $\bar{Q}$ ) dira; irteerak, berriz, R eta S



6.2. irudia

Egia-taula egiteko, **R-S eta J-K biegonkorrei** begiratuko diegu.

J	K	Q <sub>n</sub>	R	S
0	0	0	X	0
0	0	1	0	X
0	1	0	X	0
0	1	1	1	0
1	0	0	0	1
1	0	1	0	X
1	1	0	0	1
1	1	1	1	0

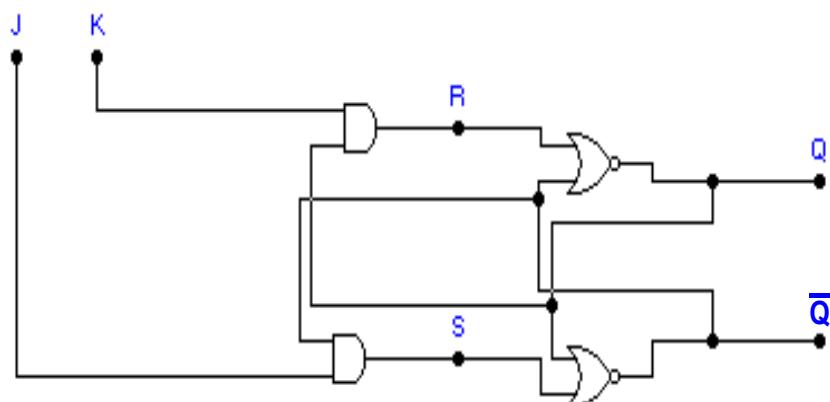
Errepara diezaigun lehen errenkadari:

- $J = 0, K = 0$  eta  $Q_n = 0$ ; kasu honetan, J-K biegonkorrauen  $Q_{n+1} = 0$  dela ikus genezake horren egia-taulan.
- Orain R-S egia-taulan,  $Q_n = 0 \rightarrow Q_{n+1} = 0$  trantsizioa  $R = S = 0$  eta  $R = 1$  eta  $S = 0$  balioekin egiten dela egiaztatzen dugu.
- Hau da, Rren balioa **zehazgabea (X)** izan daiteke eta Sren balioak 0 izan behar du.

Karnaughen metodoa erabiliz, Rren eta Sren funtzio laburrak lortzen ditugu:

$$R = K \cdot Q_n \quad \text{eta} \quad S = J \cdot \bar{Q}_n$$

Ondoko irudian daukagu zirkuitu osoa:



## 6.2. KONTAGAILUAK

---

Kontagailua erabilera orokorreko zirkuitu sekuentziala da, non irteerek sarreran ezartzen diren pultsuen kopurua kode jakin batez adierazten baitute.

Bata bestearekin konektaturiko biegonkorrez osatzen da. Horien irteerek egoera aldatzen dute sarreretan pultsuak ezartzen direnean.

Kontagailu baten ahalmena bere irteeretan, kode bitarretako edozeinetan, adieraz daitekeen zenbakirik altuena da.

Gehienez kontatu nahi diren pultsuen kopurua gehi bat (bat horrek sarreran pultsurik ez dagoela adierazten du) eginez gero, egoera posibleen kopurua lortuko dugu.

Kontagailuak, bere ahalmenaren barruan dagoen balio maximora heltzen denean, berriro ere zerotik hasita kontatzeari ekiten dio hurrengo pultsua ezartzean.

Kontagailuak bi eratakoak izan daitezke:

- **Sinkronoak**
- **Asinkronoak**

Sinkronoetan, erloju-pultsuak aldi berean ezartzen zaizkie biegonkor guztiei, eta, ondorioz, irteera guztiak aldi berean aldatzen dira.

Asinkronoetan, erlojuaren seinalea lehen biegonkorraren sarrerari ezartzen zaio, horren irteera hurrengo biegonkorraren sarrerari eta, halaxe, bata bestearen atzetik. Horrelako gailuen hedapen-denbora sinkronoenean baino luzeagoa da.

Eragiketaren arabera, kontagailuak honelakoak izan daitezke:

- **Goranzkoak**
- **Beheranzkoak**
- **Bietakoak**

Goranzkoetan, euren pultsu bakoitzarekin gehitu egiten dira.

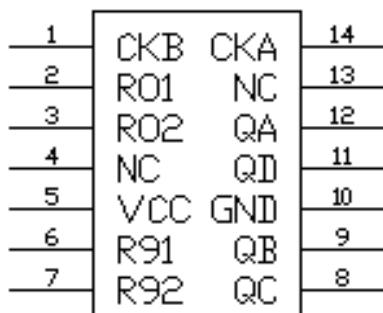
Beheranzkoetan, euren edukia gutxitu egiten da.

Bietakoak (*up/down* kontagailuak).

### 6.2.1. KONTAGAILU ASINKRONO HAMARTARRA (SN7490)

Kontagailu asinkrono hamartar hau saihets negatiboz desarraturiko lau J-K biegonkorrez osatua dago.

Lehenengo biegonkorra ez dago besteekin konektatua, hartara malgutasun handiagoa lortzen da erabiltzeko orduan.



7490

**R91 eta R92** kontagailua pizteko (irteera guztiak 1ean) sarrerak dira, maila altuan daudela betetzen dute beren funtzioa.

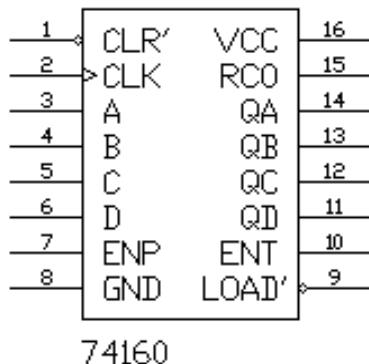
**R01 eta R02** kontagailua ezabatzeko (irteera guztiak 0an) sarrerak dira; horretarako, biek maila altuan egon behar dute.

Otik 9ra kontatu nahi badugu, **Q0 ® QB**-ra konektatu beharko dugu.

### 6.2.2. KONTAGAILU SINKRONOA SN 74160

Kontagailu hau sinkronoa da: barne-biegonkor guztiak batera komutatzen dira, erloju-seinalez.

Programagarria da: **A.....D** sarrera paraleloetan dagoen informazioa kontagailuari karga dako **LOAD** sarreran maila baxua jarrita.

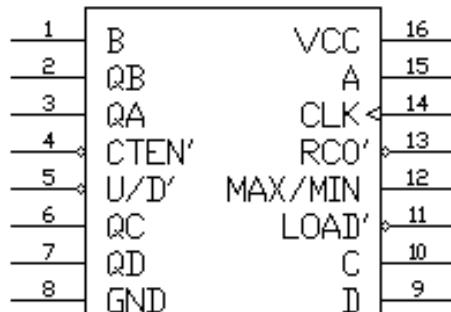


Beraz, pinen funtzioa honakoa da:

- \* **CLR**: ezabatzeko sarrera da. Irteera guztiak 0an ipintzen ditu, bere maila aktiboa baxua da.
- \* **ENP** eta **ENT**: eragozpen-sarrerak dira (gaitzea). Biek maila altuan egon behar dute gailuak konta dezan.
- \* **CLK**: erloju-sarrera. Saihets positiboa da aktiboa.
- \* **RCO**: bururako irteera (*carry*) da, 9 kontatzen duenean, maila altua ematen du.
- \* **LOAD**: karga-sarrera (LOAD) da. Maila baxua dagoenean, erlojuaren saiheits positiboa sarrera paraleloetan dagoen datua kargatzen da kontagailuan.

### 6.2.3. UP/DOWN KONTAGAILUA (SN74190)

Kontagailu hauek kontatu eta deskontatu egin dezakete. SN 74190 kontagailuak pultsuak 14. hankatxotik hartzen ditu, eta 5. hankatxoan duen mailaren arabera kontatu eta deskontatu egingo du (**H** → *down* eta **L** → *up*).



74190

- Kontatzeko, **LOAD** sarrerak maila altuan egon behar du.
- Ezabatzeko, sarrera paraleloetan 0 ipini eta kargatu, ez dauka CLR sarrerarik.
- **RCO (ripple clock)**: maila baxua ematen du azkeneko kontaketan.
- **MAX/MIN**: maila altua ematen du azkeneko kontaketan.
- **Fmax = 20 MHz**

## 6.3. DESPLAZAMENDU-ERREGISTROAK

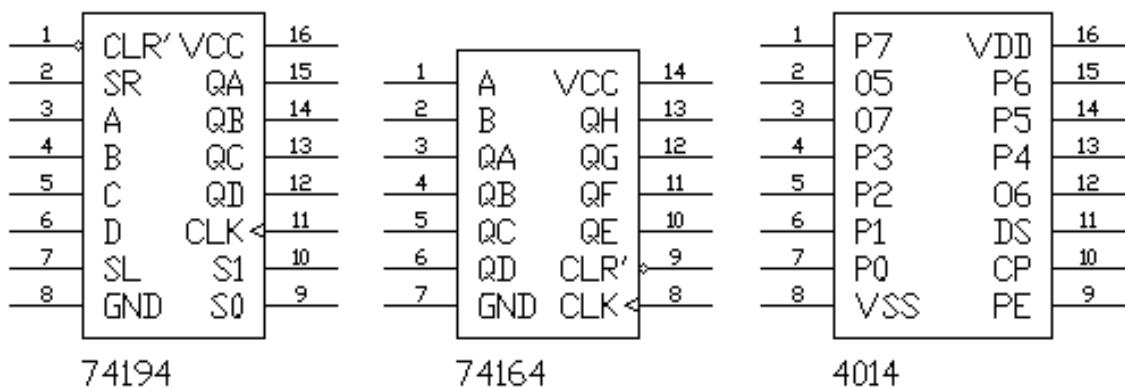
Desplazamendu-erregistroak, kontagailuak bezala, erabilera orokorreko zirkuitu sekuentzialak dira, kaskadan konektaturiko hainbat biegonkorrez osatuak.

Biegonkorrek bit bat gorde dezaketen bezala, erregistroek hitz bitar bat gorde dezakete. Zenbat biegonkorrek osatzen duten erregistroa, hainbat bitek osa dezakete hitza. Datuak metatzeko eta gordetzeko ahalmena izateaz gain, erregistroek informazioa sarreretan paraleloan sartu eta irteeretan seriean edo paraleloan ateratzeko ahalmena dute, baita seriean sartu eta paraleloan edo seriean ateratzeko ahalmena ere.

Beraz, erregistroak honako lau talde hauetan sailka daitezke:

- **Serieko sarrera, serieko irteera.**
- **Paraleloko sarrera, paraleloko irteera.**
- **Serieko sarrera, paraleloko irteera.**
- **Paraleloko sarrera, serieko irteera.**

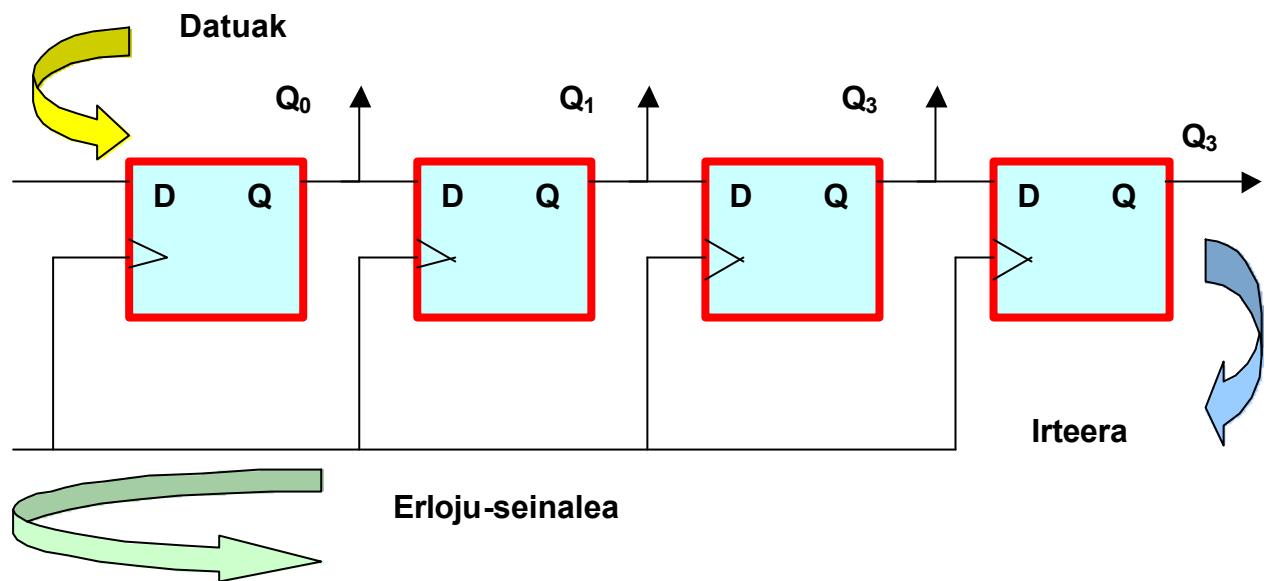
Horiez gainera, badira zenbait erregistro informazioa noranzko bietan erabil dezaketenak. Halako gailuei noranzko bikoak deitzen zaie.



### 6.3.1. SERIEKO SARRERA – SERIEKO IRTEERA ERREGISTROA

Erregistrorik oinarrizkoena biegonkorrez osaturik dagoena eta informazioa seriean sartu eta seriean ateratzen duena da.

Irudian, saihets positiboz desarraturiko D motako lau biegonkorrez osaturiko erregistro bat erakusten da.

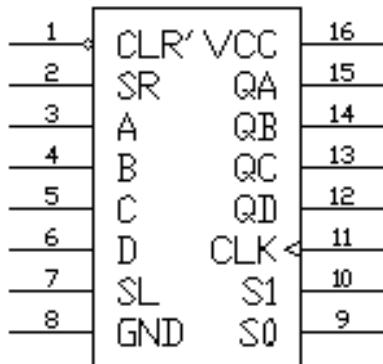


Serie – serie erregistroetan datuak banan-banan sartzen dira, baina datuak aldi berean sartu ahal dira (irakurketa paraleloa). Horretarako, informazioa sarreretan prestatzen da, eta nahi denean, pultsu batez, automatikoki erregistratzen da.

Ondoren, 74194 gailua azaltzen da: zirkuitu integratuz osatuta dago eta azaldu ditugun eragiketa-mota guztiak egin ditzake.

1	CLR' VCC	16
2	SR QA	15
3	A QB	14
4	B QC	13
5	C QD	12
6	D CLK	11
7	SL S1	10
8	GND S0	9

74194

**74194ak noranzko bietan eraman dezake informazioa**

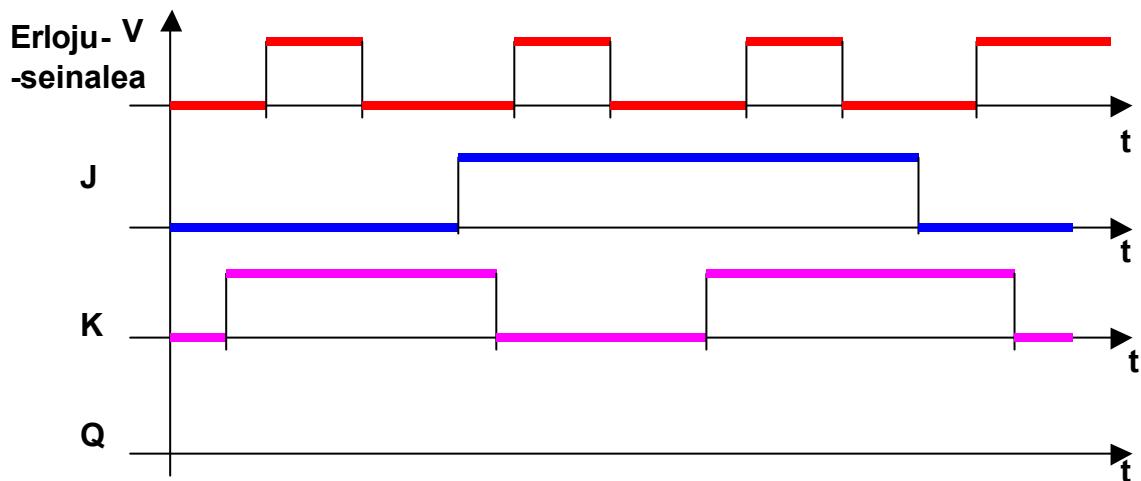
74194

S0	S1	ERAGIKETA-MOTA
1	1	D0...D3 sarreretako informazioa Q0...Q3 irteeretara pasatzen du.
0	1	Ezkerrerantz desplazatu
1	0	Eskuinerantz desplazatu
0	0	Ez dago desplazamendurik

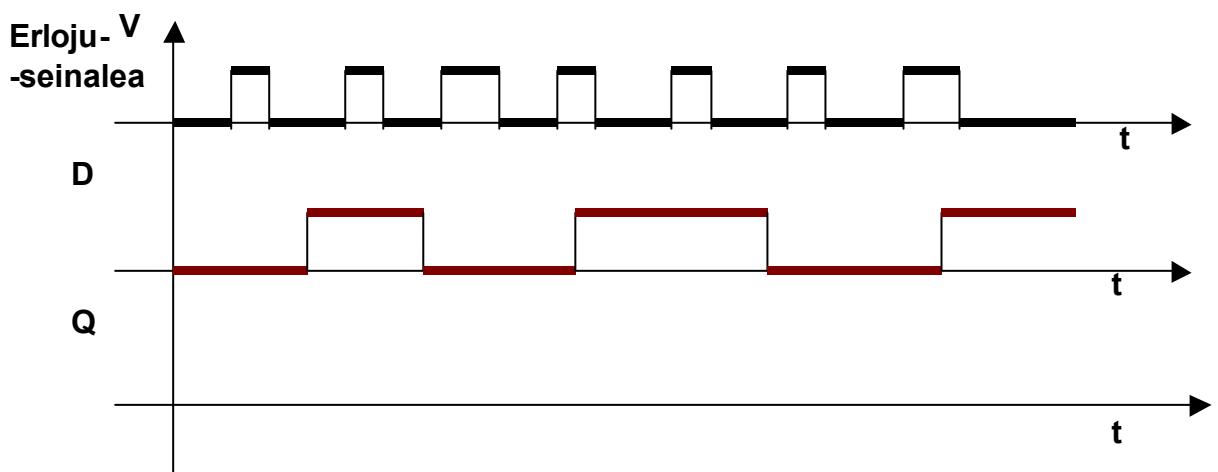
- A, B, C eta D datuak paraleloan sartzeko sarrerak dira.
- DSR datuak seriean sartzeko, eskuinerantz desplazatu nahi denean.
- DSL datuak seriean sartzeko, ezkerrerantz desplazatu nahi denean.
- CLR ezabatzeko sarrera da.

## Ariketak

1. J-K biegonkorri (7476 motakoa) ondoko kronogramaren seinaleak aplikatu eta Q irteeraren seinalea lortu.



2. Eraiki ezazue D erako latch-a, NOR R-S batetik abiatuz.
3. D biegonkorri (7474 motakoa) ondorengo seinaleak sartuko bagenizkio, zer seinale lortuko genuke Q irteeran?



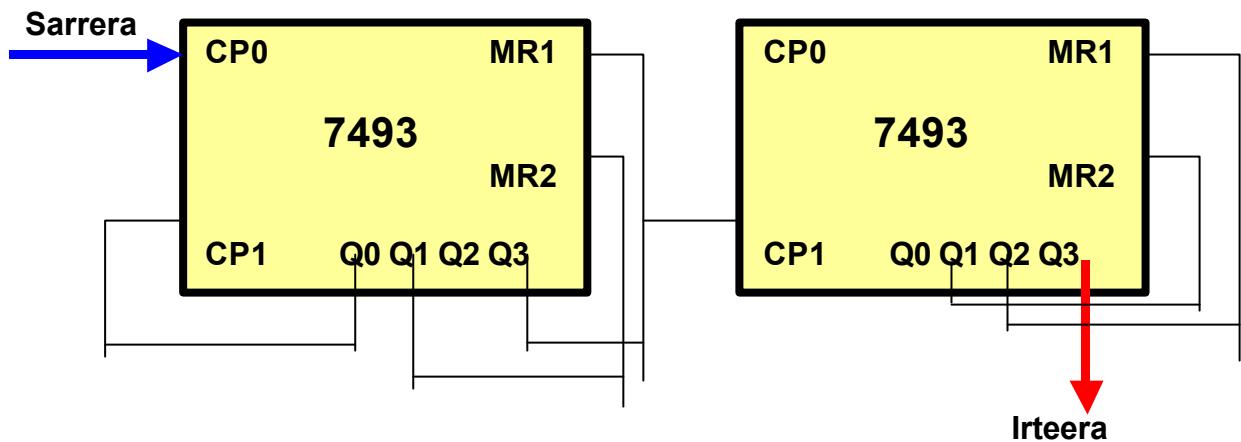
4. 7493 zirkuitu integratua erabiliz, 0tik 9ra zenbatzen duen kontagailua eraiki.
5. Zirkuitu bera erabiliz, 13 moduluko kontagailua eraiki.
6. Konekta ezazue 74163 kontagailua, 5etik 12ra konta dezan.
7. Konekta ezazue 74160 kontagailua, 2tik 9ra konta dezan.
8. Konekta itzazue 74190 motako bi kontagailu, 0tik 99ra zenbatu eta deskonta dezaten.
9. Lau J-K biegonkor erabiliz, eraiki ezazue serie-serie desplazamendu-erregistroa. Datuak eta erloju-seinalea kommutadore logikoz ezarri. Irteerak LED diodoetara atera.
10. *Universal Trainer* deritzon entrenatzaile logikoan mutua ezazue 74194 desplazamendu-erregistroa, eta egiaztatu bere funtzionatzeko era guztiak. Sartu 1.000 datu, eta gero eratzun-kontagailu bezala funtzionarazi, ezkerretara lehenbizi eta gero eskuinetara. Erloju-seinalea uhin-sorgailuz ezari ( $F=1\text{Hz}$ ).
11. Zer zirkuitu sekuentzial egingo zenukete ondorengo ekuazioa bete dezan?

$$F_1 = F_2 / 8$$

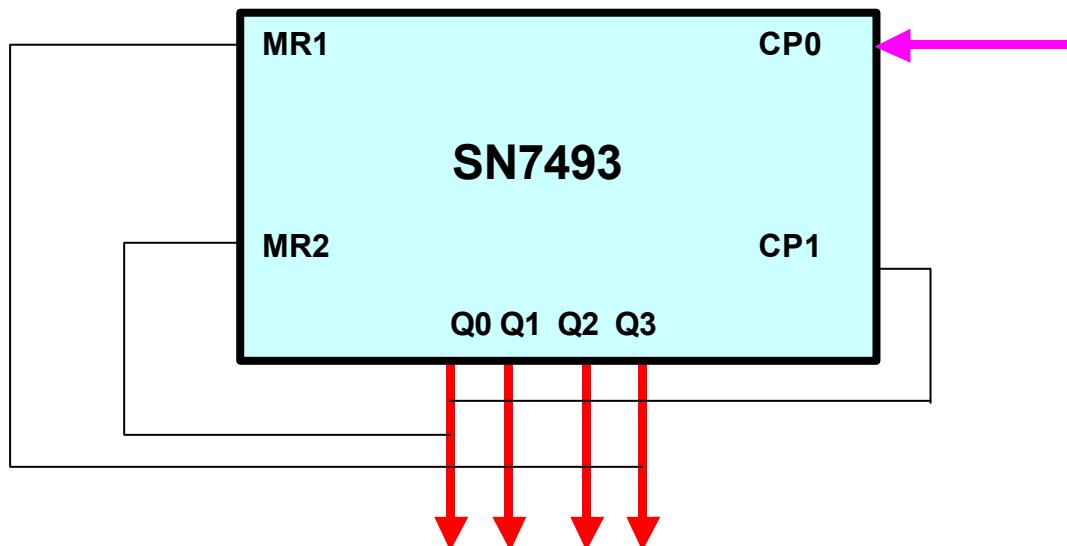
$F_1$  = irteerako maiztasuna  
 $F_2$  = sarrerako maiztasuna

## Ariketa ebatziak

- Hurrengo ariketan, 7493 zirkuitu integratuak erabiliz, 60 moduluko kontagailua eraikiko dugu:



- 7493 kontagailuaren pinak ondo konektatu 0tik 8ra zenba dezan. Horrek esan nahi du 9 pultsuan berriro 0an jarri beharko duela.



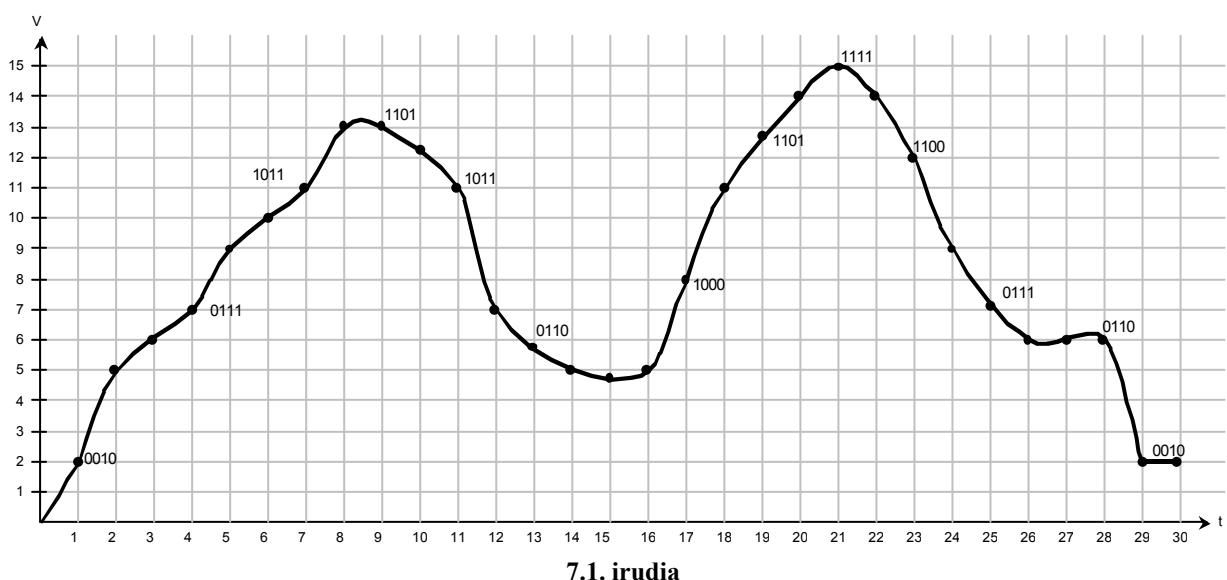
# 7. KAPITULUA

## AD/DA bihurgailuak

### 7.1. SEINALE ANALOGIKOAK ETA DIGITALAK

Magnitude analogiko batek muga batzuen artean balio jarraituak hartzen ditu, eta seinale digital batek, aldiz, balio diskretuak. Ezberdintasun hori adierazteko, har dezagun 0V eta +15 Va artean aldatzen den tentsioa. Magnitude horren adierazpen analogikoak 0 eta +15Va arteko infinitu diren balio *guztiak* hartzen ditu.

Seinale hori digitalki adierazi nahi badugu, 4 biteko kode bitarra erabiliz, 16 balio bakarrik defini ditzakegu. Balio gehiago nahi izanez gero, bit gehiago erabili beharko ditugu. Horrela, edozein balio analogiko kode digital baten bitartez adieraz dezakegu, eta kode digitaleko bit-kopuruak doitasun-maila markatzen du. Ondoko 7.1. irudian azaltzen da kontzeptu hori; irudikatzen den funtziotan analogikoa 0-15Va artean aldatzen den kurba jarraitua da.

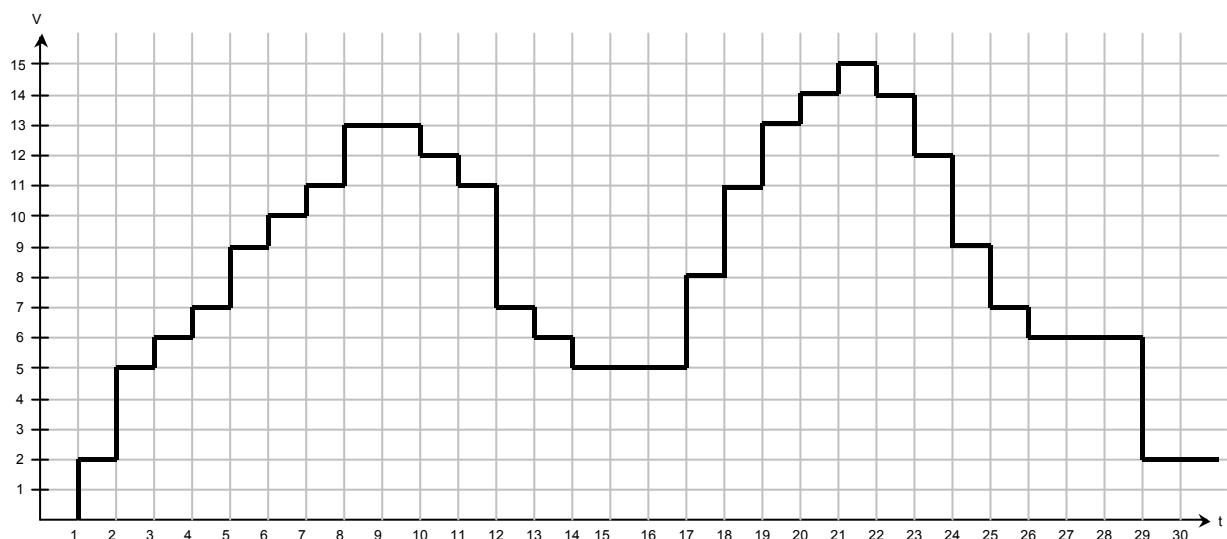


7.1. irudia

Kurba hori lagindu eta neurtu egiten da hogeita hamar tarte ezberdineta. Tarte bakoitzeko tentsioa lau biteko kodeaz adierazten da (tarte batzuetakoak idatzira dira irudian). Horrela, kurba analogikoaren zenbaki bitarrez adierazitako zenbait tentsio-balio ditugu. Hau da bihurketa analogiko/digitalaren oinarrizko ideia.

Zenbaki digital horien sekuentziari jarraituz, kurba analogikoa berregin daiteke; ez da izango berdin-berdina, balio-kopuru mugatua adierazten dugulako (kasu honetan, 30). Hogeita hamar puntuengatik balio digitalak grafikora eramanez gero lortzen dugu funtzioa. Ikus dezakegunez, grafikoa ez da jatorrizko kurbaren berdina, puntuengatik arteko balioak ezagutzen ez ditugulako. Zehaztasun handiagoa lortuko dugu kurbaren laginak sarriago hartzen baditugu eta lagin bakoitzak hartzeko erabilitako bit-kopurua handitzen badugu.

Ondoko 7.2. irudian ikusten dugu 7.1. irudian irudikatu dugun seinale analogikoaren emaitza digitalizatua.

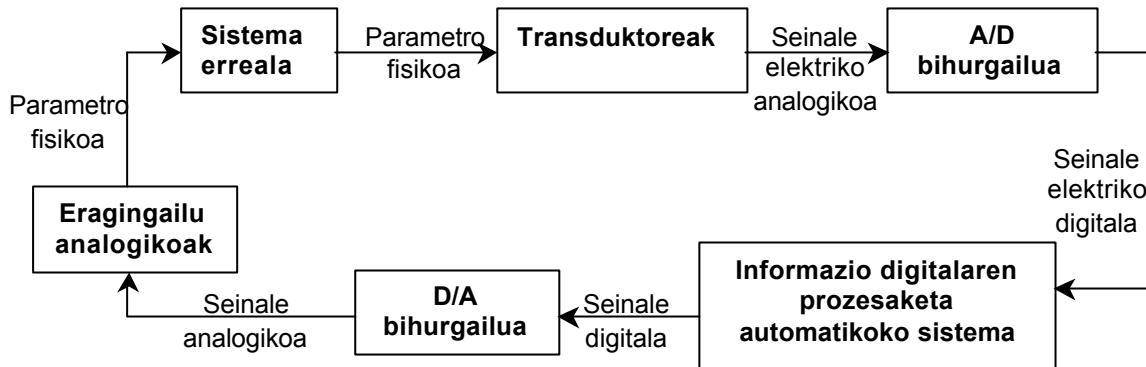


7.2. irudia

Eremu fisikoan sortzen den “informazio” gehienak izaera analogikoa du. Hau da, “informazio” horrek, muga batzuen artean, etengabeko aldaketa nozitzen du denboran zehar.

Informazioa askoz errazago prozesatzen (amplifikatu, iragazi, metatu, etab.) da automatikoki digitalizatua badago; horregatik, askotan informazio analogikoa digital bihurtu behar izaten da. Bihurketa hori burutzen duten sistemak A/D bihurgailuak dira.

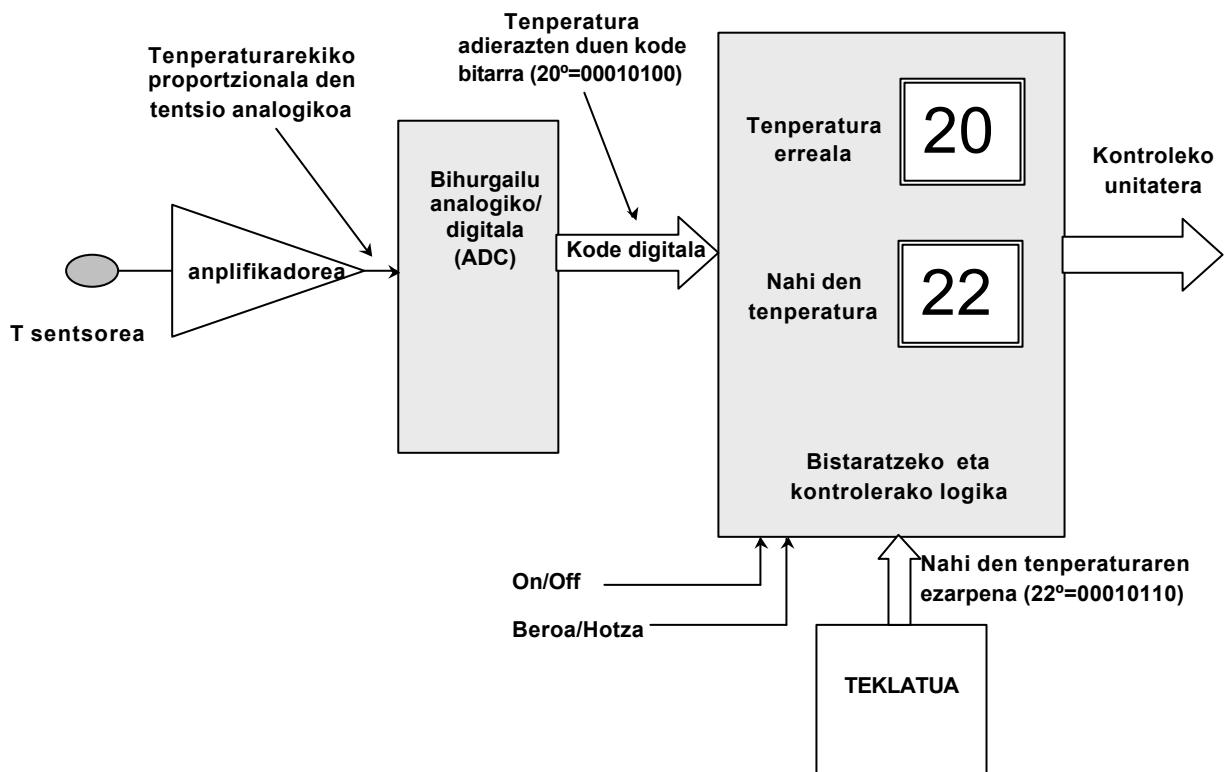
Prozesaketa digital horrek eragingailu analogikoak kontrolatu edo elikatu behar dituzten seinale digitalak sortzen ditu, eta horretarako beharrezko egokitzapena D/A bihurgailuak egiten du. Ondoren agertzen den 7.3. irudian ikusten da sistema erreal baten kontrol automatikoaren oinarrizko eskema.



7.3. irudia

Gure ingurune teknologiko hurbilean hainbat adibide ditugu:

- **Termostato elektronikoa.** Gela bateko sentsoreak temperaturarekiko proportzionala den tensio analogikoa sortzen du; tensio hori, amplifikatu ondoren, analogiko/digital bihurgailuari aplikatzen zaio (ADC, *analog-to-digital converter*), eta kode bitarrean jartzen da. Ondoren, digitalki prozesatzen da, kontsigna batez konparatzeko edota temperatura bistaratzeko.



7.4. irudia

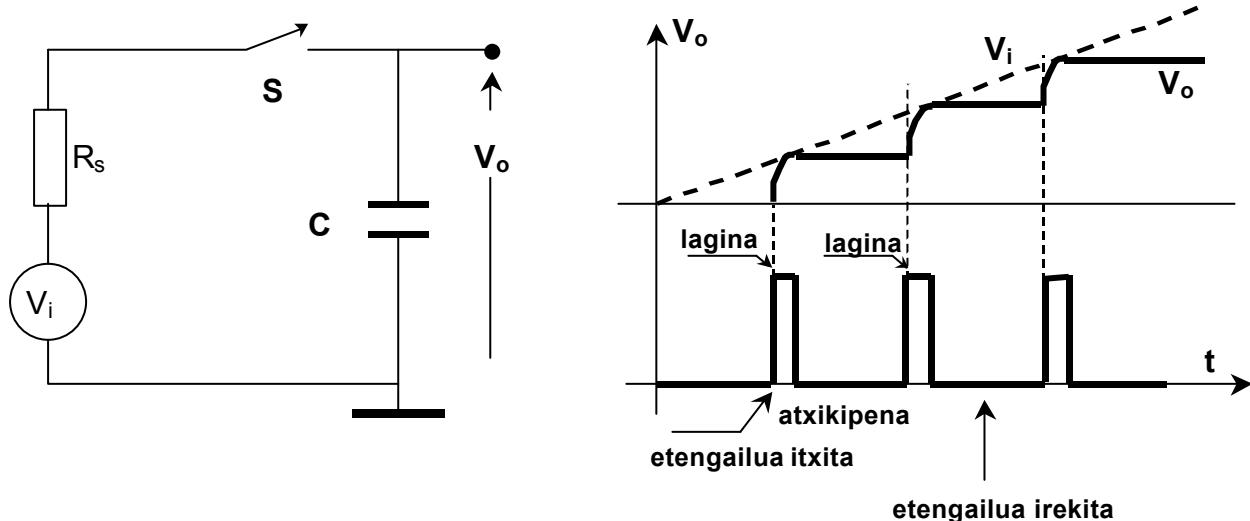
- **CD erreproduzitzalea.** Audioko seinalea era digitalean grabatzen da CDan, laser izpiez detektatzen (irakurtzen) diren zulotxo batzuk dituelarik. Seinale kodetu hori amplifikatu ondoren, ditalietik analogikorako bihurgailuak (DAC) formatu analogikoan jartzen du, eta gero, amplifikatu ondoren, bozgorailuei aplikatzen zaie.

## 7.2. LAGINKETA ETA ATXIKIPENA (SAMPLE/HOLD)

Zirkuitu horiek denbora-tarte zehatz batean seinale analogiko baten laginketa egiteko erabiltzen dira, gero lagin hori nahi den beste denboran atxikitzeo. Laginketaren denbora-tarteak eta atxikipenaren denbora kontroleko seinale logiko baten menpe daude. Atxikipenen arteko tarteak aplikazioen araberakoa da.

Laginketa eta atxikipeneko zirkuitu gehienak kondentsadore bat erabiltzen dute lagin-tentsioari eusteko.

Elektronikoki kontrolaturiko etengailuen bidez kargatzen da kondentsadorea lagin-tentsioraino, gero sarrera azkar irekiz, kondentsadoreak mantentzen dezana nahi den tentsio hori.



7.5. irudia. Laginketa/atxikipeneko zirkuitua eta seinalea.

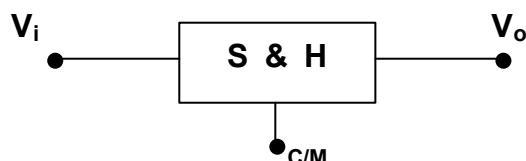
Irudian ikusten dugu horrelako zirkuitu bat. Iturri analogikoa  $V_i$  da eta  $R_S$  barne-erresistentzia. Etengailua ixten bada eta  $R_S \cdot C$  denbora-konstantea txikia bada,  $V_o = V_i$ , eta balio horrekin jarraituko du  $S$  irekitzen denean;  $V_o$  konstante mantentzen da atxikipenak irauten duen bitartean (kondentsadorea ezin da deskargatu).

Praktikan etengailuak eta kondentsadoreak idealak ez direnez, ezaugarri garrantzitsuak dira **irekitze-denbora** eta **eskuratze-denbora**.

**Irekitze-denbora.** Kontrolerako logikak irekitzeko agindua ematen duenetik etengailua irekitzen den bitarteko denbora-tartea da. Milisegundoetakoa onartzen bada,  $S$  erreala izan daiteke FET transistoreak erabiliz, ordea, 50 ns-koa izaten da. Etengailua aukeratzerakoan kontuan hartu bere irekitze-denborak laginketaren erritmoaren alderantzizko baino askoz txikiagoa izan behar duela.

**Eskuratze-denbora.** Laginketa-agindua eman denetik irteerako tentsioa sarrerakoaren berdina izatera heltzen den bitarteko denbora da. Denbora horretan zerikusi handia du  $R_S$ -ren balioak. Horren balioa txikiagoa izan dadin, sarrera analogikoaren eta etengailuaren artean amplifikadore operazional bat tartekatzen da tentsio-jarraitzairez bezala.

Zirkuitu horien adierazpen sinplifikatua 7.6. irudikoa da. A/D bihurgailuetan erabiltzen dira.



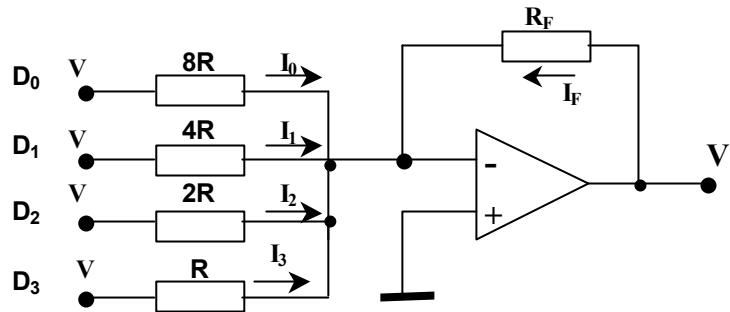
7.6. irudia

## 7.3. BIHURGAILU DIGITAL/ANALOGIKOAK

Bihurgailu hauek egiteko amplifikadore operazionala erabiltzen da. Era honetako bi mota aztertuko ditugu

### 7.3.1. ERRESISTENTZIA HAZTATUA DITUZTENAK

Anplifikadore operazionalez egindako batutzaileak dira; bere sarrerako erresistentziak haztatuak dituztenak. Kasu honetan, erresistentzia bakoitza aurrekoaren bikoitza da (irteeran kode bitarra lortzeko).



7.7. irudia

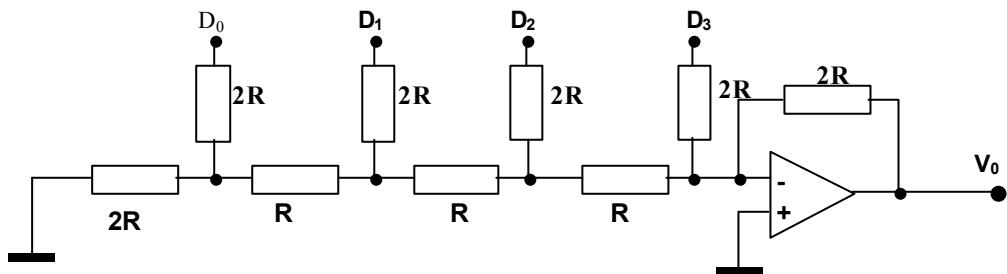
Jakina denez:

$$V_o = -I_f \cdot R_f = -(I_o + I_1 + I_2 + I_3) \cdot R_f \quad \text{eta} \quad V_o = -\left(\frac{V}{8R} + \frac{V}{4R} + \frac{V}{2R} + \frac{V}{R}\right) \cdot R_f$$

Ikusten denez, irteera sarreren pisuekiko proportzionala da. Zirkuitu hauen eragozpena balio ezberdineko erresistentzia-kopuruan datza; izan ere, balio estandarrekin zaila da doitasun minimoa izango duen DACa egitea.

### 7.3.2. R/2R ESKAILERA DUTENAK

Beste metodo bat 7.8. irudian agertzen dena da, lau bitekoa.



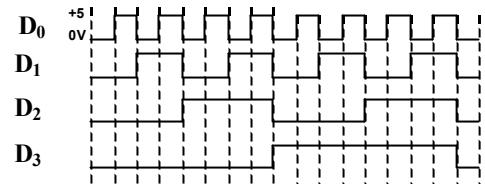
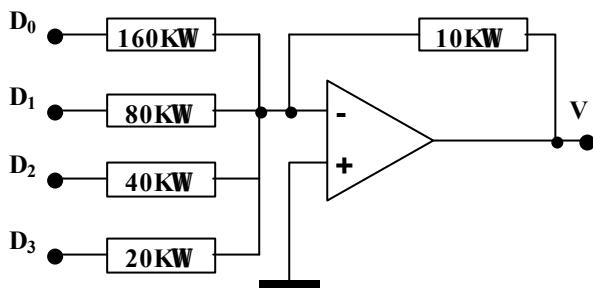
7.8. irudia

Sarrera bakoitzak irteeran ondoko pisu handiagokoaren tentsioaren erdia ematen duela egiazta daiteke. Horregatik, irteerako tentsioa sarrerako biten pisu bitarrarekiko proportzionala da.

Horrela, aurreko DACak zuen eragozpena konpontzen da, erresistentzia-balio ezberdin bi bakarrik behar baititu.

**Ariketa**

Aurkitu 7.9. irudiko DACaren irteera, sarreretan alboan agertzen diren seinaleak aplikatu ondoren (A sarrera). D<sub>0</sub> da pisu txikieneko bita (LSB).



7.9. irudia

Ebazpena:

Lehenik, sarrera bakotzeko korrontea kalkulatuko dugu:

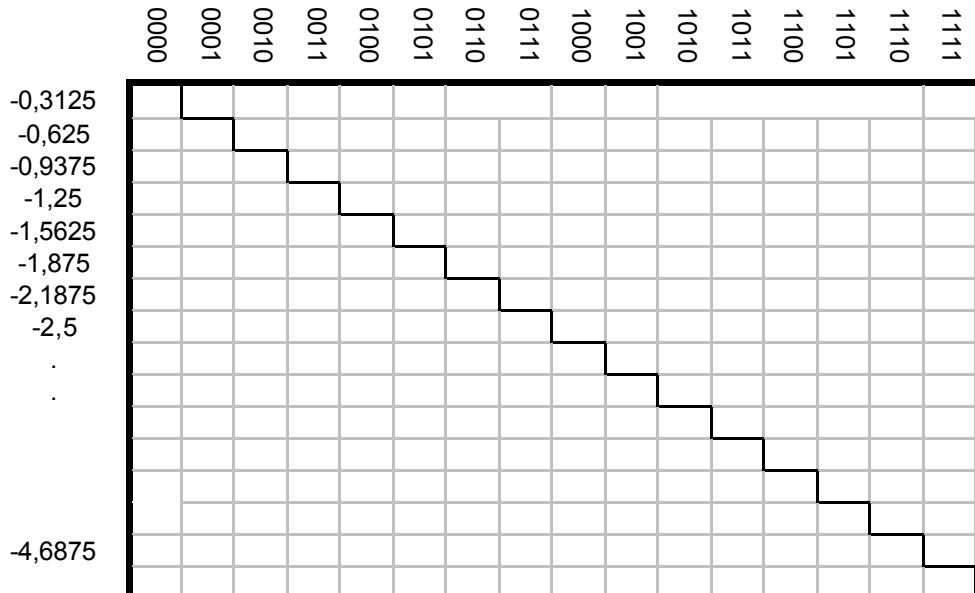
$$I_0 = \frac{5V}{160K\Omega} = 0,03125mA \quad I_1 = \frac{5V}{80K\Omega} = 0,0625mA \quad I_2 = 0,125 mA$$

$$I_3 = 0,25 mA$$

AOaren sarrera-alderanzgailutik ia korronterik ez doanez, bere impedantzia altua dela eta, korronte guztia berrelkadurako erresistentziatik ( $R_f$ ) doa, horregatik:

$$V_{out(D0)} = (10k\Omega)(-0,03125mA) = -0,3125V, \quad V_{out(D1)} = -0,625V,$$

$$V_{out(D2)} = -1,25V, \quad V_{out(D3)} = -2,5V$$



## 7.4. BIHURGAILU DIGITAL/ANALOGIKOEN EZAUGARRIAK

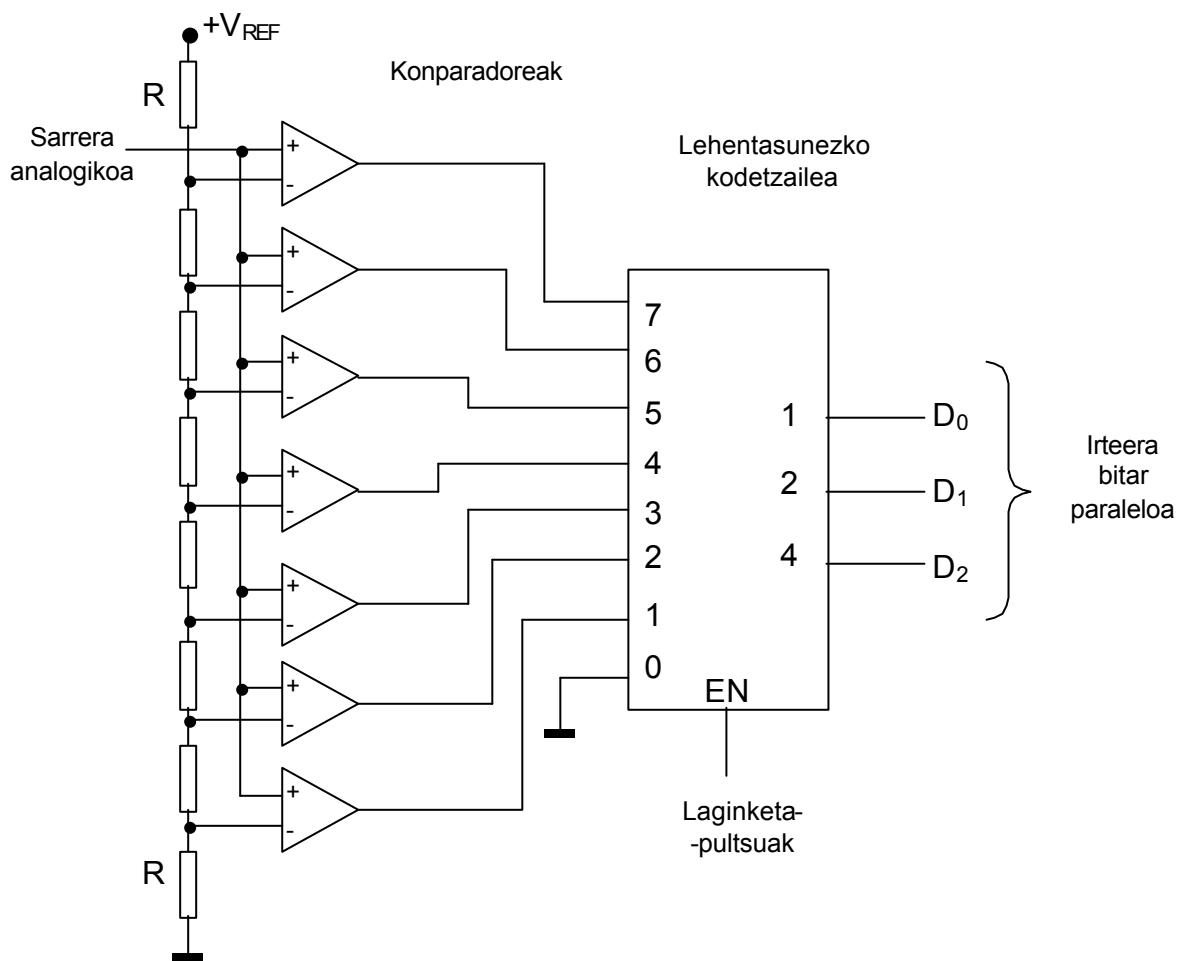
- **Bereizmena.** Sarrerako bit-kopuruaren araberakoa da. Zenbat eta bit gehiago, orduan eta bereizmen handiagoa. 4 bitekoak 1etik 15erako bereizmena du, baina 10 bitekoak 1etik 1.024rakoa izango du. Zenbat eta bereizmen handiagoa izan, ematen duen seinale analogikoak dituen jauziak (mailak) orduan eta txikiagoak dira. Jauzi diskretuen kopurua  $2^n - 1$  da, n bit-kopurua izanik. Bereizmena bihurtzen diren bit-kopuru bezala adieraz dezakegu.
- **Doitasuna.** DACaren irteera idealaren eta errealaaren arteko aldea da. Irteerako tentsio maximoaren ehuneko gisa ematen da. Adibidez, bihurgailu baten irteera maximoa 10 V-ekoa eta doitasuna  $\pm 0,1$ ekoa badira, errore handiena (edozein tentsiotarako)  $10 \cdot 0,001 = 10$  mV-ekoa izango da. Oro har,  $\pm \frac{1}{2}$ LSB izan behar luke. Hau da, 8 biteko bihurgailuarentzat:  $1/256 = 0,39 \rightarrow$  doitasunak  $\pm 0,2$  izan beharko luke.
- **Ezарpen-denbora.** Sarreran kode-aldeketa gertatu ondoren, irteerako azkeneko balioaren  $\pm \frac{1}{2}$ LSB barruan geratzeko, DACak behar duen denbora da.

## 7.5. BIHURGAILU ANALOGIKO/DIGITALAK

Magnitude analogikoa digital bihurtzen duen bihurgailu-mota asko dago:

### 7.5.1. KONPARADOREDUNA (FLASH)

Sarrerako tentsio analogikoa konparadore batzuen erreferentziazko tentsioaz konparatzen da. 7.10. irudikoa hiru bitekoa denez, 7 konparadore dauzka. Bihurgailu honek, ordea, badu arazo bat: konparadore-kopuru handia behar du. Oro har,  $n$  biteko kode bitarrera bihurtzeko  $2^n - 1$  konparadore behar dira.

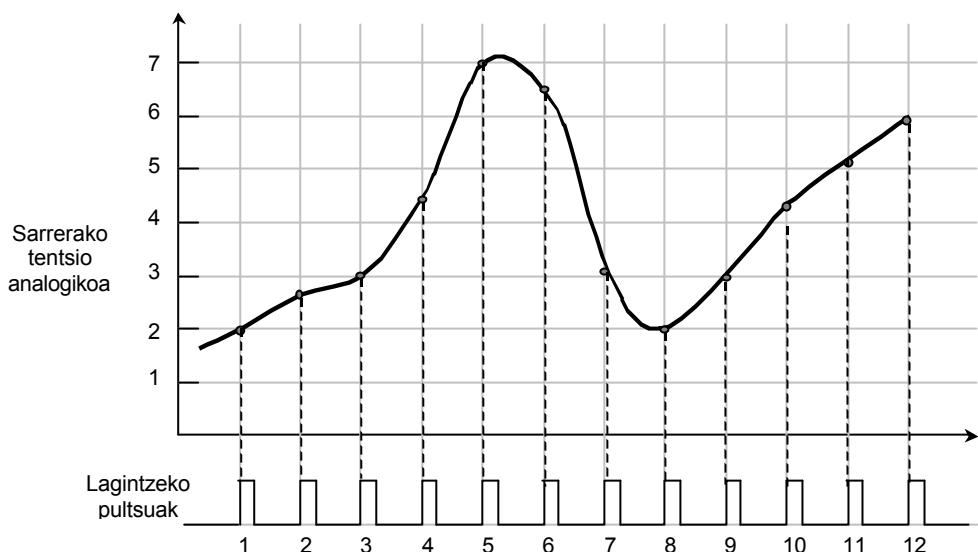


7.10. irudia

Konparadore bakoitzaren erreferentzia tentsio-zatigailuaz lortzen da eta irteerak lehentasunezko kodetzailearen sarrerekkin konektatzen dira. Gaitze-sarreran (EN) pultsuak emanez lagintzen da kodetzailea, eta pisu handiena duen sarrera aktiboak ezartzen du irteerako kodea.

Lagintze-abiadurak ematen du ADCaren doitasun-maila. Zenbat eta abiadura handiagoa izan, orduan eta analogikoaren itxura handiagoa izango du seinale digitalak.

Aurki ezazu hiru bitetako flash-erako ADCak irteeran emango duen kode bitarra; lagintzeko pultsuak 7.11. irudian adierazten dira.  $V_{REF} = 7 \text{ V}$ .

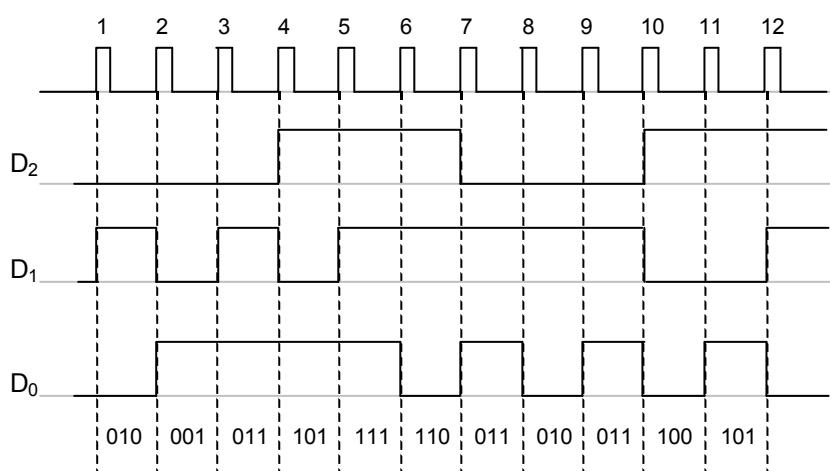


7.11. irudia

Irteeran lortuko dugun sekuentzia digitala honakoa izango da:

010, 001, 011, 101, 111, 110, 011, 010, 011, 100, 101, 110

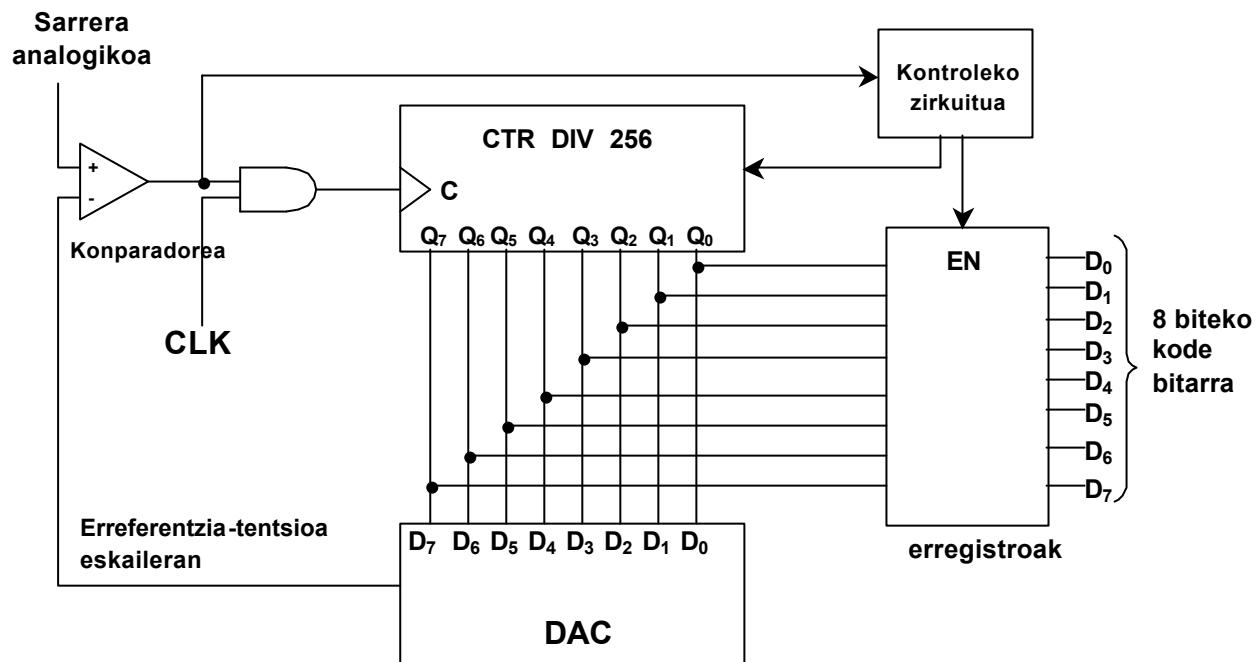
Eta pultsuen araberako kronograma 7.12. irudikoa litzateke.



7.12. irudia

### 7.5.2. ERREFERENTZIA-TENTSIOA ESKAILERAN DUENA EDO KONTAGAILUDUNA

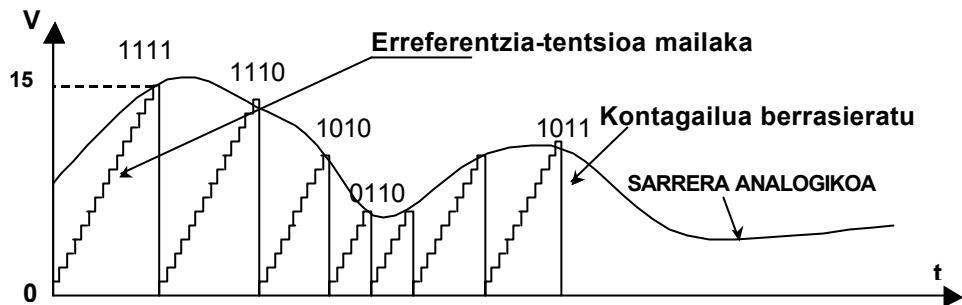
Metodo honetan DACa eta kontagailu bitarra erabiltzen dira sarrera analogikoaren baliokide digitala lortzeko. 7.13. irudian agertzen da diagrama:



7.13. irudia

Hasieran kontagailua berrasieratua eta DACaren irteera 0V dela suposatzen dugu. Sarrera analogikoa ezartzean, eta horrek erreferentzia-tentsioa gainditzen duenez, konparadoreak maila altua ematen du eta AND ateak erlojuaren pultsuei kontagailura pasatzen uzten die. Kontagailua aurrera doan heinean, DACaren erreferentzia-tentsioa mailaka handituz joango da. Tentsio horrek sarrera analogikoa gainditzen duenean, konparadoreak maila baxua ematen du eta kontagailua geratu egiten da. Kontagailuak duen eduki bitarra da sarreraren baliokide digitala. Kontroleko logikak zenbaki bitar hori erregistroetan kargatzen du eta kontagailua berrasieratzen du; eta, berriz, beste sekuentzia bat hasten da sarrerako balioa lagintzeko.

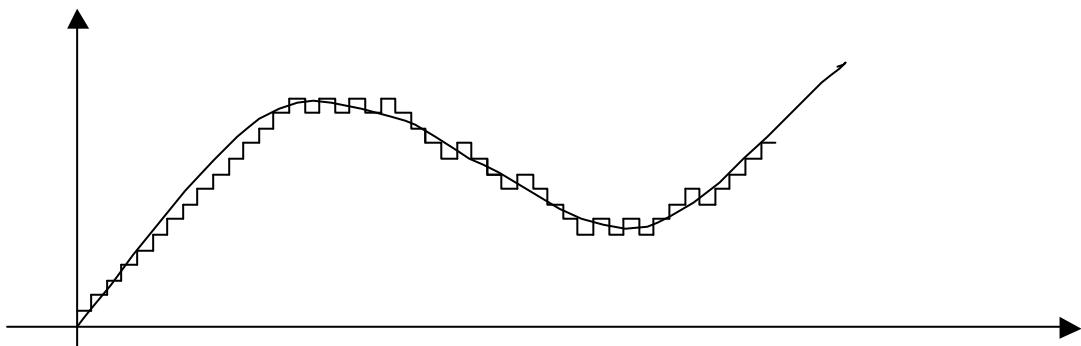
Bihurgailu hau aurrekoan baino geldoagoa da, sarrera maximoaz (kasurik txarrena) kontagailuak egoera guztietatik pasatu behar baitu laginketa egiteko (8 bit → 256 egoera). Laginketa bakoitzean kontagailua 0tik abiatzen da. Ikus 7.14. irudia.



7.14. irudia

### 7.5.3. JARRAIPEN BIDEZKOA

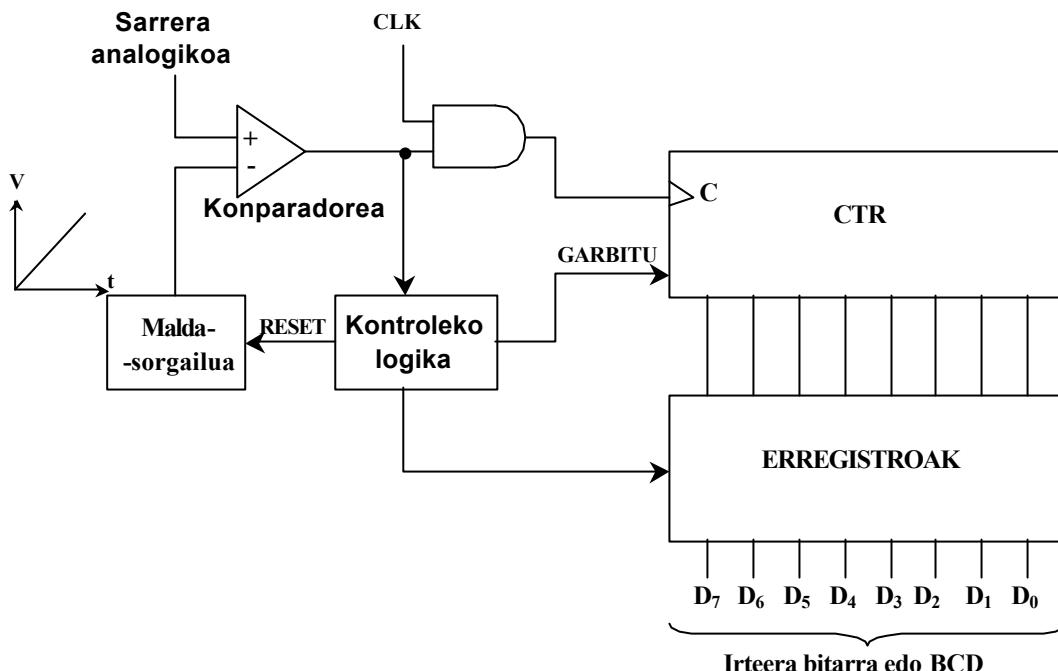
Metodo hau aurrekoaren antzekoa da. Kontagailua *up/down* erakoa da. Kontagailua ez da 0tik abiatzen laginketa bakoitzean, kontatu egiten bai sarrera analogikoa erreferentziako baino handiagoa bada, eta deskontatu sarrera txikiagoa bada. Horrela, kontagailuak sarreraren balioari ‘jarraitu’ egiten dio. Ikus 7.15. irudia.



7.15. irudia

### 7.5.4. MALDA BAKUNEKO BIHURGAILU ANALOGIKO/DIGITALA

Metodo honean ez dugu DACrik behar. Maldakonstantea duen erreferentzia-tentsioa erabiltzen dugu. Eskema 7.16. irudian ikus daiteke.



7.16. irudia

Hasieran, kontagailua 0n dago eta maldak 0V du, sarrera analogikoa erreferentzia baino handiagoa da; ondorioz, konparadoreak maila altua ematen du. Maila altu horrek erloju-seinalea gaitzen du kontagailuarentzat eta maldasorgailua abiarazten du.

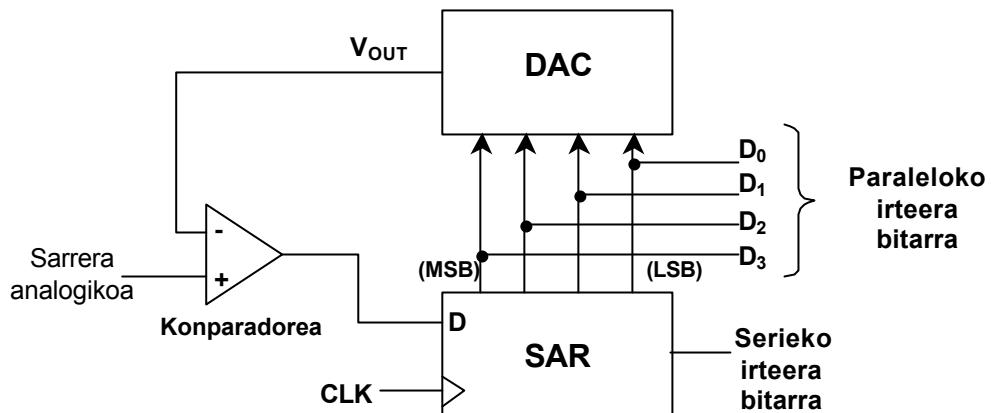
Sarrera maldaren berdina izatera heltzen denean, kontroleko logikak maldar 0an jartzen du, eta kontaketa bitarra erregistroetan metatzet.

Metodo hau erabiltzen dute zenbait polimetro digitalek.

Metodo honen itxurakoa da **malda bikoitzekoa**, baina doitasun handiagoa lortzen du.

### 7.5.5. ONDOZ ONDOKO HURBILKETA BIDEZKOA

Gehien erabiltzen den metodoa da, besteek baino askoz bihurketa-denbora txikiagoa duelako. Denbora hori finkoa da sarrerako edozein baliotarako.

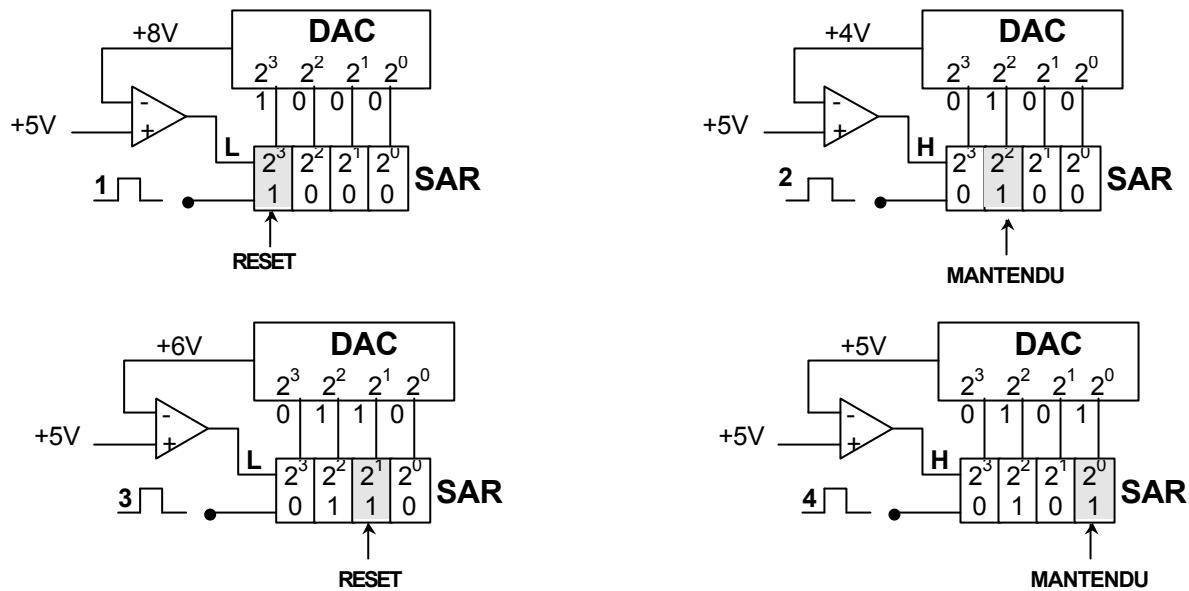


7.17. irudia

7.17. irudian 4 bitekoak ikusten da. DACak, ondoz ondoko hurbilketen erregistroak (SAR, *Succesive-Approximation-Register*) eta konparadore batek osatzen dute. Honela funtzionatzen du:

DACean bitak banan-banan aktibatzen dira MSBtik hasita, bit bat aktibatzen den bakoitzean, konparadoreak sarrera analogikoa eta DACaren irteera konparatzen ditu. DACaren irteera sarrera analogikoa baino handiagoa bada, maila baxua ematen du eta erregistroan bit hori "0"an ipintzen du. Irteera sarrera analogikoa baino txikiagoa bada, erregistroan bit hori "1" balioan mantentzen da. Sistemak MSBtik hasi eta bit guztiekin errepikatzen du eragiketa, bihurketa-zikloa osatuz. Ziklo hori beti iraupen berekoa da. Sarrera analogikoaren balioak ez du zerikusirik.

Adibidez, 4 biteko bihurketa-prozesua pausoaz aztertuko dugu. Sarreran +5V-eko tentsioa aplikatzen diogu eta DACaren irteera-ezaugarriak honakoak dira: MSB bitarentzat ( $2^3$ )  $V_{out} = 8V$ ,  $2^2$  bitarentzat  $V_{out} = 4V$ ,  $2^1$  bitarentzat  $V_{out} = 2V$  eta  $V_{out} = 1V$   $2^0$  bitarentzat.



7.18. irudia

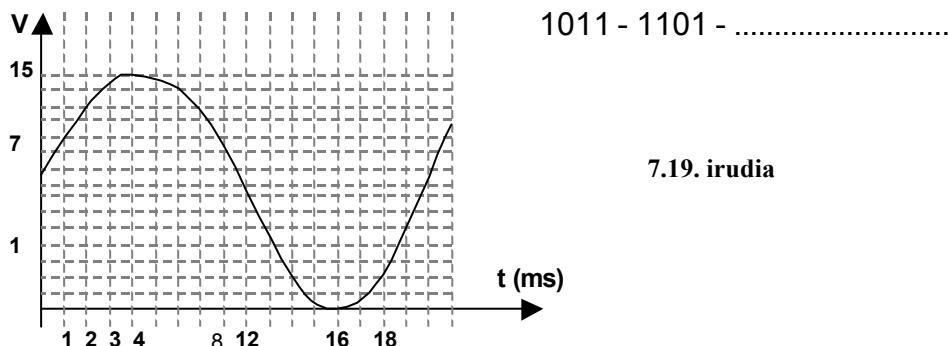
Irudian azter daitezke lau bitak. Bihurketa-zikloak erlojuaren 4 pultsu behar ditu. Ziklo bakoitza hastean SARa ezabatu egiten da (urretik bere edukia erregistroetara pasatzen da).

## Galderak

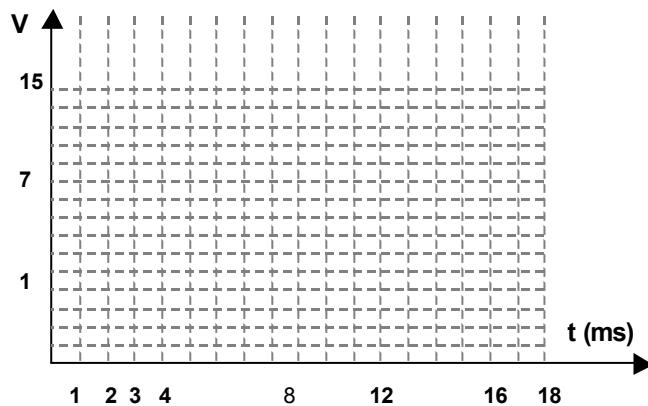
- 7.1. Erresistentzia haztatuak dituen DAC bitarrean, sarrerako erresistentziek:
  - Seinale analogikoaren anplitudea mugatzen dute
  - Sarrera digitalen pisuak finkatzen dituzte
  - Potentzia-kontsumoa mugatzen dute
- 7.2. Erresistentzia haztatuak dituen DAC bitarrean, erresistentzia bakoitzaren balioa:
  - Eta gainerakoena berdinak dira
  - Berrelkadura-erresistentziaren erdia da
  - Balio txikienaren bikoitza da
  - Balio txikienaren erdia da
- 7.3. R/2R batean:
  - Lau erresistentzia-balio ezberdin daude
  - Erresistentzia-balio bat dago
  - Erresistentzia-balio bi daude
  - Erresistentzia-balio kopurua eta sarrera-kopurua berdinak dira
- 7.4. 8 biteko DAC baten bereizmena:
  - % 0,1ekoa da
  - % 0,392koa da
  - % 1ekoa da
  - % 3,92koa da
- 7.5. 4 biteko DACaren doitasun idealetan gutxi gora-behera:
  - % 3,1ekoa da
  - % 0,31ekoa da
  - % 6,2koa da
  - % 0,62koa da
- 7.6. Bihurketa-denbora luzeena duen ADCa:
  - Flash-erakoa da
  - Malda digitalekoa da
  - Jarraipen bidezkoa da

**Ariketak**

- 7.1. Kalkula ezazu 12 biteko D/A bihurgailuaren doitasuna.
- 7.2. Kalkula ezazu 4, 8 eta 10 biteko D/A bihurgailuen LSB bitaren pisu bitarra.
- 7.3. Kalkula ezazu aurreko 7.1. ariketako DACaren errore maximoa.
- 7.4. Erresistentzia hartzatuak dituen 10 biteko D/A bihurgailuan, LSBtik pasatzen den korrontea 200 mA-koa da. Kalkulatu MSBtik pasatzen den korrontea.
- 7.5. 7.19. irudiko kurba analogikoa 1 ms-tik behin lagentzen da. Adieraz ezazu kurba osoa 4 biteko zenbaki bitarrez.



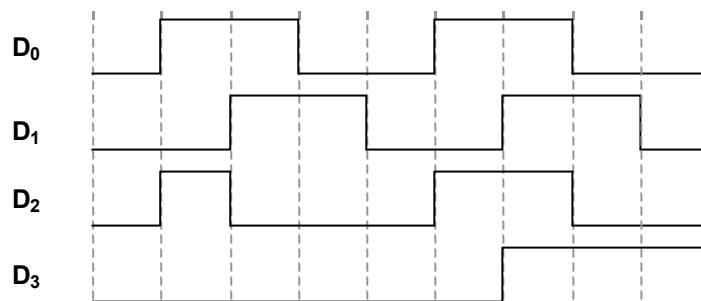
- 7.6. Marraz ezazu aurreko ariketaren irudikapen digitala.



**7.7. Irudika ezazu grafiko batean, hurrengo zenbaki bitarren sekuentziak adierazitako funtzio analogikoa.**

1111, 1110, 1100, 1010, 1001, 1000, 0111, 0110, 0101, 0100, 0101,  
0110, 0111, 1000, 1001, 1010, 1011, 1100, 1100, 1100, 1011, 1010, 1001.

**7.8. Kalkula ezazu 7.7. irudiko DACaren irteera, sarrerei 7.20. irudiko grafikoan adierazten diren sekuentziak aplikatuta.**



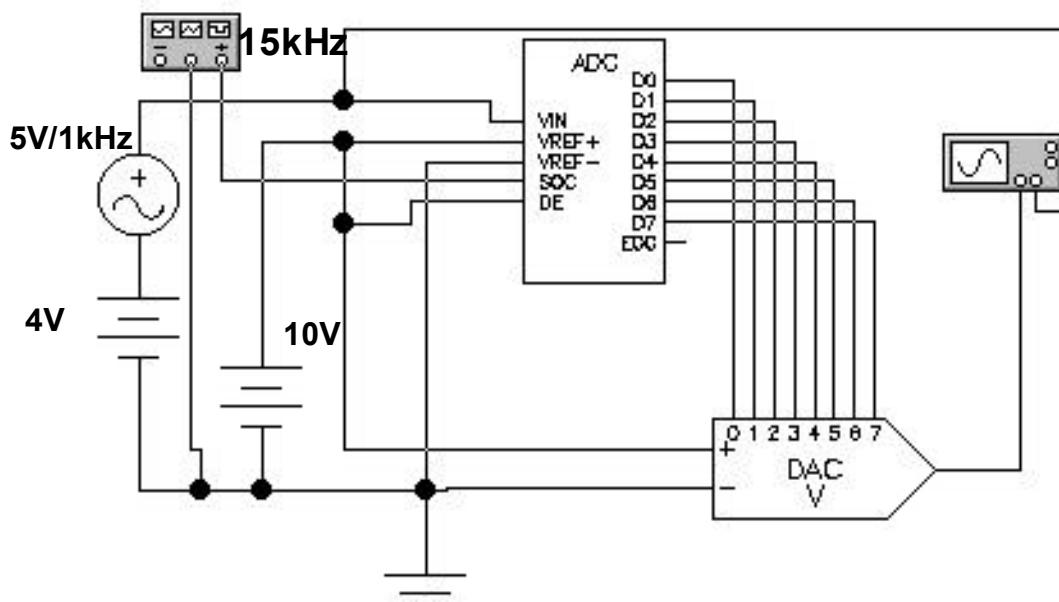
7.20. irudia

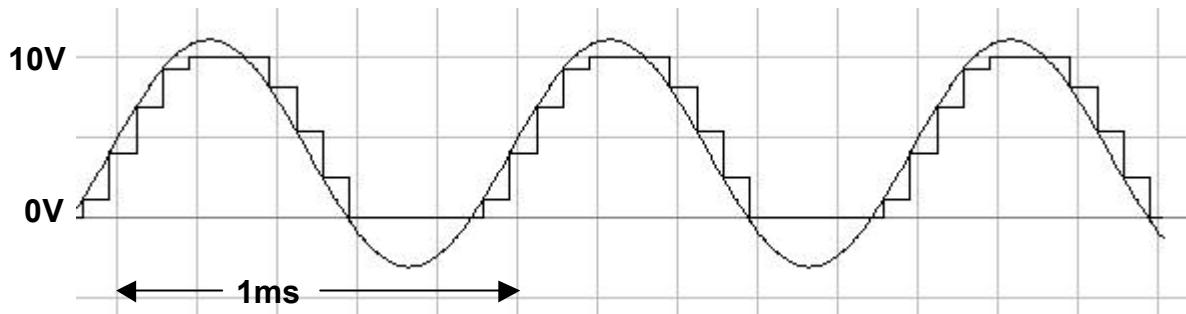
## Simulazio praktikoak EWB softwarea erabiliz

**7.A)** Zirkuitu honetan, balio efikaza 5 V-ekoa eta maiztasuna 1 kHz-ekoa duen seinale analogiko sinusoidala digitalizatu egiten da ADC batez; gero DAC bat erabiliz, berriz analogiko bihurtzen da.

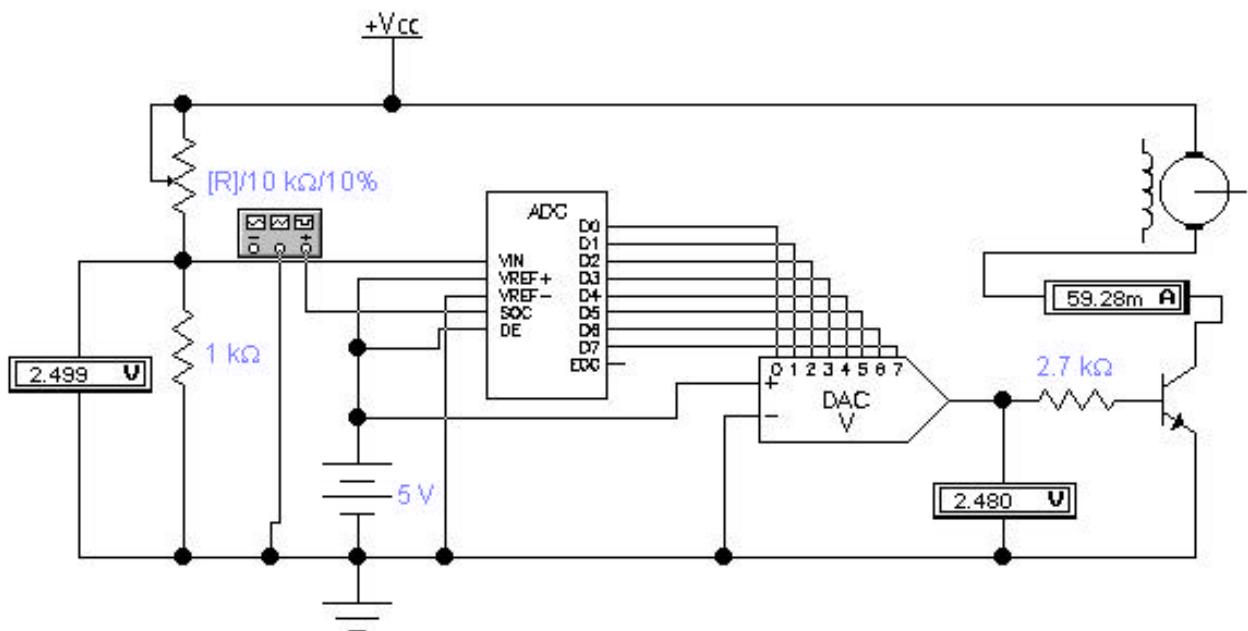
*ADC bihurgailuaren sarrerak:*

- ❖ **V<sub>IN</sub>**: digitalizatu nahi dugun seinale analogikoa aplikatzen da
- ❖ **V<sub>REF+</sub> eta V<sub>REF-</sub>**: zein balioren artean digitalizatu nahi dugun. V<sub>REF+</sub> balioa emango du bit guztiak “1” direnean eta V<sub>REF-</sub> balioa bit guztiak “0” direnean.
- ❖ **SOC (Start of conversion)**: erloju-seinalea aplikatuko dugu bihurketa bakoitza (*Sample & hold*) egingo den maiztasunaz. Gure kasuan F = 15 kHz; horrek esan nahi du bihurketa bakoitza t = 66 µs edo gutxiagoan burutu behar duela ADC123ak.
- ❖ **OE (Output enable)**: maila altuan egon behar du. Irteerak ahalbidetzen ditu.
- ❖ **EOC (End of conversion)**: bihurketa noiz bukatzen den adierazten duen pultsua ematen du.
- ❖ **D<sub>0</sub> ... D<sub>7</sub>**: irteera digitalak (bit)



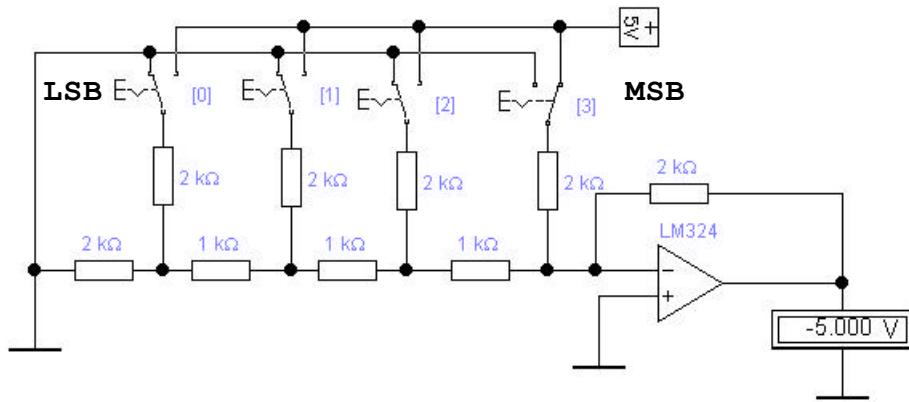


**7.B)** ALECOP 2000 PLAKAREKIN EGIN DAITEKEEN PRAKTIKA. LDRz, edo erresistentzia aldakor batez, lortzen dugun tentsio aldakorra ADCari ezartzen diogu. SOC (Start) bezala uhin-sorgailua erabili. ADCak ematen duen seinale digitala DACari ezarri eta, horren irteeraz, KZ-ko motorra elikatu.



**7.C)** R/2R eskaileran dituen bihurgailu digital analogikoa. Osa ezazu ondoko taula, zure ustez lortuko diren balioekin; gero EWB erabiliz, egiaztatu.

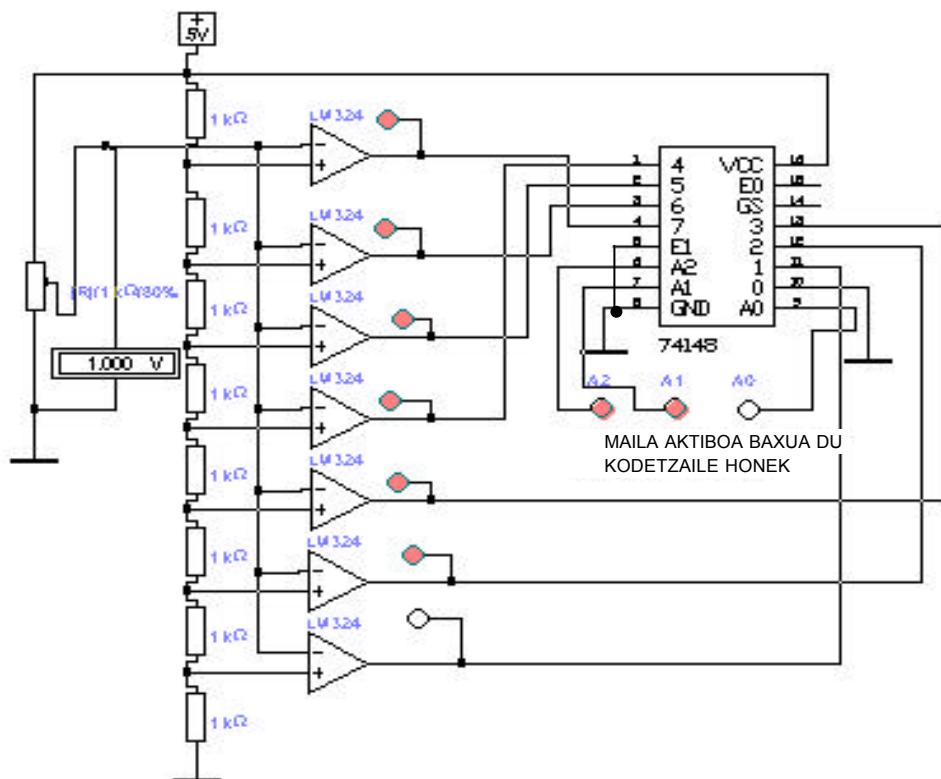
0010	0100	0001	1000	0011	0111	
------	------	------	------	------	------	--



#### 7.D) Flash-erako bihurgailu analogiko digitala (maila aktiboa = baxua).

Kalkula ezazu horren irteera hurrengo taula osatuz. Ondoren egiaztatu, EWB erabiliz

Sarrera (V)	1	1'5	2'3	2'6	3'2	3'7	4	4'7
Irteera								



# 8. KAPITULUA

## Memoriak

### 8.1. OINARRIZKO JAKINGARRIAK

Gaur egun sistema dittaletan erabiltzen diren memoriak LSI zirkuitu integratuak dira. Gailu hori funtsezko pieza da osagai programagarriez eraturiko zirkuituetan.

Informazioa modu bitarrean, gordetzeko edukiera duten gailu digital oinarrizkoenak biegonkorak eta erregistroak dira. Lehenengoak bigarrenak baino simpleagoak dira.

Gorde daitekeen informazio-unitaterik txikiena bita da, eta horretarako nahikoa da bere garaian aipaturiko edozein biegonkor-motaz baliatzea. Hainbat bit (gehienetan 4 edo 8) metatu nahi direnean, ordea, erregistro bat behar da. Eta esanda dagoen moduan, erregistroa MSI zirkuitu integratu bat da, gorde beharreko bit adina biegonkor dituena.

Memoriak sarrera eta irteera komunak dituzten hainbat erregistro bilduz, informazio-kantitate handiak gordetzeko edukiera duten zirkuitu integratuak dira. Memoriako erregistro horietako bakoitzari posizio deitzen zaio, eta, gehienetan, bit bat, 4 edo 8 metatzeko beharrezkoak diren osagaiez osatzen da. Posizio bakoitzarentzat helbide jakin bat izaten da, zenbaki bidez adierazten dena. Eta posizio jakin batera jo nahi bada, dagokion helbidea aktibatu behar da.

Mikroprozesadorea oinarritzat hartzen duten sistema digitalek, gehienbat, 8 biteko multzoekin lan egiten dute, eta gaur egun erabiltzen diren memoriak 8 biteko posizio helbideragarriz osatzen dira. 8 biteko multzo horri byte (zortzikote) deitzen zaio.

Memorietan datuak gorde edo metatu egin daitezke. Eragiketa horri **idazketa** deritzo. Aurrez posizio batean idatzi denaren balioa lortzea ere egin daitekeena da. Eragiketa horri **irakurketa** deitzen zaio. Memoriako irakurketa eta idazketa ia beti mikroprozesadoreak egiten du, helbide-busaren bidez, nahi den posizioa hautatu, eta idatzi edo irakurri nahi den informazio hori datu-busaren bidez eramanez.

## 8.2. MEMORIEN EZAUGARRI NAGUSIAK

---

Elektronika integratua garatu aurreko garaietan, beste osagai batzuk erabili izan dira informazioa metatzeko. Ferritazko nukleoaz eginko memoriek betebehar garrantzitsua izan dute lehenengo belaunaldietako ordenagailuetan.

Informazio digitala metatzeko erabili diren sistema guztietai asmoa beti berbera izan da: hura osatzen duten erregistro guztietaira iristeko denbora laburrena lortzeko gailuak asmatzea, datuak metatzeko edukiera handia edukitzea, kontsumo txikia izatea eta merkea izatea.

LSI zirkuituekin beste edozein teknologiarekin baino askoz etekin hobeak ateratzen dira. Hargatik, beste sistema guztiak baztertu egin dira gaur egun.

Memorien ezaugarri nagusiak ondokoak dira:

*Idazketa/irakurketa denbora.*

*Transferentziaren abiadura.*

*Informazio-dentsitatea.*

*Ezabaketa.*

*Edukiera.*

**Idazketa-denbora** da informazioa sarreran agertzen denetik grabatu arte egiten duen denbora. **Irakurketa-denbora**, berriz, irakurtzeko ematen zaionetik irteeran agertzen den bitartekoak.

**Abiadura** goiko denbora horiekin erlazionatuta dago, izan ere, memoriak idazteko edo irakurtzeko informazioa bidali edo jasotzen duen abiadura da.

**Informazio-dentsitatea** integrazio-escalarekin dago erlazionatuta. Espazio fisikoaren unitateko meta dezakeen datu-kopurua da.

**Ezabaketa** metaturiko datuak galtzeko aukera da, gailua osatzen duten osagaien ezaugarriengatik zein elikadura elektrikoaren etenagatik.

**Edukiera** azaltzeak xehetasun gehiago eskatzen du. Hargatik, hurrengo atal osoa eskainiko diogu arlo horri.

### 8.3. MEMORIA BATEN EDUKIERA

Memorien ezaugarri nagusia hura osatzen duten posizioen kopurua izaten da. Lehenengo eta behin, memoria baten edukiera osoa eta erabilgarri dauden posizio-kopuru osoa bereizi behar ditugu:

Edukiera osoa deitzen zaio memoria batek meta dezakeen bit-kopuru osoari. Beraz, beti izango da handiagoa edukiera osoa posizio-kopurua baino, posizio bakoitzeko bit-kopurua bat denean salbu. Hala denean, biak gauza bera dira eta.

Oro har, posizio bakoitzeko bit-kopuruari **n** deitzen badiogu eta posizio-kopuruari **m**, bit-kopuru osoa, **N** izenekoa, honako hau izango da:

$$N = n \cdot m$$

Memoriaren posizio jakin bat hautatzeko eragiketari helbideratzea deitzen zaio. Helbide-busa edo, beste modu batera esanda, memoria baten **m** posizioak helbideratzeko beharrezkoa den lerro-kopurua **n<sub>1</sub>** izango da, baina betiere ondoko berdintza bete beharko da:

$$2^{n_1} = m$$

Memoria bakoitzak daukan posizio-kopurua eta posizio bakoitzean meta daitekeen datu edo hitzaren luzera izaten da memoriaren bereizgarria. Hala, 32 x 4 adierazpenak, esate baterako, 32 posizio ( $m = 32$ ) eta 4 biteko hitz-luzera ( $n = 4$ ) dituen memoria dugula esan nahi du.

Edukiera handiko memorien posizio-kopurua **k**-tan (kiloaren laburdura) neurten da. **k** batek 1.024 posizio balio ditu, eta horiek guztiak helbideratzeko hamar lerro edo aldagai bitar behar dira, izan ere:

$$2^{10} = 1.024$$

**k** ikurra memoria baten edukieraren laburduratzat darabilgunean, k horrek beste magnitude fisiko batzuetarako daukan esanahi berbera duka. Baina  $1.024 = 2^{10}$  hautatu bada, 1.000 zenbakira gehien hurbiltzen den 2ren berretura delako izan da.

Hitzaren luzera 8 bitekoa denean, hots, byte bat denean, edukiera edo posizio-kopurua kilobytetan neurten da.

Bere posizio guztiak helbideratzeko,  $n_1$  aldagai bitar darabiltzan memoriaren edukiera, **k**-tan adierazita, honako hau izango da:

$$2^{n_1 \cdot 10}$$

Beraz,  $n_1 = 14$  hautaketa-sarrera dituen memoriak 16 k posizioko edukiera edukiko du.

Azkenik, edukiera osoaren gaira itzuliz, 16 kilobyteko (hau da, 16 k x 8) memoriaren bit-kopurua aterako dugu, era orokorrago batean adierazita:

$$N = 16 \times 1.024 \times 8 = 131.072 \text{ bit}$$

## 8.4. MEMORIA-MOTAK

MEMORIAK						
Zuzeneko atzipena				Atzipen sekuentziala		
Irakurketa/idazketa RAM		Irakurketa bakarrik ROM		FIFO LIFO	Desplazamendu- -erregistroak	CCD
Estatikoak SRAM	Dinamikoak DRAM	Programariak fabrikazio-prozesuan	Programariak erabiltzaileentzat PROM EPROM EEPROM			
TTL/MOS	MOS	TTL/MOS	TTL/MOS	TTL/MOS	TTL/MOS	MOS

8.1. taula. Memoria-motak.

Atzipen aleatoriodeun memorien taldean, batetik, irakurketa eta idazketa onartzen dituztenak eta, bestetik, irakurri besterik ez daitezkeenak daude. Dena dela, memoria hitza darabilgunean, gehienetan, lehenengoei buruz ari gara, horien izena RAM den arren. Bigarrenei ROM deitzen zaie, *Read Only Memory*-ren laburdura ("irakurtzeko bakarrik den memoria").

Orobat, ROM memoriak azpitalde hauetan banatzen dira:

**ROM:** fabrikazio-prozesuan maskaraz programaturiko memoriak dira.

**PROM:** sistema diseinatzeko orduan erabiltzaileak programa ditzakeenak dira. Behin bakarrik programa daitezke.

**EPROM:** honakoak berriro ere programa daitezke. Izpi ultramorez ezabatzen dira.

**EEPROM:** elektrikoki ezabagarriak eta birprogramagarriak dira. ROM memoriak ez dira ezabagarriak, hau da, ez dute informazioa galtzen elikadurako tentsioa deskonektatu arren. Gailu hau sistemaren kontrol-programa gordetzeko uzten da.

RAM memoriak, edo irakurketa/idazketakoak, ezabagarriak dira. Normalean, prozesuaren tarteko edo amaierako emaitzak gordetzeko erabiltzen dira, baita kanpoko gailuetatik (disko edo zinta magnetikoetatik) datozen programak gordetzeko ere.

RAM memoriak estatiko eta dinamikoetan banatzen dira. Estatikoek elikadura elektrikoak diraen artean gordetzen dute informazioa. Dinamikoek (MOS teknologia bakarrik darabiltenak eta aurrekoek baino erregistro-kopuru handiagoa dutenak) zirkuituko seinale ‘freskagarri’ bat behar dute informazioa gordetzeko, programa nagusiak kontrolatzen duena.

Atzipen sekuentziala duten memorietako posizio jakin batean idatzi edo irakurtzeko, haren aurrean dauden posizio guztiak irakurri behar dira. Hargatik, posizio batera sartzeko behar den denbora posizio horrek gailuaren barne-egituraren daukan kokapenaren araberakoa izango da.

Talde horretakoak dira FIFO/LIFO memoriak, CCD akoplamenduzko kargakoak eta desplazamenduzko erregistroak.

Bai FIFO (*First In First Out*) eta bai LIFO (*Last In First Out*) memoriak informazioa metatzeko eta ateratzeko moduagatik deitzen dira horrela. FIFOetan, lehenengo sartzen den datua da lehenengo irteten dena ere. LIFOetan, ordea, azkena sartzen den informazioa da lehenengo irteten dena. FIFOak itxarote-ilara bat dira eta LIFOak, berriz, datu-metak.

Desplazamenduzko erregistrodun atzipen sekuentzialeko memoriak, berriz, erregistro batez eta zirkulazio-sare batez osatzen dira. Azken hau ate logikoz eratuta dago, barruko datu batera sartzean informaziorik gal ez dadin.

Eta azkenik, akoplamenduzko kargako metatze-gailuek ere aurrekoek bezalaxe funtzionatzen dute. Ezberdintasun bakarra hau da: memoria horietako erregistroa CCD (*Charge Coupled Device*) osagaiz eratuta dagoela. Osagai horiek siliziozko substratu baten azalean karga elektrikoa metatzeko edukiera daukate eta desplazatu egin daiteke azal horren gainean dauden elektrooen potenzial elektrikoa aldatzean.

8.1. taulan ikus daitekeen bezala, aipaturiko memoria gehienak MOS eta TTL teknologietan egiten dira. TTL delakoa teknologia bipolar izenez ere ezagutzen da.

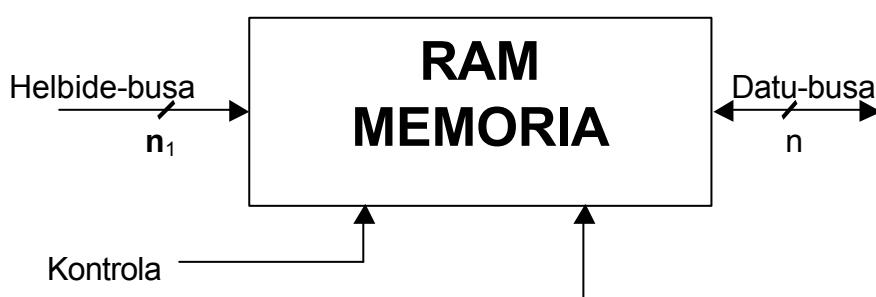
**FLASH** memoriak: dentsitate handiko eta hegazkorrik ez diren irakurketa/idazketa memoriak dira. Memoria horietan datuak denbora askoan meta daitezke elikatze-iturririk gabe. Dentsitate handi hori (azalera txikian gelaxka-kopuru handia) metatze-gelaxkak MOS transistore bakar batez osatzen direlako lortzen da.

MEMORIA-MOTA	HEGAZKORRA	DENTSITATE HANDIA	TRANSISTORE BAKARREKO GELAXKA	EZABAKETA
FLASH	EZ	BAI	BAI	BAI
SRAM	BAI	EZ	EZ	BAI
DRAM	BAI	BAI	BAI	BAI
ROM	EZ	BAI	BAI	EZ
EPROM	EZ	BAI	BAI	EZ
EEPROM	EZ	EZ	EZ	BAI

8.2. taula. Memoria batzuen arteko konparaketa.

## 8.5. MEMORIA BATEN KANPOKO EGITURA

8.1. irudian erakusten da RAM memoria baten irudi sinbolikoa bere helbide, datu eta kontrolen sarrera-irteerekin. Honen eta ROM baten arteko ezberdintasuna hau da: ROMak ez du R/W kontrol-sarrerarik. Sarrera horrexek ematen du, ezarritako maila logikoaren funtzioan, irakurtzeko eta idazteko aukera.



8.1. irudia. RAM memoriaren irudikapen sinbolikoa bere sarrera-irteerekin.

Gailu honek datu-sarrera eta -irteera komunak dauzka.  $n$  letraren balioa posizio bakoitzean idatz edo irakur daitezkeen hitzen luzeraren araberakoa da. Memoria batzuetan sarrerak eta irteerak bereizita daude.

Erregistro bakoitzaren helbideratzeko hautaketa-lerroak  $n_1$  ikurrak adierazten ditu. Horien balioa memoriako posizio-kopuruaren araberakoa da.

Gailu honen kontroleko sarrerak CS (*Chip Select*) eta lehen aipaturiko R/W (*Read/Write*) dira. Zenbait zirkuitugilek beste letra batzuk erabiltzen ditu terminal horiek adierazteko, baina, edonola ere, horien eginkizuna beti berdina da. Batzueta kontroleko bi sarrera baino gehiago egoten dira.

CS hankatxoa txip jakin bat hautatzeko erabiltzen da, memoria hainbat zirkuitu integratuz osatzen denean. Sarrera horretan maila logiko jakin bat ezarriz, behar den zirkuitua hautatzen da, eta idatzi eta irakurri egin daiteke bertan (R/W hankatxoari ezarri zaion balio logikoaren arabera). Kontrako balioa ezartzen denean, zirkuitua desgaituta geratzen da. Bistakoa denez, txip bat hautatu denean, memoria eratzen duten gainerako guztiak desgaituta utzi behar dira.

## 8.6. MEMORIA BATEN BARNE-ANTOLAMENDUA

---

LSI teknologiako memoria-zirkuituak, funtsean, honako osagai hauez eratzen dira:

Matrizea, bit bana meta dezaketen hainbat zelulak osatzen dutena.

Deskodetzaile bat edo bi, matrizearen posizioetako bakoitza hautatzeko.

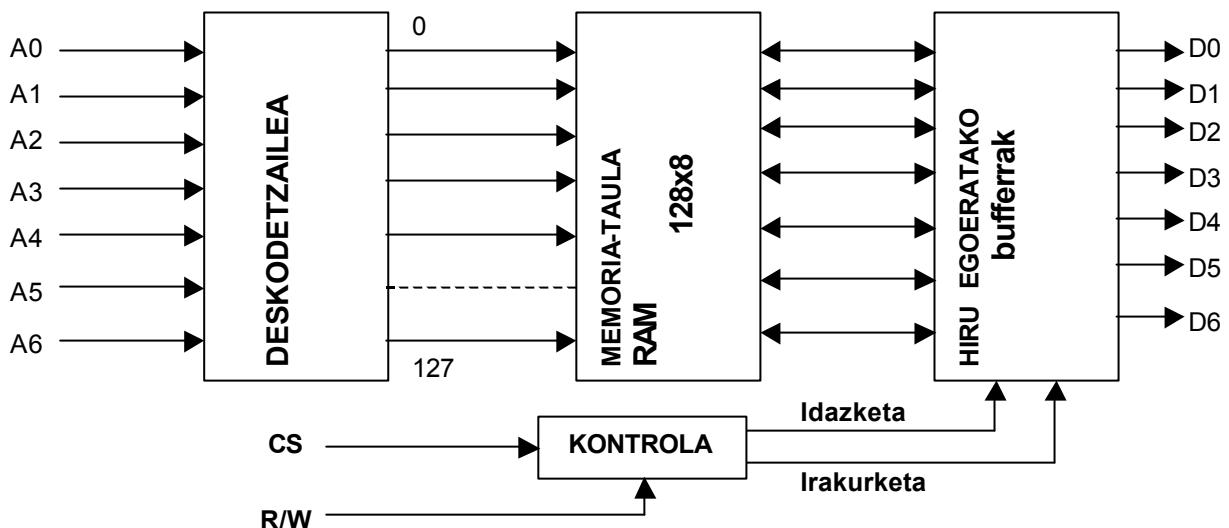
Buffer-multzo bat, hiru egoeratako atez eratua, eta ‘kontroleko zirkuitu baten bidez’ datuen sarrera eta irteera gobernatzen duena. Zirkuitu horri ezartzen zaizkio irakurketa-idazketa (R/W) eta txip-hautaketarako (CS) seinaleak.

Hiru osagai horiez baliatuta, hainbat modutan era daitezke barrutik memoriak. Erarik errazenari hautaketa lineala deitzen zaio. Halakoetan, matrizea errenkada eta zutabeka antolatzen da. Errenkada bakoitza eratzen duten gelaxken kopurua hitz bakoitzeko bit-kopuruari dagokiona da, eta errenkada-kopurua hitz- edo posizio-kopuruari dagokiona. Zutabe-kopurua, berriz, hitza osatzen duten bit-kopuruaren parekoa izango da.

Gailuak **m** posizio badauzka, deskodetzaileak **m** irteera eduki beharko ditu.

Eraketa hori apropos-aproposa da edukiera urriko memorietarako, zeren irakurketan edo idazketan sartzeko denbora oso laburra izaten baita. Posizio askotako gailuetarako, ostera, egitura hori ezinezkoa da, zeren irteera asko dituen deskodetzaile bat beharko bailuke, eta halako osagai batek leku handia hartuko luke txipean.

8.2. irudian azaltzen da aipatu dugun eraketa daukan gailu baten blokeen diagrama. Hor ageri den memoria hori RAM (irakurketa-idazketa) motakoa da eta noranzko biko datu-erroak dauzka. Matrizeak 128 posizio ditu.



8.2. irudia

## 8.7. HIRU EGOERATAKO ATEAK ETA KONEXIO-BUSAK

Sistema logiko baten osagai guztiak bata bestearekin konektatuta daude informazioa seinale elektriko gisara daraman bide batzuen bidez. Bide horiei bus deritze. Informazio hori noranzko bakarrean nahiz bietan ibil daiteke.

Sistemako bus nagusiak hiru dira:

**Helbide-busa.** Sistemak zenbat helbide-bit onartzen dituen, hainbat eroale edukiko ditu busak.

**Datu-busa.** Zenbat bit dituen sistemak darabilen hitz edo datuak, hainbat eroale izango ditu bus honek. Gehien erabiltzen diren formatuak 4, 8, 16, 32 edo 64 bitekoak dira.

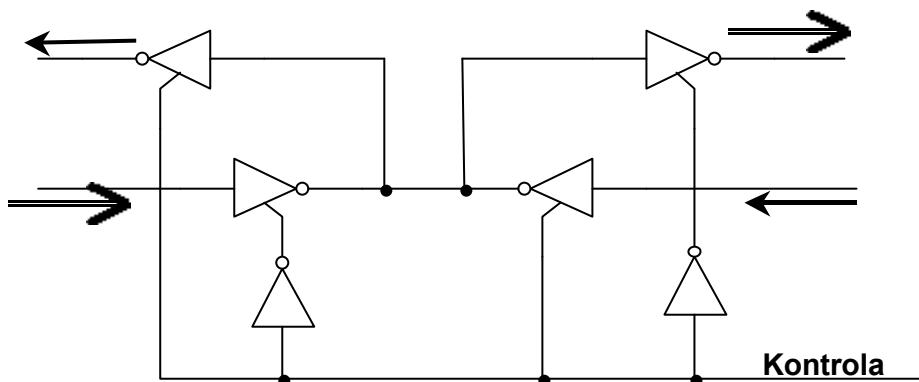
**Kontrol-busa.** Sistemaren osagai guztiak koordinatzeko beharrezkoak diren seinaleak garraiatzeko erabiltzen diren eroaleen multzoa da.

Bus horiez baliatuz, sistemako osagai guztiak paraleloan konektatzeko aukera dago eta aukera horrek mikrokonputagailuen belaunaldi berri bat zabaldu dizkie bideak.

Eta urrats hori **hiru egoeratako ate** logikoen (ingelesezko *tri-state* atea) izenez ere ezagutzen direnak) garapenari esker gauzatu da. Osagai horiek erabili ahal izateak eroale beretik noranzko bietan informazioa bidaltzeko aukera ematen du. Beraz, bus bakoitza bakarra eta noranzko bikoa izan daiteke.

Hiru egoeratako ateen ezaugarria hau da: hirugarren egoera bat eskaintzen dute bat eta zero logikoaz gainera. Irteeran, impedantzia handiko egoera bat agertzen da, sarreran ezartzen den mailak eragiten ez diona.

Esan dugun bezala, noranzko biko busak hiru egoeratako ateei esker daude. 8.3. irudian ikusiko duzue, kontrol-sarreran bateko logiko bat ezarriz gero, informazioa eskuinetik ezkerrerantz joango dela, eta zero bat ezartzen badugu, ostera, informazioa alderantzizko noranzkoan abiatuko dela.

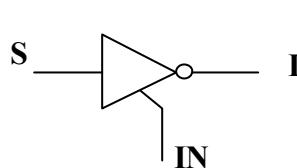


8.3. irudia

Hirugarren egoera gailua gaitu edo desgaitzen duen terminal batek kontrolatzen du; ate alderantzikatzaile soil gisa lan egiten uzten dio IN sarrerari ezarritako maila logikoak bat balio duenean.

Egoera berri horri esker, sarreran horrelako osagaiak dituzten zirkuituek, dagokion terminalari desgaitze-seinalea ezartzen zaionean, sistematik deskonektatua balego bezala joka dezakete.

8.4. irudian azaltzen da sinbolikoki hiru egoeratako atea eta bere egia-taula.



S	IN	I
0	0	Impedantzia handia
1	0	Impedantzia handia
0	1	1
1	1	0

8.4. irudia

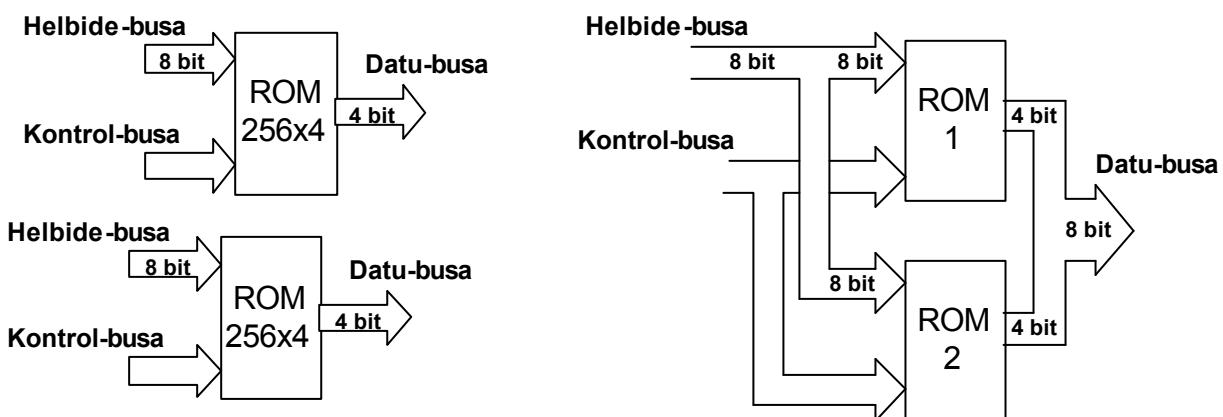
Bus batez ezin daiteke bi gailu edo gehiago aldi berean baliatu; hala eginez gero, kalte konponezinak eragingo lirateke zirkuituan.

## 8.8. MEMORIA HANDITZEA

Memoriak handitu egin daitezke hitz-luzera gehitzeko (helbide bakotzeko bit-kopurua), edukiera gehitzeko (helbide-kopurua), edo biak.

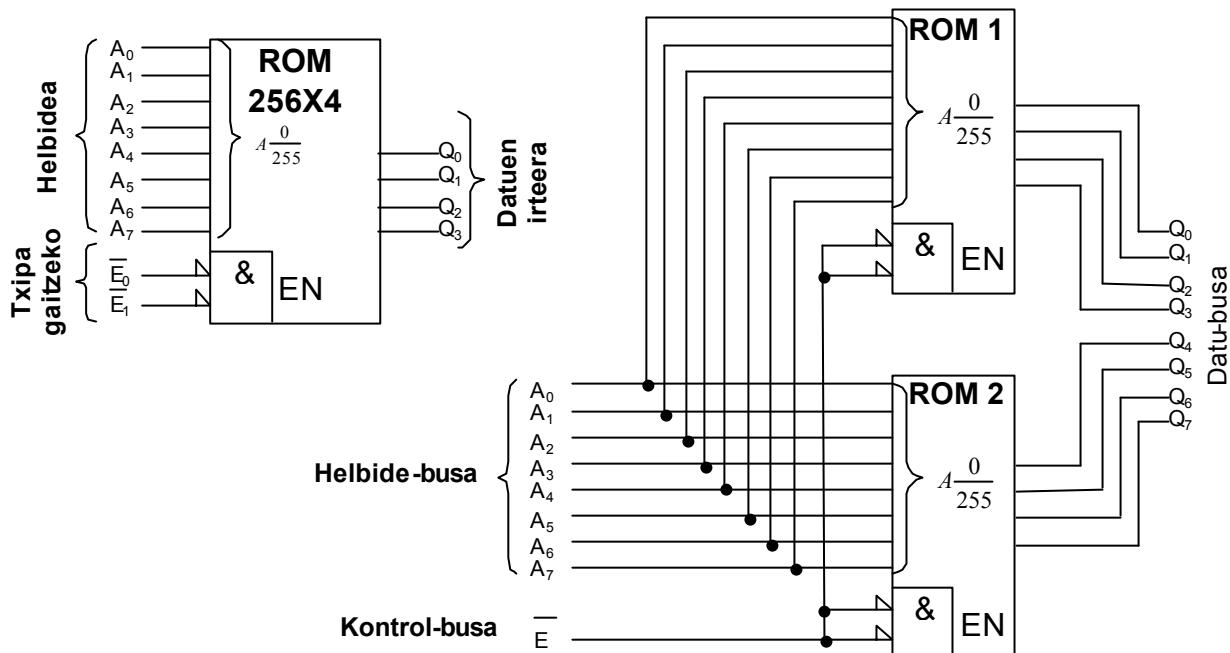
### 8.8.1. HITZ-LUZERA HANDITZEA

Memoria baten hitz-luzera handi genezake datu-busaren bit-kopurua handituz. Adib.: 8 biteko hitzak lor genitzake 4 biteko bi memoria erabiliz. 8.5. irudian ikus daiteke nola egiten den eragiketa hori. Helbide-busa (8 bit) bi memoriei konektatzen zaie; lortzen dugun memoria konbinatuak ( $2^8 = 256$ ) helbide-kopuru bera izango du. Datu-busak elkartu egiten dira, horrela helbide bat aukeratzean datu-busean 8 bit agertzen dira, 4 memoria bakotzetik.



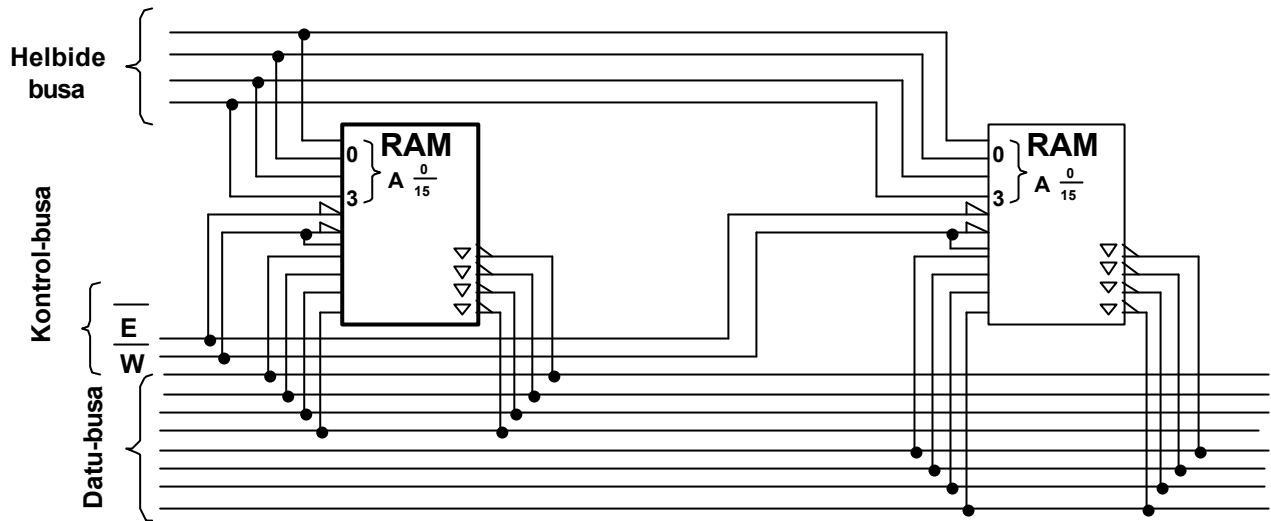
8.5. irudia

**ADIBIDEA:** 256x4 ROM memoria erabiliz, 256x8 ROM memoria lortu. ROM1 eta ROM2 memorien helbide berbera aukeratzen dugu aldi berean. Bakotzaren 4 bitak paraleloan eramanez, datu-busaren 8 bitak lortzen dira. Kontrol-busa ere berbera dute.



8.6. irudia

ROM memoriek datu-irteerak bakarrik dituzte; RAM memoriek, aldiz, datu-irteerak eta -sarrerak dituzte. RAMa handitzean, datu-busa datuen irteerek eta sarrerek osatzen dute, eta, sarrerak eta irteerak elkarrekin konektatu behar direnez, hiru egoeratako atek erabiltzen dira.

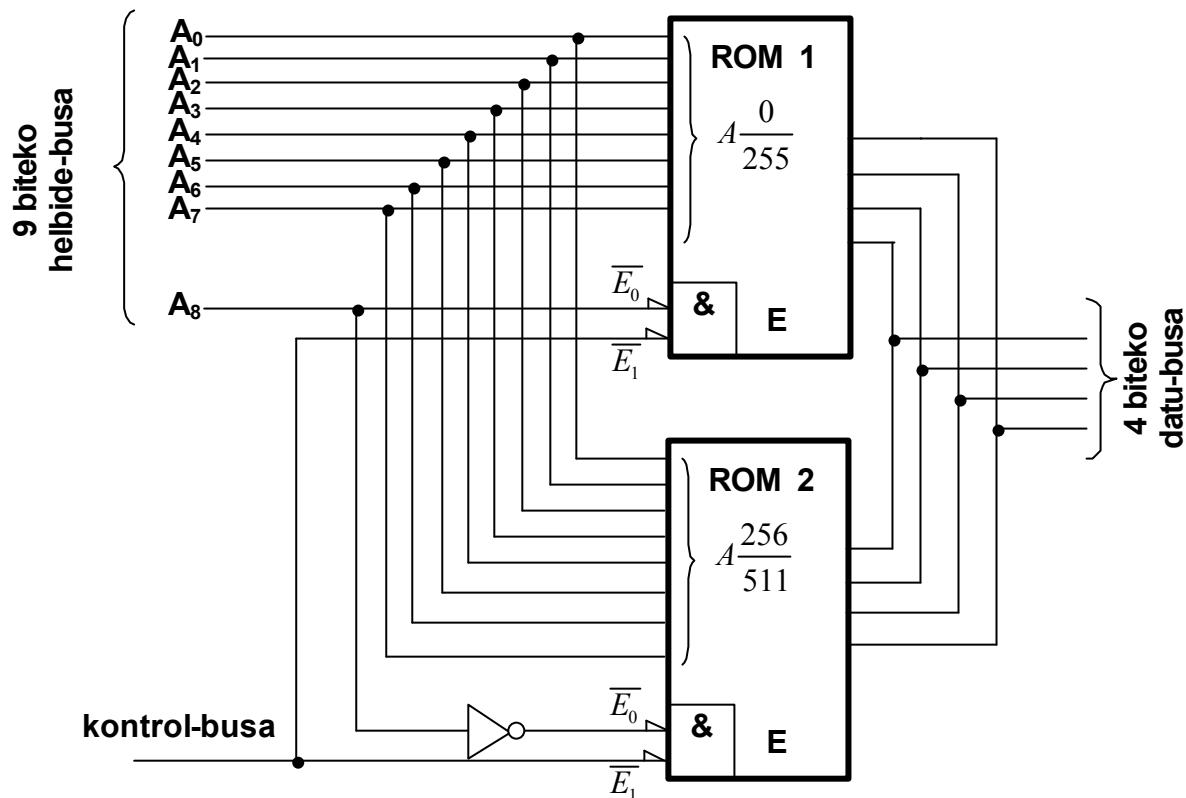


8.7. irudia

**ADIBIDEA:** 16X4 SRAM erabiliz, 16X8 SRAM lortu.  $\bar{W}$  sarrera maila baxuan dagoenean, hiru egoeratako irteerak desaktibatu egiten dira, idazketa egin ahal izateko.

### 8.8.2. EDUKIERA EDO POSIZIO-KOPURUA HANDITZEA

Edukiera handitzeko helbide-kopurua handitu behar dugu, horretarako baimen-sarrera helbide bezala erabiltzen da, 8.8. irudian ikus daitekeen bezala.



8.8. irudia

Helbide-buseko bederatzigarren bita (A<sub>8</sub>) maila baxuan dagoenean, ROM 1 aukeratzen da (ROM 2a desaktibatuta dago), eta pisu gutxieneko helbide-buseko 8 bitak ROM 1eko helbideetara sartzen uzten digute. Maila altuan dagoenean (A<sub>8</sub> = 1), ROM 1 desaktibatzen da eta ROM 2aren 256-511 helbideetarako atzipena dugu.

Memoria-gailuak edukiera handitzeko konbinatzen direnean, errenkada bakarreko moduluetan muntatzen dira (*Single-In-line Memory Module*; SIMM), edo errenkada bikoetan (*Dual-In-Line Memory Module*, DIMM).

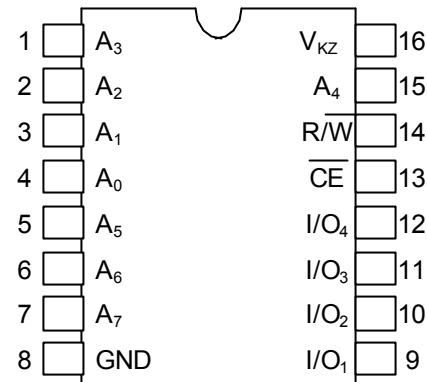
## 8.9. MEMORIA KOMERTZIALEN AZTERKETA

### 8.9.1. SRAM MEMORIA BATEN AZTERKETA (SRAM 2112)

Ondoko 8.9. irudian agertzen den memoriaren hankatxoei begiratuz, esan zein den:

- Meta dezakeen hitz-kopurua
- Metatzen duen hitz-luzera
- Meta dezakeen bit-kopurua
- Memoria-mota

Eskuliburueta aurkitu ezaugarri elektrikoak eta egiturazkoak.



8.9. irudia

Hankatxoei begiratuz esan dezakegu:

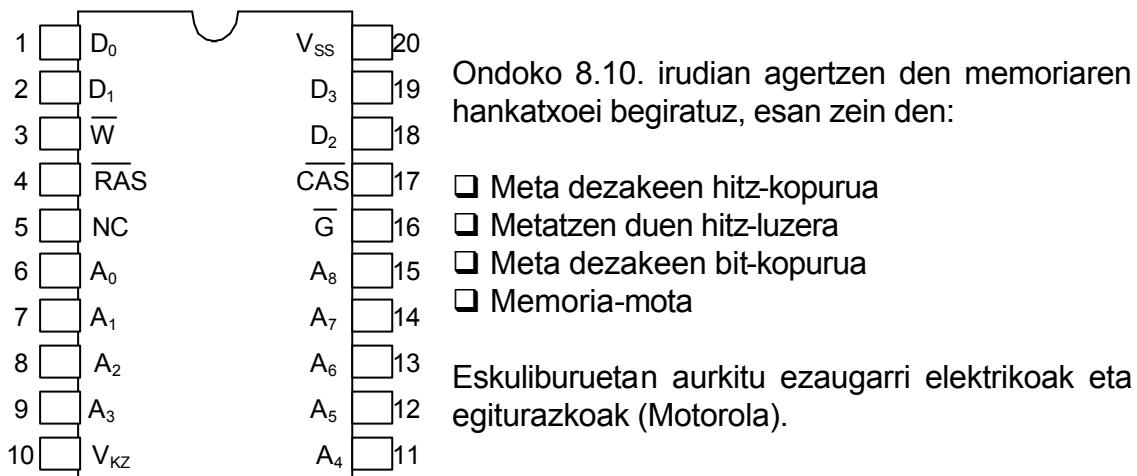
- Helbide-busa osatzen duten lerroak 8 dira: A<sub>0</sub>, A<sub>1</sub>, A<sub>2</sub>, A<sub>3</sub>, A<sub>4</sub>, A<sub>5</sub>, A<sub>6</sub> eta A<sub>7</sub>. Ondorioz, memoria honek dituen helbideak 256 dira ( $2^8 = 256$  helbide). Hau da, 256 hitz meta ditzake.
- Datu-busa osatzen duten lerroak 4 dira: I/O<sub>1</sub>, I/O<sub>2</sub>, I/O<sub>3</sub>, I/O<sub>4</sub>. Horregatik, hitz-luzera 4 bitekoa da.
- Meta dezakeen bit-kopurua  $256 \times 4 = 1.024$  da.
- Integratuaren gainerakoak aztertuz, honako hauek ikus daitezke: elikatzeko ohiko bi hankatxoak (V<sub>KZ</sub> eta GND), txipa aukeratzeko CE hankatxoa eta R / W hankatxoa, RAM memoria dela esaten diguna.

Eskuliburuan (*National Semiconductor*) aurki dezakegun informazioa honakoa da:

- Mota: RAM estatikoa
- Antolaketa: 256 x 4
- Teknologia: NMOS
- Elikadura: + 5 V
- Kapsulaketa: DIL, 16 pinekoa
- TTL logika bateragarria
- Potentzia-kontsumoa : 225 mW
- Atzipen-denbora maximoa : 350 ns

Memoria hauek biegonkorrez edo flip-flopez osatuak daude, set (1) egoeran edo reset (0) egoeran egoten dira, elikadura kentzen ez zaien bitartean edo kommutatzen ez diren bitartean. Horregatik deitzen zaie **estatiko**.

### 8.9.2. DRAM MEMORIA BATEN AZTERKETA (MCM514256A)



8.10. irudia

Hankatxoei begiratuz esan dezakegu:

- Helbide-busa osatzen duten lerroak 9 dira: A<sub>0</sub>, A<sub>1</sub>, A<sub>2</sub>, A<sub>3</sub>, A<sub>4</sub>, A<sub>5</sub>, A<sub>6</sub>, A<sub>7</sub> eta A<sub>8</sub>, baina memoria dinamikoek beren helbide-busa multiplexatua izaten dutenez, 18 lerro balitu bezala funtzionatzen du. Ondorioz, memoria honek dituen helbideak 256K dira ( $2^{18} = 262.144$  helbide). Hau da, 262.144 hitz meta ditzake.
- Datu-busa osatzen duten lerroak 4 dira: D<sub>0</sub>, D<sub>1</sub>, D<sub>2</sub>, D<sub>3</sub>. Horregatik, hitz-luzera 4 bitekoa da.
- Meta dezakeen bit-kopurua  $262.144 \times 4 = 1.048.576$  da.
- Beste hankatxoei erreparatuz, honako hauek ikus daitezke: elikatzeko ohiko bi hankatxoak (V<sub>DD</sub> eta V<sub>ss</sub>), RAM memoria dela adierazten diguna, W, eta bi CAS eta RAS hankatxoak memoria horiek beharrezkoa duten freskatze-prozedura egitea ahalbidetzen dutenak.

Eskuliburuko (Motorola) informazioa:

- Mota: RAM dinamikoa
- Antolaketa: 256K x 4
- Teknologia: NMOS
- Elikadura: 5 V
- Kapsulaketa: DIL, 20 pinekoa
- TTL logikaz bateragarria

- Kontsumo tipikoa: 80 mA
- Helbideratze multiplexatua
- Atzipen-denbora maximoa: 80 ns
- Freskatze-tartea: 8 ms

DRAM memoria hauetan metatze-gelak kondentsadore txikiak dira, eta korronte-ihesak direla-eta kondentsadore horiek deskargatu egiten direnez, informazioa gal dezakete. Hori gerta ez dadin memoria hauek berridazketa-prozesu bat jasan behar dute, prozesu horri freskatzea deritzo (milisegundoero gertatzen da).

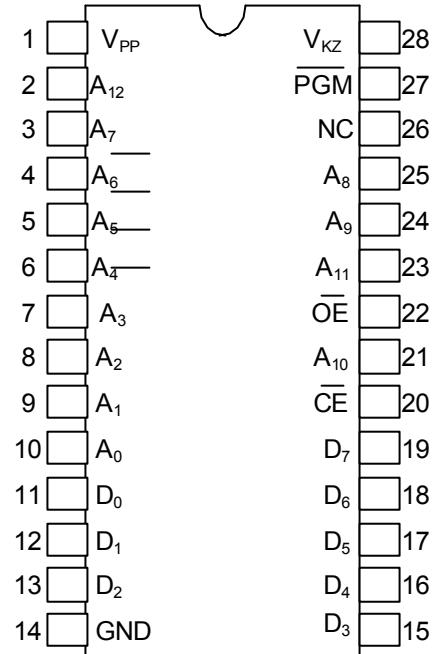
### 8.9.3. EPROM MEMORIA BATEN AZTERKETA (27C64A)

Ondoko 8.11. irudian agertzen den memoriaren hankatxoei begiratuz, esan zein den:

- Meta dezakeen hitz-kopurua
- Metatzen duen hitz-luzera
- Meta dezakeen bit-kopurua
- Memoria-mota

Eskuliburueta aurkitu ezaugarri elektrikoak eta egiturazkoak (Philips).

- Berehalaxe ikusten da zirkuitu integratu honek ez duela R/W hankatxorik, ROM memoria-mota delako. PGM hankatxoa duenez, PROM edo EPROM izango da, horien artean bereizteko kristalezko leihoa duen begiratuko dugu; hala denez, EPROM memoria da.
- Helbide-busa osatzen duten lerroak 13 dira: A<sub>0</sub>, A<sub>1</sub>, A<sub>2</sub>, A<sub>3</sub>, A<sub>4</sub>, A<sub>5</sub>, A<sub>6</sub>, A<sub>7</sub>, A<sub>8</sub>, A<sub>9</sub>, A<sub>10</sub>, A<sub>11</sub> eta A<sub>12</sub>, hau da, memoria honek dituen helbideak 8.192 dira ( $2^{13} = 8.192$  helbide). Ondorioz, 8.192 hitz meta ditzake.
- Datu-busak 8 lerro ditu: A<sub>0</sub>, A<sub>1</sub>, A<sub>2</sub>, A<sub>3</sub>, A<sub>4</sub>, A<sub>5</sub>, A<sub>6</sub> eta A<sub>7</sub>, horiekin 8 biteko hitzekin lan egingo du.
- Meta dezakeen bit-kopurua :  $8.192 \times 8 = 65.536$  bit



8.11. irudia

Eskuliburuuan jasotako informazioa:

- Mota: EPROM
- Antolaketa: 8 K x 8
- Teknologia: NMOS
- Elikadura: + 5V

- Kapsulaketa: DIL, 28 pinekoa
- TTL logikaz bateragarria
- Kontsumo maximoa: 20 mA
- Atzipen-denbora maximoa: 100 ns
- Programatzeko era: + 12-14 V  $V_{PP}$  hankatxoan
- Programatzeko baimena: + 5V PGM hankatxoan

## 8.10. MEMORIA BEREZIAK

**FIFO** memoriak desplazamendu-erregistro multzo batez osatuak daude. Izena ingelesezko *First In First Out* esalditik datorkio, horretan oinarritzen baita bere funtzionamendua. Lehen idazten den datua lehen irakurtzen dena izaten da.

Ohiko desplazamendu-erregistroetan, datua beste datu batzuk sartutakoan mugitzen da eskuinerantz; FIFO erregistroetan, aldiz, sartu ahala hutsik dagoen eskuineko posiziora pasatzen da.

Ohiko desplazamendu-erregistroa

SARRERA	X	X	X	X	IRTEERA
0	0	X	X	X	→
1	1	0	X	X	→
1	1	1	0	X	→
0	0	1	1	0	→

**X= datu ezezagunak**

FIFO erregistroa

SARRERA	--	--	--	--	IRTEERA
0	--	--	--	0	→
1	--	--	1	0	→
1	-	1	1	0	→
0	0	1	1	0	→

**-- = hutsik dauden posizioak**

FIFOen aplikazioetako bat abiadura ezberdineko bi sistemen arteko erlazioa ahalbidetzea izaten da. Datuak FIFOOn sar daitezke abiadura batean eta beste batean irten.

**LIFO** memoriak mikroprozesadoreetan erabiltzen dira. Datuak metatu ondoren, alderantzikor ordenan ateratzen dira, hau da, idatziriko azken datu-bytea irakurtzen den lehen datu-bytea izango da (*Last In First Out*).

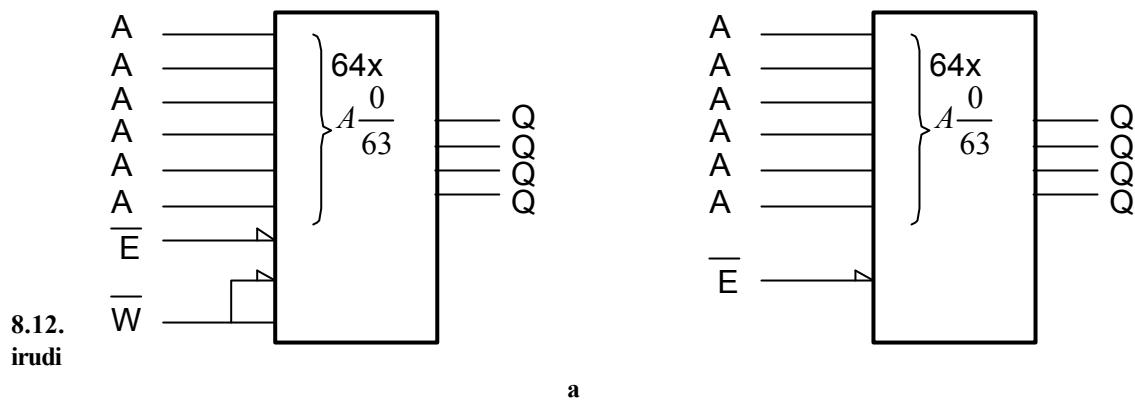
Memoria horiei ‘pila’ esaten zaie (*push-down* pila). RAM memoria baten barruan ere antolatu ahal dira. Horretarako, **pilako erakuslea** (*stack pointer*) deituko den helbidea markatu behar dugu. Helbide horretan sartuko da lehen datua eta pilako erakuslea automatikoki hurrengo helbidean geratuko da. Datuak ateratzeko erakusleak adierazten duenaren aurreko helbidekoa aterako da eta erakuslea aurreko helbidera pasatzen da.

## Ariketak

---

- 8.1. Zein da helbide bakoitzean 8 bit eta 1.024 helbide dituen memoriaren edukiera?:**
  - a) 1.024
  - b) 8.192
  - c) 4.096
  - d) 8
  
- 8.2. 32 biteko hitzak dauzka:**
  - a) 2 byte
  - b) 4 byte
  - c) 4 nibble
  - d) 3 byte eta 1 nibble
  
- 8.3. Noiz metatzen dira zuzeneko atzipeneko memoriaren (RAM) datuak?:**
  - a) irakurtzean
  - b) baimentzean
  - c) idaztean
  
- 8.4. Noiz galtzen dira RAM memorian metaturiko datuak?:**
  - a) elikadura etetean
  - b) helbide horretako datuak irakurtzean
  - c) helbide horretan datu berriak idaztean
  - d) a eta c kasuetan
  
- 8.5. Zein da 256 helbideko memoriaren helbide-busaren lerro-kopurua?:**
  - a) 256
  - b) 8
  - c) 1
  - d) 6
  
- 8.6. Flash memoria:**
  - a) hegazkorra da
  - b) irakurketa/idazketa memoria da
  - c) ez da hegazkorra

**8.7. Identifikatu ondoko 8.12. irudian RAM eta ROM memoria**



**8.8. Azaldu zergatik diren zuzeneko atzipenekoak ROM eta RAM memoriak.**

**8.9. Zein helbide adierazten dute hurrengo zenbaki hamaseitarrek?:**

- a) 0AH
- b) 3FH
- c) CDH
- d) 72H

**8.10. Zenbat lerro ditu 16Kx4 RAM baten helbide-busak? Eta datu-busak?**

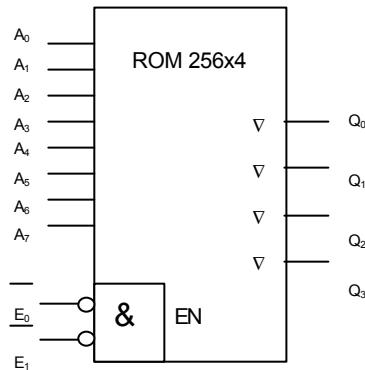
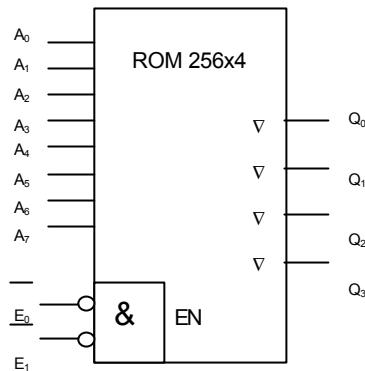
**8.11. 4096x8 RAM memoria baten azken 64 helbideak LIFO memoria egiteko erabiltzen dira; zehaztu hamaseitarrean nondik norainokoa den pila hori.**

**8.12. Aurreko ariketako memorian 16 byte sartzen dira. Zein helbidetan dago lehen sarturiko datua? Eta azkena sarturikoa?**

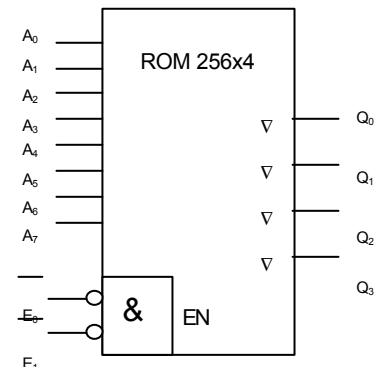
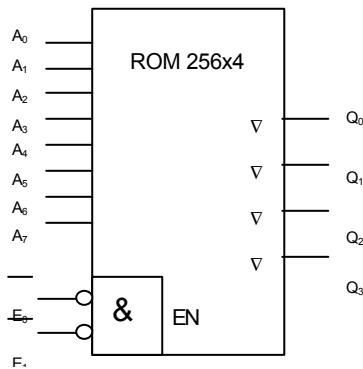
**8.13. 32Kx8 ROM memoria batean:**

- a) Zenbat lerro beharko ditu helbide-busak posizio guztietara heltzeko?
- b) Zenbat gelaxka izango ditu memoria honen matrizeak?
- c) Kode hamaseitarren, zein izango da posiziorik altuena?

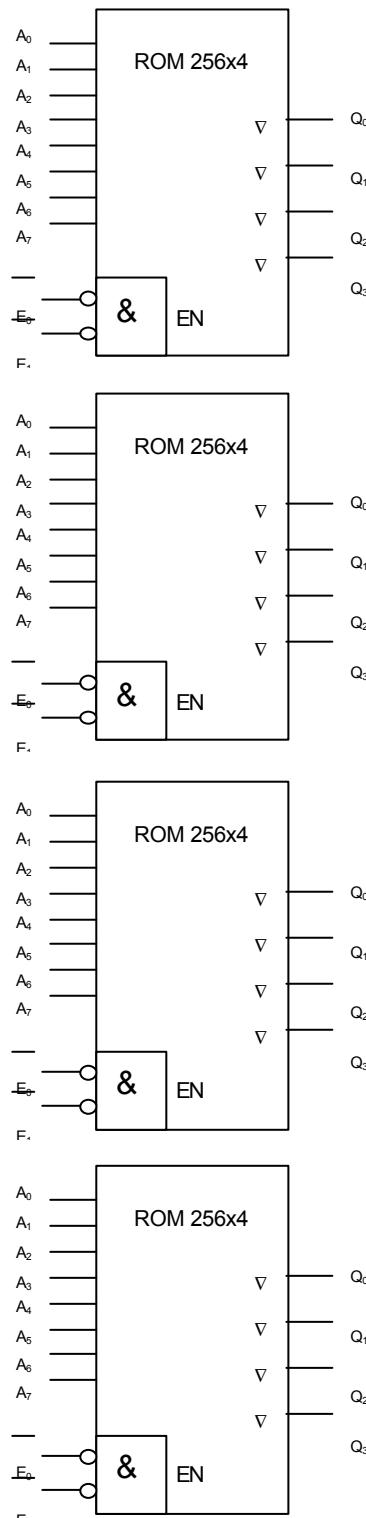
### 8.14. 256x4 ROM memoriak erabiliz, lor ezazu 256x8 ROM memoria



### 8.15. 256x4 ROM memoriak erabiliz, lor ezazu 512x4 ROM memoria



### 8.16. 256x4 ROM memoriak erabiliz, lor ezazu 512x8 ROM memoria



# 9. KAPITULUA

## PLD Gailu Logiko Programagarriak (*Programable Logic Devices*)

### 9.1. SARRERA

Eginkizun bakunentzat, funtzi finkoa duten zirkuitu integratuak (ZI) erabiltzen dira. Adibidez, 74XX (*TTL* seriea), 40XX (*CMOS* seriea) edo mikroprozesadoreak.

Beste aplikazio konplexuagoetan, txip asko erabili ordez, neurriera eginiko zirkuituak erabiltzen dira: *ASIC* (*Application Specific Integrated Circuit*).

#### 9.1.1. *ASIC*

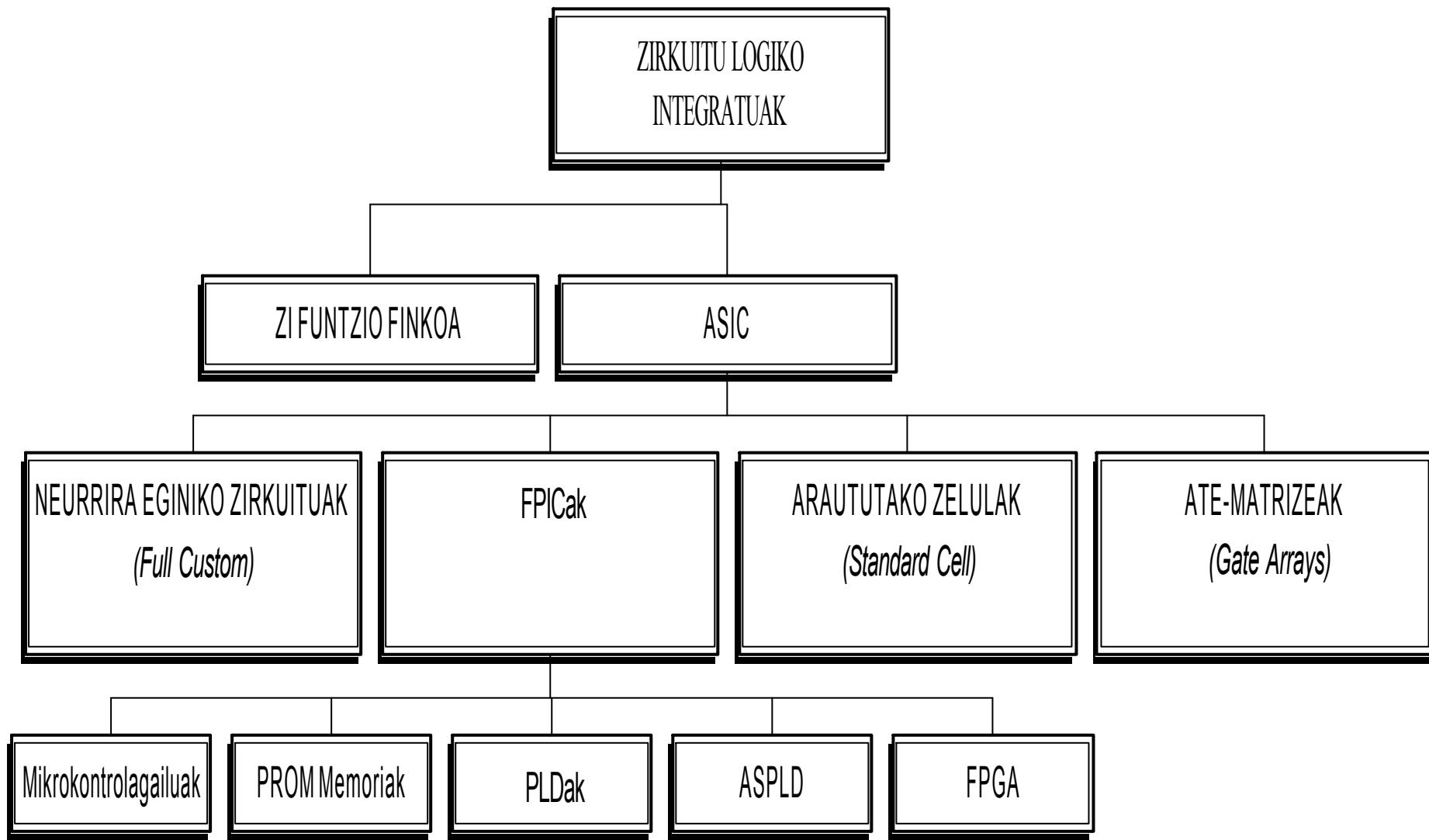
9.1. irudian zirkuitu integratuen sailkapena ikus daiteke.

##### 9.1.1.1. Neurriera eginiko zirkuituak

Bezeroak eskatutako zirkuitua diseinatu egin behar da (*Full Custom*). Diseinua oso garestia denez, kopuru oso handia behar denean baino ez da erabiltzen. Hilabeteak edo urteak behar dira zirkuitu hauek egiteko.

##### 9.1.1.2. Ate-matrizeak

*Gate Array* matrizeak siliziozko zati txikiak dira. Horietan, transistore eta ate logiko asko daude integraturik. Erabilgarria izateko, konexio batzuk falta zaizkio, zeinak metalizazio-prozesuen bidez egiten diren. Aurrekoak baino merkeagoak izan arren, oraindik garestiak dira, eta unitate asko behar denean hautatzen da mota hau. Ate-matrizeak egiteko asteak edo hilabeteak behar izaten dira.



9.1. irudia

### 9.1.1.3. Araututako zelulak

*Standard Cell* zelulak, neurri batean, *Gate Array* matrizeak bezalakoak dira, baina transistoreak eta atea izateaz gain, zirkuitu konplexuagoak ere baditu (*MSI* zirkuituak, *RAM* estatikoak, erregistroak, etab.). Hauek ere nahiko garestiak dira.

### 9.1.1.4. *FPI*Cak

*Field Programmable Integrated Circuits* txipak erabiltzaileak programatzen ditu programagailu komertzialekin. Talde honetan zirkuitu-mota asko sartzen dira: memoriak, mikrokontroladoreak, *PLD*, *FPGA*, *ASPL*.

Arrakasta handia daukate, izan ere, nahiko merkeak izateaz gain, erraz programatzen dira.

#### 9.1.1.4.1. *PLD*ak

*Programmable Logic Devices* deiturikoak *ASIC* txikiak dira eta funtzi logikoak egiten dituzte. Memoriak ere *PLD*ak dira, baina askotan datuak gordetzeko erabiltzen direnez (eta ez funtzi logikoak egiteko), beste multzo batean sartzen dira.

#### 9.1.1.4.2. *ASPLD*ak

*Application Specific Programmable Logic Devices* deiturikoak *PLD* bereziak dira, oso azkarrak dira eta funtzi konplexuak egiten dituzte.

#### 9.1.1.4.3. *FPGA*ak

*Field Programmable Gate Arrays* deiturikoek bloke logikoak dituzte, beraien artean independenteak dira eta bakoitza *PLD* bat bezain konplexua da. Bloke horiek elkarrekin konektatzen dira. Bi motatakoak daude, batzuek bloke gutxi eta konplexuak dituzte, eta besteek bloke asko eta bakunak.

### 9.1.2. *PLD*ak

Zirkuitu hauek erdi-mailakoak dira eta abantaila batzuk dauzkate. Ez dira garestiak eta, gainera, diseinua erraza eta azkarra da. *PLD* batek ZI normaletako dozena batzuk ordezka ditzake. Hona hemen ezaugarri nagusiak:

#### ☞ **Diseinu erraza**

Hasieran (1975) oso neketsua eta korapilatsua zen programazioa, eskuz eta fusiblez fusible egiten zen. Baina, gaur egun, logika programagarrien

konpiladoreak direla eta, oso erraza da (*PALASM*, *AMAZE*, *ABEL*, *CUPL*, *OrCAD/PLD*).

☞ **Ezaugarriak**

Komutatzeko denbora txikia da (zirkuitu normalak baino txikiagoak). Eta kontsumoa ere ordezkatzen dituen txipen kontsumoa baino txikiagoa.

☞ **Fidagarritasuna**

Zirkuitu bat zenbat eta konplexuagoa izan orduan eta akats gehiago izan ditzake. *PLD*ekin, txip gutxiago jartzen ditugunez, akatsak egiteko arriskua gutxitu egiten da.

☞ **Ekonomia**

Txip gutxiago daude, beraz, plakan ere aurrezten dugu. Batzuetan, plaka-kopurua ere txikitzen dugu eta, hori dela eta, konektoreetan ere aurrezten dugu.

☞ **Segurtasuna**

*PLD*ek segurtasuneko fusible bat daukate. Fusible berezi hori apurtuz gero, ezin izango da *PLD*a irakurri.

## 9.2. PLD ZIRKUITUEN EGITURA

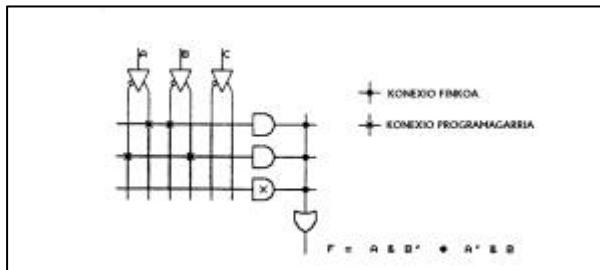
---

Gaur egun 300 mota ezberdin daude. Ez ditugu denak ikusiko, noski, baina bai nagusiak.

### 9.2.1. PLD MOTAK

*PLD*ko ate logikoek sarrera-pila bat dituzte, eta guztiak marraztea ezinezkoa denez, beste sinbologia bat (9.2. irudia) erabiltzen da:

- ☞ AND ateentzat, sarrera-marra bakarra idazten da.
- ☞ Marra hori sarrera bakoitzak dituen bi marrarekin (zuzena eta kontrakoa), gurutzatzen da.
- ☞ Marra bakarra idatzi arren, AND ate bakoitzak ebaketa-puntu adina sarrera ditu.
- ☞ Ebaketa-puntu bakoitzean fusible bat dago. *PLD*ak fusibleak errez programatzen dira.
  - \* Ebaketa-puntuan X idatzita, fusiblea osorik dagoela adierazten da.
  - \* X-rik ez badago, fusiblea erretea dagoela adierazten da.



- Konexio finkoa
- \* Fusiblea badagoela, baina erre daitekeela.
- + Fusiblea erreta.

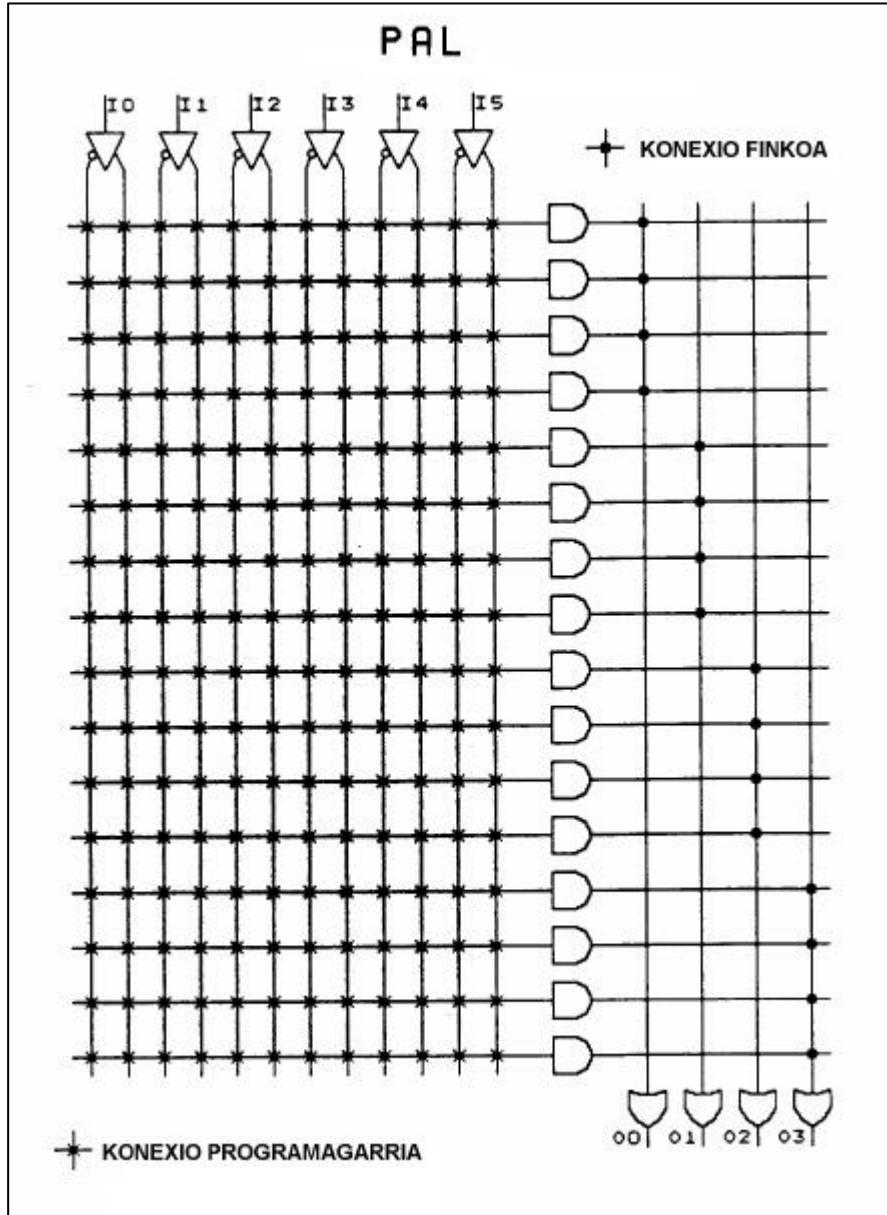
9.2. irudia

9.2. irudian, 18 fusibleko matrize bat programatuta ikusten dugu. Azken ateak **X** bat dauka barruan, horrek esan nahi du fusible guztiak osorik daudela; beraz, bere irteera **1** da.

AND ate batean fusible guztiak erretzen baditugu, bere irteera **1** izango da.

### 9.2.1.1. PAL (*Programmable Array Logic*)

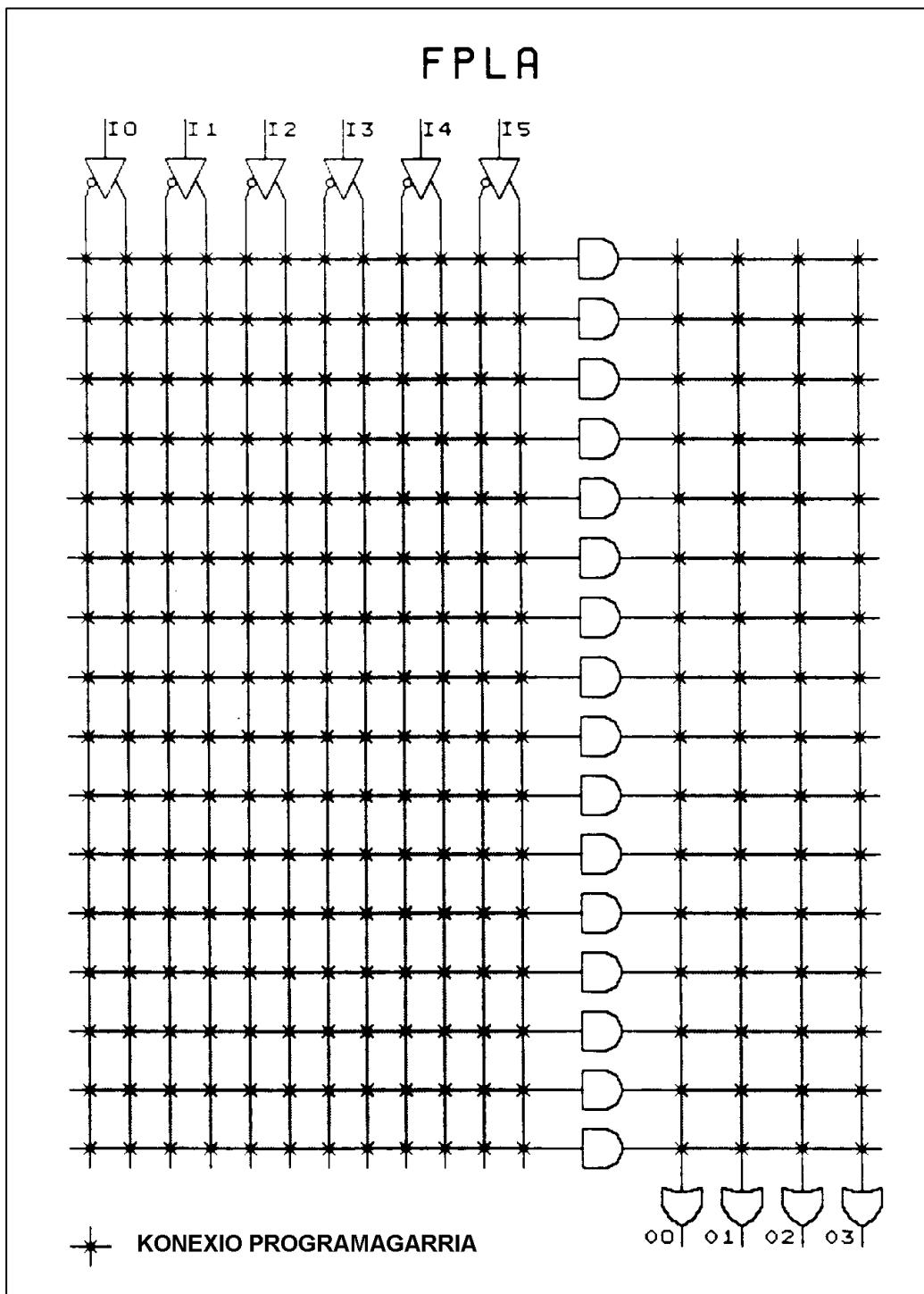
*PLD* mota honetan, *AND* ateen matrizea programagarria da, baina *OR* ateen matrizea, aldiz, finkoa da (programatuta dator). Hauek dira, hain zuzen, erabilienak eta hedatuenak.



9.3. irudia

### 9.2.1.2. FPLA (Field Programmable Logic Array)

PLD mota honetan, bi matrizeak programagarriak dira. Gaitasun handiagoa dute, baina tamaina eta abiadurari dagokionez, okerragoak dira, OR ateen matrizean transistore gehiago baitaramatzate.

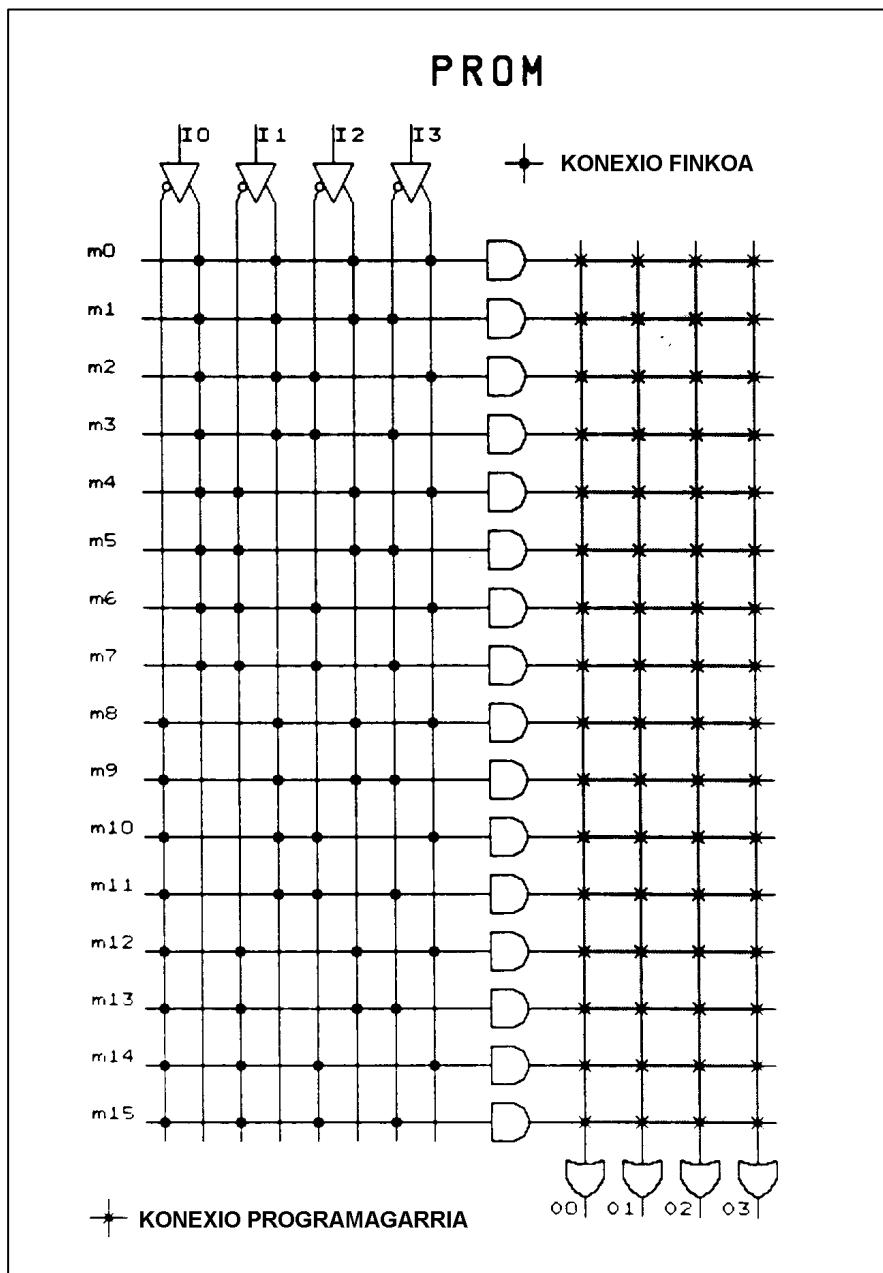


9.4. irudia

### 9.2.1.3. PROM (Programmable Read Only Memory)

Mota honetan, AND ateen matrizea finkoa da, eta programa daitekeena OR ateen matrizea da.

Programatuta datozen konexioak helbideak dira, eta besteetan datuak sartzen dira.



9.5. irudia

## 9.2.2. PAL MOTAK

PALen sailkapena txiparen pin-kopuruaren arabera egiten da. Horrela, hiru talde dauzkagu: 20 pinekoak, 24 pinekoak eta 28 pinekoak.

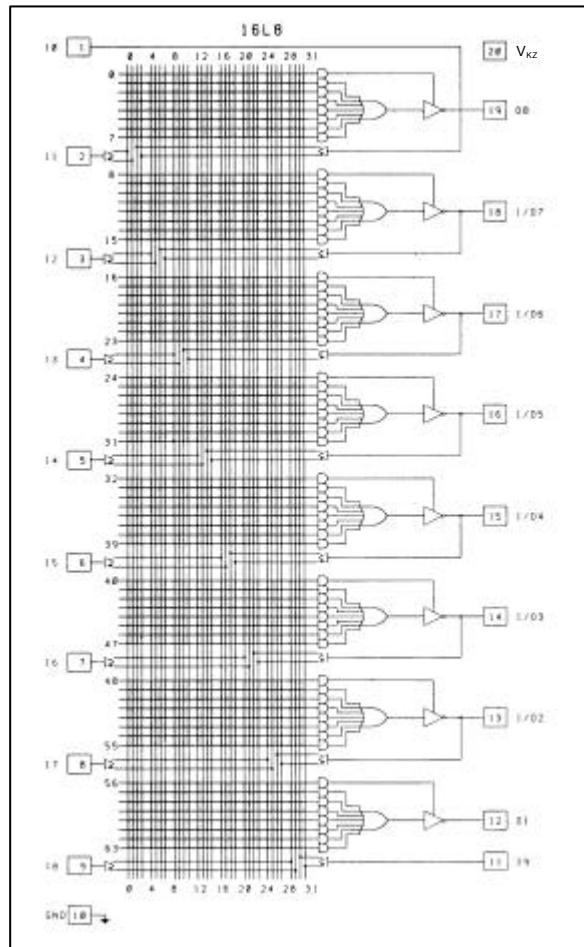
### 9.2.2.1. 20 pineko serie orokorra

Hau da gehien erabiltzen den seriea. Hasieran, MMI (Monolithic Memories Inc.) etxeak egiten zituen, baina gaur egun etxe askok egiten ditu horrelakoak (AMD, Cypress, etab.).

#### 9.2.2.1.1. PAL16L8

PAL honek 8 irteera dauzka. Irteera bakoitzean zazpi sarrerako OR ate bat daukagu. Sarrera horietako bakoitza AND ate batetik dator, non AND bakoitzak 32 sarrera baititu.

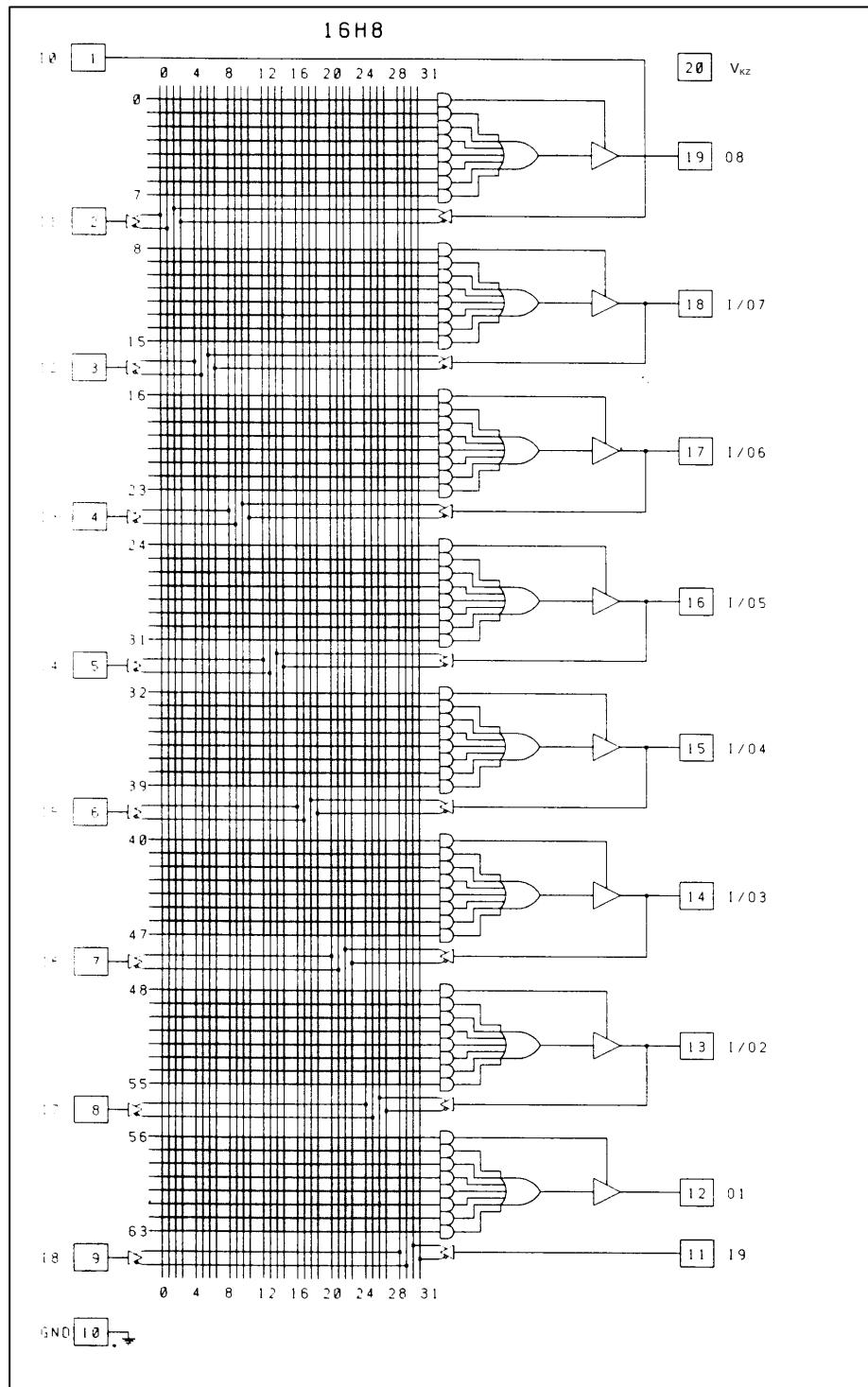
9.6. irudian ikusten denez, pin batzuk sarrera edo irteera (io *input/output*) izan daitezke. Era berean, aipagarria da irteeretan daukagun alderanzgailua.



9.6. irudia

#### 9.2.2.1.2. PAL16H8

PAL honen egitura **PAL16L8** motarenaren bezalakoa da aldaketa txiki batekin: irteeran ez du alderanzgailurik.

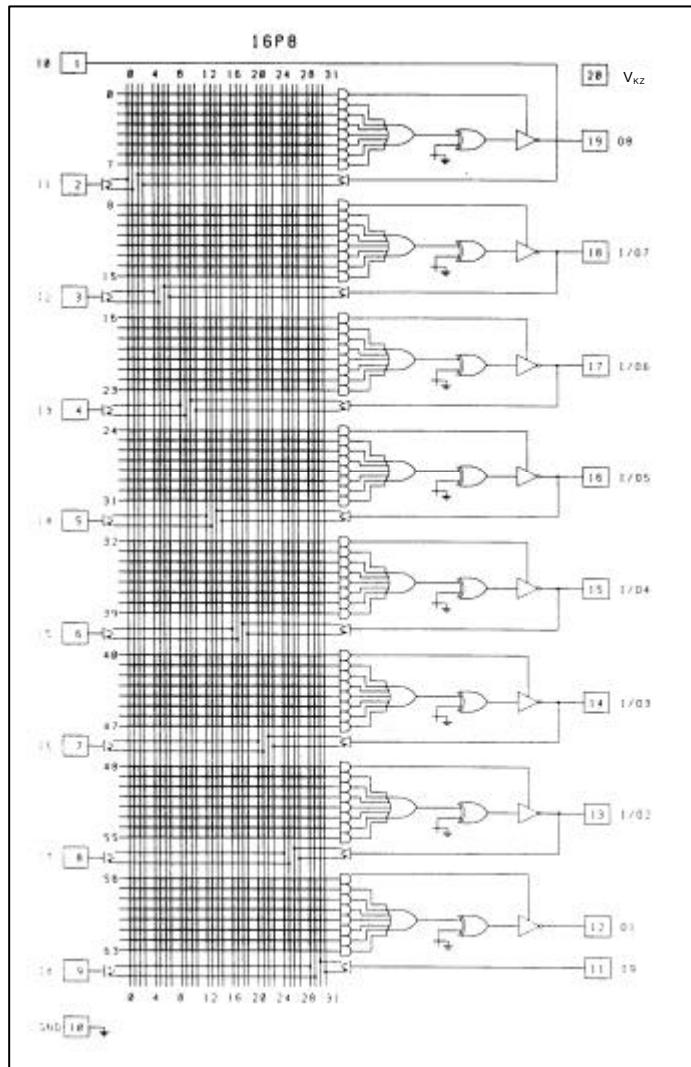


9.7. irudia

### 9.2.2.1.3. PAL16P8

PAL honetan, polaritatea programagarria da XOR ateen bidez. XOR ate horiek sarrera bat, fusible batekin, masara loturik daukate. Fusible hori erreta (sarrera = 1), XOREk alderanzgailu-lana egingo du, eta bi alderanzgailu jarraian izango ditugunez, PAL16P8 PAL16H8 bihurtuko da. Erre gabe uzten badugu (sarrera = 0), XOR zirkuitulaburra izango da, eta, beraz, PAL16P8 PAL16L8 bihurtuko da.

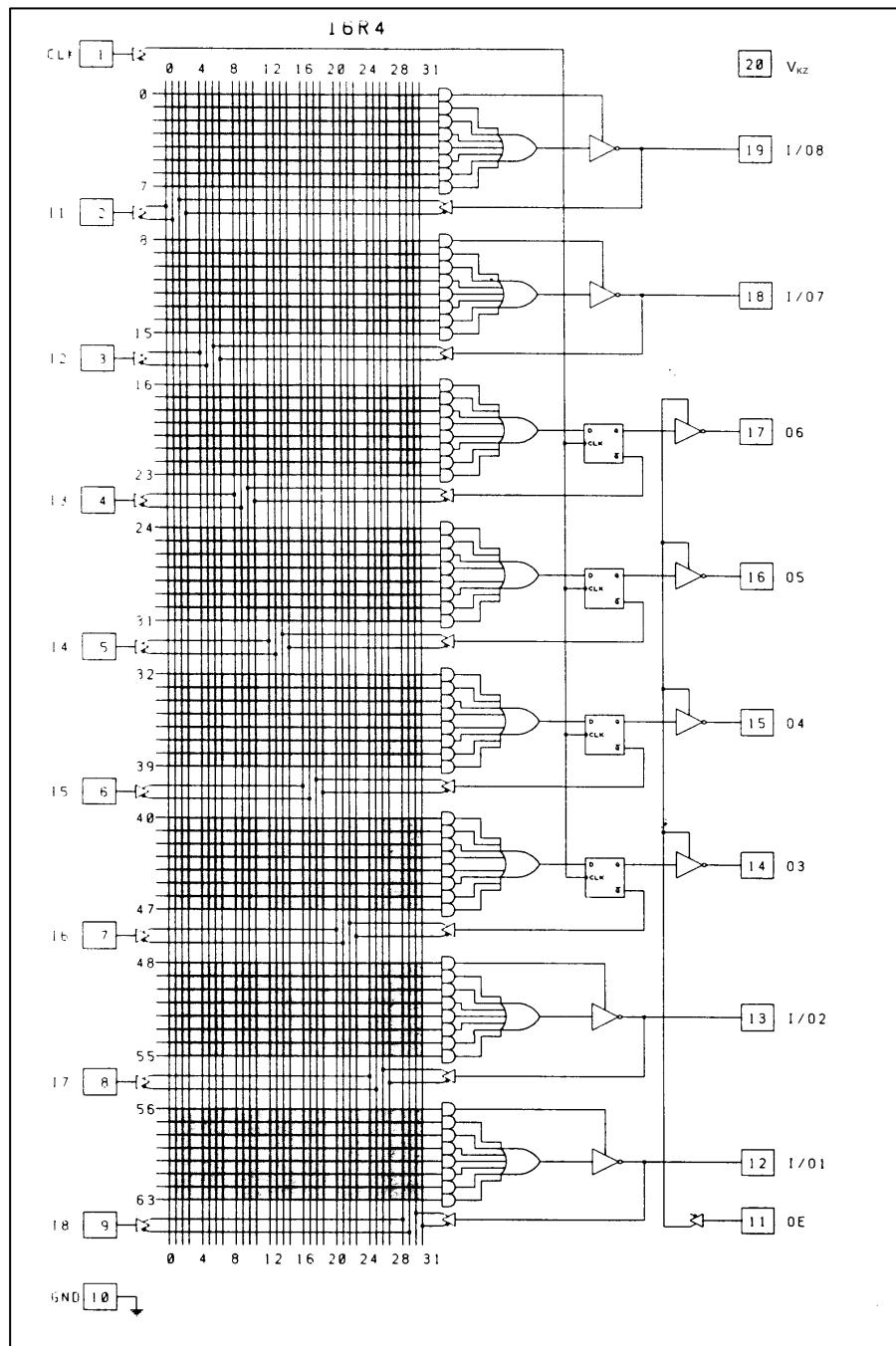
Fusiblea	Sarrera	Irteera	
0	0	0	Fusiblea erre gabe. Sartzen dena irteten da.
0	1	1	
1	0	1	Fusiblea erreta. Alderanzgailu bihurtzen da.
1	1	0	



9.8. irudia

#### 9.2.2.1.4. PAL16R4

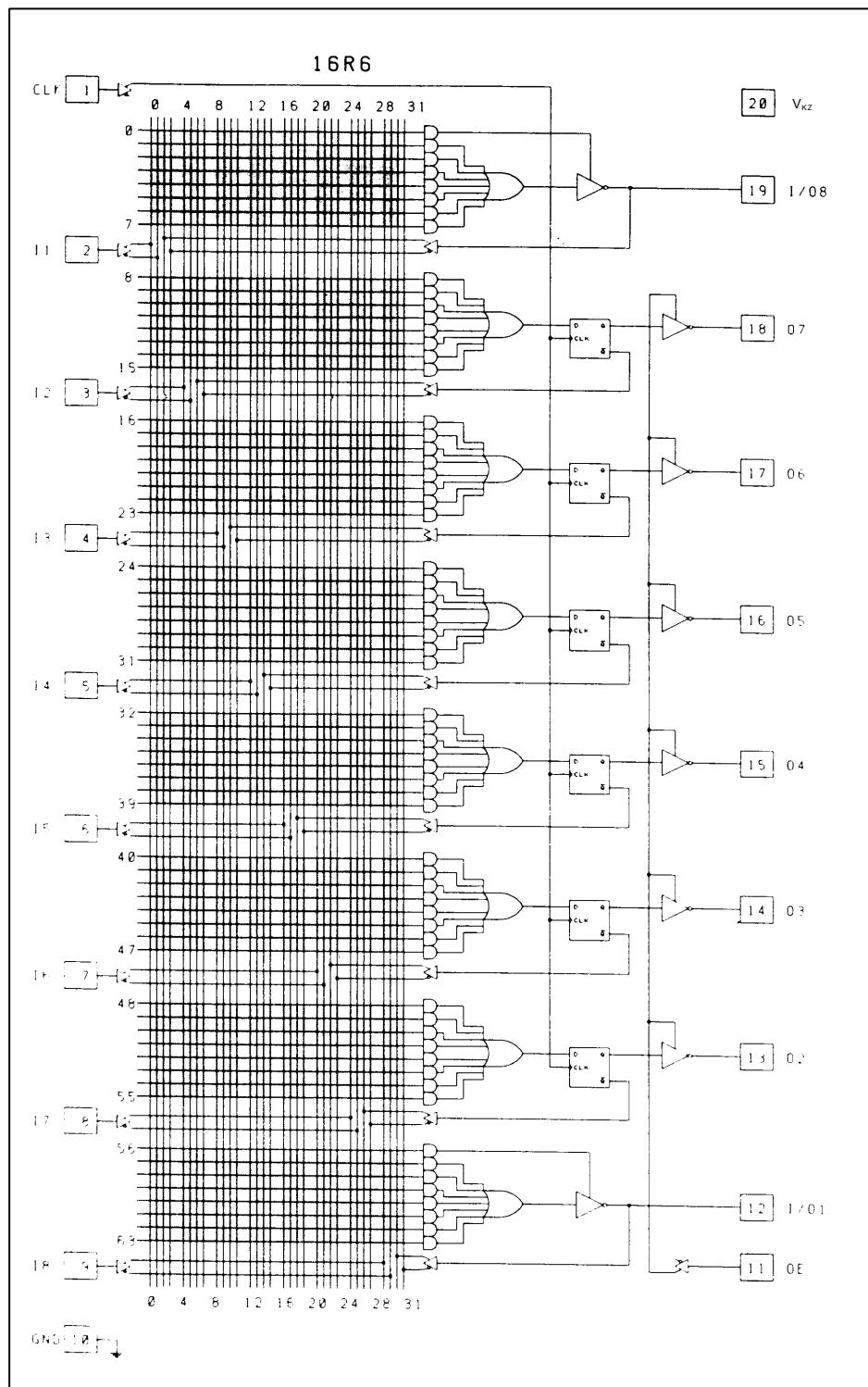
Zirkuitu sekuentzialak egin ahal izateko, *PAL* batzuei bere irteeretan erregistroak (normalean D baskulak) jartzen zaizkie. Horiek izenean **R** letra eramaten dute. Mota horietakoak dira **PAL16R4**, **PAL16R6**, **PAL16R8**. Hemen beste bi pin-mota agertzen dira *CLK* eta *OE* erregistrodun irteerak ahalbidetzeko.



9.9. irudia

#### 9.2.2.1.5. PAL16R6

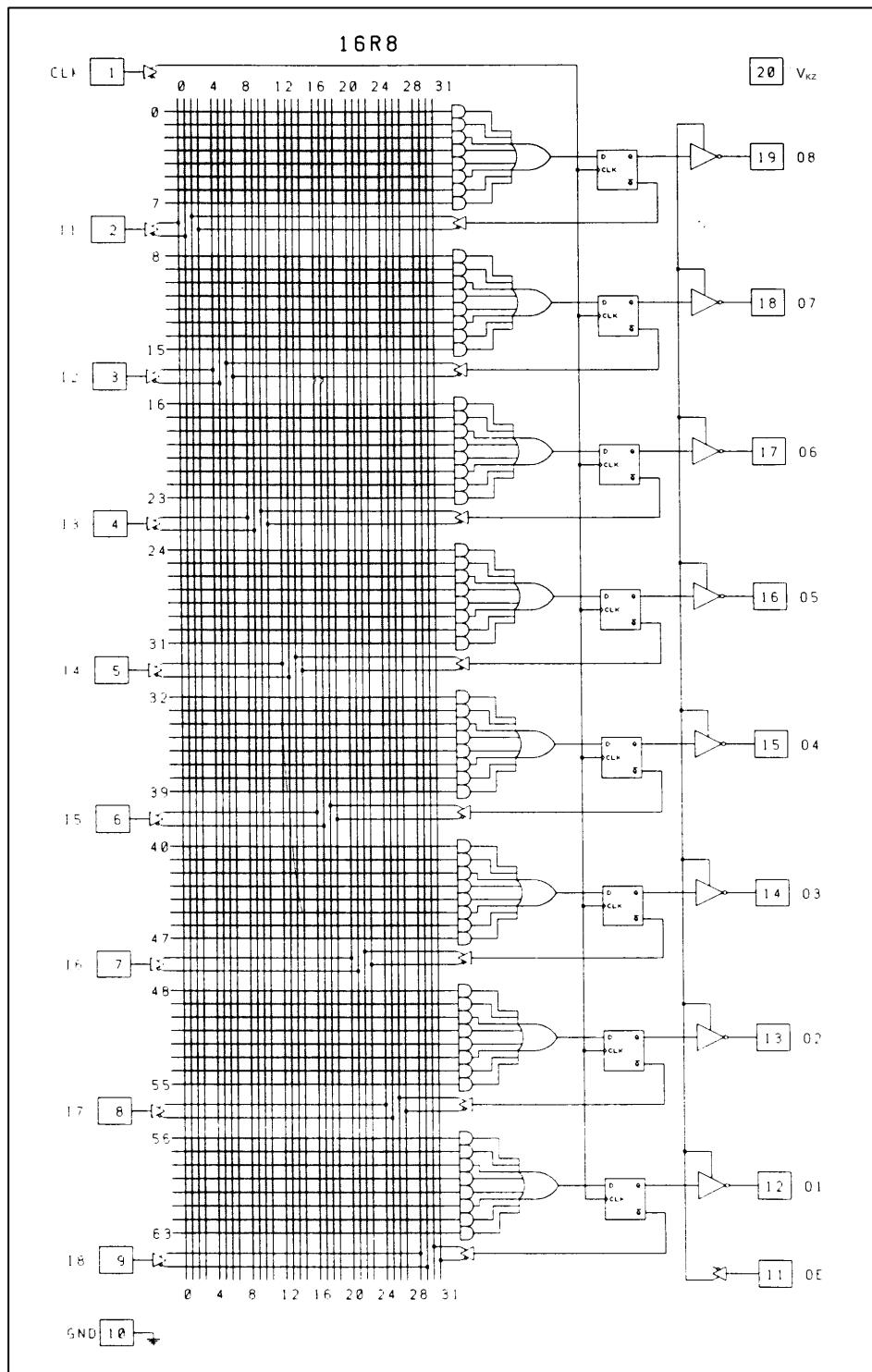
Zortzi irteeratik sei erregistrodunak dira.



9.10. irudia

### 9.2.2.1.6. PAL16R8

Zortzi irteerak erregistrodunak.

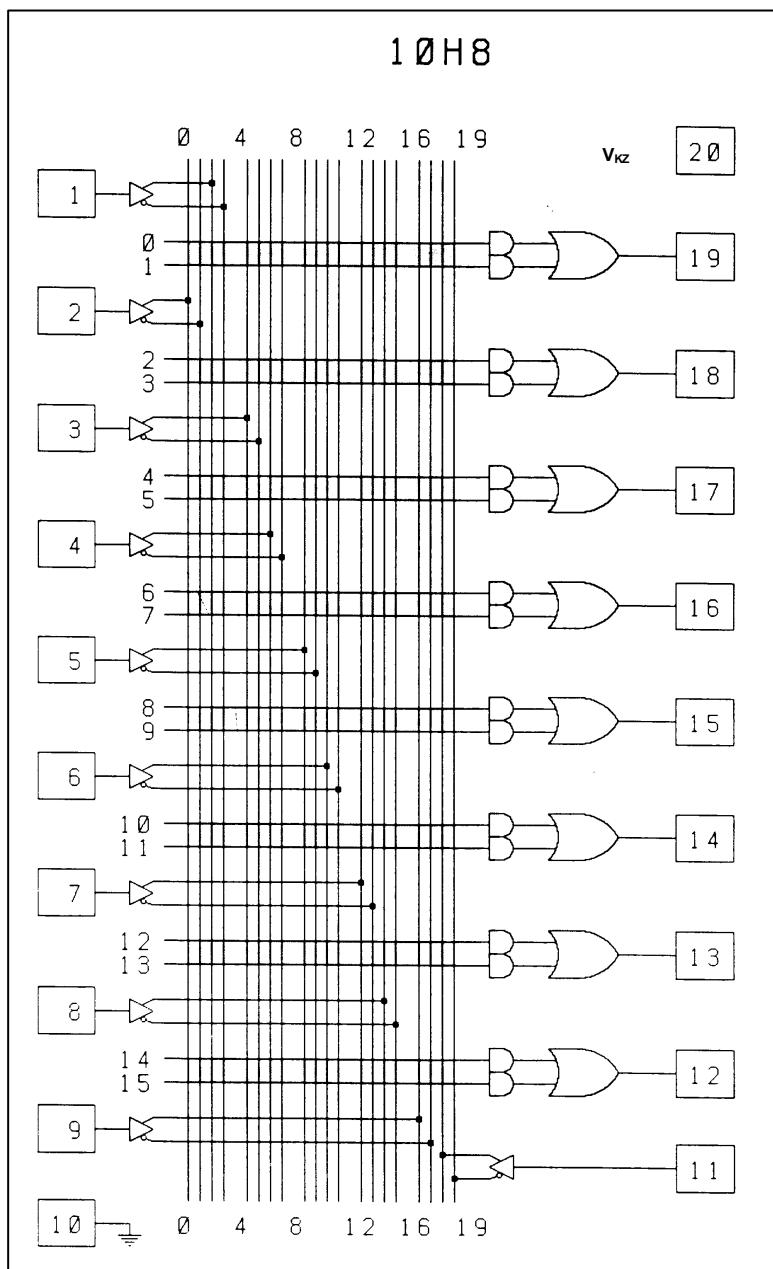


9.11. irudia

### 9.2.2.2. 20 pineko serie berezia

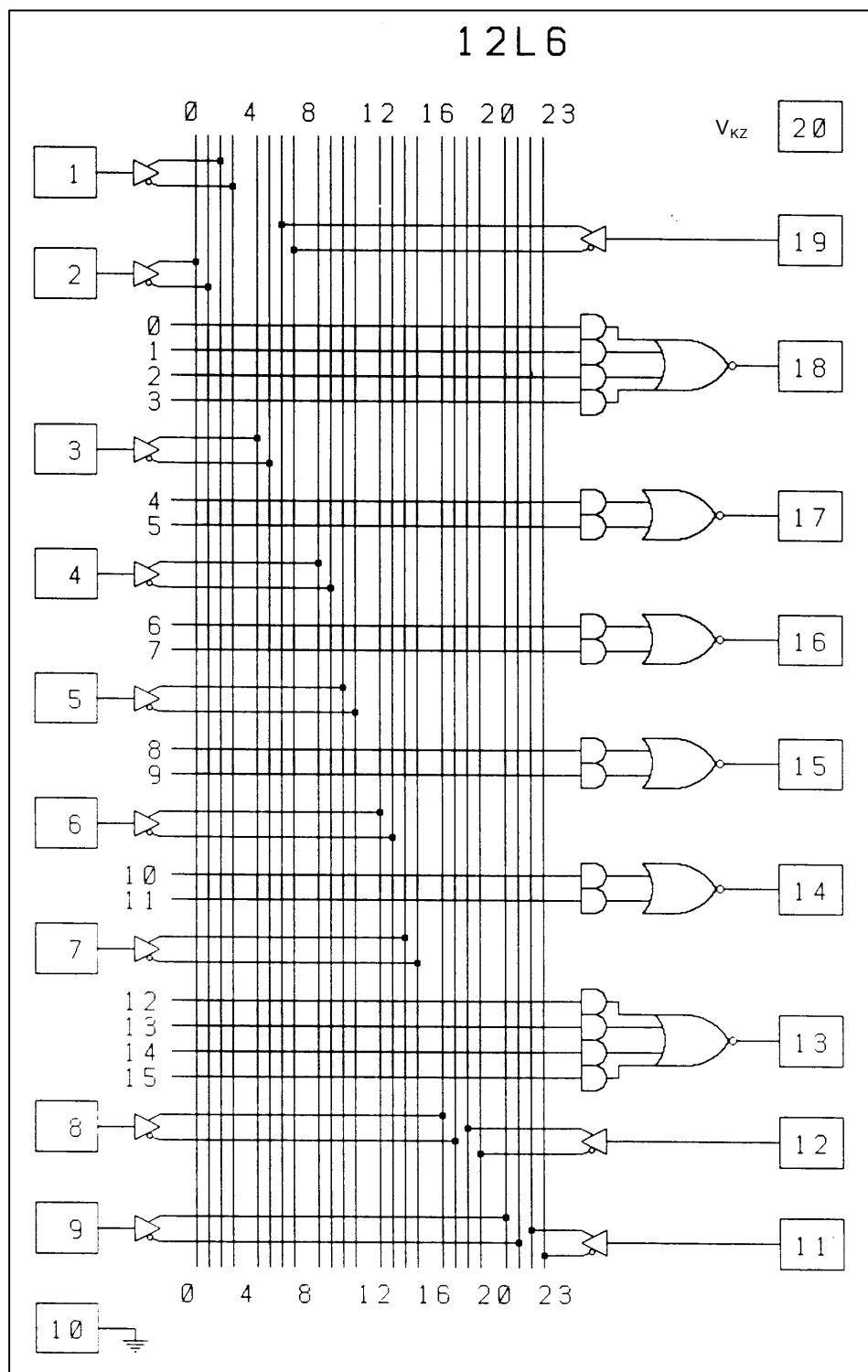
Serie berezi hauetan, fusible gutxiko *PAL*ak daude. Oso produktu-gai gutxi dauzkate, eta funtzio bakunak baino ezin dituzte egin. Hori dela eta, oso gutxi erabiltzen dira. Ezagunenak hauexek dira:

#### 9.2.2.2.1. *PAL10H8*



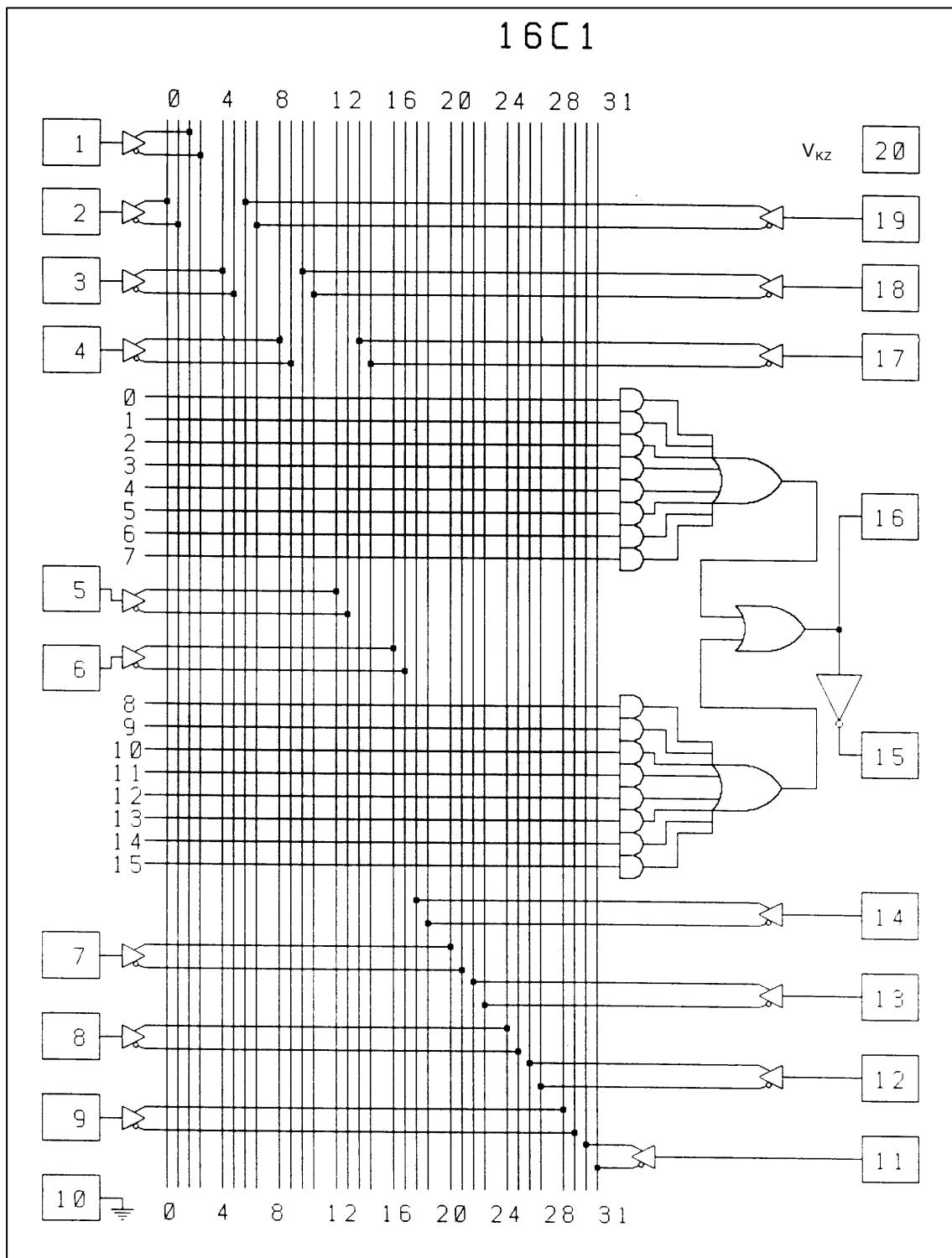
9.12. irudia

### 9.2.2.2.2. PAL12L6



9.13. irudia

### 9.2.2.2.3. PAL16C1



9.14. irudia

### 9.2.2.3. 24 pineko serie orokorra

Serie hau 20 pinekoa bezalakoa da, baina 4 pin gehiagorekin. Normalean, irteera hauek 8 produktu-gai dauzkate. Hurrengo hauek dira ezagunenak: **PAL20L8**, **PAL20H8**, **PAL20P8**, **PAL20R4**, **PAL20R6**, **PAL20R8**, **PAL20RP4**, **PAL20RP6** eta **PAL20RP8**.

### 9.2.2.4. 24 pineko serie berezia

Serie berezi honetan, fusible gutxiko PALak daude. Oso produktu-gai gutxi dauzkate. Adibidez: **PAL12L10**, **PAL12H10**, **PAL12P10**, **PAL14L8**, **PAL14H8**, **PAL14P8**, **PAL16L6**, **PAL16H6**, **PAL16P6**, **PAL18L4**, **PAL18H4**, **PAL18P4**, **PAL20C1**, **PAL20L2**, **PAL20H2**, **PAL20P2...**

### 9.2.2.5. 28 pineko serie orokorra

Barruko egitura besteena bezalakoa da baina sarrera-irteera gehiagorekin. Adibideak: **PAL24L10**, **PAL24R10**, **PAL24R8...** Serie honek ez du arrakastarik izan.

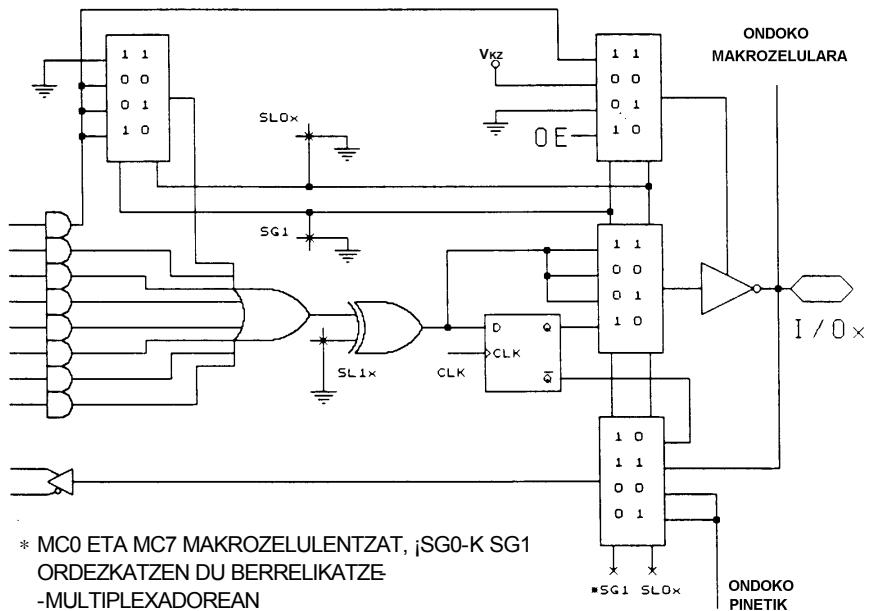
### 9.2.2.6. GAL (Generic Array Logic)

PAL hauetako serie bat edozein PAL ordezka dezakete. Esan dezakegu PAL bakar batean serie oso bat daukagula. Hiru hauetako hedatuenak: **PAL16V8**, **PAL22V10**, **PAL26V12**, 20, 24 eta 28 serieak ordezkatzen dituztenak, hurrenez hurren.

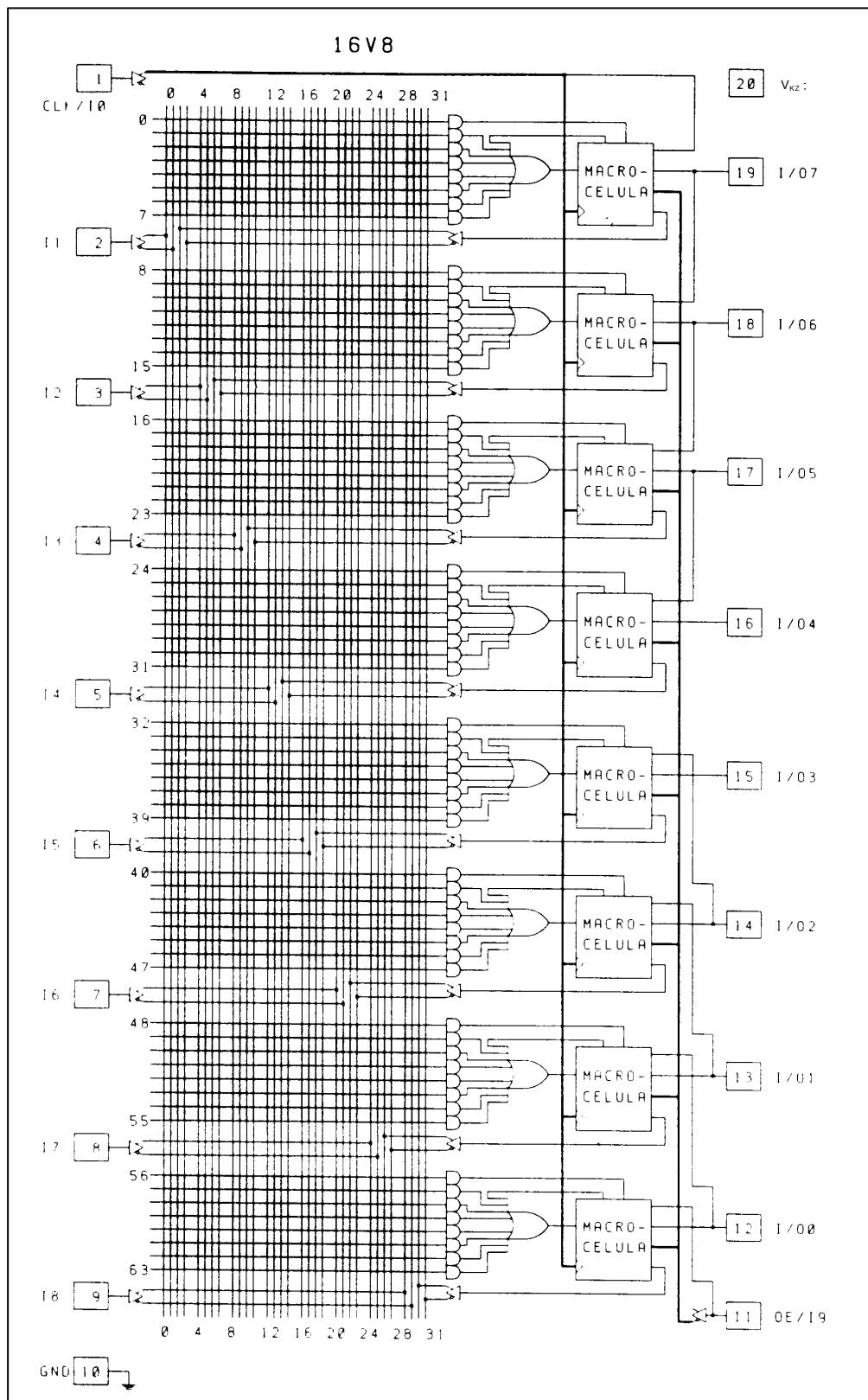
PAL hauen irteera bakoitzean, **makrozelula** izeneko zirkuitu berezi bat dago. Makrozelula horietan, normalean, baskula bat eta multiplexadore batzuk daude. Multiplexadoreak fusible berezi batzuekin programatzen dira, nahi dugun funtzioa egin ahal izateko.

9.15. irudian, **PAL16V8** txiparen makrozelula ikus daiteke, eta 9.16. irudian, AMDk fabrikatutako **PALCE16V8**aren barruko egitura. CMOS teknologiarekin egina dago eta gutxienez 100 aldiz ezaba daiteke.

PALCE16V8

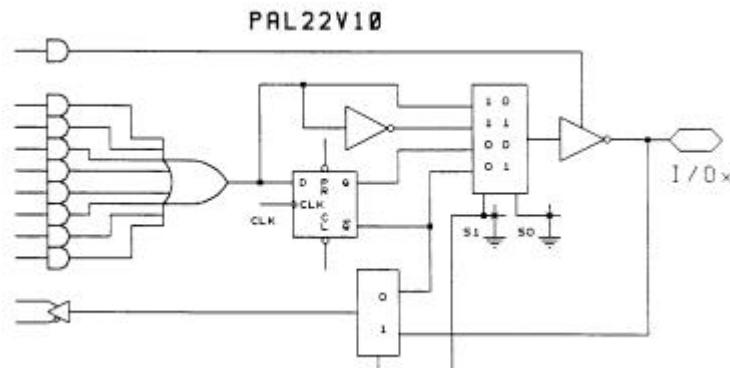


9.15. irudia



9.16. irudia

9.17. irudian, berriz, **22V10 PAL**aren makrozelula.



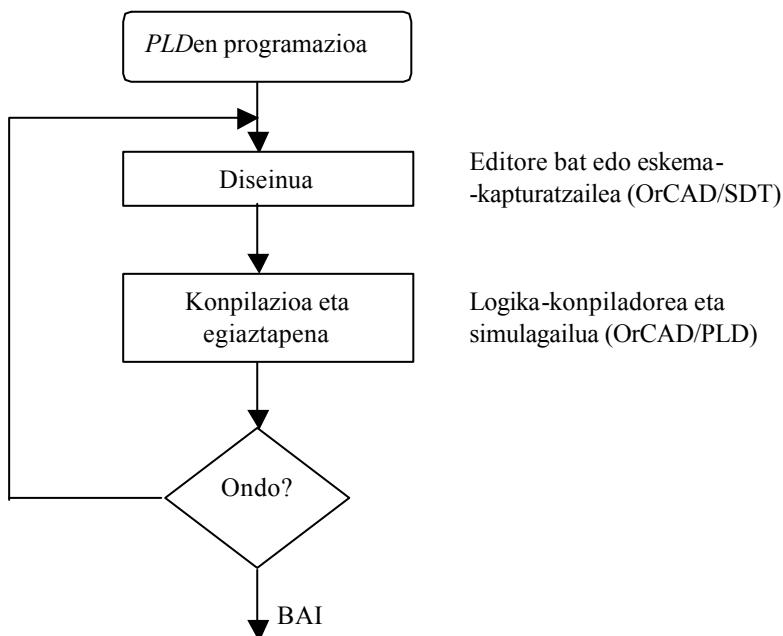
9.17. irudia

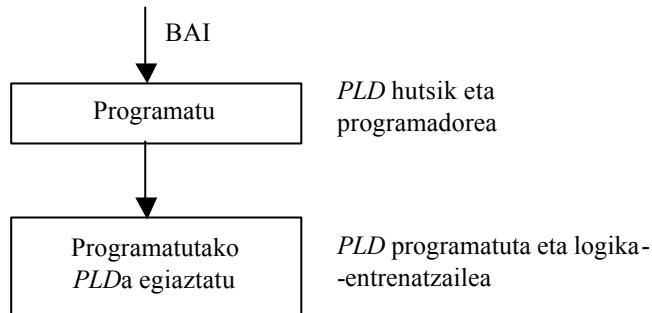
### 9.3. PLD ZIRKUITUAK PROGRAMATZEKO SOFTWAREA

PLDak, memoriak bezala, tresna bereziekin –programagailuekin– programatzen dira. Aparatu horri zer fusible erre behar duen eta zein utzi behar duen erre gabe esan behar diogu. Lan hori errazteko, itzultzaila ditugu. “Logika Programagarriaren konpiladoreak”.

Honako hauek dira ezagunenak: *PALASM*, *ABEL*, *CUPL*, *PLAN*, *AMAZE*, *OrCAD/PLD*,...

#### 9.3.1. PLDak PROGRAMATZEKO URRATSAK





### 9.3.2. ORCAD/PLD SOFTWAREAREKIN ADIBIDE BAT

ASCII editorea erabiliz (*EC*, *CN*, *EDIT...*), iturburu-fitxategia egiten dugu. Horri **.PLD** luzapena jartzen diogu.

#### 9.3.2.1. Iturburu-fitxategia ATEAK.PLD

```

|PAL12H6      in:(A, B, C, D, E, F, G),      out:Y[1..6]

|      Y1 = A & B
|      Y2 = C # D
|      Y3 = B ## F
|      Y4 = G'
|      Y5 = (B & C)'
|      Y6 = (E # F)'

| Vectors:
| {
| Display A, B, C, "    A & B = ", Y1      , "    (B & C)' = ", Y5
| Test A, B, C
| end
| }
```

Konpiladoreak luzapen ezberdinak dituzten hiru **-PLD.EXE-** fitxategi sortzen ditu. Bat **ATEAK.LST** fitxategia da, non akatsak, baldin badaude, agertzen diren eta horrez gain, informazio baliagarria aurrerago aztertuko dugun. Besteak simulaziorako **ATEAK.VEC** da. Eta azkena, **ATEAK.JED**, programatzaileari zer fusible erre behar duen eta zein ez adierazten diona da.

#### 9.3.2.2. ATEAK.LST

Hau da zerrenda-fitxategia:

OrCAD PLD COMPILER-386 v1.00 11/25/91 (Source file .\ATEAK.PLD)

```

1 |PAL12H6      in:(A, B, C, D, E, F, G),      out:Y[1..6]
2 |
```

```

3 | Y1 = A & B
4 | Y2 = C # D
5 | Y3 = B ## F
6 | Y4 = G'
7 | Y5 = (B & C)'
8 | Y6 = (E # F)'
9
10| Vectors:
11| {
12|Display A, B, C, " (A & B)=", Y1, " (B & C)'=", Y5
13|Test A, B, C
14|end
15| }
16
17

```

RESOLVED EXPRESSIONS (Reduction 2)

Signal name	Row	Terms
Y1	0	A B
Y2	4	C
	5	D
Y3	6	B' F
	7	B F'
Y4	8	G'
Y5	10	B'
	11	C'
Y6	12	E' F'

SIGNAL ASSIGNMENT

Pin	Signal name	Column	Rows			Activity
			Beg	Avail	Used	
1.	A	2	-	-	-	High
2.	B	0	-	-	-	High
3.	D	4	-	-	-	High
4.	E	8	-	-	-	High
5.	F	10	-	-	-	High
6.	G	12	-	-	-	High
7.	-	14	-	-	-	
8.	-	16	-	-	-	
9.	-	20	-	-	-	
11.	-	22	-	-	-	
12.	-	18	-	-	-	
13.	Y6	-	12	4	1	High
14.	Y5	-	10	2	2	High
15.	Y4	-	8	2	1	High

16.	Y3	-	6	2	2	High
17.	Y2	-	4	2	2	High
18.	Y1	-	0	4	1	High
19.	C	6	-	-	-	High
-----						
			16	9	( 56 % )	

I200 No fatal errors found in source code.  
I201 No warnings.

OrCAD PLD-386  
Type: PAL12H6  
\*  
QP20\* QF384\* QV1024\*  
F0\*  
L000 01 01 11 11 11 11 11 11 11 11 11 11 11 \*  
L096 11 11 11 01 11 11 11 11 11 11 11 11 11 \*  
L120 11 11 01 11 11 11 11 11 11 11 11 11 11 \*  
L144 10 11 11 11 11 01 11 11 11 11 11 11 11 \*  
L168 01 11 11 11 11 10 11 11 11 11 11 11 11 \*  
L192 11 11 11 11 11 10 11 11 11 11 11 11 11 \*  
L240 10 11 11 11 11 11 11 11 11 11 11 11 11 \*  
L264 11 11 11 10 11 11 11 11 11 11 11 11 11 \*  
L288 11 11 11 11 10 10 11 11 11 11 11 11 11 \*  
C19D5\*

I202 12/31/99 3:35 pm (Friday)  
I203 Memory usage 18K  
I204 Elapsed time 2 seconds

Fitxategi honen goiburukoan konpiladorearen bertsioa eta iturburu-fitxategiaren izena ageri dira; gero, iturburu-kodea, baina lerroak zenbatuta.

“RESOLVED EXPRESSIONS (REDUCTION 2)” atalean, simplifikatzeko zein metodo erabili den adierazten da.

Hurrengo atalean, “SIGNAL ASSIGNMENT” izenekoan, seinale bakoitzari esleitu zaien pin-zenbakia ikus dezakegu. Era berean, zenbat produktu-gai erabili diren ere jakingo dugu.

Gero, ezinbesteko errorerik (*fatal errors*) eta abisurik (*warning*) ez dagoela esaten digu. Konpiladoreak erroreak aurkituz gero, konpilazioa bertan behera uzten du.

Eta bukatzeko, txipa programatzeko behar den informazioa JEDEC formatuan azaltzen da. Nahi izanez gero, fusible-*mapa* ere sor daiteke.

Ikusten denez, fitxategi honetan informazio guztia dago, beraz, proiektuaren dokumentaziorako apropoa da.

### 9.3.2.3. ATEAK.VEC

```
{
  Vector:  A,B,D,E,F,G,"000N00", (Y6,Y5,Y4,Y3,Y2,Y1)L,C,"N";
  Object:  "JEDEC";

  Y1 = (A&B);

  Y2 = ((C)
    #(D));

  Y3 = ((B'&F)
    #(B&F'));

  Y4 = (G');

  Y5 = ((B')
    #(C'));

  Y6 = (E'&F');
}
```

Fitxategi hau simulaziorako erabiltzen da. Simulatu ondoren, **ATEAK.LOG** sortzen da eta egiaztatze-bektoreak **.JED** fitxategiari gehitzen zaizkio.

### 9.3.2.4. ATEAK.LOG

OrCAD TEST VECTOR GENERATOR V1.20E 6/6/90

```
| {
| display A, B, C, "A & B = ", Y1, " (B & C)' = ", Y5
| test A, B, C
0 0 0  A & B = 0  (B & C)' = 1
0 0 1  A & B = 0  (B & C)' = 1
0 1 0  A & B = 0  (B & C)' = 1
0 1 1  A & B = 0  (B & C)' = 0
1 0 0  A & B = 0  (B & C)' = 1
1 0 1  A & B = 0  (B & C)' = 1
1 1 0  A & B = 1  (B & C)' = 1
1 1 1  A & B = 1  (B & C)' = 0
| end
```

Fitxategi honek simulazioaren emaitza zerrendatzen digu. Bertan, programaren funtzionamendua azter daiteke.

### 9.3.2.5. ATEAK.JED

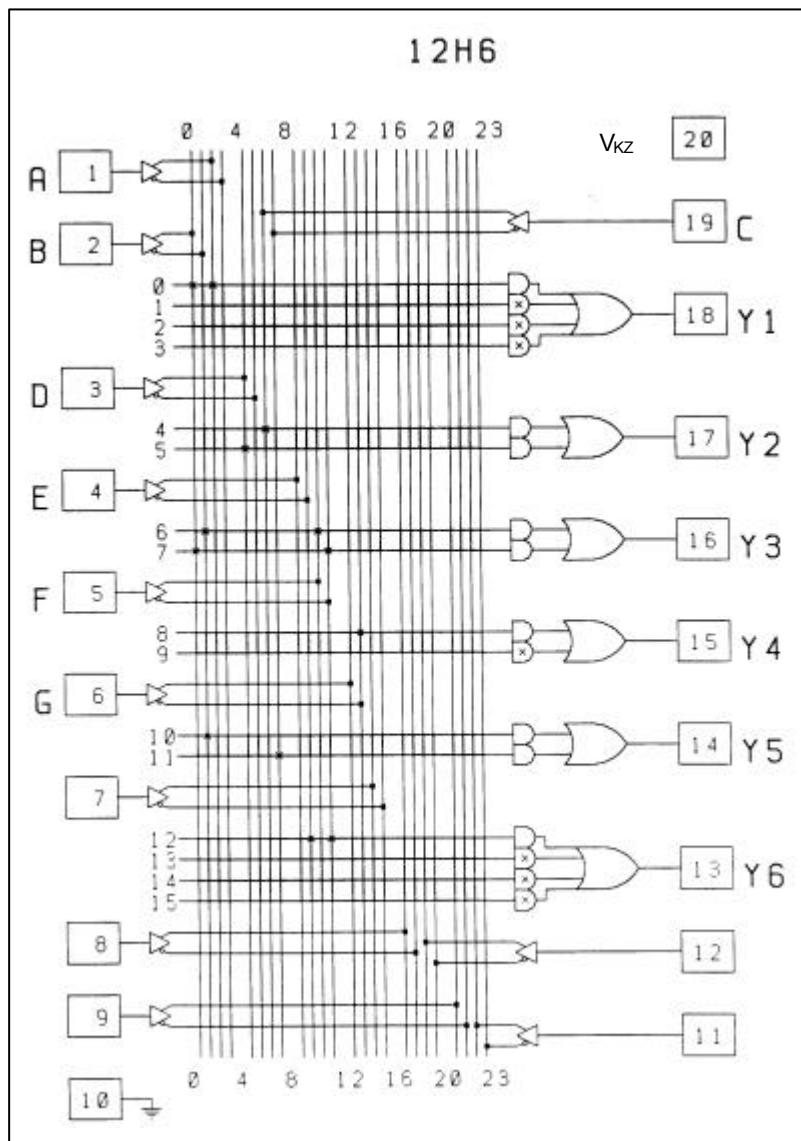
```
OrCAD/PLD
Type:      PAL12H6
*
QP20* QF384* QV1024*
F0*
```

```
L000 01 01 11 11 11 11 11 11 11 11 11 11 11 11 *  
L096 11 11 11 01 11 11 11 11 11 11 11 11 11 11 11 *  
L120 11 11 01 11 11 11 11 11 11 11 11 11 11 11 11 *  
L144 10 11 11 11 01 11 11 11 11 11 11 11 11 11 11 *  
L168 01 11 11 11 11 10 11 11 11 11 11 11 11 11 11 *  
L192 11 11 11 11 11 10 11 11 11 11 11 11 11 11 11 *  
L240 10 11 11 11 11 11 11 11 11 11 11 11 11 11 11 *  
L264 11 11 11 10 11 11 11 11 11 11 11 11 11 11 11 *  
L288 11 11 11 11 10 10 11 11 11 11 11 11 11 11 11 *  
C19D5*  
V00001 000000000N00HHHLLL0N*  
V00002 000000000N00HHHLHL1N*  
V00003 010000000N00HHHLLL0N*  
V00004 010000000N00HLHHHL1N*  
V00005 100000000N00HHHLLL0N*  
V00006 100000000N00HHHLHL1N*  
V00007 110000000N00HHHHLH0N*  
V00008 110000000N00HLHHHH1N*  
8289
```

Fitxategi hau *PLD* programatzeko erabiltzen da. JEDEC formatuan adierazita dago eta, bertan, programatzaileak behar duen informazio guztia dago.

### 9.3.2.6. PAL12H6 programatuta

Hona hemen PAL12H6 bat ‘ateak’ programarekin programatuta.



9.18. irudia

## 9.4. EKUAZIO BOOLEARRAK

PLDn sartu nahi dugun logika adierazteko bi era ditugu, bata OrCAD/SDT erabiliz zirkuitua egitea, eta, bestea, programa baten bidez egitea. Horretarako, lengoaia berezi bat erabili behar da. Horixe da orain aztertuko duguna.

### 9.4.1. KODEA ETA OHARRAK

Kode-lerro guztiak marra bertikal batekin “|” hasi behar dute. Lerro batean marrarik ez badago, edo bigarren marratik aurrera, ohartzat hartuko du konpiladoreak.

#### 9.4.2. SEINALEEN IZENAK

Letra batekin hasten dira, eta, gero, beste letra batzuk edo zenbakiak jar daitezke. Normalean, letra larriak erabiltzen dira. Adibidez: A1, A2, A3, A4. Kortxeteak erabiliz indexa daitezke: A[1..4] = A1, A2, A3 eta A4 edo A[1..3,7..5] = A1, A2, A3, A7, A6 eta A5.

#### 9.4.3. ZENBAKIEN INDEXAZIOA

Zenbakiak ere indexa daitezke:

13[0]	= 1101b[0]	= 1b
13[1]	= 1101b[1]	= 0b
13[2]	= 1101b[2]	= 1b
13[3]	= 1101b[3]	= 1b
13[3..1]	= 1101b[3..1]	= 110b
13[0..3]	= 1101b[0..3]	= 1011b

#### 9.4.4. ERAGILEAK

Eragile asko daude: logikoak, esleipen-eragilea, matematikoak, konparatzaileak eta hardwarekoak.

##### 9.4.4.1. Esleipen-eragilea

Ikur honen bidez ("="), espresio bat irteerako seinale bati esleitzen zaio.

##### 9.4.4.2. Logikoak

'	NOT
!	NOT
&	AND
#	OR
# #	XOR
&'	NAND
# '	NOR
# # '	XNOR

##### 9.4.4.3. Aritmetikoak

+	Batuketa
-	Kenketa
*	Biderketa
/	Zatiketaren zatidura

\ Zatiketaren hondarra  
 \*\* Berretzailea

#### 9.4.4.4. Konparaketak

Eragiketa hauek bit batean ematen dute emaitza. Bit horrek “1” hartzen du konparaketa egia bada, eta “0” hartzen du gezurra bada.

== Berdina  
 /= Ezberdina  
 < Txikiagoa  
 > Handiagoa  
 <= Txikiagoa edo berdina  
 >= Handiagoa edo berdina

#### 9.4.5. EKUAZIOAK

Ekuazioak seinaleen izenak eta operadoreak konbinatuz idazten dira. Adibide batzuk:

|Y1 = (A & (B ## C) # D')'  
 edo  
 |Y1 = A & (B == 1)

##### 9.4.5.1. Ekuazioak idazteko era

Hurrengo lau ekuazio hauek:

$$\begin{aligned} Y1 &= A1 \& B1 \\ Y2 &= A2 \& B2 \\ Y3 &= A3 \& B3 \\ Y4 &= A4 \& B4 \end{aligned}$$

Horrela idatz daitezke:

$$Y[1..4] = A[1..4] \& B[1..4]$$

Edota indexatuz:

$$i = 1..4: Y[i] = A[i] \& B[i]$$

Indize gisa, hurrengo letra xeheak bakarrik erabil ditzakegu: i, j, k, l, m eta n.

#### 9.4.6. PINEN ESLEIPENA

OrCAD/PLD-k zazpi pin ezberdin bereizten ditu:

<i>in</i>	Bakarrik sarrera direnak.
<i>out</i>	Bakarrik irteera direnak.
<i>io</i>	Sarrera edo irteera izan daitezkeenak.
<i>clock</i>	Erregistroentzat <i>clock</i> -a da.
<i>enable</i>	Irteeretan <i>tristate</i> -ak aktibatzeko.
<i>reset</i>	Barruko erregistroak zeroan jartzeko.
<i>preset</i>	Barruko erregistroak batean jartzeko.

Konpiladoreak pinen esleipena automatikoki egiten du horrela adieraziz gero:

|GAL16V8 in:(A,B,C,D), io:Y[2..0]

Konpiladoreak hurrengo esleipen hau egingo du:

(in) 2 3 4 5	(io) 19 18 17
↓ ↓ ↓ ↓	↓ ↓ ↓
A B C D	Y2 Y1 Y0

Dena den, nahi dugun esleipena egin dezakegu. Horretarako, seinaleak horrela definitu beharko ditugu:

|GAL16V8 2:B, 3:C, 4:A, 7:D, 15:Y2, 17:Y1, 19:Y0

## Ariketa ebatziak ekuazioekin

Ekuazioen bidez logika adierazten duten lau adibide azaltzen dizuegu.

### DESKODETZAILEA 4:10 EKUAZIO ARRUNKIN

```
|PAL22V10  in:(A[0..3],CS), io:Q[0..9]
|
|Title: "Deskodetzaile hamartarra (4 → 10) ekuazioekin"
|
| Q0 = (A0' & A1' & A2' & A3' & CS)'
| Q1 = (A0' & A1' & A2' & A3' & CS)'
| Q2 = (A0' & A1 & A2' & A3' & CS)'
| Q3 = (A0' & A1 & A2' & A3' & CS)'
| Q4 = (A0' & A1' & A2 & A3' & CS)'
| Q5 = (A0' & A1' & A2 & A3' & CS)'
| Q6 = (A0' & A1 & A2 & A3' & CS)'
| Q7 = (A0 & A1 & A2 & A3' & CS)'
| Q8 = (A0' & A1' & A2' & A3 & CS)'
| Q9 = (A0 & A1' & A2' & A3 & CS)'
|
| Vectors:
| {
|   Display CS ,(A[3..0])d ,(Q[0..9])b
|   Test CS,A[3..0]
|   end
| }
```

### DESKODETZAILEA 4:10 EKUAZIO INDEXATUEKIN

```
|PAL22V10
| in:(A[0..3],CS), io:Q[0..9]
|
|Title: "Deskodetzailea 4:10 ekuazio indexatuekin"
|
|i = 9..0: Q[i] = (A[3..0]==i & CS)' | Bigarren gaiaren ezeztapenarekin, irteerak "0"
batekin aktiba daitezela adierazten ari gara.

|Vectors:
|{
| Set Q[0..9]
| Display CS, A[3..0], " -> ", Q[0..9]
| Test CS,A[3..0]
| end
|}
```

## 8 KANALEKO MUX-A

```
|PAL22V10 in:I[0..7], io:(S[2..0], Q)
|
| Title: "8 kanaleko MUX-A"
|
| n=0..7:   Q = I[n] & S[2..0] == n
|
| Vectors:
| {
|   Display I[7..0],(S[2..0])d," ----> ", Q
|   Test I[7..0],S[2..0]
|   end
| }
```

## LAU BITEKO BATUTZALEA

Hau ekuazio batean idatz daiteke “+” operadorea erabiliz:

$$Y[4..0] = A[3..0] + B[3..0]$$

Baina, horrela, irteera batzuek 36 produktu-gai behar dituzte eta *PLD* nahiko handia beharko genuke. Ondorioz, diseinua garestia izango litzateke. Logika beste modu batera adieraziz gero, produktu-gai gutxiagorekin emaitza bera lortzen da. Adibidez:

```
|PAL16L8    in: (A[3..0], B[3..0], C[0]),
|          io: (C[3..1], S[3..1]),           | 16L8 PLDak 6 io eta 2 out baino ez ditu
|          out: (C[4], S[0])
|
| Title: "Lauko batutzaile osoa"
|
| i=3..0: { S[i]  = (A[i] + B[i] + C[i])[0]
|           C[i+1] = (A[i] + B[i] + C[i])[1] }
|
| Vectors:
| { Display C[0], "+", (A[3..0])d, "+", (B[3..0])d, \
|   " = ", (C[4], S[3~0])d, \
|   " Tarteko carry-ak: ", C[3..1]
|   Test C[0], A[3..0], B[3..0]
|   End }
```

## 9.5. EGIA-TAULAK

Logika adierazteko erarik errazena egia-taula egitea da. Hau da horren egitura:

| Table: sarrerako seinaleak -> irteerako seinaleak {taula}

Ikus dezagun adibide bat.

```
|PAL14H4 in:(A, B, C, D), out:Y
|
| Table: A, B, C, D -> Y
| { 000Xb -> 0
|   0010b -> 0
|   0011b -> 1
|   0100b -> 0
|   0101b -> 1
|   0110b -> 0
|   0111b -> 1
|   100Xb -> 0
|   101Xb -> 1
|   11XXb -> 1 }
```

Ikusten denez, “X” gai bat baino gehiago adierazteko erabil daiteke. Konbinazio bat bi aldiz jartzen badugu, jarritako lehena hartzen da, bestea albora utziz. Aipatutako arau horretan oinarrituz, adibidea honela adieraz daiteke:

```
|PAL14H4 in:(A, B, C, D), out:Y
|
| Table: A, B, C, D -> Y
| { 000Xb -> 0
|   0010b -> 0
|   0100b -> 0
|   0110b -> 0
|   100Xb -> 0
|   0XXXXb -> 1
| }
```

Ezkerreko gaiak sistema hamartarrean ere adieraz daitezke. Baita bat baino gehiago batera adierazi ere. Ikus ditzagun bi adibide berdin:

```
| Table: A, B, C, D -> Y
| { 0..2 -> 0
|   3 -> 1
|   4 -> 0
|   5 -> 1
|   6 -> 0
|   7 -> 1
|   8..9 -> 0
|   10..15 -> 1 }
| Table: A, B, C, D -> Y | Beste era bat.
```

```
| { 0~2, 4, 6, 8..9 -> 0  
|   3, 5, 7, 10..15 -> 1 }
```

Bi kasuetan, simulazioa egiteko hau erabiliko dugu.

```
| Vectors:  
| { Display (A, B, C, D), " -> ", Y  
|   Test   A, B, C, D  
| End }
```

Ariketa gisa, saia zaitez ekuazioekin eginiko programak taulekin egiten.

## 9.6. STREAM-AK

---

Programatzeko era hau taula bezalakoa da, baina hemen irteerak baino ez dira adierazten. Irteeren ordena naturala errespetatu behar da. Ikus dezagun aurreko adibidea *stream*-ekin adierazita:

```
| Stream: A, B, C, D -> Y  
| { 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1 }
```

Elkarketak ere onartzen ditu:

```
| Stream: A, B, C, D -> Y  
| { 3(0), 3(1, 0), 0, 6(1) }
```

Ariketa gisa, saia zaitez ekuazioekin eginiko programak *stream*-ekin egiten.

## 9.7. MAP-AK

---

Sarrera eta irteeren arteko zenbaki-erlazio garbia dagoenean, *map*-ekin erraz programa daiteke. Logika-mota hau adierazteko, **n** letra erabiltzen da. **n** indize-aldagaiak sarrerak hartzen dituzten balio guztiak ordezkatzen ditu. Esate baterako, sarrerako seinaleak A2, A1 eta A0 baldin badira, **n**-k zeroaren eta zortziaren arteko balio guztiak hartzen ditu.

*Map*-ak oso egokiak dira kontagailuak definitzeko. *Map*-en egitura hau da:

**|Map: Sarrerako seinaleak -> Irteerako seinaleak { map }**

Espresio bat baldintzatu nahi badugu, lerro berean eta koma baten atzetik baldintza idatzi behar da.

## Ariketa ebatziak MAP-ekin

### MAP-EKIN KONTAGAILU ARRUNTA

Adibide honetan kontagailu arrunt bat adierazten ari gara:

```
| PAL16V8  in:R, clock:CLK, io:Q[3..0]
| Title: " Kontagailua map-ekin"
|
| Register: CLK // Q[3..0]
| Map: Q[3..0] -> Q[3..0]
| { n -> n+1, R'
|   n -> 0, R
| }
|
| Vectors:
| {
|   Display (CLK)c,Q[3..0]
|   Test R=1; CLK
|   Clear R
|   Test CLK=15(0,1)
|   End
| }
```

Baskulekin lan eginez gero, **Vectors**-en **Test R=1; CLK** eta **Clear R** komandoak beharrekoak dira. Horien bidez, baskulak hasieratzen ditu (kasu honetan zeroarekin); bestela, simulagailuak egoera mugagabea esleitzen die, eta ezin izango da simulazioa egin.

### HOTELEKO IGOGAILUAREN KONTAGAILUA

Bigarren adibide hau hoteleko igogailuaren kontagailua da. Jakin denez, hoteletan ez dago hamahiru zenbakia daraman pisurik, beraz, hamahiruari dagokion egoera kendu behar dugu. Kontuan izan, gorantz zein beherantz kendu behar dela.

```
| PAL16V8  in:(UP,R), io:Q[3..0], clock:CLK
|
| Register: CLK // Q[3..0]
|
| map: Q[3..0] -> Q[3..0]
| {
|   n -> n+1, R' & UP & Q[3..0]!=12
|   n -> n-1, R' & UP'& Q[3..0]!=14
|   n -> 0, R
|   n -> n+2, R' & UP & Q[3..0]==12
|   n -> 12, R' & UP' & Q[3..0]==14
| }
```

```

| Vectors:
| {
| Display (CLK)c,R,UP," SOLAIRUA: ",(Q[3..0])d
| Test R=1; CLK
| Clear R
| Test UP=1; CLK=14(0,1)
| Test UP=0; CLK=14(0,1)
| End
| }

```

Map-aren lehendabiziko ekuazioa  $n \rightarrow n+1, R' \& UP \& Q[3..0] \neq 12$  horrela irakurtzen da: erloju-pultsu bat dagoenean ( $\rightarrow$ ), baskulak  $n$  egoeratik  $n+1$  egoerara pasatuko dira, baldin eta  $R=0$ ,  $UP=1$  eta egoera **12** ez bada.

## HIRUROGEIRAKO KONTAGAILUA

Beste kontagailu honek 0tik 59ra zenbatuko du. Ikusten denez, egoera kritikoan, 59an alegia, kontagailua zerora pasarazten dugu.

```

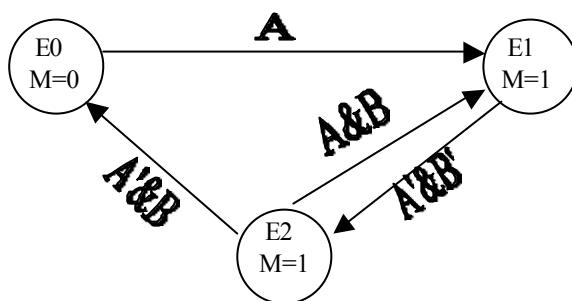
|PAL22V10 in:R, io:Q[5..0], clock:CLK
|
| Register: CLK // Q[5..0]
| map: Q[5..0] -> Q[5..0]
| {
|   n -> n+1, R' & Q[5..0] != 59
|   n -> 0, R # Q[5..0] == 59
| }
|
| Vectors:
| {
|   Display (CLK)c,(Q[5..0])d
|   Test R=1; CLK
|   Clear R
|   Test CLK=64(0,1)
|   End
| }

```

## 9.8. ZIRKUITU SEKUENTZIALAK

Jakina denez, zirkuitu digitalak konbinaziozkoak edo sekuentzialak izan daitezke. Konbinaziozko zirkuitu batean, irteerak sarreren arabera bakarrik aldatzen dira. Sekuentzialetan, berriz, irteeren balioak memorizatuta gelditzen dira, eta irteerak aldatzeko, bai sarrerak zein aurreko uneeko irteeren balioak eragina izango dute.

Zirkuitu sekuentzialei *egoera-makinak* ere deitzen zaie. Zirkuituak har dezakeen **egoera** bakoitzean gertaera bat memorizatzen du. Zirkuitu horiek egoera-diagramen bidez irudikatzen dira.



9.19. irudia

9.19. irudian agertzen den egoera-makinan bi sarrera daude, A eta B, eta irteera bakarra, M. Horiekin hiru egoera ezberdin adierazten ditugu. Egoerak memorizatzeko *PLD*ko baskulak erabiltzen dira; **n** baskulekin  $2^n$  egoera bereiz daitezke.

Makina horiek programatzeko, **prozeduren** bidezko metodoa daukagu. Ikus dezagun nola programatzen den aurreko adibidea.

```
|GAL16V8 in:(A,B,RESET), io:(M, Q[1..0]), clock:CLK
| Register: CLK // Q[1..0]
```

```
|Signature: "98/11/26"
>Title: "Adibide sekuentziala"
|Procedure: RESET, Q[1..0]
|{
|States: E0 = 0
|E0.      M = 0  A ?    -> E1
|          |           -> E0
|E1.      M = 1  A'B' ? -> E2
|          |           -> E0
|E2.      M = 1  A'B ?   -> E0
|          |           A'B ?   -> E1
|          |           |           -> E2
```

```

    }

|Vectors:
|{
| Display RESET, (CLK)c,A,B, " Egoera: ", Q[1..0], "Motorra:", M

|Set RESET
|Test CLK
|Clear RESET

|Set A
|Test CLK           |E1 egoerara pasatuko da eta M=1
|Set A,B = 00b
|Test CLK           |E2 egoerara
|Test A,B = 01b
|Test CLK           |E0 egoerara berriro eta M=0.
|End
|}

}

```

**“Procedure:**” hitza ipini eta gero, bi seinale-mota agertzen dira:

- ☞ Reset seinalea. Hau aktibatzen denean makina **0** egoerara joaten da.
- ☞ Bigarrenak dira egoera-seinaleak, eta egoerak deskodifikatzeko erabiltzen dira.

Gero giltzen artean egoeren arteko trantsizioak adierazten dira:

- ☞ Lehendabiziko egoera zaharra, atzean puntu bat jarri.
- ☞ Gero, egoera horri dagokion irteera-balioa.
- ☞ Ondoren, sarrerako baldintza galdera-marka batekin.
- ☞ Eta, bukatzeko, gezi baten ostean, zein egoeratara pasatu behar duen.

Vectors-en, set eta test komandoekin dauden trantsizio ezberdinak simulatuko ditugu.

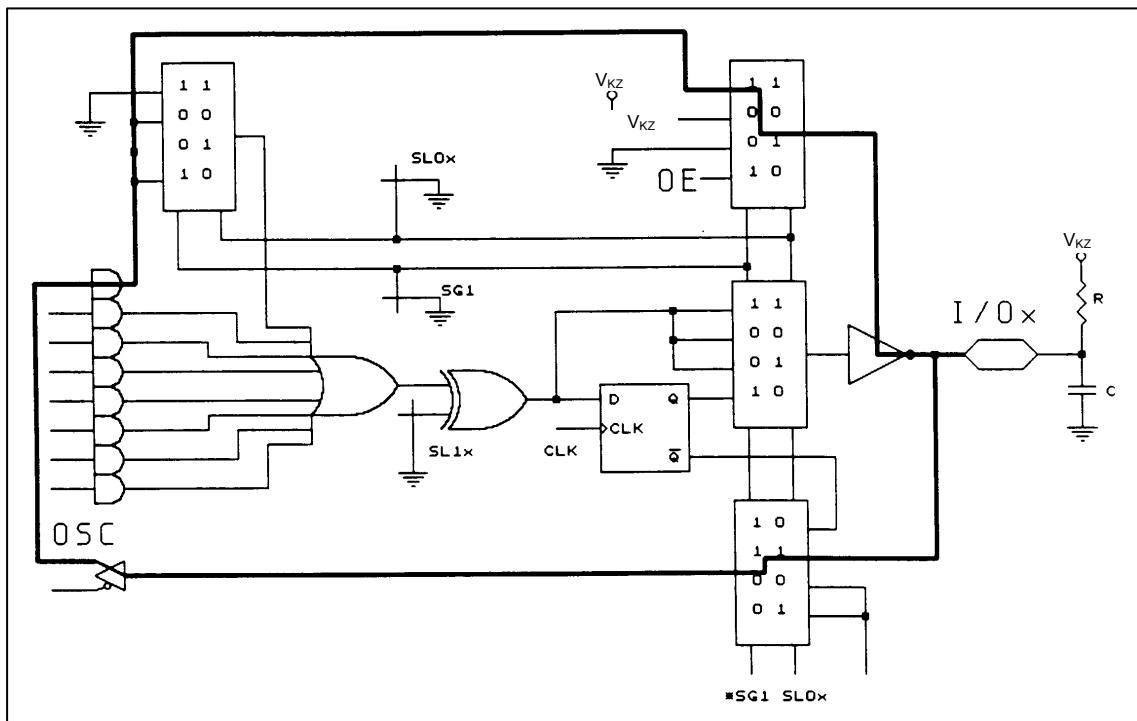
Kasu gehienetan, trantsizioak gertatu ahal izateko, erloju-pultsu bat behar dugu. Erloju-pultsu horiek kanpoko osziladore batetik lor daitezke, edo bestela, 16v8 PALen barruko makrozelula batean implementa daiteke, erresistentzia eta kondentsadorea kanpoan jarri. Programazioa erraza da:

```

|GAL16V8      io:(OSCI,OSCO)
|
|OSCI = OSCO ?? 0
|OSCO = OSCI

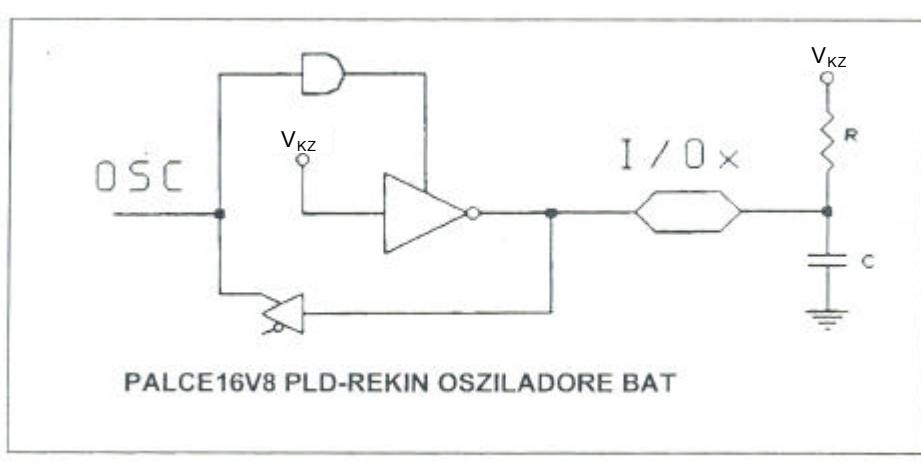
```

9.20. irudian, txipa programatu ondoren, makrozelula nola gelditzen den ikus daiteke. Marra beltzak erabiltzen den zirkuitu itxia irudikatzen du. Zirkuitu horretan bi ate daudenez, osziladorea bi hedapen-denboratan **1**ean egongo da ( $\approx 50$  ns).



9.20. irudia

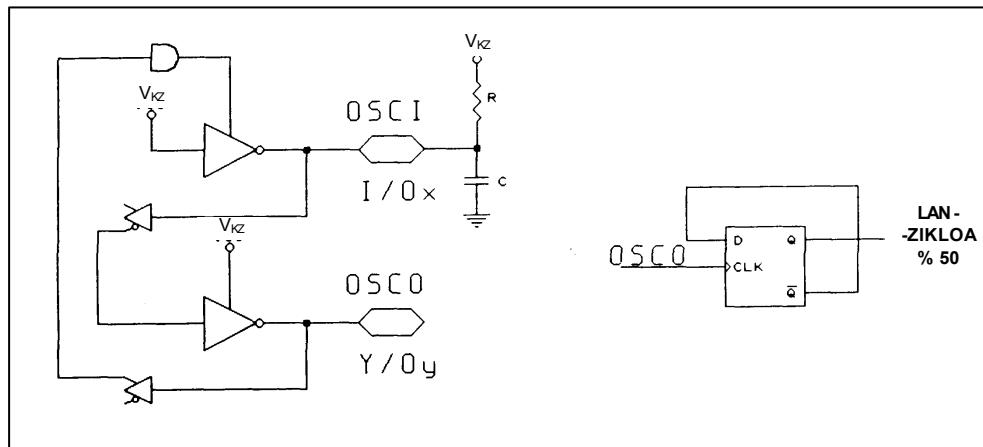
9.21. irudian zirkuitu bera laburtuta ikusten da:



9.21. irudia

Osziladorearen irteera beste makrozelula batetik egingo dugu. Programa horri dagozkion barruko konexioak 9.22. irudian irudikatzen dira.

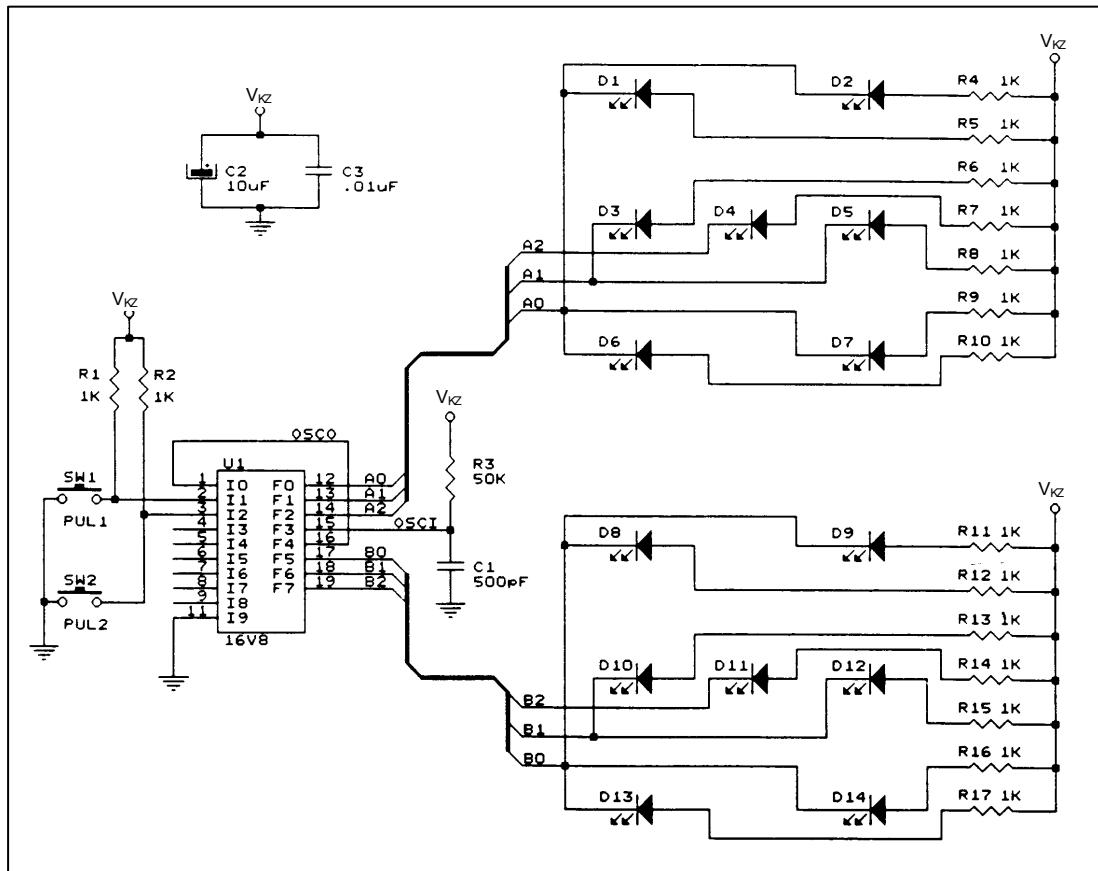
```
|GAL16V8      io:(OSCI,OSCO)
|OSCI = OSCO ?? 0
|OSCO = OSCI
```



9.22. irudia

### 9.8.1. ADIBIDE BAT PROZEDURAK ERABILIZ (BI DATU)

Programa honetan, *PAL16V8k* bi datu kontrolatzen ditu. 9.23. irudian adibidearen hardwarea ikusten dugu.



9.23. irudia

Hardwarea egin ondoren, *PAL* programatzeko softwarea hau da:

```

PAL16V8 BATEKIN BI DATU KONTROLATU

PAL16V8 4:Reset,
in: D[1..2], io: (B[2..0], OSCO, OSCI, A[2..0]), clock:CLK

Signature: "2 DATU"
Title: "Datu automatikoak"
Registers: CLK // A[2..0], B[2..0]

OSCI = OSCO ?? 0
OSCO = OSCI

Procedure: Reset, A[2..0] | 1 DATUAREN SEKUENTZIA:
{
states: BAT    = 011b,    BI     = 101b,
        HIRU   = 001b,    LAU    = 110b,
        BOST   = 010b,    SEI    = 100b,
        ZERO   = 000b,    ZAZPI  = 111b
BAT.    D1'? -> BOST
        -> BAT
BI.     D1'? -> LAU
        -> BI
HIRU.   D1'? -> SEI
        -> HIRU
LAU.    D1'? -> BAT
        -> LAU
BOST.   D1'? -> HIRU
        -> BOST
SEI.    D1'? -> BI
        -> SEI
ZERO.   -> BAT | Interesatzen ez zaizkigun 0 eta 7 egoerak
beste egoera batera pasarazikо ditugu inolako baldintzarik
ZAZPI.  -> BAT | jarri gabe.
}

Procedure: Reset, B[2..0] | 2 DATUREN SEKUENTZIA:
{
states: BAT    = 011b,    BI     = 101b,
        HIRU   = 001b,    LAU    = 110b,
        BOST   = 010b,    SEI    = 100b,
        ZERO   = 000b,    ZAZPI  = 111b
BAT.    D2'? -> HIRU
        -> BAT
BI.     D2'? -> BOST
        -> BI
HIRU.   D2'? -> SEI
        -> HIRU
LAU.    D2'? -> BI
        -> LAU
BOST.   D2'? -> BAT
        -> BOST

```

```

| SEI.      D2' ?   ->  LAU
|           ->  SEI
| ZERO.     ->  BAT      | Honetan ere 0 eta 7 egoerekin gauza
|           | bera egin behar da.
| ZAZPI.    ->  BAT
| }

| Vectors:
{
| Display CLK, " 1datu ", D1, A[2..0], " 2datu ", D2, B[2..0]
| Test Reset=1; CLK
| Clear Reset
| Test CLK = 6(0,1)
| Set D[1..2] = 10b
| Test CLK = 6(0,1)
| Set D[1..2] = 01b
| Test CLK = 6(0,1)
| Set D[1..2] = 11b
| Test CLK = 6(0,1)
| End
}

```

## 9.9. SIMULAZIO LOGIKOA

Zirkuitu bat programatu ondoren, ondo diseinatuta dagoen ala ez jakin behar dugu, nahi genuen funtzioa betetzen duen, alegia. OrCAD/PLD-k programa bat dauka horretarako: **VECTORS.EXE**. Horrek hurrengo bi funtzio hauek betetzen ditu:

1. Zirkuituaren funtzionamendua simulatu.
2. Test-bektoreak sortu. Programatzaila (EXPRO) egiaztapen fisikoa egiteko erabiliko ditu horiek.

Jakina denez, simulazioaren emaitzak **.LOG** lizapena duen fitxategi batean gordetzen dira. Gainera, **.JED** fitxategiari test-bektoreak gehitzen zaizkio.

### 9.9.1. SIMULAZIORAKO KOMANDOAK

Simulaziorako, **Vectors**: hitz berezia jartzen da, eta gero, giltzen artean, nahi ditugun komandoak.

#### 9.9.1.1. DISPLAY Komandoa

Komando honetan, ikusi nahi ditugun aldagaiak adierazi egin behar dira, baita zein formatutan agertuko diren ere. Komandoan honakoak jar daitezke:

1. Edozein kate komatxoen artean. Dagoen bezala ikusiko da.
2. Seinaleak bakarka edo multzoka. Seinale horien balioak ikusiko dira. Gainera, seinaleen balioak hainbat formatutan ikus daitezke:

- b Bitar formatuan.
- o Zortzitar formatuan.
- d Hamartarrean.
- s Formatu hamartarrean zeinuarekin.
- h Hamaseitarrean.
- L Balioak L/H formatuan ikusiko dira eta ez 1/0.
- c Kronograma bertikala agertuko da.

#### 9.9.1.2. ***SET*** Komandoa

Simulazioa hasterakoan, sarrera guztiak ***zeroan*** jartzen dira eta irteerak definitu uzten dira. ***SET*** komandoarekin, sarrera bati balio zehatz bat (0 edo 1) eman diezaiokegu.

Set A[4..1] = 0101b	A4=0, A3=1, A2=0, A1=1
Set B0 = 1	B0=1

#### 9.9.1.3. ***CLEAR*** Komandoa

Aurrekoaren antzekoa da, baina bakarrik zeroan jartzeko erabil daiteke:

Clear R	R=0
---------	-----

#### 9.9.1.4. ***TEST*** komandoa

Hau da komandorik garrantzitsuena. Komando honen atzetik sarrerako seinaleak jartzen dira. Eragiketa hauek egiten ditu:

1. Sarrerako konbinazio bakoitzari dagozkion irteerako balioak kalkulatzen ditu.
2. Balio horiek pantailan erakusten ditu.
3. **.JED** fitxategiari sortutako *test*-bektoreak gehitzen dizkio.

Test A1,B1
------------

Sarrerako lau konbinazio horientzat, irteerako balioak kalkulatzen ditu. Erloju-pultsuak sortzeko horrela egiten da:

Test CLK = 15(0,1)	Hamabost erloju-pultsu sortuko ditu
--------------------	-------------------------------------

Set komandoaren funtzioa ere egin dezake. Beraz, hurrengo bi espresio hauek baliokideak dira:

| Set R  
| Test CLK

eta

| Test R=1; CLK

#### 9.9.1.5. *RIPPLE* komandoa

Normalean, simulagailuak ez ditu emaitzak ematen seinaleak egonkortu arte. Komando honekin, seinaleak aldatzen ari direla ikus ditzakegu. Hiru aukera daude (lehendabiziko biak baliokideak dira).

| Ripple  
| Ripple on  
| Ripple off

#### 9.9.1.6. *QUIT* eta *END* komandoak

Biek simulazioa bukatutzat emateko balio dute. *QUIT* erabiliz gero, test-bektoreak ez dira gordetzen, *END* komandoarekin bukatzen badugu, berriz, bai.

### 9.10. KONPILADOREAREN HITZ BEREZIAK ETA FUNTZIOAK

---

Hitz bereziek esanahi berezia dute *OrCAD/PLD* konpiladorearentzako, bi puntuz bereizten dira, eta ezin dira erabili seinale-izenak jartzeko. Hauek dira konpiladore honen hitz bereziak:

Active-high:	Fuse:	Netlist:	Register:	Table:
Active-low:	Fuses:	Out:	Reset:	Title:
Clock:	In:	Part:	Signature:	Type:
Conditioning:	Io:	Preset:	State:	Types:
Configuration:	Library:	Procedure:	States:	Value:
Enable:	Map:	Register:	Stream:	Vectors:

Funtzioak konpiladoreak dituen azpiprogramak dira. Oso lagungarriak dira funtziobatzuk definitzeko. Hurrengo hauek dira *OrCAD/PLD*ek dituenak:

<b>Abs(a)</b>	<b>Iexp(p,q,a)</b>	<b>Min(a)</b>
<b>Bit(n,a)</b>	<b>Iln(p,q,a)</b>	<b>Ones(a)</b>
<b>Bitcount(a)</b>	<b>Ilog2(a)</b>	<b>Parity(a)</b>
<b>Dff(a)</b>	<b>Iscale(p,q,a)</b>	<b>Sevenseg(a)</b>
<b>Gray(a)</b>	<b>Isin(p,q,a)</b>	<b>Sqrt(a)</b>
<b>Icos(p,q,a)</b>	<b>Itan(p,q,a)</b>	<b>Tff(a)</b>
<b>Icot(p,q,a)</b>	<b>Max(a)</b>	<b>Zeros(a)</b>

## 9.11. ESKEMEN BIDEZ LOGIKA DEFINITU

Eskemen bitartez ere logika defini daiteke *OrCAD/SDT* moduluarekin. Horretarako, bi irizpide hauek errespetatu behar dira:

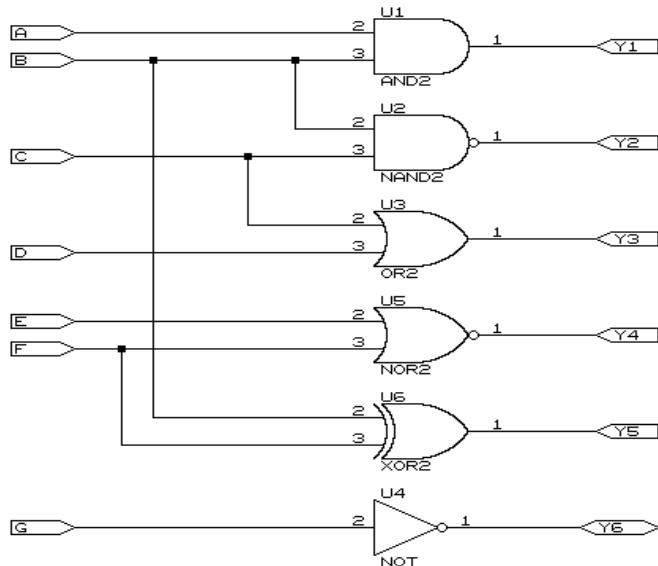
1. Ateak, zirkuitu integratuak eta abar **PLDGATES.LIB** liburutegitik hartu behar dira. Ez beste inondik.
2. Sarrerako eta irteerako seinaleak **module ports**-en bidez adierazten dira.

## 9.12. PROZESA

Adibide baten bidez (hasieran erabilitakoa, ateak.pld), pauso pauso ikusiko dugu prozesua zein den.

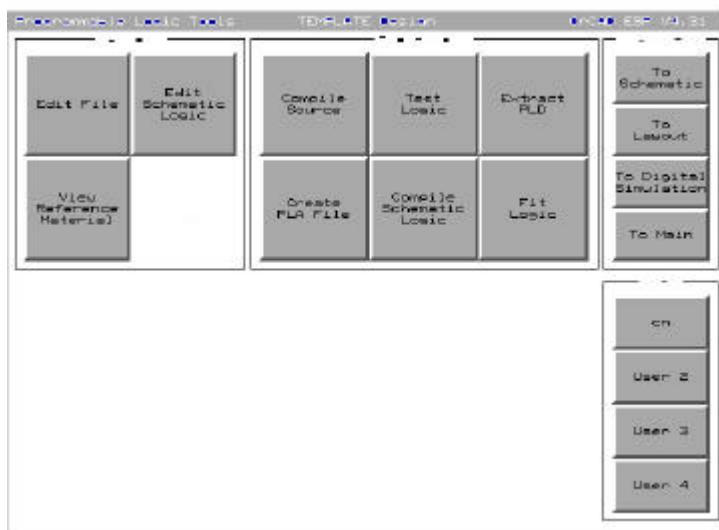
### 1. Eskema egin

*PLD* moduluan, **Edit Schematic Logic** aukeratuko dugu, eta eskema egin eta gero, **ATEAK.SCH** izenarekin gordeko dugu.



9.24. irudia

9.25. irudian *OrCAD/PLD*ren pantaila ikus daiteke.



9.25. irudia

## 2. Eskema konpilatu

Honen bidez, konexio-zerrenda lortzen da eta **ATEAK.NET** fitxategian gordetzen da. Eskema konpilatzeko, “Compile Schematic Logic” botoia aukeratu, eta Local Configuration programatuko dugu horrela:

Tresnak	Iturburua	Helburua	Abisuari kasurik ez
CLEANUP	ATEAK.SCH		ON
ANNOTATE	ATEAK.SCH		ON
INET	ATEAK.SCH		ON
ILINK	ATEAK.INF		ON
IFORM	ATEAK	ATEAK.NET	ON

**IFORM** aukera hautatu behar da, *netlist* formatua, kasu honetan **PLDNET.CCF**.

Hori egin eta gero, **EXECUTE** aukeratu eta eskema konpilatuko da. Fitxategi batzuk sortuko dira horien artean **ATEAK.NET**.

### 3. ATEAK.PLD editatu

**EDIT FILE** botoia sakatuz, fitxategi hau sortuko dugu. Fitxategi horretan, sarrerako eta irteerako seinaleak definituko ditugu (eskeman definitutakoekin bat etorri behar dute). Ondoren, lerro hau jarriko da:

| =>Ateak.net

Bukatzeko, *vectors*-en komandoak jartzen dira. Hona hemen adibide bat:

```
|PAL12H6      in:(A, B, C, D, E, F, G),  out:Y[1..6]
|
|=>ateak.net

| Vectors:
| {
| display A, B, C, " A & B = ", Y1 , " (B & C)' = ", Y5
| test A, B, C
| end
| }
```

Fitxategi hau besteak bezala konpilatu eta simulatu egiten da.

## Ariketa ebatziak

### 1. PRAKTIKA

Bi biteko bi zenbaki (A1,A0 eta B1,B0) batutzaile/kentzaile egin nahi dugu. KEN sarrera bat izango du. Sarrera hori “1” bada, A-B eragiketa egingo du; “0” bada, berriz, A+B egingo du. Emaitzak emateko hiru irteera izango ditu S2,S1,S0.

```
| PAL22V10 In:(A[1..0], B[1..0],KEN), Io: S[3..0]
|
| S[3..0]=((A[1..0]+B[1..0])&KEN) #((A[1..0]-B[1..0])&KEN')
| Vectors:
| {
| Display (A[1..0])d,(B[1..0])d,KEN, "    ",(S[3..0])s
| Test   KEN,A[1..0],B[1..0]
| End
| }
```

### 2. PRAKTIKA

Zortzi kanaleko multiplexadore bat lortu nahi dugu ekuazio indexatuen bidez.

Datu-sarrerak .....: I[ 0..7 ]  
Kontrol-sarrerak ....: S [ 0..2 ]  
Irteera .....: D

```
| GAL22V10 in:I[0..7], io:(S[2..0],D)
|
| n=0..7: D=I[n]&S[2..0]==n
|
| Vectors:
| {
| Display I[0..7],(S[2..0])d," ---> ",D
| Test   I[0..7],S[2..0]
| End
| }
```

### 3. PRAKTIKA

Zortzi biteko zenbakien paritate-sortzaile bat programatu *parity(n)* funtzioa erabiliz. Funtzio horrek “0” itzultzen du n zenbakian batekoen kopurua bikoitia bada, eta “1” itzultzen du bakoitia bada.

```
|GAL22V10 in:A[0..3], io:B[0..3]
|Map: A[0..3]->B[0..3]
|{
```

```
| n->parity(n)
|}
|Vectors:
|{
|Display A[0..3]," ",B[0..3]
|Test A[0..3]
|End
|}
```

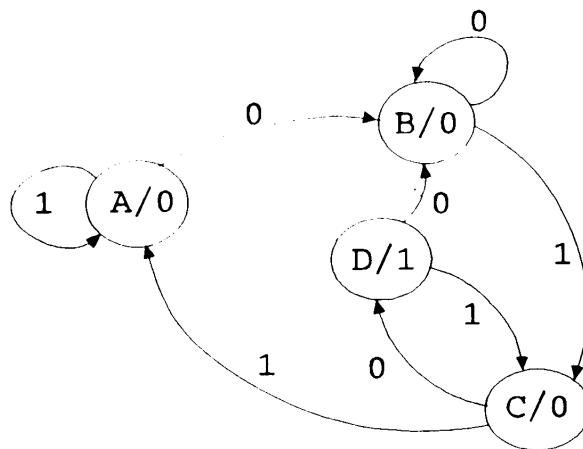
#### **4. PRAKTIKA**

Egia-taulak erabiliz, 8-3 lehentasunarekin kodetzailea diseinatu. *Enable* sarrera ere edukiko du.

```
|GAL16V8  in:(ST,A[7..0]),io:S[2..0]
|
|Table:ST,A[7..0]->S[2..0]
|{
|  100000000B->000
|  100000001B->000
|  10000001XB->001
|  1000001XXB->010
|  100001XXXB->011
|  10001XXXXB->100
|  1001XXXXXB->101
|  101XXXXXXB->110
|  11XXXXXXXB->111
|  0XXXXXXXXB->000
|}
|Vectors:
|
|{
|Display ST,(A[7..0])B,"->",(S[2..0])B
|Test ST,A[7..0]
|End
|}
```

## 5. PRAKTIKA

Diagrama honetatik programaziora pasatu.



```

|GAL16V8 in:(ENT,RESET), io:(S,Q[1..0]),clock:CLK
|
|Register: CLK//Q[1..0]
|
|Procedure:RESET,Q[1..0]
|{
| State: A=0
|
| A. S=0
|     ENT'?->B
|     ->A
|
| B. S=0
|     ENT?->C
|     ->B
|
| C. S=0
|     ENT?->A
|     ->D
|
| D. S=1
|     ENT?->C
|     ENT'?->B
| }
|
|Vectors:
|{
|Display (CLK)C,RESET,ENT,(Q[1..0])D,S
|Set RESET
|Test CLK
  
```

```
|Clear    RESET
|
|Test    ENT=0;CLK=2(0,1)
|Test    ENT=1;CLK
|Test    ENT=0;CLK=3(0,1)
|Test    ENT=1;CLK
|Test    ENT=0;CLK
|Test    ENT=1;CLK=3(0,1)
|End
|}
```

## ***Proposatutako ariketak***

### **1. PRAKTIKA**

**Ekuazio boolearrak** erabiliz, hurrengo funtziotan hauek egingo dituen *PLD*a programatu.

$$S_0 = (A \oplus B) + D$$

$$S_1 = \overline{BC + D}$$

$$S_2 = \overline{A} + B$$

$$S_3 = S_0 \oplus S_1 \oplus S_2$$

### **2. PRAKTIKA**

Kode bitarretik gray kodera pasatuko duen bihurgailua egin nahi dugu. Lau biteko zenbakiekin egingo da lana. Lehendabizi egia-taula egitea komeni da.

### **3. PRAKTIKA**

**Ekuazio boolearrak** erabiliz, 3-8 deskodetzailea programatu. Irteerak zeroan aktibatuko dira, eta STROBE seinalea ere izango du.

### **4. PRAKTIKA**

Aurreko praktikan egindako zirkuitu bera programatu, baina ekuazio indexatuak erabiliz.

### **5. PRAKTIKA**

Lau biteko bi zenbakien ( $A[3..0]$  eta  $B[3..0]$ ) konparatzzailea programatu. Hiru irteerak  $A=B$ ,  $A>B$ , eta  $A<B$ , 15,14 eta 16 pinetan, hurrenez hurren, nahi ditugu.

### **6. PRAKTIKA**

Zeinua aldatzen duen zirkuitua diseinatu. Sarreretan 8 biteko zenbaki bat hartuko du eta irteeretan zenbaki bera baina negatiboa utziko du. Beti bezala zenbaki negatiboak biko osagarrian adieraziko dira.

### **7. PRAKTIKA**

Bigarren praktikan egindako bihurgailua taulekin diseinatu. Irteera bakoitzeko taula bat egin (4 taula guztira), eta taula bakoitzean sarrerako konbinazioak adierazteko era ezberdina erabili.

## **8. PRAKTIKA**

Taulak erabiliz, BCDtik zazpi segmentuko deskodetzaile bat diseinatu.

## **9. PRAKTIKA**

Aurreko praktikan egindako deskodetzaile bera egin, baina, kasu honetan, **mapak** eta **sevenseg(n)** funtzioa erabiliz.

## **10. PRAKTIKA**

Bigarren praktikako bihurgailua **stream**-en bidez egin.

## **11. PRAKTIKA**

**Stream**-en bidezko metodoa erabiliz, 8-3 lehentasuneko kodetzailea diseinatu. Enable sarrera ere edukiko du.

## **12. PRAKTIKA**

Bigarren praktikan egindako zirkuitua errepikatu, baina kasu honetan **gray(n)** funtzioa erabiliz.

## **13. PRAKTIKA**

Kontagailu hamartar bat UP/DOWN diseinatu nahi dugu. Hurrengo pin hauek izango ditu:

RESET .. kontagailua zeroan jarriko du

UP .....: batean badago, gorantz kontatuko du; bestela, beherantz.

CLK .....: Erloju-sarrera

Q [ 3..0 ] ..: Irteerak

## **14. PRAKTIKA**

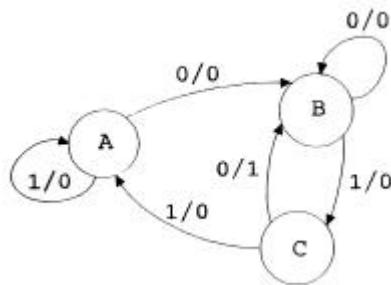
Hotel bateko igogailurako kontagailua egin nahi da. Jakina denez, hotel gehienetan, sineskeria dela eta, ez dago hamahiru zenbakia daraman pisurik eta hamabigarrenetik hamalaugarrenera pasatzen da. Gure kasuan, zerotik hamalauraino kontatu beharko du, gorantz eta beherantz noski.

## **15. PRAKTIKA**

Mapak erabiliz, gorantz binaka eta beherantz banaka zenbatzen duen kontagailua diseinatu.

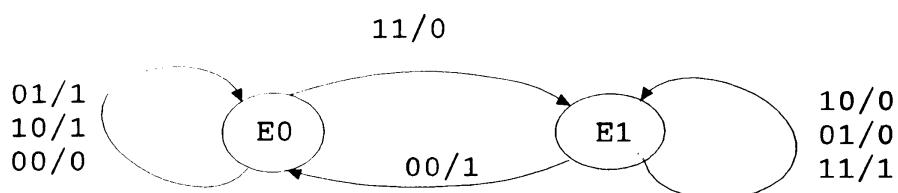
## **16. PRAKTIKA**

Diagrama honetatik, prozedurak erabiliz, programaziora pasatu.



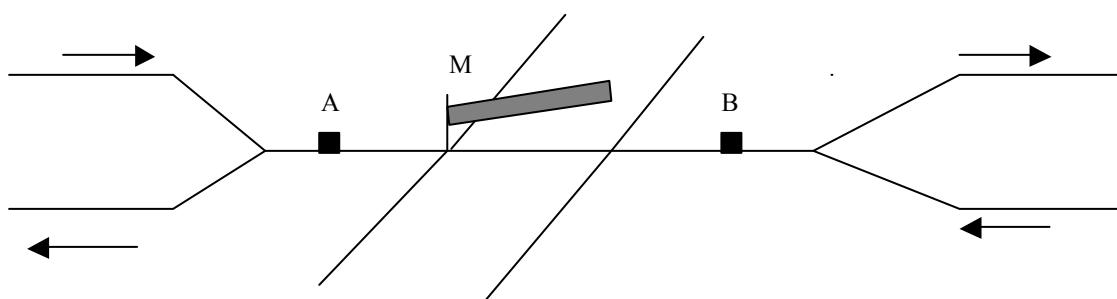
## **17. PRAKTIKA**

Diagrama honetatik programaziora pasatu.



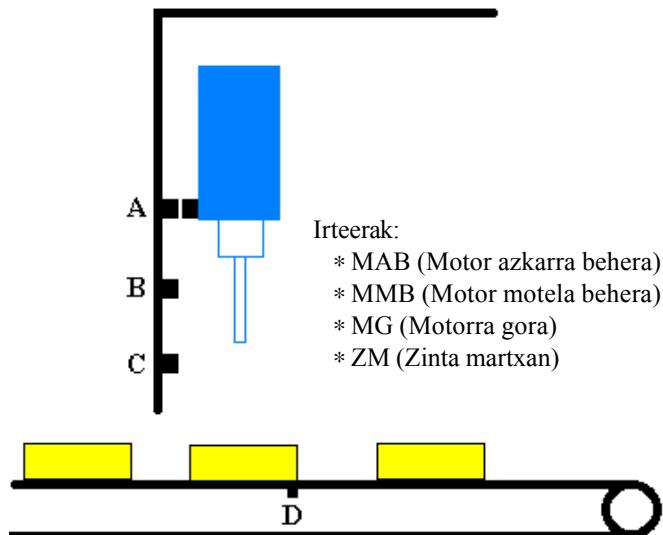
## **18. PRAKTIKA**

Trenbide-barrera automatikoa programatu. Trenak bi aldetatik eter daitezke.



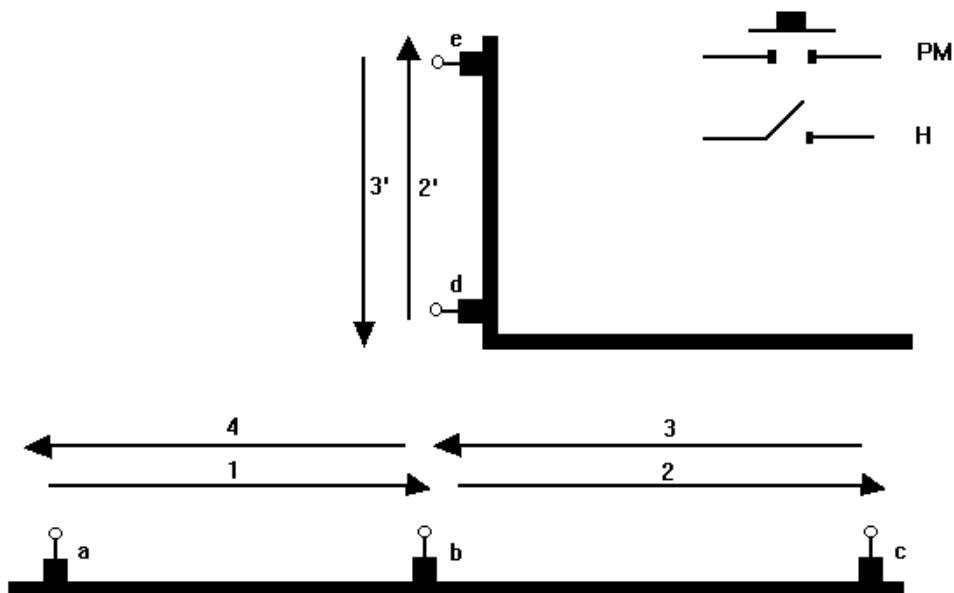
## **19. PRAKTIKA**

Irudian agertzen den daratuluaren funtzionamendua automatizatu *PLD* baten bidez.



## **20. PRAKTIKA**

Egin automatismo hau *PLD* batekin. **H** etengailua batean badago, 1-2-3-4 sekuentzia egingo du; **H** zeroan badago, berriz, 1-2'-3'-4 egingo du. Bi kasuetan, zikloa bukatzean gelditu egingo da, berriro **PM** sakatu arte.



## **21. PRAKTIKA**

Bigarren praktikan egindako bihurgailua eskemekin egin.

## **22. PRAKTIKA**

Suposa dezagun sistema batek aldi berean hainbat gauza egin behar dituela, adibidez,  $\mu$ P batek 10 *display* aldi berean kontrolatzen dituela ataka bakar baten bidez. *Display*-ak *enable* seinale batez kontrolatzen dira.  $\mu$ Pak *display* bakoitzari dagokion datua aterako du erloju-pultsu batekin.

Erloju-seinalean oinarritzen den zirkuitua egin behar dugu, *display*-ak piztu eta itzali egingo duena.

## ***Bibliografía***

Logica Programable. Ejercicios resueltos con *OrCAD/PLD*  
Mariano Barrón Ruiz, Mc Graw Hill

# 10. KAPITULUA

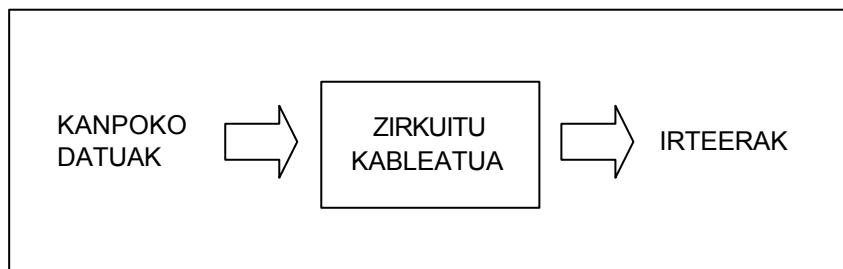
## Mikroprozesadoreak

### 10.1. ZIRKUITU PROGRAMAGARRIAK

#### 10.1.1. LOGIKA KABLEATUA

XX. mendearren erdialdera arte, prozesu industrialak kontrolatzeko erabiltzen ziren zirkuitu elektronikoak osagai analogikoekin egiten ziren (transistore operazionalak...). Geroztik, elektronika digitalaren aurrerapena dela eta, osagai logikoak erabiltzen hasi ziren (ate logikoak, baskulak...).

Zirkuitu hauek eginkizun bakarra egiteko gai dira, eta kontrolean zerbait aldatu behar bada, berriro zirkuitu osoa aldatu behar da.



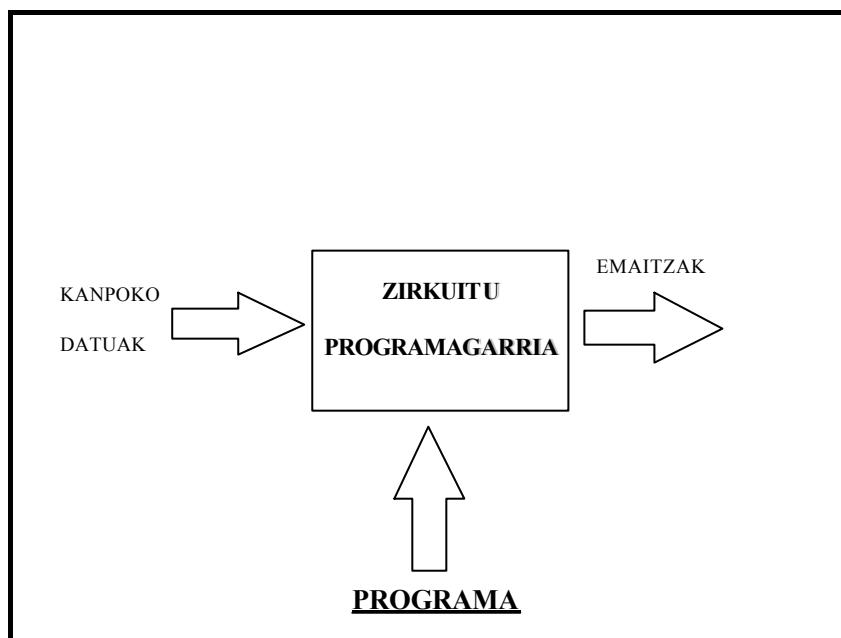
Horrez gain, ez dira oso fidagarriak, osagai asko eramatzen dituztelako, eta zirkuituak konplikatu samarrak direlako. Diseinua ere ez da erraza kontrola konplikatzen denean.

Beraz, eginkizun errazentzat egokiak izan daitezke, baina egin beharrekoa konplikatuz gero, hobe izango dugu beste zirkuitu batzuk egitea, zirkuitu programagarriak hain zuzen ere.

### 10.1.2. ZIRKUITU PROGRAMAGARRIAK

Arazo horiek konpontzeko sortu ziren zirkuitu elektroniko PROGRAMAGARRIAk, besteak beste mikroprozesadoreak, mikrokontroladoreak, PLDak eta abar... Zirkuitu horiek, funtzio bat baino gehiago egin ditzakete, fisikoki ezer aldatu gabe. Une bakoitzean egin behar duena jakiteko, aginduak eman behar dizkiogu. Agindu-sekuentzia bati programa deituko diogu.

Makina hauetan, gauza fisikoei (eskuekin uki daitezkeenak) hardware deitzen zaie, eta programei software.



Zirkuitu hauen artean aurrerapen gehien ekarri diguna mikroprozesadorea izan da. Horri esker beste arlo berri bat sortu baita, informatika hain zuen. Jakina denez, ordenagailuaren zerebroa mikroprozesadorea da. Horiek normalean (eta gero eta gehiago) oso ahaltsuak dira, esate baterako, *INTEL* etxeko 80486, PENTIUM P6...

Beraz, lehenengo μPak oso txikiak geratu dira, eta aspalditik ez dira honetarako erabiltzen. Baino prozesu industrial gehienak kontrolatzeko, oso egokiak suertatu dira, eta logika kableatua ordezkatu dute.

## 10.2. MIKROPROZESADOREAK

---

### 10.2.1. FUNTSEZKO EGITURA

Hiru bloke handitan bana dezakegu:

- ❑ **Sarrera / irteerako elementuak.** Gailu hauekin, makina kanpoko munduarekin erlazionatzen da, eta informazioa hartu edo ematen du.
- ❑ **Memoriak.** Zirkuitu elektroniko hauek datu bitarrak gordetzen dituzte. Jakina denez,  $\mu$ Pak 0 eta 1 baino ez du ulertzen. Memoria hauetan datuak zein programak gordetzen dira. Beraz, aginduak (programak) ere bitarrean eman behar zaizkio.
- ❑ **PUZ.** Zati honetan, aginduak interpretatzen ditu, eta horiek betetzeko egin beharreko ekintza guztiak antolatu eta martxan jartzen ditu. PUZ honen barruan beste hiru gauza bereiziko ditugu:
  - ❑ **UAL.** Unitate aritmetiko eta logikoa. Hemen eragiketa guztiak, bai aritmetikoak bai logikoak, egiten dira.
  - ❑ **ERREGISTROAK.** Hauek datuak gordetzen dituzte, normalean eragiketetan erabiltzen direnak, baita emaitzak ere.
  - ❑ **KU.** Kontrol-unitatea. PUZaren zerebroa.

### 10.2.2. MEMORIA

Memoria batek datuak zein programak gordetzen ditu. Programak ere, azken finean, 0 eta 1 erabiltzen du. Gelaxkaz osaturik dago; gelaxka bakoitzak, helbide bat dauka (izena bailitzan) besteekin ez nahasteko eta **BYTE** bat gordetzen du.

- ❑ **BYTEA** (8 biteko multzoa)
- ❑ **HITZA** (16 biteko multzoa)
- ❑ **HITZ BIKOITZA** (32 biteko multzoa)

MEMORIA								
HELBIDEAK	DATUAK							
	D0	D1	D2	D3	D4	D5	D6	D7
0000H	1	0	0	1	0	1	0	1
0001H	1	1	1	1	0	0	1	0
0002H	1	1	0	0	1	0	1	0
0003H	0	0	0	0	0	0	0	0
•	•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•	•
FFFFDH	1	1	0	1	1	0	1	0
FFFEH	0	1	1	0	0	1	1	0
FFFFH	1	1	1	1	0	0	0	0

10.1. taula

10.1. taulan memoria baten egitura ikus daiteke.

Kasu honetan, gelaxka bakoitzak 8 bit gordetzen ditu, eta 16 bitekin ematen ditugu helbideak, beraz,  $65.536 \approx 64K$  posizio ezberdin izango ditugu.

### 10.2.3. BUSAK

Mikroprozesadorea beste txipekin (memoriak, sarrera/irteera elementuak...) komunikatzen da, hari elektrikoaren bidez. Hari horiek hiru multzotan banatzen dira, bere funtziaren arabera. Hari-multzo horiei **BUS** deitzen zaie. Mikroprozesadore-sistema batean hiru BUS ditugu:

-  **HELBIDE-BUSA**
-  **DATU-BUSA**
-  **KONTROL-BUSA**

**HELBIDE-BUSAk** noranzko bakarra dauka eta  $\mu$ Ptik ateratzen da. Honen bidez,  $\mu$ Pak memorien helbide guztiak kontrola ditzake.

**DATU-BUSAk** datuak bi noranzkotan eraman ditzake. Izan ere, datuak sartu edo atera egin daitezke  $\mu$ Ptik.

**KONTROL-BUSA.** Multzo honetan, kontrolatzeko balio duten hariak kokatzen ditugu. Batzuk noranzko bakarrekoak eta beste batzuk bikoak.

## 10.2.4. PROGRAMAK

Agindu-multzo batek (bata bestearen atzetik), ordena batir jarraituz, programa bat osatzen du. Horri, lehen esan dugun bezala, **SOFTWARE** esaten zaio.

### 10.2.4.1. Aginduak

$\mu$ P bakoitzak bere **agindu-sorta** dauka. Agindu horiek, memoriatik mikrora datu-busetik joaten dira. Guk ikusiko ditugun mikroen datu-busak 8 hariz osaturik daude, edo, beste era batera esanda, 8 biteko busak dira.

Beraz, 8 bitekin  $2^8 = 256$  konbinazio ezberdin lor ditzakegu. Horrek esan nahi du gehienez 256 agindu ezberdin onartuko dituela.

Aginduak kode bitarrean edo hamaseitarrean adierazten dira **makina-lengoain**.

### 10.2.4.2. Programazioa

Gure hizkuntza eta  $\mu$ Paren lengoia guztiz ezberdinak dira. Mikroari aginduak bere lengoian eman behar dizkiogu, hain zuzen, zeroak eta batak erabiliz. Guretzat hori oso neketsua izateaz gain, hanka sartzea oso erraza da. Hori gerta ez dadin, programatzeko hitzen laburdurak edo **nemonikoak** erabiliko ditugu.

Agindu bakoitzak bere nemonikoa du. Nemoniko horiek ingelesez daude, eta, normalean, egiten duten funtzioa adierazten dute. Adibidez, batuketa aginduari dagokion nemonikoa **ADD** da, ingelesezko *adder* hitzetik.

Datuak adierazteko, berri, sistema hamaseitarra erabiliko dugu, bitarra baino askoz erosagoa delako.

AGINDUAK	NEMONIKOA	MAKINA-LENGOIAIA	HAMASEITARRA
BATU	ADD	1111 0000	F0
KENDU	SUB	1010 1010	AA
GEHITU	INR	1111 0011	F3
GORDE	STA	0010 1111	2F
HARTU	LDA	0111 0011	73

Programatzeko nemonikoak erabiliko ditugu. Horretarako, mikroaren **agindu-sorta** zein den jakin behar dugu, gogora dezagun mikro bakoitzak berea duela. Bainak mikroak programatzeko hainbat lengoia daude.

### 10.2.4.3. Programatzeko lengoaiak

Programatzeko lengoaia-pila bat dago. Bakoitzak bere ezaugarriak ditu eta, normalean, arlo ezberdinetara moldatzen dira. Bi bloketan bana ditzakegu:

#### BEHE-MAILAKOAK

- Mihiztzailea
- C

#### GOI-MAILAKOAK

- Basic
- Cobol
- Fortran
- Pascal
- C

Lengoaia horiek guztiak espresio ezberdinak erabiltzen dituzte eragiketak adierazteko, baina azkenean mikroak bete behar ditu aipatutako eragiketak edo funtzioak. Horrek esan nahi du, edozein delarik ere erabilitako hizkuntza, itzuli egin behar dela makina-lengoaiara.

Itzulpen hori guk egin dezakegu eskuz, baina badaude itzulpen hori egiteko gauza diren programa informatikoak. Besteak beste, mihiztzaileak, konpiladoreak, estekatzaileak...

Guk egindako programari **iturburu-programa** deitzen zaio. Konpiladoreak programa hori **programa exekutagarri** bihurtzen du.

#### ITURBURU-PROGRAMA

```

ADD   B
MVI   A,4CH
STA   1200H
MOV   D,C
    
```

#### ITZULTZAILEAK

MIHIZTZAILEA  
KONPILADOREA  
ESTEKATZAILEA

#### MAKINA-LENGOAIA

```

1101 1111
0111 0011
0010 1101
0011 1110
1010 0101
1101 0010
    
```

### 10.2.5. SISTEMA OSOA

Mikroprozesadore batek berak bakarrik ezin du ezer egin, beste zirkuitu batzuk (periferikoak) behar ditu. Gutxienez hauek beharko lituzke:

- *EPROM Memoria* programak gordetzeko.
- *RAM Memoria* datuak gordetzeko.
- *I/O periferikoak* kanpoko aldagaietkin komunikatzeko.
- *UART* komunikazio-seriea egiteko.
- *TIMERak* temporalizazioak egiteko.
- *A/D eta D/A* eta abar.

### 10.3. MIKROPROZESADORE-FAMILIAK

---

Intel etxeak 1971. urtean lau biteko datu-busa zuen 4004a atera zuen. Gero 8 biteko 8080a eta 8085a irten ziren; 64 kbyteko memoria helbidera zezaketen 16 lerroko helbide-busaz.

Gero, 1978. urtean, **X86** deituriko arkitekturako mikroprozesadoreak atera zituzten. Lehen belaunaldian 8086a, 8088a eta 80186a izan ziren. Gero 80286a, 80386a eta 80486a. Pentiuma izan da Intelen bosgarren belaunaldiko mikroprozesadorea.

**Pentiuma** 1993an agertu zen. 32 biteko helbide-busa eta 64 biteko datu-busa ditu. Bestalde, 8 kbyteko cache memoria bi ditu (memoria hori prozesadorearen eta memoria nagusiaren artean kokatzen da, memoriarako atzipena askoz azkarragoa izanik). 166 MHz-ean lan egin dezake.

Motorola etxeak 6800a atera zuen 1975. urtean, gero 6802, 6803 eta 6809a atera zituen. 68000a izan genuen 16 biteko datu-busaz eta 24 biteko helbide-busaz.

68020, 68030, 68040, 68060 eta azkenik **PowerPC** agertu dira. Azken hori 64 biteko datu-busa, 32 kbyteko cache memoria eta RISC erako mikroprozesadorea da.

### 10.4. ZORTZI BITEKO MIKROPROZESADOREAREN EGITURA

---

8 biteko mikroprozesadore guztien funtzionamendua eta barne-egitura oso antzekoak dira. Irizpide hori kontuan hartuta, INTELen 8085 mikroprozesadoreea aztertzea erabaki da, bibliografia eta dokumentazio tekniko ugaria duelako.

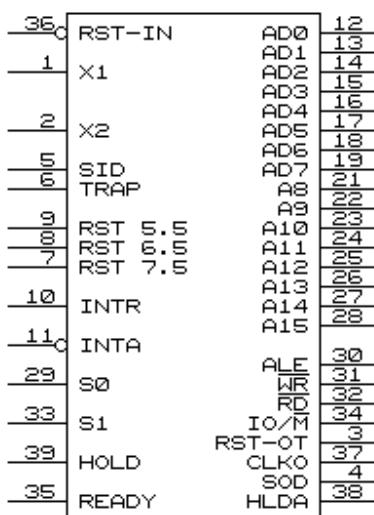
8085 mikroprozesadoreea 8 biteko hitz batekin diharduen gailua da eta, lehendik dakigun bezala, bere 16 lerroekin 64 K memoria-posizioraino helbideratzeko ahalmena dauka. NMOS teknologian egina dago eta 6.200 transistore dauzka. 40 adartxo dituen *dual in line* kapsula bat da zirkuitu integratu hori.

Kontroleko programatik agintzen zaizkion eragiketak egiteko, 8085 mikroprozesadoreak barnean 74 agindu-mota ezberdin deskodetzeko ahalmena dauka. Hartara, zenbaki horrexek ematen du agindu-sorta.

Elikatze-tentsioa bakarra da, KZ-eko 5 voltekoa.  $V_{KZ}$  (+) eta  $V_{SS}$  (masa) terminaletan ezartzen da. Sistema digital programagarri bat eratzen duen edozein mikroprozesadorek bezala, erloju edo osziladore bat behar du sistemak egiten dituen eragiketak sinkronizatzeko. Kasu honetan, erlojua txipean bertan bilduta dator eta 3,125 MHz-eko maiztasuneko seinale digital bat sortzen du.

## 10.5. 8085aren TERMINALEN IZENAK

10.1. irudian azaltzen da 8085 mikroprozesadorearen sinbolo logikoa, bere berrogei terminalen izenekin. Terminal horiexen bidez bidaltzen eta jasotzen ditu mikroprozesadoreak seinale digitalak. Ondoren, haietako bakoitzari dagokion eginkizuna azalduko dugu. Dena hobeto ulertzeko, batzuetan terminal gisa aipatuko ditugu eta beste batzuetan, berriz, seinale gisa. Kasu guztietan, sarrera ala irteera (ala biak) den (diren) aipatuko dugu.



10.1. irudia

A8-A15. Mikroprozesadorearen sistemako helbide-buseko 8 bit garrantzi-tsuenen irteerak dira.

AD0-AD7. Sarrera-irteera diren terminal-multzo honek bi eginkizun dauzka. Lehenengo, multiplexadore baten bidez, helbide-buseko pisu gutxien duten 8 bitekin komunikatzen da mikroprozesadorea eta, gero, datuen noranzko biko busarekin.

ALE. *Adress Latch Enable* (helbideen latch biegonkorren aktibazioa). AD0-AD7 lerroetan dagoen informazioa datu bat edo helbide bat buruzkoa den igartzen duen irteera bat da. Helbide baten behoko aldea memorian gordetzen duen erregistro bat desarratzeko erabiltzen da.

SO-S1. Datu-busaren egoeraren berri ematen duten irteerak dira. Mikroprozesadorea egiten ari den ekintza-mota adierazten du: bilaketa, irakurketa, idazketa edo geldialdia.

RD .Read. Hautatu den memoria-posizioa edo S/I gailua PUZak irakurri egin nahi duela eta datu-busa transferentzia egiteko prest dagoela adierazteko erabiltzen den irteera.

WR .Write. Datu-busean dagoen informazioa helbideraturiko memoria-posizioan edo S/I gailuan idatzi behar dela adierazten duen irteera.

READY. Maila altuan dagoenean, datu-busari buruz memoriatik edo S/I gailuetatik datozen datu baliagarriak daudela mikroprozesadoreari adierazten dion sarrera. Maila baxuan dagoenean, mikroprozesadorea zain geratzen da. Seinale hauxe erabiltzen da sistemako gailuren batek mikroprozesadorearen erlojuak finkaturiko denbora baino gehiago hartzen duenean.

HOLD. Beste prozesadore edo kontroladore batek helbide- eta datu-busak erabiltzeko baimena eskatzen duela adierazten duen sarrera. Mikroprozesadoreak utzi egiten ditu busak, egiten ari zen eginkizuna bete ondoren, eta inpendantzia handiko egoeran jartzen ditu honako sarrera eta irteera hauek: A8-A15, AD0-AD7, RD, WR eta IO/M.

HLDA. HOLD sarrerari erantzunez, mikroprozesadoreak sortzen duen irteera-seinalea, busak eta gainerako terminalak inpendantzia handiko egoeran daudela adierazteko.

INTR. *Interrupt request* (eteteko eskaera). Sarrera-seinale hau kanpoko gailuek erabiltzen dute mikroprozesadorearen berehalako arreta behar dutenean, eta programaren sekuentzia eteten dute. Etenaldi-eskaera ez onartzea ere gerta daiteke. Etenaldia onartzea edo ez onartzea programa nagusitik kontrolatzen da.

INTA. *Interrupt acknowledge* (etenaldiaren aitormena). Irteera-seinale bat da. Behin etenaldia onartu denean, etena eragin duen gailua aktibatzeko erabiltzen da (RDren ordez).

RST5.5, RST6.5, RST7.5. *Restart Interrupts*. INTRaren antzeko hiru sarrera dira.

TRAP. Hau ere eteteko sarrera da. Lehentasuna dauka lehen aipatu ditugunen aldean.

RESET IN. Bai mikroprozesadorearen erregistro bat, programaren kontagailua deritzona, bai etenak baliagarri bihurtzen dituzten biegonkorra eta bai HLDA seinalea zeroan jartzen dituen sarrera da. Programa hasteko erabiltzen da.

RESET OUT. Mikroprozesadorearen RESET egoera adierazten duen sarrera da. Sistemako beste gailuak zeroan jartzeko erabil daiteke.

IO/M. Irakurketa- edo idazketa-memorian edo S/I gailuetan egiten dena adierazten duen irteera da.

X1-X2. Sarrerako terminal hauek, barruko erlojuaren sorgailua egonkortzeko asmoarekin, kuartzozko kristal bat edo LC edo RC sare bat konektatzeko erabiltzen dira.

SID eta SOD. Serieko datuen sarrera eta irteera, hurrenez hurren.

CLK OUT. Mikroprozesadorearen erloju-seinalearen irteera, sistemako erlojutzat erabiltzeko.

$V_{KZ}$  eta  $V_{SS}$ . Elikatzeko terminalak dira. 40. eta 20. adartxoak dira.

## 10.6. 8085 MIKROPROZESADOREAREN EGITURA

---

8085 mikroprozesadorea, funtsean, hiru bloke handik osatzen dute: **kontrol-unitatea, erregistro-multzoa eta unitate aritmetiko logikoa**. Gaitza da gailu bat bestearengandik bereiztea, beren artean harreman edo lotura handia dago eta. Hala ere, oraingoz, gauzak hobeto ulertzeko, bereizita aztertuko ditugu. 10.2. irudian azaltzen da 8085 delakoaren blokeen diagrama.

- a) **Kontrol-unitatea.** Zlaren barruan, bloke logiko kableatua da kontrol-unitatea. Sistemaren atal honek datuen transferentzia eta hainekin egiten diren eragiketak kontrolatu eta sinkronizatzen ditu.

Alde batetik, kontrol-unitateari mikroprozesadorearen erloju-seinalea ezartzen zaio eta, bestetik, sistemako beste elementu batzuen kontrol-seinaleak, programaren sekuentzia eta datu-busa eteteko, aurrez interpretaturiko aginduen deskodetzailearen bidez, unitateak seinale egokiekin erantzuteko modua izan dezan.

Kontrol-unitateko irteera batzuk (memoria eta S/I gailuak) mikroprozesadorearen kanpoko elementuetarantz abiatzen dira, esate baterako, egiten ari den eragiketa irakurketa edo idazketa dela adierazteko. Beste irteera batzuk PUZaren barne-erregistroetara abiatzen dira.

Labur esateko, kontrol-unitateak mikroprozesadorearen eginkizun nagusia erregulatzen du, hau da, aginduak bilatzea eta, ondoren, betetzea. Eragiketa hori ziklikoa da geratzeko agindu batekin (HALT) eteten ez den artean. Bilaketa-aldian agindu bat bidaltzen da memoriatik mikroprozesadorera eta betetze-aldian agindutako eragiketa egiten da.

- b) **Mikroprozesadorearen barne-erregistroak.** 8085 delakoak helburu orokorreko hainbat erregistro dauzka, B, C, D, E, H eta L izenekoak, eta, zortzina bit baino ez daukaten arren, binaka lan egin dezaketenak. Barruko

aldaketan bidez, eragiketetan malgutasun eta bizkortasun handiagoa lortzeko erabiltzen dira.

Badira helburu bereziak dituzten beste erregistro batzuk ere. Hauexek dira: *programaren kontagailua* (PK), 16 bitekoa; pilako erakuslea (SP), hau ere 16 bitekoa; W-Z aldi baterako erregistroak, zortzina bit dauzkatenak, baina biek batera lan egin dezaketenak; *helbide-erregistroa*, 16 bitekoa, zeinaren orteak mikroprozesadorearen helbide-busaren adartxoak baitira; eta azkenik, *aginduen erregistroa* (RI), 8 bitekoa, lehenago ere aipatu duguna.

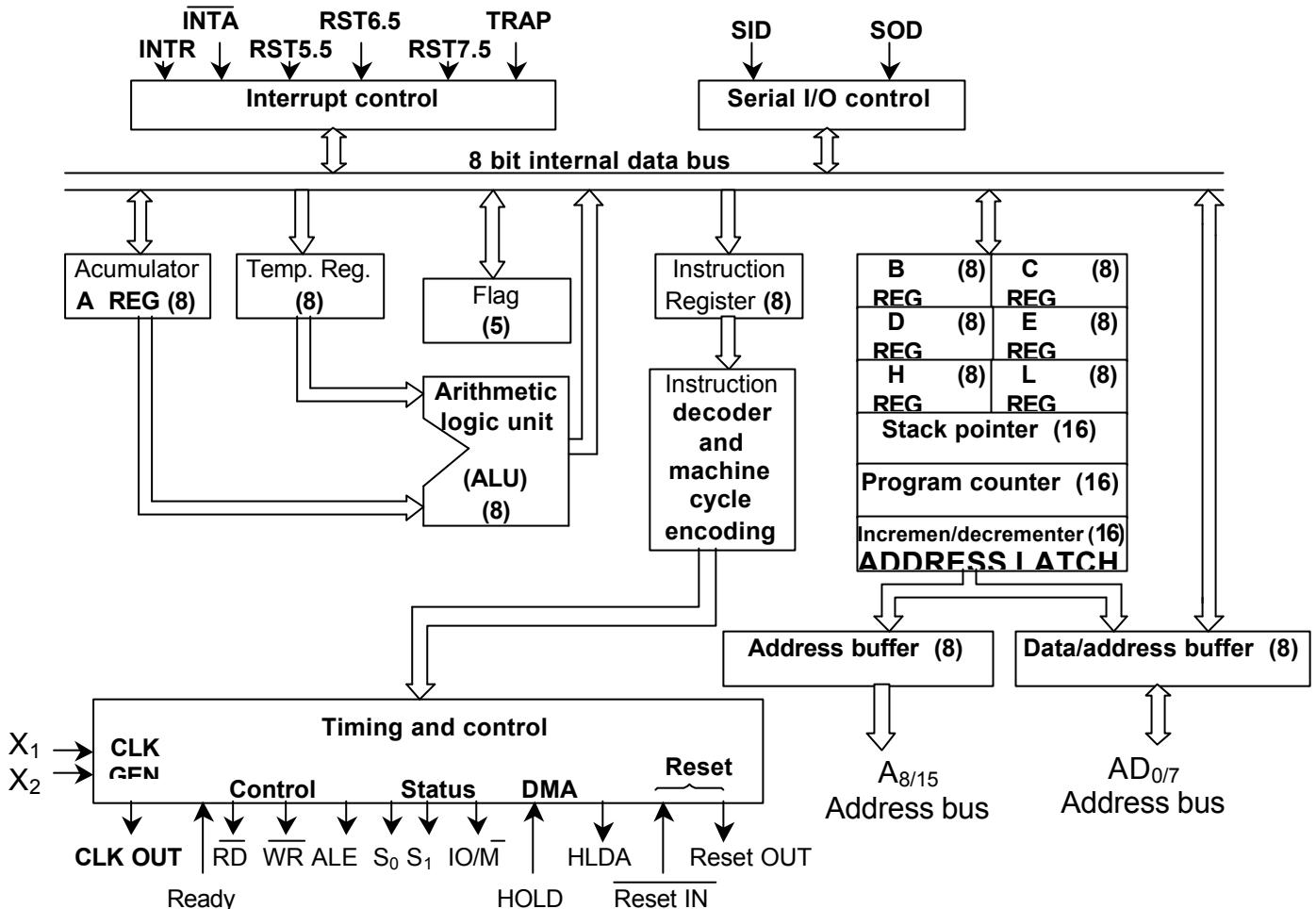
*Programaren kontagailua* bete behar den hurrengo agindua edo byte bat baino gehiago duen agindu baten zati baten helbidea memorian gordetzen duen erregistroa da. Aurrerago ikusiko dugun bezala, 8085aren aginduak, eta, oro har, 8 biteko mikroprozesadore guztien aginduak, memorian bata bestearen atzean ordenatuta daude, eta 1, 2 edo 3 bytekoak izan daitezke.

Kontrol-unitateak automatikoki bateko bat gehitzen dio PKren edukiari, bilaketa-ziklo bat amaitzen duen bakoitzean, HALT (geratzeko) agindua agertzen denean salbu. RESET seinaleak, kontrol-unitateko sarreretako bat denak, zeroan jartzen du PK, eta berriro hasten du programaren exekuzioa.

Programaren lehenengo agindua lortzeko, PKn dagoen helbidea (0000) sistemako helbide-busean jartzen da datu-busaren bidez. Kontrol-unitateak irakurketa-seinale bat sortzen du memorian. Helbideraturiko datua mikroprozesadorera transferitzen da datu-busaren bidez, eta aginduen erregistrora heltzen da.

Agindu baten lehenengo bytea, beti, eragiketa-kodea (EK) izaten da. Aginduen deskodetzaileak interpretatu egiten du eragiketen kode hori eta kodea kontrol-unitatera igarotzen da, zeinak mikroeragiketa-sekuentzia bat sortzen baitu agindua bete dadin. Batzuetan, agindu bat bete ahal izateko, informazio gehiago behar izaten da (agindu batzuen bigarren eta hirugarren byteak). Informazio hori memorian dago metatuta EKren ostean.

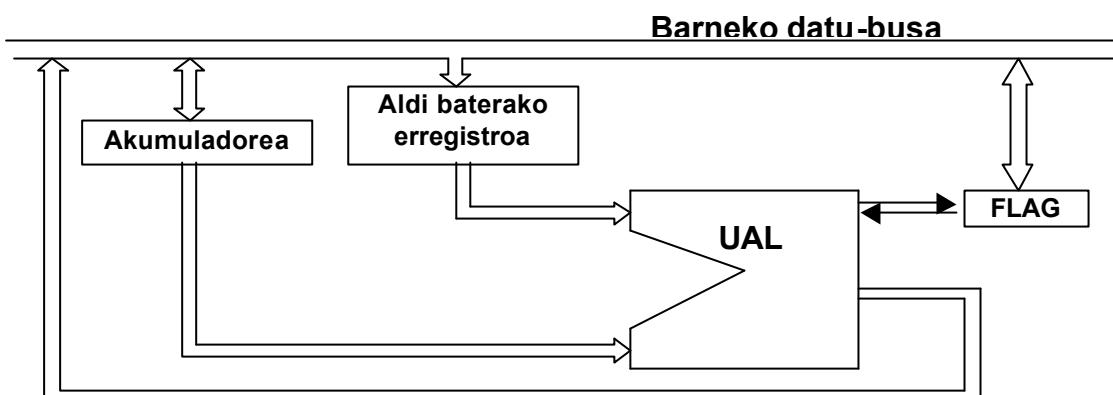
Behin EK deskodetu ondoren, kontrol-unitateak agindu horrek byte gehiago behar duela adierazten duenean, PK-k unitate bat gehiago hartzen du eta berriz ere bere edukia helbide-busera itzultzen du. Helbideraturiko posizioaren edukia W-Z aldi baterako erregistroetako batera pasatzen da, datu-busaren bidez. Erregistro horiek, izenak adierazten duen moduan, aldi batean, agindu baten 2 edo 3 byteak (egonez gero behintzat) metatzeko erabiltzen dira, harik eta mikroprozesadore barneko edo kanpoko beste erregistro batzuetara pasatzen diren arte. Litekeena da aginduaren bigarren eta hirugarren byte hori memoriako helbide bat edo S/I gailuaren hautaketa-helbide bat izatea. Hala balitz, 16 bitak aldi baterako W-Z erregistroetan gordetzen dira. Ondoren, eduki hori PK gainean jartzen da dagokion gailura iristeko.



10.2. irudia

Aginduak ez dira beti memorian idatzita dauden ordena berean betetzen. Esan dugun bezala, posible da programako *azpirrutina* izeneko sailetara jauziak egitea ere. Hartarako, mikroprozesadoredun sistemek *Stack Pointer* edo pilako erregistro bat daukate mikroprozesadorean bertan, baita *pila* edo memoria nagusiko gune erreservatu bat ere. Agindu baten bidez, azpirruttinari dei egiten zaionean, PK-ko edukia pilan kokatzen da, hau da, SPak eta hurrengoak (izan ere, PK-ren edukia 16 bitekoa da) adierazitako memoria-posizioan, harik eta azpirrutina hori burutzen den arte. Behin burutu ondoren, PK memoriako pilan metaturiko balioarekin kargatzen da berriro programa nagusiarekin jarraitzeko. Erregistroarekin bakarrik nahikoa izan beharrean pila bat erabili behar izateko arrazoia kateaturiko azpirruttinak egotea da; posible baita programa nagusitik azpirrutina bati deitu ondoren, honek bigarren bati deitza, bigarrenak hirugarrenari eta hala bata bestearen atzetik jarraitzea. Kasu horretan, hasierako helbide guztiak metatu behar dira gero lehengora itzuli ahal izateko.

- c) **Unitate aritmetiko logikoa.** 8085aren prozesuko unitate osoko (10.3. irudia) osagaiak honako hauek dira: UAL (Unitate Aritmetiko Logikoa) bat, zeina batuketak, kenketak eta eragiketa logikoak egin ditzakeen konbinaziozko zirkuitua baita; akumuladore (M) izeneko 8 biteko erregistro bat; eta bost biegonkor edo flag multzo bat, egoera-erregistro deritzona eta martxan dauden eragiketa aritmetiko edo logikoen emaitzen berri ematen duena.



**10.3. irudia.** Unitate aritmetiko logiko osoa.

Eragiketa bat egiten denean, akumuladoreak edukitzen du eragigaietako bat eta W-Z aldi baterako erregistroetako batek bestea. Eragiketaren emaitza akumuladorean jartzen da, aurretik zeukan datu edo eragigaiaren ordez.

Flag bakotzaren izena eta kokapena 10.4. irudian azaltzen da. CY bururakoaren adierazle bat da, eta bere edukia 1 izango da batuketa bat egitean bururakoa agertzen denean. AC bururako laguntzailearen adierazlea da. BCDan eragiketak egiten direnean erabiltzen da. Horren balioa 1 izango da, eskuinetik hasita laugarren lekuaren dauden eragigaien biten batuketan bururakoa agertzen denean. S zeinu-flaga da, eta horren balioa 1 izango da emaitzaren pisurik handieneko bita 1 bada. Bestela, zero izango da. Z zeroaren adierazlea da, eta 1ean jartzen da emaitza zero denean.

S	Z		AC		P		CY
---	---	--	----	--	---	--	----

**10.4. irudia.** Egoera-erregistroa, flag bakotzaren kokapenarekin.

## 10.7. HELBIDERATZEKO ERAK

8085 mikroprozesadorearen aginduek agerian edo ezkutuan daramatzaten datuen erreferentzia egiten dute hainbat eratan.

Datu erreferentzia egiteko lau modu edo helbideratzeko era daude: berehalakoa, zuzena, erregistrozkoa eta zeharkako erregistrozkoa.

**Berehalako** helbideratzean, aginduak eragiketa-kodearen hurrengo bytean edo bytetan dauzka datuak, esate baterako, ADI 04. Eta 04 hamaseitarra (aginduaren bigarren bytea) batu egiten zaio akumuladoreari.

Helbideratzeko erarik simpleena **zuzena** da. Kasu honetan, aginduko bigarren eta hirugarren byteek zehazten dute datua non dagoen adierazten duen memoria-posizioaren helbidea, esate baterako, LDA 003F. 003F posizioaren edukia akumuladorera pasatzen da.

**Erregistrozko** helbideratzean, aginduak zehazten du zein erregistro edo erregistro bikoitzetan dagoen datua, esate baterako, ADD B. Eta B erregistroko edukia batu egiten zaio akumuladoreari.

Azkenik, **zeharkako erregistrozko** helbideratzean, aginduan zehazturiko erregistro bikoitzak datua daraman memoria-helbidea dauka, esate baterako, LDAX B. Memoria-helbidearen edukia, zeinaren helbidea B eta C erregistro-parean baitago, akumuladorerantz mugitzen da.

## 10.8. INSTRUKZIO-MOTAK

---

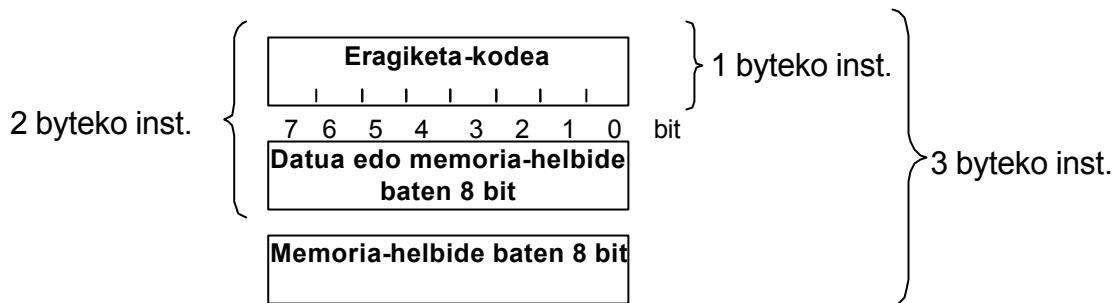
Mikroprozesadore bakoitzak uler eta exekuta ditzakeen instrukzio-kopurua ezberdina da kasu bakoitzean. Bi egitura daude: CISC (*Complex Instruction Set Computer*), mikro klasikoetan (8085) ikus daitekeena; eta RISC (*Reduced Instruction Set Computer*), askoz ere instrukzio gutxiago dituena eta denak denbora berean exekuta daitezkeenak.

Azertzen ari garen 8085ak 74 oinarrizko instrukzio ditu; horiek 246 eragiketa ezberdin egin ditzake. Instrukzio horiek 5 taldetan sailka daitezke:

- Erregistroen artean edo memoriaren eta erregistroen arteko datu-transferentzia
- Eragiketa aritmetikoak
- Eragiketa logikoak
- Adarkatzeak
- Pila, S/I eta makinaren kontrolerako instrukzioak

## 10.9. INSTRUKZIOEN FORMATUA

Instrukzioak 1, 2 edo 3 bytez osatuak daude. 10.5. irudian ikus daitezke instrukzio baten byte-kopurua eta horien edukia



10.5. irudia

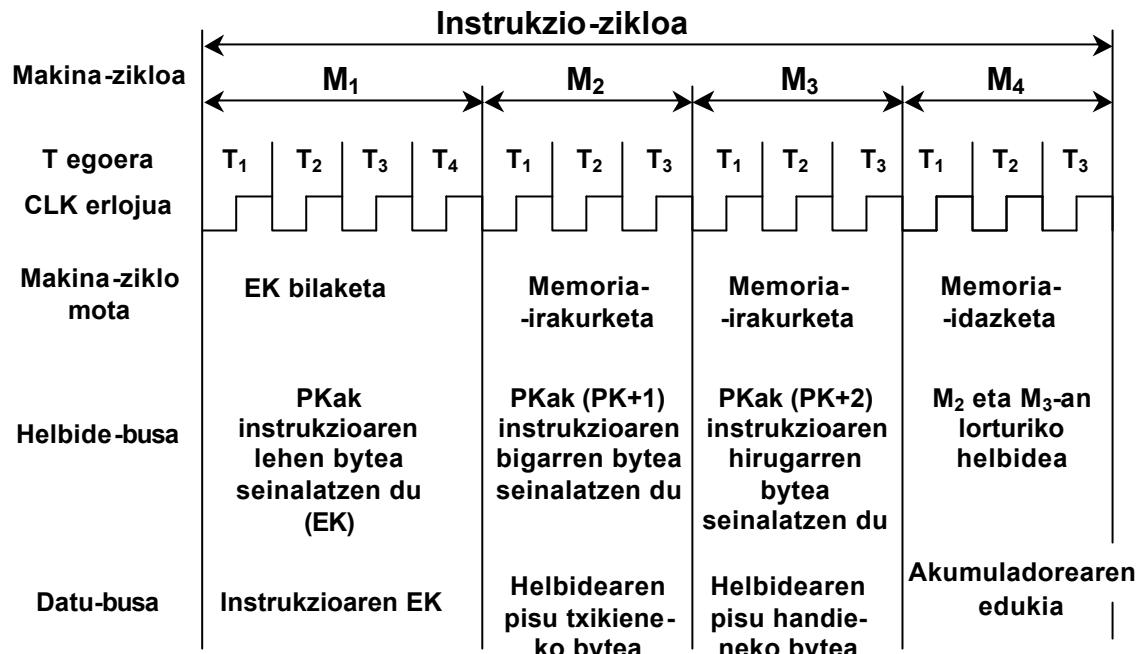
## 10.10. INSTRUKZIO-ZIKLOA ETA MAKINA-ZIKLOA

Zenbait makina-ziklo irauten duen instrukzioaren **bilaketak** eta **execuzioak** **instrukzio-zikloa** osatzen du. Era berean, makina-zikloak hainbat egoera (erloju nagusiaren zikloa) behar ditu.

8085 mikroprozesadorean honako makina-zikloak bereiz daitezke:

1. Eragiketa-kodearen bilaketa
2. Memoria-irakurketa
3. Memoria-idazketa
4. S/I gailuaren irakurketa
5. S/I gailuaren idazketa
6. Eteteari erantzutea
7. Bus ez-aktiboa

Adibidez, azter dezagun STA 00FF instrukzioa: hiru byteko instrukzioa dugu eta akumuladorearen edukia EKren ondoren adierazten den helbidera transferitzen du. Ikus 10.6. irudia.



10.6. irudia

## 10.11. SINTESI-ADIBIDEA

Ondoren azalduko dugun adibidearen bidez, batez ere, mikroprozesadore osoaren funtzionamendua eta hark memoriarekin hartu-emanak edukitzeko daukan era azaltzen ahaleginduko gara.

A adibidea. Sistema bateko memoriaren 006A eta 006B posizioetan dauden bi zenbaki batu eta emaitza bertako 0010 posizioan gorde.

Eragiketa hori egiteko behar den programa, hamaseitarrean eta nemonikoekin (mihiztatzailea) adierazita, honako hau izango da:

HELBIDEA	HAMASEITARRA	NEMONIKOA	IRUZKINA
0000	3A6A00	LDA 006A	Adierazitako helbidearen edukia akumuladorean kargatu.
3	47,00	MOV B,A	Akumuladoreko edukia B erregistrora pasatu.
4	3A6800	LDA 0068	Adierazitako helbidearen edukia akumuladorean kargatu.
7	80,00	ADD B	B-ren edukia akumuladorearen edukiarekin batu
8	STA 0010	STA 0010	Batuketaren emaitza adierazitako helbidean idatzi.

**Eragiketen sekuentzia:**

**1. Lehen agindua.** Programaren kontagailuak helbide-busean jartzen du informazioa, eta mikroprozesadoreak memorian irakurketa bat egiteari ekiten dio. Ondorioz, memoriak 3A ipintzen du datu-busean eta informazio hori aginduen erregistroan jasotzen da.

3A-ren deskodeketak (EK), EKren hurrengo bi bytek daramaten helbidearen edukiarekin, akumuladorea kargatzeko agindu bat dela adierazten du. Hartarako, PK bi aldiz gehitzen da eta memorian beste bi irakurketa egiten dira. 6A da lehenengoaren edukia eta 00 bigarrenarena. Informazio hori aldi baterako erregistroetan kargatzen da. Aginduaren bilatze-aldia amaitu da.

Aldi baterako erregistroen edukia helbide-busean jartzen da eta memorian irakurketa bat egiten da. Datu-busean 02 agertzen da eta informazio hori akumuladorera pasatzen da. Lehenengo agindua burutu da.

**2. Bigarren agindua.** PK gehitu egiten da eta 0003 memoria-posiziorantz apuntatzen du. Aginduaren bilaketari ekiten dio. Apuntaturiko helbidearen edukia 47 da, eta aginduen deskodetzaileak B helburu orokorreko erregistroan akumuladoreko edukia kargatzeko agindua ulertzten du. Hori gertatzen denean, agindua burutu da.

**3. Hirugarren agindua.** Hirugarren agindua lehenengoaren berdina da.

**4. Laugarren agindua.** Programako kontagailuaren edukia orain 0007 da. Helbideraturiko posizioan 80 datua ageri da, eta hori aginduaren EKri dagokio. EK deskodetzailera garraiatzen da eta han interpretatu egiten da, B erregistroaren edukia akumuladoreko edukiarekin batzeko ulertz. Eragiketa hori egiten denean amaitu da laugarren agindua.

**5. Bosgarren agindua.** PK-k 0008 posiziorantz apuntatzen du, non EK baitago. Bere edukia (32) aztertu egiten du deskodetzaileak, akumuladoreko edukia 0010 memoria-posizioan metatzeko ulertz. Helbidea, aldi baterako, W-Z erregistroetan gordetzen da. Erregistro horiek beren edukia helbide-busean jartzen dute eta idazketa-eragiketa bati ekiten zaio, akumuladorean dagoen 07 datua idazteari hain zuzen ere, memoriaren 0010 posizioan .

## 10.12. 8085aren INSTRUKZIOAK

### 10.12.1. DATUEN TRANSFERENTZIA

NEMONIKOA	DESKRIBAPENA	
MOV r <sub>1</sub> , r <sub>2</sub>	(r <sub>1</sub> ) ← (r <sub>2</sub> ) r <sub>2</sub> erregistroaren edukia r <sub>1</sub> erregistrora pasatzen da	Ziklo-kopurua: 1 Egoera-kopurua: 4 Helbideratzea:erregistrozkoa
MOV r , M	(r) ← [(H) (L)] HL erregistroek adierazten duten memoria-posizioaren edukia r erregistrora	Ziklo-kopurua: 2 Egoera-kopurua: 7 Helbideratzea: zeharkako erregis.
MOV M ,r	[(H) (L)] ← (r) r-ren edukia HL erregistroek adierazten duten posiziora	Ziklo-kopurua: 2 Egoera-kopurua: 7 Helbideratzea: zeharkako erregis.
MVI r , datua	(r) ← (2. bytea) instrukzioaren 2. bytearen edukia r erregistrora	Ziklo-kopurua: 2 Egoera-kopurua: 7 Helbideratzea: berehal.
MVI M , datua	[(H) (L)] ← (2. bytea) instrukzioaren 2. bytearen edukia HL erregistroek adierazten duten posiziora	Ziklo-kopurua: 3 Egoera-kopurua: 10 Helbideratzea: zeharkako erregis.
LXI rp, 16-ko datua	(rh) ← (3. bytea) / (rl) ← (2. bytea) instrukzioaren 3. bytea pisu handieneko erregistrora eta 2. bytea pisu txikieneko erregistro-parera	Ziklo-kopurua: 3 Egoera-kopurua: 10 Helbideratzea: berehalakoa
LDA helbidea	←[(3. byte) (2. byte)] 2. eta 3. byteak adierazitako posizioaren edukia akumuladorera	Ziklo-kopurua: 4 Egoera-kopurua: 13 Helbideratzea: zuzena
STA helbidea	(3. byte) (2. byte)] ← (A) akumuladorearen datua 2. eta 3. byteak adierazitako posiziora	Ziklo-kopurua: 4 Egoera-kopurua: 13 Helbideratzea: zuzena
LHLD helbidea	(L) ←[(3. bytea) (2. bytea)] (H) ←[(3. bytea) (2. bytea)+1] 2. eta 3. byteak adierazitako posizioaren edukia L erregistrora eta hurrengo posizioarena H-ra	Ziklo-kopurua: 5 Egoera-kopurua: 16 Helbideratzea:zuzena
SHLD helbidea	[(3. bytea) (2. bytea)] ← (L) [(3. bytea) (2. bytea)+1] ← (H) L erregistroaren datua 2. eta 3. byteak adierazitako posiziora doa eta H-rena hurrengo posiziora	Ziklo-kopurua: 5 Egoera-kopurua: 16 Helbideratzea:zuzena
LDAX rp	(A) ← [(rp)] rp erregistro bikoitzak adierazten duen posizioko datua akumuladorera (BC/DE)	Ziklo-kopurua: 2 Egoera-kopurua: 7 Helbideratzea: zeharkako erregistroa
STAX rp	[(rp)] ← (A) Akumuladoreko datua rp erregistro bikoitzak adierazten duen posiziora	Ziklo-kopurua: 2 Egoera-kopurua: 7 Helbideratzea: zeharkako erregis.
XCHG	(H) ↔ (D) (L) ↔ (E)	Ziklo-kopurua: 1 egoera-kopurua: 4 Helbideratzea: erregis.

### 10.12.2. ERAGIKETA ARITMETIKOAK

NEMONIKOA	DESKRIBAPENA	
ADD r	$\leftarrow (A) + (r)$ r erregistroko datua eta akumuladoreko batu egiten dira, eta emaitza akumuladorean geratzen da	Ziklo-kopurua: 1 Egoera-kopurua: 4 FLAG: Z,S,P,CY,AC
ADD M	$\leftarrow (A) + [(H)(L)]$ HL erregistroek adierazten duten memoria-posizioko datua akumuladorekoarekin batzen da. Emaitza akumuladorean.	Ziklo-kopurua: 2 Egoera-kopurua: 7 FLAG: Denak
ADI datua	$\leftarrow (A) + (2. \text{ bytea})$ instrukzioko 2. bytearen datua akumuladorekoarekin batzen da. Emaitza akumuladorean.	Ziklo-kopurua: 2 Egoera-kopurua: 7 FLAG: denak
ADC r	$\leftarrow (A) + (r) + CY$ r erregistroko datua eta bururakoa akumuladoreko edukiarekin batzen dira. Emaitza akumuladorean.	Ziklo-kopurua: 1 Egoera-kopurua: 4 FLAG: denak
ADC M	$\leftarrow (A) + [(H)(L)] + CY$ HL erregistroek adierazten duten memoria-posizioko datua eta bururakoa akumuladorekoarekin batzen dira. Emaitza akumuladorean.	Ziklo-kopurua: 2 Egoera-kopurua: 7 FLAG: denak
ACI datua	$\leftarrow (A) + (2. \text{ bytea}) + CY$ instrukzioko 2. bytearen datua eta bururakoa akumuladorekoarekin batzen dira. Emaitza akumuladorean.	Ziklo-kopurua: 2 Egoera-kopurua: 7 FLAG: denak
SUB r	$\leftarrow (A) - (r)$ r erregistroaren edukia akumuladorearenari kentzen zaio. Emaitza akumuladorera.	Ziklo-kopurua: 1 Egoera-kopurua: 4 FLAG: denak
SUB M	$\leftarrow (A) - [(H)(L)]$ HL erregistroek adierazten duten memoria-posizioko datua akumuladorearenari kentzen zaio. Emaitza akumuladorera.	Ziklo-kopurua: 2 Egoera-kopurua: 7 FLAG: denak
SUI datua	$\leftarrow (A) - (2. \text{ bytea})$ instrukzioko 2. bytearen datua akumuladorearenari kentzen zaio. Emaitza akumuladorera.	Ziklo-kopurua: 2 Egoera-kopurua: 7 FLAG: denak
SBB r	$\leftarrow (A) - (r) - CY$ r erregistroaren edukia bururakoarenari kentzen zaio. Emaitza akumuladorera.	Ziklo-kopurua: 1 Egoera-kopurua: 4 FLAG: denak
SBB M	$\leftarrow (A) - [(H)(L)] - CY$ HL erregistroek adierazten duten memoria-posizioko datua eta bururakoa akumuladorearenari kentzen zaio. Emaitza akumuladorera.	Ziklo-kopurua: 2 Egoera-kopurua: 7 FLAG: denak
SBI datua	$\leftarrow (A) - (2. \text{ bytea}) - CY$ instrukzioko 2. bytearen datua eta bururakoa akumuladoarenari kentzen zaio. Emaitza akumuladorera.	Ziklo-kopurua: 2 Egoera-kopurua: 7 FLAG: denak
INR r	$(r) \leftarrow (r) + 1$ r erregistroaren edukiari unitate bat gehitzen zaio.	Ziklo-kopurua: 1 Egoera-kopurua: 4 FLAG: Z, S, P, AC
INR M	$[(H)(L)] \leftarrow [(H)(L)] + 1$ HL erregistroek adierazten duten memoria-posizioko edukiari unitate bat kentzen zaio.	Ziklo-kopurua: 3 Egoera-kopurua: 10 FLAG: Z, S, P, AC
DCR r	$(r) \leftarrow (r) - 1$ r erregistroaren edukiari unitate bat gutxitzen zaio.	Ziklo-kopurua: 1 Egoera-kopurua: 4 FLAG: Z, S, P, AC
DCR M	$[(H)(L)] \leftarrow [(H)(L)] - 1$ HL erregistroek adierazten duten memoria-posizioko edukiari unitate bat kentzen zaio.	Ziklo-kopurua: 3 Egoera-kopurua: 10 FLAG: Z, S, P, AC

INX rp	(rh) (rl) $\leftarrow$ (rh) (rl) +1 rp erregistro bikoitzaren edukiari unitate bat gehitzen zaio.	Ziklo-kopurua: 1 Egoera-kopurua: 6 FLAG: ----
DCX rp	(rh) (rl) $\leftarrow$ (rh) (rl) -1 rp erregistro bikoitzaren edukiari unitate bat kentzen zaio.	Ziklo-kopurua: 1 Egoera-kopurua: 6 FLAG: ----
DAD rp	(H) (L) $\leftarrow$ (H) (L) + (rh) (rl) rp erregistro bikoitzaren edukia HL erregistroen edukiaz batzen da. Emaitza HL-n geratzen da.	Ziklo-kopurua: 3 Egoera-kopurua: 10 FLAG:CY
DAA	Akumuladoreko 8 biteko datua BCDko 4 biteko bi digitu bihurtzen da	Ziklo-kopurua: 1 Egoera-kopurua: 4 FLAG: denak

### 10.12.3. ERAGIKETA LOGIKOAK

NEMONIKOA	DESKRIBAPENA	
ANA r	(A) $\leftarrow$ (A) . (r) r erregistroaren eta akumuladorearen edukien arteko AND logikoa burutzen du. Emaitza akumuladorean.	Ziklo-kopurua: 1 Egoera-kopurua: 4 FLAG: denak
ANA M	$\leftarrow$ (A) . [(H) (L)] HL erregistroek adierazitako posizioko eta akumuladorearen edukien arteko AND logikoa egiten du. Emaitza akumuladorean.	Ziklo-kopurua: 2 Egoera-kopurua: 7 FLAG: denak
ANI datua	$\leftarrow$ (A) . (2. bytea) Instrukzioko 2. bytea eta akumuladorearen edukien arteko AND logikoa egiten du. Emaitza akumuladorera.	Ziklo-kopurua: 2 Egoera-kopurua: 7 FLAG: denak
XRA r	$\leftarrow$ (A) $\oplus$ (r) r erregistroaren eta akumuladorearen edukien arteko XOR logikoa egiten du. Emaitza akumuladorean.	Ziklo-kopurua: 1 Egoera-kopurua: 4 FLAG: denak
XRA M	(A) $\leftarrow$ (A) $\oplus$ [(H) (L)] HL erregistroek adierazitako posizioko eta akumuladorearen edukien arteko XOR logikoa egiten du. Emaitza akumuladorean.	Ziklo-kopurua: 2 Egoera-kopurua: 7 FLAG: denak
XRI datua	$\leftarrow$ (A) $\oplus$ (2. bytea) Instrukzioko 2. bytea eta akumuladorearen edukien arteko XOR logikoa egiten du. Emaitza akumuladorera.	Ziklo-kopurua: 2 Egoera-kopurua: 7 FLAG: denak
ORA r	$\leftarrow$ (A) + (r) r erregistroaren eta akumuladorearen edukien arteko OR logikoa egiten du. Emaitza akumuladorean.	Ziklo-kopurua: 1 Egoera-kopurua: 4 FLAG: denak
ORA M	$\leftarrow$ (A) + [(H) (L)] HL erregistroek adierazitako posizioko eta akumuladorearen edukien arteko OR logikoa egiten du. Emaitza akumuladorean.	Ziklo-kopurua: 2 Egoera-kopurua: 7 FLAG: denak
ORI datua	$\leftarrow$ (A) + (2. bytea) Instrukzioko 2. bytea eta akumuladorearen edukien arteko OR logikoa egiten du. Emaitza akumuladorera.	Ziklo-kopurua: 2 Egoera-kopurua: 7 FLAG: denak
CMP r	- (r) r erregistroaren edukia akumuladorearenari kentzen zaio. Akumuladorea ez da aldatzen Z=1 (A) = (r) bada eta CY = 1 (A) < (r) bada.	Ziklo-kopurua: 1 Egoera-kopurua: 4 FLAG: denak

CMP M	- $[(H) (L)]$ HL erregistroek adierazitako posizioko edukia akumuladorearenari kentzen zaio. Akumuladorea ez da aldatzen $Z = 1 (A) = [(H) (L)]$ bada eta CY = 1 (A) $< [(H) (L)]$ bada	Ziklo-kopurua: 2 Egoera-kopurua: 7 FLAG: denak
CPI datua	- (2. bytea) Instrukzioko 2. bytea akumuladorearenari kentzen zaio. Akumuladorea ez da aldatzen $Z = 1 (A) = (2. bytea)$ bada eta CY = 1 (A) $< (2. bytea)$	Ziklo-kopurua: 2 Egoera-kopurua: 7 FLAG: denak
RLC	$(A_{n+1}) \leftarrow (A_n)$ ; $(A_0) \leftarrow (A_7)$ ; CY $\leftarrow (A_7)$ Akumuladorearen edukia ezkerrerantz mugitzen da posizio bat	Ziklo-kopurua: 1 Egoera-kopurua: 4 FLAG: CY
RRC	$(A_n) \leftarrow (A_{n+1})$ ; $(A_7) \leftarrow (A_0)$ ; CY $\leftarrow (A_0)$ Akumuladorearen edukia eskuinerantz mugitzen da posizio bat	Ziklo-kopurua: 1 Egoera-kopurua: 4 FLAG: CY
RAL	$(A_{n+1}) \leftarrow (A_n)$ ; CY $\leftarrow (A_7)$ ; $(A_0) \leftarrow CY$ Akumuladorearen edukia ezkerrerantz mugitzen da posizio bat, baina CY flag-etik zehar. Pisu handieneko bita CY-ra doa eta hau pisu txikienekeko bitera.	Ziklo-kopurua: 1 Egoera-kopurua: 4 FLAG: CY
RAR	$(A_n) \leftarrow (A_{n+1})$ ; CY $\leftarrow (A_0)$ ; $(A_7) \leftarrow CY$ Akumuladorearen edukia eskuinerantz mugitzen da posizio bat, baina CY flag-etik zehar. Pisu txikieneko bita CY-ra pasatzen da eta hau pisu handieneko bitera	Ziklo-kopurua: 1 Egoera-kopurua: 4 FLAG: CY
CMA	$\leftarrow (\bar{A})$ Akumuladoreko datua bere osagarri bihurtzen da (zeroak batera eta alderantziz)	Ziklo-kopurua: 1 Egoera-kopurua: 4
CMC	CY $\leftarrow \overline{CY}$ CY bandera bere osagarrira pasatzen da	Ziklo-kopurua: 1 Egoera-kopurua: 4 FLAG: CY
STC	CY $\leftarrow 1$ CY bandera 1ean ipintzen da	Ziklo-kopurua: 1 Egoera-kopurua: 4 FLAG: CY

#### 10.12.4. ADARKATZEAK

NEMONIKOA	DESKRIBAPENA	
JMP helbidea	(PK) $\leftarrow (3.\text{bytea}) (2.\text{bytea})$ 2. eta 3. bytean adierazitako helbidean dagoen instrukzioari dagokio kontrola	Ziklo-kopurua: 3 Egoera-kopurua: 10 Berehalakoa
JXX helbidea	Baldintza betetzen bada, 2. eta 3. bytean adierazitako helbidean dagoen instrukzioari dagokio kontrola, bestela programak sekuentzia normala jarraitzen du.	Ziklo-kopurua: 2/3 Egoera-kopurua: 7/10 Berehalakoa
CALL helbidea	$[(SP) - 1] \leftarrow (PKH)$ ; $[(SP) - 2] \leftarrow (PKL)$ $(SP) \leftarrow (SP) - 2$ ; $(PK) \leftarrow (3.\text{bytea}) (2.\text{bytea})$ PK-ren edukia pilan metatzen da. Kontrola 2. eta 3. bytek adierazten duten helbidean dagoen instrukzioari dagokio.	Ziklo-kopurua: 5 Egoera-kopurua: 18 Berehalakoa/Zeharkakoa
CXX helbidea	Baldintza betetzen bada, CALL instrukzioan adierazitako ekintzak egiten dira $[(SP) - 1] \leftarrow (PKH)$ ; $[(SP) - 2] \leftarrow (PKL)$ $(SP) \leftarrow (SP) - 2$ ; $(PK) \leftarrow (3.\text{bytea}) (2.\text{bytea})$	Ziklo-kopurua: 2/5 Egoera-kopurua: 9/18 Berehalakoa/Zeharkakoa

RET	$(PKL) \leftarrow [(SP)] ; (PKH) \leftarrow [(SP) + 1] ; (SP) \leftarrow (SP) + 2$ Baldintza gabeko jauzia egiten da ohiko sekuentzia apurtu den programaren puntura	Ziklo-kopurua: 3 Egoera-kopurua: 10 Zeharkakoa
RXX	Baldintza betetzen bada, RET bezala dabil $(PKL) \leftarrow [(SP)] ; (PKH) \leftarrow [(SP) + 1] ; (SP) \leftarrow (SP) + 2$	Ziklo kopurua: 1/3 Egoera kopurua: 6/12 Zeharkakoa
RST n	$[(SP) - 1] \leftarrow (PKH) ; [(SP) - 2] \leftarrow (PHL)$ $(SP) \leftarrow (SP) - 2 ; (PK) \leftarrow 8(NNN)$ Kontrola 8 aldiz NNNren edukia den helbidean dagoen instrukzioari dagokio.	Ziklo-kopurua: 3 Egoera-kopurua: 12 Zeharkakoa
PCHL	$(PKH) \leftarrow (H) ; (PKL) \leftarrow (L)$ HL erregistroen edukia programa-kontagailura pasatzen da.	Ziklo-kopurua: 1 Egoera-kopurua: 6 Erregistrozkoa

#### 10.12.5. PILA, S/I ETA MAKINA-KONTROLA

NEMONIKOA	DESKRIBAPENA	
PUSH rp	$[(SP) - 1] \leftarrow (rh) ; [(SP) - 2] \leftarrow (rl) ; (SP) \leftarrow (SP) - 2$ rp erregistro bikoitzaren edukia pila edo stack-era pasatzen da.	Ziklo-kopurua: 3 Egoera-kopurua: 12
PUSH PSW	Programaren egoera-hitza (PSW), akumuladorearen eta flag-en edukiaz osatua, pilan metatzen da	Ziklo-kopurua: 3 Egoera-kopurua: 12
POP rp	$(rl) \leftarrow [(SP)] ; (rh) \leftarrow [(SP) + 1] ; (SP) \leftarrow (SP) + 2$ Pilaren edukia rp erregistro bikoitzera pasatzen da	Ziklo-kopurua: 3 Egoera-kopurua: 10
POP PSW	Pilan gorderiko programaren egoera-hitza berreskuratuz, akumuladorean eta flag-eten kargatzten da	Ziklo-kopurua: 3 Egoera-kopurua: 10
XTHL	$(L) \leftrightarrow [(SP)] ; (H) \leftrightarrow [(SP) + 1]$ HL erregistroen edukia pilako lehen bi posizioetakoekin trukatzen da	Ziklo-kopurua: 5 Egoera-kopurua: 16
SPHL	$(SP) \leftarrow (H) (L)$ HL erregistroen edukia SP erregistrora pasatzen da	Ziklo-kopurua: 1 Egoera-kopurua: 6
IN ataka	$\leftarrow$ (datua) Instrukzioan adierazitako atakak datu-busean ipinitako datua akumuladorera pasatzen da	Ziklo-kopurua: 3 Egoera-kopurua: 10
OUT ataka	(datua) $\leftarrow$ (A) Akumuladoreko datua, datu-busetik zehar, instrukzioan adierazitako atakara pasatzen da.	Ziklo-kopurua: 3 Egoera-kopurua: 10
EI	Etete-sistema gaitzen du	Ziklo-kopurua: 1 Egoera-kopurua: 4
DI	Etete-sistema desgaitzen du	Ziklo-kopurua: 1 Egoera-kopurua: 4
HLT	Alto. PUZA geldialdi-egoeran sartzen da. Erregistroak eta banderak ez dira aldatzen	Ziklo-kopurua: 1 Egoera-kopurua: 5
NOP	Ez da inolako eragiketarik egiten	Ziklo-kopurua: 1 Egoera-kopurua: 5
RIM	Etendura-maskara irakurtzen da	Ziklo-kopurua: 1 Egoera-kopurua: 4
SIM	Etendura-maskararen idazketa	Ziklo-kopurua: 1 Egoera-kopurua: 4

### 10.12.6. SINBOLOGIA ETA LABURDURAK

SINBOLOA	ESANAHIA
helbidea	16 biteko adierazpen bitarra
datua	8 biteko adierazpen bitarra
ataka	S/I gailu baten 8 biteko helbidea
r, r <sub>1</sub> , r <sub>2</sub>	A, B, C, D, E, H, L erregistroetako bat
rp	B-C, D-E, H-L erregistro bikoitzetako bat, B-k B-C adierazten du, D-k D-E eta H-k H-L.
XX	Baldintzapeko jauzietako baldintzak NZ (Z=0), Z (Z=1), NC (CY=0), C (CY=1), PO (P=0), PE (P=1), P (S=0) eta M (S=1)
n	Otik 7ra bitarteko zenbaki hamartarra
rh	rp erregistro bikoitzaren edukiaren pisu handieneko 8 bitak
rl	rp erregistro bikoitzaren edukiaren pisu txikieneño 8 bitak
PKH eta PKL	Programa-kontagailuaren PK pisu handieneko eta txikieneño 8 bitak

### 10.13. UP 2000 PLAKA (SISTEMA GARATUA)

#### 10.13.1. HARDWAREA

DIN A4ko zirkuitu inprimatuko txartelean garaturiko sistema dugu. Txartelak honako osagaiak ditu:

- ✓ Intel 8085 8 biteko mikroprozesadorea, 4,195 Mhz-eko kuartzozko erlojuarekin.
- ✓ Programa-monitorea duen 4 kbyteko EPROM (2732) memoria.
- ✓ Erabiltzailearen programak gordetzeko erabiltzen den RAM memoriako (2x4802) 4kbyte.
- ✓ Ordenagailu-interfazea, 8155 (UART) integratuaren bitartez.
- ✓ Erabiltzailearen eskura 22 S/I lerro. Hiru konektoretan: bi 8koak eta bat 6koak.
- ✓ RS-232 serie-irteera bi, SID eta SOD terminalekin bata eta 8251 USARTaz bestea.
- ✓ Beste 24 S/I lerro ematen digun 8255 PIA.
- ✓ Teklatua eta bistaratzea kontrolatzen dituen 8279 integratua.
- ✓ Hedapen-busa, bertan datu-busa, helbide-busa eta kontrol-busa amplifikatuak edo indartuak daude.

### 10.13.2. SOFTWAREA

Programa-monitorearen oinarrizko funtioak hauek dira:

- ✓ Memoria eta erregistroetako edukia aztertu.
- ✓ RAM memoria eta erregistroen edukia aldatu.
- ✓ Erabiltzailearen programak exekutatu.
- ✓ Programak pausoz pauso exekutatu.
- ✓ Memoria-zonak lekuz aldatu.
- ✓ Ordenagailutik programak kargatu.

(INIC) tekla sakatu ondoren, aginduak jasotzeko prest geratzen da. Datuak eta helbideak hamaseitarrean ulertzen ditu. Monitoreak, aginduaren zain dagoenean, ezkerraldean “—“ karakterea agertzen du. Parametro bat itxaroten duenean, puntuagertzen du datu- edo helbide-eremuaren eskuinaldean.

### 10.13.3. MONITOREAREN TEKLATU-KOMANDOAK (AGINDUAK)

**INIC** (hasieratzea). Hardwarea berrasieratzen du eta 8085 mezua bistaratzendik.

**S.ME** (memoria ordezkatu). EPROM memoria irakurri eta RAMa idatzi edo irakurtzea ahalbidetzen du.

Adibidea: S.ME <helbidea> POST <datua> ... POST <datua> EJEC  
POST teklak hurrengo helbidera pasarazten gaitu. ANT teklak aurrekora.  
EJEC komandoarekin, sartutako datuak memorizatzen dira.

**E. REG** (erregistroak aztertu). Erregistroen edukia ikusteko edo aldatzeko.

Erregistroen bistaratzeko ordena honakoa da:

<b>1.-A</b>	<b>Akumuladorea</b>		
<b>2.-B</b>	<b>B erregistroa</b>	<b>3.-C</b>	<b>C erregistroa</b>
<b>4.-D</b>	<b>D erregistroa</b>	<b>5.-E</b>	<b>E erregistroa</b>
<b>6.-F</b>	<b>Flag-en bytea</b>	<b>7.-I</b>	<b>Etendura-maskara</b>
<b>8.-H</b>	<b>H erregistroa</b>	<b>9.-L</b>	<b>L erregistroa</b>
<b>10.-SPH</b>	<b>Pilako erakuslearen pisu handieneko bytea</b>		
<b>11.-SPL</b>	<b>Erakuslearen pisu txikiesteko bytea</b>		
<b>12.-PKH</b>	<b>Programa-kontagailuaren pisu handieneko bytea</b>		
<b>13.-PKL</b>	<b>Programa-kontagailuaren pisu handieneko bytea</b>		

**GO** (programaren exekuzioa). GO tekla sakatzean, programa-kontagailuaren edukia bistaratzen da helbide-eremuan, eskuinaldean hamartarren puntu batekin. Programa-kontagailua alda dezakegu orain. EJEC tekla sakatuz gero, programa-kontagailuak adierazitako helbideko instrukzioak hartzen du kontrola, datu- eta helbide-eremuak ezabatu egiten dira eta “E” bat agertzen da helbide-eremuaren ezkerraldean (EJEC sakatu ordez beste teklaren bat sakatuz gero, errore-mezua agertzen da).

- Kontrola monitorera itzultzen da INIC tekla sakatzean, edo exekutaturiko programan RST 0, RST 1 edo JMP 0 instrukzioa badago.
- Komando honek ondo funtzionatzeko, pilak RAM memorian egon behar du. Horretarako, 1FH sartu PKH erregistroan eta FFH PKL erregistroan.

#### 10.13.4. ASM85 MIHIZTATZAILEA

Honen lehen agindua instrukzioen hasierako helbidea (ORG) markatzea izango da. Gure kasuan ORG 1000H.

Balioak hamartarrean, bitarrean (B), zortzitarrean (O,Q) eta hamaseitarrean (H) adieraz daitezke. Hamaseitarra erabiliko dugu, edo bitarra kasu berezietan.

Programaren lehen eremua (ezkerretik) etiketarena izango da, eta hori erregistro edo instrukzio batena ez den letraz hasiko da.

Eragigaiaren eremuan etiketak, balioak edo adierazpenak joan daitezke. Balioa letra batez hasten bada (hamaseitarrean), aurretik “0” bat ipini behar zaio, bestela etiketa dela ulertuko du mihiztatzzaileak.

Etiketak, konstanteak, eragile matematikoak edo logikoak osa dezakete adierazpena: A+B, A-B, A\*B, A/B, NOT B, a AND B, A XOR B

Iruzkinak eta oharrak (;) ondoren idatzi behar dira.

#### 10.13.5. MIHIZTATZAILEAREN SASIAGINDU ARTEZKARIAK

**DB.-** Mihiztatuko ez den zortzikotea hasten du DB 43H, 12H, 09H

**DW.-** Berdina, baina hitzkin. DW 3FFFH, 1078H, 2F0CH

**EQU.-** Berdintasuna

**END.-** Mihiztatzalearen bukaera (ez programarena).

**ORG.-** Hasiera

## 10.13.6. DDT 85

## L Programa kargatu

**D** (helbide baxua), (helbide altua) > Memoria bistaratu

**G** (helbidea), (break helbidea), (break helbidea) > Exekutatu, gelditu, gelditu

**M** (helbide baxua), (helbide altua), (helburuko helbidea) > memoria mugitu

X (erreq.) > Bistaratu eta aldatu erregistroak

I > Txertatu H > Bertan behera utzi

**Mibitztadura:** Mibitztadura longeakoa lehiakako programak martxa longeakoa itzultzea helburutzat duen programa itzultzailea.

**Mintzatadura-lengoak.** Beste-mintzak lengoak sinbolikoa. Lengoak horietan idatzitako agindu bakoitzak makina-lengoaiako agindu bakarra sortzen du.

**Konpliadore:** Gai-mailako lengoaldeko idatzitako programa jakin bat ordenagailuaren makina-lengoaiara itzultzen duen programa.

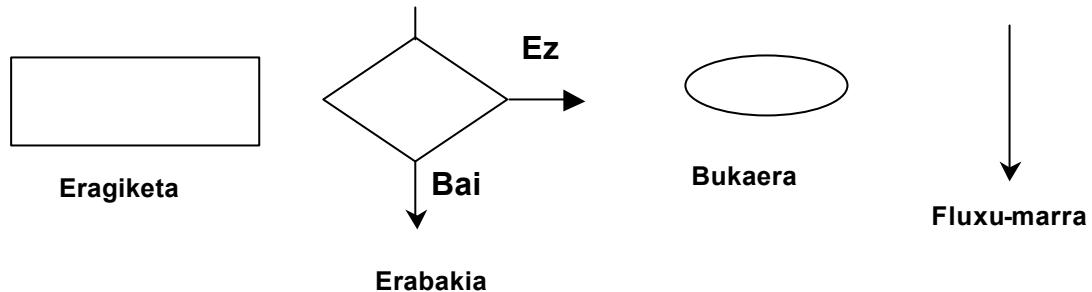
**Estekatzaile:** Konpiladoreak sortutako objektu-moduluak elkartu eta haien arteko lotura-helbideak ebazten dituen oinarrizko softwarearen programa.

## **10.14. PROGRAMAZIOA**

Egin nahi dena ondo definitu ondoren, fluxu-diagrama edo organigrama eraikitzen da. Fluxu-diagraman adierazten dira grafikoki eta ordenan aplikazio jakin bat ebazteko behar diren oinarrizko eragiketa guztiak.

Organigraman erabiltzen diren sinboaloak ondoren azaltzen dira:

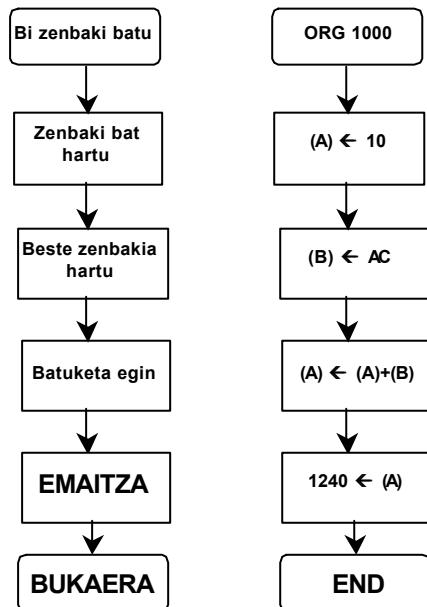
- **Eragiketazkoak:** “laukizuzena”. Eragiketa zehatz bat adierazten dute, adibidez, batu, akumuladorea kargatu...
  - **Erabakizkoak:** “erronboa”. Eragiketen sekuentzia ordenatua puskatzea ahalbidetzen dute baldintza bat betetzen denean. Baldintza horren arabera, bide batetik edo bestetik jarraituko du programak.
  - **Bukaerakoak:** “laukizuzen biribildua”. Hasieran eta bukaeran erabiltzen da.
  - **Fluxu-marra:** “gezia”. Bidea adierazten du.



10.7. irudia

## ADIBIDEA

Programari zuzenean emango zaizkion bi zenbaki batu eta emaitza 1240H memoria-helbidean gorde.



HELVIDEA	Kodea EK	Nemonikoa	Iruzkinak
1000	31	LXI SP,100C8H	SP < 10C8
1001	C8		
1002	10		
1003	3E	MVI A,10H	A < 10
1004	10		
1005	06	MVI B,ACH	B < AC
1006	AC		
1007	80	ADD B	A < (A)+(B)
1008	32	STA 1240H	1240 < A
1009	40		
100A	12		
100B	76	HLT	GELDI
	10		
	AC		
1240			

### 10.14.1. AZPIPROGRAMAK EDO AZPIRUTINAK

Programa batean sarritan errepikatzen den edo programa ezberdinietan era berean jokatzen duen instrukzio-multzoak osatzen du azpiprograma. Interesgarriak dira, zeren behin bakarrik idatzita hainbat aldiz programa ezberdinietan erabil baitaitezke.  $\mu$ P2000 plakaren programa-monitorean badaude batzuk:

HELBIDEAK	NEMONIKOA	AZALPENA
04C9	UPDAD	HL erregistro bikoitzaren edukia helbide-eremuan bistaratzen da. Erregistro guztiak aldatzen dira.
04D5	UPDDT	A erregistroaren edukia notazio hamaseitarrean bistaratzen da datu-eremuan. Erregistro guztiak aldatzen dira.
03C3	HXDSP	Digitu hamaseitarrak hedatzen ditu bistaratzeko. Sarrera : DE . A, H, L eta F aldatzen ditu.
0B74	DELAY	1 ms-ko atzerapena sortzen du.
044E	RDKBD	Teklatutik irakurri. Programa honek teklatutik karaktere bat sartu arte itxaroten du, akumuladorean gordez. A, H, L eta F erregistroak aldatzen ditu. Azpirrutina honek ondo funtziona dezan RST5.5 etendura desmaskaratu egin behar dugu SIM instrukzio batez.
041D	OUTPUT	Display-tik karaktereak ateratzeko. A = 0 → helbide-eremuan A = 1 → datu-eremuan B = 0 → hamartarren komarik gabe B = 1 → hamartarren koma batekin HL → karaktereak hasten diren posizioa
037D	GTHEX	Digitu hamaseitarrak onartzen ditu teklatutik...
02BF	TODIR	Teklatutik helbidea onartzen du eta DE erregistroan gorde. Bestalde, display-a ezabatzen du. Erregistro denak aldatzen ditu.
02CF	VISNU	Datu-eremuan, akumuladorearen edukia agertzen du. Erregistro guztiak aldatzen dira.

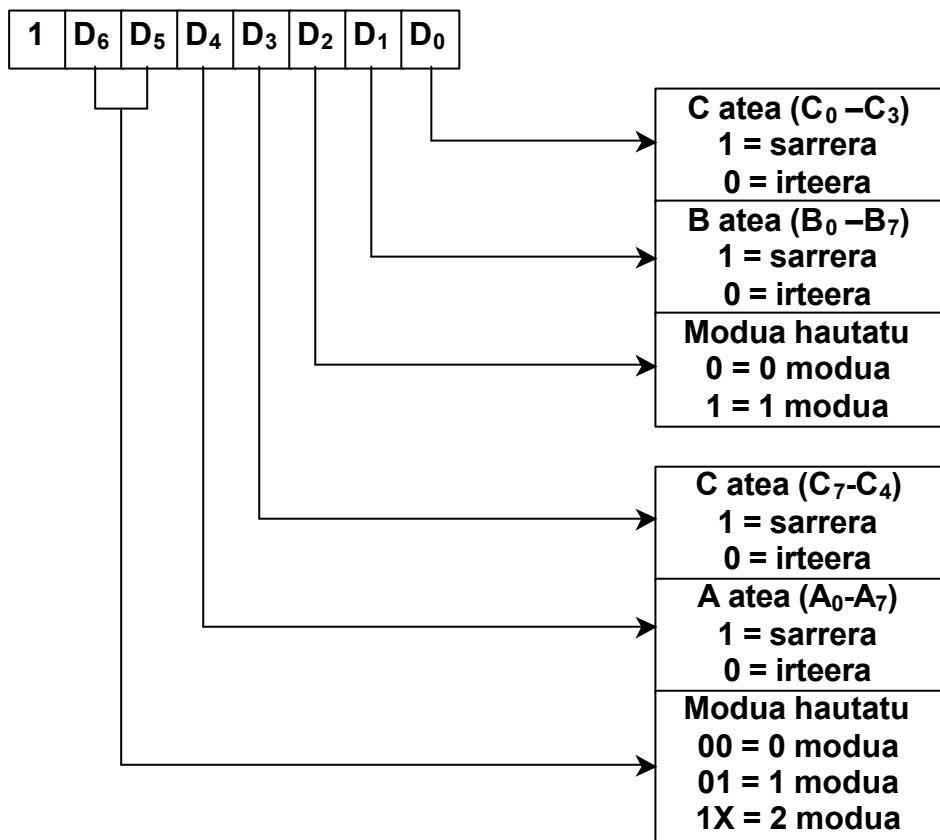
## 10.15. PERIFERIKOAK: 8255 PPI (*PROGRAMMABLE PERIPHERAL INTERFACE*)

Sarrera/irteera programagarriak dituen periferikoa da. 24 S/I lerro ditu, eta 8 biteko hiru atetan banatzen dira. Lau erregistro ditu: A atea, B atea, C atea eta kontroleko erregistroa. C atea lau lerroko bi ate bezala programa daiteke (C<sub>7</sub>-C<sub>4</sub> eta C<sub>0</sub>-C<sub>3</sub>).

A<sub>1</sub> eta A<sub>0</sub> helbide-buseko lerroek lau erregistro hauek hautatzen dituzte.

$A_1$	$A_0$	erregistroa
0	0	A ATEA
0	1	B atea
1	0	C atea
1	1	Kontrola

Bestalde, hiru modutara funtziona dezake. 0 modua bakarrik aztertuko dugu. Horretarako, kontroleko erregistroan hitz egokia idatzi behar da. 8 bit horiek honako hau adierazten dute:



Erregistro honetan markatzen dugu zein ate izango den sarrera eta zein irteera, gero ate bakoitzera edo kontrol-erregistrarra joateko  $A_0$  eta  $A_1$  helbide-lerroak ditugu, eta  $\mu$ P-2000 plakan ( $\overline{CS}$ ) A atea 3800 helbidean dago, B atea 3801ean, C atea 3802an eta Kontrola 3803 helbidean. Idazketa eta irakurketa guztiak akumuladoretik egiten dira.

Adibidea: A atea eta C altua sarrera bezala eta B atea eta C baxua irteera bezala programatu

Kontroleko atea

1	0	0	1	1	0	0	0
---	---	---	---	---	---	---	---

**Zenbaki hau 98H da**

Lehendabizi akumuladorean kargatzen dugu 98h eta gero datu hori kontroleko atearren 3803 helbidera eramango dugu.

MVI            A, 98H  
STA            3803H

A atean dagoen informazioa irakurri nahi badugu:

LDA            3800H

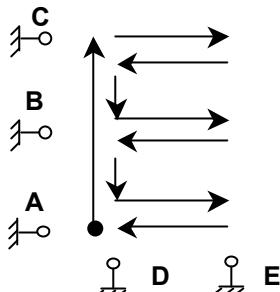
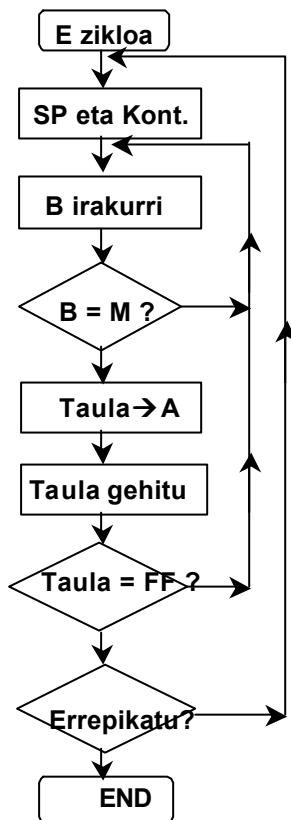
B atean datu bat idatzi nahi badugu:

STA            3801H

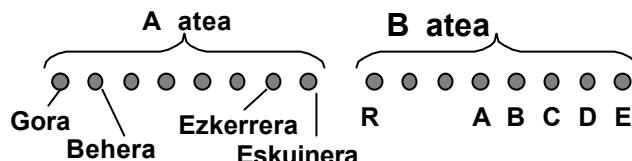
LDA eta STA instrukzioak erabili ordez IN eta OUT erabiltzen badira, helbideak honako hauek dira: **A atea → 38, B atea → 39, C atea → 3A eta Kontrola → 3B**

MVI            A, 98H  
OUT            3BH

**Praktika: E ziklo sekuentziala egiten duen automatismoaren kontrola**



B sarrerak			A irteerak		
1500	12H	AD	GORA	80H	1400
1501	06H	CD	ESK.	01H	1401
..	05H	CE	EZK.	02H	..
..	..	CD	..	..	..
..	..	BD	..	..	..
..	..	..	..	..	..
..	..	..	..	..	..
..	..	AE	..	..	..
..	..	AD	..	..	..
150A	FFH			00H	



## Ariketak

---

- 10.1. Hiru zenbaki batu. Horren baturak ez du 255 (FFH) gainditu behar. Kasu honetan zenbakioek erabili: 23h, 47h eta 12h. Emaitza 1200 helbidean gorde.
- 10.2. 1030 eta 104A posizioetan dauden X eta Y zenbakiak batu eta emaitza 1250 posizioan utzi (M datua erabiliz).
- 10.3. Bi aldagai, 1300 eta 1301 posizioetan daudenak, biderkatu. Biderkadura 1302 posizioan gorde (urretik aldagai horien balioak idatziko ditugu, adb.: 6 eta 8)
  - ✓ Pausoz pauso exekutatu programa
  - ✓ Dagokion erregistroaren eta memoria-posizioen edukia aztertu, instrukzio bakoitza egikaritu ondoren.
- 10.4. Lehendabiziko N zenbakien batura emango digun programa egin. Baturak ez du  $256_{10}$  gainditu behar.
- 10.5. Bi zenbaki konparatu, baten balioa finkoa izanik (1CH) eta bestearena 1300 posizioan gordeko den aldakorra. Zenbaki aldakorra konstantea baino handiagoa bada, bien batura 1301 posizioan gorde; txikiagoa bada, bien arteko kendura 1302an. Azkenik, bi zenbakiak berdinak badira, memoria-posizio biak zeroan geratzen dira.
- 10.6. Datu-eremuan agertuko den segundo-orratza egin. (0, 1, ....60, 0, 1, ...). Temporizadoreak egiteko: FFH datua kargatu erregistro batean eta gutxitzen joan. FFFFH erregistro bikoitzean kargatu eta gutxitzen joan. XXH, bi edo hiru erregistro kargatu eta bata bestearen barruan dauden begiztak sortu.
- 10.7. Erlojua. Orduak, minutuak eta segundoak bistaratuko dituen erlojua programatu. Bestalde, orduan ipintzeko aukera izan behar du.
- 10.8. Mezu keinukaria. Helbide-eremuan karaktere batzuk bistaratu. Programa hau gauzatzeko, monitorearen OUTPUT azpiprograma erabili behar da.
- 10.9. M zikloa egiten duen automatismoaren kontrola.
- 10.10. Keinukaria. A atearen pisu handieneko bita aktibatuz gero, B atearen bit guztiak keinuka jartzen dira. Pisu gutxienekoa aktibatzen bada, eskuinera biratzen da B atearen bit bat, bestela ezkerrera.

# 11. KAPITULUA

## 8051/52/552 mikrokontroladoreak

### 11.1. SISTEMA OSOA

Mikroprozesadore batek berak bakarrik ezin du ezer egin, beste zirkuitu batzuk (periferikoak) behar ditu. Gutxienez hauek beharko lituzke:

- *EPROM memoria* Programak gordetzeko.
- *RAM memoria* Datuak gordetzeko.
- *I/O periferikoak* Kanpoko aldagaiekin komunikatzeko.
- *UART* Komunikazio-seriea egiteko.
- *TIMERak* Temporalizazioak egiteko.
- *A/D eta D/A eta abar.*

### 11.2. MIKROKONTROLADOREA

Teknologiaren aurrerapenarekin, periferiko horiek guztiak integratu ziren txip bakar batean. Horrela jaio zen mikrokontroladorea. Mikrokontroladore batek honako elementu hauek eduki ditzake:

- PUZ.
- RAM.
- ROM edo EPROM.
- Paraleloko I/O.
- Serieko I/O.
- Kontagailuak eta temporizadoreak.
- Etenduren kontrola.
- A/D eta D/A bihurgailuak.

## 11.3. APLIKAZIOAK

Txip honen hedapena izugarria izan da azken urte hauetan. Normalean lan zehatz eta ez oso konplikatua egiten dute. Beraz, oso erraza da mikrokontroladoreak horrelako arloetan aurkitzea:

- Informatikan, elementu bat kontrolatzeko. Adibidez, inprimagailu, teklatu, modem, disko-unitate, etab.etc.
- Etxetresna elektrikoetan: ikuzgailu, sukalde elektriko, berogailu, etab.etc.
- Ibilgailuetan: motor-kontrol, barruko ordenagailu, serbopalazta, etab.etc.
- Audio/bideo aparatueta: telebista, bideo, pletina, etab.etc.
- Industrian:
  - ❖ Erregulazio-lanetan: tenperatura, abiadura, maila, etab.etc.
  - ❖ Edozein prozesu automatizatzeko: ate automatiko, muntaia-kate, igogailu, etab.etc.
  - ❖ Automata gehienak mikrokontroladore batekin egiten dira.
  - ❖ Robotak kontrolatzeko.
- Medikuntzan.
- Aplikazio militarretan (armak).
- Domotikan (etxe adimendunak).

## 11.4. AUKERAKETA

Ez da erraza esatea noiz komeni den mikroprozesadorea erabiltzea eta noiz mikrokontroladorea. Normalean, diruak esango digu zer komeni den, baina beste gauza batzuk ere kontuan hartu beharko ditugu, adibidez, memoria gutxi edo I/O lerro asko behar izanez gero, mikrokontroladorea aukeratuko dugu.

Gero, 8 edo 16 bitekoa izango den erabaki behar da. Eta zer etxetakoa hartuko den: Motorola, Intel, etab.

Kapitulu honetan, Intelen 8051 familia aztertuko dugu.

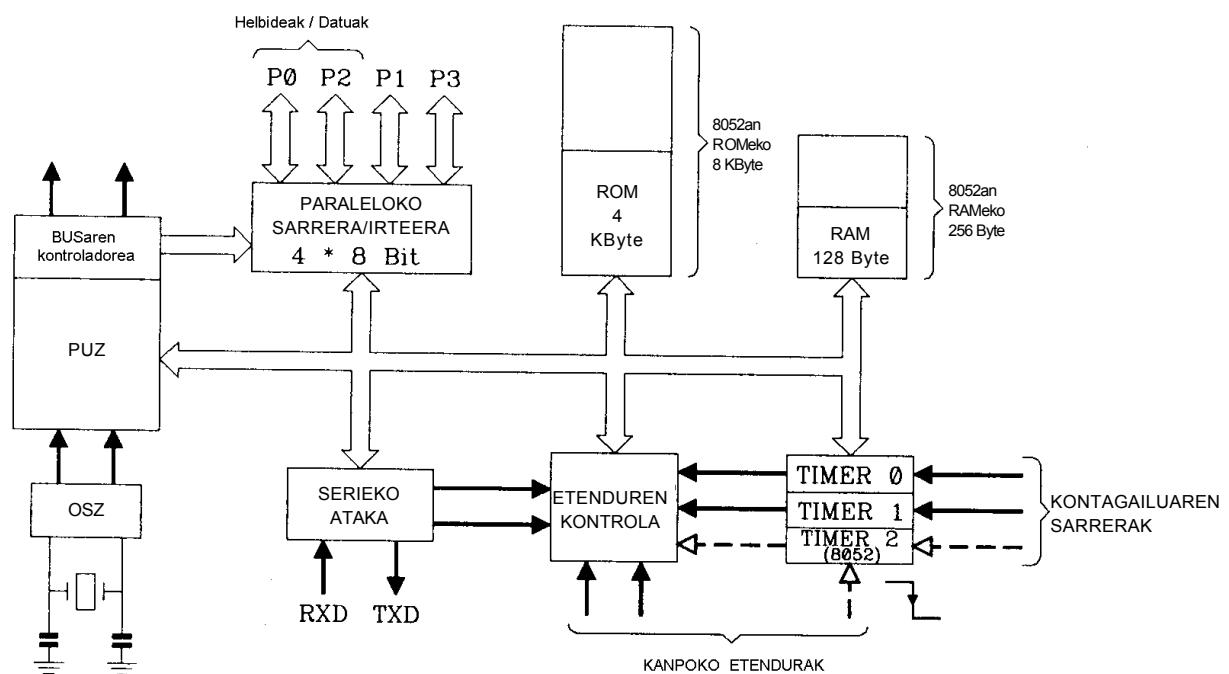
## 11.5. 51 FAMILIA

Familia honen barruan, mikrokontroladore-pila bat daukagu, hainbat aldaerarekin. Normalean, 8XX1 eta 8XX2 dira. 11.1. taulan ospetsuenak ikus ditzakegu.

ROM dutenak	ROM gabeak	EPROM dutenak	ROMeko bytak	RAMeko bytak	TIMER-ak	Teknologia
8051	8031	8751	4K	128	2	HMOS
8051AH	8031AH	8751H	4K	128	2	HMOS
8052 <sup>a</sup> H	8032AH	8752BH	8K	256	3	HMOS
80C51BH	80C31BH	87C51	4K	128	2	CHMOS

11.1. taula

Ikusten denez, 8XX2 taldekoek memoria handiagoa dute, baita timer bat gehiago ere. Gainerakoan berdinak dira, beraz, 8051 batean funtzionatzen duen programak 8052 batean ere funtzionatuko duela esan dezakegu, baina alderantziz agian ez. 11.1. irudian 8051 eta 8052 mikrokontroladoreen arteko ezberdintasunak ikusten dira.



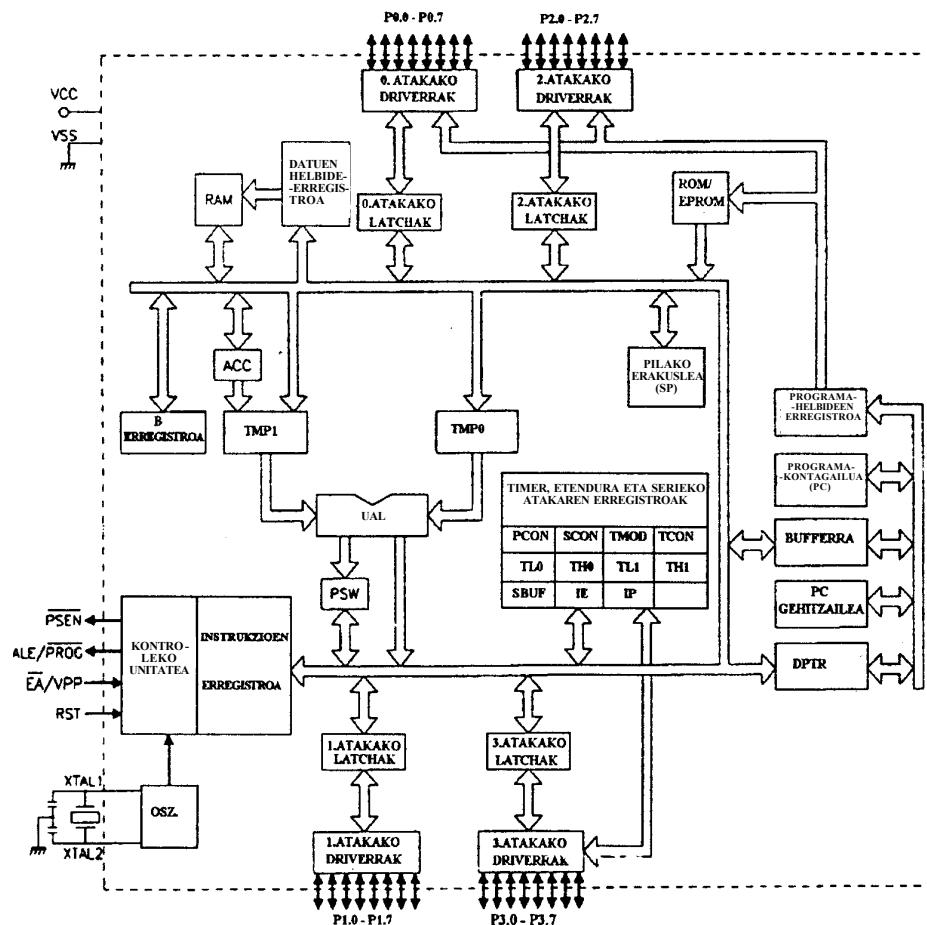
11.1. irudia

## 11.6. 8051/8052 MIKROKONTROLADOREEN EZAGARRIAK

Mikro hauen ezaugarriak honako hauek dira:

- 8 biteko PUZA.
- Biteko aginduak (Prozesadore boolearra).
- 8 biteko 4 ataka.
- 256 byteko barruko *RAMa* (8052), edo 128 bytekoa (8051).
- 8KB barruko *ROMa* (8052) edo *EPROMa* (8752); 4KB (8X51).
- Kanpoko “*programa-memoria*” 64KB-eraino irits daiteke.
- Kanpoko “*datu-memoria*” ere 64KB-ekoa.
- 3 *Timer/kontagailu* (8052) edo 2 bakarrik (8051).
- Serieko linea asinkronoa eta erabateko *duplexa*.
- 6 etendura-maila, lehentasuna programagarria izanik. 5 (8051).
- Barruko erloju-seinalearen sortzailea.

11.2. irudian, mikro honen bloke nagusiak ikus daitezke.



11.2. irudia

## 11.7. 8051/52 MIKROKONTROLADOREEN PIN-AK

**V<sub>KZ</sub>** Elikadura +5V.

**V<sub>ss</sub>** Lurra 0V.

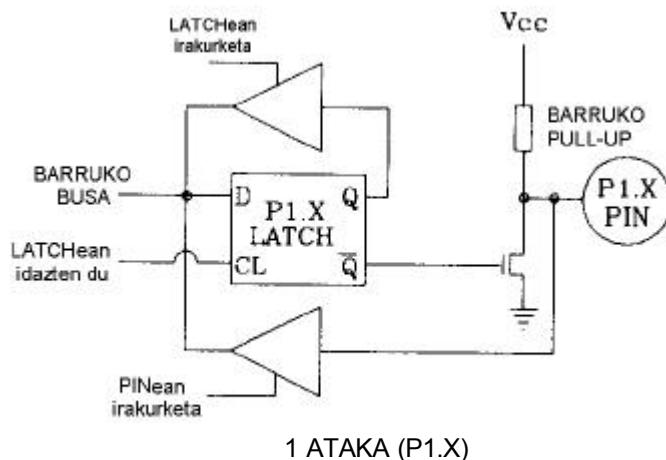
**Port.** 8 biteko 4 ataka noranzko biko ditu. I/O atakak izateaz gain, beste hainbat funtzio izaten dituzte.

**P0** 8085ean bezala, datu-busa eta helbide-busaren zati baxua, denboran multiplexatuta ateratzen du.

Horrez gain, barruko EPROMa programatzean, datuak hemendik sartzen dira.

**P1** Barruko EPROMa programatzean, hemendik sartzen da helbide-busaren zati baxua. 8052an, P1.0k eta P1.1ek Timer2 kontrolatzen dute.

11.3. irudian pin horien barruko egitura edo hardwarea ikus daiteke.



11.3. irudia

**P2** Hau helbide-busaren zati altua da, kanpoko memoria helbideratzeko. Baita barruko EPROMa programatzeko ere.

**P3** Ataka honetan bit bakotzak funtzi bat dauka:

Pinak	Funtzioa
P3.0	RXD (serieko atakaren sarrera)
P3.1	TXD (serieko atakaren irteera)
P3.2	$\overline{INT0}$ (etendura 0. Kanpokoa)
P3.3	$\overline{INT1}$ (etendura 1. Kanpokoa)
P3.4	T0 ( <i>Timer0</i> . Kanpoko sarrera)
P3.5	T1 ( <i>Timer1</i> . Kanpoko sarrera)
P3.6	$\overline{WR}$ (kanpoko memorian idazteko seinalea)
P3.7	$\overline{RD}$ (kanpoko memorian irakurtzeko seinalea)

## 11.2. taula

**ALE / PROG** ALE (Address Latch Enable) 8085ean bezala. Eta PROG barruko EPROMa programatzeko pultsuak.

**PSEN** PSEN (Program Strobe Enable) Kanpoko **programa-memorian** irakurtzeko seinalea. Mikrokontroladore honetan kanpoko memoria bi motatakoa da: “*programa-memoria*” eta “*datu-memoria*”; bi mota horiek bereizteko, PSEN erabiltzen da.

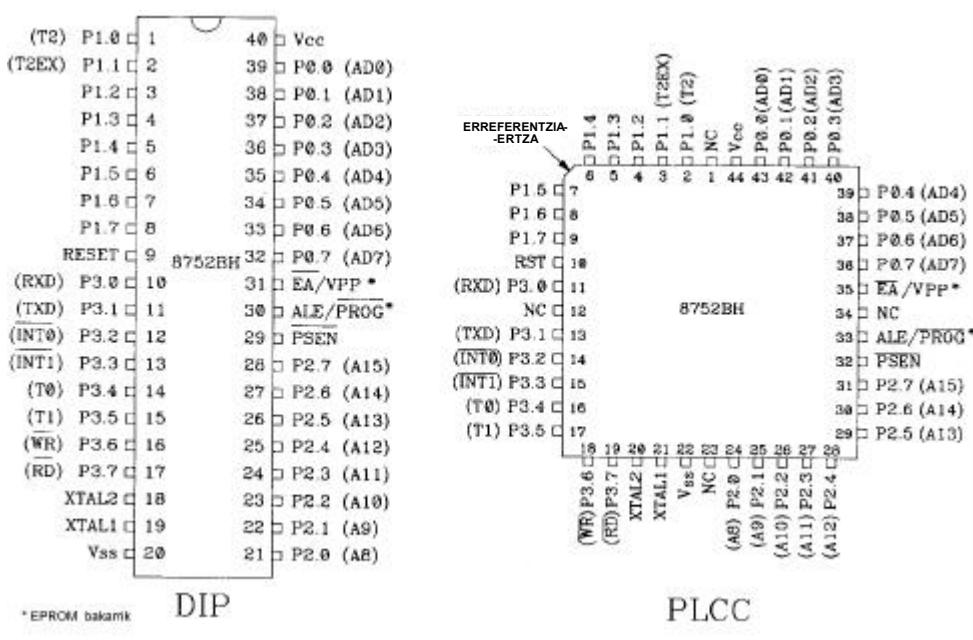
**EA / VPP** EA (External Access) Aktibatzen denean (“0”), kanpoko memorian dagoen programa egikaritzen da. Aktibatuta ez dagoenean (“1”), barruko ROM memoriako programa egikaritzen da 1FFFH (8052) edo 0FFFH (8051) helbideraino. Handik aurrera beti kanpoko memoriara doa (barruan ez dagoelako). Beraz:

EA = 1, mikrokontroladore gisa egiten du lana.  
 EA = 0, mikroprozesadore gisa funtzionatuko du.

VPP barruko EPROM programatzeko tentsioa.

**XTAL1 eta XTAL2** Barruko osziladorearen sarrerak. Normalean, 12MHz-eko kuartzoa jartzen da, nahiz eta modelo batzuk 33MHz onartu.

**RESET** Sistema berrabiarazteko. Programa-kontagailua (PK) 0000H helbidea-rekin kargatzen da.

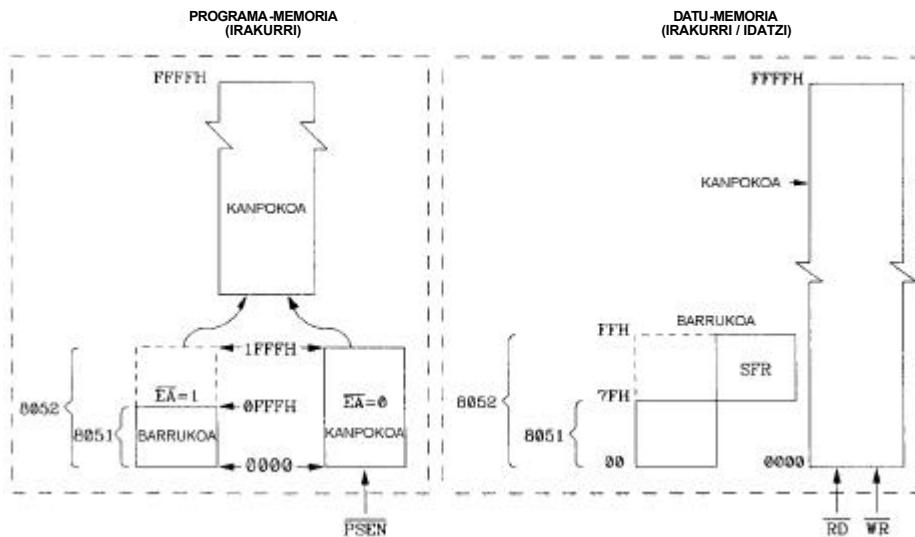


11.4. irudia

## 11.8. MEMORIAREN ANTOLAKETA 51/52 MIKROKONTROLA-DOREETAN

### 11.8.1. DATU-MEMORIA ETA PROGRAMA-MEMORIA

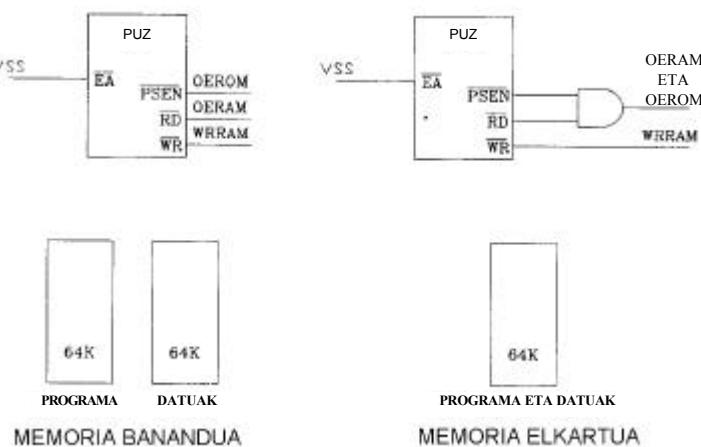
Mikrokontroladore honek programa-memoria eta datu-memoria bereizten ditu. Programa-memoriatik irakurtzeko PSEN seinalea erabiltzen du. Datu-memoriarekin lan egiteko, berriz, RD eta WR:



11.5. irudia

Bi memoria-mota horiek bereizteari **HARVARD** arkitektura deitzen zaio. Bereiziko ez balitz, **VON NEUMANN** arkitektura liztateke. 8085ean bezala, 8051k ere **VON NEUMANN** arkitekturarekin lan egin dezake aldaketa txiki bat eginez.

11.6. irudian, egin behar den aldaketa ikus daiteke.

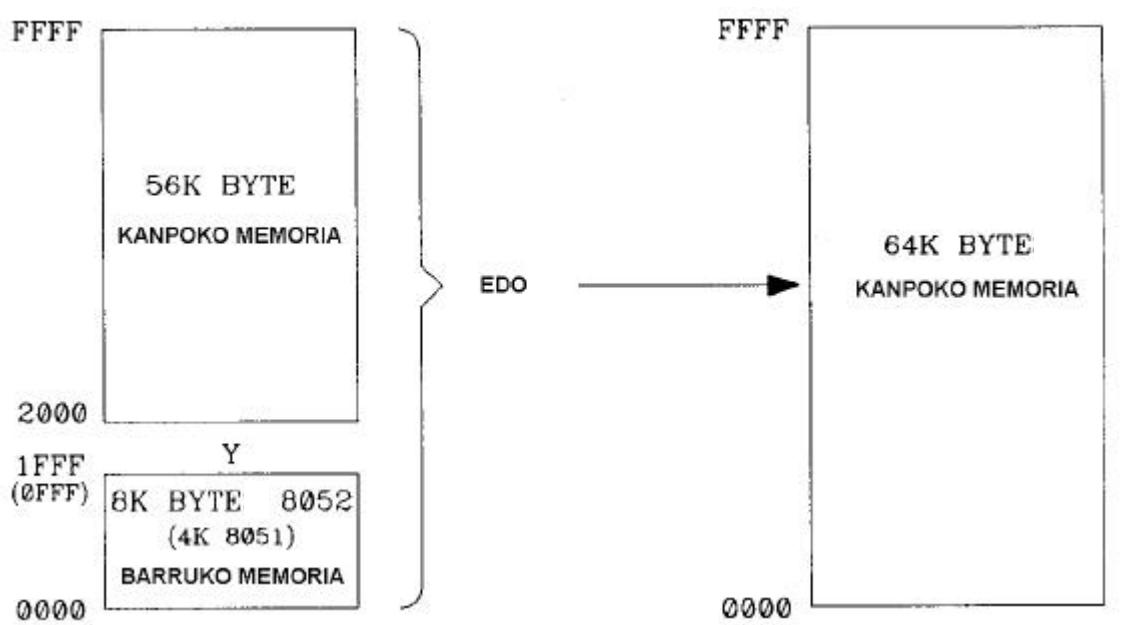


11.6. irudia

### 11.8.2. PROGRAMA-MEMORIA

Ikusi dugunez, 64K izan ditzakegu kanpoan eta 4K/8K barruan. Hemen programa izango dugu, beraz, mikrokontroladoreak hemen dauden aginduak irakurri eta egikaritu egingo ditu. PSEN seinalearekin memoria-mota hau aktibatuko du.

Lehenengo helbideetan, kanpoko eta barruko memoria bereizteko, EA seinalea erabiltzen da.



11.7. irudia

### 11.8.3. DATU-MEMORIA

Hemen ere kanpokoa eta barrukoa ditugu:

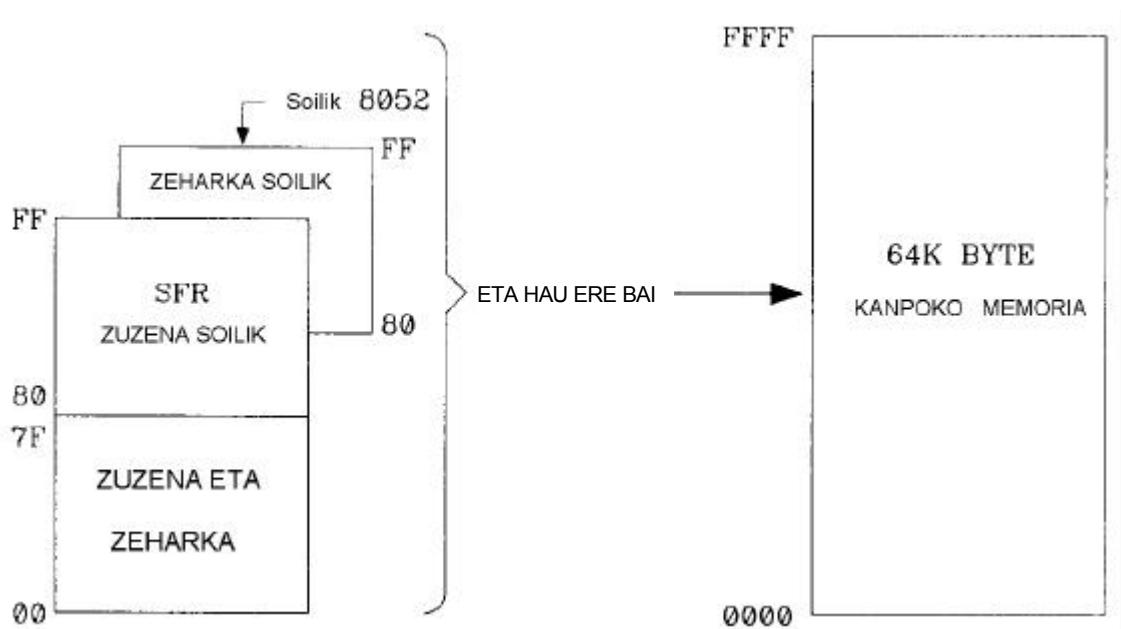
#### 11.8.3.1. Kanpoko datu-memoria

Jakina denez, 64KB ditugu eta, datuak gordetzen direnez, irakurri eta idatzi egin daiteke. Mikrokontroladoreak WR eta RD seinaleak horretarako erabiltzen ditu.

Memoria-mota hau helbideratzeko, MOVX agindua erabiltzen da.

#### 11.8.3.2. Barruko datu-memoria

Barruko memoria 3 bloketan banaturik dago: 128B baxuak, 128B altuak (8052 bakarrik) eta *Funtzio bereziko erregistroen* eremua (*SFR*). Memoria hau 8 bitekin bakarrik helbideratzen da.



11.8. irudia

#### 11.8.3.2.1. Zeharkako helbideratzea bakarrik (8052)

11.8. irudian ikusten dugu *SFR* eremuak ere helbide berberak dituela (80 -FF), baina helbideratzeko modua ezberdina da.

#### 11.8.3.2.2. Helbideratze zuzena eta zeharkakoa (8051/52)

128 byte horiek (00 -7F) hiru bloketan bana ditzakegu.

### ■ Erregistro-bankuak

Lau *banku* ditugu eta banku bakoitzean 8 erregistro (R0-R7). Lau banku horiek, (32 erregistro guztira) lehenengo 32 helbidetan kokatzen dira. *RESET* egin ondoren, 0 bankua aktibatzen da; bankuz aldatzeko, PSW erregistroan ditugu bi bit (RS0 eta RS1).

*SP*, RAM-zati honetan kokatzen da. *RESET* bat egin ondoren, 08 helbidean hasieratzen da (1 bankua R0n), eta handik handitzen joaten da. Banku hori edo besteak erabili ahal izateko, *SP* beste helbide batean hasiera daiteke.



11.9. irudia

### ■ Bitez bit helbideratutako eremua

Zati honetan 16 bit ditugu (20H - 2FH). Guztira 128 bit ditugu (00 -7FH) eta bere izenarekin (00H - 7FH) helbidera ditzakegu, edo bere bytekin (20.0, 20.1, 20.2,...,2F.6, 2F.7).

00H	20.0
01H	20.1
02H	20.2
08H	21.0
09H	21.1
7DH	2F.5
7EH	2F.6
7FH	2F.7

11.3. taula

### ■ Scratch pad eremua

Zati hau (30H-7FH) atzipen azkarreko RAMa da.

#### 11.8.4. FUNTZIO BEREZIKO ERREGISTROAK (SFR)

Area honetan, funtziobereziko erregistroak kokatzen dira. 11.10. irudian, zein diren eta zein helbidetan kokatzen diren ikus daiteke.

8 byte							
F8							
F0	B						
E8							
E0	ACC						
D8							
D0	PSW						
C8	(T2CON)		(RCAP2L)	(RCAP2H)	(TL2)	(TH2)	
C0							
B8	IP						
B0	P3						
A8	IE						
A0	P2						
98	SCON	SBUF					
90	P1						
88	TCON	TMOD	TL0	TL1	TH0	TH1	
80	P0	SP	DPL	DPH			PCON

11.10. irudia

Orain banan-banan aztertuko ditugu:

- A Akumuladorea da.
- B Biderketak eta zatiketak egiteko diseinatuta dago, baina erregistro normal bat bezala erabil daiteke.
- SP **Pilako erakuslea (Stack Pointer)**. 8085ean bezala, baina kasu honetan 8 bitekoa da eta pilan datu bat sartutakoan, gehitu egiten da.

**DPL - DPH** **Datuena erakuslea (Data Pointer)** edo *DPTR* kanpoko RAMarekin transferentziak egiteko erabiltzen da.

**PSW** **Programaren egoera-hitza (Program Status Word)**. Flag-ak ditu, baita erregistro-bankuak kontatzeko bi bit ere.

PSW							
b7	b6	b5	b4	b3	b2	b1	b0
C	AC	F0	RS <sub>1</sub>	RS <sub>0</sub>	OV	-	P

BITA	AZALPENA
b0	<b>P</b> : Akumuladorearen paritatearen flag-a ※ P=1 orduan, akumuladorearen bateko kopurua bakoitia da. ※ P=0 orduan, akumuladorearen bateko kopurua bikoitia da.
b1	<b>-</b> : Libre eta erabilgarria.
b2	<b>OV</b> : Overflow flag-a
b3-b4	<b>RS<sub>0</sub>-RS<sub>1</sub></b> : Erregistro-bankuen hautapena.
	<b>RS<sub>1</sub> RS<sub>0</sub></b> <b>BANKUAK</b>
	0 0      0 bankua (00-07H) 0 1      1 bankua (08-0FH) 1 0      2 bankua (10-17H) 1 1      3 bankua (18-1FH)
b5	<b>F0</b> : Ez du funtziorik. Erabiltzaileak defini dezake.
b6	<b>AC</b> : Carry lagungariaren flag-a.
b7	<b>OV</b> : Carry flag-a.

11.4. taula

**P0, P1, P2, P3** Ataken *latch*-ak dira.

**SBUF** Serie-datuengen bufferra (*Serial data Buffer*).

**TH0-TL0, TH1-TL1, TH2-TL2** Timer erregistroak (*Timers Registers*).

**RCAP2H-RCAP2L** Atzipen-erregistroak (*Capture Registers*).

**PCON** Elikaduraren kontrol-erregistroa (*Power Control Register*). Kontsumo txikiko moduak erregistro honetan jartzen dira. Serieko linearen abiadura ere erregistro honekin alda dezakegu.

**TCON** Timer/kontagailua kontrol-erregistroa (*Timer/Counter Control Register*) (*Timer0 eta timer1*).

**TMOD** Timer/kontagailua moduaren kontrol-erregistroa (*Timer/Counter Mode Control Register*) (*Timer0 eta timer1*).

**T2COM** Timer/kontagailu 2 kontrol-erregistroa (*Timer/Counter 2 Control Register*) (*Timer2*).

**IE** Etenduren gaitze-erregistroa (*Interrupt Enable Register*).

**IP** Etenduren lehentasun-erregistroa (*Interrupt Priority Register*).

**SCON** Serieko atakaren kontrol-erregistroa (*Serial Port Control register*).

## 11.9. HELBIDERATZE-MOTAK

---

Memoria-mota ezberdinak ditugunez, eta askotan bi posizio ezberdinek helbide bera dutenez, bereizi ahal izateko helbideratze-mota ezberdinak erabili beharko ditugu. Hona hemen mikrokontroladore honetan daudenak.

- **Berehalako helbideratza**

Eragigaia zuzenean ematen da aginduan bertan, aurrean "#" jarri. Adibidez:

ADD A,#3BH  $\Rightarrow$  A +3B

- **Helbideratze zuzena**

Aginduan ematen dugu eragigaia dagoen **helbidea**. Era honetan bakarrik helbidera daiteke barruko **RAMa** (lehenengo 128 bytak) eta SFR eremua. Adibidez:

ADD A,**3BH**  $\Rightarrow$  A + barruko memoriaren 3BH helbidean dagoen datua.

- **Erregistro bidezko helbideratza**

Datua erregistro batean dago (R0-R7). Adibidez:

MOV A,**R3**

- **Zeharkako helbideratza**

Helbideratze honetarako erregistro bat erabiltzen da. Erregistro horretan helbide bat dago eta helbide horretan dago eragigaia.

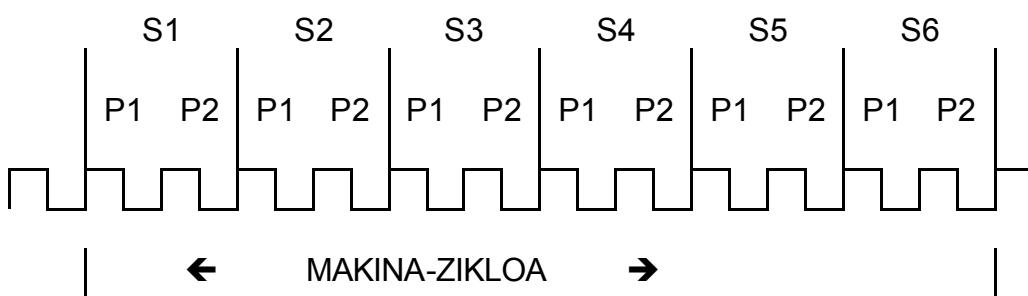
Horrela helbidera daiteke barruko **RAM** osoa, **R0 edo R1** erregistroak (8 bit) erabiliz. Eta horrela helbideratzen da kanpoko **RAMa**, **DPTR** erregistroa (16 bit) erabiliz.

Helbideratze-mota hau “@” ikurrarekin adierazten da. Adibidez:

ADD A,@R0  
MOVX A,@DPTR

## 10.10. MAKINA-ZIKLOA

Mikrokontroladore honetan makina-ziklo batek 6 egoera ditu, eta egoera bakoitzak 2 erloju-periodo edo fase (1fase, 2fase). Beraz, makina-ziklo bakoitzak 12 erloju-periodo ditu, eta, kuartzoa 12MHzekoa bada (normala denez), makina-ziklo baten iraupena mikrosegundo batekoa izango da.



11.11. irudia

## 11.11. INSTRUKZIOAK

Hasieran esan dugun bezala, mikro hau mikiztatzailean edo C-n programa daiteke. Mihiztatzailean programatu ahal izateko, mikro honen aginduak edo instrukzioak zein diren jakin behar dugu.

Hemen daukagu instrukzio horiek erabiltzen dituzten eragigaien esanahia:

Rn .....	Hautatutako bankuen R0-R7 erregistroak.
Direct .....	Barruko RAM helbidea (0-127) edo SFR bat (128-255)
@Ri .....	R1 edo R0-k apuntatutako barruko RAM helbide bat.
#data .....	8 biteko berehalako datu bat.
#data16.....	16 biteko berehalako datu bat.
#addr16.....	16 biteko helbidea (LCALL, LJMP)
#addr11.....	11 biteko helbidea (ACALL, AJMP)
rel .....	Jauzi erlatiboa (-128 +127)
bit .....	Barruko RAMeko bit bat edo SFRko bit bat.

**INSTRUKZIO ARITMETIKOAK**

<b>AGINDUA</b>		<b>AZALPENA</b>	<b>BYTE/Z IKLO</b>	<b>P OV AC C</b>
ADD	A,Rn	$A \leftarrow A + Rn$	1/1	P OV AC C
ADD	A,direct	$A \leftarrow A + direct$	2/1	P OV AC C
ADD	A,@Ri	$A \leftarrow A + @Ri$	1/1	P OV AC C
ADD	A,#data	$A \leftarrow A + #data$	2/1	P OV AC C
ADDC	A,Rn	$A \leftarrow A + Rn + C$	1/1	P OV AC C
ADDC	A,direct	$A \leftarrow A + direct + C$	2/1	P OV AC C
ADDC	A,@Ri	$A \leftarrow A + @Ri + C$	1/1	P OV AC C
ADDC	A,#data	$A \leftarrow A + #data + C$	2/1	P OV AC C
SUBB	A,Rn	$A \leftarrow A - Rn - C$	1/1	P OV AC C
SUBB	A,direct	$A \leftarrow A - direct - C$	2/1	P OV AC C
SUBB	A,@Ri	$A \leftarrow A - @Ri - C$	1/1	P OV AC C
SUBB	A,#data	$A \leftarrow A - #data - C$	2/1	P OV AC C
INC	A	$A \leftarrow A + 1$	1/1	
INC	Rn	$Rn \leftarrow Rn + 1$	1/1	
INC	Direct	$Direct \leftarrow direct + 1$	2/1	
INC	@Ri	$@Ri \leftarrow @Ri + 1$	1/1	
INC	DPTR	$DPTR \leftarrow DPTR + 1$	1/2	
DEC	A	$A \leftarrow A - 1$	1/1	
DEC	Rn	$Rn \leftarrow Rn - 1$	1/1	

DEC	Direct	Direct $\leftarrow$ direct - 1	2/1	
DEC	Ri	$@Ri \leftarrow @Ri - 1$	1/1	
MUL	AB	$BA \leftarrow A \times B$ (A-n uzten ditu pisu handieneko 8 bitak eta B-n pisu txikienekoak)	1/4	P OV C
DIV	AB	A eta B $\leftarrow A / B$ (A-n uzten du emaitzaren zati osoa eta B-n hondakina)	1/4	P OV C
DA	A	Akumuladorearen doikuntza hamartarra BCD kodean. Beti egin beharko da eragiketa baten ostean	1/1	P C

INSTRUUKZIO LOGIKOAK				
AGINDUA		AZALPENA	BYTE/Z IKLO	P OV AC C
ANL	A,Rn	$A \leftarrow A \& Rn$	1/1	P
ANL	A,direct	$A \leftarrow A \& direct$	2/1	P
ANL	A,@Ri	$A \leftarrow A \& @Ri$	1/1	P
ANL	A,#data	$A \leftarrow A \& #data$	2/1	P
ANL	direct,A	$direct \leftarrow direct \& A$	2/1	
ANL	direct,#data	$direct \leftarrow direct \& #data$	3/2	
ORL	A,Rn	$A \leftarrow A (OR) Rn$	1/1	P
ORL	A,direct	$A \leftarrow A (OR) direct$	2/1	P
ORL	A,@Ri	$A \leftarrow A (OR) @Ri$	1/1	P
ORL	A,#data	$A \leftarrow A (OR) #data$	2/1	P
ORL	direct,A	$direct \leftarrow direct (OR) A$	2/1	
ORL	direct,#data	$direct \leftarrow direct (OR) #data$	3/2	
XRL	A,Rn	$A \leftarrow A (XOR) Rn$	1/1	P
XRL	A,direct	$A \leftarrow A (XOR) direct$	2/1	P
XRL	A,@Ri	$A \leftarrow A (XOR) @Ri$	1/1	P
XRL	A,#data	$A \leftarrow A (XOR) #data$	2/1	P
XRL	direct,A	$direct \leftarrow direct (XOR) A$	2/1	
XRL	direct,#data	$direct \leftarrow direct (XOR) #data$	3/2	
CLR	A	$A \leftarrow 00H$	1/1	P
CPL	A	$A \leftarrow \bar{A}$	1/1	P
SWAP	A	$A_{(3..0)} \leftrightarrow A_{(7..4)}$	1/1	
RL	A	Akumuladorean biratu egiten da bitez bit ezkerrerantz	1/1	
RLC	A	Akumuladorean biratu egiten da bitez bit ezkerrerantz carry-a kontuan hartuz. ( $A_0 \leftarrow C$ eta $C \leftarrow A_7$ )	1/1	P C
RR	A	Akumuladorean biratu egiten da bitez bit eskuinerantz	1/1	

RRC	A	Akumuladorean biratu egiten da bitez bit eskuinerantz <i>carry-a</i> kontuan hartuz.(A <sub>7</sub> $\leftarrow$ C eta C $\leftarrow$ A <sub>0</sub> )	1/1	P C
-----	---	--	-----	-----

DATUAK MUGITZEKO INSTRUKZIOAK				
AGINDUA		AZALPENA	BYTE/Z IKLO	P OV AC C
MOV	A,Rn	A $\leftarrow$ Rn	1/1	P
MOV	A,direct	A $\leftarrow$ direct	2/1	P
MOV	A,@Ri	A $\leftarrow$ @Ri	1/1	P
MOV	A,#data	A $\leftarrow$ #data	2/1	P
MOV	Rn,A	Rn $\leftarrow$ A	1/1	
MOV	Rn,direct	Rn $\leftarrow$ direct	2/2	
MOV	Rn,#data	Rn $\leftarrow$ #data	2/1	
MOV	direct,A	direct $\leftarrow$ A	2/1	
MOV	direct,Rn	direct $\leftarrow$ Rn	2/2	
MOV	direct,direct	direct $\leftarrow$ direct	3/2	
MOV	direct,@Ri	direct $\leftarrow$ @Ri	2/2	
MOV	direct,#data	direct $\leftarrow$ #data	3/2	
MOV	@Ri,A	@Ri $\leftarrow$ A	1/1	
MOV	@Ri,direct	@Ri $\leftarrow$ direct	2/2	
MOV	@Ri,#data	@Ri $\leftarrow$ #data	2/1	
MOV	DPTR,data16	DPTR $\leftarrow$ #data16	3/2	
MOVX	A,@Ri	A $\leftarrow$ @Ri	1/2	P
MOVX	A,@DPTR	A $\leftarrow$ @DPTR	1/2	P
MOVX	@Ri,A	@Ri $\leftarrow$ A	1/2	
MOVX	@DPTR,A	@DPTR $\leftarrow$ A	1/2	
MOVC	A,@(A+DPTR)	A $\leftarrow$ @(A+DPTR)	1/2	P
MOVC	A,@(A+PC)	A $\leftarrow$ @(A+PC)	1/2	P
PUSH	direct	Pila $\leftarrow$ direct eta SP=SP+1	2/2	
POP	direct	direct $\leftarrow$ pila eta SP=SP-1	2/2	
XCH	A,Rn	A $\leftrightarrow$ Rn	1/1	P
XCH	A,direct	A $\leftrightarrow$ direct	2/1	P
XCH	A,@Ri	A $\leftrightarrow$ @Ri	1/1	P
XCHD	A,@Ri	A <sub>(3..0)</sub> $\leftrightarrow$ @Ri <sub>(3..0)</sub>	1/1	P

BITEZ BIT ERAGIKETA BOOLEARRAK EGITEKO INSTRUKZIOAK				
AGINDUA		AZALPENA	BYTE/Z IKLO	P OV AC C
ANL	C,bit	$C \leftarrow C \& \text{bit}$	2/2	C
ANL	$C, \overline{\text{bit}}$	$C \leftarrow C \& \overline{\text{bit}}$	2/2	C
ORL	C,bit	$C \leftarrow C (\text{OR}) \text{ bit}$	2/2	C
ORL	$C, \overline{\text{bit}}$	$C \leftarrow C (\text{OR}) \overline{\text{bit}}$	2/2	C
MOV	C,bit	$C \leftarrow \text{bit}$	2/1	C
MOV	Bit,C	$\text{bit} \leftarrow C$	2/2	C
CLR	C	$C \leftarrow 0$	1/1	C
CLR	bit	$\text{bit} \leftarrow 0$	2/1	
SETB	C	$C \leftarrow 1$	1/1	C
SETB	bit	$\text{bit} \leftarrow 1$	2/1	
CPL	C	$C \leftarrow \overline{C}$	1/1	C
CPL	bit	$\text{bit} \leftarrow \overline{\text{bit}}$	2/1	

JAUZIAK EGITEKO INSTRUKZIOAK				
AGINDUA		AZALPENA	BYTE/Z IKLO	P OV AC C
JMP	@A+DPTR	$PC \leftarrow A+DPTR$ (256 byteko orrialdeetan)	1/2	
SJMP	rel	$PC \leftarrow PC+rel$	2/2	
AJMP	Addr11	$PC_{(10..0)} \leftarrow \text{addr11}$ (2KB-eko orrialdeetan)	2/2	
LJMP	Adrr16	$PC \leftarrow \text{addr16}$	3/2	
JZ	adrr	$A=0, \dots, PC \leftarrow \text{adrr}$ $A \neq 0, \dots, PC \leftarrow PC+2$	2/2	
JNZ	adrr	$A \neq 0, \dots, PC \leftarrow \text{adrr}$ $A=0, \dots, PC \leftarrow PC+2$	2/2	
DJNZ	Rn,adrr	Lehendabizi <b>Rn → Rn-1</b> eta: $Rn=0, \dots, PC \leftarrow PC+2$ $Rn \neq 0, \dots, PC \leftarrow \text{addr}$	2/2	
DJNZ	direct,adrr	Lehendabizi <b>direct → direct-1</b> eta: $\text{direct}=0, \dots, PC \leftarrow PC+2$ $\text{direct} \neq 0, \dots, PC \leftarrow \text{addr}$	2/2	

JC	adrr	C=0,..... PC ← adrr C ≠ 0,.... PC ← PC+2	2/2	
JNC	adrr	C ≠ 0,.... PC ← adrr C=0,..... PC ← PC+2	2/2	
JB	Bit,adrr	bit=0,..... PC ← adrr bit ≠ 0,.... PC ← PC+2	3/2	
JNB	Bit,adrr	bit ≠ 0,.... PC ← adrr bit=0,..... PC ← PC+2	3/2	
JBC	Bit,adrr	bit=0,..... PC ← adrr bit ≠ 0,.... PC ← PC+2 eta bi kasuetan bit ← 0 (clear bit)	3/2	
CJNE	A,direct,addr	A eta direct konparatzen ditu. Berdinak ez badira, jauzi egiten du. Gainera: If A < direct, C ← 1 If A ≥ direct, C ← 0	3/2	C
CJNE	A,#data,addr	A eta #data konparatzen ditu. Berdinak ez badira, jauzi egiten du. Gainera: If A < #data, C ← 1 If A ≥ #data, C ← 0	3/2	C
CJNE	Rn,#data,addr	Rn eta #data konparatzen ditu. Berdinak ez badira, jauzi egiten du. Gainera: If Rn < #data, C ← 1 If Rn ≥ #data, C ← 0	3/2	C
CJNE	@Ri,#data,addr	@Ri eta #data konparatzen ditu. Berdinak ez badira, jauzi egiten du. Gainera: If @Ri < #data, C ← 1 If @Ri ≥ #data, C ← 0	3/2	C

AZPIRRUTINA-DEIAK ETA ITZULERAK EGITEKO INSTRUKZIOAK				
AGINDUA		AZALPENA	BYTE/Z IKLO	P OV AC C
ACALL	Addr11	Jauzi egin baino lehen, egikaritu beharko lukeen hurrengo instrukzioaren helbidea pilan gordetzen da. Eta gero: $PC_{(10..0)} \leftarrow \text{addr11}$ (2 Kbyteko orrialdeetan)	2/2	

LCALL	Addr16	Jauzi egin baino lehen, egikaritu beharko lukeen hurrengo instrukzioaren helbidea pilan gordetzen da. Eta gero: PC $\leftarrow$ addr16 (64 Kbytetan. Memoria osoan)	3/2	
RET		Azpirrutina batek instrukzio honekin bukatu beharko du. Jauzi egin baino lehen pilan gordetako helbidea berreskuratuko du mikroak. PC $\leftarrow$ pilan dagoen helbidea	1/2	
RETI		Etendura bateko azpirrutina batek instrukzio honekin bukatu beharko du. Etendurara jauzi egin baino lehen pilan gordetako helbidea berreskuratuko du mikroak. PC $\leftarrow$ pilan dagoen helbidea	1/2	
NOP		Ezer egiten ez duen instrukzioa.	1/1	

## 11.12. TIMER-AK/KONTAGAILUAK

### 11.12.1. SARRERA

8051 kontroladoreak bi *Timer/kontagailu* ditu: T0 eta T1. 8052ak beste bat T2. Kapitulu honetan T0 eta T1 aztertuko ditugu. Biak funtzionamendu berdina daukate, beraz, bakarra ikusiko dugu.

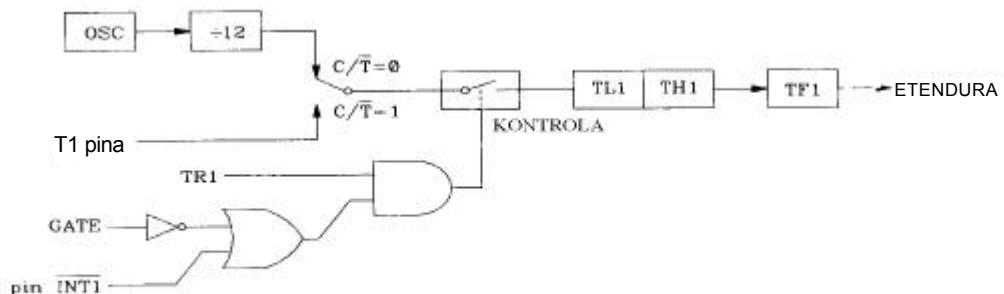
### 11.12.2. TIMER0 (T0) ETA TIMER1 (T1)

*Timer0-k* bi erregistro ditu kontatzeko: TH0 eta TL0. Horietan kontatu behar den balioa gordeko da. TMOD erregistroan, bi *timer* horien lan egiteko modua programatzen da.

Temporizadore bezala lan egiten badu, barruko erlojuaren pultsuak honela kontatzen ditu: **hamabi pultsuko unite bat**.

Kontagailu bezala lan eginez gero, T0 (15. pin) kanpoko pinean, beheranzko saihetsak kontatzen ditu.

11.12. irudian, laneko aukerak dagozkien kontrolekin ikusten dira.



11.12. irudia

C / $\bar{T}$ -arekin, *timer-a* edo *kontagailua* hautatzen dugu. Beheko partean, zirkuitu logiko bat dago, horrek kontatzen uzten du edo ez. Bi aukera ematen dizkigu:

- ❖ SOFT-en bidez, horretarako GATE=0 eta TR0=1
- ❖ HARD-en bidez, kasu honetan, GATE =1 eta TR0=1 kontrola INT0 pinarekin eginez.

TL0 eta TH0 erregistroek kontatzen den balioa edukiko dute. Balio hori, kontatu ahala, gehituz joango da.

Zenbaketa horretan *overflowgertatzen* denean (FFFFtik 0000ra pasatzean), **TF0 flag**-bita aktibatzen da. *Timer-a* programa daiteke, *overflow*-arekin batera etendura sor dezan.

TMOD							
b7	b6	b5	b4	b3	b2	b1	b0
GATE	C / $\bar{T}$	M1	M0	GATE	C / $\bar{T}$	M1	M0
Timer1				Timer0			

BITA	AZALPENA												
	<u>MODUA</u>	<u>M1</u>	<u>M0</u>	<u>ERAGIKETA</u>									
b0-b1	0	0	0	13 biteko temporizadorea									
	1	0	1	16 biteko temporizadorea									
	2	1	0	8 biteko temporizadorea autobirkargarekin									
	3	1	1	temporizadore gehiago									
b2	<b>C / T̄</b> : Temporizadorea edo zenbatzailea hautatzen du. * <b>C / T̄ =0</b> orduan, barruko erlojuaren pultsuekin lan egingo du temporizadore moduan. * <b>C / T̄ =1</b> orduan, T0 pinetik iristen diren pultsuak zenbatzen ditu.												
b3	<b>GATE</b> : INT0 kanpoko sarrera gaitzen du. * <b>GATE =1</b> orduan, INT0 gaitzen du, baldin eta TR0=1 bada. * <b>GATE =0</b> orduan, INT0 desgaitzen du, beraz, TR0-k kontrolatuko du temporizadorea (software-kontrola).												
b4,b5 b6,b7	<i>Timer1-en konfigurazioa. Timer0-rena bezalakoa baina erregistro hauek ordezkatuz:</i> T0 → T1 INT0 → INT1 TR0 → TR1												

### 11.5. taula

11.5. taulan ikusten denez, kontrol batzuk TMOD erregistroan daude. Besteak TCON erregistroan, TR eta TF besteak beste. Erregistro hori, gainera, bitez bit helbidera daiteke.

TCON							
b7	b6	b5	b4	b3	b2	b1	b0
<b>TF1</b>	<b>TR1</b>	<b>TF0</b>	<b>TR0</b>	<b>IE1</b>	<b>IT1</b>	<b>IE0</b>	<b>IT0</b>

BITA	AZALPENA
b0	<b>IT0</b> : kanpoko 0 etenduraren kontrola. ※ <b>IT0 =0</b> orduan, maila baxuaz aktibatzen da. ※ <b>IT0 =1</b> orduan, beheranzko saihetsez aktibatzen da.
b1	<b>IE0</b> : kanpoko 0 etenduraren <i>flag-a</i> . ※ kanpoko 0 etendura sortzen denean, 1ean jartzen da. ※ Etendura saihetsez aktibatzen bada, etendura bukatu ondoren automatikoki 0an jartzen da.
b2	<b>IT1</b> : kanpoko 1 etenduraren kontrola.
b3	<b>IE1</b> : kanpoko 0 etenduraren <i>flag-a</i> .
b4	<b>TR0</b> : 0 temporizadorea abian jartzen du edo geldiarazi egiten du. ※ <b>TR0 =1</b> orduan, 0 temporizadorea abian jartzen du. ※ <b>TR0 =0</b> orduan, 0 temporizadorea geldiarazten du.
b5	<b>TF0</b> : <i>Timer0-k zenbaketa bukatu duela adierazten duen flag-a</i> . Automatikoki 0an jartzen da etendura bukatu ondoren.
b6	<b>TR1</b> : 1 temporizadorea abian jartzen du edo geldiarazi egiten du.
b7	<b>TF1</b> : <i>Timer1-ek zenbaketa bukatu duela adierazten duen flag-a</i> .

11.6. taula

### 11.12.3. TIMER-EN LAN EGITEKO MODUAK

#### 11.12.3.1. 0 Modua. 13 biteko timer/kontagailua

Modu honetan, konta daitekeen balorerek handiena 13 bitekin zehazten da ( $2^{13} = 8192 = 1FFFH$ ). 8 bit TH0an eta beste 5 TL0an. TL0ko beste hirurak ez dira erabiltzen.

Modu hau ez da ia erabiltzen. Aurreko mikroarekin (8048) bateragarritasuna ez galtzeko sartu zuten.

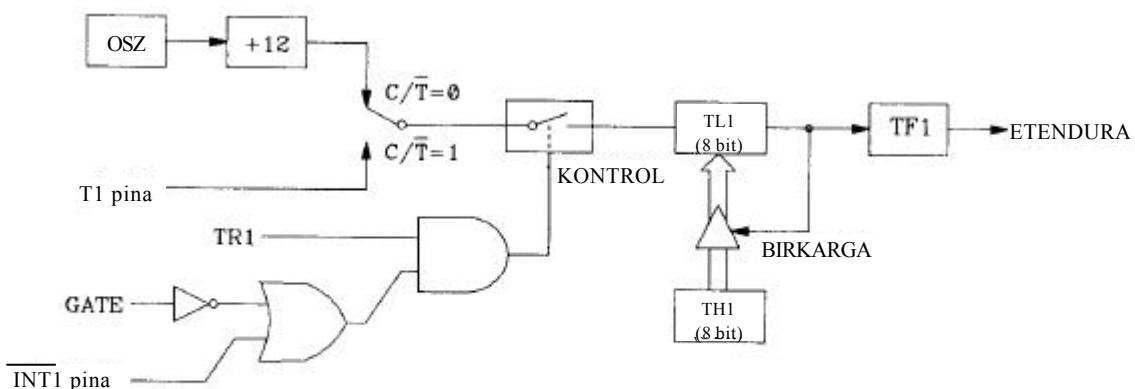
### 11.12.3.2. 1 Modua. 16 biteko *timer/kontagailua*

Kasu honetan, 16 bitak erabiltzen direnez,  $2^{16} = 65536 = \text{FFFFH}$ -raino konta daiteke.

Honek zenbaketa bakarra egingo du, eta gero geldirik geratuko da. Beste zenbaketa berri bat egiteko, berriro balioak kargatu beharko ditugu.

### 11.12.3.3. 2 Modua. 8 biteko *timer/kontagailua autobirkargarekin*

Modu honetan, TL0-k zenbatzen du eta TH0-k zenbatu behar den balioa gordetzen du. TL0 erregistroan *overflow*gertatzean, TF0 aktibatzeaz gain, TH0-k duen balioa TL0-an automatikoki kargatzen da. Karga hori kopia baino ez da, beraz, TH0-k beti gordeko du programatutako balioa.



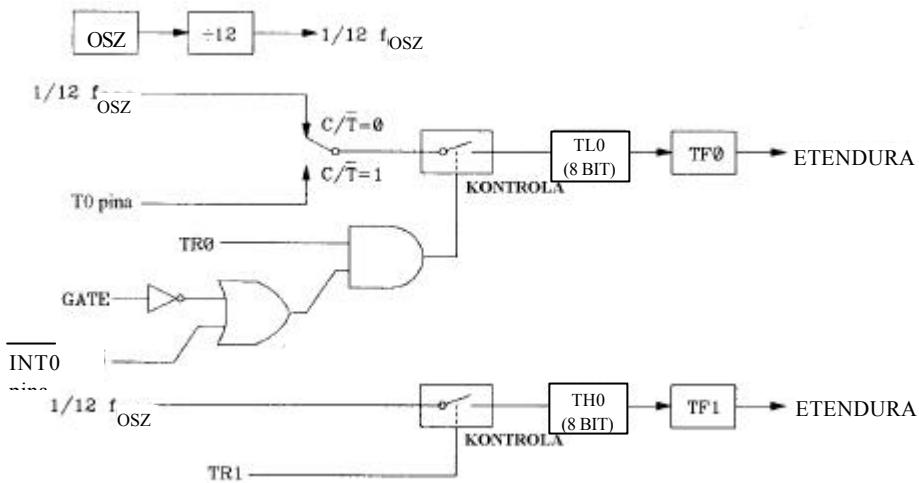
11.13 irudia

### 11.12.3.4. 3 Modua. *timer/kontagailu bat gehiago lortzeko*

Kasu honetan, TL0 eta TH0 8 biteko bi kontagailu independente dira. TL0 *timer-a* edo *kontagailua* izan daiteke, eta *timer0*-aren kontrolak erabiltzen ditu.

TH0, aldiz, *timer* baino ezin da izan eta, horrez gain, *timer* bateko kontrol batzuk erabiltzen ditu.

Horrela, *timer1* muga batzuekin gelditzen da, adibidez, ezin izango ditu etendurak sortu.



11.14. irudia

## 11.13. ETENDURAK

### 11.13.1. SARRERA

Etendurak beharrezkoak dira,  $\mu$ Kontroladoreak periferikoak dituenean horiekin batera lan egiteko.

Jo dezagun  $\mu$ K batek prozesu industrial bat kontrolatzen duela. Prozesu horretan piezak zulatu egin behar dira, baina hamar egin eta gero, txirbilak haizez garbitu eta kutxa aldatu behar da. Kanpoan kontagailu bat egongo da piezak kontatzeko. Egoera horretan, mikroak nola jakingo du hamargarren piezara iritsi direla? Bi aukera daude:

- Periferikoari galdetu. Hori sarritan egin behar izaten da. (*polling* konsultak soft-en bidez).
- Periferikoak 10 pieza zenbatu dituenean etendura bat sortu.

Ikusten denez, etendurak erabiliz, mikroak lan zehatzak uzten dizkie periferikoei. Horrela, bera beste gauza batzuez arduratzen da.

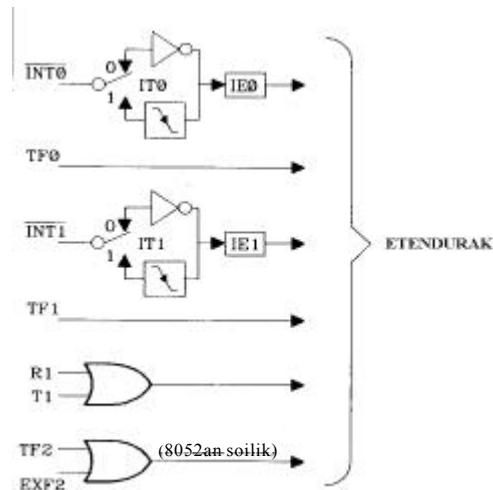
### 11.13.2. ETENDURA-MOTAK

Etendura mikrotik kango dagoen elementu batek sor dezake, edo mikroaren barruko elementu batek.

Mikrokontroladoreak etendura bat onartzen duenean, esku artean zeukan lana alde batera uzten du geldirik une batez eta etendurari dagokion azpiprograma egikaritzen du. Hori bukatu eta gero ekiten dio berriro aurreko lanari. Aipatutako azpiprograma memoria-posizio zehatz batean dago.

8051 kontroladoreak bost etendura-iturri ditu (8052ak sei).

- 2 kanpokoak  $\overline{INT0}$  eta  $\overline{INT1}$
- 2 barrukoak. *Timer0* eta *Timer1*-eko (8052an *Timer2*-koa ere badago)
- 1 barrukoa. Serieko lineakoa.



11.15. irudia

11.7. taulan, etendura bakoitzari dagokion memoria-helbidea ikusten dugu. Helbide horietatik aurrera programak idatzi beharko ditugu. Programa horietan, mikroak etendura jasotzen duenean zer egin beharko duen adieraziko dugu.

HELBIDEEN TAULA	
ITURRIAK	HELBIDEAK
$\overline{INT0}$	0003H
$TIMER0$	000BH
$\overline{INT1}$	0013H
$TIMER1$	001BH
RI + TI	0023H
$TIMER2$	002BH

11.7. taula

Une kritikoetan, ezin izango dugu mikroa eten. Lan garrantzitsu edo arriskutsuak egiten ari denean, hain zuzen.

Hori dela eta, etendurak maskaratu egin daitezke. Horrela, nahiz eta etendura eskatu, mikroak ez dio kasurik egingo. Etendura horiek mikroak egikaritzeko prest egon arte itxarongo dute.

Hau **IE** erregistroan programatzen da. Bertan, guztiak batera edo banaka maskara daitezke.

Kanpoko etendurak TCON erregistroan programatzen dira.

<b>IE</b>							
b7	b6	b5	b4	b3	b2	b1	b0
<b>EA</b>	X	<b>ET2</b>	<b>ES</b>	<b>ET1</b>	<b>EX1</b>	<b>ET0</b>	<b>EX0</b>

<b>BITA</b>	<b>AZALPENA</b>	
b0	<b>EX0:</b>	* EX0 =1 INT0 kanpoko etendura gaitzen du. * EX0 =0 INT0 kanpoko etendura desgaitzen du.
b1	<b>ET0:</b>	* ET0 =1 Timer0-aren etendura gaitzen du. * ET0 =0 Timer0-aren etendura desgaitzen du.
b2	<b>EX1:</b>	* EX1 =1 INT1 kanpoko etendura gaitzen du. * EX1 =0 INT1 kanpoko etendura desgaitzen du.
b3	<b>ET1:</b>	* ET1 =1 Timer1-aren etendura gaitzen du. * ET1 =0 Timer1-aren etendura desgaitzen du.
b4	<b>ES:</b>	* ES =1 Serieko atakaren etendura gaitzen du. * ES =0 Serieko atakaren etendura desgaitzen du.
b5	<b>ET2:</b>	* ET2 =1 Timer2-aren etendura gaitzen du. * ET2 =0 Timer2-aren etendura desgaitzen du.
b6	Ez da erabiltzen.	
b7	<b>EA:</b>	* EA =1 etendura guztiak gaitzen ditu (bakarka gaituak izan badira). * EA =0 etendura guztiak desgaitzen ditu.

11.8. taula

### 11.13.3. ETENDUREN PROZESUA

Makina-ziklo bakoitzean, etenduraren bat dagoen begiratzen du mikroak. Horrela bada, eta **IE** registroan gaituta badago, horrela jokatzen du:

1. Egikaritzen ari den instrukzioarekin bukatzen du.
2. Programa nagusian, egikaritu beharko lukeen hurrengo instrukzioaren helbidea pilan gordetzen du, etendurari kasu egin eta gero, helbide horretara itzuli beharko baitu.
3. Etendurari dagokion helbidera (ikus 11.9. taula) joaten da eta bertan dagoen programa egikaritzen du. Programa horrek **RETI** instrukzioarekin bukatuko du.
4. Mikroak RETI instrukzioa aurkitzen duenean, lehen gordetako helbidea berreskuratzen du pilatik eta programa nagusia egikaritzen jarraitzen du.

Etendura bakoitzak *flag* bat du (bit bat). Bit hori batean badago, etendura hori eskatzen ari dela adierazten du.

ETENDUREN ITURRIAK	FLAG-A	HELBIDEA
0 kanpoka INT0	IE0	0003H
Timer0 TIMER0	TF0	000BH
1 kanpoka INT1	IE1	0013H
Timer1 TIMER1	TF1	001BH
Serieko ataka RI	RI	0023H
Serieko ataka TI	TI	0023H
Timer2 TIMER2	TF2	002BH
Timer2 / 2 kanpoka T2EX	EXF2	002BH

11.9. taula

Etendurari kasu egin eta gero, *flag* hori ezabatu egin beharko da beste etendura bat sortu ahal izateko. *Flag* batzuk automatikoki ezabatzen dira, mikroak etendurari kasu egin eta gero (**hardware-ezabapena**). Beste batzuk ez dira automatikoki ezabatzen, beraz, guk programan ezabatu behar dugu (**software-ezabapena**).

Hardware-ezabapena	Software-ezabapena
Timer0	Timer2
Timer1	Serieko ataka
Int0	
Int1	

11.10. taula

#### 11.13.4. ETENDUREN LEHENTASUN-MAILAK

Askotan, etendura bat baino gehiago egon daitezke aldi berean aktibaturik. Kasu horietan, lehentasun-ordena bat behar dugu. Mikroak berak lehenetsitako lehentasunarekin lan egiten du. Hauxe da:

Lehentasuna	Etendura
(handiena) 1	INT0
2	TIMER0
3	INT1
4	TIMER1
5	SERIEKO ATAKA
(txikieta) 6	TIMER2

11.11. taula

Dena den, markatuta datorren lehentasuna alda dezakegu **IP** erregistroa programatzuz. Horrek bi lehentasun-maila bereizten ditu, lehentasun handia eta lehentasun txikia.

IP (INTERRUPT PRIORITY ERREGISTROA)							
b7	b6	b5	b4	b3	b2	b1	b0
X	X	PT2	PS	PT1	PX1	PT0	PX0

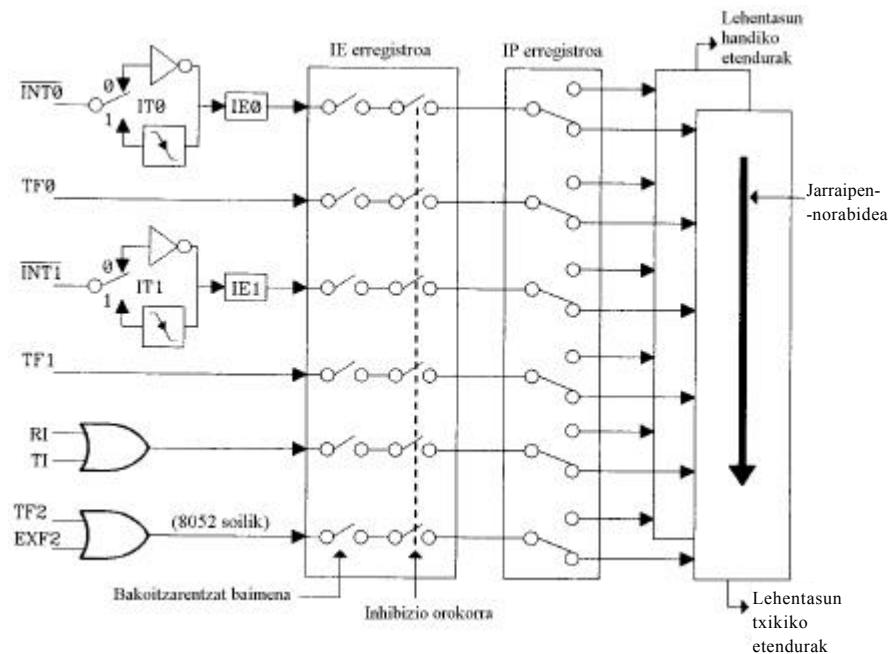
BITA	AZALPENA
b0	<b>PX0</b> : PX0 =1 INT0 etendurari lehentasun handia esleitzten dio.
b1	<b>PT0</b> : PT0 =1 Timer0-aren etendurari lehentasun handia esleitzten dio.
b2	<b>PX1</b> : PX1 =1 INT1 etendurari lehentasun handia esleitzten dio.
b3	<b>PT1</b> : PT1 =1 Timer1-en etendurari lehentasun handia esleitzten dio.
b4	<b>PS</b> : PS =1 Serieko atakaren etendurari lehentasun handia esleitzten dio.
b5	<b>PT2</b> : PT2 =1 Timer2-ren etendurari lehentasun handia esleitzten dio.
b6	Ez da erabiltzen.
b7	Ez da erabiltzen.

11.12. taula

Mikroa lehentasun handia dutenekin hasten da eta gero lehentasun txikiago-koarekin jarraitzen du.

Lehentasun-maila berdina dutenen artean, lehenetsitako lehentasuna aplikatzen da.

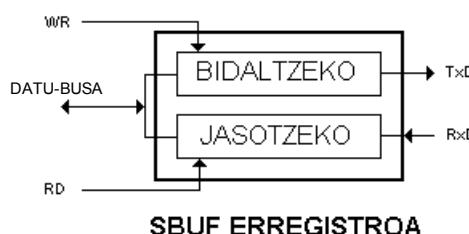
11.16. irudian, inhibizioak eta lehentasun-mailak irudikatzen dira.



11.16. irudia

## 11.14. SERIEKO LINEA

Mikrokontroladore honek serieko linea bat dauka integraturik. *Erabateko duplex* moduan lan egiten du, hau da, datuak jaso eta bidali egin dezake aldi berean. Datuak bidali eta jaso SBUF erregistroaren bidez egiten da: datu bat bidaltzeko erregistro horretan kargatzen da eta jasotako datua erregistro horretatik irakurri beharko da.



11.17. irudia

Lan egiteko lau modu ezberdin onartzen ditu:

## **0 MODUA**

Modu honetan transmisioa sinkronoa da. Datuak RxD pinetik sartu eta ateratzen dira, eta datu bakoitzeko 8 bit baino ez dira bidali behar. TxD pinetik erloju-pultsuak ateratzen dira. Transmisió-abiadura finkoa da: mikrokontroladorearen erlojuaren maiztasuna zati 12.

## **1 MODUA**

Gehien erabiltzen den modu honek transmisió asinkronoa erabiltzen du. Datuak TxD pinetik bidaltzen dira eta RxD pinetik jasotzen dira. 10 bit behar ditu datu bat bidaltzeko edo jasotzeko:

- ※ Bit bat, START (0), transmisióaren hasiera adierazteko (transmisiórik gabe linea batean dago).
- ※ Datuko 8 bit (pisu txikieneko bitarekin □LSB□ hasten da).
- ※ Bit bat, STOP (1), transmisióaren bukaera adierazteko.

Transmisióaren abiadura askoren artean aukera daiteke.

## **2 MODUA**

Modu honetan datuak TxD pinetik bidaltzen dira eta RxD pinetik jasotzen dira. 11 bitez osatutako kateak transmititzen dira:

- ※ Bit bat, START (0), transmisióaren hasiera adierazteko.
- ※ Datuko 8 bit (pisu txikieneko bitarekin □LSB□ hasten da).
- ※ Datuko bit bat: 9. bita (programagarria).
- ※ Bit bat, STOP (1), transmisióaren bukaera adierazteko.

Kasu honetan bi abiadura aukera daitezke: osziladorearen 1/32 edo 1/64.

## **3 MODUA**

Modu hau 2 modua bezalakoa da, baina abiadura askoren artean hauta daiteke.

2 modua eta 3 modua mikrokontroladoreen arteko komunikazioa egiteko erabiltzen dira, maisu-morroi egiturarekin lan egiteko, hain zuzen ere. Guk gehien erabiltzen den 1 modua aztertuko dugu.

### 11.14.1. SCON SERIEKO LINEAREN KONTROL-ERREGISTROA

SCON (*Serial Port Control Register*) erregistroak serieko linea programatzen du. SFR erregistroa izanik, hurrengo hauek kontrolatzen ditu:

SCON (SERIAL PORT CONTROL REGISTER)							
b7	b6	b5	b4	b3	b2	b1	b0
<b>SM0</b>	<b>SM1</b>	<b>SM2</b>	<b>REN</b>	<b>TB8</b>	<b>RB8</b>	<b>TI</b>	<b>RI</b>

BITA	AZALPENA															
b0	<b>RI</b> : Jasotzeko etenduren <i>flag</i> -a. Azken bita jasotzen denean, automatikoki aktibatzen da.															
b1	<b>TI</b> : Bidaltzeko etenduren <i>flag</i> -a. Azken bita bidaltzen denean, automatikoki aktibatzen da.															
b2	<b>RB8</b> : 2 eta 3 moduetan jasotzen den 9. bita da. 1 moduan eta SM2 bitak 0 balioa duenean, jasotako datuaren STOP bita da. 0 moduan ez da erabiltzen.															
b3	<b>TB8</b> : 2 eta 3 moduetan bidaltzen den 9. bita da. Paritate-bitak izan ohi da.															
b4	<b>REN</b> : <b>REN =1</b> , datuak jaso daitezke. <b>REN =0</b> , datuak ezin dira jaso.															
b5	<b>SM2</b> : 2 eta 3 moduetan aktibatuta badago eta 9. bita zeroa bada, RI flag-a ez da piztuko. 1 moduan aktibatuta badago eta STOP bita jasotzen ez bada, RI ez da aktibatuko. 0 moduan zeroan egon behar du.															
b6-b7	<b>SM0-SM1</b> : Lan egiteko modua hautatzeko erabiltzen dira  <table style="margin-left: auto; margin-right: auto;"> <tr> <th>SM0</th> <th>SM1</th> <th>MODUA</th> </tr> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>2</td> </tr> <tr> <td>1</td> <td>1</td> <td>3</td> </tr> </table>	SM0	SM1	MODUA	0	0	0	0	1	1	1	0	2	1	1	3
SM0	SM1	MODUA														
0	0	0														
0	1	1														
1	0	2														
1	1	3														

11.12. taula

### 11.14.2. PCON KONSUMOA KONTROLATZEKO ERREGISTROA

PCON (POWER CONTROL REGISTER)							
b7	b6	b5	b4	b3	b2	b1	b0
<b>SM0D</b>				<b>GF1</b>	<b>GF2</b>	<b>PD</b>	<b>IDL</b>

SMOD bitak serieko linearen abiaduran eragina dauka 1, 2 eta 3 moduetan. GF1 eta GF2 ez daude definituak, eta erabiltzaileak nahi duenarako erabil ditzake. PD eta IDL *Power Down* eta *Idle*, hurrenez hurren, kontsumo gutxiko moduetan sartzeko erabiltzen dira.

#### 11.14.2.1. *IDLE* modua

IDL bita aktibatzen duen instrukzioa izango da egikaritzen den azkena, izan ere, mikrokontroladorea gelditu egiten da eta mikroa *IDLE* moduan sartu dela esaten da.

Modu honetan, barruko erloju-seinalea ez da PUZera iristen, baina bai serieko linea, *timer* eta etenduretara. Erregistro eta ataka guztiak beren balioak mantentzen dituzte.

Bi modu daude egoera honetatik ateratzeko (mikroa esnatzea), etendura baten bidez edo reset eginez. Mikroa esnatzen denean, gelditu zen lekutik hasita jarraitzen du egikaritzen.

#### 11.14.2.2. *Power down* modua

Modu honetan, erlojua gelditu egiten da. Barruko memoria, erregistro eta ataka guztiak beren balioak mantentzen dituzte. Modu honetan elikatzeko tentsioa 2 voltera jaits daiteke. Modu honetatik ateratzeko era bakarra reset egitea da.

### 11.14.3. KOMUNIKAZIOAREN ABIADURA

Komunikazio-abiadura bandetan, hau da, bit/s-tan, adierazten da. 0 moduan hau da:

$$\text{BAUDAK 0 moduan} = \frac{\text{osziladorearen maiztasuna}}{12}$$

2 moduan, abiadura PCON erregistroaren 7. bitak (SMOD) zehazten du. Bi aukera daude, erlojuaren maiztasunaren 1/32 edo 1/64:

$$\text{BAUDAK 2 moduan} = \frac{2^{\text{SMOD}}}{64} \text{ osziladorearen maiztasuna}$$

1 eta 3 moduetan 1. *timer*-a erabiltzen da abiadura zehazteko.

#### 11.14.3.1. 1. *Timer*-a baud-sortzaile gisa

1. *timer*-a erabiltzen denean (1 eta 3 moduetan) transmisio-abiadura kontrolatzeko, *timer*-aren karga-balioak markatzen du abiadura. *Timer*-a kontagailu edo temporizadore bezala eta hiru moduetan programa daiteke, hala ere, aplikazio gehienetan **temporizadorea eta 2 moduan** (birkargarekin) erabiltzen da. Kasu horretan, formula hau da:

$$\text{BAUDAK 1 eta 3 moduetan} = \frac{2^{\text{SMOD}}}{32} \times \frac{\text{osziladorearen maiztasuna}}{12 \times (256 - (\text{TH1}))}$$

11.13. taulan, abiadura arruntenak lortzeko, *timer*-aren karga-balioak ikus daitezke.

BAUDAK	XTAL	SMOD	1. TIMER-A		
			C/T	MODUA	KARGA
62.500	12MHz	1	0	2	FFH
19.200	12MHz	1	0	2	FDH
9.600	12MHz	0	0	2	FDH
4.800	12MHz	0	0	2	FAH
2.400	12MHz	0	0	2	F4H
1.200	12MHz	0	0	2	E8H

11.13. taula

## ***Ariketa ebatziak mihiztatzailean***

Praktika hauek DISEN-51 entrenatzailearekin eginak daude. Beraz, horren memoria-mapa eta beste zenbait berezitasun kontuan hartuz egin dira. Beste ekipo bat erabiliz gero, agian aldaketa txiki batzuk egin beharko dira.

### **1. PRAKTIKA (BARRUKO RAM MEMORIA)**

Barruko RAM memo riako bi helbidetan (34H eta 35H) dauden bi datu hamartar (guk sartu behar ditugu programa egikaritu baino lehen) batu behar dira. Emaitza 36H helbidean utzi. Zeharkako helbideratzea erabiliko dugu.

```

INCLUDE      MACROS.INC

ORG          00H
SJMP         INICIO

INICIO
ORG          33H
MOV          SP,#B9H      ;Stack pointer hasieratu
MOV          R0,#35H      ;R0 eta R1 helbideak izango dituzte
MOV          R1,#36H
MOV          A,@R1
ADD          A,@R0
DA           A            ;Akumuladorearen doikuntza hamartarra
MOV          R0,#36H
MOV          @R0,A        ;Emaitza utzi
TERMINA
END

```

### **2. PRAKTIKA (KANPOKO RAM MEMORIA)**

Kanpoko RAM memoriako bi helbidetan (1134H eta 1135H) dauden bi datu hamartar (guk sartu behar ditugu programa egikaritu baino lehen) batu behar dira. Emaitza 1136H helbidean utzi.

```

INCLUDE      MACROS.INC

ORG          00H
SJMP         INICIO

INICIO
ORG          33H
MOV          SP,#B9H
MOV          DPTR,#1134H   ;DPTR lehendabiziko helbidearekin kargatu
MOVX         A,@DPTR      ;Lehenengo datua akumuladorera
MOV          R7,A          ;R7n gorde
INC           DPTR
MOVX         A,@DPTR
ADD          A,R7
DA           A
INC           DPTR

```

```

MOVX      @DPTR,A      ;Emaitza utzi
TERMINA
END

```

### 3. PRAKTIKA (ARGI-JOKOA)

P1 ataka 8 ledetara konektatuko dugu. Led diodo bat piztuko da eta desplazatuz joango da. Atzerapena ipini beharko da begi-bistan ikusi ahal izateko.

```

ORG      00H
SJMP    INICIO

INICIO   ORG      33H
        MOV      A,#01H
ZIKLO    MOV      P1,A      ;Datua atakatik atera
        ACALL   ATZERA
        RL       A      ;Akumuladorearen biraketa ezkerrera
        SJMP    ZIKLO

```

Azpirrutina honek gutxi gorabehera 0,1 segundoko atzerapena sortzen du.

```

ATZERA  MOV      R7,#FFH
        MOV      R6,#FFH
GEHIO    DJNZ   R7,JARRAITU
        RET
JARRAITU DJNZ   R6,JARRAITU
        SJMP   GEHIO
END

```

### 4. PRAKTIKA (SERIEKO LINEA)

Teklatuan OK sakatzen denean, esaldi bat bidaliko da mikrotik ordenagailuaren pantailara.

```

INCLUDE MACROS.INC

ORG      0
SJMP    INICIO

INICIO   ORG      33
        MOV      SCON,#54H      ; Serieko linea hasieratu. Modu1, hartzea
                                ; baimenduta, RB8 = stop eta RI=TI=0.
        MOV      TMOD,#20H      ; 1 timer-a hasieratu. Timer, 2 moduan, soft
                                ; kontrola
        MOV      PCON,#80H      ; SMOD=0
        MOV      TH1,#FDH      ; 9600 baudak lortzeko.
        SETB   TR1      ; Timer-a abian jartzeko.
        MOV      DPTR,#TEXTO      ; DPTRn kargatzten dugu testua hasten den
                                ; helbidearekin
CICLO1  JNB      RI,$      ; Datu bat jaso ez bada, bertan ($) itxaron.
        MOV      A,SBUF      ; SBUF irakurri
        CLR      RI      ; RI ezabatu gehiago jaso ahal izateko.

```

	CJNE	A,#52H,CICLO1	; R letra ez bada, itxaron beste bat jaso arte.
	MOVX	A,@DPTR	; Lehenengo letra akumuladorera.
TRANS	JNB	TI,\$	; Itxaron transmisorea prest egon arte.
	CLR	TI	; Hurrengo datua bidali ahal izateko.
	MOV	SBUF,A	; Bidaltzen dugu.
	INC	DPTR	; Hurrengo karakterea akumuladorera.
	MOVX	A,@DPTR	
	CJNE	A,#00H,TRANS	; Azkena ez bada, bidaltzera joango gara.

TERMINA

TEXTO DB 'TESTUA BIDALTZEKO PROBA',0AH,0DH,00H

END

## ***Proposatutako ariketak***

### **1. PRAKTIKA (BIDERKETA ETA ZATIKETA)**

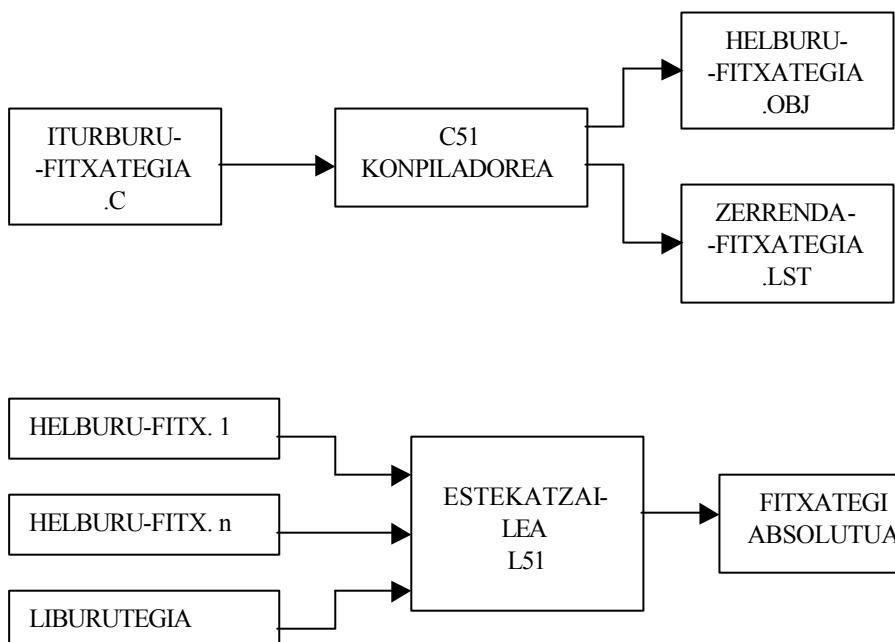
Kanpoko RAM memoriaren 1100H eta 1101H helbideetan dauden bi datu biderkatu eta zatitu. Biderketaren emaitza 1102H eta 1103H helbideetan utzi eta zatiketaren emaitza 1104H eta 1105H helbideetan utzi.

### **2. PRAKTIKA (SARRAILA AUTOMATIKOA)**

Alez kanpoko pantaila batean (ordenagailuarenean) “ATEA ITXITA” mezua azalduko zaigu. Teklatu batean (ordenagailuarenean) pasahitz zuzena sartuz, atea irekiko da (P1.0 aktibatuko da) eta pantailan mezu hau agertuko da: “ATEA

## 11.15. C51 KONPILADOREA

C51 konpiladoreak C lengoiaian idatzitako kodea makina-kode bihurtzen du. Bi pausotan egiten da: C51 konpiladorearekin helburu-kodea lortzen da (.obj), helbideak esleituta ez dituena; eta gero L51 estekatzailarekin makina-kodea, emuladore edo simulagailu batean programaren funtzionamendua egiazatzeko.



Keil etxeko C51 ez da 51ra egokitutako konpiladore orokorra, baizik eta berariaz 51rako diseinatutako konpiladorea. Ondorioz, kode txikia eta azkarra sortzen du eta ANSI estandarrean oinarritzen da.

OH51 programak fitxategi absolutua INTEL formatuko fitxategi (.HEX) bihurtzen du. Formatu hori da, hain zuen, EPROM programatzeko erabiltzen dena.

### 11.15.1. C51-K JASATEN DITUEN DATU-MOTAK

11.14. taulan konpiladoreak onartzen dituen datu-motak ikus ditzakegu. Array-ak egitura eta loturekin lan egin dezake. Aldagaietara erakusleen bidez joan daiteke. Sbit eta sfr mota bereziekin funtzio berezikorregistroetara (SFR) joan daiteke.

Datu-moten arteko bihurketa automatikoki egiten da emaitzak beste mota bat behar duenean.

Datu-mota	Luzera	Balioen tartea
bit	Bit bat	0 edo 1
signed char	Byte bat	-128 → +127
unsigned char	Byte bat	0 → 255
signed int	2 byte	-32.768 → +32.767
unsigned int	2 byte	0 → 65.535
signed long	4 byte	-2.147.483.648 → +2.147.483.647
unsigned long	4 byte	0 → 4.294.967.295
float	4 byte	±1,176E-38 → ±3,40E+38
erakuslea	1-3 byte	Objektuen arabera
sbit	Bit bat	0 edo 1
sfr	Byte bat	0 → 255
sfr16	2 byte	0 → 65.535

11.14. taula

### 11.15.2. MEMORIA-MOTAK

Konpiladore honek 51 familiaren arkitektura du. Aldagai bakoitza memoria-mota batean koka daiteke. Barruko memoriara (IDATA) askoz azkarrago joaten da kanpoko memoriara (XDATA) baino; beraz, askotan erabiltzen diren aldagaiak barruko memorian kokatzea gomendatzen da.

Memoria-mota	Azalpena
Data	Zuzenean helbidera daitekeen barruko RAM memoria (128 byte); hor lortzen da azkartasun haundiengatik.
Bdata	Bitez bit helbidera daitekeen barruko RAM memoria (16 byte)
Idata	Zeharka helbidera daitekeen barruko RAM memoria. Hau da, barruko RAM osoa (255 byte)
Pdata	Orrikatutako kanpoko RAM memoria (256 byte). MOVX @Ri instrukzioarekin helbideratzen da.
Xdata	Kanpoko RAM memoria osoa (64K byte). MOVX @DPTR instrukzioarekin helbideratzen da.
Code	Kode-memoria (64K byte). MOVC @A+DPTR instrukzioarekin helbideratzen da.

11.15. taula

Ondoren azaltzen diren aldagaien deklarazioak baliagarriak dira:

```
char data aldagai1;
char code testu[] = "SARTU DATUA:";
```

```

unsigned long xdata zenbaki[100];
float idata x;
unsigned int pdata balioa;
unsigned char xdata dim[10][5];
sfr P0 = 0x80;
sbit RI = 0x98;
char bdata flag;

```

Aldagaiak deklaratzean memoria-mota jartzen ez badugu, lehenetsitako memoria-motan kokatuko dira. Lehenetsitakoa SMALL, COMPACT edo LARGE aukeren bidez zehazten da.

### 11.15.3. MEMORIA-EREDUAK

Eredu hauek aldagaientzat lehenetsitako memoria zehazten dute.

Memoria-mota	Azalpena
SMALL	Aldagaiak zuzenean helbidera daitekeen barruko RAM memorian (data) kokatzen dira (gehienez 120 byte).
COMPACT	Aldagaiak orrikatutako kanpoko RAM memorian (pdata) kokatzen dira (256 byte).
LARGE	Aldagaiak kanpoko RAM memorian (xdata) kokatzen dira (64k byte).

11.16. taula

### 11.15.4. ERAKUSLEAK

Konpiladore honetan erakusle orokorrak edo memoria zehatz batekoak izan daitezke. Erakusle orokor batek 3 byte behar ditu; byte bat memoria-mota zehazteko eta beste biak helbidea gordetzeko. Erakusle orokorrek edozein memoriatako aldagaiak kontrola ditzakete. 11.17. taulan adibide batzuk daude.

Deklarazioa	Zabalera	Memoria-mota
char *p;	3 byte	Edozein memoria-mota.
char data *dp;	byte 1	<i>data</i> memoria-eremua soilik.
int idata *ip;	byte 1	<i>idata</i> memoria-eremua soilik.
long pdata *lp;	byte 1	<i>pdata</i> memoria-eremua soilik.
char xdata *p1;	2 byte	<i>xdata</i> memoria-eremua soilik.
int code *p2;	2 byte	<i>code</i> memoria-eremua soilik.

11.17. taula

### 11.15.5. ETENDURAK

C51 konpiladorearekin etendurak funtzi baten bidez programatzen dira. Funtzio horren barruan etendurak egin behar duen lana adierazten da eta kito. Funtzio horien eredu hau da:

**Izena() interrupt n [using n]**

Funtzio horiek ez dute ezer itzultzen, eta ez zaie ezer pasatu behar. **Interrupt** hitzak adierazten du funtzi hori etendura batena dela, eta **n** zenbakiarekin zein etendura den zehazten dugu. Hauek dira erabili behar diren **n** zenbakiak:

ZENBAKIA	ETENDURA
0	INT0 : Kanpoko 0 etendura.
1	TIMER0 : <i>Timer 0-aren</i> etendura.
2	INT1: Kanpoko 1 etendura.
3	TIMER1: <i>Timer 1-en</i> etendura.
4	Serieko ataka.

Beste parametroa **using** aukerakoa da. Etendurak zein erregistro-bankurekin egingo duen lana zehazten du.

Etendura egikaritzean, mikroak automatikoki eragiketa hauek egingo ditu:

- ◆ ACC, B, DPH, DPL eta PSW pilan gordetzen dira (beharrezkoa bada).
- ◆ Erregistro-bankua aldatzen ez badugu (*using* jartzen ez badugu), etendurak erabiliko dituen erregistroak (Rn) pilan gordeko ditu.
- ◆ Funtzia egikaritu ondoren eta programa nagusira itzuli baino lehen, erregistroen balio guztiak pilatik berreskuratuko dira.

**Adibidea:** Timer0-a programatuko dugu 2 moduan eta FF (= 255) kontatzeko. Kontu bakoitzean  $\frac{12}{12 \cdot 10^6} \cdot 255 = 255 \cdot 10^{-6}$  segundoko etendura bat sortuko du. Beraz,  $\frac{1}{255 \cdot 10^{-6}} = 3921$  etendura sortu beharko dira segundo bat igarotzeko. Etenduraren barruan segundoak kontatzeko programa idatziko dugu.

```
unsigned int interruptcnt=0;
unsigned char segundoak=0;

denbora() interrupt 1      /* Timer0-aren etendura eta erregistro-
 */
{
    if(++interruptcnt == 3921)
    {
        segundoak++;          /* segundoen kontagailua */
        interruptcnt = 0;      /* etenduren kontagailua hasieratu */
    }
} /*denbora-amaiera */
```

### 11.15.6. KONPILATZEKO ERA

C51 konpiladoreari deitzeko bere izena, fitxategiaren izena eta parametroak jartzen dira:

#### **C51 fitxategiaren\_izena.c51 [parametroak]**

Konpilatu ondoren estekatu egin behar da. Horretarako, L51 erabiliko da era honetan:

#### **L51 fitxategiaren\_izena.obj [parametroak]**

### 11.15.7. C51-REN LIBURUTEGI-FUNTZIOAK

#### **CTYPE.H (karaktereekin lan egiteko funtzioak)**

- ※ **Bit isalpha(char);** Pasatutako parametroa letra (A-Z edo a-z) bada, 1 itzultzen du; bestela, 0.
- ※ **Bit isalnum(char);** Pasatutako parametroa letra edo zenbakia (A-Z, a-z edo 0-9) bada, 1 itzultzen du; bestela, 0.
- ※ **Bit iscntrl(char);** Pasatutako parametroa kontroleko karakterea (0x00-0x1F edo 0x7F) bada, 1 itzultzen du; bestela, 0.
- ※ **Bit isdigit(char);** Pasatutako parametroa zenbakia (0-9) bada, 1 itzultzen du; bestela, 0.
- ※ **Bit isgraph(char);** Pasatutako parametroa inprimagarria (0x21-0x7E) bada, 1 itzultzen du; bestela, 0.
- ※ **Bit isprint(char);** isgraph bezalakoa baina honek tartea (0x20) ere onartzen du.
- ※ **Bit ispunct(char);** Pasatutako parametroa puntuazio-karakterea edo tartea bada, 1 itzultzen du; bestela, 0.

- \* **Bit islower(char);** Pasatutako parametroa letra xehea (a-z) bada, 1 itzultzen du; bestela, 0.
- \* **Bit isupper(char);** Pasatutako parametroa letra larria (A-Z) bada, 1 itzultzen du; bestela, 0.
- \* **Bit isspace(char);** Pasatutako parametroa zuria, tabuladorea, return, lerro berria edo tabuladore bertikala bada, 1 itzultzen du; bestela, 0.
- \* **Bit isxdigit(char);** Pasatutako parametroa zenbaki hamaseitarra (0-9, A-F edo a-f) bada, 1 itzultzen du; bestela, 0.
- \* **Char tolower(char);** Pasatutako parametroa letra larria (A-Z) bada, xehe bihurtzen zen; bestela ez du ezer egiten.
- \* **Char toupper(char);** Pasatutako parametroa letra xehea (a-z) bada, letra larri bihurtzen du; bestela ez du ezer egiten.

### **STDIO.H (Sarrera/irteera funtzioak)**

- \* **Char getchar();** Karaktere bat irakurtzen du 8051ren serieko lineatik. Karaktere bat jaso arte itxaron egingo du.
- \* **Char \* gets(char \*s, int n);** Kate bat irakurtzen du eta s array-an uzten du. N-k markatzen du katearen karaktere-kopurua, dena den, katea bukatutzat ematen da *return* batekin.
- \* **Putchar(char);** char bidaltzen du 8051ren serieko lineatik.
- \* **Int printf(char \*...);** Kateak eta zenbakiak formateatzen ditu eta serieko lineatik bidaltzen ditu funtzi estandarrak egiten duen antzera. Formatuak eremu hauek ditu:

**% [flag-ak][zabalera].[doitasuna]mota**

<b>Flag</b>	<b>Esanahia</b>
-	Irteera ezkerrean justifikatuta.
+	Zenbakien zeinua bistaratuko da.
Zuria	Balio positiboak zuri batekin hasiko dira.
#	<b>x</b> edo <b>X</b> motekin erabilita, <b>0x</b> edo <b>0X</b> ipintzen da zenbakiaren hasieran. <b>g</b> , <b>f</b> , edo <b>e</b> motekin hamartarren puntuoa bistaratuko da.
B,b	d, i, o, u, x eta X moten aurrikzia da. Horrela, parametroa [unsigned] char bezala hartuko da. Adibideak: %bu, %bd, %bX
L,I	d, i, o, u, x eta X moten aurrikzia da. Horrela, parametroa [unsigned] long bezala hartuko da. Adibideak: %lu, %ld, %lX

**Zabalera.** Zenbat karaktere bistaratuko diren zehazten du. Karaktere gutxiago badago, ezkerrean zuriak jartzen dira; baina eremu hori **0** batez hasten bada, ezkerrean zeroak ipiniko dira.

**Doitasuna.** Zenbat digitu hamartar bistaratuko diren zehazten du.

<b>Mota</b>	<b>mota</b>	<b>Irteera-formatua</b>
d	int	zenbaki hamartarra zeinuarekin.
u	int	zenbaki hamartarra zeinurik gabe.
o	int	zenbaki zortzitarra zeinurik gabe.
X, x	int	zenbaki hamaseitarra zeinurik gabe.
f	float	zenbakia hamartarrekin. ddd.ddd
E, e	float	zenbakia hamartarrekin era zientifikoan. dddEdd
c	char	karaktere bat.
s	pointer	kate batera apuntatzen duen erakuslea.
p	pointer	erakuslea, memoria-mota eta offset adieraziz. <b>M:aaaa</b> M: C(ode), D(ata), I(data), X(data), P(data) aaaa: offset.

- \* **int puts(char \*s);** Kate bat bidaltzen du serieko lineatik.
- \* **Int scanf(char ...);** Serieko lineatik datuak jasotzen ditu. Erakuslea pasatu behar zaio parametro bezala, eta erakusleak apuntatzen duen helbidean utziko ditu datuak. Formatuak eremu hauek ditu:

**%[flag][zabalera] mota**

<b>Flag</b>	<b>Esanahia</b>
b, h	d, i, o, u, x eta X moten aurritzka da. Horrela, parametroa [unsigned] char bezala hartuko da. Adibideak: %bu, %bd, %bX
L,I	d, i, o, u, x eta X moten aurritzka da. Horrela, parametroa [unsigned] long bezala hartuko da. Adibideak: %lu, %ld, %lX

**Zabalera.** Gehienez zenbat karaktere onartuko diren zehazten du.

<b>Mota</b>	<b>mota</b>	<b>Irteera formatua</b>
d	ptr int	zenbaki hamartarra zeinuarekin.
u	ptr int	zenbaki hamartarra zeinurik gabe.
o	ptr int	zenbaki zortzitarra zeinurik gabe.
x	ptr int	zenbaki hamaseitarra zeinurik gabe.
F, e, g	ptr float	zenbakia hamartarrekin.
c	ptr char	karaktere bat.
s	ptr pointer	kate batera apuntatzen duen erakuslea.

## STRING.H (kateekin funtzioak)

- \* **Char memcmp(\*s1, \*s2, int len);** Bi kate horien lehendabiziko **len** karaktere konparatzen ditu. Berdinak badira 0 itzultzen du; s1 handiagoa (ASCII balioa) bada, zenbaki positiboa; eta txikiagoa bada, zenbaki negatiboa.
- \* **memcpy(\*s1, \*s2, int len);** S2-katearen lehendabiziko **len** karaktere kopiatzen ditu s1ean. Bi memoria-eremu horiek ezin dira teilakatu.
- \* **memmove(\*s1, \*s2, int len);** memcpy bezalakoa baina bi memoria-eremuak teilaka daitezke.
- \* **memset(\*s, char val, int len);** s katearen lehendabiziko **len** byteak **val** balioarekin jartzen ditu.
- \* **Char \* strcat(char \*s1, char \*s2);** s2 katea s1 katearen atzetik kopiatzen du. S1 katearen eremuak behar bezain handia izan behar du. Itzulitako erakusleak s1 katearen lehendabiziko byteria apuntatzen du.
- \* **Char \* strncat(char \*s1, char \*s2, int n);** s2 katearen lehendabiziko **n** karaktere s1 katearen atzetik kopiatzen ditu. Itzulitako erakusleak s1 katearen lehendabiziko byteria apuntatzen du.
- \* **Char strcomp(char \*s1, char \*s2);** Bi kateak konparatzen ditu. Berdinak badira, 0 itzultzen du; s1 handiagoa (ASCII balioa) bada, zenbaki positiboa; eta txikiagoa bada, zenbaki negatiboa.
- \* **Char strncomp(char \*s1, char \*s2, int n);** Bi kateen lehendabiziko **n** karaktere konparatzen ditu. Berdinak badira, 0 itzultzen du; s1 handiagoa (ASCII balioa) bada, zenbaki positiboa; eta txikiagoa bada, zenbaki negatiboa.
- \* **Char \* strcpy(char \*s1, char \*s2);** s2 katea s1 katearen gainean kopiatzen du, baita NULL karakterea ere. Itzulitako erakusleak s1 katearen lehendabiziko byteria apuntatzen du.
- \* **Char \* strncpy(char \*s1, char \*s2, int n);** strcpy bezalakoa baina kasu honetan lehendabiziko **n** karaktereekin bakarrik.
- \* **int strlen(char \*s);** s kateak duen karaktere-kopurua itzultzen du, NULL karakterea kontuan izan gabe.

### **STDLIB.H (funtzio estandarrak)**

- \* **double atof(char \*s);** atof funtzioak s katea double bihurtzen du, eta hori da, hain zuzen, itzultzen duena. S kateak izan behar du float baten formatua. Tarteak ez ditu kontuan hartzen, eta zerbait arraroa aurkitzen badu, bukatu egiten du. C51 konpiladorearentzat float eta double mota berdinak dira.
- \* **long atol(char \*s);** atol funtzioak s katea long bihurtzen du, eta hori da, hain zuzen, itzultzen duena. S kateak izan behar du long baten formatua. Tarteak ez ditu kontuan hartzen, eta zerbait arraroa aurkitzen badu, bukatu egiten du.
- \* **int atoi(char \*s);** atoi funtzioak s katea int bihurtzen du, eta hori da, hain zuzen, itzultzen duena. S kateak izan behar du int baten formatua. Tarteak ez ditu kontuan hartzen, eta zerbait arraroa aurkitzen badu, bukatu egiten du.

### **MATH.H (funtzio matematikoak)**

- \* **int abs(int val);** val osoaren balio positiboa itzultzen du.
- \* **char cabs(char val); float fabs(float val); long labs(long val);** abs bezalakoa baina char, float eta long motekin, hurrenez hurren.
- \* **float exp(float x);**  $e^x$  itzultzen du.
- \* **float log(float x);**  $\ln x$  itzultzen du.
- \* **float log10(float x);**  $\log x$  itzultzen du.
- \* **float sqrt(float x);**  $\sqrt{x}$  itzultzen du.
- \* **int rand();** zenbaki sasialeatorio bat (0 - 32767) itzultzen du.
- \* **float cos(float x); float sin(float x); float tan(float x);** x balioaren sinua, kosinua eta tangentea itzultzen ditu, hurrenez hurren.
- \* **float ceil(float x);** x zenbakia handiagora biribiltzen du.  
Adib: 35.765 → 36.000
- \* **float floor(float x);** x zenbakia txikiagora biribiltzen du.  
Adib: 35.765 → 35.000
- \* **float pow(float x, float y);**  $x^y$  itzultzen du. X==0 eta y<=0; eta x<0 eta y osoa balio ez-bikote hauekin ez du funtzionatzen.

### **ABSACC.H (memoria-helbide zehatzak maneiatzeko funtzioak)**

- \* **CBYTE, DBYTE, PBYTE** eta **XBYTE** makroekin 8051en memoria-helbideak kontrola daitezke: CODE, DATA, PDATA eta XDATA, hurrenez hurren. Helbide bakar batekin (unsigned char) lan egiten dute.  
Memoria irakurtzeko:  
a=XBYTE[0x1000];  
b=DBYTE[0x0035];  
Memorian idazteko:  
XBYTE[0x1000]= 0x3A;  
DBYTE[0x0035]= 0xFF;
- \* **CWORD, DWORD, PWORD** eta **XWORD** besteak bezalakoak dira, baina, kasu honetan, int-ekin lan egiten da, hau da, bi memoria-helbiderekin.

## Ariketa ebatziak C lengoain

Praktika hauek DISEN-51 entrenatzailearekin eginak daude. Beraz, horren memoria-mapa eta beste zenbait berezitasun kontuan hartuz egin dira. Beste ekipo bat erabiliz gero, agian aldaketa txiki batzuk egin beharko dira.

### **1. PRAKTIKA** (P1 atakatik datu bat atera)

DELAY funtzioa erabiliz, P1 atakaren P1.3 pinean dardara sortu. Kontuan hartu behar da, DELAY funtzioak pasatzen zaizkion milisegundoetako atzerapena sortzen duela.

Led-ak P1 atakara konektatu behar dira praktika aztertzeko. Programan aldaketa txiki bat eginez, dardara-erritmo ezberdinak lor daitezke.

```
#include <stdio.h>
#include <reg552.h>
void delay(int denbora);           /* delay funtzioaren protipoa */

void main(void)
{
    while(1)                      /* begizta amaiezina */
    {
        P1_0 ^= 1;                /* P1 atakako 0 pina osatu funtzio bateratzailearekin */
        delay(100);               /* 0,1 segundo itxaron */
    }
}/* main bukaera*/

*****
Funtzioa: void delay(int denbora)
pasatzen zaizkion milisegundoetako atzerapena sortzen du.
*****
```

```
void delay(int denbora)
{
    while(denbora)
    {
        unsigned char i;
        for(i=226; i>0; i--) /* ~0,5 ms galtzen ditu */
        ;
        for(i=226; i>0; i--) /* ~0,5 ms galtzen ditu */
        ;
        denbora--;
    }
}/* delay BUKAERA*/
```

## **2. PRAKTIKA** (Datuak ataka batetik atera eta mezua PRINFTrekin)

Programa honetan, P1 atakatik kode bitarrean eta atzerapen txiki batekin 0 eta FF bitarteko datuak atera behar dira (aldi berean, unitate bat gehitzen zaio).

Gainera, ateratako datu bakoitzarekin, serieko lineatik mezu bat bidali behar duzue pantailara, "X datuak atera ditut" dioena, une bakoitzean ateratako datua X izanik hamaseitarrean.

```
#include <stdio.h>
#include <reg552.h>
void delay(int denbora);           /* delay funtziaren protipoa */

void main(void)
{
    unsigned char i;
    for(i=0; i<255; i++)
    {
        P1 = i;
        delay(100);
        printf("\n %bx datuak atera ditut", i);
    }
}/* main */

*****
Funtzioa: void delay(int denbora)
pasatzen zaizkion milisegundoetako atzerapena sortzen du.
*****
```

```
void delay(int denbora)
{
    while(denbora)
    {
        unsigned char i;
        for(i=226; i>0; i--) /* ~0,5 ms galtzen ditu */
        ;
        for(i=226; i>0; i--) /* ~0,5 ms galtzen ditu */
        ;
        denbora--;
    }
}/* delay BUKAERA */
```

## **3. PRAKTIKA** (P1 ataka sarrera)

P1.7 pinean dagoen datua pantailatik ateratzeko programa bat egin ezazue. Datua etengabe eguneratu.

```
#include <stdio.h>
```

```
#include <reg552.h>

void main(void)
{
    int i;                  /* printf-rekin ezin dira bitak inprimatu, horregatik int definitu dugu */
    while(1)
    {
        i=P1_7;
        printf("\r P1_7 bitak duen balioa da: %d",i);
    }
}/* main */
```

#### **4. PRAKTIKA** (XBYTE[] funtziaren bidez, kanpoko RAMa helbideratu)

XBYTE[] izeneko funtzioa erabiliz, kanpoko memoriaren 1000H eta 2000H bitarteko posizioetan, 7FH datua kokatu behar da. XBYTE[] funtziaren prototipoa ABSACC.H fitxategian dago.

Programa egikaritu baino lehen eta egikaritu ondoren, Display (D) komandoarekin memoriaren helbide horiek egiaztu.

```
#include <stdio.h>
#include <reg552.h>
#include <absacc.h>

void main(void)
{
    int i;
    unsigned char j=35;
    for(i=0x1000;i<0x2000;i++)
        XBYTE[i]=j;
}/* main */
```

#### **5. PRAKTIKA** (erakusle bat erabiliz, barruko RAMa helbideratu)

Barruko RAM memoriaren 10H eta FFH bitarteko posizioetan 7FH datua idatzi behar da. Helbideak erakusle baten bidez helbideratu behar dira.

Gainera, idatzitako datu bakoitzean mezu bat igorri behar da, “XX barruko memoriaren posizioan 7F datua idatzi dut” dioena.

```
#include <stdio.h>
#include <reg552.h>
#define dato 0x33

void main(void)
{
    char idata *xdata punt;      /* Definitzen dugu idata memorira apuntatuko duela eta erakuslea
xdata memorian gordeko dela */

```

```
int xdata i;          /* Aldagai hau non gorde behar den zehazten badugu, segurtasunagatik
                      barruko memorian gordeko da. Horrek gatazkak sortuko ditu, barruko
                      memoriaren balioak aldatuko baitugu. Beraz, segurtasunagatik kanpoko
                      memorian gordetzea gomendatzen da. */
punt=0x10;
for(i=0;i<200;i++,punt++)
{
    *punt=datu;
    printf("\n Memoriaren %x helbidean%x idatzi dut",punt, datu);
}
/* main */
```

## **6. PRAKTIKA (TIMER0. Soft testeoa)**

Atzerapena sortzeko, *TIMER0* erabiliz, P1 atakan konektatutako led bat dardara eragin iezaziozue.

TF0 *flag*-aren testeoa soft-en bidez egingo da.

```
#include <stdio.h>
#include <reg552.h>
void denbora(void);

void main(void)
{
    TMOD = 0x01;      // Timer0 16 bitekoa
    while(1)
    {
        P1_0 ^= 1;    // P1_0 osatu
        denbora();
    }
}
/* main */

void denbora(void)
{
    TL0 = TH0 = 0;    // Kontatzeko balio handiena 65.535 µs.
    TR0=1;           // Timer0 abian jarri
    while(TF0==0)     // Itxaron kontua bukatu arte. Soft testeoa
    ;
    TF0=0;           // Flag hau ezabatu egin behar da beste kontu bat detektatu ahal izateko.
}
```

## **7. PRAKTIKA (TIMER0. Etendurak)**

Aurreko praktikan aipatutakoa egin behar dugu, baina kasu honetan, *timer*-ak mikrokontroladorea eten behar du.

Gainera, “XX etenaldi sortu dira” dioen mezuak agertu behar du.

```
#include <stdio.h>
```

```
#include <reg552.h>

int a=0;      /* aldagai globala */
timer0() interrupt 1      /* etenduraren funtzioa */
{
    TH0 = TL0 = 0x00;      /* Balio maximoa: 65.535 µs
    P1_0 ^= 1;      /* XOR funtzioarekin bitaren osagarria */
    a++;
    /* etenduraren bukaera */

void main(void)
{
    TMOD &= 0xf0;  /* Timer1 ez aldatzen, serieko linearekin erabili behar baitugu */
    TMOD |= 1;      /* Timer0 16 bitekoa */
    ET0 = 1;        /* Timer0-aren etendura gaitu */
    EA = 1;         /* Etendura guztiak gaitzeko */
    TH0=TL0=0x00;
    TR0 = 1;        /* Timer0 abian jarri */
    while(1)
    {
        printf("\r%3d etendura daramatzagu ",a);
    }
}/* main */
```

## **8. PRAKTIKA** (Kit gisako argi-jokoa)

Hiru array-tan, hiru argi-sekuentzia sar itzazue. Atera ezazue sekuentzia bakoitza 5 aldiz P1 atakatik led-etara.

Funtzio bat egin beharko dugu, datuak array-etatik led-etara atera ditzan.

```
#include <reg552.h>
#include "funtzioak.h"      /* delay funtzioa fitxategi honetan idatzi dugu */

void atakatik_atera(char *);

void main(void)
{
    unsigned char argiak1[8]={0x81,0xc3,0xe7,0xff,0xff,0xe7,0xc3,0x81};
    unsigned char argiak2[8]={0x80,0xc0,0xe0,0xf0,0xf8,0xfc,0xfe,0xff};
    unsigned char argiak3[8]={0x01,0x03,0x07,0x0f,0x1f,0x3f,0x7f,0xff};
    char *parray;
    while(1)
    {
        parray=argiak1;
        atakatik_atera(parray);
        parray=argiak2;
        atakatik_atera(parray);
        parray=argiak3;
        atakatik_atera(parray);
    }
}
```

```
/* main */  
*****  
***  
Funtzio honek zortzi laukiko array batera apuntatzen duen erakusle bat jasotzen du. Array-aren balioak  
atzerapen batekin ateratzen ditu, array-a bost aldiz errepikatuz.  
*****  
*/  
void atakatik_atera(char *punt)  
{  
    char i,j,*puntaux;  
    puntaux=punt;  
    for(i=0;i<5;i++)  
    {  
        punt=puntaux; // punt hasieratu behar da array-a berriro hasieratik hasteko.  
        for(j=0;j<8;j++)  
        {  
            P1=*punt;  
            punt++;  
            delay(100);  
        }  
    }  
}
```

## ***Proposatutako ariketak***

### **9. PRAKTIKA** (kanpoko RAMa helbideratu erakuslea erabiliz)

Hirugarren praktikan egindakoa egin ezazue, baina ariketa honetan, XBYTE funtzioa erabili ordez, erakusle batekin helbideratu behar da.

Gainera, idatzitako datu bakoitzean mezu bat igorri behar da, “XXXX posizioan

### **10. PRAKTIKA** (DBYTE[] funtzioarekin barruko RAMa helbideratu)

DBYTE funtzioa erabiliz, memoriaren 10H eta FFH bitarteko posizioetan 7FH datua idatzi behar da. DBYTE funtzioaren prototipoa ABSACC.H fitxategian dago.

Programa egikaritu baino lehen eta egikaritu ondoren, Display (D) komandoarekin memoriaren helbide horiek egiaztatu.

### **11. PRAKTIKA** (Kontagailua *Timer0*-arekin)

*Timer0* programa ezazue, T0 pinean aplikatutako kanpoko pultsuak kontatu ahal izateko.

Pantailan mezu batek agertu behar du, une bakoitzean sartutako pultsuak adierazten dituena.

Zenbaketa bukatzen denean, *timer*-ak etendura sortu behar du, eta mezu bat aterako da etendura sortu dela adieraziz. Kontagailuak hamar pultsu besterik ez ditu kontatu behar.

### **12. PRAKTIKA** (INT1 kanpoko etendura)

INT0 kanpoko etenaldia programatu behar duzue, beheranzko saihetsan aktibatzeko.

Programa nagusia “While bukaezina” izango da. Horrek mezu bat agerrarazi behar du, **“Hemen ez da ezer gertatzen, kanpoko etendura baten zain nago”** dioena.

Etendura sortzen denean, beste mezu bat agertuko da, **“Jadanik, XX aldiz eten duzue”** dioena.

**13. PRAKTIKA** (kanpokotik kontrolatutako kit argiak):

Zortzigarren praktikan aipatutako sekuentziak erabili behar dituzue. INT0 kanpoko etendura programatu behar dugu, beheranzko saihetsekin aktiba dadin.

Programa “While bukaezina” izango da, zeinak argiaren sekuentzia bat aterako bai. Etendura agertzen den bakoitzean hurrengo sekuentziara aldatuko da, eta horrela bata bestearen segidan.

**15. PRAKTIKA** (1. erlojua):

DELAY funtzioa erabiliz, erlojua programatu behar da. Orduak, minutuak eta segundoak pantailan aterako dira.

**16. PRAKTIKA** (2. erlojua):

Aurreko praktikan, uK-a erlojuaren funtzionamenduarekin guztiz lanpeturik dago. Hori saihesteko eta, aldi berean, uK-ak beste gauza bat egin dezan, *timer0* eta horrek sortzen duen etendura erabiliko dugu denbora kontatzeko.

**17. PRAKTIKA** (teklatu hamaseitarra):

Matrize-teklatu bat irakurtzeko programa bat egin ezazue. Sakaturiko teklaren balioa pantailan aterako da.

**18. PRAKTIKA** (LCD):

LCD batetik mezu batzuk atera itzazue.

# 12. KAPITULUA

## PIC mikrokontroladoreak

### 12.1. MIKROKONTROLADOREEN ARKITEKTURA OROKORRAK

Mikrokontroladoreen instrukzioen arabera, hiru arkitektura ezberdin bereiz daitezke: CISC, RISC eta SISC.

#### Cisc

Mikrokontroladore-mota hauek instrukzio-joko konplexua daukate, laurogeitik gora, eta batzuk oso konplexuak direnez, makina-ziklo asko behar dituzte egikaritzeko. Horien abantaila nagusia instrukzioen ahalmena da.

#### risc

Filosofia honekin lan egiten dutenek instrukzio-joko txikia daukate. Gainera, instrukzioak oso simpleak direnez, ziklo bakar batean egikaritzen dira. Horrek abantaila ematen digu hardwarea eta softwarea optimizatzeko.

Gaur egun, fabrikatzaileek filosofia honetara jotzen dute, batez ere, *Microchip*-en arrakasta ikusita.

#### sisc

Funtzio zehatza izango duten mikrokontroladoreentzat, oso instrukzio gutxi dituzten eta, gainera, funtzio hori egiteko egokitzen direnak.

### 12.2. MIKROKONTROLADOREEN BARNEKO EGITURA

Horien atal nagusiak honako hauek dira :

- 1) Prozesadorea
- 2) Programa edukitzeko memoria hila
- 3) Datuak gordetzeko RAM memoria

- 4) Periferikoen kontrolerako S/I lerroak
  - Paraleloko komunikazioa
  - Serieko komunikazioa
  - Zenbait komunikaziorako ate ( $I^2C$  busa, *USB*...)
- 5) Baliabide laguntzaileak
  - Erlojuaren zirkuitua
  - Temporizadoreak
  - *Watchdog* edo “zakur zaindaria”
  - AD/DA bihurgailuak
  - Konparatzaile analogikoak
  - Kontsumo txikiko edo atsedeneko egoera

## 12.3. PIC-ETAN ROM MEMORIA-MOTA EZBERDINAK

---

Programa gordetzeko hainbat memoria-mota dituzten mikrokontroladoreak aurki daitezke. Hauek dira merkatuan daudenak:

### ROM MASKARAREKIN

ROM memoria hau txipa fabrikatzean grabatzen da. Maskararen diseinua oso garestia denez, milaka ale behar direnean bakarrik gomendatzen da.

### OTP

Memoria-mota hau (*One Time Programmable*) erabiltzaileak grabatzale baten bidez grabatzen du, baina **behin bakarrik**. Gero ezin da ezabatu, ezta birprogramatu ere.

### EPROM

Erabiltzaileak grabatu eta ezaba dezake hainbat aldiz. *EPROM* (*Erasable Programmable Read Only Memory*) memoriak grabatzale simple batekin grabatzen dira. Leihoa txiki bat daukate goialdean eta bertatik, izpi ultramoren bidez, ezabatzen dira. OTP memoriak baino garestiagoak dira.

### EEPROM

EEPROM (*Electrical Erasable Programmable Read Only Memory*) memoria hauek elektrikoki programatu eta ezaba daitezke. Grabatzalean bertan grabatu eta ezabatzen dira.

## FLASH

Hauek ere grabatu eta ezaba daitezke, baina zirkuituan bertan, eta grabatutako datuak ez ditu galtzen, hau da, RAM eta ROM batera bezalakoa da. Txikia, azkarra eta kontsumo txikikoa da.

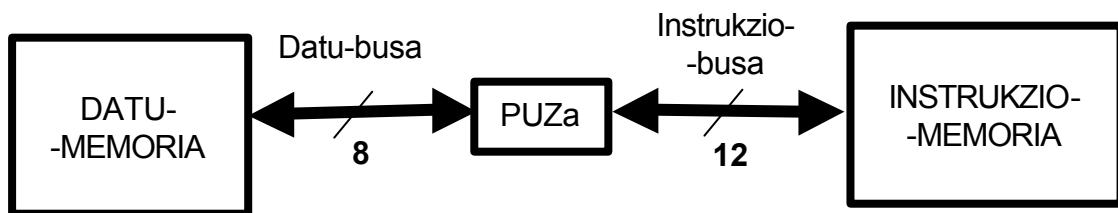
## **12.4. PIC MIKROKONTROLADOREEN EZAUGARRI NAGUSIAK**

Ondoren aipatuko ditugu txip hauek dituzten ezaugarriak:

### HARVARD ARKITEKTURA

HARVARD arkitektura erabiltzen du bi bus independenterekin. Egitura honek ahalbidetzen du aldi berean eta independenteki datuetara eta instrukzioetara joatea.

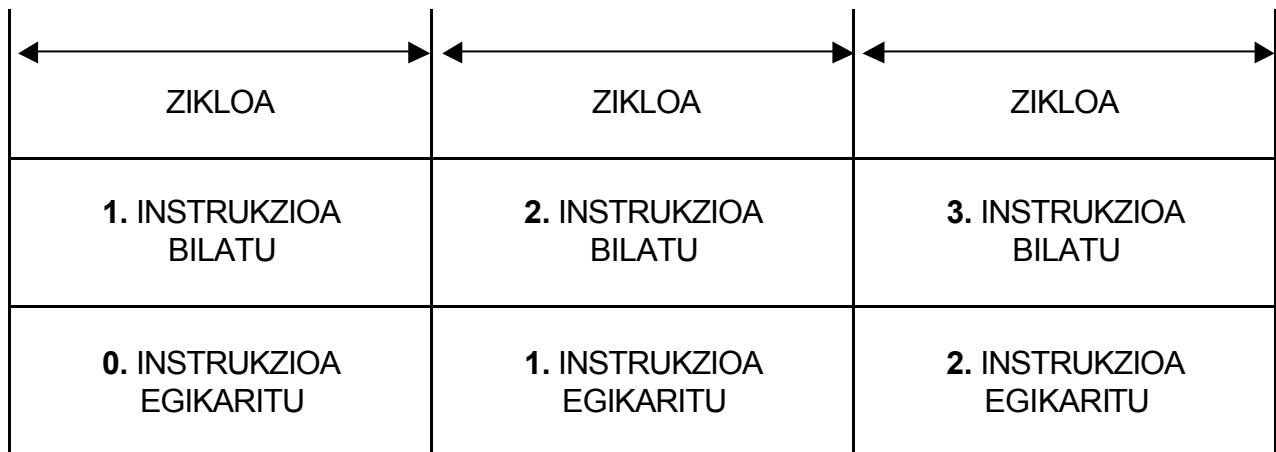
12.1. irudian egitura honen eskema ikus daiteke.



12.1. irudia. Hardvard arkitektura.

### INSTRUKZIOAK EGIKARITZEAN SEGMENTAZIO-TEKNIKA

Segmentazio-teknikari esker, prozesadorea instrukzio bat egikaritzen ari den bitartean hurrengo instrukzioa bilatzen ari da. Horrela, instrukzioak ziklo bakar batean egikaritzen dira.



### **INSTRUUKZIO GUZTIAK LUZERA BERDINA**

Agindu guztiak luzera berdina daukate: gama baxukoek 12 bit, gama ertainekoek 14 bit eta gama altukoek 16 bit. Hori abantaila da instrukzio-memoria optimizatzeko eta mikztatzaileak eta konpiladoreak diseinatzeko.

### **RISC PROZESADOREA**

Instrukzio-joko txikia eta simplea daukate: gama baxukoek 33 agindu, gama ertainekoek 35 eta gama altukoek 60.

### **INSTRUUKZIO GUZTIAK ORTOGONALAK DIRA**

Mikrokontroladorearen elementu guztiak izan ditzakete eragigai edo helburu-instrukzio guztiengandik.

### **ERREGISTRO-BANKUETAN OINARRITUTAKO EGITURA**

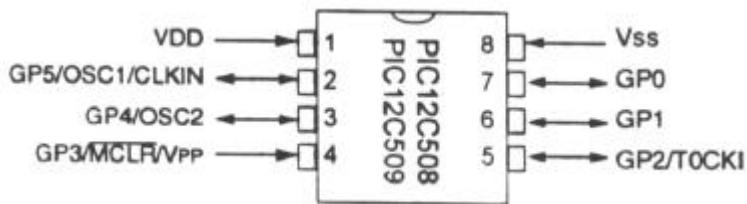
Sistemaren objektu guztiak (erregistroak, S/I atakak, temporizadoreak, memoria-posizioak, etab.) fisikoki erregistro bezala eginak daude.

## **12.5. MICROCHIP ETXEKO LAU GAMA EZBERDIN**

*Microchip* etxeak lau gamatan eskaintzen ditu bere mikrokontroladoreak. Gaur egun hirurogei mikrokontroladore ezberdin baino gehiago egiten ditu (urtero berriak azaltzen dira), beraz, aplikazio bakoitzerako egokiena hautatzeko aukera zabala da.

### 12.5.1. GAMA OSO BAXUA: PIC12CXXX (8 PINEKOA)

Gama hau berriena da. Ezaugarri nagusia bere tamaina txikia da, guztiak 8 pinekoak direlako. Elikadura 2,5-5 volt bitartekoa. Kontsumoa ere oso txikia da: 2mA baldin eta 5V eta 4MHz-eten lan egiten badute. Instrukzioen luzera 12 edo 14 bitekoa izan daiteke eta 33-35 instrukzio dauzkate.



12.1. taulan PIC hauen ezaugarriak azaltzen dira.

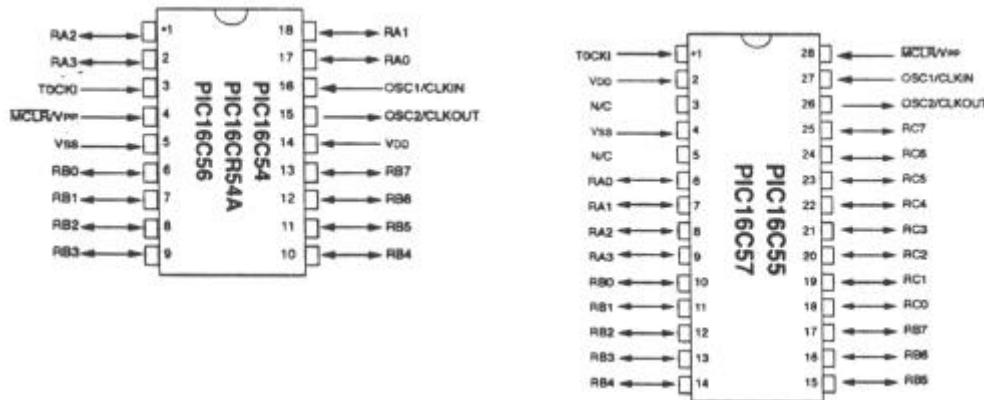
TXIPA	PROGRAMA-MEMORIA	DATU-MEMORIA	CLOCK	I/O	ADC 8 BIT	TENPORIZADOREAK	PINAK
<b>PIC12C508</b>	512x12	25x8	4MHz	6		TMR0 + WDT	8
<b>PIC12C509</b>	1024x12	41x8	4MHz	6		TMR0 + WDT	8
<b>PIC12C670</b>	512x14	80x8	4MHz	6		TMR0 + WDT	8
<b>PIC12C671</b>	1024x14	128x8	4MHz	6	2	TMR0 + WDT	8
<b>PIC12C672</b>	2048x14	128x8	4MHz	6	4	TMR0 + WDT	8
<b>PIC12F680</b>	512x12 FLASH	80x8 16x8 EEPROM	4MHz	6	4	TMR0 + WDT	8
<b>PIC12F681</b>	1024x14 FLASH	80x8 16x8 EEPROM	4MHz	6		TMR0 + WDT	8

12.1. taula

Letren esanahia: **C**: instrukzioen EEPROM memoria  
**F**: instrukzioen FLASH memoria  
**CR**: instrukzioen ROM memoria

### 12.5.2. GAMA BAXUA: PIC16C5X (18-28 PINEKOA)

PIC hauen baliabideak mugatuak dira, baina, aitzitik, kostu/ezaugarri erlazio oso ona daukate. 2,5-6,25 volt-eko elikadura eta kontsumo txikiagatik egokiak dira pilekin elikatzen diren prototipoentzat. 12 biteko 33 instrukzio ditu. Ez du etendurarik onartzen eta pilak bi maila baino ez ditu.



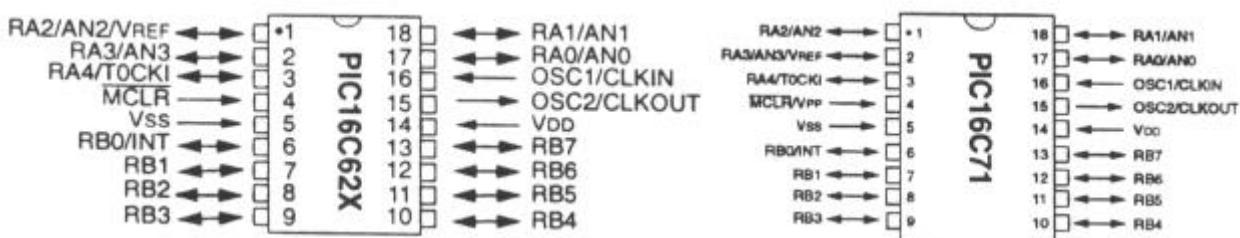
12.2. taulan PIC hauen ezaugarriak azaltzen dira.

TXIPA	PROGRAMA-MEMORIA (12 bit)	DATU-MEMORIA (bytetan)	CLOCK	I/O	TENPORIZADOREAK	PINAK
PIC16C52	384	25	4MHz	4	TMR0 + WDT	18
PIC16C54	512	25	20MHz	12	TMR0 + WDT	18
PIC16C54A	512	25	20MHz	12	TMR0 + WDT	18
PIC16CR54A	512	25	20MHz	12	TMR0 + WDT	18
PIC16C55	512	25	20MHz	20	TMR0 + WDT	28
PIC16C56	1K	25	20MHz	12	TMR0 + WDT	18
PIC16C57	2K	72	20MHz	20	TMR0 + WDT	28
PIC16CR57B	2K	72	20MHz	20	TMR0 + WDT	28
PIC16C58A	2K	73	20MHz	12	TMR0 + WDT	18

12.2. taula

### 12.5.3. GAMA ERTAINA: PIC16CXXX (18-40 PINEKOA)

Gama honetako 35 instrukzioak 14 bitekoak dira. Etendurak onartzen dituzte eta pilak 8 maila ditu azpiprogrametarako. Aipatzekoa da gama honetako PIC16F84 mikrokontroladore ospetsua.



Dauzkaten ezaugarrien arabera azpitalde hauek bereiz daitezke:

- \* Gama ertaineko estandarra (PIC16C55X)
- \* Konparadore analogikoak dituztenak (PIC16C62X/64X/66X)
- \* Kaptura-moduluak, PWM eta serieko ataka dituztenak (PIC16C6X)
- \* 8 biteko A/D dituztenak (PIC16C7X)
- \* Doitasunezko A/D dituztenak (PIC14000)
- \* FLASH eta EEPROM memoriadunak (PIC16X8X)
- \* LCD *driver*-a dutenak (PIC16C92X)

12.3. taulan PIC hauen ezaugarriak azaltzen dira.

TXIPA	PROGRAM. MEMORIA (12 bit)	DATU MEM.	CLOCK	I/O	A/D	ETEN- DURAK	TENPORIZA- DOREAK	PIN
<b>PIC16C61</b>	1k (eprom)	36	20MHz	13		3	TMR0	18
<b>PIC16C62</b>	2k (eprom)	128	20MHz	22		10	TMR0,TMR1,TMR2	28
<b>PIC16C63</b>	4k (eprom)	192	20MHz	22		10	TMR0,TMR1,TMR2	28
<b>PIC16C64</b>	2k (eprom)	128	20MHz	33		8	TMR0,TMR1,TMR2	40
<b>PIC16C65</b>	4k (eprom)	192	20MHz	33		11	TMR0,TMR1,TMR2	40
<b>PIC16C620</b>	512	80	20MHz	13		4	TMR0	18
<b>PIC16C621</b>	1k (eprom)	80	20MHz	13		4	TMR0	18
<b>PIC16C622</b>	2k (eprom)	128	20MHz	13		4	TMR0	18
<b>PIC16C71</b>	1k (eprom)	36	20MHz	13	4ch	4	TMR0	18
<b>PIC16C73</b>	4k (eprom)	192	20MHz	22	5ch	11	TMR0,TMR1,TMR2	28
<b>PIC16C74</b>	4k (eprom)	192	20MHz	33	8ch	12	TMR0,TMR1,TMR2	40
<b>PIC16C84</b>	1k (eprom)	36+64	20MHz	13		4	TMR0	18
<b>PIC16F84</b>	1k (flash)	36+64	20MHz	13		4	TMR0	18

12.3. taula

#### 12.5.4. GAMA ALTUA: PIC17CXXX (40-68 PINEKOA)

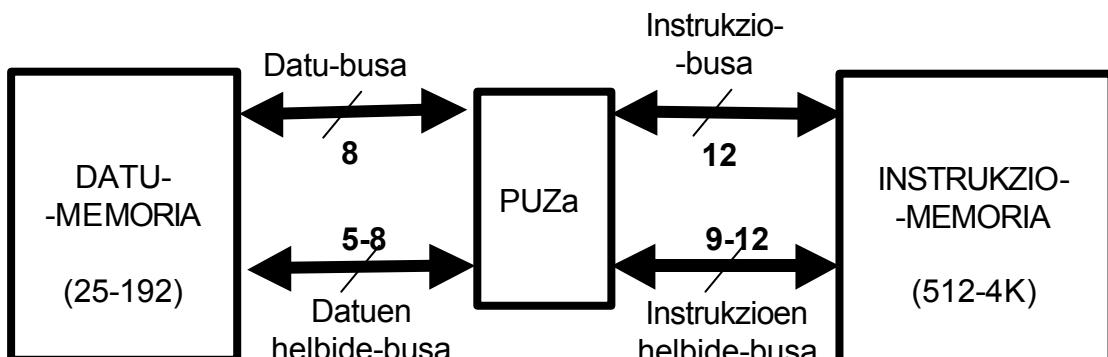
Mikrokontroladore hauek oso ahaltsuak dira eta egitura irekia daukate, hau da, busak kanpora atera daitezke memoria handitzeko. 8051ren parekoak dira ezaugarriei dagokienez. 16 biteko 58 instrukzio dituzte.

TXIPA	PROGRAM. MEMORIA (16 bit)	DATU MEM.	CLOCK	I/O	KAPTU-RA	PWM	ETEN- DURAK	TENPORIZA- DOREAK	PIN
<b>PIC17C42</b>	2k (eprom)	232	25MHz	33	2	2	11	TMR0-1-2-3	40
<b>PIC17C43</b>	4k (eprom)	454	25MHz	33	2	2	11	TMR0-1-2-3	40
<b>PIC17C44</b>	8k (eprom)	454	25MHz	33	2	2	11	TMR0-1-2-3	40

## 12.6. PIC-EN ARKITEKTURA

Gama guztiako PICek antzeko arkitektura daukate. Guk gama ertainekoekin egingo dugu azalpena, batez ere, 16F84rekin, hori baita arrakasta handiena lortu duena. PICek hiru ezaugarri berezi dituzte: RISC prozesadorea, instrukzioak egikaritzeko segmentazio-teknika eta HARDVARD arkitektura. Horrela, instrukzio gehienak, jauzietakoak izan ezik, ziklo batean egikaritzen dira.

HARDVARD arkitekturak ahalbidetzen du aldi berean eta independenteki datuetara eta instrukzioetara joatea.



12.2. irudia. Hardvard arkitektura.

Datuek eta instrukzioek luzera ezberdina izan dezakete; datuak 8 bitekoak dira eta instrukzioak 12, 14 edo 16 bitekoak. Gainera, bi helbide-bus independente dazkagu, horregatik joan daiteke aldi berean instrukzioetara eta datuetara.

### 12.6.1. PROGRAMA-MEMORIA

Programa-memoria: PIC16X84 mikroetan 0000H-tik 03FFH helbidera heltzen da, hau da, 1.024 posizio. PCaren pisu handieneko hiru bitak ez dira kontuan hartzen. Bitarrean adieraziz: x xx00 0000 0000-tik x xx11 1111 1111-ra, PCak 13 bit dauzka. Memoria honen 0000h helbidea *RESET* funtzoarentzat gordetzen da (*reset* egiten denean PC = 0000H) eta 0004H helbidea etendura-bektorearentzat.

Orrikatze-sistema erabiltzen du. Hori kontuan hartu behar da, batez ere, mihiztatzailean programatzeko, eta GOTO eta CALL instrukzioak erabiltzean.

### 12.6.2. DATU-MEMORIA. BERARIAZKO ERREGISTROAK

Datu-memoria bi bankutan banatzen da. Bi banku horien lehendabiziko posizioetan mikroaren berariazko erregistroak (SFR) kokatzen dira. Erregistro batzuk bi bankuetan errepikatuak daude errazago eskuratzeko (INDF, EGOERA, FSR...). Beste helbideak lanerako erabilgarriak dira. Hona hemen 16X84aren datu-memoria:

	<b>0 BANKUA</b>	<b>1 BANKUA</b>	
00	INDF	INDF	80
01	TMR0	OPTION	81
02	PCL	PCL	82
03	STATUS	STATUS	83
04	FSR	FSR	84
05	PORTA	TRISA	85
06	PORTB	TRISB	86
07			87
08			88
09			89
0A	PCLATCH	PCLATCH	8A
0B	INITCON	INITCON	8B
0C	PIR1	PIE1	8C
0D			8D
0E		PCON	8E
0F			8F
10			90
..			..
..			..
1F	CMCON	VRCON	9F
..			..
..			..
7F			FF

Grisez azaltzen diren posizioak ezin dira erabili eta irakurtzean beti 0 irakurtzen da. Reset baten ondoren, 0 bankua aukeratzen da automatikoki.

Nahi den bankua aukeratzeko, STATUS erregistroaren 5. bita (RP0) manipulatu behar dugu: RP0 = 0 → 0 bankua, RP0 = 1 → 1 bankua.

Memoria honez gain, 16X84 mikroak beste datu-memoria bat ere badu: 64 byteko EEPROM motakoa.

Taula honetan erregistroen xehetasunak ikusten dira:

Helb.	Izena	7 bita	6 bita	5 bita	4 bita	3 bita	2 bita	1 bita	0 bita	RESET ondoren	Beste RESET					
<b>0 BANKUA</b>																
00h	<b>INDF</b>	FSR edukia erabiltzen du helbide bezala								---	---					
01h	<b>TMR0</b>	8 biteko kontagailua/erlojua								xxxx xxxx	uuuu uuuu					
02h	<b>PCL</b>	PCaren pisu txikieneko bytea								0000 0000	0000 0000					
03h	<b>STATUS</b>	IRP	RP1	RP0	TO#	PD#	Z	DC	C	0001 1xxx	000q quuu					
04h	<b>FSR</b>	Zeharkako helbideratzea INDFaz								---x xxxx	uuuu uuuu					
05h	<b>A ATEA</b>	....	....	....	RA4/ T0CK	RA3	RA2	RA1	RA0	Xxxx xxxx	uuuu uuuu					
06h	<b>B ATEA</b>	RB7	RB6	RB5	RB4	RB3	RB2	RB2	RB1	Xxxx xxxx	uuuu uuuu					
07h		"0" irakurtzen da														
08h	<b>EEDAT</b>	EEPROM datu-erregistroa								Xxxx xxxx	uuuu uuuu					
09h	<b>EEADR</b>	EEPROM helbide-erregistroa								Xxxx xxxx	uuuu uuuu					
0Ah	<b>PCLATH</b>	...	...	...	PChren 5 bit					--- 0 0000	---0 0000					
0Bh	<b>INTCOM</b>	GIE	EEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF	0000 000x	0000 000u					
<b>1 BANKUA</b>																
80h	<b>INDF</b>	FSR edukia erabiltzen du helbide bezala								.....	.....					
81h	<b>OPTION</b>	RBPU#	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 111					
82h	<b>PCL</b>	PCaren pisu txikieneko bytea								0000 0000	0000 0000					
83h	<b>EGOERA</b>	IRP	RP1	RP0	TO#	PD#	Z	DC	C	0001 1xxx	000q quuu					
84h	<b>FSR</b>	Zeharkako helbideratzea INDFaz								Xxxx xxxx	uuuu uuuu					
85h	<b>TRISA</b>	...	...	....	A atearen konfigurazioa					--- 1 1111	---1 1111					
86h	<b>TRISB</b>	B atearen konfigurazioa								1111 1111	1111 1111					
87h		"0" irakurtzen da														
88h	<b>EECON1</b>	...	....	...	EEIF	WRERR	WREN	WR	RD	--- 0 x000	---- q000					
89h	<b>EECON2</b>	EEPROM kontrol-erregistroa								---	----					
0Ah	<b>PCLATH</b>	...	...	...	A atearen konfigurazioa					-- 0 0000	---0 0000					
0Bh	<b>INTCON</b>	GIE	EEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF	0000 000x	0000 000u					
0Ch																
...	<b>Erabilera orokorreko erregistroak (68)</b>															
4Fh																

Azter ditzagun erregistro horiek:

#### 12.6.2.1. STATUS (egoera)

UALaren eragiketen ondorioak (*flag*-ak).

R/W	R/W	R/W	R	R	R/W	R/W	R/W	
C: 8.	<b>IRP</b>	<b>RP1</b>	<b>RP0</b>	<b>TO</b>	<b>PD</b>	<b>Z</b>	<b>DC</b>	<b>C</b>

BITEAN

**BURURAKOA**

1: batuketa egitean *carry-a* sortu da.

0: *ez da carry-rik sortu.*

**DC: 4. BITEAN BURURAKOA (DOIKETA HAMARTARRA EGITEKO)**

1: batuketa egitean *carry* laguntzailea sortu da.

0: *ez da carry* laguntzailerik sortu.

**Z: ZERO**

1: emaitza 0 izan da

0: emaitza ez da 0 izan

**PD : Power Down**

1: balioa hartzen du V<sub>DD</sub> konektatzean edo CLRWDT egikaritzean

0: balioa hartzen du SLEEP instrukzioa egikaritzean.

**TO : Timer Out**

1: balioa hartzen du V<sub>DD</sub> konektatzean edo CLRWDT edo SLEEP egikaritzean.

0: balioa hartzen du WDT gainezkatzean

**IRP: ZEHARKAKO HELBIDERATZEAN DATU-MEMORIAKO BANKU-HAUTAKETA**

0: 0 eta 1 bankuak (00H-FFH)

1: 2 eta 3 bankuak (100H-1FFH)

**RP1-RP0: programa-memorian orrialdea hautatzeko.**

00: 0. orrialdea (00-7FH).

01: 1. orrialdea (80-FFH).

10: 2. orrialdea (100-17FH).

11: 3. orrialdea (180-1FFH).

**12.6.2.2. INTCON etenduren erregistroa**

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
<b>GIE</b>	<b>PEIE</b>	<b>TOIE</b>	<b>INTE</b>	<b>RBIE</b>	<b>TOIF</b>	<b>INTF</b>	<b>RBIF</b>

**GIE : etendura guztiak gaitzen ditu**

1 : etendurak gaituta

0 : etendura guztiak desgaituta

**EEIE : periferikoen etendura gitzeko (EEPROMean idazketa, konparatzailea...)**

1 : gaituta.

0 : desgaituta.

**TOIE : TMR0-k gainezka egiteagatik etendura gitzeko.**

1 : gaituta.

0 : desgaituta.

**INTE : RB0/INT HANKATXOAREN KANPOKO ETENDURA GAITZEKO.**

1 : gaituta.

0 : desgaituta.

**RBIE : RB7:RB4 HANKATXOAREN EGOERA-ALDAKETAGATIK ETENDURA GAIZTEKO**

1 : gaituta.

0 : desgaituta.

**TOIF : TMR0-K GAINEZKA EGIN DUELAKO SEINALEA (FLAG) GAINEZKATZEAN**

1 : TMR0-k ginezka egin du. Software bidez ezabatu behar da.

0 : TMR0-k ez du ginezka egin.

**INTF : RB0/INT KANPOKO ETENDURAREN EGOERAREN SEINALEA (FLAG)**

1 : kanpoko etendura aktibatu da. Software bidez ezabatu behar da.

0 : kanpoko etendura ez da aktibatu.

**RBIF : RB7:RB4 egoera aldatu den ala ez jakiteko seinalea (flag).**

1 : RB7:RB4 pinen baten balioa aldatu da. Software ezabatu behar da.

0 : RB7:RB4 pinetan ez da aldaketarik izan.

*GIE = 0 denean, ez da onartzen inolako etendurarak.*

*GIE = 1 denean, dagokion bitak baimendutako etendurak onartzen dira.*

### 12.6.2.3. EECON1

PIC16X84 mikroak badu EEPROM datu-memoria txiki bat (64 byte). Hor idatzitako datuak ez dira galduko elikadura etetean. Memoria-zati hori maneatzeko fitxategi berezi bat daukagu: EECON1.

***	***	***	<b>EEIFE</b>	<b>WRER</b>	<b>WREN</b>	<b>WR</b>	<b>RD</b>
-----	-----	-----	--------------	-------------	-------------	-----------	-----------

**EEIFE: IDAZKETA ERAGIKETAREN ETENDURAREN ADIERAZLEA**

1: idazketa amaitu da

0: idazketa ez da amaitu edo ez da hasi

**WRER: AKATS-ADIERAZLEA**

1: idazketa behar baino lehenago amaitu da

0: idazketa amaitu da

**WREN: IDAZTEKO BAIMENA EMATEN DU**

1: idazten uzten du

0: ezin da idatzi EEPROM memorian

**WR: IDAZKETAREN KONTROLA**

1: idazketa-ziklo bat hasi da.

0: idazketa-ziklo bat bukatu da.

**RD: IRAKURKETAREN KONTROLA**

1: irakurketa-ziklo bat hasi da.

0: irakurketa-ziklo bat bukatu da.

### 12.6.3. ERLOJUA ETA INSTRUKZIO-ZIKLOA

Mikrokontroladore hauetako gehienek 20MHz-ean lan egin dezakete, beraz, periodoa 50 ns-koa da. Instrukzio bat egikaritzeko, 4 erloju-periodo behar dira, beraz, 200 ns.

Lau osziladore-mota onartzen dituzte. OTP eta EPROM bertsioetan, txiparen kapsulan markatuta dago konektatu behar den osziladore-mota. FLASH eta EEPROM bertsioetan, edozein onartzen dute. Osziladore-mota hauexek dira:

- RC.** Kostu txikiko osziladorea, RC sare batez egina. Maiztasunean erdi-mailako egonkortasuna lortzen du.
- HS.** Abiadura handiko osziladorea (8-20MHz). Kuartzoz egina.
- XT.** Osziladore estandarra (100kHz-4MHz). Kuartzoz egina.
- LP.** Kontsumo txikiko osziladorea (32-200kHz). Kuartzoz egina.

Osziladore-mota horiek ***konfigurazio-hitzean*** hautatzen dira. Hitz hori txipa grabatzeko unean idazten da

-	-	-	-	-	CP	WDTE	FOSC1	FOSC0
---	---	---	---	---	----	------	-------	-------

#### FOSC1-FOSC0: osziladore-mota

- 11 = RC osziladorea
- 10 = HS (8-20MHz)
- 01 = XT (100kHz-4MHz)
- 00 = LP (32-200kHz)

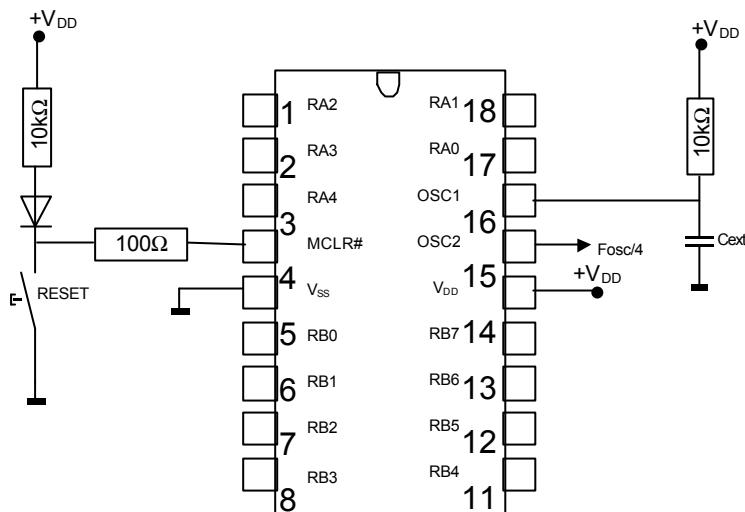
#### WDTE: *watchdog* aktibatzeko bita (WDT)

- 1 = WDT aktibatuta.
- 0 = WDT desaktibatuta.

#### CP: kodea babesteko bita. Segurtasunezkoa.

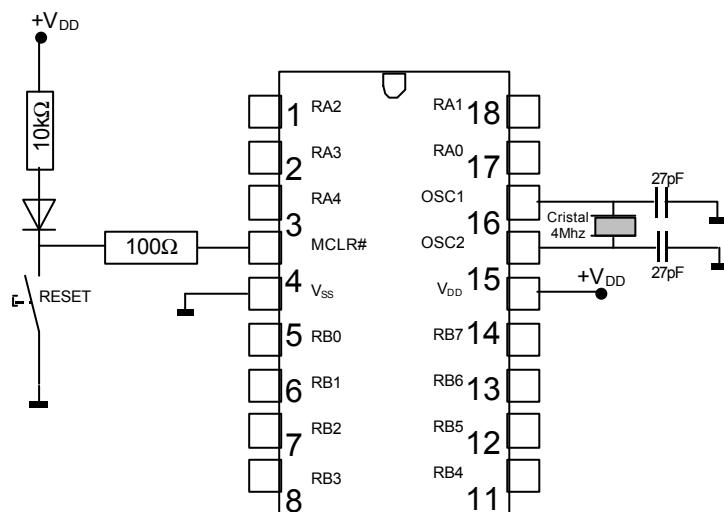
- 1 = kodea babestuta.
- 0 = kodea babestu gabe.

### 12.6.3.1. Oszilazio-zirkuituak



RC motako osziladorea, kanpotiko kondentsadore batez eta erresistentzia batez osatua. Ondoko taulan, maiztasun zehatz batzuk lortzeko erabili behar diren balioak adierazten dira.

F	R	C
625kHz	10k	20pF
80kHz	10k	220pF
80Hz	10k	0,1uF



XT motako osziladorea, kuartzo-kristalezko osziladoreaz. 100kHz-4MHz bitarteko maiztasun estandarra.

## 12.7. PIC16X84 TXIPAREN HANKATXOEN DESKRIBAPENA

**V<sub>DD</sub>** : elikadura (+5v)

**V<sub>SS</sub>** : lurra.

**OSC1/CLKIN**: osziladoretik zirkuitura sartzeko, kuartzo edo RC sare batekin.

**OSC2/CLKOUT**: osziladorea zirkuitutik irteteko, kuartzo edo RC sare batekin.

**T0CKI:** hankatxo honetatik TMR0 temporizadoreak erabiltzen duen kanpoko erloju-seinalea sartzen da. Erabiltzen ez bada, masa edo positibora konektatzea gomendatzen da kontsumoa txikitzeo.

**MCLR/Vpp:** hankatxo honetatik zeroa sartuz mikroa berrasieratzen da. Grabatzean, hankatxo honetatik grabatzeko tentsioa sartzen da.

**RA0-RA3:** A atakaren sarrera-irteerako lau linea.

**RB0-RB7:** B atakaren sarrera-irteerako zortzi linea.

## 12.8. PIC16X84 MIKROAREN LABURPEN-TAULA

---

PROGRAMA-MEMORIA : 1K x 14, EEPROM (PIC16C84) edo FLASH (PIC16F84)
DATUEN RAM MEMORIA : 22 byte SFR (berariazko erregistroak) eta 36 byte GPR (erregistro orokorrak) (PIC16C84) / 68 byte (PIC16F84)
DATUEN EEPROM MEMORIA : 64 byte bi modeloetan
PILA (Stack) : 8 mailakoa
ETENDURAK : 4 mota ezberdin
INSTRUKZIOAK : 35
KAPSULA : 18 hankatxoko plastikozko DIPa
LAN-MAIZTASUN MAXIMO : 10 MHz maximoa
TENPORIZADOREAK : TMR0 eta “zakur zaindaria” (WDT)
S/I LERRO DIGITALAK : 13 (5 A atean eta 8 B atean)
ELIKADURA : 2-6 V                    GRABAZIO-ELIKADURA : 12-14 V
Xurga dezakeen korronte maximoa : 80 mA A atek eta 150 mA B atek
Eman dezakeen korronte maximoa : 50 mA A atek eta 100 mA B atek
Lerroko xurga dezakeen korronte maximoa : 25 mA
Lerroko eman dezakeen korronte maximoa : 20 mA

## 12.9. TENPORIZADOREAK

---

Zirkuitu hauen betebeharra denbora-tarteak markatzea edo kanpotiko pultsuak kontatzea da. PICetan ekintza horiek programa nagusian zehaztu beharrean, mikroprozesadoreetan egiten den bezala, baliabide zehatz batzuen bidez egiten da.

PIC16X8X-k 8 biteko temporizadore/kontagailua dute: TMR0 Bi eratara erabiltzen da:

- RA4/TOCK1 hankatxotik sartzen zaizkion pultsuen kontagailu bezala. FFH baliora heltzen denean, berriz 00H-ra pasatzen da, eta adierazle edo etendura bat aktibatzen du.
- Temporizadore bezala. Horretarako OPTION konfiguratu behar da.

PIC hauek beste temporizadore bat ere badute, (*Watchdog*) Zakur Zaindaria; horren eginkizuna programa “ez esekitzea” da. WDTa denbora zehatz baterako programatzen da, eta programa begizta infinitu batean sartu delako edo denbora hori gainditzen badu, ezer egin gabe sistema osoa berrasieratzen du WDTk.

TMR0k eta WDTk denbora luzeak kontrolatu behar dituztenerako maiztasun-zatitzairen bat daukate, eta OPTION erregistroan programatu behar da. TMR0 eginkizun orokorreko erregistroa da (SFR) 01 helbideko 0 bankuan dago, helbide berdineko 1 bankuan OPTION erregistroa dago.

Temporalizazioa neurtzeko bi formula hauek dauzkagu:

$$\text{temporalizazioa} = 4 \times \text{Tosc} \times (0x100 - \text{kargatutako balioa}) \times (\text{zatitzairen})$$

$$\text{Kargatzeko balioa} = 0x100 - \frac{\text{temporalizazioa}}{4 \times \text{Tosc} \times \text{zatitzairen}}$$

### 12.9.1. OPTION

R/W	R/W	R/W	R	R	R/W	R/W	R/W
RBPO#	INTEDG	TOCS	TOSE	Psa	PS2	PS1	PS0

#### PS2-PS1-PS0: maiztasun zatitzairenaren balioa markatzeko

PS2	PS1	PS0	TMR0aren zatidura	WDTren zatidura
0	0	0	1:2	1:1
0	0	1	1:4	1:2
0	1	0	1:8	1:4
0	1	1	1:16	1:8
1	0	0	1:32	1:16
1	0	1	1:64	1:32
1	1	0	1:128	1:64
1	1	1	1:256	1:128

**PSA : MAIZTASUN-ZATITZAILEA ESLEITU**

- 1 : maiztasun-zatitzalea WDTri esleitzen zaio.
- 0 : maiztasun-zatitzalea TMR0ri esleitzen zaio.

**TOSE: TOCK1en fronte-mota. (TMR0rako)**

- 1 : beheranzko frontea
- 0 : goranzko frontea

**TOCS: TMR0-rako erloju-mota**

- 1 : kontagailua, TOCK1etik sartutako pultsuak kontatzen ditu.
- 0 : temporizadorea, Fosc/4

**INTEDG:etenduraren fronte aktiboa**

- 1 : goranzkoa.
- 0 : beheranzkoa.

**RBPO : B atearen pull-up erresistentziak**

- 1 : desaktibatuak.
- 0 : aktibatuak

## 12.10. SARRERA/IRTEERA ATEAK

---

Sarrera/irteera ate bi bakarrik dauzkate PIC16X8X-ek. A ateak bost lerro ditu: RA0-RA4. RA4k bi funtzi betetzen ditu: S/I-ko lerroa, eta T0CK1, TMR0 temporizadorearen pultsu-sarrera. B ateak 8 lerro ditu: RB0-RB7. RB0k bi funtzi ditu: RB0/INT, sarrera/irteera eta kanpotiko etendura-sarrera.

RA3-RA0 lerroek sarrera bezala TTL mailak onartzen dituzte eta CMOS irteera bezala. RA4/T0CK1 lerroak *Schmitt Trigger* bat dauka, zaratarekiko babes ona izanik, eta irteeran drainatzaile irekia du.

**TRISA erregistroko bitak A atearen lerroak sarrera bezala programatzen ditu “1”-ean badaude, eta irteera bezala “0”-an badaude.**

Irteera bezala jokatzen duten lerroak “lach” bihurtuta daude, hau da, A atearen erregistroan (05) kargaturiko azken maila logikoa ematen dute.

**Movf atea,w** instrukzioaz atea irakurtzen dugu eta **Movwf atea** instrukzioaz idatzi.

**Bsf atea,bit** eta **Bcf atea,bit** instrukzioez atearen bit zehatza alda daiteke.

B atearen informazioa 06 helbideko 0 bankuko erregistroan daukagu eta 1 bankuan eta helbide berean dago TRISB erregistroa.

Ate horren lerro guztiei balio handiko *pull-up* erresistentziak konektatzen diren erakundeak elikaduraren polo positibora; horretarako, OPTION erregistroaren RBPU# bita = 0. Reset egin ondoren, lerro guztiak sarrera bezala geratzen dira, eta *pull-up* erresistentziak deskonektatuak.

RB6 grabaziorako serieko lerroa da eta RB7 serieko datuen sarrera da.

Bestalde, RB4-RB7 pisu handieneko lau lerroak, sarrera bezala daudenean, beren maila logikoaren aldaketa detektatzeko moduan programa daitezke. Horietako bat aldatzen denean, etendura sortzen da. Teklatuaren kontrolerako erabiltzen dira.

Adibidea.: A atearen pisu txikieneko bi lerroak irteera bezala eta beste hirurak sarrera bezala programatu:

Bsf	egoera, rp0	; 1 bankua aukeratu
Movlw	0X1C	; A atearen konfigurazioa (TRISA)
Movwf	trisa	; RA4-RA2 sarrerak, RA0-RA1 irteerak

## 12.11. EEPROM MEMORIA

Memoria hau 0x0000tik 0x1FFFra doa, eta 8K dira; baina horietatik 1K bakarrik dago implementatua, 0x0000tik 0x3FFFra doana.

0	IMPLEMENTATUA	PROGRAMA-MEMORIA
3FF	IMPLEMENTATU GABE	KONFIGURAZIO-MEMORIA
400	ID POSIZIOA	
1FFF	ID POSIZIOA	
2000	ID POSIZIOA	
2001	ID POSIZIOA	
2002	ID POSIZIOA	
2003	ID POSIZIOA	
2004	ERRESERBATUA	
2005	ERRESERBATUA	
2006	ERRESERBATUA	
2007	KONFIGURAZIO-HITZA	
200F	IMPLEMENTATUA	
2010	IMPLEMENTATU GABE	
3FFF	IMPLEMENTATU GABE	

## 12.12. ETENDURAK

PICek lau etendura-iturri dituzte: **RB0/INT** hankatxoan aktibatzea (INTF adierazlea); **TMR0-k** gainezka egitea gainezkatzea (TOIF adierazlea); **RB7:4** hankatxoen egoera-aldaera (RBIF); **EEPROM** memorian idazketa-amaiera (EEIFE).

Lehen hirurak INTCON erregistroan daude eta laugarrena EECON1 erregistroan. Denak PCa 04 helbidera bidaltzen dute, baina ezer egin baino lehen, etendura zeinek sortu duen jakin beharko dugu adierazleei begiratuz eta gero azpiprograma bat edo bestea exekutatuz. Horretarako dugu hurrengo programa:

```
Btfsc intcon,1      ; INTF=0 bada, jauzia
Call Kanpo        ; kanpotiko etendura denez ...
Btfsc intcon,2      ; TOIF = 0 → jauzia
Call Timer
Btfsc intcon,0      ; RBIF=0 → jauzia
Call RB74
Btfsc eecon1,4      ; EEIF=0 → jauzia
Call EEPROM
```

## 12.13. BIHURGAILU ANALOGIKO/DIGITALA

16C71 mikrokontroladoreak duen bihurgailu A/D 8 bitekoa da, eta 4 sarbideko multiplexadorea du. Sarbide-kopurua eta horien izaera softwarez kontrolatzen da, kontroleko erregistroan biten egoera aukeratzu.

Bihurketarako hartzen den erreferentzia-tentsioa  $V_{DD}$  edo  $RA_3$  hankatxoan ezarritakoa izan daiteke. Bihurgailuaren sarrerak A atekoak dira, eta horiek sarrera gisa konfiguratu behar dira.

Bihurgailuak atzipen eta laginketa-teknika (*Sample&Hold*) erabiltzen duenez, sarrerako multiplexadoreak ez dauka beti konektatua egon beharrik, atzipen-denboran bakarrik baizik, denbora hori PICaren erlojuaren maiztasunaren eta nahi dugun bihurketa-abiaduraren menpe dago.

Berariazko hiru erregistro erabiltzen ditu: kontroleko bi, ADCON0 (0 bankuaren 08 helbidean) eta ADCON1 (1 bankuko 08 helbidean), eta emaitzarena, ADRES (0 eta 1 bankuko 09 helbidean).

<b>ADCS1</b>	<b>DACS0</b>	-----	<b>CHS1</b>	<b>CHS0</b>	<b>GO/DONE</b>	<b>ADIF</b>	<b>ADON</b>
--------------	--------------	-------	-------------	-------------	----------------	-------------	-------------

### 12.13.1. ADCON0

**ADON.** Bit honek 1ean egon behar du bihurgailua aktibatzeko, 0an badago ez du energiarik xahutzen eta desgaituta dago.

**ADIF (AD Interrupt Flag).** Bit hau 1ean ipintzen da bihurketa bukatzen denean. 0an softwarez ipini behar da.

**GO/DONE.** Bit hau 1ean ipintzean bihurketa abiarazten da, eta 1ean mantentzen da bihurketak irauten duen bitartean. 0ra pasatzen da bihurketa bukatzean, emaitza ADRESen irakur daitekeen seinalea da.

**CHS0 eta CHS1 (Chanel Select).** Bit hauek aukeratzen dituzte sarbideak, alboko taulan adierazten den eran. PIC16C74 mikroak baditu 8 kanal, beraz, beste bit bat gehiago behar du kanala hautatzeko: CHS2. Bit hori 5. bitean kokatzen da.

CHS1	CHS0	HAUTATUTAKO KANALA
0	0	0 kanala (RA0/AN0)
0	1	1 kanala (RA1/AN1)
1	0	2 kanala (RA2/AN2)
1	1	3 kanala ( $V_{REF}$ /AN3)

**5. bita:** ez da erabiltzen

**ADCS1 eta ADCS0: AD Clock Select.** Bit hauekin bihurketa-erlojuaren maiztasuna aukeratzen da, alboko taulan adierazten den eran.

ADCS1	ADCS0	BIHURKETA-ERLOJUA
0	0	$F_{OSZ}/2$
0	1	$F_{OSZ}/8$
1	0	$F_{OSZ}/32$
1	1	RC BARNE-ERLOJUA

Reset egindakoan denak 0an ipintzen dira.

### 12.13.2. ADCON1

---	---	---	---	---	---	PCFG1	PCFG0
-----	-----	-----	-----	-----	-----	-------	-------

0 eta 1 bitak: *Port Configuration Port*. Bit hauek A atearen pisu txikieneko lau hankatxoak definitzeko balio digute, ondorengo taulan adierazten den eran:

PCFG1	PCFG0	RA0	RA1	RA2	RA3	$V_{REF}$
0	0	A	A	A	A	$V_{DD}$
0	1	A	A	A	$V_{REF}$	Ra3
1	0	A	A	D	D	$V_{DD}$
1	1	D	D	D	D	$V_{DD}$

A = sarrera analogikoa

D = sarrera/ irteera digitala

Programatzean honako pausoak eman behar dira:

- Bihurgailua konfiguratu, kontroleko erregistroko bitak programatz.
- Bihurgailutik sor daitezkeen etendurak gaitu eta aktibatu edo ez INTCON erregistroa programatz.
- Bihurketa abiarazi GO/DONE bita 1ean ipiniz.
- Bihurketaren bukaera detektatu, aipaturiko bitari begiratuz (0an), edo gaitua izan bada, bihurgailuak bidaltzen duen etenduraz irakur daiteke.
- Emaitza ADRES erregistroan.

Ondoren, beste bihurketa bat abiaraz daiteke, eta hori beste sarbide batekoa izan daiteke. Baino denbora bat itxaron behar da ( $t \geq 2 T_{atz}$ ).

ADON eta GO/DONE bitak ezin dira instrukzio berdinaz 1ean ipini, bi instrukzio erabiliz baizik.

$T_{atz}$  edo atzipen-denborak  $2 \mu s$  baino handiagoa izan behar du, eta bihurketa-erlojua aukeratzeko, bihurketak  $10T_{atz}$  segundo behar dituela izan behar dugu kontuan.

## 12.14. ATSEDEN-EGOERA (SLEEP)

---

Egoera honetan sartuz gero, kontsumoa asko txikitzen da. Modu honetan SLEEP instrukzioa egikaritzen hasten da eta hurrengo hau gertatzen da:

- a) *Watchdog* martxan baldin bazegoen, bere balioa zeroan jartzen da, baina kontatzen jarraitzen du.
- b) Sistemaren osziladorea gelditu egiten da.
- c) Sarrera/irteerako ateek zeuzkaten balioak mantentzen dituzte.
- d) Egoera-erregistroko PD eta TO bitek 1 eta 0 balioak hartzen dituzte, hurrenez hurren.

Atseden-egoera honetatik ateratzeko hiru aukera daude:

- 1) Kanpoko reset bat sortu, MCLR hankatxoa aktibatuz.
- 2) Zakur zaindariak (*Watchdog*) gainezka egiten du eta reset bat sortzen du. Horretarako, aktibatua egon behar zuen SLEEP egoeran sartu aurretik.
- 3) Kanpoko etendura bat sortzen da.

Kontsumoa gehiago jaisteko, T0CK1 hankatxoa  $V_{DD}$ -ra edo lurrera eta MCLR/V<sub>PP</sub>  $V_{DD}$ -ra konektatzea gomendatzen da.

## 12.15. INSTRUKZIOAK

ERREGISTROAK MANEIATZEN DITUZTEN INSTRUKZIOAK			
SINTAXIA	ERAGIKETA	ZIK.	ADIERAZLEAK
ADDWF f,d	W eta f batu, emaitza W-ra (d=0) edo f-ra (d=1)	1	C, DC, Z
ANDWF f,d	AND W eta f, emaitza W-ra (d=0) edo f-ra (d=1)	1	Z
CLRF f	f ezabatzen du (bitak 0an)	1	Z
CLRW	W ezabatzen du	1	Z
COMF f,d	f alderantzikatu (osagarria)	1	Z
DECF f,d	f gutxitu, emaitza W-ra (d=0) edo f-ra (d=1)	1	Z
INCF f,d	f gehitu, emaitza W-ra (d=0) edo f-ra (d=1)	1	Z
IORWF f,d	W eta f artean OR, emaitza W-ra (d=0) edo f-ra (d=1)	1	Z
MOVF f,d	f mugitu, W-ra (d=0) edo f-ra (d=1)	1	Z
MOVWF f	W f-ra mugitzen du	1	Z
NOP	Ez du ezer egiten	1	Z
RLF f,d	f ezkerrera biratu bururakoaz	1	Z
RRF f,d	f eskuinera biratu bururakoaz	1	Z
SUBWF f,d	f-ri W kendu, emaitza W-ra (d=0) edo f-ra (d=1)	1	Z
SWAPF f,d	Nibble-ak trukatu	1	Z
XORWF f,d	W eta f-ren XOR, emaitza W-ra (d=0) edo f-ra (d=1)	1	Z
BITAK MANEIATZEN DITUZTEN INSTRUKZIOAK			
BCF f,b	f-ren b bita ezabatzen du	1	....
BSF f,b	1ean ipintzen du f-ren b bita	1	....
“JAUZI” INSTRUKZIOAK			
BTFSZ f,b	f-ren bit bat aztertu eta jauzia 0 bada	1(2)	....
BTFSZ f,b	f-ren bita aztertu eta jauzia 1 bada	1(2)	....
DECFSZ f,d	f gutxitu eta jauzia 0 bada	1(2)	....
INCFSZ f,d	f gutxitu eta jauzia 1 bada	1(2)	....
BEREHALAKO ERAGILEAK MANEIATZEN DITUZTEN INSTRUKZIOAK			
ADDLW k	Berehalako batuketa W-az	1	C, DC, Z
ANDLW k	Berehalako AND W-az	1	Z
IORLW k	Berehalako OR W-az	1	Z
MOVLW k	W-ra berehalako datua pasatu	1	....
SUBLW k	Berehalako datuari W kendu	1	C, DC, Z
XORLW k	W-az XOR	1	....
KONTROLERAKOAK ETA INSTRUKZIO BEREZIAK			
CALL k	Azpiprogramari deia	2	TO#, PD#
CLRWDT	Zakur zaindaria ezabatu edo berritu	1	....
GOTO k	Baldintzarik gabeko jauzia	2	....
RETFIE	Etenduraren itzulera (GIE=1)	2	....
RETLW k	Azpiprogramaren itzulera eta W=k kargatu	2	....
RETURN	Azpiprogramaren itzulera	2	....
SLEEP	Atseden-egoerara pasatzen da	1	TO#, PD#
ESANAHIA			
f	Iturburu-erregistroa. 7 bitekoa → 0x00-tik 0x7F-raino edozein izan daiteke		
k	Berehalako datua		
d	Helburu-erregistroa. “0” denean W da eta “1” denean iturburukoa bera		
b	0.....7 bita		
“jauziak”	Baldintza betetzen denean, jarraian duten instrukzioa “saltatzen” dute		

## 12.16. SOFTWAREA

---

Bere izena PICME\_TR da eta MS-DOS sistema eragilearen pean lan egiten du. Software honekin jarduteko, plakak ordenagailuari konektatua egin behar du eta RB6 eta RB7 Ierroetako beste periferiko guztiak deskonektatuta.

### 12.16.1. PROGRAMAZIOA

Lehendabizi, editore bat erabiliz, **XX.ASM** fitxategia sortu behar dugu, arau batzuk kontuan izanik: *zutabeetan idatzi eta lehen Ierroetan mihiztatzailearen sasinstrukzioak idatzi*.

**Lehen zutabean** etiketak joango dira (32 karaktere bitartekoak). Gogoratu etiketak azpiprogramen izenak edota helbide bat direla (jauziak).

**Bigarren zutabean** instrukzio-kodea doa, edo mihiztatzailearen sasinstrukzioa (EQU,ORG,LIST...).

**Hirugarren zutabea** datuen eremua da. Datua erregistro baten edukia, erregistro baten bita, etiketa edo konstante bat izan daiteke. Datu bat baino gehiago badoaz, komaz bereiziko dira.

**Laugarren zutabean** iruzkinak eta oharrak doaz, (:) puntu eta komaren ondoren.

Hurrengo pausoa idatzi dugun fitxategia mihiztatzea edo makina-kodera itzultzea izango da; horretarako, MICROCHIP etxeko MPASM mihiztatzailea exekutatuko dugu. Programa horrek sortzen dituen fitxategiak honakoak hauek dira:

- XX.ERR → Mihiztatzean sorturiko erroreentzako zerrenda.
- XX.REF → Gurutzaturiko erreferentziak zerrendaren fitxategia (etiketen zerrenda).
- XX.LST → Inprimatuko dugun fitxategia, interesatzen zaigun informazioa duena.
- XX.HEX → Exekutagarria, makina-kodean dagoena eta PICCari kargatzeko dioguna.
- XX.OBJ → Espreski adierazten ez badugu, ez da sortzen.

Bukatzeko, PICa grabatu aurretik simulazio birtuala (PICME28) egitea komenida.

## 12.17. **m**PIC TRAINER ARKITEKTURA

---

$\mu$ PIC Trainer sistema hau ARIZONA MICROCHIPS INC.-en PIC mikrokontroladoreen zenbait aplikazio praktiko ebaluatzeko prestatuta dago.

Telefono mugikorrok, edozein motatako alarmak, tresna elektrikoen automatizazioa eta kontrola, automobilgintza, instrumentazioa, prozesu industrialetako datuen ikuskatzea, bistaratzeara eta jasotzea dira mikrokontroladoreen aplikazio tipikoak.

Garapen-sistema honek oinarrizko S/I-ko periferikoak dauzka aplikazio praktikoen funtzionamendua egiaztatzeko, baita 18/28 hankatxoko PIC guztiak grabatzeko beharrezkoak diren zirkuituak ere.

Sistemaren ezaugarriak honakoak hauek dira:

1. Elikadura 12 VAC-eko transformadore batez edo 9 VDC-eko bi pilaz. Finkatze-zirkuitua plakan dago, + 5V ( $V_{KZ}$ ) tentsio nagusia eta grabazioetarako +13,8 V lortuz.
2. 18 eta 28 hankatxo bitarteko edozein PIC onartzen du. Lan-maiztasuna 4 MHz-ekoa da (kuartzozko kristala aldatuz alda daitekeena).
3. RESET automatikoa (POR) eta eskuzkoa, erabiltzaileak pultsadorea sakatuz.
4. Bost kommutadorez aktibaturiko 5 sarrera digital.
5. Potentziometro bidez aldagariak diren 4 sarrera analogiko.
6. 8 irteera digital LED erako adierazle argidunetara edo/eta 7 segmentuko *display*-ra konektatuak. *Jumper* batzuen bitartez aukeratzen dira.
7. Bistaratzeko modulua LCD erakoa 2 X 16 (16 karaktereko 2 lerroduna).
8. Sarrerako edo irteerako periferikoak dagokien *jumper*-ez aktiba edo desaktiba daitezke. Karga elektrikoa gutxitzeko erabiltzen ez direnak deskonektatzen dira.
9. PIC-BUS hedapena. S/I guztiak beste periferiko batzuekin erabil daitezke.
10. RA4/T0CKI sarrera edo irteera digital bezala erabil daiteke, edota 0 *timer*-arentzat erloju-sarrera bezala.
11. RB0/INT sarrera edo irteera digitala izan daiteke, edo kanpotiko etendura.
12. Zirkuitu grabatzailea. Zokaloan dagoen PICa grabatzen du. 16C84 modeloarentzat oso aproposa da, EEPROM memoria duenez, nahi bestetan graba dezakegulako.
13. Grabaketa adierazten duen LED adierazlea (VPP tentsioa).
14. DB25 konektorea, ordenagailu baten ataka paralelora konektatzeko.

### 12.17.1. ELIKATZE-ITURRIA

Korronte alternoa aplikatzen bazaio, D<sub>1</sub> zubiak artezten du eta C<sub>1</sub>, C<sub>2</sub> kondentsadoreek iragazten dute. Horrela lorturiko korronte zuzena +5V-ean finkatzen da 7805 erreguladoreaz (U<sub>6</sub>) eta + 13,8 V-ean 7812 eta 1N4007 D<sub>2</sub>-D<sub>4</sub> hiru diodoez.

### 12.17.2. MIKROKONTROLADOREA

Bi zokalo daude, 18 nahiz 28 hankatxoko PICak erabili ahal izateko. Lehenak 5 lerroko (RA0-RA4) A atea eta 8 lerroko (RB0-RB7) B atea ditu; bigarrenak 6 lerroko (RA0-RA5) A atea, 8 lerroko (RB0-RB7) B atea eta 8 lerroko (RC0-RC7) C atea ditu. Plaka honetan A ateko 5 lerroak eta B-ko 8 lerroak erabiltzen dira, PIC guztientzat komunak direnak. Besteak PIC-BUS hedapenean daude.

Lan-maiztasuna X<sub>1</sub> kuartzozko kristalak eta C<sub>6</sub>, C<sub>7</sub> kondentsadoreek finkatzen dute 4 MHz-ean.

### 12.17.3. SARRERA DIGITALAK

A atearen RA0-RA4 lerroei konektaturiko bost etengailuk (SW2-SW6) sortzen dituzte. RA0-RA3 lerroak sarrera analogikoak ere izan daitezke eta RA4 (T0CKI) 0 *timer*-aren erloju-pultsuen sarrera, horrela programatz gero. J8(0) – J8(3) *jumper*-ek aukeratzen dute RA0-RA3 lerro bakoitzak digitala (SW2-SW5) edo analogikoa (P2-P5) izan behar duen, edo deskonektatuak geratuko diren beste periferiko batzuekin erabiltzeko.

J9 *jumper*-ak erabakitzentz du RA4 SW6 etengailuaz sarrera digitala den, J2(1) konektoreaz kanpotik aplikaturiko T0CKI-ren erloju-sarrera den edo deskonektatua geratu behar duen.

### 12.17.4. SARRERA ANALOGIKOAK

P2-P5 potentziometroak sortzen dituzte 4 sarrera hauek. 0V-etik +5V-erako tensio aldakorra lortzen da. J8(0) – J8(3) *jumper*-ek erabakitzentz du PICaren atean sartzen diren ala ez.

### 12.17.5. IRTEERA DIGITALAK

B atera konektatuak daude. Batetik, argidun LED diodoak daude, RB0-RB7 seinaleen egoera adierazten digutenak eta, bestetik, 7 segmentuko zenbakizko *display*-a. Gehiegizko kontsumoa ekiditeko, J7 eta J5 *jumper*-ak daude, behar ez direnean deskonektatzeko.

RPACK 1 erresistentzia-integratuak 330 Ω-eko 8 erresistentzia mugatzaire ditu.

D8 barrak LED erako 10 diodo ditu. 8 lehenak S0-S7, J7-az gaituak daudenean, RB0-RB7 seinaleen egoera adierazten dute. Bederatzigarrena ez da erabiltzen eta hamargarrenak +5V (V<sub>KZ</sub> ON) tentsioaren presentzia adierazten du.

RB0 lerroak INT kanpotiko etendura-sarrera izan daiteke, seinale hori J2(3) konektoretik aplikatzen da. J10 *jumper*-az lortzen dugu RB0 seinalea INT etendurara, LEDetara edo *display*-ra konektatzea. Bestalde, B atera konektaturiko periferiko guztiak deskonekta ditzakegu dagokien *jumper*-ez. PIC-BUS hedapenean beste aplikazio batzuetarako erabil daitezke.

#### 12.17.6. LCD MODULUA

Kristal likidozko bistaratzeko modulu da, 16 karaktereko 2 lerro dituena. Kode ezbardinekin efektu bereziak lor daitezke: keinuak, *scroll*, e.a. Karaktere bereziak ere sor daitezke.

D0-D7 datu-lerroak B atearen 8 lerroetara konektatuak daude (RB0-RB7). Ate horrek PICaren irteera eta moduluaren sarrera bezala jokatzen du batzuetan, eta besteetan, PICerako sarrera bezala.

Bestalde, modulu hau PICaren A ateko RA0, RA1 eta RA2 lerroetara konektatua dago eta hurrengo kontrol-seinaleak bidaltzeko erabiltzen dira:

- **RS = 0** Instrukzio-erregistroaren hautaketa
- **RS = 1** Datu-erregistroaren hautaketa
- **R/W = 0** LCD moduluan idazketa-zikloa
- **R/W = 1** LCD moduluan irakurketa-zikloa
- **E = 0** LCD modulua desaktibatua
- **E = 1** LCD modulua aktibatua

E seinale hori RA2ra J6 *jumper*-az konektatzen da. J6 itxita badago, RA2k kontrolatzen du modulu; baina irekita badago, E seinalea lurrera (0 maila) konektatua geratzen da R16an zehar, eta modulu deskonektatua geratzen da.

#### 12.17.7. ZIRKUITU GRABATZAILEA

“Zirkuituan” dagoela PICaren grabazioa (irakurri, idatzi, ezabatu...) ahalbidetzen du. J3 konektorearen bitartez, PC/XT/AT ordenagailuen ataka paralelora (LPT1) konekta daiteke, kontroleko softwareak kudeatzen ditu eragiketa guztiak (irakurri, ezabatu, konfigurazio-hitza aldatu...).

Eragiketa horiek seriean burutzen dira. Datuen bitak RB7an zehar sekuentzialki irakurri edo idazten dira, eta U4D eta U4F amplifikadoreek ataka paraleloa aplikatzen dituzte.

RB6 seinaleak erloju-sarrera bezala jokatzen du datuak sinkronizatzeko, softwareak sortzen duen seinaleaz.

Q1 kommutazio-transistoreari U4B amplifikadoreaz aplikatzen zaion seinalea ere softwareak sortzen du, PICaren MCLR/VPP hankatxoan +13,8 V ezarriz.

Azkenik, softwareak U4A amplifikadoreaz MCLR seinalea sortzen du, mikrokontroladorea resesteatu eta abiarazten duena.

PICaren irakurketa/grabaketa seinaleak RB6 eta RB7 direnez, oso garrantzitsua da J5, J6 eta J7 *jumper*-ekin beste periferikoak (LED, *display*-a, LCD modulua) deskonektatzea informazioa alda ez dezaten.

#### **12.17.8. HEDAPEN-KONEKTOREA**

26 kontaktu ditu eta bertan seinale guztiak eskura daude, beste periferiko batuetara konektatzeko edo nahi denerako. Ez ahaztu kanpoan erabili behar diren seinaleak *jumper*-ekin plakan deskonektatzeaz.

## Galdeketa

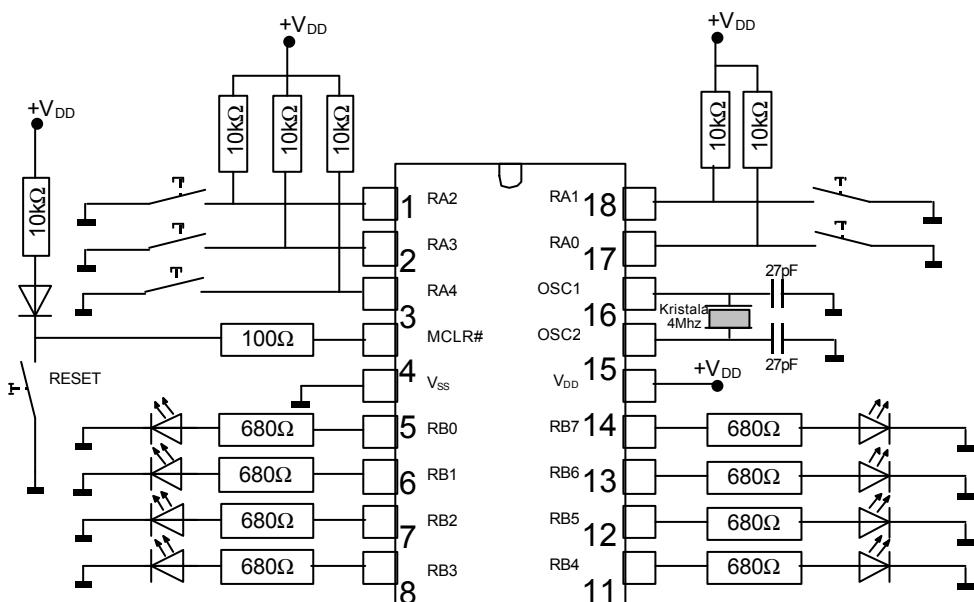
- 12.1)** PIC16C84-k 4 MHz-eko kristalaz funtzionatzen badu eta Zakur Zaindaria 18 ms-ra gainezka egiteko programatua badago, kalkula ezazu zenbat instrukziotik behin berritu beharko den Zakurra **Clrwt** instrukzioaz.
- 12.2)** Azal itzazu “itzulerako” hiru instrukzioen arteko differentziak  
**Return // Retlw k //Retfie**
- 12.3)** Zein instrukzio erabiliko zenuke akumuladorea unitate batean gehitzeko (+1)?
- 12.4)** Eta W gutxitzeko? **Addlw 0xff**
- 12.5)** Eta W alderantzikatzeko ? **Xorlw 0xff**
- 12.6)** W-aren biko osagarria lortzeko? **Xorlw 0xff + addlw 0x01**
- 12.7)** W-aren zenbait bit zeroan ipintzeko besteak aldatu gabe?  
**Andlw 0xXX** (adibideak)
- 12.8)** W-aren zenbait bit batean ipintzeko besteak aldatu gabe? **Iorlw 0xXX**
- 12.9)** Nola egiaztu bi erregistroen edukia berdina dela?  
Movf reg1,0  
Xorwf reg2,0  
Btfsc egoera,2  
Call berdin
- 12.10)** Nola egiaztu erregistro baten edukia beste batena baino handiagoa dela?  
Movf reg1,0  
Subwf reg2,0  
Btfss egoera,0  
Call handi
- 12.11)** Nola egiaztu erregistro baten edukia beste batena baino txikiagoa dela?  
Movf reg2,0  
Subwf reg1,0  
Btfss egoera,0  
Call txiki

- 12.12) FSR erregistroaren edukia 0-raino gutxitzea nahi dugu eta gero W ezabatu.
- 12.13) EGOERA erregistroaren 4. bitaren balioa zein den jakin nahi dugu. 1 balio badu, W ezabatzen da. Eta 0 balio badu, bit hori 1ean ipintzen da.
- 12.14) Hasieran W = 0xff eta OPTION = 0x00 badira, nola geratuko dira ondorengo programa exekutatutakoan?
- |      |          |
|------|----------|
| Bsf  | option,2 |
| Comf | option,0 |
| Swap | option,1 |
- 12.15) Suposa dezagun hasieran 0 balio dutela W-k eta datuen memoriako 0x01 eta 0x02 helbideko-erregistroek, zein balio izango dute programa hau exekutatu ondoren?
- |       |        |
|-------|--------|
| Incf  | 0x01,0 |
| Incf  | 0x01,1 |
| Comf  | 0x02,0 |
| Iorwf | 0x02,0 |
| Xorwf | 0x01,1 |
- 12.16) Zein da erregistro baten edukia erregistro berdinera mugitzeko instrukzioa erabiltzeko arrazoia? **Movf reg1,1**
- 12.17) Zenbat etendura-arrazoi daude PICetan?
- RB0/INT hankatxoaren aktibazioa
  - TMR0 temporizadoreak gainezka egitea
  - B atearen pisu handieneko lau hankatxoen (RB4:RB7) egoera-aldeketa
  - EEPROM datu-memorian idazketa-zikloaren amaiera
- 12.18) Zer egiten du hurrengo programak?
- |        |                 |            |
|--------|-----------------|------------|
| Berriz | Bcf             | intcon,gie |
|        | Btfsc           | intcon,gie |
|        | Goto            | berriz     |
|        | (jarraitzen du) |            |
- 12.19) Kalkula ezazu zenbatean behin sortuko duen etendura TMR0k  $40_{10}$  balioaz kargatzen badugu, zatigailua 1:8 barrutian programatua badago eta erabiltzen dugun kristala 1 MHz-ekoia bada.
- 12.20) W erregistroaren edukia 0xf0 eta EGOERA erregistroarena 0x0f bada, nola geratuko dira **andwf EGOERA,1** instrukzioa exekutatu ondoren?

## ***Programa ebatziak mihiztatzailean***

**12.1)** Sarrera bezala, A atearen bost lerroetan bost pultsadore ditugu, reset egiteko pultsadorea eta, irteera bezala, 5 LED diodo RB0tik eta RB4ra konektatuak. Diseina ezazu PICaren eskema elektronikoa, eta egin programa sarreran aktibatzen den pultsadoreari dagokion irteerako LED diodoa aktiba dadin

LIST	p=16c84	; PIC16C84 erabiltzen dugu
RADIX	hex	; sistema hamaseitarra erabiltzen da
W	EQU 0	; helmuga W denean, d = 0
F	EQU 1	; helmuga f erregistroa denean, d = 1
AATEA	EQU 0x05	; A atea 0 bankuko 5 helbidea dauka, ; eta konfigurazio-erregistroak 1 bankuaren 5 helbidea
BATEA	EQU 0x06	; gauza bera B atearentzat, 6 helbidean
EGOERA	EQU 0x03	; egoera-erregistroak 3 helbidea okupatzen du
	ORG 0	; programa 0 helbidean hasten da (reset bektorea)
	Goto hasi	; “hasi” etiketara jauzia
	ORG 5	; hurrengo instrukzioari 5 helbidea esleitzen zaio
hasi	bsf EGOERA,5	; egoera-erregistroko 5. bita 1ean ipintzen da. 1 bankurako atzipena
	clrfl BATEA	; B atearen lerroak irteera bezala konfiguratzentzira
	movlw 0xff	; W erregistroa 1ekin kargatzen da
	movwf AATEA	; A atearen lerroak sarrera bezala konfiguratzentzira
	bcf EGOERA,5	; egoera-erregistroko 5. bita 0an ipintzen da. 0 bankurako atzipena.
bira	movf ATEAA,W	; A atearen datuak W-n kargatzen dira
	comf W,0	; sarrerak “0”-z aktibatzen direnez, alderantzizkatu.
	movwf ATEAB	; W-ren edukia B atearen datu erregistroan kargatzen da.
	goto bira	; begizta infinitu bat sortzen da.
	END	; programaren bukaera



**12.2)** TMR0 temporizadorea erabiliz, RB0 irteeran keinukaria lortu. Milisegundoko denbora-tarteak osa daitezke era honetara.

```
***** DEFINIZIOAK *****
list    P=16c84
list    C=80
aatea  equ   05h
batea  equ   06h
egoera equ   03h
tm0    equ   01h
konta  equ   0ch

***** HASIERA *****
org    0           ; reset bektorea
goto  hasi         ; hasierara jauzia
org    5           ; hasierako helbidea

***** PROGRAMA NAGUSIA *****
hasi   bsf   egoera,5      ; 1 bankua hautatu
       movlw 0x1f          ; A atea sarrerak
       movwf aatea
       movlw 0               ; B atea irteerak
       movwf batea
       movlw 0x87          ; OPTION:TMR0 hautatu eta zatigailua 1/256
       movwf tm0
       bcf   egoera,5      ; 0 bankua hautatu
bat    bsf   batea,0        ; RB0 piztu
       call  egon          ; temporizazio-azpiprogramari deitu
       bcf   batea,0        ; RB0 itzali
       call  egon          ; itxaron
       goto bat            ; itzuli eta errepikatu

***** TENPORIZAZIO-AZPIPROGRAMA *****
egon   clrf  tm0          ;TMR0 ezabatu eta abiarazi
berriz btfss tm0,7        ; begiratu ea TMR0-ren azken bita piztuta dagoen,
                           ; ala bada, hurrengo instrukzioa exekutatu gabe pasatu.
                           ;jauzia egin "berriz"-era.
                           ;itzuli programa nagusira
                           ;programaren bukaera.
```

**12.3)** PIC16F84 mikrokontroladoreaz eta 4 MHz-ean funtzionatz, kontagailua programatu nahi da. KONTA deituriko kontagailua zeroan ipiniz hasiko da, gero banan-banan gehituko da 0x5Dh baliora heldu arte, ondoren ezer egiten ez duen begizta infinituan sartuz. Kontagailuaren balioa bitarrean adieraziko du B atera konektaturiko 8 LED diodoekin.

```
LIST P=16C84
RADIX HEX
```

```
W      EQU 0x00
F      EQU 0x01
BATEA EQU 0x06
EGOERA EQU 0x03
```

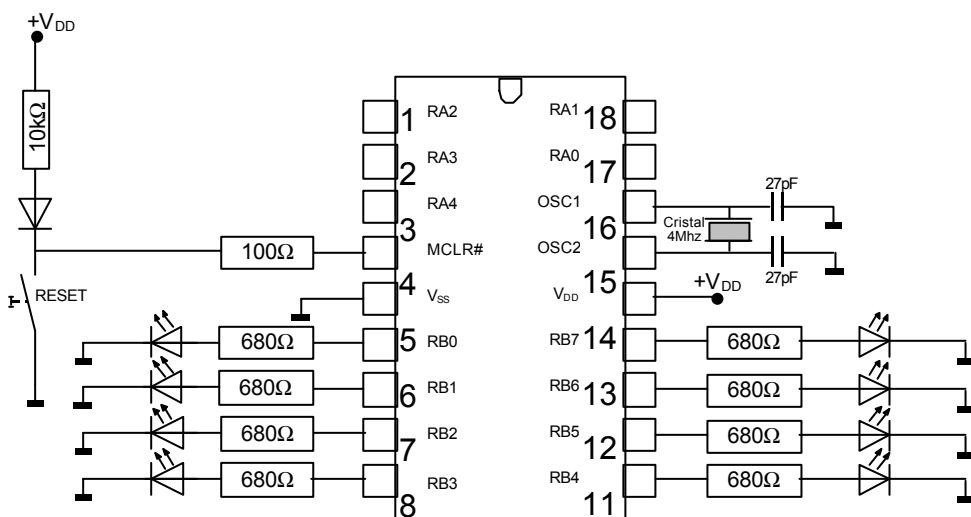
```

KONTA    EQU    0x0C
KONTA2   EQU    0x0D ;itxaronaldirako barne-kontagailua.

ORG      0       ; programa 0 helbidean hasten da eta
goto    inicio   ; 5 helbidera jauzi egiten du etendurak
ORG      5       ; bektorea gainetik pasatzeko.
inicio  bsf     EGOERA,5 ; 1 bankua hautatzen du
        movlw   0x00  ; B atea irteera bezala egituratzen da
        movwf   BATEA
        bcf     ESTADO,5 ; 0 bankua hautatu
        clrf    KONTA   ; KONTA = 0
bat     incf    KONTA,F  ; KONTA + 1 --> KONTA
        movf    KONTA,W ; W-n KONTA kargatzen da
        movwf   BATEA
        movlw   0xff
        movwf   KONTA2  ; KONTA2 hasieratzen da (0xff)
delay   decf    KONTA2  ; KONTA2 unitatean gutxitzen da
        nop
        nop
        nop
        nop
        movf    KONTA2,1 ; Z eguneratzen da
        btfss   EGOERA,2 ; Z aztertu eta 1 balio badu, "jauzia" egin
        goto    delay
        movlw   0x5d  ; W <- 0x5d
        subwf   KONTA,W ; KONTA - W --> W
        btfss   EGOERA,2 ; Z aztertu eta 1 balio badu, "jauzia" egin
        goto    bat
        bi      goto    bi
        goto    end
        bi

```

#### Eskema praktikoa



**12.4)** Keinukaria: TMR0 eta kontagailu bat erabiliz. Adibide honetan, denbora-tarte luzeak lortu nahi ditugu (xxx ms), horretarako kontagailu bat kargatuko dugu kopuru jakin batera, begizta bat hainbat aldiz errepikatuz nahi dugun denbora lortzeko. Bestalde, biraketa-instrukzioak ere erabiliko ditugu.

**Ezaugarri teknikoak:**

- A** ateko 3. bita aktibatua badago, **B** ate osoa piztu eta itzali ibiliko da keinuka, 0,5 segundoko kадentziaz.
- A** ateko 0. bita aktibatua badago, **B** ateko bit bakarra piztuko da aldiko eta ezkerrera biratuko du.
- A** ateko 0. bita desaktibatua badago, **B** atean pizturiko bit bakarrak eskuinera biratuko du.

	LIST	P=16C84	;PIC16F84 PROZESADOREA
	LIST	C=80	;LERROK0 80 KARAKTERE
aatea	EQU	0X05	;a-ren helbidea (0 atea 1 trisa)
batea	EQU	0X06	;b atea (0 atea 1 trisb)
egoera	EQU	0X03	;egoera hitzaren helbidea
tmr0_op	EQU	0X01	;TMR0-ren helbidea (0)
konta	EQU	0X0C	;kontagailua egiteko
	org	0	;reset bektorea
	goto	5	;4 helbidea libre utzi nahi dira, etendura bat gertatzen ;denean, programa kontagailuak helbide horretara jauzi ;egiten duelako.
	org	5	
	bsf	egoera,5	;1 bankua hautatu
	movlw	0x1f	;a atea denak sarrerak
	movwf	aatea	
	movlw	0x00	;b atea denak irteerak
	movwf	batea	
	movlw	0xd7	;zatigailua (:256), pull-up erresistentziarik ez
	movwf	tmr0_op	
	bcf	egoera,5	;0 bankua hautatu
	clrf	batea	;b atea itzali
	bsf	batea,3	;b ateko bat piztu
bat	btfsr	aatea,4	;ra4 = 0? ez bada, jauzi
	goto	keinu	
	btfsr	aatea,0	
	goto	ezker	
	call	egon	
	rrf	batea,1	;eskuinera biratu
	goto	bat	
ezker	call	egon	
	rif	batea,1	;eskerrera biratu
	goto	bat	
keinu	movlw	0xff	;denak pizteko
	movwf	batea	
	call	egon	;itxaron apur bat
	movlw	0x00	
	movwf	batea	;denak itzali
	call	egon	
	bsf	batea,3	;piztu 3. LEDa keinukaria kentzen dugunerako
(ra4)	goto	bat	
egon	movlw	d'15'	;kontagailuak 15era kargatu
	movwf	konta	;15 aldiz tmr0-k zenbatuko du
hiru	call	zenbat	
	clrf	tmr0_op	;tmr0 0-ra eta kontatzen hasi
	decf	konta,0	;(konta - 1) --> w
	movwf	konta	
	btfsr	egoera,2	;fz = 1?
	goto	hiru	

```

        return
zenbat btfss      tmr0_op,6    ;tmr0 = 64d?
        goto      zenbat
        return
end

```

**12.5) "KONTAGAILUA".** Adibide honetan kanpotik ezarritako pultsuak kontatu nahi ditugu eta display-an zenbatu (0tik 9ra)

```

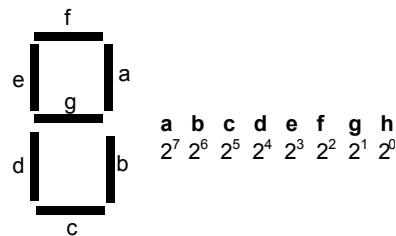
list      P=16c84
list      C=80
aatea    equ      05h
batea    equ      06h
egoera   equ      03h
tm0      equ      01h
PCL      equ      02h
konta   equ      0ch

org      0x00      ; reset bektorea
goto    HASI       ; hasierara jauzia
org      0x05      ;hasierako helbidea, 4a libre uzteko

HASI     bsf      egoera,5   ;1 bankua hautatu
        movlw   0x1f      ;A atea sarrerak
        movwf   aatea
        movlw   0x00      ;B atea irteerak
        movwf   batea
        movlw   0xA0      ;OPTION:T0CK1 pultsuak sartzeko
        movwf   tm0
        bcf      egoera,5   ;0 bankua hautatu
bi       clrf     tm0
        clrw
bat      movf      tm0,0
        movwf   konta
        call     display
        movwf   batea
        goto    bat

; ***** BIHURKETA-TAULA *****
display  addwf   PCL,1      ;pcl + w --> W
        retlw   0x3F      ;itzultzen da, 0 zenbakia dagokion kodeaz
        retlw   0x06      ;1aren kodea
        retlw   0x5b      ;2aren kodea
        retlw   0x4f      ;3aren kodea
        retlw   0x66      ;4aren kodea
        retlw   0x6d      ;5aren kodea
        retlw   0x7d      ;6ren kodea
        retlw   0x07      ;7aren kodea
        retlw   0x7f      ;8aren kodea
        retlw   0x5f      ;9aren kodea
        goto    bi
end

```

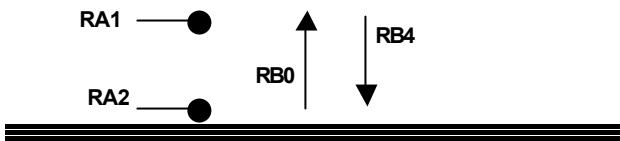


**12.6) GARAJEKO ATEAREN KONTROLA.** Urruneko aginteaz RA0 aktibatuz, RB0 aktibatu atea irekitzeko RA1 ukitu arte. Goian 30 segundo egon eta gero, RB4k atea itxiko du RA2 ukitu arte. Ktean RA0 aktibatzen bada, berehala RA2=0 eta RA0=1.

Atea zabaltzen ari den edo goian dagoen bitartean argi berdea piztuko da. Atea ixten ari den bitartean argi gorria. RB6=BERDE RB7=GORRI

Bost minutuko epean RB0 hamar aldiz aktibatzen bada RB0, atea 10 minutuz irekita egongo da.

List	p=16f84	
List	c=80	
Pa	equ 05h	RA1
Pb	equ 06h	RB0
Ego	equ 03h	RB4
Tm0	equ 01h	
Intcon	equ 0bh	
Kont	equ 0dh	
Konta	equ 0eh	
Kontad	equ 0fh	
	org 0h	
	goto hasi	
	org 0x05	
Hasi	bsf ego,5	
	clrf pb	;pb irteerak
	movlw 3fh	
	movwf pa	;pa sarrerak
	clrf intcon	
	bcf ego,5	
	clrf pb	
Ikus	btfss pa,0	;begiratu ea rb0 aktibatu den
	goto ikus	
Zabal	bcf pb,4	;piztu igotzekoa eta itzali jaistekoa
	bsf pb,0	
	bsf pb,6	;argi berdea piztu eta gorria itzali
	bcf pb,7	
Ber	btfss pa,1	;begiratu goraino heldu den
	goto ber	
	bcf pb,0	;gora heldu eta geratu (30 segundo)
	movlw 0x9f	
	movwf kont	
Hiru	movlw 0xff	
	movwf konta	;hiru bloke hauetan lortzen dugu
Bi	movlw 0xff	;
	movwf kontad	t segundo
Bat	decfsz kontad,1	
	goto bat	; t1 = (3x255)+2 = 767 ms
	decfsz konta,1	; t2 = [(767+3)x255]+2 = 196.352 ms
	goto bi	; t3 = [(196352+3)x159]+2 = 30.584.447 ms
	decfsz kont,1	; t3 = 30'58 s
	goto hiru	
	bsf pb,4	; piztu jaisteko eta itzali igotzeko
	bcf pb,6	
	bsf pb,7	;argi gorria piztu eta berdea itzali
Bis	btfsc pa,0	
	goto zabal	;aztertu behera heldu den eta bai



```
btfss  pa,2           ;rb0 aktibatu den (sartzeko eskaera)
goto   bis
bcf    pb,4
bcf    pb,7
goto   ikus
end
```

## **12.18. CCS ETXEKO PCM C KONPILADOREA**

PCM izena duen CCS (*Custom Computer Services*) etxeko konpiladore honek arrakasta lortu du nahiko merkea delako eta, aldi berean, prestazio onak ematen dituelako. Hauexek dira konpiladore honek onartzen dituen PICak:

**PCM:** PIC12C671, PIC12C672, PIC14000, PIC16C554, PIC16C556, PIC16C558, PIC16C61, PIC16C62, PIC16C620, PIC16C621, PIC16C622, PIC16C62A, PIC16C63, PIC16C63EM, PIC16C64, PIC16C641, PIC16C642, PIC16C64A, PIC16C65, PIC16C65A, PIC16C66, PIC16C661, PIC16C662, PIC16C67, PIC16C70, PIC16C71, PIC16C710, PIC16C711, PIC16C715, PIC16C71A, PIC16C72, PIC16C73, PIC16C73A, PIC16C73EM, PIC16C74, PIC16C74A, PIC16C76, PIC16C77, PIC16C83, PIC16C84, PIC16C84A, PIC16C923, PIC16C924, PIC16CR84, PIC16F83, PIC16F84

PIC bakoitzeko fitxategi bat dago (.h) beren definizioekin. Programa bat egitean, fitxategi hori sartu behar dugu #include komandoaren bidez: **#include <16C84.h>**.

Komeni da .h fitxategi horietan zer definizio dauden jakitea, horiek programazioa errazten baitute. Hona hemen adibide gisa 16C84.h fitxategian dagoena:

```

///////// Standard Header file for the PIC16C84 device ///////
#define PIC16C84
#nolist
//////////////////////////// I/O definitions for INPUT() and OUTPUT_xxx()
#define PIN_A0 40
#define PIN_A1 41
#define PIN_A2 42
#define PIN_A3 43
#define PIN_A4 44

#define PIN_B0 48
#define PIN_B1 49
#define PIN_B2 50
#define PIN_B3 51
#define PIN_B4 52
#define PIN_B5 53
#define PIN_B6 54
#define PIN_B7 55

//////////////////////////// Useful defines
#define FALSE 0
#define TRUE 1

#define BYTE int
#define BOOLEAN short int

#define getc getch
#define getchar getch
#define puts(s) {printf(s); putchar(13); putchar(10);}
#define putc putchar

```

```
/////////////////// Constants used for RESTART_CAUSE()
#define WDT_FROM_SLEEP 0
#define WDT_TIMEOUT 8
#define MCLR_FROM_SLEEP 16
#define NORMAL_POWER_UP 24
/////////////////// Constants used for SETUP_COUNTERS()
#define RTCC_INTERNAL 0
#define RTCC_EXT_L_TO_H 32
#define RTCC_EXT_H_TO_L 48
#define RTCC_DIV_2 0
#define RTCC_DIV_4 1
#define RTCC_DIV_8 2
#define RTCC_DIV_16 3
#define RTCC_DIV_32 4
#define RTCC_DIV_64 5
#define RTCC_DIV_128 6
#define RTCC_DIV_256 7
#define WDT_18MS 8
#define WDT_36MS 9
#define WDT_72MS 10
#define WDT_144MS 11
#define WDT_288MS 12
#define WDT_576MS 13
#define WDT_1152MS 14
#define WDT_2304MS 15
#define L_TO_H 0x40
#define H_TO_L 0

#define RTCC_ZERO 0x0B20 // Used for ENABLE/DISABLE INTERRUPTS
#define INT_RTCC 0x0B20 // Used for ENABLE/DISABLE INTERRUPTS
#define RB_CHANGE 0x0B08 // Used for ENABLE/DISABLE INTERRUPTS
#define INT_RB 0x0B08 // Used for ENABLE/DISABLE INTERRUPTS
#define EXT_INT 0x0B10 // Used for ENABLE/DISABLE INTERRUPTS
#define INT_EXT 0x0B10 // Used for ENABLE/DISABLE INTERRUPTS

#define GLOBAL 0x0B80 // Used for ENABLE/DISABLE INTERRUPTS
#define _GLOBAL 0x0B80 // Used for ENABLE/DISABLE INTERRUPTS
#define INT_EEPROM 0x0B40 // Used for ENABLE/DISABLE INTERRUPTS

#define list
```

Konpiladore hau oso erraza da erabiltzeko, konpiladore estandarren antza baitu. Irudian ikus dezakegu:

```

MS PICC
File Edit Compile Info Program Window Options
[*] D:\ELEKTRON\PICS\PICC\EXAMPLES\EX_LED.C
#include <16c56.h>
#fuses xt,nowdt,noprotect

#use delay(clock=20000000)
#use rs232(baud=1200,xmit=PIN_A3,recv=PIN_A2)

#byte port_b=6

byte CONST LED_MAP[10] = {0x90,0xb7,0x19,0x15,0x36,0x54,0x50,0xb5,0,0x24};

void wait() { // This function waits for either ~2ms or until a
    int countdown; // event happens (in this case a rs232 character)
    countdown=200;
    while((--countdown!=0)&&!kbhit())
        delay_us(10);
}

void display_segs(char c) {
    if((c>'9')||(c<'0'))
        1:1

```

Ikus ditzagun ondoren konpiladore honen berezitasunak: aurreprozesadorearen komandoak, adierazpenak eta funtzio bereziak. Ez ditugu guztiak aipatuko, garrantzitsuenak baizik.

### 12.18.1. AURREPROZESADOREAREN KOMANDOAK

Aurreprozesadorearen komando guztiak # ikurrarekin hasiko dira. Batzuk C konpiladore estandarrenak bezalakoak dira, besteak beste:

#DEFINE	#UNDEF	#IF	#INCLUDE <fitxategia>
#IFDEF id	#IFNDEF id	#ELSE	#INCLUDE "fitxategia"
#ENDIF	#LIST	#NOLIST	

---

#### #DEVICE chip

Komando honekin adierazten da zein PICekin lan egin nahi dugun. Kontuan hartu behar da .h fitxategietan, lehendabiziko lerroan, komando hau sartuta dagoela, beraz, #include komandoarekin horrelako fitxategi bat txertatuz gero, programan ez da #device komandoa sartu behar.

#### #device PIC16F84

---

## #FUSES aukerak

Komando honekin hardwareen aukerak zehazten dira. Batzuk programadorean bertan defini daitezke. Honako hauek onartzen ditu:

LP, XT, HS, RC  
WDT, NOWDT  
PROTECT, NOPROTECT, PROTECT\_50%, PROTECT\_75%  
PUT, NOPUT  
BROWNOUT, NOBROWNOUT

Lehenetsitako balioak hauexek dira: #fuses RC,NOWDT,PROTECT,PUT  
Adibidea: #fuses HS,WDT

---

#INT_EXT	KANPOKO ETENDURA
#INT_RTCC	TIMER0 (RTCC) OVERFLOW
#INT_RB	B7..B4 PIN BATEAN ALDAKETA
#INT_AD	A/D BIHURGAILUA
#INT_EPROM	EEPROM-EAN IDAZKETA BUKATUTA
#INT_TIMER1	TIMER1 OVERFLOW
#INT_TIMER2	TIMER2 OVERFLOW
#INT_CCP1	CCP1 CAPTURE
#INT_CCP2	CCP2 CAPTURE
#INT_SSP	I <sup>2</sup> C SERIAL PORT
#INT_TBE	SERIAL DATA TRANSMITTED
#INT_RDA	SERIAL DATA RECEIVED
#INT_COMP	COMPARATOR INTERRUPT

Komando hauek funtzi baten aurretik jartzen dira, eta funtzi hori etendura bateko funtzioa dela adierazten dute eta, hain zuzen, zein etendurarekin loturik dagoen. Horretarako, etendurak gaituta egon behar du, baina hori funtzi batekin egingo dugu, `enable_interrupt(level)` funtzioarekin, hain zuzen ere.  
Adibidea:

```
#int_rtcc
timer_etendura()
{
    kont++;
}
```

---

## #PRIORITY

Komando honek etenduren arteko lehentasuna esleitzeko balio du. Lehenak bigarrenak baino lehentasun handiagoa izango du eta honek hirugarrenak baino handiagoa, etab. Adibidea:

```
#priority rtcc, rb, ext
```

---

### #USE DELAY(CLOCK=maitzasuna)

Komando honekin, konpiladoreari esaten diogu zein maitzasunarekin lan egingo duen gure mikroak. Hori beharrezko da `DELAY_US(time)` eta `DELAY_MS(time)` funtziok ondo funtziona dezaten. Adibidea:

```
#use delay(clock=4000000)
```

---

### #USE RS232(BAUD=baud,XMIT=pin,RCV=pin)

Komando honekin, **nahiz eta txipa serieko linea ez izan**, bi I/O hankatxoak serieko ataka bat balira bezala funtziona dezatela egiten da. Komando hau jarriz gero, GETCH, PUTCHAR eta PRINTF funtziok erabili daitezke. Gogoratu maila hauek TTL-koak direla; beraz, MAX232 txipa beharko dugu. BAUD, XMIT eta RCV-k abiadura, datuak bidaltzeko pina eta datuak jasotzeko pina zehazten dute, hurrenez hurre. Paritatea ere zehatz daiteke PARITY=x aldagaiairekin x N,E edo O izanda. Adibidea:

```
#use rs232(baud=9600,xmit=PIN_A2,rcv=PIN_A3)
```

---

### #USE I2C(MASTER, SDA=pin, SCL=pin)

Gauza bera I<sup>2</sup>C busa softwareen bidez izateko. Aldagai gehiago onartzen ditu: slave, address=nn, fast, slow. Adibidea:

```
#use I2C(slave, sda=PIN_C4, scl=PIN_C3, address=0x0a)
```

---

### #BYTE id=x

Komando honekin id hitzari x memoria-helbidean dagoen objektua esleitzen diogu. Horrela, programa ulergarriagoa lortuko dugu. Adibidea:

```
#byte b_atea=06
```

#### 12.18.2. DATU-DEFINIZIOA

Aldagaiak STATIC (globala) edo AUTO (bertakoa) defini daitezke. AUTO ez da erabiltzen lehenetsitakoa delako. Hauek dira deklara daitezkeen datu-motak:

UNSIGNED	8 biteko zeinurik gabeko zenbakia.
UNSIGNED INT	8 biteko zeinurik gabeko zenbakia.
INT	8 biteko zeinurik gabeko zenbakia.
CHAR	8 biteko zeinurik gabeko zenbakia.
LONG	16 biteko zeinurik gabeko zenbakia.
LONG INT	16 biteko zeinurik gabeko zenbakia.
SHORT	Bit bat.
SHORT INT	Bit bat.

Enum, struct eta union.

Datuak hauexek izan daitezke:

123	hamartarra
0x123	hamaseitarra
0b010111	bitarra
'x'	karakterea
'\x'	karaktere berezia. x izan daiteke: n,t,b,r,f,',\'
"abcde"	katea.

### 12.18.3. KONPILADOREAREN FUNTZIOAK

Konpiladore hauekin funtzi hauek erabil daitezke:

---

#### DELAY\_US(time)

Funtzio honek *time* mikrosegundoko atzerapena sortzen du. *Time* konstantea bada, 0-65025 tartean egon daiteke; aldagaia bada, berriz, 0-255 tartean. Ondo funtzionatu ahal izateko #use delay komandoa jarri behar da. Adibidez:

```
#use delay(clock=4000000)
delay_us(100);           /* 100 mikrosegundoko atzerapena */
```

---

#### DELAY\_MS(time)

Funtzio honek *time* milisegundoko atzerapena sortzen du. *Time* konstantea bada, 0-65025 tartean egon daiteke; aldagaia bada, berriz, 0-255 tartean. Ondo funtzionatu ahal izateko #use delay komandoa jarri behar da. Adibidez:

```
#use delay(clock=4000000)
delay_ms(100);           /* 100 milisegundoko atzerapena */
```

---

#### DELAY\_CYCLES(zenbaki)

Funtzio honek *zenbaki* makina-zikloaren atzerapena sortzen du. Gogoratu makina-ziklo batean instrukzio bat egikaritzen dela eta makina-ziklo batek lau erloju-ziklo dituela. Kasu honetan #use delay komandoa ez da beharrezkoa. Adibidez:

```
delay_cycles(1);          // NOP egikaritzea bezalakoa da.
```

---

## SLEEP()

Funtzio honek *sleep* instrukzioa egikaritzen du eta mikroa atseden-egoeran uzten du (kontsumo txikiko egoeran). Adibidez:

```
sleep();
```

---

## RESTART\_WDT()

Funtzio honekin “zakur zaindaria” berrabiarazten dugu. Noizean behin egikaritu beharko da (“zakur zaindariak” gainezka egin baino lehen). Adibidez:

```
While(1)
{
    restart_wdt();
    .....
    .....
}
```

---

## OUTPUT\_LOW(pin)

Irteerako pin batean “0” jartzen du. Adibidez:

```
output_low(PIN_A0); // RA0 = 0
```

---

## OUTPUT\_HIGH(pin)

Irteerako pin batean “1” jartzen du. Adibidez:

```
output_high(PIN_A2); // RA2 = 1
```

---

## OUTPUT\_BIT(pin,balioa)

Funtzio honekin pina zehazten dugu eta pin horrek hartuko duen irteerako balioa (0 edo 1). Adibidez:

```
output_bit(PIN_A2,0); // Hau output_low(PIN_A2) bezalakoa da.
output_bit(PIN_A0, input(PIN_B1)); /* RB1 pinetik sartzen dena
RA0-tik aterako da*/
```

---

## INPUT(pin)

Funtzio honekin pin bat irakurtzen dugu. SHORT INT mota itzultzen du.

Adibidez:

```
a=input(PIN_B0); // RB0 pinak duen balioa a aldagaien gordetzen da.
```

**Oharra:** pinen izenak .h fitxategian zehazten dira. Bertan, nahi izanez gero, alda daitezke.

---

**SET\_TRIS\_A(BALIOA), SET\_TRIS\_B(BALIOA), SET\_TRIS\_C(BALIOA),  
SET\_TRIS\_D(BALIOA), SET\_TRIS\_E(BALIOA),**

Funtzio hauek ataketako pinak irteera edo sarrera definitzeko balio dute. 1-ak pina sarrerakoa dela adierazten du eta 0-ak irteerakoa dela. Adibidez:

```
SET_TRIS_B(0X0F); // RB0, RB1,RB2 eta RB3 sarrera izango dira,  
// eta RB4, RB5, RB6 eta RB7 irteera.
```

---

### **GETCH(), GETC(), GETCHAR()**

Funtzio hauek RS232 lineako RCV pinetik karaktere bat jaso arte itxaroten dute, eta jaso ondoren karaktere hori bera itzultzen dute. Aurrez, #USE RS232 definitu behar da eta pin hori beste RS232 linea batekin (ordenagailuaren serieko linea batekin, adibidez) lotu behar da, baina MAX232 txip baten bidez tentsio-mailak egokitzea. Adibidea:

```
printf("Jarraitu: (Y/N)?");  
do{  
    erantzuna=getch();  
}while(erantzuna!='Y' && erantzuna!='N');
```

---

### **PUTCHAR(char), PUTCH(char)**

Funtzio hauek karaktere bat bidaltzen dute RS232 lineako XMIT pinetik. Aurrez, #USE RS232 definitu behar da eta pin hori beste RS232 linea batekin (ordenagailuaren serieko linea, adibidez) lotu behar da, baina MAX232 txip baten bidez tentsio-mailak egokitzea. Adibidea:

```
if(cheksum==0)  
    putchar(ACK);  
else  
    putchar(NACK);
```

---

### GETS(char \*string)

Funtzio honek, GETC() erabiliz, kate bat hartzen du eta string erakuslean gordetzen du. Katea bukatutzat ematen da 0 bat aurkitzen denean. Aurrez, #USE RS232 definitu behar da. Adibidea:

```
char izena[20];
gets(izena);
```

---

### PUTS(string)

Funtzio honek, PUTC() erabiliz, kate bat bidaltzen du. Aurrez, #USE RS232 definitu behar da. Adibidea:

```
char izena[20]={"Joseba"};
puts("-----");
puts(izena);
puts("-----");
```

---

### PRINTF(function,string,...)

Funtzio honek, PUTC() erabiliz, kate bat formatuarekin bidaltzen du. Aurrez, #USE RS232 definitu behar da. Honako hauek onartzen ditu:

%c	Karakterea	%d	signed int
%u	Unsigned int	%lx	long int hamaseitarrean (xehez)
%x	Int hamaseitarrean (letra xehez)	%IX	long int hamaseitarrean (larriz)
%X	Int hamaseitarrean (letra larriz)	%%	% ikurra.

Formatuen adibideak:

Formatua	Balioa=0x12	Balioa=0xfe
% 03u	018	254
% u	18	254
% 2u	18	254
% 5u	18	254
% d	18	-2
% x	12	fe
% X	12	FE
% 4X	0012	00FE

Adibideak:

```
byte x,y,z;  
printf("Kaixo lagunak");  
printf("Timer0-aren balioa= %2X", get_rtcc());  
printf("Neurtutako balioak: %2u, %X, %4x", x,y,z);  
  
char izena[20]="Joseba";  
puts("-----");  
puts(izena);  
puts("-----");
```

---

### I2C\_START(), I2C\_STOP(), I2C\_WRITE(byte), I2C\_READ(), I2C\_POLL()

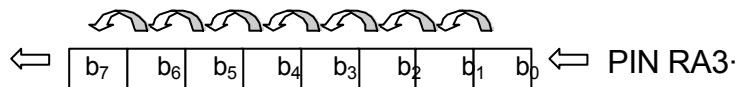
I2C busarekin lan egiteko funtzioak.

---

### SHIFT\_LEFT(address,byte,balioa)

Bitak ezkerrera desplazatzen dira, eta bytearen posizioa eta balioa eskuinetik sartzen dira. Ateratzen den bita itzultzen du. Adibidez:

```
shift_left(&b,1,RA3)
```

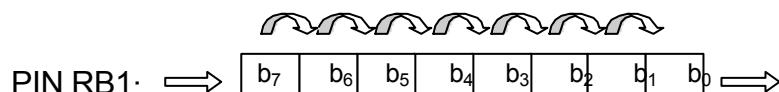


---

### SHIFT\_RIGHT(address,byte,balioa)

Bitak eskuinera desplazatzen dira, eta bytearen posizioa eta balioa ezkerretik sartzen dira. Ateratzen den bita itzultzen du. Adibidez:

```
shift_right(&b,1,input(RB1))
```



Beste adibide bat datu bat pin batetik seriean ateratzeko, LSB bitetik hasita:

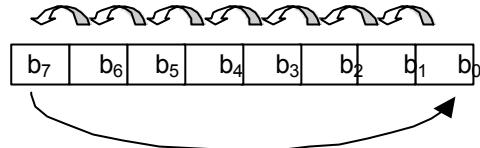
```
for(i=0;i<8;++i)  
    output_bit(PIN_A0, shift_right(&datua,1,0));
```

---

**ROTATE\_LEFT(address,byte)**

Bitak ezkerrera desplazatzen dira, eta bytearen posizioa eta b<sub>7</sub> bita b<sub>0</sub>-ra pasatzen dira. Adibidez:

```
rotate_left(&b,1)
```



Beste adibide bat:

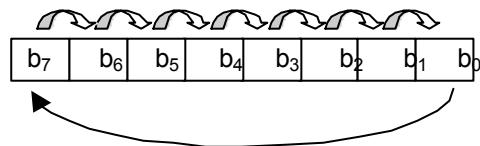
```
byte x= 0x86;           // bitarrean: 1000 0110
rotate_left(&x,1);      // balio berria: 0000 1101 == 0xd
```

---

**ROTATE\_RIGHT(address,byte)**

Bitak eskuinera desplazatzen dira, eta bytearen posizioa eta b<sub>0</sub> bita b<sub>7</sub>-ra pasatzen dira. Adibidez:

```
rotate_right(&b,1)
```



Beste adibide bat:

```
byte x= 0x86;           // bitarrean: 1000 0110
rotate_right(&x,2);     // balio berria: 1010 0001 == 0xc1
```

---

**BIT\_SET(byte,bit)**

Bytetako bita batean jartzen du. Adibidea:

```
Int x=5;      // x= 0000 0101
Bit_set(x,3) // x= 0000 1101
```

Bytea ataka bateko helbidea bada, ataka horretako bit batetik “1” atera dezakegu.

---

### **BIT\_CLEAR(byte,bit)**

Bytetako bita zeroan jartzen du. Adibidea:

```
int x=5;      // x= 0000 0101
bit_set(x,2); // x= 0000 0001
```

Bytea ataka bateko helbidea bada, ataka horretako bit batetik “0” atera dezakegu.

---

### **BIT\_TEST(byte,bit)**

Funtzio honek TRUE itzultzen du byteko bita “1” bada, FALSE itzultzen du “0” bada. Adibidea:

```
if(bit_test(x,3) || !bit_test(x,1))
// x3 bita bat bada edo x1 bita zero bada...
```

---

### **SWAP(byte)**

Byte horretako NIBLEak elkar trukatzen dira. Adibidea:

```
int x=0x45;
swap(x);      // x=0x54
```

---

### **MEMCPY(dest,source,n)**

RAM memorian *n* helbide kopiatzen ditu *source*-tik *dest*-era. *Source*-k eta *dest*-ek erakusleak izan behar dute nahitaez. Adibidea:

```
byte *p1,*p2;
p1=0x12;
p2=0x18;
memcpy(p1,p2,3);
```

---

### **MEMSET(dest,value,n)**

RAM memorian *n* helbide *value* balioarekin ipintzen ditu, *dest* helbidetik hasita. *Dest*-ek erakuslea izan behar du nahitaez. Adibidea:

```
byte *p1;
p1=0x12;
memset(p1,0xff,5);      // 12,13,14,15,16 posizioetan 0xff idazten du.
```

---

### STRCPY(dest,source)

RAM memorian kate bat kopiatzen du. Adibidea:

```
char katea[10];
strcpy(katea,"Kaixo");
```

---

### CTYPE.H

Hemen funtziostea estandarrak daude:

TRUE itzultzen dute baldin eta:

ISALNUM(x)	x bada 0..9, 'A'..'Z', edo 'a'..'z'.
ISALPHA(x)	x bada 'A'..'Z', edo 'a'..'z'.
ISDIGIT(x)	x bada '0'..'9'.
ISLOWER(x)	x bada 'a'..'z'.
ISUPPER(x)	x bada 'A'..'Z'.
ISSPACE(x)	x bada tarteak.
ISXDIGIT(x)	x bada '0'..'9', 'A'..'Z', edo 'a'..'z'.
TOLOWER(x)	x bada 'A'..'Z', itzultzen du 'a'..'z'.
TOUPPER(x)	x bada 'a'..'z' , itzultzen du 'A'..'Z'.

---

### SETUP\_COUNTERS(rtcc\_state,ps\_state)

Funtzioteari TIMER0 eta WDT programatu eta abian jartzen ditugu. **Rtcc\_state** komandoarekin kontatzeko pultsuen iturria zehazten dugu: barruko erlojua edo T0CK1 hankatxotik sartzen diren pultsuak. **Ps\_state** parametroarekin maiztasun zatitzalea programatzen da. Hauek dira bi parametro horiek har ditzaketen balioak:

rtcc\_state balioak:

RTCC_INTERNAL	
RTCC_EXT_L_TO_H	// goranzko fronteak
RTCC_EXT_H_TO_L	// beheranzko fronteak

ps\_state balioak:

RTCC_DIV_2	
RTCC_DIV_4	
RTCC_DIV_8	
RTCC_DIV_16	
RTCC_DIV_32	
RTCC_DIV_64	
RTCC_DIV_128	
RTCC_DIV_256	
WDT_18MS	

WDT\_36MS  
WDT\_72MS  
WDT\_144MS  
WDT\_288MS  
WDT\_576MS  
WDT\_1152MS  
WDT\_2304MS

Adibidea:

```
setup_counters(RTCC_INTERNAL,WDT_2304MS); // WDT 2,304 segundotik  
// behin  
//gainezka egingo du
```

---

**GET\_RTCC()**  
**GET\_TIMER0()**  
**GET\_TIMER1()**  
**GET\_TIMER2()**

Funtzio honek, une horretan bertan, *timer*-ak duen balioa itzultzen du. RTCC eta TIMER0 gauza bera dira. TIMER1 16 bitekoa da, besteak 8 bitekoak.

Adibidea:

```
while(get_timer0() != 0)
```

---

**SET\_RTCC(value)**  
**SET\_TIMER0(value)**  
**SET\_TIMER1(value)**  
**SET\_TIMER2(value)**

Funtzio honek *timer*-etan balio bat ipintzeako balio du. *Timer*-ek gorantz kontatzen dutenean gomendatzen da balioak biko osagarrian adieraztea; hau da, 4 pultsu kontatzea nahi badugu —4 biko osagarrian (0xFC), edo 0x100-4 (0xFC), jarriko dugu. Adibidea:

```
Set_rtcc(0xF0); // 16 pultsu kontatzeko.
```

---

**SETUP\_TIMER\_1(mode)**

Funtzio honek TIMER1 programatzen du. *Mode*-k onartzen dituen balioak hauek dira:

T1\_DISABLED  
T1\_INTERNAL  
T1\_EXTERNAL

T1\_CLK\_OUT  
T1\_DIV\_BY\_1  
T1\_DIV\_BY\_2  
T1\_DIV\_BY\_4  
T1\_DIV\_BY\_8

Adibidea:

```
setup_timer_1(T1_INTERNAL | T1_DIV_BY_8);
```

---

### **SETUP\_TIMER\_2(mode)**

*Timer* hau mikro gutxi batzuek daukate. PWM moduluarekin lan egiteko diseinatuta dago. Informazio gehiago konpiladorearen eskuliburuan.

---

### **SETUP\_CCP1(mode)**

### **SETUP\_CCP2(mode)**

*Capture/comparator/PWM* moduluak programatzeko. Informazio gehiago konpiladorearen eskuliburuan.

---

### **SET\_PWM1\_DUTY(value)**

### **SET\_PWM2\_DUTY(value)**

PWM moduluak programatzeko. Informazio gehiago konpiladorearen eskuliburuan.

---

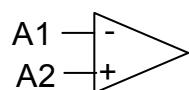
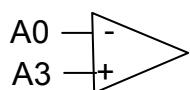
### **SETUP\_COMPARATOR(mode)**

Konparatzeko bi moduluak programatzeko. Hauek dira balioak:

A0\_A3\_A1\_A2  
NC\_NC\_A1\_A2  
NC\_NC\_NC\_NC  
A0\_VR\_A2\_VR  
A3\_VR\_A2\_VR  
A3\_A2\_A1\_A2

Lau balio hauek adierazten dute: C1-, C1+, C2-, C2+. Adibidea:

```
setup_comparator (A0_A3_A1_A2);
```



---

**SETUP\_VREF(mode | value)**

Konparatzeko barruko tentsio-erreferentzia (VR) programatzeko. Hiru balio hauek onartzen ditu:

FALSE	→	off, desaktibatuta.
REF_LOW	→	$\frac{V_{cc} \times \text{balioa}}{24}$
VREF_HIGH	→	$\frac{V_{cc} \times \text{balioa}}{32} + \frac{V_{cc}}{4}$

Adibidea:

```
Setup_vref(VREF_HIGH | 6); // VKZ =5V bada, orduan VO=2,1875
```

---

**value = READ\_EEPROM(helbidea)**

Honekin EEPROM memoriaren helbide bat irakur dezakegu.

---

**WRITE\_EEPROM(helbidea, balioa)**

Honekin EEPROM memoriaren helbide batean datu bat idatz dezakegu.

---

**SETUP\_PORT\_A(mode)**

Funtzio honekin A atakaren hankatxoak digitalak edo analogikoak diren zehazten dugu. Hauek dira mode parametroak har ditzakeen balioak:

NO_ANALOGS	guztiak digitalak
ALL_ANALOG	guztiak analogikoak. 5V erreferentzia.
ANALOG_RA3_REF	0,1,2 analogikoak. 3 erreferentzia
RA0_RA1_ANALOG	0,1 analogikoak, 2,3 digitalak (74-an izan ezik)

---

**SETUP\_ADC(mode)**

Bihurgailu analogiko/digitala programatzeko. Balio hauekin:

```
ADC_OFF  
ADC_CLOCK_DIV_2  
ADC_CLOCK_DIV_8  
ADC_CLOCK_DIV_32  
ADC_CLOCK_INTERNAL
```

---

### SET\_ADC\_CHANNEL(chan)

Bihurgailu analogiko/digitalaren zein kanal erabiliko den hurrengo deian programatzeko. Adibidea:

```
set_adc_channel(2);
```

---

### READ\_ADC()

Bihurgailuaren balioa irakurtzeko. Adibidea:

```
balioa = read_adc;
```

---

### ENABLE\_INTERRUPTS(level) DISABLE\_INTERRUPTS(level)

Etendurak gitzeko edo desgitzeko, hurrenez hurren. Level parametroak balio hauek har ditzake:

GLOBAL	ADC_DONE	RTCC_ZERO
RB_CHANGE	EXT_INT	INT_TIMER1
INT_TIMER2	INT_CCP1	INT_CCP2
INT_SSP	INT_PSP	INT_RDA
INT_TBE		

Adibidea:

```
disable_interrupts(global);           // etendura guztiak desgaituta.  
enable_interruptsadc_done);  
enable_interrupts(rb_change);         // bi hauek funtziona dezakete, baina,  
                                      // globalki, guztiak gaitzen baditugu.  
enable_interrupts(global);
```

---

### PORT\_B\_PULLUPS(flag)

B ataka sarrera denean, barruko *pull-up* erresistentziak aktibatzeko. TRUE aktibatzeko eta FALSE desaktibatzeko. Adibidea:

```
port_b_pullups(TRUE);
```

---

### EXT\_INT\_EDGE(edge)

Kanpoko etenduraren fronte aktiboa zein izango den programatzeko. Edge parametroaren balioak: L\_TO\_H edo H\_TO\_L. Adibidea:

```
Ext_int_edge(H_TO_L);           // beheranzko frontea.
```

---

## Ariketa ebatziak C lengoain

### 1. ADIBIDEA

Mihiztatzailean egin dugun lehendabiziko adibidea orain C lengoain egingo dugu. Hau da, A atetik sartzen dugun informazioa B atetik aterako dugu.

```
#include <16C84.H>
#fuses XT,NOWDT,NOPROTECT
#use delay(clock=4000000)
#byte port_a=5
#byte port_b=6

main()
{
    set_tris_a(0xff);           // A atakaren hankatxo guztiak sarrera.
    set_tris_b(0x00);           // B atakaren hankatxo guztiak irteera.
    while (TRUE)
        port_b = port_a;
}
```

### 2. ADIBIDEA (UHIN KARRATUA)

Programa honek mezu bat bidaltzen du pantailara eta itxaron egiten du tekla bat sakatu arte. Tekla bat sakatzean, RB0 hankatxotik 1kHz-eko uhin karratu bat ateratzen du.

```
#include <16C84.H>
#fuses XT,NOWDT,NOPROTECT
#use delay(clock=4000000)
#use rs232(baud=9600, xmit=PIN_A3, rcv=PIN_A2)

main()
{
    printf("Tekla bat sakatu sorgailua abian jartzeko\n\r");
    getc();
    printf("1 kHz-eko seinalea aktibatuta\n\r");
    while (TRUE)
    {
        output_high(PIN_B0);
        delay_us(500);
        output_low(PIN_B0);
        delay_us(500);
    }
}
```

**Oharra:** ataketako pin batekin lan egiteko, INPUT, OUTPUT\_HIGH edo OUTPUT\_LOW funtzioak erabiltzen dira, besterik gabe. Ataka osoa erabiltzeko, berriz, hankatxoak definitu behar dira sarrera edo irteera bezala SET\_TRIS\_X funtzioarekin eta atakaren helbidearekin lan egin behar da lehendabiziko adibidean ikusten den moduan.

### 3. ADIBIDEA (TIMER0 DENBORA-NEURGAILUA)

Programa honek TIMER0-a etendurekin erabiltzen du denbora-tarte bat neurtzeko. Kontagailua martxan jartzen da tekla bat sakatzean, eta gelditu egiten da beste tekla bat sakatzean. Iragan den denbora pantailan ikusiko da segundotan adierazita.

```
#include <16C84.H>
#fuses XT,NOWDT,NOPROTECT
#use delay(clock=4000000)
#use rs232(baud=9600, xmit=PIN_A3, rcv=PIN_A2)

#define ETEND_SEGUNDOKO 15 // (4000000/(4*256*256))

byte etend_kont; // etenduren kopurua zenbatzeko
byte segundoak; // segundoak zenbatzeko

#int_rtcc // hurrengo funtzioa TIMER0-aren etendurari dagokio.
rtcc_gainezkatuta() // (255->0). Programa horretan 15 aldiz segundoko gutxi gorabehera
{
    if(--etend_kont==0)
    {
        ++segundoak;
        etend_kont=ETEND_SEGUNDOKO;
    }
}

main()
{
    byte start;
    etend_kont=ETEND_SEGUNDOKO;
    set_rtcc(0);
    setup_counters(RTCC_INTERNAL, RTCC_DIV_256);
    enable_interrupts(RTCC_ZERO);
    enable_interrupts(GLOBAL);
    do {
        printf("Sakatu tekla bat hasteko.\n\r");
        getc();
        start=segundoak;
        printf("Sakatu beste tekla bat gelditzeko.\n\r");
        getc();
        printf("%u segundoak.\n\r",segundoak-start);
    } while (TRUE);
}
```

#### 4. ADIBIDEA (A/D BIHURGAILUA)

Programa honek sarrera analogikoa irakurtzen du eta dagokion balio digitala pantailaratzen du.

```
#include <16C73.H>
#fuses XT,NOPROTECT,NOWDT
#use delay(clock=4000000)
#use rs232(baud=2400,xmit=PIN_C6,recv=PIN_C7)

main()
{
    int i,value,min,max;

    setup_port_a(ALL_ANALOG);
    setup_adc(ADC_CLOCK_INTERNAL);
    set_adc_channel(0);

    while(TRUE)
    {
        delay_ms(100);           // A/D bihurgailuari denbora emateko.
        printf("\rIrakurritako balioa: %2X ", Read_ADC());
    }
}
```

#### 5. ADIBIDEA (LCD PROGRAMATU)

Programa honetan LCDa kontrolatzeko behar diren oinarrizko funtziok egin ditugu. Hau oinarria izan daiteke beste funtzio batzuk ateratzeko eta programa osatzeko. Konexioak hauek direla kontuan hartu behar da:

```
RB0-7 --> D0-7
RA0 --> R/S
RA1 --> R/W
RA2 --> E
```

```
#include <16c84.h>
#include <stdio.h>
#fuses xt,nowdt,noprotect

#use delay(clock=4000000)
// #use rs232(baud=9600,xmit=PIN_A3,recv=PIN_A2)

#byte port_b=6

***** 
**      LCD_E
**  Mikrosegundo bateko pultsua sortzen du RA2 (E) pinean
**  Fabrikatzaileak horrela eskatzen baitu.
***** /
```

```
void LCD_E0
{
    output_high(PIN_A2); // E batean jarri
    delay_us(1);
    output_low(PIN_A2); // E zeroan jarri
}
/*********************************************
**          LCD_BUSY
** LCD libre egon arte itxaroten du.
** IR (instrukzio-erregistroa) irakurri behar da, bertan, 7. bita busy flag da.
********************************************/
```

```
void LCD_BUSY()
{
    output_low(PIN_A0);           // RS : instrukzioak
    output_high(PIN_A1);          // R/W: irakurri
    set_tris_b(0xff);            // B ataka sarrera moduan
    output_high(PIN_A2);          // E seinalea aktibatu
    delay_us(1);
    while(input(PIN_B7)) // Itxaron egiten du BUSY "1" izan arte
    ;
    output_low(PIN_A2);          // E seinalea desaktibatu
    set_tris_b(0);               // B ataka irteera moduan
    output_low(PIN_A1);          // R/W: idatzi
}
/*********************************************
**          LCD_INSTR
** Instrukzio bat idazten du LCDan.
********************************************/
```

```
void LCD_INSTR(byte datui)
{
    LCD_BUSY();                  // LCDa prest egon arte itxaroten du
    output_low(PIN_A0);          // RS : instrukzioak
    output_low(PIN_A1);          // R/W: idatzi
    port_b = datui;              // instrukzioa bidali
    LCD_E0();                   // enable pultsu bat
}
/*********************************************
**          LCD_DATU
** Datu bat idazten du LCDan.
********************************************/
```

```
void LCD_DATU(byte datud)
{
    LCD_BUSY();                  // LCDa pres egon arte itxaroten du
    output_high(PIN_A0);         // RS : datuak
    output_low(PIN_A1);          // R/W: idatzi
    port_b = datud;              // datua bidali
    LCD_E0();                   // enable pultsu bat
}
/*********************************************
**          LCD_KATEA
** Kate bat idazten du LCDan.
********************************************/
```

```
void LCD_KATEA(char *p1)
{
    while (*p1 != '\0')
    {
        LCD_DATU(*p1);
        p1++;
    }
}

/******************
**          LCD_INI
** LCDa hasieratzen du.
**
*****/



void LCD_INI()
{
    byte num;
    for(num=3;num!=0;num--)
    {
        LCD_INSTR(0x38);           // 8 biteko modua
        delay_ms(5);              // guztira 15 ms
    }
    LCD_INSTR(0x1);             // LCDa ezabatu, kurtsorea lehenengo posiziora eta gehikuntza
    LCD_INSTR(0x0f);            // kurtsorea dardarka eta display on
}

/******************
**          DARDARA
** lcd-ak dar-dar egiten du, "aldiak" aldiz.
*****/



void DARDARA(byte aldiak)
{
    while (aldiak != 0)
    {
        LCD_INSTR(0x0b);         // LCDa itzali (off)
        delay_ms(500);
        LCD_INSTR(0x0f);         // LCDa piztu (on)
        delay_ms(500);
        aldiak--;
    }
}

/******************
main
*****/


void main(void)
{
    byte a;
    char katea1[14]={" Ongi etorri"};
    char katea2[14]={" ZAMALBIDERA"};

    set_tris_b(0);           // RB irteera
    set_tris_a(0x18);         // RA0-2 irteerak
    output_low(PIN_A0);       // RS = 0
    output_low(PIN_A2);       // E = 0
```

```
LCD_INI();
LCD_KATEA(katea1);
DARDARA(4);
LCD_INSTR(0xc0);           // Bigarren lerroaren hasieran pasatzen da
LCD_KATEA(katea2);
DARDARA(8);
for(a=0;a<16;a++)
{
    LCD_INSTR(0x18);       // Display-a 16 posizio desplazatzen da eskuinerantz
    delay_ms(500);
}
for(a=0;a<16;a++)
{
    LCD_INSTR(0x1c);       // Display-a 16 posizio desplazatzen da ezkerrerantz
    delay_ms(500);
}
}
```

## ***Ariketa proposatua C lengoain***

### **1. PRAKTIKA (keinukaria)**

RB3 hankatxoan keinukaria egin.

### **2. PRAKTIKA (keinukaria)**

B ataka osoan keinukaria egin.

### **3. PRAKTIKA (hankatxoen balioak pantailaratu)**

Ordenagailuaren pantailan RA0 eta RA1 hankatxoen balioak bistaratzen. Balioak aldatu ahala eguneratuko dira.

### **4. PRAKTIKA (Timer0 soft.)**

Timer0-a erabiliz, keinukaria egin hankatxo batean. Gainezka egiten duen jakiteko, testeoa egiteko softwarea egindo dugu. TMR0-an kargatu behar den balioa:

$$TMR0 = \frac{\text{denbora}}{4 \times \text{XTAL} \times \text{zatitzale a}}$$

### **5. PRAKTIKA (Timer0 etendura)**

Timer0-a erabiliz, keinukaria egin hankatxo batean. Gainezka egiten duen jakiteko, TIMER0-aren etendura erabiliko dugu. TMR0-an kargatu behar den balioa:

$$TMR0 = \frac{\text{denbora}}{4 \times \text{XTAL} \times \text{zatitzale a}}$$

### **6. PRAKTIKA (Timer0 etendura)**

Timer0-a bere etendurarekin erabiliz eta 3. adibidea abiapuntutzat hartuta diseinatu orduak, minituak eta segundoak adierazten dituen erlojua.

## **7. PRAKTIKA (Display)**

Ordenagailuaren teklatutik sartzen dugun zenbakia B atakan konektatutako zazpi segmentuko *display*-an agerrarazi. Hauexek dira mikropic\_trainer plakan eginak dauden konexioak:

LED seg a	Pin RB0
LED seg b	Pin RB1
LED seg c	Pin RB2
LED seg d	Pin RB3
LED seg e	Pin RB4
LED seg f	Pin RB5
LED seg g	Pin RB6
LED seg dp	Pin RB7

Eta hauexek dira atakatik atera behar diren datuak:

0 -> 0x3f	1 -> 0x06	2 -> 0x5b	3 -> 0x4f
4 -> 0x66	5 -> 0x6d	6 -> 0x7c	7 -> 0x07
8 -> 0x7f	9 -> 0x67		

## **8. PRAKTIKA (kontagailua)**

RA4/TOCK1 hankatxotik sartzen diren pultsuak zenbatu. Horretarako, TIMER0-a erabiliko dugu kontagailu bezala programatuta.

## **9. PRAKTIKA (Watchdog)**

*Watchdog*-aren funtzionamendua ikusteko, begizta bat egingo dugu *Watchdog*-ek gainezka egin gabe. Gero begiztak aterako gara eta beste begizta batean sartuko gara, baina horretan gainezka egiten utziko diogu. “Zakur zaindariak” mikroa berrasieratu duela egiazatzeko, *printf* bat jarriko dugu programaren hasieran.

## **10. PRAKTIKA (A/D)**

RA0 kanal analogikoa 100 aldiz irakurri behar da 10 segundoan. Hartutako balioen artean txikiena eta handiena pantailalaratu.

### **11. PRAKTIKA (A/D)**

A0, A1, A2 eta A3 kanal analogikoak irakurri, bakoitza 20 segundoan, balioak led-etan eta pantailan agerraraziz.

### **12. PRAKTIKA (teklatua)**

Teklatu matriziala irakurri PORTb etendura erabiliz. Irakurritako balioak pantailaratu.

### **13. PRAKTIKA (teklatua eta LCD)**

Teklatu matriziala irakurri PORTb etendura erabiliz. Irakurritako balioak LCDn agerrarazi.

# ELHUYAR

---

## ESKOLA-LIBURUAK

### HEZIKETA-ZIKLOAK

- Administrazio publikoa
- Antolakuntza eta ekoizpen-prozesuaren kontrola
- Elektronika analogikoa
- Elikagaien merkaturatzea
- Elikagaigintza-prozesuak I
- Elikagaigintza-prozesuak II
- Enpresa txiki eta ertainen administrazio eta kudeaketa
- Fiskalitatea
- GIRO NATURALEKO SOLDADURA  
Metalurgia
- GIRO NATURALEKO SOLDADURA  
Soldadura oxiazetilenikoa
- Kalitatea eta Etengabeko Hobekuntza.  
Ikaslearen liburua
- Kalitatea eta Etengabeko Hobekuntza.  
Irakaslearen liburua
- Lan-giroko harremanak
- Lan-prestakuntza eta orientabidea
- Logika Digitala eta Mikroprogramagarriak
- Lorategi eta zona berdeak
- Materia-ezagupena
- MEKANIZAZIO-PROZEDURAK  
Tornua: eragiketak eta erremintak, eta  
piezen mekanizazio-prozesu eta  
-prozedurak
- MEKANIZAZIO-PROZEDURAK  
Zulo-mota batzuk: zulatzeko makina, eta  
abar
- Merkataritzan Euskaraz
- Sistemen elektronika