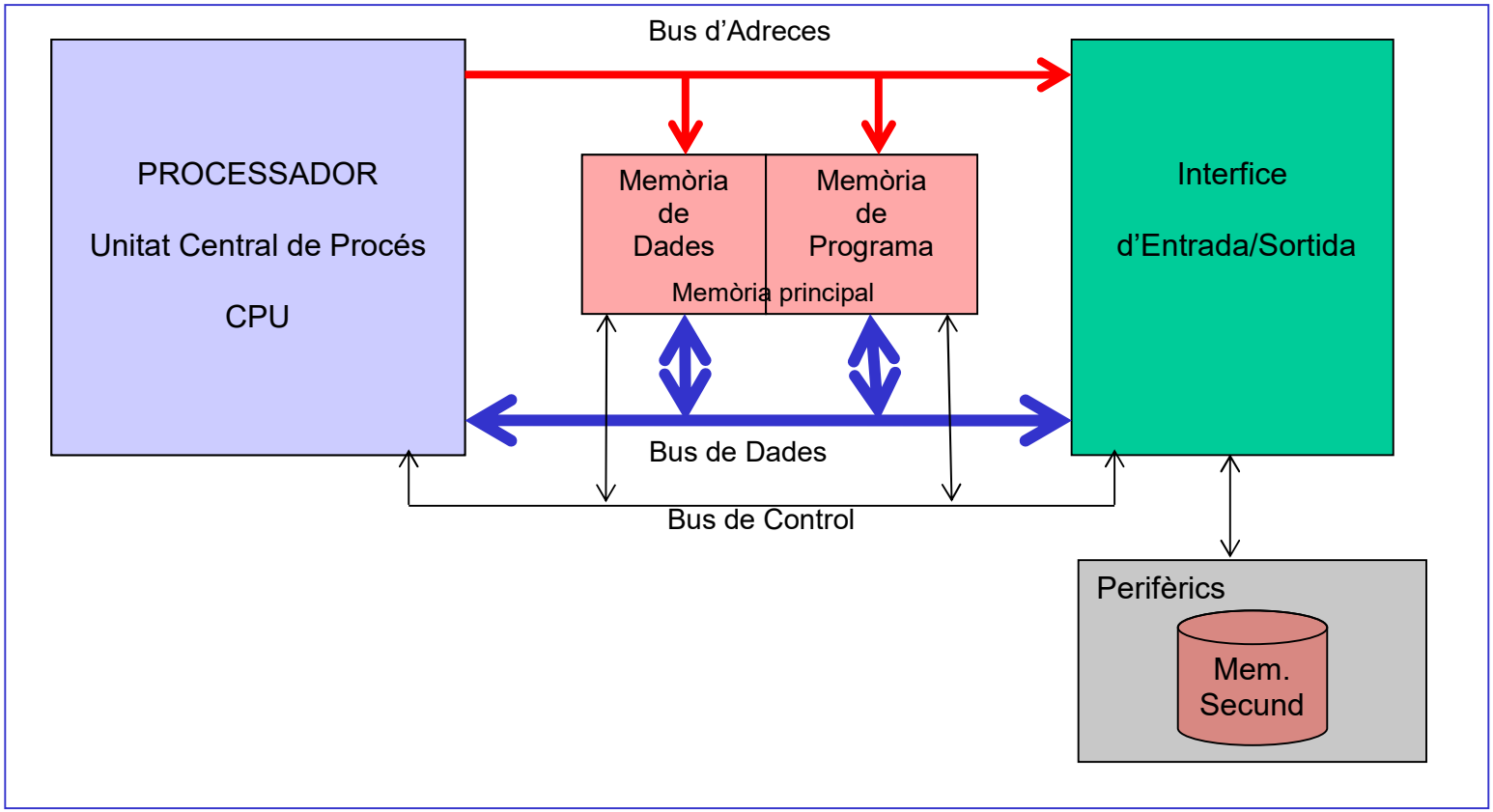
Organització
I
Estructura
dels Computadors

Funcionament d'un computador tipus Von Neumann

- Programes guardats a memòria secundària: (disc dur; CD,....)
- Unitat E/S porta el programa a executar fins a la memòria principal
- Execució d'instruccions una-a-una



Funcionament- Organització i estructura

CPU:

- Responsable d'executar les instruccions
- Controlar el funcionament de la resta d'elements del computador.
- Llegir les instruccions de memòria,
- Interpretar-les
- Realitzar les accions necessàries per executar-les.

- UNITAT DE CONTROL
- UNITAT DE PROCÉS O EXECUCIÓ
- REGISTRES
- Memòria Caché (Nivells 1 i 2)

La CPU Unitat de Procés o Unitat d'Execució

- On s'executen les instruccions.
- Multitud de Tasques:
- Càlcul, operacions
- Moviment de dades
- Càlcul d'adreces

Elements típics:

- Una o més ALUs:
- Operacions aritmètiques: suma, resta, desplaçament, complement
- Operacions lògiques: AND, OR, XOR, NOT
- Altre recurs de càlcul: multiplicador, desplaçadors (opcional)
- Un conjunt de registres:
- operands i resultats
- Nombre de registres = potència de la CPU
- Generador d'adreces.
- Cerca d'instruccions,
- Llegir, escriure dades de/a memòria

Unitat Aritmetico Lògica (I)

La Unitat Aritmètico Lògica s'encarrega de tractar les dades, executant les operacions definides d'acord amb el programa en curs que s'està executant.

Tal i com s'ha comentat (i com defineix el seu nom) la ALU fa bàsicament

Operacions Aritmètiques
- +
- -
- desplaçaments
- multiplicacions
- divisions

Operacions Lògiques
- AND
- OR
- NOT
- NAND
- NOR

Com es dissenya això??

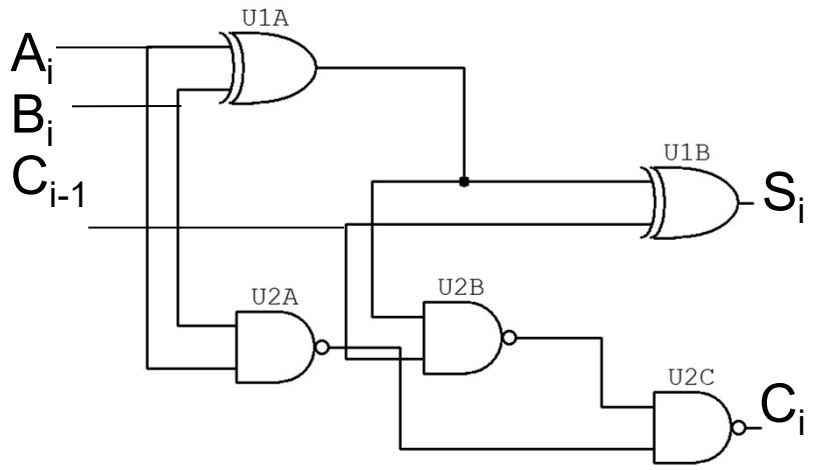
Unitat Aritmetico Lògica (II)

Pel que fa a la part aritmètica... SUMADOR DE 1 BIT

El sumador es senzill

A	B	C-1	S	C
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

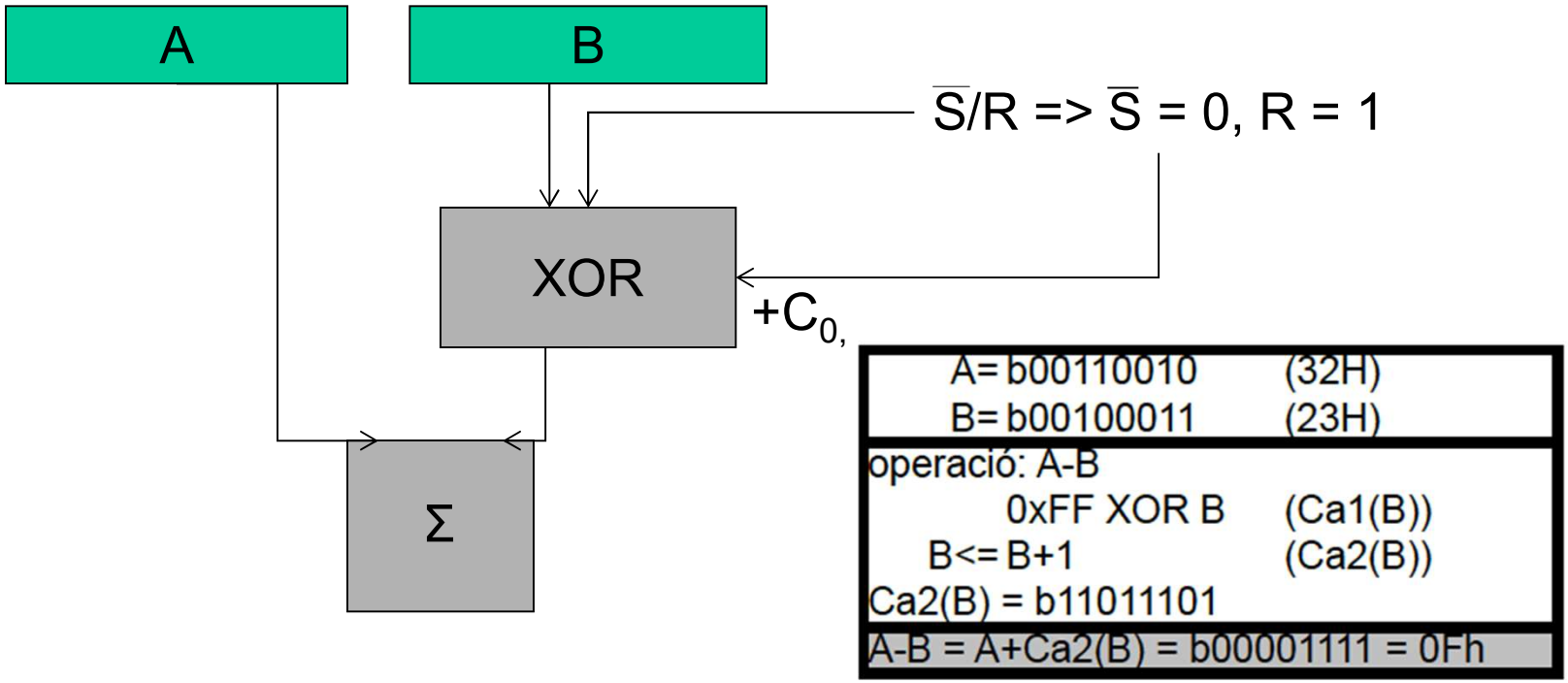
D'on es treu que...
Si = Ai XOR Bi XOR Ci-1
Ci = AiBi + AiCi-1 + BiCi-1



Unitat Aritmetico Lògica (III)

A partir de la suma podem obtenir senzillament la resta...

$A - B = A + Ca2(B)$



Unitat Aritmetico Lògica (IV)

Operacions de desplaçament:

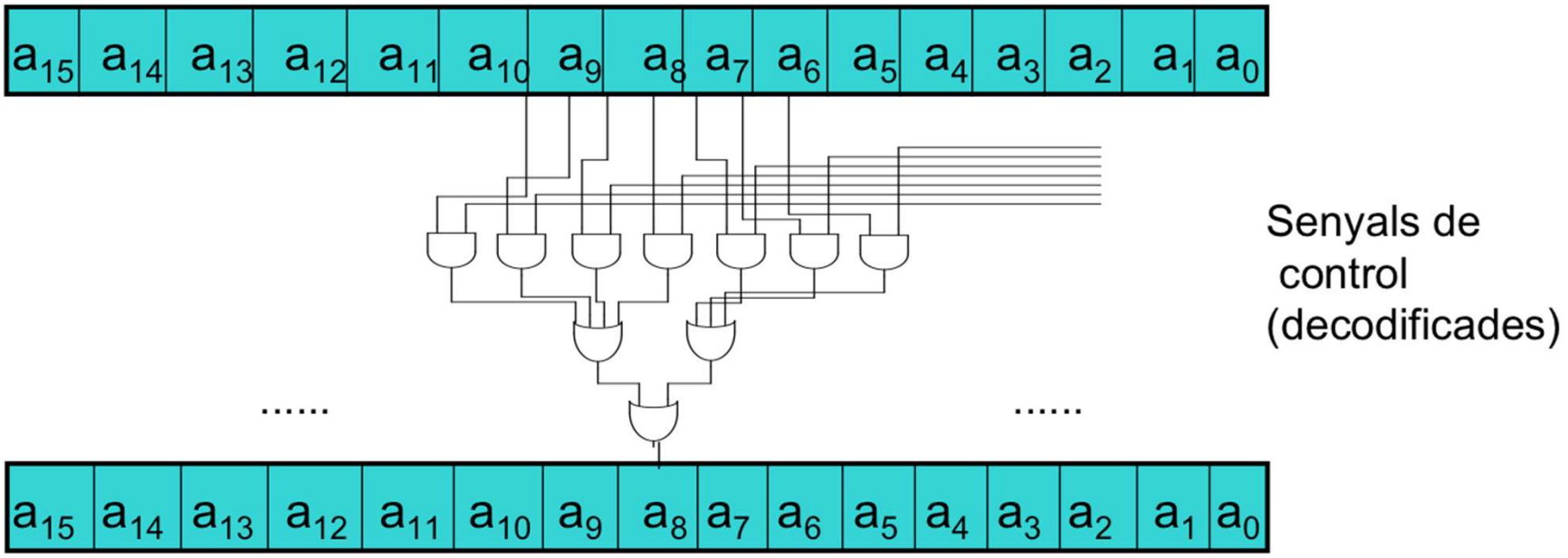
Consisteix en traslladar els bits d'una paraula cap a l'esquerra o cap a la dreta.

Els computadors més senzills només permeten desplaçaments d'una sola posició cap a la dreta o cap a l'esquerra.

Els computadors més complexos permeten realitzar desplaçaments múltiples, fent servir un operador específic o aplicant n cops el desplaçament unitari

Unitat Aritmetico Lògica (V)

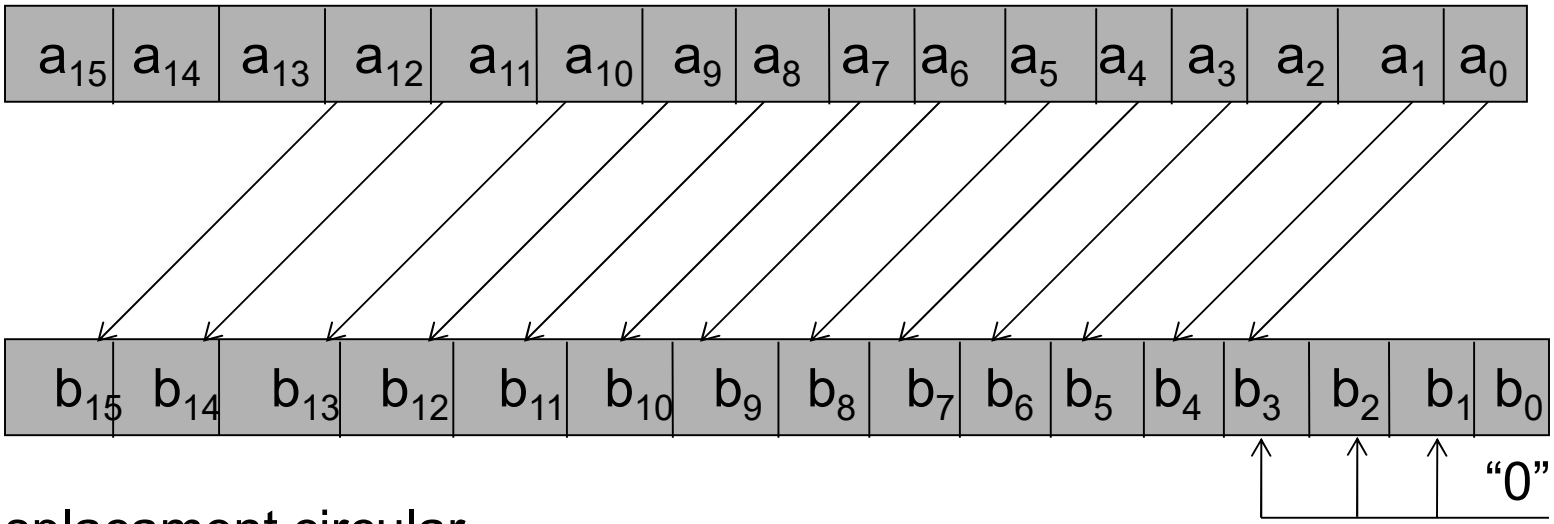
Sistema de desplaçament de múltiple. A l'exemple tenim un màxim de 3 posicions



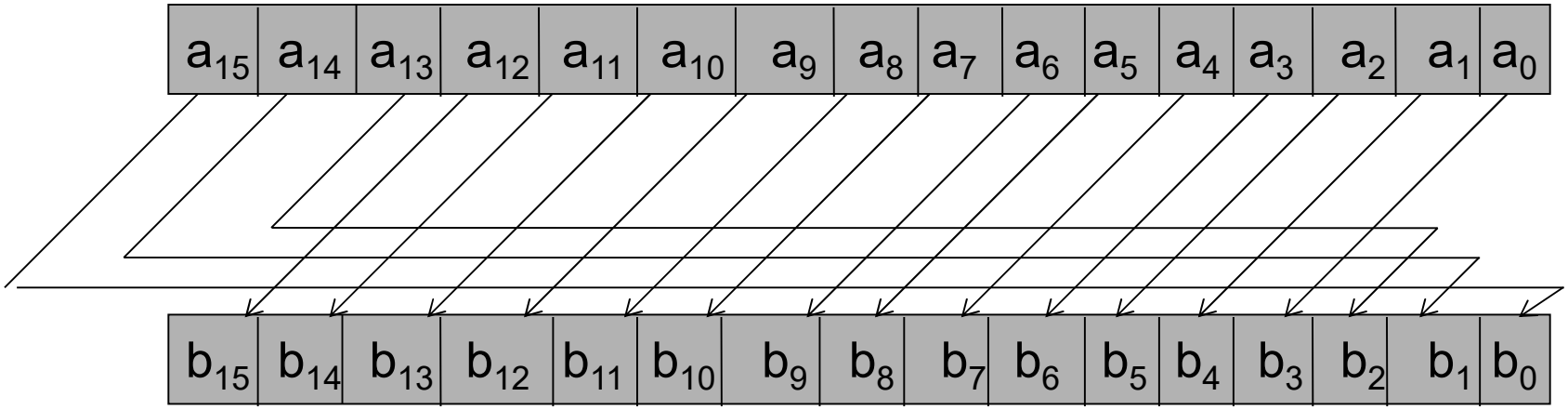
Estructura de desplaçament amb dos registres

Unitat Aritmetico Lògica (VI)

Desplaçament lògic

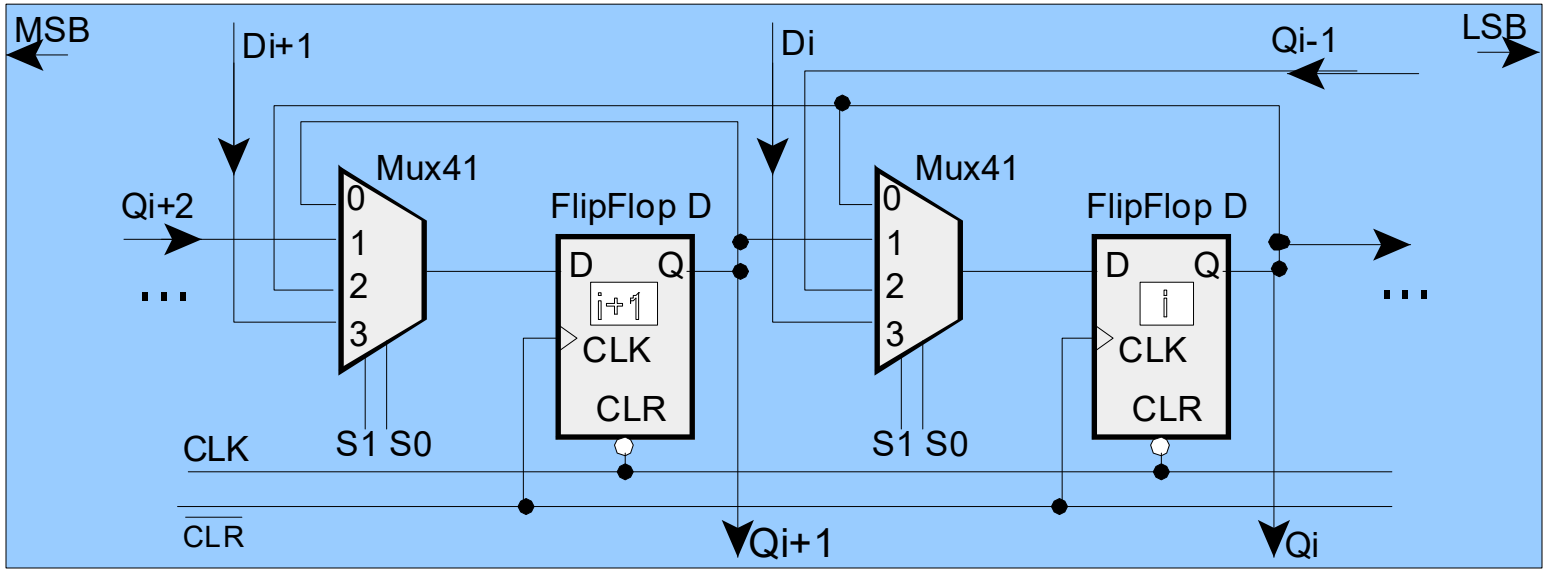


Desplaçament circular



Unitat Aritmetico Lògica (VII)

Registre de desplaçament.



Permet desplaçament circular. El desplaçament en aquesta funcionalitat és de tan sols un bit

Unitat Aritmetico Lògica

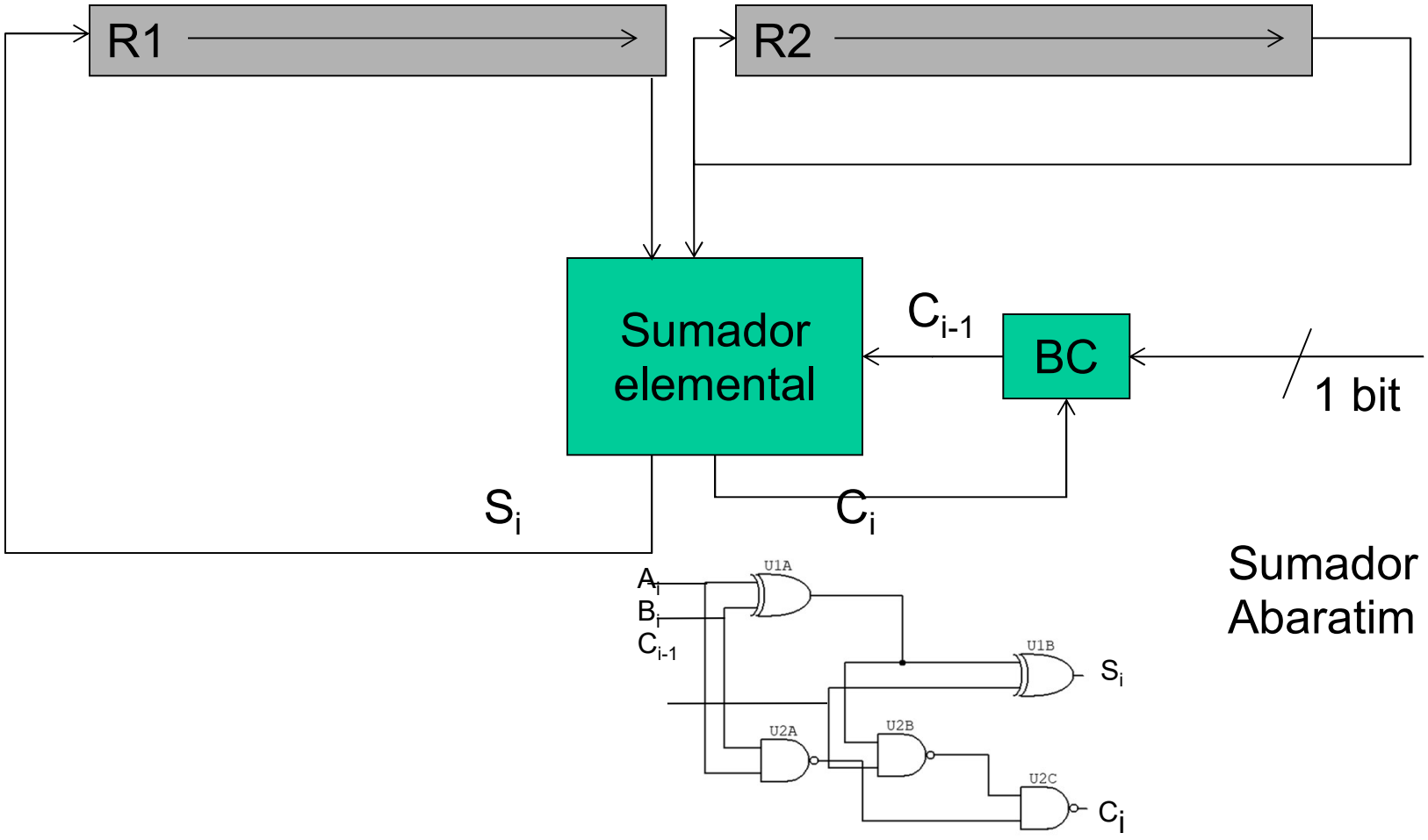
Desplaçaments aritmètics

- Consisteix en una multiplicació o divisió per una potència de dos.
- Per realitzar correctament el desplaçament, s'ha de conservar el signe de l'operand.
- En el cas de la multiplicació, s'han d'introduir dígits nuls per la dreta, per obtenir el resultat correcte.
- En el cas de quantitats negatives s'ha d'anar més en compte (Ca1 o Ca2)
- Incorporen un detector de desbordament (overflow)

Biestable de desbordament

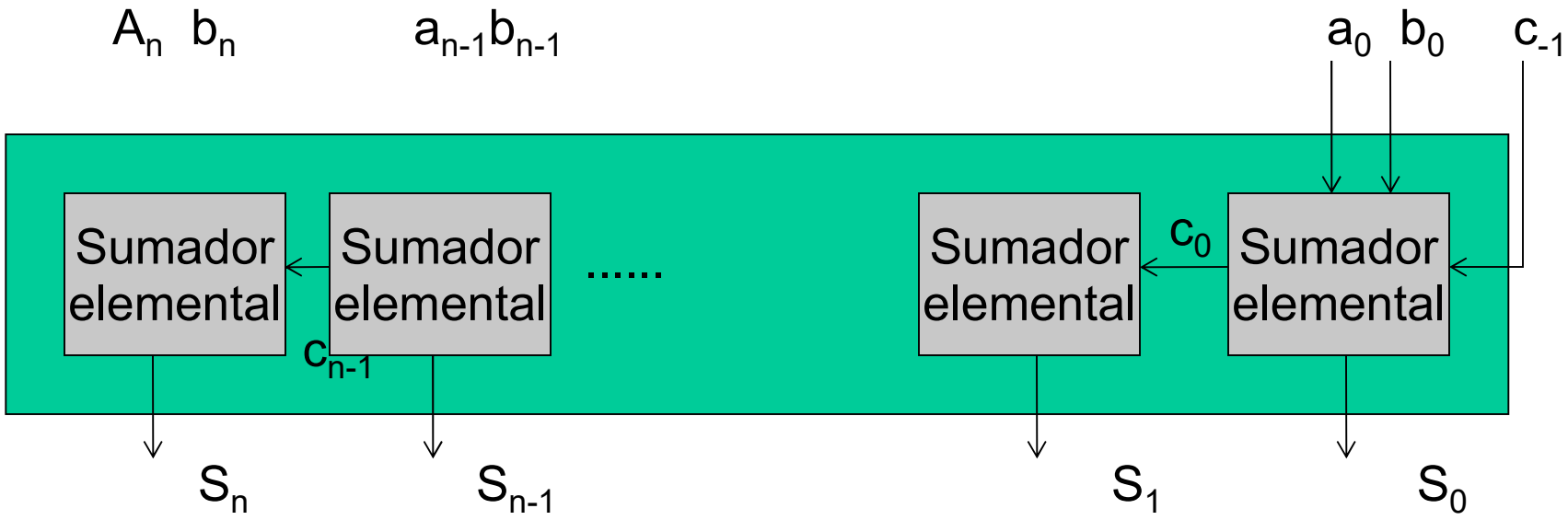
Unitat Aritmetico Lògica (VIII)

Amb la utilització de registres de desplaçament podem realitzar les operacions en sèrie, es a dir

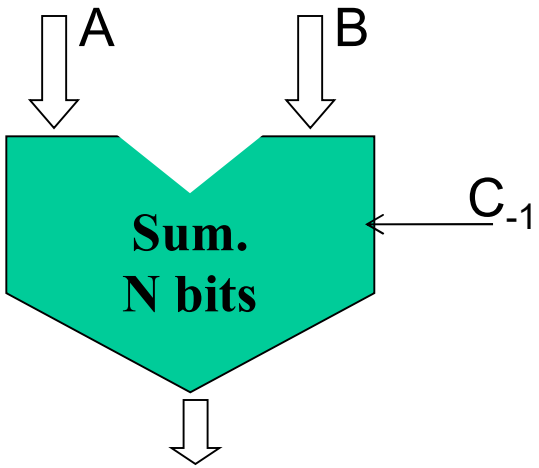


Sumador sèrie.
Abaratim costos

Unitat Aritmetico Lògica (IX)



Representació simplificada del sumador d'n bits

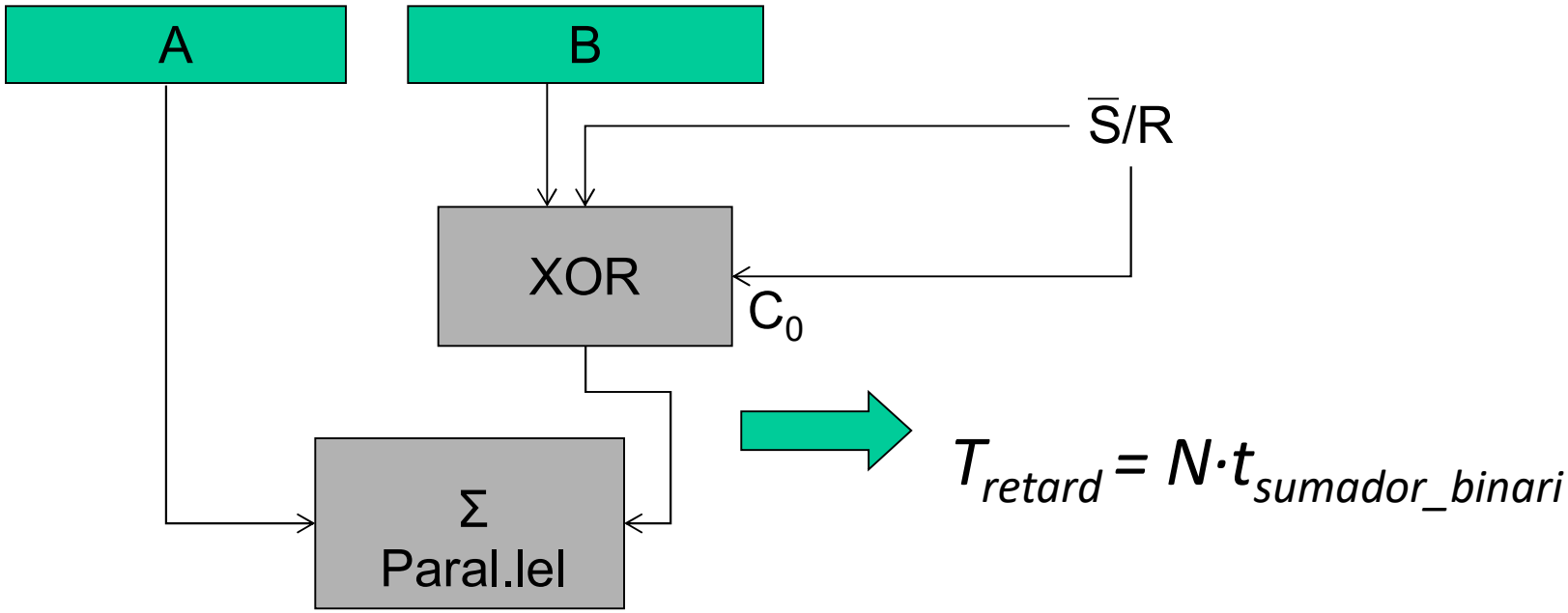


Sumador paral.lel

Més ràpid

Unitat Aritmetico Lògica (X)

El sumador paral·lel clàssic té un temps de suma que ve imposat pel retard associat a la propagació del carry (**Ripple-Carry Adder** o CPA).
És més lent com més gran és el nombre de bits dels registres
S_{31} depèn de C_{30} que depèn de C_{29} que depèn... D'aquí es diu que el Carry ondula o navega a través de la cadena de carrys



Propagació del Carry

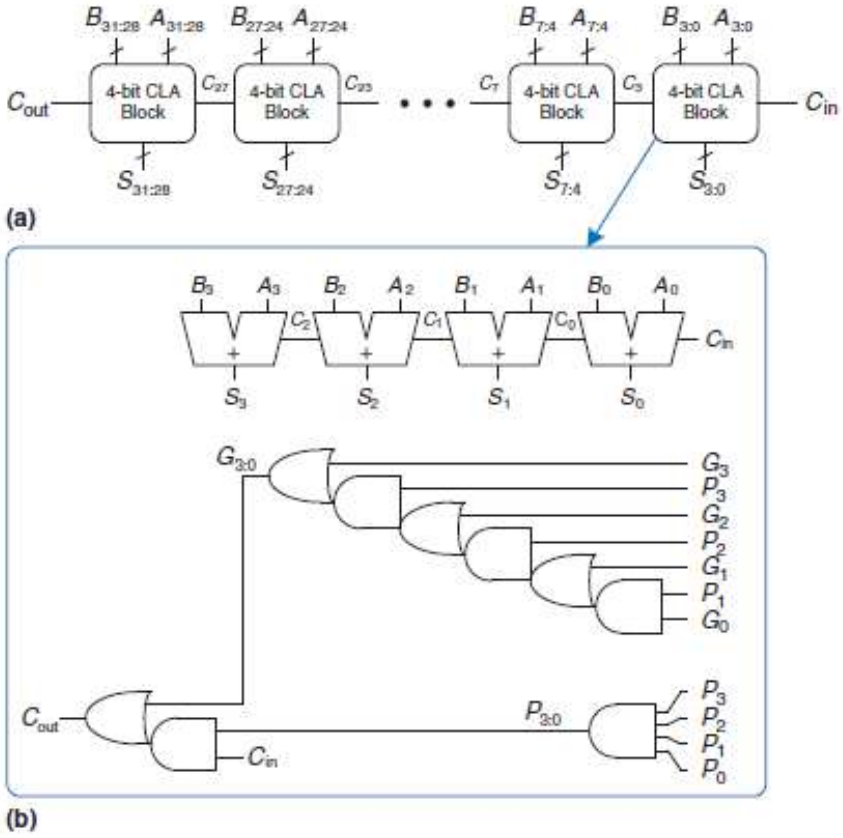
-El primer mètode utilitzat és el d'"***avançar el valor del carry***" amb el mètode de la propagació del Carry (Carry lookahead Adder (CLA)).
-Divideix el sumador en blocs i proporciona circuiteria per determinar el Carry de sortida de un bloc tant aviat com es conegut el carry d'entrada
-Utilitza senyals de generació i propagació
-Dintre del bloc el carry es propaga igual que en el cas anterior.

$G_i=A_iB_i$

$P_i=A_i+B_i$ $\qquad\qquad$ $C_i=A_iB_i+(A_i+B_iG)C_{i-1}=G_i+P_i\ C_{i-1}$

Carry lookahead Adder (CLA)

- $t_{CLA}=t_{pg}+ t_{pg\_bloc}+(N/k -1)\ t_{AND\_OR}+ kt_{sumac}$



Unitat Aritmetico Lògica (XI)

PREFIX ADDER => Calculen el generador i el propagador abans de Fer la suma

El disseny d'aquest tipus de circuit requereix de dos funcions lògiques: G i P: generador de Carry i propagador de Carry

$g_i = a_i \cdot b_i \quad p_i = a_i \text{ XOR } b_i$

Els valors que s'obtenen d'aquestes expressions no necessiten cap propagació, obtenint-se de forma immediata

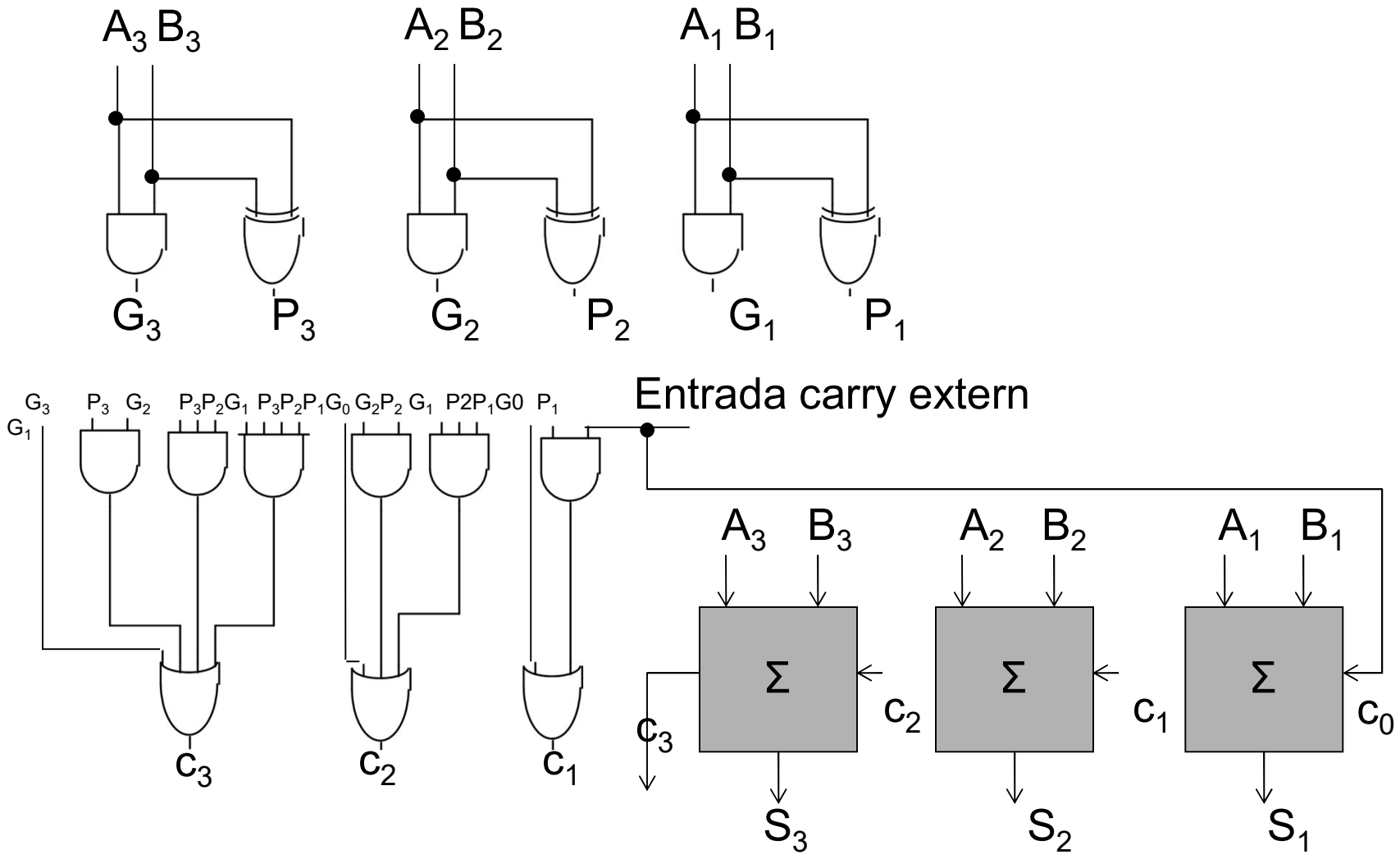
$c_i = g_i + p_i \cdot c_{i-1}$
$s_i = p_i \text{ XOR } c_{i-1}$

Aplicant aquestes formules repetidament s'obté:

$c_i = g_i + p_i \cdot g_{i-1} + p_i \cdot p_{i-1} \cdot g_{i-2} + \ldots + p_i \ldots p_0 \cdot c_{-1}$

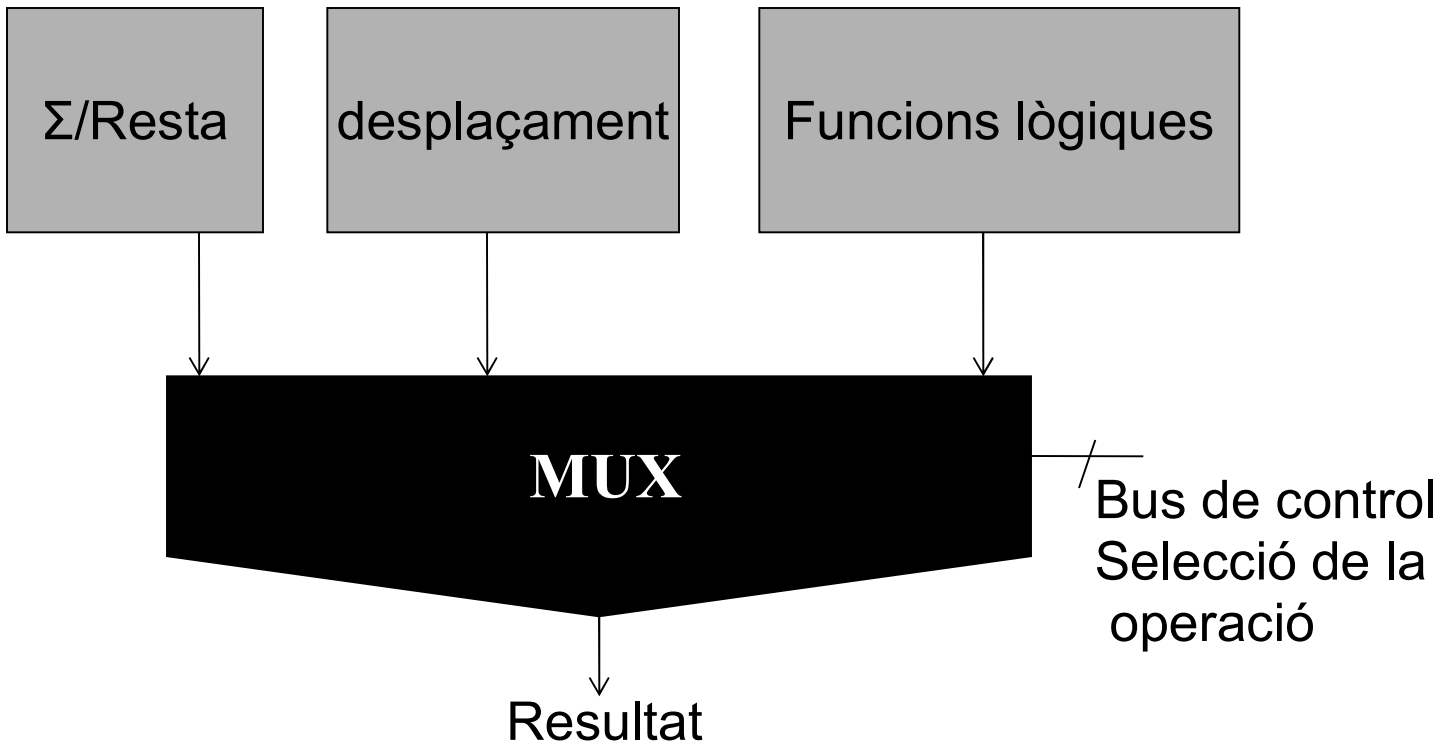
Unitat Aritmetico Lògica (XII)

L'aplicació d'això per a un sumador de 3 bits serà...



Unitat Aritmetico Lògica (XIII)

Per tant, la estructura de la ALU genèrica (a nivell funcional) serà...



Unitat Aritmetico Lògica (XIII)

Bus de dades:

Tres arquitectures de Camins de dades seqüencialitzats (Depenent del nº de BUSos utilitzats

1.- Només un BUS

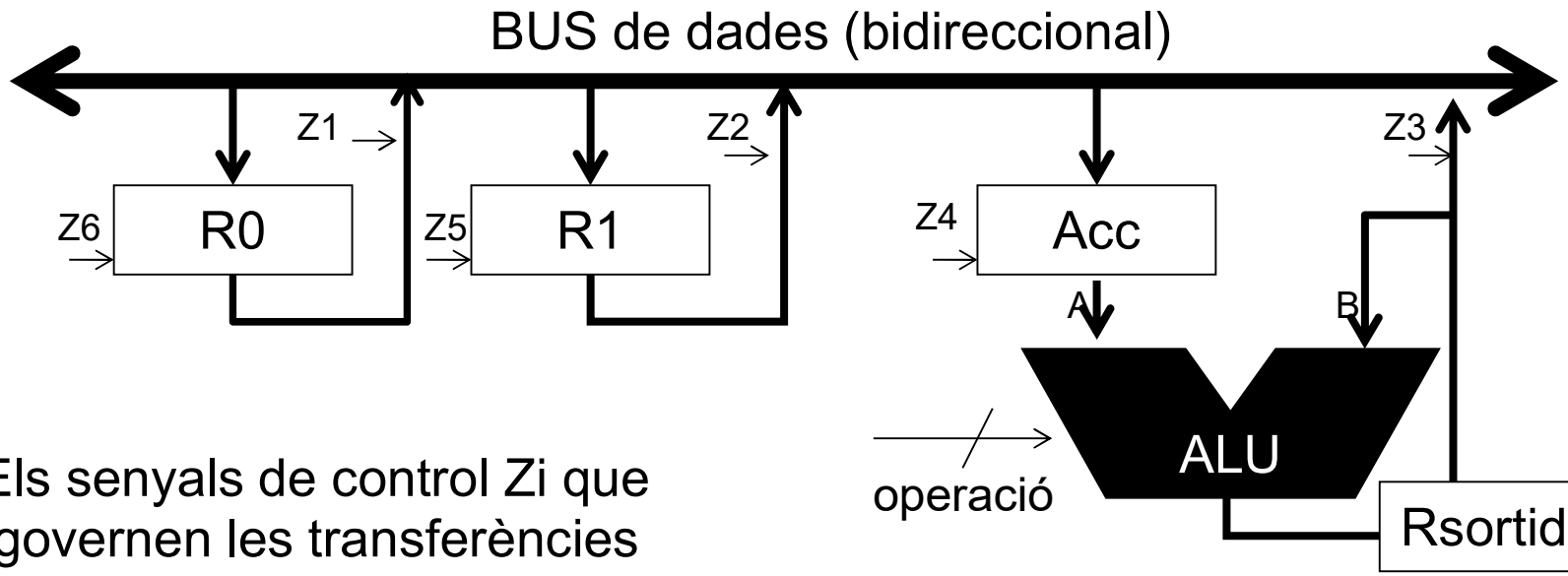
2.- Amb dos BUSos

3.- Amb tres BUSos (no s'utilitza gaire degut al seu elevat cost)

Unitat Aritmetico Lògica (XIV)

Camí de dades seqüencialitzat amb un BUS

-Problema: La seva lentitud al haver de realitzar més cicles per implementar la mateixa instrucció. El seu esquema general és:



Els senyals de control Zi que governen les transferències són generades per la UC al interpretar cada fase de la instrucció en fase d'execució (en curs)

Unitat Aritmetico Lògica (XV)

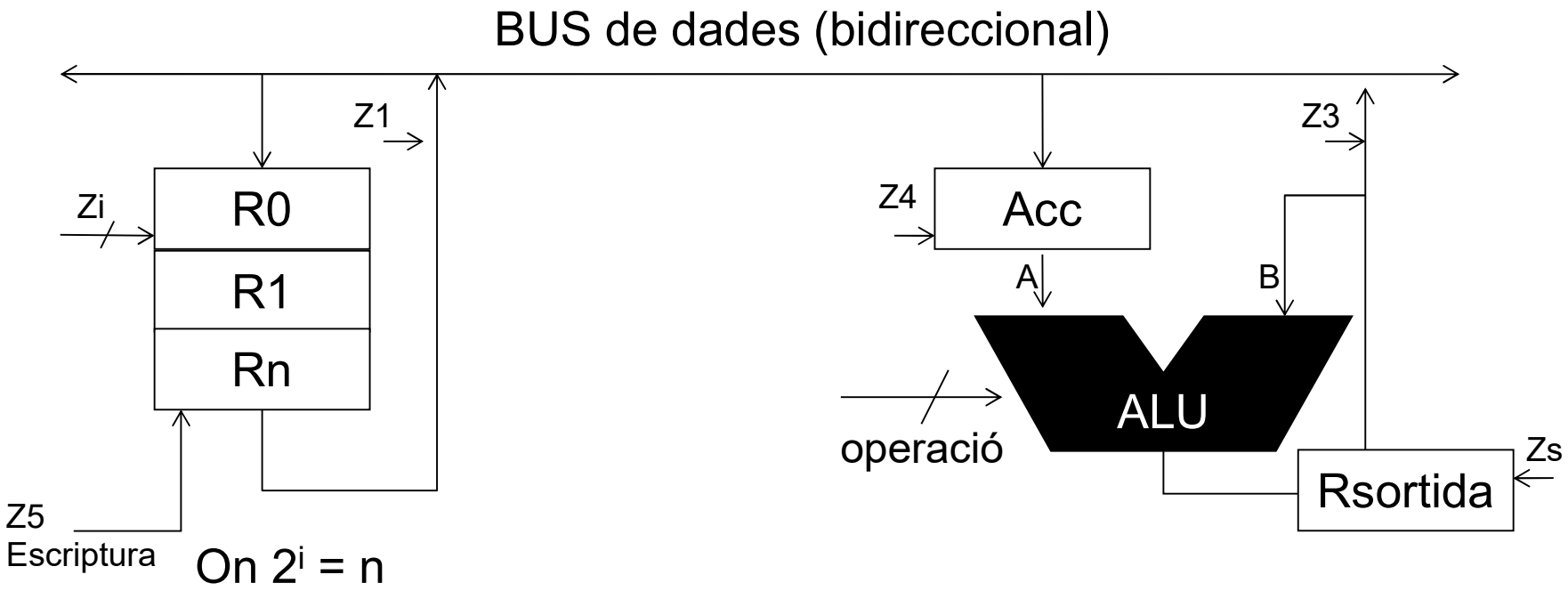
1.- El nº de senyals de control que es precisen per utilitzar els registres és el doble de la quantitat de registres que existeix, ja que cada registre precisa d'un senyal per ser escrit i un altre per ser llegit.

2.- El BUS es carrega en funció del valor dels senyals de control. Quan Zi = 1 es produeix l'activació de la sortida => Això implica que si s'activen simultàniament dos o més senyals de control es produirà un conflicte al bus.

Unitat Aritmetico Lògica (XVI)

Millora de l'esquema de la unitat de procés.

Agrupació dels registres en un banc de registres => Reduïm el nº de senyals de control



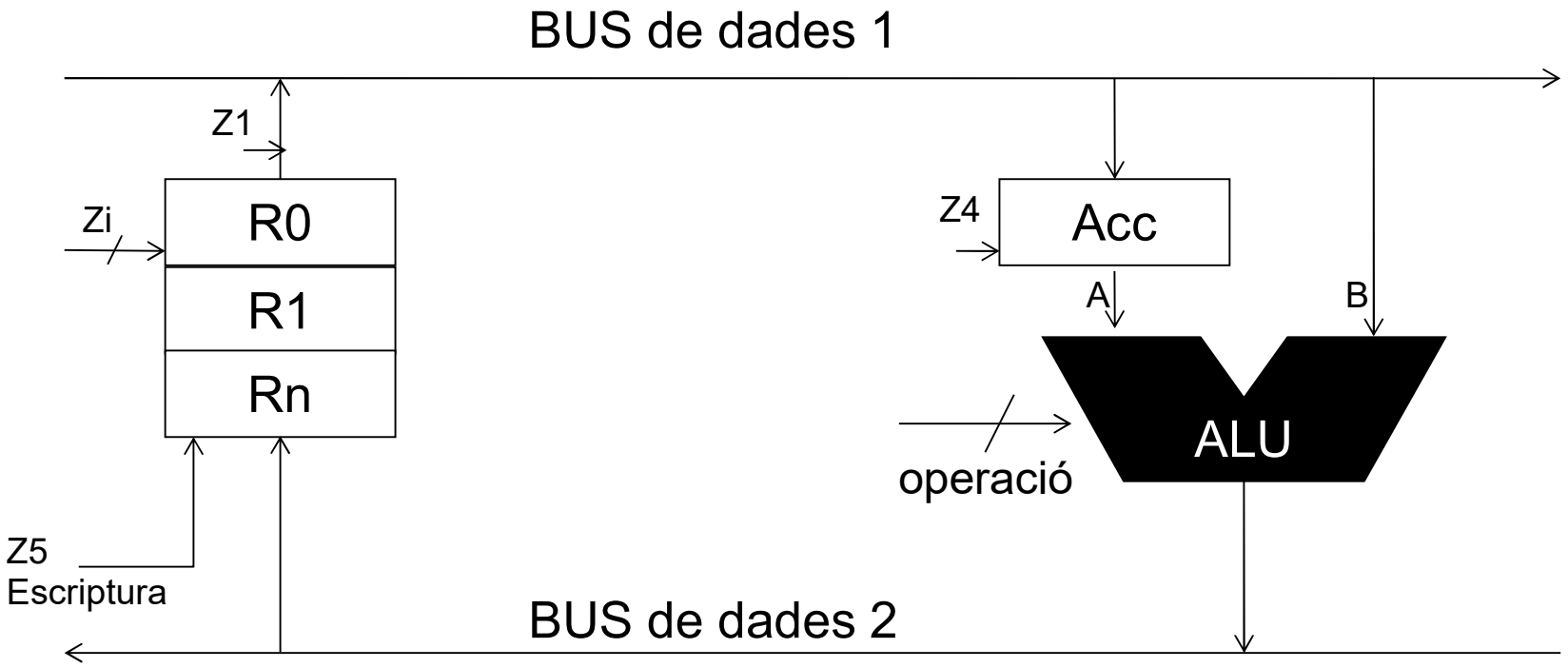
Unitat Aritmetico Lògica (XVII)

Exercici exemple: Definir l'esquema d'una unitat de procés que disposa d'un banc de 4 registres. Indiqueu el valor que han de prendre els senyals de control per carregar a l'acumulador el contingut del registre R2

Unitat Aritmetico Lògica (XVIII)

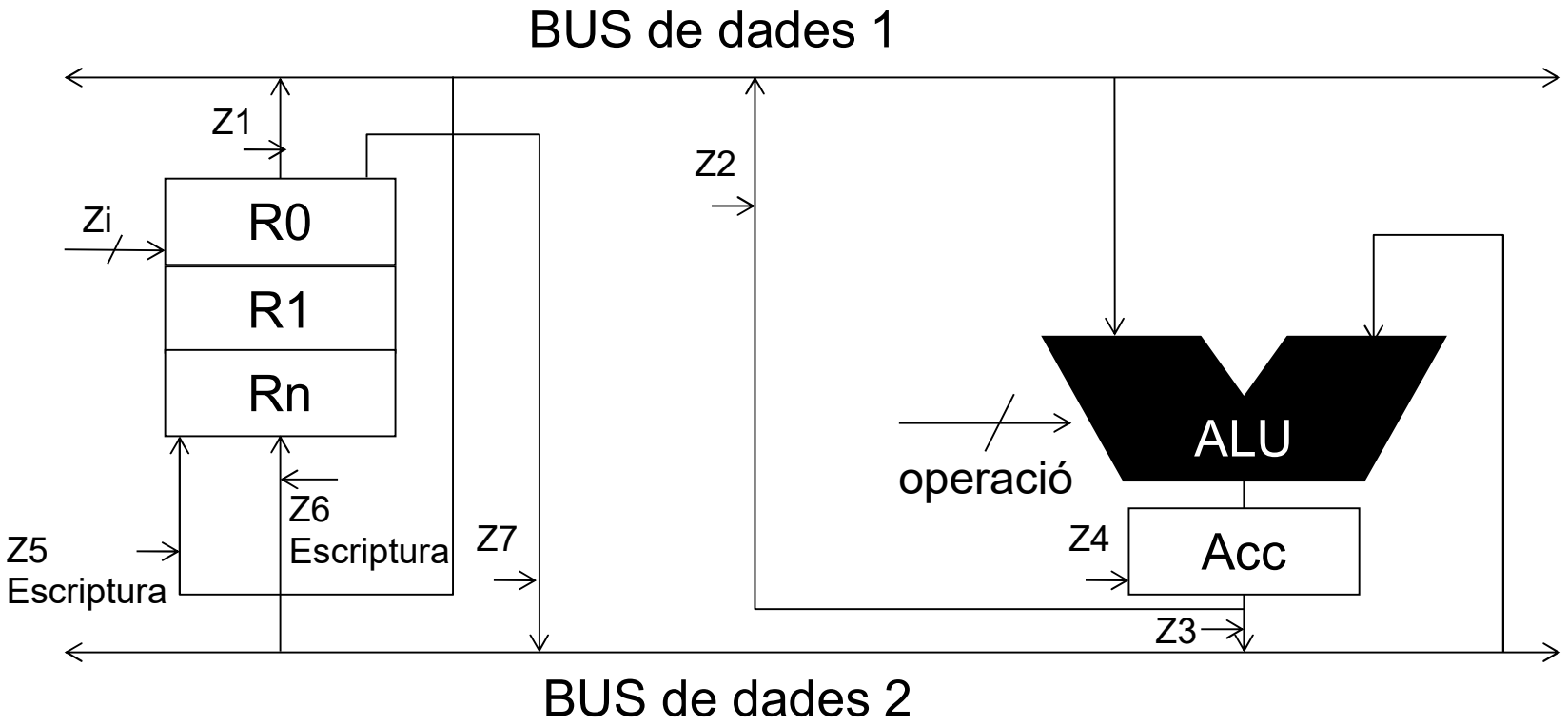
Unitat de Procés seqüencialitzada amb dos BUSos

Tot i que representa un increment de cost substancial, millora la velocitat al reduir els cicles necessaris per fer una operació

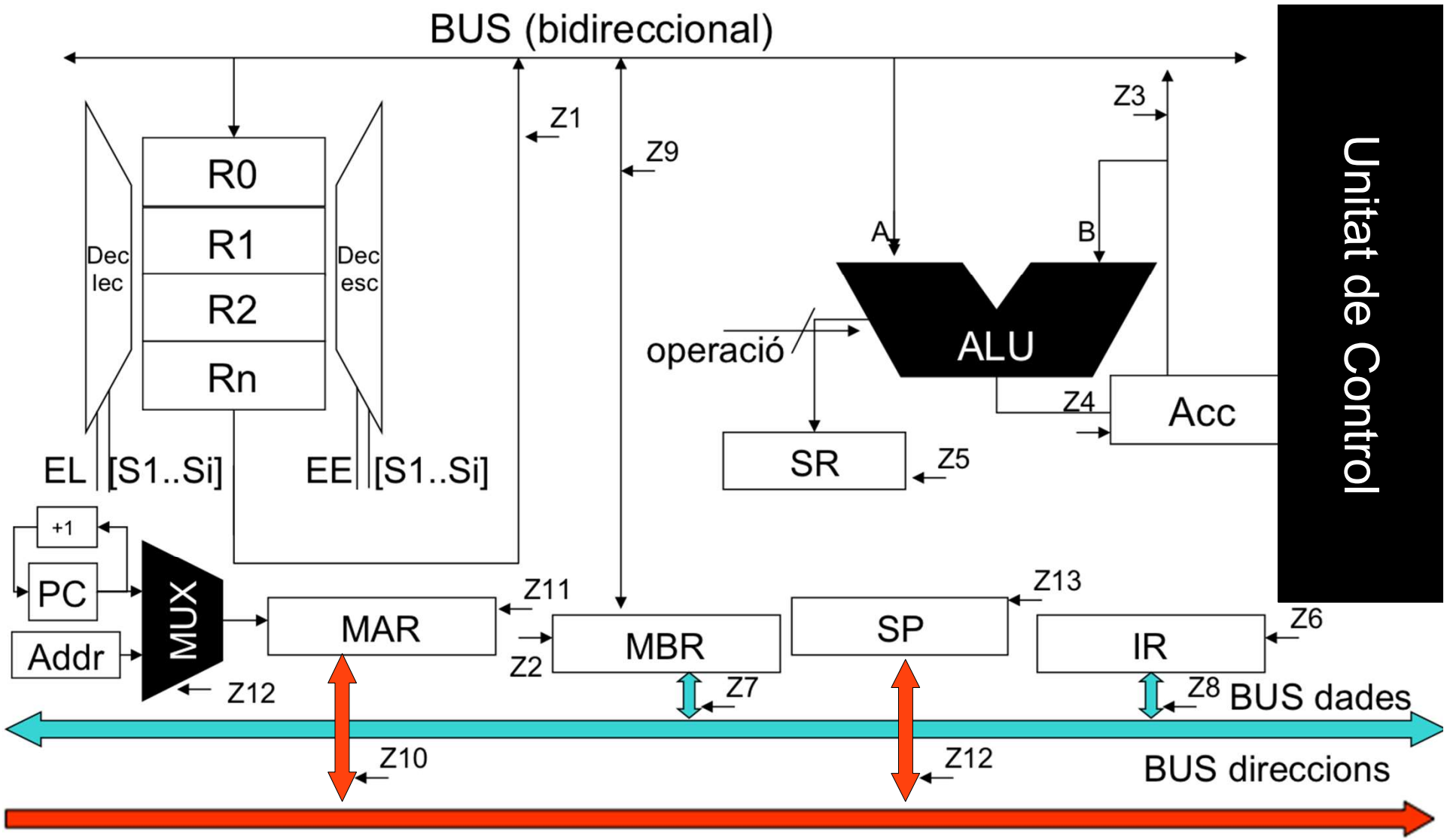


Unitat Aritmetico Lògica (XIX)

Optimització de l'arquitectura anterior. BUSos bidireccionals. Augment del rendiment i millora de la capacitat de processament.



Estructura de la CPU



Exemple CPU S1C88

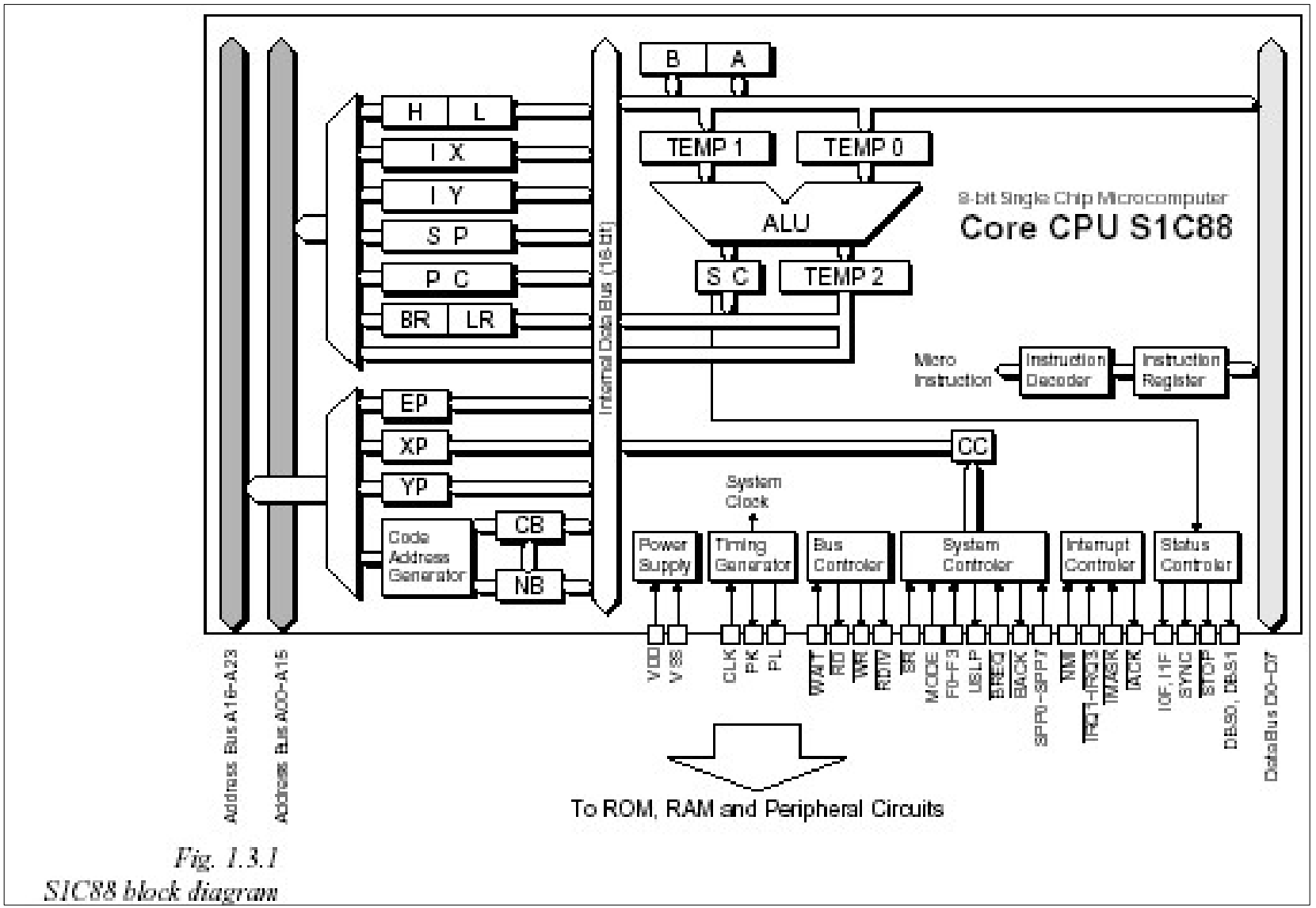
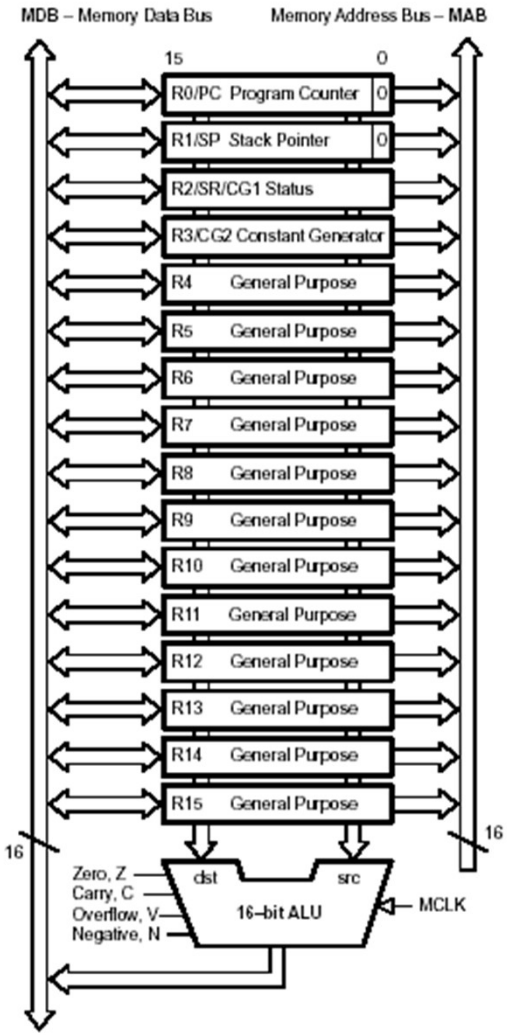


Fig. 1.3.1
S1C88 block diagram

Exemple CPU MSP430

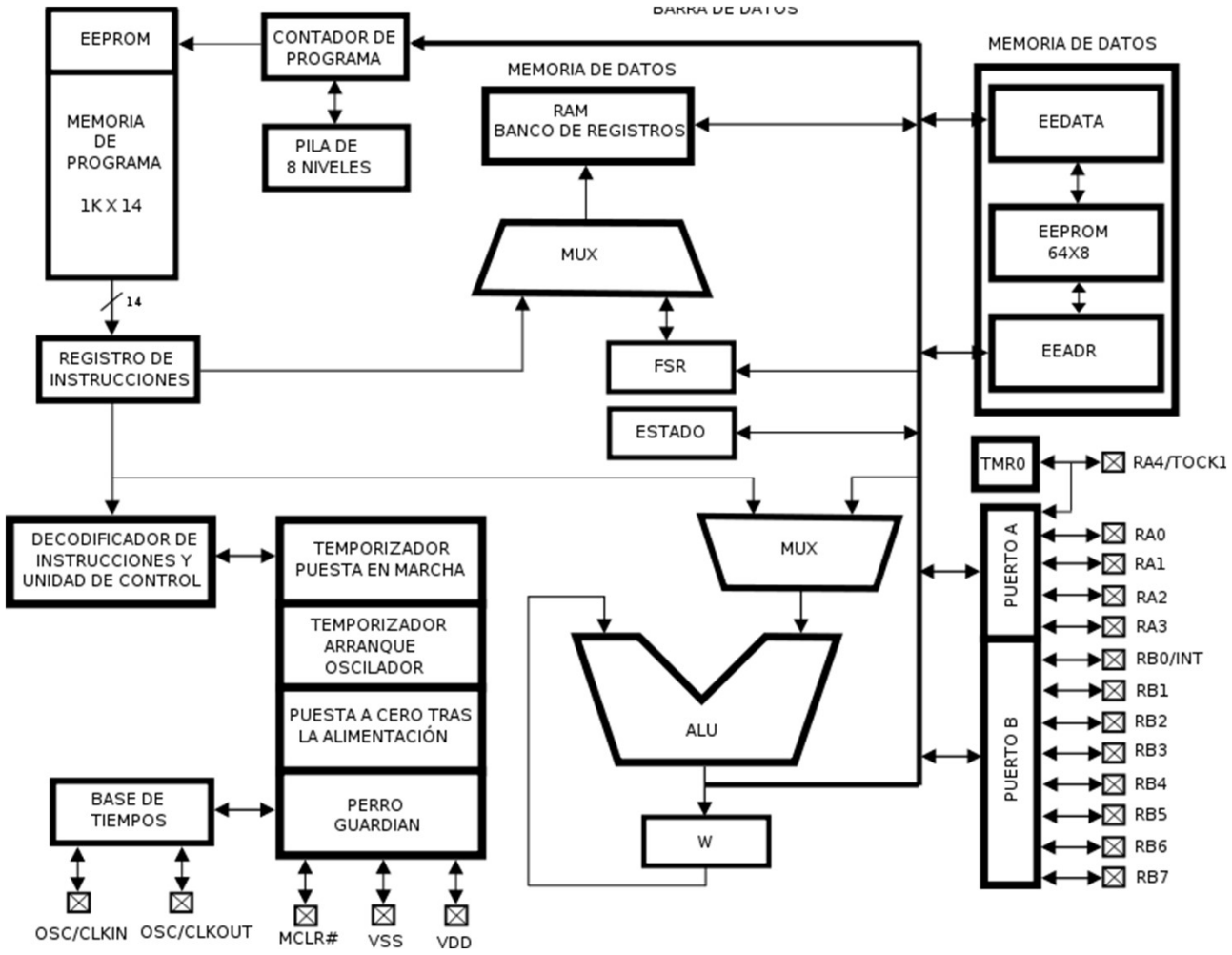


The CPU features include:

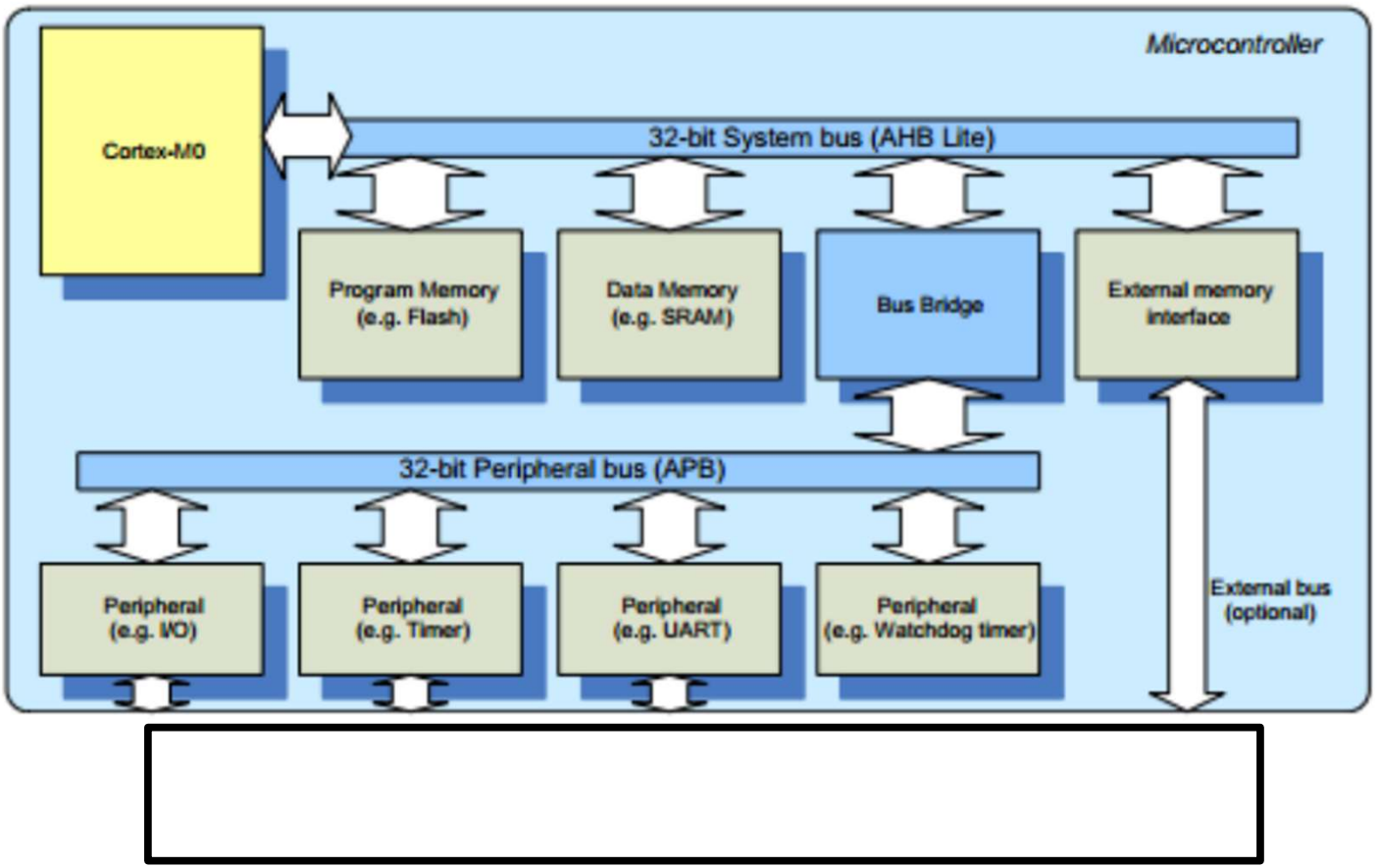
- RISC architecture with 27 instructions and 7 addressing modes.
- Orthogonal architecture with every instruction usable with every addressing mode.
- Full register access including program counter, status registers, and stack pointer.
- Single-cycle register operations.
- Large 16-bit register file reduces fetches to memory.
- 16-bit address bus allows direct access and branching throughout entire memory range.
- 16-bit data bus allows direct manipulation of word-wide arguments.
- Constant generator provides six most used immediate values and reduces code size.
- Direct memory-to-memory transfers without intermediate register holding.
- Word and byte addressing and instruction formats.

The block diagram of the CPU is shown in Figure 3–1.

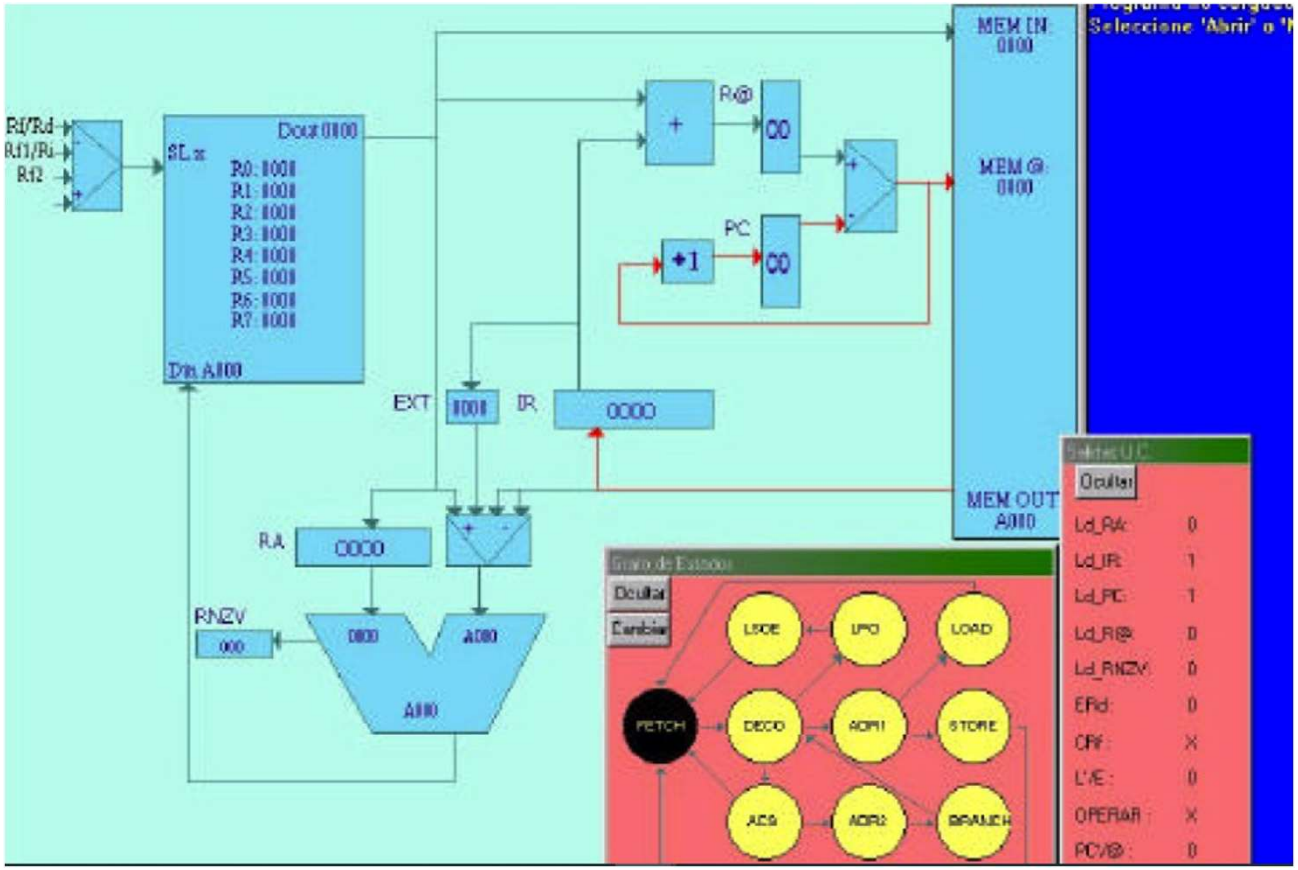
Exemple CPU Arduino



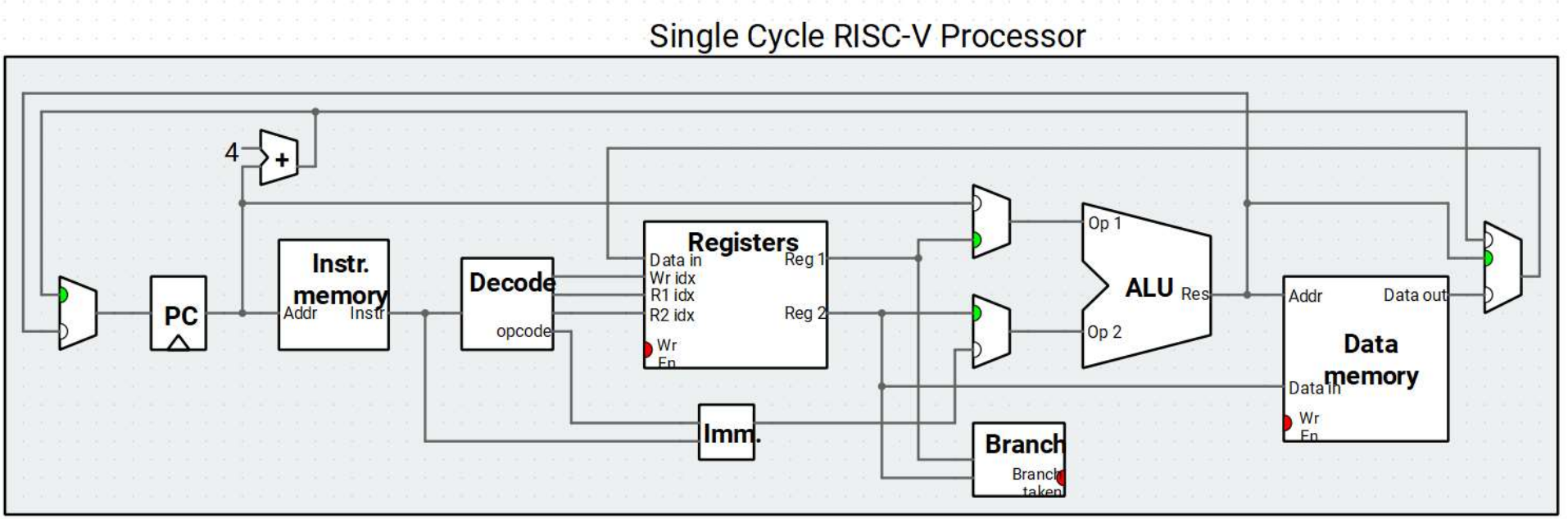
Exemple CPU ARM Cortex M0



Exemple CPU SiMR



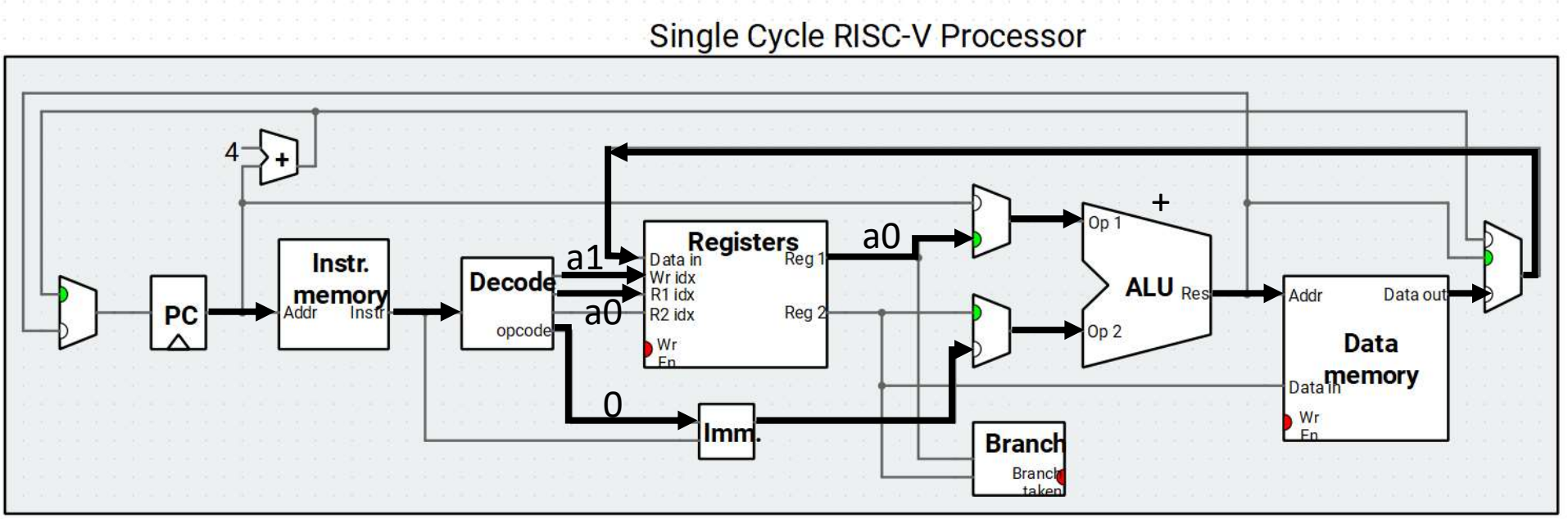
Exemple arquitectura RISC-V



ML1

Aplic en arquitectura RISC-V

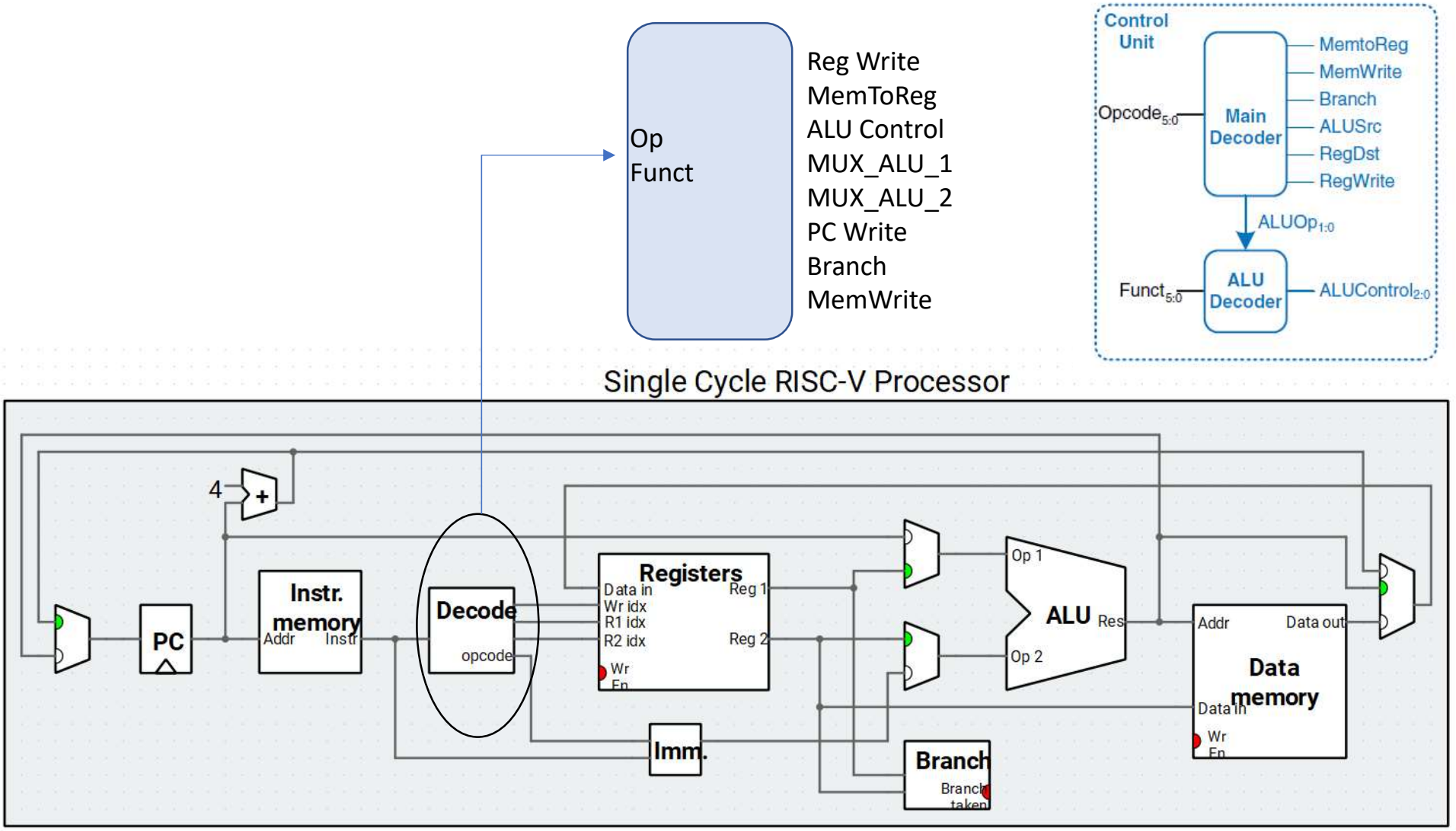
- Instrucció LOAD => lw a1, 0(a0)



Diapositiva 35

ML1 Manel Lopez; 16/03/2021

Control en arquitectura RISC-V



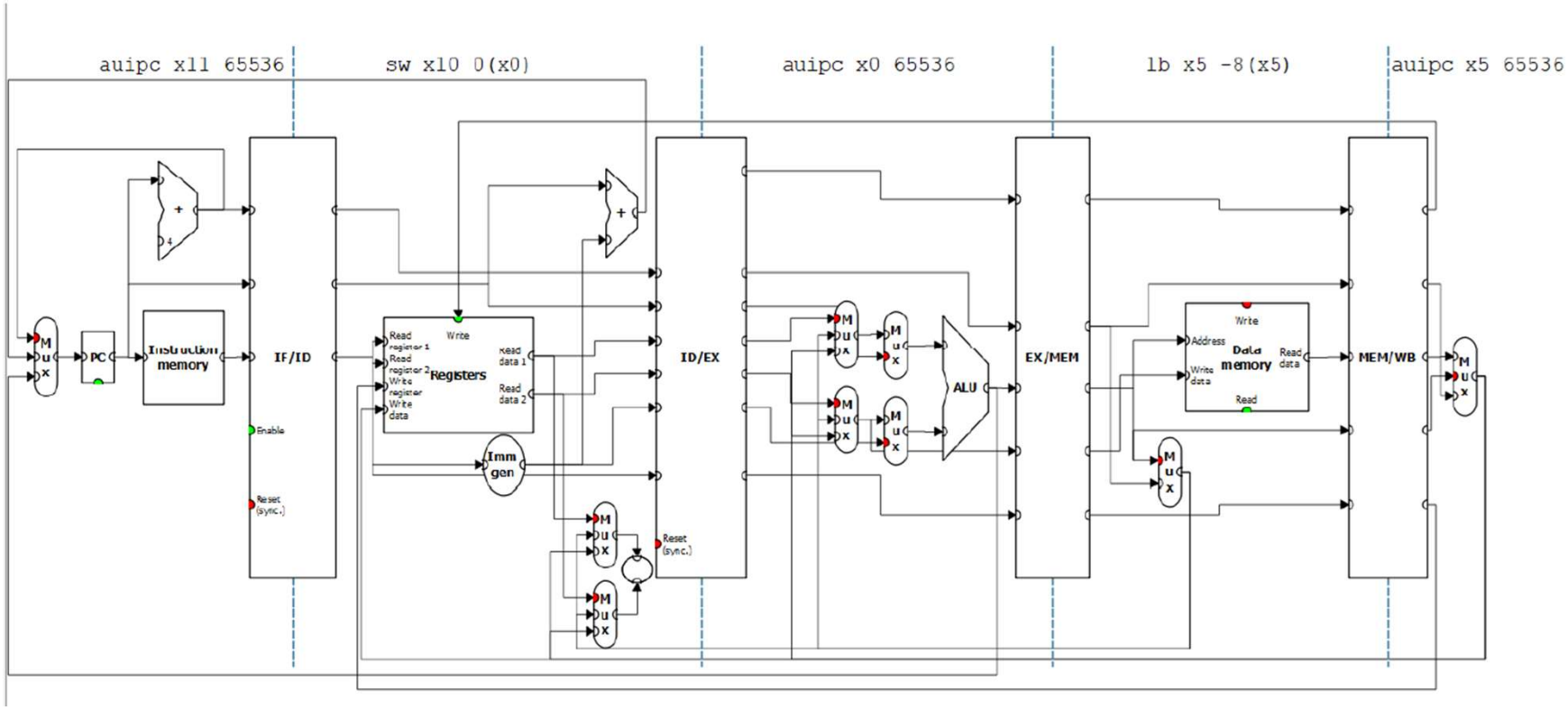
Control en arquitectura RISC-V

ALUOp	Meaning
00	add
01	subtract
10	look at `funct` field
11	n/a

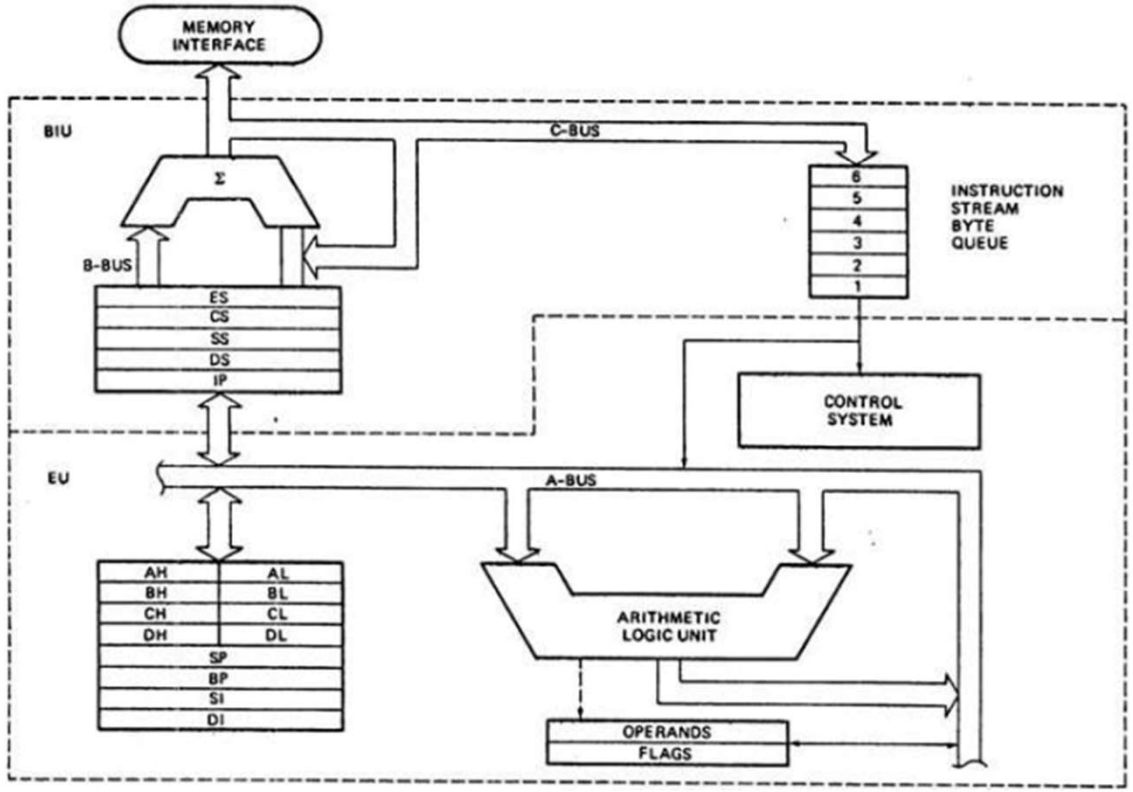
ALUOp	Funct	ALUControl
00	X	010 (add)
X1	X	110 (subtract)
1X	100000 (`add`)	010 (add)
1X	100010 (`sub`)	110 (subtract)
1X	100100 (`and`)	000 (and)
1X	100101 (`or`)	001 (or)
1X	101010 (`slt`)	111 (set less than)

Instruction	Opcode	RegWrite	RegDst	ALUSrc	Branch	MemWrite	MemtoReg	ALUOp
R-type	000000	1	1	0	0	0	0	10
`lw`	100011	1	0	1	0	0	1	00
`sw`	101011	0	X	1	0	1	X	00
`beq`	000100	0	X	0	1	0	X	01

Estructura processador RISC-V

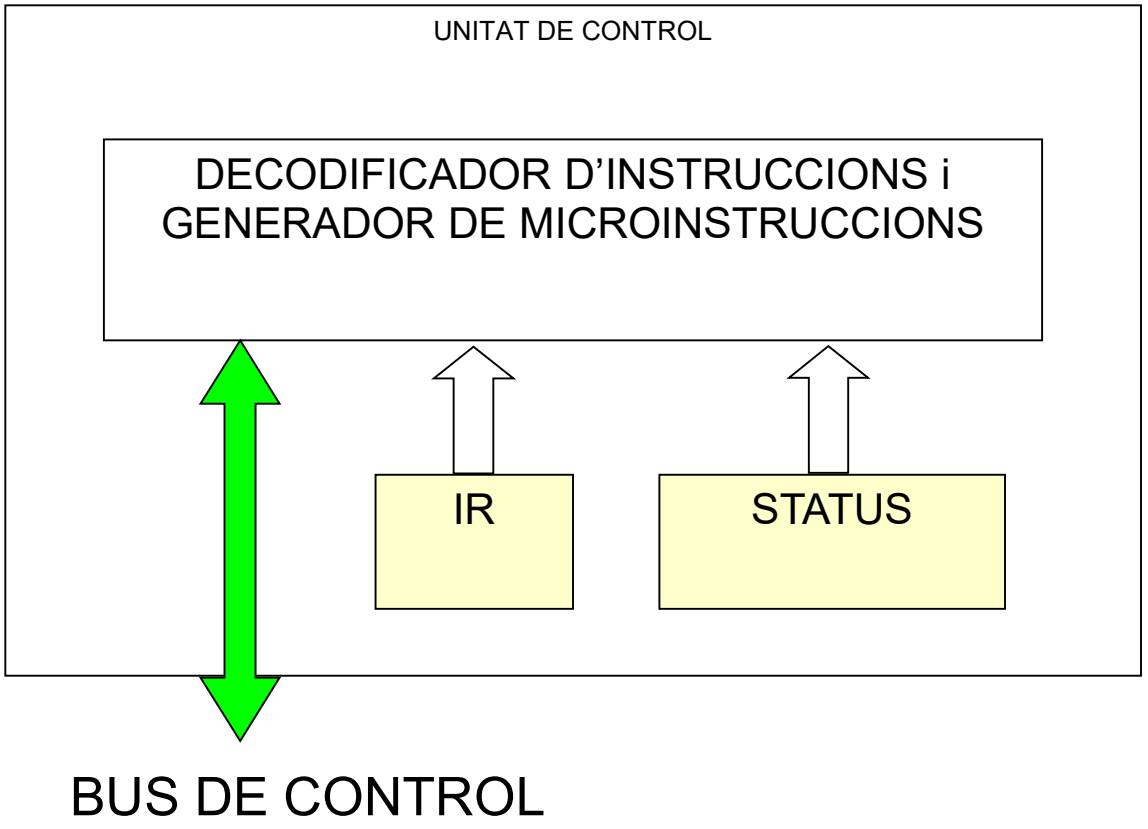


Unitat PROCÉS INTEL 8086

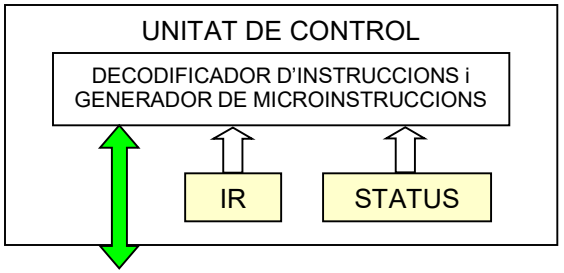


Arquitectura INTEL 8086

La CPU La Unitat de Control



La CPU La Unitat de Control(II)



Unitat de Control:

- Dirigeix i controla tota l'activitat de la Unitat de Procés.
- Rep de la memòria el codi màquina de les instruccions a executar
- Envia **comandes** de control dels seus components
- (Recursos; Registres; Connexió)
- Rep senyals de **estat**
- Interpreta i Descodifica les instruccions.
- Controla l'execució de les instruccions, generant les senyals adequades
- Generar la seqüència de senyals que activen els elements del sistema, per executar la instrucció. **Microinstruccions.**

CPU La Unitat de Control(III)

Elements de la UC:

1.Descodificador d'instruccions. Màquina d'estats per generar les Microinstruccions.

2. Entrades de la UC:

   1.Registre d'Instruccions **(IR)** guarda la instrucció durant l'execució

   2.Registre Estat (**Status Register)**. Info. resultats d'operacions ALU (sig, carry, zero, ovf).

   El contingut del registre influeix sobre la microinstrucció i sobre els salts.

3. Seqüenciador d'instruccions

4. Sortida UC. Bus de control. Comptador de Programa **PC. Reg. Addr**

Essencialment és un màquina d'estats.
Les UCs poden ser de dos tipus

1.Unitat de control **cablejada** (*hardwired*)
•Màquina d'estats convencional
2.Unitat de control **microprogramada**
•ROM + Registre

CPU La Unitat de Control(IV)

Unitat de control cablejada (*hardwired)*

- Es una unitat de control dissenyada en ***hardware***.
- Utilitza els recursos de l'electrònica digital (portes, registres, MUX,…)
- Elevada complexitat de disseny. Gran velocitat (commutacions HW)
- Prima la potencia per davant del cost.

Unitats de control microprogramades

Les variables d'entrada i les variables d'estat actual componen una adreça d'una ROM d'on s'obtenen les variables d'estat següent i les variables de control del procés (comandes de la UP: pex. selecció de funcionament dels recursos, activació d'operacions amb registres, etc.)

micro-instrucció: Cada conjunt de valors de les variables d'estat següent i de variables de control

Una microinstrucció provoca el moviment de dades entre registres. Equival a una instrucció RTL.

Microprograma: Seqüència de microinstruccions.
Microinstrucció: Comandaments a línies de control per efectuar operacions amb registres (microcomandament)
Microoperació: Operació controlada per un microcomandament

Unitats de control microprogramades

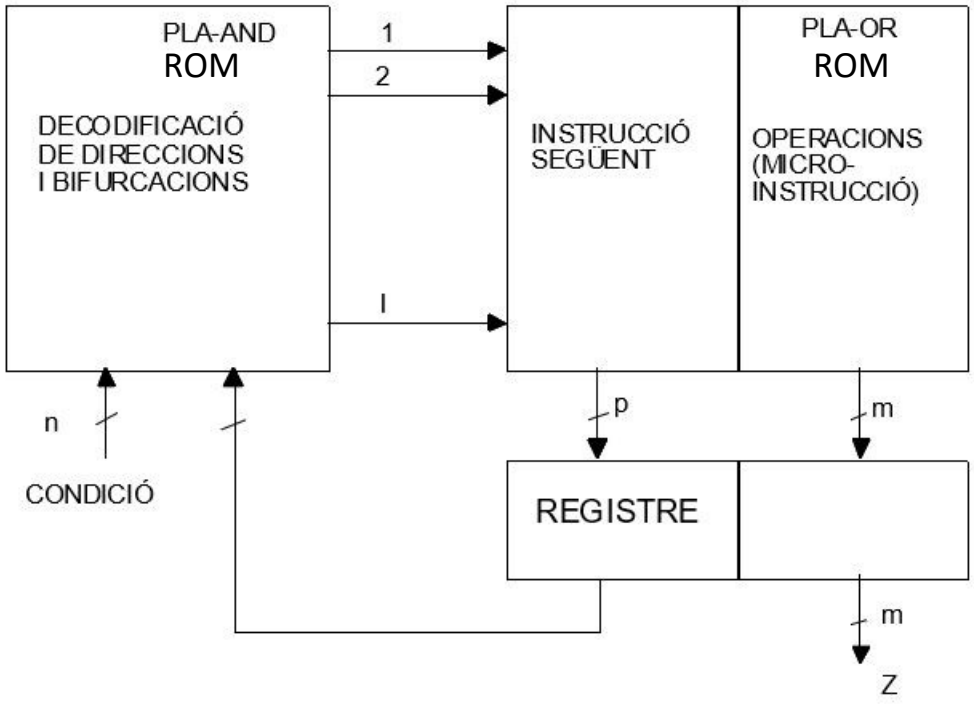
- Canviabilitat de l'arquitectura
- Flexibilitat
- Adaptabilitat
- Compatibilitat de *software*
- Diagnosi

- No és econòmic per a sistemes senzills
- Pèrdua de velocitat
- Cost del *software* de suport

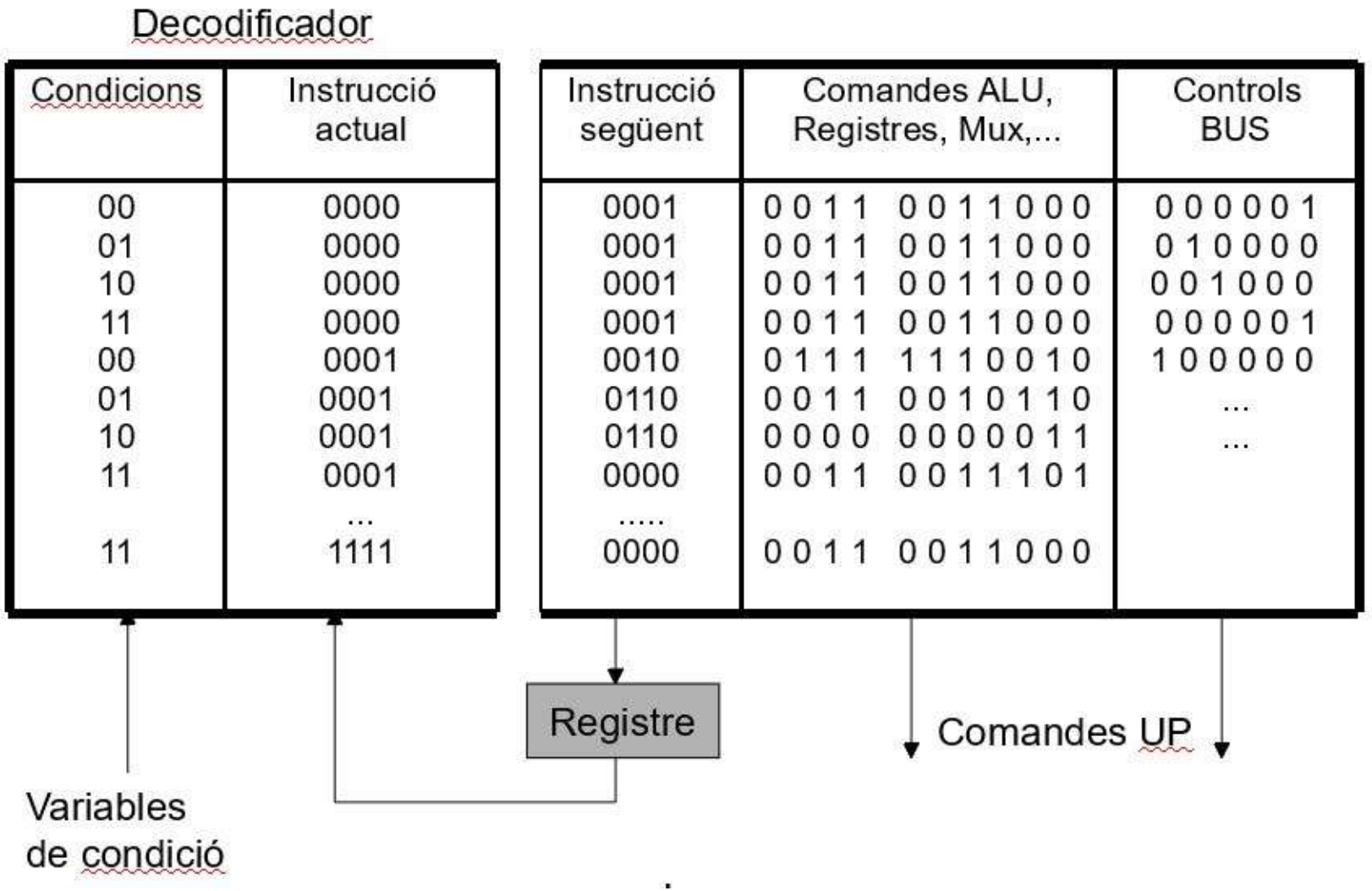
ROM i un registre d'estat.

Adreça constituïda per: variables d'estat actual i les variables d'entrada

Contingut de cada posició estat futur i operacions o variables de sortida



Unitats de control microprogramades



CPU La Unitat de Control (VIII)

- L'execució d'una instrucció no es pot realitzar en un sol ***step*** ja que sovint s'utilitza varies vegades el mateix recurs. L'execució es porta a terme en un conjunt de fases que es consideren operacions elementals o **microinstruccions.** Cada una d'aquestes microinstruccions es realitza en un temps màxim d'un cicle de rellotge.

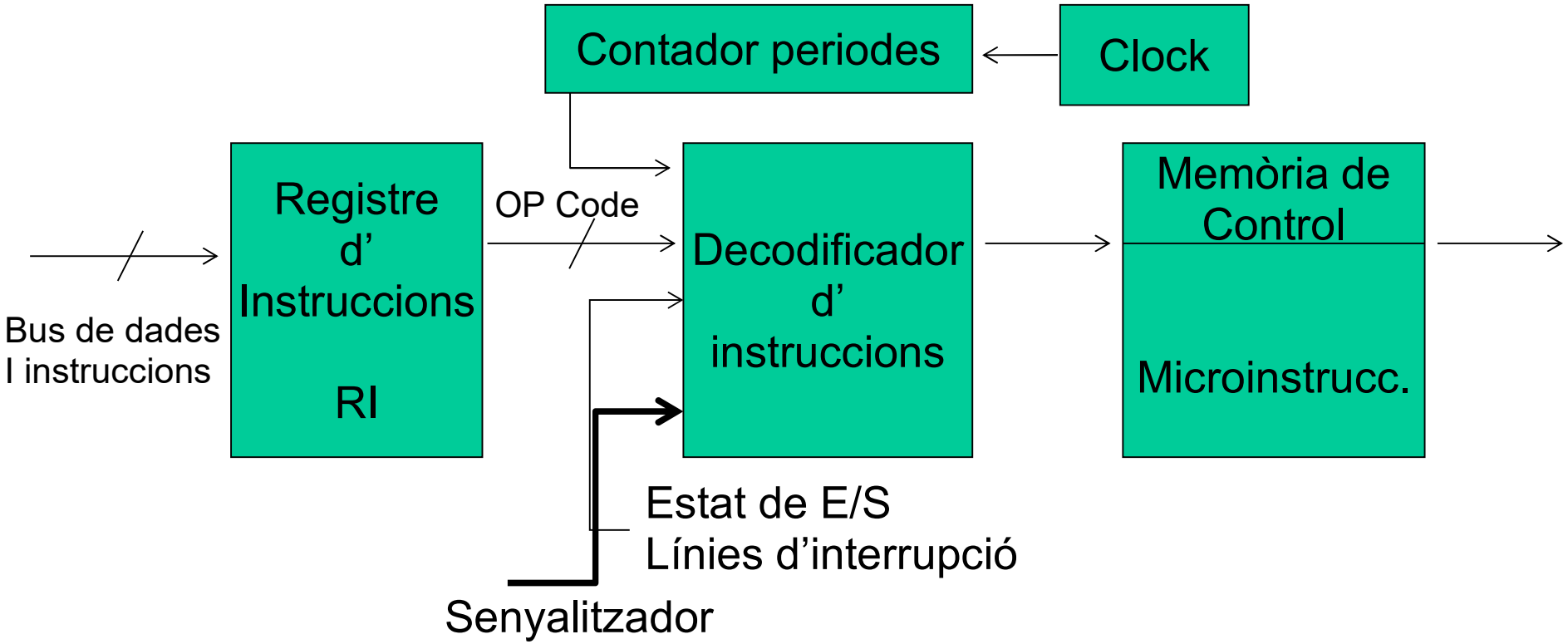
- Cada operació elemental requereix que es generin els nivells lògics que controlen els dispositius del computador. Aquesta informació binària està gravada en una memòria ROM anomenada **memòria de Control**

- El **descodificador d'instruccions** selecciona les posicions de la memòria de control que corresponen a les operacions elementals que formen la instrucció en fase d'execució. Finalment cal definir la **seqüencia** encarregada d'executar les microinstruccions i distribuir les senyals de control als diferents elements del sistema

CPU La Unitat de Control (IX)

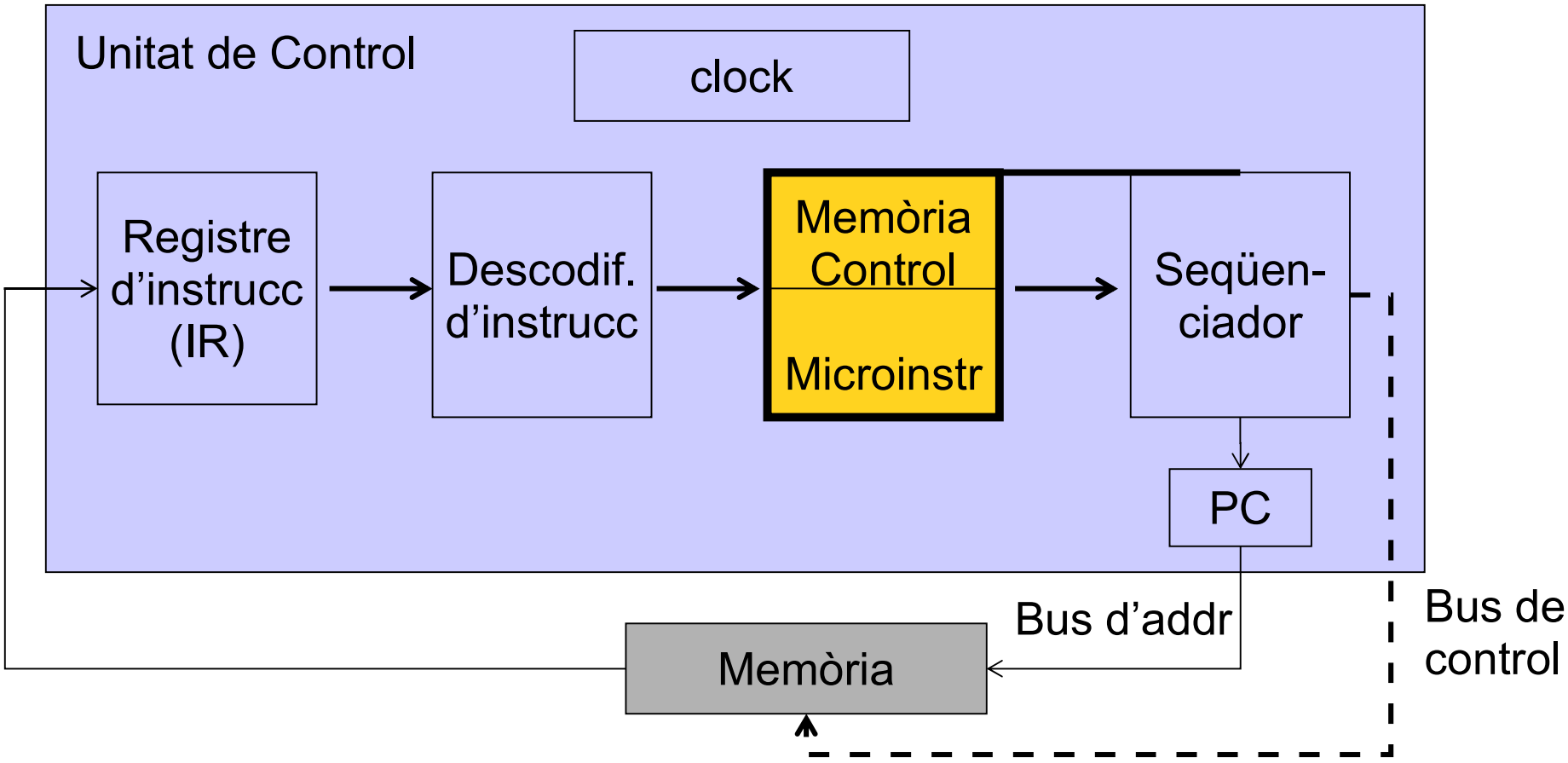
El Descodificador d'Instruccions utilitza la següent informació:

- Codi d'operació procedent del RI
- Estat dels senyalitzadors
- Nº de microinstruccions ja realitzades
- Estat de les línies d'interrupció + E/S



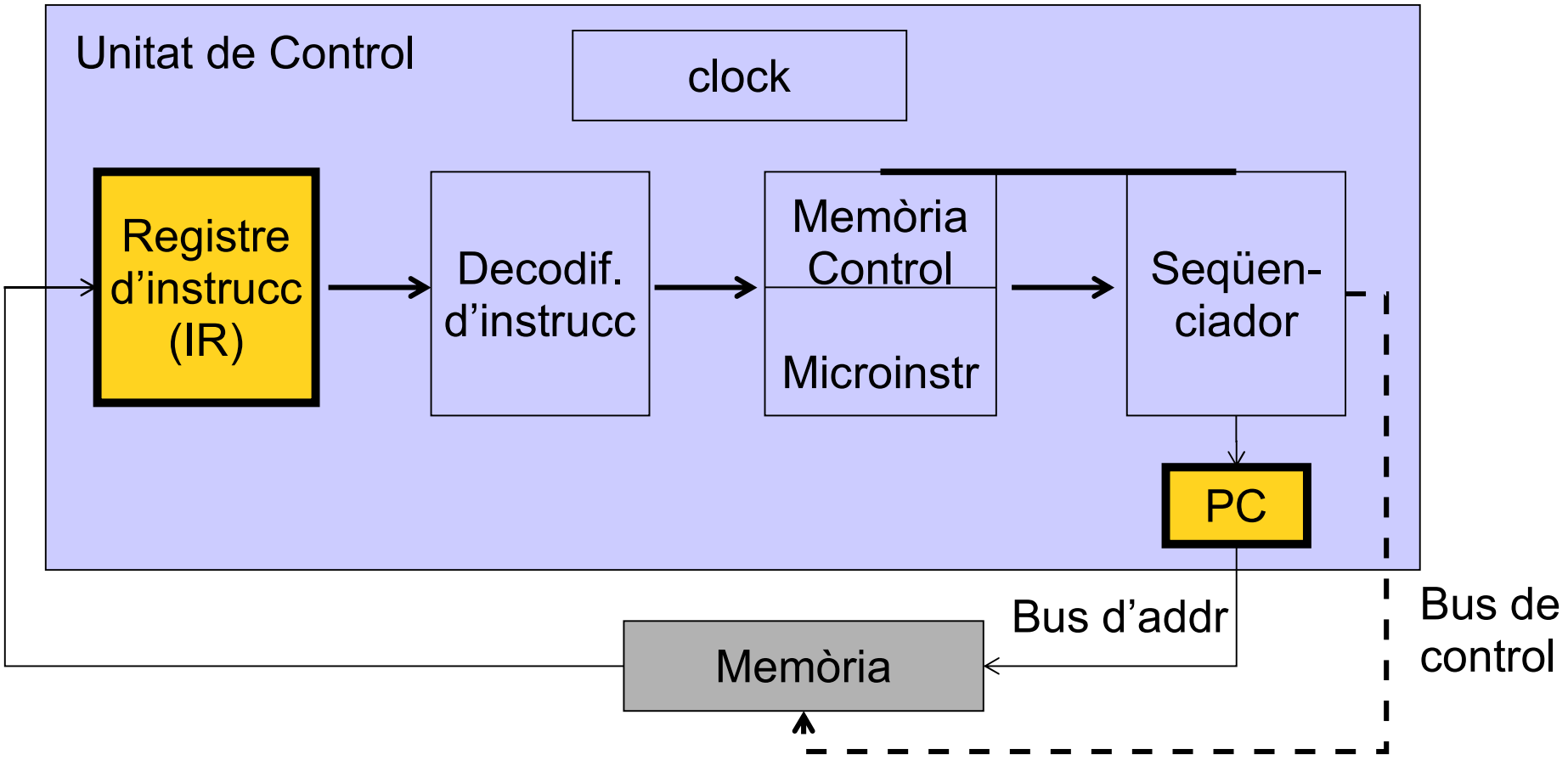
CPU La Unitat de Control (X)

· Cada operació elemental requereix que es generin els nivells lògics que controlen els dispositius del computador. Aquesta informació binaria es troba gravada a la memòria de control (ROM)



CPU La Unitat de Control (XI)

- El PC enviarà la direcció de memòria on es troba el codi de la instrucció a executar, l'OPCode es guardarà en el IR. Amb aquest es descodificarà la instrucció i es buscaran les posicions associades a la memòria de control. Els bits de les posicions de la següent ul es depositen en el seqüenciador.



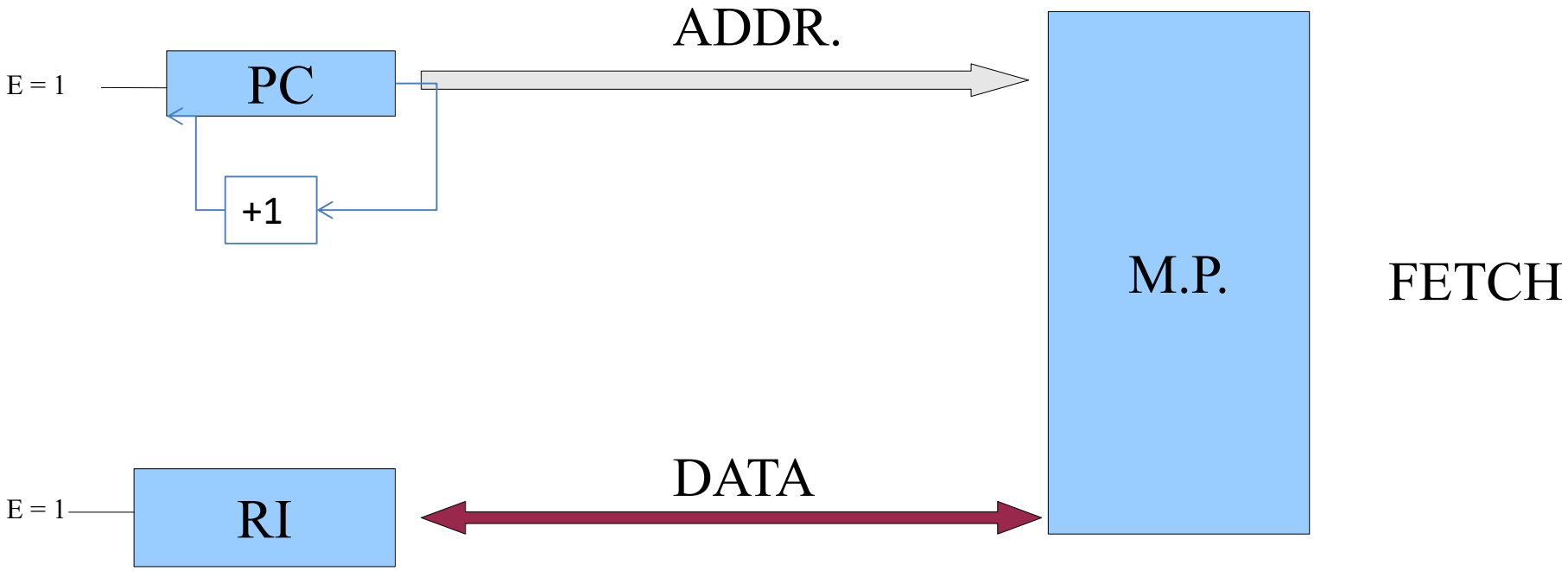
Fases d'execució de una instrucció (genèrica)

- FETCH ó càrrega de la instrucció — FETCH

- DESCODIFICACIÓ de la instrucció
- LECTURA DE OPERANDS
- EXECUCIÓ i/o guardar resultats

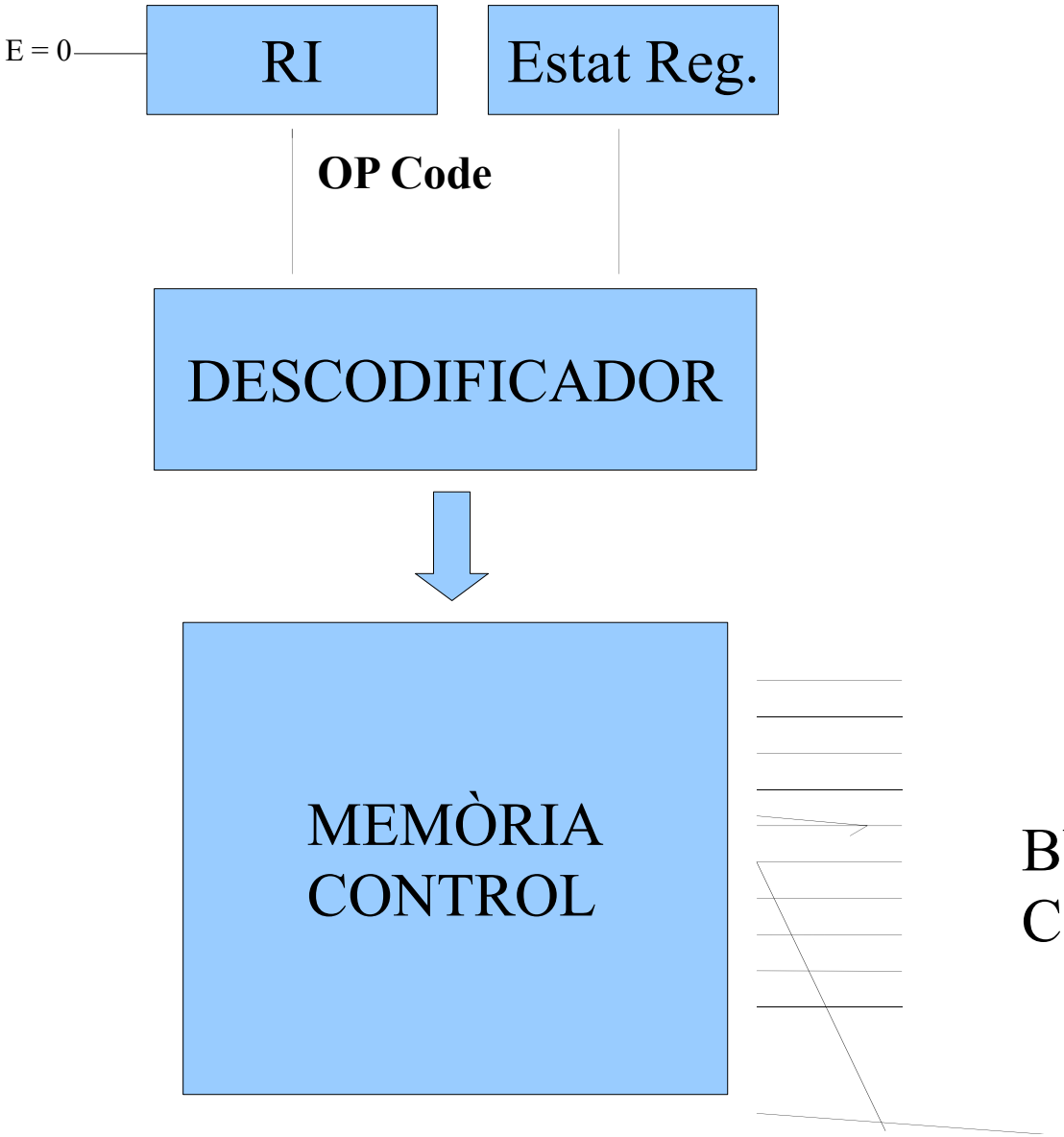
— EXECUCIÓ

CPU La Unitat de Control (XI)



1 cicle de clk

CPU La Unitat de Control (XI)



Fases posteriors:
Decod.
Exec. Exec 1er Op
Exec 2on Op
Exec. Instr.
...

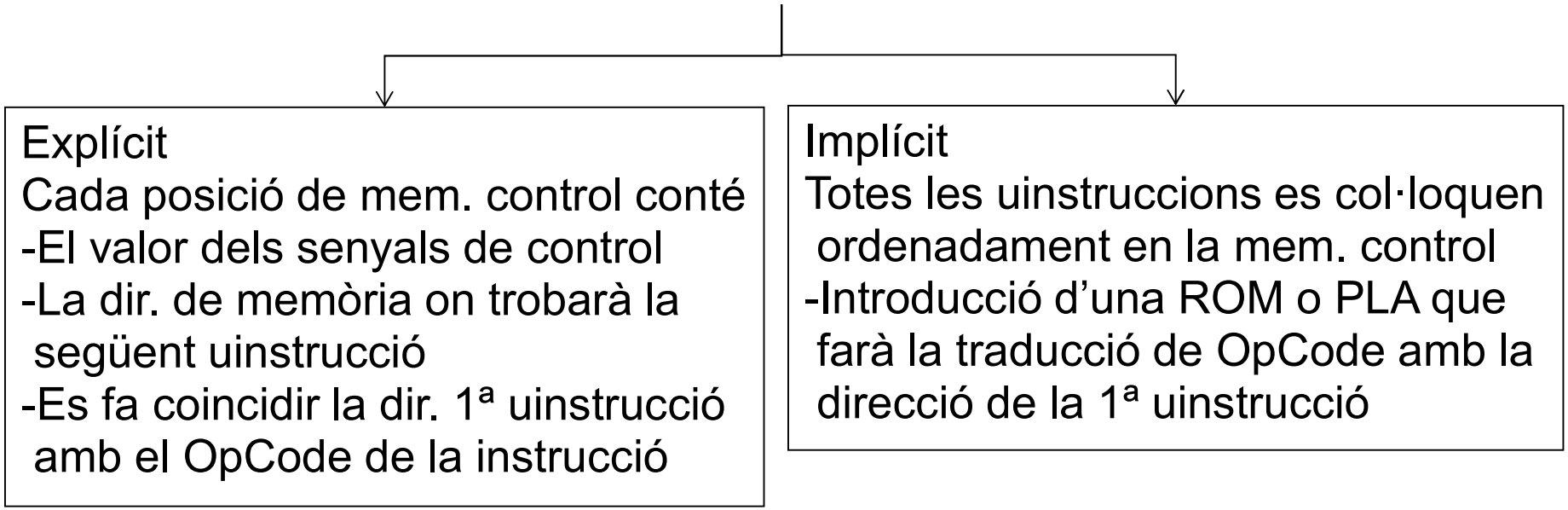
BUS DE
CONTROL

CPU La Unitat de Control (XI)

La unitat de control té funcionalment dues tasques:

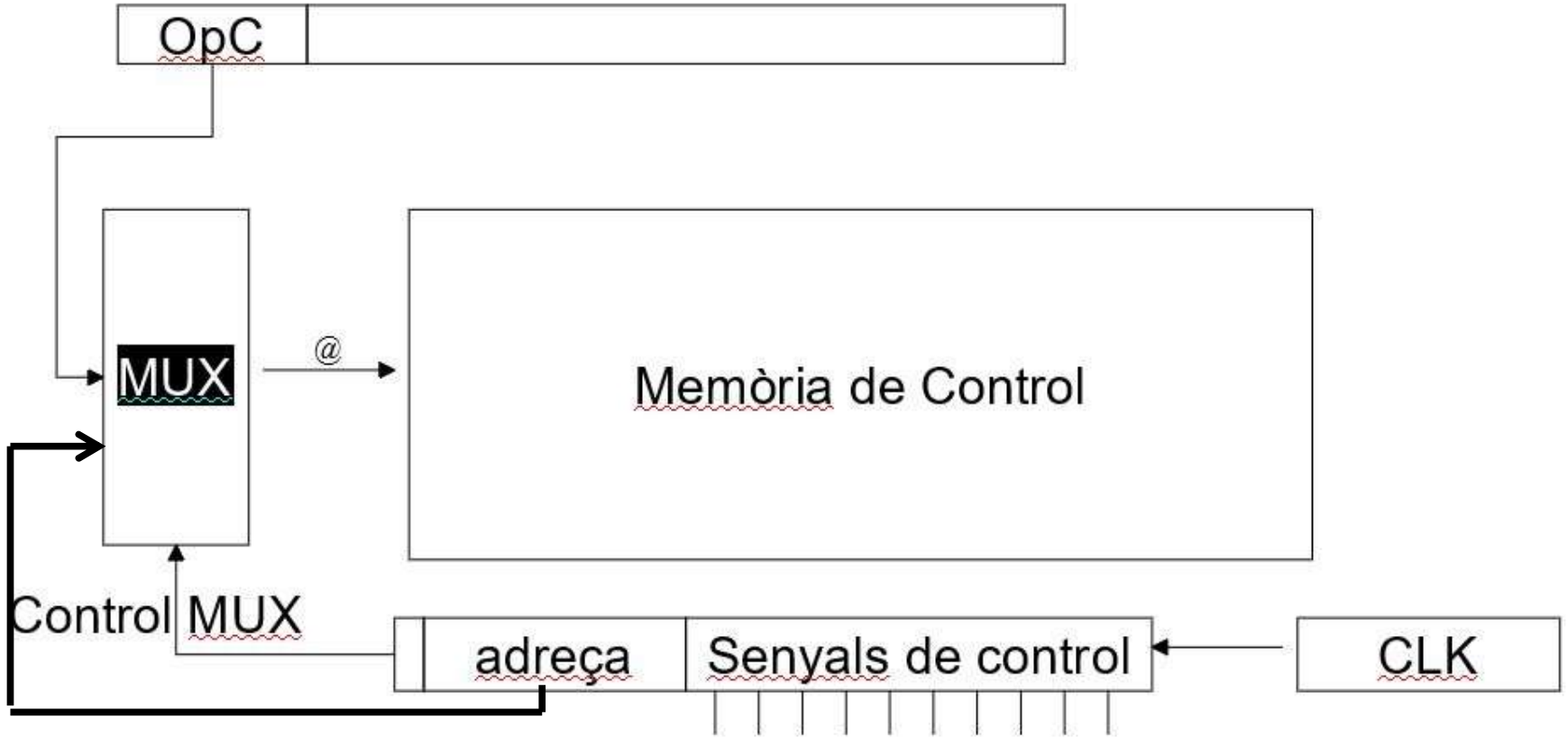
1.- Traduir el codi màquina de cada instrucció al microprograma

2.- Llegir successivament les microinstruccions de un microprog. (Seqüenciador)



•Seqüenciament Explícit (Wilkes 1949)

- Les microinstruccions poden estar desordenades
- No obliga a cap mecanisme de seqüènciament ja que cada instrucció suministra la nova adreça



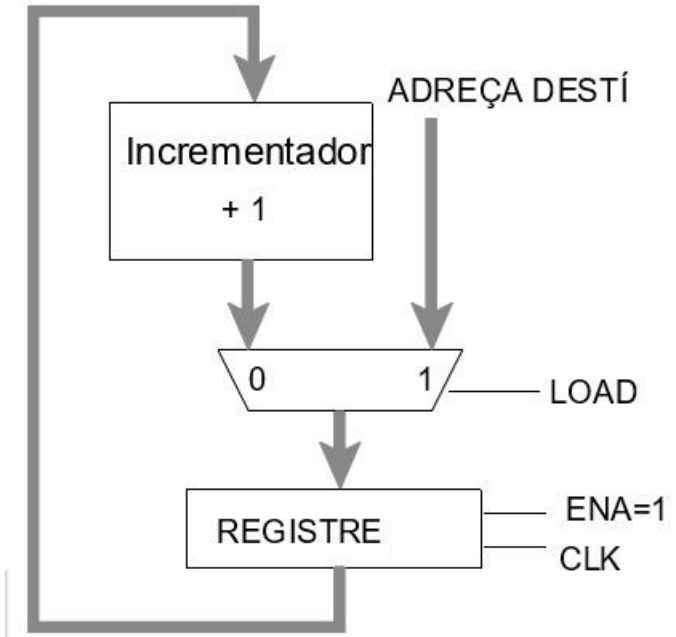
Seqüenciador Implícit

En les UC ***explícites*** el 1r camp és l'adreça d'instrucció següent. Generalment UC=seqüències +o- llargues d'instruccions sense bifurcació, junt amb instruccions amb certa probabilitat de bifurcació. Així majoritàriament:

$$y^{n+1} = y^n + 1$$

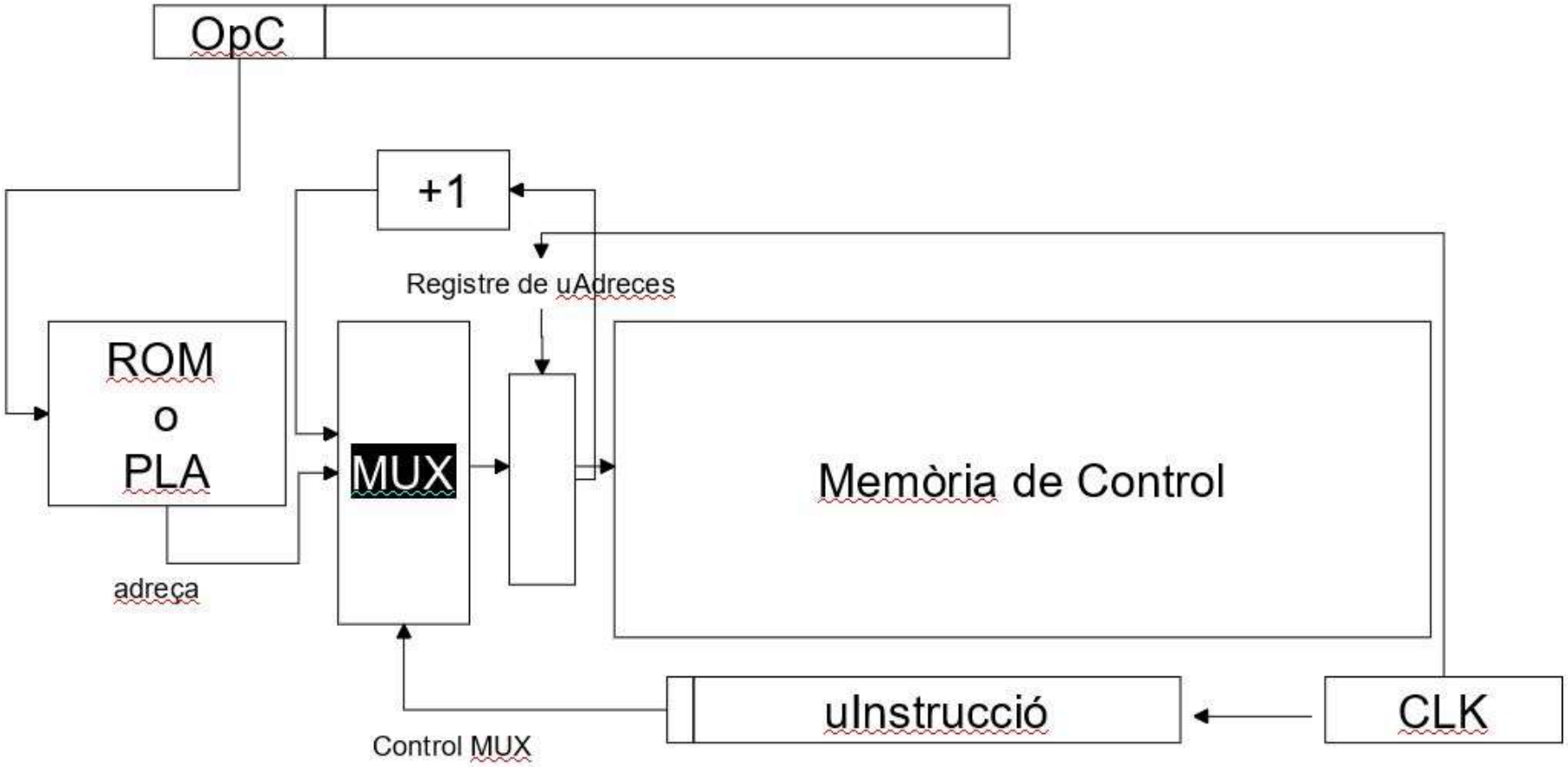
Estalvi en mida de ROM si el control d'anar a les uinstruccions següents el fa una unitat específica i utilitzem, en aquestes instruccions, l'espai d'adreça futura per altres comandes.

Aquesta unitat rep el nom de **Seqüenciador**.



No bifurcació: LOAD=0
Bifurcació: LOAD=1

Seqüenciador Implícit



CPU La Unitat de Control (XIII)

Amb seqüenciador explícit

Instrucció Microprograma

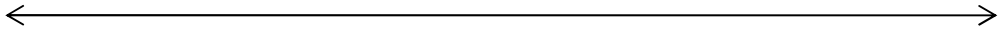
Adreça propera	Senyals de control d'ALU i Registres	Control BUS Dades	Senyals Control memòria

Amb seqüenciador implícit

- Tipus d'instrucció:
- Instruccions EXE (nova adreça=adreça següent) => Seqüenciador
- Instruccions de salt (nova adreça=adreça de salt)

Reducció mida memòria ROM

Codi de Tipus instrucció	Adreça o Comandes	Senyals Control memòria
Tipus Instrucció EXE	Comandes de control ALU, Registres, BUS	
Tipus Instrucció SALT	Condició	Adreça de salt	Senyals Control BUS i memòria



Línia de la ROM

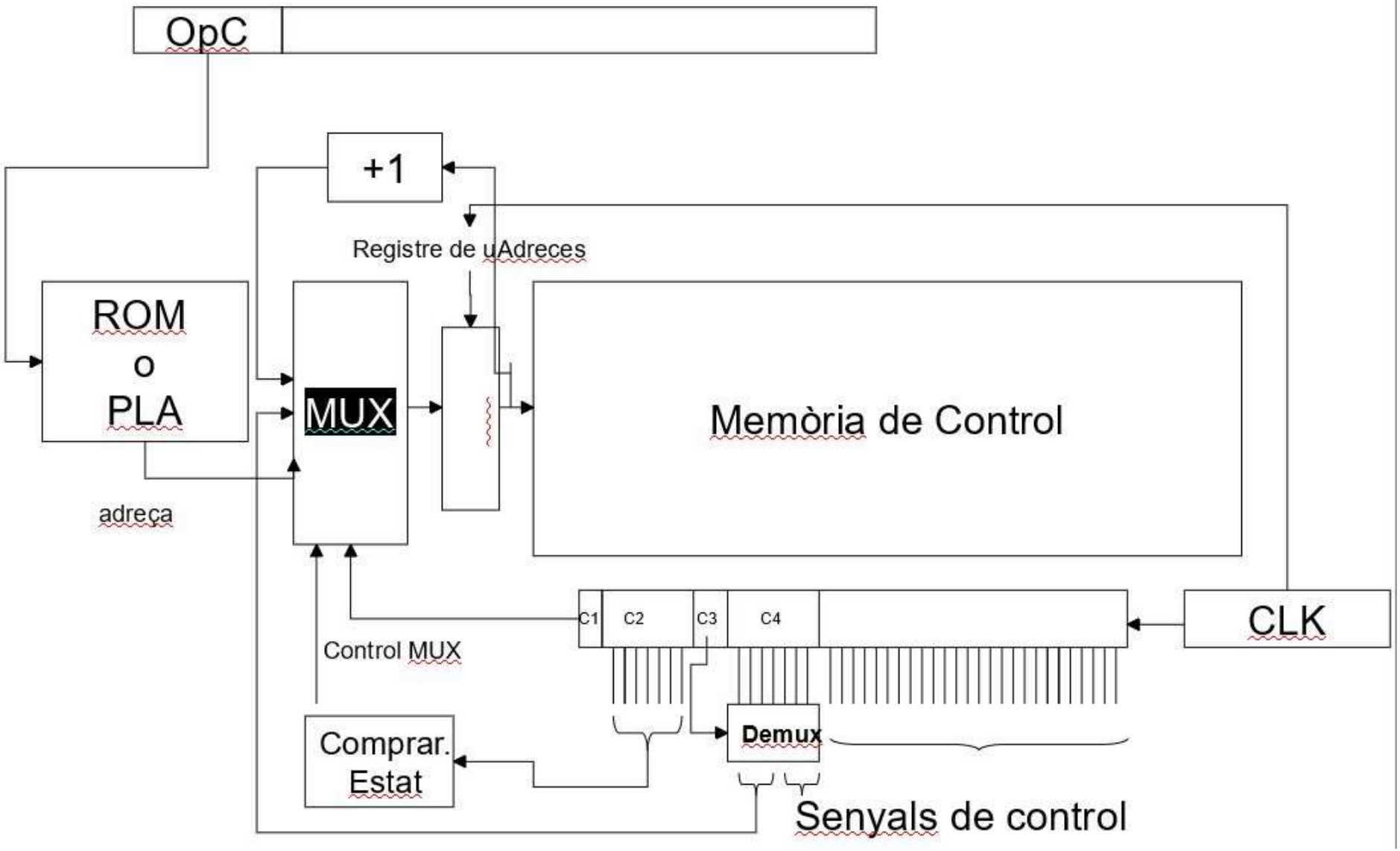
•Microbifurcacions condicionals

Planteja la possibilitat d'escollir entre dos adreces per tal d'anar per dos camins alternatius.

En seq. Explícit: La uInstrucció haurà de tenir dos adreces completes. Molt costós. El que es fa es que les dues adreces variïn només en 1 bit. Aquest bit pren el valor del resultat del comparador, obtenint les dues adreces.

En seq. Implícit: S'ha de poder escollir entre la uInstrucció següent o una diferent. La uinstrucció de bifurcació ha de tenir la nova adreça. Aquest és un cas adequat per a l'ús de camps solapats, donat que només cal quan tenim uinstruccions de bifurcació

CPU La Unitat de Control (XIII)



- El bit resultat del comparador es fa servir per escollir la següent uinstrucció al mux d'adreces. Bé l'anterior + 1, bé la proporcionada pel camp C4
- El camp C2 estableix la condició de bifurcació
- C1 indica si continuem executant la mateixa instrucció o passem a una nova
- C3 permet "des-solapar" el camp C4, fent servir un demux de dos sortides

CPU Un senzill exemple...

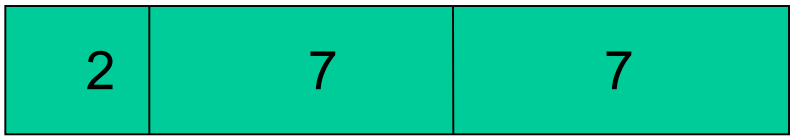
Suposem que volem dissenyar una màquina simple (MS) que executi les següents instruccions:

-Suma: ADD Font, Destí: guarda el resultat en el destí
-Mou: MOV Font, Destí: Col·loca el que té a la font a la direcció destí
-Compara: CMP Font, Destí: FZ (senyalitzador)
1 Si F=D
0 si F<>D
-Salt condicional: BEQ D: Dependent del valor de FZ
FZ=1 -> PC<= D
FZ=0 -> PC = PC+1

Tenim una memòria de 128 POSICIONS de 16bits per el nostre disseny:

CPU Un senzill exemple...

Co1	Co0	Instrucció
0	0	ADD
0	1	CMP
1	0	MOV
1	1	BEQ



OP. Code Dir. Font Dir. Destí

El nostre disseny consta de 2 registres (A, B) que no es poden manipular per software on es carreguen els valors de les direccions de memòria per tal de fer les operacions aritmètiques (ADD) i lògiques (CMP)

CPU Un senzill exemple...

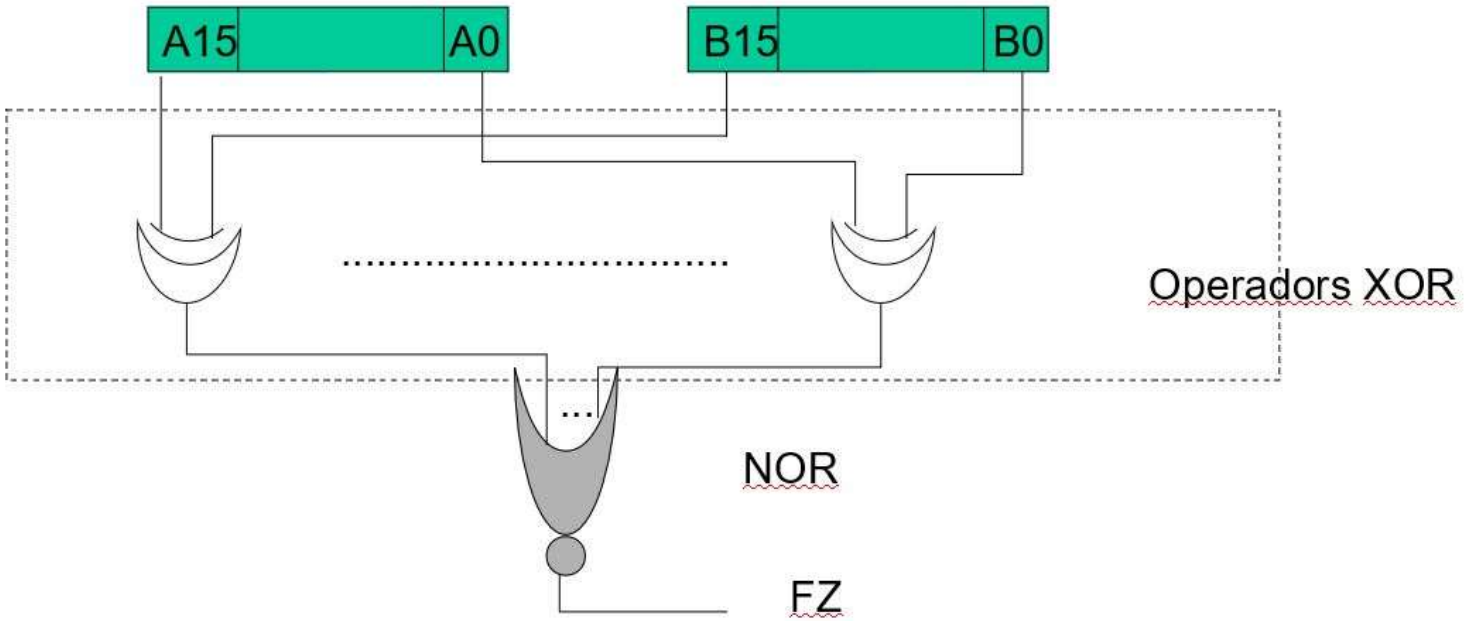
Nº de registres totals de que disposem:

IR: Amb un camp de dos bits que indica el codi d'op.
7 bits per la direcció de memòria Font i 7 bits per la direcció de memòria Destí
PC: Program Counter
Registre general A
Registre general B

A més tindrem un bit que ens indicarà si s'ha produït un zero en una operació

CPU Un senzill exemple...

La unitat Aritmètico-Lògica pot fer tant comparacions com sumes.
La comparació...



CPU Un senzill exemple...

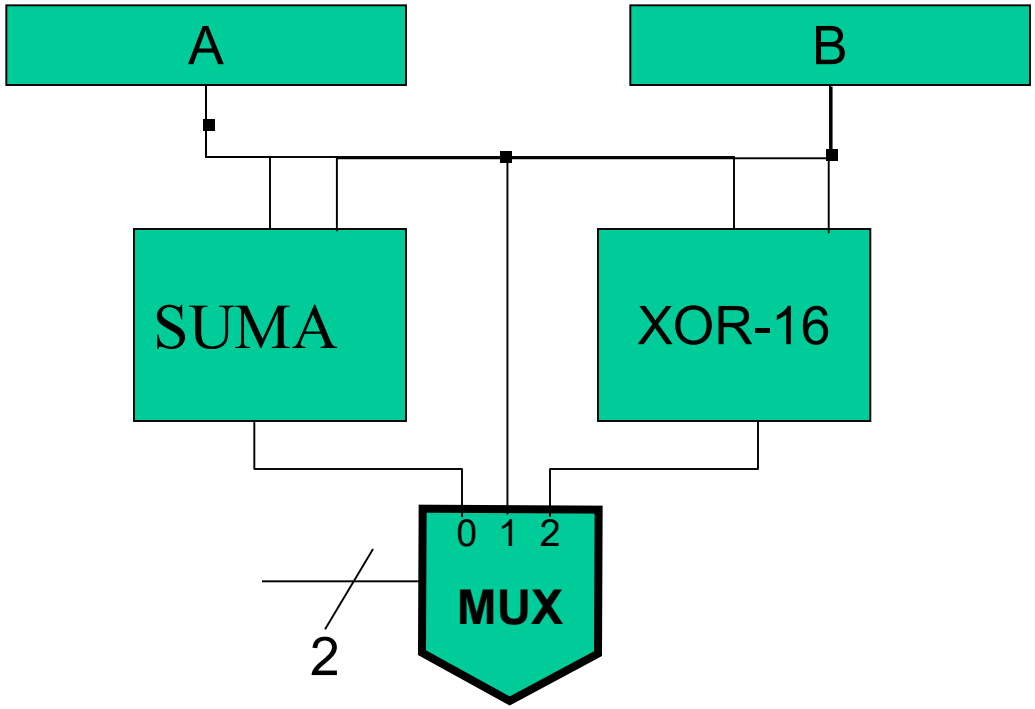
...i la suma...

Ai	Bi	Ci-1	Si	Ci
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

D'on es treu que...
Si = Ai XOR Bi XOR Ci-1
Ci = Ai AND Bi OR Ai AND Ci-1 OR Bi AND Ci-1

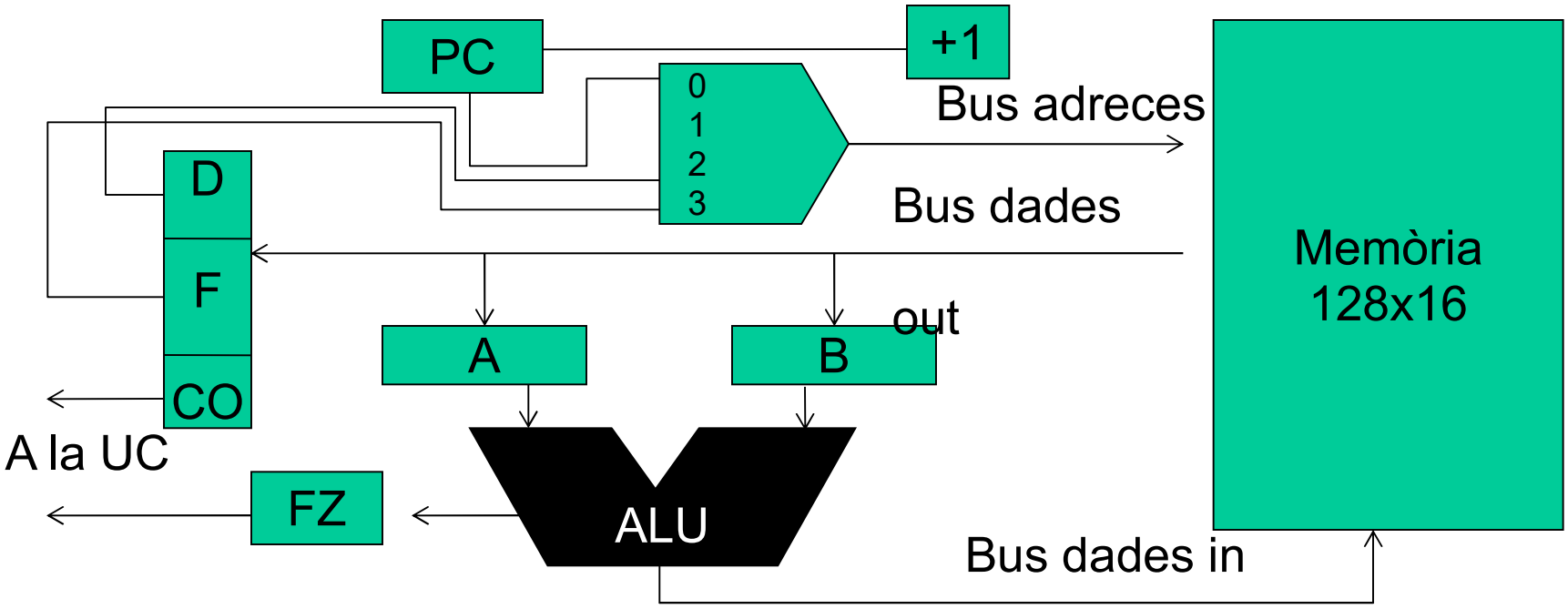
CPU Un senzill exemple...

...per tant la ALU de l'exercici serà...



CPU Un senzill exemple...

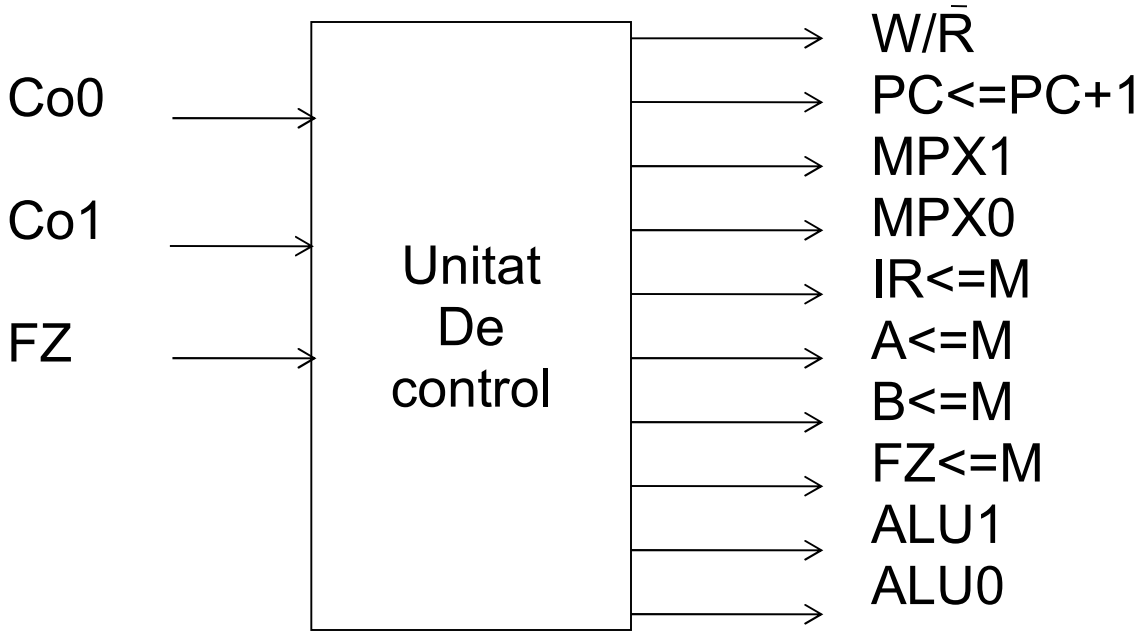
Incorporem també el registre “Program Counter” (PC) per tal d’accedir a la memòria (fent servir el bus d'adreces). A més, podem accedir a la memòria a partir de l'adreça Font i de l'adreça Destí carregada al IR (implica multiplexar).
La unitat de procés quedarà doncs de la següent forma:



CPU Un senzill exemple...

...queda per definir la Unitat de Control

El nostre disseny consta de 3 BUSos:
· Bus d'adreces: Unidireccional, 7 línies (128 posicions)
· Bus de dades: 16 bits, bidireccional
· Bus de control: Haurà de tenir les següents línies:



CPU Un senzill exemple...

Amb els bits Co i FZ la U.C. ha de conèixer quina instrucció ha d'executar

Cada instrucció constarà d'una sèrie de microinstruccions

Cada microinstrucció s'executa en un cicle de rellotge

En aquest cicle de clock la UC proporciona 10 senyals de sortida que regulen el comportament de la màquina

CPU Un senzill exemple...

Totes les instruccions s'executen seguint unes determinades fases d' execució:

1.- Fase de búsqueda (FETCH)

Es produeix a l'inici de cada instrucció. Comú a totes les instruccions.
El PC deposita el seu contingut al bus d'adreces.
Es procedeix a llegir la posició adreçada y el contingut es carrega en el IR.

MPX1=MPX0=W/R=0
M => IR
PC<=PC+1

CPU Un senzill exemple...

2.- Fase de Decodificació:
A partir del Op. Code de IR + altres informacions la UC decidirà la instrucció a executar i les microinstruccions que la formen

Analitzem els 2 bits de CO
Analitzem el bit de FZ

3.- Fase de Búsqueda d'operands:
Es llegeix de la memòria els operands que participen en la operació

4.- Execució i enmagatzenament del resultat:
S'executa la instrucció i es guarda el resultat en memòria si es cal

CPU Un senzill exemple...

Instrucció ADD

Estat	Fase	Operació
0	1	M(PC)=>IR; PC+1=>PC
1	2	Decod. Co0 i Co1
2	3	M(F)=>B
6	3	M(D)=>A
7	4	A+B=>M(D); Z =>1

Instrucció CMP

Estat	Fase	Operació
0	1	M(PC)=>IR; PC+1=>PC
1	2	Decod. Co0 i Co1
3	3	M(F)=>B
8	3	M(D)=>A
9	4	A XOR B; Z=1

Instrucció MOV

Estat	Fase	Operació
0	1	M(PC)=>IR; PC+1=>PC
1	2	Decod. Co0 i Co1
4	3	M(F)=>B
10	4	B=>M(D); Z=>1

Instrucció BEQ

Estat	Fase	Operació
0	1	M(PC)=>IR; PC+1=>PC
1	2	Decod. Co0 i Co1
5	3	Eval. De FZ
11	4	Si FZ=1 => D+1 =>PC

CPU Un senzill exemple...

	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	
MPX1	0		X		1	1	1	X	1	1	1	X	1	1
MPX0	0		X		0	0	0	X	1	1	1	X	1	1
ALU1	X		X		X	X	X	X	X	0	X	0	1	X
ALU0	X		X		X	X	X	X	X	0	X	1	0	X
W/R	0		0		0	0	0	0	0	1	0	0	1	0
PC+1	1		0		0	0	0	0	0	0	0	0	0	1
M=>IR	1		0		0	0	0	0	0	0	0	0	0	1
M=>A	0		0		0	0	0	0	1	0	1	0	0	0
M=>B	0		0		1	1	1	0	0	0	0	0	0	0
Z=>FZ	0		0		0	0	0	0	0	1	0	1	1	0

CPU Un senzill exemple...

	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	
MPX1	0		X		1	1	1	X	1	1	1	X	1	1
MPX0	0		X		0	0	0	X	1	1	1	X	1	1
ALU1	X		X		X	X	X	X	X	0	X	0	1	X
ALU0	X		X		X	X	X	X	X	0	X	1	0	X
W/R	0		0		0	0	0	0	0	1	0	0	1	0
PC+1	1		0		0	0	0	0	0	0	0	0	0	1
M=>IR	1		0		0	0	0	0	0	0	0	0	0	1
M=>A	0		0		0	0	0	0	1	0	1	0	0	0
M=>B	0		0		1	1	1	0	0	0	0	0	0	0
Z=>FZ	0		0		0	0	0	0	0	1	0	1	1	0

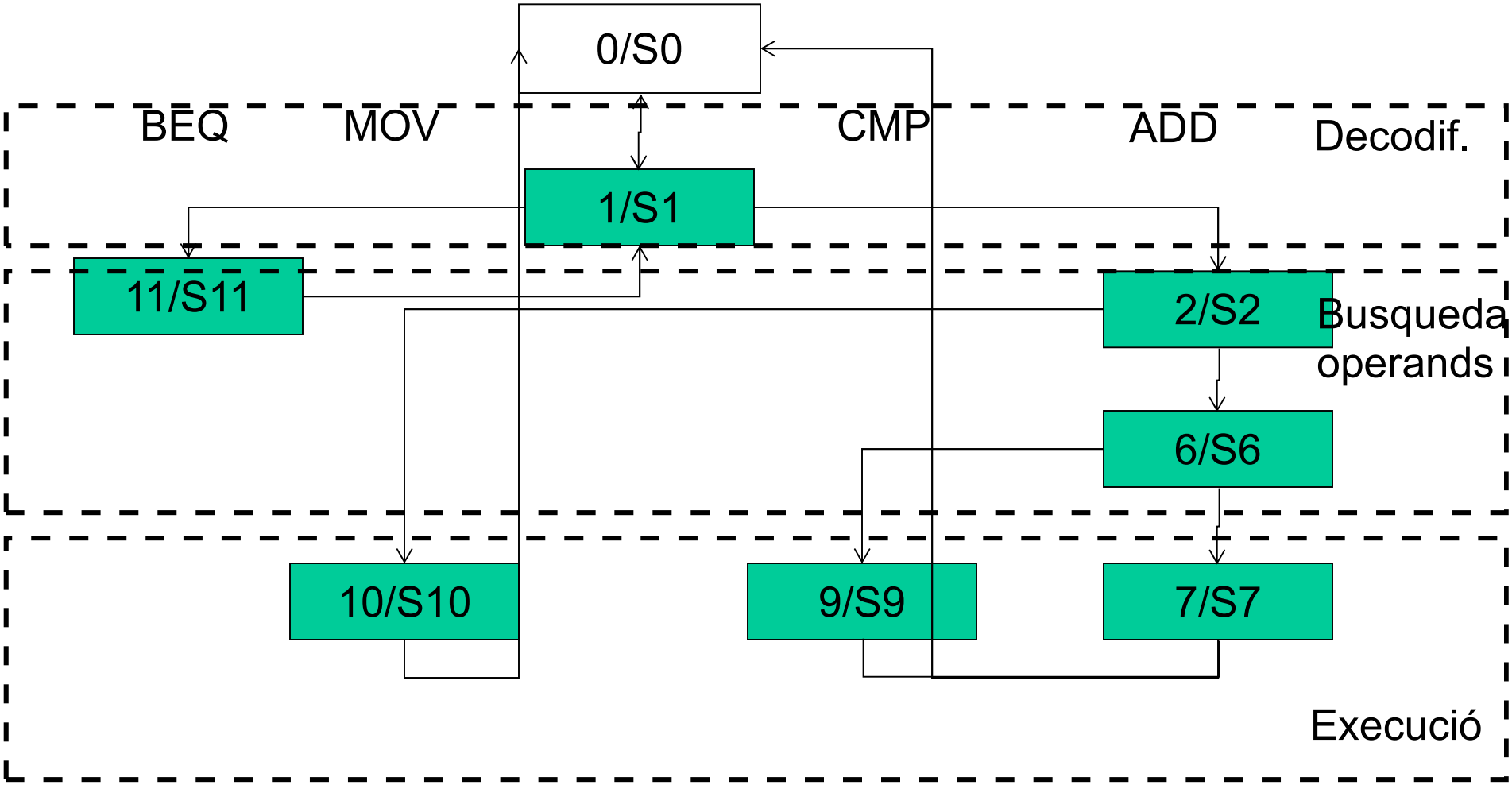
Iguals!!!

Iguals!!!

Iguals!!!

8 estats diferents

CPU Un senzill exemple...



CPU Un senzill exemple...

Microprograma corresponent a la instrucció ADD

Estat	Fase	Operació	MPX1	MPX0	ALU1	ALU0	W/R	PC+1	IR	A	B	FZ
S0	1	M(PC)=>IR	0	0	X	X	0	1	1	0	0	0
S1	2	Decodif.	X	X	X	X	0	0	0	0	0	0
S2	3	M(F)=>B	1	0	X	X	0	0	0	0	1	0
S6	3	M(D)=>A	1	1	X	X	0	0	0	1	0	0
S7	4	A+B=>M(D)	1	1	0	0	1	0	0	0	0	1

CPU Un senzill exemple...

Seqüenciador a la MS:

Tenim 8 estats, cada un amb 10 senyals de control.
Precisem d'una memòria de 8 posicions de 10bits cada una

Per especificar la direcció ROM necessitem de 3 bits:
D'0, D'1 i D'2
Per determinar en cada cicle de rellotge la posició a llegir s'ha de tenir en compte :

1.- L'estat anterior definit per la direcció prèvia de la ROM (D0, D1 i D2)
2.- Els dos bits de l'OpCode
3.- El valor del FZ

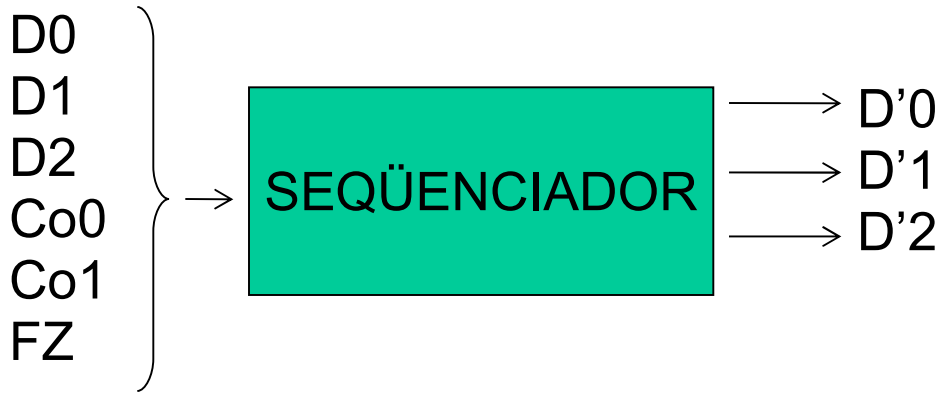
CPU Un senzill exemple...

Estat/Direcció	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
(s0)@ 0	X	x	X	X	X	X	0	0	x	x	0	1	1	0	0	0
(s1) @ 1	X	X	X	X	X	X	x	x	x	x	0	0	0	0	0	0
(s2) @ 2	X	X	X	X	X	X	1	0	x	x	0	0	0	0	1	0
(s3) @ 3	X	X	X	X	X	X	1	1	0	0	1	0	0	0	0	1
(s4) @ 4	X	X	X	X	X	X	1	1	x	x	0	0	0	1	0	0
(s5) @ 5	X	X	X	X	X	X	x	x	0	1	0	0	0	0	0	1
(s6) @ 6	X	X	X	X	X	X	1	1	1	0	1	0	0	0	0	1
(s7) @ 7	x	X	x	X	X	X	1	1	x	x	0	1	1	0	0	0

A cada posició de memòria s'assigna un estat

CPU Un senzill exemple...

...Per tant, a partir d'unes determinades condicions, obtenim la posició següent de la microinstrucció a la memòria de control



D'0, D'1 i D'2 especifiquen la direcció de la ROM. Per determinar la posició a llegir s'ha de tenir en compte :

1. L'estat anterior, que ve donat per la direcció prèvia de la ROM
2. Els dos bits que ens indiquen el codi de la instrucció
3. El factor de senyalització FZ

CPU Un senzill exemple...

Donat que el seqüenciador és un circuit combinacional,el seu disseny ha de seguir les fases de qualsevol circuit d'aquest estil:

1 . .- Confecció de la taula de veritat
2 . .- Obtenció de les equacions lògiques de la taula
3 . .- Simplificació de les equacions
4 . ..- Implementació de les equacions amb portes lògiques

CPU Un senzill exemple...

Recordem...

Co1	Co0	Instrucció
0	0	ADD
0	1	CMP
1	0	MOV
1	1	BEQ

Entrades						Sortides		
Estat present			Senyals de entrada			Estat següent		
D2	D1	D0	C1	C0	FZ	D'2	D'1	D'0
0	0	0 (S0)	x	x	x	0	0	1 (S1)
0	0	1 (S1)	0	x	x	0	1	0 (S2)
			x	0	X	0	1	0 (S2)
			1	1	0	0	0	0 (S0)
			1	1	1	1	1	1 (S11)
0	1	0 (S2)	0	x	x	0	1	1 (S6)
			1	x	x	1	1	0 (S10)
0	1	1 (S6)	x	1	X	1	0	1 (S9)
			x	0	x	1	0	0 (S7)
1	0	0 (S7)	x	x	x	0	0	0 (S0)
1	0	1 (S9)	x	x	x	0	0	0 (S0)
1	1	0 (S10)	x	x	x	0	0	0 (S0)
1	1	1 (S11)	x	x	x	0	0	1 (S1)

CPU Un senzill exemple...

- $D'0 = \overline{D2} \cdot \overline{D1} \cdot \overline{D0} + \overline{D2} \cdot \overline{D1} \cdot D0 \cdot C1 \cdot C0 \cdot FZ + \overline{D2} \cdot D1 \cdot \overline{D0} \cdot \overline{C1} + D2 \cdot D1 \cdot D0 \cdot C1 + D2 \cdot D1 \cdot D0$

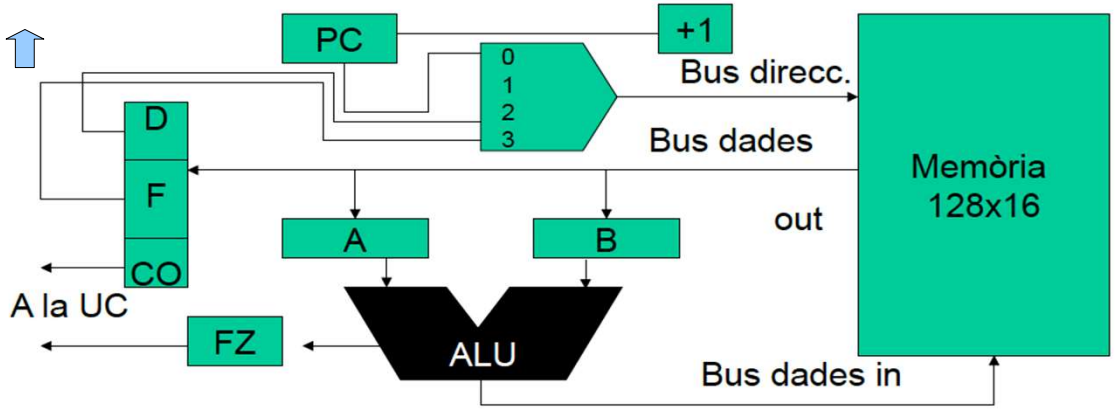
- $D'1 = \overline{D2} \cdot \overline{D1} \cdot D0 \cdot \overline{C1} + \overline{D2} \cdot \overline{D1} \cdot D0 \cdot \overline{C0} + \overline{D2} \cdot \overline{D1} \cdot D0 \cdot \overline{C1} \cdot C0 \cdot FZ + \overline{D2} \cdot D1 \cdot \overline{D0} \cdot \overline{C1} + D2 \cdot D1 \cdot \overline{D0} \cdot C1$

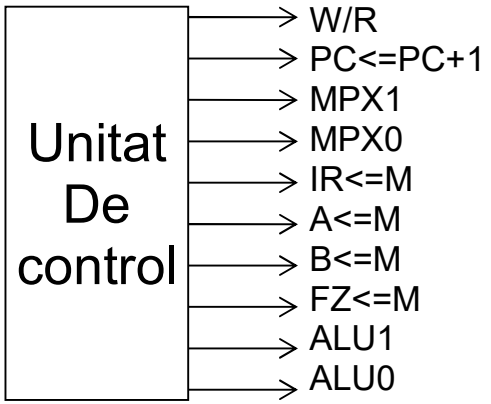
- $D'2 = \overline{D2} \cdot \overline{D1} \cdot D0 \cdot C1 \cdot C0 \cdot FZ + \overline{D2} \cdot D1 \cdot \overline{D0} \cdot C1 + \overline{D2} \cdot D1 \cdot D0 \cdot C0 + \overline{D1} \cdot C1 \cdot D0 \cdot \overline{C0}$

CPU Un senzill exemple

Entrades						Sortides		
Estat present			Senyals de entrada			Estat següent		
D2	D1	D0	C1	C0	FZ	D'2	D'1	D'0
0	0	0 (S0)	x	x	x	0	0	1 (S1)
0	0	1 (S1)	0	x	x	0	1	0 (S2)
			x	0	X	0	1	0 (S2)
			1	1	0	0	0	0 (S0)
			1	1	1	1	1	1 (S11)
0	1	0 (S2)	0	x	x	0	1	1 (S6)
			1	x	x	1	1	0 (S10)
0	1	1 (S6)	x	1	X	1	0	1 (S9)
			x	0	x	1	0	0 (S7)
1	0	0 (S7)	x	x	x	0	0	0 (S0)
1	0	1 (S9)	x	x	x	0	0	0 (S0)
1	1	0 (S10)	x	x	x	0	0	0 (S0)
1	1	1 (S11)	x	x	x	0	0	1 (S1)

Estat/Direcció	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
(s0)@ 0	X	x	X	X	X	X	0	0	x	x	0	1	1	0	0	0
(s1) @ 1	X	X	X	X	X	X	x	x	x	x	0	0	0	0	0	0
(s2) @ 2	X	X	X	X	X	X	1	0	x	x	0	0	0	0	1	0
(s3) @ 3	X	X	X	X	X	X	1	1	0	0	1	0	0	0	0	1
(s4) @ 4	X	X	X	X	X	X	1	1	x	x	0	0	0	1	0	0
(s5) @ 5	X	X	X	X	X	X	x	x	0	1	0	0	0	0	0	1
(s6) @ 6	X	X	X	X	X	X	1	1	1	0	1	0	0	0	0	1
(s7) @ 7	x	X	x	X	X	X	1	1	x	x	0	1	1	0	0	0





FINAL DE L'EXEMPLE

Unitat de Comunicació amb l'exterior

Permet connectar la CPU amb els elements externs:

- Memòria
- Interfícies Entrada/Sortida
- Connectats a través del BUS de CPU
- BUS de dades
- BUS d'adreces
- BUS de control
- Buffers sortida.
- Algun registre:
- MAR (MEMORY ADDRESS REGISTER)

Memòria principal

- Directament accessible per la CPU, (a diferencia de la secundaria)
- Memòria semiconductor (velocitat igual o relativament similar a CPU)
- Adreça especificada a BUS d'adreces

Memòries d'accés aleatori:
Adreça – Dada
Alguna escriptura – Múltiples lectures (**ROM**, PROM, EPROM, EEPROM)
Escriptura i lectura múltiples(**RAM** : SRAM, DRAM)

- Organitzada en bytes o words de diferent longitud (32b)
- N (línies d'adreça) => 2^N posicions de memòria

- Velocitat
- Temps d'accés
- Temps de cicle de memòria

Memòria principal

•La dividim en 2 grups (poden estar implementats en mateix dispositiu):
•**Memòria de programa**
•**Memòria de dades**

Memòria de programa,

•Conté el conjunt d'instruccions del programa a executar properament.
•Podem tenir 2 tipus de memòria on emmagatzemar els programes
•Memòria de lectura/escriptura (**RAM**) (solen posar-se els programes d'usuari)
•Memòria sols de lectura (**ROM**) (solen estar programes bàsics que necessita el computador per funcionament correcte (BIOS....)).

Memòria de dades

•Usada per la CPU per emmagatzematge temporal de dades.
•Aquesta memòria sempre ha de ser tipus lectura/escriptura (**RAM**).

RAM = volàtil
ROM = permanent

Interfície d'entrada/sortida (E/S)

- Permet la connexió del subsistema format per
  [CPU, memòria, elements connectats a BUS] amb l'exterior.
- Permet intercanvi d'informació entre aquests 2 'móns'.

- Necessitat d'aquest element:
- La naturalesa dels senyals utilitzats internament és molt diferent de la dels sistemes externs.
- Sistema intern controlat per una seqüència de temporitzacions molt estrictes, els elements externs no poden seguir directament. P.ex: monitor o impressora no poden treballar a = velocitat que CPU.
- Connexió de perifèrics per executar el programa; accés a dades externes; guardar resultats.

BUS de sistema

3 tipus de BUS agrupats segons les funcions

- **Bus de dades**, Transport d'informació dins del sistema,
- Instruccions i
- Dades que s'utilitzen durant l'execució d'aquestes instruccions.

- **Bus d'adreces**, indiquen on es troba o on ha d'anar la informació que viatjarà pel bus de dades.
- Indiquen una posició a memòria o en un dispositiu de E/S.

- **Bus de control**, Senyals que sincronitzen les transferències entre els diferents dispositius.

Altres elements

Perifèrics,

Connexions del computador amb el món exterior. Diverses funcions:

- Introduir dades: Teclats, discs, sensors amb plaques d'adquisició, etc.
- Treure dades: Impressores, pantalles, discos, etc.
- Introduir i gravar programes en discos.

Circuit de rellotge.

- Base de temps per sincronitzar funcionament dels elements del sistema
- En principi a major freqüència de rellotge major velocitat de processament dels programes
- Paràmetre que depèn molt de la tecnologia del sistema.
- Les prestacions no tant sols depenen de freq. de rellotge sinó també depenen d'organització de CPU i sistema. (Podem trobar processadors a freqüències menors amb prestacions superiors)

Exemple genèric

El funcionament d'un processador genèric (no tant senzill com el del exemple plantejat amb la MS) constarà d'altres registres:

MAR
MBR
Status

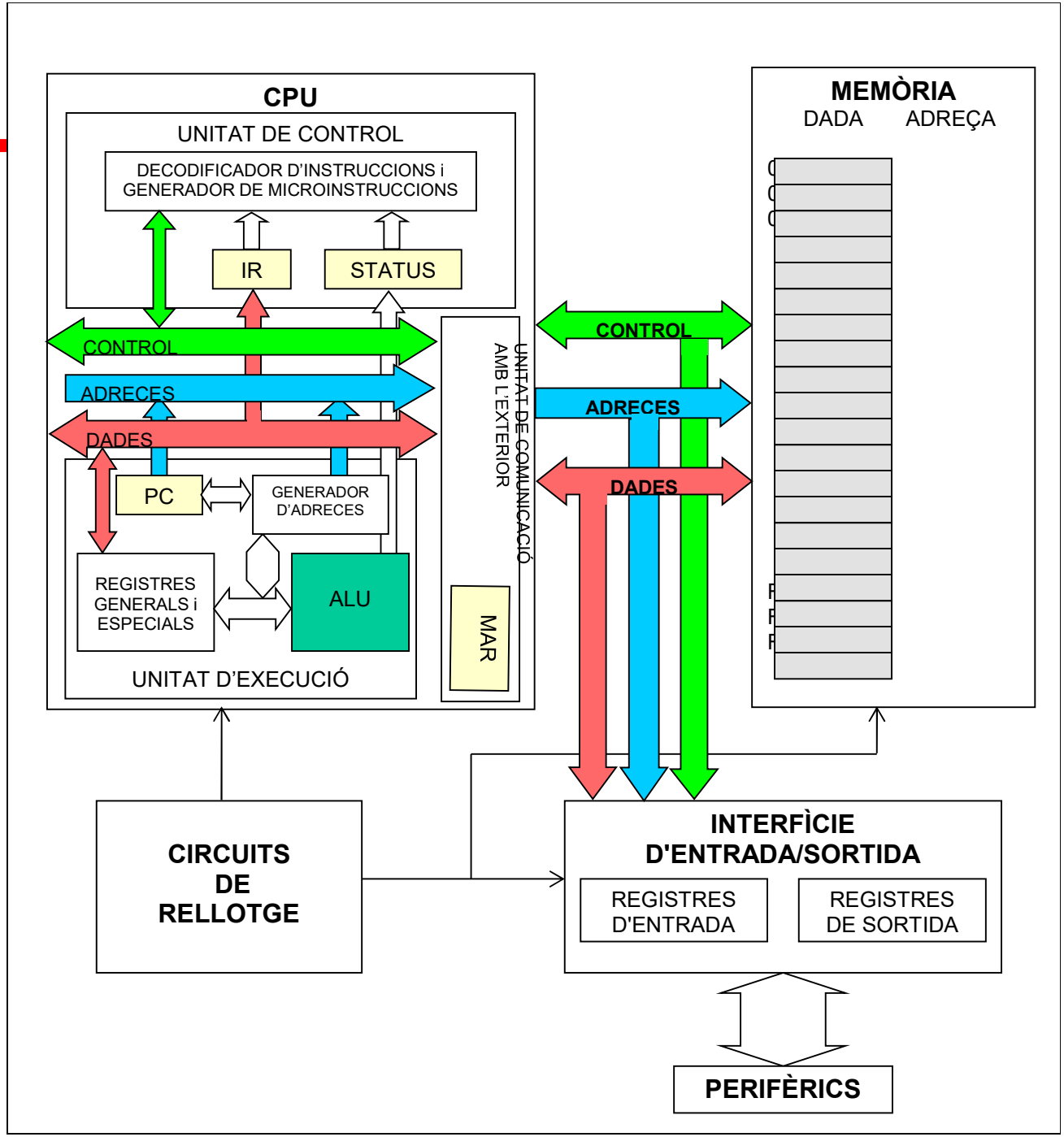
A més, haurà de tenir en compte les unitats d'entrada/sortida i d'altres registres genèrics que tenim a la CPU i als quals podem accedir i treballar.

Funcionament

CPU:
UC
UP (unitat d'execució)

Memòria Principal

E/S



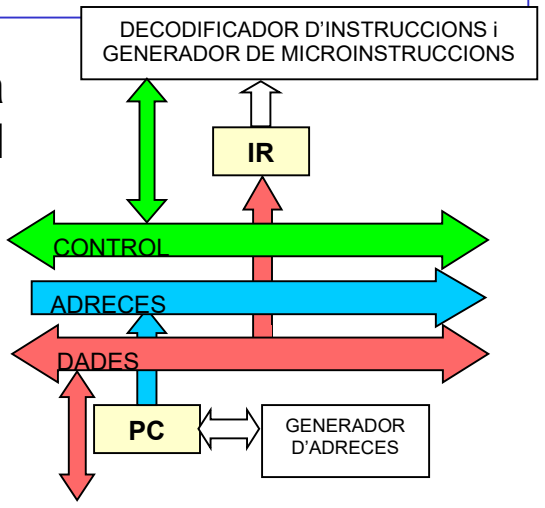
Fases d'una Instrucció

L'execució de cada instrucció consta dels següents passos o fases:

1. Fetch: Portar la instrucció de memòria a CPU
2. Decodificació: CPU mira les ordres que especifica la instrucció
3. Lect. d'Operands: CPU localitza les dades que precisa la instrucció
4. Execució: CPU fa les operacions indicades per instrucció
5. Guardar resultats: Es guarden, si cal, els resultats d'operacions

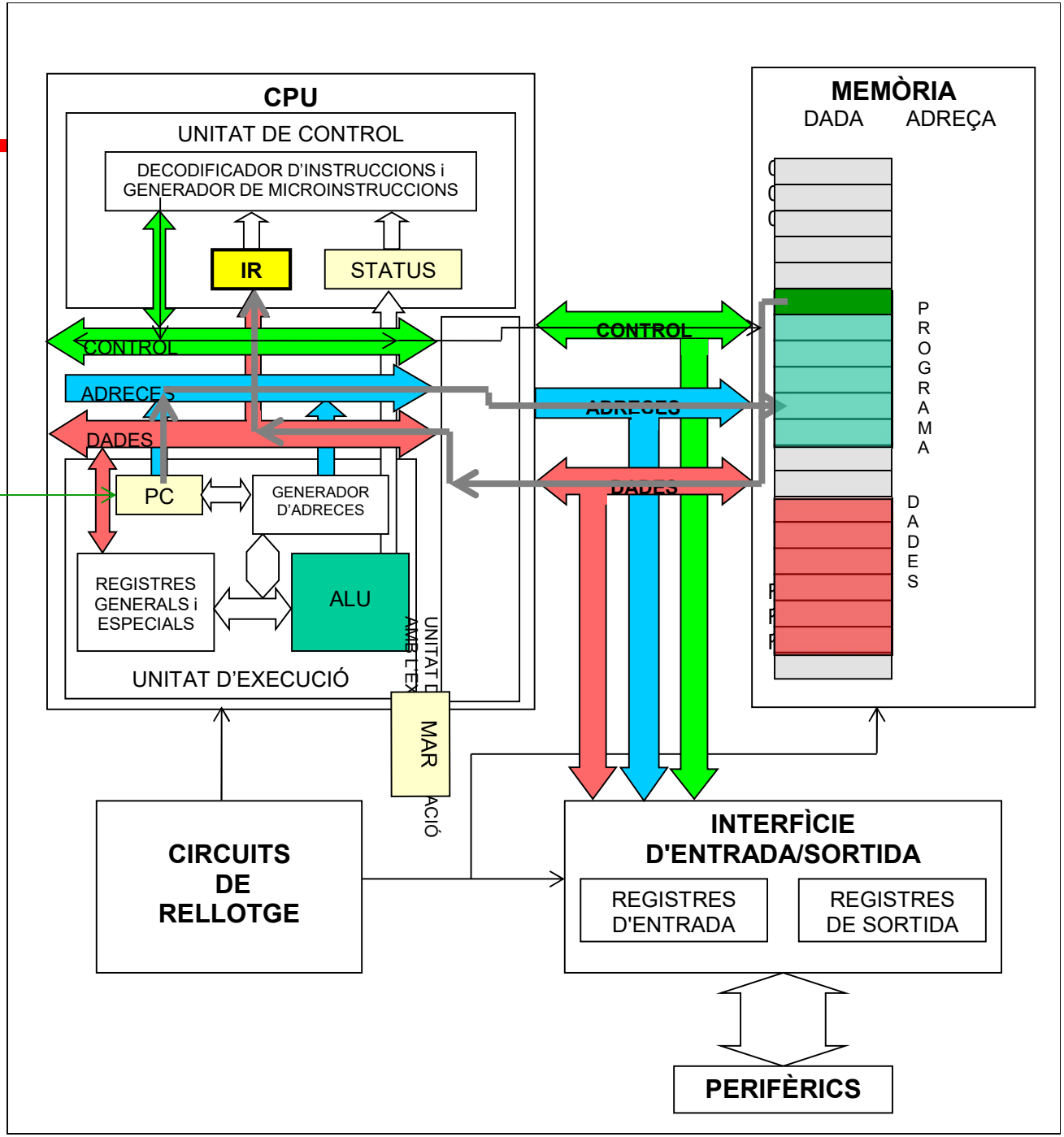
Aquests passos són els típics i es repeteixen a cada instrucció, successivament fins a completar l'execució del programa

El PC (comptador de programa) indica la posició de memòria de la propera instrucció: següent o salt. S'autoincrementa.



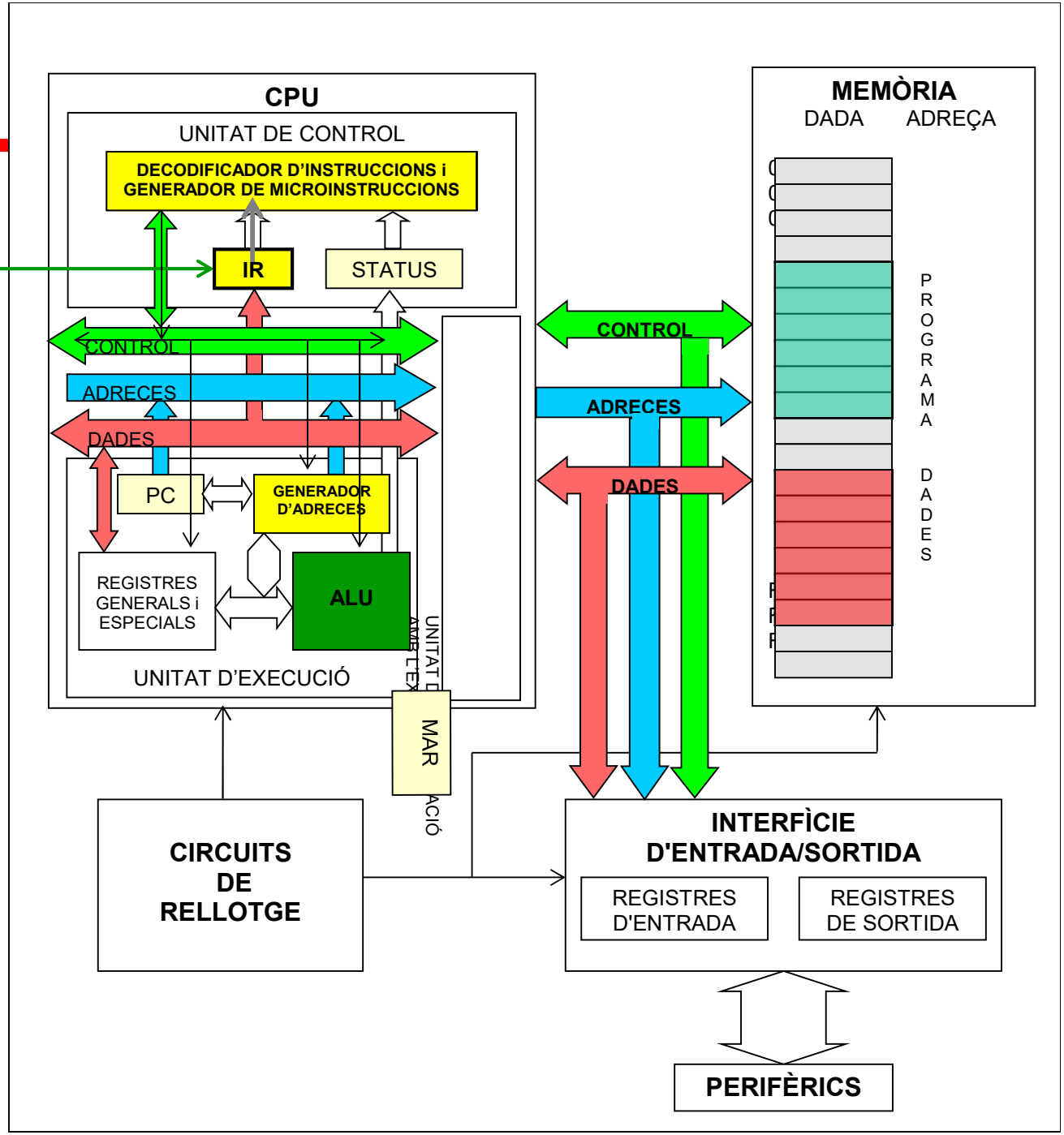
Fases

FETCH


CPU
UNITAT DE CONTROL
DECODIFICADOR D'INSTRUCCIONS i GENERADOR DE MICROINSTRUCCIONS
IR
STATUS
CONTROL
ADRECES
DADES
PC
GENERADOR D'ADRECES
REGISTRES GENERALS i ESPECIALS
ALU
UNITAT D'EXECUCIÓ
MAR
MEMÒRIA
DADA
ADREÇA
PROGRAMA
DADES
CIRCUITS DE RELLOTGE
INTERFÌCIE D'ENTRADA/SORTIDA
REGISTRES D'ENTRADA
REGISTRES DE SORTIDA
PERIFÈRICS

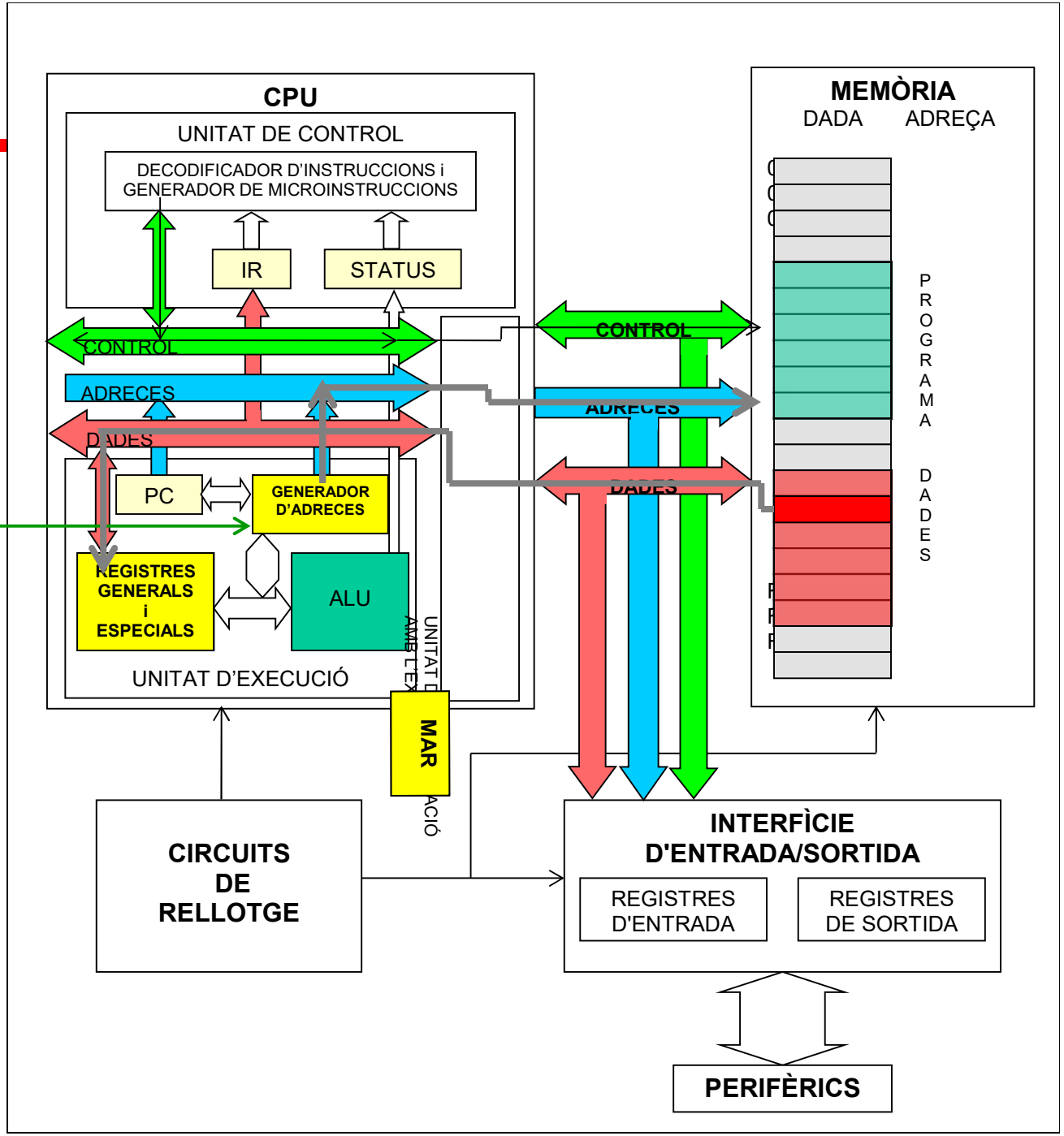
Fases

DECODIFICAR



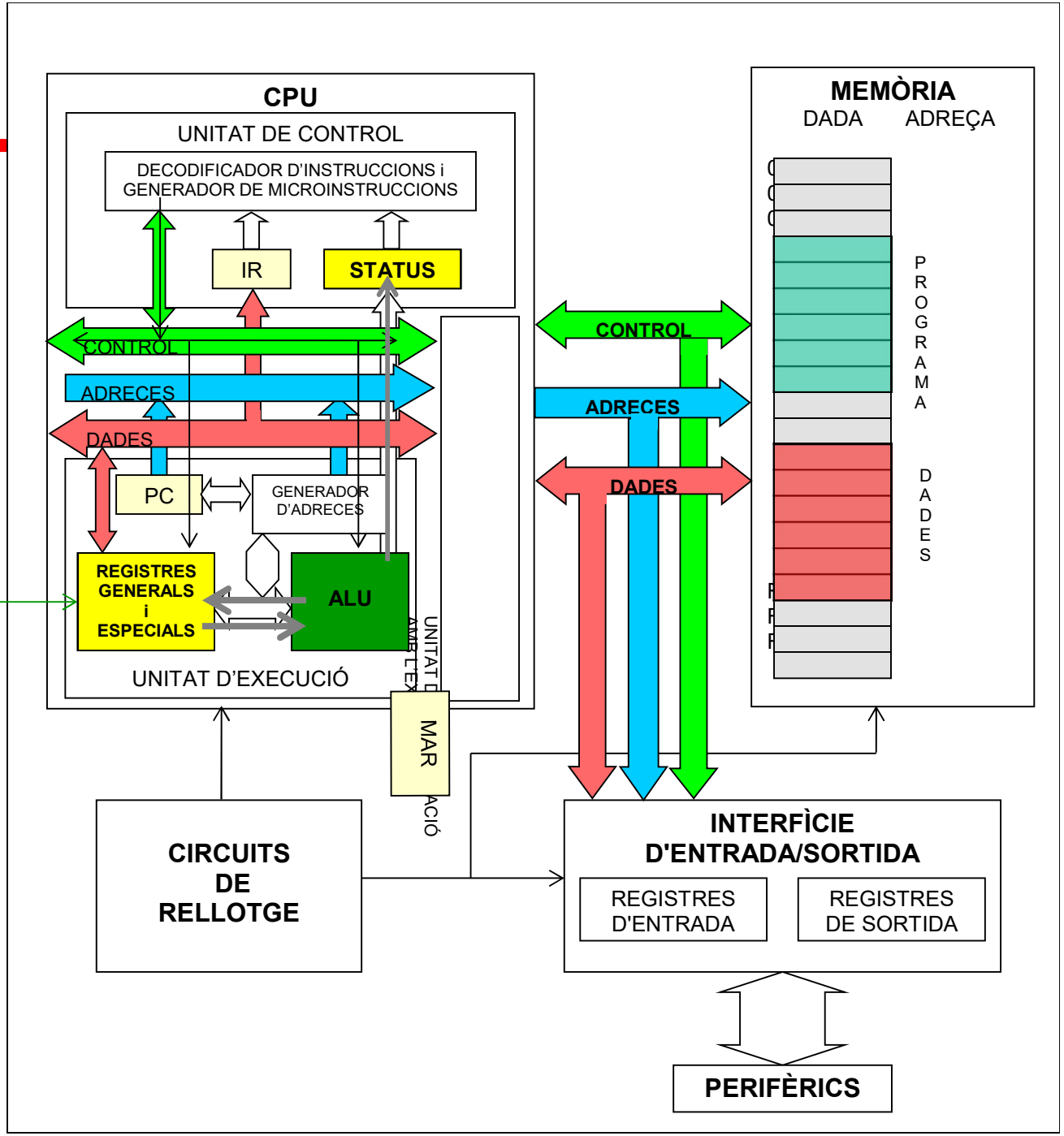
Fases

LECTURA D'OPERANDS



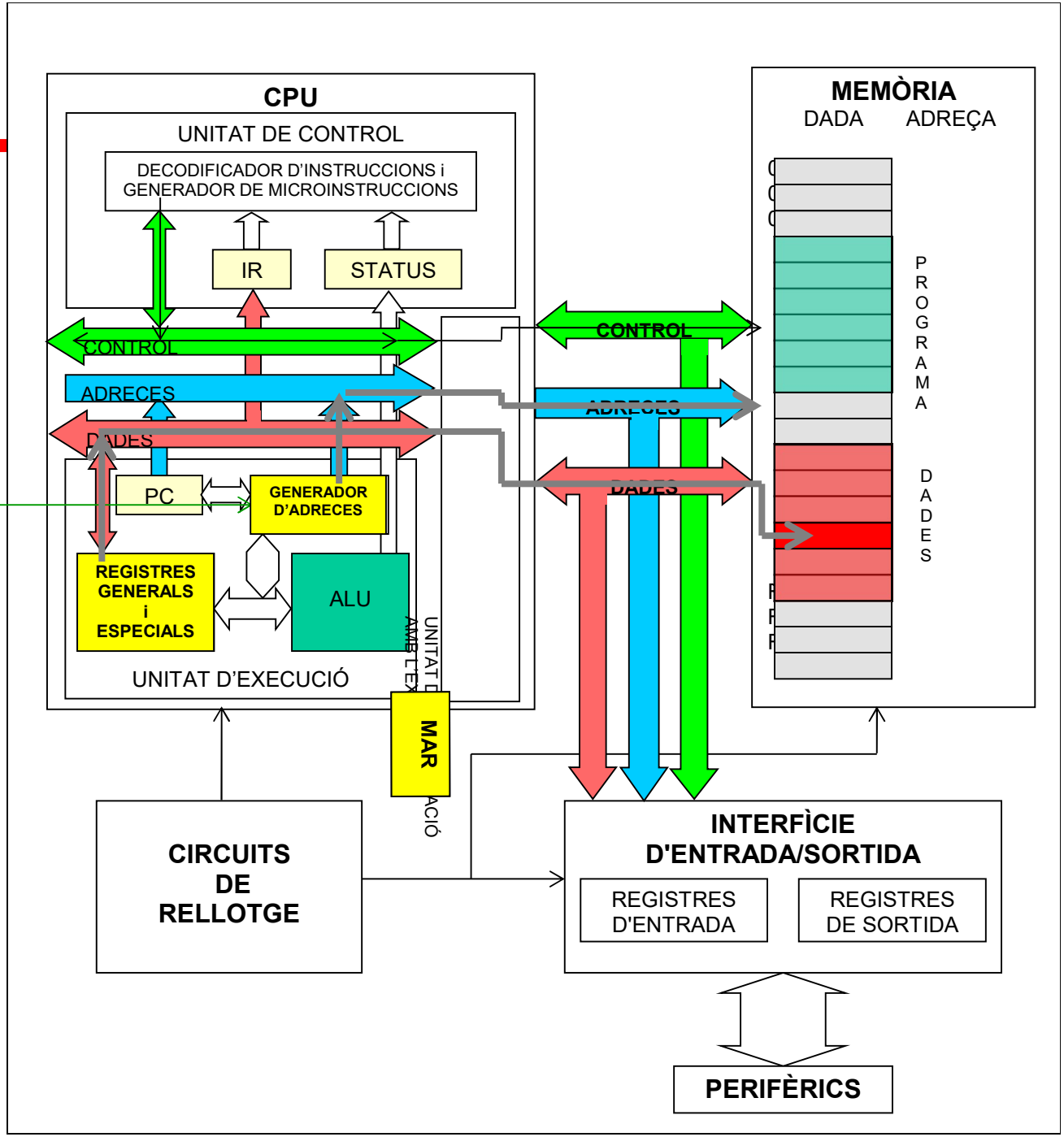
Fases

EXECUCIÓ



Fases

GUARDAR RESULTATS



Execució d'un programa

Execució d'un programa pas a pas.

Suposem un programa que fa el següent:

•Suma contingut de 2 registres, R_0 i R_1

nombre de registres és petit, podem posar un identificador per a cada reg. (en lloc d'adreça)

•Resultat es guardarà a una posició de memòria, pex. h8000

•BUS d'adreces 16 bits.

•Suposem Memòria organitzada en Bytes.

•Suposem que el programa comença a la posició h2000. És a dir, que la primera instrucció està situada a l'adreça de memòria 2000.

•Si la memòria està organitzada en bytes generalment les instruccions ocuparan + d'una posició

Execució d'un programa

Programa en ensamblador

ADD R0, R1	Suma el contingut del registre R_0 al del R_1 i el resultat a R_0 $R_0 <= R_0 + R_1$
STO 8000, R0	Guarda contingut de R_0 a posició de mem. 8000. Mem[8000] <= R_0
JMP 2004	salta a la posició de memòria 2004

- Suposarem que la 1a instrucció es pot codificar en un sol byte
  Conjunt de registres de 4 registres => 2 bits d'adreça
  (codi <1B, adreces de R0 i R1: 2 bits cadascun)

- La 2a ocupa més.
  (2 B Per guardar l'adreça '8000' + 1B pel codi)

- La 3a igual que la 2a
  (2 B Per guardar l'adreça '2004' + 1B pel codi)

Adreces Conj. Registres	
0 0	R0
0 1	R1
1 0	R2
1 1	R3

Execució d'un programa

Aquest programa queda en un bucle infinit saltant a la línia 2004 contínuament.

Els processadors no saben 'no fer res', sempre han d'estar executant alguna instrucció.

Una vegada tenim el programa a la memòria els passos per executar-lo són:

DADA	ADREÇA
	0000
	0001
	0002
ADD R0,R1	2000
STO R0	2001
80	2002
00	2003
JMP	2004
20	2005
04	2006
resultat	8000
	FFFD
	FFFE
	FFFF

PROGRAMA

Execució d'un programa

1.D'alguna forma, al **PC** hi ha un **2000**.

2.CPU: contingut del **PC al BUS d'adreces**, UC activa senyals per indicar la memòria que es vol llegir la informació que hi ha a aquesta posició (generalment un senyal que es diu **READ o R//W**). A més, **PC =PC+1** per apuntar a la propera instrucció del programa.

3.La memòria col·loca la informació al **bus de dades**. La CPU posa la informació que hi ha al **bus de dades al IR**

4.La **UC decodifica** la instrucció del IR, (ADD R0,R1) i comença a generar les microinstruccions (els senyals de control) que realitzen la suma:

1.Dirigir el contingut de R0 a una entrada de l'ALU, i R1 a l'altre entrada

2.Activar la ALU per que realitzi la suma

3.Posar el resultat de la operació al registre R0

Execució d'un programa

5.- Tant sols s'ha executat la 1a instrucció.

Cicle d'Instrucció: temps que es tarda en executar-se una instrucció. El cicle d'instrucció varia segons la instrucció i = varis **Cicles de Rellotge**.

El següent pas consisteix en tornar a començar la seqüència: el contingut del **PC al bus d'adreces** (ara 2001) i activació dels senyals de control per llegir de la memòria. PC<=PC+1 (PC=2002).

6.- Mem. Posa valor de **posició 2001 BUS-dades** i CPU la posa al **IR**

7.- **UC decodifica el contingut de IR**, UC es dona conta que necessita més informació per executar la instrucció (per ara tant sols té STO R0, necessita l'adreça destí), per obtenir-la fa les següents passes

1.[PC] a BUS-adr, activa **mem. per llegir**, i **[PC]=[PC]+1** (PC= 2003).

2.Mem. posa **contingut d'adreça 2002 a BUS-dades** (80 part MSB de l'adreça destí. **La CPU guardarà això** en un registre (el MAR)

3.[PC] a BUS-adr, activa **mem. per llegir**, i **[PC]=[PC]+1** (PC= 2004).

4.Mem. posa **contingut d'adreça 2003 a BUS-dades**. (00 = part LSB d'adreça destí). **La CPU junta 00** amb el 80 (MAR). CPU té l'adreça destí complerta.

Execució d'un programa

8. La CPU té tota la informació per poder executar la instrucció (STO R_0,8000). Per tant, la UC genera les següents microinstruccions:

    1 Posar el contingut de **R_0 al bus de dades**.

    2. Posar el contingut del **MAR al bus d'adreces**.

Activar senyals del **bus de control per que la memòria guardi** el que hi ha al bus de dades.

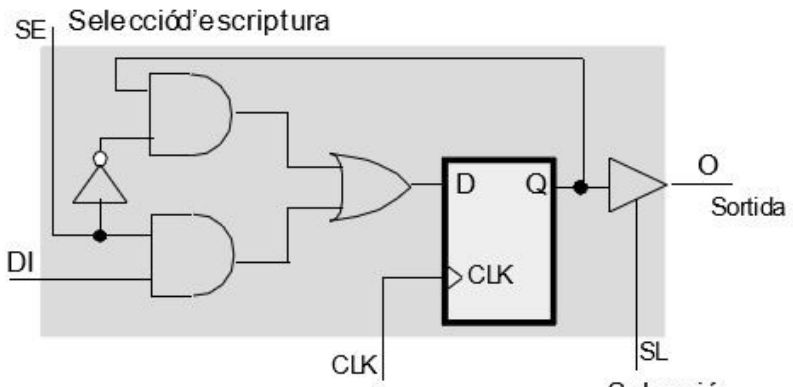
9 Es torna a començar la seqüència per que s'executi la instrucció JMP 2004.

En general, cada microinstrucció s'executa en 1 cicle de rellotge.

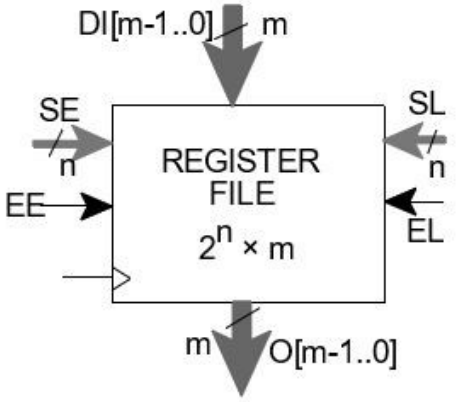
El fabricant de cada CPU indica quants cicles de rellotge necessita cada instrucció per executar-se, de quants cicles de clk consta cada cicle d'instrucció.

Coneixent la freqüència de clk, càlcul de temps d'execució d' instrucció, temps d' execució de programa (sempre que no hi hagin mecanismes de pipeline).

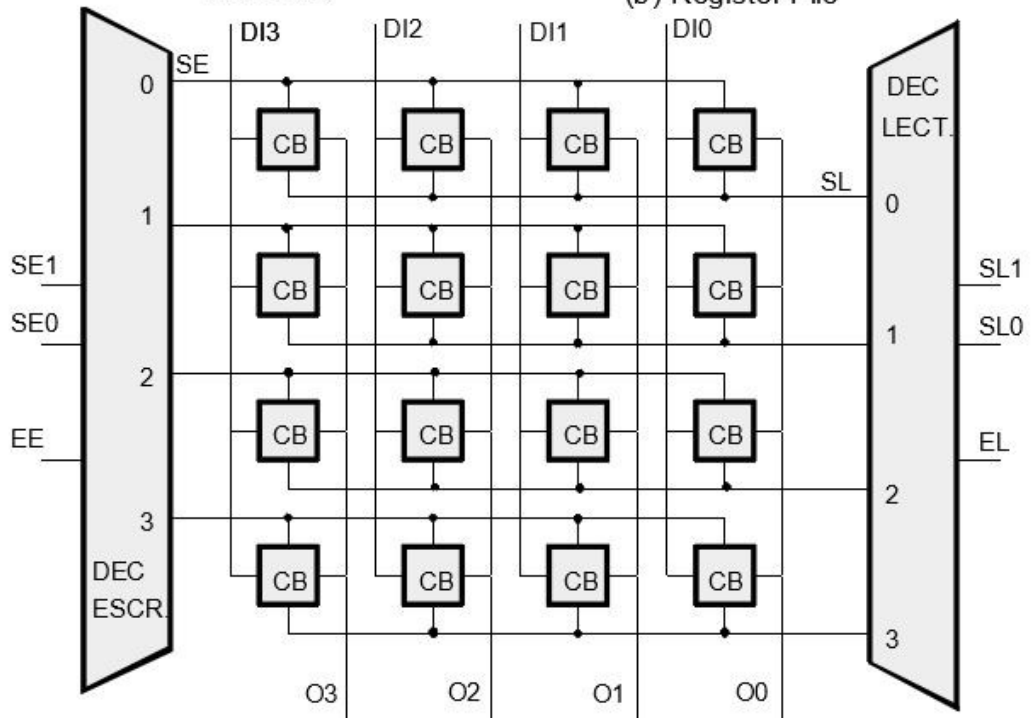
CPU conjunt de registres



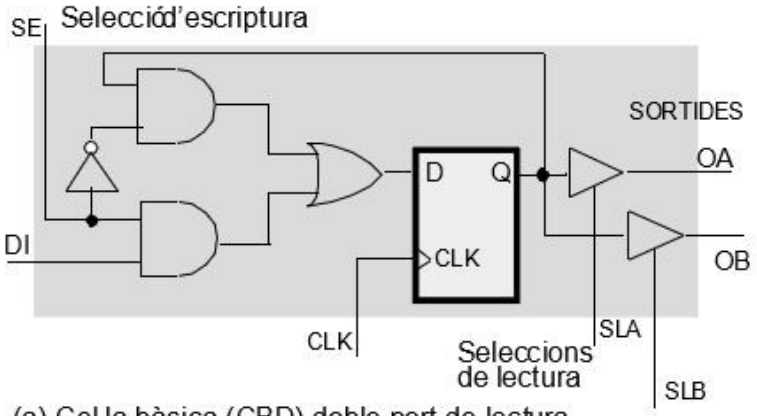
(a) Cel·la bàsica (CB)



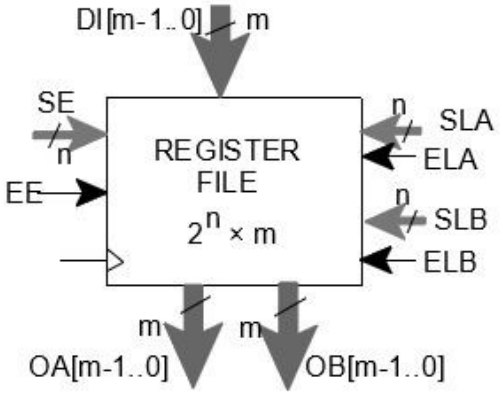
(b) Register File



CPU conjunt de registres



(a) Cel·la bàsica (CBD) doble port de lectura



(b) *Register File* de doble port de lectura