

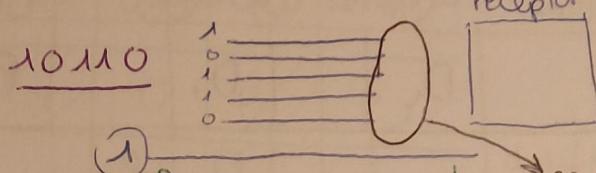
CÓDIGO HAMMING

(XOR)

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Emitor y receptor

Paridad PAR



B'

3 und, añadimos otro
(paridad par)

receptor

genera paridad (bit)
↓
Comprueba.

m nº bits de dato a T

2^m combinaciones

r bits comprobación

$m+r = n$ 2^n combinaciones posibles $\rightarrow 2^m$

Distancia Hamming. (la menor de las diferencias de dos posiciones Hamming)

d nº errores de 1 bit.

Detectarse DISTANCIA HAMMING $\geq d+1$

Corregirse DISTANCIA HAMMING $\geq 2d+1$

$m=4$, $r=3$, DH=3 $\rightarrow 3 \geq d_1 + 1 \rightarrow d_1 = 2$ errores.

$3 \geq 2d_2 + 1 \rightarrow d_2 = 1$ error corregible.

Para corregir el error de un bit: $m+r \leq 2^r - 1$

- Bit de paridad en posiciones de potencia de 2.

- MSB _____ LSB

- Bit de paridad $b_j = \text{XOR}$ bit datos 1 en posición 2^j

- Dato b $b_1, b_2, \dots, b_m / b_1 + b_2 + \dots + b_m = b$

$m=4$, $r=3$ Dato D₄ D₃ D₂ D₁

b_7	b_6	b_5	b_4	b_3	b_2	b_1	
D ₄	D ₃	D ₂	P ₃	D ₁	P ₂	P ₁	
1	1	0	1	0	0	0	
D ₄	D ₂	D ₁	P ₁				

b_7	b_6	b_5	b_4	b_3	b_2	b_1
D_4	D_3	D_2	P_3	D_1	P_2	P_1
111	110	101	100	011	010	001
D_4	D_3	D_2	D_1		P_1	
D_4	D_3			D_1	P_2	
D_4	D_3	D_2	P_3			

$D_4 \ D_3 \ D_2 \ P_3 \ D_1 \ P_2 \ P_1$

- Receptor vuelve a generar bit de paridad.
 - Compone con los que te llegan.
 - Si son distintos, complementa el bit malo

A	B	C	$A \oplus B$	$(A \oplus B) \oplus C$	
0	0	0	0	0	
un (1)	0	0	1	1	Paridad 1
un (1)	0	1	1	1	Paridad 1
dos (2)	0	1	1	0	Paridad 0
1	0	0	1	1	
1	0	1	1	0	
1	1	0	0	0	
1	1	1	0	1	

$$D_4 D_3 D_2 D_1 = 1010$$

b_7	b_6	b_5	b_4	b_3	b_2	b_1
1 1 1	1 1 0	1 0 1	1 0 0	0 0 1	0 1 0	0 0 1
1	0	1	p_3	0	p_2	p_1
1		1		0		$p_1 = 0$
1	0		p_3	0	$p_2 = 1$	p_2
1	0	1	$p_3 = 0$			

dos (1) $\rightarrow p_1 = 0$
 un (1) $\rightarrow p_2 = 1$
 dos (11) $\rightarrow p_3 = 0$

1 0 1 0 0 1 0

este código le mandamos al receptor, es lo que recibe.

b_7	b_6	b_5	b_4	b_3	b_2	b_1	prueba de paridad
1 1 1	1 1 0	1 0 1	1 0 0	0 0 1	0 1 0	0 0 1	
1	0	1	0	0	1	0	Palabra recibida.
1		1		0		$p_1 = 0$	correcto
1	0			0	$p_2 = 1$	$p_2 = 0$	correcto
1	0	1	$p_3 = 0$				correcto

D_4	D_3	D_2	P_3	D_1	P_2	P_1	comprobació	bit comp	enviado con error
b_7	b_6	b_5	b_4	b_3	b_2	b_1			
001	000	000	000	000	000	000			
palabra recibida	0	0	1	0	0	1	0		
	0		1		0		$P_1 \neq 1$	error	1
	0	0			0	$P_2 = 0$		error	1
	0	0	1	$P_3 = 1$				error	1

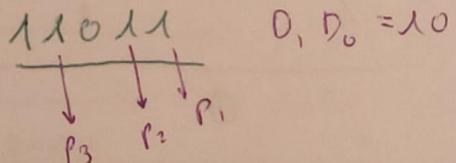
$\rightarrow \text{XOR} \rightarrow P_i$ que viene con p_i calculado

$\rightarrow P_3 P_2 P_1 = 111 \rightarrow \text{error en línea 7.}$

PAS

m	4	5	6	7	8	9	10	11	12
r	3	4	4	4	4	4	4	4	5

$$m+r \leq 2^r - 1$$



$$\cdot r=1 \quad m+1 \leq 2^1 - 1 \quad m=0$$

$$\cdot r=2 \quad m+2 \leq 2^2 - 1 \quad m \leq 1$$

$$\cdot r=3 \quad m+3 \leq 2^3 - 1 \quad m \leq 4 \quad \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array}$$

$$\cdot r=4 \quad m+4 \leq 2^4 - 1 \quad m \leq 11 \quad \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \end{array}$$

$$\cdot r=5 \quad m+5 \leq 2^5 - 1 \quad m \leq 26 \quad \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \\ 15 \\ 16 \\ 17 \\ 18 \\ 19 \\ 20 \\ 21 \\ 22 \\ 23 \\ 24 \\ 25 \\ 26 \end{array}$$

1001

D₄ D₃ D₂ D₁

b ₇	b ₆	b ₅	b ₄	b ₃	b ₂	b ₁
D ₄	D ₃	D ₂	P ₃	D ₁	P ₂	P ₁
1	1	0	1	0	1	0
1	0	1	0	1	0	0
1	0	0	1			
1	0	0	0	1	1	0
1	0	0	0	0	1	1
1	0	0	0	0	0	0

Receptor : 1010010

D ₄ b ₇	P ₃ b ₆	D ₂ b ₅	P ₃ b ₄	D ₁ b ₃	P ₂ b ₂	P ₁ b ₁	Comprobación	bit
1	1	1	0	1	0	1		
1	1	0	1	0	1	1		
1	0	1	1		1	1	error	1
1	0		1	1	P ₂ =0		error	1
1	0	1	P ₃ =0				correcto	
1	0	1	P ₃ =0				error	0

$\Rightarrow 1010010$

Si el error sale en un bit de paridad, no hay corrección posible, no sabemos dónde. Hay que solicitar que se envíen de nuevo los datos.

emisor: 1010010

receptor: 1010001

b_7	b_6	b_5	b_4	b_3	b_2	b_1	P_2	P_1	
1	1	0	1	0	0	0	1		
1	1	0	1	0	0	1	$P_1 = 0$	error	1
1	0	1	0	0	1	0	$P_2 = 1$	error	1
1	0	1	0	0	1	1	$P_3 = 0$	all	0
1	0	1	0	0	1	1			

PA 0.2 Hamming

Detector $DH \geq d+1$

Corregir $DH \geq (2d+1)$

$$m=4, r=3 \quad DH=3$$

$$\boxed{m+r \leq 2^r - 1}$$

Para detectar un error (d)

r : número de bits de paridad

$$r=3 \rightarrow m+3 \leq 2^3 - 1; \quad m \leq 4$$

$$r=4 \rightarrow m+4 \leq 2^4 - 1; \quad m \leq 11$$

$$r=5 \rightarrow m+5 \leq 2^5 - 1; \quad m \leq 26$$

$$r=6 \rightarrow m+6 \leq 2^6 - 1; \quad m \leq 57$$

② $b_5 b_4 b_3 b_2 b_1$

$$a) \begin{matrix} 1 & 1 & 0 & 1 & 1 \\ P_3 & P_2 & P_1 \end{matrix} \rightarrow D_2 D_1 = 10$$

$$b) \begin{matrix} 1 & 0 & 1 & 0 & 1 & 1 \end{matrix} \rightarrow D_4 D_3 D_2 D_1 = 1011$$

$$P_3 \quad P_2 \quad P_1$$

$$c) b_9 \dots b_1 = \underbrace{1 \ 1 \ 1 \ 1}_{P_4} \underbrace{0 \ 0 \ 0 \ 0}_{P_3} \underbrace{1 \ 0 \ 1 \ 0}_{P_2 \ P_1} \rightarrow D_5 D_4 D_3 D_2 D_1 = 11100$$

③ $d_2 d_1 = 10$

$$d_2 \ P_3 \ d_1 \ P_2 \ P_1$$

$$\begin{array}{ccccc} b_5 & b_4 & b_3 & b_2 & b_1 \\ \boxed{1} & \boxed{0} & \boxed{1} & \boxed{0} & \boxed{0} \end{array}$$

$$\begin{array}{ccccc} b_5 & b_4 & b_3 & b_2 & b_1 \\ 1 & 1 & 0 & 0 & 1 \end{array}$$

$$P_1 = b_3 \oplus b_5 \quad \left\{ \begin{array}{l} = 1 \\ = 0 \end{array} \right.$$

$$P_2 = b_3 \quad \left\{ \begin{array}{l} = 0 \\ = 1 \end{array} \right.$$

$$P_3 = b_5 \quad \left\{ \begin{array}{l} = 1 \\ = 0 \end{array} \right.$$

b) $\lambda_{011} = d_4 d_3 d_2 d_1$

$$\begin{array}{ccccccc} D_4 & D_3 & D_2 & P_3 & D_1 & P_2 & P_1 \\ b_7 & b_6 & b_5 & b_4 & b_3 & b_2 & b_1 \\ 111 & 110 & 101 & 100 & 011 & 010 & 001 \end{array}$$

$$\left. \begin{array}{l} P_1 = b_3 \oplus b_5 \oplus b_7 = d_1 \oplus d_2 \oplus d_4 = 1 \\ P_2 = b_3 \oplus b_6 \oplus b_7 = d_1 \oplus d_3 \oplus d_4 = 0 \\ P_3 = b_5 \oplus b_6 \oplus b_7 = d_2 \oplus d_3 \oplus d_4 = 0 \end{array} \right]$$

1010101

c) $\lambda_{11100} = d_5 \dots d_1$

$$\begin{array}{ccccccc} D_5 & P_4 & D_4 & D_3 & D_2 & P_3 & D_1 & P_2 & P_1 \\ b_9 & b_8 & b_7 & b_6 & b_5 & b_4 & b_3 & b_2 & b_1 \\ 1001 & 1000 & 111 & 110 & 101 & 100 & 011 & 010 & 001 \end{array}$$

$$\left. \begin{array}{l} P_1 = b_3 \oplus b_5 \oplus b_7 \oplus b_9 = 0 \\ P_2 = b_3 \oplus b_6 \oplus b_7 = 0 \\ P_3 = b_5 \oplus b_6 \oplus b_7 = 0 \\ P_4 = b_9 = 1 \end{array} \right]$$

111100000

(4)

a) $M_{10000}^{D_2 D_6 D_5 P_1 D_4 D_3 D_2 P_3 D_1 P_2 P_1} (b_{11} \dots b_1)$

$$\begin{array}{cccccccccc} D_7 & D_6 & D_5 & P_4 & D_4 & D_3 & D_2 & P_3 & D_1 & P_2 & P_1 \\ b_7 & b_{10} & b_9 & b_8 & b_7 & b_6 & b_5 & b_4 & b_3 & b_2 & b_1 \\ 1011 & 1010 & 1001 & 1000 & 111 & 110 & 101 & 100 & 011 & 010 & 001 \end{array}$$

$$\left. \begin{array}{l} P_1 = D_1 \oplus D_2 \oplus D_4 \oplus D_5 \oplus D_7 = 1 \rightarrow \checkmark \\ P_2 = D_1 \oplus D_3 \oplus D_4 \oplus D_6 \oplus D_7 = 1 \rightarrow \checkmark \\ P_3 = D_2 \oplus D_3 \oplus D_4 = 1 \rightarrow \checkmark \\ P_4 = D_5 \oplus D_6 \oplus D_7 = 0 \rightarrow \checkmark \end{array} \right]$$

no hay error;
 $D_7 D_6 D_5 D_4 D_3 D_2 D_1 = 1100011$

b) $D_4 D_3 D_2 D_1 D_0 P_3 P_2 P_1$ $(b_7 \dots b_1)$
 111 110 101 100 011 010 001

$$P_1 = D_1 \oplus D_2 \oplus D_4 = 1 \rightarrow X \quad \text{error: } 011 (3) \text{ (d.)}$$

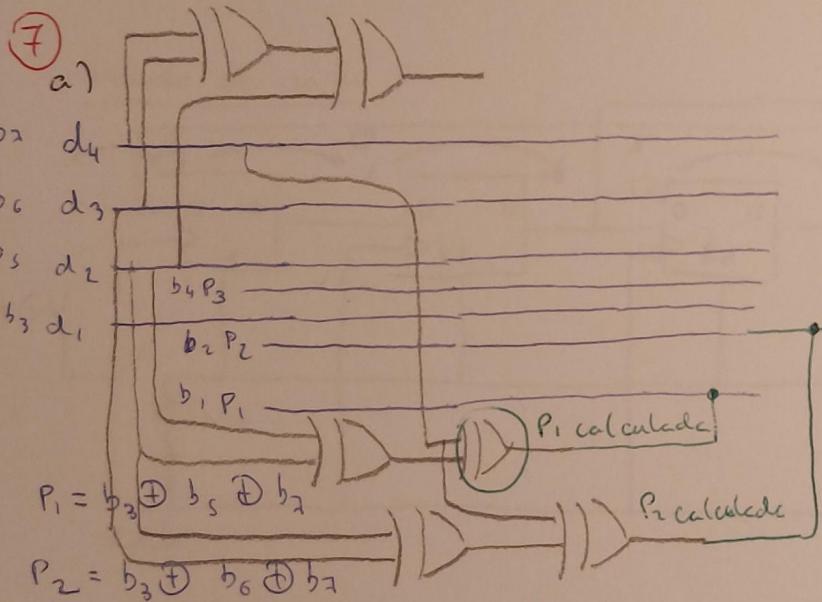
$$P_2 = D_1 \oplus D_3 \oplus D_4 = 1 \rightarrow \checkmark \quad \text{dato: } 0111$$

$$P_3 = D_2 \oplus D_3 \oplus D_4 = 0 \rightarrow X$$

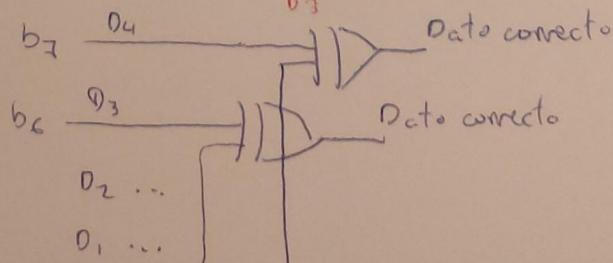
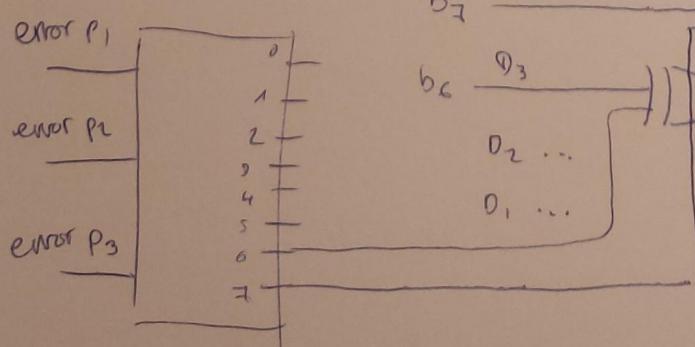
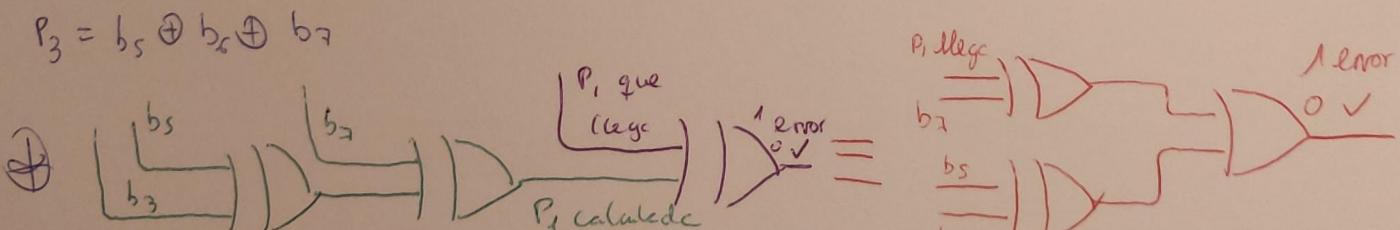
⑤ El código hamming detecta errores pero no se puede corregir.

⑥ Puertos OR exclusivo para generar y corregir errores.

Decodificador para detectar qué palabra hay que corregir.



CONTINÚA ...



$A \setminus B$	$A \oplus B$
$[0] 0$	0 $\neq B$
$[0] 1$	1 $\neq B$
$[1] 0$	1 $\neq \bar{B}$

Flip-Flops

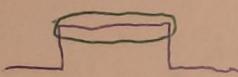
SR	Q^*
00	Q
01	0
10	1
11	X

JU	Q^*
00	Q_n
01	0
10	1
11	\bar{Q}_n

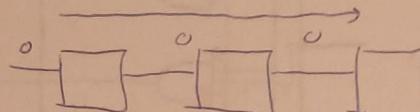
T	JU	Q^*
0	00	Q_n
1	11	\bar{Q}_n

D	JU	Q^*
0	01	0
1	10	1

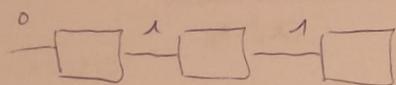
Activación por nivel :



Activación por flanco :

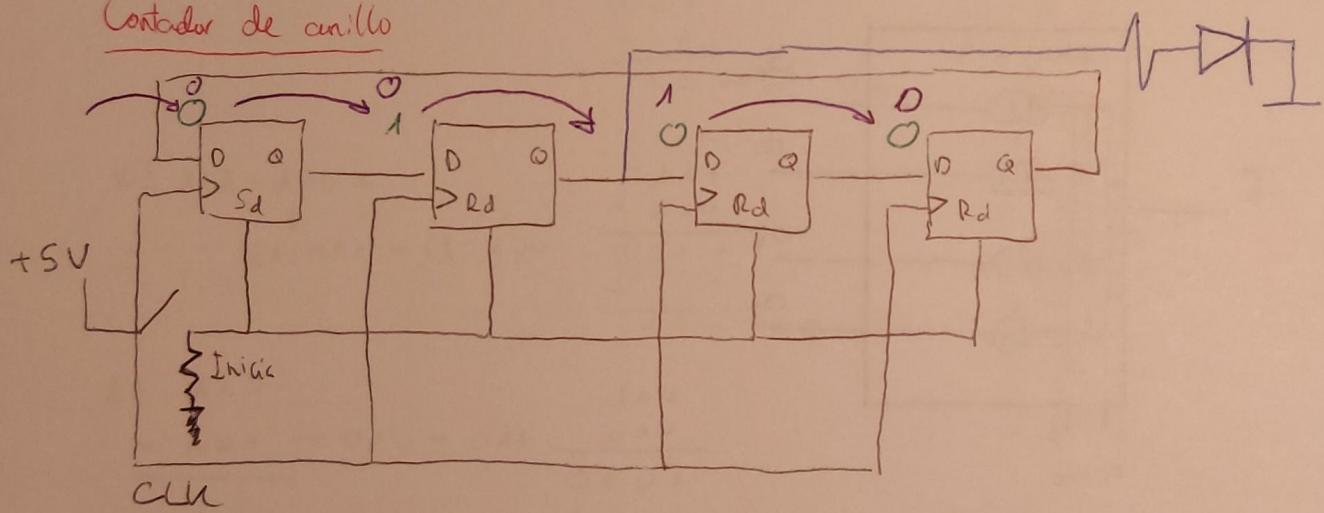


Agregaremos el 0 por todo el sistema



Depende del flip-flop tomará una
valor un otro.

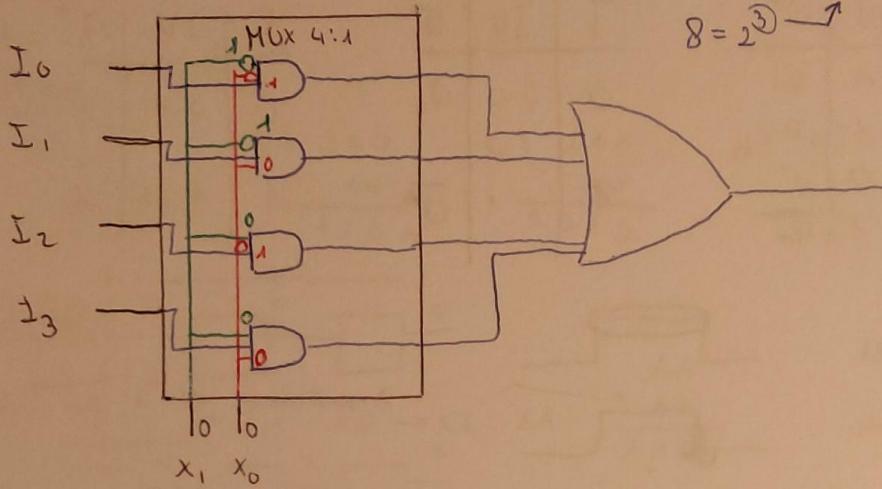
Contador de anillo



Multiplexor: Seleccionar 1 de 4. Se necesitan dos variables de control.

$$4 = 2^2 \rightarrow \text{variables de control}$$

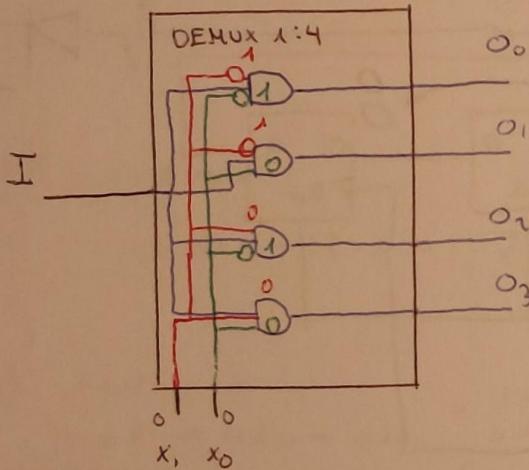
$$8 = 2^3 \rightarrow$$



Demultiplexor: Seleccionar 4 de 1. Se necesitan dos variables de control.

$$4 = 2^2 \rightarrow$$

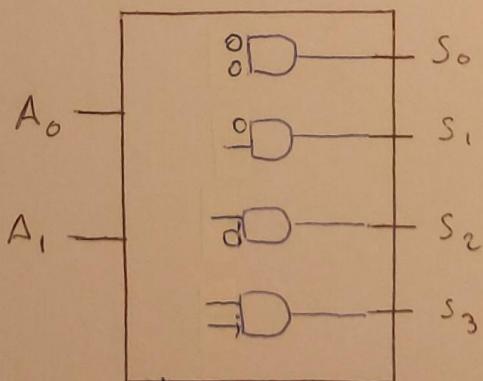
$$8 = 2^3 \rightarrow$$



Codificador con prioridad:

I_3	I_2	I_1	I_0	O_1	O_2
0	0	0	1	0	0
0	0	1	x	0	1
0	1	x	x	1	0
1	x	x	x	1	1

Decodificador:



ESTRUCTURA de COMPUTADORES - PRÁCTICAS de AULA .

PA 0

1.

SUMA

$$\begin{array}{r} 11 \\ + 11 \\ \hline 110 \end{array}$$

$$\begin{array}{r} 100 \\ + 10 \\ \hline 110 \end{array}$$

$$\begin{array}{r} 111 \\ + 11 \\ \hline 1010 \end{array}$$

$$\begin{array}{r} 110 \\ + 100 \\ \hline 1010 \end{array}$$

RESTA

$$\begin{array}{r} 11 \\ - 01 \\ \hline \end{array} \rightarrow C1: 10 \rightarrow C2: 11$$

$$\begin{array}{r} 11 \\ + 11 \\ \hline \textcircled{1} 10 \\ + 1 \\ \hline 11 \end{array} \rightarrow C1: 00$$

$$\begin{array}{r} 11 \\ - 01 \\ \hline 10 \end{array}$$

$$\begin{array}{r} 11 \\ - 10 \\ \hline \end{array} \rightarrow C1: 01 \rightarrow C2: 10$$

$$\begin{array}{r} 11 \\ + 10 \\ \hline \textcircled{1} 01 \\ + 1 \\ \hline 10 \end{array} \rightarrow C1: 01$$

$$\begin{array}{r} 11 \\ - 10 \\ \hline 01 \end{array}$$

$$k) 111 - 100 =$$

$$\downarrow$$

$$C1: 011 \rightarrow C2: 100$$

$$\begin{array}{r} 111 \\ + 100 \\ \hline \textcircled{1} 011 \\ + 1 \\ \hline 100 \end{array} \rightarrow 011$$

$$\begin{array}{r} 111 \\ - 100 \\ \hline 011 \end{array}$$

$$l) 110$$

$$\begin{array}{r} 110 \\ - 101 \\ \hline \end{array} \rightarrow 010 \rightarrow 011$$

$$\begin{array}{r} 110 \\ + 011 \\ \hline \textcircled{1} 001 \\ + 1 \\ \hline 010 \end{array} \rightarrow 101$$

$$\begin{array}{r} 110 \\ - 101 \\ \hline 001 \end{array}$$

$$m) 111$$

$$\begin{array}{r} 111 \\ - 11 \\ \hline \end{array} \rightarrow 00 \rightarrow 01$$

$$\begin{array}{r} 111 \\ + 01 \\ \hline \textcircled{1} 000 \\ + 1 \\ \hline 001 \end{array} \rightarrow 110$$

$$\begin{array}{r} 111 \\ - 11 \\ \hline 100 \end{array}$$

RESTA : $0 - 0 = 0$

$1^c - 0 = 1$

$1 - 0 = 1$

$1 - 1 = 0$

PRODUCTO

$$e) \begin{array}{r} 11 \\ \times 1 \\ \hline 11 \end{array}$$

$$f) \begin{array}{r} 11 \\ \times 11 \\ \hline 11 \\ + 11 \\ \hline 1001 \end{array}$$

$$g) \begin{array}{r} 111 \cdot 101 \rightarrow \\ \rightarrow 111 \\ \times 101 \\ \hline 111 \\ + 000 \\ \hline 111 \\ \hline 100011 \end{array}$$

$$h) \begin{array}{r} 1011 \\ \times 1001 \\ \hline 1011 \\ + 0000 \\ 0000 \\ \hline 1011 \\ \hline 1100011 \end{array}$$

DIVISIÓN

$$n) \begin{array}{r} \overline{110} \\ 11 \\ \hline 000 \\ - 00 \\ \hline 00 \end{array}$$

$$\begin{array}{r} 11 \\ \hline 10 \end{array}$$

$$o) \begin{array}{r} \overline{110} \quad \overline{110} \\ 10 \quad 11 \\ \hline 010 \\ - 10 \\ \hline 00 \end{array}$$

2. a) $15_{10} = 1111_2$

b) $59_{10} = 64 \quad 32 \quad 16 \quad 8 \quad 4 \quad 2 \quad 1 \quad \downarrow 1000101_2$

$$\begin{array}{r} 64 \\ 59 \\ \hline 05 \\ - 4 \\ \hline 1 \end{array} \quad \begin{array}{r} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \end{array} \rightarrow 111011_2$$

$$\begin{array}{r} 59 \\ - 32 \\ \hline 27 \\ - 16 \\ \hline 11 \\ - 8 \\ \hline 3 \end{array}$$

$$\begin{array}{r} 59 \quad \underline{2} \\ \textcircled{1} \quad 29 \quad \underline{2} \\ \textcircled{1} \quad 14 \quad \underline{2} \\ \textcircled{1} \quad 3 \quad \underline{2} \\ \textcircled{1} \quad \textcircled{1} \quad \underline{2} \\ \textcircled{1} \quad \textcircled{1} \quad \textcircled{1} \end{array} \rightarrow 111011_2$$

c) $0^1 3125 =$

$$0^1 3125 \cdot 2 = 0^1 6250$$

$$0^1 6250 \cdot 2 = 1^1 2500$$

$$0^1 2500 \cdot 2 = 0^1 5$$

$$0^1 5 \cdot 2 = 1^1 0$$

$$0 \cdot 2 = 0$$

$$\left. \begin{array}{r} 0^1 0101 \\ \downarrow \\ 1^1 2500 \end{array} \right\} 1^1 0101_2$$

d) $31,125_{10} = 11111,001_2$

$$\begin{array}{r} 32 \quad 16 \quad 8 \quad 4 \quad 2 \quad 1 \\ \quad 1 \quad 1 \quad 1 \quad 1 \end{array}$$

$$\begin{array}{r} 31 \\ - 16 \\ \hline 15 \\ - 8 \\ \hline 7 \\ - 4 \\ \hline 3 \end{array}$$

$$\left| \begin{array}{l} 0^1 125 \cdot 2 = 0^1 250 \\ 0^1 250 \cdot 2 = 0^1 5 \\ 0^1 5 \cdot 2 = 1 \\ 0 \cdot 2 = 0 \end{array} \right.$$

3.

a) $\underbrace{1100}_{12} \underbrace{10100}_{10} \underbrace{1010}_{5} \underbrace{111}_{7} =$
 $C A S F_{16}$

10 - A
11 - B
12 - C
13 - D
14 - E
15 - F

b) $\underbrace{0110}_{3} \underbrace{100}_{4} \underbrace{1101}_{13} =$
 $34D_{16}$

4.

a) $10A4_{16} = 0001\ 0000\ 1010\ 0100_2$

b) $CF8E_{16} = 1100\ 1111\ 1000\ 1110_2$

c) $9742_{16} = 1001\ 0111\ 0100\ 0010_2$

5.

a) $650_{10} = \begin{array}{r} 650 \\ 10 \end{array} \begin{array}{r} 16 \\ -40 \\ \hline 25 \end{array} \rightarrow 28A_{16}$

b) $4025_{10} = \begin{array}{r} 4025 \\ 82 \end{array} \begin{array}{r} 16 \\ 25 \\ -24 \\ \hline 1 \end{array} \begin{array}{r} 16 \\ 9 \\ -8 \\ \hline 1 \end{array} \rightarrow FB9_{16}$

6.

a) $1386_{10} \rightarrow 6 = 6 \cdot 10^0 \rightarrow$ bit de peso 0 .

b) $54'692_{10} \rightarrow 6 = 6 \cdot 10^1 \rightarrow$ bit de peso -1 .

c) $631'920_{10} \rightarrow 6 = 6 \cdot 10^2 \rightarrow$ bit de peso 2 .

7.

a) $10 = 1 \cdot 10^1$

b) $100 = 10^2$

c) $10000 = 10^4$

d) $1000000 = 10^6$

9.

a) $11 = 1 \cdot 2^0 + 1 \cdot 2^1 = 3$

b) $100 = 1 \cdot 2^2 = 4$

c) $111 = 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 7$

d) $1000 = 1 \cdot 2^3 = 8$

e) $11101 = 1 \cdot 2^0 + 1 \cdot 2^1 + 1 \cdot 2^3 + 1 \cdot 2^4 = 29$

f) $11,011 = 1 \cdot 2^0 + 1 \cdot 2^1 + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} = 3^1 \left(\frac{1}{4} + \frac{1}{8} \right)$

8. $10^4 - 1 = 9999$

$\frac{1}{4} + \frac{1}{8} = \frac{3}{8} = 0'375 \rightarrow 3'375$

10.

$$a) 2^2 - 1 = 3$$

$$b) 2^7 - 1 = 127$$

$$c) 2^{10} - 1 = 1023$$

11.

$$a) 17 \quad 2^4 = 16 \rightarrow 5 \text{ bits}$$

$$b) 81 \quad 2^7 = 128 \rightarrow 7 \text{ bits}$$

$$c) 35 \quad 2^5 = 32 \rightarrow 6 \text{ bits}$$

$$d) 32 \quad 2^5 = 32 \rightarrow 6 \text{ bits} \quad (2^5 - 1 = 31)$$

12.

$$a) ES_{16} = 14 \cdot 16^1 + 5 \cdot 16^0 = 229_{10}$$

$$b) B2F8_{16} = 11 \cdot 16^3 + 2 \cdot 16^2 + 15 \cdot 16 + 8 \cdot 16^0 = 45846_{10}$$

$$c) 2374_8 = 2 \cdot 8^3 + 3 \cdot 8^2 + 7 \cdot 8 + 4 \cdot 8^0 = 1276_{10}$$

13.

$$a) 2'08_{10} = 10,000101_2$$

$$0'08 \cdot 2 = 0'16 \quad \text{Error relative} =$$

$$0'16 \cdot 2 = 0'32$$

$$0'32 \cdot 2 = 0'64$$

$$0'64 \cdot 2 = 1'28$$

$$0'28 \cdot 2 = 0'56$$

$$0'56 \cdot 2 = 1'12$$

$$0'12 \cdot 2 = 0'24$$

$$b) 73'625_{10} = 1001001,101_2$$

$$0'625 \cdot 2 = 1'250 \quad \text{Error relative} = 0$$

$$0'250 \cdot 2 = 0'5$$

$$0'5 \cdot 2 = 1$$

$$\begin{array}{ccccccc} 73 & & & & & & \\ -64 & & & & & & \\ \hline 9 & & & & & & \end{array}$$

$$64 \quad 32 \quad 16 \quad 8 \quad 4 \quad 2 \quad 1$$

$$1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1$$

14.

$$a) \underbrace{0100 \quad 00101}_{\text{exp}} \quad \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{matrix} \quad 9001 \quad 1011 \quad 0000 \quad 0000 \quad 0000 \quad 0000 \quad 0000_2$$

$$+ \quad \begin{matrix} 2^7 + 2^2 = 132 \\ \text{exp} \end{matrix} \quad N = (-1)^0 \cdot 2^5 \cdot (1 + 2^{-3} + 2^{-4} + 2^{-5} + 2^{-6}) = 38'75$$

$$132 - 127 = 5$$

$$b) \underbrace{0011 \quad 1011 \quad 0001}_{\text{exp}} \quad \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 \end{matrix} \quad 0110 \quad 1011 \quad 1011 \quad 1011 \quad 1001 \quad 1000_2$$

$$+ \quad \begin{matrix} 2 + 2^4 + 2^5 + 2^6 = 118 \\ \text{exp} \end{matrix} \quad N = (-1)^0 \cdot 2^{-9} \cdot (1 + 2^{-2} + 2^{-3} + 2^{-4} + 2^{-5} + 2^{-6} + 2^{-7} + 2^{-8} + 2^{-9} + 2^{-10} + 2^{-11} + 2^{-12} + 2^{-13} + 2^{-14} + 2^{-15} + 2^{-16} + 2^{-17} + 2^{-18})$$

$$118 - 127 = -9$$

$$2^{-9}$$

$$= 0'002300011658$$

$$c) 470BBBDD_{16}$$

$$\underbrace{0100 \quad 0111 \quad 0000}_{\text{exp}} \quad \begin{matrix} 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 \end{matrix} \quad 1011 \quad 1011 \quad 1011 \quad 1011 \quad 1101 \quad 1101_2$$

$$+ \quad \begin{matrix} 2^7 + 2^3 + 2^2 + 2 = 142 \\ \text{exp} \end{matrix} \quad N = (-1)^0 \cdot 2^{15} \cdot (1 + 2^{-4} + 2^{-5} + 2^{-6} + 2^{-7} + 2^{-8} + 2^{-9} + 2^{-10} + 2^{-11} + 2^{-12} + 2^{-13} + 2^{-14} + 2^{-15} + 2^{-16} + 2^{-17} + 2^{-18} + 2^{-19} + 2^{-20} + 2^{-21} + 2^{-22} + 2^{-23})$$

$$142 - 127 = 15$$

$$2^{15}$$

$$= 35.771'96328$$

d) $7F004381_{16}$

$$\begin{array}{ccccccccc}
 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & \\
 + & & & & & & & & \\
 & 2^8 & -1 = 255 & & & & & & \\
 255 & -127 & = 128 & & & & & & \\
 2^{128} & & & & & & & & \\
 & & & N = (-1)^0 \cdot 2^{128} \cdot (1 + 2^{-1} + 2^{-3} + 2^{-9} + 2^{-14} + 2^{-15} + 2^{-16} + 2^{-23}) & & & & & \\
 & & & = 5'536598469 \cdot 10^{38} \infty, N_e N & & & & & \\
 \end{array}$$

i) OJO! Exponente todo 1s \rightarrow infinito (PSL, PDSO)

15.

$$a) D_3 D_2 D_1 D_0 = 1001$$

$$m=4; r=3$$

$$b) D_6 D_5 D_4 D_3 D_2 D_1 D_0 = 0110111$$

$$m=7 \rightarrow m+r \leq 2^r - 1 \rightarrow [r=4]$$

16.

$$a) D_3 D_2 D_1 D_0 \quad m=4 \\ 1 \quad 0 \quad 0 \quad 1 \quad r=3$$

b_7	b_6	b_5	b_4	b_3	b_2	b_1
D_3	D_2	D_1	P_3	D_0	P_2	P_1
1	0	0		1		
1	0	0		1		
1	0			1	$P_1=0$	
1	0	0	$P_3=1$			
1	0	0	1	1	0	0

$$b) D_6 D_5 D_4 D_3 D_2 D_1 D_0 = 0110111 \quad m=7 \\ r=4$$

b_{11}	b_{10}	b_9	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1
D_6	D_5	D_4	P_4	D_3	D_2	D_1	P_3	D_0	P_2	P_1
0	1	1		0	1	1		1		
0	1	0		0	1	0		1	$P_1=1$	
0	1			0	1			1	$P_2=1$	
				0	1	1	$P_3=0$			
1	1	1	$P_4=1$							
1	1	1	1	0	1	1	0	1	1	1

17.

a) 1111010 $m=4$
 $r=3$

b_{11}	b_{10}	b_9	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1
				D_3	D_2	D_1	P_3	D_0	P_2	P_1
	1	.	.	1	1	1	1	0	1	0
				111	110	101	100	011	010	001
				1		1		0		$P_1 = 0$
				1	1			0	$P_2 = 0$	$\times 1$
				1	1	1	$P_3 = 1$			$\checkmark 0$
				1	1	1				

error en $B_2 \rightarrow$ dato correcto: 10110

$P_3 P_2 P_1 = 010 \rightarrow \text{pos} = D_2 \quad \underline{1}$

b) 111010001001010 $m=7$
 $r=4$

b_{11}	b_{10}	b_9	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1
D_6	D_5	D_4	P_4	D_3	D_2	D_1	P_3	D_0	P_2	P_1
0	1	1	0	0	0	1	0	1	1	1
101010	10100	10001	1000	00101	00100	0100	0100	00101	0010	0001
0	1		0	0		1		1		$P_1 = 1$ $\checkmark 0$
0	1		0	0			1	1	$P_2 = 0$	$\times 1$
0	1	1	$P_4 = 0$							$\times 1$

$P_3 P_2 P_1 = \underbrace{1110}_{14} \rightarrow \underbrace{0110}_6 \rightarrow \text{dato correcto} = 1110110$

1000010 \times
 1100110 ✓ al revés
 ↓
 error

c) 111111

b_7	b_6	b_5	b_4	b_3	b_2	b_1
D_3	D_2	D_1	P_3	D_0	P_2	P_1
111	110	101	100	011	010	001
1	1	1	1	1	1	1
1		1		1		$P_1=1$
1	1			1		\checkmark
1	1	1		$P_2=1$		0
1	1	1	$P_3=1$			\checkmark
1	1	1				0

$$P_3 P_2 P_1 = 000 \rightarrow \text{No error} \rightarrow \text{Data: } 1111$$

d) 111101011 $m=5$ $r=4$

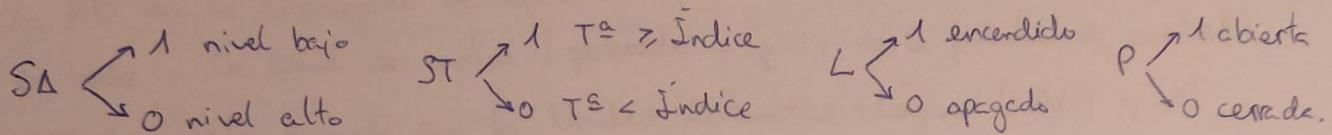
b_9	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1
D_4	P_4	D_3	D_2	D_1	P_3	D_0	P_2	P_1
1	1	1	1	0	1	0	1	1
1001	1000	0111	0110	0101	0100	0011	0010	0001
1		1		0		0		$P_1=0$
		1	1			0	$P_2=0$	$\times 1$
		1	1	0	$P_3=0$			$\times 1$
1	$P_4=1$							$\checkmark 0$

$$P_4 P_3 P_2 P_1 = 0111 \rightarrow b_7 (D_3) \rightarrow \text{Data: } 10100$$

Ejercicio repaso sistemas digitales, Tema 0, Página 26:

Diseña el sistema que active la alarma de coche cuando se dé alguno de los siguientes casos:

- a) el sensor de aceite (SA) indica nivel bajo
- b) el sensor de temperatura (ST) indica nivel alto
- c) luces (L) encendidas y puerta (P) abierta.



SA	ST	L	P	Y alarma
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

$$y = \overline{11} M(0, 1, 2)$$

$$y = (SA + ST + L + P)(SA + ST + L + P^1) \\ \cdot (SA + ST + L^1 + P)$$

KARNAUGH

		L, P			
		00	01	11	10
SA, ST	00	0	0	1	0
	01	1	1	1	1
	11	1	1	1	1
	10	1	1	1	1

Minterms (1): $SA + LP + ST$

Maxterms (0): $(SA + ST + L)(SA + ST + P)$

UNIDAD CENTRAL DE PROCESOS (CPU)

La componen la UNIDAD de CONTROL y la RUTA de DATOS.

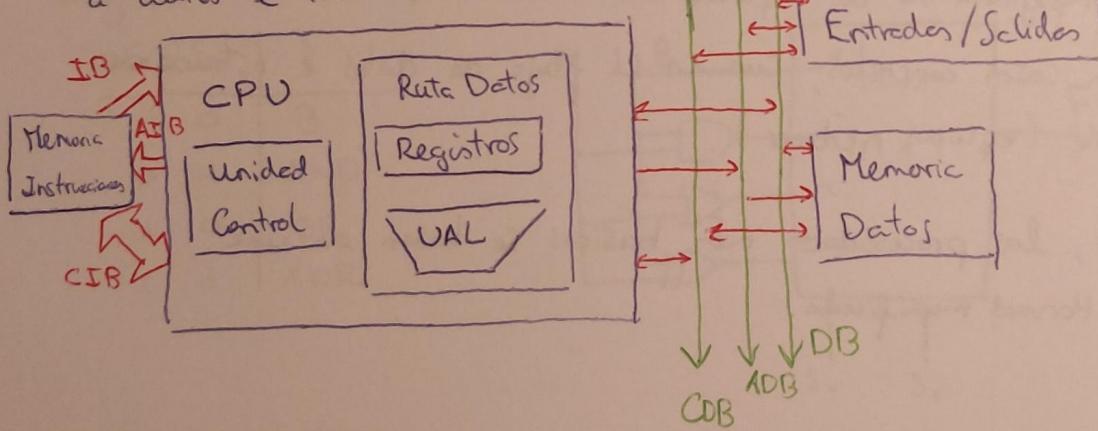
La UC es la parte inteligente, encargada de generar señales de selección que dirigen el funcionamiento de la ruta de datos. Es decir, genera la palabra de control para traducir la instrucción para la ruta de datos.

La RD es la encargada de ejecutar operaciones aritméticas y lógicas en base a la señal de control (generada a partir de la palabra de control) recibida. Está formada por la Unidad Aritmética - Lógica (UAL) y el bloque de registros de la CPU.

ARQUITECTURAS

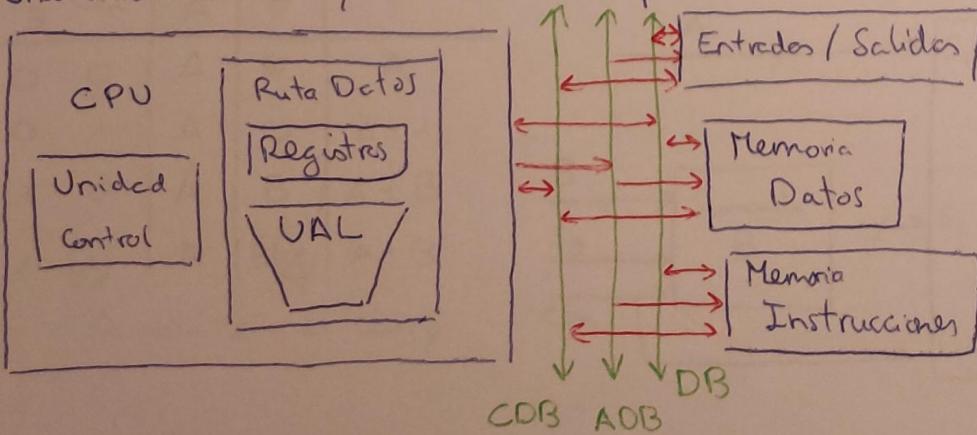
HARVARD

Memorias separadas para instrucciones y datos. Instrucciones y datos tienen sus propios buses, lo que permite un acceso simultáneo a datos e instrucciones.



VON NEUMANN

Una única memoria y un único bus para instrucciones y datos.



Harvard VS. Von Neumann

La arquitectura Von Neumann, basada en un único espacio de memoria, por lo que imposibilita leer una instrucción mientras se ejecuta otra.

Únicamente tiene un bus para direccionar tanto instrucciones como datos, esto es, los mismo buses direccionan ambas información, buses competitivos. Además, todas las instrucciones, incluso las de movimiento de datos, tienen que pasar por la UAL, provocando un cuello de botella.

Por su parte, las características de la arquitectura Harvard contraponen las de Von Neumann, permitiendo un acceso simultáneo a instrucciones y datos, y por lo tanto, logrando mayor velocidad de procesamiento.

Sin embargo, a pesar del cuello de botella que dificulta la programabilidad e impide el paralelismo, la arquitectura Von Neumann es la más extendida por:

- el amplio conjunto de programas desarrollados para Von Neumann -
- la complejidad de la arquitectura Harvard, que solo le hace rentable en casos concretos, cuando el flujo de datos e instrucciones es comparable (equipos médicos).

Hoy en día, los procesadores más básicos se basan en una arquitectura Harvard modificada.

RUTA de DATOS

Está formada por registros (Bloque de Registros), multiplexores, sumador, desplazador... Esencialmente está compuesta por un Sumador, un desplazador y puertas lógicas (UAL).

Son elementos combinacionales, que dan un resultado u otro en función de los señales de selección generadas por la Unidad de Control.

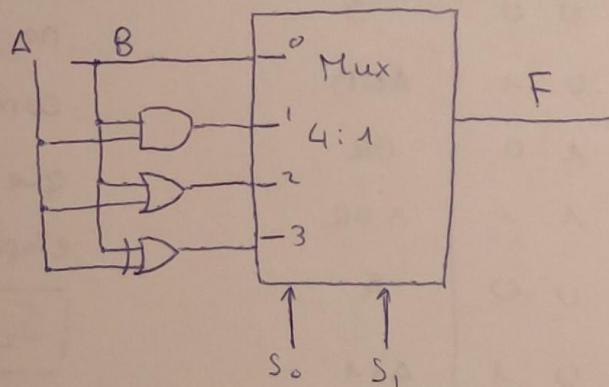
Para simplificar, diremos que el Bloque de Registros y la UAL forman la Ruta de Datos.

P.A. 1.1.

- ① Diseña una ALU que teniendo como entradas los números A y B de cuatro bits ($A(A_3A_2A_1A_0)$, $B(B_3B_2B_1B_0)$), realice las siguientes operaciones:

Lógicos: AND ($A \cdot B$) ; OR ($A+B$) ; XOR ($A \oplus B$) ; B

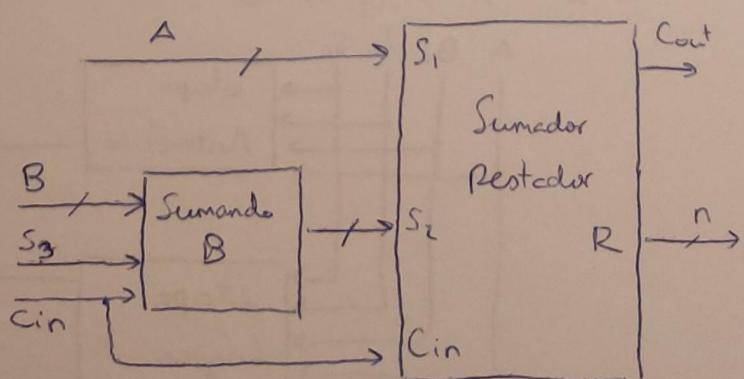
S_1, S_0	F
0 0	B
0 1	AND
1 0	OR
1 1	XOR



Aritméticos: A; A+1; A+B; A-B

S_3, Cin	A	B	F
0 0	A	0	A
0 1	A	0	$A+1$
1 0	A	B	$A+B$
1 1	A	\bar{B}	$A-B$

Ya está en el carry



Modificación del sumando B:

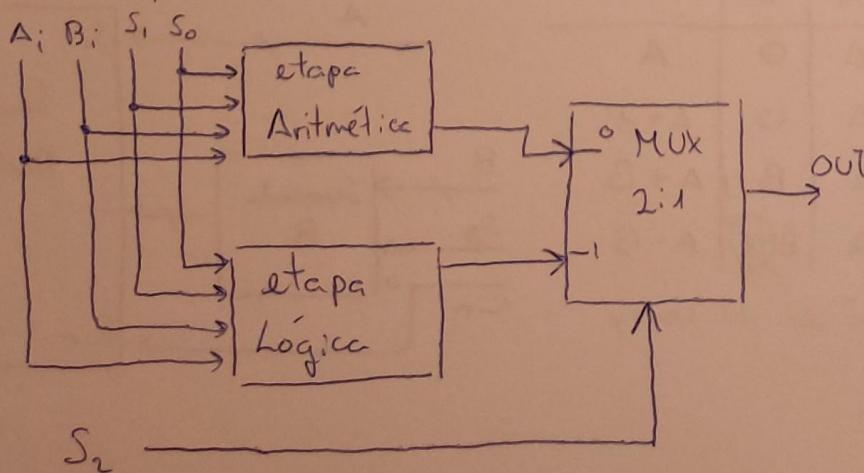
S_3	Cin	B_i	F_i
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Se han utilizado 4 señales de selección, Sin embargo, para realizar 8 operaciones son necesarios únicamente 3; $2^3 = 8$

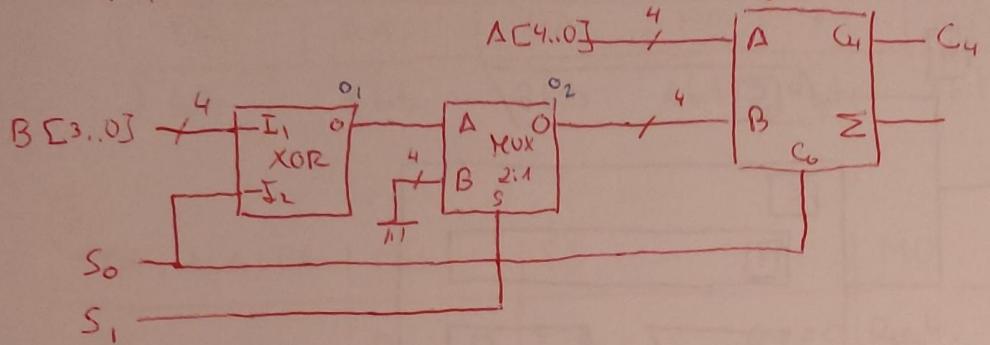
Cin	S_1, S_0	Operación
0	0 0	B
0	0 1	AND
0	1 0	OR
0	1 1	XOR
1	0 0	A
1	0 1	$A+1$
1	1 0	$A+B$
1	1 1	$A-B$

Mediante esta simplificación, no sería correcto identificar el S_2 como Cin , dado que el bit que actuará como carry en la etapa aritmética será el S_0 .

$[S_2 \ S_1 \ S_0 (Cin)]$



② Identifica y pon en la tabla las funciones que realiza la pequeña unidad aritmética de la figura:



S_1	S_0	O_1	O_2	Suma
0	0	B	0,	$A+B+0 = A+B$
0	1	\bar{B}	0,	$A+\bar{B}+1 = A-B$
1	0	B	0	$A+0+0 = A$
1	1	\bar{B}	0	$A+1+0 = A+1$

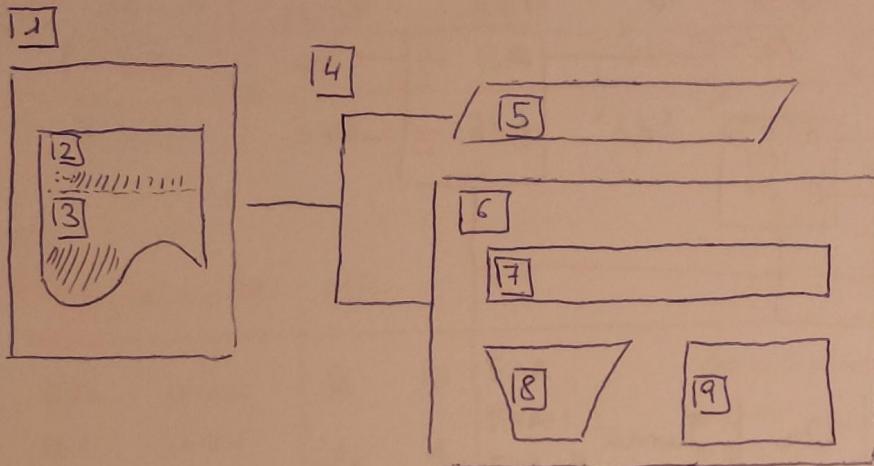
* XOR

X	Y	S
0	0	0
0	1	1
1	0	1
1	1	0

Invierte la Y.

P.A. 1.2.

①



1- Memoria (única)

2- Memoria datos (área)

3- Memoria instrucciones (área)

4- Bus del sistema

5- Dispositivo de E/S.

6- CPU

7- Bloque de Registros

8- UAL

9- Unidad de Control

② Enumerar los principales características de la arquitectura Von Neumann:

1. Una única memoria para datos e instrucciones.
2. Un único bus para direccionar tanto datos como instrucciones.
3. Todas las instrucciones, incluso las de movimiento de datos, tienen que pasar por la UAL, generando un cuello de botella.
No obstante, a pesar de ello, es la arquitectura más usada, en parte por su simplicidad en comparación con otras arquitecturas como la Harvard, y por tanto, por su menor coste.

③ ¿Qué es la palabra de control? ¿Quién la utiliza y para qué?

Es una secuencia de bits que utiliza la Ruta de Datos para dirigir el funcionamiento de la UAL. Es la combinación de los señales de selección (16 bits).

④ ¿Quién genera la palabra de control? ¿A partir de qué información?

La palabra de control la genera la Unidad de Control a partir de la instrucción dada.

Palabra de Control (P.A. 1.2)

15-13	12-10	9-7	6	5-2	1	0
DA	AA	BA	MB	FS	MD	RW
3 bits	3 bits	3 bits	1 bit	4 bits	1 bit	1 bit

DA, A ₂ A, BA	MB	FS	MD	RW
R0 000	R 0	F=A F=A+1	0000 0001	Result. 0
R1 001	Cte 1	F=A+B	0010	Escribir 1
R2 010	MB: escoger entre valor de un registro	F=A+B+1	0011	
R3 011		F=A+B	0100	RW: habilitar la escritura en el registro seleccionado
R4 100	o una constante	F=A+B+1	0101	
R5 101		F=A-1	0110	
R6 110		F=A	0111	
R7 111		F=A+B	1000	
DA		F=A+B	1001	
AA		F=A	1010	
BA/OS:		F=B	1011	
		F=S ₁ B	1100	
		F=S ₁ L B	1101	

?	Microoperación	DA	AA	BA	MB	FS	MD	RW
?	(R ₄) ← S ₁ (R ₆)	100	XXX	110	0	1110	0	1
?	(R ₁) ← (R ₀) + 2	001	000	XXX	1	0010	0	1
?	Bs ₁ lida - dato ← (R ₃)	XXX	XXX	011	0	XXXX	X	0
?	(R ₁) ← (R ₂) + (R ₃) + 1	001	010	011	0	0101	0	1
?	(R ₇) ← (R ₇) + 1	111	111	XXX	X	0001	0	1
?	(R ₄) ← entrada de datos.	100	XXX	XXX	X	XXXX	1	1

UNIDAD de CONTROL

La Unidad de Control genera a partir del conjunto de bits de una INSTRUCCIÓN la secuencia de PALABRAS de CONTROL necesarias para su ejecución y mandarlos por la RUTA de DATOS. Está compuesta por elementos combinacionales.

Instrucciones: conjunto de bits dividido por CAMPOS:

- Código de Operación: m bits $\rightarrow 2^m$ códigos

- Dirección: direcciones de los operandos.

15	9 8	6 5	3 2	0	CAMPOS
Cód. op. (7bits)	3 bits	3 bits	3 bits		

FORMATOS

• Registro:

15	9 8	6 5	3 2	0
Código Operación	Registro de Destino (DA) DR	Registro Fuente A (AA) SA	Registro Fuente B (BA) SB	

• Inmediato:

15	9 8	6 5	3 2	0
Código Operación	Registro de Destino (DA) DR	Registro Fuente A (AA) SA	Operando (OP) OP	

• Salto y bifurcación:

15	9 8	6 5	3 2	0
Código Operación	Registro de Destino (DA)	Registro Fuente A (AA)	Operando (OP)	

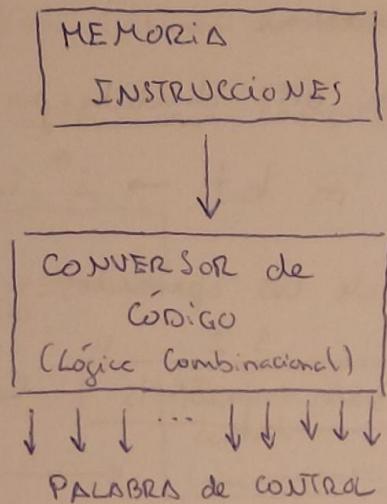
AD SA AD

dirección en complemento a 2:
 $2^6 [32, 31]$

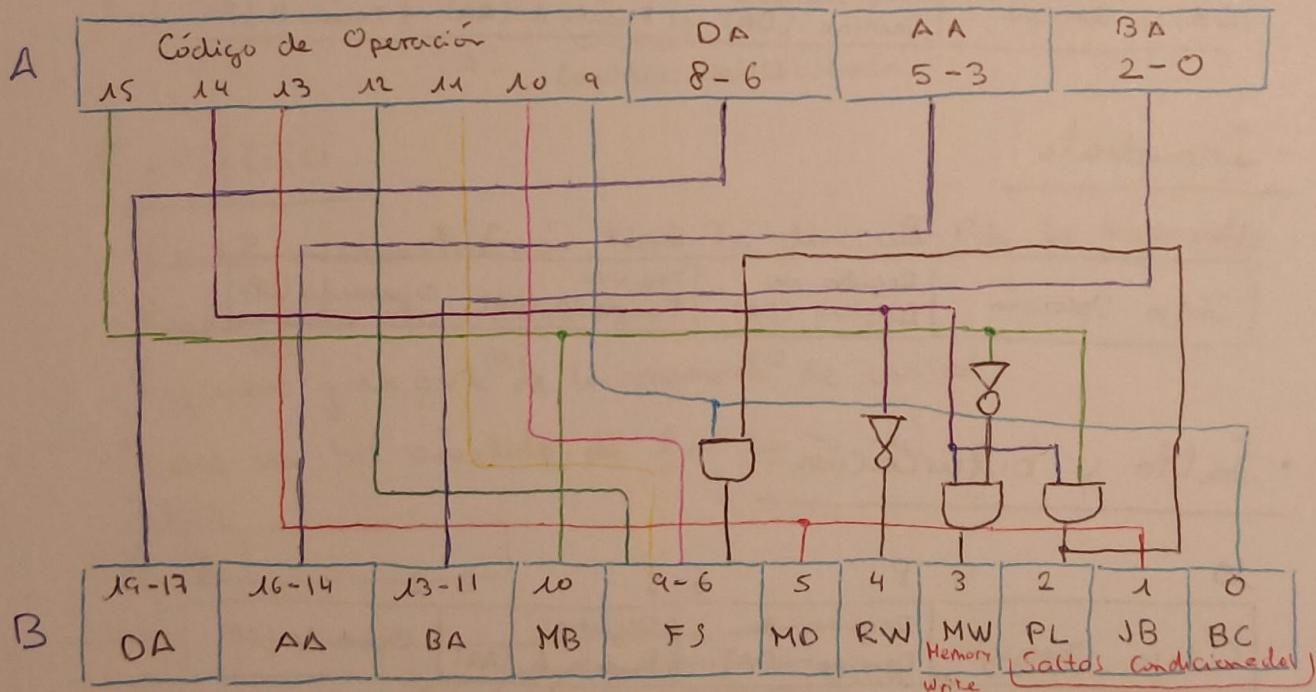
TIPOS / MÉTODOS :

CONTROL CABLEADO

- Diseño laborioso y costoso.
- Difícil de modificar, sin posibilidad de rediseño.



Conversor de código:



$$B_0 = \bar{A}_9$$

$$B_5 = A_{13}$$

$$B_{10} = A_{15}$$

$$B_{15} = A_4$$

$$B_1 = A_{13}$$

$$B_6 = A_9 \cdot (\bar{A}_{14} \cdot A_{15})$$

$$B_{11} = A_0$$

$$B_{16} = A_5$$

$$B_2 = A_{14} \cdot \bar{A}_{15}$$

$$B_7 = A_{10}$$

$$B_{12} = A_1$$

$$B_{17} = A_6$$

$$B_3 = A_{14} \cdot \bar{A}_{15}$$

$$B_8 = A_{11}$$

$$B_{13} = A_2$$

$$B_{18} = A_7$$

$$B_4 = \bar{A}_{14}$$

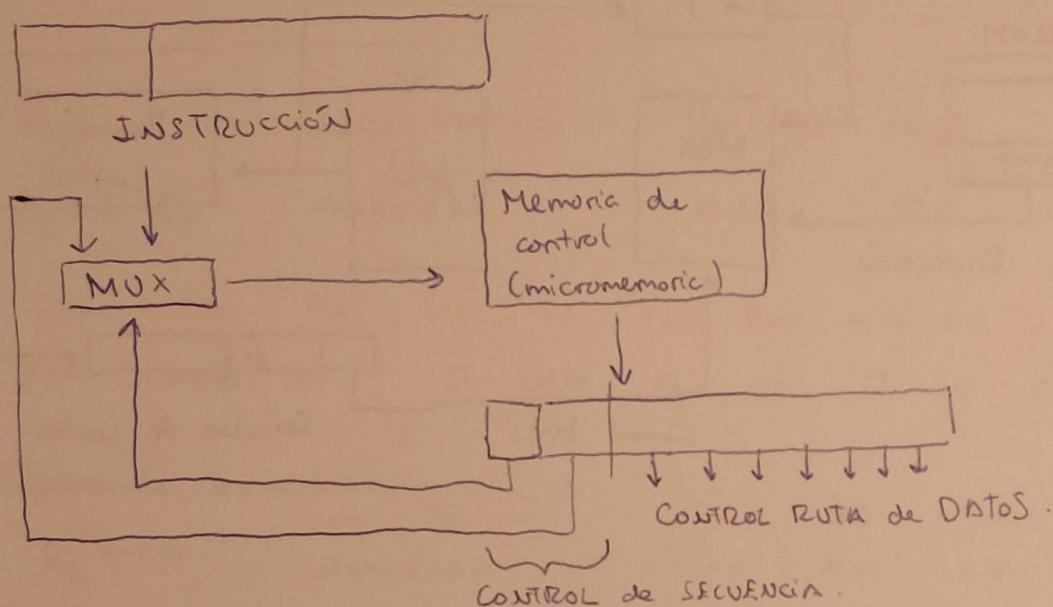
$$B_9 = A_{12}$$

$$B_{14} = A_3$$

$$B_{19} = A_8$$

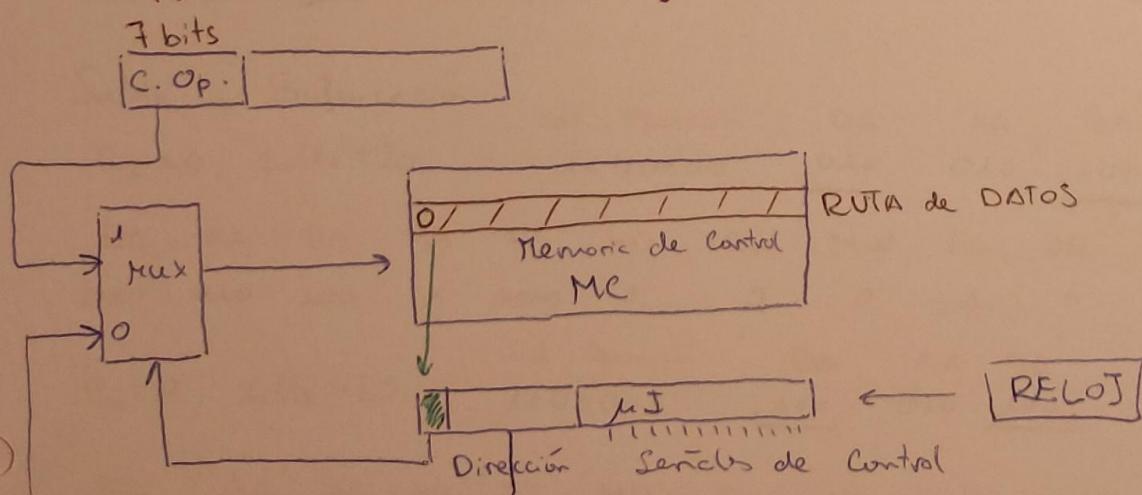
CONTROL MICROPROGRAMADO

- Una instrucción → un microprograma = conjunto de microinstrucciones ordenadas.
- Microcódigo en memoria (firmware)
- Fácil de modificar (ahorro en diseño hardware → + firmware)
- Pueden incluir instrucciones complejas.



EXPLÍCITO :

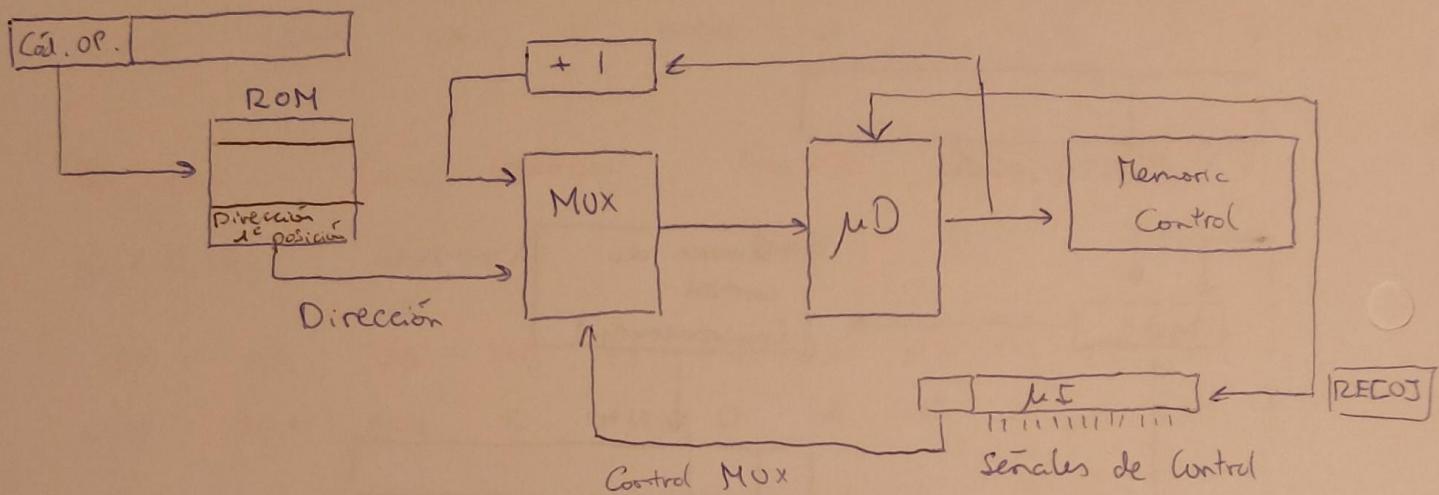
- Cada microinstrucción tiene la dirección de la siguiente.
- Pueden estar desordenadas.
- Requiere gran parte de la memoria de control.
- Tiene un bit añadido de fin (sí o no)



- 0: no metas más códigos que no he terminado
- 1: terminado, se introduce nuevo código operación.

IMPLÍCITO:

- Cada microinstrucción está almacenada en orden .
- Conversor de código (ROM, PLA) : puede que no coincidan los bits del código operación .
- Registro microdirecciones e incrementador .



Práctica de Aula 1.3.

①

Registros

Código Operación

Reg Disp.
DA

Reg A
AA

Reg B
BA

$R_2 \leftarrow R_5$

000 000 0

010

101

XXX

DA	AA	BA	MB	FS	MD	RW	MW	PL	JB	BC
010	001	xxx	0	0000	0	1	0	0	0	0

Registros

Código Operación

Reg Disp.
DA

Reg A
AA

Reg B
BA

$R_2 \leftarrow R_1 + R_5$

0000010

010

001

101

DA	AA	BA	MB	FS	MD	RW	MW	PL	JB	BC
010	001	101	0	0010	0	1	0	0	0	0

Operando inmediato:

Código Operación

$R_2 \leftarrow S$

1001100

DA

AA

BA

010

xxx

101

DA	AA	BA	MB	FS	MD	RW	MW	PL	JB	BC
010	xxx	101	1	100	0	1	0	0	0	0

Código Operación

$R_2 \leftarrow R_1 + S$

1000010

DA

AA

BA

010

001

101

DA	AA	BA	MB	FS	MD	RW	MW	PL	JB	BC
010	001	101	1	0010	0	1	0	0	0	0

Salto y Bifurcación:

cód. Operación

$R_2 = 0$, salto +20

1100000

010

010

100

010100 (20)

DA	AA	BA	MB	FS	MD	RW	MW	PL	JB	BC
010	010	100	1	0000	0	0	0	1	0	0

Cód. Operación

$R_2 = 0$, salto -20

1100001

101

010

100

101100 ($C_2 C_{20}$)

DA	AA	BA	MB	FS	MD	RW	MW	PL	JB	BC
101	010	100	1	0001	0	0	0	1	0	1

②	19-12 DA	16-14 AA	13-11 BA	10 MB	9-6 FS	5 MD	4 RW	3 MW	2 PL	1 JB	0 BC
	011	010	011	1	0010	0	1	0	0	0	0

$$B_0 = A_9 = 0$$

$$B_8 = A_{11} = 0$$

$$B_1 = A_{13} = 0$$

$$B_9 = A_{12} = 0$$

$$B_2 = A_{14} \cdot A_{15} = 0$$

$$B_{10} = A_{15} = 1$$

$$B_3 = A_{14} \cdot \overline{A_{15}} = 0 \quad 1 \cdot \overline{A_{15}} = 0$$

$\overline{A_{15}} = 0 \rightarrow A_{15} = 1$

$$B_4 = A_{14} = 1$$

$$B_{11} = A_0 = 1$$

$$B_5 = A_{13} = 0$$

$$B_{12} = A_1 = 1$$

$$B_6 = A_9 \cdot (\overline{A_{15}} \cdot \overline{A_{15}}) = 0$$

$$B_{13} = A_2 = 0$$

$$B_7 = A_{10} = 1$$

$$B_{14} = A_3 = 0$$

$$B_{15} = A_4 = 1$$

$$B_{16} = A_5 = 0$$

$$B_{17} = A_6 = 1$$

$$B_{18} = A_7 = 1$$

$$B_{19} = A_8 = 0$$

Código Operación .

11 00010

DA AA BA

011 101 011

PIPE-LINE : Computadora en canalización .

IF: Instruction Fetch
buscar la instrucción

DOF: Decode and Operand Fetch
se buscan operandos

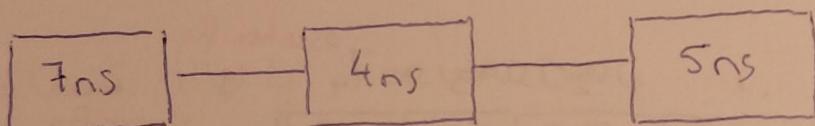
EX: Execution

se hace la operación (resultado en la salida de la ALU; circuito combinacional)

WB: Write back

se escribe en el registro .

Sin canalización :

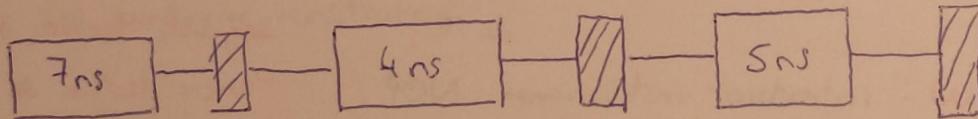


$$f = \frac{1}{T_{\max}} = \frac{1}{T_{1 \text{ instrucción}}}$$

$$T_{\max} = T_{1 \text{ instrucción}}$$

cada etapa toma un tiempo

Con canalización :



cada etapa toma su tiempo + el de la plataforma.

$T_{\text{etapa m\'axima}}$: nos dar\'a la velocidad (en este caso: 7ns + pl\'atofrma)

Tiempo n instrucciones: $(n^{\circ} \text{ instrucciones} - 1 + n^{\circ} \text{ etapas ins}) \cdot T_{\max}$

Tiempo medio l instrucción: $\frac{(n^{\circ} \text{ instrucciones} - 1 + n^{\circ} \text{ etapa})}{n^{\circ} \text{ instrucciones}} \cdot T_{\max}$

* si $n^{\circ} \text{ instrucciones} \rightarrow \infty \Rightarrow \text{Tiempo medio l instrucción} = T_{\max}$.

$$f = \frac{1}{T_{\max}} \quad ! T_{\max} \neq T_{1 \text{ instrucción}} !$$

Ejecución en concurrencia:

	T _i	7 ns	14 ns	21 ns
IF	1 ^a instrucción	2 ^a instrucción	3 ^c instrucción	4 ^a instrucción
DOF		1 ^a instrucción	2 ^c instrucción	3 ^a instrucción
EX			Cálculo 1 ^a Inst.	Cálculo 2 ^a Inst.
WB				Escritura 1 ^c Instrucción.

Periodos de reloj
Instrucciones

	1	2	3	4	5	6
1	IF	DOF	Ex	WB		
2		IF	DOF	Ex	WB	
3			IF	DOF	Ex	WB

Riesgo de datos:

	1	2	3	4	5	6
MOV R1, RS	$R_1 \leftarrow R_S$	IF	DOF	Ex	WB	escribir R_1
ADD R2, R1, RC	$R_2 \leftarrow R_1 + R_C$	IF	DOF	Ex	WB	escribir R_2
ADD R3, R1, R2	$R_3 \leftarrow R_1 + R_2$	IF	DOF	Ex	WB	

↓
Primer lectura de R_1 ↓
Segunda lectura de R_1

Solución SW: introducir instrucciones NOP.

Primer lectura de R_2

	1	2	3	4	5	6	7	8
MOV R1, RS	$R_1 \leftarrow R_S$	IF	DOF	Ex	WB	escribir R_1		
NOP	-	IF	DOF	Ex	WB			
ADD R2, R1, RC	$R_2 \leftarrow R_1 + R_C$	IF	DOF	Ex	WB			
NOP	-	IF	DOF	Ex	WB			
ADD R3, R1, R2	$R_3 \leftarrow R_1 + R_2$	IF	DOF	Ex	WB			

↓
Primer lectura de R_1

Solución HW 1: en lugar de introducir instrucciones NOP, detener la cancelación para evitar el riesgo de datos.

En las siguientes etapas se dice que se ha introducido una BURBUJA en la cancelación.

En cuanto al retraso la penalización es la misma que en la solución SW.

MOV R₁, R₃

#1	#2	#3	#4		
#1	①	#2	#3	#4	
	①	#1	#2	#3	#4

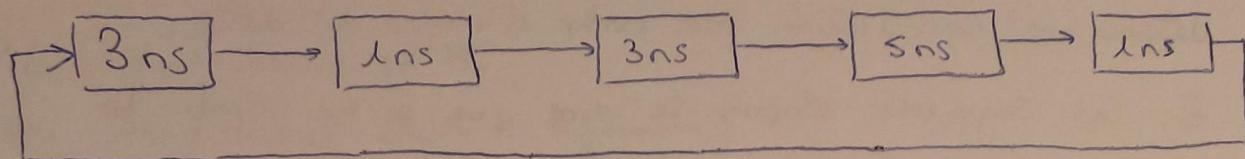
ADD R₂, R₁, R₆

Solución HW 2: FORWARDING.

Detectar el riesgo de datos y mediante un multiplexor acercar la salida de la UAL al segmento que lo necesita. Con el MUX podemos seleccionar entre un operando nuevo o el resultado de la anterior instrucción.

P. 1.4.

①



a) ¿Cuánto tardaré en ejecutar 5 instrucciones?

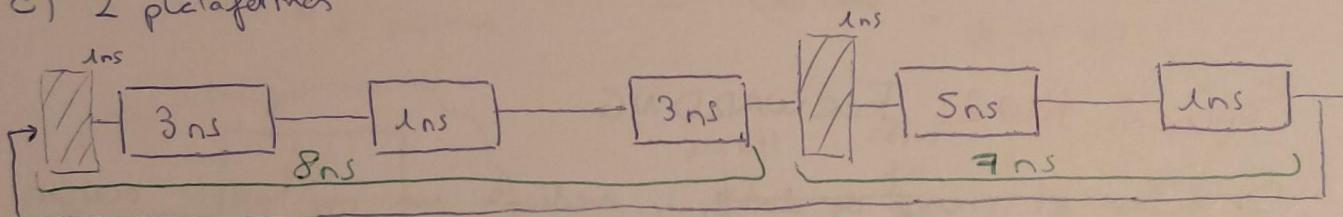
Todavía este sin conciliar. 1 instrucción tardará: $3+1+3+5+1 = 13 \text{ ns}$.

$$5 \text{ instrucciones: } 5 \cdot 13 = 65 \text{ ns}.$$

b) ¿Frecuencia máxima del reloj?

$$f = \frac{1}{T_{\text{ins}}} = \frac{1}{13 \cdot 10^{-9}} = 76'9 \text{ MHz}.$$

c) 2 plataformas



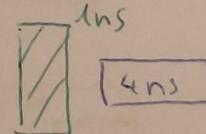
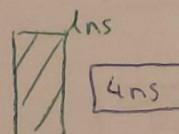
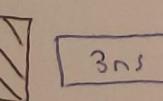
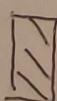
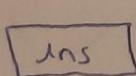
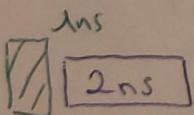
$$\text{d) 1 instrucción: } (n^{\circ} \text{ inst} - 1 + n^{\circ} \text{ etapas}) \cdot T_{\text{máx}} = (1-1+2) \cdot 8 = 16 \text{ ns}.$$

$$\times 5 \text{ instrucciones: } 16 \cdot 5 = 80 \text{ ns. } \times$$

$$\text{Tiempo crítico} = 8 \text{ ns.}$$

$$\begin{aligned} \text{Tiempo 5 inst} &= 8 \text{ ns} \cdot (n^{\circ} \text{ inst} - 1 + n^{\circ} \text{ etapas}) = 8 \cdot (5-1+2) \\ &= 8 \cdot 6 = 48 \text{ ns} \end{aligned}$$

②



$$\text{a) } 2+1+3+4+4 = 14 \text{ ns} \rightarrow f = \frac{1}{14 \cdot 10^{-9}} = 71'43 \text{ MHz}.$$

$$\text{b) } t_{\text{inst}} = T_{\text{máx}} \cdot n^{\circ} \text{ etapas} = 7 \cdot 3 = 21 \text{ ns etapas críticas: } 7 \text{ ns}$$

$$f = \frac{1}{T_{\text{máx}}} = \frac{1}{7 \cdot 10^{-9}} = 142'85 \text{ MHz. Tiempo 5 instrucciones: } 7 \cdot (5 - 1 + 3) =$$

$$t_{\text{inst}} = \frac{49 \text{ ns}}{5} = 9'8 \text{ ns. } = 7 \cdot 7 = 49 \text{ ns}$$

$$\text{c) } t_{\text{inst}} = 18 \text{ ns}$$

$$\text{Tiempo crítico} = 5 \text{ ns.}$$

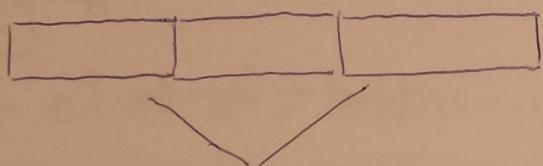
P.A. 1.5.

PL : Load Enable for PC.

JB: Jump Branch (Salto a bifurcación)

. BC: Branch Condition Select (Nº 7)

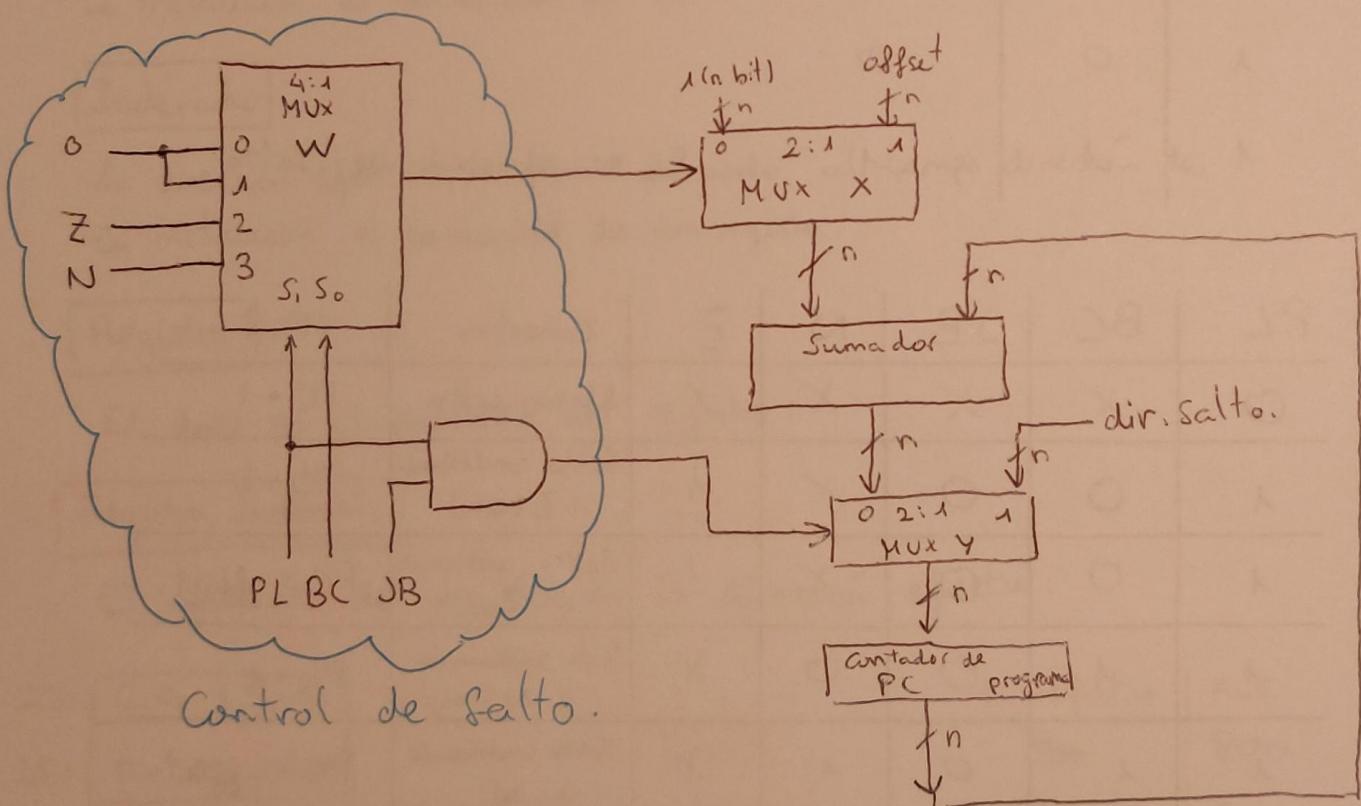
Offset: Dirección A + Dirección B ; a dónde saltar.
(3 bits) (3 bits)



Dirección de salto: salto incondicional (Jump)

N : signo $\left\{ \begin{array}{l} 1: \text{negativo} \\ 0: \text{positivo} \end{array} \right.$

Z: does zero? ↗ 1: zero
 2: no zero.



PL	BC	W mux out	X mux out
0	0	0	1
0	1	0	1
1	0	Z ↗ ⁰	1 offset
1	1	N ↗ ⁰	1 offset

$x \rightarrow$ Sumador PC + 1 ; offset \rightarrow Sumador PC + offset.

PL	JB	Y mux out	
0	0	0	PC \leftarrow salida sumador
0	1	0	
1	0	0	
1	1	1	PC \leftarrow dirección de salto

PL	BC	JB	N	Z	Comentario	¿PC?
0	X	X	X	X	No hay salto	PC + 1
1	0	0	X	0	Salto condicionado a Z (cero)	PC + 1
1	0	0	X	1	Salto condicionado a Z	PC + offset.
1	1	0	0	X	Salto condicionado a N (negativo)	PC + 1
1	1	0	1	X	Salto condicionado a N	PC + offset
1	X	1	X	X	Salto incondicional	Nueva dirección

MODOS de DIRECCIONAMIENTO

Directo

El campo dirección de la instrucción contiene la dirección efectiva en la que se encuentra el dato.

Inmediato

El campo dirección de la instrucción contiene la información (el dato).

Indirecto

El campo dirección de la instrucción contiene la dirección de memoria en la que se encuentra la dirección efectiva.

Relativo (a PC)

La dirección efectiva se forma sumando al campo dirección de la instrucción el contenido de PC.

Indexado

La dirección efectiva se forma sumando al campo dirección de la instrucción el contenido de un registro.

Registro

El dato es el contenido del registro.

Registro Indirecto

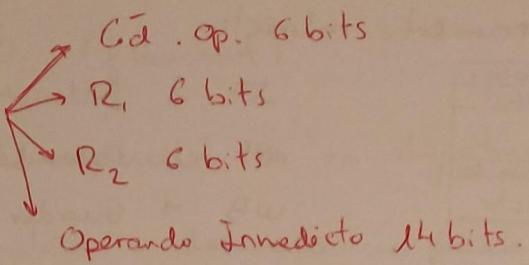
El contenido del registro es la dirección efectiva.

C. Op	Modo
250	
251	Dir. /Op. = 500
252	Siguiente instrucción
400	700
500	800
752	600
800	300
900	200

Dir. Efectiva	ACC
500	800
251	500
800	300
500 + 252 = 752	600
500 + 400 = 900	200
— (R1)	400
400	700

P. A. 2.1.

① Instrucciones 32 bits



a) Número máximo operaciones : Código Operación 6 bits

$$2^6 = 64 \text{ operaciones}$$

b) R₁ + R₂ = 6 + 6 = 12 bits

$$2^{12} = 4096 \text{ registros}$$

c) Sin Signo : $[0, 2^{14}-1] = [0, 16.383]$

d) Signo y Magnitud : $[-(2^{14}-1), 2^{14}-1] = [-2^{13}+1, 2^{13}-1]$

Complemento a 2: $[-2^{14}, 2^{14}-1] = [-2^{13}, 2^{13}-1]$

②

a) 110 instrucciones

$$2^m \geq 110, m \in \mathbb{Z} \Rightarrow m=7 \rightarrow 2^7 = 128$$

Código Operación tendrá 7 bits.

b) Memoria de 32 bits

Bloque de 32 registros \rightarrow Operando tendrá 5 bits $\rightarrow 2^5 = 32$

c) Sin Signo: $[0, 2^5-1] = [0, 31]$. $\rightarrow 6 \text{ bits} \rightarrow 2^6 = 64$
 $[0, 63]$

d) Complemento a 2:

5 bits : $[-2^{5-1}, 2^{5-1}-1] = [-2^4, 2^4-1] = [-16, 15]$

6 bits : $[-2^{6-1}, 2^{6-1}-1] = [-2^5, 2^5-1] = [-32, 31]$

③ Instrucciones 32 bits.

Cód Op. empieza en 0 (4 bits) : 2^3 (quedan 3 bits)

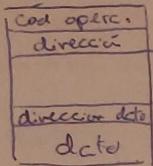
$$2^3 + 2^4 + 2^6 = 88$$

Cód Op. empieza en 10 (6 bits) : 2^4 (quedan 4 bits)

11 (8 bits) : 2^6 (quedan 6 bits)

códigos operación.

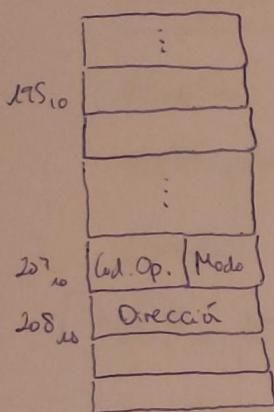
④ Direcciónamiento indirecto



- IF 2 Instruction Fetch
OF 2 Operando Fetch
WB 1 Guardar dato

5 accesos a memoria

⑤ Memoria



a) Direcciónamiento relativo

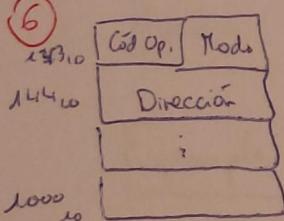
$$PC + \text{Dirección} = 195$$

$$209 + \text{Dirección} = 195$$

$$D_i = -14_{10}$$

b) $1110_2 \rightarrow C_2 : 0010_2 \xrightarrow{(6\text{bit})} 110010_2$

⑥

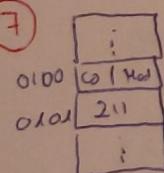


$$PC + \text{dir} = 1000$$

$$1000 - 144 = \text{dir} = 858_{10}$$

$$0358_{16}$$

⑦



$$R_i = 189$$

Direcciónamiento	Directo	Indirecto	Relativo	Registro Inmediato	Indexado
Dir. efectiva	211	101	$211+102_{313}$	189	$189+211_{400}$
Dato	?	211	?	?	?

⑧

$$A = 35_{16}$$

$$B = B9_{16}$$

$$0011 \quad 0101$$

$$1011 \quad 1001$$

$$\boxed{\text{AND}} : 00110001 = 31_{16}$$

$$\boxed{\text{OR}} : 10111101 = B0_{16}$$

$$\boxed{\text{XOR}} : 10001100 = 8C_{16}$$

$$9) \quad \begin{array}{cccccccccc} 15 & 14 & 13 & 12 & 11 & 10 & 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\ 10 & 10 & 10 & 10 & 01 & 01 & 11 & 11 & 00 & 00 & 00 & 00 & 00 & 11 & 11 & 11 & 11 \end{array}$$

a) $\boxed{\text{OR}}$

$$\begin{array}{r} 1010100111111111 \\ 0000000011111111 \\ \hline 1010100111111111 \end{array}$$

c) $\boxed{\text{AND}}$

$$101010010111100$$

$$0101010101010101$$

$$\hline 0000000101010100$$

b) $\boxed{\text{XOR}}$

$$\begin{array}{r} 101010010111100 \\ 1010101010101010 \\ \hline 0000001111010110 \end{array}$$

A	B	\oplus
0	0	0
0	1	1
1	0	1
1	1	0

$$= 03D6_{16}$$

DIAGRAMAS de FLUJO (flowchart)

Representa los pasos y la evolución de un algoritmo o proceso.

Se utilizan para diseñar, analizar o documentar un proceso o programa.

Permiten localizar fallos o cuellos de botella.

Bloques estructurales:

elipse: bloque de comienzo o fin.

rectángulo: proceso genérico, actividad.

rombo: decisión (T/F)

rectángulo con rayas en ambos lados: subrutinas

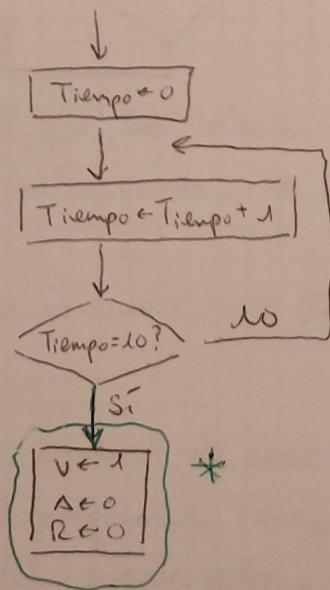
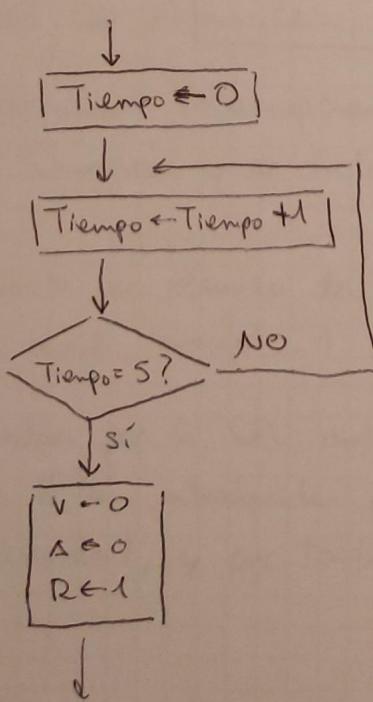
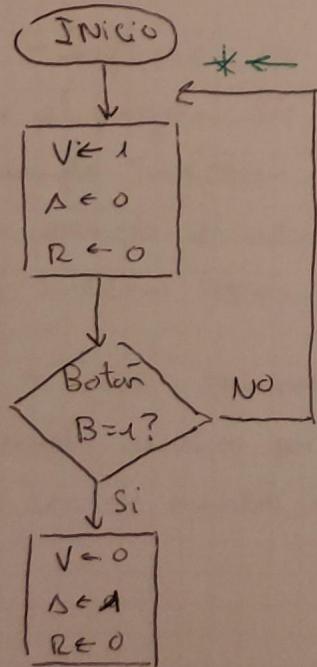
paralelogramo: entrada / salida.

Flécha: flujo del control.

Ejemplo:

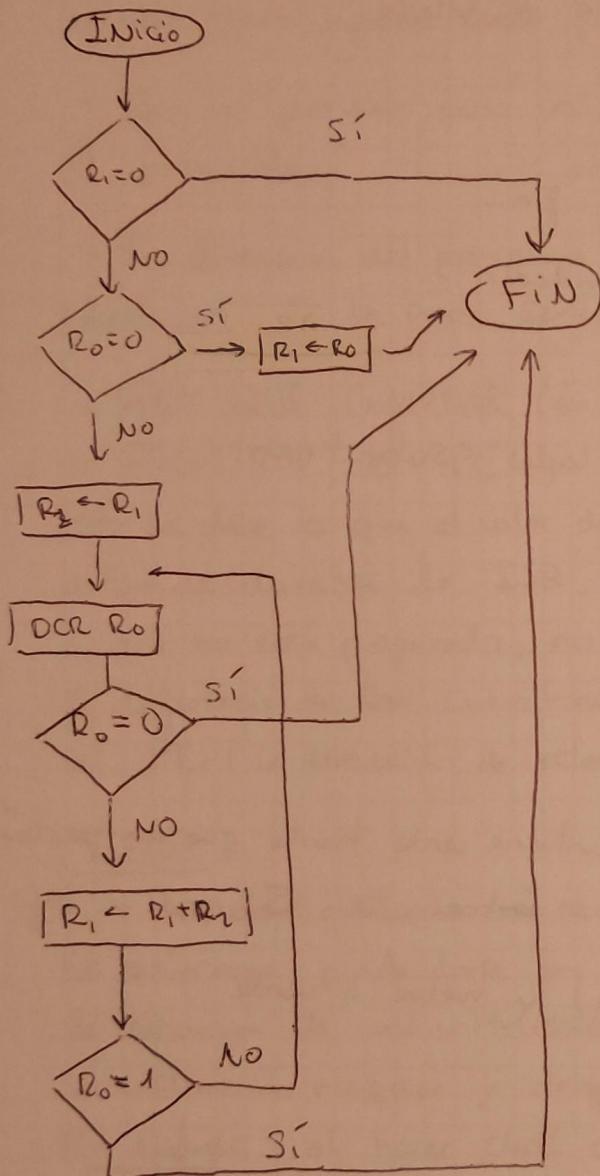
Un semáforo (para vehículos) se mantiene verde hasta que un peatón pulse el botón de solicitud. Pasa a ámbar, tras 5s pasa a rojo (los peatones tienen 10s para cruzar) y vuelve a verde.

$$f = 1 \text{ Hz} = \frac{1}{T} \rightarrow T = 1 \text{ s.}$$

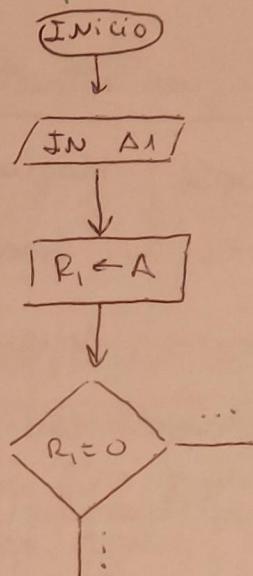


P. A. 2.2.

1. Realiza el diagrama de flujo que realiza la multiplicación en base a la suma. Los números están en los registros (R_1 y R_0). El resultado debe quedar en R_1 . Los valores son positivos.



'desde puertos externos':



INTERRUPCIONES

Estas suceden cuando se rompe al ejecución normal del programa para ejecutar la rutina de Servicio (ISR) con el fin de atender la interrupción, volviendo tras esto a la ejecución normal del programa, al punto en el que estaba (con la misma información, en el mismo estado tanto registros como flags) cuando sucedió la interrupción.

- No la provoca una instrucción, sino una señal (externa/interna) no previsible.
- La dirección del programa de atención (ISR) se facilita mediante hardware. Es la propia señal quien la indica.
- HAY QUE GUARDAR (en la pila) LA INFORMACIÓN del BLOQUE de REGISTROS, el PC y LOS FLAGS de ESTADOS (SR, Status Register). Esto se debe a que el valor de los registros y los flags puede alterarse durante la ISR, pero como es algo repentino, no deseado o que no esté preparado, no controlamos qué cambios puede producir. A diferencia de las subrutinas, en estas solo es imprescindible guardar el PC (la dirección de retorno).

TíPOS:

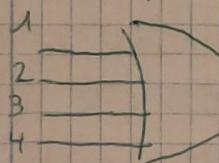
- Hardware: cuando un recurso hardware requiere atención. Es asincrónica, puede darse en cualquier momento, en general, durante la ejecución de una instrucción. En ese caso, primero se termina la instrucción máquina y después se atiende la ISR.
Por ejemplo: al hacer click con el mouse.
- Software: no son propiamente interrupciones ya que son introducidas por el programador. Son sincrónas y se trata de instrucciones. Se sabe cuando sucederán.
Por ejemplo: el usuario solicita un recurso del núcleo mediante una llamada al sistema (open, write, read, mount...)

- Traps: trampas generadas por la CPU ante una condición de error (por ejemplo, división por cero). Están introducidas por el programador (aunque no sea de manera intencionada), y por tanto, están sincronizadas.

¿CÓMO SABE LA CPU QUIÉN SOLICITA ISR?

- Multinivel: Cada dispositivo tiene una entrada asociada. Sencillo de gestionar pero muy caro.
- Línea única más "polling": una línea única indica a la CPU que algún dispositivo requiere atención, por lo que la CPU realiza un sondeo.
- Vectorizadas: el dispositivo que interrumpe no solo activa la señal de Solicitud de Interrupción, sino que además pone en el Bus de DATOS un vector que lo identifica.

DISPOSITIVOS:



Instrucción terminada

EI : interrupciones habilitadas

Interrupción hardware
vectorizada

INTACK : reconocimiento de la interrupción

IVAD : dirección del vector
de interrupción .

PC

A la pila

GESTIÓN de la INTERRUPCIÓN

La Rutina de Servicio/Atención (IRS: Interrupt Request Service) comienza de la siguiente manera, guardando en la pila el PC, SR y Bloques de Registros:

1. $SP \leftarrow SP - 1$ actualizamos Stack Pointer (hacer sitio en la pila).
2. $M[SP] \leftarrow PC$ se guarda en la pila la dirección de retorno (PC)
3. $SP \leftarrow SP - 1$
4. $M[SP] \leftarrow PSR$ se guarda la pila de estado (acumulador y flags)
5. $EI \leftarrow 0$ se deshabilitan las interrupciones.
6. $INTACK \leftarrow 1$ se acepta la interrupción.
7. $PC \leftarrow IVAD$ se carga en el PC la dirección de la RSI.

SUBRUTINAS

Se denomina subrutina a una porción de código que realiza una operación en base a unos valores dados como parámetros y que puede ser invocado desde cualquier parte del código, incluso desde sí misma. Utiliza los mnemónicos CALL y RET.

Ventajas del uso de Subrutinas:

1. División del problema en tareas más fáciles de escribir y depurar.
2. Evita código redundante.
3. Encapsulamiento del código.

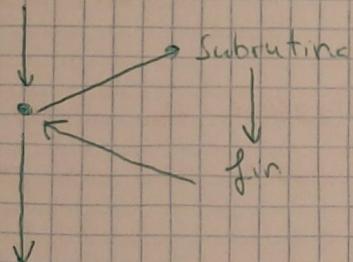
Las tareas se comunican a través de parámetros, por lo que un cambio en una de las tareas no implica cambios en el resto del programa.

Además, permite su reutilización en más de un programa (bibliotecas).

Generalidad

Una misma Subrutina (tarea) puede realizarse con diferentes datos (parámetros) y debe poder ser invocada desde en diferentes momentos.

Proceso Llamador



Gestión de la Subrutina

CALL: en lugar de emplear la dirección absoluta se utilizan etiquetas. El ensamblador se encargará de traducir la etiqueta por la dirección correspondiente.

Dirección de Retorno: al ejecutar un "CALL", se salva en la pila el valor del PC, para conocer la dirección a la que debemos volver una vez finalizada la subrutina.

Paso de parámetros:

- por Registro: eficiente para subroutines con pocos parámetros.
- por Memoria: se necesita una zona de memoria conocida para el programa llamador y para el llamado.
- por Pila: durante la subrutina puede utilizarse la pila, por lo que para no perder información aparece el:

Bloque de Activación + su Puntero.

El Bloque de Activación es la porción de memoria de la pila que contiene el espacio para los resultados, los parámetros, la dirección de retorno y la copia del valor original del registro. Es decir, es una zona reservada de la pila que no puede ser usada por la Subrutina.

P.A. 3.1 - Gestión de SUBRUTINAS

a) ¿Con qué instrucción terminan ambas rutinas?

RET, para volver a la ejecución principal del programa.

b) ¿Cómo se sabe a qué instrucción se debe volver después de la Subrutina? Porque previamente al salto se ha almacenado la dirección a la que apuntaba el PC en la pila.

c) En ambos casos hay que guardar el contenido de los 6 registros en la pila. ¿Pero dónde escribirían las instrucciones del guardado de los registros en ambos casos? Antes de realizar la llamada a la Subrutina (CALL). En el caso de las interrupciones, se realiza al inicio de atenderlos.

d) ¿Estaría bien la siguiente secuencia de instrucciones? ¿Por qué?

900 #ETIQUETA

Guardar A en la Pila.

Guardar flags en la Pila.

Guardar B en la pila.

Recuperar A de la Pila.

Recuperar Flags de la Pila.

Recuperar B de la Pila.

No. Recordemos la estructura de la pila:

LIFO (Last In First Out).

P.A. 3.2 - Gestión de SUBRUTINAS

1- ¿Cuáles son las tres ventajas fundamentales que aporta el uso de Subrutinas?

① División del problema en tareas más fáciles de escribir y depurar.

② Evita código redundante.

③ Encapsulamiento del código. Las tareas se comunican a través del paso de parámetros y resultados, por lo que un cambio en una de las tareas no implica cambios en el resto del programa. Además, permite su reutilización en más de un programa (bibliotecas).

2- ¿Qué es el bloque de activación? La porción de memoria de la pila que contiene el espacio para los resultados, los parámetros, la dirección de retorno y la copia del valor original del registro en el que se guardará el SP (Stack Pointer). Esto se debe a que la Subrutina puede hacer uso también de la pila para almacenar variables locales, parámetros temporales o resultados parciales).

3- Cuando la CPU recibe una interrupción, tiene que ejecutar una Rutina de Servicio a la interrupción, volviendo luego al punto en el que estaba. ¿Qué diferencias hay entre una subrutina y una interrupción hardware? ¿O son iguales? Mientras que en las subrutinas se producen llamadas y saltos, en las interrupciones únicamente se detiene la ejecución normal de programa para ejecutar la ISR (Rutina de Servicio) * dado que es el periférico el que debe identificarse.

No obstante, en ambas (tanto subrutinas como interrupciones) se debe almacenar el PC, la información del bloque de registros y los flags de estado (SR, Status Register), puesto que pueden modificarse durante el otro proceso.

* ISR : Interrupt Service Request.

4- ¿A qué instrucción de pila corresponden los siguientes movimientos?

¿Qué es lo que indican las siguientes dos descripciones?

$RP \leftarrow M[SP]$ el contenido de la dirección de memoria apuntada por el SP se lleva a un registro.
 $SP \leftarrow SP + 1$

Se incrementa el Stack Pointer (SP). (POP).

Se están recuperando las informaciones almacenadas en la pila.

POP A [A, F]

A $\leftarrow M[SP]$

SP $\leftarrow SP + 1$

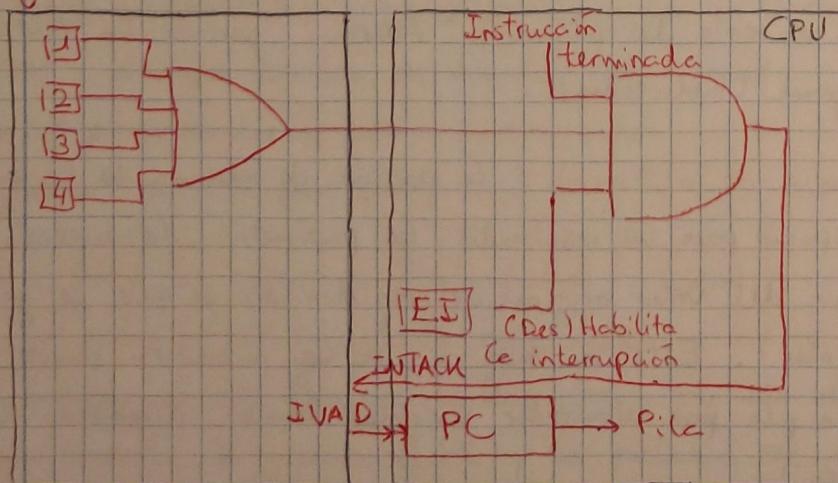
F $\leftarrow M[SP]$

SP $\leftarrow SP + 1$

5- ¿Dónde se salvan las direcciones de retorno de las subrutinas?

En la pila.

6-



Instrucción terminada

CPU

Se trata de una interrupción hardware vectorizada.

INTACK: Reconocimiento de la interrupción

IVAO: Dirección del vector de interrupción.

7 - ¿Qué 2 acciones se realizan cuando se ejecuta un call, (llamada a Subrutina)? Guardar el valor que en ese momento tiene el PC (Program Counter) y después cargar en el PC la dirección de la Subrutina. Dicha dirección viene indirectamente especificada mediante la etiqueta (TAG).

8 - Cuando se produce una interrupción, ¿Cómo sabe la CPU qué dispositivo a interrumpido? Hay 3 opciones:

① Multinivel : cada dispositivo tiene una entrada asociada. Sencillo de gestionar pero coste muy elevado.

② Línea única + polling : una línea única indica a la CPU que algún dispositivo requiere atención, y esta realizar un sondeo.

③ Vectorizados: además de activar la señal de solicitud de interrupción, el dispositivo que interrumpe pone en el Bus de Datos un vector que lo identifique.

9 - ¿Qué 3 tipos de interrupciones existen? ¿En qué se basa cada una de ellas? Poner un ejemplo de cada uno de los tipos de interrupción.

- Interrupción HARDWARE : un recurso hardware requiere atención. Es asincrónica, puede darse en cualquier momento. Suelen ocurrir durante la ejecución de una instrucción, que primero se termine y después se atienda (ISR).

Ejemplo: el mouse hace click.

- Interrupción SOFTWARE : no son propiamente instrucciones ya que son introducidas por el programador. Son instrucciones y se conoce cuándo sucedrán. Son sincrónas.

Ejemplo: el usuario solicita un recurso del núcleo mediante una llamada al sistema (open, write, read, ...).

- Traps (TRAMPAS) : generadas por la CPU ante una condición de error. Aunque no sea de manera intencionada, están introducidas por el programador y están sincronizadas.

Ejemplo: división por cero.

10 - Ante una interrupción (hardware), ¿qué ha de guardarse, además de la información que se guarda ante una subrutina? ¿Dónde?

La información del Bloque de Registros, no solo el PC. De forma que, en general, se guarde el PC y la palabra de estado (valor del acumulador y flags de estado -SR, Status Register-).

La manera más correcta de guardarla es en la pila.

11 - ¿Cómo se denomina el que una subrutina puede ser invocada dentro de otra subrutina? Anidamiento de subroutines. (Subroutine nesting)

- 12 - (1) $SP \leftarrow SP - 1$ Posición anterior en la pila al Stack Pointer.
- (2) $M[SP] \leftarrow PC$ Se almacena el valor del PC en la pila.
- (3) $SP \leftarrow SP - 1$
- (4) $M[SP] \leftarrow PSR$ Se guarda la palabra de estado en la pila → flags de estado
- (5) $EI \leftarrow 0$ Se deshabilitan las interrupciones.
- (6) $INTACK \leftarrow 1$ Se acepta la interrupción.
- (7) $PC \leftarrow IVAD$ Se carga en el PC la dirección de la interrupción.

13 - ¿Cuándo tienen sentido este código? Cuando se procede a atender a una interrupción. De hecho, son los pasos a seguir en la gestión de una interrupción. Se da al principio de la Rutina de Servicio (ISR).

14 - ¿Antes de acceder a una subrutina o a dar servicio a una interrupción se guarda la información relevante en los registros internos de la CPU? ¿Por qué? ¿Se guarda la misma información en cada caso?

En el caso de las subrutinas, la información relevante se guarda antes de acceder a ellas, antes de realizar el salto. La razón es el salto que se produce, se altera el PC y no es posible guardar la información desde dentro de la subrutina.

En cuanto a las interrupciones, el primer paso de la Rutina de Servicio (ISR) es almacenar la información relevante. Se puede hacer de esta manera puesto que en las interrupciones no se produce ningún salto, únicamente se detiene la ejecución normal del programa para atender la interrupción.

En las subrutinas resulta imprescindible guardar la dirección de retorno (PC), mientras que en las interrupciones será necesario almacenar la dirección de retorno (PC), la información del bloque de registros y los flags de estado (es decir, la palabra de control).

BUSES

Los buses son caminos eléctricos que unen dispositivos.

Representan el movimiento de información en el computador, puesto que la información generada por la CPU del procesador, almacenada y/o generada en un dispositivo es transportada a través de estos.

3 tipos de buses:

- Internos del procesador : unión de CPU con ALU y registros.
- Comunicación CPU con Memoria. (Bus de alta velocidad)
- Destinados a E/S . (Bus de expansión)

3 tipos de información ! (para buses de comunicación y E/S)

- Bus de Datos
- Bus de Direcciones
- Bus de control.

Normalmente, estos buses discurren a través de la "Placa Madre" (PCB).

* 1 byte puede transmitirse por 8 líneas de bus.

CARACTERÍSTICAS :

Ancho del bus: número de señales eléctricas que transportan información (cuántos pistos de cobre).

50-100 líneas divididas en tres grupos funcionales:

Líneas de Datos: bus de Datos ; anchura de 8-64 .

Líneas de Direcciones: bus de Direcciones ; anchura de 8-64 .

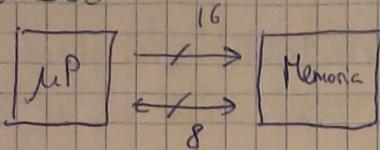
Determina la máxima capacidad de memoria en el sistema. También dirige puertos de E/S El direccionamiento es 2^8 .

Líneas de Control: dado que los buses pueden ser compartidos por datos e instrucciones (tipo multiplexado), será necesario una señal de control que nos indique qué se está transmitiendo. No presentan problemas respecto a su tamaño.

Cuanto más ancho sea el bus, más cara será la placa.

Tipos de buses:

Dedicado:



un bus de datos y otro
bus de direcciones. (7 de control)

Multiplexado: mismo bus para datos y direcciones. Una señal (línea) de control será necesaria para indicar qué se está transferiendo.

Temporización:

Bus Síncrono: el intercambio de información está regido por un reloj (CLK). Es fácil de gestionar pero los periféricos lentos no consiguen seguir al reloj. En estos casos, se añade una señal /WAIT, siendo el bus Semi-Síncrono. Además, se necesitan 3 señales de control:

- /MREQ (petición de memoria)
- /RD ("1" escritura - WR)
- /WAIT

Bus Asíncrono: los dispositivos irán a la velocidad que puedan ir.

Se necesitan las siguientes señales de control:

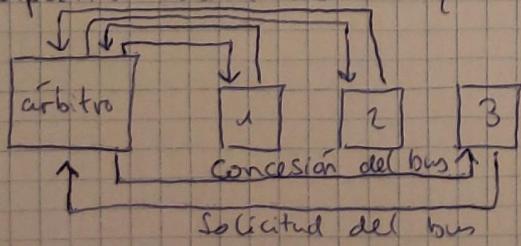
- /MREQ (petición de memoria)
- /RD ("1" escritura - WR)
- /MSYN Master Synchronization
- /SSYN Slave Synchronization

Utiliza el protocolo hardware "Handshake":

- Activación de /MSYN (por parte del dispositivo que controla el bus -MASTER)
- Reconocimiento de /MSYN (SLAVE) y activación de /SSYN (SLAVE)
- Desactivación de /MSYN (MASTER) al reconocer a /SSYN
- Desactivación de /SSYN (SLAVE) al reconocer la desactivación de /MSYN.

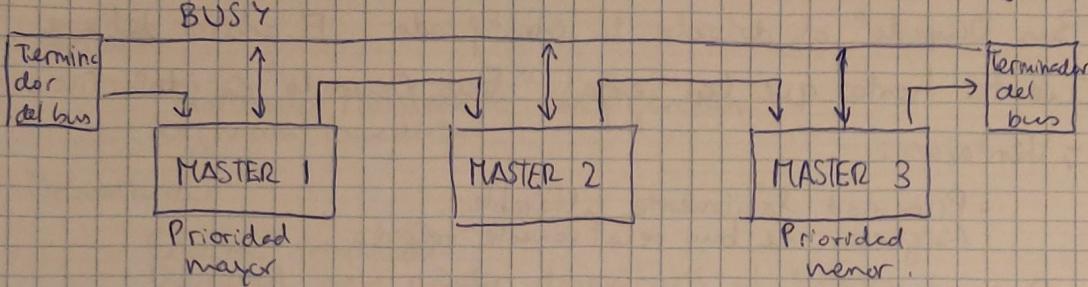
Arbitraje del bus: cuando dos o más dispositivos quieren acceder al bus al mismo tiempo se presenta un conflicto de colisión. Esto se resuelve introduciendo un árbito del bus.

Centralizado: controlador de bus /árbitro integrado en CPU o ser un dispositivo diferenciado. Es quien controla el bus. No conoce qué dispositivo solicita el bus y está conectado con todos.



Para evitar retrasos en la actividad de la CPU con la memoria, se recurre a disponer la memoria en un Bus distinto al dedicado a E/S.

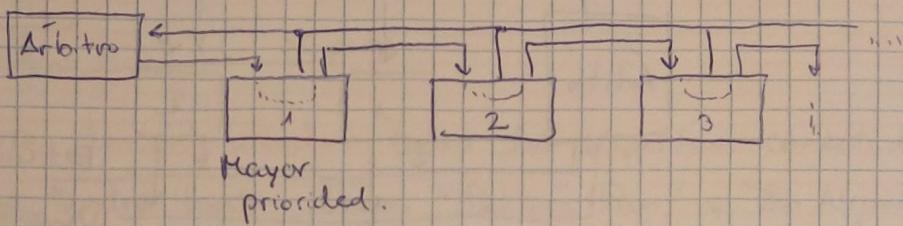
Descentralizado / Distribuidos: los dispositivos se organizan entre ellos para utilizar el dispositivo. Todos ver el estado del bus, por lo que conocer si están siendo servidos o serán el siguiente.



Estrategia de Arbitraje:

Daisy - Chain (Serie):

Centralizado:



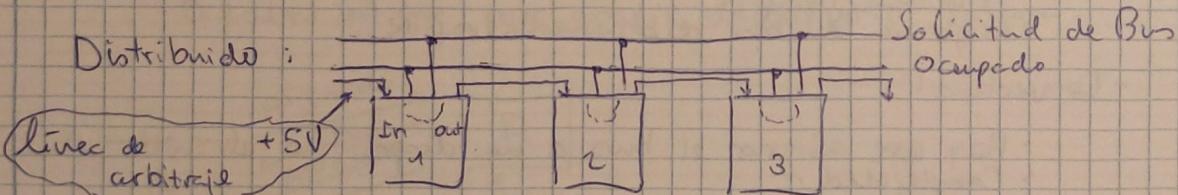
- Ventajas:

- Simplicidad
- Facilidad para añadir dispositivos.

- Desventajas:

- Propenso a fallos
- Prioridades fijas
- Lentitud de funcionamiento.

Distribuido:



- Ventajas: Simplicidad

- Desventajas:

- Propenso a fallos (muy sensible al ruido eléctrico)
- Prioridades fijas
- Lentitud de funcionamiento.

Centralizado a dos niveles: se agrupan los dispositivos de prioridad similar.

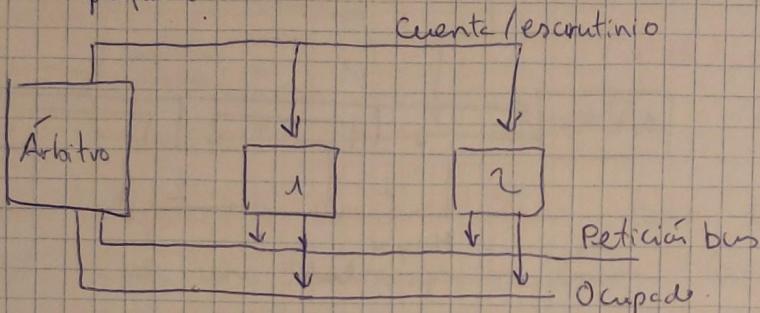
Encuesta centralizada: Cada dispositivo posee un número de identificación. Cuando se activa la línea "Petición Bus" el árbitro inicia una cuenta. El dispositivo que desea usar el bus activa "Bus Ocupado" al detectar su identificador. El árbitro detiene la cuenta hasta que la señal "Bus Ocupado" se desactive.

- Ventajas:

- Prioridad fácilmente alterable.
- Asignación de bus relativamente rápida.

- Desventajas:

- Complejidad.



Encuesta distribuida: el dispositivo cuyo código aparece en el escrutinio puede optar al bus activando "Bus aceptado". Se le cederá el control y el arbitraje del bus.

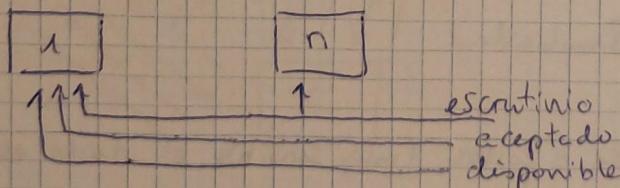
Cuando termina, inicia la cuenta y active "Bus disponible".

- Ventajas:

- Prioridad no fija.
- Asignación del bus relativamente rápida.

- Desventajas:

- Hay que asignar el bus a un dispositivo al iniciar el sistema.
- Complejidad.



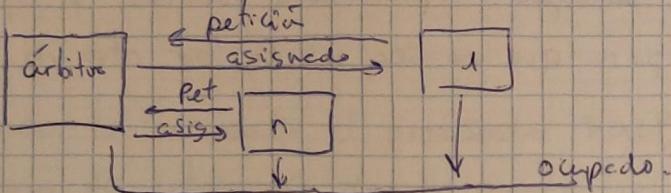
Petición independiente centralizado: Cada dispositivo posee una línea "petición bus" que va al árbitro y una línea "Bus asignado" procedente del árbitro. El árbitro concede el bus.

- Ventajas:

- Prioridades totalmente configurables
- Asignación inmediata.

- Desventajas:

- Enorme complejidad
- Dificultad para añadir dispositivos al sistema.



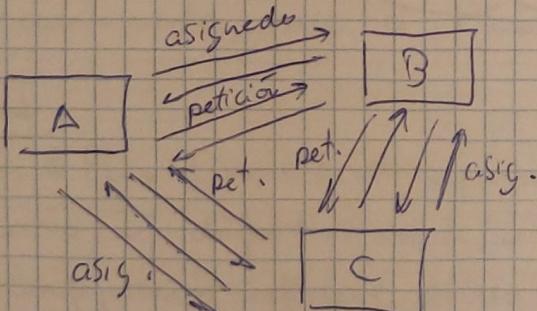
Petición independiente descentralizado: todos los dispositivos conectados entre sí. Cada uno tendrá tantas líneas "Petición Bus" y "Bus Asignado" como dispositivos haya. Cada uno puede enviar "Bus Asignado" al que deseé ceder el control del bus.

- Ventajas:

- Prioridades totalmente configurables
- Asignación inmediata

- Desventajas:

- Enorme complejidad
- Dificultad extrema para añadir dispositivos al sistema.



Transferencia de datos:

Por Bloques: por ejemplo, la memoria caché, cuando hay que traer de la MP a la MC una palabra, es necesario traer la línea (bloque). Será más eficiente traer palabras a palabra, dado que serán accesos consecutivos a la MP.

Acceso por Semaforización: (escribiendo y leyendo): arquitectura multiprocesador, cuando un procesador quiera acceder al área de memoria, leerá primero el bit "semáforo" (previamente escrito) y si está "verde", lo pondrá a "rojo" para que no sea usado por otro de los procesadores.

Escritura y Lectura:

Escritura multiplexada:

Dirección	Dato
-----------	------

Tiene que indicar qué se manda.

Lectura multiplexada:

Dirección	-	Dato
-----------	---	------

Manda la dirección y tiene que esperar a que el periférico mande el dato.

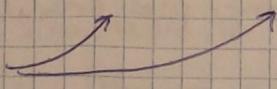
Escritura no multiplexada:

Dirección

Se mandan al mismo tiempo datos e instrucciones.

-	Dato
---	------

Lectura no multiplexada:



ISA

- Basado en el procesador 8088 de Intel. 62 conectores.
- 20 líneas para el Direccionamiento. ($A_0 - A_{19}$)
- 8 líneas para los Datos ($D_0 - D_7$)
- 5 líneas para lectura de memoria (MEMR), Escritura (MEMW), lectura E/S (IOR), escritura E/S (IOW) y registro de dirección (ALE)
- 5 líneas de solicitud de Interrupciones (IRQ3 - IRQ7)
- ...

Buses dedicados.

Cuando se introdujo el procesador 80286, que tenía 16 bits para el bus de datos, se decide ampliar el bus ISA incorporando un nuevo conector. De esta manera, se mantenían las conexiones anteriores y las tarjetas existentes seguían siendo funcionales.

- 24 líneas para el Direccionamiento ($A_0 - A_{23}$)
- 16 líneas para los Datos ($D_0 - D_{15}$)

Inconvenientes:

- Velocidad insuficiente para aplicaciones modernas:
 - ISA 8 bits (4 MHz) \rightarrow 4 MB/s
 - ISA 16 bits (8 MHz) \rightarrow 16 MB/s
- No es Plug & Play, sino mediante switches.

Consecuentemente, nace el PCI.

PCI

Plug & Play (característica pionera) para conectar dispositivos de E/S.
Máximo 32 dispositivos de E/S.

La configuración software asigna direcciones y prioridades.

No es lo suficientemente bueno para ser bus de memoria.

No es compatible con las tarjetas para ISA.

Solución: usar 3 o más buses (Surgen los puentes)

- Northbridge
- Southbridge

Características:

- Voltaje 3.3V ó 5V.
- Capacidad 32 ó 64 bits.
- Temporización 33 ó 66 MHz
- Arbitraje centralizado.

Transferencias por ráfagas

- Transacción: operación de transferencia completa
- Fases: transferencias individuales.

Inversión del bus: dado que el bus está multiplexado, si se está transmitiendo dirección y se quiere transmitir dato, hay un periodo de inversión del bus.

* Página 186 Diapositivas: ¿Por qué en el segundo tarda dos ciclos de reloj más? Porque hay un momento en el que el Initiator (origen; IRDY #) no está preparado, y otro momento en el que no está preparado el Target (destino; TRDY #).

Arbitraje (Bus Mastering): centralizado

• Solicitar bus:

El periférico activa # REQ.

Espera hasta que el árbitro habilite su # GNT

El dispositivo puede utilizar el bus en el siguiente ciclo.

Una concesión es válida para 1 transacción. Si un dispositivo quiere ejecutar una segunda y ningún otro dispositivo está solicitando el bus, puede continuar dejando un ciclo ocioso entre transacciones.

SCSI

Buses dedicados (8-16 bits)

Son posibles dos niveles de tensión (3.3 V o 5 V)

Arbitraje distribuido de prioridad fija (el controlador #7 es el de máxima prioridad)

Velocidad de transferencia: en función de la longitud y del número de dispositivos conectados (5 MB/s a 640 MB/s)

Habilidad para solapar peticiones → bus de altas prestaciones.

BSY → señal de ready (RDY)

Los dispositivos conectados al SCSI no son parte del espacio de direcciones del procesador.

Arbitraje y Selección:

Distribuido de prioridad fija. El controlador que haya ganado el arbitraje activa /SEL y la linea del dispositivo seleccionado.

Transferencia de información:

Se transfieren órdenes, respuestas o datos. Se utilizan señales de diálogo de conformidad para controlar la transferencia (/REQ, /ACK).

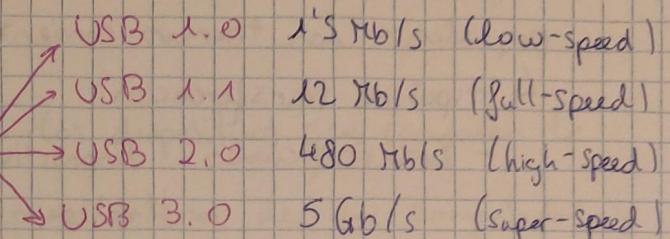
USB

Mecanismo simple y de bajo coste para conectar una amplia variedad de dispositivos. No se tendrá que actuar sobre microinterruptores en el dispositivo, por lo que será fácil de utilizar.

Además, no hay que abrir el computador para incorporar nuevos dispositivos. Los dispositivos de entrada/salida serán alimentados desde el cable del bus y podrán conectarse hasta 127 dispositivos por computador. Conexión "plug & play" y trabaja con dispositivos en tiempo real (phone)

Características :

- Bus SERIE
- Velocidad (tráfico dividido)



• Inicialización :

- Cable principal donde se conectan los dispositivos
 - Al conectar se interrumpe al S.O.
 - En un primer ciclo, se detecta el tipo de dispositivo y el ancho de banda permitido.
- * Si el ancho de banda es separado por el S.O., le asigna una dirección de 7 bits (1-127, inicialmente 0)
- El reloj se transmite implícito en los datos.

Topología :

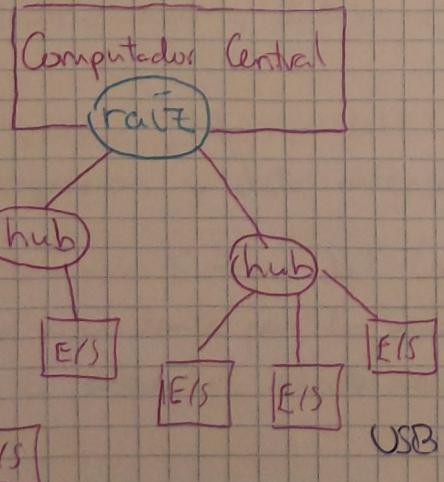
Estructura en forma de árbol que permite añadir o quitar dispositivos.

Cada nodo tiene un concentrador (hub) que actúa como punto intermedio entre el computador y los dispositivos de E/S.

Cada concentrador tiene un número de puertos en los que se pueden conectar los dispositivos u otros "hubs".

Los dispositivos de E/S serían las hojas.

Un concentrador raíz conecta el árbol entero al computador.

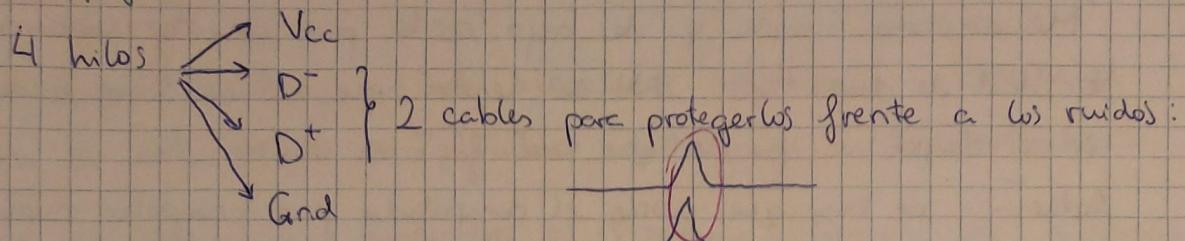


El concentrador copia los mensajes que recibe del computador los transmite a TODOS los E/S; sólo el direccinado responde.

Un mensaje de E/S se envía sólo hacia la raíz, no lo ven los demás, y la raíz lo copia en otro E/S. No pueden comunicarse entre ellos.

USB opera mediante polling (Sondeo).

Configuración Hardware:



La diferencia está protegida frente a los ruidos.

Las transferencias se denominan tramas y estas están formadas por paquetes. Las tramas comienzan por SOF: Start Of Frame.

4 tipos de transferencias:

- **Interrupciones**: dispositivos más lentos, información con poca frecuencia (ratones, teclados...)
- **Por bloques**: dispositivos que mueven grandes cantidades de información en cada transferencia (impresoras...)
- **Isócrona**: flujo de datos constante y en tiempo real, sin aplicar detección ni corrección de errores (altavoces USB...)
- **Control**: configurar dispositivos, emisión de órdenes y revisión del estado.

MEMORIAS

Son dispositivos capaces de almacenar información. Desde el punto de vista de un computador, distinguiremos 3 apartados:

1. La memoria que contendrá el Programa (Conjunto de instrucciones) y los datos de trabajo.
2. Elemento sencillo con gran diversidad de tipos, tecnologías, prestaciones ...
3. Se da una jerarquía entre los dispositivos que componen la memoria. Unas se localizan en el interior de un computador y otras en el exterior.

Un dispositivo de memoria debe presentar una determinada estructura:

- Medio: Soporte físico y composición
 - óptico
 - magnético
 - eléctrico
- Transductor: convierte una magnitud física en otra.
- Mecanismo de direccionamiento: permite accede al dato / lugar deseado para procedimientos de Lectura / Escritura de información.

CARACTERÍSTICAS

LOCALIZACIÓN:

- Memoria Interna del Procesador (CPU): alta velocidad, tamaño pequeño. Memoria de trabajo para almacenamiento temporal de instrucciones y datos.
 - Registros
 - Memoria Caché.
- Memoria Interna del Computador: memoria principal, residen los programas y datos obtenidos.
- Memoria Externa del Computador: memoria secundaria o auxiliar, la más lenta pero la más grande. Limitada a la velocidad del Bus y a aspectos electromecánicos.
 - CD
 - DVD

TAMAÑOS de MEMORIAS:

1 bit

-

1 Kb 1024 bits = 2^{10} bits

1 nibble

4 bits

1 Kb 1024 Kb = 2^{20} bits

1 byte

8 bits

1 Gb 1024 Mb = 2^{30} bits

1 Tb 1024 Gb = 2^{40} bits

1 word 16-32 bits

1 long word 32-64 bits.

CAPACIDAD:

Es la cantidad de información que puede almacenar el dispositivo.

La práctica habitual es medirla en múltiplos de byte (octetos).

1B un byte

1 Kb 1K bit (1024 bits, 2^{10})

1 KB es una memoria de extensión 1K (posiciones) de 8 bit de tamaño cada posición.

$$* 1 \text{ Kb} = 1 \cdot 2^{10} \text{ bits} = 1024 \text{ bits}$$

$$1 \text{ KB} = 1 \text{ K} \times 8 = 1 \cdot 2^{10} \times 8 = 2^0 \cdot 2^{10} \times 8 = 2^{10} \times 8 = 1024 \times 8 \text{ grupos de}$$

$$2 \text{ KB} = 2 \text{ K} \times 8 = 2 \cdot 2^{10} \times 8 = 2^1 \cdot 2^{10} \times 8 = 2^{\text{11}} \times 8 = 2048 \times 8 \text{ bit cada una de los pa-$$

MÉTODO de ACCESO:

• **Acceso Secuencial (SAM)**: para acceder a un dato, hay que recorrer todos los anteriores. Método LENTO, útil en grandes sistemas de almacenamiento. P.E.: cintas y buffers de memoria (datos están ordenados)

• **Acceso Aleatorio (RAM)**: se accede a un dato de manera directa a través de su dirección. El tiempo de acceso es siempre el mismo.

• **Acceso Asociativo (CAM)**: se accede a un dato por su contenido, se busca en toda la memoria simultáneamente. Una vez encontrado, se da la dirección donde se ubica. Muy rápido que la RAM pero más cara.

* SAM: Sequential Access Memory

RAM: Random Access Memory

CAM: Content Accessible Memory

VELOCIDAD:

3 parámetros para medir el rendimiento de una memoria.

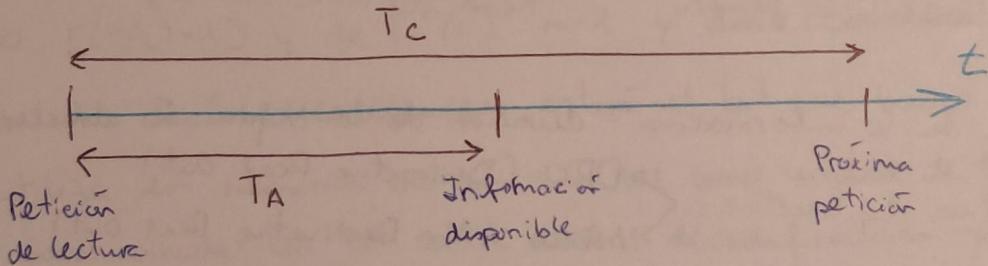
• Tiempo de Acceso (T_A):

→ RAM: tiempo que transcurre desde el instante en el que se presenta una dirección a la memoria hasta que el dato es memorizado o disponible para su lectura.

→ CAM / SAM: tiempo empleado en activar el mecanismo de L/E en la posición deseada, esto es, tiempo empleado en localizar la dirección.

• Tiempo de Ciclo de Memoria (T_c):

tiempo transcurrido desde la orden de operación de L/E hasta que se puede dar la siguiente orden de L/E. Por debajo de este tiempo la memoria no responde. Interesa que sea lo menor posible.



• Velocidad de Transferencia (V_T):

la velocidad a la que se pueden transferir datos.

$$\rightarrow \text{RAM: } V_T = 1/T_c$$

$$\rightarrow \text{CAM / SAM: } T_n = T_A + (N/V_T)$$

* T_n : tiempo medio de L/E de N bits. Tiempo de disponibilidad de datos.

N: número de bits a transferir

V_T : velocidad de transferencia (bits/s) una vez encontrado el dato.

T_A : tiempo de acceso al dato. ↗ SAM: variable
CAM: fijo.

DISPOSITIVO FÍSICO:

En un principio se utilizaba la "memoria de ferrite". Era de lectura destructiva, después de leer había que sobreescribir. Sin ello, habrían sido muy rápidas.

Memoria principal: dispositivos semiconductores.

Memoria secundaria:

Memoria magnéticas: cintas, discos, ...

Memoria ópticas: CD ROM

Memorias magneto-ópticas:

ASPECTOS FÍSICOS: características físicas.

Alterabilidad: posibilidad de alterar el contenido de una memoria.

Los hay de solo lectura o de lectura/escritura.

(ROM)

Read Only Memory

(RWM)

Read Writable Memory

Permanencia de la información: duración de la información almacenada en memoria.

→ Lectura destructiva: ↗ DRO (Destructive Read Out)
↗ NDRO (Non Destructive Read Out)

→ Volatilidad: posibilidad de destrucción de la información almacenada cuando se produce un corte en el suministro eléctrico.

Memorias VOLÁTILES y NO VOLÁTILES.

→ Almacenamiento Estático / Dinámico:

Estática: la información que contiene no varía con el tiempo

(SRAM: Static Random Access Memory). Muy rápidas

Dinámica: la información se va perdiendo conforme transcurre el tiempo. Se debe "refrescar".

(DRAM: Dynamic Random Access Memory). Más lentas.

ORGANIZACIÓN: disposición física de los bits para formar palabras.

Depende del tipo de memoria. Para una memoria Semiconductora distinguimos tres tipos de organización:

• Organización 2D: ram de 2^m palabras de ancho " n " bits cada una. La matriz de celdas está formada por 2^m filas (n^o de posiciones de la memoria interna que van de la 0 a la 2^{m-1} en el bus de direcciones) y n columnas (n^o bits en el registro de la memoria o ancho de palabra).

Organización lineal que se emplea en memorias de poca capacidad y gran rapidez de acceso.

• Organización 2 1/2 D: utiliza dos decodificadores con $m/2$ entradas y $2^{m/2}$ salidas. El bus de dir. de la organización 2D se divide en dos buses encaminados a dos decodificadores que van de $0..[^{(m/2)-1}]$ y de $[^{m/2}..m-1]$ y donde coincidan las líneas de salida de los decodificadores estará el bit que busco.

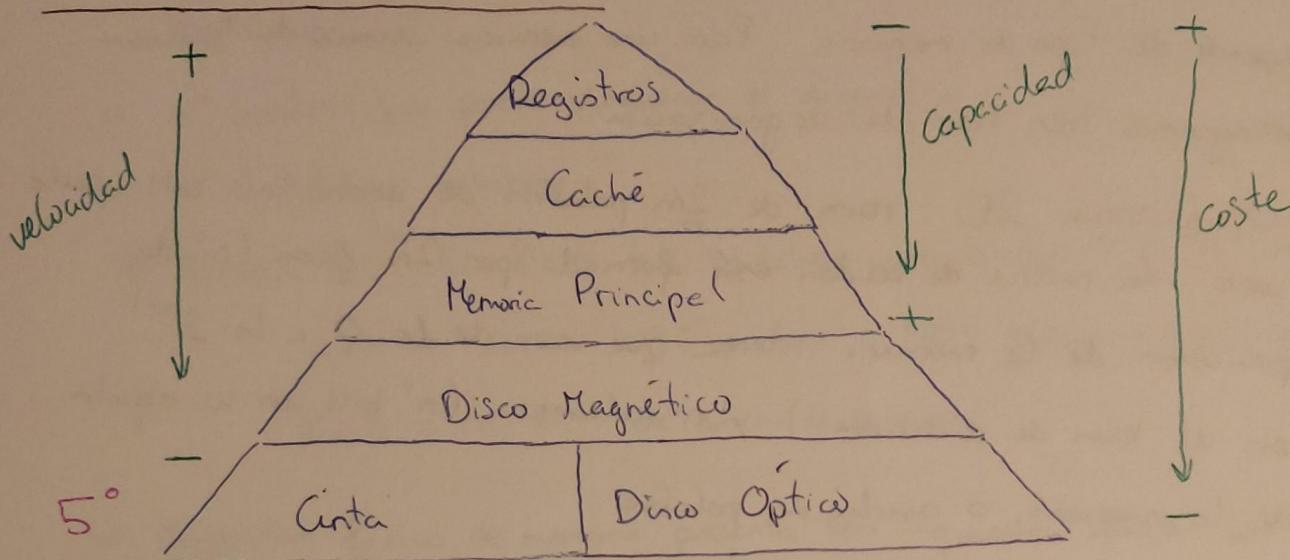
Se utiliza en memorias de un solo bit como unidad de transferencia pues solo coincide un bit entre ambos decodificadores y precisa de menor tamaño para la memoria que la arg. 2D \rightarrow menor n^o de puertos lógicos.

• Organización 3D: similar a la 2 1/2 D, pero la palabra de n bits se almacena en n planos y dentro de cada plano se selecciona la posición " x " y la posición " y ", por lo que solo se habilitarán los bits que estén en el mismo plano.

Diseño más complicado y para memorias de acceso lento.

Son chips de mayor densidad de silicio y, por tanto, más anchos para poder ampliar el ancho de palabra fácilmente.

JERARQUÍA de MEMORIAS:



5º nivel : externos al computador.

MEMORIA PRINCIPAL SEMICONDUCTORA:

TIPOS :

① Solo lectura :

Escritura destructiva : ROM (Read Only Memory), PROM (Programmable ROM)

Escritura no destructiva : EEPROM (Erasable Programmable ROM)

② L/E :

Lectura principalmente : EEPROM (Electrically EEPROM), FLASH

L/E (RAM) : DRAM (Dynamic RAM), SRAM (Static RAM)

SODRAM
(Synchronous DRAM)

DDR
(Double Data Rate DRAM)

QDR (Quad Data Rate)
ZBT (Zero Bus Turnaround)

RAM	Clase L/E	Borrado Eléctrico por bytes	Escritura Eléctricamente	Volatilidad Volatile
ROM	↳ Solo lectura	NO	Máscara	
PROM				No volatile
EPROM	L/E Fundamentalmente Lectura	Luz ultravioleta	Eléctricamente	
FLASH		Eléctricamente por bloques		
EEPROM		Eléctricamente por bytes y bloques		

DISEÑO : organizado internamente como una matriz de celdas de memoria $n \times m$.

n : n° palabras que puede almacenar el dispositivo

m : n° de bits por palabra.

* Memoria de $4K \times 8 \rightarrow 4K$ palabras con ancho de 8-bit cada una.

$$4K = 4 \cdot 1024 = 4096 \text{ palabras} = 4096 \text{ direcciones o posiciones de memoria}$$

8 bits = m bits de ancho de palabra.

Un dispositivo físico de memoria presenta los siguientes señales en sus patillas:

n patillas : bus Direcciones , se direccionan 2^n palabras .

m patillas : bus Datos , ancho de palabra de m bits .

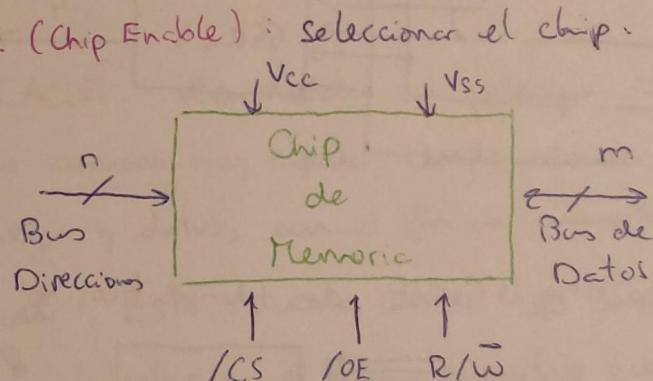
Read /Write : operación de L/o E . También

WE (Write enable)
OE (Output Enable)

/CS (Chip select) ó /CE (Chip Enable) : seleccionar el chip.

Vcc : 5V.

Vss : Tiene (0 V) .



MAPA de MEMORIA :

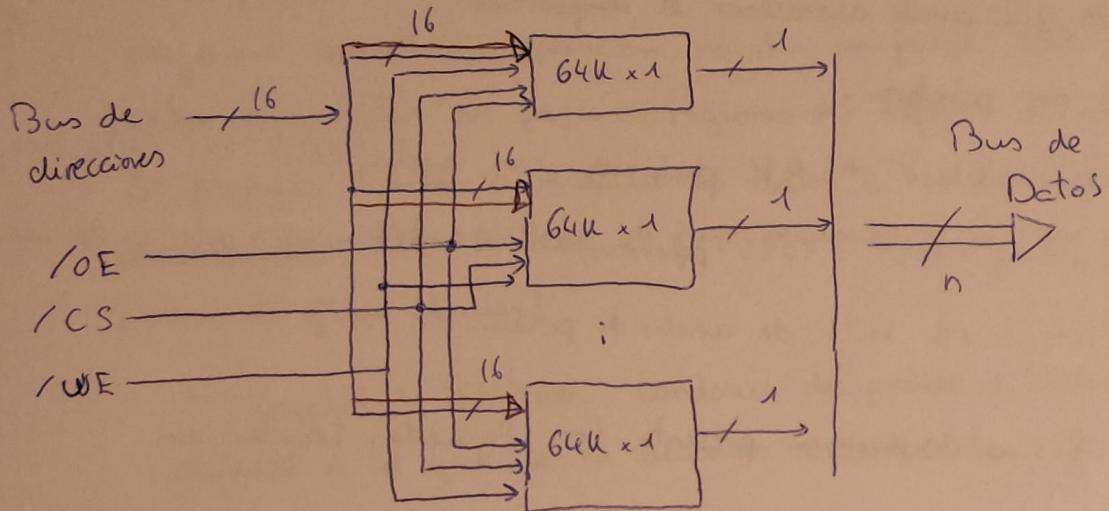
Memoria de $64K \times 8 = 64KB$

- N° líneas de dirección : $64K = 64 \cdot 2^{10} = 2^6 \cdot 2^{10} = 2^{16}$ posiciones de memoria
16 líneas (bits) de dirección .

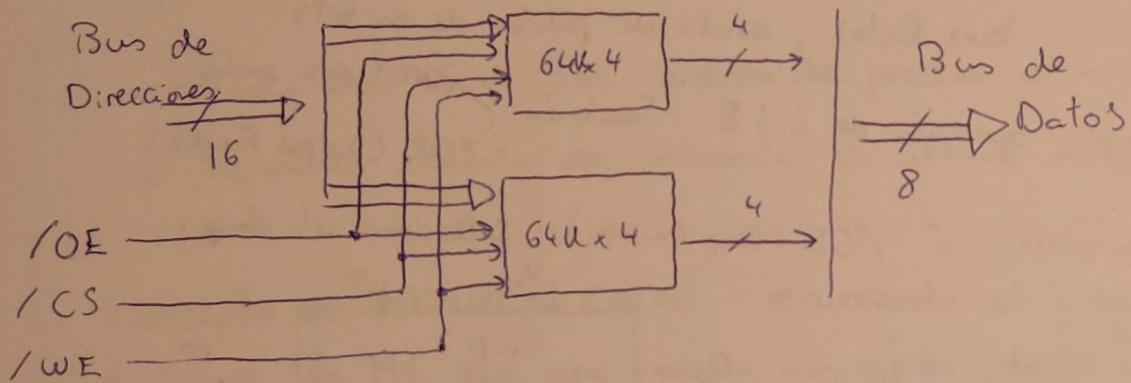
- Datos accedidos : $2^8 = 2^{3 bits} = 256$ datos diferentes accedidos .

AMPLIACIONES de MEMORIA:

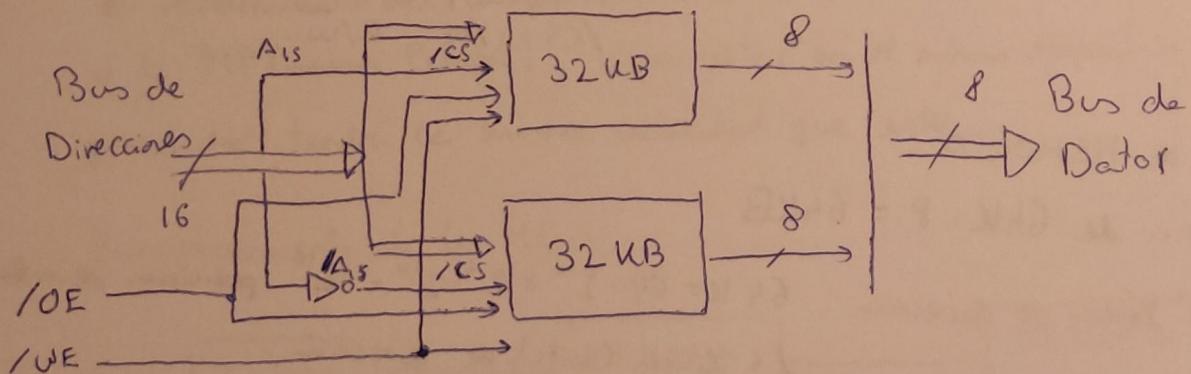
- Memoria de n bits con dispositivos de $64K \times 1$:



- Memoria de 8 bits con dispositivos de $64K \times 4$:



- Memoria de 64KB con dispositivos de 32KB:



MEMORIA CACHÉ

Principio de LOCALIDAD: Las direcciones que genera un programa durante su ejecución no son uniformes, sino que tienden a estar concentradas en pequeñas regiones del espacio de direcciones.

Los programas tienden a reutilizar los datos e instrucciones que han utilizado recientemente. Esto se debe al flujo secuencial y a las estructuras de control de flujo (bucles).

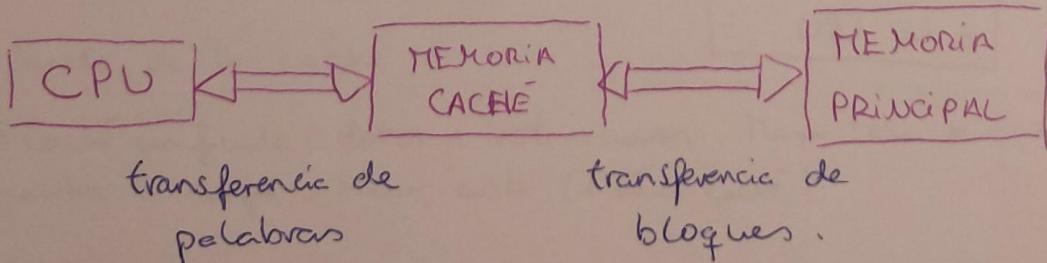
→ Localidad TEMPORAL: tendencia del proceso a hacer referencia a elementos a los que se ha accedido recientemente.

→ Localidad ESPACIAL: tendencia del proceso a efectuar referencias en la vecindad de la última referencia que ha realizado.

→ Localidad SECUENCIAL: tendencia del proceso a hacer referencia al siguiente elemento en secuencia al último referenciado.

Concepto de MEMORIA CACHÉ: recurriendo al principio de localidad, la MC será una memoria muy rápida donde almacenar bloques secuenciales de código y datos, con el fin de aumentar el rendimiento del computador. Está situada entre el procesador y la MP (memoria Principal), y al estar en el mismo dispositivo que el procesador tendrá la misma velocidad que este.

Funcionamiento de la MC:



MP: 2^n palabras, M bloques (K palabras/bloque); n : n° bits bus Direcciones.
 $M = \frac{2^n}{K}$ número de bloques

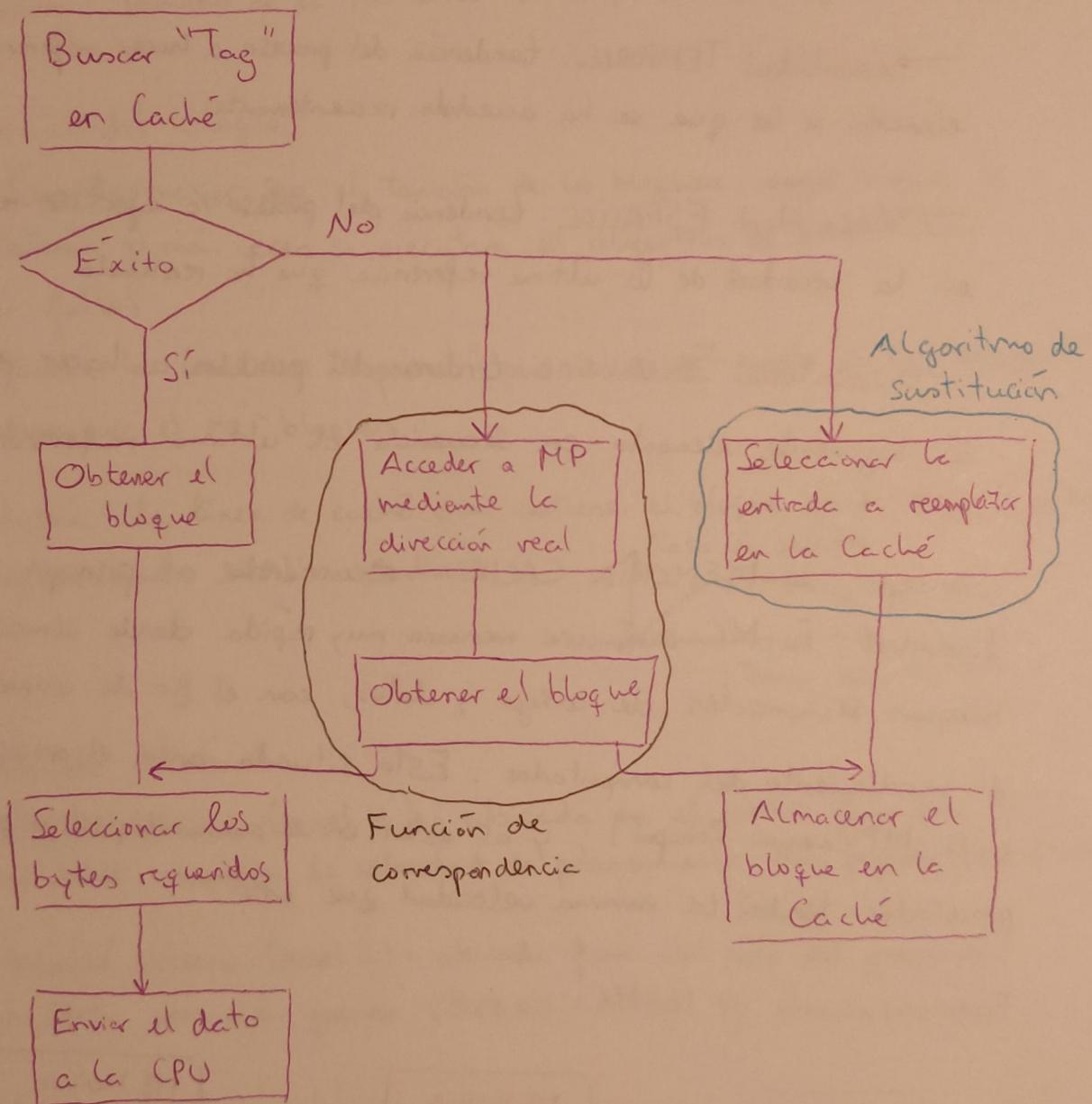
MC: una línea de cache contiene un bloque (K palabras consecutivas)

"Tag" representa la dirección del bloque en la MP.

Tasa de acierto: cuando la CPU necesita una palabra de memoria y la encuentra en MC, se produce un acierto ("hit"), sino fallo ("miss").

$$\text{tasa-de-aciertos} = \frac{n^{\circ} \text{ de aciertos}}{n^{\circ} \text{ de aciertos} + n^{\circ} \text{ de fallos}}$$

ORGANIGRAMA de FUNCIONAMIENTO:



PARÁMETROS de DISEÑO :

• Tamaño de la MC:

Suficientemente pequeña para que el coste medio por bit esté próximo al de la MP.

Suficientemente grande para que el tiempo de acceso total (MP + MC) este lo más próximo posible al de la caché.

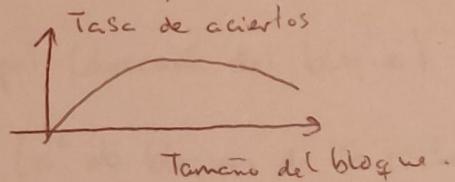
Conclusión: tamaño de la MC entre 1KB y 1MB.

• Tamaño del Bloque:

Cuanto mayor sea el tamaño de los bloques, menos bloques se instalarán y más veces se ejecutará el algoritmo de sustitución (más falso).

Se favorece la localidad "espacial" pero va en detrimento de la "temporal" al disminuir el nº bloques.

Conclusión: la línea de caché que contiene el bloque es de entre 4 y 64 bytes (excepcionalmente 128) consecutivos.



• Número de cachés:

→ Caché interna (nivel 1): ubicada en el mismo chip que el procesador, aumenta la velocidad de procesamiento. Muy pequeña (16-64 KB).

→ Caché externa (nivel 2): ubicada fuera del chip del procesador. Más lenta pero más grande (512 KB - 1 MB)

• Contenido de la caché:

→ Caché unificada: datos e instrucciones. Mayor tasa de aciertos, porque equilibra la carga. Menor coste (1 única caché)

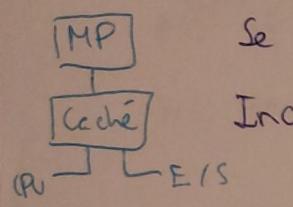
→ Caché dividida: 1 caché para datos y otro para instrucciones.

Elimina la competición entre el procesador de instrucciones y la unidad de ejecución. Duplica el ancho de banda del sistema de memoria.

• Escritura de Datos - Política de Actualización :

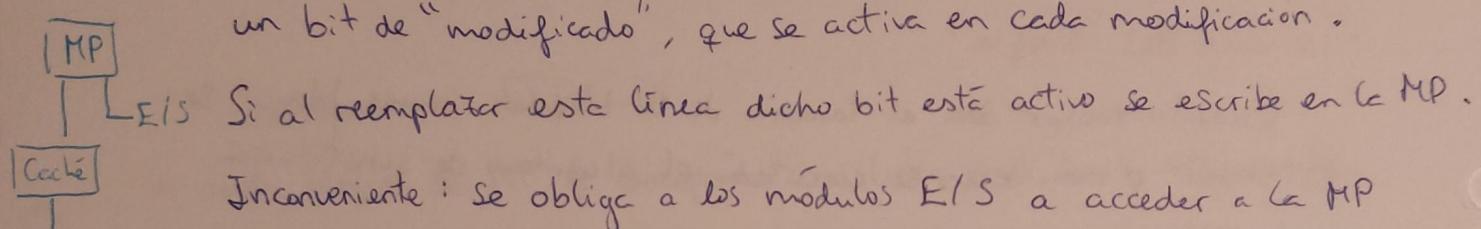
Un dato que está en Caché y ha sido modificado, hay que actualizarlo en la MP.

→ Escritura Inmediata o Directa (Write Through) : todas las escrituras se hacen en MC y MP a la vez.



Inconveniente : Se genera mucho tráfico entre ambas memorias.

→ Post-Escritura (Write Back) : cada línea de caché tiene un bit de "modificado", que se activa en cada modificación.



Inconveniente : Se obliga a los módulos E/S a acceder a la MP a través de la caché.

Ventaja : existe menos tráfico entre MP y MC.

Esta arquitectura es la más común.

CORRESPONDENCIA - Organización de la Caché:

Existen menos líneas que bloques, por lo que es necesario un algoritmo (correspondencia)

$$MP \rightarrow 1KB = 2^{10} \times 8 ; \text{ tamaño del bloque } 8B = 2^3$$

$$MC \rightarrow 32B = 2^5$$



$$MP \rightarrow 2^{10} / 2^3 = 256 \text{ bloques (8 bits de direccionamiento)}$$

$$MC \rightarrow 2^5 / 2^3 = 8 \text{ líneas (3 bits direccionamiento)}$$

* Correspondencia DIRECTA :

A cada bloque de MP le corresponde una línea. A varios bloques les corresponde la misma línea.

Etiqueta ("Tag")	Línea	Palabra/Byte
------------------	-------	--------------

Función de correspondencia : $i = j \text{ módulo } m$

i : nº línea de caché.

j : nº de bloque de la memoria principal (dirección del bloque)

m : nº líneas de la memoria caché (nº de bloques de la caché)

Tag: identifica el nº de bloque.

Línea: línea de la caché ocupada por el bloque.

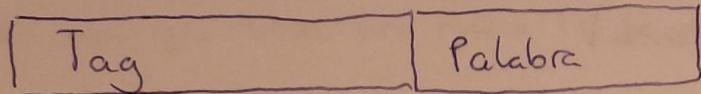
Palabra/Byte: identifica cada palabra dentro de un bloque.

Ventaja: es simple y poco costosa de implementar (requiere poco hardware).

Inconveniente: hace una posición concreta de caché para cada bloque dado. Puede existir un traspaso alternativo de bloques que ocupan el mismo lugar.

• Correspondencia ASOCIATIVA:

Un bloque de MP puede ir en cualquier línea de la caché.



Tag: identifica únicamente un bloque de MP.

Palabra: identifica cada palabra dentro de un bloque de MP.

Para determinar si un bloque está en la memoria caché, se debe examinar simultáneamente todas las etiquetas de las líneas de memoria caché. Si la etiqueta está, entonces es un "hit".

En caso de "hit" → la palabra escogida es la que corresponde en la caché.

"miss" → la dirección es la que nos dice dónde está en la MP.

Ventaja: mayor tasa de aciertos.

Inconveniente: necesidad de una circuitería bastante compleja.

No es tan rápida como la de acceso directo.

• Correspondencia ASOCIATIVA por CONJUNTOS:

Aunar las 2 correspondencias anteriores. MC dividida en T conjuntos de L líneas:

$$C = T \times L \quad ; \quad i = j \text{ módulo } T$$

nº líneas de MC nº de conjunto MC nº de bloque de MP .

Tag	Conjunto (Set)	Palabra / Byte
-----	----------------	----------------

Conjunto: especifica uno de los conjuntos de la MC. Lleva directamente a la zona de MC donde está el conjunto.

Tag + Conjunto: identifica un bloque de MP.

Tag: lleva al bloque del conjunto de MC.

ALGORITMOS de SUBSTITUCIÓN:

Algoritmos que se utilizan para quitar un elemento de la MC cuando hay que traer uno nuevo (y la correspondencia no es directa):

- LRU (Least Recently Used):

Requiere disponer de la "edad" de los bloques de la MC.

Es el algoritmo que aporta la mejor tasa de aciertos.

- FIFO (First In First Out):

- LFU (Least Frequently Used):

Se necesita registrar el número de accesos a los bloques.

El bloque que tenga el menor contador será sustituido.

- Aleatorio (no thrashing):

Tasa de aciertos menor al resto de métodos.

MEMORIA ASOCIATIVA (CAM)

Concepto: se caracteriza por el hecho de que el acceso a una posición de memoria se realiza especificando su contenido o parte de él y no su dirección.

En software \rightarrow Array asociativo ; En hardware \rightarrow Memoria CAM.

Estructura: "n" palabras con "m" bits/palabra (ORGANIZACIÓN)

\rightarrow Registro Argumento (RA) : "m" bits , lo que quiero averiguar si está en el contenido de la memoria .

\rightarrow Registro Máscara (k) : "m" bits , podré quedarme con el que quiero averiguar que esté en la memoria .

\rightarrow Registro Marca (M) : "n" bits , pone un bit a 1 donde encuentre el argumento buscado y a 0 donde NO esté. Sirve para saber si en esa palabra (fila) está el argumento que busco.

0	San Sebastián	0
---	---------------	---

\leftarrow A

0	1	0
---	---	---

\leftarrow K

Aitor	Vitoria	945
G.Ilen	San Sebastián	788
Juan	Pamplona	946
Naic	San Sebastián	944

0
1
0
1

\leftarrow M

En general, para una máscara dada no hay dos filas iguales contenidas en la memoria CAM .

Los CAM se utilizan con memorias caché de forma que la identificación de la etiqueta de cada línea se realice de forma simultánea . (P.e. : TAG RAM \rightarrow Pentium de Intel)

Tiempo de acceso a una CAM : 4-20 ns .

MEMORIA VIRTUAL

Concepto: "virtual" hace referencia a sistemas en los que el usuario dispone de un espacio de direcciones virtual más grande que el número de posiciones de memoria real con que cuenta.

Cuando un programa es muy grande y no cabe en la memoria principal (MP), el programador lo dividía en bloques (Superposiciones, overlays) y se encargaba de cargar y descargar de memoria los bloques que fueran necesarios durante la ejecución del programa.

Solución: "Memoria Virtual", es decir, separar los conceptos de espacio de direcciones y posiciones de memoria.

Además, gestiona automáticamente los dos niveles de la jerarquía de memoria principal - memoria secundaria (MS).

Por tanto, permite usar el disco duro como una extensión de la RAM.

Diseño: dividir tanto el espacio de direcciones virtual como el espacio de direcciones real en BLOQUES de igual tamaño.

En un sistema paginado, un bloque se denomina página y un falso de memoria virtual, falta de página.

Tamaño de página: potencia de 2; $512 - 4096$ palabras

La CPU direcciona 2^n direcciones

Tamaño real MP es 2^m

Los programas verán MP de 2^n (memoria virtual)

Definiciones:

- Dirección virtual: direcciones lógicas que usa el proceso.
- Dirección física: dirección real en memoria física.
- Mapeado: mecanismo por el que las direcciones virtuales se traducen a direcciones física.
- Marco de página: cada uno de los bloques en los que se divide la MP (potencia de 2)
- Página: cada uno de los bloques en los que se divide la memoria Virtual (de igual tamaño que los Marcos de página). Las páginas se almacenan en MS hasta que se necesiten.
- Paginación: proceso de copiado de una página del disco duro a un marco de página en MP.
- Fallo de página: evento que sucede cuando se solicita una página que no está en MP.

Implementación:

La CPU produce direcciones virtuales que hay que interpretar; traducir en una dirección real de la MP.

Los páginas tienen una ubicación fija en la MS y variable en la MP.

→ Dirección Virtual:

- N° de página: número de página en la MS (MSB). Determina el número de páginas direccionables.
- Desplazamiento: localización de la palabra dentro de la página.

→ Dirección real:

• N° de marco: número de marco en la MP. Determina el nº de marcos direccionables de memoria real.

• Desplazamiento: localización de la palabra en el marco.

MP		MS	
000	A	0000	
001	B	0001	
010	C	0010	
011	D	0011	
100	E	0100	A
101	F	0101	B
110	G	0110	C
111	H	0111	D
		1000	
		1001	
		1010	
		1011	
		1100	E
		1101	F
		1110	G
		1111	H

a) 64K de memoria, 2 bit/página
4 páginas

$$16 \text{ bits} - 2 \text{ bits} = 14 \text{ bits} \rightarrow 4 \text{ páginas de } 16 \text{ K} \\ (2^{14} = 16 \text{ K})$$

b) 64K de memoria, 4 bit / página
16 páginas

$$16 \text{ bits} - 4 \text{ bits} = 12 \text{ bits} \rightarrow 16 \text{ páginas de } 4 \text{ K} \\ (2^{12} = 4 \text{ K})$$

¿Cómo se calculan las correspondencias \uparrow ?

a) $64 \text{ K} = 2^6 \cdot 2^{10} = 2^{16} \rightarrow 16 \text{ bits}$

Tamaño de página: 2 bit $\rightarrow 2^2 = 4 \text{ páginas}$

$$16 \text{ bits} - 2 \text{ bits} = 14 \text{ bits} \rightarrow 2^{14} = 16 \text{ K} = 2^4 \cdot 2^{10} = 16 \text{ K}$$

b) $64 \text{ K} = 2^6 \cdot 2^{10} = 2^{16} \rightarrow 16 \text{ bits}$

Tamaño de página: 4 bit $\rightarrow 2^4 = 16 \text{ páginas}$

$$16 \text{ bits} - 4 \text{ bits} = 12 \text{ bits} \rightarrow 2^{12} = 2^2 \cdot 2^{10} = 4 \text{ K}$$

Resultado: 16 páginas de 4K

Gestión: Una página puede residir en cualquier marco, por lo que se necesita un mecanismo para encontrarla y relacionar "marcos" con "páginas", ("Table de Páginas")

→ Si la página no está en MP:

1. Procede fallo de página.

2. Expulsa un marco y trae la página según la política de Sustitución.

La CPU se complementa con MMU (Memory Management Unit), que libera a la MP de la Tabla de Páginas, ubicándose en registros de MMU.

DISPOSITIVOS de E/S

Introducción: para poder interactuar con el exterior el ordenador tendrá unos determinados dispositivos, denominados periféricos.

Según en qué sentido se transfieran los datos, serán de:

→ Entrada: teclado, ratón, micrófono, webcam, escáner...

→ Salida: monitor, impresora, altavoz...

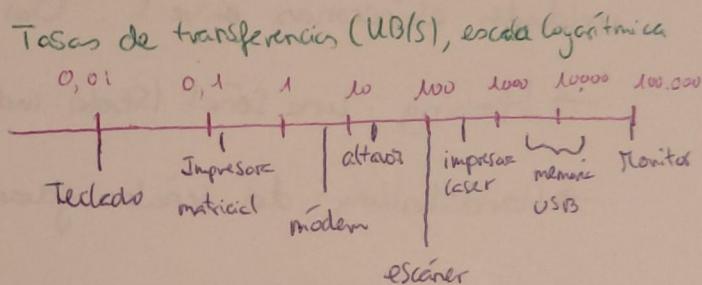
→ e/S: pantalla táctil, módem, discos regrabables, memoria USB (flash)...

Además, tienen varios diferenciaos entre ellos:

→ electromecánica

→ información y formatos

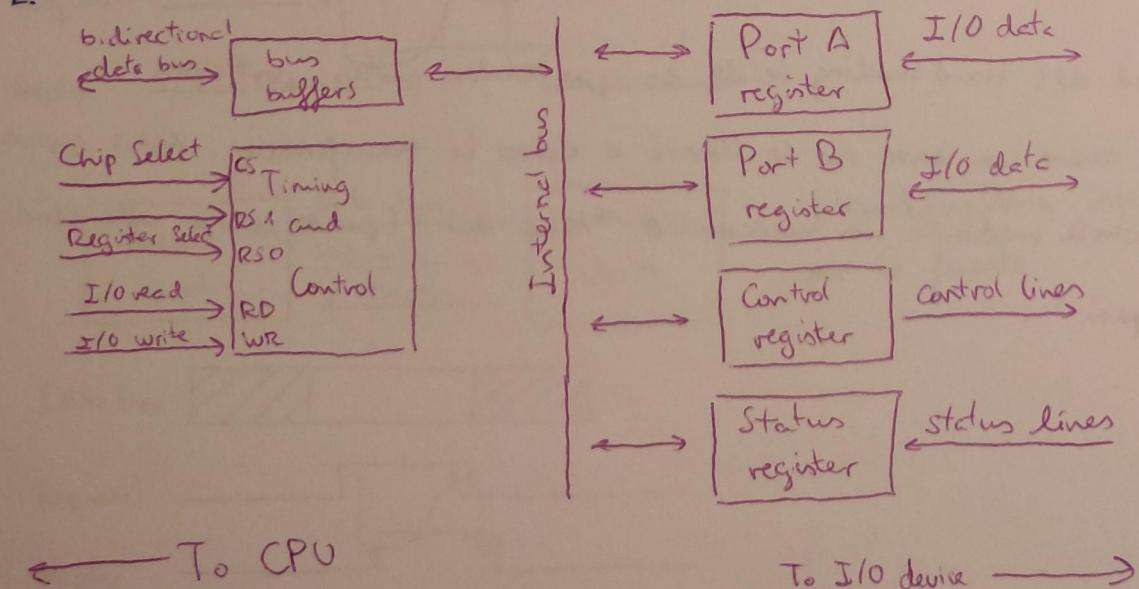
→ temporización



Controlador: se hace necesario el uso de controladores de periféricos.

Cada dispositivo tendrá su propio controlador conectado al bus de expansión.

Estructura de un controlador:



Estructura CPU-Periféricos: la CPU debe comunicarse con la unidad de memoria y con los dispositivos de E/S a través de un bus de direcciones y otro de datos:

→ E/S mapeada en memoria: buses de datos, direcciones y control comunes (amplio repertorio de instrucciones)

→ E/S independiente (aislada): buses de datos y direcciones comunes, pero diferentes líneas de control (rango completo de direcciones para ambos).

Sincronización: la CPU, la interfaz y el dispositivo de E/S son unidades asíncronas entre sí. Dos métodos de sincronización:

→ Strobing: una señal (Strobe) indica a la otra parte que quiere read/write.

→ Handshaking: dos señales garantizan la fiabilidad.

Strobing es sencillo, pero:

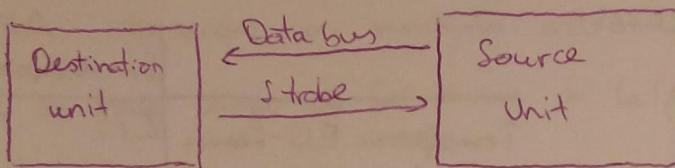
1. La fuente no sabe si el destino ha recibido el dato.

2. El destino no sabe si ha leído los datos deseados (no se le asegura que la fuente haya puesto los datos en el bus).

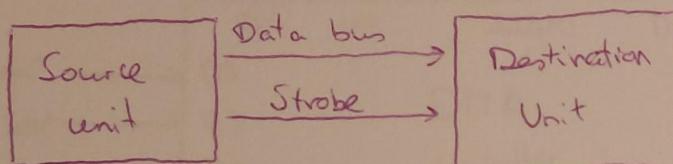
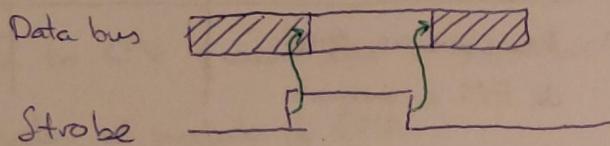
3. El tiempo para cualquier transferencia deberá ser el de la unidad más lenta.

La fiabilidad del Handshaking reside en que ambas partes participan.

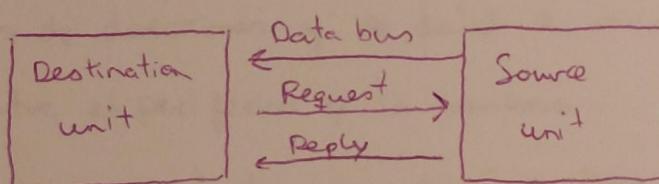
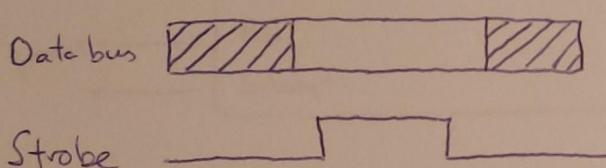
Si una parte tiene un error no se llevará a cabo la transferencia. Estos errores pueden detectarse mediante un mecanismo "time-out" (puede usarse esta señal como interrupción).



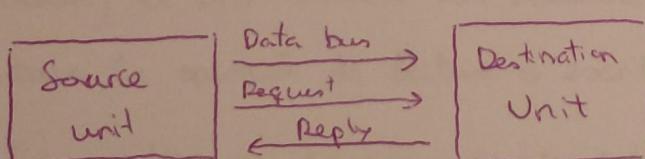
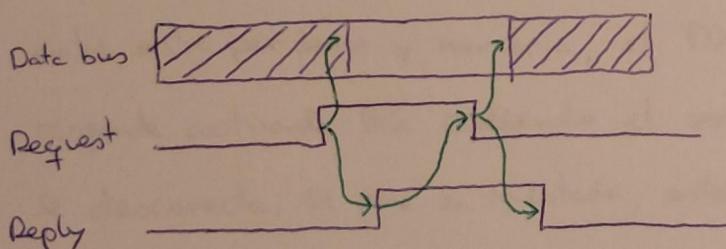
Strobing iniciada en destino



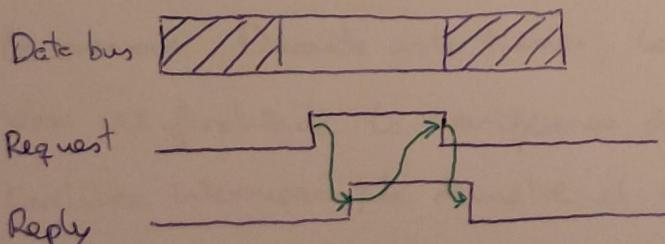
Strobing iniciada por la fuente



Handshaking iniciada por la destino



Handshaking iniciada por la fuente



Comunicación CPU - Periféricos :

→ E/S programada (polling)

→ E/S mediante interrupciones

→ DMA (Acceso directo a memoria)

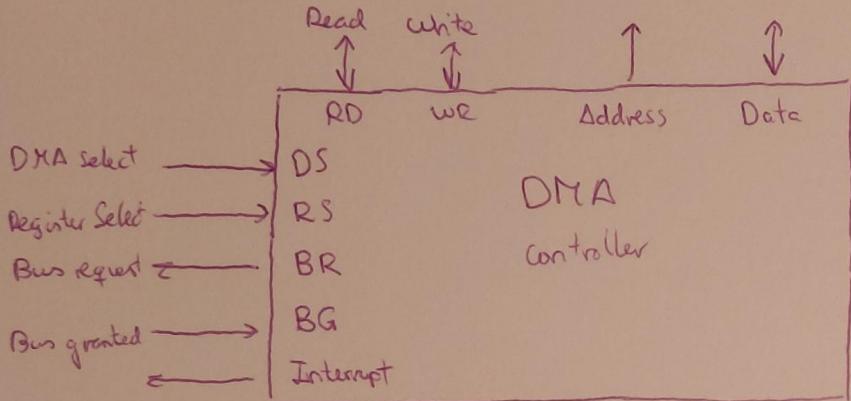
	Sin interrupciones	Con
E/S	E/S programada	E/S mediante interrupciones
DMA		DMA

Transferencia E/S - memoria
o través de CPU

Transferencia directa
de E/S a memoria

DMA: requiere un módulo adicional en el bus del sistema que se hace cargo de los buses de la CPU, quien tiene prohibido temporalmente el acceso a memoria y el control de sus buses.

Si la CPU quiere leer o escribir un bloque de datos envía una orden al módulo de DMA:



Funcionamiento: la CPU se comunica con el DMA a través del bus de direcciones y de datos e inicializa la transferencia de datos entre el periférico y la memoria.

Cuando se solicita una petición de DMA para realizar una transferencia de datos entre periférico y memoria, el DMA activa BR y la CPU responde activando BG cediendo el control de los buses al DMA (CPU se desconecta, se pone en triestado, estado de alta impedancia, CPU disabled).

Entonces, el DMA manda un reconocimiento de DMA al periférico y controla la operación de lectura o escritura entre periférico y memoria.

Así se realiza una transferencia de datos directa entre el periférico y la memoria. Durante este tiempo, la CPU no puede acceder al bus.

Una vez finalizada la transferencia de datos, el DMA avisa a la CPU con una interrupción y le devuelve el control del bus. (BR=0)

* Observaciones: los 4 elementos (CPU, memoria, DMA, periférico) están conectados al bus de datos y de direcciones. El periférico, concretamente se conecta al bus de direcciones mediante el DMA.