

Grupo ARCOS

uc3m | Universidad **Carlos III** de Madrid

Tema 4 (I)

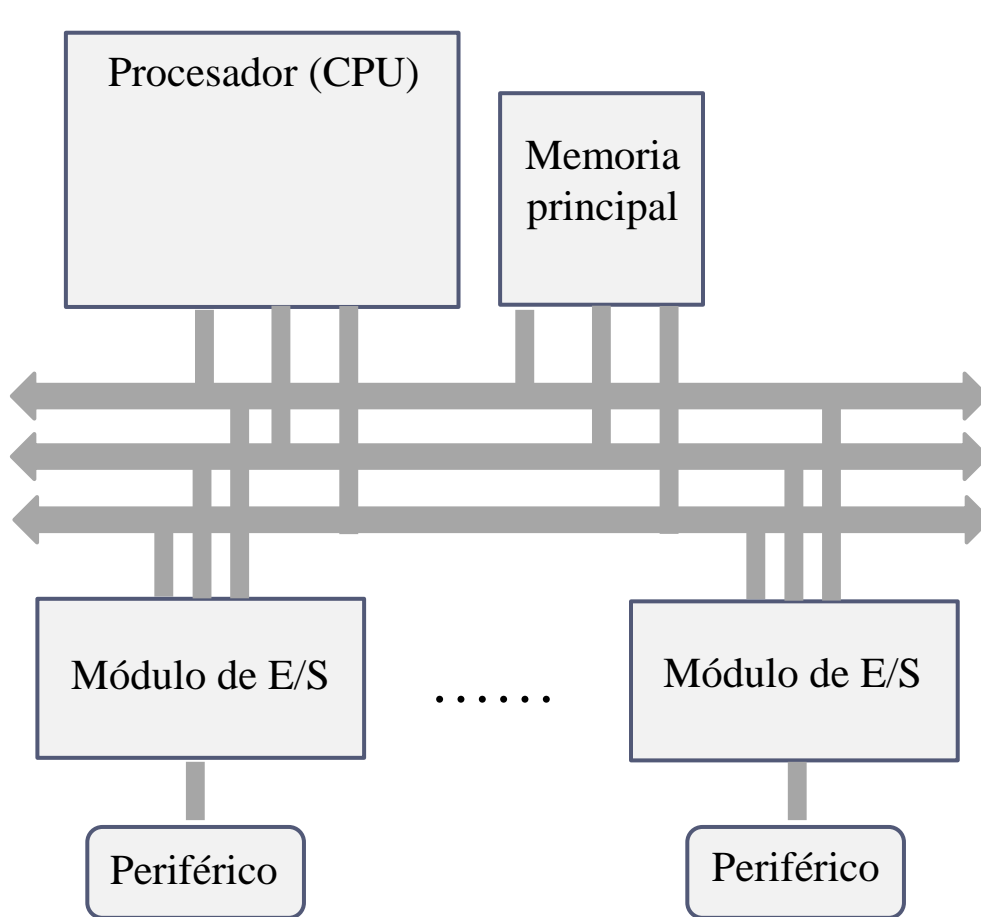
El procesador

Estructura de Computadores
Grado en Ingeniería Informática

Contenido

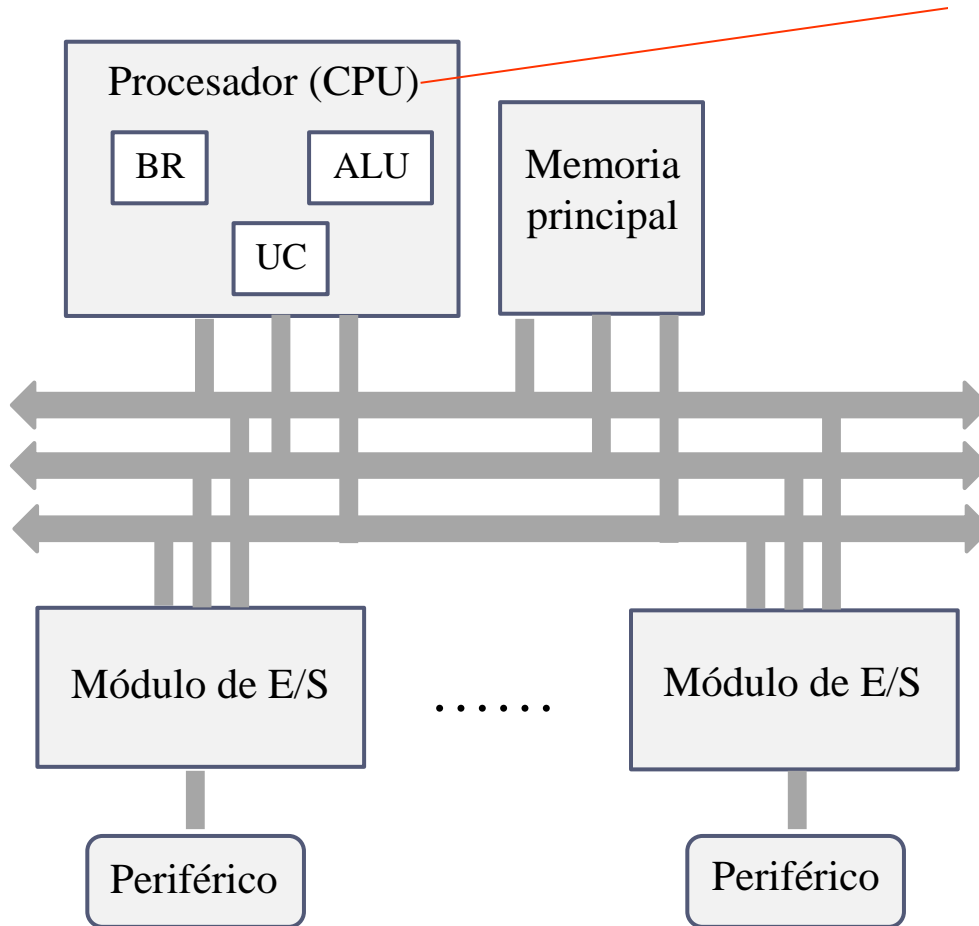
1. Elementos de un computador
2. Organización del procesador
3. La unidad de control
4. Ejecución de instrucciones
5. Diseño de la unidad de control
6. Modos de ejecución
7. Interrupciones
8. Arranque de un computador
9. Prestaciones y paralelismo

Componentes de un **computador** **recordatorio**



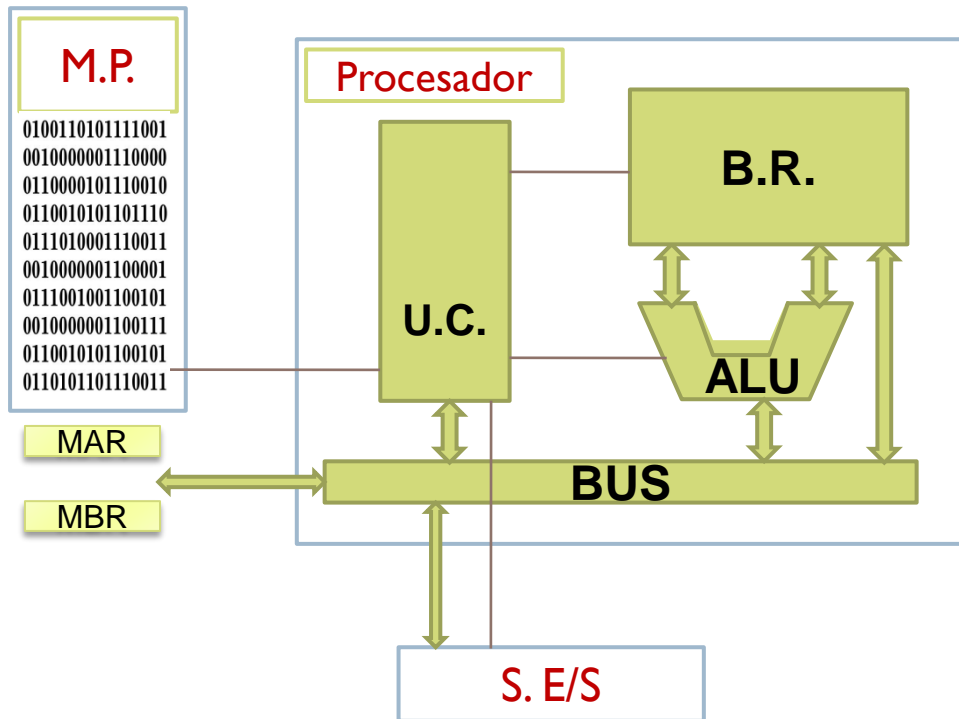
- ▶ Procesador
- ▶ Memoria principal
- ▶ Módulo de E/S
 - ▶ Periférico

Componentes de un **procesador** **recordatorio**



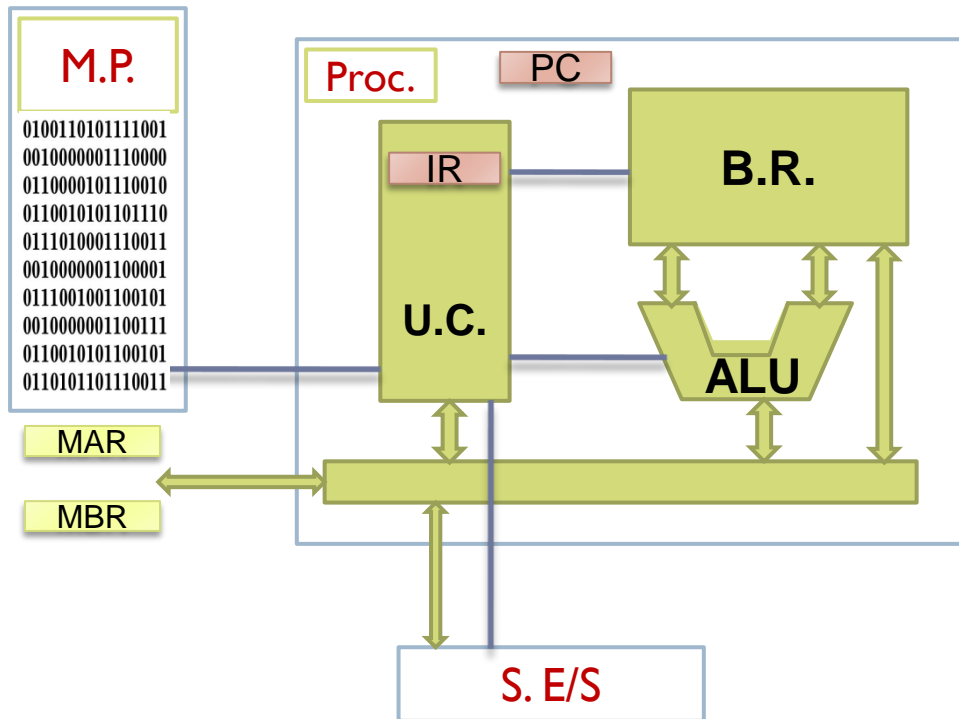
- ▶ Banco de registros
- ▶ Unidad aritmético-lógica
- ▶ Unidad de control
- ▶ Memoria caché

Motivación



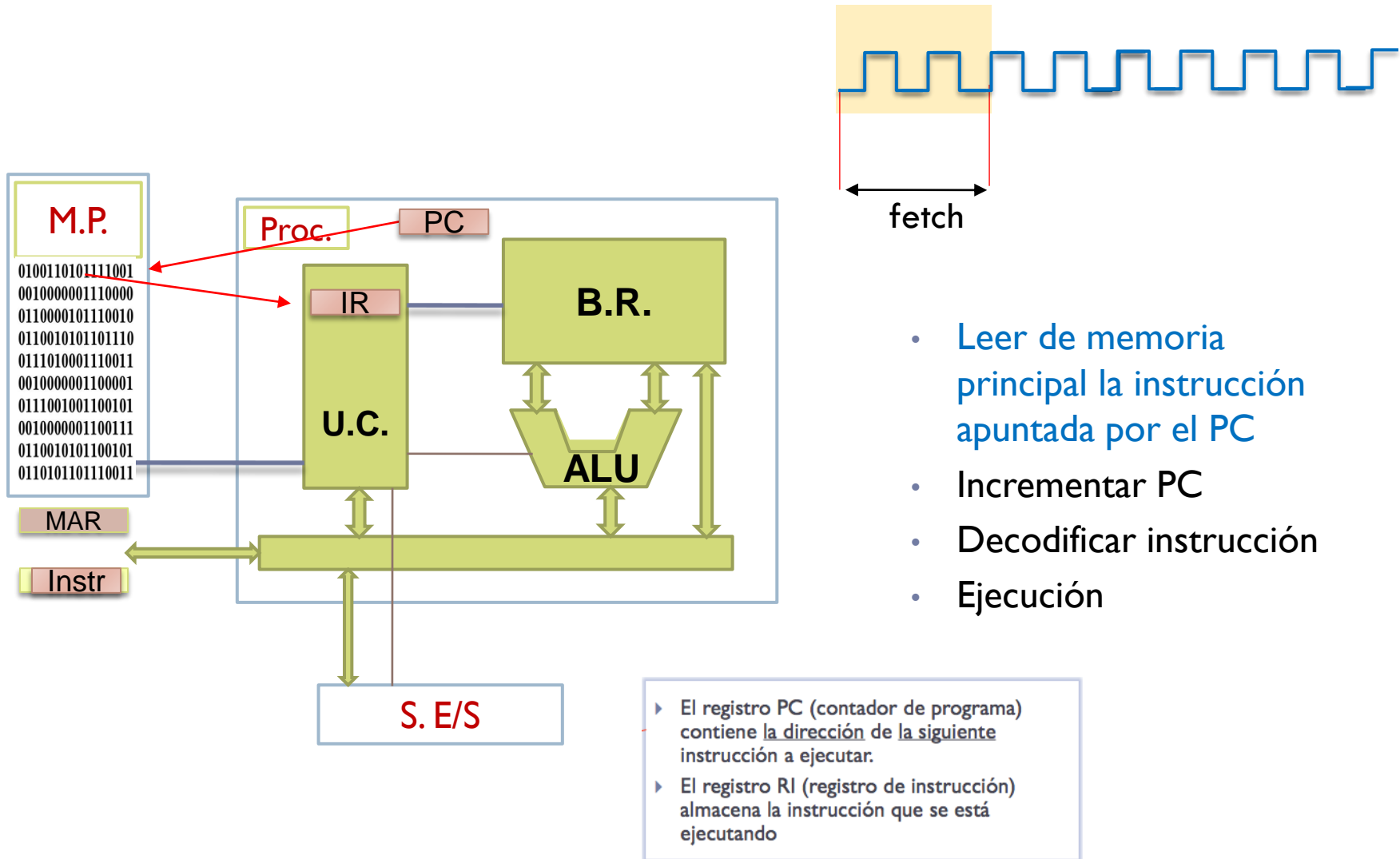
- En el tema 3 se estudian las instrucciones máquina
- En el tema 4 se estudia cómo se ejecutan las instrucciones en el computador

Funcionamiento básico de la U.C.

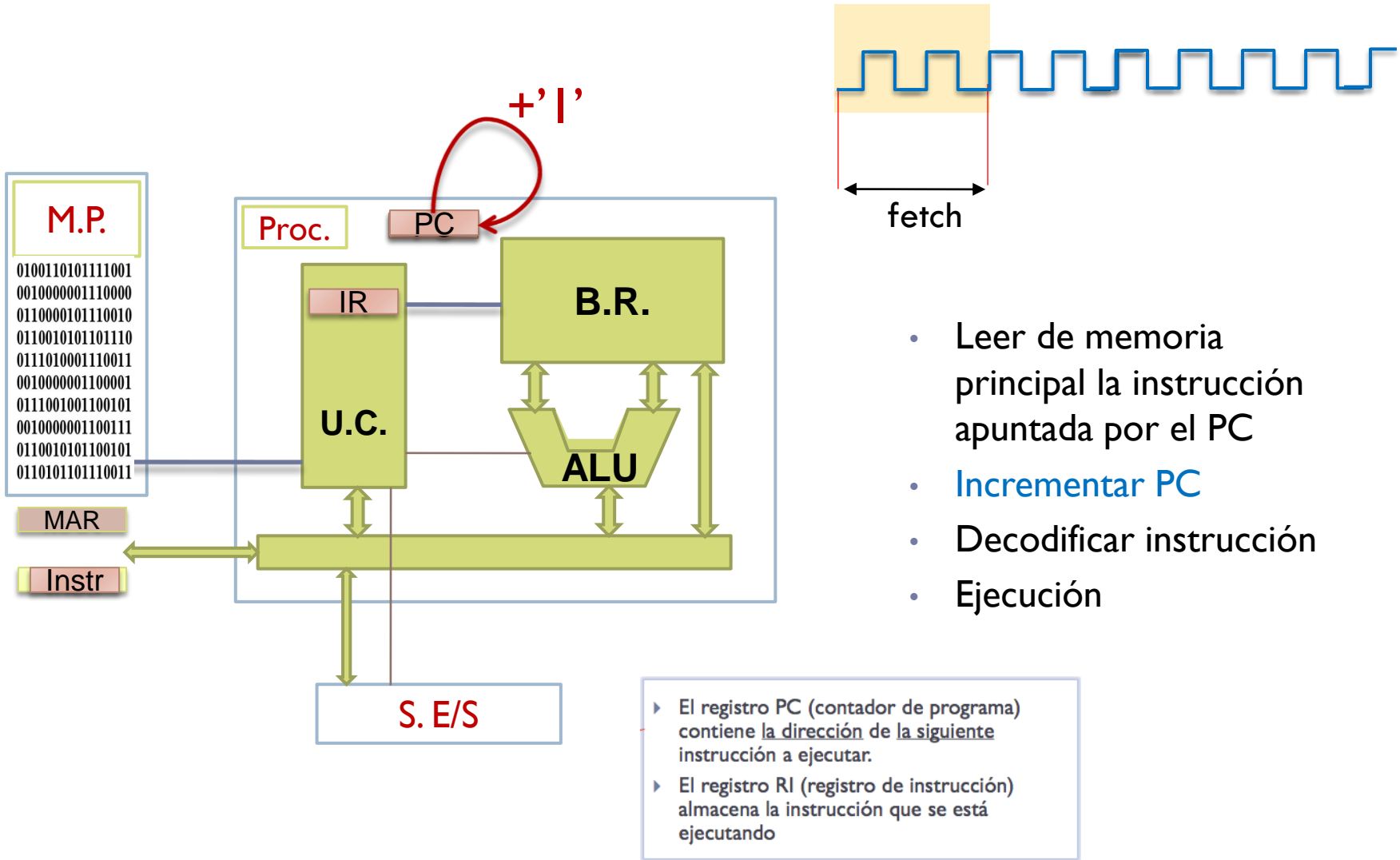


- **Ejecutar instrucciones máquina**
 - En cada ciclo de reloj envía la Unidad de Control (U.C.) por los cables del bus de control las señales de control
 - Cada elemento del computador tiene entradas, salidas y señales de control que indican qué valor a la salida se ha de tener:
 - Mover de una entrada a salida: $S=E_x$
 - Transformar una entrada: $S=f(E)$

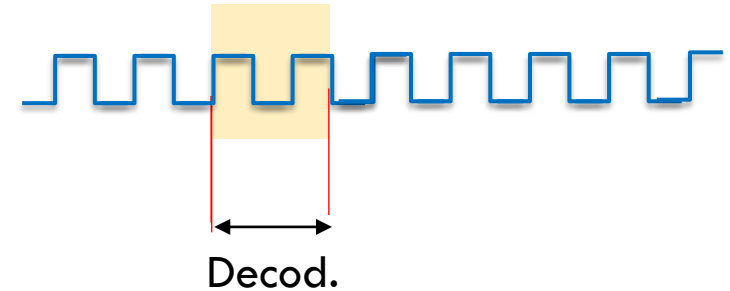
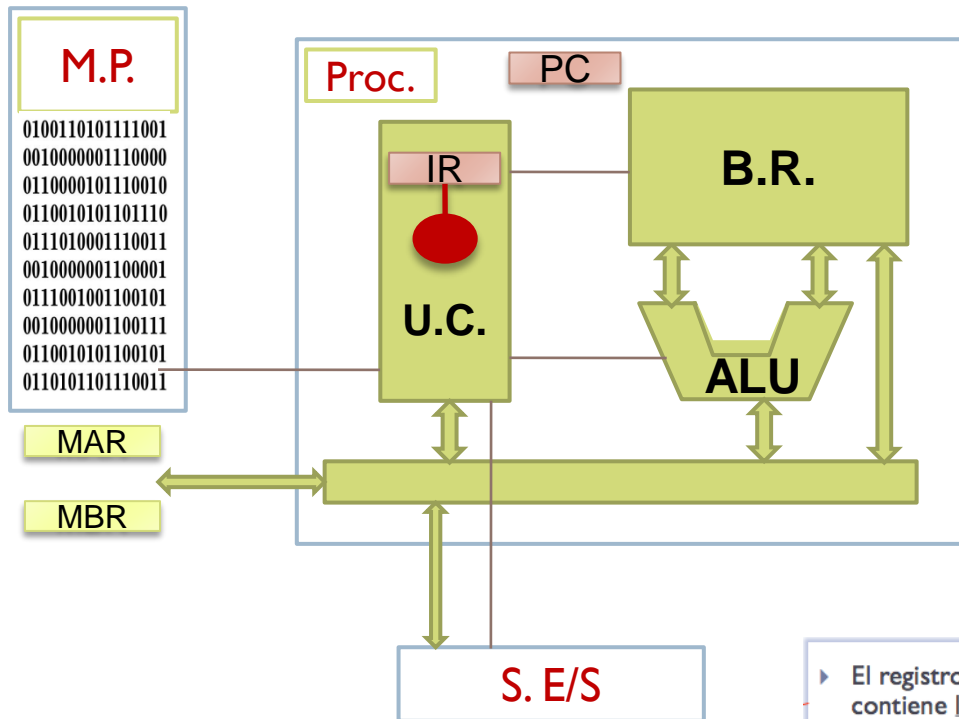
Funcionamiento básico de la U.C.



Funcionamiento básico de la U.C.



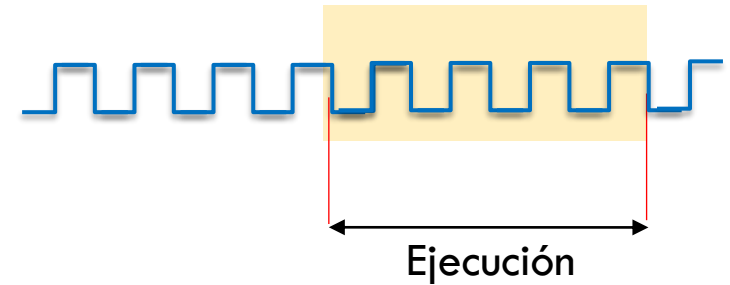
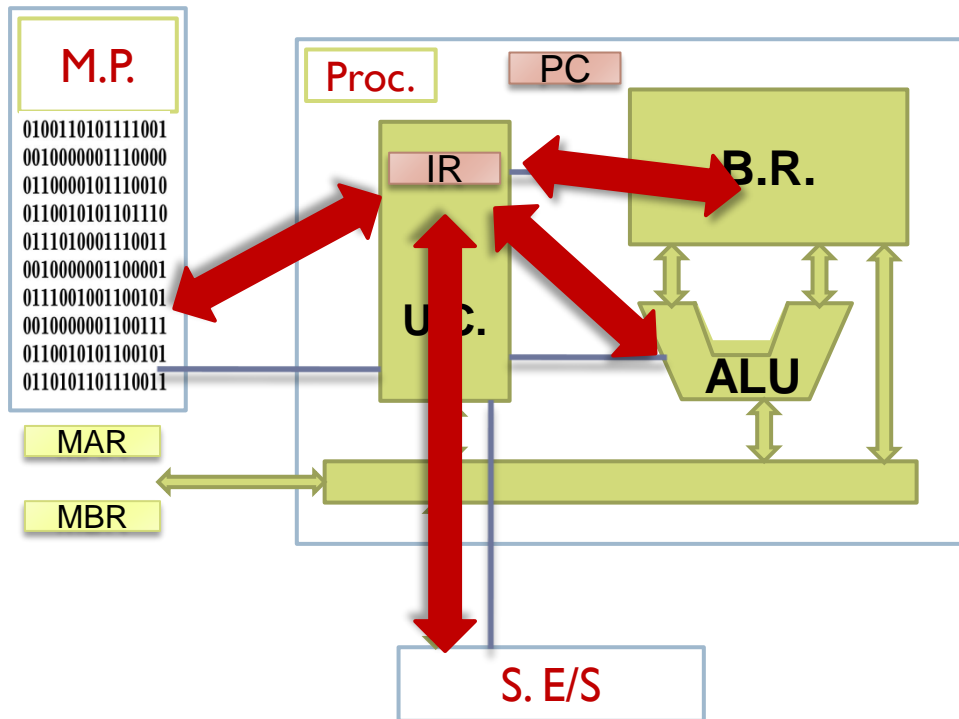
Funcionamiento básico de la U.C.



- Leer de memoria principal la instrucción apuntada por el PC
- Incrementar PC
- **Decodificar instrucción**
- Ejecución

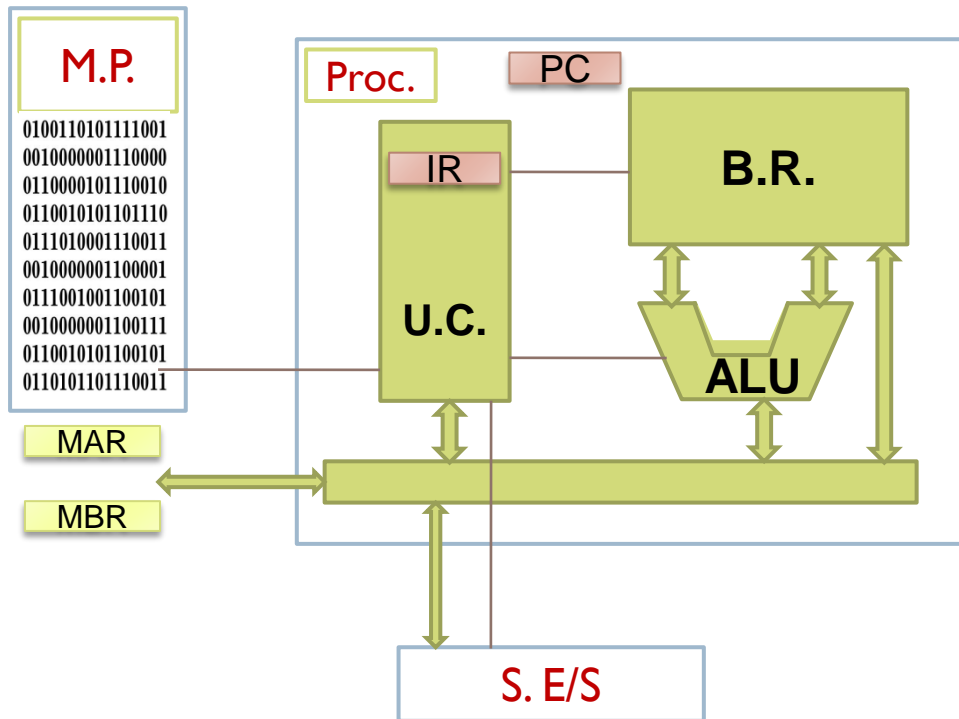
- ▶ El registro PC (contador de programa) contiene la dirección de la siguiente instrucción a ejecutar.
- ▶ El registro RI (registro de instrucción) almacena la instrucción que se está ejecutando

Funcionamiento básico de la U.C.



- Leer de memoria principal la instrucción apuntada por el PC
- Incrementar PC
- Decodificar instrucción
- Ejecución

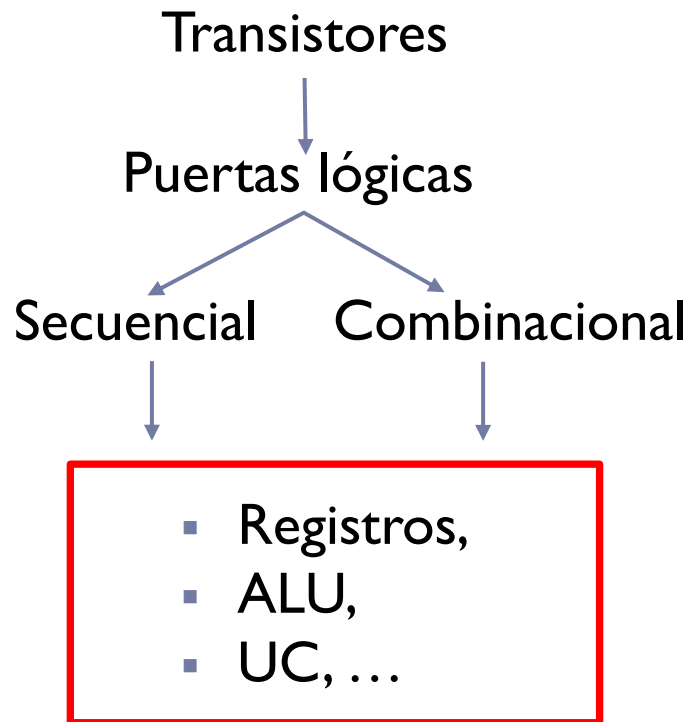
Otras funciones de la U.C.



- Resolver situaciones anómalas
 - Instrucciones ilegales
 - Accesos a memoria ilegales
 - ...
- Atender las interrupciones
- Controlar la comunicación con los periféricos

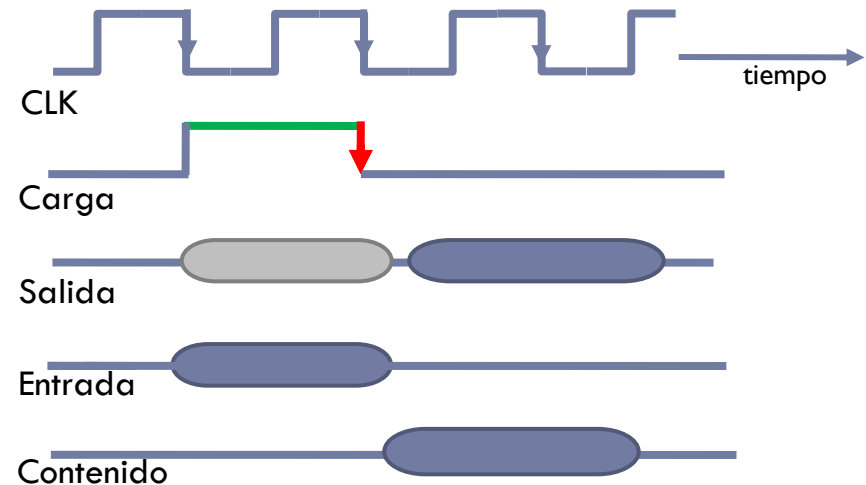
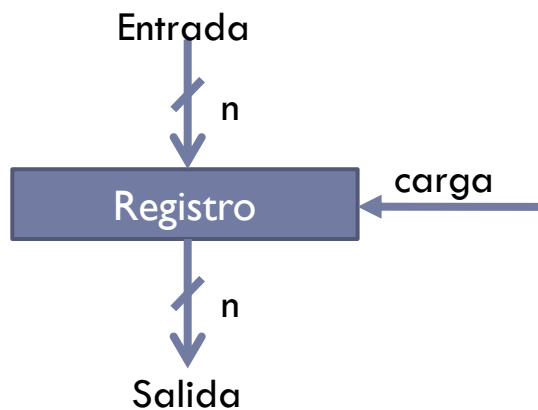
Recordatorio

- ▶ Sistema digital basado en 0 y 1
- ▶ Elementos constructivos:

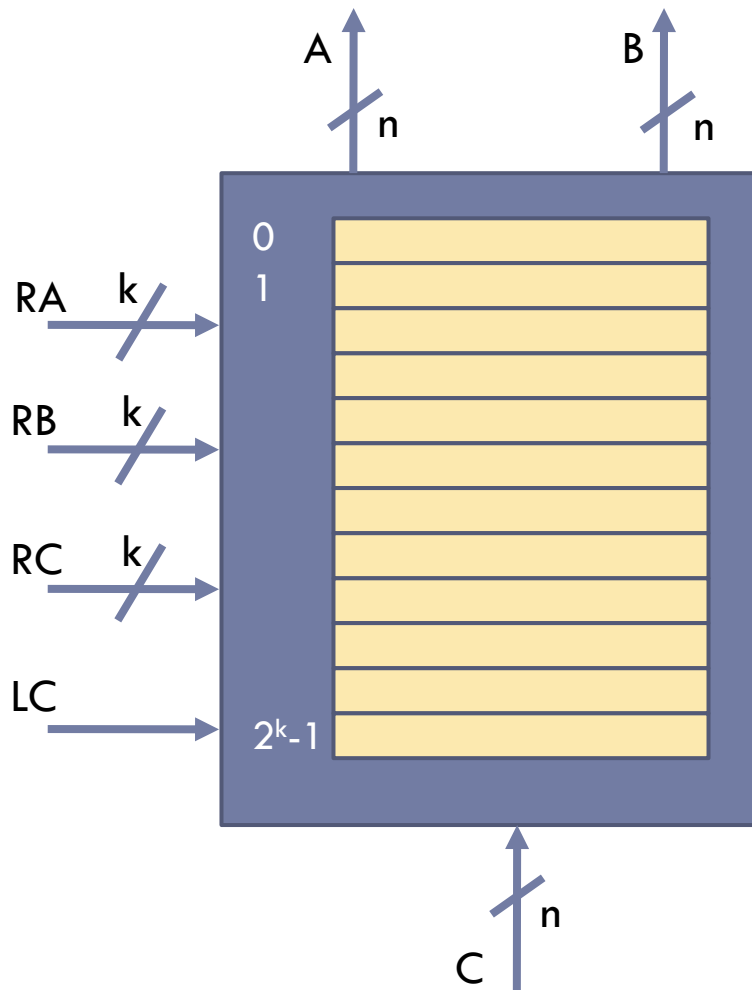


Registro

- ▶ Elemento que almacena n bits a la vez
 - ▶ Salida: 1
 - ▶ Durante **el nivel** la salida es el valor guardado en el registro
 - ▶ Entrada: 1
 - ▶ Posible nuevo valor a guardar
 - ▶ Control: 1 o 2
 - ▶ Carga: en el **flanco de bajada** se guarda el nuevo valor
 - ▶ Reset: puede existir una señal para poner el registro a cero

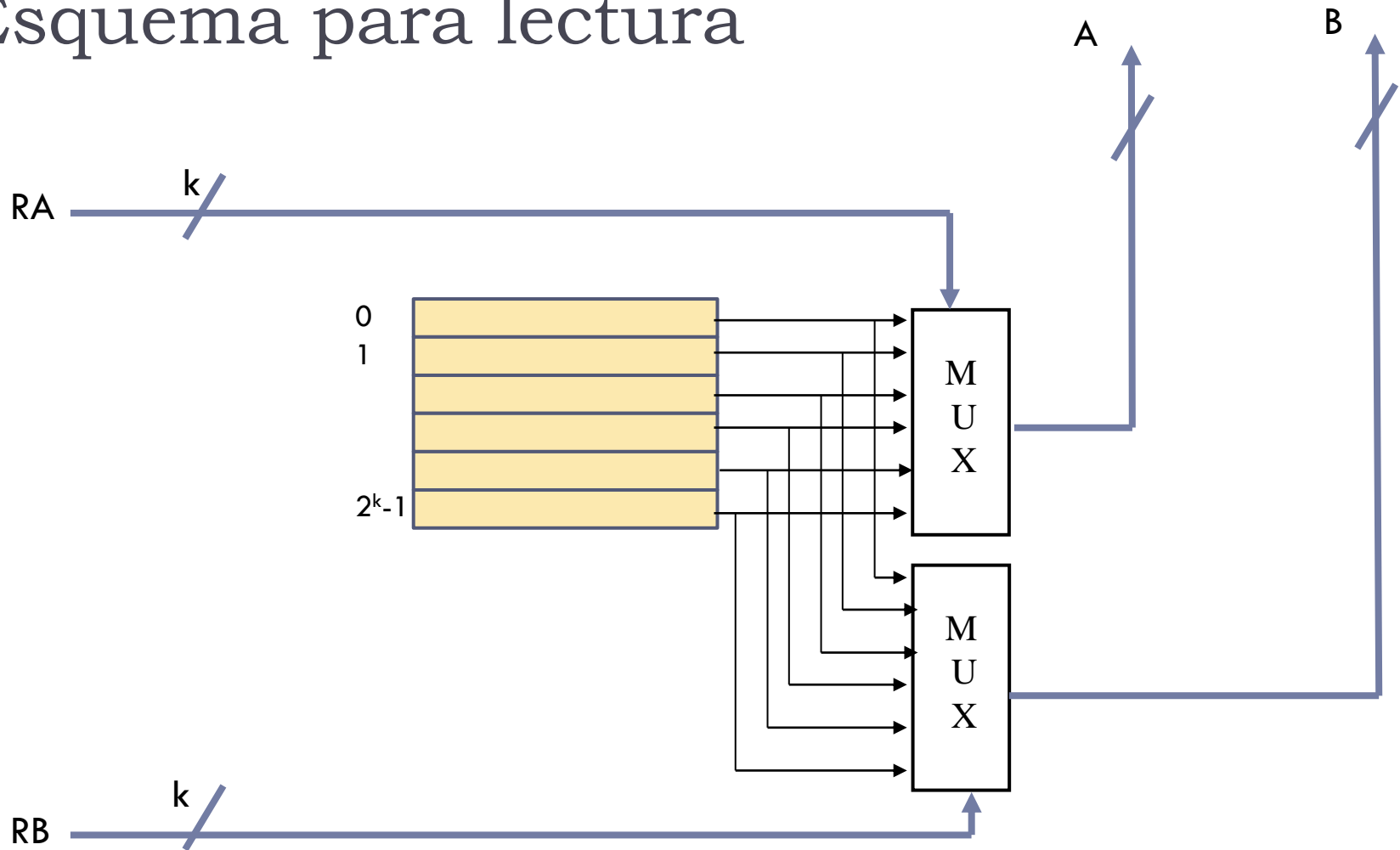


Banco de registros (BR)



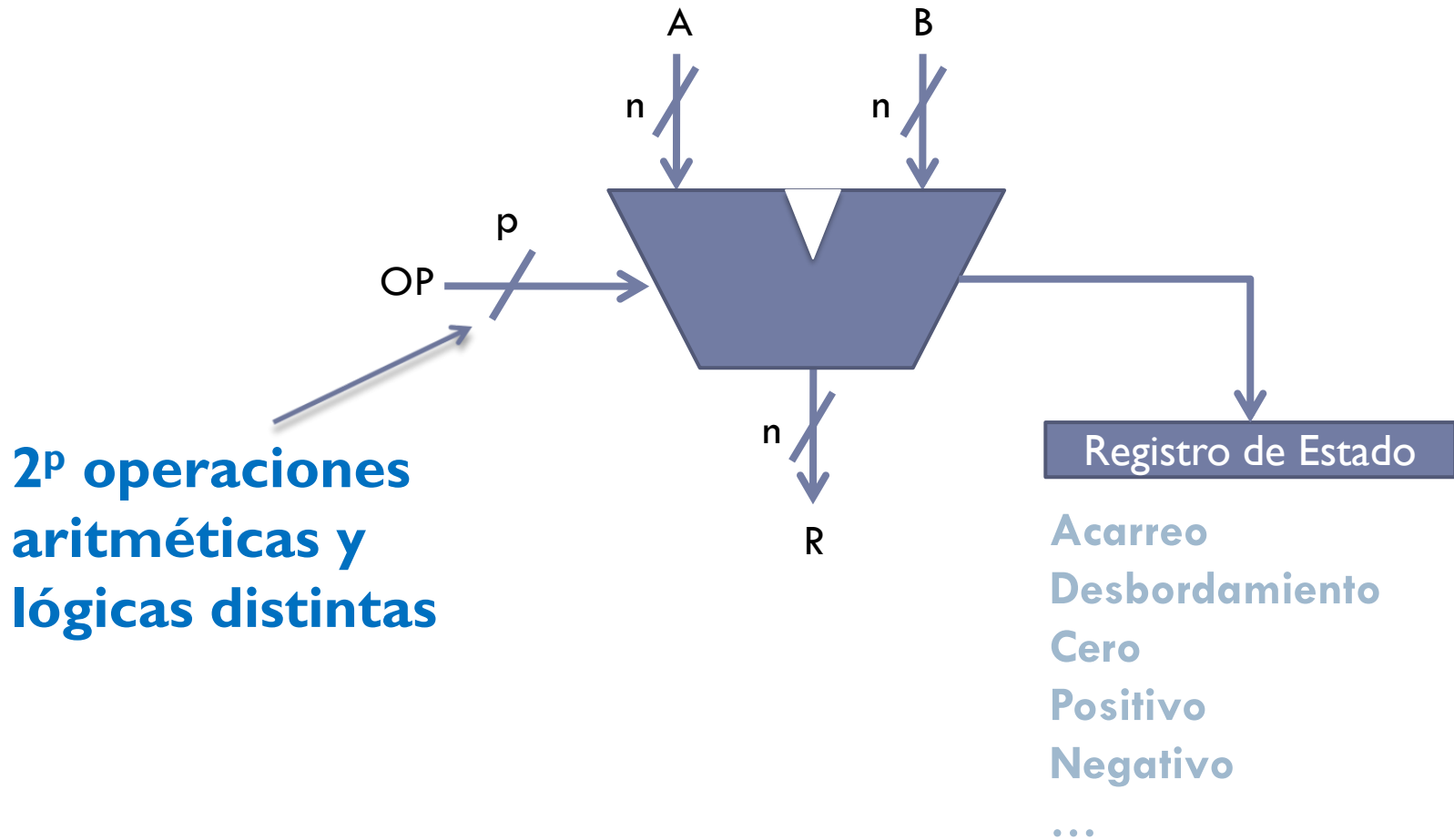
- ▶ Agrupación de registros.
 - ▶ Típicamente un número de registros potencia de 2.
 - ▶ n registros $\rightarrow \log_2 n$ bits para seleccionar cada registro
 - ▶ k bits de selección $\rightarrow 2^k$ registros
 - ▶ Ej.: con 32 registros, $k=5$
 - ▶ Elemento fundamental de almacenamiento.
 - ▶ Acceso muy rápido.
- ¿Qué valor tiene que tener RA para sacar por A el contenido del registro 14?

Esquema para lectura

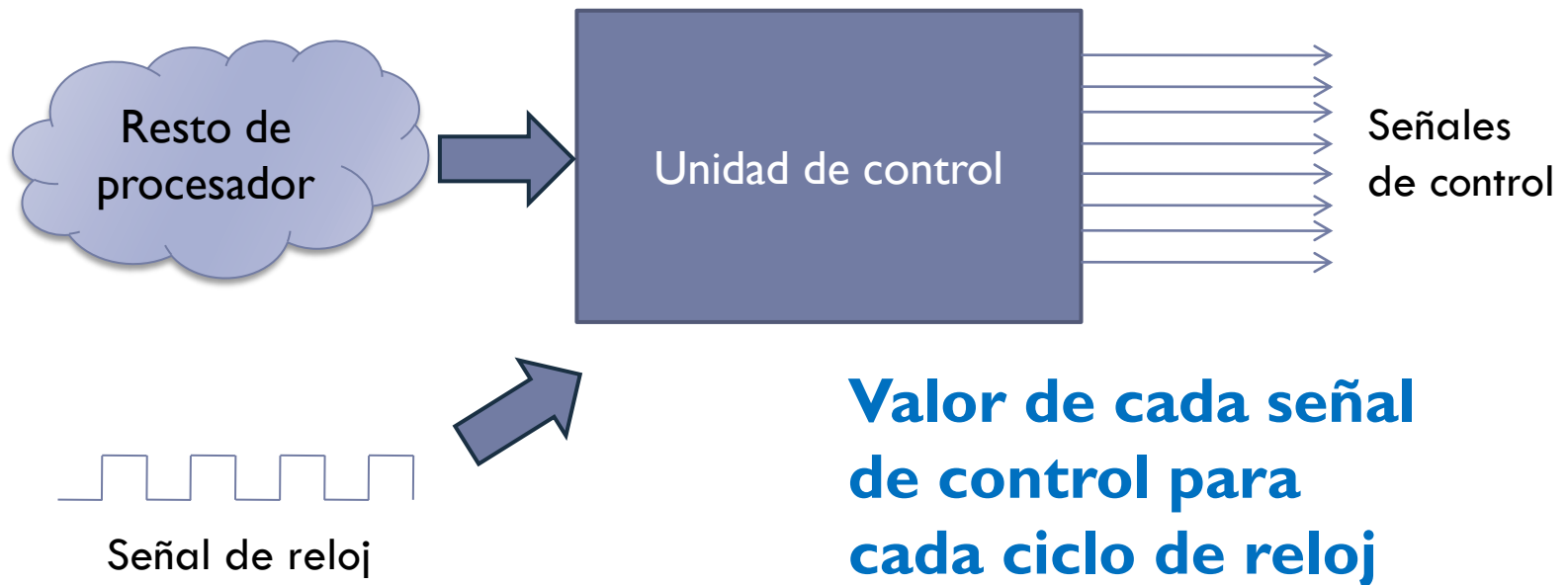


¿Qué valor tiene que tener RA para sacar por A el contenido del registro 14?

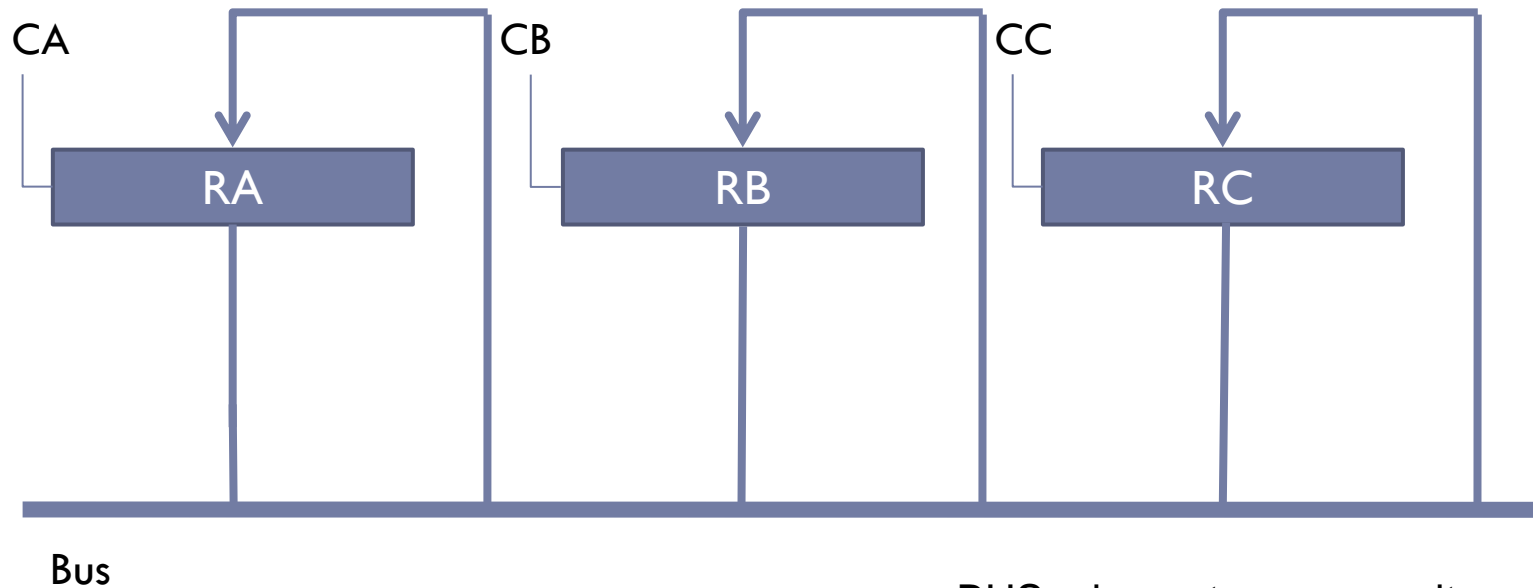
Unidad aritmético lógica (ALU)



Unidad de control (UC)

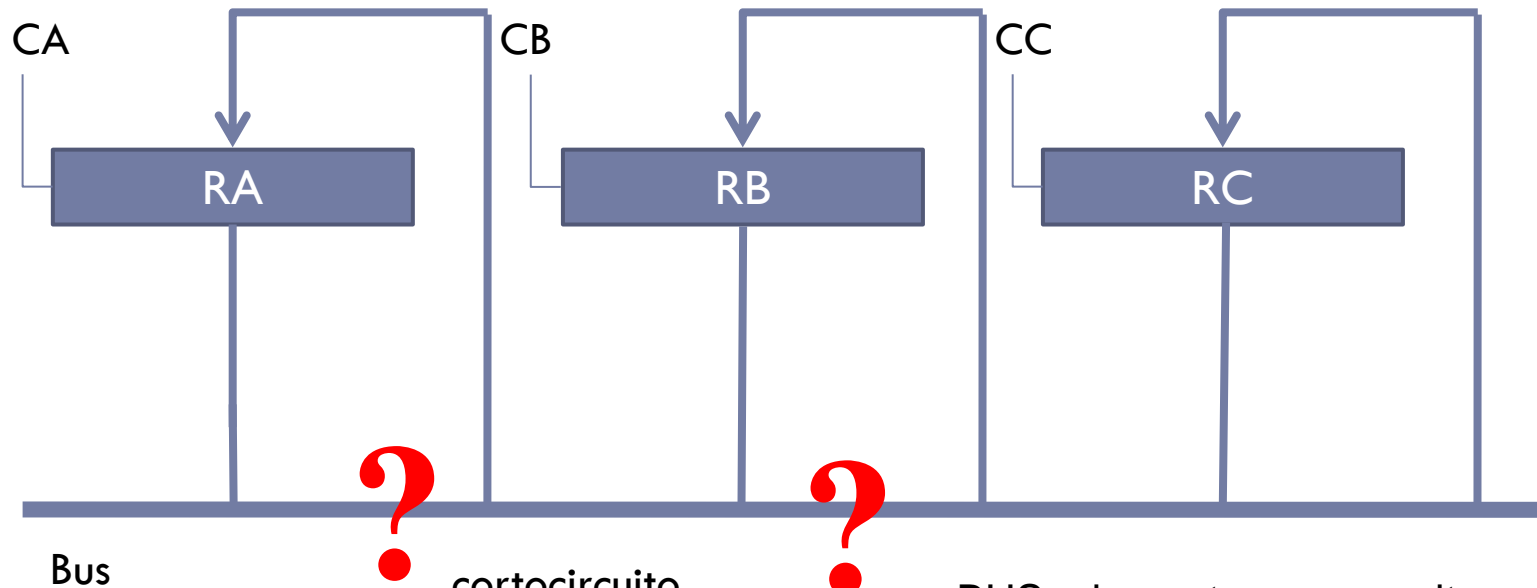


Conexión de registros a un bus



BUS: elemento que permite transmitir varios bits entre elementos de almacenamiento

Conexión de registros a un bus

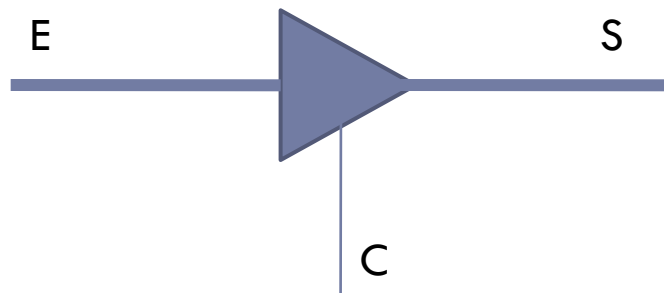


cortocircuito
múltiples conexiones a
un mismo punto

BUS: elemento que permite
transmitir varios bits entre
elementos de almacenamiento

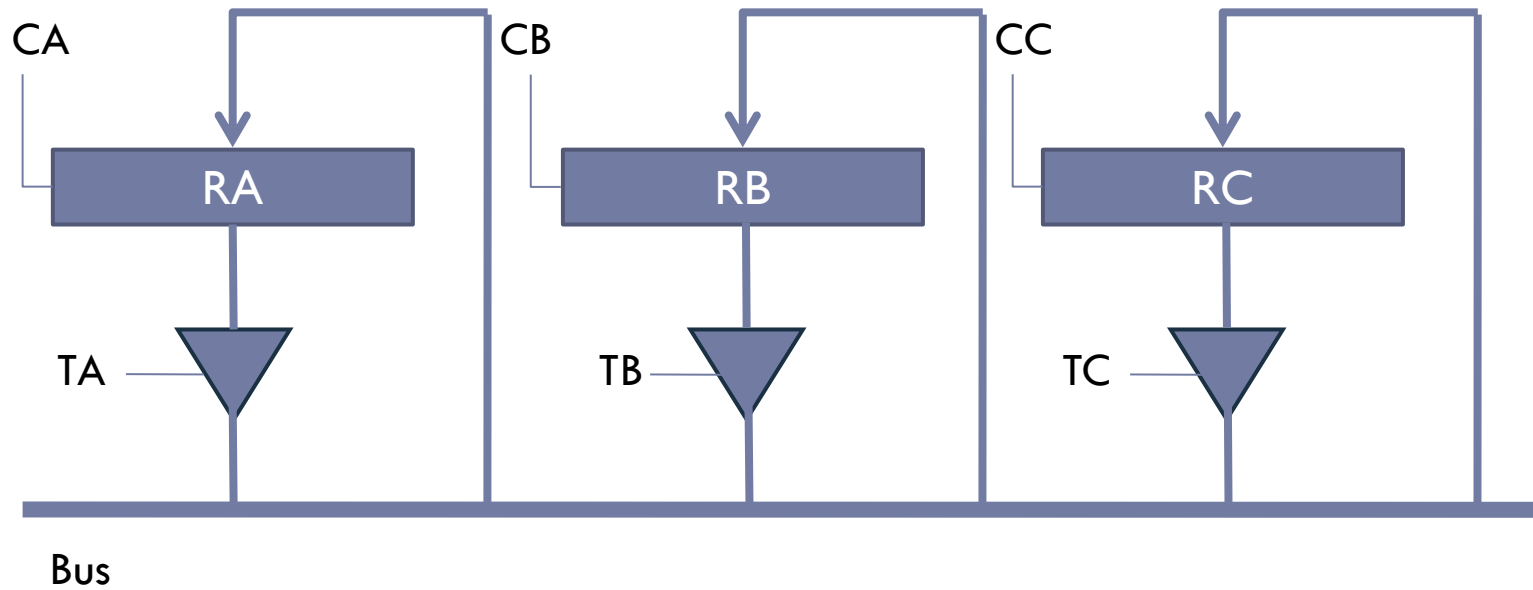
Búffer triestado

- ▶ Tipo especial de puerta lógica que puede poner su salida en alta impedancia (Z)
- ▶ Útil para permitir múltiples conexiones a un mismo punto

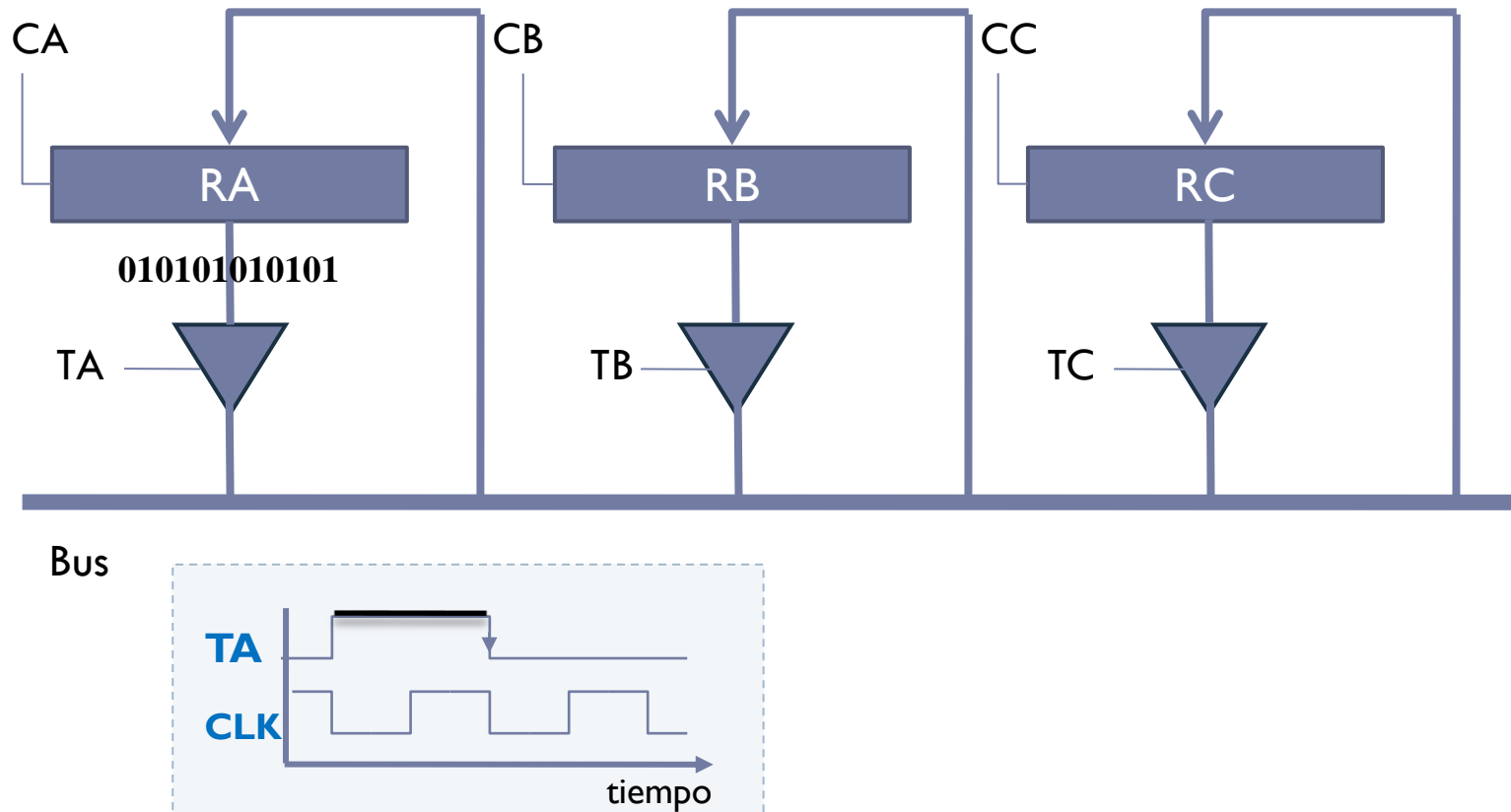


E	C	S
0	0	Z
1	0	Z
0	1	0
1	1	1

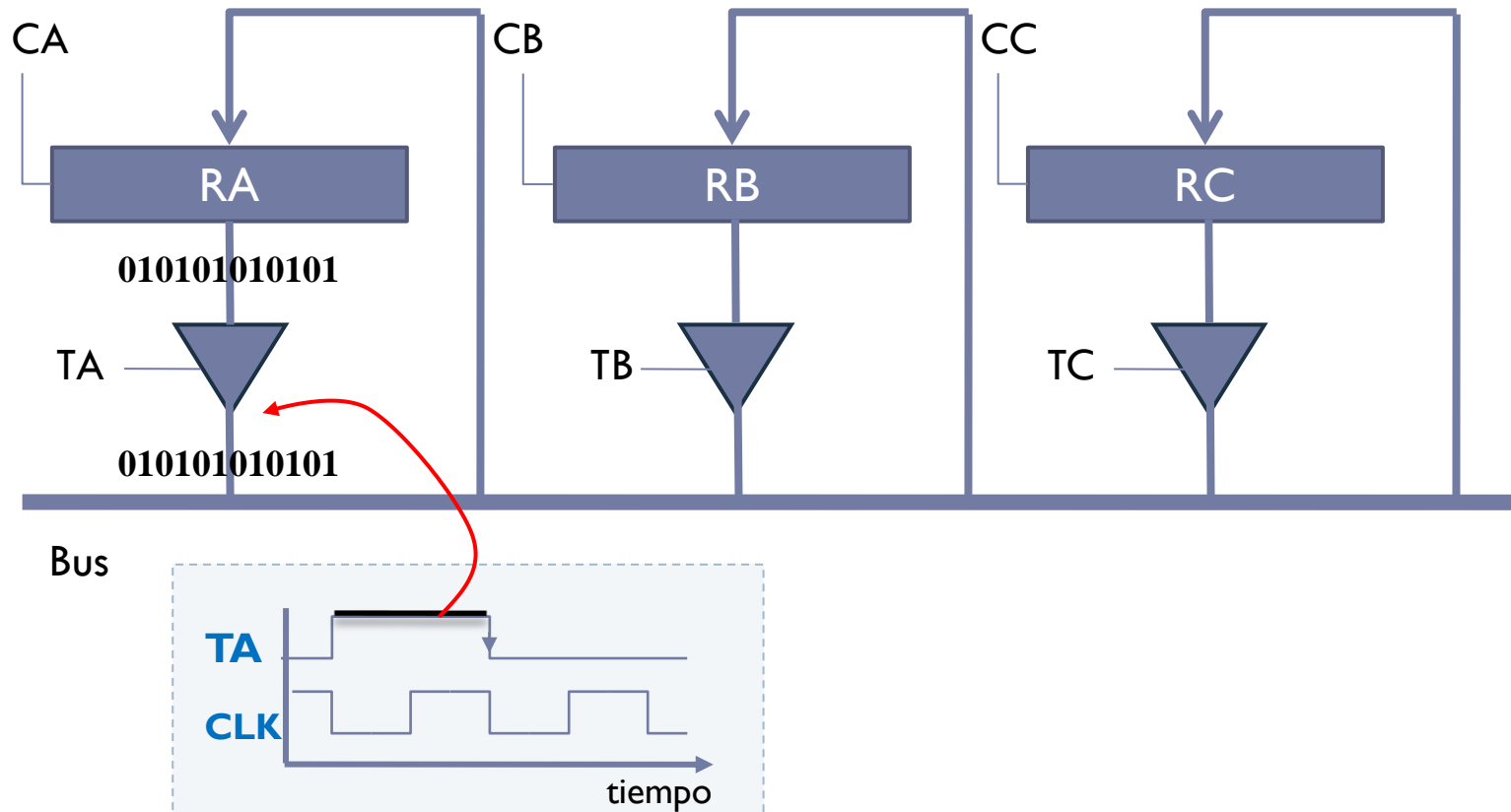
Acceso a un bus



Acceso a un bus

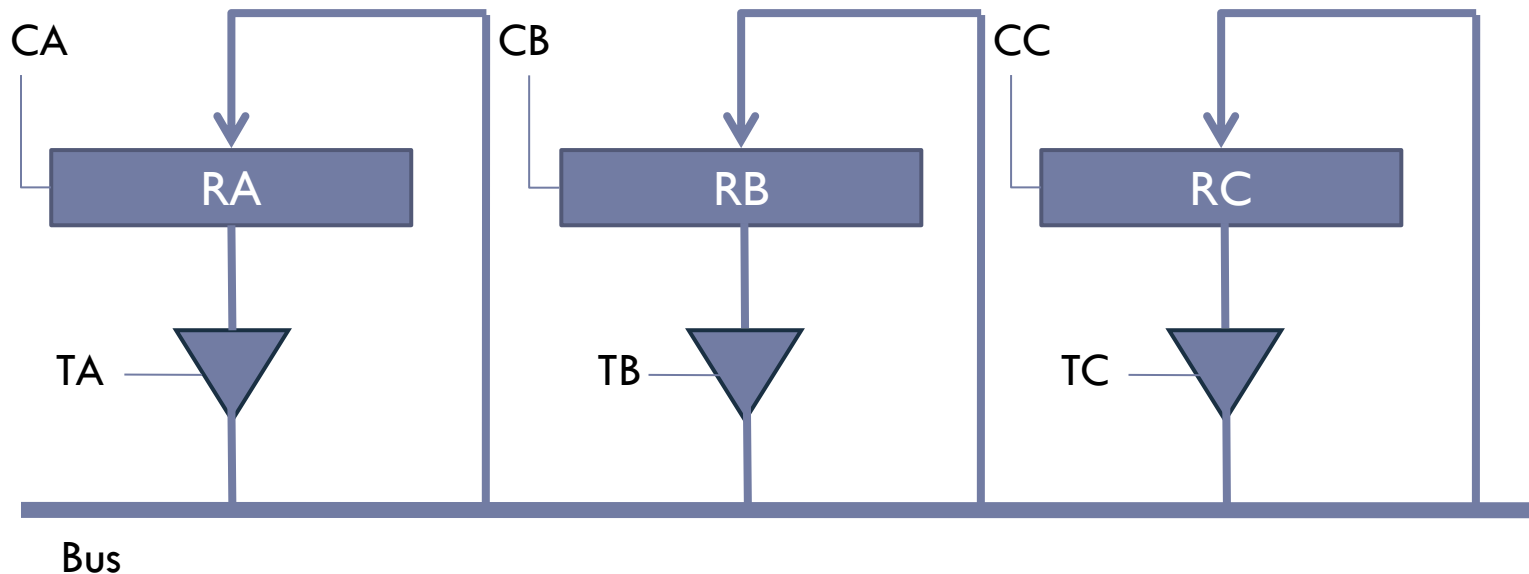


Acceso a un bus



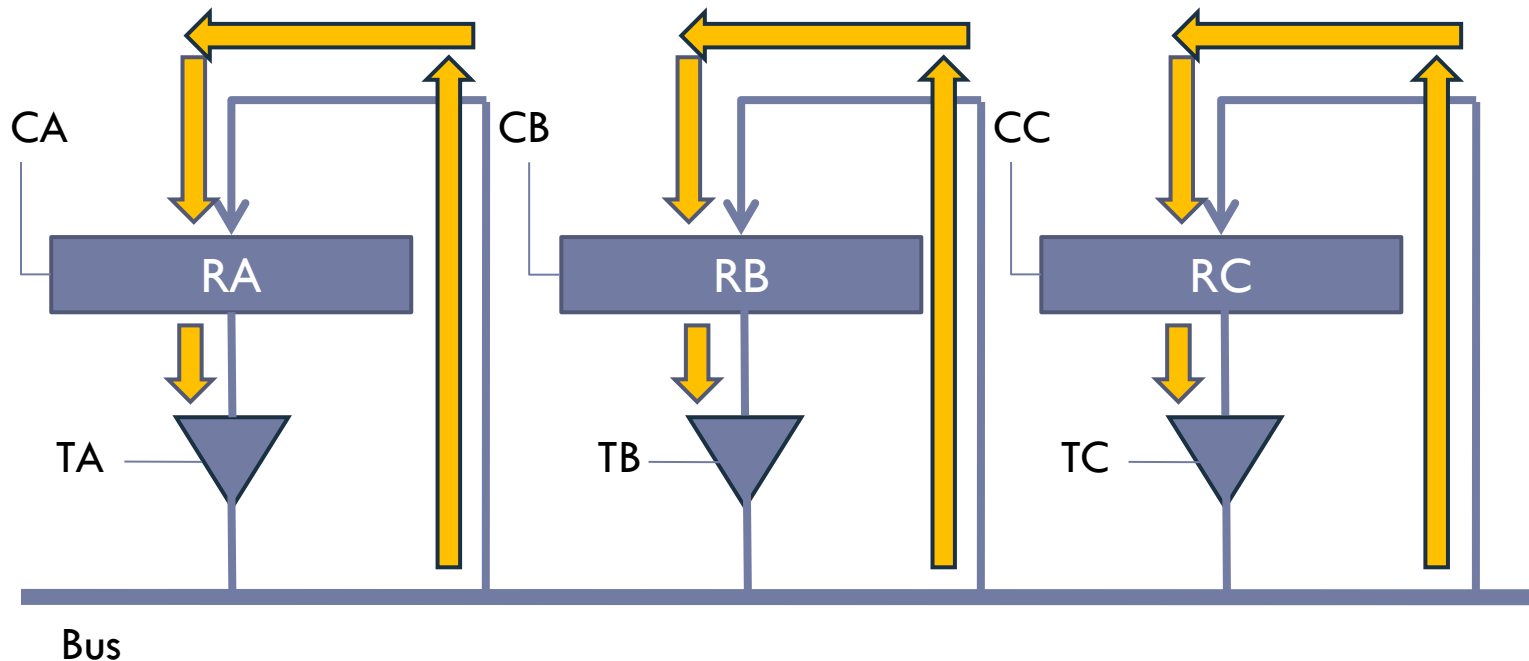
Ejemplo

- ▶ ¿Qué señales de control hay que activar para cargar el contenido de RA en RB?



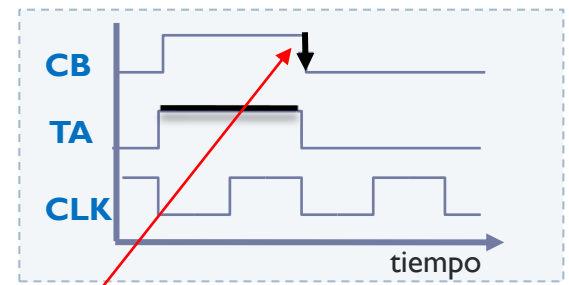
Ejemplo

- ▶ Camino de datos $RB \leftarrow RA$
- ▶ **Situación inicial con todas las señales desactivadas**



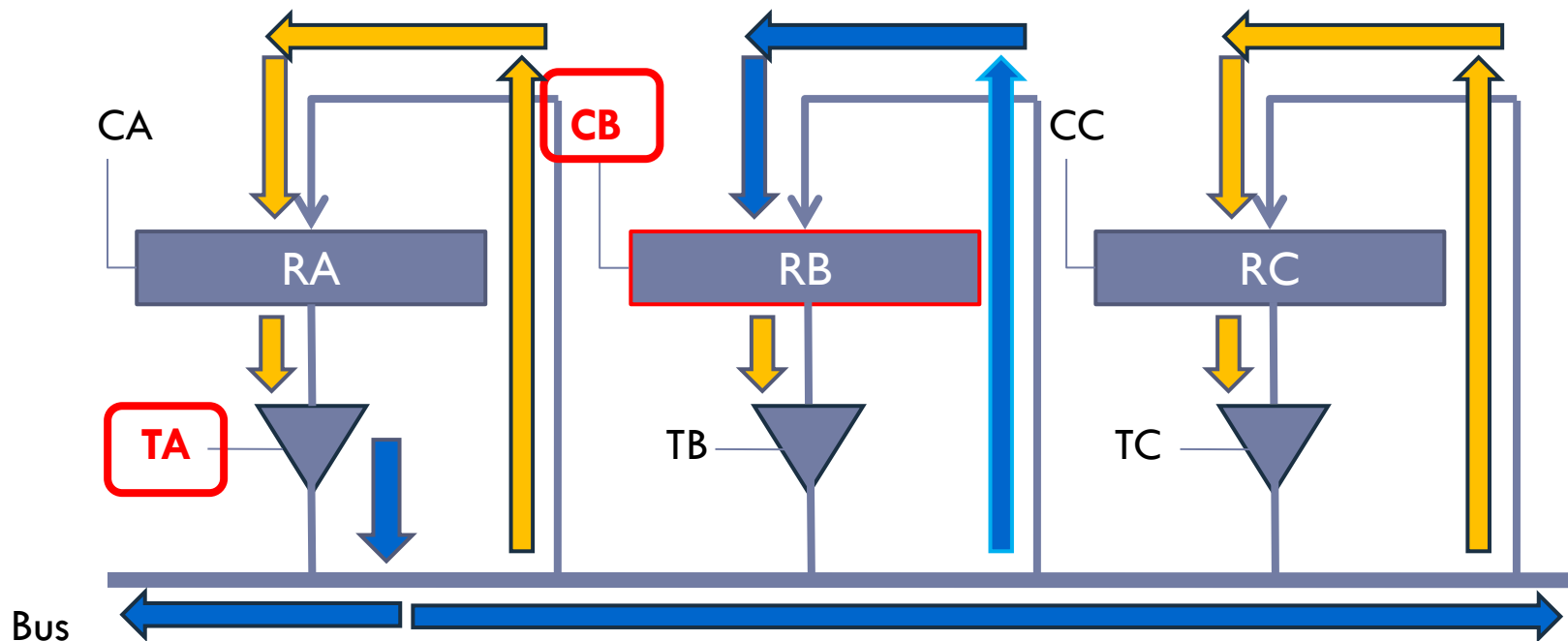
Ejemplo

- ▶ Camino de datos $RB \leftarrow RA$
- ▶ **La carga en RB se produce en el flanco de bajada**



IMPORTANTE

No se puede activar dos o más triestados al mismo bus al mismo tiempo



Lenguaje RT y Operaciones elementales

► Lenguaje RT

- Lenguaje de nivel de **transferencia** de **registros**.
- Especifica lo que ocurre en el computador mediante **operaciones elementales** entre registros.

► Operaciones elementales:

- Operaciones de **transferencia**
 - $MAR \leftarrow PC$
- Operaciones de **proceso**
 - $R1 \leftarrow R2 + R2$

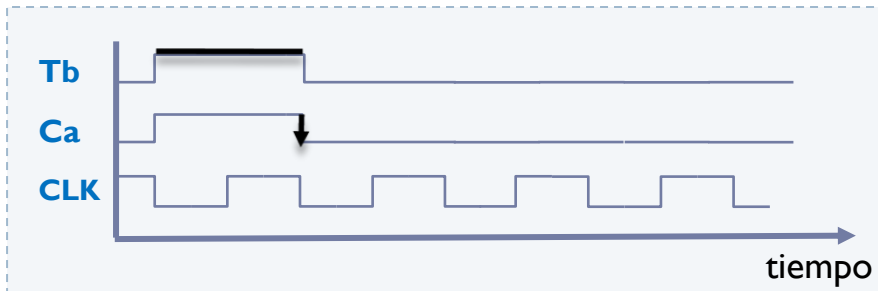
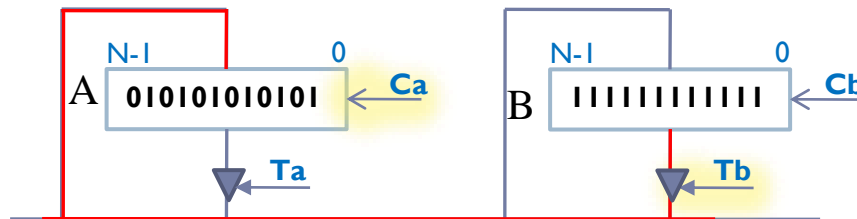


$Reg \leftarrow Reg$



$Reg \leftarrow \varphi(Reg, Reg)$

Ejemplo de operación elemental de transferencia



► Operación elemental de transferencia:

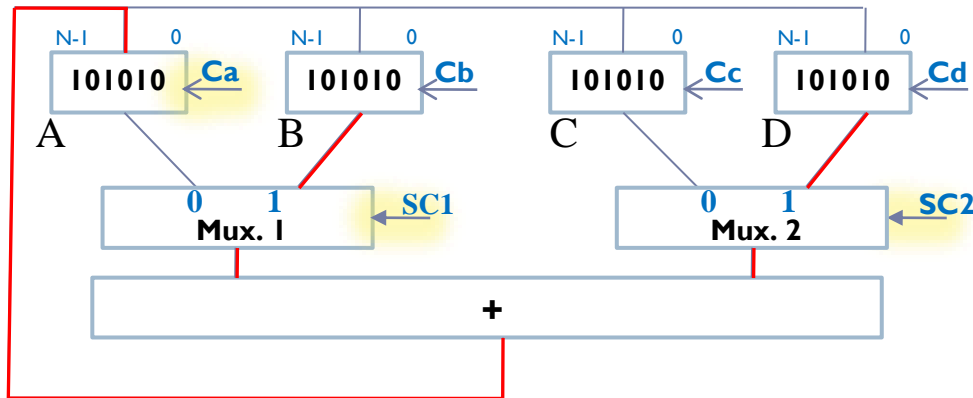
- Elemento de almacenamiento origen
- Elemento de almacenamiento destino
- Se establece un camino

xx: $A \leftarrow B$ [Tb, Ca]

► IMPORTANTE

- Establecer el camino entre origen y destino en un mismo ciclo
- En un mismo ciclo NO se puede:
 - atravesar un registro
 - llevar a un bus dos valores a la vez.

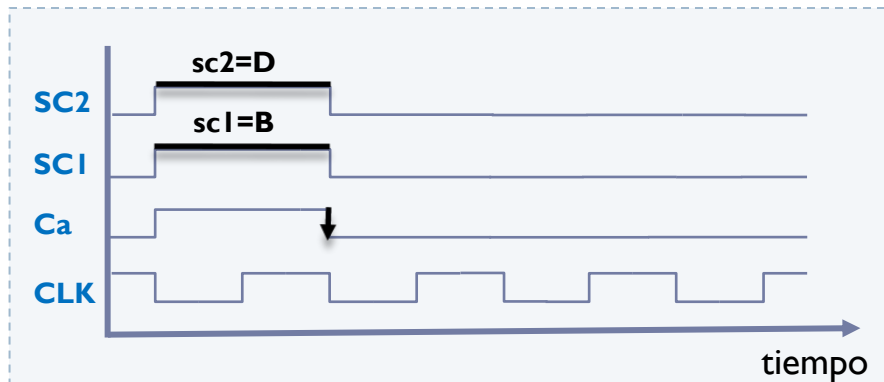
Ejemplo de operación elemental de procesamiento



► Operación elemental de procesamiento:

- Elemento(s) de origen
- Elemento destino
- Operación de transformación en el camino

yy: $A \leftarrow B + D$ [SC1=b, SC2=d, Ca]

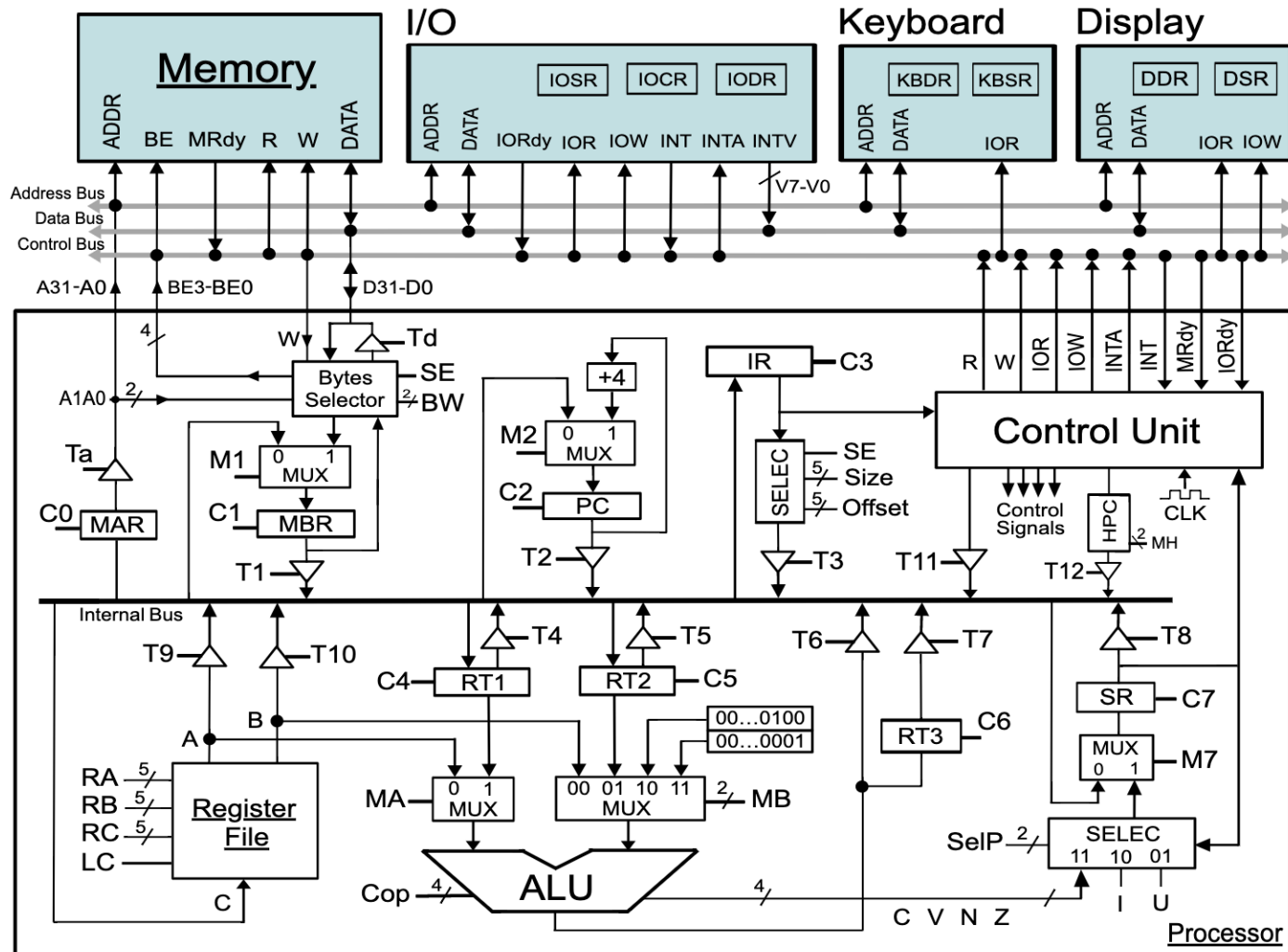


► IMPORTANTE

- Establecer el camino entre origen y destino en un mismo ciclo
- En un mismo ciclo NO se puede:
 - atravesar un registro
 - llevar a un bus dos valores a la vez.

Estructura de un computador elemental y simulador WepSIM

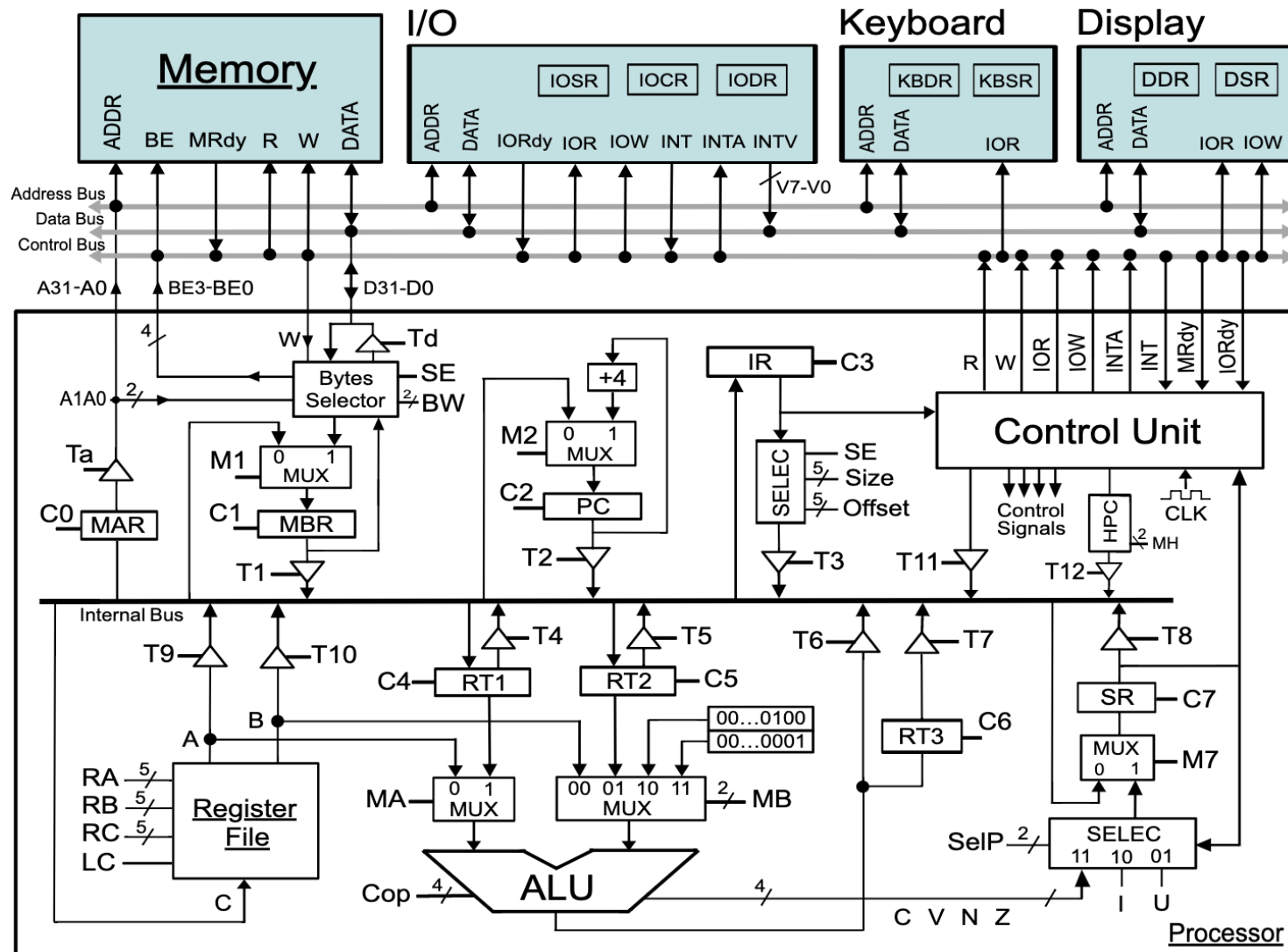
<https://wepsim.github.io/wepsim/>



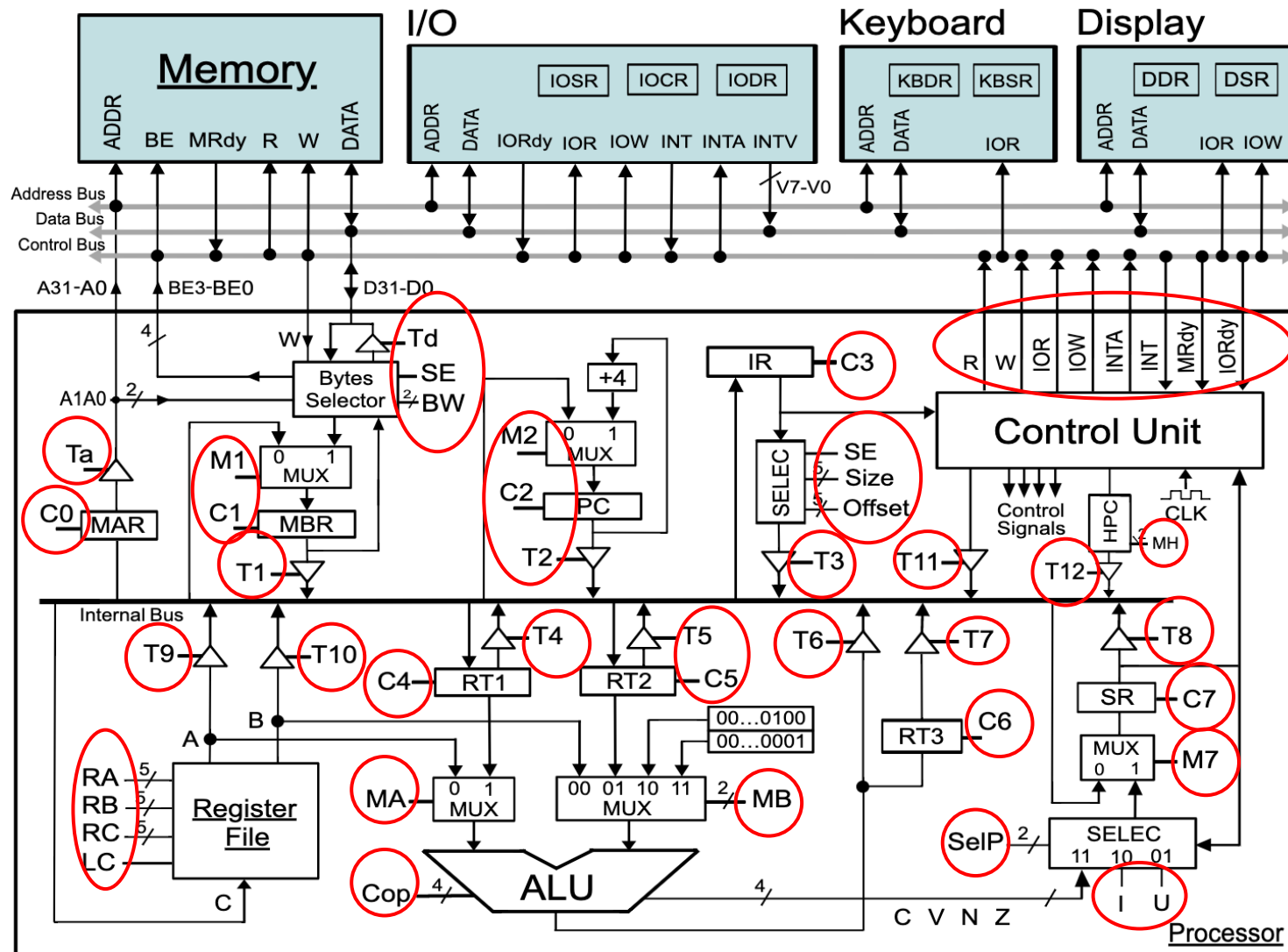
Principales características

- ▶ Características del computador elemental (EP)
 - ▶ Computador de 32 bits
 - ▶ La memoria principal:
 - ▶ Se direcciona a nivel de byte
 - ▶ Un ciclo para las operaciones de lectura y escritura
 - ▶ Diversos tipos de registros disponibles:
 - ▶ Banco de 32 registros visibles al programador (R0...R31)
 - Asumir como en el MIPS: R0 = 0 y SP = R29
 - ▶ Registros no visibles al programador (RT1, RT2 y RT3)
 - Posible uso para cálculos intermedios dentro de una instrucción
 - ▶ Registros de control (PC, IR, MAR, MBR) y registro de estado (SR)
 - MAR, MBR, PC, SR, IR
- ▶ Simulador WepSIM implementa el EP:
 - ▶ <https://wepsim.github.io/wepsim/>

Estructura de un computador elemental



Señales de control



Señales de control

- ▶ Señales de acceso a memoria
- ▶ Señales de carga en registros
- ▶ Señales de control de las puertas triestado
- ▶ Señales de selección de los MUX
- ▶ Señales de control del banco de registros
- ▶ Otras señales de selección

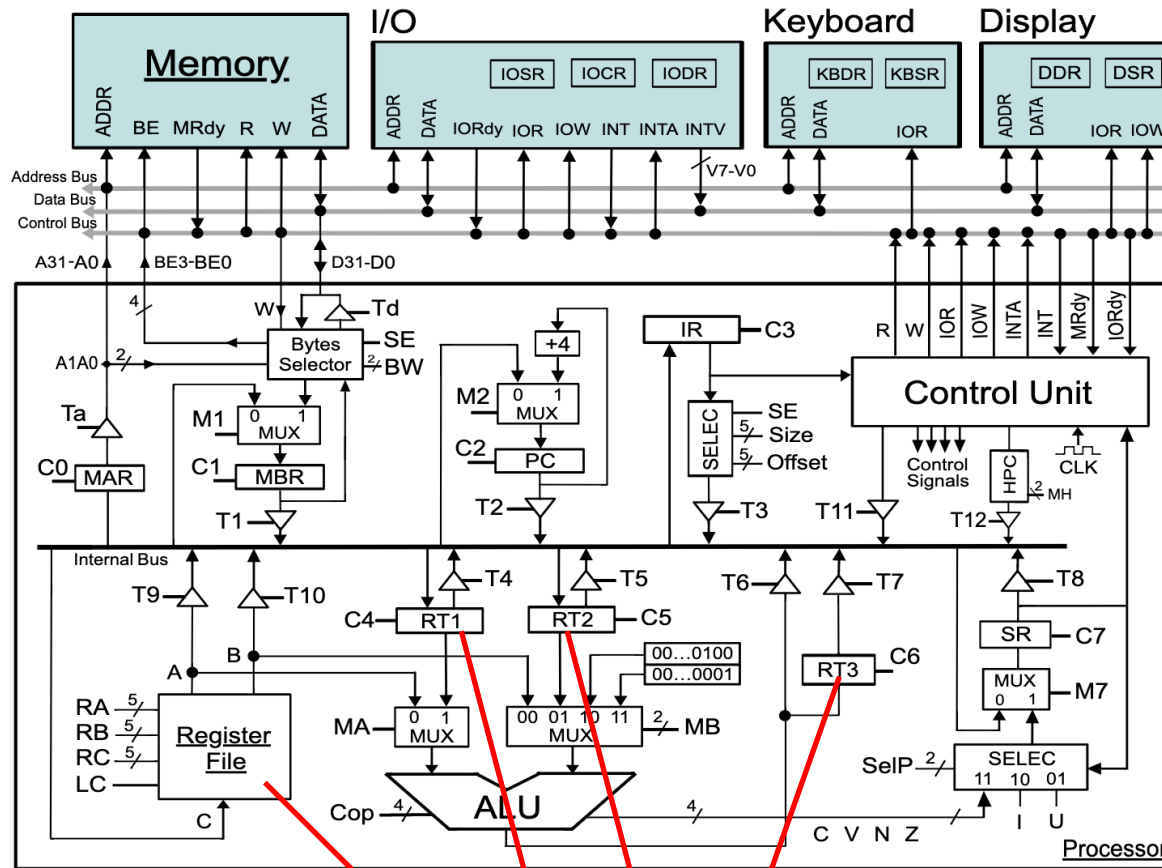
Nomenclatura general:

- Mx: Selección en multiplexor
- Tx: Señal de activación triestado
- Cx: Señal de carga de registro
- Ry: Selección de registros del banco de registros

Registros

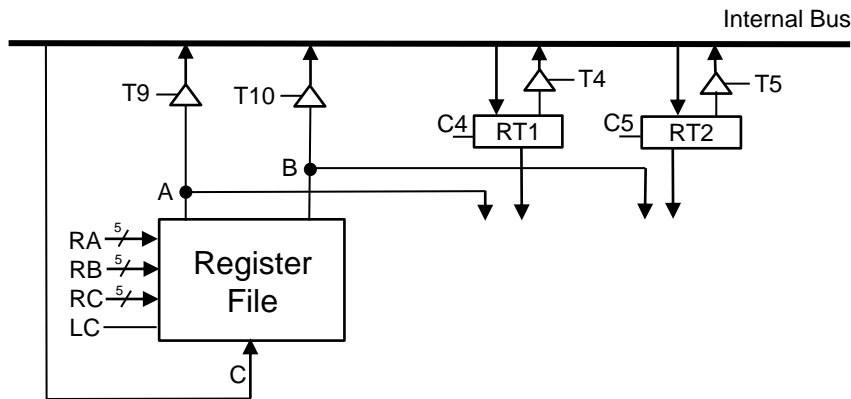
- ▶ **Registros visibles al programador**
 - ▶ Registros del banco de registros (Ej. MIPS: \$t0, \$t1, etc.)
- ▶ **Registros de control y estado:**
 - ▶ PC: contador de programa o *program counter*
 - ▶ IR: registro de instrucción o *instruction register*
 - ▶ SP: puntero de pila (en el banco de registros)
 - ▶ MAR: registro de direcciones de memoria o *memory address register*
 - ▶ MBR: registro de datos de memoria o *memory buffer register*
 - ▶ SR: registro de estado o *status register*
- ▶ **Registros no visibles al usuario:**
 - ▶ RT1, RT2 y RT3: registros temporales internos de la CPU

Estructura de un computador elemental



Banco de registros y
registros auxiliares (RT1, RT2 y RT3)

Señales de control



Nomenclatura:

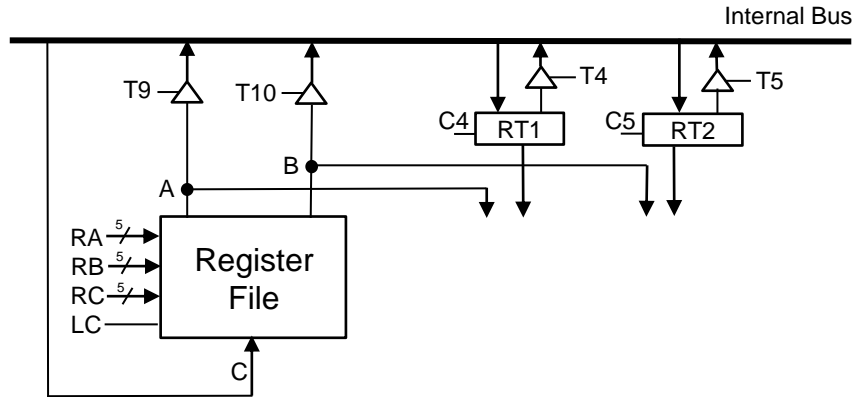
- Ry -> Identificador de registro para el punto y
- Mx -> Selección en multiplexor
- Tx -> Señal de activación triestado
- Cx -> Señal de carga de registro

► Banco de registros y registros RT1 y RT2

- RA – salida de registro RA por A
- RB – salida de registro RB por B
- RC – entrada por C al registro RC
- LC – activa la escritura para RC
- T9 – copia de A al bus interno
- T10 – copia de B al bus interno
- C4 – del bus interno al RT1
- T4 – salida de RT1 al bus interno
- C5 – del bus interno al RT2
- T5 – salida de RT2 al bus interno

Ejemplo

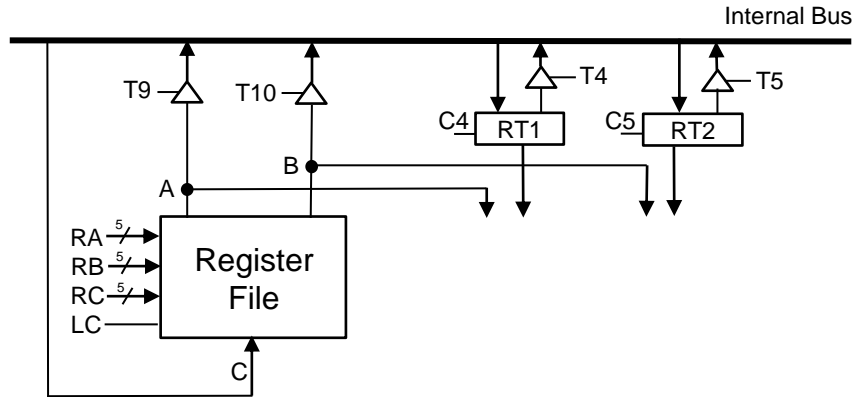
operaciones elementales en registros



► **SWAP R1 R2**

Ejemplo

operaciones elementales en registros

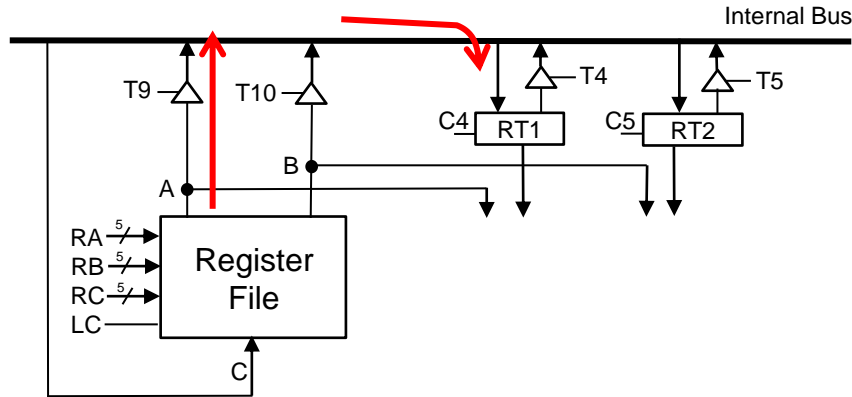


► SWAP R1 R2

O. Elemental	Señales

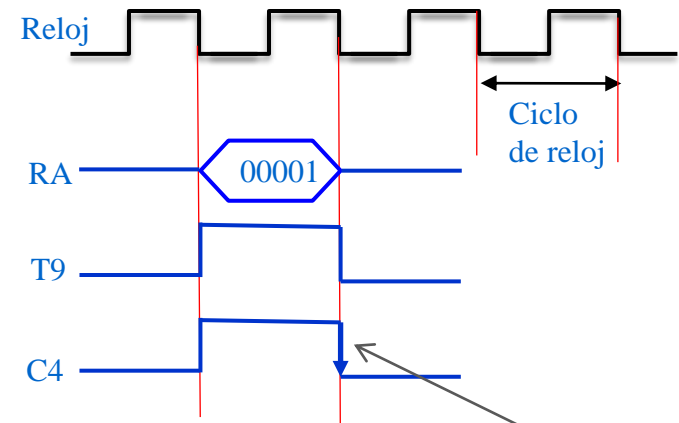
Ejemplo

operaciones elementales en registros



► SWAP R1 R2

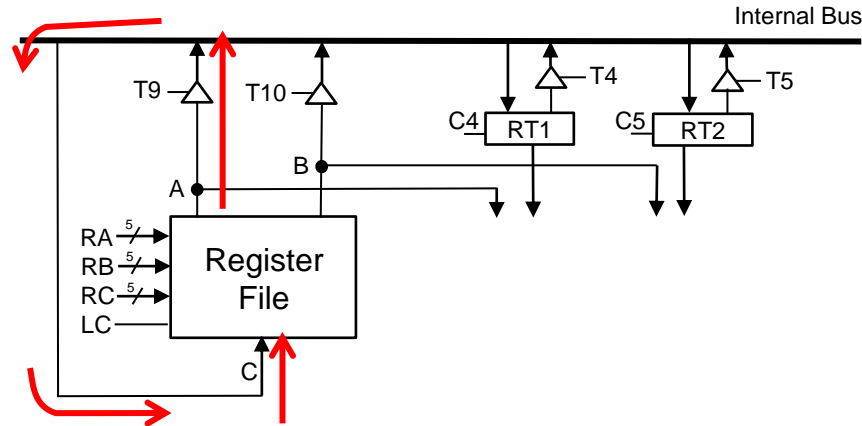
O. Elemental	Señales
$RT1 \leftarrow R1$	$RA=00001, T9, C4$



La carga del dato se realiza en RT1 en el flanco de bajada. Estará disponible en RT1 durante el siguiente ciclo.

Ejemplo

operaciones elementales en registros

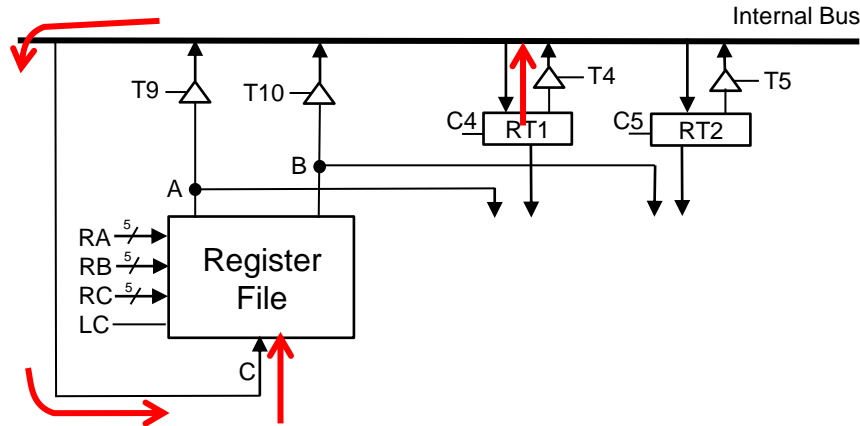


► **SWAP R1 R2**

O. Elemental	Señales
$RT1 \leftarrow R1$	RA=00001, T9, C4
$R1 \leftarrow R2$	RA=2 (00010), T9, RC=1, LC

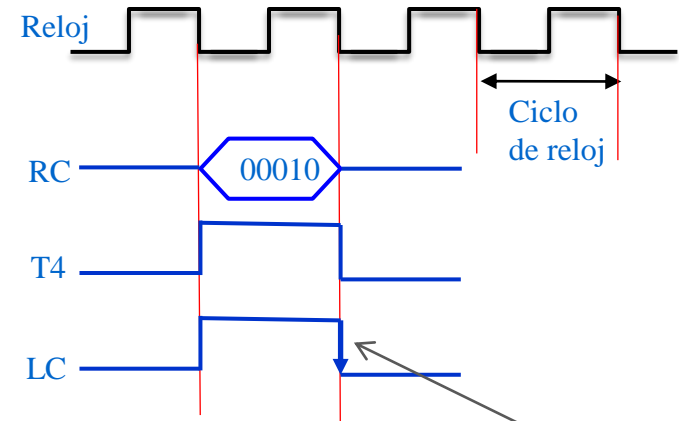
Ejemplo

operaciones elementales en registros



► SWAP R1 R2

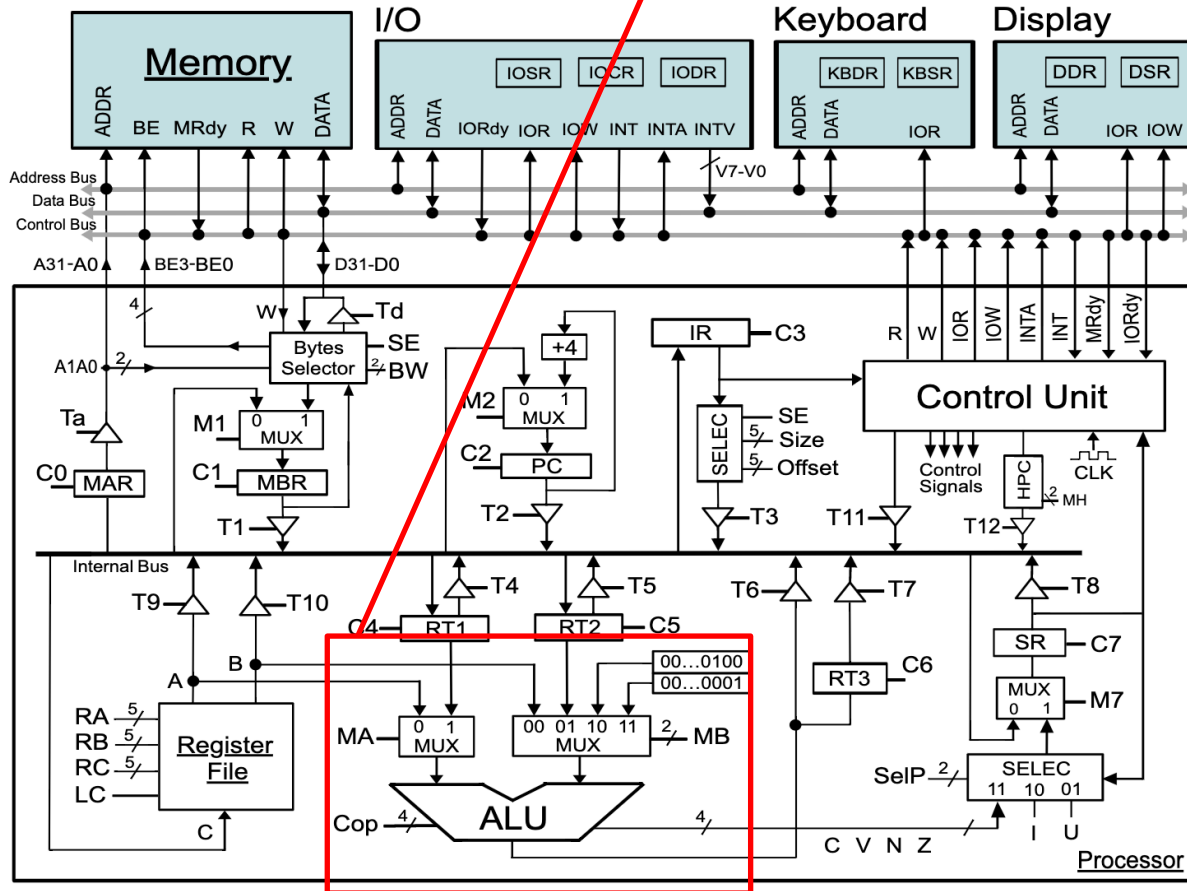
O. Elemental	Señales
$RT1 \leftarrow R1$	RA=00001, T9, C4
$R1 \leftarrow R2$	RA=2 (00010), T9, RC=1, LC
$R2 \leftarrow RT1$	T4, RC=2 (00010), LC



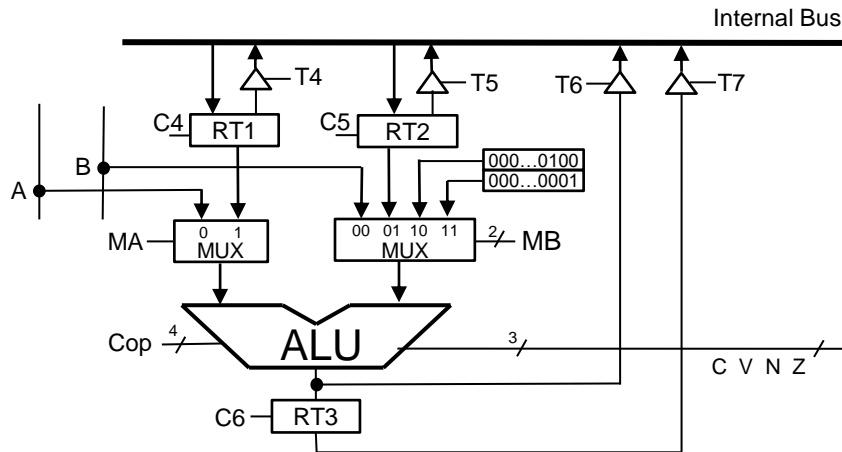
La carga del dato se realiza en R2 en el flanco de bajada. Estará disponible en R2 durante el siguiente ciclo.

Estructura de un computador elemental

Unidad Aritmético-Lógica (ALU)



Señales de control

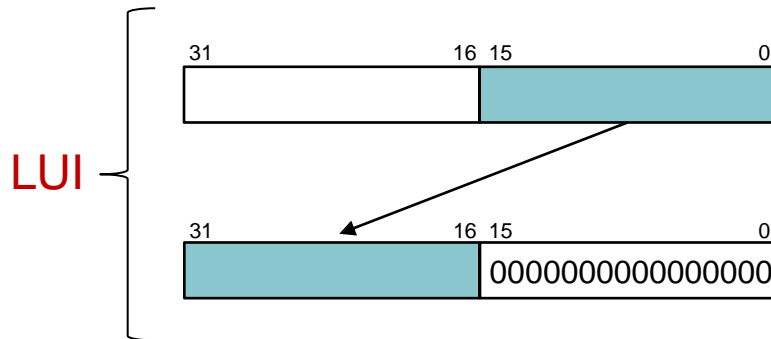
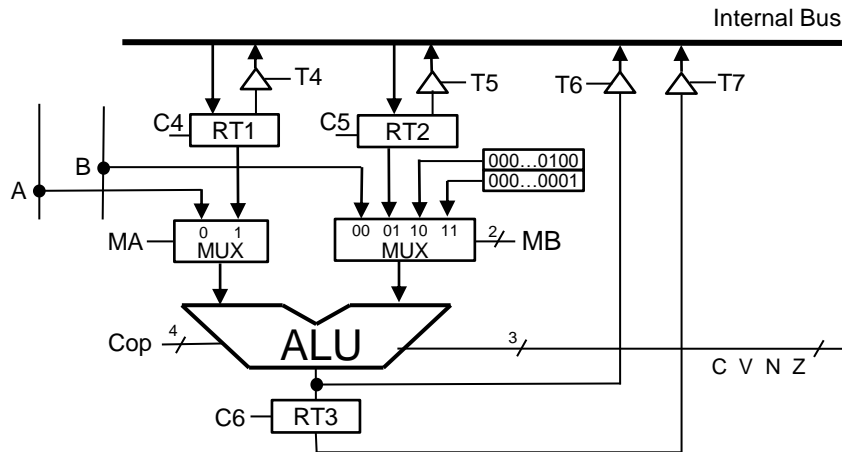


▶ ALU

- ▶ MA – selección de operando A
- ▶ MB – selección de operando B
- ▶ Cop – código de operación

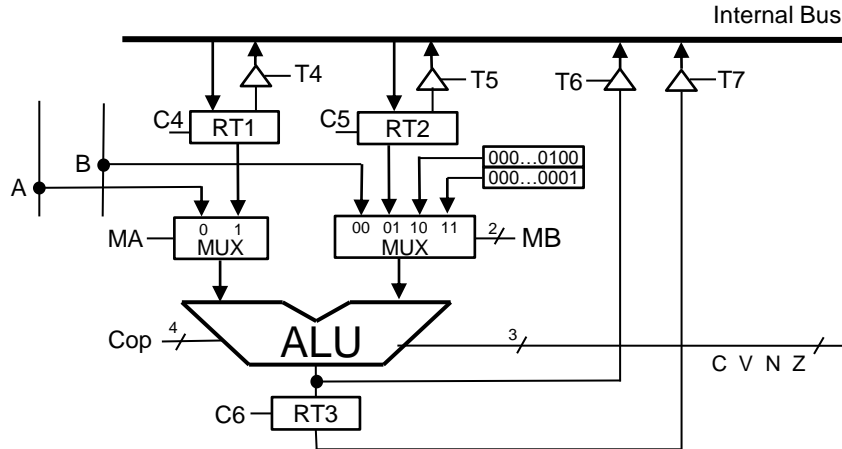
Cop (Cop ₃ -Cop ₀)	Operación
0000	NOP
0001	A and B
0010	A or B
0011	not (A)
0100	A xor B
0101	Shift Right Logical (A) B= number of bits to shift
0110	Shift Right Arithmetic (A) B= number of bits to shift
0111	Shift left (A) B= number of bits to shift
1000	Rotate Right (A) B= number of bits to rotate
1001	Rotate Left (A) B= number of bits to rotate
1010	A + B
1011	A - B
1100	A * B (with overflow)
1101	A / B (integer division)
1110	A % B (integer division)
1111	LUI (A)

Señales de control



Cop (Cop ₃ -Cop ₀)	Operación
0000	NOP
0001	A and B
0010	A or B
0011	not (A)
0100	A xor B
0101	Shift Right Logical (A) B= number of bits to shift
0110	Shift Right Arithmetic (A) B= number of bits to shift
0111	Shift left (A) B= number of bits to shift
1000	Rotate Right (A) B= number of bits to rotate
1001	Rotate Left (A) B= number of bits to rotate
1010	A + B
1011	A - B
1100	A * B (with overflow)
1101	A / B (integer division)
1110	A % B (integer division)
1111	LUI (A)

Señales de control

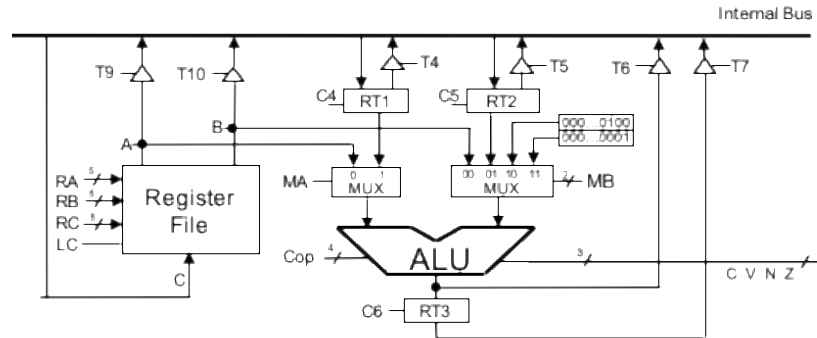


Resultado	C	V	N	Z
Resultado positivo (0 se considera +)	0	0	0	0
Resultado == 0	0	0	0	1
Resultado negativo	0	0	1	0
Desbordamiento de la operación	0	1	0	0
División por cero	0	1	0	1
Acarreo en el bit 32	1	0	0	0

Cop (Cop ₃ -Cop ₀)	Operación
0000	NOP
0001	A and B
0010	A or B
0011	not (A)
0100	A xor B
0101	Shift Right Logical (A) B= number of bits to shift
0110	Shift Right Arithmetic (A) B= number of bits to shift
0111	Shift left (A) B= number of bits to shift
1000	Rotate Right (A) B= number of bits to rotate
1001	Rotate Left (A) B= number of bits to rotate
1010	A + B
1011	A - B
1100	A * B (with overflow)
1101	A / B (integer division)
1110	A % B (integer division)
1111	LUI (A)

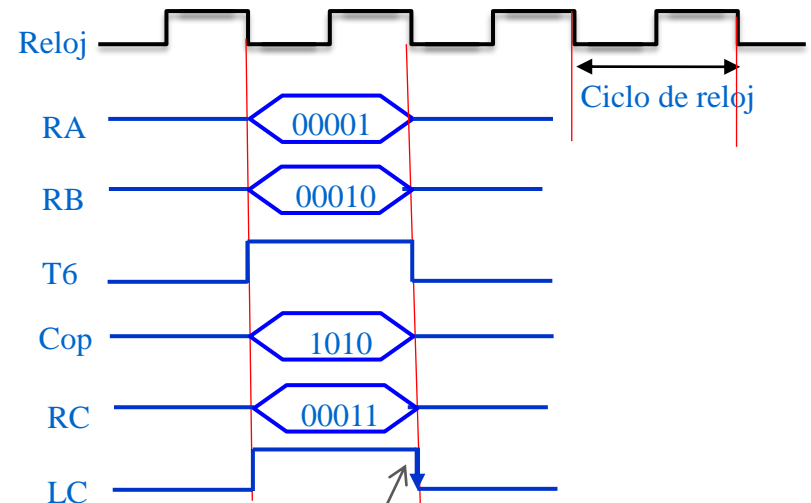
Ejemplo

operaciones elementales en ALU



► ADD R3 R1 R2

O. Elemental	Señales
$R3 \leftarrow R1 + R2$	RA=R1, RB=R2, Cop=+, T6, RC=R3, LC=1

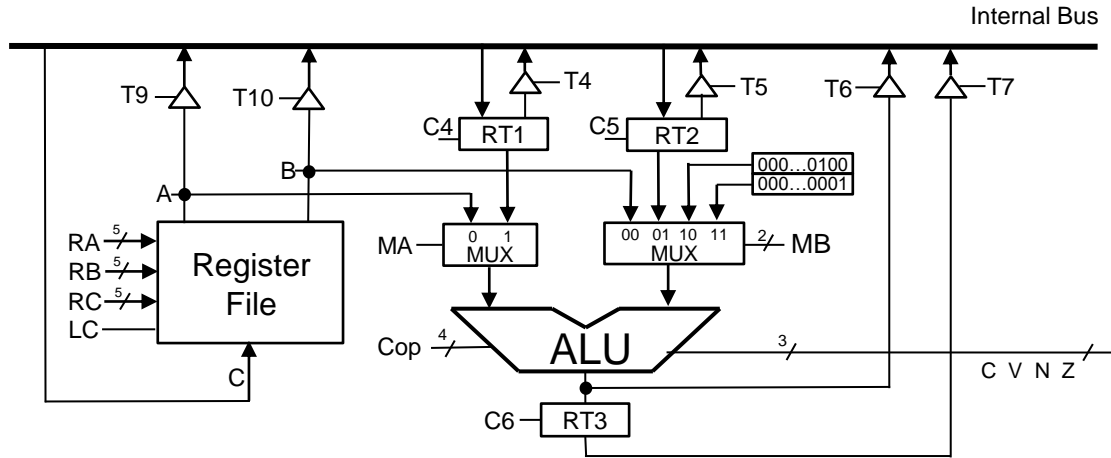


Resto de señales a 0.

La carga se realiza en R3 en el flanco de bajada.

El dato está disponible en el registro R3 durante el siguiente ciclo

operaciones elementales en ALU



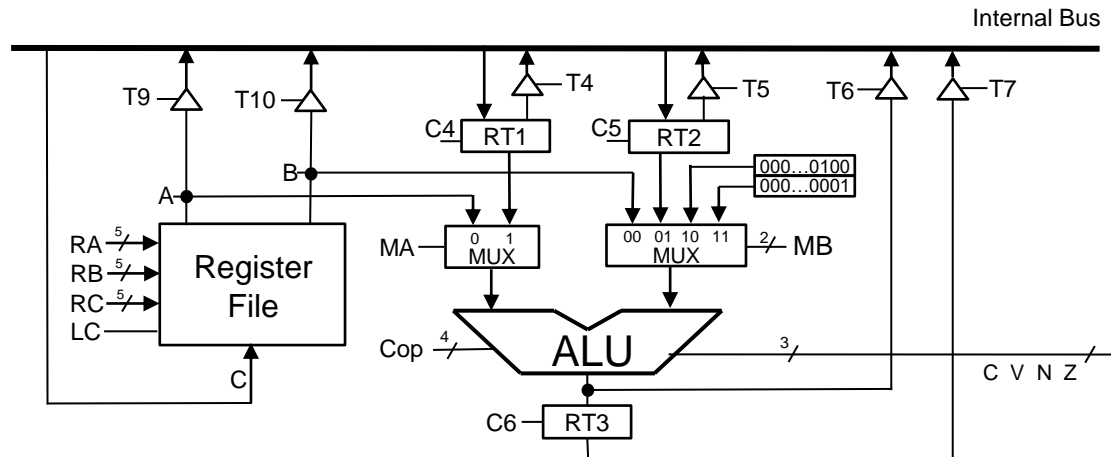
► **SWAP R1 R2**

► **SWAP R1, R2 sin R_{tmp}**

O. Elemental	Señales
$RT1 \leftarrow R1$	RA=1, T9, C4
$R1 \leftarrow R2$	RA=2, T9, RC=1, LC
$R2 \leftarrow RT1$	T4, RC=2, LC

Ejemplo

operaciones elementales en ALU



► SWAP R1 R2

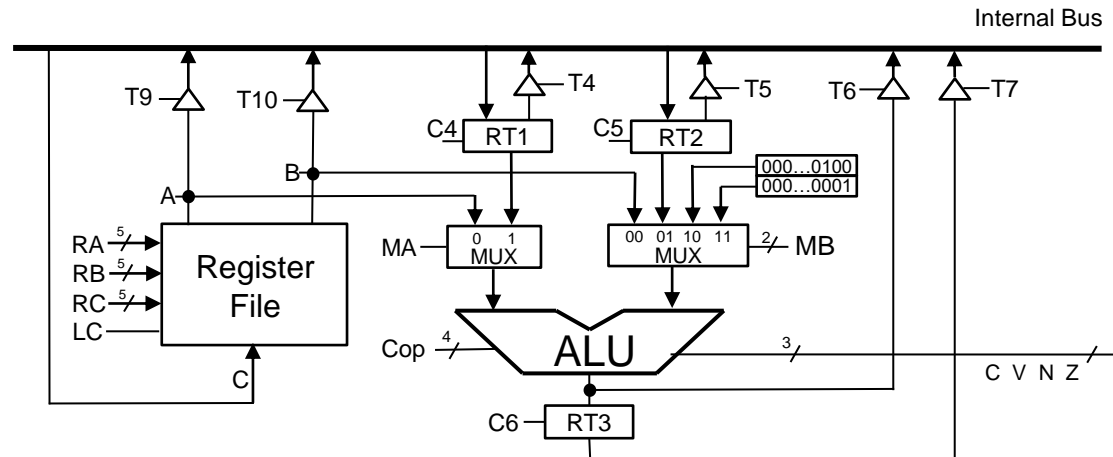
O. Elemental	Señales
$RT1 \leftarrow R1$	RA=1, T9, C4
$R1 \leftarrow R2$	RA=2, T9, RC=1, LC
$R2 \leftarrow RT1$	T4, RC=2, LC

► SWAP R1, R2 sin R_{tmp}

O. Elemental	
$R1 \leftarrow R1 \wedge R2$	$R1 \leftarrow (R1 \wedge R2)$
$R2 \leftarrow R1 \wedge R2$	$R2 \leftarrow (R1 \wedge R2) \wedge R2$
$R1 \leftarrow R1 \wedge R2$	$R1 \leftarrow (R1 \wedge R2) \wedge R1$

Ejemplo

operaciones elementales en ALU



► SWAP R1 R2

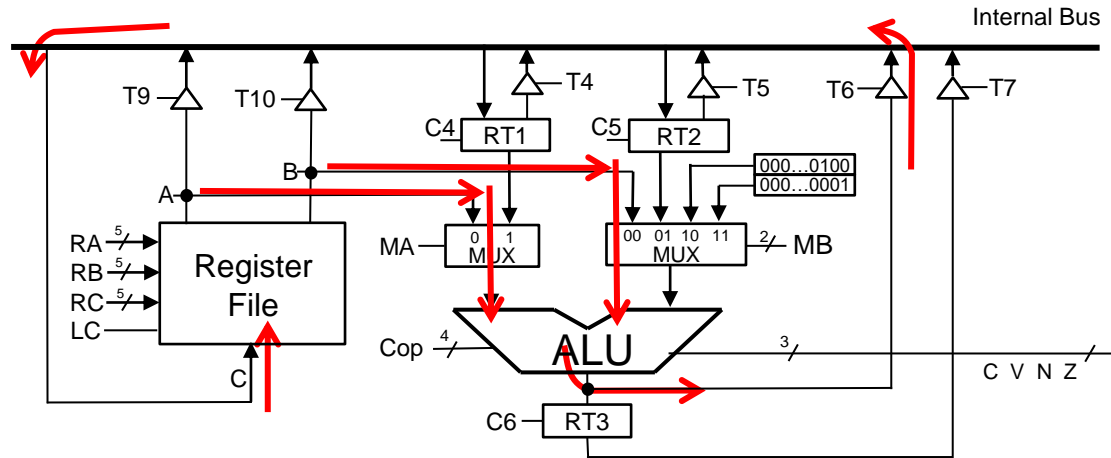
O. Elemental	Señales
$RT1 \leftarrow R1$	RA=1, T9, C4
$R1 \leftarrow R2$	RA=2, T9, RC=1, LC
$R2 \leftarrow RT1$	T4, RC=2, LC

► SWAP R1, R2 sin R_{tmp}

O. Elemental	Señales
$R1 \leftarrow R1 \wedge R2$	RA=1, RB=2, Cop= \wedge , T6, RC=1, LC
$R2 \leftarrow R1 \wedge R2$	RA=1, RB=2, Cop= \wedge , T6, RC=2, LC
$R1 \leftarrow R1 \wedge R2$	RA=1, RB=2, Cop= \wedge , T6, RC=1, LC

Ejemplo

operaciones elementales en ALU



► SWAP R1 R2

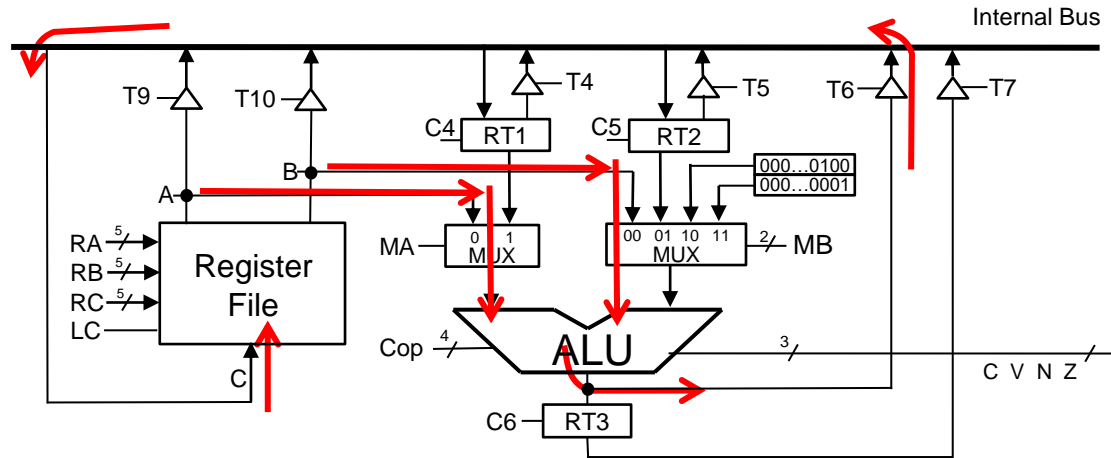
O. Elemental	Señales
$RT1 \leftarrow R1$	RA=1, T9, C4
$R1 \leftarrow R2$	RA=2, T9, RC=1, LC
$R2 \leftarrow RT1$	T4, RC=2, LC

► SWAP R1, R2 sin R_{tmp}

O. Elemental	Señales
$R1 \leftarrow R1 \wedge R2$	RA=1, RB=2, Cop=^, T6, RC=1, LC
$R2 \leftarrow R1 \wedge R2$	RA=1, RB=2, Cop=^, T6, RC=2, LC
$R1 \leftarrow R1 \wedge R2$	RA=1, RB=2, Cop=^, T6, RC=1, LC

Ejemplo

operaciones elementales en ALU



► SWAP R1 R2

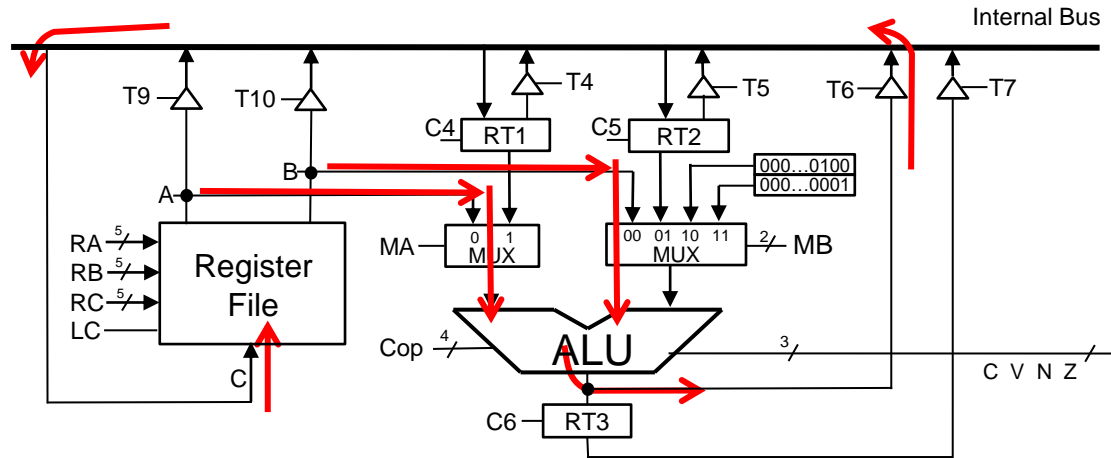
O. Elemental	Señales
$RT1 \leftarrow R1$	RA=1, T9, C4
$R1 \leftarrow R2$	RA=2, T9, RC=1, LC
$R2 \leftarrow RT1$	T4, RC=2, LC

► SWAP R1, R2 sin R_{tmp}

O. Elemental	Señales
$R1 \leftarrow R1 \wedge R2$	RA=1, RB=2, Cop=^, T6, RC=1, LC
$R2 \leftarrow R1 \wedge R2$	RA=1, RB=2, Cop=^, T6, RC=2, LC
$R1 \leftarrow R1 \wedge R2$	RA=1, RB=2, Cop=^, T6, RC=1, LC

Ejemplo

operaciones elementales en ALU



► SWAP R1 R2

O. Elemental	Señales
$RT1 \leftarrow R1$	RA=1, T9, C4
$R1 \leftarrow R2$	RA=2, T9, RC=1, LC
$R2 \leftarrow RT1$	T4, RC=2, LC

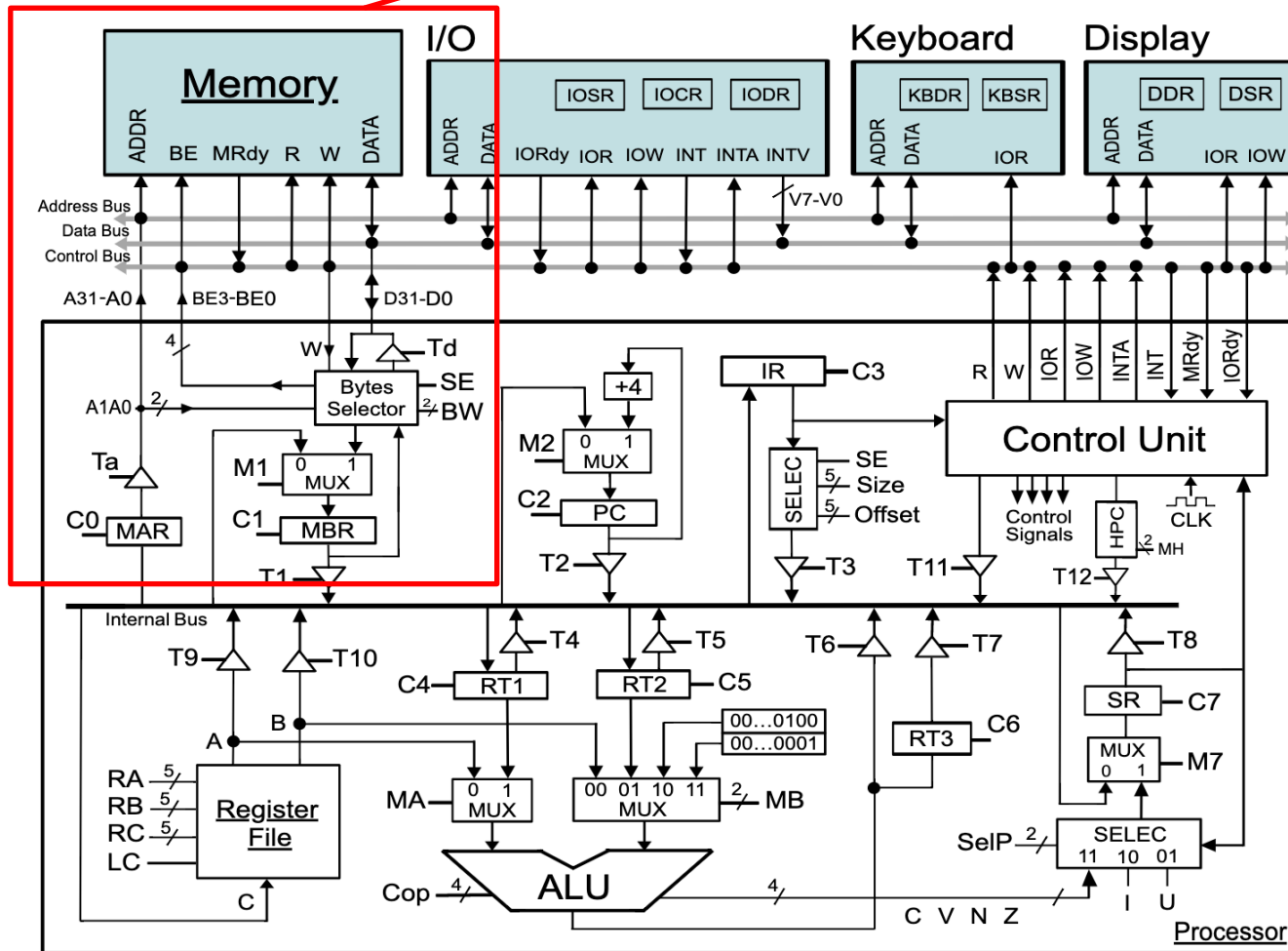
► SWAP R1, R2 sin R_{tmp}

O. Elemental	Señales
$R1 \leftarrow R1 \wedge R2$	RA=1, RB=2, Cop= \wedge , T6, RC=1, LC
$R2 \leftarrow R1 \wedge R2$	RA=1, RB=2, Cop= \wedge , T6, RC=2, LC
$R1 \leftarrow R1 \wedge R2$	RA=1, RB=2, Cop= \wedge , T6, RC=1, LC

Estructura de un computador elemental

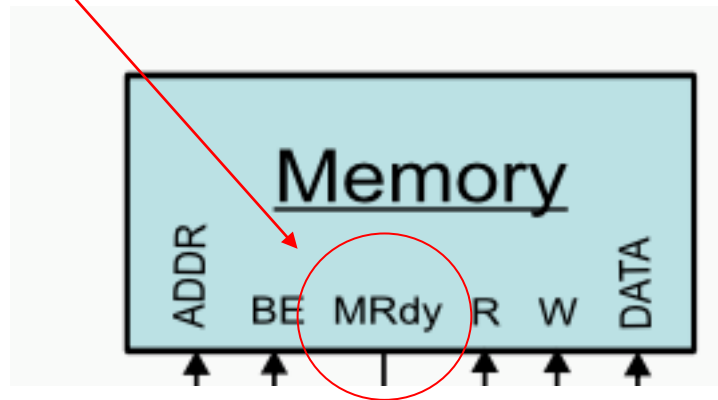
Memoria principal,

registro de direcciones y de datos



Acceso a Memoria

- ▶ Síncrono: la memoria requiere un número determinado de ciclos
- ▶ Asíncrono: la memoria indica cuándo finaliza la operación



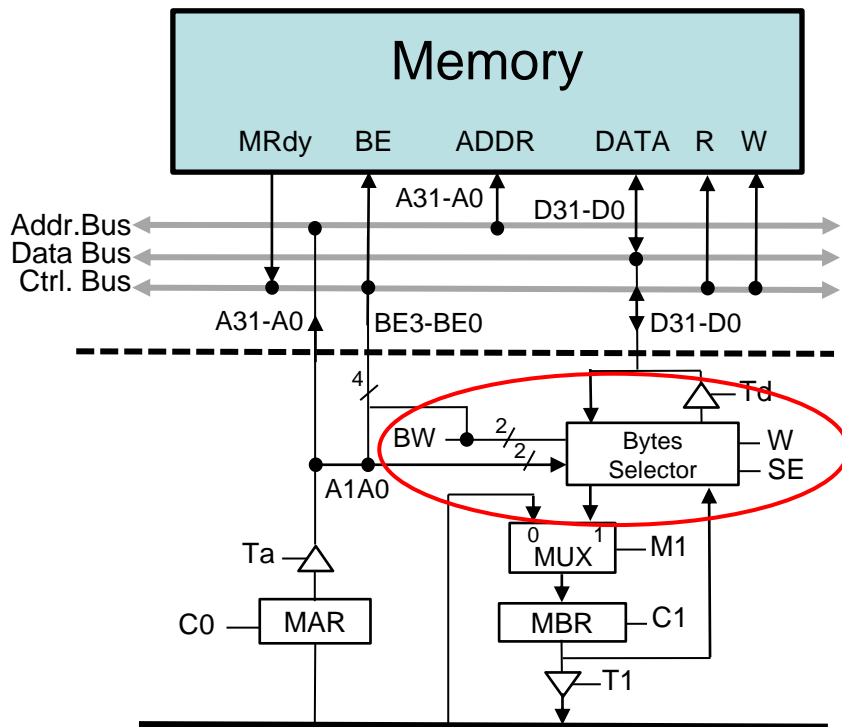
Señales BE (Byte-Enable) para lectura

Bytes en memoria				Selección de bytes				Salida al BUS			
D31-D24	D23-D16	D15-D8	D7-D0	BE3	BE2	BE1	BE0	D31-D24	D23-D16	D15-D8	D7-D0
Byte 3	Byte 2	Byte 1	Byte 0	0	0	0	0	---	---	---	Byte 0
Byte 3	Byte 2	Byte 1	Byte 0	0	0	0	1	---	---	Byte 1	---
Byte 3	Byte 2	Byte 1	Byte 0	0	0	1	0	--	Byte 2	---	---
Byte 3	Byte 2	Byte 1	Byte 0	0	0	1	1	Byte 3	---	---	---
Byte 3	Byte 2	Byte 1	Byte 0	0	1	0	X	---	---	Byte 1	Byte 0
Byte 3	Byte 2	Byte 1	Byte 0	0	1	1	X	Byte 3	Byte 2	---	---
Byte 3	Byte 2	Byte 1	Byte 0	1	1	X	X	Byte 3	Byte 2	Byte 1	Byte 0

Señales BE (Byte-Enable) para escritura

Dato en el bus				Selección de bytes				Bytes escritos en memoria			
D31-D24	D23-D16	D15-D8	D7-D0	BE3	BE2	BE1	BE0	D31-D24	D23-D16	D15-D8	D7-D0
Byte 3	Byte 2	Byte 1	Byte 0	0	0	0	0	---	---	---	Byte 0
Byte 3	Byte 2	Byte 1	Byte 0	0	0	0	1	---	---	Byte 1	---
Byte 3	Byte 2	Byte 1	Byte 0	0	0	1	0	--	Byte 2	---	---
Byte 3	Byte 2	Byte 1	Byte 0	0	0	1	1	Byte 3	---	---	---
Byte 3	Byte 2	Byte 1	Byte 0	0	1	0	X	---	---	Byte 1	Byte 0
Byte 3	Byte 2	Byte 1	Byte 0	0	1	1	X	Byte 3	Byte 2	---	---
Byte 3	Byte 2	Byte 1	Byte 0	1	1	X	X	Byte 3	Byte 2	Byte 1	Byte 0

Tamaño de acceso a memoria



- ▶ Bytes Selector: selecciona qué bytes se almacenan en MBR en lectura y se vuelcan al bus en escritura
- ▶ Acceso a bytes: BW=0
- ▶ Acceso a media palabra: BW=01
- ▶ Acceso a palabra: BW = 11
- ▶ SE: extensión de signo
 - ▶ 0: no extiende el signo en accesos más pequeños de una palabra
 - ▶ 1: extiende el signo en accesos más pequeños de una palabra

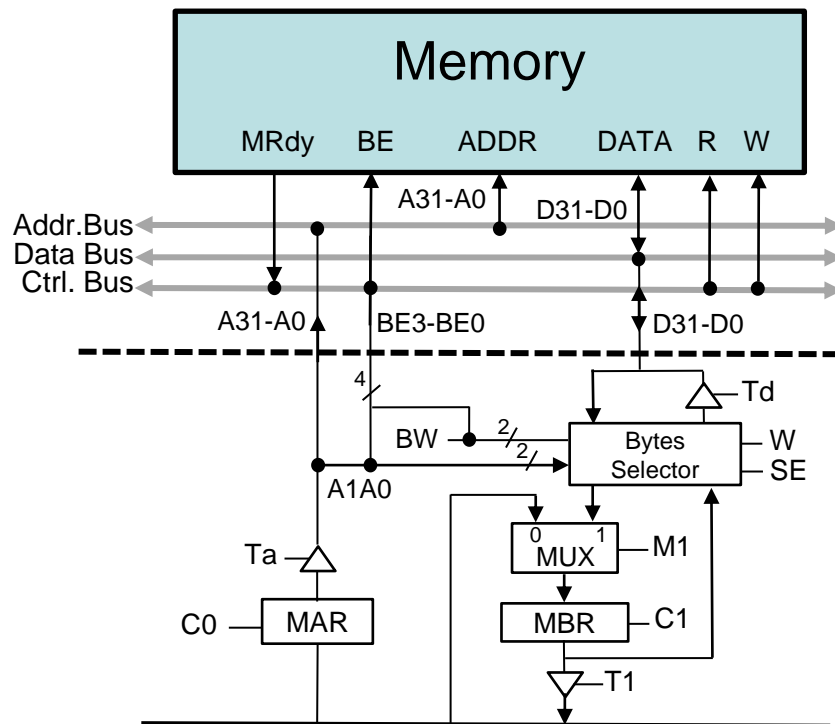
Nomenclatura:

- MAR -> registro de direcciones
- MBR -> registro de datos

Ejemplo

operaciones elementales para usar la memoria

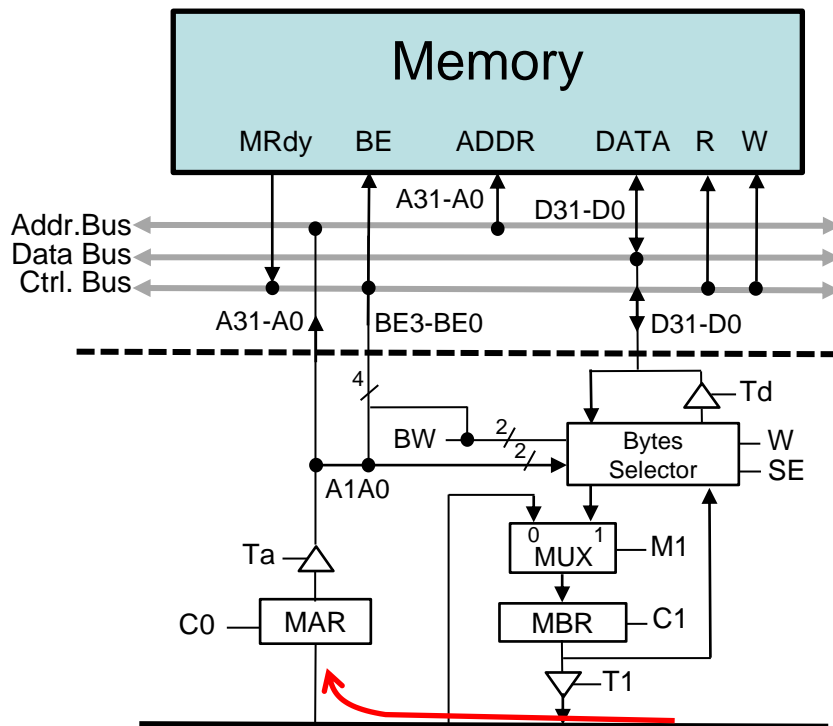
► Lectura



Ejemplo

Acceso a memoria síncrona de 1 ciclo

► Lectura

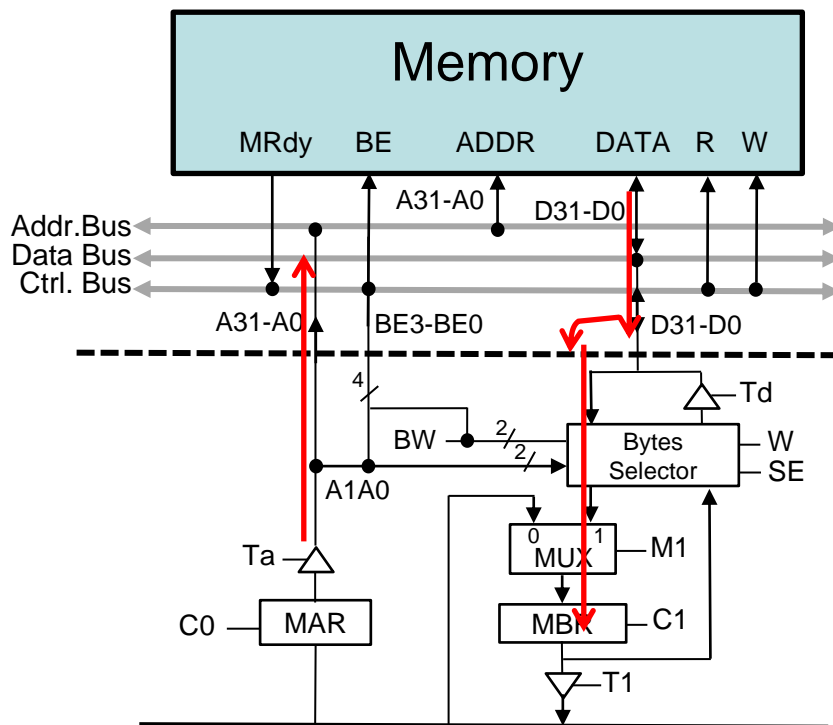


O. Elemental	Señales
MAR ← <dirección>	..., C0

Ejemplo

Acceso a memoria síncrona de 1 ciclo

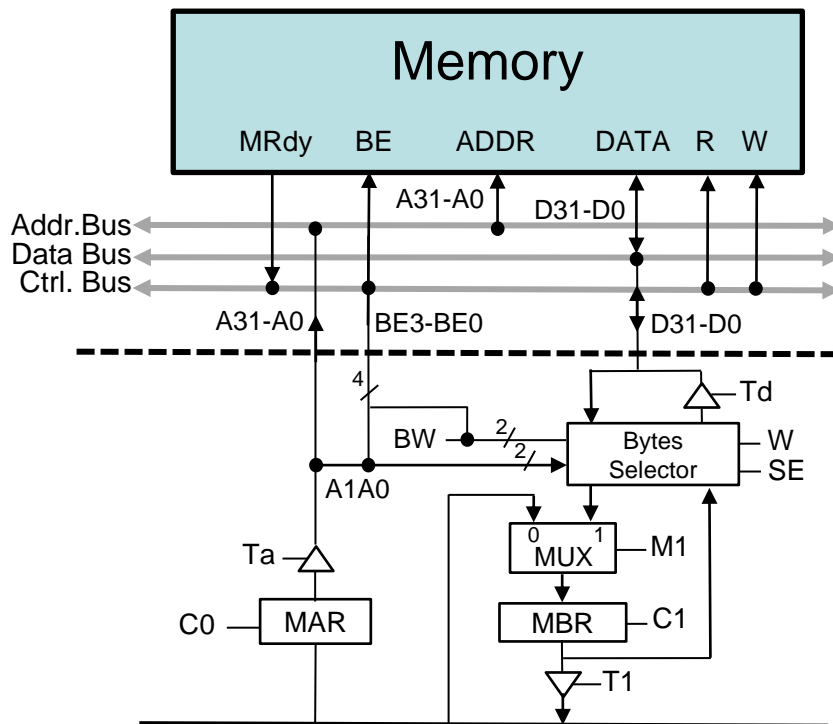
► Lectura de una palabra



O. Elemental	Señales
MAR \leftarrow <dirección>	..., C0
MBR \leftarrow MP[MAR]	Ta, R, M1, C1, BW=11

Ejemplo

Acceso a memoria síncrona de 1 ciclo



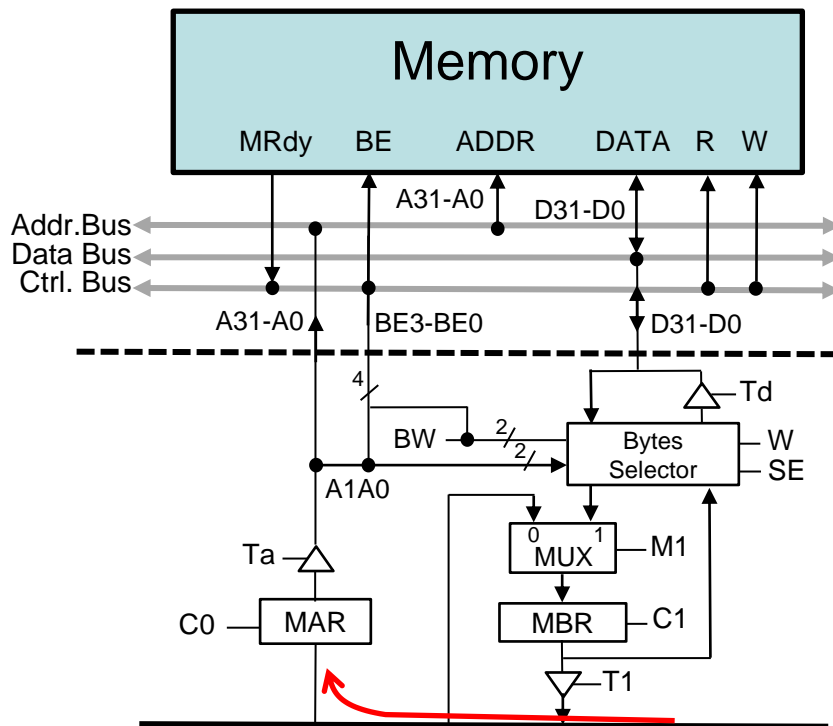
► Lectura de una palabra

O. Elemental	Señales
$MAR \leftarrow \langle \text{dirección} \rangle$..., C0
$MBR \leftarrow MP[MAR]$	Ta, R, M1, C1, BW=11

► Escritura de una palabra

Ejemplo

Acceso a memoria síncrona de 1 ciclo



► Lectura

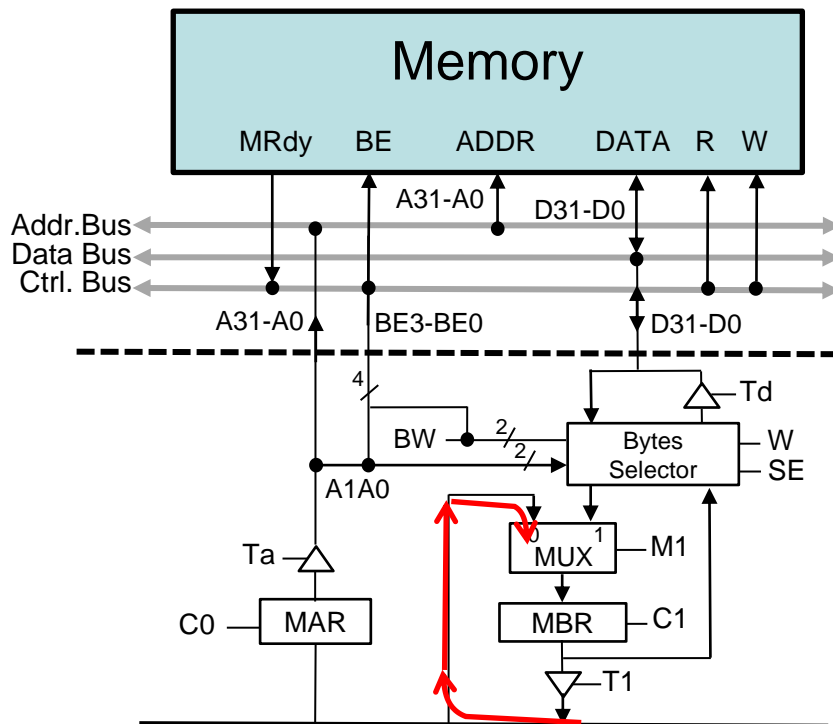
O. Elemental	Señales
MAR \leftarrow <dirección>	..., C0
MBR \leftarrow MP[MAR]	Ta, R, M1, C1

► Escritura de una palabra

O. Elemental	Señales
MAR \leftarrow <dirección>	..., C0

Ejemplo

Acceso a memoria síncrona de 1 ciclo



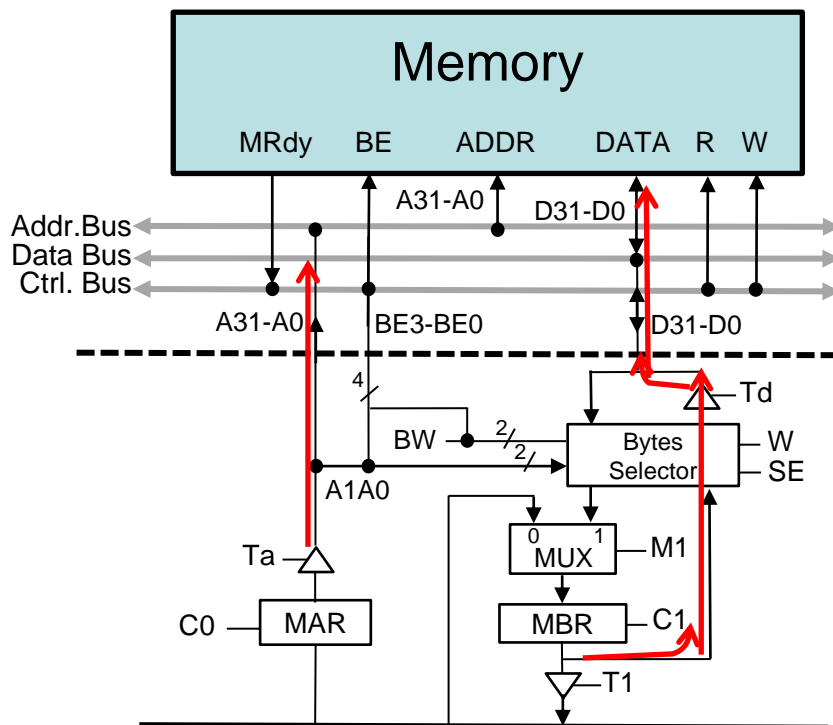
► Lectura

O. Elemental	Señales
MAR \leftarrow <dirección>	..., C0
MBR \leftarrow MP[MAR]	Ta, R, M1, C1

► Escritura

O. Elemental	Señales
MAR \leftarrow <dirección>	..., C0
MBR \leftarrow <dato>	..., C1

Acceso a memoria síncrona de 1 ciclo



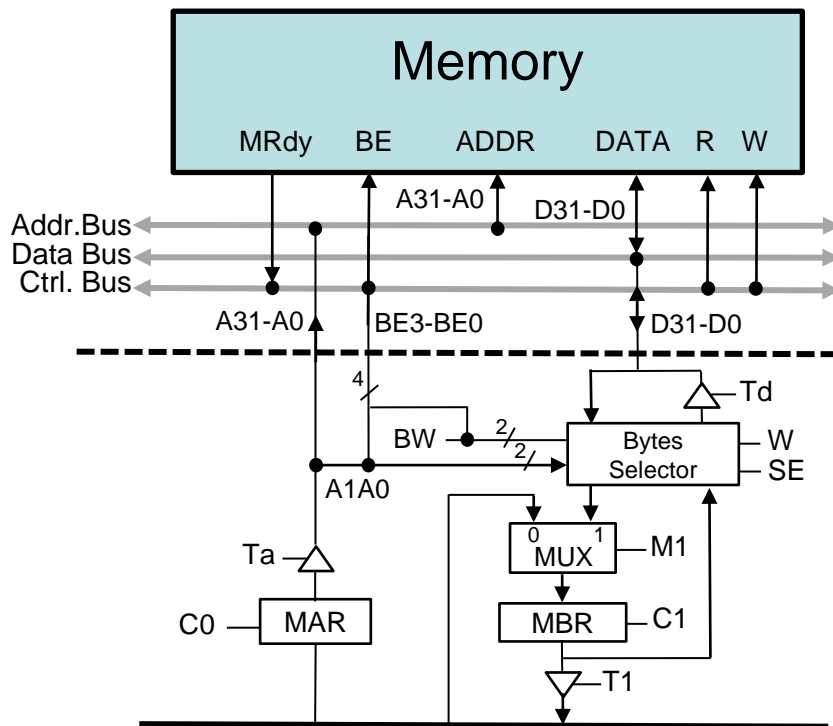
► Lectura

O. Elemental	Señales
MAR \leftarrow <dirección>	..., C0
MBR \leftarrow MP[MAR]	Ta, R, M1, C1

► Escritura

O. Elemental	Señales
MAR \leftarrow <dirección>	..., C0
MBR \leftarrow <dato>	..., C1
Ciclo de escritura	Ta, Td, W, BW=11

Acceso a memoria síncrona de 1 ciclo



► Lectura

O. Elemental	Señales
MAR \leftarrow <dirección>	..., C0
MBR \leftarrow MP[MAR]	Ta, R, M1, C1

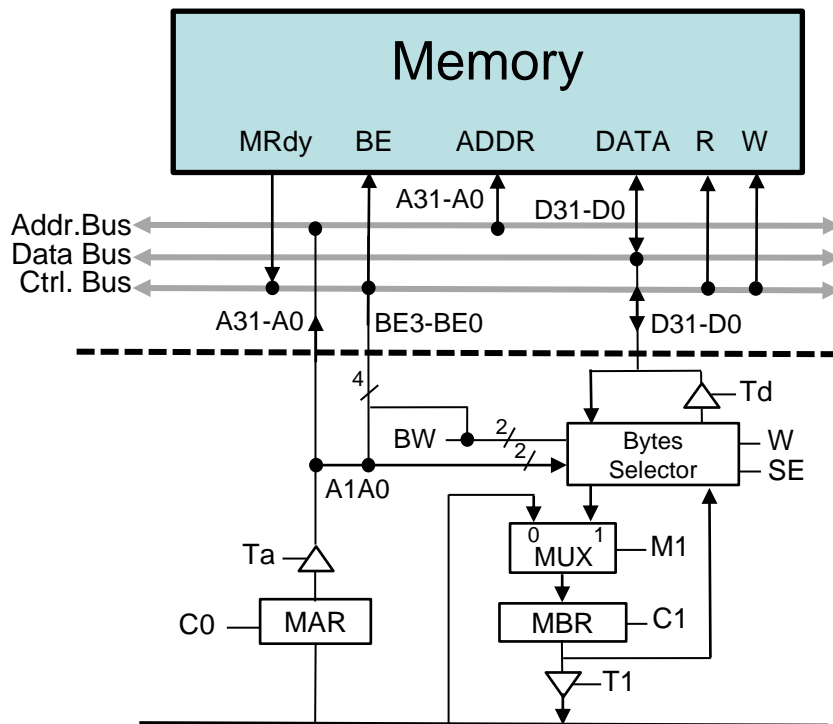
► Escritura

O. Elemental	Señales
MAR \leftarrow <dirección>	..., C0
MBR \leftarrow <dato>	..., C1
Ciclo de escritura	Ta, Td, W, BW=11

Ejemplo

Acceso a memoria síncrona de 2 ciclo

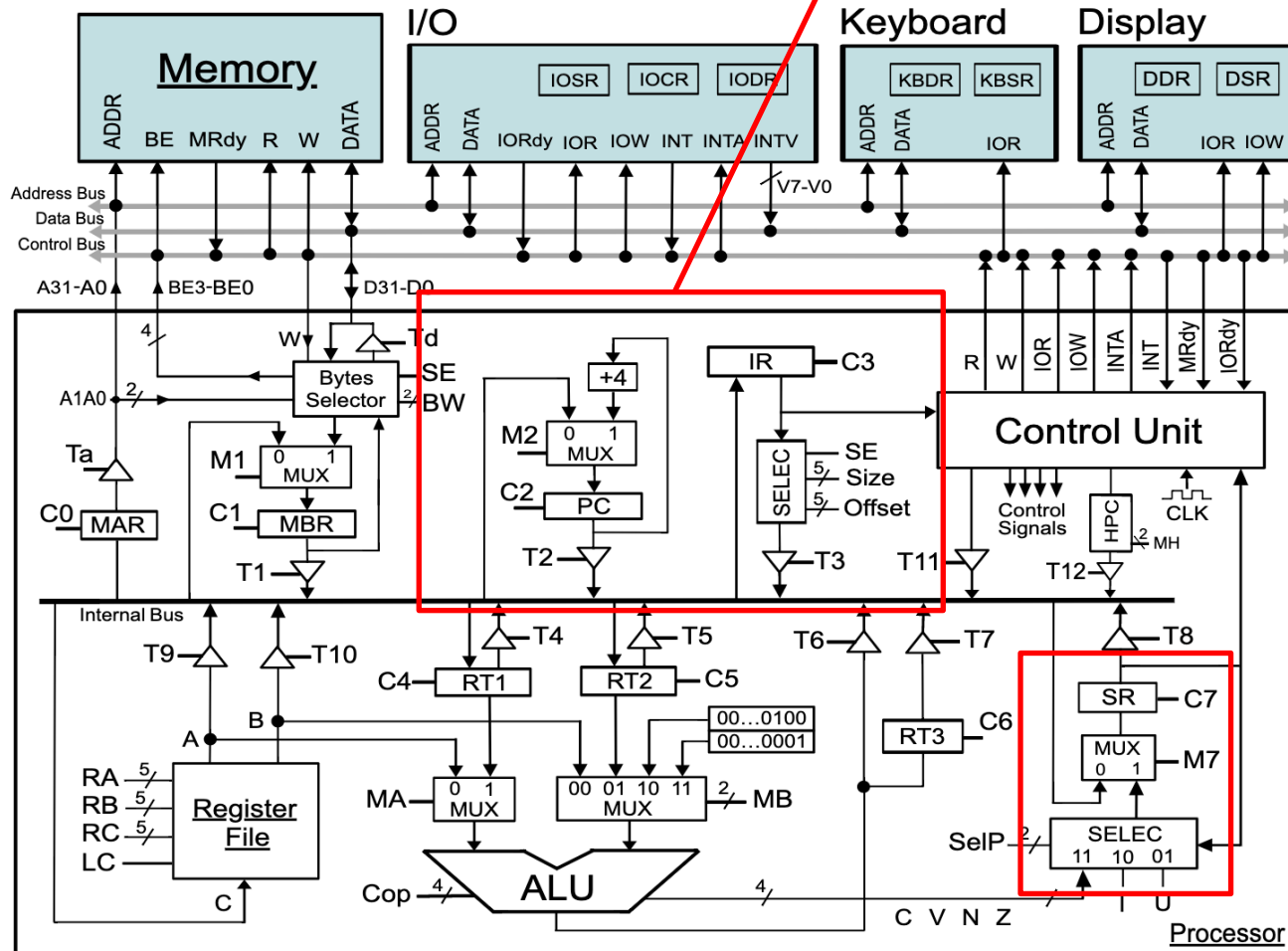
► Lectura de una palabra



O. Elemental	Señales
MAR \leftarrow <dirección>	..., C0
ciclo de lectura	Ta, R,
ciclo de lectura MBR \leftarrow MP[MAR]	Ta, R, M1, C1, BW=11

Estructura de un computador elemental

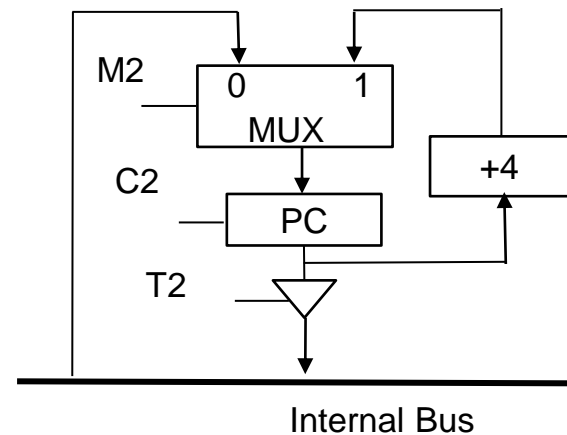
Registros SR, PC y IR



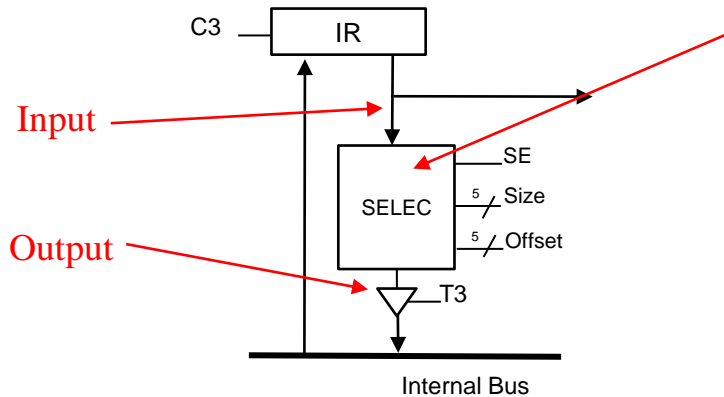
Contador de programa

► Contador de programa PC:

- C2, M2
 - $PC \leftarrow PC + 4$
- C2 – del bus interno al PC
- T2 – de PC a bus interno



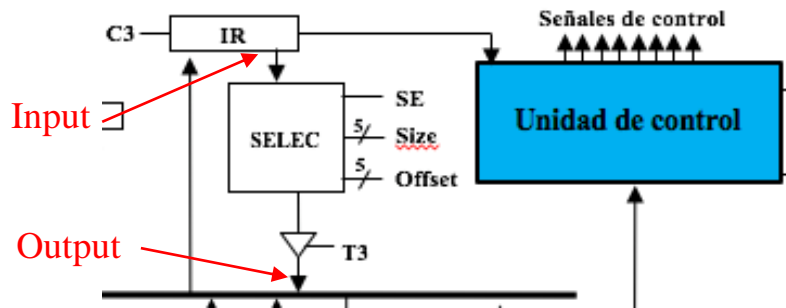
Registro de instrucción (IR)



- ▶ C3: carga del bus interno al IR
- ▶ SELEC: Transfiere el contenido de IR al bus
 - ▶ Size: tamaño
 - ▶ Offset: desplazamiento
 - ▶ Bit de inicio (menos significativo)
 - ▶ SE: extensión de signo

Selector de registro

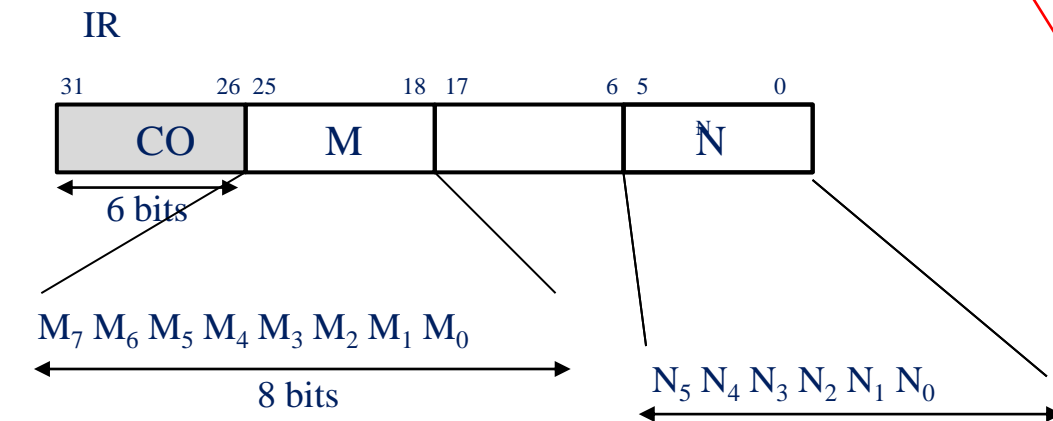
Selección **sin** extensión de signo (SE = 0)



Size	Offset	Output			
01000	10010	31	24 23	16 15	8 7 0
		0	0	0	$M_7..M_0$
00110	00000	31	24 23	16 15	8 7 0
		0	0	0	$00N_5...N_0$

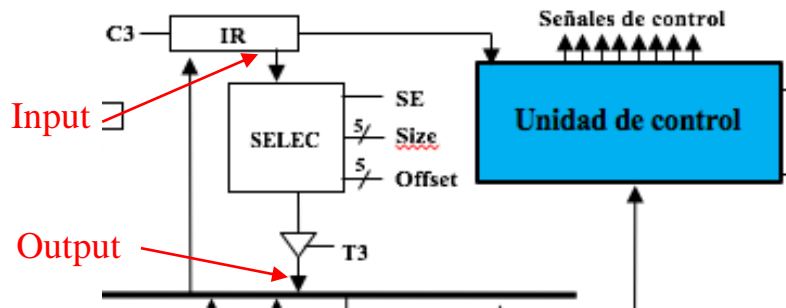
Bit de inicio
(menos significativo)

Tamaño en bits



Selector de registro

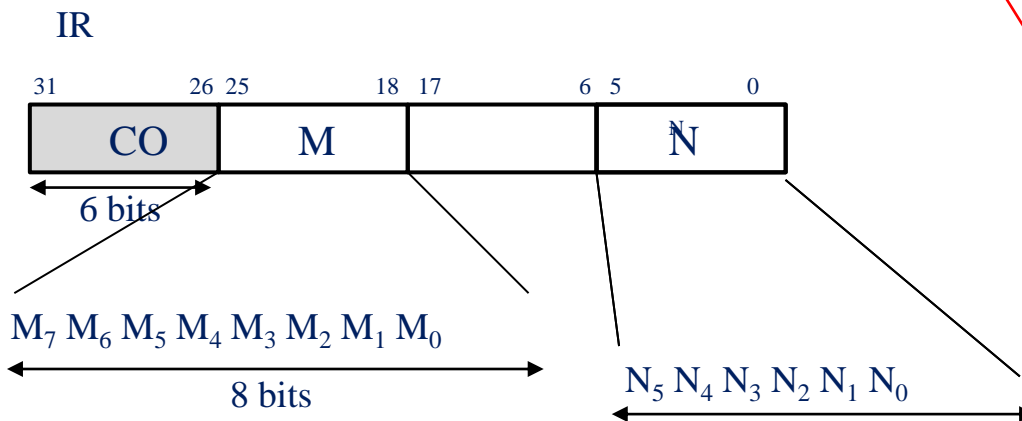
Selección **con** extensión de signo (SE = 1)



Size	Offset	Output			
01000	10010	31	24 23	16 15	8 7 0
		$M_7..M_7$ $M_7..M_7$ $M_7..M_7$ $M_7..M_0$			
00110	00000	31	24 23	16 15	8 7 0
		$N_5..N_5$ $N_5..N_5$ $N_5..N_5$ $N_5N_5N_5..N_0$			

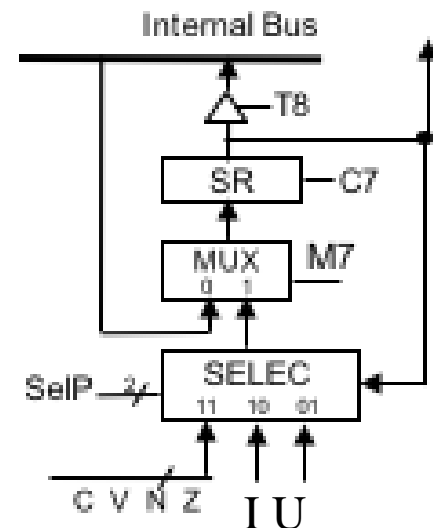
Bit de inicio
(menos significativo)

Tamaño en bits

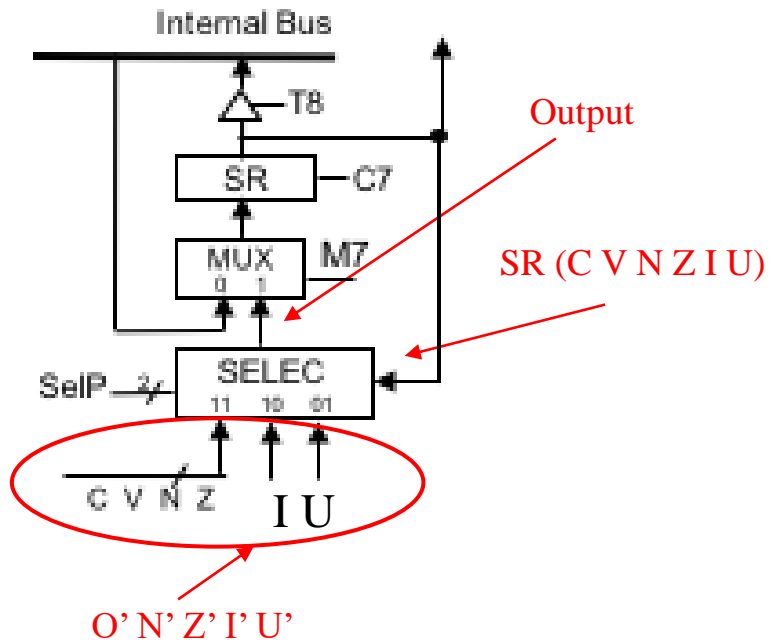


Registro de estado

- ▶ Almacena información (bits de estado) sobre el estado del programa que se está ejecutando en el procesador:
 - ▶ Resultado de la última operación en la ALU: C, V, N, Z
 - ▶ Si el procesador ejecuta en modo núcleo o modo usuario (U)
 - ▶ Si las interrupciones están habilitadas o no (I)
- ▶ Señales:
 - ▶ C7 – de bus interno al SR
 - ▶ SelP, M7 – flags de ALU, I, o U a SR
 - ▶ T8 – del SR al bus interno



Registro de estado



Operación de SELEC:

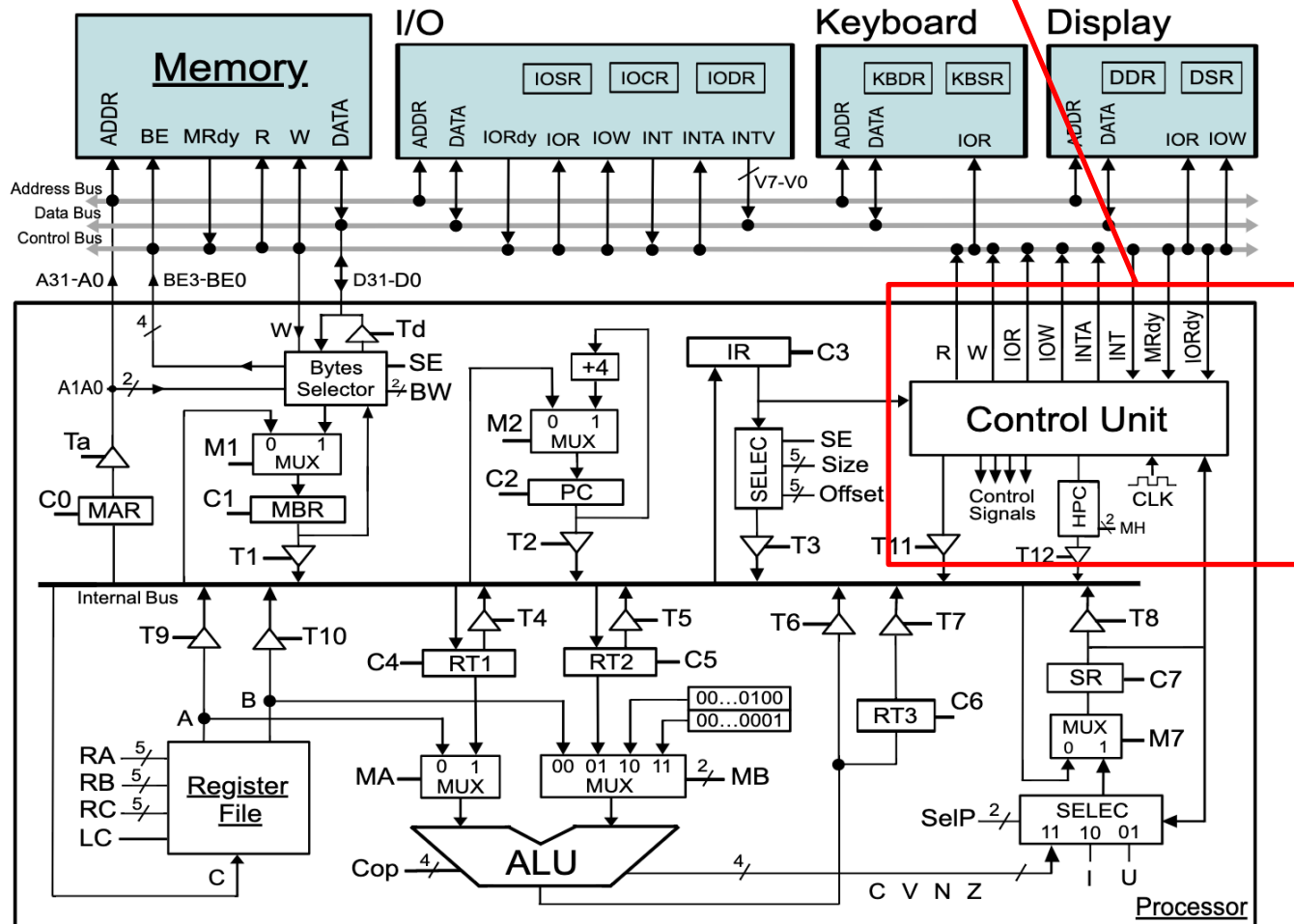
if (SelP1 = 1 AND SelP0 == 1)
Output = **C' V' N' Z'** I U

if (SelP1 == 1 AND SelP0 == 0)
Output = C V N Z **I'** U

if (SelP1 == 0 AND SelP0 == 1)
Output = C V N Z I **U'**

Estructura de un computador elemental

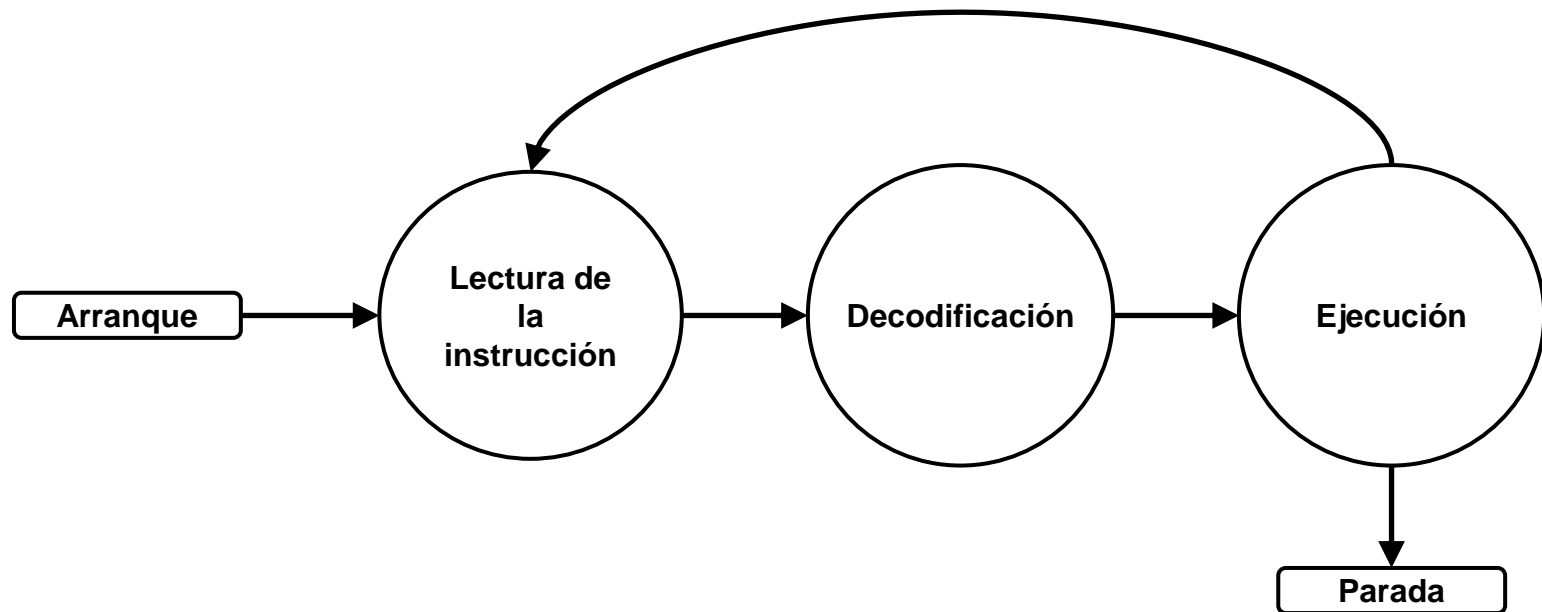
Unidad de Control (UC)



Unidad de control

Fases de ejecución de una instrucción

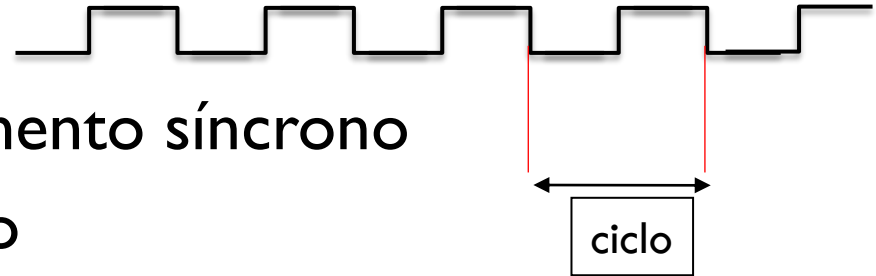
- ▶ Funciones básicas
 - ▶ Lectura de instrucciones de la memoria
 - ▶ Decodificación
 - ▶ Ejecución de instrucciones



Fases de ejecución de una instrucción

- ▶ **Lectura de la instrucción**, captación o *fetch*
 - ▶ Leer la instrucción almacenada en la dirección de memoria indicada por PC y llevarla a RI.
 - ▶ Incremento del PC
- ▶ **Decodificación**
 - ▶ Análisis de la instrucción en RI para determinar:
 - ▶ La operación a realizar.
 - ▶ Direccionamiento a aplicar.
 - ▶ Señales de control a activar
- ▶ **Ejecución**
 - ▶ Generación de las señales de control en cada ciclo de reloj.

Reloj

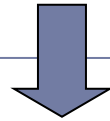


- ▶ Un computador es un elemento síncrono
- ▶ Controla el funcionamiento
- ▶ El reloj temporiza las operaciones
 - ▶ En un ciclo de reloj se ejecutan una o más operaciones elementales siempre que no haya conflicto
 - ▶ Durante el ciclo se mantienen activadas las señales de control necesarias
- ▶ En un mismo ciclo se puede realizar
 - ▶ $MAR \leftarrow PC$ y $RT3 \leftarrow RT2 + RT1$
- ▶ En un mismo ciclo **no** se puede realizar
 - ▶ $MAR \leftarrow PC$ y $RI \leftarrow RT3$ ¿por qué?

Descripción de la actividad de la U.C.

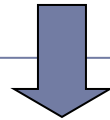
Instrucción

mv R0 R1

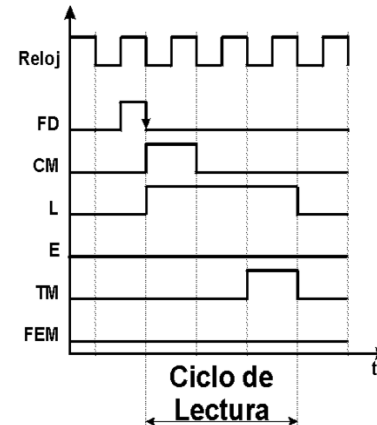


Secuencia de **operaciones elementales**

- $RI \leftarrow [PC]$
- $PC++$
- decodificación
- $R0 \leftarrow R1$

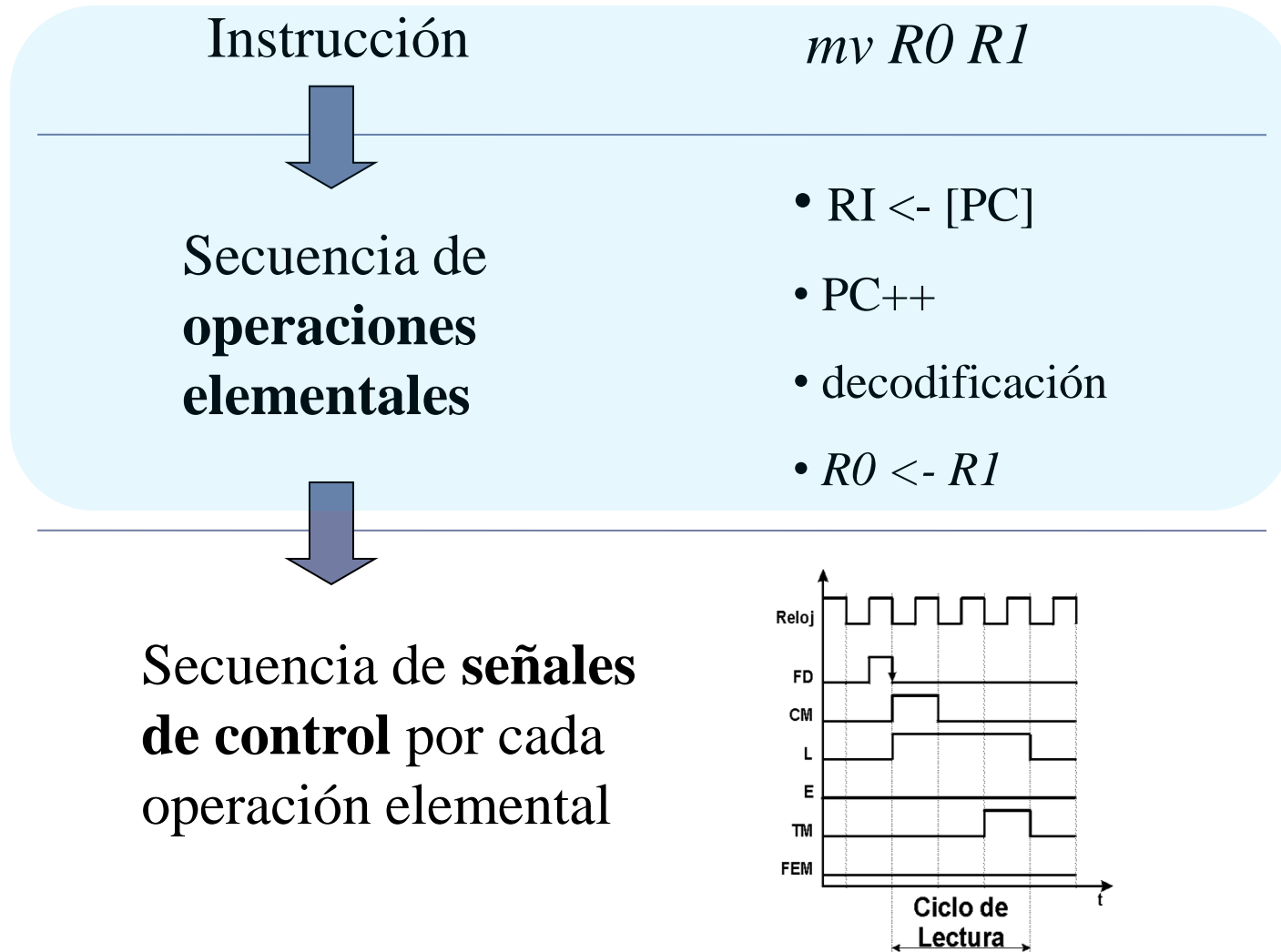


Secuencia de **señales de control** por cada operación elemental



+ nivel de detalle Hw.

Descripción de la actividad de la U.C.



Lectura de una instrucción

Ciclo	Op. Elemental
C1	$MAR \leftarrow PC$
C2	$PC \leftarrow PC + 4$
C3	$MBR \leftarrow MP$
C4	$IR \leftarrow MBR$



Ciclo	Op. Elemental
C1	$MAR \leftarrow PC$
C2	$PC \leftarrow PC + 4,$ $MBR \leftarrow MP$
C3	$IR \leftarrow MBR$

Posibilidad de operaciones simultáneas

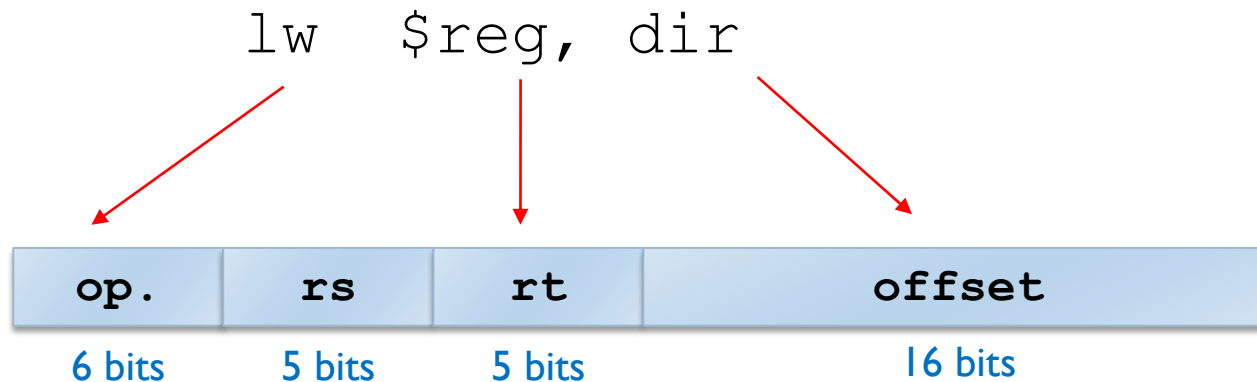
Señales de control del ciclo de fetch

- Especificación de las señales de control activas en cada ciclo de reloj.
 - Se puede generar a partir del nivel RT.

Ciclo	Op. Elemental	Señales de control activadas
C1	$MAR \leftarrow PC$	T2, C0
C2	$PC \leftarrow PC + 4,$ $MBR \leftarrow MP$	C2, M2 Ta, R, CI, MI, BW=11
C3	$IR \leftarrow MBR$	T1, C3

Ejecución de la instrucción de MIPS

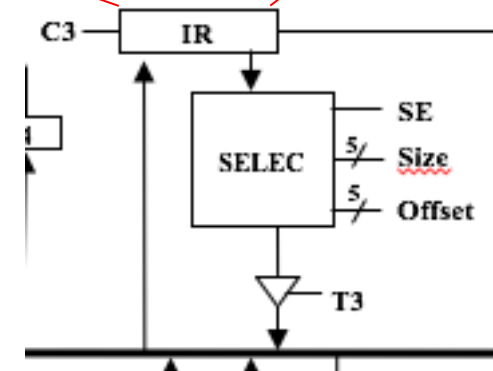
► `lw $reg, dir`



Ejecución de `lw $reg, dir`



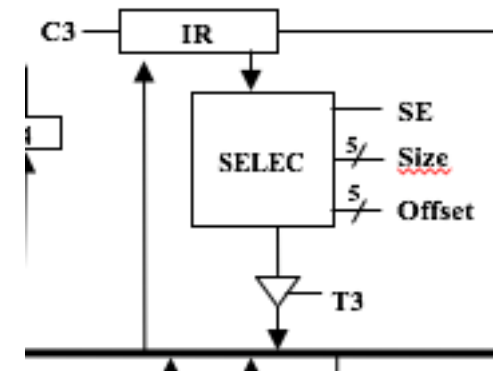
Ciclo	Op. Elemental	Señales de control
C1	$MAR \leftarrow PC$	T2, C0
C2	$PC \leftarrow PC + 4,$ $MBR \leftarrow MP$	C2, M2 Ta, R, CI, MI, BW=II
C3	$IR \leftarrow MBR$	T1, C3
C4		
C5		
C6		
C7		



Ejecución de lw \$reg, dir



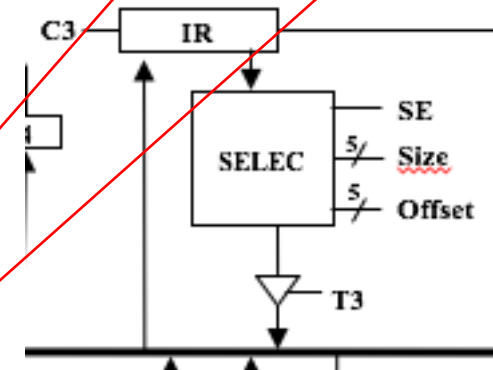
Ciclo	Op. Elemental	Señales de control
C1	$MAR \leftarrow PC$	T2, C0
C2	$PC \leftarrow PC + 4,$ $MBR \leftarrow MP$	C2, M2 Ta, R, CI, MI, BW=II
C3	$IR \leftarrow MBR$	T1, C3
C4	Decodificación	
C5		
C6		
C7		



Ejecución de `lw $reg, dir`



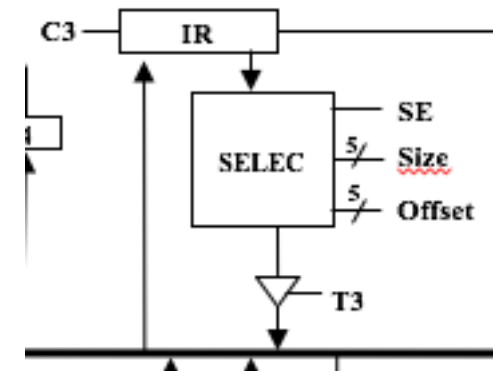
Ciclo	Op. Elemental	Señales de control
C1	$MAR \leftarrow PC$	T2, C0
C2	$PC \leftarrow PC + 4,$ $MBR \leftarrow MP$	C2, M2 Ta, R, CI, MI, BW=11
C3	$IR \leftarrow MBR$	T1, C3
C4	Decodificación	
C5	$MAR \leftarrow RI(dir)$	C0, T3, Size = 10000 Offset = 00000
C6		
C7		



Ejecución de lw \$reg, dir



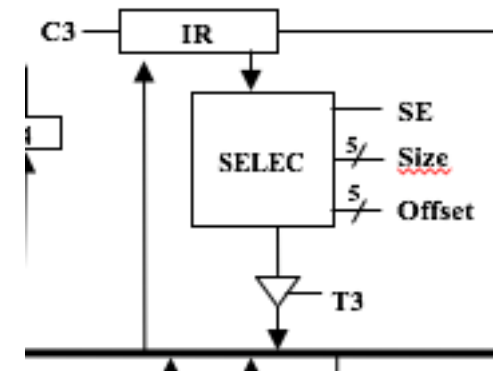
Ciclo	Op. Elemental	Señales de control
C1	$MAR \leftarrow PC$	T2, C0
C2	$PC \leftarrow PC + 4,$ $MBR \leftarrow MP$	C2, M2 Ta, R, CI, MI, BW=11
C3	$IR \leftarrow MBR$	T1, C3
C4	Decodificación	
C5	$MAR \leftarrow RI(dir)$	C0, T3, Size = 10000 Offset = 00000
C6	$MBR \leftarrow MP$	Ta, R, CI, MI, BW=11
C7		



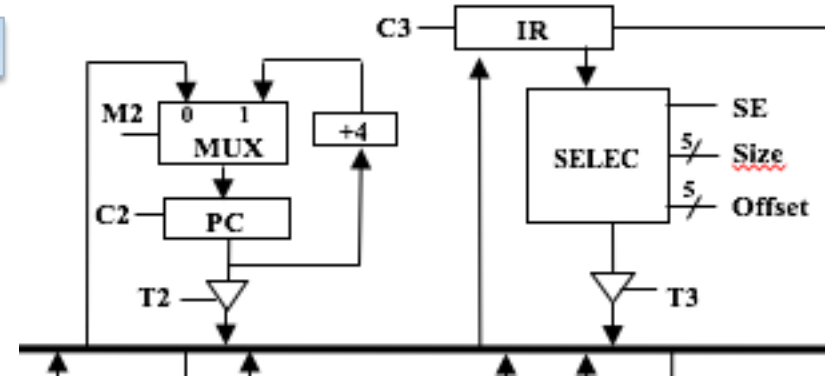
Ejecución de `lw $reg, dir`



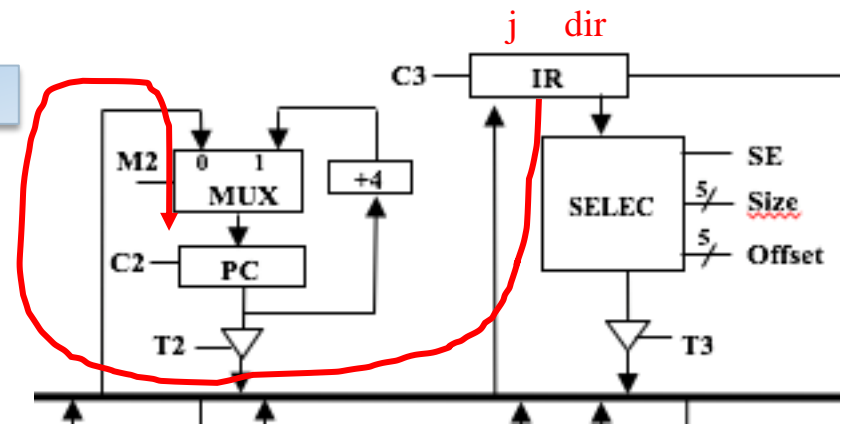
Ciclo	Op. Elemental	Señales de control
C1	$MAR \leftarrow PC$	T2, C0
C2	$PC \leftarrow PC + 4,$ $MBR \leftarrow MP$	C2, M2 Ta, R, CI, MI, BW=11
C3	$IR \leftarrow MBR$	T1, C3
C4	Decodificación	
C5	$MAR \leftarrow RI(dir)$	C0, T3, Size = 10000 Offset = 00000
C6	$MBR \leftarrow MP$	Ta, R, CI, MI, BW=11
C7	$\$reg \leftarrow MBR$	T1, RC=id \$reg, LC



Ejecución de j dir



Ejecución de j dir



Ciclo	Op. Elemental	Señales de control
C1	$MAR \leftarrow PC$	T2, C0
C2	$PC \leftarrow PC + 4,$ $MBR \leftarrow MP$	C2, MI Ta, R, CI, MI, BW=11
C3	$IR \leftarrow MBR$	T1, C3
C4	Decodificación	
C5	$PC \leftarrow RI(dir)$	C2, T3, Size = 11010 (26) Offset = 00000

Ejercicio

► Instrucciones que caben en una palabra:

- `sw $reg, dir`
- `add $rd, $ro1, $ro2`
- `addi $rd, $ro1, inm`
- `lw $reg1, desp($reg2)`
- `j dir`
- `jr $reg`
- `beq $ro1, $ro2, desp`

beqz \$reg, desplaz

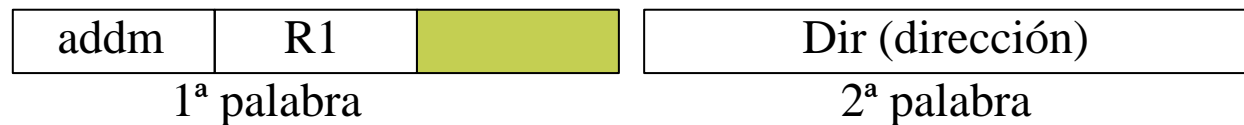
Ciclo	Op. Elemental
C1	$MAR \leftarrow PC$
C2	$PC \leftarrow PC + 4,$ $MBR \leftarrow MP$
C3	$IR \leftarrow MBR$
C4	Decodificación
C5	$\$reg + \0
C6	Si $SR.Z == 0$ salto a fetch
C7	$RT2 \leftarrow PC$
C8	$RT1 \leftarrow IR(\text{desplaz})$
C9	$RT1 \leftarrow RT1 * 4$
C10	$PC \leftarrow RT1 + RT2$

Si $\$reg == 0$
 $PC \leftarrow PC + \text{desp} * 4$

Instrucciones que ocupan varias palabras

Ejemplo: `addm R1, dir` $R1 \leftarrow R1 + MP[dir]$

Formato:



Ciclo	Op. Elemental
C1	$MAR \leftarrow PC$
C2	$PC \leftarrow PC + 4,$ $MBR \leftarrow MP$
C3	$IR \leftarrow MBR$
C4	Decodificación
C5	$MAR \leftarrow PC$

Ciclo	Op. Elemental
C6	$MBR \leftarrow MP,$ $PC \leftarrow PC + 4$
C7	$MAR \leftarrow MBR$
C8	$MBR \leftarrow MP$
C9	$RTI \leftarrow MBR$
C10	$RI \leftarrow RI + RTI$

Ejemplo

ADD (R_2) R_3 (R_4)

A. Fetch + Decodif.

- 1.- $MAR \leftarrow PC$
- 2.- $RI \leftarrow Memoria(MAR)$
- 3.- $PC \leftarrow PC + "4"$
- 4.- Decodificación de la instrucción

B. Traer operandos

- 5.- $MAR \leftarrow R_4$
- 6.- $MBR \leftarrow Memoria(MAR)$
- 7.- $RTI \leftarrow MBR$

C. Ejecutar

- 8.- $MBR \leftarrow R_3 + RTI$

D. Guardar resultados

- 9.- $MAR \leftarrow R_2$
- 10.- $Memoria(MAR) \leftarrow MBR$

Atención

Recordatorio de no es posible, cualquier otra cosa si...

- ▶ **No es posible atravesar un registro en el ciclo de reloj**
- ▶ **No es posible llevar a un bus dos valores a la vez.**
- ▶ **No es posible establecer un camino entre dos elementos si no hay circuitería para ello.**