

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

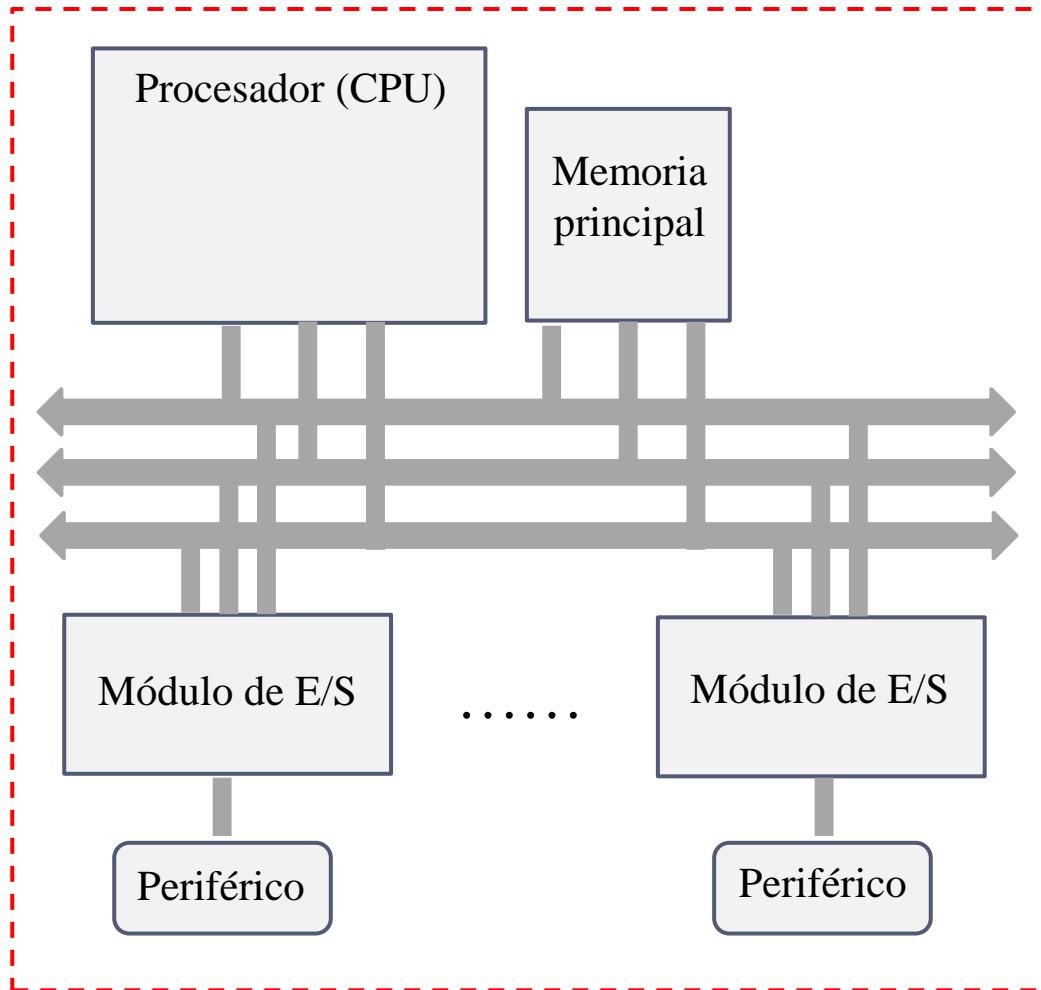
Contenido

1. Elementos de un
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

- 1) Motivación y objetivos
- 2) Funcionamiento básico de la unidad de control
- 3) Señales de control y operaciones elementales
- 4) Introducción al procesador elemental

Componentes de un computador

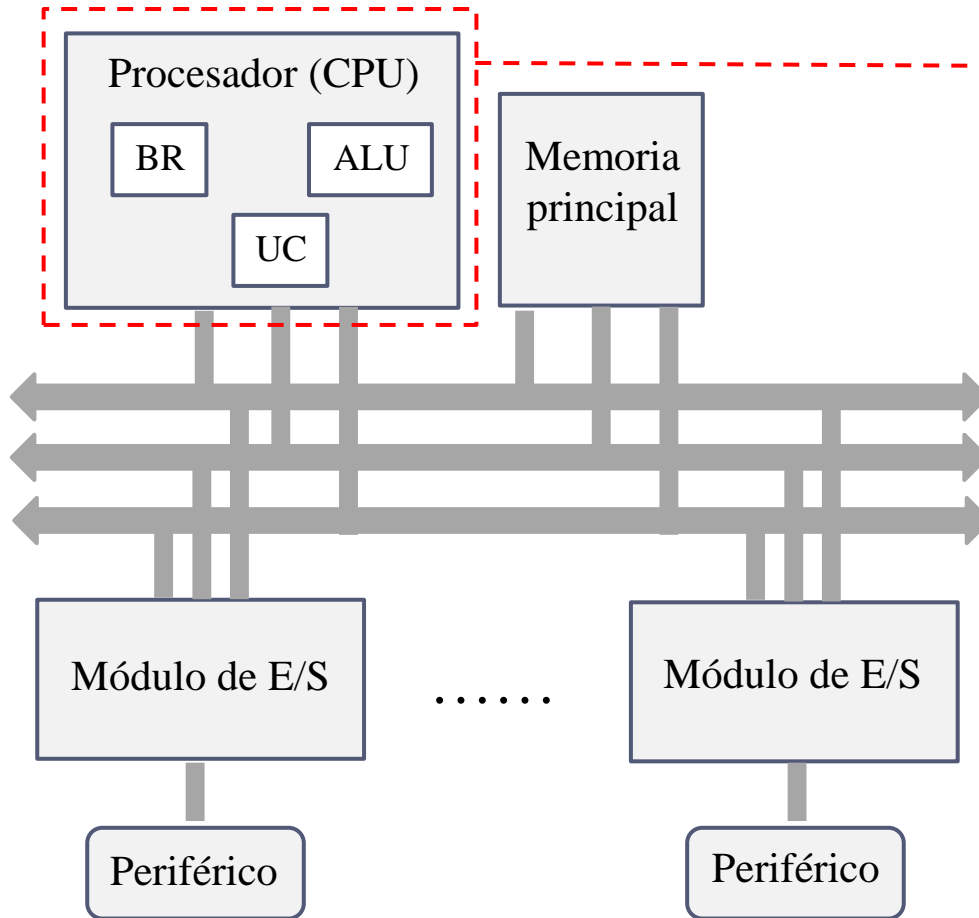
Recordatorio



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

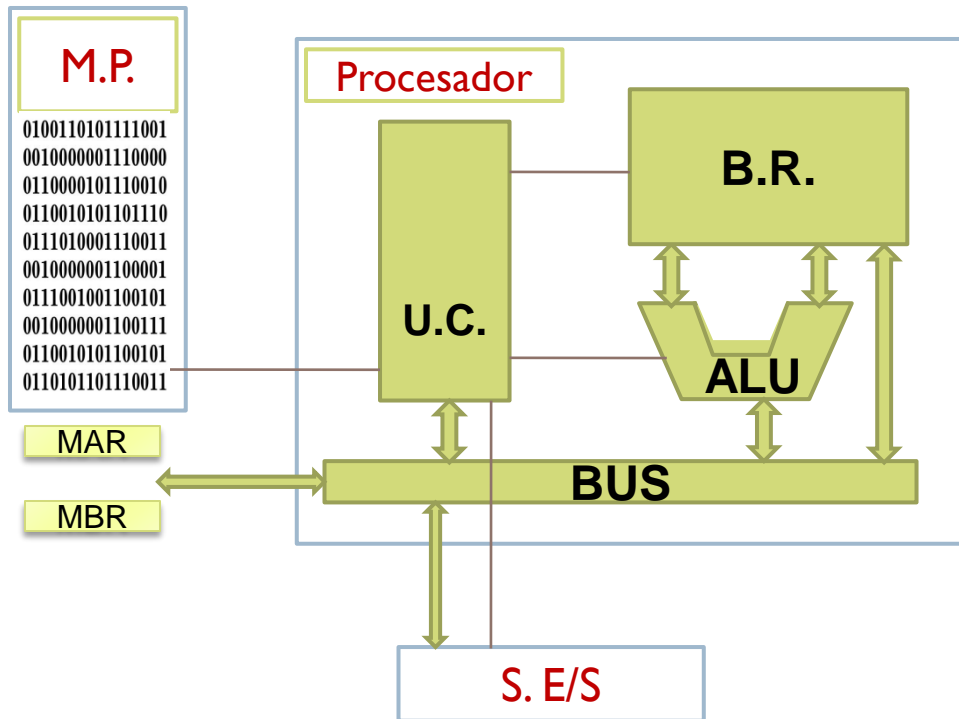
Componentes de un procesador

Recordatorio



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

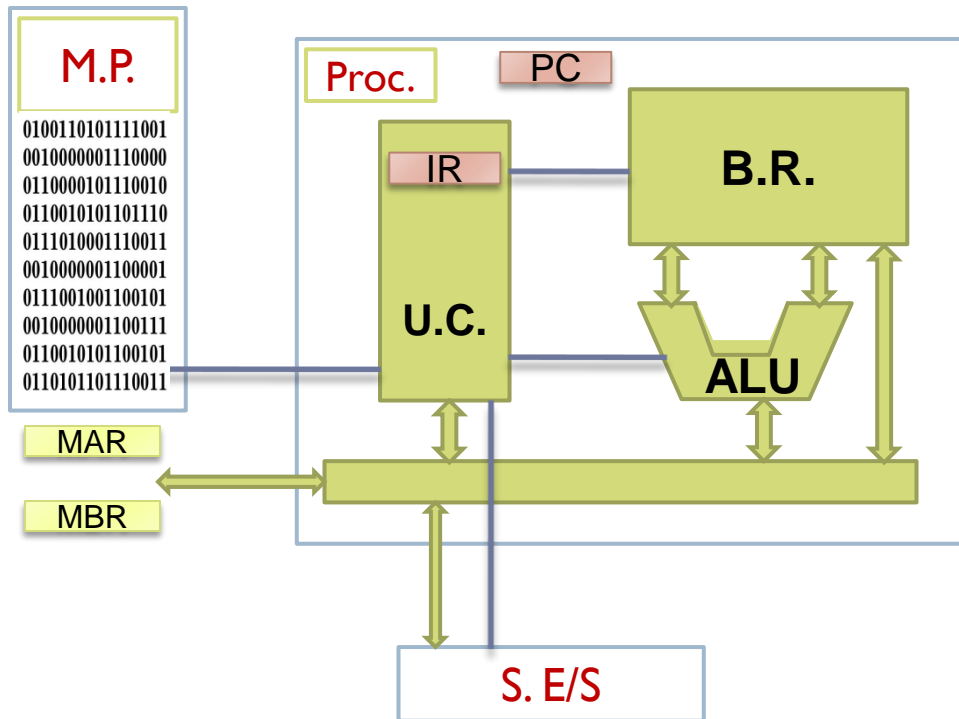
Introducción: motivación



- 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



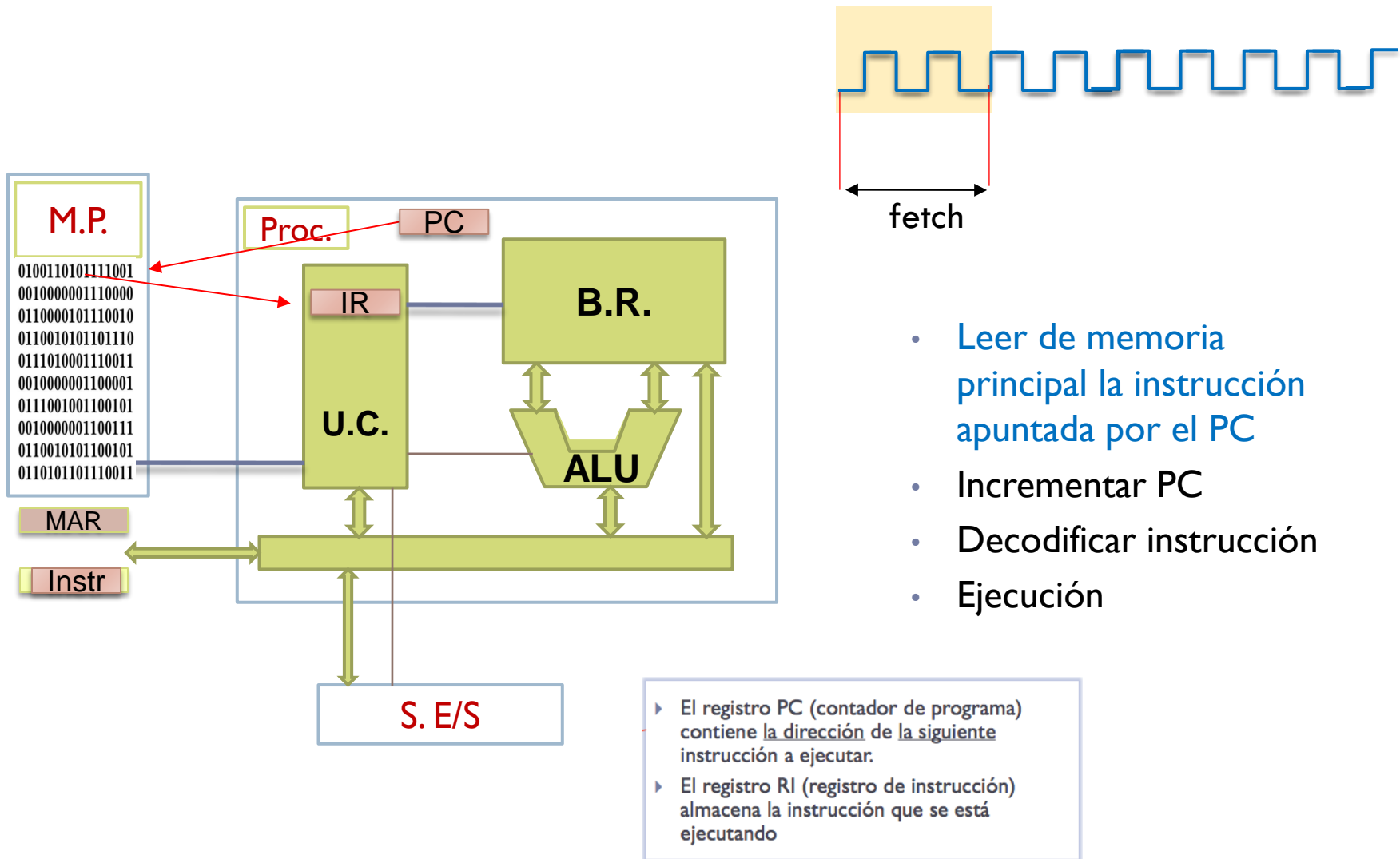
- En cada ciclo de reloj envía la Unidad de Control (U.C.) por los cables del bus de control las señales de control
- Cada elemento del computador tiene entradas, salidas y señales de control que indican qué valor a la salida se ha de tener:
 - Mover de una entrada a salida: $S=E_x$
 - Transformar una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

Contenido

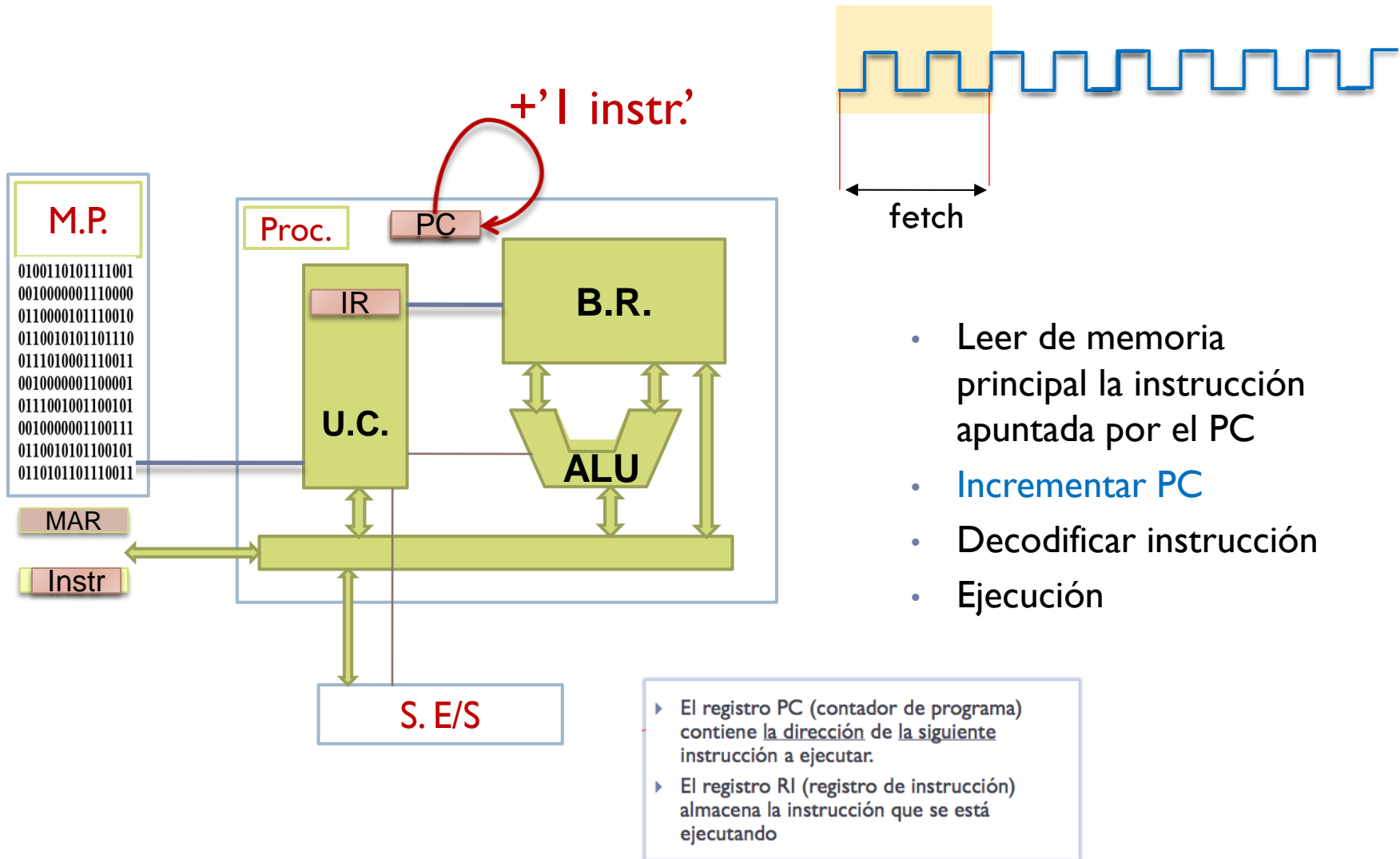
1. Elementos de un
2. Organización del procesador
3. La unidad de control
4. Ejecución de instrucciones
5. Diseño de la unidad de control
6. Modos de ejecución
7. Interrupciones
8. Arranque de un computador
9. Prestaciones y paralelismo

- 1) Motivación y objetivos
- 2) Funcionamiento básico de la unidad de control
- 3) Señales de control y operaciones elementales
- 4) Introducción al procesador elemental

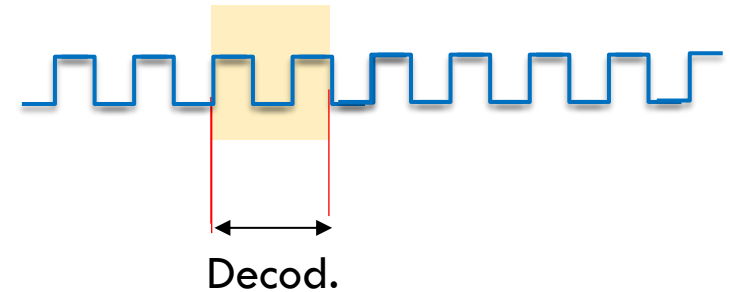
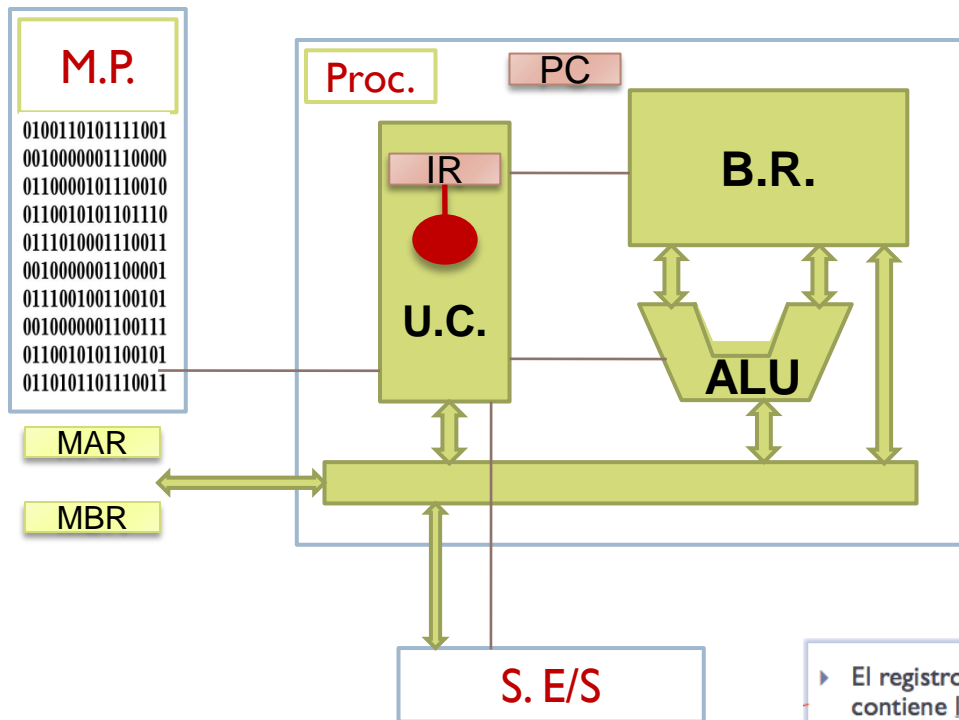
Funcionamiento básico de la U.C.



Funcionamiento básico de la U.C.



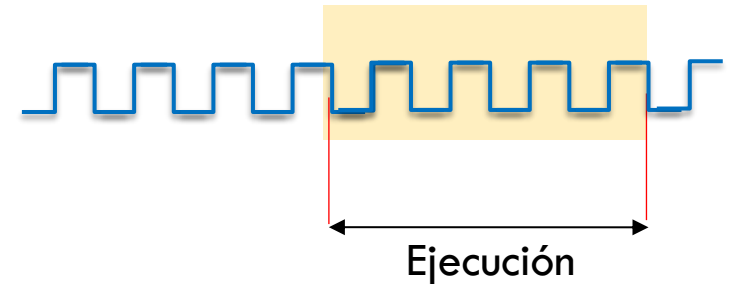
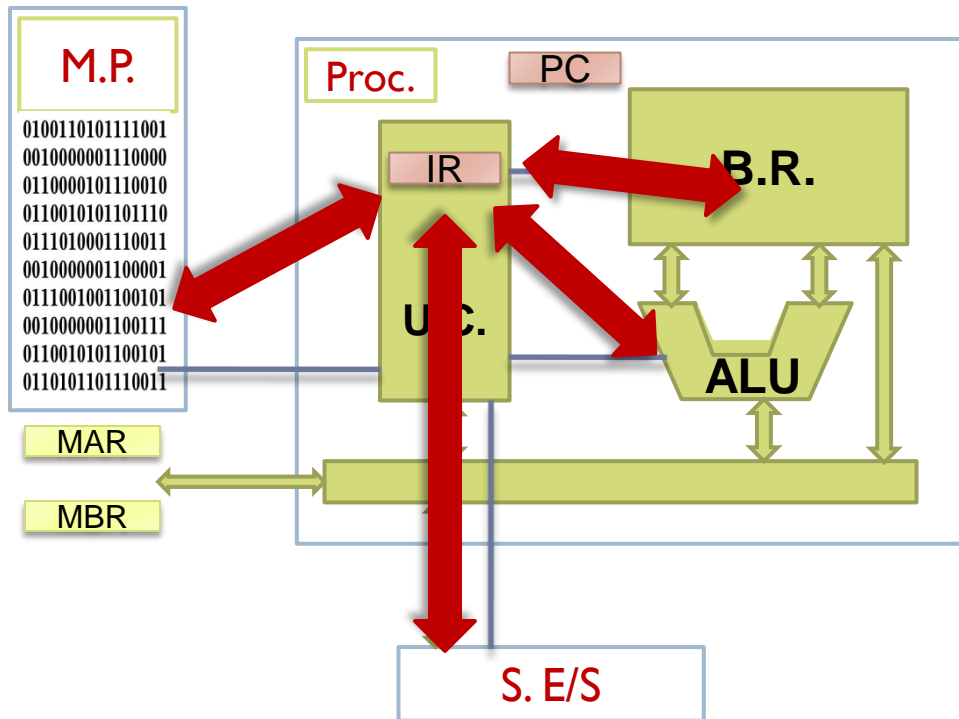
Funcionamiento básico de la U.C.



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

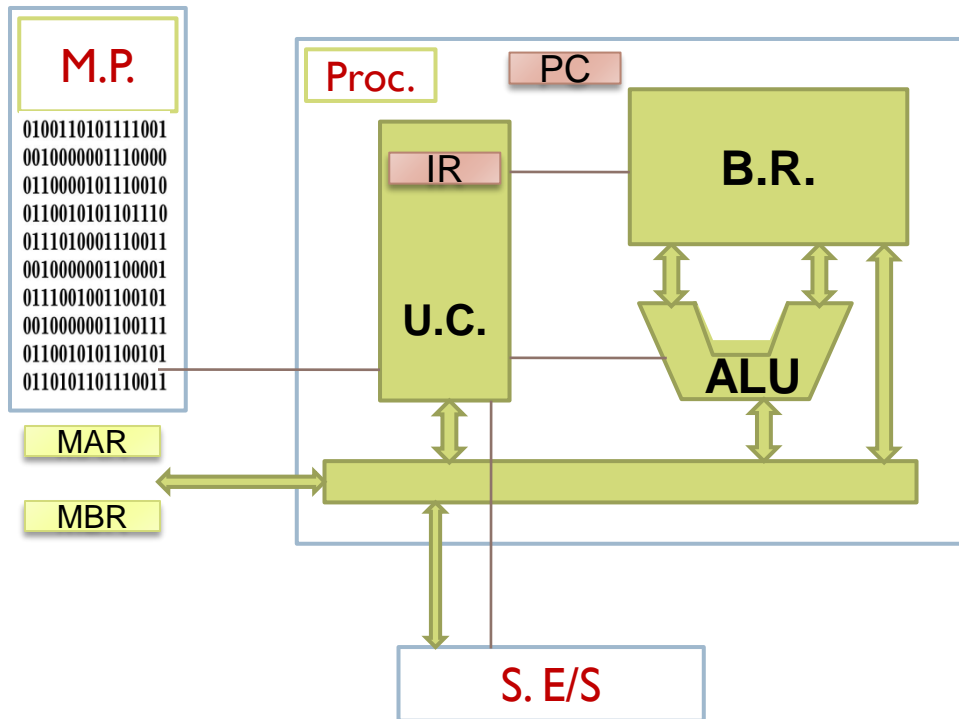
- ▶ El registro PC (contador de programa) contiene la dirección de la siguiente instrucción a ejecutar.
- ▶ El registro RI (registro de instrucción) almacena la instrucción que se está ejecutando

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.



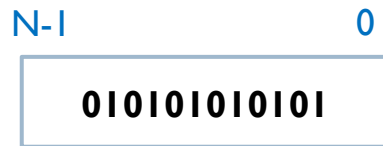
- Resolver situaciones anómalas
 - Instrucciones ilegales
 - Accesos a memoria ilegales
 - ...
- Atender las interrupciones
- Controlar la comunicación con los periféricos

Contenido

1. Elementos de un
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

- 1) Motivación y objetivos
- 2) Funcionamiento básico de la unidad de control
- 3) Señales de control y operaciones elementales
- 4) Introducción al procesador elemental

Registro y bus



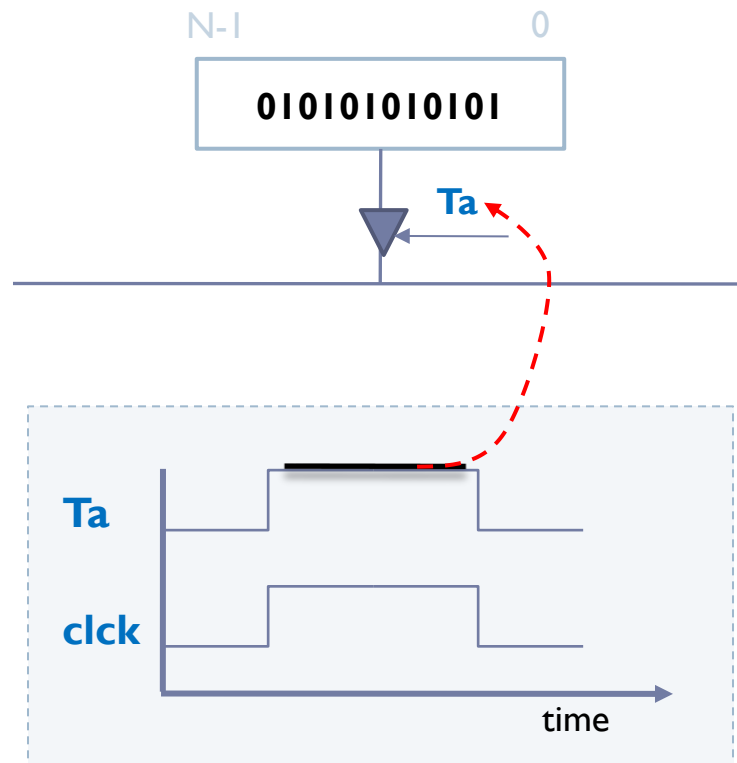
▶ Registro

- ▶ Permite almacenar una lista de bits

▶ Bus

- ▶ elemento que permite transmitir varios bits entre elementos de almacenamiento

Señales: salida triestado



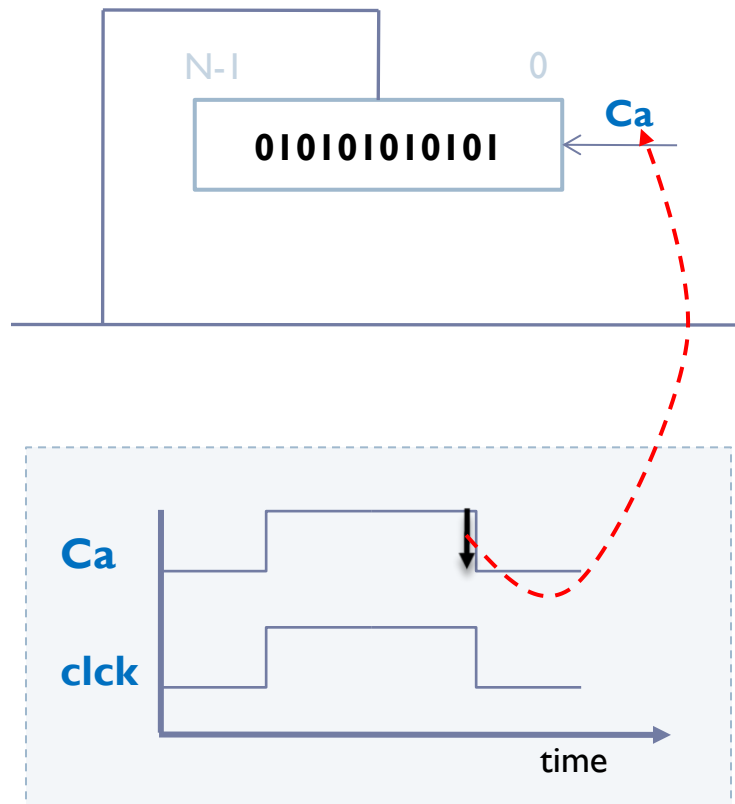
► Triestado

- Usado para conectar un elemento al bus.
- Permite enviar datos al bus.

► IMPORTANTE

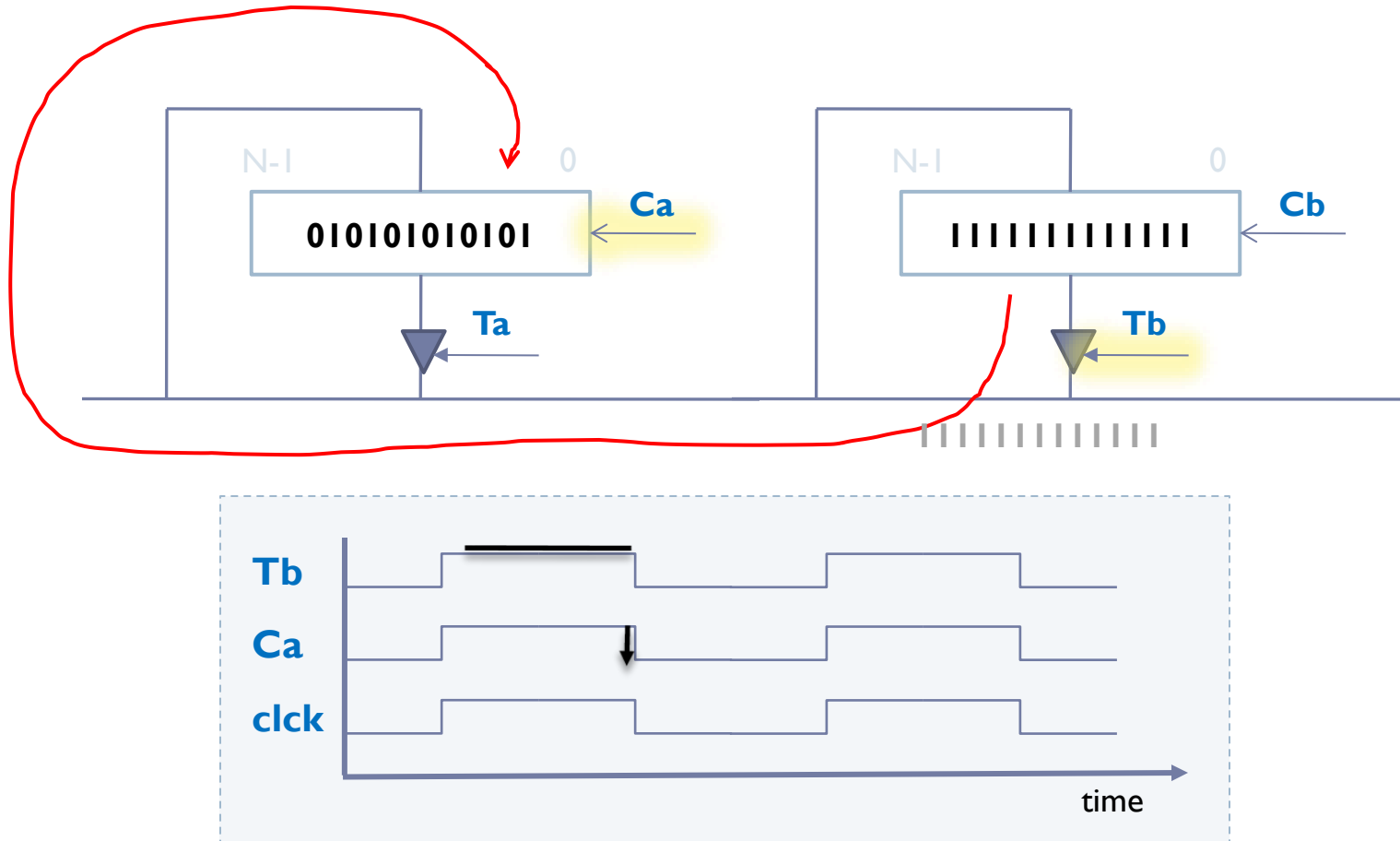
- Dos o más triestados no pueden activarse en el mismo bus al mismo tiempo.

Señales: carga en registro

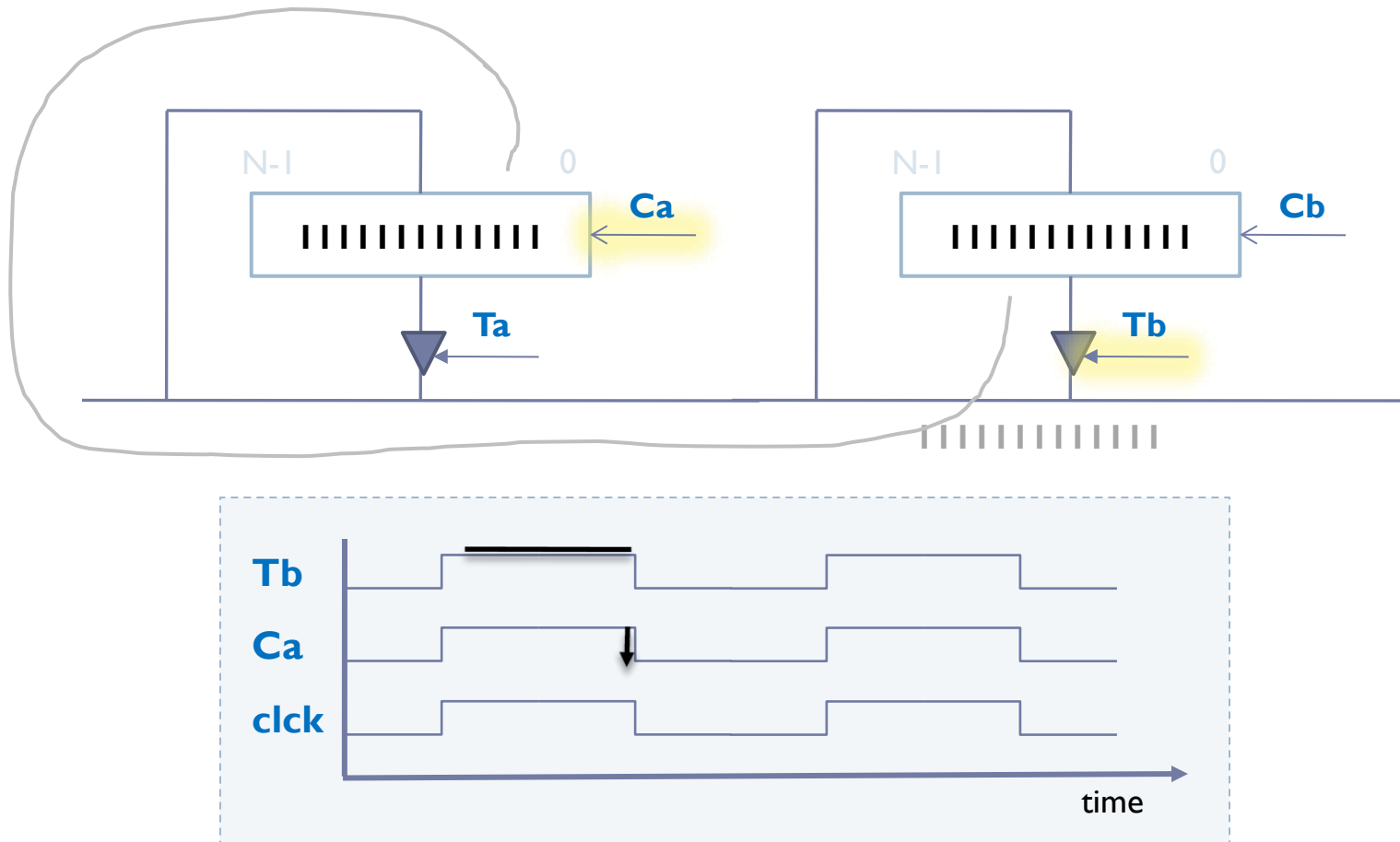


- ▶ **Carga en registro**
 - ▶ Permite almacenar el valor en la entrada en el flanco de bajada del reloj:
 - ▶ Durante el nivel de reloj se mantiene el valor interno (viejo)
 - ▶ Al final del ciclo de reloj (flanco de bajada) es cuando el valor se actualiza.
- ▶ **IMPORTANTE**
 - ▶ Por lo tanto, en el siguiente ciclo, el nuevo valor se verá en la salida

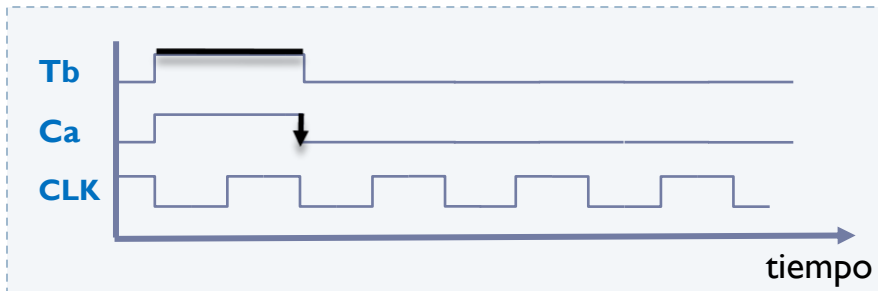
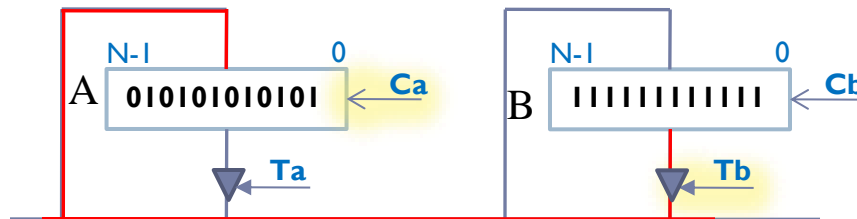
Secuencia de señales



Secuencia de señales



Ejemplo de operación elemental de transferencia



► Operación elemental de transferencia:

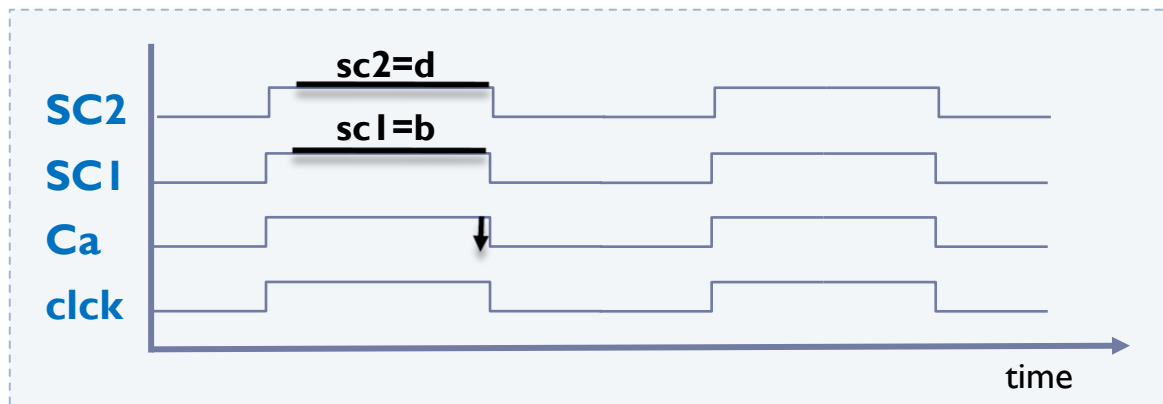
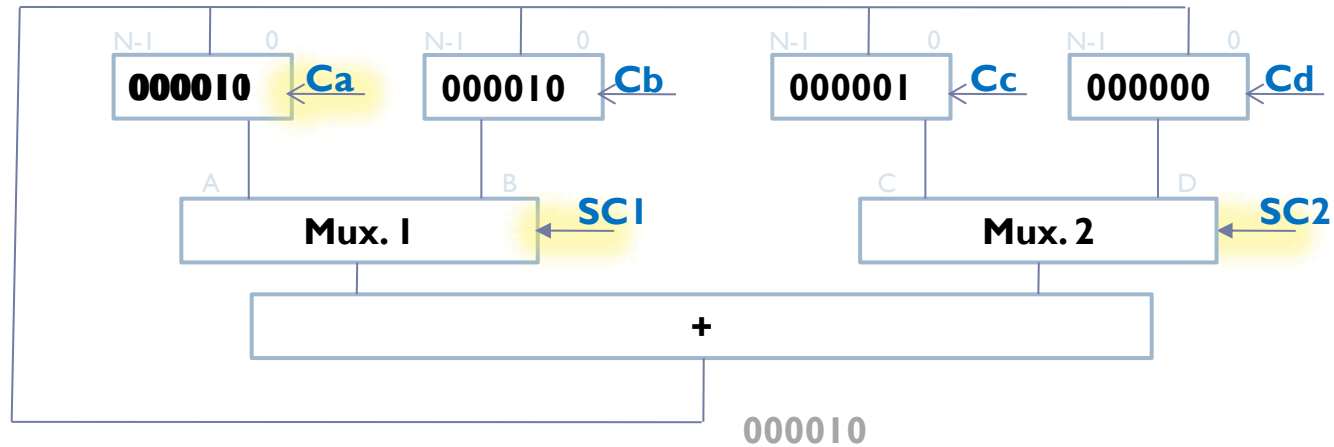
- Elemento de almacenamiento origen
- Elemento de almacenamiento destino
- Se establece un camino

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

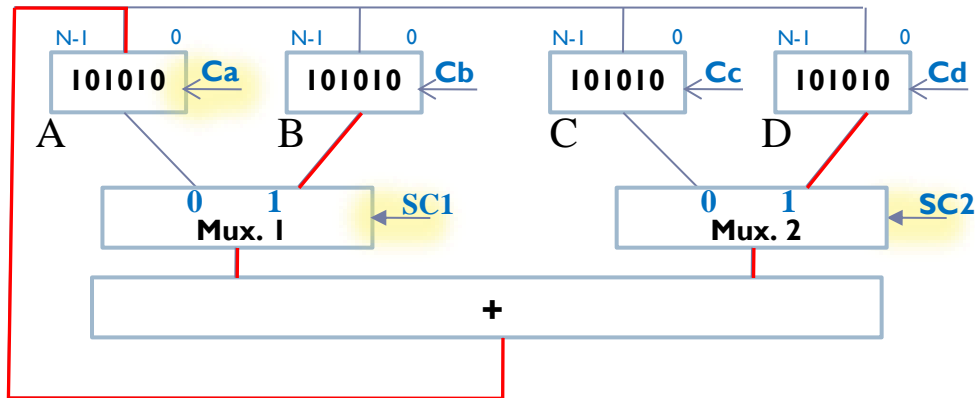
► IMPORTANTE

- 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.
 - Hacer lo que la circuitería no deja.

Secuencia de señales



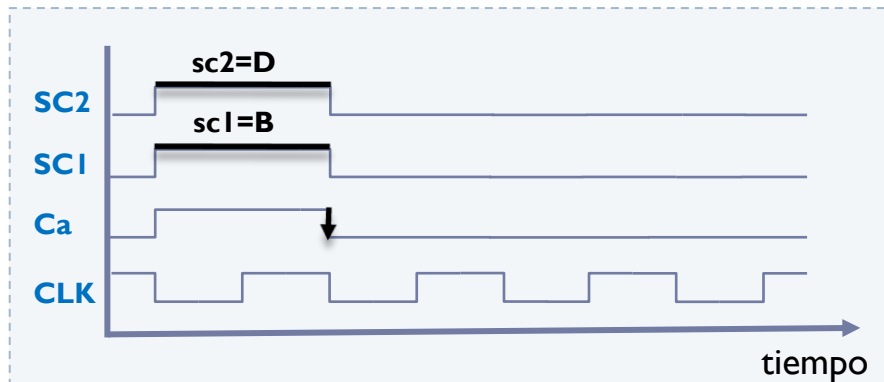
Ejemplo de operación elemental de *procesamiento*



► 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.
 - Hacer lo que la circuitería no deja.

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**
 - $R1 \leftarrow R2 + R3$



$Reg \leftarrow Reg$

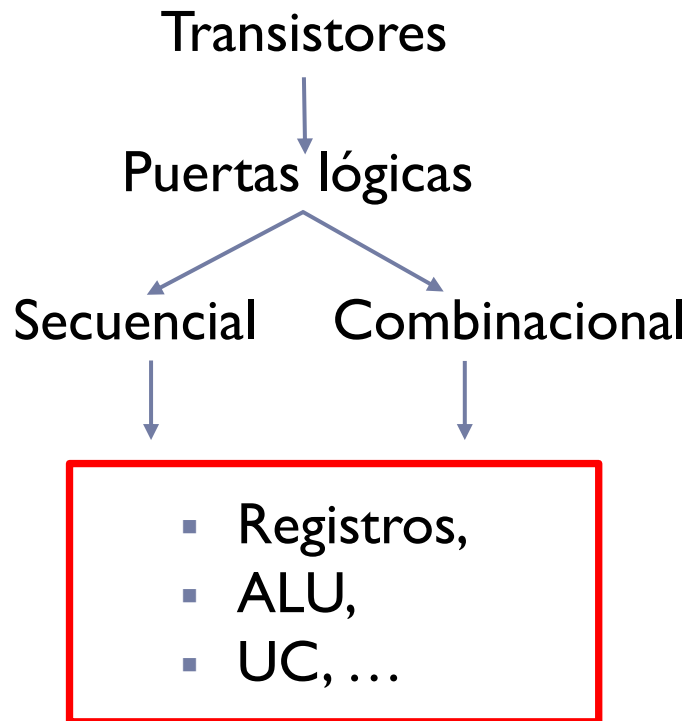


$Reg \leftarrow \varphi(Reg, Reg)$

De transistores... ...a operaciones elementales

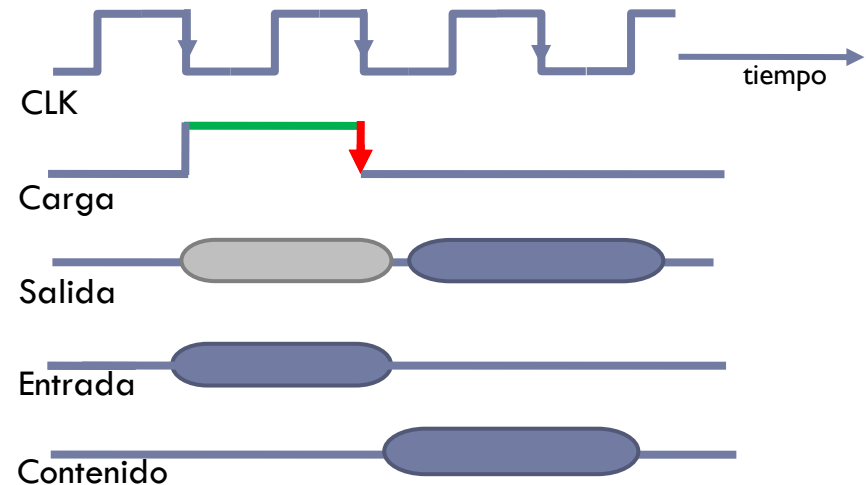
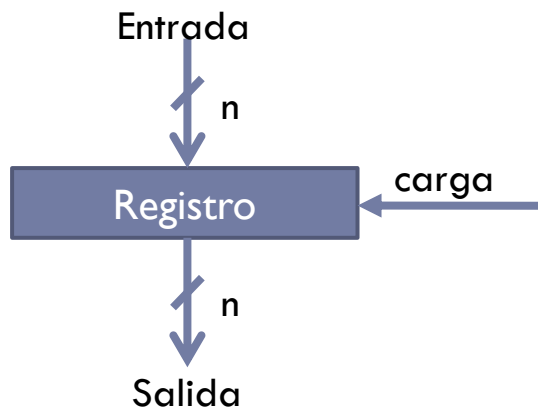
Recordatorio

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

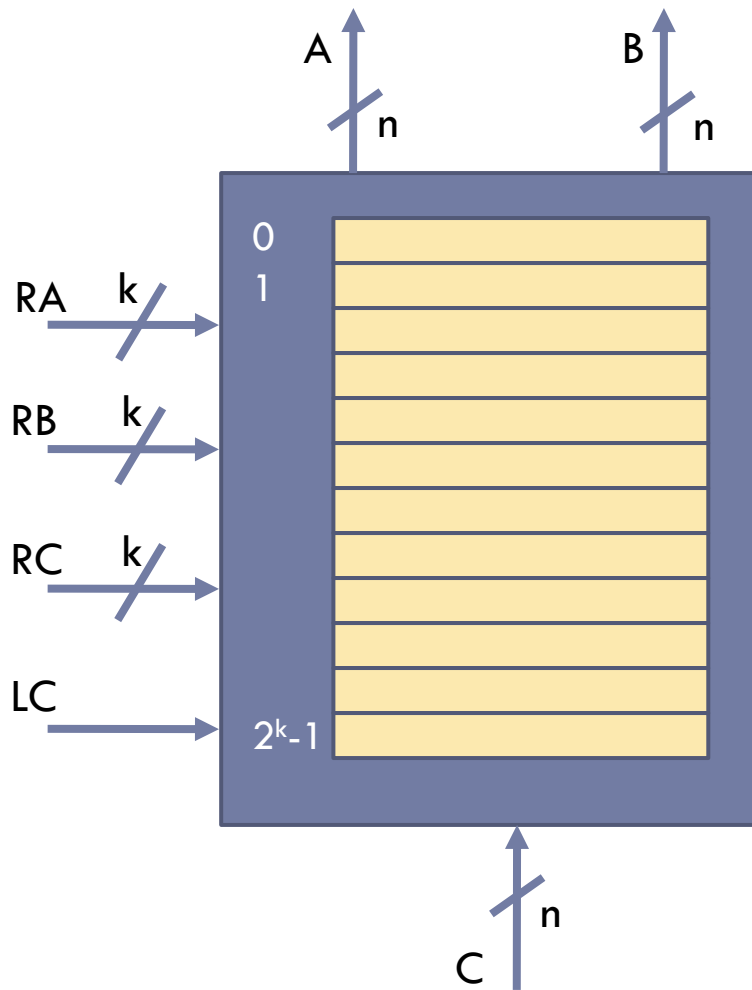


Registro

- ▶ 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

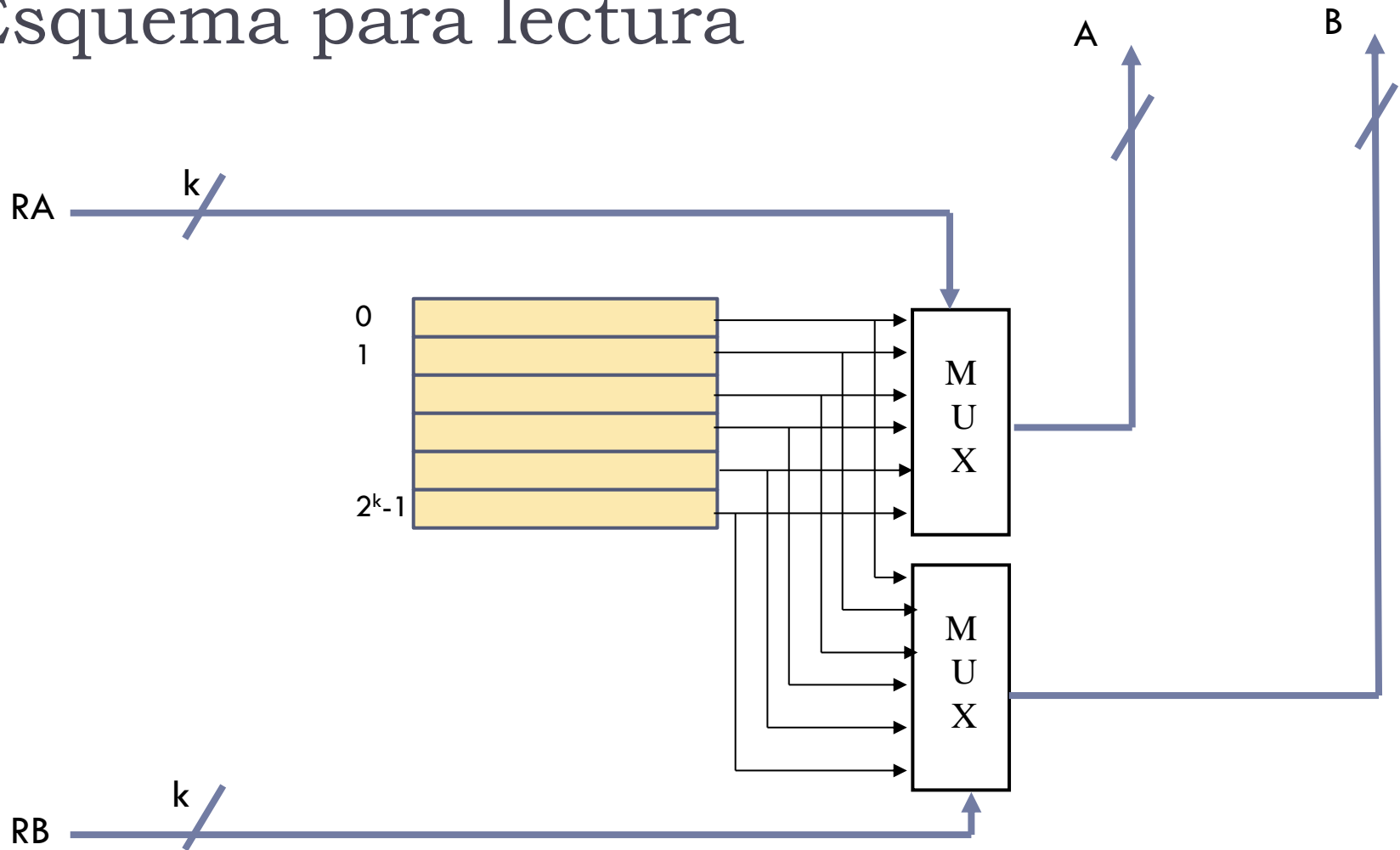


Banco de registros (BR)



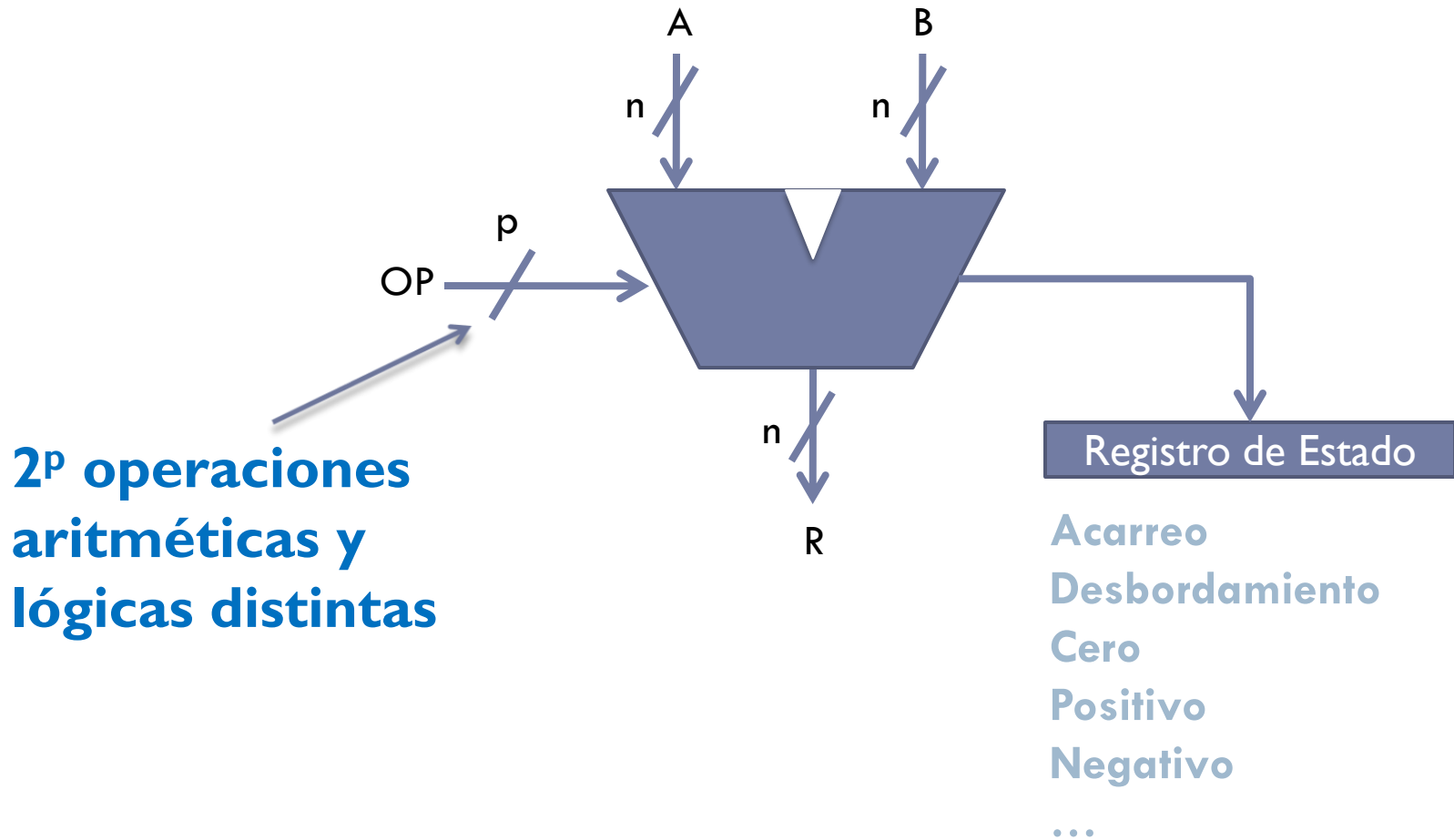
- ▶ Agrupación de registros.
 - ▶ Típicamente un número de registros potencia de 2.
 - ▶ n registros $\rightarrow \log_2 n$ bits para seleccionar cada registro
 - ▶ k bits de selección $\rightarrow 2^k$ registros
 - ▶ Ej.: con 32 registros, $k=5$
 - ▶ Elemento fundamental de almacenamiento.
 - ▶ Acceso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

Esquema para lectura

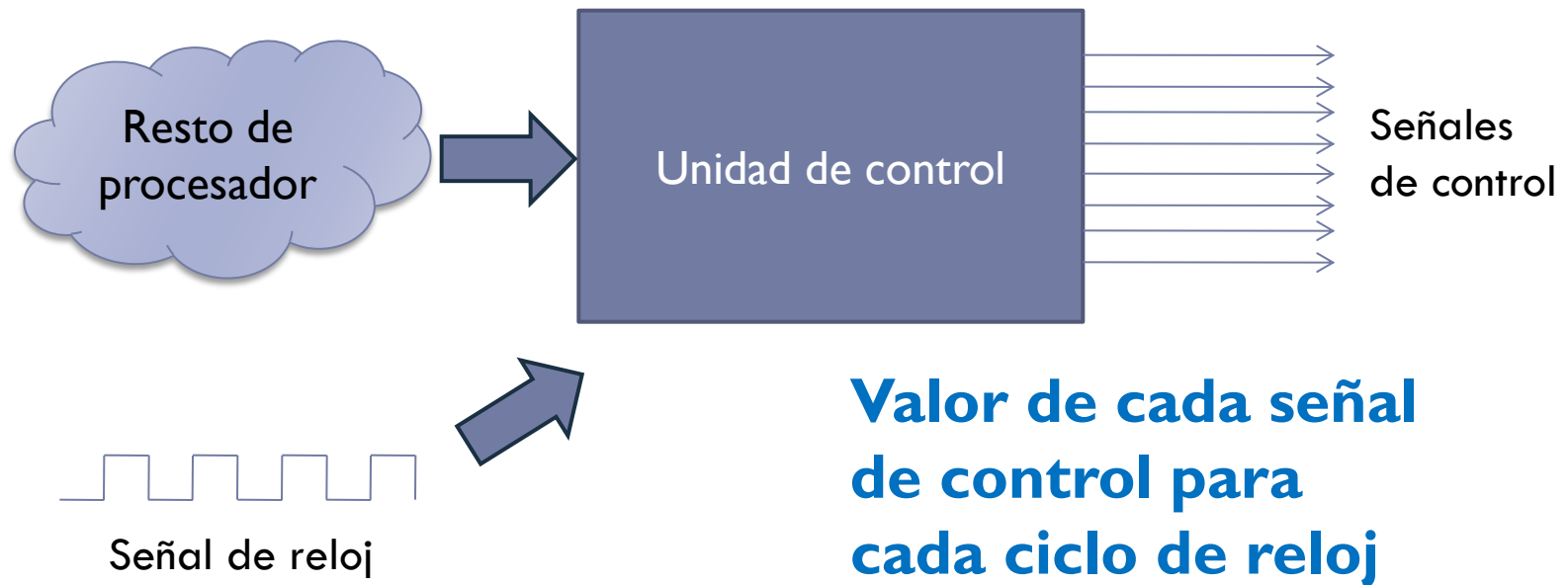


¿Qué valor tiene que tener RA para sacar por A el contenido del registro 14?

Unidad aritmético lógica (ALU)



Unidad de control (UC)



Contenido

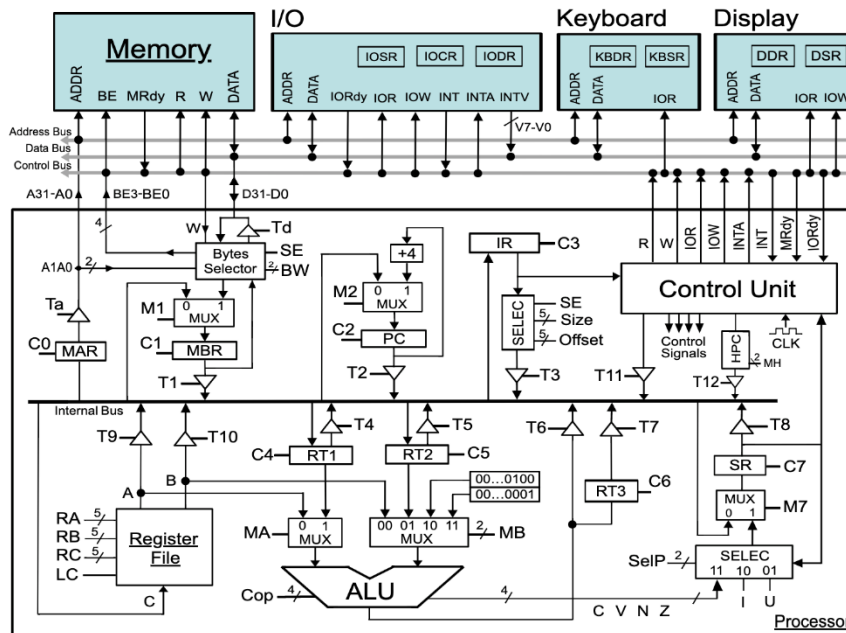
1. Elementos de un
2. Organización del procesador
3. La unidad de control
4. Ejecución de instrucciones
5. Diseño de la unidad de control
6. Modos de ejecución
7. Interrupciones
8. Arranque de un computador
9. Prestaciones y paralelismo

- 1) Motivación y objetivos
- 2) Funcionamiento básico de la unidad de control
- 3) Señales de control y operaciones elementales
- 4) Introducción al procesador elemental

Estructura de un procesador elemental & Simulador WepSIM

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

► Elemental Processor (E.P.):

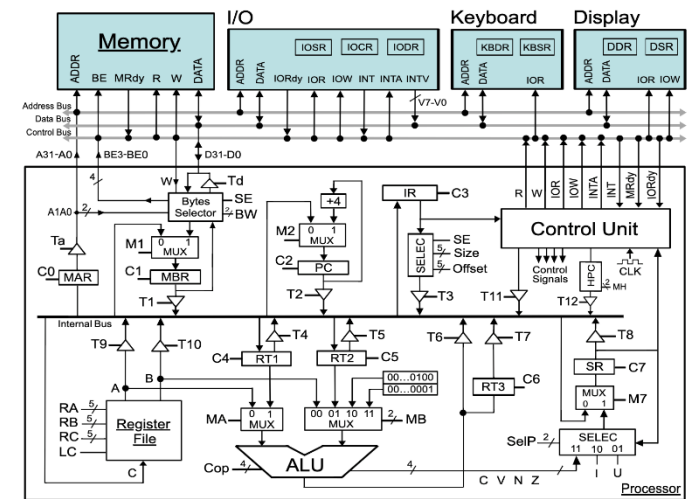


► WepSIM simula el E.P.:

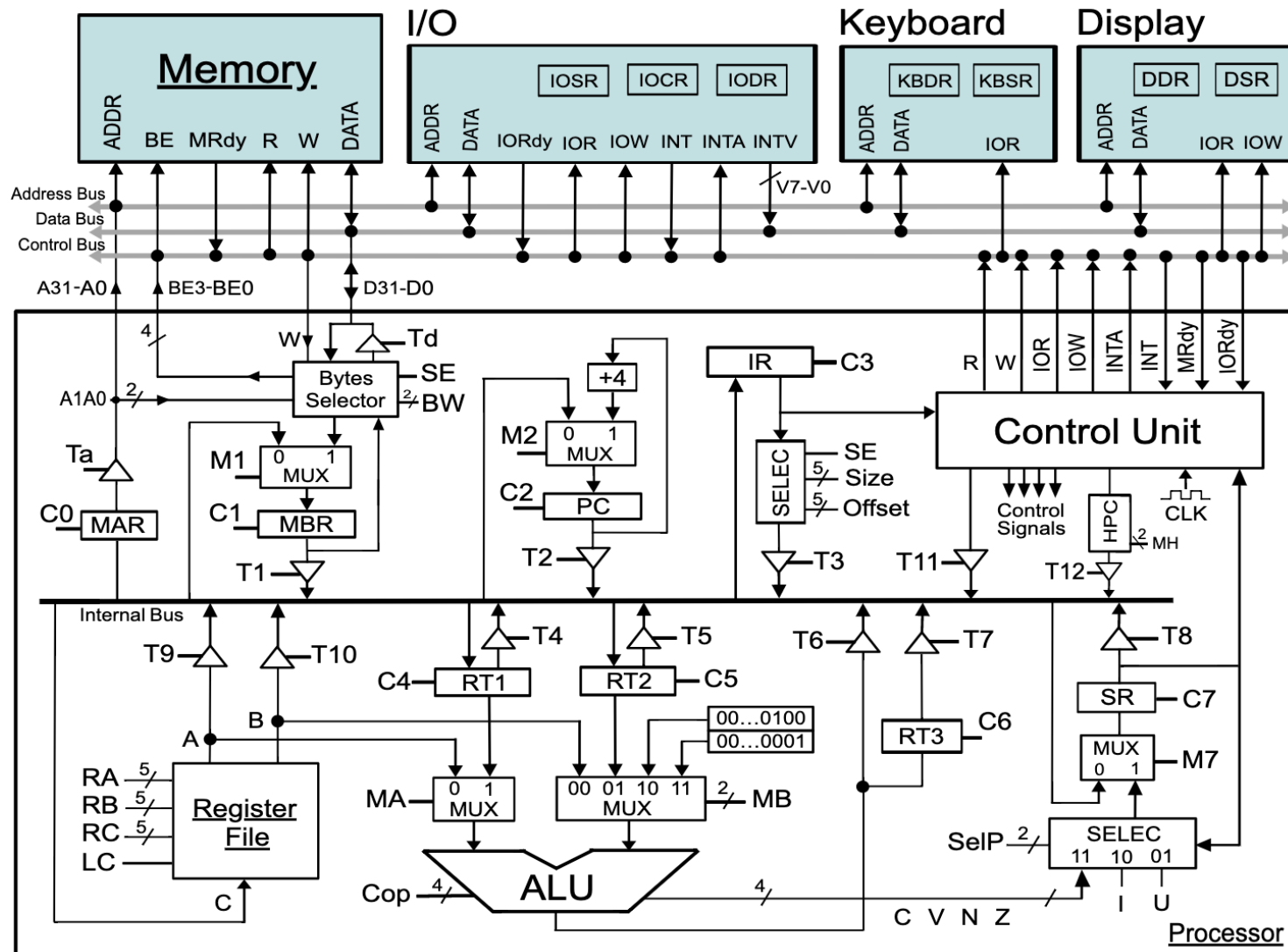
► <https://wepsim.github.io/wepsim/>

Principales características

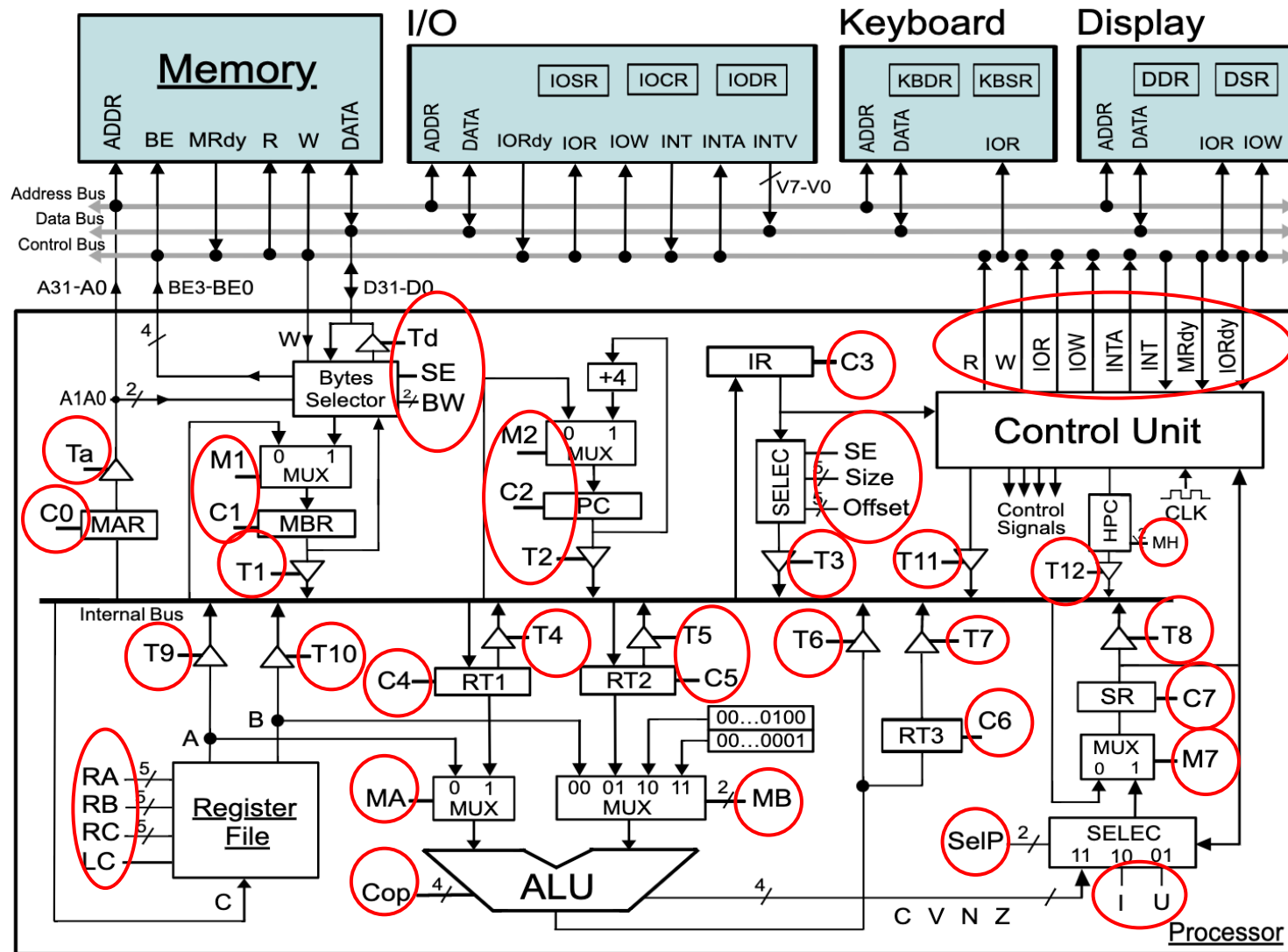
- ▶ **Computador elemental (EP):**
 - ▶ Computador de 32 bits
 - ▶ La memoria principal:
 - ▶ Se direcciona a nivel de byte
 - ▶ Operaciones de lectura y escritura de 1 ciclo
 - ▶ Banco de 32 registros (R0...R31) de 32 bits cada uno
 - ▶ Asumir para RISC-V: R0 = 0 y SP = R2
 - ▶ Asumir para MIPS: R0 = 0 y SP = R29
 - ▶ Registros de control (PC, IR, ...) de estado (SR) y temporales no visibles (RT1...RT3)
- ▶ **Simulador WepSIM implementa el EP:**
 - ▶ <https://wepsim.github.io/wepsim/>



Estructura de un computador elemental



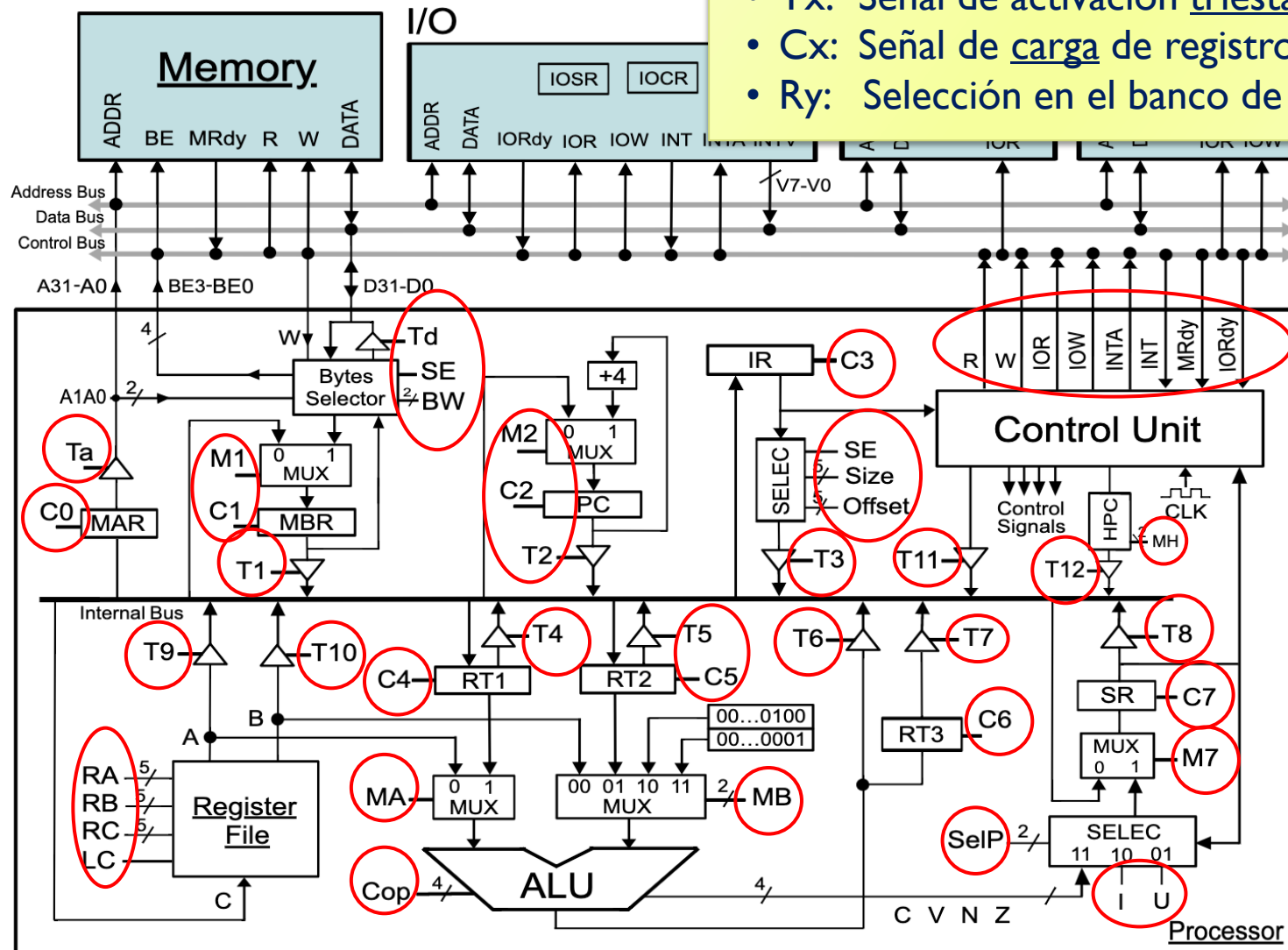
Señales de control



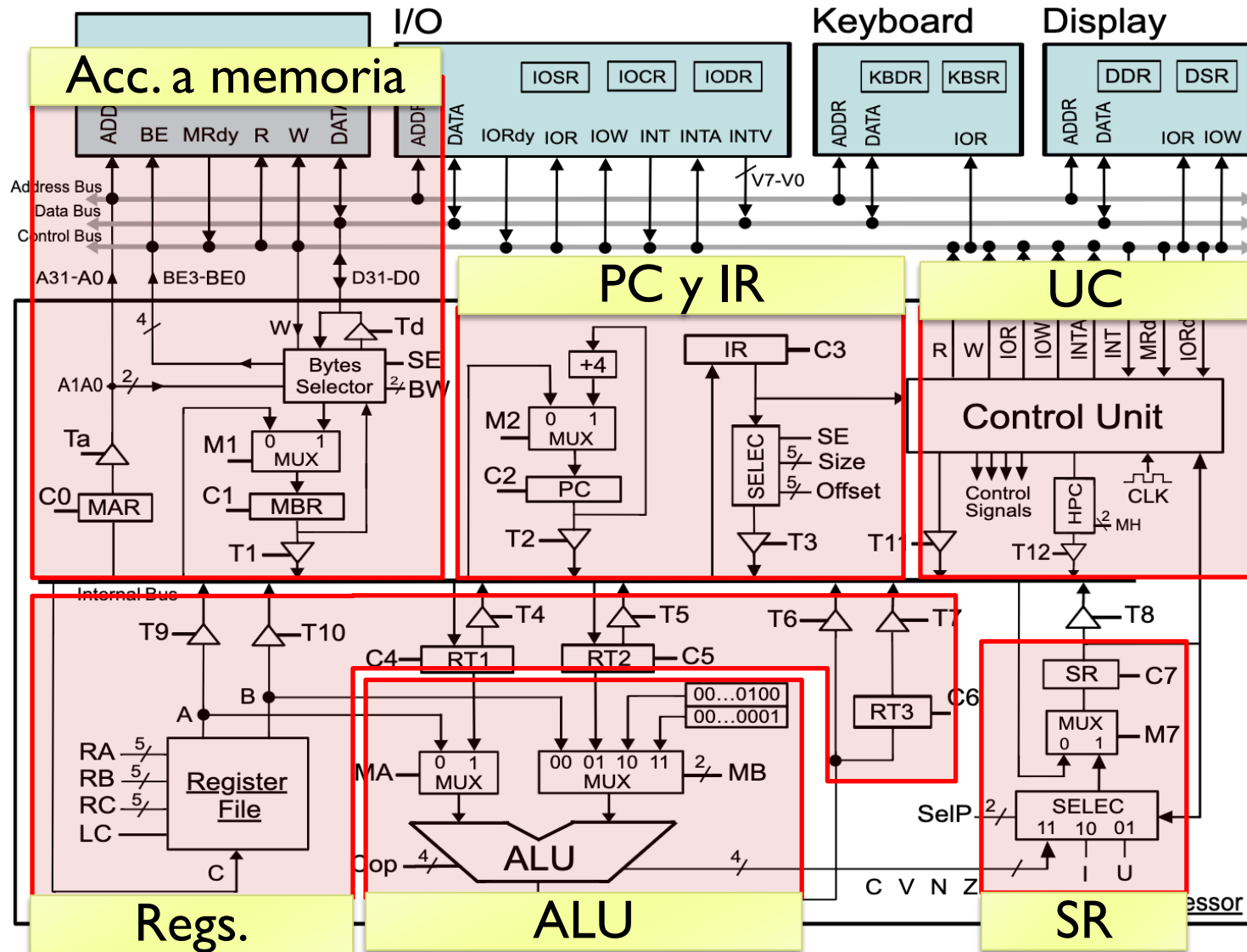
Señales de control

Nomenclatura general:

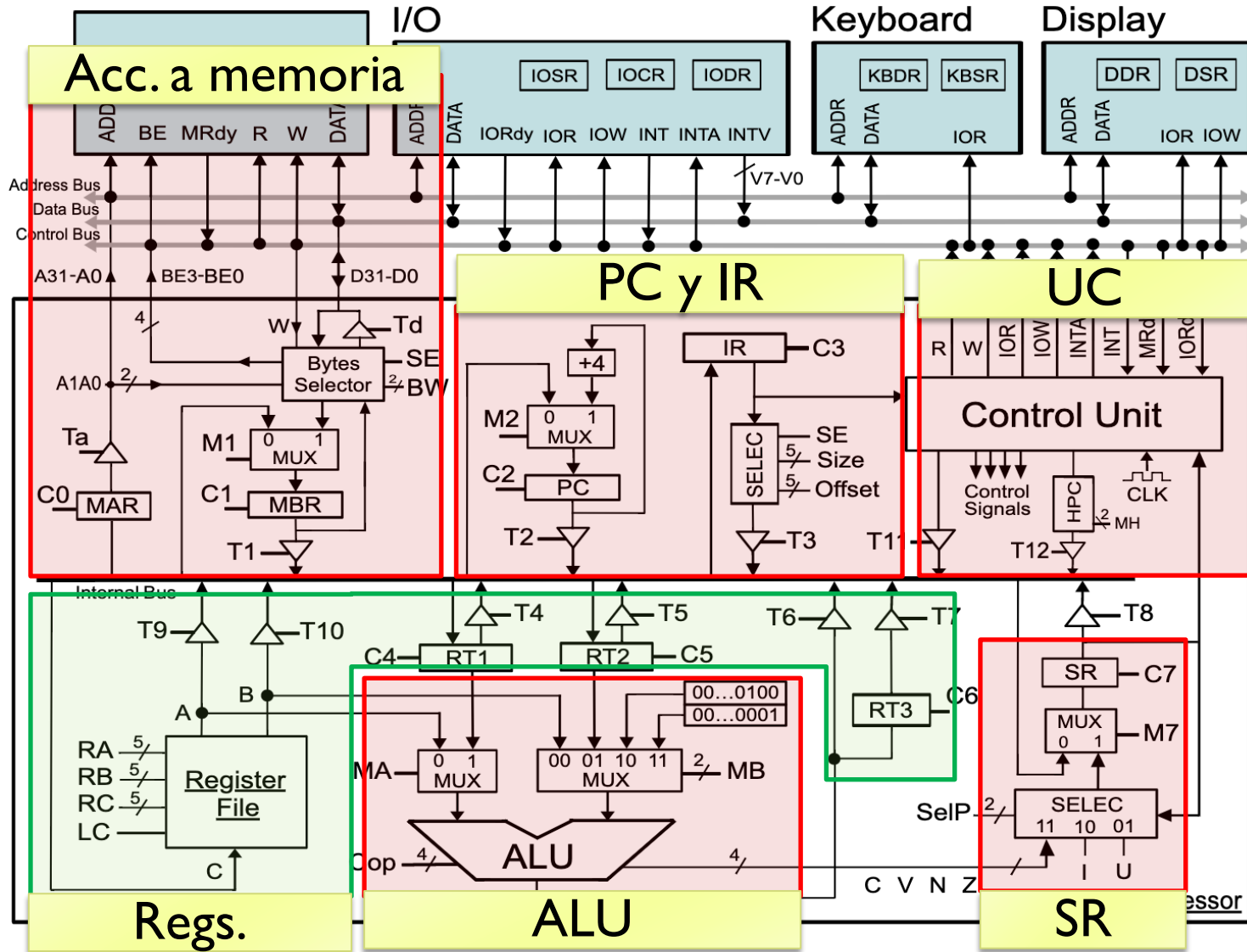
- Mx: Selección en multiplexor
- Tx: Señal de activación triestado
- Cx: Señal de carga de registro
- Ry: Selección en el banco de registros



Procesador elemental: señales de control



Procesador elemental: señales de control



Registros

- ▶ Registros visibles al programador/a:

- ▶ Banco de registros (Ej.: RISC-V: t0, t1, etc.)

- ▶ Registros de control:

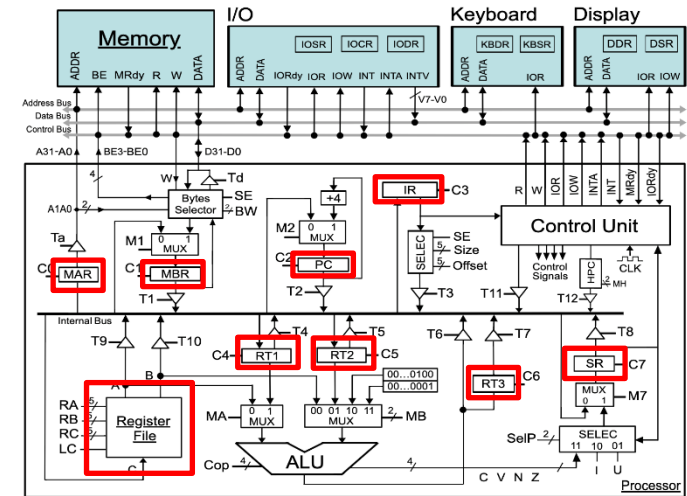
- ▶ PC: contador de programa o *program counter*
- ▶ IR: registro de instrucción o *instruction register*
- ▶ SP: puntero de pila o *stack pointer* (en el banco de registros)
- ▶ MAR: registro de direcciones de memoria o *memory address register*
- ▶ MBR: registro de datos de memoria o *memory buffer register*

- ▶ Registro de estado:

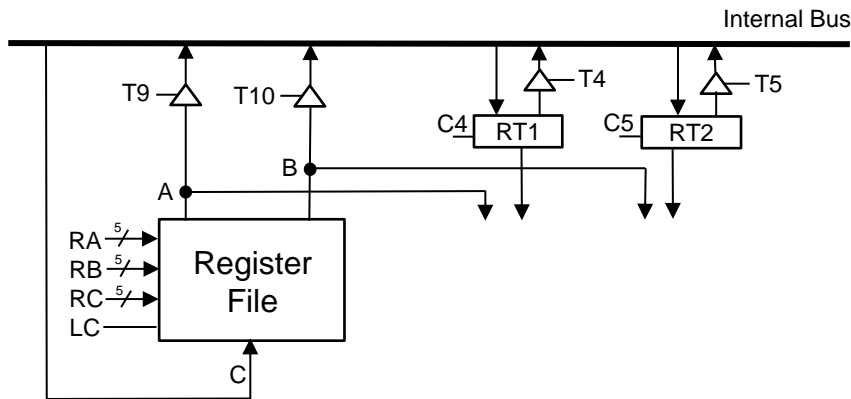
- ▶ SR: registro de estado o *status register*

- ▶ Registros no visibles al programador/a:

- ▶ RT1, RT2 y RT3: registros temporales internos de la CPU



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

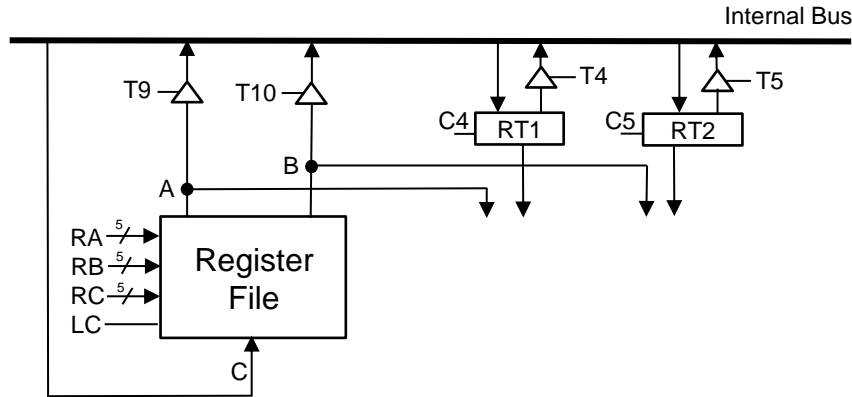
- RA – salida de registro RA por A
- RB – salida de registro RB por B
- RC – entrada por C al registro RC
- LC – activa la escritura para RC
- T9 – copia de A al bus interno
- T10 – copia de B al bus interno

► registros RT1 y RT2

- 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

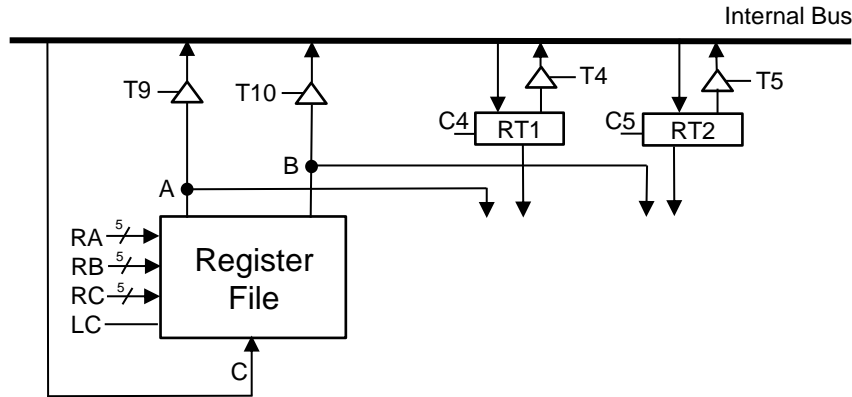
operaciones elementales en registros



► **SWAP R1 R2**

Ejemplo

operaciones elementales en registros

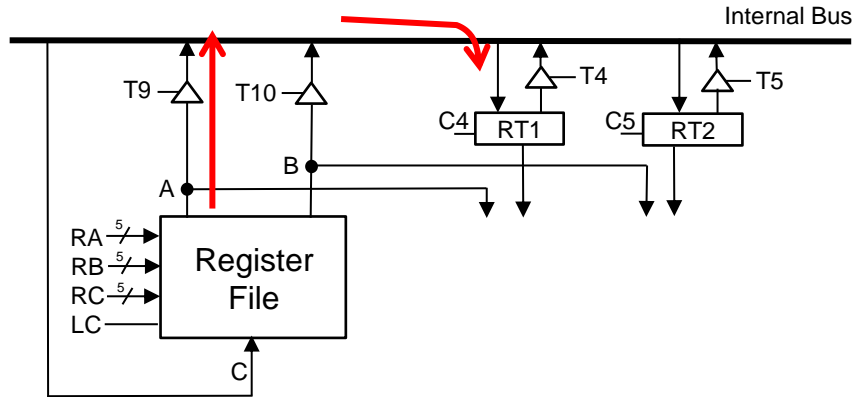


► **SWAP R1 R2**

O. Elemental	Señales

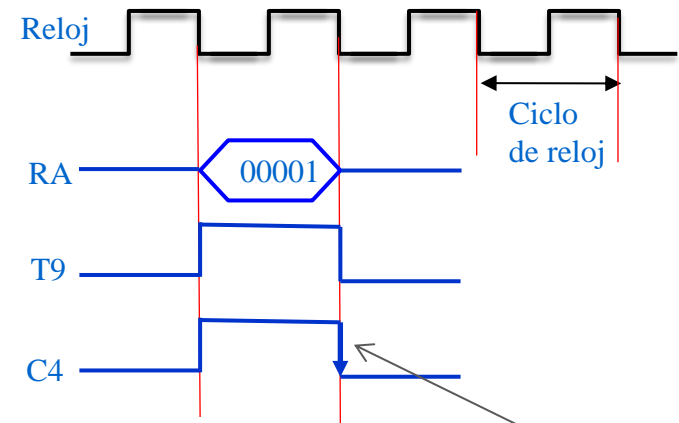
Ejemplo

operaciones elementales en registros



► SWAP R1 R2

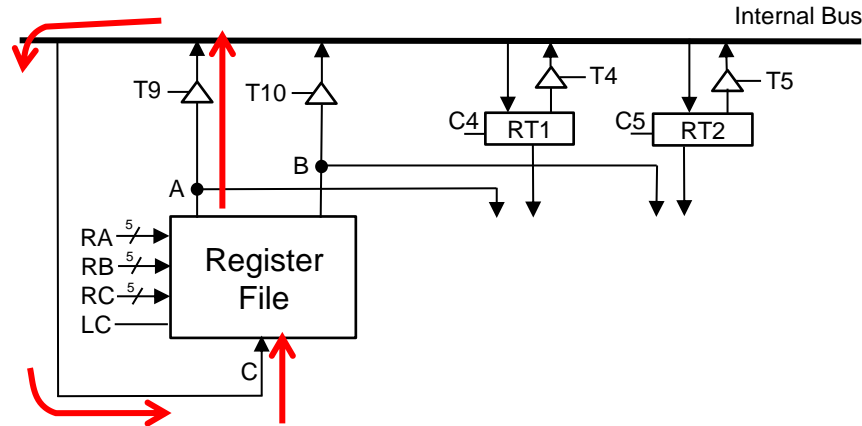
O. Elemental	Señales
$RT1 \leftarrow R1$	$RA=00001, T9, C4$



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

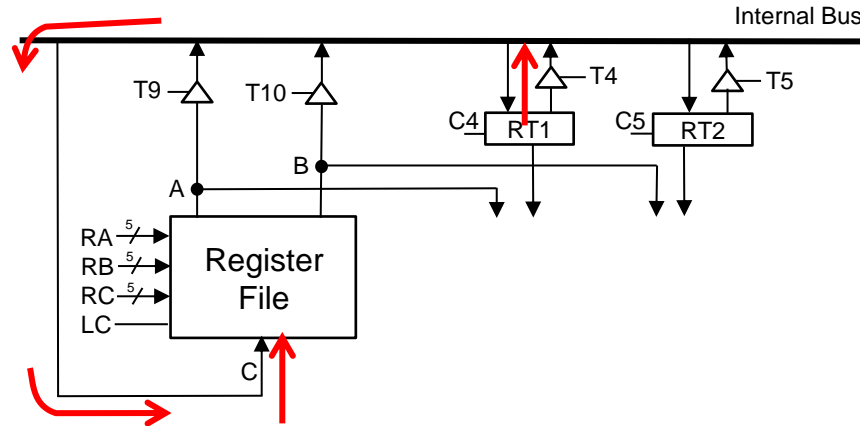


► **SWAP R1 R2**

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

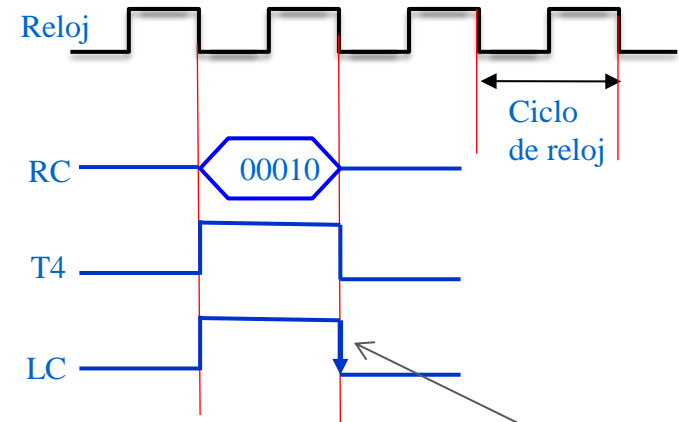
Ejemplo

operaciones elementales en registros



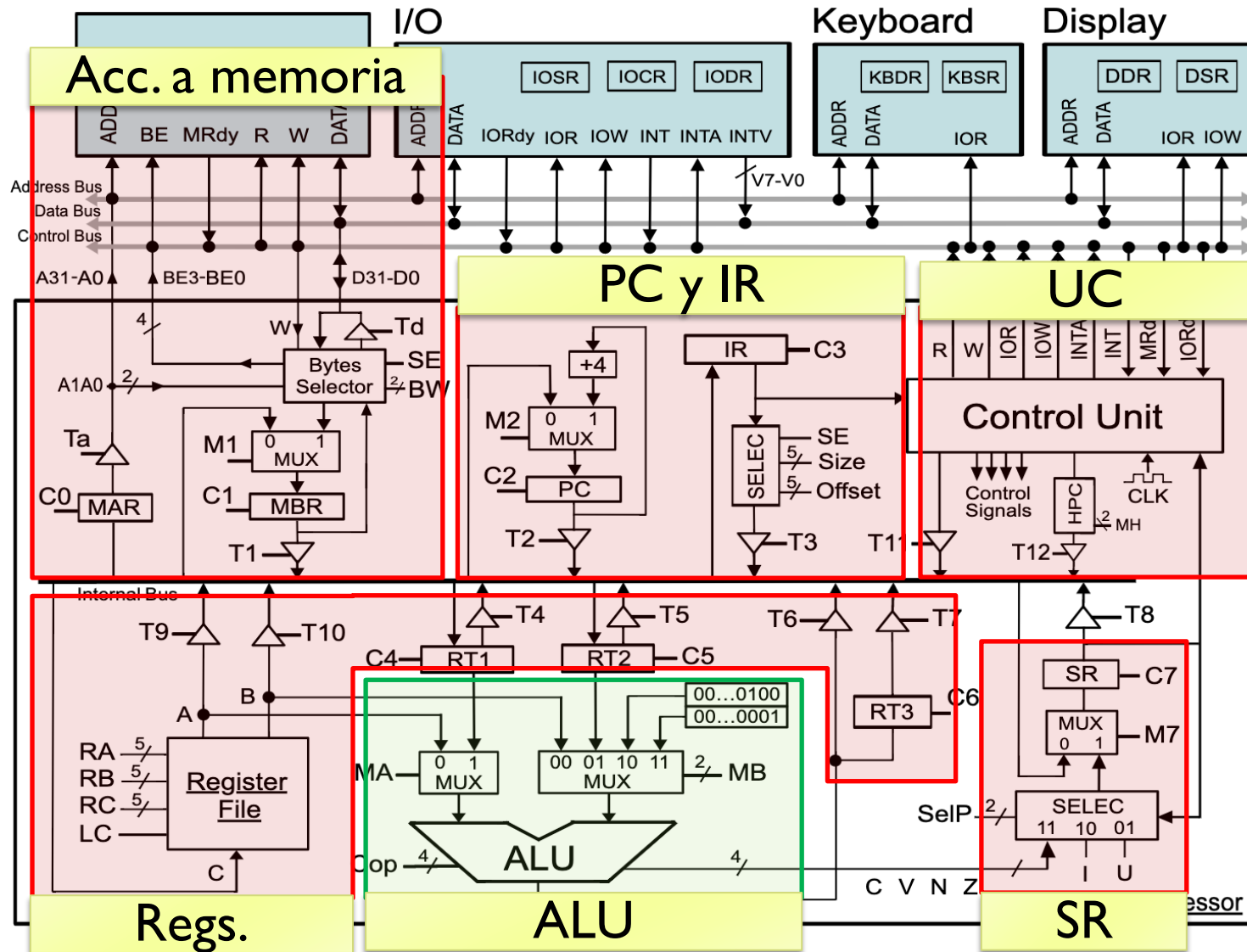
► SWAP R1 R2

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

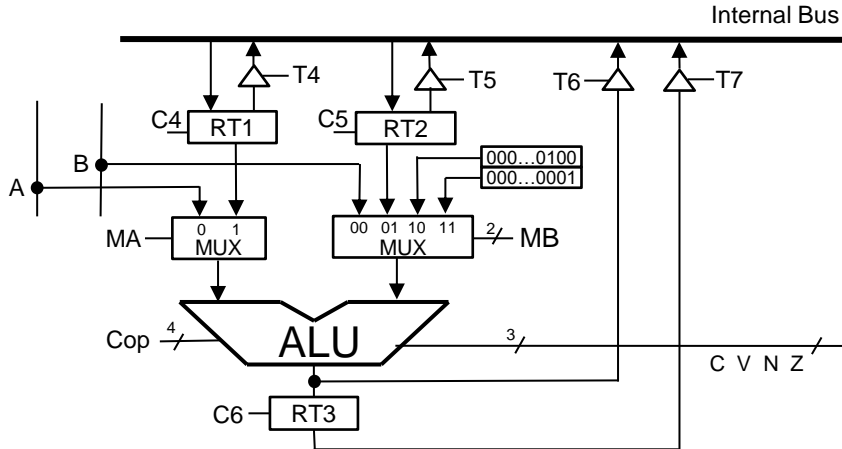


La carga del dato se realiza en R2 en el flanco de bajada. Estará disponible en R2 durante el siguiente ciclo.

Procesador elemental: señales de control



Señales de control

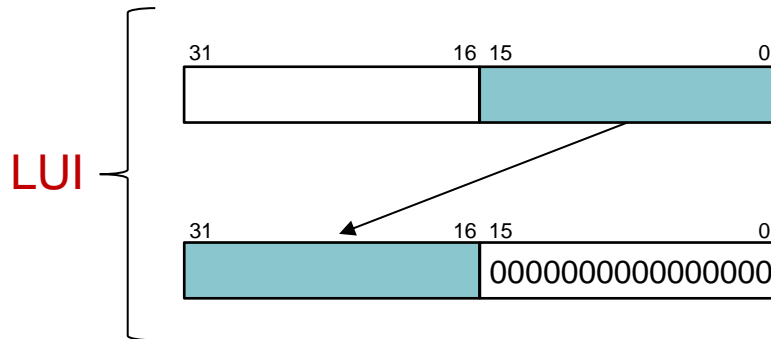
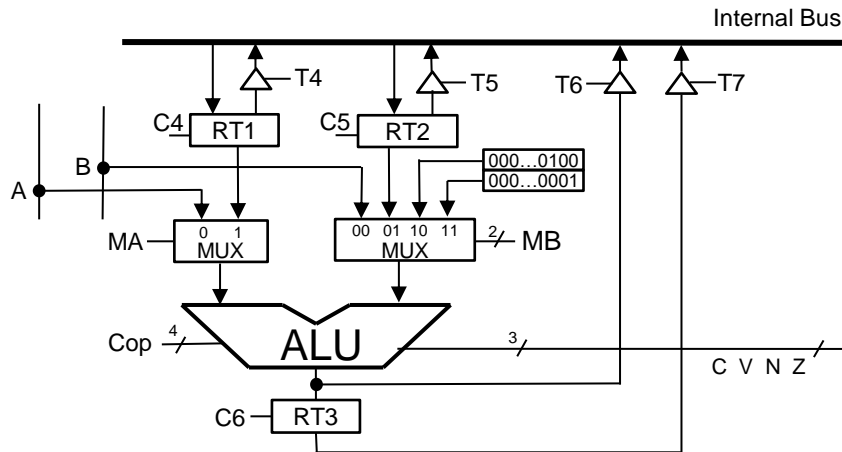


► **ALU**

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

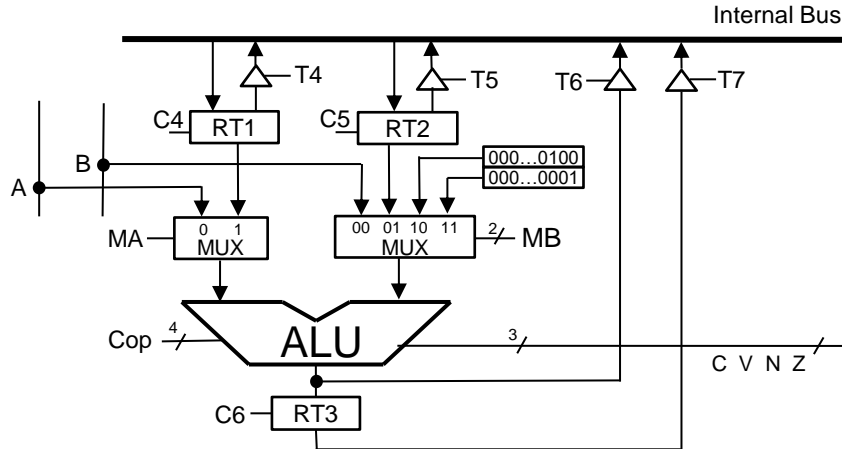
Cop (Cop ₃ -Cop ₀)	Operación
0000	NOP
0001	A and B
0010	A or B
0011	not (A)
0100	A xor B
0101	Shift Right Logical (A) B= number of bits to shift
0110	Shift Right Arithmetic (A) B= number of bits to shift
0111	Shift left (A) B= number of bits to shift
1000	Rotate Right (A) B= number of bits to rotate
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

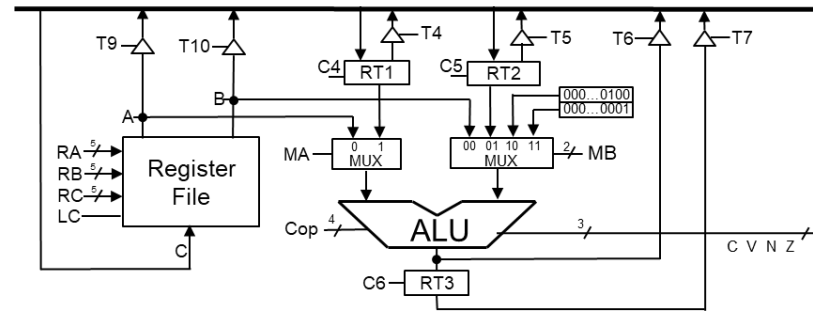


Resultado	C	V	N	Z
Resultado positivo (0 se considera +)	0	0	0	0
Resultado == 0	0	0	0	1
Resultado negativo	0	0	1	0
Desbordamiento de la operación	0	1	0	0
División por cero	0	1	0	1
Acarreo en el bit 32	1	0	0	0

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

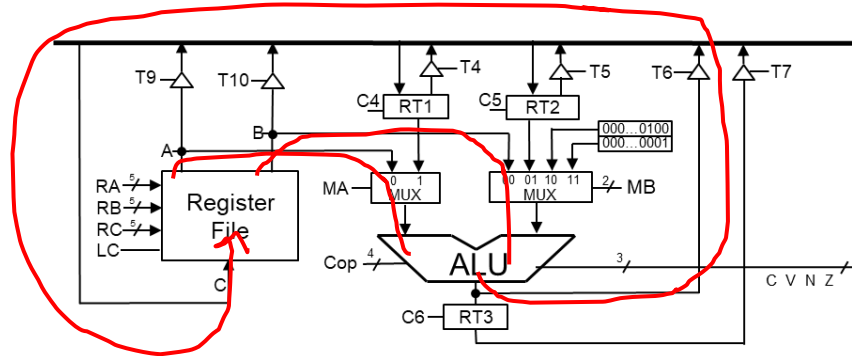


► ADD R3 RI R2

O. Elemental	Señales

Ejemplo

operaciones elementales en ALU

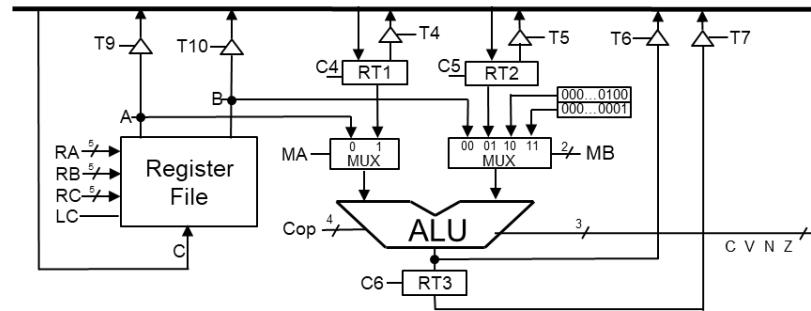


► ADD R3 RI R2

O. Elemental	Señales

Ejemplo

operaciones elementales en ALU

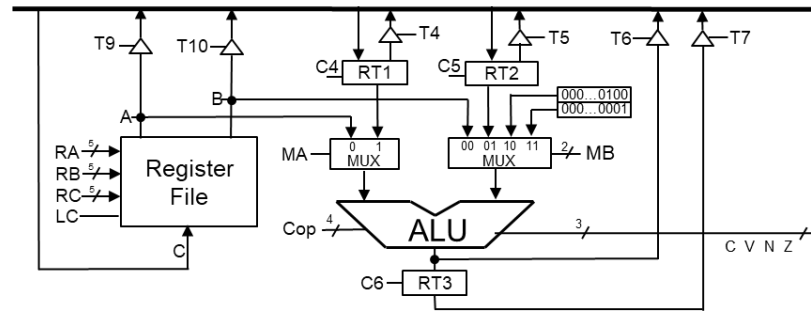


► ADD R3 R1 R2

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

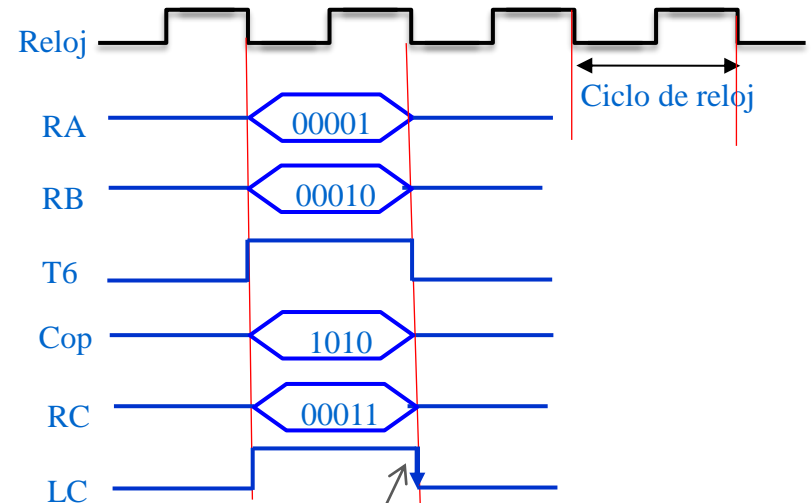
Ejemplo

operaciones elementales en ALU



► ADD R3 R1 R2

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



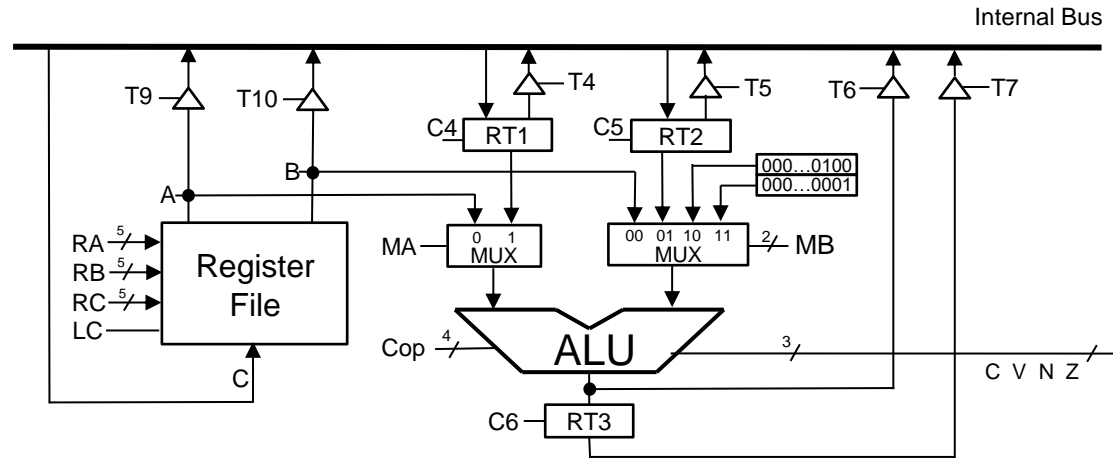
Resto de señales a 0.

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

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

Ejemplo

operaciones elementales en ALU



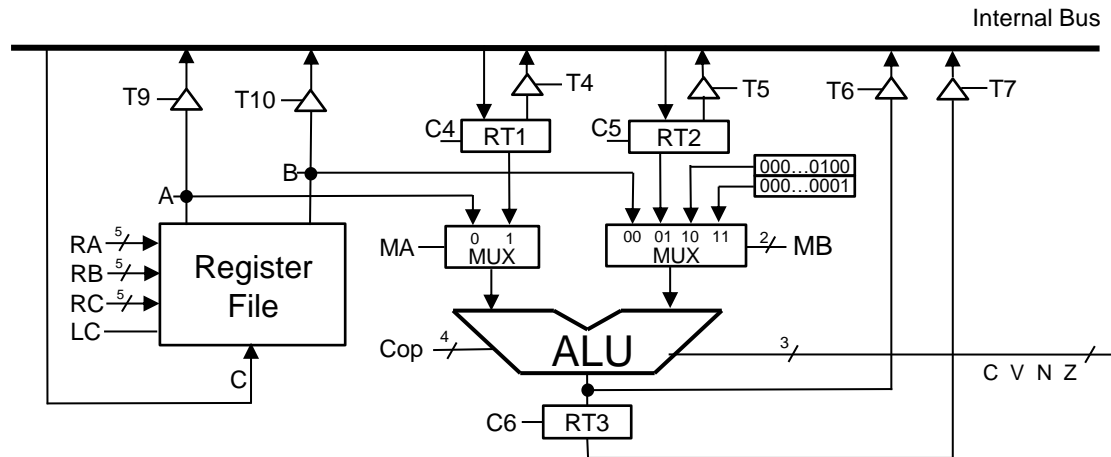
► SWAP R1 R2

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}

Ejemplo

operaciones elementales en ALU



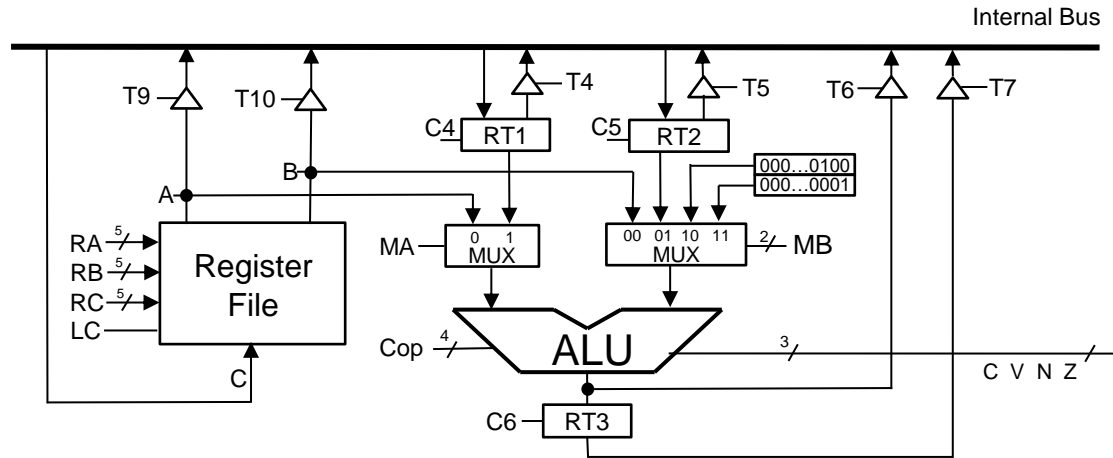
► SWAP R1 R2

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	
$R1 \leftarrow R1 \wedge R2$	$R1 \leftarrow (R1 \wedge R2)$
$R2 \leftarrow R1 \wedge R2$	$R2 \leftarrow (R1 \wedge R2) \wedge R2$
$R1 \leftarrow R1 \wedge R2$	$R1 \leftarrow (R1 \wedge R2) \wedge R1$

operaciones elementales en ALU



► **SWAP R1 R2**

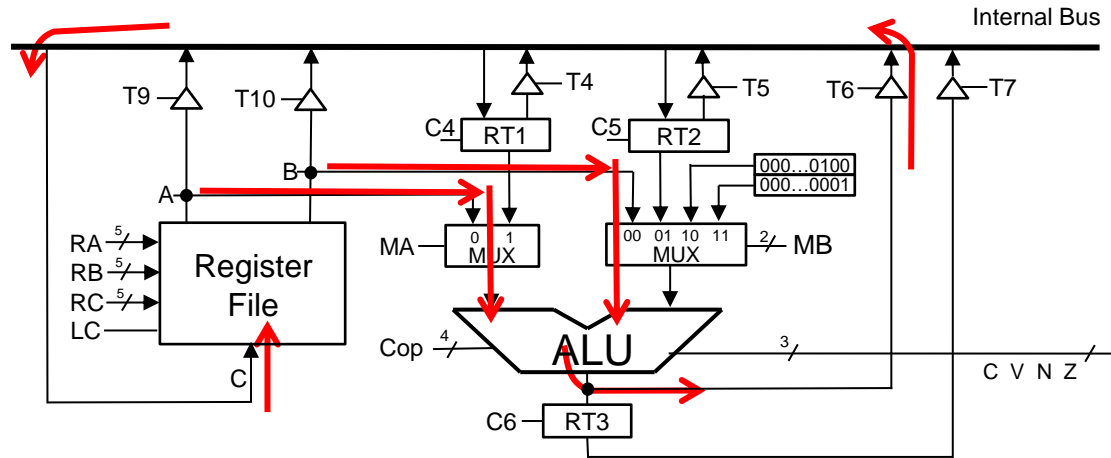
O. Elemental	Señales
$RT1 \leftarrow R1$	RA=1, T9, C4
$R1 \leftarrow R2$	RA=2, T9, RC=1, LC
$R2 \leftarrow RT1$	T4, RC=2, LC

► **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

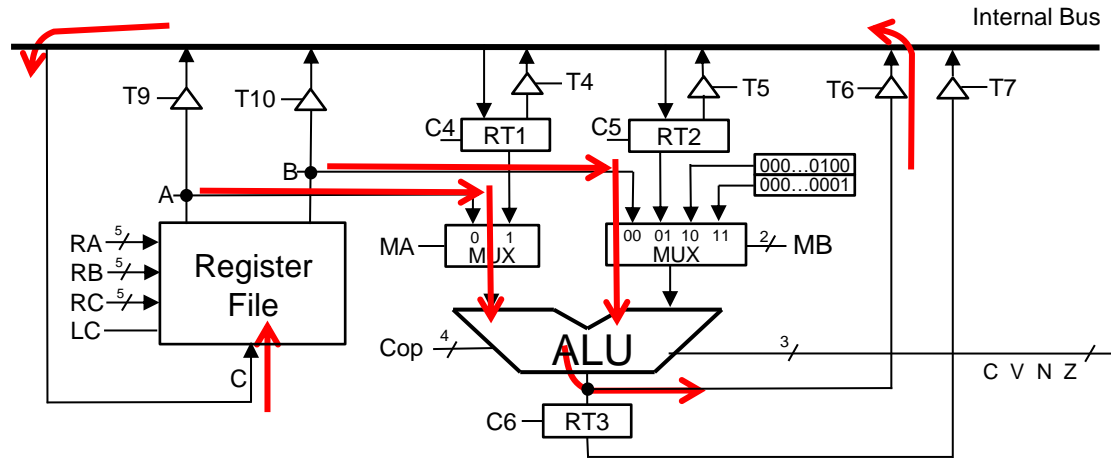
O. Elemental	Señales
$RT1 \leftarrow R1$	RA=1, T9, C4
$R1 \leftarrow R2$	RA=2, T9, RC=1, LC
$R2 \leftarrow RT1$	T4, RC=2, LC

► 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

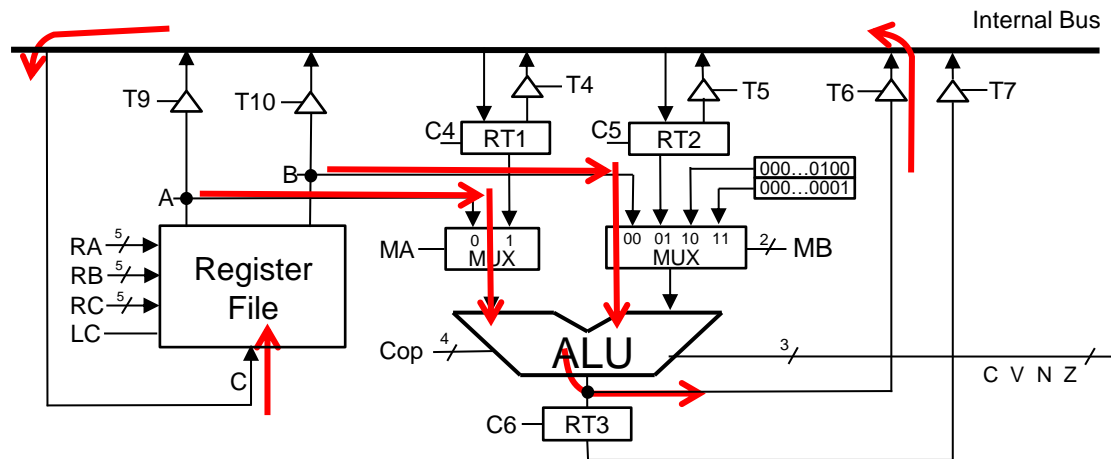
O. Elemental	Señales
$RT1 \leftarrow R1$	RA=1, T9, C4
$R1 \leftarrow R2$	RA=2, T9, RC=1, LC
$R2 \leftarrow RT1$	T4, RC=2, LC

► SWAP R1, R2 sin R_{tmp}

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



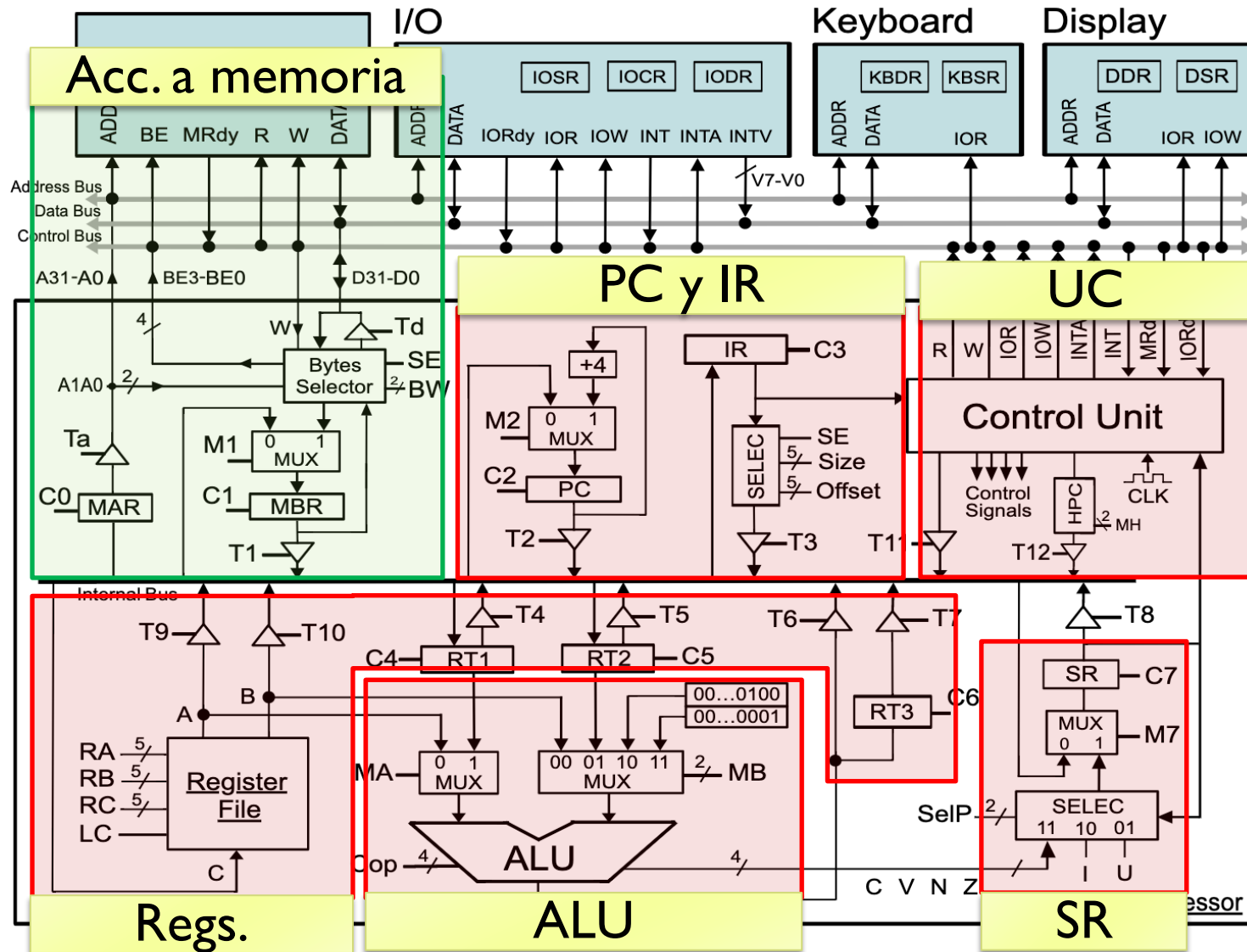
► SWAP R1 R2

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

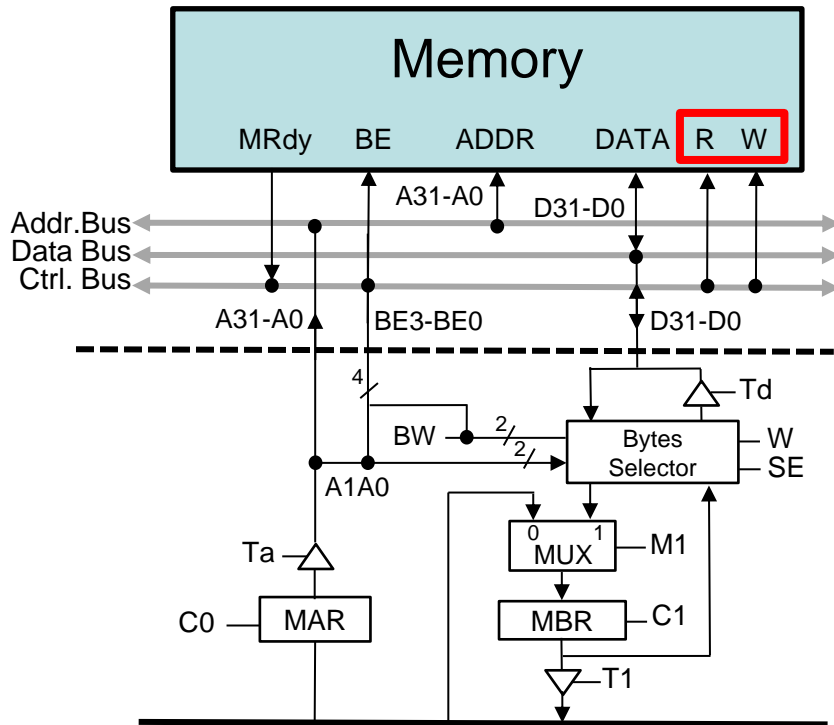
Procesador elemental: señales de control



Señales de control

▶ Memoria principal

- ▶ R – lectura
- ▶ W – escritura



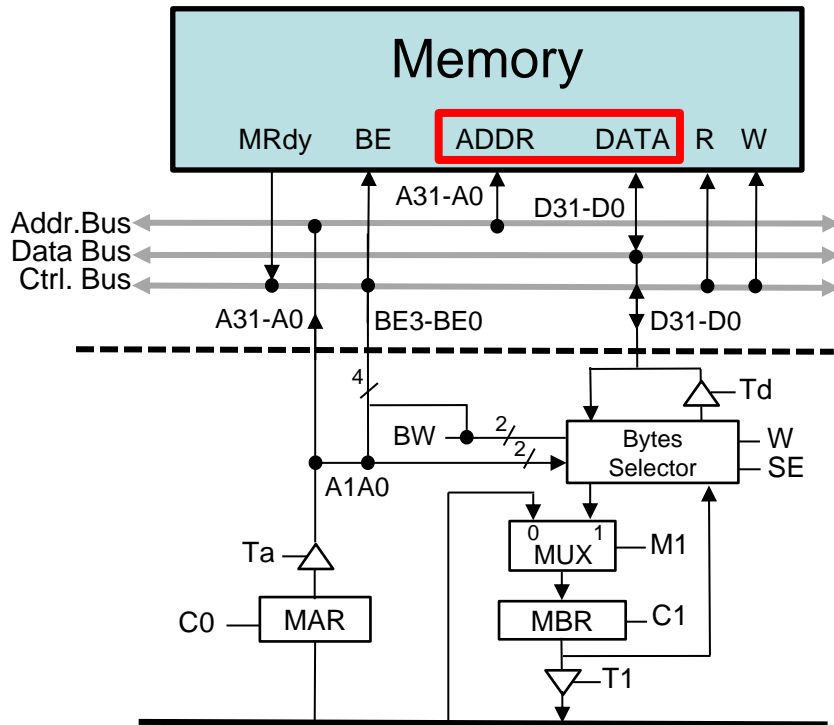
Nomenclatura:

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

Señales de control

▶ Memoria principal

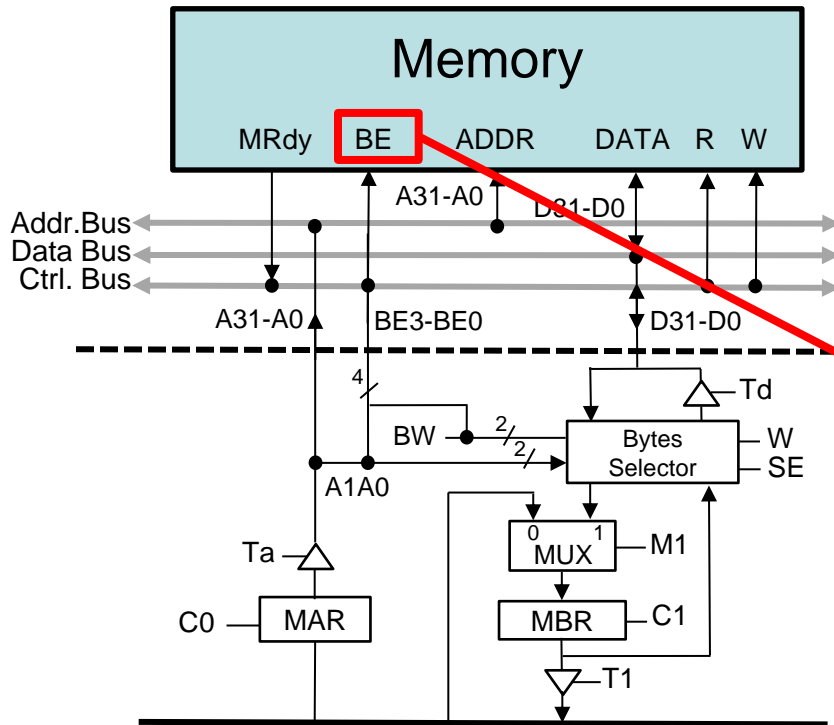
- ▶ R – lectura
- ▶ W – escritura
- ▶ DATA – datos desde/a memoria
- ▶ ADDR – dirección



Nomenclatura:

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

Señales de control



Nomenclatura:

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

Memoria principal

- ▶ R – lectura
- ▶ W – escritura
- ▶ DATA – datos desde/a memoria
- ▶ ADDR – dirección
- ▶ $BE3-BE0 = A1A0 + BW$
 - ▶ Tamaño acceso (byte, palabra, $\frac{1}{2}$ palabra)

BW: byte selector

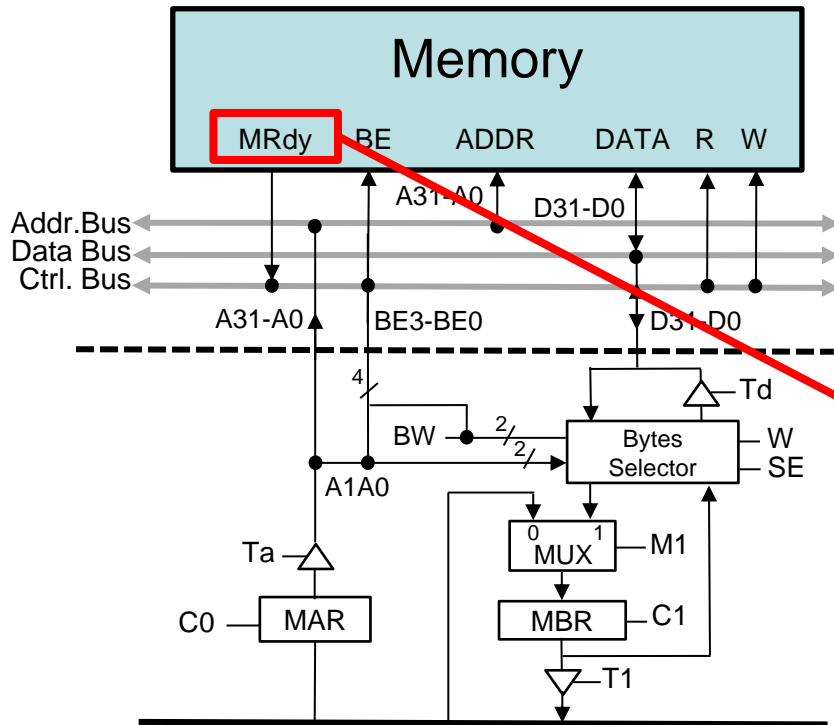
Selecciona qué bytes se almacenan en el MBR durante la lectura y se copian en el bus durante la escritura.

- ▶ **BW=0:** acceso a **byte**
- ▶ **BW=01:** acceso a **media palabra**
- ▶ **BW=11:** acceso a **palabra**

SE: extensión de signo

- ▶ **0:** no extiende el signo en acceso más pequeño que palabra
- ▶ **1:** extiende el signo en acceso más pequeño que palabra

Señales de control



Nomenclatura:

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

Memoria principal

- ▶ R – lectura
- ▶ W – escritura
- ▶ DATA – datos desde/a memoria
- ▶ ADDR – dirección
- ▶ $BE3-BE0 = A1A0 + BW$
 - ▶ Tamaño acceso (byte, palabra, $\frac{1}{2}$ palabra)
- ▶ MRdy – operación finalizada
[solo para asíncrono]

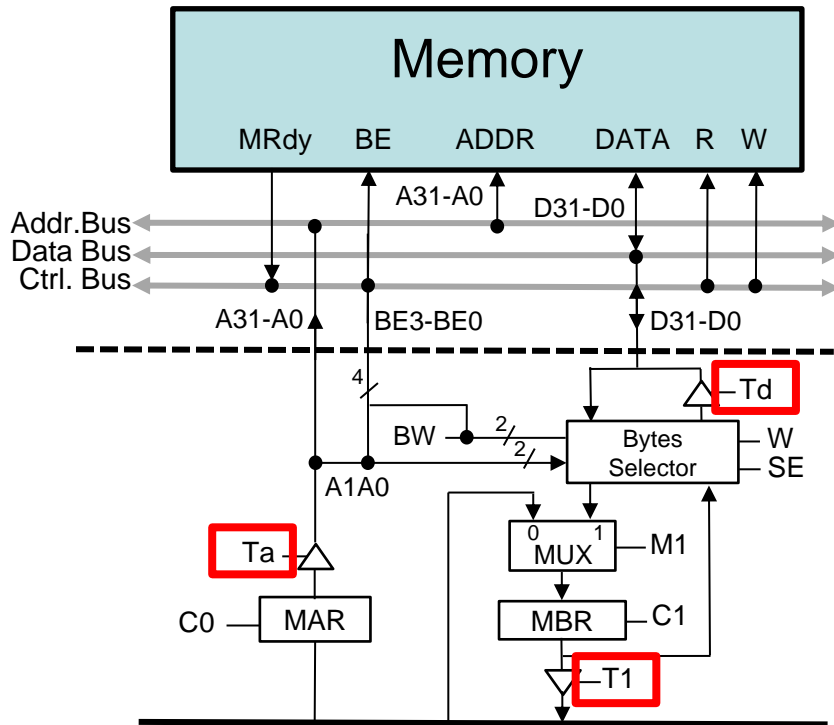
Síncrona:

- ▶ La memoria requiere un cierto número de ciclos para todas las operaciones.

Asíncrona:

- ▶ Número no fijo de ciclos de reloj para operaciones de memoria.
- ▶ La memoria indica cuándo finaliza la operación.

Señales de control



Nomenclatura:

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

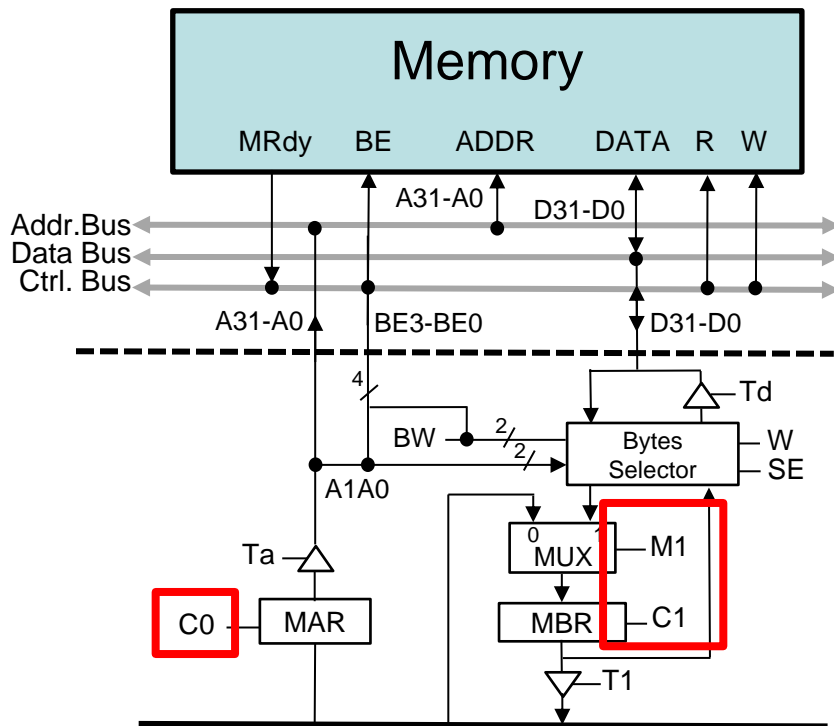
▶ Memoria principal

- ▶ R – lectura
- ▶ W – escritura
- ▶ DATA – datos desde/a memoria
- ▶ ADDR – dirección
- ▶ $BE3-BE0 = A1A0 + BW$
 - ▶ Tamaño acceso (byte, palabra, 1/2 palabra)
- ▶ MRdy – operación finalizada
[solo para asíncrono]

▶ MAR y MBR

- ▶ Ta – salida de MAR al bus de direcciones
- ▶ Td – salida de MBR al bus de datos
- ▶ T1 – salida de MBR al bus interno

Señales de control



Nomenclatura:

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

▶ Memoria principal

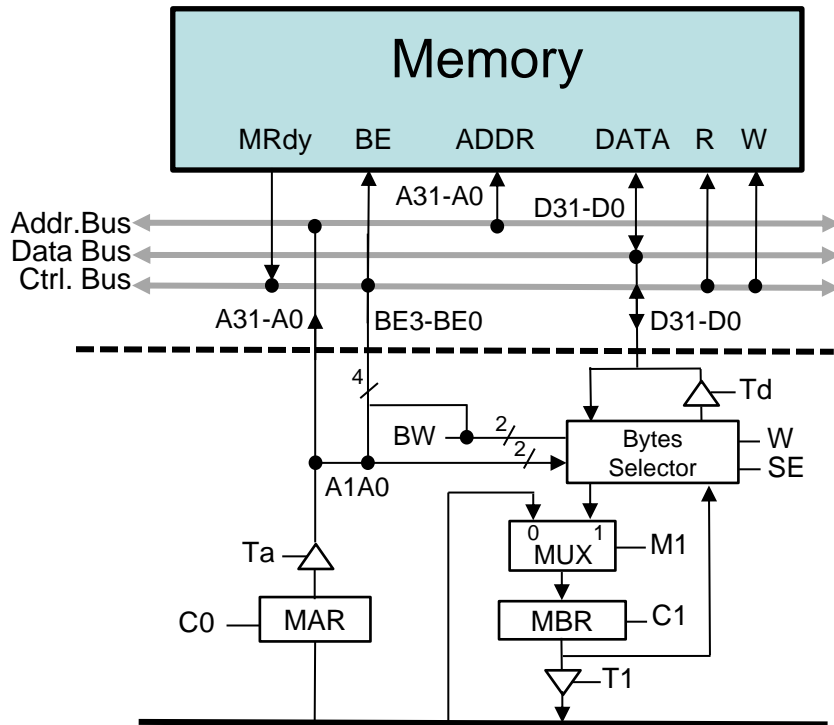
- ▶ R – lectura
- ▶ W – escritura
- ▶ DATA – datos desde/a memoria
- ▶ ADDR – dirección
- ▶ $BE3-BE0 = A1A0 + BW$
 - ▶ Tamaño acceso (byte, palabra, $\frac{1}{2}$ palabra)
- ▶ MRdy – operación finalizada
[solo para asíncrono]

▶ MAR y MBR

- ▶ Ta – salida de MAR al bus de direcciones
- ▶ Td – salida de MBR al bus de datos
- ▶ T1 – salida de MBR al bus interno
- ▶ M1 – selección para MBR: de memoria o bus interno
- ▶ C1 – del bus de datos al MBR
- ▶ C0 – del bus interno al MAR

Señales de control

summary



Nomenclatura:

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

▶ Memoria principal

- ▶ R – lectura
- ▶ W – escritura
- ▶ DATA – datos desde/a memoria
- ▶ ADDR – dirección
- ▶ $BE3-BE0 = A1A0 + BW$
 - ▶ Tamaño acceso (byte, palabra, $\frac{1}{2}$ palabra)
- ▶ MRdy – operación finalizada
[solo para asíncrono]

▶ MAR y MBR

- ▶ Ta – salida de MAR al bus de direcciones
- ▶ Td – salida de MBR al bus de datos
- ▶ T1 – salida de MBR al bus interno
- ▶ M1 – selección para MBR: de memoria o bus interno
- ▶ C1 – del bus de datos al MBR
- ▶ C0 – del bus interno al MAR

Señales BE (*Byte Enable*)

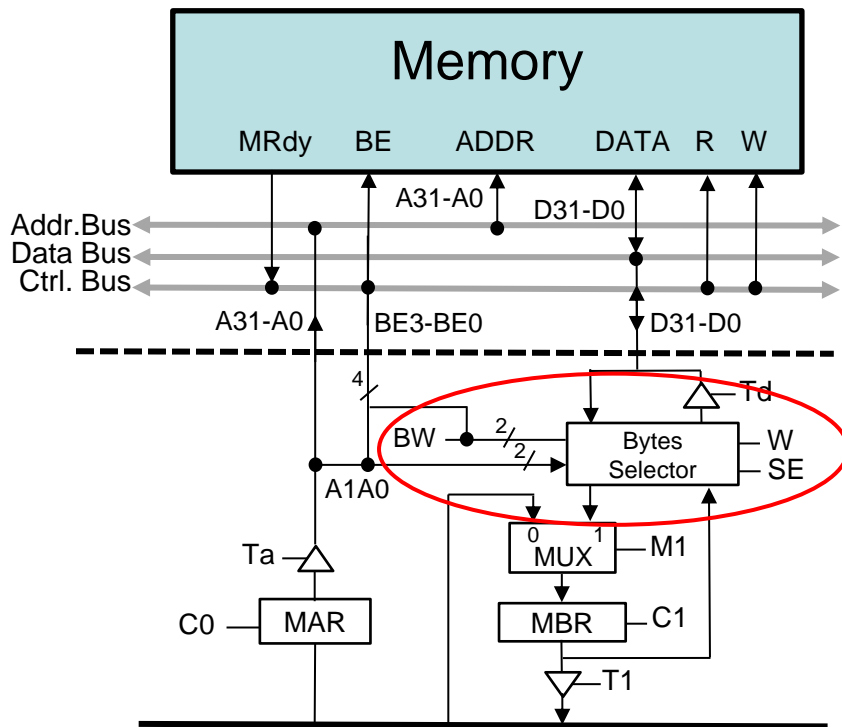
► Para lectura:

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

► 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



Nomenclatura:

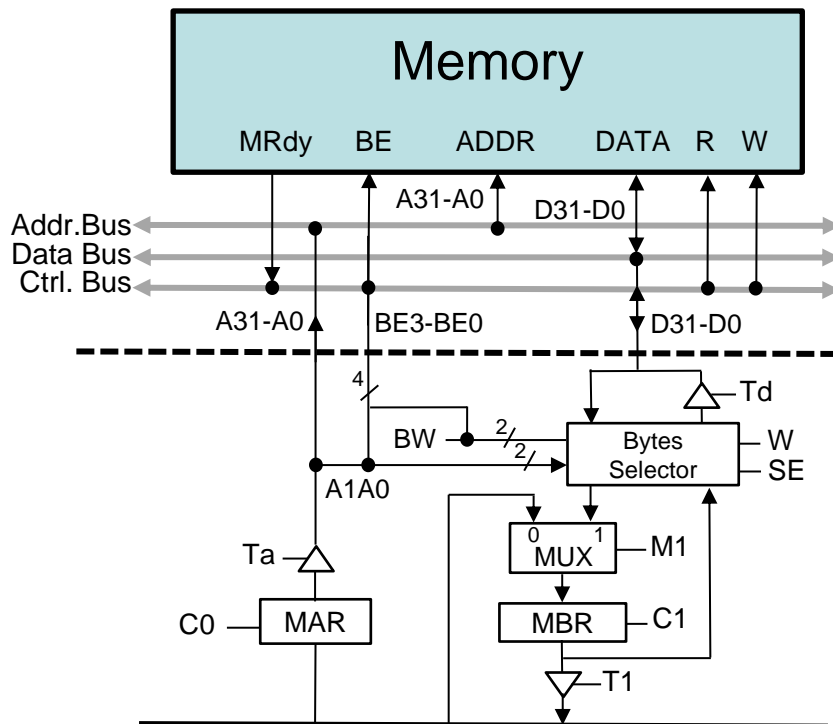
- MAR -> registro de direcciones
- MBR -> registro de datos

- ▶ Bytes Selector: selecciona qué bytes se almacenan en MBR en lectura y se vuelcan al bus en escritura
- ▶ BW: bytes a transferir
 - ▶ **BW=00**: Acceso a bytes
 - ▶ **BW=01**: Acceso a media palabra
 - ▶ **BW=11**: Acceso a palabra
- ▶ SE: extensión de signo
 - ▶ **SE=0**: no extiende el signo en accesos más pequeños de una palabra
 - ▶ **SE=1**: extiende el signo en accesos más pequeños de una palabra

Ejemplo

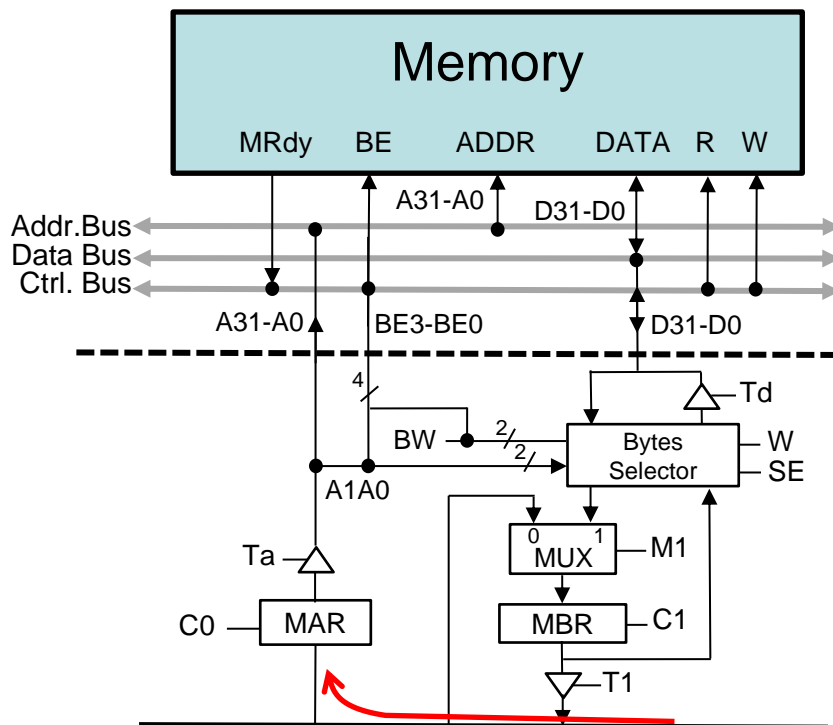
operaciones elementales para usar la memoria

► Lectura



Acceso a memoria síncrona de 1 ciclo

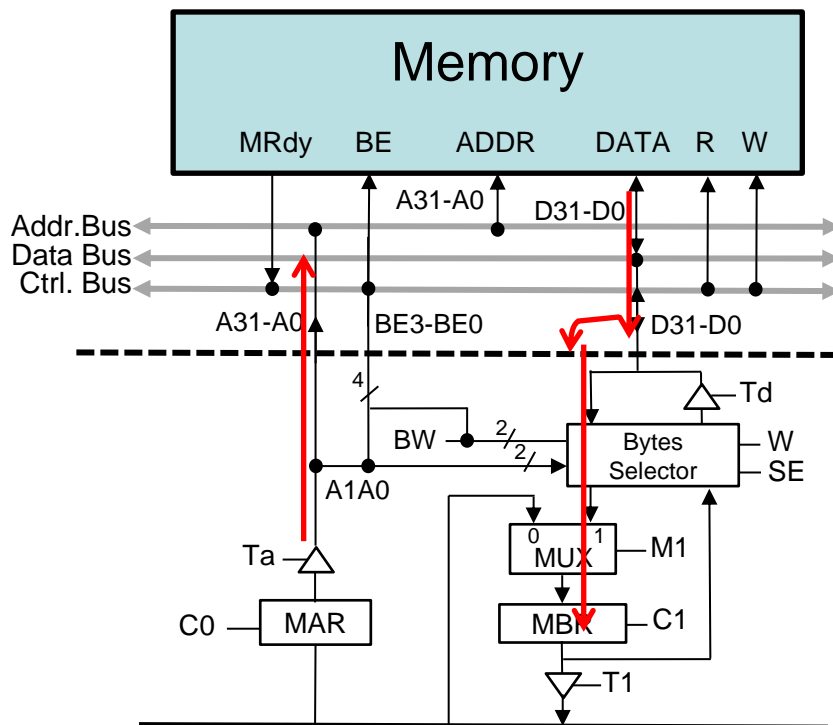
► Lectura



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

Acceso a memoria síncrona de 1 ciclo

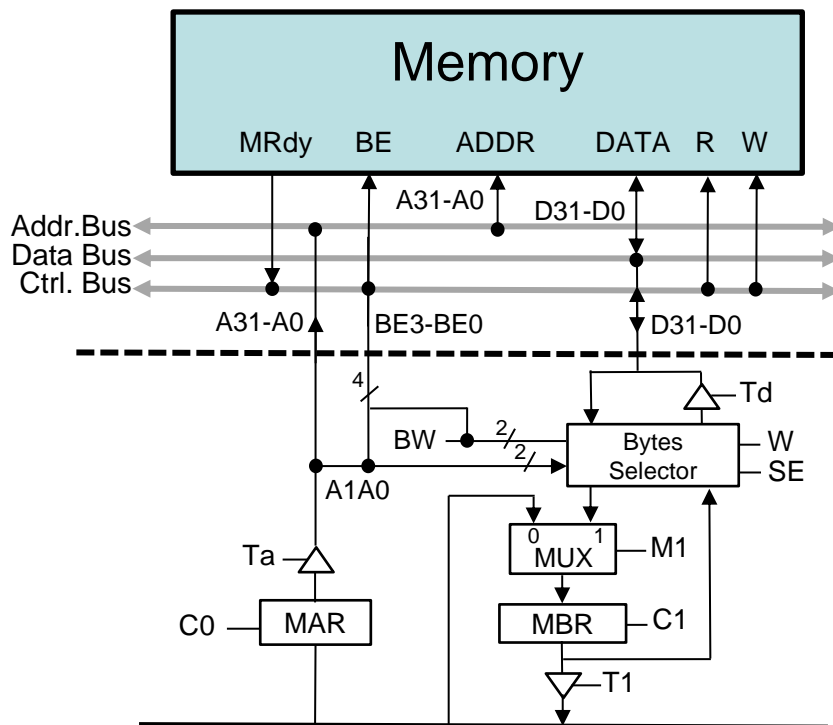
► Lectura de una palabra



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



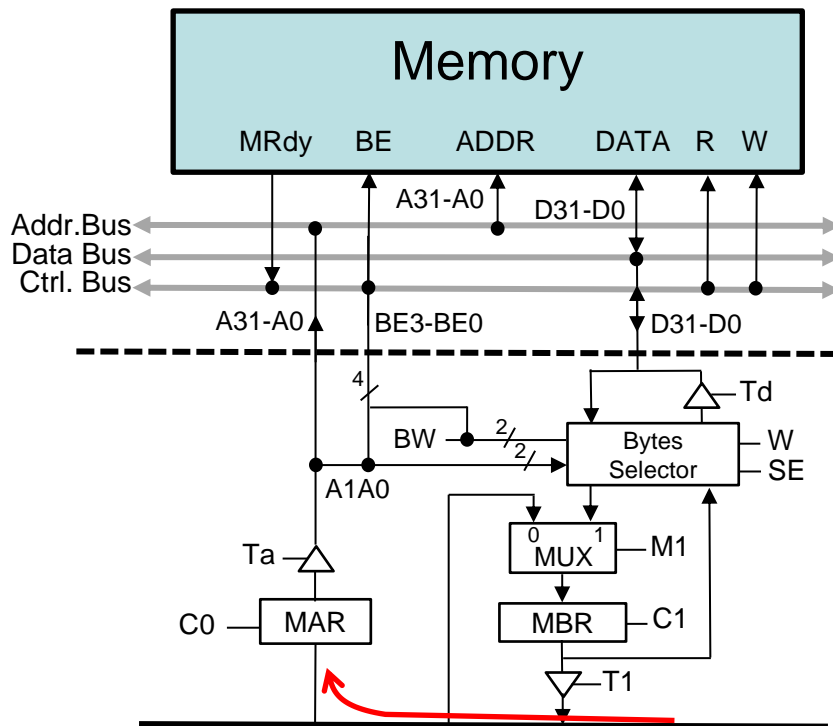
► Lectura de una palabra

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

► Escritura

Ejemplo

Acceso a memoria síncrona de 1 ciclo



► Lectura de una palabra

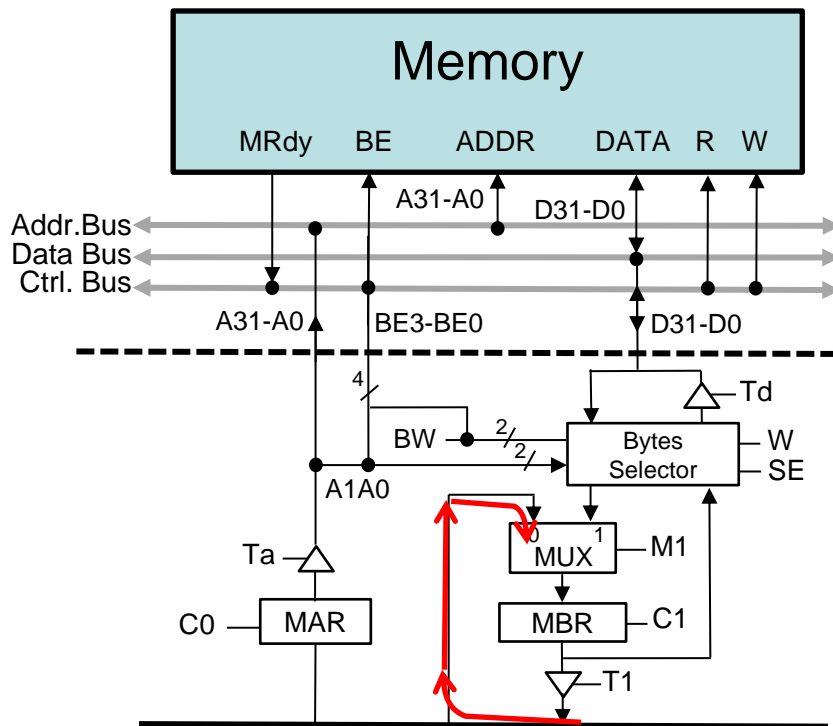
O. Elemental	Señales
$MAR \leftarrow \langle \text{dirección} \rangle$..., C0
$MBR \leftarrow MP[MAR]$	Ta, R, M1, C1, BW=11

► Escritura

O. Elemental	Señales
$MAR \leftarrow \langle \text{dirección} \rangle$..., C0

Ejemplo

Acceso a memoria síncrona de 1 ciclo



► Lectura de una palabra

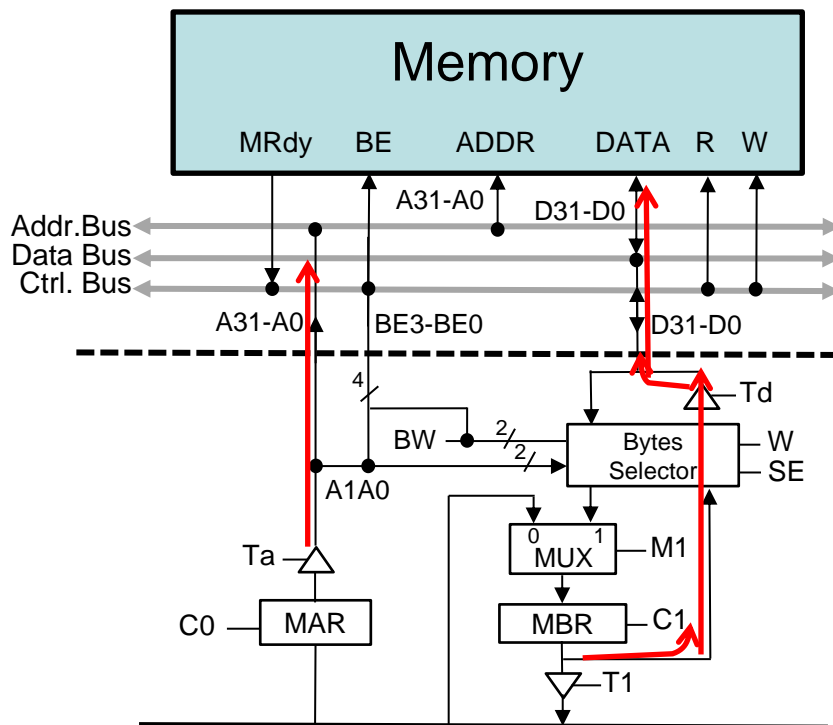
O. Elemental	Señales
MAR \leftarrow <dirección>	..., C0
MBR \leftarrow MP[MAR]	Ta, R, M1, C1, BW=11

► Escritura

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

Ejemplo

Acceso a memoria síncrona de 1 ciclo



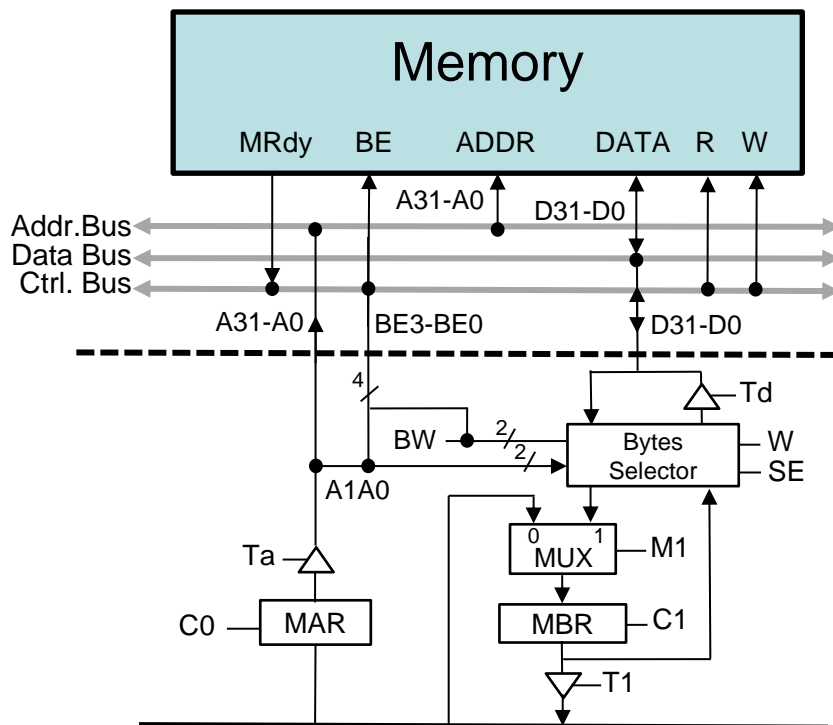
► Lectura de una palabra

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

► Escritura de una palabra

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



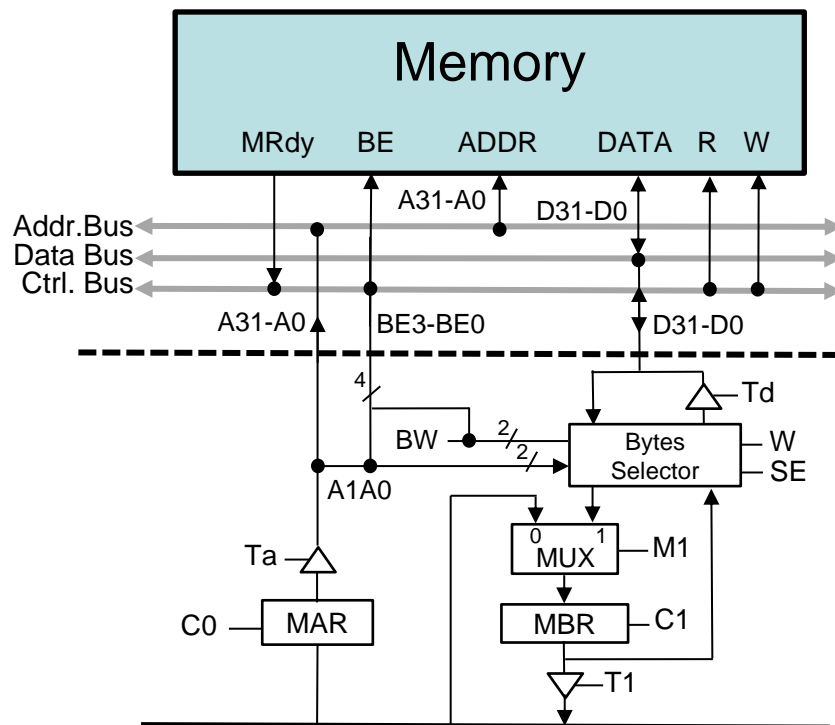
► Lectura de una palabra

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

► Escritura de una palabra

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

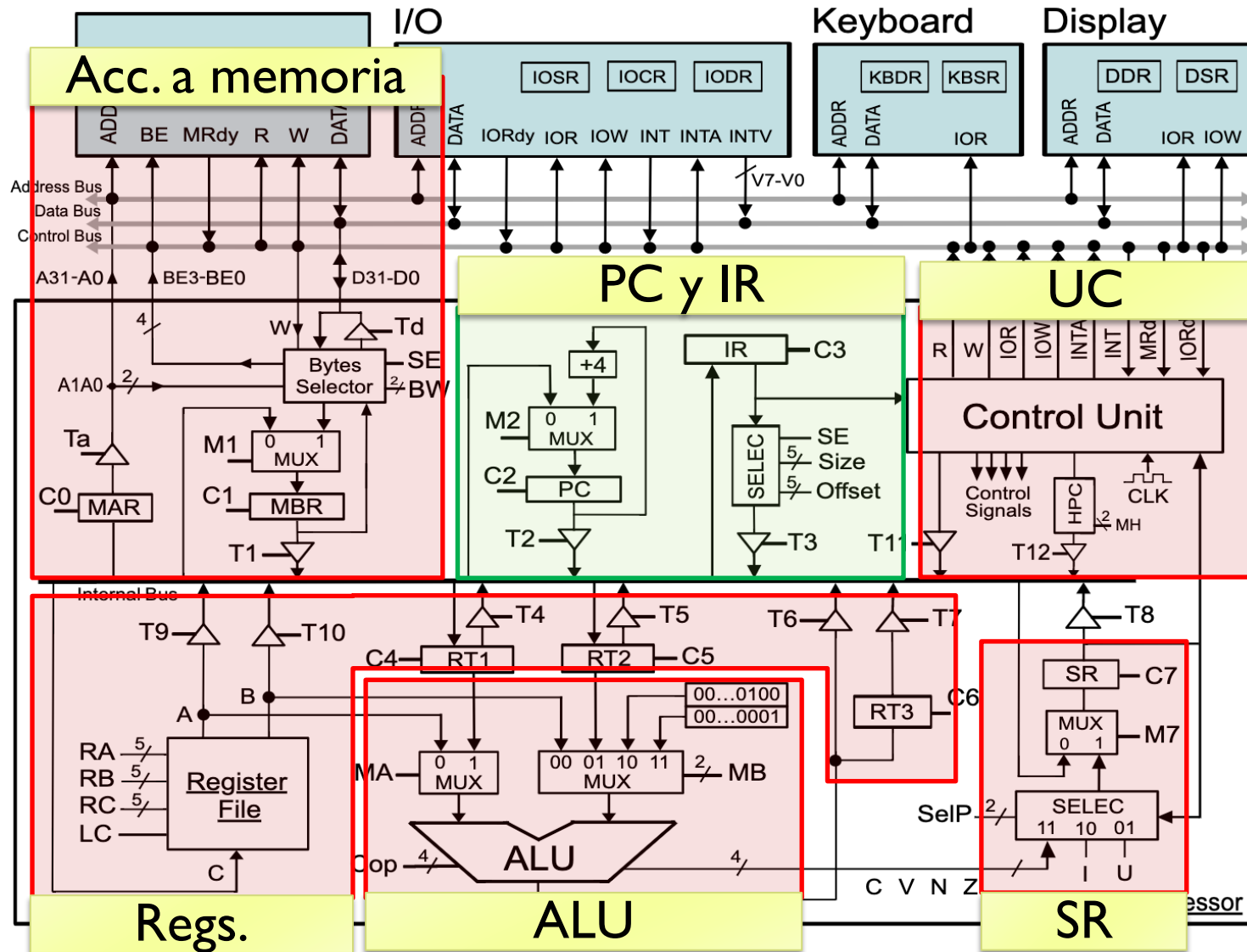
Acceso a memoria síncrona de **2** ciclo



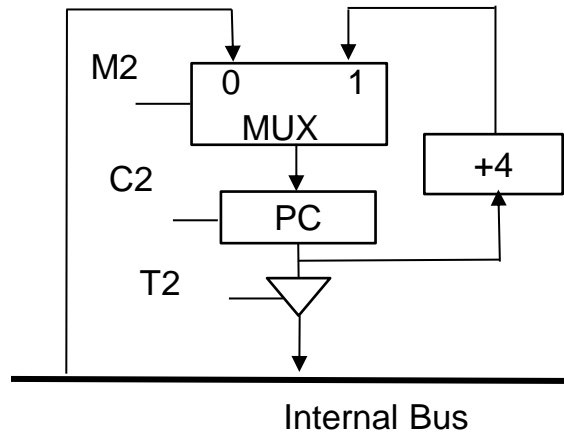
► Lectura de una palabra

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

Procesador elemental: señales de control



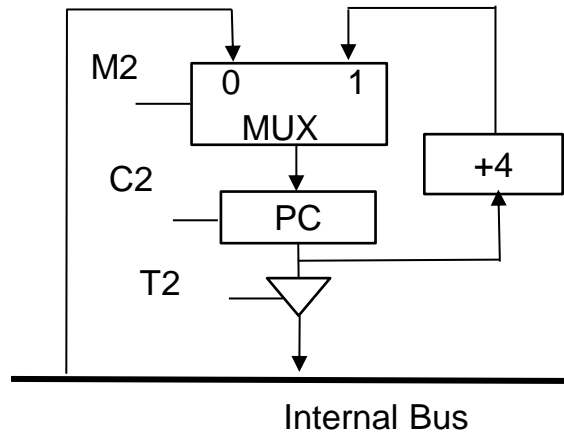
PC: Contador de programa



► PC

- C2 – elegido por M2 al PC
- T2 – de PC a bus interno
- M2 – bus interno ó PC+4

PC Mux: IB/PC+4

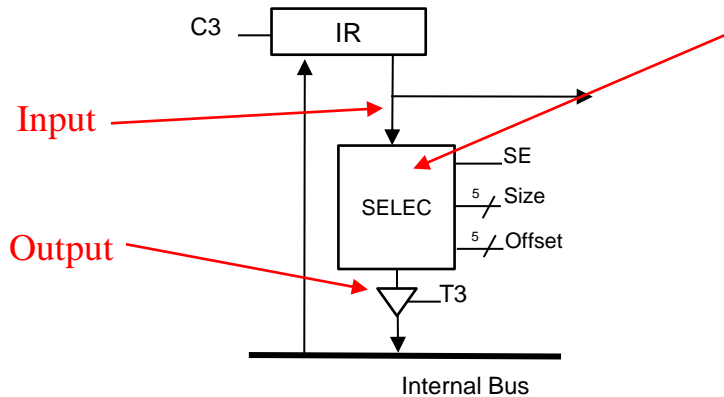


► PC

- C2 – elegido por M2 al PC
- T2 – de PC a bus interno
- M2 – bus interno ó PC+4

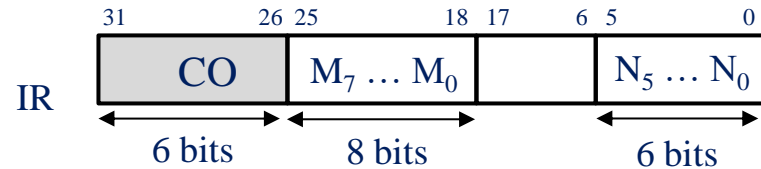
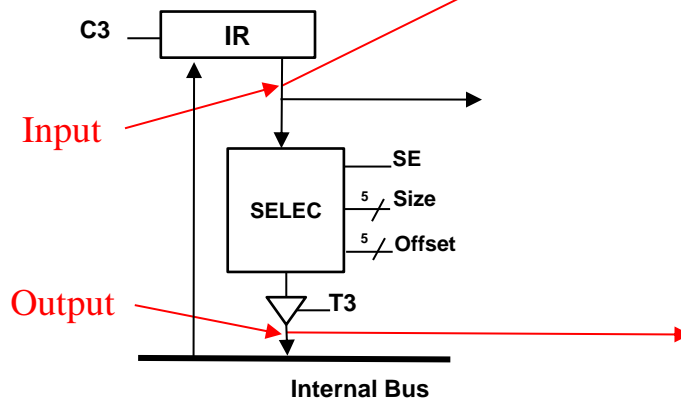
- C2, M2
 - $PC \leftarrow PC + 4$
- C2, M2=0
 - $PC \leftarrow \text{<internal bus>}$

IR: Registro de instrucción



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

SELEC: circuito selector



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

Size	Offset	Output
01000	10010	
00110	00000	

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

Size	Offset	Output
01000	10010	
00110	00000	

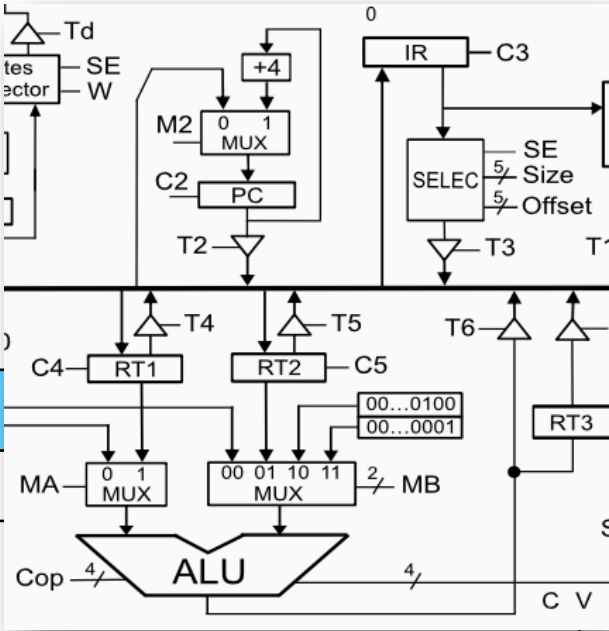
Tamaño en bits

Bit de inicio (menos significativo)

Ejecución de j addr

op .	address
6 bits	26 bits

Ciclo	Op. elem.	Señales de control



Ejecución de j addr

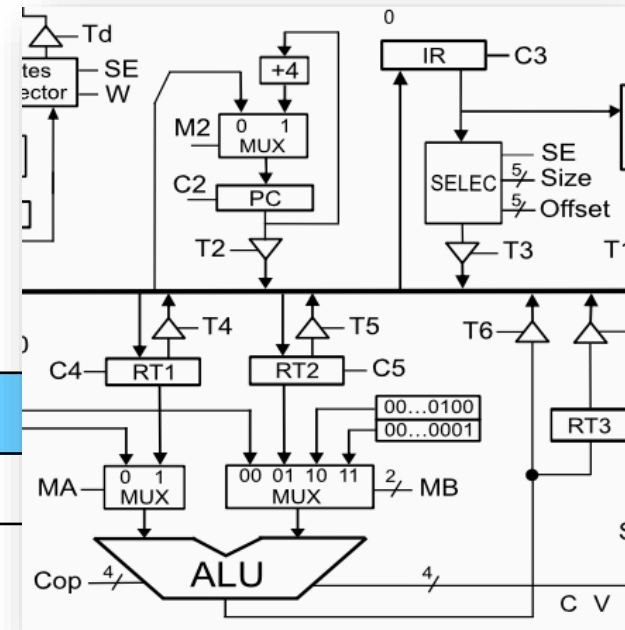
TIP

Fases generales:

- Fetch + Decodificar
- Traer operandos
- Ejecución
- Almacenar resultados

Ciclo	Op. elem.	Señales de control

op .	address
6 bits	26 bits



Ejecución de j addr

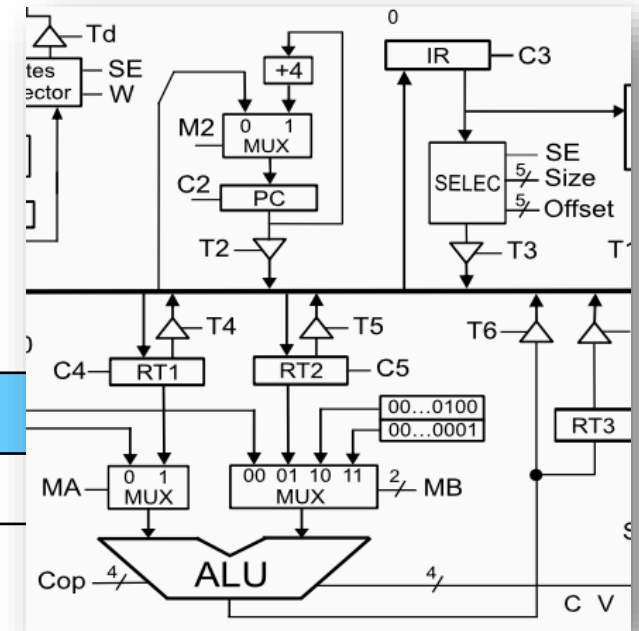
TIP

No es posible en el mismo ciclo de reloj:

1. Atravesar un registro (~~C4, MA=1~~)
2. Enviar varios valores al bus (~~T4, T5~~)
3. Establecer un camino de datos que la circuitería no permite (~~IDB → RT3~~)

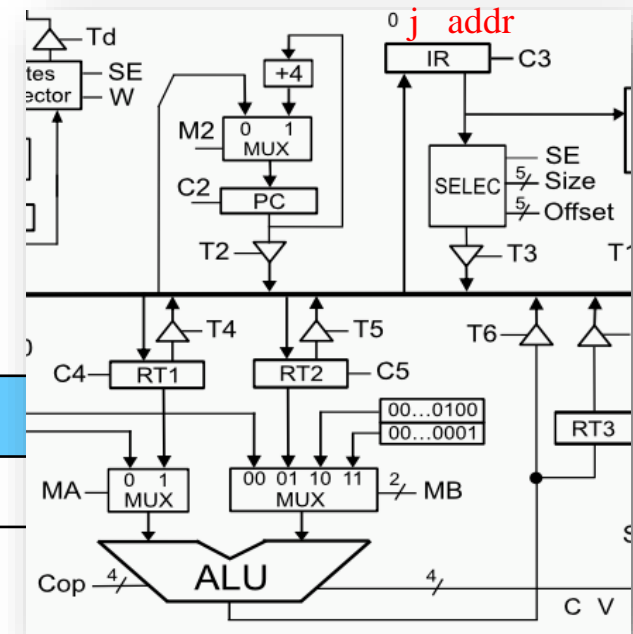
Ciclo	Op. elem.	Señales de control

op .	address
6 bits	26 bits



Ejecución de j addr

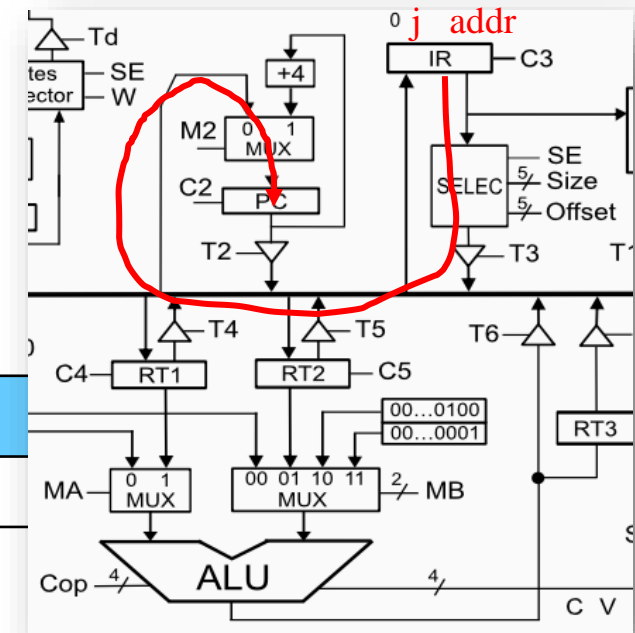
op.	address
6 bits	26 bits



Ciclo	Op. elem.	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	Decoding	A0, B=0, C=0

Ejecución de j addr

op.	address
6 bits	26 bits

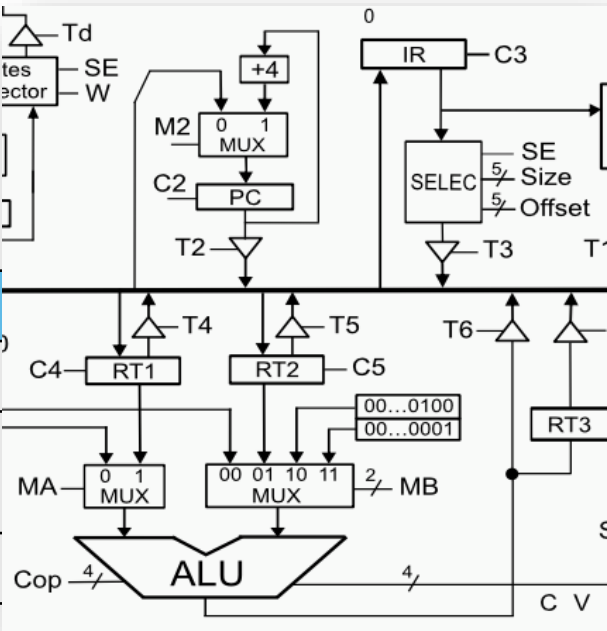


Ciclo	Op. elem.	Señales de control
C1	$MAR \leftarrow PC$	T2, C0
C2	$PC \leftarrow PC + 4$, $MBR \leftarrow MP$	C2, M1 Ta, R, C1, M1, BW=11
C3	$IR \leftarrow MBR$	T1, C3
C4	Decoding	A0, B=0, C=0
C5	$PC \leftarrow RI(dir)$	Size = 11010 (26), Offset = 00000, SE=0, C2, T3
C6	Salto a fetch	A0, B=1, C=0

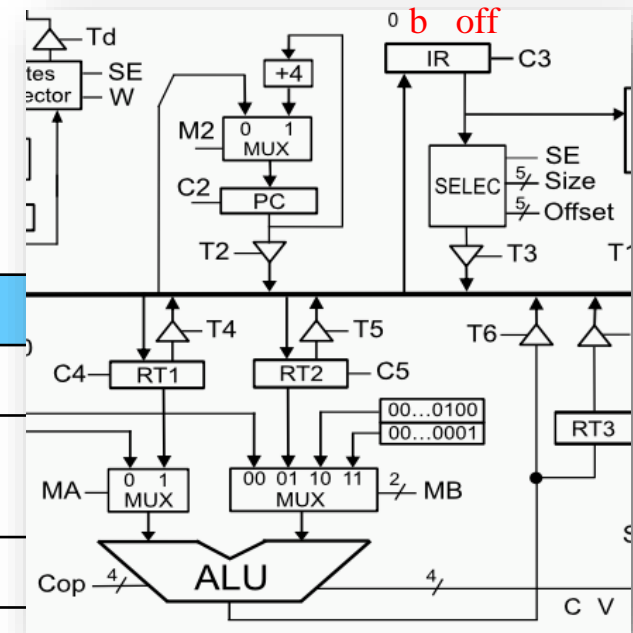
Ejecución de b offset



Ciclo	Op. elem.	Señales de control

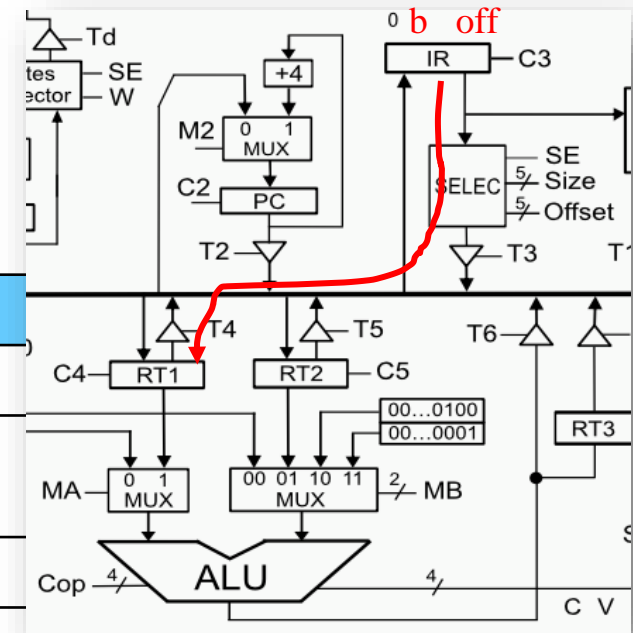


Ejecución de b offset



Ciclo	Op. elem.	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	Decoding	A0, B=0, C=0

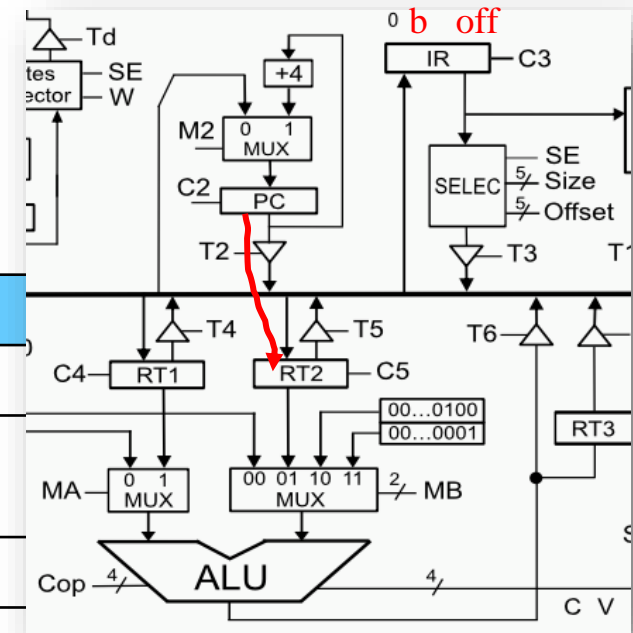
Ejecución de b offset



Ciclo	Op. elem.	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	Decoding	A0, B=0, C=0
C5	$RT1 \leftarrow RI(off)$	Size=10000 (16), Offset=0, SE=1, T3, C4

Ejecución de b offset

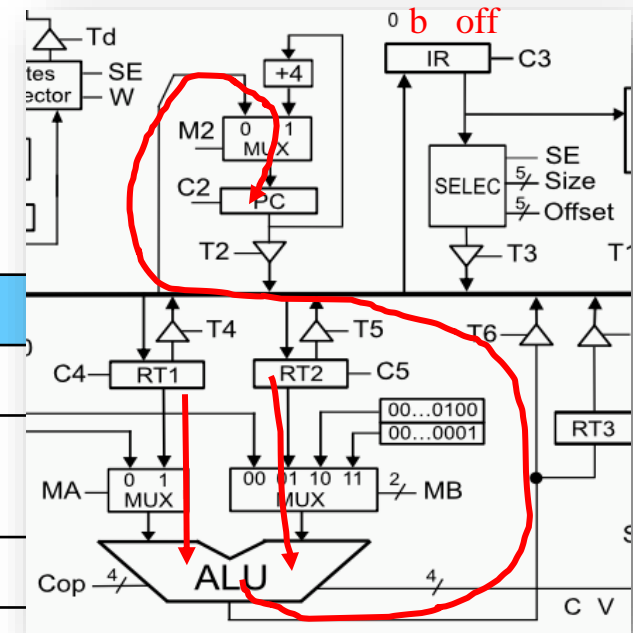
op.	...	offset
6 bits		16 bits



Ciclo	Op. elem.	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	Decoding	A0, B=0, C=0
C5	$RT1 \leftarrow RI(off)$	Size=10000 (16), Offset=0, SE=1, T3, C4
C6	$RT2 \leftarrow PC$	T2, C5

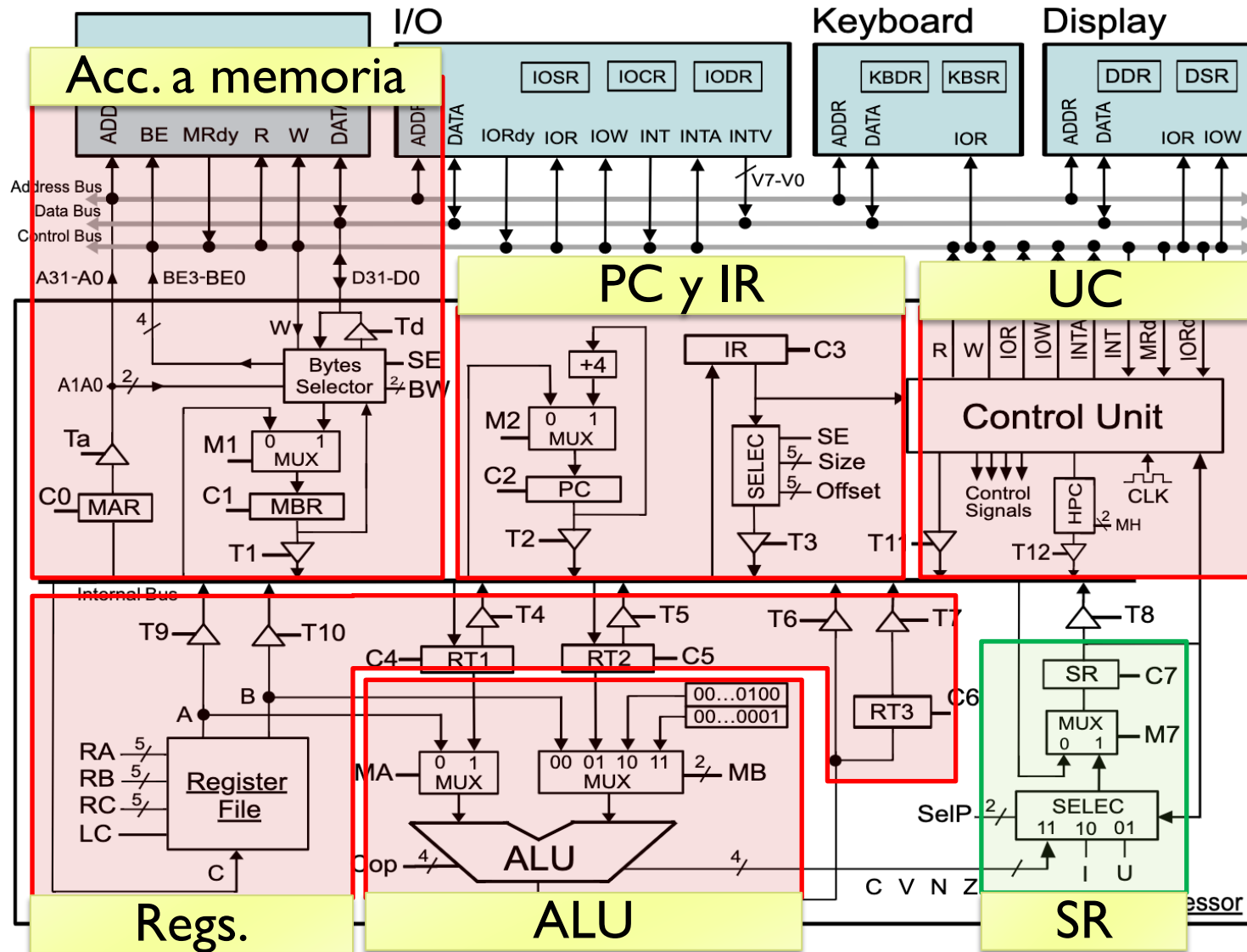
Ejecución de b offset

op.	...	offset
6 bits		16 bits

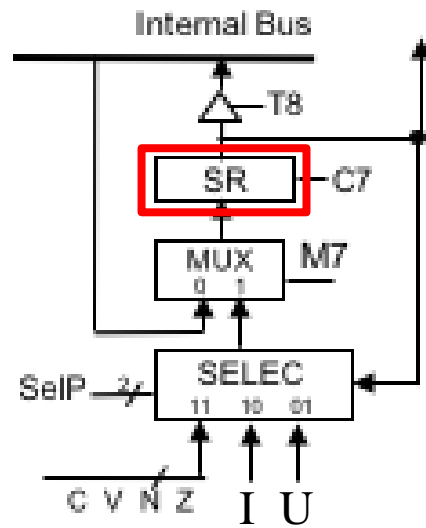


Ciclo	Op. elem.	Señales de control
C1	$MAR \leftarrow PC$	T2, C0
C2	$PC \leftarrow PC + 4,$ $MBR \leftarrow MP$	C2, M1 Ta, R, C1, M1, BW=11
C3	$IR \leftarrow MBR$	T1, C3
C4	Decoding	A0, B=0, C=0
C5	$RT1 \leftarrow RI(off)$	Size=10000 (16), Offset=0, SE=1, T3, C4
C6	$RT2 \leftarrow PC$	T2, C5
C7	$PC \leftarrow RT1 + RT2$	MA=1, MB=1, MC=1, SELCOP=+, T6, M2=0, C2
C6	Salto a fetch	A0, B=1, C=0

Procesador elemental: señales de control

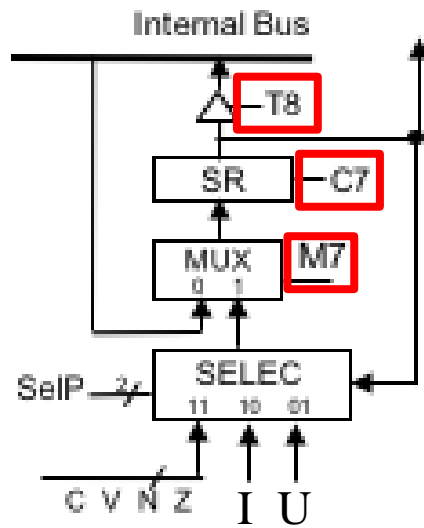


SR: Registro de estado



- ▶ Almacena información (bits de estado) sobre **el estado del programa que se está ejecutando en el procesador.**
- ▶ Bit de estado típicos:
 - ▶ C, V, N, Z:
Resultado de la **última operación en la ALU**
 - ▶ U:
CPU ejecuta en **modo núcleo o modo usuario**
 - ▶ I:
interrupciones están habilitadas o no

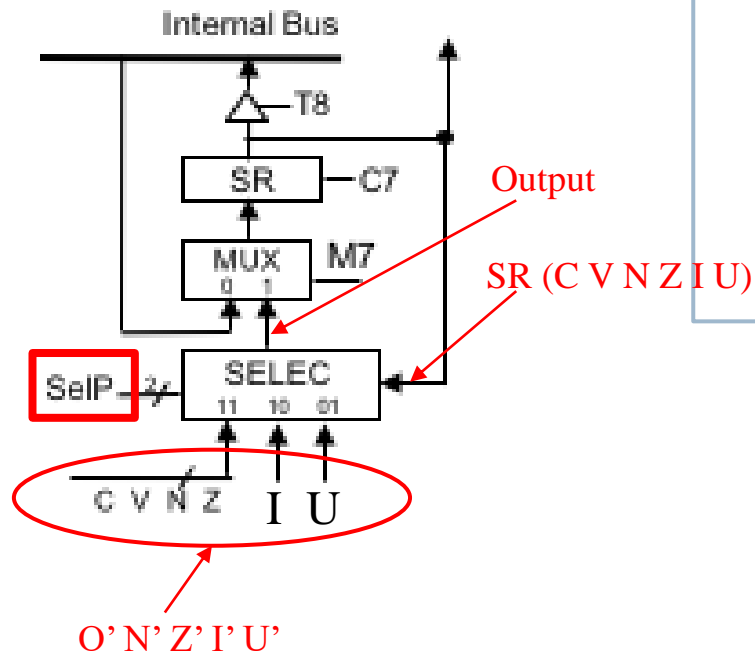
SR: Registro de estado



► SR

- T8 – del SR al bus interno
- C7 – de bus interno al SR
- M7 – flags de IDB o SELEC

SR: Registro de estado



► SR

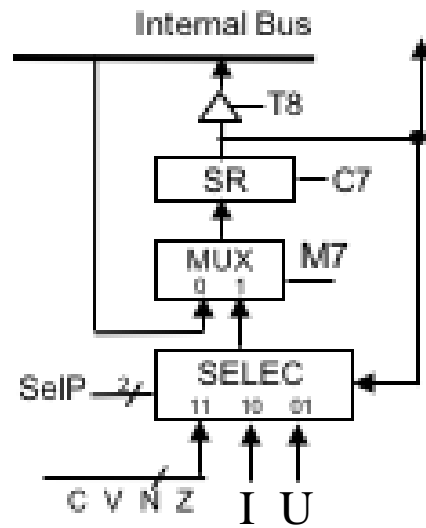
- T8 – del SR al bus interno
- C7 – de bus interno al SR
- M7 – flags de IDB o SELEC
- SelP – actualiza flags: ALU / I / U

if (SelP == 11)
 Output = **C' V' N' Z' I U**

if (SelP == 10)
 Output = **C V N Z I' U**

if (SelP == 01)
 Output = **C V N Z I U'**

SR: Salva/Restaura + Actualizar desde ALU

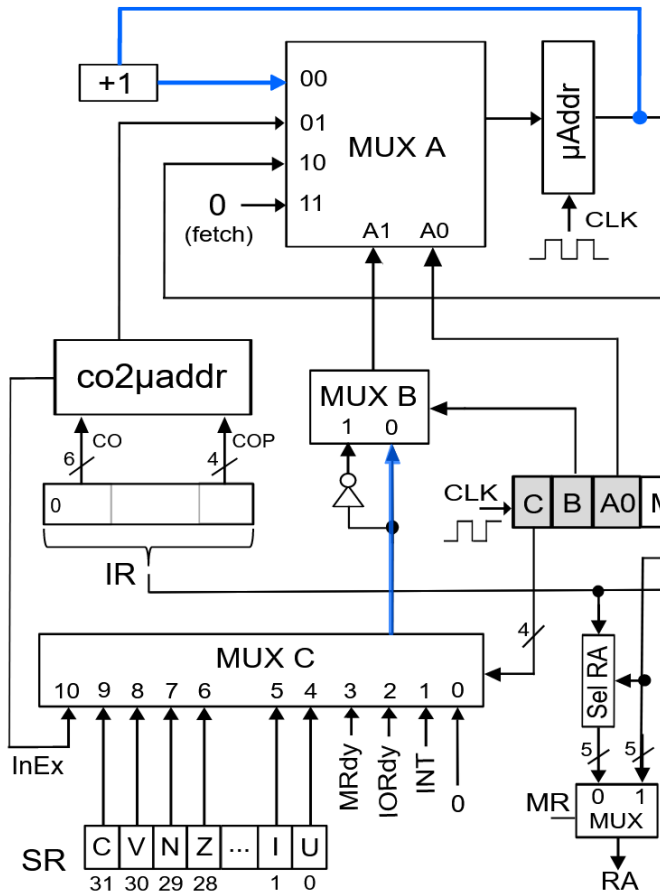


► SR

- T8 – del SR al bus interno
- C7 – de bus interno al SR
- M7 – flags de IDB o SELEC
- SelP – actualiza flags: ALU / I / U

- T8, C4
 - $RTI \leftarrow SR$
- T4, M7=0, C7
 - $SR \leftarrow RTI$
- SelP=11, M7, C7
 - $SR \leftarrow \langle \text{flags de ALU} \rangle$

SR: comprobar flag...



► **SR**

- ▶ C – condición a seleccionar
- ▶ B – condición o no condición
- ▶ A0 – I: fetch/co2maddr
0: desde maddr/maddr+I

- ▶ $C=[4\dots 9], B=0, A_0=0$
 - ▶ If $C \text{ maddr} \leftarrow \text{MADDR}$
else $\text{maddr} \leftarrow \text{maddr}+1$
- ▶ $C=0, B=1, A_0=0$
 - ▶ $\text{maddr} \leftarrow \text{MADDR}$

beq r1, r2, offset

Ciclo	Op. elem.
C1	$MAR \leftarrow PC$
C2	$PC \leftarrow PC + 4,$ $MBR \leftarrow MP$
C3	$IR \leftarrow MBR$
C4	Decodificar
C12	Jump to fetch

Si $r1 == r2$
 $PC \leftarrow PC + \text{offset}$

beq r1, r2, offset

Ciclo	Op. elem.
C1	$MAR \leftarrow PC$
C2	$PC \leftarrow PC + 4,$ $MBR \leftarrow MP$
C3	$IR \leftarrow MBR$
C4	Decodificar
C5	$MBR \leftarrow SR$
C6	$r1 - r2$
C7	Si $SR.Z \neq 0$ salta a C11
C11	$SR \leftarrow MBR$
C12	Jump to fetch

Si $r1 == r2$
 $PC \leftarrow PC + \text{offset}$

beq r1, r2, offset

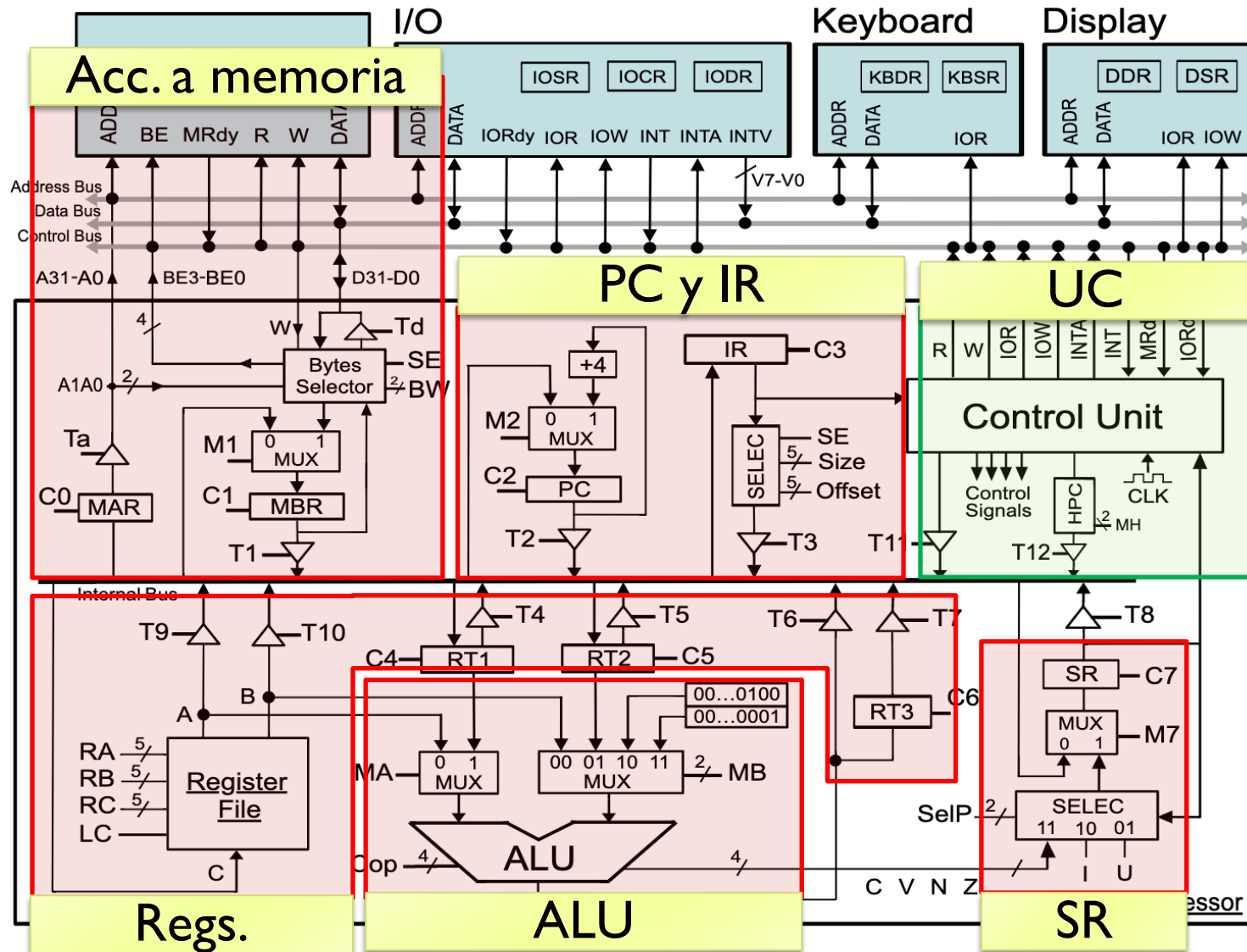
Ciclo	Op. elem.
C1	$MAR \leftarrow PC$
C2	$PC \leftarrow PC + 4,$ $MBR \leftarrow MP$
C3	$IR \leftarrow MBR$
C4	Decodificar
C5	$MBR \leftarrow SR$
C6	$r1 - r2$
C7	Si $SR.Z \neq 0$ salta a C11
C8	$RT1 \leftarrow PC$
C9	$RT2 \leftarrow IR(\text{offset})$
C10	$PC \leftarrow RT1 + RT2$
C11	$SR \leftarrow MBR$
C12	Jump to fetch

Si $r1 == r2$
 $PC \leftarrow PC + \text{offset}$

beq r1, r2, offset

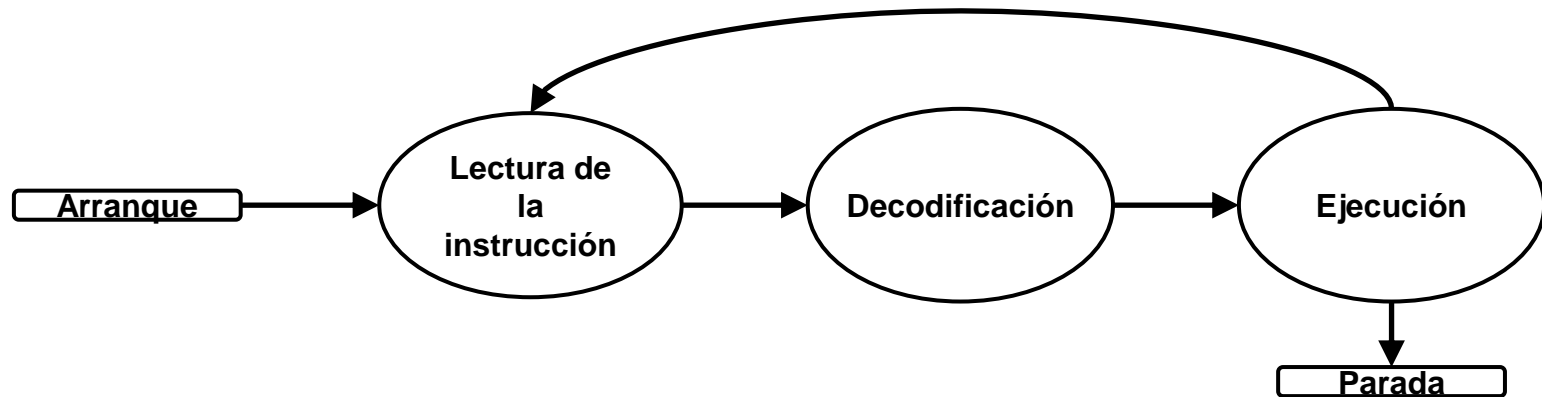
Ciclo	Op. elem.	Señales de control
C1	$MAR \leftarrow PC$	T2, C0
C2	$PC \leftarrow PC + 4,$ $MBR \leftarrow MP$	C2, M1, Ta, R, C1, M1, BW=11
C3	$IR \leftarrow MBR$	T1, C3
C4	Decodificar	A0, B=0, C=0
C5	$MBR \leftarrow SR$	T8, C1
C6	r1 - r2	SELA=10101, SELB=10000, MC=1, SELCOP=1011, SELP=11, M7, C7
C7	Si SR.Z != 0 salta a C11	A0=0, B=1, C=110, MADDR=beq11
C8	$RT1 \leftarrow PC$	T2, C4
C9	$RT2 \leftarrow IR(\text{offset})$	Size=10000, Offset=0, SE=1, T3, C5
C10	$PC \leftarrow RT1 + RT2$	MA=1, MB=1, MC=1, SELCOP=+, T6, C2
C11	$SR \leftarrow MBR$	T1, M7=0, C7
C12	Jump to fetch	A0, B=1, C=0

Procesador elemental: señales de control



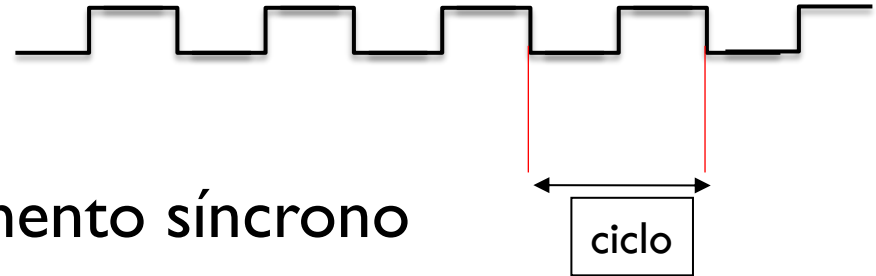
Unidad de control

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.
 - ▶ Señales de control a activar
- ▶ **Ejecución**
 - ▶ Generación de las señales de control en cada ciclo de reloj.

Reloj

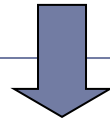


- ▶ Un computador es un elemento síncrono
 - ▶ Controla el funcionamiento
- ▶ El reloj temporiza las operaciones
 - ▶ En un ciclo de reloj se ejecutan una o más operaciones elementales siempre que no haya conflicto
 - ▶ En un mismo ciclo se puede realizar
 - $MAR \leftarrow PC$ y $RT3 \leftarrow RT2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

Descripción de la actividad de la U.C.

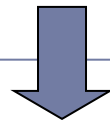
Instrucción

mv R0 R1

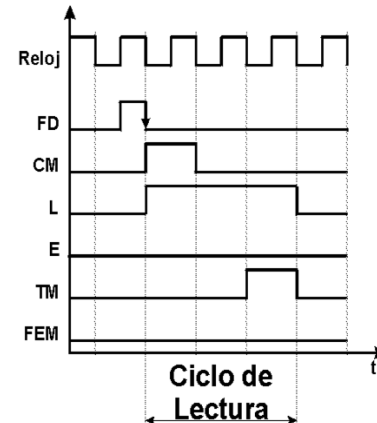


Secuencia de **operaciones elementales**

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

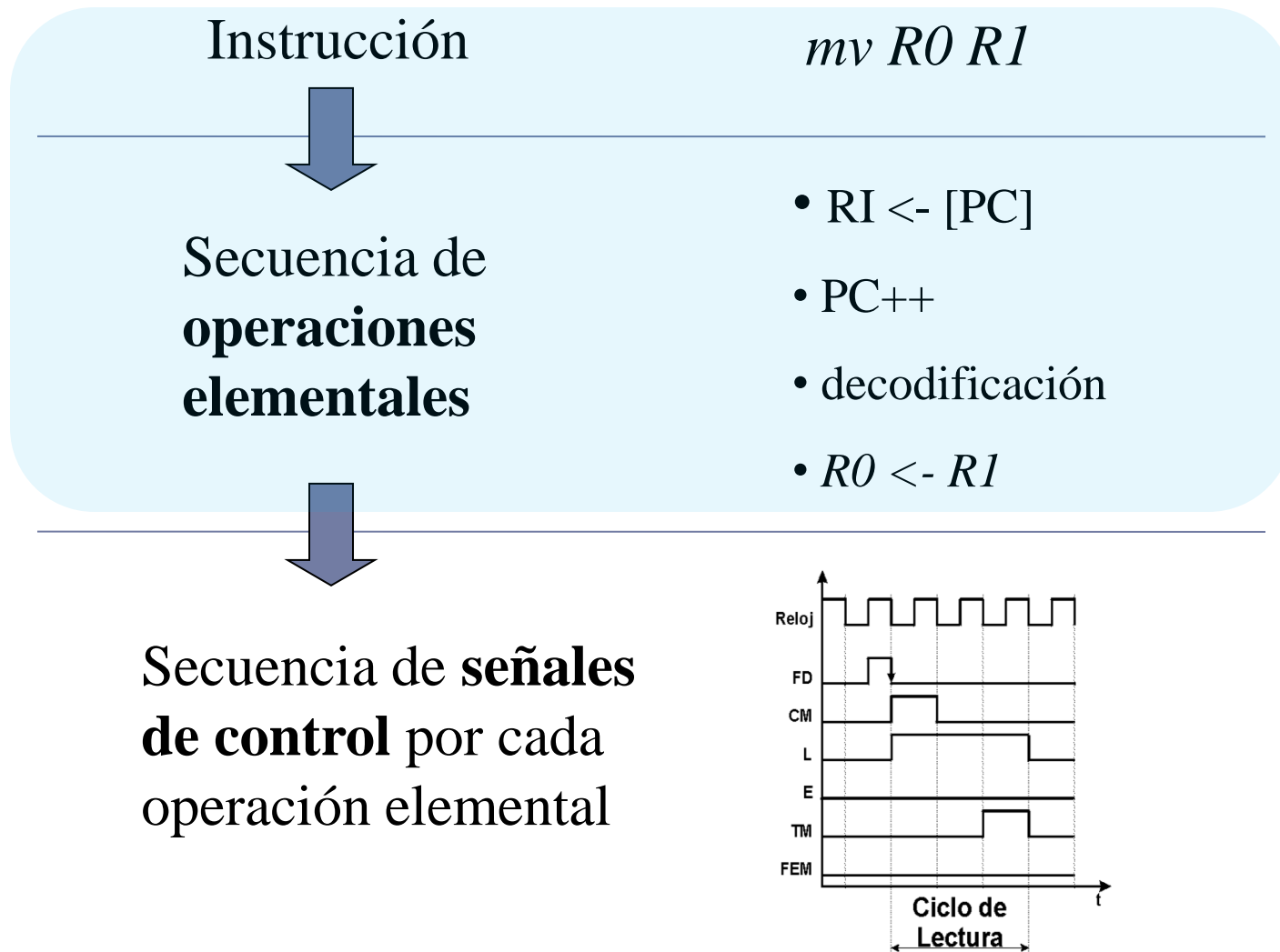


Secuencia de **señales de control** por cada operación elemental



+ 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

Descripción de la actividad de la U.C.

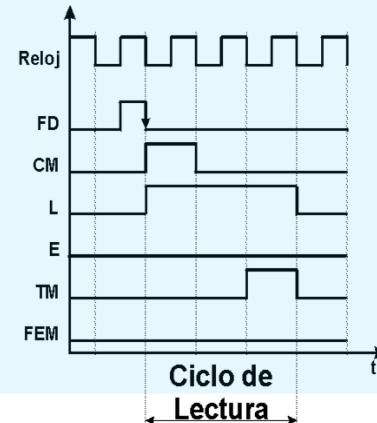
Instrucción

mv R0 R1

Secuencia de
**operaciones
elementales**

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

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



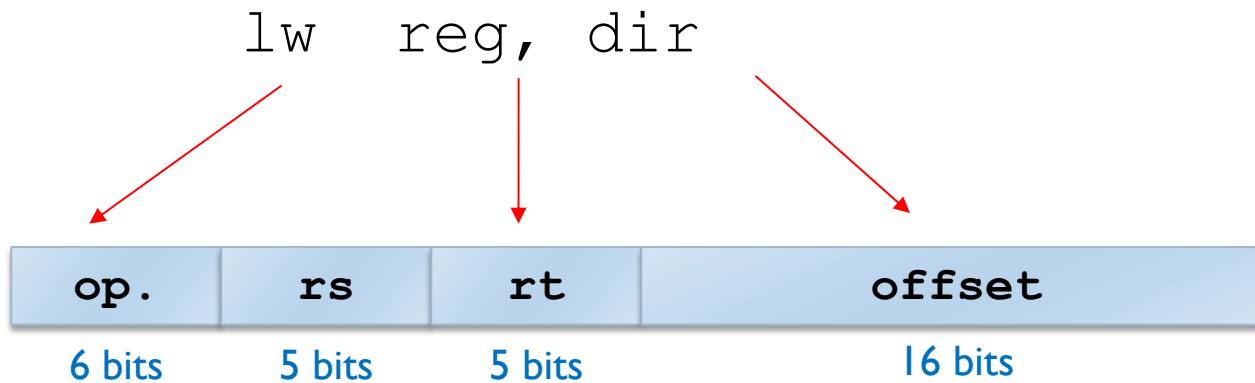
Señales de control del ciclo de fetch

- 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

Ejemplo

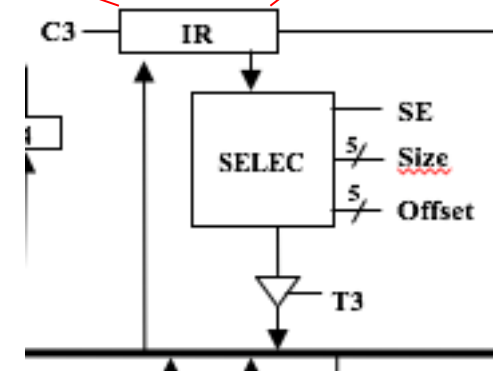
► `lw reg, dir`



Ejecución de lw reg, dir



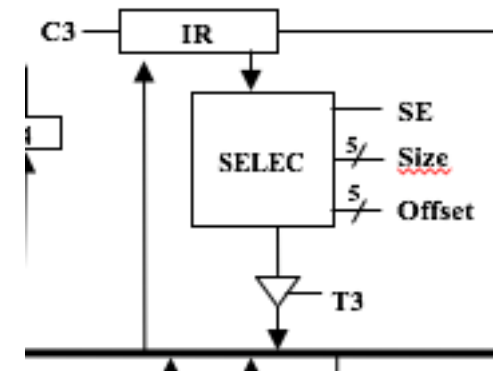
Ciclo	Op. Elemental	Señales de control
C1	$MAR \leftarrow PC$	T2, C0
C2	$PC \leftarrow PC + 4,$ $MBR \leftarrow MP$	C2, M2 Ta, R, CI, MI, BW=II
C3	$IR \leftarrow MBR$	T1, C3
C4		
C5		
C6		
C7		



Ejecución de lw reg, dir



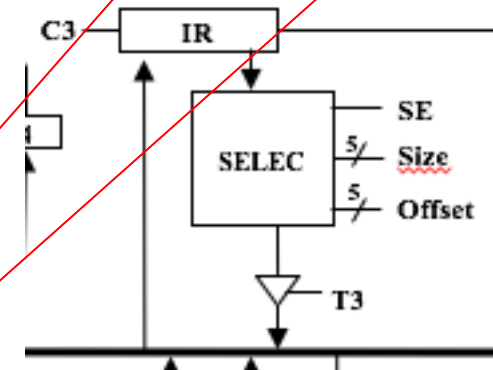
Ciclo	Op. Elemental	Señales de control
C1	$MAR \leftarrow PC$	T2, C0
C2	$PC \leftarrow PC + 4,$ $MBR \leftarrow MP$	C2, M2 Ta, R, CI, MI, BW=II
C3	$IR \leftarrow MBR$	T1, C3
C4	Decodificación	
C5		
C6		
C7		



Ejecución de lw reg, dir



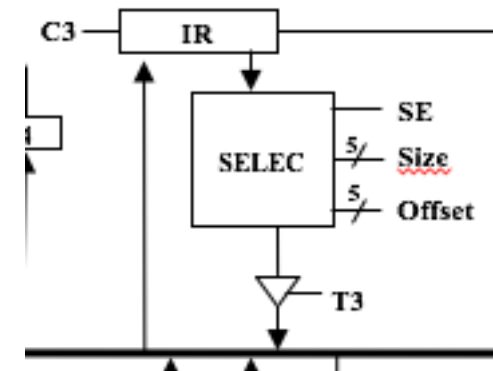
Ciclo	Op. Elemental	Señales de control
C1	$MAR \leftarrow PC$	T2, C0
C2	$PC \leftarrow PC + 4,$ $MBR \leftarrow MP$	C2, M2 Ta, R, CI, MI, BW=11
C3	$IR \leftarrow MBR$	T1, C3
C4	Decodificación	
C5	$MAR \leftarrow RI(dir)$	C0, T3, Size = 10000 Offset = 00000
C6		
C7		



Ejecución de lw reg, dir



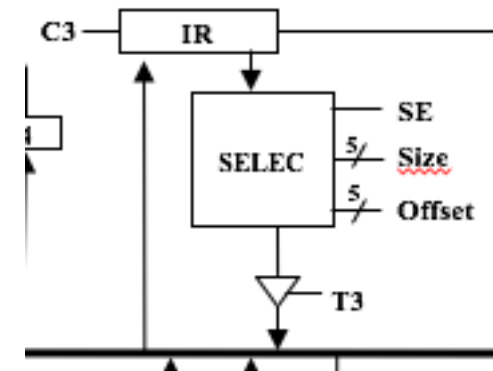
Ciclo	Op. Elemental	Señales de control
C1	$MAR \leftarrow PC$	T2, C0
C2	$PC \leftarrow PC + 4,$ $MBR \leftarrow MP$	C2, M2 Ta, R, CI, MI, BW=11
C3	$IR \leftarrow MBR$	T1, C3
C4	Decodificación	
C5	$MAR \leftarrow RI(dir)$	C0, T3, Size = 10000 Offset = 00000
C6	$MBR \leftarrow MP$	Ta, R, CI, MI, BW=11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



Ejecución de lw reg, dir



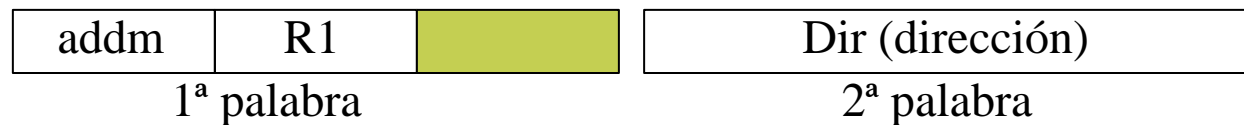
Ciclo	Op. Elemental	Señales de control
C1	$MAR \leftarrow PC$	T2, C0
C2	$PC \leftarrow PC + 4,$ $MBR \leftarrow MP$	C2, M2 Ta, R, CI, MI, BW=11
C3	$IR \leftarrow MBR$	T1, C3
C4	Decodificación	
C5	$MAR \leftarrow RI(dir)$	C0, T3, Size = 10000 Offset = 00000
C6	$MBR \leftarrow MP$	Ta, R, CI, MI, BW=11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



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	$RI \leftarrow RI + RTI$

Consejos simples (1/2)

fases generales de una instrucción

- A. Fetch + Decodif.
- B. Traer operandos
- C. Ejecutar
- D. Guardar resultados

Ejemplo

ADD (R_2) R_3 (R_4)

A. Fetch + Decodif.

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

B. Traer operandos

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

C. Ejecutar

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

D. Guardar resultados

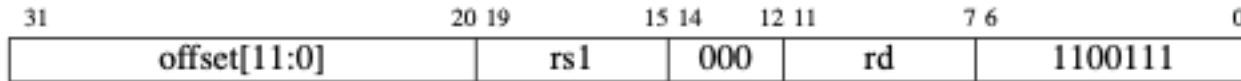
- 9.- $MAR \leftarrow R_2$
- 10.- $\text{Memoria}(MAR) \leftarrow MBR$

Consejos simples (2/2)

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.**

Ejecución de jr rsl



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=II
C3	$IR \leftarrow MBR$	T1, C3
C4	Decodificación	
C5	$PC \leftarrow rsl$	C2, RA=nI de RSI, T9, C2

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	$PC \leftarrow RT1 + RT2$

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

► Instrucciones que caben en una palabra:

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

Grupo ARCOS

uc3m | Universidad **Carlos III** de Madrid

Tema 4 (I)

El procesador

Estructura de Computadores
Grado en Ingeniería Informática