Juan Manuel Orduña Huertas

Fundamentos de computadores II

VNIVERSITAT DE VALÈNCIA

Escola Tècnica Superior d'Enginyeria ETSE-UV

VICERECTORAT DE PARTICIPACIÓ I PROJECCIÓ TERRITORIAL

Universitat i Societat

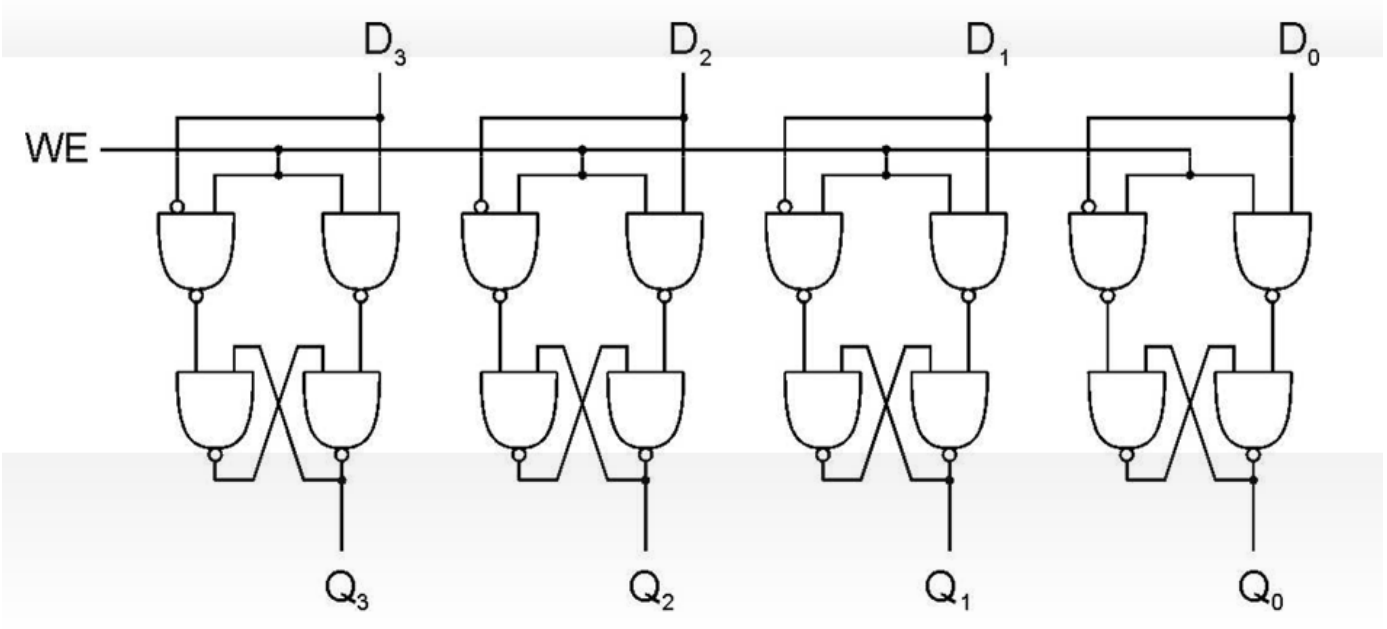
CULLERA

Fundamentos de computadores

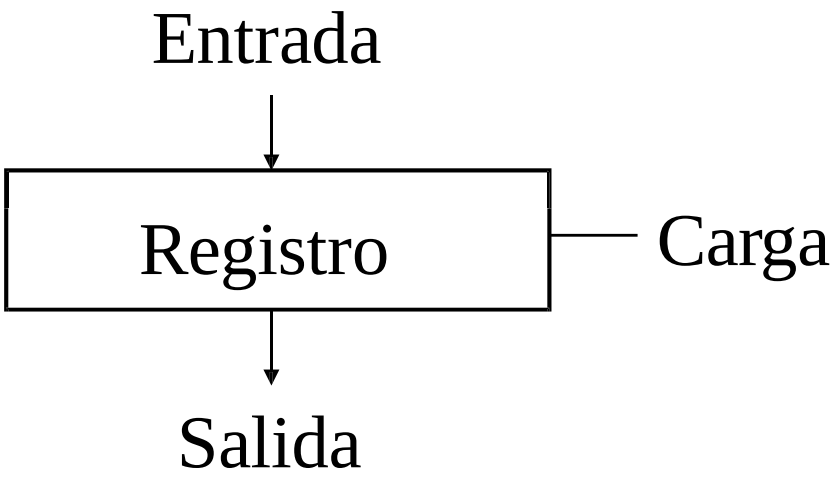
1. Sistemas de numeración
2. Circuitos combinacionales y secuenciales
3. **Estructura de computadores**

Elementos de memoria
Registro

- Elemento que almacena un conjunto de bits



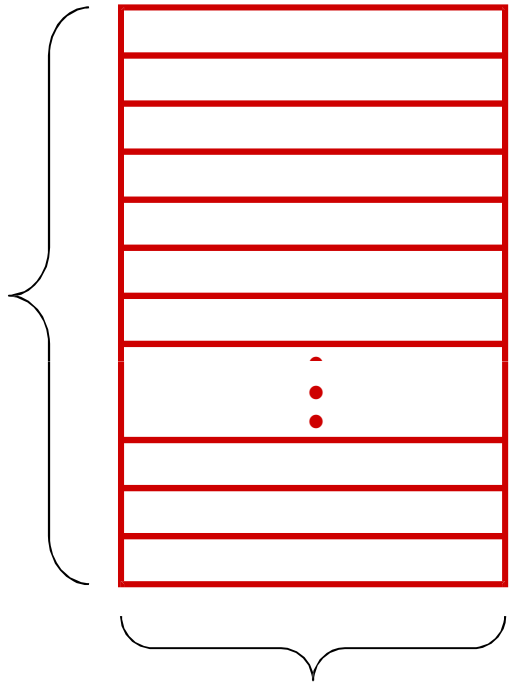
Esquema de un registro



Memoria

Espacio de direcciones:
Número de posiciones

$k = 2^n$ posiciones

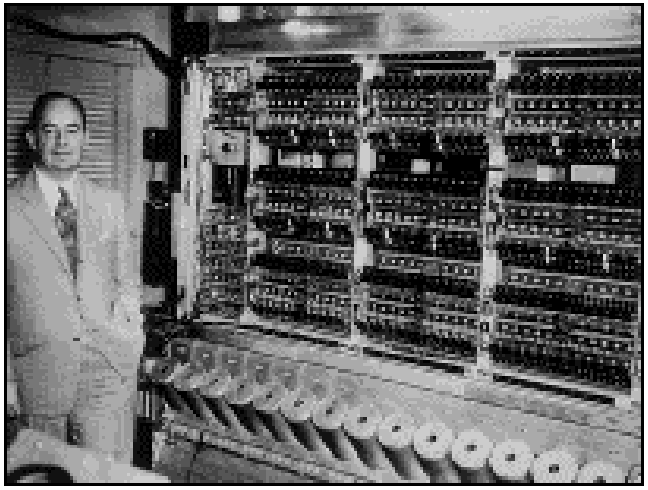


m bits

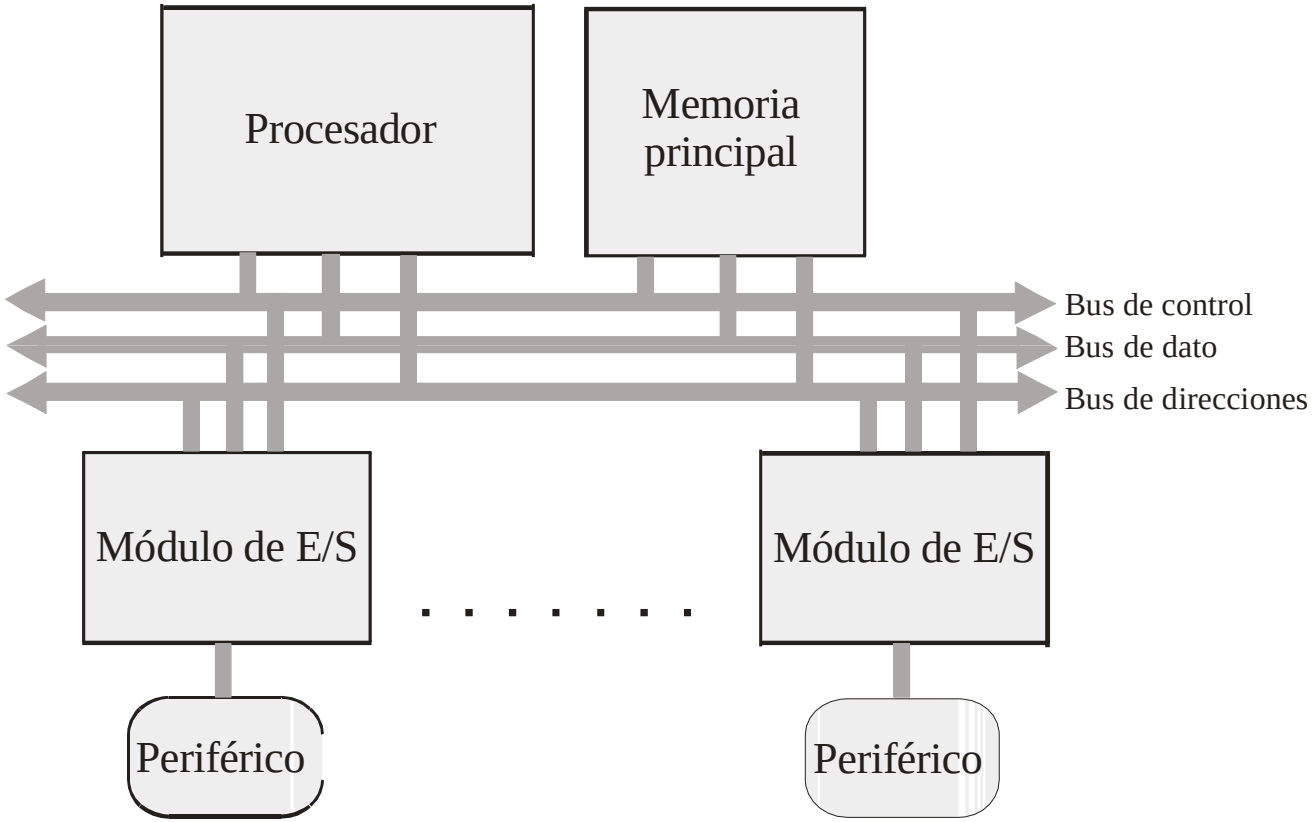
Tamaño de cada posición:
Número de bits por posición

Computador von Neumann

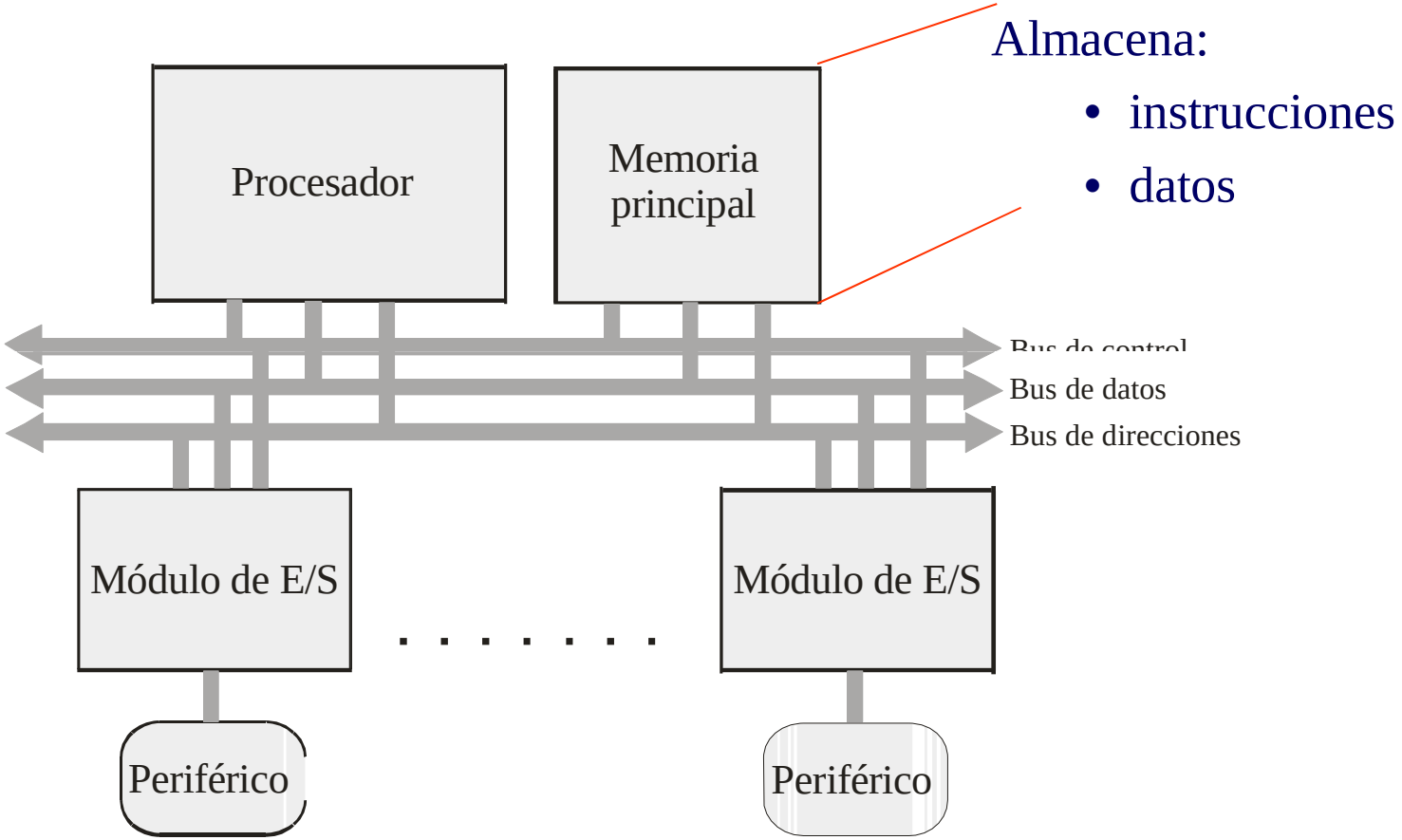
- Máquina capaz de ejecutar una serie de instrucciones elementales: instrucciones máquina
  - Instrucciones almacenadas en memoria
  - Leídas
  - Ejecutadas



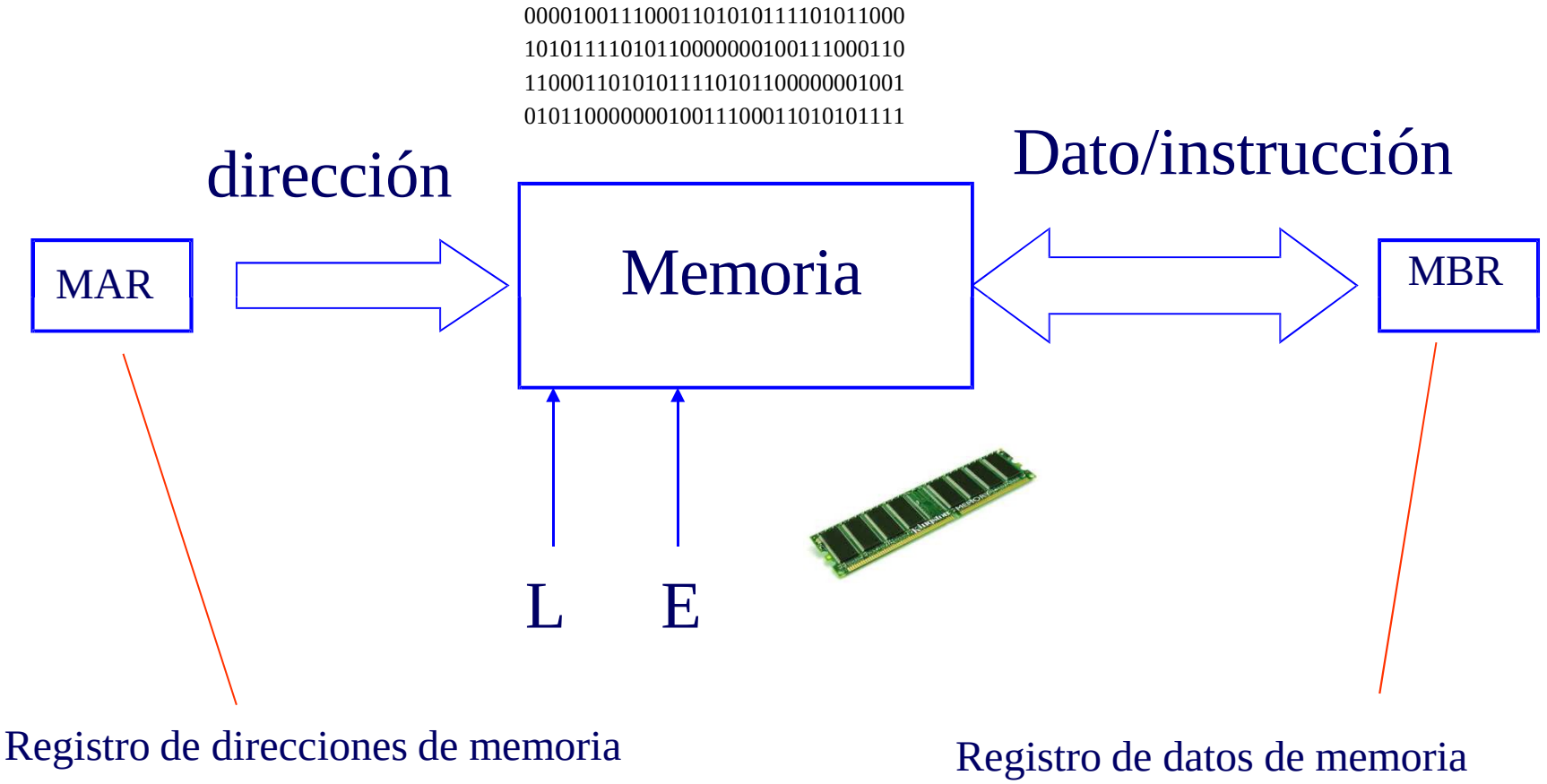
Arquitectura Von Neumann



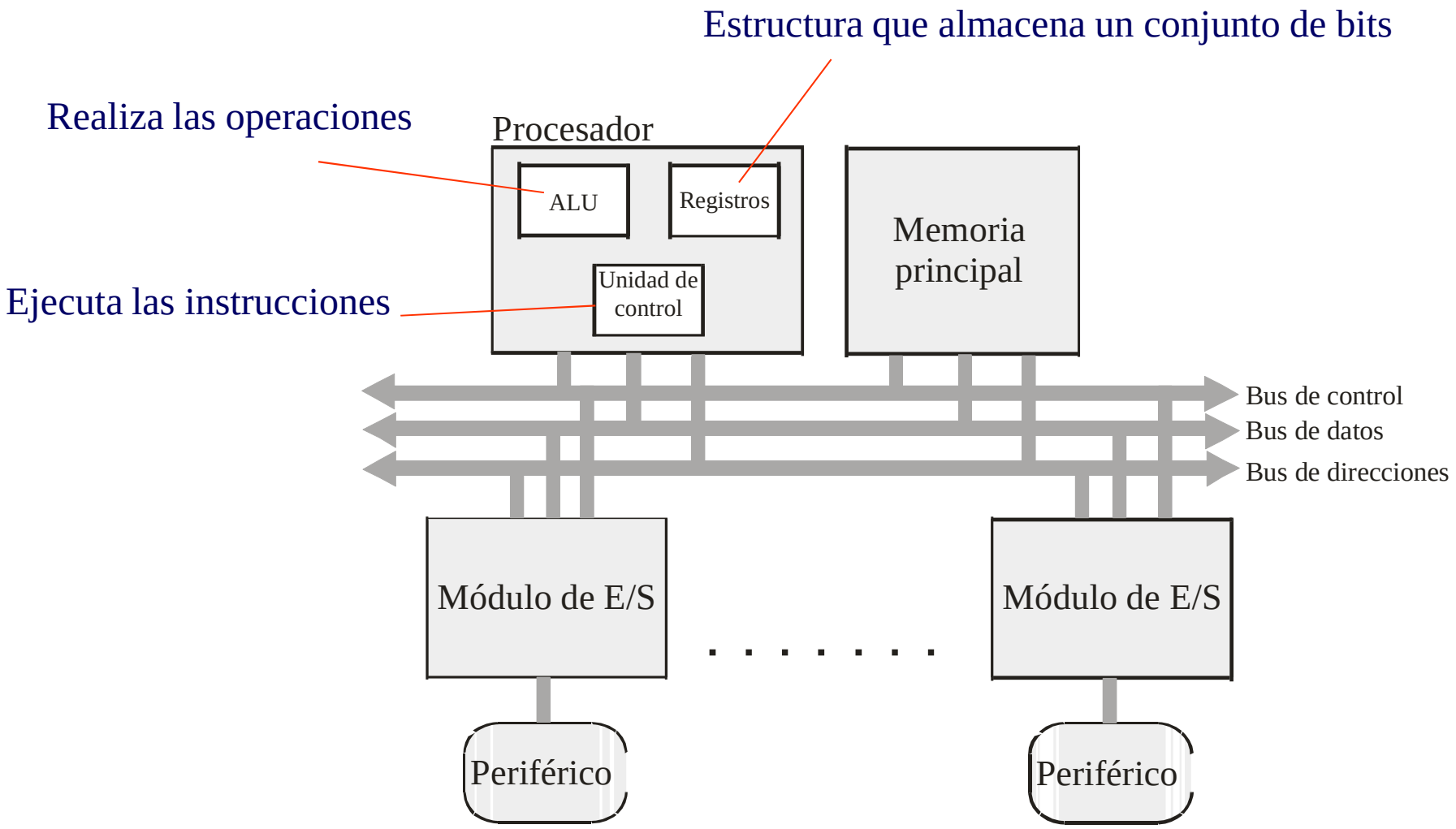
Memoria


Procesador
Memoria principal
Almacena:
• instrucciones
• datos
Bus de control
Bus de datos
Bus de direcciones
Módulo de E/S
Módulo de E/S
Periférico
Periférico

Memoria


00001001110001101010111101011000
10101111010110000000100111000110
11000110101011110101100000001001
01011000000010011100011010101111
dirección
Dato/instrucción
MAR
Memoria
MBR
L
E
Registro de direcciones de memoria
Registro de datos de memoria

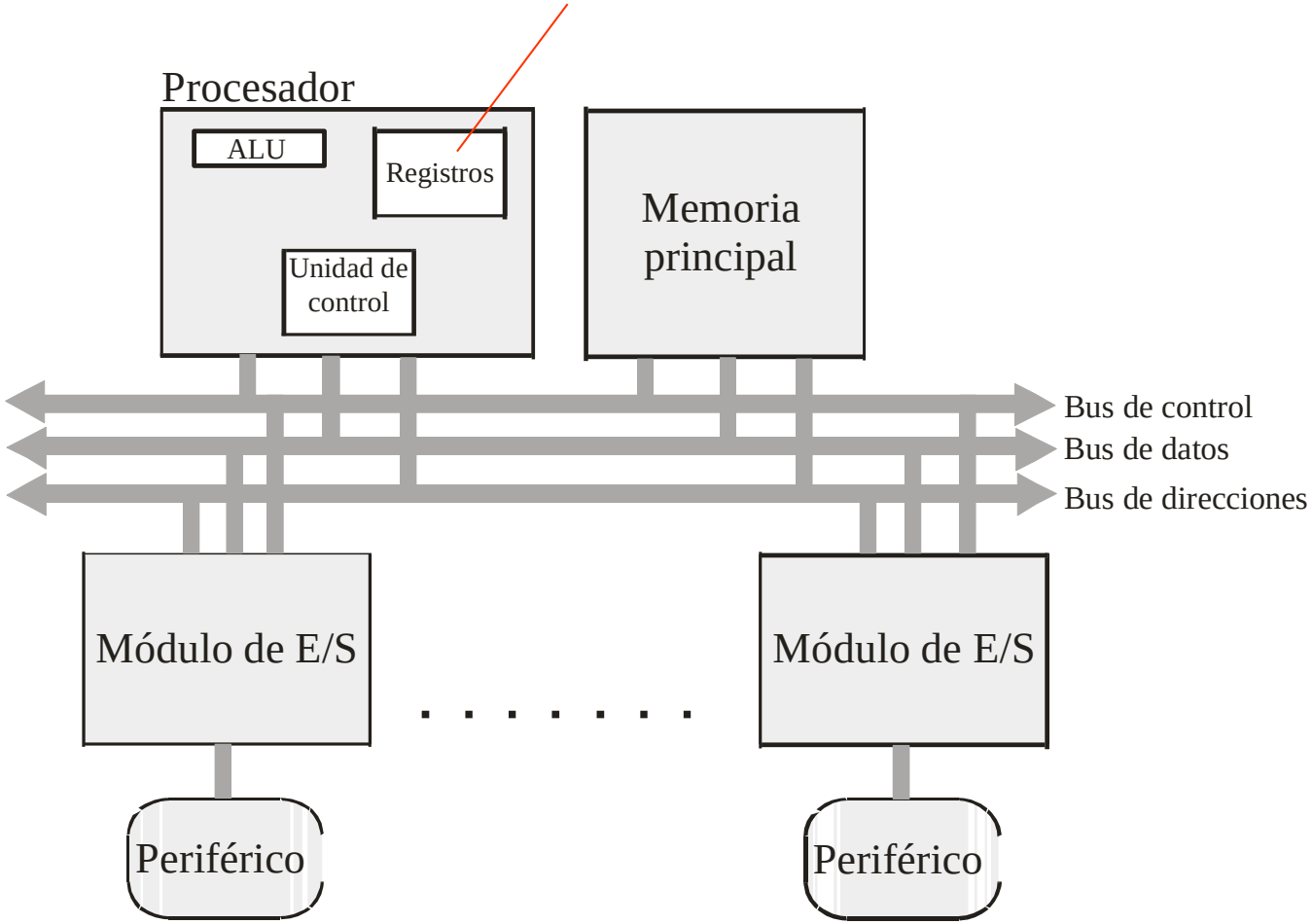
Procesador


Estructura que almacena un conjunto de bits
Realiza las operaciones
Procesador
ALU
Registros
Memoria principal
Unidad de control
Ejecuta las instrucciones
Bus de control
Bus de datos
Bus de direcciones
Módulo de E/S
Módulo de E/S
Periférico
Periférico

Registros especiales

PC: contador de programa

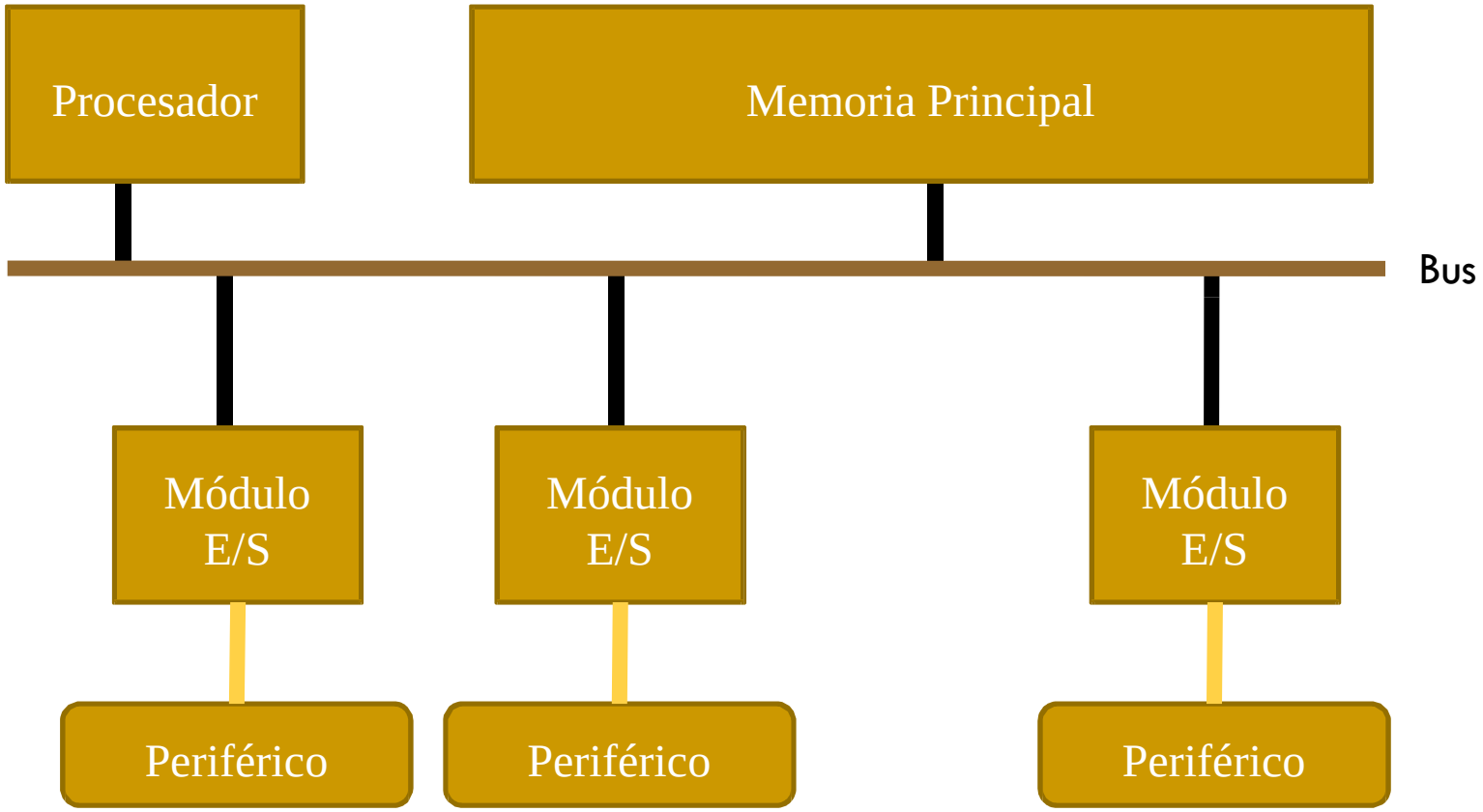
RI: registro de instrucción



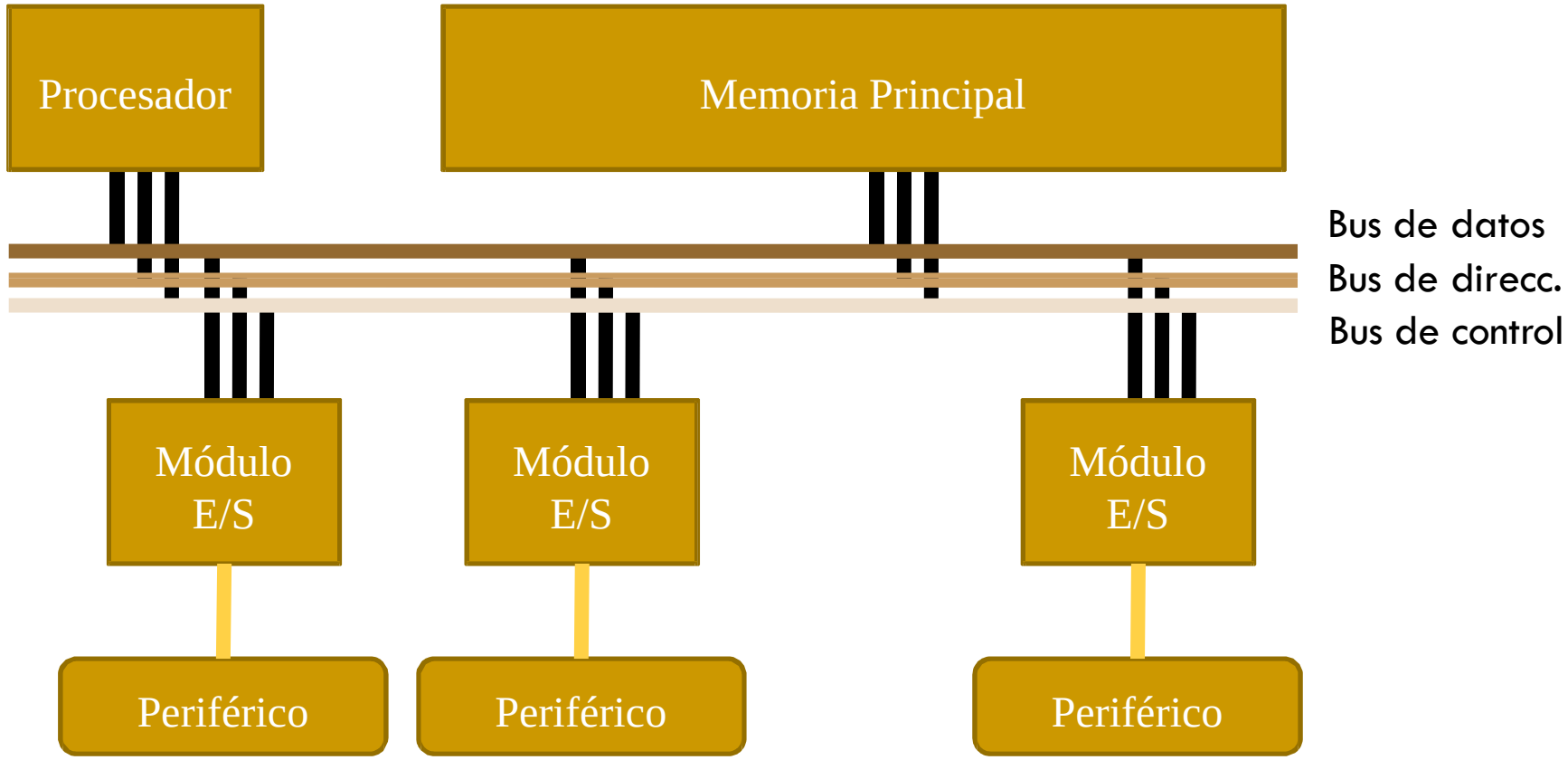
Unidad aritmético-lógica

- Realiza operaciones elementales sobre los datos:
  - Operaciones aritméticas
  - Operaciones lógicas

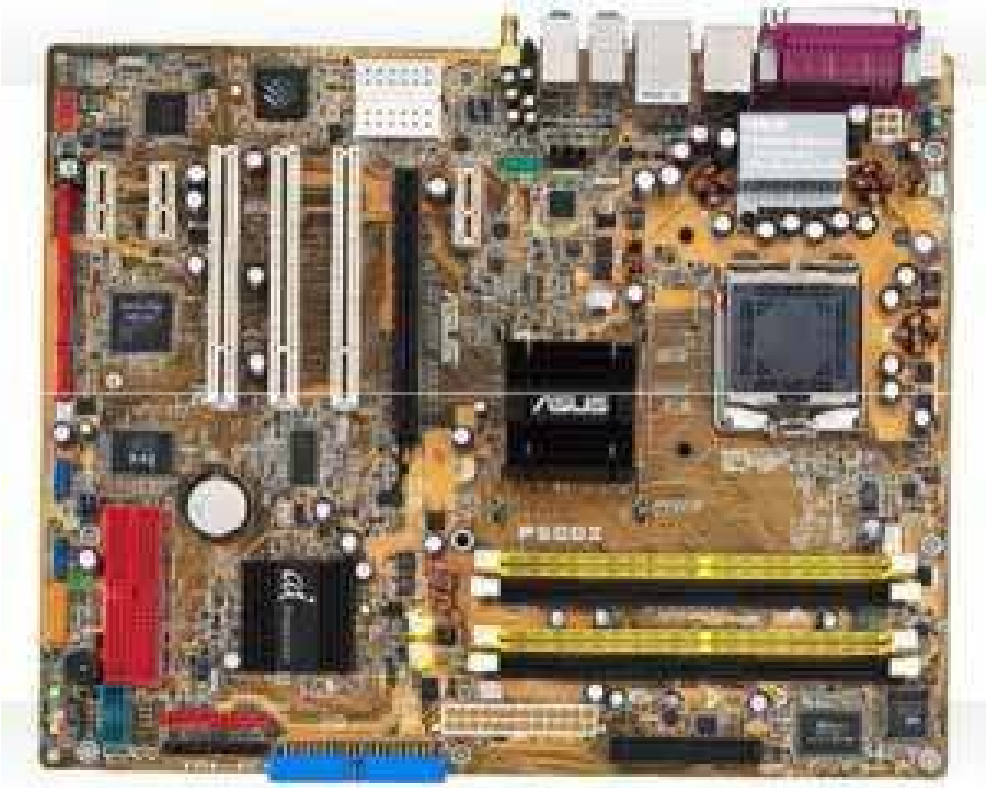
Componentes de un computador



Interconexión



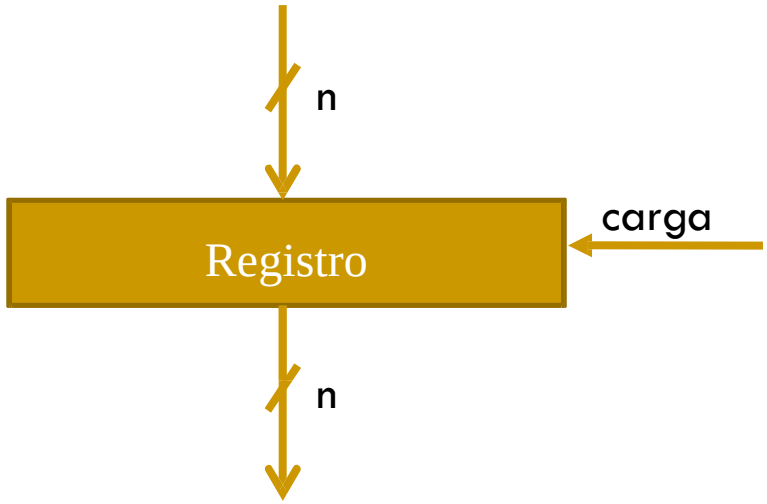
La placa base



Componentes del procesador

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

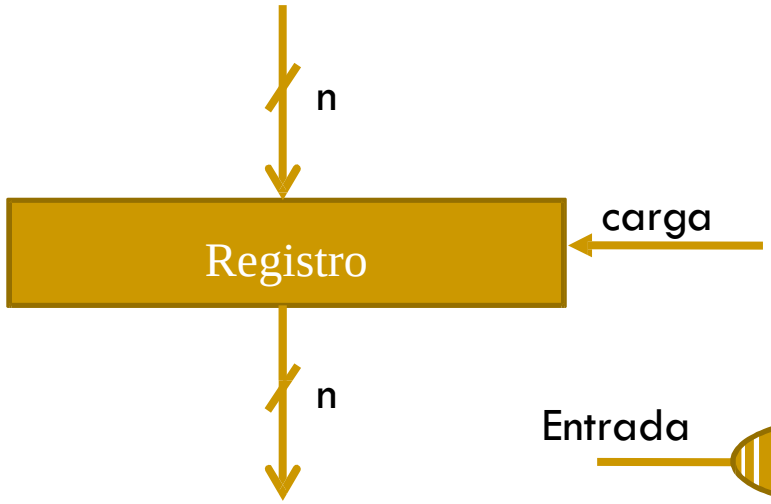
Registros



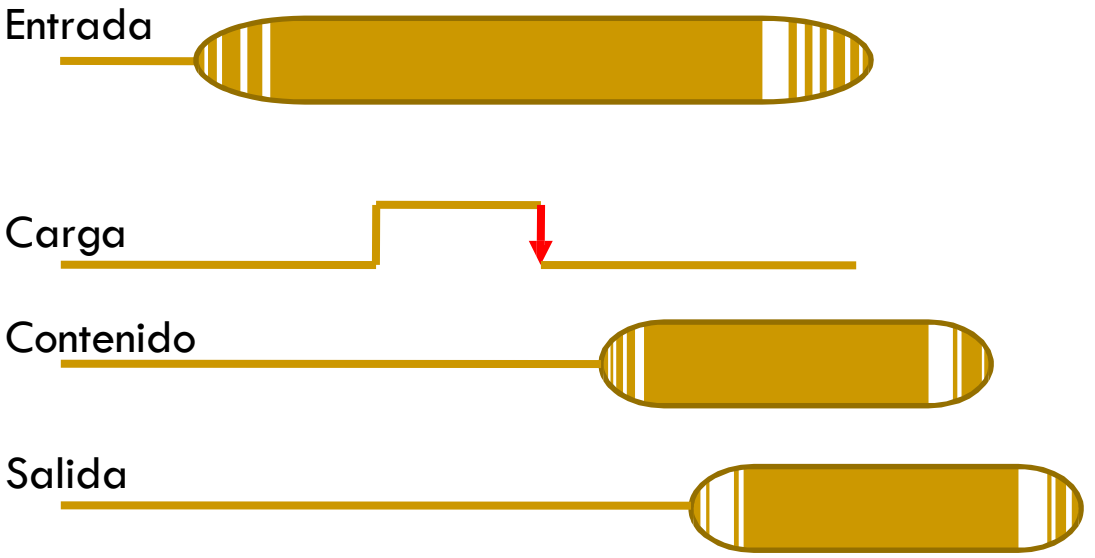
Elemento que almacena un conjunto de bits

Puede existir otra señal para poner el registro a cero

Registros



Puede existir otra señal para poner el registro a cero



Tipos de registros

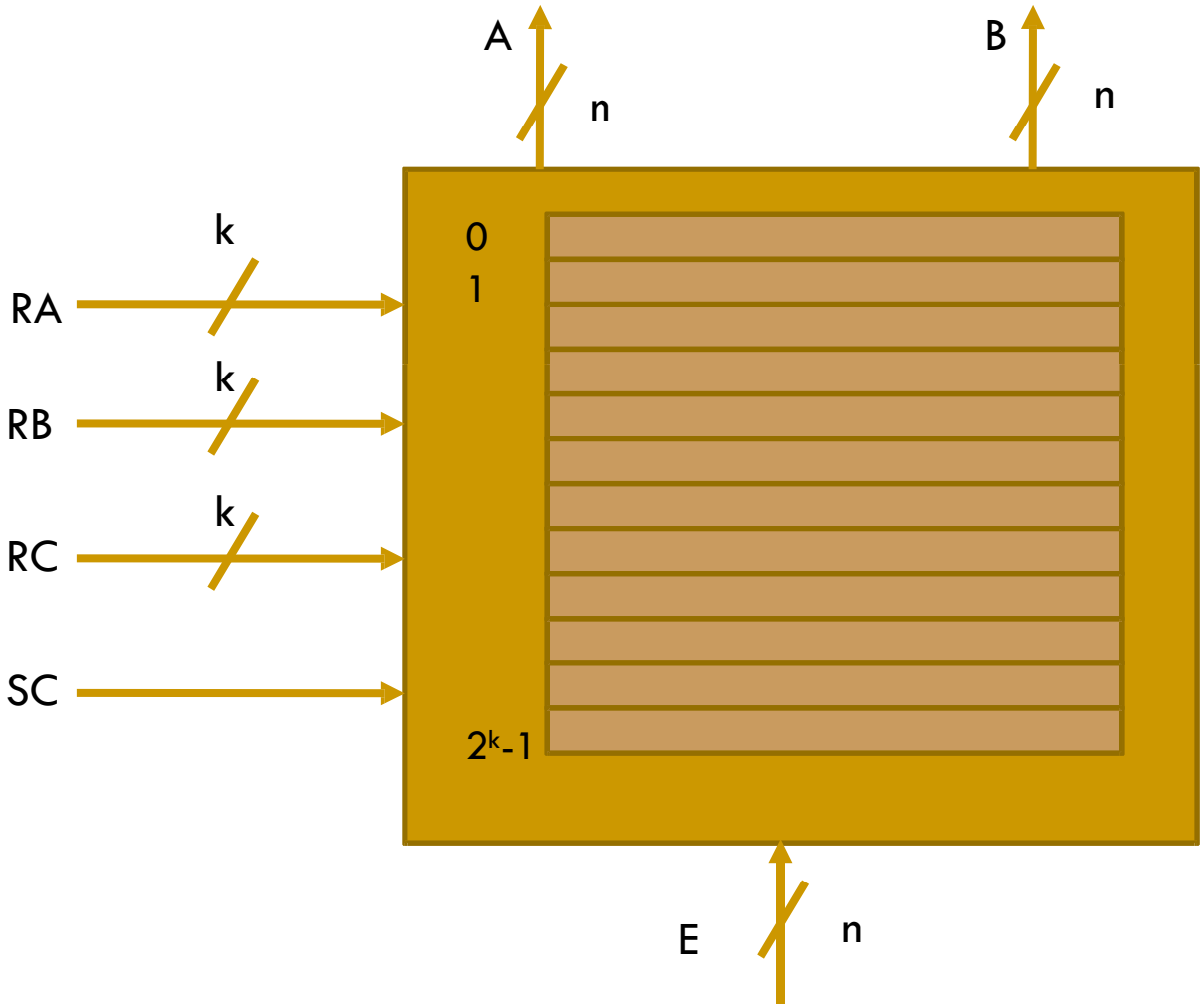
- Visibles al programador
- Registros no visibles
  - Registros temporales

- Registros de control y estado
  - Contador de programa, PC
  - Registro de instrucción, RI
  - Registro de direcciones de memoria, MAR
  - Registro de datos de memoria, MBR
  - Registro de estado, RE

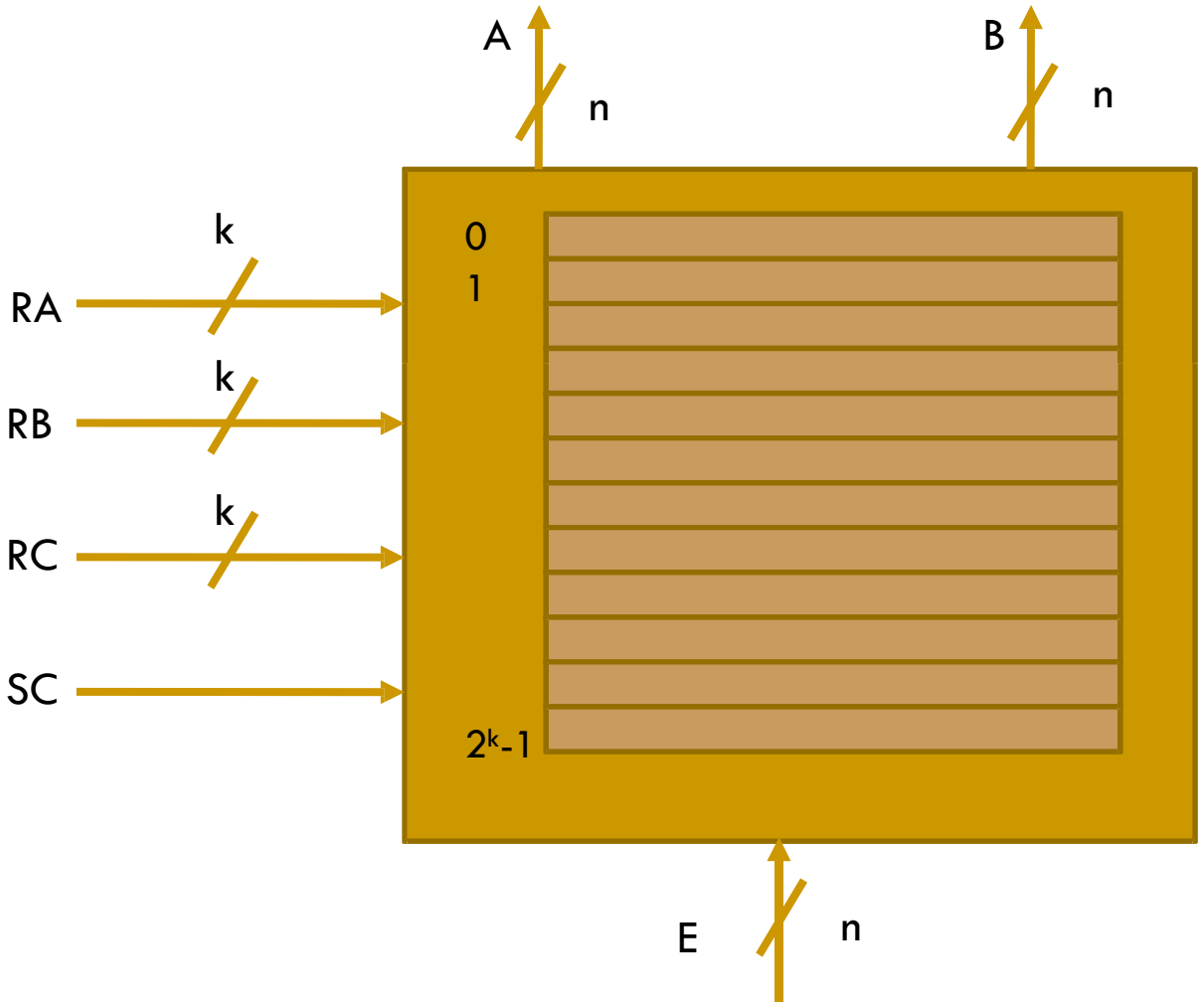
Banco de registros

- Agrupación de registros.
- Típicamente un número de registros potencia de 2.
  - n registros → log2n bits para seleccionarlos.
  - k bits de selección podemos selccionar 2^k registros.
- Elemento fundamental de almacenamiento.
  - Acceso muy rápido.

Banco de registros



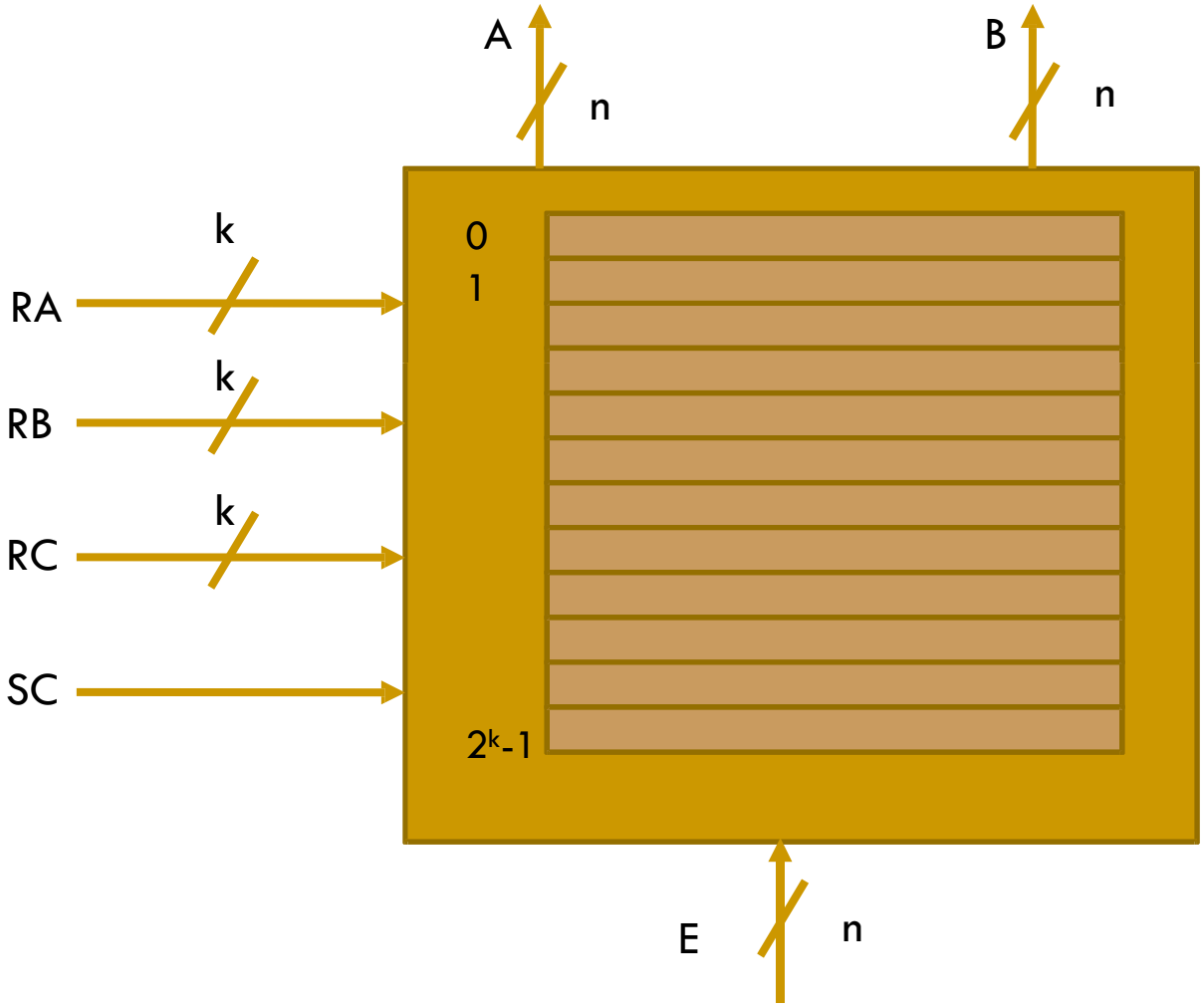
Banco de registros



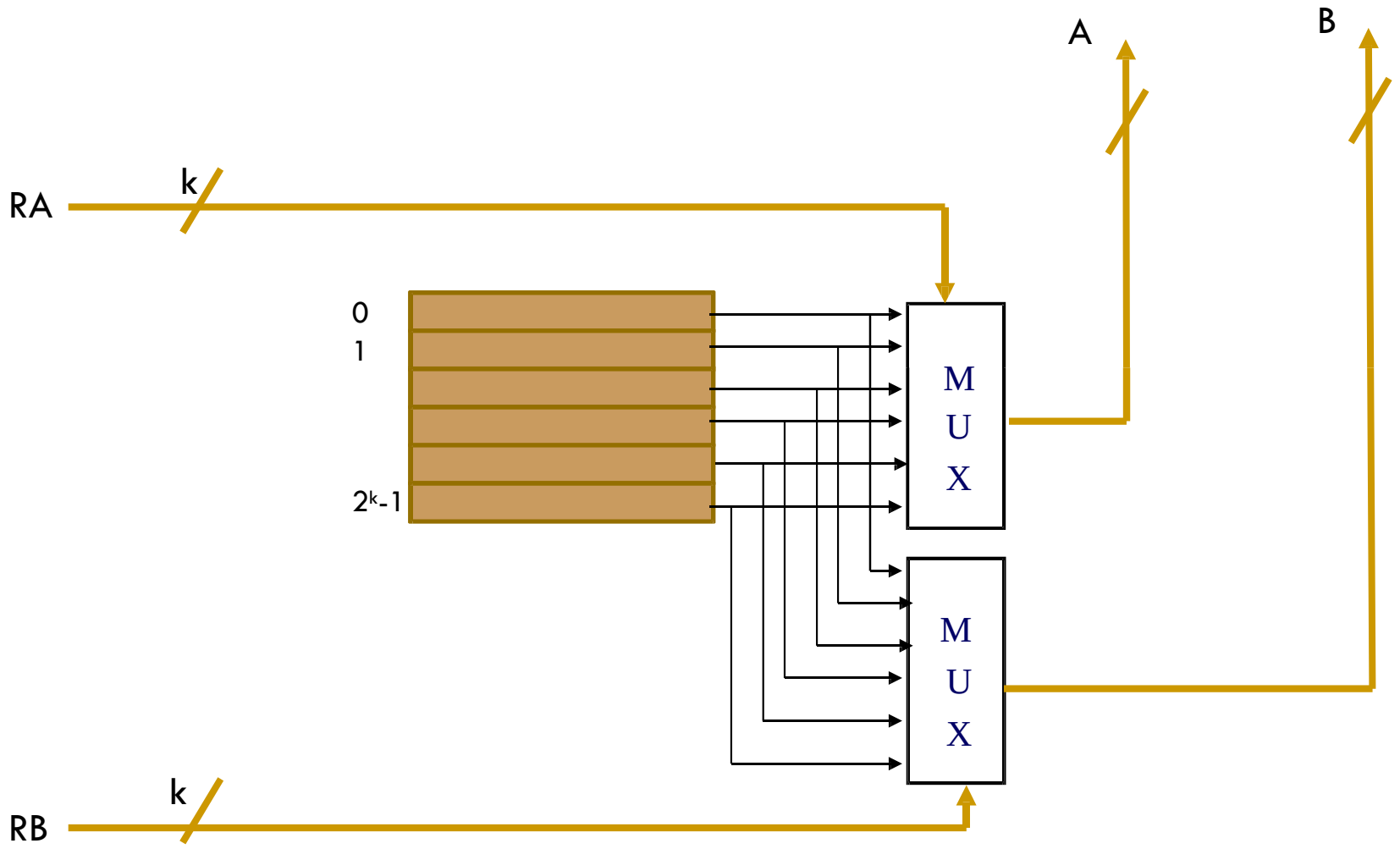
Con 32 registros, k=5

Banco de registros

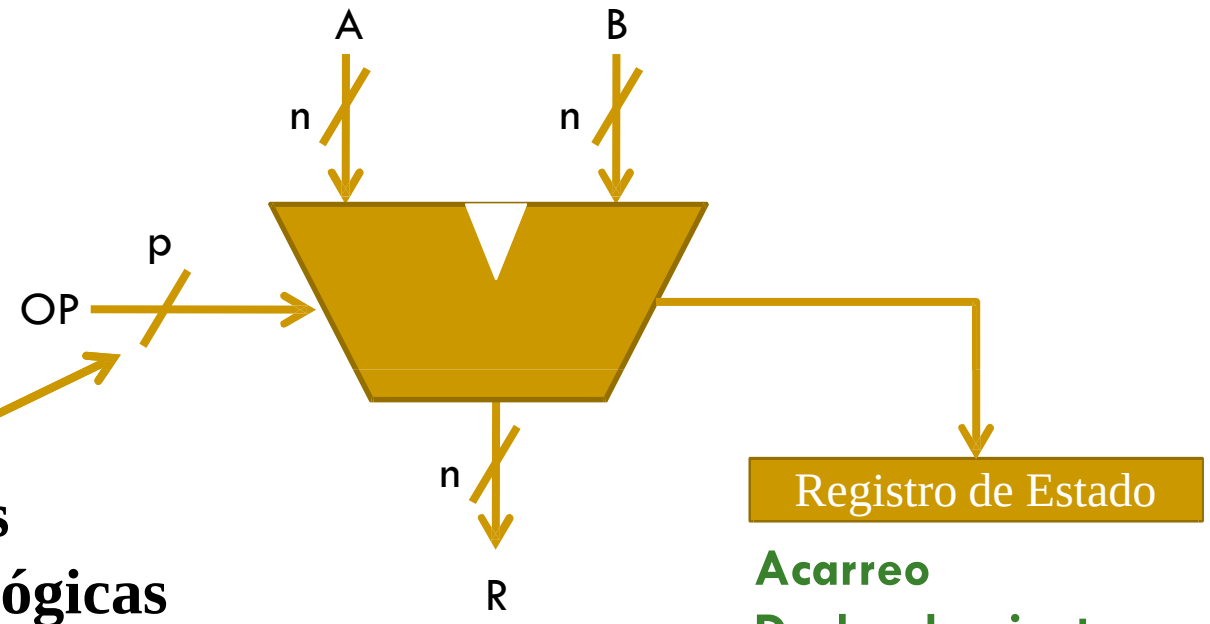
¿Qué valor tiene que tener RA para sacar por A el contenido del registro 14?



Esquema para lectura



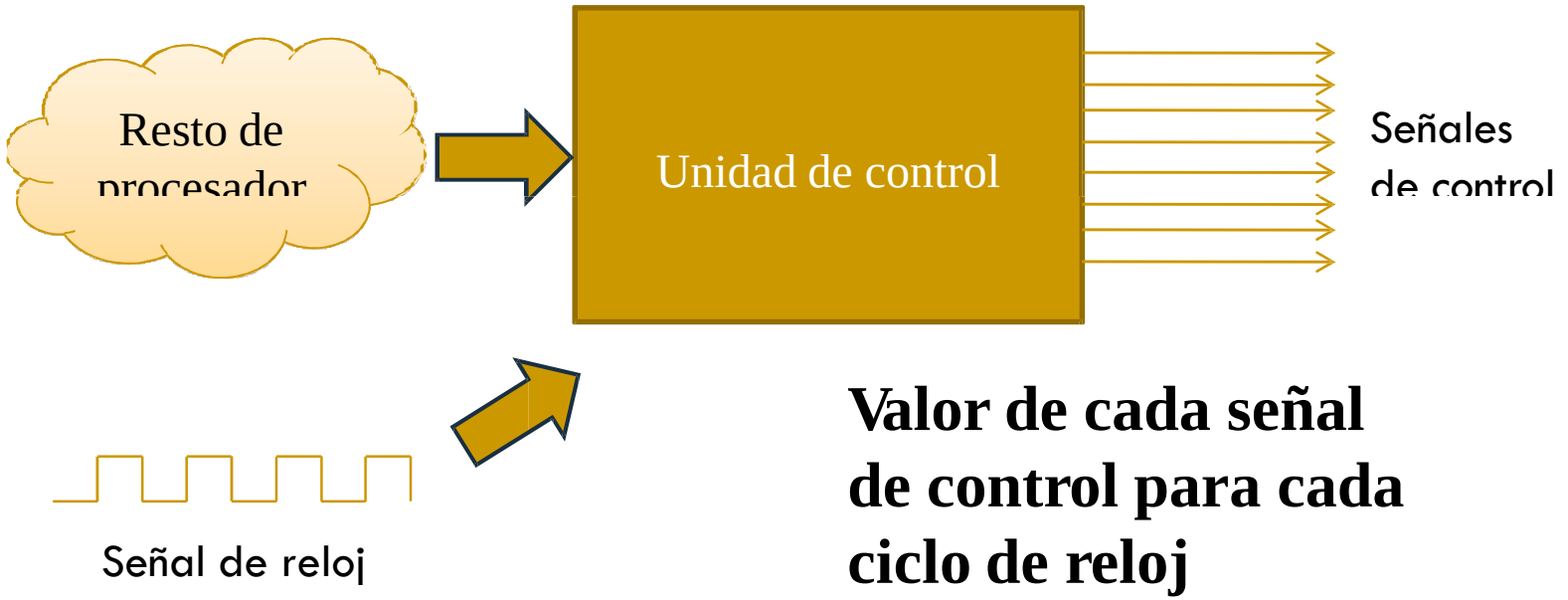
Unidad aritmético lógica



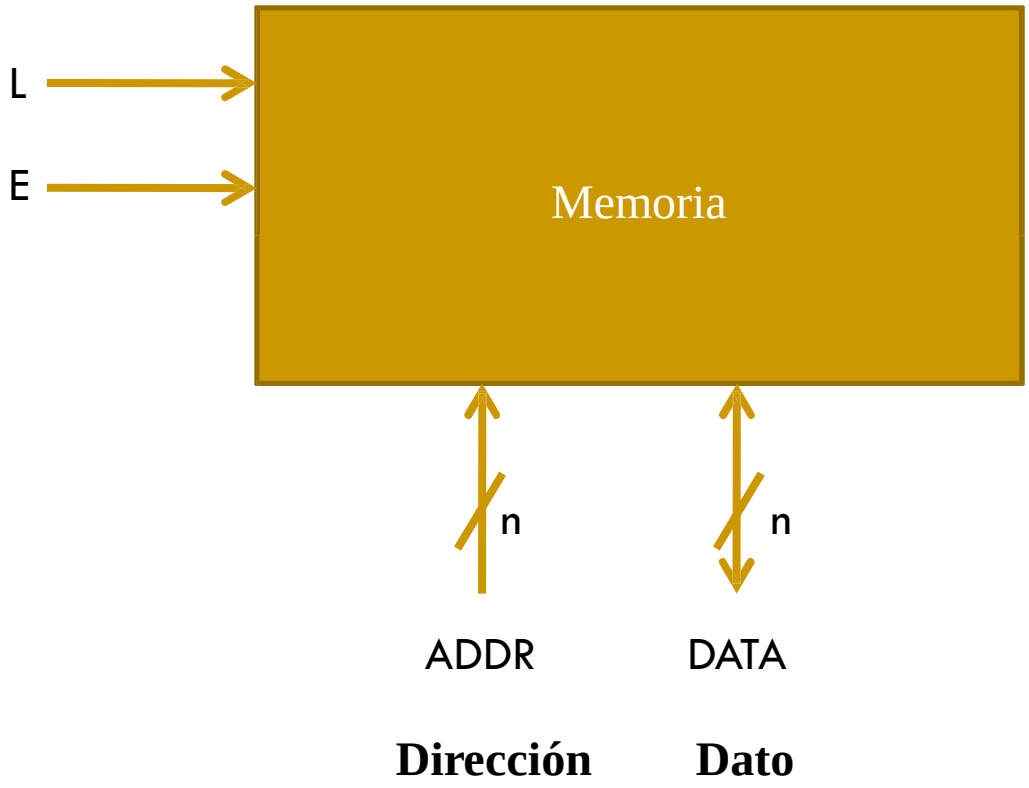
2^p operaciones aritméticas y lógicas distintas

- Acarreo
- Desbordamiento
- Cero
- Positivo
- Negativo
- …

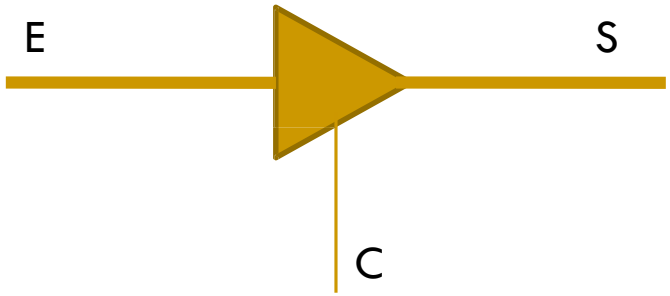
Unidad de control



Acceso a la memoria

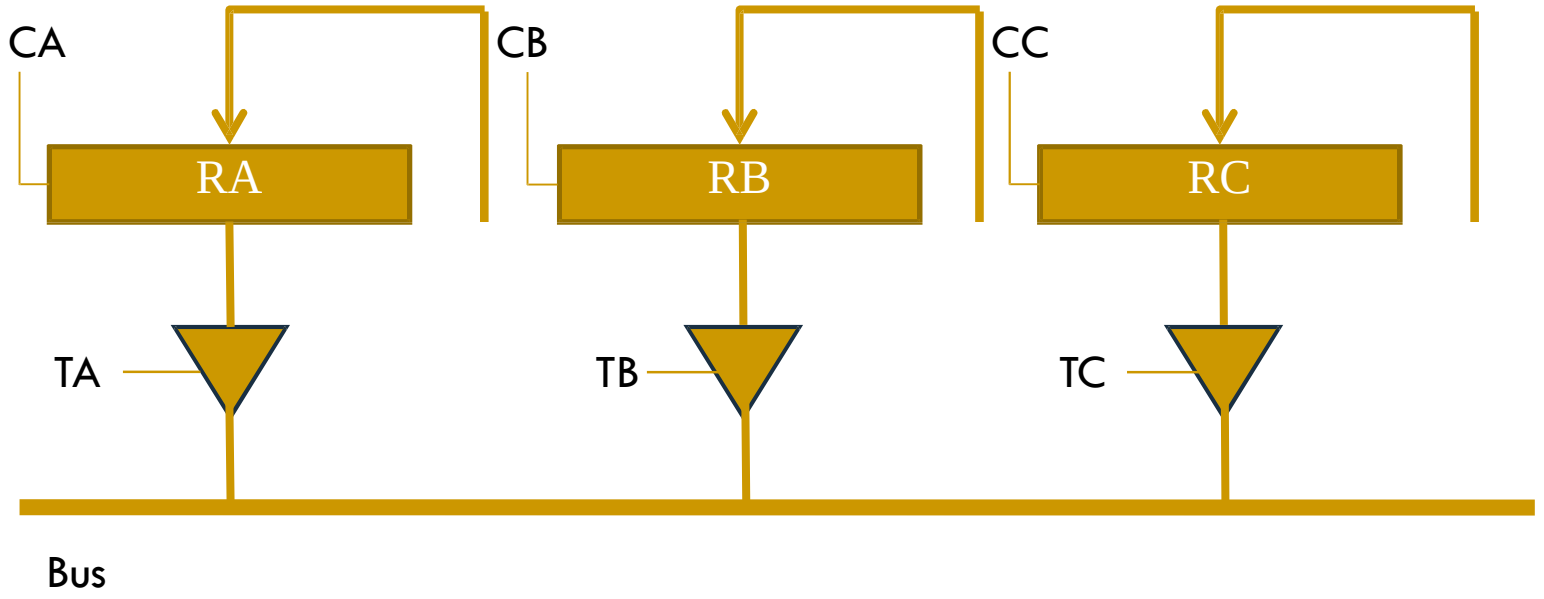


Búfer triestado



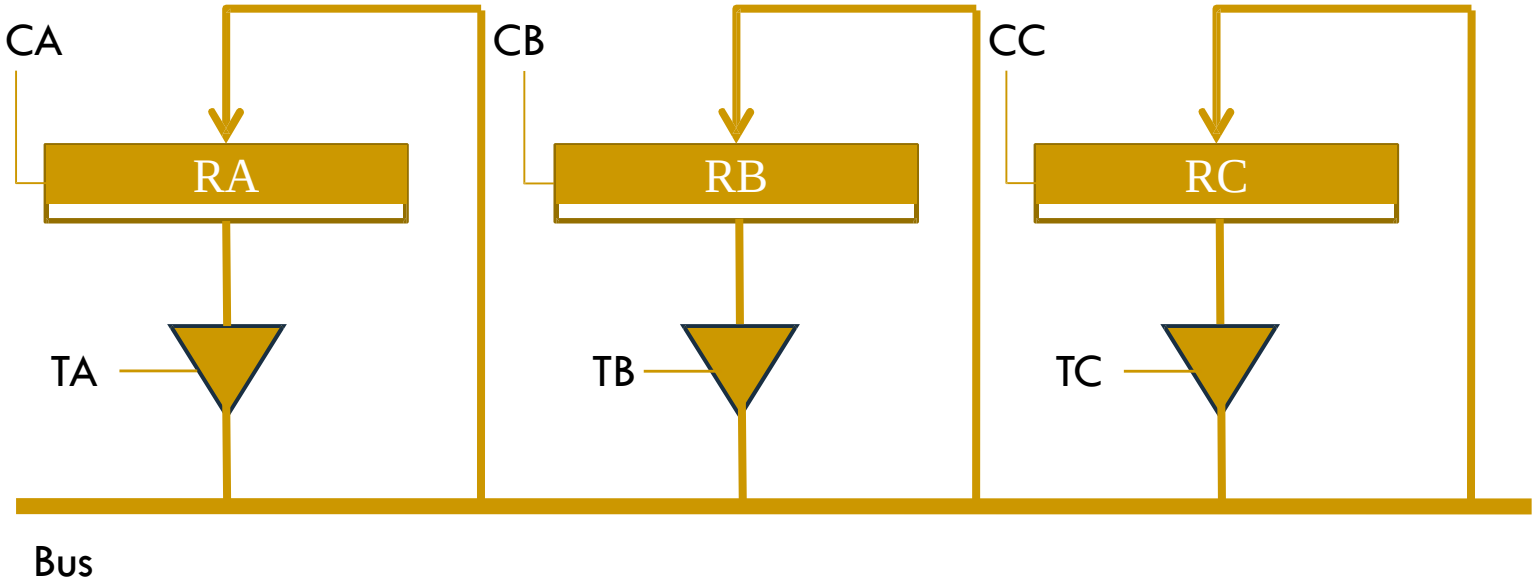
E	C	S
0	0	Z
0	0	Z
0	1	0
1	1	1

Acceso a un bus

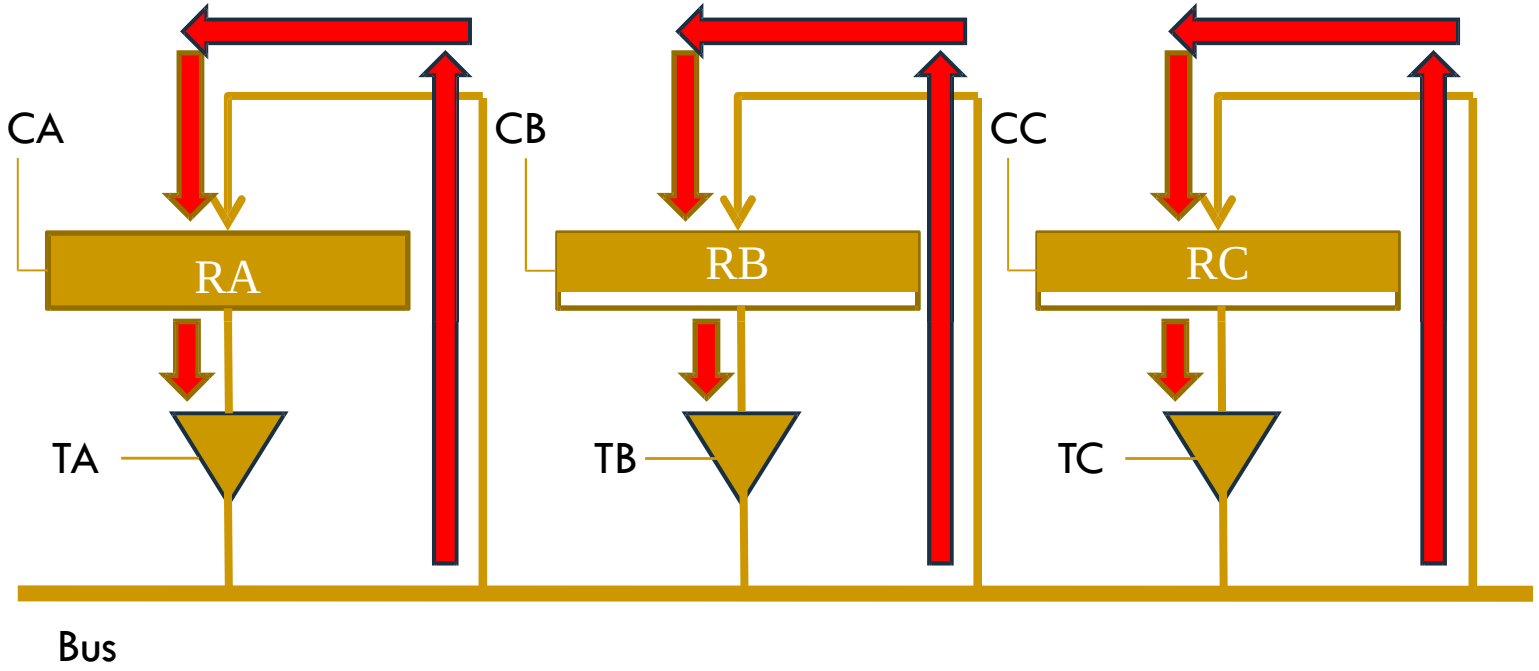


Ejemplo

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

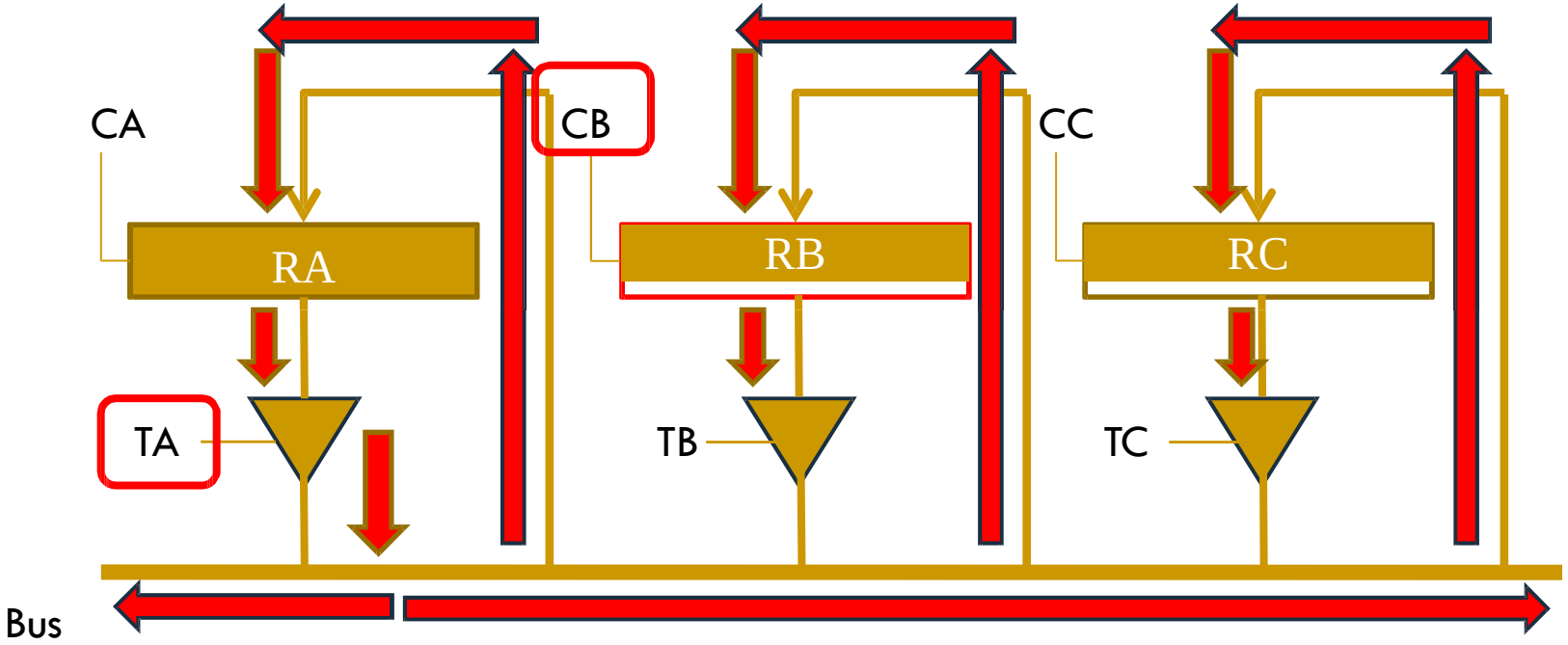


Camino de datos (RB ← RA)

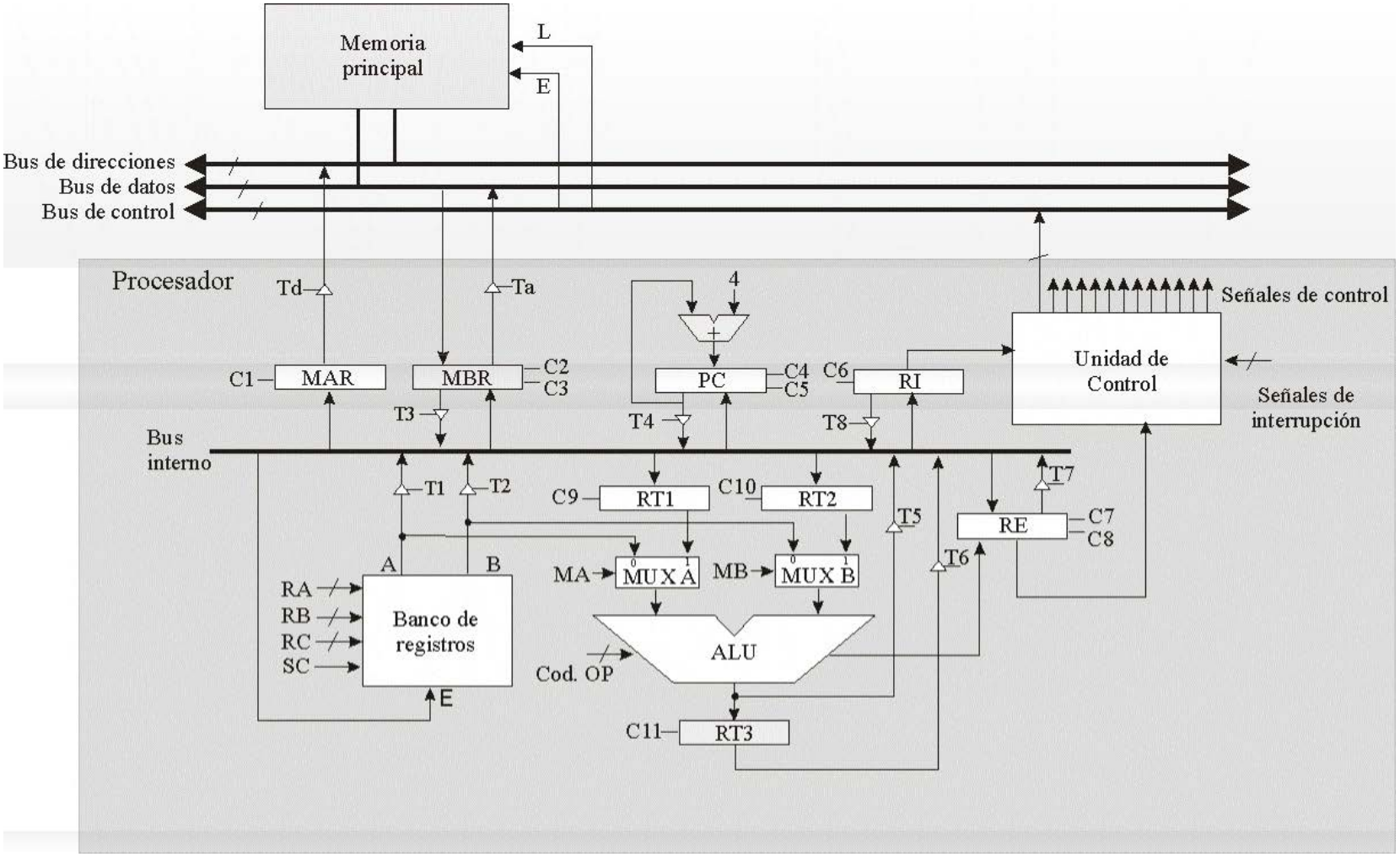


Situación con todas las señales desactivadas

Camino de datos (RB ← RA)



Estructura de un computador y procesador basado en bus



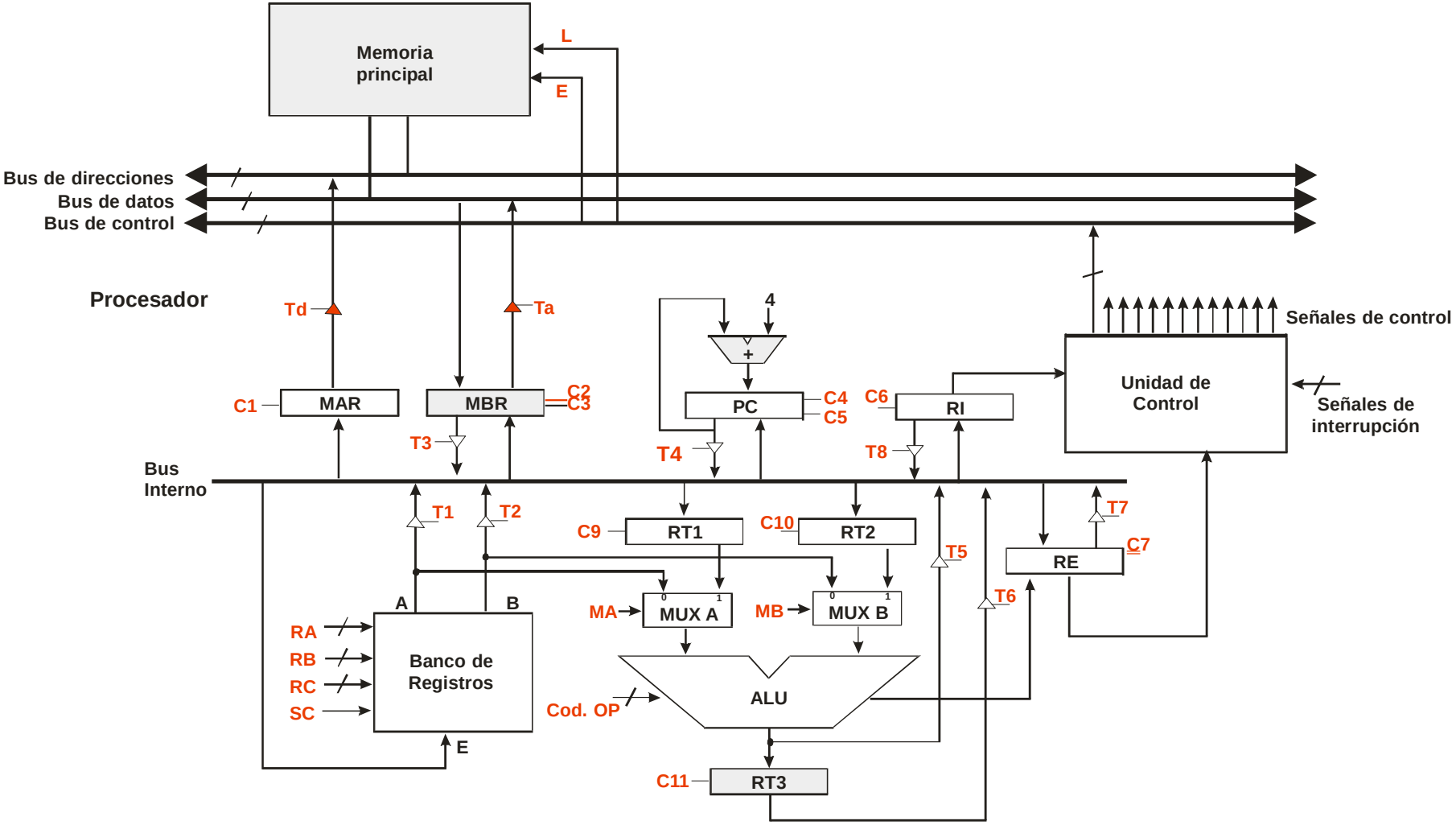
Otros registros

- PC: contador de programa
- RI: registro de instrucción
- SP: puntero de pila
- MAR: registro de direcciones de memoria
- MBR: registro de datos de memoria
- RE: registro de estado
- Registros transparentes al usuario: RT1, RT2, RT3.

Características

- Computador de 32 bits
- La memoria se direcciona por bytes
  - Un ciclo de lectura y escritura
- Banco de 32 registros visibles
  - R0..R31
  - Asumir como en el MIPS R0 = 0 y SP = R29
- Registros temporales
  - RT1, RT2, RT3: no visibles
- Otros registros de control y estado
  - MAR, MBR, PC, RE, RI

Señales de control

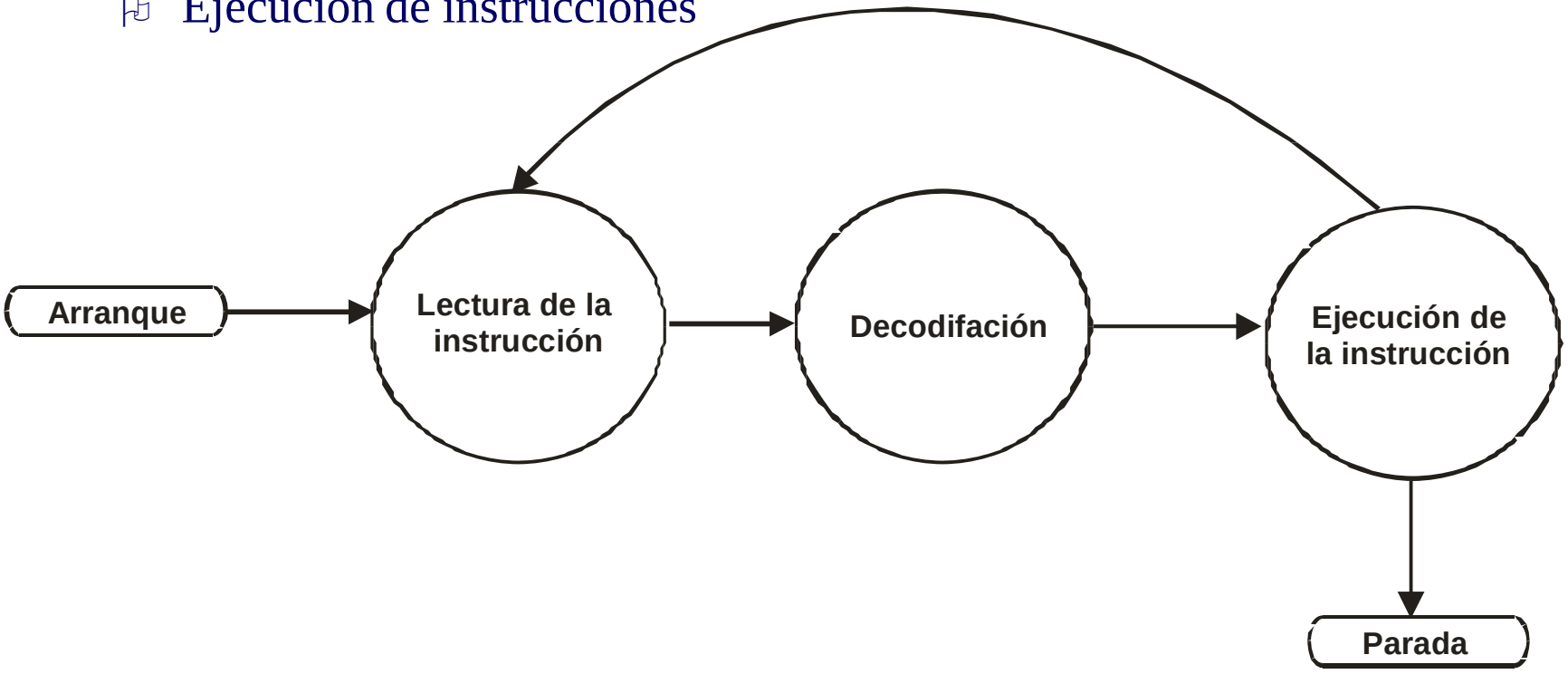


Señales de control

- 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 (RA, RB, RC y SC)

Unidad de control

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



Lenguaje nivel RT

- Lenguaje de nivel de transferencia de registros.
  - Registro1 ← Registro2

- Especifica lo que ocurre en el computador mediante transferencias de datos entre registros.

Operaciones elementales

- Operaciones de transferencia
  - MAR ← PC
- Operaciones de proceso
  - R1 ← R2 + RT2

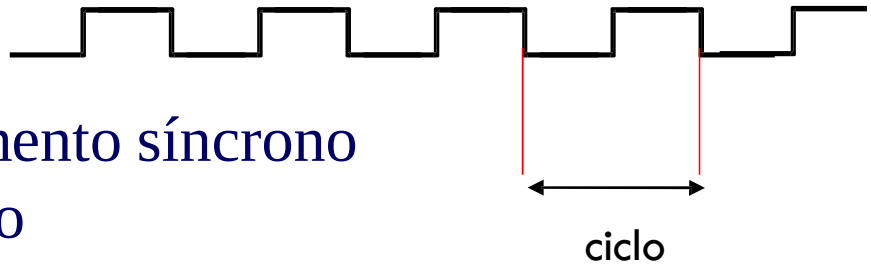
- Lenguaje RT
  - Lenguaje de nivel de transferencia de registros.
  - Especifica lo que ocurre en el computador mediante transferencias de datos entre registros.

Señales de control

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

Op. Elemental	Señales de control activadas (resto 0)
MAR ← PC	T4, C1
R1 ← R2 + RT1	RB = 00011 (R2) MA = 1 MB = 0 Cod Op. = SUMA T5 RC = 00001 (R1) SC

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 ← PC y RT3 ← RT2 + RT1
- En un mismo ciclo no se puede realizar
  - MAR ← PC y R1 ← RT3 ¿por qué?

Ejercicio

- ¿Cuál es la duración del ciclo de un computador con una frecuencia de reloj de 1 GHz?

Fases de la ejecución de instrucciones

- 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.

Ejemplo: Lectura de la instrucción

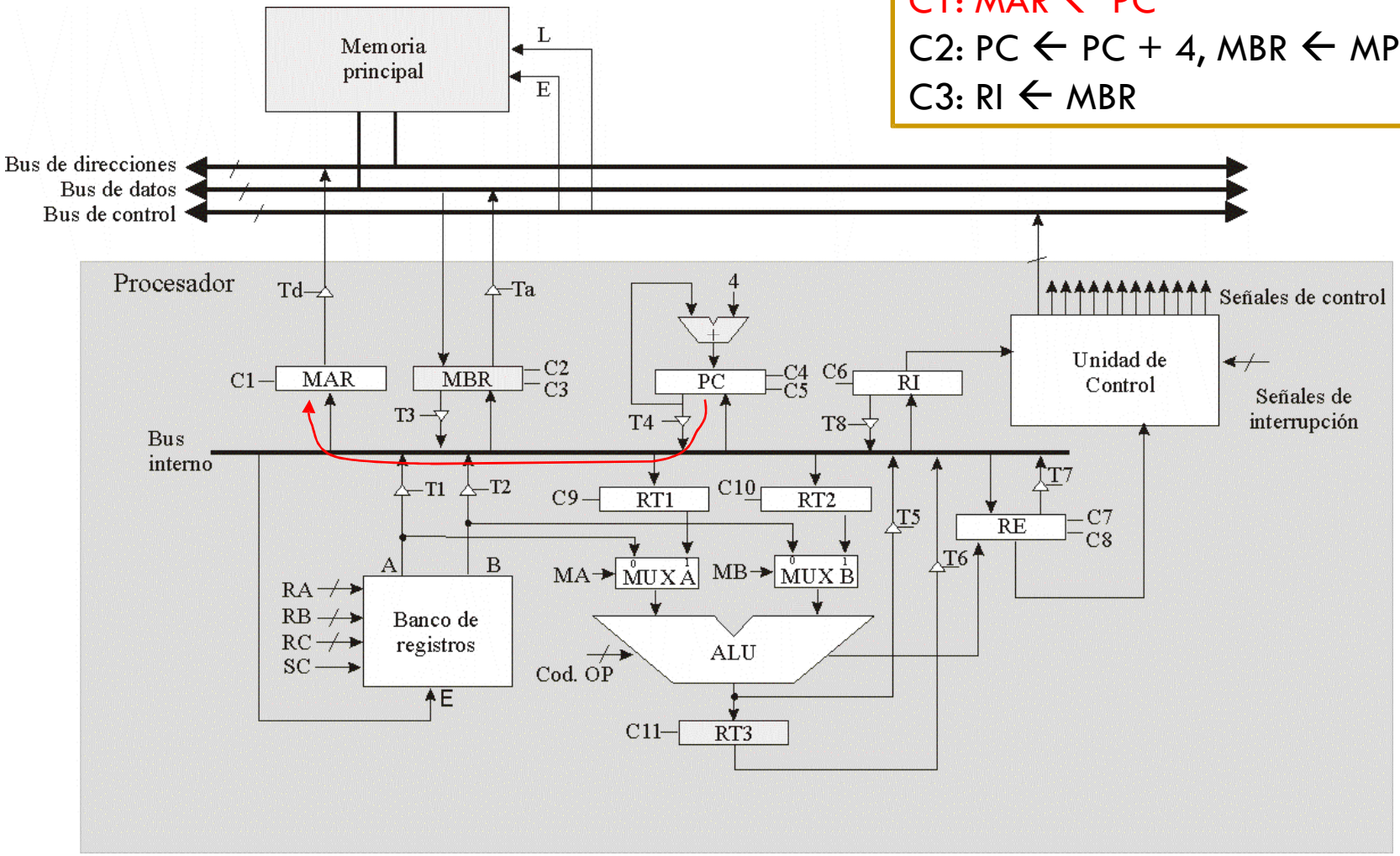
C1: MAR ← PC
C2: PC ← PC + 4
C3: MBR ← MP
C4: RI ← MBR

C1: MAR ← PC
C2: PC ← PC + 4, MBR ← MP
C3: RI ← MBR

Posibilidad de operaciones simultáneas

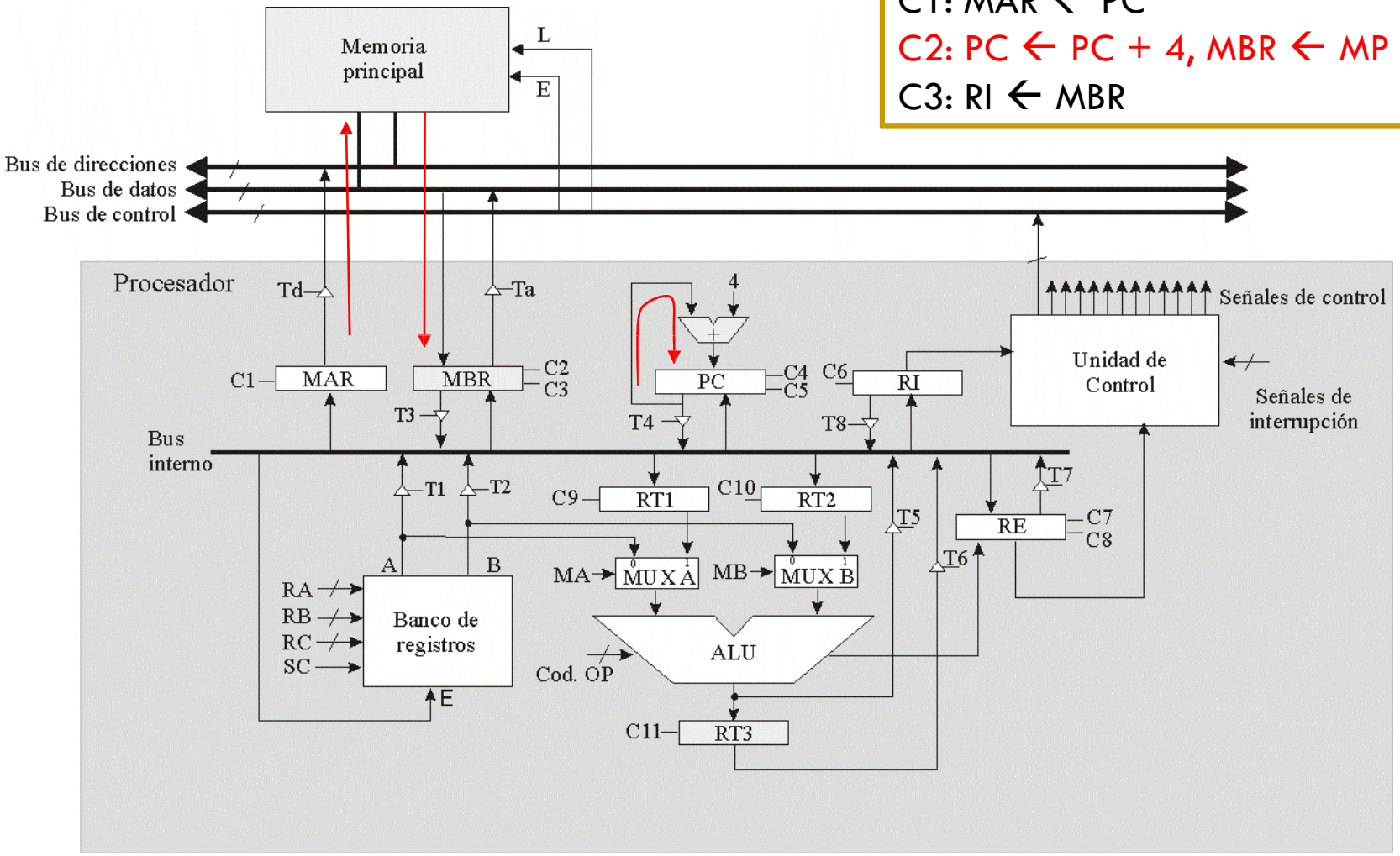
Lectura de la instrucción (c1)

C1: MAR ← PC
C2: PC ← PC + 4, MBR ← MP
C3: RI ← MBR



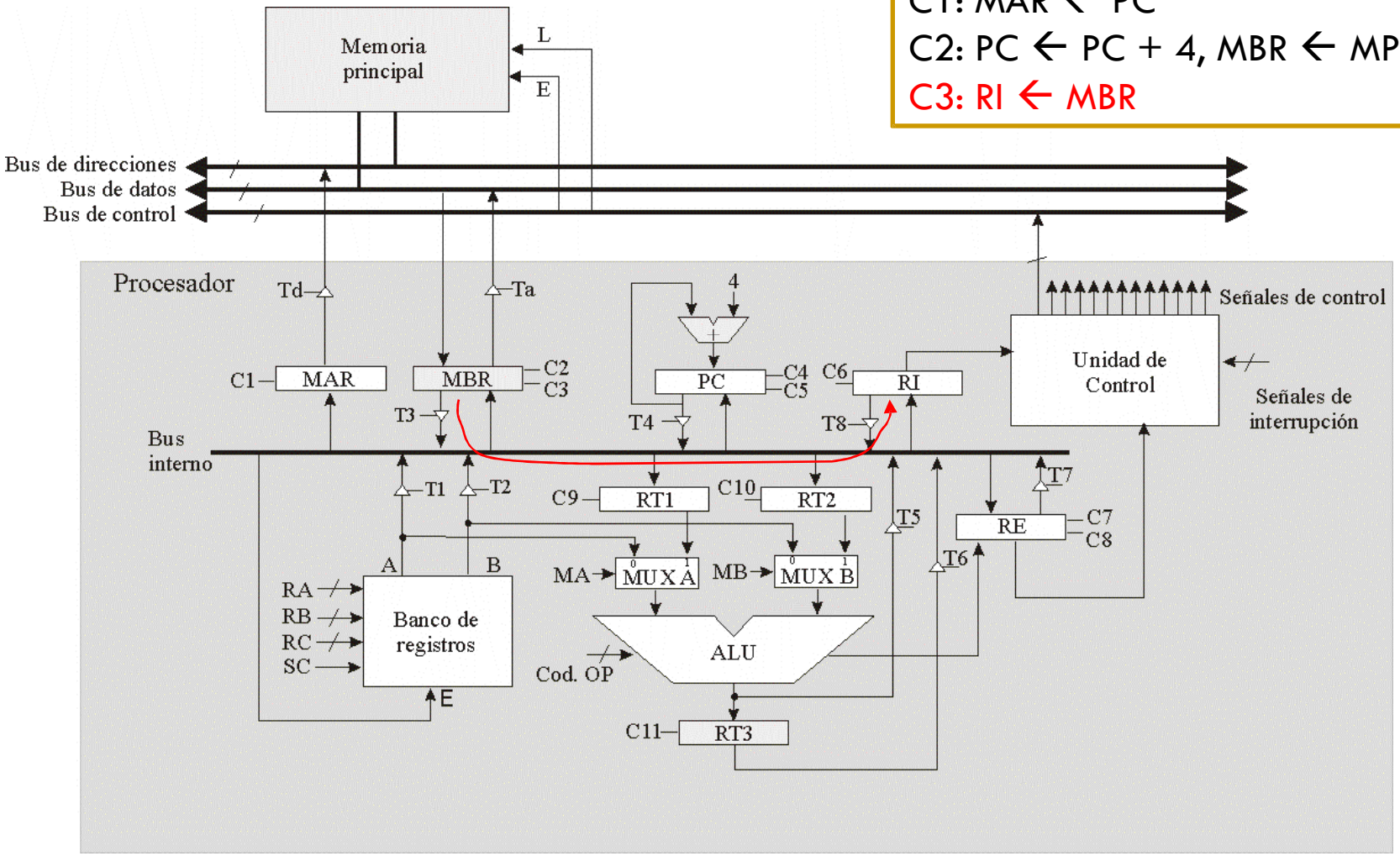
Lectura de la instrucción (c2)

C1: MAR ← PC
C2: PC ← PC + 4, MBR ← MP
C3: RI ← MBR



Lectura de la instrucción (c3)

C1: MAR ← PC
C2: PC ← PC + 4, MBR ← MP
C3: RI ← MBR



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.

C1: MAR ← PC
C2: PC ← PC + 4, MBR ← MP
C3: RI ← MBR

C1: T4, C1
C2: C4, Td, L, C2,
C3: T3, C6

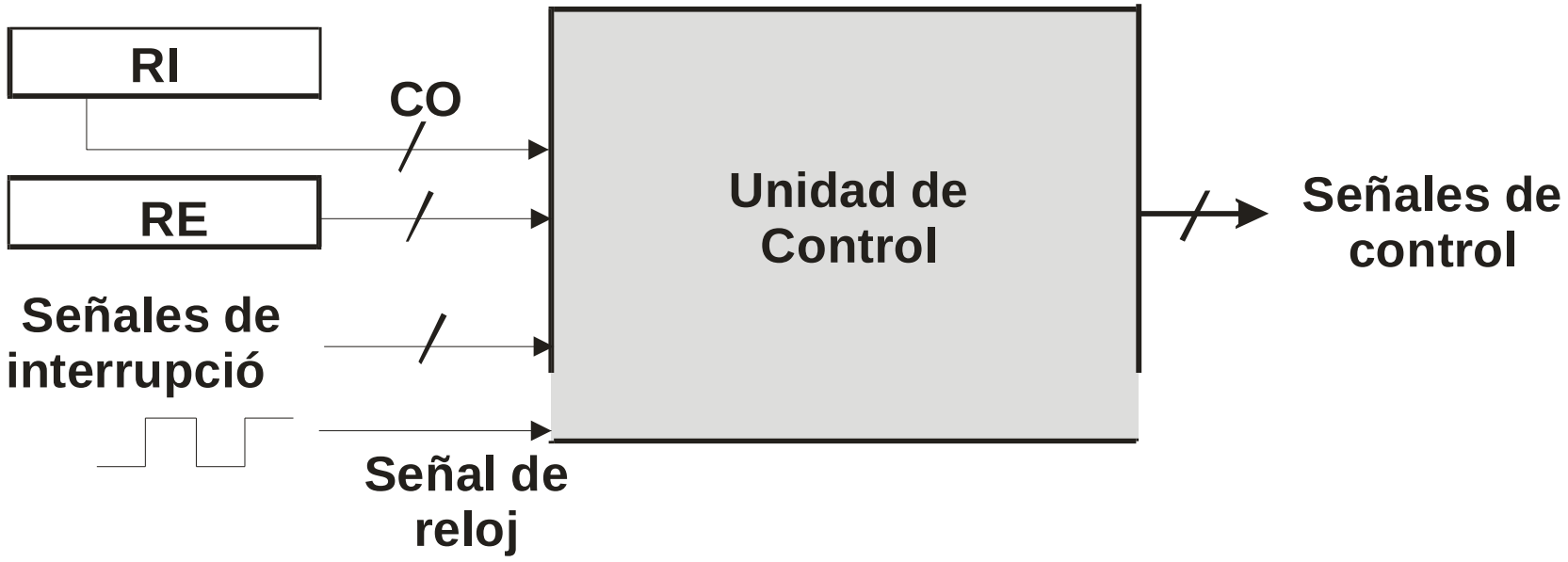
Ejecución completa de lw $reg, dir

C1: MAR ← PC
C2: PC ← PC + 4, MBR ← MP
C3: RI ← MBR
C4: Decodificación
C5: MAR ← RI(dir)
C6: MBR ← MP
C7: $Reg ← MBR

Ejercicio: operaciones elementales para otras instrucciones

- Instrucciones que caben en una palabra:
  - `sw $reg, dir`
  - `add $rd, $ro1, $ro2`
  - `addi $rd, $ro1, inm`
  - `lw $reg1, desp($reg2)`
  - `j dir`
  - `jal dir`
  - `jr $reg`
  - `beq $ro1, $ro2, desp`
- Instrucciones que ocupan varias palabras:
  - `addm R1, dir` R1 ⟵ R1 + MP[dir]

Diseño de la unidad de control



- Cada una de las señales de control es función del valor de:
  - El contenido del RI
  - El contenido de RE
  - El momento del tiempo

Diseño de la unidad de control

- Para cada instrucción máquina:
  - Definir el comportamiento en lenguaje RT en cada ciclo de reloj
  - Traducir el comportamiento a valores de cada señal de control en cada ciclo de reloj
  - Diseñar un circuito que genere el valor de cada señal de control en cada ciclo de reloj

Arranque de un computador

- El *Reset* carga valores predefinidos en registros
  - PC ← dirección de arranque del programa iniciador (memoria ROM)
- Se ejecuta el programa iniciador
  - Test del sistema
  - Carga en memoria el cargador del sistema operativo
  - PC ← dirección del programa cargador del sistema operativo
- Se ejecuta el programa cargador del SO que carga el resto del sistema operativo
- Se pasa a ejecutar el sistema operativo

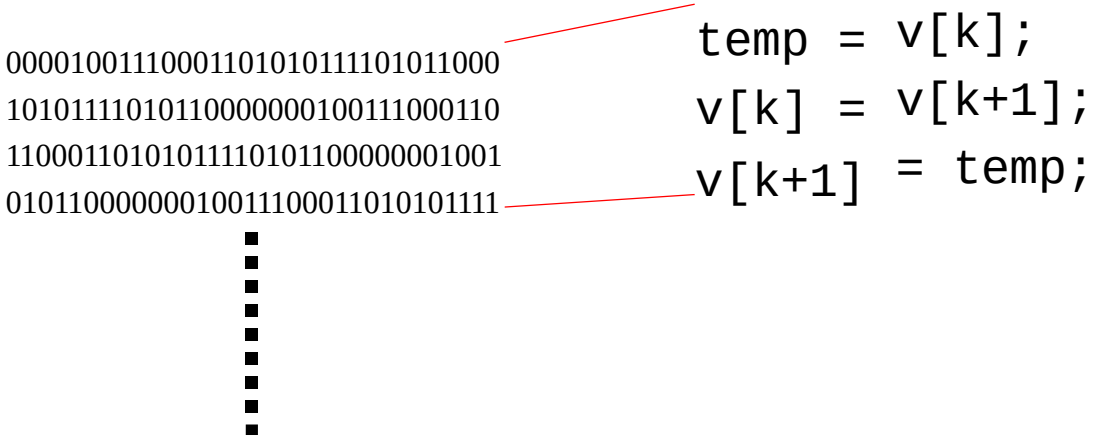
Programa

- Secuencia consecutiva de instrucciones máquina

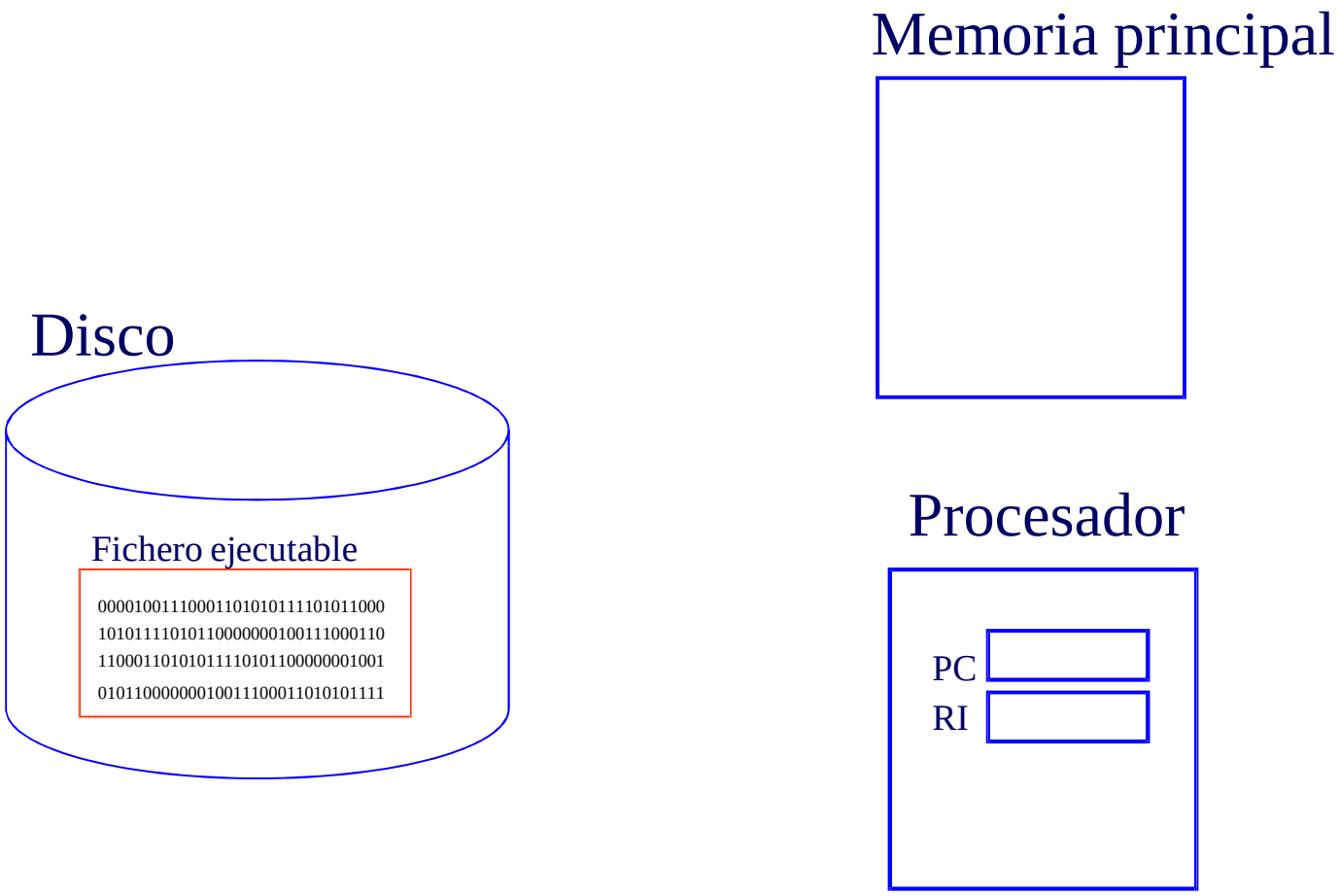
```
00001001110001101010111101011000
10101111010110000000100111000110
11000110101011110101100000001001
01011000000010011100011010101111
```

Programa

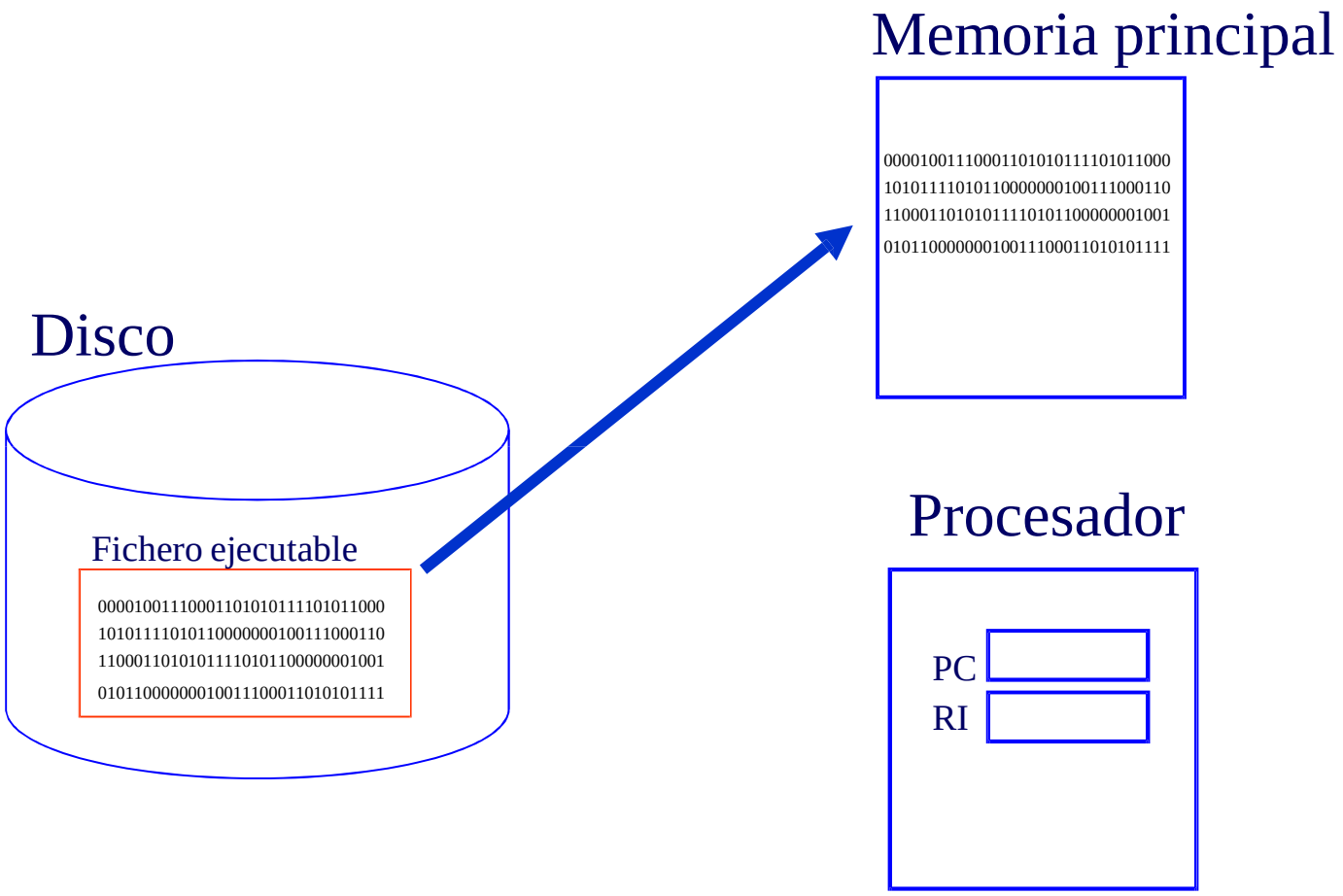
- Secuencia consecutiva de instrucciones máquina



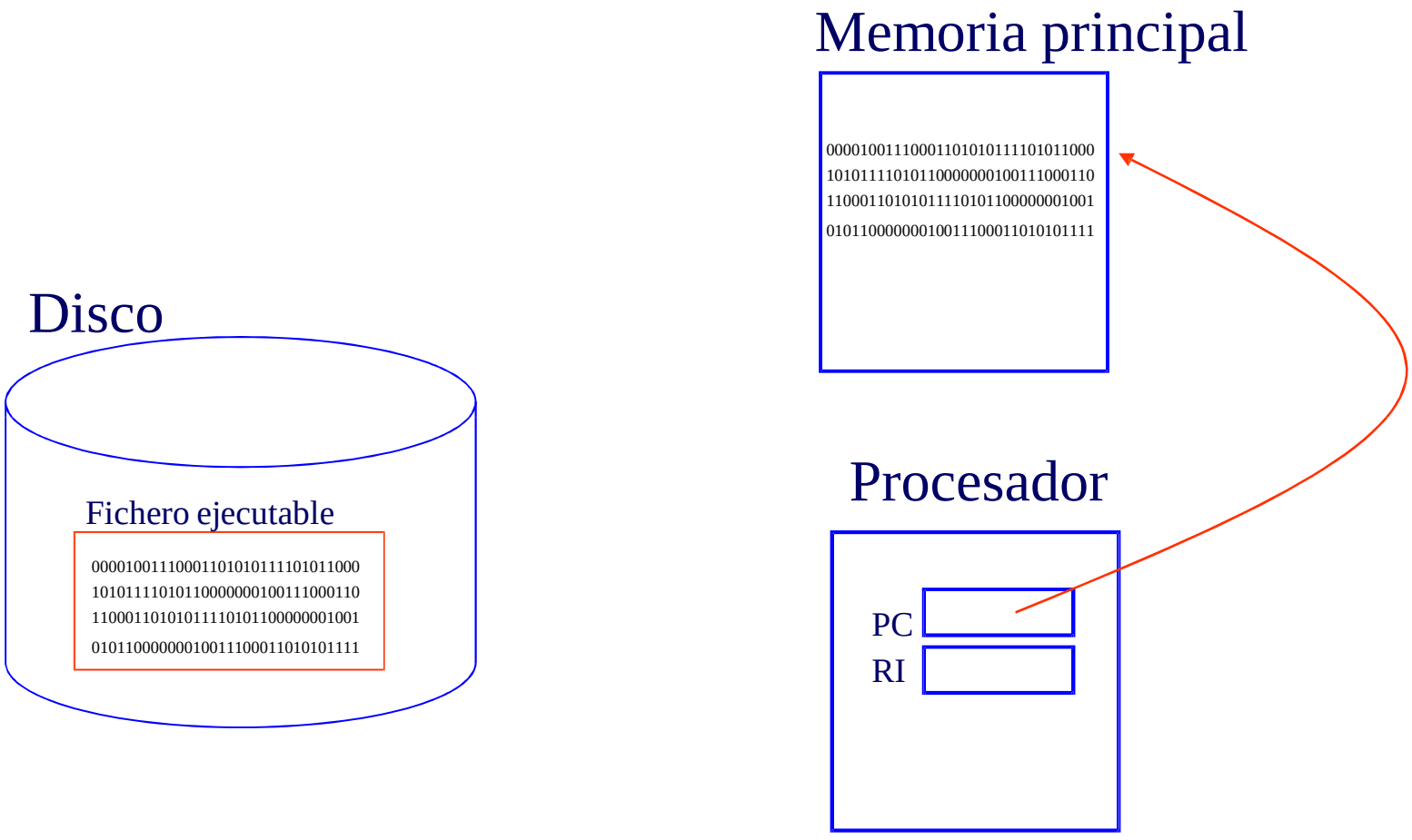
Ejecución de un programa



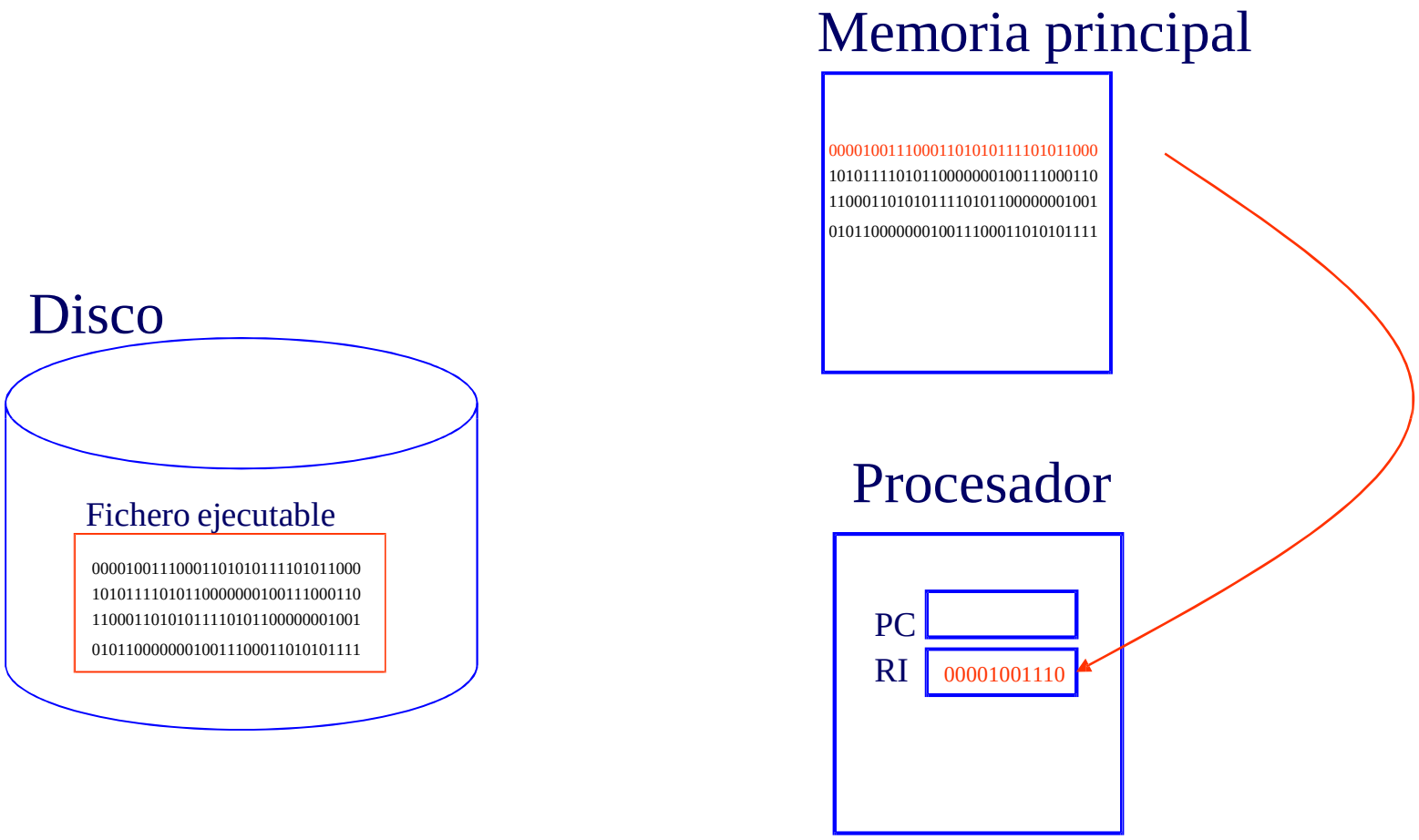
Ejecución de un programa



Ejecución de un programa



Ejecución de un programa



Ejemplo

- Conjunto de instrucciones con las siguientes características:
  - Tamaño de una posición de memoria: 16 bits
  - Tamaño de la instrucción: 16 bits
  - Código de operación: 3 bits
    - ¿Cuántas instrucciones diferentes puede tener este computador?
    - Número de registros de propósito general: 4
    - Identificadores simbólicos:
      - R0
      - R1
      - R2
      - R3
    - ¿Cuántos bits se necesitan para representar estos 4 registros?

Ejemplo

- Conjunto de instrucciones con las siguientes características:
  - Tamaño de una posición de memoria: 16 bits
  - Tamaño de la instrucción: 16 bits
  - Código de operación: 3 bits
    - ¿Cuántas instrucciones diferentes puede tener este computador? 8 instrucciones
    - Número de registros de propósito general: 4 (2 bits)
    - R0 (00)
    - R1 (01)
    - R2 (10)
    - R3 (11)

Ejemplo. Juego de instrucciones

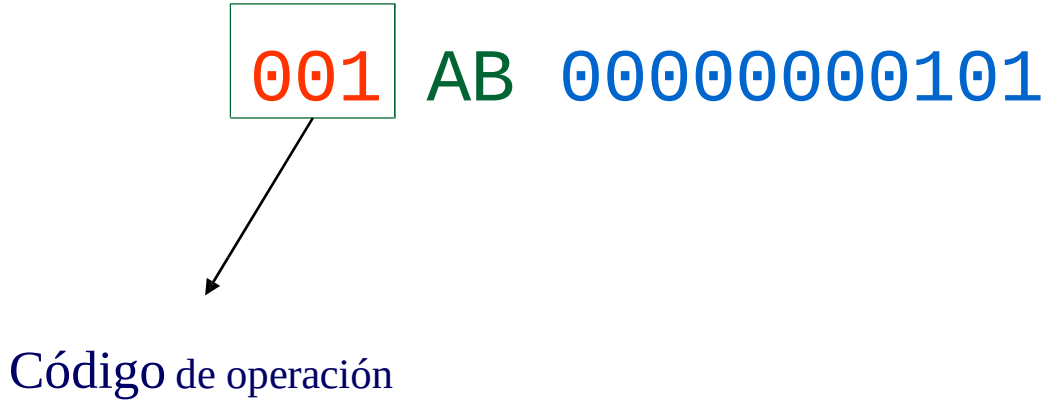
Instrucción	Descripción
000EFABCDXXXXXXX	Suma el registro AB con el CD y deja el resultado en EF
001AB00000000101	Almacena en el registro AB el valor 00000000101
010AB00000001001	Almacena en el registro AB el valor almacenado en la posición de memoria 00000001001
011AB00000001001	Almacena en la posición de memoria 00000001001 el contenido del registro AB
100000000001001	Se salta a ejecutar la instrucción almacenada en la posición de memoria 000000001001
101ABCD000001001	Si el contenido del registro AB es igual al del registro CD se salta a ejecutar la instrucción almacenada en 000001001

Siendo A,B, C, D, E, F = 0 o 1

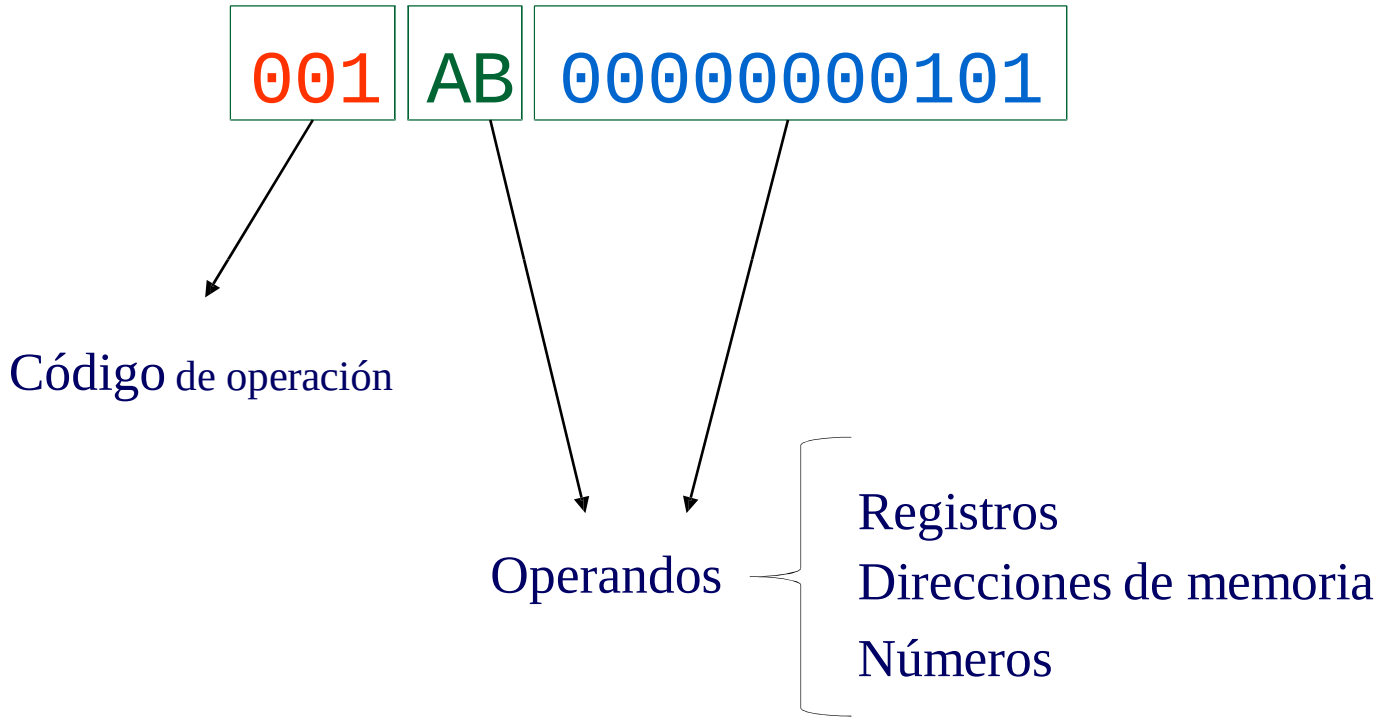
Formato de una instrucción máquina

```
001 AB 00000000101
```

Formato de una instrucción máquina



Formato de una instrucción máquina



Ejemplo. Juego de instrucciones

Instrucción	Descripción
000010010XXXXXXX	Suma el registro 00 con el 10 y deja el resultado en 01
0010100000000101	Almacena en el registro 01 el valor 0000000101
0100100000001001	Almacena en el registro 01 el valor almacenado en la posición de memoria 0000001001
0110100000001001	Almacena en la posición de memoria 0000001001 el contenido del registro 01
1000000000001001	Se salta a ejecutar la instrucción almacenada en la posición de memoria 000000001001
1010100000001001	Si el contenido del registro 01 es igual al del registro 00 se salta a ejecutar la instrucción almacenada en 000001001

Ejemplos

- Instrucción que almacena un 5 en el registro 00
- Instrucción que almacena un 7 en el registro 01
- Instrucción que suma el contenido del registro 00 y el registro 01 y deja el resultado en el registro 10
- Instrucción que almacena el resultado anterior en la posición de memoria 1027 (en decimal)

Ejemplos

Instrucción	Descripción
00001001 10XXXXXX	Suma el registro 00 con el 10 y deja el resultado en 01
00101000 00000101	Almacena en el registro 01 el valor 00000000101
01001000 00001001	Almacena en el registro 01 el valor almacenado en la posición de memoria 00000001001
01101000 00001001	Almacena en la posición de memoria 00000001001 el contenido del registro 01
10000000 00001001	Se salta a ejecutar la instrucción almacenada en la posición de memoria 0000000001001
10101001 00000001	Si el contenido del registro 01 es igual al del registro 00 se salta a ejecutar la instrucción almacenada en 000001001

- Instrucción que almacena un 5 en el registro 00
- Instrucción que almacena un 7 en el registro 01

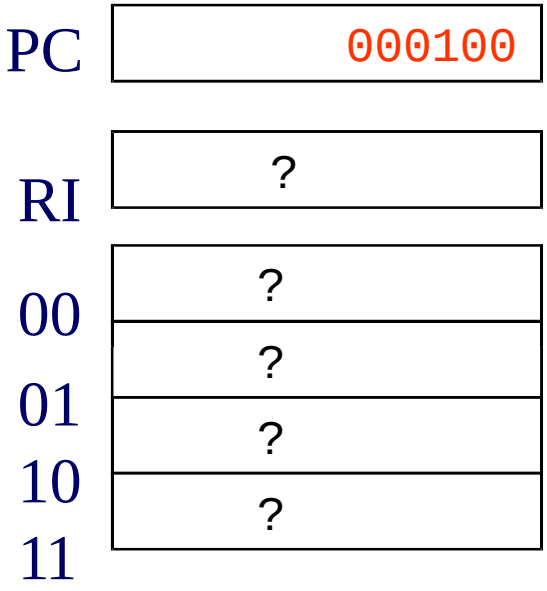
Ejemplos Solución

- Instrucción que almacena un 5 en el registro 00
  - 0010000000000101
- Instrucción que almacena un 7 en el registro 01
  - 0010100000000111
- Instrucción que suma el contenido del registro 00 y el registro 01 y deja el resultado en el registro 10
  - 0001000001XXXXXX
- Instrucción que almacena el resultado anterior en la posición de memoria 1027 (en decimal)
  - 0111010000000011

Ejemplo de programa

Dirección	Contenido
000100	0010000000000000
000101	0010100000000100
000110	0011000000000001
000111	0011100000000000
001000	1010001000001100
001001	0001111100000000
001010	0000000100000000
001011	1000000000001000
001100	0111100000100000

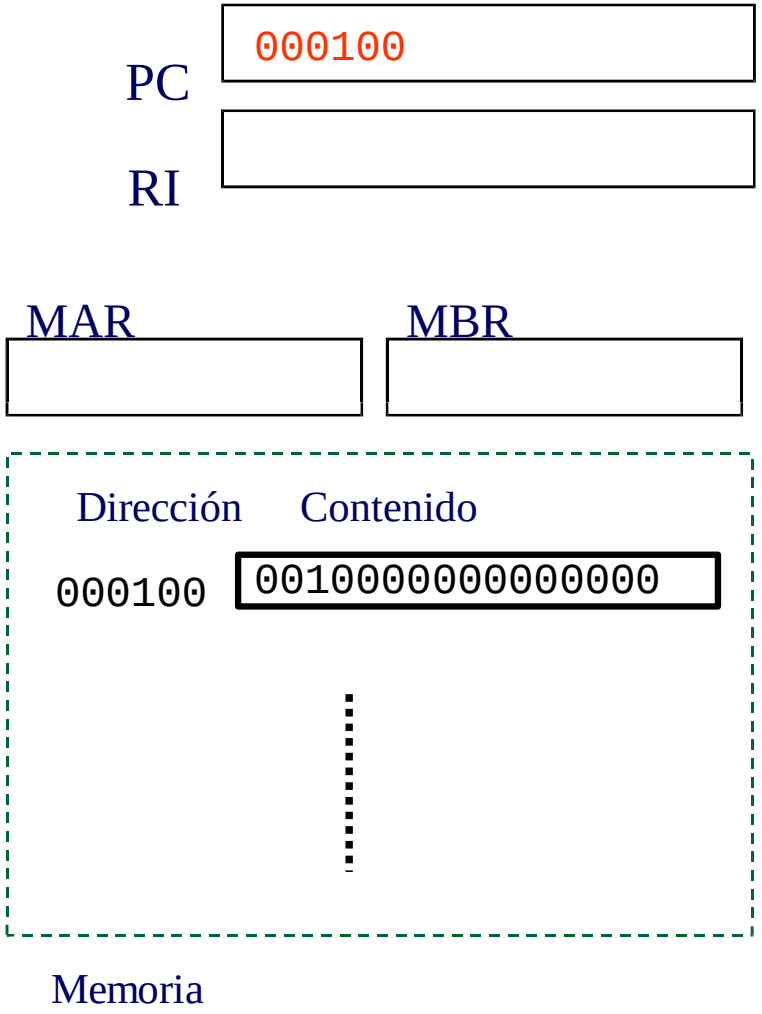
Inicio de la ejecución del programa



Dirección	Contenido
000100	0010000000000000
000101	0010100000000100
000110	0011000000000001
000111	0011100000000000
001000	1010001000001100
001001	0001111100000000
001010	0000000100000000
001011	1000000000001000
001100	0111100000100000

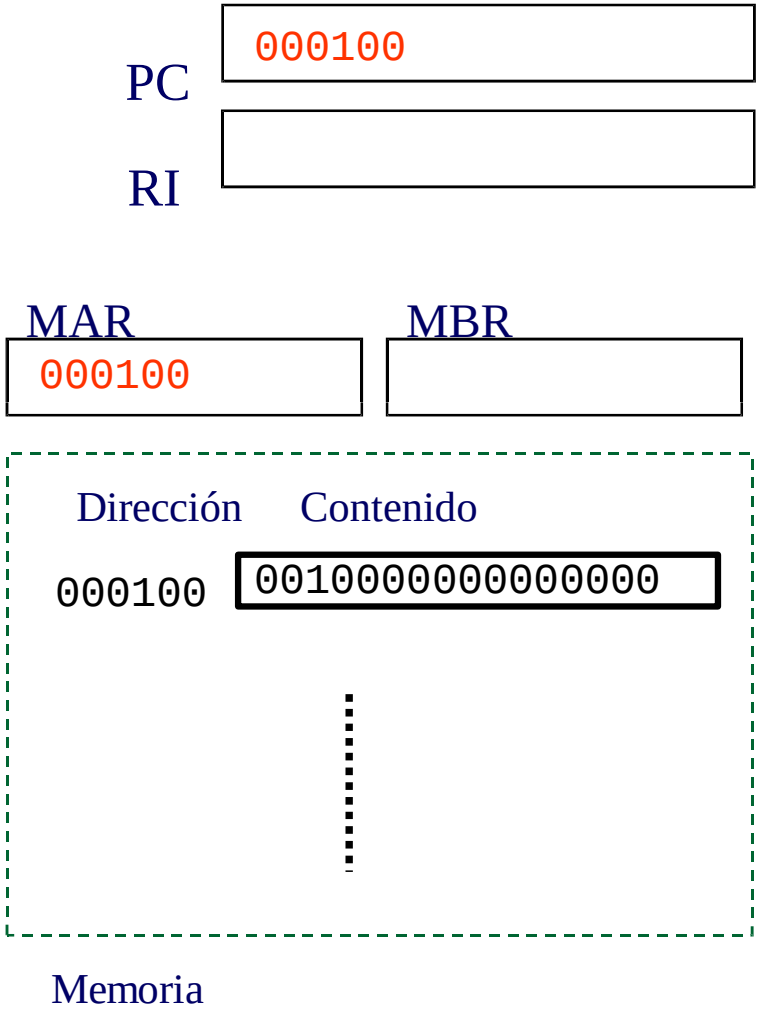
Fases de ejecución de una instrucción

- Lectura de la instrucción (ciclo de *fetch)*
  - MAR ← PC
  - Lectura
  - MBR ← Memoria
  - PC ← PC + 1
  - RI ← MBR
- Ejecución de la instrucción
- Volver a *fetch*



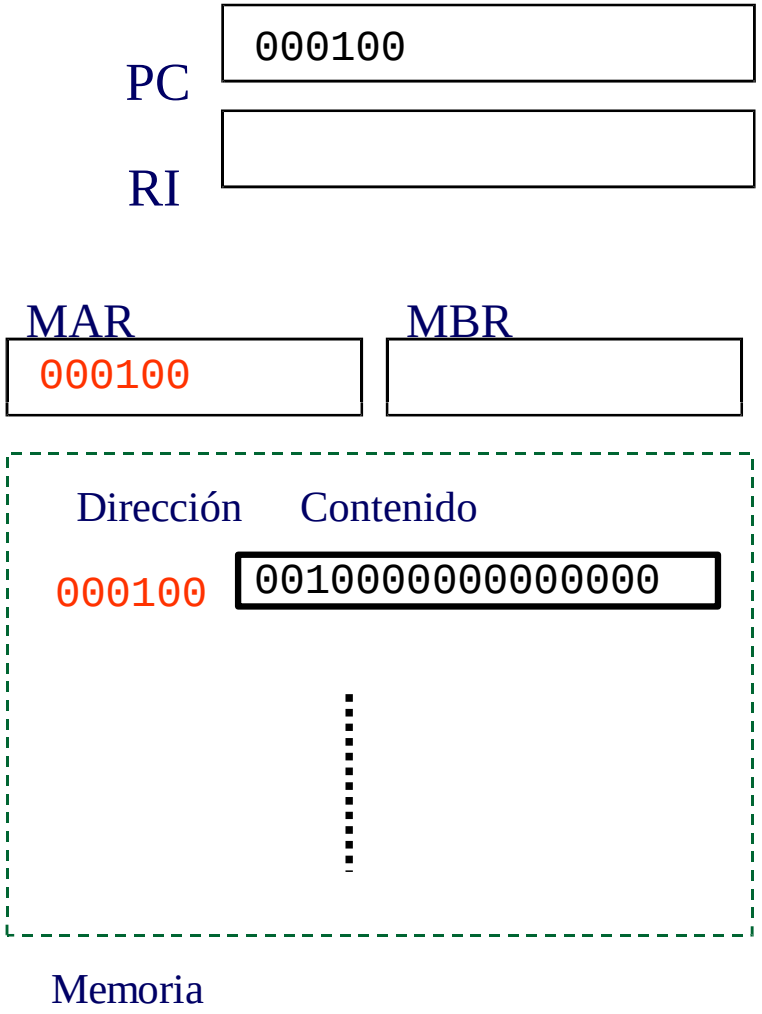
Fases de ejecución de una instrucción

- Lectura de la instrucción (ciclo de *fetch)*
  - MAR ← PC
  - Lectura
  - MBR ← Memoria
  - PC ← PC + 1
  - RI ← MBR
- Ejecución de la instrucción
- Volver a *fetch*



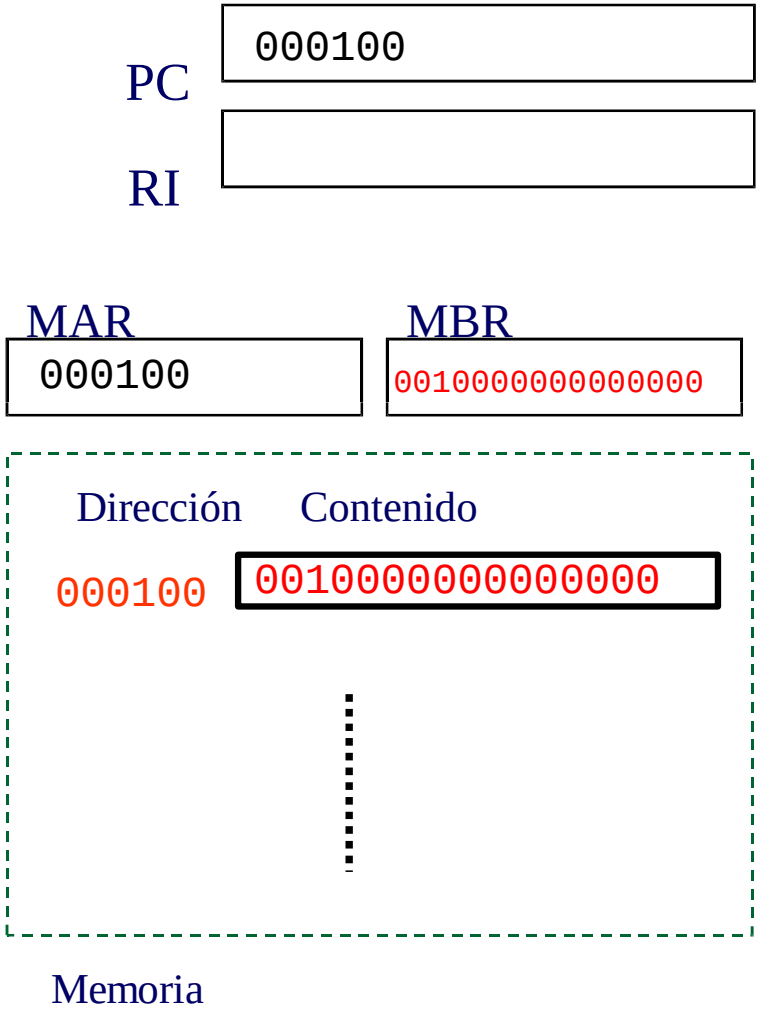
Fases de ejecución de una instrucción

- Lectura de la instrucción (ciclo de *fetch)*
  - MAR ← PC
  - Lectura
  - MBR ← Memoria
  - PC ← PC + 1
  - RI ← MBR
- Ejecución de la instrucción
- Volver a *fetch*



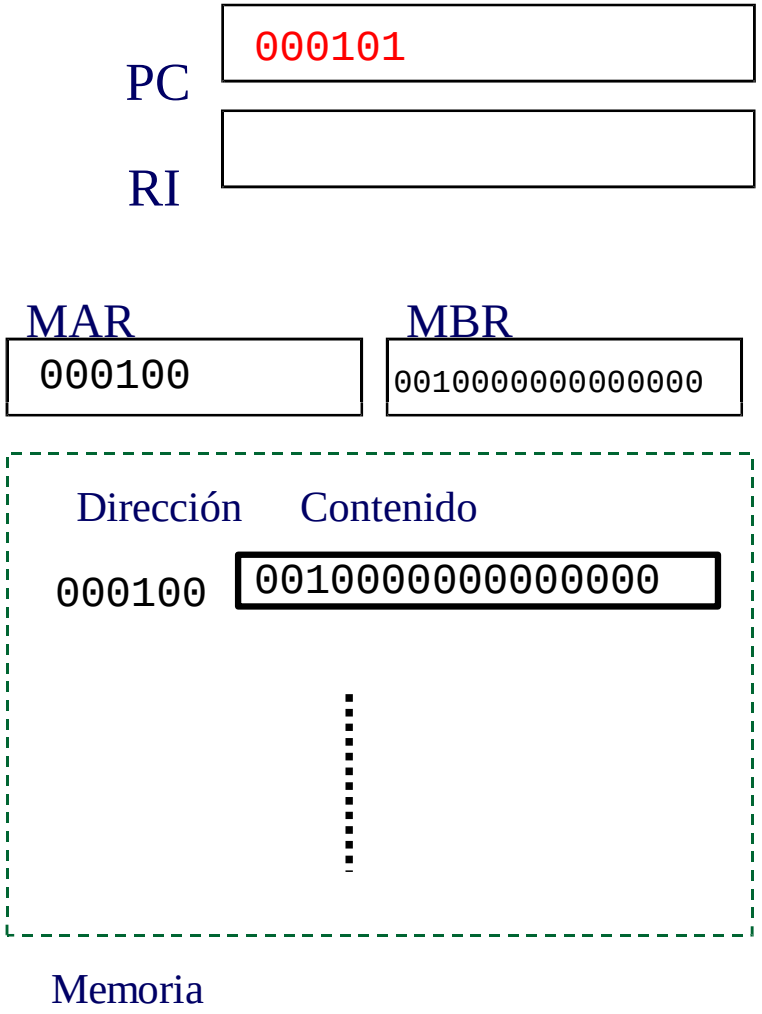
Fases de ejecución de una instrucción

- Lectura de la instrucción (ciclo de *fetch)*
  - MAR ← PC
  - Lectura
  - MBR ← Memoria
  - PC ← PC + 1
  - RI ← MBR
- Ejecución de la instrucción
- Volver a *fetch*



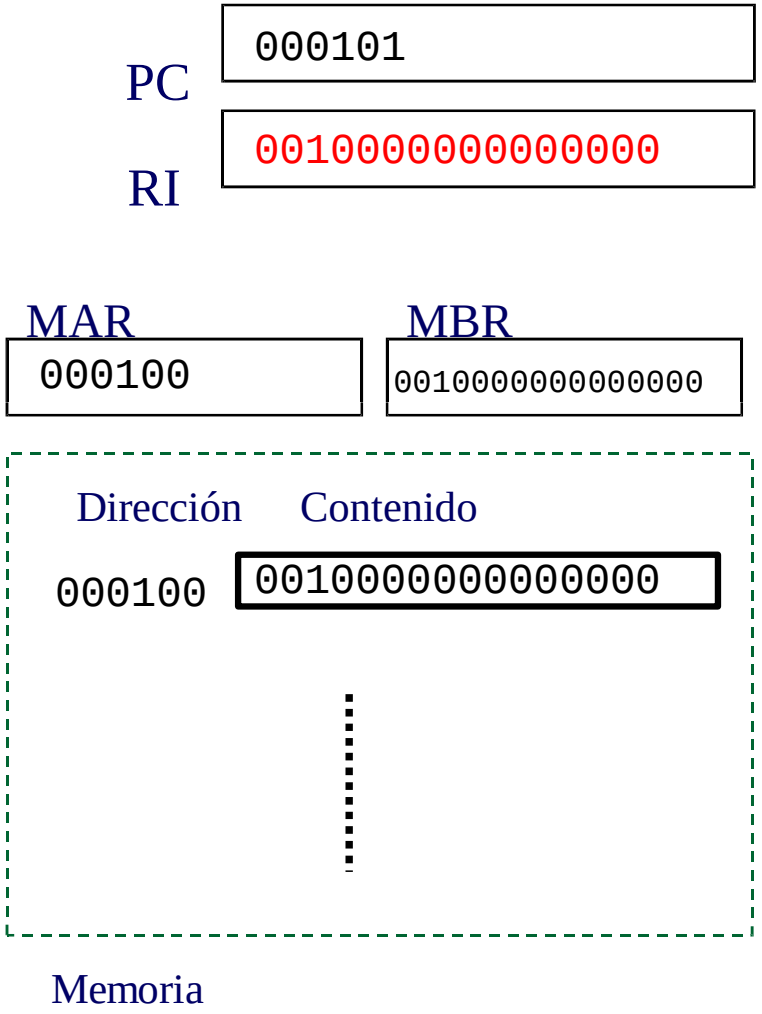
Fases de ejecución de una instrucción

- Lectura de la instrucción (ciclo de *fetch)*
  - MAR ← PC
  - Lectura
  - MBR ← Memoria
  - PC ← PC + 1
  - RI ← MBR
- Ejecución de la instrucción
- Volver a *fetch*



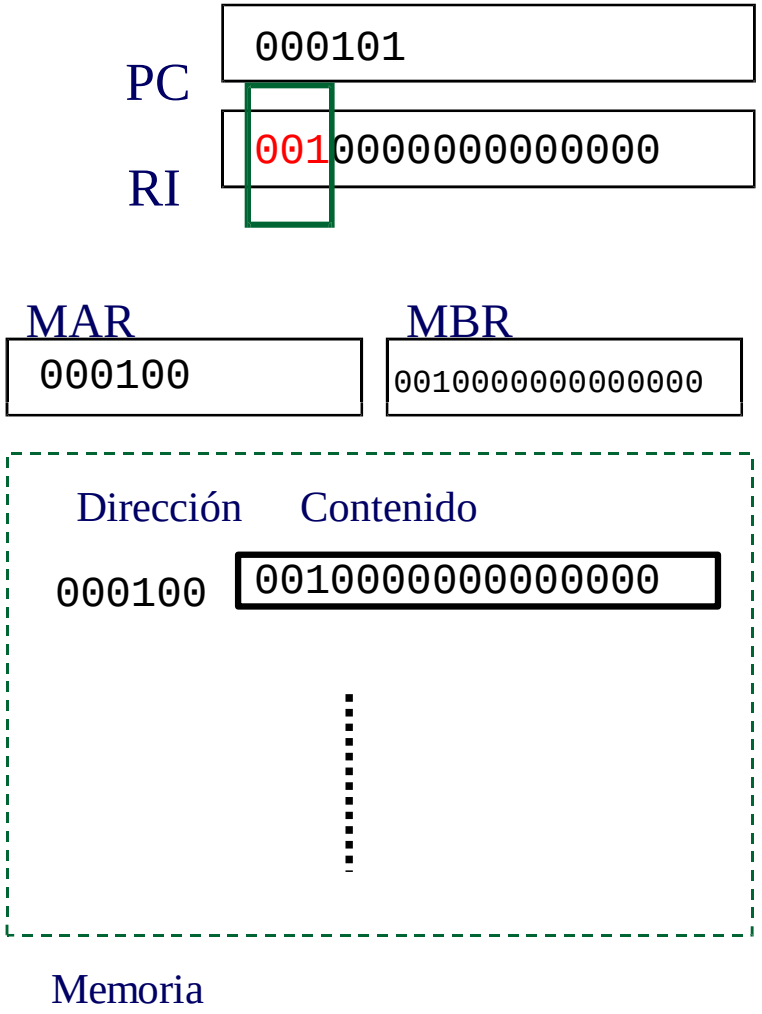
Fases de ejecución de una instrucción

- Lectura de la instrucción (ciclo de *fetch)*
  - MAR ← PC
  - Lectura
  - MBR ← Memoria
  - PC ← PC + 1
  - RI ← MBR
- Ejecución de la instrucción
- Volver a *fetch*



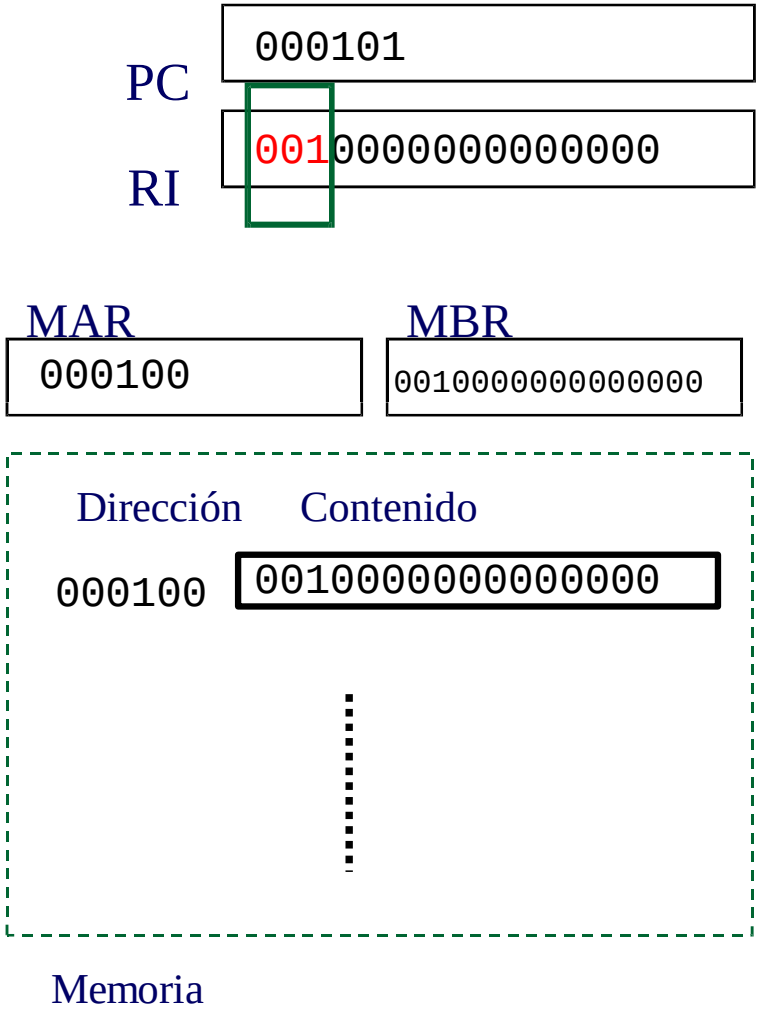
Fases de ejecución de una instrucción

- Lectura de la instrucción (ciclo de *fetch)*
  - MAR ← PC
  - Lectura
  - MBR ← Memoria
  - PC ← PC + 1
  - RI ← MBR
- Ejecución de la instrucción
- Volver a *fetch*

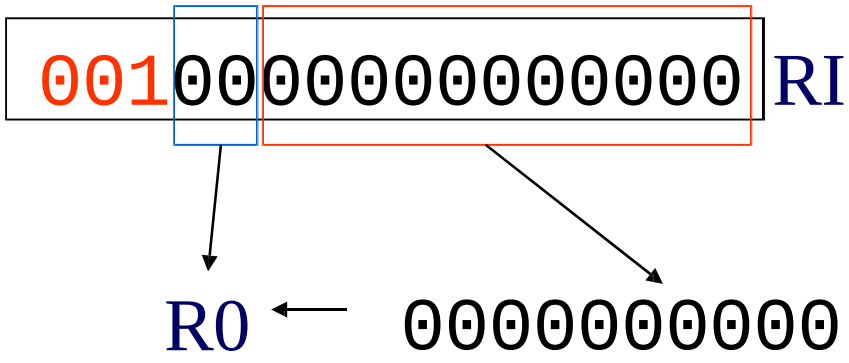


Fases de ejecución de una instrucción

- Lectura de la instrucción (ciclo de *fetch)*
  - MAR ← PC
  - Lectura
  - MBR ← Memoria
  - PC ← PC + 1
  - RI ← MBR
- Ejecución de la instrucción
- Volver a *fetch*

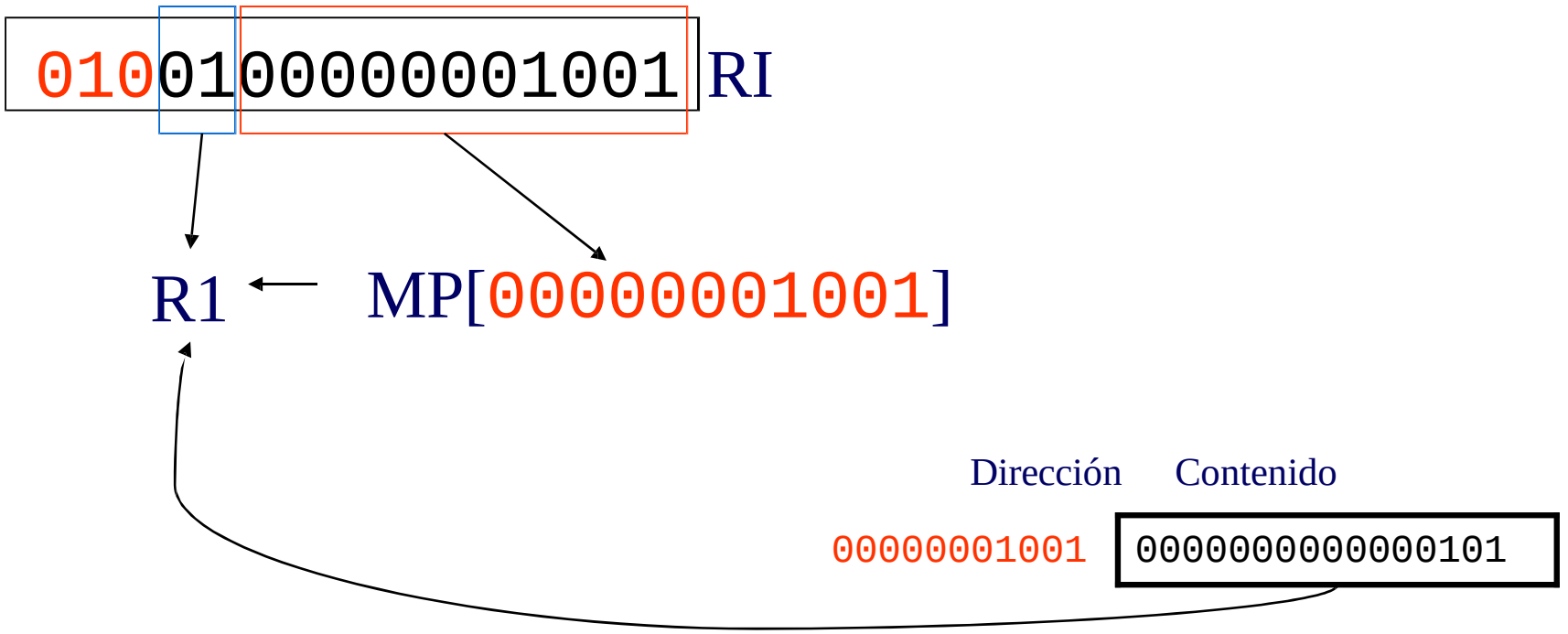


Ejecución de instrucciones



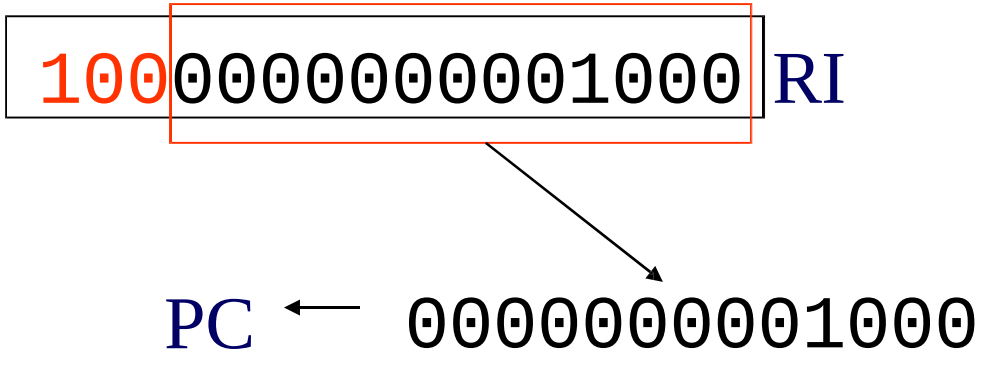
Se carga en R0 el valor 0

Ejecución de instrucciones



Se carga en R1 el contenido de la posición de memoria 00000001001

Ejecución de instrucciones

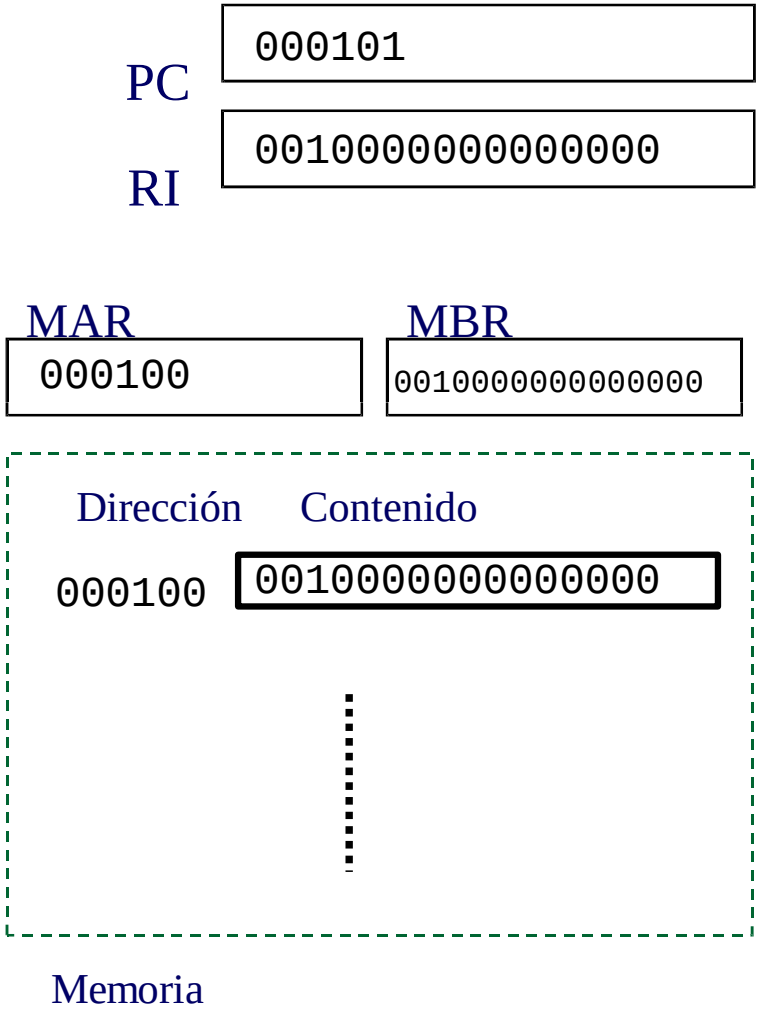


Se modifica el PC con la dirección `0000000001000`

de forma que la siguiente instrucción a ejecutar es la que se encuentra en `0000000001000`

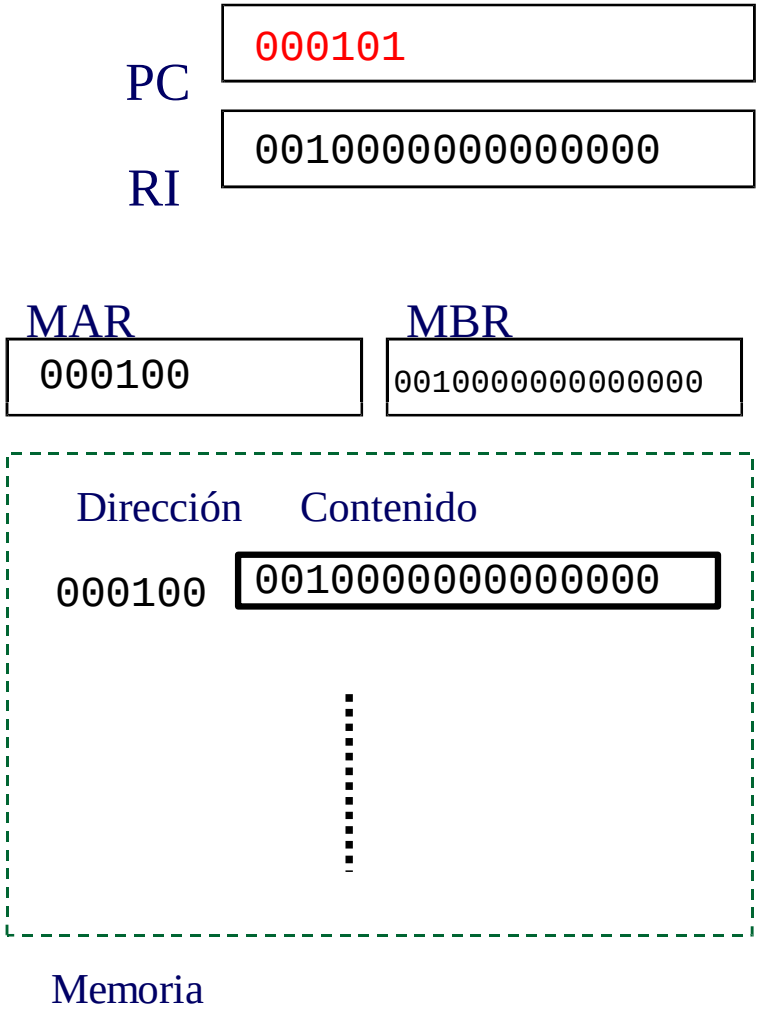
Fases de ejecución de una instrucción

- Lectura de la instrucción (ciclo de *fetch)*
  - MAR ← PC
  - Lectura
  - MBR ← Memoria
  - PC ← PC + 1
  - RI ← MBR
- Ejecución de la instrucción
- Volver a *fetch*

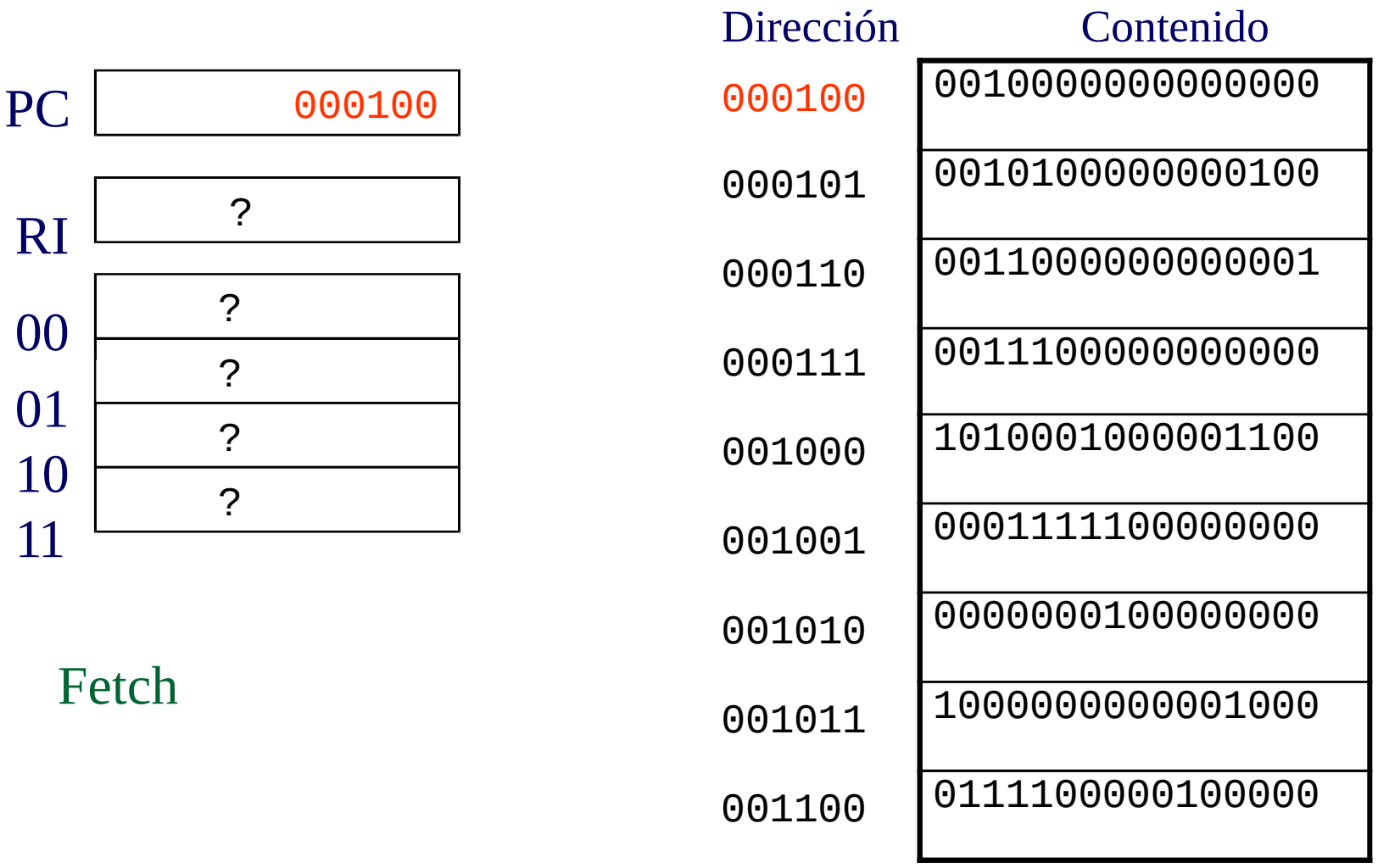


Fases de ejecución de una instrucción

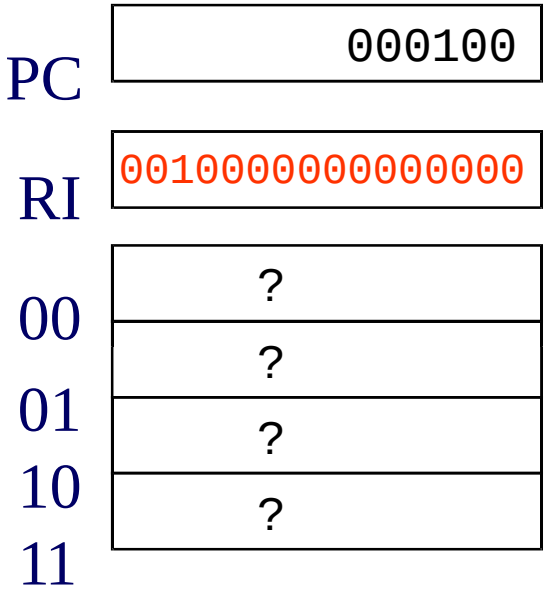
- Lectura de la instrucción (ciclo de *fetch)*
  - MAR ← PC
  - Lectura
  - MBR ← Memoria
  - PC ← PC + 1
  - RI ← MBR
- Ejecución de la instrucción
- Volver a fetch



Ejecución del programa



Ejecución del programa

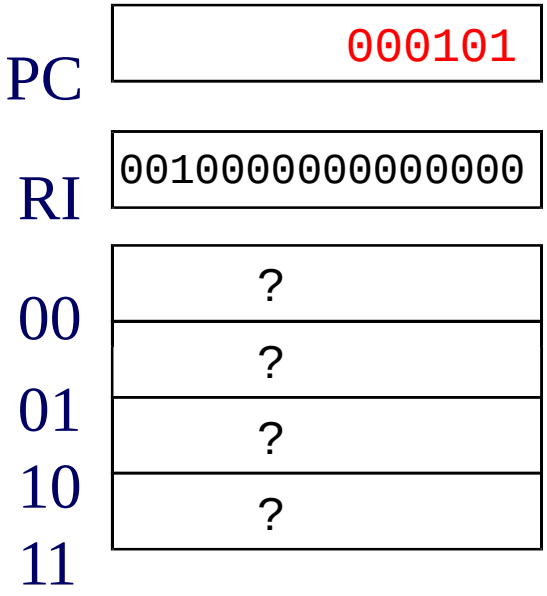


Fetch

Lectura de la Inst.

Dirección	Contenido
000100	0010000000000000
000101	0010100000000100
000110	0011000000000001
000111	0011100000000000
001000	1010001000001100
001001	0001111100000000
001010	0000000100000000
001011	1000000000001000
001100	0111100000100000

Ejecución del programa

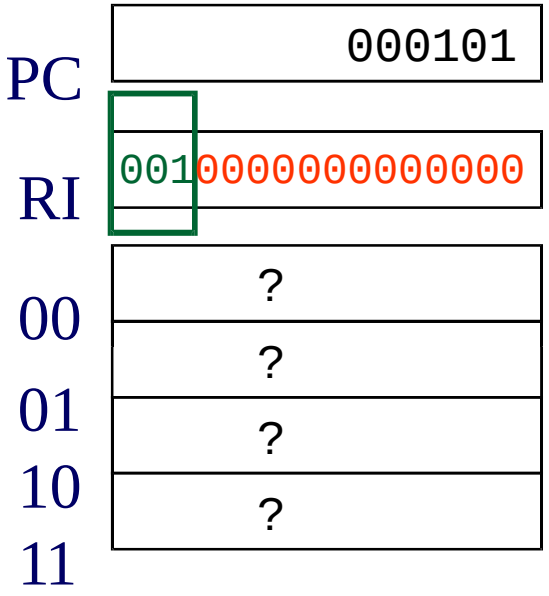


Fetch

PC ← PC + 1

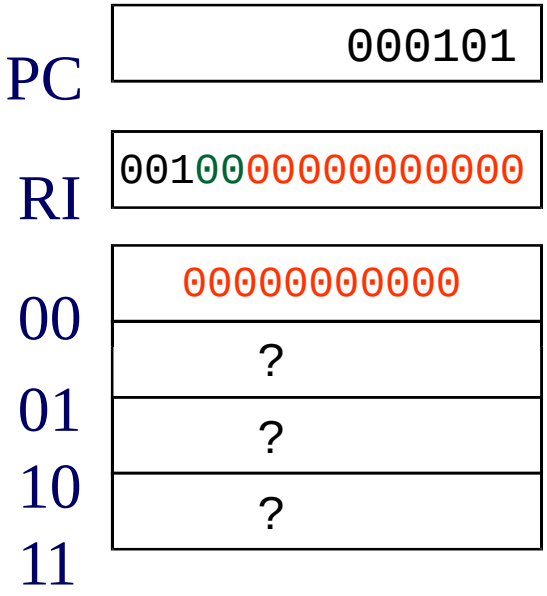
Dirección	Contenido
000100	0010000000000000
000101	0010100000000100
000110	0011000000000001
000111	0011100000000000
001000	1010001000001100
001001	0001111100000000
001010	0000000100000000
001011	1000000000001000
001100	0111100000100000

Ejecución del programa



Dirección	Contenido
000100	0010000000000000
000101	0010100000000100
000110	0011000000000001
000111	0011100000000000
001000	1010001000001100
001001	0001111100000000
001010	0000000100000000
001011	1000000000001000
001100	0111100000100000

Ejecución del programa

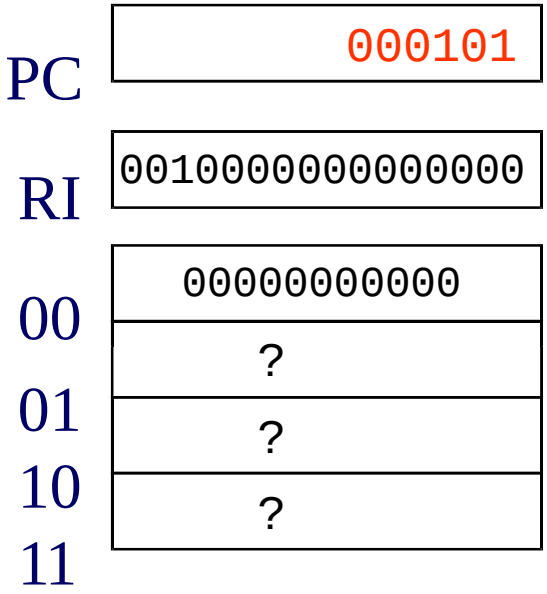


Ejecución

R0 ← 00000000000

Dirección	Contenido
000100	0010000000000000
000101	0010100000000100
000110	0011000000000001
000111	0011100000000000
001000	1010001000001100
001001	0001111100000000
001010	0000000100000000
001011	1000000000001000
001100	0111100000100000

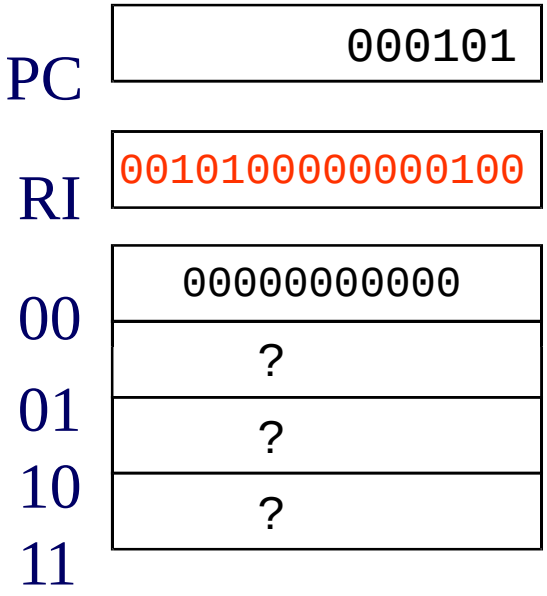
Ejecución del programa



Fetch

Dirección	Contenido
000100	0010000000000000
000101	0010100000000100
000110	0011000000000001
000111	0011100000000000
001000	1010001000001100
001001	0001111100000000
001010	0000000100000000
001011	1000000000001000
001100	0111100000100000

Ejecución del programa

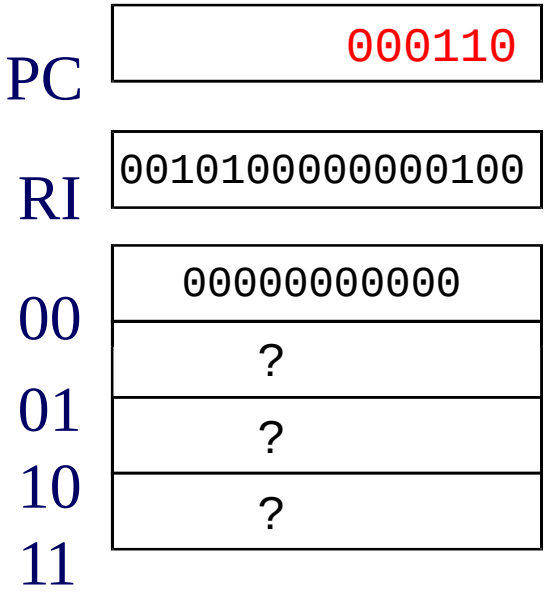


Fetch

Lectura de la Inst.

Dirección	Contenido
000100	0010000000000000
000101	0010100000000100
000110	0011000000000001
000111	0011100000000000
001000	1010001000001100
001001	0001111100000000
001010	0000000100000000
001011	1000000000001000
001100	0111100000100000

Ejecución del programa

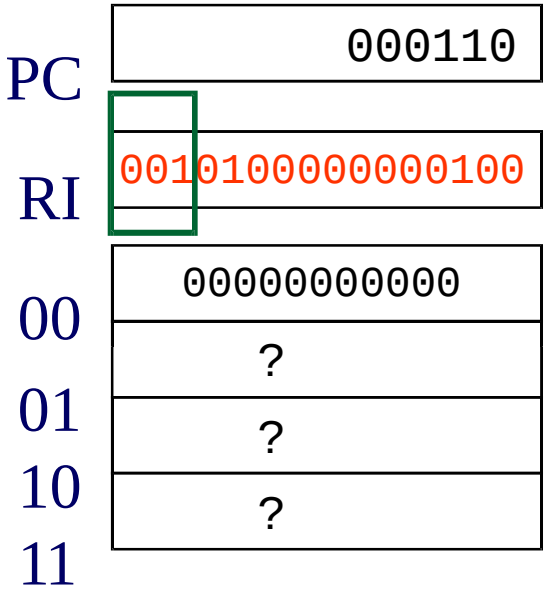


Fetch

PC ← PC +1

Dirección	Contenido
000100	0010000000000000
000101	0010100000000100
000110	0011000000000001
000111	0011100000000000
001000	1010001000001100
001001	0001111100000000
001010	0000000100000000
001011	1000000000001000
001100	0111100000100000

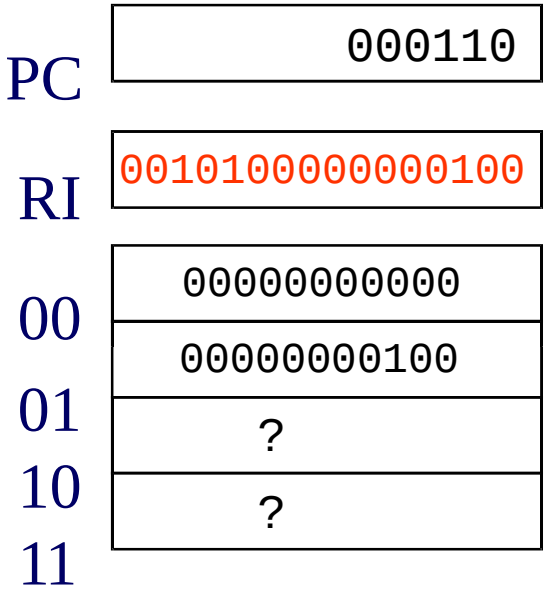
Ejecución del programa



Decodificación

Dirección	Contenido
000100	0010000000000000
000101	0010100000000100
000110	0011000000000001
000111	0011100000000000
001000	1010001000001100
001001	0001111100000000
001010	0000000100000000
001011	1000000000001000
001100	0111100000100000

Ejecución del programa



Dirección	Contenido
000100	0010000000000000
000101	0010100000000100
000110	0011000000000001
000111	0011100000000000
001000	1010001000001100
001001	0001111100000000
001010	0000000100000000
001011	1000000000001000
001100	0111100000100000

Ejecución

R1 ← 00000000100

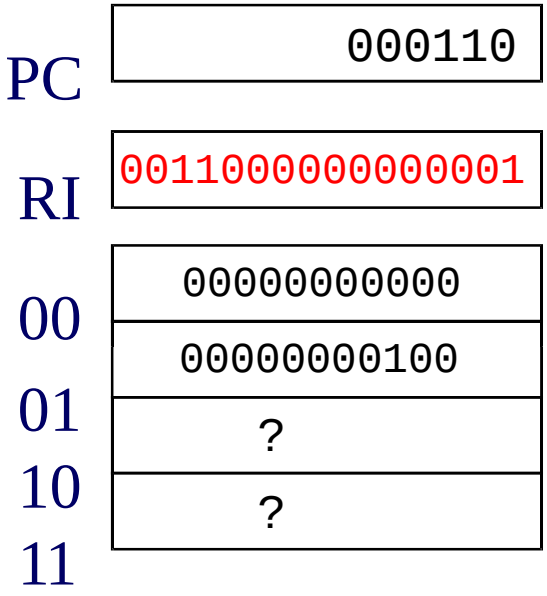
Ejecución del programa

PC	000110
RI	0010100000000100
00	00000000000
	00000000100
01	?
10	?
11	

Fetch

Dirección	Contenido
000100	0010000000000000
000101	0010100000000100
000110	0011000000000001
000111	0011100000000000
001000	1010001000001100
001001	0001111100000000
001010	0000000100000000
001011	1000000000001000
001100	0111100000100000

Ejecución del programa

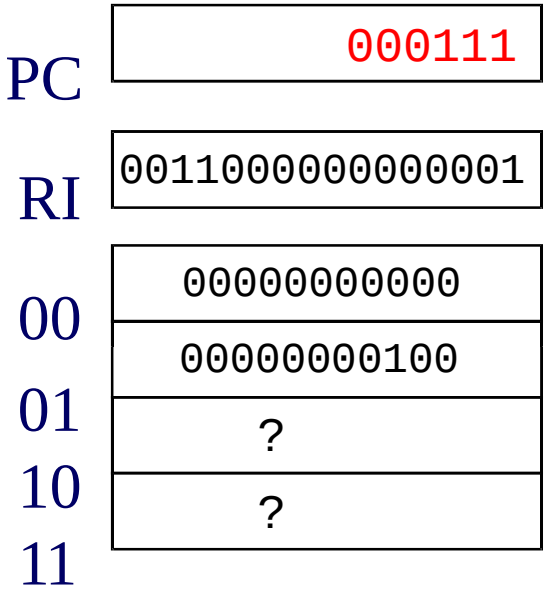


Fetch

Lectura de la Inst.

Dirección	Contenido
000100	0010000000000000
000101	0010100000000100
000110	0011000000000001
000111	0011100000000000
001000	1010001000001100
001001	0001111100000000
001010	0000000100000000
001011	1000000000001000
001100	0111100000100000

Ejecución del programa

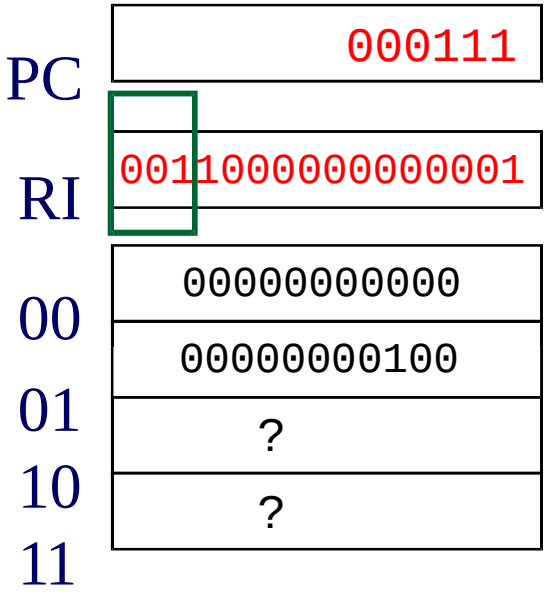


Fetch

PC ← PC +1

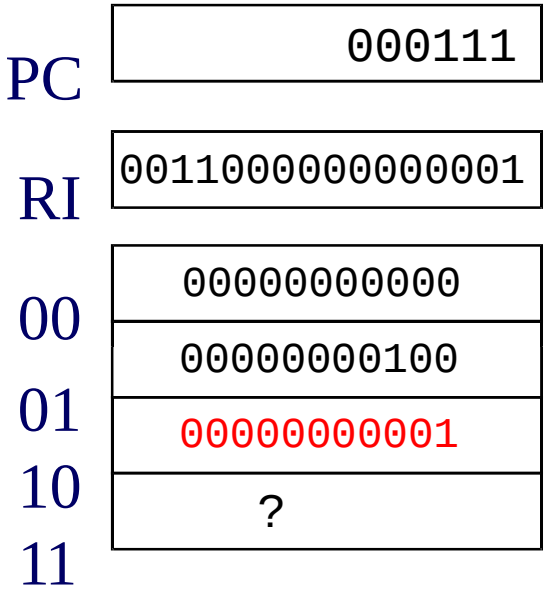
Dirección	Contenido
000100	0010000000000000
000101	0010100000000100
000110	0011000000000001
000111	0011100000000000
001000	1010001000001100
001001	0001111100000000
001010	0000000100000000
001011	1000000000001000
001100	0111100000100000

Ejecución del programa



Dirección	Contenido
000100	0010000000000000
000101	0010100000000100
000110	0011000000000001
000111	0011100000000000
001000	1010001000001100
001001	0001111100000000
001010	0000000100000000
001011	1000000000001000
001100	0111100000100000

Ejecución del programa



Dirección	Contenido
000100	0010000000000000
000101	0010100000000100
000110	0011000000000001
000111	0011100000000000
001000	1010001000001100
001001	0001111100000000
001010	0000000100000000
001011	1000000000001000
001100	0111100000100000

Ejecución

R2 ← 00000000001

Ejecución del programa

PC	000111
RI	0011000000000001
00	00000000000
01	00000000100
10	00000000001
11	?

Fetch

Dirección	Contenido
000100	0010000000000000
000101	0010100000000100
000110	0011000000000001
000111	0011100000000000
001000	1010001000001100
001001	0001111100000000
001010	0000000100000000
001011	1000000000001000
001100	0111100000100000

Ejecución del programa

PC	000111
RI	0011100000000000

00	00000000000
01	00000000100
10	00000000001
11	?

Fetch

Lectura de la Inst.

Dirección	Contenido
000100	0010000000000000
000101	0010100000000100
000110	0011000000000001
000111	0011100000000000
001000	1010001000001100
001001	0001111100000000
001010	0000000100000000
001011	1000000000001000
001100	0111100000100000

Ejecución del programa

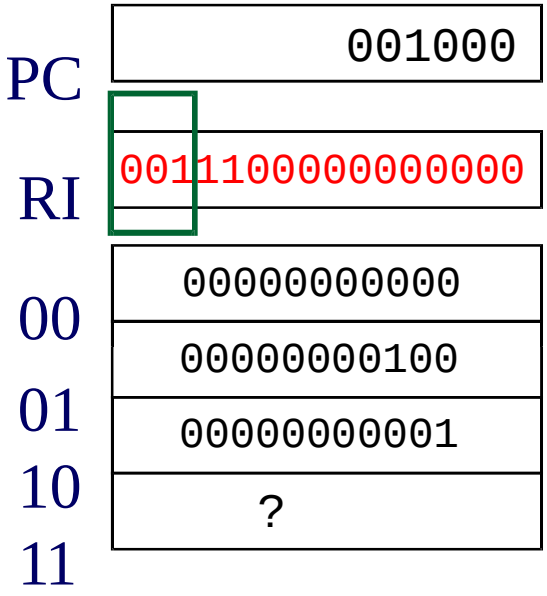
PC	001000
RI	0011100000000000
00	00000000000
01	00000000100
10	00000000001
11	?

Fetch

PC ← PC +1

Dirección	Contenido
000100	0010000000000000
000101	0010100000000100
000110	0011000000000001
000111	0011100000000000
001000	1010001000001100
001001	0001111100000000
001010	0000000100000000
001011	1000000000001000
001100	0111100000100000

Ejecución del programa



Dirección	Contenido
000100	0010000000000000
000101	0010100000000100
000110	0011000000000001
000111	0011100000000000
001000	1010001000001100
001001	0001111100000000
001010	0000000100000000
001011	1000000000001000
001100	0111100000100000

Ejecución del programa

PC	001000
RI	0011100000000000

00	00000000000
01	00000000100
10	00000000001
11	00000000000

Dirección	Contenido
000100	0010000000000000
000101	0010100000000100
000110	0011000000000001
000111	0011100000000000
001000	1010001000001100
001001	0001111100000000
001010	0000000100000000
001011	1000000000001000
001100	0111100000100000

Ejecución

R3 ← 00000000000

Ejecución del programa

PC	001000
RI	0011100000000000
00	00000000000
01	00000000100
10	00000000001
11	00000000000

Fetch

Dirección	Contenido
000100	0010000000000000
000101	0010100000000100
000110	0011000000000001
000111	0011100000000000
001000	1010001000001100
001001	0001111100000000
001010	0000000100000000
001011	1000000000001000
001100	0111100000100000

Ejecución del programa

PC	001000
RI	1010001000001100

00	00000000000
01	00000000100
10	00000000001
11	00000000000

Dirección	Contenido
000100	0010000000000000
000101	0010100000000100
000110	0011000000000001
000111	0011100000000000
001000	1010001000001100
001001	0001111100000000
001010	0000000100000000
001011	1000000000001000
001100	0111100000100000

Fetch

Lectura de la Inst.

Ejecución del programa

PC	001001
RI	1010001000001100
00	00000000000
01	00000000100
10	00000000001
11	00000000000

Fetch

PC ← PC +1

Dirección	Contenido
000100	0010000000000000
000101	0010100000000100
000110	0011000000000001
000111	0011100000000000
001000	1010001000001100
001001	0001111100000000
001010	0000000100000000
001011	1000000000001000
001100	0111100000100000

Algoritmo del programa anterior

```
i=0;
s = 0;
while (i < 4)
{
  s = s + 1;
  i = i + 1;
}
```

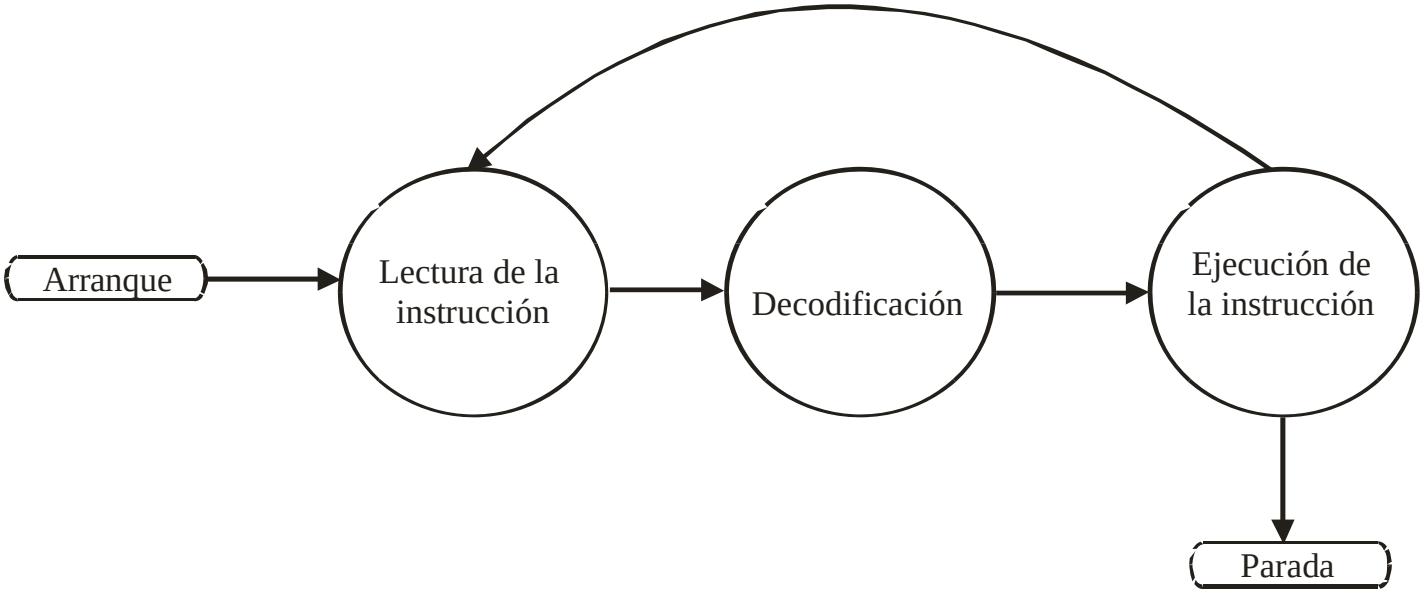
El programa almacena en la posición de memoria `00000100000`

el valor: 1 + 1 + 1 + 1

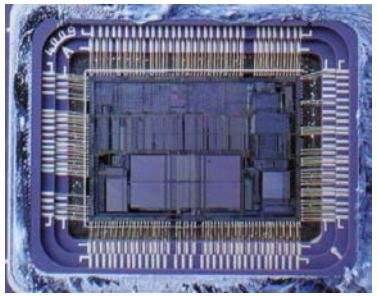
Lenguaje ensamblador

```
        li R0, 0
        li R1, 4
        li R2, 1
        li R3, 0
bucle:  beq R0, R1, fin
        add R3, R3, R2
        add R0, R0, R2
        b bucle
fin:    sw R3, 100000
```

Secuencia de ejecución de instrucciones



Ley de Moore



- Doblar la densidad implica reducir las dimensiones de sus elementos en un 30%
- En 1971 el Intel 4004 tenía 2300 transistores con tamaños de 10 micrometros
- Hoy en día se consiguen chips con distancias de 45 nanometros
- Para cumplir la ley de Moore se necesita tecnología cuyo precio se dobla cada 4.4 años

Parámetros característicos de un computador

- Ancho de palabra
- Memoria principal o Memoria RAM
- Memoria auxiliar
- Ancho de banda
- MIPS
- MFLOPS

Prefijos

Nombre	Abr	Factor	SI
Kilo	K	2^{10} = 1,024	10^{3} = 1,000
Mega	M	2^{20} = 1,048,576	10^{6} = 1,000,000
Giga	G	2^{30} = 1,073,741,824	10^{9} = 1,000,000,000
Tera	T	2^{40} = 1,099,511,627,776	10^{12} = 1,000,000,000,000
Peta	P	2^{50} = 1,125,899,906,842,624	10^{15} = 1,000,000,000,000,000
Exa	E	2^{60} = 1,152,921,504,606,846,976	10^{18} = 1,000,000,000,000,000,000
Zetta	Z	2^{70} = 1,180,591,620,717,411,303,424	10^{21} = 1,000,000,000,000,000,000,000
Yotta	Y	2^{80} = 1,208,925,819,614,629,174,706,176	10^{24} = 1,000,000,000,000,000,000,000,000

Ejercicio

- ¿Cuántos bytes almacena un disco duro de 250 GB ?

Prefijos

Nombre	Abr	Factor	SI
Kilo	K	2^{10} = 1,024	10^{3} = 1,000
Mega	M	2^{20} = 1,048,576	10^{6} = 1,000,000
Giga	G	2^{30} = 1,073,741,824	10^{9} = 1,000,000,000
Tera	T	2^{40} = 1,099,511,627,776	10^{12} = 1,000,000,000,000
Peta	P	2^{50} = 1,125,899,906,842,624	10^{15} = 1,000,000,000,000,000
Exa	E	2^{60} = 1,152,921,504,606,846,976	10^{18} = 1,000,000,000,000,000,000
Zetta	Z	2^{70} = 1,180,591,620,717,411,303,424	10^{21} = 1,000,000,000,000,000,000,000
Yotta	Y	2^{80} = 1,208,925,819,614,629,174,706,176	10^{24} = 1,000,000,000,000,000,000,000,000

- 1 KB = 1024 bytes, pero en el SI es 1000 bytes
- Los fabricantes de disco duros y en telecomunicaciones emplea el SI.
  - Un disco duro de 30 GB almacena 30 x 10^{9} bytes
  - Una red de 1 Mbit/s transfiere 10^{6} bps.