

## Tema 4: La CPU (*Unidad Central de Proceso*)

4.1- Introducción: conceptos básicos y estructura interna.

4.2- La CPU: estudio a nivel de Máquina Convencional.

- *Introducción.*
- *Juego de instrucciones.*
- *Ejemplos.*
- *Concepto de Compilación.*

4.3- La CPU: estudio a nivel de Micromáquina.

- *Introducción.*
- *Pasos de ejecución.*

4.4- La Unidad de Control (UC)

- *UC cableada*
- *UC microprogramada*



### 4.1- Introducción

- **Objetivo de la CPU** → Ejecutar instrucciones de algoritmos que solucionen problemas.
- **Instrucción:**
  - Operación que se realiza sobre uno o varios datos.
  - Un conjunto de instrucciones constituirá un algoritmo.
  - Una instrucción será representada por una secuencia de bits que la CPU reconocerá.
  - Esa secuencia de bits se denomina *código de la instrucción*.
  - Una secuencia de instrucciones (algoritmo) en binario se denomina *código máquina* del algoritmo.
  - Las instrucciones están almacenadas en el Sistema de Memoria.
  - Existen varios tipos de instrucciones: de transferencia de información, aritmético-lógicas y de control del flujo de un programa.



**Elementos internos de la CPU:**

- **CAMINO DE DATOS (Datapath):** es la parte ejecutiva de la CPU. Sobre él se ejecutan las instrucciones. Consta básicamente de 3 elementos:
  - Registros: almacenan datos.
  - Unidades de procesamiento: realizan el procesamiento de los datos.
    - ALU (Aritmetic Logic Unit): realiza cálculos con datos enteros. Presente en todas las CPUs
    - FPU (Floating Point Unit): realiza cálculos con datos en coma flotante. Presente sólo en algunas CPUs
  - Buses internos: interconectan unidades de procesamiento y registros.

Los elementos del Datapath se gobiernan mediante señales de control.
- **UNIDAD DE CONTROL (UC):** genera todas las señales de control necesarias y una secuenciación adecuada de las mismas para que las instrucciones puedan ejecutarse sobre el camino de datos.

**Parámetros de una CPU:**

- **Ancho de la CPU:** Número de bits de los operandos que maneja.
- **Número de Palabras de Memoria direccionables (m):** nº de palabras que se pueden direccionar (es decir, a las que se puede acceder) mediante el nº de líneas disponibles en el Bus de Direcciones.
 
$$m = 2^a, \text{ siendo 'a' el nº de líneas de direcciones.}$$
- **Tamaño de las palabras de Memoria a las que se accede (n):** En el caso más simple, coincide con el nº de líneas de datos, 'd'.



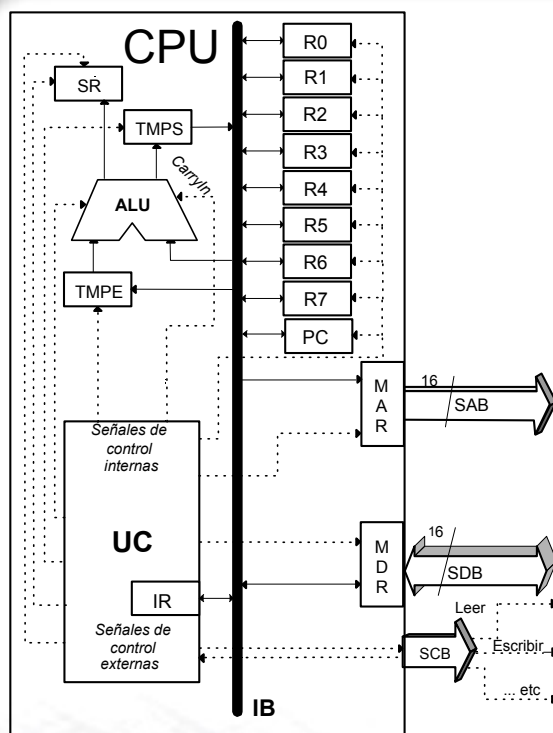
Para ejemplificar los conceptos básicos se usará una CPU de ejemplo a la que llamaremos "la CPU teórica".

### Parámetros de la CPU teórica:

- **Ancho de la CPU:** 16 bits (registros, ALU, etc.).
- **Número de Palabras de Memoria direccionables (m):**

Líneas bús direcciones=16  $\rightarrow m = 2^{16} = 65536 = 64K$  palabras

- **Tamaño de las palabras de Memoria a las que se accede (n):** 16 bits.  
16 bits  $\rightarrow$  líneas bús datos= 16



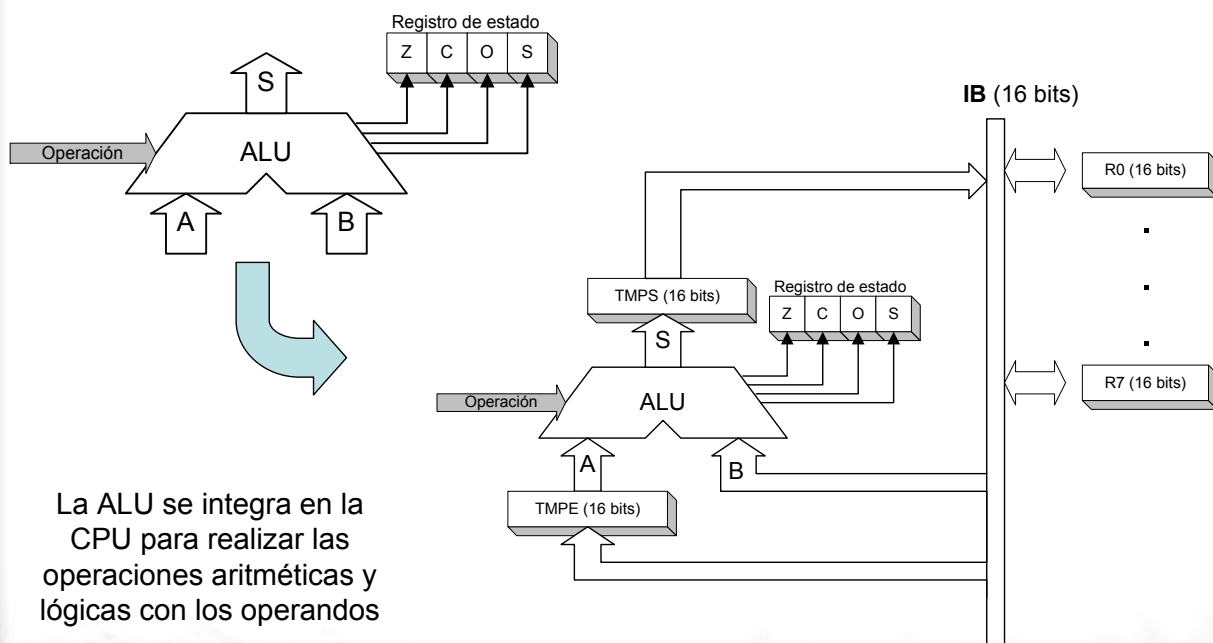
### Buses:

- **IB (Internal Bus):** Bus Interno. Interconecta los elementos internos que componen la CPU.
- **SAB (System Address Bus):** Bus de direcciones del sistema. Por él circulan las direcciones de memoria o de E/S en las que se recogerá o procesará un dato. Su ancho determina la cantidad de memoria a direccionar.
- **SDB (System Data Bus):** Bus de datos del sistema. Por él circulan los datos y las instrucciones.
- **SCB (System Control Bus):** Bus de control del sistema. Por él circulan las señales de control que gestionan el intercambio de información entre las distintas unidades (CPU, Memoria y E/S).

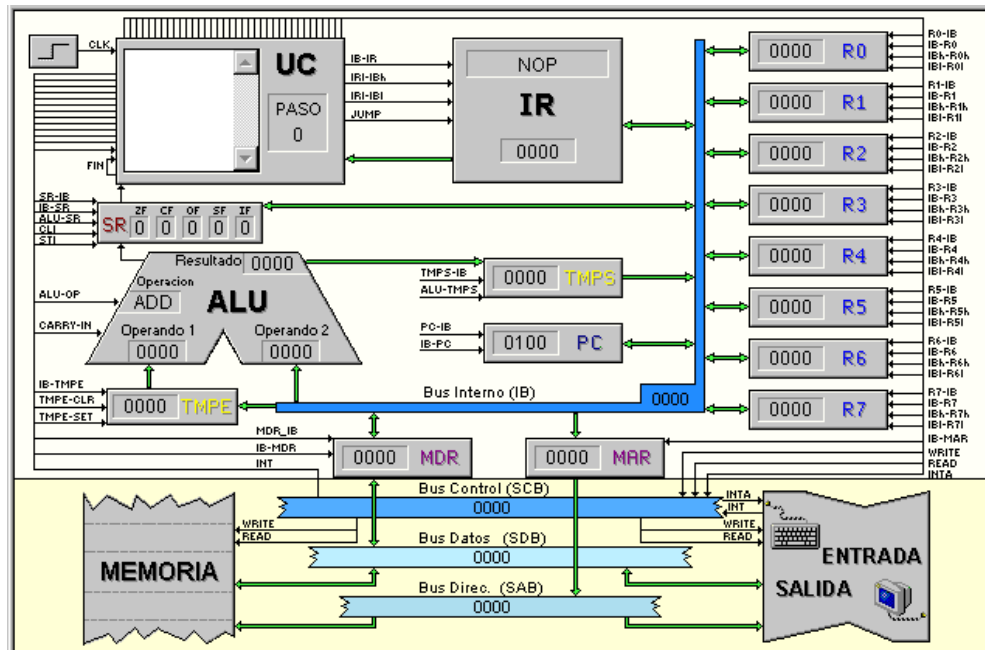


**Registros:**

- **R0, R1, R2, R3, R4, R5, R6, R7:** Registros de propósito general (p.e. almacenar operandos).
- **PC** (Program Counter): contador de programa. Contiene la dirección de memoria en la que está almacenada la siguiente instrucción a ejecutar.
- **MDR** (Memory Data Register): Registro de datos de memoria (para comunicación con el SDB).
- **MAR** (Memory Address Register): Registro de direcciones de memoria (para comunicación con el SAB).
- **IR** (Instruction Register): Registro de instrucciones. Almacena el código máquina de la instrucción a ejecutar.
- **SR** (Status Register): Registro de estado. Almacena los valores de los bits de estado asociados a la última operación realizada por la ALU.
- **TMPE:** Registro temporal de Entrada (a la ALU).
- **TMPS:** Registros temporal de Salida (de la ALU).

**Integración de la ALU en la CPU teórica.**

## La CPU teórica en el simulador (1): estructura interna.



## La CPU teórica en el simulador (2): señales de control.

<b>R0</b> <input type="checkbox"/> R0-IB <input type="checkbox"/> IB-R0 <input type="checkbox"/> IBh-R0h <input type="checkbox"/> IBi-R0i	<b>R1</b> <input type="checkbox"/> R1-IB <input type="checkbox"/> IB-R1 <input type="checkbox"/> IBh-R1h <input type="checkbox"/> IBi-R1i	<b>R2</b> <input type="checkbox"/> R2-IB <input type="checkbox"/> IB-R2 <input type="checkbox"/> IBh-R2h <input type="checkbox"/> IBi-R2i	<b>R3</b> <input type="checkbox"/> R3-IB <input type="checkbox"/> IB-R3 <input type="checkbox"/> IBh-R3h <input type="checkbox"/> IBi-R3i	<b>SR</b> <input type="checkbox"/> IB-SR <input type="checkbox"/> SR_IB <input type="checkbox"/> ALU_SR <input type="checkbox"/> CLI <input type="checkbox"/> STI	<b>IR</b> <input type="checkbox"/> IB-IR <input type="checkbox"/> IRH-IBh <input type="checkbox"/> IRH-IBi <input type="checkbox"/> JUMP
<b>R4</b> <input type="checkbox"/> R4-IB <input type="checkbox"/> IB-R4 <input type="checkbox"/> IBh-R4h <input type="checkbox"/> IBi-R4i	<b>R5</b> <input type="checkbox"/> R5-IB <input type="checkbox"/> IB-R5 <input type="checkbox"/> IBh-R5h <input type="checkbox"/> IBi-R5i	<b>R6</b> <input type="checkbox"/> R6-IB <input type="checkbox"/> IB-R6 <input type="checkbox"/> IBh-R6h <input type="checkbox"/> IBi-R6i	<b>R7</b> <input type="checkbox"/> R7-IB <input type="checkbox"/> IB-R7 <input type="checkbox"/> IBh-R7h <input type="checkbox"/> IBi-R7i	<b>PC</b> <input type="checkbox"/> PC-IB <input type="checkbox"/> IB-PC	<b>ALU</b> <input type="checkbox"/> ADD <input type="checkbox"/> SUB <input type="checkbox"/> OR <input type="checkbox"/> AND <input type="checkbox"/> XOR <input type="checkbox"/> CarryIn
<b>TMPE</b> <input type="checkbox"/> IB-TMPE <input type="checkbox"/> TMPE-SET <input type="checkbox"/> TMPE-CLR	<b>TMPS</b> <input type="checkbox"/> TMPS-IB <input type="checkbox"/> ALU_TMPS	<b>MEMORIA</b> <input type="checkbox"/> WRITE <input type="checkbox"/> READ	<b>E/S</b> <input type="checkbox"/> INTA <input type="checkbox"/> MAR <input type="checkbox"/> IB-MAR	<b>MDR</b> <input type="checkbox"/> MDR-IB <input type="checkbox"/> IB-MDR	<input type="checkbox"/> FIN

Desactivar todas    Aceptar    Cancelar

**Niveles de estudio de una CPU:**

- Nivel de Máquina Convencional: entender cómo una CPU es capaz de ejecutar las instrucciones de un algoritmo.
- Nivel de Micromáquina: entender cómo se ejecuta cada instrucción y cómo colaboran los elementos de la CPU para ejecutarla.

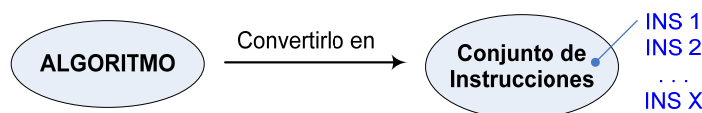


## 4.2- Nivel de Máquina Convencional: Introducción

**Objetivo** de un computador → resolver problemas mediante el diseño e implementación de algoritmos que procesan información.

En general, para resolver un problema deben completarse 3 pasos:

- 1) Expresar el algoritmo en instrucciones máquina.



- 2) Cargar las instrucciones máquina del algoritmo en una zona de memoria del computador.
- 3) Ejecutar las instrucciones (por parte de la CPU).



## 4.2- Nivel de Máquina Convencional: Introducción

### 1) Expresar el algoritmo en instrucciones máquina.

- 1.a) Conocer JUEGO DE INSTRUCCIONES.
- 1.b) PROGRAMA = Secuencia de Instrucciones.
- 1.c) INSTRUCCIONES: Codificadas como secuencias de bits.
- 1.d) Instr. CPU TEÓRICA: Codificadas únicamente con secuencias de 16 bits.

**Condición de diseño**

### 2) Cargar las instrucciones máquina del algoritmo en una zona de memoria del computador.

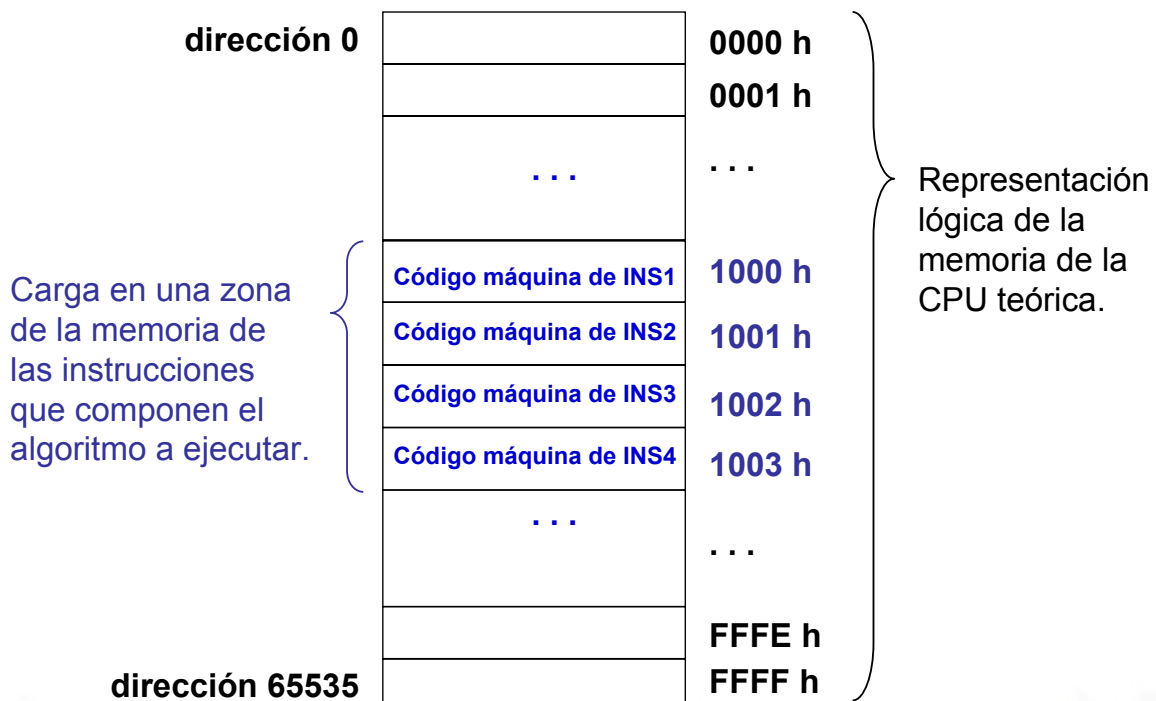
- 2.a) CPU teórica: Una instrucción se codifica como una secuencia de 16 bits. Como cada palabra de memoria de la CPU teórica tiene un ancho de 16 bits → cada instrucción se podrá almacenar en una dirección de memoria.

- 2.b) Ej. de algoritmo: INS1, INS2, INS3, INS4 (→ 4 instr.)

**Consecuencia 1**



## 4.2- Nivel de Máquina Convencional: Introducción





### 3) Ejecución de las instrucciones por parte de la CPU.

- Se parte de un algoritmo formado por un conjunto de instrucciones máquina almacenadas en una determinada zona de la memoria.
- Ejecutar el algoritmo es ejecutar las instrucciones que lo componen.
- Para que el procesador sepa la instrucción que debe ejecutarse en cada momento, deberá conocer la instrucción en curso (registro IR) y la posición de memoria en la que está almacenada la siguiente instrucción a ejecutar (registro PC).
- La ejecución de una instrucción se divide en **2 fases**:
  - a) **FASE 1: Búsqueda de la instrucción e incremento de PC.**
  - b) **FASE 2: Ejecución de la instrucción propiamente dicha.**

*Ambas fases se verán en detalle cuando se estudie la CPU a nivel de micromáquina.*



## 4.2- Nivel de Máq. Convencional: Juego de instrucciones

### INSTRUCCIÓN

- Operación simple realizada con una serie de operandos fuente, cuyo resultado se lleva a un operando destino.
- Ubicación de los operandos de una instrucción: elementos de almacenamiento dentro del computador (registros, memoria).
- Las CPUs se diseñan para que puedan ejecutar un conjunto de instrucciones (también llamado *Instruction Set Architecture*, ISA) concreto.
  - Ejemplo: El Core 2 de Intel y el Athlon 64 de AMD tienen el mismo ISA. El PowerPC de IBM tiene un ISA distinto a los anteriores

### OPERANDO

- Dato con el que trabaja una instrucción.





### MODO DE DIRECCIONAMIENTO

- Cada una de las distintas formas de especificar dónde se almacenan los operandos de una instrucción.
- En el caso de la CPU teórica existen 3 modos de direccionamiento:
  - **Direccionamiento a registro:** El operando está almacenado en un registro (normalmente, uno de propósito general).
  - **Direccionamiento a memoria:** El operando se encuentra almacenado en una posición de memoria cuya dirección hay que suministrar. En la CPU teórica esa dirección siempre está en un registro de propósito general.
  - **Direccionamiento inmediato:** El operando forma parte de la codificación de la instrucción.
- Otras CPUs tienen muchos más modos de direccionamiento.

#### Consecuencia 2



### CONCEPTOS DE CODIFICACIÓN

- Cada instrucción de una CPU debe tener asociada una secuencia de bits distinta. Ejemplo:

Operación:	$R1 \leftarrow R0$ ; (copiar en R1 el contenido de R0)
Código:	0000100100000000
- Puesto que es difícil recordar o saber qué hace una instrucción observando una secuencia de bits, se emplean mnemónicos.
- **Nmemónico:** Representación simbólica de una instrucción. Ejemplo:

Código:	0000100100000000
Mnemónico:	MOV R1, R0
- **Codificación:** Conversión de un mnemónico a la secuencia de bits asociada.



## 4.2- Nivel de Máq. Convencional: Juego de instrucciones

### CODIFICACIÓN EN LA CPU TEÓRICA

- Cada instrucción de la CPU teórica se codifica con una secuencia de 16 bits.
- Los primeros 5 bits se utilizan para identificar el tipo de instrucción. Los restantes 11 sirven para indicar los operandos.
- Existe una hoja de codificación que indica cómo codificar cada instrucción. **Esta hoja se debe traer a clase y a los exámenes.**
- Ejemplo:

	Código de operación	Parámetros		
		Rd	Rs	
		3	6	
MOV Rd, Rs ↔	00001	xxx	xxx	00000
MOV R3, R6 ↔	00001	011	110	00000 ↔ 0BC0 (16)

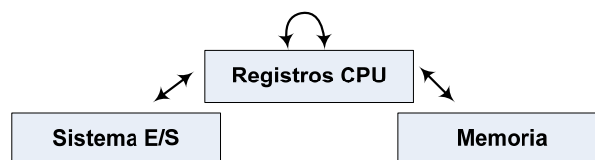


## 4.2- Nivel de Máquina Convencional: Juego de instrucciones

Las instrucciones se pueden agrupar en **3 tipos básicos**:

### 1) Instrucciones de transferencia de información.

Permiten mover datos desde la CPU a otras unidades funcionales y viceversa.



### 2) Instrucciones aritmético-lógicas (suma, resta, and, or, xor, not, etc.)

### 3) Instrucciones de control del flujo de un programa.

Permiten romper la ejecución secuencial del programa realizando ramificaciones en el flujo del mismo.



## 4.2- Nivel de Máquina Convencional: Juego de instrucciones

1) **Instrucciones de movimiento:** llevan información digital de un elemento de almacenamiento a otro.

### ❶ MOV Rd, Rs

- COMETIDO: Mover la información contenida en un registro de propósito general a otro.
- CODIFICACIÓN: 00001 Rd Rs 00000
- EJEMPLO: `MOV R1, R0`  $\equiv 0900_{(16)}$  ( $R1 \leftarrow R0$ )

SITUACIÓN INICIAL

R0	F42E
R1	01D4

`MOV R1, R0`

SITUACIÓN FINAL

R0	F42E
R1	F42E



## 4.2- Nivel de Máquina Convencional: Juego de instrucciones

### ❷ MOV Rd, [Ri]

- COMETIDO: Mover la información contenida en una posición de la memoria a un registro de propósito general.
- CODIFICACIÓN: 00010 Rd Ri 00000
- EJEMPLO: `MOV R1, [R0]`  $\equiv 1100_{(16)}$  ( $R1 \leftarrow [R0]$ )

SITUACIÓN INICIAL

CPU	
R0	A 0 0 7
R1	B 3 B 3

Memoria	
0000 h	
...	
1 8 7 3	A007 h
1 9 7 4	1004 h
...	
FFFF h	

`MOV R1, [R0]`

SITUACIÓN FINAL

CPU	
R0	A 0 0 7
R1	1 8 7 3

Memoria	
0000 h	
...	
1 8 7 3	A007 h
1 9 7 4	1004 h
...	
FFFF h	

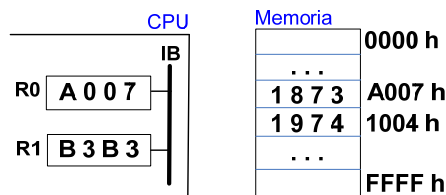


## 4.2- Nivel de Máquina Convencional: Juego de instrucciones

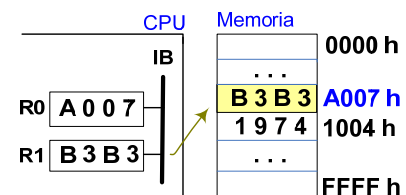
### ③ MOV [Ri], Rs

- COMETIDO: Mover la información contenida en un registro de propósito general a una posición de memoria.
- CODIFICACIÓN: 00011 Ri Rs 00000
- EJEMPLO: `MOV [R0], R1`  $\equiv 1820_{(16)}$  ( $[R0] \leftarrow R1$ )

#### SITUACIÓN INICIAL



#### SITUACIÓN FINAL



## 4.2- Nivel de Máquina Convencional: Juego de instrucciones

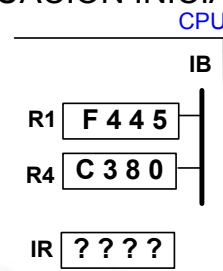
### ④ MOVL Rd, Inm8

- COMETIDO: Mover un valor de 8 bits al byte bajo (el menos significativo) de un registro de propósito general.
- CODIFICACIÓN: 00100 Rd Inm8
- EJEMPLO: `MOVL R1, 19h`  $\equiv 2119_{(16)}$  ( $R1_L \leftarrow 19h$ )

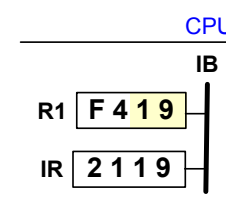
#### Consecuencia 3a

19h = 19<sub>(16)</sub>

#### SITUACIÓN INICIAL



#### SITUACIÓN FINAL



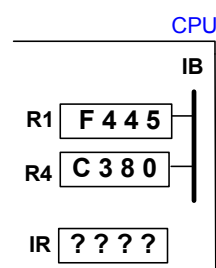
## 4.2- Nivel de Máquina Convencional: Juego de instrucciones

### Consecuencia 3b

#### ⑤ MOVH Rd, Inm8

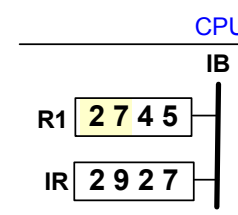
- COMETIDO: Mover un valor de **8 bits al byte alto** (el más significativo) de un registro de propósito general.
- CODIFICACIÓN: 00101 Rd Inm8
- EJEMPLO: `MOVH R1, 27h`  $\equiv 2927_{(16)}$  ( $R1_H \leftarrow 27h$ )

#### SITUACIÓN INICIAL



`MOVH R1, 27h`

#### SITUACIÓN FINAL



## 4.2- Nivel de Máquina Convencional: Juego de instrucciones

### Ejercicio:

#### Estado inicial:

R0: 34B1  
R1: FF00  
R2: 102A

#### Memoria:

Dirección Contenido

FF00	ABCD
FF01	EF12
FF02	4567

#### Programa:

`MOV R0, R1`  
`MOV [R0], R2`  
`MOVL R1, 01h`  
`MOV R2, [R1]`  
`MOVH R2, 0`

#### ¿Estado final?

R0: ????  
R1: ????  
R2: ????

#### Memoria:

Dirección Contenido

FF00	????
FF01	????
FF02	????



## 4.2- Nivel de Máquina Convencional: Juego de instrucciones

### 2) Instrucciones aritmético-lógicas:

- Utilizan la ALU para su ejecución.
- Característica común a todas ellas: modifican el valor de los bits del Registro de Estado, SR (excepto el bit de IF).
- Registro de Estado: único registro de la CPU teórica que no es de 16 bits.
  - IF → Bit de interrupción (se comentará su función cuando se vea el tema de interrupciones).
  - ZF → Bit de Cero (coincide con el bit de Cero generado por la ALU de la CPU teórica).
  - CF → Bit de Carry (coincide con el bit de Carry generado por la ALU de la CPU teórica).
  - OF → Bit de Overflow (coincide con el bit de Overflow generado por la ALU de la CPU teórica).
  - SF → Bit de Signo (coincide con el bit de Signo generado por la ALU de la CPU teórica).
- Tras ejecutar una instrucción aritmética tienen sentido ZF, CF, OF y SF.
- Tras ejecutar una instrucción lógica sólo tiene sentido el bit ZF.



## 4.2- Nivel de Máquina Convencional: Juego de instrucciones

### 1 ADD Rd, Rs1, Rs2

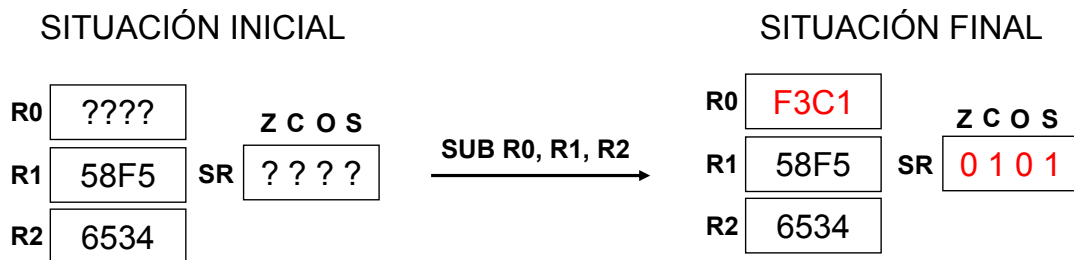
- COMETIDO:  $Rd \leftarrow Rs1 + Rs2$  (suma aritmética).
- CODIFICACIÓN: 01000 Rd Rs1 Rs2 00
- EJEMPLO: `ADD R0, R1, R2`  $\equiv 4028_{(16)}$



## 4.2- Nivel de Máquina Convencional: Juego de instrucciones

### ② SUB Rd, Rs1, Rs2

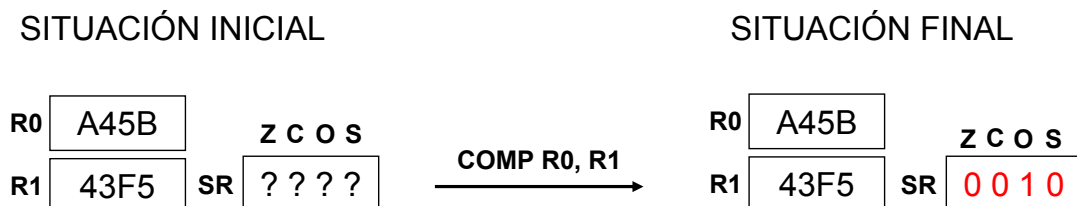
- COMETIDO:  $Rd \leftarrow Rs1 - Rs2$  (resta aritmética).
- CODIFICACIÓN: 01001 Rd Rs1 Rs2 00
- EJEMPLO: `SUB R0, R1, R2`  $\equiv 4828_{(16)}$



## 4.2- Nivel de Máquina Convencional: Juego de instrucciones

### ③ COMP Rs1, Rs2

- COMETIDO:  $Rs1 - Rs2$  (comparación).
- CODIFICACIÓN: 01101 Rs1 Rs2 00000
- EJEMPLO: `COMP R0, R1`  $\equiv 6820_{(16)}$

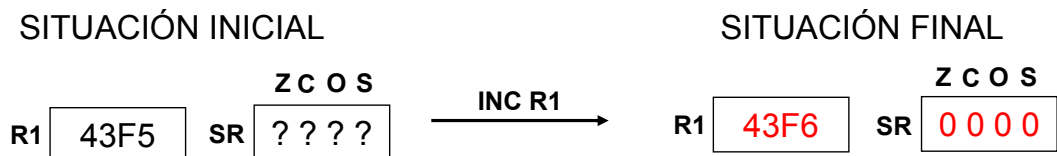




## 4.2- Nivel de Máquina Convencional: Juego de instrucciones

### ④ INC Rd/s

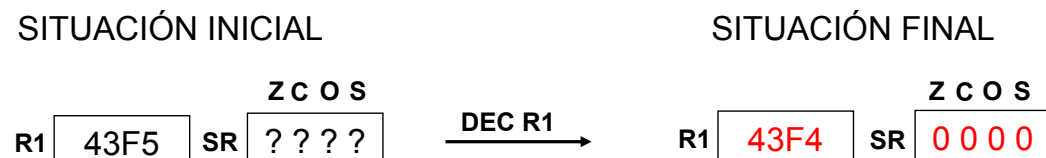
- COMETIDO:  $Rd/s \leftarrow Rd/s + 1$  (incremento en una unidad).
- CODIFICACIÓN: 10001 Rd/s 00000000
- EJEMPLO: `INC R1`  $\equiv 8900_{(16)}$



## 4.2- Nivel de Máquina Convencional: Juego de instrucciones

### ⑤ DEC Rd/s

- COMETIDO:  $Rd/s \leftarrow Rd/s - 1$  (decremento en una unidad).
- CODIFICACIÓN: 10010 Rd/s 00000000
- EJEMPLO: `DEC R1`  $\equiv 9100_{(16)}$



## 4.2- Nivel de Máquina Convencional: Juego de instrucciones

### ⑥ NEG Rd/s

- COMETIDO:  $Rd/s \leftarrow Rd/s_{(C-2)} = Rd/s \leftarrow Rd/s_{(C-1)} + 1 = Rd/s \leftarrow 0 - Rd/s$
- CODIFICACIÓN: 10011 Rd/s 00000000
- EJEMPLO: `NEG R1`  $\equiv 9900_{(16)}$

SITUACIÓN INICIAL

R1 43F5 SR Z C O S  
? ? ? ?

$\xrightarrow{\text{NEG R1}}$

SITUACIÓN FINAL

R1 BC0B SR Z C O S  
0 1 0 1



## 4.2- Nivel de Máquina Convencional: Juego de instrucciones

### ⑦ OR Rd, Rs1, Rs2

- COMETIDO:  $Rd \leftarrow Rs1 + Rs2$  (suma lógica).
- CODIFICACIÓN: 01010 Rd Rs1 Rs2 00
- EJEMPLO: `OR R0, R1, R2`  $\equiv 5028_{(16)}$

SITUACIÓN INICIAL

R0 ????  
R1 43F5  
R2 6534 SR Z C O S  
? ? ? ?

$\xrightarrow{\text{OR R0, R1, R2}}$

SITUACIÓN FINAL

R0 67F5  
R1 43F5  
R2 6534 SR Z C O S  
0 0 0 0



## 4.2- Nivel de Máquina Convencional: Juego de instrucciones

### ⑧ AND Rd, Rs1, Rs2

- COMETIDO:  $Rd \leftarrow Rs1 \cdot Rs2$  (producto lógico).
- CODIFICACIÓN: 01011 Rd Rs1 Rs2 00
- EJEMPLO: `AND R0, R1, R2`  $\equiv 5828_{(16)}$

SITUACIÓN INICIAL

R0	????	SR	Z C O S ? ? ? ?
R1	43F5		
R2	6534		

AND R0, R1, R2

SITUACIÓN FINAL

R0	4134	SR	Z C O S 0 0 0 0
R1	43F5		
R2	6534		



## 4.2- Nivel de Máquina Convencional: Juego de instrucciones

### ⑨ XOR Rd, Rs1, Rs2

- COMETIDO:  $Rd \leftarrow Rs1 \oplus Rs2$  (OR-exclusivo).
- CODIFICACIÓN: 01100 Rd Rs1 Rs2 00
- EJEMPLO: `XOR R0, R1, R2`  $\equiv 6028_{(16)}$

SITUACIÓN INICIAL

R0	????	SR	Z C O S ? ? ? ?
R1	43F5		
R2	6534		

XOR R0, R1, R2

SITUACIÓN FINAL

R0	26C1	SR	Z C O S 0 0 0 0
R1	43F5		
R2	6534		



## 4.2- Nivel de Máquina Convencional: Juego de instrucciones

### ⑩ NOT Rd/s

- COMETIDO:  $Rd/s \leftarrow Rd/s_{(C-1)}$
- CODIFICACIÓN: 10000 Rd/s 00000000
- EJEMPLO: NOT R1  $\equiv 8100_{(16)}$

SITUACIÓN INICIAL

R1 43F5 SR Z C O S  
? ? ? ?

NOT R1

SITUACIÓN FINAL

R1 BC0A SR Z C O S  
0 0 0 1



## 4.2- Nivel de Máquina Convencional: Juego de instrucciones

### Ejercicios propuestos:

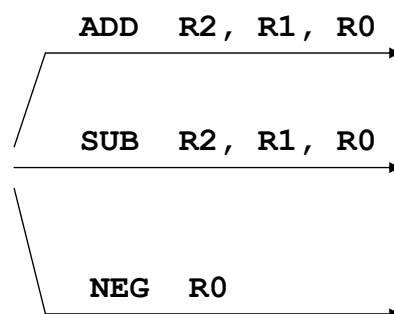
#### Situación inicial:

R2 = XXXX

R1 = F000h

R0 = 1000h

ZCOS = xxxx

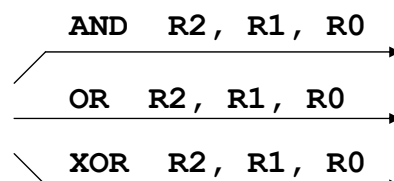


R2	R1	R0	ZCOS
????	????	????	????
????	????	????	????
????	????	????	????

#### Situación inicial:

R0 = B10Ah

R1 = 2D0Eh



R2 = ????
R2 = ????
R2 = ????



## 4.2- Nivel de Máquina Convencional: Juego de instrucciones

### Ejercicio propuesto:

- **PROBLEMA:** Sumar tres números almacenados en posiciones consecutivas de la memoria y guardar el resultado en memoria, en la siguiente posición.
- **TAREAS A REALIZAR PARA SOLUCIONAR UN PROBLEMA CON UN COMPUTADOR:**
  1. Expresar los datos en un sistema de representación y colocarlos en una zona de memoria.
  2. Construir el algoritmo con las instrucciones de la CPU (la CPU teórica, en nuestro caso) y ubicar dichas instrucciones en una zona de la memoria. Esto implica seleccionar qué elementos de la CPU se van a utilizar para qué cosas.
  3. Ejecutar el algoritmo.



## 4.2- Nivel de Máquina Convencional: Juego de instrucciones

### 3) Instrucciones de control del flujo de un programa:

- Todos los programas formados exclusivamente por instrucciones de movimiento, aritméticas y lógicas se ejecutan de forma secuencial.

*Por ejemplo, si INS1, INS2, INS3 e INS4 son instrucciones de movimiento, aritméticas y lógicas, y el programa comienza la ejecución en la posición de memoria 1000h, se ejecutará primero la instrucción INS1 (almacenada en la dirección 1000h), a continuación la INS2 (almacenada en la dirección 1001h), luego INS3 (dirección 1002h) y por último la INS4 (almacenada en la dirección 1003h).*

0000h	
0001h	
-----	
1000h	INS1
1001h	INS2
1002h	INS3
1003h	INS4
-----	
FFFEh	
FFFFh	



## 4.2- Nivel de Máquina Convencional: Juego de instrucciones

- Se pueden insertar instrucciones que **modifiquen el valor del registro PC** para alterar el orden de ejecución de los programas. Es necesario tener en cuenta el funcionamiento del PC y las dos fases de ejecución de una instrucción.

0000h	
0001h	
-----	
1000h	INS1
1001h	PC←PC+2
1002h	INS2
1003h	INS3
1004h	INS4
1005h	PC←PC-3
-----	
FFFEh	
FFFFh	

- Orden de Ejecución:  
INS1, INS4, INS3, INS4, INS3, INS4, ...

- Tipos de saltos: **incondicionales** (el valor de PC se altera siempre) y **condicionales** (el valor de PC se altera si se cumple una determinada condición).



## 4.2- Nivel de Máquina Convencional: Juego de instrucciones

### Instrucciones de Salto Incondicional.

#### ❶ JMP Rx

- COMETIDO:  $PC \leftarrow Rx$  (salto absoluto).
- CODIFICACIÓN: 11001 Rx 00000000
- EJEMPLO:  $JMP\ R3 \equiv CB00_{(16)}$

SITUACIÓN INICIAL

R3	A45B
PC	????

JMP R3

SITUACIÓN FINAL

R3	A45B
PC	A45B



## 4.2- Nivel de Máquina Convencional: Juego de instrucciones

### Instrucciones de Salto Incondicional.

#### ② JMP Inm8

- COMETIDO:  $PC \leftarrow PC + \text{Ext16}(\text{Inm8})$  (salto relativo).
- CODIFICACIÓN: 11000 000 Inm8
- EJEMPLO:  $\text{JMP } -2 \equiv \text{C0FE}_{(16)}$

SITUACIÓN INICIAL

PC BF85

JMP -2

SITUACIÓN FINAL

PC BF84

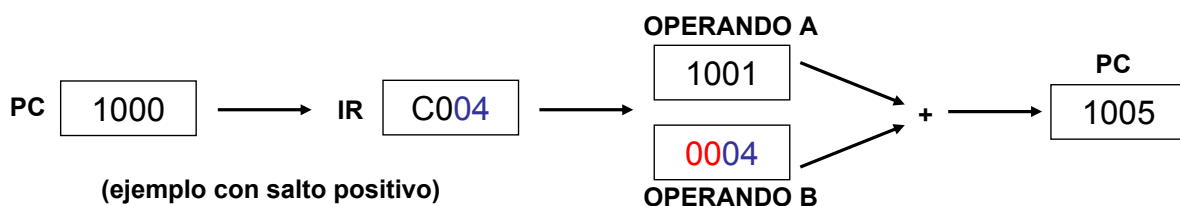
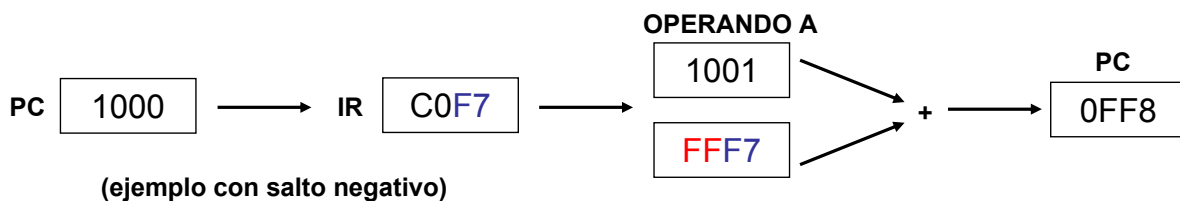
**PROBLEMA:** ¿Cómo sumar una cantidad de 16 bits (PC) y otra de 8 bits (Inm8) en C-2?

**SOLUCIÓN:** Utilizar la **EXTENSIÓN DE SIGNO**. (Consecuencia 4)



## 4.2- Nivel de Máquina Convencional: Juego de instrucciones

### Extensión de signo.





## 4.2- Nivel de Máquina Convencional: Juego de instrucciones

### Instrucciones de Salto Condicional.

#### ① **BRCOND** Inm8

- COMETIDO: Si COND es cierta,  $PC \leftarrow PC + \text{Ext16}(\text{Inm8})$   
(salto relativo condicional)
- CODIFICACIÓN: 11110 COND Inm8

#### COND

C → 000 → Salta si C = 1 → BRC  
 NC → 001 → Salta si C = 0 → BRNC  
 O → 010 → Salta si O = 1 → BRO  
 NO → 011 → Salta si O = 0 → BRNO  
 Z → 100 → Salta si Z = 1 → BRZ  
 NZ → 101 → Salta si Z = 0 → BRNZ  
 S → 110 → Salta si S = 1 → BRS  
 NS → 111 → Salta si S = 0 → BRNS

- EJEMPLO: BRNO 80h  $\equiv$  F380<sub>(16)</sub>

#### SITUACIÓN INICIAL

PC 43F5    SR **Z C O S**  
                  1 0 0 1

BRNO 80h →

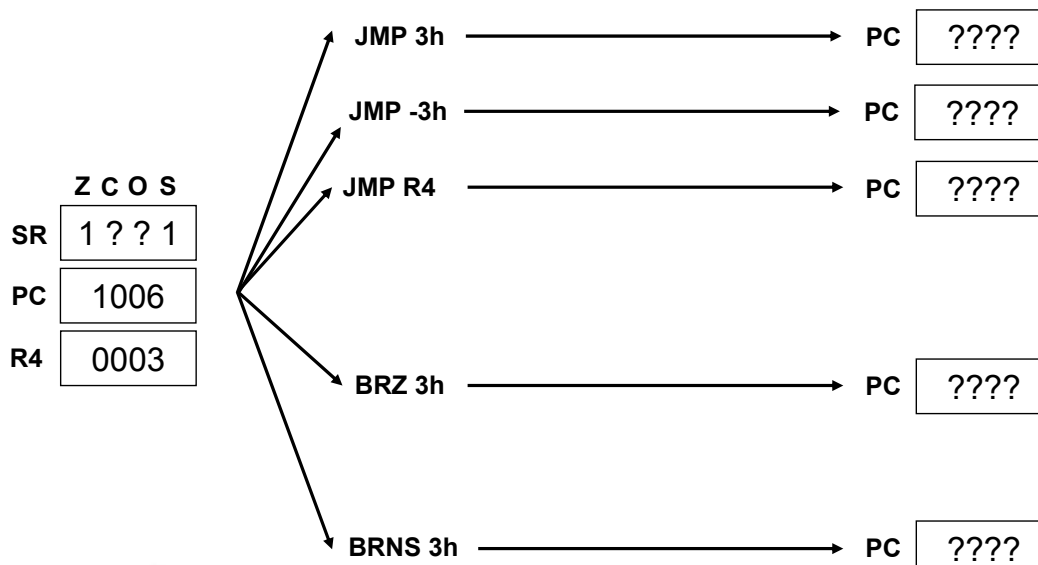
#### SITUACIÓN FINAL

PC **4376**    SR **Z C O S**  
                  1 0 0 1



## 4.2- Nivel de Máquina Convencional: Juego de instrucciones

### Ejemplos de saltos:



## 4.2- Nivel de Máquina Convencional: Juego de instrucciones

### Ejercicio propuesto:

- **PROBLEMA:** Sumar 40 números almacenados en posiciones consecutivas de la memoria y guardar el resultado en la posición siguiente de la memoria.



## 4.2- Nivel de Máquina Convencional: Juego de instrucciones

### Ejercicios propuestos:

- ❶ Indicar el orden de ejecución en este fragmento de código, sabiendo que el registro PC de la CPU teórica contiene inicialmente **1001h**. No incluir en la respuesta las sentencias JMP. En caso de llegar a un bucle sin fin, poner puntos suspensivos para indicarlo. Ejemplo: INS1, INS3, INS2,...

----	----
1000h	INS1
1001h	JMP 1
1002h	INS2
1003h	INS3
1004h	INS4
1005h	JMP -3
----	----

- ❷ Se desea saltar 2 instrucciones (después de la de salto) cuando en una operación aritmética previa se ha obtenido cero como resultado. Codificar en hexadecimal la instrucción de salto condicional que se debe emplear.



## 4.2- Nivel de Máquina Convencional: Juego de instrucciones

### Ejercicios de Codificación propuestos:

<u>INSTRUCCIÓN</u>	<u>FORMATO GENERAL</u>	<u>CODIFICACIÓN (bin)</u>	<u>COD (hex)</u>
• NOP	00000 000000000000	00000 000000000000	0000
• MOV R5, R3	00001 Rd Rs 00000	00001 101 011 00000	0D60
• MOV R4, [R6]	00010 Rd Ri 00000	00010 100 110 00000	14C0
• MOV [R1], R2	00011 Ri Rs 00000	00011 001 010 00000	1940
• MOVL R1, 25h	00100 Rd Inm_8	00100 001 00100101	2125
• MOVH R1, F0h	00101 Rd Inm_8	00101 001 11110000	29F0
• ADD R3, R4, R5	01000 Rd Rs1 Rs2 00	01000 011 100 101 00	4394
• NOT R1	10000 Rd/s 00000000	10000 001 00000000	8100
• JMP 4	11000 000 Inm_8	11000 000 00000100	C004
• JMP -4	11000 000 Inm_8	11000 000 11111100	C0FC



## 4.2- Nivel de Máquina Convencional: Ejercicios propuestos

### ❶ El siguiente fragmento de código:

- 1) Toma los datos (números enteros) que están almacenados a partir de una posición de memoria apuntada por R0 y
- 2) comprueba para, cada uno de ellos, si el dato es menor o igual que 15;
- 3) en caso de serlo, se incrementará un contador (R1).

El proceso se repite el nº de veces indicado en el registro R6.

Se pide:

- a) Especificar la condición de la instrucción **BR??** para que el programa funcione correctamente.
- b) Codificación (en hexadecimal) de la instrucción **BRNZ -7**.

```
XOR R1, R1, R1
MOVL R2, 15
MOVH R2, 0
MOV R5, [R0]
COMP R2, R5
BR?? +1
INC R1
INC R0
DEC R6
BRNZ -7
```



## 4.2- Nivel de Máquina Convencional: Ejercicios propuestos

- ② El siguiente fragmento de código recorre una lista de valores, almacenada a partir de la dirección contenida en el registro R0, hasta que encuentre 2 valores consecutivos que sean iguales. En ese momento finalizará el recorrido y continuará ejecutando las instrucciones ubicadas a partir del salto incondicional:

```
MOV R1, [R0]
INC R0
MOV R2, R1
MOV R1, [R0]
COMP R2, R1
BR?? +2
INC R0
JMP -6
. . .
```

Se pide:

- Especificar la condición de la instrucción **BR??** para que el programa funcione correctamente. (Solución: **BRZ +2**)
- Codificación (en hexadecimal) de la instrucción **JMP -6**. (Solución: **C0FAh**)



## 4.3- Nivel de Micromáquina: Introducción

- Las instrucciones se ejecutan mediante micro-operaciones sobre los elementos del camino de datos (Registros, IB y ALU).
- Los elementos de la CPU tienen entradas de control para recibir órdenes de gobierno (por ejemplo: la carga o descarga de un registro).
  - R5-IB → señal de descarga del registro R5
  - IB-R5 → señal de carga del registro R5
  - ADD → señales de la ALU Op0 = 1, Op1 = 1, Resta = 0, Cin = 0
- La Unidad de Control es el elemento de la CPU que genera las órdenes de gobierno.

**Ejemplo:** Señales de control para copiar la información del registro R5 al R7.

R5-IB → Descarga del registro R5 sobre el IB

IB-R7 → Carga del registro R7 desde el IB



**Ejemplo:** Señales de control para ADD R0, R1, R2

R1-IB, IB-TMPE, R2-IB, ADD, ALU-TMPS, ALU-SR, TMPS-IB, IB-R0

No se pueden hacer todas las señales a la vez porque:

- IB no puede tener a la vez R1, R2 y el resultado.
- ALU-TMPS y ALU-SR no se pueden activar hasta que estén los operandos en las entradas de la ALU y se haya hecho la suma

**Conclusión:** hay que sincronizar las señales y agruparlas en pasos de ejecución



- La Unidad de Control (U.C.) genera las órdenes de control siguiendo las siguientes pautas:
  - La U.C. tiene memorizado el conjunto de acciones que deben llevarse a cabo para la ejecución de cada instrucción.
  - Las acciones a ejecutar se organizan en **pasos de ejecución** (durante un paso la información del IB permanece inalterada).
  - La U.C. está gobernada por una señal de reloj, de forma que cada paso de ejecución corresponde a un periodo de la señal de reloj.



### Transferencia de información con la memoria

- La velocidad de las CPUs no suele coincidir con la de las memorias  
→ cuando se le pide un dato a la memoria hay que sincronizarse con ella para saber cuándo lo tiene listo
- En la CPU teórica, supondremos que la memoria tarda un ciclo de reloj en leer o escribir un dato → cuando se escriba o lea un dato, se debe esperar un ciclo, que se denominará **ciclo de espera**
- La comunicación entre la CPU y la memoria en la CPU teórica se hace a través de los registros MAR y MDR junto con los buses del sistema SDB, SAD y SCB.



### a) Lectura de una posición de memoria

- a.1) La CPU coloca en el registro MAR la dirección de memoria que quiere leer, y activa la señal de control READ.
- a.2) La CPU espera por el dato durante 1 ciclo.
- a.3) La memoria coloca el dato solicitado en el bus de datos, quedando disponible en el registro MDR.

### b) Escritura en una posición de memoria

- b.1) La CPU coloca en MAR la dirección de memoria en la que quiere escribir, en MDR coloca el dato, y activa la señal de control WRITE.
- b.2) La CPU espera durante 1 ciclo a que la memoria sea actualizada.



### Señales de control de la CPU teórica

## 4.3- Nivel de Micromáquina: Pasos de Ejecución

- Fases de ejecución de las instrucciones:

- I) Búsqueda de la instrucción e incremento de contador de programa, PC (igual para todas las instrucciones).

Paso	Señales de control activas
1	PC-IB, IB-MAR, READ, TMPE_CLR, CARRY_IN, ADD, ALU-TMPS
2	TMPS-IB, IB-PC (PC ← PC+1, y además se aprovecha como ciclo de espera del READ)
3	MDR-IB, IB-IR



■ IR ← [PC]

■ PC ← PC + 1

- II) Interpretación y ejecución de la instrucción (específica para cada instrucción).

*Se incluyen a continuación los pasos de ejecución de algunas instrucciones (juego completo disponible en apuntes).*



### 4.3- Nivel de Micromáquina: Pasos de ejecución

#### A) Instrucciones de Transferencia de información (algunos ejemplos)

**MOV R2, R4**

Paso	Señales de control activas
4	R4-IB, IB-R2, FIN

**MOV R2, [R4]**

Paso	Señales de control activas
4	R4-IB, IB-MAR, READ
5	Ciclo de espera
6	MDR-IB, IB-R2, FIN

**MOV [R4], R2**

Paso	Señales de control activas
4	R4-IB, IB-MAR
5	R2-IB, IB-MDR, WRITE
6	FIN (y ciclo de espera)



### 4.3- Nivel de Micromáquina: Pasos de ejecución

**MOVL R6, Inm8**

Paso	Señales de control activas
4	IR <sub>L</sub> -IB <sub>L</sub> , IB <sub>L</sub> -R6 <sub>L</sub> , FIN

Ejemplo: MOVL R6, 07h → 2607 (código máquina)  
IR<sub>L</sub>

**MOVH R7, Inm8**

Paso	Señales de control activas
4	IR <sub>L</sub> -IB <sub>H</sub> , IB <sub>H</sub> -R7 <sub>H</sub> , FIN

Ejemplo: MOVH R7, 10h → 2F10 (código máquina)  
IR<sub>L</sub>



### 4.3- Nivel de Micromáquina: Pasos de ejecución

#### B) Instrucciones Aritmético-Lógicas (algunos ejemplos)

##### ADD R0, R1, R2

Paso	Señales de control activas
4	R1-IB, IB-TMPE
5	R2-IB, ADD, ALU-TMPS, ALU-SR
6	TMPS-IB, IB-R0, FIN

##### SUB R0, R1, R2

Paso	Señales de control activas
4	R1-IB, IB-TMPE
5	R2-IB, SUB, ALU-TMPS, ALU-SR
6	TMPS-IB, IB-R0, FIN

##### COMP R1, R2

Paso	Señales de control activas
4	R1-IB, IB-TMPE
5	R2-IB, SUB, ALU-SR, FIN

La instrucción COMP es una resta de la que no interesa el resultado, sólo los valores del registro SR.



### 4.3- Nivel de Micromáquina: Pasos de ejecución

##### INC R0

$R0 \leftarrow R0 + 1$

Paso	Señales de control activas
4	R0-IB, TMPE_CLR, CarryIn, ADD, ALU-TMPS, ALU-SR
5	TMPS-IB, IB-R0, FIN

##### DEC R0

$R0 \leftarrow R0 - 1$

Paso	Señales de control activas
4	R0-IB, TMPE_SET, ADD, ALU-TMPS, ALU-SR
5	TMPS-IB, IB-R0, FIN

##### NEG R0

$R0 \leftarrow 0 - R0$

Paso	Señales de control activas
4	R0-IB, TMPE_CLR, SUB, ALU-TMPS, ALU-SR
5	TMPS-IB, IB-R0, FIN

##### NOT R0

$R0 \leftarrow \overline{R0}$

Paso	Señales de control activas
4	R0-IB, TMPE_SET, XOR, ALU-TMPS, ALU-SR
5	TMPS-IB, IB-R0, FIN



### 4.3- Nivel de Micromáquina: Pasos de ejecución

#### C) Instrucciones de Control del Flujo (algunos ejemplos)

##### JMP R0

$PC \leftarrow R_s$

Paso	Señales de control activas
4	R0-IB, IB-PC, FIN

Salto incondicional absoluto.

##### JMP Inm8

$PC \leftarrow PC + \text{Ext16}(\text{Inm8})$

Paso	Señales de control activas
4	PC-IB, IB-TMPE
5	JUMP, ADD, ALU-TMPS
6	TMPS-IB, IB-PC, FIN

Salto incondicional relativo.

##### BR<sub>cond</sub> Inm8

Si **cond** cierta:

$PC \leftarrow PC + \text{Ext16}(\text{Inm8})$

Paso	Señales de control activas
4	FIN

Si **cond** falsa.

Paso	Señales de control activas
4	PC-IB, IB-TMPE
5	JUMP, ADD, ALU-TMPS
6	TMPS-IB, IB-PC, FIN

Si **cond** cierta.

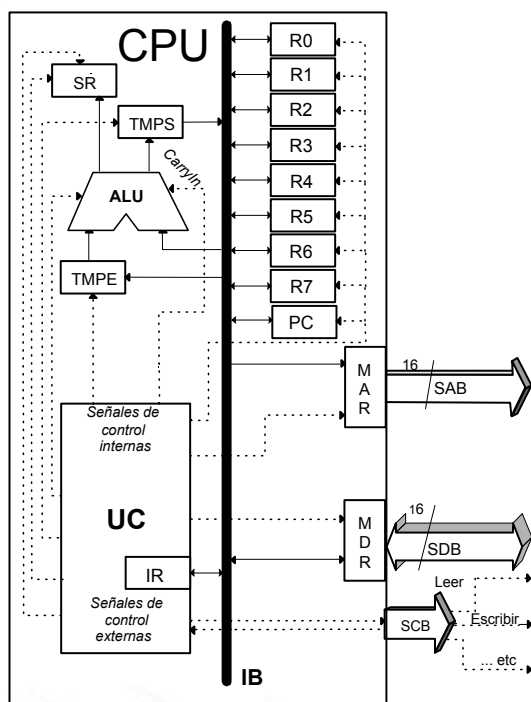


Área de Arquitectura y Tecnología de Computadores  
Departamento de Informática de la Universidad de Oviedo

Estructura de Computadores  
Tema 4: La CPU

63

### 4.3- Nivel de Micromáquina: ejemplo gráfico



	0000 h
	0001 h
...	...
C. M. de la INS 1	1000 h
C. M. de la INS 2	1001 h
C. M. de la INS 3	1002 h
C. M. de la INS 4	1003 h
...	...
	FFFE h
	FFFF h

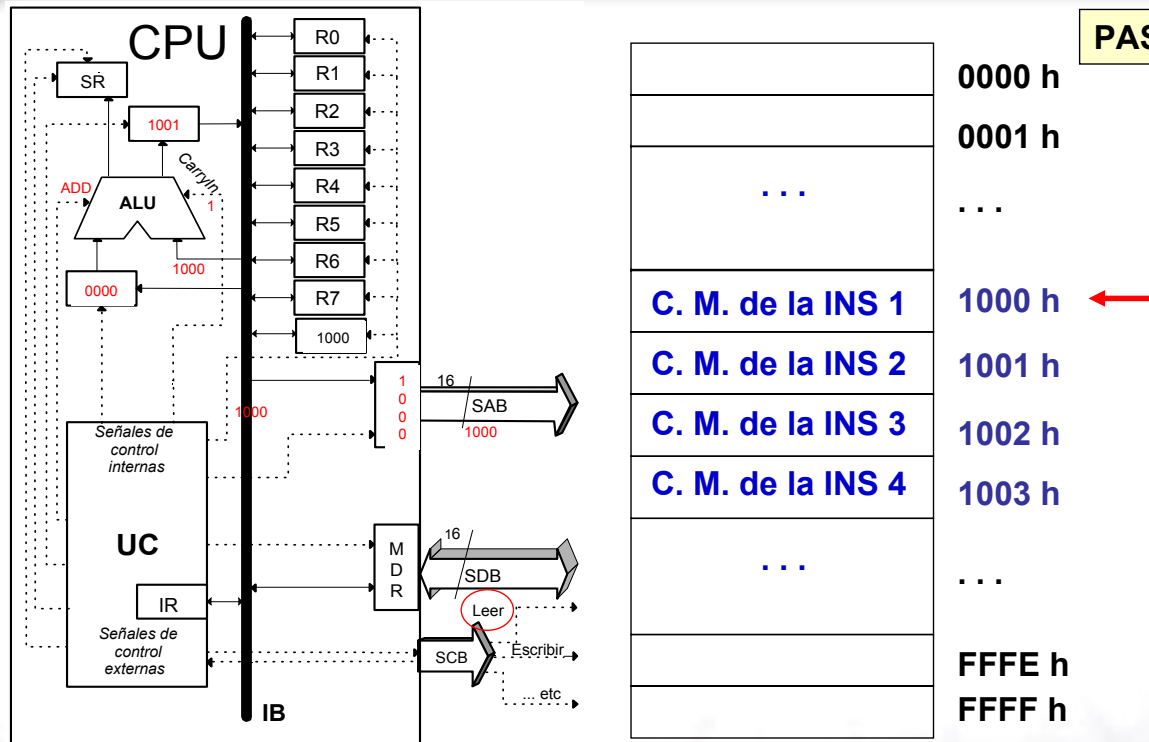


Área de Arquitectura y Tecnología de Computadores  
Departamento de Informática de la Universidad de Oviedo

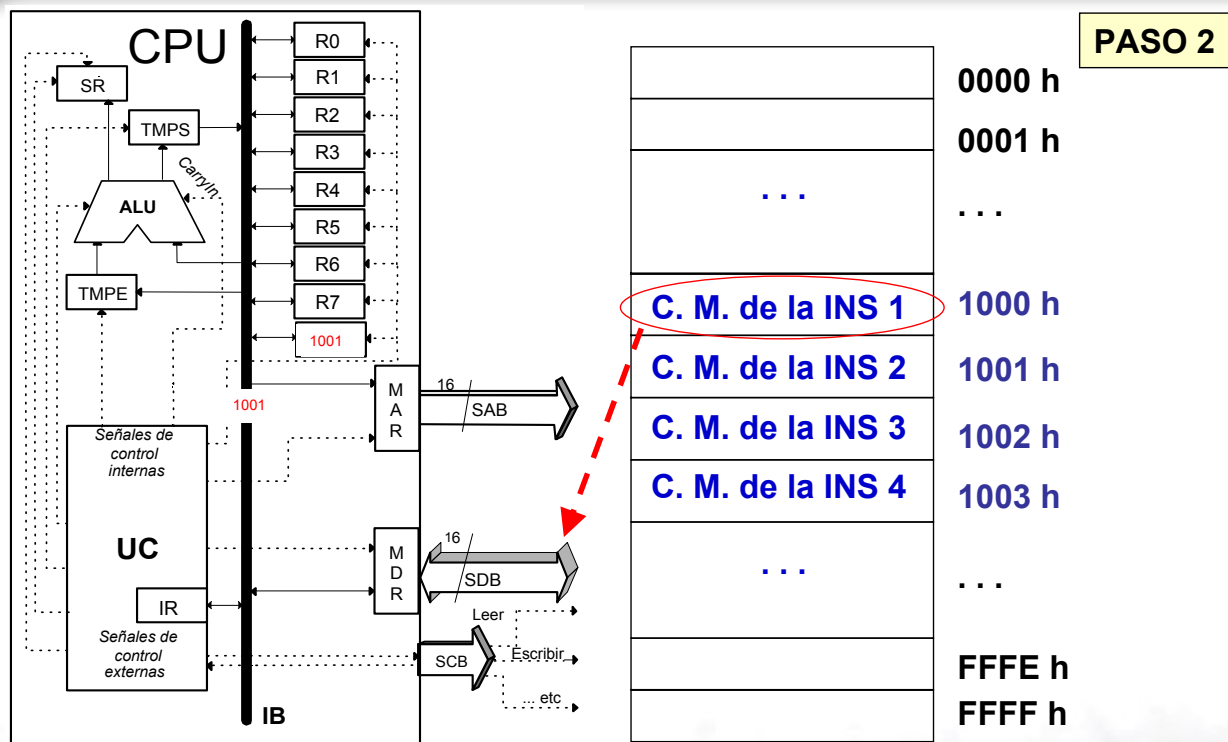
Estructura de Computadores  
Tema 4: La CPU

64

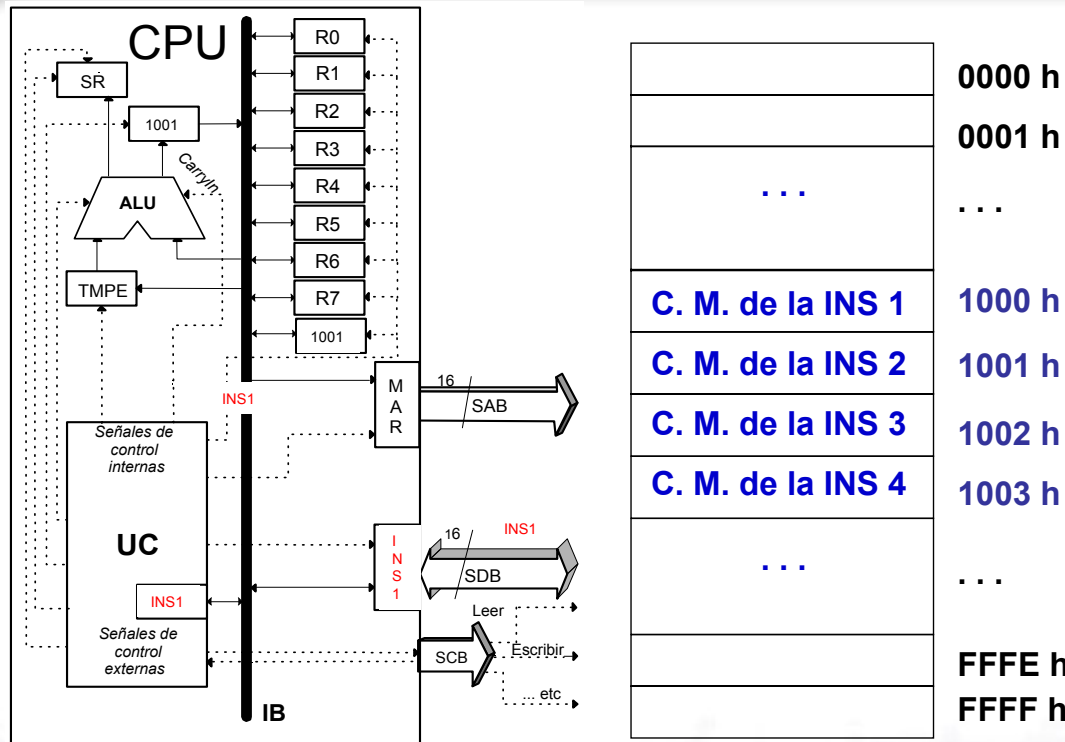
#### 4.3- Nivel de Micromáquina: ejemplo gráfico



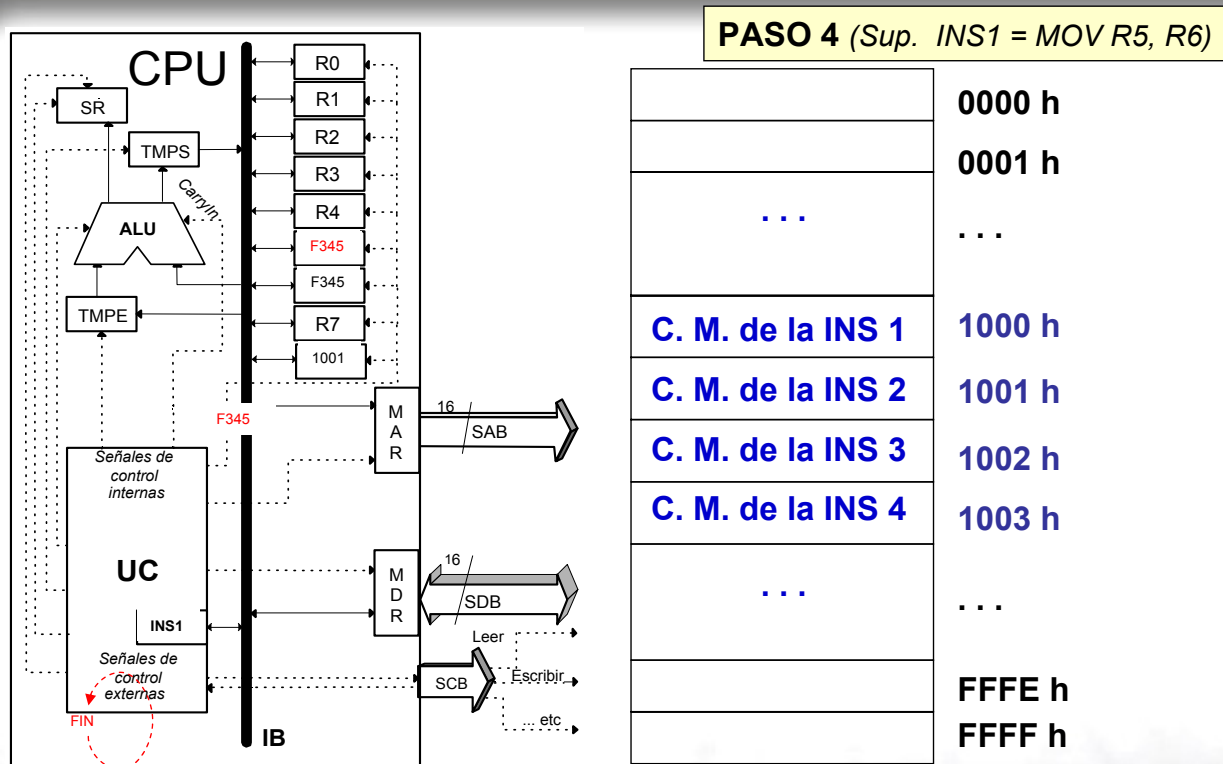
#### 4.3- Nivel de Micromáquina: ejemplo gráfico



### 4.3- Nivel de Micromáquina: ejemplo gráfico

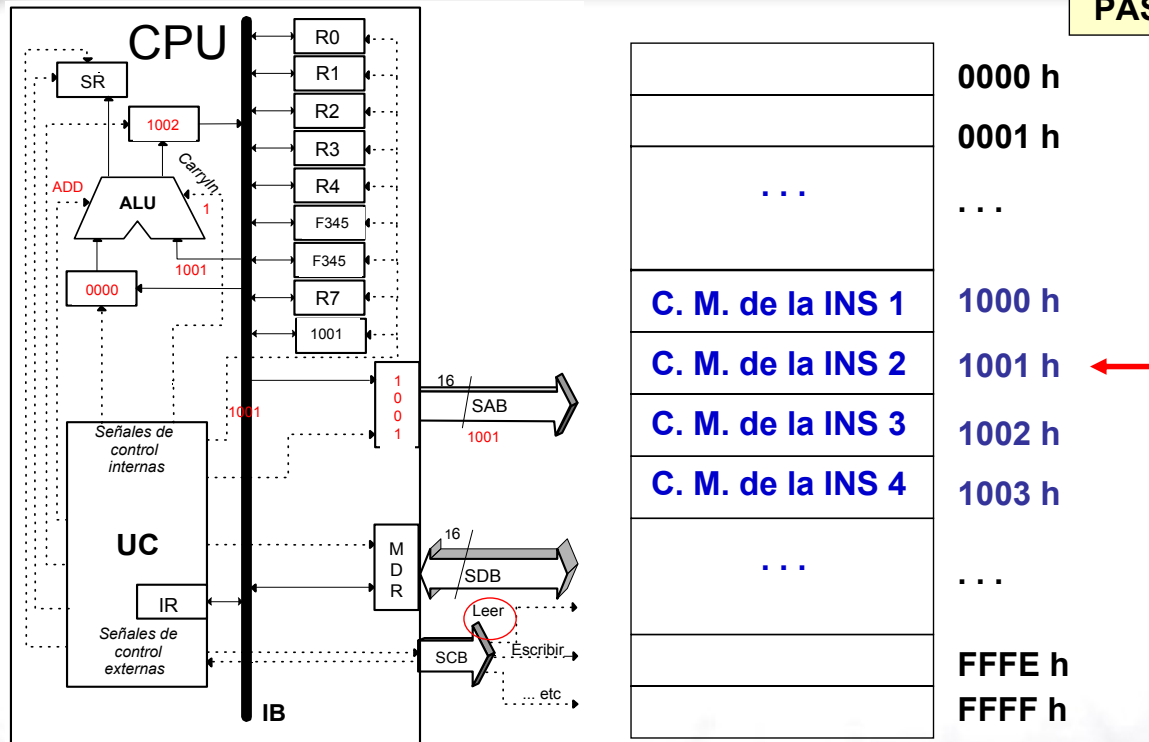


### 4.3- Nivel de Micromáquina: ejemplo gráfico



### 4.3- Nivel de Micromáquina: ejemplo gráfico

**PASO 1**



### 4.3- Nivel de Micromáquina: Ejercicios

❶ Dada la siguiente secuencia de señales:

Paso	Señales de control
1	ALU-TMPS, TMPE_CLR, ADD, PC-IB, READ, CarryIn
2	TMPS-IB, IB-PC
3	MDR-IB, IB-IR
4	R6-IB, TMPE_SET, XOR, ALU-TMPS, ALU-SR
5	TMPS-IB, IB-R6, FIN

¿A qué instrucción corresponden dichas señales de control?:

- a) DEC R6
- b) INC R6
- c) NOT R6
- d) NEG R6



## 4.3- Nivel de Micromáquina: Ejercicios

- ② Se quiere implementar una nueva instrucción en la CPU teórica, la cual sume un dato inmediato de 8 bits al valor de un registro de propósito general (Rs). La sintaxis de dicha instrucción sería:

ADD Rs, Inm8

Indicar cuántos y qué pasos de ejecución que serían necesarios para ejecutar la instrucción ADD R2, 5h.

**Solución:**

*Serían necesarios 6 pasos de ejecución:*

Paso	Señales de control activas
1	PC-IB, IB-MAR, TMPE_CLR, CarryIn, ADD, ALU-TMPS, READ
2	TMPS-IB, IB-PC
3	MDR-IB, IB-IR
4	R2-IB, IB-TMPE
5	JUMP, ADD, ALU-TMPS, ALU-SR
6	TMPS-IB, IB-R2, FIN



## 4.4- La Unidad de Control (UC)

- Elemento de la CPU que genera las señales de control necesarias para que se ejecuten las instrucciones.
- Es un circuito digital secuencial síncrono, es decir, no sólo genera las señales necesarias sino también una secuenciación adecuada de las mismas.
- Dichas secuencias se dividen en pasos.
- Cada paso dura 1 periodo (ciclo) de reloj.

**Ejemplo:** ¿cuánto tardará una CPU en ejecutar una instrucción que tiene 6 pasos de ejecución, si la frecuencia de su reloj es de 1MHz?

$$f = 1 \text{ MHz} \rightarrow T = 1/f = 1/10^6 = 1\mu\text{s}$$

La instrucción tardará en ejecutarse:  $6 \times 1 \mu\text{s} = 6 \mu\text{s}$





## 4.4- La Unidad de Control: Ejercicios propuestos

- ❶ Si se construye un computador con la CPU teórica vista en clase, y se dispone de un reloj de frecuencia 10 GHz, ¿cuánto tiempo tardará en ejecutarse el siguiente fragmento de código:

```
XOR R0, R0, R0
MOVH R1, 2Dh
MOVL R1, 5
SUB R2, R1, R0
COMP R1, R0
BRNZ +2
```



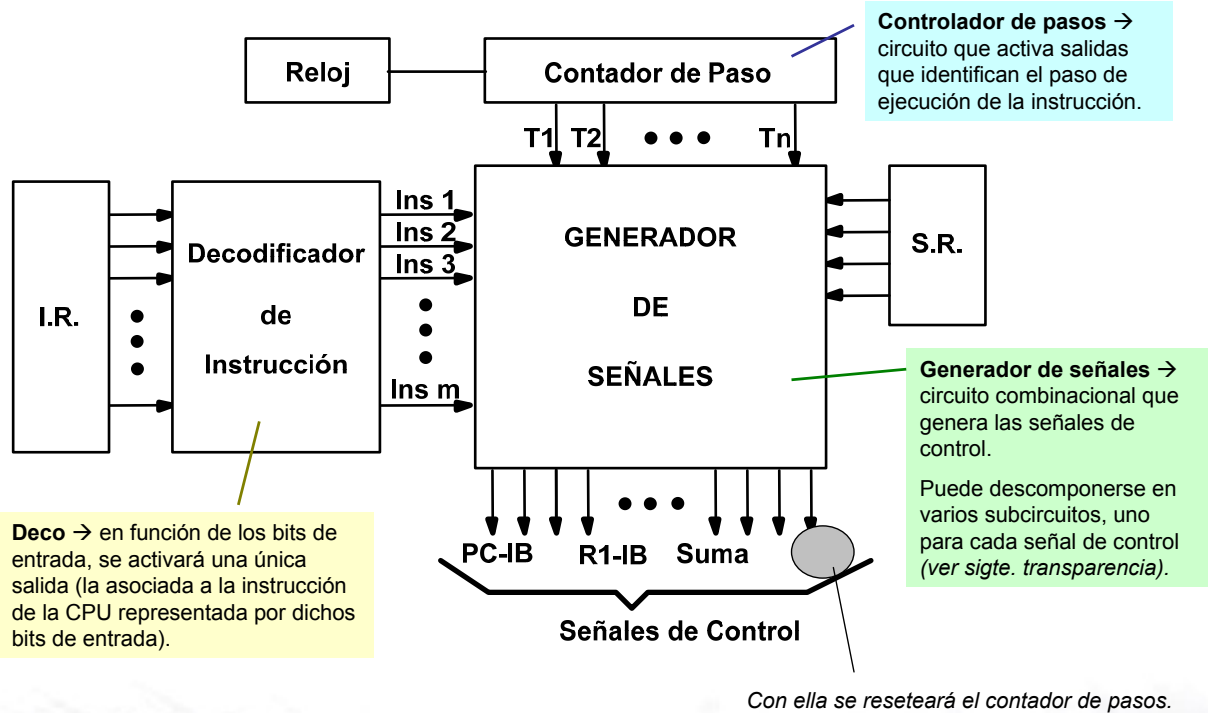
## 4.4- La Unidad de Control (UC)

### ESTRUCTURA INTERNA DE UNA U.C.

- Que una señal se active en un momento dado dependerá de:
  - i) La instrucción que se esté ejecutando (→ *registro IR*)
  - ii) El paso de ejecución de dicha instrucción (→ *reloj*).
  - iii) Valores ZCOS del Registro de Estado (→ *SR*) (*para los saltos*).
- Por tanto: Entradas → registro IR + señal de Reloj + registro SR  
Salidas → Secuencia de señales de control.
- Dependiendo del tipo de circuitos empleados para obtener las señales de control, las U.C. pueden clasificarse en:
  - a) U.C. cableada.
  - b) U.C. microprogramada.

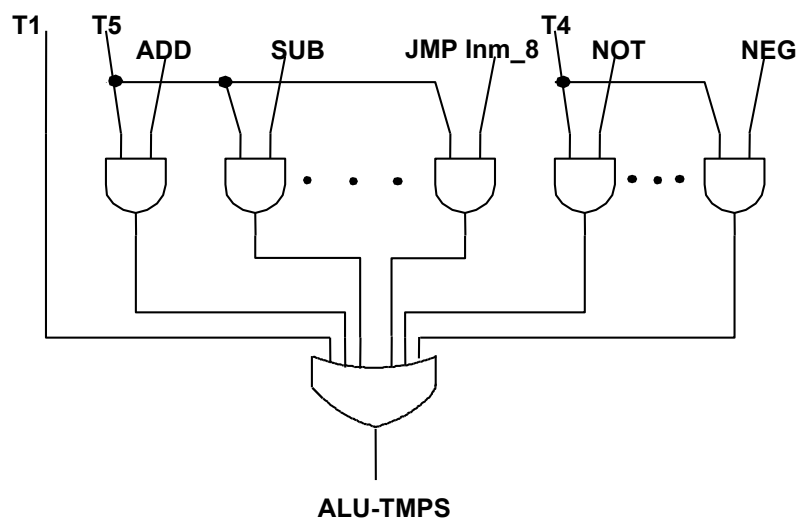


## 4.4- La Unidad de Control (UC): Estructura general



## 4.4- La Unidad de Control: U.C. cableada

**Ejemplo:** generación de la señal ALU-TMPS

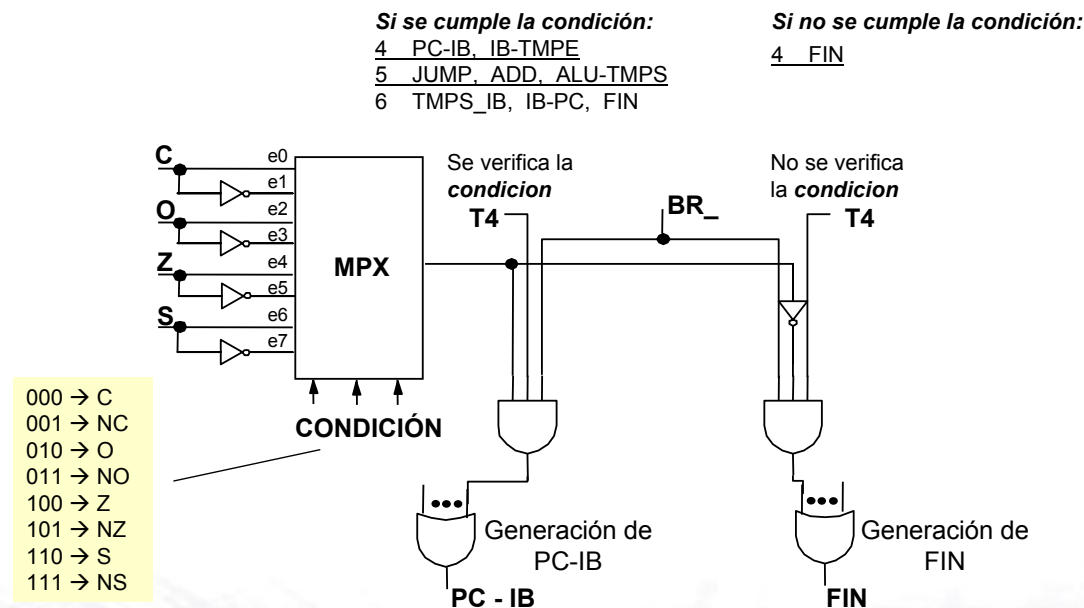


Cada señal de control se genera como resultado de una función lógica cuyas entradas son:

- La instrucción que activa esa señal
- Los pasos en los que se activa la señal
- El valor de los *flags* del registro de estado

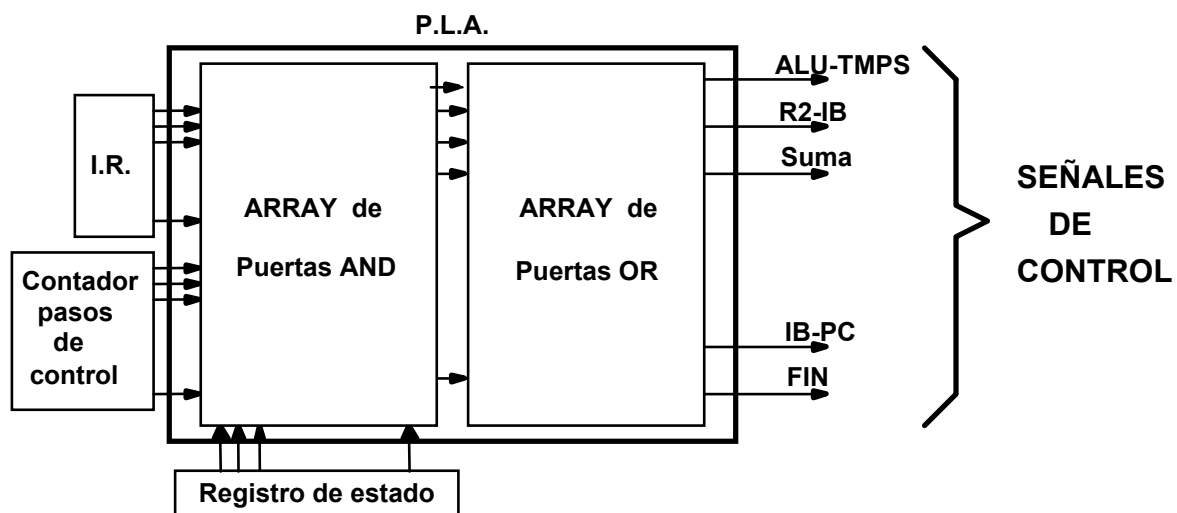
## 4.4- La Unidad de Control: U.C. cableada

Han de tenerse en cuenta en las instrucciones de control de flujo condicional. Se tienen en cuenta de la siguiente forma:



## 4.4- La Unidad de Control: Construcción de una UC cableada

Construcción mediante un PLA:

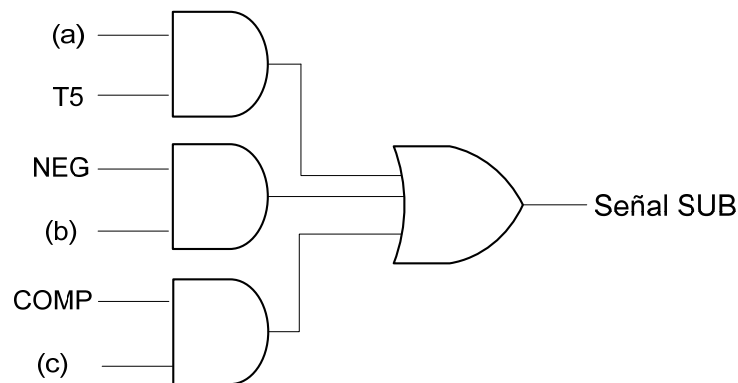


## 4.4- La Unidad de Control: Ejercicio propuesto

### Ejercicio:

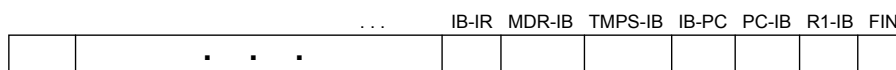
La figura muestra un trozo del circuito de una UC cableada, la cual genera las mismas señales de control que las vistas en el caso de la CPU teórica. ¿Qué mnemónicos o pasos faltan en (a), (b) y (c)?

[ Solución: (a)=SUB, (b)=T4, (c)=T5 ]



## 4.4- La Unidad de Control: Construcción UC microprogramada

- Plantear el nº total de posibles señales de control manejadas por la CPU (en el caso de la CPU teórica existen 61 señales de control diferentes).
- Construir palabras que tengan tantos bits como señales de control existan. A estas palabras se las denomina **PALABRAS DE CONTROL**.



- Cualquier paso de ejecución de cualquier instrucción puede expresarse mediante una palabra de control que tenga a "1" los bits correspondientes a las señales de control que se activan para esa instrucción en ese paso.

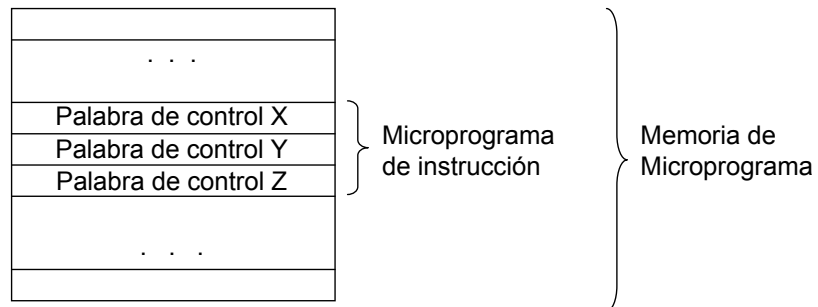
**Ejemplo:**

2		TMPS-IB, IB-PC	→	...	0	0	1	1	0	0	0
3		MDR-IB, IB-IR	→	...	1	1	0	0	0	0	0



## 4.4- La Unidad de Control: Construcción UC microprogramada

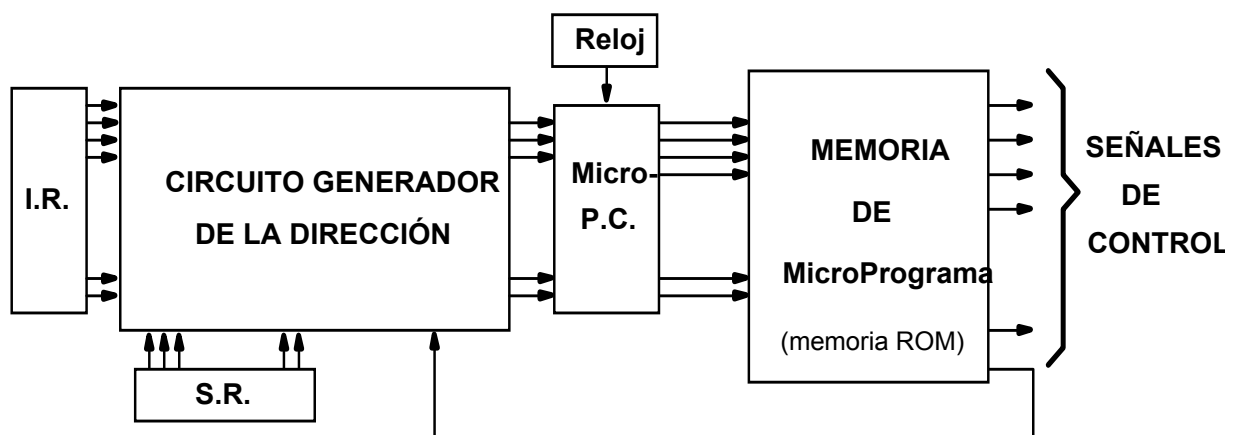
- Una instrucción constará de tantas secuencias (palabras de control) como pasos de ejecución tenga (**microprograma de instrucción**).
- Las palabras de control necesarias para ejecutar todas y cada una de las instrucciones se pueden almacenar en una memoria → **MEMORIA DE MICROPROGRAMA**.



- Para acceder a cada una de las palabras de control almacenadas en la memoria de microprograma se utiliza un **registro  $\mu$ PC**.

## 4.4- La Unidad de Control: Construcción UC microprogramada

¿Cómo se ejecutaría una instrucción?



## 4.4- La Unidad de Control: Comparación

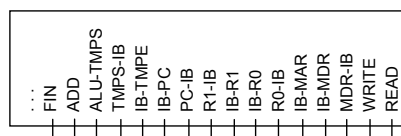
- Las unidades de control cableadas son más rápidas que las microprogramadas
- Pero son más complejas y menos flexibles
- En la actualidad se usan unidades de control híbridas:
  - Para las instrucciones sencillas y frecuentes: cableada
  - Para las instrucciones complejas y poco frecuentes: microprogramada



## 4.4- La Unidad de Control: Ejercicio propuesto

### Ejercicio:

En la siguiente figura se muestra parte de la UC microprograma de una CPU, análoga a la CPU teórica vista en clase.



Se pide:

- ¿Qué palabra de control, expresada en hexadecimal, generará dicha UC en el ciclo 4 de la instrucción **MOV R1, R0**? (Solución: **80A0h**)
- ¿A qué instrucción corresponderá la siguiente secuencia de palabras de control (expresadas en hexadecimal), sabiendo que el resto de señales están a 0?:

**0110**

**802A**

(Solución: **MOV [R1], R0**)



## 4.4- La Unidad de Control: Ejercicios propuestos

- 4 Se ha implementado una nueva instrucción en la CPU teórica, cuyo microprograma es el mostrado a continuación (se han omitido las 3 primeras palabras del microprograma, así como las señales de control que no intervienen en la instrucción).

Nº palabra	ALU-TMPS	ALU-SR	WRITE	XOR	READ	IB-MDR	TMPE-SET	IB-MAR	TMPS-IB	MDR-IB	R0-IB	FIN
4	0	0	0	0	1	0	0	1	0	0	1	0
5	0	0	0	0	0	0	0	0	0	0	0	0
6	1	1	0	1	0	0	1	0	0	1	0	0
7	0	0	1	0	0	1	0	0	1	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	1

¿De qué instrucción se trata? Escribir su mnemónico.

(Solución: NOT [R0])

