

Tema 3

La Unidad Central de Proceso (CPU)

Curso 2007-2008

Objetivos

- Conocer la estructura interna de un procesador y comprender su principio de funcionamiento.
- Conocer un procesador a nivel de programador: juego de instrucciones, registros,... (*Visión externa*).
- Conocer cómo el procesador ejecuta instrucciones a nivel de señales de control. (*Visión interna*).

Esquema

- 1 Conceptos preliminares
- 2 Presentación de la CPU elemental
- 3 Nivel de máquina convencional
- 4 Nivel de micromáquina

Esquema del apartado 1

- 1 Conceptos preliminares
 - Introducción
 - Funcionamiento
 - Parámetros de una CPU
 - Niveles de descripción
- 2 Presentación de la CPU elemental
- 3 Nivel de máquina convencional
- 4 Nivel de micromáquina

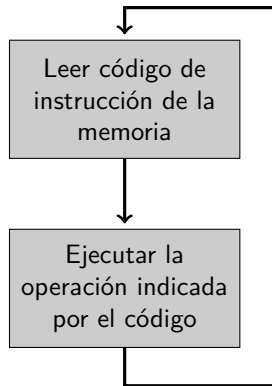
Objetivo de la CPU

Ejecutar instrucciones

Instrucción

- Es un código almacenado en el Sistema de Memoria
- Indica a la CPU que realice una operación:
 - Copiar datos de un lugar a otro
 - Operar con los datos
 - Cambiar el orden de ejecución de las instrucciones

¡Bucle infinito!



Memoria: direcciones y datos

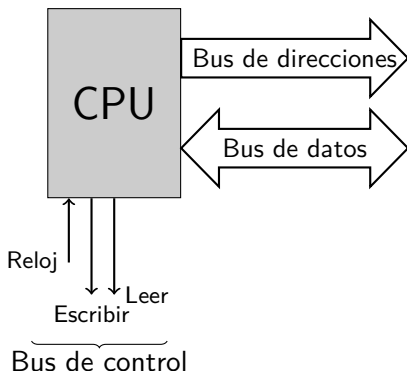
- La memoria consta de una serie de *posiciones*.
- Cada posición se identifica por su *dirección*.
- Cada posición contiene un *dato*.

Las direcciones también son números.

⇒ Se pueden codificar en binario.

Direcc. 0	Dato
Direcc. 1	Dato
Direcc. 2	Dato
Direcc. 3	Dato
	⋮
	Dato
Direcc. $N - 2$	Dato
Direcc. $N - 1$	Dato

Interfaz de la CPU



Lectura de memoria

- Poner dirección en bus direcc.
- Activar "Leer" y esperar respuesta
- Recoger respuesta en bus de datos

Escritura en memoria

- Poner dirección en bus direcc.
- Poner dato en bus de datos
- Activar "Escribir" y esperar

Elementos internos de la CPU

Camino de datos

Es la parte en la que se operan los datos, y se almacenan temporalmente.

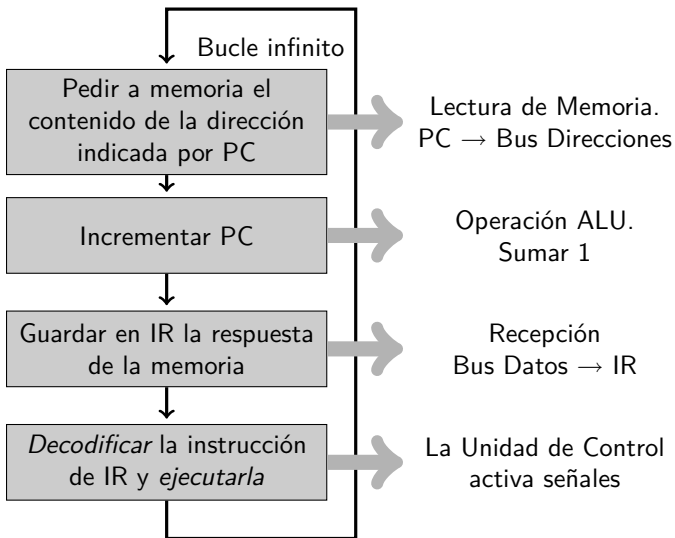
- Registros: Almacenes temporales de datos *y direcciones*.
 - Mínimos imprescindibles: PC e IR.
- Unidad Aritmético Lógica (ALU)
- Bus Interno (IB): Interconecta lo anterior.

Unidad de control

Encargada de *ejecutar* una instrucción dada.

Genera señales que son enviadas a los registros y la ALU. .

Bucle de funcionamiento detallado



Parámetros de una CPU

- **Número de líneas en el bus interno (N_{IB})**
Con cuántos bits opera la ALU.
- **Número de líneas del bus de datos (N_D)**
Cuántos bits puede leer/escribir simultáneamente.
- **Número de líneas en el bus de direcciones (N_A)**
Cuántas direcciones diferentes puede generar
(*Espacio direccionable* $= 2^{N_A}$)
- **Número de bits almacenado en cada dirección de memoria (N_M)**

Ejemplos

Arquitectura	N_{IB}	N_D	N_A	N_M	Esp. Direcc.
z80	8	8	16	8	64 KiB
8086	16	16	20	8	1 MiB
CPU elemental	16	16	16	16	64 KiPal
IA-32	32	32	32	8	4 GiB
AMD64	64	64	40	8	1 TiB

- Es habitual que N_{IB} coincida con N_D . En cambio es habitual que N_M sea 8, con independencia de los otros.
- La “CPU elemental” es particularmente *ortogonal* (todos sus parámetros son iguales).

Una CPU se puede definir a dos niveles:

- **Nivel externo** o *nivel de máquina convencional*
“Visión del programador”:
 - Juego de instrucciones
 - Conjunto de registros
 - Modos de direccionamiento
 - Uso de lo anterior para escribir *algoritmos*
- **Nivel interno** o *nivel de micromáquina*
“Visión del diseñador”:
 - Organización interna
 - Señales de control
 - Unidad de control y microoperaciones

Esquema del apartado 2

- 1 Conceptos preliminares
- 2 Presentación de la CPU elemental**
- 3 Nivel de máquina convencional
- 4 Nivel de micromáquina

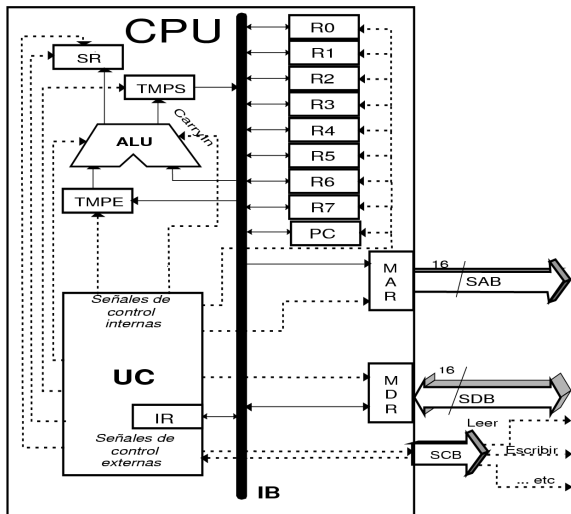
Parámetros de la arquitectura

- CPU de 16 bits (N_{IB} , N_D , N_A).
- Direccionamiento a palabras de 16 bits (N_M)
- Códigos de instrucción de 16 bits

Juego de registros

Nombre	Significado	Uso
R0, ... R7	<i>Register 0, ... 7</i>	Propósito general
SR	<i>Status Register</i>	Bits de estado de la ALU
PC	<i>Program Counter</i>	Dirección de la próxima instrucción
IR	<i>Instruction Register</i>	Código de la instrucción a ejecutar
TMPE	<i>Temporal Entrada</i>	Información temporal de entrada
TMPS	<i>Temporal Salida</i>	... y salida de la ALU
MAR	<i>Mem. Address Register</i>	Interfaz con el bus de direcciones
MDR	<i>Mem. Data Register</i>	Interfaz con el bus de datos

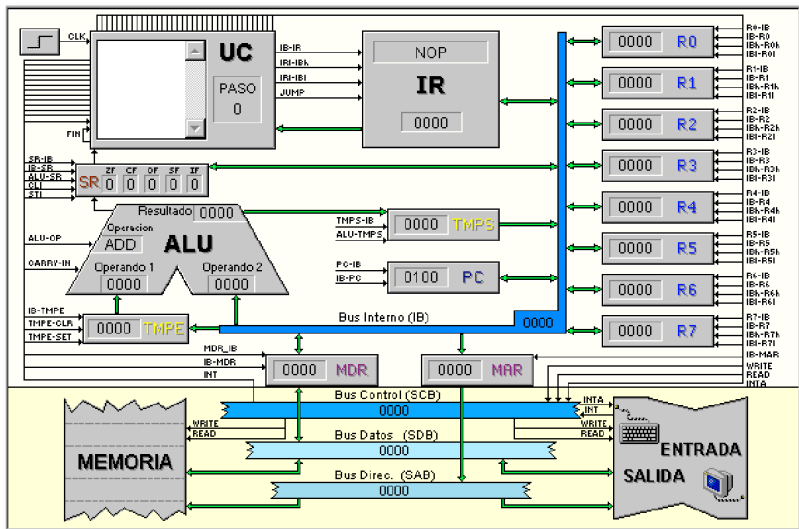
Estructura interna



Buses

- **SAB**
System Address Bus
- **SDB**
System Data Bus
- **SCB**
System Control Bus
- **IB**
Internal Bus

El simulador de la CPU elemental



Las señales de control

R0 <input type="checkbox"/> R0-IB <input type="checkbox"/> IB-R0 <input type="checkbox"/> IBh-R0h <input type="checkbox"/> IBI-R0I	R1 <input type="checkbox"/> R1-IB <input type="checkbox"/> IB-R1 <input type="checkbox"/> IBh-R1h <input type="checkbox"/> IBI-R1I	R2 <input type="checkbox"/> R2-IB <input type="checkbox"/> IB-R2 <input type="checkbox"/> IBh-R2h <input type="checkbox"/> IBI-R2I	R3 <input type="checkbox"/> R3-IB <input type="checkbox"/> IB-R3 <input type="checkbox"/> IBh-R3h <input type="checkbox"/> IBI-R3I	SR <input type="checkbox"/> IB-SR <input type="checkbox"/> SR_IB <input type="checkbox"/> ALU_SR <input type="checkbox"/> CLI <input type="checkbox"/> STI	IR <input type="checkbox"/> IB-IR <input type="checkbox"/> IRI-IBh <input type="checkbox"/> IRI-IBI <input type="checkbox"/> JUMP
R4 <input type="checkbox"/> R4-IB <input type="checkbox"/> IB-R4 <input type="checkbox"/> IBh-R4h <input type="checkbox"/> IBI-R4I	R5 <input type="checkbox"/> R5-IB <input type="checkbox"/> IB-R5 <input type="checkbox"/> IBh-R5h <input type="checkbox"/> IBI-R5I	R6 <input type="checkbox"/> R6-IB <input type="checkbox"/> IB-R6 <input type="checkbox"/> IBh-R6h <input type="checkbox"/> IBI-R6I	R7 <input type="checkbox"/> R7-IB <input type="checkbox"/> IB-R7 <input type="checkbox"/> IBh-R7h <input type="checkbox"/> IBI-R7I	PC <input type="checkbox"/> PC-IB <input type="checkbox"/> IB-PC	ALU <input type="checkbox"/> ADD <input type="checkbox"/> SUB <input type="checkbox"/> OR <input type="checkbox"/> AND <input type="checkbox"/> XOR <input type="checkbox"/> CarryIn
TMPE <input type="checkbox"/> IB-TMPE <input type="checkbox"/> TMPE-SET <input type="checkbox"/> TMPE-CLR	TMPS <input type="checkbox"/> TMPS-IB <input type="checkbox"/> ALU_TMPS	MEMORIA <input type="checkbox"/> WRITE <input type="checkbox"/> READ	E/S <input type="checkbox"/> INTA MAR <input type="checkbox"/> IB-MAR	MDR <input type="checkbox"/> MDR-IB <input type="checkbox"/> IB-MDR	<input type="checkbox"/> FIN

Esquema del apartado 3

- 1 Conceptos preliminares
- 2 Presentación de la CPU elemental
- 3 Nivel de máquina convencional**
 - Registros
 - Modos de direccionamiento
 - Codificación
 - Juego de instrucciones
- 4 Nivel de micromáquina

Características a estudiar

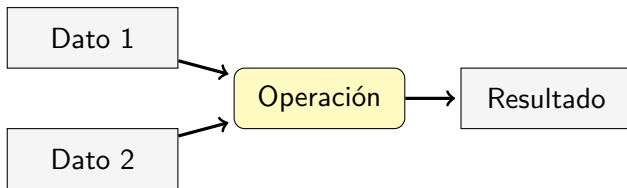
- Qué registros están disponibles para el programador
- Modos de direccionamiento que soporta la CPU
- Juego de instrucciones
- Codificación

Registros accesibles al programador

Registro	Cometido
R0, . . . , R7	Registros de propósito general. Pueden leerse y escribirse, y pueden contener datos o direcciones. R7 tiene un cometido especial (Puntero Pila)
PC	Contador de programa. Indica la dirección de la instrucción a ejecutar. No se puede manipular directamente, sino sólo mediante instrucciones de <i>control de flujo</i>
SR	Registro de estado. Contiene los bits de estado de la última operación realizada por la ALU. Se consulta en las instrucciones de salto condicional.

Concepto de modo de direccionamiento

Las instrucciones realizan una operación sobre uno o más datos y producen un resultado



¿Almacén de los datos?

- Los datos de entrada pueden estar en un registro de la CPU, una posición de la memoria o formar parte de la instrucción.
- El resultado puede ir a otro registro o a la memoria.

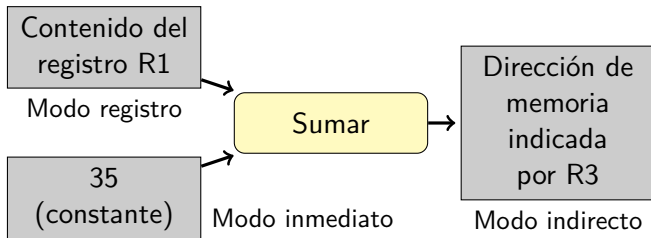
Modos de direccionamiento

Son los mecanismos que tienen las instrucciones para especificar cómo obtener (o donde dejar) el dato (o resultado).

Modos de direccionamiento típicos

- **Modo inmediato.** El dato viene en la propia instrucción
- **Modo registro.** El dato se toma de (o se deja) en un registro.
- **Modo indirecto.** El dato se toma de (o se deja en) memoria. La instrucción especifica de alguna forma la dirección

En teoría es posible combinar varios modos de direccionamiento en la misma instrucción, para sus diferentes datos y destino:



En la práctica, cada instrucción soporta sólo algunos modos de direccionamiento en cada operando.

Modos de direccionamiento en la CPU elemental

Modo inmediato

- El dato se da en la instrucción.
- Sólo soportado en las instrucciones MOVH y MOVL.
- El dato inmediato está limitado a 8 bits.

Ejemplo

MOVL R3, 12

MOVH R3, 149

Modos de direccionamiento en la CPU elemental

Modo indirecto

- El dato está en memoria, y la dirección en un registro que se especifica entre corchetes.
- Sólo soportado en la instrucción MOV (copia de un dato).
- Sólo para uno de los operandos.
- El otro operando debe ser registro.

Ejemplos

MOV R3, [R1]

MOV [R1], R3

Modos de direccionamiento en la CPU elemental

Modo registro

- El dato está en un registro, o se deja en él.
- Es el único modo soportado por las operaciones aritmético-lógicas.
- También soportado por MOV (copia de datos)

Ejemplos

INC **R1**

MOV **R1**, [R3]

MOV **R5**, **R2**

Codificación por campos

- Cada código de instrucción ocupa 16 bits
- Los 16 bits se dividen en grupos.
- Cada grupo tiene diferente significado

Ejemplos

- Los dos primeros bits del código indican el tipo de instrucción:
 - 00: Instrucciones de copia de datos
 - 01: Instrucciones de operación
 - etc.
- Con qué registro opera la instrucción suele especificarse con un campo de 3 bits.

Tablas de codificación

JUEGO DE INSTRUCCIONES DE LA CPU DE EJEMPLO

La siguiente tabla muestra el juego de instrucciones que maneja la CPU del simulador.

Nomenclatura utilizada:

Rd: Tres bits que codifican el registro destino de una operación.

Rc: Tres bits que codifican el registro fuente de una operación.

Rd1: Tres bits que codifican el registro fuente 1 de una operación.

Rd2: Tres bits que codifican el registro fuente 2 de una operación.

Rn: Tres bits que codifican el registro índice para direccionamiento indirecto.

Rd1d5: Tres bits que codifican un registro que es a la vez fuente y destino de una operación.

Rn: Tres bits que codifican el registro que contiene la dirección de destino para saltos indirectos.

Imm_8: Valor numérico de 8 bits.

Instrucciones de movimiento

Código instrucción	Descripción	Mnemónico	Operación
00-000-000000000	Interacción nula.	NOP	—
00-001 Rd Rc 00000	Transferencia a un registro.	MOV Rd, Rc	Rd ← Rc
00-000 Rd Rc 00000	Copiar el contenido de la posición de memoria cuya dirección está en Rc en la posición de memoria cuya dirección está en Rd.	MOV Rd, [Rc]	Rd ← [Rc]
00-011 Rd Rc 00000	Copiar el contenido del registro Rc en la posición de memoria cuya dirección está en Rd.	MOV [Rd], Rc	[Rd] ← Rc
00-100 Rd Imm_8	Copiar en los 8 bits menos significativo de Rd el dato codificado en los 8 bits del campo Imm_8.	RdImm ← Imm_8	
00-101 Rd Imm_8	Copiar en los 8 bits más significativo de Rd el dato codificado en los 8 bits del campo Imm_8.	RdImm ← Imm_8	
00-110 Rc 00000000	Agitar el contenido del registro Rc.	PSH Rc	Pila ← Rc
00-111 Rd 00000000	Descargar un valor en el registro Rd.	POP Rd	Rd ← Pila

Instrucciones Aritmético-Lógicas

De tres operandos	Descripción	Mnemónico	Operación
01-000 Rd Rc Rd 000	Sumar el contenido de los registros Rd1 y Rd2 y almacenar el resultado en Rd.	ADD Rd, Rd1, Rd2	Rd ← Rd1 + Rd2
01-001 Rd Rc Rd 000	Restar el contenido del registro Rd2 al registro Rd1 y almacenar el resultado en Rd.	SUB Rd, Rd1, Rd2	Rd ← Rd1 - Rd2
01-010 Rd Rc Rd 000	Realiza la operación OR con el contenido de los registros Rd1 y Rd2 y almacena el resultado en Rd.	OR Rd, Rd1, Rd2	Rd ← Rd1 OR Rd2
01-011 Rd Rc Rd 000	Realiza la operación AND con el contenido de los registros Rd1 y Rd2 y almacena el resultado en Rd.	AND Rd, Rd1, Rd2	Rd ← Rd1 AND Rd2
01-100 Rd Rc Rd 000	Realiza la operación XOR con el contenido de los registros Rd1 y Rd2 y almacena el resultado en Rd.	XOR Rd, Rd1, Rd2	Rd ← Rd1 XOR Rd2

Operando	Descripción	Mnemónico	Operación
CS:00010	Restar el contenido del registro Rd2 al CS:0001 Rd1, Rd2	CS:0001 Rd1, Rd2	Rd1 - Rd2

Operando	Descripción	Mnemónico	Operación
00000000	Realiza la operación lógica NOT con los bits del registro Rd1.	NOT Rd1	Rd1 ← ~Rd1
00000000	Incrementa el contenido del registro Rd1 en una unidad.	INC Rd1	Rd1 ← Rd1 + 1
00000000	Decrementa el contenido del registro Rd1 en una unidad.	DEC Rd1	Rd1 ← Rd1 - 1
00000000	Cambia de signo complementando a 2's el contenido del registro Rd1.	NEG Rd1	Rd1 ← -Rd1 + 1

Operaciones

Operación	Descripción	Mnemónico	Operación
00000000	Forzar a cero el flag de overflow de interrupción.	CLI	IF ← 0
00000000	Forzar a uno el flag de overflow de interrupción.	STI	IF ← 1
Imm_8	Genera la interrupción software con vector de interrupción Imm_8.	INT Imm_8	Pila ← SR PC ← PC + Imm_8
00000000	Reservar de una interrupción directa un hardware a software.	IRET	PC ← Pila SR ← Pila

Operaciones de control de flujo

Operación	Descripción	Mnemónico	Operación
Imm_8	Realiza un salto relativo.	JMP Imm_8	PC ← PC + Imm_8
00000000	Realiza un salto indirecto absoluto a la posición de memoria codificada en el registro Rc.	JMP [Rc]	PC ← [Rc]

Procedimientos

Operación	Descripción	Mnemónico	Operación
Imm_8	Realiza un salto relativo a un procedimiento.	CALL Imm_8	Pila ← PC PC ← PC + Imm_8
00000000	Realiza un salto indirecto absoluto a un procedimiento.	CALL [Rc]	Pila ← PC PC ← [Rc]

Descripción	Mnemónico	Operación
Realiza un procedimiento indirecto.	SR ← Pila	

Descripción	Mnemónico	Operación
Incrementa con el signo extendido Imm_8 la PC siempre y cuando la PC esté codificada en el campo Cond.	SR, Cond Imm_8 PC ← PC + Imm_8	Si CondImm_8 PC ← PC + Imm_8

bits que codifican la condición de salto de la siguiente forma:

000 → CF = 1
001 → CF = 0
010 → OF = 1
011 → OF = 0
100 → ZF = 1
101 → ZF = 0
110 → SF = 1
111 → SF = 0

Clasificación de las instrucciones

- Movimiento de datos
 - Carga de datos en registros
 - Copia de un registro a otro
 - Copia entre registro y memoria
- Operaciones aritméticas y lógicas
 - Operan siempre entre registros
 - El resultado va a otro registro
- Control de flujo
 - Operan sobre el contador de programa
 - Alteran el flujo secuencial de ejecución
- Otras

Movimiento de datos

Carga de un registro

- Permite cargar una constante en un registro
- La constante usa modo de direccionamiento inmediato
- La constante es de 8 bits

Pero...

Constante de 8 bits, registros de 16 bits... \Rightarrow ¿?

Hay dos instrucciones de carga

- MOVH carga la constante en los 8 bits superiores del registro
- MOVL carga la constante en los 8 bits inferiores del registro

Movimiento de datos: carga directa

R1

0 0 0 0

Ejemplo

```
MOVH R1, 12      ; Carga 12 (0Ch) en parte alta de R1
MOVL R1, 7Ah     ; Carga 122 (7Ah) en parte baja de R1
MOVH R1, 0       ; Carga 0 (00h) en parte alta de R1
```

MOVH y MOVL: Codificación



Ejemplos

Codifica las siguientes instrucciones:

- MOVH R1, 12
- MOVL R5, 21h

Movimiento de datos

Copia entre registros

Se copian los 16 bits de un registro a otro

Sintaxis

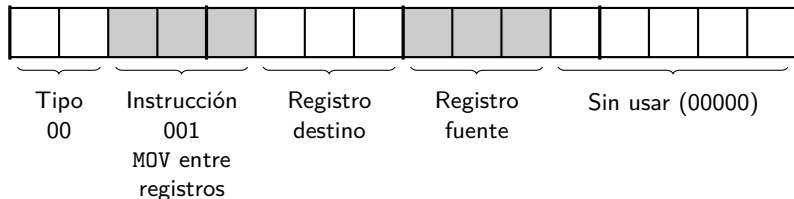
MOV R_d, R_s

R_s Registro fuente (*source*), contiene el dato a copiar

R_d Registro destino, recibe el dato.

Es como una asignación $R_d \leftarrow R_s$

MOV entre registros: Codificación



Ejemplos

Codifica las siguientes instrucciones:

- MOVH R1, R2
- MOVL R3, R7

Movimiento de datos

Copia entre un registro y memoria

Se copian los 16 bits que hay en una posición de memoria a un registro, o viceversa.

La dirección de memoria debe especificarse mediante otro registro

Sintaxis

MOV R_d , $[R_i]$

MOV $[R_i]$, R_s

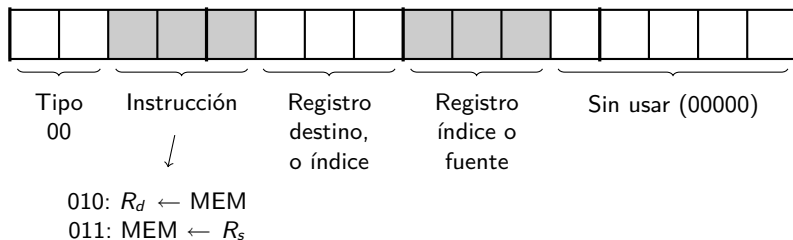
R_s Registro fuente (*source*), contiene el dato a copiar

R_d Registro destino, recibe el dato.

R_i Registro índice, contiene la dirección de memoria en la que se lee o escribe el dato.

Observar que el registro índice se escribe entre corchetes.

MOV entre un registro y memoria: Codificación



Ejemplos

Codifica las siguientes instrucciones:

- `MOVH [R1], R2`
- `MOVL R3, [R7]`

Movimiento de datos

Copia de datos entre registros y la pila

La pila es un almacén temporal al que se pueden enviar datos para recuperarlos más tarde (en orden inverso al envío).

Sintaxis

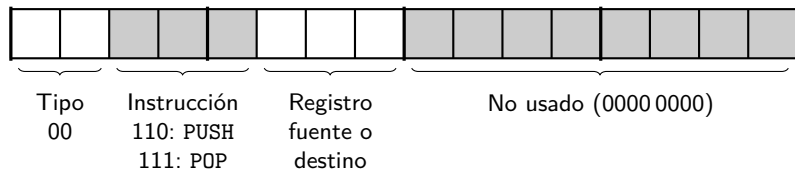
PUSH R_s Envía un dato a la pila Siendo

POP R_d Recupera un dato de la pila

R_s el registro que contiene el dato a enviar

R_d el registro que recibirá el dato recuperado

PUSH y POP: codificación



Ejemplos

Codifica las siguientes instrucciones:

- PUSH R3
- POP R6

Operaciones aritmético-lógicas

- Son operaciones realizadas por la ALU
- Permiten *transformar* los datos, al operar entre ellos
- Además del resultado producen unos bits de estado que informan de ciertas condiciones sobre el resultado obtenido

Bits de estado

Bit	Nombre	Cuándo se pone a 1
ZF	Bit Zero	Cuando el resultado sale cero
SF	Bit de Signo	Cuando el resultado es negativo
CF	Bit de Carry	Cuando al sumar los datos se produce acarreo en la última columna
OF	Bit de Overflow	Cuando hay contradicción de signo entre los operandos y la operación solicitada

Operaciones lógicas

De tres operandos

Realizan una operación lógica bit a bit entre dos registros, dejando el resultado en un tercero.

Sintaxis

OP R_d , R_{s1} , R_{s2}

Siendo:

R_{s1} , R_{s2} Registros fuente, contienen los datos a operar

R_d Registro destino, almacenará el resultado.

OP Operación a realizar (AND, OR o XOR).

Es decir: $R_d = \text{OP}(R_{s1}, R_{s2})$

Aunque los bits CF y OF pueden activarse, no significan nada.

Operaciones lógicas

Ejemplo

```
AND  R3, R2, R1      ; R3 ← R2 AND R1
OR   R3, R2, R1      ; R3 ← R2 OR R1
XOR  R1, R2, R3      ; R1 ← R2 XOR R3
```

~~Result~~ Resultado Final:

R1

A 7 F 2

R2

0 3 0 1

R3

9 9 B B

Operaciones lógicas

De un solo operando

El único operando suministrado proporciona el dato a operar y es a la vez el lugar donde se almacenará el resultado.

Sintaxis

Sólo hay una instrucción en esta categoría: NOT, que realiza la negación lógica de cada bit de un registro.

NOT $R_{d/s}$

Donde $R_{d/s}$ es el registro que actúa a la vez como fuente y destino.

Ejemplo

¿Cuál será el valor de R3 tras ejecutar el siguiente código?

MOVH R3, 10

MOVL R3, 89h

NOT R3

Operaciones aritméticas

De tres operandos

Suman o restan dos registros, dejando el resultado en un tercero.

Sintaxis

OP R_d , R_{s1} , R_{s2}

Siendo:

R_{s1} , R_{s2} Registros fuente, contienen los datos a operar

R_d Registro destino, almacenará el resultado.

OP Operación a realizar (ADD o SUB)

El bit CF sólo tiene sentido si los datos son naturales, y el OF sólo si son enteros en C-2.

Operaciones aritméticas

El bit de préstamo

Cuando la operación solicitada es SUB, el bit de estado *carry* contiene el llamado *bit de préstamo*.

Ejemplos

Si inicialmente R1=03F2, R2=FFF2, R3=FFFF, responder, qué resultado recibiría R5 y qué valores toman los bits de estado en cada uno de los casos siguientes:

- ADD R5, R1, R2
- SUB R5, R2, R3
- SUB R5, R1, R1
- ADD R5, R2, R3

Operaciones aritméticas

De dos operandos

Realizan una operación entre dos datos, y desechan el resultado (pero actualizan los bits de estado)

Sintaxis

Sólo hay una instrucción en esta categoría:

COMP R_{s1}, R_{s2}

Que realiza la resta $R_{s1} - R_{s2}$, desechando el resultado.

Permite comparar dos registros

Operaciones aritméticas

De un solo operando

El único operando suministrado proporciona el dato a operar y es a la vez el lugar donde se almacenará el resultado.

Sintaxis

OP $R_{d/s}$

Donde $R_{d/s}$ es el registro que actúa a la vez como fuente y destino, y OP es una de las siguientes:

- INC: Incrementa el registro
- DEC: Decrementa el registro
- NEG: Calcula el C-2 del registro

Codificación de las instrucciones aritmético-lógicas

No ofrece mayor dificultad. Se trata de aplicar directamente la tabla de codificación.

Ejemplos

Codificar todas las instrucciones aritmético lógicas aparecidas en los ejemplos anteriores.

AND R3, R2, R1	
OR R3, R2, R1	
XOR R1, R2, R3	
NOT R3	
ADD R5, R1, R2	
SUB R5, R2, R3	
SUB R5, R1, R1	
ADD R5, R2, R3	

Instrucciones de control de flujo

Concepto clave

Son instrucciones que modifican el contador de programa (PC)

Existen dos tipos:

- **Con condición** (saltos condicionales): sólo modifican PC si se cumple una condición
- **Sin condición** (saltos incondicionales): modifican siempre PC

Existe además un salto especial que permite volver más adelante al punto desde el cual se saltó. Se denomina “llamada” y “retorno”, y se explicará en otro tema.

Salto incondicional

Concepto

Cuando esta instrucción se ejecuta, se le suma al PC una cantidad indicada en la instrucción.

Sintaxis

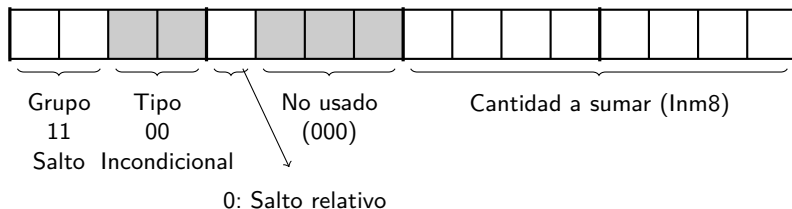
`JMP Inm8`

Siendo *Inm8* una constante de 8 bits que codifica la cantidad a sumar a PC.

Extensión de signo

Ya que la cantidad a sumar va codificada en 8 bits, pero PC tiene 16, antes de hacer la suma, la CPU tiene que “extender” el dato de 8 bits hasta que ocupe 16. Esto lo hace replicando el bit de signo hacia la izquierda.

Salto incondicional: Codificación



Ejemplos

- JMP +12
- JMP -9

Salto incondicional: funcionamiento

Importante

PC se incrementa mientras la CPU espera el código de instrucción.

⇒ Cuando una instrucción se está ejecutando, PC apunta a la siguiente.

Ejemplo

Dada la situación del PC y de la memoria mostrada en la figura, ¿Cuál será la próxima instrucción que se ejecutará? ¿Y la siguiente?

PC 02BB

	⋮	...
02B6		2200
02B7		2A00
02B8		21FF
02B9		29FF
02BA		4F44
02BB		C0FC
02BC		0940
	⋮	...

Salto condicional

Concepto

Cuando esta instrucción se ejecuta pueden ocurrir dos cosas, según se cumpla o no la *condición*:

- Que se sume al PC una cantidad indicada en la instrucción.
- Que no se le sume

La *condición* depende del valor que en ese momento tenga un bit de estado.

Sintaxis

`BRX Inm8`

Donde *X* es una letra o dos que indica la *condición* e *Inm8* es la cantidad a sumar a PC si se cumple (como en JMP)

Salto condicionales: condiciones

Hay cuatro bits de estado. Cada uno puede tener dos valores. Esto hace un total de ocho posibles condiciones.

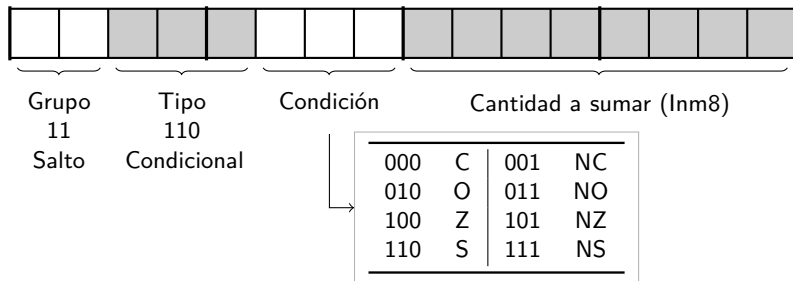
Condición	Salta si	Condición	Salta si
C	CF=1	NC	CF=0
O	OF=1	NO	OF=0
Z	ZF=1	NZ	ZF=0
S	SF=1	NS	SF=0

Ejemplos

BRZ +3 : Sumará 3 a PC si ZF=1

BRNS -9 : Sumará -9 a PC si SF=0

Saltos condicionales: codificación



Ejemplos

BRZ +7	
BRNC -12	
BRO 200	

Esquema del apartado 4

- 1 Conceptos preliminares
- 2 Presentación de la CPU elemental
- 3 Nivel de máquina convencional
- 4 Nivel de micromáquina
 - Registros internos y transferencia entre registros
 - La Unidad Aritmético-Lógica
 - Señales de control
 - Pasos de ejecución de instrucciones
 - Ejemplos de instrucciones paso a paso
 - La Unidad de Control

Registros no accesibles por el programador

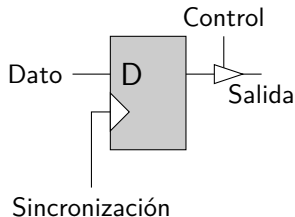
Existen una serie de registros que no aparecen en la sintaxis de las instrucciones, y por tanto no se pueden manipular directamente. Son usados internamente por las instrucciones.

Registros no accesibles

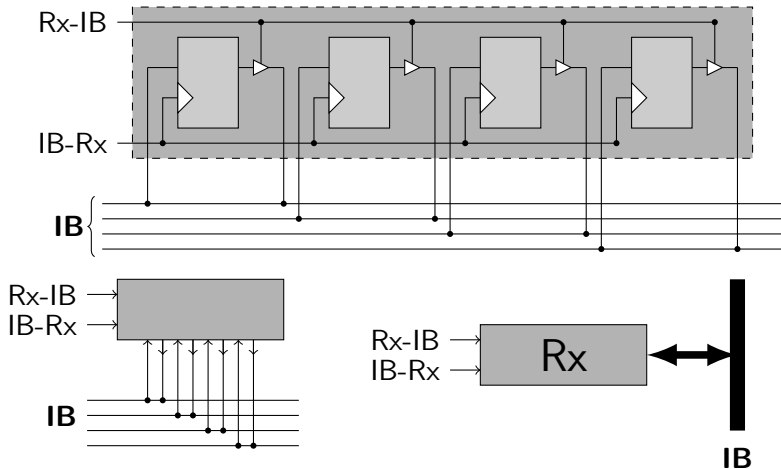
Nombre	Uso
IR	Contiene el código de la instrucción a ejecutar. En ocasiones, como parte del código hay un dato
TMPE	Mantiene temporalmente uno de los datos con los que operará la ALU
TMPS	Almacena temporalmente el resultado generado por la ALU
MAR	Conectado para salida con el bus de direcciones del sistema
MDR	Conectado para salida o entrada con el bus de datos del sistema

Composición de un registro

- Un registro es un grupo de biestables D.
- Un biestable D es una memoria mínima, capaz de almacenar un bit
- Entradas y salidas del biestable:
 - Entrada de datos
 - Entrada de sincronización
 - Salida de datos
- A la salida se pone un *buffer triestado*, que permite conectarla y desconectarla

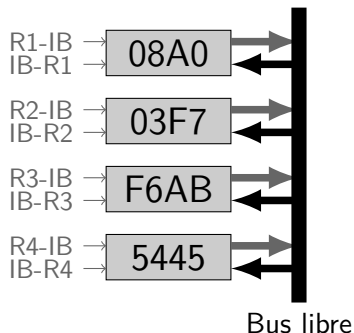


Estructura de un registro conectado a un bus



Transferencia entre registros

- 1 Activar la señal R1-IB
- 2 Activar la señal IB-R2
- 3 Desactivar ambas señales



Ciclo de ocupación del bus

Mientras esté activa una señal Registro-IB, no puede activarse otra de este tipo.

Generación de señales desde la Unidad de Control

Implementación

- La unidad de control recibe una señal de reloj.
- Cada vez que esta señal cambia de 0 a 1, tiene lugar un ciclo de ocupación del bus.
- La UC activa un conjunto de señales en ese ciclo.
- Cuando el ciclo finaliza, las desactiva todas.

Señal de reloj



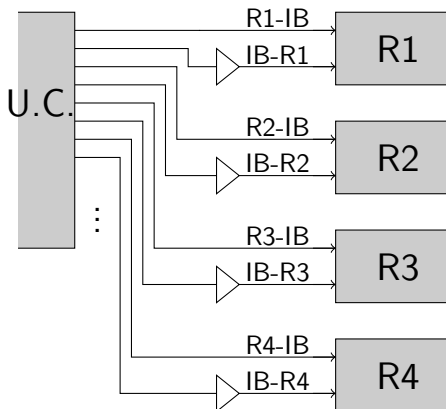
Ejemplo

Copia de R1 al R2: se requiere un único ciclo

Ciclo	Señales activas
1	R1-IB, IB-R2

Generación de señales desde la Unidad de Control

- Para simplificar el diseño de la UC, se hace que genere simultáneamente todas las señales del ciclo.
- En las señales tipo IB-Registro se inserta un retardo, para que lleguen al registro en el orden correcto.



La Unidad Aritmético-Lógica

Cometido

Recibe:

- Dos datos de entrada (de 16 bits cada uno)
- Unas señales que indican una operación a realizar

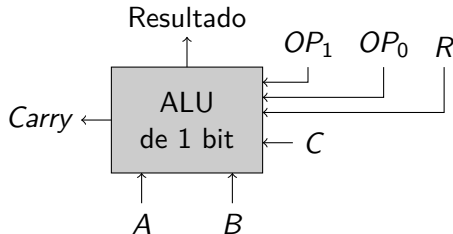
Produce:

- Un resultado (de 16 bits)
- Una serie de *bits de estado* (ZCOS)

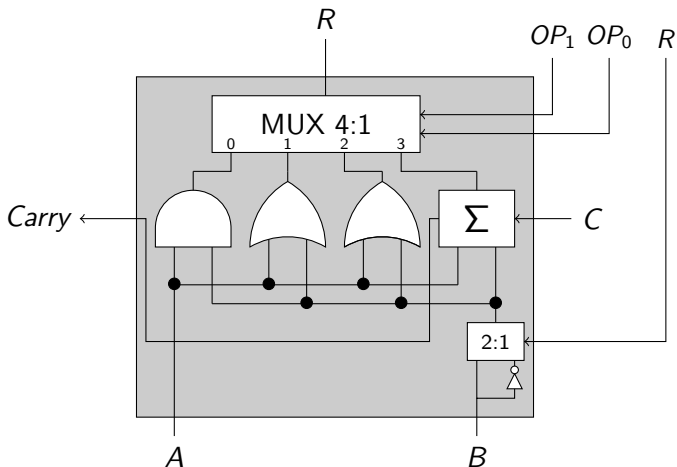
Las señales que gobiernan la operación a realizar se comprenderán mejor tras estudiar la estructura interna de la ALU.

Estructura interna de la ALU

- La ALU de 16 bits se compone de 16 ALUs de 1 bit.
- Cada ALU de 1 bit recibe 3 datos de 1 bit cada uno (A , B y C), y 3 señales de control (OP_1 , OP_0 y R). Produce como resultado 1 bit y un *carry*
- Las señales OP_1 y OP_0 seleccionan una de cuatro posibles operaciones (AND, OR, XOR, ADD). La señal R selecciona si el dato B debe ir negado o no.

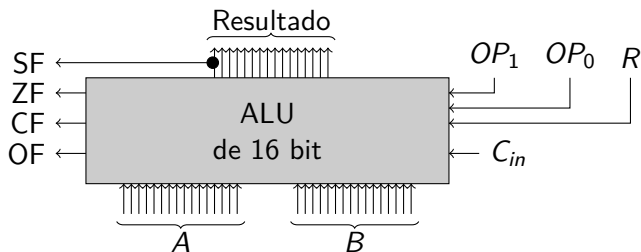


Estructura interna de la ALU

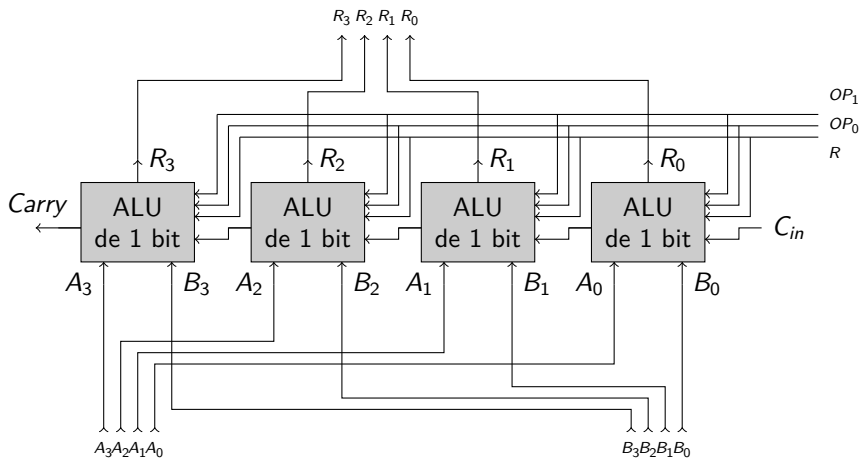


Estructura interna de la ALU

- 16 ALUs de 1 bit se conectan para crear la ALU de 16 bits.
- Cada una recibe un bit de A y un bit de B , y produce un bit del resultado.
- Todas reciben las mismas señales OP_1 , OP_0 y R , por lo que todas hacen la misma operación.
- El Carry de salida de una se conecta a la entrada C de la siguiente, excepto en la primera y la última.



Ejemplo de conexión para una ALU de 4 bits



Señales y operaciones en la ALU

OP_1	OP_0	R	Operación realizada	Comentario
0	0	0	$A \text{ AND } B$	AND
0	0	1	$A \text{ AND } \neg B$	No usada
0	1	0	$A \text{ OR } B$	OR
0	1	1	$A \text{ OR } \neg B$	No usada
1	0	0	$A \text{ XOR } B$	XOR
1	0	1	$A \text{ XOR } \neg B$	No usada
1	1	0	$A + B + C$	ADD
1	1	1	$A + (\neg B) + C$	SUB si $C = 1$ No usada si $C = 0$

Señales de control

- Hemos visto cómo la activación en secuencia de algunas señales causa que ocurran “cosas” en la CPU:
 - Las señales tipo **Registro-IB** transmiten un dato de un registro al bus IB y por tanto a todas partes del *camino de datos*
 - Las señales tipo **IB-Registro** permiten a un registro capturar lo que hay en el bus IB
 - Las señales de la ALU le indican qué operación realizar.
- Hay muchas más señales de control
- Seguidamente las veremos todas, y cómo se secuencian para ejecutar cada instrucción.

Todas las señales de control

R0 <input type="checkbox"/> R0-IB <input type="checkbox"/> IB-R0 <input type="checkbox"/> IBh-R0h <input type="checkbox"/> IBI-R0l	R1 <input type="checkbox"/> R1-IB <input type="checkbox"/> IB-R1 <input type="checkbox"/> IBh-R1h <input type="checkbox"/> IBI-R1l	R2 <input type="checkbox"/> R2-IB <input type="checkbox"/> IB-R2 <input type="checkbox"/> IBh-R2h <input type="checkbox"/> IBI-R2l	R3 <input type="checkbox"/> R3-IB <input type="checkbox"/> IB-R3 <input type="checkbox"/> IBh-R3h <input type="checkbox"/> IBI-R3l	SR <input type="checkbox"/> IB-SR <input type="checkbox"/> SR_IB <input type="checkbox"/> ALU_SR <input type="checkbox"/> CLI <input type="checkbox"/> STI	IR <input type="checkbox"/> IB-IR <input type="checkbox"/> IRI-IBh <input type="checkbox"/> IRI-IBl <input type="checkbox"/> JUMP
R4 <input type="checkbox"/> R4-IB <input type="checkbox"/> IB-R4 <input type="checkbox"/> IBh-R4h <input type="checkbox"/> IBI-R4l	R5 <input type="checkbox"/> R5-IB <input type="checkbox"/> IB-R5 <input type="checkbox"/> IBh-R5h <input type="checkbox"/> IBI-R5l	R6 <input type="checkbox"/> R6-IB <input type="checkbox"/> IB-R6 <input type="checkbox"/> IBh-R6h <input type="checkbox"/> IBI-R6l	R7 <input type="checkbox"/> R7-IB <input type="checkbox"/> IB-R7 <input type="checkbox"/> IBh-R7h <input type="checkbox"/> IBI-R7l	PC <input type="checkbox"/> PC-IB <input type="checkbox"/> IB-PC	ALU <input type="checkbox"/> ADD <input type="checkbox"/> SUB <input type="checkbox"/> OR <input type="checkbox"/> AND <input type="checkbox"/> XOR <input type="checkbox"/> CarryIn
TMPE <input type="checkbox"/> IB-TMPE <input type="checkbox"/> TMPE-SET <input type="checkbox"/> TMPE-CLR	TMPS <input type="checkbox"/> TMPS-IB <input type="checkbox"/> ALU_TMPS	MEMORIA <input type="checkbox"/> WRITE <input type="checkbox"/> READ	E/S <input type="checkbox"/> INTA MAR <input type="checkbox"/> IB-MAR	MDR <input type="checkbox"/> MDR-IB <input type="checkbox"/> IB-MDR	<input type="checkbox"/> FIN

Señales de volcar registro a bus

Cada registro tiene una señal de tipo Rx-IB

Excepto:

- El registro TMPE, que va directamente conectado a la ALU
- El registro MAR, que sólo lee de IB y vuelca en el bus del sistema.
- El registro IR, que no tiene una señal genérica para volcar al bus, sino tres especiales.

Recordatorio

En cada ciclo de ocupación del bus (ciclo de reloj) sólo una de estas señales puede estar activa.

Señales de cargar registro desde el bus

Cada registro tiene una señal de tipo IB-Rx

Excepto:

- El registro TMPE, que no carga del bus, sino de la ALU.

Además

- Todos los registros de propósito general tienen dos señales extra para cargar del bus:
 - IBl-Rxl, carga sólo la parte baja del registro
 - IBh-Rxh, carga sólo la parte alta del registro
- El registro SR puede cargar de la ALU, además de IB.

Otras señales de control

Señales para la ALU

- La ALU recibe señales que indican qué operación realizar.
- Además recibe una señal **CarryIn**, que se suma al resultado en la operación ADD.

Señales para el registro TMPE

Este registro es especial. Además de poder cargarse desde el bus (IB-TMPE), puede recibir valores predeterminados:

- **TMPE-CLR** Pone todos sus bits a cero
- **TMPE-SET** Pone todos sus bits a uno

Otras señales de control

Señales para el registro SR

- **CLI** Pone un 0 en el bit IF
- **STI** Pone un 1 en el bit IF

Líneas de control que salen de la CPU

Las señales **INTA**, **READ** y **WRITE** activan las líneas del mismo nombre que salen de la CPU al bus de control del sistema.

La señal FIN

- Indica que la instrucción ha terminado y debe traerse otra.
- Es recibida por la Unidad de Control

Pasos de ejecución

Concepto

- El “paso” de ejecución es el conjunto de señales que la Unidad de Control activa en un ciclo de reloj.
- Es por tanto un ciclo de ocupación de IB

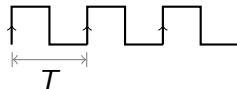
Recordatorios

- Las señales de un paso llegan en el orden adecuado a sus destinos, gracias a los bloques de retardo.
- Escribiremos sólo qué señales se activan en cada paso, sin prestar atención al orden en que llegarán
- Nunca se pueden poner dos señales de tipo Rx-IB en el mismo paso.

La señal de reloj

Misión

La señal de reloj marca el instante en que debe comenzar un nuevo paso.



Características

- Periodo T : distancia entre dos flancos de subida
Se mide en segundos o sus múltiplos (ms, μ s, etc.)
- Frecuencia f : número de ciclos por segundo. Se mide en Hertzios o sus múltiplos.
- $f = 1/T$, ej: si $T = 0,2\mu$ s, entonces $f = 5$ MHz.

La velocidad del reloj afecta directamente a la velocidad de ejecución de las instrucciones.

Ciclo de espera

Concepto

Es un paso en el que la Unidad de Control no genera señales.

Cuándo se usan

Cuando se debe esperar a que la memoria complete una operación, ya sea de **lectura** o de **escritura**.

¿Cuántos ciclos esperar?

- Depende de la velocidad de la memoria.
- En la CPU elemental asumiremos que tanto la lectura como la escritura requieren 1 ciclo de espera.
 - Es decir, si se hace una operación con la memoria en el ciclo N , ésta finalizará al final del ciclo $N + 1$, y por tanto el resultado de la memoria podrá usarse en el ciclo $N + 2$.

Pasos iniciales: traer código e incrementar PC

Los primeros pasos que la UC genera, tienen por objetivo:

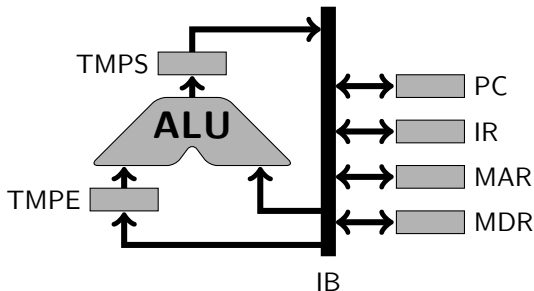
- ① Solicitar a la memoria el código apuntado por PC
- ② Incrementar PC
- ③ Recoger la respuesta de la memoria y almacenarla en IR

Los restantes pasos que generará, dependen del código que se haya recibido.

Cómo incrementar PC

- Se trata de pedirle a la ALU que sume 1
- Introducimos PC en una entrada, pero ¿Cómo introducir un 0001 en la otra?

Pasos iniciales: traer código e incrementar PC



Paso	Señales activas
1	PC-IB, IB-MAR, Leer TMPE-CLR, CarryIn, ADD, ALU-TMPS
2	TMPS-IB, IB-PC
3	MDR-IB, IB-IR

NOP

Paso	Señales activas
1	PC-IB, IB-MAR, Leer, TMPE-CLR, CarryIn, ADD, ALU-TMPS
2	TMPS-IB, IB-PC
3	MDR-IB, IB-IR → IR=0000
4	
5	
6	
...	

AND R2, R3, R1

Paso	Señales activas
1	PC-IB, IB-MAR, Leer, TMPE-CLR, CarryIn, ADD, ALU-TMPS
2	TMPS-IB, IB-PC
3	MDR-IB, IB-IR
4	
5	
6	
...	

→ IR=5A64

SUB R2, R3, R1

Paso	Señales activas
1	PC-IB, IB-MAR, Leer, TMPE-CLR, CarryIn, ADD, ALU-TMPS
2	TMPS-IB, IB-PC
3	MDR-IB, IB-IR → IR=4264
4	
5	
6	
...	

COMP R4, R0

Paso	Señales activas
1	PC-IB, IB-MAR, Leer, TMPE-CLR, CarryIn, ADD, ALU-TMPS
2	TMPS-IB, IB-PC
3	MDR-IB, IB-IR → IR=6C00
4	
5	
6	
...	

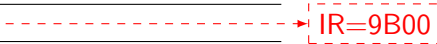
NOT R5

Paso	Señales activas
1	PC-IB, IB-MAR, Leer, TMPE-CLR, CarryIn, ADD, ALU-TMPS
2	TMPS-IB, IB-PC
3	MDR-IB, IB-IR
4	
5	
6	
...	

→ IR=8500

NEG R3

Paso	Señales activas
1	PC-IB, IB-MAR, Leer, TMPE-CLR, CarryIn, ADD, ALU-TMPS
2	TMPS-IB, IB-PC
3	MDR-IB, IB-IR
4	
5	
6	
...	

IR=9B00

INC R2

Paso	Señales activas
1	PC-IB, IB-MAR, Leer, TMPE-CLR, CarryIn, ADD, ALU-TMPS
2	TMPS-IB, IB-PC
3	MDR-IB, IB-IR
4	
5	
6	
...	

→ IR=8A00

DEC R1

Paso	Señales activas
1	PC-IB, IB-MAR, Leer, TMPE-CLR, CarryIn, ADD, ALU-TMPS
2	TMPS-IB, IB-PC
3	MDR-IB, IB-IR → IR=9100
4	
5	
6	
...	

MOV R1, R0

Paso	Señales activas
1	PC-IB, IB-MAR, Leer, TMPE-CLR, CarryIn, ADD, ALU-TMPS
2	TMPS-IB, IB-PC
3	MDR-IB, IB-IR → IR=0900
4	
5	
6	
...	

MOV R1, [R0]

Paso	Señales activas
1	PC-IB, IB-MAR, Leer, TMPE-CLR, CarryIn, ADD, ALU-TMPS
2	TMPS-IB, IB-PC
3	MDR-IB, IB-IR → IR=1100
4	
5	
6	
...	

MOV [R1], R0

Paso	Señales activas
1	PC-IB, IB-MAR, Leer, TMPE-CLR, CarryIn, ADD, ALU-TMPS
2	TMPS-IB, IB-PC
3	MDR-IB, IB-IR → IR=1900
4	
5	
6	
...	

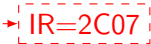
MOVL R4, 3Ch

Paso	Señales activas
1	PC-IB, IB-MAR, Leer, TMPE-CLR, CarryIn, ADD, ALU-TMPS
2	TMPS-IB, IB-PC
3	MDR-IB, IB-IR
4	
5	
6	
...	

→ IR=243C

MOVH R4, 7

Paso	Señales activas
1	PC-IB, IB-MAR, Leer, TMPE-CLR, CarryIn, ADD, ALU-TMPS
2	TMPS-IB, IB-PC
3	MDR-IB, IB-IR
4	
5	
6	
...	



IR=2C07

JMP +5

Paso	Señales activas
1	PC-IB, IB-MAR, Leer, TMPE-CLR, CarryIn, ADD, ALU-TMPS
2	TMPS-IB, IB-PC
3	MDR-IB, IB-IR
4	
5	
6	
...	

→ IR=C005

JMP -7

Paso	Señales activas
1	PC-IB, IB-MAR, Leer, TMPE-CLR, CarryIn, ADD, ALU-TMPS
2	TMPS-IB, IB-PC
3	MDR-IB, IB-IR
4	
5	
6	
...	

→ IR=C0F9

BRNS +5

Paso	Señales activas
1	PC-IB, IB-MAR, Leer, TMPE-CLR, CarryIn, ADD, ALU-TMPS
2	TMPS-IB, IB-PC
3	MDR-IB, IB-IR → IR=F705
4	
5	
6	
...	

Los pasos dependen de SF

- Si SF=0, como en JMP
- Si SF=1, como en NOP

(Continuará...)