Tema 2

INSTRUCCIONES DE UN COMPUTADOR

ÍNDICE

- Definiciones
- Formatos de instrucción
- Modos de direccionamiento
- Tipos de instrucciones
- Diagrama de flujo
- Lenguaje del computador

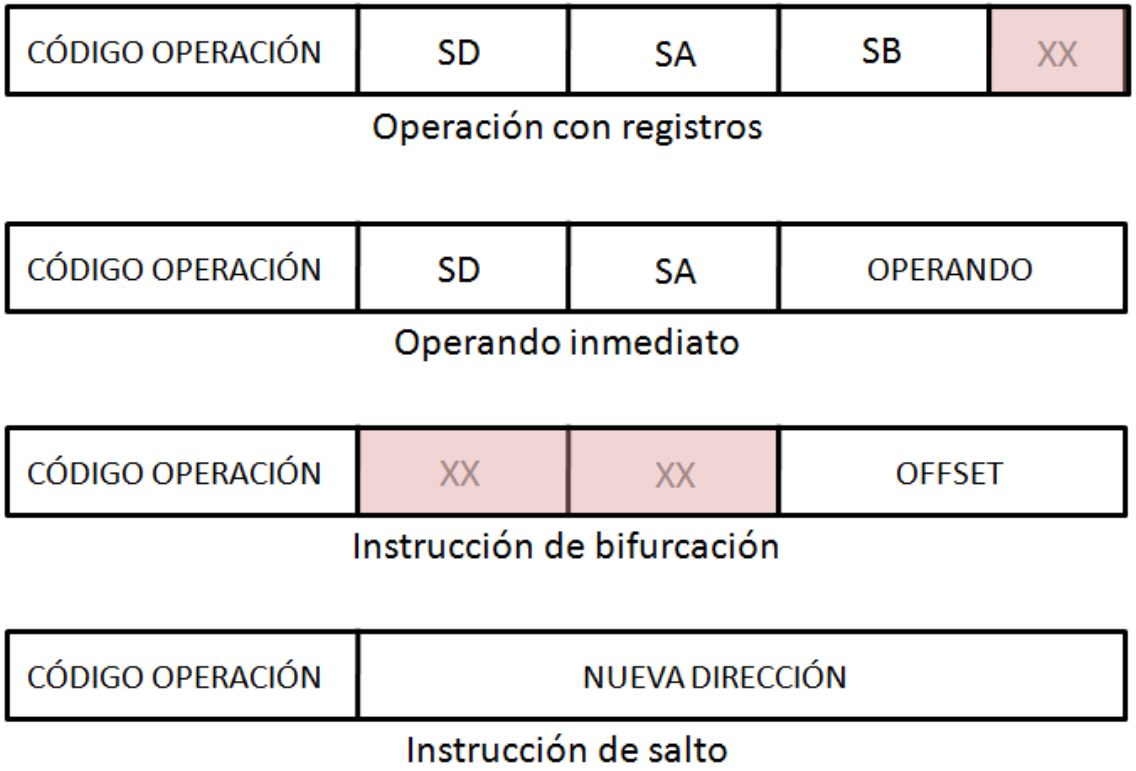
Definiciones: conceptos de arquitectura computacional

- *Lenguaje máquina*: lenguaje binario para escribir las instrucciones y almacenarlas en memoria.
- *Lenguaje ensamblador (lenguaje de bajo nivel)*: lenguaje simbólico que sustituye los códigos binarios de instrucciones y direcciones por nombres simbólicos.
- *Arquitectura:* la forman la *Arquitectura del conjunto de instrucciones* (ISA, nombre simbólico y código binario de la instrucción), organización y el hardware.
- *Dirección efectiva:* dirección de memoria en la que se encuentra el operando.

Definiciones: estructura de la instrucción

- Un computador tiene (generalmente) :
  - Un juego de instrucciones
  - Algunos formatos de instrucción
- *Formato de instrucción* = es la forma en que se representan los bits del código binario de la instrucción, y se separa en campos. que se han de interpretar según lo que representen. Por ejemplo:
  - Campo de código de instrucción
  - Campo de dirección
  - Campo de modo
- En este tema nos interesa el campo de modo ya que la combinación de bits en dicho campo nos indica cómo debemos interpretar el campo de dirección.

Formatos de instrucción



En nuestro ejemplo del tema anterior veíamos estos formatos de instrucción. Es un ejemplo sencillo que no tiene campo de modo.

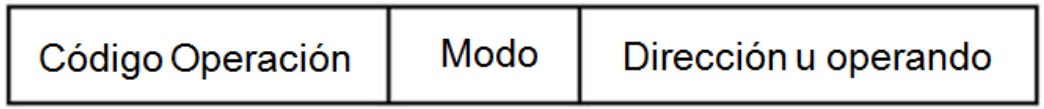
Ciclo de ejecución básico de un computador

Recordando el ciclo de ejecución básico de una instrucción:

1. Instrucciones de memoria → a un registro de la U.C.
2. Decodificar instrucción → operación y modo de direccionamiento
3. Buscar operandos (campo de dirección y modo) → dirección efectiva
4. Recuperar operandos
5. Ejecutar operación (en la ruta de datos)
6. Almacenar resultado
7. Volver al paso 1 a por la siguiente instrucción (PC)

Formato de instrucción: arquitectura de direccionamiento

A la hora de definir el conjunto de instrucciones y el formato de las mismas se puede pensar en instrucciones de diferente número de direcciones explícitas.



ADD R0, R1, R2	R0 ← R1 + R2	R0 R1 R2	3 direcciones
ADD R1, R2	R1 ← R1 + R2	R1 R2	2 direcciones
ADD R2	A ← A + R2	R2	1 dirección
PUSH	M[SP] ← R2 SP ← SP+1		0 direcciones

Formato de instrucción: arquitectura de direccionamiento

- Para una misma operación, el código resultante es más compacto cuanto más campos de dirección tenga. Veamos un ejemplo con 4 datos (D1 a D4) que están en memoria: R0=(D1+D2) ·(D3+D4)

Con 3 direcciones:

```
R1 ← M[D1] + M[D2]
R2 ← M[D3] + M[D4]
R0← R1 × R2
```

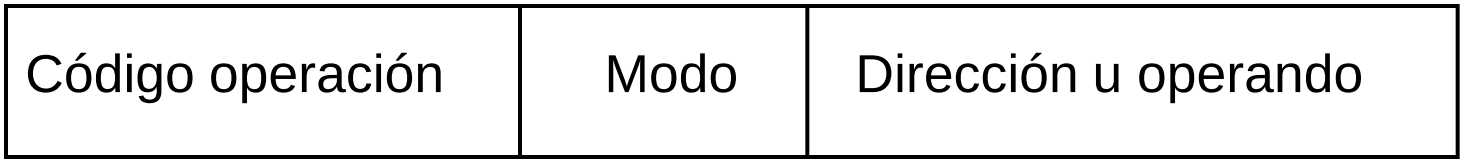
Con 2 direcciones:

```
R1 ← M[D1]
R1 ← R1 + M[D2]
R0 ← M[D3]
R0 ← R0 + M[D4]
R0 ← R0 × R1
```

Con 1 dirección:

```
ACC ← M[D1]
ACC ← ACC + M[D2]
T1 ← ACC
ACC ← M[D3]
ACC ← ACC + M[D4]
ACC ← ACC × M[T1]
R0 ← ACC
```

Formato de instrucción: arquitectura de direccionamiento



Formato de instrucción

Diferentes modos de direccionamiento:

✓ - ofrece flexibilidad al usuario

✓ - reduce el número de bits del campo de dirección

✗ - puede reducir el rendimiento

✗ - a menudo no se utilizan de manera efectiva (compilador)

Modos de direccionamiento

1. DIRECTO
2. INMEDIATO
3. INDIRECTO
4. RELATIVO
5. INDEXADO
6. REGISTRO
7. REGISTRO INDIRECTO

Modos de direccionamiento

- DIRECTO
  - El campo ***dirección*** de la instrucción (tanto para una transferencia como transformación del dato) contiene la dirección de memoria dónde está el dato (dirección efectiva)

    Ejemplo: (A) ← M[9100]

Modos de direccionamiento

- INMEDIATO
  - El dato se especifica en la propia instrucción (se puede hablar de campo ***operando*** en lugar de campo de ***dirección***).
  - *Útil para inicializar registros a una constante*

    Ejemplo: (A) ← 06

Modos de direccionamiento

- INDIRECTO
  - El campo ***dirección*** de la instrucción contiene la dirección de memoria en la cual se encuentra la dirección efectiva.

    Ejemplo: (A) ← M[M[9100]]

    Ver diferencia con el directo: (A) ← M[9100]

Modos de direccionamiento

- RELATIVO
  - La dirección efectiva se forma sumando al campo ***dirección*** de la instrucción el contenido de un registro específico de la CPU (generalmente el PC)
    - Útil cuando sabemos a qué distancia de la actual instrucción se encuentra el dato deseado

Ejemplo: (A) ← M[PC+9100]

Modos de direccionamiento

- INDEXADO
  - Dirección efectiva = campo ***dirección*** de la instrucción (tiene la dirección base) + contenido de un registro (distancia entre esa dirección inicial y el dato). Puede ser uno especial o uno cualquiera del bloque de registros. Puede ser uno o una pareja de registros.
    - *Útil para recorrer un array (incrementando el registro)*
    - Si hay un registro expresamente para esto (implícito)
    - Si puede ser cualquiera debe especificarse (explícito)

Ejemplo: (A) ← M [9100+(HL)]

Modos de direccionamiento

- REGISTRO
  - El contenido del registro es el dato (no hay dirección de memoria)

    Ejemplo: (A)← (R)

Modos de direccionamiento

- REGISTRO INDIRECTO
  - El contenido del registro es la dirección efectiva (la dirección de memoria que contiene el dato). El registro (o pareja de registros) funciona como un apuntador a memoria.

    ejemplo: (A) ← M[(HL)]

Modos de direccionamiento: ejemplo

DIRECCIÓN	CONTENIDO DE MEMORIA	
00FF	C. OPERACIÓN	MODO
0100	DIRECCIÓN : 0200	
0101	010A	
	• • •	
0200	0300	
0201	0F00	
	• • •	
0250	0400	
0251	0701	
	• • •	
0300	0120	
0301	100E	
	• • •	
0450	0890	
	• • •	

Si el código de operación indica "carga en el acumulador" y para los siguientes modos de direccionamiento, indica cuál será la dirección efectiva y el dato.

PC	0101

R	0250

MODO	DIR. EFECT.	DATO
DIRECTO	0200	0300
INMEDIATO	0100	0200
INDIRECTO	0300	0120
RELATIVO (a PC)	0301	100E
INDEXADO (R)	0450	0890
REGISTRO	----	0250
REGISTRO INDIRECTO	0250	0400

Tipos de instrucciones

1. Instrucciones de transferencia de datos
2. Instrucciones de procesamiento de datos
    1. Aritméticas
    2. Lógicas
    3. De desplazamiento
3. Instrucciones de control de secuencia

Tipos de instrucciones: instrucciones de transferencia de datos

- Cambian de lugar los datos sin modificarlos
  - Registros del μP↔ Memoria
  - Registros del μP ↔ Registros de E/S
  - Registros del μP ↔ Registros del μP

- *Tabla (cada instrucción con el mnemónico recomendado por la norma **IEEE std. 694-1985**)*

Tipos de instrucciones: instrucciones de transferencia de datos

Nombre	Mnemónico	
Cargar (leer)	LD	*Load*
Almacenar	ST	*Store*
Copiar	MOVE	*Move*
Intercambiar	XCH	*Exchange*
Meter en pila (empujar)	PUSH	*Push*
Sacar de pila (saltar)	POP	*Pop*
Entrada	IN	*Input*
Salida	OUT	*Output*

Tipos de instrucciones: de transferencia de datos (pila).

- Memoria pila → baja el rendimiento ➔ sólo para información de estado (llamada/retorno de procedimiento e interrupciones)
- POP → Lee TOS (cima de pila) y actualiza SP (se escribirá encima con la siguiente instrucción PUSH)
- Inicializar SP (puntero de pila)→ se actualiza automáticamente y siempre es accesible → el microprocesador no tiene que producir dirección de memoria
- El SP puede representar el TOS o la siguiente posición y la pila puede crecer hacia direcciones crecientes o decrecientes, según la organización de la pila

SP←SP-1	R1←M[SP]
M[SP]←R1	SP←SP+1

Instrucciones de procesamiento de datos: aritméticas

Instrucciones básicas:

1. Suma
2. Resta
3. Multiplicación
4. División

Los computadores simples sólo las dos primeras, el resto mediante programación

Instrucciones de procesamiento de datos: aritméticas

Nombre	Mnemónico	
Incrementar	INC	*increment*
Decrementar	DEC	*Decrement*
Sumar	ADD	*Add*
Restar	SUB	*Substract*
Multiplicar	MUL	*Multiply*
Dividir	DIV	*Divide*
Suma con acarreo	ADDC	*Add with carry*
Resta con préstamo	SUBB	*Substract with borrow*
Resta inversa	SUBR	*Substract reverse*
Negación	NEG	*Negate*

Instrucciones de procesamiento de datos: lógicas y de bits

- Realizar operaciones binarias con los datos de memoria o registros → útil para manejar información binaria de grupos de bits (cada bit es una variable lógica)

- El segundo operando se llama máscara:

AND

B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0
0	0	0	0	1	0	0	0
0	0	0	0	B_3	0	0	0

OR

B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0
1	1	0	1	1	1	0	1
1	1	B_5	1	1	1	B_1	1

Instrucciones de procesamiento de datos: lógicas y de bits

Nombre	Mnemónico	
Poner a '0'	CLR	*clear*
Poner a '1'	SET	*set*
Negar	NOT	*complement*
Producto lógico	AND	*AND*
Suma lógica	OR	*OR*
Suma exclusiva	XOR	*exclusive OR*
Poner a '0' acarreo	CLCRC	*clear carry*
Poner a '1' acarreo	SETC	*set carry*
Negar acarreo	COMC	*complement carry*

Instrucciones de procesamiento de datos: desplazamiento

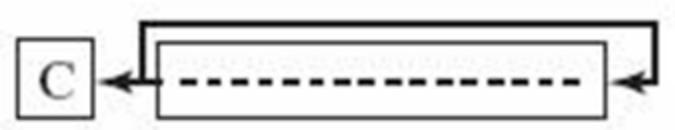
- Despl. lógico a la derecha → meter '0'
- Despl. lógico a la izquierda→ meter '0' (*)
- Despl. aritm. a la derecha→ mantener signo
- Despl. aritm. a la izquierda → meter '0' (*)

(*) igual, pero el aritmético modifica el flag V

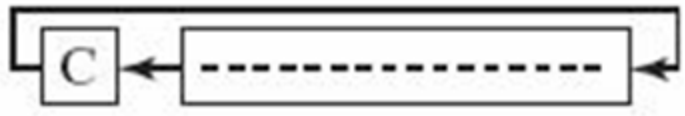
Números en complemento a 2

Instrucciones de procesamiento de datos: desplazamiento

- Rotar = desplazamiento circular



- Rotar con acarreo = el acarreo funciona como extensión de 1 bit para el dato que rotamos



Instrucciones de procesamiento de datos: desplazamiento

Nombre	Mnemónico
Desplazamiento lógico a la derecha	SHR
Desplazamiento lógico a la izquierda	SHL
Desplazamiento aritmético a la derecha	SHRA
Desplazamiento aritmético a la izquierda	SHLA
Rotar a la derecha	ROR
Rotar a la izquierda	ROL
Rotar a la derecha con acarreo	RORC
Rotar a la izquierda con acarreo	ROLC

Instrucciones de control de secuencia

Nombre	Mnemónico	
Salto	BR	*Branch*
Salto incondicional	JMP	*Jump*
Avanzar	SKP	*Skip*
Llamada a procedimiento	CALL	*Call*
Retorno de procedimiento	RET	*Return*
Comparación (resta)	CMP	*Compare*
Comparación (AND)	TEST	*Test*

Diagrama de flujo

- El diagrama de flujo (*flowchart*) es un tipo de diagrama que representa los pasos y la evolución de un algoritmo o proceso.
- Dichos pasos se representan mediante "cajas" y el orden en que se suceden está indicado mediante flechas.
- Los diagramas de flujo se utilizan para diseñar, analizar o documentar un proceso o programa.
- Permiten ver gráficamente los pasos del proceso, facilitando su entendimiento y la localización de fallos o cuellos de botella.

Diagrama de flujo

Bloques estructurales

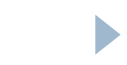
- **Elipse** (o cuadrado redondeado): bloque de comienzo o fin.
- **Rectángulo**: paso de proceso genérico, actividad.
- **Rombo**: decisión (plantea una cuestión necesaria con respuesta del tipo sí/no (T/F)).
- **Rectángulo con rayas en ambos lados**: subrutinas (pasos del proceso complejos).
- **Paralelogramo**: entrada/salida.
- **Flecha**: muestra el flujo del control; puede ir de un bloque a otro o de un bloque a una flecha (mostrando que la ejecución del proceso se halla dentro de un bucle (loop)).

Diagrama de flujo

- Ejemplo (programación basada en eventos): paso de peatones simplificado.

  Un semáforo (para vehículos) se mantiene verde hasta que un peatón pulse el botón de solicitud; entonces pasa a ámbar, tras 5 s pasa a rojo (los peatones tienen 10 s para cruzar); transcurridos los 10 s vuelve a verde.

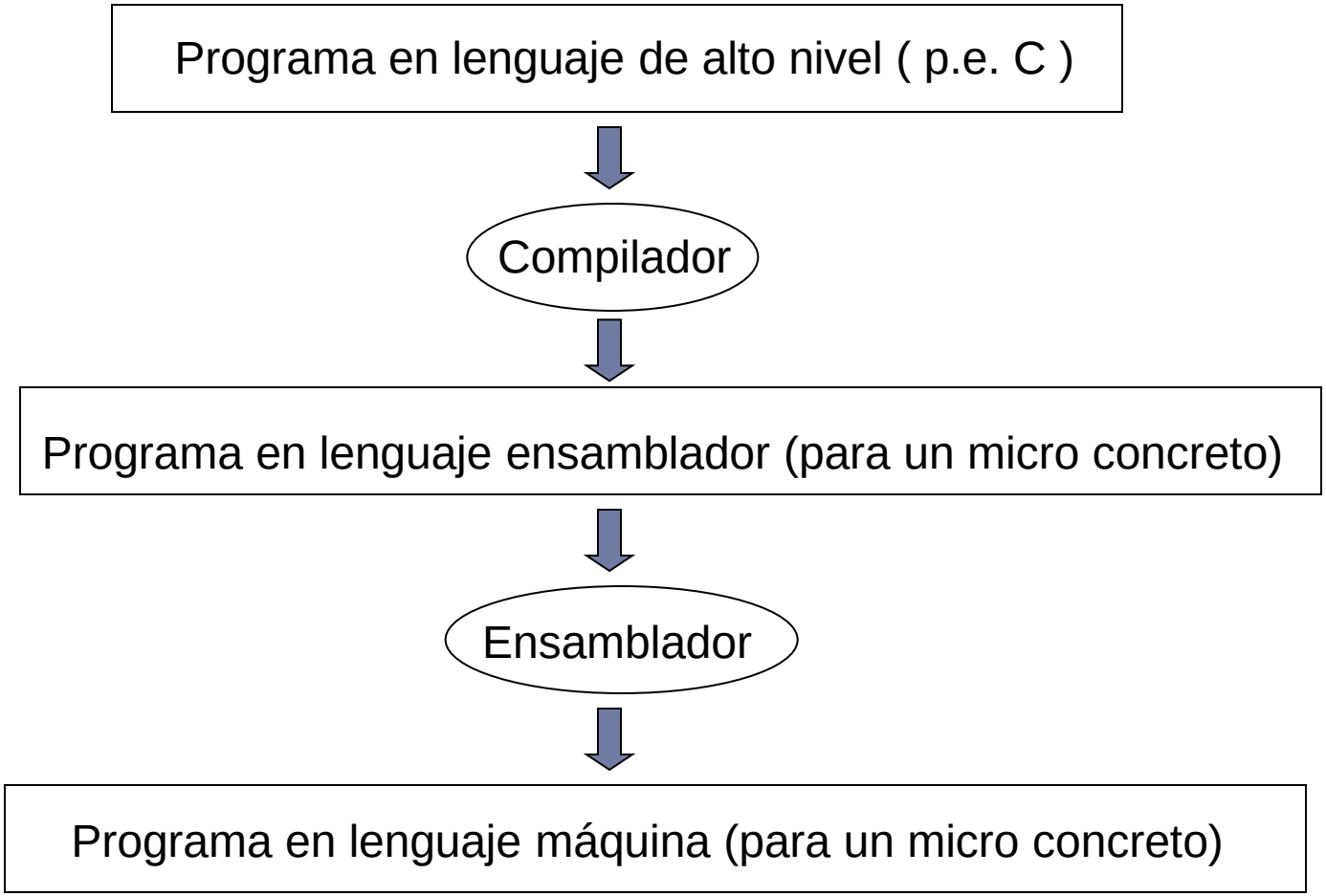
  Realiza el diagrama de flujo del programa que controla este semáforo suponiendo que la frecuencia del reloj del sistema es de 1 Hz, la variable que indica el estado del semáforo es "S" y la variable binaria "solicitud" indica si se ha pulsado ('1') o no ('0') dicho botón.



El lenguaje del computador: niveles de abstracción

Nivel		
6	Usuario	Programas ejecutables
5	Lenguaje de alto nivel	C, Java, etc.
4	Ensamblador	Lenguaje ensamblador
3	Software del sistema	Sistema operativo
2	Máquina	Arquitectura del conjunto de instrucciones (ISA)
1	Control	Control cableado o microprogramado
0	Lógica digital	Puertas lógicas, C.I., etc.

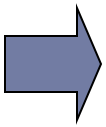
El lenguaje del computador



Ejemplo

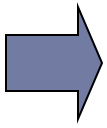
Lenguaje de alto nivel

```
#include <aaaaa.a>
#define bbb B
#define ccc C
void main(void) {
while (1) {
bbb = ccc + 1;
}
```



Lenguaje ensamblador

```
MOV A,B
bucle:
ADI 01
JNZ bucle
MOV C,A
```



Lenguaje máquina

```
01111000
11000110
00000001
11000010
00000001
10000000
01001111
```

Arquitectura del juego de instrucciones (ISA)