Programación en assembler Simulador MSX88

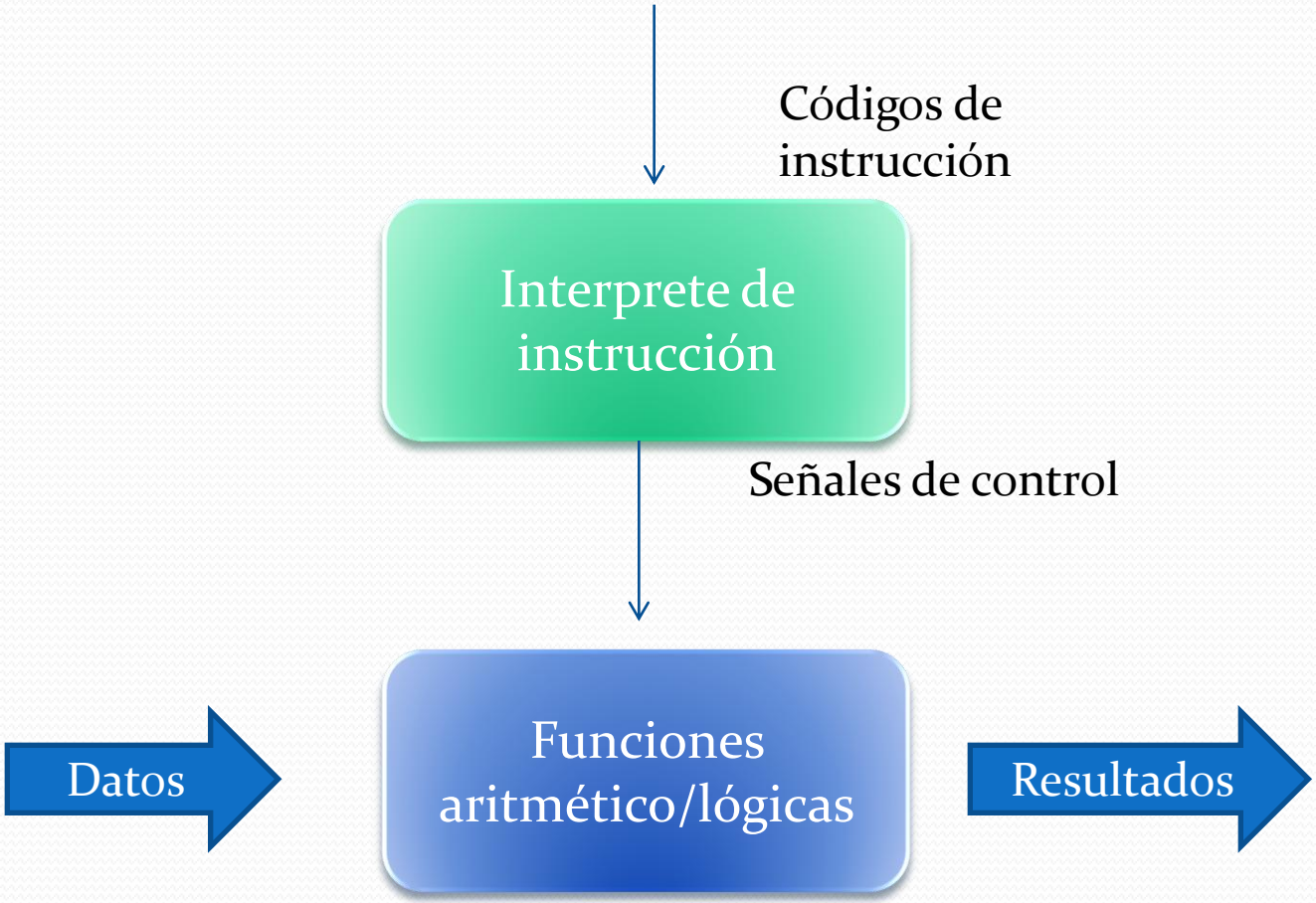
Repaso concepto de programa

- ANTES se tenían **<u>sistemas cableados</u>**

- Programación en hardware: cuando cambiamos las tareas, debemos cambiar el hardware

Repaso concepto de programa

AHORA



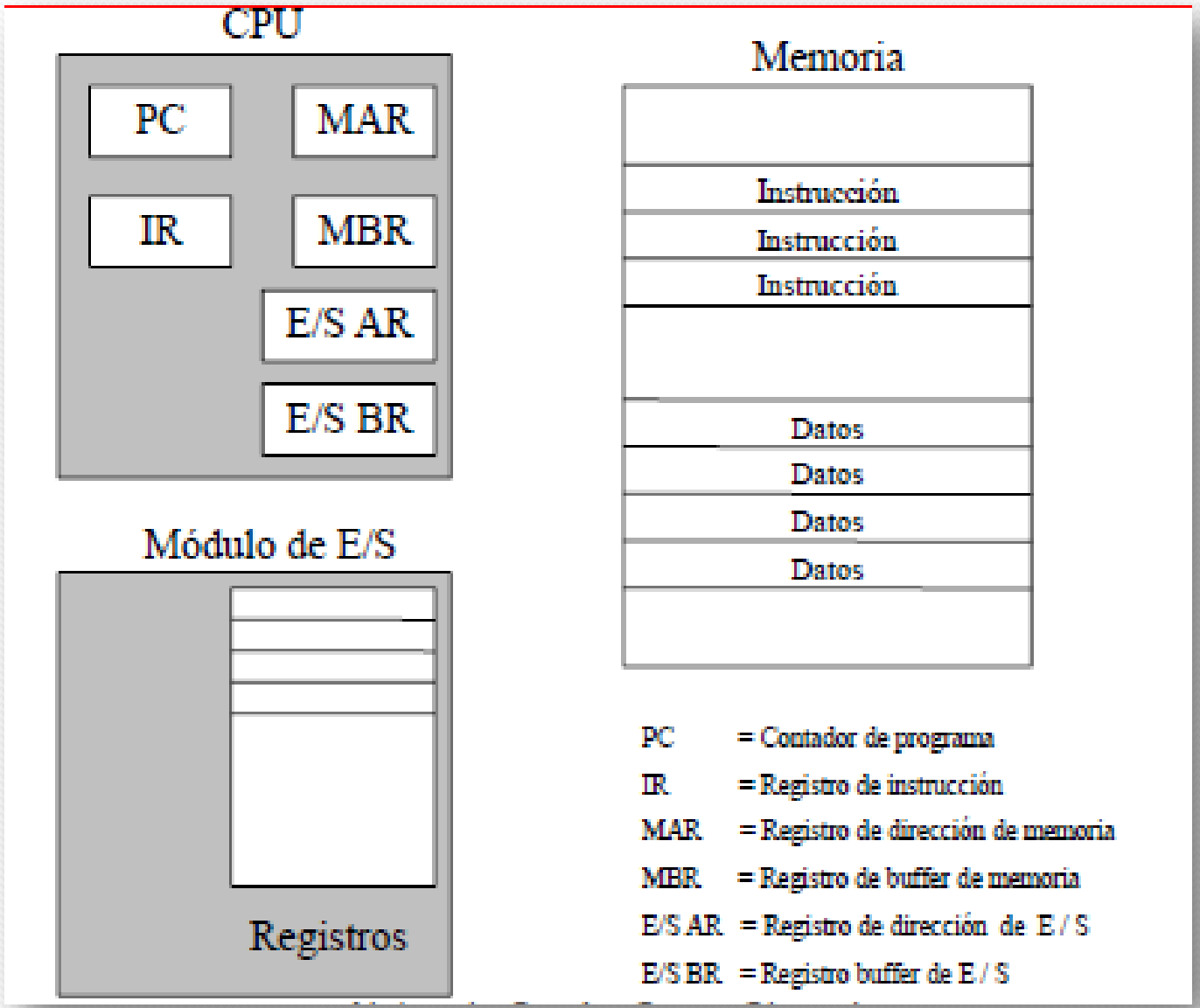
- Programación en software: en cada paso se efectúa alguna operación sobre los datos

Repaso concepto de programa

- Para cada paso se necesita un nuevo conjunto de señales de control.
- Las instrucciones proporcionan esas señales de control.
- Aparece el nuevo concepto de programación.

No hay que cambiar el hardware !!!

Componentes de una computadora



Repertorio (set) de instrucciones

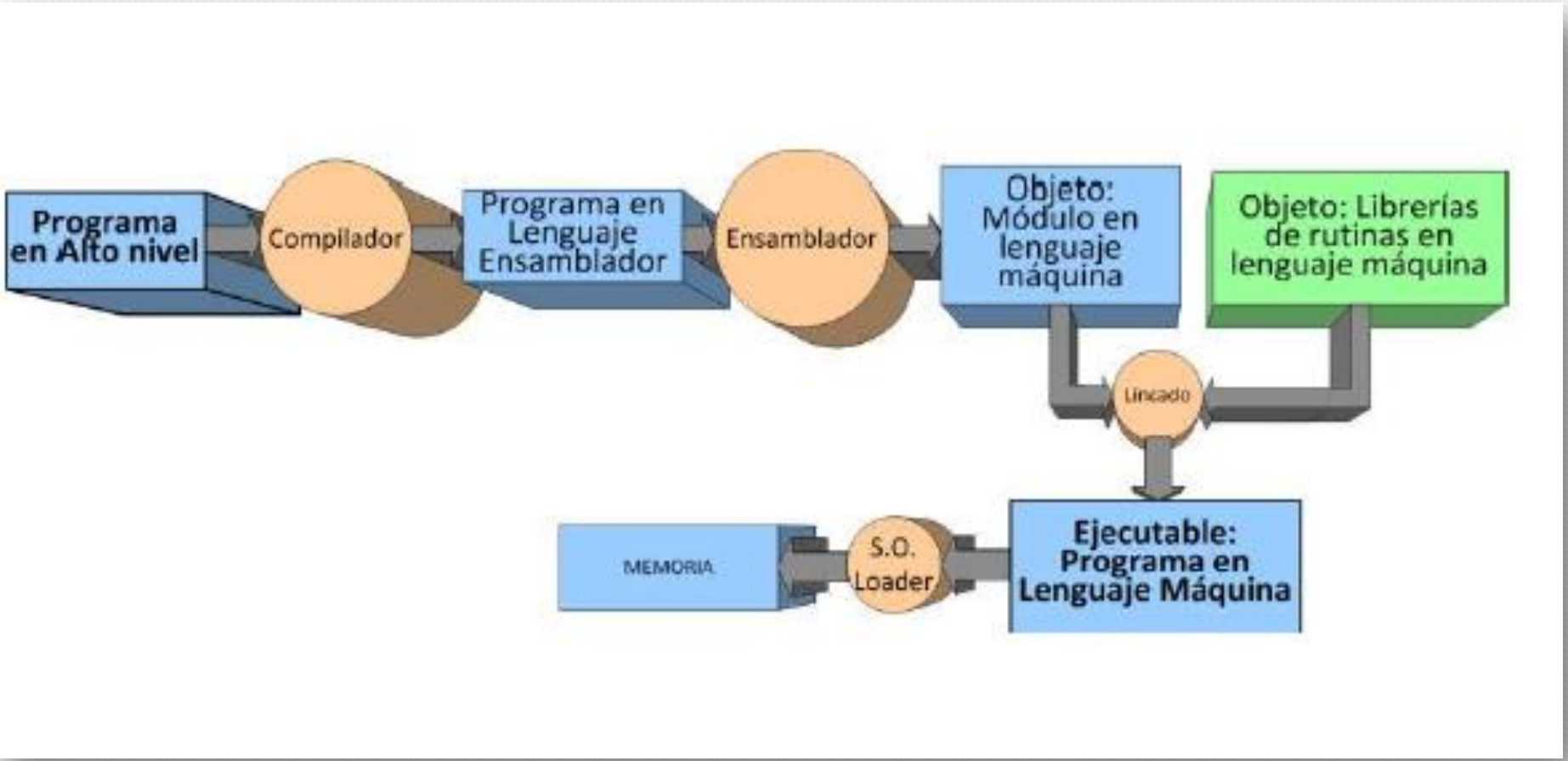
Es el conjunto completo de instrucciones que se realizan en una CPU.

- Código máquina
- Binario

Representado simbólicamente por un conjunto de códigos de ensamblaje

- **de operaciones**: ADD (sumar), SUB (restar), LOAD (cargar datos en un registro)
- **de operandos:** ADD BX, PEPE; se interpreta como sumar contenidos de reg BX y dirección PEPE, el resultado se guarda en reg BX

Del Alto nivel al lenguaje de maquina



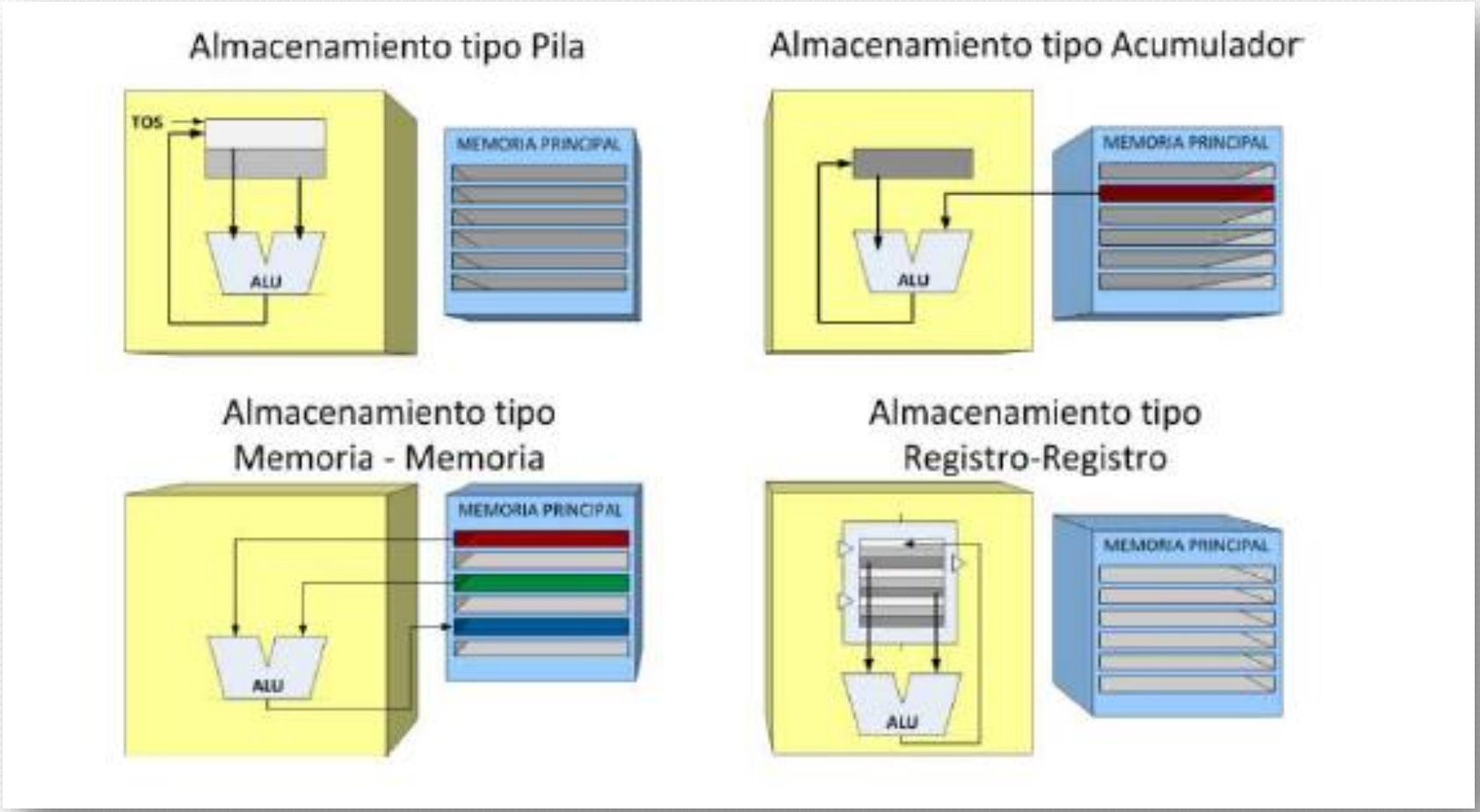
Elementos de una instrucción

- Código de operación (“Cod Op”)
- Referencia a operandos fuentes
- Referencia al operando resultado
- Referencia a la siguiente instrucción

¿Dónde se almacenan los operandos?

- Memoria principal
- o memoria virtual o en memoria cache
- Registro de la CPU
- Dispositivo de E/S

Diferentes tipos de almacenamiento



Tipos de instrucciones

- **Procesamiento de datos:**

  instrucciones aritmético-lógicas
- **Almacenamiento de datos:**

  instrucciones de memoria
- **Transferencia de datos:**

  instrucciones de E/S
- **Control:**

  instrucciones de testeo y flujo del programa

¿Cuantas direcciones de memoria por instrucción se debe tener?

Más direcciones por instrucción

- Instrucciones más complejas
- Más registros:

  Las operaciones entre los registros son más rápidas.
- Menos instrucciones por programa

Menos direcciones por instrucción

- Instrucciones menos complejas
- Más instrucciones por programa

  La captación/ejecución de las instrucciones es más rápida.

Decisiones en el diseño del conjunto de instrucciones

- **Tipos de operandos (datos)**

- **Repertorio de operaciones**

  ¿Cuántas operaciones se considerará?

  ¿Cuáles operaciones se realizarán?

  ¿Cuán compleja será cada una de ellas?

- **Formatos de instrucciones:**

  Longitud de instrucción

  Número de direcciones

  Tamaño de los campos

Decisiones en el diseño del conjunto de instrucciones

- **Registros**

  Número de registros de la CPU referenciables

  ¿En qué registros se pueden ejecutar qué operaciones?
- **Modos de direccionamiento**

  ¿cómo es especificada la ubicación de un operando o una instrucción?
- **RISC contrapuesto a CISC**

(Computadora de conjunto reducido de instrucciones) a (Computadora de conjunto complejo de instrucciones)

Tipos de operandos

- **Direcciones**
- **Números**

  punto fijo ó punto flotante
- **Caracteres**

  ASCII, EBCDIC ...etc.
- **Datos lógicos**

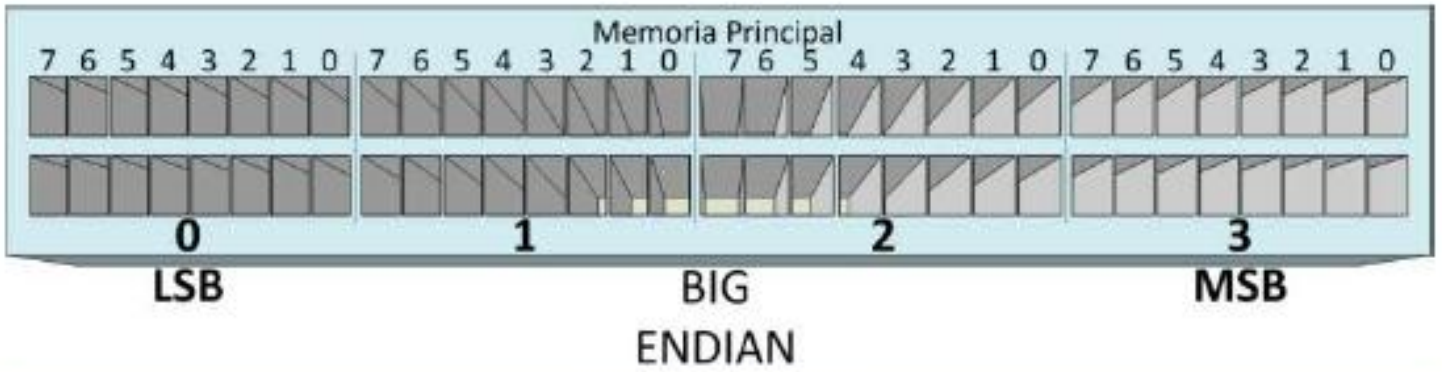
  Bits (1 ó 0)

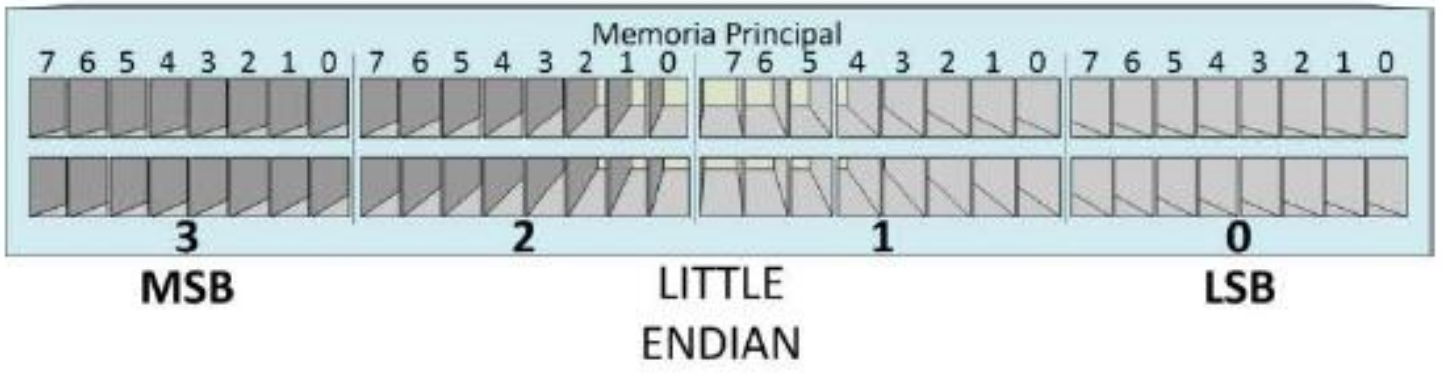
  Ej: flags o indicadores

Orden de los bytes

Supongamos una memoria **direccionable de a byte**

¿En qué orden se leen aquellos números que ocupan más de un byte? Ejemplo: La palabra doble 98765432H (32 bits) se puede almacenar en 4 bytes consecutivos de las siguientes 2 formas:





Orden de los bytes

Dir. de byte	Forma 1	Forma 2
00	98	32
01	76	54
02	54	76
03	32	98

¿cuál forma uso?

Big endian: el byte más significativo en la dirección con valor numérico más bajo

Little endian: el byte menos significativo en la dirección con valor numérico más bajo

Orden de los bytes

- Intel 80x86, Pentium y VAX son "little-endian".
- IBM S/370, Motorola 680x0 (Mac) y la mayoría de los RISC son "big-endian".

Incompatibilidad !!!

Tipos de operaciones

- Transferencias de datos
- Aritméticas
- Lógicas
- Conversión
- Entrada/Salida
- Control del sistema
- Control de flujo

Transferencia de datos

- Debe especificarse:
  - Ubicación del operando fuente
  - Ubicación del operando destino
  - Tamaño de los datos a ser transferidos
  - Modo de direccionamiento
- Diferentes movimientos ->diferentes instrucciones
  - Reg-Reg, Reg-Mem o Mem-Reg
- O una instrucción y diferentes direcciones
  - MOV destino, fuente ; copia fuente a destino

Aritméticas

- Operaciones básicas:

  Add, **Sub**stract, **Mul**tiply y **Div**ide
  - Números enteros sin/con signo.
  - ¿Números en punto flotante?
- Pueden incluirse otras operaciones ...
  - **Inc**rement o **Dec**rement (en 1 el operando)
  - **Neg**ate: cambia el signo del operando (Ca2).
  - **Abs**olute: toma el valor absoluto del operando.
  - Shift left/right: desplaza bits a izq/der un lugar

Lógicas-Conversión

Operaciones que manipulan bits individualmente

- Operaciones Booleanas.

  AND, OR, XOR, NOT

- Otras operaciones
  - Rotate left/right: rota las posiciones de los bits a izq/der

Operaciones para cambiar formatos de datos

- Conversión de binario a decimal o de EBCDIC a ASCII

Entrada/Salida

- Pocas instrucciones pero de acciones específicas
  - IN ó OUT
- Se pueden realizar utilizando instrucciones de movimiento de datos
  - MOVE
- Se pueden realizar a través de un controlador aparte: DMA (Direct Memory Access)

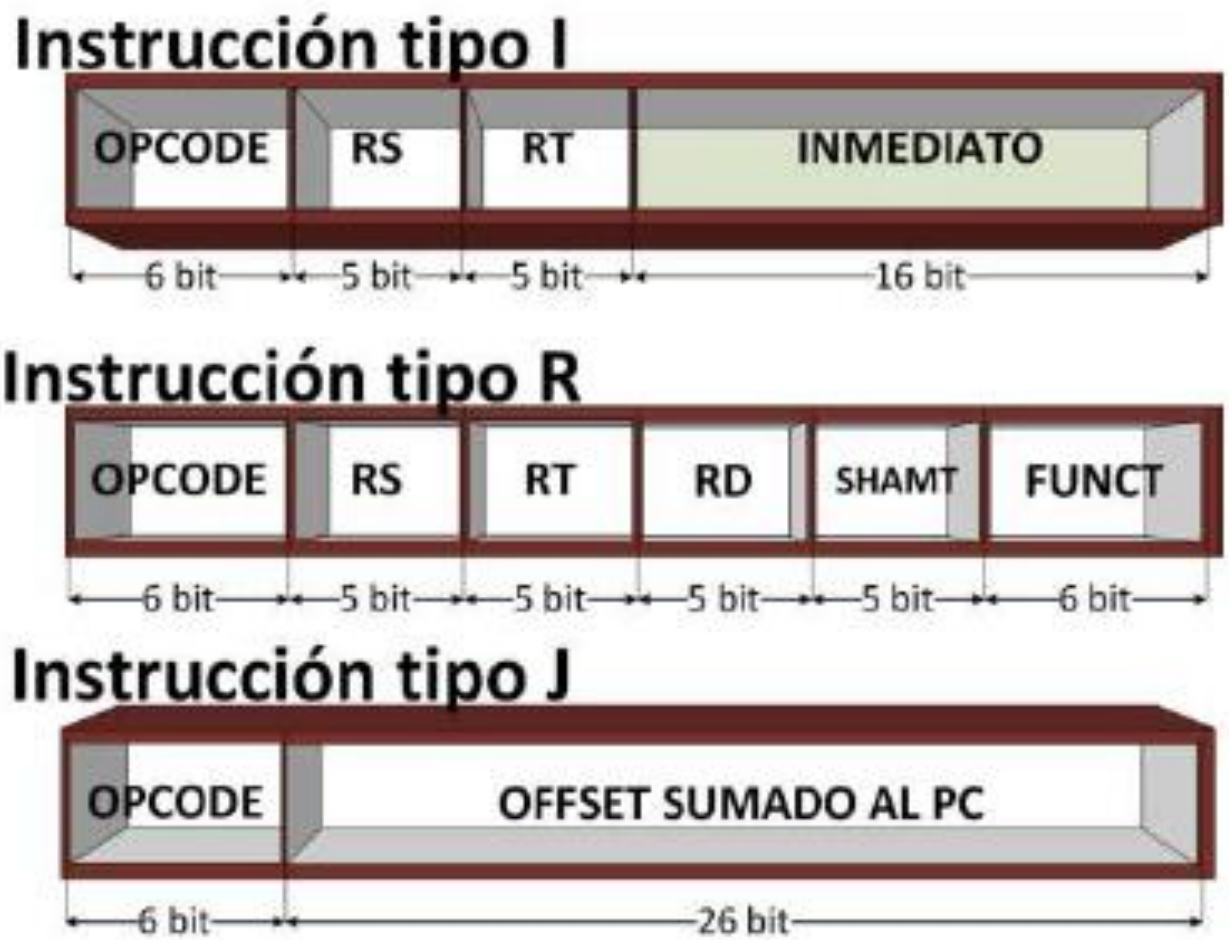
Control de Flujo

Modifican el valor contenido en el registro PC

- Salto Incondicional
  - **JMP** equis ; saltar a la posicion 'equis'
- Salto Condicional
  - **JZ** equis ; saltar a la posición 'equis', si bandera Z=1
- Salto con retorno o llamada a subrutina
  - **CALL** subrut ;saltar a la posición 'subrut'

  Para retornar al programa que llamó, se debe utilizar la instrucción **RET** como última instrucción del cuerpo de subrutina

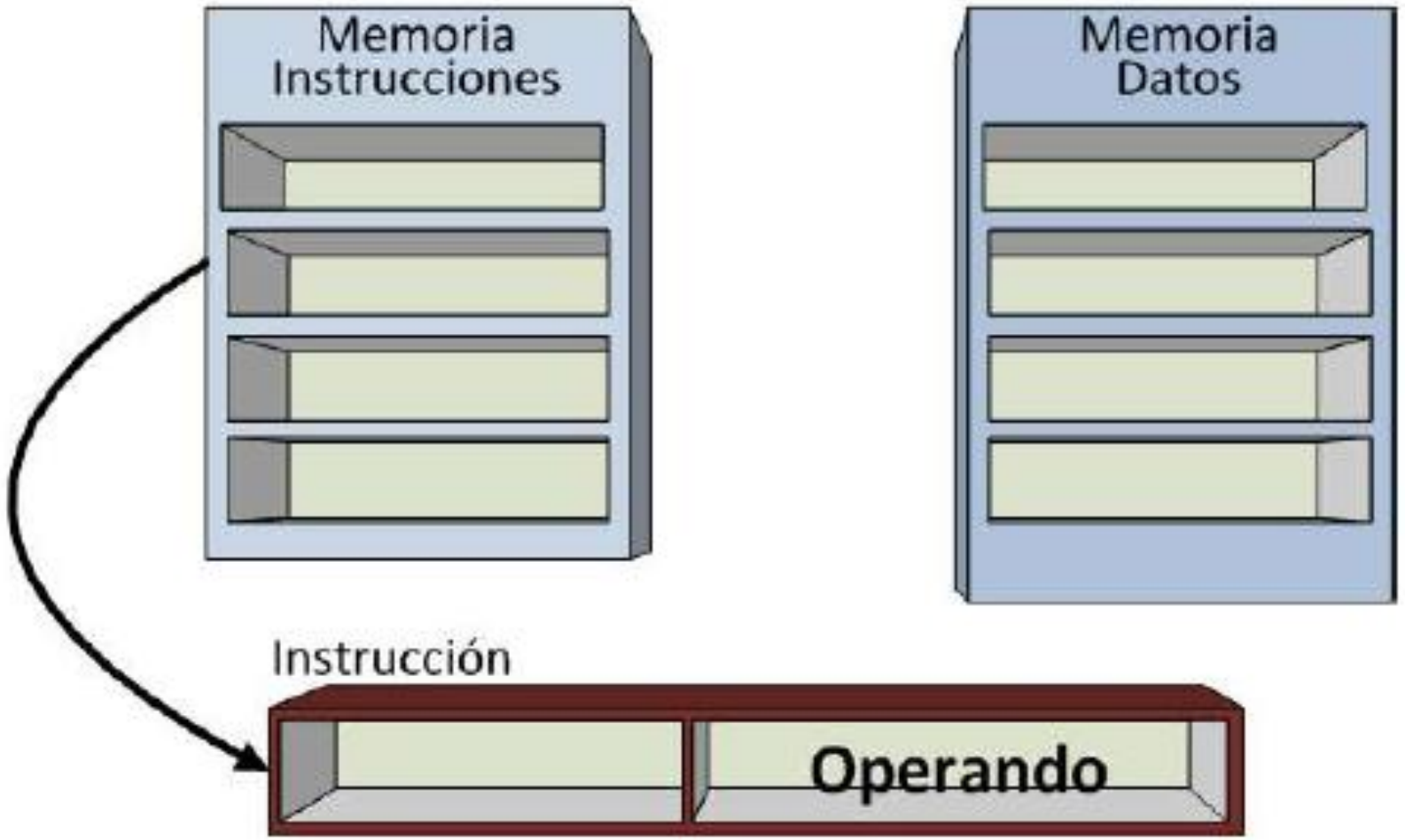
Formatos de instrucción



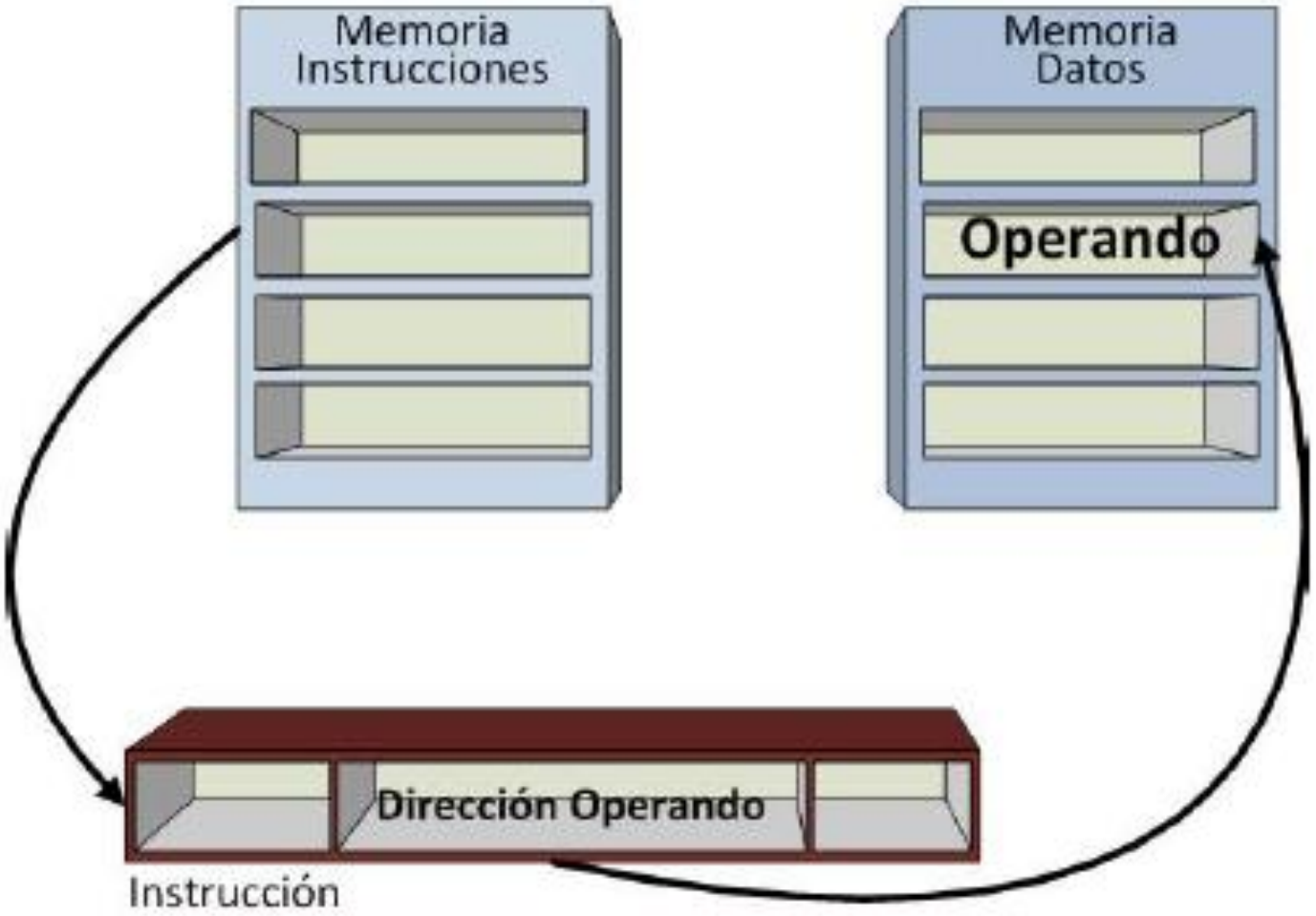
Modos de direccionamiento

- Inmediato
- Directo de memoria o Absoluto
- Directo de Registro
- Indirecto de memoria (en desuso)
- Indirecto con registro
- Indirecto con Desplazamiento
  - basado, indexado o relativo al PC
  - Pila (o relativo al SP)

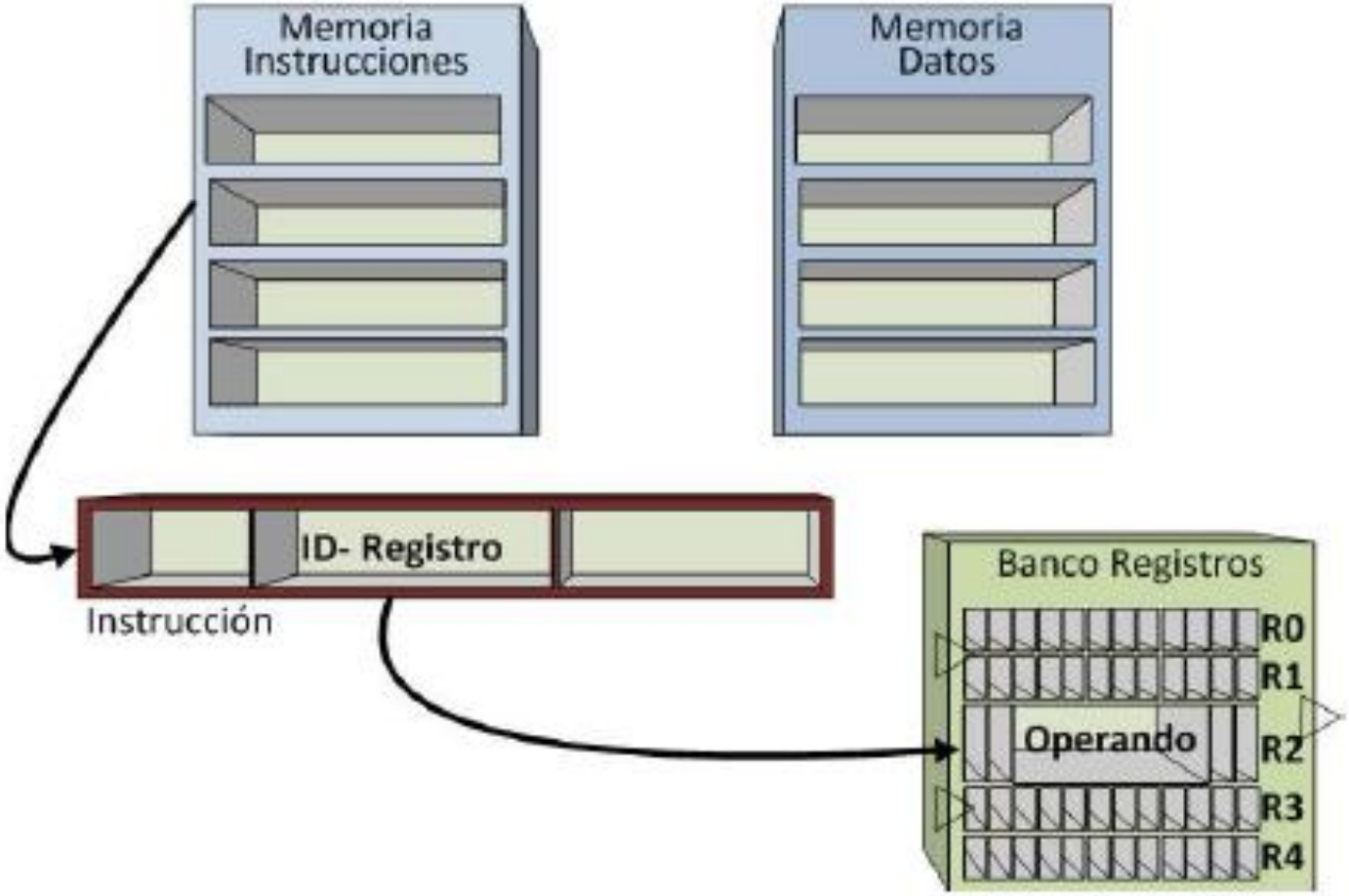
MDD Inmediato



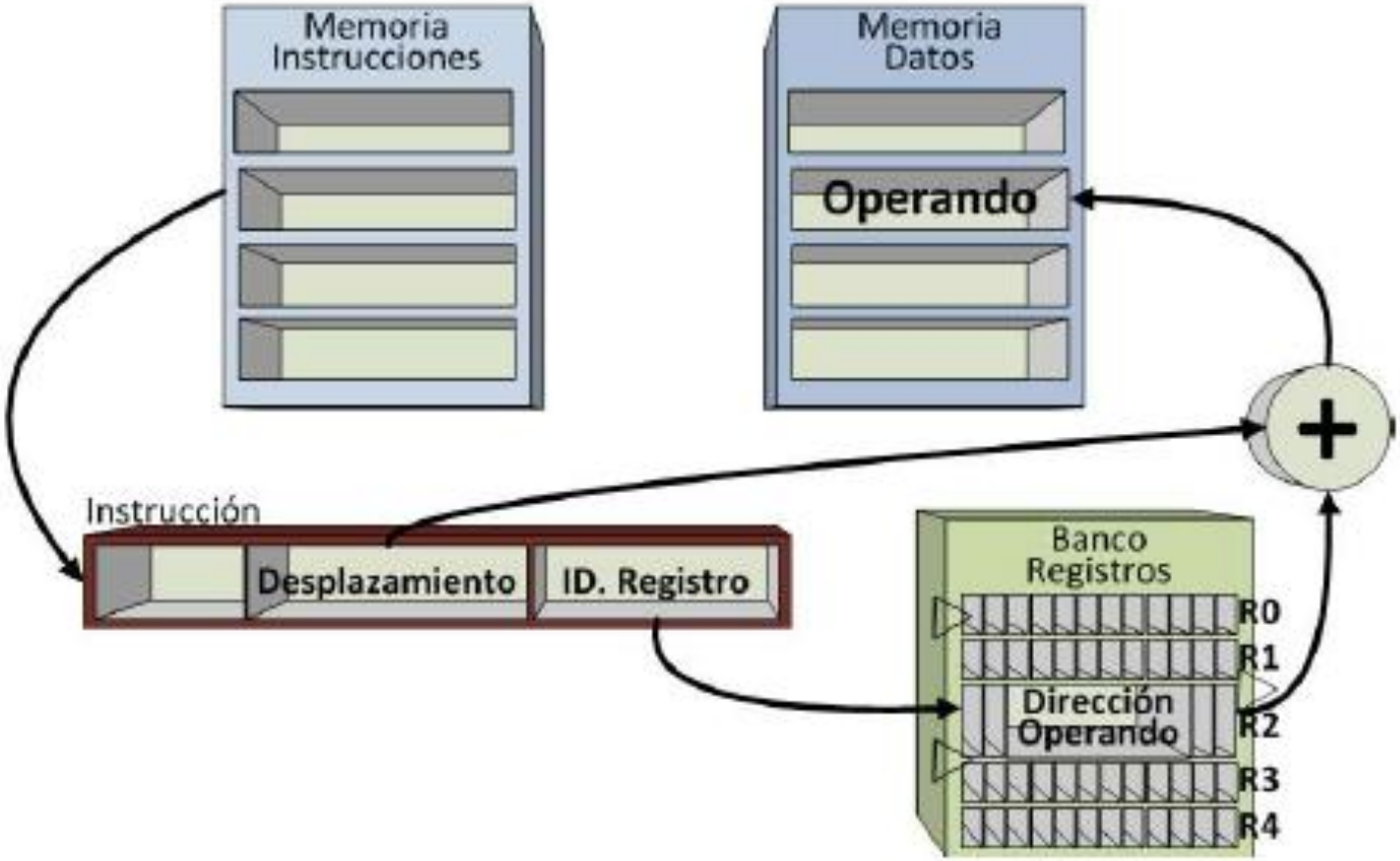
MMD Directo o Absoluto (de memoria)



MMD Directo de registro

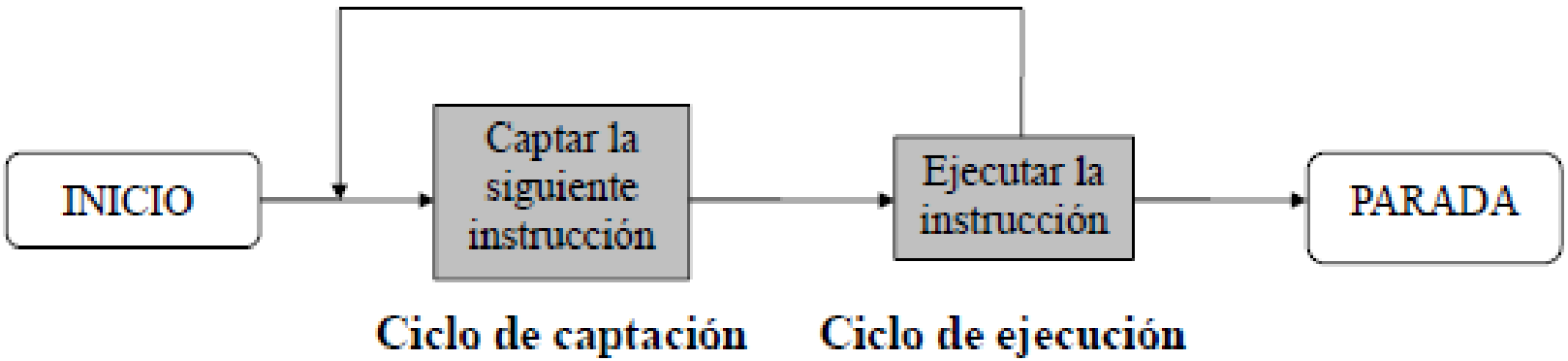


MDD Indirecto con desplazamiento



Ciclo de instrucción básico

- Dos pasos:
  - Captación
  - Ejecución



Ciclo de captación

- La dirección de la instrucción que se debe captar se encuentra en el registro Contador de Programa (PC)
- La UC **capta** la instrucción desde la Memoria
  - La instrucción va al registro de instrucción (IR)
- El registro PC se incrementa
  - a no ser que se indique lo contrario.
- La UC interpreta la instrucción captada y debe lleva a cabo la acción requerida

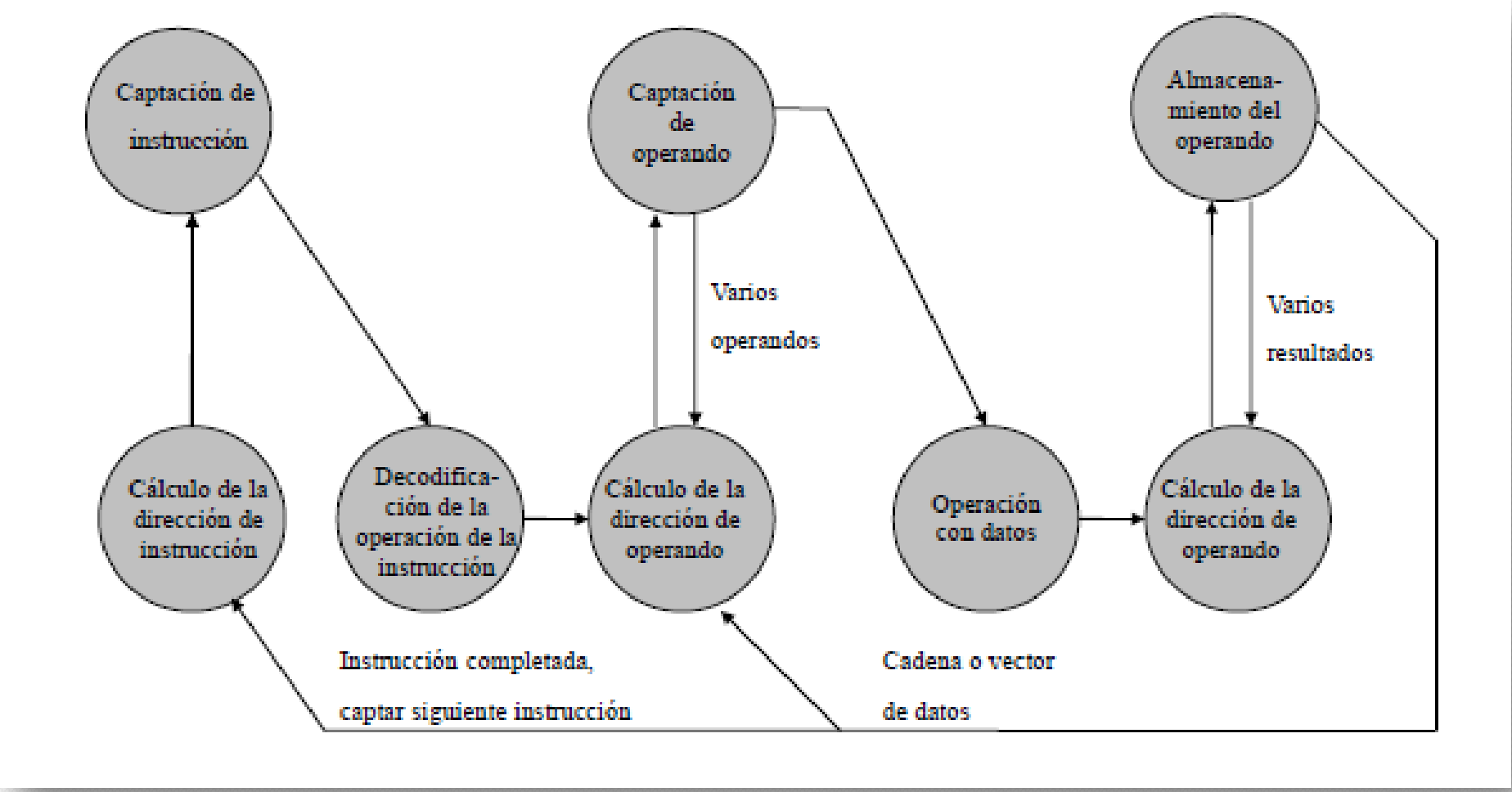
Ciclo de ejecución

Acciones posibles:

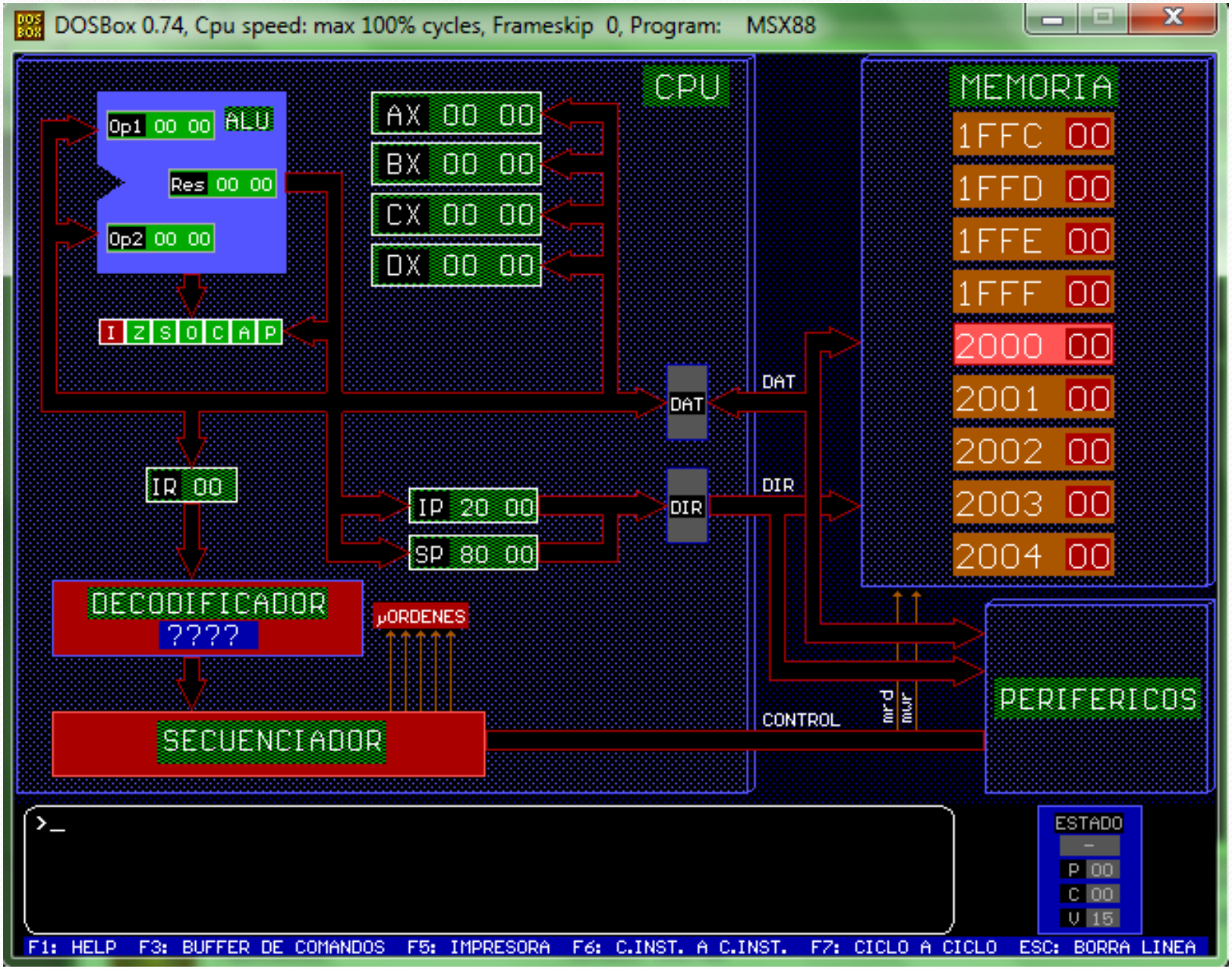
- Procesador - memoria
  - Transferencia de datos CPU - Memoria.
- Procesador - E/S
  - Transferencias de datos CPU y módulo de E/S.
- Procesamiento de datos
  - Alguna operación aritmética o lógica con los datos.
- Control
  - Alteración de la secuencia de ejecución.
    - Instrucción de salto

ó combinación de las acciones anteriores

Diagrama de estados del ciclo de instrucción



Simulador MSX88



MSX88 Instrucciones de transferencia

1	MOV *dest,fuente*	Copia *fuente* en *dest*	(*dest*)←(*fuente*)
2	PUSH *fuente*	Carga *fuente* en el tope de la pila	(SP)←(SP)-2;[SP+1:SP] ←(*fuente*)
2	POP *dest*	Desapila el tope de la pila y lo carga en *dest*	(*fuente*)←[SP+1:SP]; (SP)←(SP)+2
2	PUSHF	Apila los flags	(SP)←(SP)-2;[SP+1:SP] ←(flags)
2	POPF	Desapila los flags	(flags)←[SP+1:SP]; (SP)←(SP)+2
3	IN *dest,fuente*	Carga el valor en el puerto *fuente* en *dest*	(*dest*)←(*fuente*)
4	OUT *dest,fuente*	Carga en el puerto *dest* el valor en *fuente*	(*dest*)←(*fuente*)

1. *dest/fuente* son: *reg/reg, reg/mem, reg/op.inm, mem/reg, mem/op.inm.*
   mem puede ser una etiqueta (dir.directo) o [BX] (dir.indirecto).
2. *dest* y *fuente* solo pueden ser registros de 16 bits.
3. *dest/fuente* son: *AL/mem, AX/mem, AL/DX, AX/DX.*
4. *dest/fuente* son: *mem/AL, mem/AX, DX/AL, DX/AX.*
   mem debe ser dirección entre 0 y 255. Puede ser un operando inmediato o una etiqueta.

MSX88 Instrucciones aritmético - lógicas

1	ADD *dest,fuente*	Suma *fuente* y *dest*	(*dest*)← (*dest*)+(*fuente*)
1	ADC *dest,fuente*	Suma *fuente*, *dest* y *flag C*	(*dest*)← (*dest*)+(*fuente*)+C
1	SUB *dest,fuente*	Resta *fuente* a *dest*	(*dest*)← (*dest*)-(*fuente*)
1	SBB *dest,fuente*	Resta *fuente* y *flag C* a *dest*	(*dest*)← (*dest*)-(*fuente*)-C
1	CMP *dest,fuente*	Compara *fuente* con *dest*	(*dest*)-(*fuente*)
5	NEG *dest*	Negativo de *dest*	(*dest*)← CA2(*dest*)
5	INC *dest*	Incrementa *dest*	(*dest*)← (*dest*)+1
5	DEC *dest*	Decrementa *dest*	(*dest*)← (*dest*)-1
1	AND *dest,fuente*	Operación *fuente* AND *dest* bit a bit	(*dest*)← (*dest*) AND (*fuente*)
1	OR *dest,fuente*	Operación *fuente* OR *dest* bit a bit	(*dest*)← (*dest*) OR (*fuente*)
1	XOR *dest,fuente*	Operación *fuente* XOR *dest* bit a bit	(*dest*)← (*dest*) XOR (*fuente*)
5	NOT *dest*	Complemento a 1de *dest*	(*dest*)← CA1(*dest*)

1. *dest/fuente* son: *reg/reg, reg/mem, reg/op.inm, mem/reg, mem/op.inm.*

5. *dest* solo puede ser *mem* o *reg.*

mem puede ser una etiqueta (dir.directo) o [BX], siendo (BX) una dirección de memoria (dir.indirecto).

MSX88 Instrucciones de transferencia de control

6	CALL *etiqueta*	Llama a subrutina cuyo inicio es *etiqueta*	
6	RET	Retorna de la subrutina	
6	JZ *etiqueta*	Salta si el último valor calculado es cero	Si Z=1, (IP)←*mem*
6	JNZ *etiqueta*	Salta si el último valor calculado no es cero	Si Z=0, (IP)←*mem*
6	JS *etiqueta*	Salta si el último valor calculado es negativo	Si S=1, (IP)←*mem*
6	JNS *etiqueta*	Salta si el último valor calculado no es negativo	Si S=0, (IP)←*mem*
6	JC *etiqueta*	Salta si el último valor calculado produjo carry	Si C=1, (IP)←*mem*
6	JNC *etiqueta*	Salta si el último valor calculado no produjo carry	Si Z=1, (IP)←*mem*
6	JO *etiqueta*	Salta si el último valor calculado produjo overflow	Si O=1, (IP)←*mem*
6	JNO *etiqueta*	Salta si el último valor calculado no produjo overflow	Si O=0, (IP)←*mem*
6	JMP *etiqueta*	Salto incondicional a *etiqueta*	(IP)←*mem*

6. *mem* es la dirección de memoria llamada *etiqueta*.

Subrutinas

- Innovación en lenguajes de programación
- Programa auto-contenido
- Puede invocarse desde cualquier punto de un programa
  - mediante instrucción CALL
- Brinda economía (código usado varias veces) y modularidad (subdivisión en unidades pequeñas).
- Requiere pasaje de argumentos (parámetros)
  - por valor (copia de una variable)
  - por referencia (dirección de la variable)

Pasaje de parámetros

- Vía registros
  - El número de registros es la principal limitación
  - Es importante documentar que registros se usan
- Vía memoria
  - Se usa un área definida de memoria (RAM).
  - Difícil de estandarizar

Pasaje de parametros

- Vía pila (stack)
  - Es el método más ampliamente usado.
  - El verdadero "pasaje de parámetros".
  - Independiente de memoria y registros.
  - Hay que comprender bien como funciona porque la pila (stack) es usada por el usuario y por el sistema.

En x86, SP apunta al último lugar usado

Funcionamiento de la pila

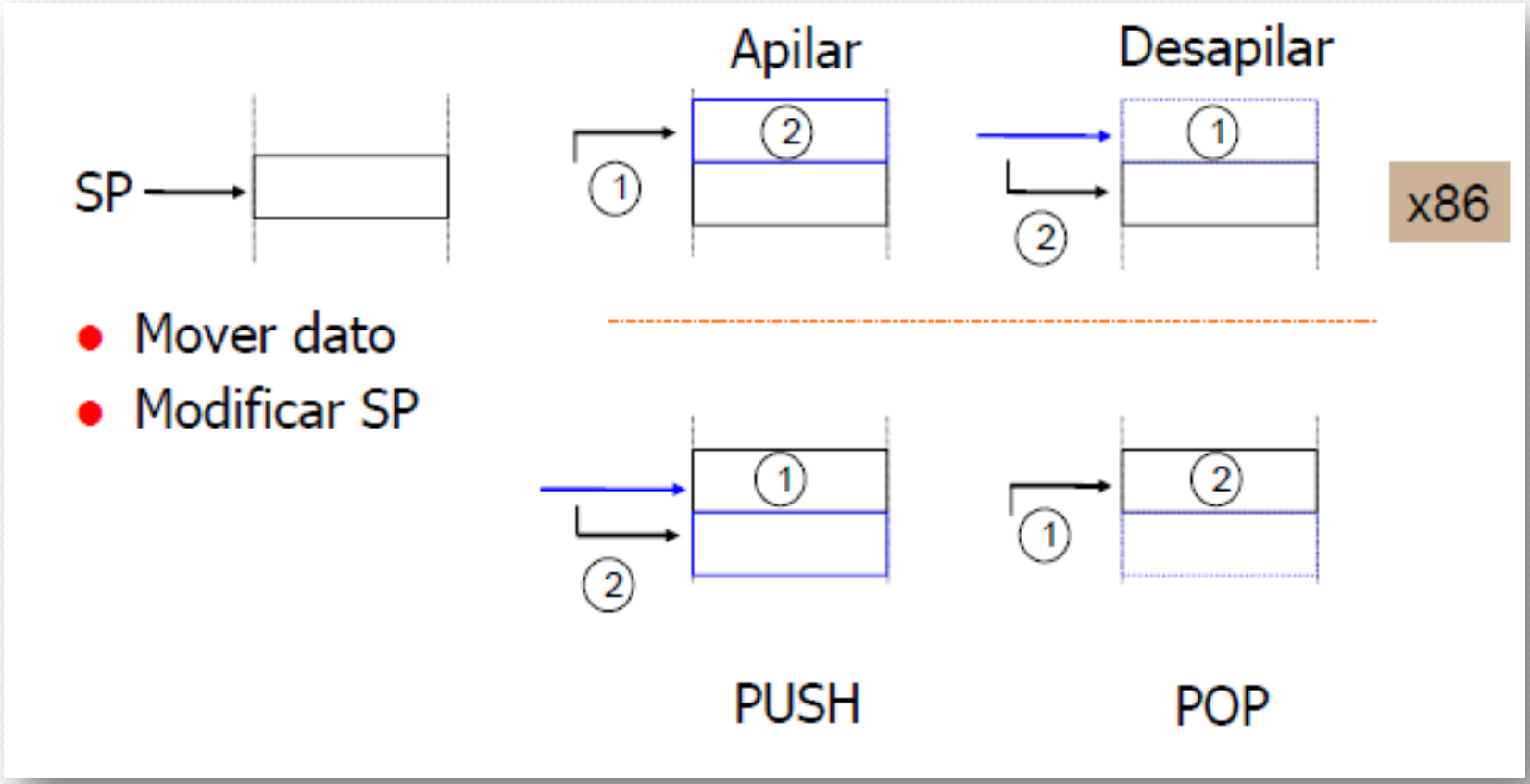
- El operando está (de forma implícita) en la cabeza de la pila
- Se requiere un registro Puntero de Pila (SP)
  - Contiene la dirección de la cabeza de la pila
- Operaciones sobre la pila
  - **PUSH** ; operación de Apilar
  - **POP** ; operación de Desapilar
  - Son inversas entre sí

Operaciones de apilar/desapilar

- Secuencia de dos acciones:
  1- Movimiento de datos Reg-Mem ó Mem- Reg
  2- Modificación del puntero antes/después de la anterior

- Tener en cuenta:
  - dónde apunta el puntero
  - cómo crece la pila

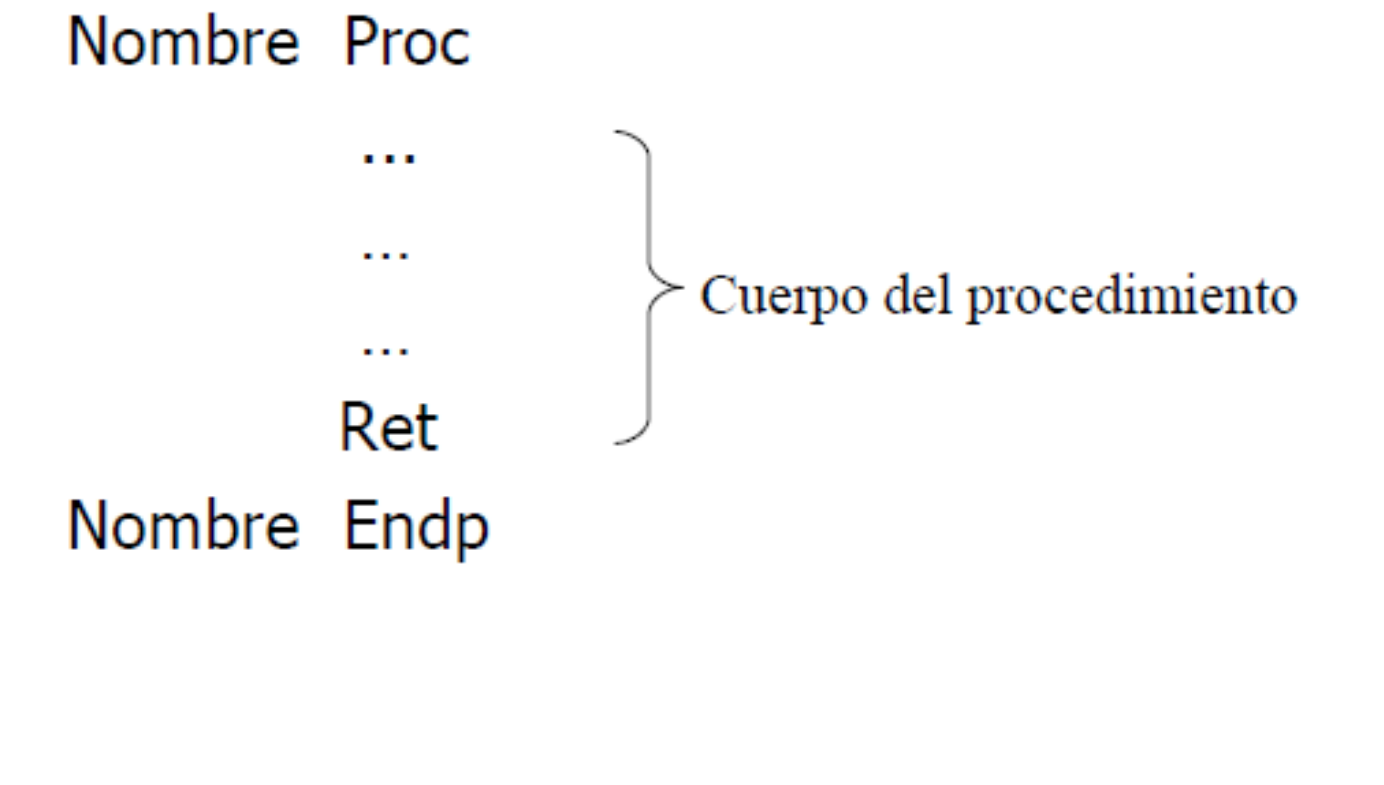
Funcionamiento de la pila



Ejemplo

```
Ejemplo:    ORG    2000H
            MOV    BX, 3000H
            MOV    AX, [BX]
            PUSH   AX
            MOV    BX, 3002H
            MOV    CX, [BX]
            PUSH   CX
            POP    AX
            POP    CX
            HLT
            ORG    3000H
datos       DB     55h, 33h, 44h, 22h
            END
```

Definición del procedimiento



Llamada al procedimiento

En programa principal

```
        ...
Push Parametro 1
Push Parametro 2
Call  Nombre
        ...
        ...
```

Ejemplo con subrutina

```
            ORG 1000H
subrutina:  NEG     AX
            RET

            ORG 2000H
            MOV     BX, 0
            MOV     AX, dato
            PUSH    AX
            CALL    subrutina
            POP     BX
            HLT
            ORG 3000H
dato:       DB      55H
            END
```

Analizar la pila y los valores finales de AX y BX

Posibles pasos en un procedimiento

1. Salvar el estado de BP (viejo BP)
2. Salvar estado de SP (BP=SP)
3. Reservar espacio para datos locales (opcional)
4. Salvar valores de otros registros (opcional)
5. Acceder a parámetros
6. Escribir sentencias a ejecutar
7. Retornar parámetro (opcional)
8. Regresar correctamente del procedimiento

Para el simulador

- Declaración del procedimiento

  nombre: instrucción

  .

  .

- En lugar de BP se usa BX

Bibliografía e información

- Organización y Arquitectura de Computadoras. W. Stallings, 5ta Ed.

  Repertorios de instrucciones
  - Capítulo 9: características y funciones
  - Capítulo 10: modos de direccionamiento y formatos
  - Apéndice 9A: Pilas

  Ciclo de instrucción:
  - Capítulo 3 apartado 3.2.
  - Capítulo 11 apartados 11.1. y 11.3.

  Organización de los registros
  - Capítulo 11 apartado 11.2.

  Formatos de instrucciones
  - Capítulo 10 apartado 10.3. y 10.4.
- Simulador MSX88

Link de interés: www.williamstallings.com