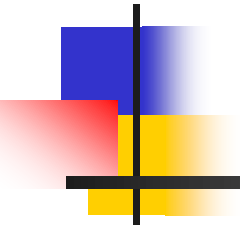
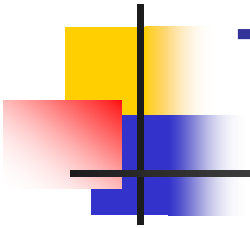
Organización de Computadoras



Clase 7



Temas de Clase

- Formatos de instrucción
- Modos de direccionamiento

Elementos de una instrucción de máquina

- Código de operación
  - especifica la operación a realizar (ej. suma).
  - es un código binario.
- Referencia del operando fuente
  - Establece dónde se encuentra el operando.
  - la operación puede involucrar uno ó más operando fuente (o de entrada).

Elementos de una instrucción de máquina (2)

- Referencia del operando resultado
  - establece dónde almacenar el resultado
- Referencia de la siguiente instrucción
  - le dice a la CPU donde buscar la siguiente instrucción después de la ejecución de la instrucción anterior.
  - en la mayoría de los casos se ubica a continuación de la instrucción actual.

Elementos de una instrucción de máquina (3)

- Los operandos fuente y resultado pueden estar en tres lugares :
  - Memoria
  - Registro de la CPU
  - Dispositivo de E/S

Representación de instrucciones

- Dentro de la computadora cada instrucción está representada mediante una secuencia de bits
- La secuencia se divide en campos en correspondencia a los elementos que la componen.
- Este esquema se conoce como *formato de la instrucción*.

Representación de instrucciones (2)

- Es difícil para el programador tratar con las representaciones binarias de las instrucciones de máquina. Por lo tanto, se usa una *representación simbólica.*
- Los códigos de operación se representan por medio de abreviaturas, llamadas *mnemónicos* que indican la operación.

Representación de instrucciones (3)

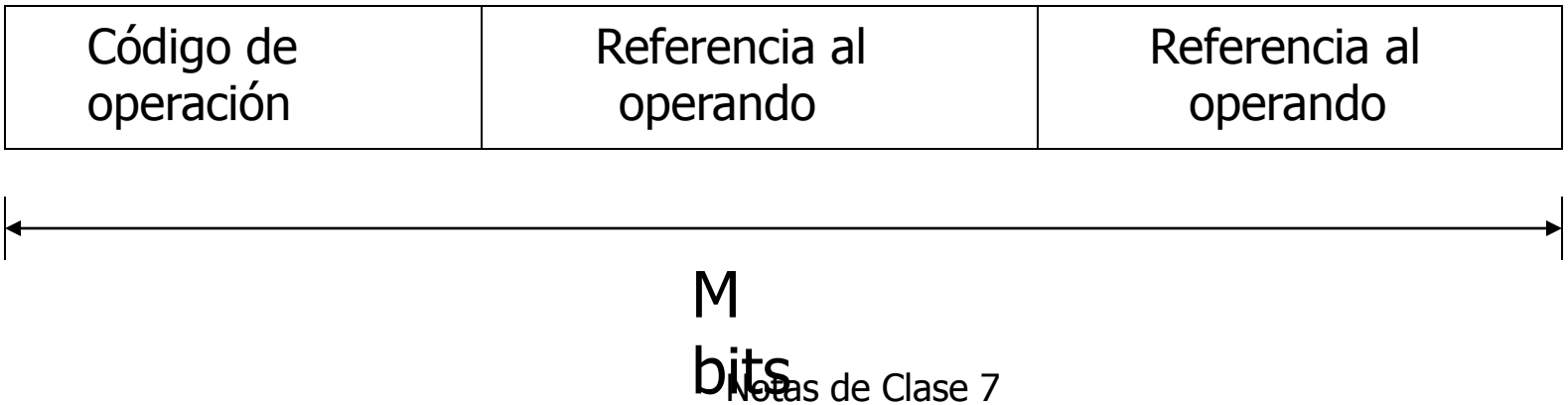
- Los ejemplos más comunes son:
  (algunos ya los vimos en el Ingreso)
  - ADD adición (suma)
  - SUB sustracción (resta)
  - MOV movimiento de datos
  - AND, OR, XOR operaciones lógicas

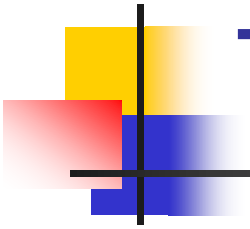
Representación de instrucciones (4)

- Los operandos también se pueden representar de manera simbólica.

Ej: MOV reg1 , memoY

- instrucción que copia el valor contenido en la posición de memoria llamada memoY, a un registro denominado reg1.





Tipos de instrucciones

- En lenguajes de alto nivel escribimos:

  X := X + Y

  - Esta instrucción suma los valores almacenados en las posiciones de memoria X e Y.
  - Esto puede implicar cargar registros, sumarlos y luego almacenar el resultado en memoria.

Tipos de instrucciones (2)

- Una instrucción de alto nivel puede requerir varias instrucciones de máquina.
- El lenguaje de alto nivel expresa operaciones en forma "concisa" usando variables.
- El lenguaje de máquina expresa las operaciones en forma "básica" involucrando movimiento de datos y uso de registros.

Tipos de instrucciones (3)

- Cualquier programa escrito en lenguaje de alto nivel se debe convertir a un lenguaje de máquina para ser ejecutado.
- El conjunto de instrucciones de máquina debe ser capaz de expresar cualquiera de las instrucciones de un lenguaje de alto nivel.

Tipos de instrucciones (4)

- Podemos categorizar las instrucciones de máquina como de:
  - Procesamiento de datos
    - operaciones aritméticas y lógicas.
  - Almacenamiento de datos
    - transferencias dentro del sistema.
  - Instrucciones de E/S
    - transferencia de datos entre la computadora y los mecanismos externos.
  - Control

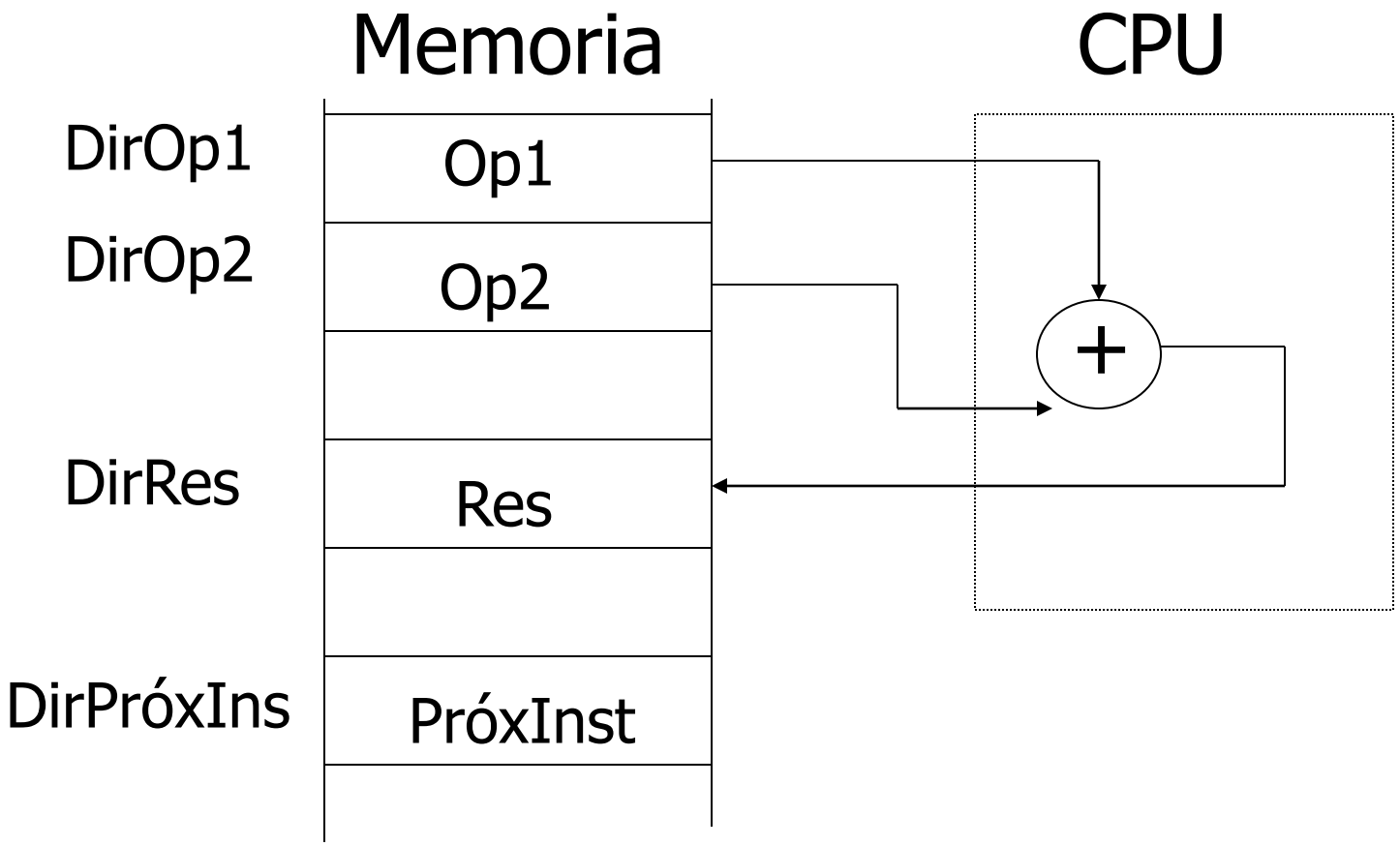
Número de direcciones

- ¿Cuántas direcciones se necesitan?
- Dos direcciones para hacer referencia a los operandos, una donde almacenar el resultado y la dirección de la próxima instrucción. Por lo tanto necesitaríamos cuatro direcciones

Add DirRes, DirOp1, DirOp2, DirPróxIns

Add	DirRes	DirOp1	DirOp2	DirPróxIns

Máquina para 4 direcciones



Máquina para 4 direcciones (2)

- ✓ Direcciones explícitas para operandos, resultado y próxima instrucción.
- ✓ Son "raras", cada campo de dirección tiene que tener bits para "acomodar" una dirección completa.
- ✓ Ej. si dirección = 24 bits, la instrucción tiene 96 bits de referencias.

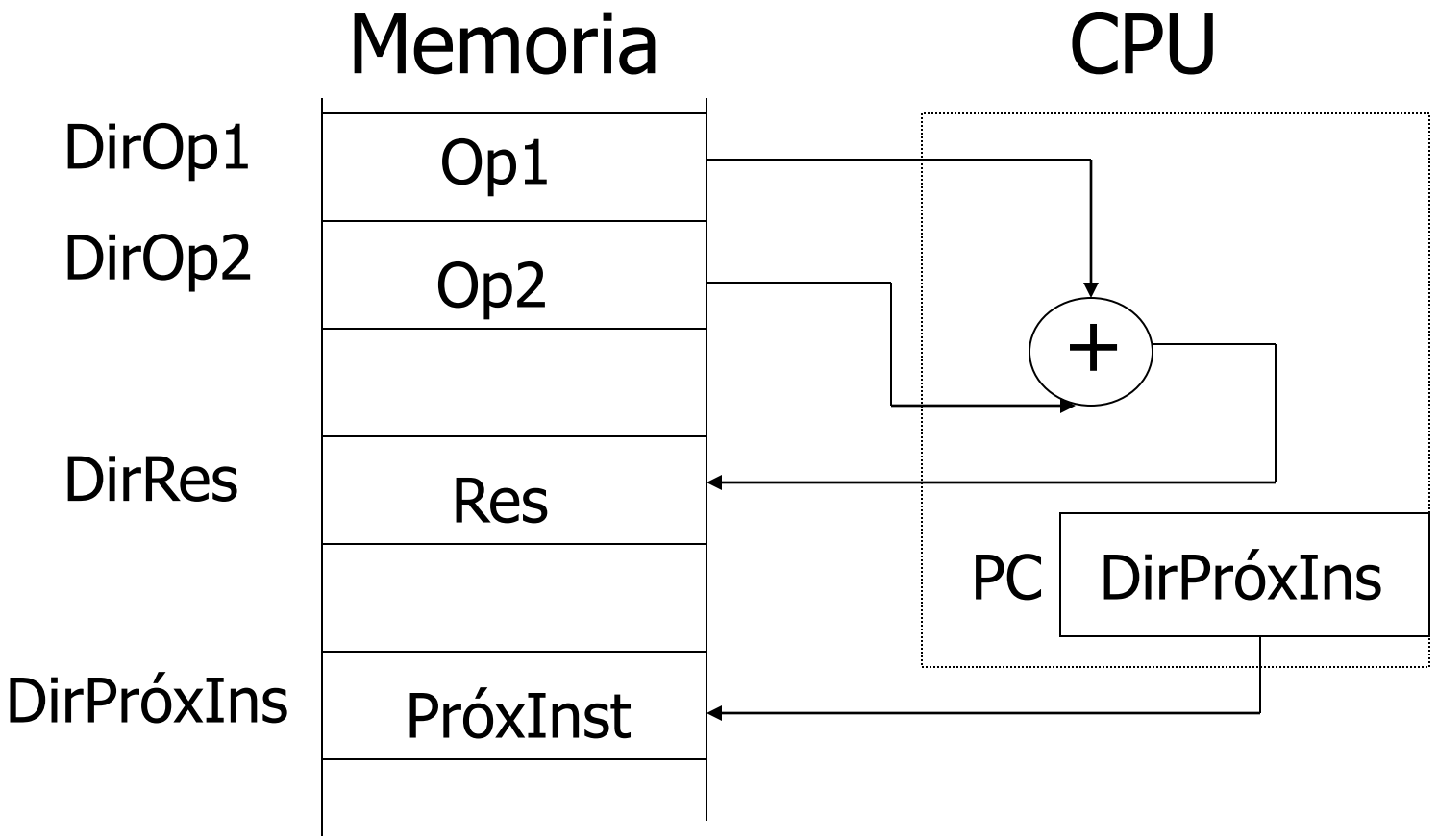
Máquina para 3 direcciones

Add DirRes, DirOp1, DirOp2

Add	DirRes	DirOp1	DirOp2

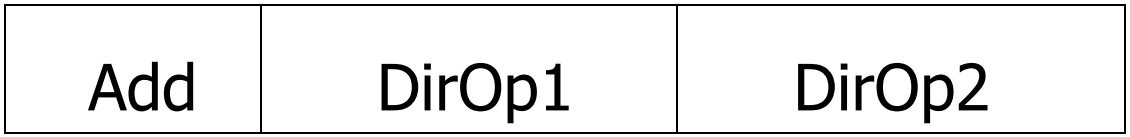
- Dirección de la próxima instrucción está almacenada en un registro de la CPU, llamado Contador de Programa PC.
- Referencias = 72 bits. Todavía larga.

Máquina para 3 direcciones (2)



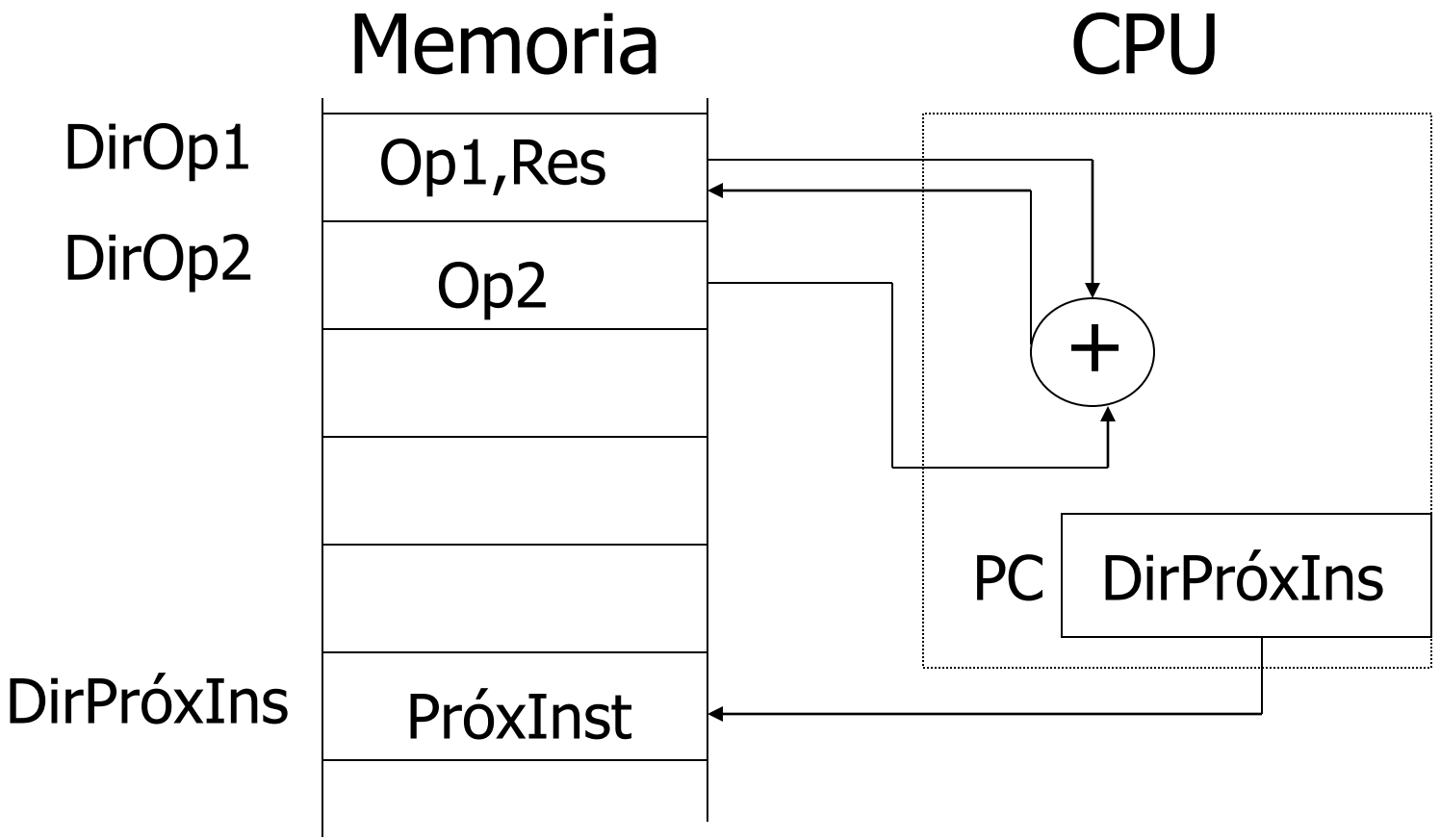
Máquina para 2 direcciones

Add DirOp1, DirOp2



- Reduce el tamaño de la instrucción.
  - 48 bits de referencias.
- Hay que mover el Op1 a un registro temporal.
- Menos elección donde guardar el resultado.

Máquina para 2 direcciones (2)

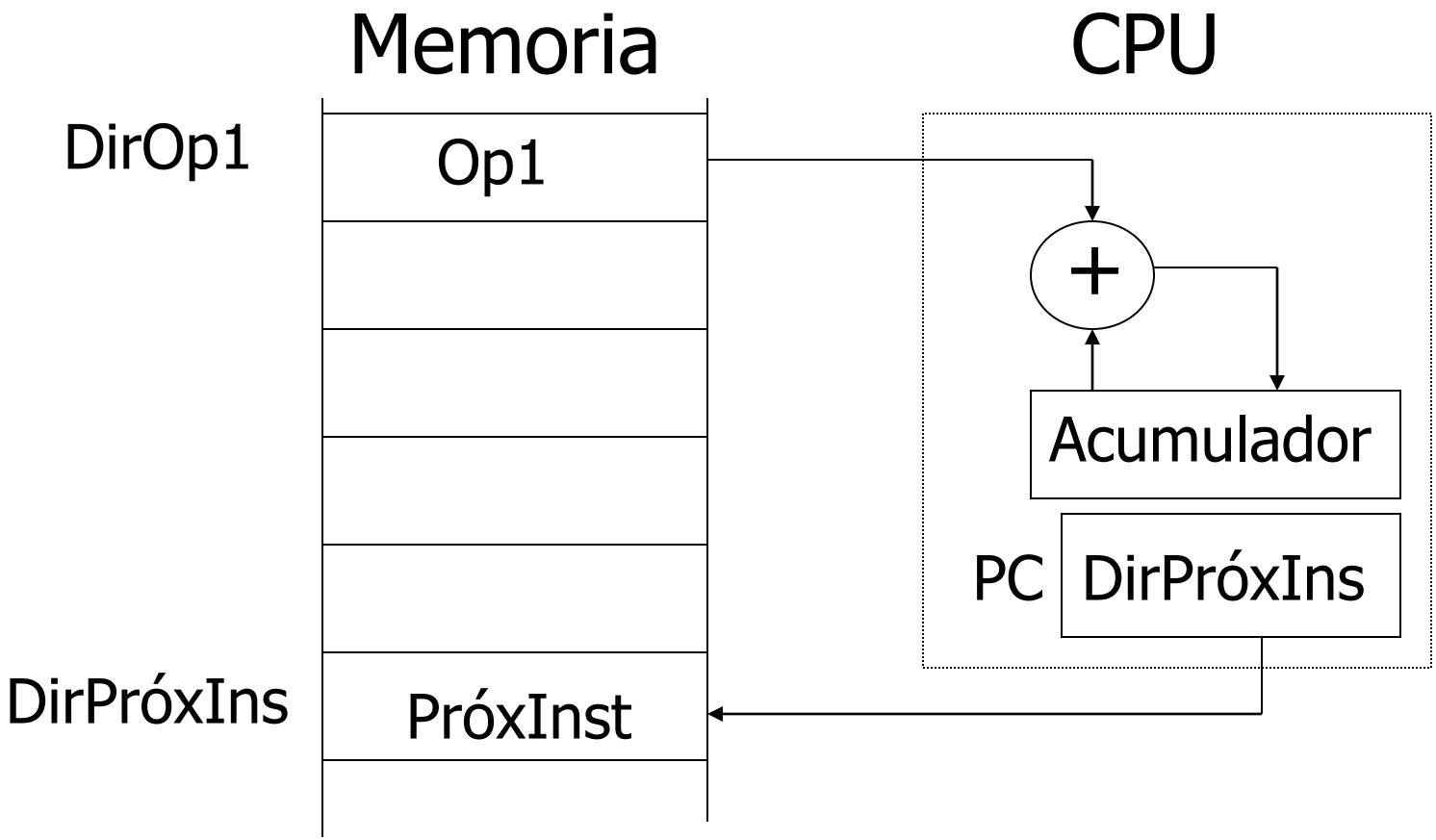


Máquina para 1 dirección

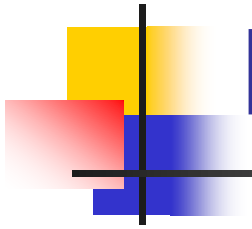
Add DirOp1

- Registros especiales en la CPU (acumulador)
- Instrucciones para cargar y descargar el acumulador.
- Un operando y resultado en lugar predefinido
- Instrucción más corta (24 bits de referencias)

Máquina para 1 dirección (2)



Ej. evaluar a=(b+c)*d - e



3 direcciones

```
add a, b, c
mul a, a, d
sub a, a, e
```

3 instruc./3 acc. MI
9 acc. MD

2 direcciones

```
mov a, b
add a, c
mul a, d
sub a, e
```

4 instruc./4 acc. MI
11 acc. MD

1 dirección

```
load  b
add   c
mul   d
sub   e
store a
```

5 instruc./5 acc. MI
5 acc. MD

Diseño del conjunto de instrucciones (1)

- El conjunto de instrucciones es el medio que tiene el programador para controlar la CPU.
- Hay que tener en cuenta:
  - Tipos de operaciones
    - cuántas y cuáles
  - Tipos de datos
    - cuáles

Diseño del conjunto de instrucciones (2)

- Formato de instrucciones
  - longitud (bits), Nº de direcciones, tamaño de cada campo
- Registros
  - cantidad que se pueden referenciar mediante instrucciones y su uso
- Direccionamiento
  - la manera de especificar la dirección de un operando o una instrucción (la próxima).

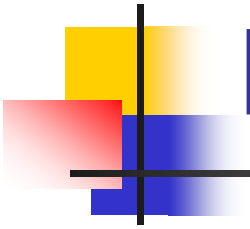
Tipos de operaciones

- ✓ Transferencia de datos: Mov (load/store)
- ✓ Aritméticas: Add, Sub, Inc, Dec, Mul
- ✓ Lógicas: And, Or, Xor, Not
- ✓ Conversión
- ✓ E/S: In, Out
- ✓ Transferencia de control: salto, bifurcación
- ✓ Control del sistema: usadas por S.O.

Tipos de datos

Los más importantes:

- Direcciones
- Números: enteros, p. fijo, p. flotante
- Caracteres: ASCII, BCD.
- Datos lógicos



Modos de direccionamiento

- Como vimos, en una instrucción se utilizan bits para expresar el código de operación: nos dice qué hacer. También se necesitan una "gran" cantidad de bits para especificar de dónde provienen los datos.
- ¿Cómo podemos reducir el tamaño de estas especificaciones?

Modos de direccionamiento (2)

Hay 2 métodos generales:

1. Si un operando va a usarse varias veces puede colocarse en un registro.
   - Usar registro para una variable tiene 2 ventajas
     - el acceso es más rápido
     - se necesitan menos bits.

     Ej. si hay 32 reg. se necesitan 5 bits para especificar c/u de ellos (menos bits que las dir. de mem.).

Modos de direccionamiento (3)

2. Especificar uno ó más operandos en forma implícita.

   Ejemplos: reg2 = reg2 + fuente1 ; el acumulador.

Los mdd tienen como objetivo:

- ✓ disminuir la cantidad de bits en la instrucción
- ✓ la dirección puede que no se conozca hasta el momento de ejecutar el programa
- ✓ manejo más eficiente de datos (arreglos)

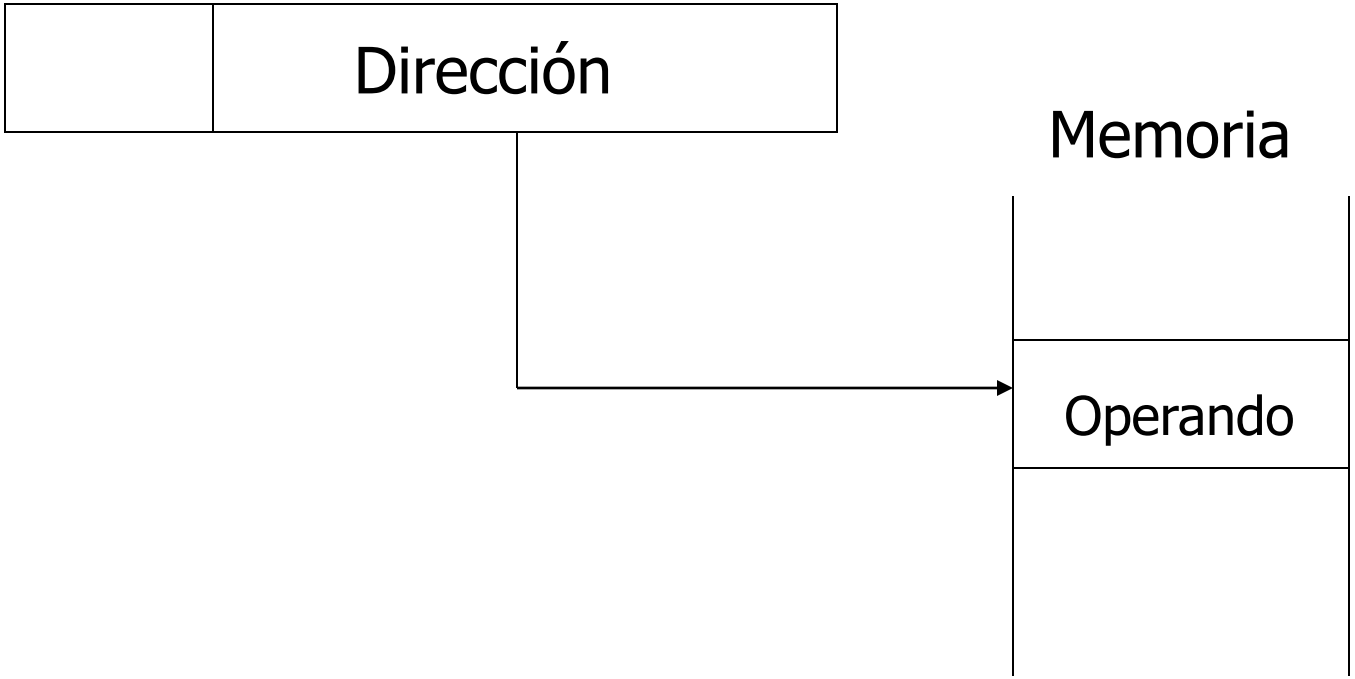
Modos de direccionamiento (4)

- Inmediato
- Directo
- Por registro
- Indirecto por memoria
- Indirecto por registro
- Por desplazamiento
- Del stack

Mdd Inmediato

- El operando se obtiene automáticamente de la memoria al mismo tiempo que la instrucción.
- No requiere una referencia extra a memoria de datos
- Se utiliza para definir constantes y para inicializar variables.
- Desventaja: tamaño del operando limitado por el tamaño del campo de direccionamiento.

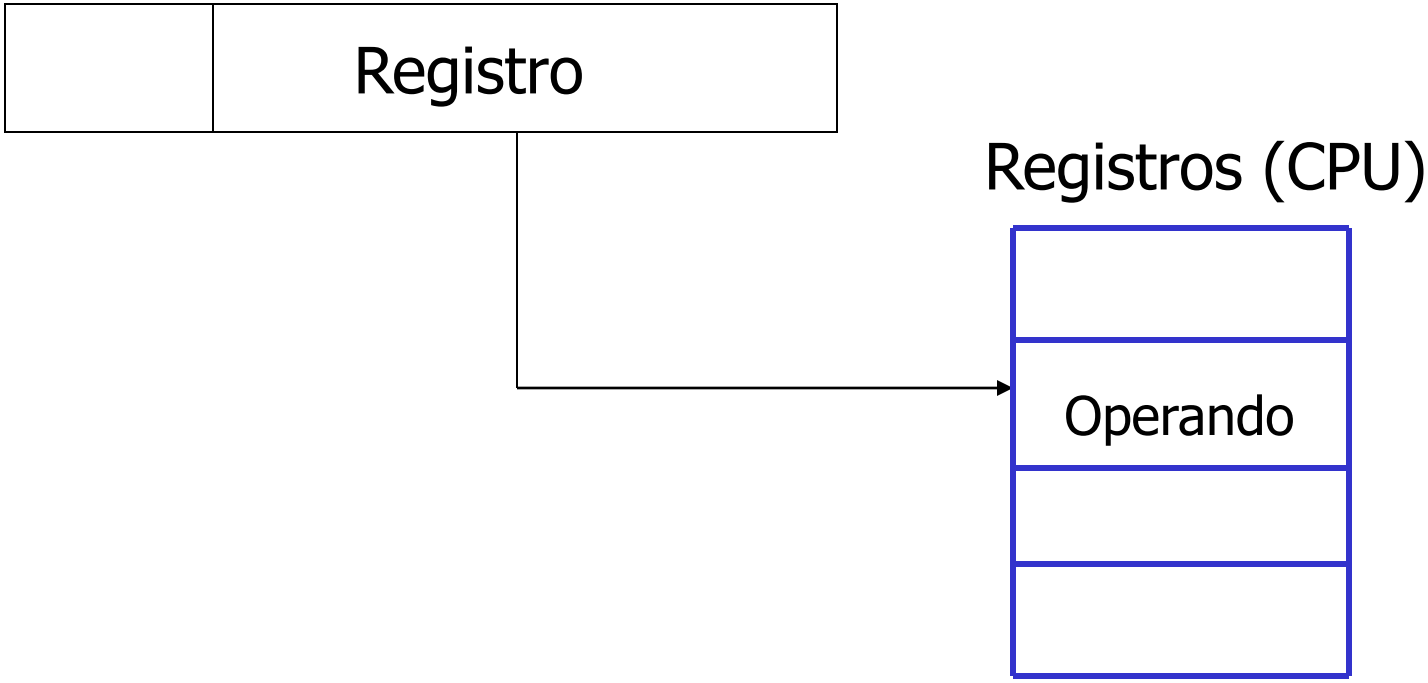
Mdd Directo

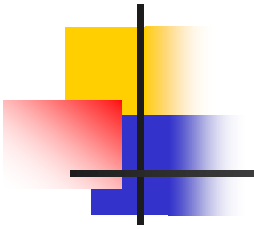


Directo (2)

- El campo de dirección tiene la dirección efectiva del operando.
- Es simple, pero tiene un espacio limitado de direcciones por cantidad de bits del campo.
- Uso: acceder a variables globales, cuya dirección se conoce en el momento de compilación.

Mdd Por registro

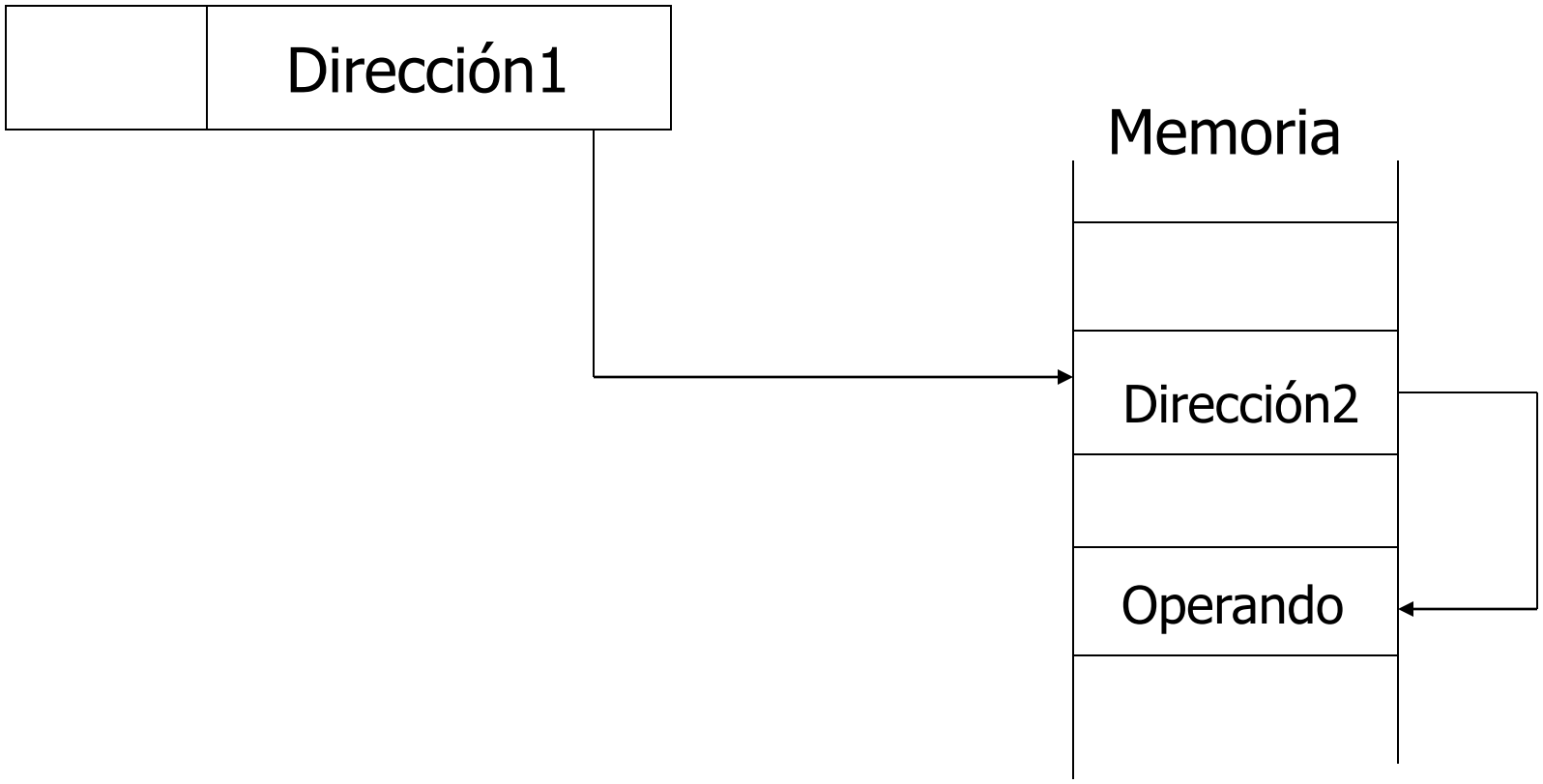




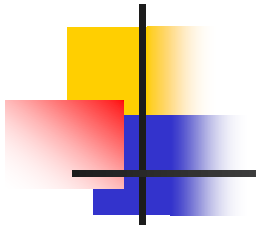
Por registro (2)

- Conceptualmente igual al directo, pero se especifica un registro en lugar de una posición de memoria.
- La referencia a registro usa menos bits que la especificación de la dirección y no requiere acceso a memoria de datos.
- Desventaja: los registros no son muchos y es un recurso preciado.

Mdd Indirecto por memoria

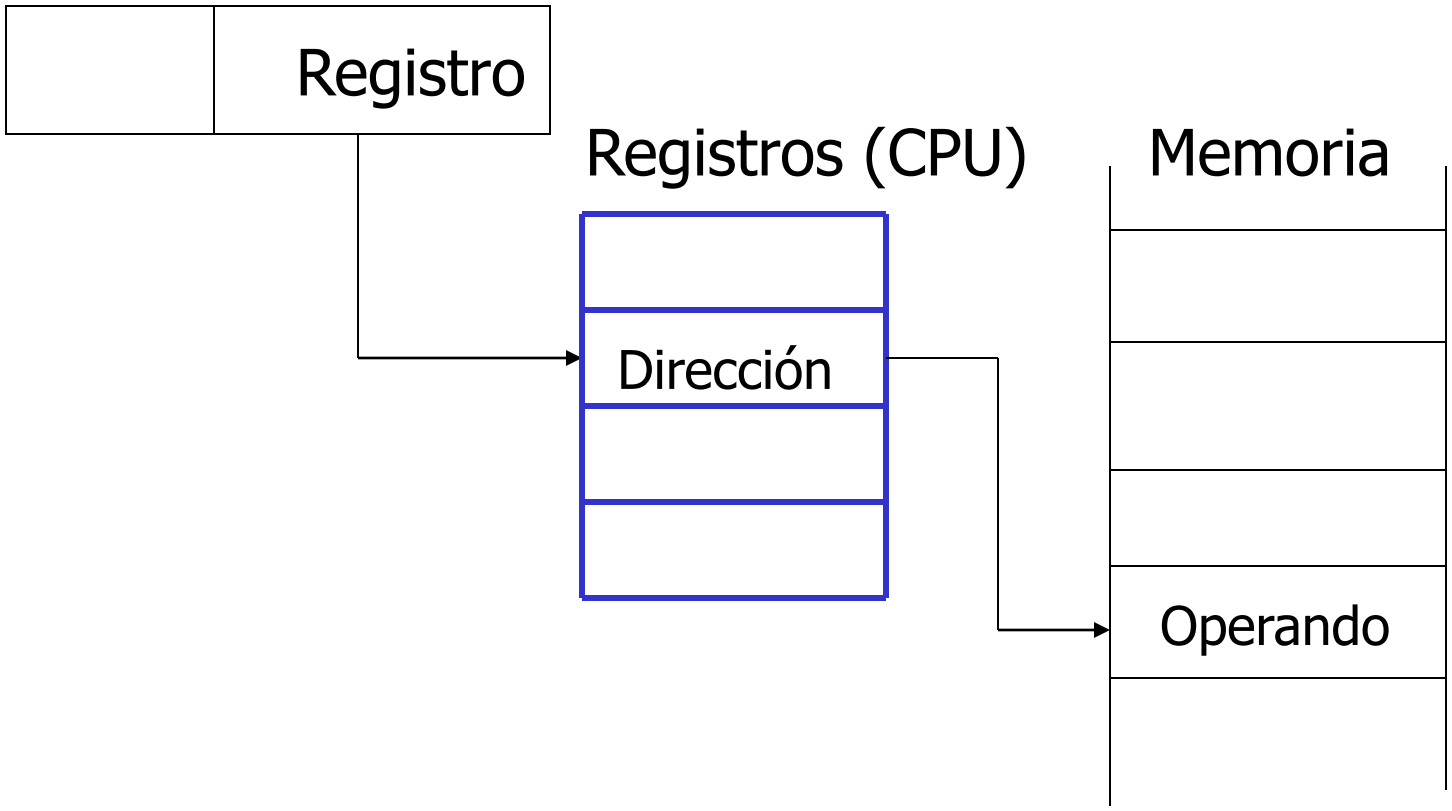


Indirecto por memoria (2)

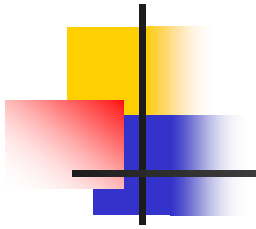


- En la instrucción está la dirección de la dirección del operando. Trata de solucionar el problema del directo. Así, con una dirección de menos bits en la instrucción, se apunta a una dirección de más bits.
- Ventaja: espacio de direccionamiento mayor
- Desventaja: múltiples accesos a memoria.

Mdd Indirecto por registro

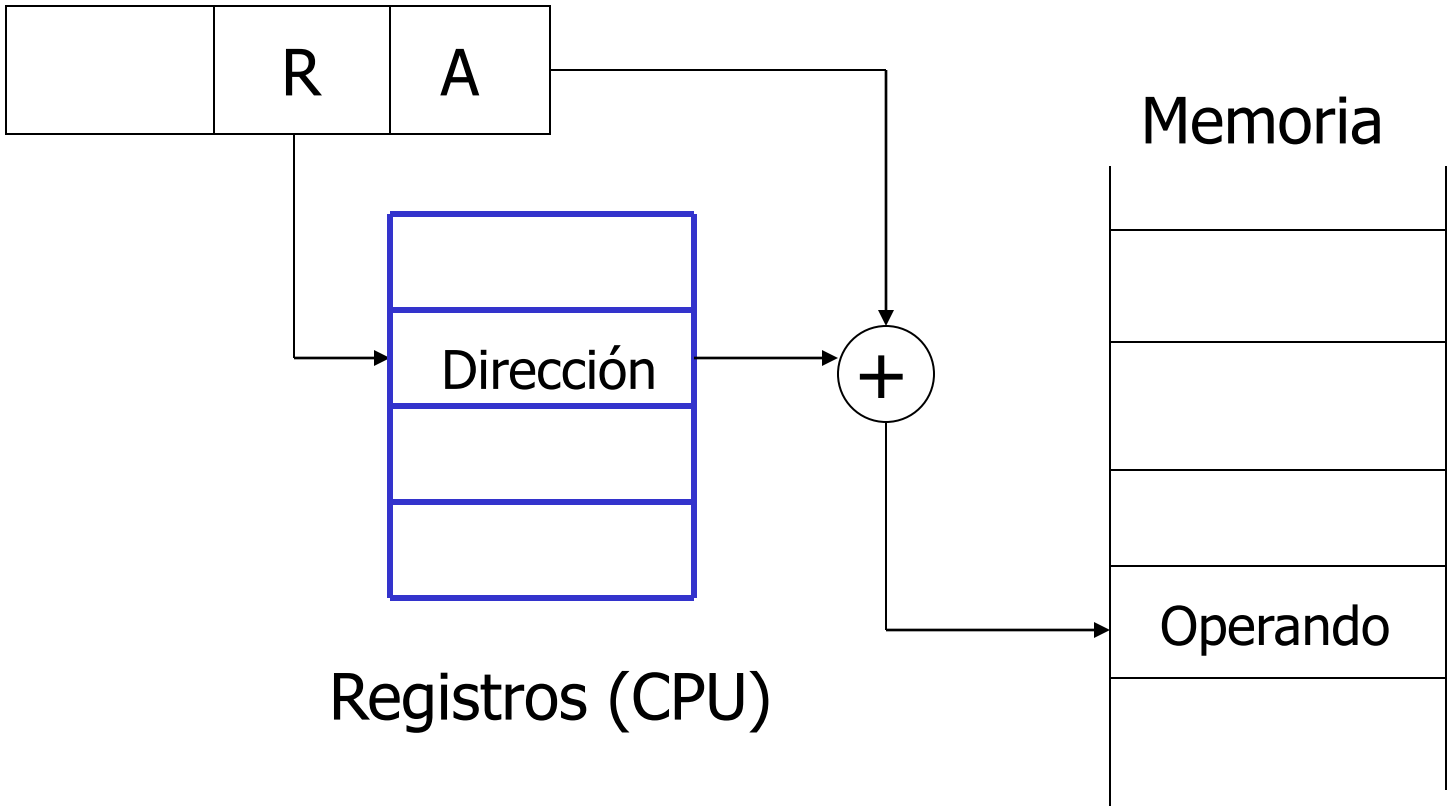


Indirecto por registro (2)



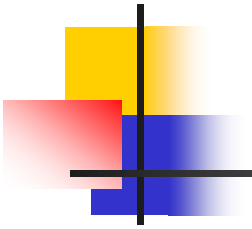
- En la instrucción se especifica el registro que tiene almacenada la dirección.
- Ventaja: menos bits para especificar el registro que la posición de memoria. Espacio de direccionamiento grande, accede una vez menos a memoria que el indirecto. La dirección así usada se llama apuntador.

Mdd Por desplazamiento



Por desplazamiento (2)

- Combina capacidades de indirecto y directo. Requiere que la instrucción tenga dos campos de dirección. Estos dos campos se suman para producir la dirección efectiva. Los más comunes:
  - Relativo
  - De registro base
  - Indexado



Relativo

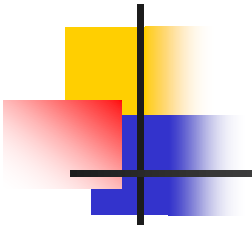
- El registro referenciado de manera implícita es el contador de programa PC.
- La dirección de la instrucción actual se suma al campo de dirección para producir la dirección efectiva.
  - El campo de dirección se trata como un número en Ca2.

De registro base

- El registro referenciado contiene una dirección de memoria y el campo de dirección tiene un desplazamiento.

Indexado

- Se direcciona la memoria con un registro más un desplazamiento.
  - Es "igual" al anterior pero se intercambian los papeles del registro y del desplazamiento.
- La Indexación proporciona un mecanismo eficiente para realizar operaciones iterativas.
- Se utiliza un registro llamado índice
  - algunas máquinas incrementan ó decrementan este registro como parte de la instrucción (Autoindexación)



Del stack

- El stack ó pila es un arreglo lineal de localidades de memoria. Es una lista ó cola donde el último en entrar es el primero en salir. Es una zona de memoria reservada.
- Asociado con la pila o stack hay un registro apuntador (o registro puntero de pila), cuyo valor es la dirección tope de pila o stack.

MSX88: inst. 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.

Inst. aritméticas y 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.*

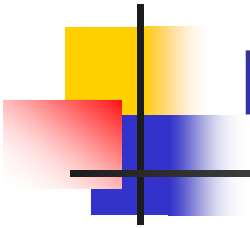
5. *dest* solo puede ser *mem* o *reg.*

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

Inst. transf. 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*.



mas información …

Repertorios de instrucciones

- Capítulo 9: características y funciones
- Capítulo 10: modos de direccionamiento y formatos
- Apéndice 9A: Pilas
  - Stallings, W., 5º Ed.

- Lenguaje Assembly
  - Apunte 4 de cátedra
- Simulador MSX88
  - En Descargas de página web de cátedra