Uso Personal

Organización de Computadoras

CURSO 2021

TURNO RECURSANTES

CLASE 6 – FORMATO DE INSTRUCCIÓN – MODOS DE DIRECCIONAMIENTO

Resumen de clase 6

- Elementos de una instrucción
- Representación de las instrucciones: lenguajes de máquina, Assembly y alto nivel
- Formato de instrucción: máquinas de 4, 3, 2, 1 y 0 dirección
- Modos de direccionamiento
- Consideraciones en el diseño del repertorio de instrucciones
- Repertorio de instrucciones del MSX88

Elementos de una instrucción de máquina

- Una instrucción de máquina es un código binario compuesto por una cantidad determinada de bits.
- Los bits son usados por la Unidad de Control (CU) para continuar y completar el proceso de ejecución de la instrucción.
- Para ello, el código debe contener suficiente “información” para que la Unidad de Control pueda resolverla.
- Las instrucciones tienen distinto grado de complejidad. Van desde las que no hacen nada, hasta las que operan sobre múltiples datos.

Elementos de una instrucción de máquina

- En general las instrucciones más complejas requieren de múltiples pasos o acciones. Las más sencillas se resuelven en pocos pasos.
- Una ejemplo de instrucción bastante “compleja” es la que busca 2 operandos y produce un resultado. Para esta instrucción la CPU necesita definir:
  - 1: tipo de operación
  - 2: Lugar del 1er operando (donde está)
  - 3: Lugar del 2do operando (donde está)
  - 4: Lugar del resultado (donde guardarlo)
  - 5: donde está la próxima instrucción

Uso Personal

Elementos de una instrucción de máquina

- Dado que hay diferentes datos que proveer, la instrucción está organizada en conjuntos de bits, comúnmente denominados campos, que proveerán esos datos.
- La cantidad de bits de un campo dependen de la cantidad de información que debe proveer.
- Por ejemplo, si un procesador dispone de 256 tipos de instrucciones, y un campo se usa para especificar el tipo de instrucción (campo de Código de Operación COP), entonces ese campo requiere de al menos 8 bits.

Uso Personal

Elementos de una instrucción de máquina

Los campos que pueden estar incluidos en una instrucción son:

- 1.- Identificación de la instrucción (Código de operación):

  Especifica la operación a realizar (ej. suma).

- 2.- Referencias a los operandos:
  - Establece la información relativa a el o los operandos fuentes (si es que se requieren). La información relativa puede ser de distinto tipo (se verá con los modos de direccionamiento).
  - La operación puede involucrar uno ó más operando fuente (o de entrada). Por ejemplo en una suma se requiere especificar 2 operandos.

Uso Personal

Elementos de una instrucción de máquina

- 3.- Referencia del resultado:
  - Establece dónde almacenar el resultado.
  - La operación puede involucrar uno o más resultados (de salida). Por ejemplo, en una suma se produce 1 resultado solamente.
- 4.- Referencia de la siguiente instrucción
  - Provee a la CPU con información para determinar donde buscar la siguiente instrucción después de la ejecución de la instrucción anterior.
  - En la mayoría de los casos, la próxima instrucción se ubica a continuación de la instrucción actual, es decir, en la siguiente posición de memoria. Esta situación se conoce como acceso secuencial a instrucciones consecutivas de memoria.

Uso Personal

Elementos de una instrucción de máquina

Respecto de los puntos 2 y 3, los operandos (fuente) y el resultado (destino) pueden residir en tres lugares :

- Memoria
- Registro de la CPU
- Dispositivo de E/S

Representación de instrucciones – Lenguaje de máquina

Lenguaje de máquina o absoluto

- El lenguaje absoluto o de máquina es el binario.
- Las instrucciones están codificadas mediante un conjunto de bits almacenados en palabras de memoria (1 palabra o más).
- El conjunto de bits que forma una instrucción se puede considerar organizado en campos, en correspondencia con la información que proveen.
- La representación de las instrucciones en memoria es puramente binaria.
- Se pueden emplear otras representaciones numéricas para facilitar su escritura (BCH, BCD, etc.).

Uso Personal

Representación de instrucciones – Lenguaje Assembly

Lenguaje simbólico (Assembly)

- Dado que es prácticamente imposible para el programador, tratar con las representaciones binarias de las instrucciones de máquina, se usan otras representaciones más "amigables".
- Estas representaciones tienen distintos grados o niveles de abstracción.
- En el nivel siguiente al nivel de lenguaje de máquina está la Representación simbólica (Lenguaje Assembly).
- En la representación simbólica, los campos que forman la instrucción se pueden representar con textos o números.

Representación de instrucciones – Lenguaje Assembly

Lenguaje simbólico (Assembly)

- Por ejemplo, el campo binario de la instrucción que contiene la información del tipo de instrucción (campo de código de operación COP) puede ser reemplazado, en el lenguaje simbólico (Assembly) por mnemónicos más sencillos de memorizar:
  - ADD (significa código de operación de) adición (suma)
  - SUB sustracción (resta)
  - MOV movimiento de datos
  - AND, OR, XOR operaciones lógicas

Uso Personal

Representación de instrucciones - Lenguaje Assembly

- Los operandos también se pueden representar de manera simbólica.

Por ejemplo, una instrucción que mueve un dato desde una posición de memoria memoY a un registro reg1 se puede escribir así:

MOV reg1 , memoY

En el ejemplo anterior se ha referenciado la posición de memoria mediante un texto simbólico memoY, y el registro mediante el texto reg1. En realidad, estas referencias simbólicas representan los campos de bits de la instrucción de máquina siguiente:

Uso Personal

Instrucciones de lenguajes de alto nivel (HLL) y de máquina

Lenguaje de alto nivel (HLL)

- En lenguajes de alto nivel las sentencias se escriben de la siguiente manera:

  X := X + Y

- Esta sentencia suma 2 variables X e Y, y el resultado se guarda en una de ellas X.
- El lenguaje no referencia el lugar físico de las variables, solo su identificación.

Uso Personal

Instrucciones de lenguajes de alto nivel (HLL) y de máquina

Lenguaje de alto nivel (HLL)

- Las sentencias en un lenguaje de alto nivel, expresan acciones en forma “concisa”.
- Las instrucciones en el lenguaje de máquina expresan acciones en forma “básica”.
- Por lo tanto, una sentencia de alto nivel requerirá de varias instrucciones de máquina para su resolución.

Uso Personal

Instrucciones de lenguajes de alto nivel (HLL) y de máquina

Lenguaje de alto nivel (HLL)

- Dado que la máquina solo “entiende” (es capaz de resolver) las instrucciones de máquina, los programa escritos en lenguajes de alto nivel se deben convertir (compilar) al lenguaje de máquina para ser ejecutados.
- El conjunto de instrucciones de máquina debe ser capaz de expresar (resolver) cualquiera de las instrucciones de un lenguaje de alto nivel.

Uso Personal

Instrucciones de lenguajes de alto nivel (HLL) y de máquina

Gap semántico

- Hay un factor de relación entre la sentencia de alto nivel y la cantidad de instrucciones de máquina que la resuelve.
- Este factor se calcula como la cantidad de instrucciones de máquina que se requieren para resolver una sentencia de un HLL.
- Esta relación se conoce como "Gap semántico"
- Este Gap depende del lenguaje de alto nivel, y del conjunto de instrucciones de máquina (y también del compilador).

Uso Personal

Formato de instrucción

Para entender la estructura de una instrucción, es necesario analizar el tipo de información que referencia cada campo de la misma.

1.- Referencias a operandos

Los operandos pueden estar en:

- Un registro, en cuyo caso la referencia en la instrucción será un número de registro
- Una posición de memoria, en cuyo caso la referencia será una dirección de memoria

2.- Referencias a resultados

Son iguales a las referencias a operandos, es decir un registro o una posición de memoria.

Uso Personal

Formato de instrucción

3.- Referencias a la próxima instrucción a ejecutar

- Una posición de memoria, en cuyo caso la referencia será una dirección de memoria

La estrategia de organización de los campos de una instrucción y la información contenida en ellos se conoce como formato de instrucción.

Uso Personal

Taxonomía basada en formato de instrucción

- Una forma de clasificación de las máquinas está basado en el formato de instrucción.
- La clasificación identifica la cantidad de campos que contienen direcciones de memoria.
- Suponiendo el caso de una instrucción de suma de 2 operandos en memoria, y resultado almacenado en memoria, se necesitan:
  - 2 campos de direcciones para hacer referencia a los operandos.
  - 1 campo de dirección para hacer referencia a donde almacenar el resultado.
  - 1 campo de dirección de referencia de la ubicación de la próxima instrucción.

Uso Personal

Formato de instrucción
Máquina de 4 direcciones

- Es una máquina cuya instrucción que requiere 2 operandos y produce 1 resultado (por ejemplo una SUMA), dispone de 4 campos de dirección:

Add	DirRes	DirOp1	DirOp2	DirPróxIns

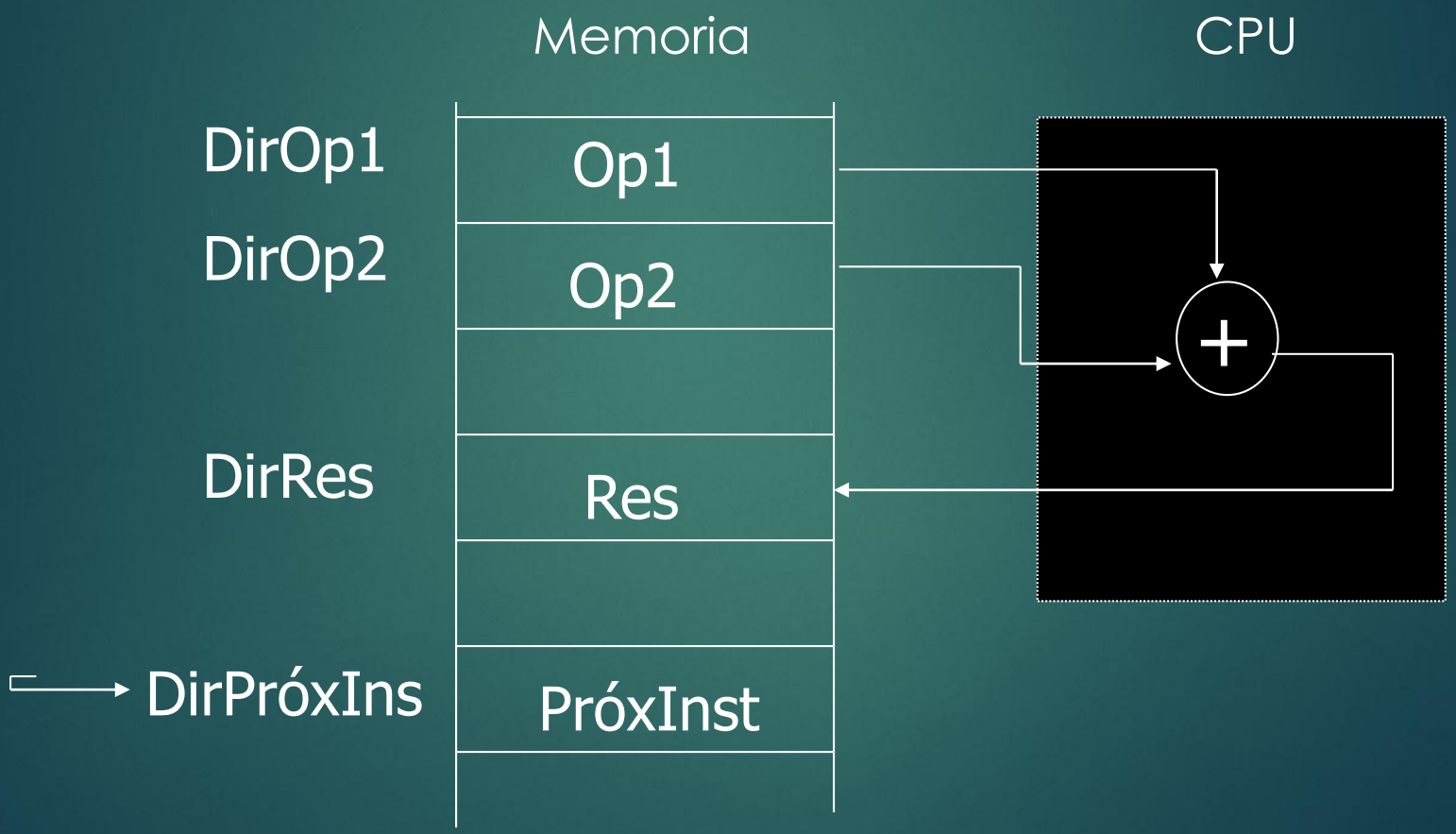
- En lenguaje Assembly, la instrucción debería tener la siguiente forma:

Add DirRes, DirOp1, DirOp2, DirProxIns

Uso Personal

Formato de instrucción
Máquina de 4 direcciones

- La operación de suma se ejecutaría de la siguiente manera:



Formato de instrucción
Máquina de 4 direcciones

Características de la máquina de 4 direcciones:

- Todas las direcciones son explícitamente expresadas: operandos, resultado y próxima instrucción.
- Cada campo de dirección tiene que tener suficientes bits para “acomodar” o referenciar una dirección completa.
- Si, por ejemplo, la dirección requiere de 24 bits, la instrucción tiene: 4 x 24= 96 bits de referencias.

Formato de instrucción

Máquina de 3 direcciones

- La máquina de 4 direcciones requiere de muchísimos bits para especificarla.
- Se puede reducir su tamaño si se elimina el campo que referencia la dirección de la próxima instrucción, y se lo reemplaza por un registro en la CPU (llamado Contador de Programa PC) que cumpla esa función.

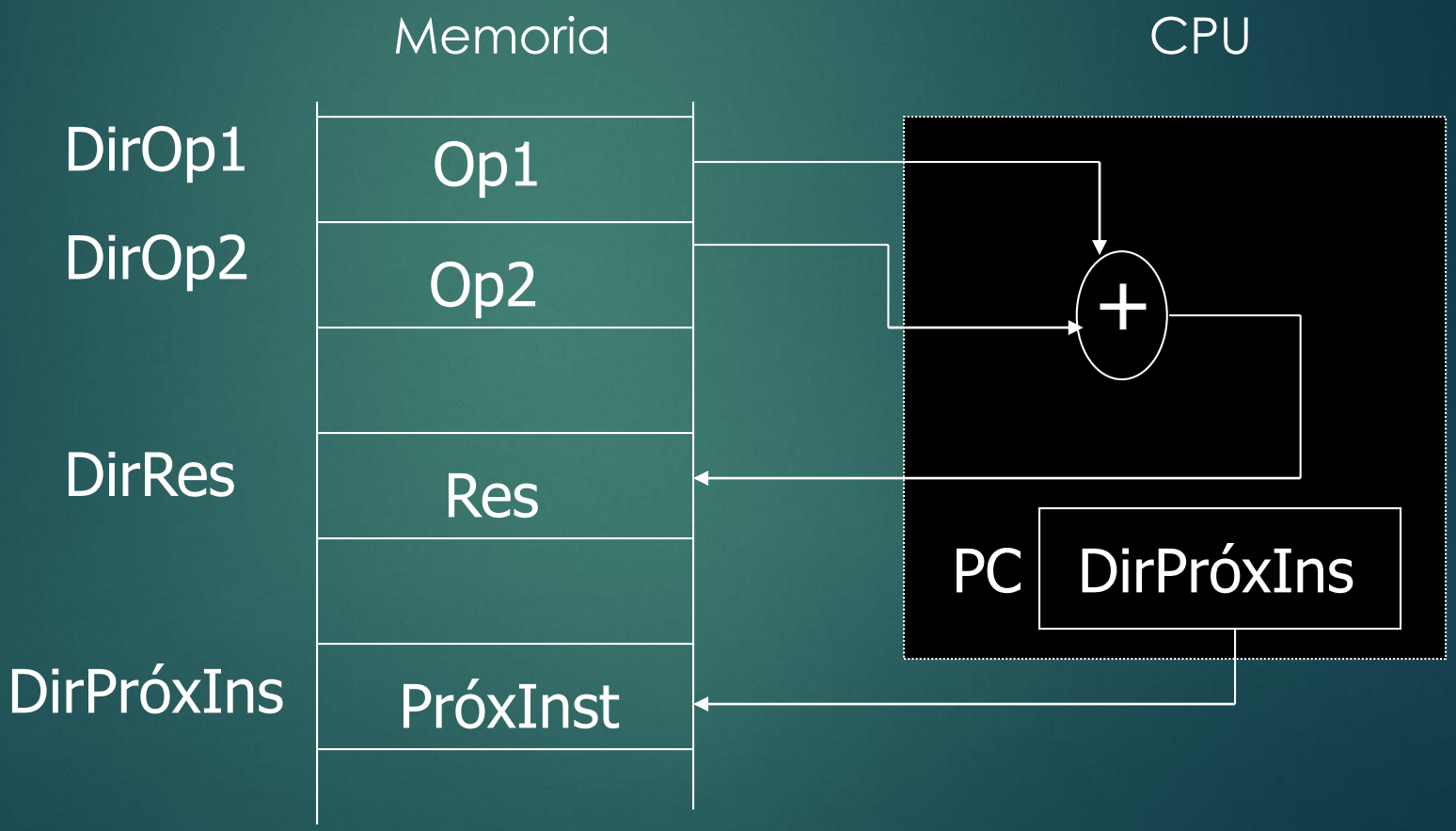
Add	DirRes	DirOp1	DirOp2

- En lenguaje Assembly, la instrucción debería tener la siguiente forma:

Add DirRes, DirOp1, DirOp2

Formato de instrucción
Máquina de 3 direcciones

- La operación de suma se ejecutaría de la siguiente manera:



Formato de instrucción
Máquina de 3 direcciones

Características de la máquina de 3 direcciones:

- Las direcciones de operandos, resultado y próxima instrucción son explícitamente referenciadas.
- Se elimina la referencia a la dirección de la próxima instrucción a ejecutar.
- Se agrega un registro específico para contener la dirección de la próxima instrucción a ejecutar. Este registro se conoce como Contador de programa (o PC: Program Counter).
- Cada campo de dirección tiene que tener suficientes bits para “acomodar” o referenciar una dirección completa.
- Si, por ejemplo, la dirección requiere de 24 bits, la instrucción tiene: 3 x 24= 72 bits de referencias.

Formato de instrucción

Máquina de 2 direcciones

- Se puede reducir aún más el tamaño de la instrucción si se elimina otro campo que referencia una dirección.
- En función del campo que se elimina, se tienen 2 opciones.
- Opción 1: eliminar la referencia al resultado.

Add	DirOp1	DirOp2

En assembly: Add DirOp1, DirOp2

- Opción 2: eliminar la referencia a un operando.

Add	DirRes	DirOp2

En assembly: Add DirRes, DirOp2

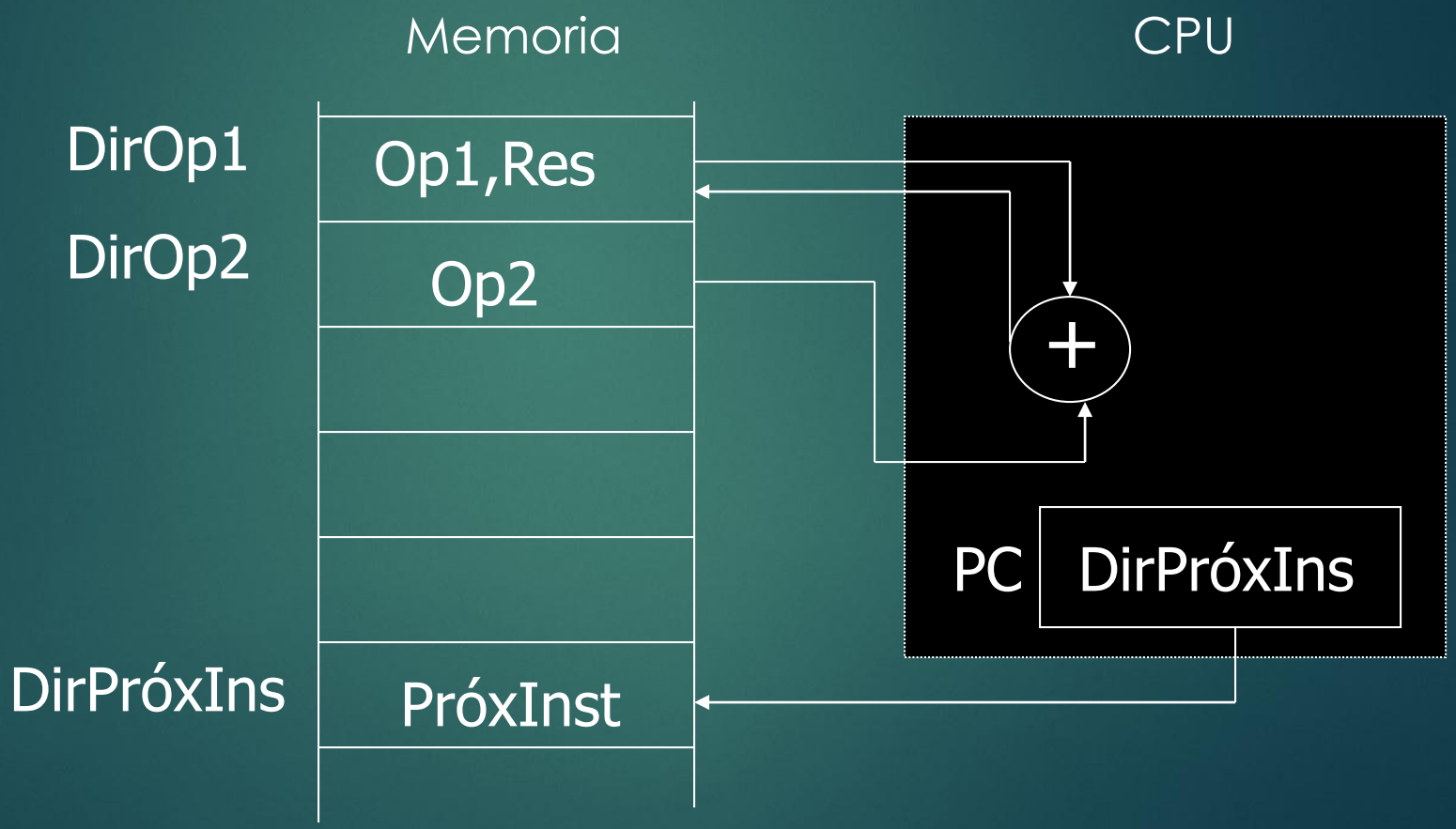
Formato de instrucción
Máquina de 2 direcciones

Características de la máquina de 2 direcciones:

- Las direcciones de los 2 operandos (opción 1), o la de 1 operando y la de resultado (opción 2) son explícitamente referenciadas.
- Cada campo de dirección tiene que tener suficientes bits para “acomodar” o referenciar una dirección completa.
- En la opción 1, el resultado “pisa” (sobrescribe) alguno de los operandos (en memoria o en registro).
- En la opción 2, el Op1 está en algún lugar predeterminado al que se debe mover previamente.
- Si, por ejemplo, la dirección requiere de 24 bits, la instrucción tiene: 2 x 24= 48 bits de referencias.

Formato de instrucción
Máquina de 2 direcciones

- La operación de suma se ejecutaría de la siguiente manera:



Uso Personal

Formato de instrucción

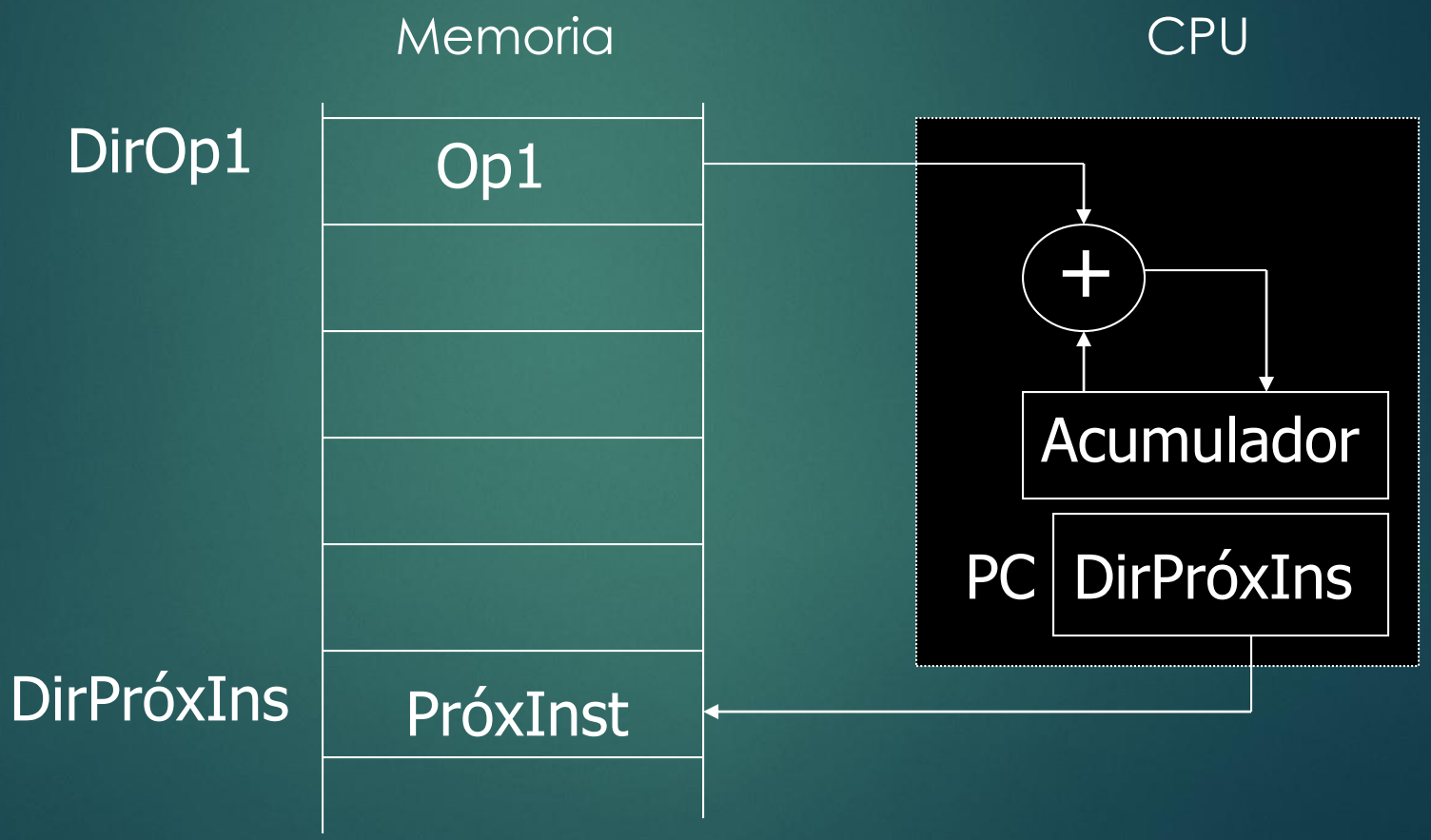
Máquina de 1 dirección

- Se puede simplificar aún más el tamaño de la instrucción si se elimina otro campo más, que referencia una dirección.
- Es decir, la instrucción dispone de 1 solo campo de dirección, que se usa para referenciar un operando o un resultado.

En assembly: Add DirOp1

Formato de instrucción
Máquina de 1 dirección

- La operación de suma se ejecutaría de la siguiente manera:



Formato de instrucción

Máquina de 1 dirección

Características de la máquina de 1 dirección

- Una sola dirección es explícitamente referenciada. El campo de dirección tiene que tener suficientes bits para "acomodar" o referenciar una dirección completa.
- Dado que la instrucción especifica una sola dirección donde reside el segundo operando Op2, se requiere:
  - El primer operando Op1 está implícitamente almacenado en algún lugar, normalmente es en un registro de la CPU, denominado Acumulador.
  - El lugar a donde se va a guardar el resultado está implícito. Normalmente es en el mismo registro que contenía el Op1, es decir, en el Acumulador.

Formato de instrucción

Máquina de 1 dirección

Características de la máquina de 1 dirección

- La CPU requiere de “registros especiales” (Acumulador)
- La CPU requiere de nuevas instrucciones para cargar y descargar los “registros especiales” (Acumulador).
- El lugar donde reside un operando está predefinido (implícito).
- El lugar donde se guarda el resultado está predefinido (implícito).
- Si, por ejemplo, la dirección requiere de 24 bits, la instrucción tiene: 1 x 24= 24 bits de referencias.

Uso Personal

Comparación entre máquinas de 3, 2 y 1 dirección

Ej. Analizar la cantidad de instrucciones, y accesos a memoria para el siguiente ejercicio: a=(b+c)*d - e

3 direcciones

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

2 direcciones

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

1 dirección

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

Resultados 3 direc.

- 3 accesos a Inst.
- 9 acc. a datos
- Total: 12 accesos

Resultados 2 direc.

- 4 accesos a Inst.
- 11 accesos a datos
- Total: 15 accesos

Resultados 1 direc.

- 5 accesos a Inst.
- 5 accesos a datos
- Total: 10 accesos

Modos de direccionamiento

- Tal como se mencionó anteriormente, en una instrucción hay un campo reservado para expresar el código de operación (COP) que identifica qué debe hacer la instrucción.
- El resto de los campos hacen referencia a los lugares donde residen los operandos y a donde se va a mandar (guardar) el resultado.

Modos de direccionamiento

Hay 3 métodos para reducir el tamaño de los campos que referencian operandos o resultados.

- Especificando registros: la cantidad de registros es pequeña, con lo que la cantidad de bits para espcificarlo es también muy baja. Normalmente, si un dato va a usarse repetidas veces, es conveniente cargarlo en un registro. Usar un registro para una variable tiene una ventaja adicional: el acceso al registro es mucho más rápido que a la memoria.
- Referenciando en forma implícita operandos y/o resultado: Ejemplo: reg2 = reg2 + fuente1
- Usando técnicas de referenciamiento: estas técnicas se conocen como modos de direccionamiento.

Uso Personal

Modos de direccionamiento

Los Modos de direccionamiento (Mdd) son las distintas formas o estrategias que tiene un procesador para referenciar operandos, resultado, y/o próxima instrucción a ejecutar.

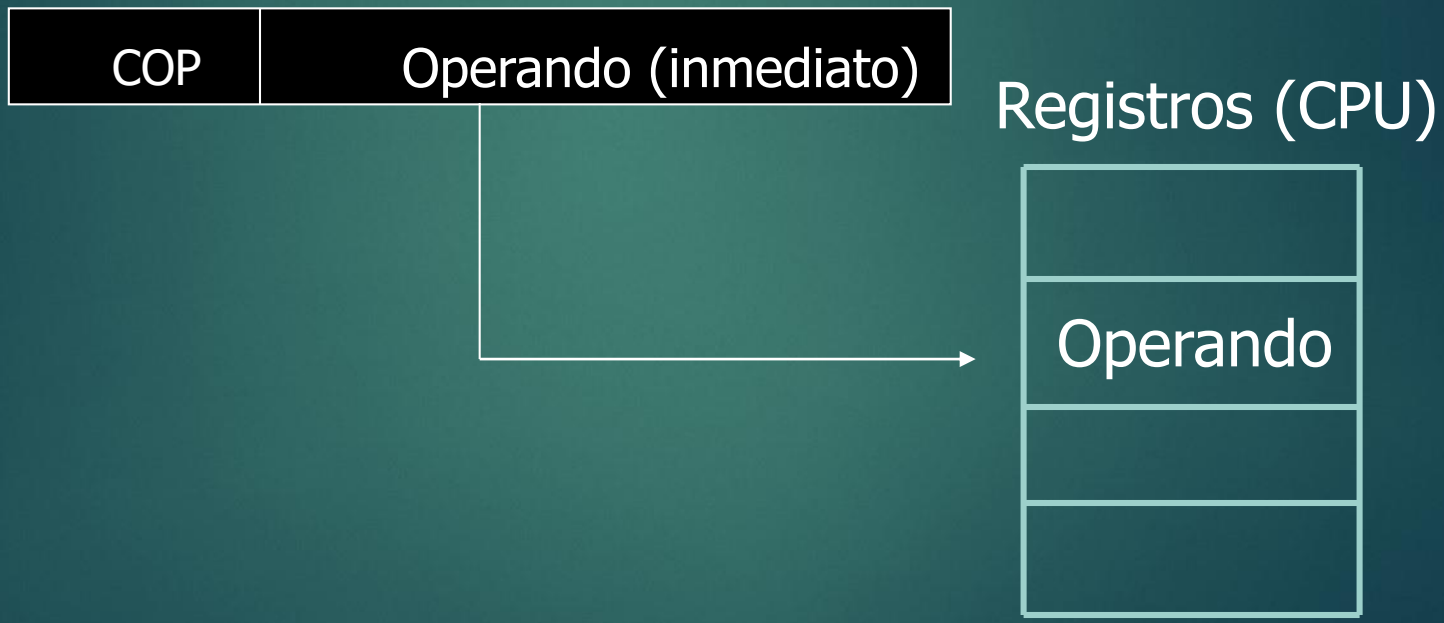
- Los Mdd tienen como objetivos:
  - Disminuir la cantidad de bits en la instrucción.
  - Posibilitar manejar referencias obtenidas durante la ejecución del programa (calculadas)
  - Permitir un manejo más eficiente de datos (arreglos).

Modos de direccionamiento

- Existe una amplia variedad de Mdd.
- Se van a analizar a continuación algunos de los más importantes:
  - Inmediato
  - Directo
  - Por registro
  - Indirecto por memoria
  - Indirecto por registro
  - Por desplazamiento
  - Del stack

Modos de direccionamiento: Inmediato

- En el modo de direccionamiento inmediato el operando está contenido en (forma parte de) la instrucción.
- Ejemplo de formato:



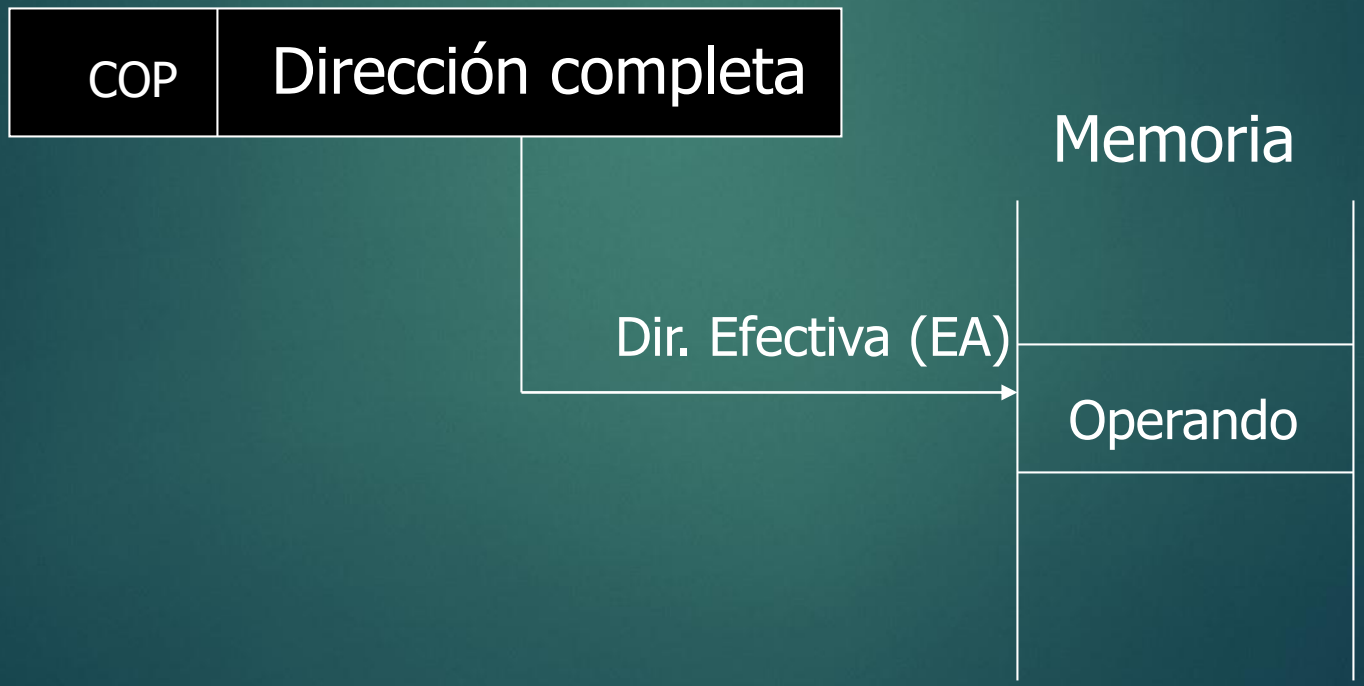
Uso Personal

Modos de direccionamiento: Inmediato

- El operando forma parte de la instrucción, y se obtiene (lee) al mismo tiempo que la instrucción.
- No requiere una referencia extra a la memoria para búsqueda del dato (al leer la instrucción se obtiene el operando).
- Uso:
  - Definición de constantes
  - Inicialización de variables: el dato con el que se inicializa la variable es conocido al momento de la compilación
- Desventaja: el tamaño del operando influye en el tamaño de la instrucción.

Modos de direccionamiento: Directo o absoluto

- En el modo de direccionamiento directo o absoluto la instrucción contiene la dirección (completa) de memoria donde reside el operando.
- Ejemplo de formato:



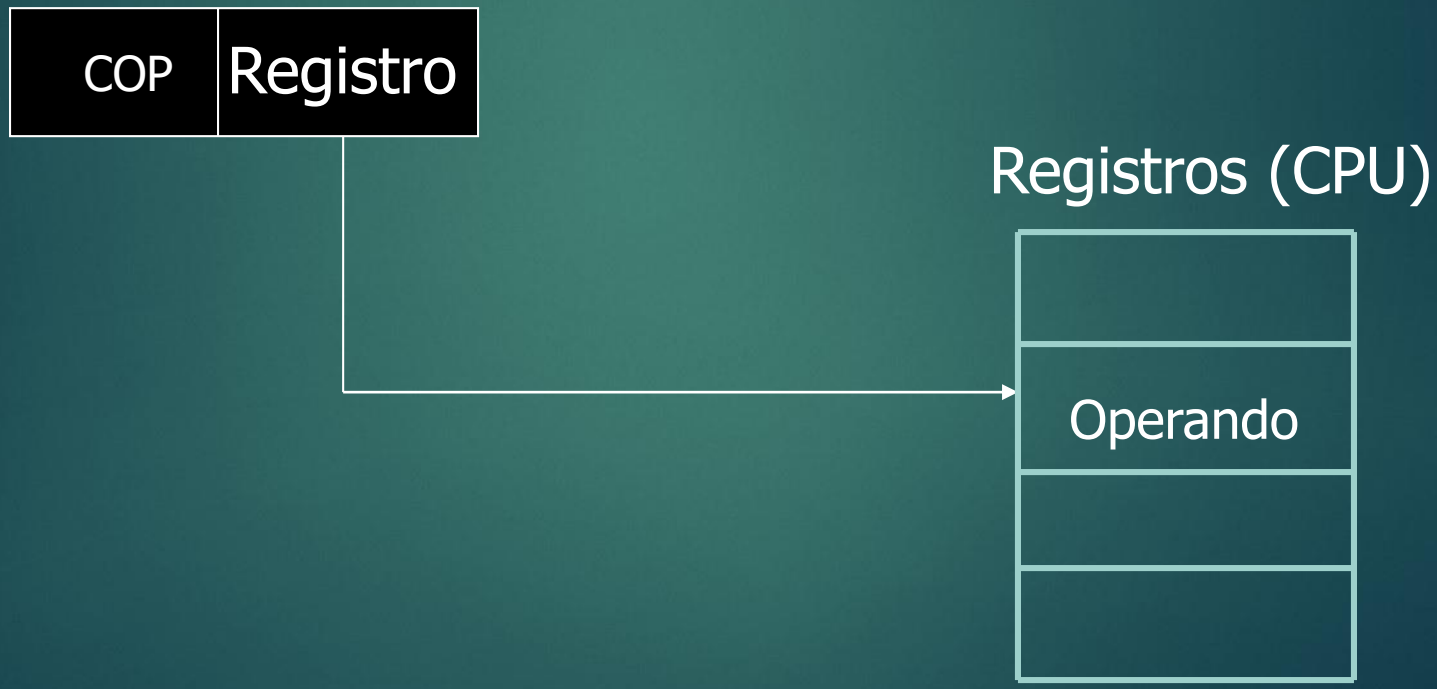
Uso Personal

Modos de direccionamiento: Directo o absoluto

- La instrucción tiene un campo que contiene la dirección (completa) donde efectivamente reside el operando. Esa dirección se llama Dirección efectiva (en inglés EA).
- El modo de direccionamiento es simple, pero tiene un espacio limitado de direcciones por cantidad de bits del campo.
- Uso:
  - Acceso a variables globales, cuya dirección se conoce en el momento de la compilación.
- Desventaja: para espacios de direcciones grandes, se necesitan muchos bits para especificar la dirección completa, lo que aumenta el tamaño de la instrucción.

Modos de direccionamiento: A Registro

- En el modo de direccionamiento a registro la instrucción contiene el número de registro donde reside el operando.
- Ejemplo de formato:



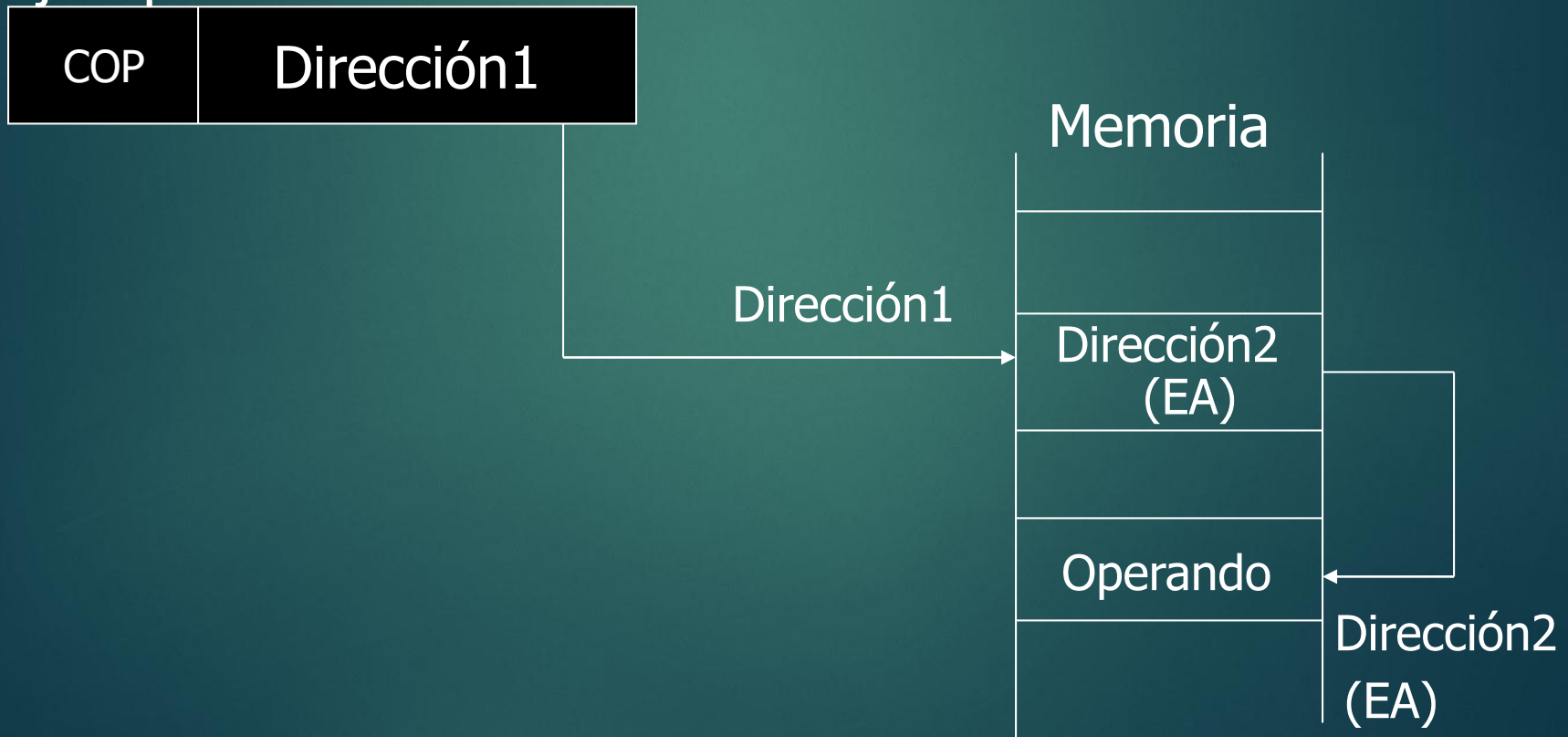
Modos de direccionamiento:

A Registro

- La instrucción tiene un campo que contiene el número de registro donde reside el operando.
- Conceptualmente es similar al direccionamiento directo, pero en este caso se especifica un número de registro en lugar de una posición de memoria.
- Uso:
  - Acceso a variables locales o globales
- Ventajas: la referencia a un registro requiere menos bits que la especificación de la dirección y no requiere de accesos extra a memoria de datos.
- Desventaja: los registros no son muchos y es un recurso muy “valioso”.

Modos de direccionamiento: Indirecto vía memoria

- En el modo de direccionamiento indirecto vía memoria la instrucción contiene la dirección de memoria donde reside la dirección del operando.
- Ejemplo de formato:



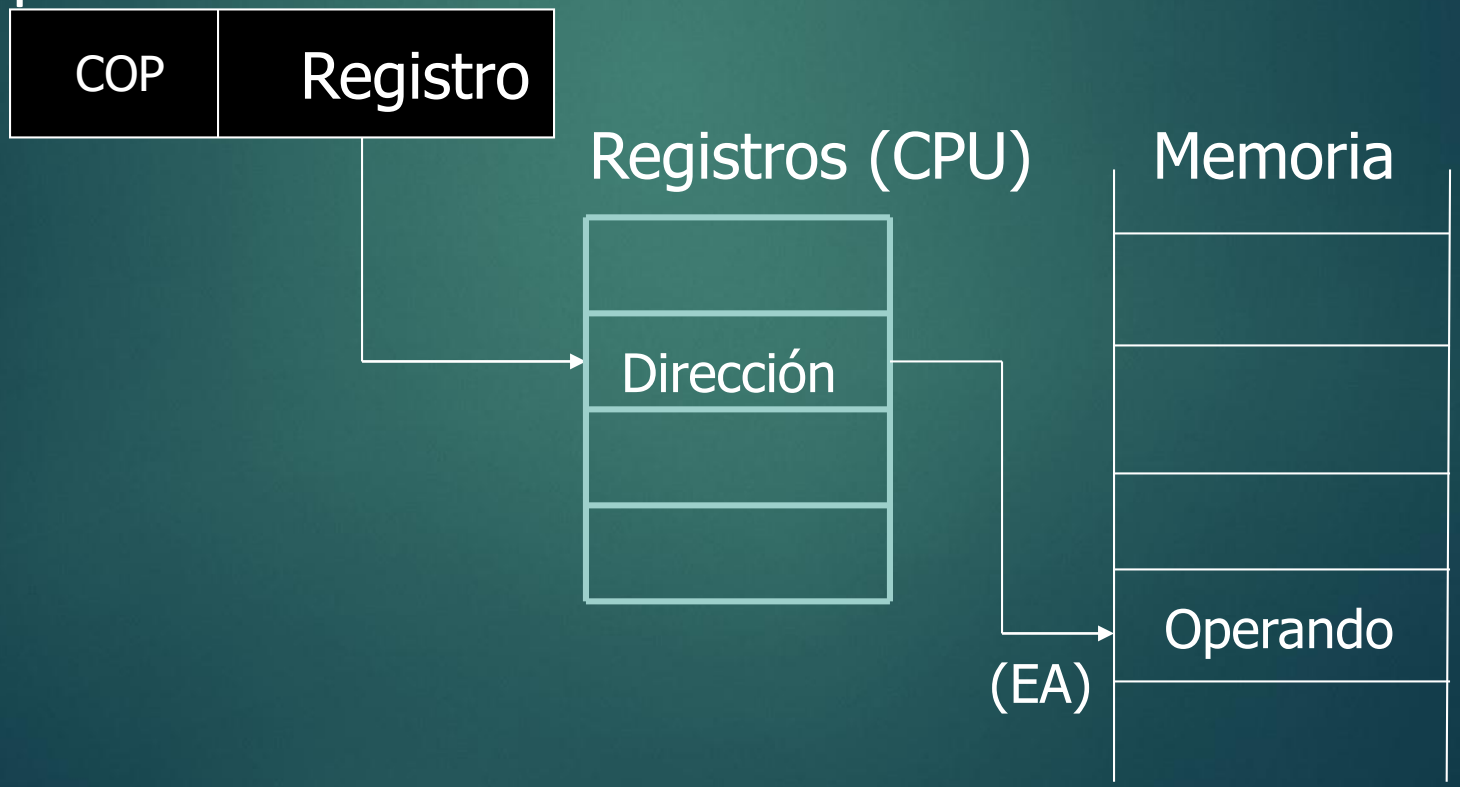
Uso Personal

Modos de direccionamiento: Indirecto vía memoria

- En la instrucción está la dirección donde reside la dirección efectiva del operando.
- A veces la Dirección1 es más “corta” que la Dirección2, con lo que se usa una dirección de menos bits (en la instrucción) para apuntar a una dirección de más bits.
- La Direccion2 se puede obtener durante la fase de ejecución del programa.
- Uso:
  - Acceso a (administración de) estructuras de datos
- Ventaja: espacio de direccionamiento mayor.
- Desventaja: múltiples accesos a memoria.

Modos de direccionamiento: Indirecto vía registro

- En el modo de direccionamiento indirecto vía registro la instrucción contiene el número de registro donde reside la dirección del operando.
- Ejemplo de formato:



Uso Personal

Modos de direccionamiento: Indirecto vía registro

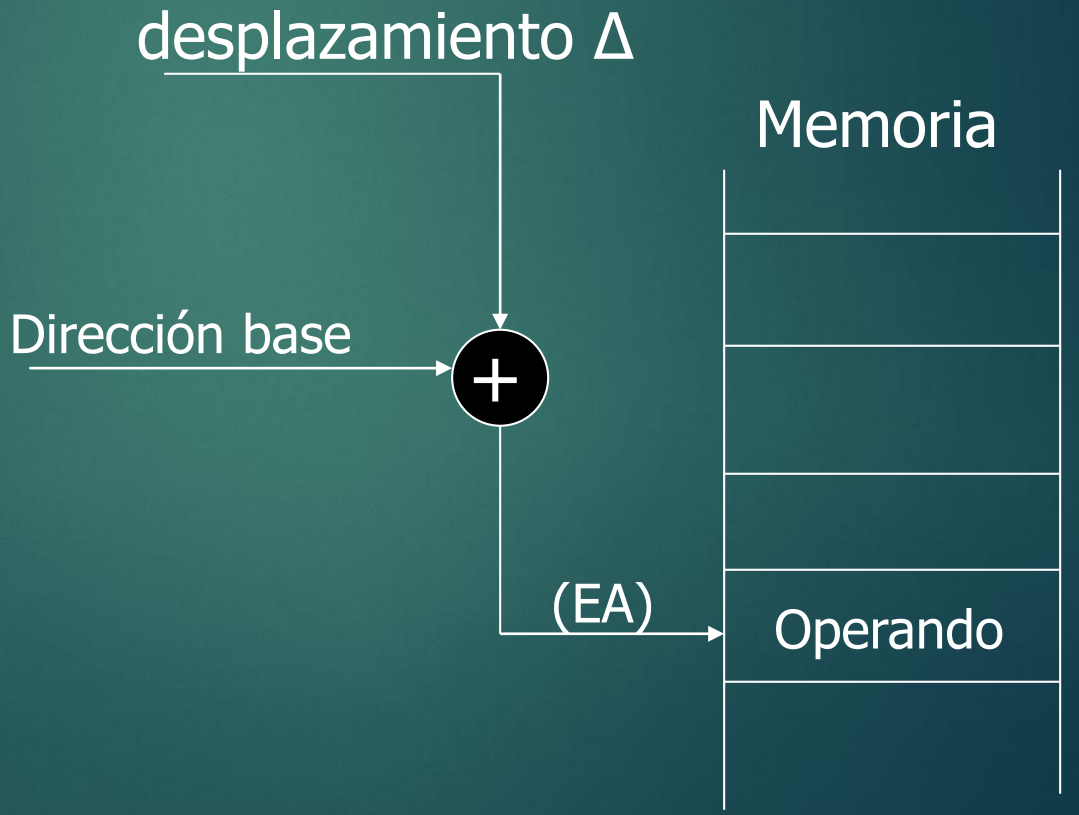
- En la instrucción se especifica el registro donde reside la dirección efectiva (EA) del operando.
- Uso:
  - Acceso a (administración de) estructuras de datos
- Ventajas:
  - Instrucción más “corta” porque se requieren menos bits para especificar el registro (respecto del Indirecto por memoria).
  - Espacio de direccionamiento grande.
  - Menos accesos a memoria porque la EA está en un registro interno.
- Desventaja: pocos registros.

Modos de direccionamiento: Direccionamientos calculados

- Los modos de direccionamiento calculados son aquellos en los que se requiere realizar una operación aritmético o lógica para obtener la dirección efectiva (EA) donde reside el operando.
- La instrucción contiene 2 o más referencias numéricas que deben “combinarse” para obtener la EA.
- La combinación básicamente es una cuenta (típicamente una suma).
- La suma es aritmética.
- Dado que se trata de direcciones, el valor resultante es un número sin signo.

Modos de direccionamiento: Direccionamientos calculados

- En los modos de direccionamiento calculados la EA se obtiene a partir de una operación aritmética entre 2 o más argumentos
- Ejemplo típico:



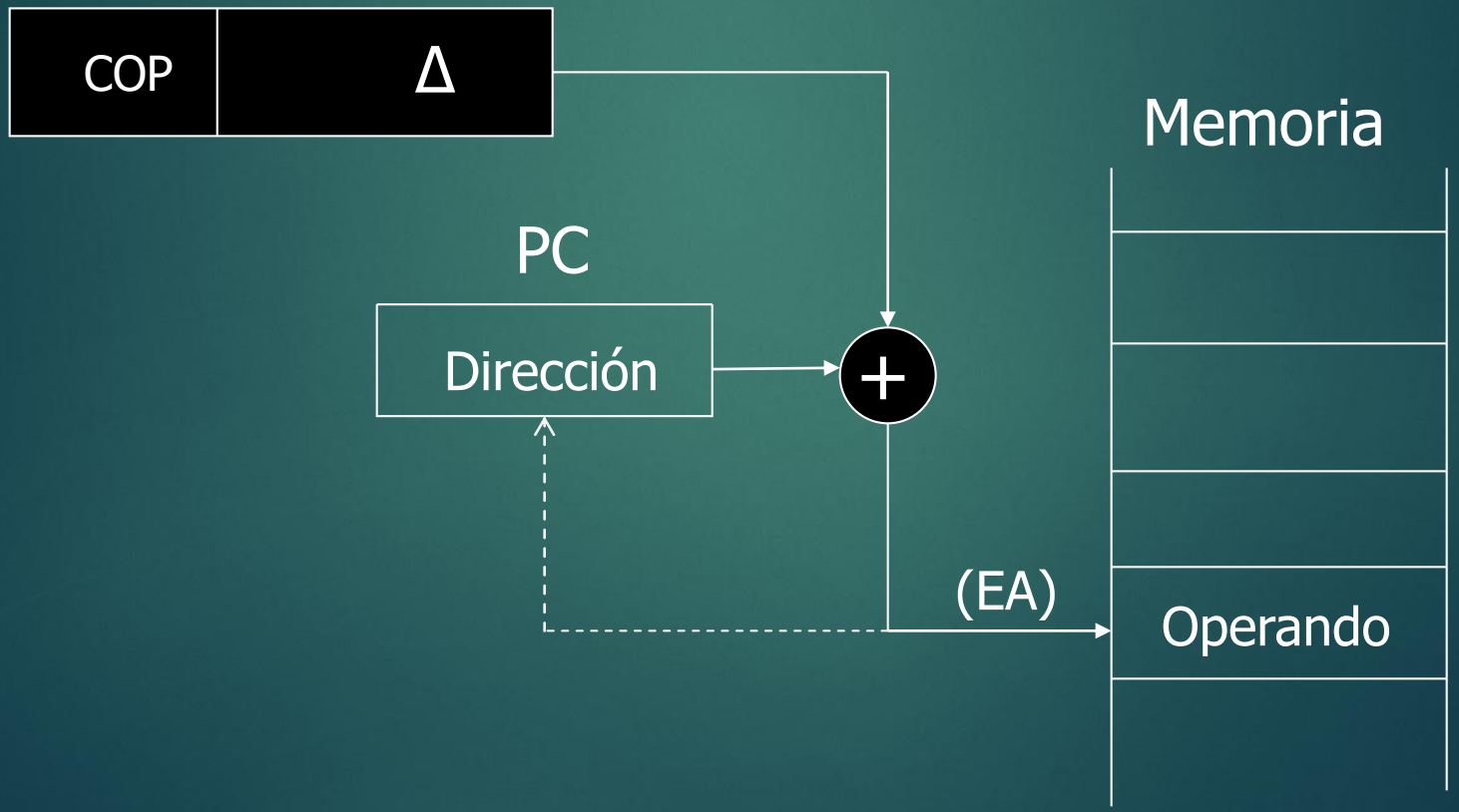
Uso Personal

Modos de direccionamiento: Direccionamientos calculados

- Hay distintos modos de direccionamiento calculados.
- Los más comunes son:
  - Relativo (al PC)
  - Base
  - Indexado

Modos de direccionamiento: Relativo al PC

- La instrucción contiene un campo numérico que es un desplazamiento Δ que se suma al (valor actual del) PC para obtener la dirección efectiva.



Modos de direccionamiento: Relativo al PC

- El registro referenciado (implícitamente) es el contador de programa PC.
- El campo numérico Δ se trata como un número en Ca2, es decir, como un número con signo (puede ser positivo o negativo).
- Como está en Ca2, el desplazamiento es con signo. Si el número es positivo, la referencia es hacia adelante, y si es negativo es hacia atrás.

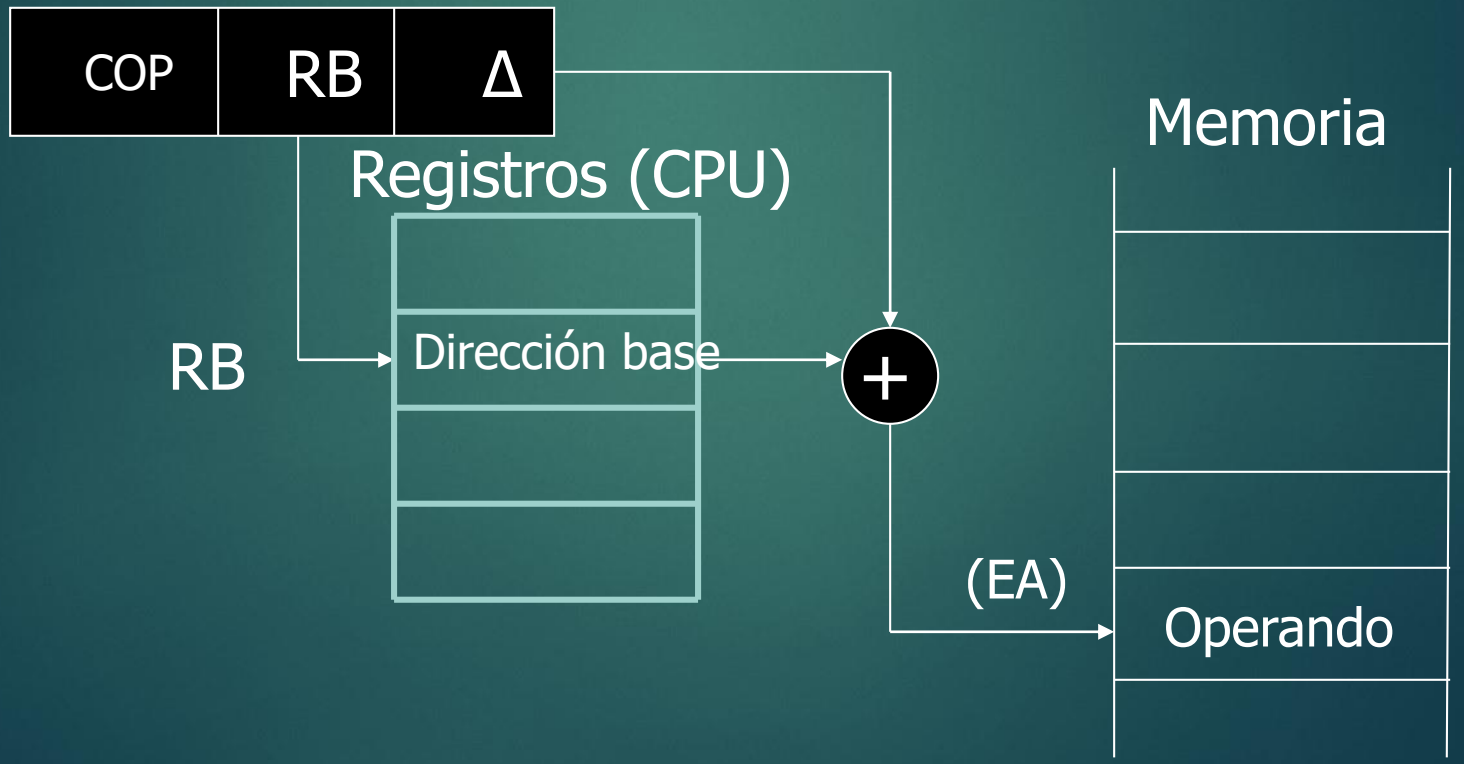
Uso Personal

Modos de direccionamiento: Relativo al PC

- El resultado de la operación de suma puede usarse de 2 maneras:
  - Obtener la EA de un operando y/o resultado (el PC no se altera).
  - Obtener la dirección de la próxima instrucción a ejecutar. En este caso, el PC se carga con el valor calculado. Como el PC contiene, por definición, la dirección de la próxima instrucción a ejecutar, el efecto del cálculo del modo de direccionamiento relativo al PC es producir un salto en el proceso de ejecución normal del programa.

Modos de direccionamiento: Base

- La instrucción contiene 2 campos: una referencia a un registro (registro base) y un campo numérico. El registro base RB contiene una dirección de memoria, y el campo numérico es un desplazamiento (offset) que se suma al contenido del registro base para obtener la dirección efectiva.



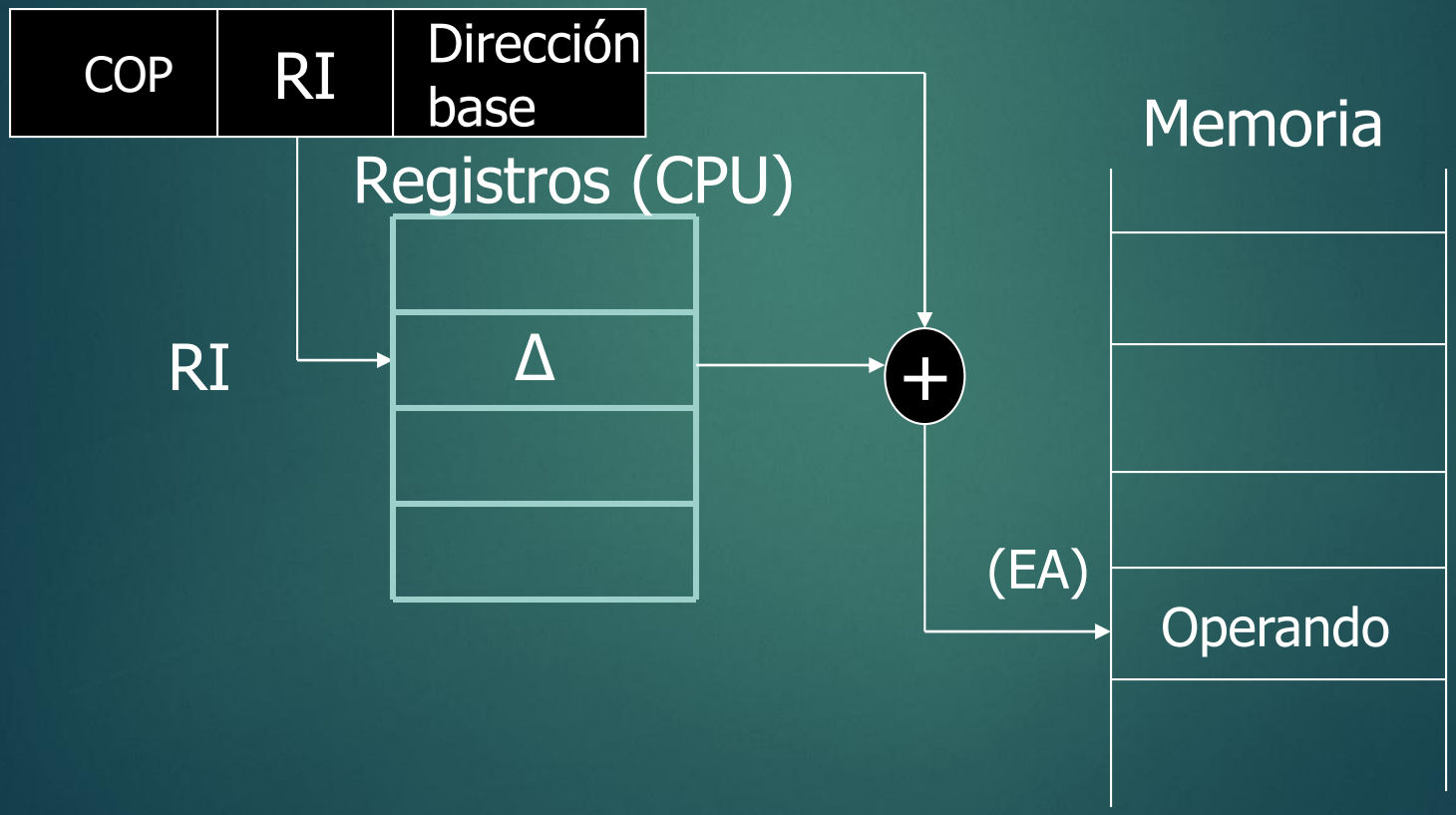
Uso Personal

Modos de direccionamiento: Indexado

- La instrucción contiene 2 campos: una dirección (base) y una referencia a un registro (Registro Indice).
- A la Dirección Base se le suma el contenido del Registro Indice para obtener la dirección efectiva.
- Es similar al anterior pero se intercambian los papeles de base y desplazamiento.
- La Indexación proporciona un mecanismo eficiente para realizar operaciones iterativas.
- En algunas máquinas, se incrementa ó decrementa el registro índice como parte de la instrucción.

Modos de direccionamiento: Indexado

- Ejemplo de formato:



Uso Personal

Modos de direccionamiento: Pila o Stack

- El modo de direccionamiento tipo Pila no tiene ninguna referencia a la dirección del operando. El operando está en el tope de una estructura tipo LIFO (List-in, First-out), comúnmente denominada Pila o Stack.
- El stack ó pila es un arreglo lineal de posiciones de memoria a las que se accede únicamente desde el tope (por ello el concepto de estructura tipo LIFO).
- Asociado con la pila o stack hay un registro apuntador (o registro puntero de pila), cuyo contenido es la dirección del tope de pila o stack.

Uso Personal

Diseño del conjunto de instrucciones

- El conjunto de instrucciones es el mecanismo por el cual el programador controla la secuencia de acciones que ejecuta la CPU.
- En e diseño del repertorio de instrucciones algunos de los puntos salientes que se deben tener en cuenta son:
  - 1) Formato de instrucciones:
    - Tamaño en bits que ocupa la instrucción
    - Número de campos que forman la instrucción
    - Cantidad de referencias explícitas a operandos y resutados
    - Tamaño en bits de cada campo
  - 2) Tipos de operaciones
    - Cuantas operaciones (instrucciones) dispone el procesador
    - Qué tipo y variedad de instrucciones provee el repertorio de instrucciones

Uso Personal

Diseño del conjunto de instrucciones

- 3) Modos de direccionamiento:
  - Cantidad y tipo de modos de direccionamiento
- 4) Registros:
  - Cantidad y tipo de registros
  - Registros referenciados por las instrucciones
- 5) Tipos de datos:
  - Numéricos: representaciones con y sin singno
  - Representaciones e punto fijo y punto flotante
  - Representaciones alfanuméricas

Uso Personal

Diseño del conjunto de instrucciones

Por ejemplo, los tipos de instrucciones más comunes, caen en alguna de las siguientes categorías:

- Transferencia de datos
  Ejemplo en Assembly: Mov (load/store)
- Aritméticas
  Ejemplo en Assembly : Add, Sub, Inc, Dec, Mul
- Lógicas
  Ejemplo en Assembly : And, Or, Xor, Not
- Conversión
  Ejemplo en Assembly : DAA
- Entrada/salida
  Ejemplo en Assembly: In, Out
- Transferencia de control (salto, bifurcación)
  Ejemplo en Assembly : JMP, CALL, RET
- Control del sistema
  Ejemplo en Assembly: HLT

Uso Personal

Diseño del conjunto de instrucciones

Los tipos de datos más comúnmente usados son:

- Entero sin signo
- Entero con signo
- Punto fijo
- Punto flotante
- Caracteres: ASCII, BCD.
- Variables lógicas

Uso Personal

Repertorio de instrucciones del MSX88

Instrucciones de movimiento de datos del MSX88

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)
3	POPF	Desapila los flags	(flags)←[SP+1:SP]; (SP)←(SP)+2
4	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.

Uso Personal

Repertorio de instrucciones del MSX88

Instrucciones aritméticas y lógicas del MSX88

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).

Repertorio de instrucciones del MSX88

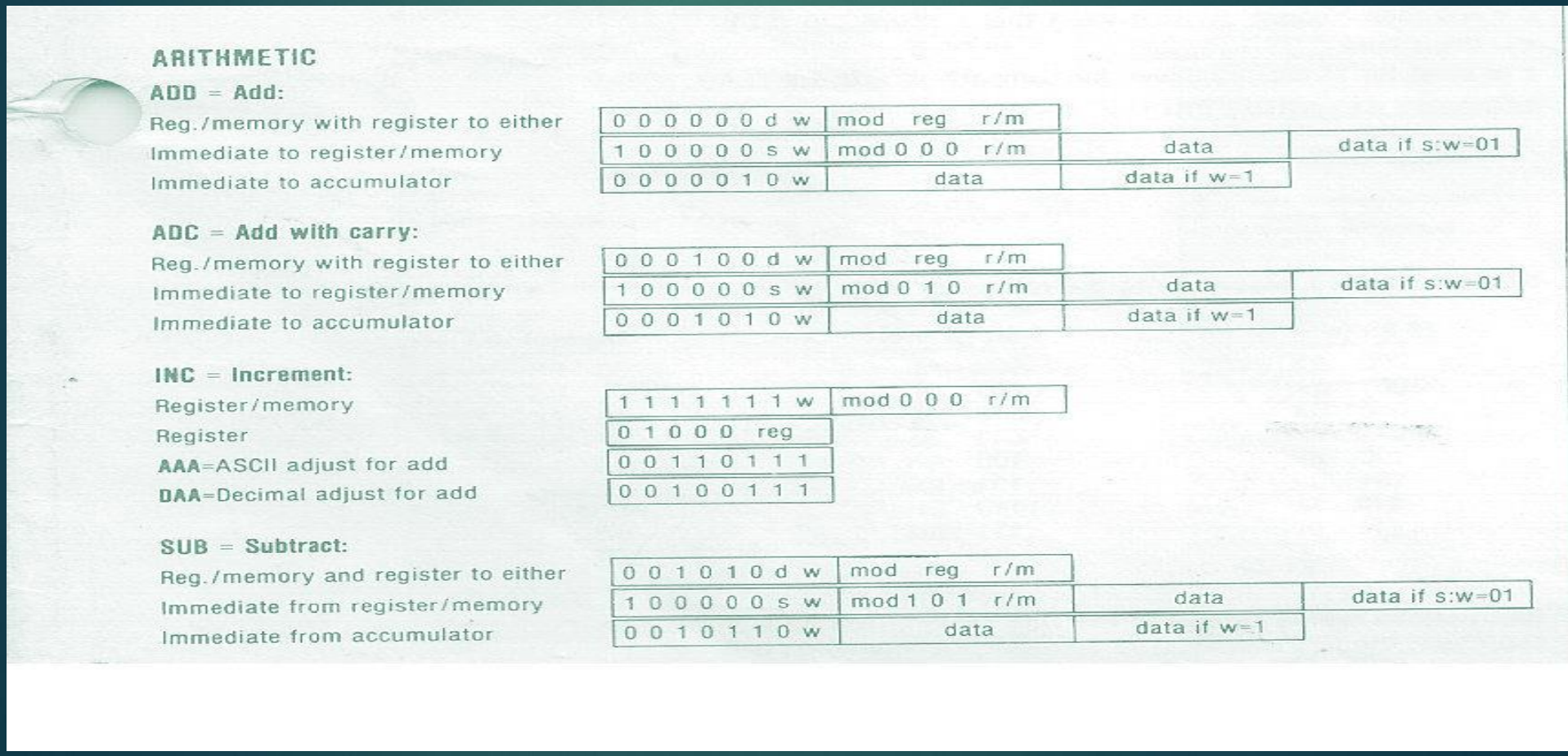
Instrucciones de transferencia de control del MSX88

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*.

Set de instrucciones 8086

A modo de ejemplo, se muestra una pequeña parte de la cartilla de instrucciones del 8086, con algunas instrucciones y la conformación de los campos incluidos en ellas.



ARITHMETIC

ADD = Add:

Reg./memory with register to either	0 0 0 0 0 0 d w	mod reg r/m		
Immediate to register/memory	1 0 0 0 0 0 s w	mod 0 0 0 r/m	data	data if s:w=01
Immediate to accumulator	0 0 0 0 0 1 0 w	data	data if w=1	

ADC = Add with carry:

Reg./memory with register to either	0 0 0 1 0 0 d w	mod reg r/m		
Immediate to register/memory	1 0 0 0 0 0 s w	mod 0 1 0 r/m	data	data if s:w=01
Immediate to accumulator	0 0 0 1 0 1 0 w	data	data if w=1	

INC = Increment:

Register/memory	1 1 1 1 1 1 1 w	mod 0 0 0 r/m
Register	0 1 0 0 0 reg	
AAA=ASCII adjust for add	0 0 1 1 0 1 1 1	
DAA=Decimal adjust for add	0 0 1 0 0 1 1 1	

SUB = Subtract:

Reg./memory and register to either	0 0 1 0 1 0 d w	mod reg r/m		
Immediate from register/memory	1 0 0 0 0 0 s w	mod 1 0 1 r/m	data	data if s:w=01
Immediate from accumulator	0 0 1 0 1 1 0 w	data	data if w=1	

Set de instrucciones 8086

Descripción de bits de los campos referenciados en las instrucciones del 8086 (ver hoja anterior).

AL = 8-bit accumulator
AX = 16-bit accumulator
CX = Count register
DS = Data segment
ES = Extra segment
Above/below refers to unsigned value.
Greater = more positive;
Less = less positive (more negative) signed values
if d = 1 then "to"; if d = 0 then "from".
if w = 1 then word instruction; if w = 0 then byte instruction

if s:w = 01 then 16 bits of immediate data form the operan
if s:w = 11 then an immediate data byte is sign extended t
form the 16-bit operand.
if v = 0 then "count" = 1; if v = 1 then "count" in (CL)
x = don't care
z is used for string primitives for comparison with ZF FLA

SEGMENT OVERRIDE PREFIX

0	0	1	reg	1	1	0

if mod = 11 then r/m is treated as a REG field
if mod = 00 then DISP = 0*, disp-low and disp-high are absent
if mod = 01 then DISP = disp-low sign-extended to 16-bits, disp-high is absent
if mod = 10 then DISP = disp-high: disp-low

if r/m = 000 then EA = (BX) + (SI) + DISP
if r/m = 001 then EA = (BX) + (DI) + DISP
if r/m = 010 then EA = (BP) + (SI) + DISP
if r/m = 011 then EA = (BP) + (DI) + DISP
if r/m = 100 then EA = (SI) + DISP
if r/m = 101 then EA = (DI) + DISP
if r/m = 110 then EA = (BP) + DISP*
if r/m = 111 then EA = (BX) + DISP

REG is assigned according to the following table:

16-Bit (w = 1)		8-Bit (w = 0)	
000	AX	000	AL
001	CX	001	CL
010	DX	010	DL
011	BX	011	BL
100	SP	100	AH
101	BP	101	CH
110	SI	110	DH
111	DI	111	BH

Uso Personal

Referencias

Repertorios de instrucciones: Stallings. 5ta Ed.

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

Lenguaje Assembly:

- Apunte 4 de cátedra
- Simulador MSX88
- Descargas en página web de cátedra