12 modos de direccionamiento

12.1 Introducción

Se entiende por **direccionamiento** la forma en que se interpretan los bits de un campo de dirección de una instrucción para localizar el operando y/o la dirección destino del resultado de la instrucción.

En esta lección vamos a ver algunos de los modos principales de direccionamiento que nos podemos encontrar en un ordenador en general, y en particular veremos ejemplos para el microprocesador Z80 de Zilog.

Los campos de bits dedicados a direcciones serán interpretados según el tipo de direccionamiento empleado en cada caso, y gracias a esta interpretación obtendremos lo que denominaremos **dirección efectiva**, que será el número que identifica la posición de memoria que deseamos utilizar (donde se encuentra la información o donde queremos depositar dicha información).

Los registros de la CPU constituyen una memoria de alta velocidad, y como tal tienen su propio mapa de direcciones de memoria, donde se asigna una dirección o número a cada uno para poderlos identificar. Por ejemplo, el VAX es una máquina con 16 registros generales nombrados de R0 a R15. Este conjunto de registros formarán un bloque de memoria que para poder ser accedido habrá que direccionar mediante el campo correspondiente dentro del formato de instrucción. Así, por ejemplo, una instrucción que sume el contenido de R3 con R8 sería una instrucción de dos direcciones, aún cuando las direcciones no son de memoria central, sino de registros de la CPU.

Veamos a continuación las diferentes maneras de realizar el direccionamiento. Como ejemplo de uso real de cada una, usaremos instrucciones del microprocesador Z80. En el Z80 los mnemónicos de las instrucciones en código máquina siempre son de la forma:

tipo de operación destino fuente

El Z80 es un microprocesador que posee un bus de direcciones de 16 bits, con lo que puede direccionar un total de 64K bytes de memoria. A este campo de direcciones hemos de añadir los 22 registros internos que posee la CPU:

- Un registro acumulador de 8 bits llamado A. Con él se realizarán las operaciones aritméticas y lógicas de 8 bits.
- Seis registros de 8 bits de propósito general: B, C, D, E, H y L, que pueden ser utilizados en forma de pares de registros para almacenar datos de 16 bits o también **punteros a memoria**. La única forma de agrupar por pares estos registros es BC, DE y HL.
- Un registro de estado F. Este registro junto con el acumulador pueden formar un par de registros llamado AF.
- Registros para guardar sólo direcciones:
  - PC o contador de programa, de 16 bits,
  - SP o puntero de pila, de 16 bits,
  - IX e IY, registros de índice, de 16 bits,
  - I, relacionado con la atención a las demandas de interrupciones, de 8 bits (almacena sólo los 8 bits más significativos del bloque de memoria en el que se encuentra el comienzo de la subrutina de tratamiento de interrupciones).

A esta lista deberemos añadir otros registros que no pueden ser manipulados directamente por el usuario, como son el R (de refresco de memoria) y el IR (registro de instrucción, sobre el que se almacena la instrucción que acaba de ser leída de la memoria central).

También hay que advertir que los registros A, B, C, D, E, F, H y L se encuentran duplicados, formando dos bancos alternativos de registros, a los que el usuario *sólo puede acceder simultáneamente a los de un grupo o a los de otro*.

12.2 Direccionamiento inmediato

La manera más simple de especificar un operando es que el campo de dirección de la instrucción contenga el propio operando, y entonces decimos que al código de operación le sigue un "**literal**".

Simbólicamente lo podemos representar de la siguiente manera:

Código de Operación	*Campo de Dirección*
	Operando

Este tipo de direccionamiento tiene la ventaja de la **rapidez** ya que no se precisan referencias adicionales a la memoria para extraer el operando. Sin embargo, presenta el inconveniente de ser una forma poco flexible de localizar un operando.

Ejemplo del Z80:

En el Z80 el dato inmediato (literal) puede ser de 8 o 16 dígitos binarios, dependiendo del tamaño del destino de la información.

*Un ejemplo son las instrucciones de la forma **LD r,n** donde r es el registro y n es el literal. Esta instrucción significa "cargar el registro r con el número n, que está especificado en el byte siguiente al código de operación". El formato en binario de esta instrucción es: 00 r 110, y después hay que especificar los 8 bits del dato numérico. Por ejemplo, la instrucción LD A,20 (el operando especificado en decimal) la instrucción en binario será:*

00 111 110 00010100

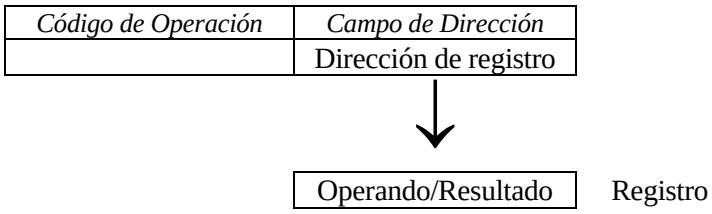
*La instrucción **LD dd,nn** significa "cargar el par de registros dd con el literal (dato) nn (16 bits) que se encuentra en los dos bytes siguientes al código de operación. El formato en binario de esta instrucción es: 00 dd0 001, seguido de los 16 bits del dato. El campo marcado con los bits dd varía según el par de registros utilizados de la forma indicada en la columna de la derecha de la tabla. Por ejemplo, para la instrucción LD HL, 400C (el operando especificado en hexadecimal), la instrucción en binario será:*

00 100 001 0100000000001100

12.3 Direccionamiento directo a registro

En el campo de dirección se especifica la dirección del registro donde se encuentra el operando o donde hay que dejar el resultado.

Simbólicamente lo podemos representar de la siguiente manera:



Ejemplo del Z80:

Un ejemplo de instrucción con direccionamiento directo a registro es la instrucción de carga ***LD r,s*** *que significa copiar el contenido del registro s (fuente) en el registro r (destino), ambos registros son de 8 bits de longitud. Esta es un instrucción con dos campos de dirección, y en ambos se emplea direccionamiento directo a registro, tanto para localizar el operando (registro r), como el destino (registro s). Si quisiéramos especificar como registro origen el E y como el A (LD E,A) la instrucción en binario será:*

01 011 111
LD E A

Entonces podríamos decir que esta instrucción en particular LD E, A utiliza direccionamiento directo a registro para localizar la fuente de la información (registro A), y direccionamiento directo a registro para identificar el destino de la información (registro E).

Otros posibles ejemplos de direccionamiento de registro son las instrucciones ADD A, r y DEC r.

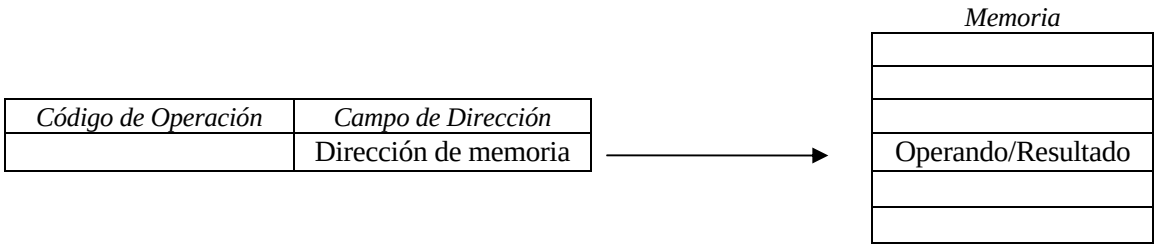
En la primera estamos indicando una operación de suma del contenido del registro A con el contenido de otro registro que queda por definir. La codificación en binario de esta instrucción es 10 000 r, donde r, al igual que antes nos permite seleccionar uno de los 8 registros de 8 bit. Un caso particular sería la instrucción ADD A, A (sumar el contenido del registro A con sigo mismo = multiplicar por 2), entonces el valor de r a sustituir en el formato de instrucción es 111, con lo que queda: 10 000 111 (binario).

La instrucción DEC r decrementa en uno el valor almacenado en el registro de 8 bits especificado. El formato de instrucción es 00 r 101. Si, por ejemplo, queremos decrementar el contenido del registro H (r=H) la instrucción será: 00 100 101.

12.4 Direccionamiento directo a memoria

En el campo de dirección se especifica la dirección de memoria donde se encuentra el operando o donde hay que dejar el resultado.

Simbólicamente lo podemos representar de la siguiente manera:



Ejemplo Z80

LD A,(nn) *que significa cargar en el registro A el valor que se encuentra almacenado en la dirección de memoria central nn (especificada por los 2 bytes siguientes al código de operación). En este caso se utiliza direccionamiento directo a memoria para localizar la fuente de la información, y direccionamiento implícito (le veremos más adelante) para identificar el destino de la información.*

Obsérvese que la dirección de memoria central que vamos a utilizar está escrita en el mnemónico "entre paréntesis". Si el dato que queremos copiar en A se encuentra, por ejemplo, en la dirección de memoria $12765_{(10}$ $(31dd_{(16)}$ *la instrucción tendría la siguiente forma, si se escribiera en hexadecimal en vez de binario para que quede una expresión más compacta:*

3a dd 31

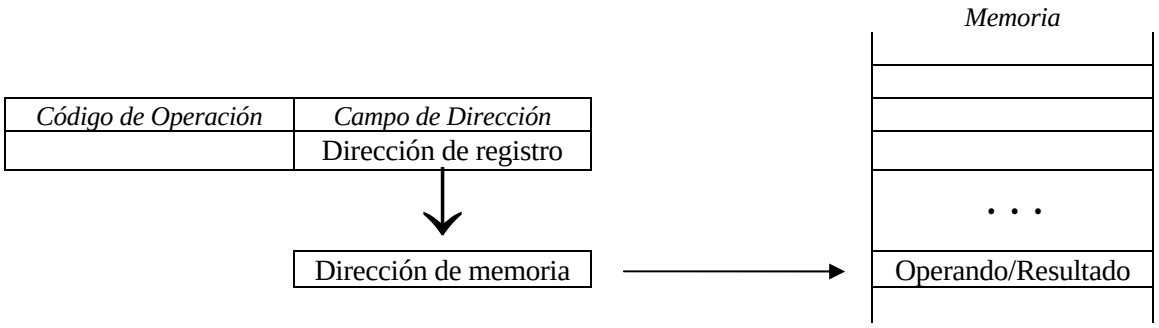
Donde 3a es el código de operación de la instrucción, dd es el byte menos significativo que identifica la dirección de memoria deseada, y 31 es el byte más significativo de la dirección de memoria deseada.

La instrucción recíproca de la que acabamos de ver es ***LD (nn),A****, que utiliza direccionamiento implícito la fuente de la información (registro A), y direccionamiento directo a memoria para localizar el destino de la información.*

12.4 Direccionamiento indirecto a registro

En el campo de dirección se especifica la dirección del registro donde se encuentra la dirección de memoria en la que se encuentra el operando o donde hay que dejar el resultado.

Simbólicamente lo podemos representar de la siguiente manera:



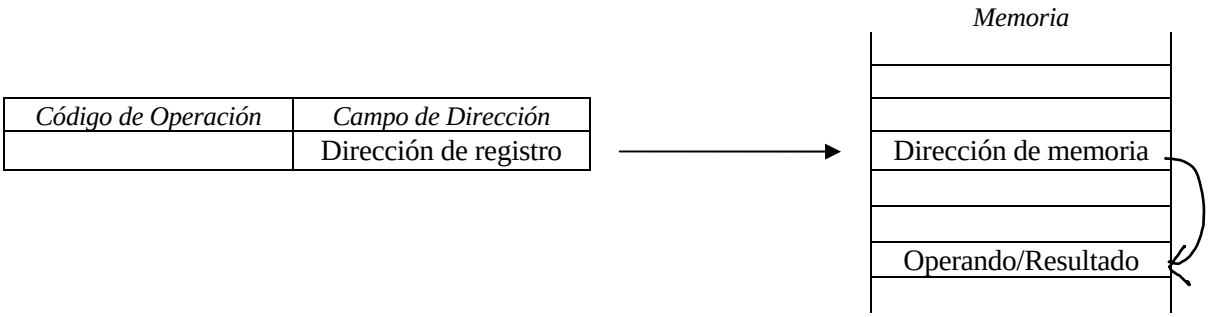
Ejemplo Z80:

En el Z80 la longitud de la palabra de dirección de memoria es de 16 bits, y por lo tanto para contener direcciones deberemos emplear pares de registros (BC, DE o HL). Una instrucción de direccionamiento indirecto es por ejemplo ***LD A,(HL)*** *que significa "cargar el registro A con el contenido de la posición de memoria indicada por el registro HL. En este caso el direccionamiento indirecto por registro se utiliza para localizar el operando, mientras que el destino se especifica mediante direccionamiento implícito.*

12.5 Direccionamiento indirecto a memoria

En el campo de dirección se especifica una dirección de memoria, cuyo contenido es a su vez otra dirección de memoria, donde se encuentra el operando o donde hay que dejar el resultado.

Simbólicamente lo podemos representar de la siguiente manera:



No existen ejemplos de este tipo de instrucciones en Z80 con este tipo de direccionamiento.

12.6 Direccionamientos relativo a PC, de registro base e indexado

Estos tres tipos de direccionamiento comparten la forma de calcular la dirección del operando o donde se deja el resultado.

El formato de instrucción general este tipo de instrucciones es:

Código de Operación	*Campo de Dirección 1*	*[Campo de Dirección 2]*
	Desplazamiento	Dirección de registro

Donde:

- El *campo de dirección 1* siempre aparece y su contenido es una cantidad, generalmente representada en complemento a dos, y que se denomina desplazamiento.
- El *campo de dirección 2* es optativo, es decir, no siempre aparece. Depende del tipo de direccionamiento y del número de registros de cada tipo que existan. Si aparece especifica un registro, es decir, es una dirección de registro, cuyo contenido se usará como veremos a continuación.

En este tipo de direccionamientos la dirección de memoria donde está el operando o donde debemos dejar el resultado, lo que se denomina **dirección efectiva**, se calcula de la siguiente manera:

dirección efectiva = contenido registro + desplazamiento

Donde el desplazamiento, como hemos visto, es el contenido del campo de dirección correspondiente, y el registro cuyo contenido se suma a éste es lo que diferencia a los 3 direccionamiento de este apartado:

- **Direccionamiento relativo a PC**: El registro cuyo contenido se suma al desplazamiento es el PC. El PC (Program Counter o Contador de Programa) es un registro especial de la CPU cuya función es contener en todo momento la dirección de memoria central donde se encuentra la siguiente instrucción a ejecutar. Como el registro PC es único, en este tipo de direccionamiento no existe el campo de dirección 2.
- **Direccionamiento de registro base:** En este caso, el registro cuyo contenido se suma al desplazamiento se denomina registro base. Si en una máquina sólo existe un único registro base, no será necesario el campo de dirección 2. Sin embargo, si existe más de uno, habrá que incluir ese campo en la instrucción y su contenido será la identificación del registro base cuyo contenido será sumado al desplazamiento.
- **Direccionamiento indexado:** Se puede repetir todo lo comentado para el de registro base, con la única diferencia de que aquí el registro cuyo contenido se suma al desplazamiento se llama registro índice.

Aparte del tipo de registro usado para el cálculo de la dirección efectiva, la diferencia entre estos tres modos de direccionamiento radica en para qué se usa cada uno:

- **Direccionamiento relativo a PC.** Se usa fundamentalmente para realizar saltos en la ejecución de un programa.
- **Direccionamiento de registro base - Direccionamiento indexado:** La diferencia radica en que en el indexado, en general, se puede referenciar toda la memoria, mientras que en el registro de base esto no está permitido. Otra diferencia, radica en las características de los registros empleados en ambos, así por ejemplo, en el indexado se suele tener la posibilidad de autoincrementar o autodecrementar el contenido del registro índice automáticamente después de ejecutar una instrucción con este tipo de direccionamiento, cosa que no se puede hacer en el de registro de base. Esta diferencia hace que el direccionamiento indexado sea especialmente útil para realizar operaciones (instrucciones) sobre un conjunto de datos almacenados en posiciones consecutivas de memoria, como por ejemplo las que se realizan sobre vectores.

Antes de ver algunos ejemplos del Z80, indicar que el hecho de que el desplazamiento sea, generalmente, un cantidad representada en complemento a 2, quiere decir que ese puede ser tanto positivo como negativo, lo que quiere decir que la dirección efectiva puede ser tanto mayor (estar por encima), como menor (estar por debajo) que la dirección almacenada en el registro correspondiente.

Ejemplo Z80

- ***Direccionamiento relativo a PC.*** *Ejemplo de instrucción:* ***JR d****, donde d es el desplazamiento, que es una constante de 8 dígitos binarios. Esta instrucción almacena en el registro PC la dirección efectiva, lo que en la práctica significa realizar un salto relativo a PC, es decir, la siguiente instrucción a ejecutar estará d posiciones por encima (si de es positivo) o d posiciones por debajo (si d es negativo) de la dirección actualmente almacenada en el PC. Por ejemplo, si la instrucción fuera JR* $d5_{(16}$ *la dirección efectiva sería:*

  dirección efectiva = contenido PC + (-43) ($d5_{(16}$ *escrito con 8 bits e interpretado en complemento a 2 es el número* $-43_{(10)}$*).*

  Lo que significa realizar un salto en la ejecución del programa de 43 posiciones hacia atrás, es decir, la siguiente instrucción a ejecutar estará 43 posiciones de memoria más atrás de la dirección actualmente almacenada en el PC.

  Para poder determinar completamente cuál es el punto destino de nuestro programa (donde se encuentra la siguiente instrucción por ejecutar), primero hemos de saber dónde se encuentra la instrucción JR d5. Supongamos que dicha instrucción se encuentra en la posición de memoria $FF00_{(16}$ *. La instrucción de salto relativo tiene una longitud de 16 dígitos binarios, o sea, ocupa en memoria dos posiciones de memoria, por lo que después de haber sido leída completamente por la CPU, el PC valdrá FF02. Bajo estos datos, la dirección efectiva será FED7, siendo éste el valor que se almacenará en el PC, por lo que la siguiente instrucción que se ejecutará será la contenida en esa dirección de memoria.*
- ***Direccionamiento de registro base.*** *El Z80 no tiene este tipo de direccionamiento.*
- ***Direccionamiento indexado.*** *El Z80 posee sólo dos registros de índice: IX e IY ambos de 16 bits, y el desplazamiento es una constante de 8 bits en* ***complemento a 2****. Un ejemplo de instrucción con este direccionamiento es:* ***LD r,(IX+d)****, que significa cargar el registro r con el valor de la posición de memoria indicada por el contenido del registro IX más d. En este caso el direccionamiento indexado se utiliza para localizar el operando, que será almacenado en un destino especificado mediante direccionamiento directo por registro.*

12.7 Direccionamiento a pila

Toda la información necesaria para localizar los operandos o el destino está almacenada en unos registros especiales de la CPU que sirven para las operaciones sobre pila, y que veremos mas adelante. Como estos registros son únicos, no hace falta especificarlos en la instrucción, por lo que para las operaciones sobre pila no hace falta incluir un campo de dirección específico en la instrucción.

Cuando hablamos de los diferentes tipos de formatos de instrucciones, dijimos que convenía hacer instrucciones lo más cortas posibles, con objeto de ahorrar tanto tiempo de CPU como espacio de memoria. El uso de este tipo de direccionamiento favorece esa reducción en las longitudes de la instrucción, al no necesitar campo de dirección.

Una pila está formada por datos elementales (palabras, caracteres, bits, etc.) almacenados en orden consecutivo en una zona determinada de la memoria central del ordenador. El primer dato introducido en la pila se dice que está en el **fondo de la pila** o en la base de la pila. El que se ha introducido más recientemente (último) se dice que está en la **cima de la pila**. Hay siempre un registro o una palabra de memoria asociada a la pila que contiene la dirección de la cima que se denomina **puntero de pila**. Este dato es el único que necesitamos, como veremos, para las operaciones sobre la pila.

Veamos el funcionamiento de la pila, mediante un ejemplo:

Dirección de memoria	Puntero de pila (dirección de memoria del último dato introducido)			
	1001	1002	1003	1002
1000	100	100	100	100
1001	40	40	40	40
1002		6	6	6
1003			75	

Supongamos que el fondo de la pila está en dirección 1000 y la cima en la primera columna, que representa el estado inicial de la pila, está en 1001. Seguidamente **apilamos** o añadimos un nuevo elemento a la pila (el número 6), que se situará justo a continuación del último dato almacenado, o sea en la dirección 1002. Cada vez que se añade un nuevo dato habrá que actualizar el puntero de pila, sustituyendo su anterior valor por la la dirección de memoria donde se ha almacenado el último dato de entrada, es decir la 1002.

Si añadiéramos un nuevo elemento a la pila (el número 75), este se almacena en la dirección 1003, y el puntero de pila se actualizará incrementando, como antes, en 1 su valor, pasando a valer 1003.

Si ahora **desapilamos** o extraemos información de la pila, lo que obtendremos es el último dato almacenado (recordar que la estructura de pila se rige por el esquema "último en entrar, primero en salir"), este será el número 75, y el puntero de pila se actualiza indicando la dirección de memoria del que pasa ahora a ser el último dato de la pila, es decir, la dirección 1002. La situación de la pila quedará como se indica en la última columna de la tabla anterior.

Vemos que sobre una pila se pueden realizar dos operaciones básicas que son la de apilar o añadir un nuevo dato a la pila, y la de desapilar o extraer un dato de la pila. Los pasos a seguir en la realización de cada una de estas operaciones depende de hacia qué direcciones crezca la pila. Se dice que una pila crece hacia direcciones superiores o altas si al añadir un nuevo dato, esto se hace en la dirección inmediatamente superior a la de la cima (el puntero de pila incrementa su valor), como en el ejemplo anterior. Se dice que una pila crece hacia direcciones inferiores o bajas si al añadir un nuevo dato, esto se hace en la dirección inmediatamente inferior a la de la cima (el puntero de pila decrementa su valor); este es el caso de Z80.

Veamos más detenidamente el paso a seguir en cada operación para cada caso de crecimiento de la pila:

- **Crecimiento hacia posiciones superiores:** dirección de la cima > dirección de la base
  - **Operación añadir dato:**
    1. Incrementar el puntero de pila.
    2. Almacenar el dato en la dirección indicada por el puntero de pila.
  - **Operación extraer dato:**
    3. Leer el dato almacenado en la dirección de memoria indicada en el puntero de pila.
    4. Actualizar el valor del puntero de pila para que apunte a la nueva cima, decrementando su valor.

- **Crecimiento hacia posiciones inferiores:** dirección de la cima < dirección de la base
  - **Operación añadir dato:**
    5. Decrementar el puntero de pila.
    6. Almacenar el dato en la dirección indicada por el puntero de pila.
  - **Operación extraer dato:**
    7. Leer el dato almacenado en la dirección de memoria indicada en el puntero de pila.
    8. Actualizar el valor del puntero de pila para que apunte a la nueva cima, incrementando su valor.

Ejemplo Z80:

Un ejemplo en el Z80 de operaciones de apilar y extraer información en la pila son las instrucciones ***PUSH qq*** *y* ***POP qq*** *respectivamente, donde qq es un par de registros (AF, BC, DE o HL).En binario cada una de esas instrucciones queda:*

11qq0101 → PUSH qq donde qq en la instrucción es la codificación del par de registros a usar.

11qq0001 → POP qq donde qq en la instrucción es la codificación del par de registros a usar.

Al realizar un PUSH el dato almacenado en el registro rr se introduce en la pila y el puntero de pila se decrementa (recordamos que en el Z80 la pila crece hacia posiciones inferiores) en 2 unidades (hemos introducido 2 bytes, uno por cada registro). Un ejemplo de instrucción de este tipo sería PUSH BC, que en binario quedaría 11000101 (siendo 00 la combinación binaria que identifica al par de registros BC).

Al extraer datos de la pila (POP), los dos bytes situados en la cima y en la posición inmediatamente superior, respectivamente, se almacenan en el par de registros qq. El puntero de pila se incrementa en dos unidades. Un ejemplo de instrucción de este tipo sería POP DE, que en binario quedaría 11010001 (siendo 01 la combinación binaria que identifica al par de registros DE).

12.8 Direccionamiento implícito

En este caso o bien el operando o bien la dirección del resultado están implícitos en el código de operación de la instrucción, no necesitando, por lo tanto, un campo de dirección en la instrucción para especificarlo.

Ejemplo Z80

Ya hemos visto, y así lo hemos señalado, a lo largo de este tema algunos ejemplos de instrucciones que localizaban alguno de sus operandos o la dirección del resultado de esta manera. Vamos aquí a añadir algún caso más. Por ejemplo, la instrucción de suma ***ADD A,n*** *realiza la siguiente operación: sumar n al contenido del acumulador, y el resultado dejarlo en el acumulador (simbólicamente: A←A+n). En binario esa instrucción se expresa de la siguiente manera: 11000110n, donde n es la combinación binaria de 8 dígitos que se sumará al acumulador. Como se puede observar, esta instrucción sólo tiene un campo de dirección cuyo contenido es n, lo cual quiere decir que ese operando se especifica mediante direccionamiento inmediato. El otro operando (el contenido del acumulador) y dónde dejar el resultado (en el acumulador) no aparecen de manera explícita*

en la instrucción, lo cual quiere decir que son direccionados mediante direccionamiento implícito: la CPU al leer el código de operación de esta instrucción (11000110), ya sabe donde localizar ambas cosas.

12.9 Direccionamiento de bit (Z80)

El direccionamiento de bit es un mecanismo de acceso a bits específicos, ya sea de una determinada posición de memoria o de un registro en particular.

Obviamente este tipo de direccionamiento sólo nos indicará si determinado bit de una posición de memoria es 1 ó 0. Recuérdese que para realizar este tipo de consultas se puede utilizar un mecanismo basado en máscaras, tal y como se vio en la lección dedicada a la UAL. Por ejemplo los microprocesadores de la familia del 8088 no poseen este tipo de instrucciones que utilicen direccionamiento de bit.

Sin embargo, el Z80 si dispone de instrucciones especiales para activar, desactivar y verificar bits específicos de un registro o una posición de memoria, y por lo tanto involucrarán direccionamiento de bit.. Naturalmente que si hemos de ser capaces de seleccionar un bit dentro de una palabra de 8 bits, dentro del código de operación habrán de ser usados como mínimo tres bits para este tipo de direccionamiento.

Ejemplo:
Las instrucciones BIT, SET y RESET manipulan bits independientes.

12.10 Comentarios finales

Hemos visto diferentes modos de direccionamiento, y conviene, para terminar, comentar brevemente acerca de cómo el hardware sabe qué tipo de direccionamiento debe utilizar con la instrucción que acaba de ser leída desde la memoria principal.

Existen diferentes maneras de solucionar este problema. Una puede ser tener un código de operación distinto para cada método, es decir, tener diferentes códigos de operación para, por ejemplo, suma directa, suma indirecta, suma indexada, etc. Otra forma consiste en hacer que el modo de direccionamiento quede descrito por un campo del formato de instrucción.

En cuanto al microprocesador Z80 podemos indicar que posee un total de 158 tipos de operaciones distintas, pero si consideramos que una misma operación (por ejemplo, la suma), puede realizarse con distintos modos de direccionamiento, llegamos a tener un total de 696 instrucciones distintas.

12.11 Bibliografía

-"Programación del Z80". Autor: Rodnay Zaks. Editorial: Anaya multimedia, 1985.
-"Organización de computadores. Un enfoque estructurado". Autor: Andrew S. Tanenbaum. Editorial Prentice Hall, 1986
-"Fundamentos de los computadores". Autor: Pedro de Miguel Anasagasti. Editorial: Paraninfo, 1988

Problemas propuestos

1.- Un ordenador utiliza un formato de instrucción de 24 bits, 8 de los ellos sirven para indicar el tipo de operación a realizar, y el resto se utiliza para localizar los operandos.
a) ¿Cuántas operaciones distintas pueden realizarse con este formato de instrucción?.
b) ¿Cuántas posiciones de memoria (registros y/o de memoria central) pueden accederse como máximo?.

Supóngase que el campo de dirección se divide en 2 partes iguales de 8 bits.

c) Si cada nuevo campo de 8 bits se dedica a direccionamiento de registros, ¿cuántos registros podremos acceder? y ¿cuántas combinaciones (pares) de registros?.
d) Si el primer grupo de 8 bits se utiliza para seleccionar un registro de índice, y el segundo para especificar una constante, ¿cuántas posiciones de memoria podemos acceder con el direccionamiento indexado?.
e) Si el campo de 16 bits se utiliza para especificar un desplazamiento relativo al PC en complemento a 2, ¿cuáles son las posiciones de memoria accesibles?, ¿cuáles son la posición más baja y la más alta de memoria accesible?. ¿Y si el desplazamiento de 16 bits está en complemento a 1?.

2.- Diseñar un código de operación con extensión para una palabra de 36 bits, que permita lo siguiente:
- 7 instrucciones con dos direcciones de 15 bits, y un campo de dirección de registro de 3 bits de longitud.
- 500 instrucciones con una dirección de 15 bits y un campo de dirección de registros de 3 bits de longitud.
- 50 instrucciones sin dirección de ningún tipo.

3.- Disponemos de una máquina que tiene un formato de instrucción de código de operación con extensión cuya longitud es de 12 bits. ¿Cuáles de las siguientes posibilidades son válidas?:
a) 5 instrucciones de tres direcciones, 21 de dos, 14 de una, y 16 de cero.
b) 8 instrucciones de tres direcciones, 16 de dos, 16 de una y 8 de cero.
c) 7 instrucciones de tres direcciones, 7 de dos, 7 de una y 8 de cero.
d) 6 instrucciones de tres direcciones, 15 de dos, 8 de una y 8 de cero.

4.- El Z80 posee dos registros de índice IX e IY ambos de 16 bits. El modo de direccionamiento indexado del Z80 utiliza una constante de 8 bits expresada en complemento a dos que se suma al registro índice correspondiente. ¿Cuáles son las posiciones de memoria más baja y más alta que se pueden localizar en el Z80 con el modo de direccionamiento indexado?.

5.- La familia de microprocesadores 8088 (utilizado por los PC) usa memoria segmentada. Esto consiste en localizar cada posición de memoria de la siguiente manera:

Pos. mem. = [contenido reg. segmento] * 16 + [desplazamiento].

Como puede observarse se trata de un mecanismo muy semejante al direccionamiento de registro base. El registro segmento tiene 16 bits de longitud. El desplazamiento es otro número natural de 16 bits de longitud. Determinar cuál es la máxima cantidad de memoria direccionable. Dentro de un mismo segmento ¿cuantas celdas de memoria podemos utilizar?.

6.- Las siguientes instrucciones son ensamblador Z80. Construya el código máquina correspondiente, e identifique siempre que sea posible la fuente y el destino de la información, así como el modo de direccionamiento empleado para localizarlas.

LD SP, HL	NOP	JR Z, -3	DEC (HL)
INC A	ADD HL, BC	XOR A	PUSH BC
DEC IX	LD A,(IX+2)	LD (HL), A	RLC (IY-5)
LD (nn), A	CALL nn	INC HL	LD A, 12

7.- Sea una estructura de pila que crece hacia arriba desde su fondo situado en la posición de memoria 100. Los datos que se pueden introducir en la pila son palabras de 8 bits de longitud. El contenido de la zona de memoria alrededor del fondo es:

Dir.	98	99	100	101	102	103	104	105	106
Cont.	22	87	43	92	75	9	250	12	32

Suponiendo que inicialmente el puntero de pila vale 104, determinar los valores que se obtendrían al realizar las siguientes operaciones, así como el valor del puntero de pila antes y después de cada una de ellas (comenzar por la columna de la izquierda):

Extrae Extrae Apila 12	Apila 0 Extrae Extrae	Extrae Extrae Apila 1	Apila 2 Extrae Extrae

8.- Usamos en este problema el microprocesador Z80. Sean las siguientes zonas de memoria:

Dir	Contenido	Dir	Contenido	Dir	Contenido
200	0	505	222	8000	223
201	24	506	12	8001	0
202	254	...	...	8002	56
203	12	1000	0	8003	57
...	...	1001	123	8004	47
500	237	1002	32	8005	-44
501	3	1003	2	8006	-61
502	15	1004	98	8007	FEh
503	245	1005	"w"	8008	00h
504	1	1006	"p"	8009	1aH

La columna Dir. indica la dirección de memoria correspondiente en decimal, y la columna Cont. indica el contenido (un dato de ocho bits). Dicho dato puede ser interpretado como un número en binario natural, complemento a 2, ASCII, en hexadecimal, ...

- si el número no lleva signo, el dato almacenado está en binario natural.
- si el número almacenado lleva signo nosotros supondremos que dicho número está expresado en complemento a 2.
- si es un carácter ASCII entrecomillado, el contenido de la posición de memoria es el código ASCII de dicho carácter,
- si Cont. son dos símbolos de {0,1, ..., 8,9 A,B,C,D,E,F} seguidos de una "H", el dato viene expresado en hexadecimal. Así "b3h" indica el número 11*16 + 3 en base 10.

a) Determine los modos de direccionamiento utilizados por cada instrucción y también cuál es el contenido de los registros/posiciones de memoria después de ejecutar las siguientes instrucciones (los apartados son independientes, es decir, cada uno de ellos observa el contenido original de memoria):

a)
```
LD A, (203)
```

b)
```
LD A, 100
```

c)
```
LD HL, 503
LD A, (HL)
```

d)
```
LD HL, (503)
LD A, (HL)
```

e
```
LD HL, 8000
LD SP, HL
POP BC
POP DE
LD HL, 876
PUSH HL
PUSH BC
PUSH DE
```

f
```
LD A, (1003)
INC A
LD (500),A
```

g
```
LD IX, (500)
LD A, (IX+0)
LD (IX-5), A
LD A, (IX+1)
LD (IX-4), A
```

b) Para la realización de este apartado tendremos en cuenta que la instrucción *JR d* del Z80 (salto relativo) tiene el formato 18d en hexadecimal: 18H es el código de operación, y d es un número de 8 bits que especifica el desplazamiento. Supóngase que el registro PC en un determinado instante vale 201 (en decimal). Calcule dónde se encuentra la siguiente instrucción a ejecutar tras la ejecución de la instrucción JR 9C.