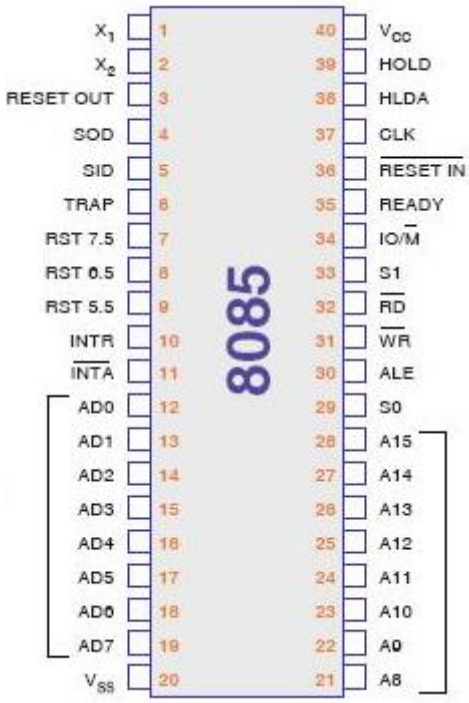
UNIVERSIDAD DE JAÉN
Departamento de Informática



30/09/2013

Referencia Lenguaje 8085

Fundamentos de Arquitectura de Computadores

Alberto Fernández Hilario / Ángel Luis García Fernández – Referencia 8085

Referencia Lenguaje 8085

Guía básica de programación en ensamblador con 8085

Contenido

Arquitectura Interna del 8085

En la siguiente figura se puede observar la organización de los diferentes componentes que se encuentran dentro del procesador 8085. Cada uno de estos componentes se explica a continuación.

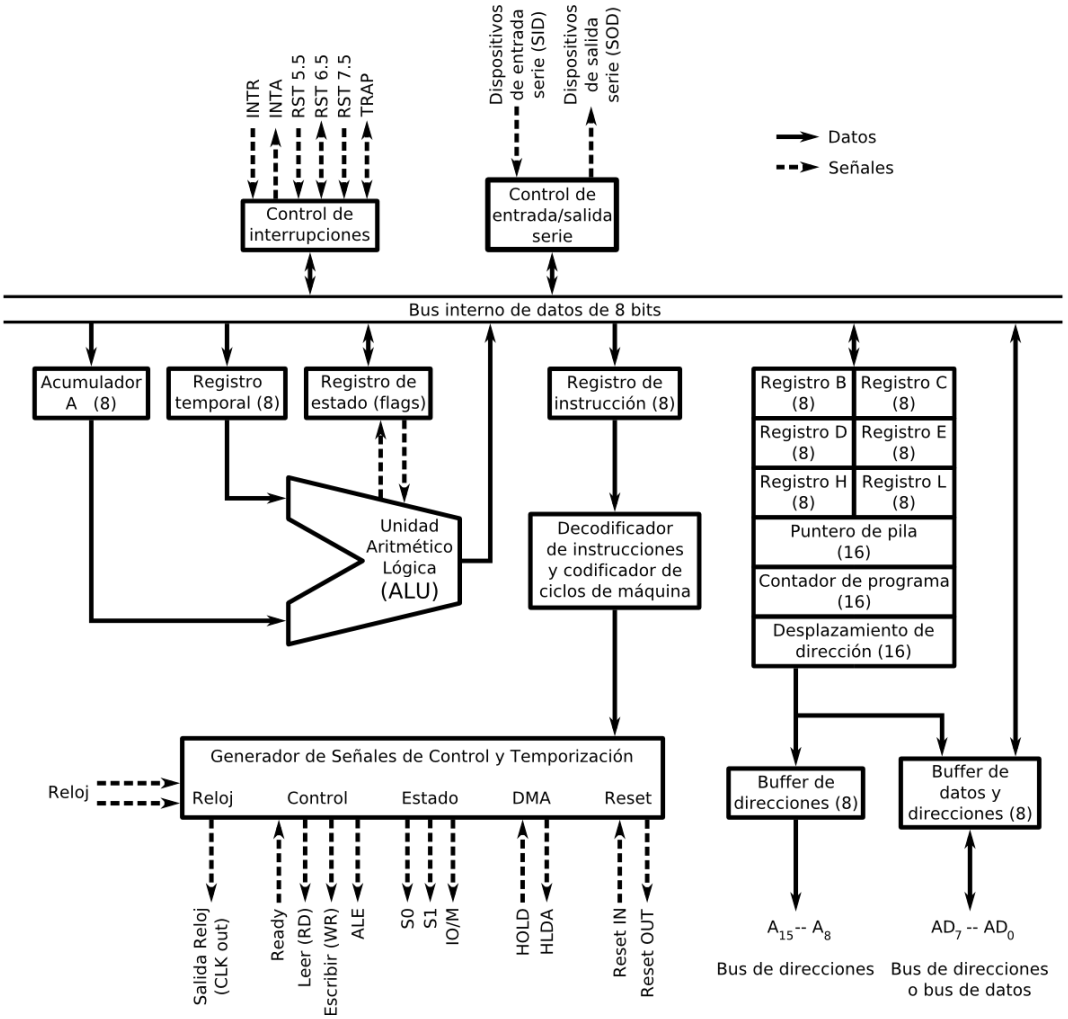


Figura 1. Arquitectura del procesador 8085

Unidad de Control (CU)

Genera señales dentro del microprocesador (µP a partir de ahora) para llevar a cabo la instrucción, que ha sido decodificada. En realidad, esta unidad hace que ciertas conexiones entre los bloques del µP se abran o cierren para que los datos vayan a donde sea necesario, y para que las operaciones de ALU se lleven a cabo.

Unidad Aritmético-Lógica (ALU)

La ALU realiza operaciones numéricas y lógicas como sumas, restas, "Y", "O", etc. Utiliza información almacenada en los registros, la memoria y el acumulador para llevar a cabo las operaciones. El

resultado de la operación realizada siempre se almacena en el acumulador, sobrescribiendo el dato que estuviera almacenado antes.

Registros

El modelo de programación del µP 8085/8080A incluye seis registros, un acumulador, y un registro de estado (flags), como se muestra en la Figura 1. Además, cuenta con dos registros de 16 bits: el puntero de pila y el contador de programa. Se describen brevemente como sigue.

El µP 8085/8080A tiene seis registros de propósito general para almacenar datos de 8 bits, que son identificados como B, C, D, E, H y L, como se muestra en la Figura 1. Se pueden combinar como pares de registros - BC, DE y HL - para llevar a cabo algunas operaciones de 16 bits. El programador puede utilizar estos registros para almacenar o copiar datos en ellos mediante el uso de las instrucciones adecuadas.

Acumulador (A)

El acumulador es un registro de 8 bits que es parte de la unidad aritmético-lógica (ALU). Este registro se utiliza para almacenar datos de 8 bits y para realizar operaciones aritméticas y lógicas. El resultado de cualquier operación se almacena en el acumulador, sobrescribiendo el dato que estuviera almacenado antes. El acumulador es también identificado como registro A.

Registro de estado (flags)

La ALU del µP 8085 incluye cinco biestables que se actualizan después de cada operación de acuerdo a las condiciones del resultado almacenado en el acumulador y otros registros. Se les llama genéricamente flags (banderas), y son los siguientes: Cero (Z), acarreo (CY), Signo (S), paridad (P), y flag auxiliar (AC). Los indicadores más utilizados son cero, acarreo, y signo. El µP utiliza estos flags para comprobar condiciones sobre los datos.

Por ejemplo: después de una suma de dos números, si el resultado es demasiado grande para representarlo con los ocho bits del acumulador, se activa el flag de acarreo (CY), que se establece en uno. Cuando se produce una operación aritmética que da como resultado cero, el flag llamado Cero (Z) se pone a uno.

Estos indicadores tienen una importancia fundamental en el proceso de toma de decisiones del µP. El estado (activado o desactivado) de los flags se comprueba a través de instrucciones software. Por ejemplo: la instrucción JC (Salta si hay acarreo) se utiliza para cambiar la secuencia de un programa cuando el flag CY se ha activado. El conocimiento del registro de estado es esencial en la escritura de programas en lenguaje ensamblador.

Contador de Programa (PC)

Este registro de 16 bits es un puntero a memoria (almacena una dirección de memoria). Las posiciones de memoria tienen direcciones de 16 bits, y es por eso que en este caso la dimensión del registro es mayor.

La función del contador de programa es apuntar a la dirección de memoria donde se encuentra la siguiente instrucción del programa. Cuando una instrucción se ha cargado de la memoria, el contador de programa se incrementa para que apunte a la siguiente instrucción en memoria.

Puntero de pila (SP)

El puntero de pila también es un registro de 16 bits utilizado como puntero de memoria. Este registro almacena una dirección en la memoria de lectura/escritura denominada pila, que se utiliza cuando se hacen llamadas a subrutinas (más adelante se explicará con más detalle).

Registro de instrucción

Este registro almacena temporalmente la instrucción del programa que se está ejecutando en ese momento. Cada instrucción se envía desde la memoria a este registro antes de su ejecución.

Registro de dirección de memoria

Almacena la dirección en memoria, recibida desde el PC, de la siguiente instrucción del programa. Alimenta el bus de direcciones con las ubicaciones de las instrucciones y los datos del programa en ejecución.

Decodificador de instrucciones

Este componente toma la instrucción almacenada en el registro de instrucción y la decodifica, pasando el resultado al generador de señales de control.

Generador de señales de control

Genera señales dentro del µP para llevar a cabo la instrucción que ha sido decodificada. En realidad, hace que ciertas conexiones entre los bloques del µP se abran o cierren para que los datos vayan a donde sea necesario, y para que las operaciones de ALU se lleven a cabo.

Microprograma

¿Cómo sabe el µP lo que significa una instrucción, sobre todo cuando es sólo un número binario? El microprograma en un microprocesador o microcontrolador está escrito por el diseñador de chips, y le dice al µP el significado de cada instrucción para que a continuación pueda llevar a cabo la operación.

Buses de sistema del 8085

Un sistema típico utiliza una serie de buses (grupos de cables), que transmiten números binarios (un bit por cable). Un µP típico se comunica con la memoria y otros dispositivos (de entrada y de salida) utilizando tres buses: el Bus de Direcciones, el Bus de Datos y el Bus de Control.

Bus de direcciones

En el µP 8085, el bus de direcciones tiene 16 cables. Como se utiliza un cable para cada bit, las direcciones de memoria son de 16 bits (ancho del bus). La dirección trasportada alerta al controlador de la memoria para “abrir” la posición designada. Los datos (binarios), se pueden entonces leer o escribir de esa posición.

Con 16 bits podemos representar 2^16 números diferentes, desde 0000000000000000 hasta 1111111111111111. Dado que cada posición de memoria tiene una dirección única, el tamaño del bus de direcciones determina el tamaño de la memoria que se puede utilizar. Para comunicarse con la memoria, el µP envía una dirección en el bus de direcciones, por ejemplo, 0000000000000011 (3 en decimal), al controlador de memoria. El controlador entonces selecciona la posición número 3 para la

lectura o escritura de datos. El bus de direcciones es unidireccional, es decir, las direcciones sólo se envían desde el microprocesador al controlador de memoria, nunca al contrario.

Pregunta: Si tenemos un chip de memoria de tamaño 256 kilobytes (256 x 1024 x 8 bits), ¿cuántos cables necesita el bus de direcciones para poder especificar una dirección en esta memoria? **Nota**: la memoria está organizada en grupos de 8 bits por ubicación, por lo tanto, ¿cuántas posiciones tiene que ser capaz de especificar el bus?

Bus de datos

Trasporta datos en forma binaria entre el µP y otras unidades externas, como la memoria. El tamaño típico es de 8 o 16 bits. Estos datos, se refieren a información, resultados de operaciones aritméticas, etc., entre la memoria y el µP.

El tamaño del bus de datos determina las operaciones que se pueden hacer. Es decir, si sólo disponemos de 8 bits de ancho en el bus de datos, el número más grande que podemos manejar es 11111111 (255 en decimal). Por lo tanto, si queremos operar con datos mayores, tenemos que dividir su representación en trozos de 8 bits, lo que ralentiza la ejecución.

El Bus de Datos también lleva las instrucciones de los programas almacenados en la memoria al microprocesador. El tamaño del bus limita por lo tanto también el número de instrucciones posibles a 256, suponiendo cada una representada por un número.

Este bus sí es bidireccional.

Bus de control

Consta de varias líneas que tienen funciones específicas de coordinación y control de las operaciones del µP. Por ejemplo: la línea IO/M, de un solo dígito binario, controla si la siguiente operación de lectura o escritura de datos se va a hacer en un puerto de entrada/salida o en una posición de memoria. También incluye una línea de reloj para temporización/sincronización, una línea para 'reset', etc. El µP no puede funcionar correctamente sin estas señales de control.

Modos de direccionamiento

Más adelante veremos con detalle las instrucciones del µP 8085, pero ahora vamos a hablar de cómo referirnos a las posiciones disponibles para almacenar datos. Lógicamente, la mayoría de las instrucciones que escribimos en cualquier lenguaje de programación hacen uso de datos disponibles en algún componente del computador (registros, memoria, dispositivos de entrada/salida,...), ya sea para consultar dichos datos o para modificarlos. Por tanto, es necesario hacer referencia a esas fuentes de datos de manera que el µP sepa sin ningún tipo de dudas dónde tiene que acceder. En lenguaje ensamblador hay cuatro posibles formas de referirnos a las fuentes de datos, denominadas modos de direccionamiento:

1. Direccionamiento inmediato
2. Direccionamiento de registros
3. Direccionamiento directo
4. El direccionamiento indirecto

Direccionamiento inmediato

En este modo de direccionamiento, los datos se incluyen en la instrucción, de manera que se usa el dato inmediatamente.

Ejemplo: *MVI A, 82h* *(carga el acumulador con el número 82h)*

Direccionamiento de registros

Los datos están contenidos en alguno de los registros del μP.

Ejemplo: *MOV B, A* *(carga el registro B con el dato almacenado en el acumulador)*

Direccionamiento absoluto (directo)

Normalmente se usa para tomar datos de dispositivos de entrada y almacenarlos en el acumulador, o para enviar los datos almacenados en el acumulador a un dispositivo de salida.

Ejemplo: *IN 00H* *(carga el acumulador con un byte procedente del puerto de entrada/salida 00h)*

Direccionamiento directo (indirecto)

En este modo de direccionamiento no se incluye el dato ni la dirección en que se encuentra éste, sino que se suministra información de dónde se puede consultar la dirección en que se encuentra el dato.

Ejemplo: *LDAX B* *(en la pareja de registros BC está almacenada la dirección de memoria donde se encuentra el dato a cargar en el acumulador)*

La ejecución de este ejemplo sería así:

1. Se toma el dato almacenado en el par de registros BC (supongamos 45h)
2. Se accede a la posición 45h de la memoria, y se toma el dato almacenado ahí (supongamos 65h)
3. Se toma el dato almacenado en la posición de memoria 65h, y se carga en el acumulador

Direccionamiento indirecto (no soportado)

En la instrucción se indica la *dirección de memoria* en la que se encuentra la **dirección de memoria** donde realmente está el dato.

Ejemplo: *MOV B, @17H* *(en la dirección de memoria 0x17 se encuentra una dirección [ej. F3] donde hay que buscar el dato a cargar en el acumulador)*

Clasificación del juego de instrucciones del microprocesador Intel 8085

Una **instrucción** es un patrón binario diseñado dentro de un microprocesador para realizar una función específica. El grupo completo de instrucciones, llamado **juego de instrucciones**, determina qué funciones puede realizar el microprocesador. Estas instrucciones se pueden clasificar en las siguientes cinco categorías funcionales: operaciones de transferencia de datos (copia), operaciones aritméticas, operaciones lógicas, operaciones de salto, y operaciones de control.

Las operaciones del microprocesador relacionadas con la manipulación de datos se pueden resumir en cuatro funciones:

1. La copia de datos
2. Realización de operaciones aritméticas
3. Realización de operaciones lógicas
4. Pruebas de condiciones y alteración de la secuencia del programa

Operaciones de transferencia de datos (copia)

Este grupo de instrucciones copian datos de un lugar llamado fuente a otro lugar llamado destino, sin modificar el contenido de la fuente. En manuales técnicos, el término "*transferencia de datos*" es el utilizado para esta función de copia. Sin embargo, el término *transferencia* es engañoso, dado que crea la impresión de que el contenido de la fuente se destruye cuando en realidad, los contenidos se mantienen sin modificación alguna.

Los diversos tipos de transferencia de datos (copia) se enumeran a continuación junto con ejemplos de cada tipo:

Tipos	**Ejemplos**
Entre registros	Copiar el contenido del registro B en el registro D
Byte de datos específico a un registro o posición de memoria	Cargar el registro B con el byte de datos 32h
Entre una posición de memoria y un registro	Copiar un byte desde la posición de memoria 2000h al registro B
Entre un dispositivo de E/S y el acumulador	Copiar un byte desde la entrada de teclado al acumulador

Operaciones aritméticas

Estas instrucciones realizan operaciones aritméticas como suma, resta, incremento y decremento.

Suma - Cualquier número de 8 bits, o el contenido de un registro, o el contenido de una posición de memoria se pueden sumar al contenido del acumulador, siendo almacenado el resultado en el mismo acumulador. No existe ningún otro par de registros de 8 bits que se puedan sumar directamente (por ejemplo, el contenido del registro B no se puede sumar directamente al contenido del registro C). La instrucción DAD es una excepción, dado que suma datos de 16 bits directamente en parejas de registros.

Resta - Cualquier número de 8 bits, o el contenido de un registro, o el contenido de una posición de memoria se pueden restar del contenido del acumulador, y el resultado será almacenado en el mismo acumulador. La resta se realiza en complemento a 2 y, si el resultado es negativo, se expresa también en complemento a 2. No existe ningún otro par de registros de 8 bits que se puedan restar directamente.

Incremento / decremento - El contenido de 8 bits de un registro o una posición de memoria puede ser incrementado o disminuido en 1. Del mismo modo, el contenido de 16 bits de un par de registros (por ejemplo, BC) se puede aumentar o disminuir en 1. Estos incrementos y decrementos difieren de las operaciones de suma y resta de una manera importante, es decir, pueden llevarse a cabo en cualquiera de los registros o en una ubicación de memoria.

Operaciones lógicas

Estas instrucciones realizan varias operaciones lógicas con el contenido del acumulador.

AND, OR, XOR - Cualquier número de 8 bits, o el contenido de un registro, o de una posición de memoria puede ser operado lógicamente mediante AND, OR o XOR con el contenido del acumulador. El resultado de la operación se almacena en el acumulador.

Desplazamiento - Los bits en el acumulador pueden ser desplazados hacia la izquierda o la derecha a la siguiente posición. Ejemplo: 00010010 desplazado a la izquierda es 00100100

Comparar - Cualquier número de 8 bits, o el contenido de un registro, o una posición de memoria pueden ser comparados mediante las condiciones de igualdad y mayor o menor que, con el contenido del acumulador. El resultado se almacena en los flags del registro de estado.

Complemento - El contenido del acumulador se puede complementar. Todos los 0's son sustituidos por 1's y todos los 1's se sustituyen por 0's.

Operaciones de Salto

Este grupo de instrucciones alteran la secuencia de la ejecución del programa, haciendo que la siguiente instrucción a ejecutar pase a ser una distinta a la que se encuentra en la posición de memoria siguiente a la actual.

Salto - Los saltos condicionales representan un aspecto importante de la toma de decisiones en la programación. Estas instrucciones chequean determinadas condiciones (por ejemplo, el flag cero o el flag de acarreo) y alteran la secuencia del programa cuando se cumple la condición. Además, el conjunto de instrucciones incluye una instrucción llamada *salto incondicional*, que provoca el salto en todo caso.

Llamadas, retorno y reinicio - Son instrucciones que cambian la secuencia de ejecución de un programa, ya sea llamando a una subrutina o regresando de la misma. Las instrucciones de llamada y retorno también pueden estar condicionadas por los valores de los flags del registro de estado.

Operaciones de control

Estas instrucciones controlan funciones de la máquina, tales como detener la ejecución, interrumpir lo que se esté haciendo, o no hacer nada.

Aspectos importantes del juego de instrucciones del Intel 8085 (resumen)

1. En las instrucciones de transferencia de datos, el contenido de la fuente no se destruye, sólo el contenido del destino cambia. Además, la ejecución de estas instrucciones no afecta a los flags del registro de estado. Por ejemplo: si almacenamos 00h en el acumulador, el flag de Cero no cambia.
2. Las operaciones aritméticas y lógicas se realizan con el contenido del acumulador, y los resultados se almacenan en el acumulador (con algunas excepciones). Los flags del registro de estado se ven afectados de acuerdo a los resultados.
3. Cualquier registro o posición de memoria se puede utilizar para el incremento y decremento.
4. La secuencia de ejecución de un programa se puede cambiar con instrucciones de salto condicionales o incondicionales.

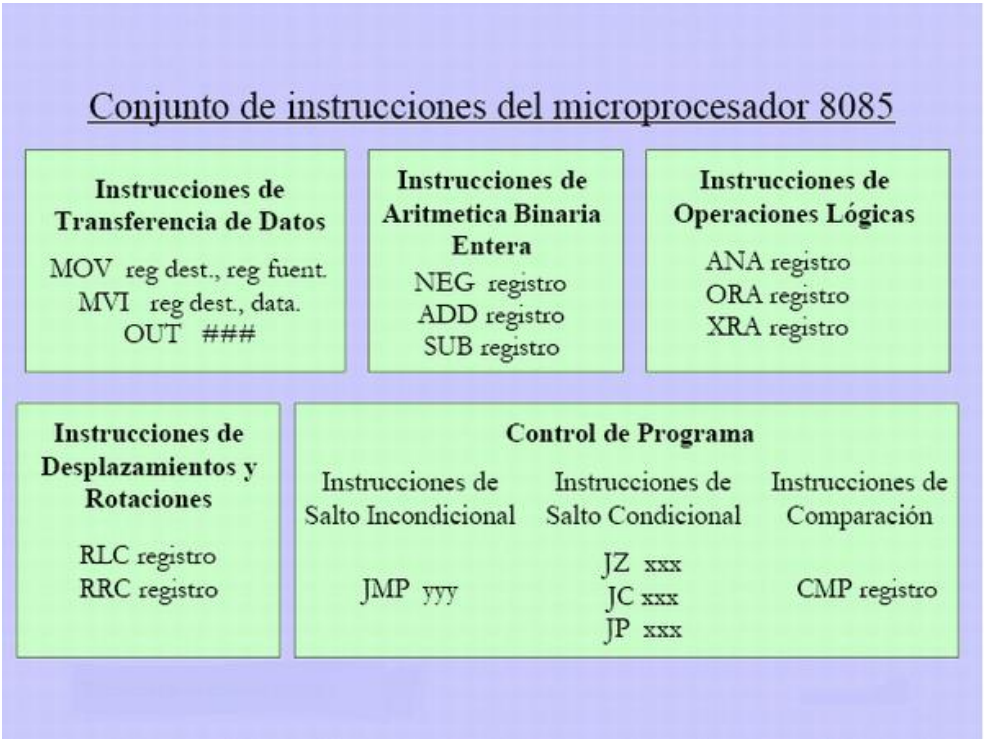


Figura 2. Resumen de algunas de las instrucciones principales del microprocesador 8085 agrupadas según su función.

Formato de las instrucciones

Una **instrucción** es una orden para el µP para realizar una determinada tarea con unos datos específicos. Cada instrucción tiene dos partes: una es la tarea a realizar, denominada **código de operación** (opcode), y la segunda son los datos sobre los que se opera, llamados **operandos**. Los operandos se pueden especificar de varias maneras, utilizando los modos de direccionamiento antes explicados. En algunas instrucciones, sin embargo, el operando está implícito.

Tamaño de las instrucciones

El juego de instrucciones del 8085 se clasifica en los siguientes tres grupos, de acuerdo con el tamaño de la palabra:

1. Instrucciones de una sola palabra o de 1 byte
2. Instrucciones de dos palabras o de 2 bytes
3. Instrucciones de tres palabras o de 3 bytes

En el 8085, "byte" y "palabra" son sinónimos, ya que es un microprocesador de 8 bits. Sin embargo, el tamaño de las instrucciones se indica normalmente en términos de bytes en lugar de palabras.

Instrucciones de un byte

Una instrucción de 1 byte incluye el código de operación y el operando en el mismo byte. El/los operandos son registros internos, y se codifican en la instrucción. Por ejemplo:

Tarea	Opcode	Operandos	Código Binario	Código HEX
Copiar el contenido del acumulador en el registro C	MOV	C, A	0100 1111	4FH
Sumar el contenido del registro B al contenido del acumulador	ADD	B	1000 0000	80H
Invertir (complementar) cada bit en el acumulador	CMA		0010 1111	2FH

Estas instrucciones son instrucciones de 1 byte que realizan tres tareas diferentes. En la primera instrucción, ambos registros operando se especifican. En la segunda instrucción, el operando B se especifica y el acumulador se supone. Del mismo modo, en la tercera instrucción, el acumulador se supone que es el operando implícito. Estas instrucciones se almacenan en 8 bits en formato binario en memoria: una posición de memoria es suficiente.

Instrucciones de dos bytes

En una instrucción de dos bytes, el primer byte especifica el código de operación y el segundo byte especifica un operando. Normalmente, el segundo operando es un byte de datos. Por ejemplo:

Tarea	Opcode	Operandos	Código Binario	Código HEX	
Cargar un dato de 1 byte en el acumulador	MVI	A, Dato	0011 1110	3E	Primer Byte
			Dato	Dato	Segundo Byte

Asumiendo que en este ejemplo el dato que queremos copiar es 32h, la instrucción en lenguaje ensamblador se escribe como:

MVI A, 32h , que en código hexadecimal resulta *3E32h*

La instrucción requeriría por tanto dos posiciones de memoria para ser almacenada. Esto es un ejemplo de direccionamiento inmediato.

Otro ejemplo:

OUT 01h

donde 01h es una dirección de 8 bits de un dispositivo de salida. En este caso el byte no representa a los datos, sino que apunta directamente a donde se encuentran, lo que se llama direccionamiento directo.

Instrucciones de tres bytes

En una instrucción de tres bytes, el primer byte especifica el código de operación y los dos siguientes bytes especifican una dirección de 16 bits. Es necesario tener en cuenta que al representar una dirección de memoria de 16 bits con 2 bytes, el segundo byte de la instrucción guarda los bits menos significativos, y el tercer byte guarda los bits más significativos.

Por ejemplo:

Tarea	Opcode	Operandos	Código Binario	Código HEX	
Trasferir la secuencia del programa a la dirección de memoria 2085H	JMP	2085H	1100 0011	C3	Primer Byte
			1000 0101	85	Segundo Byte
			0010 0000	20	Tercer Byte

Es decir, la instrucción JMP 2085h se almacena en memoria como C3 85 20h, y por tanto requeriría tres posiciones de memoria para su almacenamiento. Esta instrucción es otro ejemplo de direccionamiento inmediato.

Debe quedar claro que, aunque en ensamblador escribimos los 16 bits de datos de la manera "natural" (es decir: los más significativos primero, y los menos significativos después), internamente la computadora almacena primero los menos significativos, y después los más significativos.

Etiquetas

En ensamblador se pueden utilizar etiquetas para identificar posiciones en el código. Así, no es necesario saber en qué dirección de memoria está almacenada una instrucción concreta, sino que basta recordar su etiqueta. Esto es especialmente útil a la hora de utilizar instrucciones de salto, que hacen que el flujo normal de ejecución del programa cambie a

Para poner una etiqueta, basta utilizar un nombre sin espacios y que no empiece por un número, y que no sea igual al nombre de ninguna instrucción ni registro o flag del procesador. Por ejemplo:

```
JMP  INICIO          (el flujo de ejecución salta a la posición etiquetada como INICIO)

MOV  A, B            (estas dos instrucciones no se ejecutan, por la acción de la instrucción
ADD  C                anterior)

INICIO: MVI  A, 10h  (se asigna la etiqueta INICIO a la dirección de memoria en que se
                      almacene la instrucción)
```

Juego Completo de Instrucciones para el microprocesador Intel 8085

A continuación se detallan todas las instrucciones que pueden emplearse para la programación en lenguaje ensamblador con respecto al µP 8085. Tal y como hemos explicado anteriormente, estas instrucciones se dividen en diferentes bloques con respecto a la función que realizan.

En este listado se utilizan las siguientes abreviaturas para describir los operandos:

R: Registro (Cualquiera de los registros A -> H)
- **Rd**: Registro de Destino (el que va a recibir los datos)
- **Rs**: Registro fuente (el que contiene los datos que se procesan)

M: Dirección de memoria de 16 bits almacenada en el par de registros HL. Se indica como "m" en ensamblador (por ejemplo: INR M)

B/D: par de registros BC, o DE
B/D/H: par de registros BC, DE o HL

B/D/H/A: par de registros BC, DE, HL o (acumulador, registro de estado)

dato: Un byte de datos expresado como un número hexadecimal
dato-16: Dos bytes de datos expresados como un número hexadecimal
etiqueta: Una etiqueta, tal y como se ha descrito antes

Instrucciones de Transferencia de Datos

Función	Opcode	Operandos	Descripción
Copiar desde el origen al destino	**MOV**	Rd, Rs M, Rs Rd, M	Esta instrucción copia el contenido de la ubicación de origen en la ubicación de destino; el contenido de la ubicación de origen no se modifica.
Movimiento inmediato de 8 bits	**MVI**	Rd, dato M, dato	Se almacena un dato de 8 bits en la ubicación de destino.
Carga de datos en el acumulador	LDA	dato-16	El contenido de una posición de memoria especificada por una dirección de 16 bits en el operando se copia en el acumulador. El contenido de la posición de memoria no se altera.
Carga indirecta del acumulador	**LDAX**	B/D	Esta instrucción copia el contenido de la posición de memoria indicada por la dirección de 16 bits almacenada en el par de registros operando en el acumulador. No se altera el contenido de la posición de memoria ni el del par de registros.
Carga inmediata de un par de registros	LXI	B/D/H, dato-16	La instrucción carga el dato de 16 bits en el par de registros designados en el operando.
Carga directa de los registros H y L	LHLD	dato-16	La instrucción copia el contenido de la posición de memoria señalado por la dirección de 16 bits en el registro L y copia el contenido de la siguiente ubicación de memoria en el registro H. El contenido de las posiciones de memoria de origen no se altera.
Almacenamiento directo del acumulador	STA	dato-16	El contenido del acumulador se copia en la ubicación de memoria especificada por el operando.
Almacenamiento indirecto del acumulador	**STAX**	B/D/H	El contenido del acumulador se copia en la posición de memoria indicada por los 16 bits almacenados en el par de registros que se pasa como operando.
Almacenamiento directo de los registros H y L	SHLD	dato-16	El contenido del registro L se almacena en la posición de memoria especificada por los 16 bits del operando, y el contenido del registro H se almacena en la siguiente posición de memoria. El contenido de los registros HL no se altera.
Intercambio de H y L con D y E	XCHG	ninguno	El contenido del registro H se intercambia con el contenido del registro D, y el contenido del registro L se intercambia con el contenido del registro E.

Copia los registros H y L al puntero de pila	SPHL	ninguno	Esta instrucción carga el contenido de los registros H y L en el registro del puntero de pila. El contenido del registro H se considera como los 8 bits más significativos de la dirección, y el contenido del registro L se considera como los 8 bits menos significativos.
Intercambio de H y L con la parte superior de la pila	XTHL	ninguno	Intercambia los 8 bits almacenados en el registro L con los de la posición de memoria indicada por el puntero de pila (SP), y los 8 bits del registro H con los de la posición de memoria siguiente a la indicada por el puntero de pila (SP+1). El puntero de pila no se modifica.
Meter el contenido de un par de registros en la pila	PUSH	B/D/H/A	El contenido del par de registros indicado como operando se copia en la pila de la siguiente forma: el puntero de pila (SP) se decrementa, y el contenido del primer registro del par (B, D, H, A) se copia en la ubicación que se corresponde con la dirección almacenada en el SP; luego, el puntero de pila se decrementa de nuevo, y el contenido del segundo registro del par (C, E, L, estado) se copia en la nueva ubicación almacenada en el SP.
Sacar datos de la pila a un par de registros	POP	B/D/H/A	El contenido de la posición de memoria almacenada en el puntero de pila (SP) se copia en el segundo registro del par (C, E, L, registro de estado) que se indica como operando. El SP se incrementa en 1, y el contenido almacenado en la posición de memoria que queda en el SP se copia en el segundo registro del par (B, D, H, A). Una vez hecho esto, el SP de nuevo se incrementa en 1.
Escribir los datos del acumulador en un puerto de entrada/salida	**OUT**	dato	El contenido del acumulador se copia en el puerto E/S especificado por el operando de 8 bits.
Cargar datos en el acumulador desde un puerto de entrada/salida	**IN**	dato	Se toma un byte del puerto E/S especificado por el operando de 8 bits, y se almacena en el acumulador.

Instrucciones Aritméticas

Función	Opcode	Operando	Descripción
Sumar registro o memoria con el acumulador	**ADD**	Rs	Si el operando es un registro, se toma su contenido y se suma al acumulador. Si el operando es M, se accede a la posición de memoria indicada por los 16 bits almacenados en el par HL, y se suma el contenido de dicha posición al acumulador. El resultado se guarda en el acumulador, y se actualizan los flags del registro de estado según dicho resultado.
		M	
Sumar registro o memoria con el acumulador usando acarreo	ADC	Rs	Igual que ADD, salvo que también se suma el valor del flag de acarreo (CY) al resultado. Al terminar, el resultado se guarda en el acumulador y se actualizan los flags de estado.
		M	
Suma inmediata con acumulador	ADI	dato	Los 8 bits del operando se suman al contenido del acumulador y el resultado se almacena en éste. Los flags de estado se actualizan según el resultado de la suma.
Suma inmediata con acumulador usando acarreo	ACI	dato	Los 8 bits del operando y el valor del flag de acarreo (CY) se suman al acumulador. Al terminar, el resultado se guarda en el acumulador y se actualizan los flags de estado.
Sumar un par de registros con los registros H y L	DAD	B/D	El contenido de 16 bits del par de registros especificado se suma al contenido del par de registros HL, y el resultado se almacena en este último. El contenido del par de registros de origen no se altera. Si el resultado no se puede almacenar en 16 bits, se activa el flag de acarreo (CY). El resto de flags no se modifican en ningún caso.
Restar registro o memoria del acumulador	**SUB**	Rs	Si el operando es un registro, se toma su contenido y se resta del acumulador. Si el operando es M, se accede a la posición de memoria indicada por los 16 bits almacenados en el par de registros HL, y se resta el contenido de dicha posición al acumulador. El resultado se guarda en el acumulador, y se actualizan los flags del registro de estado según dicho resultado.
		M	
Restar registro o memoria del acumulador usando acarreo	SBB	Rs	Igual que SUB, salvo que también se resta el valor del flag de acarreo (CY) del acumulador. Al terminar, el resultado se guarda en el acumulador y se actualizan los flags de estado.
		M	
Resta inmediata del acumulador	SUI	dato	Los 8 bits del operando se restan del contenido del acumulador y el resultado se almacena en éste. Los flags de estado se actualizan según el resultado de la resta.

Resta inmediata del acumulador usando acarreo	SBI	dato	Los 8 bits del operando y el valor del flag de acarreo (CY) se restan del acumulador. Al terminar, el resultado se guarda en el acumulador y se actualizan los flags de estado.
Incremento de registro o de la memoria en uno	**INR**	R	Si el operando es un registro, se incrementa su contenido en 1 y el resultado se almacena en el mismo lugar. Si el operando es M, se toma el contenido del par de registros HL como la dirección de memoria cuyo contenido se incrementa en 1.
		M	
Incremento de un par de registros en 1	**INX**	B/D/H	El contenido del par de registros designado se incrementa en 1, y el resultado se almacena en el mismo lugar.
Decremento de registro o de la memoria en uno	**DCR**	R	Si el operando es un registro, se decrementa en 1 su contenido, y el resultado se almacena en el mismo lugar. Si el operando es M, se toma el contenido del par de registros HL como la dirección de memoria cuyo contenido se decrementa en 1.
		M	
Decremento de un par de registros en 1	**DCX**	B/D/H	El contenido del par de registros designado se decrementa en 1, y el resultado se almacena en el mismo lugar.
Ajuste decimal del acumulador	DAA	ninguno	El contenido del acumulador se cambia de un número binario de 8 bits a dos números de 4 bits con codificación BCD. Esta es la única instrucción que utiliza el flag auxiliar (AC). La conversión es como sigue: - Cada cifra del hexadecimal que representa el valor del acumulador se convierte a un número de 4 bits de la manera habitual. - Si la cifra menos significativa del hexadecimal es mayor de 9, o si el flag AC está activo, se suma 0110 al número de 4 bits que la representa. El acarreo resultante se suma al resultado de convertir la cifra más significativa. - Si la cifra más significativa del hexadecimal es mayor de 9, o si el flag CY (acarreo) está activo, se suma 0110 al número de 4 bits que la representa. El acarreo resultante se guarda en el flag CY.

Instrucciones de ramificación o salto

<table>
<tr><th>Función</th><th colspan="4">Opcode</th><th>Operando</th><th>Descripción</th></tr>
<tr><td>Saltar sin condiciones</td><td colspan="4">JMP</td><td>dato-16
etiqueta</td><td>Hace que el flujo de ejecución del programa continúe a partir de la instrucción alojada en la dirección de memoria que se pasa como operando.</td></tr>
<tr><td>Saltar según una condición</td><td colspan="4"><table><tr><th>Op</th><th>Descripción</th><th>Condición</th></tr><tr><td>JC</td><td>Saltar si acarreo</td><td>CY = 1</td></tr><tr><td>JNC</td><td>Saltar si no acarreo</td><td>CY = 0</td></tr><tr><td>JP</td><td>Saltar si positivo</td><td>S = 0</td></tr><tr><td>JM</td><td>Saltar si negativo</td><td>S = 1</td></tr><tr><td>JZ</td><td>Saltar si cero</td><td>Z = 1</td></tr><tr><td>JNZ</td><td>Saltar si no cero</td><td>Z = 0</td></tr><tr><td>JPE</td><td>Saltar con paridad par</td><td>P = 1</td></tr><tr><td>JPO</td><td>Saltar con paridad impar</td><td>P = 0</td></tr></table></td><td>dato-16
etiqueta</td><td>Todas estas instrucciones son similares. Hacen que el flujo de ejecución del programa continúe a partir de la instrucción alojada en la posición de memoria que se le pasa como operando, siempre que se cumpla la condición correspondiente en los flags de estado.</td></tr>
<tr><td>Llamada incondicional a subrutina</td><td colspan="4">CALL</td><td>dato-16
etiqueta</td><td>Hace que el flujo de ejecución del programa continúe a partir de la instrucción alojada en la dirección de memoria que se pasa como operando. Antes de hacer el salto, la dirección de memoria de la instrucción siguiente al CALL (almacenada en el contador de programa) se guarda en la pila de la misma manera que si hiciéramos un PUSH.</td></tr>
<tr><td>Llamada condicional a subrutina</td><td colspan="4"><table><tr><th>Op</th><th>Descripción</th><th>Condición</th></tr><tr><td>CC</td><td>Llamada si acarreo</td><td>CY = 1</td></tr><tr><td>CNC</td><td>Llamada si no acarreo</td><td>CY = 0</td></tr><tr><td>CP</td><td>Llamada si positivo</td><td>S = 0</td></tr><tr><td>CM</td><td>Llamada si negativo</td><td>S = 1</td></tr><tr><td>CZ</td><td>Llamada si cero</td><td>Z = 1</td></tr><tr><td>CNZ</td><td>Llamada si no cero</td><td>Z = 0</td></tr><tr><td>CPE</td><td>Llamada con paridad par</td><td>P = 1</td></tr><tr><td>CPO</td><td>Llamada con paridad impar</td><td>P = 0</td></tr></table></td><td>dato-16
etiqueta</td><td>Todas estas instrucciones son similares. Hacen que el flujo de ejecución del programa continúe a partir de la instrucción alojada en la posición de memoria que se le pasa como operando, siempre que se cumpla la condición correspondiente en los flags de estado. Antes de hacer el salto, la dirección de memoria de la instrucción siguiente al CALL (almacenada en el contador de programa) se guarda en la pila de la misma manera que si hiciéramos un PUSH.</td></tr>
</table>

<table>
<tr><td>Retorno incondicional desde subrutina</td><td colspan="3">RET</td><td>ninguno</td><td>El flujo de ejecución vuelve al punto en el que estaba antes de la llamada a la subrutina. Para esto, se toma la dirección de la instrucción siguiente al CALL que se almacenó en la pila anteriormente (como si hiciéramos un POP), y se copia en el contador de programa.</td></tr>
<tr><td>Retorno condicional desde subrutina</td><td colspan="3"><table>
<tr><th>Op</th><th>Descripción</th><th>Condición</th></tr>
<tr><td>RC</td><td>Retorno si acarreo</td><td>CY = 1</td></tr>
<tr><td>RNC</td><td>Retorno si no acarreo</td><td>CY = 0</td></tr>
<tr><td>RP</td><td>Retorno si positivo</td><td>S = 0</td></tr>
<tr><td>RM</td><td>Retorno si negativo</td><td>S = 1</td></tr>
<tr><td>RZ</td><td>Retorno si cero</td><td>Z = 1</td></tr>
<tr><td>RNZ</td><td>Retorno si no cero</td><td>Z = 0</td></tr>
<tr><td>RPE</td><td>Retorno con paridad par</td><td>P = 1</td></tr>
<tr><td>RPO</td><td>Retorno con paridad impar</td><td>P = 0</td></tr>
</table></td><td>ninguno</td><td>Todas estas instrucciones son similares. Hacen que el flujo de ejecución vuelva al punto en el que estaba antes de la llamada a la subrutina, siempre que se cumpla la condición correspondiente en los flags de estado. Para esto, se toma la dirección de la instrucción siguiente al CALL que se almacenó en la pila anteriormente (como si hiciéramos un POP), y se copia en el contador de programa.</td></tr>
<tr><td>Cargar el contador de programa con el contenido de HL</td><td colspan="3">PCHL</td><td>ninguno</td><td>El contenido del par de registros HL se copia en el contador de programa. El contenido de H se toma como el byte más significativo, y el contenido de L como el byte menos significativo.</td></tr>
<tr><td>Reiniciar</td><td colspan="3">RST</td><td>0-7</td><td>Estas instrucciones están normalmente asociadas a interrupciones que se activan en la computadora a través de hardware externo, aunque también se pueden incluir como una instrucción más en un programa. Provocan el salto del flujo de ejecución del programa a las instrucciones alojadas en una dirección de memoria concreta. Las posibilidades son:

Instrucción — Dirección de salto
RST 0 — 0000H
RST 1 — 0008H
RST 2 — 0010H
RST 3 — 0018H
RST 4 — 0020H
RST 5 — 0028H
RST 6 — 0030H
RST 7 — 0038H

El 8085 tiene cuatro interrupciones adicionales que generan instrucciones RST internamente y por lo tanto no requieren ningún hardware externo para lanzarlas. Estas instrucciones y sus direcciones de salto son:

Interrupción — Dirección de salto
TRAP — 0024H
RST 5.5 — 002CH
RST 6.5 — 0034H
RST 7.5 — 003CH</td></tr>
</table>

Instrucciones Lógicas

Función	Opcode	Operando	Descripción
Comparación de registro o memoria con acumulador	**CMP**	Rs	Si el operando es un registro, se compara su contenido con el contenido del acumulador, y se actualizan los flags de estado. Si el operando es M, se toma el contenido del par de registros HL como la dirección de memoria cuyo contenido se compara con el acumulador. Ni el acumulador ni el operando se modifican. El resultado de la comparación se observa en los flags de estado: si A < operando: se activa el flag CY si A = operando: se activa el flag Z si A > operando: se desactivan los flags CY y Z
		M	
Comparación inmediata con acumulador	CPI	dato	El operando se compara con el contenido del acumulador. Los valores que se comparan no cambian. El resultado de la comparación se observa en los flags de estado: si A < operando: se activa el flag CY si A = operando: se activa el flag Z si A > operando: se desactivan los flags CY y Z
AND de registro o memoria con acumulador	ANA	Rs	Si el operando es un registro, se calcula el AND lógico de su contenido con el contenido del acumulador. Si el operando es M, se toma el contenido del par de registros HL como la dirección de memoria cuyo contenido se opera con el acumulador. El resultado de la operación se guarda en el acumulador y los flags S, Z y P se actualizan según el resultado, mientras que el flag CY se pone a 0, y el flag AC se pone a 1.
		M	
AND inmediato con acumulador	ANI	dato	Se calcula el AND lógico del contenido del acumulador con el operando, y el resultado se coloca en el acumulador. Los flags S, Z y P se actualizan según el resultado de la operación, mientras que el flag CY se pone a 0, y el flag CA se pone a 1.
OR exclusivo de registro o memoria con el acumulador	XRA	Rs	Si el operando es un registro, se calcula el OR exclusivo de su contenido con el contenido del acumulador. Si el operando es M, se toma el contenido del par de registros HL como la dirección de memoria cuyo contenido se opera con el acumulador. El resultado de la operación se guarda en el acumulador y los flags S, Z y P se actualizan según el resultado, mientras que los flags CY y AC se ponen a 0.
		M	

OR exclusivo inmediato con el acumulador	XRI	dato	Se calcula el OR exclusivo del contenido del acumulador con el operando, y el resultado se coloca en el acumulador. Los flags S, Z y P se actualizan según el resultado de la operación, mientras que los flags CY y AC se ponen a 0.
OR de registro o memoria con el acumulador	ORA	Rs	Si el operando es un registro, se calcula el OR de su contenido con el contenido del acumulador. Si el operando es M, se toma el contenido del par de registros HL como la dirección de memoria cuyo contenido se opera con el acumulador. El resultado de la operación se guarda en el acumulador y los flags S, Z y P se actualizan según el resultado, mientras que los flags CY y AC se ponen a 0.
		M	
OR inmediato con el acumulador	ORI	dato	Se calcula el OR lógico del contenido del acumulador con el operando, y el resultado se coloca en el acumulador. Los flags S, Z y P se actualizan según el resultado, mientras que los flags CY y AC se ponen a 0.
Desplazamiento del acumulador a la izquierda	**RLC**	ninguno	Cada bit del acumulador se desplaza a la izquierda una posición. El bit 7 se coloca en la posición del bit 0, así como en el flag CY. El resto de flags no cambian.
Desplazamiento del acumulador a la derecha	**RRC**	ninguno	Cada bit del acumulador se desplaza a la derecha una posición. El bit 0 se coloca en la posición del bit 7, así como en el flag CY. El resto de flags no cambian.
Desplazamiento del acumulador a la izquierda con acarreo	**RAL**	ninguno	Cada bit del acumulador se desplaza a la izquierda una posición. El bit 7 se coloca en la posición del flag CY, y el valor que tuviera éste pasa a la posición del bit 0. El resto de flags no cambian.
Desplazamiento del acumulador a la derecha con acarreo	**RAR**	ninguno	Cada bit del acumulador se desplaza a la derecha una posición. El bit 0 se coloca en la posición del flag CY, y el valor que tuviera éste pasa a la posición del bit 7. El resto de flags no cambian.
Complemento del acumulador	CMA	ninguno	El contenido del acumulador se complementa (se cambian 0's por 1's y viceversa). Ningún flag cambia.
Complemento del acarreo	CMC	ninguno	El contenido del flag CY se complementa. Ningún otro flag cambia.
Ajustar acarreo	STC	ninguno	El flag CY se pone a 1. Ningún otro flag cambia.

Instrucciones de Control

Función	Opcode	Operando	Descripción
Ninguna operación	NOP	ninguno	No se realiza ninguna operación. La instrucción se carga desde memoria y se decodifica. Sin embargo, no se ejecuta ninguna operación.
Detener y entrar en estado de espera	**HLT**	ninguno	El procesador termina de ejecutar la instrucción actual y se detiene. Se necesita una interrupción o un reset para salir del estado de espera.
Deshabilitar interrupciones	DI	ninguno	Se desactivan todas las interrupciones excepto la TRAP. Ninguna otra interrupción será atendida.
Habilitar interrupciones	EI	ninguno	Todas las interrupciones son habilitadas de nuevo. Cada vez que el sistema atiende una interrupción, se deshabilitan automáticamente todas las interrupciones excepto la TRAP, por lo que es necesario ejecutar esta instrucción para habilitar el resto.
Leer la máscara de interrupciones	RIM	ninguno	Esta instrucción sirve tanto para cargar el estado de las interrupciones 7.5, 6.5 y 5.5, como para leer datos de la entrada serie. Se cargan ocho bits en el acumulador con la siguiente interpretación. Bit 0: si vale 1, la interrupción 5.5 está habilitada Bit 1: si vale 1, la interrupción 6.5 está habilitada Bit 2: si vale 1, la interrupción 7.5 está habilitada Bit 3: si vale 1, interrupciones están habilitadas Bit 4: si vale 1, hay interrupciones pendientes Bit 5: si vale 1, hay interrupciones pendientes Bit 6: si vale 1, hay interrupciones pendientes Bit 7: bit leído de la entrada serie
Ajustar la máscara de interrupciones	SIM	Ninguno	Esta instrucción se utiliza tanto trabajar con las interrupciones 7.5, 6.5 y 5.5 del 8085 como para escribir datos por la salida serie. Se interpreta el contenido del acumulador como sigue: Bit 0: si vale 1, la interrupción 5.5 se habilita. Si vale 0, se deshabilita Bit 1: si vale 1, la interrupción 6.5 se habilita. Si vale 0, se deshabilita Bit 2: si vale 1, la interrupción 7.5 se habilita. Si vale 0, se deshabilita Bit 3: si vale 1, se consideran los bits 0, 1 y 2. Si vale 0, estos bits no se tienen en cuenta Bit 4: si vale 1, la interrupción 7.5 se desactiva Bit 5: sin uso Bit 6: si vale 1, se activa la salida serie. Es necesario que este bit esté activo para que se use el valor del bit 7 Bit 7: si el bit 6 vale 1, se toma este bit y se “saca” por la línea de salida serie