

Ensamblador para 8085

5.1. Introducción

Si examinamos el contenido de la memoria de un computador, un programa aparece como una serie de dígitos hexadecimales indistinguibles unos de otros. El procesador o CPU interpreta estos dígitos como códigos de instrucción, direcciones o datos.

Sería posible escribir un programa en esta forma, pero resultaría un proceso lento y costoso. Por ejemplo, el siguiente programa se almacena en la memoria como se muestra:

repite:	MOV A, M CPI 0h JNZ fin MVI M, FFh INX H JMP repite fin:	0100 7Eh 0101 FEh 00h 0104 C2h 0Dh 01h 0107 36h FFh 0109 23h 010A C3h 00h 01h 010D 76h
---------	--	--

- El byte 7E es interpretado por el procesador como el código de la instrucción MOV de transferencia de memoria (indicada por HL) al registro A.
- Los bytes FE 00 se corresponden con el código de la instrucción de comparación CPI, del acumulador con el dato inmediato 00h.
- Los bytes C2 0D 01 indican una instrucción de salto condicional a la dirección 010Dh.
- Los bytes 36 FF realizan la transferencia de FF a la memoria.
- El byte 23 indica que el par de registros HL debe incrementarse como si fueran un registro de 16 bits.
- Los bytes C3 00 01 indican un salto incondicional a la dirección 1000h.
- El byte 76 es la instrucción HALT o parada del procesador.

El texto de izquierda es un programa escrito en ensamblador, mientras que el de la derecha es un volcado directo de la memoria. Las diferencias entre ambos son obvias en cuanto a términos de legibilidad.

El programa anterior realiza una tarea muy sencilla. Utilizando el par de registros HL como dirección de memoria carga el contenido en el acumulador. Si el contenido es diferente de 00 termina, si no, cambia el contenido por FFh y continua por la siguiente dirección. Incluso en un programa tan simple vemos lo costoso que puede ser trabajar directamente sobre la memoria.

Sin embargo, existe un problema añadido. Supongamos que deseamos introducir una instrucción más al programa propuesto. Por ejemplo, supongamos que quiere especificar una dirección inicial de memoria en HL. Esto se puede hacer con la secuencia de dígitos hexadecimales: 21 LL HH, donde LL representa la parte baja de la dirección de memoria y HH la alta.

LXI H,A000h	0100	21h 00h A0h
repite: MOV A,M	0103	7Eh
CPI 0h	0104	FEh 00h
JNZ fin	0107	C2h 10h 01h
MVI M, FFh	010A	36h FFh
INX H	010C	23h
JMP repite	010D	C3h 03h 01h
fin: HLT	0110	76h

En negrita se han marcado los cambios sobre el programa original. Como vemos, en la memoria la introducción de una nueva instrucción da lugar a efectos laterales que obligan a modificar otras del programa. El riesgo de cometer un fallo es mayor.

La ilegibilidad del programa agrava el riego de error si se intentan añadirse más instrucciones y efectuar más cambios.

Para evitar esta forma engorrosa y tediosa de trabajar, sobre todo en programas de cierta complejidad, el primer paso está en utilizar el lenguaje ensamblador, que

proporciona una notación de las instrucciones completamente legible, y evita al programador tener que referirse a direcciones de memoria específicas.

El lenguaje ensamblador es, en síntesis, una secuencia de instrucciones que se convierten a un código hexadecimal ejecutable por la máquina a través de un programa llamado Ensamblador.



El Ensamblador convierte el programa fuente, escrito en lenguaje ensamblador, en su equivalente en hexadecimal, denominado programa objeto. El programa objeto es muy similar a la representación en memoria que tendrá el programa.

5.2. Sintaxis

En un programa ensamblador se distinguirán los siguientes tipos de elementos: Directivas, Instrucciones de ensamblaje e Instrucciones de la maquina.

- DIRECTIVAS:

Las Directivas ofrecen información al ensamblador sobre el tipo de elementos que se va a encontrar a continuación y la dirección de memoria donde debe disponerlos (si corresponde). Se caracterizan por ir precedidas por un punto.

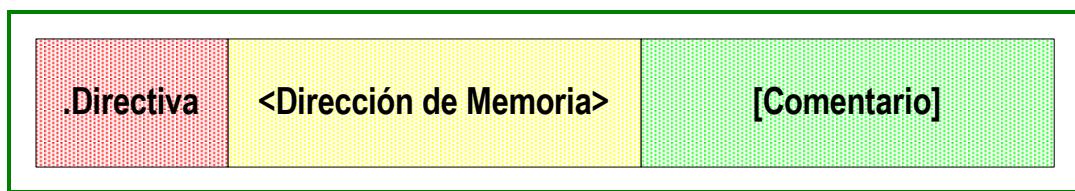


Figura 5.1. Estructura de una Directiva.

Disponemos de tres directivas distintas. Estas directivas nos permiten hacer definiciones (*define*), introducir datos (*data*) e introducir instrucciones (*org*). Cada directiva declara, por tanto, un bloque dentro del programa.

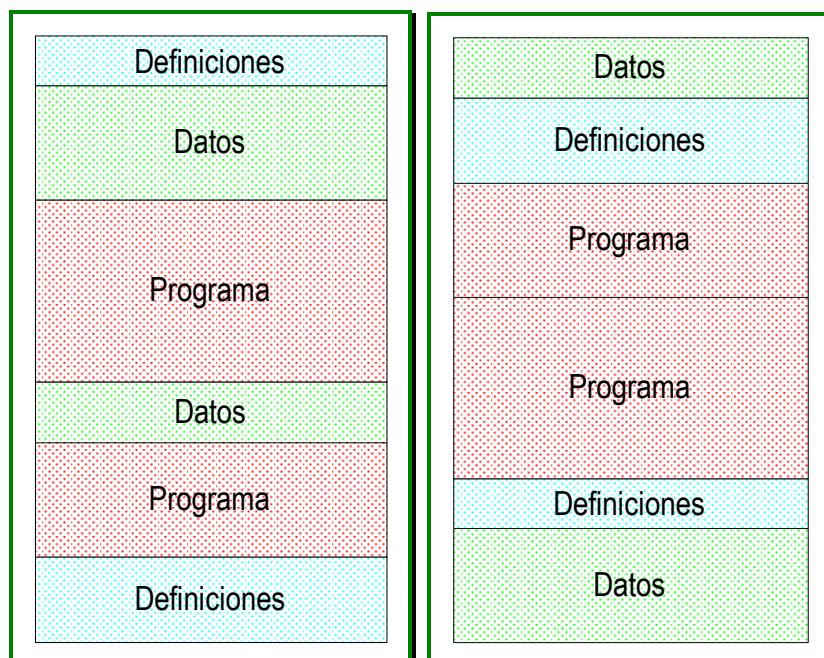


Figura 5.2. Organización Aleatoria de Bloques.

La disposición de bloques dentro del programa ensamblador no está sujeta a ningún tipo de restricción inicial. Además, es posible declarar varios bloques de un mismo tipo.

Igualmente, ningún bloque es imprescindible. Se pueden construir programas sin declaraciones, datos e, incluso, instrucciones. Aunque esto último parezca poco razonable nos puede llegar a ser útil si se quiere únicamente introducir datos en la memoria para usarlos con otro programa. En último extremo podremos construir un programa vacío, que será ensamblado como un programa vacío.

Generalmente, la forma más usual de un programa será la siguiente:

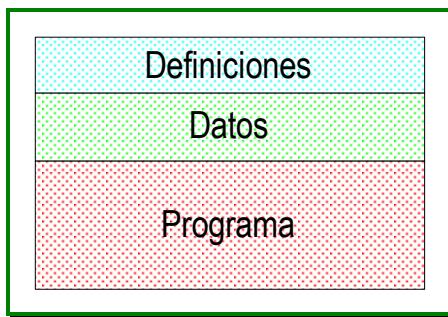


Figura 5.3. Organización Usual de Bloques.



¿La distribución usual por bloques es obligatoria?

No, puede organizar las directivas a su gusto. La distribución usual es la recomendada. La distribución representada en la figura anterior se obtiene empleando las directivas por este orden: DEFINE, DATA y por último ORG.

- **INSTRUCCIONES DE ENSAMBLAJE:**

Son un tipo de instrucciones especiales que únicamente se tienen en cuenta en el proceso de ensamblaje del programa, pero que realmente no tienen ejecución dentro de la máquina, esto es, una vez terminado aquel proceso.

Las instrucciones de ensamblaje se emplean únicamente en el proceso de ensamblaje. De esta forma, no tienen una aparición explícita dentro del código objeto, sin embargo si aparecen de forma implícita.

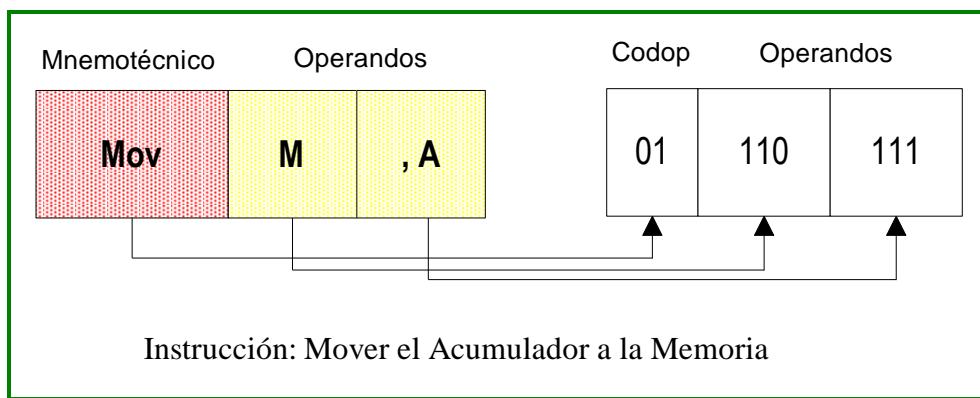
Las instrucciones de ensamblaje difieren según el bloque, o directiva previamente declarada, en la que estemos. Por ello, veremos cada una de ellas dentro de su ámbito correspondiente. Por ahora solo diremos que hay de dos clases, una para hacer definiciones y otra para declarar datos.

- INSTRUCCIONES DE LA MAQUINA:

A diferencia de las anteriores, las instrucciones de la maquina, o simplemente instrucciones, “corren” o se ejecutan en la maquina. Cada instrucción tiene su correspondiente operación en la maquina.

Las instrucciones se introducen tras la directiva “.org”, o lo que es lo mismo, cuando se declara un bloque de programa. Más adelante veremos con más detalle la sintaxis dentro de este bloque, aquí solo haremos una pequeña introducción.

Básicamente, el ensamblador realiza una traducción entre un nemotécnico con unos operadores a un número. Cada nemotécnico representa una operación y se denominan así porque este permite acordarnos fácilmente de lo que realiza la operación correspondiente:



En este caso el ensamblador convierte de forma automática la expresión: “mov M, A” a 78h.

Realmente, el ensamblador permite complicar todo esto mucho más facilitando el trabajo al programador. Como ya dijimos, esto lo veremos más adelante.

5.2.1. Sintaxis de las Definiciones

El objetivo de una definición es, simplemente, poner un nombre a un número. Durante el ensamblaje, cada vez que nos encontramos con un operando que contiene el nombre este será sustituido de forma automática por el valor correspondiente con el que se ha definido.

Más adelante, si es necesario alterar este número por alguna razón, únicamente se tendrá que cambiar la declaración, lo que nos evita la tarea de buscar todas las apariciones del número en el texto, comprobar si se trata realmente del número que queremos alterar y modificarlo.

Las definiciones solo se pueden realizar si previamente se ha declarado la directiva de definiciones “.define”. Si no es así se producirá un error, ya que el ensamblador no entenderá la secuencia de léxicos.

La estructura de un bloque de definiciones es la siguiente:

```
. define
    [definición]
    [definición]
    [definición]
    ....
```



¿Existe alguna restricción sobre la tabulación y uso de espacios en la directiva y resto de definiciones?

No, las tabulaciones mostradas anteriormente solo son indicativas. No tienen porque realizarse de forma estricta. La palabra ‘define’ puede estar junto al punto sin separarse por espacios y las definiciones al principio de línea.

Cada definición tiene la siguiente estructura sintáctica:

Nombre	Valor	[Comentario]
--------	-------	--------------

Figura 5.4. Estructura de una Definición.

donde “nombre” declara el nombre, mientras que en “valor” se asigna el valor correspondiente. *Nombre* y *Valor* deben estar separados por al menos un espacio en blanco. Por ejemplo:

```
. define
    maximo FFFFh
    minimo 0
    umbral 0CA1h
    euro 166
```

Los identificadores maximo, minimo, umbral y euro son nombres, mientras que FFFFh, 0, 0CA1h y 166 son valores que se asignan respectivamente a los nombres.

Los identificadores de nombre no pueden ser números, es decir, deben comenzar al menos por una letra. Como máximo, deben tener un tamaño no superior a 255 caracteres. Igualmente, un nombre ya usado en una definición no puede volver a emplearse nuevamente en ninguna otra definición. En este caso el ensamblador muestra un mensaje de advertencia y asigna el nombre al valor correspondiente de la primera definición.

Aunque es un caso poco probable, puesto que si seguimos la organización usual por bloques no es posible, un nombre de definición no puede ser un nombre ya usado en una etiqueta declarada con anterioridad. En el apartado “*Sintaxis de las Instrucciones*” cuando se explican las etiquetas se hace referencia a esta limitación. También puede encontrar un ejemplo de este error en el capítulo “**Mensajes producidos por el Ensamblador**”.

Cada definición es una instrucción de ensamblaje. Como vemos no hay una instrucción maquina análoga a ella, de forma que no queda expresada de forma explícita. Sin embargo, la instrucción es tomada por el ensamblador que reemplaza cada nombre por el valor correspondiente.



¿Es una definición sensible a la capitalización?

A diferencia del resto de elementos del ensamblador, el identificador de una definición es sensible a minúsculas y mayúsculas. El objetivo de esto es obtener una mayor diversidad de nombres.

5.2.2. Sintaxis de los Datos

Dentro de un programa se establece una clara diferencia entre datos e instrucciones. Los datos constituyen básicamente los operandos de las instrucciones.

La distribución en memoria de datos e instrucciones suele estar separada y claramente diferenciada. Esto se debe a que tanto datos como instrucciones son tratados de la misma forma por la maquina según la situación en la que este, es decir, un dato no es un dato, ni una instrucción una instrucción, por sí sola. Depende de como sea tratada por la maquina.

Los datos pueden ser de tres tipos. Se han distinguido tres tipos distintos de datos simples con los que trabajar en el ensamblador. La diferencia fundamental entre estos tipos es el tamaño, como muestra la Tabla 5.1.

Tabla 5.1. Tamaño de los datos.

Tipo	Tamaño	Ejemplos
Byte	8 bits	1, 10, 255, 127, -127, ...
Word	16 bits	FE00h, 65535, FFFFh, ...
String	8 bits x nº caracteres	“Esto es un ejemplo”, “Salida del programa”, ...

Generalmente la mayoría de los datos serán de tipo *Byte*, puesto que el 8085 solo opera con datos de 8 bits. Los datos de tipo *Word* se emplearan con mayor probabilidad para definir direcciones de memoria, ya que el 8085 es capaz de direccionar direcciones de hasta 16 bits. Las cadenas de caracteres solo proporcionan una mayor legibilidad al programa ensamblador puesto que se pueden definir alternativamente mediante una secuencia de *Bytes*:

“Esto es un ejemplo”

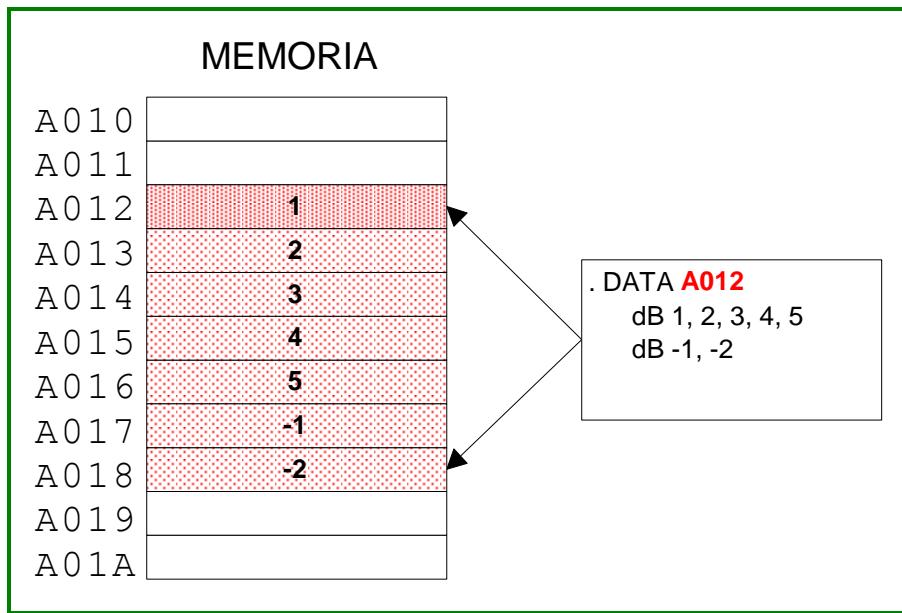
45h 73h 74h 6Fh 20h 65h 73h 20h 75h 6Eh 20h 65h 6Ah 65h 6Dh 70h 6Ch 6Fh

En la primera línea se muestra la cadena original, mientras que la segunda la cadena de bytes correspondiente. Como vemos, es mucho más sencillo de construir y modificar la notación para cadenas de caracteres.

La estructura de un bloque de datos es la siguiente:

```
. data <dirección origen>
    [declaración datos]
    [declaración datos]
    [declaración datos]
    ....
```

En dirección origen se indica la dirección de memoria en la que se comenzara a introducir los datos.



Cada declaración de datos tiene la siguiente sintaxis:

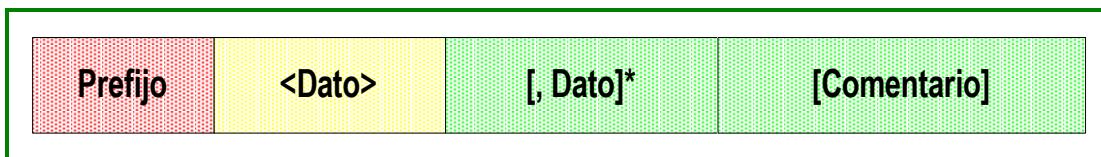


Figura 5.5. Estructura de una Declaración.

donde:

- El *prefijo* indica el tipo de dato que vamos a introducir. Los posibles prefijos coinciden con los tipos descritos anteriormente:
 - *dB*: declaración de Bytes o números de 8 bits.
 - *dW*: declaración de Words o números de 16 bits.
 - *dS*: declaración de cadenas de caracteres.

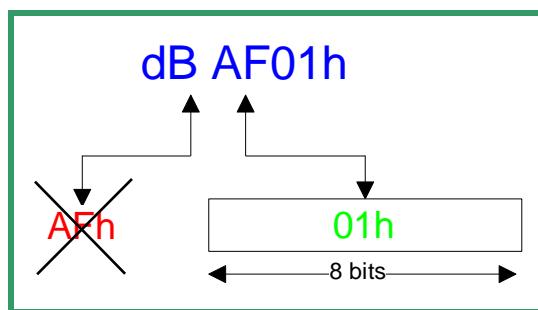
Los prefijos no son sensibles a la capitalización de letras. Por ejemplo, ‘*db*’, ‘*dB*’, ‘*DB*’ y ‘*Db*’ se reconocen igualmente por un prefijo de declaración de Bytes.

- Los *datos* simplemente son una secuencia de números o expresiones lógicas y aritméticas separados por comas o una cadena de caracteres.

Cuando se realiza una declaración de un dato (ya sea explícitamente o como resultado de una expresión aritmética y/o lógica) como un byte, este se almacena en el próximo byte útil de memoria.

Cuando la declaración es de un word **los 8 bits menos significativos de la expresión se almacenan en el próximo byte útil de memoria, mientras que los ocho bits más significativos se almacenan en el siguiente**. Como se ve, la dirección está colocada en la memoria en orden inverso. Normalmente, las direcciones se encuentran en la memoria en esta forma. Esta operación se utiliza básicamente para crear una tabla de direcciones constantes.

Si la representación del dato excede el número de bits con que ha sido declarado entonces se informa al programador mediante una advertencia de ensamblaje. **Únicamente quedaran almacenados los bits menos significativos.**



A continuación mostramos un ejemplo de declaración de datos:

```
.DATA 100h
    dB      1, 2 , 3, 4, 5, 6, 7, 8, 9, 0      ;cifras
    dB      32, 25, FFh, 32, -5, -120
    dW      0, FFFFh, 01A0h                      ;mis posiciones de memoria
    dS      "Pulse una tecla"                   ;mensaje para continuar
    dS      "Fin del programa"                  ;mensaje de salida
```

Además, es posible emplear etiquetas durante una definición, esto nos permite poder referenciar más adelante en el programa un determinado dato. El uso de etiquetas durante una definición pone las bases de lo que, en lenguajes de más alto nivel, serán las variables.

Finalmente la estructura de una declaración de datos es:

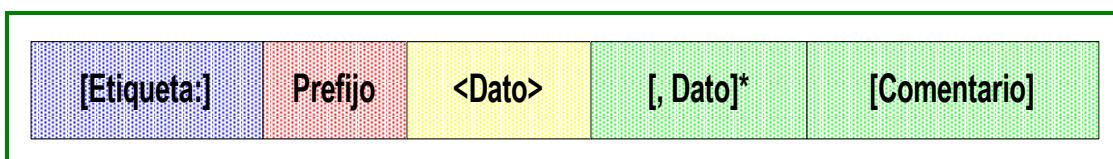


Figura 5.6. Estructura Extendida de una Declaración.

En el apartado 5.2.3 veremos con más detenimiento las etiquetas y como se emplean estas en las instrucciones. De esta forma veremos la utilidad real de la definición de etiquetas durante una declaración de datos.

5.2.3. Sintaxis de las Instrucciones

La estructura de un bloque de instrucciones es la siguiente:

```
. org <dirección origen>
    [instrucción]
    [instrucción]
    [instrucción]
    ....
```

Las instrucciones del lenguaje ensamblador vienen dadas por una serie de reglas que conforman la sintaxis de este. Dentro de cada instrucción hay cuatro partes o campos separados:

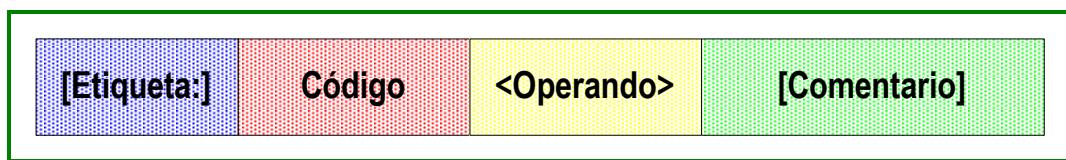


Figura 5.7. Estructura de una Instrucción.

Los campos Etiqueta y Comentario son prescindibles y no tienen porqué estar necesariamente en cada instrucción ensamblador. El Código de Operación, también conocido por *Codop*, es completamente necesario si queremos definir una instrucción. Los operadores dependerán del Código de Operación correspondiente, ya que hay instrucciones que no requieren ningún operador mientras que otras necesitan varios de ellos.

Estos campos están separados por espacios en blanco, no existe ninguna restricción sobre el número de estos que debe haber entre cada dos campos, siempre que al menos halla uno.

ETIQUETA

Es un campo de utilización opcional que, cuando está presente, puede tener una longitud de 1 a 255 caracteres. El primer carácter de la etiqueta debe ser una letra del alfabeto. De no ser así, como veremos más adelante podría ser confundido con una dirección real de memoria. Estos caracteres deben ir seguidos de dos puntos (:). Los códigos de instrucción están especialmente definidos por el ensamblador y no pueden ser utilizados como etiquetas. Nuestro ensamblador es capaz de diferenciar entre ellos, por lo que esta restricción no está presente, sin embargo se recomienda no mezclar etiquetas de salto con códigos de instrucción.

El objetivo de una etiqueta es referenciar una dirección de memoria. Esto es, cuando una etiqueta se sitúa en la declaración de un dato o delante una instrucción de programa esta haciendo referencia a la dirección de memoria en la que se encuentra dicho dato o instrucción.

Gracias a las etiquetas el programador de ensamblador no tiene que calcular a mano la dirección de un determinado dato que necesita ni la de una instrucción a la que se debe saltar. De esta manera se puede simplificar enormemente la tarea del programador con respecto a los saltos y a la carga o almacenamiento de datos.

Si tenemos en cuenta el significado de una etiqueta es muy fácil reconocer las limitaciones que el uso de estas conlleva. Por un lado, una etiqueta define solamente una dirección. No puede repetirse. Por tanto, esto no es posible:

```
Salto:    mov a, b
          ...
Salto:    call sub
          ...
          jmp Salto
```

Es obvio que el ensamblador no puede determinar qué dirección es a la que debe ir la instrucción JMP.

No obstante, la situación contraria es posible (aunque poco útil), esto es, asignar la misma dirección a dos etiquetas. La siguiente secuencia de instrucciones es válida:

```
Salto:
          ...
Salto2:   mov a, b
          ....
          jmp Salto
```

Ahora podemos justificar porque una etiqueta debe al menos comenzar por una letra. De no ser así, la etiqueta podría ser una secuencia completa de dígitos. Si esto ocurre el ensamblador es incapaz de distinguir una etiqueta de un número igual. Esta ambigüedad daría lugar a fallos.

Tabla 5.2. Ejemplo de etiquetas.

Dirección	Instrucción
0100	Mov a, b
0101	0100: Mov c, d
0102	Jmp 0100

Como podemos ver existe una clara ambigüedad sobre lo que realmente se quiere codificar. Por un lado se podría interpretar como que el salto de la dirección 0102 es hacia donde indica la etiqueta 0100 (dirección 0101). Pero por otro lado también es posible decir que realmente se quiere saltar a la dirección 0100.

Tampoco es posible que una etiqueta tenga el mismo nombre que una definición previa. Como en el caso anterior, el ensamblador no es capaz de distinguir a qué elemento, si etiqueta o definición, esta haciendo referencia el nombre encontrado.

Por ultimo, al igual que los nombres de definición, las etiquetas son sensibles a mayúsculas y minúsculas.



Resumiendo :

- Las etiquetas son sensibles a mayúsculas y minúsculas.
- Tienen que empezar al menos por una letra.
- Tienen que ser menores de 255 caracteres.
- No pueden coincidir con definiciones anteriores.
- No deben coincidir con etiquetas anteriores.

CODIGO DE OPERACIÓN

Es el campo más importante de la instrucción ya que en él se define la operación a realizar por la máquina (suma, resta, salto, etc.). Cada una de las instrucciones tiene un código determinado, que es privativo de la misma, y que debe aparecer en el campo del código. Por ejemplo, las letras "JMP" definen exclusivamente la operación de "salto incondicional" y ninguna otra instrucción. No existen dos instrucciones con el mismo código ni dos códigos para una misma instrucción.

Después de las letras del campo del código, debe haber como mínimo un espacio en blanco:

Jmp Salto	Salto incondicional a Salto (etiqueta)
Adi 7Bh	Sumar 7Bh con acumulador
Mov a, b	Transferir registro B al acumulador

A cada secuencia de letras que definen exclusivamente una operación se denomina nemotécnico. Los nemotécnicos a diferencia que las etiquetas y los identificadores de definiciones no son sensibles a las mayúsculas ni minúsculas. Esto es “Jmp”, “JMP”, “jmp” o “JmP” se refieren al mismo nemotécnico.

OPERANDO

La información contenida en este campo se usa conjuntamente con el campo del código a fin de definir con precisión la operación a ejecutar por la instrucción. Según el contenido del campo del código, el campo del operando puede no existir, o consistir en una cifra o palabra, o bien en dos, separados ambas por una coma.

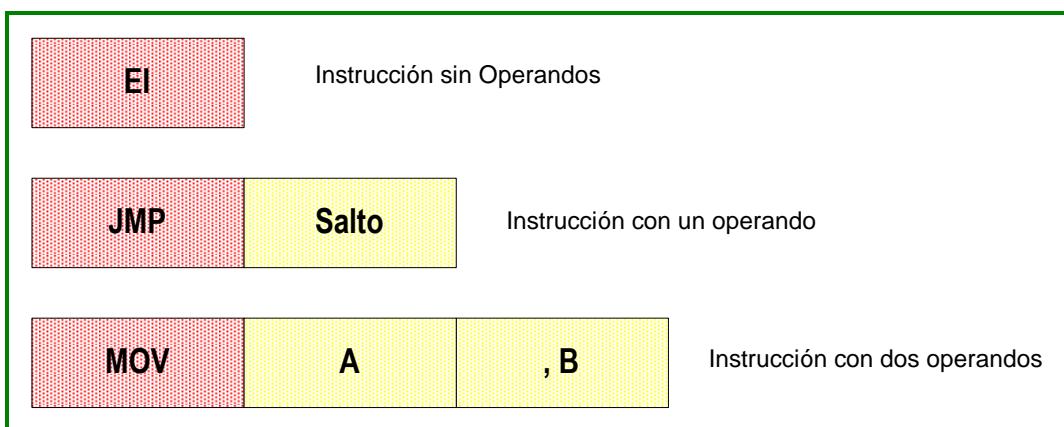


Figura 5.8. Tipos de Instrucciones.

Hay cuatro tipos de información válidos como campo del operando:

- **Registro.** Un registro (o código indicativo de una referencia a memoria) definido como fuente o destino de datos en una operación. Un número puede usarse para especificar el registro o referencia a memoria mencionados, pudiéndose obtener dicho número a partir de una expresión, pero el número finalmente evaluado debe estar entre 0 y 7. La correspondencia se muestra en la tabla 5.3.

Tabla 5.3. Números de registros.

Valor	Registro
0	B
1	C
2	D
3	E
4	H
5	L
6	Referencia a memoria
7	Acumulador (A)

Por ejemplo, la instrucción MVI permite la carga inmediata del segundo operador en el registro indicado por el primer operador. Esto es, usos típicos de esta instrucción serían:

MVI A, 1	Transferir 1 al acumulador
MVI H, dir_alta	Transferir dir_alta (definición) al registro H
MVI L, dir_baja	Transferir dir_baja (definición) al registro L

Cada uno de los registros se pueden sustituir de diversas formas:

MVI 7, 1	7 es el acumulador
MVI 8/2, dir_alta	8/2 es 4 que se corresponde con H
MVI regL, dir_baja	RegL es una definición con valor 5 (L)

- **Par de registros.** Un par de registros utilizado como fuente o destino de una operación con datos. Los pares de registros se especifican como se ve en la tabla 5.4.

Tabla 5.4. Pares de registros.

Especificación	Par de Registros
B	Registros B y C
D	Registros D y E
H	Registros H y L
PSW	Un byte que indica el estado de los bits de condición y el registro A.

SP	El registro de 16 bits del puntero de stack.
----	---

Por ejemplo:

INX B	Incrementar BC (como un registro de 16 bits)
PUSH PSW	Guardar en pila los bits de estado y el registro A
LXI D, 0h	Cargar HL con 0h

- **Dato inmediato.** Un dato expresado de forma inmediata en la instrucción. El dato puede ser resultado de una expresión. Por ejemplo:

ADI 5+5	Sumar 10 al contenido del acumulador
ORI 0Fh	OR del acumulador con 0Fh
LXI D, 10h+ (MOV A, B)	Cargar HL con 10h+78h

- **Dirección de 16 bits.** Una dirección de memoria de 16 bits o la etiqueta de una instrucción en memoria. Por ejemplo:

JMP 34F0h	Saltar a la dirección 34F0
LXI H, A000h	Cargar HL con A000h

COMENTARIOS

La única regla que rige la utilización de los comentarios es que éstos deben ir precedidos de un punto y coma (;).

AQUI: MVI C, DADH ; esto es un comentario.
--

El campo del comentario puede aparecer sólo en una línea, a pesar de que no exista instrucción en la misma:

;comentario sin instrucción

5.2.4. Sintaxis del Operando

Existen nueve formas de especificar el operando. Estas nuevas formas se detallan a continuación:

- Dato Hexadecimal.** Todo número hexadecimal esta expresado en base 16 para cuya representación emplea los dígitos del 0 al 9 y las 6 primeras letras del alfabeto (A, B, C, D, E y F). Son los más comunes cuando se trabaja en ensamblador. Los datos hexadecimales van seguidos de una letra H. Por ejemplo:

A2h (162), 10AFh (4271), FFFFh (65535), 11Ah (282)

- Dato Decimal.** Expresados en base 10, son los números que todos conocemos con dígitos entre 0 y 9. Pueden ir seguidos de la letra D, o bien ir solos. Por ejemplo:

0, 1, 255d, 35, 1050d

- Dato Octal.** Expresados en base 8, utilizan solo los dígitos del 0 al 7. Deben ir seguidos de la letra O ó Q para que sean reconocidos como tales. Por ejemplo:

777q (511), 123o (83), 242q (162), 22o (18)

- Dato Binario.** Números binarios expresados en base 2 (0,1). Van seguidos de la letra B. Se emplean generalmente para definir máscaras de bits. Por ejemplo:

0101011101B (349), 11110000B (240)

- El contenido actual del contador de programa.** Este se define con el carácter \$ y equivale a la dirección de la instrucción en ejecución.

- Una constante ASCII.** Son uno o más caracteres ASCII encerrados entre comillas simples.

'a' (65), 'b' (66), '*' (42), '?' (63)

7. Una instrucción encerrada entre paréntesis. Una instrucción entre paréntesis puede emplearse como operando. El valor concreto del operando será el código hexadecimal con el que la instrucción es codificada.

(MOV A, B) (65), (INX H) (35), (PUSH PSW) (245)

8. Identificadores de Etiquetas. Ya sea que se les ha asignado un valor numérico por el propio ensamblador, como vimos en el caso de los registros representados por números. O bien etiquetas definidas por el programador que aparecen en el campo de la etiqueta de otra instrucción o declaración de dato.

9. Expresiones Lógicas y Aritméticas. Todos los operandos descritos anteriormente son expresiones. Las expresiones lógicas y aritméticas son expresiones unidas mediante los operadores: + (suma), - (resta o cambio de signo), * (multiplicación), / (división), MOD (módulo), los operadores lógicos NOT, AND, OR, XOR, SHR (rotación hacia la derecha), SHL (rotación hacia la izquierda), y paréntesis a derecha e izquierda.

Todos los operadores tratan sus argumentos como cantidades de 16 bits y genera como resultados cantidades de 16 bits. Cada operador realiza la siguiente operación:

- El operador + genera la suma aritmética de sus operandos.

$$\boxed{\text{A01Bh}} + \boxed{00FFh} = \boxed{A11Ah}$$

- El operador - genera la resta aritmética de sus operandos, cuando se usa como sustracción (operador binario), o como aritmética negativo cuando se utiliza como cambio de signo (operador unario).

$$\boxed{\text{A01Bh}} - \boxed{00FFh} = \boxed{9F1Ch}$$

- El operador * indica el producto aritmético de los dos operandos.

$$\boxed{\text{A01Bh}} * \boxed{00FFh} = \boxed{9F7AE5h}$$

- El operador **/** calcula el cociente entero entre los dos operandos, el resto de la división se descarta.

$$\boxed{\text{A01Bh}} / \boxed{00FFh} = \boxed{\text{A0h}}$$

- El operador **MOD** calcula el resto de la división entre los dos operandos descartando el cociente.

$$\boxed{\text{A01Bh}} \text{ MOD } \boxed{00FFh} = \boxed{\text{BBh}}$$

- El operador **NOT** realiza el complemento de cada bit del operando.

$$\text{NOT } \boxed{0110110b} = \boxed{1001001b}$$

- El operador **AND** (o **&**) realiza la operación lógica AND bit a bit entre los operandos.

$$\boxed{011101b} \text{ AND } \boxed{111000b} = \boxed{011000b}$$

- El operador **OR** (o **|**) realiza la operación lógica OR bit a bit entre los operandos.

$$\boxed{011101b} \text{ OR } \boxed{111000b} = \boxed{111101b}$$

- El operador **XOR** (o **^**) realiza la operación lógica O-EXCLUSIVO bit a bit entre los operandos.

$$\boxed{011101b} \text{ XOR } \boxed{111000b} = \boxed{100101b}$$

- Los operadores **SHR** y **SHL** realizan un desplazamiento del primer operando a derecha e izquierda respectivamente el número de posiciones definidas por el segundo operando, introduciendo ceros en las nuevas posiciones.

$$011101b \text{ SHR } 3 = 000011b$$

$$011101b \text{ SHL } 3 = 101000b$$

El programador debe asegurarse de que el resultado generado por una de estas operaciones cumple los requisitos necesarios. En caso de que así no fuera los bits más significativos que no pudieran almacenarse se perderían. Por ejemplo:

MVI A, NOT 0	NOT 0 es FFFFh, MVI espera un dato inmediato de 8 bits.
MVI A, -1	-1 es FFFFh en complemento a dos.
MVI A, -1& (FFh)	Forma correcta de especificación.

Múltiples operadores pueden estar presentes en una expresión. Esta claro que el orden en que estos se apliquen va a dar lugar a diferentes resultados. Por esta razón las expresiones producidas por los operadores deben ser evaluadas en el orden de prioridad mostrado en la tabla 5.5.

Tabla 5.5. Operadores de una expresión.

Prioridad	
1	Expresiones entre paréntesis
2	Multiplicación (*), División (/), MOD, SHL, SHR
3	Suma (+), Resta (-)
4	NOT
5	AND
6	OR, XOR

En el caso de las expresiones entre paréntesis, el contenido entre los mismos debe evaluarse primero.

5.3. Especificación Formal de la Sintaxis del Ensamblador

La estructura del lenguaje ensamblador viene dada por una serie de reglas que conforman la sintaxis de este. El lenguaje ensamblador es el primer escalón dentro de los niveles de abstracción de la maquina, es decir, está muy cerca de la propia maquina, por lo que su complejidad como lenguaje es significativamente menor que cualquier otro lenguaje de más alto nivel.

La sintaxis de un lenguaje de programación se especifica mediante la gramática independiente del contexto que lo genera. Una gramática independiente del contexto es una cuádrupla (**N, T P, S**) en la que:

- a) **N** es el conjunto de símbolos no terminales del lenguaje,
- b) **T** es el conjunto de símbolos terminales del lenguaje,
- c) **P** es el conjunto de reglas de producción, y
- d) **S** es el axioma o símbolo inicial de la gramática,

y además cumple que todas las reglas de producción adoptan la forma:

$$A \rightarrow \alpha, \text{ donde } A \in N, \alpha \in (N \cup T)^*$$

En otras palabras, es una gramática de tipo 2.

Notación empleada

Para especificar formalmente la sintaxis de los lenguajes de programación se suelen usar BNF. BNF es un metalenguaje que se usa para especificar la sintaxis de los lenguajes de programación. Esto se consigue especificando en BNF la gramática que genera el correspondiente lenguaje.



BNF, Backus Naur Form, establecido por John Backus y Peter Naur (Junio 1.959, 2 de enero 1.960). La equivalencia entre las gramáticas independientes del contexto y BNF, en el sentido de tener la misma potencia expresiva, fue demostrada por S. Ginsburg, y H.G. Pice en 1.962

En BNF la gramática de un lenguaje de programación se expresa por medio de sus reglas de producción, por lo que la sintaxis con la que se escriban estas reglas deberá poner claramente de manifiesto cuales son:

- el símbolo inicial,
- los símbolos no terminales y
- los símbolos terminales de la gramática del lenguaje

Para ello el símbolo inicial de la gramática deberá aparecer en la parte izquierda de la primera regla de producción Y el resto de símbolos no terminales deberán aparecer en la parte izquierda de al menos una regla de producción. Se dice que esa regla define al símbolo no terminal.

A partir del concepto de gramática independiente del contexto, es obvio que ningún símbolo terminal podrá aparecer en la parte izquierda de ninguna regla de producción.

El orden natural que se suele seguir al escribir las reglas de producción es, dada una regla, escribir a continuación de ella las reglas correspondientes a los símbolos no terminales que aparezcan en su parte derecha. Si alguno o algunos, de estos símbolos son comunes a varias reglas la regla que define este símbolo se pondrá al final de todas ellas. Habitualmente las reglas de producción en cuya parte derecha solo aparecen símbolos terminales figuran al final de la especificación.

Para especificar las reglas de producción, la notación BNF consta de los siguientes metasímbolos:

- < > que se usan como delimitadores de los símbolos no terminales de la gramática al escribir las reglas de producción.
- ::= que se utiliza para separar las partes izquierda y, derecha de las reglas de producción, Y se lee “*se define como*”.
- | que se use como separador de las diversas alternativas que puedan aparecer en la parte derecha de una regla de producciones y se lee “*o*”.
- [] para representar que lo encerrado entre los corchetes es opcional.

- { } para representar que lo encerrado entre los corchetes se repite 0 o varias veces.

5.3.1. Notación BNF del Ensamblador

<Programa> ::= {<Bloque>}

<Bloque> ::= <Bloque definición> | <Bloque declaración> | <bloque Programa>

<Bloque definición> ::= <directiva define> {<definiciones>}

<Bloque declaración> ::= <directiva datos> {<declaraciones>}

<Bloque Programa> ::= <directiva programa> {<instrucciones>}

*<directiva define> ::= <punto> **DEFINE** [; <comentarios>]*

*<directiva datos> ::= <punto> **DATA** <expresión> [; <comentarios>]*

*<directiva programa> ::= <punto> **ORG** <expresión> [; <comentarios>]*

<definición> ::= <identificador> <expresión> [; <comentarios>]

<declaración> ::= [<identificador> :] <prefijo> <expresión> , {<expresión>} [; <comentarios>]

<instrucciones> ::= <código operación> [<expresión reg>] [, <expresión reg>] [; <comentarios>]

<expresión reg> ::= <expresión> | <registro>

<expresión> ::= <termino> { <operador débil> <termino> }

<término> ::= <elemento> { <operador fuerte> <elemento> }

<elemento> ::= <identificador> | <constante> | (<expresión>)

<identificador> ::= <letra> | <identificador> <letra> | <identificador> <dígito>

<constante> ::= <constante hex> | <constante decimal> | <constante octal> |
 <constante bin> | <constante carácter>

<constante hex> ::= <dígito hex> {<dígito hex>} H

<constante dec> ::= <dígito> {<dígito>} D

<constante octal> ::= <dígito oct> {<dígito oct>} O | <dígito oct> {<dígito oct>} Q

<constante bin> ::= <dígito bin> {<dígito bin>} B

<constante carácter> ::= ‘ <letra> | <dígito dec> ’

<código de operación> ::= ANA | AND | ACI | ADI | | XCHG | XTHL

<prefijo> ::= dB | dW | dS

<registro> ::= A | B | C | D | E | F | H | S | PSW

<operador débil> ::= + | - | AND | NOT | OR

<operador fuerte> ::= * | /

<dígito> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<dígito hex> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F

<dígito oct> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7

<dígito bin> ::= **0** | **1**

<letra> ::= **a** | **b** | **c** | ... | **z** | **A** | **B** | **C** | ... | **Z**

<punto> ::= .

5.4. Mensajes de Error producidos por el Ensamblador

5.4.1. Introducción

Los mensajes de error se generan cuando existe algún problema en el código fuente a ensamblar. Estos errores provocan que si el ensamblador siguiera convirtiendo el fichero en ensamblador, el programa resultante fuera totalmente incorrecto por lo que el proceso de ensamblaje se termina cada vez que se produce un error.

También es posible que el ensamblador genere mensajes de advertencia. Estos mensajes de advertencia indican incoherencias o ambigüedades en el código fuente. Estas ambigüedades no generan un programa totalmente incorrecto por lo que el proceso de compilación continua. Aunque funcione este conservara ciertos errores que deberían eliminarse. Más adelante veremos estos casos.

Los mensajes de error se muestran en rojo, mientras que los mensajes de advertencia se muestran en amarillo.

A continuación describiremos de forma detallada los mensajes de error y de advertencia de los que informa el ensamblador. Además mostraremos ejemplos en los que estos errores se producen.

5.4.2. Mensajes de Error

Errores en Directivas

Las Directivas ofrecen información al ensamblador sobre el tipo de elementos que se va a encontrar a continuación y la dirección de memoria donde debe disponerlos (si corresponde). Se caracterizan por ir precedidas por un punto.

Error: cada bloque debe comenzar por una directiva. Línea <*Nº línea*>

Cada bloque que se defina debe comenzar por la directiva correspondiente. Si no es así se lanza este error. Usualmente este error solo aparece cuando empezamos a introducir instrucciones en el programa o declaraciones de datos desde cero, sin definir ninguna directiva previa.

- *Nº de línea.* Línea en la que se encuentra el error.

Ejemplos del error:

```
; .org 100H           ;.data 10H
    mvi b, 5
salto : mov a,b          dB   0, 127, FFh
    add b             dS   "Error en directiva"
    jmp salto
```

	Error:	Identificador precediendo directiva no valido. Línea < <i>Nº línea</i> >
--	---------------	--

Antes del punto (.) que marca el comienzo de una directiva no puede haber nada salvo espacios en blanco o tabulaciones. En caso de que esto no se cumpla se genera el mensaje de error anterior.

- *Nº de línea.* Línea en la que se encuentra el error.

Ejemplo del error:

```
Directiva .org 100H
    mvi b, 5
salto : mov a,b
    add b
    jmp salto
```

	Error:	Directiva no reconocida. Línea < <i>Nº línea</i> >
--	---------------	--

Solo existen las directivas mencionadas anteriormente, esto es, DEFINE para las definiciones, DATA para las declaraciones de datos y ORG para comienzo del

programa. El ensamblador no es sensible a la capitalización de estas letras, siempre y cuando se correspondan con alguna directiva valida.

- *Nº de línea.* Línea en la que se encuentra el error.

Ejemplo del error:

```
.mi_directiva 100H
    mvi b, 5
salto : mov a,b
        add b
        jmp salto
```

Error:	Dirección incorrecta en la directiva. Línea < <i>Nº línea</i> >
--------	---

Las directivas DATA y ORG necesitan una dirección de memoria inicial a partir de la cual colocar los datos e instrucciones respectivamente. Tanto si no se introduce esta dirección así como si es inválida se genera este error.

- *Nº de línea.* Línea en la que se encuentra el error.

Ejemplos del error:

<pre>.org 1A mvi b, 5 salto : mov a,b add b jmp salto</pre>	<pre>.data dB 0, 127, FFh dS "No hay dirección origen"</pre>
---	--

En el primer caso la expresión es errónea, ya que para definir un valor hexadecimal es necesario introducir el sufijo ‘h’. El número no es reconocido correctamente dando lugar al error. Este caso se puede extender a cualquier otro en el que la expresión que decide la dirección de memoria es incorrecta.

En el segundo caso no se ha puesto ninguna dirección. Esto da también lugar a un error de ensamblado que genera el mensaje actual.

Errores en Definiciones

El objetivo de una definición es, simplemente, poner un nombre a un número. Durante el ensamblaje, cada vez que nos encontramos con un operando que contiene el nombre este será sustituido de forma automática por el valor correspondiente con el que se ha definido.

	Error:	Valor ' <i><Valor></i> ' asociado a definición ' <i><Definición></i> ' incorrecto. Línea <i><Nº línea></i>
--	--------	--

El valor que se ha asociado a un identificador de una definición no es correcto. En una definición sólo se puede asignar un valor (o cualquier expresión que se resuelva en un valor) a un identificador. Este mensaje de error indica que el valor que se ha introducido no es valido.

- *Valor*. Secuencia de símbolos que no se pueden reconocer como un valor.
- *Definición*. Nombre del idenficator al que se intenta asignar un valor incorrecto o no reconocible.
- *Nº de línea*. Línea en la que se encuentra el error.

Ejemplos del error:

<pre>.define pvp 180A iva 16 .org 100H ...</pre>	<pre>.define sin_valor varios 16 17 .org 100H ...</pre>
--	---

En el primer ejemplo el valor asociado a “pvp” es incorrecto. Es un valor en hexadecimal al que le falta el sufijo ‘h’. Esto es extensible a toda expresión que contenga errores.

En el segundo ejemplo se muestran dos errores. En el primero no se ha especificado un valor al que asignar el identificado “sin_valor”, el segundo intenta asignar varios valores a la vez en la misma línea.

		Error: Colisión con la declaración previa de la etiqueta: '< <i>Nombre</i> >'. Línea < <i>Nº línea</i> >
--	--	---

Se pretende emplear un nombre identificador en una definición que ya ha sido previamente empleado para una etiqueta.

- *Nombre*. Identificador de la declaración que ya se ha empleado anteriormente.
- *Nº de línea*. Línea en la que se encuentra el error.

Ejemplo del error:

.org 100H

```
salto: mvi a,1
      add a
      jmp salto
```

.define

```
salto 1000h
```

		Error: El identificador '< <i>Nombre</i> >' no es valido para una definición. Línea < <i>Nº línea</i> >
--	--	--

El nombre que se le quiere asignar a la definición no es valido. Esto se debe a que, o bien contiene caracteres inválidos, o que comienza por un número.

Recordemos que el primer carácter de un nombre de definición debe ser una letra del alfabeto. De no ser así, como ya vimos antes podría ser confundido con un número empleado en el programa.

- *Nombre*. Nombre de la definición que no es válido.
- *Nº de línea*. Línea en la que se encuentra el error.

Ejemplos del error:

```
.define
```

```
1salto 1000h
```

```
.define
```

```
12345 1000h
```

Errores en Declaraciones de Datos

Dentro de un programa se establece una clara diferencia entre datos e instrucciones. Los datos constituyen básicamente los operandos de las instrucciones.

	Error:	Declaración no valida. Expresión incorrecta: '<Valor>'. Línea <Nº línea>
--	---------------	--

Existe un error en los valores introducidos en una declaración de datos. Ya sea de 8 (bytes) o 16 (words) bits, si el valor correspondiente no se puede calcular, debido a algún error tipográfico, se genera este error.

- *Valor*. Secuencia de símbolos que no se pueden reconocer como un valor.
- *Nº de línea*. Línea en la que se encuentra el error.

Ejemplos del error:

```
.define
```

```
1salto 1000h
```

```
.define
```

```
12345 1000h
```

	Error:	Declaración incorrecta. No se han abierto comillas. Línea <Nº línea>
--	---------------	--

En la declaración de una cadena no se han abierto comillas antes de introducir la secuencia de letras correspondiente. Las comillas son necesarias para delimitar la

cadena que se quiere almacenar en memoria. Por esta razón es necesario que las cadenas de caracteres se expresen delimitadas por comillas.

- *Nº de línea.* Línea en la que se encuentra el error.

Ejemplo del error:

```
.data 0H
    ds "No se han abierto comillas"
.org 100H

salto: mv a,1
      add a
      jmp salto
```

Error:	Declaración incorrecta. No se han cerrado comillas. Línea < <i>Nº línea</i> >
--------	---

Como antes, pero el error contrario. Tras abrir comillas es necesario indicar el final de la secuencia de caracteres, esto se realiza mediante cierre de comillas. Aunque parezca una formalidad, el cierre de comillas puede evitar errores del programa que se deben a simples errores tipográficos.

- *Nº de línea.* Línea en la que se encuentra el error.

Ejemplo del error:

```
.data 0H
    ds "No se han abierto comillas" ; Esto se metería
.org 100H

salto: mv a,1
      add a
      jmp salto
```

Como vemos, el problema de introducir cadenas de caracteres es que si estas no se delimitan correctamente se pueden introducir caracteres espurios.

		Error: Declaración incorrecta. Solo se puede introducir una cadena. Línea < <i>Nº línea</i> >
--	--	--

El prefijo de declaración de cadenas (dS) solo permite introducir una cadena de caracteres, a diferencia que dB y dW no es posible introducir secuencias consecutivas en la misma línea separadas por comas.

- *Nº de línea.* Línea en la que se encuentra el error.

Ejemplo del error:

```
.data 0H
    ds "Cadena valida", "Cadena no valida"
.org 100H

salto: mvi a,1
        add a
        jmp salto
```

		Error: Prefijo '<Prefijo>' desconocido. Línea < <i>Nº línea</i> >
--	--	--

El ensamblador solo reconoce tres prefijos distintos. Estos son, el prefijo de enteros de 8 bits o Bytes (*dB*), el prefijo de palabras de 16 bits o Words (*dW*) y el prefijo de cadenas de caracteres o Strings (*dS*).

El ensamblador no es sensible a la capitalización de los prefijos, pero cualquier otro prefijo es rechazado generando este mensaje de error.

- *Prefijo.* Secuencia de caracteres que el ensamblador no reconoce como prefijo.
- *Nº de línea.* Línea en la que se encuentra el error.

Ejemplos del error:

```
.data 0H
dB FFh
dW FFFFh
dS "Cadena valida"
dI 0a123          ;Prefijo no valido
iva 16            ;Prefijo no valido
mvi a,1           ;Prefijo no valido

.org 100H

salto: mvi a,1
       add a
       jmp salto
```

Errores en Instrucciones

Las instrucciones del lenguaje ensamblador vienen dadas por una serie de reglas que conforman la sintaxis de este.

	Error:	Mnemotécnico '<Mnemotécnico>' desconocido. Línea <Nº línea>
--	---------------	---

Los mnemotécnicos representan el código de cada instrucción. Cada una de las instrucciones tiene un código determinado, que es privativo de la misma, y que debe aparecer en el campo del código. Los códigos ensamblador están claramente definidos por este.

- *Mnemotécnico*. El mnemotécnico no se reconoce como una instrucción propia del procesador 8085.
- *Nº de línea*. Línea en la que se encuentra el error.

Ejemplo del error:

```
.data 0H
dB FFh
dW FFFFh
dS "Cadena valida"

.org 100H

salto: mvi a,1
       paddsb a,b
       add a
       jmp salto
```

		Error: Sintaxis incorrecta en la instrucción. Primer operando no valido. Línea <i><Nº línea></i>
--	--	--

El primer operando de la instrucción no es un operando válido. Esto se puede deber a diferentes causas.

- *Nº de línea.* Línea en la que se encuentra el error.

Ejemplos del error:

.org 100H	.org 100H
salto: mvi a, 5	salto: mvi a, 1
add a	push a
jmp salto+(3*)	jmp salto

En el primer ejemplo se produce este error debido a que la expresión introducida es errónea.

A la vez que se genera este error se muestra también el error de expresión correspondiente.

En el segundo caso se intenta emplear un registro no valido para la instrucción (push a), por esta razón también se muestra un error.

		Error: Sintaxis incorrecta en la instrucción. Segundo operando no valido. Línea <i><Nº línea></i>
--	--	---

El segundo operando de la instrucción no es un operando válido. Esto se puede deber a las diferentes causas explicadas anteriormente.

- *Nº de línea.* Línea en la que se encuentra el error.

Ejemplos del error:

<pre>.org 100H salto: mvi a, 5+3*(5+) add a jmp salto</pre>	<pre>.org 100H salto: mov a, 1 add a jmp salto</pre>
---	--

En el primer ejemplo se produce este error debido a que la expresión introducida es errónea.

A la vez que se genera este error se muestra también el error de expresión correspondiente.

En el segundo caso se intenta emplear la función *mov* con una carga inmediata, lo cual no es posible (para eso está *mvi*). Aquí también se genera el mensaje de error.

Error: Sintaxis incorrecta en la instrucción. Falta operando. Línea <*Nº línea*>

La información contenida en el campo operando se usa conjuntamente con el campo del código (mnemotécnico) a fin de definir con precisión la operación a ejecutar por la instrucción.

Según el contenido del campo del código, el campo del operando puede no existir, o consistir en una cifra o palabra, o bien en dos, separados ambas por una coma.

Si la instrucción necesita más operandos de los que hay especificados se producirá este error.

- *Nº de línea*. Línea en la que se encuentra el error.

Ejemplo del error:

```
.org 100H
```

```
salto: mov a
      add a
      jmp
```

MOV es una instrucción que necesita necesariamente dos operandos. Solo hemos introducido uno, por lo que se genera un error.

De igual forma *JMP* necesita un operando.

	Error:	Sintaxis incorrecta en la instrucción. Sobra operando. Línea < <i>Nº línea</i> >
--	---------------	--

Situación contraria a la anterior en la que no falta un operando, sino que al contrario, hay operandos de más.

- *Nº de línea*. Línea en la que se encuentra el error.

Ejemplo del error:

```
.org 100H
```

```
salto: mov a,b,c
      add a
      jmp salto, salto2
```

Cualquier instrucción en el 8085 no requiere más de dos operadores. *MOV* tiene tres, por lo que se genera un error.

De igual forma *JMP* solo necesita un operando.

	Error:	Sintaxis incorrecta en la instrucción. MOV no permite dos direcciones de memoria como operandos. Línea < <i>Nº línea</i> >
--	---------------	--

La instrucción de transferencia permite tres tipos de transferencias: transferencia entre registros (direcciónamiento registro), transferencia desde la memoria (direcciónamiento registro indirecto) y transferencia hacia la memoria (direcciónamiento registro indirecto). En ningún caso permite la transferencia a la vez desde y hacia la memoria, esto es, especificar ambos operandos como direcciones de memoria.

- *Nº de línea.* Línea en la que se encuentra el error.

Ejemplo del error:

```
.org 100H
```

```
salto: mov M,M
       add a
       jmp salto
```

Errores en Expresiones

Una expresión es una secuencia de números y operadores que pueden resolverse. En ensamblador, a diferencia que otros lenguajes de más alto nivel, las expresiones han de resolverse en tiempo de compilación y no de ejecución, ya que el ensamblador solo traduce instrucciones, no compila. Por esta razón, las expresiones en ensamblador no contienen variables (aunque sí definiciones) ni registros.

Error:	Expresión incorrecta. Falta operando en expresión entre paréntesis. Línea < <i>Nº línea</i> >
--------	---

En las expresiones es posible utilizar paréntesis. Si realizando una operación se nos olvida poner un operando y cerramos el paréntesis se generará este error.

- *Nº de línea.* Línea en la que se encuentra el error.

Ejemplos del error:

$$(5+)$$

$$16 * 75 + (-)$$

$$('A' - 15 *)$$

		Error: Expresión incorrecta. Operando tras operador no encontrado. Línea < <i>Nº línea</i> >
--	--	---

Tras un operador no se encuentra el operando sobre el que se aplica.

- *Nº de línea.* Línea en la que se encuentra el error.

Ejemplos del error:

$$5+$$

$$16 * 75 +$$

$$'A' - 15 *$$

		Error: Expresión incorrecta. Operador no reconocido. Línea < <i>Nº línea</i> >
--	--	---

Se ha empleado un operador distinto de los disponibles.

- *Nº de línea.* Línea en la que se encuentra el error.

Ejemplos del error:

$$\text{Sqrt}(5)$$

$$\text{Solve}(x+5=0)$$

$$5 \text{ Modulo} 2$$

		Error: Expresión incorrecta. Falta operador en expresión entre paréntesis. Línea < <i>Nº línea</i> >
--	--	---

Al definir una operación de mayor prioridad mediante los paréntesis no se ha especificado el operador correspondiente a dicha operación.

- *Nº de línea.* Línea en la que se encuentra el error.

Ejemplos del error:

$$\begin{array}{c} (5 \ 16) \\ (16 * 5 \ 75) \\ (('A' \ 15) \ (5+8)) \end{array}$$

	Error:	Expresión incorrecta. No se han cerrado todos los paréntesis. Línea < <i>Nº línea</i> >
--	---------------	---

Se han abierto más paréntesis de los que se han cerrado.

- *Nº de línea.* Línea en la que se encuentra el error.

Ejemplos del error:

$$\begin{array}{c} (16 * 75 \\ ('A' - 15) * (5 + 5) \\ (\text{NOT} ((5 + 4) * 6) \end{array}$$

	Error:	Expresión incorrecta. División por cero. Línea < <i>Nº línea</i> >
--	---------------	--

En la expresión se intenta llevar a cabo una división por cero.

- *Nº de línea.* Línea en la que se encuentra el error.

Ejemplos del error:

$$\begin{array}{c} 7 / 0 \\ (55 * 13) / \text{NOT} (\text{FFh}) \\ 15 / ((16 * 32 + 2) - (64 * 8) - 2) \end{array}$$

Errores en Etiquetas

El objetivo de una etiqueta es referenciar una dirección de memoria. Esto es, cuando una etiqueta se sitúa en la declaración de un dato o delante una instrucción de programa esta haciendo referencia a la dirección de memoria en la que se encuentra dicho dato o instrucción.

Error:	No se ha dado nombre a la etiqueta. Línea < <i>Nº línea</i> >
--------	---

Este error aparece cuando se han puesto los dos puntos (:) pero no se ha dado nombre a la supuesta etiqueta.

- *Nº de línea*. Línea en la que se encuentra el error.

Ejemplo del error:

```
.org 100H
      mvi b, 5
: mov a,b
  add b
  jmp salto
```

Como vemos, aunque existe el símbolo indicativo de etiqueta (:) esta realmente no existe.

Error:	Identificador de etiqueta no valido. Línea < <i>Nº línea</i> >
--------	--

Cuando al nombre de una etiqueta le anteceden caracteres diferentes de espacio o tabulador se genera este error. El nombre empleado para una etiqueta no puede contener espacios en blanco.

- *Nº de línea*. Línea en la que se encuentra el error.

Ejemplo del error:

```
.org 100H

        mvi b, 5
Etiqueta de salto: mov a,b
                      add b
                      jmp 123
```

	Error:	El identificador '<Nombre>' no es valido para la etiqueta. Línea < <i>Nº linea</i> >
--	---------------	--

El nombre que se le quiere asignar a la etiqueta no es valido. Esto se debe a que, o bien contiene caracteres inválidos, o que comienza por un número. Recordemos que el primer carácter de la etiqueta debe ser una letra del alfabeto. De no ser así, como ya vimos antes podría ser confundido con una dirección real de memoria.

- *Nombre*. Identificador asignado a la etiqueta que no es válido.
- *Nº de línea*. Línea en la que se encuentra el error.

Ejemplo del error:

```
.org 100H

        mvi b, 5
123 :   mov a,b
                      add b
                      jmp 123
```

	Error:	Colisión con la definición previa de: <Nombre>. Línea < <i>Nº linea</i> >
--	---------------	---

Se intenta establecer una etiqueta con un nombre identificativo que ya ha sido previamente empleado en una definición.

- *Nombre*. Identificador de la etiqueta que ya se ha empleado anteriormente.

- *Nº de línea.* Línea en la que se encuentra el error.

Ejemplo del error:

```
.define
    salto 100h
.data (25)

.org 100H

        mvi b, 5
salto : mov a,b
        add b
        jmp salto
```

Como vemos, existe una ambigüedad sobre hacia donde se debe saltar. El identificador *salto* puede referirse o bien a la etiqueta o a la definición.

5.4.3. Mensajes de Advertencia

Advertencia:	Declaración previa de la etiqueta: '<Nombre>'. Línea <Nº línea>
---------------------	---

El nombre con el que se quiere definir la etiqueta fue previamente empleado para definir otra etiqueta, por lo que no debería reutilizarse. El ensamblador tomara la ultima declaración de la etiqueta como la válida.

- *Nombre.* Identificador de la etiqueta que ya se ha empleado anteriormente.
- *Nº de línea.* Línea en la que se encuentra el error.

Ejemplo de la advertencia:

```
.org 100H

salto : mvi b, 5
salto : mov a,b
        add b
        jmp salto
```

Como vemos, existe una ambigüedad sobre hacia donde se debe saltar. Por defecto el ensamblador optara por la última vez en que se declara la etiqueta de salto.

	Advertencia: Declaración previa de la definición: '< <i>Nombre</i> >'. Línea < <i>Nº línea</i> >
--	---

El nombre al que se quiere aplicar la definición fue previamente empleado en otra definición, por lo que no debería reutilizarse. El ensamblador tomara la ultima definición como la válida.

- *Nombre*. Identificador de la definición que ya se ha empleado anteriormente.
- *Nº de línea*. Línea en la que se encuentra el error.

Ejemplo de la advertencia:

```
.define
    maximo 100
    minimo 0
    media 50
    maximo 120

.org 100H

salto : mvi b, 5
        add b
        jmp salto
```

El nombre *maximo* se utiliza dos veces para realizar una definición. De esta forma el valor de *maximo* puede ser 100 o 120. Por defecto será 120.

	Advertencia: La directiva define no necesita dirección origen. Línea < <i>Nº línea</i> >
--	---

Solo las directivas .DATA y .ORG requieren una dirección de origen a partir de la cual situar datos o instrucciones respectivamente.

Las definiciones solo se emplean en tiempo de compilación, realizándose las oportunas sustituciones. Si se especifica una dirección, esta es reconocida pero no se empleará en ningún momento.

- *Nº de línea.* Línea en la que se encuentra el error.

Ejemplo de la advertencia:

```
.define A0h
    maximo 100
    minimo 0
    media 50

.org 100H

salto : mvi b, 5
        add b
        jmp salto
```

El nombre *maximo* se utiliza dos veces para realizar una definición. De esta forma el valor de *maximo* puede ser 100 o 120. Por defecto será 120.

Advertencia:	La dirección '<dirección>' no se puede almacenar en 16 bits. Línea < <i>Nº línea</i> >
---------------------	---

El procesador 8085 solo puede trabajar con direcciones de 16 bits. Cualquier otra dirección mayor de 65535 (o FFFFh) no es direccionable.

Si una directiva .DATA o .ORG reciben una dirección origen mayor a FFFFh se generará esta advertencia. Solo los 16 bits menos significativos de la dirección se almacenarán.

- *Nº de línea.* Línea en la que se encuentra el error.

Ejemplo del error:

```
.org 1FFFFH

    mvi b, 5
salto : mov a,b
        add b
        jmp salto
```

	Advertencia: El valor '<Valor>' no se puede almacenar en un byte. Línea <Nº línea>
--	---

Al realizar una declaración de datos en forma de bytes estos deben de ser representables en 8 bits. Si no ocurre esto se lanza esta advertencia.

- *Nº de línea.* Línea en la que se encuentra el error.

Ejemplo del error:

```
.data 0H
    dB 256, 100n, 100000000b

.org 100H

    mvi b, 5
salto : mov a,b
        add b
        jmp salto
```

	Advertencia: El valor '<Valor>' no se puede almacenar en un word. Línea <Nº línea>
--	---

Como en el caso anterior, con las palabras de 16 bits. Si el valor declarado no se puede representar en 16 bits se lanza esta advertencia.

- *Nº de línea.* Línea en la que se encuentra el error.

Ejemplo del error:

```
.data 0H
    dw 65536, 10000h, 1000000000000000b

.org 100H

    mvi b, 5
salto : mov a,b
        add b
        jmp salto
```

Advertencias:	El primer operando (<i><Op></i>) no se puede representar en 8 bits. Línea < <i>Nº línea</i> >
	El segundo operando (<i><Op></i>) no se puede representar en 8 bits. Línea < <i>Nº línea</i> >

Ciertas instrucciones permite especificar un operando inmediato de 8 bits. Estas advertencias aparecen cuando el operando inmediato no puede representarse en 8 bits.

- *Nº de línea.* Línea en la que se encuentra el error.

Ejemplo del error:

```
.org 100H

    mvi b, 100h
salto : mov a,b
        add b
        jmp salto
```

Advertencia:	El primer operando (<i><Op></i>) no se puede representar en 16 bits. Línea < <i>Nº línea</i> >
---------------------	--

Ciertas instrucciones permite especificar un operando inmediato de 16 bits. Esta advertencia aparece cuando el operando inmediato no puede representarse en 16 bits.

- *Nº de línea.* Línea en la que se encuentra el error.

Ejemplo del error:

```
.org 100H  
    mvi b, FFh  
salto : mov a,b  
        add b  
        jmp 10000h
```

	Advertencia: El segundo operando (<i><Op></i>) no se puede representar en 16 bits. Línea <i><Nº línea></i>
--	--

Ciertas instrucciones permiten especificar un operando inmediato de 16 bits. Esta advertencia aparece cuando el operando inmediato no puede representarse en 16 bits.

- *Nº de línea.* Línea en la que se encuentra el error.

Ejemplo del error:

```
.org 100H  
    mvi b, FFh  
salto : mov a,b  
        add b  
        LXI H, 10000h
```

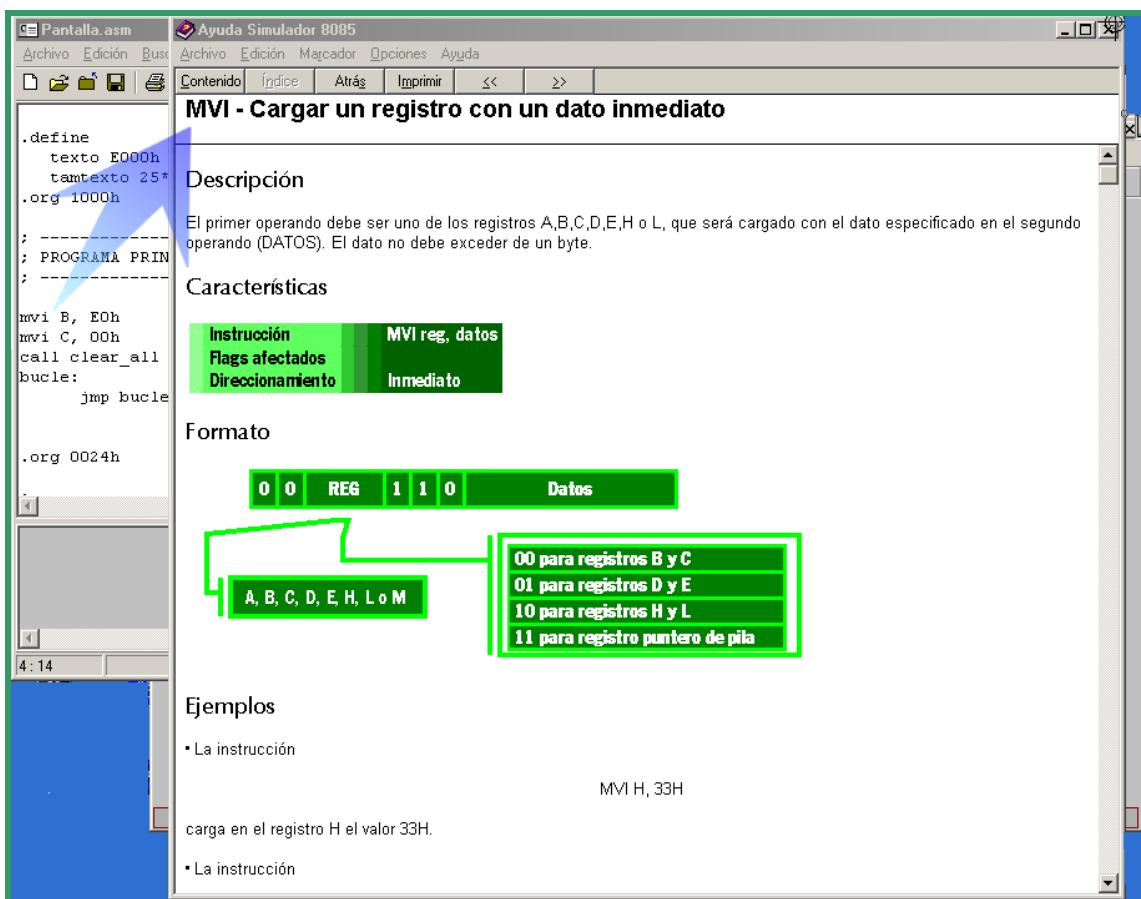
Apéndice A1

Conjunto de instrucciones del 8085

A1.1. Introducción

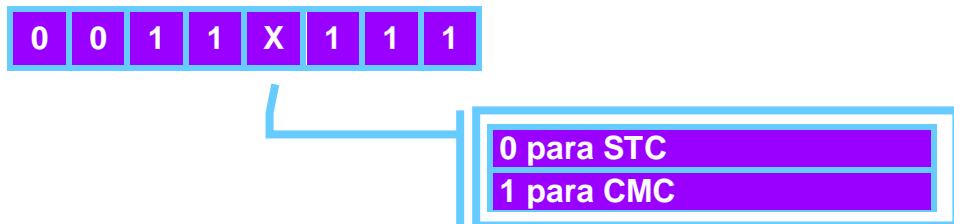
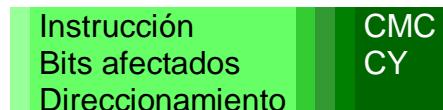
Aunque el conjunto de instrucciones puede encontrarse en cualquier libro que trate este tema específicamente, en este apartado se muestra un "diccionario" de las mismas. La descripción se lista funcionalmente para una localización más inmediata y todas están descritas de forma detallada.

Si desea hacer una búsqueda alfabética de las instrucciones puede usar la ayuda incorporada en el programa simulador.

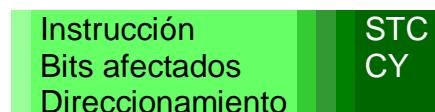


A1.2.  **Instrucciones del bit de acarreo**

A continuación se describen las instrucciones que operan directamente sobre el bit de acarreo. Estas instrucciones utilizan un byte en la forma siguiente:

**A1.2.1.**  **CMC Complementar acarreo****Descripción**

Si el bit de acarreo es 0, se pone a 1. Si es un 1, se pone a 0.

Formato**A1.2.2.**  **STC Activar acarreo****Descripción**

El bit de acarreo se pone a 1.

Formato

0	0	1	1	0	1	1	1
---	---	---	---	---	---	---	---

A1.3. Instrucciones de un registro

A continuación se describen las instrucciones que operan sobre un solo registro o posición de memoria. Si se especifica una referencia a memoria, la dirección de la misma viene dada por el contenido de los registros H y L, donde el registro H contiene los ocho bits más significativos de la dirección, y el registro L los restantes.

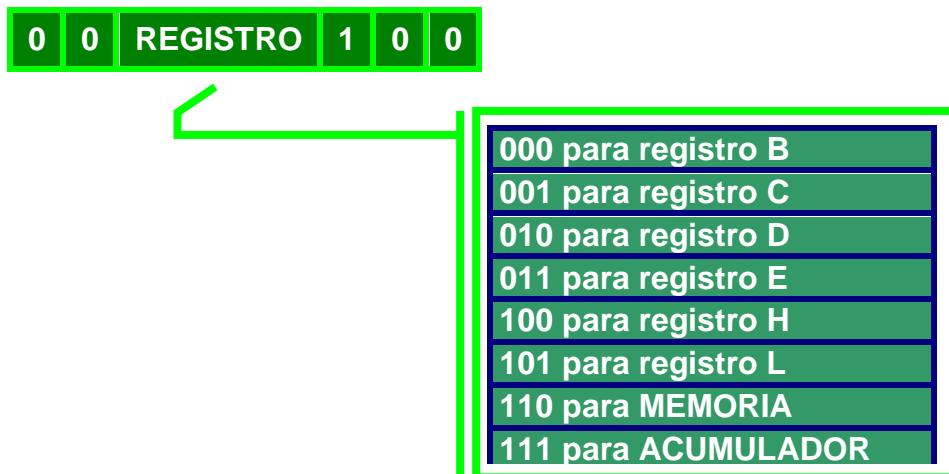
A1.3.1. INR Incrementar registro o memoria

Instrucción	INR reg
Bits afectados	Z, S, P, AC
Direccionamiento	Registro indirecto

Descripción

El contenido del registro o posición de memoria especificados se incrementa en una unidad.

Formato



Ejemplo

Si el registro A contiene 98H, la instrucción

INR A

hará que este registro contenga la cantidad 99H.

A1.3.2. DCR Decrementa registro o memoria

Instrucción	DCR
Bits afectados	Z, S, P, AC
Direccionamiento	Registro

Descripción

El contenido del registro o posición de memoria especificados se decrementa en una unidad.

Formato

Ejemplo

Si el registro A contiene 99H, la instrucción

DCR A

hará que este registro contenga la cantidad 98H.

A1.3.3. CMA Complementar acumulador

Instrucción	CMA
Bits afectados	
Direccionamiento	

Descripción

Cada uno de los bits del acumulador se complementa (operación denominada a uno).

Formato**A1.3.4. DAA Ajuste decimal del acumulador**

Instrucción	DAA
Bits afectados	Z, S, P, CY, AC
Direccionamiento	Registro

Descripción

El número hexadecimal de 8 bits contenido en el acumulador se ajusta como dos dígitos de 4 bits codificados en binario decimal, según el proceso siguiente:

- (1). Si los cuatro bits menos significativos del acumulador representan un número mayor que 9, o si el bit de acarreo auxiliar es igual a uno, el acumulador se incrementa en seis unidades. Si no se presentan tales condiciones, el contenido del acumulador no varía.

- (2). Si los cuatro bits más significativos del acumulador representan ahora un número mayor que 9, o si el bit de acarreo es uno, los cuatro bits más significativos se incrementan en seis unidades. Asimismo, si no tienen lugar las circunstancias expuestas, el contenido del acumulador no se incrementa.

Si hay acarreo de los cuatro bits menos significativos durante el paso (1), el bit de acarreo auxiliar se pone a 1; si no lo hay, se pone a 0. Por otra parte, si hay acarreo de los cuatro bits más significativos durante el paso (2), se activará el bit de acarreo, poniéndose a cero si no se produce acarreo.

Formato

0	0	1	0	0	1	1	1
---	---	---	---	---	---	---	---

Nota

Esta instrucción se utiliza en las operaciones de suma de números decimales. Es la única instrucción cuya operación depende del bit de acarreo auxiliar.

Ejemplo

Supongamos que queremos realizar una suma decimal de dos números (40 + 80). Ambos bits de acarreo están a cero.

La secuencia de instrucciones podría ser:

- (1).MVI B,80H
- (2).MVI A,40H ; Acumulador = 40H
- (3).ADD B ; Acumulador = 40H + 80H = C0H
- (4).DAA ; Acumulador = 20H
; Bit de acarreo = 1

La instrucción DAA opera de la siguiente forma:

1. Como el contenido de los bits [0 – 3] del acumulador es menor que 9 y el bit de acarreo auxiliar es cero, el contenido del acumulador no varía.
2. Como los 4 bits más significativos del acumulador representan un número mayor que 9, estos 4 bits se incrementan en 6 unidades, poniendo a uno el bit de

acarreo.

Acumulador	C0H	1 1 0 0 0 0 0 0	CY = 0
+ 6	60H	0 1 1 0 0 0 0 0	CY = 0
Nuevo acumulador			20H
0 0 1 0 0 0 0 0			CY = 1

A1.4.  **Instrucción NOP**

Instrucción	NOP
Bits afectados	
Direccionamiento	

Descripción

No se realiza ninguna operación.

Formato

0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

A1.5. Instrucciones de transferencia de datos

Esta serie de instrucciones transfieren datos entre los registros, la memoria y el acumulador. Ocupan un byte en la forma siguiente:

A1.5.1. MOV Movimiento

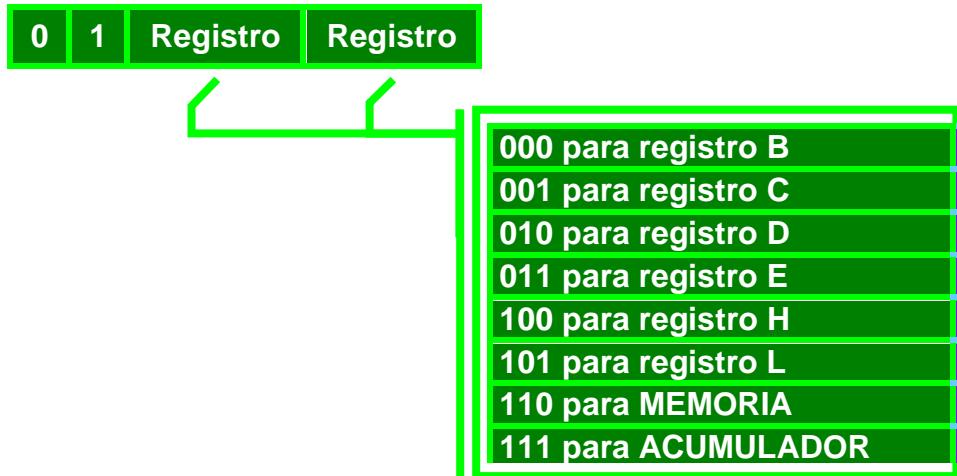
Instrucción	MOV reg, reg
Bits afectados	
Direccionamiento	Registro o registro indirecto

Descripción

Podemos distinguir 3 casos:

- (A). Transferencia entre registros (direccionamiento registro).
- (B). Transferencia desde la memoria (direccionamiento registro indirecto).
- (C). Transferencia a la memoria (direccionamiento registro indirecto).
 - (A). **MOV R₁, R₂**
El contenido del registro R₂ es transferido al registro R₁. R₁ y R₂ pueden ser los registros B, C, D, E, H, L o el acumulador A.
 - (B). **MOV R, M**
El contenido de la dirección de memoria, cuya dirección está en los registros H-L, es transferido al registro R. R puede ser cualquiera de los registros A, B, C, D, E, H o L.
 - (C). **MOV M, R**
El contenido del registro R es transferido a la dirección de memoria indicada por los registros H-L.

Formato



Ejemplos

- Supongamos que el registro B contiene 00H y el registro C contiene 30H. La instrucción

MOV B,C

almacenará 30H en el registro B.

- Supongamos que el registro H contiene 00H y el registro L contiene 30H. La instrucción

MOV M, A

almacenará el contenido del acumulador en la posición de memoria 0030H.

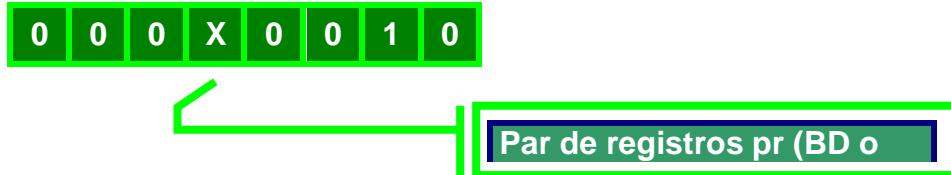
A1.5.2. STAX Almacenar el contenido del acumulador

Instrucción	STAX rp
Bits afectados	
Direccionamiento	Registro indirecto

Descripción

El contenido del acumulador se almacena en la posición de memoria especificada por los registros B y C, o los registros D y E.

Formato



Ejemplo

Si el registro B contiene 3FH y el registro C contiene 16H, la instrucción

STAX B

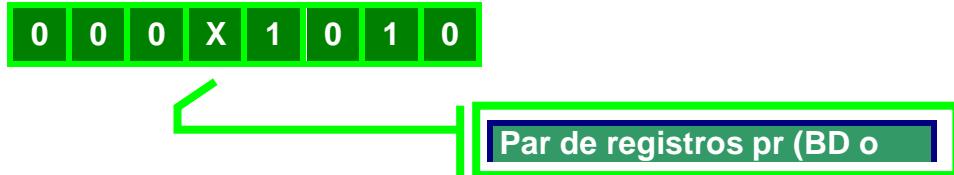
almacenará el contenido del acumulador en la posición de memoria 3F16H.

A1.5.3. LDAX Cargar el acumulador



Descripción

El contenido de la posición de memoria especificada por los registros B y C, o los registros D y E, reemplaza el contenido del acumulador.

Formato**Ejemplo**

Si el registro D contiene 3FH y el registro E contiene 16H, la instrucción

LDAX D

cargará en el acumulador el contenido de la posición de memoria 3F16H.

A1.6. Instrucciones de registro o memoria y acumulador

A continuación vamos a ver las instrucciones que operan con el contenido del acumulador y el de uno de los registros o posición de memoria. Estas instrucciones ocupan un byte en la forma siguiente:



La instrucción opera sobre el contenido del acumulador, con la cantidad definida por el registro especificado por REGISTRO. Si se especifica una referencia a memoria, la cantidad utilizada por la instrucción es la correspondiente a la posición de memoria determinada por los registros H y L, en los que el registro H guarda los 8 bits más significativos, y el registro L, los 8 restantes. Tanto el contenido del registro como de la posición de memoria no varían al finalizar la instrucción, guardándose el resultado en el acumulador.

A1.6.1. ADD Sumar registro o memoria al acumulador

Instrucción	ADD reg
Bits afectados	Z, S, P, CY, AC
Direccionamiento	Registro

Descripción

El contenido del registro o posición de memoria especificados se suma al contenido del

acumulador, usando aritmética de complemento a dos. El resultado se guarda en el acumulador.

Formato



Ejemplos

- Si el registro B contiene el valor 3AH y el acumulador contiene 6CH, la instrucción

ADD B

realiza la siguiente suma:

Registro B	3AH	0 0 1 1 1 0 1 0
Acumulador	6CH	0 1 1 0 1 1 0 0
Nuevo acumulador	A6H	1 0 1 0 0 1 1 0

- La instrucción

ADD A

duplica el contenido del acumulador.

A1.6.2. ADC Sumar registro o memoria al acumulador con acarreo

Instrucción Bits afectados Direccionamiento	ADC reg Z, S, P, CY, AC Registro indirecto
---	--

Descripción

El contenido del registro o posición de memoria especificados más el contenido del bit de acarreo, se suman al contenido del acumulador.

Formato



Ejemplo

Supongamos que el registro B contiene el valor 30H, el acumulador 76H, y el bit de acarreo está puesto a cero. La instrucción

ADC C

realizará la siguiente suma:

Registro B	30H	0 0 1 1 0 0 0 0
Acumulador	76H	0 1 1 1 0 1 1 0
Bit de acarreo		0
<hr/>		
Nuevo acumulador	A6H	1 0 1 0 0 1 1 0

El nuevo contenido del acumulador será A6H, mientras que todos los bits de condición quedarán puestos a cero excepto los de signo y paridad.

Si el bit de acarreo hubiera sido 1 antes de realizar la operación, hubiera tenido lugar la siguiente suma:

Registro B	30H	0 0 1 1 0 0 0 0
Acumulador	76H	0 1 1 1 0 1 1 0
Bit de acarreo		1
<hr/>		
Nuevo acumulador	A7H	1 0 1 0 0 1 1 1

El acumulador contendría ahora A7H y todos los bits de condición excepto el de signo, estarían puestos a cero.

A1.6.3. **SUB** Restar registro o memoria del acumulador

Instrucción	SUB reg
Bits afectados	Z, S, P, CY, AC
Direccionamiento	Registro

Descripción

El contenido del registro o posición de memoria especificados se resta al contenido del acumulador, usando aritmética de complemento a dos. El resultado se guarda en el acumulador.

Si no hay acarreo del bit de más peso, el bit de acarreo se pone a uno, y viceversa, al contrario de lo que ocurre con la operación de suma

Formato

1	0	0	1	0	REGISTRO
---	---	---	---	---	----------

Ejemplos

Antes de entrar en los ejemplos recordamos que restar utilizando aritmética de complemento a dos equivale a complementar cada bit del segundo operando y sumar 1.

1. Si el acumulador contiene 60H y el registro E contiene 28H, la instrucción

SUB E

realizará la siguiente operación de resta:

Acumulador	60H	0 1 1 0 0 0 0 0
+(-Registro B)	+(-28H)	1 1 0 1 0 1 1 1
Bit de acarreo		1
Nuevo acumulador	38H	0 0 1 1 1 0 0 0

Al final de la operación el acumulador contendrá 38H y el bit de acarreo se pondrá a cero debido a que ha habido acarreo del bit más significativo.

2. La instrucción

SUB A

restará al acumulador a sí mismo, obteniéndose un resultado de cero. Se utiliza en muchas ocasiones para poner a cero el bit de acarreo y el acumulador.

A1.6.4. SBB Restar registro o memoria del acumulador con acarreo

Instrucción	SBB reg
Bits afectados	Z, S, P, CY, AC
Direccionamiento	Registro indirecto

Descripción

El valor del bit de acarreo se suma internamente al contenido del registro o posición de memoria especificados. Este valor se resta del acumulador usando aritmética de complemento a dos.

Esta instrucción es de gran utilidad en la realización de restas de varios bytes, pues tiene en cuenta el valor positivo o negativo de la sustracción anterior.

Formato**Ejemplo**

Si el registro C contiene 08H, el acumulador almacena 05H y el bit de acarreo está activado, la instrucción

SBB C

efectúa la siguiente operación:

1. $08H + \text{bit de acarreo} = 09H$.
2. Complemento a dos de $09H = 11110111$ (F7H)
3. Lo anterior se suma al acumulador:

Acumulador	05H	0 0 0 0 0 1 0 1
	F7H	1 1 1 1 0 1 1 1
<hr/>		
Nuevo acumulador	FCH	1 1 1 1 1 1 1 0 0

4. No hay acarreo al final por lo que el bit de acarreo se queda a uno. Los bits de signo y paridad están puestos a uno mientras que el bit de cero está a cero.

A1.6.5. ANA**Función lógica AND entre registro o memoria con acumulador**

Instrucción	ANA reg
Bits afectados	Z, S, P, CY, AC
Direccionamiento	Registro indirecto

Descripción

Se realiza la función lógica AND bit a bit entre el contenido del registro o posición de memoria especificados y el contenido del acumulador. El bit de acarreo se pone a cero.

Formato

1	0	1	0	0	REGISTRO
---	---	---	---	---	----------

Nota

La tabla de verdad de la función lógica AND es:

A	B	R
0	0	0
0	1	0
1	0	0
1	1	1

Ejemplo

Si el registro B contiene 6CH y el acumulador almacena 3AH, la instrucción

ANA B

realiza la siguiente operación:

Acumulador	3AH	0 0 1 1 1 0 1 0
Registro B	6CH	0 1 1 0 1 1 0 0
<hr/>		
Nuevo acumulador	28H	0 0 1 0 1 0 0 0

A1.6.6. XRA Función lógica O-EXCLUSIVO entre registro o memoria con acumulador

Instrucción Bits afectados Direccionamiento	XRA reg Z, S, P, CY, AC Registro
---	--

Descripción

Se realiza la función lógica O-EXCLUSIVO bit a bit entre el contenido del registro o posición de memoria especificados y el contenido del acumulador, guardándose el resultado en este último.

Formato



Nota

La tabla de verdad de la función lógica O-EXCLUSIVO es:

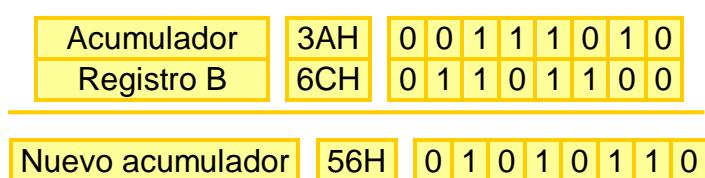
A	B	R
0	0	0
0	1	1
1	0	1
1	1	0

Ejemplos

- Si el registro B contiene 6CH y el acumulador almacena 3AH, la instrucción

XRA B

realiza la siguiente operación:



2. La función O-EXCLUSIVO de cualquier bit con uno da lugar al complemento del mismo. Así, si el acumulador contiene todo unos, la instrucción

XRA B

produce el complemento a uno del contenido del registro B, y lo guarda en el acumulador.

3. En algunas ocasiones, un byte se utiliza para reflejar los estados de ciertas condiciones dentro de un programa, donde cada uno de los ocho bits puede responder a una determinada condición de falso o verdadero, actuado o inhibido, etc.

Mediante la función O-EXCLUSIVO podemos determinar cuántos bits de la palabra han cambiado de estado en un determinado lapsus de tiempo.

A1.6.7. ORA Función lógica OR entre registro o memoria con acumulador

Instrucción	ORA reg
Bits afectados	Z, S, P, CY, AC
Direccionamiento	Registro indirecto

Descripción

Se realiza la función lógica AND bit a bit entre el contenido del registro o posición de memoria especificados y el contenido del acumulador, quedando en este último el resultado. El bit de acarreo se pone a cero.

Formato

1	0	1	1	0	REGISTRO
---	---	---	---	---	----------

Nota

La tabla de verdad de la función lógica OR es:

A	B	R
0	0	0
0	1	1
1	0	1
1	1	1

Ejemplo

Como sea que la función OR de cualquier bit con un uno da como resultado uno, y de cualquier bit con cero, lo deja invariable, esta función se utiliza frecuentemente para poner a uno grupos de bits.

Si el registro B contiene OFH y el acumulador almacena 33H, la instrucción

ORA B

realiza la siguiente operación:

Acumulador	33H	0 0 1 1 0 0 1 1
Registro B	0FH	0 0 0 0 1 1 1 1
Acumulador	3FH	0 0 1 1 1 1 1 1

Este ejemplo concreto garantiza que los cuatro bits de menos peso del acumulador son unos, mientras que los demás permanecen invariables.

A1.6.8. **CMP** Comparar registro o memoria con acumulador

Instrucción	CMP reg
Bits afectados	Z, S, P, CY, AC
Direccionamiento	Registro indirecto

Descripción

El contenido del registro o posición de memoria especificados se compara con el contenido del acumulador. Esta comparación se realiza restando internamente el contenido del registro al del acumulador, permaneciendo éste invariable, y colocando los bits de condición en función del resultado. Concretamente, el bit de cero se pone a uno si las cantidades comparadas son iguales, y se pone a cero si son desiguales. Como sea que se realiza una operación de resta, el bit de acarreo se pondrá a uno si no hay acarreo del bit 7, indicando que el contenido del registro o posición de memoria es mayor que el contenido del acumulador, y se pondrá a cero si es mayor que el acumulador.

Si las dos cantidades difieren en signo, el acarreo adopta el valor contrario a lo anteriormente expuesto.

Formato

1	0	1	1	1	REGISTRO
---	---	---	---	---	----------

Ejemplos

A continuación exponemos 3 ejemplos de esta operación.

1. Si el acumulador almacena 0AH y el registro B contiene 05H, cuando se realiza la instrucción

CMP B

Tiene lugar la siguiente resta interna:

Acumulador	0AH	0 0 0 0 1 0 1 0
+(- Registro B)	-5H	1 1 1 1 1 0 1 1
Acumulador	05H	0 0 0 0 0 1 0 1

Existe acarreo en el bit 7 por lo que el bit de acarreo se pone a cero. El acumulador sigue almacenando 0AH y el registro B, 05H. No obstante, el bit de acarreo se pone a cero, así como el bit de cero, indicando que el contenido del registro B es menor que el acumulador.

2. Ahora el acumulador tiene el valor 02H. Entonces:

Acumulador	02H	0 0 0 0 0 0 1 0
+(- Registro B)	-5H	1 1 1 1 1 0 1 1
Acumulador	FDH	1 1 1 1 1 1 0 1

En este el bit de acarreo se pone a uno (no existe acarreo del bit 7) y el bit de cero estará a cero, indicando que el contenido del registro B es mayor que el acumulador.

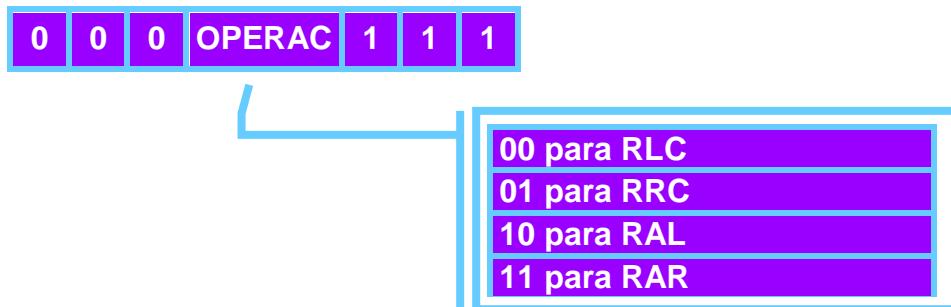
3. Por último supongamos un valor -1BH para el acumulador. En esta situación:

Acumulador	-1BH	1 1 1 0 0 1 0 1
+(- Registro B)	-5H	1 1 1 1 1 0 1 1
Acumulador	E0H	1 1 1 0 0 0 0 0

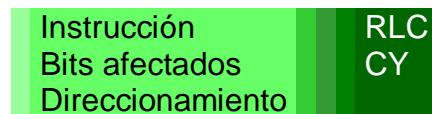
Aquí el bit de acarreo está a cero. Como los dos números difieren en signo, el hecho de que el bit de acarreo esté a cero indica que el contenido del registro B es mayor que el del acumulador, al contrario de cómo ocurría en el ejemplo anterior.

A1.7. Instrucciones de rotación del acumulador

A continuación se describen las instrucciones que provocan una rotación del contenido del acumulador. Esta operación únicamente puede realizarse con el acumulador, no con ningún registro o posición de memoria.



A1.7.1. RLC Desplazar el acumulador a la izquierda



Descripción

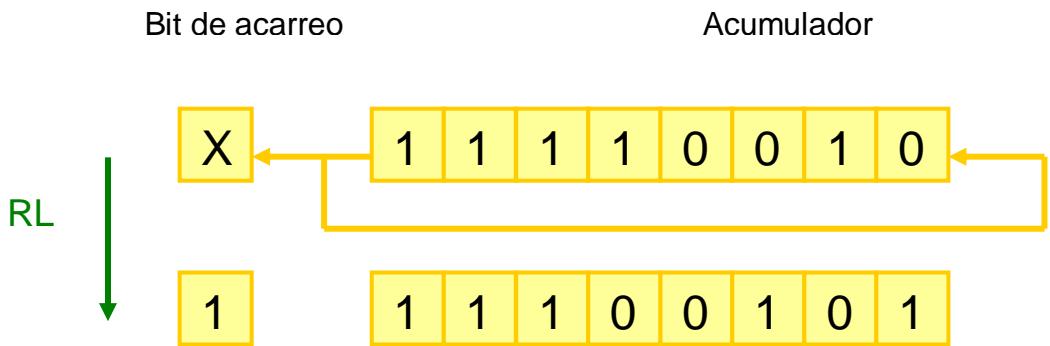
RLC rota un bit hacia la izquierda todo el contenido del acumulador, transfiriendo el bit de más alto orden al bit de acarreo y al mismo tiempo a la posición de menor orden del acumulador.

Formato



Ejemplo

Supongamos que el acumulador contiene 82H.



A1.7.2. RRC Desplazar el acumulador a la derecha

Instrucción	RRC
Bits afectados	CY
Direccionamiento	

Descripción

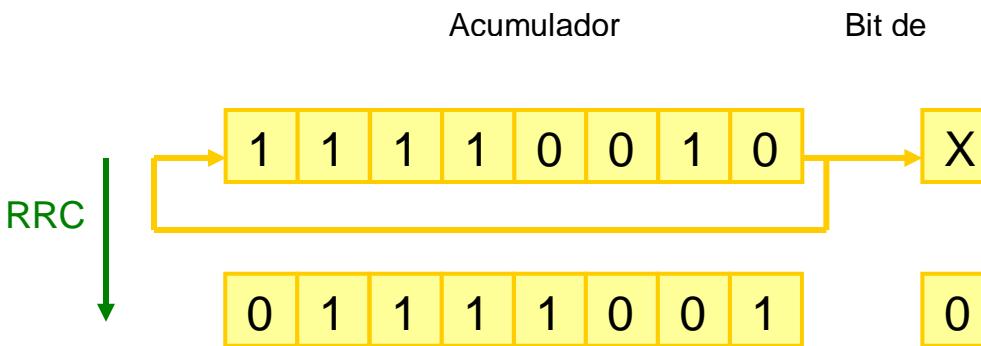
RRC rota el contenido del acumulador un bit a la derecha, transfiriendo el bit de más bajo orden a la posición de más alto orden del acumulador, además pone el bit de acarreo igual al bit de menor orden del acumulador.

Formato



Ejemplo

Supongamos que el acumulador contiene F2H. La instrucción RRC realizará la siguiente operación sobre el acumulador y el bit de acarreo:



A1.7.3. RAL Desplazar el acumulador hacia la izquierda a través del bit de acarreo

Instrucción	RAL
Bits afectados	CY
Direccionamiento	

Descripción

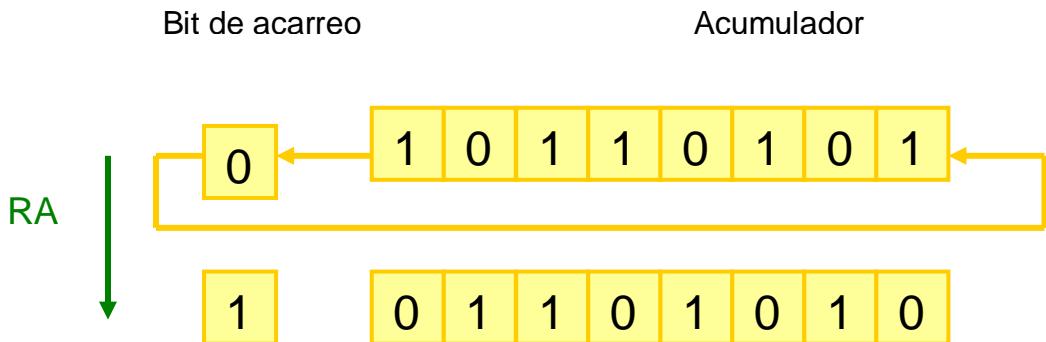
RAL hace girar el contenido del acumulador y el bit de acarreo un espacio de un bit hacia la salida (izquierda). El bit de acarreo que es tratado como si fuera del acumulador, se transfiere el bit de menor orden del acumulador. El bit de mayor orden del acumulador se transfiere al bit de acarreo. No tiene operandos.

Formato



Ejemplo

Supongamos que el acumulador contiene B5H. La instrucción RAL efectuará las siguientes modificaciones en el registro acumulador y en el bit de acarreo:



A1.7.4. RAR Desplazar el acumulador hacia la derecha a través del bit de acarreo

Instrucción	RAR
Bits afectados	CY
Direccionamiento	

Descripción

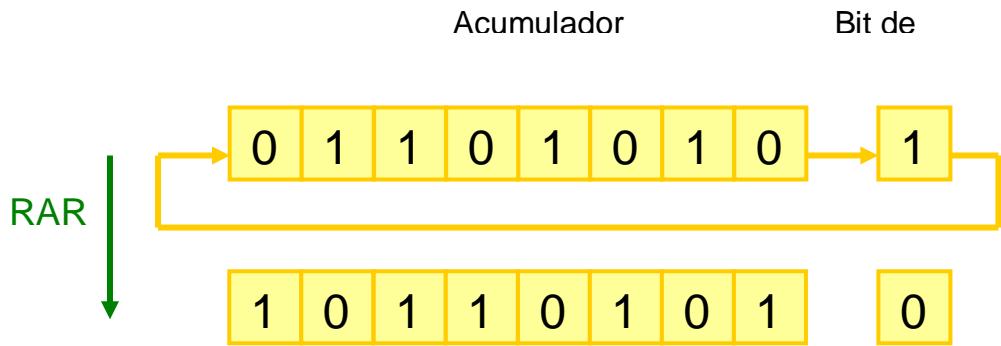
RAR rota el contenido del acumulador y del bit de acarreo 1 bit de posición a la derecha. El bit de acarreo que es tratado como si fuera parte del acumulador se transfiere al bit de más alto orden del acumulador. El bit de menor peso del acumulador se carga en el bit de acarreo. No existen operandos en la instrucción RAR.

Formato



Ejemplo

En este caso el acumulador contendrá el valor 6AH. La instrucción RAL efectuará las siguientes modificaciones en el registro acumulador y en el bit de acarreo:



A1.8. Instrucciones con pares de registros

A continuación se describen las instrucciones que dan lugar a operaciones con pares de registros.

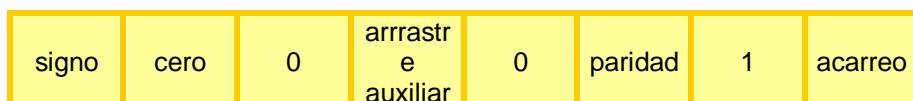
A1.8.1. PUSH Colocar datos en stack

Instrucción	PUSH pr
Bits afectados	
Direccionamiento	Registro indirecto

Descripción

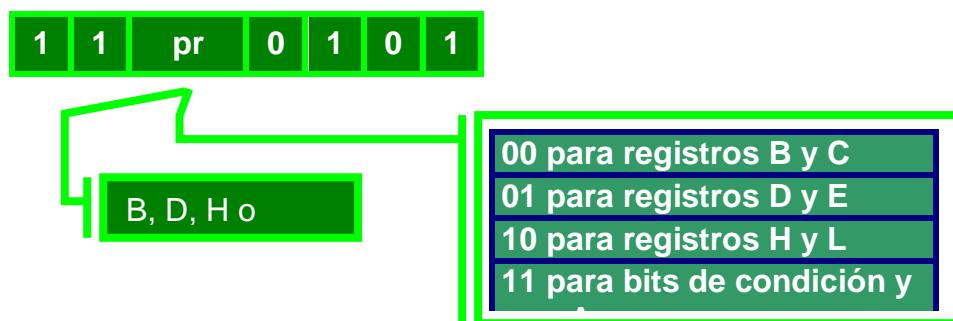
El contenido del par de registros especificado se guarda en dos bytes de memoria definidos por el puntero de stack.

El contenido del primer registro se guarda en la posición de memoria inmediatamente inferior a la del puntero de stack. El contenido del segundo registro del par se guarda en la posición de memoria dos unidades inferior al puntero de stack. Si se hace referencia al par de registros PSW, en el primer byte de información se guarda el estado de los cinco bits de condición. El formato de este byte es el siguiente:



Sea cual sea el par de registros especificado, una vez que los datos se han guardado, el puntero de pila se decrementa en dos unidades.

Formato



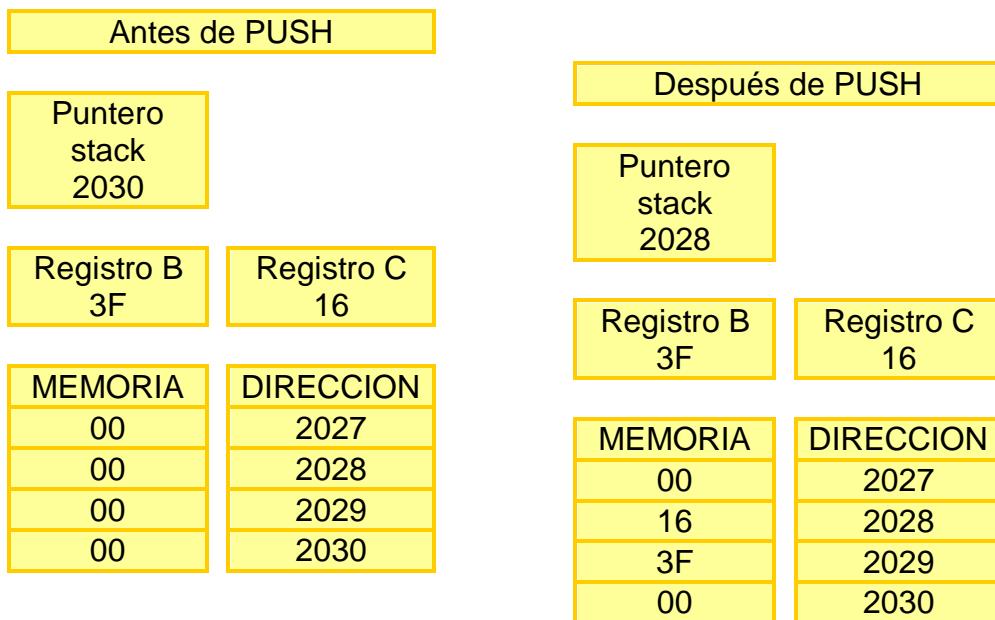
Ejemplos

- Supongamos que el registro B contiene 3FH, el registro C contiene 16H y el puntero de pila vale 2030H. La instrucción

PUSH B

almacenará el contenido del registro B en posición de memoria 2029H, el contenido del registro C en la dirección de memoria 2028H, y decrementa dos unidades el puntero de stack, dejándolo en 2028H.

Gráficamente podemos ver el proceso anterior:



- Supongamos ahora que el acumulador contiene 33H, el puntero de pila tiene 102AH, y los bits de condición de cero, acarreo y paridad están a uno, mientras que los de signo y acarreo auxiliar están a cero. La instrucción

PUSH PSW

Almacena el contenido del acumulador en la posición de memoria 1028H, y coloca el valor 47H, correspondiente a los citados estados de los bits de

condición, en la posición 1029H, mientras que en el puntero de pila queda el valor 1028H.

A1.8.2. **POP** Sacar datos del stack

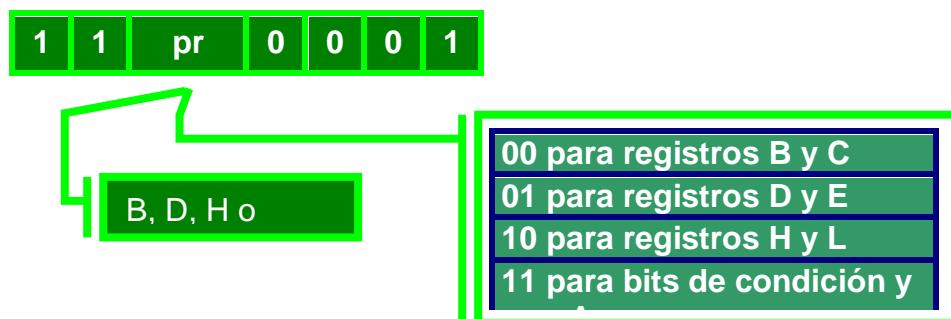
Instrucción	POP pr
Bits afectados	
Direccionamiento	Registro indirecto

Descripción

POP PR copia el contenido de la posición de memoria direccionada por el stack pointer en el registro de bajo orden del par de registros especificados. A continuación se incrementa el stack pointer en 1 y copia el contenido de la dirección resultante en el registro de más alto orden del par. Luego se incrementa el stack pointer otra vez de modo que se apunta al siguiente dato del stack. El operando debe especificar el par de registros BC, DE, HL o PSW.

POP PSW usa el contenido de la localización de memoria especificada por el stack pointer para restablecer los bits de condiciones.

Formato



Ejemplos

- Supongamos que las posiciones de memoria 2028H y 2029H contienen respectivamente 16H y 3FH, mientras que el puntero de pila contiene 2028H. La instrucción

POP B

Carga el registro C con el valor de 16H de la posición de memoria 2028H, carga el registro B con el valor 3FH de la posición 2029H, e incrementa dos unidades el puntero de stack, dejándolo en 2030H. Gráficamente podemos ver este proceso:

Antes de POP		Después de POP	
Puntero stack 2028		Puntero stack 2030	
Registro B 00	Registro C 00	Registro B 3F	Registro C 16
MEMORIA 00	DIRECCION 2027	MEMORIA 00	DIRECCION 2027
16	2028	16	2028
3F	2029	3F	2029
00	2030	00	2030

2. Si las posiciones de memoria 1008H y 1009H poseen respectivamente 00H y 16H, y el puntero de pila vale 1008H, la instrucción

POP PSW

carga 00H en el acumulador y pone los bits de estado de la siguiente forma:

16h =	S	Z		AC		P		CY
	0	0	0	1	0	1	1	0

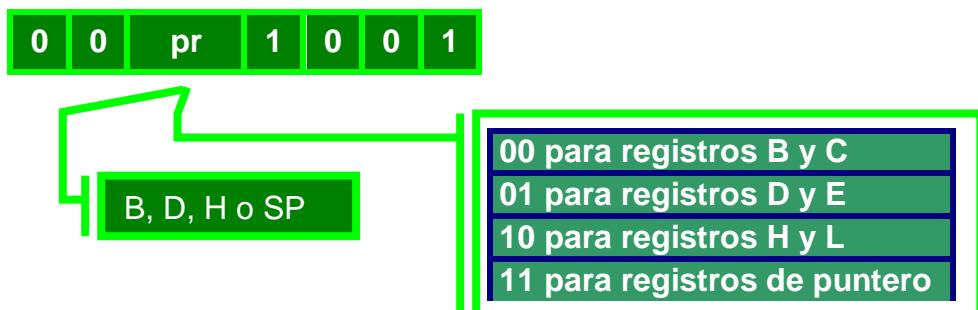
A1.8.3. DAD Suma doble

Instrucción	DAD pr
Bits afectados	CY
Direccionamiento	Registro

Descripción

DAD RP suma el valor de un dato de 16 bits contenido en un determinado par de registros (PR) al contenido del par de registros HL. El resultado es almacenado en el par HL. Los operandos (PR) pueden ser B = BC, D = DE, H = HL, SP. Téngase en cuenta que la letra H debe ser empleada para especificar que el par de registros HL debe ser doblado. DAD pone el bit de acarreo a 1 si hay una salida de acarreo de los registros HL. Y además no afecta a ningún otro bit.

Formato



Ejemplos

- Supuesto que los registros D, E, H y L contienen 33H, 9FH, A1H y 7BH respectivamente, la instrucción

DAD D

Realiza la siguiente suma:

$$\begin{array}{r}
 \begin{array}{|c|c|} \hline
 B - C & 339F \\ \hline
 H - L & A17B \\ \hline
 \end{array}
 \\
 \hline
 \begin{array}{|c|c|} \hline
 H - L & 051A \\ \hline
 \end{array}
 \end{array}$$

- Al ejecutar la instrucción

DAD H

se duplica el valor del número de 16 bits contenido en H – L (equivale a desplazar los 16 bits una posición hacia la izquierda).

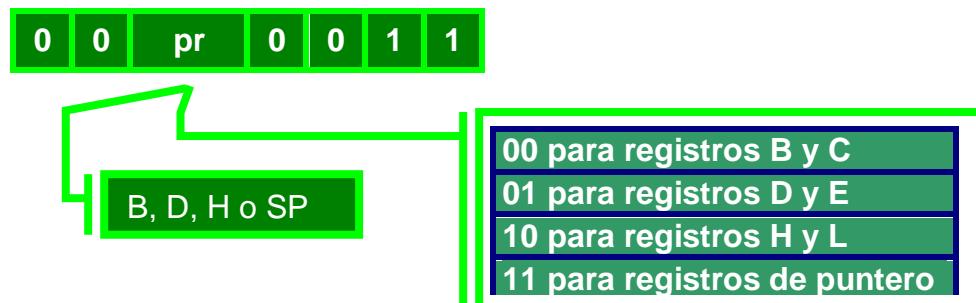
A1.8.4. **INX** Incrementar par de registros

Instrucción	INX pr
Bits afectados	
Direccionamiento	Registro

Descripción

El número de 16 bits contenido en el par de registros especificado se incrementa en una unidad.

Formato



Ejemplos

- Suponiendo que los registros H y L contienen respectivamente 30H y 00H, la instrucción

INX H

hace que el registro H contenga 30H y el registro L el valor 01H.

2. Si el puntero de pila contiene FFFFH, la instrucción

INX SP

hace que éste contenga 0000H.

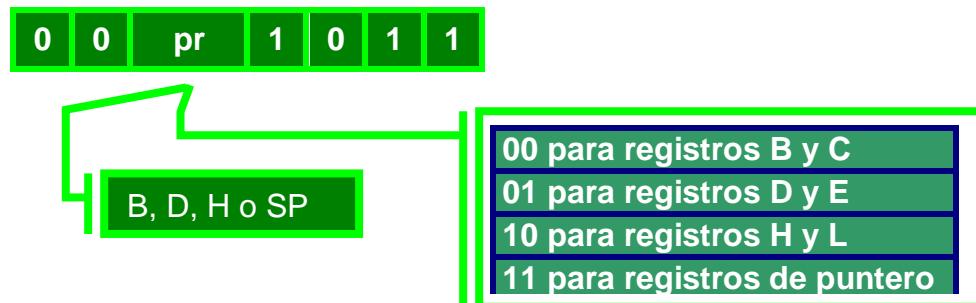
A1.8.5. DCX Decrementar par de registros

Instrucción	DCR pr
Bits afectados	
Direccionamiento	Registro

Descripción

El número de 16 bits contenido en el par de registros especificado se decrementa en una unidad.

Formato



Ejemplo

Suponiendo que los registros H y L contienen respectivamente 30H y 00H, la instrucción

DCX H

hace que el registro H contenga 2FH y el registro L el valor FFH.

A1.8.6. XCHG Intercambiar datos entre registros

Instrucción	XCHG
Bits afectados	
Direccionamiento	Registro

Descripción

XCHG cambia el contenido de los registros H y L con el contenido de los registros D y E.

Formato

1	1	1	0	1	0	1	1
---	---	---	---	---	---	---	---

Ejemplo

Si los registros H, L, D y E contienen respectivamente 01H, 02H, 03H y 04H, la instrucción XCHG realiza el siguiente intercambio:

Antes de ejecutar XCHG				Después de ejecutar XCHG			
D	E	H	L	D	E	H	L
03	04	01	02	01	02	03	04

A1.8.7. XTHL Intercambiar datos con el stack

Instrucción	XTHL
Bits afectados	
Direccionamiento	Registro indirecto

Descripción

XTHL cambia los dos bytes de la posición más alta del stack con los dos bytes almacenados en los registros H y L. Así XTHL salva el contenido actual del par HL y

carga nuevos valores en HL.

XTHL cambia el contenido del L con la posición de memoria especificada por el stack pointer y el registro H es intercambiado con el contenido del SP+1.

Formato

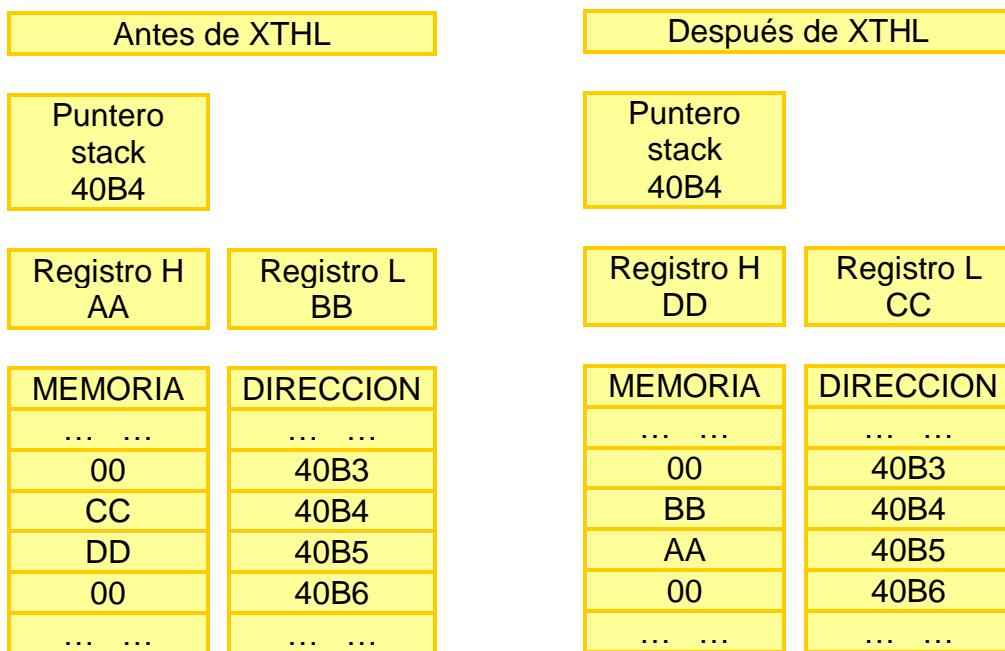
1	1	1	0	0	0	1	1
---	---	---	---	---	---	---	---

Ejemplo

Si el puntero de pila contiene 40B4H, los registros H y L contienen AAH y BBH respectivamente, y las posiciones de memoria 40B4H y 40B5H contienen CCH y DDH respectivamente, la instrucción

XTHL

realizará la siguiente operación:



A1.8.8. SPHL **Cargar el puntero de stack desde los registros H y L**

Instrucción	SPHL
Bits afectados	
Direccionamiento	

Descripción

Los 16 bits contenidos en los registros H y L reemplazan el contenido del puntero de stack. El contenido de los registros H y L permanece invariable.

Formato

1	1	1	1	1	0	0	1
---	---	---	---	---	---	---	---

Ejemplo

Si los registros H y L contienen respectivamente 50H y 6CH, la instrucción

SPHL

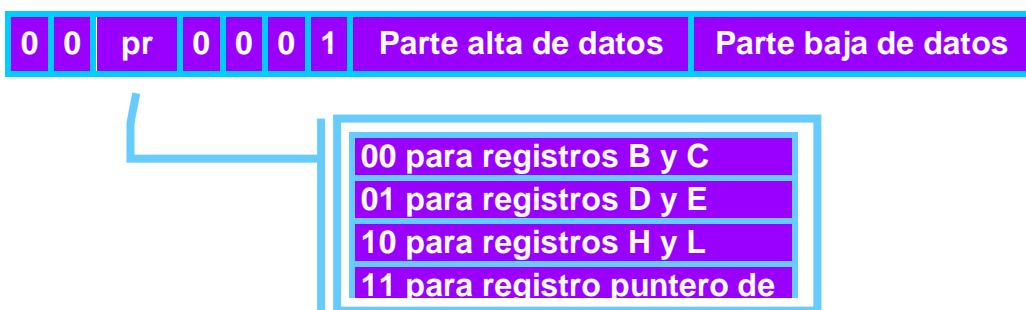
carga el puntero de stack con el valor 506CH.

A1.9. Instrucciones con datos inmediatos

A continuación se describen las instrucciones que realizan operaciones utilizando bytes de datos que forman parte de la propia instrucción.

Estas instrucciones forman un grupo amplio en el que no todas tienen la misma longitud y formato. Las instrucciones ocupan dos o tres bytes del siguiente modo:

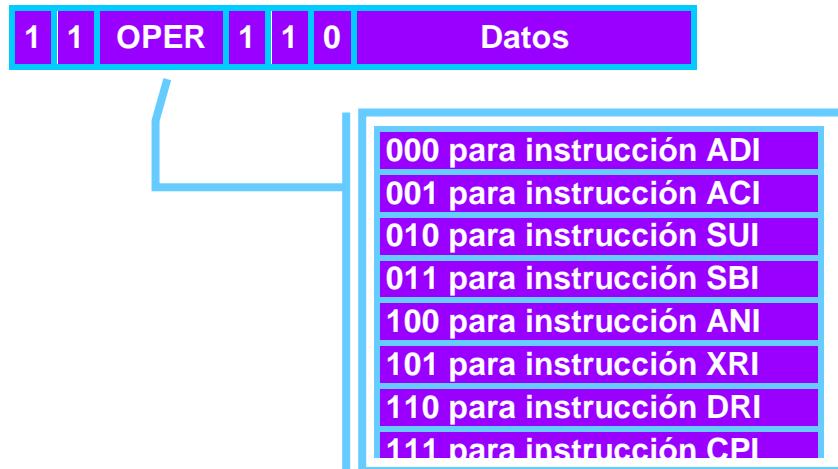
- La instrucción LXI ocupa 3 bytes con el siguiente formato:



- La instrucción MVI ocupa 2 bytes siguiente el formato siguiente:



- Por último, para el resto de las instrucciones, que ocupan 2 bytes, se cuenta con este formato:



La instrucción LXI opera sobre el par de registros especificado por PR, usando dos bytes de datos inmediatos.

La instrucción MVI opera sobre el registro especificado por REG, usando un byte de datos inmediatos. Si se hace referencia a la memoria, la instrucción opera sobre la posición de memoria de la misma determinada por los registros H y L. El registro H contiene los 8 bits más significativos de la dirección, mientras que el registro L contiene los 8 bits menos significativos.

Las restantes instrucciones operan sobre el acumulador, usando un byte de datos inmediatos. El resultado sustituye al contenido del acumulador.

A1.9.1. **LXI** Cargar un par de registros con un dato Inmediato

Instrucción	LXI pr, datos
Bits afectados	
Direccionamiento	Inmediato

Descripción

LXI es una instrucción de 3 bytes; su segundo y tercer byte contienen el dato que ha de ser cargado en el par de registros (PR). El primer operando debe especificar el par de registros a ser cargados, pueden ser los pares BC, DE, HL, o el SP. El segundo operando especifica los dos bytes a ser cargados. LXI es la única instrucción inmediata

que acepta un valor de 16 bits. El resto trabajan con datos de 8 bits.

Si el par de registros especificados es SP, el segundo byte de la instrucción sustituye a los 8 bits menos significativos del puntero de pila, mientras que el tercer byte de la instrucción reemplaza a los 8 bits más significativos del puntero de pila.

Formato



Ejemplos

1. La instrucción

LXI B, 00FFH

carga en el registro B el valor 00H y en el registro C el valor FFH.

2. La siguiente instrucción carga en el puntero de pila el valor 1000H

LXI SP, 1000H

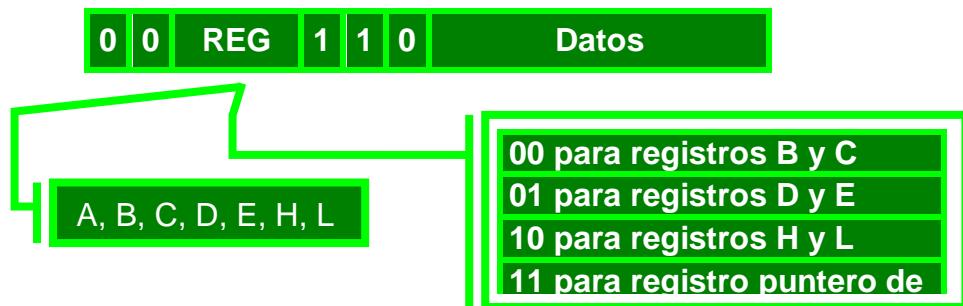
A1.9.2. **MVI** Cargar un registro con un dato Inmediato

Instrucción	MVI reg, datos
Bits afectados	
Direccionamiento	Inmediato

Descripción

El primer operando debe ser uno de los registros A, B, C, D, E, H o L, que será cargado con el dato especificado en el segundo operando (DATOS). El dato no debe exceder de un byte.

Formato



Ejemplos

(1). La instrucción

MVI H, 33H

carga en el registro H el valor 33H.

(2). La instrucción

MVI L, 44H

carga en el registro L el valor 44H.

(3). Supuestos los dos ejemplos anteriores, la instrucción

MVI M, 2AH

carga en la posición de memoria 3344H (dirección aportada por los registros H y L) el valor 2AH.

A1.9.3. ADI Sumar al acumulador un dato Inmediato

Instrucción	ADI
Bits afectados	Z, S, P, CY, AC
Direccionamiento	Inmediato

Descripción

Suma el valor del byte especificado en la instrucción (DATOS), al contenido del acumulador y deja el resultado en el acumulador. El dato debe ser expresado en forma de número, un ASCII constante, la etiqueta de un valor previamente definido o una expresión. El dato no debe exceder de un byte.

Se utiliza aritmética de complemento a dos.

Formato

1 1 0 0 0 1 1 0	Datos
-------------------------------	-------

Ejemplo

A continuación presentamos un ejemplo con 3 instrucciones:

(1).MVI A, 34

(2).ADI 20

(3).ADI -20

En todas las instrucciones se utilizan datos en base decimal. Así, por ejemplo, en la instrucción (2) el valor 20 es 14H.

La instrucción (1) carga en el acumulador el valor 22H.

La instrucción (2) realiza la siguiente suma:

Acumulador	22H	0 0 1 0 0 0 1 0
Dato inmediato	14H	0 0 0 1 0 1 0 0
		Nuevo acumulador 33H 0 0 1 1 0 1 1 0

El bit de paridad se pone a uno y el resto se quedan a cero.

La instrucción (3) restaura el valor del acumulador realizando la siguiente suma:

Acumulador	33H	0 0 1 1 0 1 1 0
Dato inmediato	ECH	1 1 1 0 1 1 0 0
		Nuevo acumulador 22H 0 0 1 0 0 0 1 0

Ahora los bits de paridad, acarreo y acarreo auxiliar se quedan a uno y el resto a cero.

A1.9.4. ACI Sumar al acumulador un dato Inmediato con arrastre

Instrucción Bits afectados Direccionamiento	ACI datos Z, S, P, CY, AC Inmediato
---	---

Descripción

Suma el contenido del byte especificado (DATOS) en la instrucción, al contenido del acumulador, añadiendo además el bit del acarreo. El resultado se almacena en el acumulador (perdiéndose así el anterior contenido del Acumulador).

El dato (DATOS) debe estar especificado en forma de número, en ASCII constante, como etiqueta de un valor previamente definido o una expresión. El dato no debe exceder de un byte.

Formato**Ejemplo**

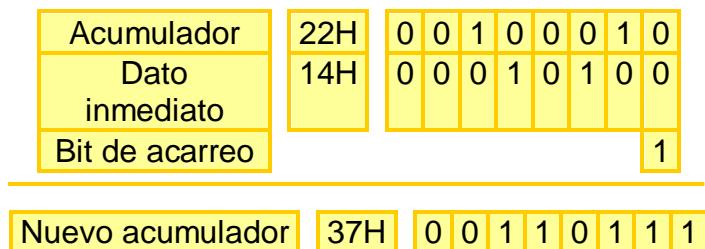
Tenemos las siguientes instrucciones:

- (1).MVI A, 34
- (2).ACI 20

y suponemos el bit de acarreo puesto a uno.

La instrucción (1) carga en el acumulador el valor 22H.

La instrucción (2) realiza la siguiente suma:



Todos los bits se ponen a cero.

A1.9.5. SUI Restar del acumulador un dato Inmediato

Instrucción Bits afectados Direccionamiento	SUI datos Z, S, P, CY, AC Inmediato
---	---

Descripción

El byte de datos inmediato se resta del contenido del acumulador usando aritmética de complemento a dos. El resultado se deja en el acumulador.

Ya que se trata de una operación de resta, el bit de acarreo se pone a uno cuando no hay acarreo del bit de más peso, y se pone a cero si tiene dicho acarreo.

Formato

1	1	0	1	0	1	1	0	Datos
---	---	---	---	---	---	---	---	-------

Ejemplo

A continuación presentamos un ejemplo con 2 instrucciones:

- (1).MVI A, B3H
- (2).SUI B3H

La instrucción (1) carga en el acumulador el valor B3H.

La instrucción (2) realiza la siguiente suma (usando el complemento a dos del dato inmediato):

Acumulador	B3H	1	0	1	1	0	0	1	1
Dato inmediato	6DH	0	1	0	0	1	1	0	1
<hr/>									
Nuevo acumulador	00H	0	0	0	0	0	0	0	0

Como era de esperar el resultado final del acumulador es cero ya que le estamos restando su propio valor. El valor 6DH del dato inmediato corresponde al complemento a dos del valor B3H que estamos restando.

Debido a que existe desbordamiento del séptimo bit se produce acarreo y se pone el bit de acarreo a cero.

El bit de paridad se pone a uno mientras que los demás permanecen inactivos.

A1.9.6. SBI Restar del acumulador un dato Inmediato con arrastre

Instrucción	SBI
Bits afectados	Z, S, P, CY, AC
Direcccionamiento	Inmediato

Descripción

El bit de acarreo se suma internamente al byte de datos inmediato. El valor obtenido se resta del contenido del acumulador usando aritmética de complemento a dos. El resultado se deja en el acumulador.

Esta instrucción, al igual que SBB, se usa preferentemente para realizar restas multi-byte.

Al igual que en el apartado anterior, el bit de acarreo se pone a uno si no hay acarreo del bit de más peso, poniéndose a cero si lo hay.

Formato

1 1 0 1 1 1 1 0	Datos
-------------------------------	-------

Ejemplo

Tenemos las siguientes instrucciones:

(1).MVI A, 00H

(2).SBI 01H

y suponemos el bit de acarreo puesto a cero.

La instrucción (1) carga en el acumulador el valor 00H.

La instrucción (2) realiza la siguiente suma (usando el complemento a dos del dato inmediato):



No hay acarreo, por lo que el bit de acarreo se pone a uno. Los bits de cero y acarreo auxiliar están a cero, mientras que los de signo y paridad se ponen a uno.

A1.9.7. **ANI** Función lógica AND entre el acumulador y un Dato Inmediato

Instrucción Bits afectados Direccionamiento	ANI datos Z, S, P, CY, AC Inmediato
---	---

Descripción

Realiza una operación Y lógica entre el dato (DATOS) especificado en la instrucción y el contenido del acumulador, el resultado queda en el acumulador. Se pone a cero el bit de acarreo. El dato, que no debe exceder de un byte, puede ser expresado en forma de número, un ASCII constante, la etiqueta de algún valor previamente definido o una expresión.

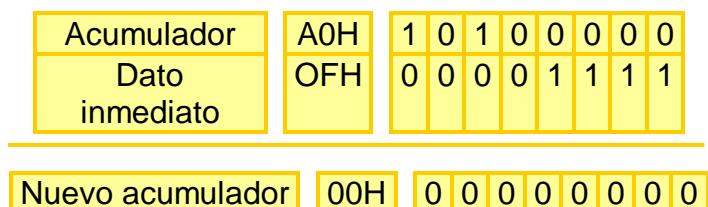
Formato**Ejemplo**

Disponemos de las siguientes instrucciones:

- (1).MVI A, A0H
- (2).ANI 0FH

La instrucción (1) carga en el acumulador el valor A0H.

La instrucción (2) realiza la siguiente operación AND bit a bit entre el acumulador y el dato inmediato 0FH:



La instrucción ANI del ejemplo pone a cero los cuatro bits de mayor peso, dejando invariables los cuatro menores. Ya que los cuatro bits de menor peso del acumulador eran cero, el resultado final es 00H con lo que el bit de cero se pondrá a cero.

A1.9.8. XRI**Función lógica O-EXCLUSIVO entre el acumulador y un dato Inmediato**

Instrucción
Bits afectados
Direccionamiento

XRI datos
Z, S, P, CY, AC
Inmediato

Descripción

Se realiza la función lógica O-EXCLUSIVO bit a bit entre un byte de datos inmediatos y el contenido del acumulador. El resultado se guarda en el acumulador. El bit de acarreo se pone a cero.

Formato



Ejemplo

Esta instrucción se suele utilizar para complementar bits específicos del acumulador dejando los restantes en su estado original. De este modo y suponiendo que el acumulador contiene ABH, la instrucción

XRI 80H

complementa el bit de más peso del acumulador, tal y como se muestra en la siguiente figura:



A1.9.9. ORI Función lógica OR entre el acumulador y un dato Inmediato

Instrucción	ORI datos
Bits afectados	Z, S, P, CY, AC
Direccionamiento	Inmediato

Descripción

ORI desarrolla una operación lógica OR entre el contenido especificado por DATOS y el contenido del acumulador. El resultado se deja en el acumulador. Los bits de acarreo y acarreo auxiliar se ponen a cero.

Formato

1	1	1	1	0	1	1	0	Datos
---	---	---	---	---	---	---	---	-------

Ejemplo

Si el acumulador inicialmente contiene 3CH, la instrucción

ORI F0H

realiza la siguiente operación OR bit a bit:

Acumulador 3CH	0 0 1 1 1 1 0 0
Dato inmediato F0H	1 1 1 1 0 0 0 0
Nuevo acumulador	FCH 1 1 1 1 1 1 0 0

Como vemos la instrucción ORI de nuestro ejemplo activa los cuatro bits de menor peso, dejando invariables los restantes.

A1.9.10. CPI Comparar el contenido del acumulador con un dato Inmediato

Instrucción Bits afectados	CPI datos Z, S, P, CY, AC
-------------------------------	------------------------------

Descripción

Compara el valor del byte especificado (DATOS) con el contenido del acumulador y posiciona los bits de cero y acarreo para indicar el resultado. El bit de cero indica igualdad. Un 0 en el acarreo indica que el contenido del acumulador es mayor que DATOS. Un 1 en el acarreo indica que el acumulador es menor que DATOS. Sin embargo, el significado del bit de acarreo es contrario cuando los valores tienen diferente signo o cuando uno de los valores está complementado. El valor de DATOS no debe exceder de un byte.

Formato



Ejemplo

Si tenemos la secuencia de instrucciones

- (1).MVI A, 25H
- (2).CPI 20H

La instrucción (1) carga en el acumulador el valor 25H.

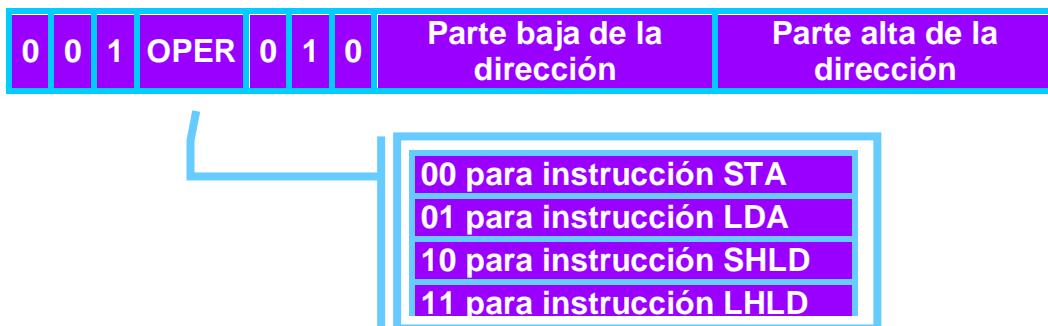
La instrucción (2) realiza la siguiente operación de suma (tomando el complemento a dos del dato inmediato, es decir, E1H):

Acumulador	25H	0 0 1 0 0 0 1 0 1
Dato inmediato	E1H	1 1 1 0 0 0 0 0 1
Nuevo acumulador		0 0 0 0 0 1 1 0

Existe desbordamiento del último bit, por lo que el bit de acarreo se pone a cero. El acumulador continua con su valor inicial pero el bit de cero está a cero, indicando que las cantidades no son iguales. Al estar el bit de acarreo a cero, nos indica que los datos inmediatos son menores que el contenido del acumulador.

A1.10. Instrucciones de direccionamiento directo

A continuación se describen las instrucciones que hacen referencia a una posición de memoria específica, cuyos dos bytes de dirección forman parte de la propia instrucción. Las instrucciones de este tipo ocupan tres bytes en la forma siguiente:



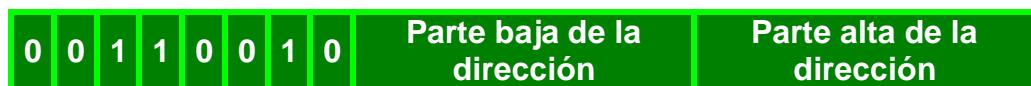
A1.10.1. STA Almacenamiento directo desde el Acumulador

Instrucción	STA	dir
Bits afectados		

Direccionamiento Directo

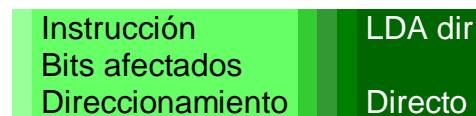
Descripción

STA DIR almacena una copia del contenido actual del acumulador en la posición de memoria especificada por DIR.

Formato**Ejemplo**

Todas las instrucciones que se muestran a continuación introducen el contenido del acumulador en la posición de memoria 0080H:

- STA 0080H // Base hexadecimal
- STA 128 // Base decimal
- STA 000000010000000B // Base binaria

A1.10.2. LDA Carga directa en el acumulador**Descripción**

LDA DIR carga el acumulador con el contenido de la memoria direccionada por DIR. La dirección puede ser puesta como un número, una etiqueta previamente definida o una expresión.

Formato**Ejemplo**

Todas las instrucciones que se muestran a continuación introducen en el acumulador el contenido de la posición de memoria 300H:

- LDA 300H
- LDA 3 * (16 * 16)
- LDA 200H + 256

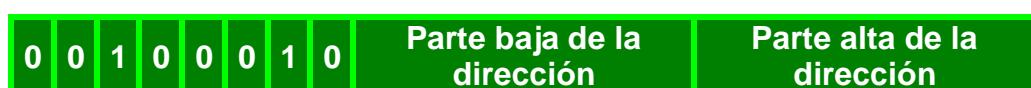
A1.10.3. **SHLD** Cargar directamente con H y L

Instrucción	SHLD
Bits afectados	dir
Direccionamiento	Directo

Descripción

Almacena una copia del registro L en la posición de memoria especificada por DIR, a continuación almacena una copia del registro H en la siguiente posición de memoria (DIR+1).

Formato



Ejemplo

Suponiendo que los registros H y L contienen respectivamente los valores 3CH y 54H, la instrucción

SHLD 34B3

efectuará las siguientes modificaciones en memoria:



A1.10.4. **LHLD** Cargar H y L directamente

Instrucción	LHLD dir
Bits afectados	
Direccionamiento	Directo

Descripción

LHLD DIR carga el registro L con una copia del byte almacenado en la posición de memoria especificada por DIR. Después carga el registro H con una copia del byte almacenado en la posición siguiente de memoria especificada por DIR. La instrucción LHLD esta prevista para cargar direcciones nuevas en los registros H y L.

Formato

0 0 1 0 1 0 1 0	Parte baja de la dirección	Parte alta de la dirección
-------------------------------	----------------------------	----------------------------

Ejemplo

En el caso en el que las posiciones de memoria AB24H y AB25H contengan respectivamente 22H y 33H, la ejecución de la instrucción

LHLD AB24H

hará que el registro L contenga 22H y el registro H contenga 33H.

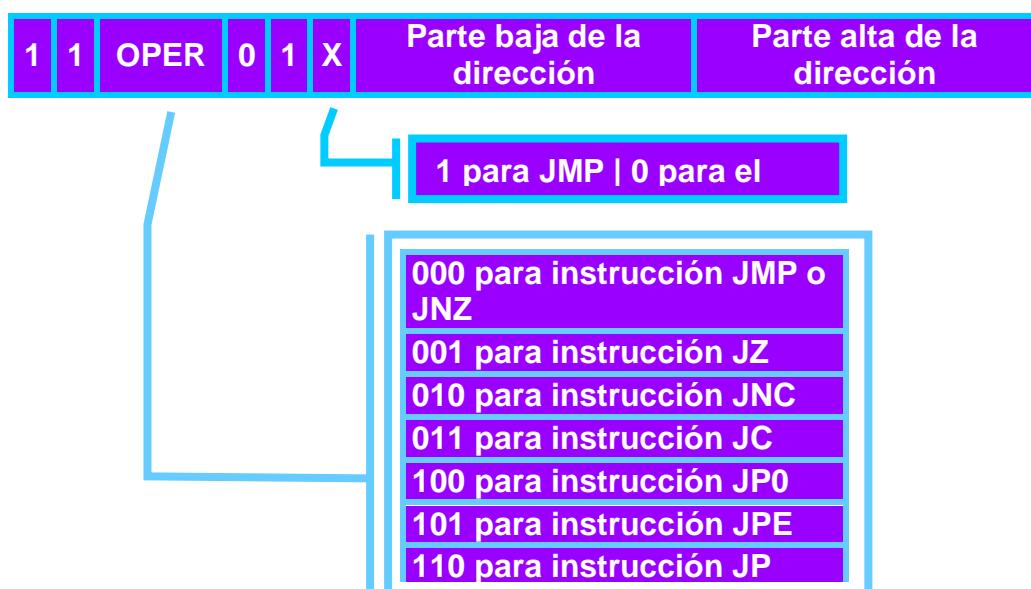
A1.11. Instrucciones de salto

A continuación se describen las instrucciones que modifican la secuencia normal de ejecución de las instrucciones de un programa. Estas instrucciones ocupan uno o tres bytes en la forma siguiente:

- La instrucción PCHL ocupa un byte con el siguiente formato:

1	1	1	0	1	0	0	1
---	---	---	---	---	---	---	---

- El resto de instrucciones de salto ocupan tres bytes con el formato siguiente:



Ciertas instrucciones de tres bytes de este tipo provocan un cambio en la secuencia normal de operaciones en la ejecución de un programa según unas determinadas condiciones.

Si la condición específica es verdadera, la ejecución del programa continúa en la dirección de memoria formada por la parte alta (tercer byte de la instrucción), y los ocho bits de la parte baja (segundo byte de la instrucción).

Si la condición específica es falsa, la ejecución del programa continúa en la próxima instrucción en secuencia.

A1.11.1. PCHL Cargar el contador de programa

Instrucción	PCHL
Bits afectados	Registro

Descripción

PCHL carga el contenido de los registros H y L en el contador de programa. Como el procesador busca la siguiente instrucción en la siguiente dirección del contador de programa, PCHL tiene el efecto de una instrucción de salto. El contenido de H va a los 8 bits más altos de contador de programa y el contenido de L va a los 8 bits más bajos.

Formato



Ejemplo

Si el registro H contiene A5H y el registro L contiene 9DH, la instrucción

PCHL

hace que el programa en curso continúe ejecutándose en la dirección de memoria A59DH.

A1.11.2. JMP Salto incondicional

Instrucción	JMP dir
Bits afectados	
Direccionamiento	Inmediato

Descripción

La instrucción JMP DIR altera la ejecución del programa cargando el valor especificado por DIR en el contador de programa.

Formato

A1.11.3. JC Saltar si hay acarreo

Instrucción	JC dir
Bits afectados	
Direccionamiento	Inmediato

Descripción

La instrucción JC DIR comprueba el valor del bit de acarreo. Si es un 1 la ejecución del programa continúa en la dirección especificada por DIR. Si es un 0 el programa

continúa su ejecución normal de forma secuencial.

Formato

1	1	0	1	1	0	1	0	Parte baja de la dirección	Parte alta de la dirección
---	---	---	---	---	---	---	---	----------------------------	----------------------------

A1.11.4. JNC Saltar si no hay acarreo

Instrucción	JNC	Bits afectados	dir	Direccionamiento	Inmediato
-------------	-----	----------------	-----	------------------	-----------

Descripción

La instrucción JNC DIR comprueba el estado del bit acarreo. Si esta a 0 el programa cambia a la dirección especificada por DIR. Si esta a 1 la ejecución del programa continúa normalmente.

Formato

1	1	0	1	0	0	1	0	Parte baja de la dirección	Parte alta de la dirección
---	---	---	---	---	---	---	---	----------------------------	----------------------------

A1.11.5. JZ Saltar si hay cero

Instrucción	JZ	Bits afectados	dir	Direccionamiento	Inmediato
-------------	----	----------------	-----	------------------	-----------

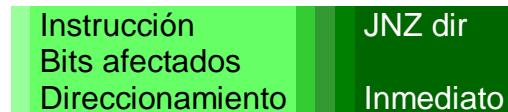
Descripción

La instrucción JZ DIR comprueba el bit de cero. Si está a 1 el programa continúa en la dirección expresada por DIR. Si está a 0 continúa con la ejecución secuencial normal.

Formato



A1.11.6. **JNZ** Saltar si no hay cero



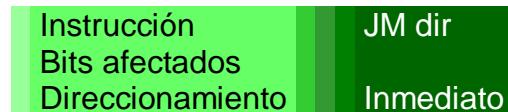
Descripción

La instrucción JNZ DIR comprueba el valor del bit de cero. Si el contenido del acumulador no es cero (Bit de cero = 0) el programa continúa en la dirección especificada por DIR. Si el contenido del acumulador es cero (Bit de cero = 1) el programa continúa su ciclo normal.

Formato



A1.11.7. **JM** Saltar si hay signo negativo



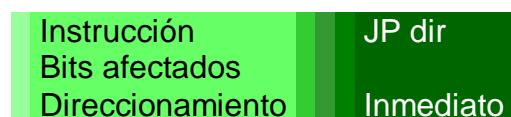
Descripción

La instrucción JM DIR comprueba el estado del bit de signo. Si el contenido del acumulador es negativo (bit de signo = 1) la ejecución del programa continúa en la dirección especificada por DIR. Si el contenido del acumulador es positivo (bit de signo = 0) continúa la ejecución de la secuencia normal.

Formato



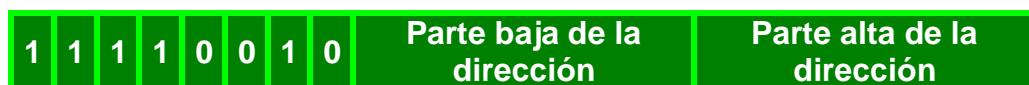
A1.11.8. **JP** Saltar si hay signo positivo



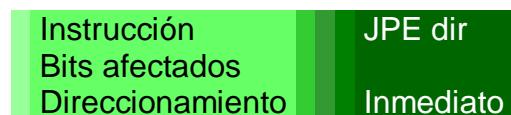
Descripción

La instrucción JP DIR comprueba el estado de bit del signo. Si el contenido del acumulador es positivo (bit de signo = 0) la ejecución del programa continúa con la dirección especificada por DIR. Si el contenido del acumulador es negativo (bit de signo = 1) continúa el programa con su ejecución normal.

Formato



A1.11.9. **JPE** Saltar si la paridad es par



Descripción

La paridad existe si el byte que esta en el acumulador tiene un número par de bits. El bit de paridad se pone a 1 para indicar esta condición.

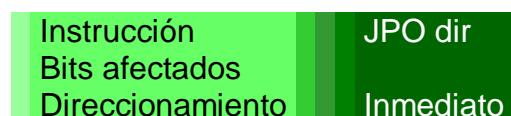
La instrucción JPE DIR comprueba la situación del bit de paridad. Si esta a 1, la ejecución del programa continúa en la dirección especificada por DIR. Si esta a 0, continúa con la siguiente instrucción de forma secuencial.

Las instrucciones JPE y JPO son especialmente usadas para comprobar la paridad de los datos de entrada. (Sin embargo con la instrucción IN los bits no actúan. Esto puede evitarse sumando 00H al acumulador para activarlos).

Formato

1	1	1	0	1	0	1	0	Parte baja de la dirección	Parte alta de la dirección
---	---	---	---	---	---	---	---	----------------------------	----------------------------

A1.11.10. JPO Saltar si la paridad es impar



Descripción

La instrucción JPO DIR comprueba el estado del bit de paridad. Si esta a 0, el programa continúa en la dirección marcada por DIR. Si está a 1 continua con la secuencia normal.

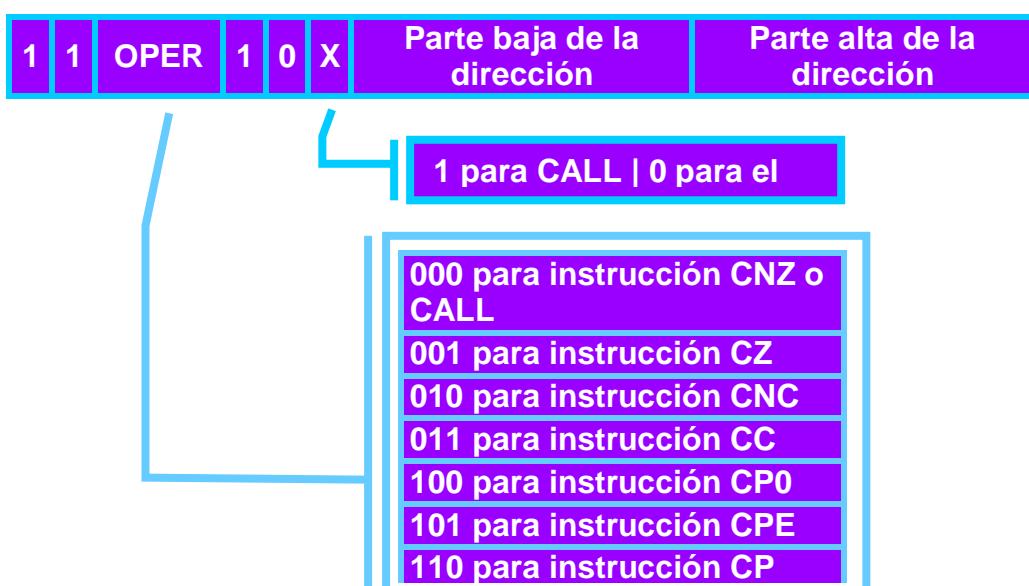
Formato

1	1	1	0	0	0	1	0	Parte baja de la dirección	Parte alta de la dirección
---	---	---	---	---	---	---	---	----------------------------	----------------------------

A1.12. Instrucciones de llamada a subrutina

A continuación se describen las instrucciones que llaman a subrutinas. Estas instrucciones operan en la misma forma que las instrucciones de salto, provocando una alteración en la secuencia de ejecución de las instrucciones, pero además, en el momento de su ejecución, se almacena en la pila una dirección de retorno, que es utilizada por las instrucciones de RETORNO (ver instrucciones de Retorno de Subrutinas en este mismo capítulo).

Las instrucciones de este tipo utilizan tres bytes en el formato siguiente:



En las instrucciones de llamada, las instrucciones se codifican como en las instrucciones de salto, es decir, almacenando en primer lugar el byte de dirección de memoria menos significativo.

A1.12.1. **CALL** Llamada incondicional

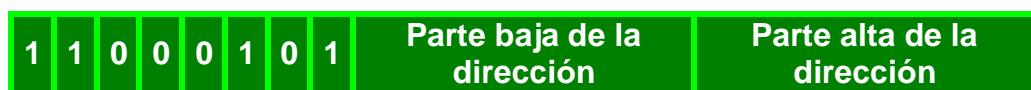
Instrucción	CALL dir
Bits afectados	
Direccionamiento	Inmediato / Registro indirecto

Descripción

CALL guarda el contenido del contador de programa (la dirección de la próxima instrucción secuencial) dentro del stack y a continuación salta a la dirección especificada por DIR.

Cada instrucción CALL o alguna de sus variantes implica una instrucción RET (retorno), de lo contrario el programa podría "perderse" con consecuencias impredecibles. La dirección debe ser especificada como un número, una etiqueta, o una expresión. La etiqueta es lo más normal (El ensamblador invierte los bytes alto y bajo de dirección durante el proceso de ensamblado). Las instrucciones CALL se emplean para llamadas a subrutinas y debemos tener presente que siempre emplean el stack.

Formato



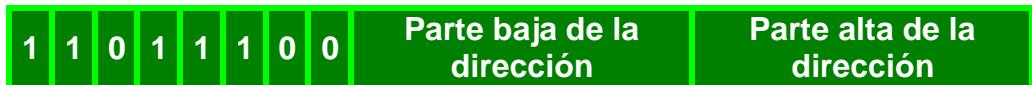
A1.12.2. CC Llamar si hay acarreo



Descripción

CC comprueba el estado del bit de acarreo. Si el bit está a 1, CC carga el contenido del contador de programa en el stack y a continuación salta a la dirección especificada por DIR. Si el bit esta a 0, la ejecución del programa continúa con la próxima instrucción de su secuencia normal. Aunque el uso de una etiqueta es lo más normal, la dirección puede ser especificada también como un número o una expresión.

Formato



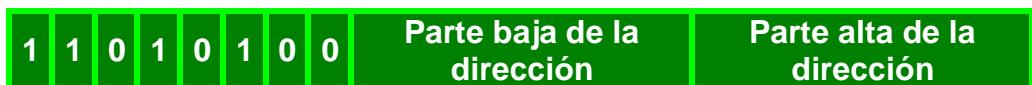
A1.12.3. CNC Llamar si no hay acarreo



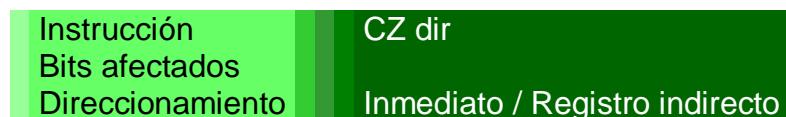
Descripción

CNC chequea el valor del bit de acarreo. Si está en cero CNC carga el contenido de contador de programa en el stack y a continuación salta a la dirección especificada por la instrucción en DIR. Si el bit está a 1, el programa continúa con su secuencia normal. Aunque el uso de una etiqueta es lo más común, la dirección puede también estar indicada por un número o por una expresión.

Formato

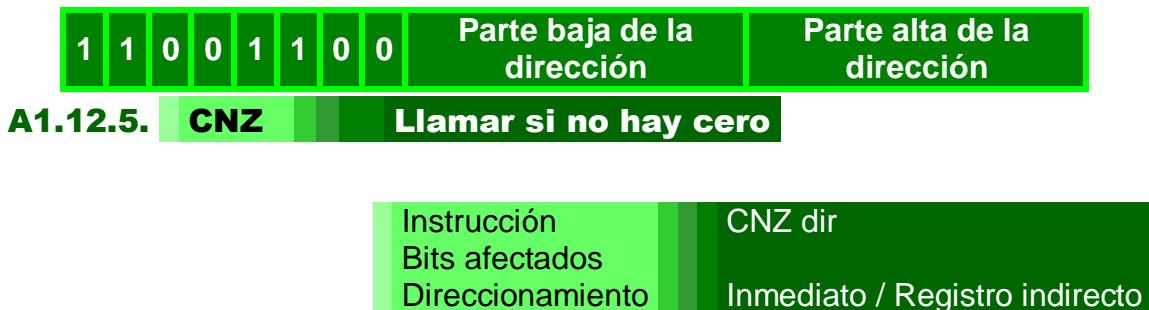


A1.12.4. CZ Llamar si hay cero



Descripción

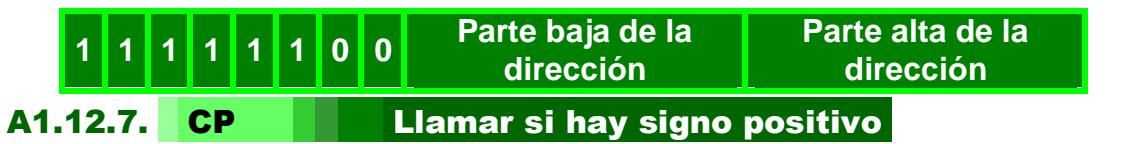
CZ chequea el bit de cero. Si el bit está a 1 (indicando que el contenido del acumulador es cero), CZ carga el contenido del contador de programa en el stack y salta a la dirección especificada en DIR. Si el bit está a 0 (indicando que el contenido del acumulador es distinto de cero) el programa continúa su desarrollo normal.

Formato**Descripción**

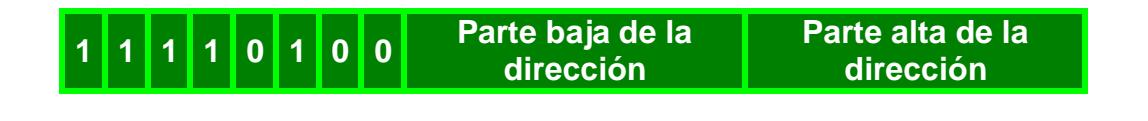
CNZ chequea el bit de Cero. Si está en 0 (indicando que el contenido del acumulador no es cero), CNZ manda el contenido del contador de programa al stack y salta a la dirección especificada por DIR. Si el bit está a 1 el programa continúa su desarrollo normal.

Formato**Descripción**

CM comprueba el estado del bit del signo. Si el bit esta a 1 (indicando que el contenido del acumulador es negativo) CM manda el contenido del contador de programa al stack y salta a la dirección especificada por DIR. Si el bit es 0 la ejecución del programa continúa con su secuencia normal. El uso de la etiqueta es lo más corriente, pero la dirección puede especificarse también por un número o una expresión.

Formato**Descripción**

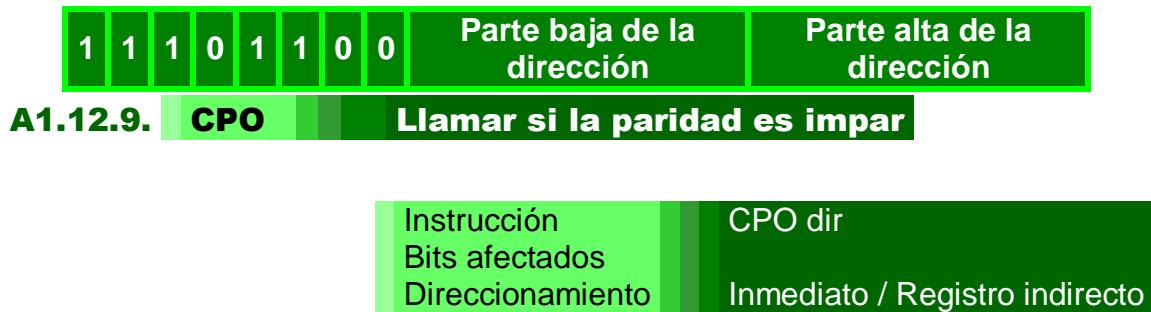
CP chequea el valor del bit de signo. Si está a 0 (indicando que el contenido del acumulador es positivo), CP envía el contenido del contador de programa al stack y salta a la dirección especificada por DIR. Si el bit tiene un 1, continúa el programa normalmente con la instrucción siguiente.

Formato**Descripción**

Existe paridad en un byte si el número de unos que tiene es par. El bit de paridad se pone a 1 para indicar esta condición. CPE chequea el valor del bit de paridad. Si tiene un 1, CPE envía el contenido del contador de programa al stack y salta a la dirección especificada por la instrucción en DIR. Si el bit tiene un cero, el programa continúa

normalmente.

Formato



Descripción

CPO chequea el bit de paridad. Si el bit esta a 0, CPO carga el contenido del contador de programa en el stack y salta a la dirección especificada en DIR. Si el bit está a 1 el programa continúa su desarrollo normal.

Formato

1	1	1	0	0	1	0	0	Parte baja de la dirección	Parte alta de la dirección
---	---	---	---	---	---	---	---	----------------------------	----------------------------

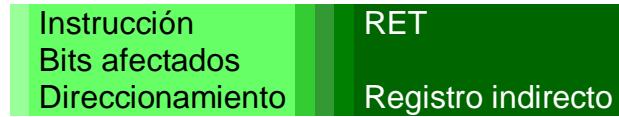
A1.13. Instrucciones de retorno desde subrutinas

A continuación se describen las instrucciones utilizadas para realizar un retorno desde las subrutinas. Estas instrucciones colocan en el contador de programa la última dirección puesta en la pila, haciendo que la ejecución del programa continúe en esta dirección.

Las instrucciones de este tipo utilizan un byte en el formato siguiente:



Ciertas instrucciones de este tipo realizan la operación de retorno bajo ciertas condiciones específicas. Si la condición especificada se cumple (es verdadera), tendrá lugar la operación de retorno. Por el contrario, si la condición no se cumple, la ejecución del programa continuará en la próxima instrucción en secuencia.

A1.13.1. RET **Retorno incondicional****Descripción**

Se realiza una operación de retorno incondicional.

La instrucción RET echa fuera dos bytes de datos del stack y los mete en el registro contador de programa. El programa continúa entonces en la nueva dirección. Normalmente RET se emplea conjuntamente con CALL.

Formato**A1.13.2. RC** **Retorno si hay acarreo****Descripción**

La instrucción RC comprueba el estado del bit de acarreo. Si tiene un 1 (indicando que hay acarreo) la instrucción saca dos bytes del stack y los mete en el contador de programa. El programa continúa en la nueva dirección suministrada. Si el bit es 0, el programa continúa en la siguiente instrucción de la secuencia normal.

Formato

A1.13.3. RNC Retorno si no hay acarreo**Descripción**

La instrucción RNC comprueba el bit de acarreo. Si está a 0 indicando que no hay acarreo, la instrucción echa fuera del stack dos bytes y los carga en el contador de programa. Si el bit está a 1 continúa el ciclo normal.

Formato

1	1	0	1	0	0	0	0
---	---	---	---	---	---	---	---

A1.13.4. RZ Retorno si hay cero**Descripción**

La instrucción RZ comprueba el bit de cero. Si está a 1, indicando que el contenido del acumulador es cero, la instrucción echa fuera del stack dos bytes y los carga en el contador de programa. Si el bit está a 0, continúa el ciclo normal.

Formato

1	1	0	0	1	0	0	0
---	---	---	---	---	---	---	---

A1.13.5. RNZ Retorno si no hay cero**Descripción**

La instrucción RNZ comprueba el bit cero. Si está a 0, indicando que el contenido del acumulador no es cero, la instrucción echa fuera del stack dos bytes y los carga en el contador de programa. Si el bit está a 1, continúa el ciclo normal.

Formato

1	1	0	0	0	0	0	0
---	---	---	---	---	---	---	---

A1.13.6. RM Retorno si hay signo negativo**Descripción**

La instrucción RM comprueba el bit de signo. Si tiene un 1, indicando dato negativo en el acumulador, la instrucción echa dos bytes fuera del stack y los mete en el contador de programa. Si el bit tiene 0, continúa el programa normal con la siguiente instrucción.

Formato

1	1	1	1	1	0	0	0
---	---	---	---	---	---	---	---

A1.13.7. RP Retorno si hay signo positivo**Descripción**

La instrucción RP comprueba el bit signo. Si está a 0, indicando que el contenido del acumulador es positivo, la instrucción echa fuera del stack dos bytes y los carga en el contador de programa. Si el bit está a 1 continúa el ciclo normal.

Formato

1	1	1	1	0	0	0	0
---	---	---	---	---	---	---	---

A1.13.8. RPE Retorno si la paridad es par**Descripción**

La instrucción RPE comprueba el bit de paridad. Si está a 1, indicando que existe paridad, la instrucción echa fuera del stack dos bytes y los carga en el contador de programa. Si el bit está a 0 continúa el ciclo normal. (Existe paridad si el byte que está en el acumulador tiene un número par de bits, colocándose el bit de paridad a 1 en este caso).

Formato

1	1	1	0	1	0	0	0
---	---	---	---	---	---	---	---

A1.13.9. RPO Retorno si la paridad es impar**Descripción**

La instrucción RPO comprueba el bit de paridad. Si está a 0, indicando que no hay paridad, la instrucción echa fuera del stack dos bytes y los carga en el contador de programa. Si el bit está a 1, continúa el ciclo normal.

Formato

1	1	1	0	0	0	0	0
---	---	---	---	---	---	---	---

A1.14. Instrucción RST



Descripción

Es una instrucción CALL para usar con interrupciones. RST carga el contenido del contador de programa en el stack, para proveerse de una dirección de retorno y salta a una de las "ocho" direcciones determinadas previamente.

Un código de tres bits incluido en el código de operación de la instrucción RST especifica la dirección de salto. Esta instrucción es empleada por los periféricos cuando intentan una interrupción.

Para volver a la instrucción en que ha tenido lugar la interrupción, se debe utilizar una instrucción de RETORNO.

Formato



donde EXP es un número binario entre 000B y 111B.

Por tanto, según la combinación de 0 y 1 que demos a EXP obtendremos los distintos formatos y las distintas direcciones de las interrupciones, que serán:

FORMATO

CONTADOR DE PROGRAMA

1100 0111 → C7H

0000 0000 0000 0000

→ 0000H

1100 1111	→ CFH	0000 0000 0000 1000	→ 0008H
1101 0111	→ D7H	0000 0000 0001 0000	→ 0010H
1101 1111	→ DFH	0000 0000 0001 1000	→ 0018H
1110 0111	→ E7H	0000 0000 0010 0000	→ 0020H
1110 1111	→ EFH	0000 0000 0010 1000	→ 0028H
1111 0111	→ F7H	0000 0000 0011 0000	→ 0030H
1111 1111	→ FFH	0000 0000 0011 1000	→ 0038H

Ejemplos

1. La instrucción

RST 3

Llama a la subrutina de la posición 0018H.

2. La instrucción

RST 10

es una instrucción ilegal ya que el argumento de RST debe ser un número entre 0 y 7.

A1.15. Instrucciones del FLIP-FLOP de interrupción

En este apartado estudiamos las instrucciones que operan directamente sobre el flip-flop de actuación de interrupciones. Todas estas instrucciones ocupan un byte siguiendo el formato que se muestra:



A1.15.1. EI Activar interrupciones

Instrucción	Bits afectados	Direccionamiento
EI		

Descripción

La instrucción EI pone en servicio el sistema de interrupciones a partir de la siguiente instrucción secuencial del programa. Esta instrucción activa el flip-flop INTE.

Se puede desconectar el sistema de interrupciones poniendo una instrucción DI al principio de una secuencia, puesto que no se puede predecir la llegada de una interrupción.

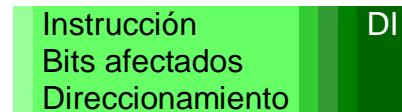
Al final de la secuencia se incluye la instrucción EI que vuelve a habilitar el

sistema de interrupciones. (RESET también pone fuera de servicio el sistema de interrupciones además de poner el contador de programa a cero).

Formato



A1.15.2. DI Desactivar interrupciones



Descripción

Esta instrucción desactiva el flip-flop INTE. Después de la ejecución de una instrucción DI, el sistema de "interrupciones" está sin posibilidad de ponerse en marcha. En aplicaciones que empleen las interrupciones, la instrucción DI se emplea solamente cuando una determinada secuencia no debe ser interrumpida. Por ejemplo, se puede poner fuera de servicio el sistema de interrupciones incluyendo una instrucción DI el principio del código de secuencia.

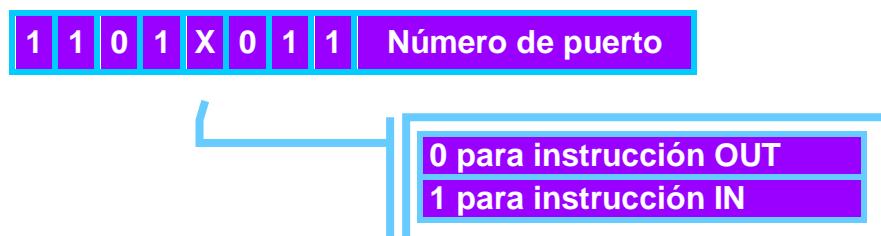
La interrupción TRAP del 8085 no puede ser puesta fuera de servicio. Esta interrupción especial está prevista para serios problemas que pueden presentarse independientemente del bit de interrupción (fallo de alimentación, etc.).

Formato



A1.16. Instrucciones de Entrada / Salida

Ahora nos ocuparemos de las instrucciones que hacen que los datos entren o salgan de la CPU. Estas instrucciones ocupan dos bytes en la forma siguiente:



El número de puerto viene definido por el hardware del sistema, no estando bajo el control del programador. Este, únicamente selecciona uno de ellos para realizar la correspondiente operación.

A1.16.1. IN Entrada

Instrucción	Bits afectados	IN port
Direcciónamiento		Directo

Descripción

La instrucción IN PORT lee los 8 bits de datos que hay en el "PORT" especificado y los carga en el acumulador. El operando debe ser un número o una expresión que produzca un valor comprendido entre 00H y FFH.

Formato**Ejemplos**

1. La instrucción

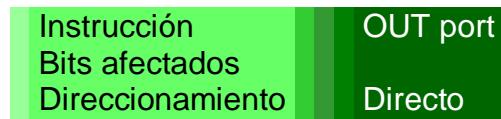
IN 2

deposita en el acumulador los datos de entrada del puerto 2.

2. La instrucción

IN 3*2

deposita en el acumulador los datos de entrada del puerto 6.

A1.16.2. OUT Salida**Descripción**

OUT PORT pone el contenido del acumulador en el bus de datos de 8 bits del puerto seleccionado. El número de puertos oscila de 0 a 255 y es duplicado en el bus de direcciones. Es la lógica externa la encargada de seleccionar el puerto y aceptar el dato de salida. El operando (PORT) debe especificar el número del puerto de salida seleccionado.

Formato**Ejemplos**

1. La instrucción

OUT 2

envía el contenido del acumulador al puerto de salida número 2.

2. La instrucción

OUT 3*2

envía el contenido del acumulador al puerto de salida número 6.

A1.17. Instrucción de alto HLT

Instrucción	HLT
Bits afectados	
Direccionamiento	

Descripción

La instrucción HLT detiene el procesador.

El contador de programa contiene la dirección de la próxima instrucción secuencial. Por otro lado los bits y registros permanecen inactivos. Una vez en estado de parada el procesador puede volver a ser arrancado solamente por un acontecimiento externo, es decir una interrupción. Por tanto debemos asegurarnos que las interrupciones estén en disposición de ser activadas antes de ejecutar la instrucción HLT.

Si se ejecuta HLT estando las interrupciones fuera de servicio, la única manera de volver arrancar el procesador será mediante un RESET o a través de la interrupción TRAP.

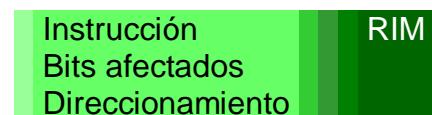
El procesador puede salir temporalmente del estado de parada para servir un acceso directo a memoria, sin embargo terminado el acceso directo vuelve al estado de parada.

Un propósito básico de la instrucción HLT es permitir una pausa al procesador mientras espera por la interrupción de un periférico.

Formato

0	1	1	1	0	1	1	0
---	---	---	---	---	---	---	---

A1.18. Instrucción RIM para la lectura de la máscara de interrupciones



Descripción

RIM carga los 8 bits de datos siguientes en el acumulador:

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
SID	I7.5	I6.5	I5.5	IE	M7.5	M6.5	M5.5

donde:

- SID: Bit presente en la entrada serie.
- I7.5: Interrupción 7.5 pendiente si está a 1.
- I6.5: Interrupción 6.5 pendiente si está a 1.
- I5.5: Interrupción 5.5 pendiente si está a 1.
- IE: Las interrupciones son autorizadas si está a 1.
- M7.5: La interrupción 7.5 está prohibida si está a 1.
- M6.5: La interrupción 6.5 está prohibida si está a 1.

- M5.5: La interrupción 5.5 está prohibida si está a 1.

Formato

0	0	1	0	0	0	0	0
---	---	---	---	---	---	---	---

A1.19.**Instrucción SIM para posicionar la máscara de interrupciones**

Instrucción	SIM
Bits afectados	
Direccionamiento	

Descripción

SIM es una instrucción de usos múltiples que utiliza el contenido del acumulador para posicionar el enmascaramiento de interrupciones para las RST 5.5, RST 6.5, RST 7.5; pone a cero el flanco sensitivo de la entrada RST 7.5 y saca el bit 7 del acumulador al latch de datos de salida serie. La estructura de la instrucción SIM es como sigue:



donde:

- SOD: Bit a presentar sobre la salida serie.
- SOE: La salida serie está autorizada si está a 1.
- R7.5: Reset de RST 7.5. Si es 1 el flip-flop se pone a 0.
- MSE: Si es 1 los enmascarados están autorizados.
- M7.5: La interrupción 7.5 queda prohibida si está a 1.
- M6.5: La interrupción 6.5 queda prohibida si está a 1.

- M5.5: La interrupción 5.5 queda prohibida si está a 1.

Si el bit 3 se pone a 1, la función poner "mask" pasa a estar permitida.

Los bits 0 al 2 ponen en servicio la correspondiente interrupción RST colocando un 0 en la interrupción que deseamos habilitar. Si colocamos un 1 en alguno de los bits 0 al 2, la interrupción correspondiente no se cumplirá.

Si el bit 3 tiene un 0, los bits 0 al 2 no tienen efectos. Se debe usar esta peculiaridad para enviar un bit de salida serie sin afectar al enmascaramiento de las interrupciones. (La instrucción DI anula la SIM).

Si el bit 6 (SOE) se pone a 1 se habilita la función de salida serie.

El bit 7 se sitúa en la salida SOD donde puede ser tratado por los aparatos periféricos. Si el bit 6 se pone a cero, el bit 7 no tendrá efecto alguno, siendo ignorado.

Formato

0	0	1	1	0	0	0	0
---	---	---	---	---	---	---	---

Apéndice A5

Programas de ejemplo

Se han escrito varios programas en ensamblador que muestran el funcionamiento de los componentes que incorpora el simulador. Realizar el negativo de una imagen de niveles de gris cargada en memoria o implementar el juego de la serpiente (*snake* en inglés) son algunos ejemplos que pasamos a comentar en las secciones siguientes.

Demostración:

Nombre: negativo.asm

Procesador: 8085

Dispositivos: Pantalla Grafica (256)

Líneas: 28

Descripción:

Programa de muestra que invierte la imagen actual en pantalla.

```
.org 100H

mvi H, 10H
mvi L, 00H ; en HL la posicion de memoria

otro:
mvi a, FFh
SUB M

mov M, a
INX H
;comprueba parte alta
mvi a, 4EH
cmp H
JZ comprueba_LO
jmp otro

comprueba_LO:           ; comprueba parte baja
mvi a, 80h
cmp L
JZ fin
jmp otro

fin:
hlt
```



Demostración:

Nombre: leds.asm
Procesador: 8085
Dispositivos: Panel de Leds (1 linea)
 Generador de Interrupciones
Líneas: 47

Descripción:

Programa que genera un movimiento ordenado y oscilante de una luz mediante un vector de leds.

```
;LUZ COCHE FANTASTICO
;
; requiere led en puerto 0
;              interrupcion rst 7.5 cada 1 segundo

.org 1000h

mvi c,128
mov d,0
salto:
mov a,c
out 0h
jmp salto

.org 003ch
;Subrutina cada 1 segundo
sub:
mov a,d
cpi 0
jnz mub
```

```

mov a,c          ;direccion -->
rar
mov c,a
mov a,c
cpi 0
jnz rub
mvi c,1
mvi d,1

mub:
mov a,c          ;direccion <--
ral
mov c,a
mov a,c
cpi 0
jnz rub
mvi c,128
mvi d,0

rub:
ei
ret

```



Ejemplo de una utilidad:

Nombre: pantalla.asm
Procesador: 8085
Dispositivos: Pantalla de Texto
 Teclado
 Generador de Interrupciones por teclado
Líneas: 59

Descripción:

Simulación de un terminal de texto.

```

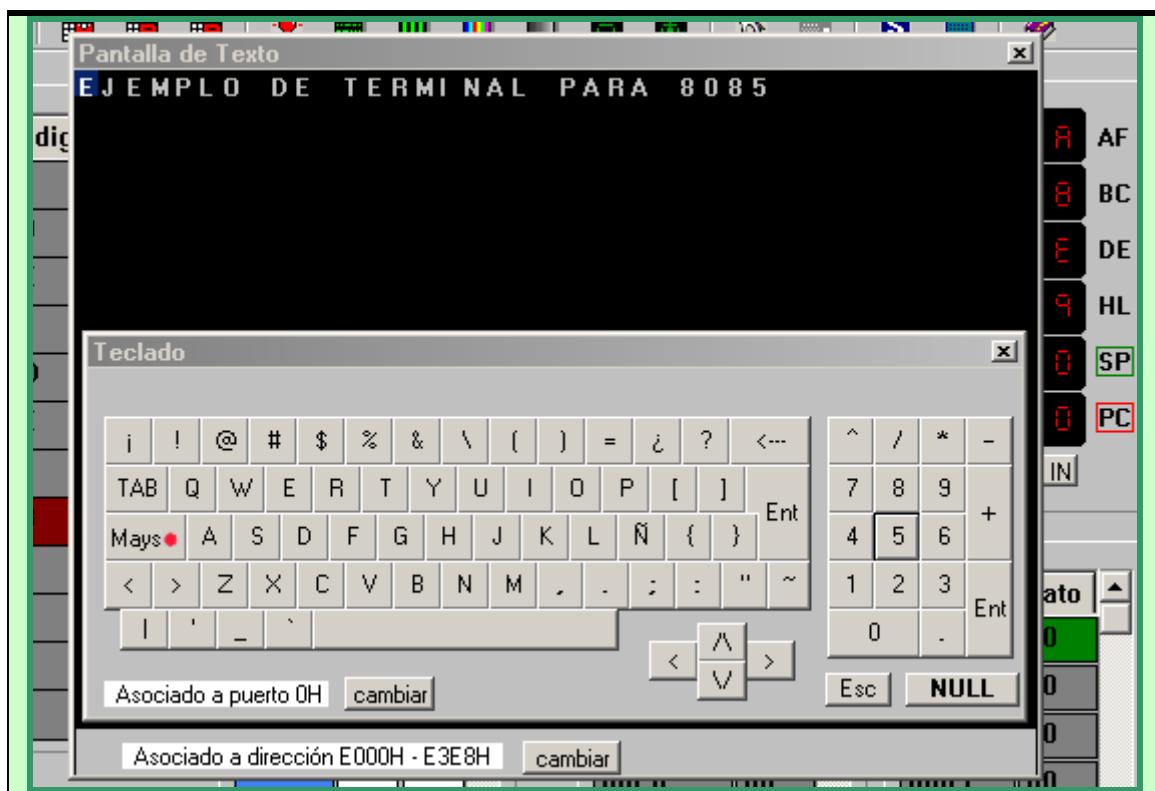
; Ejemplo de programa
; Simulador de terminal
; Asociado a interrupción TRAP

.define
    texto E000h
    tamtexto 25*40
.org 1000h

; -----
; PROGRAMA PRINCIPAL

```

```
; -----  
mvi B, E0h  
mvi C, 00h  
call clear_all  
bucle:  
    jmp bucle  
  
.org 0024h           ; Dirección de interrupción TRAP  
  
; -----  
; RUTINA QUE LEE DEL TECLADO Y ESCRIBE EN MEMORIA  
; -----  
    in 00h  
    cpi 0  
    jz no_tecla  
    stax B  
    inx B  
no_tecla:  
    ret  
  
clear_all:  
    LXI H, texto          ;cargamos origen  
    LXI D, texto+tamtexto ;cargamos fin  
repite_c:  
    MVI A,32  
    MOV M,A              ;borrar punto de memoria  
    INX H                ;incrementar dirección  
    call comparador  
    cpi 1  
    jz fin_clear  
    jmp repite_c  
fin_clear:  
    ret  
  
comparador:           ;compara DE con HL (en 16 bits). Devuelve  
A=1 si igual  
    MOV A,E  
    CMP L  
    JNZ no_igual  
    MOV A,D  
    CMP H  
    JNZ no_igual  
    MVI A, 1  
    ret  
no_igual:  
    MVI A,0  
    ret
```



Ejemplo de una utilidad:

Nombre: Reloj.asm

Procesador: 8085

Dispositivos: Visualizador de 7 segmentos
Generador de Interrupciones

Líneas: 83

Descripción:

Programa que convierte a un 8085 en un reloj digital con segundero y minutero.

```
; Ejemplo de programa
; Reloj digital
; Asociado a interrupción TRAP

.data DDH
    DB 77h, 44h, 3Eh, 6Eh, 4Dh, 6Bh, 7Bh, 46h, 7Fh, 4Fh
    ;dgitos sin punto
    DB F7h, C4h, BEh, EEh, CDh, EBh, FBh, C6h, FFh, CFh
    ;dgitos con punto

.org 1000h

; -----
; PROGRAMA PRINCIPAL
; -----
        mvi B, 00h
        mvi C, DDh
        mvi D, 00h
```

```
mvi E, DDh
mvi L, E7h
mvi H, DDh
mvi A, 77h
out 6d
out 7d
out 4d
mvi A, F7h
out 5d
bucle:
    jmp bucle

.org 0024h           ; Dirección de interrupción TRAP

; -----
; RUTINA QUE AUMENTA EL TIEMPO
; -----
ldax B
cpi 4Fh
jz suma_segundo
inx B
ldax B
out 7d
ret
suma_segundo:
    mvi C, DDh
    ldax B
    out 7d
    ldax D
    cpi 6Bh
    jz suma_minutol
    inx D
    ldax D
    out 6d
    ret
suma_minutol:
    mvi E, DDh
    ldax D
    out 6d
    mov D, C
    mov C, L
    ldax B
    cpi CFh
    jz suma_minuto2
    inx B
    ldax B
    out 5d
    mov L, C
    mov C, D
    mov D, 00h
    ret
suma_minuto2:
    mvi L, E7h
    mov D, C
    mov C, L
    ldax B
    out 5d
    mov D, C
    mov C, H
    inx B
```

```

ldax B
out 4d
mov H, C
mov C, D
mov D, 00h
ret

```



Ejemplo de un Juego Interactivo:

Nombre: Snake.asm
Procesador: 8085
Dispositivos: Teclado
 Pantalla Grafica
 Generador de Interrupciones
Líneas: 236

DESCRIPCIÓN:

Conocido juego de la serpiente, consiste en comer los puntos de comida si mordernos a nosotros mismos.

```

; Ejemplo de Programa en ensamblador para el simulador de 8085
; SNAKE 8085
; Comer sin mordenos a nosotros mismos

.define
    memVideo A000h          ;Origen de la memoria de Video
    sizeVideo 160*100        ;Tamaño de la memoria de Video
    mitadVideo memVideo+sizeVideo/2 ;Posicion intermedia
    teclado 0h               ;Puerto del teclado
    up      (-160)&FFFFh
    down   160
    left   -1
    right  1
    tecla_up  1Eh
    tecla_down 1Fh
    tecla_left 11h
    tecla_right 10h
    comienzo  mitadVideo+80

.data 0b
cuanto: dB 10h
cola: dW comienzo
pos: dB 0
pos_pantalla: dW
memVideo+580H,memVideo+1000H,memVideo+2500H,memVideo+3000H

```

```

.org 500H
;      call clear_all
LXI H, comienzo
call pon_comida
repite:
    IN teclado
    jmp repite

comparador:           ; compara DE con HL (en 16 bits). Devuelve
A=1 si igual
    MOV A,E
    CMP L
    JNZ no_igual
    MOV A,D
    CMP H
    JNZ no_igual
    MVI A, 1
    ret
no_igual:
    MVI A,0
    ret

compar_inf:           ; compara DE con HL (en 16 bits). Devuelve
A=1 si menor DE
    MOV A,D
    CMP H
    JM menor
    JZ comp_menor
    MVI A, 0
    ret
menor:
    MVI A,1
    ret
comp_menor:
    MOV A,E
    CMP L
    JM menor
    MVI A, 0
    ret

compar_sup:           ; compara DE con HL (en 16 bits). Devuelve
A=1 si mayor DE
    MOV A,H
    CMP D
    JM menor
    JZ comp_menor2
    MVI A, 0
    ret
menor2:
    MVI A,1
    ret
comp_menor2:
    MOV A,L
    CMP E
    JM menor2
    MVI A, 0
    ret

```

```

clear_all:
    LXI H, memVideo           ;cargamos origen memoria video
    LXI D, memVideo+sizeVideo ;cargamos fin memoria video
repite_c:
    MVI A,0
    MOV M,A                  ;borrar punto de memoria
    INX H                     ;incrementar direccion
    call comparador
    cpi 1
    jz fin_clear
    jmp repite_c
fin_clear:
    ret

moverse:                      ;en registro A el movimiento
    CPI tecla_up
    cz haz_arriba
    CPI tecla_down
    cz haz_abajo
    CPI tecla_left
    cz haz_izqda
    CPI tecla_right
    cz haz_decha
    ret

comprobador:                 ;comprueba que no se excede de la memoria
de video

    LXI D, memVideo
    call compar_sup
    cpi 1                  ;Si es 0 es q DE no es mayor que HL
    jz fin
    LXI D, memVideo+sizeVideo
    call compar_inf
    cpi 1                  ;Si es 0 es que DE no es menor que HL
    jz fin
    ret

haz_arriba:                   ;moverse arriba
    LXI D, up
    call pon_punto
    ret

haz_abajo:                    ;moverse abajo
    LXI D, down
    call pon_punto
    ret

haz_izqda:                   ;moverse izda
    LXI D, left
    call pon_punto
    ret

haz_decha:                   ;moverse decha
    LXI D, right
    call pon_punto
    ret

pon_punto:

```

```

DAD d
call comprobador
mov A,M
cpi FFh
jz fin
cpi FFh/2
CZ pon_comida
MVI M, FFh           ;pintar
call comp_borra
ret

comp_borra:
push psw
LDA cuento
cpi 0
jz borra_punto      ; si el cuento es 0 hay que borrar
DCR a
STA cuento          ; decrementamos y almacenamos el cuento
pop psw
ret

borra_punto:
push h
push d
lhld cola
LXI d, up           ; mirar arriba
DAD d
mov a,M
cpi FFh             ;si hay punto hay que borrarlo
jz elimina
lhld cola
LXI d, down         ; mirar abajo
DAD d
mov a,M
cpi FFh             ;si hay punto hay que borrarlo
jz elimina
lhld cola
LXI d, left          ; mirar izquierda
dad d
mov a,M
cpi FFh             ;si hay punto hay que borrarlo
jz elimina
lhld cola
LXI d, right         ; mirar izquierda
DAD d

elimina:
MVI a, 00h
mov M,a
SHLD cola           ;almacenamos la nueva cola
pop d
pop h
pop psw
ret                 ;regresa al call de comp_borra

pon_comida:
push psw
push d
LDA cuento
adi 10h
STA cuento

```

```

LDA pos          ;cargamos la posicion actual
rlc
Mov e,a
rrc
Mvi d,0
INR a          ;incrementamos el desplazamiento
ANI 11b         ;impedimos que sea mayor que 4
STA pos          ;guardamos la posicion actual

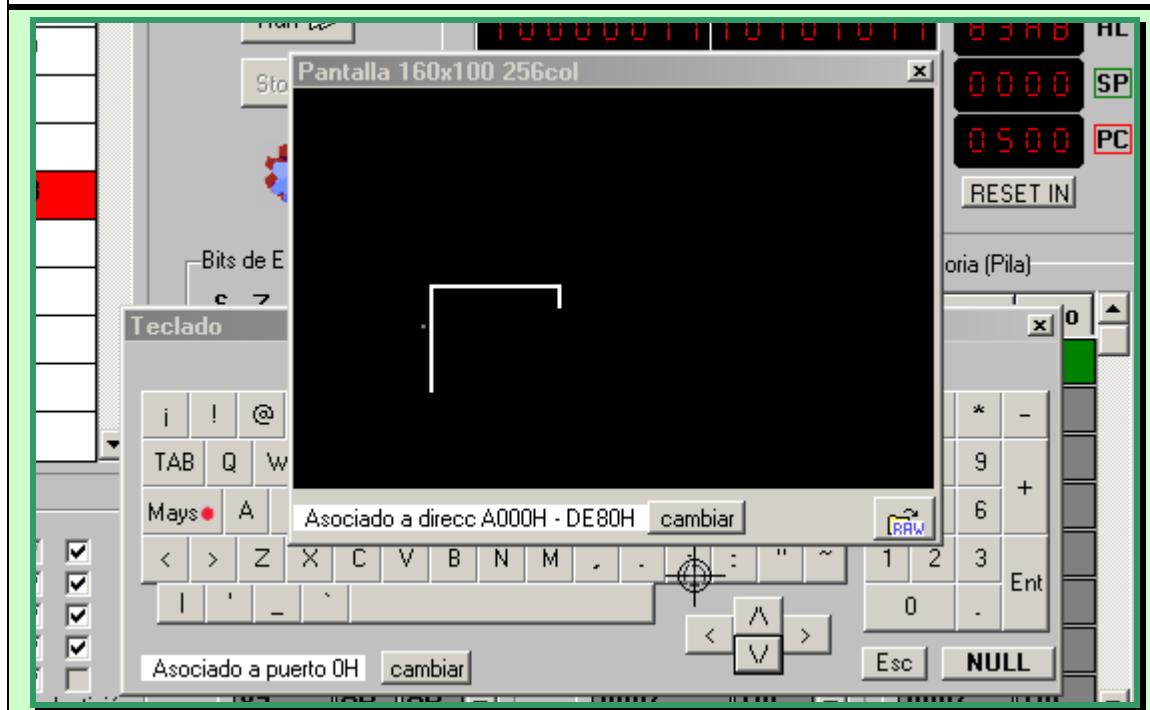
push h
LXI h, pos_pantalla
DAD d
Mov E,M
INX h
Mov D,M
XCHG           ;ya tenemos la direccion en HL
mvi a, FFh/2
mov M, a         ;coloreamos el punto

pop h           ;recuperamos lo guardado
pop d
pop psw
ret

fin:
hlt

.org 3Ch        ;Interrupcion del timer (RST 7.5)
call moverse
EI
Ret

```



Simulador del 8085

Manual de usuario

4.1. Requerimientos e instalación del programa

Lee detenidamente este apartado del manual para conocer qué características debe tener su equipo y los pasos a seguir para instalar con éxito el simulador.

4.1.1. Requerimientos

Configuración mínima recomendada:

- Windows 95/98.
- PC con procesador a 166 MHz.
- 32 MB de RAM.
- Lector de CD-ROM de 8x o superior.
- 10 MB de espacio libre en su disco duro.

4.1.2. Instalación del simulador en su equipo

El formato de distribución de Simulador 8085 es el CD, soporte que permite almacenar la versión completa de nuestro simulador.

Conociendo ya las características técnicas que debe tener su máquina, vamos a comenzar la instalación de Simulador 8085 en nuestro sistema. Para ello bastará con insertar el CD-ROM en su lector, lo que causará la autoejecución del programa de instalación. En caso de que tenga desactivada esta característica de Windows tendrá que ejecutar el programa Instalar.exe que se encuentra en el directorio raíz del CD, lo que

puede hacer cómodamente desde el Explorador de Windows o bien pulsando sobre el icono “Mi PC” y abriendo la carpeta correspondiente al lector de CD-ROM.

Realizado el paso anterior, aparecerá en pantalla una ventana. En la figura 4.1 puede ver el aspecto que muestra la ventana de instalación del simulador.

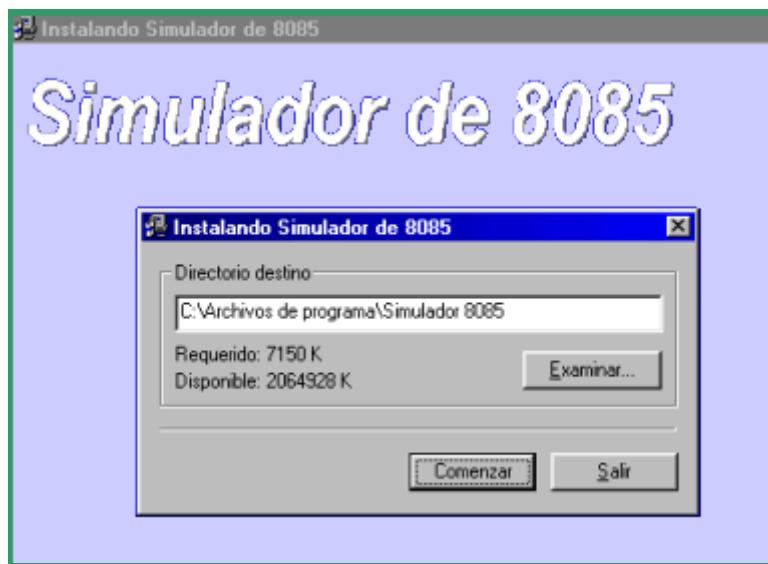


Figura 4.1. Ventana inicial del programa de instalación.

Pulsando en botón Examinar que aparece en la figura puede escoger la ubicación de los ficheros que forman el programa. Por defecto se utiliza el directorio Archivos de Programa\Simulador 8085 de la unidad C de su computador. Debajo de la ubicación se indica el espacio requerido por el programa y el espacio disponible en su disco duro.

Para salir de la instalación de Simulador 8085 pulse el botón Salir que aparece en la parte inferior derecha de la ventana.

Pulse el botón Comenzar para emprender el proceso de instalación. Se inicia entonces la copia de ficheros en su equipo. Tal y como muestra la figura 4.2, el programa de instalación irá indicando en todo momento el curso del proceso. La duración del trabajo de instalación dependerá principalmente de la velocidad de su lector de CD-ROM.

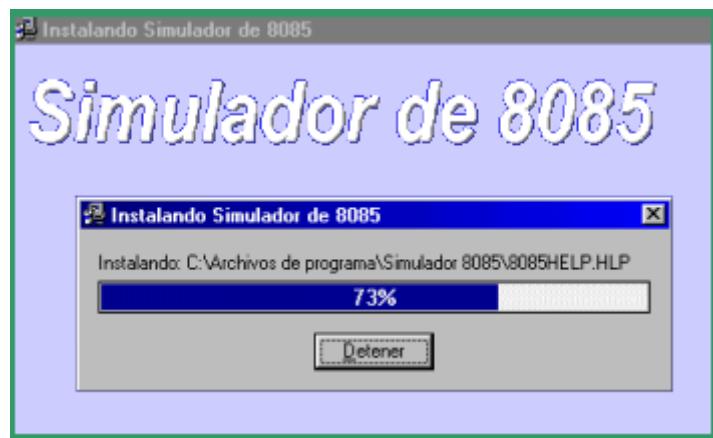


Figura 4.2. Proceso de copia de archivos al camino de destino.

Cuando la instalación haya llegado a su fin verá aparecer en pantalla una nueva ventana con información de última hora. Aparece también una casilla marcada indicando que, una vez terminada la instalación, el Simulador 8085 se ejecutará. Si desea que no se inicie el simulador, desactive la casilla.

Una última pulsación del botón Aceptar le llevará al término de la instalación, quedando la carpeta de Simulador 8085 tal y como se muestra en la figura 4.3. Ya puede ejecutar Simulador 8085, simplemente haciendo doble click sobre el ícono correspondiente.



Figura 4.3. Carpeta de archivos de Simulador 8085.

4.2. Empezar con el simulador de 8085

En este apartado aprenderá a entrar y salir del simulador de 8085 y a identificar las partes del mismo que hay en la pantalla.

Iniciar Simulador 8085 para Windows

Inicie el simulador desde el menú Inicio de Windows. Siga estos pasos:

1. Abra el menú Inicio pulsando el botón Inicio.
2. En el menú Inicio, pulse Programas.
3. En el siguiente menú, pulse Simulador 8085.
4. Por último, pulse Simulador 8085.



¿No puede encontrar Simulador 8085 en el menú?

Debe instalar el simulador en su sistema antes de poder usarlo. Por favor, consulte el manual de instalación para ver más instrucciones en el apartado 4.1.2 de este capítulo.

Si intenta iniciar el simulador de 8085 habiendo una copia del programa ejecutándose en ese momento, se mostrará en pantalla un pequeño mensaje informativo, como el de la figura 4.1.

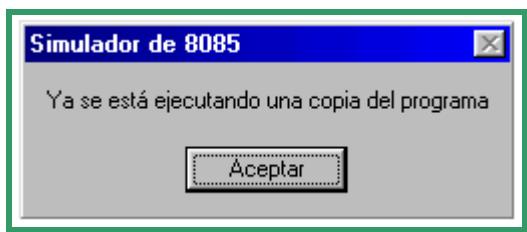


Figura 4.4. Mensaje de advertencia.



Velocidad de su ordenador...

Si la máquina en la que ejecuta el simulador no es demasiado potente, deberá esperar unos momentos a que se cargue el programa. Mientras se produce la carga, aparece en pantalla una imagen de presentación.

Entender la pantalla de Simulador 8085

Cuando inicie el simulador, verá una pantalla con varios componentes que se pueden modificar. Sin embargo, antes de comenzar necesita conocer las diferentes partes de la pantalla (véase Figura 4.2). Usará estos elementos que están descritos en la Tabla 4.1. al trabajar con sus programas.

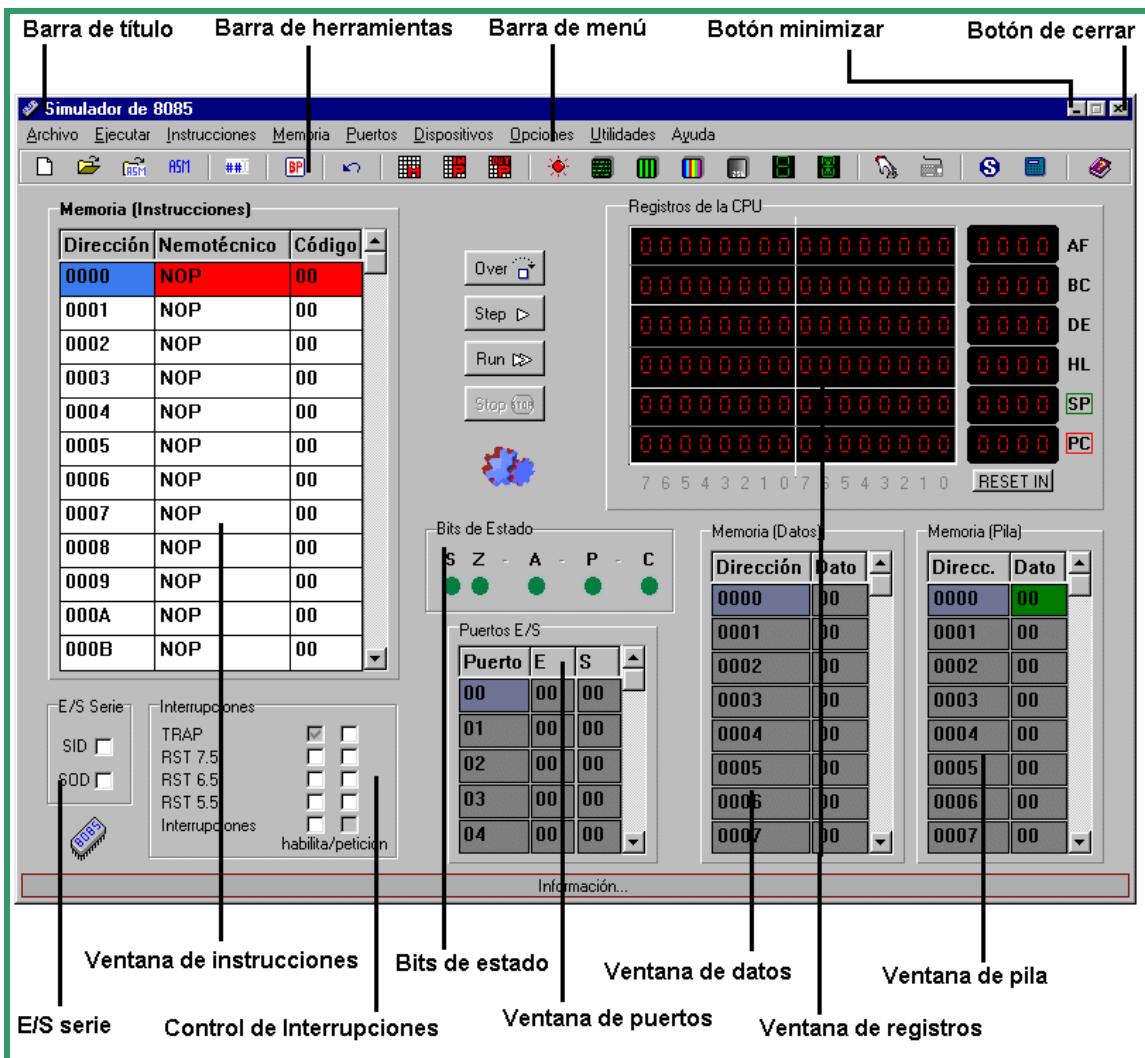


Figura 4.5. Aspecto inicial del simulador.

Tabla 4.1. Elementos de la pantalla de Simulador 8085.

Elemento de la pantalla	Función
Barra de título	El nombre del programa en el que se está trabajando. En

Tabla 4.1. Elementos de la pantalla de Simulador 8085.

Elemento de la pantalla	Función
Barra de menú	el extremo derecho de la barra de título están los botones de minimizar, restaurar y cerrar el programa
Barra de herramientas	Los encabezamientos de esta barra le permiten acceder a las órdenes de menú de Simulador 8085.
Barra de estado	Los pequeños dibujos o botones de la barra de herramientas le permiten seleccionar las órdenes que necesita más a menudo con sólo pulsar el ratón.
Barras de desplazamiento	El simulador visualiza la información sobre el estado del programa en la barra de estado.
Botón minimizar	Pulse las barras de desplazamiento para desplazarse por los componentes de la pantalla.
Botón Cerrar	Pulse este botón para ocultar el simulador temporalmente. Después pulse el botón de Simulador 8085 sobre la barra de tarea de su pantalla para volver al simulador.
Botón Restaurar/Maximizar	Pulse este botón para cerrar el programa.
Memoria de Instrucciones	Este botón está inhabilitado.
Memoria de Datos	En esta zona de la ventana se cargarán los programas a simular. También podrá editar y cambiar los datos contenidos en la lista.
Memoria de Pila	Consulte esta lista para ver los datos definidos en su programa.
Puertos de E/S	Con esta lista podrá conocer el estado de la pila durante la ejecución de un programa.
Registros de la CPU	El procesador 8085 tiene 256 puertos de entrada y 256 puertos de salida.
Bits de estado	El estado de todos los registros internos de microprocesador 8085 se pueden visualizar en esta zona de la pantalla.
	Mediante el encendido o apagado de unos leds podrá conocer la situación de los indicadores de estado.

Tabla 4.1. Elementos de la pantalla de Simulador 8085.

Elemento de la pantalla	Función
Panel de Interrupciones	Pulse en las casillas correspondientes para habilitar o deshabilitar ciertas interrupciones.
E/S Serie	Se permite la entrada o salida de un bit a través del puerto serie del microporcesador.
Control de ejecución	Está formado por un conjunto de botones mediante los que se podrá controlar la ejecución de un programa cargado en memoria.

Usar menús y barras de herramientas

Mientras esté trabajando con Simulador 8085, le dará órdenes para comunicarle las acciones que quiere llevar a cabo. Puede realizar la mayoría de las órdenes de Simulador 8085 usando los menús o la barra de herramientas. El método que escoja dependerá únicamente de sus preferencias personales.

Para seleccionar una orden de menú:

1. Abra un menú pulsando el título de menú sobre la barra del menú. También puede abrir un menú pulsando la tecla Alt al mismo tiempo que mantiene pulsada la letra subrayada del título de menú. Por ejemplo, pulse Alt + A (pulse Alt y mantenga presionada A) para abrir el menú Archivo.
2. Sobre el menú abierto, pulse la orden deseada o pulse la letra subrayada del nombre de la orden.

A lo largo de este capítulo, se utilizará una abreviatura para especificar las órdenes del menú. Por ejemplo, se indica pulse Archivo, Salir, significa abrir el menú Archivo para seleccionar después la opción Salir.



¿Cambia de idea?

Si cambia de idea sobre la opción del menú, pulse la tecla Esc dos veces o pulse cualquier sitio fuera del menú para cerrarlo si hacer ninguna selección.

La figura 4.3 muestra el menú Dispositivos abierto. Hay numerosos elementos que el simulador usa en sus menús para darle información adicional. La tabla 4.2 explica estos elementos.



Figura 4.6. Abriendo un menú del simulador.

Tabla 4.2. Partes de un menú.

Elementos de menú	Funciones
Botón	Si a la opción menú le corresponde un botón en la barra de herramientas, este botón está visualizado junto a la opción del menú.
Flecha de submenú	Indica que la opción menú conduce a otro menú (llamado submenú).
Selección con teclado	Identifica qué teclas puede usar para seleccionar la opción del menú usando el teclado.

Puede emplear una combinación de teclas para seleccionar algunas órdenes sin utilizar para nada los menús. Las selecciones con combinación de teclas aparecen en el menú junto al nombre de la opción correspondiente. En la Figura 4.3, por ejemplo, puede ver que la selección con combinación de teclas para la opción Teclado es Ctrl+T. Esto significa que pulsando Ctrl+T (pulsar sin soltar la tecla Ctrl, la tecla T y después soltar ambas teclas) tiene el mismo efecto que pulsar en Dispositivos, Teclado.

Para usar la barra de herramientas, utilice simplemente el ratón para pulsar el botón deseado. Los botones contienen dibujos que ayudan a identificar las funciones de cada uno de ellos. Puede refrescar su memoria dejando el cursor del ratón sobre un

botón durante unos pocos segundos sin pulsarlo. El simulador mostrará un pequeño rótulo junto al botón que identifica su función.



¿Necesita ayuda?

Mantenga durante unos pocos segundos el cursor del ratón sobre el botón deseado. Aparecerá un pequeño rótulo con ayuda.

Trabajar en los cuadros de diálogo

Muchas opciones del simulador se sirven de un *cuadro de diálogo*. Simulador 8085 utiliza los cuadros de diálogo para obtener la información adicional requerida para realizar una orden. Cada cuadro de diálogo es diferente, pero todos ellos tienen los mismos elementos básicos.

En un cuadro de diálogo pulse la tecla Tab para desplazarse de una opción a otra; pulse Mayús + Tab para volver atrás. Puede pulsar sobre una opción o pulsar Alt más la letra subrayada para seleccionar una opción. Cuando las selecciones del cuadro de diálogo estén cumplimentadas, pulse Intro o pulse el botón Aceptar para dar por buena la opción. Pulse el botón Cancelar o pulse Esc para cerrar el cuadro de diálogo sin hacer ninguna selección.

Salir del simulador

Cuando termine de trabajar con el simulador, hay varias opciones para salir del programa. Todos estos métodos tienen el mismo resultado:

1. Pulse Archivo, Salir.
2. Pulse Alt + F4.
3. Pulse el botón de Cerrar en la barra de título.

4.3. Conocer las partes del simulador

En el apartado 4.1 ha aprendido a iniciar y salir del simulador, a usar los menús y las barras de herramientas y también ha adquirido pequeños conocimientos de las partes constituyentes del programa.

Es el momento ahora de conocer, con más detalle, cada uno de los componentes que forman la pantalla principal del simulador de 8085.

Registros de la CPU

Le recordamos que el microprocesador 8085 cuenta con varios registros internos, los registros B, C, D, E, H y L, de 8 bits cada uno, el registro A (que actúa de acumulador) y el registro F (donde se encuentran los bits de estado), también de 8 bits, y los registros SP (puntero de pila) y PC (contador de programa), de 16 bits.

La figura 4.4 muestra la parte del simulador que incluye los registros antes mencionados. Como puede ver, se facilita el valor de cada registro tanto en codificación binaria como en hexadecimal.

Los registros AF, BC, DE y HL, se presentan por parejas, ya que como el lector sabe, muchas de las instrucciones del 8085 usan estas agrupaciones de registros.

Registros de la CPU		
00000000	00000000	0000 AF
01000011	00000000	4300 BC
00000000	00000000	0000 DE
00000000	00000000	0000 HL
00000000	00000010	0002 SP
00000000	00000011	0003 PC
7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	RESET IN

Figura 4.7. Registros de la CPU.

La mayoría de las instrucciones alteran alguno/s de los registros en su ejecución. Estos cambios que se producen se reflejan en la ventana de la figura 4.4. Sin embargo, usted también puede cambiar el contenido de los registros en cualquier momento siguiendo estos pasos:

1. Sitúe el ratón sobre el dígito binario o hexadecimal que desee cambiar. Al realizar esta acción, verá que el cursor del ratón cambia para convertirse en una pequeña mano apuntando con el dedo índice.
2. Pulse el botón izquierdo del ratón.

Si lo que cambia es un dígito binario, la pulsación del ratón hará que el bit cambie a cero o a uno, dependiendo de su valor inicial.

Si lo que cambia es un dígito hexadecimal, el hecho de pulsar el ratón hace que el dígito parpadee, tal y como muestra la figura 4.5. El parpadeo le indica que el dígito está seleccionado y que puede ser cambiado a su nuevo valor. Ahora puede teclear el valor para el dígito.



Figura 4.8. Cambiando un registro.



¿Qué valor puede introducir?

Recuerde que un dato hexadecimal puede contener 16 posibles valores, del 0 al 9, A, B, C, D, E o F. Si intenta introducir otro valor, la pulsación no tendrá efecto.

Se incorporan tres funciones más, que pueden serle útiles:

- ➊ Una vez que un dígito parpadee puede moverse por el resto de registros utilizando los cursores del teclado.
- ➋ Si modifica un dígito hexadecimal, el parpadeo se transmitirá al siguiente dígito. Aproveche esta propiedad para agilizar sus modificaciones.
- ➌ El botón RESET IN que aparece en la parte inferior de los registros permite ponerlos todos a cero.

Como es lógico, los valores binarios y hexadecimales están asociados, es decir, la modificación de uno de ellos provoca el cambio de los otros. Además el registro F es un caso particular ya que, como veremos más adelante, un cambio en él produce la alteración de los bits de estado.

Bits de estado

Una parte de la ventana principal del simulador está reservada para los indicadores o bits de estado. Como ilustra la figura 4.6, existen cinco bits, signo, cero, acarreo auxiliar, paridad y acarreo.

Aparecen en forma de pequeños leds. Si un led está encendido indica que el bit está activo y si está apagado, el bit estará inactivo. Así, en la figura 4.6, los bits de signo, acarreo auxiliar y acarreo están activados.

Situando el puntero del ratón sobre los leds es posible cambiar su estado, de encendido a apagado, y viceversa.



Figura 4.9. Bits de estado.

Como dijimos en el apartado anterior, los bits están recogidos en el registro F. Por tanto cualquier modificación de éste último implica la modificación de los leds.

Puertos de Entrada y de Salida

El procesador 8085 cuenta con 256 puertos de entrada y 256 puertos de salida de 8 bits cada uno (2 dígitos hexadecimales). En la figura 4.7 puede ver el aspecto de la zona que contiene la información de los puertos en el simulador. Todos los puertos aparecen en

una misma lista, en la que en la columna izquierda se indica el número de puerto correspondiente.

Puerto	E	S
00	44	00
01	4F	00
02	00	FF
03	00	00
04	4E	00

Figura 4.10. Puertos de E/S.



¿Cómo moverse?

Utilice la barra de desplazamiento vertical que se muestra en la figura 4.7 para localizar el puerto deseado, ya sea de entrada o salida.

Para modificar el valor de un puerto, tanto de entrada como de salida, realice los pasos siguientes:

1. Localice el puerto deseado. Para ello, haga uso de la barra de desplazamiento o los cursores.
2. Con la ayuda del ratón pulse sobre la casilla que contiene el dato. De esta manera la casilla se iluminará con un color ■.
3. Vuelva a pulsar con el ratón o pulse la tecla Intro. Modifique ahora el dato y pulse Intro. Para introducir un valor puede utilizar notación hexadecimal (sufijo H), decimal (sin sufijo o sufijo D), octal (sufijo O o Q) o binaria (sufijo B).
4. Comprobará que el valor del puerto se ha cambiado.

Recuerde también que:

- ① Puede utilizar los cursores y las teclas Re Pág y Av Pág para moverse a través de la lista de puertos.

- ① Después de modificar el dato y pulsar la tecla Intro, la casilla siguiente, correspondiente al próximo puerto, se iluminará, facilitando así la labor de modificación de varios puertos a la vez.

Por último, si pulsa con el botón derecho sobre la cabecera de la ventana de puertos, aparece un menú emergente, como el de la figura, con una opción:

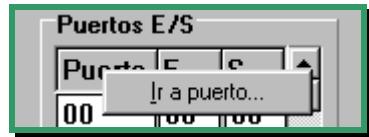


Figura 4.11. Menú emergente para puertos.

- Ir a puerto...: Aparece la ventana de la figura 4.9 en la que puede introducir un número de puerto (entre 00h y FFh).



Figura 4.12. Ir a un puerto.

Una vez introduzca el valor y pulse el botón Aceptar, la primera posición que aparece en la ventana de puertos es la que corresponde al valor introducido.

Memoria de instrucciones

El microprocesador 8085 dispone de una memoria de 65536 bytes. En esta memoria se cargan las instrucciones de los programas que usted escribe, algunas de estas instrucciones ocupan un byte, otras dos bytes, y otras tres bytes. La figura 4.10 muestra

el aspecto del componente que alberga las instrucciones en la pantalla de nuestro simulador.

Dirección	Nemotécnico	Código
0000	NOP	00
0001	NOP	00
0002	NOP	00
0003	NOP	00
0004	NOP	00
0005	NOP	00
0006	NOP	00
0007	NOP	00
0008	NOP	00
0009	NOP	00
000A	NOP	00
000B	NOP	00

Figura 4.13. Memoria de instrucciones.

En la figura aparecen tres columnas, esto es:

- **Dirección:** indica la posición de memoria. Aparece en hexadecimal y puede ir entre 0000h y FFFFh.
- **Nemotécnico:** muestra el nemotécnico de la instrucción asociada a la dirección de memoria.
- **Código:** contiene el código de la instrucción correspondiente. Aparece en hexadecimal y puede ir entre 00h y FFh.

También se puede apreciar que dos de las filas están iluminadas con distinto color que las demás:

- Este color señala la posición del contador de programa, es decir, la instrucción que se está ejecutando o que se va a ejecutar.

- Este otro color no tiene un significado especial, tan solo indica una selección. Es la posición de memoria que puede ser modificada o editada por usted.

El simulador muestra simultáneamente 12 posiciones de las 65536 posibles. La Tabla 4.3 presenta las acciones que puede realizar para llegar a una posición de memoria.

Tabla 4.3. Moverse en la memoria de instrucciones.

Ir a una posición...	Hago esto
A la siguiente posición	Pulse ↓ o pulse sobre la flecha inferior de la barra de desplazamiento vertical.
A la anterior posición	Pulse ↑ o pulse sobre la flecha superior de la barra de desplazamiento vertical.
Mostrar las 12 siguientes posiciones	Pulse Av Pág o pulse sobre la zona de desplazamiento de la barra.
Mostrar las 12 anteriores posiciones	Pulse Re Pág o pulse sobre la zona de desplazamiento de la barra.
A cualquier posición	Utilice los cursores o pinche con el ratón sobre el recuadro de la barra de desplazamiento.

Durante una simulación podrá modificar tanto el nemotécnico como el código de una instrucción. En cualquiera de los casos realice estos pasos:

1. Sitúese en la posición de memoria que desea modificar. Para ello utilice la barra de desplazamiento vertical, los cursores o el ratón.
Una vez esté en la posición adecuada verá que se ilumina con el color ■.
2. El siguiente paso es posicionarse en el lugar correcto. Utilice los cursores del teclado o pulse con el ratón sobre la casilla Nemotécnico o sobre la casilla Código dependiendo de lo que quiera modificar. La casilla seleccionada tomará el color ■.
3. Ahora pulse la tecla Intro o pulse de nuevo con el ratón sobre la casilla.

5. Modifique el valor del campo y pulse la tecla Intro. Para introducir un valor puede utilizar notación hexadecimal (sufijo H), decimal (sin sufijo o sufijo D), octal (sufijo O o Q) o binaria (sufijo B).
4. Comprobará que el valor de la casilla ha cambiado al nuevo valor.

Recuerde también que:

- ① Después de modificar una casilla y pulsar la tecla Intro, la casilla siguiente, correspondiente a la próxima posición de memoria, se iluminará, facilitando así la labor de modificación.

En esta zona de la pantalla, dispone de varios menús emergentes. Pulse con el botón derecho del ratón sobre cualquier posición de memoria y se mostrará un menú como el de la figura 4.4.

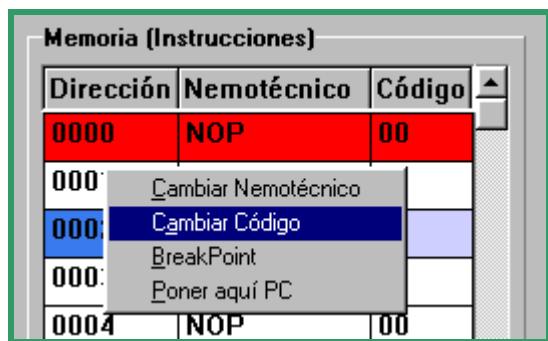


Figura 4.14. Menú emergente para las instrucciones.

Con este menú podrá realizar tareas ya comentadas en este apartado, como son los cambios del nemotécnico o el código de una instrucción, o nuevas tareas, como la posibilidad de insertar un punto de ruptura en una posición de memoria o situar el contador de programa en la posición seleccionada.



¿Qué es un “BreakPoint”?

Es un punto de ruptura, permite detener la ejecución del programa cuando el contador de programa llega a su altura.

Cuando inserte un punto de ruptura, junto al número que indica la posición de memoria, se insertará [BP].

0002	NOP	00		0002	NOP	00
0003	MOV B,E	43	→	0003 (BP)	MOV B,E	43
0004	NOP	00		0004	NOP	00

Figura 4.15. Insertando un punto de ruptura.

Del mismo modo, pulsando con el botón derecho del ratón sobre la cabecera de la ventana de instrucciones aparece un nuevo menú emergente, como el que muestra la figura siguiente, con 3 opciones:

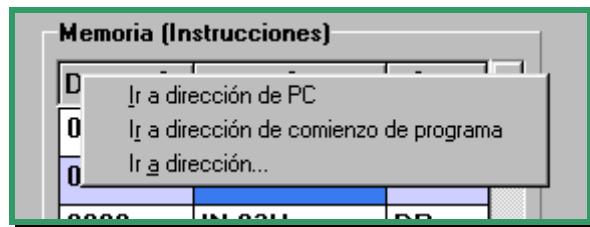


Figura 4.16. Otro menú emergente.

- Ir a dirección de PC: Al pulsar en esta opción, la primera posición de la memoria de instrucciones muestra la posición en la que se encuentra situado el contador de programa, es decir, la posición de memoria de color ■.
- Ir a comienzo de programa: Al pulsar en esta opción, la primera posición de la memoria de instrucciones muestra la posición que indica la primera directiva .ORG de comienzo de programa.
- Ir a dirección...: Aparece la ventana de la figura 4.14 en la que puede introducir una dirección de la memoria de instrucciones (entre 0000h y FFFFh).

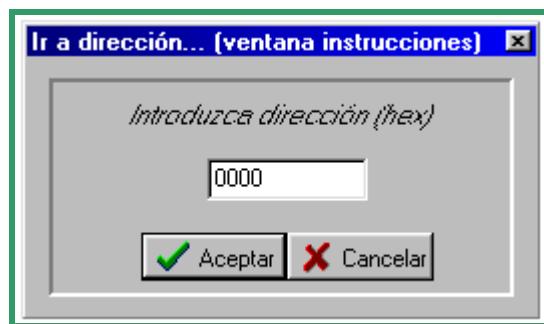


Figura 4.17. Cuadro para ir a una dirección.

Una vez introduzca el valor y pulse el botón Aceptar, la primera posición que aparece en la ventana de instrucciones es la que corresponde al valor introducido.

- *Situar PC en comienzo de programa:* Se sitúa el PC en la posición que indica la primera directiva .ORG de comienzo de programa.

Por último, en lo referente a la labor de programación, debemos aclarar varias cosas. En primer lugar y muy importante, usted será responsable de los cambios que realice, es fundamental que conozca las características de las instrucciones del 8085.

Como hemos dicho existen instrucciones de distintos tamaños y, por tanto, algunas de ellas ocupan más de una posición de memoria, no superando los 3 bytes. Entonces surge la pregunta,

¿Qué pasa con el segundo y tercer byte de estas instrucciones?

La figura 4.15 muestra lo que hace el simulador al introducir la instrucción JMP 0003h, que ocupa tres posiciones de memoria, en la posición 0000h.

Memoria (Instrucciones)		
Dirección	Nemotécnico	Código
0000	JMP 0003H	C3
0001		03
0002		00
0003	NOP	00
0004	NOP	00

Figura 4.18. Una instrucción de 3 bytes.

Las posiciones 0001h y 0002h se dejan vacías y, aunque intente modificar sus casillas de Nemotécnico como hemos explicado anteriormente, no podrá. Sin embargo si podrá modificar las casillas de Código reemplazando de esta forma la dirección de salto de la instrucción JMP.

Ahora supongamos que desea cambiar la instrucción JMP 0003h por una que ocupa un solo byte, como es la instrucción STC. Probablemente se pregunte qué pasará con las posiciones de memoria 0001h y 0002h que antes estaban ocupadas por la instrucción JMP, ¿Seguirán ocupadas ahora?, ¿Qué valor tendrán?, ¿Se podrán volver a utilizar?. En la figura 4.16 se puede ver la solución que toma el simulador de 8085 en este caso concreto.

Memoria (Instrucciones)		
Dirección	Nemotécnico	Código
0000	STC	37
0001	NOP	00
0002	NOP	00
0003	NOP	00
0004	NOP	00

Figura 4.19. La instrucción de 3 bytes se ha reemplazado.

Como vemos la nueva instrucción STC se ha introducido con éxito en la posición 0000h y las posiciones 0001h y 0002h, antes ocupadas, se han rellenado con sendas instrucciones NOP, pudiendo ser modificadas cuando usted lo deseé.

Ahora supongamos otro caso, como el que muestra la figura 4.17.

Memoria (Instrucciones)		
Dirección	Nemotécnico	Código
0000	NOP	00
0001	NOP	00
0002	STC	37
0003	NOP	00
0004	NOP	00

Figura 4.20. Instrucción STC en posición 0002h.

Supongamos que usted desconoce que la instrucción JMP 0003h ocupa tres posiciones de memoria y la introduce en la dirección 0000h. La figura 4.18 muestra el resultado de esta operación.

La instrucción STC que había en la posición 0002h se ha sobrescrito. Si usted no se percata de las consecuencias de su acción seguirá programando y, posteriormente, al ejecutar el programa, verá que no hace lo que usted quería.

Es un caso claro de los errores que puede llevar el desconocimiento del conjunto de instrucciones del microprocesador 8085.

Memoria (Instrucciones)		
Dirección	Nemotécnico	Código
0000	JMP 0003H	C3
0001		03
0002		00
0003	NOP	00
0004	NOP	00

Figura 4.21. La instrucción STC ha desaparecido.

Si la instrucción que había en la posición 0002h hubiera sido JMP 000Ah en lugar de STC, las posiciones 0003h y 0004h que, como usted ya sabe estarían también ocupadas, también serían reemplazadas por NOP al escribir en la posición 0000h la instrucción JMP 0003h. Para aclarar esto mire la siguiente figura.

Memoria (Instrucciones)		
Dirección	Nemotécnico	Código
0000	NOP	00
0001	NOP	00
0002	JMP 000AH	C3
0003		0A
0004		00
0005	NOP	00

Memoria (Instrucciones)		
Dirección	Nemotécnico	Código
0000	JMP 0003H	C3
0001		03
0002		00
0003	NOP	00
0004	NOP	00
0005	NOP	00

Figura 4.22. Introduciendo la instrucción JMP 0003H.

Memoria de datos y memoria de pila

Estas dos listas que aparecen en el simulador tienen una estrecha relación con la memoria de instrucciones que acabamos de ver. En realidad todas ellas contienen los mismos datos. La figura 4.20 muestra las dos memorias en un momento de simulación.

Memoria (Datos)		Memoria (Pila)	
Dirección	Dato	Direcc.	Dato
0000	DE	0000	DE
0001	05	0001	05
0002	00	0002	00
0003	43	0003	43
0004	11	0004	11
0005	AE	0005	AE
0006	00	0006	00
0007	37	0007	37

Figura 4.23. Memorias de datos y de pila.

Ambas memorias disponen de 65536 posiciones en las que aparecen, en valor hexadecimal, el contenido de cada posición de memoria.

En todo momento aparece una casilla en color ■ que, al igual que vimos en otros componentes, corresponde a la selección actual. Para cambiar cualquier posición de las memorias o situarse en una posición determinada, no hay más que seguir los pasos descritos en el caso de los puertos de entrada y salida.

En el caso de la memoria de pila siempre aparece una posición resaltada con el color ■ indicando la posición del puntero de pila. Puede probar a cambiar, en la zona de Registros de la CPU, el valor del registro SP y verá cómo la posición de la memoria de pila, correspondiente al valor del registro SP, se ilumina del color ■.

Por defecto la pila comienza en la posición de memoria 0000h.

Como ya le hemos dicho, todas las zonas de memoria del simulador comparten los mismos datos y, por tanto, cualquier modificación en una posición de memoria, involucra a todas las demás zonas de memoria.



Cambiando los datos...

Debe tener cuidado al cambiar el contenido de cualquier posición de la memoria de datos y de la memoria de pila.

Los cambios que realice tendrán efecto sobre la memoria de instrucciones pudiendo cambiar el significado de su programa.

Para terminar, le recordamos que si pulsa con el botón derecho sobre la cabecera de la memoria de datos o pila, aparece un menú emergente, como el de la figura, con 2 opciones:

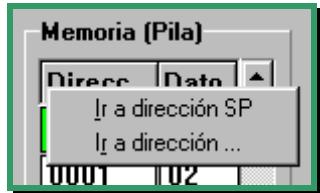


Figura 4.24. Menú emergente para datos y pila.

- Ir a dirección SP: Esta opción no es igual para la memoria de pila y para la memoria de datos.

Para la memoria de pila, al pulsar en esta opción, la primera posición de la memoria de pila muestra la posición a la que apunta el puntero de pila, es decir, la posición de color ■.

En el submenú de la memoria de datos, esta opción se llama Ir a dirección de comienzo de datos. Al pulsarla, la primera posición de la memoria de datos muestra la posición en la que se encuentra la primera directiva .DATA de comienzo de datos de su programa.

- Ir a dirección...: Tanto para el caso de la memoria de datos como para la memoria de pila, aparece un cuadro como el de la figura 4.22 en el que puede introducir una posición de memoria (entre 0000h y FFFFh).



Figura 4.25. Ir a una posición de memoria.

Una vez introduzca el valor y pulse el botón Aceptar, la primera posición que aparece en la memoria de pila (o en la memoria de datos) es la que corresponde al valor introducido.

Control de ejecución

La figura 4.23 presenta el aspecto de esta parte de la pantalla. Contiene 4 botones desde los que es posible controlar la ejecución de sus programas.



Figura 4.26. Botones para el control de una ejecución.

Cada botón realiza una tarea distinta:

- ▶ **Botón Step:** Ejecuta la siguiente instrucción. La instrucción donde se encuentra el contador de programa se ejecuta, desplazándose éste a la siguiente instrucción.
- ▶ **Botón Over:** Tiene la misma función que el botón anterior salvo que, cuando se llega a una instrucción de llamada a subrutina, el simulador ejecuta las instrucciones pertenecientes a la subrutina en modo continuo, y después deja

el contador de programa en la siguiente instrucción a la instrucción de llamada. Utilice este botón para agilizar sus comprobaciones y evitar entrar en cada una de las subrutinas.

- ▶ **Botón Run:** Ejecuta en modo continuo. Se ejecutan las instrucciones a partir de la posición donde se encuentra el contador de programa.
- **Botón Stop:** Para la ejecución en modo continuo. La ejecución se detiene, quedando el contador de programa en la posición de memoria siguiente a la última instrucción ejecutada.

Como es lógico, el simulador controla el estado de estos botones, impidiendo su pulsación cuando así se requiera. De esta forma, no se podrá pulsar el botón Stop cuando la ejecución en modo continuo no haya comenzado, o no se podrá pulsar el botón Run cuando la ejecución ya ha comenzado.

Panel de interrupciones

El microprocesador 8085 permite detener la ejecución normal de un programa mediante una serie de interrupciones. En la figura 4.24 aparecen las interrupciones que usted podrá utilizar durante una simulación.

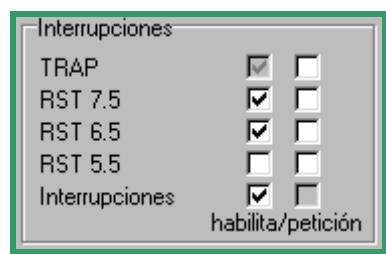


Figura 4.27. Control de interrupciones.

Los parámetros que controlan las interrupciones se han representado por casillas que pueden tomar dos posibles valores, (activa) o (inactiva).

Como puede ver en la figura existen 4 tipos de interrupciones, TRAP, RST 7.5, RST 6.5 y RST 5.5. Además existe un elemento más, llamado INTR, el cual le indica si las interrupciones están permitidas o no.

Cada interrupción tiene asociadas dos casillas que ahora pasamos a comentar.

La primera casilla le indica si una interrupción está permitida o no. Verá que la interrupción TRAP siempre tiene activa esta casilla, es decir, esta interrupción siempre podrá ser utilizada.

Por lo que respecta a las otras interrupciones, su activación no sólo depende de esta primera casilla sino también del valor que tome la casilla correspondiente al parámetro INTR. De esta forma, en nuestro ejemplo de la figura 4.24, las interrupciones RST 7.5 y RST 6.5 están permitidas mientras que la interrupción RST 5.5 no lo está.

Siempre que desactive la casilla INTR se inhabilitarán para la simulación las interrupciones RST 7.5, RST 6.5 y RST 5.5, aunque su casilla siga activada. En el ejemplo anterior, si pulsamos en la casilla INTR, no podremos llamar a ninguna interrupción RST x.5. Pero si más adelante volvemos a pulsar en la casilla INTR, tendremos nuevamente habilitadas RST 7.5 y RST 6.5, porque sus casillas estaban activas

La segunda casilla le permite realizar las peticiones de interrupción. En el momento que quiera realizar una petición no tiene más que activar la casilla correspondiente.

Ahora bien, la activación de una petición solamente tendrá efecto en dos casos:

- ① Cuando la casilla que active la petición corresponde a la interrupción TRAP.
Como sabe esta interrupción siempre está permitida.
- ② Cuando la casilla que active la petición corresponde a una interrupción que tiene activada su otra casilla, la casilla que da permiso de interrupción.

En cualquier otro caso, la activación de la segunda casilla no tendrá ningún efecto sobre la ejecución del programa.

Una vez que una petición de interrupción ha sido aceptada, una rutina de interrupción la atenderá tomando momentáneamente el control del programa (véase el conjunto de instrucciones de un capítulo anterior para conocer más acerca de las interrupciones en el microprocesador 8085).

Entrada / Salida serie

Nos queda por ver una pequeña parte de la ventana de nuestro simulador. En la esquina inferior izquierda aparece un pequeño panel con dos casillas, tal y como muestra la figura 4.25, que le permitirá controlar la entrada y salida serie.



Figura 4.28. Casillas para la E/S serie.

Para entender el significado de cada casilla debemos hacer referencia a dos instrucciones que incorpora el 8085:

- Casilla SID: Si está activa, cuando se ejecute la instrucción RIM, se cargará en el bit 7 (último bit) del acumulador el valor 1. En caso contrario, ese bit se carga con el valor 0.
- Casilla SOD: Cuando se ejecute la instrucción SIM, si el bit en la posición 6 del acumulador está activado, la casilla SOD tomará el valor del bit 7 del acumulador. En caso contrario la casilla permanecerá inalterada.

4.4. Conociendo los menús del simulador

Hasta ahora solamente hemos visto algunas operaciones que puede realizar en el panel frontal del simulador. Muchas de estas operaciones se pueden habilitar también desde los menús y no nos detendremos demasiado en volver a explicarlas.

Otras, sin embargo, deben ser realizadas expresamente desde uno de los menús que se presentan a lo largo de la parte superior del simulador. En este apartado del manual nos ocuparemos de todas estas funciones, incidiendo en las que consideramos más útiles o que son utilizadas con más frecuencia.

Explicaremos los menús por orden, es decir, comenzando por el menú Archivo, en la parte izquierda de su pantalla, hasta terminar con el menú Ayuda.

4.4.1. Archivo Nuevo, cargar, salir, ...

En este primer menú podrá realizar operaciones tales como limpiar la memoria de instrucciones, cargar un programa objeto, llamar al editor de textos o salir del simulador.

Opción Nuevo.

La memoria de instrucciones se limpia introduciendo en cada posición la instrucción NOP.

Antes de proceder a la operación y destruir el contenido actual de la memoria de instrucciones, se presenta un mensaje de confirmación, tal y como el de la figura 4.26.

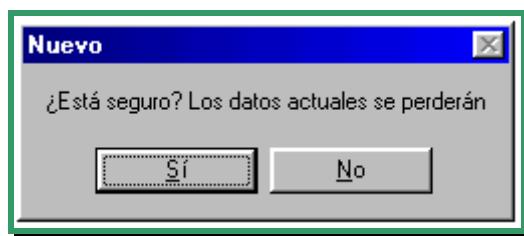


Figura 4.29. Mensaje de confirmación.

❖ Opción **Cargar OBJ.**

Permite cargar en la memoria de instrucciones programas previamente ensamblados en el editor de textos, con extensión **.O**.

Al elegir esta opción aparece un cuadro de diálogo para abrir el fichero deseado. Cuando lo haya hecho, pulse el botón Abrir. Verá que el simulador vuelve a mostrar un mensaje como el de la figura 4.26 para confirmar la operación y destruir los datos actuales.

El simulador también le pedirá permiso para destruir o no el contenido del resto de memoria que no se sobrescribe al cargar un nuevo programa objeto. De esta forma usted podrá cargar un programa en una zona de memoria sin destruir posibles programas que tuvieran en otras zonas de memoria.

❖ Opción **Cargar ASM.**

Posibilita la carga de ficheros ASM en el editor de programas que incorpora el simulador.

Al pulsar la opción, aparece un cuadro de diálogo para abrir un fichero ASM. Cuando lo haya escogido, el editor de programas se abre con el fichero que se ha seleccionado.

❖ Opción **Editor ASM.**

Se muestra el editor de textos que incorpora el simulador. Con este editor podrá escribir sus programas de 8085 y ensamblarlos.

Una vez abierto el editor, el simulador queda en segundo plano sin ocultarse por completo. Quedan en pantalla por tanto dos ventanas, el editor y el simulador. Pulse la tecla F2 de su teclado para activar una u otra ventana..

Para conocer más acerca del editor, vea el capítulo siguiente en el que se explican cada una de las opciones que incorpora.

❖ Opción **Salir.**

Salida del simulador.

4.4.2. **Ejecutar** **Paso a paso, continuo, parar, ...**

Este es el menú desde el que podrá controlar la ejecución de sus programas. Podrá simular en modo continuo, paso a paso o, incluso, ejecutar una instrucción en modo directo.

☞ **Opción Paso a paso.**

Ejecución paso a paso.

Véase botón Step en el apartado Control de ejecución visto anteriormente.

☞ **Opción Paso a paso (sin subrutinas).**

Ejecución paso a paso sin entrar en las subrutinas.

Véase botón Step Over en el apartado Control de ejecución visto anteriormente.

☞ **Opción Continuo.**

Ejecución en modo continuo.

Véase botón Run en el apartado Control de ejecución visto anteriormente.

☞ **Opción Parar.**

Parar la ejecución.

Véase botón Stop en el apartado Control de ejecución visto anteriormente.

☞ **Opción Continuo (Ignora BP).**

Tiene la misma función que la opción **Continuo** pero ignorando los puntos de ruptura que hubiera podido introducir en la memoria de instrucciones.

☞ **Opción Instrucción (Modo directo).**

Aparece una pequeña ventana como muestra la figura 4.27. Desde aquí puede ejecutar una instrucción sin necesidad de que esté en la memoria de instrucciones.

Como puede ver, por defecto se ejecuta la instrucción NOP. Usted puede introducir la instrucción que desee, siempre y cuando siga la sintaxis correcta del conjunto de instrucciones del 8085. Puede usar cualquier tipo de parámetros de las instrucciones tanto constantes como las expresiones que soporta el

ensamblador. Para más detalles, puede consultar el capítulo dedicado al lenguaje ensamblador.

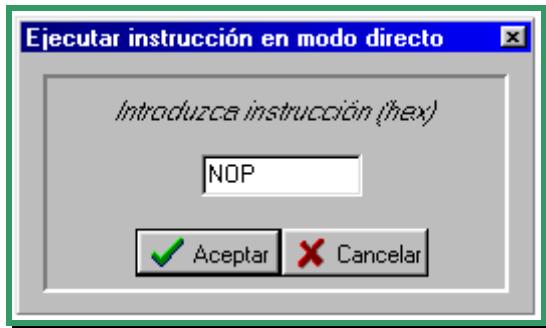


Figura 4.30. Instrucción en modo directo.

4.4.3. **Instrucciones** | **Cambiar, deshacer, ir a dirección,...**

Aquí se incluyen opciones que ya hemos visto al comentar los componentes del simulador y opciones nuevas, todas ellas relacionadas con la memoria de instrucciones.

- ❖ Opciones **Cambiar Nemotécnico**, **Cambiar Código**, **BreakPoint (Si/No)**, **Situar PC**, **Situar PC en comienzo de programa**.

Ya se vieron al describir la Memoria de Instrucciones en el apartado reservado a explicar las partes del simulador (ver apartado 4.3 de este capítulo).

- ❖ Opción **Deshacer último cambio**.

Con esta opción deshace el último cambio que realizó en la memoria de instrucciones. Sólo podrá realizar esta operación si previamente ha modificado el nemotécnico o el código de una instrucción, o se ha producido algún cambio en la memoria de instrucciones, sea cual sea el origen de este cambio.

- ❖ Opción **Eliminar todos BreakPoints**.

Todo punto de ruptura que aparezca en una instrucción será eliminado.

Puede utilizar esta opción cuando el número de puntos de ruptura que aparecen en su programa sea grande y, el eliminarlos uno a uno, lleve mucho tiempo.

⇨ Opción **Ir a dirección....**

Aparece la ventana de la figura 4.28 en la que puede introducir una dirección de la memoria de instrucciones (entre 0000h y FFFFh). Una vez introduzca el valor y pulse el botón Aceptar, la primera posición que aparece en la ventana de instrucciones es la que corresponde al valor introducido.



Figura 4.31. Ir a una dirección de la memoria de instrucciones.

⇨ Opción **Ir a dirección de PC.**

Al pulsar en esta opción, la primera posición de la memoria de instrucciones muestra la posición en la que se encuentra situado el contador de programa, es decir, la posición de memoria de color ■.

⇨ Opción **Ir a dirección de programa.**

Al pulsar en esta opción, la primera posición de la memoria de instrucciones muestra la posición en la que se encuentra la primera directiva .ORG de comienzo de programa.

4.4.4. **Memoria** **Rellenar, deshacer, datos, pila, ...**

Entre las funciones que incorpora este menú podrá llenar una porción de memoria o ir a una posición de la memoria de datos o pila.

⇨ Opción **Rellenar.**

Se muestra una ventana como la de la figura 4.29. Las posiciones de memoria incluidas entre el campo Desde y el campo Hasta se llenan con el valor que

usted introduzca. En nuestro ejemplo de la figura, la zona de memoria comprendida entre las posiciones 0000h y 00AAh (ambas inclusive) se llenaran con la instrucción NOP (valor 00h).



Figura 4.32. Rellenar memoria.

El valor que introduzca puede ir entre 00h y FFh. Existen dos casos excepcionales en los que puede introducir una cadena de caracteres:

- RAND: Las posiciones de memoria que usted indique se llenarán con valores aleatorios entre 00h y FFh.
- ORD: Las posiciones de memoria que usted indique se llenarán con valores ordenados, comenzando por el 00h.

富民 Opción Deshacer último cambio.

Con esta opción deshace el último cambio que realizó en la memoria de instrucciones. Sólo podrá realizar esta operación si previamente ha modificado el nemotécnico o el código de una instrucción, o se ha producido algún cambio en la memoria de instrucciones, sea cual sea el origen de este cambio.

Ahora veamos en detalle dos submenús que aparecen bajo las dos opciones comentadas anteriormente. Ambos submenús son muy parecidos en cuanto a las opciones que incluyen y referencian datos que comparten la memoria de datos y la memoria de pila.

Submenú Datos

Opción **Ir a dirección.**

Tiene el mismo efecto que la opción Ir a dirección... del menú Instrucciones, pero para la memoria de datos.

Opción **Ir a dirección de datos.**

Al pulsar en esta opción, la primera posición de la memoria de datos muestra la posición en la que se encuentra la primera directiva .DATA de comienzo de datos de su programa.

Submenú Pila

Opción **Ir a dirección.**

Tiene el mismo efecto que la opción Ir a dirección... del menú Instrucciones, pero para la memoria de pila.

Opción **Ir a dirección de SP.**

Al pulsar en esta opción, la primera posición de la memoria de pila muestra la posición a la que apunta el puntero de pila, es decir, la posición señalada con el color .

4.4.5. Puertos Ir a..., salida, entrada, ...

Utilice este menú para desplazarse a un puerto determinado o llenar un conjunto de puertos, ya sean de entrada o de salida.

Opción **Rellenar puertos de salida.**

Tiene el mismo efecto que la opción Rellenar del menú Memoria, pero para la lista de puertos de salida.

Opción **Rellenar puertos de entrada.**

Tiene el mismo efecto que la opción Rellenar del menú Memoria, pero para la lista de puertos de entrada.

🔗 Opción **Ir a puerto....**

Tiene el mismo efecto que la opción Ir a dirección... del menú Instrucciones, pero para la tabla de puertos.

4.4.6. **Dispositivos** LEDs, visualizadores, teclado, ...

El simulador proporciona varios dispositivos, tanto de salida como de entrada. Algunos de ellos están asociados a puertos y otros lo están a la memoria. Veamos cada uno por separado.

Dispositivos de salida

Aquí se incluyen cuatro tipos de dispositivos:

- Panel de LEDs.
- Pantalla de texto.
- Pantalla gráfica.
- Visualizadores.

Muchos de ellos son fáciles de manejar, otros sin embargo, necesitan algunas aclaraciones sobre su funcionamiento.

🔗 Opción **Panel de LEDs.**

Al elegir esta opción aparece un panel parecido al que muestra la figura 4.30.

Como sabemos un puerto almacena un dato de ocho bits y, por tanto, hemos asociado, a cada bit, un LED. De esta forma un puerto tendrá ocho LEDs asociados.

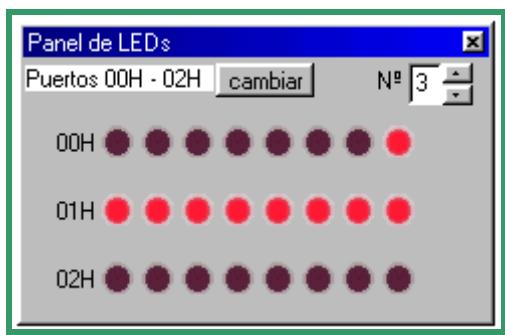


Figura 4.33. Panel de LEDs.



¿Qué es un LED?

Es un diodo emisor de luz.

En nuestro simulador se representan por pequeñas luces de color rojo.

En el ejemplo de la figura, se muestran tres puertos, del 00h al 02h, tal y como se indica en la parte superior izquierda del panel.

Puede cambiar los puertos asociados pulsando el botón Cambiar. Al pulsarlo, éste cambia y pasa a ser un campo de edición. Introduzca en el campo un valor (entre 00h y FFh) y pulse Intro. Entonces los puertos a los que se asocian los LEDs, comenzarán a partir del puerto introducido.

Entre uno y ocho puertos de salida pueden tener asociados LEDs simultáneamente. Para establecer este número utilice el campo que aparece en la esquina superior derecha del panel. En el ejemplo de la figura 4.30 este campo contiene el valor 3.

Para cambiar el estado de un LED (encendido o apagado) modifique en el simulador el valor del puerto asociado al LED. Así, en nuestro ejemplo, el puerto 00h tiene el valor 01h, el puerto 01h el valor FFh y el puerto 02h el valor 00h.

Otra posible forma de cambiar el estado de los LEDs es utilizar el contenido del acumulador y la instrucción OUT del microprocesador 8085.

☞ Opción **Pantalla de texto**.

Aparece en pantalla una ventana de texto como la de la figura 4.31 incluyendo una serie de caracteres ASCII, en nuestro ejemplo 256 caracteres.

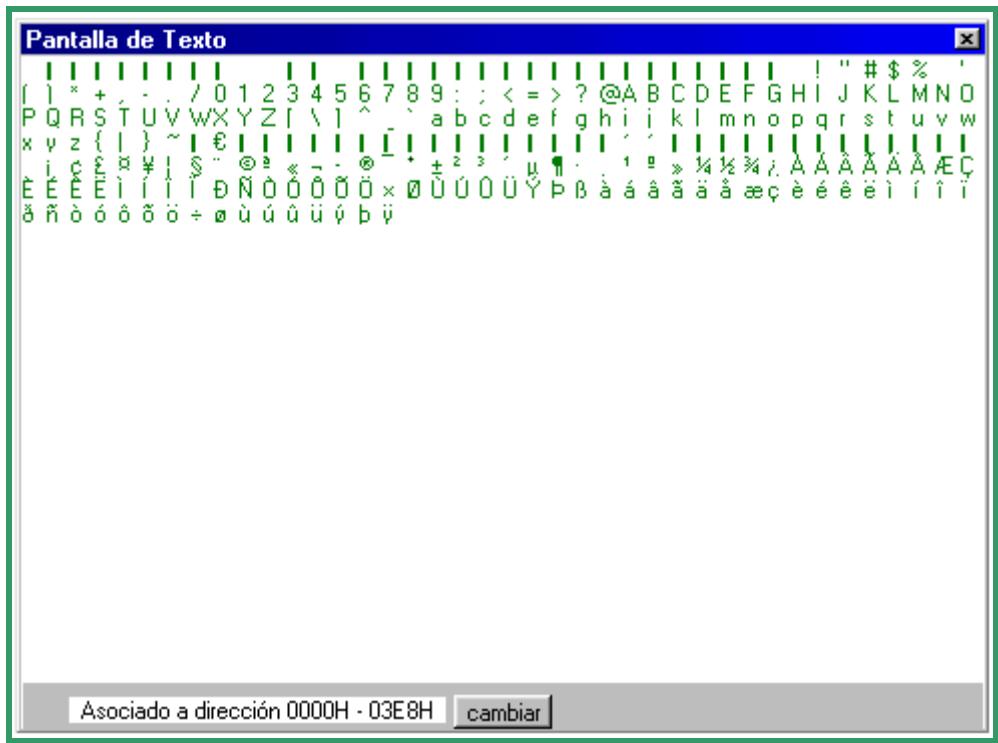


Figura 4.34. Pantalla de texto.

Como sabe, cada posición de la memoria del 8085 es de un byte (8 bits), es decir, puede representar 256 posibles valores. Hemos aprovechado esta propiedad para utilizar cada posición de memoria como un carácter ASCII, ya que este código utiliza 8 bits.



¿Qué es el código ASCII?

Los datos de tipo carácter representan elementos individuales de conjuntos finitos y ordenados de caracteres. El conjunto de caracteres representado depende del computador. Uno de los conjuntos más usuales es el ASCII (American Standard Code for Information Interchange).

Solamente es una parte de la memoria del 8085 la que está asociada a la pantalla de texto en un determinado momento, concretamente 1000 (40x25) posiciones de memoria. En el ejemplo de la figura 4.31, las posiciones van desde la 0000h

hasta la 03E8h. Siempre que modifique alguna de estas posiciones, los cambios se verán reflejados también en la pantalla de texto.

Por defecto la pantalla de texto comienza en la posición E000h y llega hasta la E3E8h. Puede asociar otras posiciones de memoria pulsando el botón Cambiar. Al pulsarlo, éste cambia y pasa a ser un campo de edición. Introduzca en el campo un valor (entre 0000h y FFFFh) y pulse Intro. Entonces las posiciones a las que se asocia la pantalla de texto, comenzarán a partir de la posición correspondiente al valor introducido.

El ejemplo de la figura anterior nos ha servido para mostrarle todo el conjunto de caracteres. Para ello hemos utilizado la opción Rellenar del menú Memoria introduciendo el valor ORD que ya le explicamos en algún apartado anterior.

Aunque parezca que solo hay 256 caracteres en la figura, en realidad se muestran 1000 caracteres, los caracteres que no se ven corresponden al carácter nulo (valor 00h).

- ① Puede consultar una tabla con los caracteres ASCII más usados en el apéndice correspondiente.

Submenú Pantalla gráfica

La forma más cómoda de adquirir información es a través de la vista, por lo que las pantallas gráficas constituyen un sistema usual de captar las salidas de un computador.

De igual forma que en el apartado anterior ha aprovechado la información contenida en la memoria del simulador, puede utilizarla también para cargar pantallas gráficas.

La pantalla gráfica del simulador no es continua, sino que es una matriz de puntos de imagen o unidades de visualización (en inglés *pixels*). Por tanto, la imagen de la pantalla se forma por medio de puntos de imagen. Para configurar una imagen se activan selectivamente distintos puntos de imagen dentro de un cuadrado determinado.

Dentro de este submenú encontrará varias opciones que dependen de dos aspectos:

- Resolución: es el tamaño de la pantalla gráfica, a mayor número de puntos de imagen, la resolución será mayor. Usted podrá elegir dos tipos de resolución, 256x200 o 160x100.
- Número de colores: es el número de posibles valores que puede tomar un punto de imagen. Si un punto de imagen ocupa un byte, será capaz de representar 256 posibles colores. Usted podrá elegir 2, 16 o 256 colores para su pantalla gráfica.

Por defecto cualquier pantalla gráfica del simulador comienza desde la posición de memoria A000h. Dependiendo del tipo de resolución y número de colores, la pantalla ocupará más o menos posiciones a partir de la A000h.

Veamos ahora detenidamente cada opción de este submenú. Según la resolución y el número de colores elegido se indica la porción de memoria que se utiliza en cada caso, así como el espacio ocupado por cada punto de imagen. También se muestra la zona de memoria que se necesita utilizar.

Opción **256x200 [2 colores]**.

Características de pantalla

Posiciones de memoria asociadas por defecto	De la A000H a la B900H
Espacio ocupado en memoria	6.400 posiciones
Tamaño de punto de imagen	1 bit
Puntos de imagen por posición de memoria	8 puntos de imagen
Número total de puntos de imagen	51.200 puntos de imagen

Al pulsar esta opción se muestra una ventana con la forma que muestra la figura 4.32, incluyendo una pantalla gráfica de 256x200 puntos imagen.

Solamente puede visualizar 2 colores, negro o verde, según el valor del bit asociado a cada punto de imagen. Si el bit está a 1, el color será verde, en caso contrario, será negro. De esta forma, una posición de memoria que contenga el valor FFh, representará 8 puntos de imagen de color verde en la pantalla gráfica.

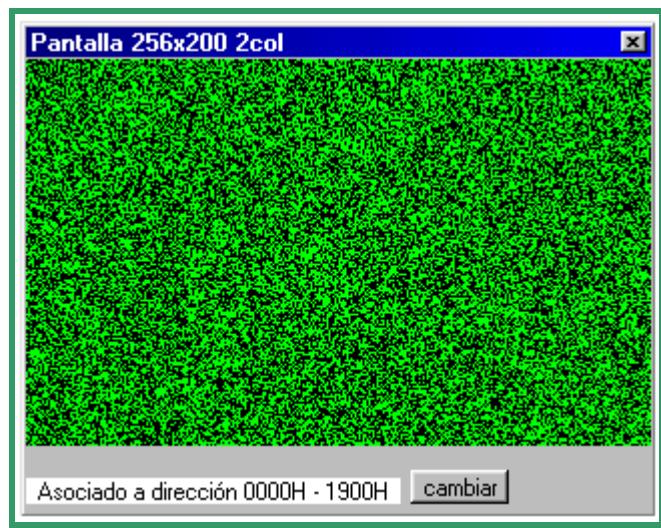


Figura 4.35. Pantalla gráfica de 2 colores.

En el ejemplo, las 6.400 posiciones de memoria que necesita la pantalla gráfica están comprendidas entre la 0000h y la 1900h, tal y como se indica en la parte inferior izquierda de la ventana. Estas posiciones se han rellenado previamente con datos aleatorios, introduciendo la palabra RAND en la opción Rellenar del menú Memoria.

Para cambiar las posiciones asociadas a la pantalla gráfica, siga los pasos que se le indicaron en la opción Pantalla de texto de este mismo menú.

⇨ Opción **160x100 [16 colores]**.

Características de pantalla

Posiciones de memoria asociadas por defecto	De la A000H a la BF40H
Espacio ocupado en memoria	8.000 posiciones
Tamaño de punto de imagen	4 bits
Puntos de imagen por posición de memoria	2 puntos de imagen
Número total de puntos de imagen	16.000 puntos de imagen

Al pulsar esta opción se muestra una ventana con la forma que muestra la figura 4.33, incluyendo una pantalla gráfica de 160x100 puntos imagen.



Figura 4.36. Pantalla gráfica de 16 colores.

En este caso se dispone de una gama de 16 colores como la que se indica en la figura 4.34. Como cada color se representa mediante 4 bits, el valor 0000b representa al negro, el 1111b al blanco y, los restantes valores, otros colores como rojo, verde, azul, etc.



Figura 4.37. Gama de 16 colores.

En el ejemplo, las 8.000 posiciones de memoria que necesita la pantalla gráfica están comprendidas entre la 0000h y la 1F40h, tal y como se indica en la parte inferior izquierda de la ventana. Estas posiciones se han rellenado previamente con datos aleatorios, introduciendo la palabra RAND en la opción Rellenar del menú Memoria.

Para cambiar las posiciones asociadas a la pantalla gráfica, siga los pasos que se le indicaron en la opción Pantalla de texto de este mismo menú.

☞ Opción **160x100 [256 colores]**.

Características de pantalla

Posiciones de memoria asociadas por defecto	De la A000H a la DE80H
Espacio ocupado en memoria	16.000 posiciones
Tamaño de punto de imagen	8 bits
Puntos de imagen por posición de memoria	1 punto de imagen
Número total de puntos de imagen	16.000 puntos de imagen

Al pulsar esta opción se muestra una ventana con la forma que muestra la figura 4.35, incluyendo una pantalla gráfica de 160x100 puntos imagen.



Figura 4.38. Pantalla gráfica de 256 colores.

En este caso se dispone de una gama de 256 niveles de gris como la que se indica en la figura 4.36. Como cada nivel se representa mediante 8 bits, el valor 00h representa al negro, el FFh al blanco y, los restantes valores, otros niveles de gris entre el blanco y el negro.

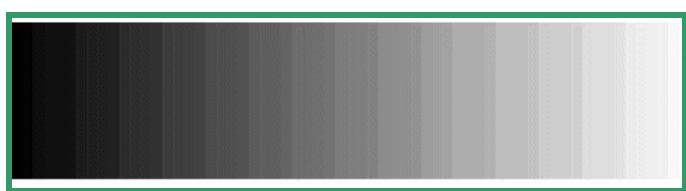


Figura 4.39. Gama de 256 niveles de gris.

En el ejemplo, las 16.000 posiciones de memoria que necesita la pantalla gráfica están comprendidas entre la 0000h y la 3E80h, tal y como se indica en la parte inferior izquierda de la ventana. Estas posiciones se han rellenado previamente con datos aleatorios, introduciendo la palabra RAND en la opción Rellenar del menú Memoria.

Para cambiar las posiciones asociadas a la pantalla gráfica, siga los pasos que se le indicaron en la opción Pantalla de texto de este mismo menú.

Submenú Visualizadores

Los visualizadores (*displays* en inglés) son pequeñas unidades de salida que permiten al usuario leer información producida por el computador.

☞ Opción **7 segmentos**.

Al pulsar en esta opción se muestran 8 visualizadores, de 7 segmentos cada uno, tales como los de la figura 4.37.



Figura 4.40. 8 visualizadores de 7 segmentos.

Siempre serán ocho los puertos que estén asociados a los visualizadores, que serán los que se indiquen en el campo de la parte inferior izquierda de la ventana.

Puede cambiar los puertos pulsando el botón Cambiar. Al pulsarlo, éste cambia y pasa a ser un campo de edición. Introduzca en el campo un valor (entre 00h y FFh) y pulse Intro. Entonces los puertos a los que se asocian los visualizadores, comenzarán a partir del puerto correspondiente al valor introducido.

Como hemos dicho, un visualizador estará asociado a un puerto. Por lo tanto, cada uno de los 8 bits que contiene el dato del puerto servirá para activar un

LED del visualizador. En la figura 4.38 se muestra la asociación que existe entre los bits del puerto y los LEDs un visualizador.

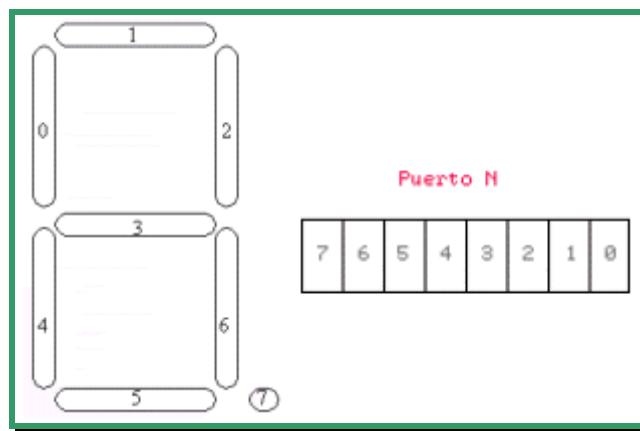


Figura 4.41. Estructura de un visualizador de 7 segmentos.

Como puede ver en la figura anterior, el último bit del puerto (bit 7), se ha utilizado para añadir un punto al visualizador.

Puede consultar la tabla del apartado A4.10 del Apéndice A4 para averiguar el valor que debe asociar a un puerto para que un visualizador muestre una determinada cifra entre 0 y 9, o un punto. No obstante, usted puede introducir cualquier valor aunque no esté en la tabla, basándose solamente en la figura 4.38 para iluminar los segmentos deseados.

富民 Opción 15 segmentos.

Con este otro visualizador podemos representar caracteres más complejos como letras, símbolos matemáticos, etc.

Al pulsar en esta opción se muestran 8 visualizadores, de 15 segmentos cada uno, tales como los de la figura 4.39.

Siempre serán dieciséis los puertos que estén asociados a los visualizadores, que serán los que se indiquen en el campo de la parte inferior izquierda de la ventana.

Puede cambiar los puertos pulsando el botón Cambiar. Al pulsarlo, éste cambia y pasa a ser un campo de edición. Introduzca en el campo un valor (entre 00h y FFh) y pulse Intro. Entonces los puertos a los que se asocian los visualizadores, comenzarán a partir del puerto correspondiente al valor introducido.



Figura 4.42. 8 visualizadores de 15 segmentos.

Como hemos dicho, cada visualizador estará asociado a dos puertos. Por lo tanto, cada uno de los 16 bits que contiene el dato de los puertos servirá para activar un LED del visualizador. En la figura 4.40 se muestra la asociación que existe entre los bits de los puertos y los LEDs un visualizador.

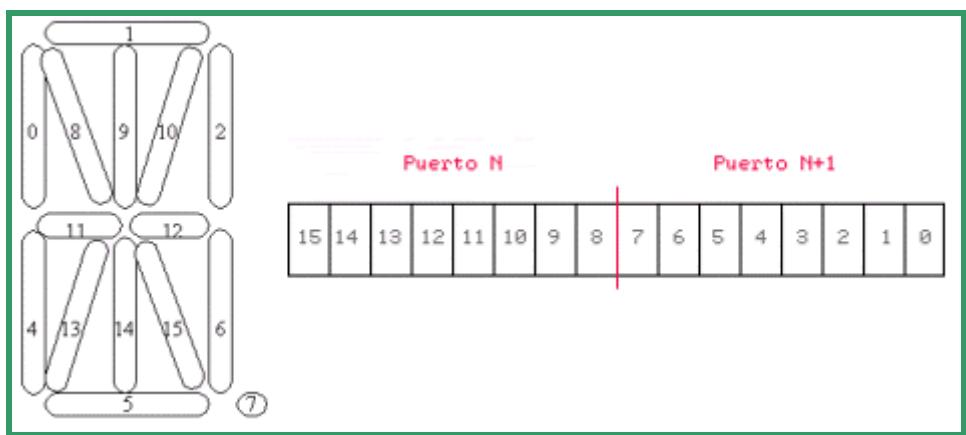


Figura 4.43. Estructura de un visualizador de 15 segmentos.

Como puede ver en la figura anterior, el mismo bit que en el visualizador anterior (7), se ha utilizado para añadir un punto.

Puede consultar la tabla del apartado A4.11 del Apéndice A4 para obtener automáticamente el valor que debe asociar a los dos puertos para que un

visualizador muestre una determinada cifra entre 0 y 9, una letra entre A y Z, o un punto.

Dispositivos de entrada

Veamos ahora los otros dispositivos que, al contrario que los de salida, que podían estar asociados tanto a memoria como a los puertos de salida, los de entrada solamente podrán estar asociados a los puertos de entrada. Son dos los tipos de dispositivos de entrada:

- Panel de interruptores.
- Teclado.

Describiremos cada uno de ellos por separado.

☞ Opción **Interruptores**.

Al elegir esta opción aparece un panel parecido al que muestra la figura 4.41.



Figura 4.44. Panel de interruptores.

Como sabemos un puerto almacena un dato de ocho bits y, por tanto, hemos asociado, a cada bit, un interruptor. De esta forma un puerto tendrá ocho interruptores asociados.

En el ejemplo de la figura, se muestran tres puertos, del 00h al 02h, tal y como se indica en la parte superior izquierda del panel.

Puede cambiar los puertos pulsando el botón Cambiar. Al pulsarlo, éste cambia y pasa a ser un campo de edición. Introduzca en el campo un valor (entre 00h y FFh) y pulse Intro. Entonces los puertos a los que se asocian los interruptores, comenzarán a partir del puerto correspondiente al valor introducido.

Entre uno y ocho puertos de entrada pueden tener asociados interruptores simultáneamente. Para establecer este número utilice el campo que aparece en la esquina superior derecha del panel. En el ejemplo de la figura 4.41 este campo contiene el valor 3.

Para cambiar el estado de un interruptor, activo (1 lógico) o inactivo (0 lógico) sitúe el puntero del ratón sobre él. Al hacerlo el puntero cambia para convertirse en una pequeña mano apuntando con el dedo índice. Pulse ahora con el botón izquierdo del ratón y verá que el interruptor cambia de estado, como ocurre en la figura 4.42.

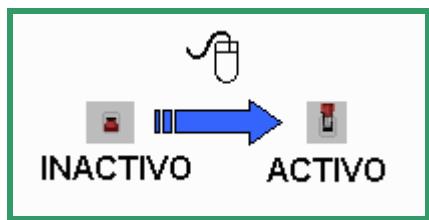


Figura 4.45. Cambiando un interruptor.

A la vez que el estado visual del interruptor cambia también lo hace el contenido del puerto de entrada asociado al interruptor. De esta forma, en nuestro ejemplo de la figura 4.41, y según el estado de los interruptores, el puerto 00h tendrá el valor AAh, el puerto 01h el valor 55h, y el puerto 02h el valor C3h.

- ❶ Recuerde que cuando un puerto esté asociado a una serie de interruptores y éstos últimos estén visibles en pantalla, no podrá modificar en la ventana de puertos del simulador el contenido de ese puerto, hasta que el panel de interruptores se oculte.

富民 Opción **Teclado**.

El teclado alfanumérico es el dispositivo típico de entrada de texto. La característica funcional más importante es la creación de un código único correspondiente a la tela pulsada.

Como comentamos en la opción Pantalla de texto de este mismo menú, el código ASCII utiliza un byte (8 bits) para representar un carácter. Este mismo código es el que se le asocia en cada tecla de un teclado normal. En esta opción del menú podrá utilizar un teclado como el de la figura 4.43.



Figura 4.46. Teclado ASCII.

La pulsación de cualquier botón del teclado está asociada al puerto que se indique en la parte inferior izquierda de la ventana. En nuestra figura, si pulsa por ejemplo el botón A, se visualizará en el puerto de entrada 00h el valor 41h, que es el código ASCII del carácter A.

Puede cambiar el puerto de entrada pulsando el botón Cambiar. Al pulsarlo, éste cambia y pasa a ser un campo de edición. Introduzca en el campo un valor (entre 00h y FFh) y pulse Intro.

Existen dos teclas especiales en el teclado:

- **Tecla Mays:** Nos indica si se está escribiendo (luz roja encendida), o no (luz apagada), en mayúsculas, ya que el código ASCII de un carácter minúscula no es el mismo que el del carácter en mayúsculas.
- **Tecla NULL:** Simula la no pulsación de una tecla, es decir, produce el valor 00h en el puerto asociado.

- *Otras teclas como ESC, TAB, BS, ENTER, etc.:* también tienen un código ASCII. Para más detalle consultar las tablas ASCII en los apéndices.

4.4.7. **Opciones** **Ejecución, interrupciones, ...**

Puede adaptar o mejorar el rendimiento del simulador a sus necesidades mediante las opciones que se incorporan en este menú. Introducir sufijos en los códigos de instrucción o cambiar la velocidad de ejecución en una simulación son algunas de las acciones que podrá realizar y que ahora le explicamos con detalle.

❖ Opción **De ejecución**.

Muestra un cuadro de diálogo como el de la figura 4.44 incluyendo distintas opciones relacionadas con el funcionamiento del simulador. Estas opciones aparecen agrupadas en tres conjuntos.

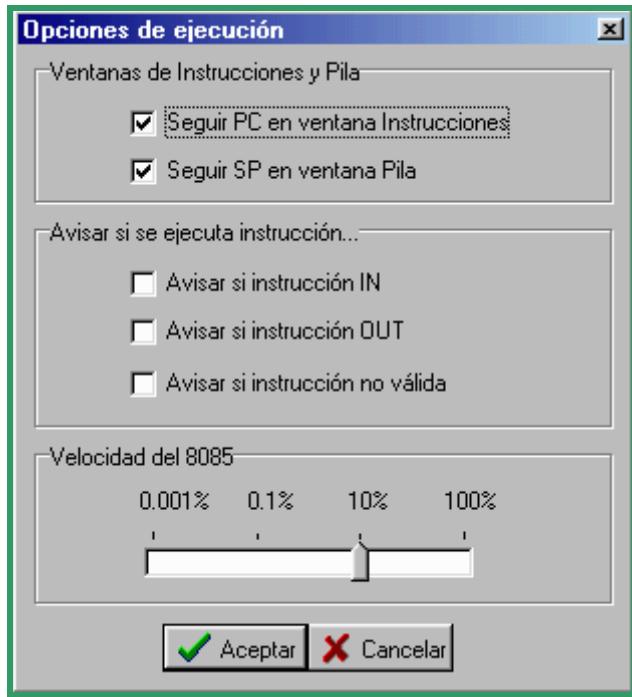


Figura 4.47. Opciones de ejecución.

El primer conjunto engloba opciones relativas a las ventanas de instrucciones y de pila:

- Seguir PC en ventana Instrucciones: Como se comentó, la posición de la memoria de instrucciones a la que apunta el contador de programa tiene el color **■**. Si activa esta opción, durante la simulación de un programa, la ventana de instrucciones se desplazará de forma que siempre sea visible la posición este color.

Por defecto esta opción está activada.

- Seguir SP en ventana Pila: De igual forma, la posición de la memoria de pila a la que apunta el puntero de pila tiene el color **■**. Si activa esta opción, durante la simulación de un programa, la ventana de pila se desplazará de forma que siempre sea visible la posición este color.

Por defecto esta opción está activada.

Las opciones del segundo conjunto hacen referencia a la ejecución de cierto tipo de instrucciones que incluyen instrucciones de entrada y salida e instrucciones no válidas. Las instrucciones no válidas son aquellas cuyo código de instrucción no ha sido utilizado a la hora de diseñar el microprocesador y por tanto no tienen ninguna funcionalidad:

- Avisar si instrucción IN: Cuando ejecute una instrucción IN aparece en pantalla un mensaje indicándole que llegan datos por el puerto de entrada afectado por la instrucción.

Por defecto esta opción está desactivada.

- Avisar si instrucción OUT: Cuando ejecute una instrucción OUT aparece en pantalla un mensaje indicándole que salen datos por el puerto de salida afectado por la instrucción.

Por defecto esta opción está desactivada.

- Avisar si instrucción no válida: Cuando ejecute una instrucción no válida aparece en pantalla un mensaje informándole de tal circunstancia.

Por defecto esta opción está desactivada.

El último grupo de opciones incluye una barra de desplazamiento horizontal, cuya posición influye en la velocidad de ejecución de una simulación. Puede tener 3 posibles estados:

- [0.001 %]: La velocidad se fija al 0.001 % de la máxima velocidad. Equivale a ejecutar aproximadamente 12 instrucciones por segundo. Muy útil si queremos ver cómo se ejecuta el programa con detalle pero sin tener que pulsar repetidamente el botón de paso a paso.
- [0.1 %]: La velocidad se fija al 0.1 % de la máxima velocidad. Se calcula que se ejecutan unas 100 instrucciones por segundo.
- [10 %]: La velocidad se fija al 10 % de la velocidad real de un 8085. Se podría decir que aproximadamente se logran unos 0.037 MIPS (*) (millones de instrucciones por segundo). [RECOMENDADA]
- [100 %]: Se obtiene la máxima velocidad, equivalente a 0.370 MIPS aproximadamente.

Por defecto la velocidad está fijada al 10 %, que es suficiente para la mayoría de los programas. Con la velocidad superior puede haber algún problema si el sistema donde se ejecuta el simulador no es muy rápido (menos de 400 MHz).

(*) La velocidad puede variar según el ordenador del usuario. Se indica siempre de forma aproximada.

Opción De interrupciones.

Puede generar automáticamente interrupciones durante la ejecución de un programa activando ciertas opciones en el cuadro de la figura 4.45.

Cada interrupción puede tener asociados 4 estados que son mutuamente excluyentes:

- “No”: En este estado no se generará la interrupción automáticamente.
- “Sí, al ejecutar instrucción no válida”: Volvemos a recordarle que las instrucciones no válidas son aquellas cuyo código de instrucción no ha sido utilizado a la hora de diseñar el microprocesador y por tanto no tienen ninguna funcionalidad. Activando esta opción, siempre que se ejecute una instrucción no válida, se producirá una interrupción.

- “Sí, al pulsar Teclado”: Siempre que el teclado que se vio en la opción Teclado del menú Dispositivos esté visible, y se pulse una de sus teclas se producirá una interrupción.
- “Sí, cada x milisegundos”: Cada vez que transcurra el tiempo indicado en el campo adyacente a la casilla de verificación de este estado, se producirá una interrupción.

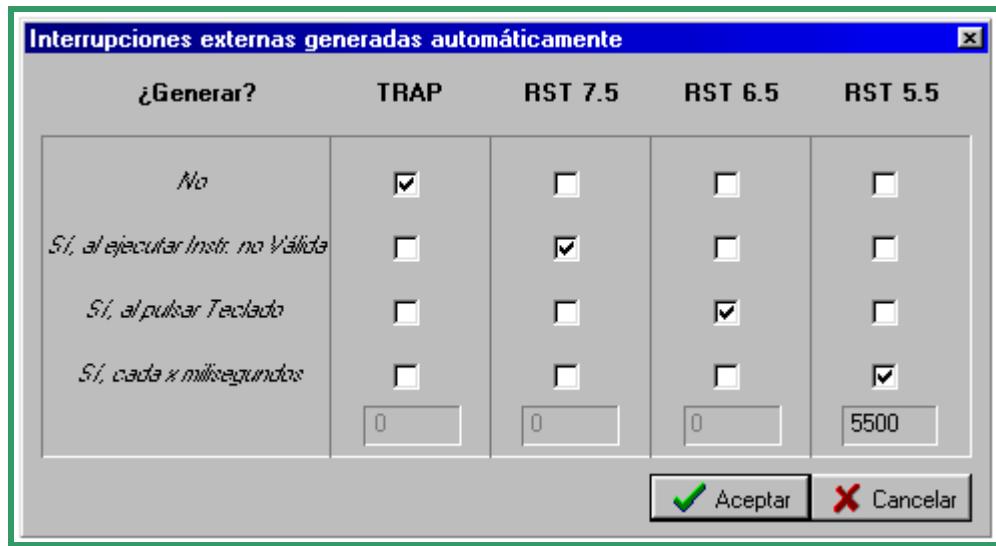


Figura 4.48. Opciones de interrupción.

Por defecto todas las interrupciones tienen el estado “No” asociado y, por tanto, no se genera ninguna interrupción automáticamente.

Estableciendo un tiempo para generar una interrupción...



Recuerde que el tiempo que desea que transcurra entre cada petición de interrupción debe expresarlo en milisegundos. No debe indicar un valor demasiado bajo, sobre todo si su ordenador no es muy rápido, pues podría provocar el mal funcionamiento de sus programas para 8085

En el ejemplo de la figura 4.45:

- La interrupción TRAP no se generará automáticamente.
- La interrupción RST 7.5 se producirá al ejecutar cualquier instrucción no válida.
- La interrupción RST 6.5 se producirá al pulsar cualquier tecla del teclado que incorpora el simulador.

- La interrupción RST 5.5 se producirá cada 5.5 segundos.

☞ Opción **De edición**.

Es en realidad un submenú que tiene dos opciones mutuamente excluyentes, tal y como puede ver en la figura 4.46. Las opciones hacen referencia a la base numérica que puede utilizar para introducir los datos en las ventanas del simulador.



Figura 4.49. Opciones de edición.

- ✓ Si la subopción Solo Hexadecimal está activa, como en el ejemplo de la figura, los datos que indican la posición de memoria, los códigos de instrucción, los números de puerto, etc, en las ventanas del simulador, aparecen sin el sufijo H.

A la hora de introducir los códigos de instrucciones, direcciones de memoria, datos, en cualquier parte del simulador, deberá hacerlo en hexadecimal (puede omitir el sufijo H o puede usarlo si así lo desea). Esta es la opción por defecto.

- ✓ Si la subopción Utilizar Sufijos está activa, las cifras que indican la posición de memoria y los códigos de instrucción en la ventana de instrucciones, aparecen con el sufijo H. Esto quiere decir que deberá obligatoriamente introducir sufijos en todos los apartados del simulador indicados anteriormente, pero a cambio de esto, puede usar todas las bases de numeración disponibles: decimal, hexadecimal, binaria y octal. (Si no indica sufijo, se tomará la numeración decimal en este caso).

4.4.8. Utilidades Simplificador, imágenes RAW

En esta parte del simulador dispone de un simplificador de expresiones, muy útil para algunas instrucciones del 8085, y un pequeño módulo para cargar y salvar imágenes en formato RAW.

☞ Opción Simplificador de expresiones.

Con esta opción podrá resolver cualquier expresión por complicada que sea. Aparece en pantalla una ventana como la de la figura 4.47. Introduzca en el campo superior una expresión utilizando sumas, divisiones, desplazamientos, módulos, etc. Para los operandos puede utilizar datos binarios (sufijo B), octales (sufijo O o Q), decimales (sin sufijo o sufijo D) o hexadecimales (sufijo H).

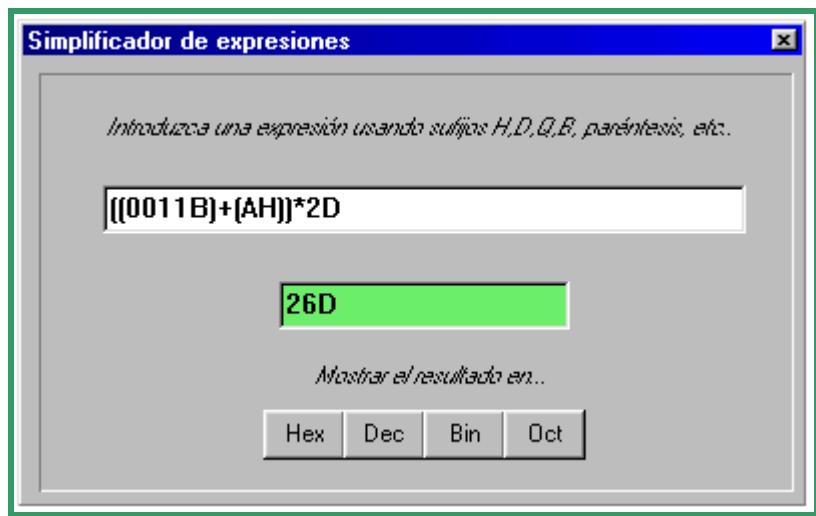


Figura 4.50. Simplificando una expresión.

En el ejemplo de la figura, primero se suma 3 (0011 en binario) y 10 (A en hexadecimal), y todo ello se multiplica por 2. El resultado se muestra en decimal, 26D, ya que se pulsado el botón Dec. También puede obtener el resultado en hexadecimal (botón Hex), binario (botón Bin) o en octal (botón Oct).

La expresión anterior es muy sencilla, introduzca expresiones tan complicadas como deseé, por ejemplo:

$$(03H * (10111000B * 45Q) SHR 2) SHR 2$$

cuyo resultado en decimal es 1276.

Por último indicarle que puede introducir tantos espacios en blanco como quiera entre un operador y un operando.

Opción **Imágenes RAW**.

Cuando alguna de las pantallas gráficas esté activa, es decir, sea visible en pantalla, la opción Imágenes RAW estará habilitada. En caso contrario, esta opción aparece en el menú pero no puede ser llamada.

RAW es un formato de imagen, que está destinado principalmente al almacenamiento el imágenes digitales de niveles de gris como matrices bidimensionales de puntos de imagen, en las que cada uno tiene asociado un nivel de luminosidad cuyos valores están en el conjunto $\{0, 1, \dots, 255\}$, de forma que el 0 indica mínima luminosidad (negro) y el 255 máxima luminosidad (blanco). Los restantes valores indican niveles intermedios de luminosidad (grises), siendo más oscuros cuanto menor sea su valor. Con esta representación, cada punto de imagen requiere únicamente un byte.

Para el almacenamiento de este tipo de imágenes, el formato RAW guarda únicamente la imagen en sí, sin ningún tipo de almacenamiento adicional, como el número de filas y columnas de la imagen. Así, una imagen de 160 filas y 100 columnas se almacenará en un tamaño de $160 \times 100 = 16000$ bytes.

Como ya se ha comentado en varias ocasiones, nuestro simulador de 8085 utilizará su memoria para almacenar las imágenes RAW.

También y, debido a la naturaleza de las imágenes RAW, le recordamos que la pantalla gráfica más adecuada para su visualización será la de tamaño 160x100 y 256 colores, ya que es la única que utiliza un byte para representar un punto de imagen.



¿No visualiza bien las imágenes RAW?

Utilice la pantalla gráfica de tamaño 160x100 y 256 colores ya que es la más adecuada a la naturaleza del formato RAW.

Con el resto de pantallas gráficas, que puede elegir en el menú Dispositivos visto anteriormente, también se visualizará la información contenida en la memoria del simulador, aunque visualmente los resultados no sean buenos.

Supongamos que usted pulsa en el menú Dispositivos, Pantalla Gráfica, 160x100 [256 colores], y después escoge la opción Imágenes RAW. En pantalla aparece el cuadro que muestra la figura 4.48.



Figura 4.51. Llamando al módulo de imágenes RAW.

Veamos por separado cada uno de los botones que contiene el módulo que aparece en pantalla.

- **Botón Cargar RAW:** al pulsarlo aparece un cuadro de diálogo para abrir un fichero con extensión RAW. Puede seleccionar un fichero y pulsar el botón Abrir.

Al abrir la imagen RAW, la pantalla gráfica que está visible cambia de aspecto y visualiza la información contenida en el fichero que ha escogido. La figura 4.49 muestra la situación después de cargar la imagen ALHAMBRA.RAW.

La carga de una imagen RAW produce el cambio en la memoria del simulador, destruyendo toda la información que esté contenida en las posiciones asociadas a la pantalla gráfica activa en ese momento. En

el ejemplo de la figura 4.49, todas las posiciones que van desde la 0000h hasta la 3E80h han sido reemplazadas.



Figura 4.52. Una imagen RAW se ha cargado.

En la figura 4.50 puede ver un nuevo ejemplo en el que se ha cargado un imagen RAW de 2 colores utilizando la pantalla gráfica de 256x200 puntos de imagen.



Figura 4.53. Una imagen RAW de 2 colores se ha cargado.

- Botón Salvar RAW: su pulsación muestra un cuadro de diálogo para guardar en disco una imagen RAW. Al realizar la operación las

posiciones de memoria asociadas a la pantalla gráfica, activa en ese momento, se salvan a disco con el nombre del fichero que usted elija.

Una vez salvada la imagen RAW, podrá recuperarla cuando lo desee mediante el botón Cargar RAW visto con anterioridad.

- Botón Cancelar: oculta el cuadro Imágenes RAW de la figura 4.48 sin realizar ninguna operación.

❖ Opción **Calculadora**.

Se muestra la calculadora de Windows, tal y como muestra la siguiente figura:

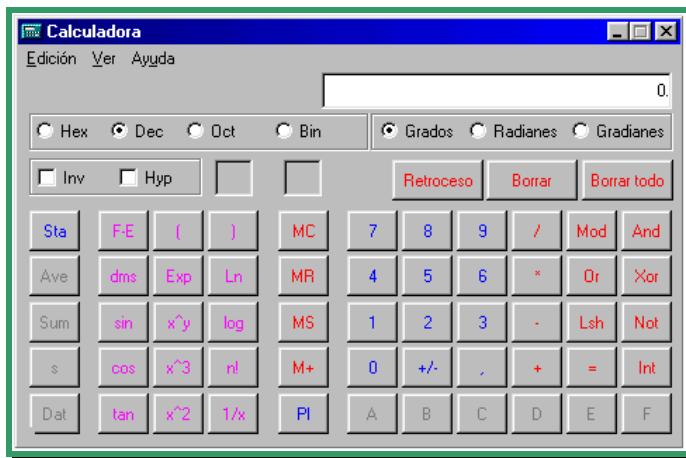


Figura 4.54. Calculadora de Windows.

4.4.9. **Ayuda** **Ayuda y créditos del programa**

Incluye un fichero de ayuda en línea y una pequeña ventana con información acerca del simulador y sus creadores.

❖ Opción **Contenidos**.

Aparece una ayuda en línea con información del microprocesador 8085. Aquí encontrará la ayuda que precise acerca de las instrucciones del 8085, junto con ejemplos de cada una de ellas, características internas del 8085 y otras peculiaridades del procesador.

Para más información, vea el apartado 4.4 de este mismo capítulo.

↪ **Opción Acerca de....**

Aparece la versión del programa y los autores.

4.5. Usar el sistema de ayuda

En este apartado aprenderá a usar el sistema de Ayuda en línea del simulador y otros componentes del programa.

Diferentes tipos de ayuda

El simulador tiene distintos métodos para obtener ayuda. Se pueden dividir en tres categorías:

- La ventana Temas de ayuda, que le da una tabla de contenidos y facilidad para buscar información de Ayuda en línea.
- Puede refrescar su memoria dejando el cursor del ratón sobre un componente durante unos pocos segundos sin pulsar. El simulador mostrará un pequeño rótulo junto al componente que identifica su función.
- Este libro, que le permite obtener información sobre cualquier elemento de la pantalla.

Usar la ventana de Temas de ayuda

Cuando pulsa Ayuda, Índice, aparece la ventana de Temas de ayuda. La ventana Temas de ayuda contiene dos pestañas, cada una de ellas corresponde a una ficha con un método diferente para buscar la información que necesita. Estas fichas se detallan en las secciones siguientes.

Los contenidos de la Tabla de ayuda

La ficha de Contenidos, que se muestra en la figura 4.52, organiza la información de Ayuda en temas de manera jerárquica. En el nivel superior hay una serie de *libros*, cada uno está identificado por un ícono de libro que hay junto al título. Cada libro puede

contener libros adicionales así como *temas* que contienen información de Ayuda. Un tema está identificado por el icono de una página con un signo de interrogación dentro.



Figura 4.55. La ficha de Contenidos en la ventana de Temas de ayuda.

Dentro de la ficha Contenidos puede realizar lo siguiente:

- Para abrir un libro y visualizar su contenido, pulse dos veces en el título del libro. Un libro abierto tiene un icono de libro abierto junto a él.
- Para cerrar un libro y ocultar su contenido, pulse dos veces en el título del libro.
- Para abrir un tema, pulse dos veces en el título.
- Para imprimir un tema, seleccione el tema y pulse Imprimir.
- Para cerrar la ventana de Ayuda, pulse el botón Cancelar o pulse el botón Cerrar de la barra de título.

Ver un Tema de ayuda

Cada Tema de ayuda contiene información diferente, pero cada tema está organizado más o menos de la misma forma. Hay varios tipos de pantallas de Temas de ayuda. La que verá con más frecuencia contiene texto. Aquí están las acciones que puede realizar:

- Pulse el término subrayado para ver una definición de tal término.
- Pulse el botón Atrás para volver al Tema de ayuda anterior (si lo hubiera).
- Pulse el botón Temas de ayuda para volver a la ficha de contenidos de la ventana Temas de ayuda.
- Pulse el botón Opciones, después seleccione Imprimir Tema, para imprimir el tema que se está viendo.
- Pulse Esc o pulse el botón de Cerrar de la esquina superior derecha de la barra de título para cerrar la Ayuda y volver al simulador.

Usar la ficha Buscar de la ventana Temas de ayuda

La ficha Buscar de la ventana de Temas de ayuda le permite buscar el tema que le interese a través de la información de Ayuda en línea. No tiene ninguna limitación en los títulos de Temas de ayuda, puede buscar las palabras por todo el sistema de Ayuda. Normalmente usará Buscar sólo cuando no hay podido localizar la información necesaria usando la ficha de Contenidos.

La primera vez que use Buscar, tiene que elaborar la lista de palabras. Haga esto con la Plantilla activa Asistente para la configuración de Buscar, que aparece automáticamente. En la primera pantalla de la Plantilla activa, se recomienda que pulse la opción Minimizar tamaño de base de datos (recomendado), después pulse Siguiente seguido de Finalizar. Elaborar la lista le puede llevar unos segundos, pero sólo tiene que hacerlo una vez.

La figura 4.53 muestra la ficha Buscar. Aquí tiene cómo usarla:

1. Escriba la palabra o la frase que desea buscar en el cuadro de texto. Teclee las palabras que quiere buscar.
2. El simulador automáticamente visualiza palabras que se corresponden o están relacionadas en el cuadro de lista Seleccione algunas palabras coincidentes para limitar la búsqueda. Inicialmente, todas estas palabras están seleccionadas. Para restringir la búsqueda, seleccione las palabras más relevantes pulsando en ellas. Para seleccionar dos o más palabras en la lista, pulse la tecla Mayús al mismo tiempo que pulsa con el ratón.
3. El cuadro de lista Haga click en un tema y después en Mostrar, muestra una relación de los temas de Ayuda que están relacionados con los términos de su búsqueda. Pulse el tema deseado, después pulse Mostrar.

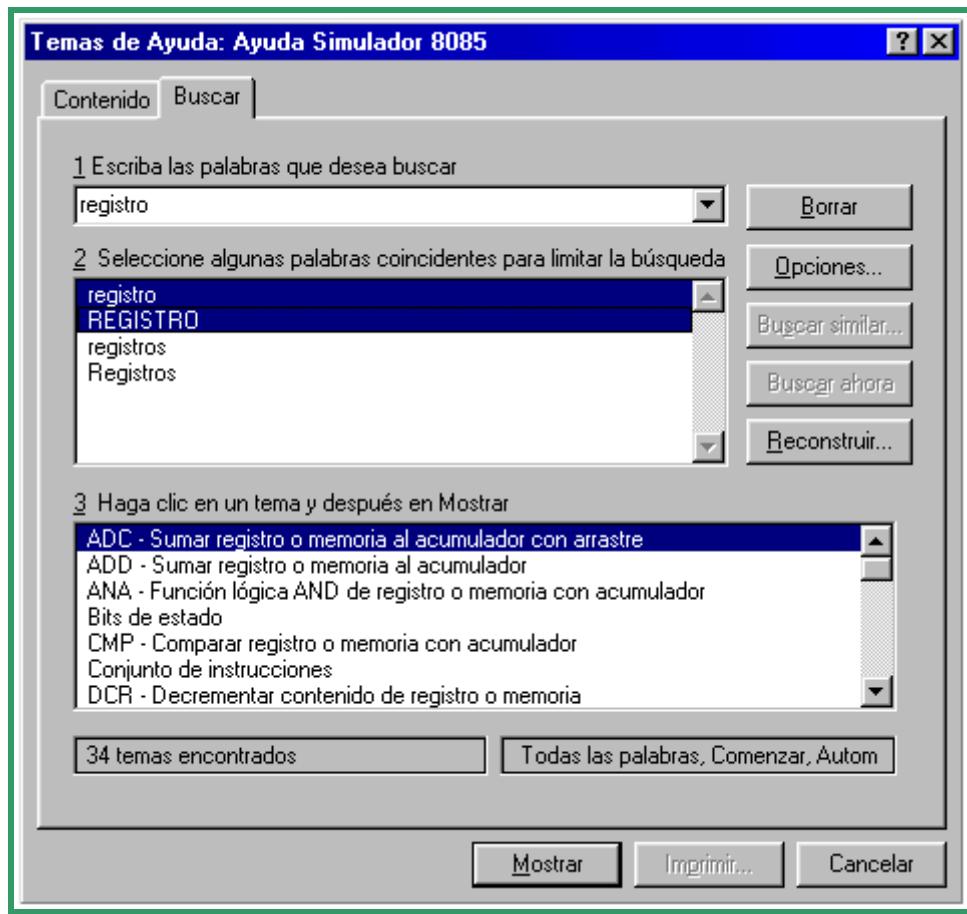


Figura 4.56. La ventana de la ficha Buscar de los Temas de ayuda.

Apéndice A4 Tablas y códigos

A4.1. Código ASCII

NULL 00	+	2B	?	3F	S	53	g	67	{	7B
BS 08	,	2C	@	40	T	54	h	68		7C
TAB 09	-	2D	A	41	U	55	i	69	}	7D
ENTER 0D	.	2E	B	42	V	56	j	6A	~	7E
ESC 1B	/	2F	C	43	W	57	k	6B		
RIGHT 10	0	30	D	44	X	58	l	6C	i	A1
LEFT 11	1	31	E	45	Y	59	m	6D	ó	BF
UP 1E	2	32	F	46	Z	5A	n	6E	ñ	D0
DOWN 1F	3	33	G	47	[5B	o	6F	Ñ	D1
SPACE 20	4	34	H	48	\	5C	p	70		
!	21	5	35	I	49]	5D	q	71	
"	22	6	36	J	4A	^	5E	r	72	
#	23	7	37	K	4B	~	5f	s	73	
\$	24	8	38	L	4C	·	60	t	74	
%	25	9	39	M	4D	a	61	u	75	
&	26	:	3A	N	4E	b	62	v	76	
'	27	;	3B	O	4F	c	63	w	77	
(28	<	3C	P	50	d	64	x	78	
)	29	=	3D	Q	51	e	65	y	79	
*	2A	>	3E	R	52	f	66	z	7A	

A4.2. Tabla de conversión Hexadecimal ↔ Binario

HEX	BIN	HEX	BIN
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

A4.3. Tabla de conversión Decimal ↔ Hexadecimal

HEX	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
8	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
9	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
A	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
B	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
C	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
D	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
E	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
F	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

A4.4. Tabla de conversión Decimal ↔ Hexadecimal en complemento a dos

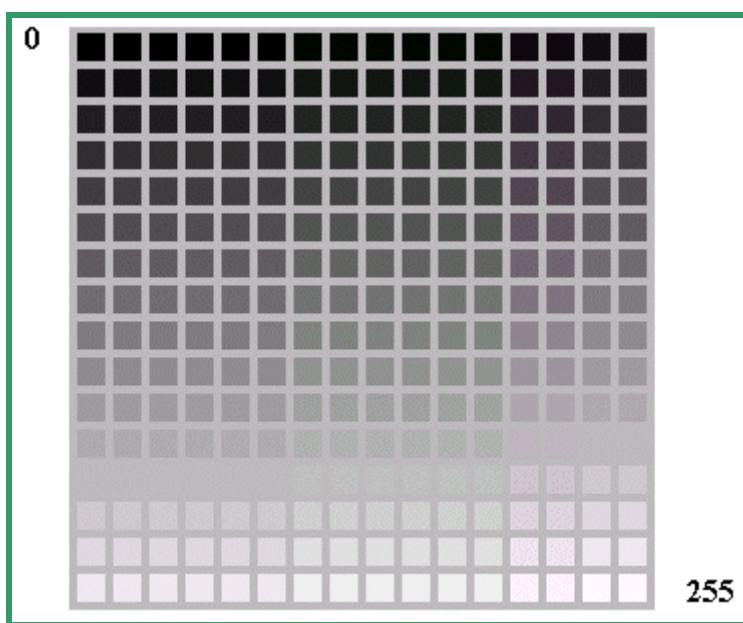
HEX	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	-128	-127	-126	-125	-124	-123	-122	-121	-120	-119	-118	-117	-116	-115	-114	-113
9	-112	-111	-110	-109	-108	-107	-106	-105	-104	-103	-102	-101	-100	-99	-98	-97
A	-96	-95	-94	-93	-92	-91	-90	-89	-88	-87	-86	-85	-84	-83	-82	-81
B	-80	-79	-78	-77	-76	-75	-74	-73	-72	-71	-70	-69	-68	-67	-66	-65
C	-64	-63	-62	-61	-60	-59	-58	-57	-56	-55	-54	-53	-52	-51	-50	-49
D	-48	-47	-46	-45	-44	-43	-42	-41	-40	-39	-38	-37	-36	-35	-34	-33
E	-32	-31	-30	-29	-28	-27	-26	-25	-24	-23	-22	-21	-20	-19	-18	-17
F	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

A4.5. Tabla de conversión Octal ↔ Binario

OCT	BIN
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

A4.6. Tabla números BCD

DECIMAL	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

A4.7. Paleta de 256 niveles de gris**A4.8.** Paleta de 16 colores

A4.9.  **Paleta de 2 colores****A4.10.**  **Visualizador de 7 segmentos**

Número	HEX	BIN
0	77	01110111
1	44	01000100
2	3E	00111110
3	6E	01101110
4	4D	01001101
5	6B	01101011
6	7B	01111011
7	46	01000101
8	7F	01111111
9	4F	01001111
.	XX	1xxxxxxx

A4.11.  **Visualizador de 15 segmentos**

Carácter	HEX	BIN	Carácter	HEX	BIN
0	24	00100100 01110111	J	00	74 00000000 01110100
1	00	44 00000000 01000100	K	C6	00 11000110 00000000
2	00	3E 00000000 00111110	L	00	31 00000000 00110001
3	00	6E 00000000 01101110	M	05	55 00000101 01010101
4	00	4D 00000000 01001101	N	81	55 10000001 01010101
5	00	6B 00000000 01101011	Ñ	81	57 10000001 01010111
6	00	7B 00000000 01111011	O	00	77 00000000 01110111
7	00	46 00000000 01000101	P	00	1F 00000000 00011111
8	00	77 00000000 01110111	Q	80	77 10000000 01110111
9	00	4F 00000000 01001111	R	80	1F 10000000 00011111
A	00	5F 00000000 01011111	S	18	63 00011000 01100011
B	52	66 01010010 01100110	T	42	02 01000010 00000010
C	00	33 00000000 00110011	U	00	75 00000000 01110101
D	42	66 01000010 01100110	V	24	11 00100100 00010001
E	00	3B 00000000 00111011	W	A0	55 10100000 01010101
F	00	1B 00000000 00011011	X	A5	00 10100101 00000000
G	10	73 00010000 01110011	Y	45	00 01000101 00000000
H	00	5D 00000000 01011101	Z	24	22 00100100 00100010
I	42	22 01000010 00100010	.	XX	XX 00000000 10000000