INTRODUCCION

Las instrucciones de control de programa direccionan el flujo de un programa. Las instrucciones de control de programa permiten que el flujo del programa cambie. Esos cambios en el flujo ocurren después de decisiones hechas con las instrucciones CMP o TEST seguidas por una instrucción de salto condicional.

El Grupo de Saltos

El salto (JMP), es el tipo principal de instrucciones de control de programa, permite al programador saltar a secciones de un programa y bifurcar a cualquier parte de la memoria para la siguiente instrucción. Las instrucciones de salto condicional permiten al programador hacer decisiones basadas sobre pruebas numéricas. Los resultados de esas pruebas numéricas son sostenidas en los bits de banderas, las cuales son probadas por las instrucciones de salto condicional.

Salto Incondicional (JMP)

Están disponibles tres tipos de instrucciones de salto incondicional (refiérase a la Figura 1) en el juego de instrucciones del microprocesador 8088/86: saltos cortos, salto cercano y salto lejano. El salto corto es una instrucción de 2 bytes que permite saltar o bifurcar a localidades de memoria dentro de +127 y -128 de la localidad de memoria seguida del salto. El salto cercano de 3 bytes permite bifurcar o saltar dentro de ±32K bytes (en cualquier lugar) de la instrucción en el segmento de código presente. Finalmente, el salto lejano de 5 bytes permite un salto a cualquier localidad de memoria dentro del sistema completo de memoria. Los saltos corto y cercano son llamados comúnmente saltos de intrasegmentos, y el salto lejano es comúnmente llamado salto de intersegmento.

Salto Corto. Los saltos cortos son llamados saltos relativos porque se mueven a cualquier lugar en memoria sin un cambio. Esto es porque no se almacena una dirección en el código de operación. En lugar de una dirección, sigue al código de operación una distancia o desplazamiento. El desplazamieno de salto corto es una distancia representada por un número con signo de 1 byte cuyos valores fluctúan entre +127 y -128. Las instrucciones de salto corto aparecen en la Figura 2. Cuando el 8088/86 ejecuta un salto corto, el desplazamiento es con signo extendido y se suma al apuntador de instrucción (IP) para generar la dirección de salto dentro del segmento de código presente. La instrucción de salto corto bifurca a esta nueva dirección para la siguiente instrucción en el programa.

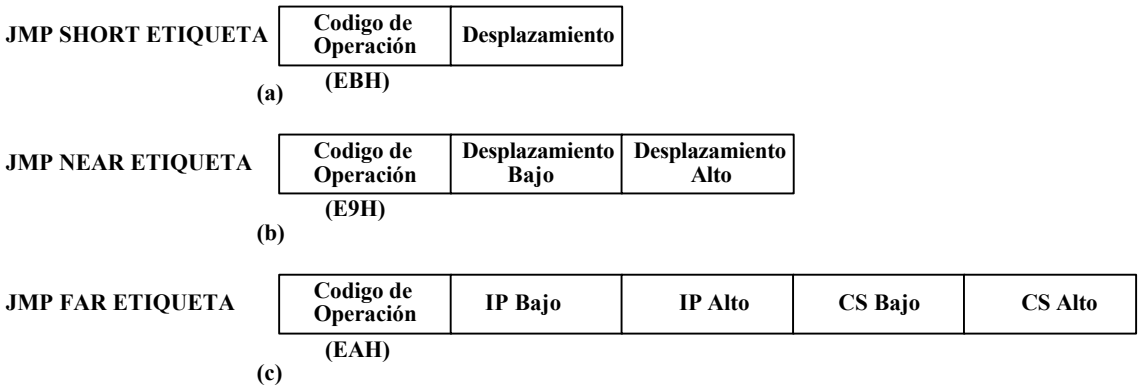

FIGURA 1. Los tipos de instrucciones de salto. (a) JMP corto (2 bytes), (b) JMP cercano (3 bytes) y (c) JMP lejano (5 bytes).

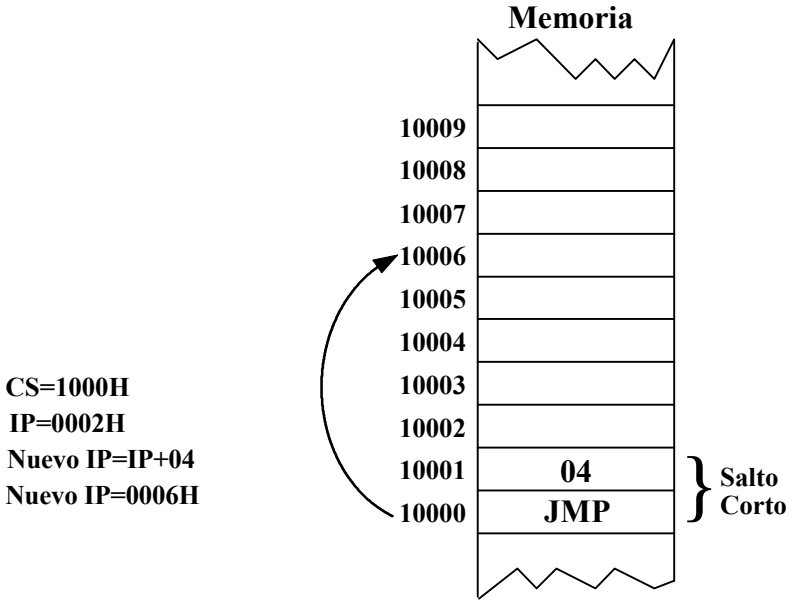

FIGURA 2. Un JMP corto a una localidad de memoria más allá de la dirección de la siguiente instrucción. (Note que esto hará saltar a los siguientes 4 bytes de memoria.)

EJEMPLO 1

```
                XOR   BX,BX
START:          MOV   AX,1
                ADD   AX,BX
                JMP   SHORT NEXT

NEXT:  MOV      BX,AX
                JMP   START
```

El ejemplo 1 muestra como las instrucciones de salto corto pasan el control de una parte del programa a otro. Este también ilustra el uso de una etiqueta con la instrucción de salto. Note como un salto (JMP SHORT NEXT) usa el directivo SHORT para forzar a un salto corto, mientras el otro no lo hace.

El ensamblador escoge la mejor forma de la instrucción de salto por lo tanto la segunda instrucción de salto (JMP START) también se ensambla como un salto corto. Si nosotros añadimos la dirección de la siguiente instrucción (0009H) a el desplazamiento con signo extendido (0017H) del primer salto, la dirección de NEXT esta en 0017H + 0009H o 0020H. Cada vez que una instrucción de salto hace referencia a una dirección, una etiqueta identifica la dirección. El JMP NEXT es un ejemplo, el cual salta a la etiqueta NEXT para la siguiente instrucción. Nunca usamos una dirección hexadecimal con cualquier instrucción de salto. La etiqueta NEXT debe ser seguida por dos puntos (NEXT:) para permitir a una instrucción referenciarla para un salto. Si los dos puntos no siguen a la etiqueta, no podrás saltar a esta.

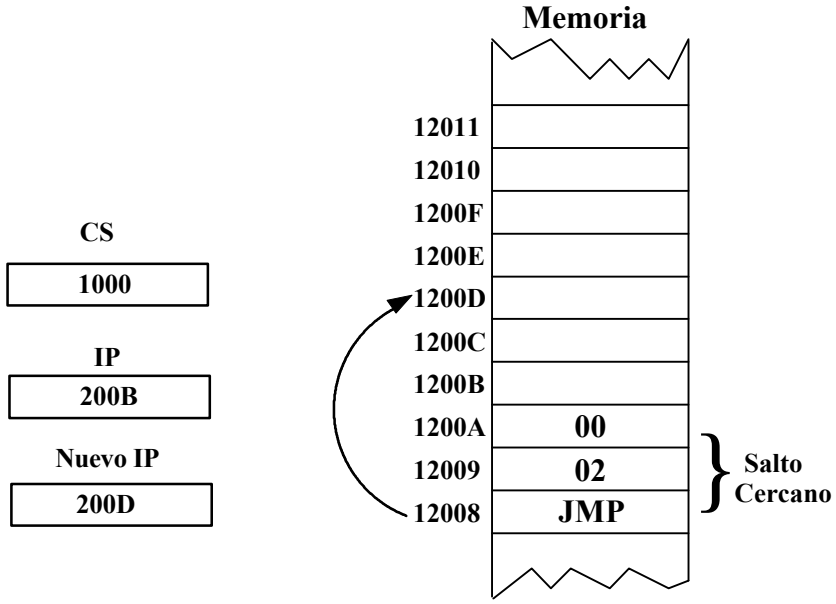

FIGURA 3 Un JMP cercano, el cual suma el desplazamiento a el contenido del registro IP.

Salto Cercano. El salto cercano es similar a el salto corto excepto que la distancia de salto es más lejos. Un salto cercano pasa el control a una instrucción en el segmento de código presente localizado dentro de ±32 Kbytes de la instrucción de salto cercano. El salto cercano es una instrucción de 3 bytes que contiene un código de operación seguido por un desplazamiento de 16 bits con signo. El desplazamiento con signo se suma a el IP para generar la dirección del salto. Ya que el desplazamiento con signo está en el rango de ±32K, un salto cercano puede saltar a cualquiera localidad de memoria dentro del segmento de código presente. La Figura 3 ilustra la operación de la instrucción de salto cercano. El salto cercano es también relocalizable como es el salto corto porque es también un salto relativo. Si el segmento de código se mueve a una nueva localidad de memoria, la distancia entre la instrucción de salto y la dirección del operando se mantiene igual. Esto permite a un segmento de código ser relocalizado moviéndolo. Esta característica, con los segmentos de datos relocalizable, hace a la familia de microprocesadores de Intel ideal para uso en sistemas de computadora de propósito general. Los programas pueden ser escritos y cargados en cualquier lugar de la memoria y funcionan sin modificaciones por los saltos relativos y los segmentos de datos relocalizables. Esta característica es cierta para pocos microprocesadores.

EJEMPLO 2

```
                XOR    BX,BX

START:          MOV    AX,1
                ADD    AX,BX
                JMP    NEXT

NEXT:           MOV    BX,AX
                JMP START
```

El ejemplo 2 muestra el mismo programa básico que aparece en el Ejemplo 1, excepto que la distancia del salto es mayor. El primer salto (JMP NEXT) pasa el control a la instrucción en la localidad de memoria 0200H dentro del segmento de código. Note que la instrucción se ensambla como E9 0200 R, la letra R indica la dirección de salto relocalizable de 0200H. La dirección relocalizable de 0200H es solo para el uso interno del ensamblador. La instrucción actual en lenguaje máquina la ensambla como E9 F6 01, la cual no aparece en el listado del ensamblador. El desplazamiento actual es un 01F6H para esta instrucción de salto. El ensamblador despliega la dirección de salto como 0200 R por lo que la dirección es más fácil de interpretar cuando desarrollamos programas. Si vemos el archivo ligado en hexadecimal, podemos ver que este salto se ha ensamblado como E9 F6 01.

Salto Lejano. Los saltos lejanos (vea Figura 4) obtienen un nuevo segmento y dirección de compensación para efectuar el salto. El byte 2 y 3 de esta instrucción de 5 bytes contienen la nueva dirección de desplazamiento, y los bytes 4 y 5 contienen la dirección del nuevo segmento.

El ejemplo 3 muestra un programa que usa una instrucción de salto lejano. La instrucción de salto lejano algunas veces aparece con la directiva FAR PTR como es ilustrado. Otro camino para obtener un salto lejano es definir una etiqueta como etiqueta lejana. Una etiqueta es lejana solo si es externa a el segmento de código presente. La instrucción JMP UP en el ejemplo hace referencia a una etiqueta lejana. La etiqueta UP esta definida como una etiqueta lejana por EXTRN UP:FAR. Las etiquetas externas aparecen en programas que contienen más de un archivo de programa.

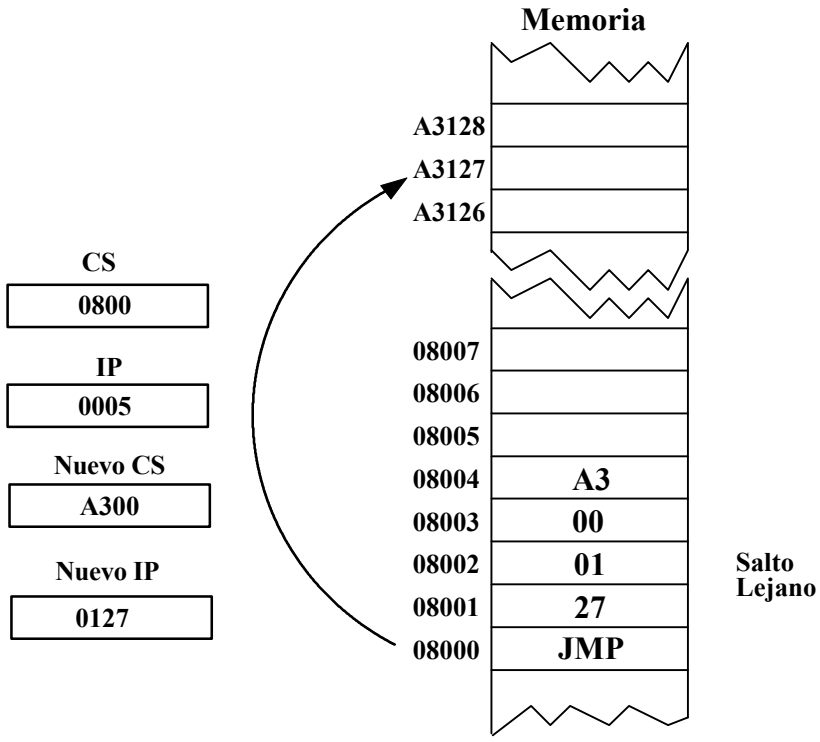

FIGURA 4 Un JMP lejano que remplaza el contenido de los registros CS e IP con los 4 bytes que siguen al código de instrucción.

EJEMPLO 3

```
                              EXTRN UP:FAR

0000 33 DB                    XOR    BX,BX

0002 B8 0001    START:        MOV    AX,1
0005 03 C3                    ADD    AX,BX
0007 E90200 R                 JMP    NEXT

0200 8B DB      NEXT:         MOV    BX,AX
0202 EA 0002 ----R            JMP FAR PTR START

0207 EA 0000 ----E            JMP UP
```

Formato:
Localidad (offset) código correspondiente a la instrucción

Cuando se unen los archivos del programa, el ligador inserta las direcciones para la etiqueta UP en la instruccion JMP UP. Este también inserta la dirección del segmento en la instrucción JMP START. La direccion de segmento en JMP FAR PTR START es desplegada como ----R por relocalizable, y la direccion del segmento en JMP UP es desplegada como ----E por externa. En ambos casos ---- es llenado por el ligador cuando liga o une los
archivos del programa.

Saltos con Registros como Operando. La instrucción de salto también puede especificar un operando como un registro de 16 bits. Esto pone automáticamente a la instrucción como un salto indirecto. La dirección del salto está en el registro especificado por la

instrucción de salto. Distinto a el desplazamiento con el salto cercano, el contenido del registro se transfiere directamente en el apuntador de instrucción. Este no se suma a el apuntador de instrucción como en los saltos cortos y cercanos. La instrucción JMP AX, por ejemplo, copia el contenido del registro AX en IP cuando el salto ocurre. Esto permite un salto a cualquier localidad dentro del segmento de código presente.

EJEMPLO 4

```
0000 03 F6                  ADD   SI,SI              ;doblar SI
0002 81 C6 000B R           ADD   SI,OFFSET TABLE    ;direc. de TABLE
0006 2E:8B 04               MOV   AX,CS:[SI]         ;obtiener dirección
0009 FF E0                  JMP   AX

000B   1000 R  TABLE:       DW    ZERO
000D   2000 R               DW    ONE
000F   3000 R               DW    TWO
```

El Ejemplo 4 muestra como la instrucción JMP AX accesa a una tabla de salto. Esta secuencia de instrucciones asume que SI contiene 0, 1, o 2. Ya que la tabla de salto contiene una dirección de desplazamiento de 16 bits, el contenido de SI se dobla a 0, 2, ó 4 así que puede ser accesada una entrada de 16 bits. La dirección del desplazamiento del inicio de la tabla de salto se suma a SI para hacer referencia a la dirección de salto. La instrucción MOV AX,CS:[SI] busca una dirección del segmento de código en la tabla de saltos por lo que la instrucción JMP AX salta a la localidad de la tabla de saltos.

Este ejemplo busca la dirección ZERO si aparece un 0 en SI. Una vez que se encuentra la dirección, la instrucción JMP AX causa que el programa salte a la dirección ZERO. Lo mismo es verdadero para un valor inicial de 1 y 2 en el registro SI. Esos números accesan las direcciones de ONE y TWO para sus localidades de salto.

Saltos Indirectos Usando un Indice. La instrucción de salto también puede utilizar la forma de direccionamiento [ ] para directamente accesar a la tabla de salto. La tabla de salto puede contener direcciones de desplazamiento para saltos cercanos indirectos o direcciones de segmentos y desplazamiento para saltos lejanos indirectos. El ensamblador asume que el salto es cercano al menos que la directiva FAR PTR indique una instrucción de salto lejano. En el Ejemplo 5 se repite el Ejemplo 4 usando JMP CS:[SI] en lugar de JMP AX. Esto reduce la longitud del programa.

EJEMPLO 5

```
0000 03 F6                  ADD   SI,SI              ;doblar SI
0002 81 C6 0009 R           ADD   SI,OFFSET TABLA    ;más direc. TABLA
0006 2E:FF 24               JMP   CS:[SI]

0009 1000 R     TABLA:      DW    ZERO
000B 2000 R                 DW    ONE
000D 3000 R                 DW    TWO
```

El mecanismo usado para accesar la tabla de salto es idéntica con una referencia de memoria normal. La instruccion JMP CS:[SI] apunta a la dirección de salto almacenada en la localidad de memoria del segmento de código direccionada por SI. Este salta a la dirección almacenada en esta localidad de memoria. Las instrucciones de registro y el índice indirecto usualmente direccionan un desplazamiento de 16 bits. Esto significa que los dos tipos de saltos son saltos cercanos. Si se ejecuta una instrucción JMP FAR PTR [SI], el microprocesador asume que la tabla de saltos contiene direcciones de doble palabra (IP y CS).

SALTOS CONDICIONADOS

Los saltos condicionados son siempre saltos cortos. Esto limita el rango del salto dentro de +127 bytes y -128 bytes de la localidad seguida del salto condicional. La Tabla 1 muestra todas las instrucciones de salto condicional con sus condiciones de prueba.

TABLA 1. Instrucciones de salto condicionales

Instrucción	Condicion probada	Comentario
JA	C=0 y Z≠0	Salta si esta arriba
JAE	C=0	Salta si esa arriba o igual
JB	C=1	Salta si esta abajo
JBE	C=1 o Z=1	Salta si esta abajo o igual
JC	C=1	Salta si hay acarreo
JE o JZ	Z=1	Salta si es igual o Sata si es 0
JG	Z=0 y S=0	Salta si es mayor
JGE	S=0	Salta si es mayor o igual a
JL	S≠0	Salta si es menor
JLE	Z=1 o S=1	Salta si es menor o igual
JNC	C=0	Salta si no hay acarreo
JNE o JNZ	Z=0	Salta si no es igual o Salta si no es 0
JNO	O=0	Salta si no hay sobre flujo
JNS	S=0	Salta si no hay signo
JNP	P=0	Salta si no hay paridad
JO	O=1	Salta si hay sobre flujo
JP	P=1	Salta si hay paridad
JS	S=1	Salta si hay signo
JCXZ	CX=0	Salta si CX=0

Los saltos condicionados examinan los siguientes bits de bandera: signo (SF), cero (ZF), acarreo (CF), paridad (PF), y desbordamiento (OF). Si la condición bajo prueba es verdadera, ocurre una bifurcación a la etiqueta asociada con la instrucción de salto. Si la condición es falsa, el siguiente paso secuencial se ejecuta en el programa.

La operación de la mayoria de las instrucciones de salto condicionados es directa porque prueban solo un bit de bandera, aunque algunos prueban más de una bandera. La comparación de magnitud relativa requiere instrucciones de salto condicional más sofisticadas que prueban más de un bit de bandera.

Ya que se usan números con signo y números sin signo, y el orden de esos números es diferente; existen dos conjuntos de instrucciones de salto condicionados de comparación de magnitud. La Figura 5 muestra el orden de los números con signo y sin signo de 8 bits. Los números de 16 bits siguen el mismo orden que los números de 8 bits solo que son más grandes. Note que un FFH esta arriba del 00H en el conjunto de los números sin signo, pero un FFH (-1) es menor que 00H para los números con signo. Por lo tanto, un FFH sin signo es mayor que 00H, pero un FFH con signo es menor que 00H.

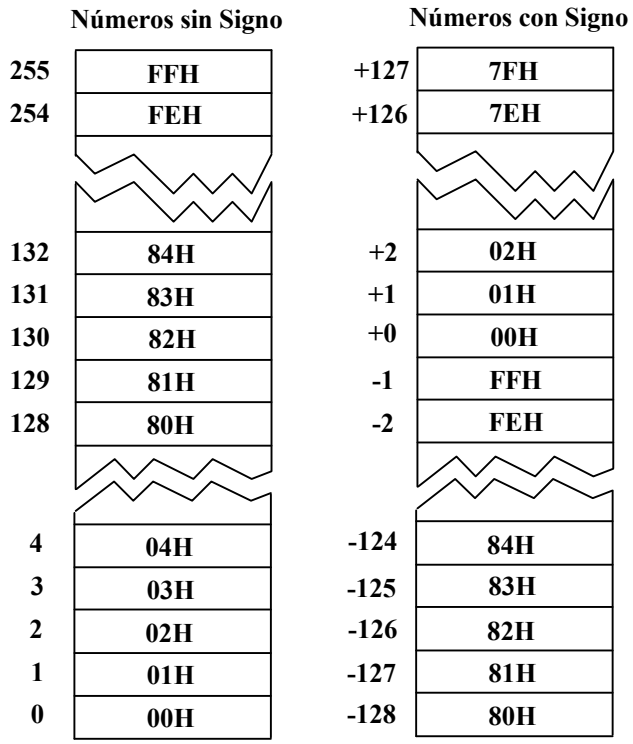

FIGURA 5. Numero con signo y sin signo.

Cuando comparamos números con signo, usamos JG, JE, JGE, JLE, JE, y JNE. Los terminos mayor que y menor que se refieren a números con signo. Cuando comparamos números sin signo, usamos JA, JB, JAE, JBE, JE, y JNE. Los términos es mayor y es menor se refieren a números sin signo.

Los saltos condicionales restantes examinan bits de banderas individuales tales como el de sobreflujo y paridad. Note que JE tiene un código de operación alternativo, JZ. Todas las instrucciones tienen una instrucción alternativa, pero muchas no son usadas en programación porque no tiene mucho significado excepto para JE/JZ y JNE/JNZ.

Por ejemplo, la instruccion JA/JBE (salta si esta arriba/salta si esta abajo o igual). Esto hace poco compresible decir salta si esta arriba o igual cuando es mas claro decir salta si esta arriba. JBE es una alternativa de JA que funciona exactamente igual.

La más radical de las instrucciones de salto condicional es JCXZ (salta si CX=0). Esta es la única instrucción de salto condicional que no examina los bits de bandera. En lugar de eso, JCXZ directamente examina el contenido del registro CX sin afectar los bits de bandera. Si CX=0, ocurre el salto, y si CX es diferente de cero, el salto no ocurre.

En el ejemplo 6 aparece un procedimiento que usa JCXZ. Aquí la instrucción SCASB busca en una tabla el valor 0AH. Después de la busqueda, una instrucción JCXZ examina CX para ver si el contador ha llegado a cero. Si el contador es cero, el valor 0AH no se encontro en la tabla. Otro método usado para ver si el dato fué encontrado es la instrucción JNE. Si JNE remplaza a JCXZ, está realiza la misma función. Después de que se ejecuta SCASB, las banderas indican una condición de no igual si el dato no fue encontrado en la tabla.

EJEMPLO 6

```
                    ;Procedimiento que busca  0AH en una tabla      ; de 100 bytes

0000          SCAN      PROC  NEAR

0000 BE 0000 R          MOV SI,OFFSET TABLE     ;direccion de TABLE
0003 B9 0064            MOV CX,100              ; carga contador
0006 B0 0A              MOV AX,0AH              ; carga dato a buscar
0008 FC                 CLD
0009 F2/AE              REPNE SCASB
000B E3 23              JCXZ  NOT_FOUND
000D C3                 RET

000E          SCAN      ENDP
```

LOOP

La instruccion LOOP es una combinación de un decremento en CX y un salto condicionado. Está decrementa a CX y si CX es diferente de cero, se salta a la dirección indicada por la etiqueta. Si CX llega a cero, se ejecuta la siguiente instrucción secuencial.

EJEMPLO 7

```
                ;procedimiento para sumar palabras de un BLOCK1 a un BLOCK2

0000            ADDS        PROC    NEAR

0000 B9 0064                MOV CX,100              ;carga contador
0003 BE 0064 R              MOV SI,OFFSET BLOCK1    ;direcc. de BLOCK1
0006 BF 0000 R              MOV DI,OFFSET BLOCK2    ;direcc. de BLOCK2

0009            AGAIN:
0009 AD                     LODSW                   ;toma dato de BLOCK1
000A 26:03 05               ADD AX,ES:[DI]          ;sumar al dato de BLOCK2
000D AB                     STOSW                   ;almacenar en BLOCK2
000E E2 F9                  LOOP AGAIN              ;repetir 100 veces
0010 C3                     RET

0011            ADDS        ENDP
```

El ejemplo 7 muestra como los datos en un bloque de memoria (BLOCK1) se suman a los datos del segundo bloque de memoria (BLOCK 2) usando LOOP para controlar cuantos números se suman. Las instrucciones LODSW y STOSW accesan a los datos en BLOCK1 y BLOCK2. La instrucción ADD AX,ES:[DI] accesa los datos de BLOCK2 localizados en el segmento extra. La unica razón por la cual está en el segmento extra es que DI direcciona los datos en el segmento extra para la instrucción de STOSW.

LOOPs Condicionados. Tal como REP, la instrucción LOOP también tiene las formas condicionales: LOOPE y LOOPNE. La instrucción LOOPE (cicla mientras sea igual) salta si CX es diferente de cero mientras una condición de igual existe. Se saldrá del ciclo si la condición no es igual o si el registro CX se decrementa a cero. La instrucción LOOPNE (cicla mientras no sea igual) salta si CX es diferente de cero mientras una condición de no igual exista. Se saldrá del ciclo si la condición es igual o si el registro CX se decrementa a cero.

Como las instrucciones de repetición de ciclo, existen instrucciones alternativas para LOOPE y LOOPNE. La instrucción LOOPE es la misma que LOOPZ y la LOOPNE es la misma que LOOPNZ. En la mayoría de los programas solo se aplican LOOPE y LOOPNE.