Manual de instrucciones de la familia x86-32/x86-64
Instrucciones del nivel de aplicación
Instrucciones de enteros

Índice

1
Símbolos utilizados

reg8 Registro de propósito general de 8 bits: AL, AH, BL, BH, CL, CH, DL, DH, SPL, BPL, SIL, DIL, R8B, R9B, R10B, R11B, R12B, R13B, R14B, R15B.

reg16 Registro de propósito general de 16 bits: AX, BX, CX, DX, BP, SP, SI, DI, R8W, R9W, R10W, R11W, R12W, R13W, R14W, R15W.

reg32 Registro de propósito general de 32 bits: EAX, EBX, ECX, EDX, EBP, ESP, ESI, EDI, R8D, R9D, R10D, R11D, R12D, R13D, R14D, R15D.

reg64 Registro de propósito general de 64 bits: RAX, RBX, RCX, RDX, RBP, RSP, RSI, RDI, R8, R9, R10, R11, R12, R13, R14, R15.

reg Cualquiera de los registros indicados por reg8, reg16, reg32 y reg64.

regseg Registro de segmento: CS, DS, SS, ES, FS, GS.

inm8 Constante inmediata de 8 bits.

inm16 Constante inmediata de 16 bits.

inm32 Constante inmediata de 32 bits.

inm64 Constante inmediata de 32 bits.

inm Constante inmediata de 8, 16, 32 o 64 bits.

mem8 Operando de memoria que referencia a un dato de 8 bits.

mem16 Operando de memoria que referencia a un dato de 16 bits.

mem32 Operando de memoria que referencia a un dato de 32 bits.

mem64 Operando de memoria que referencia a un dato de 64 bits.

mem128 Operando de memoria que referencia a un dato de 128 bits.

mem Operando de memoria que referencia a un dato de 8, 16, 32 o 64 bits.

rel8 Desplazamiento relativo inmediato de 8 bits.

rel16 Desplazamiento relativo inmediato de 16 bits.

rel32 Desplazamiento relativo inmediato de 32 bits.

ptr16:16 Puntero lejano inmediato de 32 bits formado por una parte de segmento de 16 bits y parte de offset de 16 bits.

ptr16:32 Puntero lejano inmediato de 48 bits formado por una parte de selector de segmento de 16 bits y una parte de offset de 32 bits.

mem16:16 Operando de memoria que referencia a un puntero lejano de 32 bits formado por una parte de segmento de 16 bits y parte de offset de 16 bits.

mem16:32 Operando de memoria que referencia a un puntero lejano de 48 bits formado por una parte de selector de segmento de 16 bits y una parte de offset de 32 bits.

mem16&16 Operando de memoria que referencia a dos palabras almacenadas consecutivamente.

mem32&32 Operando de memoria que referencia a dos dobles palabras almacenadas consecutivamente.

- Las instrucciones condicionales se agrupan dando la raíz común del mnemotécnico seguida por *cc*, siendo *cc* una secuencia de letras que representan la condición que se comprueba. La sustitución de *cc* por las distintas combinaciones de letras que expresan cada condición dará lugar a las distintas instrucciones.
- En la sección indicadores se usarán los siguientes símbolos para describir como la instrucción afecta a los indicadores:

Símbolo	**Significado**
0	La instrucción pone a cero el indicador.
1	La instrucción pone a uno el indicador
?	La instrucción afecta al indicador de forma indeterminada.
–	La instrucción no afecta al indicador.
↕	La instrucción afecta al indicador poniéndolo a cero o a uno dependiendo del resultado de su ejecución y de acuerdo con la función del indicador.

2
Descripción de las instrucciones

AAA

Función Ajuste ASCII tras la suma.

Sintaxis AAA

Descripción Ajusta la suma de dos números BCD desempaquetados para dar un resultado correcto. También puede utilizarse para otras conversiones BCD. El registro AL actúa como operando fuente implícito y AL y AH como registros destino implícitos.

La instrucción funciona de la siguiente manera: si el nibble bajo de AL es mayor de 9 o si el indicador AF está activado se suma 6 a AL, AH se incrementa en una unidad y los indicadores CF y AF se ponen a 1. En caso contrario ni AL ni AH se alteran y tanto CF como AF se ponen a 0. El nibble alto de AL se pone a 0 en todos los casos.

Esta instrucción no es válida en modo de 64 bits.

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
?	–	–	–	?	?	↕	?	↕

AAD

Función Ajuste ASCII antes de la división.

Sintaxis AAD

Descripción Ajusta las dos cifras BCD desempaquetadas almacenadas en AX (AH cifras más significativa y AL cifra menos significativa) de forma que una instrucción de división posterior dé un resultado BCD desempaquetado correcto. La instrucción se utiliza antes de una instrucción de división que divida A entre una cifra BCD desempaquetada.

También puede utilizarse para convertir dos cifras BCD desempaquetadas almacenadas en AL y AH en un dato entre 0 y 99 equivalente en AL.

La instrucción realiza las operaciones:

```
AL = 10*AH + AL
AH = 0
```

El código máquina de la instrucción AAM es 0xD5 0x0A, donde 0xD5 es el código de operación y 0x0A es la base de ajuste. Realmente, puede conseguirse que la instrucción realice el ajuste en otra base distinta a la decimal indicándola en el segundo byte. Por ejemplo, la instrucción máquina 0xD5 0x08 realiza el ajuste de AL en base octal. Sin embargo, los ensambladores siempre traducen la instrucción AAD por los códigos 0xD5 0x0A por lo que para realizar el ajuste en una base distinta a la decimal es necesario dar su código máquina directamente, por ejemplo, con la directiva DB:

```
DB 0xD5, 0x08
```

Esta instrucción no es válida en modo de 64 bits.

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
?	–	–	–	↕	↕	?	↕	?

AAM

Función Ajuste ASCII tras la multiplicación.

Sintaxis AAM

Descripción La multiplicación de dos números BCD desempaquetados produce un resultado que, en general, no es válido interpretado como número BCD desempaquetado ya que las instrucciones de multiplicación realizan una multiplicación binaria. La instrucción AAM se puede utilizar para obtener un resultado correcto compuesto por dos cifras BCD en el registro AX (AH contendrá la cifra más significativa y AL la cifra menos significativa).

También puede emplearse para convertir un dato entre 0 y 99 almacenado en AL en dos cifras BCD desempaquetadas en AH y AL. La instrucción almacena en AH y AL el cociente entero y resto, respectivamente, de la división AL/10:

```
AL = AL/10
AH = AL mod 10
```

El código máquina de la instrucción AAM es 0xD4 0x0A, donde 0xD4 es el código de operación y 0x0A es la base de ajuste. Realmente, puede conseguirse que la instrucción realice el ajuste en otra base distinta a la decimal indicándola en el segundo byte. Por ejemplo, la instrucción máquina 0xD4 0x08 realiza el ajuste de AL en base octal. Sin embargo, los ensambladores siempre traducen la instrucción AAM por los códigos 0xD4 0x0A por lo que para realizar el ajuste en una base distinta a la decimal es necesario dar su código máquina directamente, por ejemplo, con la directiva DB:

```
DB 0xD4, 0x08
```

Esta instrucción no es válida en modo de 64 bits.

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
?	–	–	–	↕	↕	?	↕	?

AAS

Función Ajuste ASCII tras la resta.

Sintaxis AAS

Descripción Ajusta la resta de dos números BCD desempaquetados para dar un resultado correcto. El registro AL actúa como operando fuente implícito y AL y AH como registros destino implícitos. La instrucción funciona de la siguiente manera: si el nibble bajo de AL es mayor de 9 o si el indicador AF está activado se resta 6 a AL, AH se decrementa en una unidad y los indicadores CF y AF se ponen a 1. En caso contrario ni AL ni AH se alteran y tanto CF como AF se ponen a 0. En todos los casos el nibble alto de AL se pone a 0.

Esta instrucción no es válida en modo de 64 bits.

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
?	–	–	–	?	?	↕	?	↕

ADC

Función Sumar con acarreo.

Sintaxis ADC *op1*, *op2*

Descripción Suma los contenidos de los operandos *op1* y *op2* y de CF y almacena el resultado en el operando *op1*.

Si *op2* es una constante inmediata con un tamaño inferior al de *op1*, la constante se extiende en signo al tamaño de *op1*.

Formas

```
ADC reg8/mem8, imm8
ADC reg16/mem16, imm8   ; imm8 se extiende en signo a 16 bits
ADC reg32/mem32, imm8   ; imm8 se extiende en signo a 32 bits
ADC reg64/mem64, imm8   ; imm8 se extiende en signo a 64 bits
ADC reg16/mem16, imm16
ADC reg32/mem32, imm32
ADC reg64/mem64, imm32  ; imm32 se extiende en signo a 64 bits
ADC reg8/mem8, reg8
ADC reg16/mem16, reg16
ADC reg32/mem32, reg32
ADC reg64/mem64, reg64
ADC reg8, mem8
ADC reg16, mem16
ADC reg32, mem32
ADC reg64, mem64
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
↕	–	–	–	↕	↕	↕	↕	↕

ADD

Función Sumar.

Sintaxis ADD *op1*, *op2*

Descripción Suma los contenidos de los operandos *op1* y *op2* y almacena el resultado en el operando *op1*.

Si *op2* es una constante inmediata con un tamaño inferior al de *op1*, la constante se extiende en signo al tamaño de *op1*.

Formas

```
ADD reg8/mem8, imm8
ADD reg16/mem16, imm8   ; inm8 se extiende en signo a 16 bits
ADD reg32/mem32, imm8 ; inm8 se extiende en signo a 32 bits
ADD reg64/mem64, imm8   ; imm8 se extiende en signo a 64 bits
ADD reg16/mem16, imm16
ADD reg32/mem32, imm32
ADD reg64/mem64, imm32  ; imm32 se extiende en signo a 64 bits
ADD reg8/mem8, reg8
ADD reg16/mem16, reg16
ADD reg32/mem32, reg32
ADD reg64/mem64, reg64
ADD reg8, mem8
ADD reg16, mem16
ADD reg32, mem32
ADD reg64, mem64
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
↕	–	–	–	↕	↕	↕	↕	↕

AND

Función Operación lógica AND.

Sintaxis AND *op1*, *op2*

Descripción Realiza la operación lógica AND (producto lógico) bit a bit entre los operandos *op1* y *op2* y deja el resultado en *op1*.

Si *op2* es una constante inmediata con un tamaño inferior al de *op1*, la constante se extiende en signo al tamaño de *op1*.

Formas

```
AND reg8/mem8, imm8
AND reg16/mem16, imm8 ; inm8 se extiende en signo a 16 bits
AND reg32/mem32, imm8   ; inm8 se extiende en signo a 32 bits
AND reg64/mem64, imm8   ; imm8 se extiende en signo a 64 bits
AND reg16/mem16, imm16
AND reg32/mem32, imm32
AND reg64/mem64, imm32  ; imm32 se extiende en signo a 64 bits
AND reg8/mem8, reg8
AND reg16/mem16, reg16
AND reg32/mem32, reg32
AND reg64/mem64, reg64
AND reg8, mem8
AND reg16, mem16
AND reg32, mem32
AND reg64, mem64
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
0	–	–	–	↕	↕	?	↕	0

ANDN

Función Operación lógica AND NOT.

Sintaxis ANDN *op1*, *op2*, *op3*

Descripción Realiza la operación lógica AND bit a bit entre el operando *op3* y el complemento a uno del operando *op2* y almancena el resultado en el operando *op1*.

La instrucción realiza una operación equivalente a:

```
NOT tmp, op2
AND op1, tmp, op3
```

Los indicadores cambian según el resultado de la operación AND.

Formas

```
ANDN reg32, reg32, reg32/mem32
ANDN reg64, reg64, reg64/mem64
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
0	–	–	–	↕	↕	?	?	0

BEXTR

Función Extraer campo de bits.

Sintaxis BEXTR *op1*, *op2*, *op3*

Descripción Extrae del operando *op2* un campo de bits especificado por el operando *op3* y lo coloca en los bits menos significativos del operando *op1*. El resto de bits del operando *op1* se ponen a 0. El operando *op2* no resulta modificado.

El operando *op3* especifica los bits a extraer de *op2* de la siguiente manera:

- Los bits 0 a 7 especifican el índice del primer bit a extraer.
- Los bits 8 a 15 especifican el número de bits a extraer.

Formas
```
BEXTR reg32, reg32/mem32, reg32/inm32
BEXTR reg64, reg64/mem64, reg64/inm32
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
0	–	–	–	?	↕	?	?	0

BLCFILL

Función Rellenar desde el bit a cero más bajo.

Sintaxis BLCFILL *op1*, *op2*

Descripción Toma el dato en el operando *op2* y encuentra el bit a cero menos significativo de dicho dato, pone a cero todos los bits de por debajo de ese y escribe el resultado en el operando *op1*. Si el dato en el operando *op2* no tiene ningún bit a cero, todos los bits del operando *op1* se ponen a cero. El operando *op2* no resulta modificado.

La instrucción realiza una operación equivalente a:

```
ADD tmp, op2, 1
AND op1, tmp, op2
```

El indicador de acarreo cambia según la operación ADD y los demás según la operación AND.

Formas
```
BLCFILL reg32, reg32/mem32
BLCFILL reg64, reg64/mem64
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
0	–	–	–	↕	↕	?	?	↕

BLCI

Función Aislar el bit a cero más bajo.

Sintaxis BLCI *op1*, *op2*

Descripción Toma el dato en el operando *op2* y encuentra el bit a cero menos significativo de dicho dato, pone a uno todos los demás bits y escribe el resultado en el operando *op1*. Si el dato en el operando *op2* no tiene ningún bit a cero, todos los bits del operando *op1* se ponen a uno. El operando *op2* no resulta modificado.

La instrucción realiza una operación equivalente a:

```
ADD tmp, op2, 1
NOT tmp, tmp
OR  op1, tmp, op2
```

El indicador de acarreo cambia según la operación ADD y los demás según la operación OR.

Formas
```
BLCI reg32, reg32/mem32
BLCI reg64, reg64/mem64
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
0	–	–	–	↕	↕	?	?	↕

BLCIC

Función Aislar bit a cero más bajo y complementar.

Sintaxis BLCIC *op1*, *op2*

Descripción Toma el dato en el operando *op2*, encuentra el bit a cero menos significativo de dicho dato, lo pone a uno, pone a cero todos los demás bits y escribe el resultado en el operando *op1*. Si el dato en el operando *op2* no tiene ningún bit a cero, todos los bits del operando *op1* se ponen a cero. El operando *op2* no resulta modificado.

La instrucción realiza una operación equivalente a:

```
ADD tmp1, op2, 1
NOT tmp2, op2
AND op1, tmp1, tmp2
```

El indicador de acarreo cambia según la operación ADD y los demás según la operación AND.

Formas
```
BLCIC reg32, reg32/mem32
BLCIC reg64, reg64/mem64
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
0	–	–	–	↕	↕	?	?	↕

BLCMSK

Función Enmascarar desde el bit a cero más bajo.

Sintaxis BLCMSK *op1*, *op2*

Descripción Toma el dato en el operando *op2*, encuentra el bit a cero menos significativo de dicho dato, lo pone a uno, pone a cero todos los bits por encima de ese y escribe el resultado en el operando *op1*. Si el dato en el operando *op2* no tiene ningún bit a cero, todos los bits del operando *op1* se ponen a cero. El operando *op2* no resulta modificado.

La instrucción realiza una operación equivalente a:

```
ADD tmp, op2, 1
XOR op1, tmp, op2
```

El indicador de acarreo cambia según la operación ADD y los demás según la operación XOR.

Formas

```
BLCMSK reg32, reg32/mem32
BLCMSK reg64, reg64/mem64
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
0	–	–	–	↕	↕	?	?	↕

BLCS

Función Poner a uno el bit a cero más bajo.

Sintaxis BLCS *op1*, *op2*

Descripción Toma el dato en el operando *op2*, encuentra el bit a cero menos significativo de dicho dato, lo pone a uno y escribe el resultado en el operando *op1*. Si el dato en el operando *op2* no tiene ningún bit a cero, el operando *op2* se copia en el operando *op1* y el indicador de acarreo se pone a uno. El operando *op2* no resulta modificado.

La instrucción realiza una operación equivalente a:

```
ADD tmp, op2, 1
OR  op1, tmp, op2
```

El indicador de acarreo cambia según la operación ADD y los demás según la operación OR.

Formas

```
BLCS reg32, reg32/mem32
BLCS reg64, reg64/mem64
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
0	–	–	–	↕	↕	?	?	↕

BLSFILL

Función Rellenar desde el bit a uno más bajo.

Sintaxis BLSFILL *op1*, *op2*

Descripción Toma el dato en el operando *op2* y encuentra el bit a uno menos significativo de dicho dato, pone a uno todos los bits de por debajo de ese y escribe el resultado en el operando *op1*. Si el dato en el operando *op2* no tiene ningún bit a uno, todos los bits del operando *op1* se ponen a uno. El operando *op2* no resulta modificado.

La instrucción realiza una operación equivalente a:

```
SUB tmp, op2, 1
OR  op1, tmp, op2
```

El indicador de acarreo cambia según la operación SUB y los demás según la operación OR.

Formas

```
BLSFILL reg32, reg32/mem32
BLSFILL reg64, reg64/mem64
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
0	–	–	–	↕	↕	?	?	↕

BLSI

Función Aislar bit a uno más bajo.

Sintaxis BLSI *op1*, *op2*

Descripción Toma el dato en el operando *op2* y encuentra el bit a uno menos significativo de dicho dato, pone a cero todos los demás bits y escribe el resultado en el operando *op1*. Si el dato en el operando *op2* no tiene ningún bit a uno, todos los bits del operando *op1* se ponen a cero. El operando *op2* no resulta modificado.

La instrucción realiza una operación equivalente a:

```
NEG tmp, op2
AND op1, tmp, op2
```

El indicador de acarreo cambia según la operación NEG y los demás según la operación AND.

Formas

```
BLCI reg32, reg32/mem32
BLCI reg64, reg64/mem64
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
0	–	–	–	↕	↕	?	?	↕

BLSIC

Función Aislar bit a uno más bajo y complementar.

Sintaxis BLSIC *op1*, *op2*

Descripción Toma el dato en el operando *op2*, encuentra el bit a uno menos significativo de dicho dato, lo pone a cero, pone a uno todos los demás bits y escribe el resultado en el operando *op1*. Si el dato en el operando *op2* no tiene ningún bit a uno, todos los bits del operando *op1* se ponen a uno. El operando *op2* no resulta modificado.

La instrucción realiza una operación equivalente a:

```
SUB tmp1, op2, 1
NOT tmp2, op2
OR  op1, tmp1, tmp2
```

El indicador de acarreo cambia según la operación SUB y los demás según la operación OR.

Formas

```
BLSIC reg32, reg32/mem32
BLSIC reg64, reg64/mem64
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
0	–	–	–	↕	↕	?	?	↕

BLSMSK

Función Enmascarar desde el bit a uno más bajo.

Sintaxis BLSMSK *op1*, *op2*

Descripción Toma el dato en el operando *op2*, encuentra el bit a uno menos significativo de dicho dato, pone a uno todos los bits por debajo de ese y escribe el resultado en el operando *op1*. Si el dato en el operando *op2* no tiene ningún bit a uno, todos los bits del operando *op1* se ponen a uno. El operando *op2* no resulta modificado.

La instrucción realiza una operación equivalente a:

```
SUB tmp, op2, 1
XOR op1, tmp, op2
```

El indicador de acarreo cambia según la operación SUB y los demás según la operación XOR.

Formas
```
BLSMSK reg32, reg32/mem32
BLSMSK reg64, reg64/mem64
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
0	–	–	–	↕	↕	?	?	↕

BLSR

Función Poner a cero el bit a uno más bajo.

Sintaxis BLSR *op1*, *op2*

Descripción Toma el dato en el operando *op2*, encuentra el bit a uno menos significativo de dicho dato, lo pone a cero y escribe el resultado en el operando *op1*. Si el dato en el operando *op2* no tiene ningún bit a uno, el operando *op2* se copia en el operando *op1* y el indicador de acarreo se pone a uno. El operando *op2* no resulta modificado.

La instrucción realiza una operación equivalente a:

```
SUB tmp, op2, 1
AND  op1, tmp, op2
```

El indicador de acarreo cambia según la operación SUB y los demás según la operación AND.

Formas
```
BLSR reg32, reg32/mem32
BLSR reg64, reg64/mem64
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
0	–	–	–	↕	↕	?	?	↕

BOUND

Función Comparar índice de array con límites.

Sintaxis BOUND *op1*, *op2*

Descripción Determina si el índice con signo de un array contenido en *op1* está entre los límites especificados por el operando *op2*. El operando *op1* debe ser un registro de 16 o 32 bits mientras que el operando *op2* es un operando de memoria que señala a dos palabras (si *op1* es un registro de 16 bits) o dos dobles palabras (si *op1* es un registro de 32 bits) colocadas consecutivamente en memoria. De estas dos palabras o dobles palabras, la primera es el límite inferior permitido al índice y la segunda el límite superior. El índice debe ser mayor o igual al límite inferior y menor o igual al límite superior. Si el índice está fuera de los límites la instrucción provocará una excepción de rango de BOUND excedido (#BR).

Esta instrucción no es válida en modo de 64 bits.

Formas
```
BOUND reg16, mem16&mem16
BOUND reg32, mem32&mem32
```

Indicadores No afectados.

BSF

Función Explorar bits hacia delante.

Sintaxis BSF *op1*, *op2*

Descripción La instrucción BSF explora los bits del operando *op2* desde el menos significativo hasta el más significativo y almacena en el operando *op1* la posición del primer bit a uno que encuentra. El operando *op1* debe ser un registro. El operando *op2* puede ser un registro o un dato almacenado en memoria. Si el contenido de *op2* es cero (no tiene ningún bit a uno), el contenido de *op1* queda indefinido y ZF se pone a 1. Si el operando *op2* es distinto de cero ZF se pone a 0.

Formas
```
BSF reg16, reg16/mem16
BSF reg32, reg32/mem32
BSF reg64, reg64/mem64
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
–	–	–	–	–	↕	–	–	–

BSR

Función Explorar bits hacia atrás.

Sintaxis BSR *op1*, *op2*

Descripción La instrucción BSR explora los bits del operando *op2* desde el más significativo hasta el menos significativo y almacena en el operando *op1* la posición del primer bit a uno que encuentra. El operando *op1* debe ser un registro. El operando *op2* puede ser un registro o un dato almacenado en memoria. Si el contenido de *op2* es cero (no tiene ningún bit a uno), el contenido de *op1* queda indefinido y ZF se pone a 1. Si *op2* es distinto de cero ZF se pone a 0.

Formas
```
BSR reg16, reg16/mem16
BSR reg32, reg32/mem32
BSR reg64, reg64/mem64
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
–	–	–	–	–	↕	–	–	–

BSWAP

Función Intercambiar bytes.

Sintaxis BSWAP

Descripción Invierte el orden de los bytes de un registro. Si el registro es de 32 bits, los bits 0 a 7 se intercambian con los bits 24 a 31 y los bits 8 a 15 se intercambian con los bits 16 a 23. Si el registro es de 64 bits, los bits 7 a 0 se intercambian con los bits 63 a 56, los bits 15 a 8 con los bits 55 a 48, los bits 23 a 16 con los bits 47 a 40 y los bits 31 a 24 con los bits 39 a 32.

Esta instrucción permite convertir una doble palabra o una cuádruple palabra del formato big endian al formato little endian y viceversa.

Formas
```
BSWAP reg32
BSWAP reg64
```

Indicadores No afectados.

BT

Función Comprobar bit.

Sintaxis BT *bit_base*, *bit_offset*

Descripción La instrucción BT selecciona dentro de la cadena de bits cuyo LSB es el bit 0 del operando *bit_base* la posición de bit designada por el operando *bit_offset* y copia su estado en CF. El operando *bit_offset* se interpreta como el desplazamiento del bit al que se accede relativo al bit 0 de *bit_base*. El operando *bit_base* puede ser un registro o una posición de memoria. El operando *bit_offset* puede ser un registro o un dato inmediato.

Si el operando *bit_base* es un registro, el valor de *bit_offset* módulo 16, 32 ó 64 (dependiendo del tamaño del registro) designa al bit dentro del registro.

Si el operando *bit_base* es una posición de memoria y el operando *bit_offset* es un registro, puede seleccionarse cualquier bit en el rango desde -2^{15} hasta $2^{15}-1$, si el operando *bit_offset* es una registro de 16 bits, desde -2^{31} hasta $2^{31}-1$, si es un registro de 32 bits, o desde -2^{63} hasta $2^{63}-1$, si es un registro de 64 bits. Por tanto, el bit accedido puede encontrarse en una posición de memoria distinta a la indicada directamente por *bit_base*.

Si el operando *bit_base* es una posición de memoria y el operando *bit_offset* es una constante inmediata, el índice del bit designado es el valor de la constante módulo 16, 32 ó 64 (dependiendo del tamaño del operando de memoria).

Formas

```
BT reg16/mem16, reg16
BT reg32/mem32, reg32
BT reg64/mem64, reg64
BT reg16/mem16, inm8
BT reg32/mem32, inm8
BT reg64/mem64, inm8
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
?	–	–	–	?	?	?	?	↕

BTC

Función Comprobar bit y complementarlo.

Sintaxis BTC *bit_base*, *bit_offset*

Descripción La instrucción BTC selecciona dentro de la cadena de bits cuyo LSB es el bit 0 del operando *bit_base* la posición de bit designada por el operando *bit_offset*, copia su estado en CF y a continuación complementa el bit de la cadena. El operando *bit_offset* se interpreta como el desplazamiento del bit al que se accede relativo al bit 0 de *bit_base*. El operando *bit_base* puede ser un registro o una posición de memoria. El operando *bit_offset* puede ser un registro o un dato inmediato.

Si el operando *bit_base* es un registro, el valor de *bit_offset* módulo 16, 32 ó 64 (dependiendo del tamaño del registro) designa al bit dentro del registro.

Si el operando *bit_base* es una posición de memoria y el operando *bit_offset* es un registro, puede seleccionarse cualquier bit en el rango desde -2^{15}

hasta $2^{15}-1$, si el operando *bit_offset* es una registro de 16 bits, desde -2^{31} hasta $2^{31}-1$, si es un registro de 32 bits, o desde -2^{63} hasta $2^{63}-1$, si es un registro de 64 bits. Por tanto, el bit accedido puede encontrarse en una posición de memoria distinta a la indicada directamente por *bit_base*.

Si el operando *bit_base* es una posición de memoria y el operando *bit_offset* es una constante inmediata, el índice del bit designado es el valor de la constante módulo 16, 32 ó 64 (dependiendo del tamaño del operando de memoria).

Formas

```
BTC reg16/mem16, reg16
BTC reg32/mem32, reg32
BTC reg64/mem64, reg64
BTC reg16/mem16, inm8
BTC reg32/mem32, inm8
BTC reg64/mem64, inm8
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
?	–	–	–	?	?	?	?	↕

BTR

Función Comprobar bit y ponerlo a cero.

Sintaxis BTR *bit_base*, *bit_offset*

Descripción La instrucción BTR selecciona dentro de la cadena de bits cuyo LSB es el bit 0 del operando *bit_base* la posición de bit designada por el operando *bit_offset*, copia su estado en CF y a continuación pone a 0 el bit de la cadena. El operando *bit_offset* se interpreta como el desplazamiento del bit al que se accede relativo al bit 0 de *bit_base*. El operando *bit_base* puede ser un registro o una posición de memoria. El operando *bit_offset* puede ser un registro o un dato inmediato.

Si el operando *bit_base* es un registro, el valor de *bit_offset* módulo 16, 32 ó 64 (dependiendo del tamaño del registro) designa al bit dentro del registro.

Si el operando *bit_base* es una posición de memoria y el operando *bit_offset* es un registro, puede seleccionarse cualquier bit en el rango desde -2^{15} hasta $2^{15}-1$, si el operando *bit_offset* es una registro de 16 bits, desde -2^{31} hasta $2^{31}-1$, si es un registro de 32 bits, o desde -2^{63} hasta $2^{63}-1$, si es un registro de 64 bits. Por tanto, el bit accedido puede encontrarse en una posición de memoria distinta a la indicada directamente por *bit_base*.

Si el operando *bit_base* es una posición de memoria y el operando *bit_offset* es una constante inmediata, el índice del bit designado es el valor de la constante módulo 16, 32 ó 64 (dependiendo del tamaño del operando de memoria).

A pesar de que el código máquina de la instrucción BTR no lo permita, algunos ensambladores admiten que se especifique un desplazamiento de bit

inmediato mayor de 31 en combinación con un operando *bit_base* de memoria. Para ello el ensamblador ajusta *bit_offset* y el campo de desplazamiento relativo del operando de memoria de forma que se tenga acceso al bit.

Formas

```
BTR reg16/mem16, reg16
BTR reg32/mem32, reg32
BTR reg64/mem64, reg64
BTR reg16/mem16, inm8
BTR reg32/mem32, inm8
BTR reg64/mem64, inm8
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
?	–	–	–	?	?	?	?	↕

BTS

Función Comprobar bit y ponerlo a uno.

Sintaxis BTS *bit_base*, *bit_offset*

Descripción La instrucción BTS selecciona dentro la cadena de bits cuyo LSB es el bit 0 del operando *bit_base* la posición de bit designada por el operando *bit_offset*, copia su estado en CF y a continuación pone a 1 el bit de la cadena. El operando *bit_offset* se interpreta como el desplazamiento del bit al que se accede relativo al bit 0 de *bit_base*. El operando *bit_base* puede ser un registro o una posición de memoria. El operando *bit_offset* puede ser un registro o un dato inmediato.

Si el operando *bit_base* es un registro, el valor de *bit_offset* módulo 16, 32 ó 64 (dependiendo del tamaño del registro) designa al bit dentro del registro.

Si el operando *bit_base* es una posición de memoria y el operando *bit_offset* es un registro, puede seleccionarse cualquier bit en el rango desde -2^{15} hasta $2^{15}-1$, si el operando *bit_offset* es una registro de 16 bits, desde -2^{31} hasta $2^{31}-1$, si es un registro de 32 bits, o desde -2^{63} hasta $2^{63}-1$, si es un registro de 64 bits. Por tanto, el bit accedido puede encontrarse en una posición de memoria distinta a la indicada directamente por *bit_base*.

Si el operando *bit_base* es una posición de memoria y el operando *bit_offset* es una constante inmediata, el índice del bit designado es el valor de la constante módulo 16, 32 ó 64 (dependiendo del tamaño del operando de memoria).

Formas

```
BTS reg16/mem16, reg16
BTS reg32/mem32, reg32
BTS reg64/mem64, reg64
BTS reg16/mem16, inm8
BTS reg32/mem32, inm8
BTS reg64/mem64, inm8
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
?	–	–	–	?	?	?	?	↕

CALL

Función Llamar a procedimiento.

Sintaxis CALL *op1*

Descripción Transferir el control de la ejecución a una subrutina o procedimiento guardando en la pila la dirección de retorno. La dirección de retorno es la dirección de la siguiente instrucción al CALL según la secuencia normal de ejecución (es decir, la que se encuentra a continuación de ella en la memoria).

El operando *op1* proporciona la dirección de memoria en la que comienza el procedimiento llamado. Dicho operando puede ser un dato inmediato, un registro de propósito general o un operando de memoria.

La instrucción CALL se ejecuta de forma diferente en función de que el procedimiento llamado esté o no en el mismo segmento de código que el procedimiento llamador y del modo de operación actual del procesador. Distinguimos así los siguientes tipos de llamada:

- Llamada cercana (near): el procedimiento llamado se encuentra en el mismo segmento de código que la instrucción CALL.
- Llamada lejana (far): el procedimiento llamado se encuentra en un segmento de código distinto a la instrucción CALL.

Formas

```
CALL    rel16
CALL    rel32
CALL    reg16
CALL    mem16
CALL    reg32
CALL    mem32
CALL    ptr16:16
CALL    mem16:16
CALL    ptr16:32
CALL    mem16:32
```

Indicadores Si no se produce conmutación de tarea los indicadores no resultan afectados. Si se produce conmutación de tarea todos los indicadores pueden resultar afectados.

CBW/CWDE/CDQE

Función Convertir extendiendo el signo.

Sintaxis CBW
CWDE
CDQE

Descripción CBW convierte el dato de 8 bits con signo en AL en un dato de 16 bits con signo y lo almacena en AX. Para ello, la instrucción copia el estado del bit de signo de AL (bit 7) en todos los bits de AH. Esta operación se denomina extensión del signo de AL a AX.

CWDE convierte el dato de 16 bits con signo en AX a un dato de 32 bits con signo en EAX mediante la extensión del signo de AX a todo EAX.

CDQE convierte el dato de 32 bits con signo en EAX a un dato de 32 bits con signo en RAX mediante la extensión del signo de EAX a todo RAX.

Los tres mnemotécnicos se refieren al mismo código de operación (98h) el cual actúa sobre AL y AX o AX y EAX dependiendo del tamaño de operandos activo en el momento de ejecutarse la instrucción. El mnemotécnico CDQE sólo es significativo en modo de 64 bits.

Indicadores No afectados.

CLC

Función Poner a cero el indicador de acarreo.

Sintaxis CLC

Descripción Pone a 0 el indicador de acarreo.

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
–	–	–	–	–	–	–	–	0

CLD

Función Poner a cero el indicador de dirección.

Sintaxis CLD

Descripción Pone a 0 el indicador de dirección (DF). Esto hace que las instrucciones de cadena procesen datos avanzando hacia direcciones de memoria crecientes.

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
–	0	–	–	–	–	–	–	–

CLFLUSH

Función Varias una línea de caché.

Sintaxis CLFLUSH *mem8*

Descripción Vacía la línea de caché especificada por la dirección lineal del operando *mem8*. La instrucción comprueba todos los niveles de la jerarquía de memorias caché internas y externas e invalida la línea en todas las cachés en las que se encuentre. Si una caché contiene una copia modificada (*dirty*) de la línea, la línea se escribe en la memoria principal antes de ser invalidada.

Indicadores No afectados.

CMC

Función Complementar el indicador de acarreo.

Sintaxis CMC

Descripción Complementa el indicador de acarreo, es decir, si el indicador de acarreo estaba a cero lo pone a uno y si estaba a uno lo pone a cero.

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
–	–	–	–	–	–	–	–	↕

CMOVcc

Función Mover condicionalmente.

Sintaxis CMOV*cc destino*, *fuente*

Descripción Si la condición expresada por *cc* es cierta, la instrucción copia el contenido del operando *fuente* en el operando *destino*. Si la condición *cc* es falsa la instrucción no realiza ninguna transferencia de datos y se sigue la ejecución con la siguiente instrucción. Los operandos *fuente* y *destino* deben ser del mismo tamaño.

La condición comprobada se refiere siempre al estado actual de uno o más indicadores y se expresa mediante una o más letras cada una de las cuales tiene un significado. El significado de cada letra es el siguiente:

Letra(s) en *cc*	**Significado**
C	Acarreo (Carry)
Z	Cero (Zero)
S	Signo (Sign)
O	Desbordamiento (Overflow)
P	Paridad (Parity)
PO	Paridad impar (Parity Odd)
PE	Paridad par (Parity Even)
E	Igual (Equal)
N	No
A	Por encima (Above). Para números sin signo
B	Por debajo (Below). Para números sin signo
G	Mayor que (Greater). Para números con signo
L	Menor que (Less). Para números con signo

Un primer grupo de instrucciones comprueban un sólo indicador:

Instrucción	**Mover si...**
CMOVC	El indicador de acarreo está a 1
CMOVNC	El indicador de acarreo está a 0
CMOVZ	El indicador de cero está a 1
CMOVNZ	El indicador de cero está a 0
CMOVO	El indicador de desbordamiento está a 1
CMOVNO	El indicador de desbordamiento está a 0
CMOVS	El indicador de signo está a 1 (1)
CMOVNS	El indicador de signo está a 0 (2)
CMOVP/CMOVPE	El indicador de paridad está a 1/si paridad par (3)
CMOVNP/CMOVPO	El indicador de paridad está a 0/si paridad impar (4)

(1)Si el indicador de signo (SF) está a 1 indica que el signo es negativo
(2)Si el indicador de signo (SF) está a 0 indica que el signo es positivo
(3)Si el indicador de paridad (PF) está a 1 indica que la paridad es par
(4)Si el indicador de paridad (PF) está a 0 indica que la paridad es impar

Otro grupo de instrucciones expresan en su mnemotécnico cual es la relación que debe existir entre el primer y segundo operando de una instrucción CMP ejecutada antes de la instrucción CMOVcc (inmediatamente antes o de forma que los indicadores aritméticos no hayan cambiado entre ambas) para que la transferencia de datos se realice:

```
CMP             op1, op2
CMOV<relación> destino,fuente
```

La instrucción CMP *op1*, *op2* compara los operandos *op1* y *op2* restándolos y ajustando los indicadores según el resultado de la resta (el resultado se desecha). A continuación la instrucción CMOV¡relación¿consulta el estado de los indicadores para comprobar si la expresión *op1* ¡relacion¿*op2* tiene valor de verdad. Si se cumple la relación, la instrucción realiza la copia del operando fuente en el operando destino; si no se cumple, no hace nada.

Puesto que una misma relación entre operandos puede ser expresada a veces de dos formas distintas, algunas instrucciones presentan dos mnemotécnicos alternativos entre los que el programador puede elegir según el contexto.

En la descripción de las instrucciones, los términos mayor que y menor que se reservan para el caso de números con signo, mientras que para el caso de números sin signo se emplean los términos por encima y por debajo.

Instruccíón	**Mover si...**
CMOVA/CMOVNBE	Por encima/si no por debajo ni igual (números sin signo)
CMOVB/CMOVNAE	Por debajo/si no por encima ni igual (números sin signo)
CMOVBE/CMOVNA	Por debajo o igual/si no por encima (números sin signo)
CMOVAE/CMOVNB	Por encima o igual/si no por debajo (números sin signo)
CMOVG/CMOVNLE	Mayor/si no menor o igual (números con signo)
CMOVL/CMOVNGE	Menor/si no mayor o igual (números con signo)
CMOVNG/CMOVLE	No mayor/si menor o igual (números con signo)
CMOVNL/CMOVGE	No menor/si mayor o igual (números con signo)
CMOVE	Iguales (números con o sin signo)
CMOVNE	No iguales (números con o sin signo)

La siguiente tabla resume todo lo anterior.

Relación	**Números sin signo**		**Números con signo**	
	Instrucción	**Flags**	**Instrucción**	**Flags**
$op1 > op2$	CMOVA/CMOVNBE	$\overline{CF} \cdot \overline{ZF}$	CMOVG/CMOVNLE	$\overline{ZF} \cdot \overline{SF \oplus OF}$
$op1 < op2$	CMOVB/CMOVNAE	CF	CMOVL/CMOVNGE	$SF \oplus OF$
$op1 \geq op2$	CMOVAE/CMOVNB	$\overline{CF}$	CMOVGE/CMOVNL	$\overline{SF \oplus OF}$
$op1 \leq op2$	CMOVBE/CMOVNA	CF + ZF	CMOVLE/CMOVNG	$ZF + (SF \oplus OF)$
$op1 = op2$	CMOVE	ZF	CMOVE	ZF
$op1 \neq op2$	CMOVNE	$\overline{ZF}$	CMOVNE	$\overline{ZF}$

Formas

```
CMOVcc  reg16, reg16/mem16
CMOVcc  reg32, reg32/mem32
CMOVcc  reg64, reg64/mem64
```

Indicadores No afectados.

CMP

Función Comparar.

Sintaxis CMP *op1*, *op2*

Descripción Resta del contenido del operando *op1* el del operando *op2* afectando a los indicadores pero desechando el resultado. Por tanto, el operando *op1* no resulta modificado.

Si *op2* es una constante inmediata con un tamaño inferior al de *op1*, la constante se extiende en signo al tamaño de *op1*.

Formas

```
CMP reg8/mem8, imm8
CMP reg16/mem16, imm8   ; imm8 se extiende en signo a 16 bits
CMP reg32/mem32, imm8   ; imm8 se extiende en signo a 32 bits
CMP reg64/mem64, imm8   ; imm8 se extiende en signo a 64 bits
CMP reg16/mem16, imm16
```

```
CMP reg32/mem32, imm32
CMP reg64/mem64, imm32  ; imm32 se extiende en signo a 64 bits
CMP reg8/mem8, reg8
CMP reg16/mem16, reg16
CMP reg32/mem32, reg32
CMP reg64/mem64, reg64
CMP reg8, mem8
CMP reg16, mem16
CMP reg32, mem32
CMP reg64, mem64
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
↕	–	–	–	↕	↕	↕	↕	↕

CMPS/CMPSB/CMPSW/CMPSD/CMPSQ

Función Comparar cadenas de bytes/palabras/dobles palabras/cuádruples palabras.

Sintaxis
CMPS *cadena1*, *cadena2*
CMPSB
CMPSW
CMPSD
CMPSQ

Descripción Compara el byte, la palabra, la doble palabra o la cuádruple palabra en la dirección DS:SI, DS:ESI o DS:RSI con el dato del mismo tamaño situado en la dirección ES:DI, ES:EDI o ES:RDI. La comparación se realiza restando del dato apuntado por DS:SI/DS:ESI/RSI el dato apuntado por ES:DI/ES:EDI/RDI. La resta afecta a los indicadores pero el resultado no se almacena. A continuación, SI/ESI/RSI y DI/EDI/RDI se actualizan en base al tamaño de los datos comparados y al estado del indicador de dirección DF. El tamaño de direcciones activo en el momento de ejecutarse la instrucción determina si se usan los registros SI y DI, ESI y EDI o RSI y RDI. El registro DS puede ser sustituido por otro mediante el correspondiente prefijo de cambio de registro de segmento.

CMPSB compara el byte en DS:SI/DS:ESI/RSI con el byte en ES:DI/ES:EDI/RDI. A continuación, si DF = 0, SI/ESI/RSI y DI/EDI/RDI se incrementan en 1, mientras que si DF = 1, SI/ESI/RSI y DI/EDI/RDI se decrementan en 1.

CMPSW compara la palabra en DS:SI/DS:ESI/RSI con la palabra en ES:DI/DS:EDI/RDI. A continuación, si DF = 0, SI/ESI/RSI y DI/EDI/RDI se incrementan en 2, mientras que si DF = 1, SI/ESI/RSI y DI/EDI/RDI se decrementan en 2.

CMPSD compara la doble palabra en DS:SI/DS:ESI/RSI con la doble palabra en ES:DI/ES:EDI/RDI. A continuación, si DF = 0, SI/ESI/RSI y

DI/EDI/RDI se incrementan en 4, mientras que si DF = 1, SI/ESI/RSI y DI/EDI/RDI se decrementan en 4.

CMPSQ compara la cuádruple palabra en DS:SI/DS:ESI/RSI con la cuádruple palabra en ES:DI/ES:EDI/RDI. A continuación, si DF = 0, SI/ESI/RSI y DI/EDI/RDI se incrementan en 8, mientras que si DF = 1, SI/ESI/RSI y DI/EDI/RDI se decrementan en 8.

Si se utiliza la forma con dos operandos explícitos (CMPS cadena1, cadena2), el ensamblador codificará una instrucción CMPSB, CMPSW, CMPSD o CMPSQ según el tamaño de los operandos, los cuales son, normalmente, los nombres de variables declaradas mediante directivas DB, DW, DD o DQ. La directiva indica el tamaño de los datos a comparar.

Puede cambiarse el registro de segmento DS colocando el correspondiente prefijo delante del operando cadena1. El registro ES usado para cadena2 no puede cambiarse. Los operandos tienen además un valor documental al señalar las variables comparadas. Sin embargo, la presencia de los operandos no implica automáticamente que la comparación se realice entre las variables designadas, sino que ésta siempre se realiza con los contenidos de las direcciones DS:SI/DS:ESI/RSI (o *regseg*:SI/*regseg*:ESI/RSI si se cambió DS) y ES:DI/ES:EDI/RDI. Por tanto, es necesario que estos registros apunten a los datos a comparar antes de ejecutar la instrucción.

Puede colocarse un prefijo REP o REPcc delante de estas instrucciones para repetirlas automáticamente CX/ECX/RCX veces o mientras los sucesivos pares de datos comparados sean o no iguales.

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
↕	–	–	–	↕	↕	↕	↕	↕

CMPXCHG

Función Comparar e intercambiar.

Sintaxis CMPXCHG *op1*, *op2*

Descripción La instrucción CMPXCHG compara el valor del registro AL, AX, EAX o RAX con el valor del operando *op1*. Si son iguales, el contenido de *op2* se almacena en *op1*. En caso contrario, el contenido de *op1* se copia en AL, AX, EAX o RAX. Los indicadores se ajustan de acuerdo con el resultado de la comparación. Los operandos *op1* y *op2* deben ser del mismo tamaño.

Si el operando *op1* es un operando de memoria, la instrucción siempre realiza una operación de lectura-modificación-escritura en el operando de memoria. Si los operandos comparados son distintos, CMPXCHG escribe en la memoria el mismo dato que fue leído.

Puede usarse con un prefijo LOCK para la instrucción se ejecute como una operación atómica.

Formas

```
CMPXCHG reg8/mem8, reg8
CMPXCHG reg16/mem16, reg16
CMPXCHG reg32/mem32, reg32
CMPXCHG reg64/mem64, reg64
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
↕	–	–	–	↕	↕	↕	↕	↕

CMPXCHG8B/CMPXCHG16B

Función CMPXCHG8B: Comparar e intercambiar 8 bytes.
CMPXCHG16B: Comparar e intercambiar 16 bytes.

Sintaxis CMPXCHG8B *mem64*
CMPXCHG16B *mem128*

Descripción CMPXCHG8B compara el valor de 64 bits contenido en EDX:EAX con el valor del operando *mem64*, el cual es un dato de 64 bits almacenado en memoria. Si los valores son iguales, el dato de 64 bits almacenado en ECX:EBX se almacena en el operando *mem64*. En caso contrario, el dato contenido en *mem64* se carga en EDX:EAX. En los pares EDX:EAX y ECX:EBX, EDX y ECX contienen los 32 bits más significativos mientras que EAX y ECX mantienen los 32 bits menos significativos. Los indicadores se ajustan de acuerdo con el resultado de la comparación.

CMPXCHG16B compara el valor de 64 bits contenido en RDX:RAX con el valor del operando *mem128*, el cual es un dato de 64 bits almacenado en memoria. Si los valores son iguales, el dato de 64 bits almacenado en RCX:RBX se almacena en el operando *mem128*. En caso contrario, el dato contenido en *mem128* se carga en RDX:RAX. CMPXCHG16B requiere que el operando *mem128* esté alineado. En los pares RDX:RAX y RCX:RBX, RDX y RCX contienen los 64 bits más significativos mientras que RAX y RCX mantienen los 64 bits menos significativos. Los indicadores se ajustan de acuerdo con el resultado de la comparación.

La instrucción puede usarse con un prefijo LOCK para que se ejecute como una operación atómica (otro procesador en el sistema no podrá acceder al operando de memoria durante la ejecución de la instrucción).

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
↕	–	–	–	↕	↕	↕	↕	↕

CPUID

Función Identificación de la CPU.

Sintaxis CPUID

Descripción Proporciona información sobre el procesador en los registros EAX, EBX, ECX y EDX. La información proporcionada depende del valor de EAX antes de ejecutarse la instrucción.

Indicadores No afectados.

CRC32

Función Realizar un paso de comprobación de redundancia cíclica de 32 bits.

Sintaxis CRC32 *op1*, *op2*

Descripción Empezando con un valor inicial en el operando *op1*, acumula valor CRC32 del operando *op2* y almacena el resultado en el operando *op1*. El polinomio CRC32 usado es 0x11EDC6F41. El operando *op1* puede ser un registro de 32 o 64 bits. El operando *op2* puede ser un registro o un dato en memoria. Si *op1* es un registro de 64 bits, el valor inicial son los 32 bits menos significativos y el resultado de 32 bits se guarda en lo 32 bits menos significativos del registro mientras que los 32 bits más significativos se ponen a 0.

Formas
```
CRC32 reg32, reg8/mem8
CRC32 reg32, reg16/mem16
CRC32 reg32, reg32/mem32
CRC32 reg64, reg8/mem8
CRC32 reg64, reg32/mem32
```

Indicadores No afectados.

CWD/CDQ/CQO

Función Convertir palabra en doble palabra/doble palabra en cuádruple palabra.

Sintaxis
CWD
CDQ
CQO

Descripción CWD convierte el dato de 16 bits con signo en AX en un dato de 32 bits con signo y lo almacena en DX:AX. Para ello la instrucción simplemente copia el estado del bit de signo de AX (bit 15) en todos los bits de DX. Esta operación se denomina extensión del signo de AX a DX:AX.

CDQ convierte el dato de 32 bits con signo en EAX a un dato de 64 bits con signo en EDX:EAX mediante la extensión del signo de EAX.

CDO convierte el dato de 64 bits con signo en RAX a un dato de 128 bits con signo en RDX:RAX mediante la extensión del signo de RAX.

Los tres mnemotécnicos se refieren al mismo código de operación (99h) el cual actúa sobre AX y DX:AX, EAX y EDX:EAX o RAX y RDX:RAX dependiendo del tamaño de operandos activo en el momento de ejecutarse la instrucción. El mnemotécnico CQO sólo es significativo en modo de 64 bits.

Indicadores No afectados.

DAA

Función Ajuste decimal tras la suma.

Sintaxis DAA

Descripción Ajusta la suma de dos números BCD empaquetados para dar un resultado correcto. El registro AL actúa como operando fuente y destino implícito.

Al usar una instrucción ADD o ADC para sumar dos números BCD empaquetados el resultado, interpretado como número BCD empaquetado, es, en general, incorrecto, ya que estas instrucciones realizan una suma binaria. En estas circunstancias puede usarse la instrucción DAA a continuación de la instrucción de suma para ajustar el resultado erróneo, que debe estar en AL, y obtener el resultado BCD correcto.

La instrucción funciona de la siguiente manera: si los bits 3-0 de AL representan un número mayor de nueve (xxxx1010-xxxx1111), o si el indicador AF está activado, se suma 6 a AL, obteniéndose la cifra BCD correcta en el nibble inferior. El indicador AF se pondrá a 1. Asimismo, esta suma interna puede activar el indicador CF si se produce un acarreo al sumar los cuatro bits de orden bajo y éste se propaga a través de los bits de orden alto, pero, en caso de que esto no ocurra, CF no se pone a cero sino que conserva su estado anterior.

Si los bits 3-0 de AL representan un número menor o igual a 9 el indicador AF se pone a 0.

Si, en este momento, el indicador de acarreo está activado o si los cuatro bits de orden alto representan un número mayor de 9 (1010xxxx-1111xxxx), dichos bits de orden alto son incrementados en seis unidades (se suma 0x60 a AL), generándose la cifra BCD correcta en el nibble superior. De nuevo, un acarreo procedente de esta suma activará el bit CF, pero en caso contrario el acarreo no se pondrá a cero sino que conservará su estado previo.

Cuando termina la instrucción, el indicador CF señala si la suma de las dos variables BCD originales es mayor que 99, permitiendo la realización de sumas decimales de múltiple precisión.

Esencialmente, la instrucción lleva a cabo el ajuste decimal sumando 0, 6, 0x60 o 0x66 a AL dependiendo del número almacenado inicialmente en el mismo y del estado de los indicadores CF y AF.

Esta instrucción no es válida en modo de 64 bits.

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
?	–	–	–	↕	↕	↕	↕	↕

DAS

Función Ajuste decimal tras la resta.

Sintaxis DAS

Descripción Ajusta la resta de dos números BCD empaquetados para dar un resultado correcto. El registro AL actúa como operando fuente y destino implícito.

Al usar una instrucción SUB o SBB para restar dos números BCD empaquetados el resultado, interpretado como número BCD empaquetado, es, en general, incorrecto ya que estas instrucciones realizan una resta binaria. En estas circunstancias puede usarse la instrucción DAS a continuación de la instrucción de resta para ajustar el resultado erróneo, que debe estar en AL, y obtener el resultado BCD correcto.

La instrucción funciona de la siguiente manera: si los bits 3-0 de AL representan un número mayor de 9 (xxxx1010-xxxx1111), o si el indicador AF está activado, se resta 6 a AL, obteniéndose la cifra BCD correcta en el nibble inferior. El indicador AF se pone a 1. Asimismo, esta resta interna puede activar el indicador CF si se produce un acarreo (debe) al restar los cuatro bits de orden bajo y éste se propaga a través de los bits de orden alto, pero, en caso de que esto no ocurra, CF no se pone a cero sino que conserva su estado anterior.

Si los bits 3-0 de AL representan un número menor o igual a 9 el indicador AF se pone a 0. Si, en este momento, el indicador CF está activado o si los cuatro bits de orden alto representan un número mayor de nueve (1010xxxx-1111xxxx), dichos bits de orden alto son decrementados en seis unidades (se resta 0x60 a AL), generándose la cifra BCD correcta en el nibble superior y el acarreo se pone a 1. En caso contrario el acarreo se pone a 0.

Cuando termina la instrucción, el indicador CF señala si la resta de las dos variables BCD originales es menor que 0, permitiendo la realización de restas decimales de múltiple precisión.

Esta instrucción no es válida en modo de 64 bits.

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
?	–	–	–	↕	↕	↕	↕	↕

DEC

Función Decrementar en 1.

Sintaxis DEC *op1*

Descripción Resta 1 al operando *op1*. El indicador de acarreo no resulta afectado por la operación. Si se quiere que el indicador de acarreo se ajuste de acuerdo con el resultado de la resta debe usarse SUB *op1*, 1.

Formas
```
DEC reg8/mem8
DEC reg16/mem16
DEC reg32/mem32
DEC reg64/mem64
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
↕	–	–	–	↕	↕	↕	↕	–

DIV

Función Dividir sin signo.

Sintaxis DIV *op1*

Descripción Realiza una división entera de números sin signo generando cociente y resto.

Si el operando *op1* es de 8 bits, se divide el operando implícito AX entre *op1*. El cociente se almacena en AL y el resto en AH.

Si el operando *op1* es de 16 bits, se divide el operando implícito DX:AX entre *op1*. El cociente se almacena en AX y el resto en DX.

Si el operando *op1* es de 32 bits, se divide el operando implícito EDX:EAX entre *op1*. El cociente se almacena en EAX y el resto en EDX.

Si el operando *op1* es de 64 bits, se divide el operando implícito RDX:RAX entre *op1*. El cociente se almacena en RAX y el resto en RDX.

Los cocientes no enteros se truncan hacia cero cuando se almacenan en su destino. El resto es siempre menor que el divisor.

Si el cociente no cabe en el destino o si el divisor *op1* es cero se produce una excepción de error de división (#DE) y el cociente y el resto quedan indeterminados.

Los indicadores aritméticos quedan indeterminados en todos los casos.

Formas
```
DIV reg8/mem8
DIV reg16/mem16
DIV reg32/mem32
DIV reg64/mem64
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
?	–	–	–	?	?	?	?	?

ENTER

Función Crear trama de pila de procedimiento.

Sintaxis ENTER *espacio_local*, *nivel_anidamiento*

Descripción ENTER crea la trama de pila de un procedimiento. El primer operando especifica el numero de bytes a ubicar para variables locales. El segundo operando indica el nivel de anidamiento o profundidad del procedimiento en la jerarquía de llamadas y debe ser un número entre 0 y 31. El nivel de anidamiento indicado determina el número de punteros de trama de pila que se copian automáticamente desde la trama de pila del procedimiento llamador a la trama de pila que está siendo creada. Esto permitirá tener acceso a los datos en las tramas de pila de procedimientos superiores en la jerarquía de llamadas.

El atributo de tamaño puntero de pila actual determina si se emplea BP, EBP o RBP como puntero de trama de pila y si se usa SP, ESP o RSP como puntero de pila.

La instrucción ENTER y su compañera LEAVE están pensadas para dar soporte a los lenguajes de alto nivel estructurados en bloques. La instrucción ENTER es típicamente la primera instrucción de un procedimiento y se usa para crear la trama de pila de dicho procedimiento. La instrucción LEAVE se usa al final del procedimiento (justo antes de la instrucción RET) para liberar el espacio usado por la trama de pila.

Frecuentemente ENTER se emplea indicando nivel de anidamiento 0, independientemente del nivel real de anidamiento. En este caso la instrucción:

```
ENTER espacio_para_variables_locales, 0
```

produce el mismo efecto que la secuencia de instrucciones:

```
PUSH  (E/R)BP
MOV   (E/R)BP, (E/R)SP
SUB   (E/R)SP, espacio_para_variables_locales
; La resta anterior no afecta a los indicadores.
```

Formas

```
ENTER inm16, 0
ENTER inm16, 1
ENTER inm16, inm8
```

Indicadores No afectados.

IDIV

Función Dividir con signo.

Sintaxis IDIV *op1*

Realiza una división entera de números con signo generando cociente y resto.

Si el operando *op1* es de 8 bits, se divide el operando implícito AX entre *op1*. El cociente se almacena en AL y el resto en AH.

Si el operando *op1* es de 16 bits, se divide el operando implícito DX:AX entre *op1*. El cociente se almacena en AX y el resto en DX.

Si el operando *op1* es de 32 bits, se divide el operando implícito EDX:EAX entre *op1*. El cociente se almacena en EAX y el resto en EDX.

Si el operando *op1* es de 64 bits, se divide el operando implícito RDX:RAX entre *op1*. El cociente se almacena en RAX y el resto en RDX.

Los cocientes no enteros se truncan hacia cero cuando se almacenan en su estino. El signo del resto es siempre igual al del dividendo. El valor absoluto del resto es siempre menor que el valor absoluto del divisor. La siguiente tabla proporciona las combinaciones de signos posibles para dividendos y divisores de valor absoluto 4 y 3, respectivamente, y los signos de los cocientes y restos resultantes.

Dividendo	Divisor	Cociente	Resto
+4	+3	+1	+1
+4	–3	–1	+1
–4	+3	–1	–1
–4	–3	+1	–1

Si el cociente no cabe en el destino o si el divisor es cero se produce una excepción de error de división (#DE) y el cociente y el resto quedan indeterminados. Los indicadores aritméticos quedan indeterminados en todos los casos.

Formas

```
IDIV    reg8/mem8
IDIV    reg16/mem16
IDIV    reg32/mem32
IDIV    reg64/mem64
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
?	–	–	–	?	?	?	?	?

IMUL

Función Multiplicar con signo.

Sintaxis IMUL *op1*
IMUL *op1*, *op2* IMUL *op1*, *op2*, *op3*

Descripción **Forma con un operando explícito**. Multiplica el operando *op1* de 8, 16, 32 o 64 bits por el operando implícito AL, AX, EAX o RAX respectivamente. Ambos operandos se consideran números enteros con signo. El resultado con signo de 16, 32, 64 o 128 bits se almacena en AX, DX:AX, EDX:EAX o RDX:RAX. Si tras la multiplicación, AH, DX, EDX o RAX (según los operandos sean de 8, 16, 32 o 64 bits) es solamente la extensión del signo de AL, AX, EAX o RAX, respectivamente, los indicadores CF y OF se ponen a 0. En caso contrario se ponen a 1. Es decir, si CF y OF quedan a cero el número de bits significativos del resultado es de 8, 16, 32 o 64 (o menor) y está totalmente contenido en AL, AX, EAX o RAX.

Forma con dos operandos. Multiplica los operandos *op1* y *op2* y deja el resultado en el operando *op1*. Si el resultado de la multiplicación está dentro del rango de números enteros con signo representable en el operando *op1* los indicadores CF y OF se ponen a 0. En caso contrario se ponen a 1. El estado de los indicadores SF, ZF, AF y PF queda indeterminado.

Forma con tres operandos. Multiplica los operandos *op2* y *op3* y deja el resultado en *op1*. Si el resultado de la multiplicación está dentro del rango de números enteros con signo representable en el operando *op1* los indicadores CF y OF se ponen a 0. En caso contrario se ponen a 1. El estado de los indicadores SF, ZF, AF y PF queda indeterminado.

Las tres formas se parecen en que el producto que generan tiene el doble de bits que los operandos fuente. Sin embargo, la forma con un operando explícito almacena el producto completo en el operando destino implícito mientras que las formas con dos y tres operandos truncan el producto al número de bits del operando destino. En estos casos, el indicador de acarreo o desbordamiento puede usarse para comprobar que no se hayan perdido bits significativos.

Las formas con dos y tres operandos pueden utilizarse también para multiplicar números enteros sin signo, ya que la mitad inferior del resultado es igual independientemente de que los operandos se consideren con signo o sin signo. Sin embargo, el estado de los indicadores CF y OF no permitirá conocer si el resultado pudo almacenarse completo en el destino.

Cuando se utiliza un operando fuente inmediato, éste se extiende en signo a la longitud del operando destino *op1*.

Formas Ver la página siguiente

Forma con un operando

```
IMUL reg8/mem8
IMUL reg16/mem16
IMUL reg32/mem32
IMUL reg64/mem64
```

Forma con dos operandos

```
IMUL reg16, reg16/mem16
IMUL reg32, reg32/mem32
IMUL reg64, reg64/mem64
```

Forma con tres operandos

```
IMUL reg16, reg16/mem16, inm8
IMUL reg16, reg16/mem16, inm16
IMUL reg32, reg32/mem32, inm8
IMUL reg32, reg32/mem32, inm32
IMUL reg64, reg64/mem64, inm8
IMUL reg64, reg64/mem64, inm32
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
↕	–	–	–	?	?	?	?	↕

IN

Función Entrada desde puerto.

Sintaxis IN *destino*, *fuente*

Descripción Transfiere un byte, una palabra o una doble palabra desde el puerto especificado por el operando *fuente* hacia el operando *destino*. El operando *fuente* puede ser una constante inmediata de 8 bits o el registro DX. Si se usa una constante inmediata de 8 bits puede accederse a los puertos entre 0 y 0xFF (0 a 255). Si se usa DX puede accederse a todo el rango se direcciones de puerto: entre 0 y 0xFFFF (0 a 65535). El operando *destino* puede ser AL, AX o EAX. Si el dato transferido es un byte se accede al puerto especificado; si es una palabra, al puerto especificado y al siguiente; si se transfiere una doble palabra se accede a cuatro puertos consecutivos a partir del especificado.

Formas

```
IN    AL, inm8
IN    AX, inm8
IN    EAX, inm8
IN    AL, DX
IN    AX, DX
IN    EAX, DX
```

Indicadores No afectados.

INC

Función Incrementar en 1.

Sintaxis INC *op1*

Descripción Suma 1 al operando *op1*. El indicador de acarreo no resulta afectado por la operación. Si se quiere que el indicador de acarreo se ajuste de acuerdo con el resultado de la suma debe usarse ADD *op1*, 1.

Formas

```
INC reg8/mem8
INC reg16/mem16
INC reg32/mem32
INC reg64/mem64
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
↕	–	–	–	↕	↕	↕	↕	–

INS/INSB/INSW/INSD

Función Entrada desde puerto a cadena de bytes/palabras/dobles palabras.

Sintaxis
INS *destino*, DX
INSB
INSW
INSD

Descripción Transfiere un dato desde el puerto de entrada/salida especificado en DX a la posición de memoria direccionada por ES:DI, ES:EDI o RDI (segmento designado por ES y offset dado por DI o EDI). A continuación DI/EDI/RDI se actualiza en base al tamaño del dato almacenado y al estado del indicador de dirección DF. El tamaño de direcciones activo en el momento de ejecutarse la instrucción determina si se usa el registro DI, EDI o RDI. El registro ES no puede cambiarse por otro. En modo de 64 bits ES no se usa.

INSB lee un byte del puerto especificado por DX y lo almacena en la posición de memoria ES:DI/ES:EDI/RDI. A continuación, si DF = 0, DI/EDI/RDI se incrementa en 1, mientras que si DF = 1 (E)DI se decrementa en 1.

INSW lee una palabra del puerto especificado por DX y la almacena en la posición de memoria ES:DI/ES:EDI/RDI y siguiente. A continuación, si DF = 0 DI/EDI/RDI se incrementa en 2 mientras que si DF = 1 DI/EDI/RDI se decrementa en 2.

INSD lee una doble palabra del puerto especificado por DX y la almacena en la posición de memoria ES:DI/ES:EDI/RDI y siguientes. A continuación, si DF = 0 DI/EDI/RDI se incrementa en 4, mientras que si DF = 1 DI/EDI/RDI se decrementa en 4.

Si se utiliza la forma con operandos explícitos (INS *destino*,DX), el ensamblador codificará una instrucción INSB, INSW o INSD según el tamaño del operando destino, el cual es, normalmente, el nombre de una variable declarada mediante una directiva DB, DW o DD. La directiva indica el tamaño del dato a manejar. Los operandos tienen además un valor documental al señalar la variable en donde de almacena el dato. Sin embargo, la presencia de los operandos no asegura que el dato se almacene automáticamente en la variable que el primero de ellos designa. El dato siempre se almacena en la dirección ES:(E)DI independientemente del operando. Por tanto, es necesario que estos registros apunten a la posición de memoria destino antes de ejecutar la instrucción.

Si el dato transferido es un byte se accede al puerto especificado en DX; si es una palabra, al puerto especificado en DX y al siguiente; y si se transfiere una doble palabra se accede a cuatro puertos consecutivos a partir del especificado en DX.

Puede utilizarse un prefijo de REP delante de una de estas instrucciones para repetirlas automáticamente (E)CX veces.

Indicadores No afectados.

INT n

Función Llamar a procedimiento de interrupción.

Sintaxis INT *n*

Descripción La instrucción INT *n* genera una llamada a la subrutina o procedimiento manejador de interrupción designado por el operando *n*. El operando *n* es una constante inmediata de 8 bits sin signo (entre 0 y 255) que recibe el nombre de número de vector de interrupción. El número de vector de interrupción se interpreta como un índice dentro de la tabla de vectores de interrupción, si el procesador está trabajando en modo real, o dentro de la tabla de descriptores de interrupción (IDT) si el procesador está trabajando en modo protegido. Los primeros 32 vectores de interrupción están reservados por Intel para el uso del propio procesador.

Indicadores Los indicadores IF, TF, NT, AC, RF y VM pueden cambiar dependiendo del modo de operación del procesador. Si, en modo protegido, la interrupción usa una puerta de tarea cualquier indicador puede cambiar.

INTO

Función Llamar a procedimiento de interrupción 4 si overflow.

Descripción La instrucción INTO provoca una llamada al procedimiento de interrupción 4 si el indicador de desbordamiento, OF, está a 1 (ver la instrucción INT *n*).

Indicadores Los indicadores IF, TF, NT, AC, RF y VM pueden cambiar dependiendo del modo de operación del procesador. Si, en modo protegido, la interrupción usa una puerta de tarea cualquier indicador puede cambiar.

Jcc

Función Saltar condicionalmente.

Sintaxis J*cc rel*

Descripción Si la condición expresada por *cc* es falsa, la instrucción no realiza ninguna operación y se continúa la ejecución con la siguiente instrucción. Si la condición *cc* es cierta, la instrucción realiza un salto, tomándose la siguiente instrucción a partir de la dirección indicada.

La dirección a la que la instrucción salta en caso de que la condición sea cierta se indica mediante el operando *rel*. El operando *rel* es una constante inmediata con signo que se suma, en tiempo de ejecución, al puntero de instrucción IP/EIP/RIP de forma que éste quede apuntando a la instrucción destino del salto; *rel* recibe por ello el nombre de desplazamiento relativo. En

un procesador anterior al 80386 *rel* sólo puede tener 8 bits. En un procesador 80386 o superior existen codificaciones adicionales que permiten especificar un desplazamiento relativo de 16 bits o de 32 bits. En este último caso, el tamaño por defecto de los desplazamientos depende del atributo de tamaño de operandos del segmento de código actual, el cual puede ser cambiado mediante el prefijo 0x66. El ensamblador elige normalmente el tamaño de desplazamiento relativo que conduce a una instrucción más corta. Los rangos permitidos para los saltos según el tamaño del desplazamiento relativo son los rangos de números enteros con signo representables en 8, 16 y 32 bits:

Desplazamiento	**Rango**
rel8	-128 a +127
rel16	-32768 a +32767
rel32	-2147483648 a +2147483647

Es necesario recordar que, cuando finaliza la decodificación de una instrucción, el puntero de instrucción IP/EIP/RIP está ya señalando al primer byte de la siguiente instrucción almacenada en memoria. Por tanto, el desplazamiento relativo se cuenta a partir del primer byte de la siguiente instrucción a la propia instrucción de salto.

Normalmente, el desplazamiento relativo no se calcula manualmente. En su lugar, se indica una etiqueta que identifica la instrucción a la que debe saltarse si la condición se cumple. Dicha instrucción estará precedida de la misma etiqueta seguida de dos puntos.

El ensamblador se encargará de calcular el desplazamiento relativo. Ejemplo:

```
        ...
        jz  Salto
        ...
        ...
Salto:  mov ah,5
        ...
```

La condición comprobada se refiere siempre al estado actual de uno o más indicadores y se expresa mediante una o más letras cada una de las cuales tiene un significado. El significado de cada letra es el siguiente:

Letra(s) en *cc*	**Significado**
C	Acarreo (Carry)
Z	Cero (Zero)
S	Signo (Sign)
O	Desbordamiento (Overflow)
P	Paridad (Parity)
PO	Paridad impar (Parity Odd)
PE	Paridad par (Parity Even)
E	Igual (Equal)
N	No
A	Por encima (Above). Para números sin signo
B	Por debajo (Below). Para números sin signo
G	Mayor que (Greater). Para números con signo
L	Menor que (Less). Para números con signo

Un grupo de instrucciones comprueban un sólo indicador. Sus mnemotécnicos señalan el indicador comprobado y si debe encontrarse a 0 o a 1 para que el salto se produzca:

Instrucción	**Saltar si...**
JC	El indicador de acarreo está a 1
JNC	El indicador de acarreo está a 0
JZ	El indicador de cero está a 1
JNZ	El indicador de cero está a 0
JO	El indicador de desbordamiento está a 1
JNO	El indicador de desbordamiento está a 0
JS	El indicador de signo está a 1 (1)
JNS	El indicador de signo está a 0 (2)
JP/JPE	El indicador de paridad está a 1/si paridad par (3)
JNP/JPO	El indicador de paridad está a 0/si paridad impar (4)

(1) Si el indicador de signo (SF) está a 1 indica que el signo es negativo

(2) Si el indicador de signo (SF) está a 0 indica que el signo es positivo

(3) Si el indicador de paridad (PF) está a 1 indica que la paridad es par

(4) Si el indicador de paridad (PF) está a 0 indica que la paridad es impar

Otro grupo de instrucciones expresan en su mnemotécnico cual es la relación que debe existir entre el primer y segundo operando de una instrucción CMP ejecutada antes de la instrucción Jcc (inmediatamente antes o de forma que los indicadores aritméticos no hayan cambiado entre ambas) para que el salto se lleve a cabo.

```
        CMP         op1,op2
        J<relación> relacion_verdadera
relacion_falsa:
        ...
        ...
relacion_verdadera:
        ...
        ...
```

La instrucción CMP *op1*, *op2* compara los operandos *op1* y *op2* restándolos y ajustando los indicadores sugún el resultado de la resta (el resultado se desecha). A continuación la instrucción J¡relación¿consulta el estado de los indicadores para comprobar si la expresión *op1* ¡relación¿*op2* tiene valor de verdad. Si se cumple la relación, la instrucción salta a la etiqueta relacion_verdadera; si no se cumple, se continúa la ejecución con la siguiente instrucción.

Puesto que una misma relación entre operandos puede ser expresada a veces de dos formas distintas, algunas instrucciones presentan dos mnemotécnicos alternativos entre los que el programador puede elegir según el contexto.

En la descripción de las instrucciones, los términos mayor que y menor que se reservan para el caso de números con signo, mientras que para el caso de números sin signo se emplean los términos por encima y por debajo.

Instruccíón	**Saltar si...**
JA/JNBE	Por encima/no por debajo ni igual (números sin signo)
JB/JNAE	Por debajo/no por encima ni igual (números sin signo)
JBE/JNA	Por debajo o igual/no por encima (números sin signo)
JAE/JNB	Por encima o igual/no por debajo (números sin signo)
JG/JNLE	Mayor/no menor o igual (números con signo)
JL/JNGE	Menor/no mayor o igual (números con signo)
JNG/JLE	No mayor/menor o igual (números con signo)
JNL/JGE	No menor/mayor o igual (números con signo)
JE	Iguales (números con o sin signo)
JNE	No iguales (números con o sin signo)

La siguiente tabla resume todo lo anterior.

Relación	**Números sin signo**		**Números con signo**	
	Instrucción	**Flags**	**Instrucción**	**Flags**
$op1 > op2$	JA/JNBE	$\overline{CF} \cdot \overline{ZF}$	JG/JNLE	$\overline{ZF \cdot \overline{SF \oplus OF}}$
$op1 < op2$	JB/JNAE	CF	JL/JNGE	$SF \oplus OF$
$op1 \geq op2$	JAE/JNB	$\overline{CF}$	JGE/JNL	$\overline{SF \oplus OF}$
$op1 \leq op2$	JBE/JNA	CF + ZF	JLE/JNG	$ZF + (SF \oplus OF)$
$op1 = op2$	JE	ZF	JE	ZF
$op1 \neq op2$	JNE	$\overline{ZF}$	JNE	$\overline{ZF}$

Formas

```
Jcc   rel8
Jcc   rel16
Jcc   rel32
```

Indicadores No afectados.

JCXZ/JECXZ/JRCXZ

Función Saltar si CX/ECX/RCX es cero.

Sintaxis JCXZ *rel8*
JECXZ *rel8*
JRCXZ *rel8*

Descripción Si el contenido del registro CX, ECX o RCX es distinto de cero se continúa la ejecución con la siguiente instrucción. Si el contenido del registro CX, ECX o RCX es cero se realiza un salto a la dirección indicada.

Los tres mnemotécnicos corresponden al mismo código de operación. El registro comprobado por la instrucción depende del modo de operación del procesador.

La dirección a la que la instrucción salta en caso de que CX, ECX o RCX sea cero se indica mediante el operando *rel8*. El operando *rel8* es una constante inmediata de 8 bits con signo que se suma, en tiempo de ejecución, al puntero de instrucción IP, EIP o RIP de forma que este quede apuntando a la instrucción destino del salto; *rel8* recibe por ello el nombre de desplazamiento relativo. Cuando finaliza la decodificación de una instrucción, el puntero de instrucción IP, EIP o RIP está ya señalando al primer byte de la siguiente instrucción almacenada en memoria. Como *rel8* es un dato de 8 bits con signo, se permiten saltos comprendidos entre -128 y +127 posiciones respecto al primer byte de la siguiente instrucción al JCXZ/JECX/JRCX.

Indicadores No afectados.

JMP

Función Saltar.

Sintaxis JMP *op1*

Descripción Transferir el control de la ejecución a un punto distinto. El operando *op1* proporciona la dirección de memoria a la que saltar. Dicho operando puede ser un dato inmediato, un registro de propósito general o un operando de memoria. La instrucción JMP puede realizar tres tipos de saltos:

- Saltos cortos (short) y cercanos (near).
- Saltos lejanos (far) en modo real o modo protegido ejecutando una tarea virtual-8086.
- Saltos lejanos en modo protegido ejecutando una tarea que no es virtual-8086.

Saltos cortos y saltos cercanos. El salto se produce a otro punto dentro del segmento de código actual, por tanto, el registro de segmento CS no cambia. La dirección a la que se salta puede especificarse como un desplazamiento relativo o como un offset absoluto dentro del segmento de código actual.

Cuando el operando de la instrucción es una dato inmediato, éste se interpreta como un desplazamiento relativo de 8, 16 o 32 bits (rel8, rel16 o rel32) que se suma en tiempo de ejecución al puntero de instrucción IP o EIP, dependiendo del tamaño de operandos seleccionado en el momento de ejecutarse la instrucción. El salto se denomina corto cuando el desplazamiento relativo es de 8 bits y cercano en los demás casos. Los rangos para los tres tamaños de desplazamiento son:

Desplazamiento	**Rango**
rel8	-128 a +127
rel16	-32768 a +32767
rel32	-2147483648 a +2147483647

Es necesario recordar que, cuando finaliza la decodificación de una instrucción, el puntero de instrucción IP o EIP está ya señalando al primer byte de la siguiente instrucción almacenada en la memoria. Por tanto, el desplazamiento relativo se cuenta a partir del primer byte de la siguiente instrucción a la propia instrucción de salto.

Normalmente, el desplazamiento relativo no se calcula manualmente. En su lugar, se indica una etiqueta que identifica la instrucción a la que se desea saltar. Dicha instrucción estará precedida de la misma etiqueta seguida de dos puntos. El ensamblador se encargará de calcular el desplazamiento relativo. Ejemplo:

```
        ...
        jmp Salto
        ...
Salto:  sub cx,8
        ...
```

Si el operando de la instrucción es un registro o un operando de memoria, el contenido del registro o posición de memoria se interpreta como el offset de la instrucción a la que debe saltarse por lo que dicho contenido se carga directamente en IP o EIP.

Saltos lejanos en modo real o virtual-8086. El salto se produce a una instrucción situada dentro de otro segmento de código. El operando de la instrucción puede ser un dato inmediato o un operando de memoria. El dato inmediato o el contenido de la posición de memoria se interpreta como un puntero lejano de 32 bits (ptr16:16 o mem16:16) o 48 bits (ptr16:32 o mem16:32) que apunta a la dirección del salto, por lo que dicho puntero se carga en el par CS:IP o CS:EIP. El tamaño del puntero utilizado queda determinado por el tamaño de operandos seleccionado en el momento de ejecutarse la instrucción.

Saltos lejanos en modo protegido no virtual-8086. En este modo un salto lejano puede dar lugar a las siguientes acciones:

- Salto lejano a un segmento de código de conformidad o no de conformidad.

- Salto lejano a través de una puerta de llamada.
- Conmutación de tarea.

Formas

```
JMP   rel8
JMP   rel16
JMP   rel32
JMP   reg16
JMP   mem16
JMP   reg32
JMP   mem32
JMP   reg64
JMP   mem64
JMP   ptr16:16
JMP   ptr16:32
JMP   mem16:16
JMP   mem16:32
```

Indicadores Si no se produce conmutación de tarea los indicadores no resultan afectados. Si se produce conmutación de tarea todos los indicadores pueden resultar afectados.

LAHF

Función Cargar AH con byte bajo de FLAGS.

Sintaxis LAHF

Descripción Carga AH con el byte bajo de FLAGS.

En algunos procesadores de 64 bits, está instrucción no es válida en modo de 64 bits.

Indicadores No afectados.

LDS/LES/LFS/LGS/LSS

Función Cargar puntero lejano.

Sintaxis
LDS *destino*, *fuente*
LES *destino*, *fuente*
LFS *destino*, *fuente*
LGS *destino*, *fuente*
LSS *destino*, *fuente*

Descripción Las instrucciones LDS, LES, LFS, LGS y LSS permiten cargar un registro de segmento y un registro de propósito general con un puntero lejano almacenado en la memoria. El operando *fuente* especifica una dirección de memoria a partir de la cual se encuentra almacenado un puntero de 32, 48 u 80 bits, dependiendo del modo de operación del procesador en el momento de ejecutarse la instrucción. La parte de segmento (o de selector) de 16 bits del puntero se carga en el registro de segmento especificado por el mnemotécnico de la instrucción: DS para LDS, ES para LES, FS para LFS, GS para LGS y SS para LSS. La parte de offset del puntero se carga en el primer operando, el cual puede ser un registro de 16, 32 o 64 bits.

Además de cargarse el puntero lejano también se carga en la parte oculta del registro de segmento información adicional procedente del descriptor de segmento designado por la parte de selector del puntero.

Formas

```
LDS reg16, mem16:16
LES reg16, mem16:16
LFS reg16, mem16:16
LGS reg16, mem16:16
LSS reg16, mem16:16

LDS reg32, mem16:32
LES reg32, mem16:32
LFS reg32, mem16:32
LGS reg32, mem16:32
LSS reg32, mem16:32

LDS reg64, mem16:64
LES reg64, mem16:64
LFS reg64, mem16:64
LGS reg64, mem16:64
LSS reg64, mem16:64
```

Indicadores No afectados.

LEA

Función Cargar dirección efectiva.

Sintaxis LEA *destino*, *fuente*

Descripción La instrucción LEA calcula la dirección efectiva del operando *fuente* y la almacena en el operando *destino*. El operando *fuente* debe ser una dirección de memoria especificada mediante uno de los modos de direccionamiento del procesador. El operando *destino* debe ser un registro de propósito general de 16, 32 o 64 bits.

Si la dirección tiene el mismo tamaño que el registro de destino la dirección se almacena tal cual.

Si la dirección tiene un tamaño mayor que el registro de destino la dirección se trunca al tamaño del registro.

Si la dirección tiene un tamaño inferior que el registro de destino la dirección se extiende con ceros al tamaño del registro.

Formas

```
LEA reg16, mem
LEA reg32, mem
LEA reg64, mem
```

Indicadores No afectados.

LEAVE

Función Eliminar trama de pila de procedimiento.

Sintaxis LEAVE

Descripción LEAVE libera el espacio ocupado por la trama de pila creada por la instrucción ENTER. El atributo de tamaño puntero de pila actual determina si se emplea BP, EBP o RBP como puntero de trama de pila y si se usa SP, ESP o RSP como puntero de pila.

LEAVE equivale a la siguiente secuencia de instrucciones:

```
MOV (E/R)SP, (E/R)BP
POP (E/R)BP
```

Indicadores No afectados.

LFENCE

Función Barrera de carga.

Sintaxis LFENCE

Descripción Actúa como una barrera para forzar a que todas las instrucciones precedentes a LFENCE que conlleven la carga de datos desde la memoria se completen antes que las instrucciones de carga siguientes a LFENCE.

Formas

```
LFENCE
```

Indicadores No afectados.

LODS/LODSB/LODSW/LODSD/LODSQ

Función Cargar desde cadena de bytes/palabras/dobles palabras/cuádruples palabras.

Sintaxis LODS *fuente*
LODSB
LODSW
LODSD
LODSQ

Descripción Carga el dato de tipo byte, palabra, doble o cuádruple palabra situado en la dirección DS:SI, DS:ESI o RSI en el registro AL, AX, EAX y RAX, respectivamente. A continuación SI/ESI/RSI se actualiza en base al tamaño del dato cargado y al estado del indicador de dirección DF. El tamaño de direcciones activo en el momento de ejecutarse la instrucción determina si se usa el registro SI, ESI o RSI. El registro DS puede ser sustituido por otro mediante el correspondiente prefijo de cambio de registro de segmento.

LODSB carga el byte en la posición de memoria DS:SI/DS:ESI/RSI en el registro AL. A continuación, si DF = 0, SI/ESI/RSI se incrementa en 1, mientras que si DF = 1, SI/ESI/RSI se decrementa en 1.

LODSW carga la palabra en la posición de memoria DS:SI/DS:ESI/RSI en el registro AX. A continuación, si DF = 0, SI/ESI/RSI se incrementa en 2, mientras que si DF = 1, SI/ESI/RSI se decrementa en 2.

LODSD carga la doble palabra en la posición de memoria DS:SI/DS:ESI/RSI en el registro EAX. A continuación, si DF = 0, SI/ESI/RSI se incrementa en 4, mientras que si DF = 1, SI/ESI/RSI se decrementa en 4.

LODSQ carga la cuádruple palabra en la posición de memoria RSI en el registro RAX. A continuación, si DF = 0, RSI se incrementa en 8, mientras que si DF = 1, RSI se decrementa en 8. Esta instrucción sólo puede usarse en modo de 64 bits.

Si se utiliza la forma con un operando explícito (LODS fuente), el ensamblador codificará una instrucción LODSB, LODSW, LODSD o LODSQ según el tamaño del operando fuente, el cual es, normalmente, el nombre de una variable declarada mediante una directiva DB, DW, DD o DQ. La directiva indica el tamaño del dato a cargar. Puede cambiarse el registro de segmento usado por la instrucción colocando el correspondiente prefijo delante del operando. El operando fuente tiene además un valor documental al señalar la variable desde la que se carga el dato. Sin embargo, la presencia del operando no asegura que la carga se realice automáticamente desde la variable que éste designa. La carga siempre se realiza desde la dirección DS:SI/DS:ESI/RSI (o *regseg*:(E)SI si se escribió un prefijo *regseg*: delante del operando) con independencia del operando. Por tanto, es necesario que estos registros apunten al dato a cargar antes de ejecutar la instrucción.

Dado que, normalmente, tras cargar un dato desde la memoria con LODS, LODSB, LODSW, LODSD o LODSQ dicho dato se procesa de determinada forma antes de cargar el siguiente, es raro utilizar un prefijo de repetición REP o REPcc con estas instrucciones.

Indicadores No afectados.

LOOP

Función Bucle con contador CX/ECX/RCX.

Sintaxis LOOP *rel8*

Descripción La finalidad de la instrucción LOOP es realizar bucles. El registro CX, ECX o RCX se emplea como contador de bucle. Se usa CX, ECX o RCX dependiendo del atributo de tamaño de direcciones en el momento de ejecutarse la instrucción.

Típicamente, la instrucción es la última de las instrucciones del bucle controlado por ella. Cuando se ejecuta la instrucción, ésta decrementa primero el contador en una unidad. Si, tras el decremento, el contador es cero se termina el bucle y la ejecución continúa con la siguiente instrucción al LOOP. Si el contador no es cero se salta a la instrucción indicada, la cual es normalmente la primera instrucción del bucle. El decremento del contador no afecta a los indicadores.

La dirección a la que la instrucción salta en caso de que el contador no sea cero se indica mediante el operando rel8. El operando rel8 es una constante inmediata de 8 bits con signo que se suma, en tiempo de ejecución, al puntero de instrucción IP, EIP o RIP de forma que éste quede apuntando a la instrucción destino del salto; rel8 recibe por ello el nombre de desplazamiento relativo. Cuando finaliza la decodificación de una instrucción, el puntero de instrucción IP/EIP/RIP está ya señalando al primer byte de la siguiente instrucción almacenada en memoria. Como rel8 es un dato de 8 bits con signo, se permiten saltos comprendidos entre -128 y +127 posiciones respecto al primer byte de la siguiente instrucción al LOOP.

Indicadores No afectados.

LOOPcc

Función Bucle condicional con contador CX/ECX/RCX.

Sintaxis LOOP*cc rel8*

Descripción La instrucción decrementa el registro contador CX, ECX o RCX en una unidad. Si, tras el decremento, el contador es igual a cero o no se cumple la condición *cc* se continúa la ejecución con la siguiente instrucción. Si el contador no es cero y se cumple la condición *cc*, se efectúa un salto sumando a IP, EIP o RIP el desplazamiento relativo de 8 bits con signo rel8. Por tanto, la instrucción puede usarse para repetir una secuencia de instrucciones mientras se cumpla la condición *cc* pero un número máximo de veces dado por el valor inicial del registro contador.

El que se utilice CX, ECX o RCX depende del atributo de tamaño de direcciones en el momento de ejecutarse la instrucción.

Para formar el mnemotécnico de una instrucción particular, *cc* se sustituye por una de las siguientes combinaciones de letras según la condición que se desee comprobar:

E	Igual. Repetir si hay igualdad (ZF = 1)
Z	Cero. Repetir si el indicador de cero está activado (ZF = 1)
NE	No igual. Repetir si no hay igualdad (ZF = 0)
NZ	No cero. Repetir si el indicador de cero está desactivado

Las condiciones E y Z, por una parte, y NE y NZ, por otra, son equivalentes. La selección la realiza el programador dependiendo del contexto.

Indicadores No afectados.

LZCNT

Función Contar el número de ceros por la izquierda.

Sintaxis LZCNT *op1*, *op2*

Descripción Cuenta el número de bits a cero consecutivos comenzando por el bit más significativo del operando *op2* y hasta encontrar el bit a uno de más peso. El resultado de la cuenta se almacena en el operando *op1*. Si el operando *op2* es cero el indicador de acarreo se pone a uno y el número de bits de *op2* se almacena en *op1*. En caso contrario, el indicador de acarreo se pone a cero.

Formas
```
LZCNT reg16, reg16/mem16
LZCNT reg32, reg32/mem32
LZCNT reg64, reg64/mem64
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
?	–	–	–	?	↕	?	?	↕

MFENCE

Función Barrera de memoria.

Sintaxis MFENCE

Descripción Actúa como una barrera para forzar a que todas las instrucciones precedentes a MFENCE que conlleven carga o almacenamiento de datos en memoria se completen antes que las instrucciones de carga o almacenamiento siguientes a MFENCE.

Formas
```
MFENCE
```

Indicadores No afectados.

MOV

Función Mover dato.

Sintaxis MOV *destino*, *fuente*

Descripción Copia el contenido del operando *fuente* en el operando *destino*.

Si el operando fuente es un registro de segmento y el operando destino es un registro de 32 o 64 bits, el valor se extiende con ceros al tamaño del registro destino.

El registro de segmento CS no puede ser el destino de una instrucción MOV.

Formas
```
MOV reg8/mem8, reg8
MOV reg16/mem16, reg16
MOV reg32/mem32, reg32
MOV reg64/mem64, reg64
MOV reg8, reg8/mem8
MOV reg16, reg16/mem16
MOV reg32, reg32/mem32
MOV reg64, reg64/mem64
MOV reg8/mem8, imm8
MOV reg16/mem16, imm16
MOV reg32/mem32, imm32
MOV reg64/mem64, imm32                  ; imm32 se extiende con ceros a 64
```

```
MOV reg64, imm64
MOV reg16/reg32/reg64/mem16, regseg     ; regseg no puede ser CS
MOV regseg, reg16/mem16                 ; regseg no puede ser CS
```

Indicadores No afectados.

MOVBE

Función Mover *big endian*.

Sintaxis MOV *op1*, *op2*

Descripción Carga un registro con un valor desde la memoria o almacena el valor de un registro en la memoria intercambiando el orden de los bytes. Esto permite cambiar de formato *big endian* a *little endian* o viceversa.

Formas

```
MOVBE reg16, mem16
MOVBE reg32, mem32
MOVBE reg64, mem64
MOVBE mem16, reg16
MOVBE mem32, reg32
MOVBE mem64, reg64
```

Indicadores No afectados.

MOVNTI

Función Mover no-temporal.

Sintaxis MOVNTI *op1*, *op2*

Descripción Almacena un registro (operando *op2*) en la memoria (operando *op1*) indicando al procesador que no es probable que este dato vaya a ser usado en breve. El procesador escribirá el dato en la memoria usando una operación de escritura combinada, que evita la escritura en las memorias caché y por tanto la "polución" de éstas.

Formas

```
MOVNTI mem32, reg32
MOVNTI mem64, reg64
```

Indicadores No afectados.

MOVS/MOVSB/MOVSW/MOVSD/MOVSQ

Función Mover cadenas de bytes/palabras/dobles palabras.

Sintaxis
MOVS *destino*, *fuente*
MOVSB
MOVSW
MOVSD
MOVSQ

Descripción Copia el byte, la palabra, la doble palabra o la cuádruple palabra almacenada en la dirección de memoria DS:SI, DS:ESI o RSI en la dirección de memoria ES:DI, ES:EDI o RDI . A continuación, SI/ESI/RSI y DI/EDI/RSI se actualizan en base al tamaño del dato copiado y al estado del indicador de dirección DF. El tamaño de direcciones activo en el momento de ejecutarse la instrucción determina si se usan SI y DI ESI y EDI o RSI y RDI. El registro DS puede ser sustituido por otro mediante el correspondiente prefijo de cambio de registro de segmento.

MOVSB copia el byte en la posición de memoria DS:SI/DS:ESI/RSI en la posición de memoria ES:DI/ES:EDI/RDI. A continuación, si DF = 0, SI/ESI/RSI y DI/EDI/RDI se incrementan en 1, mientras que si DF = 1, SI/ESI/RSI y DI/EDI/RDI se decrementan en 1.

MOVSW copia la palabra en la posición de memoria DS:SI/DS:ESI/RSI y siguiente en la posición de memoria ES:DI/ES:EDI/RDI y siguiente. A continuación, si DF = 0, SI/ESI/RSI y DI/EDI/RDI se incrementan en 2, mientras que si DF = 1, SI/ESI/RSI y DI/EDI/RDI se decrementan en 2.

MOVSD copia la doble palabra en la posición de memoria DS:SI/DS:ESI/RSI y siguientes en la posición de memoria ES:DI/ES:EDI/RDI y siguientes. A continuación, si DF = 0, SI/ESI/RSI y DI/EDI/RDI se incrementan en 4, mientras que si DF = 1, SI/ESI/RSI y DI/EDI/RDI se decrementan en 4.

MOVSQ copia la cuádruple palabra en la posición de memoria RSI y siguientes en la posición de memoria RDI y siguientes. A continuación, si DF = 0, RSI y RDI se incrementan en 8, mientras que si DF = 1, RSI y RDI se decrementan en 8. Esta instrucción sólo es válida en modo de 64 bits.

Si se utiliza la forma con dos operandos explícitos (MOVS *destino*, *fuente*), el ensamblador codificará una instrucción MOVSB, MOVSW, MOVSD MOVSQ según el tamaño de los operandos, los cuales son, normalmente, los nombres de variables declaradas mediante directivas DB, DW, DD o DQ. La directiva indica el tamaño del dato a copiar. Puede cambiarse el registro de segmento DS colocando el correspondiente prefijo delante del operando fuente. El uso de ES para destino no puede cambiarse. Los operandos tienen además un valor documental al señalar las variables sobre las que actúa la instrucción. Sin embargo, la presencia de los operandos no implica automáticamente que la copia se realice desde la variable indicada en el operando fuente a la variable dada en el operando destino. La copia siempre se realiza desde la posición DS:SI/DS:ESI/RSI (o *regseg*:(E)SI si se cambió DS) a la posición ES:DI/ES:EDI/RDI independientemente de los operandos. Por

tanto, es necesario que estos registros apunten a las direcciones fuente y destino antes de ejecutar la instrucción.

Puede utilizarse un prefijo de REP delante de la instrucción para repetirla automáticamente CX/ECX/RCX veces y así copiar de forma eficiente una cadena completa de datos de un lugar a otro de la memoria.

Indicadores No afectados.

MOVSX

Función Mover extendiendo el signo.

Sintaxis MOVSX *destino*, *fuente*

Descripción Copia el valor entero con signo del operando *fuente* (registro o memoria) en el operando *destino* (registro) extendiendo el bit de signo del operando fuente cuando el tamaño de éste es inferior al del operando destino.

Formas

```
MOVSX reg16, reg8/mem8
MOVSX reg32, reg8/mem8
MOVSX reg64, reg8/mem8
MOVSX reg32, reg16/mem16
MOVSX reg64, reg16/mem16
```

Indicadores No afectados.

MOVSXD

Función Mover doble palabra extendiendo el signo.

Sintaxis MOVSXD *op1*, *op2*

Descripción Copia el valor del operando de 32 bits *op2* en el operando de 64 bits *op1* extendiendo el bit de signo de *op2* a los 32 bits más significativos de *op1*.

Esta instrucción sólo es válida en modo de 64 bits.

Formas

```
MOVSXD reg64, reg32/mem32
```

Indicadores No afectados.

MOVZX

Función Mover extendiendo con ceros.

Sintaxis MOVZX *destino*, *fuente*

Descripción Copia el valor entero sin signo del operando *fuente* (registro o memoria) en el operando *destino* (registro) rellenando con ceros los bits más significativos del destino si su tamaño es superior al del operando fuente.

Formas
```
MOVZX reg16, reg8/mem8
MOVZX reg32, reg8/mem8
MOVZX reg64, reg8/mem8
MOVZX reg32, reg16/mem16
MOVZX reg64, reg16/mem16
```

Indicadores No afectados.

MUL

Función Multiplicar sin signo.

Sintaxis MUL *op1*

Descripción Multiplica el operando *op1* de 8, 16, 32 O 64 bits por el operando implícito AL, AX, EAX o RAX, respectivamente. Los operandos se consideran números enteros sin signo. El resultado sin signo de 16, 32, 64 o 128 bits se almacena en AX, DX:AX, EDX:EAX o RDX:RAX, respectivamente. Si, tras la multiplicación, AH, DX, EDX o RAX (según los operandos sean de 8, 16, 32 o 64 bits) es igual a cero, los indicadores CF y OF se ponen a 0. En caso contrario se ponen a 1. El estado de los indicadores SF, ZF, AF y PF queda indeterminado.

Formas
```
MUL reg8/mem8
MUL reg16/mem16
MUL reg32/mem32
MUL reg64/mem64
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
↕	–	–	–	?	?	?	?	↕

NEG

Función Calcular complemento a dos.

Sintaxis NEG *op1*

Descripción Sustituye el contenido de *op1* por su complemento a dos. El cálculo del complemento a dos se realiza restando de cero el valor inicial del operando *op1*. Los indicadores quedan afectados como en el caso de una instrucción SUB que realizase esta operación. El indicador de acarreo se pone siempre a 1 excepto si el valor inicial del operando es 0.

Formas
```
NEG reg8/mem8
NEG reg16/mem16
NEG reg32/mem32
NEG reg64/mem64
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
↕	–	–	–	↕	↕	↕	↕	↕

NOP

Función No operar.

Sintaxis NOP *op1*

Descripción La instrucción NOP no realiza ninguna operación.

Formas
```
NOP
NOP reg16/mem16
NOP reg32/mem32
NOP reg64/mem64
```

Indicadores No afectados.

NOT

Función Calcular complemento a uno.

Sintaxis NOT *op1*

Descripción Sustituye el contenido del operando *op1* por su complemento a uno. Los bits de *op1* resultan invertidos: los bits que estaban a 0 se ponen a 1 y los que estaban a 1 se ponen a 0.

Formas
```
NOT reg8/mem8
NOT reg16/mem16
NOT reg32/mem32
NOT reg64/mem64
```

Indicadores No afectados.

OR

Función Operación lógica OR.

Sintaxis OR *op1*, *op2*

Descripción Realiza la operación lógica OR (suma lógica) bit a bit entre los operandos *op1* y *op2* y deja el resultado en *op1*.

Si *op2* es una constante inmediata con un tamaño inferior al de *op1*, la constante se extiende en signo al tamaño de *op1*.

Formas

```
OR reg8/mem8, imm8
OR reg16/mem16, imm16
OR reg32/mem32, imm32
OR reg64/mem64, imm32   ; imm32 se extiende en signo a 64 bits
OR reg16/mem16, imm8    ; imm8 se extiende en signo a 16 bits
OR reg32/mem32, imm8    ; imm8 se extiende en signo a 32 bits
OR reg64/mem64, imm8    ; imm8 se extiende en signo a 64 bits
OR reg8/mem8, reg8
OR reg16/mem16, reg16
OR reg32/mem32, reg32
OR reg64/mem64, reg64
OR reg8, reg8/mem8
OR reg16, reg16/mem16
OR reg32, reg32/mem32
OR reg64, reg64/mem64
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
0	–	–	–	↕	↕	?	↕	0

OUT

Función Salida por puerto.

Sintaxis OUT *destino*, *fuente*

Descripción Transfiere el byte, la palabra o la doble palabra almacenada en el operando *fuente* al puerto especificado por el operando *destino*. El operando *fuente* puede ser AL, AX o EAX. El operando *destino* puede ser una constante inmediata de 8 bits o el registro DX. Si se usa una constante inmediata de 8 bits puede accederse a los puertos entre 0 y 0xFF. Si se usa DX puede accederse a todo el rango se direcciones de puerto: entre 0000h y 0xFFFF. Si el dato transferido es un byte se accede al puerto especificado; si es una palabra, al puerto especificado y al siguiente; si se transfiere una doble palabra, se accede a cuatro puertos consecutivos a partir del especificado.

Formas

```
OUT   inm8, AL
OUT   inm8, AX
```

```
OUT   inm8, EAX
OUT   DX, AL
OUT   DX, AX
OUT   DX, EAX
```

Indicadores No afectados.

OUTS/OUTSB/OUTSW/OUTSD

Función Salida por puerto de cadena de bytes/palabras/dobles palabras.

Sintaxis OUTS DX, *fuente*
OUTSB
OUTSW
OUTSD

Descripción Transfiere un dato desde la posición de memoria direccionada por DS:SI o DS:ESI (segmento designado por DS y offset dado por SI o ESI) al puerto de entrada/salida especificado en DX a. A continuación (E)SI se actualiza en base al tamaño del dato transferido y al estado del indicador de dirección DF. El tamaño de direcciones activo en el momento de ejecutarse la instrucción determina si se usa el registro SI (tamaño de direcciones de 16 bits) o el ESI (tamaño de direcciones de 32 bits). El registro DS puede cambiarse con un prefijo de cambio de registro de segmento.

OUTSB lee un byte de la posición de memoria DS:(E)SI y lo transfiere al puerto especificado por DX. A continuación, si DF = 0, (E)SI se incrementa en 1, mientras que si DF = 1, (E)SI se decrementa en 1.

OUTSW lee una palabra de la posición de memoria DS:(E)SI y la transfiere al puerto especificado por DX. A continuación, si DF = 0, (E)SI se incrementa en 2, mientras que si DF = 1, (E)SI se decrementa en 2.

OUTSB lee una doble palabra de la posición de memoria DS:(E)SI y la transfiere al puerto especificado por DX. A continuación, si DF = 0, (E)SI se incrementa en 4, mientras que si DF = 1, (E)SI se decrementa en 4.

Si se utiliza la forma con operandos explícitos (OUTS DX, fuente), el ensamblador codificará una instrucción OUTSB, OUTSW o OUTSD según el tamaño del operando fuente, el cual es, normalmente, el nombre de una variable declarada mediante una directiva DB, DW o DD. La directiva indica el tamaño del dato a manejar. Los operandos tienen además un valor documental al señalar la variable de donde se lee el dato. Sin embargo, la presencia de los operandos no asegura que el dato se lea automáticamente en la variable que el segundo de ellos designa. El dato siempre se lee de la dirección DS:(E)SI (o *regseg*:(E)SI si se incluyó un prefijo de cambio de registro de segmento) independientemente de los operandos. Por tanto, es necesario que estos registros apunten a la posición de memoria fuente antes de ejecutar la instrucción.

Si el dato transferido es un byte se accede al puerto especificado en DX; si es una palabra, al puerto especificado en DX y al siguiente; y si se transfiere una doble palabra se accede a cuatro puertos consecutivos a partir del especificado en DX.

Puede utilizarse un prefijo de REP delante de una de estas instrucciones para repetirlas automáticamente (E)CX veces.

Indicadores No afectados.

PAUSE

Función Pausa.

Sintaxis PAUSE

Descripción Mejora las prestaciones al ejecutar un *spin loop*. La instrucción indica al procesador que está en un *spin loop* haciendo que mejore el rendimiento de los accesos a memoria y se reduzca el consumo de energía mientras se ejecuta el *spin loop*.

Formas `PAUSE`

Indicadores No afectados.

POP

Función Extraer dato de la pila.

Sintaxis POP *destino*

Descripción Extrae de la pila una palabra, doble palabra o cuádruple palabra y la almacena en el operando *destino*. A continuación el puntero de pila se incrementa en dos, cuatro u ocho unidades, respectivamente.

El atributo de tamaño de puntero de pila determina si se usa SP, ESP o RSP como puntero de pila.

El atributo de tamaño de operandos determina si se extrae una palabra, doble palabra o cuádruple palabra.

Si se utiliza POP para extraer el contenido de un registro de segmento cuando el tamaño de operandos es de 32 bits, se extrae de la pila una doble palabra cuya palabra baja se almacena en el registro de segmento mientras que la palabra alta se ignora. De esta forma la pila puede permanecer alineada. La instrucción POP CS no es válida.

En modo de 64 bits, el operando *destino* no puede ser un registro de segmento.

En modo de 64 bits, el operando *destino* no puede ser de 32 bits.

Si una instrucción POP *mem* utiliza ESP o RSP como registro base en el cálculo de la dirección efectiva del operando fuente mem, dicho cálculo empleará el valor de ESP o RSP después de que éste sea incrementado.

La instrucción POP SP/ESP/RSP incrementa SP/ESP/RSP y después copia en SP/ESP/RSP el valor en la cima de la antigua pila; SP/ESP/RSP queda por tanto con este valor.

Formas

```
POP reg16/mem16
POP reg32/mem32 ; No puede codificarse en modo de 64 bits
POP reg64/mem64
POP regseg      ; regseg no puede ser CS
                ; No válida en modo de 64 bits
```

Indicadores No afectados.

POPA/POPAD

Función Extraer de la pila los registros de propósito general.

Sintaxis
POPA
POPAD

Descripción Extrae de la pila el contenido de los ocho registros de propósito general. Los dos mnemotécnicos referencian la misma instrucción máquina la cual extrae de la pila los registros de 16 bits o los extendidos de 32 bits dependiendo del tamaño de operandos actual. Si se emplea POPA se extraerán de la pila los registros de 16 bits o de 32 bits dependiendo del tamaño de operandos por defecto. Si se emplea POPAD el ensamblador insertará, cuando sea necesario, un prefijo de cambio de tamaño de operandos por defecto para forzar la extracción de los registros de 32 bits. Esta instrucción suele utilizarse en combinación con PUSHA/PUSHAD.

El orden en el que se extraen los registros es:

Para registros de 16 bits: DI, SI, BP, (SP se ignora), BX, DX, CX, AX.

Para registros de 32 bits: EDI, ESI, EBP, (ESP se ignora), EBX, EDX, ECX, EAX.

El valor para SP o ESP que se extrae de la pila no se almacena en el registro. En su lugar, SP o ESP se incrementa el número adecuado de unidades después de cargar cada registro.

La instrucción POPA/POPAD no es válida en modo de 64 bits.

Indicadores No afectados.

POPCNT

Función Contar población de bits.

Sintaxis POPCNT *op1*, *op2*

Descripción Cuenta el número de bits a uno en el operando *op2* y almacena el resultado en el operando *op1*. Si el operando *op2* es cero el operando *op1* queda a cero y el indicador de cero se activa. En caso contrario el indicador de cero se desactiva.

Formas
```
POPCNT reg16, reg16/mem16
POPCNT reg32, reg32/mem32
POPCNT reg64, reg64/mem64
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
0	–	–	–	0	↕	0	0	0

POPF/POPFD/POPFQ

Función Extraer de la pila el registro FLAGS/EFLAGS/RFLAGS.

Sintaxis POPF
POPFD
POPFQ

Descripción Extrae de la pila una palabra o una doble palabra y la almacena en FLAGS, EFLAGS o RFLAGS, respectivamente. Los tres mnemotécnicos referencian la misma instrucción máquina. Si se usa POPF se extraerá de la pila una palabra, almacenándola en FLAGS, una doble palabra, almacenándola en EFLAGS, o una cuádruple palabra, almacenándola en RFLAGS, dependiendo del tamaño de operandos por defecto actual. Si se usa POPFD o POPFQ el ensamblador insertará, cuando sea necesario, un prefijo adecuado para obligar a que se extraiga EFLAGS o RFLAGS.

Indicadores Ver descripción.

PREFECTH/PREFECTHW

Función Precargar línea de caché de datos L1.

Sintaxis PREFECTH *mem8*
PREFETCHW *mem8*

Descripción Carga una línea de caché completa de 64 bytes alineada que contiene la dirección de memoria especificada por el operando *mem8*. La posición de la dirección de memoria dentro de la línea de caché es irrelevante. Si la línea ya está en la caché o si se produce una falla de memoria, la instrucción se trata como un NOP.

La instrucción PREFETCHW carga la línea de caché especificada y ajusta el estado de la línea de caché a Modificada, en anticipación a escritura de datos en la línea subsiguientes. La instrucción PREFECTH, por contra, típicamente ajusta el estado de la línea a Exclusiva (dependiendo de la implementación).

Indicadores No afectados.

PREFECTHlevel

Función Precargar línea de caché de datos de nivel *level*.

Sintaxis
PREFECTHNTA mem8
PREFECTH0 mem8
PREFECTH1 mem8
PREFECTH2 mem8

Descripción Carga una línea de caché que contiene la dirección de memoria especificada por el operando *mem8* en el nivel de memoria caché especificado:

- PREFECTHNTA: (Non Temporal Access) Carga la línea de caché minimizando la polución de la caché. Está indicado para datos que se usarán sólo una vez y no repetidamente.
- PREFECTH0: Todos los niveles de caché.
- PREFECTH1: Niveles 2 y superior.
- PREFECTH2: Niveles 3 y superior.

Indicadores No afectados.

PUSH

Función Introducir dato en la pila.

Sintaxis PUSH *fuente*

Descripción El puntero de pila SP, ESP o RSP se decrementa en dos unidades, cuatro u ocho unidades dependiendo del tamaño del operando *fuente*. A continuación, el contenido del operando *fuente* se almacena a partir de la dirección apuntada por SS:SP, SS:ESP, o RSP.

El atributo de tamaño de puntero de pila determina si se usa SP, ESP o RSP como puntero de pila.

El atributo de tamaño de operandos determina si se introduce una palabra, doble palabra o cuádruple palabra.

Si se utiliza PUSH para introducir en la pila el contenido de un registro de segmento cuando el tamaño de operandos es de 32 bits, se introduce en la pila una doble palabra en la que la palabra baja es el contenido del registro de segmento y la palabra alta está a cero. De esta forma la pila puede permanecer alineada.

En modo de 64 bits, el operando *fuente* no puede ser un registro de segmento.

En modo de 64 bits, el operando *fuente* no puede ser de 32 bits.

Si se especifica un operando inmediato de 8 bits, el valor del operando se extiende en signo a 16, 32 o 64 bits antes de almacenarse en la pila.

La instrucción PUSH ESP/RSP introduce en la pila el valor de ESP/RSP previo a la ejecución de la instrucción. Si una instrucción PUSH *mem* utiliza ESP/RSP como registro base en el cálculo de la dirección del operando fuente mem, dicho cálculo empleará el valor de ESP/RSP previo a la ejecución de la instrucción.

PUSH SP/ESP/RSP introduce en la pila el valor de SP previo a la ejecución de la instrucción. Sin embargo, en el 8086 original, PUSH SP introduce en la pila el valor que toma SP después de ejecutarse la instrucción, es decir, después de haber sido decrementado en dos unidades.

Formas

```
PUSH reg16/mem16
PUSH reg32/mem32 ; No codificable en modo de 64 bits
PUSH reg64/mem64
PUSH inm8        ; Se extiende en signo a 16, 32 o 64 bits
PUSH inm16
PUSH inm32       ; En modo 64 bits, se extiende en signo a 64 bits
PUSH regseg      ; No válida en modo 64 bits
```

Indicadores No afectados.

PUSHA/PUSHAD

Función Introducir en la pila los registros de propósito general.

Sintaxis
PUSHA
PUSHAD

Descripción Introduce en la pila el contenido de los ocho registros de propósito general. Los dos mnemotécnicos referencian la misma instrucción máquina la cual introduce en la pila los registros de 16 bits o de los extendidos de 32 bits dependiendo del tamaño de operandos actual. Si se emplea PUSHA se introducirán en la pila registros de 16 bits o de 32 bits dependiendo del tamaño de operandos por defecto. Si se emplea PUSHAD el ensamblador insertará, cuando sea necesario, un prefijo de cambio de tamaño de operandos por defecto para forzar la introducción de los registros de 32 bits. Esta instrucción suele utilizarse en combinación con POPA/POPAD.

El orden en el que se introducen los registros es:

Para registros de 16 bits: AX, CX. DX, BX, SP, BP, SI, DI.

Para registros de 32 bits: EAX, ECX, EDX, EBX, ESP, EBP, ESI, EDI.

El valor de SP o ESP que se introduce en la pila es el inmediatamente anterior a la ejecución de la instrucción.

La instrucción PUSHA/PUSHAD no es válida en modo de 64 bits.

Indicadores No afectados.

PUSHF/PUSHFD/PUSHFQ

Función Introducir en la pila el registro FLAGS/EFLAGS/RFLAGS.

Sintaxis
PUSHF
PUSHFD
PUSHFQ

Descripción Introduce en la pila el contenido del registro FLAGS, EFLAGS o RFLAGS. Los tres mnemotécnicos referencian la misma instrucción máquina. Si se usa PUSHF se introducirá en la pila FLAGS, EFLAGS o RFLAGS dependiendo del tamaño de operandos por defecto actual. Si se usa PUSHFD o PUSHFQ el ensamblador insertará, cuando sea necesario, un prefijo para obligar a que se introduzca FLAGS, EFLAGS o RFLAGS.

Indicadores No afectados.

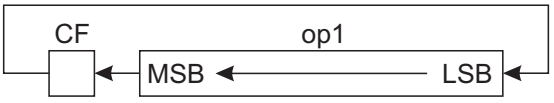
RCL

Función Rotar a la izquierda a través del acarreo.

Sintaxis RCL *op1*, *op2*

Descripción Rota a la izquierda el conjunto de bits formado por los bits de *op1* y el indicador de acarreo, CF, tantas posiciones como indica *op2*. En cada operación de rotación individual cada bit de *op1* se mueve al bit de peso inmediatamente superior mientras que el MSB (bit más significativo) se mueve al indicador de acarreo; el valor anterior del indicador de acarreo se mueve al LSB (bit menos significativo) de *op1*. Esta operación se repite tantas veces como indique *op2* por lo que el indicador de acarreo queda con el valor del último bit en salir de *op1*. Si *op2* es 1 el indicador de desbordamiento, OF, es el OR exclusivo de CF (tras la rotación) y el MSB del resultado (es decir, OF se pone a 1 si *op1* cambió de signo y a 0 en caso contrario). Si *op2* es mayor que 1 OF queda indefinido. Los demás indicadores nunca resultan afectados. Si *op2* es 0 la instrucción no afecta a ningún indicador.

El 8086 no enmascara la cuenta de rotaciones. En cambio, a partir del 80286 sólo se consideran los 5 bits menos significativos de *op2* para evitar rotar más de 31 posiciones. Si el operando *op1* es de 64 bits, sólo se consideran los 6 bits menos significativos de *op2* para evitar rotar más de 63 posiciones.



Formas

```
RCL reg8/mem8, 1
RCL reg16/mem16, 1
RCL reg32/mem32, 1
RCL reg64/mem64, 1
RCL reg8/mem8, imm8
RCL reg16/mem16, imm8
RCL reg32/mem32, imm8
RCL reg64/mem64, imm8
RCL reg8/mem8, CL
RCL reg16/mem16, CL
RCL reg32/mem32, CL
RCL reg64/mem64, CL
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
↕	–	–	–	–	–	–	–	↕

(Ver descripción)

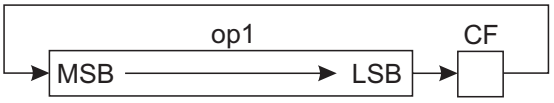
RCR

Función Rotar a la derecha a través del acarreo.

Sintaxis RCR *op1*, *op2*

Descripción Rota a la derecha el conjunto de bits formado por los bits de *op1* y el indicador de acarreo tantas posiciones como indica *op2*. En cada operación de rotación individual cada bit de *op1* se mueve al bit de peso inmediatamente inferior mientras que el LSB (bit menos significativo) se mueve al indicador de acarreo; el valor anterior del indicador de acarreo se mueve al MSB (bit más significativo) de *op1*. Esta operación se repite tantas veces como indique *op2* por lo que el indicador de acarreo queda con el valor del último bit en salir de *op1*. Si *op2* es 1 el indicador de desbordamiento, OF, es el OR exclusivo de los dos bits más significativos del resultado (es decir, OF se pone a 1 si *op1* cambió de signo y a 0 en caso contrario). Si *op2* es mayor que 1 OF queda indefinido. Los demás indicadores nunca resultan afectados. Si *op2* es 0 la instrucción no afecta a ningún indicador.

El 8086 no enmascara la cuenta de rotaciones. En cambio, a partir del 80286 sólo se consideran los 5 bits menos significativos de *op2* para evitar rotar más de 31 posiciones. Si el operando *op1* es de 64 bits, sólo se consideran los 6 bits menos significativos de *op2* para evitar rotar más de 63 posiciones.



Formas

```
RCR reg8/mem8, 1
RCR reg16/mem16, 1
RCR reg32/mem32, 1
RCR reg64/mem64, 1
RCR reg8/mem8, imm8
RCR reg16/mem16, imm8
RCR reg32/mem32, imm8
RCR reg64/mem64, imm8
RCR reg8/mem8, CL
RCR reg16/mem16, CL
RCR reg32/mem32, CL
RCR reg64/mem64, CL
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
0	–	–	–	–	–	–	–	↕

(Ver descripción)

RDFSBASE/RDGSBASE

Función Leer base de FS/leer base de GS.

Sintaxis RDFSBASE *op1*
RDGSBASE *op1*

Descripción Copia el campo de base del registro FS o GS en el operando *op1*, el cual debe ser un registro de 32 o 64 bits. Estas instrucciones solo son válidas en modo de 64 bits.

Formas
```
RDFSBASE reg32
RDFSBASE reg64
RDGSBASE reg32
RDGSBASE reg64
```

Indicadores No afectados.

REP

Función Prefijo repetir CX/ECX/RCX veces.

Sintaxis REP *instrucción_de_cadena*

o bien

REP
instrucción_de_cadena

Descripción El prefijo REP puede colocarse delante de una instrucción de cadena para hacer que ésta se repita tantas veces como indique el valor del registro CX/ECX/RCX antes de ejecutarse la instrucción. Se usa CX/ECX/RCX dependiendo del tamaño de direcciones activo en ese momento (16, 32 o 64 bits, respectivamente). El pseudocódigo siguiente resume los efectos de anteponer un prefijo REP a una instrucción de cadena:

```
MIENTRAS CX/ECX/RCX != 0 HACER
    EJECUTAR INSTRUCCIÓN DE CADENA
    CX/ECX/RCX = CX/ECX/RCX - 1  (sin afectar a los indicadores)
FIN MIENTRAS
```

En ensamblador, el prefijo puede colocarse en la misma línea que la instrucción de cadena o en la línea inmediatamente anterior a la misma.

Indicadores El prefijo REP no afectan a los indicadores pero las instrucciones de cadena a las que pueden preceder si pueden afectarlos.

REPcc

Función Prefijo repetir mientras se cumpla condición *cc*.

Sintaxis REP*cc* *instrucción_de_cadena*

o bien

REP*cc*
instrucción_de_cadena

Descripción Los prefijos REP*cc* pueden colocarse delante de una instrucción de cadena para hacer que ésta se repita mientras se cumpla la condición expresada por *cc*. La condición se comprueba tras cada iteración. En cualquier caso, el número máximo de veces que se repite la instrucción de cadena queda determinado por el valor de CX/ECX/RCX antes de su ejecución. Se usa CX/ECX/RCX según el tamaño de direcciones en el momento de ejecutarse la instrucción sea de 16 bits, 32 o 64 bits, respectivamente. Para formar el mnemotécnico de una prefijo particular, *cc* se sustituye por una de las siguiente combinaciones de letras según la condición que se desee comprobar:

E	Igual. Repetir si hay igualdad (si ZF = 1).
Z	Cero. Repetir si el último resultado fue cero (si ZF = 1).
NE	No igual. Repetir si no hay igualdad (si ZF = 0).
NZ	No cero. Repetir si el último resultado no fue cero (si ZF = 0).

Los prefijos REPE y REPZ son equivalentes. Los prefijos REPNE y REPNZ son equivalentes. La selección del mnemotécnico la realiza el programador dependiendo del contexto.

El pseudocódigo siguiente resume la operación de los prefijos REPcc:

```
MIENTRAS CX/ECX/RCX != 0 HACER
    EJECUTAR INSTRUCCIÓN DE CADENA
    CX/ECX/RCX = CX/ECX/RCX - 1  (sin afectar a los indicadores)
    SI NO SE CUMPLE cc SALIR DEL BUCLE
FIN MIENTRAS
```

Cuando sea necesario saber la causa que terminó la repetición (no se cumplió la condición, CX/ECX/RCX alcanzó cero o ambas simultáneamente) pueden utilizarse instrucciones JZ/JE, JNZ/JNE o JCXZ/JECXZ/JRCXZ después de la instrucción de cadena.

En ensamblador, el prefijo puede colocarse en la misma línea que la instrucción de cadena o en la línea inmediatamente anterior a la misma.

Indicadores Los prefijos REP*cc* no afectan a los indicadores pero las instrucciones de cadena a las que pueden preceder si pueden afectarlos.

RET

Función Retornar de procedimiento.

Sintaxis RET
RET *inm16*

Descripción Transfiere la ejecución a la dirección de retorno ubicada en la cima de la pila. Para ello la instrucción extrae de la pila un valor para (E)IP o para CS y (E)IP. Normalmente la dirección de retorno fue guardada en la pila por la instrucción CALL que pasó el control al procedimiento actual.

El operando inmediato de 16 bits opcional especifica el número de bytes a eliminar de la cima de la pila una vez extraída la dirección de retorno. Puede emplearse para eliminar los parámetros colados en la pila por el procedimiento llamador.

La instrucción RET puede emplearse para realizar tres tipos de retornos:

- Retorno cercano (near).
- Retorno lejano en modo real o modo protegido ejecutando una tarea virtual-8086.
- Retorno lejano en modo protegido ejecutando una tarea que no es virtual-8086.

Indicadores No afectados.

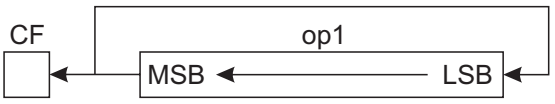
ROL

Función Rotar a la izquierda.

Sintaxis ROL *op1*, *op2*

Descripción Rota a la izquierda los bits de *op1* tantas posiciones como indica *op2*. En cada operación de rotación individual cada bit de *op1* se mueve al bit de peso inmediatamente superior mientras que el MSB (bit más significativo) se mueve al LSB (bit menos significativo) y al indicador de acarreo, CF. Esta operación se repite tantas veces como indique *op2* por lo que el indicador de acarreo queda con el valor del último bit en salir de *op1*. Si el valor de *op2* es 1, el indicador de desbordamiento, OF, es el OR exclusivo de CF (tras la rotación) y el MSB del resultado (es decir, OF se pone a 1 si *op1* cambió de signo y a 0 en caso contrario). Si el valor de *op2* es mayor que 1 OF queda indefinido. Los demás indicadores nunca resultan afectados. Si el valor de *op2* es 0 la instrucción no afecta a ningún indicador.

El 8086 no enmascara la cuenta de rotaciones. En cambio, a partir del 80286 sólo se consideran los 5 bits menos significativos de *op2* para evitar rotar más de 31 posiciones. Si el operando *op1* es de 64 bits, sólo se consideran los 6 bits menos significativos de *op2* para evitar rotar más de 63 posiciones.



Formas

```
ROL reg8/mem8, 1
ROL reg16/mem16, 1
ROL reg32/mem32, 1
ROL reg64/mem64, 1
ROL reg8/mem8, imm8
ROL reg16/mem16, imm8
ROL reg32/mem32, imm8
ROL reg64/mem64, imm8
ROL reg8/mem8, CL
ROL reg16/mem16, CL
ROL reg32/mem32, CL
ROL reg64/mem64, CL
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
↕	–	–	–	–	–	–	–	↕

(Ver descripción)

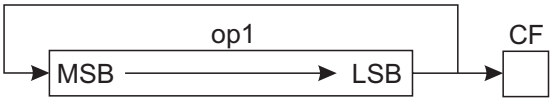
ROR

Función Rotar a la derecha.

Sintaxis ROR *op1*, *op2*

Descripción Rota a la derecha los bits de *op1* tantas posiciones como indica *op2*. En cada operación de rotación individual cada bit de *op1* se mueve al bit de peso inmediatamente inferior, mientras que el LSB (bit menos significativo) se mueve al MSB (bit más significativo) y al indicador de acarreo, CF. Esta operación se repite tantas veces como indique *op2* por lo que el indicador de acarreo queda con el valor del último bit en salir de *op1*. Si el valor de *op2* es 1, el indicador de desbordamiento, OF, es el OR exclusivo de los dos bits más significativos del resultado (es decir, OF se pone a 1 si *op1* cambió de signo y a 0 en caso contrario). Si el valor de *op2* es mayor que 1 OF queda indefinido. Los demás indicadores nunca resultan afectados. Si el valor de *op2* es 0 la instrucción no afecta a ningún indicador.

El 8086 no enmascara la cuenta de rotaciones. En cambio, a partir del 80286 sólo se consideran los 5 bits menos significativos de *op2* para evitar rotar más de 31 posiciones. Si el operando *op1* es de 64 bits, sólo se consideran los 6 bits menos significativos de *op2* para evitar rotar más de 63 posiciones.



Formas

```
ROR reg8/mem8, 1
ROR reg16/mem16, 1
ROR reg32/mem32, 1
ROR reg64/mem64, 1
ROR reg8/mem8, imm8
ROR reg16/mem16, imm8
ROR reg32/mem32, imm8
ROR reg64/mem64, imm8
ROR reg8/mem8, CL
ROR reg16/mem16, CL
ROR reg32/mem32, CL
ROR reg64/mem64, CL
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
↕	–	–	–	–	–	–	–	↕

(Ver descripción)

RORX

Función Rotar a la derecha sin afectar a los indicadores.

Sintaxis RORX *op1*, *op2*, *op3*

Descripción Rota a la derecha los bits de *op2* tantas posiciones como indica *op3* y deposita el resultado en *op1*. En cada operación de rotación individual cada bit de se mueve al bit de peso inmediatamente inferior, mientras que el bit menos significativo se mueve al bit más significativo. Esta operación se repite tantas veces como indique *op3*. La instrucción no afecta a ningún indicador.

Formas

```
RORX reg32, reg32/mem32, inm8
RORX reg64, reg64/mem64, inm8
```

Indicadores No afectados.

SAHF

Función Almacenar AH en byte bajo de FLAGS.

Sintaxis SAHF

Descripción Copia el estado de los bits 0, 2, 4, 6 y 7 de AH en los indicadores CF, PF, AF, ZF y SF del registro FLAGS, respectivamente. Los bits 1, 3 y 5 de AH se ignoran; los correspondientes bits reservados de FLAGS quedan con sus valores previos.

En algunos procesadores de 64 bits, esta instrucción no es válida en modo de 64 bits.

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
				Bit	Bit	Bit	Bit	Bit
–	–	–	–	7	6	4	2	0
				AH	AH	AH	AH	AH

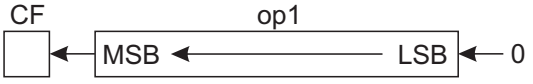
SAL/SHL

Función Desplazamiento aritmético/lógico a la izquierda.

Sintaxis SAL *op1*, *op2*
SHL *op1*, *op2*

Descripción Desplaza a la izquierda los bits de *op1* tantas posiciones como indica *op2*. En cada operación de desplazamiento individual cada bit de *op1* se mueve al bit de peso inmediatamente superior mientras que el MSB (bit más significativo) se mueve al indicador de acarreo. El LSB (bit menos significativo) de *op1* se pone a cero. Esta operación se repite tantas veces como indique *op2* por lo que el indicador de acarreo queda con el valor del último bit en salir de *op1*. Si el valor de *op2* es 1 el indicador de desbordamiento, OF, es el OR exclusivo de CF (tras la rotación) y el MSB del resultado (es decir, OF se pone a 1 si *op1* cambió de signo y a 0 en caso contrario). Si el valor de *op2* es mayor que 1 OF no resulta afectado. Los indicadores SF, ZF y PF cambian de acuerdo al resultado. Si el valor de *op2* no es 0 AF queda indefinido. Si *op2* vale 0 la instrucción no afecta a ningún indicador.

El 8086 no enmascara la cuenta de desplazamientos. En cambio, a partir del 80286 sólo se consideran los 5 bits menos significativos de *op2* para evitar desplazar más de 31 posiciones. Si el operando *op1* es de 64 bits, sólo se consideran los 6 bits menos significativos de *op2* para evitar desplazar más de 63 posiciones.



Formas

```
SAL/SHL reg8/mem8, 1
SAL/SHL reg16/mem16, 1
SAL/SHL reg32/mem32, 1
SAL/SHL reg64/mem64, 1
SAL/SHL reg8/mem8, imm8
SAL/SHL reg16/mem16, imm8
SAL/SHL reg32/mem32, imm8
SAL/SHL reg64/mem64, imm8
SAL/SHL reg8/mem8, CL
SAL/SHL reg16/mem16, CL
SAL/SHL reg32/mem32, CL
SAL/SHL reg64/mem64, CL
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
0	–	–	–	↕	↕	?	↕	↕

(Ver descripción)

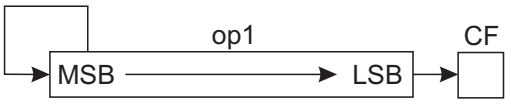
SAR

Función Desplazamiento aritmético a la derecha.

Sintaxis SAR *op1*, *op2*

Descripción Desplaza a la derecha los bits de *op1* tantas posiciones como indica *op2*. En cada operación de desplazamiento individual cada bit de *op1* se mueve al bit de peso inmediatamente inferior mientras que el LSB (bit menos significativo) se mueve al indicador de acarreo. El MSB (bit más significativo o bit de signo) de *op1* no cambia. Esta operación se repite tantas veces como indique *op2* por lo que el indicador de acarreo queda con el valor del último bit en salir de *op1*. Si el valor de *op2* es 1 el indicador de desbordamiento, OF, se pone a 0 (*op1* nunca cambia de signo). Si el valor de *op2* es mayor que 1 OF no resulta afectado. Los indicadores SF, ZF y PF cambian de acuerdo al resultado. Si el valor de *op2* no es 0 AF queda indefinido. Si el valor de *op2* es 0 la instrucción no afecta a ningún indicador.

El 8086 no enmascara la cuenta de desplazamientos. En cambio, a partir del 80286 sólo se consideran los 5 bits menos significativos de *op2* para evitar rotar más de 31 posiciones. Si el operando *op1* es de 64 bits, sólo se consideran los 6 bits menos significativos de *op2* para evitar rotar más de 63 posiciones.



Formas

```
SAR reg8/mem8, 1
SAR reg16/mem16, 1
SAR reg32/mem32, 1
SAR reg64/mem64, 1
SAR reg8/mem8, imm8
SAR reg16/mem16, imm8
SAR reg32/mem32, imm8
SAR reg64/mem64, imm8
SAR reg8/mem8, CL
SAR reg16/mem16, CL
SAR reg32/mem32, CL
SAR reg64/mem64, CL
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
0	–	–	–	↕	↕	?	↕	↕

(Ver descripción)

SARX

Función Rotar a la derecha sin afectar a los indicadores.

Sintaxis SARX *op1*, *op2*, *op3*

Descripción Desplaza a la derecha los bits de *op2* tantas posiciones como indica *op3* y deja el resultado en *op1*. En cada operación de desplazamiento individual cada bit se mueve al bit de peso inmediatamente inferior. El bit más significativo o bit de signo no cambia. Esta operación se repite tantas veces como indique *op3*. La instrucción no afecta a ningún indicador.

Formas
```
SARX reg32, reg32/mem32, inm8
SARX reg64, reg64/mem64, inm8
```

Indicadores No afectados.

SBB

Función Restar con debe.

Sintaxis SBB *op1*, *op2*

Descripción Resta del contenido del operando *op1* el resultado de sumar el contenido del operando *op2* y el acarreo y deja el resultado en el operando *op1*.

Si *op2* es una constante inmediata con un tamaño inferior al de *op1*, la constante se extiende en signo al tamaño de *op1*.

Formas
```
SBB reg8/mem8, imm8
SBB reg16/mem16, imm16
SBB reg32/mem32, imm32
SBB reg64/mem64, imm32  ; imm32 se extiende en signo a 64 bits
SBB reg16/mem16, imm8   ; imm8 se extiende en signo a 16 bits
SBB reg32/mem32, imm8   ; imm8 se extiende en signo a 32 bits
SBB reg64/mem64, imm8   ; imm8 se extiende en signo a 64 bits
SBB reg8/mem8, reg8
SBB reg16/mem16, reg16
SBB reg32/mem32, reg32
SBB reg64/mem64, reg64
SBB reg8, reg8/mem8
SBB reg16, reg16/mem16
SBB reg32, reg32/mem32
SBB reg64, reg64/mem64
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
↕	–	–	–	↕	↕	↕	↕	↕

SCAS/SCASB/SCASW/SCASD/SCASQ

Función Buscar en cadena de bytes/palabras/dobles palabras/cuádruples palabras.

Sintaxis
SCAS *fuente*
SCASB
SCASW
SCASD
SCASQ

Descripción Compara el byte en AL, la palabra en AX, la doble palabra en EAX o la cuádruple palabra en RAX con el dato del mismo tamaño almacenado en la dirección ES:DI/ES:EDI/RDI. La comparación se realiza restando del contenido de AL, AX, EAX o RAX el dato en ES:DI/ES:EDI/RDI. La resta afecta a los indicadores pero el resultado se desecha. A continuación, DI/EDI/RDI se actualiza en base al tamaño de los datos comparados y al estado del indicador de dirección DF. El tamaño de direcciones activo en el momento de ejecutarse la instrucción determina si se usa el registro DI, EDI o RDI. El registro ES no puede sustituirse por otro.

SCASB compara el byte en AL con el byte en ES:DI/ES:EDI/RDI. A continuación, si DF = 0, DI/EDI/RDI se incrementa en 1, mientras que si DF = 1, DI/EDI/RDI se decrementa en 1.

SCASW compara la palabra en AX con la palabra en ES:DI/ES:EDI/RDI. A continuación, si DF = 0, DI/EDI/RDI se incrementa en 2, mientras que si DF = 1, DI/EDI/RDI se decrementa en 2.

SCASD compara la doble palabra en EAX con la doble palabra en ES:DI/ES:EDI/RDI. A continuación, si DF = 0, DI/EDI/RDI se incrementa en 4, mientras que si DF = 1, DI/EDI/RDI se decrementa en 4.

SCASQ compara la doble palabra en RAX con la cuádruple palabra en ES:DI/ES:EDI/RDI. A continuación, si DF = 0, DI/EDI/RDI se incrementa en 8, mientras que si DF = 1, DI/EDI/RDI se decrementa en 8.

Si se utiliza la forma con un operando explícito (SCAS *fuente*), el ensamblador codificará una instrucción SCASB, SCASW, SCASD o SCASQ según el tamaño del operando, el cual es, normalmente, el nombre de una variable declarada una directiva DB, DW, DD o DQ. La directiva indica el tamaño de los datos a comparar. El operando tienen además un valor documental al señalar la variable que se compara con AL, AX, EAX o RAX. Sin embargo, la presencia del operando no asegura que AL, AX, EAX o RAX se compare con la variable designada. La comparación siempre se realiza con el dato almacenado en la dirección ES:DI/ES:EDI/RDI con independencia del operando. Por tanto, es necesario que estos registros apunten al dato apropiado antes de ejecutar la instrucción.

Puede colocarse un prefijo de REP o REPcc delante de estas instrucciones para repetirlas automáticamente CX/ECX/RCX veces o mientras AL, AX, EAX o RAX resulte igual o distinto a los sucesivos datos en memoria.

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
↕	–	–	–	↕	↕	↕	↕	↕

SETcc

Función Poner byte a uno o a cero condicionalmente.

Sintaxis SET*cc destino*

Descripción Si la condición expresada por *cc* es falsa la instrucción carga 0 en el operando *destino*. Si la condición *cc* es cierta la instrucción carga 1 en el operando *destino*. El operando *destino* puede ser un registro de 8 bits o un operando de memoria que designa un dato de 8 bits.

La condición comprobada se refiere siempre al estado actual de uno o más indicadores y se expresa mediante una o más letras cada una de las cuales tiene un significado. El significado de cada letra es el siguiente:

Letra(s) en *cc*	Significado
C	Acarreo (Carry)
Z	Cero (Zero)
S	Signo (Sign)
O	Desbordamiento (Overflow)
P	Paridad (Parity)
PO	Paridad impar (Parity Odd)
PE	Paridad par (Parity Even)
E	Igual (Equal)
N	No
A	Por encima (Above). Para números sin signo
B	Por debajo (Below). Para números sin signo
G	Mayor que (Greater). Para números con signo
L	Menor que (Less). Para números con signo

Un primer grupo de instrucciones comprueban un sólo indicador:

Instrucción	**Poner a 1 si...**
SETC	El indicador de acarreo está a 1
SETNC	El indicador de acarreo está a 0
SETZ	El indicador de cero está a 1
SETNZ	El indicador de cero está a 0
SETO	El indicador de desbordamiento está a 1
SETNO	El indicador de desbordamiento está a 0
SETS	El indicador de signo está a 1 (1)
SETNS	El indicador de signo está a 0 (2)
SETP/SETPE	El indicador de paridad está a 1/si paridad par (3)
SETNP/SETPO	El indicador de paridad está a 0/si paridad impar (4)
En caso contrario poner a 0	

(1) Si el indicador de signo (SF) está a 1 indica que el signo es negativo

(2) Si el indicador de signo (SF) está a 0 indica que el signo es positivo

(3) Si el indicador de paridad (PF) está a 1 indica que la paridad es par

(4) Si el indicador de paridad (PF) está a 0 indica que la paridad es impar

Otro grupo de instrucciones expresan en su mnemotécnico cual es la relación que debe existir entre el primer y segundo operando de una instrucción CMP ejecutada antes de la instrucción SETcc (inmediatamente antes o de forma que los indicadores aritméticos no hayan cambiado entre ambas) para que se cargue 1.

```
CMP           op1, op2
SET<relación> destino
```

La instrucción CMP *op1*, *op2* compara los operandos *op1* y *op2* restándolos y ajustando los indicadores sugún el resultado de la resta (el resultado se desecha). A continuación la instrucción SET¡relación¿consulta el estado de los indicadores para comprobar si la expresión *op1* ¡relación¿*op2* tiene valor de verdad. Si se cumple la relación, la instrucción carga 1 en el operando destino; si no se cumple, se carga 0.

Puesto que una misma relación entre operandos puede ser expresada a veces de dos formas distintas, algunas instrucciones presentan dos mnemotécnicos alternativos entre los que el programador puede elegir según el contexto.

En la descripción de las instrucciones, los términos mayor que y menor que se reservan para el caso de números con signo, mientras que para el caso de números sin signo se emplean los términos por encima y por debajo.

Instruccíón	**Poner a 1 si...**
SETA/SETNBE	Por encima/no por debajo ni igual (números sin signo)
SETB/SETNAE	Por debajo/no por encima ni igual (números sin signo)
SETBE/SETNA	Por debajo o igual/no por encima (números sin signo)
SETAE/SETNB	Por encima o igual/no por debajo (números sin signo)
SETG/SETNLE	Mayor/no menor o igual (números con signo)
SETL/SETNGE	Menor/no mayor o igual (números con signo)
SETNG/SETLE	No mayor/menor o igual (números con signo)
SETNL/SETGE	No menor/mayor o igual (números con signo)
SETE	Iguales (números con o sin signo)
SETNE	No iguales (números con o sin signo)
En caso contrario poner a 0	

La siguiente tabla resume todo lo anterior.

Relación	Números sin signo		Números con signo	
	Instrucción	**Flags**	**Instrucción**	**Flags**
$op1 > op2$	SETA/SETNBE	$\overline{CF} \cdot \overline{ZF}$	SETG/SETNLE	$\overline{ZF} \cdot \overline{SF \oplus OF}$
$op1 < op2$	SETB/SETNAE	CF	SETL/SETNGE	$SF \oplus OF$
$op1 \geq op2$	SETAE/SETNB	$\overline{CF}$	SETGE/SETNL	$\overline{SF \oplus OF}$
$op1 \leq op2$	SETBE/SETNA	CF + ZF	SETLE/SETNG	$ZF + (SF \oplus OF)$
$op1 = op2$	SETE	ZF	SETE	ZF
$op1 \neq op2$	SETNE	$\overline{ZF}$	SETNE	$\overline{ZF}$

Formas `SETcc reg8/mem8`

Indicadores No afectados.

SFENCE

Función Barrera de almacenamiento.

Sintaxis SFENCE

Descripción Actúa como una barrera para forzar a que todas las instrucciones precedentes a SFENCE que conlleven el almacenamiento de datos en memoria se completen antes que las instrucciones de almacenamiento siguientes a SFENCE.

Formas

`SFENCE`

Indicadores No afectados.

SHLD

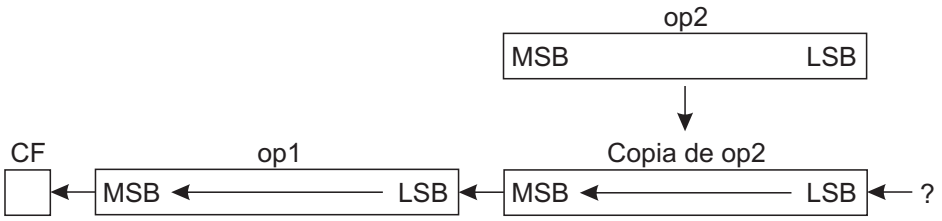
Función Desplazamiento lógico a la izquierda doble.

Sintaxis SHLD *op1*, *op2*, *op3*

Descripción Desplaza a la izquierda el conjunto de bits formado por los bits de *op1* y una copia de los bits de *op2* tantas posiciones como indica *op3*. En cada operación de desplazamiento individual cada bit de *op1* se mueve al bit de peso inmediatamente superior mientras que el MSB se mueve al indicador de acarreo. Al mismo tiempo, cada bit de la copia de *op2* se mueve al bit de peso inmediatamente superior mientras que el MSB se mueve al LSB de *op1*. La operación se repite tantas veces como indique *op3* por lo que el indicador de acarreo queda con el valor del último bit en salir de *op1*. Si el valor de *op3* es 1 el indicador de desbordamiento, OF, es el OR exclusivo de CF (tras la rotación) y el MSB del resultado (es decir, OF se pone a 1 si *op1*

cambió de signo y a 0 en caso contrario). Si el valor de *op3* es mayor que 1 OF queda indefinido. Los indicadores SF, ZF y PF cambian de acuerdo al resultado dejado en *op1*. Si el valor de *op3* no es 0 AF queda indefinido. Si el valor de *op3* es 0 la instrucción no afecta a ningún indicador.

Si el operando destino es de 16 o 32 bits sólo se tienen en cuenta los 5 bits más significativos de *op3* para evitar desplazar más de 31 posiciones. Si el operando destino es de 64 bits sólo se tienen en cuenta los 6 bits menos significativos de *op3*. Si el valor de los bits considerados de *op3* es mayor que el número de bits del operando *op1* (igual al de *op2*) el resultado que queda en *op1* es indefinido.



Formas

```
SHLD  reg16, reg16, inm8
SHLD  mem16, reg16, inm8
SHLD  reg16, reg16, CL
SHLD  mem16, reg16, CL
SHLD  reg32, reg32, inm8
SHLD  mem32, reg32, inm8
SHLD  reg32, reg32, CL
SHLD  mem32, reg32, CL
SHLD  reg64, reg64, inm8
SHLD  mem64, reg64, inm8
SHLD  reg64, reg64, CL
SHLD  mem64, reg64, CL
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
0	–	–	–	↕	↕	?	↕	↕

(Ver descripción)

SHLX

Función Desplazar a la izquierda sin afectar a los indicadores.

Sintaxis SHLX *op1*, *op2*, *op3*

Descripción Desplaza a la izquierda los bits de *op2* tantas posiciones como indica *op3* y deposita el resultado en *op1*. En cada operación de desplazamiento individual cada bit de se mueve al bit de peso inmediatamente superior, mientras

que el bit más significativo se pierde. Esta operación se repite tantas veces como indique *op3*. La instrucción no afecta a ningún indicador.

Si el tamaño de los operandos es de 32 bits sólo se tiene en cuenta el valor de los bits 0 a 4 del operando *op3*, mientras que los bits 5 a 31 se ignoran. Si el tamaño de los operandos es de 64 bits sólo se tiene en cuenta el valor de los bits 0 a 5 del operando *op3*, mientras que los bits 6 a 63 se ignoran.

Formas

```
SHLX reg32, reg32/mem32, reg32
SHLX reg64, reg64/mem64, reg64
```

Indicadores No afectados.

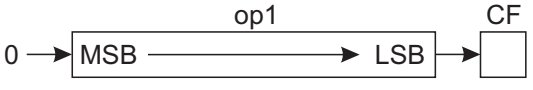
SHR

Función Desplazamiento lógico a la derecha.

Sintaxis SHR *op1*, *op2*

Descripción Desplaza a la derecha los bits de *op1* tantas posiciones como indica *op2*. En cada operación de desplazamiento individual cada bit de *op1* se mueve al bit de peso inmediatamente inferior mientras que el LSB (bit menos significativo) se mueve al indicador de acarreo. El MSB (bit más significativo) de *op1* se pone a cero. Esta operación se repite tantas veces como indique *op2* por lo que el indicador de acarreo queda con el valor del último bit en salir de *op1*. Si el valor de *op2* es 1 el indicador de desbordamiento, OF, toma el valor del MSB del dato original de *op1* (es decir, OF se pone a 1 si *op1* cambió de signo y a 0 en caso contrario). Si el valor de *op2* es mayor que 1 OF no resulta afectado. Los indicadores SF, ZF y PF cambian de acuerdo al resultado. Si el valor de *op2* no es 0 AF queda indefinido. Si el valor de *op2* es 0 la instrucción no afecta a ningún indicador.

El 8086 no enmascara la cuenta de desplazamientos. En cambio, a partir del 80286 sólo se consideran los 5 bits menos significativos de *op2* para evitar rotar más de 31 posiciones. Si el operando *op1* es de 64 bits, sólo se consideran los 6 bits menos significativos de *op2* para evitar desplazar más de 63 posiciones.



Formas

```
SHR reg8/mem8, 1
SHR reg16/mem16, 1
SHR reg32/mem32, 1
SHR reg64/mem64, 1
SHR reg8/mem8, imm8
SHR reg16/mem16, imm8
SHR reg32/mem32, imm8
```

```
SHR reg64/mem64, imm8
SHR reg8/mem8, CL
SHR reg16/mem16, CL
SHR reg32/mem32, CL
SHR reg64/mem64, CL
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
↕	–	–	–	↕	↕	?	↕	↕

(Ver descripción)

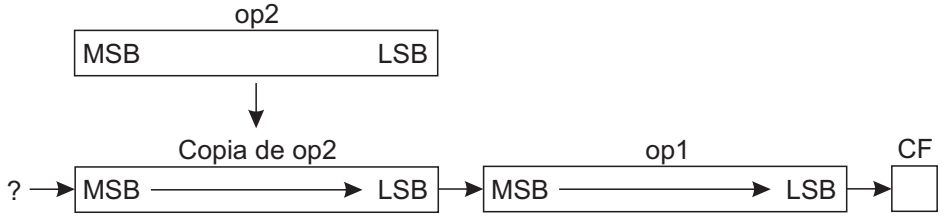
SHRD

Función Desplazamiento lógico a la derecha doble.

Sintaxis SHRD *op1*, *op2*, *op3*

Descripción Desplaza a la derecha el conjunto de bits formado por los bits de *op1* y una copia de los bits de *op2* tantas posiciones como indica *op3*. En cada operación de desplazamiento individual cada bit de *op1* se mueve al bit de peso inmediatamente inferior mientras que el LSB se mueve al indicador de acarreo. Al mismo tiempo cada bit de la copia de *op2* se mueve al bit de peso inmediatamente inferior mientras que el LSB se mueve al MSB de *op1*. Esta operación se repite tantas veces como indique *op3* por lo que el indicador de acarreo queda con el valor del último bit en salir de *op1*. Si el valor de *op3* es 1 el indicador de desbordamiento, OF, es el OR exclusivo de los dos bits más significativos del resultado en *op1* (es decir, OF se pone a 1 si *op1* cambió de signo y a 0 en caso contrario). Si el valor de *op3* es mayor que 1 OF queda indefinido. Los indicadores SF, ZF y PF cambian de acuerdo al resultado dejado en *op1*. Si el valor de *op3* no es 0 AF queda indefinido. Si el valor de *op3* es 0 la instrucción no afecta a ningún indicador.

Si el operando destino es de 16 o 32 bits sólo se tienen en cuenta los 5 bits más significativos de *op3* para evitar desplazar más de 31 posiciones. Si el operando destino es de 64 bits sólo se tienen en cuenta los 6 bits menos significativos de *op3*. Si el valor de los bits considerados de *op3* es mayor que el número de bits del operando *op1* (igual al de *op2*) el resultado que queda en *op1* es indefinido.



Formas

```
SHRD reg16, reg16, inm8
SHRD  mem16, reg16, inm8
SHRD  reg16, reg16, CL
SHRD  mem16, reg16, CL
SHRD  reg32, reg32, inm8
SHRD  mem32, reg32, inm8
SHRD  reg32, reg32, CL
SHRD  mem32, reg32, CL
SHRD  reg64, reg64, inm8
SHRD  mem64, reg64, inm8
SHRD  reg64, reg64, CL
SHRD  mem64, reg64, CL
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
0	–	–	–	↕	↕	?	↕	↕

(Ver descripción)

SHRX

Función Desplazar a la derecha sin afectar a los indicadores.

Sintaxis SHRX *op1*, *op2*, *op3*

Descripción Desplaza a la derecha los bits de *op2* tantas posiciones como indica *op3* y deposita el resultado en *op1*. En cada operación de desplazamiento individual cada bit de se mueve al bit de peso inmediatamente inferior, mientras que el bit menos significativo se pierde. Esta operación se repite tantas veces como indique *op3*. La instrucción no afecta a ningún indicador.

Si el tamaño de los operandos es de 32 bits sólo se tiene en cuenta el valor de los bits 0 a 4 del operando *op3*, mientras que los bits 5 a 31 se ignoran. Si el tamaño de los operandos es de 64 bits sólo se tiene en cuenta el valor de los bits 0 a 5 del operando *op3*, mientras que los bits 6 a 63 se ignoran.

Formas

```
SHRX reg32, reg32/mem32, reg32
SHRX reg64, reg64/mem64, reg64
```

Indicadores No afectados.

STC

Función Poner a uno el indicador de acarreo.

Sintaxis STC

Descripción Pone a 1 el indicador de acarreo.

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
–	–	–	–	–	–	–	–	1

STD

Función Poner a uno el indicador de dirección.

Sintaxis STD

Descripción Pone a 1 el indicador de dirección (DF). Esto hace que las instrucciones de cadena procesen datos retrocediendo hacia direcciones de memoria decrecientes.

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
–	1	–	–	–	–	–	–	–

STOS/STOSB/STOSW/STOSD/STOSQ

Función Almacenar en cadena de bytes/palabras/dobles palabras/cuádruples palabras.

Sintaxis STOS *destino*
STOSB
STOSW
STOSD
STOSQ

Descripción Almacena el byte byte en AL, la palabra en AX, la doble palabra en EAX o la cuádruple palabra en RAX la dirección de memoria ES:DI/ ES:EDI/RDI. A continuación DI/EDI/RDI se actualiza en base al tamaño del dato almacenado y al estado del indicador de dirección DF. El tamaño de direcciones activo en el momento de ejecutarse la instrucción determina si se usa el registro DI, EDI o RDI. El registro de segmento ES no puede sustituirse por otro.

STOSB almacena el byte contenido en AL en la posición de memoria ES:DI/ES:EDI/RDI. A continuación, si DF = 0, DI/EDI/RDI se incrementa en 1, mientras que si DF = 1, DI/EDI/RDI se decrementa en 1.

STOSW almacena la palabra contenida en AX en la posición de memoria ES:DI/ES:EDI/RDI y siguiente. A continuación, si DF = 0, DI/EDI/RDI se incrementa en 2, mientras que si DF = 1, DI/EDI/RDI se decrementa en 2.

STOSD almacena la doble palabra contenida en EAX en la posición de memoria ES:DI/ES:EDI/RDI y siguientes. A continuación, si DF = 0, DI/EDI/RDI

se incrementa en 4, mientras que si DF = 1, DI/EDI/RDI se decrementa en 4.

STOSQ almacena la cuádruple palabra contenida en RAX en la posición de memoria ES:DI/ES:EDI/RDI y siguientes. A continuación, si DF = 0, DI/EDI/RDI se incrementa en 8, mientras que si DF = 1, DI/EDI/RDI se decrementa en 8.

Si se utiliza la forma con un operando explícito (STOS *destino*), el ensamblador codificará una instrucción STOSB, STOSW, STOSD o STOSQ según el tamaño del operando destino, el cual es, normalmente, el nombre de una variable declarada mediante una directiva DB, DW, DD o DQ. La directiva indica el tamaño del dato a almacenar. El operando tiene además un valor documental al señalar la variable en donde de almacena el dato. Sin embargo, la presencia del operando no asegura que el dato se almacene automáticamente en la variable que éste designa. El dato siempre se almacena en la dirección ES:DI/ES:EDI/RDI independientemente del operando. Por tanto, es necesario que estos registros apunten a la posición de memoria destino antes de ejecutar la instrucción.

Puede utilizarse un prefijo de REP delante de una de estas instrucciones para repetirla automáticamente CX/ECX/RCX veces y así rellenar una zona de memoria con un mismo dato.

Indicadores No afectados.

SUB

Función Restar.

Sintaxis SUB *op1*, *op2*

Descripción Resta del contenido del operando *op1* el del operando *op2* y deja el resultado en *op1*.

Si *op2* es una constante inmediata con un tamaño inferior al de *op1*, la constante se extiende en signo al tamaño de *op1*.

Formas

```
SUB reg8/mem8, imm8
SUB reg16/mem16, imm16
SUB reg32/mem32, imm32
SUB reg64/mem64, imm32  ; imm32 se extiende en signo a 64 bits
SUB reg16/mem16, imm8   ; imm8 se extiende en signo a 8 bits
SUB reg32/mem32, imm8   ; imm8 se extiende en signo a 32 bits
SUB reg64/mem64, imm8   ; imm8 se extiende en signo a 64 bits
SUB reg8/mem8, reg8
SUB reg16/mem16, reg16
SUB reg32/mem32, reg32
SUB reg64/mem64, reg64
SUB reg8, reg8/mem8
SUB reg16, reg16/mem16
```

```
SUB reg32, reg32/mem32
SUB reg64, reg64/mem64
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
↕	–	–	–	↕	↕	↕	↕	↕

TEST

Función Comparación lógica.

Sintaxis TEST *op1*, *op2*

Descripción Realiza la operación lógica AND (producto lógico) bit a bit entre los operandos *op1* y *op2*, afectando a los indicadores pero desechando el resultado. Por tanto, el operando *op1* no resulta modificado.

Si *op2* es una constante inmediata con un tamaño inferior al de *op1*, la constante se extiende en signo al tamaño de *op1*.

Formas

```
TEST reg8/mem8, imm8
TEST reg16/mem16, imm16
TEST reg32/mem32, imm32
TEST reg64/mem64, imm32 ; imm32 se extiende en signo a 64 bits
TEST reg64/mem64, imm8  ; imm8 se extiende en signo a 64 bits
TEST reg8/mem8, reg8
TEST reg16/mem16, reg16
TEST reg32/mem32, reg32
TEST reg64/mem64, reg64
TEST reg8, reg8/mem8
TEST reg16, reg16/mem16
TEST reg32, reg32/mem32
TEST reg64, reg64/mem64
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
0	–	–	–	↕	↕	?	↕	0

TZCNT

Función Contar el número de ceros por la derecha.

Sintaxis LZCNT *op1*, *op2*

Descripción Cuenta el número de bits a cero consecutivos comenzando por el bit menos significativo del operando *op2* y hasta encontrar el bit a uno de menos peso. El resultado de la cuenta se almacena en el operando *op1*. Si el operando *op2* es cero el indicador de acarreo se pone a uno y el número de bits de *op2* se almacena en *op1*. En caso contrario, el indicador de acarreo se pone a cero.

Formas

```
TZCNT reg16, reg16/mem16
TZCNT reg32, reg32/mem32
TZCNT reg64, reg64/mem64
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
?	–	–	–	?	↕	?	?	↕

TZMSK

Función Enmascarar desde el primer uno por la derecha.

Sintaxis TZCNT *op1*, *op2*

Descripción Encuentra el primer bit a uno comenzando por la derecha, pone a uno todos los bits por debajo de ese, pone a cero todos los demás bits (incluido el bit a uno encontrado) y escribe el resultado en el operando *op1*. Si el bit menos significativo de *op2* es uno, el operando *op1* queda a cero.

La instrucción realiza una operación equivalente a:

```
SUB tmp1, op2, 1
NOT tmp2, op2
AND op1, tmp1, tmp2
```

El indicador de acarreo cambia según la operación SUB y los demás según la operación AND.

Formas

```
TZMSK reg16, reg16/mem16
TZMSK reg32, reg32/mem32
TZMSK reg64, reg64/mem64
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
?	–	–	–	?	↕	?	?	↕

WRFSBASE/WRGSBASE

Función Escribir base de FS/escribir base de GS.

Sintaxis

Sintaxis WRFSBASE *op1*
WRGSBASE *op1*

Descripción Escribe el campo de base del registro FS o GS con el dato contenido en el operando *op1*, que debe ser un registro de 32 o 64 bits. La dirección escrita en el campo de base debe estar en forma canónica o se producirá una excepción de protección general. Esta instrucción sólo es válida en modo de 64 bits.

Formas
```
WRFSBASE reg32
WRFSBASE reg64
WRGSBASE reg32
WRGSBASE reg64
```

Indicadores No afectados.

XADD

Función Intercambiar y sumar.

Sintaxis XADD *op1*, *op2*

Descripción Carga *op2* con el contenido inicial de *op1* y carga *op1* con la suma de los contenidos iniciales de *op1* y *op2*. El operando *op2* debe ser un registro.

Formas
```
XADD reg8/mem8, reg8
XADD reg16/mem16, reg16
XADD reg32/mem32, reg32
XADD reg64/mem64, reg64
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
↕	–	–	–	↕	↕	↕	↕	↕

XCHG

Función Intercambiar.

Sintaxis XCHG *op1*, *op2*

Descripción Intercambia los contenidos de los operandos *op1* y *op2*.

Formas
```
XCHG reg8/mem8, reg8
XCHG reg8, reg8/mem8
XCHG reg16/mem16, reg16
XCHG reg16, reg16/mem16
XCHG reg32/mem32, reg32
XCHG reg32, reg32/mem32
XCHG reg64/mem64, reg64
XCHG reg64, reg64/mem64
```

Indicadores No afectados.

XLAT/XLATB

Función Traducir mediante tabla.

Sintaxis XLAT
XLATB

Descripción Las instrucciones XLAT y XLATB leen de una tabla de bytes la entrada cuyo índice se indica en AL antes de ejecutarse la instrucción y copian la entrada de la tabla en AL. La dirección base de la tabla es DS:BX, DS:EBX o RBX según el modo de operación actual. La instrucción XLAT equivaldría a una instrucción MOV AL,[BX+AL], MOV AL,[EBX+AL] o MOV AL, [RBX+AL] (que no existen). Por tanto, XLAT y XLATB transforman o traducen el valor de AL en base a la consulta de una tabla. El registro de segmento DS, usado por defecto, puede cambiarse mediante el correspondiente prefijo.

A nivel del lenguaje ensamblador, existen dos formas para esta instrucción: la forma con operando de memoria explícito y la forma sin operandos. La forma con operando de memoria explícito permite especificar la dirección base de la tabla con un símbolo. No obstante, es muy importante tener en cuenta que el operando de memoria explícito se da únicamente con fines de documentación del código fuente y no implica directamente que la instrucción use el operando como dirección base sino que dicha dirección base debe cargarse en DS:BX/DS:EBX/RBX previamente a la ejecución de la instrucción. Otra razón de la existencia de la forma con operando explícito es que ésta ofrece la oportunidad de cambiar el registro de segmento mediante un prefijo *regseg*: antepuesto al operando. En la forma sin operandos no hay oportunidad de cambiar el registro de segmento por defecto.

Formas
```
XLAT mem8
XLATB
```

Indicadores No afectados.

XOR

Función Operación lógica OR exclusiva.

Sintaxis XOR *op1*, *op2*

Descripción Realiza la operación lógica OR exclusiva bit a bit entre los operandos *op1* y *op2* y deja el resultado en *op1*.

Si *op2* es una constante inmediata con un tamaño inferior al de *op1*, la constante se extiende en signo al tamaño de *op1*.

Formas
```
XOR reg8/mem8, imm8
XOR reg16/mem16, imm16
XOR reg32/mem32, imm32
XOR reg64/mem64, imm32     ; imm32 se extiende en signo a 64 bits
```

```
XOR reg16/mem16, imm8      ; imm8 se extiende en signo a 16 bits
XOR reg32/mem32, imm8      ; imm8 se extiende en signo a 32 bits
XOR reg64/mem64, imm8      ; imm8 se extiende en signo a 64 bits
XOR reg8/mem8, reg8
XOR reg16/mem16, reg16
XOR reg32/mem32, reg32
XOR reg64/mem64, reg64
XOR reg8, reg8/mem8
XOR reg16, reg16/mem16
XOR reg32, reg32/mem32
XOR reg64, reg64/mem64
```

Indicadores

OF	DF	IF	TF	SF	ZF	AF	PF	CF
0	–	–	–	↕	↕	?	↕	0