Repertorio de Instrucciones del 8086/8088

A continuación se desribe el repertorio de instrucciones del 8086/8088. Estas instrucciones se encuentran más detalladas en [Rod00] y [PCG00].

AAA – Ajuste ASCII en suma *(Ascii Adjust for Addition)*

Uso: AAA

Flags que modifica: AF CF (OF,PF,SF,ZF indefinidos)

Descripción: Corrige el resultado en AL de una suma de dos números decimales desempaquetados, convitiéndolo en un valor decimal desempaquetado. La operación que realiza es la siguiente:

```
Si bits 3 a 0 de AL > 9 ó AF = 1
        AL = AL + 6
        AH = AH + 1
        AF = 1, CF = AF
        Bits 7 a 4 de AL = 0000b
```

AAD – Ajuste ASCII en división *(Ascii Adjust for Division)*

Uso: AAD

Flags que modifica: SF ZF PF (AF,CF,OF indefinidos)

Descripción: Realiza un ajuste del dividendo en AL antes de hacer la divsión de dos números decimales desempaquetados, para que el resultado de esta división (cociente) sea un valor decimal desempaquetado. La lógica que usa es la siguiente:

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

AAM – Ajuste ASCII en multiplicación *(Ascii Adjust for Multiplication)*

Uso: AAM

Flags que modifica: PF SF ZF (AF,CF,OF indefinidos)

Descripción: Corrige el resultado en AX del producto de dos números decimales desempaquetados, convirtiéndolo en un valor decimal desempaquetado. La lógica que emplea es la siguiente:

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

AAS –Ajuste ASCII en resta *(Ascii Adjust for Subtraction)*

Uso: AAS

Flags que modifica: AF CF (OF,PF,SF,ZF indefinidos)

Descripción: Corrige el resultado en AL de la resta de dos números decimales desempaquetados, convirtiéndolos en un valor decimal desempaquetado. La lógica que emplea es la siguiente:

```
Si bits 3 a 0 de AL > 9 ó AF = 1
        AL = AL –6
        AH = AH –1
        AF = 1,    CF = AF
        Bits 7 a 4 de AL = 0000b
```

ADC – Sumar con acarreo *(Add With Carry)*

Uso: ADC dest,src

Flags que modifica: AF CF OF SF PF ZF

Descripción: Suma los dos operandos. Además, añade 1 a la suma si está activada la bandera de accarreo (CF). El resultado se almacena en el operando destino. Los operandos deben ser del mismo tipo (byte o palabra).

ADD – Suma aritmética *(Arithmetic Addition)*

Uso: ADD dest,src

Flags que modifica: AF CF OF PF SF ZF

Descripción: Suma los dos operandos. El resultado se almacena en el operando destino. Los operandos deben ser del mismo tipo (byte o palabra).

AND – Y lógico *(Logical And)*

Uso: AND dest,src

Flags que modifica: CF OF PF SF ZF (AF indefinidos)

Descripción: Operación "y lógica" a nivel de bit entre los dos operandos. El resultado se almacena en el operando destino. Los operandos deben ser del mismo tipo (byte o palabra).

CALL – Llamada a una función o procedimiento *(Procedure Call)*

Uso: CALL destination

Flags que modifica: Ninguno

Descripción: Bifurca a un procedimiento fuera de línea, salvando previamente en la pila la dirección de la instrucción siguiente, para poder volver a esta instrucción una vez ejecutado el procedimiento. El procedimiento llamado puede estar:

- dentro del mismo segmento (llamada NEAR). En este caso, se almacena en la pila el desplazamiento de la instrucción siguiente.
- en otro segmento (llamada FAR). En este caso, se almacena primero en la pila el segmento y segundo el desplazamiento de la instrucción siguiente, en ese orden.

La llamada puede ser, a su vez:

- Directa, es decir, a etiquetas relativas a CS, no a variables. Se supone siempre NEAR, a menos que se especifique FAR en la declaración de destino.
- Indirecta, es decir, a variables, pudiendo especificarse WORD PTRT ó D WORD PTR para llamadas NEAR o FAR, respectivamente. Si se especifica una variable tipo WORD, el contenido de esa palabra es la nueva dirección (desplazamiento). Si se especifica un dirección tipo DWORD; la primera palabra contiene el desplazamiento y la segunda el segmento. También se pueden usar registros de 16 bits entre corchetes.

CBW – Convertir byte en palabra. *(Convert Byte to Word)*

Uso: CBW

Flags que modifica: Ninguno

Descripción: Copia el bit 7 del registro AL en todos los bits del registro AH; es decir, expande el bit de signo de AL.

CLC – Borrar bandera de acarreo *(Clear Carry)*

Uso: CLC

Flags que modifica: CF

Descripción: Borra la bandera de acarrero (CF) sin afectar a ninguna otra bandera.

CLD – Borrar bandera de dirección *(Clear Direction Flag)*

Uso: CLD

Flags que modifica: DF

Descripción: Pone a 0 la bandera de direción (DF), por lo que en la ejecución de las instrucciones de manejo de cadenas los registros índices SI y/o DI se autoincrementan en una unidad si el (los) operando(s) son de tipo byte, y en dos, si son de tipo word.

CLI – Borrar (desactivar) bandera de interrupciones *(Clear Interrupt Flag)*

Uso: CLI

Flags que modifica: IF

Descripción: Borra la bandera de activación de interrupciones (IF) y desactiva las interrupciones enmascarables (las que aparecen sobre la línea INTR del procesador). Las interrupciones enmascarables se pueden activar (permitir) o desactivar (no permitir). Las interrupciones no enmascarables (las que aparecen sobre la línea NMI) no se pueden desactivar.

CMC – Complementar bandera de acarreo *(Complement Carry Flag)*

Uso: CMC

Flags que modifica: CF

Descripción: Complementa la bandera de acarreo (CF); es decir, la pone en el estado contrario.

CMP – Comparar dos operandos *(Compare)*

Uso: CMP dest,src

Flags que modifica: AF CF OF PF SF ZF

Descripción: Resta fuente de destino, pero no retorna el resultado. Los operandos quedan inalterados, pero las banderas se actualizan, pudiéndose consultar a continuación mediante una instrucción de bifurcación condicional. Los operandos pueden ser de tipo byte o palabra, pero ambos del mismo tipo.

CMPS - Comparar cadenas *(Compare String -Byte, Word or Doubleword-)*

Uso: CMPS dest,src
CMPSB
CMPSW

Flags que modifica: AF CF OF PF SF ZF

Descripción: Sirve para comparar dos operandos de memoria (tipo byte o palabra). Para ello realiza la operación:

Cadena fuente – cadena destino

afectando a las banderas, pero sin almacenar el resultado. La cadena fuente está direccionada por SI en el segmento de datos, es decir, DS:[SI]. La cadena destino está direcionada por DI en el segmento extra, es decir, ES:[DI]. Los registros SI y DI se actualizan para que apunten al siguiente elemento de la cadena. Si la bandera de dirección es cero (DF=0), ambos SI y DI se incrementan. Si DF=1, ambos se decrementan.

En la instrucción CMPSB, se realiza la resta entre bytes DS:[SI] – ES:[DI].

En la instrucción CMPSW, se realiza la resta entre palabras DS:[SI] – ES:[DI].

CWD – Convertir palabra en doble palabra *(Convert Word to Doubleword)*

Uso: CWD

Flags que modifica: Ninguno

Descripción: Expande el bit de signo del registro AX (bit 15) sobre el registro DX.

DAA – Ajuste decimal en suma *(Decimal Adjust for Addition)*

Uso: DAA

Flags que modifica: AF CF PF SF ZF (OF indefinidos)

Descripción: Corrige el resultado en AL de la suma de dos números decimales empaquetados, convirtiéndolo en un valor decimal empaquetado.

DAS – Ajuste decimal en resta *(Decimal Adjust for Subtraction)*

Uso: DAS

Flags que modifica: AF CF PF SF ZF (OF indefinidos)

Descripción: Corrige el resultado en AL de la resta de dos número decimales empaquetados, convirtiéndolo en una valor decimal empaquetado.

DEC – Decrementar destino en uno *(Decrement)*

Uso: DEC dest

Flags que modifica: AF OF PF SF ZF

Descripción: Resta una unidad del operando destino. El operando puede ser byte o palabra.

DIV – División *(Divide)*

Uso: DIV src

Flags que modifica: (AF,CF,OF,PF,SF,ZF indefinidos)

Descripción: Dividie, sin considerar el signo, un número contenido en el acumulador y su extensión (AH, AL si el operando es de tipo byte o DX,AX si el operando es de tipo palabra) entre el operando fuente. El cociente se almacena en el acumulador (AL o AX, según que el operando fuente sea de tipo byte o palabra). El resto se almacena en la extensión del acumulador (AH o DX, según que el operando fuente sea de tipo byte o palabra).

ESC – Escape *(Escape)*

Uso: ESC immed,src

Flags que modifica: Ninguno

Descripción: Esta instrucción suministra un mecanismo de comunicación con otros coprocesadores. Se usa para indicar a un coprocesador que comienza a realizar alguna tarea. A su vez, el 8086/8088 continúa ejecutando la instrucción siguiente. Se usa en combinación con WAIT.

El primer operando es un valor inmediato de 6 bits (0 a 63) y sirve para denotar, a la vez, al coprocesador y la operación a ejecutar. El segundo operando es tipo registro o memoria y especifica el dato a pasar al coprocesador.

HLT – Detener CPU – *(Halt CPU)*

Uso: HLT

Flags que modifica: Ninguno

Descripción: El procesador se para. Sólo abandona ese estado en uno de estos casos:

- Se restaura el sistema (activación de la línea RESET).
- Se recibe una interrupción no enmascarable sobre la línea NML
- Se recibe una interrupción enmascarable sobre la línea INTR (si las interrupciones están permitidas).

IDIV – Dividir con signo *(Signed Integer Division)*

Uso: IDIV src

Flags que modifica: (AF,CF,OF,PF,SF,ZF indefinidos)

Descripción: Divide, considerando el signo, un número contenido en el acumulador y su extensión (AH, AL si el operando es de tipo byte o DX, AX si es de tipo palabra) entre el operando fuente. El cociente se almacena en el acumulador (AL o AX, según que el operando fuente sea de tipo byte o palabra). El resto se almacena en la extensión del acumulador (AH o DX, según que el operando fuente sea de tipo byte o palabra).

IMUL – Multiplicar con signo *(Signed Multiply)*

Uso: IMUL src

Flags que modifica: CF OF (AF,PF,SF,ZF indefinidos)

Descripción: Multiplica, considerando el signo, el acumulador (AL o AX) por el operando fuente, según que el tipo de este operando fuente sea byte o palabra. Si el operando fuente es de tipo byte, el resultado se almacena en AX. Si el operando fuente es de tipo palabra, el resultado se almacena en AX (palabra inferior) y DX (palabra superior).

IN – Entrada de byte o palabra *(Input Byte or Word From Port)*

Uso: IN accum,puerto

Flags que modifica: Ninguno

Descripción: Transfiere un byte o una palabra de una puerta de entrada del procesador al registro AL o AX, respectivamente. El número de la puerta se puede especificar mediante:

- un valor fijo (de 0 a 255).
- un valor variable, el contenido en el registro DX (de 0 a 65535), pudiéndose acceder a 64K puertas de entrada.

INC – Incremento *(Increment)*

Uso: INC dest

Flags que modifica: AF OF PF SF ZF

Descripción: Suma una unidad del operando destino. El operando puede ser byte o palabra.

INT – Interrupción *(Interrupt)*

Uso: INT num

Flags que modifica: TF IF

Descripción: INT activa el procedimiento de interrupción especificado por el operando. La dirección del vector de interrupción se calcula multiplicando por 4 el operando, que es un valor entre 0 y 255. El vector de interrupción se compone de dos palabras: desplazamiento y segmento. Se bifurca a la dirección especificada por el vector de interrupción, salvando previamente las banderas y la dirección de retorno (CS,IP).

INTO – Interrupción si desbordamiento *(Interrupt on Overflow*)

Uso: INTO

Flags que modifica: IF TF

Descripción: Genera una interrupción tipo 4 si existe desbordamiento (*overflow*), es decir, si OF=1. En caso contrario se continúa la ejecución con la instrucción siguiente, sin ejecutar el procedimiento de interrupción.

IRET – Retornar de una interrupción *(Interrupt Return)*

Uso: IRET

Flags que modifica: AF CF DF IF PF SF TF ZF

Descripción: Devuelve el control a la dirección de retorno salvada en la pila por una operación de interrupción previa y restaura los registros de banderas. IRET se utiliza para finalizar un procedimiento de interrupción.

Jxx – Saltos condicionales *Conditional Jumps*

Instrucción	Descripción	Condición de salto
JA	Saltar si Superior	CF=0 y ZF=0
JAE	Saltar si Superior o Igual	CF=0
JB	Saltar si Inferior	CF=1
JBE	Saltar si Inferior o Igual	CF=1 o ZF=1
JC	Saltar si hay acarreo	CF=1
JCXZ	Saltar si CX es cero	CX=0
JE	Saltar si Igual	ZF=1
JG	Saltar si Mayor (con signo)	ZF=0 y SF=OF
JGE	Saltar si Mayor o Igual (con signo)	SF=OF
JL	Saltar si Menor (con signo)	SF != OF
JLE	Saltar si Menor o Igual (con signo)	ZF=1 o SF != OF
JNA	Saltar si No Superior	CF=1 o ZF=1
JNAE	Saltar si No Superior o Igual	CF=1
JNB	Saltar si No Inferior	CF=0
JNBE	Saltar si No Inferior o Igual	CF=0 y ZF=0
JNC	Saltar si No hay acarreo	CF=0
JNE	Saltar si No Igual	ZF=0
JNG	Saltar si No Mayor (con signo)	ZF=1 o SF != OF
JNGE	Saltar si No Mayor o Igual (con signo)	SF != OF
JNL	Saltar si No Menor (con signo)	SF=OF
JNLE	Saltar si No Menor o Igual (con signo)	ZF=0 y SF=OF
JNO	Saltar si No Overflow (con signo)	OF=0
JNS	Saltar si Positivo	SF=0
JNZ	Saltar si No Cero	ZF=0
JO	Saltar si Overflow (con signo)	OF=1
JP /JPE	Saltar si Par	PF=1
JPO/JNP	Saltar si Impar	PF=0
JS	Saltar si negativo (si hay signo)	SF=1
JZ	Saltar si cero	ZF=1

JCXZ – Bifurcar si CX es cero (*Jump if Register CX is Zero)*

Uso: JCXZ label

Flags que modifica: Ninguno

Descripción: Transfiere el control a la instrucción IP + desplazamiento si se cumple la condición de CX = 0. Si CX no es 0, no hay transferencia de control. El desplazamineto debe estar comprendido entre –128 y +127 bytes de esta instrucción, es decir, desplazamiento es un valor con signo de 8 bits (1 byte).

JMP – Salto incondicional *(Unconditional Jump)*

Uso: JMP dirección

Flags que modifica: Ninguno

Descripción: Transfiere el control incondicionalmente a la dirección indicada. El salto puede ser cercano (dentro del mismo segmento) o lejano (a otro segmento).

LAHF – Cargar AH con banderas *(Load Register AH From Flags)*

Uso: LAHF

Flags que modifica: Ninguno

Descripción: Copia las banderas SF, ZF, AF, PF y CF en los bits 7, 6, 4, 2 y 0, respectivamente del registro AH. Los otros bits quedan indefinidos. El contenido de AH queda, entonces, así:

AH ← SF ZF xx AF xx PF xx CF

LDS – Cargar registro de segmento de datos *(Load Pointer Using DS)*

Uso: LDS dest,src

Flags que modifica: Ninguno

Descripción: Transfiere un puntero de 32 bits (dirección completa compuesta por desplazamiento y segmento, en este orden) correspondiente al segundo operando (que debse ser un operando de memoria de doble palabra).

LEA – Cargar dirección efectiva *(Load Effective Address)*

Uso: LEA dest,src

Flags que modifica: Ninguno

Descripción: Transfiere el desplazamiento del operando fuente al operando destino. El operando fuente debe ser un operando de memoria (byte o palabra). El operando destino es un registro de 16 bits, pero no un registro de segmento. Permite especificar registros índices en el operando fuente, al contrario que con el operador OFFSET.

LES – Cargar registro de segmento extra *(Load Pointer Using ES)*

Uso: LES dest,src

Flags que modifica: Ninguno

Descripción: Transfiere un puntero de 32 bits (dirección completa compuesta por desplazamiento y segmento, en este orden) correspondiente al segundo operando (que debe ser un operando de memoria de doble palabra).

LOCK – Bloquear el bus *(Lock Bus)*

Uso: LOCK

Flags que modifica: Ninguno

Descripción: LOCK es un prefijo de un byte que acompaña a una instrucción y que activa la señal LOCK mientras dicha instrucción se ejecuta; es decir, impide la utilización del bus por otros procesadores, impidiendo de esta forma el acceso a los recursos compartidos por éstos. En sistemas de múltiples procesadores con recursos compartidos es necesario un mecanismo de este tipo para el control del acceso a estos recursos.

LODS – Cargar cadena (*Load String - Byte, Word or Double-)*

Uso: LODS src
LODSB
LODSW

Flags que modifica: Ninguno

Descripción: Transfiere un byte o una palabra de la cadena fuente (direccionada por DS:SI) al registro acumulador AL o AX. Actualiza el registro SI para que apunte al siguiente elemento de la cadena a cargar:

- Si el operando es de tipo byte, se transfiere un byte y el registro SI cambia una unidad.
- Si el operando es de tipo palabra, se transfiere una palabra y el registro SI cambia dos unidades.

- Si la bandera de dirección es cero (DF=0), SI se incrementa. Si DF=1, se decrementa (véanse las instrucciones CLD y STD).

En la instrucción LODSB se transfiere el byte DS:(SI) al registro AL. SI se actualiza en una unidad. En la instrucción LODSW se transfiere la palabra DS:(SI) al registro AX. SI se actualiza en dos unidades.

El operando especificado en LODS lo utiliza el ensamblador únicamente para verificar el tipo (byte o palabra) y para ver si se ha especificado un registro de segmento. LODS mueve realmente el byte o palabra de DS:(SI) al registro AL o AX, sin usar realmente el operando de la instrucción.

LOOP – Bucle *(Decrement CX and Loop if CX Not Zero)*

Uso: LOOP label

Flags que modifica: Ninguno

Descripción: Decrementa el registro contador (CX). Si CX es distinto de cero, entonces IP = IP + desplazamiento (expandiendo el signo a 16 bits). Si CX es cero, entonces se ejecuta la siguiente instrucción.

El desplazamiento debe estar comprendido entre -128 y +127 bytes de esta instrucción, es decir, desplazamiento es un valor con signo de 8 bits (1 byte).

Mediante esta instrucción es posible implementar bucles. Un bucle es un conjunto de instrucciones que se ejecutan una serie de veces. El esquema es el siguiente:

```
        MOV   CX,contador    ; CX = número de veces que se va a
                             ; ejecutar el bucle.
BUCLE:  ...
        ...
        LOOP  BUCLE          ; CX = CX-1 y bifurca a BUCLE
                             ; si CX ≠ 0
```

Es posible realizar bucles anidados, es decir, uno dentro de otro. El esquema para dos bucles es:

```
        MOV   CX,contador1   ; CX = contador bucle 1
BUCLE1: ...
        ...
           PUSH CX           ; salvar CX
           MOV  CX,contador2 ; CX = contador bucle 2
BUCLE2: ...
        ...
        LOOP BUCLE2          ; CX = CX-1 y bifurca a BUCLE2
                             ; si CX ≠ 0
        POP   CX             ; recuperar CX
        ...
        ...
        LOOP BUCLE1          ; CX = CX-1 y bifurca a BUCLE1
                             ; si CX ≠ 0
```

LOOPE/LOOPZ – Bucle si igual / Bucle si cero *(Loop While Equal / Loop While Zero)*

Uso: LOOPE label
LOOPZ label

Flags que modifica: Ninguno

Descripción: Decrementa el registro contador (CX). Si ZF = 1 y CX es distinto de cero, entonces IP = IP + desplazamiento (expandiendo el signo del desplazamiento a 16 bits). Si ZF = 0 o CX = 0, entonces se ejecuta la instrucción siguiente. El desplazamiento debe estar comprendido ente -128 y +127 bytes de esta instrucción, es decir, desplazamiento es un valor con signo de 8 bits (1 byte).

LOOPNZ – Bucle si no cero *(Loop While Not Zero)*
LOOPNE –Bucle si distinto *(Loop While Not Equal)*

Uso: LOOPNZ label
LOOPNE label

Flags que modifica: Ninguno

Descripción: Decrementa el registro contador (CX). Si ZF = 0 y CX es distinto de cero, entonces IP = IP + desplazamiento (expandiendo el signo del desplazamiento a 16 bits). Si ZF = 1 o CX = 0, entonces se ejecuta la instrucción siguiente.

El desplazamiento debe estar comprendido entre -128 y +127 bytes de esta instrucción, es decir, desplazamiento es un valor con signo de 8 bits (1 byte).

MOV –Mover *(Move Byte or Word)*

Uso: MOV dest,src

Flags que modifica: Ninguno

Descripción: Transfiere un byte o una palabra desde el operando fuente al operando destino. El operando destino puede ser un registro o un elemento de memoria (byte o palabra). El operando fuente puede ser un registro, un elemento de memoria o un valor inmediato. Ambos operandos debe ser del mismo tipo (byte o palabra).
El contenido especificado por el elemento fuente se copia sobre el elemento de destino, quedando inalterado el elemento fuente.

MOVS – Mover cadena *(Move String - Byte or Word-)*

Uso: MOVS dest,src
MOVSB
MOVSW

Flags que modifica: Ninguno

Descripción: Transfiere un byte o una palabra de la cadena fuente (direccionada por SI) en el segmento de datos a la cadena de destino (direccionada por DI) en el segmento extra. Actualiza los registros SI y DI para que apunten al siguiente elemento de la cadena:
Si los operandos son de tipo byte, se transfiere un byte y los registros SI y DI cambian una unidad. Si los operando son de tipo palabra, se transfiere una palabra y los registros SI y DI cambian dos unidades.

Si la bandera de dirección es cero (DF = 0), ambos SI y DI se incrementan. Si DF = 1, ambos se decrementan (véanse las instrucciones CLD y STD).

En la instrucción MOVSB, se transfiere el byte DS:[SI] a ES:[DI]. En la instrucción MOVSW, se transfiere la palabra DS:[SI] a ES:[DI].

Los operandos especificados en MOVS los utiliza el ensamblador únicamente para verificar el tipo (byte o palabra) y para ver si se ha especificado un registro de segmento. MOVS mueve realmente el byte o la palabra apuntada por DS:[SI] a ES:[DI], sin usar realmente los operandos de la instrucción.

Cuando se usa con REP, se realiza una transferencia de una cadena completa (de bytes o de palabras), siendo el número de elementos a mover el especificado en el registro CX.

En MOVS se puede reasignar el elemento fuente, pero no el destino; es decir, se puede especificar, por ejemplo,

MOVS DESTINO,ES:FUENTE

En este caso, habría una transferencia de un elemento del segmento, direccionado por ES, al mismo segmento. Para mover elementos dentro del segmento direccionado por DS, habría que hacer ES = DS.

MUL – Multiplicar sin signo *(Unsigned Multiply)*

Uso: MUL src

Flags que modifica: CF OF (AF,PF,SF,ZF indefinidos)

Descripción: Multiplica, sin considerar el signo, el acumulador (AL o AX) por el operando fuente, según que el tipo de este operando fuente sea byte o palabra. Si el operando fuente es de tipo palabra, el resultado se almacena en AX (palabra inferior) y DX (palabra superior). Si la mitad superior del resultado (AH para el caso de operando tipo byte o DX para el caso de operando tipo palabra) no es cero, se activan las banderas CF y OF, indicando que esta mitad superior contiene dígitos significativos del resultado.

NEG – Complemento a dos (*Two's Complement Negation)*

Uso: NEG dest

Flags que modifica: AF CF OF PF SF ZF

Descripción: Calcula el valor negativo del operando, es decir, resta el operando de cero y devuelve el resultado en el mismo operando (byte o palabra). Para hacer esto, el operando destino se resta del número compuesto por todo unos y se le añade 1. Esto es lo mismo que el complemento a 2 del número.

NEG destino es equivalente a las instrucciones

```
NOT destino
INC destino
```

NOP – No operación *(No Operation -90h-)*

Uso: NOP

Flags que modifica: Ninguno

Descripción: El procesador no hace nada. Pasa a ejecutar la instrucción siguiente.

NOT – No lógico *(One's Compliment Negation -Logical NOT-)*

Uso: NOT dest

Flags que modifica: Ninguno

Descripción: Cambia los bits unos por ceros y los bits ceros por unos, es decir, realiza el complemento a uno del operando y devuelve el resultado en el mismo operando (byte o palabra).

OR – O lógico inclusivo *(Inclusive Logical OR)*

Uso: OR dest,src

Flags que modifica: CF OF PF SF ZF (AF indefinidos)

Descripción: Realiza la operación "o lógico inclusivo" a nivel de bit entre los dos operandos. El resultado se almacena en destino.

La tabla de la operación OR (para las cuatro combinaciones posibles de bits) es:

Op1	Op2	Dest
0	0	0
0	1	1
1	0	1
1	1	1

OUT – Salida de byte o palabra *(Output Data to Port)*

Uso: OUT port,accum

Flags que modifica: Ninguno

Descripción: Transfiere un byte o una palabra del registro AL o AX a una puerta de salida del procesador. El número de la puerta se puede especificar mediante:

- un valor fijo (de 0 a 255);
- un valor variable, el contenido en el registro DX (de 0 a 65535), pudiéndose acceder a 64K puertos de salida.

OUTS – Salida de cadena *(Output String to Port)* (80188+)

Uso: OUTS port,src
OUTSB
OUTSW

Flags que modifica: Ninguno

Descripción: Transfiere un byte (OUTSB) o un word (OUTSW) desde src al puerto especificado. En las instrucciones OUTSB y OUTSW el operando se encuentra en DS:SI, y el registro SI se incrementa o decrementa con el tamaño del dato leído según el valor del flag DF.

POP – Quitar palabra de la pila *(Pop Word off Stack)*

Uso: POP dest

Flags que modifica: Ninguno

Descripción: Transfiere el elemento (tipo palabra) que se encuentra en lo alto de la pila (apuntando por el registro SP) al operando destino (tipo palabra), y luego incrementa en dos el registro SP. El registro CS no se puede especificar como destino. La instrucción que realiza la función opuesta de POP es PUSH.

POPA – Quitar registros de la pila *(Pop All Registers onto Stack)* (80188+)

Uso: POPA

Flags que modifica: Ninguno

Descripción: Transdiere los 8 registros de propósito general desde la pila, en el siguiente orden: DI; SI; BP; SP, DX, CX y AX.

POPF – Quitar banderas de la pila *(Pop Flags off Stack)*

Uso: POPF

Flags que modifica: all flags

Descripción: Transfiere bits específicos de la palabra que se encuentra en lo alto de la pila (apuntado por el registro SP) a las banderas, reemplazando así los valores que contenían previamente. El registro SP se incrementa luego en 2.

```
bit 11 -- OF
bit 10 -- DF
bit  9 -- IF
bit  8 -- TF
bit  7 -- SG
bit  6 -- ZF
bit  4 -- AF
bit  2 -- PF
bit  0 -- CF
```

La instrucción que realiza la función opuesta de POPF es PUSHF.

PUSH – Poner palabra en la pila *(Push Word onto Stack)*

Uso: PUSH src
PUSH immed (80188+ only)

Flags que modifica: Ninguno

Descripción: Decrementa el puntero de la pila (SP) en 2 y luego transfiere la palabra especificada en el operando fuente a lo alto de la pila (ahora apuntada por el registro SP). El registro CS no se puede especificar como fuente.

La instrucción que realiza la función opuesta de PUSH es POP.

PUSHA – Poner registros en la pila *(Push All Registers onto Stack)* (80188+)

Uso: PUSHA

Flags que modifica: Ninguno

Descripción: Transdiere a la pila los 8 registros de propósito general, en el siguiente orden: DI; SI; BP; SP, DX, CX y AX.

PUSHF – **Poner banderas en la pila** *(Push Flags onto Stack)*

Uso: PUSHF

Flags que modifica: Ninguno

Descripción: Decrementa el puntero de la pila (SP) en 2 y luego transfiere los valores de las banderas a bits específicos de la palabra de la pila direccionada por el registro SP:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	OF	DF	IF	TF	SF	ZF	-	AF	-	PF	-	CF

La instrucción que realiza la función opuesta de PUSHF es POPF.

RCL – **Rotar a la izquierda a través del acarreo** *(Rotate Through Carry Left)*

Uso: RCL dest,count

Flags que modifica: CF OF

Descripción: Rotar a la izquierda los bits del operando destino junto con la bandera de acarreo (CF) el número de bits especificado en el segundo operando.

Si el número de bits a desplazar es 1, se puede especificar directamente. Si es mayor que 1, su valor debe cargarse en CL y especificar CL como segundo operando.

RCR – **Rotar a la derecha a través del acarreo** *(Rotate Through Carry Right)*

Uso: RCR dest,count

Flags que modifica: CF OF

Descripción: Rotar a la derecha los bits del operando destino junto con la bandera de acarreo (CF) el número de bits especificado en el segundo operando.

Si el número de bits a desplazar es 1, se puede especificar directamente. Si es mayor que 1, su valor debe cargarse en CL y especificar CL como segundo operando.

REP – **Repetir operación de cadena** *(Repeat String Operation)*

Uso: REP

Flags que modifica: Ninguno

Descripción: Hace que la siguiente instrucción de cadena se repita un determinado número de veces, el especificado en el registro contador (CX).

REP se usa en conjunción con las instrucciones MOVS (mover cadena) y STOS (almacenar cadena).

REPE y REPZ son idénticas y generan el mismo código (un byte) que REP, pero se utilizan en las instrucciones CMPS (comparar cadena) y SCAS (explorar cadena), pero necesitan que ZF = 1 (es decir, que el resultado de la comparación o exploración haya dado igual o cero).

REPNE y REPNZ son idénticas y generan el mismo código (un byte). Se utilizan como (REPE y REPZ) en las instrucciones CMPS y SCAS, pero necesitan que ZF = 0 (es decir, que el resultado de la comparación o exploración haya dado diferente o distinto de cero).

En resumen:

Instrucción	Función	Instrucciones
REP	Repetir CX veces	MOVS; STOS
REPE = REPZ	Repetir CX veces mientras ZF = 1	CMPS, SCAS
REPNE, REPNZ	Repetir CX veces mientras ZF=0	CMPS, SCAS

Las operaciones repetitivas con cadenas son interrrumpibles, es decir, el procesador reconocerá la interrupción antes de procesar el siguiente elemento. Esta instrucción genera un byte prefijo. Otros bytes prefijos son:

-· LOCK.

-· especificación de registro de segmento.

Se pueden combinar diferentes prefijos, pero deben desactivarse las interrupciones, pues la interrupción devuelve el control a la instrucción interrumpida o al prefijo anterior a esa instrucción.

REPE/REPZ – Repetir operación de cadena si igual / si cero
(Repeat Equal / Repeat Zero)

Uso: REPE

REPZ

Flags que modifica: Ninguno

Descripción: (Ver instrucción REP)

REPNE/REPNZ – Repetir operación de cadena si disntinto / si no cero
(Repeat Not Equal / Repeat Not Zero)

Uso: REPNE

REPNZ

Flags que modifica: Ninguno

Descripción: (Ver instrucción REP)

RET/RETF – Retornar de un procedimiento *(Return From Procedure)*

Uso: RET nBytes

RETF nBytes
RETN nBytes

Flags que modifica: Ninguno

Descripción: Retorna de un procedimiento, previamente invocado mediante CALL, utilizando como dirección de retorno la dirección salvada en la pila por CALL, que corresponde a la instrucción siguiente a dicha sentencia CALL.

El ensamblador genera un retorno distinto, según se haya definido el procedimiento:

- Procedimiento definido como NEAR:
  En este caso, se quita de la pila una palabra, que corresponde al desplazamiento de la dirección de retorno. Se genera un retorno dentro del mismo segmento.

- Procedimiento definido como FAR:
  En este caso, se quitan de la pila dos palabras, que corresponden al desplazamiento (primera palabra) y al segmento (segunda palabra) de la dirección de retorno. Se genera un retorno a un segmento distinto.

El valor opcional que se especifica en RET es el valor que hay que sumar al registro SP, con objeto de descartar parámetros. Si los parámetros son direcciones de memoria, cada parámetro ocupa:
- 1 palabra (desplazamiento), si procedimiento NEAR.
- 2 palabras (desplazamiento y segmento), si procedimiento FAR.

ROL – Rotación a la izquierda *(Rotate Left)*

Uso: ROL dest,count

Flags que modifica: CF OF

Descripción: Rotar a la izquierda los bits del operando destino el número de bits especificado en el segundo operando. Si el número de bits a desplazar es 1, se puede especificar directamente. Si es mayor que 1, su valor debe cargase en CL y especificar CL como segundo operando.

ROR – Rotación a la derecha *(Rotate Right)*

Uso: ROR dest,count

Flags que modifica: CF OF

Descripción: Rotar a la derecha los bits del operando destino el número de bits especificado en el segundo operando. Si el número de bits a desplazar es 1, se puede especificar directamente. Si es mayor que 1, su valor debe cargarse en CL y especificar CL como segundo operando.

SAHF – Almacenar AH en las banderas *(Store AH Register into FLAGS)*

Uso: SAHF

Flags que modifica: AF CF PF SF ZF

Descripción: Transfiere bits específicos del registro AH a los registro de banderas SF, ZF, PF y CF.

```
SF = bit 7
ZF = bit 6
AF = bit 4
PF = bit 2
CF = bit 0
```

SAL – Despalzamiento aritmético a la izquierda *(Shift Arithmetic Left)*

Uso: SAL dest,count

Flags que modifica: CF OF PF SF ZF (AF indefinidos)

Descripción: SHL y SAL realizan la misma operación y son físicamente la mima instrucción. Desplaza a la izquierda los bits del operando destino el número de bits especificado en el segundo operando. Los bits de la derecha se rellenan con cero.

Si el número de bits a desplazar es 1, se puede especificar directamente. Si es mayor que 1, su valor debe cargarse en CL y especificar CL como segundo operando.

SAR – Despalzamiento aritmético a la izquierda *(Shift Arithmetic Right)*

Uso: SAR dest,count

Flags que modifica: CF OF PF SF ZF (AF indefinidos)

Descripción: Desplaza a la derecha los bits del operando destino el número de bits especificado en el segundo operando. Los bits de la izquierda se rellenan con el bit del signo del primer operando.

Si el número de bits a desplazar es 1, se puede especificar directamente. Si es mayor que 1, su valor debe cargarse en CL y especificar CL como segundo operando.

SBB – Restar con acarreo *(Subtract with Borrow/Carry)*

Uso: SBB dest,src

Flags que modifica: AF CF OF PF SF ZF

Descripción: Resta el operando fuente del operando destino. Resta uno si está activada la bandera de acarreo (CF). El resultado se almacena en el operando destino. Los operando deben ser del mismo tipo (byte o palabra).

SCAS – Explorar cadena *(Scan String -Byte, Word or Doubleword-)*

Uso: SCAS string
SCASB
SCASW

Flags que modifica: AF CF OF PF SF ZF

Descripción: Sirve para explorar una cadena de bytes o palabras. Para ello realiza la operación:

acumulador (AL o AX)-cadena_destino

afectando a las banderas, pero sin almacenar el resultado. La cadena destino está direccionada por DI en el segmento extra, o sea, ES:[DI]. Es decir, que se realiza la operación:

acumulador (Al o AX)-ES:[DI]

Actualiza el registro DI para que apunte al siguiente elemento de la cadena:

- Si los operandos son de tipo byte, la resta es a nivel byte (con AL) y el registro DI cambia una unidad.
- Si los operandos son de tipo palabra, la resta es a nivel palabra (con AX) y el registro DI cambia dos unidades.

Si la bandera de dirección es cero (DI = O), ambos SI y DI se incrementan.

Si DF = 1, ambos se decrementan (véanse las instrucciones CLD y STD).

En la instrucción SCASB se realiza la resta entre bytes AL-ES:[DI].

En la instrucción SCASW se realiza la resta entre palabras AX-ES:[DI].

El operando especificado en SCAS lo utiliza el ensamblador únicamente para verificar el tipo (byte o palabra) y para ver si se ha especificado un registro de segmento. SCAS realiza la operación sin usar realmente el operando de la instrucción.

Se pueden utilizar los prefijos REPE (REPZ) o REPNE (REPNZ). Se aplican para realizar una búsqueda de un elemento de la cadena que cumpla alguna condición determinada. El número de elementos a explorar se especifica en el registro CX.

SHL – Despalzamiento lógico a la izquierda *(Shift Logical Left)*

Uso: SHL dest,count

Flags que modifica: CF OF PF SF ZF (AF indefinidos)

Descripción: SHL y SAL realizan la misma operación y son físicamente la mima instrucción. Desplaza a la izquierda los bits del operando destino el número de bits especificado en el segundo operando. Los bits de la derecha se rellenan con ceros. Si el número de bits a desplazar es 1, se puede especificar directamente. Si es mayor que 1, su valor debe cargarse en CL y especificar CL como segundo operando.

SHR – Despalzamiento lógico a la derecha *(Shift Logical Right)*

Uso: SHR dest,count

Flags que modifica: CF OF PF SF ZF (AF indefinidos)

Descripción: Desplazar a la derecha los bits del operando destino el número de bits especificado en el segundo operando. Los bits de la izquierda se rellenan con cero. Si el número de bits a desplazar es 1, se puede especificar directamente. Si es mayor que 1, su valor debe cargarse en CL y especificar CL como segundo operando.

STC – Poner bandera de acarreo *(Set Carry)*

Uso: STC

Flags que modifica: CF

Descripción: Pone a 1 la bandera de acarreo (CF) sin afectar a ninguna otra bandera.

STD – Poner bandera de dirección *(Set Direction Flag)*

Uso: STD

Flags que modifica: DF

Descripción: Pone a 1 la bandera de acarreo (DF), por lo que en la ejecución de las instrucciones de manejo de cadenas los registros índices SI y/o DI se autodecrementan de modo automático:
-· en 1 si el(los) operando(s) son de tipo byte.
-· en 2 si el(los) operando(s) son de tipo palabra.

STI – Poner bandera de interrupción *(Set Interrupt Flag -Enable Interrupts-)*

Uso: STI

Flags que modifica: IF

Descripción: Pone a 1 la bandera de activación de interrupciones (IF) y activa las interrupciones enmascarables (las que aparecen sobre la línea INTR del procesador). Una interrupción pendiente no será reconocida hasta que no se haya ejecutado la instrucción que sigue a STI.

STOS – Almacenar cadena *(Store String -Byte, Word or Doubleword-)*

Uso: STOS dest
STOSB
STOSW
STOSD

Flags que modifica: Ninguno

Descripción: Transfiere un byte o una palabra desde el registro acumulador AL o AX al operando destino (direccionado por ES:DI). Actualiza el registro DI para que apunte al siguiente elemento de la cadena a almacenar:

- Si el operando es de tipo byte, se transfiere un byte y el registro DI cambia una unidad.
- Si el operando es de tipo palabra, se transfiere una palabra y el registro DI cambia dos unidades.

Si la bandera de dirección es cero (DF = 0), DI se incrementa. Si DF = 1, se decrementa (véanse las instrucciones CLD y STD).

En la instrucción STOSB se transfiere el registro AL al byte ES:[DI]. DI se actualiza en una unidad. En la instrucción STOSW se transfiere el registro AX a la palabra ES:[DI]. DI se actualiza en una unidad. El operando especificado en STOS lo utiliza el ensamblador únicamente para verificar el tipo (byte o palabra) y para ver si se ha especificado un registro de segmento. STOS mueve realmente el registro AL o AX a ES:[DI], sin usar realmente el operando de la instrucción. Cuando se usa con REP, se puede rellenar una cadena con un valor determinado, especificándose el número de elementos de la cadena en el registro CX.

SUB – Restar *(Subtract)*

Uso: SUB dest,src

Flags que modifica: AF CF OF PF SF ZF

Descripción: Resta el operando fuente del operando destino. El resultado se almacena en el operando destino. Los operandos deben ser del mismo tipo (byte o palabra).

TEST – Comparación lógica *(Test For Bit Pattern)*

Uso: TEST dest,src

Flags que modifica: CF OF PF SF ZF (AF indefinidos)

Descripción: Realiza la operación "y lógica" a nivel de bit entre los dos operandos, pero el resultado no se almacena en destino y únicamente se alteran los flags. Tras esta instrucción se pueden consultar las banderas mediante una instrucción de bifurcación condicional.

WAIT – Esperar *(Event Wait)*

Uso: WAIT

Flags que modifica: Ninguno

Descripción: Esta instrucción, junto con ESC, permiten la comunicación con otros coprocesadores. WAIT sirve para poner al procesador en estado de espera, estado que abandona cuando se activa la línea TEST. La línea TEST la activa el coprocesador cuando está libre o preparado para ejecución. WAIT chequea la línea TEST a intervalos fijos (cinco intervalos de reloj).

XCHG – Intercambiar *(eXCHGange)*

Uso: XCHG dest,src

Flags que modifica: Ninguno

Descripción: Intercambia el contenido entre los dos operandos (tipo byte o palabra). No pueden utilizarse registros de segmento como operandos.

XLAT – Traducir *(Translate)*

Uso: XLAT translation-table

Flags que modifica: Ninguno

Descripción: XLAT realiza un posicionamiento sobre una tabla de bytes para obtener el valor correspondiente a un índice. El valor de AL se usa como índice (desplazamiento) sobre la tabla. El valor correspondiente se carga sobre el propio AL. Se supone que el registro BX apunta al comienzo de la tabla. La longitud máxima de la tabla es de 256 bytes.
El primer byte de la tabla tiene desplazamiento cero. La tabla debe ser tipo byte.
La mejor **Descripción:** de esta instrucción es MOV AL,[BX+AL]

XOR – O exclusiva *(Exclusive OR)*

Uso: XOR dest,src

Flags que modifica: CF OF PF SF ZF (AF indefinidos)

Descripción: Operación "o lógico exclusivo" a nivel de bit entre los dos operandos. El resultado no almacena en destino.

La tabla de la operación XOR (para las cuatro combinaciones posibles de bits) es:

Op1	Op2	Dest
0	0	0
0	1	1
1	0	1
1	1	0