Conjunto de Instrucciones del 8088

Las instrucciones del 8088 se pueden clasificar en los siguientes grupos:

- Transferencia de datos
- Aritméticas (Entera binaria, BCD y ASCII)
- Operaciones Lógicas
- Corrimientos y Rotaciones
- Gestión de Bits
- Gestión de Cadenas
- Control del Programa
- Control del Sistema

Transferencia de datos

Las instrucciones de transferencia de datos son las encargadas de mover datos de un sitio a otro de la computadora como pueden ser la memoria, el espacio de E/S y los registros del CPU. Las instrucciones mas comunes en los programas escritos en lenguaje ensamblador son típicamente éstas de transferencia de datos. El conjunto de instrucciones del 8088 incluye 14 instrucciones de transferencia de datos las cuales mueven datos del tamaño de un byte o de una palabra. La Tabla 1 lista estas instrucciones y da una breve nota sobre la operación de cada una de ella.

Tabla 1. Instrucciones de transferencia de datos

Código de operación	Función
MOV	Transfiere una palabra o un byte
PUSH	Transfiere una palabra a la pila
PUSHF	Introduce el registro de bandera a la pila
LAHF	Transfiere el registro de banderas a AH
SAHF	Transfiere AH al registro de bandera
POP	Extrae una palabra de la pila
POPF	Carga el registro de banderas con una palabra de la pila
IN	Lee un dato desde un dispositivo de E/S al acumulador
OUT	Transfiere un dato del acumulador a un dispositivo de E/S
XCHG	Intercambia una palabra o byte
XLAT	Traduce (utiliza una Tabla {Lookup Table})
LEA	Carga la dirección efectiva
LDS	Carga DS y el operando con una dirección de 32 bits
LES	Carga ES y el operando con una dirección de 32 bits

PUSH/POP

PUSH y POP son instrucciones importantes utilizadas para empujar o sacar datos del segmento de pila. El microprocesador 8088 posee la instrucción PUSH y POP puede operar sobre *registros, registro/memoria*, *registro de segmento* y *banderas*.

En la operación con registros, siempre opera sobre registros de 16 bits para transferir la información al segmento de pila. En registro/memoria almacena el contenido de 16 bits de alguna localidad de memoria (2 localidades) al segmento de pila. Cuando se opera sobre registros de segmento es permitido transferir cualquier registro de segmento al segmento de pila pero el segmento de código (CS) no puede ser recuperado de la pila. Cuando las instrucciones PUSH y POP operan sobre el registro de banderas se puede transferir o recuperar el estado de este registro.

PUSH

La instrucción PUSH siempre transfiere 2 bytes de información a la pila. Cuando un dato es metido a la pila, el primer byte (mas significativo) es almacenado en la pila en la localidad direccionada por SP-1. El segundo byte (menos significativo) es almacenado en la pila en la localidad direccionada por SP-2. Después de que el dato se a empujado a la pila, el registro SP es decrementado en 2. La Figura 1 muestra la ejecución de la instrucción PUSH AX, la cual transfiere el contenido de AX en la pila ([SP-1]=AH, [SP-2]=AL y SP=SP-2).

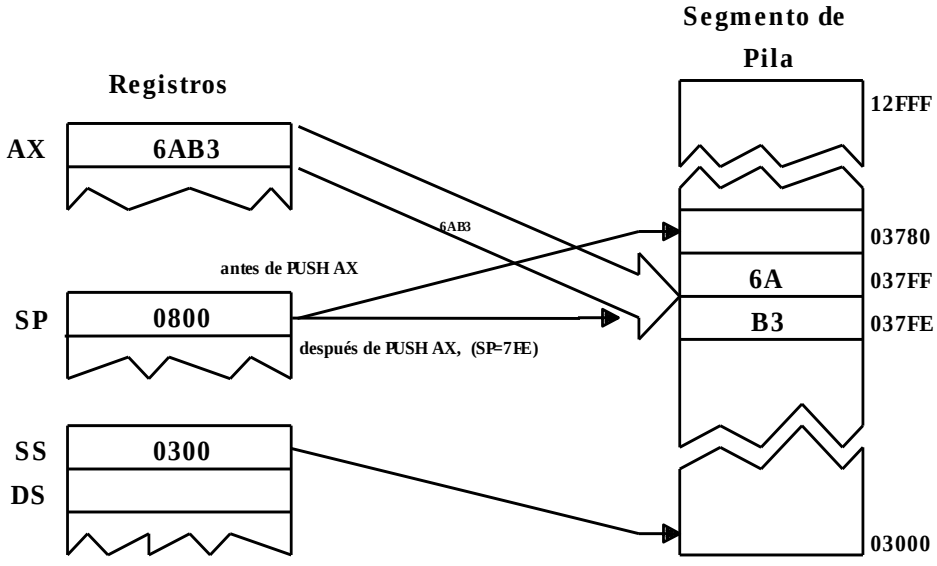

Figura 1. El efecto de una ejecución de una instrucción PUSH AX sobre el registro SP.

POP

La instrucción POP realiza lo inverso a la instrucción PUSH. POP remueve datos de la pila a un registro dado. Supóngase que la instrucción POP BX es ejecutada. El primer byte del dato es removido de la pila , esto de la localidad direccionada por SP y es almacenado en registro BL. El segundo byte es removido de la localidad dada por SP+1 y colocado en el registro BH. Después de que ambos datos fueron extraídos de la pila, el apuntador de pila (SP) es incrementado en 2. La figura 2 muestra la ejecución de la instrucción POP BX.

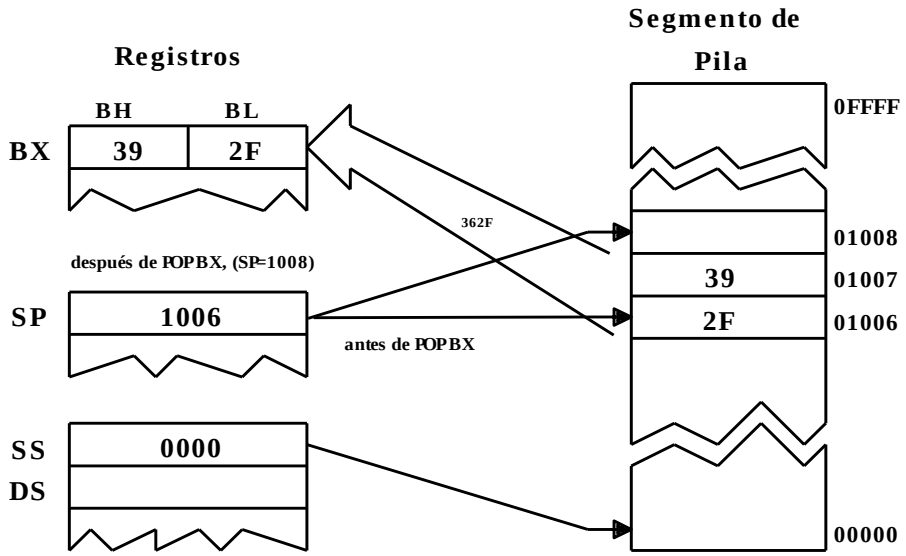

Figura 2. El efecto de una ejecución de una instrucción POP BX sobre el registro SP.

PUSHF

La instrucción PUSHF copia el contenido del registro de bandera a la pila. Al igual que la instrucción PUSH, PUSHF almacena la parte mas significativa del registro de banderas en la localidad SP-1 y la menos significativa en la localidad SP-2. Después del almacenar el registro de banderas, SP se decrementa 2 unidades.

POPF

La instrucción POPF realiza la operación inversa de PUSHF, POPF remueve de la pila un dato de 16 bits que es cargado como el contenido del registro de banderas.

LAHF

Carga el registro AH con banderas, LAHF copia el bytes de orden inferior del registro de banderas en AH. Después de la ejecución de esta instrucción, los bits 7,6,4,2 y 1 de AH son iguales a las banderas S, Z, A, P y C respectivamente.

SAHF

Almacena AH en el registro de banderas, SAHF copia el contenido de AH en el bytes de orden inferior del registro de banderas. Después de la ejecución de esta instrucción las banderas S, Z, A, P y C son iguales a los bits 7, 6, 4, 2 y 1 de AH, respectivamente.

IN y **OUT**

La Tabla 2 muestra una lista de la formas de las instrucciones IN y OUT. Note que solo el registro AL y AX están siendo utilizados para la transferencia de datos entre los dispositivos de E/S y el microprocesador. La instrucción IN transfiere un dato desde un puerto E/S al registro AL o AX, y la instrucción OUT transfiere una dato del registro AX o AL a un puerto de E/S.

Existen dos formas para el direccionamiento de puertos con la instrucción IN y OUT las cuales son: *puerto fijo* y *puerto variable.* El direccionamiento con puerto fijo permite la transferencia de datos entre AL o AX y un puerto de E/S con dirección de 8 bits. Se le llama direccionamiento "puerto fijo" porque la dirección de puerto se almacena con la instrucción (análogamente al direccionamiento inmediato). El direccionamiento puerto variable permite la transferencia de datos entre AL o AX y un puerto de E/S con dirección de 16 bits. A este direccionamiento se de llama puerto variable porque la dirección del puerto se almacena en el registro DX, el cual puede ser cambiado por el programador.

Tabla 2. Instrucciones IN y OUT.

Código de operación	Función
IN AL, pp	Un dato de 8 bits se transfiere del puerto pp a AL
IN AX,pp	Un dato de 16 bits se transfiere del puerto pp a AX
IN AL,DX	Un dato de 8 bits se transfiere del puerto DX a AL
IN AX,DX	Un dato de 16 bits se transfiere del puerto DX a AX
OUT pp,AL	Un dato de 8 bits se transfiere de AL al puerto DX
OUT pp,AX	Un dato de 16 bits se transfiere de AX al puerto pp
OUT DX,AL	Un dato de 16 bits se transfiere de AL al puerto DX
OUT DX,AX	Un dato de 16 bits se transfiere de AX al puerto DX

Nota: pp= un puerto de E/S de con dirección de 8 bits y DX = contiene la dirección de un puerto de E/S con dirección de 16 bits.

XCHG

La instrucción XCHG intercambia el contenido de cualquier registro con el contenido de cualquier otro registro o localidad de memoria. no incluyendo los registros de segmento o intercambios de memoria a memoria. La Tabla 3 muestra las forma de la instrucción XCHG y el tamaño de la misma.

Tabla 3. Instrucción XCHG

Código de operación	Tamaño de la instrucción
XCHG AX,reg	1 byte
XCHG reg,reg	2 byte
XCHG reg,mem	2 byte

XLAT

La instrucción XLAT (translate) convierte el contenido del registro AL en un numero almacenado en una tabla. Esta instrucción se utiliza para realizar una técnica directa de conversión de un código a otro (lookup table). Una instrucción XLAT primero suma el contenido de AL con el contenido del registro BX para formar una dirección del segmento de datos, luego el dato almacenado en esta dirección es cargado en el registro AL. Esta instrucción no posee operando, ya que siempre opera sobre AL.

LEA

La instrucción LEA se utiliza para cargar un registro con la dirección de un dato especificado por un operando (Variable). En el primer ejemplo de la Tabla 4 la dirección de DATA se almacena en el registro AX, cabe notar que la dirección y no el contenido de la dirección DATA se almacena en el registro AX.

En una comparación de la instrucción LEA con MOV se puede observar lo siguiente: LEA BX,[DI] carga la dirección especificada por [DI] (contenido de DI) en el registro BX, MOV BX[DI] carga el contenido de la localidad direccionada por DI en el registro BX.

Tabla 4. Instrucción XCHG

Código de operación	Función
LEA AX,DATA	AX se carga con la dirección de DATA (16 bits)
LDS DI,LIST	DI y DS se cargan con la dirección de LIST (32 bits- DS:DI)
LES BX,CAT	BX y ES se cargan con la dirección de CAT (32 bits -ES:BX)

Ahora ¿por que si la directiva OFFSET realiza la misma tarea que la instrucción LEA está disponible?. La respuesta es que OFFSET se puede usar solamente para un operando sencillo tal como LIST, DATA etc., esta directiva no puede utilizarse en operandos como [DI], LIST[SI]. La directiva OFFSET es mas eficiente que LEA para operandos sencillos, esto es, el microprocesador 8088 toma ocho ciclos de reloj para ejecutar la instrucción en cambio OFFSET requiere solo cuatro ciclos de reloj.

LDS y **LES**

Las instrucciones LDS y LES cargan un registro de 16 bits con un desplazamiento (dirección) y el registro segmento DS o ES con una nueva dirección de segmento. Esta instrucciones utilizan cualquier modo de direccionamiento a memoria válido para seleccionar la localidad del nuevo desplazamiento y nuevo valor de segmento. En la Figura 3 se muestra la operación de la instrucción LDS BX[DI] la cual transfiere la dirección de 32 bits al registro BX y DS que inicia en la localidad de memoria direccionada por el registro DI.

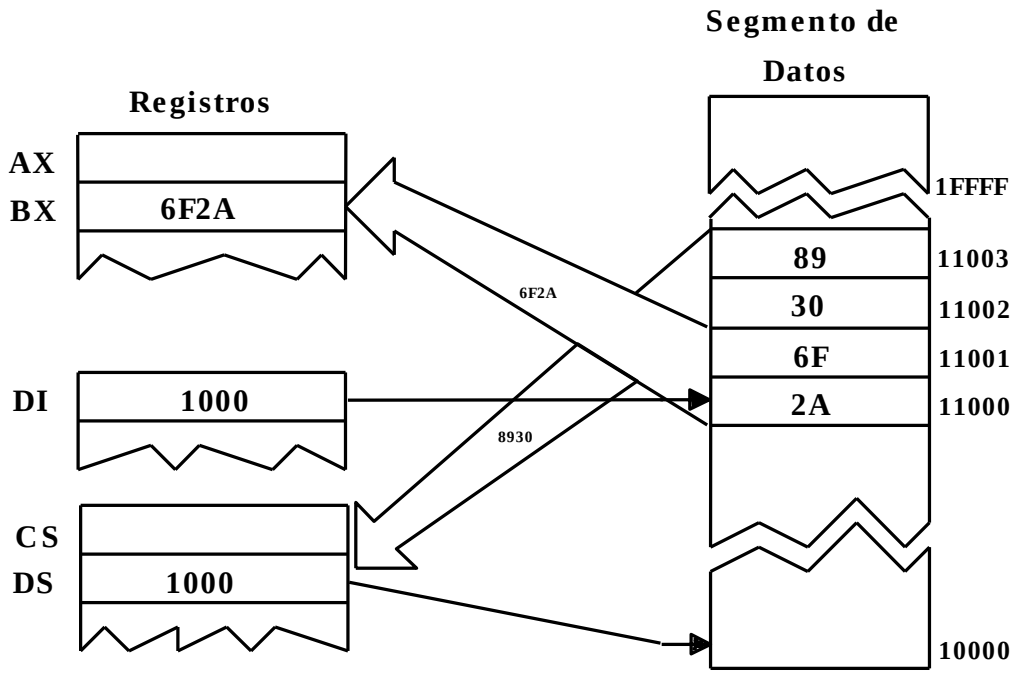

Figura 3. LDS BX,[DI] carga el registro BX con el contenido de la localidad 11000H y 11001H y el registro DS con el contenido de la localidad 11002H y 11003H. Esta instrucción cambia el rango de segmento 1000H-1FFFFH a 89300H-992FFH.

Instrucciones Aritméticas

El 8088 es capaz de sumar, restar, multiplicar y dividir datos ya sean del tipo byte o palabra. Este microprocesador suma y resta utilizando datos (bytes o palabras) con signo o sin signo y datos BCD o ASCII. La multiplicación y división se realiza con operandos de 8 o 16 bits con signo, sin signo o números ASCII. La Tabla 5 muestra las instrucciones aritméticas que posee el microprocesador 8088.

Tabla 5. Instrucciones Aritméticas

Código de operación	Función
ADD	Suma bytes o palabras
ADC	Suma bytes o palabras con acarreo
AAA	Ajusta a ASCII después de la suma
DAA	Ajusta a BCD después de la suma
INC	Incrementa 1 a un byte o palabra
SUB	Resta bytes o palabras
SBB	Resta bytes o palabras con préstamo
NEG	Cambia el signo de un bytes o palabra
CMP	Compara bytes o palabras
AAS	Ajusta a ASCII después de la resta
DAS	Ajusta a BCD después de la resta
DEC	Decrementa 1 a un byte o palabra
MUL	Multiplica bytes o palabras sin signo
IMUL	Multiplica bytes o palabras con signo
AAM	Ajuste a ASCII después de la multiplicación
DIV	Divide bytes o palabras sin signo
IDIV	Divide bytes o palabras con signo
CBW	Convierte un byte a palabra
CWD	Convierte una palabra a doble-palabra
AAD	Ajuste a ASCII antes de la división

SUMA (ADD)

La Tabla 6 indica los modos de direccionamiento soportados por la instrucción ADD (casi todos los modos de direccionamiento mencionados en la unida II). Puesto que hay cerca de 1000 variaciones posibles de la instrucciones de suma en el juego de instrucciones del 8088 es imposible incluirlas todas en esta tabla .

Los único tipos de suma no permitidos son memoria-memoria y registro de segmento. Los registros de segmento sólo se pueden mover, introducir o extraer de la pila.

Suma de registro

El ejemplo 1 muestra un programa que usa la instrucción de suma en los registros para sumar el contenido de varios registros. En este ejemplo se suman los contenidos de AX, BX, CX y DX. Este ejemplo forma un resultado de 16 bits que se almacena en el registro AX.

Tabla 6. Instrucciones de Suma.

Instrucciones	Comentario
ADD AL,BL	AL = AL + BL
ADD CX,DI	CX = CX + DI
ADD CL,44H	CL = CL + 44H
ADD BX,35AFH	BX = BX+35AFH
ADD [BX],AL	DS:[BX] = DS:[BX]+AL
ADD CL,[BP]	CL=CL+SS:[BP]
ADD BX,[SI+2]	BX=BX+DS:[SI+2]
ADD CL,TEMP	CL=CL+[offset TEMP]
ADD BX,TEMP[DI]	BX=BX+[offset TEMP + DI]
ADD [BX+DI],DL	DS:[BX+DI]=DS:[BX+DI]+DL

Ejemplo 1

```
ADD  AX,BX
ADD  AX,CX
ADD  AX,DX
```

Siempre que una instrucción aritmética o lógica se ejecuta, cambia el contenido del registro de banderas. Las banderas denotan el resultado de la operación aritmética. Cualquier instrucción ADD modifica el contenido de las banderas de signo, cero, acarreo, acarreo auxiliar, paridad y sobreflujo.

Suma inmediata. La suma inmediata se usa cuando se suma una constante o dato conocido. En el ejemplo 2 aparece una suma inmediata de 8 bits. En este ejemplo, primero se carga 12H en DL. A continuación se suma un 33H a el 12H. Ambas son instrucciones inmediatas. Después de la suma, el resultado de la suma (45H) se carga al registro DL y las banderas cambian de la siguiente forma:

```
ZF  = 0 resultado distinto de cero.
CF  = 0 no hay acarreo.
AF  = 0 no hay acarreo medio.
SF  = 0 resultado positivo.
PF  = 0 paridad impar.
OF  = 0 no hay sobreflujo.
```

Ejemplo 2

```
MOV  DL,12H
ADD  DL,33H
```

Suma de memoria a registro. Supóngase que una aplicación requiere la suma de un dato de memoria y el registro AL. El ejemplo 3 muestra un ejemplo que suma dos byte consecutivos de datos, almacenados en el segmento de datos en las localidades NUMB y NUMB+1, al registro AL.

Ejemplo 3

```
MOV  DI,OFFSET NUMB    ;direcciona NUMB
MOV  AL,0              ;limpia sum
ADD  AL,[DI]       ;suma NUMB
ADD AL,[DI+1]          ;suma NUMB+1
```

La secuencia de instrucciones primero carga el contenido del registro índice destino DI con la dirección de NUMB. El registro DI usado en este ejemplo direcciona datos en el segmento de datos empezando desde la localidad de memoria NUMB. A continuación la instrucción ADD AL,[DI] suma el contenido de la localidad de memoria NUMB a AL. Esto ocurre porque DI direcciona la localidad de memoria NUMB, y la instrucción suma su contenido al registro AL. A continuación la instrucción ADD AL,[DI+1] suma el contenido de la localidad de memoria NUMB+1 al registro AL. Después que se han ejecutado las dos instrucciones ADD, el resultado aparece en el registro AL formando la suma de los contenidos de NUMB y NUMB+1.

Suma de arreglos. Los arreglos de memoria contienen listas de datos. Supongamos que un arreglo de datos ARRAY contiene 10 byte de datos numerados del elemento 0 al elemento 9. El ejemplo 4 muestra un programa que suma los contenidos de los elementos 3, 5 y 7. (El programa y los elementos del arreglo que suma se eligieron para demostrar el uso de algunos modos de direccionamiento del microprocesador 8088.

Ejemplo 4

```
MOV  AL,0              ;limpia suma
MOV  SI,3              ;direcciona elemento 3
ADD  AL,ARRAY[SI]      ;suma elemento 3
ADD  AL,ARRAY[SI+2]    ;suma elemento 5
ADD  AL,ARRAY[SI+4]    ;suma elemento 7
```

Este ejemplo limpia el registro AL a cero, de tal forma que pueda ser utilizado como acumulador. A continuación se carga el registro SI con un 3 para direccionar inicialmente al tercer elemento del arreglo. La instrucción ADD AL,ARRAY[SI] suma el contenido del elemento número 3 del arreglo a la suma en el registro AL. Las instrucciones que siguen suman los elementos 5 y 7 del arreglo a la suma en el registro AL, usando un 3 en SI más un desplazamiento de 2 para direccionar el elemento 5 y un desplazamiento de 4 para direccionar el elemento 7.

Suma-Incremento. La suma-incremento o simplemente incremento suma 1 a un registro o a una localidad de memoria. cualquier registro o localidad de memoria puede incrementarse, excepto el registro de segmento. La Tabla 7 ilustra las posibles formas de la instrucción incremento, disponible en el microprocesador 8088.

Como con otras instrucciones presentadas anteriormente es imposible mostrar todas las variaciones de la instrucción INC debido al gran número disponible de ellas.

Tabla 7. Instrucciones de Incremento.

Instrucciones	Comentarios
INC BL	BL = BL + 1
INC SP	SP = SP + 1
INC BYTE PTR[BX]	DS:[BX]=DS:[BX] + 1
INC WORD PTR[SI]	DS:[SI]=DS:[SI] + 1
INC DATA1	DS:DATA1=DS:DATA1+1

Con incrementos indirectos a memoria, el tamaño del dato debe ser descrito usando alguna de las directivas Byte PTR ó WORD PTR. La razón es que el programa ensamblador no puede determinar si, por ejemplo, la instrucción INC [DI] es un incremento de un byte o de una palabra. La instrucción INC Byte PTR [DI] indica claramente un dato en memoria de un byte y la instrucción INC WORD PTR [DI] indica un dato en memoria de una palabra.

Ejemplo 5

```
MOV  DI,OFFSET NUMB    ;direcciona NUMB
MOV  AL,0              ;limpia suma
ADD AL,[DI]            ;suma NUMB
INC DI              ;direcciona NUMB+1
ADD  AL,[DI]        ;suma NUMB+1
```

El ejemplo 5 muestra como se puede modificar el programa del ejemplo 3 para usar un incremento direccionando NUMB y NUMB+1. Aquí, la instrucción INC DI cambia el contenido del registro DI de la dirección de compensación NUM a la dirección de compensación NUMB+1. Los programas de los ejemplos 3 y 5 suman los contenidos de NUMB y NUMB+1. La diferencia en los programas es la forma en que esos datos son direccionados por medio de los contenidos del registro DI.

La instrucción incremento afecta los bits de bandera como lo hacen la mayoría de los operadores aritméticos y lógicos, la única diferencia es que la instrucción INC no afecta la bandera de acarreo. La razón es que con frecuencia se usan las instrucciones incremento en programas que dependen del contenido de la bandera de acarreo.

Suma con acarreo. Una instrucción de suma con acarreo realiza la suma del bit de la bandera de acarreo C y un dato (operando). Esta instrucción aparece en software en sumas de números que ocupan más de 16 bit.

La Tabla 8 lista varias instrucciones de suma con acarreo (ADC) con comentarios que explican su operación. Como la instrucción ADD, ADC afecta las banderas después de la suma.

Tabla 8. Instrucciones Suma con acarreo.

Instrucciones	Comentarios
ADC AL,AH	AL = AL + AH + CF
ADC CX,BX	CX = CX + BX + CF
ADC [BX],DH	DS:[BX]=DS:[BX] + DH + CF
ADC BX,[BP+2]	BX=BX+SS:[BP+2]+CF

Suponga que un número de 32 bit contenido en los registros BX y AX se suma a un número de 32 bit contenido en los registros DX y CX. La figura 4 muestra esta suma de tal forma que pueda entenderse la función del acarreo. Esta suma no puede llevarse a cabo sin la suma del bit de acarreo porque el 8088 suma sólo números de 8 y 16 bit. El ejemplo 6 muestra cómo ocurre la suma en un programa. Aquí, el contenido de los registros AX y CX se suma para formar los 16 bit menos significativos de la suma. Esta suma puede o no generar un acarreo. Si la suma es mayor que FFFFH aparece un acarreo. Puesto que es imposible predecir un acarreo, los 16 bit más significativos de la suma son sumados con la bandera de acarreo usando la instrucción ADC. Este programa suma BX-AX a DX-CX con el resultado de la suma en BX-AX.

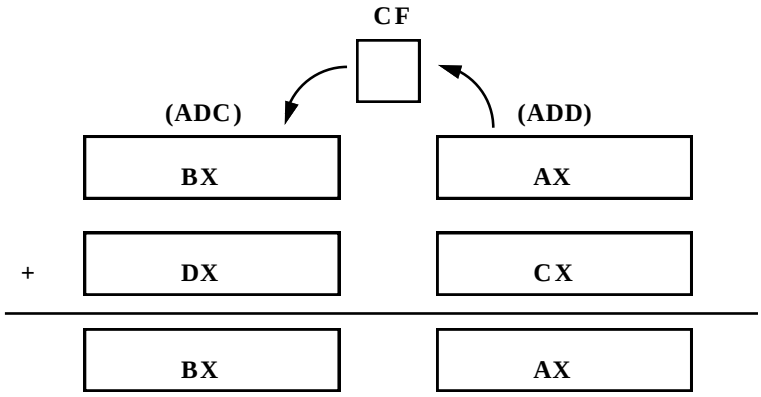

Figura 4. Suma con acarreo mostrando como la bandera de acarreo (CF) une dos sumas de 16 bits para formar suma de 32 bits.

Ejemplo 6

```
ADD  AX,CX
ADC  BX,DX
```

RESTA (SUB)

Son muchas las formas de la instrucción de resta (SUB) que están disponibles en el juego de instrucciones del 8088. Estas formas usan cualquier modo de direccionamiento y datos de 8 o 16 bit. Una forma especial de resta (decremento) substrae una unidad de cualquier registro o localidad de memoria. Al igual que la suma, frecuentemente es necesario hacer restas con números de más de 16 bit, la instrucción de resta con acarreo (SBB) realiza este tipo de substracción.

La Tabla 9 lista los modos de direccionamiento permitidos con la instrucción de resta (SUB). Hay cerca de 1000 instrucciones posibles de resta, demasiadas para ser listadas. A cerca de los únicos tipos de resta no permitidos son resta de memoria a memoria y de registro de segmento. como otras operaciones aritméticas, la instrucción de resta afecta las banderas.

Tabla 9. Instrucciones de Resta.

Instrucciones	Comentarios
SUB CL,BL	CL = CL - BL
SUB AX,SP	AX = AX - SP
SUB DH,6FH	DH = DH - 6FH
SUB AX,0CCCCH	AX=AX-DS:0CCCCH
SUB [DI],CH	DS:[DI]=DS:[DI]-CH
SUB CH,[BP]	CH=CH-SS:[BP]
SUB AH,TEMP	AH=AH-DS:TEMP
SUB DI,TEMP[BX]	DI=DI-TEMP[BX]

Resta de registro. El ejemplo 7 muestra un programa que realiza una resta de registro. Este programa resta los 16 bit contenidos en CX y DX del contenido del registro BX. Después de cada resta, el microprocesador modifica el contenido del registro de banderas. como se mencionó anteriormente, las banderas cambian con la mayoría de instrucciones aritméticas y lógicas.

Ejemplo 7

```
SUB  BX,CX
SUB  BX,DX
```

Resta inmediata. Al igual que la suma, el 8088 también permite operandos inmediatos para la resta. El ejemplo 8 presenta un pequeño programa que resta un 44H de un 22H. Aquí, primero se carga el 22H en CH usando una instrucción de movimiento (asignación) inmediato. A continuación, la instrucción SUB, usando el dato inmediato 44H, substrae el 44H del 22H. Después de la resta la diferencia (DEH) se mueve (asigna) al registro CH. Las banderas cambian de la siguiente manera para esta resta:

ZF = 0 resultado distinto de cero.
CF = 1 préstamo.
AF = 1 medio préstamo.
SF = 1 resultado negativo.
PF = 1 paridad par.
OF = 0 no hay sobreflujo.

Ejemplo 8

```
MOV  CH,22H
SUB  CH,44H
```

Las dos banderas de acarreo (CF y AF) actúan como préstamos después de una resta, en lugar de acarreo, como lo hace después de una suma. Notar en este ejemplo que no hay sobreflujo. Este ejemplo substrae 44H (+68) de un 22H (+34) resultando un DEH (-34). Puesto que el resultado correcto con signo de 8 bit es un -34 no hay sobreflujo en este ejemplo. Un sobreflujo de 8 bit sólo ocurre si el resultado con signo es mayor que +127 o menor que -128.

Resta-decremento. La resta-decremento (o simplemente decremento) (DEC) substrae un 1 del contenido de un registro o del contenido de una localidad de memoria. La Tabla 10 muestra algunas instrucciones de decremento que ilustran decrementos en registros y memoria.

Tabla 10. Instrucciones de Decremento.

Instrucciones	Comentarios
DEC BH	BH = BH - 1
DEC SP	SP = SP - 1
DEC BYTE PTR[DI]	DS:[DI]=DS:[DI]-1
DEC WORD PTR[BP]	SS:[BP]=SS[BP]-1
DEC NUMB	DS:NUMB=DS:NUMB-1

La instrucción de decremento de dato en memoria requiere, ya sea de un Byte PTR o WORD PTR puesto que el ensamblador no puede distinguir un byte de una palabra cuando un registro índice direcciona memoria. Por ejemplo DEC [SI] es vago puesto que el ensamblador no puede determinar si la localidad direccionada por SI es un byte o una palabra. El uso de ya sea de BYTE PTR o WORD PTR indica el tamaño del dato.

Resta con préstamo. Una instrucción resta con préstamo (SBB) funciona como una resta regular, excepto que la bandera de acarreo (CF), la cual actúa como préstamo también se substrae de la diferencia. El uso más común de esta instrucción es en substracciones mayores de 16 bit. Estas substracciones requieren que el préstamo se propague a través de la resta, de la misma forma que con adiciones grandes se requiere que se propague el acarreo. La Tabla 11 lista muchas instrucciones SBB con comentarios que definen su operación. Al igual que la instrucción SUB, SBB afecta las banderas. Notar que la instrucción de resta de memoria inmediata en esta tabla requiere de una directiva ya sea Byte PTR O WORD PTR.

Tabla 11. Instrucciones de Resta con Acarreo.

Instrucciones	Comentarios
SBB AH,AL	AH = AH - AL - CF
SBB AX,BX	AX = AX - BX - CF
SBB CL,3	CL = CL -3 - CF
SBB BYTE PTR[DI],3	DS:[DI]=DS:[I]-3-CF
SBB [DI],AL	DS:[DI]=DS:[DI]-AL-CF
SBB DI,[BP+2]	DI=DI-SS:[BP+2]-CF

Cuando un número de 32 bit almacenado en BX y AX se substrae de un número de 32 bit almacenado en SI y DI, la bandera de acarreo propaga el préstamo entre las dos substracciones de 16 bit requeridas para realizar la operación. La figura 5 muestra como se propaga el préstamo por medio de la bandera de acarreo para esta tarea. El ejemplo 9 muestra como se realiza esta resta por un programa. Con restas grandes, los 16 bit menos significativos son substraídos con la instrucción SUB. Todos los datos siguientes son substraídos usando la instrucción SBB. El ejemplo usa la instrucción SUB para substraer DI de AX, luego SBB para substraer con préstamo SI de BX.

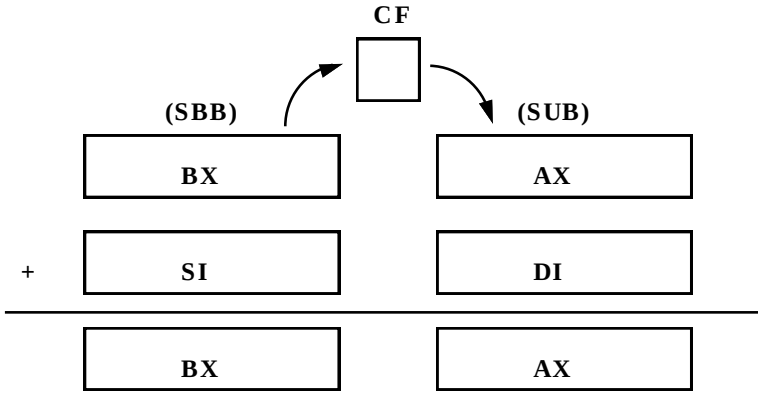

Figura 5. Resta con préstamo mostrando como la bandera de acarreo (CF) propaga el préstamo.

Ejemplo 9

```
SUB  AX,DI
SBB  BX,SI
```

COMPARACION

La instrucción de comparación (CMP) es una resta que no cambia nada excepto las banderas. Se emplea una comparación para verificar el contenido de un registro o de una localidad de memoria con otro valor. Normalmente después de un CMP hay una instrucción de salto condicional, la cual prueba la condición en las banderas.

La Tabla 12 muestra una variedad de instrucciones de comparación que usan los mismos modos de direccionamiento que la suma y resta ya presentadas. De igual forma, la única forma de comparación no permitida es memoria-memoria y registro de segmento.

Ejemplo 10

```
CMP  AL,10h        ;compara con 10H
JAE    SUBER       ;Si 10H o mayor
```

Tabla 12. Instrucciones de Comparación.

Instrucciones	Comentarios
CMP CL,BL	Realiza:CL-BL. CL y BL no cambian
CMP AX,SP	Compara: AX-SP
CMP AX,0CCCCH	Compara:AX-DS:0CCCC
CMP [DI],CH	Compara: DS:[DI]-CH
CMP CL,[BP]	Compara: CL-SS:[BP]
CMP AH,TEMP	Compara: AH-DS:TEMP
CMP DI,TEMP[BX]	Compara: DI-DS:TEMP[BX]

El ejemplo 10 muestra una comparación seguida de una instrucción de salto condicional . En este ejemplo, el contenido de AL se compara con 10H. Las instrucciones de salto condicional que frecuentemente siguen a la instrucción CMP son JA (salta si es mayor) o JB (salta si es menor). Si JA sigue a la comparación, el salto ocurre si el valor en AL es mayor que 10H. Si JB sigue a la comparación, el salto ocurre si el valor en AL es menor que 10H. En este ejemplo, la instrucción JAE sigue a la instrucción CMP. Esto provoca que el programa continúe en la localidad de memoria SUBER si el valor en AL es 10H o menor. Hay también una instrucción JBE (Salta si es menor o igual) que puede ser colocada después de CMP para saltar si el resultado es menor o igual que 10H. El capítulo 5 ofrece más detalles para la instrucción comparación y las instrucciones de salto condicional.

MULTIPLICACIÓN Y DIVISION

Unicamente los microprocesadores más modernos o anteriormente el 8086/8088 contienen instrucciones de multiplicación y división. Los microprocesadores anteriores de 8 bit no eran capaces de multiplicar y dividir directamente. Esas tareas necesitaban de un programa específico que multiplicara o dividiera. Debido a que los fabricantes de microprocesadores notaron esta deficiencia, incorporaron instrucciones de multiplicación y división al juego de instrucciones de los nuevos microprocesadores.

Multiplicaciones

La multiplicación, en el microprocesador 8088, se realiza en byte o palabra y puede ser con signo entero (IMUL) o sin signo (MUL). El resultado después de la multiplicación es siempre un número de doble ancho. Si se multiplican dos números de 8 bit se genera un producto de 16 bit, y si multiplicamos dos números de 16 bit se genera un producto de 32 bit.

Algunas banderas (OF o CF) cambian cuando la instrucción multiplicación se ejecuta y produce resultados predecibles. Las otras banderas también cambian pero sus resultados son impredecibles y por tanto no son usados. En una multiplicación de 8 bit los bit más significativos del resultado son cero, las banderas CF y OF son igual a cero. Estas banderas pueden usarse para encontrar si el resultado es de 8 bit o de 16 bit. En una multiplicación de 16 bit, si los 16 bit más significativos del producto son cero, CF y OF son cero.

Multiplicación de 8 bit. Con la multiplicación de 8 bit, ya sea con signo o sin signo, el multiplicando está siempre en el registro AL. El multiplicador puede estar en cualquier registro de 8 bit o en cualquier localidad de memoria. La multiplicación inmediata no se permite pero sí la instrucción de multiplicación inmediata especial, discutida posteriormente en esta sección, aparece en un programa. La instrucción de multiplicación contiene un operando puesto que siempre multiplica "operando" veces el contenido del registro AL. Un ejemplo es la instrucción MUL BL. La instrucción multiplica el contenido sin signo de BL y deja el producto sin signo en AX. La Tabla 13 lista algunas instrucciones de multiplicación de 8 bit.

Tabla 13. Instrucciones de Multiplicación de 8 bits.

Instrucciones	Comentarios
MUL CL	AX = AL * CL
IMUL DH	AX = AL * DH
IMUL BYTE PTR[BX]	AX = AL * DS:[BX]
MUL TEMP	AX = AL * DS:TEMP

Supóngase que BL y CL contienen cada uno un número sin signo de 8 bit. El número se debe multiplicar para formar un producto de 16 bit almacenado en DX. Esto no se puede

realizar con una sola instrucción puesto que sólo se puede multiplicar un número de veces el registro AL para multiplicación de 8 bit.
El ejemplo 11 muestra un pequeño programa que genera DX=BL x CL. Este ejemplo carga los registros BL y CL con los datos prueba 5 y 10. El producto, un 50, se asigna en DX desde AX después de la multiplicación mediante la instrucción MOV DX,AX.

Ejemplo 11

```
MOV  BL,5          ;carga dato
MOV  CL,10
MOV  AL,CL         ;posición del dato
MUL  BL            ;multiplica
MOV  DX,AX;posiciona el producto
```

Para multiplicaciones con signo el producto está en forma binaria si es positivo y en complemento a dos si es negativo. Lo mismo es cierto para todos los números con signo positivo y negativo en el microprocesador 8088. Si el programa del ejemplo 11 multiplica dos números con signo sólo debe cambiarse la instrucción MUL por IMUL.

Multiplicación de 16 bit. La multiplicación de palabra es similar a la multiplicación de byte. Las diferencias son que AX contiene el multiplicando en lugar de AL y el producto parece en DX-AX en lugar de AX. El registro DX siempre contiene los 16 bit más significativos del producto y AX los 16 bit menos significativos. Como en la multiplicación de 8 bit, la elección del multiplicador se deja a elección del programador. La Tabla 14 muestra varias instrucciones diferentes de 16 bit.

Tabla 14. Instrucciones de Multiplicación de 16 bits.

Instrucciones	Comentarios
MUL CX	DX-AX = AX * CX
IMUL DI	DX-AX = AX * DI
MUL WORD PTR[SI]	DX-AX=AX*DS:[SI]

División

Cómo en la multiplicación, la división en el 8088 se lleva a cabo con números de 8 y 16 bit que son números enteros con signo (IDIV) o sin signo (DIV). El dividendo es siempre de doble ancho, que es dividido por el operando. Esto quiere decir que una división de 8 bit, divide un número de 16 bit entre uno de 8 bit, la división de 16 bit divide un numero de 32 bit entre uno de 16 bit. El 8088 no tiene una instrucción de división inmediata disponible.

Ninguna de las banderas cambian en forma predecible con una división. Una división puede resultar en dos tipos diferentes de errores. Uno de estos es intentar dividir entre cero y el otro es un sobreflujo por división. Un error de sobreflujo por división ocurre cuando se divide un número muy pequeño entre uno muy grande. Por ejemplo, supongamos que AX= 3000 y que lo dividimos entre dos. Puesto que el cociente de una

división de 8 bit se almacena en AL, y 1500 no cabe, entonces el resultado causa un sobreflujo por división. En ambos casos, el microprocesador genera una interrupción si ocurre un error por división.

División de 8 bit. Una división de 8 bit emplea el registro AX para almacenar el dividendo que será dividido entre el contenido de un registro de 8 bit o una localidad de memoria. Después de la división, el cociente se almacena en AL y el residuo en AH. Para una división con signo el cociente es positivo o negativo, pero el residuo siempre es un número entero positivo. Por ejemplo, si AX=0010H (+16) y BL=FDH (-3) y se ejecuta la instrucción IDIV BL, entonces AX queda AX=01FBH. Esto representa un cociente de -5 con un residuo de 1. La Tabla 15 muestra algunas de las instrucciones de división de 8 bit.

Tabla 15. Instrucciones de División de 8 Bits.

Instrucciones	Comentarios
DIV CL	AL= AX / CL, AH=Residuo
IDIV BL	AL= AX / BL, AH=Residuo
DIV BYTE PTR[BP]	AX=AX / SS:[BP], AH=Residuo

Con la división de 8 bit, normalmente los números son de 8 bit. Esto significa que uno de ellos, el dividendo, debe convertirse a uno de 16 bit en AX. Esto se realiza en forma diferente para números con signo y sin signo. Para números sin signo el bit más significativo se debe de poner a cero (cero extendido). Para los números con signo, los 8 bit menos significativos son con signo extendidos en los 8 bit más significativos. En el microprocesador 8088 existe una instrucción especial que signa números extendidos de AL en AH o convierte un número con signo de 8 bit en AL en uno con signo de 16 bit en AX. La instrucción CBW (Convierte de Byte a Palabra) realiza esta conversión.

Ejemplo 12

```
MOV AL NUMB     ;obtiene NUMB
MOV AH,0        ;cero extendido
DIV NUMB1       ;divide entre NUMB1
MOV ANSQ,AL     ;Guarda el cociente
MOV ANSR,AH     ;Guarda el residuo.
```

El ejemplo 12 muestra un programa que divide un byte sin signo de 8 bit de la localidad de memoria NUMB por el contenido sin signo de la localidad NUMB1. Aquí, se almacena el cociente en la localidad de memoria ANSQ y el residuo en la localidad ANSR. Notar que el contenido de la localidad NUMB es traído de memoria y luego se extiende a cero para formar un número sin signo de 16 bit para el dividendo.

Ejemplo 13

```
MOV AL,NUMB     ;obtiene NUMB
CBW             ;cero extendido
IDIV NUMB1      ;se divide entre NUMB1
MOV ANSQ, AL    ;guarda cociente
MOV ANSR,AH     ;guarda residuo
```

El ejemplo 13 muestra el mismo programa básico excepto que los números son números con signo. Esto quiere decir que en vez de extender a cero AL en AH, nosotros extendemos el signo usando la instrucción CBW.

División de 16 bit. La división de 16 bit es igual que la división de 8 bit excepto que en lugar de dividir en AX se divide en DX-AX, es decir, se tiene un dividendo de 32 bit. Después de la división de 16 bit, el cociente se guarda en AX y el residuo en DX. La Tabla 16 lista algunas de las instrucciones de división de 16 bit.

Tabla 16. Instrucciones de División de 16 bits.

Instrucciones	Comentarios
DIV CX	AX=(DX-AX)/CX,DX=Residuo
IDIV SI	AX=(DX-AX)/SI,DX=Residuo
DIV NUMB	AX=(DX-AX)/DS:NUMB, DX=Residuo

Como con la división de 8 bit, el dividendo se debe convertir a una forma adecuada. Si se empieza con un número sin signo en AX, entonces DX se debe de poner a ceros. Si AX es un número con signo de 16 bit la instrucción CWD (Convierte de Palabra a Doble Palabra) extiende el signo a un número con signo de 32 bit.

Ejemplo 14

```
MOV AX, -100        ;carga -100
MOV CX,9            ;carga +9
CWD                 ;extiende el signo
IDIV CX
```

El ejemplo 14 muestra la división de dos números con signo de 16 bit. Aquí, un -100 en AX se divide entre +9 en CX. Antes de la división, la instrucción CWD convierte el -100 en AX a -100 en DX-AX. Después de la división, los resultados parecen en DX-AX como un cociente de -11 en DX y un residuo de 1 en DX.

ARITMETICA BCD Y ASCII

El microprocesador permite la manipulación directa de datos en BCD (Decimal Codificado en Binario) y ASCII (Código Estándar Americano de Intercambio de Información). Esto se realiza mediante instrucciones que ajustan los números para BCD y ASCII.

Las operaciones en BCD se usan en sistemas tales como cajas registradoras y otros sistemas que rara vez requieren aritmética. Las operaciones ASCII se realizan en datos ASCII usados por algunos programas. Hoy en día rara vez se usa la aritmética BCD o ASCII.

ARITMETICA BCD

Para la operación de datos BCD existen dos técnicas: suma y resta. El juego de instrucciones del 8088 cuenta con dos instrucciones que corrigen el resultado de una suma o resta BCD. La instrucción DAA (ajuste decimal después de suma) sigue a la suma BCD, y DAS (ajuste decimal después de resta) sigue a la resta BCD. Ambas instrucciones corrigen el resultado de la suma y resta de tal forma que sea un número en BCD.

Para datos en BCD, el número siempre aparece en formato BCD y se almacena con dos dígitos BCD por byte. Las instrucciones de ajuste sólo funcionan en el registro AL después de una suma o resta.

Instrucción DAA. La instrucción DAA sigue a las instrucciones ADD o ADC para convertir el resultado en un resultado en BCD. Supongamos que DX y BX contienen cada uno números de 4 dígitos binarios. El ejemplo 17 proporciona un programa de muestra que suma los números BCD en DX y BX y almacena el resultado en CX.

Ejemplo 17

```
MOV DX,1234H      ;carga 1,234
MOV BX,3099H      ;carga 3,099
MOV AL,BL         ;suma suma BL y DL
ADD AL,DL
DAA               ;ajusta
MOV CL,AL         ;responde a CL
MOV AL,BH         ;suma BH con DH y CF
ADC AL,DH         ;
DAA               ;ajusta
MOV CH,AL         ;responde a CH
```

Debido a que la instrucción DAA sólo funciona en el registro AL, esta operación se debe realizar en 8 bit. Después de sumar los registros BL y DL, el resultado se ajusta con la instrucción DAA después de ser almacenado en CL. A continuación se suman los registros BH y DH con el acarreo y el resultado nuevamente se ajusta con DAA después de almacenarse en CH. En este ejemplo, 1,234 se suma a 3,099 para generar una suma de 4,333 que se almacena en CX después de la suma. Notar que 1,234 BCD es lo mismo que 1234H.

Instrucción DAS. La instrucción DAS funciona de la misma forma que la instrucción DAA, excepto que sigue a una resta en lugar de una suma. El ejemplo 18 es básicamente el mismo que el ejemplo 17, excepto que resta DX y BX en lugar de sumarlos. La diferencia principal en estos programas es que la instrucción DAA cambia a DAS y las instrucciones ADD y ADC cambian a SUB y SBB.

Ejemplo 18

```
MOV DX,1234H      ;carga 1,234
MOV BX,3099H      ;carga 3,099
MOV AL,BL         ;resta DL de BL
SUB AL,DL
DAS               ;ajusta
MOV CL,AL         ;responde a CL
MOV AL,BH         ;resta DH y CF de BH
SBB AL,DH         ;
DAS               ;ajusta
MOV CH,AL         ;responde a CH
```

ARITMETICA ASCII

Las instrucciones aritméticas en ASCII funcionan con números codificados en ASCII. El rango de números en valor es de 30H a 39H para los números del 0-9. Hay cuatro operaciones empleadas con operaciones aritméticas ASCII: AAA (Ajuste a ASCII después de la suma), AAD (ajuste a ASCII antes de la división), AAM (ajuste a ASCII después de la multiplicación), y AAS (ajusta a ASCII después de la resta). Estas instrucciones usan el registro AX como la fuente y como el destino.

Instrucción AAA. La suma de dos números codificados en ASCII de 1 dígito no resultará para cualquier dato usado. Por ejemplo, si se suma 31H más 39H el resultado es 6AH. La suma ASCII (1+9) debe producir un ASCII de dos dígitos equivalente a 10 decimal, el cual es 31H y un 30H en código ASCII. Si se ejecuta la instrucción AAA después de esta suma, el registro AX contendrá 100H. Aunque esto no es un código ASCII, puede convertirse sumando 3030H., el cual genera 3130H. La instrucción AAA limpia el registro AH si el resultado es menor que 10 y suma un 1 a AH si el resultado es mayor que 10.

Ejemplo 19

```
MOV AX,31H        ;carga el ASCII 1
ADD AL,39H        ;suma el ASCII 9
AAA               ;ajusta
ADD AX,3030H      ;responde en ASCII
```

El ejemplo 19 muestra como se realiza la suma ASCII en el microprocesador 8088. Nótese que se limpia AH antes de efectuar la suma mediante la instrucción MOV AX,31H. El operando de 0031H coloca un 00H en AH y un 31H en AL.

Instrucción AAD. Mientras que todas las demás instrucciones de ajuste van después de su respectiva instrucción, la instrucción AAD aparece antes de una división. La instrucción AAD requiere que el registro AX contenga un número de dos dígitos en BCD "no empaquetado" (no ASCII) antes de ejecutarse. Después de ajustar el registro AX con AAD, se divide por un número BCD desempaquetado para generar un resultado de un sólo dígito en AL con el residuo en AH.

Ejemplo 20

```
MOV AL,5        ;carga el multiplicando
MOV CL,5        ;carga el multiplicador
MUL CL
AAM             ;ajusta
```

El ejemplo 20 muestra como un número 72 en BCD no empaquetado se divide por 9 y produce un coeficiente de 8. El 0702H cargado en el registro AX se ajusta, mediante la instrucción AAD a 0048H. Notar que esto convierte un número BCD no empaquetado en un número binario de tal forma que se pueda dividir mediante la instrucción de división binaria DIV. La instrucción AAD convierte el número BCD no empaquetado entre 00 y 99 en binario.

Instrucción AAM. La instrucción AAM sigue a la instrucción de multiplicación después de multiplicar dos números BCD no empaquetados. El ejemplo 21 muestra un programa que multiplica 5 veces 5. El resultado después de la multiplicación es 0018H en el registro AX. Este es el resultado BCD no empaquetado de 25. Si se suma 3030H a 0205H, esto tiene un resultado ASCII de 3235H.

Ejemplo 21

```
MOV AL,5        ;carga el multiplicando
MOV CL,5        ;carga multiplicador
MUL CL
AAM             ;ajusta
```

Un beneficio de la instrucción AAM es que AAM convertirá de binario a BCD no empaquetado. Si un número binario entre 0000H y 0063H se encuentra en el registro AX, la instrucción AAM lo convierte a BCD. Si AX contiene un 0060H antes de AAM, contendrá 0906H después que AAM se ejecute. Esto es, el BCD no empaquetado equivalente a un 96. Si se suma 3030H a 0906H, se habrá cambiado el resultado a ASCII.

Instrucción AAS. Como otras instrucciones de ajuste a ASCII, AAS ajusta el registro AX después de una resta ASCII. Por ejemplo, supóngase que un 31H se substrae de un 39H . El resultado será 04H, el cual no requiere de corrección. Aquí, la instrucción AAS no modificará a AH ni a AL. Por otro lado, si 38H se substrae de 37H, entonces AL será 09H y el número en AL se decrementará en uno. Este decremento permite substraer números ASCII de múltiples dígitos.

Instrucciones Lógicas y de Manipulación de Bits

El microprocesador 8088 contiene 12 instrucciones de manipulación de bit que incluyen *operaciones lógicas*, *corrimientos* y *rotaciones*. La Tabla 17 muestra cada una de estas instrucciones un una descripción breve de su funcionamiento.

Tabla 17. Instrucciones Lógicas y de Manipulación de bits

Código de operación	Función
AND	Realiza la operación AND a bytes o palabras
OR	Realiza la operación OR a bytes o palabras
XOR	Realiza la operación XOR a bytes o palabras
NOT	Realiza la operación NOT a un byte o palabra
TEST	Verifica bytes o palabras (AND)
SHL/SAL	Corrimiento lógico/aritmético a la izquierda
SHR	Corrimiento lógico a la derecha
SAR	Corrimiento aritmético a la derecha
ROL	Rota un byte o palabra a la izquierda
RCL	Rota un byte o palabra a la izquierda a través del acarreo
ROR	Rota un byte o palabra a la derecha
RCR	Rota un byte o palabra a la derecha a través del acarreo

Instrucciones Lógicas Básicas

Las instrucciones lógicas básicas son: AND, OR y OR exclusiva y NOT. Otra instrucción lógica es TEST, la cual se explica en esta sección del texto debido a que la operación de la instrucción TEST es una forma especial de la instrucción AND. También se explica es la instrucción NEG, la cual es similar a la instrucción NOT.

Las operaciones lógicas permiten manipulación binaria de bit mediante software de bajo nivel. Las instrucciones lógicas permiten que los bit se pongan a cero, a uno, o complementados. El software de bajo nivel tiene la forma de lenguaje máquina o lenguaje ensamblador y frecuentemente controla los dispositivos de entrada-salida en un sistema. Todas las instrucciones lógicas modifican los bit de banderas. Las operaciones lógicas siempre ponen el bit de acarreo y el de sobreflujo en ceros, mientras que otras banderas cambian para reflejar la condición del resultado.

Cuando se manipulan datos binarios en un registro o localidad de memoria, la posición del bit más a la derecha se numera bit 0. Los números de posición de bit se incrementan desde el bit 0 hacia la izquierda hasta el bit 7, para un byte, y hasta el bit 15 para una palabra. Una doble palabra (32 bit) usa la posición 31 al bit de más a la izquierda.

AND

La operación AND realiza una multiplicación lógica como se ilustra en la tabla de verdad de la Figura 6. Aquí, dos bit son multiplicados lógicamente para producir el resultado X.

Como es indicado por la tabla de verdad X es un 1 lógico cuando A y B son unos lógicos. Para cualquier otra combinación de entrada A y B, el resultado es 0 lógico. Es importante recordar que 0 AND CUALQUIER-COSA es 0 lógico siempre, y que 1 AND 1 es siempre 1.

A	*	B	=	X
0		0		0
0		1		0
1		0		0
1		1		1

Figura 6. Tabla de verdad para la operación AND (multiplicación lógica).

La operación AND también limpia los bit de un número binario. La tarea de limpiar un bit en un número binario se llama "enmascaramiento". La Figura 7 ilustra el proceso de enmascaramiento. Notar que los cuatro bit de más a la izquierda se limpian a cero, puesto que "0 AND CUALQUIER-COSA" es 0 lógico. Las posiciones de los bit que son multiplicados lógicamente por "unos", permanecen sin cambio. Esto ocurre puesto que 1 AND 1 siempre da 1, y 1 AND 0 da 0.

X X X X	X X X X	(Patrón desconocido)
0 0 0 0	1 1 1 1	(Mascara)
0 0 0 0	X X X X	(Resultado)

Figura 7. Operación AND de una mascara (0000 1111) con un numero desconocido. Note que los 4 bits mas significativos se borran (0) al realizar la operación AND.

La instrucción AND usa cualquier modo de direccionamiento excepto los direccionamientos memoria-memoria y registro de segmento. Véase la Tabla 18 para una lista de algunas instrucciones AND y algunos comentarios a cerca de su operación.

Tabla 18. Instrucciones AND

Instrucciones	Comentarios
AND AL,BL	AL = AL AND BL
AND CX,DX	CX = CX AND DX
AND CL,33H	CL = CL AND 33H
AND DI,4FFFFH	DI = DI AND 4FFFFH
AND AX,[DI]	AX = AX AND DS:[DI]
AND ARRAY[SI],AL	ARRAY[SI]=ARRAY[SI] AND AL

Ejemplo 22

```
MOV BX,3135H        ;carga un ASCII
AND BX,0F0FH        ;enmascara BX
```

Un número codificado en ASCII puede convertirse a BCD usando la instrucción AND para "desenmascarar" las cuatro posiciones de bit más a la izquierda. Esto convierte el ASCII de 30H a 39H en 0 a 9. El ejemplo 22 muestra un programa que el contenido ASCII de BX en BCD. La instrucción AND en este caso convierte dos dígitos de ASCII a BCD en forma simultánea.

OR

La operación OR desempeña la suma lógica y frecuentemente se le llama función OR inclusiva. La función OR genera un 1 si cualquiera de las entradas es 1. Y aparase un 0 a la salida sólo cuando todas las entradas son 0. La tabla de verdad para la función OR aparece en la Figura 8. Aquí, las entradas A y B se suman lógicamente para producir la salida X. Es importante que recordemos que 1 sumado lógicamente con cualquier valor resulta un 1.

A	+	B	=	X
0		0		0
0		1		1
1		0		1
1		1		1

Figura 8. Tabla de verdad para la operación OR (suma lógica).

La Figura 9 muestra cómo la compuerta OR pone a 1 cualquier bit de un número binario. Aquí, un número desconocido (XXXX XXXX) sumada lógicamente con 0000 1111 produce un resultado de XXXX 1111. Los cuatro bit de más a la derecha se ponen a 1, mientras los cuatro bit de más a la izquierda permanecen sin cambio. La operación OR pone a 1 cualquier bit y la operación AND pone a 0 cualquier bit.

```
   X X X X      X X X X      (Patrón desconocido)
+  0 0 0 0      1 1 1 1      (Patrón de prueba)
-------------------------
   X X X X      1 1 1 1      (Resultado)
```

Figura 9. Operación OR de una patrón de prueba (0000 1111) con un numero desconocido. Note que los 4 bits mas significativos pasan sin cambio (X) al resultado.

La instrucción OR usa cualquiera de los modos de direccionamiento permitidos para cualquier otra instrucción excepto el direccionamiento de registro de segmento. La Tabla 19 ilustra varios ejemplos de instrucciones OR con comentarios a cerca de su operación.

Tabla 19. Instrucciones OR

Instrucciones	Comentarios
OR AH,BL	AH = AH OR BL
OR SI,DX	SI = SI OR DX
OR DH,0A3H	DH = DH OR 0A3H
OR SP,990DH	SP = SP OR 990DH
OR DX,[BX]	DX = DX OR DS:[BX]
OR DATES[DI+2],AL	DATES[DI+2]=DATES[DI+2] OR AL

Supongamos que se multiplican dos números BCD y que se ajustan con la instrucción AAM. El resultado aparece en AX como un número BCD no empaquetado de 2 dígitos. El ejemplo 23 ilustra esta multiplicación y muestra como cambia el resultado a un número `ASCII de dos dígitos, usando la instrucción OR. Aquí, OR AX,3335H convierte 0305H encontrado en AX a 3335H. la operación OR puede reemplazarse por ADD AX,3030H y obtener los mismos resultados.

Ejemplo 23

```
MOV AL,5          ;carga dato.
MOV BL,7
MUL BL
AAM               ;ajusta
OR AX,3030H       ;a ASCII
```

OR EXCLUSIVA

La instrucción OR EXCLUSIVA (XOR) difiere de la OR inclusiva (OR) en que a una condición de 1,1 la función OR produce un 1, mientras que la operación OR EXCLUSIVA produce un 0. La operación OR EXCLUSIVA excluye esta condición mientras la OR INCLUSIVA la incluye.

La Figura 10 muestra la tabla de verdad de la operación OR EXCLUSIVA. (Compare esta figura con la Figura 8 para apreciar la diferencia entre las dos funciones OR). Si las entradas de la función OR EXCLUSIVA son ambas 0 o ambas 1, la salida es cero. Si las entradas son diferentes la salida es 1. Debido a esto, la función OR EXCLUSIVA a veces se llama comparador.

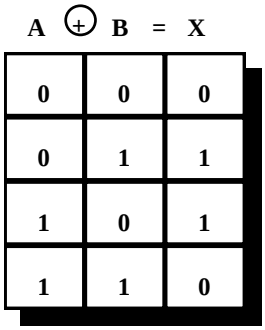

Figura 10. Tabla de verdad para la operación XOR.

La instrucción OR EXCLUSIVA usa cualquier modo de direccionamiento excepto el direccionamiento de registro de segmento. La Tabla 20 muestra varias formas de la instrucción OR EXCLUSIVA con comentarios a cerca de su operación.

Tabla 20. Instrucciones OR Exclusiva

Instrucciones	Comentarios
XOR CH,DH	CH = CH XOR DH
XOR SI,BX	SI =SI XOR BX
XOR CH,0EEH	CH = CH XOR 0EEEH
XOR DI,00DDH	DI = DI XOR 00DDH
XOR DX,[SI]	DX = DX XOR DS:[DI]
XOR DATE[DI+2],AL	DATE[DI+2]=DATE[DI+2] XOR AL

La instrucción OR EXCLUSIVA es usada si tienen que invertir algunos bit de un registro o localidad de memoria. Esta instrucción permite que parte de un número sea invertido o complementado. La Figura 11 muestra cómo se puede invertir sólo una parte de una cantidad desconocida, mediante una XOR. Notar que "1 XOR X" resulta X' y un "0 XOR X" resulta X.

Supongamos que los 10 bit de más a la izquierda del registro BX deben invertirse sin afectar los 6 bit de más a la derecha. La instrucción XOR BX,OFFC0H desempeña esta tarea. La instrucción AND pone a 0 los bit, la instrucción OR los pone a 1 los bit, y ahora, la operación OR EXCLUSIVA invierte los bit.

Estas tres instrucciones permiten a un programa tener control completo sobre cualquier bit almacenado en cualquier registro o localidad de memoria. Esto es ideal para aplicaciones de sistemas de control donde el equipo se debe encender (1), apagar (0), y conmutar de 0 a 1 ó de 1 a 0.

	X X X X	X X X X	(Patrón desconocido)
$\oplus$	0 0 0 0	1 1 1 1	(Patrón de prueba)
	X X X X	$\overline{x}$ $\overline{x}$ $\overline{x}$ $\overline{x}$	(Resultado)

Figura 11. Operación XOR de una patrón de prueba (0000 1111) con un numero desconocido. Note que los 4 bits menos significativos se invertidos ($\overline{x}$) al resultado.

TEST

La instrucción TEST realiza una operación AND. La diferencia es que la instrucción AND cambia el operando destino , mientras que la instrucción TEST no lo hace. Una operación TEST sólo afecta la condición del registro de banderas, el cual indica el resultado de la prueba. La instrucción TEST usa los mismos modos de direccionamiento que la instrucción AND. La Tabla 21 muestra algunas de las formas de la instrucción TEST con comentarios a cerca de su operación en cada forma.

Tabla 21. Instrucciones TEST.

Instrucciones	Comentarios
TEST DL,DH	Realiza: DL AND DH
TEST CX,BX	Realiza: CX AND BX
TEST AX,04H	Realiza: AH AND 04

La instrucción TEST funciona de la misma forma que lo hace la instrucción CMP. La diferencia es que usualmente la instrucción TEST prueba únicamente un bit, mientras que la instrucción CMP prueba el byte o la palabra entera. La bandera ZF (bandera de cero) es 1 si el bit bajo prueba es 0 y ZF=0 (indicando un resultado distinto de cero) si el bit bajo prueba es distinto de cero.

Usualmente la instrucción prueba es seguida ya sea por la instrucción JZ (salta si es cero) o por JNZ (salta si es distinto de cero). Normalmente el operando destino se prueba contra un dato inmediato. El valor del dato inmediato es 1, para probar el bit más a la derecha, 2 para probar el siguiente bit, 4 para probar el tercer bit, y así sucesivamente.

El ejemplo 24 lista un programa que prueba el bit más a la derecha y el bit más a la izquierda del registro AX. Aquí, 1 selecciona el bit más a la derecha y 128 selecciona el bit más a la izquierda. La instrucción JNZ sigue a cada instrucción de prueba para saltar a diferentes localidades de memoria dependiendo del resultado de la prueba realizada. La instrucción JNZ salta a la dirección del operando (RIGHT o LEFT en el ejemplo) si el bit bajo prueba es distinto de cero.

Ejemplo 24

```
TEST AL,1        ;prueba el bit derecho
JNZ RIGHT        ;si es 1
TEST AL,128      ;prueba el bit izquierdo
JNZ LEFT         ;si es 1
```

NOT y NEG

La inversión lógica o complemento a unos (NOT) y la inversión de signo aritmético o complemento a dos (NEG) son las dos últimas funciones lógicas presentadas a excepción de las instrucciones de desplazamiento y rotación de la siguiente sección de este texto. Estas son de las pocas instrucciones que contienen sólo un operando. La Tabla 21 muestra algunas de las variaciones de las instrucciones NOT y NEG. Como con la mayoría de las instrucciones las instrucciones NOT y NEG pueden usar cualquier modo de direccionamiento excepto el direccionamiento a registro de segmento.

Tabla 22. Instrucciones NOT y NEG.

Instrucciones	Comentarios
NOT CH	CH=CMPLT1(CH)
NEG CH	CH=CMPLT2(CH)
NEG AX	AX=CMPLT2(AX)
NOT TEMP	TEMP=CMPLT1(TEMP)
NOT BYTE PTR[BX]	[BX]=CMPLT1([BX])

La instrucción NOT invierte todos los bit de un byte o palabra. La instrucción NEG complementa a dos un número, lo cual significa que el signo aritmético de un número cambia de positivo a negativo o de negativo a positivo. La función NOT se considera una operación lógica y la función NEG se considera una operación aritmética.

CORRIMIENTO Y ROTACION

Las instrucciones de Corrimiento y de rotación manipulan números binarios al nivel bit, como la hacen las instrucciones AND, OR, OR EXCLUSIVA y NOT. Los Corrimientos y rotaciones encuentran su máxima aplicación en software de bajo nivel usado para controlar dispositivos de entrada-salida. El microprocesador 8088 tiene un juego completo de instrucciones de corrimiento y rotación usadas para desplazar o rotar cualquier dato de memoria o registro.

Corrimientos

Las instrucciones de corrimiento posicionan o mueven números a la izquierda o a la derecha dentro de un registro o localidad de memoria. También realizan operaciones aritméticas simples tales como multiplicación por potencias de 2^n (corrimiento a la izquierda) y división por potencias de 2^{-n} (corrimiento a la derecha). El juego de instrucciones del 8088 posee cuatro instrucciones diferentes de corrimiento, dos son corrimientos lógicos y dos son corrimientos aritméticos. Todos las instrucciones de Corrimientos aparecen en la Figura 12.

Notar que en la figura 12 hay dos corrimientos diferentes a la derecha y dos corrimientos diferentes a la izquierda. La instrucción lógica pone un 0 en el bit de más a la derecha para una corrimiento aritmético a la izquierda. Las corrimientos a la derecha lógica y aritmética son idénticos. Los corrimientos a la derecha, lógico y aritmético son diferentes puesto que el corrimiento aritmético a la derecha copia el bit de signo a través del número mientras el corrimiento lógico a la derecha copia un cero a través del número.

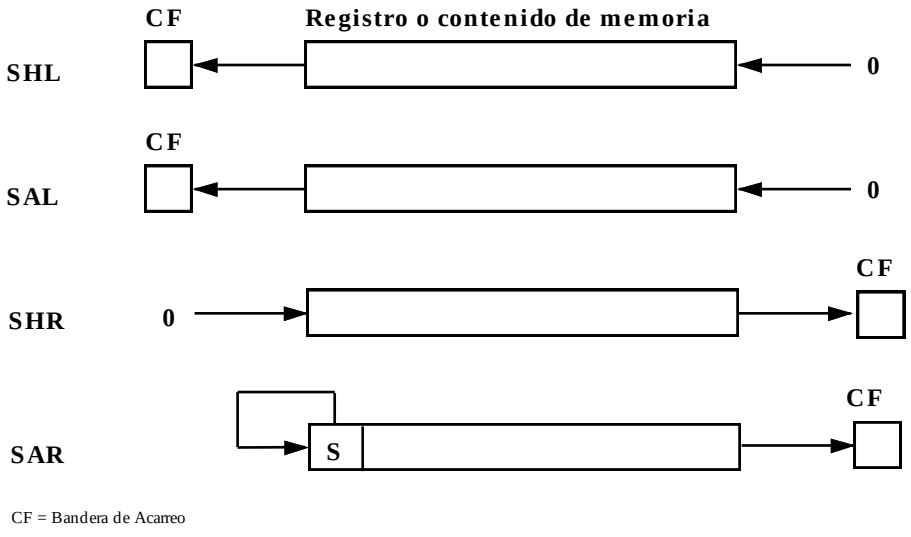

Figura 12. Conjunto de operaciones de corrimiento disponibles en el 8088. Nota: Se puede operar sobre un registro de 8 o 16 bits o una localidad de memoria excepto sobre registros de segmento.

Las operaciones de corrimiento lógico funcionan con números sin signo y los corrimiento aritmético funcionan con números con signo. Las operaciones lógicas multiplican o dividen datos sin signo y los corrimientos aritméticos multiplican o dividen datos con signo. Un corrimiento a la izquierda siempre multiplica por dos cada posición desplazada y un corrimiento a la derecha siempre divide por dos cada bit desplazado.

La Tabla 23 ilustra algunos modos de direccionamiento permitidos para las distintas instrucciones de Corrimiento. Hay dos formas diferentes de desplazamientos que permiten desplazar cualquier registro (excepto el registro de segmento) o localidad de memoria. Un modo usa un Corrimiento inmediato y el otro usa el registro CL para almacenar la cuenta de Corrimientos.

Tabla 23. Instrucciones de Corrimiento.

Instrucciones	Comentarios
SHL AX,1	Corrimiento de AX 1 lugar a la izquierda
SHR BX,1	Corrimiento de BX 1 lugares a la derecha
SAL DATA1,CL	Corrimiento aritmético de DATA CL lugares a la izquierda
SAR SI,CL	Corrimiento aritmético de SI CL lugares a la derecha

Nótese que CL debe almacenar la cuenta de desplazamientos. Cuando CL es la cuenta del corrimiento, no cambia cuando se ejecuta la instrucción de desplazamiento. Notar que la cuenta de corrimientos es una cuenta con módulo 32. Esto significa que una cuenta de corrimientos de 33 desplazará el dato un lugar (33/32 = residuo 1).

El ejemplo 25 muestra cómo el registro DX se desplaza 14 lugares a la izquierda de dos diferentes formas. El ejemplo carga 14 en CL y luego usa CL como cuenta de desplazamiento. Ambas instrucciones cambian el contenido del registro DX lógicamente a la izquierda 14 posiciones o lugares.

Ejemplo 25

```
MOV CL,14
SHL DX,CL
```

Supongamos que el contenido de AX se debe multiplicar por 10, como en el ejemplo 26. Esto puede hacerse de dos diferentes maneras: mediante la instrucción MUL o mediante Corrimientos y sumas. Un número se duplica cuando se desplaza un lugar a la izquierda.

Cuando un número se duplica y luego se suma 8 veces el número, el resultado es 10 veces el número. El número 10 decimal es 1010 en binario. Un 1 lógico aparece en la segunda y en la octava posición. Haciendo uso de esta técnica, puede escribirse un programa para multiplicar por cualquier constante. Esta técnica frecuentemente se ejecuta más rápidamente que la instrucción de multiplicación.

Ejemplo 26

```
;Multiplica AX por 10 (1010)

        SHL AX,1        ;2 veces AX
        MOV BX,AX
        SHL AX,1        ;4 veces AX
        SHL AX,1        ;8 veces AX
        ADD AX,BX       ;10 veces AX

;Multiplica AX por 18 (10010)

        SHL AX,1        ;2 veces AX
        MOV BX,AX
        SHL AX,1        ;4 veces AX
        SHL AX,1        ;8 veces AX
        SHL AX,1        ;16 veces AX
        ADD BX,AX       ;18 veces AX
```

Rotaciones

Las instrucciones de rotación posicionan datos binarios mediante la rotación de la información en un registro o localidad de memoria ya sea de un extremo u otro o a través de la bandera de acarreo. Se usan frecuentemente para desplazar o posicionar números que son mayores de 16 bit en el microprocesador 8088. Las cuatro instrucciones de rotación disponibles aparecen en la Figura 13.

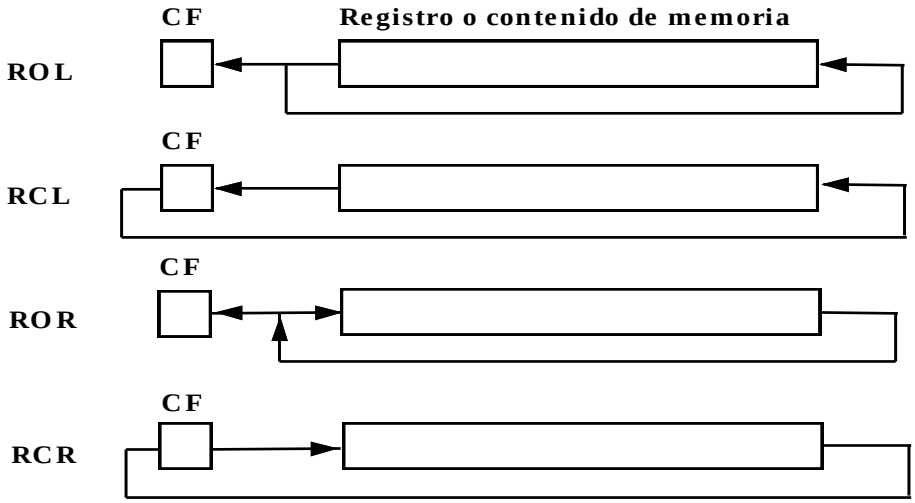

Figura 13. Conjunto de operaciones de rotación disponibles en el 8088.

Los números se rotan a través de un registro y una localidad de memoria y la bandera de acarreo CF sólo a través de un registro y una localidad de memoria. Con cualquier tipo de instrucción de rotación el programador puede usar rotaciones a la derecha o a la izquierda. Los modos de direccionamiento usados para las rotaciones son los mismos que los usados para los Corrimientos. Una cuenta de rotación puede ser inmediata o estar almacenada en el registro CL.

La Tabla 24 muestra algunas de las posibles instrucciones de rotación. Si CL se usa como una cuenta de rotación, no cambia. Como con los Corrimientos, la cuenta en CL es una cuenta de módulo 32.

Tabla 24. Instrucciones de Rotación.

Instrucciones	Comentarios
ROL SI,1	Rota a SI 4 lugares a la izquierda
ROR AX,CL	Rota a AX CL lugares a la derecha
RCL BL,1	Rota a BL 1 lugares a la izquierda, a través de CF
RCR AH,CL	Rota a AH CL lugares a la derecha, a través de CF

Las rotaciones frecuentemente se usan para desplazar números anchos (de más de 32 bit) de izquierda a derecha. El programa listado en el ejemplo 27 desplaza el número de 48 bit en los registros DX, BX y AX hacia la izquierda un lugar binario. Notar que los 16 bit menos significativos (AX) se desplazan primero a la izquierda.
Este Corrimiento, por sí mismo, mueve el bit menos significativo de AX a la bandera de acarreo CF. A continuación la instrucción de rotación BX rota CF en BX y su bit de más a la izquierda pasa a la bandera CF. La última instrucción rota CF en DX y se completa el Corrimiento.

Ejemplo 27.

```
SHL AX,1
RCL BX,1
RCL DX,1
```