Instrucciones aritmético-lógicas

1. Instrucciones de suma

1.1 Enumeración y descripción

- **ADD (sumar)**

  (s) add {R|M}, {R|M|I}

  (e) add al, bl ; al ← al + bl

- **INC (incrementar)**

  (s) inc {R|M}

  (e) inc eax ; eax ← eax + 1

1.2 Las instrucciones de suma y el registro de estado: ejemplos

A continuación se muestran ejemplos de cómo se modifican los bits del registro de estado después de ejecutar algunas instrucciones de suma. A la derecha de cada instrucción se muestran en negrita los valores de los operandos a los que accede las instrucción; y a la derecha de los operandos, se muestra cómo se modifican los bits del registro de estado después de ejecutar cada instrucción.

		eax	ebx	O	C	S	Z
add	al, bl	30AB39**F7**	512A47**09**	0	1	0	1
add	eax, ebx	**30AB39F7**	**512A4709**	1	0	1	0

2. Instrucciones de diferencia

2.1 Enumeración y descripción

- **SUB (restar)**

  (s) sub {R|M}, {R|M|I}

  (e) sub eax, ebx ; eax ← eax - ebx

- **DEC (decrementar)**

  (s) dec {R|M}

  (e) dec eax ; eax ← eax - 1

- **NEG (negar = cambiar de signo = hacer complemento a 2)**

  (s) neg {R|M}

  (e) neg al ; al ← -al

2.2 Las instrucciones de diferencia y el registro de estado: ejemplos

		eax	ebx	O	C	S	Z
sub	al, bl	1AB123**09**	21B434**09**	0	0	0	1
sub	eax, ebx	**1AB12309**	**21B43409**	0	1	1	0
sub	al, bl	00000**70**	000023**B0**	1	1	1	0

3. Instrucciones lógicas

Las instrucciones lógicas realizan operaciones lógicas bit a bit con sus operandos.

3.1 Enumeración y descripción

- **AND (producto lógico)**

  (s) and {R|M}, {R|M|I}

  (e) and al, bl ; al ← al AND bl

- **OR (suma lógica)**

  (s) or {R|M}, {R|M|I}

  (e) or al, bl ; al ← al OR bl

- **XOR (O-exclusiva)**

  (s) xor {R|M}, {R|M|I}

  (e) xor al, bl ; al ← al XOR bl

- **NOT (negación lógica)**

  (s) not {R|M}

  (e) not al ; al ← $\overline{al}$

3.2 Las instrucciones lógicas y el registro de estado

Las instrucciones lógicas de dos operandos (*and*, *or* y *xor*) ponen siempre a '0' los bits CF y OF de registro de estado después de ejecutarse (debe tenerse en cuenta que el estado de estos bits sólo tiene significado tras la ejecución de las instrucciones aritméticas). Los bits SF y ZF se modifican siguiendo los mismos criterios que en las instrucciones aritméticas.

La instrucción lógica de un operando (*not*) no modifica ningún bit del registro de estado tras su ejecución.

3.3 Ejemplos de ejecución de instrucciones lógicas

En la tabla mostrada a continuación se proporcionan ejemplos de ejecución de todas las instrucciones lógicas. A la derecha de cada instrucción se muestran los valores de los operandos sobre los que dichas instrucciones actúan y, después de la barra de separación sombreada en gris, se muestran los resultados obtenidos tras ejecutar las instrucciones y el valor alcanzado por los bits del registro de estado.

		al	bl	al	SF	ZF
and	al, bl	C5	30	00	0	1
or	al, bl	C5	30	05	0	0
xor	al, bl	C5	30	F5	1	0
not	al	C5		3A		

4. Instrucciones de desplazamiento

Las instrucciones de desplazamiento son cuatro: *shl*, *shr*, *sar* y *sal*; y su objetivo es desplazar los bits de un operando un determinado número de posiciones a la izquierda o a la derecha. La estructura de los operandos manejados por estas instrucciones y su significado es idéntico para las cuatro instrucciones. Todas ellas trabajan sobre un operando fuente y un operando destino.

- El *operando destino* contiene el dato que va a ser objeto del desplazamiento y debe ser del tipo, registro o memoria.

- El *operando fuente* determina la cantidad de posiciones que va a ser desplazado el operando destino. El operando fuente sólo puede ser un dato inmediato de 8 bits (*I8*) o bien el registro *CL*.

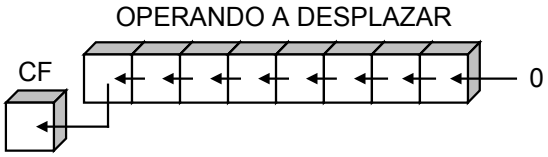
4.1 Enumeración y descripción

- **SHL (Shift Left = desplazamiento a la izquierda)**

  (s) shl {R|M}, {cl|I8}

  (explicación)

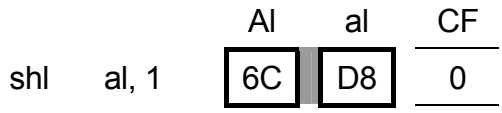
  Se desplazan a la izquierda los bits del operando destino tantas posiciones como indique el operando fuente. El desplazamiento de una posición se realiza de la siguiente forma: el bit de mayor peso del operando se desplaza al bit CF del registro de estado, el resto de los bits se desplazan una posición hacia la izquierda, y la posición de menor peso se rellena con un 0. Este proceso se representa en la figura siguiente.



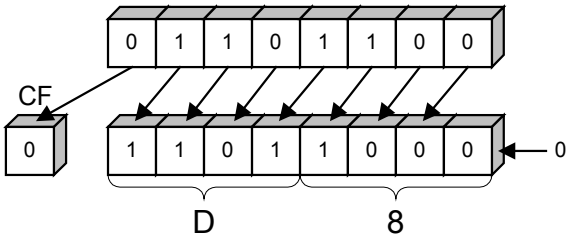
  Si el desplazamiento es de *n* posiciones en vez de una, se repite el proceso anterior *n* veces.

  (ejemplo1)

  Supongamos que se desea desplazar el contenido del registro AL una posición a la izquierda. A continuación se muestra la instrucción de desplazamiento a utilizar, el estado del registro AL antes de ejecutar dicha instrucción y, después de la barra sombreada en gris, el estado de AL y del bit CF del registro de estado después de ejecutar la instrucción.

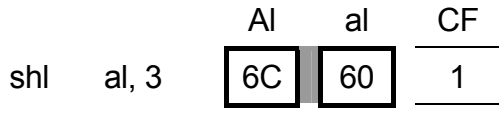


  La instrucción anterior opera de la siguiente forma:

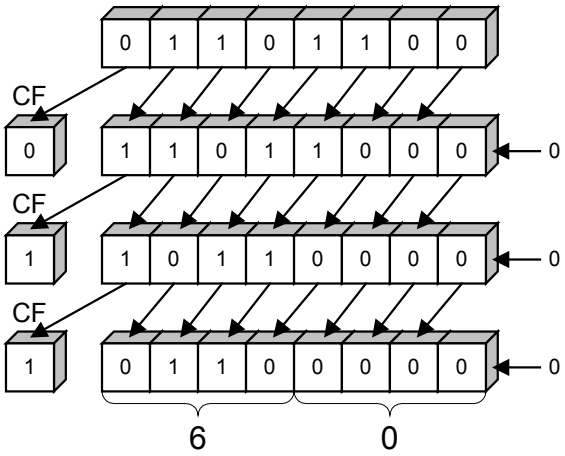


  (ejemplo2)

Desplazar el contenido del registro AL tres posiciones a la izquierda.



La instrucción anterior opera de la siguiente forma:



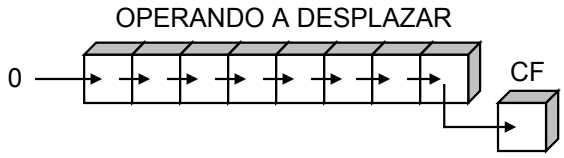
Como puede observarse en la figura, el proceso de desplazamiento se realiza tres veces.

- **SHR (Shift Right = desplazamiento a la derecha)**

  (s) shr {R|M}, {cl|I8}

  (explicación)

  La instrucción *shr* funciona de la misma forma que *shl*, pero desplazando los bits a la derecha en lugar de a la izquierda, tal y como se muestra en la figura siguiente.

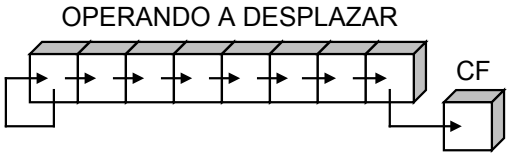


- **SAR (Shift Arithmetic Right = desplazamiento aritmético a la derecha)**

  (s) sar {R|M}, {cl|I8}

  (explicación)

  Esta instrucción desplaza los bits del operando destino a la derecha tantos bits como indique el operando fuente. Esta forma de funcionamiento es similar a la de la instrucción *shr*; sin embargo, ambas instrucciones se diferencian en que *sar*, en vez introducir ceros por la izquierda del operando, replica el bit de mayor peso (bit de signo) en cada desplazamiento. esquemáticamente, la instrucción *sar* trabaja de la siguiente forma:



El desplazamiento a la derecha realizado por la instrucción *sar* recibe el nombre de *aritmético* porque sirve para dividir un operando entre una potencia entera de 2.

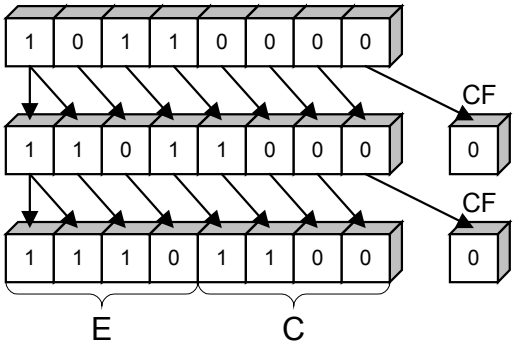
El desplazamiento aritmético a la derecha de un operando (considerado con signo) *n* posiciones equivale a la división entera del operando entre 2^n.

(ejemplo)

Supongamos que se desea desplazar aritméticamente el contenido del registro AL dos posiciones a la izquierda, y que el contenido de este registro antes de realizar el desplazamiento es B0h, que interpretado en complemento a 2 equivale al entero -80. Para llevar a cabo este desplazamiento se ejecuta la instrucción que se muestra a continuación:

```
                          al    al    CF
sar     al, 2           | B0 | EC |  0 |
```

La instrucción anterior opera de la siguiente forma:



Si analizamos el resultado obtenido, ECh, podemos comprobar fácilmente que corresponde al entero -20, que es el resultado de dividir 80 entre 4 (2^2).

Al tipo de división realizada por la instrucción *sar* se le denomina *división entera*. Esto es debido a que sólo se considera el *cociente* de la división. Por ejemplo, Si el registro AL tuviera almacenado el valor 14 decimal y lo desplazamos dos posiciones mediante la instrucción *sar*, el resultado obtenido en el registro AL sería 3, que es el cociente de realizar la división entera, 14/4.

- **SAL** **(Shift Arithmetic Left = desplazamiento aritmético a la izquierda)**

  (s) sal {R|M}, {cl|1}

  (explicación)

  El objetivo de un desplazamiento aritmético a la izquierda es multiplicar un operando, interpretado con signo, por una potencia de 2.

Para llevar a cabo este tipo de desplazamiento, hay que desplazar los bits del operando hacia la izquierda introduciendo ceros por su derecha. En realidad, este tipo de desplazamiento es idéntico al llevado a cabo por la instrucción *shl*; por tanto, *sal* y *shl* son, de hecho, la misma instrucción y se codifican con el mismo código máquina.

El desplazamiento aritmético a la izquierda de un operando (interpretado con signo) *n* posiciones equivale a multiplicar dicho operando por 2^n. El microprocesador detecta si se produce overflow al realizar el desplazamiento, registrándose este hecho en el bit OF del registro de estado.

(ejemplos)

En los ejemplos siguientes se aplica un desplazamiento aritmético a la izquierda de una y dos posiciones al operando CCh, que se encuentra almacenado en el registro AL. CCh interpretado con signo representa el número -52. A la derecha de la barra sombreada en gris se muestra el resultado del desplazamiento y el estado de los bits CF y OF tras su ejecución.

		al	al	CF	OF
sal	al, 1	CC	98	1	0
sal	al, 2	CC	30	1	1

Si interpretamos los resultados anteriores, comprobamos que al desplazar CCh una sola posición obtenemos 98h, que interpretado con signo es -104. Este número corresponde al operando de partida, -52, multiplicado por 2. Como -104 cabe en el rango de los números interpretados con signo codificables con ocho bits, el resultado del desplazamiento no genera *overflow*.

Cuando el operando CCh se desplaza dos posiciones a la izquierda, el objetivo perseguido es multiplicar este operando por 4. Sin embargo, la operación -52 x 4 debería generar como resultado, -208, número no codificable con 8 bits. Debido a esto, el resultado obtenido al ejecutar la segunda instrucción *sal*, 30h, no es un resultado representativo, hecho que se indica con la consiguiente activación del bit OF del registro de estado.

Nomenclatura

(s)	Sintaxis: Indica las expresiones que representan la **s**intaxis genérica de las instrucciones.
(e)	Ejemplo
R	Direccionamiento a registro (8, 16 y 32 bits)
M	Direccionamiento a memoria (8, 16 y 32 bits)
I	Direccionamiento inmediato (8, 16 y 32 bits)
I8	Direccionamiento inmediato de 8 bits