Operadores aritméticos

FLAGS

- São utilizados para monitorar o resultado das operações aritméticas e lógicas
- Os seis principais flags são o ***ZF*** (flag zero)***,CF*** (carry flag)***,OF*** (overflow flag)***,SF*** (sign flag)***,AF*** (auxiliary flag) ***e PF*** (parity flag)
- Quando uma operação aritmética é realizada, alguns dos flags são atualizados para indicar certas propriedades da operação.

Flag *ZF*

- Se o resultado da operação é zero então ***ZF=1***
- Exemplos para ***ZF=1***

```
mov     EAX,1
dec     EAX
```

```
mov     AX,0FFFFH
inc     AX
```

- O ***ZF*** é utilizado também em instruções de comparação e de salto

Flag *CF*

- O flag ***CF*** registra quando o resultado de uma operação aritmética sem sinal está fora da faixa (overflow ou underflow)

```
mov     AL,0FH
add     AL,0F1H
```

- O exemplo acima produz 100h que requer 9 bits para representá-lo. Portanto, o registrador de destino *AL* só comporta 8 bits. (***CF =1***)
- É utilizado em operações de salto

Flag CF

- Faixa de referência para ***CF***

Size (bits)	Range
8	0 to 255
16	0 to 65,535
32	0 to 4,294,967,295

Flag *OF*

- Faz o mesmo papel de CF para números com sinal

Size (bits)	Range
8	−128 to +127
16	−32,768 to +32,767
32	−2,147,483,648 to +2,147,483,647

- O código abaixo resulta em 80h (128d), portanto OF=1

```
mov     AL,72H   ; 72H = 114D
add     AL,0EH   ; 0EH = 14D
```

Flag *SF*

- Indica o sinal do resultado da operação

```
mov     EAX,15
add     EAX,97
```

- O resultado acima é 112d (01110000b). Isso indica que o resultado é um número de sinal positivo.
- Portanto SF=0

Flag *SF*

- O resultado da operação abaixo dará um número negativo -82d (10101110b) ou AEh

```
mov     EAX,15
sub     EAX,97
```

```
  00001111B
+ 10011111B
----------
  10101110B
```

- Desde que o bit mais significativo é 1 então o número é negativo
- Portanto SF=1

Flag *AF*

- Indica se uma operação tem gerado um carry out nos quatro bits (nibble) de baixa ordem.

```
                1  ← carry generated from lower to upper nibble
  43D  = 00101011B
  94D  = 01011110B
 137D  = 10001001B
```

AF=1

```
mov     AL,43
add     AL,84
```

AF=0

```
mov     AL,43
sub     AL,92
```

AF=1

Flag *PF*

- Indica a paridade dos 8-bit resultantes de uma operação.

```
mov     AL,53
add     AL,89
```

```
  53D = 00110101B
  89D = 01011001B
------   --------
 142D = 10001110B
```

PF=1

```
mov     AX,23994
sub     AX,9182
```

```
  23994D = 01011101 10111010B
+ -9182D = 11011100 00100010B
-------    ---------------
 14813D = 00111001 11011100B
```

PF=0

Exercícios

- Avalie no nasm se o valor dos flags correspondem aos exemplos abaixo. Justifique

	Code		AL	CF	ZF	SF	OF	PF
Example 1	`mov`	`AL,-5`						
	`sub`	`AL,123`	80H	0	0	1	0	0
Example 2	`mov`	`AL,-5`						
	`sub`	`AL,124`	7FH	0	0	0	1	0
Example 3	`mov`	`AL,-5`						
	`add`	`AL,132`	7FH	1	0	0	1	0
	`add`	`AL,1`	80H	0	0	1	1	0
Example 4	`sub`	`AL,AL`	00H	0	1	0	0	1
Example 5	`mov`	`AL,127`						
	`add`	`AL,129`	00H	1	1	0	0	1

O comando *Mul*

- ***mul eax, ebx*** ;errado

  Em uma multiplicação o tamanho do seu resultado tende a dobrar. Portanto, atribuir o resultado da multiplicação entre registrados a um registrador de mesmo tamanho não é correto

$$\underset{(255D)}{11111111} \times \underset{(255D)}{11111111} = \underset{(65025D)}{11111110\,11111111}$$

- A sintaxe do ***mul*** é ***mul reg***, onde reg é um registrador.

  Se executar o comando ***mul bh*** então ***ax = al * bh***

O comando ***Mul***

- ***mul bx*** significa que ***dx:ax=ax*bx***
- ***Mul ebx*** significa que ***edx:eax=eax*ebx***

- ***Mul*** signnão possui forma imediata. Por exemplo o comando ***mul 8 ;errado***

Exercício

- Qual é o valor dos bits de flag CF e OF após a execução das instruções abaixo ?

```
mov     AL,10
mov     DL,25
mul     DL
```

```
mov     AL,10
mov     DL,26
mul     DL
```

Imul

- É utilizado para números com sinal
- A sintaxe do ***imul*** é ***imul reg***, onde reg é um registrador.

```
mov     DL,0FFH     ; DL = -1
mov     AL,42H      ; AL = 66
imul    DL
```

Resultado: 1111111110111110

CF=0 ***OF=0***

Imul

```
mov     DL,0FFH     ; DL = -1
mov     AL,0BEH     ; AL = -66
imul    DL
```

Resultado: 0000000001000010 (+66)

CF=0 ***OF=0***

Exercício

- Qual é o valor dos bits de flag CF e OF após a execução das instruções abaixo ?

```
mov     DL,25     ; DL = 25
mov     AL,0F6H   ; AL = -10
imul    DL
```

O comando ***div***

	dividendo	resto	quociente
32-bit	edx:eax	edx	eax
16-bit	dx:ax	dx	ax
8-bit	ax	ah	al

Se ax = 17 e bh = 2 então

O comando ***div bh*** fará ah = 1, al = 8

Exercício

- Indique os valores e registradores que armazenam o quociente e o resto das divisões abaixo

```
mov     AX,251
mov     CL,12
div     CL
```

```
mov     AX,141BH    ; AX = 5147D
mov     CX,012CH    ; CX = 300D
div     CX
```

idiv

- Divisão de números com sinal. Mesma sintaxe do ***div***.
- Quando o dividendo é um número negativo o registrador precisa estender o sinal

`cbw`	(convert byte to word)	Entre al e ax
`cwd`	(convert word to doubleword)	Entre ax e dx
`cdq`	(convert doubleword to quadword)	Entre eax e edx

idiv

- Qual é o valor dos bits de flag CF e OF após a execução das instruções abaixo ?
- Indique os valores e registradores que armazenam o quociente e o resto das divisões abaixo.

```
mov     AX,-5147
cwd                     ; DX = FFFFH
mov     CX,300
idiv    CX
```

idiv

- Qual é o valor dos bits de flag CF e OF após a execução das instruções abaixo ?
- Indique os valores e registradores que armazenam o quociente e o resto das divisões abaixo

```
mov     AL,-95
cbw                     ; AH = FFH
mov     CL,12
idiv    CL
```

Inc e ***Dec***

- ***Inc*** adiciona 1 para o seu operando.

  Ex: Inc ax ;ax=ax+1
- ***Dec*** decrementa 1 para o seu operando.

  Ex: Dec ax ;ax=ax-1

NEG

- Realiza o complemento de 2 de um destino.
- A sintaxe é: ***neg reg***

  neg mem

- ***Exemplos:***

```
Neg j                ;  j = -j

mov ax, k
neg ax
mov j, ax            ; j=-k
```