```
              ¡-----------------------------------------------©
              ¦ RBT   ¦   Curso de Assembly   ¦   Aula Nº 06 ¦
              È-----------------------------------------------¥
```

Por: Frederico Pissarra

```
¡-------------©
¦ ASSEMBLY VI ¦
È-------------¥
```

Instruções aritiméticas são o tópico de hoje. Já discuti, brevemente, os flags e os sistemas de numeração. Aqui vai uma aplicação prática:

¦ Soma:

A soma é feita através das instruções ADD e ADC. A diferença entre elas é que uma faz a soma normalmente e a outra faz a mesma coisa acrecentando o conteúdo do flag CARRY. Eis a sintaxe:

```
+-------------------------------------------------------------------+
¦  ADD AL,10h                                                        ¦
¦  ADC AH,22h                                                        ¦
¦                                                                   ¦
¦  ADD AX,2210h                                                     ¦
+-------------------------------------------------------------------+
```

As duas primeiras instruções fazem exatamente a mesma coisa que a terceira. Note que na primeiria somamos AL com 10h e o resultado ficará em AL (se ocorrer "vai um" nesta soma o flag CARRY será setado). A segunda instrução soma AH com 22h MAIS o carry resultante da primeira instrução e o resultado ficará em AH (novamente setando o flag carry se houver outro "vai um"!). A terceira instrução faz a mesma coisa porque soma 2210h a AX, ficando o resultado em AX e o possível "vai um" no carry.

Todos os flags são afetados após a execução de uma das instruções de soma, exceto: I, D e Trap.

¦ Subtração

Semelhante as instruções de soma, existem duas instruções de subtração: SUB e SBB. A primeira faz a subtração simples e a segunda faz a mesma coisa subtraindo também o conteúdo prévio do flag CARRY (como é uma subtração o CARRY é conhecido como BORROW!).

A sintaxe:

```
+-------------------------------------------------------------------+
¦  SUB AL,1                                                         ¦
¦  SBB AH,0                                                         ¦
¦                                                                   ¦
¦  SUB AX,1                                                         ¦
+-------------------------------------------------------------------+
```

Como no exemplo anterior, as duas primeiras instruções fazem exatamente o que a terceira faz... Os flags afetados seguem a mesma regra das instruções de soma!

¦ Incremento e decremento:

As instruções INC e DEC são usadas no lugar de ADD e SUB se quisermos incrementar ou decrementar o conteúdo de algum registrador (ou de uma posição de memória) de uma unidade. A sintaxe é simples:

```
DEC AX
INC BL
```

Os flags afetados seguem a mesma regra de uma instrução de soma ou uma de subtração!

## ¦ Multiplicação:

Os processadores da família 80x86 possuem instruções de multiplicação e divisão inteiras (ponto flutuante fica pro 8087). Alguns cuidados devem ser tomados quando usarmos uma instrução de divisão (que será vista mais adiante!).

Uma coisa interessante com a multiplicação é que se multiplicarmos dois registradores de 16 bits obteremos o resultado necessariamente em 32 bits. O par de registradores DX e AX são usados para armazenar esse número de 32 bits da seguinte forma: DX será a word mais significativa e AX a menos significativa.

Por exemplo, se multiplicarmos 0FFFFh por 0FFFFh obteremos: 0FFFE0001h (DX = 0FFFEh e AX = 0001h).

Eis a regra para descobrir o tamanho do restultado de uma operação de multiplicação:

| A * B = M | | |
|---|---|---|
| A | B | M |
| 8 bits | 8 bits | 16 bits |
| 16 bits | 16 bits | 32 bits |

A multiplicação sempre ocorrerá entre o acumulador (AL ou AX) e um outro operando. Eis a sintaxe das instruções:

```
MUL BL      ; AX = AL * BL
IMUL CX     ; DX:AX = AX * CX
```

A primeira instrução (MUL) não considera o sinal dos operandos. Neste caso, como BL é de 8 bits, a multiplicação se dará entre BL e AL e o resultado será armazenado em AX.

A segunda instrução leva em consideração o sinal dos operandos e, como CX é de 16 bits, a multiplicação se dará entre CX e AX e o restultado será armazenado em DX e AX. Lembrando que o sinal de um número inteiro depende do seu bit mais significativo!

## ¦ Divisão:

Precisamos tomar cuidado com a divisão pelo seguinte motivo: Se o resultado não couber no registrador destino, um erro de "Division

by zero" ocorrerá (isto näo está perfeitamente documentado nos diversos manuais que li enquanto estudava assembly 80x86... Vim a descobrir este 'macete' numa antiga edição da revista PC MAGAZINE americana). Outro cuidado é com o divisor... se for 0 o mesmo erro ocorrerá!

A divisão pode ser feita entre um número de 32 bits e um de 16 ou entre um de 16 e um de 8, veja a tabela:

| A / B = Q e resto | | |
|---|---|---|
| A | B | Q e resto |
| 32 bits | 16 bits | 16 bits |
| 16 bits | 8 bits | 8 bits |

Assim como na multiplicação o número (dividendo) de 32 bits é armazenado em DX e AX.

Depois da divisão o quociente é armazenado em AL e o resto em AH (no caso de divisão 16/8 bits) ou o quociente fica em AX e o resto em DX (no caso de divisão 32/8 bits).

Exemplo da sintaxe:

```
DIV CX      ; AX=DX:AX / CX, DX=resto
IDIV BL     ; AL=AX / BL, AH=resto
```

O primeiro caso é uma divisão sem sinal e o segundo com sinal. Note os divisores (CX e BL no nosso exemplo).

Na divisão 16/8 bits o dividendo é armazenado em AX antes da divisão... No caso de 32/8 bits DX e AX são usados...

Mais um detalhe: Os flags, depois de uma multiplicação ou divisão não devem ser considerados.