```
i-----------------------------------------------©
¦ RBT   ¦   Curso de Assembly   ¦   Aula Nº 04 ¦
È-----------------------------------------------¥
```

Por: Frederico Pissarra

```
i-------------©
¦ ASSEMBLY IV ¦
È-------------¥
```

Começaremos a ver algumas instruçöes do microprocessador 8086 agora. Existem os seguintes tipos de instruçöes:

¦ Instruçöes Aritiméticas
¦ Instruçöes Lógicas
¦ Instruçöes de Controle de Fluxo de Programa
¦ Instruçöes de manipulaçäo de flags
¦ Instruçöes de manipulaçäo da pilha
¦ Instruçöes de manipulaçäo de blocos
¦ Instruçöes de manipulaçäo de registradores/dados
¦ Instruçöes de Entrada/Saída

Vamos começar com as instruçöes de manipulaçäo de registradores/dados por serem estas as mais fáceis de entender.

```
i---------------©
¦ Instruçäo MOV ¦
È---------------¥
```

MOV tem a finalidade de MOVimentar um dado de um lugar para outro. Por exemplo, para carregar um registrador com um determinado valor. Isto é feito com MOV:

```
+-----------------------------------------------------------------+
¦  MOV AX,0001h                                                    ¦
+-----------------------------------------------------------------+
```

É a mesma coisa que dizer: "AX = 1". Na verdade, movimentamos o valor 1 para dentro do registrador AX.

Podemos mover o conteúdo de um registrador para outro:

```
+-----------------------------------------------------------------+
¦  MOV BH,CL                                                      ¦
+-----------------------------------------------------------------+
```

É a mesma coisa que "BH = CL"!

Os registradores de segmento näo podem ser inicializados com MOV tomando um parametro imediato (numérico). Esses registradores säo inicializados indiretamente:

```
+-----------------------------------------------------------------+
¦  MOV DS,0    ; ERRADO!!!                                        ¦
¦                                                                 ¦
¦  MOV AX,0                                                       ¦
¦  MOV DS,AX   ; CORRETO!                                         ¦
+-----------------------------------------------------------------+
```

Carregar um registrador com o conteúdo (byte ou word, depende da instruçäo!) armazenado em um segmento é simples, basta especificar o offset do dado entre colchetes. Atençäo que o segmento de dados (DS) é assumido por default com algumas excessöes:

```
+-------------------------------------------------------------+
|  MOV AL,[0FFFFh]                                            |
+-------------------------------------------------------------+
```

A instrução acima, pega o byte armazenado no endereço DS:FFFFh e coloca-o em AL. Sabemos que um byte vai ser lido do offset especificado porque AL tem 8 bits de tamanho.
Ao invés de usarmos um offset imediato podemos usar um registrador:

```
+-------------------------------------------------------------+
|  MOV BX,0FFFFh                                              |
|  MOV CH,[BX]                                                |
+-------------------------------------------------------------+
```

Neste caso, BX contém o offset e o byte no endereço DS:BX é armazenado em CH. Note que o registrador usado como indice obrigatoriamente deve ser de 16 bits.

Uma observação quanto a essa modalidade: Dependendo do registrador usado como offset, o segmento default poderá ser DS ou SS. Se ao invés de BX usassemos BP, o segmento default seria SS e não DS - de uma olhada no diagrama de distribuição dos registradores no texto anterior. BP foi colocado no mesmo bloco de SP, indicando que ambos estão relacionados com SS (Segmento de pilha) - Eis uma tabela das modalidades e dos segmentos default que podem ser usados como offset:

Offset usando registros	Segmento default
[SI + deslocamento]	DS
[DI + deslocamento]	DS
[BP + deslocamento]	SS
[BX + deslocamento]	DS
[BX + SI + deslocamento]	DS
[BX + DI + deslocamento]	DS
[BP + SI + deslocamento]	SS
[BP + DI + deslocamento]	SS

O "deslocamento" pode ser suprimido se for 0.

Você pode evitar o segmento default explicitando um registrador de segmento na instrução:

```
+-------------------------------------------------------------+
|  MOV DH,ES:[BX]      ;Usa ES ao invés de DS                 |
|  MOV AL,CS:[SI + 4]  ;Usa CS ao invés de DS                 |
+-------------------------------------------------------------+
```

Repare que tenho usado os registradores de 8 bits para armazenar os dados... Pode-se usar os de 16 bits também:

```
+-------------------------------------------------------------+
|  MOV ES:[BX],AX         ; Poe o valor de AX para ES:BX      |
+-------------------------------------------------------------+
```

Só que neste caso serão armazenados 2 bytes no endereço ES:BX. O primeiro byte é o menos significativo e o segundo o mais signigicativo. Essa instrução equivale-se a:

```
+-------------------------------------------------------------+
```

```
 ¦  MOV ES:[BX],AL            ; Instruçöess que fazem a mesma     ¦
 ¦  MOV ES:[BX + 1],AH        ;coisa que MOV ES:[BX],AX           ¦
 +--------------------------------------------------------------+
```

Repare também que näo é possível mover o conteúdo de uma posição da memória para outra, diretamente, usando MOV. Existe outra instrução que faz isso: MOVSB ou MOVSW. Veremos essas instruçöes mais tarde.

Regra geral: Um dos operandos TEM que ser um registrador! Salvo no caso da movimentaçäo de um imediato para uma posição de memória:

```
 +--------------------------------------------------------------+
 ¦  MOV [DI],[SI]       ; ERRO!                                 ¦
 ¦  MOV [BX],0          ; OK!                                   ¦
 +--------------------------------------------------------------+
```

Para ilustrar o uso da instruçäo MOV, eis um pedaço do código usado pela ROM-BIOS do IBM PS/2 Modelo 50Z para verificar a integridade dos registradores da CPU:

```
 +--------------------------------------------------------------+
 ¦  ...                                                         ¦
 ¦  MOV AX,0FFFFh            ;Poe 0FFFFh em AX                  ¦
 ¦  MOV DS,AX                                                   ¦
 ¦  MOV BX,DS                                                   ¦
 ¦  MOV ES,BX                                                   ¦
 ¦  MOV CX,ES                                                   ¦
 ¦  MOV SS,CX                                                   ¦
 ¦  MOV DX,SS                                                   ¦
 ¦  MOV SI,DX                                                   ¦
 ¦  MOV DI,SI                                                   ¦
 ¦  MOV BP,DI                                                   ¦
 ¦  MOV SP,BP                                                   ¦
 ¦  ...                                                         ¦
 +--------------------------------------------------------------+
```

Se o conteúdo de BP näo for 0FFFFh entäo a CPU está com algum problema e o computador näo pode funcionar! Os flags säo testados de uma outra forma... :)

XCHG

Esta instrução serve para trocarmos o conteúdo de um registrador pelo outro. Por exemplo:

```
 +--------------------------------------------------------------+
 ¦  XCHG    AH,AL                                               ¦
 +--------------------------------------------------------------+
```

Se AH=1Ah e AL=6Dh, após esta instrução AH=6Dh e AL=1Ah por causa da troca...

Pode-se usar uma referência à memória assim como em MOV... com a mesma restrição de que um dos operandos TEM que ser um registrador. Näo há possibilidade de usar um operando imediato.

MOVSB e MOUSW

Essas instruções suprem a deficiência de MOV quanto a movimentação de dados de uma posição de memória para outra diretamente. Antes de ser chamada os seguintes registradores tem que ser inicializados:

```
+-------------------------------------------------------------+
|  DS:SI   <- DS e SI têm o endereço fonte                    |
|  ES:DI   <- ES e DI têm o endereço destino                  |
+-------------------------------------------------------------+
```

Dai podemos executar MOVSB ou MOVSW.

MOVSB move um byte, enquanto MOVSW move um word (16 bits).

Os registradores SI e DI sao incrementados ou decrementados de acordo com o flag D (Direction) - Veja discussão sobre os flags na mensagem anterior. No caso de MOVSW, SI e DI serao incrementados (ou decrementados) de 2 posições de forma que DS:SI e ES:DI apontem sempre para a próxima word.

STOSB e STOSW

Essas instruções servem para armazenar um valor que está em AX ou AL (dependendo da instrução usada) no endereço apontado por ES:DI. Então, antes de ser chamada, os seguintes registradores devem ser inicializados:

```
+-------------------------------------------------------------+
|  AX      -> Valor a ser armazenado se usarmos STOSW         |
|  AL      -> Valor a ser armazenado se usarmos STOSB         |
|  ES:DI   -> Endereço onde o dado será armazenado            |
+-------------------------------------------------------------+
```

Depois da execução da instrução o registrador DI será incrementado ou decrementado de acordo com o flag D (Direction). DI será incrementado de 2 no case de usarmos STOSW, isto garante que ES:DI aponte para a proxima word.

LODSB e LODSW

Essas instruções servem para ler um valor que está no endereço apontado por DS:SI e armazená-lo em AX ou AL (dependendo da instrução usada). Então, antes de ser chamada, os seguintes registradores devem ser inicializados:

```
+-------------------------------------------------------------+
|  DS:SI   -> Endereço de onde o dado será lido               |
+-------------------------------------------------------------+
```

Depois da execução da instrução o registrador SI será incrementado ou decrementado de acordo com o flag D (Direction). No caso de usarmos LODSW, SI será incrementado de 2 para garantir que DS:SI aponte para a próxima word.

Outras instruções de manipulação de registros/dados

Existem ainda as instruções LEA, LES e LDS.

¦ LEA:

LEA é, basicamente, igual a instrução MOV, com apenas uma diferença: o operando "fonte" é um endereço (precisamente: um "offset"). LEA simplesmente calcula o endereço e transfere para o operando "destino", de forma que as instruções abaixo sao equivalentes:

```
+-----------------------------------------------------------------+
¦  MOV     BX,100h                                                ¦
¦  LEA     BX,[100h]                                              ¦
+-----------------------------------------------------------------+
```

Porém, a instrução:

```
+-----------------------------------------------------------------+
¦  LEA     DX,[BX + SI + 10h]                                     ¦     ¦
+-----------------------------------------------------------------+
```

Equivale a:

```
+-----------------------------------------------------------------+
¦  MOV     DX,BX                                                  ¦
¦  ADD     DX,SI          ; DX = DX + SI                          ¦
¦  ADD     DX,10h         ; DX = DX + 10h                         ¦
+-----------------------------------------------------------------+
```

Repare que apenas uma instrução faz o serviço de três!! Nos processadores 286 e 386 a diferença é significativa, pois, no exemplo acima, LEA gastará 3 (nos 286) ou 2 (nos 386) ciclos de máquina enquando o equivalente gastará 11 (nos 286) ou 6 (nos 386) ciclos de máquina! Nos processadores 8088/8086 a diferença näo é tao grande...

Obs:
Consideremos cada ciclo de máquina seria, aproximadamente, num 386DX/40, algo em torno de 300ns - ou 0,0000003s. É uma medida empirica e näo expressa a grandeza real (depende de uma série de fatores näo considerados aqui!).

O operando "destino" é sempre um registrador. O operando "fonte" é sempre um endereço.

¦ LDS e LES

Existe uma forma de carregar um par de registradores (segmento:offset) de uma só vez. Se quisermos carregar DS:DX basta usar a instrução LDS, caso o alvo seja ES, usa-se LES.

Suponhamos que numa posição da memória tenhamos um double word (número de 32 bits) armazenado. A word mais significativa correspondendo a um segmento e a menos signigicativa a um offset (esse é o caso da tabela dos vetores de interrupção, que descreverei com poucos detalhes em uma outra oportunidade!). Se usamos:

```
+-----------------------------------------------------------------+
¦  LES BX,[SI]                                                    ¦
+-----------------------------------------------------------------+
```

O par ES:BX será carregado com o double word armazenado no endereço apontado por DS:SI (repare no segmento default que

discutimos em um texto anterior!). A instrução acima é equivalente a:

```
MOV     BX,[SI+2]
MOV     ES,BX
MOV     BX,[SI]
```

De novo, uma instrução substitui três!

Manipulando blocos... parte I

As instruções MOVSB, MOVSW, STOSB, STOSW, LODSB e LODSW podem ser usadas para lidar com blocos de dados. Para isto, basta indicar no registrador CX a quantidade de dados a serem manipulados e acrescentar REP na frente da instruçao. Eis um trecho de uma pequena rotina que apaga o video em modo texto (80 x 25 colorido):

```
MOV AX,0B800h
MOD ES,AX           ; Poe em ES o segmento do vídeo
MOV DI,0            ; Começa no Offset 0
MOV AH,7            ; AH = atributo do caracter
                    ;      7 = cinza com fundo preto
MOV AL,' '          ; AL = caracter usado para apagar
MOV CX,2000         ; CX = contador (4000 bytes ou
                    ;      2000 words).
REP STOSW           ; Preenche os 2000 words com AX
```

O modificador REP diz a instrução que esta deve ser executada CX vezes. Note que a cada execução de STOSW o registrador DI apontará para a proxima word.

Suponha que queiramos mover 4000 bytes de alguma posição da memória para o video, preenchendo a tela com esses 4000 bytes:

```
MOV AX,0B800h
MOD ES,AX           ; Poe em ES o segmento do vídeo
MOV AX,SEG TABELA
MOV DS,AX           ; Poe em DS o segmento da tabela
MOV SI,OFFSET TABELA ; Começa no offset inicial da tabela
MOV DI,0            ; Começa no Offset 0
MOV CX,4000         ; CX = contador (4000 bytes)
REP MOVSB           ; Copia 4000 bytes de DS:SI para ES:DI
```

Nota: O modificador REP só pode ser preceder as seguintes instruções: MOVSB, MOVSW, STOSB, STOSW, LODSB, LODSW, CMPSB, CMPSW, SCASB, SCASW, OUTSB, OUTSW, INSB, INSW. As instruções nao vistas no texto acima serão detalhadas mais tarde...

Existem mais algumas instruções de manipulação de registradores/dados, bem como mais algumas de manipulação de blocos. Que ficarão para uma próxima mensagem.