

## 9. Arrays e modos de endereçamento

### 9.1 Arrays unidimensionais

**Array uni-dimensional:** é uma lista ordenada de elementos do mesmo tipo.

- por **ordem** entende-se que há um primeiro, um segundo, um terceiro elemento, e assim por diante até um último elemento;
- em matemática, um **array A** de seis elementos é denotado por:

$$A[1], A[2], A[3], A[4], A[5], A[6]$$

- exemplos de *arrays*:

MSG DB 'abcde' ;array composto por um *string* de  
;5 caracteres ASCII

W DW 1010h,1020h,1030h ;array de 3 valores de 16 bits

onde *W* se situa na memória a partir do *offset* de endereço 0200h como

	Offset de endereço	Endereço simbólico	Conteúdo
W ->	0200h	W	10h
	0201h		10h
	0202h	W+2	20h
	0203h		10h
	0204h	W+4	30h
	0205h		10h



## 9.2 Modos de endereçamento

A forma em que um **operando** é especificado numa **instrução** é conhecido como **Modo de Endereçamento**

Modos de enderçamento:

1. **Modo Registrador**: o operando é um registrador da CPU
2. **Modo Imediato**: o operando é uma constante fornecida imediatamente
3. **Modo Direto**: o operando é uma variável declarada no .DATA, ou seja uma posição de memória com endereço bem determinado.
4. **Modo Registrador Indireto**
5. **Modo por Base**
6. **Modo Indexado**
7. **Modo por Base Indexado**

## 9.2.1 Modo Registrador Indireto

- O **offset do endereço** do operando está contido num registrador
- O registrador especificado atua como **ponteiro** para a posição de memória
- Formato do operando: **[registrador]**
- Registradores utilizados:
  - BX, SI, DI** juntamente com o registrador de segmento **DS**  
o endereço é formado por **DS:[registrador]**
  - BP** juntamente com o registrador de segmento **SS**  
o endereço é formado por **SS:[BP]**

Exemplo1: Supondo que SI = 0100h e que a palavra contida na posição de memória de *offset* 0100h seja 1234h:

```
MOV AX,SI      ;AX recebe 0100h
MOV AX,[SI]    ;AX recebe 1234h
```

Exemplo 2: Escreva um trecho de programa que acumule em AX a soma dos 10 elementos do *array* LISTA abaixo:

```
LISTA    DW      10,20,30,40,50,60,70,80,90,100
...
        XOR AX,AX      ;inicializa AX com zero
        LEA SI,LISTA   ;SI recebe o offset de end. de LISTA
        MOV CX, 10     ;contador inicializado no. de elementos
SOMA:   ADD AX,[SI]    ;acumula AX com o elemento de LISTA
        ;apontado por SI
        ADD SI,2       ;movimenta o ponteiro para o próximo
        ;elemento de LISTA (que é do tipo DW)
        LOOP SOMA     ;faz o laço até CX = 0
...

```

### 9.2.2 Modo por Base (*Based Mode*)

- O **offset do endereço** do operando é obtido adicionando um **deslocamento** ao conteúdo de um registrador base
- O **deslocamento** pode ser:
  - o *offset* de endereço de uma variável
  - uma constante (positiva ou negativa)
  - o *offset* de endereço de uma variável mais ou menos uma constante
- Formatos possíveis do operando:
  - [registrador + deslocamento]** ou **[deslocamento + registrador]**
  - [registrador] + deslocamento** ou **deslocamento + [registrador]**
  - deslocamento [registrador]**
- Registradores utilizados:
  - BX** (*base register*) juntamente com o registrador de segmento **DS**
  - BP** (*base pointer*) juntamente com o registrador de segmento **SS**

Exemplo1: Supondo que BX contendo o valor 4d:

```
LISTA    DW    10,20,30,40,50,60,70,80,90,100
MOV AX, [LISTA + BX]    ;resulta que AX = 30
```

Exemplo 2: Escreva um trecho de programa que acumule em AX a soma dos 10 elementos do *array* LISTA abaixo, usando endereçamento por Base:

```
LISTA    DW    10,20,30,40,50,60,70,80,90,100
...
XOR AX,AX    ;inicializa AX com zero
XOR BX,BX    ;limpa o registrador base
MOV CX, 10   ;contador inicializado no. de elementos
SOMA: ADD AX,LISTA+[BX] ;acumula AX com o elemento de LISTA
                        ;apontado por offset de LISTA + BX
ADD BX,2     ;incrementa base
LOOP SOMA    ;faz o laço até CX = 0
...
```

### 9.2.3 Modo Indexado (*Indexed Mode*)

- O **offset do endereço** do operando é obtido adicionando um **deslocamento** ao conteúdo de um registrador indexador
- As opções de **deslocamento** são as mesmas do Modo por Base
- Formatos possíveis do operando:
  - [registrador + deslocamento]** ou **[deslocamento + registrador]**
  - [registrador] + deslocamento** ou **deslocamento + [registrador]**
  - deslocamento [registrador]**
- Registradores utilizados:
  - SI e DI** juntamente com o registrador de segmento **DS**

Exemplo1: Supondo que SI contenha o *offset* de endereço de LISTA:

```

LISTA    DW    10,20,30,40,50,60,70,80,90,100

LEA SI,LISTA    ;SI recebe o offset de LISTA
MOV AX, [SI + 12] ;resulta que AX = 70

```

Exemplo 2: Converta a mensagem abaixo para maiúsculas:

```

MSG      DB    'isto e uma mensagem'
...
MOV CX,19    ;inicializa no. de caracteres de MSG
XOR SI,SI    ;SI = 0
TOPO:    CMP MSG[SI], ' ' ;compara caracter com branco
JE      PULA  ;igual, não converte
AND MSG[SI],0DFh ;diferente, converte para maiúscula
PULA:    INC SI    ;incrementa indexador
LOOP TOPO    ;faz o laço até CX = 0
...

```

## Acessando a pilha por meio de endereçamento por base:

- SP sempre aponta para o topo da pilha
- Problema: como obter cópias de dados existentes na pilha sem modificá-la
- Solução: endereçamento por base usando **BP** (*base pointer*)

Exemplo: Escreva um trecho de programa que carregue AX, BX e CX com as três palavras mais superiores da pilha, sem modificá-la.

```

...
MOV BP,SP           ;BP aponta para o topo da pilha
MOV AX, [BP]        ;coloca o conteúdo do topo em AX
MOV BX, [BP+2]      ;coloca a 2a. palavra (abaixo do topo) em BX
MOV CX, [BP+4]      ;coloca o 3a. palavra em CX
...

```

Neste caso, no Modo de endereçamento por Base:

- BP pode especificar o offset de um endereço na pilha
- pode ser somado um deslocamento (positivo ou negativo)
- o operando final especificado pode navegar para dentro da pilha

## **Segment Override**

Se for necessário acessar dados em outro segmento diferente de DS, por exemplo ES:

```
MOV AX, ES:[SI]
```

Pode-se utilizar **segment override** nos modos de registro indireto, por base e indexado.

## Algumas diretivas

### PTR

A instrução `MOV [BX],1` é ilegal, pois o Montador não pode determinar se `[BX]` aponta para uma informação na memória do tipo byte ou do tipo word.

Soluções:

```
MOV  BYTE PTR [BX],1 ;define o destino como byte
MOV  WORD PTR [BX],1 ;define o destino como word
```

### LABEL

É uma **pseudo-instrução** para alterar tipo de variáveis (**override**)

Exemplo:

```
TEMPO      LABEL  WORD
HORAS      DB     10
MINUTOS    DB     20
```

Esta declaração feita no segmento de dados (`.DATA`), permite que:

- TEMPO e HORAS recebam o mesmo endereço pelo Montador;
- TEMPO (16 bits) engloba HORAS e MINUTOS (8 + 8 bits);
- são legais as seguintes instruções:

```
MOV  AH,HORAS
MOV  AL,MINUTOS
```

e

```
MOV  AX,TEMPO ;produz o mesmo efeito das acima
```

### 9.3 Arrays bidimensionais

- Um **array bidimensional**, ou matriz, é um **array de arrays**.
- Arranjo idealizado em linhas e colunas:
  - matriz **A** de 3 linhas por 4 colunas
  - cada elemento é identificado como **A[i,j]**
  - **i** varia de 1 a 3 e **j** varia de 1 a 4
- A memória se organiza apenas em uma dimensão (**arranjo linear**)
- **Como organizar os dados?**

**ordem por linha:** os dados da linha 1 são organizados em ordem crescente, seguidos dos dados da linha 2, sucessivamente, até a linha N;

ex: A[1,1] a A[1,4], A[2,1] a A[2,4], A[3,1] a A[3,4] -> **método mais usual**

**ordem por coluna:** idem, organizando-se por colunas

ex: A[1,1] a A[3,1], A[1,2] a A[3,2], A[1,3] a A[3,3], A[1,4] a A[3,4]

**Exemplo:** Supondo a **matriz A (3 x 4)** abaixo, organizar os dados no segmento de dados em ordem por linha e ordem por coluna.

	1	2	3	4
1	10	20	30	40
2	50	60	70	80
3	90	100	110	120

organização por linha:

```

...
.DATA
A      DW      10,20,30,40
        DW      50,60,70,80
        DW      90,100,110,120
...

```

organização por coluna:

```

...
.DATA
A      DW      10,50,90
        DW      20,60,100
        DW      30,70,110
        DW      40,80,120
...

```

### 9.3.1 Modo por Base Indexado (*Based Indexed Mode*)

- O **offset do endereço** do operando é obtido somando:
  - o conteúdo de um **registrador de base** (BX ou BP);
  - o conteúdo de um **registrador índice** (SI ou DI);
  - opcionalmente, o *offset* de endereço de uma **variável**;
  - opcionalmente, uma **constante** (positiva ou negativa).
- Formatos possíveis do operando:
  - variável [reg\_de\_base] [reg\_índice]**
  - variável [reg\_de\_base + reg\_índice + constante]**
  - constante [reg\_de\_base + reg\_índice + variável]**
  - [reg\_de\_base + reg\_índice + variável + constante]**
- Registradores utilizados:
  - SI, DI e BX** juntamente com o registrador de segmento **DS**
  - SI, DI e BP** juntamente com o registrador de segmento **SS**

Exemplo1: Supondo a variável LISTA abaixo, e que SI = 14 e BX = 2:

```
LISTA    DW    10,20,30,40,50,60,70,80,90,100
```

```
MOV AX, LISTA [BX][SI]    ;resulta que AX = LISTA+2+14 = 90
```

Exemplo 2: Suponha que A é uma matriz 3X4 com elementos de tipo DW. Escreva um trecho de programa que zere os elementos da 2a. linha de A.

```

...
MOV BX,8                ;BX indica o 1o. elemento da linha 2
MOV CX,4                ;CX contem o número de elementos de linha
XOR SI,SI               ;inicializa o indexador de coluna
LIMPA: MOV A [BX][SI], 0 ;carrega zero no operando calculado
      ADD SI,2           ;incrementa 2 em SI -> tipo DW = 2 bytes
      LOOP LIMPA        ;faz o laço até que CX seja zero

```

...

## 9.4 A instrução XLAT

Instrução **sem operando** que **converte** um valor binário de tipo *byte* em outro valor binário, que é procurado numa **tabela de conversão**.

- O valor a ser convertido (1 byte) é assumido estar em **AL**
- O *offset* do endereço onde se inicia a tabela de conversão é assumido estar em BX.
- A instrução **XLAT**:
  - **soma** o conteúdo de AL ao *offset* dado por BX e localiza a posição resultante dentro da tabela;
  - **substitui** o conteúdo AL pelo valor localizado na tabela.

Exemplo: Conversão do conteúdo binário de 8 bits de AL, suposto ser um valor hexadecimal, para o carácter ASCII correspondente.

```
.DATA
Tabela    DB    30h,31h,32h,33h,34h,35h,36h,37h,38h,39h
           DB    41h,42h,43h,44h,45h,46h
...
.CODE
...
MOV AL,0Ch    ;exemplo, converter 0Ch para carácter ASCII 'C'
LEA BX,TABELA ;BX recebe o offset de TABELA
XLAT          ;é feita a conversão e AL recebe 43h = 'C'
...
```

## Exercícios sugeridos

1) Suponha que  $AX = 0500h$ ,  $BX = 1000h$ ,  $SI = 1500h$  e  $DI = 2000h$  e que:

*offset* de endereço 1000h contem 0100h  
*offset* de endereço 1500h contem 0150h  
*offset* de endereço 2000h contem 0200h  
*offset* de endereço 3000h contem 0400h  
*offset* de endereço 4000h contem 0300h

e a variável BETA é do tipo DW com *offset* de endereço 1000h. Para cada uma das seguintes instruções, aplicadas independentemente para as condições iniciais acima, se forem legais (corretas), dê o *offset* de endereço calculado ou registrador indicado e o valor armazenado no destino.

- |                    |                            |
|--------------------|----------------------------|
| a) MOV DI,SI       | f) ADD [SI],[DI]           |
| b) MOV DI,[DI]     | g) ADD BH,[BL]             |
| c) ADD AX,[SI]     | h) ADD AH,[SI]             |
| d) SUB BX,[DI]     | i) MOV AX,[BX + DI + BETA] |
| e) LEA BX,BETA[BX] |                            |

2) Escreva um programa que leia uma linha de texto a partir do teclado e exiba este mesmo texto convertendo todas as letras em maiúsculas. O texto inicial pode conter qualquer caracter textual: letras maiúsculas, minúsculas, dígitos numéricos e caracteres de pontuação. Dica: defina uma tabela com apenas os códigos hexadecimais referentes aos caracteres ASCII letras minúsculas e utilize convenientemente a instrução XLAT.

3) Escreva uma subrotina para exibição no monitor de vídeo de números hexadecimais, a partir do conteúdo binário em BX, com nome de SAIHEX2 (uma alternativa ao trecho de programa apresentado na página 6.19). Utilize

XLAT juntamente com uma tabela adequada para converter cada 4 dígitos binários de BX em um dígito hexa e exiba este dígito no monitor.

4) Suponha que a variável CLASSE armazene na memória o nome e as quatro notas (notas inteiras entre 0 e 100), referentes a quatro alunos de um curso, sendo definida como:

CLASSE	DB	'Antonio Carlos	,50,49,19,25
	DB	'Carlos Roberto	,35,74,100,86
	DB	'Maria Carla	,45,57,63,67
	DB	'Valeria Cristina	,35,0,34,23

Suponha que o campo de nome contenha 20 *bytes* reservados, ou seja, a primeira nota está armazenada somente no *byte* 21 de cada linha. Escreva um trecho de programa, baseado na definição da variável CLASSE acima, que exiba no monitor de vídeo o **nome do aluno**, seguido de sua **média final** (que será truncada para o inteiro inferior à média real), seguido da palavra **aprovado** se a média for maior ou igual a 50 e **reprovado**, se menor que 50.