Atividade 7

O objetivo desta atividade é permitir que o aluno seja capaz de criar um programa em linguagem assembly que utilize vetores e instruções de acesso à memória.

Parte 1 – Declaração de variáveis e vetores

A declaração de variáveis em linguagem assembly é sempre feita no segmento .DATA do programa, declarando:

NOME TIPO VALOR

NOME: nome da variável;

TIPO: tipo da variável, que pode ser DB (1 byte), DW (2 bytes), DD (4 bytes), DQ (8 bytes) ou DT (10 bytes);

VALOR: o valor inicial da variável. O valor ? pode ser utilizado para não inicializar a variável com um valor pré-definido e neste caso a variável conterá um lixo de memória.

Ex:

```
.DATA
  ; define uma variável chamada X que ocupa 1 byte na memória com valor não inicializado
   X            DB      ?
  ; define uma variável chamada IDADE que ocupa 1 byte na memória com valor 27
   IDADE        DB      27
  ; define uma variável chamada SEXO que ocupa 1 byte na memória com valor “M”
   SEXO         DB      “M”
  ; define uma variável chamada SALARIO que ocupa 2 bytes na memória com valor 2870
   SALARIO      DW      2870
```

A declaração de vetores em linguagem assembly é sempre feita no segmento .DATA do programa, declarando:

NOME TIPO VALOR1,VALOR2,...,VALORn

NOME: nome do VETOR;

TIPO: tipo de cada elemento, que pode ser DB (1 byte), DW (2 bytes), DD (4 bytes), DQ (8 bytes) ou DT (10 bytes);

VALORES: o valor inicial de cada posição do vetor. O valor ? pode ser utilizado para não inicializar uma posição do vetor com um valor pré-definido e neste caso essa posição conterá um lixo de memória.

Ex:

```
.DATA
  ; define um vetor chamado IDADES de 5 posições
  ; onde cada posição ocupa 1 byte na memória com os valores não inicializados
   IDADES       DB    ?,?,?,?,?
  ; define um vetor chamado LETRAS de 4 posições
  ; onde cada posição ocupa 1 byte na memória com os valores “A”, “B”, “C” e “D”
   LETRAS       DB    “ABCD”
  ; define um vetor chamado SALARIOS de 3 posições
  ; onde cada posição ocupa 2 bytes na memória com os valores 2870, 3500  e 1800
   SALARIOS   DW   2870,3500,1800
```

Uma forma alternativa de declarar vetores sem ter que inicializar cada posição individualmente é utilizando a diretiva DUP, declarando:

NOME TIPO Qtde DUP(Valor)

NOME: nome do VETOR;

TIPO: tipo de cada elemento, que pode ser DB (1 byte), DW (2 bytes), DD (4 bytes), DQ (8 bytes) ou DT (10 bytes);

Qtde: quantidade de posições do vetor;

Valor: valor de cada posição do vetor. O valor ? pode ser utilizado para não inicializar uma posição do vetor com um valor pré-definido e neste caso essa posição conterá um lixo de memória.

Ex:

```
.DATA
  ; define um vetor chamado NOTAS de 50 posições
  ; onde cada posição ocupa 1 byte na memória e todas são inicializadas com o valor 0
   NOTAS      DB    50 DUP(0)
  ; define um vetor chamado PESOS de 20 posições
  ; onde cada posição ocupa 2 bytes na memória e todas não são inicializadas
   PESOS      DW    20 DUP(?)
  ; define um vetor chamado NOME de 15 posições
  ; onde cada posição ocupa 1 byte na memória e todas são inicializadas com o valor “A”
   NOME       DB    15 DUP(“A”)
```

Parte 2 – Acesso à memória

O acesso à memória em linguagem assembly pode ser feito de diversas formas distintas. Uma delas é utilizando o nome do vetor e o operador [] em conjunto com os registradores BX, SI ou DI. O valor em BX, SI ou DI sempre informa qual o deslocamento **em bytes** a partir do início do vetor. Caso 2 registradores sejam utilizados, o deslocamento total será a soma dos valores dos dois registradores.

Ex:

```
.DATA
   VET          DB      1,2,3,4
.CODE
  MOV AX,@DATA
  MOV DS,AX             ; inicializa o registrador DS para permitir o acesso às variáveis
  MOV BX,1
  MOV VET[BX],0         ; move o valor 0 para a segunda posição do vetor (onde está o valor 2)
  MOV SI,2
  MOV AL,VET[SI]        ; move o valor da terceira posição do vetor (valor 3) para AL
  MOV VET[BX][SI],5     ; move o valor 5 para a quarta posição do vetor (onde está o valor 4)
```

O exemplo a seguir mostra um programa que lê caracteres do teclado até que o usuário digite a tecla ENTER ou digite no máximo 15 caracteres, armazenando os caracteres lidos no vetor NOME.

Ex:

```
TITLE LeNome
.MODEL SMALL
.STACK 100h
.DATA
  ; declara um vetor NOME de 15 posições contendo o caracter "$" em cada posição
  NOME          DB      15 DUP("$")
.CODE
 ; Inicializa o registrador DS para acesso às variáveis em memória
 MOV AX,@DATA
 MOV DS,AX
 ; Inicializa o contador CX com o valor 15 (quantidade máxima de caracteres)
 MOV CX,15
 ; Inicializa o registrador BX com o valor 0 (posição onde o caracter será armazenado no vetor)
 MOV BX,0
 INICIO:
   ; Lê uma letra do teclado (caracter lido é armazenado em AL)
   MOV AH,1
   INT 21h
   ; Caso o caracter digitado seja o caracter ENTER (13), finaliza a leitura
   CMP AL,13
   JE FIM
   ; Armazena o caracter digitado no vetor NOME na posição indicada por BX
   MOV NOME[BX],AL
   ; Incrementa BX avançando para a próxima posição do VETOR
   INC BX
; Retorna ao INICIO caso o número de caracteres seja menor do que 15
LOOP INICIO
FIM:
; Finaliza o programa (retornando para o MSDOS)
MOV AH,4Ch
INT 21h
END
```

Laboratório de
Organização de Computadores

Atividade 7

Atividade para entrega

Crie um programa em linguagem assembly chamado **ATIV7.ASM** que lê uma frase do teclado e exibe na linha seguinte a frase invertida.

Exemplo:

C:\> ATIV7.EXE

Digite uma frase: **Organizacao de Computadores**

Frase invertida: **serodatupmoC ed oacazinagrO**

ENTREGA

Cada aluno deve:

1) Criar uma pasta em seu escaninho no AVA com o nome **Atividade7.**
2) Postar o arquivo **ATIV7.ASM** dentro da pasta **Atividade7**.