ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

MACROS

- **MACRO: é um bloco de texto que recebe um nome especial consiste de instruções, diretivas, comentários ou referências à outras macros.**

  A macro é chamada no momento da montagem e é expandida. O montador copia o bloco texto na posição de cada chamada de macro as expansões podem ser vistas no arquivo .LST

  - **Uso:**
    - **criar novas instruções**
    - **operacionalizar tarefas freqüentes e repetitivas**

Vantagens x desvantagens		
	Macros	**Subrotinas**
Tempo para montar	**maior**	**menor**
Quantidade de código de máquina (.EXE)	**maior**	**menor**
Tempo de execução	**menor**	**maior**
Pequenas tarefas	**x**	
Grandes tarefas		**x**

ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

Observação:

- **Para incluir um arquivo no arquivo do programa que será montado, use a diretiva: INCLUDE caminho\nome_do_arquivo_texto**

EXEMPLO

Crie uma macro para salvar e restaurar registradores na pilha.

- **Exemplo de chamada:**

```
.code
    ...
    ;definição das macros em um arquivo (lib.txt)
    INCLUDE         LIB.TXT
    salva_regs AX,BX,CX
    ...
    restaura_regs CX,BX,AX
    .....
```

ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

O arquivo LIB.TXT contem:

```
SALVA_REGS    MACRO R1,R2,R3
              PUSH  R1
              PUSH  R2
              PUSH  R3
              ENDM


RESTAURA_REGS      MACRO S1,S2,S3
              POP         S1
              POP         S2
              POP         S3
              ENDM
```


ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

Programas .COM:

- **estrutura simples**
- **pilha, dados e código ficam localizados todos no mesmo segmento**
- **ocupam menos espaço do que os equivalentes .EXE**
- **possuem versatilidade (?) e tamanho limitados -> 1 segmento = 64 kbytes**

Estrutura de escrita de um programa .COM:

- **diretiva ORG posição**
  - **define a localização do *location counter*, e a partir dele, inicia-se o programa**
- **só existe um único segmento de código (.CODE)**
- **dados -> ficam definidos logo no início e o programa deve saltar esta região**
- **pilha -> o SP aponta para o último word do segmento (FFFEh)**
- **a pilha cresce em direção ao início**

ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

Exemplo .COM
Versão .EXE

```
TITLE TESTE_1: BOM DIA!
.MODEL SMALL
.STACK   100h.
DATAMENSAGEM  DB  'BOM DIA!$'
.CODE
MAIN  PROC
MOV  AX,@DATA
MOV  DS,AX
MOV  AH,9
LEA    DX,MENSAGEM
INT    21h
MOV  AH,4Ch
INT    21h
MAIN ENDP
END   MAIN
```


ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

Exemplo .COM

```
TITLE TESTE_2: BOM DIA!
.MODEL SMALL
.CODE
ORG 0100h
START:
JMP    MAIN
MENSAGEM  DB  'BOM DIA!$'
MAIN  PROC
MOV  AH,9
LEA    DX,MENSAGEM
INT    21h
MOV  AH,4Ch
INT    21h
MAIN ENDP
END   START
```

Procedimento para geração do .COM no TASM:

```
C:\> TASM TESTE_2.ASM
C:\> TLINK TESTE_2.OBJ /t
```

ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

DEFINIÇÃO COMPLETA DE SEGMENTOS

- **Definição simplificada: uso da diretivas .MODEL, .STACK, .DATA e .CODE**
- **A diretiva SEGMENT**

  name* SEGMENT *align combine 'class'

  ;diretivas, dados e instruções

  ***name* ENDS**

Tipos de alinhamento	Descrição
PARA	O segmento inicia no próximo parágrafo* disponível
BYTE	O segmento inicia no próximo byte disponível
WORD	O segmento inicia no próximo word disponível
PAGE	O segmento inicia na próxima página** disponível

Observação

(*) Parágrafo: dígito menos significante do endereço físico em hexa valendo 0.

(**) Página: 2 dígitos menos significante do endereço físico em hexa valendo 00.

ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

DEFINIÇÃO COMPLETA DE SEGMENTOS

Tipos de combinação	Descrição
PUBLIC	Segmentos com o mesmo nome são concatenados para formar um único bloco contínuo de memória
COMMON	Segmentos com mesmo nome são sobrepostos na memória
STACK	Mesmo efeito de PUBLIC, exceto que o offset de todos os dados nele contidos estão referidos ao SS; SP inicia apontando para o final do segmento
AT *paragraph*	Indica que o segmento deve iniciar-se num parágrafo especificado

ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

Tipo de *'class'* #	Descrição
'CODE'	**Especifica que o segmento é de código**
'DATA'	**Especifica que o segmento é de dados**

Observação

 # Dependendo da seqüência de declarações dos segmentos, o tipo *class* permite definir a ordem dos segmentos na memória.

ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

EXEMPLO

```
TITLE TESTE_3: Programa de conversão para maiúscula - MÓDULO1
EXTRN     CONVERTE:NEAR                  ;informa que necessita do módulo
;CONVERTE, alocado no mesmo
;segmento
S_SEG     SEGMENT    STACK
          DB         100 DUP (0)         ;pilha com 50 words
S_SEG     ENDS
D_SEG     SEGMENT    BYTE    PUBLIC      'DATA'
MSG       DB         'Entre com uma letra minúscula: $'
D_SEG     ENDS
C_SEG     SEGMENT    BYTE    PUBLIC      'CODE'
          ASSUME     CS:C_SEG,DS:D_SEG,SS:S_SEG
MAIN      PROC
          MOV        AX,D_SEG
          MOV        DS,AX               ;inicializa DS
;
INCLUDE   A:\MY_LIB.TXT
          MOV        AH,9
          LEA        DX,MSG
          INT        21h                 ;exibe a primeira mensagem
          MOV        AH,01h
          INT        21h                 ;lê um caracter do teclado
          MOV        BL,AL               ;foi preciso salvar AL em BL !!!!!
          NOVA_LINHA                     ;macro para mudança de linha
          CALL       CONVERTE            ;chama a rotina de conversão
          RETORNO_DOS                    ;macro de retorno ao DOS
MAIN      ENDP
C_SEG     ENDS
          END        MAIN
```

ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

```
TITLE TESTE_4: Programa de conversão para maiúscula – MÓDULO2
PUBLIC    CONVERTE                          ;informa que pode
     ser usada em
;módulo diferente
D_SEG     SEGMENT BYTE      PUBLIC    'DATA'
MSG                 DB          'Convertendo para maiúscula:   '
LETRA     DB        ?,'$'
D_SEG     ENDS
C_SEG     SEGMENT BYTE      PUBLIC    'CODE'
          ASSUME    CS:C_SEG,DS:D_SEG
CONVERTE            PROC      NEAR
;converte um caracter existente em AL para maiúscula
;entrada:  AL = caracter
;saída:    nenhuma, apenas a exibição na tela do monitor
          PUSH      DX        ;salva DX pois é usado pela função 09h
          AND       BL,0DFh   ;máscara que converte para maiúscula
          MOV       LETRA,BL  ;coloca na posição para exibição
          MOV       AH,09h    ;função de exibição de string
          LEA       DX,MSG    ;mensagem de modificação
          INT       21h       ;exibe no monitor
          POP       DX
          RET                 ;retorno à rotina principal
CONVERTE            ENDP
C_SEG     ENDS
          END                 ;forma padrão de terminar neste caso
```


ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

Como gerar o .EXE

- **TASM TESTE_3.ASM (monta-se cada módulo em separado)**
- **TASM TESTE_4.ASM**
- **TLINK TESTE_3 + TESTE_4 (liga-se o conjunto)**