MACROS e PROGRAMA .COM

✍ 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** e 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

✍ **Rótulos locais** (*local labels*):

diretiva **LOCAL** lista_de _labels

informa ao montador que os labels internos são locais

✍ **Biblioteca de macros** (*macro library*):

diretiva **INCLUDE** caminho\nome_do_arquivo_texto

informa ao montador onde se encontra uma coleção de macros pré-definidas

✍ **EXEMPLO**

Crie uma macro para realizar a cópia de *strings*

```
COPY        MACRO       FONTE,DESTINO,QUANTIDADE
                LOCAL   REPETE
                SALVA_REGS      CX,SI,DI        ;existente na biblioteca
                LEA             SI,FONTE
                LEA             DI,DESTINO
                CLD
                MOV             CX,QUANTIDADE
REPETE:
                MOVSB
                LOOP    REPETE
                RESTAURA_REGS   DI,SI,CX        ;existente na biblioteca
                ENDM
```

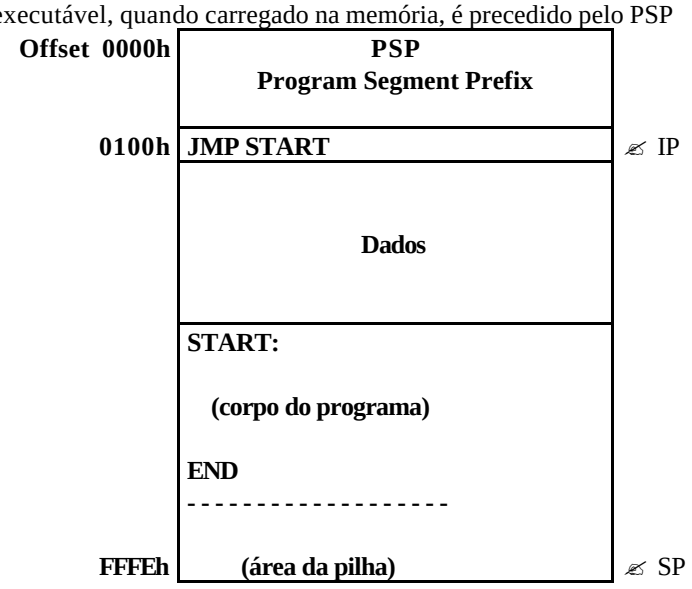
Exemplo de chamada:

```
                ...
                ;definição da macro COPY
                ;definição de outras macros
                INCLUDE         A:\MY_LIB.TXT
                ; depois da subrotina que lê o nome de um arquivo
                COPY    NOME_1,NOME_2,BX
                ...
                ;determine o que resulta da expansão desta macro.
```

No arquivo **MY_LIB.TXT**, parte do texto 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
```

- **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
- **Estrutura na memória**:
  - todo arquivo executável, quando carregado na memória, é precedido pelo PSP



- **Exemplo**: criando uma versão .COM

Versão .EXE

```
TITLE TESTE_1: BOM DIA!
.MODEL SMALL
.STACK  100h
.DATA
MENSAGEM  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
```

Versão .COM

```
TITLE TESTE_2: BOM DIA!
.MODEL SMALL
.CODE
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 montagem:

C:\> TASM TESTE_2.ASM
C:\> TLINK TESTE_2.OBJ
C:\> EXE2BIN TESTE_2.EXE TESTE_2.COM

Execute ambos os arquivos .EXE e .COM e compare os tamanhos de ambos e verifique que o .COM é bem menor.
Todo arquivo .EXE contem adicionalmente um bloco de cabeçalho de **512 bytes** (*header*), que orienta o DOS durante o seu carregamento na memória e informa detalhes de tamanho dos diversos segmentos e posição das variáveis e rótulos.