MC404

ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

2006

Prof. Paulo Cesar Centoducatte
ducatte@ic.unicamp.br
www.ic.unicamp.br/~ducatte

MC404

ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

"Introdução à linguagem assembly do 8086"

Introdução à linguagem assembly do 8086
Sumário

Introdução à linguagem assembly do 8086

A sintaxe assembly do 8086

- A linguagem montadora não é sensível à letra maiúscula ou minúscula
- Para facilitar a compreensão do texto do programa, sugere-se:
  - uso de letra maiúscula para código;
  - uso de letra minúscula para comentários.

Introdução à linguagem assembly do 8086

Declarações (*statements*):

- Instruções, que são convertidas em código de máquina
- Diretivas, que instruem o montador a realizar alguma tarefa específica:
  - Alocar espaço de memória para variáveis;
  - Criar uma sub-rotina (*procedure* ou procedimento).

Introdução à linguagem assembly do 8086

- Formato de uma declaração (linha de programa):

[Nome:] [Cod. oper.] [Operando(s)] [;Comentário]

Exemplo:

```
INICIO:  MOV CX,5h    ;inicializar contador
```

Observação:

A separação entre os campos deve ser do tipo <espaço> ou <tab>.

Introdução à linguagem assembly do 8086

Campo Nome:

- Pode ser um rótulo de instrução, um nome de sub-rotina, um nome de variável, contendo de 1 a 31 caracteres, iniciando por uma letra ou um caracter especial e contendo somente letras, números e os caracteres especiais ? . @ _ : %

Exemplos:

nomes válidos	nomes inválidos
LOOP1:	DOIS BITS
.TEST	2abc
@caracter	&A2.25
SOMA_TOTAL4	#33

Observação:

O Montador traduz os nomes por endereços de memória ou valores constantes.

Introdução à linguagem assembly do 8086

Campo de Código de Operação:

- Contem o código de operação simbólico (mnemônico)
- No caso de diretivas, contem o código de pseudo-instrução

Exemplos:

instruções	diretivas
MOV	.MODEL
ADD	.STACK
INC	nome PROC
JMP	

Introdução à linguagem assembly do 8086

Campo de operandos:

- Instruções podem conter 0, 1 ou 2 operandos no 8086.

Exemplos:

```
NOP        ; sem operandos: instrui para fazer nada
INC  AX    ; um operando: soma 1 ao conteúdo de AX
ADD A,2d   ; dois operandos: soma 2 ao conteúdo da
           ; palavra de memória A (variável A)
ADD A,2
```


Introdução à linguagem assembly do 8086

Campo de operandos (cont.):

- No caso de instruções de dois operandos:
  - o primeiro, operando destino: registrador ou posição de memória onde o resultado será armazenado, o conteúdo inicial será modificado;
  - o segundo, operando fonte: não modificado pela instrução;
  - os operandos são separados por uma vírgula.
- No caso de diretivas, o campo de operandos contem mais informações acerca da diretiva.

Introdução à linguagem assembly do 8086

Campo de Comentário:

- Um ponto-e-vírgula (;) marca o início deste campo;
- O Montador ignora tudo após este marcador até o fim da linha;
- Comentários são opcionais, mas imprescindíveis.

- *OBS.: Uma boa prática de programação é comentar tudo e incluir a informação acerca da idéia por trás da codificação (o algorítmo).*

Introdução à linguagem assembly do 8086

Campo de Comentário (cont.):

- Exemplos:

```
MOV CX,0    ;movimenta 0 para CX (óbvio! -
                                  evitar)
MOV CX,0    ;CX conta no. de caracteres,
            ;inicialmente vale 0
;          (linhas em branco: separação)
;
; inicialização dos registradores (linha inteira de comentário)
```

Introdução à linguagem assembly do 8086

Formato de dados, variáveis e constantes

- Números:

  Exemplos:

  - binário: 1110101b ou 1110101B
  - decimal: 64223 ou 64223d ou 64223D, 1110101 é considerado decimal (ausência do B),
  - hexa: 64223h ou 64223H, 0FFFFh (começa com um decimal e termina com h), 1B4Dh

Introdução à linguagem assembly do 8086

- Exemplos de números ilegais:
  - 1,234 caracter estranho (vírgula)
  - FFFFh não começa por número de 0 a 9, difícil distinguir do nome de uma variável
  - 1B4D não termina com h ou H e contém dígito não decimal

Introdução à linguagem assembly do 8086

- Caracteres ASCII e strings:
  - Caracteres isolados ou *strings* de caracteres devem estar escritos dentro de aspas simples (') ou duplas (").

Exemplos:

" A" ou ' A '
'ola, como vai'
"EXEMPLO"

Introdução à linguagem assembly do 8086

Variáveis:

- Variável é um nome simbólico para um dado atualizável pelo programa.
  - Cada variável possui um tipo e é associada a um endereço de memória;
  - Usa-se diretivas para definir o tipo da variável;
  - O Montador atribui o endereço de memória.

Introdução à linguagem assembly do 8086

DIRETIVAS	SIGINIFICADO
DB	define byte (8 bits)
DW	define word (16 bits, 2 bytes consecutivos)
DD	define doubleword (2 palavras, 4 bytes consecutivos)
DQ	define quadword (4 palavras, 8 bytes consecutivos)
DT	define ten bytes (10 bytes consecutivos)

Introdução à linguagem assembly do 8086

- Definição de variáveis de tipo byte:

Nome DB valor_ inicial

Exemplos:

```
Alfa    DB   0h     ; equivale a 00h
A       DB   10h
B       DB   0150h  ; ilegal, por que?
BIT     DB   ?      ; não inicializada
```

Introdução à linguagem assembly do 8086

- Definição de variáveis de tipo word:

Nome DW valor_inicial

Exemplos:

```
WORD1  DW  0h      ; equivale a 0000h
CONTA  DW  0150h   ; OK!, por que?
C      DW  ?       ; não inicializada
WORD1  DW  1234h   ; byte baixo 34h, endereço  WORD1
                   ; byte alto 12h endereço WORD1+1
```

WORD1+1	12h
WORD	34h

Introdução à linguagem assembly do 8086

- *Array*: sequência de bytes ou words consecutivos na memória
  - Armazenar dados relacionados;
  - Armazenar caracteres ASCII organizados (ex: texto).

Exemplos:

```
BYTE_ARRAY      DB  10h,20h,30h
WORD_ARRAY      DW 1000h,123h,0h,0FFFFh
```

- Um *array* pode conter um *string* de caracteres, sendo definido como:

```
LETRAS  DB 'abC' ; e´ equivalente aos caracteres ASCII
LETRAS  DB  61h,62h,43h ;depende se maiúscula ou
                        ; minúscula
```


Introdução à linguagem assembly do 8086

- Combinação de caracteres e números numa mesma definição:

MENSAGEM DB 'Alo!', 0Ah,0Dh,'$'

OBS.: Para alguns serviços da BIOS o caracter '$' marca o fim de uma *string* de caracteres (e não é exibido).

Introdução à linguagem assembly do 8086

- Constantes: é um nome simbólico para um dado de valor constante, que seja muito utilizado num programa.
  - Para atribuir um nome a uma constante, utiliza-se a pseudo-instrução EQU (*equates* -> igual a) e a sintaxe:

Nome EQU valor_da_constante

Exemplos:

```
LF      EQU  0Ah   ;caracter Line Feed como LF
CR      EQU  0Dh   ;caracter Carriage return como CR
LINHA1 EQU  'Digite seu nome completo'
MENSAGEM   DB LINHA1,LF,CR
```

Observação: Constantes não geram código de máquina e nem "ocupam" espaço de memória em tempo de execução.

Introdução à linguagem assembly do 8086

- Algumas instruções básicas do 8086
  - MOV destino, fonte
    - Usada para transferir dados entre:
      - registrador e registrador
      - registrador e uma posição de memória
      - mover um número diretamente para um registrador ou posição de memória

Introdução à linguagem assembly do 8086

- Instrução MOV

Operando fonte	Operando destino		
	Registrador dados	Registrador segmento	Posição memória
Registrador Dados	sim	sim	sim
Registrador Segmento	sim	não	sim
Posição memória	sim	sim	não
Constante	sim	não	sim

Introdução à linguagem assembly do 8086

- Exemplos de instruções MOV válidas:

```
MOV AX,WORD1      ;movimenta o conteúdo da posição de
                  ;memória WORD1 para o registrador AX

MOV AH,'A'   ;transfere o caracter ASCII 'A' para AH

MOV AH,41h  ;idem anterior: 41h corresponde ao caracter A

MOV AH,BL    ;move o conteúdo do byte baixo de BX
             ;o byte alto de AX

MOV AX,CS    ;transfere cópia do conteúdo de CS para AX
```


Introdução à linguagem assembly do 8086

- MOV AX,WORD1

Antes	Depois
AX	AX
0006h	8FFFh
WORD1	WORD1
8FFFh	8FFFh

- Obs: para a instrução MOV não é permitido operar de posição de memória para posição de memória diretamente, por motivos técnicos do 8086.

Introdução à linguagem assembly do 8086

- Por exemplo:

```
MOV  WORD1,WORD2 ;instrução inválida. Esta restrição é
                 ;contornada como segue
;
MOV  AX,WORD2  ;primeiro o conteúdo de WORD2 vai para AX

MOV  WORD1,AX  ;depois, o conteúdo de AX é movido para a
               ;posição de memória WORD1
```


Introdução à linguagem assembly do 8086

XCHG destino, fonte

- Usada para troca de dados (nos dois sentidos) entre:
  - registrador e registrador
  - registrador e uma posição de memória
  - não é permitido trocas diretas entre duas posições de memória

Introdução à linguagem assembly do 8086

- XCHG destino, fonte

Operando fonte	Operando destino	
	Registrador dados	Posição memória
Registrador Dados	sim	sim
Registrador Segmento	não	não
Posição memória	sim	não

Introdução à linguagem assembly do 8086

- Exemplos de instruções válidas:

```
XCHG AX, WORD1    ;troca o conteúdo da posição de memória
                  ; WORD1 com o do registrador AX
XCHG AH, BL       ;troca o conteúdo do byte baixo de BX
                  ;com o do byte alto de AX

XCHG  AX,BX
```

Antes	Depois
AX	BX
0006h	FFFFh
AX	BX
FFFFh	0006h

Introdução à linguagem assembly do 8086

- ADD destino,fonte
- SUB destino,fonte
  - Usadas para adicionar (ou subtrair) dados entre:
    - registrador e registrador
    - registrador e uma posição de memória
    - adicionar (ou subtrair) um número diretamente a (de) um registrador ou posição de memória

Operando fonte	Operando destino	
	Registrador dados	Posição memória
Registrador Dados	sim	sim
Posição memória	sim	não
Constante	sim	sim

Introdução à linguagem assembly do 8086

- Exemplos de instruções válidas:

```
ADD AX,BX   ;soma o conteúdo de BX com AX, resultado em AX

ADD AX,WORD1  ;soma o conteúdo da posição de memória
              ;WORD1 a AX e resultado em AX

SUB WORD2,AX  ;subtrai o conteúdo de AX do conteúdo da
              ;posição de memória WORD2, resultado em
              ;WORD2

SUB BL,5      ;subtrai a quantidade 5 decimal do conteúdo
              ; de BL
```


Introdução à linguagem assembly do 8086

Observações:

```
ADD  BYTE1,BYTE2  ;instrução inválida esta restrição
                  ; é contornada como segue

MOV  AL,BYTE2  ;primeiro o conteúdo de BYTE2 vai para AL
ADD  BYTE1,AL  ;depois, o conteúdo de AL é somado ao da
               ; posição de memória BYTE1, resultado final
               ; em BYTE1
```

O resultado de SUB, se for negativo, estará armazenado no registrador destino em complemento de 2.

Introdução à linguagem assembly do 8086

- INC destino
- DEC destino
  - Usadas para adicionar 1 (incrementar) ou subtrair 1 (decrementar) ao/do conteúdo de:
    - um registrador;
    - uma posição de memória.

Exemplos:

```
INC CX      ;incrementa o conteúdo de CX
INC WORD1   ;incrementa conteúdo  posição  memória WORD1
DEC BYTE2   ;decrementa conteúdo posição de memória BYTE2
DEC CL      ;decrementa o conteúdo de CL (byte baixo de CX)
```


Introdução à linguagem assembly do 8086

- NEG destino
  - Usada para substituir o conteúdo *destino* pelo seu complemento de 2, operando sobre:
    - um registrador;
    - uma posição de memória.

Exemplos:

```
NEG BX      ; gera o complemento de 2
            ; do conteúdo de BX
NEG WORD1   ; idem, no conteúdo da posição de
            ; memória WORD1
```


Introdução à linguagem assembly do 8086

- Tradução de expressões matemáticas em Linguagem de Alto Nível para Linguagem Montadora

Exemplo1: B = A

```
MOV  AX,A   ; transfere o conteúdo da posição de
            ; memória A para AX  e
MOV  B,AX   ; transfere AX para a posição de
            ; memória B
```

Introdução à linguagem assembly do 8086

Exemplo 2: A = 5 - A

```
NEG A     ; gera o complemento de 2 da posição
          ; de memória A e
ADD A,5   ; realiza  (-A) + 5, que equivale a  5 - A
```

Exemplo 3: A = B - 2A

```
MOV AX,B      ; AX contem a variável B
SUB AX,A      ; AX contem B - A
SUB AX,A      ; AX contem B - 2A
MOV A,AX      ; movimenta o resultado para A
```


Estrutura de um programa em Linguagem Montadora

- Modelos de memória - TASM
  - O tamanho que os segmentos de código e de dados devem ter é especificado pelo modelo de memória por meio da diretiva .MODEL.
  - Sintaxe: .MODEL modelo_de_memória

Modelo	Descrição
SMALL	Código em 1 segmento; Dados em 1 segmento
MEDIUM	Código em mais de 1 segmento; Dados em 1 segmento
COMPACT	Código em 1 segmento; Dados em mais de 1 segmento
LARGE	Código em mais de 1 segmento; Dados em mais de 1 segmento; Nenhum array maior que 64 Kbytes
HUGE	Código em mais de 1 segmento; Dados em mais de 1 segmento; Arrays maiores que 64 Kbytes

Estrutura de um programa em Linguagem Montadora

- Segmento de dados
  - Contem a definição e declaração das variáveis.
  - Pode-se também fazer a atribuição de símbolos para constantes.

Sintaxe: .DATA (segment data)

Exemplo:

```
.DATA
WORD1       DW   A8h
BYTE1       DB   5
MENSAGEM    DB   'Isto e uma mensagem'
LF          EQU  0Ah
```


Estrutura de um programa em Linguagem Montadora

- Segmento de pilha (*stack segment*)
  - Reserva um bloco de posições de memória consecutivas para armazenar a pilha.
  - Deve ter espaço suficiente para suportar a pilha no seu máximo tamanho.

Sintaxe: .STACK tamanho (segment stack)

Exemplo:

```
.STACK 100h   ; reserva 100h bytes para a área
              ; de pilha, um tamanho razoável
              ; para a maioria das aplicações não
              ; recursivas
```


Estrutura de um programa em Linguagem Montadora

- Segmento de código
  - Contem propriamente as instruções do programa.
  - Dentro do segmento de código, as instruções são organizadas em procedimentos ou sub-rotinas.

Sintaxe: .CODE (segment code)

Estrutura de um programa em Linguagem Montadora

Exemplo:

```
.CODE
nome        PROC
;
;corpo da procedure -> instruções
;
nome ENDP
;
;outras procedures seguem abaixo, se existirem
```

onde:

nome -> identificação da *procedure*
PROC e ENDP -> pseudo-instruções usadas para delimitar a *procedure*
para um programa simples, não há necessidade de se definir a *procedure*.

Estrutura de um programa em Linguagem Montadora

Exemplo de uma estrutura de programa assembly completa

```
TITLE nome_do_programa
    .MODEL    SMALL
    .STACK    100h
    .DATA
    ;
    ;definição dos dados: variáveis e constantes
    ;
    .CODE
    EXEMPLO PROC
    ;
    ;seqüência de instruções
    ;
    EXEMPLO ENDP
    ;
    ;segue outras procedures
    ;
    END EXEMPLO
```

Obs:
se não houver definição de *procedure*, usa-se apenas END.

Introdução à linguagem assembly do 8086

Instruções de entrada e saída
IN e OUT -> instruções *Assembly* para acessar portas de E/S para periféricos

Não são utilizadas na maioria das aplicações:
os endereços das portas de E/S variam conforme o modelo do PC é mais fácil utilizar o SO (DOS) ou o BIOS para Funções de E/S

Para acessar as rotinas de E/S do BIOS ou DOS utiliza-se a instrução:
INT número_de_interrupção

Introdução à linguagem assembly do 8086

Observação:

Em uma chamada do BIOS (ou função do DOS) o programa em curso é interrompido, passando o controle para o DOS, que realiza a operação de E/S e retorna o controle para o programa.

Exemplo:

```
INT 21h ; acessa um grande número de funções
        ; de E/S do DOS
```


Introdução à linguagem assembly do 8086

- Algumas funções DOS de E/S

Função 1h: Entrada de um caracter simples pelo teclado
Acesso: AH = 1h
Resultado: AL = código ASCII do caracter digitado no teclado

Função 2h: Exibição de caracter simples no monitor de vídeo
Acesso: AH = 2h
DL = código ASCII do caracter a exibir
Resultado: exibição na tela do monitor

Introdução à linguagem assembly do 8086

Exemplos:
Trecho padrão de programa para providenciar a entrada de um caracter ASCII pelo teclado:

```
MOV AH,1h ;prepara para entrar caracter pelo
          ; teclado o processador espera até
          ; que o usuário digite o caracter
          ; desejado
INT  21h  ; após a digitação, caracter ASCII
          ; em AL. Se um caracter não-ASCII
          ; for digitado, AL = 0h
```

Obs: o caracter teclado também aparece no monitor (eco), por causa do DOS.

Introdução à linguagem assembly do 8086

Trecho padrão de programa para providenciar a saída de um caracter ASCII para o monitor de vídeo:

```
MOV   AH,2h  ; prepara para exibir caracter no monitor
MOV   DL,'?' ; o caracter é '?'
INT   21h    ; exibe (monitor apresenta '?')
             ; após a exibição, o cursor da tela avança
             ; para a próxima posição da linha (se já for
             ; atingido o fim da linha, vai para o início da
             ; próxima linha)
```

Introdução à linguagem assembly do 8086

Obs: também se pode "exibir" caracteres ASCII de controle:

Código ASCII	Símbolo	Função
07h	BEL	*Bell* (som de bip)
08h	BS	*Back Space* (espaço para trás)
09h	HT	*Tab* (tabulação)
0Ah	LF	*Line Feed* (ir para uma nova linha)
0Dh	CR	*Carriage Return* (ir para inicio linha)

Estrutura de um programa em Linguagem Montadora

Criando e Rodando um Programa

- Especificação do programa ECO DO TECLADO NA TELA:
  - ler um caracter do teclado
  - exibir o caracter lido na próxima linha da tela do monitor
  - retornar ao SO

Criando e Rodando um Programa

- Escrevendo as partes

a) O programa estimula o usuário a interagir apresentando um '?':

```
MOV AH,2        ; funcao DOS para exibir caracter
MOV DL,'?'      ; caracter '?'
INT 21H         ; exibir
```

b) Lendo o caracter teclado pelo usuário e salvando-o em num registrador:

```
MOV AH,1        ; funcao DOS para leitura de caracter
INT 21H         ; caracter e' lido em AL
MOV BL,AL       ; salvando-o em BL
```


Criando e Rodando um Programa (cont.)

c) Movendo o cursor da tela para o início da próxima linha:

```
MOV AH,2        ; funcao DOS para exibir caracter
MOV DL,0DH      ; caracter ASCII <CR> - return
INT 21H         ; executando
MOV DL,0AH      ; caracter ASCII <LF> - line feed
INT 21H         ; executando
```

d) Recuperando o caracter lido e exibindo-o:

```
MOV DL,BL       ; recuperando o caracter salvo
INT 21H         ; exibir
```


O programa ECO completo:

```
TITLE PGM4_1:  PROGRAMA DE
   ECO
 DO TECLADO NA TELA
.MODEL  SMALL
.STACK  100H
.CODE
MAIN    PROC
;
;apresentacao do prompt '?'
      MOV AH,2   ;funcao para
   exibir caracter
      MOV DL,'?'  ;caracter '?'
      INT 21H      ;exibir
;entrada do caracter pelo teclado
      MOV AH,1    ;funcao para
   leitura de caracter
      INT 21H       ;caracter e'
   lido em AL
      MOV BL,AL   ;salvando-o em
   BL
;movendo de linha
   MOV AH,2       ;funcao para exibir caracter
   MOV DL,0DH     ;caracter  <CR> - return
   INT 21H        ;executando
   MOV DL,0AH     ;caracter <LF> - line feed
   INT 21H        ;executando exibindo na
                  ;tela o caracter lido: efeito
                  ; de ECO
   MOV DL,BL      ;recuperando caracter salvo
   INT 21H                ;exibir

;retorno ao DOS
   MOV AH,4CH   ; funcao para saida
   INT 21H      ; saindo

MAIN    ENDP
END MAIN
```

Como Obter o Programa ECO.EXE Executável.

1. Edite o program ECO utilizando um editor de texto simples, com saída em texto ASCII. Sugestão: use o EDIT do DOS. O arquivo (texto ASCII) deve ter a extensão . ASM
   C:\ > EDIT ECO.ASM <enter>

OBS.: Se usar NASM atenção para o uso das Diretivas

2. Rode o programa Montador TASM (Borland). Como resultado, aparece em seu diretório de trabalho um arquivo ECO.OBJ
   C:\ > TASM ECO. ASM <enter>

Como Obter o Programa ECO.EXE Executável.

3. Rode o programa Lincador TLINK. Como resultado, aparece em seu diretório de trabalho um arquivo ECO.EXE.
   C:\ > TLINK ECO.OBJ <enter>

4. Rode o programa ECO.EXE, respondendo ao '?' com uma letra K, por exemplo.
   C:\ > ECO.EXE <enter>
   ?K <- letra K digitada pelo usuário
   K <- eco da letra K aparece na tela
   C:\ > <- note que o controle retorna ao DOS

Exercício.: Tente com outras letras ou procure modificar o programa para obter outros efeitos com caracteres digitados no teclado.

Mais Funções DOS de E/S

Função 4Ch: Termina o processo corrente e transfere controle para o DOS
   Acesso: AH = 4Ch
   Resultado: saída para o DOS

Função 9h: Exibição de *string* de caracteres no monitor de vídeo
   Acesso: AH = 9h
   DX = offset do endereço onde começa o *string*
   Resultado: *string* exibido

Obs: o *string* de caracteres deve terminar com o caracter '$', que marca o fim da sequência e não é exibido.

Para exibição de um *string* de caracteres há dois problemas:

a) DS inicialmente não está apontando para o segmento de dados do programa recém iniciado (DS ainda aponta para algum segmento de dados do DOS);
b) deve-se colocar em DX o *offset* do endereço do *string* que queremos exibir

Como Apontar DS para o Segmento de Dados do Programa

@DATA → palavra reservada para obter o número do segmento de dados definido pela diretiva .DATA, que contem as variáveis e constantes.

Exemplo:
Para inicializar corretamente DS para o programa corrente:

```
.DATA
...
.CODE

MOV   AX,@DATA   ;coloca o número do segmento de dados em AX
MOV   DS,AX      ;pois DS não pode receber @DATA diretamente
```

Observação:
- O programa Montador traduz o nome @DATA pelo número de segmento onde se encontram os dados definidos pela diretiva .DATA.

Como Colocar em DX o *Offset* do Endereço de um *String* a Exibir

- LEA destino,fonte
  - Significa Load Effective Address -> coloca uma cópia do *offset* do endereço da posição de memória fonte no registrador destino.

Exemplo:

```
.DATA
MENSAGEM    DB      'Adoro ISB!$'
...
.CODE
LEA     DX,MENSAGEM  ;DX carregado com o offset de MENSAGEM
```

Obs: após esta operação, DX conterá o *offset* da posição de memória onde inicia o *string* MENSAGEM

Programa para Imprimir um *String* de Caracteres

```
TITLE   PROG PARA IMPRESSAO DE
   'STRING'
.MODEL  SMALL
.STACK  100H                          ; exibindo a MENSAGEM
.DATA                                 ;
MSG DB 'ALO! Como voces estao indo!$'   MOV AH,9   ; funcao DOS
.CODE                                             ; para exibir 'string'
MAIN    PROC                          INT 21H     ; exibindo
;                                     ;
; inicializando o registrador DS      ; retorno ao DOS
;                                     ;
  MOV AX,@DATA                          MOV AH,4CH  ; funcao DOS para
  MOV DS,AX ; segmento dados                        ; saida
            ; inicializado             INT 21H     ; saindo
;                                     MAIN   ENDP
; obtendo offset posição  memória de Msg  END MAIN

    LEA DX,MSG  ;offset endereço vai
                ; para DX
```

Reescreva o programa usando NASM

Exercício

- Programa de conversão de letra minúscula para maiúscula.
  - Especificação do programa:
    - apresente ao usuário uma mensagem do tipo:
      Entre com uma letra minuscula:
    - ler um caracter do teclado (não é necessário testar se é letra)
    - apresente uma segunda mensagem do tipo:
      Em maiuscula ela fica:
    - apresente em seguida a letra convertida
    - retornar ao SO
- **OBS.: Repita o programa testando a validade do caracter digitado**