LABORATÓRIO DE LINGUAGEM DE MONTAGEM

INSTRUÇÕES QUE MANIPULAM ***STRINGS***

- **STRING -> *ARRAY* de bytes ou words definidos no .DATA**

- **RESUMO DAS INSTRUÇÕES**

INSTRUÇÃO	Destino	Fonte	Forma com bytes	Forma com words
Mover string	**ES:DI**	**DS:SI**	**MOVSB**	**MOVSW**
Comparar 2 strings	**ES:DI**	**DS:SI**	**CMPSB**	**CMPSW**
Armazenar string (store)	**ES:DI**	**AL ou AX**	**STOSB**	**STOSW**
Carregar string (load)	**AL ou AX**	**DS:SI**	**LODSB**	**LODSW**
Varrer string (scan)	**ES:DI**	**AL ou AX**	**SCASB**	**SCASW**

OS OPERANDOS SÃO **IMPLÍCITOS**: NÃO FAZEM PARTE DAS INSTRUÇÕES

FORMA ALTERNATIVA	EQUIVALENTE	CONDIÇÃO
MOVS STRING1, STRING2	MOVSB ou MOVSW	DI -> STRING1 e SI -> STRING 2
CMPS STRING3, STRING4	CMPSB ou CMPSW	DI -> STRING3 e SI -> STRING 4
STOS STRING5	STOSB ou STOSW	DI -> STRING5
LODS STRING6	LODSB ou LODSW	SI -> STRING6
SCAS STRING7	SCASB ou SCASW	DI -> STRING7

- **Os STRINGS são automaticamente percorridos segundo uma DIREÇÃO DEFINIDA**

- **FLAG DE DIREÇÃO: *Direction Flag* (DF)**
  - **Se DF = 0 , os registradores indexadoes SI e DI são incrementados**
    - **percorre o STRING da esquerda para a direita**
  - **Se DF = 1, SI e DI são decrementados**
    - **percorre o STRING da direita para a esquerda**

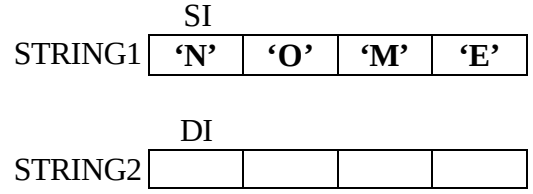
- **Instuções para limpar e setar DF, definindo a direção**
  - **CLD (clear direction flag) -> faz DF = 0**
  - **STD (set direction flag) -> faz DF = 1**

Instruções de movimentação: MOVSB e MOVSW

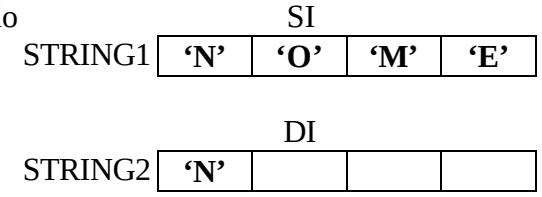
Exemplo:

```
...
.DATA
STRING1     DB     'NOME'
STRING2     DB     4 DUP (?)
...
.CODE
MOV   AX, @DATA
MOV   DS, AX              ;inicializa o DS no segmento de dados
MOV   ES, AX              ;inicializa ES no mesmo segmento
LEA   SI, STRING1         ;SI aponta para o início de STRING1
LEA   DI, STRING2         ;DI aponta para o início de STRING2
CLD                       ;limpa DF (direção crescente: esquerda para direita)
MOV   CX, 4               ;inicializa CX com o número de caracteres de STRING1
REP MOVSB                 ;repete MOVSB até que CX = 0
...
```

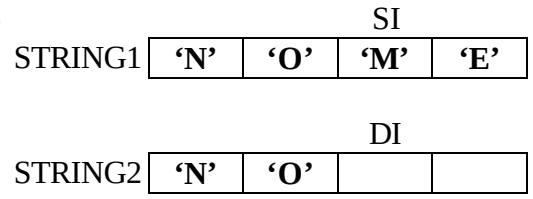
Antes de iniciar



Após a primeira execução



Após a segunda execução



MOVSW

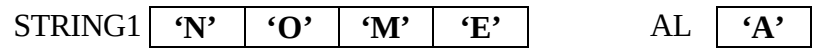
- Opera words (formato DW)
- SI e DI são incrementados (ou decrementados) de dois em dois

Instruções de armazenamento: STOSB e STOSW

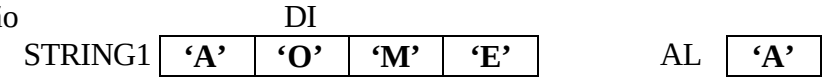
Exemplo:

```
...
.DATA
STRING1      DB     'NOME'
...
.CODE
MOV   AX, @DATA
MOV   ES, AX              ;inicializa ES no segmento de dados – VAMOS USAR DI
LEA   DI, STRING1         ;DI aponta para o inicio de STRING1 – operando destino
CLD                       ;limpa DF (direção crescente: esquerda para direita)
MOV   AL, 'A'             ;AL é operando fonte e contem o caracter 'A'
STOSB                     ;armazena o primeiro 'A'
STOSB                     ;armazena o segundo 'A'
...
```

Antes de iniciar



Após a primeira execução



Após a segunda execução

STOSW

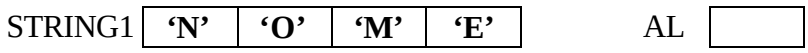
- Opera words (formato DW)
- DI é incrementado (ou decrementado) de dois em dois
- Operando fonte se encontra em AX

Instruções de carregamento: LODSB e LODSW

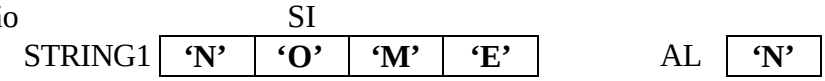
Exemplo:

```
...
.DATA
STRING1      DB     'NOME'
...
.CODE
MOV   AX, @DATA
MOV   DS, AX              ;inicializa o DS no segmento de dados
LEA   SI, STRING1         ;SI aponta para o inicio de STRING1 – operando fonte
CLD                       ;limpa DF (direção crescente: esquerda para direita)
; operando destino é AL
LODSB                     ;carrega o primeiro caracter de STRING1
LODSB                     ;carrega o segundo caracter de STRING1
...
```

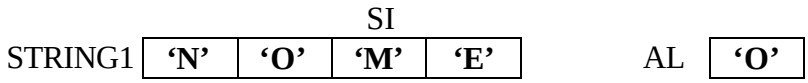
Antes de iniciar



Após a primeira execução



Após a segunda execução



STOSW

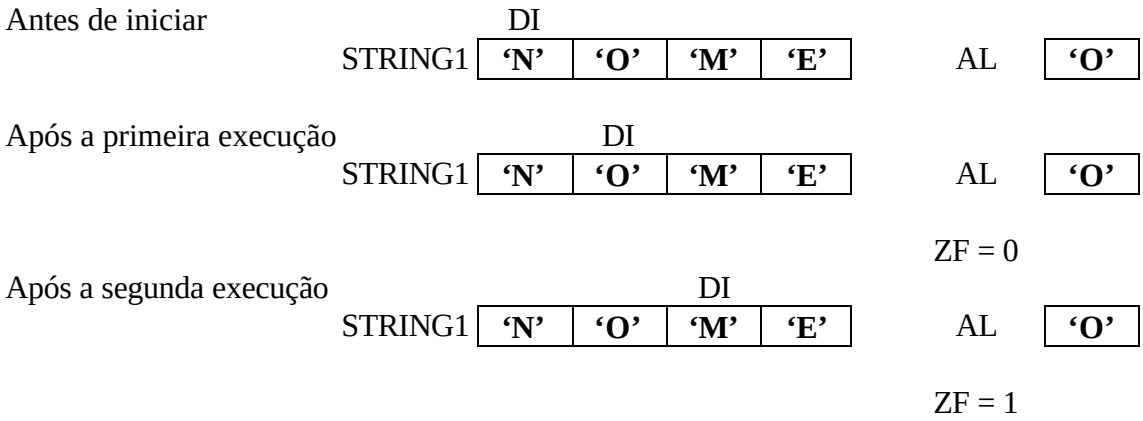
- Opera words (formato DW)
- SI é incrementado (ou decrementado) de dois em dois
- Operando destino se encontra em AX

Instruções de varredura (*Scan*): SCASB e SCASW

- **Usadas para examinar o conteúdo de um string contra um determinado alvo**
  - **Para SCASB -> Byte alvo deve estar contido em AL**
- **Realiza uma operação de subtração de conteúdos**
  - **AL – STRING[DI]**
- **Flags ZF, SF, OF,CF são afetados após cada execução**

Exemplo:

```
...
.DATA
STRING1      DB     'NOME'
...
.CODE
MOV   AX, @DATA
MOV   ES, AX              ;inicializa o ES no segmento de dados
LEA   DI, STRING1         ;DI aponta para o inicio de STRING1
MOV   AL, 'O'
CLD                       ;limpa DF (direção crescente: esquerda para direita)
; alvo está em AL
REPNE  SCASB              ;repete SCASB enquanto o conteúdo do STRING1 não for
                          ;igual ao conteúdo de AL
...
```



SCASW

- Opera words (formato DW)
- DI é incrementado (ou decrementado) de dois em dois
- Operando alvo se encontra em AX
- Realiza a operação de subtração **AX – STRING[DI]**, alterando os Flags

Instruções de comparação: CMPSB e CMPSW

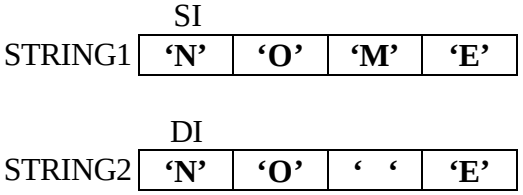
CMPSB

- **realiza uma operação de subtração de conteúdos; STRING[SI] – STRING[DI]**
- **Flags ZF, SF, OF,CF são afetados após cada execução**

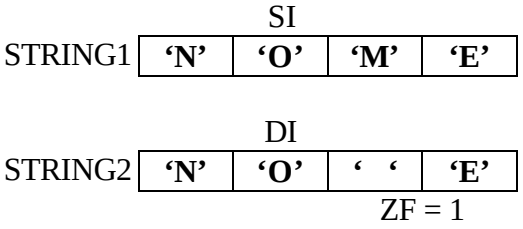
Exemplo:

```
...
.DATA
STRING1     DB     'NOME'
STRING2     DB     'NO   E'
...
.CODE
MOV   AX, @DATA
MOV   DS, AX                ;inicializa o DS no segmento de dados
MOV   ES, AX                ;inicializa ES no mesmo segmento
LEA   SI, STRING1           ;SI aponta para o inicio de STRING1
LEA   DI, STRING2           ;DI aponta para o inicio de STRING2
CLD                         ;limpa DF (direção crescente: esquerda para direita)
REPE CMPSB          ;repete CMPSB enquanto os conteúdos dois dois STRINGs
                            ;sejam iguais
...
```

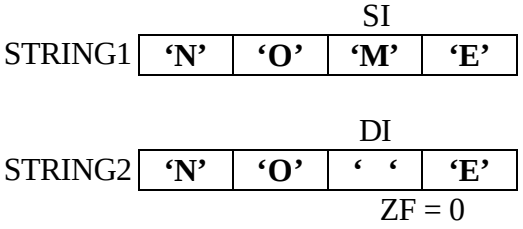
Antes de iniciar



Após a primeira execução



Após a segunda execução



CMPSW

- Opera words (formato DW)
- SI e DI são incrementados (ou decrementados) de dois em dois
- Realiza a operação de subtração **STRING[SI] – STRING[DI]**, alterando os Flags

ALGUMAS SUBROTINAS INTERESSANTES

1. Subrotina para leitura de STRING pelo teclado

```
LER_STRING PROC  NEAR
;Lê e armazena um STRING a partir do teclado até que CR seja digitado
;permite correção de caracer através do BACKSPACE
;entrada:      DI contem offset do STRING
;saída:        DI contem offset do STRING
;              BX contem o número decaracteres lidos
               PUSH  AX
               PUSH  DI
               CLD                ;STRING será lido da esquerda para direita
               XOR   BX,BX        ;inicia BX com zero
               MOV   AH,1
               INT   21H          ;leitura de caracter em AL
RETORNO:
               CMP   AL,0DH       ;caracter é CR?
               JE    FIM1
               CMP   AL,8H        ;caracter é BACKSPACE?
               JNE   ARMAZENA     ;não é, então armazenar
               DEC   DI           ;se é, remover
               DEC   BX           ;diminuir a contagem
               JUMP  LER_NOVO     ;ir ler novo caracter
ARMAZENA:
               STOSB              ;armazena o caracter no STRING
               INC   BX           ;incrementa contador de caracteres
LER_NOVO:
               INT   21H          ;leitura de caracter em AL
               JUMP  RETORNO      ;continuar o loop
FIM1:
               POP   DI
               POP   AX
               RET
LER_STRING ENDP
```

Obs: Ver Cap. 11, página 210, do livro texto.

ALGUMAS SUBROTINAS INTERESSANTES - continuação

2. Subrotina para exibição de STRING no monitor (DISPLAY)

```
MOSTRA_STRING   PROC  NEAR
;Exibe um STRING na tela do monitor de video
;entrada:       SI contem offset do STRING
;               BX contem o número de caracteres a serem exibidos
;saída:         nenhuma variável
                PUSH  AX
                PUSH  BX
                PUSH  CX
                PUSH  DX
                PUSH  SI
                MOV   CX,BX       ;número de caracteres no reg. Contador
                JCXZ  FIM2        ;termina se não há caracteres
                CLD               ;STRING será lido da esquerda para direita
                MOV   AH,2        ;prepara para exibir
                INT   21H         ;leitura de caracter em AL
TOPO:
                LODSB             ;caracter do STRING vem para AL
                MOV   DL,AL       ;colocar em DL
                INT   21H         ;exibe
                LOOP  TOPO        ;entra em loop até que CX seja zero
FIM2:
                POP   SI
                POP   DX
                POP   CX
                POP   BX
                POP   AX
                RET
MOSTRA_STRING   ENDP
```

Obs: Ver Cap. 11, página 212, do livro texto.

Interrupção INT 21H – Função 0AH

Usada para ler automaticamente um STRING de caracteres ASCII digitado a partir do teclado.

A primeiro byte do array deve ser inicializado pelo programa com o número de caracteres esperados.

Após a execução do INT 21H, o segundo byte conterá o número real de caracteres digitados.

A introdução de caracteres pode ser interrompida através do CR, que é armazenado mas não contado. Por isso, deve-se dimensionar o STRING para N+1 bytes

Se o usuário exceder a quantidade máxima, o computador produzirá um BIP

Exemplo:

```
.DATA
...
STRING          LABEL       BYTE
QUANT_MAX       DB      20
QUANT_REAL      DB      ?
CARACTERES      DB      21 DUP (?)      ; área reservada para a digitação + CR final
...
.CODE
...
```

EXERCÍCIOS:

- **EXERCÍCIO 1**: FAÇA UM PROGRAMA QUE PEÇA AO USUÁRIO PARA DIGITAR CARACTERES NO TECLADO, LEIA E ARMAZENE O STRING (COM NO MÁXIMO 80 CARACTERES) E EM SEGUIDA EXIBA NA PRÓXIMA LINHA DO MONITOR OS 10 PRIMEIROS CARACTERES.

- **EXERCÍCIO 2:** MODIFIQUE O PROGRAMA ANTERIOR USANDO A FUNÇÃO **0AH** DA INTERRUPÇÃO **INT 21H** PARA LEITURA DO STRING. SUGESTÃO: ORGANIZE AS VARIÁVEIS NO **.DATA** COMO SEGUE

```
STRING          LABEL   BYTE
QUANT_MAX       DB      20
QUANT_REAL      DB      ?
CARACTERES      DB      21 DUP (?)
```

- **EXERCÍCIO 3:** CRIE UM PROGRAMA QUE PEÇA AO USUÁRIO PARA DIGITAR SUA SENHA, COMPARE-A COM UMA SENHA INTERNA JÁ PROGRAMADA E EXIBA MENSAGENS INFORMANDO SE TAL SENHA ESTÁ CORRETA OU ERRADA.

- **EXERCÍCIO 4:** EM APLICAÇÕES ENVOLVENDO PLANILHAS, É MUITO ÚTIL EXIBIR NÚMEROS JUSTIFICADOS PELA DIREITA, COMO POR EXEMPLO:

```
   1345
 342545
     56
```

  ESCREVA UM PROGRAMA QUE LEIA 3 NÚMEROS QUAISQUER ATÉ 10 DÍGITOS E EXIBA-OS NO MONITOR JUSTIFICADOS TAL COMO ACIMA.