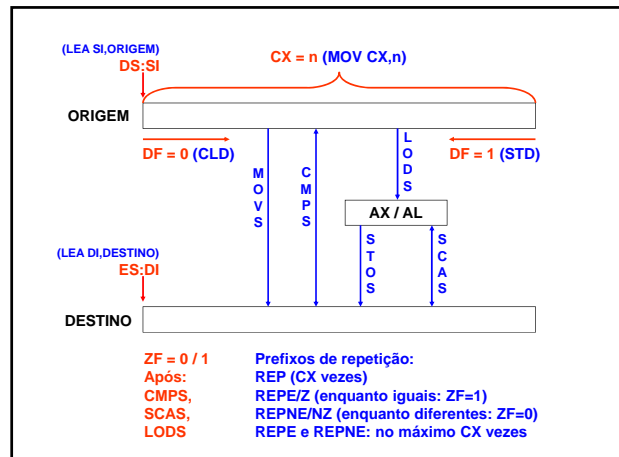


## Manipulação de cadeias de caracteres

movimentos memória a memória



## Instruções de manipulação de strings

- Registadores implícitos

[E]SI	índice para string fonte
[E]DI	índice para string destino
ES	segmento do string destino
[E]CX	contador
AL/AX/EAX	valor de busca (destino p/ LODS, fonte p/ STOS)
DF	0 (auto incremento p/ DI, SI) 1 (auto decremento p/ DI, SI)
ZF	condição de término para busca e comparação

## Instruções primitivas

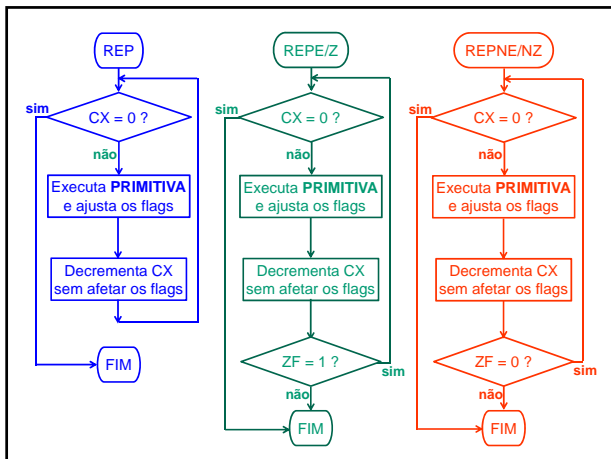
- MOVS** move source string to destination string
- CMPS** compare source string with destination string
- SCAS** scan destination string
- LODS** load into AL/AX/EAX from source string
- STOS** store AL/AX/EAX into destination string
- INS** input from I/O port (in DX) into destination
- OUTS** output from source to I/O port (in DX)

## Instruções primitivas

- Realiza operação primitiva
  - sobre um **byte**, uma **palavra** ou uma **palavra dupla**
- Atualiza registradores de índice (SI e/ou DI):
  - Se **DF=0**, então incrementa registradores de índice:
    - de um, se operação for a byte
    - de dois, se operação for a palavra
    - de quatro, se a operação for a palavra dupla
  - Se **DF=1**, então decrementa registradores de índice:
    - de um, se operação for a byte
    - de dois, se operação for a palavra
    - de quatro, se a operação for a palavra dupla

## Instruções de repetição

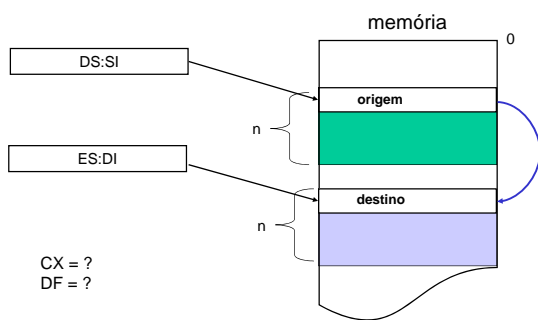
- REP repeat  
 REPE / REPZ repeat while equal/repeat while zero  
 REPNE/REPZ repeat while not equal/repeat while not zero
- Se **CX = 0**, então não executa a primitiva e encerra repetição
  - Se **CX > 0**, então:
    - Executa a operação primitiva, atualizando os flags
    - Decrementa CX, sem afetar os flags
    - Volta para a repetição, de acordo com o prefixo:
      - Se for REP, repete incondicionalmente
      - Se for REPE/Z, somente repete se ZF=1
      - Se for REPNE/NZ, somente repete se ZF=0



## Instruções Primitivas: MOVS

- MOVS **move** source string to destination string
  - MOVSB move um byte de DS:SI para ES:DI
  - MOVSW move uma palavra de DS:SI para ES:DI
  - MOVSD move uma palavra dupla de DS:SI para ES:DI
- REP MOVS - movimenta CX elementos
- REPE MOVS - não faz sentido usar
- REPNE MOVS - não faz sentido usar

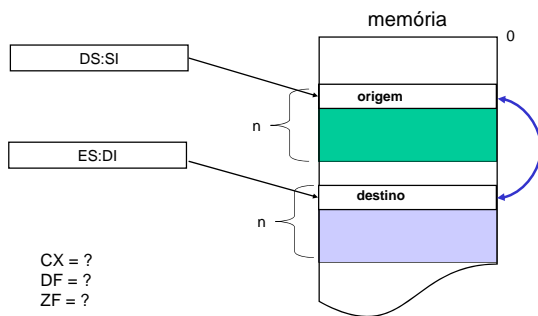
## MOVS



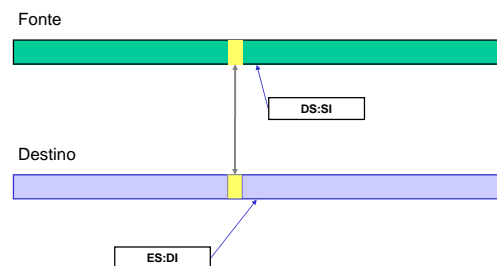
## Instruções Primitivas: CMPS

- CMPS
- **compare** source string with destination string
  - CMPSB subtrai byte ( $\text{mem}(\text{DS:SI}) - \text{mem}(\text{ES:DI})$ )
  - CMPSW subtrai palavra ( $\text{mem}(\text{DS:SI}) - \text{mem}(\text{ES:DI})$ )
  - CMPSD subtrai palavra dupla ( $\text{mem}(\text{DS:SI}) - \text{mem}(\text{ES:DI})$ )
  - afeta flags de forma idêntica a CMP
- REP CMPS - não faz sentido usar
- REPE CMPS - repete enquanto forem iguais
- REPNE CMPS - repete enquanto forem diferentes
- Obs: ponteiros param DEPOIS do ponto !!

## CMPS



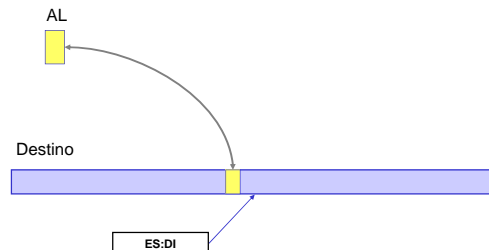
## Ponteiros após a comparação



### Instruções Primitivas

- SCAS scan **destination** string
  - SCASB subtrai byte em AL ( $AL - \text{mem}(\text{ES:DI})$ )
  - SCASW subtrai palavra em AX ( $AX - \text{mem}(\text{ES:DI})$ )
  - SCASD subtrai palavra dupla em EAX ( $EAX - \text{mem}(\text{ES:DI})$ )
  - AL, AX ou EAX contém valor sendo buscado
- REP SCAS - não faz sentido usar
- REPE SCAS - repete enquanto for igual
- REPNE SCAS - repete enquanto for diferente
- Obs: ponteiro pára DEPOIS do ponto !!

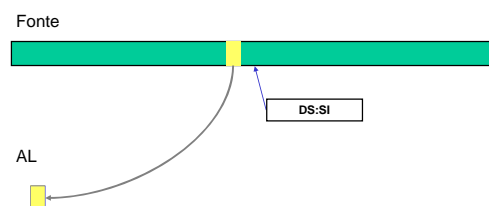
### Ponteiro após a varredura



### Instruções Primitivas

- LODS **load** into AL/AX/EAX from **source** string
  - LODSB carrega byte de mem(DS:SI) em AL
  - LODSW carrega palavra de mem(DS:SI) em AX
  - LODSD carrega palavra dupla de mem(DS:SI) em EAX
- REP LODS - não faz sentido usar
  - fica com o último elemento do string no acumulador
- REPE LODS
  - repete até carregar valor diferente de zero
- REPNE LODS
  - repete até carregar valor igual a zero

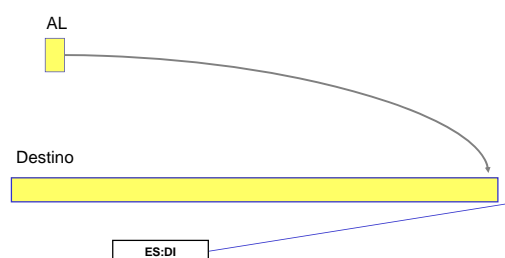
### Ponteiro após a carga



### Instruções Primitivas

- STOS store AL/AX/EAX into destination string
  - STOSB armazena AL no byte de mem (ES:DI)
  - STOSW armazena AX na palavra de mem (ES:DI)
  - STOSD armazena EAX na palavra dupla de mem (ES:DI)
- REP STOS
  - preenche o string com valor do acumulador
- REPE STOS
  - não faz sentido usar
- REPNE STOS
  - não faz sentido usar

### REP STOSB



### Instruções Primitivas

- **INS**      input from I/O port (in DX) into **destination**
  - **INSB**
    - move byte da porta especificada em DX para ES:DI
  - **INSW**
    - move palavra da porta especificada em DX para ES:DI
  - **INSD**
    - move palavra dupla da porta especificada em DX para ES:DI
- **REP INS** - lê CX elementos
- **REPE INS** - repete até ler valor diferente de zero
- **REPNE INS** - repete até ler valor igual a zero

### Instruções Primitivas

- **OUTS**      output from **source** to I/O port (in DX)
  - **OUTSB**
    - move byte de DS:SI para porta de E/S especificada em DX
  - **OUTSW**
    - move palavra de DS:SI para porta especificada em DX
  - **OUTSD**
    - move palavra dupla de DS:ES para porta especificada em DX
- **REP OUTS** - envia CX elementos para a saída
- **REPE OUTS** - não faz sentido usar
- **REPNE OUTS** - não faz sentido usar

### Exemplo 1 - copiar memória

- Copiar "Area1" para "Area2"
- Sejam
 

AREA1	DB	500 DUP (0)
AREA2	DB	500 DUP (0)

### Exemplo 1 - copiar memória

AREA1	DB	500 DUP (0)	byte a byte
AREA2	DB	500 DUP (0)	

; Supondo DS e ES inicializados adequadamente  
 LEA SI, AREA1  
 LEA DI, AREA2  
 MOV CX, 500  
 CLD  
 REP MOVSB

### Exemplo 1 - copiar memória

AREA1	DB	500 DUP (0)	word a word
AREA2	DB	500 DUP (0)	

; Supondo DS e ES inicializados adequadamente  
 LEA SI, AREA1  
 LEA DI, AREA2  
 MOV CX, 250                      ; ou então 125  
 CLD  
 REP MOVSW                      ; ou MOVSD

### Exemplo 2 - procurar e contar caractere

- Procurar quantas vezes o caractere "A" aparece em Area1
- Sejam:
 

AREA1	DB	500 DUP (0)
VEZES	DW	?

### Exemplo 2 – esboço de solução

- usar SCASB

- acha caracterer
- incrementa contador de caracteres achados
- volta ao loop


ZF=1, CX > 0



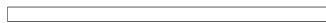
- condições de término

- chega ao fim da área
  - com caracter procurado na última posição
  - sem caracter procurado na última posição

ZF=1, CX = 0



ZF=0, CX = 0



### Exemplo 2 - procurar e contar caractere

```

AREA1      DB      500 DUP (0)
VEZES      DW      ?
; Supondo DS e ES inicializados adequadamente
LEA        DI, AREA1
MOV        CX, 500
CLD
MOV AL, 'A'
MOV        VEZES, 0
DENOVO:    REPNE SCASB
JNE        FIM
INC        VEZES
JCXZ       FIM
JMP        DENOVO
; se achou no fim do string

FIM:        ....
    
```

### Exemplo 3 - procurar caractere

- Procurar a posição (endereço) do primeiro caractere "A" em Area1.
- Colocar zero no endereço se não encontrar o caractere.
- Sejam:

```

AREA1      DB      500 DUP (0)
ENDereco   DW      ?
    
```

### Exemplo 3 - procurar caractere

```

AREA1      DB      500 DUP (0)
ENDereco   DW      ?
; Supondo DS e ES inicializados adequadamente
LEA        DI, AREA1
MOV        CX, 500
CLD
MOV AL, 'A'
MOV        ENDereco, 0
REPNE SCASB
JNE        FIM
DEC        DI
MOV        ENDereco, DI
FIM:        ....
    
```

### Exemplo 4 - concatenar strings

- Dados dois strings de caracteres, na mesma área, separados por um ou mais espaços (caractere 20H)
  - concatenar os dois strings, eliminando os espaços em branco
  - Obs: não existe nenhum espaço em branco "dentro" dos strings
- Sejam:

```

AREA1      DB 500 DUP (?) ; área que contém os strings
END1       DW 0           ; endereço do primeiro espaço encontrado
END2       DW 0           ; endereço do prim caractere do segundo string
    
```

### Exemplo 4 - concatenar strings

```

; Supondo DS e ES inicializados adequadamente
LEA        DI, AREA1
MOV        CX, 500
CLD
MOV        AL, 20H ; Código ASCII do caractere espaço
REPNE SCASB
JNE        FIM ; Não achou nenhum espaço
MOV        END1, DI
DEC        END1 ; Corrige endereço do primeiro espaço
REPE SCASB
JE         FIM ; Só tem espaços até o fim
MOV        END2, DI
DEC        END2 ; Corrige endereço do primeiro caractere
MOV        SI, END2 ; Endereço do último espaço + 1
MOV        DI, END1
INC        CX ; Corrige Contador
REP        MOVSB ; Concatena segundo string com o primeiro
    
```

### Exemplo de questão de prova 1

Supondo que DS e ES tenham sido inicializados adequadamente, e dada a mensagem abaixo, qual será a mensagem resultante após a execução do seguinte trecho de programa:

```
DADOS          SEGMENT
MENSAGEM       DB 'O Gremio nao vai entregar o jogo !',CR,LF
.....
DADOS          ENDS
CODIGO         SEGMENT
.....
                LEA     DI,MENSAGEM
                MOV     CX,34 ; tamanho da mensagem sem o CR,LF
                CLD
                MOV     AL,'n'
                REPNE   SCASB
                JNE     FIM      ; Não achou nenhum 'n'
                MOV     BYTE PTR [DI],' '
                MOV     BYTE PTR [DI+1],' '
                MOV     BYTE PTR [DI+1],' '
```

### Exemplo de questão de prova 2

Supondo que DS e ES tenham sido inicializados adequadamente, e dada a mensagem abaixo, qual será a mensagem resultante após a execução do seguinte trecho de programa:

```
DADOS          SEGMENT
MENSAGEM       DB 'A data de entrega do trabalho nao sera prorrogada !!!',CR,LF
.....
                LEA     DI,MENSAGEM
                MOV     SI,DI
                MOV     CX,53 ; tamanho da mensagem sem o CR,LF
                CLD
LACO:          LODSB
                CMP     AL,'a'
                JB      FIM_LACO
                CMP     AL,'z'
                JA      FIM_LACO
                AND     AL,0DFH
FIM_LACO:      STOSB
                LOOP    LACO
.....
```

### Exemplo de questão de prova 3

Quais das alternativas abaixo são soluções corretas para mover um bloco de 500 bytes de AREA1 para AREA2, dos endereços mais altos para os mais baixos ?

DADOS	SEGMENT	(c)	
AREA1	DB 500 DUP (?)	LEA	SI,AREA1+500
AREA2	DB 500 DUP (?)	LEA	DI,AREA2+500
.....		MOV	CX,125
		STD	
		REP	MOVSD
(a)		(d)	
LEA	DI,AREA1+498	LEA	SI,AREA1+496
MOV	SI,OFFSET AREA2+498	LEA	DI,AREA2+496
MOV	CX,250	MOV	CX,125
STD		REP	MOVSD
REP	MOVSW		
(b)		(e)	
LEA	SI,AREA1	LEA	SI,AREA1+499
MOV	DI,OFFSET AREA2	LEA	DI,AREA2+499
MOV	CX,500	MOV	CX,500
CLD		CLD	
REP	MOVSB	REP	MOVSB