Loop

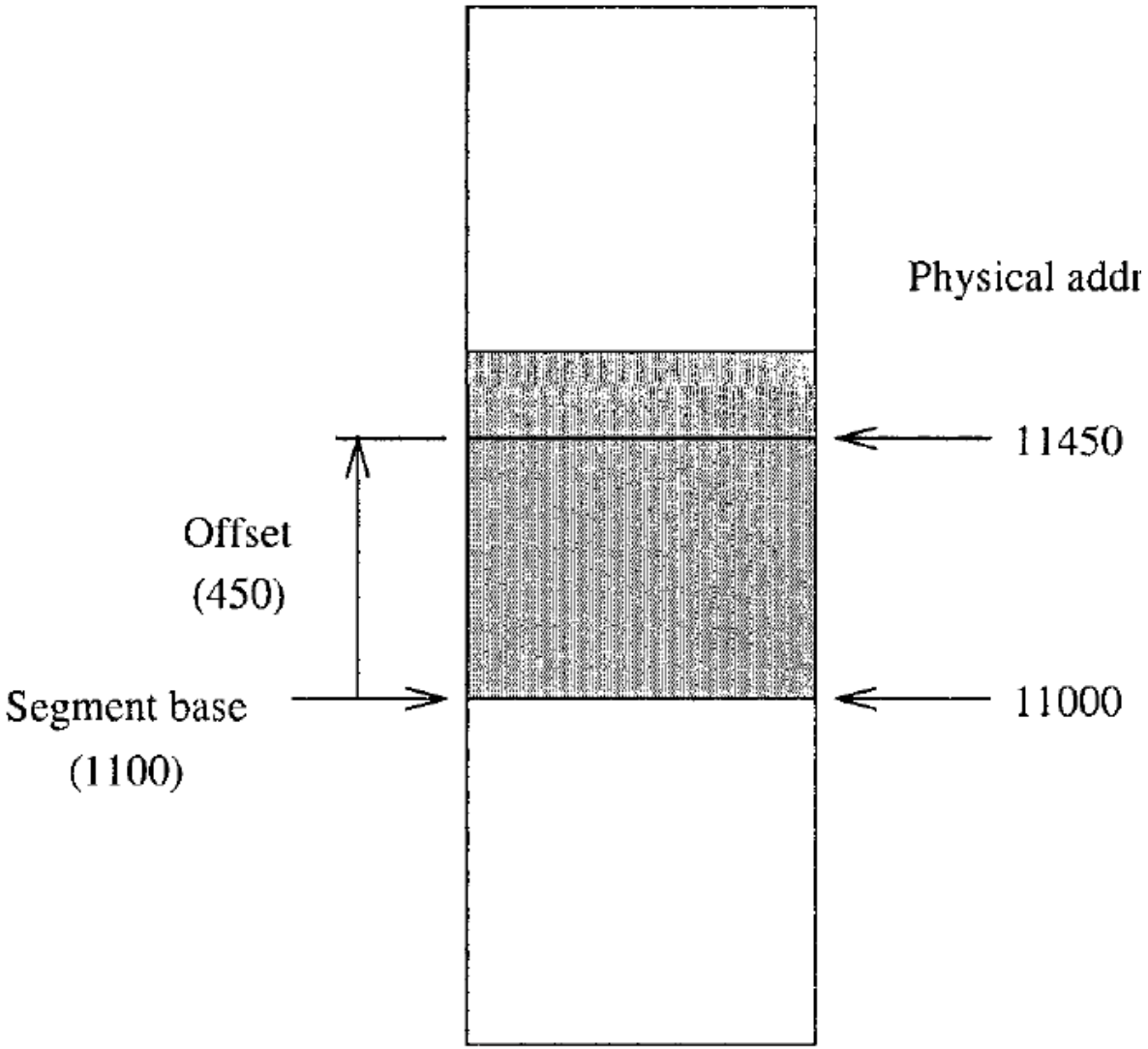
Loop

- Instrução de repetição que mantem a contagem no registrador cx

Mnemonic		Meaning	Action
`loop`	`target`	loop	ECX = ECX − 1 if CX ≠ 0 jump to target
`loope`	`target`	loop while equal	ECX = ECX − 1
`loopz`	`target`	loop while zero	if (ECX ≠ 0 and ZF = 1) jump to target
`loopne`	`target`	loop while not equal	ECX = ECX − 1
`loopnz`	`target`	loop while not zero	if (ECX ≠ 0 and ZF =0) jump to target

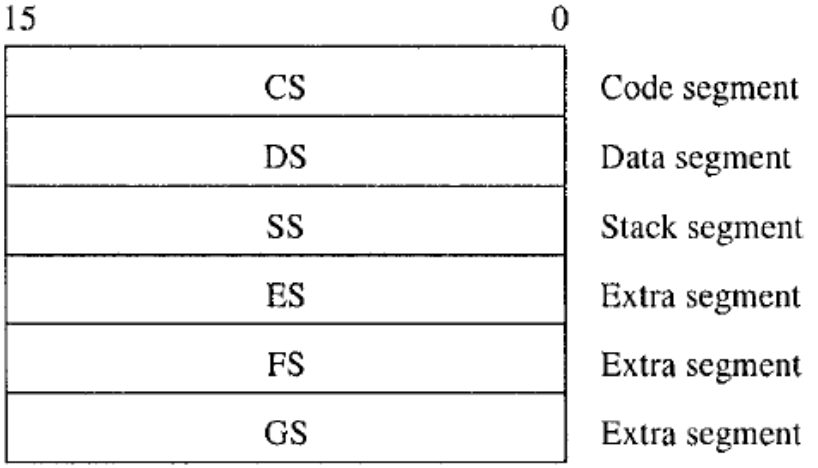
Operações com strings

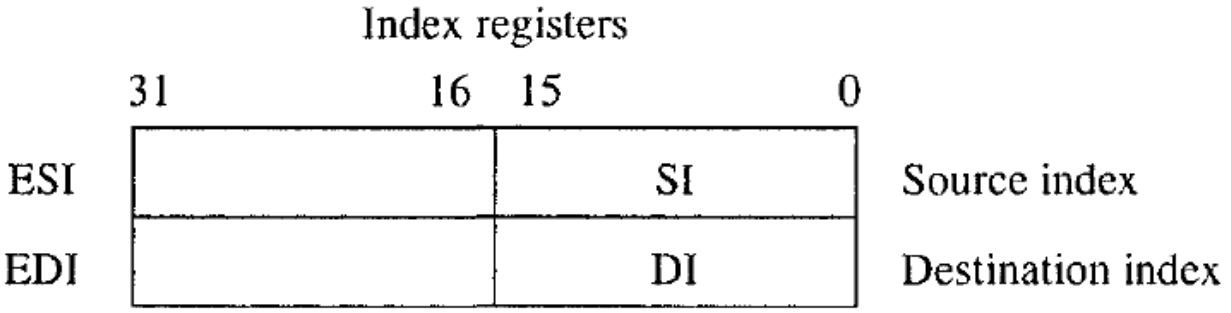
O Segmento base e o offset



Especificando operandos

- Cada instrução de strings pode requerer um operando fonte, um operando destino, ou ambos.
-





Instruções de manipulação de strings

- Existem 5 instruções para manipular strings:

Mnemonic	Meaning	Operand(s) required
LODS	LOaD String	source
STOS	STOre String	destination
MOVS	MOVe String	source & destination
CMPS	CoMPare Strings	source & destination
SCAS	SCAn String	destination

Prefixos de repetições

- São divididos em duas categorias: repetição condicional e incondicional.

unconditional repeat	
`rep`	REPeat
conditional repeat	
`repe/repz`	REPeat while Equal REPeat while Zero
`repne/repnz`	REPeat while Not Equal REPeat while Not Zero

Prefixos de repetições

Mnemonic	Loop while
`rep`	ECX>0
`repz/repe`	ECX>0 and ZF=1
`repnz/repne`	ECX>0 and ZF=0

Rep

- É um prefixo de repetição incondicional que repete enquanto o registrador ECX é diferente de zero.

```
while (ECX ≠ 0)
        execute the string instruction;
        ECX := ECX–1;
end while
```

Repe/Repz

- A diferença para o rep é que repe realiza repetição condicional em relação ao flag ZF.

```
while (ECX ≠ 0)
        execute the string instruction;
        ECX := ECX–1;
        if (ZF = 0)
        then
                exit loop
        end if
end while
```

Repne/Repnz

```
while (ECX ≠ 0)
        execute the string instruction;
        ECX := ECX–1;
        if (ZF = 1)
        then
                exit loop
        end if
end while
```

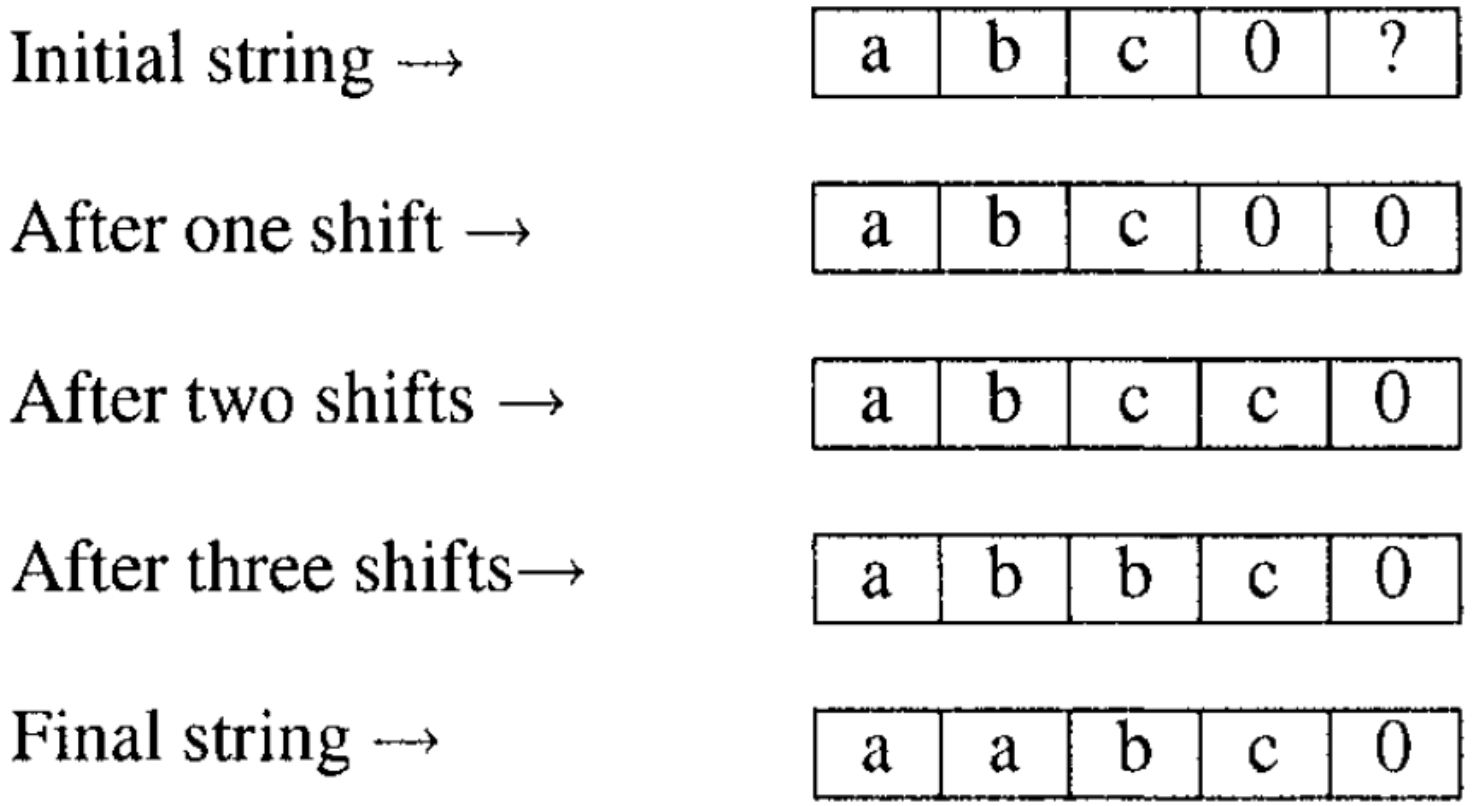
Flag de direção

- A direção da operação na string depende do flag de direção DF.
- Se DF=0 a operação procede da esquerda para a direita senão se DF=1 vai da direita para a esquerda.
- Algumas instruções para manipular o flag de direção:

`std`	set direction flag (DF = 1)
`cld`	clear direction flag (DF = 0)

Flag de direção

- Exemplo da direção da manipulação da direita para a esquerda



Instruções de movimentação de strings

- Movs, lods e stos são instruções de movimentação.
- Movs copia uma string do fonte para o destino

```
movs    dest_string,source_string
movsb
movsw
movsd
```

movsb

```
movsb — move a byte string
      ES:EDI := (DS:ESI)          ; copy a byte
      if (DF = 0)                 ; forward direction
      then
            ESI := ESI+1
            EDI := EDI+1

      else                        ; backward direction
            ESI := ESI–1
            EDI := EDI–1
      end if
```

movs

Mnemonic	Element size
rep movsb	byte
rep movsw	word
rep movsd	doubleword

Exemplo movsb

```
.DATA
string1     db     'The original string',0
strLen      EQU    $-string1
.UDATA
string2     resb     80
.CODE
    .STARTUP
    mov     ECX,strLen      ; strLen includes NULL
    mov     ESI,string1
    mov     EDI,string2
    cld                     ; forward direction
    rep     movsb
```

lods

- Copia um valor do fonte DS:ESI da memória para o registrador al (lodsb) ou ax (lodsw) ou eax (lodsd).

```
lodsb — load a byte string
        AL := (DS:ESI)      ; copy a byte
        if (DF = 0)         ; forward direction
        then
            ESI := ESI+1
        else                ; backward direction
            ESI := ESI−1
        end if
```

lods

Mnemonic	Element size
`lodsb`	byte
`lodsw`	word
`lodsd`	doubleword

stos

- Copia o valor em al (stosb) ou ax (stosw) ou eax (stosd) para o registrador ES:EDI na memória.

```
stosb — store a byte string
        ES:EDI := AL          ; copy a byte
        if (DF = 0)           ; forward direction
        then
            EDI := EDI+1
        else                  ; backward direction
            EDI := EDI–1
        end if
```

stos

Mnemonic	Element size
`stosb`	byte
`stosw`	word
`stosd`	doubleword
`rep stosb`	byte
`rep stosw`	word
`rep stosd`	doubleword

Exemplo stosw

```
.UDATA
array1    resw    100
.CODE
    .STARTUP
    mov     ECX,100
    mov     EDI,array1
    mov     AX,-1
    cld                     ; forward direction
    rep     stosw
```

cmps

- Pode ser utilizado para comparar duas strings.

```
cmpsb — compare two byte strings
        Compare the two bytes at DS:ESI and ES:EDI and set flags
        if (DF = 0)             ; forward direction
        then
            ESI := ESI+1
            EDI := EDI+1
        else                    ; backward direction
            ESI := ESI–1
            EDI := EDI–1
        end if
```

cmps

Mnemonic	Element size
cmpsb	byte
cmpsw	word
cmpsd	doubleword
repe cmpsb	byte
repe cmpsw	word
repe cmpsd	doubleword
repne cmpsb	byte
repne cmpsw	word
repne cmpsd	doubleword

Exemplo cmps

```
.DATA
string1     db     'abcdfghi',0
strLen      EQU    $-string1
string2     db     'abcdefgh',0
.CODE
     .STARTUP
     mov     ECX,strLen
     mov     ESI,string1
     mov     EDI,string2
     cld                        ; forward direction
     repe    cmpsb
```

scas

- Busca um valor ou caractere em uma string. O valor deve estar em al (scasb) ou ax (scasw) ou eax (scasd) e ES:EDI aponta para a string a ser consultada.

```
scasb — scan a byte string
        Compare AL to the byte at ES:EDI and set flags
        if (DF = 0)             ; forward direction
        then
            EDI := EDI+1
        else                    ; backward direction
            EDI := EDI−1
        end if
```

scas

Mnemonic	Element size
`scasb`	byte
`scasw`	word
`scasd`	doubleword
`repe scasb`	byte
`repe scasw`	word
`repe scasd`	doubleword
`repne scasb`	byte
`repne scasw`	word
`repne scasd`	doubleword

Exemplo scas

```
.DATA
string1     db     'abcdefgh',0
strLen      EQU    $ - string1
.CODE
     .STARTUP
     mov     ECX,strLen
     mov     EDI,string1
     mov     AL,'e'          ; character to be searched
     cld                     ; forward direction
     repne   scasb
     dec     EDI
```

Exercícios

- Faça um programa que compare duas strings e indique se elas são iguais
- Faça um programa que conte quantos caracteres existem em uma string
- Faça um programa que busque um caractere em uma string e diga a sua quantidade
- Faça um programa que remova os espaços em branco de uma string

Exercicios

- Faça um programa que dado uma string em maiúsculo converta os seus caracteres para minúsculos
- Faça um programa que converta os caracteres minúsculos de uma string em maiúsculos.

Exercicio

- Quais serão os valores de esi (00010000) , ecx e edi (00010005) durante a execução do programa abaixo ?

```
source    BYTE   "brown"
dest      BYTE   "brine"

 lea   esi, source
 lea   edi, dest
 cld
 mov   ecx, 5
 repne cmpsb
```