**Seminar 5 – String instructions**

|  |  |
| --- | --- |
| Without string instructions   * Initializari registrii index: 0 sau ultimul element din sir (in functie de probl) * Accesarea elementului current de prelucrat din sir in mod „manual” * Trecerea la urmatorul elem din sir: modificarea reg index in mod „manual” | Cu instructiuni speciale pentru siruri  Initializari:   * Se copiaza offset-ul/adresa sirului sursa in ESI * Se copiaza offset-ul/adresa sirului destinatie in EDI * Se reseteaza/seteaza DF (CLD => DF=0 si cu STD => DF=1) => se stabileste directia de parcurgere S->D (DF=0) sau D->S (DF=1)   Au operanzi impliciti si care sunt folosite astfel:   * Preia elementul curent al sirului automat * Trece la elementul urmator automat |
|  |  |
| Accesare 1 byte dintr-un sir s  Mov esi, 0  Mov al, byte [s+esi]  Add esi, 1  Salvare byte-ul al in sirul d  Mov edi, 0  Mov [d+edi], al  Add edi,1 | Cld ; stabilire dir de parcurgere: st -> dr  Mov esi, s ; initializare esi cu adresa/offsetul lui s  Lodsb  ; se copiaza in al byteul de la la adresa s+esi si creste esi cu 1  Mov edi, d  Stosb ; se copiaza un memorie, la adresa [d+edi], valoarea din reg al si creste edi cu 1 |
| Accesare 1 word dintr-un sir s  Mov esi, 0  Mov ax, word [s+esi]  Add esi, 2  Salvare word-ul ax in sirul d  Mov edi, 0  Mov [d+edi], ax  Add esi, 2 | Cld ; stabilire dir de parcurgere: st -> dr  Mov esi, s ; initializare esi cu adresa/offsetul lui s  Lodsw  ; se copiaza in ax wordul l de la la adresa s+esi si creste esi cu 2  Mov edi, d  Stosw ; se copiaza un memorie, la adresa [d+edi], valoarea din reg ax si creste edi cu 2 |
| Accesare 1 doublewordword dintr-un sir s  Mov esi, 0  Mov eax, dword [s+esi]  Add esi, 4  Salvare doubleword-ul eax in sirul d  Mov edi, 0  Mov [d+edi], eax  Add edi, 4 | Cld ; stabilire dir de parcurgere: st -> dr  Mov esi, s ; initializare esi cu adresa/offsetul lui s  Lodsd  ; se copiaza in eax doublewordul l de la la adresa s+esi si creste esi cu 4  Mov edi, d  Stosd ; se copiaza un memorie, la adresa [d+edi], valoarea din reg eax si creste edi cu 4 |

Obs. Pentru parcurgere de la drepata la stanga, initializam cu ultimul element si directia de parcurgere este de la dr la st (DF=1, instructiunea **STD – Set Direction Flag**).

**XCHG instruction:**

Xchg varmem, reg

XCHG reg, varmem

XCHG reg, reg

varmem and reg – same datatye (same dimensions)

ex:

al = 12h

bl = 34h

xchg al, bl => al =34h and bl = 12h

In caz de necesar suplimentar registrii index: ebp, ebx sau esp (initiializati conform cerintei probl)

Problems:

1. A strings of bytes (decimal, positive and negative numbers is given).

Create 2 new strings: p – only with positive numbers and n – with negative numbers.

s db 12h, 0Fah, -3, 0, 1

p: 12h, 0, 1

n: 0Fah, -3

|  |  |
| --- | --- |
| **Fara instructiuni speciale**  **Data segment**  s db 12h, 0Fah, -3, 0, 1  ls equ $ - s ; $ - contor locatii – det cati bytes s-au generat in memorie  n times ls db 0ffh; ff in spatiul rezervat sirului n  p times ls db 0eeh ; ee in spatiul rezervat sirul p  code segment:  mov esi, 0 ; s  mov edi, 0 ; pentru n  mov ebp, 0; pentru p  mov ecx, ls ; pentru a prelucra toate elem din sir  repeta:  mov al, [s+esi]  cmp al, 0  JL negative  JGE pozitive  negative:  mov [n+edi], al  inc edi  jmp end\_repeta  pozitive:  mov [p+ebp], al  inc ebp  end\_repeta:  inc esi  loop repeta | Cu instructiuni speciale  **Data segment**  s db 12h, 0Fah, -3, 0, 1  ls equ $ - s ; $ - contor locatii – det cati bytes s-au generat in memorie  n times ls db 0ffh; ff in spatiul rezervat sirului n  p times ls db 0eeh ; ee in spatiul rezervat sirul p  code segment:  mov esi, s ; s  mov edi, n ; pentru n  mov ebp, p; pentru p  mov ecx, ls ; pentru a prelucra toate elem din sir  ;stabilire dir de parcurgere st->dr  CLD  repeta:  lodsb ;al=[s+esi], inc esi  cmp al, 0  JL negative  JGE pozitive  negative:  stosb  jmp end\_repeta  pozitive:  ; ebp - > edi  Xchg ebp, edi  Stosb  Xchg ebp, edi  end\_repeta  Loop repeta |

2. A strings of **doublewords is given.**

Create a new **strings of words in** reversed order from the doublewords initial string.

s dd 12345678h, 1A2B3C4Dh

;la final d dw 3C4D, 1A2B, 5678, 1234

ls equ ($-s)/4 ; 1 dd = 4 bytes ; ls=2

d times ls\*2 dw -1

s in memory: conf little-endian, elem cu elem:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 78 | 56 | 34 | 12 | 4d | 3c | 2b | 1a | Val din s |
| S+0 | S+1 | S+2 | S+3 | S+4 | S+5 | S+6 (esi) | S+7 | adrese |

mov ebp, s

esi = s+4 , mov ax, word[s+esi], ax = 3c4d => ax-> [d+edi]= 3c4d, edi=edi+2

add esi, 2 , esi = s+6 , mov ax, word[s+esi], ax = 1a2b, ax =>[d+edi]= 1a2b, edi=edi+2

sub esi, 6

%xchg ebp, esi

|  |  |
| --- | --- |
| Code segment  Mov esi, ls \* 4 – 4 ; 1 dd= 4 bytes  Mov edi, 0  Mov ecx, ls  Repeta:  Mov ax, word[s+esi]  Mov [d+edi], ax  Add esi, 2  Add edi, 2  Mov ax, word[s+esi]  Mov [d+edi], ax  Add edi, 2  Sub esi, 6  Loop repeta | Code segment  Mov esi, s  Add esi, ls \* 4 – 4 ; 1 dd= 4 bytes  Mov edi, d  Mov ecx, ls  ; esi si edi sa creasca: adrese mici la adrese mari=> CLD  Repeta  Lodsw ; esi=esi+2  Stosw  Lodsw ; esi=esi+2  stosw  Sub esi, 8  Loop repeta |

3. A strings of **words is given in hexadecimal.**

For each **higher bytes,** compute the numbers of bits with value 1 and create a new string with the coressponding number of bits with value 1 for each **higher bytes** from the string.

Ex: s = 1234h, 1A28h, 00FFh

s in memorie:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 34 | 12 | 28 | 1a | ff | 00 |
| S+0 | S+1 | S+2 | S+3 | S+4 | S+5 |

Daca convertim:

s in binary: 0001 0010 0011 0100b, 0001 1010 0010 1000b, 0000 0000 1111 1111b

* d: 2, 3, 0

|  |  |
| --- | --- |
| Data segement  s dw 1234h, 1A28h, 00FFh  ls equ ($-s)/2  d times ls db -1  copie dd 0  Code segment  Mov esi, 0  Add esi, 1  Mov ecx, ls  Mov ebx, 0  Mov edi, 0  Repeta:  Mov al, byte[s+esi]  ; pentru al trebuie sa numaram bitii setati  Mov [copie], ecx  Mov ecx, 8  numara:  Shl al, 1  JC calculeazabitiisetati ;JC = JUMP IF CF=1  JNC end\_numara  calculeazabitiisetati:  inc bl ; nr de biti setati pt al  end\_numara:  Loop numara  Mov ecx, [copie]  Mov [d+edi], bl  Add edi, 1  Add esi, 2  Loop repeta | Code segment  Mov esi, s  Add esi, 1  Mov ecx, ls  Mov ebx, 0  Mov edi, d  CLD  Repeta:  losdb  ; pentru al trebuie sa numaram bitii setati  Mov [copie], ecx  Mov ecx, 8  numara:  Shl al, 1  JC calculeazabitiisetati ;JC = JUMP IF CF=1  JNC end\_numara  calculeazabitiisetati:  inc bl ; nr de biti setati pt al  end\_numara:  Loop numara  Mov ecx, [copie]  Mov al, bl  Stosb  Add esi, 1  Loop repeta |