**Seminar 3 – Reprezentarea little endian. Instructiuni de salt neconditionat. Instructiuni de salt conditionat. Operatii pe siruri**

**Reprezentarea little endian a numerelor in memorie**

*Pentru reprezentarea in memorie a unui numar (reprezentat pe mai mult de un octet) se aplica principiul little endian, adica* ***cel mai putin semnificativ octet din reprezentarea numarului se afla la adresa cea mai mica.***

Exemple:

A db 12h

B dw 8937h

C dd 76587643h

D dq 9876543212345678h

Cum se reprezinta in memorie?

|12h|37h|89h|43h|76h|58h|76h|78h|56h|34h|12h|32h|54h|76h|98h|

A B C D

Intrebari:

* Q: La ce adresa se gaseste octetul high din cuvantul low al dublucuvantului high din D?  
  A: La adresa D+5
* Q: La ce adresa se gaseste octetul low din cuvantul high al dublucuvantului C?  
  A: La adresa C+2

Exercitiul 1: Scrieti un program in limbaj de asamblare care determina valoarea expresiei x= a\*b + c\*d, unde a,b,c,d sunt reprezentate fara semn pe un cuvant

|  |  |
| --- | --- |
| Data segment | Code segment |
| Varianta 1 | |
| a dw 3456h  b dw 4320h  c dw 1234h  d dw 3300h  x dd 0 | Mov Ax,[a]  Mov Bx,[b]  Mul bx ; dx:ax = a\*b  Push dx  Push ax  Pop ecx ; ecx = a\*b  Mov ax,[c]  Mov bx,[d]  Mul bx; dx:ax = c\*d  Push dx  Push ax  Pop Edx ; edx = c\*d  Add ecx,edx ; ecx = a\*b + c\*d  Mov [x],ecx |
| Varianta 2 | |
| a dw 3456h  b dw 4320h  c dw 1234h  d dw 3300h  x dd 0 | Mov ax, [a]  Mov bx, [b]  Mul bx ; dx:ax = a \* b  Mov word[x], ax  Mov word[x + 2], dx; x = a \* b  Mov ax, [c]  Mul word [d]; dx:ax = c \* d  Add word[x], ax  Adc word[x + 2], dx |

Reprezentare in memorie:

|56h|34h|20h|43h|34h|12h|00h|33h|00h|00h|00h|00h|

A B C D X

**Instructiuni de salt neconditionat.**

* JMP eticheta – salt neconditionat la eticheta

**Instructiuni de salt conditionat.**

**“IF”** = o instructiune de comparatie (determina relatia de ordine dintre operanzi) si o instructiune de salt conditionat

**Instructiuni de comparatie**

* CMP op1, op2 – realizeaza un SUB op1, op2 FICTIV, adica op1 nu isi modifica valoarea, insa flag-urile sunt setate in mod corespunzator
* TEST op1, op2 - realizeaza un AND op1, op2 FICTIV, adica op1 nu isi modifica valoarea, insa flag-urile sunt setate in mod corespunzator

**Instructiuni de salt conditionat – verifica valoarea unor flag-uri si in functie de valoarea acestora efectueaza un salt conditionat la o eticheta**

Interpretarea fara semn (J = jump; B = below; A = above; E = equal; N = not):

* JB eticheta – jump if below (se face salt la eticheta daca op1 < op2)
* JBE eticheta – jump if below or equal (se face salt la eticheta daca op1 <= op2)
* JNB eticheta – jump if not below (se face salt la eticheta daca op1 >= op2)
* JNBE eticheta – jump if not below or qual (se face salt la eticheta daca op1 > op2)
* JA eticheta – jump if above (se face salt la eticheta daca op1 > op2)
* JAE eticheta – jump if above or equal (se face salt la eticheta daca op1 >= op2)
* JNA eticheta – jump if not above (se face salt la eticheta daca op1 <= op2)
* JNAE eticheta – jump if not above or qual (se face salt la eticheta daca op1 < op2)

Interpretarea cu semn (J = jump; L = less; G = greater; E = equal; N = not):

* JL eticheta – jump if less (se face salt la eticheta daca op1 < op2)
* JLE eticheta – jump if less or equal (se face salt la eticheta daca op1 <= op2)
* JNL eticheta – jump if not less (se face salt la eticheta daca op1 >= op2)
* JNLE eticheta – jump if not less or qual (se face salt la eticheta daca op1 > op2)
* JG eticheta – jump if greater (se face salt la eticheta daca op1 > op2)
* JGE eticheta – jump if greater or equal (se face salt la eticheta daca op1 >= op2)
* JNG eticheta – jump if not greater (se face salt la eticheta daca op1 <= op2)
* JNGE eticheta – jump if not greater or qual (se face salt la eticheta daca op1 < op2)

**Operatii pe siruri**

A dw 1234h

MOV AX, [A]

B DW 1,2,3,4,5,6,7,8,9

MOV AX, [B]  
MOV AX, [B+2]  
MOV AX, [B+4]  
….

* Pentru a opera cu siruri de octeti, cuvinte, dublucuvinte, avem nevoie de ADRESA COMPLETA, formata din:
  + OFFSET – o adresa data de numele unei variabile; offset-ul este un numar reprezentat pe 32 de biti si reprezinta un pointer in interiorul zonei de memorie
  + SELECTORUL DE SEGMENT – un numar reprezentat pe 16 biti care reprezinta un pointer catre zona de memorie si care este stocat intr-un registru (CS, DS, ES, SS) de catre sistemul de operare, inainte de inceperea executiei programului
* Pentru a defini un OFFSET se pot folosi 4 valori: BAZA, INDEX, SCALA si CONSTANTA

OFFSET = [BAZA] + [INDEX \* SCALA] + [CONSTANTA]  
 EAX EAX 1 o valoare reprezentata pe 8 biti  
 EBX EBX 2 o valoare reprezentata pe 16 biti  
 ECX ECX 4 o valoare reprezentata pe 32 biti  
 EDX EDX 8  
 ESI ESI  
 EDI EDI  
 ESP -  
 EBP EBP

Exemple:

MOV AX, [A]; pentru a defini offsetul s-a folosit CONSTANTA  
MOV AX, [B+4]; pentru a defini offsetul s-a folosit CONSTANTA  
MOV AX, [A+EBX+EAX]; constanta, index, baza  
MOV AX, [A+4+EBX\*2]; constanta, index, scala  
MOV AX, [EAX + EBX\*4 + 20]; baza, index, scala, constanta

Exercitiul 2: Se da un sir de octeti care contine litere mici. Se cere sa se construiasca un nou sir de octeti care sa contina literele din sirul initial transformate in majuscule.

‘a’ -> ‘A’

‘b’ -> ‘B’

…

LiteraMica - ‘a’ = LiteraMare - ‘A’ => LiteraMare = LiteraMica - ‘a’ + ‘A’ = LiteraMica - (‘a’ - ‘A’)

|  |  |
| --- | --- |
| Data segment | Code segment |
| VARIANTA 1 | |
| S db ‘abcdef’; <=> S db ‘a’,’b’,’c’,’d’,’e’  ; $ = contor de locatii  LS EQU $-S; $-S numara cati octeti au fost scrisi de la adresa lui S si pana in prezent  D TIMES LS db 0; D db 0,0,0,0,0,0 | MOV ESI, 0  repeta:  MOV AL, [S+ESI]  SUB AL, ‘a’-’A’  MOV [D+ESI], AL  INC ESI  CMP ESI, LS  JB repeta |
| VARIANTA 2 | |
| S db ‘abcdef’; <=> S db ‘a’,’b’,’c’,’d’,’e’  ; $ = contor de locatii  LS EQU $-S; $-S numara cati octeti au fost scrisi de la adresa lui S si pana in prezent  D TIMES LS db 0; D db 0,0,0,0,0,0 | MOV ESI, S  MOV EDI, D  MOV ECX, LS  JECXZ final  repeta:  MOV AL, [ESI]  SUB AL, ‘a’-’A’  MOV [EDI], AL  INC ESI  INC EDI  DEC ECX  CMP ECX,0  JA repeta  final: |

Reprezentarea in memorie

|’a’|’b’|’c’|’d’|’e’|’f’|00h|00h|00h|00h|00h|00h|

S D

|61h|62h|63h|64h|65h|66h|00h|00h|00h|00h|00h|00h|

S D

**Instructiuni de ciclare**

* LOOP eticheta – se reia executia blocului de instructiuni atata timp cat ECX != 0; LOOP efectueaza prima data DEC ECX, apoi se verifica daca ECX !=0 si apoi se face saltul la eticheta
* LOOPE eticheta – se reia executia blocului de instructiuni atata timp cat ECX != 0; Ciclul se termina fie daca ECX = 0 fie daca ZF = 0
* LOOPNE eticheta – se reia executia blocului de instructiuni atata timp cat ECX != 0; Ciclul se termina fie daca ECX = 0 fie daca ZF = 1

|  |  |
| --- | --- |
| Data segment | Code segment |
| VARIANTA 3 | |
| S db ‘abcdef’; <=> S db ‘a’,’b’,’c’,’d’,’e’  ; $ = contor de locatii  LS EQU $-S; $-S numara cati octeti au fost scrisi de la adresa lui S si pana in prezent  D TIMES LS db 0; D db 0,0,0,0,0,0 | MOV ESI, S  MOV EDI, D  MOV ECX, LS  JECXZ final  repeta:  MOV AL, [ESI]  SUB AL, ‘a’-’A’  MOV [EDI], AL  INC ESI  INC EDI  LOOP repeta  final: |

Exemple:

1) De cate ori se executa secventa?  
  
MOV ECX,5  
bucla:  
LOOP bucla

Raspuns:

P0: ECX = 5

P1: ECX = 4  
P2: ECX = 3  
P3: ECX = 2  
P4: ECX = 1  
P5: ECX = 0 => se iese din bucla dupa 5 pasi

2) De cate ori se executa secventa?  
  
MOV ECX,5  
bucla:  
 DEC ECX  
LOOP bucla

Raspuns:

P0: ECX=5

P1: ECX=4; ECX=3

P2: ECX=2; ECX=1

P3: ECX=0 ECX=-1

… => ciclu infinit

3) De cate ori se executa secventa?  
  
MOV ECX,6  
bucla:  
 DEC ECX  
LOOP bucla

Raspuns:

P0: ecx=6

P1: ECX=5; ECX=4

P2: ECX=3; ECX=2

P3: ECX=1; ECX=0 => 3 pasi

4) De cate ori se executa secventa?  
  
MOV ECX,12h  
bucla:  
 SHR ECX, 1  
LOOP bucla

Raspuns:

P0: ECX = 12h = 00010010b

P1: ECX = 00001001b; ECX = 00001000b

P2: ECX = 00000100b; ECX = 00000011b

P3: ECX = 00000001b; ECX = 0b => se iese din loop după 3 pași

5) Reprezentati in memorie urmatorul segment de date:

A1 db ‘24’, ‘68’; |’2 ‘|’4’|’6’|’8’|   
A2 dw 24, 68 ;|18h|00h|44h|00h|  
A3 db 2,4,6,8 ;|02h|04h|06h|08h|  
A4 dw 2,4,6,8 ;|02h|00h|04h|00h|06h|00|08h|00h|  
;A5 db 2468h ;2468h este word si nu incape pe byte => eroare  
A6 dw 24h, 68h ;|24h|00h|68h|00h|  
A7 dd 02040608h ;|08h|06h|04h|02h|  
A8 db 24h, 68h ;|24h|68h|  
A9 dw 2468h ;|68h|24h|

24 = 18h

68 = 44h

6) Se da un sir de cuvinte. Sa se adune in cuvantul A numerele formate de bitii 6-9 din fiecare cuvant al sirului.

Ex:   
S dw 1234h, 4567h, 7654h  
1234h = 000100**1000**110100b => 1000b = 8  
4567h …  
7654h …  
8+ … + … = rezultatul in cuvantul A

|  |  |
| --- | --- |
| Data segment | Code segment |
| Varianta 1 | |
| S dw 1234h, 4567h, 7654h LS EQU ($-S) / 2 A DW 0 | mov ebx, 0  repeta:  Mov ax, [s+ebx]  and ax, **1111**000000b  Mov cl, 6  Shr ax, cl  Add [A], ax  Add ebx, 2  **mov edx, ebx**  **shr edx,1**  **Cmp edx, LS**  JB repeta |
| Varianta 2 | |
| S dw 1234h, 4567h, 7654h LS EQU ($-S) / 2 A DW 0 | **MOV ESI, S**  **MOV ECX, LS**  **JECX final**  **Repeta:**  **Mov AX,[ESI]**  **PUSH ECX**   **MOV CL,6**  **SHR AX, CL**  **POP ECX**  **AND AX,1111b**  **ADD [A], AX**  **ADD ESI, 2**  **Loop Repeta**  **Final:** |