Reprezentarea little-endian a numerelor în memorie. Salt necondiționat, condiționat. Operații pe șiruri.

Reprezentarea little endian a numerelor in memorie

* Reprezentare in memorie a numerelor repr. Pe mai mult de un octet
* Principiul little endian: cel mai putin semnificati octet din reprezentarea numarului in memorie se afla la adresa cea mai mica

A db 12h

B dw 1234h

C dd 12345678h

D dq 1122334455667788h

Cum se reprezinta in memorie?

12h | 34h | 12h | 78h | 56h | 34h | 12h | 88h | 77h | 66h | 55h | 44h| 33h | 22h | 11h

La ce adresa se gaseste octetul high din cuvantul low al dublucuvantului high din d?

Dublucuvantul high: d+4

Cuvantul low din dublu cuvantul high: d+4

Octetul high din Cuvantul low din dublu cuvantul high: d+5

Ex1. X=a\*b+c\*d, a,b,c,d – cuvinte; I.f.s

|  |  |
| --- | --- |
| Data segment | Code segment |
| A dw 1  B dw 1  C dw 1  D dw 1  X dd 0 | Mov ax,[a]  Mov bx,[b]  Mul bx  ; dx:ax=a\*b  Mov [x],ax  Mov [x+2],dx  Mov cx,[c]  Mov ax,[d]  Mul cx  ; dx:ax=c\*d  Add [x],ax  Adc [x+2],dx |

Instructiuni de salt conditionat:

* Instructiuni similare cu instructiunea “IF”
* In asamblare o instructiune “IF” este compusa din:
  + O instructiune de comparare

TEST dest, sursa; dest AND sursa => fictiv! => se seteaza flag-urile

CMP dest, sursa; dest – sursa fictiv => seteaza flag-urile in mod coresp.

* + O instructiune de salt conditionat

**Instructiuni care interpreteaza numerele fara semn:**

J = jump; B = below; A = above; E = equal; N = not

JB eticheta; jump if below la eticheta <=> daca dest < sursa atunci sari la eticheta

JBE eticheta; <=

JNB eticheta; >=

JNBE eticheta; >

JA eticheta; >

JAE eticheta; >=

JNA eticheta; <=

JNAE eticheta; <

**Instructiuni care interpreteaza numerele cu semn**

J = jump; L = less; G = greater; E = equal; N = not

JL eticheta; <

JLE eticheta; <=

JNL eticheta; >=

JNLE eticheta; >

JG eticheta; >

JGE eticheta; >=

JNG eticheta; <=

JNGE eticheta; <

**Instructiuni care se uita la un singur flag**

JC eticheta

JNC eticheta

JP eticheta

JNP eticheta

JO eticheta

JNO eticheta

JS eticheta

JNS eticheta

JZ eticheta

JNZ eticheta

**Instructiuni care verifica valoarea din registrii CX/ECX**

JECXZ eticheta

JCXZ eticheta

**Instructiuni de salt necoditionat**

JMP eticheta

**Operatii pe siruri:**

A dw 7

...

Mov ax,[a]

b db 1,2,3,4,5

….

Mov al, [b+0]

Mov al, [b+1]

Mov al, [b+2]

…

C dw 1,2,3,4,5

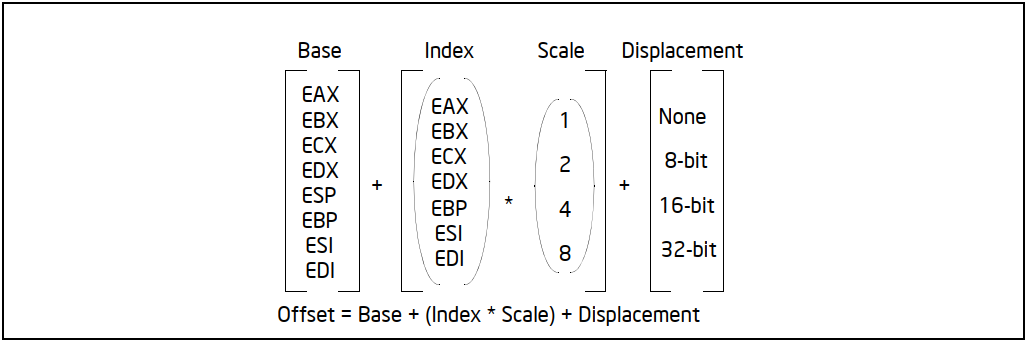
…

Mov ax,[c+0]

Mov ax,[c+2]

Mov ax,[c+4]

Adresa completa a unei variabile se face folosind selectorul segmentului (16 biti) si offsetul din cadrul segmentului (32 biti).



Mov ax, [b]; doar constanta

Mov ax, [b+eax]; constanta si baza/index

Mov ax, [b+eax\*2]; constanta si index si scala 2

Mov ax, [esp+esi]; baza esp si indexul esi si scala 1

Ex2. Se da un sir de octeti care contine litere mici. Sa se obtina un nou sir care sa contina literele mari corespunzatoare literelor mici din sirul initial

Varianta 1

|  |  |
| --- | --- |
| Data segment | Code segment |
| A db ‘a’,’b’,’c’,’d’,’e’ ; A db ‘abcde’  L equ $-A  D TIMES L db 0 | MOV ESI,0  Transforma:  MOV AL, [A+ESI]  SUB AL, ‘a’-’A’; ‘b’-’a’+’A’=1+’A’=’B’  MOV [D+ESI], AL  INC ESI  CMP ESI, L  JB Transforma  PUSH dword 0  CALL [exit] |

Varianta 2

|  |  |
| --- | --- |
| Data segment | Code segment |
| A db ‘a’,’b’,’c’,’d’,’e’ ; A db ‘abcde’  L equ $-A  D TIMES L db 0 | MOV ESI, A  MOV EDI, D  MOV ECX, L  JECXZ Final  Transforma:  MOV AL, [ESI]  SUB AL, ‘a’-’A’; ‘b’-’a’+’A’=1+’A’=’B’  MOV [EDI], AL  INC ESI  INC EDI  DEC ECX  CMP ECX, 0  JA Transforma  Final:  PUSH dword 0  CALL [exit] |

Varianta 3

**Instructiuni de ciclare:**

LOOP eticheta – reia executia blocului de instructiuni atata timp cat ECX!=0. Se efectueaza mai intai DEC ECX si apoi se verifica daca ECX!=0

LOOPE eticheta – reia executia blocului de instructiuni atata timp cat ECX!=0. Ciclul se termina daca ECX =0 sau ZF=0

LOOPNE eticheta – reia executia blocului de instructiuni atata timp cat ECX!=0. Ciclul se termina daca ECX =0 sau ZF=1

|  |  |
| --- | --- |
| Data segment | Code segment |
| A db ‘a’,’b’,’c’,’d’,’e’ ; A db ‘abcde’  L equ $-A  D TIMES L db 0 | MOV ESI, A  MOV EDI, D  MOV ECX, L  JECXZ Final  Transforma:  MOV AL, [ESI]  SUB AL, ‘a’-’A’; ‘b’-’a’+’A’=1+’A’=’B’  MOV [EDI], AL  INC ESI  INC EDI  LOOP Transforma  Final:  PUSH dword 0  CALL [exit] |

MOV ECX,5

Bucla:

DEC ECX

LOOP Bucla

De cate ori se executa secventa?

P0: ECX = 5

P1: ECX = 4, ECX =3

P2: ECX = 2, ECX = 1

P3: ECX = 0, ECX = -1

….

Ciclu infinit

MOV ECX,6

Bucla:

DEC ECX

LOOP Bucla

De cate ori se executa secventa?

P0: ECX = 6

P1: ECX = 5, ECX = 4

P2: ECX = 3, ECX = 2

P3: ECX = 1, ECX = 0 ! S-a terminat => 3 pasi

MOV ECX, 12h ;

Bucla:

SHR ECX,1

LOOP Bucla

De cate ori se executa secventa?

P0 : ecx = 00010010b -> ecx = 00001001b -> ecx = 00001000b

P1 : ecx = 00000100b -> ecx = 00000011b

P2 : ecx = 00000001b -> ecx = 00000000b

Raspuns : 3 ori