Arhitektura računara

Nizovi

Načini adresiranja x86 procesora

- **Opšti format adresiranja:**
  - **adresa(%baza, %indeks, množilac)**
  - adresa + baza + množilac*indeks
- **Neposredno:** `movl $5, %eax`
- **Registarsko:** `movl %ebx, %ecx`
- **Direktno:** `movb %al, bajt`
- **Indirektno:** `movw %ax, (%ebx)`
- **Indeksno:** `movw niz1(, %esi, 2), %dx`
- **Kombinacija:** `movl %ebx, niz2(%ebx, %edi, 4)`

Pristup elementima niza

```
.section .data
    niz: .long -3, 6, 7, 4, 12
.section .text
…
    movl $2, %eax                   # indeks -> eax
    movl niz(, %eax, 4), %ebx       # 7 -> ebx

    movl $niz, %eax                 # adresa niza -> eax
    addl $8, %eax                   # računanje adrese elementa
    movl (%eax), %ebx               # 7 -> ebx

    movl $niz, %ecx                 # adresa niza -> ecx
    movl $2, %eax                   # indeks -> eax
    movl (%ecx, %eax, 4), %ebx      # 7 -> ebx
```

Prikaz nizova u dibageru

- **32-bitni nizovi**
  - u liniju na vrhu uneti **niz@elem**, gde je “niz” ime niza, a “elem” broj elemenata koje treba prikazati
- Ostali nizovi:
  - *Data/Memory* dijalog

Suma elemenata niza, C program

```
#define NELEM 10
int niz[NELEM] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
int suma = 0;
int i;

for(i = 0; i < NELEM; i++) {
    suma = suma + niz[i];
}
```

Suma elemenata niza, asembler

- **Indeksno adresiranje**

```
NELEM = 10
niz: .long 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
suma: .long 0
…
main:
    movl $0, %esi       # indeksni registar
    movl $0, %eax       # eax se koristi za računanje sume
petlja:
    addl niz(, %esi, 4), %eax
    incl %esi
    cmpl $NELEM, %esi
    jl petlja
kraj:
    movl %eax, suma
    ...
```

- Prikaz u dibageru:
  - **niz(, %esi, 4)**

Suma elemenata niza, asembler

- **Indirektno adresiranje**

```
NELEM = 10
niz: .long 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
suma: .long 0
…
main:
    movl $niz, %esi     # registar za indirektno adresiranje
    movl $0, %eax       # eax se koristi za računanje sume
petlja:
    addl (%esi), %eax
    addl $4, %esi       # long zauzima 4 bajta
    cmpl $niz+NELEM*4, %esi
    jl petlja
kraj:
    movl %eax, suma
    ...
```

- Prikaz u dibageru:
  - **(%esi)**

Suma elemenata niza, asembler

- **Kombinacija adresiranja**

```
NELEM = 10
niz: .long 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
suma: .long 0
…
main:
    movl $niz, %esi         # bazni registar
    movl $NELEM-1, %edi     # indeksni registar
    movl $0, %eax           # eax se koristi za računanje sume
petlja:
    addl (%esi, %edi, 4), %eax
    decl %edi
    jns petlja
kraj:
    movl %eax, suma
    ...
```

- Prikaz u dibageru:
  - **(%esi, %edi, 4)**

Suma elemenata niza, asembler

- **Kombinacija adresiranja + loop**

```
NELEM = 10
niz: .long 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
suma: .long 0
…
main:
    movl $niz, %esi         # bazni registar
    movl $NELEM, %ecx       # indeksni registar
    movl $0, %eax           # eax se koristi za računanje sume
petlja:
    addl -4(%esi, %ecx, 4), %eax
    loopl petlja
kraj:
    movl %eax, suma
    ...
```

- Prikaz u dibageru:
  - **(%esi, %ecx, 4)**

Zauzimanje bloka memorije

- Direktiva **.fill** za zauzimanje memorije ima format:
  - **.fill broj_el, veličina_el, vrednost_el**
- Služi za zauzimanje memorije:
  - broj_el elemenata
  - dužine veličina_el bajtova (1, 2 ili 4)
  - sa inicijalnom vrednošću vrednost_el
- Parametri veličina_el i vrednost_el su opcioni
  - podrazumevane vrednosti su 1 i 0, respektivno
- Primer:
  - `niz1: .long 0,0,0,0,0,0,0,0,0,0`
  - `niz2: .fill 10, 4, 0`

Naredbe za množenje i deljenje

- **Množenje neoznačenih brojeva**
  - **`mulb operand`** `# al * operand → ax`
  - **`mulw operand`** `# ax * operand → dx:ax`
  - **`mull operand`** `# eax * operand → edx:eax`
- **Deljenje neoznačenih brojeva**
  - **`divb operand`** `# ax / operand → al, ostatak → ah`
  - **`divw operand`** `# dx:ax / operand → ax, ostatak → dx`
  - **`divl operand`** `# edx:eax / operand → eax, ostatak → edx`
- **Označeni brojevi**
  - **`imul, idiv`**