Nizovi u asembleru

Računarstvo i informatika, III godina

April 2013. godine

1 Rad sa nizovima

Niz je kontinualni blok podataka u memoriji. Svaki element niza mora biti istog tipa i mora da koristi isti broj bajtova za skladištenje. Zbog ovih osobina, nizovi omogućavaju efikasni pristup bilo kom svom članu na bilo kojoj poziciji (tj. indeksu).

Adresa svakog elementa niza može biti izračunata ako su poznate sledeće tri činjenice:

1. adresa prvog elementa niza
2. broj bajtova koji zauzima svaki element
3. indeks traženog elementa

I u asembleru je, kao i u C-u uobičajeno da indeks prvog člana niza bude 0.

U sledećem primeru koda su dati neki od načina za definisanje nizova. Ako se niz definiše u `.data` segmentu, dodeljuju mu se početne vrednosti (manuelno ili uz pomoć times direktive), dok se za nizove u `.bss` segmentu samo rezerviše odgovarajući prostor u memoriji:

```
segment .data
; definisi niz od 10 duplih reci inicijalizovanih na 1,2,...,10
a1 dd 1,2,3,4,5,6,7,8,9,10
; definisi niz od 10 reci inicijalizovanih na 0
a2 dw 0,0,0,0,0,0,0,0,0,0
; isto sto i prethodno, ali koriscenjem TIMES direktive
a3 times 10 dw 0
; definisi niz bajtova od 200 nula i 100 jedinica
a4 times 200 db 0
   times 100 db 1

segment .bss
; definisi niz od 10 neinicijalizovanih duplih reci
a5 resd 10
; definisi niz od 100 neinicijalizovanih reci
a6 resw 100
```

Pristup ovako definisanim članovima niza obavlja se korišćenjem bazne adrese niza kojoj se dodaje odgovarajući broj bajtova u zavisnosti od indeksa traženog člana i veličine jednog člana u bajtovima. U sledećem primeru, niz `niz1` je definisan kao niz bajtova, a `niz2` kao niz podataka tipa `word` (2 bajta).

```
mov al, [niz1]          ; al=niz1[0]
mov al, [niz1+1]        ; al=niz1[1]
mov [niz1+3], al        ; niz1[3]=al
```

```
mov ax, [niz2]          ; ax=niz2[0]
mov ax, [niz2+2]        ; ax=niz2[1] (ne niz2[2]!!!)
mov [niz2+6], ax        ; niz2[3]=ax
mov ax, [niz2+1]        ; ax=???
```

Obratiti pažnju na 5. liniju.

PRIMER 1.1 (OSNOVNI PRIMER) *Ucitati niz od 10 clanova i odrediti njihovu sumu.*

```
; Zadatak: Ucitati niz od 10 clanova i odrediti njihovu sumu
;
; kompajliranje i linkovanje
; nasm -f elf niz.asm
; gcc -m32 -o niz niz.o asm_io.o driver.c
;
;#include <stdio.h>
;#define DUZINA_NIZA 10
;
;int main()
;{
; unsigned int i, suma;
; unsigned int niz1[DUZINA_NIZA];
;
; for (i=0; i<DUZINA_NIZA; i++)
; {
; printf("Unesi clan niza: ");
; scanf("%u", &niz1[i]);
; }
;
; suma=0;
; for (i=0; i<DUZINA_NIZA; i++)
; suma+= niz1[i];
;
; printf("Suma clanova niza je %d\n", suma);
;
; return 0;
;}
;
extern printf, scanf
%define DUZINA_NIZA 10

segment .data
    poruka1 db "Suma clanova niza je %d", 10, 0
    poruka2 db "Clan niza: ", 0
    fmt     db "%d", 0

segment .bss
    niz1 resd 10

segment .text
    global  main
main:
```

```
    enter   0,0                 ; rutina za inicijalizaciju
    pusha

    ; Ovde pocinje koristan kod
    mov ecx, 0                  ; u ecx registru je brojac i
    mov esi, 0                  ; indeksni registar esi postavi na nulu

petlja_unos:
    cmp ecx, DUZINA_NIZA        ; uporedi ecx na DUZINA_NIZA
    je kraj_petlja_unos         ; ako je jednako, izadji iz petlje

    push ecx                    ; mora da se sacuva ecx, jer ga printf/scanf promene
    push poruka2                ; stampaj poruku i ucitaj clan niza
    call printf
    add esp, 4

    mov ebx, niz1               ; formiraj adresu u registru ebx
    add ebx, esi
    push ebx                    ; spusti adresu na stek
    push fmt
    call scanf
    add esp, 8
    pop ecx                     ; vrati sacuvanu vrednost ecx sa pocetka petlje

    add esi, 4                  ; povecaj esi za 4 (4-double word)
    inc ecx                     ; povecaj brojac za 1
    jmp petlja_unos
kraj_petlja_unos:

; Niz je ucitan, u sledecim linijama se racuna suma njegovih clanova
    mov edx, 0                  ; u edx registru se cuva suma
    mov ecx, DUZINA_NIZA    ; brojac postavljen na DUZINA_NIZA jer ide unazad
    mov esi, 0                  ; index registar postavljen na nulu

petlja:
    add edx, [niz1+esi]         ; edx += niz1[i]
    add esi, 4                  ; pomeri se za 4 bajta (jedan clan niza je duzine 4 bajta)
    loop petlja                 ; kraj petlje

    push edx                    ; rezultat
    push poruka1                ; poruka
    call printf
    add esp, 8

    popa
    mov eax, 0                  ; izlazni kod
    leave
    ret
```

Literatura

[1] Paul Carter, PC Assembly Language, 2006.