

# Arhitektura računara 1

## ASSEMBLER - 5. termin

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:

- adresa prvog elementa niza
- broj bajtova koji zauzima svaki element
- indeks traženog elementa

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

U sledecem 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:

```
1  segment .data
2  ; define array of 10 double words initialized to 1,2,...,10
3  a1          dd  1, 2, 3, 4, 5, 6, 7, 8, 9, 10
4  ; define array of 10 words initialized to 0
5  a2          dw  0, 0, 0, 0, 0, 0, 0, 0, 0, 0
6  ; same as before using TIMES
7  a3          times 10 dw 0
8  ; define array of bytes with 200 0's and then a 100 1's
9  a4          times 200 db 0
10         times 100 db 1
11
12 segment .bss
13 ; define an array of 10 uninitialized double words
14 a5          resd 10
15 ; define an array of 100 uninitialized words
16 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, kao i od veličine jednog člana u bajtovima:

```
1  mov  al, [array1]          ; al = array1[0]
2  mov  al, [array1 + 1]     ; al = array1[1]
3  mov  [array1 + 3], al     ; array1[3] = al
4  mov  ax, [array2]        ; ax = array2[0]
5  mov  ax, [array2 + 2]    ; ax = array2[1] (NOT array2[2]!)
6  mov  [array2 + 6], ax    ; array2[3] = ax
7  mov  ax, [array2 + 1]    ; ax = ??
```

Obratiti pažnju na liniju 5!!!

## Primer 1: Računanje zbir članova niza koji se unosi sa tastature

```
; Zadatak: Učitati niz od 10 članova 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 član niza: ");
;       scanf("%u", &niz1[i]);
;   }
;
;   suma=0;
;   for (i=0; i<DUZINA_NIZA; i++)
;       suma+= niz1[i];
;
;   printf("Suma članova niza je %d\n", suma);
;
;   return 0;
;}

#include "../asm_io.inc"
#define DUZINA_NIZA 10

segment .data
    poruka1      db      "Suma članova niza je ", 0
    poruka2      db      "Član niza: ", 0

segment .bss
    niz1         resd    10

segment .text
    global  asm_main
asm_main:
    enter 0,0          ; rutina za inicijalizaciju
    pusha

; Ovde počinje 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, izađi iz petlje

    mov  eax, poruka2    ; stampaj poruku i učitaj član niza
    call print_string
    call read_int       ; u eax se nalazi učitana vrednost

    mov  [niz1+esi], eax ; postavi član na odgovarajuće mesto u nizu
    add  esi, 4          ; povećaj esi za 4 (4-double word)
    inc  ecx             ; povećaj brojac za 1
    jmp  petlja_unos

kraj_petlja_unos:

; Niz je učitano, u sledećim linijama se računa suma njegovih članova
    mov  edx, 0        ; u edx registru se čuva 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 član niza je dužine 4 bajta)
    loop petlja         ; kraj petlje
```

```
mov    eax, poruka1      ; stampanje rezultata
call   print_string

mov    eax, edx          ; rezultat se nalazi u edx, kopiraj u eax
call   print_int
call   print_nl

popa
mov    eax, 0            ; vrati se nazad u C
leave
ret
```