Arhitektura računara 1

ASEMBLER - 6. termin

Zadatak 1: Formirati Fibonacijev niz od 40 clanova, a zatim u petlji stampati one clanove niza cije indekse korisnik unosi sa tastature, sve dok ne unese 0.

```
; Formirati Fibonacijev niz od 40 clanova, a zatim u petlji stampati one clanove niza
; cije indekse korisnik unosi sa tastature, sve dok ne unese 0.
;
;#include <stdio.h>
;
;int main()
;{
;      unsigned int fib[30];
;      unsigned int index, i;
;
;      fib[0]=1;
;      fib[1]=1;
;
;      for (i=2; i<40; i++)
;            fib[i] = fib[i-2]+fib[i-1];
;
;      printf("Unesite indeks fibonacijevog niza: ");
;      scanf("%u", &index);
;
;      while (index!=0)
;      {
;            printf("Clan Fibonacijevog niza sa unetim indeksom je %d\n", fib[index]);
;            printf("Unesite indeks fibonacijevog niza: ");
;            scanf("%u", &index);
;      }
;
;      return 0;
;}
;
%include "../asm_io.inc"

segment .data
       poruka1             db      "Unesite indeks fibonacijevog niza: ", 0
       poruka2             db     "Clan Fibonacijevog niza sa unetim indeksom je ", 0


segment .bss
       fib          resd    40   ; rezervacija niza od 40 double word clanova, svaki je 4
bajta


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

       ; Ovde pocinje koristan kod
       mov    dword [fib], 1            ; fib[0]=1
       mov    dword [fib+4], 1    ; fib[1]=1
       mov    ecx, 38                   ; u ecx se nalazi brojac, postavi ga na 38
       mov    esi, 8              ; indeksni registar esi postavi na 8, tj. fib[2]

petlja1:
```

```
	mov	eax, [fib+esi-8]	; eax je jednak fib[i-2], tj. 8 bajtova pre trenutnog clana
	add	eax, [fib+esi-4]	; dodaj na eax vrednost fib[i-1], tj. 4 bajta pre trenutnog
clana
	mov	[fib+esi], eax		; fib[i] = eax (izracunati zbir)
	add	esi, 4			; pomeri indeksni registar za 4 bajta
	loop	petlja1
kraj_petlja1:

	mov	eax, poruka1		; stampanje poruke da se unese indeks
	call	print_string
	call	read_int		; unos indeksa u eax
petlja2:
	cmp	eax, 0			; while (index!=0)
	je	kraj_petlja2		; izadji iz petlje ako je index==0
	mov	esi, eax		; esi=eax
	shl	esi, 2			; pomeri esi za 2 mesta ulevo, tj. pomnozi sa 4, da dobijes
mem. lokaciju clana
	mov	eax, poruka2		; stampaj poruku
	call	print_string
	mov	eax, [fib+esi]		; stampaj odgovarajuci clan niza
	call	print_int
	call	print_nl

	mov	eax, poruka1		; stampanje poruke da se unese novi indeks
	call	print_string
	call	read_int		; ucitavanje indeksa u eax
	jmp	petlja2			; skok na pocetak petlje

kraj_petlja2:

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

Zadatak 2: Formirati i odstampati niz od prvih n prostih brojeva gde se n unosi sa tastature. Da li je dati broj prost proveriti u potprogramu.

```
; formirati i odstampati niz od prvih n prostih brojeva
; gde se n unosi sa tastature. Da li je dati broj prost proveriti u potprogramu.
;
;#include <stdio.h>
;
;int prost_broj;

;int prost(int x)
;{
;      int cinilac=2;
;
;      while(cinilac<x && x%cinilac!=0)
;            cinilac+=1;
;
;      if (x==cinilac)
;            prost_broj = 1;
;      else
;            prost_broj = 0;
;}
;
;int main()
;{
;      unsigned int niz[100];
;      unsigned int n, broj, brojac_niz, i;
;
;      printf("Unesite N: ");
;      scanf("%d", &n);
;
;      brojac_niz = 0;
;      broj = 1;
;      while (brojac_niz<n)
;      {
;            prost(broj);
;            if (prost_broj) niz[brojac_niz++] = broj;
;            broj++;
;      }
;
;      for (i=0; i<n; i++)
;            printf("%d ", niz[i]);
;      printf("\n");
;
;      return 0;
;}
;

%include "../asm_io.inc"

segment .data
      poruka1             db      "Unesite N: ", 0

segment .bss
      niz          resd    100  ; rezervacija niza od 100 double word clanova, svaki je 4
bajta
      n            resd   1     ; broj clanova niza
      duzina_niza  resd   1     ; ukupna duzina niza
      prost_broj   resb   1     ; boolean promenljiva koja ima vrednost 1 ako je broj
prost, 0 ako nije


segment .text
        global  asm_main
asm_main:
```

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

	; Ovde pocinje koristan kod
	mov    eax, poruka1        ; stampaj poruku o unosu n
	call   print_string
	call   read_int            ; unos sa tastature
	mov    [n], eax            ; n=eax
	shl    eax, 2              ; eax *= 4, pomeranje ulevo za 2 mesta
	mov    [duzina_niza], eax  ; duzina_niza = 4*n, ukupna duzina niza u bajtovima

	mov    ecx, 1              ; brojac se nalazi u ecx, pocinje od 1
	mov    esi, 0              ; indeksni registar se setuje na 0
petlja1:
	cmp    esi, [duzina_niza]  ; da li se stiglo do kraja niza?
	je     kraj_petlja1        ; ako jeste, izadji iz petlje
	mov    eax, ecx            ; eax=ecx
	call   prost_sub           ; pozovi potprogram
	cmp    byte [prost_broj], 0      ; [prost_broj]==0 ?
	je     nije_prost          ; ako jeste, preskoci THEN blok
	mov    [niz+esi], ecx            ; dodaj broj na niz
	add    esi, 4              ; pomeri se za 4 bajta, tj. predji na sledeci clan
nije_prost:
	inc    ecx                 ; povecaj brojac u ecx za 1, ecx++
	jmp    petlja1                   ; skoci na pocerak petlje
kraj_petlja1:
	;
	; obicna petlja za stampanje niza
	;
	mov    ecx, [n]            ; brojac u ecx se setuje na n
	mov    esi, 0              ; indeksni registar na pocetkku niza
petlja_stampa:
	mov    eax, [niz+esi]            ; stampaj clan
	call   print_int
	mov    eax, " "            ; stampaj "space"
	call   print_char
	add    esi, 4              ; pomeri se na sledeci clan
	loop   petlja_stampa       ; skoci na pocetak petlje, ecx++

	call   print_nl

	popa
	mov     eax, 0               ; vrati se nazad u C
	leave
	ret
;
; Potprogram za odredjivanje da li je broj koji se nalazi u registru eax prost.
; Potprogram vraca vrednost 0 u [prost_broj] ako broj nije prost, a 1 ako jeste.
;
segment .bss
	broj   resd   1            ; broj za koji se trazi da li je prost

segment .text

prost_sub:
	pusha                      ; stavi vrednosti registara na stek
	mov    [broj], eax         ; [broj] = eax, zapamti broj za koji se trazi da li je
prost
	mov    ebx, 2              ; u registru ebx se cuva cinilac = 2
while_cinilac:
	cmp    ebx, [broj]         ; uporedjivanje
	jnb    end_while_cinilac   ; if !(cinilac < broj)
	mov    eax, [broj]         ; eax = [broj], deljenik
	mov    edx, 0              ; pri deljenju, uvek je edx = 0
	div    ebx                 ; podeli sa ebx, ostatak je edx = edx:eax % ebx
	cmp    edx, 0              ; da li je ostatak 0 ?
```

```
       je     end_while_cinilac   ; if (broj % cinilac == 0)
       inc    ebx                 ; cinilac++;
       jmp    while_cinilac
end_while_cinilac:

       ; if uslov koji vraca nulu ili jedinicu na [psrost_broj]
       mov    byte [prost_broj], 0      ; postavi eax na 0 <=> podrazumevano broj nije prost
       cmp    [broj],      ebx
       jne    end_if              ; if !(broj==cinilac) izadji iz ifa
       mov    byte [prost_broj], 1      ; broj je prost, vrati 1 na [prost_broj]
end_if:
       popa                       ; vrati vrednosti registara postavljene na pocetku
podprograma
       ret
```