UNIVERZITET CRNE GORE
ELEKTROTEHNIČKI FAKULTET
STUDIJE PRIMIJENJENOG RAČUNARSTVA

SEMINARSKI RAD

Projektni zadatak broj 10

PROJEKAT

Predmetni profesor:

Prof. dr Nevena Radović

Studenti:

Vuk Adžić 1/17
Teodora Petranović 12/17
Anja Mičetić 34/17

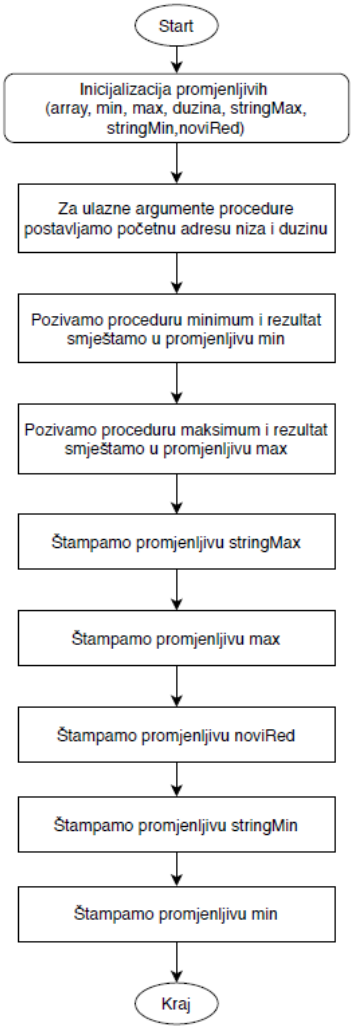
Sadržaj:

1. Opis zadatka

Tema ovog seminarskog rada je realizacija projektnog zadatka koji za ulazni podatak ima niz cijelih brojeva dužine 15 elemenata. Pomenuti niz je unaprijed zadat i sastoji se iz sljedećih elemenata: 13, 34, 16, 61, 28, -24, 58, 11, 26, 41, 0, 7, -38, 12, 13. Ideja zadatka je da se odredi najveći i najmanji element niza i to realizacijom i pozivom procedura maximum i minimum. Nakon poziva procedura, glavni program treba da vrati poruke sljedećeg sadržaja: "Maksimum zadatog niza je: X" i "Minimum zadatog niza je: Y", pri čemu umjesto X i Y treba ispisati najveći i najmanji element niza, respektivno. Program mora raditi za sve nizove cijelih brojeva koji sadrže 15 elemenata.

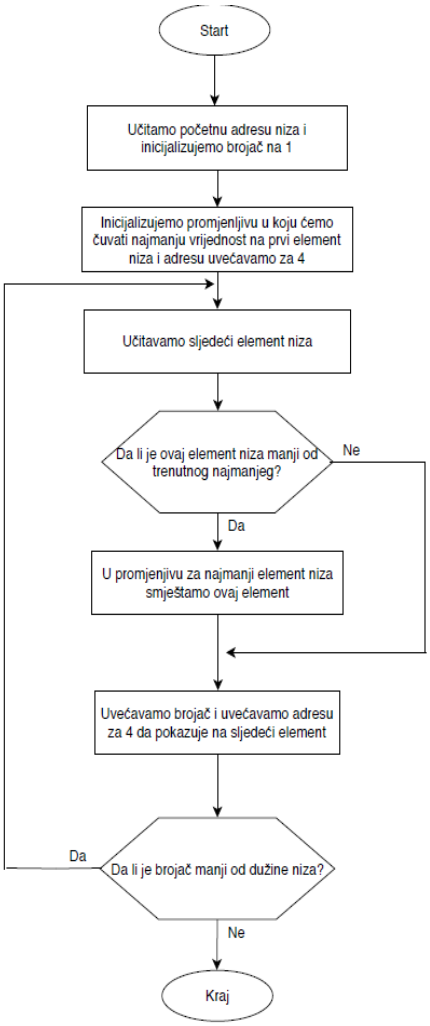
2. Algoritamske šeme

Pri rješavanju nekog problema, dobar je pristup podijeliti ga na manje djelove i rješavati ih pojedinačno, a na kraju ih povezati u cjelinu. U tu svrhu se koriste algoritamske šeme koje prikazuju tok rješavanja svakog od djelova. U nastavku će biti prikazani algoritmi za predmetni projektni zadatak, i to za glavni program i dvije procedure koje glavni program poziva.



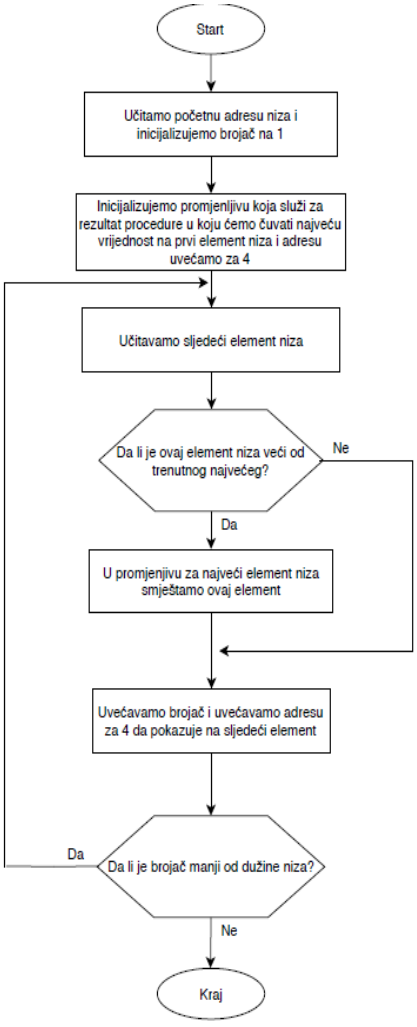
Slika 1.

Algoritam glavnog programa



Slika 2.

Algoritam procedure minimum



Slika 3.

Algoritam procedure maximum

3. Realizacija programa u višem programskom jeziku C++

Radi pravljenja paralele između programiranja u asembleru i u višim programskim jezicima, prikazano je rješenje projektnog zadatka u programskom jeziku C++.

```
#include <iostream>
using namespace std;

int minimum ( int *x, int N )
 {
    int i;
    int min = x[0];
    for ( i=1; i<N; i++ ){
       if( x[i] < min )
          min = x[i];
    };
    return min;
 }
int maximum ( int *x, int N )
 {
    int i;
    int max = x[0];
    for ( i=1; i<N; i++ )
    {
       if ( x[i] > max )
          max = x[i];
    };
    return max;
 }
```

```
int main()
    {
      int niz[ ] = {13,34,16,61,28,-24,58,11,26,41,0,7,-38,12,13};

      int min = 0;

      int max = 0;

      int duzina = 15;

      min = minimum ( niz, duzina );

      max = maximum ( niz, duzina );

      cout << "Minimum zadatog niza je: " << min;

      cout << "\nMaksimum zadatog niza je: " << max;
    }
```

Kao što će se u nastavku vidjeti, u poređenju sa mips kôdom, kôd u višem programskom C++ jeziku je znatno kraći i jednostavniji da bi bio razumljiviji programerima. Međutim, da bi bio razumljiv samom računaru, kôd se mora prvo kompajlirati, a zatim asemblirati, dok se mips kôd ne kompajlira i samim tim izvršavanje je znatno brže, ali je sintaksa kompleksnija.

4. Realizacija programa u MIPS asemblerskom jeziku

4.1 Deklaracija promjenljivih

Na samom početku vrši se deklaracija promjenljivih u okviru sekcije za zapis podataka (data section). Ova sekcija počinje asemblerskom direktivom *.data*, a deklaracija samih promjenljivih se vrši na sljedeći način:

<naziv_promjenljive>: *.<tip_podatka>* <početna_*vrijednost*>

Tip podatka .word označava 32-bitnu integer (cjelobrojnu) vrijednost. Ovaj tip podatka je izabran kako bi se spriječilo moguće prekoračenje u memoriji, odnosno da bi program bio u mogućnosti da radi i sa većim brojevima (iz opsega od −2,147,483,648 do 2,147,483,647).

Tip podatka .asciiz označava ASCII string terminisan nulom.

```
.data
array: .word 13, 34, 16, 61, 28, -24, 58, 11, 26, 41, 0, 7, -38, 12, 13
min: .word 0
max: .word 0
duzina: .word 15
stringmax: .asciiz "Maksimum zadatog niza je: "
stringmin: .asciiz "Minimum zadatog niza je: "
novired: .asciiz "\n"
```

U promjenljivu *array* se smješta niz od 15 cjelobrojnih elemenata koji su zadati projektnim zadatkom. Promjenljive *min* i *max* se koriste za smještanje rezultata procedura i na početku su inicijalizovane na nulu. Dužina je takođe zadata zadatkom (inicijalizovana na 15) i koristiće se pri prolasku kroz niz kao granična vijednost.

Preostale tri promjenljive su tipa *.asciiz* (string). *Stringmax* i *stringmin* služe pri ispisivanju rezultata programa - da bi korisniku bilo jasnije šta se ispisuje, dok se promjenljiva *novired* koristi za prelazak u novi red radi preglednosti pri ispisivanju rezultata.

4.2 Glavni program

Nakon deklaracije promjenljivih (data section) slijedi sam kod zadatka. On počinje direktivom *.text* koja se navodi na početku sekcije za zapis programskog koda (text section). Početak glavne procedure se označava *.globl main* direktivom i labelom *main:*

```
.text
.globl main
main:
```

U nastavku je prikazan kôd glavnog programa.

```
la $a0, array   # $a0 -> adresa prvog elementa niza
lw $a1, duzina  # $a1 -> dužina niza

jal minimum     # poziv procedure minimum
sw $v0, min     # min -> $v0

jal maximum     # poziv procedure maximum
sw $v0, max     # max -> $v0
```

Za prosljeđivanje argumenata procedurama koriste se registri $a0 - $a3. Prema tome, prije poziva procedura *minimum* i *maximum*, potrebno je dužinu niza i adresu prvog elementa smjestiti u odgovarajuće registre ($a0 i $a1).

Pomoću instrukcije *la* (load address) u registar $a0 učitava se adresa prvog elementa niza *array,* dok se instrukcijom *lw* (load word) u registar $a1 smješta dužina niza.

Pozivanje procedure se vrši instrukcijom *jal* (jump and link) kojom se „skače“ na proceduru i adresa sljedeće linije kôda se smješta u registar $ra koji služi za vraćanje iz procedure u glavni program, odnosno na narednu instrukciju poslije njenog poziva. Rezultat procedure se smješta u registre $v0 i $v1.

Instrukcijom *jal minimum* se poziva procedura *minimum*, a zatim se njen rezultat iz registra $v0 smješta u promjenljivu *min* instrukcijom *sw* (store word). Slično, instrukcijom *jal maximum* se poziva procedura *maximum*, a zatim se njen rezultat iz registra $v0 smješta u promjenljivu *max* instrukcijom *sw* (store word).

```
li $v0, 4          # call code, print string - stringmax
la $a0, stringmax
syscall            # system call

li $v0, 1          # call code, print integer - max
lw $a0, max
syscall            # system call

li $v0, 4          # call code, print string - novired
la $a0, novired
syscall            # system call

li $v0, 4          # call code, print string - stringmin
la $a0, stringmin
syscall            # system call

li $v0, 1          # call code, za print integer - min
lw $a0, min
syscall            # system call
```

Štampanje promjenljivih zahtijeva pozivanje specifičnih funkcija koje nudi QtSpim Simulator – pozivni kodovi (call code). Ove funkcije se pozivaju tako što se u registar $v0 učita odgovarajući pozivni kod instrukcijom *li* (load immediate). Konkretnije, 1 za štampanje integer vrijednosti (print integer), a 4 za štampanje asciiz promjenljivih (print string). Nakon toga je potrebno u registar $a0 učitati adresu/vrijednost promjenljive koja se štampa instrukcijama *la* (load address) i *lw*

(load word). Na kraju, da bi se samo štampanje izvršilo, potreban je sistemski poziv (*syscall*).

Za štampanje promjenljive *stringmax* učitavamo 4 u registar $v0 i adresu promjenljive u registar $a0, a zatim za štampanje vrijednosti promjenljive *max* učitavamo 1 u registar $v0 i vrijednost promjenljive u registar $a0. Nakon toga radi preglednijeg prikaza štampamo novi red korišćenjem pozivnog koda za print string. Promjenljive *stringmin* i *min* štampaju se na gore pomenuti nacin.

```
li $v0, 10      # terminate
syscall         # system call
.end main
```

Za završetak programa potreban je pozivni kod 10 i sistemski poziv. Glavna procedura *main* se završava direktivom *.end main* koja je ujedno i posljednja linija u izvršavanju programa.

4.3 Procedura *minimum*

Ova procedura za ulazne argumente ima adresu prvog elementa niza *array* i dužinu niza, a kao rezultat vraća najmanji element tog niza.

```
minimum:
add $t0,$a0,0           # $t0 -> $a0
li $t1,1                # $t1 -> 1
lw $v0,($t0)            # $v0 -> array[0]
add $t0,$t0,4           # $t0 = $t0 + 4
```

U pomoćni registar $t0 smješta se adresa prvog elementa niza *array* tako što se ulazni argument ($a0) instrukcijom *add* sabere sa 0 i rezultat smjesti u $t0. Za prolazak kroz niz koristi se brojač i to na način što se u registar $t1 smješta 1 instrukcijom *li* (load immediate). Zatim, instrukcijom *lw* (load word) u registar $v0 smješta se prvi element niza i time se za trenutni minimum postavlja upravo taj element. Potom, adresa se uvećava za 4 kako bi se prešlo na naredni element.

```
Loop:
  lw $t2,($t0)         # $t2 -> trenutni element niza
  ble $v0,$t2,Label    # if ($v0 ≤ $t2) -> Label
  add $v0,$t2,0        # $v0 -> $t2
Label:
  add $t1,$t1,1        # $t1 = $t1 + 1
  add $t0,$t0,4        # $t0 = $t0 + 4
  blt $t1,$a1,Loop     # if ($t1 < $a1) -> Loop
```

Ovaj dio kôda predstavlja petlju kojom se prolazi kroz niz od drugog do posljednjeg elementa i poredi svaki sa trenutno najmanjim. Ukoliko je element koji se nalazi u registru $t2 manji od trenutno najmanjeg ($v0), u registar $v0 se upisuje taj element ($t2) i on postaje novi minimum. Ipak, ukoliko ovaj uslov nije zadovoljen, „skače“ se na *Label* naredbom za uslovni skok *ble* (be less or equal than).

Ispod označene labele (*Label*) nalaze se naredbe vezane za prelazak na naredni element. Naredbom *add* brojač se prvo uvećava za 1, a zatim se pomoću iste naredbe adresa uvećava za 4 i prelazi se na naredni element niza. Na kraju petlje se provjerava da li je brojač ($t1) i dalje manji od dužine niza ($a1). Ako jeste, "skače" se na labelu *Loop,* a u suprotnom izvršavaju se naredne linije koda. Ovo se postiže branch instrukcijom *blt* (be less than).

```
jr $ra        # povratak iz procedure u glavni program
```

Instrukcija *jr* (jump register) se koristi za povratak u glavni program, odnosno na sljedeću liniju glavnog programa nakon poziva procedure, čija se adresa nalazi u registru *$ra*.

4.3 Procedura *maximum*

Sličan postupak korišćen je i za određivanje najvećeg elementa niza, ali će radi lakšeg razumijevanja biti objašnjen cijeli postupak.

Ova procedura za ulazne argumente ima adresu prvog elementa niza *array* i dužinu niza, a kao rezultat vraća najveći element tog niza.

```
maximum:
add $t0,$a0,0           # $t0 -> $a0
li $t1,1                # $t1 -> 1
lw $v0,($t0)            # $v0 -> array[0]
add $t0,$t0,4           # $t0 = $t0 + 4
```

U pomoćni registar $t0 smješta se adresa prvog elementa niza *array* tako što se ulazni argument ($a0) instrukcijom *add* sabere sa 0 i rezultat smjesti u $t0. Za prolazak kroz niz koristi se brojač i to na način sto se u registar $t1 smješta 1 instrukcijom *li* (load immediate). Zatim, instrukcijom *lw* (load word) u registar $v0 smješta se prvi element niza i time se za trenutni maksimum postavlja upravo taj element. Potom, adresa se uvećava za 4 kako bi se prešlo na naredni element.

```
Loop1:
  lw $t2,($t0)          # $t2 -> trenutni element niza
  bge $v0,$t2,Label1    # if ($v0 ≥ $t2) -> Label1
  add $v0,$t2,0         # $v0 -> $t2
Label1:
  add $t1,$t1,1         # $t1 = $t1 + 1
  add $t0,$t0,4         # $t0 = $t0 + 4
  blt $t1,$a1,Loop1     # if ($t1 < $a1) -> Loop1
```

Ovaj dio kôda predstavlja petlju kojom se prolazi kroz niz od drugog do posljednjeg elementa i poredi svaki sa trenutno najvećim. Ukoliko je element koji se nalazi u registru $t2 veći od trenutno najvećeg ($v0), u registar $v0 se upisuje taj

element ($t2) i on postaje novi maksimum. Ipak, ukoliko ovaj uslov nije zadovoljen, „skače“ se na *Label1* naredbom za uslovni skok *bge* (be greater or equal than).

Ispod označene labele (*Label1*) nalaze se naredbe vezane za prelazak na naredni element. Naredbom *add* brojač se prvo uvećava za 1, a zatim se pomoću iste naredbe adresa uvećava za 4 i prelazi se na naredni element niza. Na kraju petlje se provjerava da li je brojač ($t1) i dalje manji od dužine niza ($a1). Ako jeste, "skače" se na labelu *Loop1,* a u suprotnom izvršavaju se naredne linije kôda. Ovo se postiže branch instrukcijom *blt* (be less than).

```
jr $ra        # povratak iz procedure u glavni program
```

Kao što je i u prethodnom dijelu rada objašnjeno, instrukcija *jr* (jump register) se koristi za povratak u glavni program, odnosno na sljedeću liniju glavnog programa nakon poziva procedure, čija se adresa nalazi u registru *$ra*.

4.4 Kompletan kôd programa

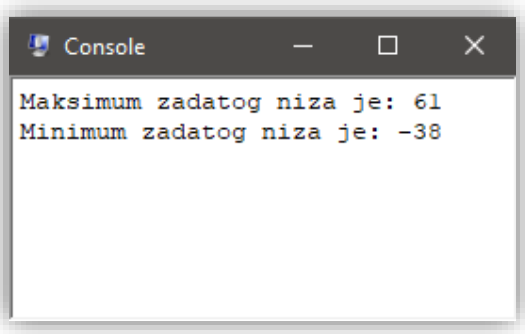
```
# Deklaracija promjenljivih
.data
array: .word 13, 34, 16, 61, 28, -24, 58, 11, 26, 41, 0, 7, -38, 12, 13
min: .word 0
max: .word 0
duzina: .word 15
stringmax: .asciiz "Maksimum zadatog niza je: "
stringmin: .asciiz "Minimum zadatog niza je: "
novired: .asciiz "\n"

#Text/kod zadatka
.text
.globl main
main:
la $a0, array       # u $a0 se učitava adresa prvog elementa niza
lw $a1, duzina      # u $a1 učitava se dužina niza
jal minimum         # poziva se proceduru minimum
sw $v0, min         # rezultat procedure smješta se u promjenljivu min
jal maximum         # poziva se procedura maximum
sw $v0, max         # rezultat procedure smješta se u promjenljivu max
```

```
li $v0, 4          # call code, print string - stringmax
la $a0, stringmax
syscall            #system call
li $v0, 1          # call code, print integer - max
lw $a0, max
syscall            #system call
li $v0, 4          # call code, print string - novired
la $a0, novired
syscall            #system call
li $v0, 4          # call code, print string - stringmin
la $a0, stringmin
syscall            #system call
li $v0, 1          # call code, za print integer - min
lw $a0, min
syscall            #system call

# Kraj programa
li $v0, 10         #terminate
syscall            #system call
.end main

minimum:
add $t0,$a0,0      # u $t0 smješta se početna adresa niza
li $t1,1           # u $t1 smješta se 1 i koristi se kao brojač
lw $v0,($t0)       # u $v0 učitava se prvi element niza (trenutno najmanji)
add $t0,$t0,4      # $t0 se uvećava za 4 (adresa narednog elementa)
Loop:
lw $t2,($t0)       # u $t2 se smješta element niza koji se poredi sa $v0
ble $v0,$t2,Label  # if ($v0 ≤ $t2) -> Label
add $v0,$t2,0      # $v0 -> $t2
Label:
add $t1,$t1,1      # brojač++
add $t0,$t0,4      # $t0 = $t0 + 4
blt $t1,$a1,Loop   # if ($t1 < $a1) -> Loop

jr $ra             # povratak iz procedure u glavni program

maximum:
add $t0,$a0,0      # u $t0 se smješta početna adresa niza
li $t1,1           # u $t1 smješta se 1 i koristi se kao brojač
lw $v0,($t0)       # u $v0 učitava se prvi element niza (trenutno najveci)
add $t0,$t0,4      # $t0 se uvećava za 4 (adresa narednog elementa)
```

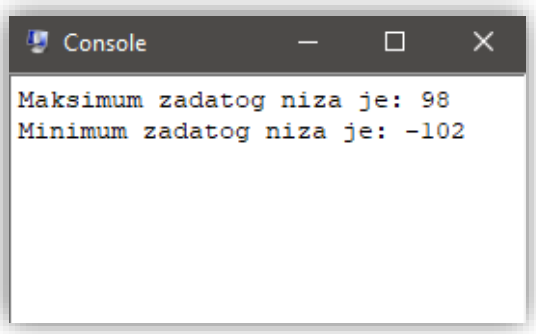
```
Loop1:
lw $t2,($t0)       # u $t2 se smješta element niza koji se poredi sa $v0
bge $v0,$t2,Label1 # if ($v0 ≥ $t2) -> Label1
add $v0,$t2,0      # $v0 -> $t2
Label1:
add $t1,$t1,1      # brojač++
add $t0,$t0,4      # $t0 = $t0 + 4
blt $t1,$a1,Loop1  # if ($t1 < $a1) -> Loop1

jr $ra             # povratak iz procedure u glavni program
```

4.5 Prikaz konzole nakon izvršenja programa

Izgled konzole nakon izvršenja prikazanog kôda u QtSpim simulatoru, prikazan je na slici 4.



Slika 4.



Slika 5.

Kao što je traženo u samom zadatku, program radi za bilo koji niz od 15 elemenata. U tu svrhu, izvršen je kod sa još jednim nizom [19, 7, 97, -39, 17, 9, 93, -102, 28, 4, 98, -7, 0, -56, 45], a na slici 5 prikazan je izgled konzole nakon njegovog izvršenja.