

Universiteti i Prishtinës
Fakulteti i Inxhinierise Elektrike dhe Kompjuterike



Detyra: C++ to MIPS Assembly (Opsioni A)

Studenti: **Ylber Gashi**

ID: **180714100025**

Lënda: *Arkitektura e kompjuterëve*

Ligjerata: *Prof. Dr. Valon Raca*

Ushtrime: *MSc. Vlera Alimehaj*

1. Hyrje

Opsioni A

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

int populloVektorin(int a[])
{
    int n;
    cout << "Jep numrin e anetareve te vektorit (max. 5):"; cin >> n;

    cout << "\nShtyp elementet nje nga nje:\n";
    for (int i = 1; i <= n; i++) { cin >> a[i]; }
    return n;
}

void unazaVlerave(int p, int n, int &min, int a[], int &loc)
{
    for (int k = p + 1; k <= n; k++)
    {
        if (min > a[k])
        {
            min = a[k];
            loc = k;
        }
    }
}

void unazaKalimit(int a[], int n)
{
    int min, loc, tmp;
    for (int p = 1; p <= n - 1; p++) // Loop for Pass
    {
        min = a[p]; // Element Selection
        loc = p;
        unazaVlerave(p, n, min, a, loc);
        tmp = a[p];
        a[p] = a[loc];
        a[loc] = tmp;
    }
    cout << "\nVlerat e vektorit ne fund: \n";
    for (int i = 1; i <= n; i++) {
        cout << a[i] << endl;
    }
}

int main()
{
    int a[5], n = 0;
    n = populloVektorin(a);
    unazaKalimit(a, n);
}
```

Ky program i shkruar në C++ është një program me anë të cilit i sortojmë (nga më i vogli tek më i madhi) dhe i printojmë vlerat e vektorit të sortuar, vlera të cilat jepen nga përdoruesi si input. Dhënia e elementeve të vektorit bëhet me anë të funksionit **populloVektorin**, ku **n** paraqet max të elementeve që do t'i përmbajë vektori, ndërsa sortimi bëhet me anë të dy funksioneve: **unazaKalimit**, ku me ndihmën e një for loop iterohet nëpër elementet e vektorit dhe brenda saj thirret funksioni **unazaVlerave** (me 5 parametra), i cili na jep vlerën minimale të vektorit (**min**) për iterimin aktual në loop-ën amë dhe adresen (lokacionin) e atij anëtarit (**loc**). Ky lloj sortimi njihet si **“selection sort”**. Printimi i vektorit të sortuar është pjese e funksionit **unazaKalimit**.

2. Realizimi i kodit në MIPS

```
.data
vektori: .space 20
n: .word 0
jepNrAnt: .asciiz "Jep numrin e anetareve te vektorit (max. 5): "
shtypElem: .asciiz "Shtyp elementet nje nga nje: \n"
vleratEvekt: .asciiz "\nVlerat e vektorit jane: \n"
endl: .asciiz "\n"

.text
main:
    la $a0, vektori
    la $a1, n
    jal populloVektorin          # thirret funksioni POPULLLO VEKTORIN
    la $a0, vektori              # vektorin e dergojme si parameter
    lw $a1, n                    # n si parametri i dyte
    jal unazaKalimit             # thirret funksioni UNAZA E KALIMIT
    li $v0, 10                  # mbyllet programi
    syscall

#-----POPULLLO VEKTORIN-----
populloVektorin:
    move $s0, $a0               # vektorin po e ruajme nga parametri ne regjistrin $s0
    move $s1, $a1                # n-in po e ruajme ne regjistrin $s1
    li $v0, 4
    la $a0, jepNrAnt             # printohet fjalia jepNrAnt
    syscall
    li $v0, 5                    # numri i anetareve i dhene nga perdoruesi
    syscall
    sw $v0, 0($s1)               # n e dhene nga input e ruajme te adresa e n pra $a1
    move $s1, $v0                # n e dhene nga inputi e vendosim edhe ne $s1
    li $v0, 4
    la $a0, endl                 # rresht i ri
    syscall
    li $v0, 4
    la $a0, shtypElem            # printohet shtypElem
    syscall

    li $t1, 1                    # int i = 1
    ruaj:
        bgt $t1, $s1, gotoMain   # $t1 eshte i, ndersa $s1 eshte n
        li $v0, 5                # inputi perdoruesit si numer
        syscall
        sw $v0, 0($s0)           # po e ruajme ne vektor vleren e dhene si input
        addi $s0, $s0, 4         # e inkrementojmë per 4 adresen ku do ta shkruajme
        numrin e radhes ne vektor
        addi $t1, $t1, 1         # i++
        j ruaj

    gotoMain:
        jr $ra

#-----UNAZA E KALIMIT-----
unazaKalimit:
    addi $sp, $sp, -4            # po e lirojme nje hapësirë 32-biteshe ne stack
    pointer sepse kemi me e ruajt adresen kthyesë $ra qe eshte 32bit
    sw $ra, 0($sp)               # adresen kthyesë te ketij funksioni e ruajme ne stack

    move $s0, $a0                # vektori
    move $s1, $a1                # vlera e n
    addi $s1, $s1, -1
```

```

li $t0, 0                                # t0 eshte p e for loop-it
loop:                                    # LOOP FOR PASS
    beq $t0, $s1, endloop

    mul $t3, $t0, 4
    add $t5, $s0, $t3                    # adresen baze te vektorit po e inkrementoyme per
vleren e $t4(p * 4), sepse nje numer i plote ne vektor eshte 4bytes, prandaj edhe
indeksi duhet te jete shumefish i 4

    lw $t1, 0($t5)                       # $t1 --> min = a[p]. $t5 paraqet adresen e vleres
qe kemi me e marr nga vektori
    move $s6, $t1                         # a[p]
    move $a2, $t0                         # e dergojme p si parameter
    move $t2, $t0                         # loc = p

    jal unazavlerave

    move $t6, $s6                         # temp = a[p]

    mul $t4, $t0, 4
    add $t5, $s0, $t4                    # e inkrementoyme adresen baze te vektorit per
vleren e $t4 e cila eshte p*4

    sw $t1, 0($t5)                       # a[p] = a[loc] ose thene ndryshe ne a[p] po e
ruajme minimumin e rezultuar nga funksioni unazaVlerave, dmth $t1 eshte a[loc] ne kete
pike
    mul $t3, $t2, 4
    add $t5, $s0, $t3                    # loc si shumefish i 4
    sw $t6, 0($t5)                       # a[loc] = temp

    addi $t0, $t0, 1
    j loop

endloop:
    lw $t0, n
    li $v0, 4
    la $a0, vleratEvekt                  # text print
    Syscall

    li $t1, 1
printoVektorin:
    bgt $t1, $t0, backHOME

    li $v0, 1
    lw $a0, 0($s0)
    syscall

    li $v0, 4
    la $a0, endl
    syscall

    addi $s0, $s0, 4
    addi $t1, $t1, 1
    j printoVektorin

backHOME:
    lw $ra, 0($sp)                       # po e rikthejme adresen kthyes te ketij funksioni
nga stack-u
    addi $sp, $sp, 4                     # po e lirojme ate pjese te stack-ut
    jr $ra

```

```

#-----UNAZA E VLERAVE-----
unazavlerave:
    move $s2, $a0          # adresa e array
    move $s3, $a1          # n
    move $s4, $a2          # p
    # $t1 eshte min
    # $t2 eshte loc

    addi $t7, $s4, 1        # k = p+1
loop2:
    beq $t7, $s3, endloop2  # $t7 eshte k, $s3 eshte n

    mul $t3, $t7, 4         # k * 4
    add $t5, $s2, $t3       # inkrementohet adresa baze e vektorit dhe kjo
                             # velere ($t5) paraqet adresen e sakte per elementin qe duam t'i qasemi

    lw $t6, 0($t5)         # a[k]

    blt $t1, $t6, goto      # if min > a[k]
    move $t1, $t6           # min = a[k]
    move $t2, $t7           # loc = k
goto:
    addi $t7, $t7, 1
    j loop2
endloop2:
    jr $ra

```

3. Testimet me QtSpim

Console

Jep numrin e anetareve te vektorit (max. 5): 5

Shtyp elementet nje nga nje:

8
7
1
4
2

Vlerat e vektorit jane:

1
2
4
7
8

Console

Jep numrin e anetareve te vektorit (max. 5): 4

Shtyp elementet nje nga nje:

7
11
5
3

Vlerat e vektorit jane:

3
5
7
11

Këtu shihet testimi i programit në QtSpim kur n-in e japim 5. Gjithçka shkon ashtu si duhet dhe në fund paraqiten vlerat e vektorit, të sortuara.

Ndërsa këtu është paraqitur rasti kur n-i zgjedhet të jetë 4. Pra shihet qartë që edhe me vlera më të vogla se 5 (max) programi funksionon pa asnjë problem.

```
Console
Jep numrin e anetareve te vektorit (max. 5): 7

Shtyp elementet nje nga nje:
5
3
1
4
2
3
6

Vlerat e vektorit jane:
1
3
5
|
```

Në rast se n -in e japim më shumë se 5, atëherë na shfaqen probleme në program. Kjo pasi që në .data vektorit i është caktuar hapësira 20 bytes (sepse 5 nr. të plotë, nga 4 bytes i bie 20 bytes).

Gjatë provave të shumta kam hasur që ky problem nuk sillet njëjtë për vlera të ndryshme të n -it (është fjala për më tepër se 5) ose thënë ndryshe nuk kam mundur ta identifikoj qartë ndonjë “pattern” të veçantë të programit në këto raste.

4. Përfundimi

Me këtë detyrë, përveç faktit që jam familjarizuar me MIPS Assembly, kam fituar një përshtypje mjaft të mirë se sa abstrakte janë gjuhët e nivelit të lartë në krahasim me këtë nivel. Po ashtu, kam vërejtur se shumë veprime mund të kryhen në disa mënyra, e sa e rëndësishme është zgjedhja e asaj më optimale.

Është e qartë që kodi që kam shkruar unë ka ende hapësirë për optimizim të mëtutjeshëm, mirëpo, meqenëse optimizimi nuk ka qene pjesë e domosdoshme e projektit, e kam parë të arsyeshme ta lë me kaq.