

# **Universiteti i Prishtinës**

## **Fakulteti i Inxhinierisë Elektrike dhe Kompjuterike**

### **Lënda: Arkitektura e Kompjutereve – 2020**



**Profesor I lendes: Valon Raca**

**Studenti: Lirim Beka**

**Data: 24/05/2020**

**ID: 180714100157**

# Permbajtja

- 1.Hyrje
- 2.Realizimi I kodit ne MIPS
- 3.Testimet me QtSpim
- 4.Perfundimi

## 1. Hyrje

```
#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);
}
```

Qellimi I ketij projekti ishte konvertimi I algoritmit te implementuar ne C++ ne gjuhen assembly MIPS.

Ky projekt kishte 3 programe si mundesi konvertimi, por une po caktoj detyren a) per raport.

Kodi ne C++ mundeson sortimin e nje array te jepur nga perdoruesi si input permes algoritmit sortues Bubble Sort.

Programi ka gjithseje tri funksioni qe mund te implementohen ne MIPS si dhe disa nga argumentet jane 'passed by reference'.

Funksioni PopulloVargun mundeson mbushjen e arrays si dhe kthen n apo gjatesine e array te jepur nga perdoruesi, pastaj funksioni UnazaKalimit mundeson qe te kontrollohet qdo integjer I dhene ne array, dhe me pas rishikimi dhe sortimi I array mundesohet nga funksioni UnazaVlerave, ku kjo therritet tek unaza kryesore e funksionit unazaKalimit dhe nese ka elemente me te madhe para elementeve me te vogla atehere behet sortimi permes snipetes `"tmp=a[p]; a[p] =a[loc]; a[loc] = tmp;"`

## 2. Realizimi I kodit ne mips

Realizimi I kodit apo konvertimi I tij nga C++ ne Mips ishte I qarte dhe jo shume I komplikuar perpos te pjesa e snipet kodit qe e ceka me lart te pika 1.

Ne ate pjese kam hasur probleme per konvertimin e sakte te kodit pasi qe kam verejtur qe nje komande e thjeshte MOVE qe mundeson ti kopjojme te dhenat e nje regjistri ne nje regjister te ri nuk e zgjedhi problemin.

```

unazaKalimit:

    li $a1, 1          # p = 1
    li $a2, 0          # min
    li $a3, 0          # loc
    li $t4, 0          # tmp

    while:
        sll $s3, $a1, 2      # shift characters to the left
        lw $t5, array($s3)   # a[p]

        move $a2, $t5        # min = a[p]
        move $a3, $a1        # loc = p

# In order to call a procedure inside a procedure(we need to put return value in stack) #
    add $sp, $sp, -4
    sw $ra, 0($sp)
    jal unazaVlerave
    lw $ra, 0($sp)
    addi $sp, $sp, 4

#####

    sll $s4, $a1, 2
    lw $t4, array($s4)      # temp = a[p]

    sll $s5, $a3, 2         # shift character to the left
    lw $t7, array($s5)      # a[loc]

    sw $t7, array($s4)      # a[p] = a[loc]
    sw $t4, array($s5)      # a[loc] = tmp

    addi $a1, $a1, 1        # p++
    blt $a1, $v1, while     # while p <= n-1 back to while
end:

```

Qe Unaza kryesore te funksioni UnazaKalimit te mundesoj sortimin e numrave, duhet therritur ne unaze funksioni unazaVlerave. Nje gje e tille u mundesua permes komandes **jal** (jump and link) por jo vetem. Ne MIPS ne rastet kur kemi te bejme me funksione te lidhura apo nested procedure, qe nenkupton therritjen e nje funksioni mbrenda nje funksioni atehere ne duhet ta hapim stackun dhe ta ruajme return address \$ra ne memorie (permes komandes **sw**), para thirrjes se atij funksioni dhe me pas menjehere pas thirrjes se pari ngarkojme return adresen \$ra ne regjistrin **\$sp**(regjistri per stack pointer) dhe me pas mbylлим stackin.

Ne kete menyre eshte mundesuar ndryshimi I variables loc dhe min ne C++ , qe jane te ruajtur ne regjistrat \$a3 dhe \$a2 perkatesisht ne MIPS.

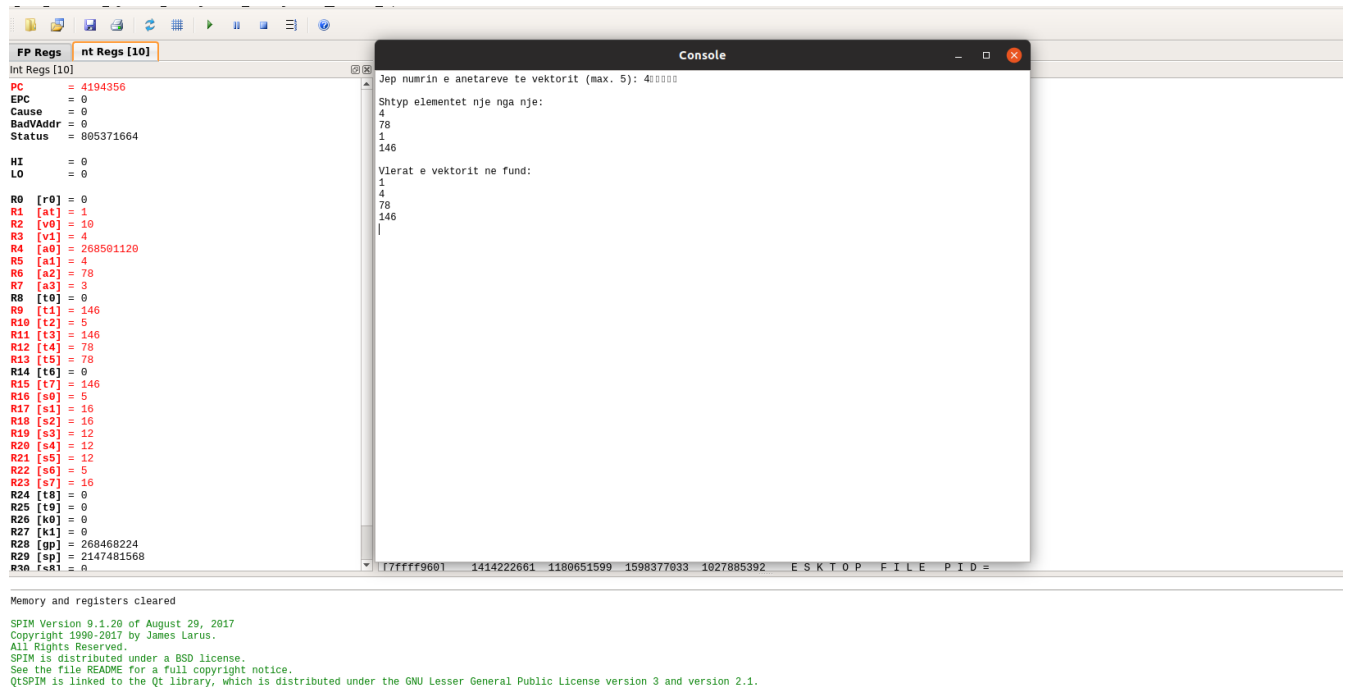
Hapi I fundit per sortimin e array eshte bere permes pjeses sic shihet me poshte ne foto. Se pari kemi inicializuar prape vargun a[p] ne menyre qe ngarkojme regjistrin \$t4 me te dhenet e ketij vargu. Pastaj eshte inicilizuar vargu a[loc] dhe me pas eshte ruajtur ne memorie regjistri \$t7 qe perfaqeson a[loc], me te dhenat e vargut a[p] dhe e ngjashme eshte bere edhe per regjistrin \$t4 qe perfaqeson tmp ku eshte ruajtur me te dhenat e vargut a[p]. Pikerisht ketu ishte edhe problemi im ku nuk kam mundur ta beje nje gje te tille permes komandes **MOVE** pasi qe sipas mendimit tim nuk ishte e mundur update e tmp.

Gjithashtu ta cek se ne rastin e incializimit te vektorit kam perdorur komanden sll qe mundeson ti lejon hapesire prej 4 bitash nje integjeri dhe **sll** (shift by left) me fjale te tjera nenkupton shumezimi I atij numri me vleren  $2^n$  e ne kete rast n eshte 2.

Pastaj eshte inkrementuar unaza, ndersa komanda **blt** (branch less than) nenkupton qe perdirdsa regjistri \$a1 eshte me I vogel se regjistri \$v1 kalo ne labelin while e qe ne kete rast eshte vete unaza.

### 3. Testimet me QtSpim

Testimi ne programin ne MIPS permes Spim ka shkuar mire perpos ne rastin se kur **n** ka kaluar hapesiren e lejuar prej 20 bitash qe perkthehet ne 5 karektere , programi ka funksionuar prape pa teleshe. Sidoqofte nje gje te tille e kam verejtur edhe te programi ne C++.



```
FP Regs  nt Regs [10]
Int Regs [10]
PC = 4194356
EPC = 0
Cause = 0
BadVAddr = 0
Status = 805371664
HI = 0
LO = 0
R0 [r0] = 0
R1 [at] = 1
R2 [v0] = 10
R3 [v1] = 4
R4 [a0] = 268501120
R5 [a1] = 4
R6 [a2] = 78
R7 [a3] = 3
R8 [t0] = 0
R9 [t1] = 146
R10 [t2] = 5
R11 [t3] = 146
R12 [t4] = 78
R13 [t5] = 78
R14 [t6] = 0
R15 [t7] = 146
R16 [s0] = 5
R17 [s1] = 16
R18 [s2] = 16
R19 [s3] = 12
R20 [s4] = 12
R21 [s5] = 12
R22 [s6] = 5
R23 [s7] = 16
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 268468224
R29 [sp] = 2147481568
R30 [s8] = 0

Jep numrin e anetareve te vektorit (max. 5): 40000
Shtyp elementet nje nga nje:
4
78
1
146
Vlerat e vektorit ne fund:
1
4
78
146
|

Memory and registers cleared
SPIM Version 0.1.20 of August 29, 2017
Copyright 1990-2017 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.
```

```
Console
Jep numrin e anetareve te vektorit (max. 5): 8
Shtyp elementet nje nga nje:
64
31
45
78
89
5
1
0
Vlerat e vektorit ne fund:
0
1
5
31
45
64
78
89
|
```

Sic shihet ne foton e pare eshte paraqitur sortimi dhe vlerat e regjistrave ne anen e djathte. Sa l perket sortimit gjithcka ka shkuar ne rregull, ndersa ndoshta perdorimi i regjistrave nuk ka qene 100% i sakte pasi qe me mungon edhe pervoja ne MIPS ne kete aspekt.

Ndersa ne foton e dyte shihen se sortimi prape ka shkuar ne rregull por qe programi nuk ka treguar diqka te pazaktone pasi qe eshte kaluar hapësira prej 20 bitash.

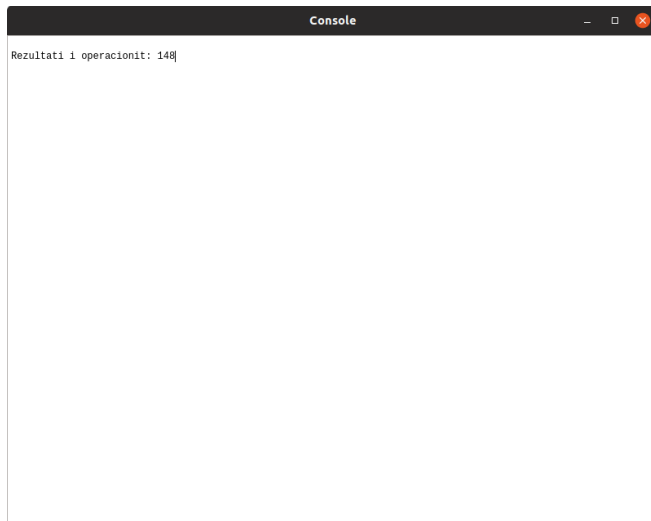
## 4. Perfundimi

Sipas mendimit tim njohuria se si funksionon logjika e Assembly eshte shume e nevojshme per shumicen e programereve apo inxhinierëve pasi qe kur kupton Assembly, ta kuptosh se si CPU punon behet shume me e qarte dhe kjo rezulton ne kod me te optimizuar ne gjuhet programuese me te larta siq jane C++, Java etj.

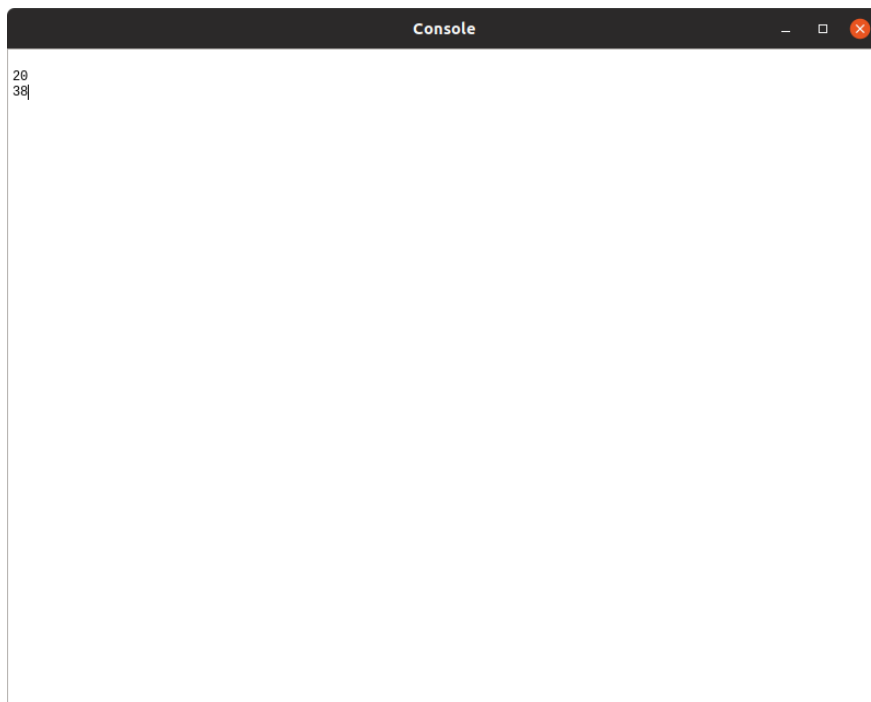
Arsye tjeter eshte sepse Assembly eshte ende gjuha me e perdorur sa l perket sistemeve **embedded** apo sistemeve nderlidhese te hardwerit me software sic jane driverat, compilerat etj.

Prandaj ne fund kam dal ne konkluzion qe nje projekte I ketij qellimi, eshte I domodoshem per mua si inxhinier softuerik I ardhshem!

P.S Na pamundesi te krijimit te raporteve te vecanta per DetyrenB dhe C po ndaj ketu dy snapshot nga QtSpim si testim I programit...



Detyra B)



Detyra C)