Universiteti i Prishtinës “Hasan Prishtina”
Fakulteti i Inxhinierisë Elektrike dhe Kompjuterike

Raport i detyës

Lënda: Arkitektura e Kompjuterëve

Student	Gjon Hajdari
ID	210756100052
Profesor	Valon Raça

Prishtinë 2022

Përmbajtja

1. Hyrje

Në këtë pjesë janë paraqitur me foto kodet në C++ për dy nga detyrat e dhëna. Opsionet janë marrë për detyrat nën **A** dhe nën **B**.

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

int factorial(int, int);

int main()
{
    int n, result, a = 1;

    cout << "Enter a non-negative number: ";
    cin >> n;

    result = factorial(a, n);
    cout << "Factorial of " << n << " = " << result;
    return 0;
}

int factorial(int a, int n)
{
    if (n > a) {
        return a * n * factorial(n - 1);
    } else {
        return a;
    }
}
```

Figura 1: Kodi në C++ për detyrën A.

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

int main() {
    int numbers[5];
    cout << "Enter 5 numbers: " << endl;

    // store input from user to array
    for (int i = 0; i < 5; ++i) {
        cin >> numbers[i];
    }

    cout << "The numbers are: ";

    // print array elements
    for (int n = 0; n < 5; ++n) {
        cout << numbers[n] << " ";
    }
    return 0;
}
```

Figura 2: Kodi në C++ për detyrën B.

2. Realizimi i kodit në MIPS

Në projekt janë dorzuar dy variante të detyrës A. Kodi i shfaqur më poshtë tek **Detyra A është i variantit të dytë.*

2.1. Detyra A

Kërkesa e detyrës A ka qenë që të gjindet faktorieli i një numri jo-negativ i shtypur nga përdoruesi. Tekstet që do të shfaqen në console janë ruajtur në segmentin **.data**.

```
.data
     question:  .asciiz "Enter a non-negative number: "
     result:    .asciiz "Factorial of "
     equals:    .asciiz " = "
```

Si fillim është shfaqur kërkesa dhe inputi i përdoruesit është ruajtur në regjistrin **$s0**, rrespektivisht në atë **$t2** pasi që me anë të tij i bëjmë kalkulimet në vazhdim.

```
li $v0, 4
la $a0, question
syscall              # ask user for the factorial base

li $v0, 5
syscall
move $s0, $v0
move $t2, $s0        # set user input as factorial base
```

Në vazhdim hyjmë në kodin pas tiketës **factorial** e cila shërben si pikë kthyese për rekurzionin e kodit. Brenda saj fillimisht kontrollojmë nëse **$t2** (n) është më e vogel se 1, me ç'rast kalkulimi i faktorielit përfundon dhe vlerat printohen.

```
factorial:
    slt $t0, $t2, $t1
    bne $t0, $zero, print       # if n < a goto print
```

Nësë n është më e madhe së 1 atëherë rezultati i ruajtur në regjistrin **$s1**, shumëzohet me vlerën a (regjistri **$t1**) dhe me numrin e rradhës në faktoriel. Kjo pastaj ruhet tek rezultati.

```
mult $s1, $t1          # multiply with a
mflo $s1               # save as new result

mult $s1, $t2          # multiply with the next number in line
mflo $s1               # save as new result
```

Pas shumëzimit të vlerave, dekrementojmë vlerën n për 1 dhe therrasim funksionin përsëri deri sa të plotësohet kushti i përmendur më lartë.

```
addi $t2, $t2, -1           # decrement n
j factorial                 # call function again
```

Si përfundim nëse kushti i faktorielit plotësohet, pra n është më i vogël së 1 atëherë ai na dërgon tek pjesa e kodit me tiketën **print** ku printohen vlerat dhe përfundon programi.

```
print:
     li $v0, 4
     la $a0, result
     syscall               # print "Factorial of "

     li $v0, 1
     move $a0, $s0
     syscall               # print base number

     li $v0, 4
     la $a0, equals
     syscall               # print " = "

     li $v0, 1
     move $a0, $s1
     syscall               # print factorial result
```

2.2. Detyra B

Detyra B kërkon që të shtypen nga përdoruesi 5 numra, të ruhen dhe pastaj të shfaqen në ekran në atë renditje. Si tek detyra A, ashtu edhe edhe në B, tekstet për t'u shfaqur janë ruajtur në segmentin **.data**.

```
.data
     enterMessage:  .asciiz "Enter 5 numbers: "
     printMessage:  .asciiz "The numbers are: "
     space:         .asciiz " "
```

Pas shfaqjes së mesazhit për të shtypur 5 vlera hyjmë në kodin pas tiketës **store** që shërben si pikë kthyese për rekurzion. Në fillim kontrollojmë nëse **$s0** (numri i rekurzioneve) është më i madh se 5 me ç'rast shkojmë në pjesën e printimit të numrave.

```
store:
     slti $t0, $s0, 5
     beq $t0, $zero, numbers      # if n > 5 print items
```

Nësë numri i rekurzioneve është më i vogel atëherë e inkrementojmë dhe pastaj lëvizim **$sp** (stack pointerin) për 4 adresa memorike lartë, duke krijuar vend për një numër të plotë 4 bajtësh.

```
addi $s0, $s0, 1        # increment n
addi $sp, $sp, -4       # add room for an integer in stack
```

Më pas lexojmë vlerën e shtypur nga përdoruesi dhe e ruajmë në regjistrin **$t1**. Këtë vlerë e vendosim në stack dhe kthehemi përsëri tek tiketa **store**.

```
li $v0, 5
syscall
move $t1, $v0               # read integer and store in $t1

sw $t1, 0($sp)              # add integer to stack
j store                     # call function again
```

Pasi të vendosen të gjitha vlerat në stack, e kthejmë numrin e rekurzioneve në gjendje fillestare (kjo na duhet për të filluar rekurzionet e shtypjes së vlerave) dhe printojmë mesazhin *"The numbers are: "*.

```
numbers:
     move $s0, $zero            # reset n

     li $v0, 4
     la $a0, printMessage
     syscall                    # print "The numbers are: "
```

Ngjajshëm sikur të leximi i vlerave përdorim të njejtën logjikë tek printimi. Ekzekutojmë kodin pas tiketës **printLoop** për aq herë sa ka numra në varg, çdo herë duke inkrementuar numrin e rekurzioneve. Vlerën e marrim nga stack-u, e largojmë duke rritur vlerën e stack pointerit për 4 adresa memorike (një integer), e printojmë dhe pastaj kthehemi tek tiketa **printLoop**. Kur numri i rekurzioneve kalon 5 atëherë programi përfundon ekzekutimin.

```
printLoop:
     slti $t0, $s0, 5           # #
     beq $t0, $zero, exit       # # if n < 1 goto exit
     addi $s0, $s0, 1           # decrement n
     lw $s1, 0($sp)             # get integer from stack
     addi $sp, $sp, 4           # pop item out of stack

     li $v0, 1
     move $a0, $s1
     syscall                    # print integer

     li $v0, 4
     la $a0, space
     syscall                    # print space between items

     j printLoop                # call function again
```

Për shkak të natyrës së renditjes First In Last Out të stack-ut numrat printohen me renditje të kundërt të shtypjes.

3. Testimet me QTSpim

Gjatë zgjidhjes së detyrave është përdorur programi **QTSpim** për të ekzekutuar kodin si dhe për të parë ecurinë e ekzekutimit të kodit dhe të sjelljes së regjistrave. Mesazhet e gabimeve sintaksore si dhe pamja e saktë e regjistrave dhe user stack-ut ka ndihmuar shumë në zgjidhjen e problemeve gjatë shkrimit dhe ekzekutimit të kodit.

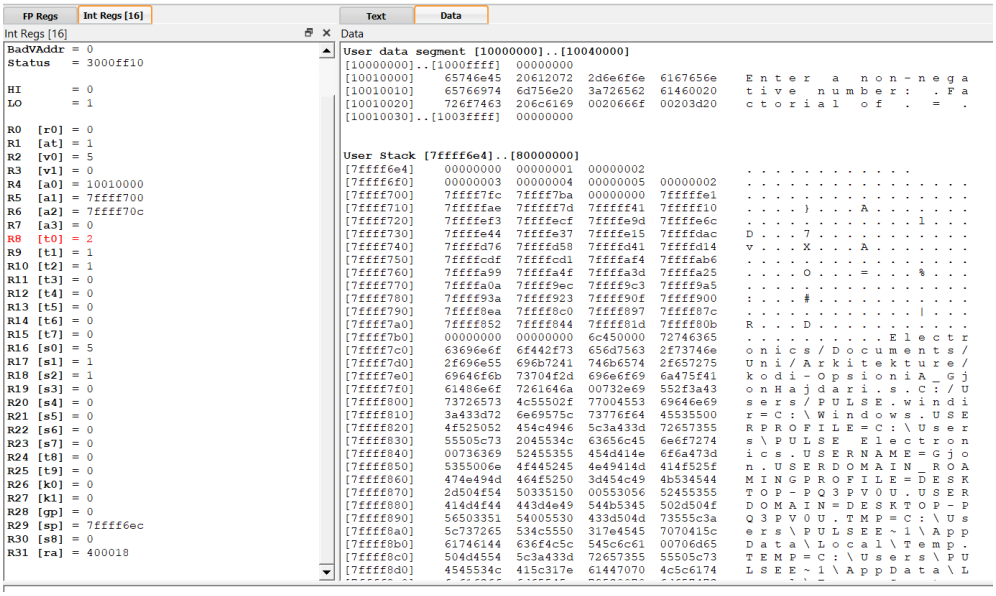

Figura 3: Pamja e të dhënave në QTSpim.

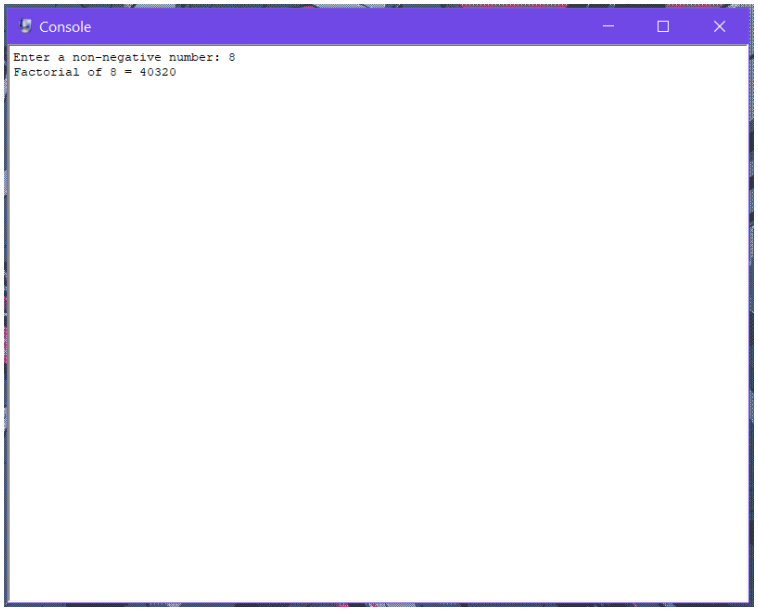

Figura 4: Pamja e console në QTSpim.