

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	3
2. Realizimi i kodit në MIPS	4
2.1. Detyra A	4
2.2. Detyra B	6
3. Testimet me QTSpim	8

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**.

```
#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.

```
#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: .ascii "Enter a non-negative number: "
result:    .ascii "Factorial of "
equals:    .ascii " = "
```

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 kthyesë për rekursionin e kodit. Brenda saj fillimisht kontrollojmë nëse **\$t2** (n) është më e vogël 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
```

Nëse n është më e madhe se 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 therrsim 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 se 1 atëherë ai na dërgon tek pjesa e kodit me etiketën **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: .asciiiz "Enter 5 numbers: "
    printMessage: .asciiiz "The numbers are: "
    space:        .asciiiz " "
```

Pas shfaqjes së mesazhit për të shtypur 5 vlera hyjmë në kodin pas tiketës **store** që shërben si pikë kthyesë për rekursion. 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
```

Nëse numri i rekurzioneve është më i vogël 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.

FP Regs	Int Regs [16]	Text	Data
Int Regs [16]			
BadVAddr = 0		User data segment [10000000]..[10040000]	
Status = 3000fff10		[10000000]..[1000ffff] 00000000	
HI = 0		[10010000] 65746e45 20612072 2d6e6f6e 6167656e	Enter a non-nega
LO = 1		[10010010] 65766974 6d756e20 3a726562 61460020	tive number: . Fa
		[10010020] 726f7463 206c6169 0020666f 00203d20	ctorial of . = .
		[10010030]..[1003ffff] 00000000	
R0 [r0] = 0		User Stack [7ffff6e4]..[80000000]	
R1 [at] = 1		[7ffff6e4] 00000000 00000001 00000002	
R2 [v0] = 5		[7ffff6f0] 00000003 00000004 00000005 00000002	
R3 [v1] = 0		[7ffff700] 7ffff7fc 7ffff7ba 00000000 7fffffe1	
R4 [a0] = 10010000		[7ffff710] 7ffff7ae 7ffff7d7 7ffff741 7fffffe10	
R5 [a1] = 7ffff700c		[7ffff720] 7ffff7f3 7ffff7ec 7ffff69d 7ffff6c6	
R6 [a2] = 7ffff70c		[7ffff730] 7ffff7e4 7ffff637 7ffffe15 7ffffdac	
R7 [a3] = 0		[7ffff740] 7ffffd76 7ffffd58 7ffffd41 7ffffd14	
R8 [t0] = 2		[7ffff750] 7ffffcdf 7ffffcd1 7ffffaf4 7ffffab6	
R9 [t1] = 1		[7ffff760] 7ffffa99 7ffffa4f 7ffffa3d 7ffffa25	
R10 [t2] = 1		[7ffff770] 7ffffa0a 7ffff9ec 7ffff9c3 7ffff9a5	
R11 [t3] = 0		[7ffff780] 7ffff93a 7ffff923 7ffff90f 7ffff900	
R12 [t4] = 0		[7ffff790] 7ffff90a 7ffff8c0 7ffff897 7ffff87c	
R13 [t5] = 0		[7ffff7a0] 7ffff852 7ffff844 7ffff81d 7ffff80b	
R14 [t6] = 0		[7ffff7b0] 00000000 00000000 6c450000 72746365	
R15 [t7] = 0		[7ffff7c0] 63696a6f 61442f73 656d7563 2f73746e	
R16 [s0] = 5		[7ffff7d0] 2f696e55 696b7241 746b6574 2f657275	
R17 [s1] = 1		[7ffff7e0] 69646f6b 73704f2d 696e6f69 6a475f41	
R18 [s2] = 1		[7ffff7f0] 61486e6f 7261646a 00732e69 552f3a43	
R19 [s3] = 0		[7ffff800] 73726573 4c55502f 77004553 69646e69	
R20 [s4] = 0		[7ffff810] 3a433d72 6e69575c 73776564 45535500	
R21 [s5] = 0		[7ffff820] 4f525052 454c4946 5c3a433d 72657355	
R22 [s6] = 0		[7ffff830] 55505c73 2045534c 63656c45 6e6f7274	
R23 [s7] = 0		[7ffff840] 00736369 52455355 454d414e 6f6a473d	
R24 [t8] = 0		[7ffff850] 5355006e 4f445245 4e49414d 414f525f	
R25 [t9] = 0		[7ffff860] 474e494d 464f5250 3d454c49 4b534544	
R26 [k0] = 0		[7ffff870] 2d504f54 50335150 00553056 52455355	
R27 [k1] = 0		[7ffff880] 414d4f44 443d4e49 544b5345 502d504f	
R28 [gp] = 0		[7ffff890] 56503351 54005530 433d504d 73555c3a	
R29 [sp] = 7ffff6ec		[7ffff8a0] 5c737265 534c5550 317e4545 7070415c	
R30 [s8] = 0		[7ffff8b0] 61746144 636f4c5c 545c6c61 00706365	
R31 [ra] = 400018		[7ffff8c0] 504d4554 5c3a433d 72657355 55505c73	
		[7ffff8d0] 4545534c 415c317e 61447070 4c5c6174	

Figura 3: Pamja e të dhënave në QTSpim.

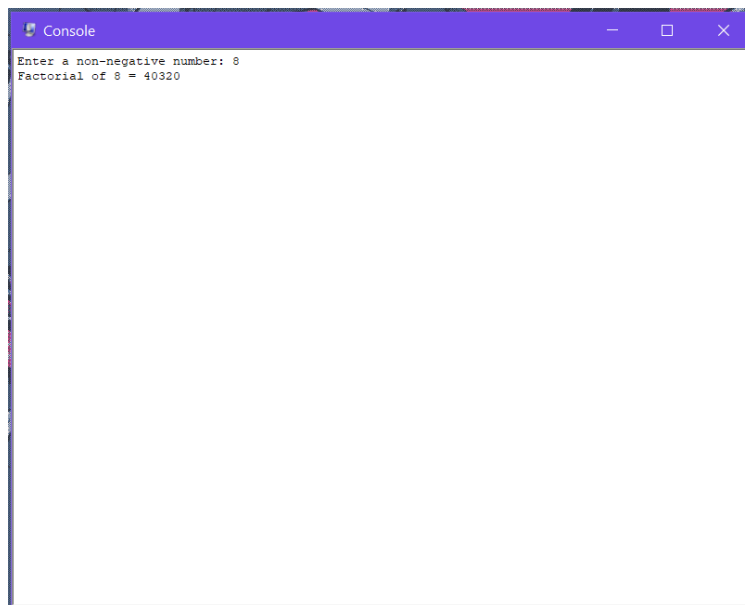


Figura 4: Pamja e console në QTSpim.