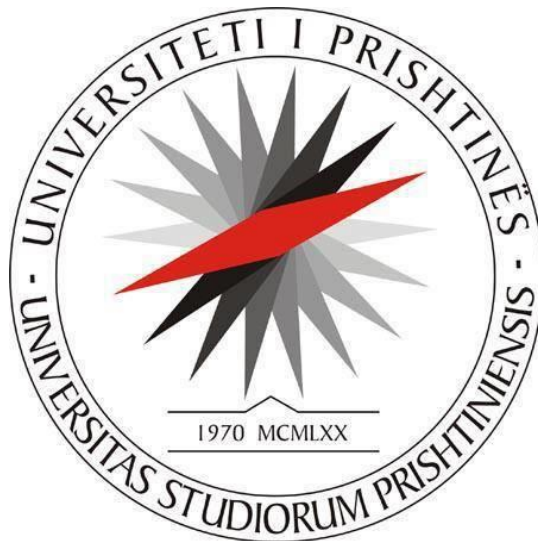


UNIVERSITETI I PRISHTINES

Fakulteti Inxhinierisë Elektrike dhe Kompjuterike



Lënda:Arkitektura e Kompjuterëve

Raporti:Detyra A

Profesori i Lëndës:Valon Raca

Studenti:Valdrin Ejupi

Id:190714100121

Përmbajtja:

Kodi ne C++	3
Realizimi i kodit ne Mips:	4
Testimet në QtSpim:	5

Kodi ne C++

Opsioni A: Ky opsion ju ofron mundësinë që të notoheni maksimalisht 10%

Të shkruhet në MIPS assembly code kodi i mëposhtëm në C++:

```
// A C++ program to demonstrate working of
// recursion
#include <bits/stdc++.h>
using namespace std;

void printFun(int test)
{
    if (test < 1)
        return;
    else {
        cout << test << " ";
        printFun(test - 1); // statement 2
        cout << test << " ";
        return;
    }
}

// Driver Code
int main()
{
    int test = 4;
    printFun(test);
}
```

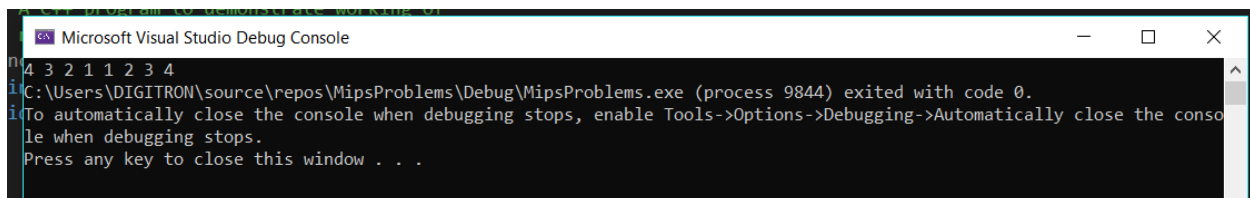
Kodi ne C++ shërben për të printuar numrat në mënyrë keq të renditur dhe mirë te renditur psh: për n=4 output-i do të ishte 4,3,2,1,1,2,3,4 pa përfshir 0

Meqëse ky program është program rekursiv për me kuptu ma lehtë e kam shtjelluar kështu problemin

Le të jetë: $f(x) = \text{printFun}(test)$, right cout $\Rightarrow \text{print}(x)$ dhe left cout $\text{print}(x)$ qka do të thotë kjo është se :

$\text{print}(4)$ do të ketë 2 cout që do paraqesin vlerën e test në console ideja është kështu:

				print(4)	f(4)	print(4)			
			print(4)	print(3)	f(3)	print(3)	print(4)		
		print(4)	print(3)	print(2)	f(2)	print(2)	print(3)	print(4)	
	print(4)	print(3)	print(2)	print(1)	f(1)	print(1)	print(2)	print(3)	print(4)
Output:	4	3	2	1		1	2	3	4



```
Microsoft Visual Studio Debug Console
4 3 2 1 1 2 3 4
C:\Users\DIGITRON\source\repos\MipsProblems\Debug\MipsProblems.exe (process 9844) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

Realizimi i kodit ne Mips:

```
1  .data
2  hapsira: .asciiz " "
3  .text
4  .globl main
5  main:
6      li $t0,4 #int test=4
7      li $t1,0 #int i=0
8      li $t2,4 #int test2=4
9      move $a0,$t0
10     jal printFun
11     addi $t0,$zero,1 #test=1
12     jal printFun2
13
14     #end of program
15     li $v0,10
16     syscall
17
18 printFun:
19     addi $sp,$sp,-8 #lirojme stekun per 2 vlera ku njera do permbaj parametrin test,tjtra return adresen
20     #per kthim ne main funksionin
21     sw $ra,4($sp) #preferohet qe return adresa te ruhet ne pjesen e eperme ngase stack punon ne baze te LIFO(last in,first out)
22     sw $t0,0($sp)
23     bgt $t0,$t1,recurse #test if(test>0) ekzekutojm pjesen else te programit
24     lw $ra,4($sp) #nxjerrim return adresen nga steku
25     lw $t0,0($sp) #nxjerrim vleren t0 nga steku
26     addi $sp,$sp,8 #zbrazim stekun
27     jr $ra #return to main
28
29 printFun2:
30     addi $sp,$sp,-8
31     sw $ra,4($sp)
32     sw $t0,0($sp)
33     ble $t0,$t2,recurse2 #testojm if(test<=4)
34     lw $ra,4($sp)
35     lw $t0,0($sp)
36     addi $sp,$sp,8
37     jr $ra #return ne main
38
39 recurse:
40     #print integer
41     li $v0,1
42     move $a0,$t0
43     syscall
44     #print hapsira
45     li $v0,4
46     la $a0,hapsira
47     syscall
48     addi $t0,$t0,-1 #printFun(test-1)
49     sw $t0,0($sp) #mbishkruejm vleren t0 ne stack
50     j printFun
51
52 recurse2:
53     #print integer
54     li $v0,1
55     move $a0,$t0
56     syscall
57     #print hapsira
58     li $v0,4
59     la $a0,hapsira
60     syscall
61     addi $t0,$t0,1
62     sw $t0,0($sp)
63     j printFun2
```

Sfida që kam hasur ka qenë në pjesën e rekurzive kur $\text{test} < 1$ fillonte që test të rritej dhe të ekzekutohej pjesa e dytë e else-it për pjesën e renditur 1,2,3,4.

Testimet në QtSpim:

The screenshot displays the QtSpim MIPS simulator interface. On the left, the 'Console' window shows a sequence of numbers: 4 3 2 1 1 2 3 4. The main window is titled 'Spim' and contains several tabs: 'Simulator', 'Registers', 'Text Segment', 'Data Segment', 'Window', and 'Help'. The 'Text Segment' tab is active, showing assembly code for the 'User Text Segment' starting at address 00400000. The code includes instructions like 'lw', 'addiu', 'sll', 'addu', 'jal', 'nop', 'ori', 'syscall', 'and', 'bne', 'lwr', 'lwr', 'addi', 'jr', and 'addi'. The 'Registers' window on the right shows the current state of the registers, with values for \$r0 through \$s0. The 'Registers' window also displays the current instruction being executed, which is 'addi \$s0, \$s0, -8' at address 0040006c.

Registers:

- \$r0 = 0
- \$at = 1
- \$v0 = a
- \$v1 = 0
- \$a0 = 10010
- \$a1 = 7ffff
- \$a2 = 7ffff
- \$a3 = 0
- \$t0 = 5
- \$t1 = 0
- \$t2 = 4
- \$t3 = 0
- \$t4 = 0
- \$t5 = 0
- \$t6 = 0
- \$t7 = 0
- \$s0 = 0
- \$s1 = 0

Assembly Code (User Text Segment):

```
00400000: 8fa40000 lw $a0, 0($29) ; 183: lw $a0 0($29) # argc
00400004: 27a50004 addiu $5, $29, 4 ; 194: addiu $a1 $29 4 # argv
00400008: 24a60004 addiu $6, $5, 4 ; 185: addiu $a2 $a1 4 # envp
0040000c: 00041080 sll $2, $4, 2 ; 186: sll $v0 $a0 2
00400010: 00c23021 addu $6, $6, $2 ; 187: addu $a2 $a2 $v0
00400014: 0c100009 jal 0x00400024 [main] ; 188: jal main
00400018: 00000000 nop ; 189: nop
0040001c: 3402000a ori $2, $0, 10 ; 191: li $v0 10
00400020: 0000000c syscall ; 192: syscall # syscall 10
(exit)
00400024: 34080004 ori $8, $0, 4 ; 30: li $t0, 4 #int test=4
00400028: 34090000 ori $9, $0, 0 ; 31: li $t1, 0 #int i=0
0040002c: 340a0004 ori $10, $0, 4 ; 32: li $t2, 4 #int test2=4
00400030: 00082021 addu $4, $0, $9 ; 33: move $a0, $t0
00400034: 0c100012 jal 0x00400048 [printFun] ; 34: jal printFun
00400038: 20080001 addi $8, $0, 1 ; 35: addi $t0, $zero, 1 #test=1
0040003c: 0c10001b jal 0x0040006c [printFun2] ; 36: jal printFun2
00400040: 3402000a ori $2, $0, 10 ; 39: li $v0, 10
00400044: 0000000c syscall ; 40: syscall
00400048: 23bdf0f8 addi $29, $29, -8 ; 42: addi $sp, $sp, -8
0040004c: afbf0004 sw $31, 4($29) ; 43: sw $ra, 4($sp)
00400050: afaf0000 sw $8, 0($29) ; 44: sw $t0, 0($sp)
00400054: 0128002a slt $1, $9, $8 ; 45: bgt $t0, $t1, recurse
#ngs($t0-$t1) psh 1>1 false
00400058: 1420000e bne $1, $0, 56 [recurse-0x00400058]
0040005c: 8fbf0004 lw $31, 4($29) ; 46: lw $ra, 4($sp)
00400060: 8fa80000 lw $8, 0($29) ; 47: lw $t0, 0($sp)
00400064: 23bd0008 addi $29, $29, 8 ; 48: addi $sp, $sp, 8
00400068: 03e00008 jr $31 ; 49: jr $ra
0040006c: 23bdf0f8 addi $29, $29, -8 ; 51: addi $sp, $sp, -8
```

Version 5.1.21 of January 17, 2020
Copyright 1990-2017 by James Larus.
All Rights Reserved.
This program is distributed under a BSD license.
See the file README for a full copyright notice.
This program is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.