

**UNIVERSITETI I PRISHTINËS “HASAN PRISHTINA”**  
**FAKULTETI I INXHINIERISE ELEKTRIKE DHE KOMPJUTERIKE**



**Lënda : Ariktektura e kompjutereve**

**DETYRA 1**

***Profesore :***

*Valon Raca*

***Punoi:***

*Shefket Bylygbashi*

*200714100079*

*Prishtinë , 2022*

## HYRJE

### Detyra:

Opsioni A :

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

```
// A C++ program to demonstrate working of  
// recursion
```

```
1. #include <bits/stdc++.h>  
2.     using namespace std;  
3. void printFun(int test)  
4. {  
5.     if (test < 1)  
6.         return;  
7.     else {  
8.         cout << test << " ";  
9.         printFun(test - 1); // statement 2  
10.        cout << test << " ";  
11.        return;  
12.    }  
13. }  
14. // Driver Code  
15. int main()  
16. {  
17.     int test = 4;  
18.     printFun(test);  
19. }
```

Ky është një program që demonstron se si punon rekursioni.

Rekursioni është procesi në të cilin një funksion thërret veten drejtpërdrejt ose tërthorazi dhe funksioni përkatës quhet funksion rekurziv. Duke përdorur algoritmin rekurziv, disa probleme zgjidhen lehtë.

Shembuj të problemeve të tilla janë : Kullat e Hanoi (TOH), Inorder/Preorder/Postorder Tree Traversals, DFS of Graph, etj.

## Realizimi i kodit ne MIPS

```

1  .text
2  .globl main
3  main:
4  addi $a0, $zero,4
5

```

Në fillim deklarojme funksionin main dhe I japim vlerën e caktuar regjistrit \$a0= 4 përmes instruksionit addi-lformat.

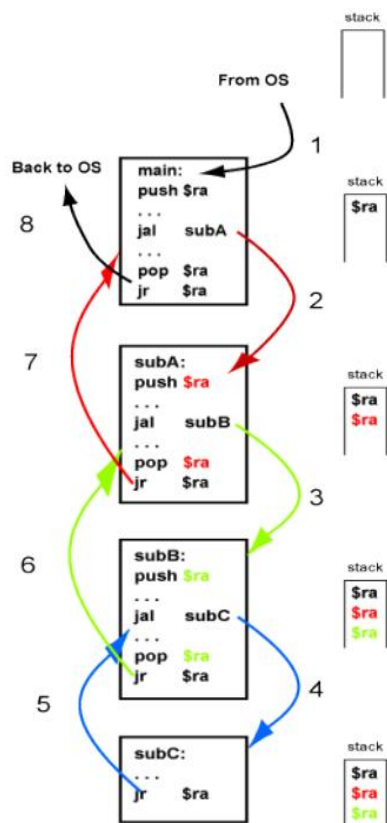
```

8  printFun:
9
10     addi $sp, $sp, -8
11     sw $ra, 4($sp)
12     sw $a0, 0($sp)
13

```

Pasi kemi me thirr procedure I vendosim në stek regjistrat te cilët kemi me I përdorë. Pasi ka me thirr vazhdimisht vetveten në form recursive , kështu që ne duhet ruajtur secilën here që e thirr vetveten return adresën në stek , për arsye që kur të kthehemi nga procedura të dim se ku duhet vazhduar ekzekutuar më tej.Në stek duhet ruajtur edhe \$a0 ,pasi te funksioni kjo zvoglohet për 1.

Pra e vendos return adresen dhe argumentin në stek, I cili punon me parimin LIFO I cili na mundeson rekursionin .



Në programin tone.

\$a0	1
\$ra	400064
	2
	400064
	3
	400064
	4
	400024

```

16
17     slti $t0, $a0, 1
18     beq $t0, $zero, else
19

```

**Shikjome a është  $a0(n)<1$  :** vendose regjistrin \$t0 ne 1 nese \$a0<1, perndryshe \$t0= 0

**beq \$t0, \$zero, else :** nese \$t0 = 0 ,shko te else

ELSE FUNKSIONI:

```

23     else:
24         li $v0 ,1
25         add $t0 , $a0 , $zero
26         syscall
27
28         li $a0, 32
29         li $v0, 11
30         syscall
31
32         addi $a0, $t0, -1
33         jal printFun

```

Tani printon vleren e \$a0 , si dhe hapsiren (ascii -code 32 vlera e space) , dhe e zvoglonë vleren e \$a0 për një .

**jal printFun :** shko pa kusht te printFun dhe ruaje adresen e instruksionit te ardhshem në \$ra.

*Në momentin kur  $n<0$  fillon të ekzekutohet pjesa tjetër e kodit:*

```

19
20     addi $sp, $sp, 8
21     jr $ra
22

```

**addi \$sp, \$sp, 8 :** e ngrit vleren në stek për 8 më lart

**jr \$ra :** shko pa kusht te instruksioni adresa e të cilit është në regjistrin \$ra

Fillojmë ti nxjerrim të dhënat nga steku:

```
35
36     lw $a0, 0($sp)
37     lw $ra, 4($sp)
38     addi $sp, $sp, 8
39
40
41     li $v0, 1
42     add $t0, $a0, $zero
43     syscall
44
45     li $a0, 32
46     li $v0, 11
47     syscall
48     jr $ra
```

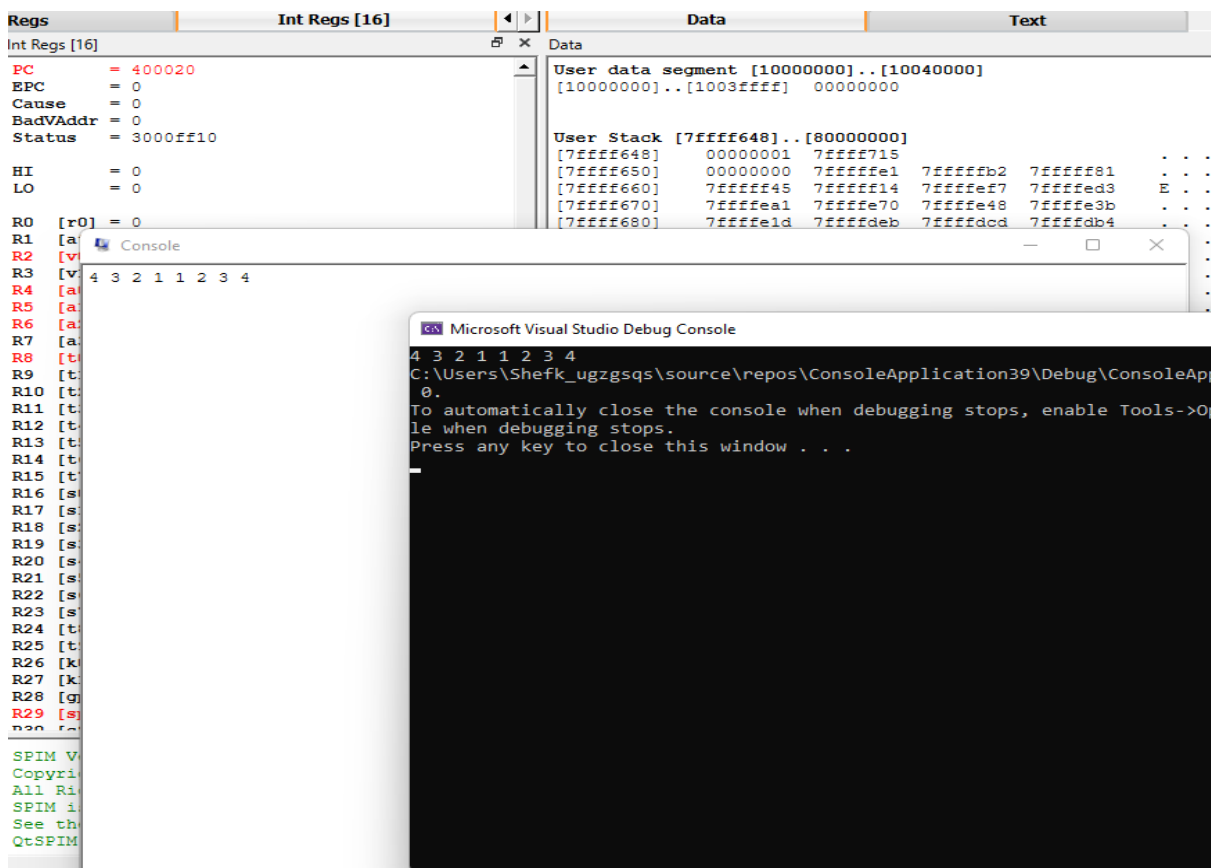
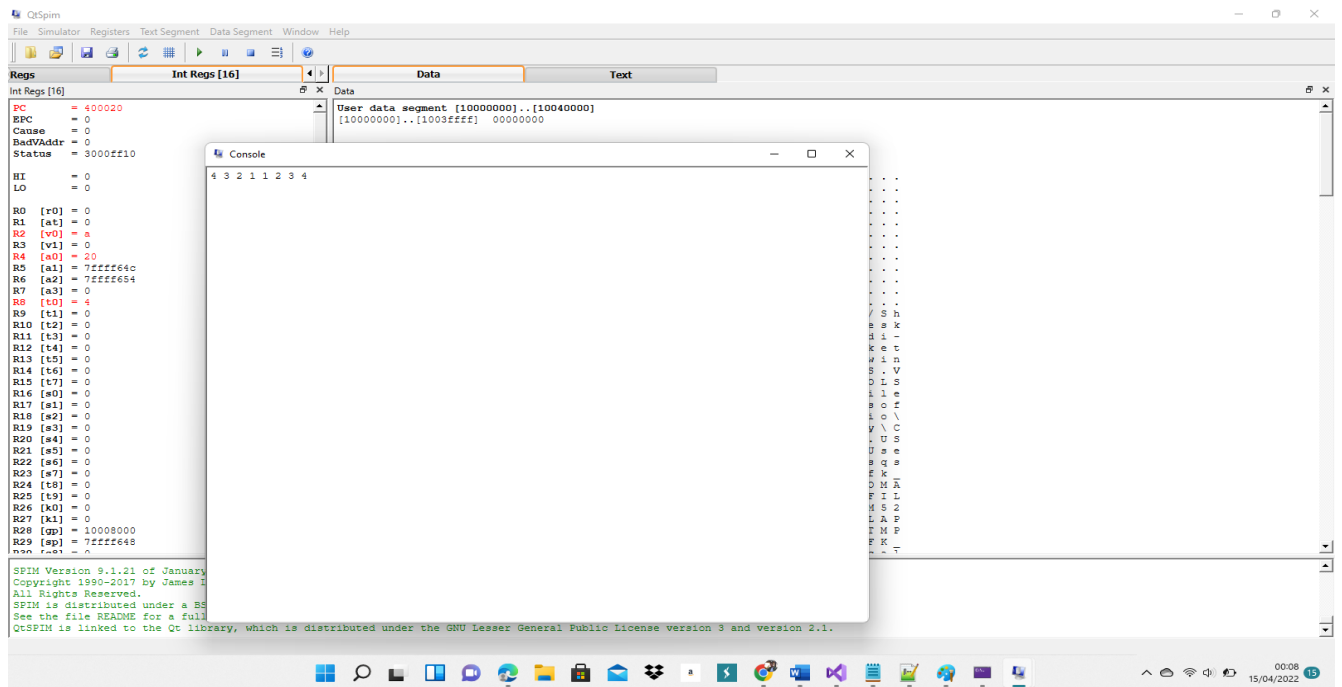
\$a0	1
\$ra	400064
	2
	400064
	3
	400064
	4
	400024

Shtypim vlerat e nxjerra dhe hapsirën pastaj vazhdon përsëritja e kodit deri në nxjerrjen e vlerës se fundit nga steku nmrin 4.

Rezultati : 4 3 2 1 1 2 3 4

## Testimet me QtSpim

Testimi I kodit që është shkruar në MIPS assembler që korrespondon me kodin e dhënë në C++ për vleren 4.



Krahasimi I Consolave si rezultat I ekzekutimit te kodit te shkruar ne C++ dhe MIPS assembler per hyrjen 12.

The screenshot displays the Visual Studio Debug Console with the MIPS architecture selected. The 'Int Regs [16]' window shows the following register values:

Register	Value
PC	400020
EPC	0
Cause	0
BadVAddr	0
Status	3000ff10
HI	0
LO	0
R0 [r0]	0

The 'Data' window shows the 'User data segment' and 'User Stack' memory dump. The 'User Stack' dump includes the following addresses and values:

Address	Value
[7ffff648]	00000001 7ffff715
[7ffff650]	00000000 7ffffe1 7ffffb2 7ffff81
[7ffff660]	7ffff45 7ffff14 7ffffef7 7ffffed3
[7ffff670]	7ffffea1 7ffffe70 7ffffe48 7ffffe3b
[7ffff680]	7ffffeld 7ffffdeb 7ffffdcd 7ffffdb4

The 'Console' window shows the output of the program, which is a sequence of numbers from 12 down to 0, followed by a message indicating the program exited with code 0. The 'Microsoft Visual Studio Debug Console' window shows the command prompt output, which includes the path to the executable and the exit code.