**UNIVERSITETI I PRISHTINËS "HASAN PRISHTINA" FAKULTETI**

**I INXHINIERISË ELEKTRIKE DHE KOMPJUTERIKE**


**Lënda:** Arkitektura e Kompjuterëve

**Tema:** Konvertimi i kodit në C++ në MIPS Assembly Code

**Studenti:** Meriton Kryeziu

**ID:** 20071410005

**Profesori:**

Valon Raca

Prishtinë, 2022

# Përmbajtja



Prishtinë, 2022

# Hyrje

**Opsioni A:**

```cpp
#include <iostream>
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);
   cout<<"\n\n"; // per printim
}
```

Detyra ka qenë që kodi i dhënë më lart në C++ të konvertohet në kod të asemblerit.

Ky program përmban funksionin kryesor main i cili thërret funksionin rekursiv printFun me parametër numrin ***test***. Funksioni printFun shikon nese *parametri hyrës* eshte më i madh ose i barabartë me 1 atëherë thërret veten me parametrin e ri hyrës për 1 më të vogël se parametri i kaluar.

## Realizimi i kodit në MIPS

```
.text
.globl main
main:
   addi $a1, $zero, 1              #int test = 4
```

Tek inicializimi i parametrit hyrës në thirrjen fillestare të funksionit rekursiv, kam pasur probleme me thirrjen e numrave nga ***.data,*** e kam marrë si vlerë imediate.

```
printFun:
   addi $sp, $sp, -8          #decrement stack pointer 2*4 to store $ra,$a1
   sw $ra, 4($sp)
   sw $a1, 0($sp)

   slti $t0, $a1, 1                #if test<1
   beq $t0, $zero, unravel         #jump to start popping and printing
#arguments from stack

   addi $sp, $sp, 8
   jr $ra
```

Funksioni printFun fillimisht krijon hapësirën e nevojshme për ruajtjen e dy integerave, duke zbritur stack pointerin për 8Byte (2*4Byte). Në 4Byte e parë e ruan vlerën e argumentit **test** (ne mips *$a1*), në 4Byte e dytë ruan adresën e kthimit.

Tek kushtëzimi i funksionit printFun, ku shikohet nëse plotësohet kushti që parametri hyrës të jetë më i vogel se 1 dhe funkisoni kthehet ku është thirrur, në mips e kam implementuar me Set Less Than Immediate dhe Branch if Equal. Duke ruajtur rezultatin në $t0 me vlerat $t0 = 1 ($a1 < 1), 0 ($a1 >=1) dhe duke shikuar nëse rezultati i $t0 është 0 atëherë kërcen tek etiketa ***unravel.*** Nëse nuk ndodh degëzimi stack pointeri duhet kthyer prap në pozitën para zvogëlimit të stack pointerit, sepse në këtë rast nuk ruajmë argumentin dhe return adresën në stack, pastaj kthehemi në adresën e kthimit.

```
unravel:                        #cout << test << " ";
   move $a0, $a1                #print argument
   li $v0, 1
   syscall

   li $a0, ' '                  #print space
   li $v0, 11
   syscall
```

Në pjesën hyrëse të etiketës ***unravel*** bëhet printimi i argumenteve në console përmes syscall (System Calls).

```
   addi $a1, $a1 -1             #test--

   jal printFun                 #printFun(test - 1); // statement 2

   lw $a1, 0($sp)               #pop argument and return address
   lw $ra, 4($sp)
   addi $sp, $sp, 8
```

Vazhdimi i etiketës ***unravel,*** në këte pjesë argumenti ose parametri hyrës i procedures printFun zvogëlohet për 1 dhe thirret procedura me parametrin e ri të zvogëluar për 1. Duke parë kodin dhe duke ditur kushtëzimin e parametrit hyrës në funksionin printFun dimë që ky funksion do ta thërret veten deri sa parametri hyrës të jetë më i madh ose i barabartë me 1. Pra nëse në hyrjen fillestare të funksionit kemi dhënë vlerën 4 atëherë e dimë që ky funksion do i ketë parametrat hyrës 4, 3, 2, 1 dhe 0 [për vlerën 0 nuk plotësohet kushti dhe kemi kthim në thirrjen paraprake me return].

```
                                #cout << test << " ";
  move $a0, $a1                 #print argument
  li $v0, 1
  syscall

  li $a0, ' '                   #print space
  li $v0, 11
  syscall

  jr $ra                        #jump to return address
```

Në vazhdim printohet argumenti i nxjerrë nga stack-u dhe printohet një hapsirë pas argumentit.

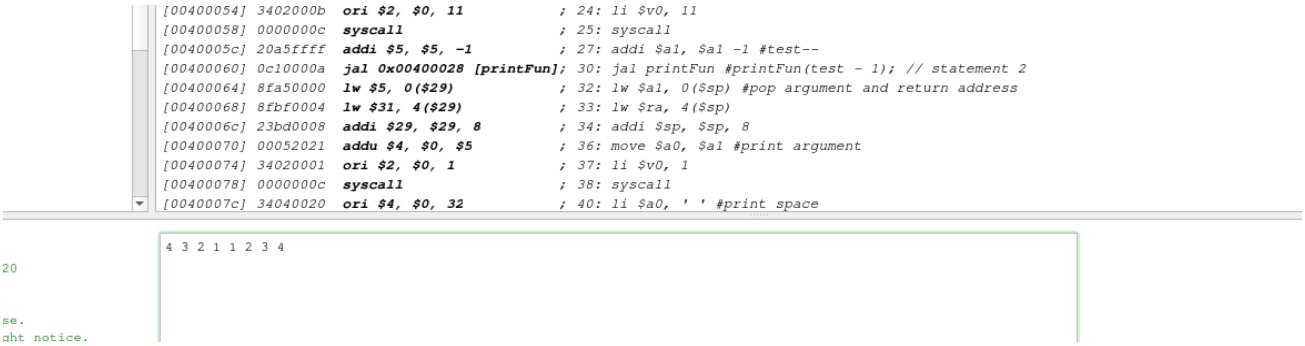
Dhe në fund të etiketës ***unravel*** kemi kërcimin te adresa e nxjerrë nga stack-u, duke ditur që pas qdo thirrje në stack kemi ruajtur vlerat e adresave kthyese dimë që me kërcimin në adresën kthyese jemi duke u kthyer mbrapa në ekzekutim me parametrat paraprak. Duke ditur që Stack-u punon me parimin First In First Out (FIFO) dhe parametrat hyrës ishin sekuence 4, 3, 2, 1 mund ta paramendojmë që kthimi do të printojë 1, 2, 3, 4.

```
exit:
  li $a0, 10
  syscall
```

Në fund të programit është bërë dalja e programit me system call.

## Testimet me QtSpim

Rezultati i ekzekutimit të programit në VSCode (c++) dhe QtSpim (MIPS) për vlerën hyrëse 4.


Rezultati i ekzekutimit të programit në VSCode (c++) dhe QtSpim (MIPS) për vlerën hyrëse 20.