

UNIVERSITETI I PRISHTINËS “HASAN PRISHTINA”
FAKULTETI I INXHINIERISË ELEKTRIKE DHE KOMPJUTERIKE
DEPARTAMENTI I INXHINIERISË KOMPJUTERIKE



Detyra: C++ to MIPS Assembly (Opsioni A)

Data: 18.04.2021

Studenti: Uran Lajçi

Veglat e përdorura: Notepad++, QtSpim

ID: 180714100065

Email: uran.lajci@student.uni-pr.edu

Lënda: Arkitektura e Kompjuterve

Mentorët: Valon Raca, Vlera Alimehaj

1. Hyrje

Opsioni A

```
#include <iostream>
using namespace std;
int fib(int x) {
    if((x==1)|| (x==0)) {
        return(x);
    } else {
        return(fib(x-1)+fib(x-2));
    }
}
int main() {
    int x , i=0;
    cout << "Enter the number of terms of series : ";
    cin >> x;
    cout << "\nFibonnaci Series : ";
    while(i < x) {
        cout << " " << fib(i);
        i++;
    }
    return 0;
}
```

Kodi i shkruar më lartë është në gjuhën programuese C++ dhe e ka për detyrë të na tregoj numrat e serisë fibonacci deri tek numri që ne i'a caktojmë.

Detyra përbëhet nga funksioni kryesorë main dhe funksioni fib. Në funksionin main kërkojmë të shënohet një numër me anë të tastierës, numër i cili më pas krahasohet në loop-en while brenda së ciles shtypen termat e series. Për t'i shfaqur termat e series brenda loop-es thërrasim funksionin fib. Kjo vazhdon deri sa i është e barabartë me x-numrin e dhënë përmes tastierës.

Funksioni fib ka një parameter dhe përbehet nga kushtet if dhe else. Nëse plotësohet kushti i parë kthehet vetë parametri x. Nëse nuk plotësohet kushti i parë atëhere kthehet $\text{fib}(x-1) + \text{fib}(x-2)$. Ky rast paraqet rekursion sepse funksioni e thërret vetveten. Funksioni përfundon atëhere kur plotësohet kushti $\text{if}((x==1)|| (x==0))$ dhe nuk kalohet në else.

2. Realizimi i kodit në MIPS

```
.data
hapsire:      .ascii " "
sheno:        .ascii "Enter the number of terms of series : "
seria:        .ascii "\nFibonnaci Series : "

.text
main:
    li $v0, 4
    la $a0, sheno    # printimi i tekstit qe gjendet tek sheno
    syscall

    li $v0, 5        # ja mundeson perdoruesit ta shenoj numrin nga tastiera
    syscall
    move $t4, $v0     # e vendosim numrin e dhene nga tastiera ne $t4

    addi $t0, $zero, 0 # deklarimi dhe inicializimi i variables i=0

    li $v0, 4
    la $a0, seria    # printimi i tekstit qe gjendet tek seria
    syscall

while:
    bgt $t4, $t0, Shtyp    # nese x=$t4 > i=$t0 atehere shko te
    j Exit                # labela Shtyp, nese jo shkon te Exit

Shtyp:
    move $a1, $t0    # e vendosim i=$t0 ne $a1 per ta derguar
                    # si argument ne funksionin fib
    jal fib          # thirrja e funksionit

    li $v0, 1
    move $a0, $v1    # printimi i rezultatit qe na e kthen funksioni
    syscall

    li $v0, 4
    la $a0, hapsire    # printimi i hapsires
    syscall

    addi $t0, $t0, 1    # i++
    j while            # kercimi te labeli while

Exit:
    li $v0, 10    # terminimi i programit
    syscall
```

```

fib:
    bne $a1, $zero, elseIf    # nese x=$a1 nuk eshte 0 shkon te labeli elseIf
    add $v1, $zero, $a1      # return(x)
    jr $ra

elseIf:
    addi $t1, $zero, 1        # $t1=1
    bne $a1, $t1, else        # nese x=$a1 nuk eshte 1 shkon te labeli else
    add $v1, $zero, $a1      # return(x)
    jr $ra

else:
    # As callee, save return address and saved registers
    addi $sp, $sp, -12        # Adjust stack for pushing 3 items
    sw $ra, 8($sp)            # Push return address to stack
    sw $s0, 4($sp)            # Push $s0 to stack
    sw $s1, 0($sp)            # Push $s1 to stack

    # As caller, save arguments
    addi $s0, $a1, -1         # Store n-1 in $s0

    # fib(n-1)
    move $a1, $s0              # Set argument to n-1
    jal fib                    # fib(n-1)
    move $s1, $v1              # Store returned value in $s0

    # fib(n-2)
    addi $s0, $s0, -1         # Calculate n-2 in $s0
    move $a1, $s0              # Set argument to n-2
    jal fib                    # fib(n-2)

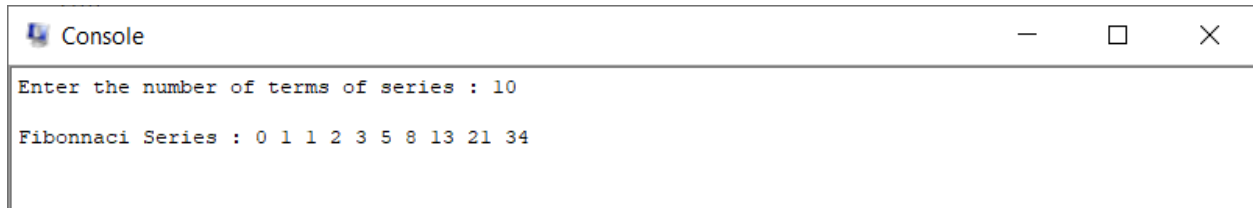
    # Set return value
    add $v1, $v1, $s1         # Store returned value in $t0

    # Pop registers from the stack
    lw $s1, 0($sp)            # Pop $s1 off stack
    lw $s0, 4($sp)            # Pop $s0 off stack
    lw $ra, 8($sp)            # Pop return address off stack
    addi $sp, $sp, 12         # Adjust stack for popping 3 items off

    jr $ra                    # Return

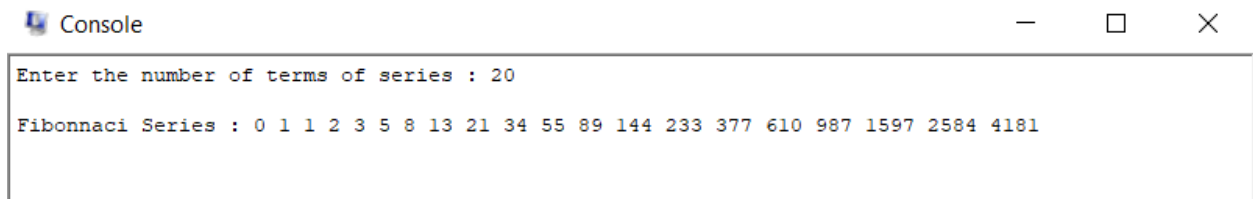
```

3. Testimet me QtSpim



```
Console
Enter the number of terms of series : 10
Fibonnaci Series : 0 1 1 2 3 5 8 13 21 34
```

Këtu shihet testimi i programit në QtSpim kur n-in e japim numrin 10. Programi kthen serinë fibonaci për 10 numrat e parë.



```
Console
Enter the number of terms of series : 20
Fibonnaci Series : 0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181
```

Këtu shihet testimi i programit në QtSpim kur n-in e japim numrin 20. Programi kthen serinë fibonaci për 20 numrat e parë. Sa me i madh të jetë numri i termave, aq më ngadalë do të shfaqet seria për shkak të llogaritjeve që bëhen për të gjetur numrat më të mëdhenjë ku funksioni e therret vetvetën shumë herë.

Rezultatet e detyrës në C++ janë të njëjta me rezultatet e detyrës në MIPS.

4. Përfundimi

Nga përvoja e krijuar në këtë detyrë kam arritur që të kuptoj MIPS Assembly deri në një nivel sa të mundem që një kod të thjeshtë në C++ apo në një gjuhë tjetër të lartë ta kthej në MIPS Assembly.

Mësimi paraprak i deklarimeve, inicializimeve, kushteve (if, else, else if), loop-at dhe funksionet më kanë ndihmuar shumë që të di ta ndaj detyrën në pjesë. Ato pjesë të kodeve të cilat i kam testuar në fund i kam bërë bashkë në një kod për të parë funksionimin e plotë të detyrës.