

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



Lënda: Arkitektura dhe Organizimi i Kompjuterëve
Tema: Konvertimi i kodit në C++ në MIPS Assembly Code

Profesori:
Valon Raca

Studenti:
Erlis Lushtaku
ID: 190714100055

Prishtinë, 2021

Përmbajtja

1. Hyrje.....	3
2. Realizimi i kodit ne MIPS	5
3. Testimet me QtSpim	7

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;
}
```

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

Ky kod merr si input numrin e plotë x nga përdoruesi dhe si rezultat kthen x terma të *Serisë së Fibonnaci-t* të ndarë me hapësira.

Seria e Fibonnacit gjenerohet në këtë mënyrë:

- Anëtari i parë është 0,
- Anëtari i dytë është 1,
- Anëtari i n -të është shuma e anëtarëve $n-1$ dhe $n-2$.

Me një unazë iterojmë duke rritur variablën i derisa është më e vogël se x . Në çdo iterim e thirrjmë funksionin rekursiv *fib* me parametër i i cili kthen anëtarin e i -të të Serisë së Fibbonaci-t dhe printojmë rezultatin.

Ky program nuk është eficient sepse për kalkulimin e secilit anëtar të serisë duhet të kalkulojë të gjithë anëtarët paraprak, gjë që është e panevojshme. Në këtë rast ka kompleksitet kohor $O(2n^2)$. Programi i rishkruar në mënyrë që të ketë kompleksitet kohor $O(n)$ është dhënë në vazhdim:

```
#include <iostream>
using namespace std;

int main() {
    int x, i = 0;
    cout << "Enter the number of terms of series : ";
    cin >> x;
    cout << "\nFibonacci Series : ";

    // Dy anëtarët e parë janë 0 dhe 1
    int t1 = 0, t2 = 1;
    cout << " " << t1 << " " << t2;

    while (i < x - 2) {
        // t1 merr vlerën e t2, ndërsa t2 merr vlerën t1 + t2
        t2 = t1 + t2;
        t1 = t2 - t1;
        cout << " " << t2;
        i++;
    }
    return 0;
}
```

Kodi në MIPS është bazuar në versionin origjinal siç është dhënë detyra.

2. Realizimi i kodit ne MIPS

```
25      # Main loop body
26      loop:
27          bge      $s1, $s0, exit  # if(i >= x) jump to exit <=> while(i < x)
28
```

Për të implementuar kushtin e unazës $i < x$ është përdorur pseudo instruksioni bge (branch if greater than or equal). Kjo ka qenë e mundur të shkruhet me anë të instruksioneve elementare:

slt \$at, \$s1, \$s0

beq \$at, \$zero, exit

```
34      # Call function
35          move     $a0, $s1          # Argument 1: i ($s1)
36          jal      fib              # Save current PC in $ra, and jump to fib
37          move     $s2, $v0          # Return value saved in $v0
38
39      # Print result of fib
40          li       $v0, 1            # print_int syscall code = 1
41          move     $a0, $s2          # Load integer to print in $a0
42          syscall
```

Për të thirrur funksionin *fib* së pari vendosim argumentin e funksionit *i* (i cili ruhet në $\$s1$) në regjistrin $\$a0$. Përdorim komandën *jal* për të ruajtur adresën e tanishme në $\$ra$ dhe për të kërcyer te labela *fib*. Pas kryerjes së funksionit ekzekutimi vazhdon aty ku ka mbetur në pjesën main. Në këtë rast ekzekutohet komanda për vendosjen e vlerës së regjistrit $\$v0$ (ku ruhet vlera kthyesë e funksionit *fib*) në regjistrin $\$s2$. Kjo bëhet sepse $\$v0$ do të mbishkruhet dhe na humbet vlera kthyesë e funksionit. Në fund printohet vlera aktuale në regjistrin $\$s2$.

```
52      fib:
53          li       $t0, 1
54          beq      $a0, $t0, returnx  # if(x == 1)
55          beq      $a0, $zero, returnx # if(x == 0)
```

Për të realizuar kodin $\text{if } (x == 1 \parallel x == 0) \{ \text{return}(x); \}$ janë përdorur dy instruksione të njëpasnjëshme *beq* ashtu që nëse të paktën njëra plotësohet të shkohet të labella *returnx* ku kthehet vlera e x dhe përfundon kjo thirrje e funksionit. Për të shqyrtuar a është vlera e $x = 1$ (x ruhet nw $\$a0$) duhet që $\$a0$ të krahasohet me një regjistër që ka vlerën 1 ($\$t0$), pasi që nuk mund të bëhet direkt krahasimi me një vlerë immediate.

```

57 # Adjust for calling fib(x - 1)
58     addi    $sp, $sp, -8      # Adjust stack pointer
59     sw      $ra, 0($sp)      # Save $ra
60
61     addi    $a0, $a0, -1      # x = x - 1
62     jal fib                                # Save current PC in $ra, and jump to fib
63
64     sw      $v0, 4($sp)      # Save $v0 (return value of fib(x - 1))

```

Para thirrjes rekursive me parametër $x - 1$ duhet që ta ruajmë në stack adresën e kthimit pasi që regjistri $\$ra$ mbishkruhet nga komanda *jal*. Me instruksionin *addi* e zvogëlojmë vlerën e x për 1 të cilën funksioni *fib* do ta marrë si parametër. Vlerën kthyesë të kësaj thirrje rekursive duhet ta ruajmë gjithashtu në stack pasi që regjistri $\$v0$ do të mbishkruhet nga thirrja rekursive me parameter $x - 2$. Meqë duhet të ruajmë në stack dy variabla integer që kanë madhësi nga 4 byte atëherë në fillim e zhvendosim pointerin e stack-ut për 8.

```

66 # Adjust for calling fib(x - 2)
67     addi    $a0, $a0, -1      # $a0 - 1 = (x - 1) - 1 = x - 2
68     jal fib                                # Save current PC in $ra, and jump to fib
69
70     addi    $a0, $a0, 2       # Reset $a0 to initial value (x)

```

Për thirrjen e funksionit rekursiv $fib(x - 2)$ duhet ta zvogëlojmë prap vlerën në $\$a0$ për 1 ashtu që të fitojmë $x - 2$. Në këtë rast nuk është e nevojshme të ruhet në stack $\$ra$ para thirrjes së funksionit sepse nuk na duhet adresa e kthimit të funksionit që e thirrëm më herët ($fib(x - 1)$) e cila për momentin është në $\$ra$, e as $\$v0$ pas thirrjes së funksionit pasi që nuk do të ketë thirrje tjetër rekursive që ta mbishkruaj. Në çdo thirrje rekursive vlera në $\$a0$ zvogëlohet për 2, prandaj në mënyrë që kur kthehem te thirrja paraprake vlera në $\$a0$ të jetë sa parametri me të cilin është bërë thirrja përkatëse e funksionit, duhet që në fund të çdo thirrje ta kthejmë $\$a0$ në vlerën fillestare duke ia shtuar 2.

```

72 # fib(x - 1) + fib(x - 2)
73     lw      $t1, 4($sp)      # $t1 = 4($sp) = fib(x - 1), $v0 = fib(x - 2)
74     add     $v0, $t1, $v0    # $v0 = fib(x - 1) + fib(x - 2)

```

Për të kalkuluar shumën $fib(x - 1) + fib(x - 2)$ së pari marrim nga stack vlerën kthyesë të funksionit $fib(x - 1)$ të cilën e kemi ruajtur më herët dhe e vendosim në një regjistrë ($\$t1$). Vlera kthyesë e funksionit $fib(x - 2)$ është në $\$v0$. Mbledhim këto dy vlera dhe po i ruajmë prap në $\$v0$ si vlerë kthyesë e tërë funksionit.

```

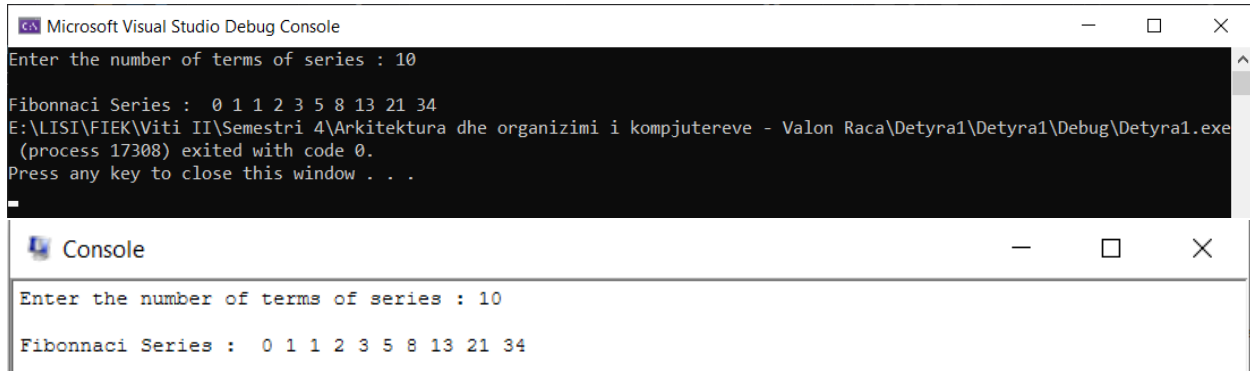
76 # Return fib(x - 1) + fib(x - 2)
77     lw      $ra, 0($sp)      # Retrieve return address
78     addi    $sp, $sp, 8      # Reset stack pointer
79     jr      $ra

```

Para se të përfundojmë funksionin duhet ta resetojmë adresën e kthimit e cila është mbishkruar nga funksionet $fib(x - 1)$ dhe $fib(x - 2)$. E resetojmë edhe stack pointerin dhe përfundojmë këtë thirrje të funksionit.

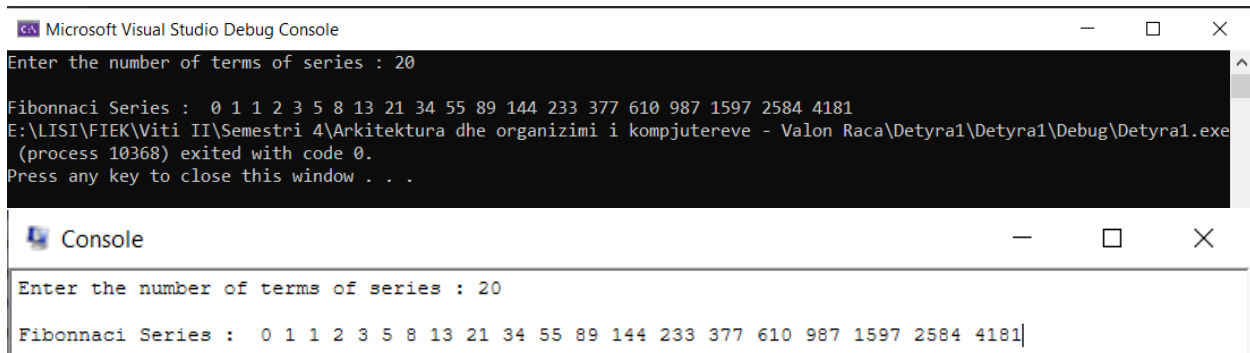
3. Testimet me QtSpim

Rezultati i ekzekutimit të kodit në C++ dhe në QtSpim për vlerën hyrëse 10:



The image shows two windows from a Microsoft Visual Studio environment. The top window is the 'Microsoft Visual Studio Debug Console', which has a black background and white text. It displays the following output: 'Enter the number of terms of series : 10', 'Fibonnaci Series : 0 1 1 2 3 5 8 13 21 34', and a file path 'E:\LISI\FIEK\Viti II\Semestri 4\Arkitektura dhe organizimi i kompjutereve - Valon Raca\Detyra1\Detyra1\Debug\Detyra1.exe (process 17308) exited with code 0.' followed by 'Press any key to close this window . . .'. The bottom window is titled 'Console' and has a white background with black text. It shows the same input and output as the debug console: 'Enter the number of terms of series : 10' and 'Fibonnaci Series : 0 1 1 2 3 5 8 13 21 34'.

Rezultati i ekzekutimit të kodit në C++ dhe në QtSpim për vlerën hyrëse 20:



The image shows two windows from a Microsoft Visual Studio environment. The top window is the 'Microsoft Visual Studio Debug Console', which has a black background and white text. It displays the following output: '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', and a file path 'E:\LISI\FIEK\Viti II\Semestri 4\Arkitektura dhe organizimi i kompjutereve - Valon Raca\Detyra1\Detyra1\Debug\Detyra1.exe (process 10368) exited with code 0.' followed by 'Press any key to close this window . . .'. The bottom window is titled 'Console' and has a white background with black text. It shows the same input and output as the debug console: 'Enter the number of terms of series : 20' and 'Fibonnaci Series : 0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181'.

Shohim se rezultatet përputhen prandaj konkludojmë se kodi në MIPS është duke funksionuar si duhet.