Nirdesh Bhandari
CS 320 Principles of Computer Organization
Project 1 QtSpim

1. Implement the Fibonacci function, and write a main function that enables user to input a number n and output the Fibonacci sequence from F0 to Fn.

Description:

This program asks user to input a non-negative integer and prints the Fibonacci sequences up to that integer.

The program runs by first setting x=0 and y =1 and then prints x then it adds x and y and saves it as z. It then saves x = y and y = z, increments the counter and loops over again. This way the last two values are saved and the Fibonacci sequence is maintained while keeping track of the last two values for the next iteration.

The following table shows how my program saves the values through each iteration of the While loop.

Iteration	Z(z=x+y)	X(x=y)	Y(y=z)	Fibonacci Print(X)
0	-	0	1	0
1	1	1	1	1
2	2	1	2	1
3	3	2	3	2
4	5	3	5	3
5	8	5	8	5

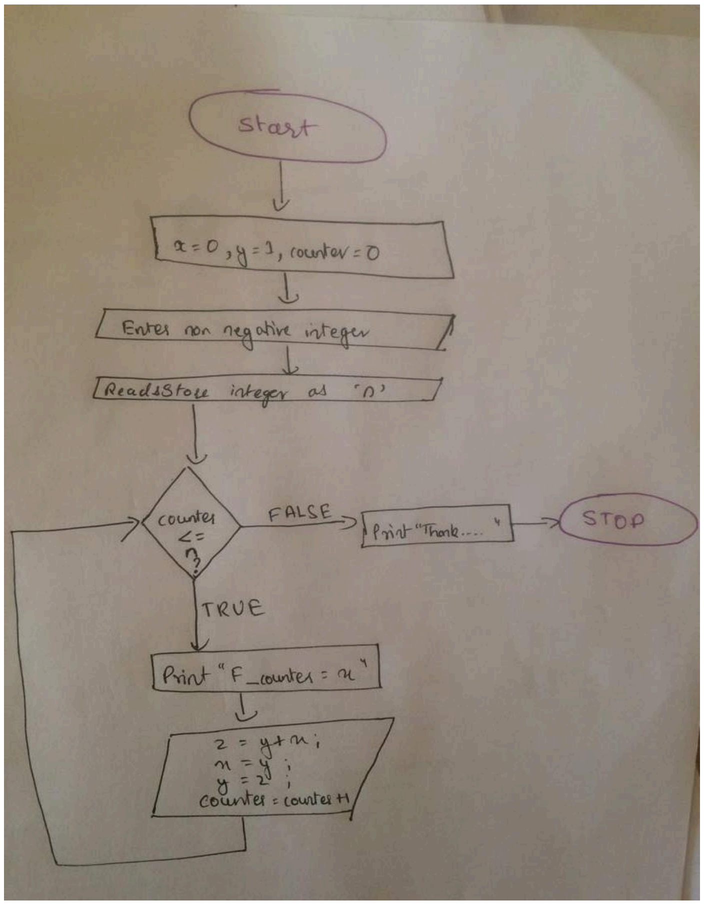
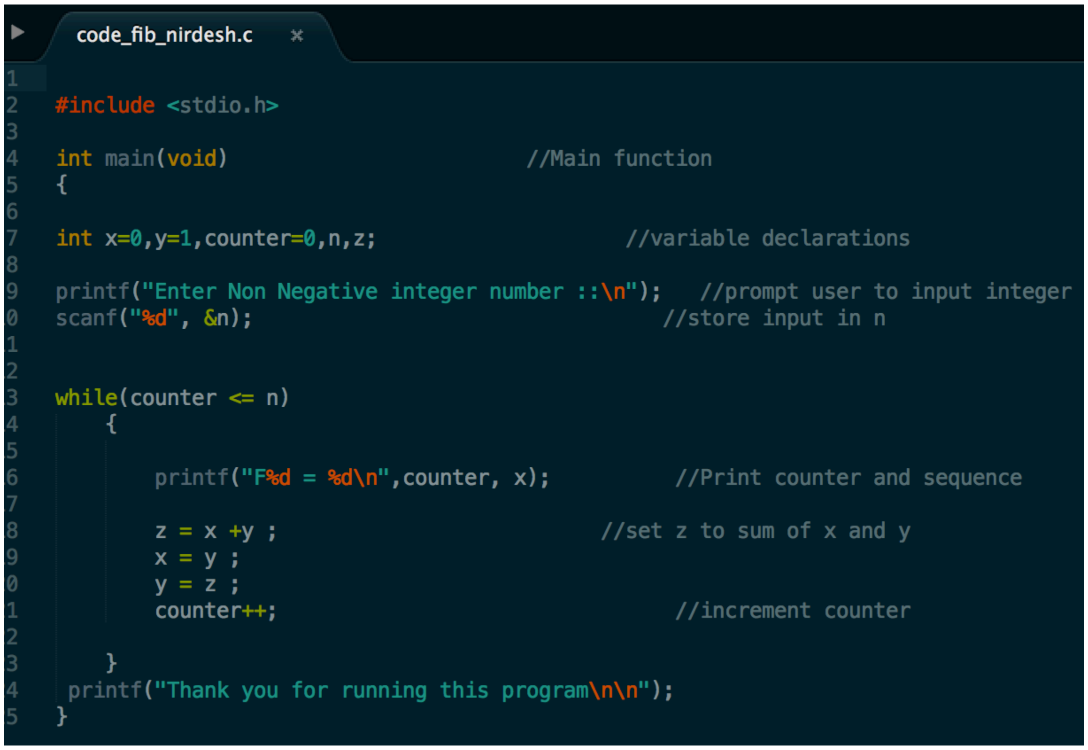


Figure: The flowchart for my C program.

I developed the following C program to convert to MIPS assembly code. Rather than using recursive calls or a different function, I kept things simple.

The C code built on this flow chart looks like the following:

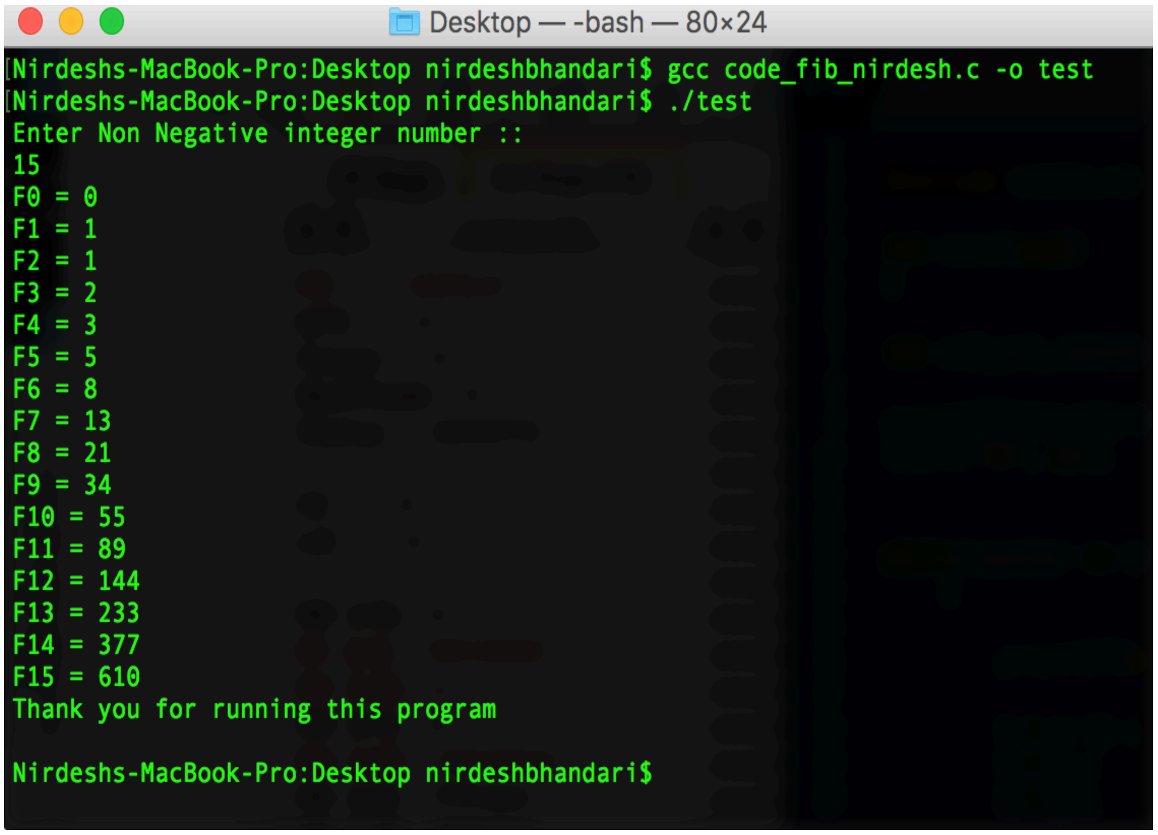

code_fib_nirdesh.c

```c

#include <stdio.h>

int main(void)                          //Main function
{

int x=0,y=1,counter=0,n,z;                      //variable declarations

printf("Enter Non Negative integer number ::\n");   //prompt user to input integer
scanf("%d", &n);                                 //store input in n


while(counter <= n)
    {

        printf("F%d = %d\n",counter, x);          //Print counter and sequence

        z = x +y ;                            //set z to sum of x and y
        x = y ;
        y = z ;
        counter++;                               //increment counter

    }
 printf("Thank you for running this program\n\n");
}
```

My code is designed to use x,y, and z as temporary variables to store the values of the last two variables and only print x after each iteration of the while loop. This way I have minimized the use of excessive amounts of registers and function calling.

To run this code simply type **$gcc code_fib_nirdesh.c –o run**
And then type: **$ ./run**

Screenshot of execution of my C program:

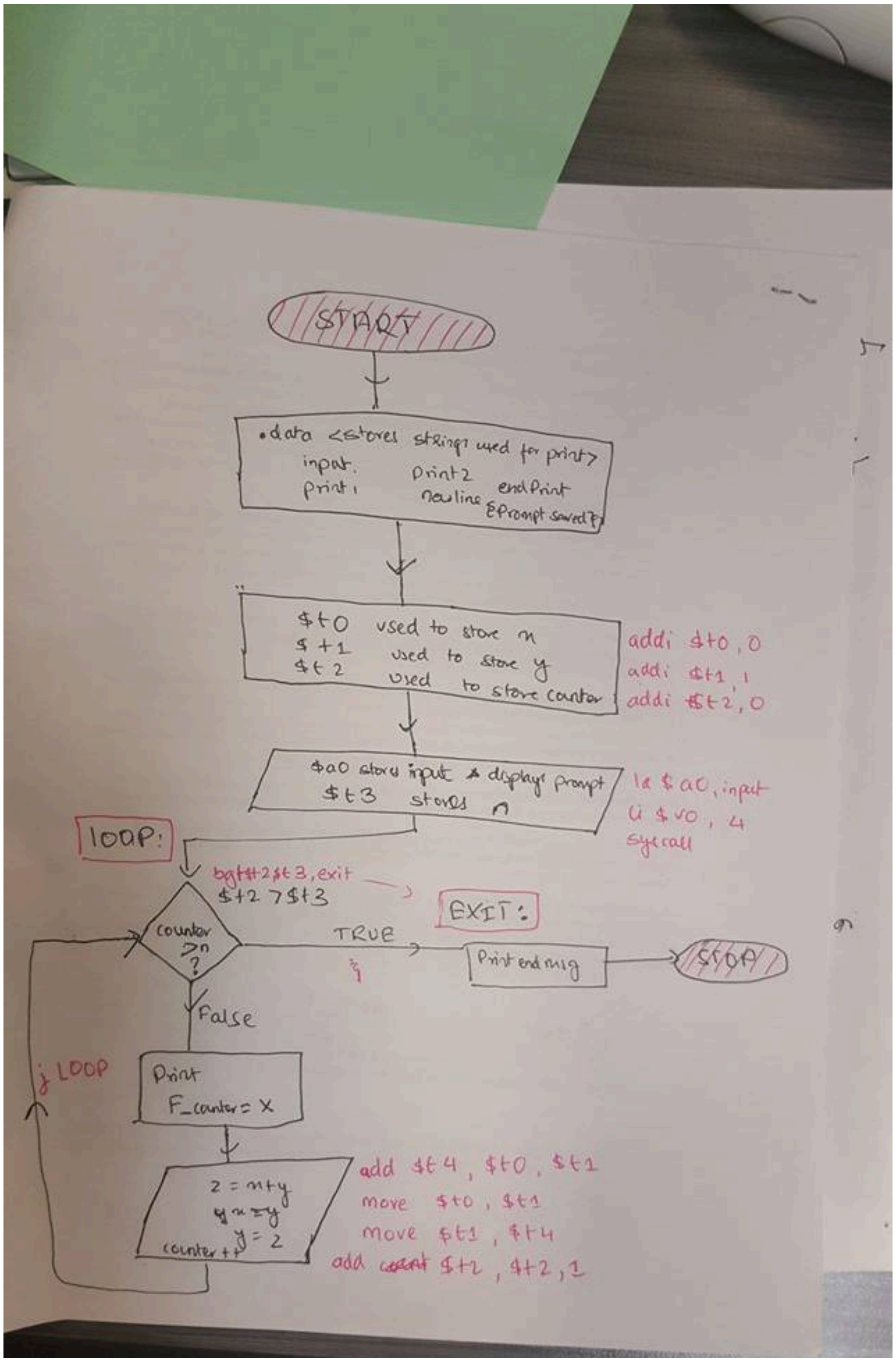


Desktop — -bash — 80×24

```
[Nirdeshs-MacBook-Pro:Desktop nirdeshbhandari$ gcc code_fib_nirdesh.c -o test
[Nirdeshs-MacBook-Pro:Desktop nirdeshbhandari$ ./test
Enter Non Negative integer number ::
15
F0 = 0
F1 = 1
F2 = 1
F3 = 2
F4 = 3
F5 = 5
F6 = 8
F7 = 13
F8 = 21
F9 = 34
F10 = 55
F11 = 89
F12 = 144
F13 = 233
F14 = 377
F15 = 610
Thank you for running this program

Nirdeshs-MacBook-Pro:Desktop nirdeshbhandari$
```

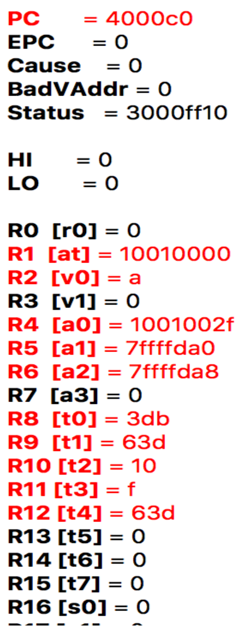
The terminal output shows how I have implemented my design to show each line and the counter to keep track of the iteration in the Fibonacci sequence. I have also added an ending prompt to mark the end of the program.

For my MIPS code, I used the following flowchart to design my code:



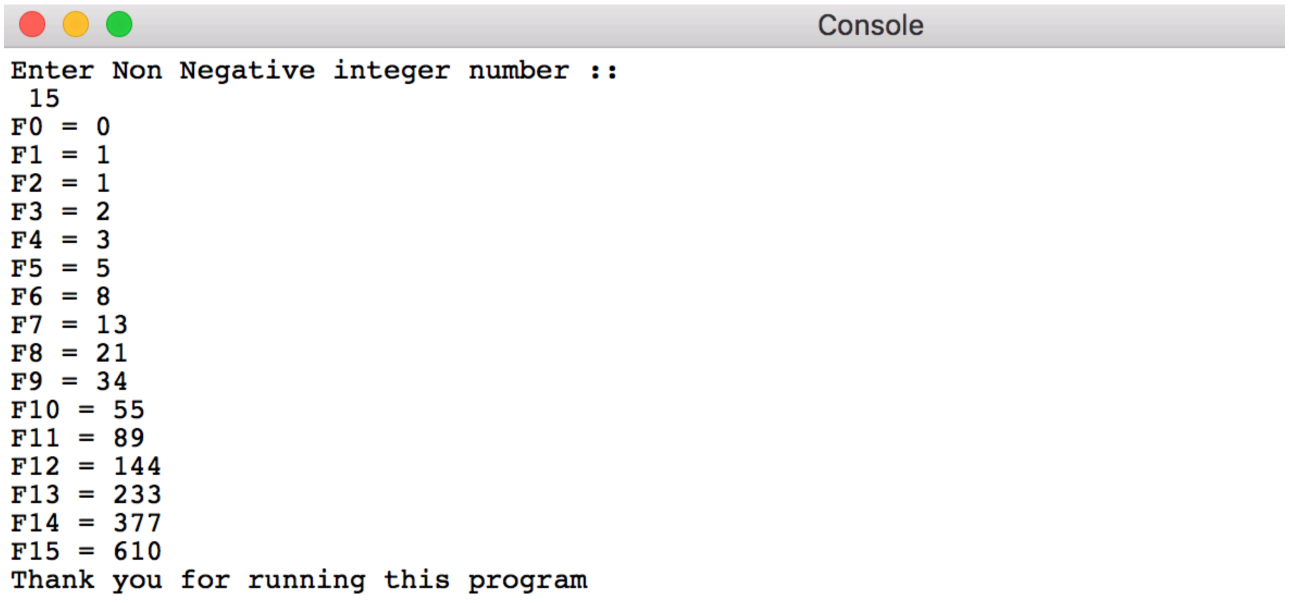
Memory Space and Registers:

$t0 - Used to store values of x
$t1 – Used to store values of y
$t2 - Used to value for counter
$t3 - Used to store value of n
$t4 - Used to store value of z
$a0 - Used to pass arguments
$v0 - System command



```
PC       = 4000c0
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 3000ff10

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 10010000
R2  [v0] = a
R3  [v1] = 0
R4  [a0] = 1001002f
R5  [a1] = 7ffffda0
R6  [a2] = 7ffffda8
R7  [a3] = 0
R8  [t0] = 3db
R9  [t1] = 63d
R10 [t2] = 10
R11 [t3] = f
R12 [t4] = 63d
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
```

Upon execution on QtSpim simulator, the results look like this :



```
Enter Non Negative integer number ::
 15
F0 = 0
F1 = 1
F2 = 1
F3 = 2
F4 = 3
F5 = 5
F6 = 8
F7 = 13
F8 = 21
F9 = 34
F10 = 55
F11 = 89
F12 = 144
F13 = 233
F14 = 377
F15 = 610
Thank you for running this program
```