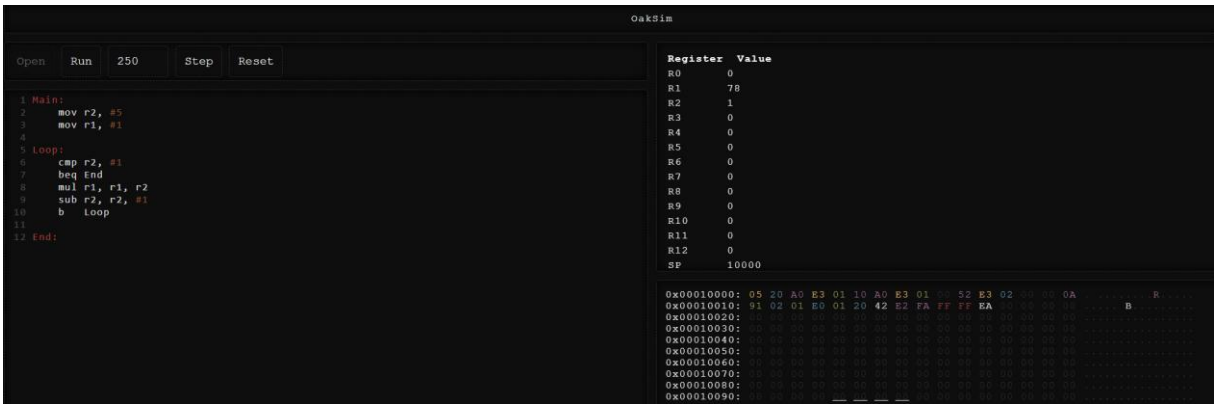
Week 4 – Software

Student number: 578438

Assignment 4.1: ARM assembly

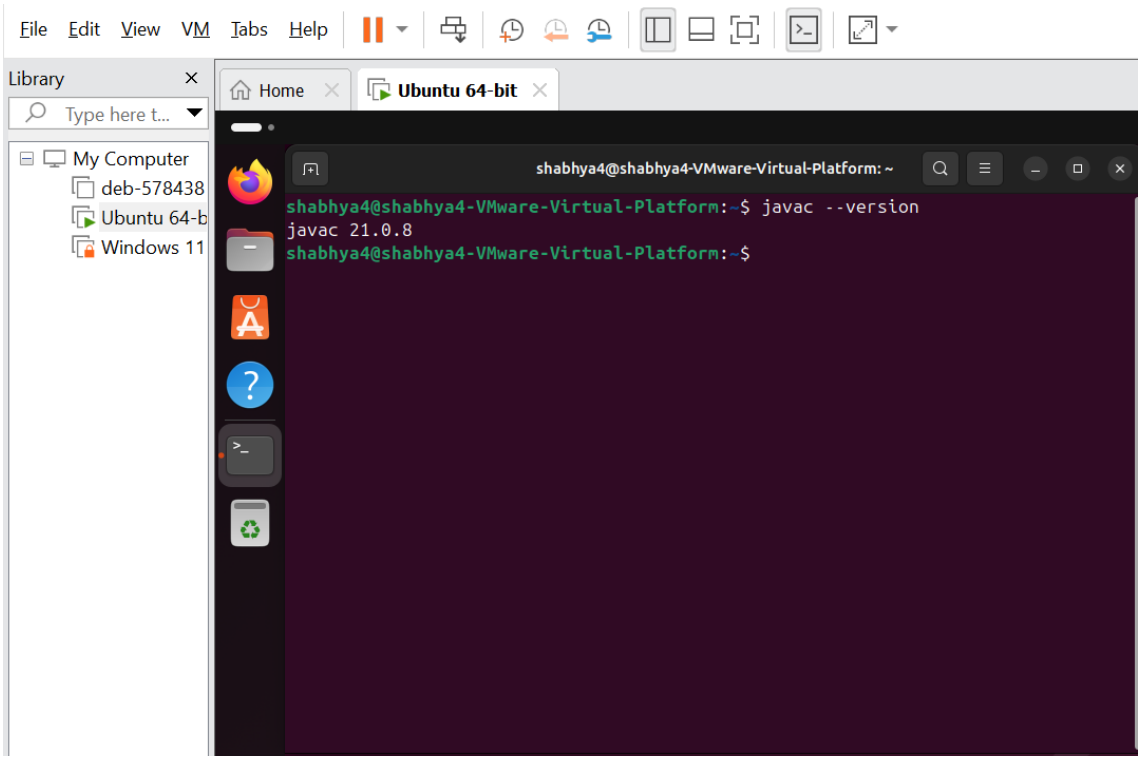
Screenshot of working assembly code of factorial calculation:



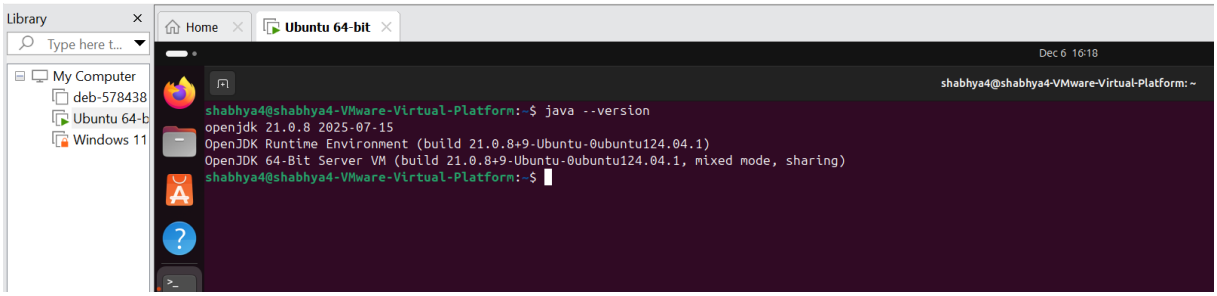
Assignment 4.2: Programming languages

Take screenshots that the following commands work:

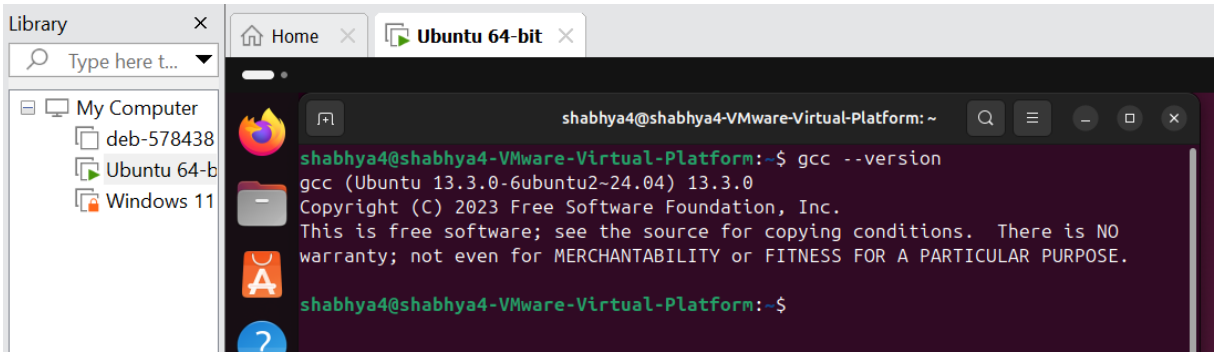
javac --version



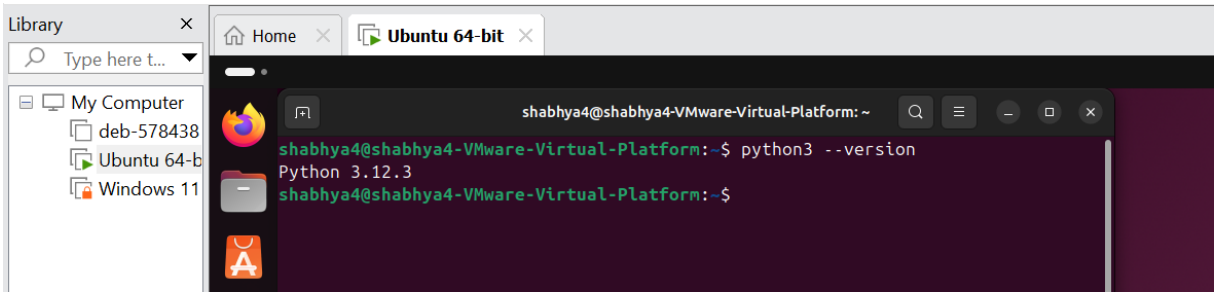
java --version



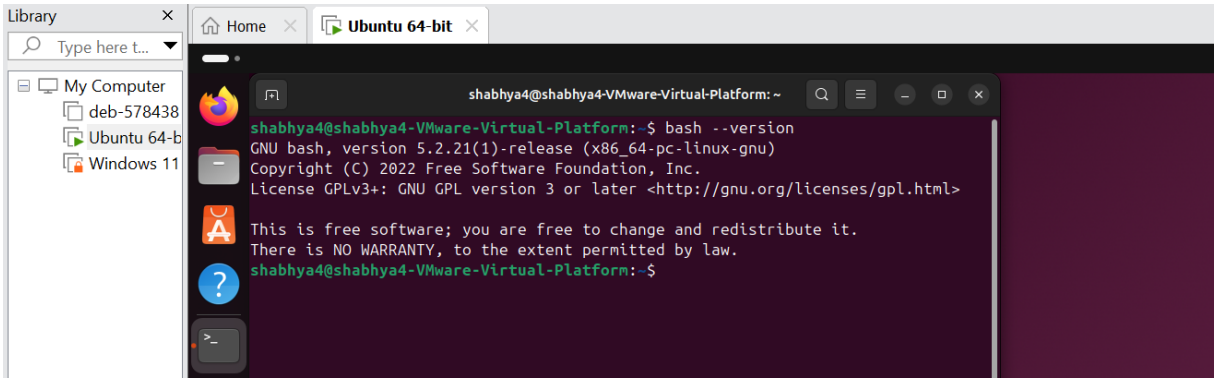
gcc --version



python3 --version



bash --version



Assignment 4.3: Compile

Which of the above files need to be compiled before you can run them?

- Fibonacci.java and fib.c

Which source code files are compiled into machine code and then directly executable by a processor?

- The C program fib.c, after compilation.

Which source code files are compiled to byte code?

- The Java program Fibonacci.java → Fibonacci.class.

Which source code files are interpreted by an interpreter?

- The Python program fib.py and the Bash script “fib.sh”.

These source code files will perform the same calculation after compilation/interpretation. Which one is expected to do the calculation the fastest?

- The compiled C program.

How do I run a Java program?

- Compile: javac Fibonacci.java
- Run:

How do I run a Python program?

- Python3 fib.py

How do I run a C program?

- Compile: gcc fib.c -o fib_c
- Run: ./fib_c

How do I run a Bash script?

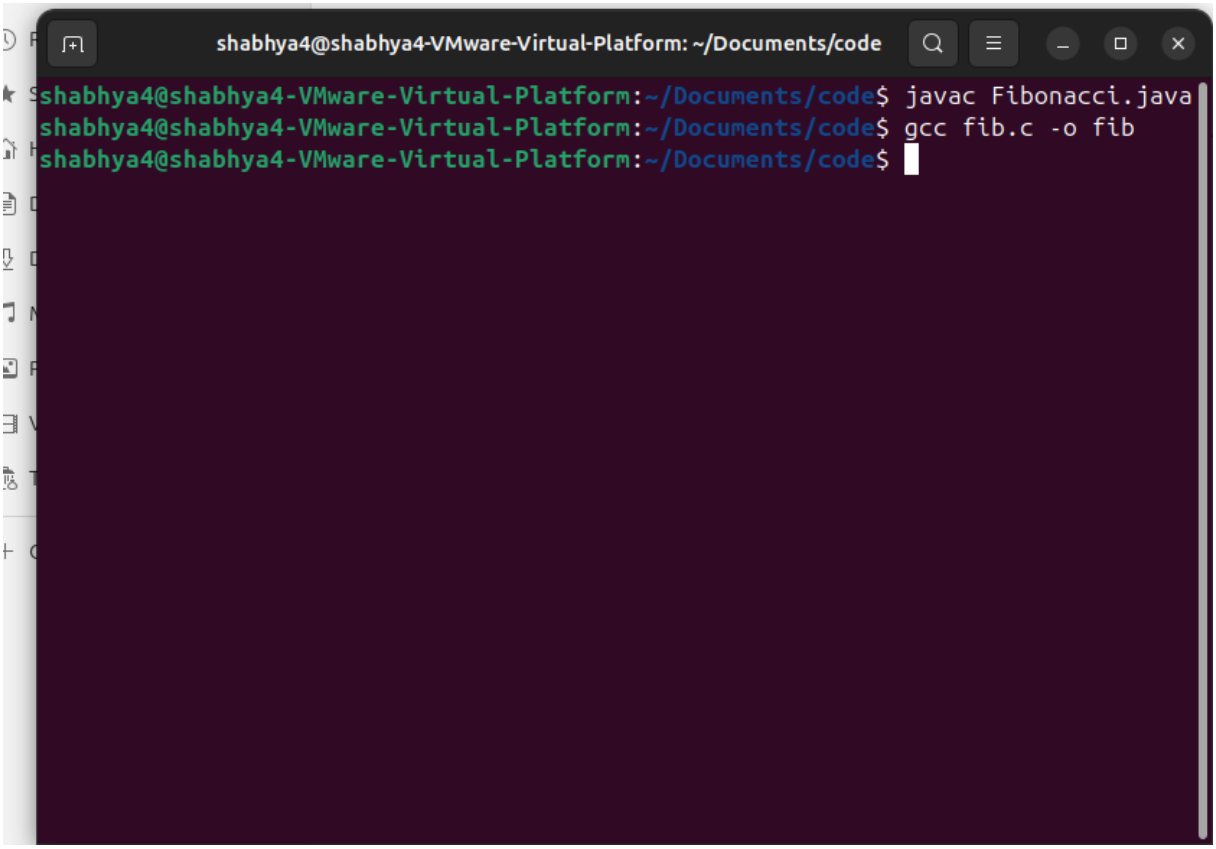
- Make executable: chmod a+x fib.sh
- Run: ./fib.sh (or bash fib.sh)

If I compile the above source code, will a new file be created? If so, which file?

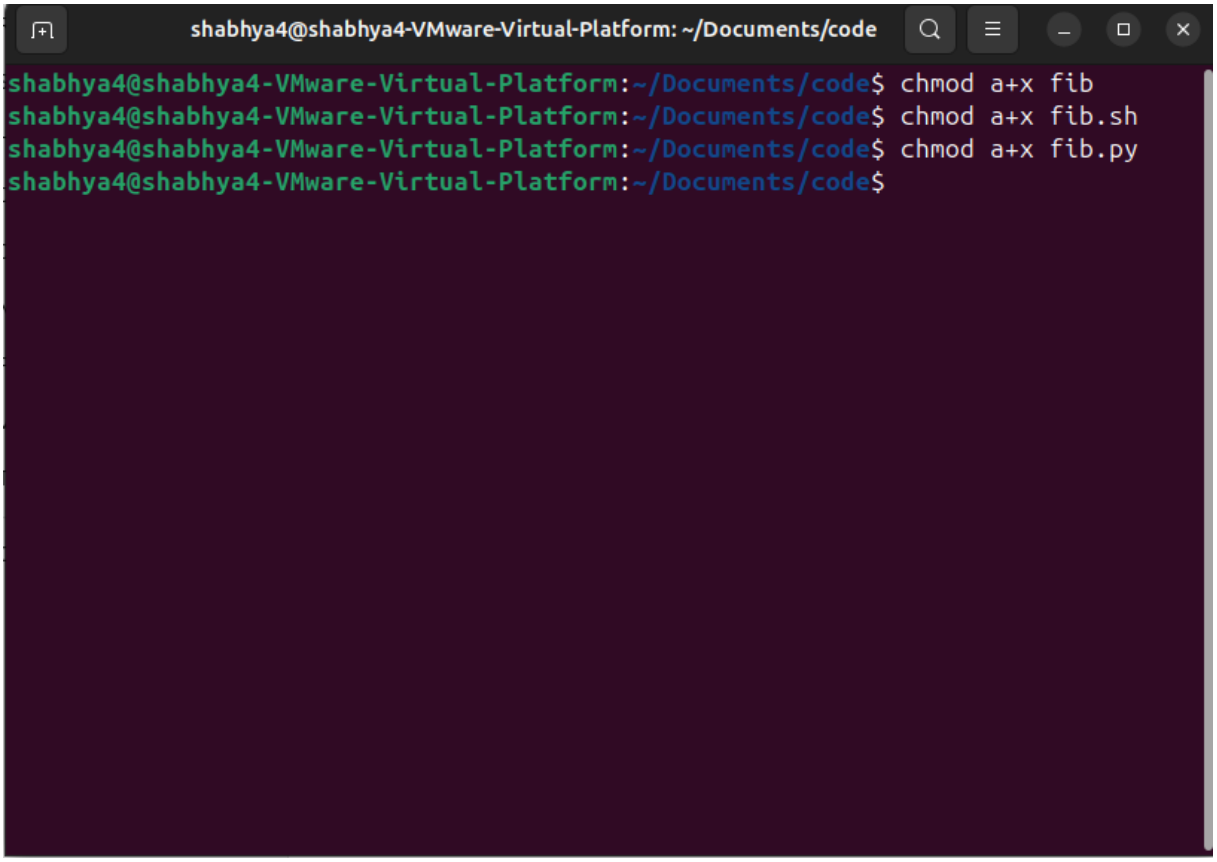
- Compiling Fibonacci.java creates Fibonacci. Class
- Compiling fib.c (e.g., gcc fib.c -o fib_c) creates the executable fib_

Take relevant screenshots of the following commands:

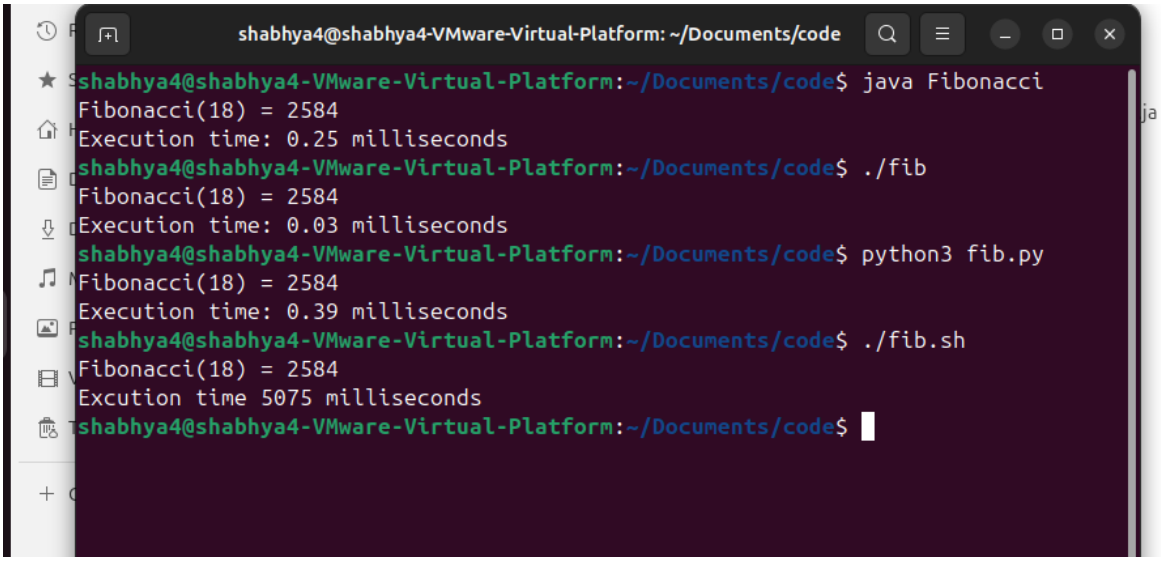
- Compile the source files where necessary



- Make them executable



- Run them



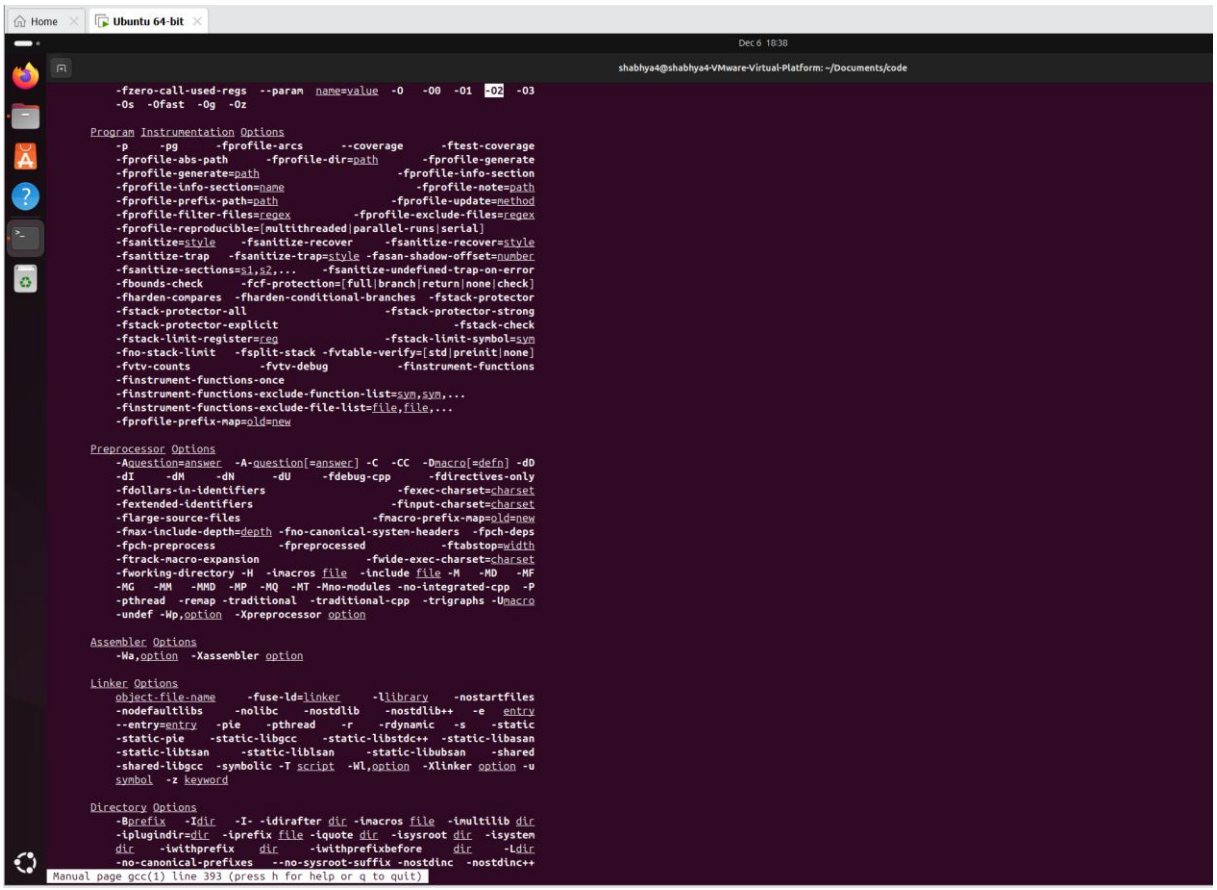
- Which (compiled) source code file performs the calculation the fastest?

  After comparing the run time C programming is fastest with execution time of 0.03 milliseconds.

Assignment 4.4: Optimize

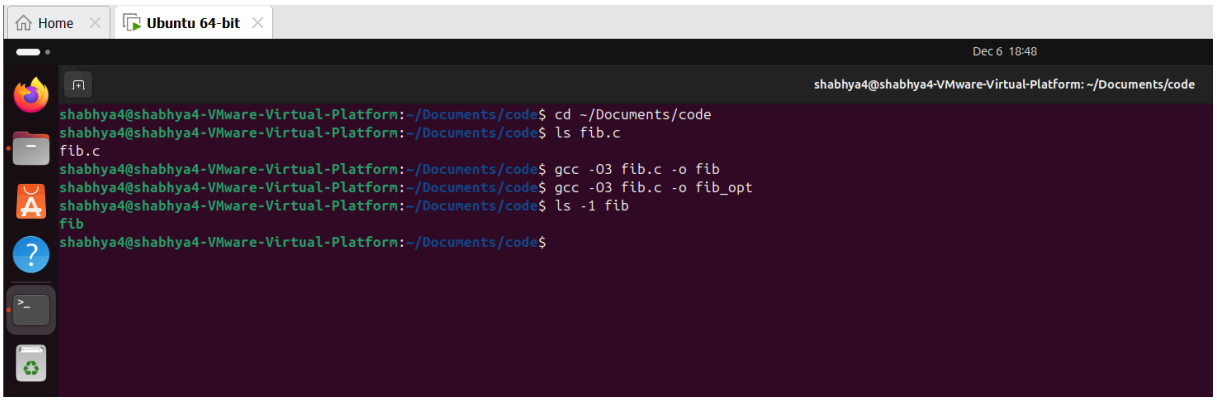
Take relevant screenshots of the following commands:

a) Figure out which parameters you need to pass to **the gcc** compiler so that the compiler performs a number of optimizations that will ensure that the compiled source code will run faster. **Tip!** The parameters are usually a letter followed by a number. Also read **page 191** of your book, but find a better optimization in the man pages. Please note that Linux is case sensitive.

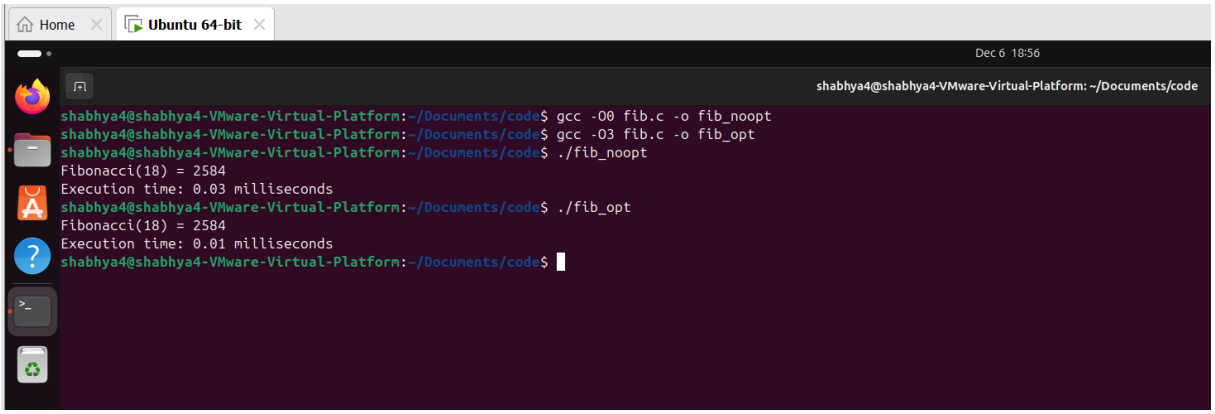


I used the gcc optimization flag -O3, found in the Optimization Options section of man gcc.

b) Compile **fib.c** again with the optimization parameters

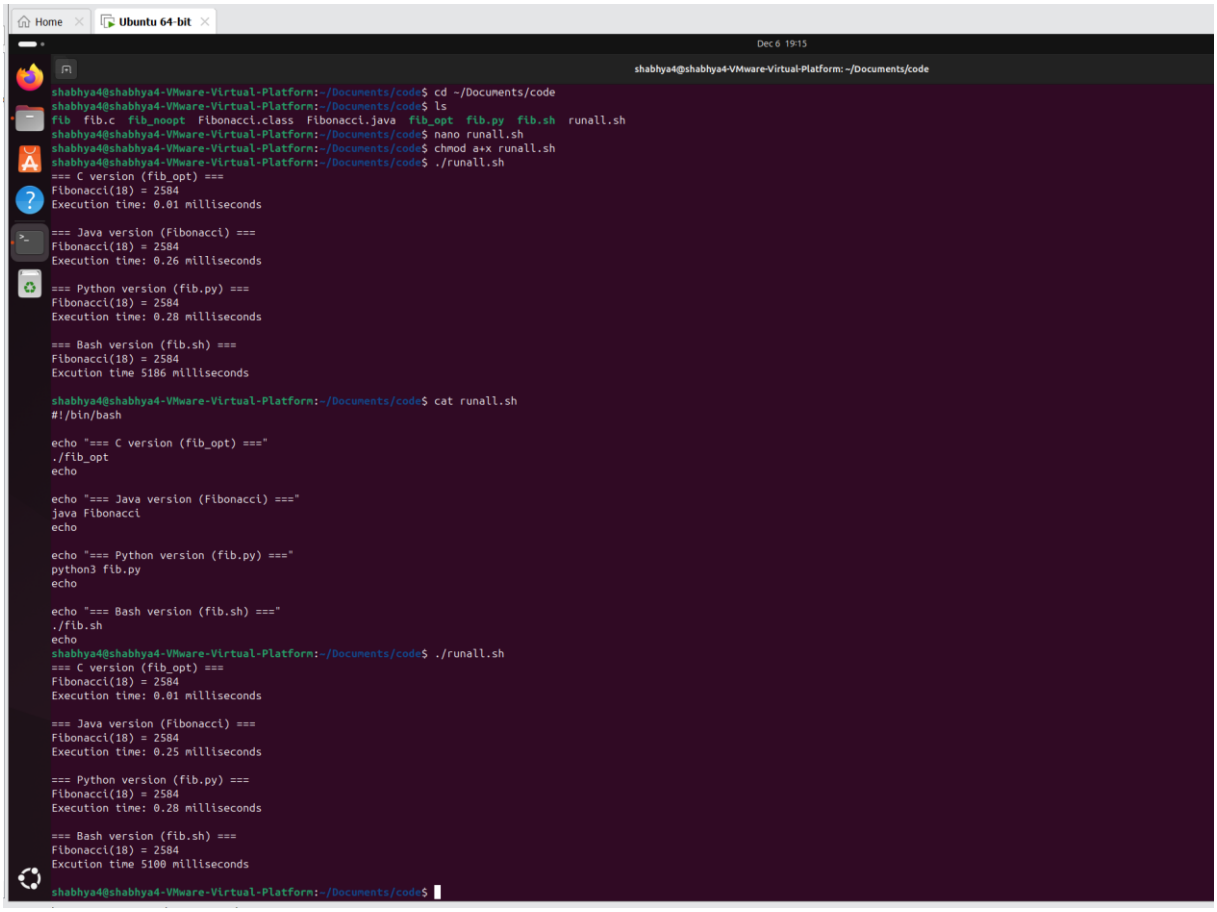


c) Run the newly compiled program. Is it true that it now performs the calculation faster?



I compiled the C program without optimizations (-O0) and with optimizations (-O3).
The unoptimized version took about 0.03 ms; the optimized one took about 0.01 ms.
So the optimized program does run faster.

d) Edit the file **runall.sh**, so you can perform all four calculations in a row using this Bash script. So the (compiled/interpreted) C, Java, Python and Bash versions of Fibonacci one after the other.



Assignment 4.5: More ARM Assembly

Like the factorial example, you can also implement the calculation of a power of 2 in assembly. For example you want to calculate $2^4 = 16$. Use iteration to calculate the result. Store the result in r0.

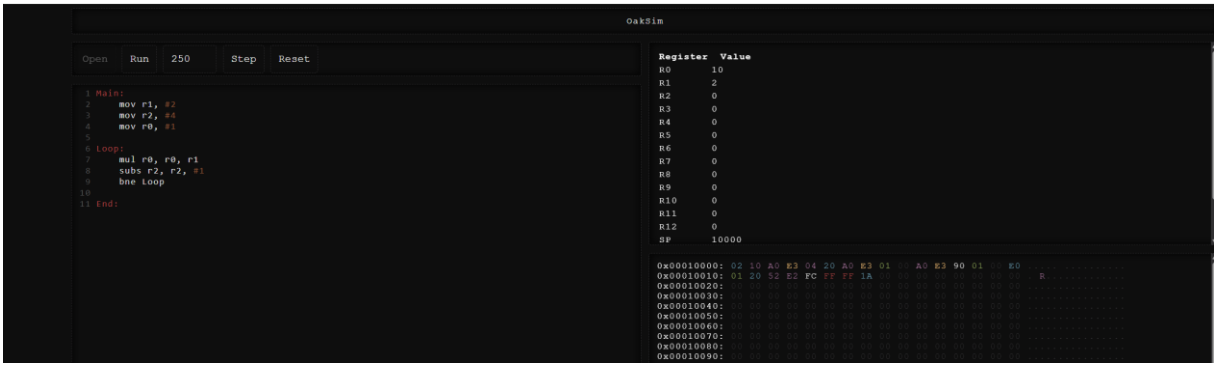
```
Main:
mov r1, #2
mov r2, #4
mov r0, #1

Loop:
    mul r0, r0, r1
    subs r2, r2, #1
    bne Loop

End:
```

Complete the code. See the PowerPoint slides of week 4.

Screenshot of the completed code here.



Ready? Save this file and export it as a pdf file with the name: **week4.pdf**