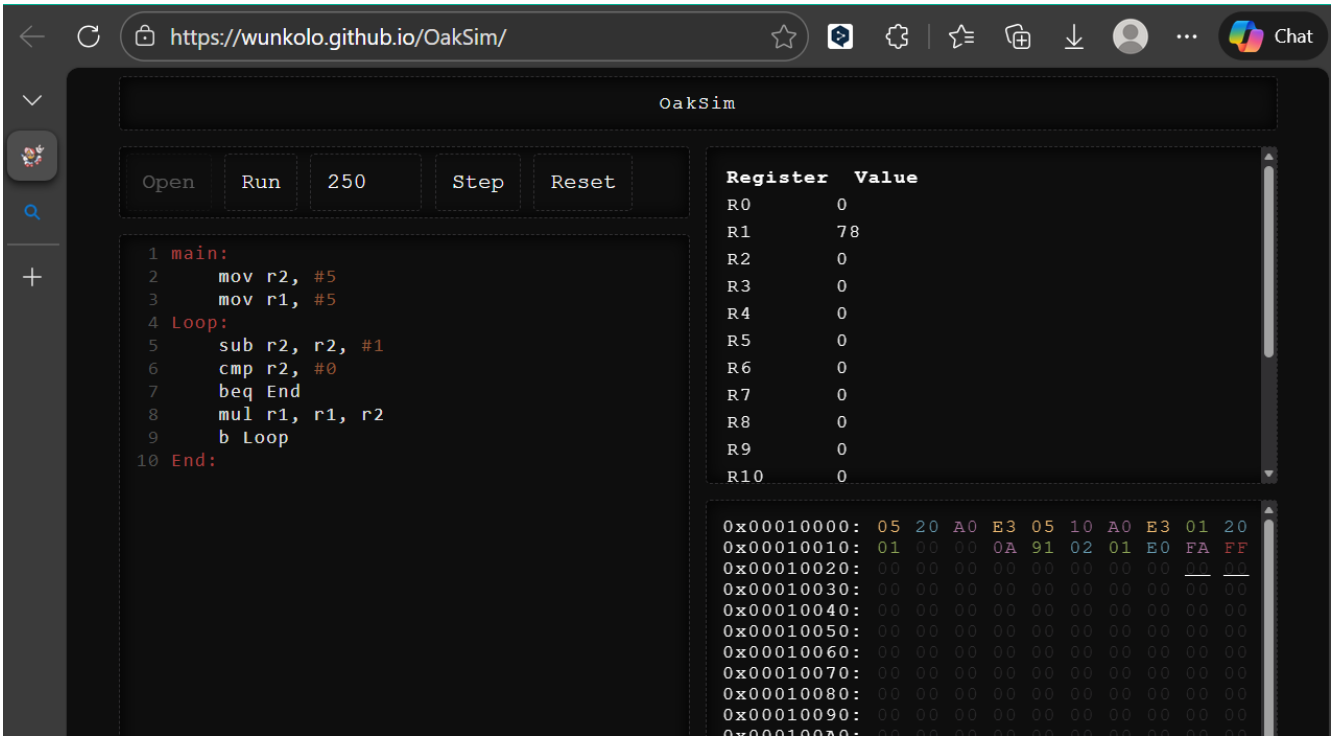
Template Week 4 – Software

Student number: 585902

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:

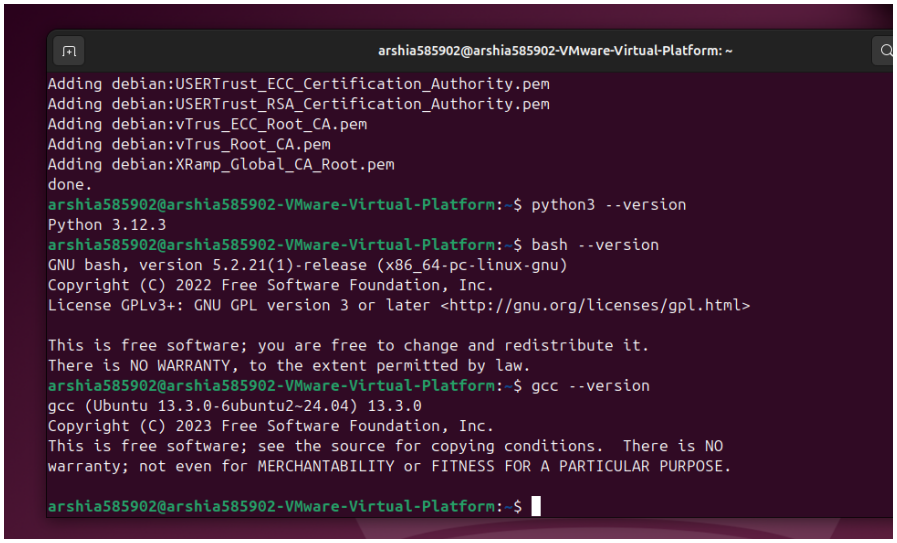
javac --version

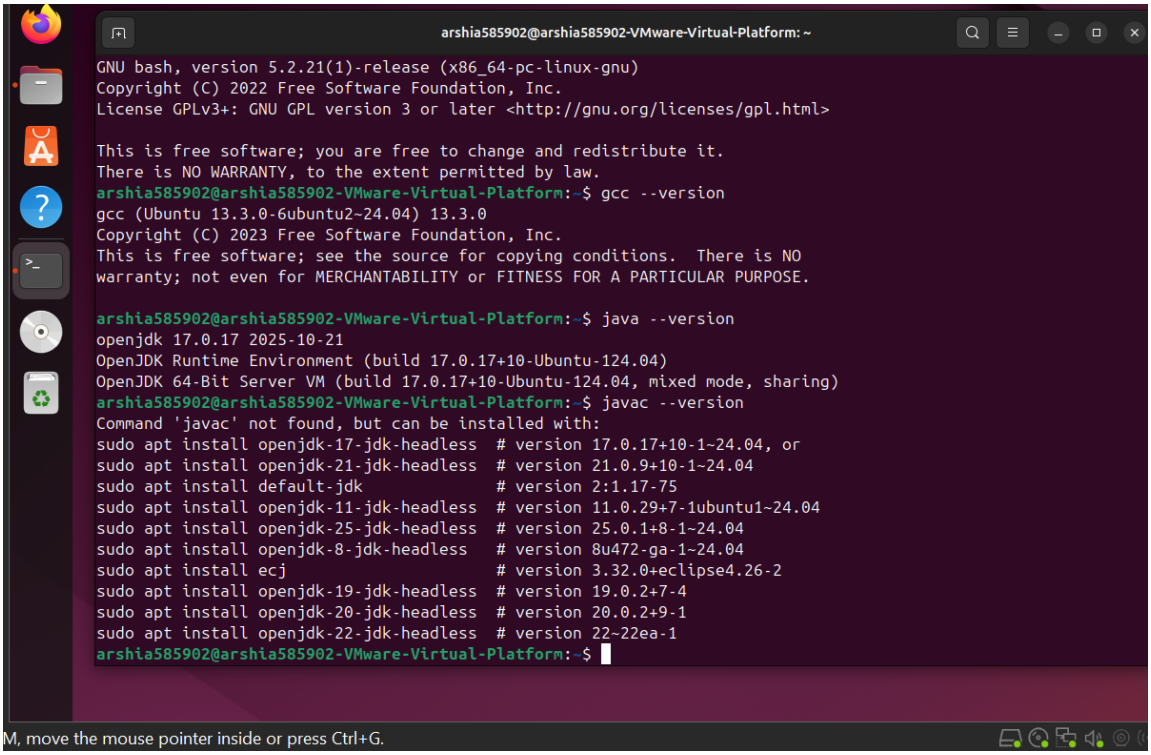
java --version

gcc --version

python3 --version

bash --version





Assignment 4.3: Compile

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

The files in C and Java language need to be compiled first

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

The file in C language is compiled to native machine code

Which source code files are compiled to byte code?

The file in Java is first compiled to bytecode and then run by Java virtual machine

Which source code files are interpreted by an interpreter?

The files in python and bash are executed line by line by a runtime interpreter

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

C because after being compiled into machine code it directly communicates with the CPU. Then comes Java and after that are python and bash since they need to be interpreted line by line; with bash being even slower with the extra barrier of the shell in between

How do I run a Java program?

Using the javac compiler I first create a ‘main’ class through a bytecode file like this: javac Main.java and then I run that main as such: java Main(referring to the name of the newly created class file) since its not directly executed by the os afterwards

How do I run a Python program?

I call python3 interpreter on it. It does make a bytecode like in java, but it doesn’t run manually because that byte code is handled in the cache rather than being visible by the user so although no “visible” additional file is created, but it still takes longer to run

How do I run a C program?

> Using the gcc compiler I create a compiled version of the file which is basically a separate file which then I can store and run anywhere I want and it goes like:
> gcc ./example.c -o customName

How do I run a Bash script?

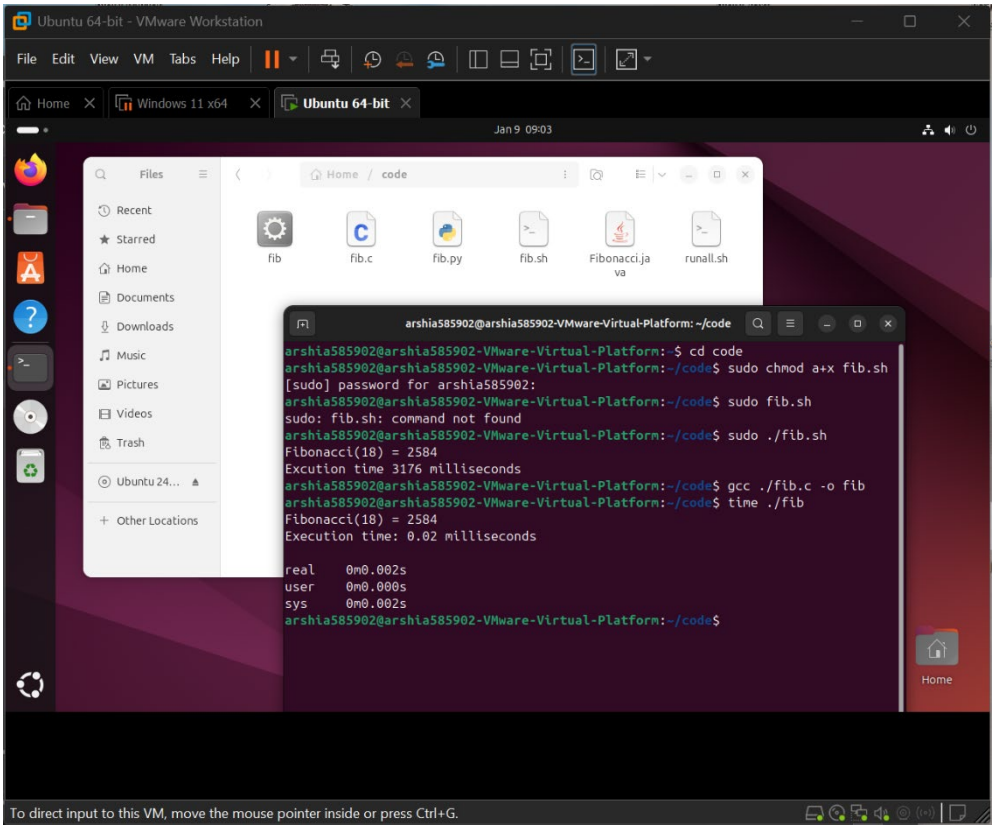
> I use bash runtime for it as such: bash name (this way its run by directlu invoking bash interpreter and passing the script to it as an argument). In this approach the script doesn't need execute permissions because bash explicitly interprets it
> Alternatively, it can be run as an executable file. after making sure that the file has execute permissions using 'chmod +x', I can then use a shebang like so #!/bin/bash to specify the shell

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

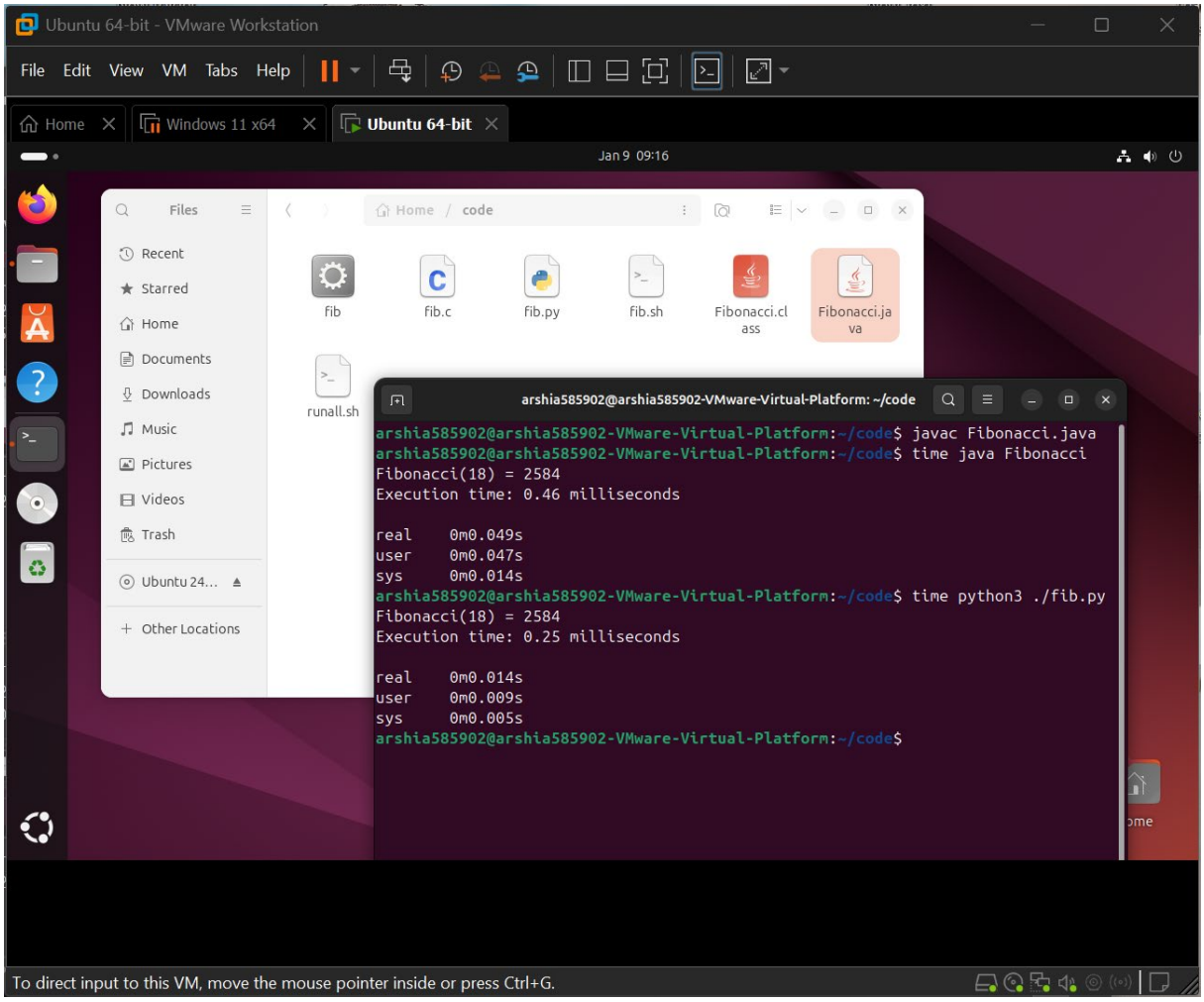
> For compiled ones like java and c yes but for the interpreted ones such as py and bash no. for c that file is a direct machine code written in hex groups for java it's the main class which appears in the directory as soon as javac compiler compiles it

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? C



Comparison between Bash and C regarding time and run method (slowest vs Fastest)



Compariton between Java and Python regarding the method to run and the time it takes

As the times show, the C file was the fastest since it turned into machine code upon being compiled, and unlike bash it didn't have an extra shell overhead. Unlike java it didn't have to create an additional class to run

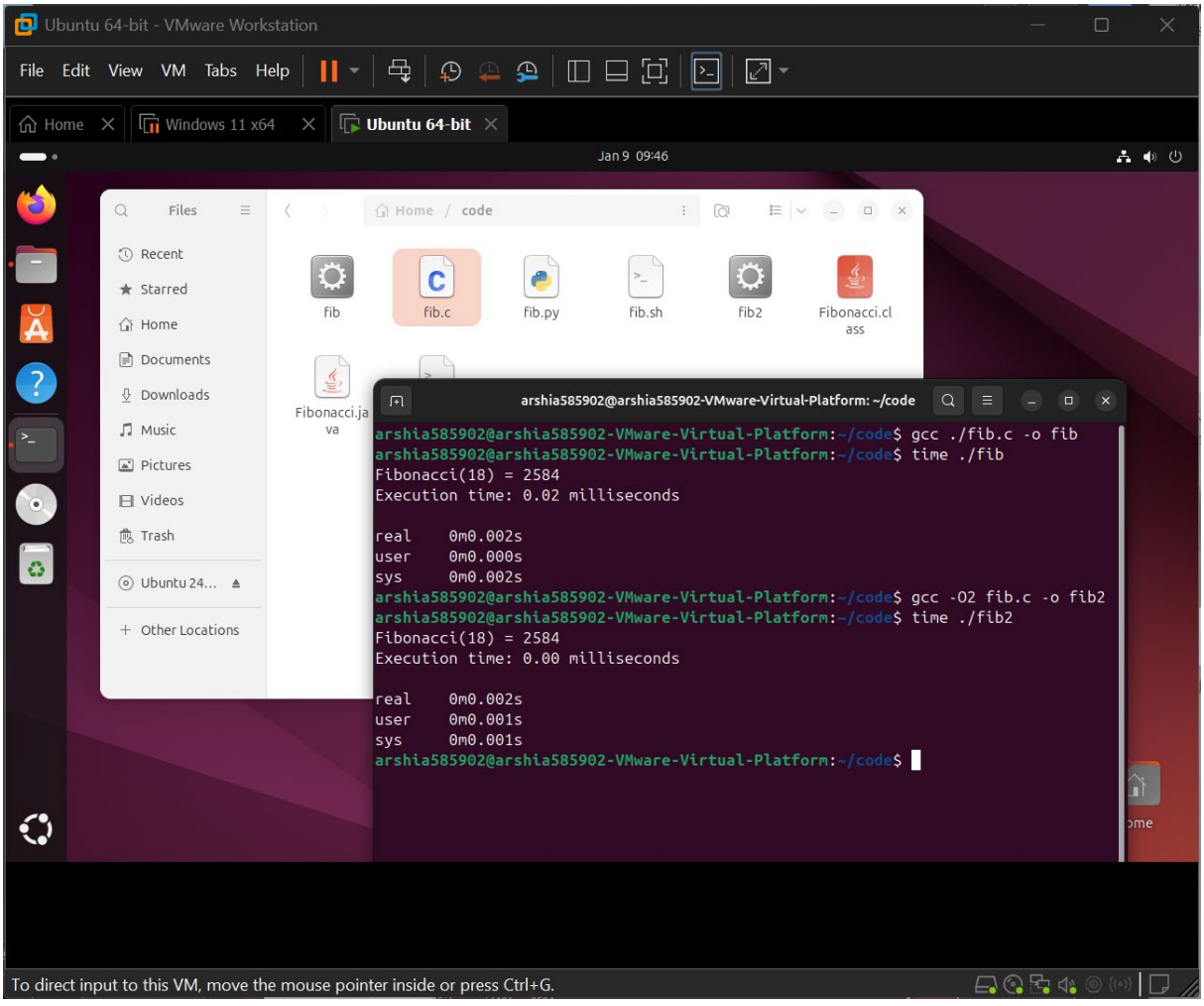
Assignment 4.4: Optimize

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.

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

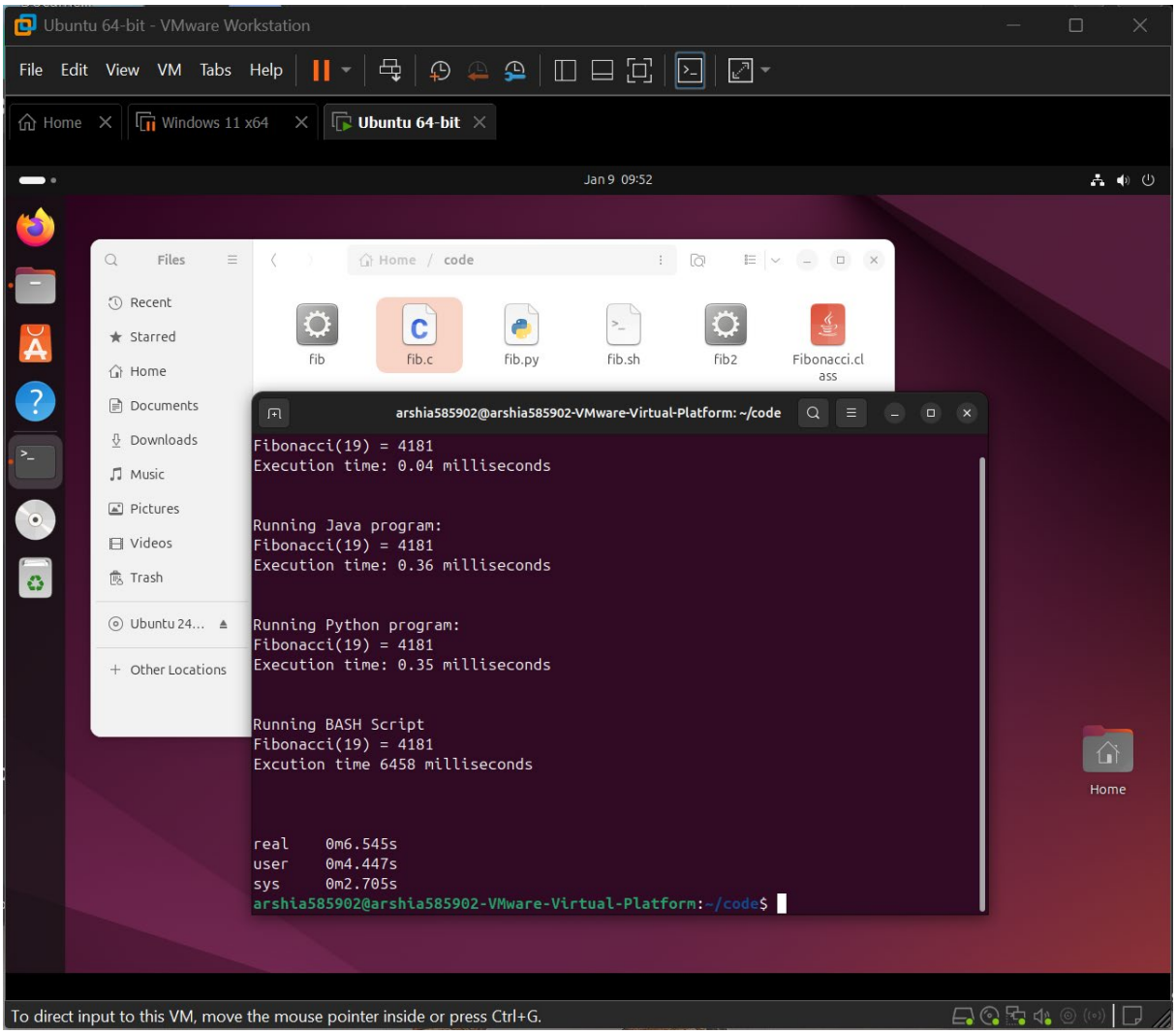
c) Run the newly compiled program. Is it true that it now performs the calculation faster?



Explanation bellow :

In this test I compiled it twice with two different names since that's a feature of c compiler which lets the creation of individual compiled files. I measured their run time in the same window so that by comparison it would be apparent that the optimized version now runs even quicker. Here I used -O2 optimization option; however, There are other optimization options as well like O3 but its not always guaranteed to be quicker

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.

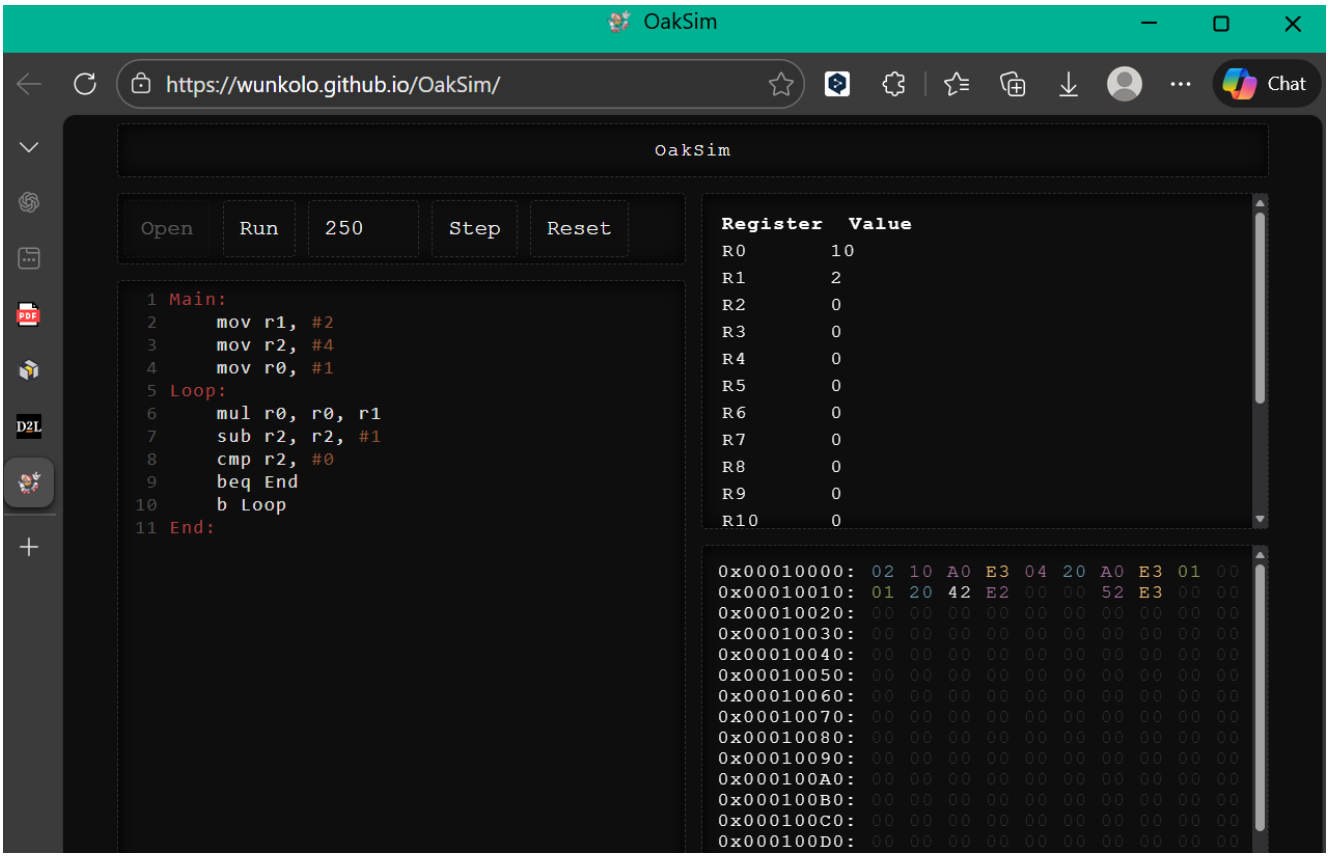
```
Main:
mov r1, #2
mov r2, #4

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**