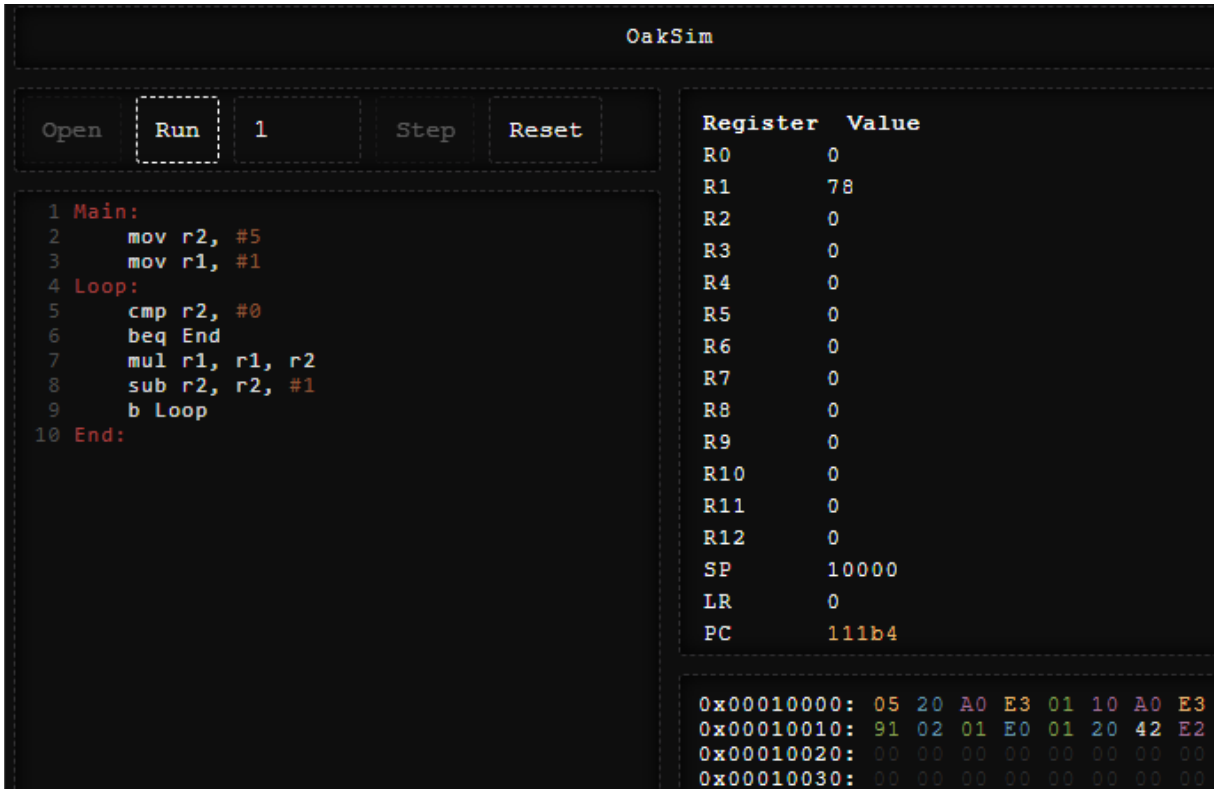
Template Week 4 – Software

Student number: 560830

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:

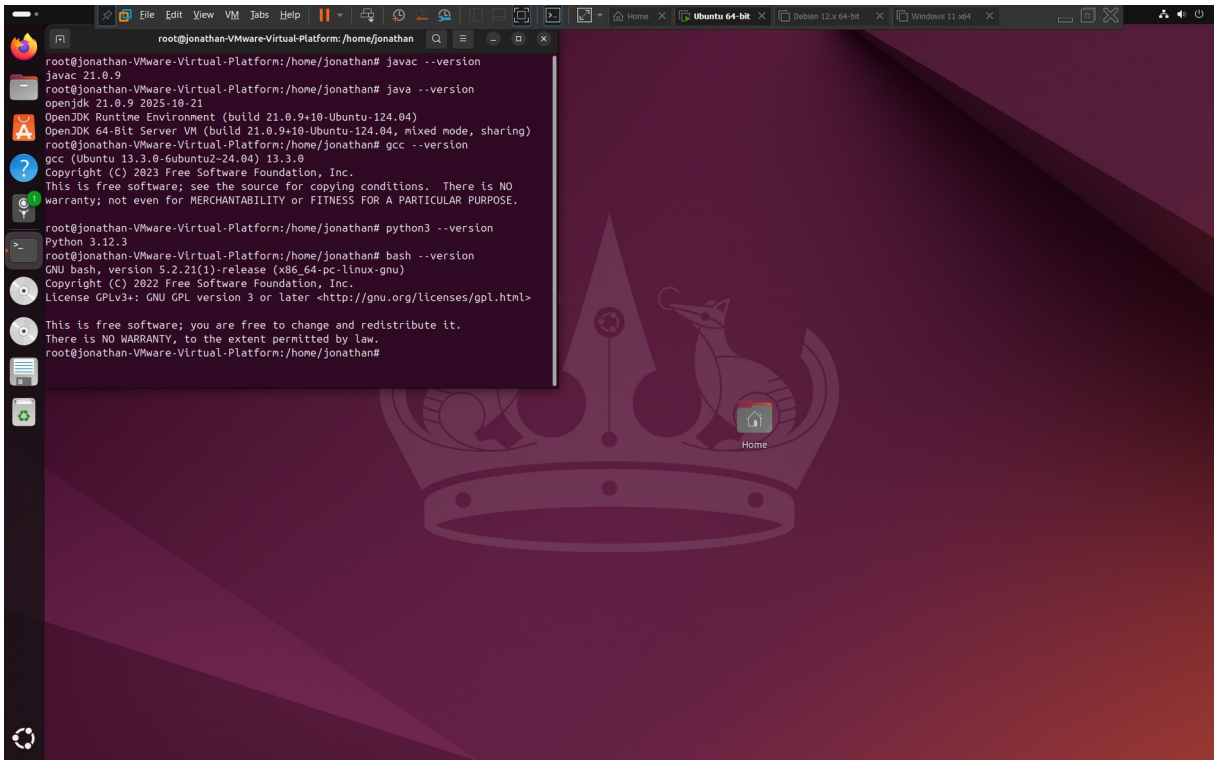
javac --version

java --version

gcc --version

python3 --version

bash --version


File Edit View VM Tabs Help
Home
Ubuntu 64-bit
Debian 12.x 64-bit
Windows 11 x64
root@jonathan-VMware-Virtual-Platform: /home/jonathan
root@jonathan-VMware-Virtual-Platform:/home/jonathan# javac --version
javac 21.0.9
root@jonathan-VMware-Virtual-Platform:/home/jonathan# java --version
openjdk 21.0.9 2025-10-21
OpenJDK Runtime Environment (build 21.0.9+10-Ubuntu-124.04)
OpenJDK 64-Bit Server VM (build 21.0.9+10-Ubuntu-124.04, mixed mode, sharing)
root@jonathan-VMware-Virtual-Platform:/home/jonathan# gcc --version
gcc (Ubuntu 13.3.0-6ubuntu2~24.04) 13.3.0
Copyright (C) 2023 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
root@jonathan-VMware-Virtual-Platform:/home/jonathan# python3 --version
Python 3.12.3
root@jonathan-VMware-Virtual-Platform:/home/jonathan# bash --version
GNU bash, version 5.2.21(1)-release (x86_64-pc-linux-gnu)
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
root@jonathan-VMware-Virtual-Platform:/home/jonathan#
Home

Assignment 4.3: Compile

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

- The C file has to be compiled to **machine code**
- The Java file has to be compiled into **byte code** and ran on a JVM
- The python file is **interpreted**
- The bash file does not have to be compiled, just made executable

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

- The C file by gcc

Which source code files are compiled to byte code?

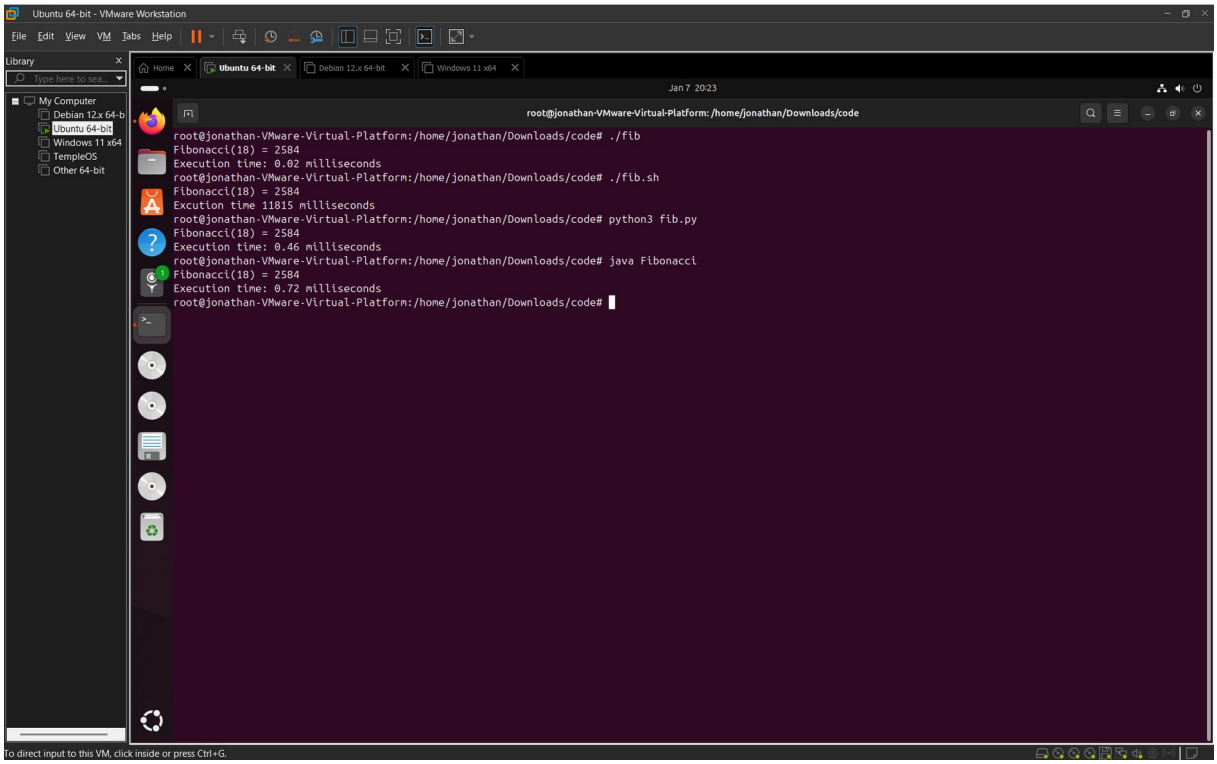
- The java file by javac (JavaCompiler)

Which source code files are interpreted by an interpreter?

- The python file, by the python interpreter.
- The Bash file by the bash shell.

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

- The fib.c file, as C is executed by the CPU directly.

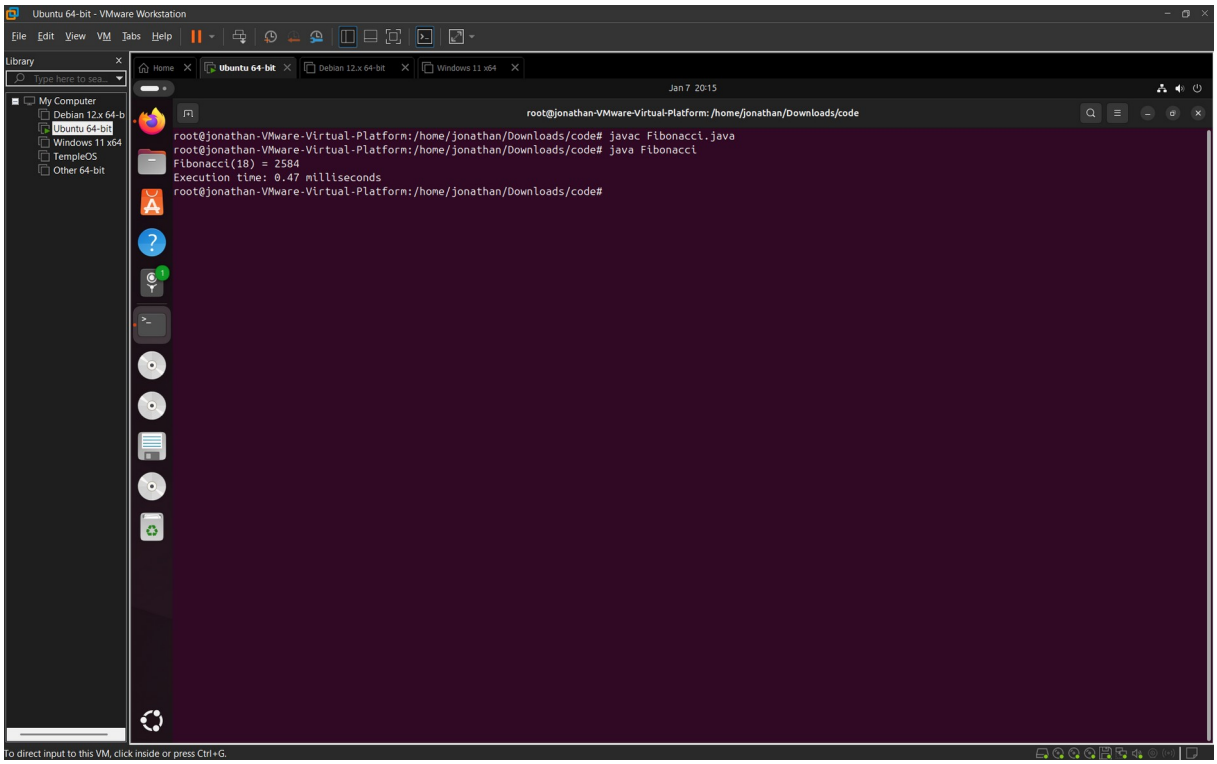


This image shows that the C file, by far, is the fastest. Surprisingly the bash file is horribly slow, clocking in at 11815ms or 11 seconds.

How do I run a Java program?

1. Compile it using **javac Fibonacci.java**

2. Execute it using **java Fibonacci**



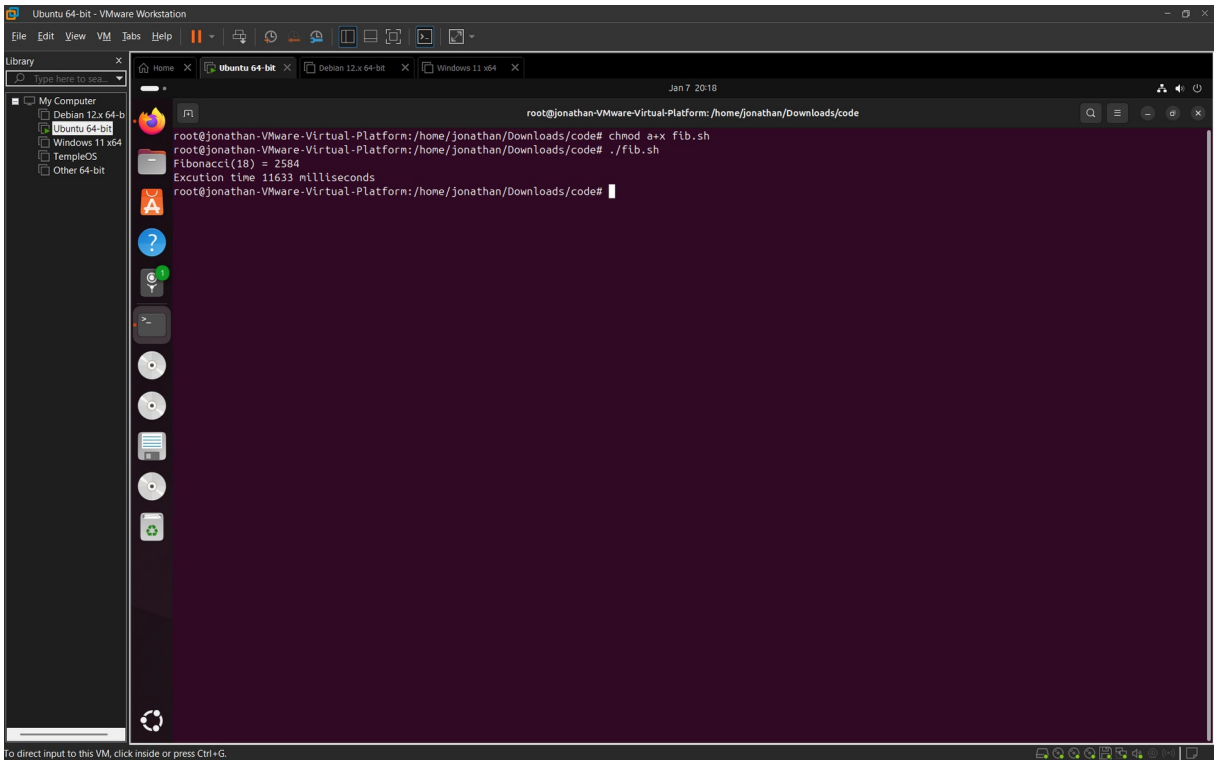
How do I run a Python program?

How do I run a C program?

1. Compile it using the GCC: **gcc -o fib fib.c**
2. Run it using **./fib**

How do I run a Bash script?

1. Compile it using **sudo chmod a+x fib.sh**
2. Run it using **sudo ./fib.sh**



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

- A java class file: “**Fibonacci.class**”
- A C executable called “**fib**”

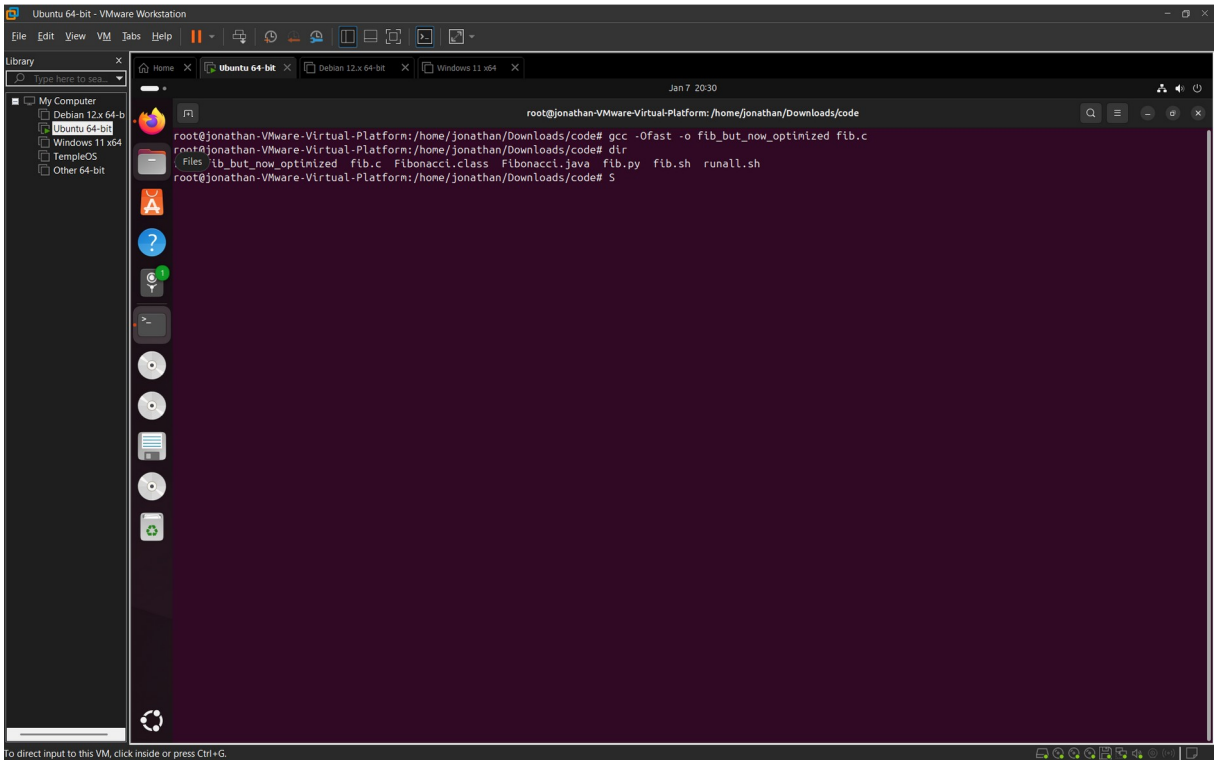
Take relevant screenshots of the following commands:

- Compile the source files where necessary
- Make them executable
- Run them
- Which (compiled) source code file performs the calculation the fastest?

Assignment 4.4: Optimize

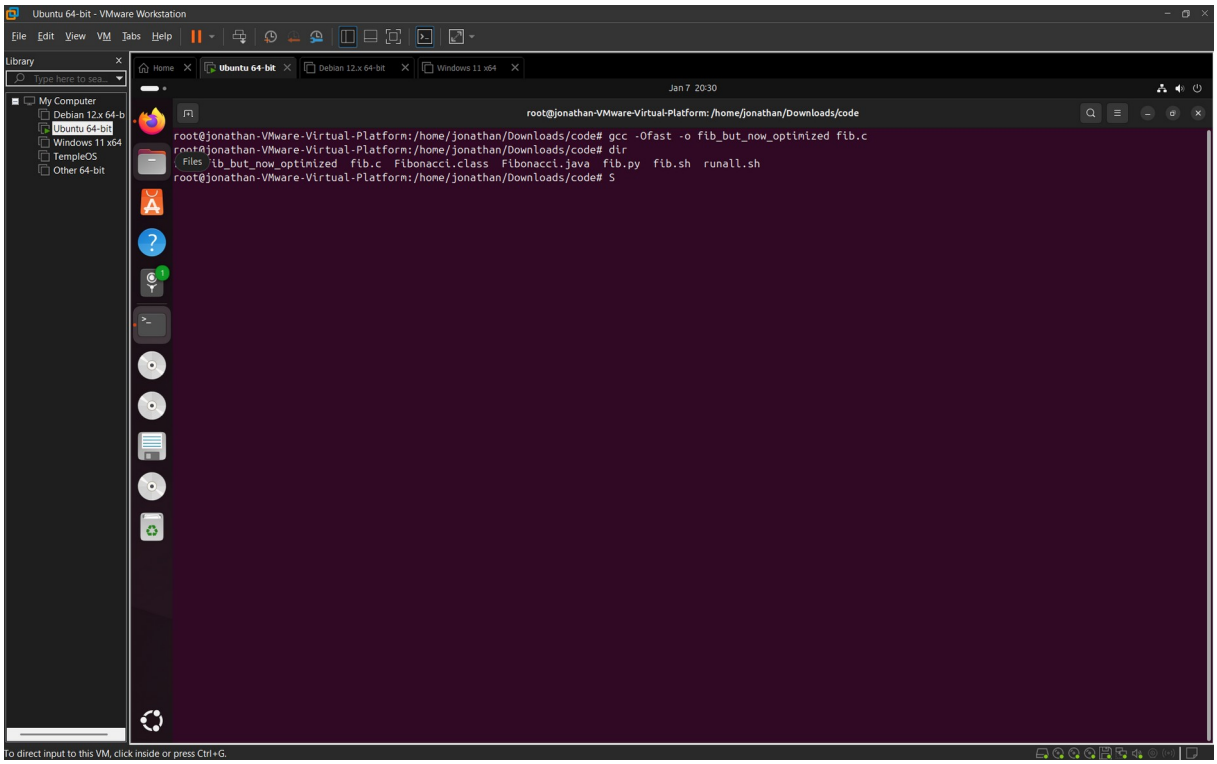
Take relevant screenshots of the following commands:

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.

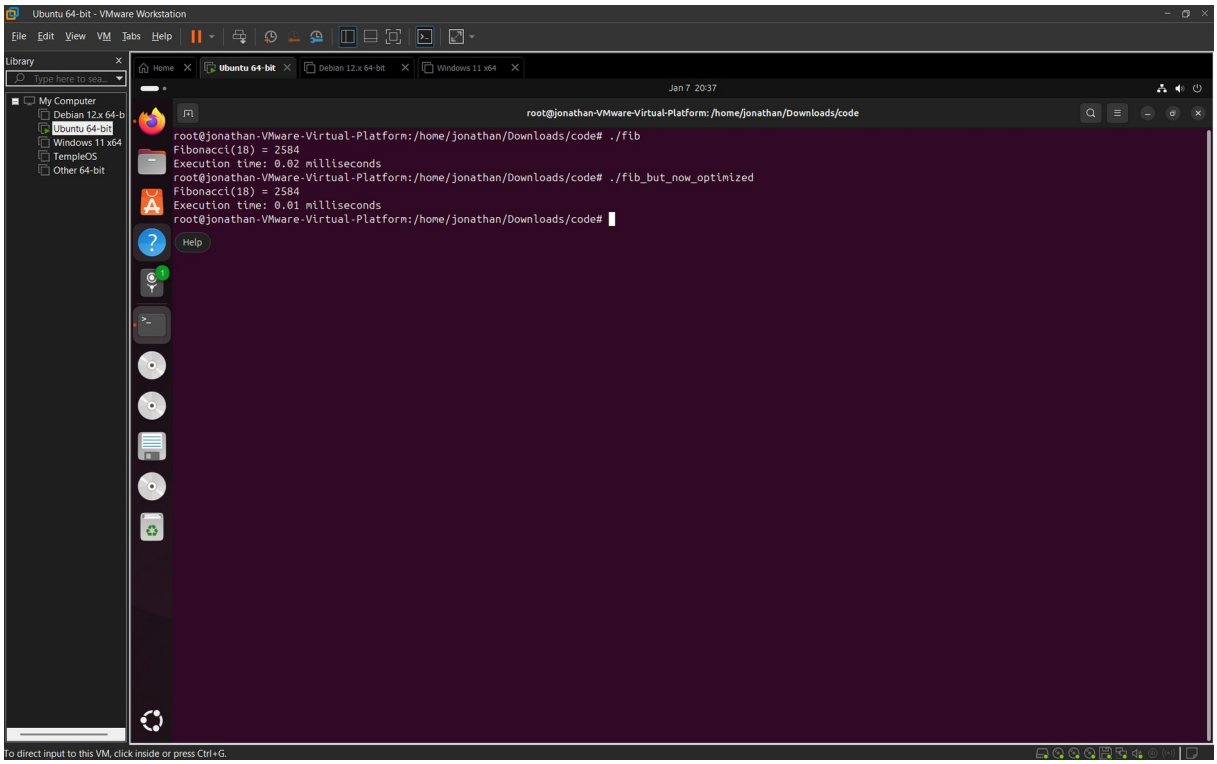


According to the manual the "-Ofast" flag is the most aggressive optimization, with -O3 being the most aggressive "safe" one. For a small program such as this, using Ofast shouldn't cause any issues.

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

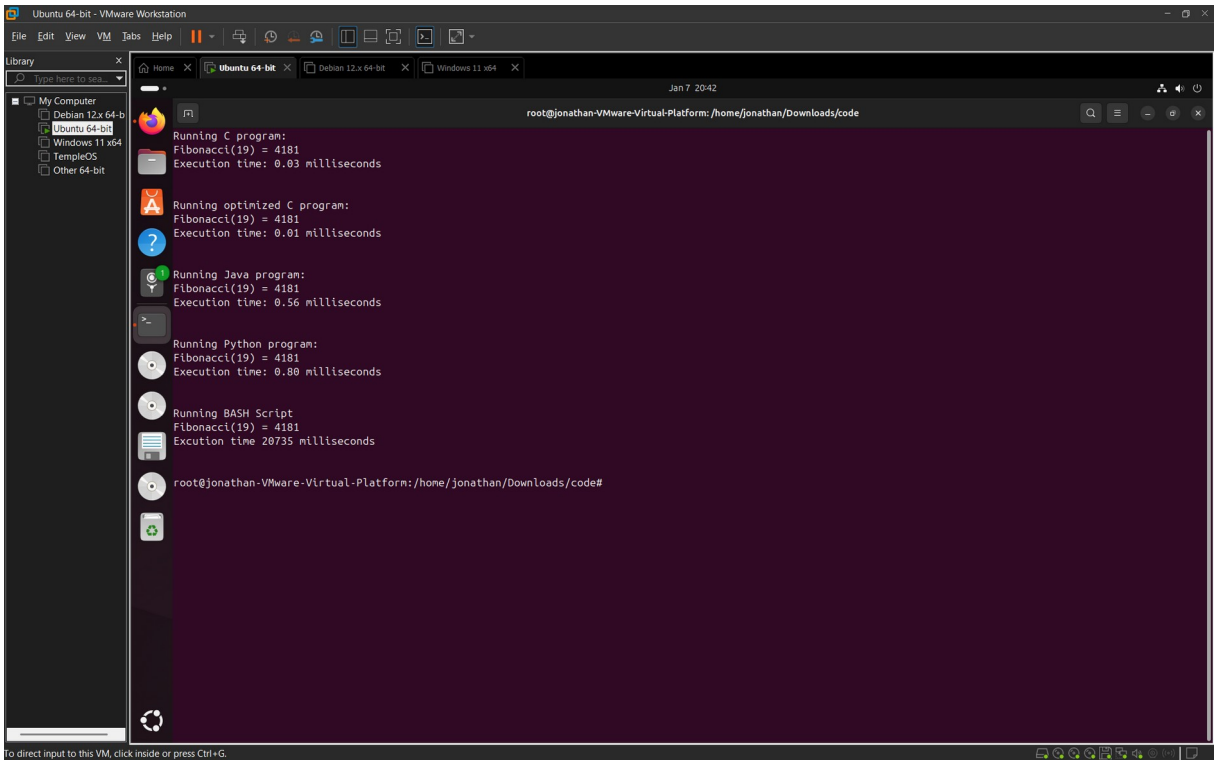


c) Run the newly compiled program. Is it true that it now performs the calculation faster?



An incredible 100% increase in speed!

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.



Somehow the Fibonacci.sh got ..slower?

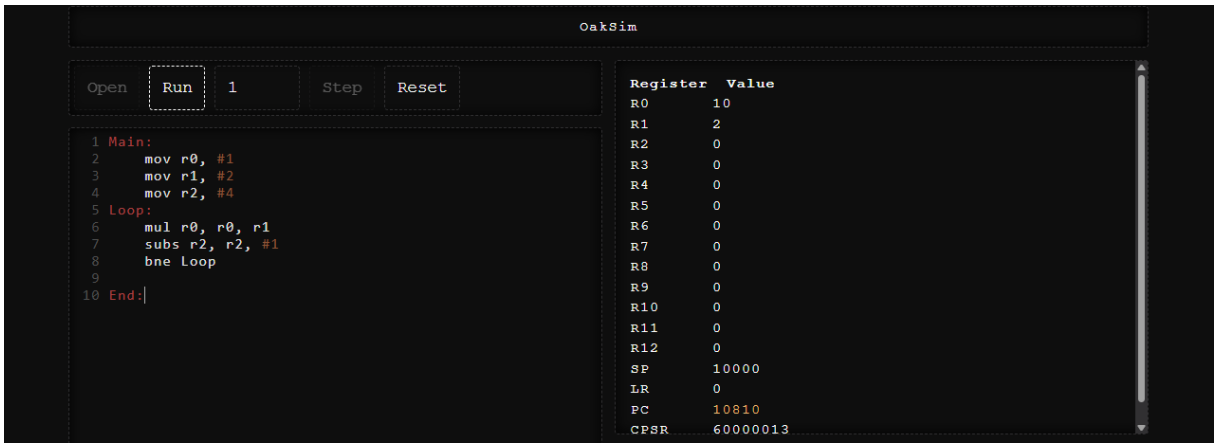
Assignment 4.5: More ARM Assembly

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 r0, #1
        mov r1, #2
        mov r2, #4
Loop:
        mul r0, r0, r1
        subs r2, r2, #1
        bne Loop

End:
```

Screenshot of the completed code here.


OakSim
Open
Run
1
Step
Reset
1 Main:
2 mov r0, #1
3 mov r1, #2
4 mov r2, #4
5 Loop:
6 mul r0, r0, r1
7 subs r2, r2, #1
8 bne Loop
9
10 End:
Register Value
R0 10
R1 2
R2 0
R3 0
R4 0
R5 0
R6 0
R7 0
R8 0
R9 0
R10 0
R11 0
R12 0
SP 10000
LR 0
PC 10810
CPSR 60000013