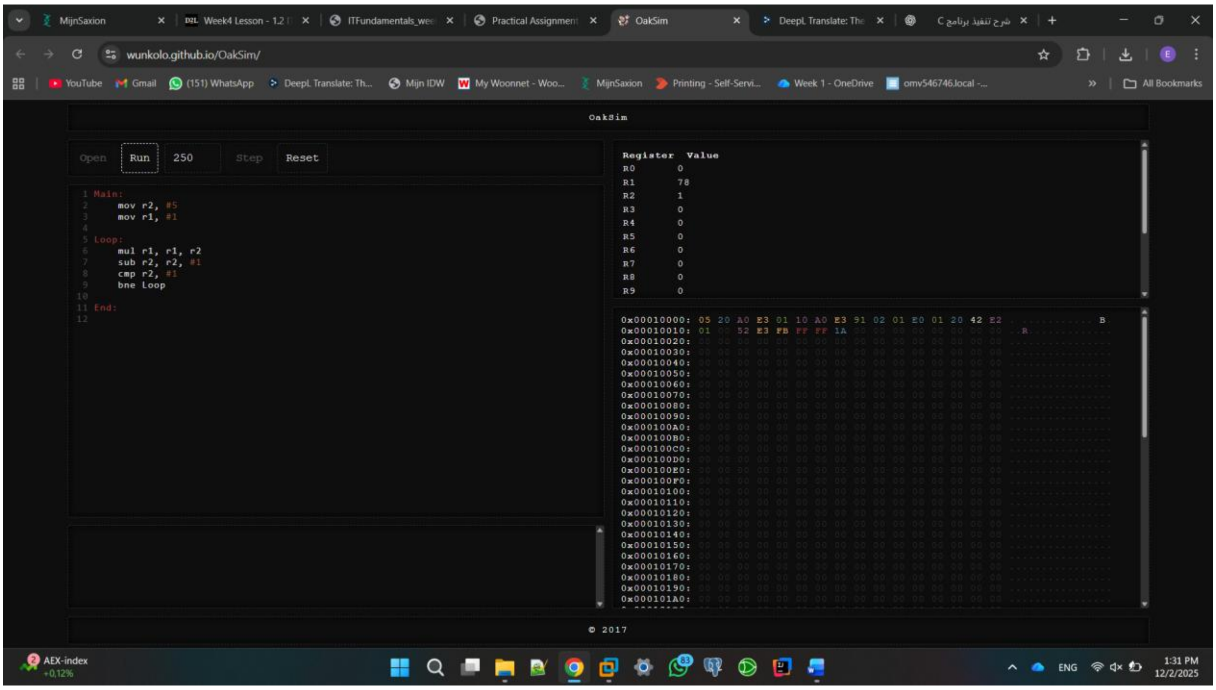
Template Week 4 – Software

Student number: 546746

Assignment 4.1: ARM assembly

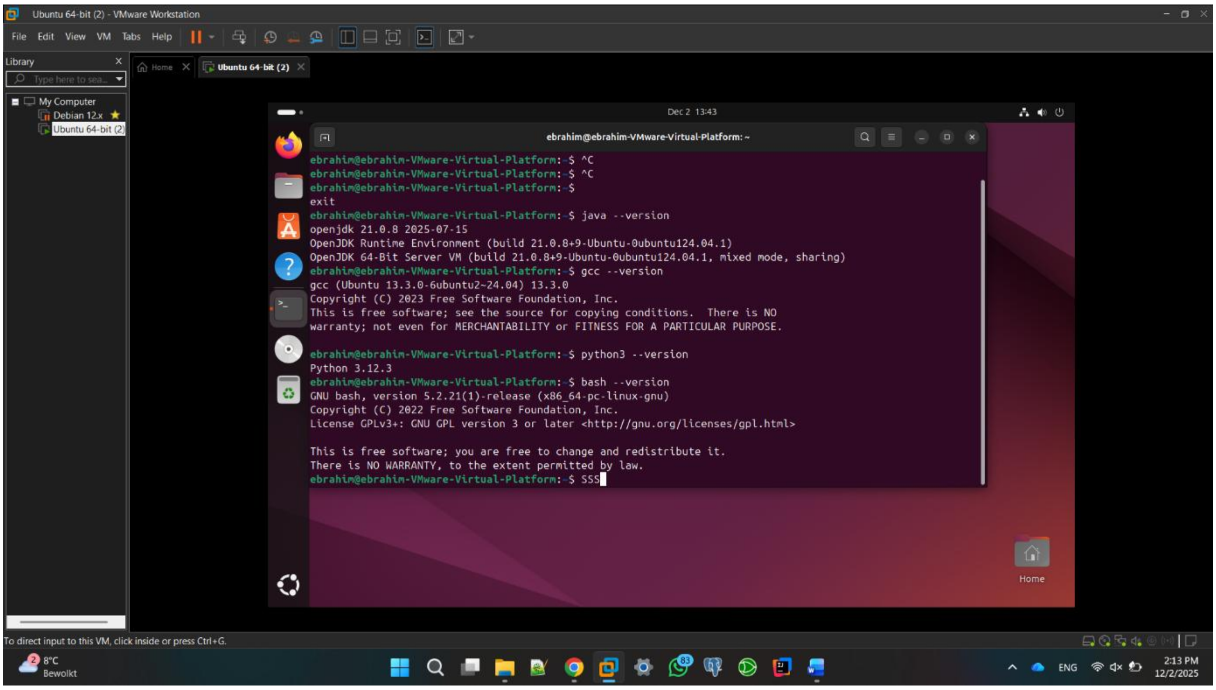
Screenshot of working assembly code of factorial calculation:



Assignment 4.2: Programming languages

Take screenshots that the following commands work:

- **Java compiler (javac):** 21.0.8
- **Java runtime (java):** OpenJDK 21.0.8
- **GCC (C/C++ compiler):** 13.3.0
- **Python interpreter (python3):** 3.12.3
- **Bash shell:** 5.2.21


Ubuntu 64-bit (2) - VMware Workstation
ebrahim@ebrahim-VMware-Virtual-Platform: ~
ebrahim@ebrahim-VMware-Virtual-Platform:~$ ^C
ebrahim@ebrahim-VMware-Virtual-Platform:~$ ^C
ebrahim@ebrahim-VMware-Virtual-Platform:~$
exit
ebrahim@ebrahim-VMware-Virtual-Platform:~$ java --version
openjdk 21.0.8 2025-07-15
OpenJDK Runtime Environment (build 21.0.8+9-Ubuntu-0ubuntu124.04.1)
OpenJDK 64-Bit Server VM (build 21.0.8+9-Ubuntu-0ubuntu124.04.1, mixed mode, sharing)
ebrahim@ebrahim-VMware-Virtual-Platform:~$ 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.
ebrahim@ebrahim-VMware-Virtual-Platform:~$ python3 --version
Python 3.12.3
ebrahim@ebrahim-VMware-Virtual-Platform:~$ 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.
ebrahim@ebrahim-VMware-Virtual-Platform:~$ SSS
To direct input to this VM, click inside or press Ctrl+G.

Assignment 4.3: Compile

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

They are **Fibonacci.java** and **fib.c**.
Java source code must be compiled by the Java compiler (javac), and C source code must be compiled by the GCC compiler. The Python file (fib.py) and the Bash script (fib.sh) do not require compilation.

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

The **C source file (fib.c)** is compiled into native machine code when using the GCC compiler. The output is a binary executable file that can be run directly by the processor without requiring an interpreter or virtual machine.

Which source code files are compiled to byte code?

The **Java source file (Fibonacci.java)** is compiled into bytecode using the javac compiler. The result is a .class file that runs on the Java Virtual Machine (JVM), not directly on the CPU.

Which source code files are interpreted by an interpreter?

The files interpreted by an interpreter are **fib.py** and **fib.sh**.

- fib.py is interpreted by the Python3 interpreter.
- fib.sh is interpreted by the Bash shell.
  Both scripts run line-by-line without compilation

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

The **C program (fib.c)** is expected to perform the calculation the fastest. This is because compiled C code runs directly as optimized machine code on the CPU, while Java bytecode runs inside a virtual machine, and Python and Bash are interpreted, which makes them slower.

How do I run a Java program?

To run a Java program, first compile the .java file using the command:
javac Fibonacci.java
This creates a Fibonacci.class file.
Then run it with the Java Virtual Machine using:
java Fibonacci

How do I run a Python program?

To run a Python program, simply execute the following command in the terminal:
python3 fib.py
Python code does not need to be compiled before execution

How do I run a C program?

First compile the C source file using GCC, for example:
gcc -o fib fib.c
This creates an executable file named fib.
Then run the compiled program using:
./fib

How do I run a Bash script?

You can run a Bash script in two ways:

1. Run it directly with the Bash interpreter:
   bash fib.sh

2. Or make it executable and run it:
   chmod +x fib.sh
   ./fib.sh

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

Yes, compiling some of the source code files will create new files.

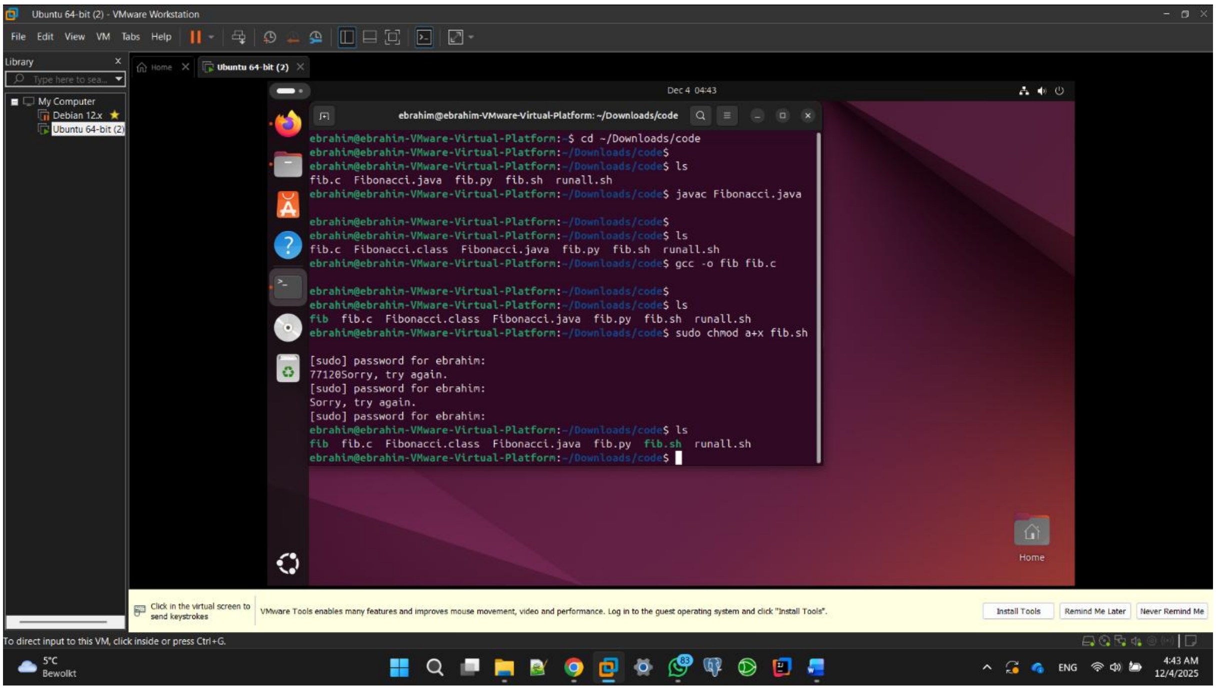
- When you compile **Fibonacci.java**, a new file named **Fibonacci.class** is created. This file contains Java bytecode for execution by the Java Virtual Machine.

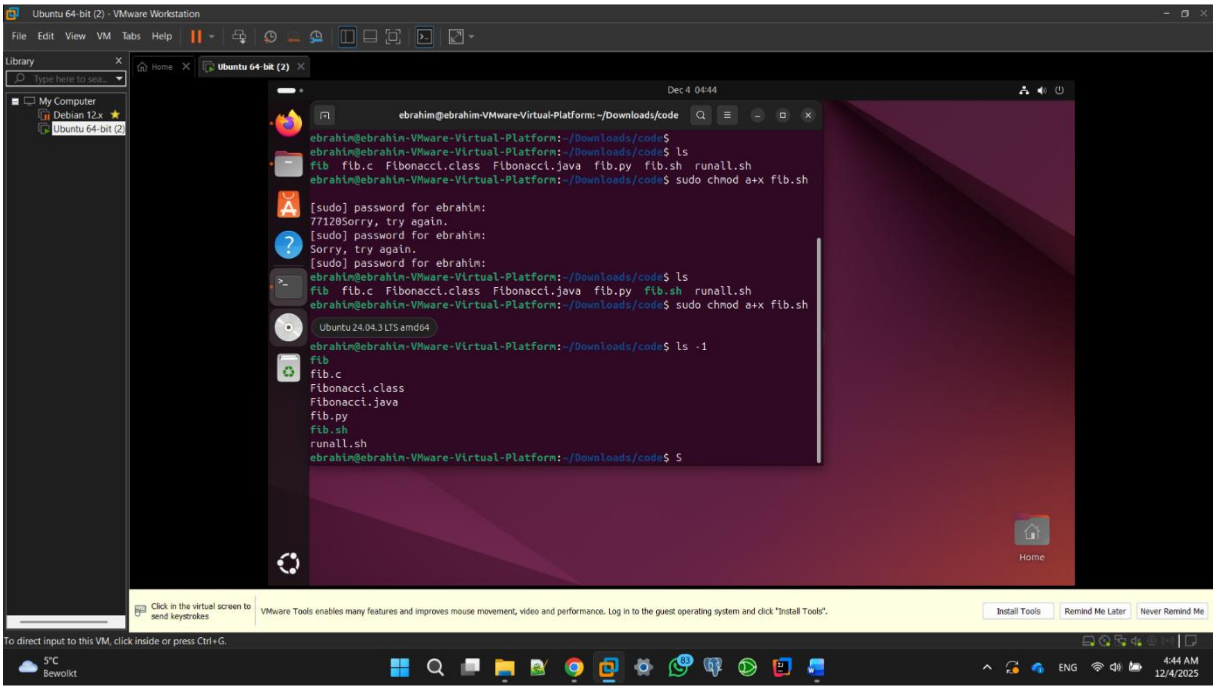
- When you compile **fib.c**, a new executable file is created. If you compile using gcc fib.c, the output file is named **a.out** by default. If you compile using gcc -o fib fib.c, the output file is named **fib**.
  Python (fib.py) and Bash (fib.sh) do not generate new files when run, because they are interpreted rather than compiled

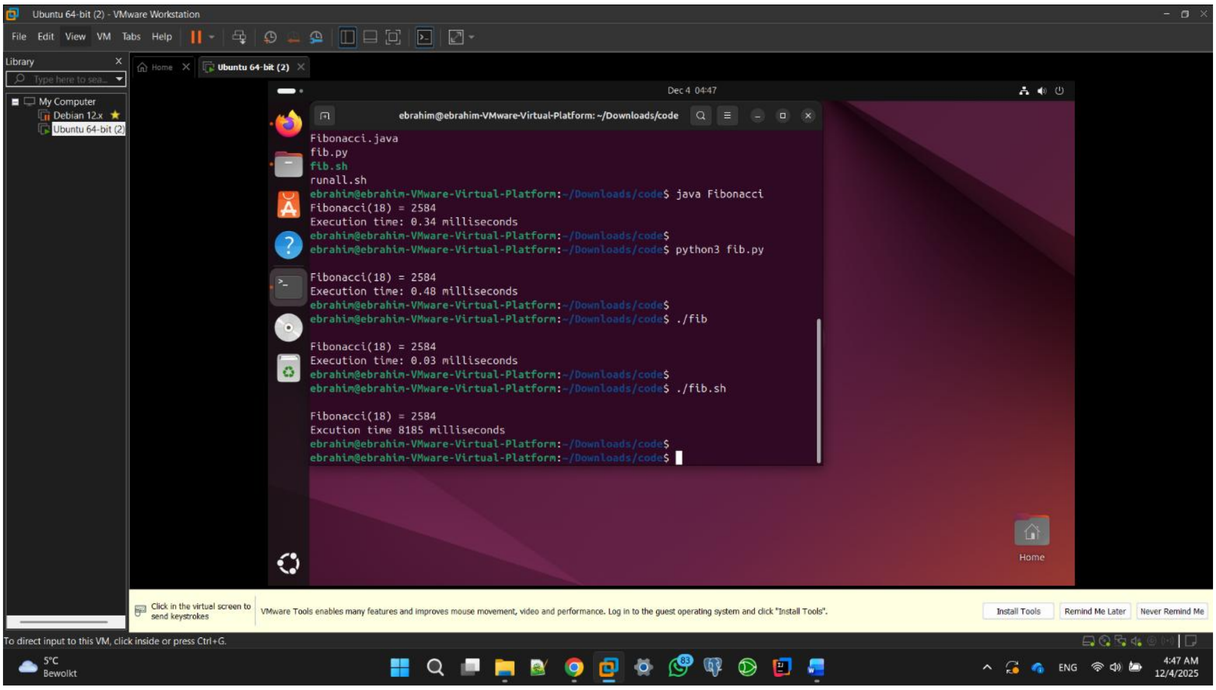
Take relevant screenshots of the following commands:

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

  The C program (fib.c) is the fastest, because it is compiled into native machine code that runs directly on the CPU.


Ubuntu 64-bit (2) - VMware Workstation
File Edit View VM Tabs Help
Library
My Computer
Debian 12.x
Ubuntu 64-bit (2)
Home
Ubuntu 64-bit (2)
Dec 4 04:43
ebrahim@ebrahim-VMware-Virtual-Platform: ~/Downloads/code
ebrahim@ebrahim-VMware-Virtual-Platform:~$ cd ~/Downloads/code
ebrahim@ebrahim-VMware-Virtual-Platform:~/Downloads/code$
ebrahim@ebrahim-VMware-Virtual-Platform:~/Downloads/code$ ls
fib.c Fibonacci.java fib.py fib.sh runall.sh
ebrahim@ebrahim-VMware-Virtual-Platform:~/Downloads/code$ javac Fibonacci.java
ebrahim@ebrahim-VMware-Virtual-Platform:~/Downloads/code$
ebrahim@ebrahim-VMware-Virtual-Platform:~/Downloads/code$ ls
fib.c Fibonacci.class Fibonacci.java fib.py fib.sh runall.sh
ebrahim@ebrahim-VMware-Virtual-Platform:~/Downloads/code$ gcc -o fib fib.c
ebrahim@ebrahim-VMware-Virtual-Platform:~/Downloads/code$
ebrahim@ebrahim-VMware-Virtual-Platform:~/Downloads/code$ ls
fib fib.c Fibonacci.class Fibonacci.java fib.py fib.sh runall.sh
ebrahim@ebrahim-VMware-Virtual-Platform:~/Downloads/code$ sudo chmod a+x fib.sh
[sudo] password for ebrahim:
77120Sorry, try again.
[sudo] password for ebrahim:
Sorry, try again.
[sudo] password for ebrahim:
ebrahim@ebrahim-VMware-Virtual-Platform:~/Downloads/code$ ls
fib fib.c Fibonacci.class Fibonacci.java fib.py fib.sh runall.sh
ebrahim@ebrahim-VMware-Virtual-Platform:~/Downloads/code$
Home
Click in the virtual screen to send keystrokes
VMware Tools enables many features and improves mouse movement, video and performance. Log in to the guest operating system and click "Install Tools".
Install Tools
Remind Me Later
Never Remind Me
To direct input to this VM, click inside or press Ctrl+G.
5°C
Bewölkt
ENG
4:43 AM
12/4/2025


Ubuntu 64-bit (2) - VMware Workstation
File Edit View VM Tabs Help
Library
My Computer
Debian 12.x
Ubuntu 64-bit (2)
Home
Ubuntu 64-bit (2)
Dec 4 04:44
ebrahim@ebrahim-VMware-Virtual-Platform: ~/Downloads/code
ebrahim@ebrahim-VMware-Virtual-Platform:~/Downloads/code$
ebrahim@ebrahim-VMware-Virtual-Platform:~/Downloads/code$ ls
fib fib.c Fibonacci.class Fibonacci.java fib.py fib.sh runall.sh
ebrahim@ebrahim-VMware-Virtual-Platform:~/Downloads/code$ sudo chmod a+x fib.sh
[sudo] password for ebrahim:
77120Sorry, try again.
[sudo] password for ebrahim:
Sorry, try again.
[sudo] password for ebrahim:
ebrahim@ebrahim-VMware-Virtual-Platform:~/Downloads/code$ ls
fib fib.c Fibonacci.class Fibonacci.java fib.py fib.sh runall.sh
ebrahim@ebrahim-VMware-Virtual-Platform:~/Downloads/code$ sudo chmod a+x fib.sh
Ubuntu 24.04.3 LTS amd64
ebrahim@ebrahim-VMware-Virtual-Platform:~/Downloads/code$ ls -1
fib
fib.c
Fibonacci.class
Fibonacci.java
fib.py
fib.sh
runall.sh
ebrahim@ebrahim-VMware-Virtual-Platform:~/Downloads/code$ S
Home
Click in the virtual screen to send keystrokes
VMware Tools enables many features and improves mouse movement, video and performance. Log in to the guest operating system and click "Install Tools".
Install Tools
Remind Me Later
Never Remind Me
To direct input to this VM, click inside or press Ctrl+G.
5°C
Bewölkt
ENG
4:44 AM
12/4/2025

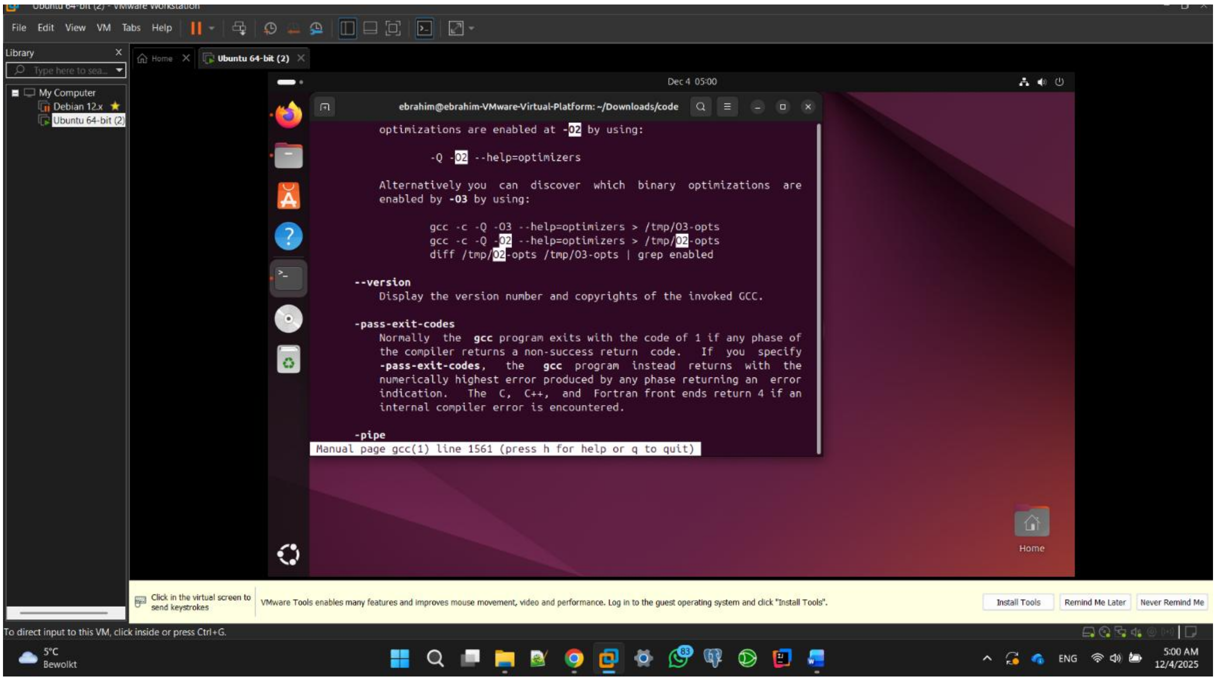


Assignment 4.4: Optimize

Take relevant screenshots of the following commands:

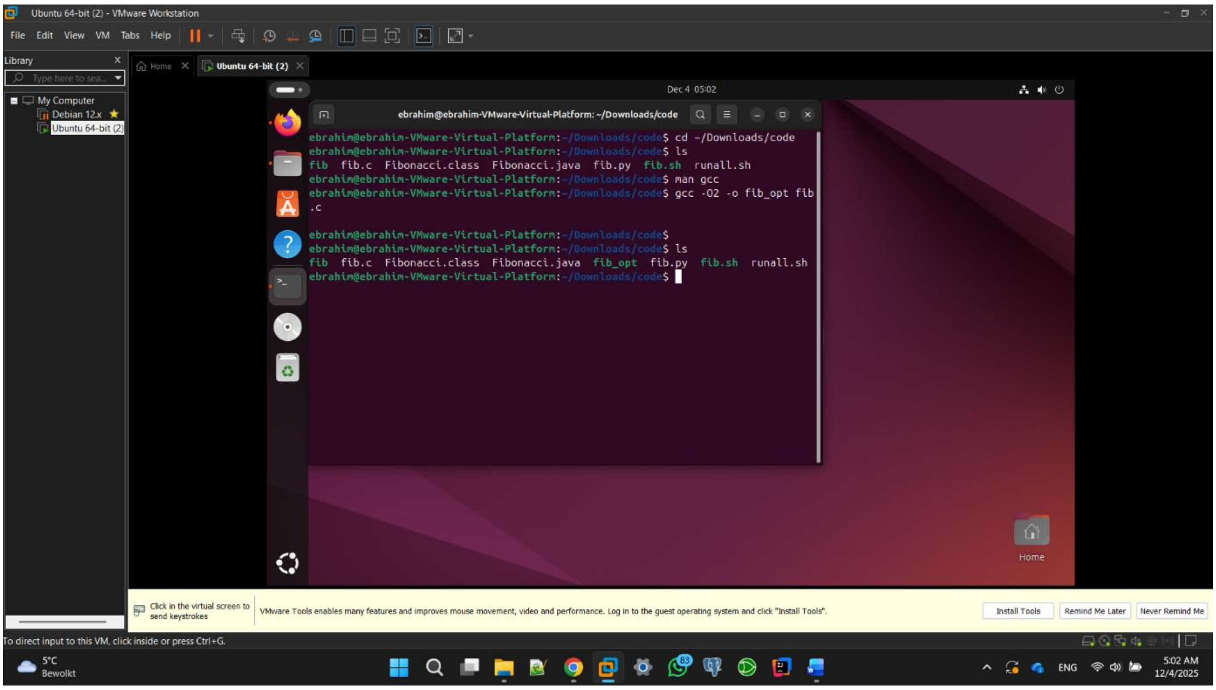
a) Figure out which parameters you need to pass to **the gcc** compiler so that the compiler performs a number of optimizations that will ensure that the compiled source code will run faster. **Tip!** The parameters are usually a letter followed by a number. Also read **page 191** of your book, but find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

A C program is usually faster because it is compiled directly to native machine code, which runs straight on the CPU. Java is compiled to bytecode and runs inside the Java Virtual Machine (JVM), and Python and Bash are interpreted line by line. The extra layers (virtual machine or interpreter) add overhead, so the C version is expected to be the fastest.



written as a capital letter **O** followed by a number. Examples are **-O1**, **-O2** and **-O3**.
The option **-O2** enables a wide range of optimizations that usually make the compiled program faster without changing its behaviour. The option **-O3** enables even more aggressive optimizations, focusing on speed, but it may increase compilation time and code size. In this assignment I used **-O2** as an optimization level for fib.c

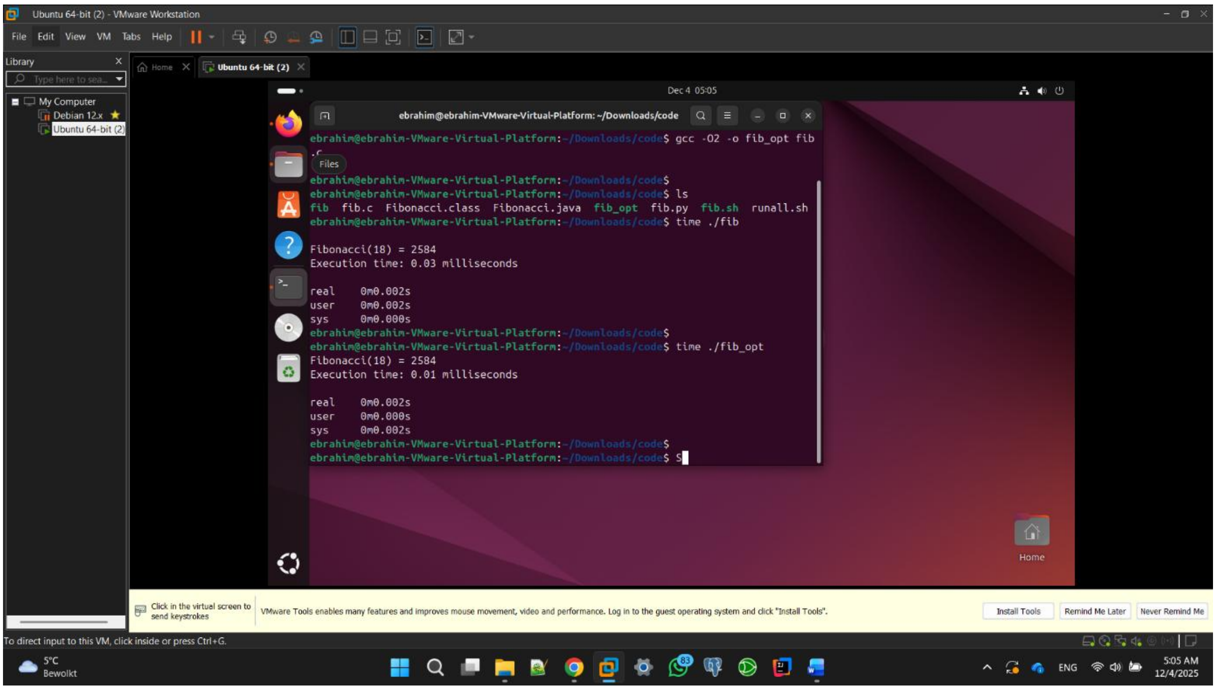
b) Compile **fib.c** again with the optimization parameters
I recompiled fib.c with optimization using the command:
gcc -O2 -o fib_opt fib.c
This creates an optimized executable named fib_opt



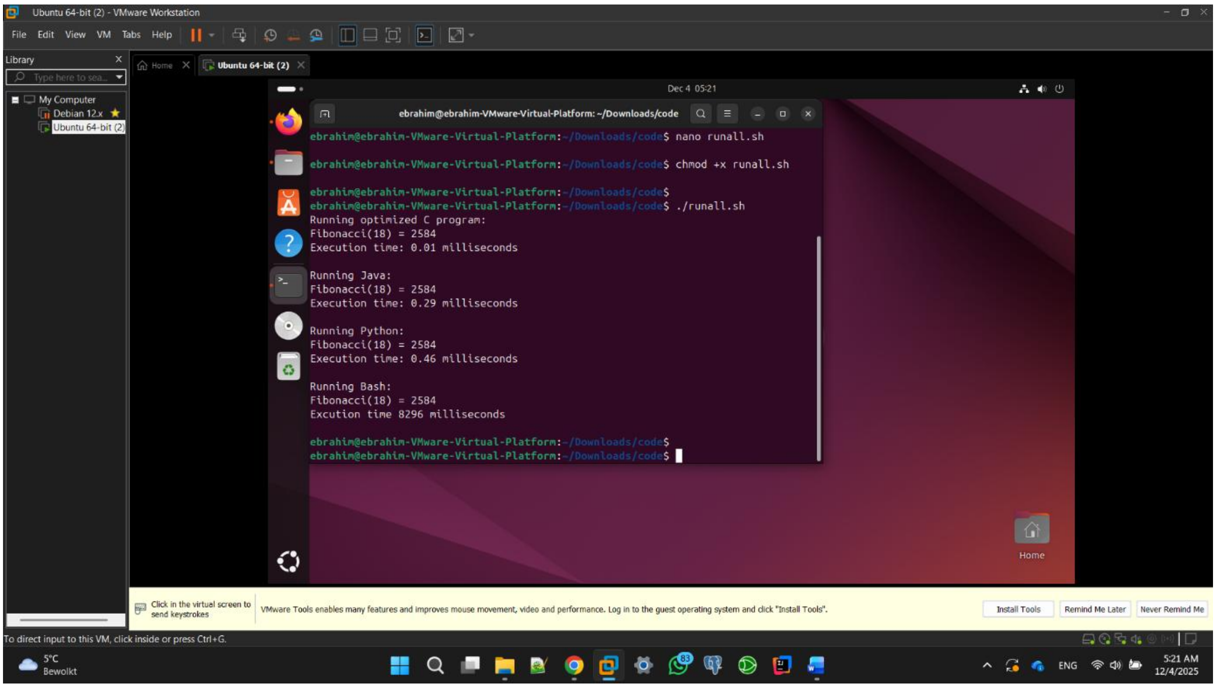
c) Run the newly compiled program. Is it true that it now performs the calculation faster?

d)

I compared the execution time of the normal C program and the optimized version using the time command:
time ./fib
time ./fib_opt
In theory, the optimized version fib_opt should run faster because of the additional compiler optimizations. In practice, for such a small program the difference is very small and sometimes hard to see, but the optimized binary is expected to be at least as fast or slightly faste



e) 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

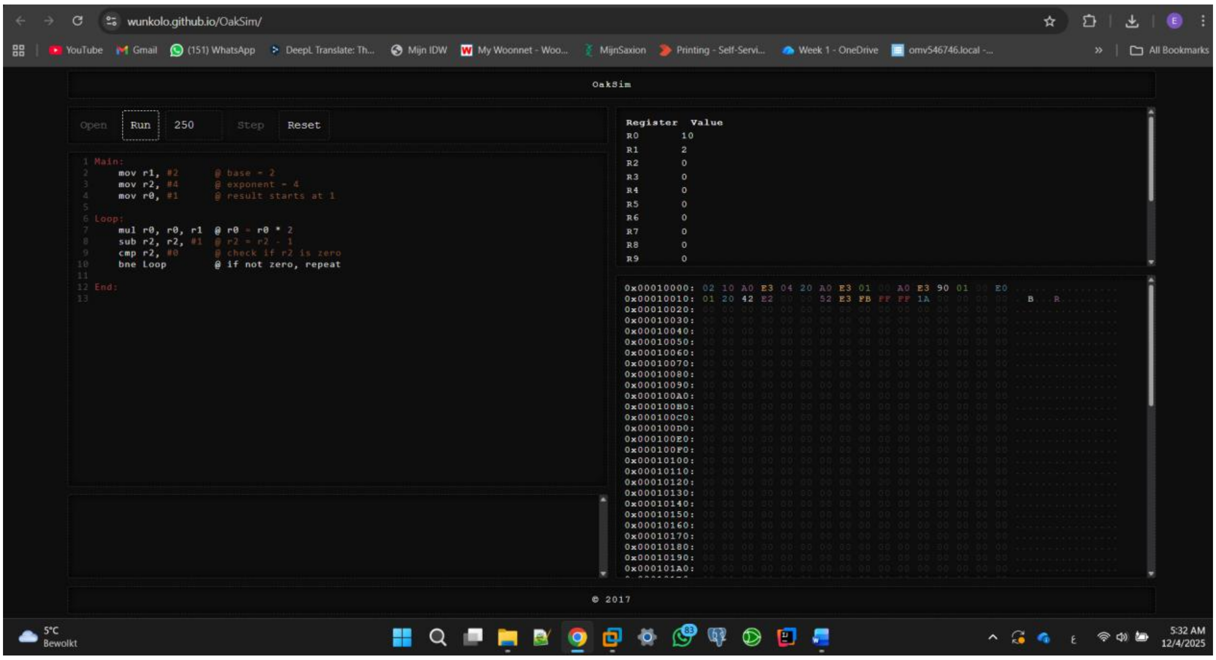
Like the factorial example, you can also implement the calculation of a power of 2 in assembly. For example you want to calculate $2^4 = 16$. Use iteration to calculate the result. Store the result in r0.

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

Loop:
    mul r0, r0, r1
    sub r2, r2, #1
    cmp r2, #0
    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**