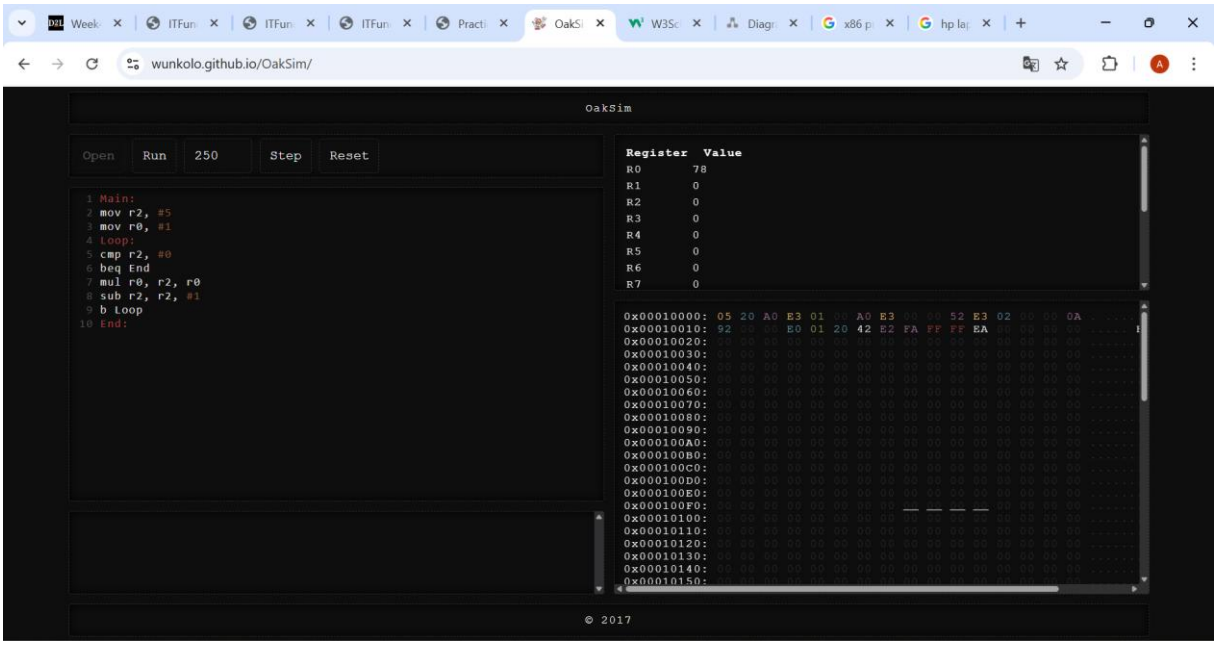
# Template Week 4 – Software

Student number: 591905

### Assignment 4.1: ARM assembly

Screenshot of working assembly code of factorial calculation:



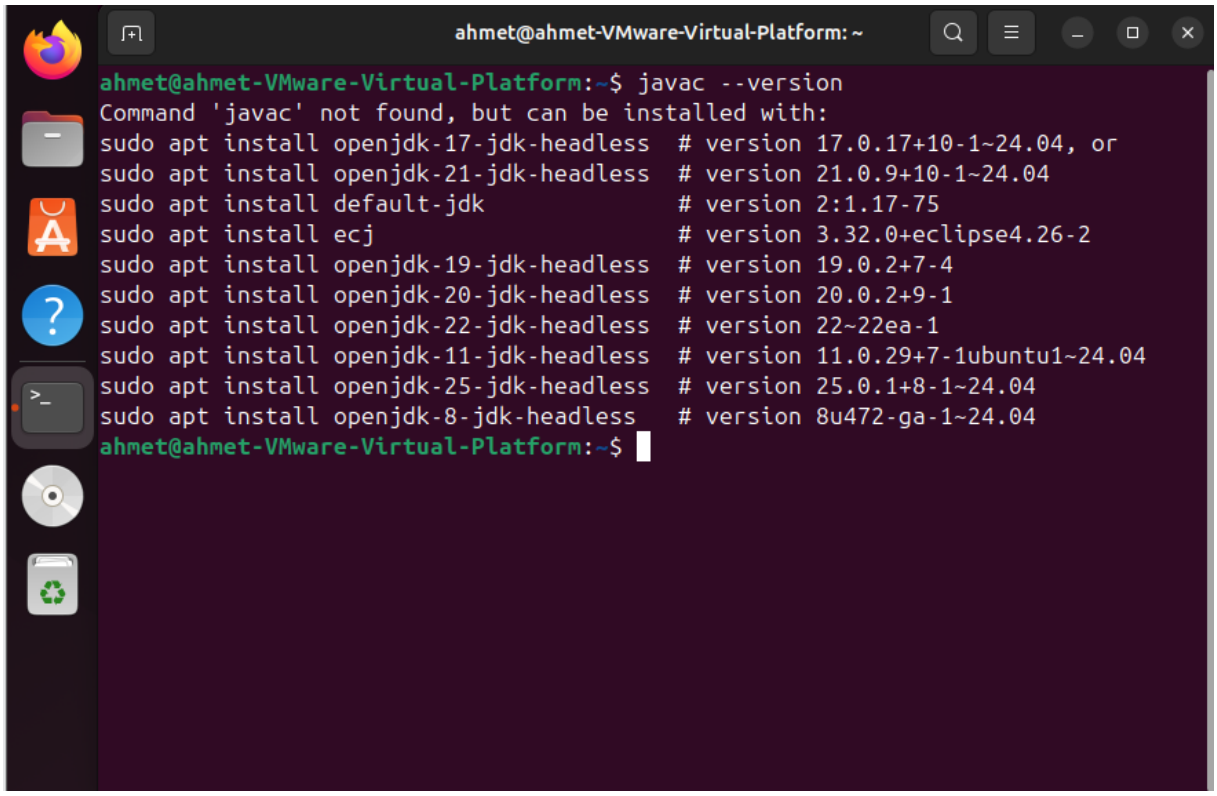

```
1  Main:
2  mov r2, #5
3  mov r0, #1
4  Loop:
5  cmp r2, #0
6  beq End
7  mul r0, r2, r0
8  sub r2, r2, #1
9  b Loop
10 End:
```

| Register | Value |
|---|---|
| R0 | 78 |
| R1 | 0 |
| R2 | 0 |
| R3 | 0 |
| R4 | 0 |
| R5 | 0 |
| R6 | 0 |
| R7 | 0 |

**Assignment 4.2: Programming languages**

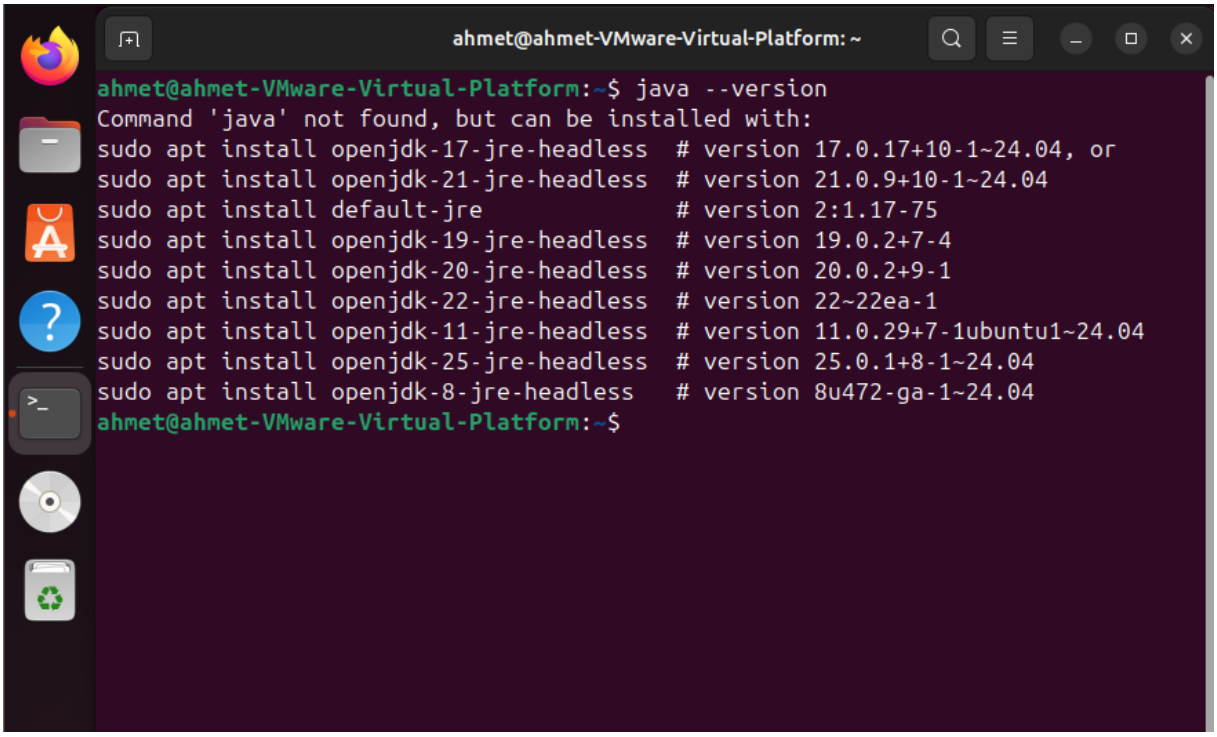
Take screenshots that the following commands work:

javac –version



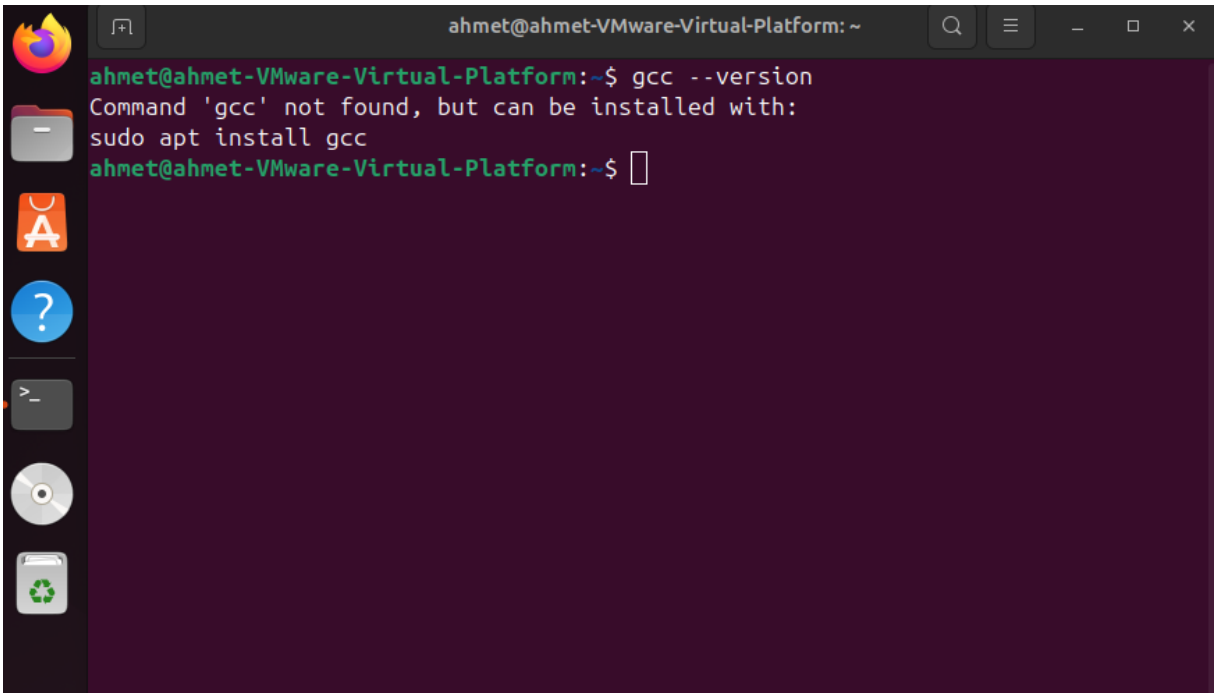

Geinstalleerd met sudo apt install openjdk-17-jdk-headless, versie 17.0.17+10- 1 ~24.04

java –version



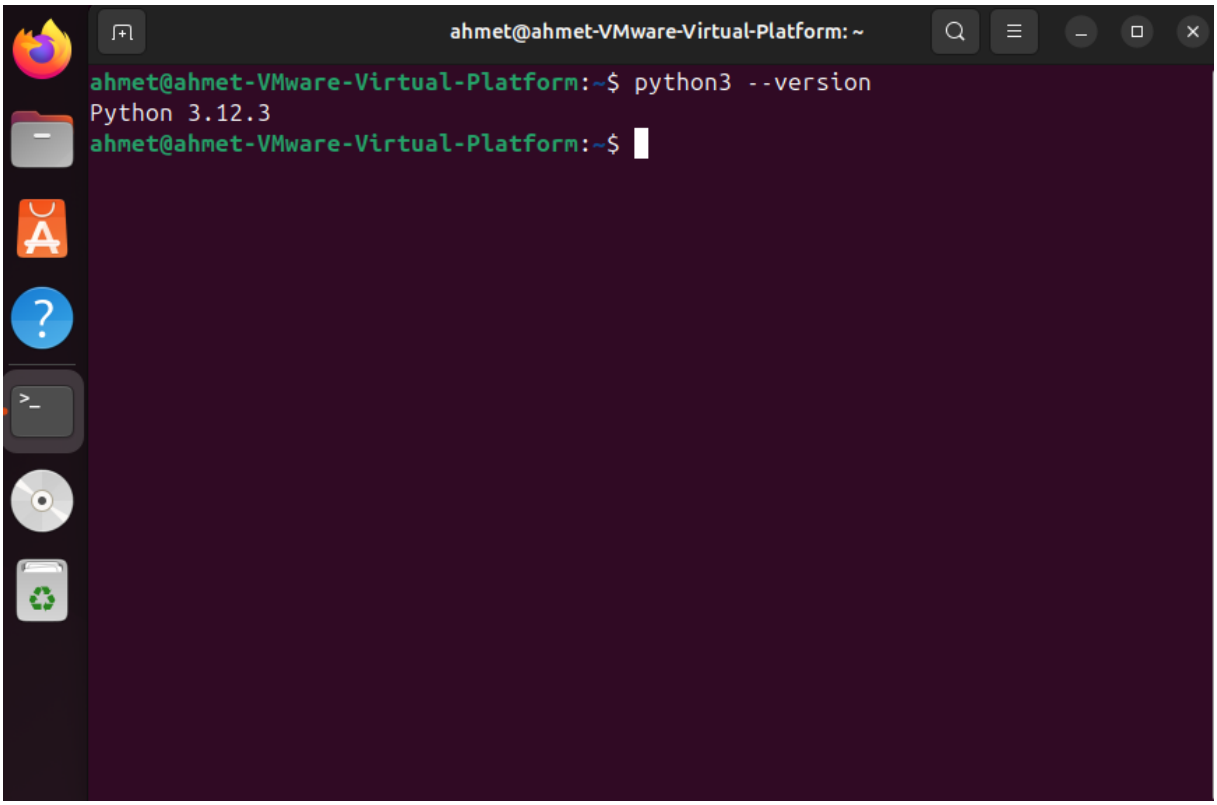

Geinstalleerd met sudo apt install openjdk-17-jre-headless, versie 17.0.17+10- 1 ~24.04

gcc –version



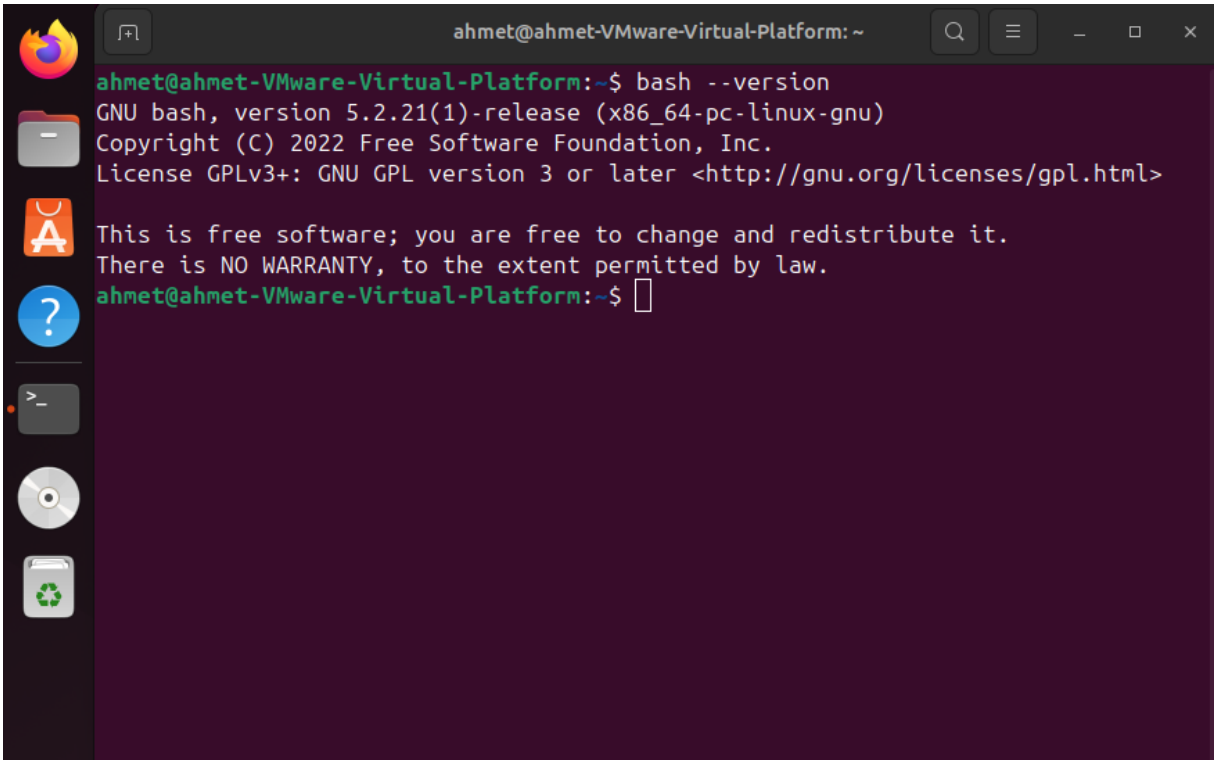

Geinstalleerd met sudo apt install gcc

python3 –version




Versie 3.12.3

bash –version




Versie 5.2.21(1)

**Assignment 4.3: Compile**

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

fib.c (C), het is een compiled source code

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

fib.c (C)

Which source code files are compiled to byte code?

Fibonacci.java (Java). Die wordt Bytecode voor de JVM.

Which source code files are interpreted by an interpreter?

Fib.py (Python) en fib.sh (Bash), het zijn interpreted source codes.

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

fib.c (C). Dit draait direct op de processor (machine code) zonder tussenlaag (intermediate code) wat zorgt voor meer snelheid.

How do I run a Java program?

Compileren: javac Fibonacci.java, en daarna uitvoeren: java Fibonacci

How do I run a Python program?

Met python3 fib.py

How do I run a C program?

Compileren: gcc fib.c, en daarna uitvoeren: ./a.out

How do I Run a Bash Script?

Met bash fib.sh

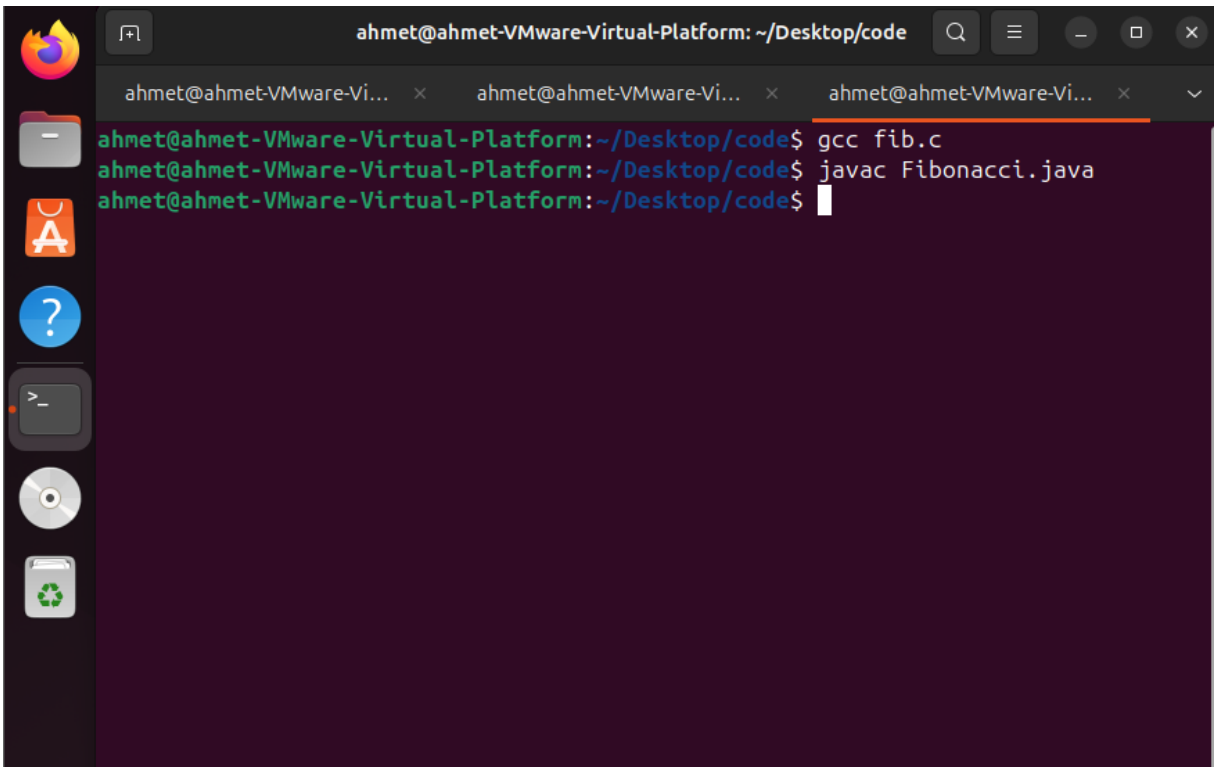
If I compile the above source code, will a new file be created? If so, which file?

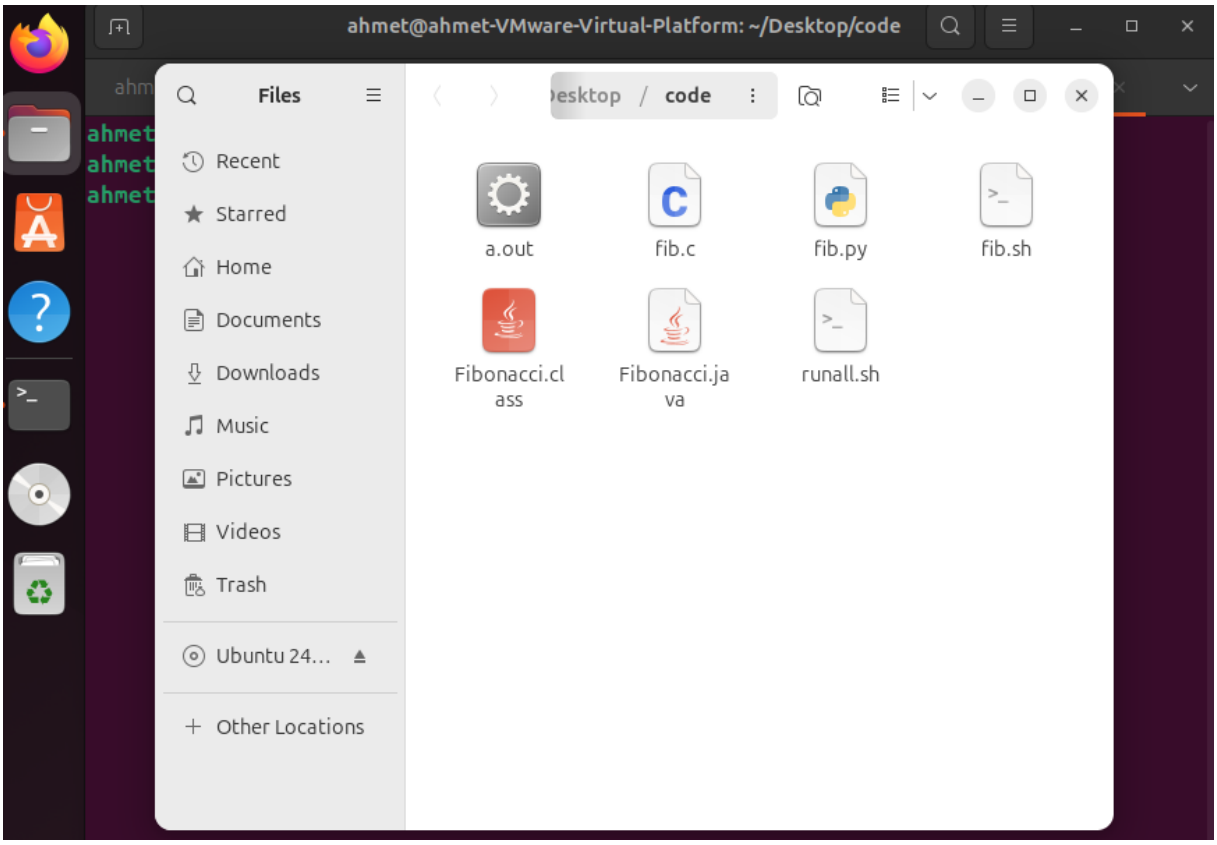
Ja, bij de compileertalen. Bij C (fib.c) ontstaat er een uitvoerbaar bestand (a.out) die in machinecode is omgezet. Bij Java (Fibonacci.java) ontstaat er een bytecode bestand (Fibonacci.class) die door de JVM zal worden uitgevoerd.

(Bij Python en Bash worden geen nieuwe bestanden aangemaakt om het te kunnen draaien omdat ze interpreted zijn).

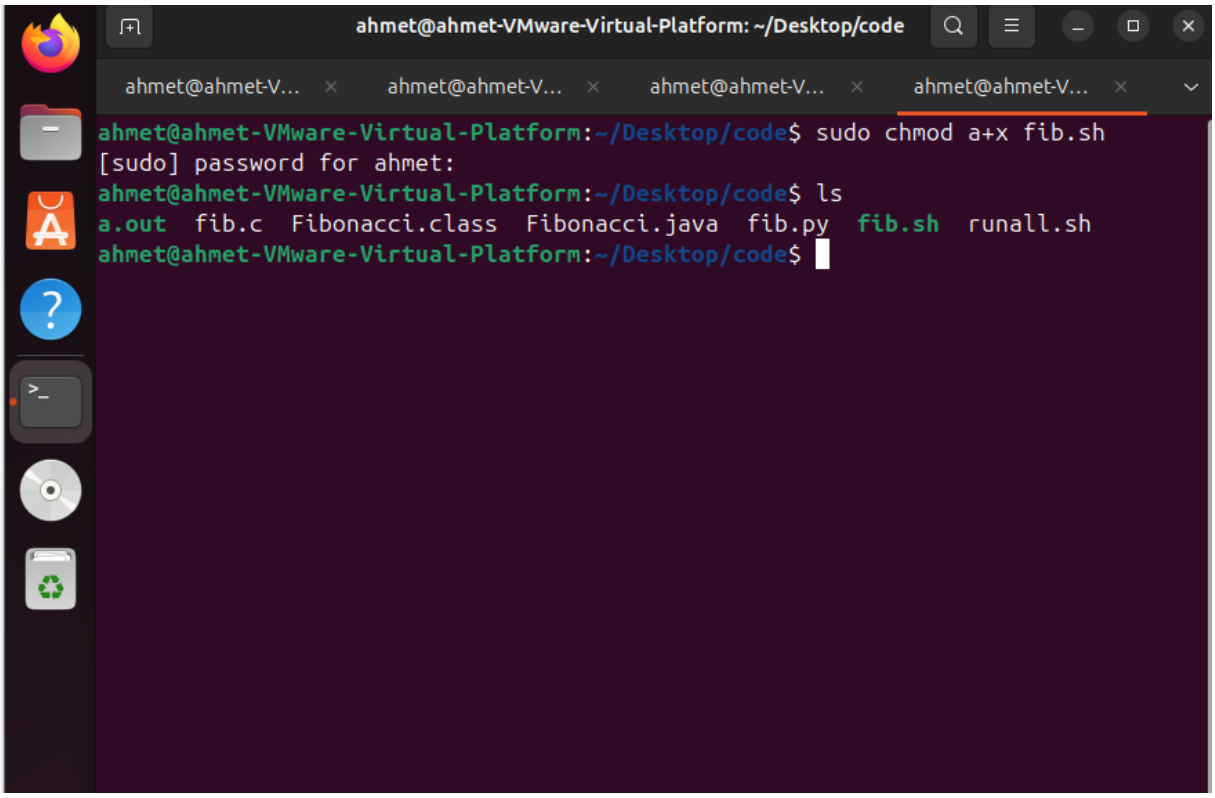
Take relevant screenshots of the following commands:

- Compile the source files where necessary

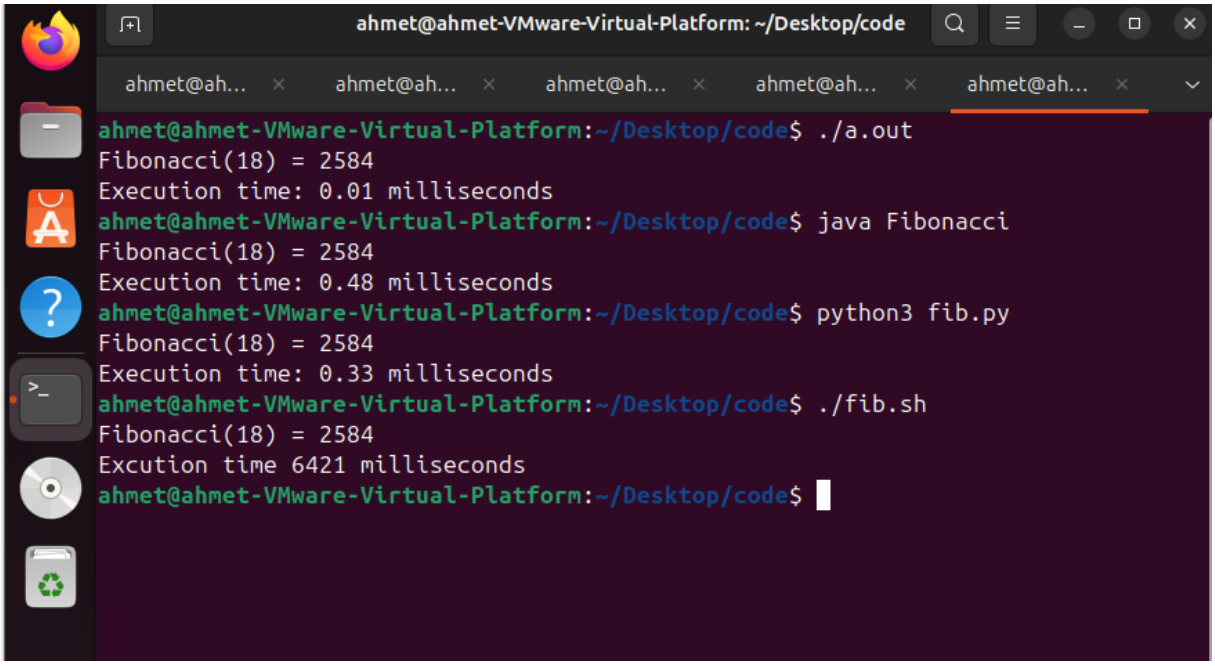





- Make them executable



- Run them




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

Het C-programma (fib.c) (a.out). C wordt direct vertaald naar machinecode (de taal van de processor). Java heeft een virtuele machine nodig en Python/Bash zijn interpreters die regel voor regel lezen, wat trager is.

**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.

In het boek staat er iets over -O (hoofdletter O).

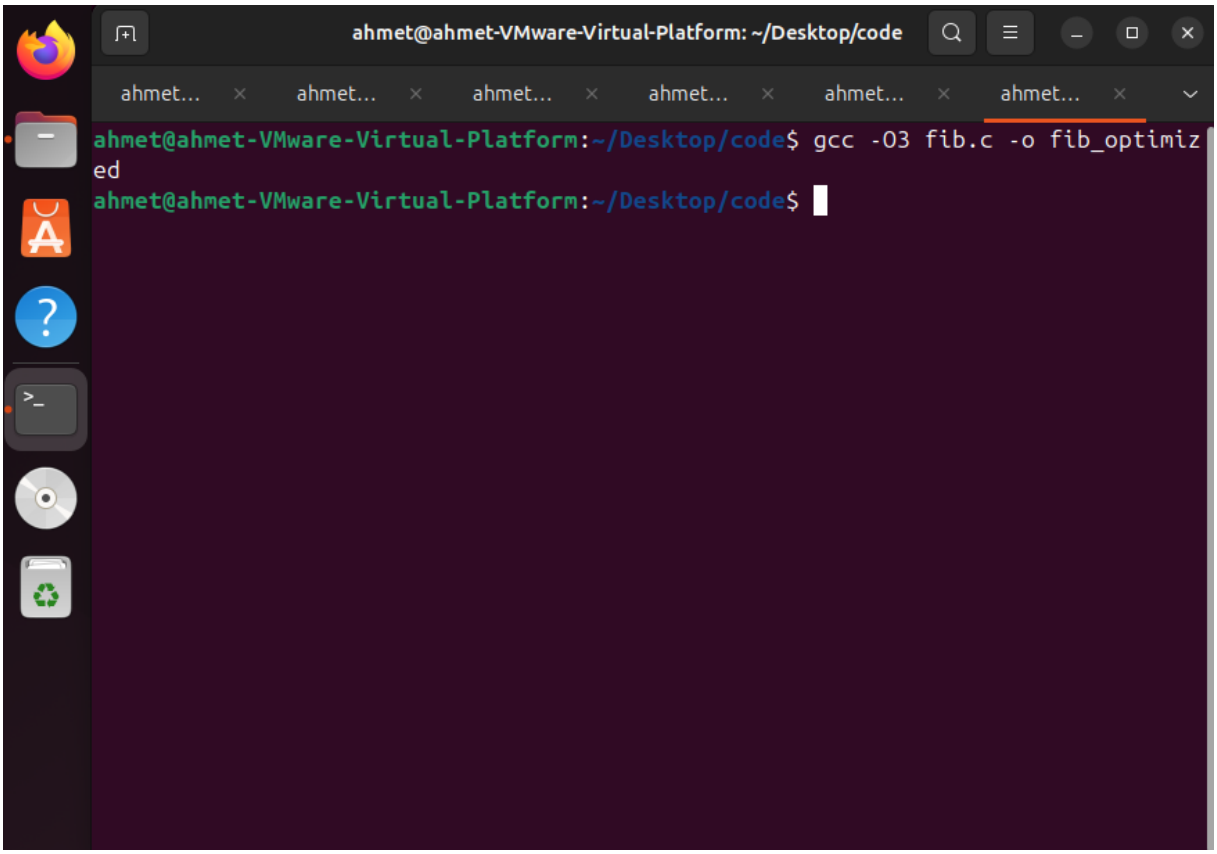
-O0: Geen optimalisatie (standaard).

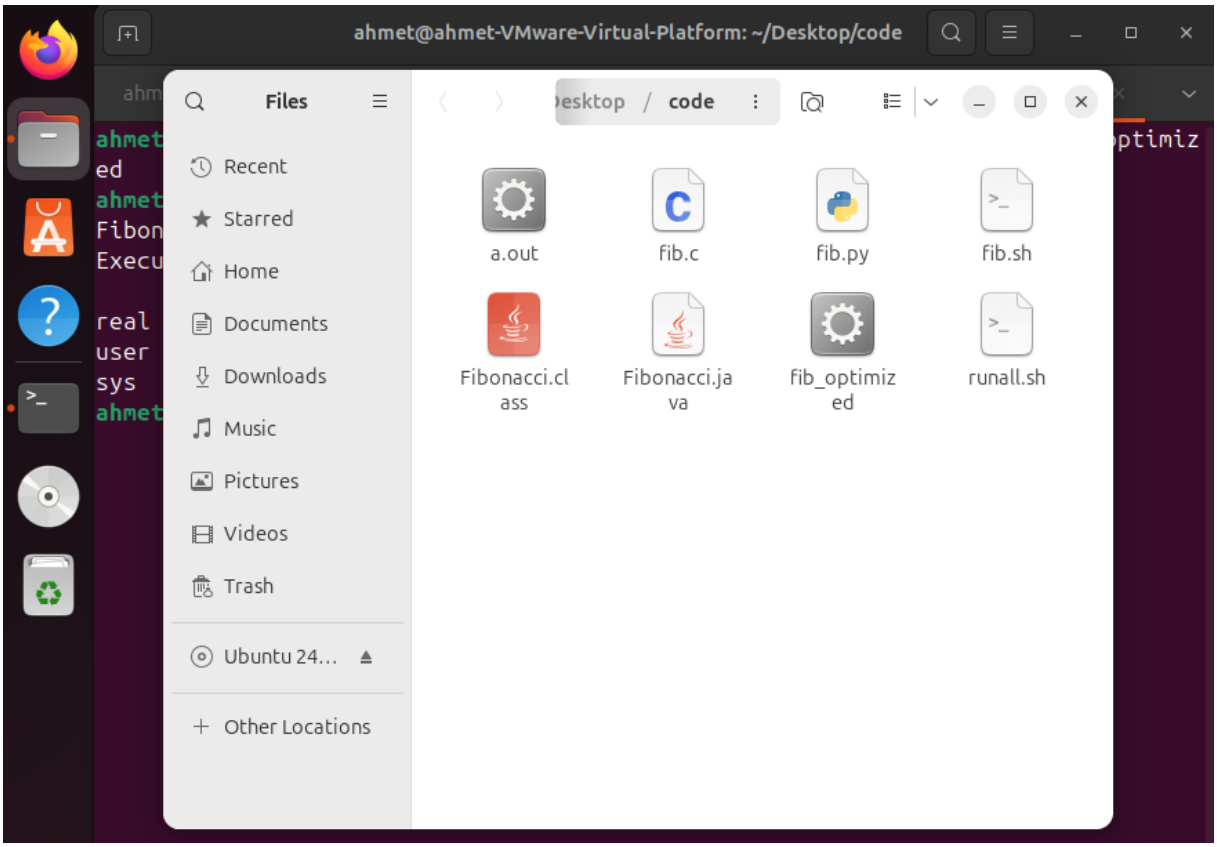
-O1: Beetje optimalisatie.

-O2: Veel optimalisatie (standaard voor programma's die je uitbrengt).

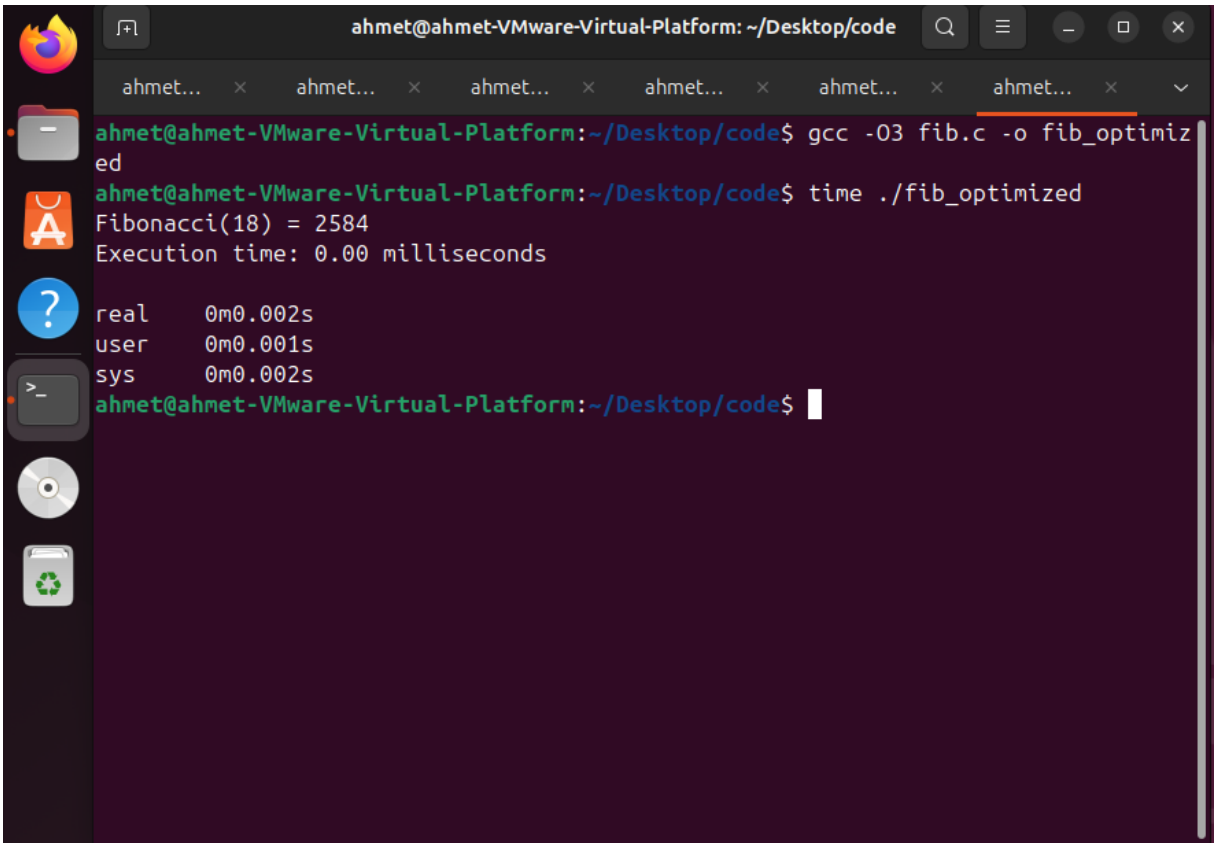
-O3: Maximale snelheid optimalisatie (agressief).

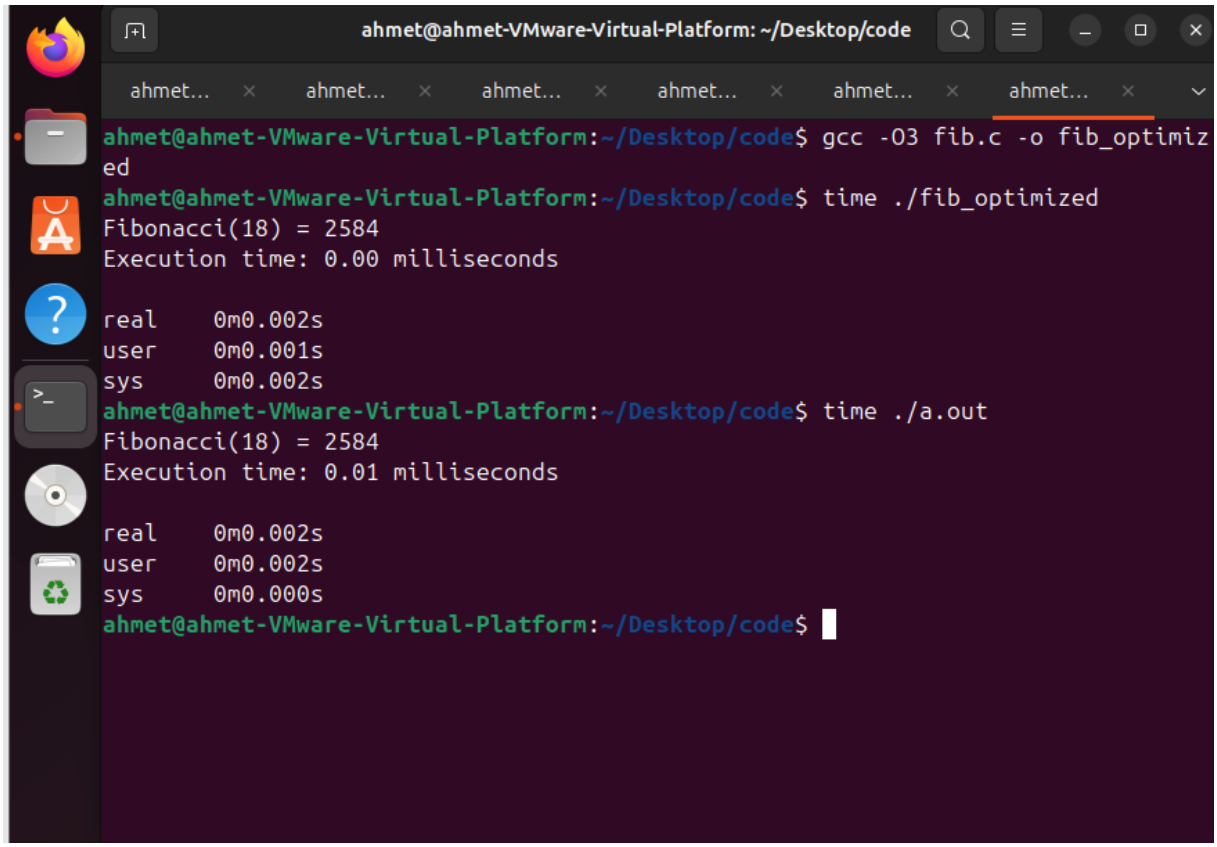
b) Compile **fib.c** again with the optimization parameters




ahmet@ahmet-VMware-Virtual-Platform: ~/Desktop/code
Files
Recent
Starred
Home
Documents
Downloads
Music
Pictures
Videos
Trash
Ubuntu 24...
Other Locations
code
a.out
fib.c
fib.py
fib.sh
Fibonacci.class
Fibonacci.java
fib_optimized
runall.sh

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



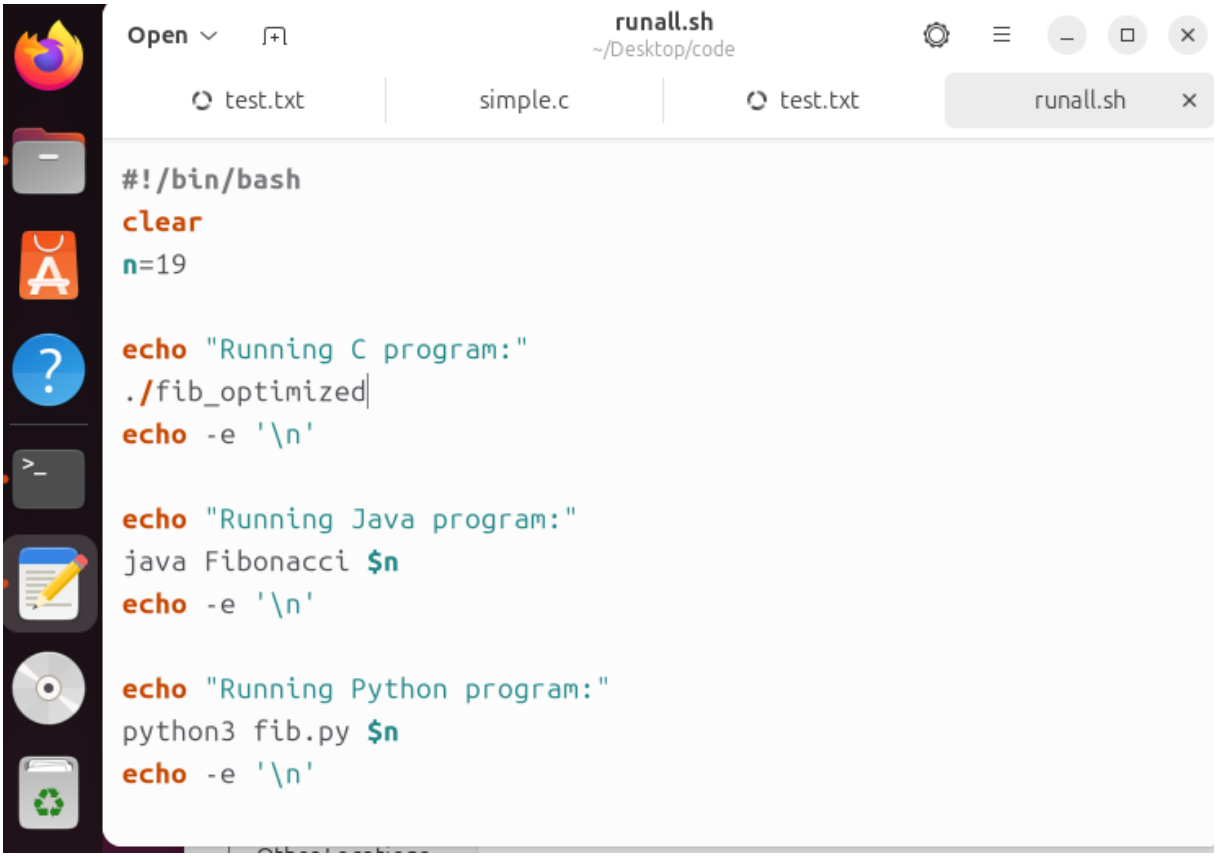
De geoptimaliseerde C-versie is sneller. Dus dat klopt

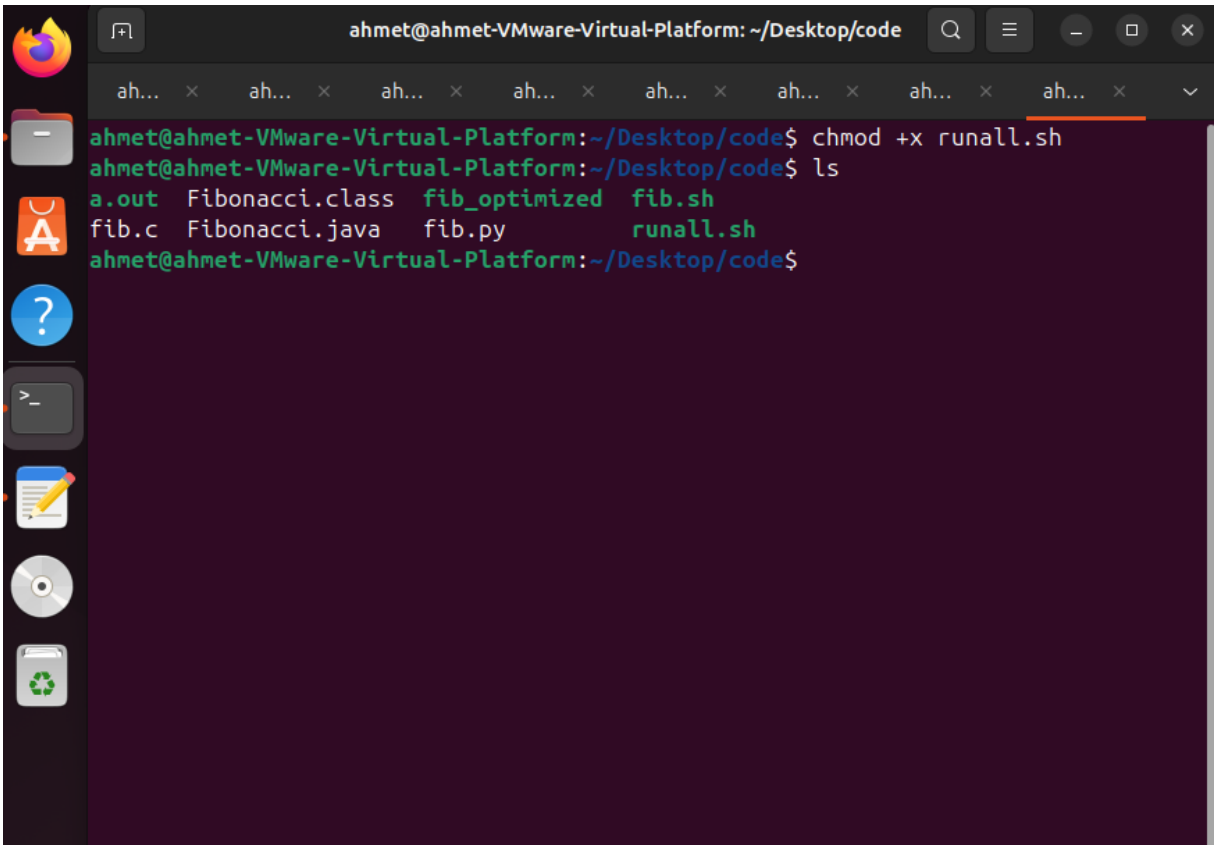
C (geoptimaliseerd): 0.00 milliseconden

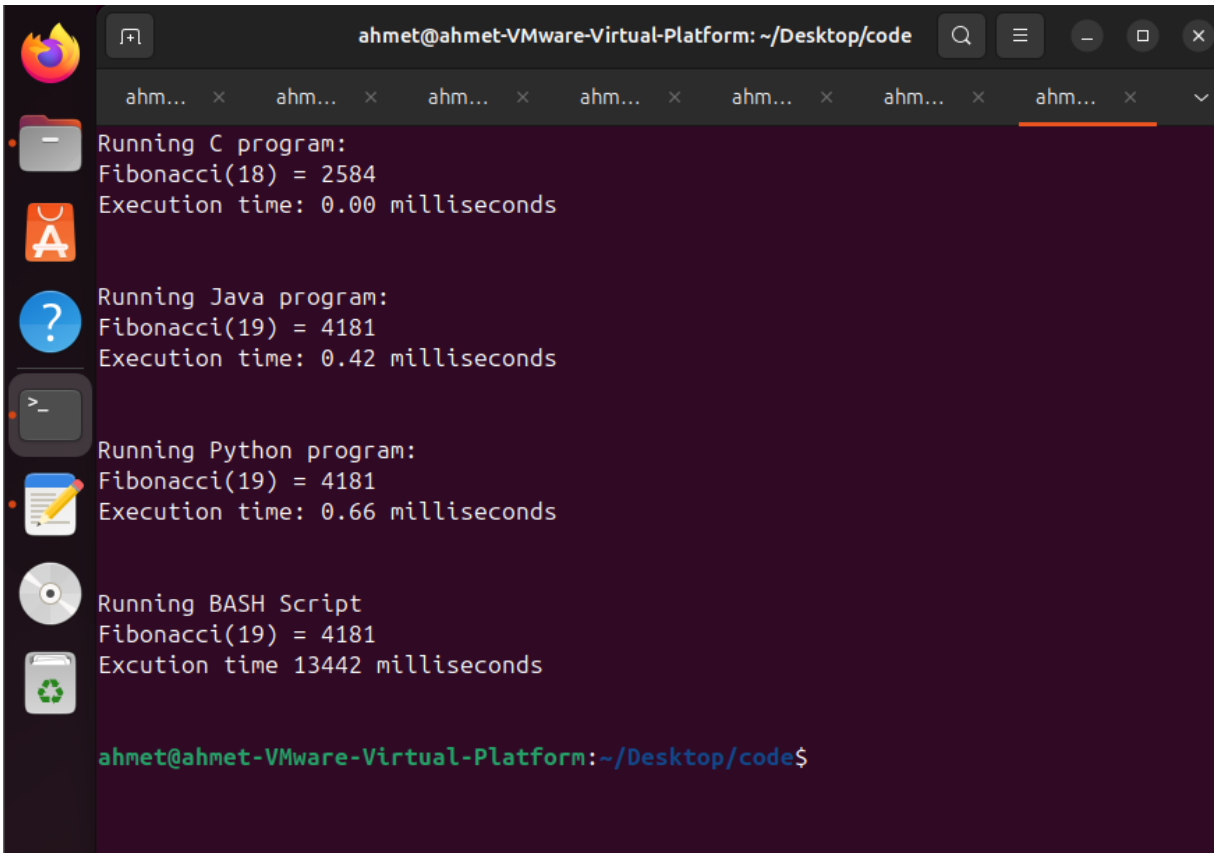
C (ongeoptimaliseerd): 0.01 milliseconden

De compiler-optimalisatie (-O3) heeft de uitvoeringstijd dus verkort.

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.



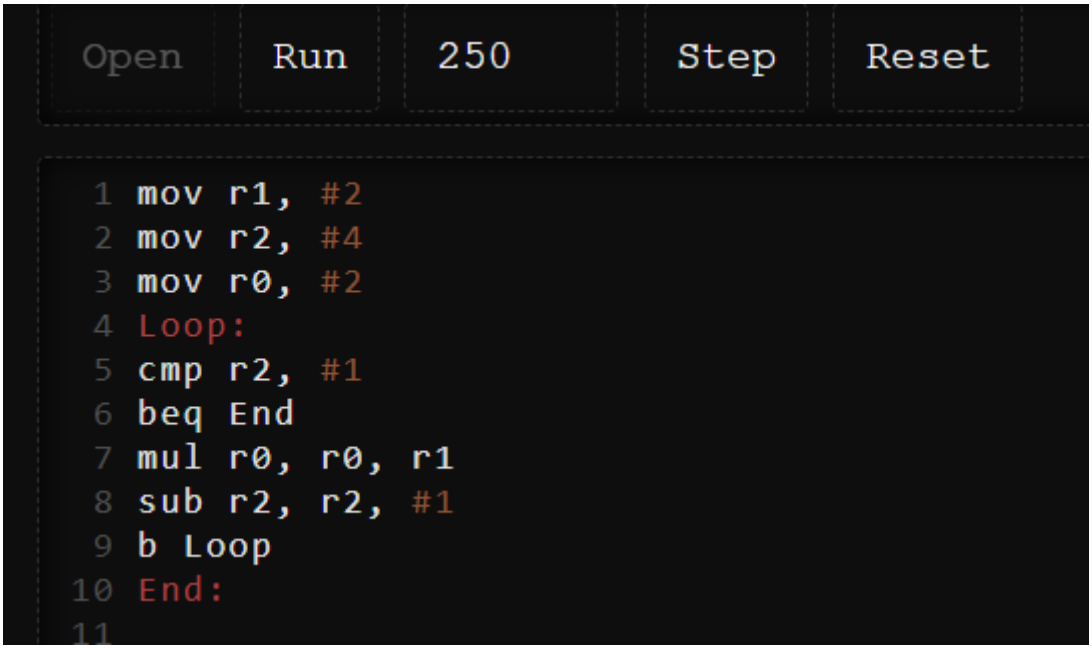

ahmet@ahmet-VMware-Virtual-Platform: ~/Desktop/code
ahmet@ahmet-VMware-Virtual-Platform:~/Desktop/code$ chmod +x runall.sh
ahmet@ahmet-VMware-Virtual-Platform:~/Desktop/code$ ls
a.out Fibonacci.class fib_optimized fib.sh
fib.c Fibonacci.java fib.py runall.sh
ahmet@ahmet-VMware-Virtual-Platform:~/Desktop/code$

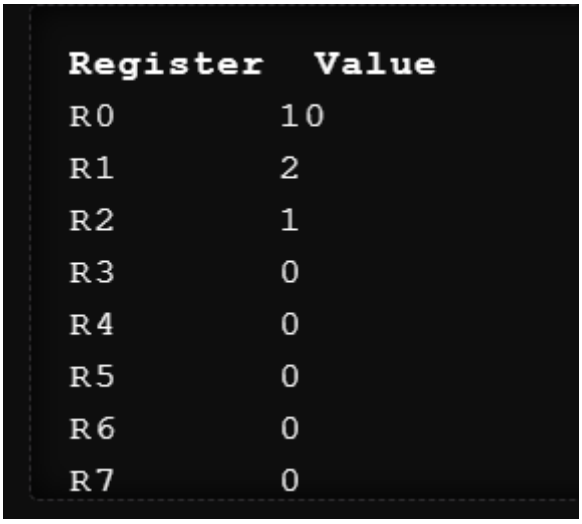

ahmet@ahmet-VMware-Virtual-Platform: ~/Desktop/code
Running C program:
Fibonacci(18) = 2584
Execution time: 0.00 milliseconds
Running Java program:
Fibonacci(19) = 4181
Execution time: 0.42 milliseconds
Running Python program:
Fibonacci(19) = 4181
Execution time: 0.66 milliseconds
Running BASH Script
Fibonacci(19) = 4181
Excution time 13442 milliseconds
ahmet@ahmet-VMware-Virtual-Platform:~/Desktop/code$

**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.

Screenshot of the completed code here.




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

```
Main:
mov r1, #2
mov r2, #4
mov r0, #2
Loop:
cmp r2, #1
beq End
mul r0, r0, r1
sub r2, r2, #1
b Loop
End:
```

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