

Template Week 4 – Software

Student number: 589531

Assignment 4.1: ARM assembly

Screenshot of working assembly code of factorial calculation:

The screenshot shows an ARM assembly simulator interface. On the left, the assembly code is displayed, and on the right, the register values and memory dump are shown.

Assembly Code:

```
1 Main:
2   mov r2, #5
3   mov r1, #1
4
5 Loop:
6   mul r1, r1, r2
7   sub r2, r2, #1
8   cmp r2, #1
9   beq End
10  b Loop
11
12 End:
13  b End
14
```

Register Values:

Register	Value
R0	0
R1	78
R2	1
R3	0
R4	0
R5	0
R6	0
R7	0
R8	0
R9	0
R10	0

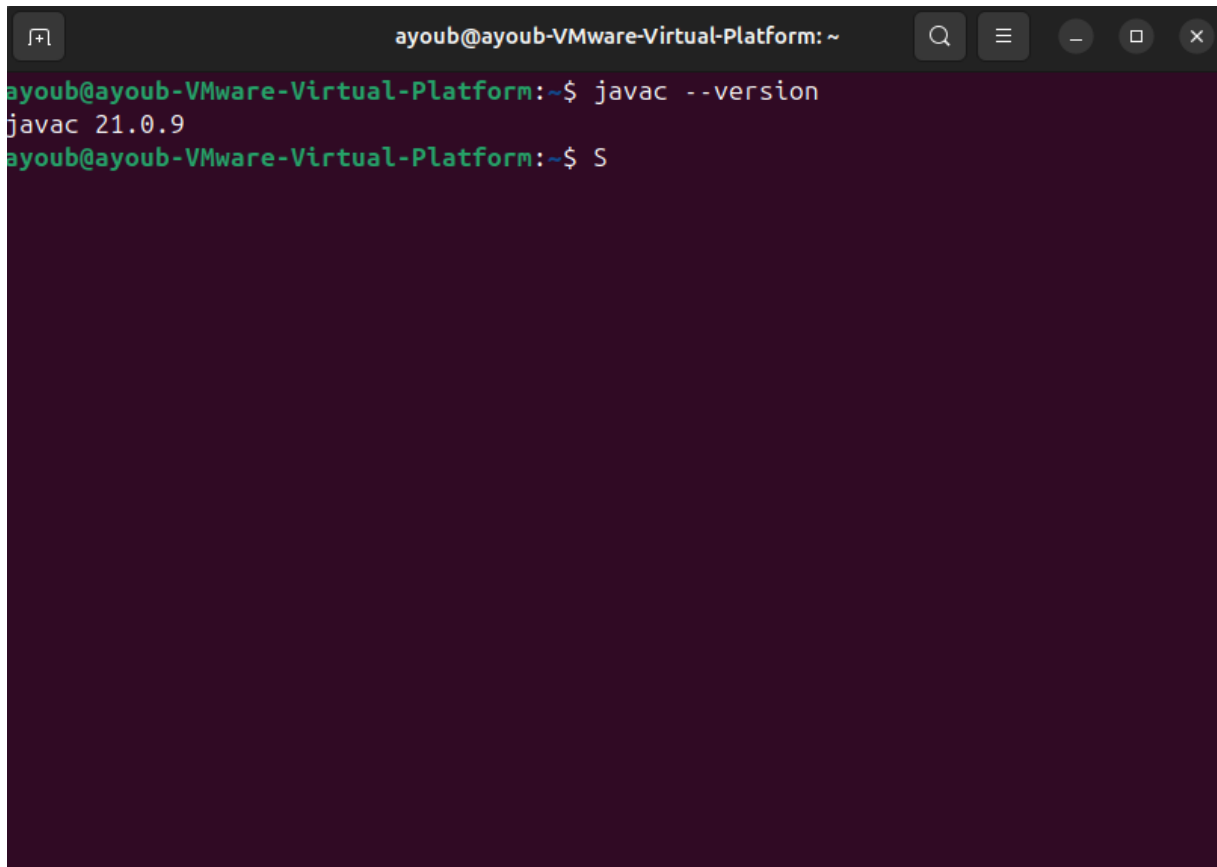
Memory Dump:

The memory dump shows the state of memory starting from address 0x00010060. The values are mostly zeros, indicating that the memory has been cleared or is uninitialized.

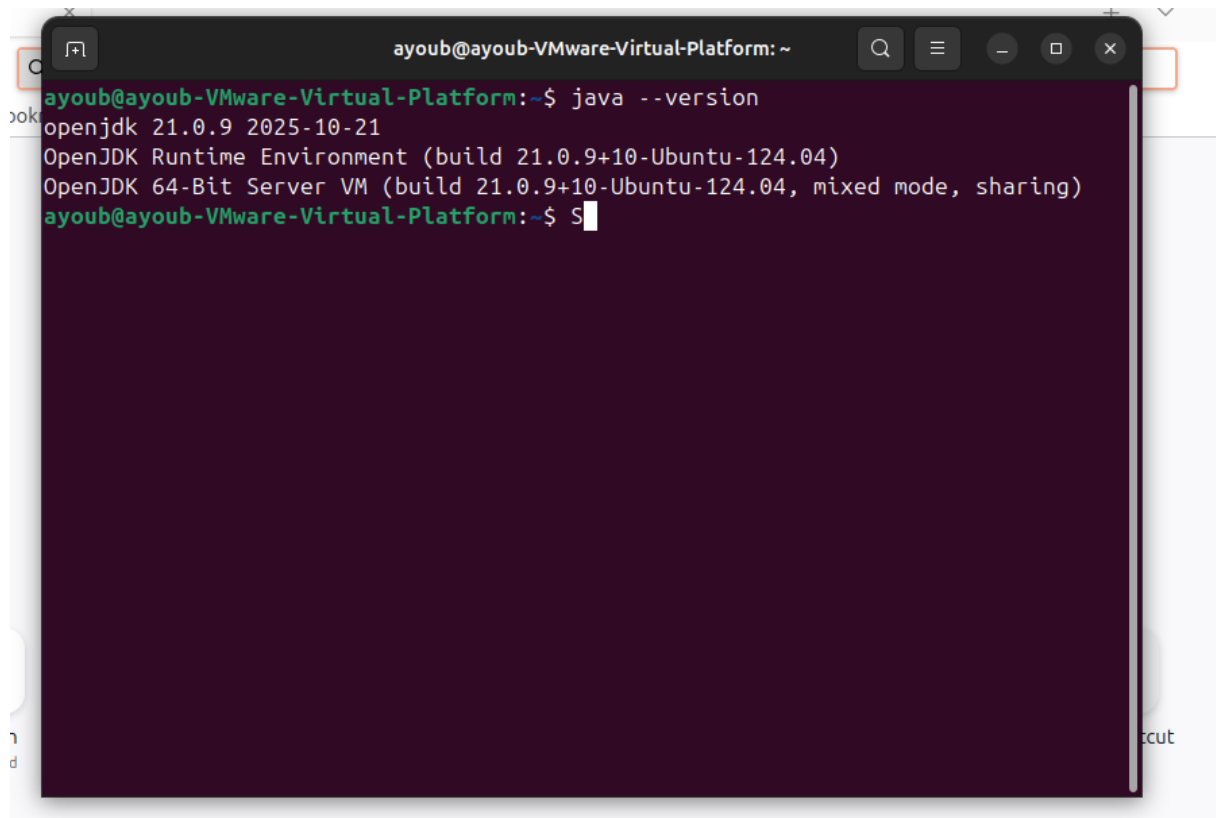
Assignment 4.2: Programming languages

Take screenshots that the following commands work:

`javac --version`

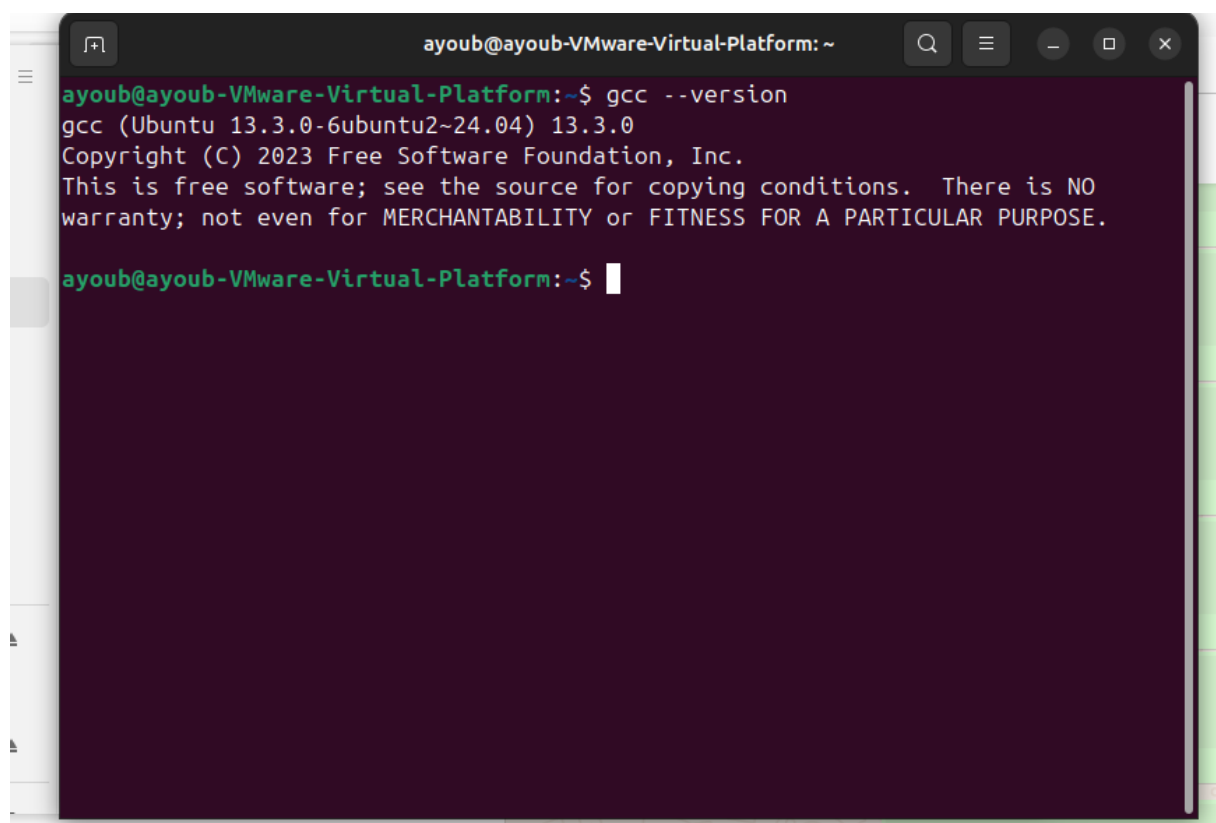
A screenshot of a terminal window with a dark background. The window title bar at the top shows 'ayoub@ayoub-VMware-Virtual-Platform: ~' and standard window control buttons. The terminal text shows the command 'javac --version' being entered and executed, resulting in the output 'javac 21.0.9'. The prompt 'ayoub@ayoub-VMware-Virtual-Platform:~\$' is visible before and after the command.

`java --version`

A terminal window titled 'ayoub@ayoub-VMware-Virtual-Platform: ~' with standard Ubuntu window controls. The prompt is 'ayoub@ayoub-VMware-Virtual-Platform:~\$'. The command 'java --version' has been executed, resulting in the following output: 'openjdk 21.0.9 2025-10-21', 'OpenJDK Runtime Environment (build 21.0.9+10-Ubuntu-124.04)', and 'OpenJDK 64-Bit Server VM (build 21.0.9+10-Ubuntu-124.04, mixed mode, sharing)'. The prompt is now 'ayoub@ayoub-VMware-Virtual-Platform:~\$ S' with a cursor.

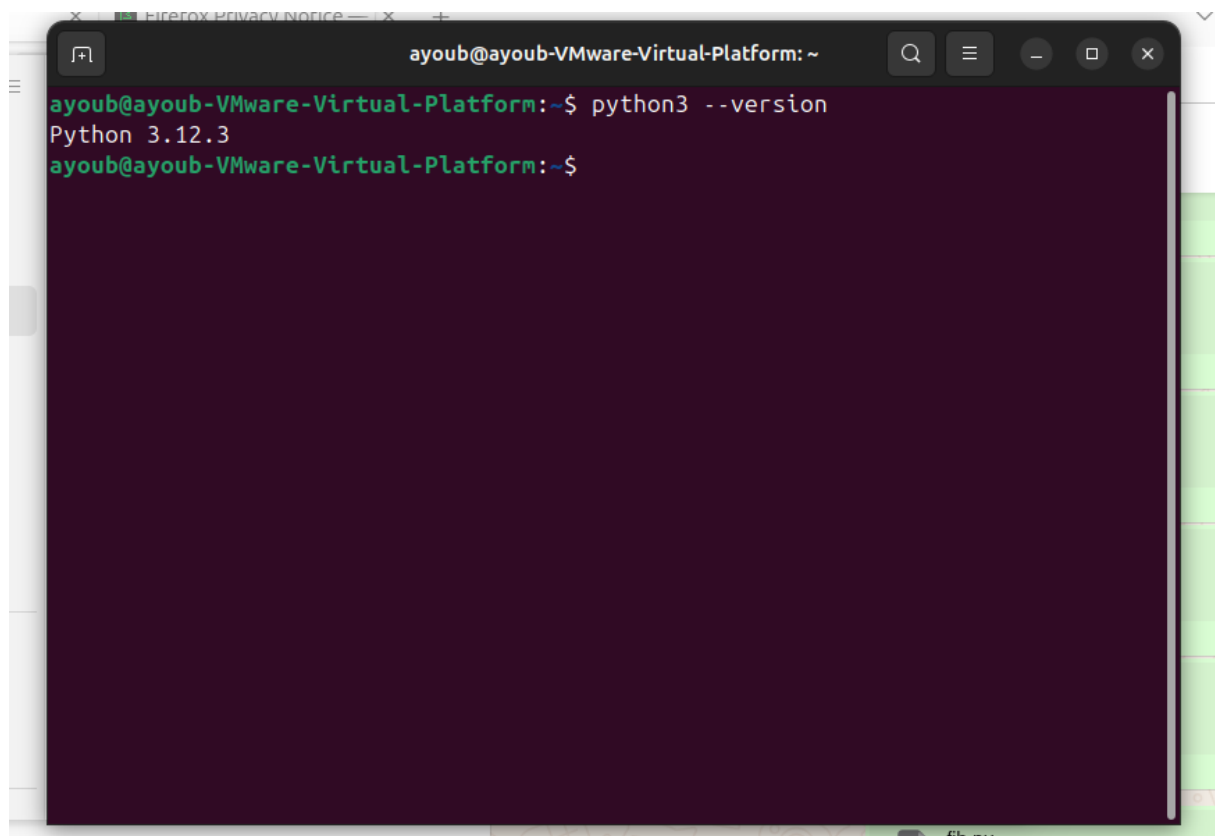
```
ayoub@ayoub-VMware-Virtual-Platform:~$ java --version
openjdk 21.0.9 2025-10-21
OpenJDK Runtime Environment (build 21.0.9+10-Ubuntu-124.04)
OpenJDK 64-Bit Server VM (build 21.0.9+10-Ubuntu-124.04, mixed mode, sharing)
ayoub@ayoub-VMware-Virtual-Platform:~$ S
```

gcc --version

A terminal window titled 'ayoub@ayoub-VMware-Virtual-Platform: ~' with standard Ubuntu window controls. The prompt is 'ayoub@ayoub-VMware-Virtual-Platform:~\$'. The command 'gcc --version' has been executed, resulting in the following output: 'gcc (Ubuntu 13.3.0-6ubuntu2~24.04) 13.3.0', 'Copyright (C) 2023 Free Software Foundation, Inc.', and 'This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.' The prompt is now 'ayoub@ayoub-VMware-Virtual-Platform:~\$' with a cursor.

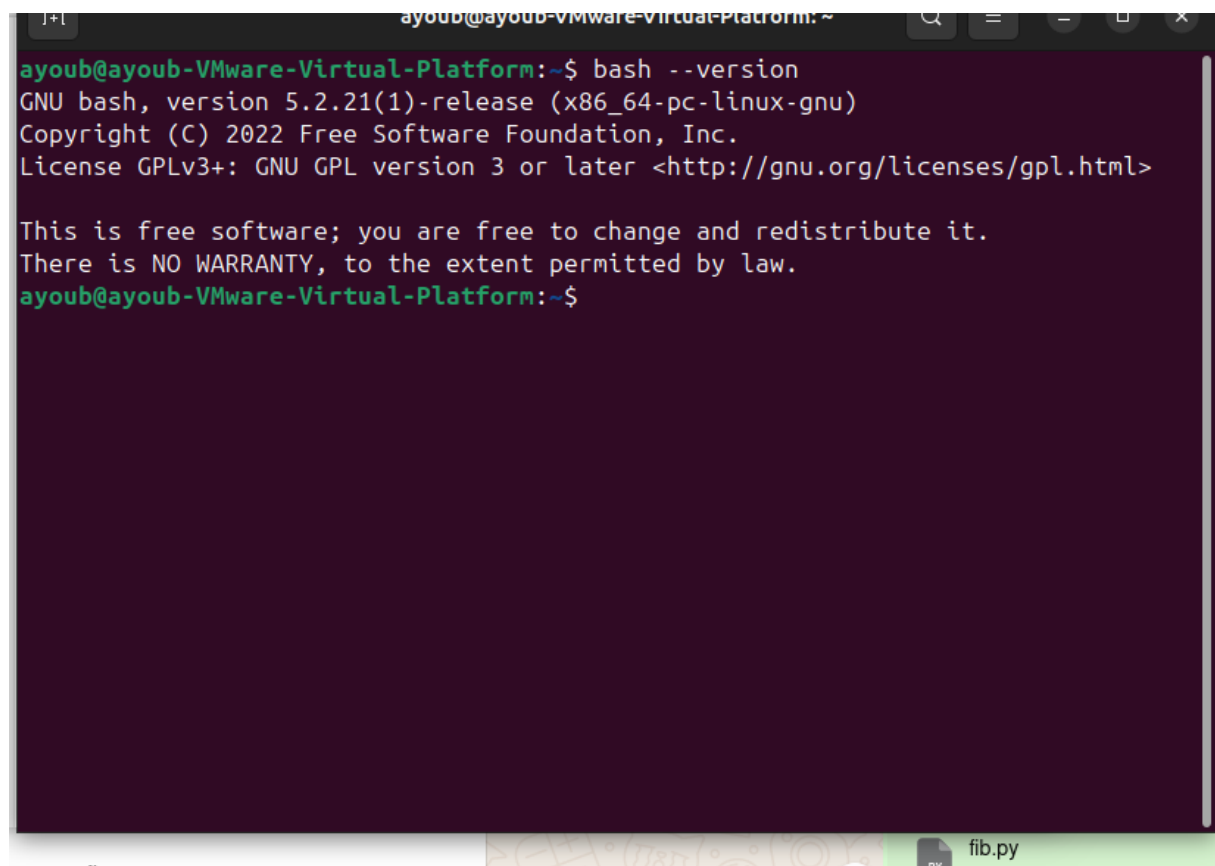
```
ayoub@ayoub-VMware-Virtual-Platform:~$ gcc --version
gcc (Ubuntu 13.3.0-6ubuntu2~24.04) 13.3.0
Copyright (C) 2023 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
ayoub@ayoub-VMware-Virtual-Platform:~$
```

python3 --version

A terminal window titled 'ayoub@ayoub-VMware-Virtual-Platform: ~' with search, menu, and window control icons. The prompt is 'ayoub@ayoub-VMware-Virtual-Platform:~\$'. The command 'python3 --version' has been entered and executed, resulting in the output 'Python 3.12.3'. The prompt is now 'ayoub@ayoub-VMware-Virtual-Platform:~\$'.

```
ayoub@ayoub-VMware-Virtual-Platform:~$ python3 --version
Python 3.12.3
ayoub@ayoub-VMware-Virtual-Platform:~$
```

bash --version

A terminal window titled 'ayoub@ayoub-VMware-Virtual-Platform: ~' with search, menu, and window control icons. The prompt is 'ayoub@ayoub-VMware-Virtual-Platform:~\$'. The command 'bash --version' has been entered and executed, resulting in the output: '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.', and 'There is NO WARRANTY, to the extent permitted by law.' The prompt is now 'ayoub@ayoub-VMware-Virtual-Platform:~\$'.

```
ayoub@ayoub-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.
ayoub@ayoub-VMware-Virtual-Platform:~$
```


Assignment 4.3: Compile

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

Fib.c en Fibonacci.java

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

Fib.c, nadat die gecompileerd is dan wordt die leesbaar voor de CPU

Which source code files are compiled to byte code?

Fibonacci.java

Which source code files are interpreted by an interpreter?

Fib.py (python interpreter) en Fib.sh (bash interpreter)

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

Fib.c want hij heeft geen interpreter nodig om te runnen.

How do I run a Java program?

Eerst het programma compilen door javac Fibonacci.java, dat creëert Fibonacci.class en dat kun je runnen.

How do I run a Python program?

Die kun je gelijk runnen zonder te compilen.

How do I run a C program?

Door de file te compilen met GCC en dat maakt een executable file.

How do I run a Bash script?

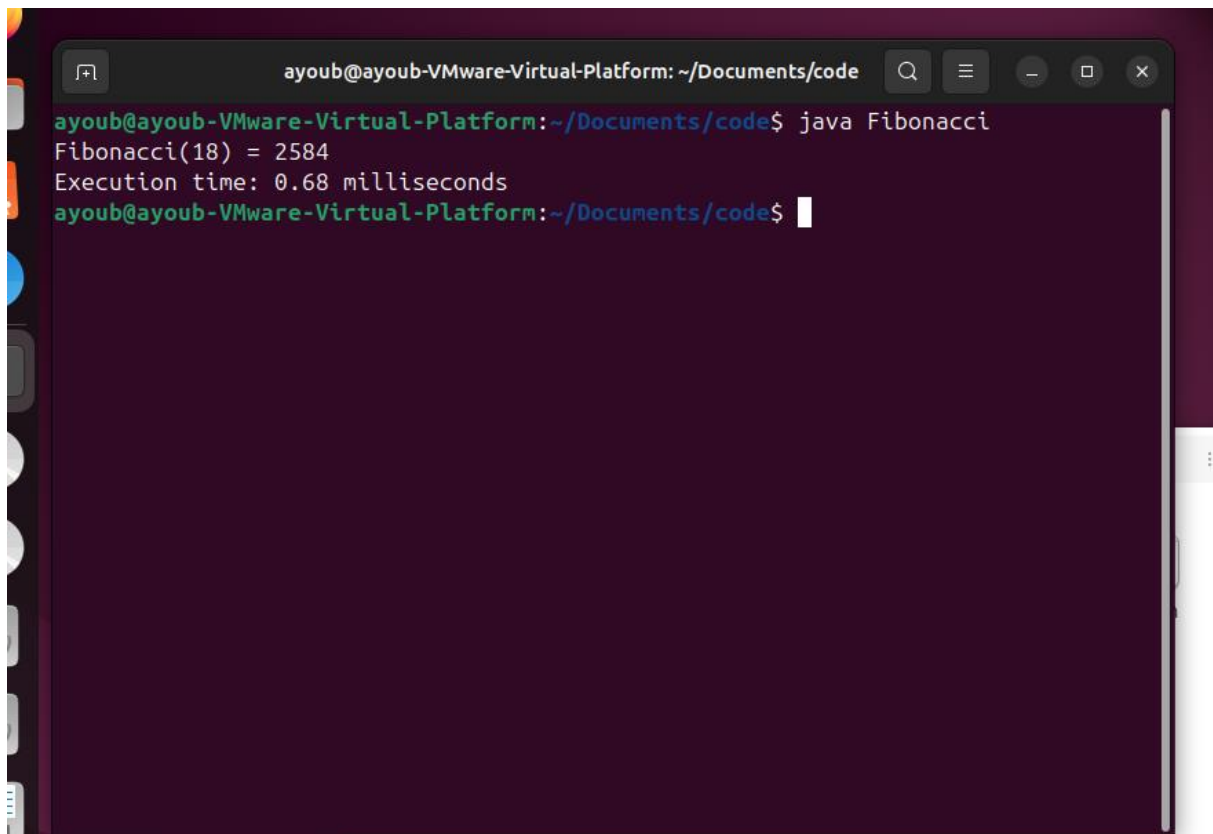
Compilen met chmod en daarna gewoon runnen.

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

Ja, voor C komt een exe file, voor Java komt er een .class file dat je kan runnen, voor python en bash niet.

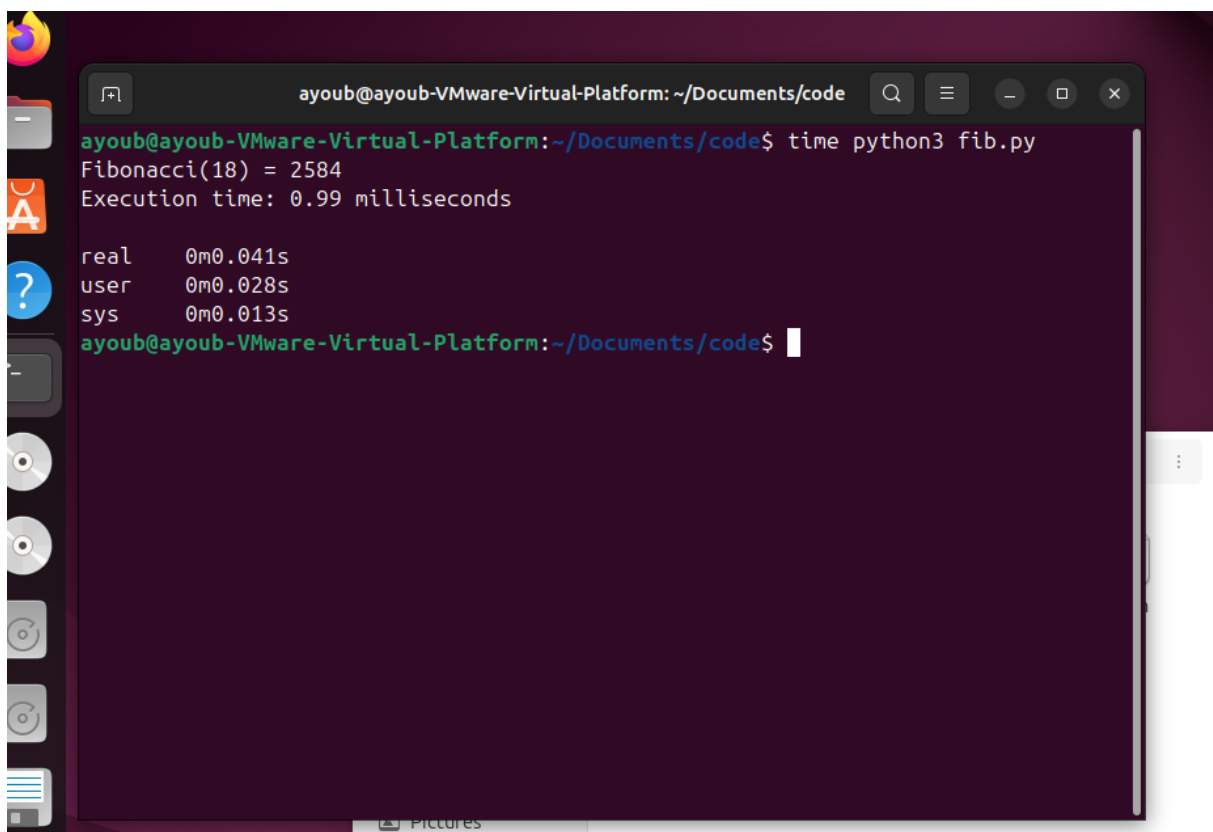
Take relevant screenshots of the following commands:

- Compile the source files where necessary
- Make them executable
- Run them
- Which (compiled) source code file performs the calculation the fastest?
De snelste bleek C te zijn met sys 0.001s echt snel snel



A terminal window titled "ayoub@ayoub-VMware-Virtual-Platform: ~/Documents/code" with search, menu, and window control icons. The prompt is "ayoub@ayoub-VMware-Virtual-Platform:~/Documents/code\$". The user enters "java Fibonacci", and the output is "Fibonacci(18) = 2584" followed by "Execution time: 0.68 milliseconds". The prompt returns.

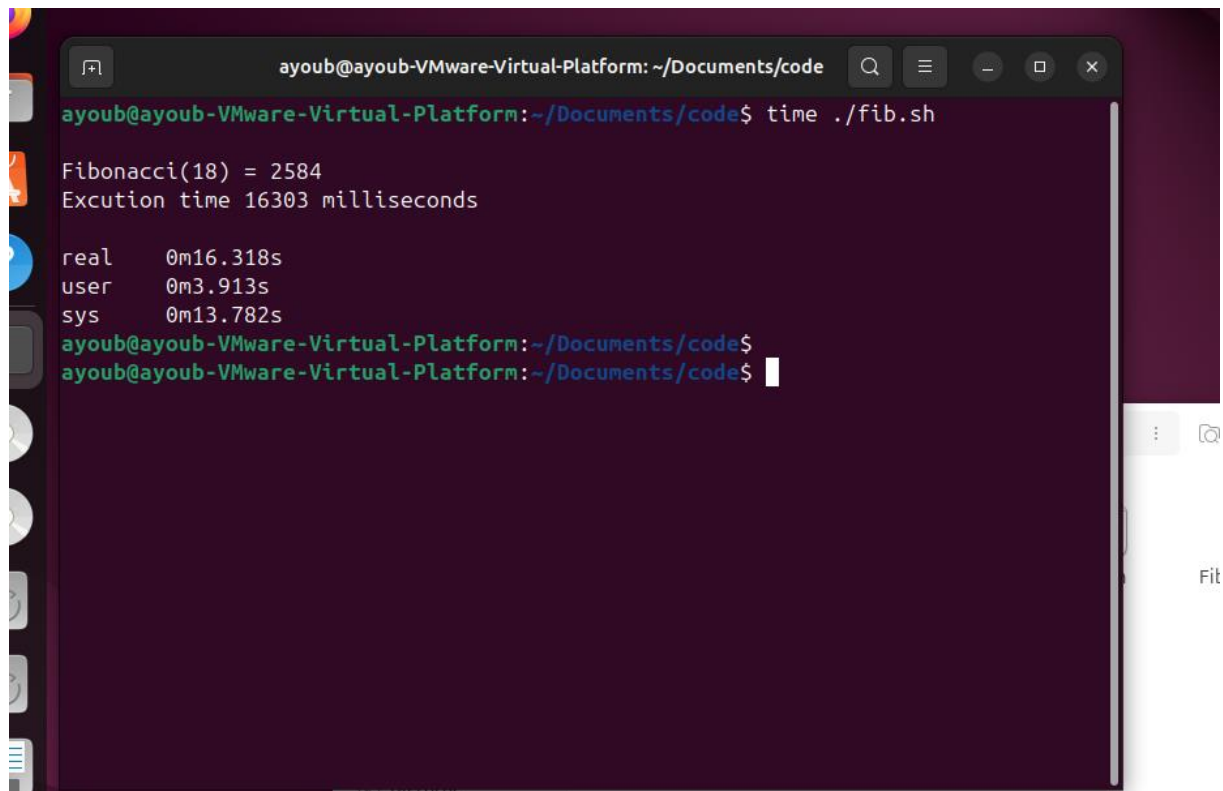
```
ayoub@ayoub-VMware-Virtual-Platform:~/Documents/code$ java Fibonacci
Fibonacci(18) = 2584
Execution time: 0.68 milliseconds
ayoub@ayoub-VMware-Virtual-Platform:~/Documents/code$
```



A terminal window titled "ayoub@ayoub-VMware-Virtual-Platform: ~/Documents/code" with search, menu, and window control icons. The prompt is "ayoub@ayoub-VMware-Virtual-Platform:~/Documents/code\$". The user enters "time python3 fib.py", and the output is "Fibonacci(18) = 2584" followed by "Execution time: 0.99 milliseconds". Then, a table of timing data is shown: "real 0m0.041s", "user 0m0.028s", and "sys 0m0.013s". The prompt returns.

```
ayoub@ayoub-VMware-Virtual-Platform:~/Documents/code$ time python3 fib.py
Fibonacci(18) = 2584
Execution time: 0.99 milliseconds

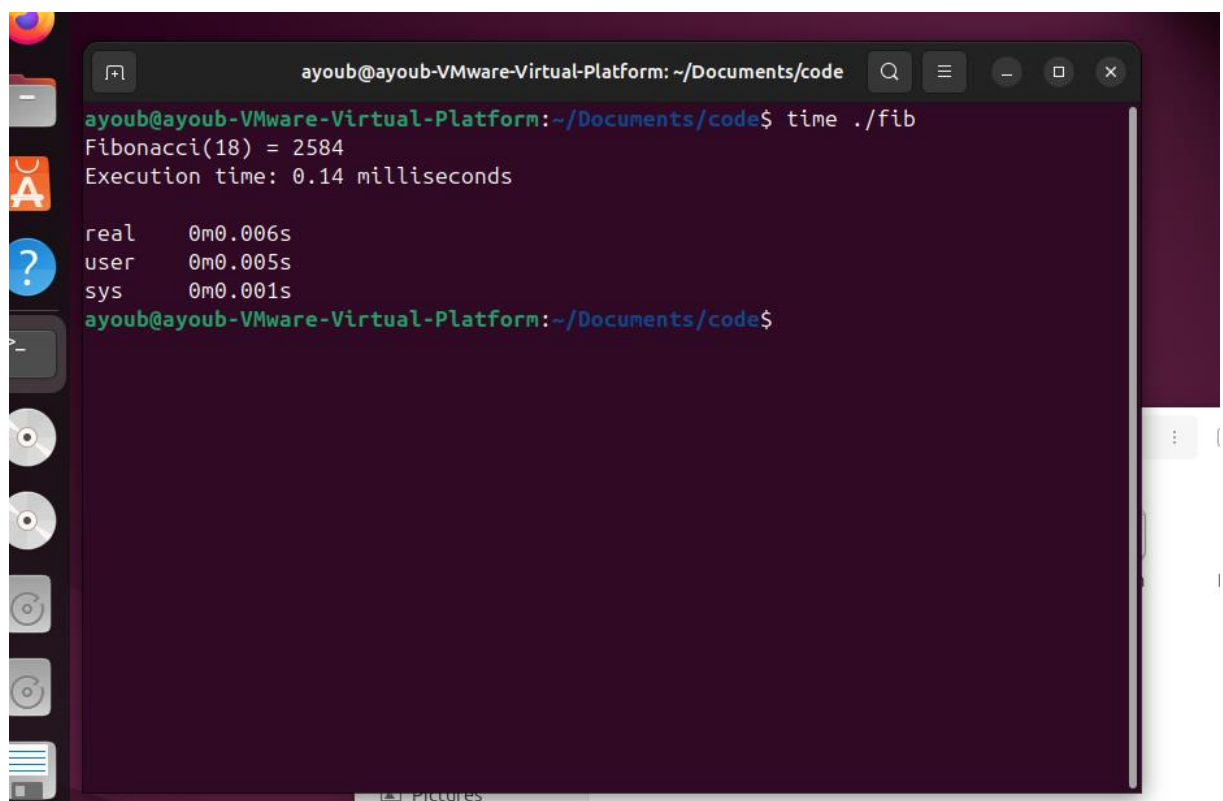
real    0m0.041s
user    0m0.028s
sys     0m0.013s
ayoub@ayoub-VMware-Virtual-Platform:~/Documents/code$
```



A terminal window titled "ayoub@ayoub-VMware-Virtual-Platform: ~/Documents/code". The prompt is "ayoub@ayoub-VMware-Virtual-Platform:~/Documents/code\$". The user enters "time ./fib.sh". The output is:

```
Fibonacci(18) = 2584
Execution time 16303 milliseconds

real    0m16.318s
user    0m3.913s
sys     0m13.782s
ayoub@ayoub-VMware-Virtual-Platform:~/Documents/code$
ayoub@ayoub-VMware-Virtual-Platform:~/Documents/code$
```



A terminal window titled "ayoub@ayoub-VMware-Virtual-Platform: ~/Documents/code". The prompt is "ayoub@ayoub-VMware-Virtual-Platform:~/Documents/code\$". The user enters "time ./fib". The output is:

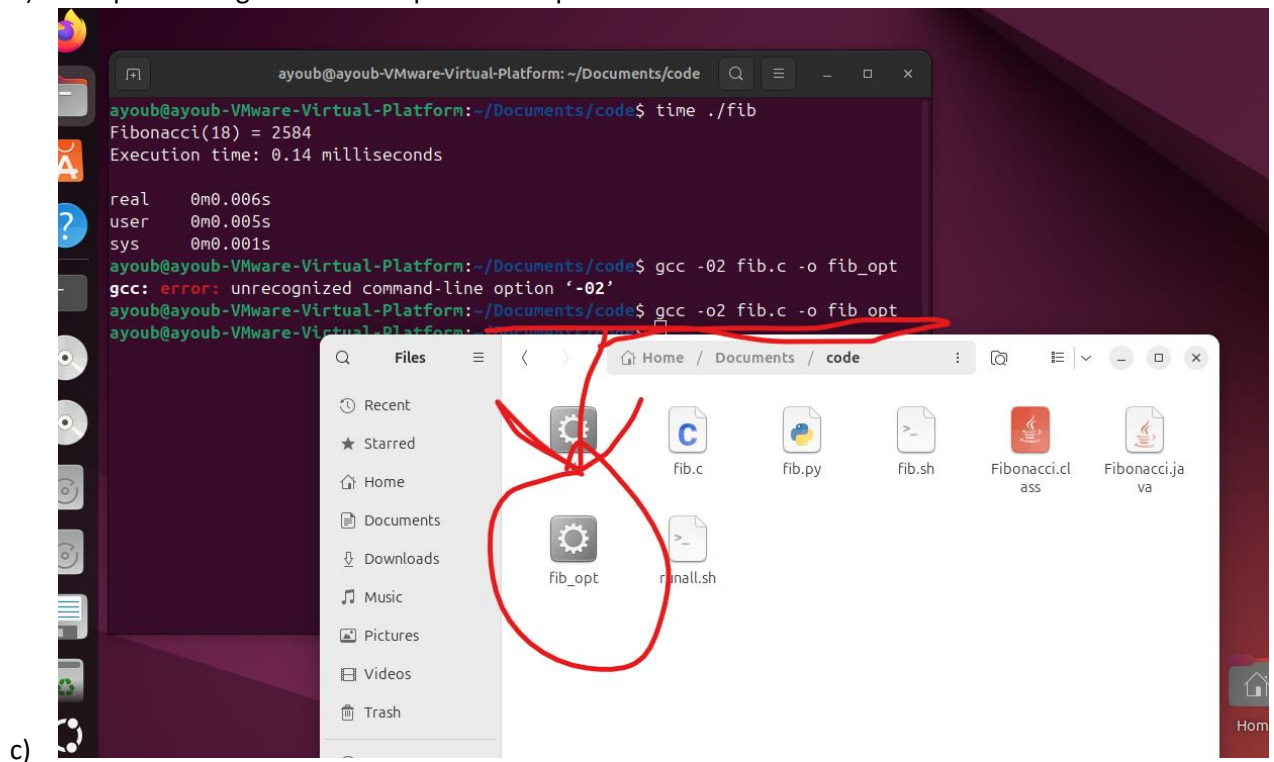
```
Fibonacci(18) = 2584
Execution time: 0.14 milliseconds

real    0m0.006s
user    0m0.005s
sys     0m0.001s
ayoub@ayoub-VMware-Virtual-Platform:~/Documents/code$
```

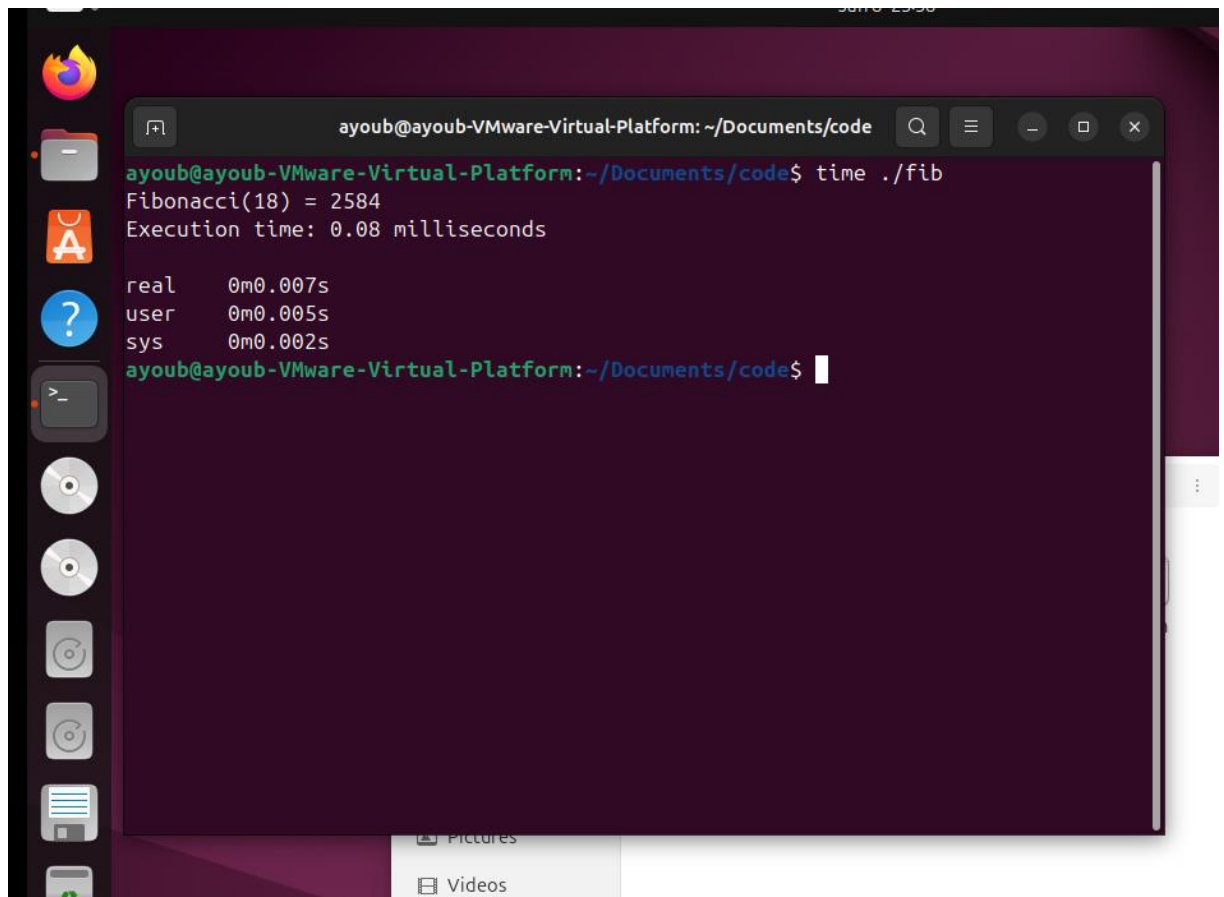

Assignment 4.4: Optimize

Take relevant screenshots of the following commands:

- 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.
- Compile **fib.c** again with the optimization parameters



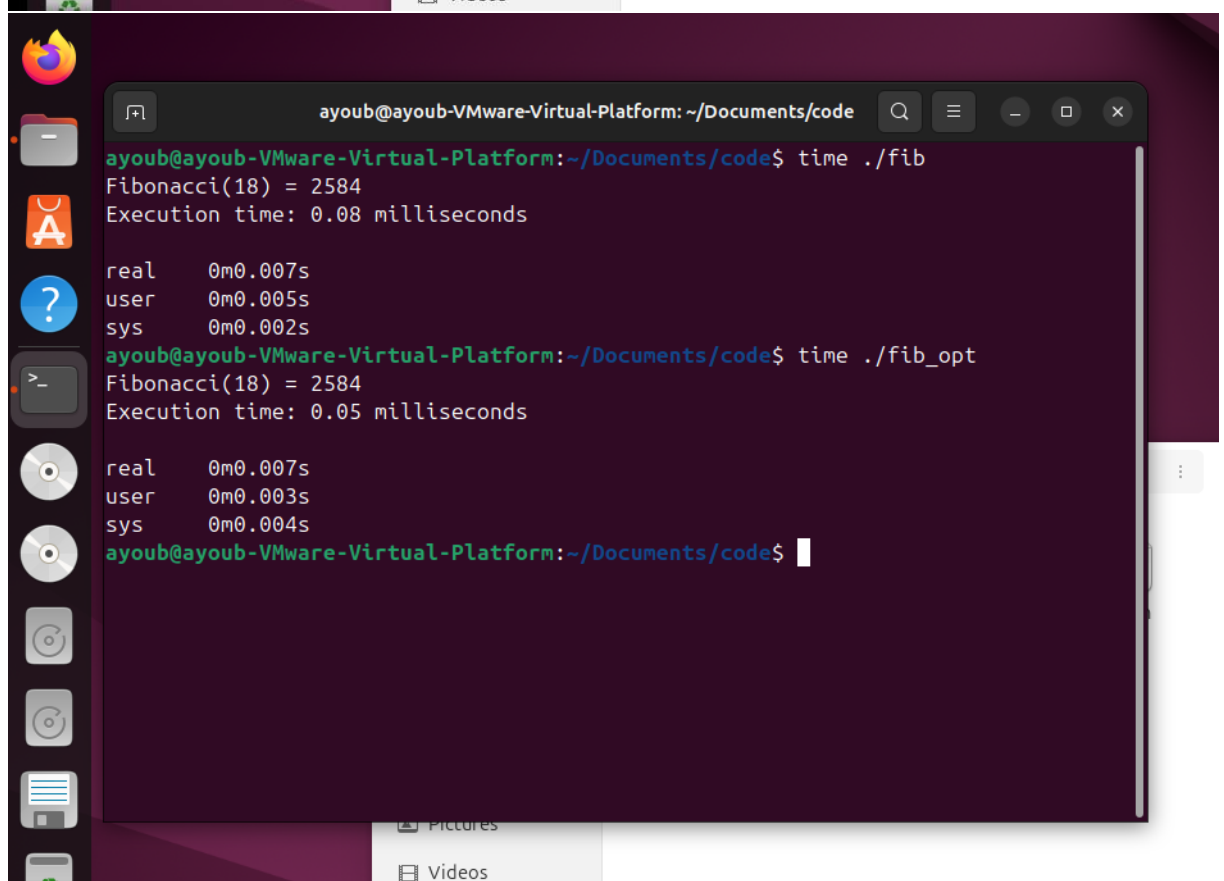
- Run the newly compiled program. Is it true that it now performs the calculation faster?



A terminal window titled "ayoub@ayoub-VMware-Virtual-Platform: ~/Documents/code" with standard window controls. The terminal shows the execution of a program named "fib". The output displays the 18th Fibonacci number as 2584 and the execution time as 0.08 milliseconds. A timing breakdown shows a real time of 0m0.007s, user time of 0m0.005s, and system time of 0m0.002s. The prompt is ready for the next command.

```
ayoub@ayoub-VMware-Virtual-Platform: ~/Documents/code$ time ./fib
Fibonacci(18) = 2584
Execution time: 0.08 milliseconds

real    0m0.007s
user    0m0.005s
sys     0m0.002s
ayoub@ayoub-VMware-Virtual-Platform: ~/Documents/code$
```



The same terminal window as above, now showing the execution of an optimized program named "fib_opt". The output shows the same Fibonacci number (2584) but with a faster execution time of 0.05 milliseconds. The timing breakdown shows a real time of 0m0.007s, user time of 0m0.003s, and system time of 0m0.004s. The prompt is ready for the next command.

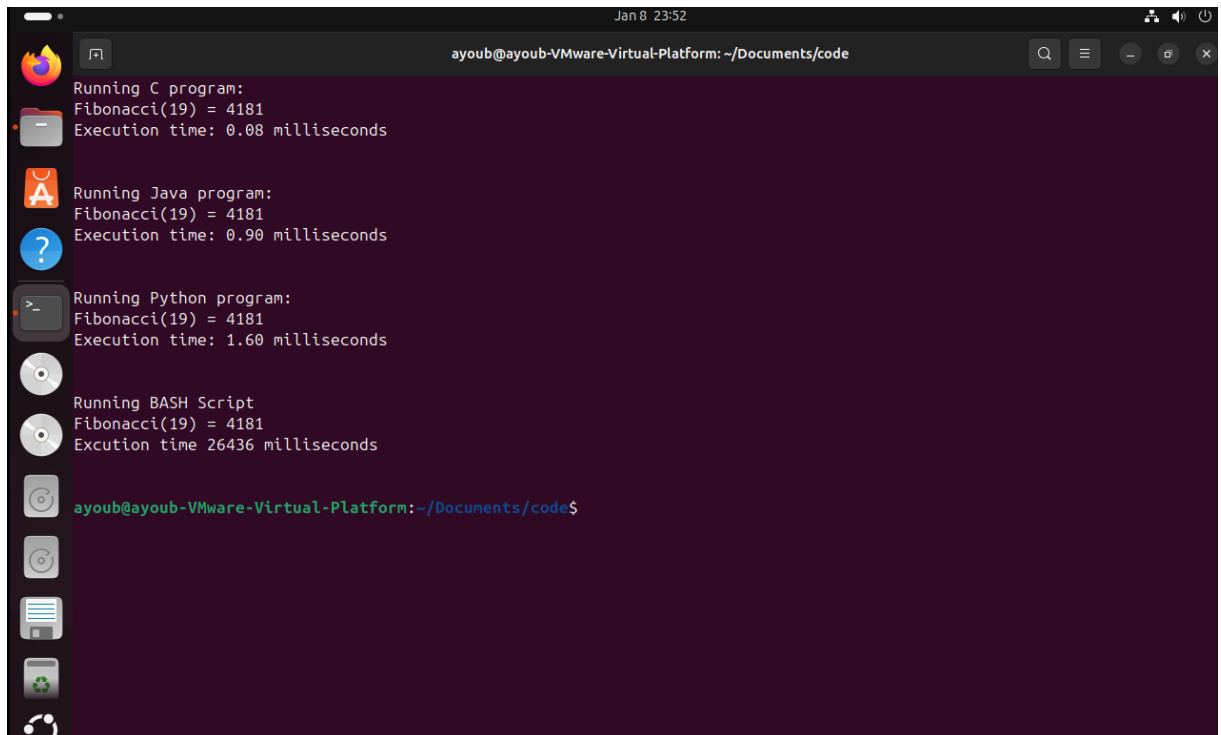
```
ayoub@ayoub-VMware-Virtual-Platform: ~/Documents/code$ time ./fib
Fibonacci(18) = 2584
Execution time: 0.08 milliseconds

real    0m0.007s
user    0m0.005s
sys     0m0.002s
ayoub@ayoub-VMware-Virtual-Platform: ~/Documents/code$ time ./fib_opt
Fibonacci(18) = 2584
Execution time: 0.05 milliseconds

real    0m0.007s
user    0m0.003s
sys     0m0.004s
ayoub@ayoub-VMware-Virtual-Platform: ~/Documents/code$
```

Zoals je kan zien runt die intotaal maar 0.03 miliseconden sneller dus inprincipe is die wel sneller ja.

- 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.



```
ayoub@ayoub-VMware-Virtual-Platform: ~/Documents/code
Running C program:
Fibonacci(19) = 4181
Execution time: 0.08 milliseconds

Running Java program:
Fibonacci(19) = 4181
Execution time: 0.90 milliseconds

Running Python program:
Fibonacci(19) = 4181
Execution time: 1.60 milliseconds

Running BASH Script
Fibonacci(19) = 4181
Execution time 26436 milliseconds

ayoub@ayoub-VMware-Virtual-Platform:~/Documents/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.

Main:

```
mov r1, #2
```

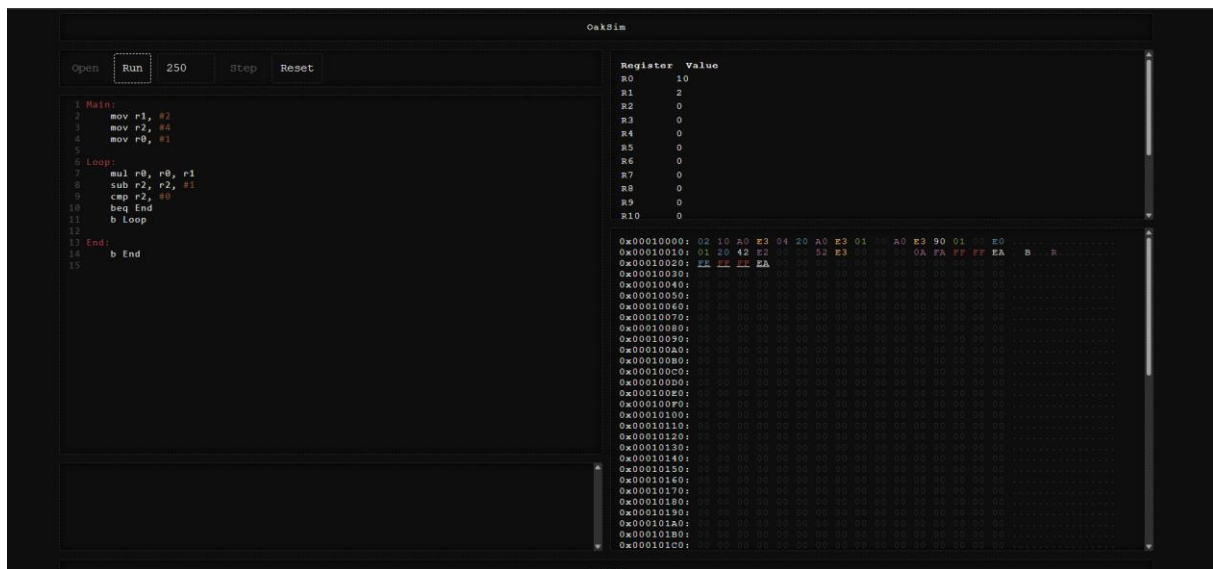
```
mov r2, #4
```

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](#)