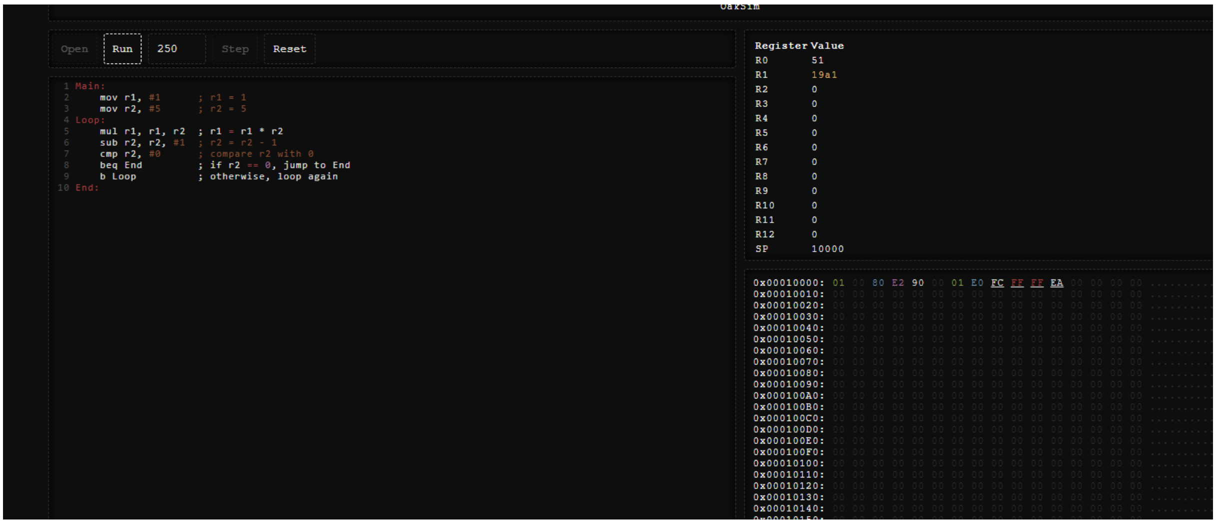
Template Week 4 – Software

Student number: 582031

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:

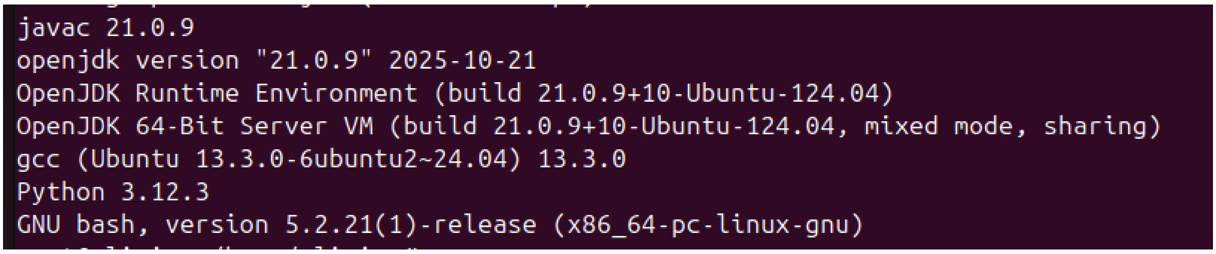
javac --version

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?

Fibonacci.java en fib.c moeten eerst gecompileerd worden voordat je ze kunt uitvoeren.

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

fib.c C code wordt gecompileerd naar native machine code die direct door de processor uitgevoerd kan worden.

Which source code files are compiled to byte code?

Fibonacci.java Java wordt gecompileerd naar bytecode (.class bestand) die door de Java Virtual Machine (JVM) wordt uitgevoerd.

Which source code files are interpreted by an interpreter?

fib.py (Python interpreter) en fib.sh (Bash interpreter) worden regel voor regel geïnterpreteerd tijdens het uitvoeren.

How do I run a Java program?

Eerst compileren met javac Fibonacci.java, dan uitvoeren met java Fibonacci

How do I run a Python program?

python3 fib.py

How do I run a C program?

Eerst compileren met gcc fib.c -o fib, dan uitvoeren met ./fib

How do I run a Bash script?

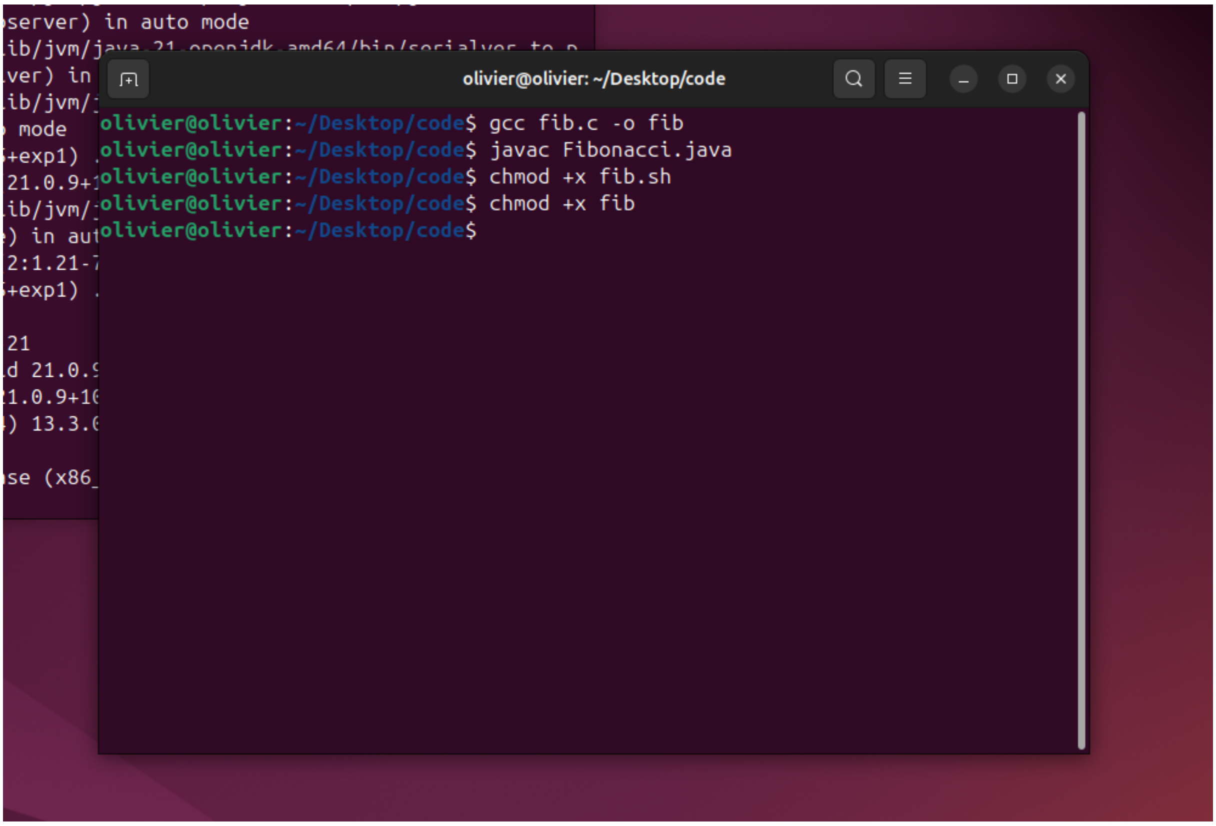
Executable maken met chmod +x fib.sh, dan uitvoeren met ./fib.sh

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

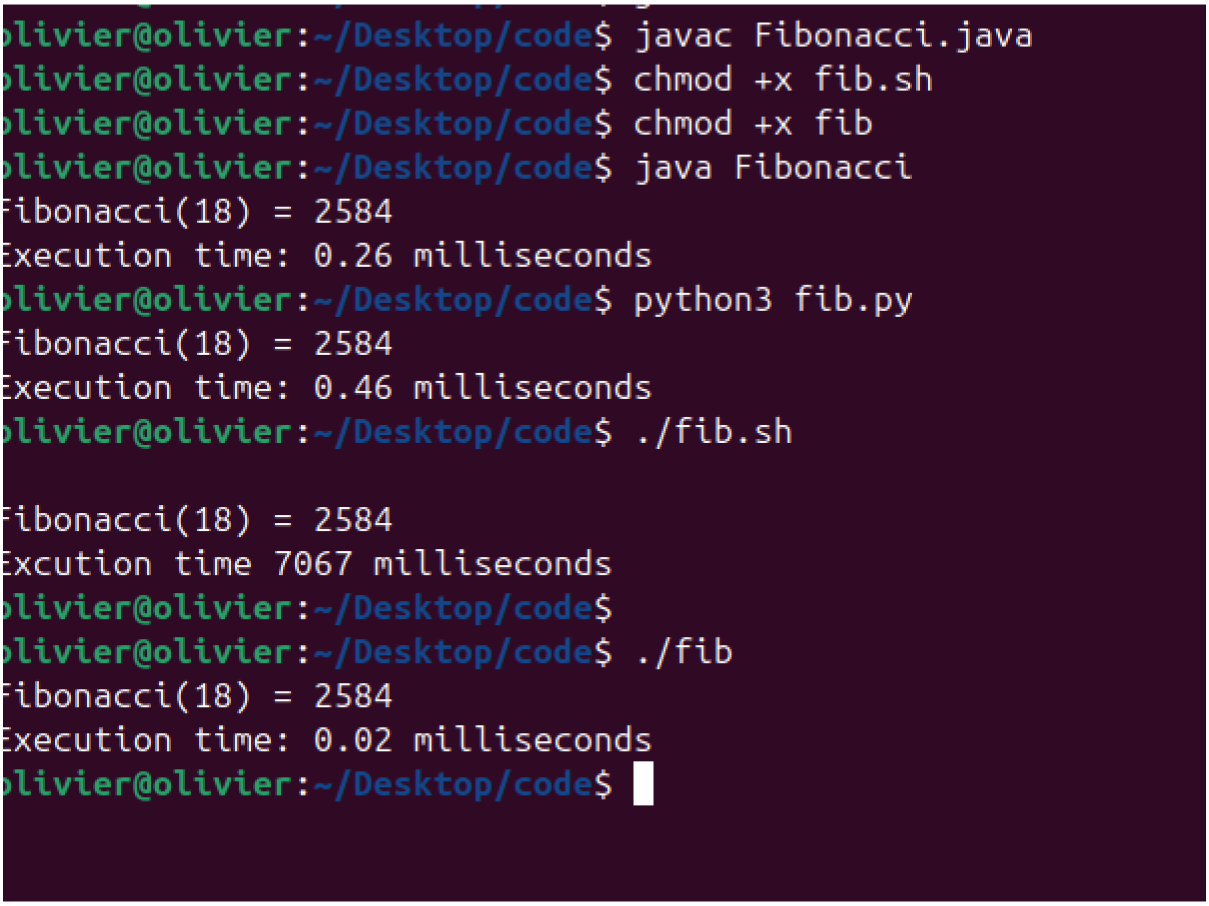
Fibonacci.java maakt Fibonacci.class en fib.c maakt fib (ligt aan -o)

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?



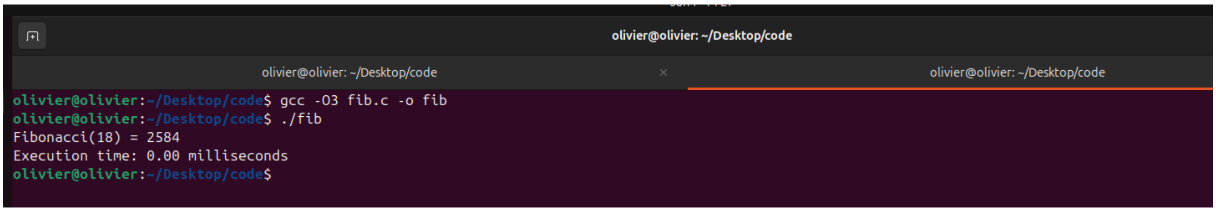
De C code was het snelst.

Assignment 4.4: Optimize

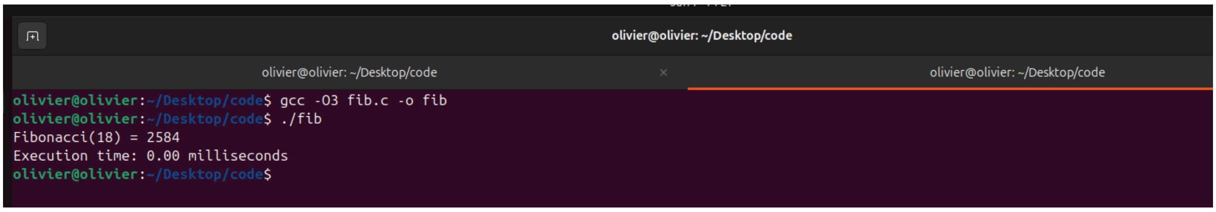
Take relevant screenshots of the following commands:

a) In de man page van gcc kun je zoeken met / en dan -O typen. De compiler heeft verschillende optimalisatie levels: -O0 (geen), -O1, -O2, -O3 en -Ofast. Hoe hoger het getal, hoe meer optimalisaties de compiler toepast. -O3 is de beste keuze voor maximale snelheid omdat deze alle beschikbare optimalisaties activeert zonder risico's met precisie.

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

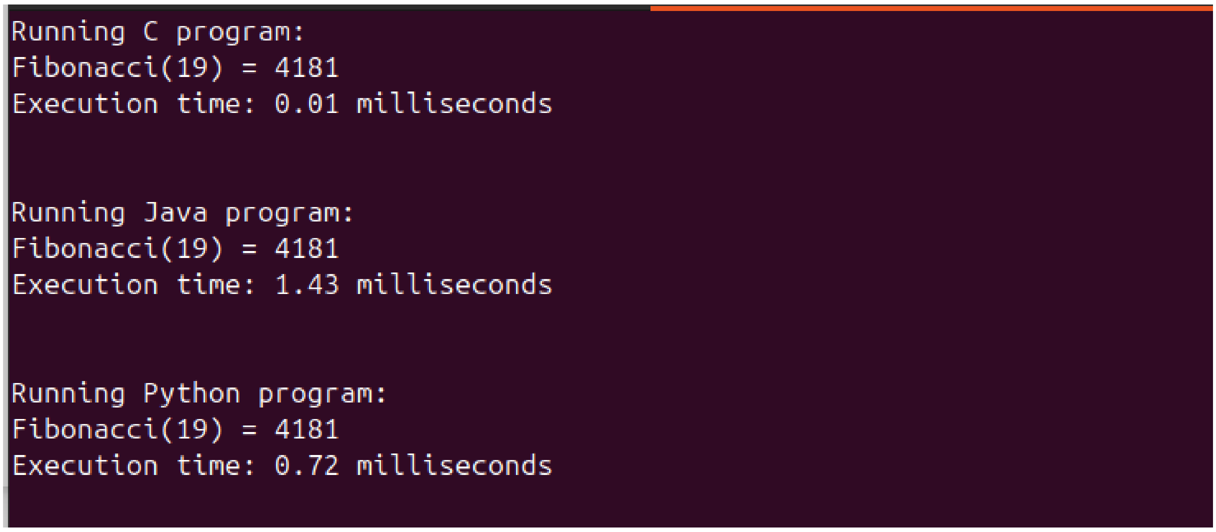


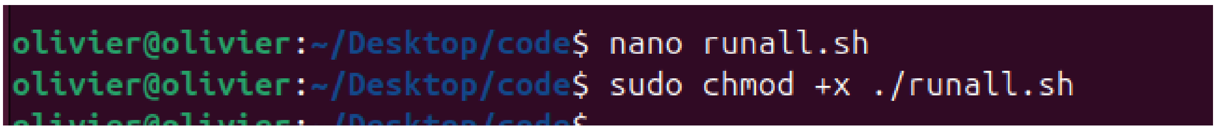
c) Run the newly compiled program. Is it true that it now performs the calculation faster?



Ja, het voert nu sneller uit dan 0,00ms

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.





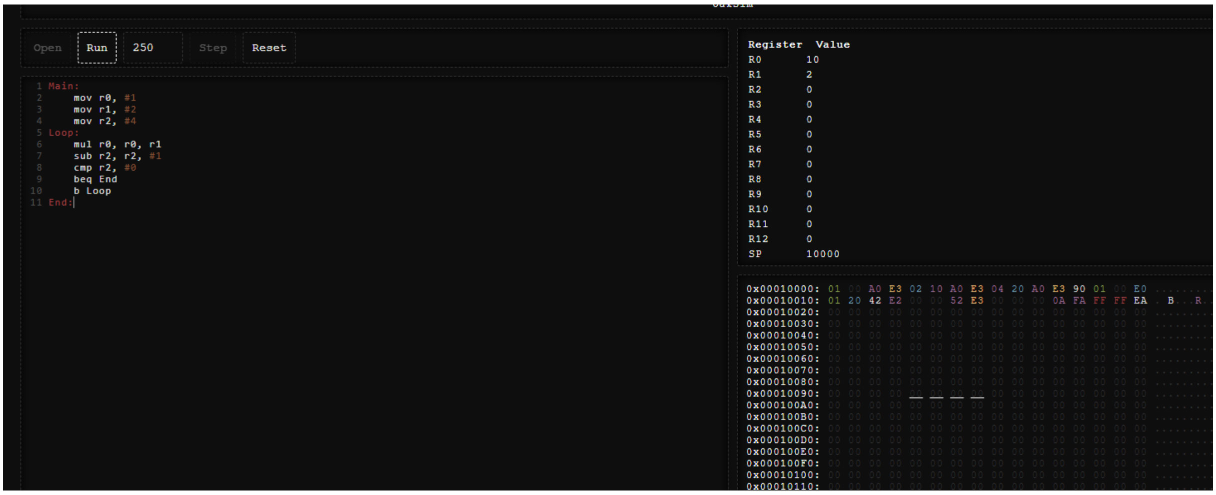
Assignment 4.5: More ARM Assembly

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**