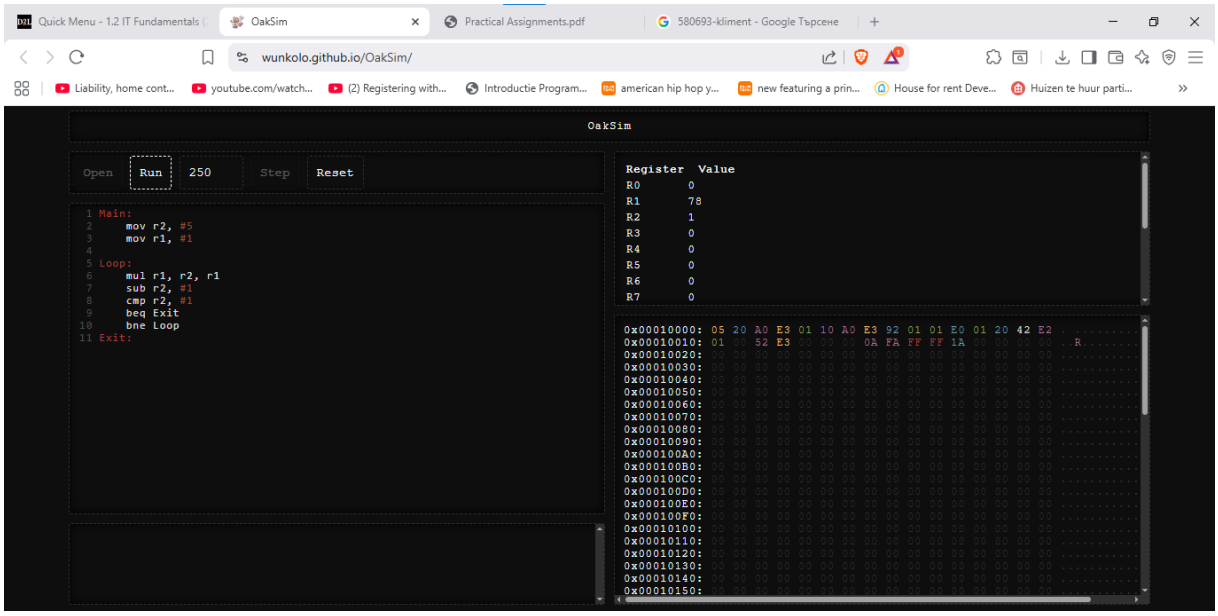
# Template Week 4 – Software

Student number:

580693

**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:

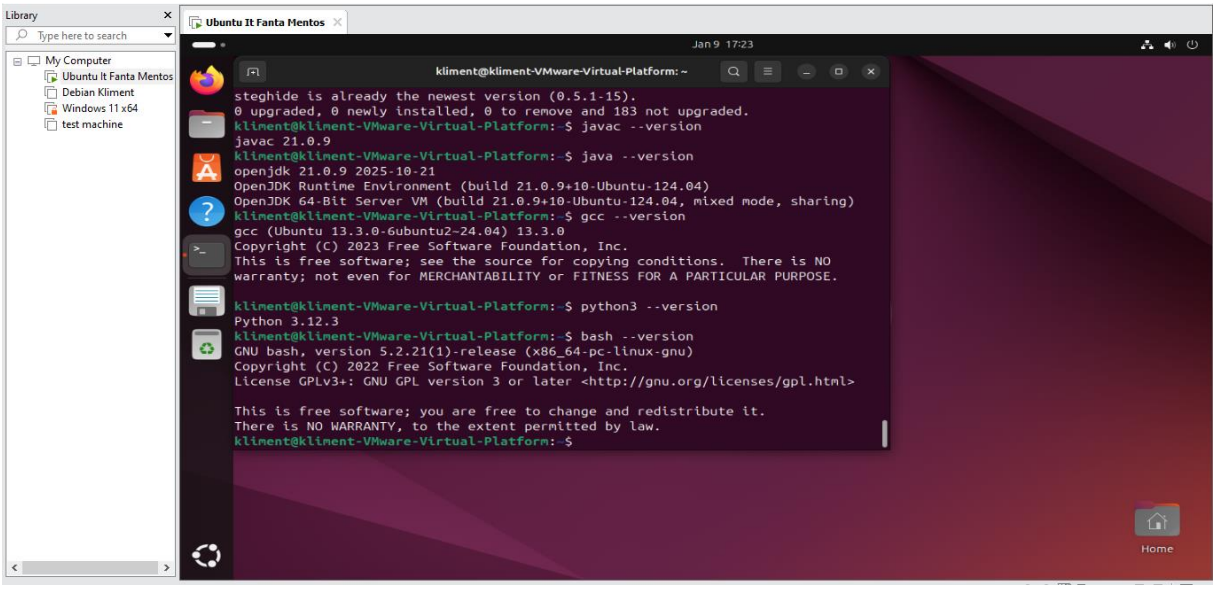
javac --version

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?

The java and c files must be compiled before executing them.

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

The python and bash files are compiled into machine code and then directly executable.

Which source code files are compiled to byte code?

Java and C are compiled to byte code.

Which source code files are interpreted by an interpreter?

Python and Bash are interpreted by an interpreter.

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

The c file will do the calculation the fastest.

How do I run a Java program?

Using the javac command to first compile it, then running the java “nameOfFile” command.

How do I run a Python program?

Using the python3 command – it doesn’t need to be compiled beforehand.

How do I run a C program?

Using the gcc command first to compile the code, then executing the executable file made from the compiling.

How do I run a Bash script?

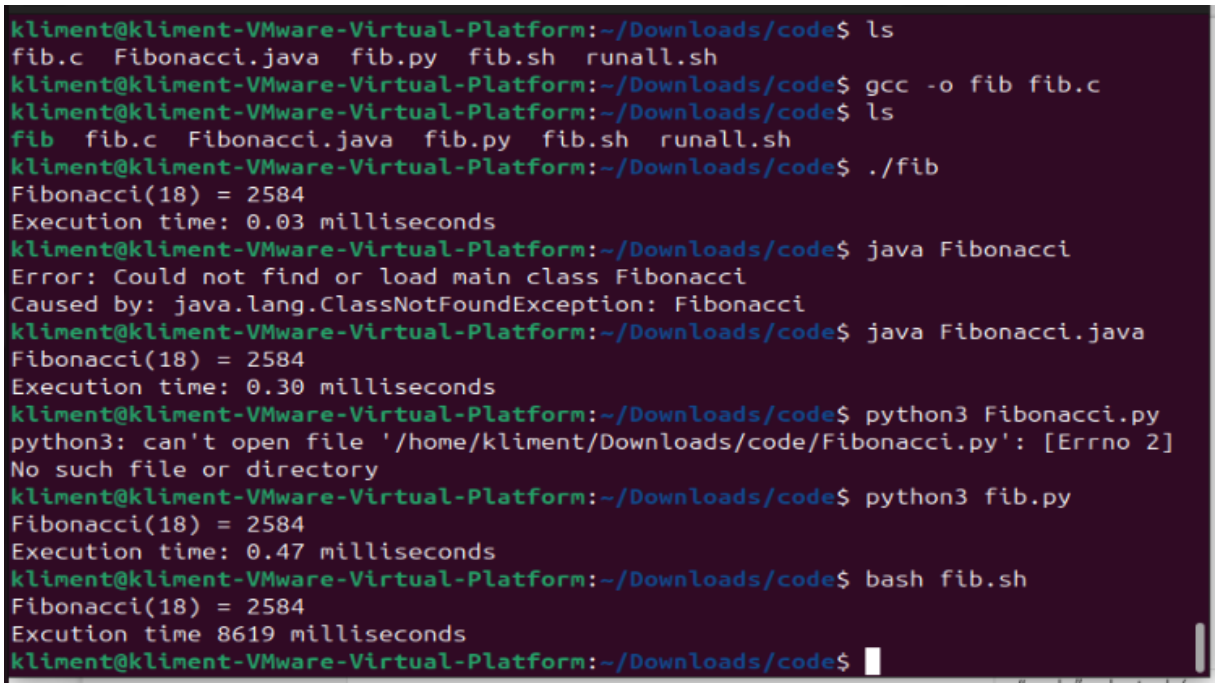
Using the bash command or just ./”nameOfFile”.

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

Yes, a .class file will be created for the java file and an executable file will be created for the c file.

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.

```
kliment@kliment-VMware-Virtual-Platform:~/Downloads/code$ gcc -03 fib.c -o fib
kliment@kliment-VMware-Virtual-Platform:~/Downloads/code$
```

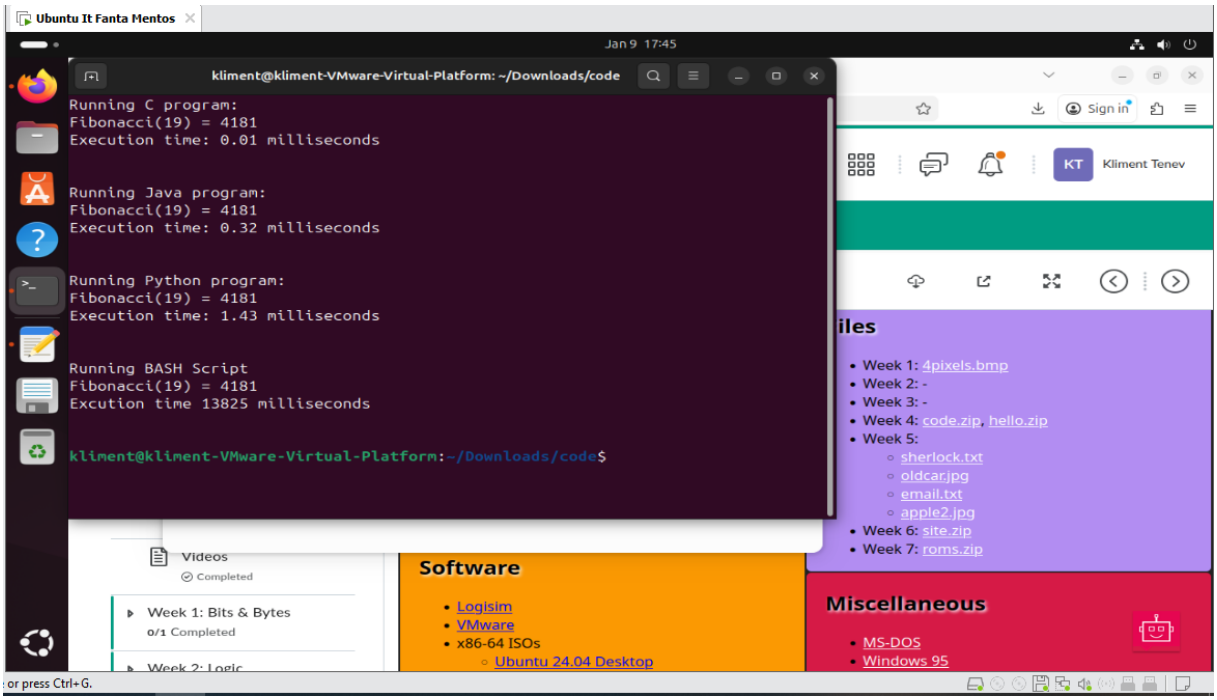
b) Compile **fib.c** again with the optimization parameters

```
kliment@kliment-VMware-Virtual-Platform:~/Downloads/code$ ./fib
Fibonacci(18) = 2584
Execution time: 0.01 milliseconds
kliment@kliment-VMware-Virtual-Platform:~/Downloads/code$
```

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

Yes it went from 0.03 to 0.01 milliseconds

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.

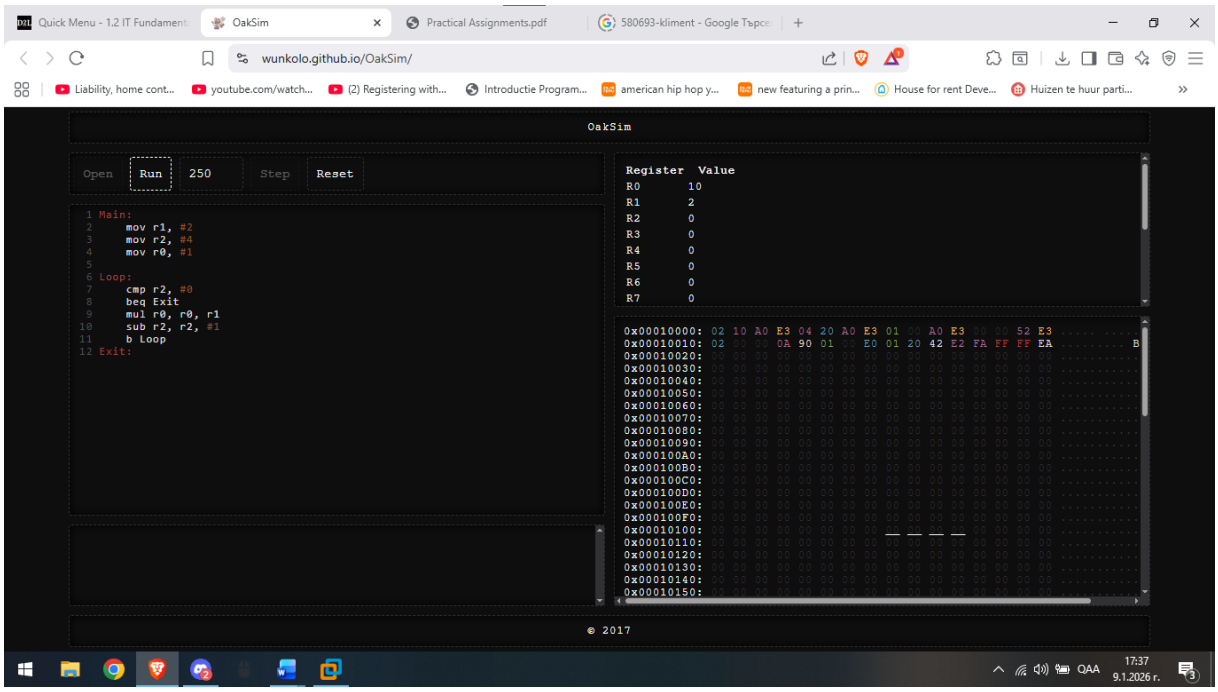
```
Main:
mov r1, #2
mov r2, #4

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**