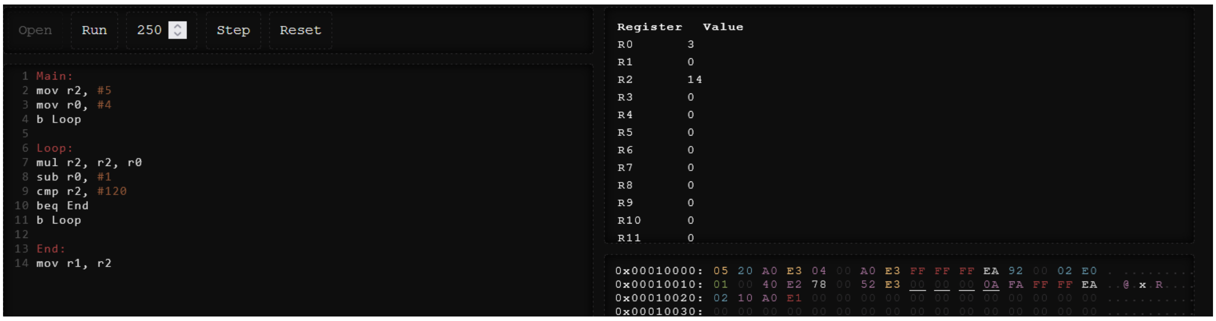
# Template Week 4 – Software

Student number: 579675

### Assignment 4.1: ARM assembly

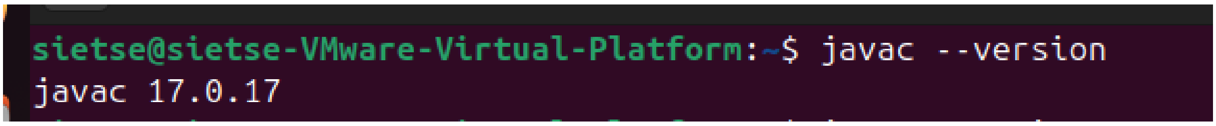
Screenshot of working assembly code of factorial calculation:



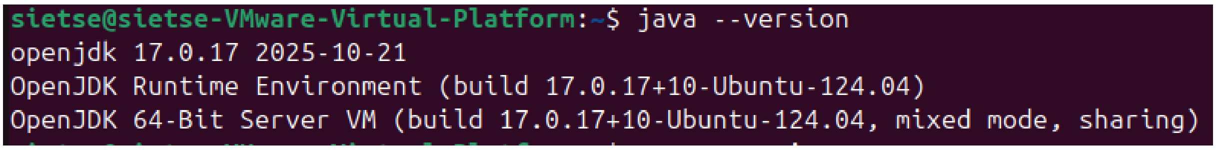

### Assignment 4.2: Programming languages

Take screenshots that the following commands work:

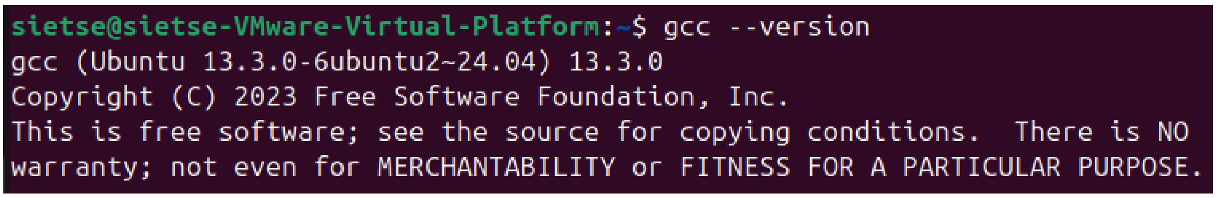
javac –version




java –version




gcc –version




python3 –version


bash –version

```
sietse@sietse-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.
```

**Assignment 4.3: Compile**

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

- Java source code needs to be compiled before running
- C source code needs to be compiled before running
- Python3 source code doesn't need to be compiled
- Bash script source code doesn't need to be compiled

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

- C is compiled into Machine code

Which source code files are compiled to byte code?

- Java is not compiled into Byte code

Which source code files are interpreted by an interpreter?

- Python is interpreted by an python interpreter
- Bash is interpreted by an 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 should work the fastest being compiled to machine code which directly gets executed on the processor

How do I run a Java program?

- javac Fibonacci.java
- java Fibonacci

How do I run a Python program?

- python3 fib.py

How do I run a C program?

- gcc fib.c -o fib
- ./fib

How do I run a Bash script?

- sudo chmod a+x fib.sh
- sudo fib.sh

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

- The files that get compiled, compiling c wil make fib.c -> fib, and compiling java wil make Fibonacci.java -> Fibonacci.class

Take relevant screenshots of the following commands:

- Compile the source files where necessary

```
sietse@sietse-VMware-Virtual-Platform:~/code$ javac Fibonacci.java
```

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

- Make them executable

```
sietse@sietse-VMware-Virtual-Platform:~/code$ sudo chmod a+x fib.sh
```

```
sietse@sietse-VMware-Virtual-Platform:~/code$ sudo chmod a+x runall.sh
```

- Run them

```
sietse@sietse-VMware-Virtual-Platform:~/code$ sudo runall.sh
```

- Which (compiled) source code file performs the calculation the fastest?
    - C is indeed the fastest to be executed, as guessed previously

```
Running C program:
Fibonacci(19) = 4181
Execution time: 0.03 milliseconds


Running Java program:
Fibonacci(19) = 4181
Execution time: 0.41 milliseconds


Running Python program:
Fibonacci(19) = 4181
Execution time: 0.67 milliseconds


Running BASH Script
Fibonacci(19) = 4181
Excution time 13864 milliseconds
```

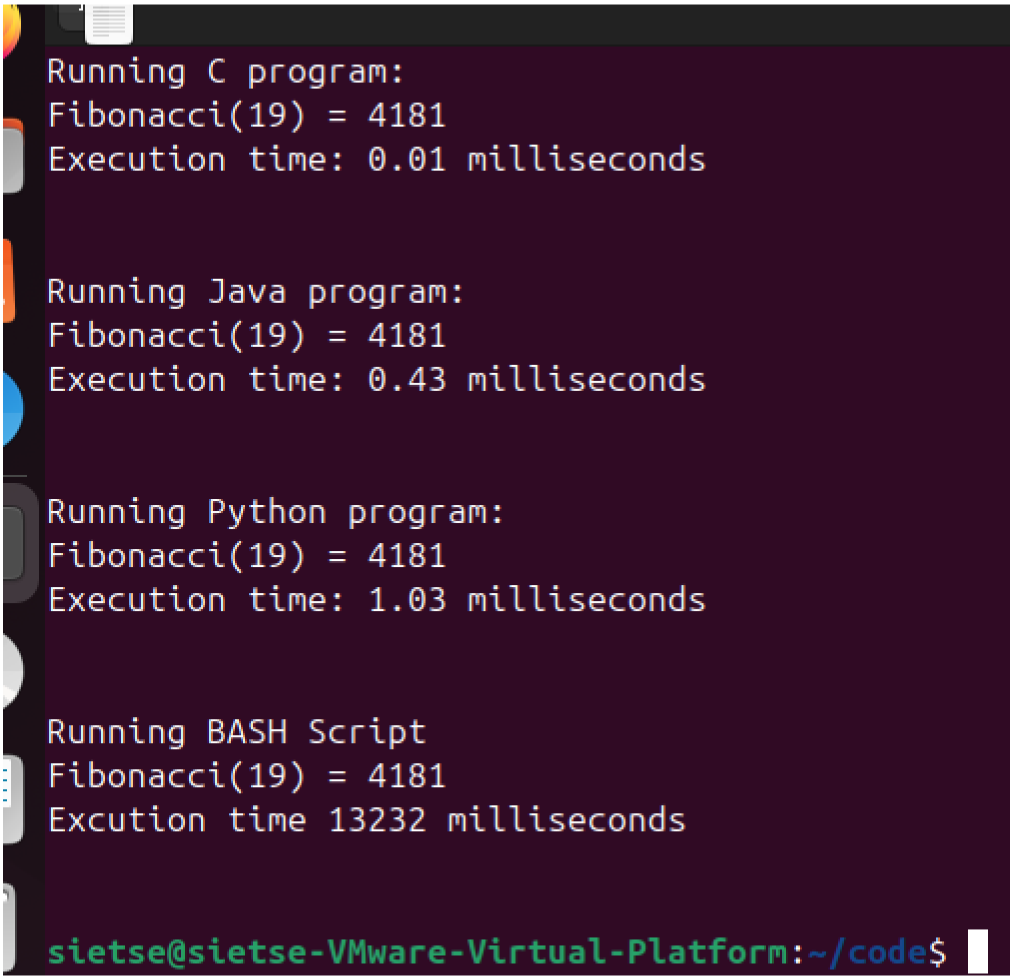
**Assignment 4.4: Optimize**

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.
You can use -O2 or -O3 to optimize a C file further


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?




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.

- No edit was needed, the runall.sh file runs all files sequentially already, the only this I needed to do was compile the runall.sh but I had done that previously before optimizing the fib.c further

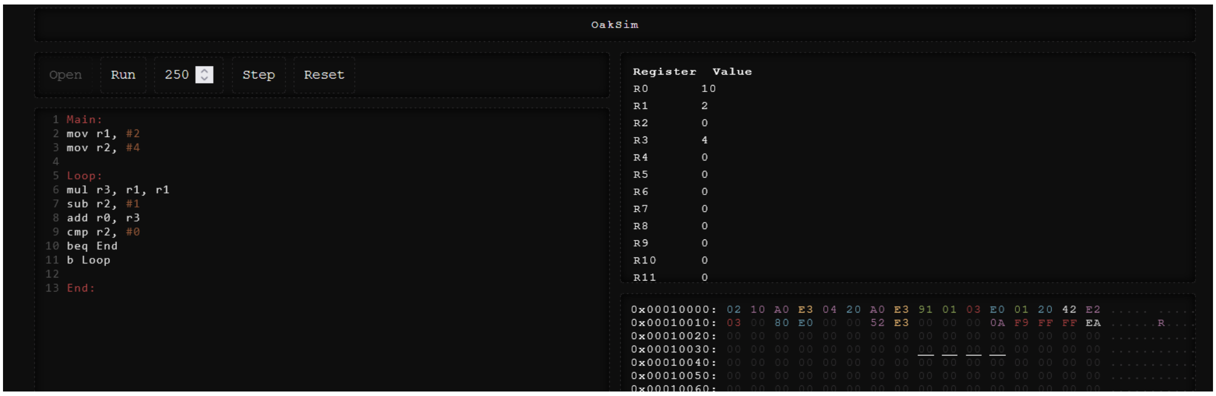
## Assignment 4.5: More ARM Assembly

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**