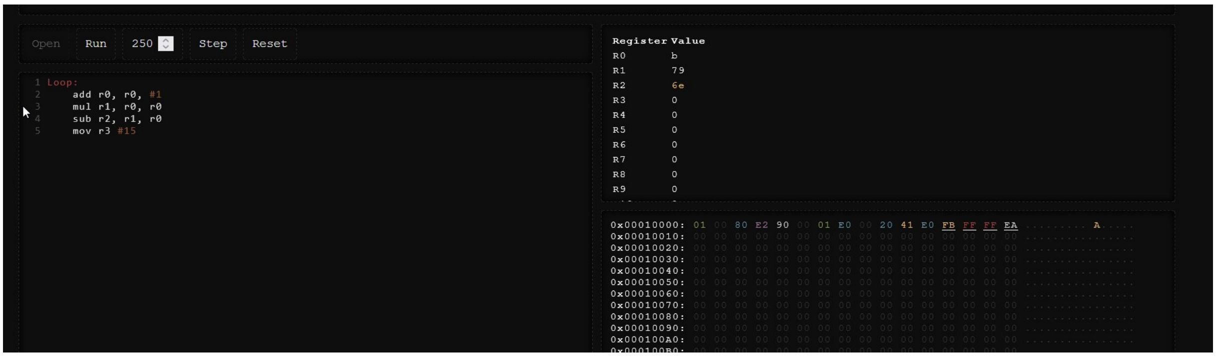
# Template Week 4 – Software

Student number: 588734

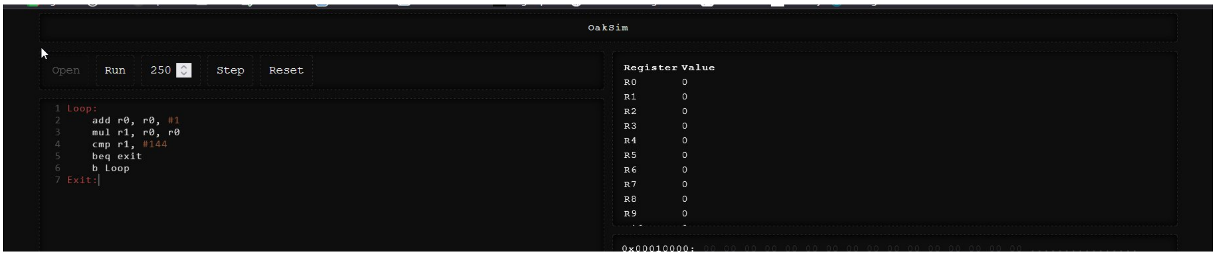
**Assignment 4.1: ARM assembly**

Screenshot of working assembly code of factorial calculation:




What is the value of:

- r0: b
- r1: 79
- r2: 6e
- r3: 0




**Assignment 4.2: Programming languages**

Take screenshots that the following commands work:

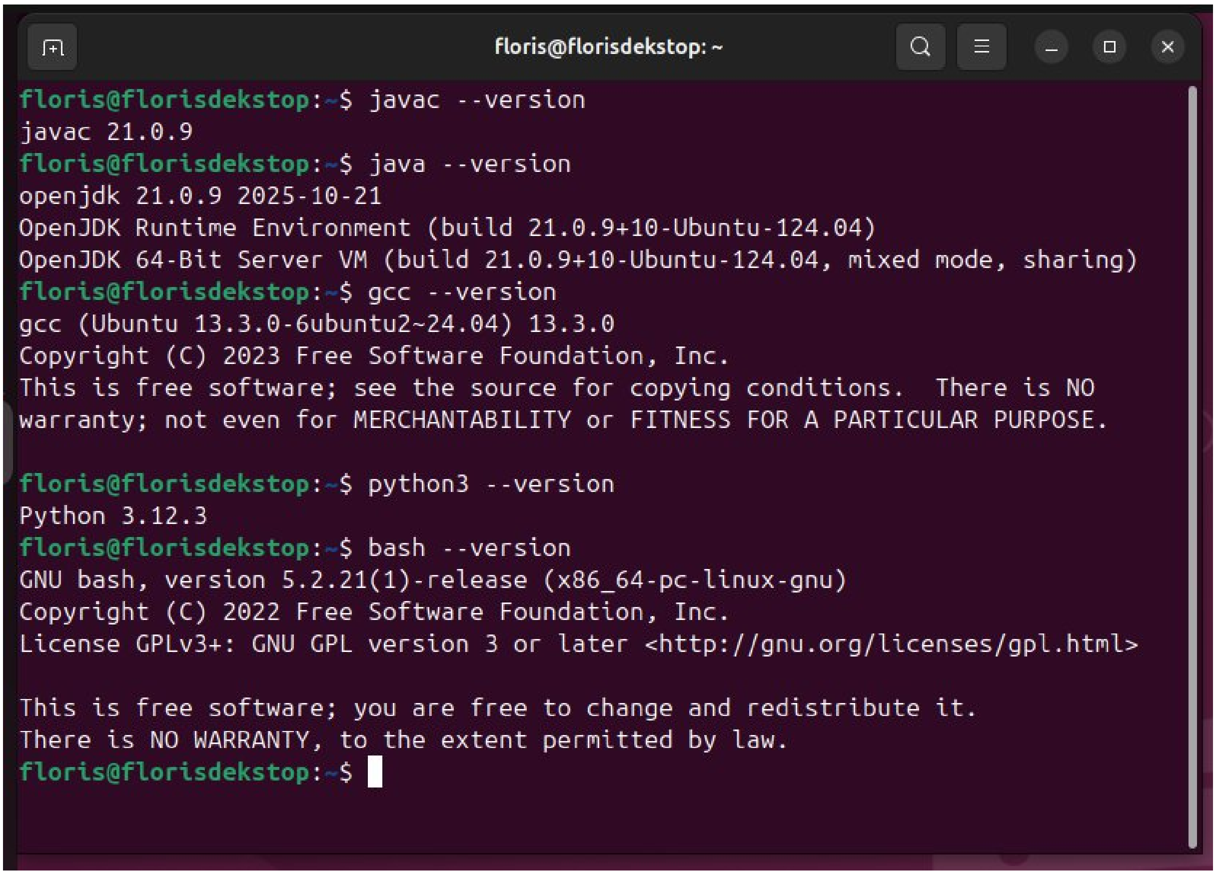
javac --version

java --version

gcc --version

python3 --version

bash --version




**JavaC: javac 21.0.9**
**Java: openjdk 21.0.9**
**GCC: 13.3.0**
**Python: 3.12.3**
**Bash: 5.2.21(1)**

**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 en Fibonacci.java

**Which source code files are compiled to byte code?**
fib.c

**Which source code files are interpreted by an interpreter?**
Fibonacci.java

**These source code files will perform the same calculation after compilation/interpretation. Which one is expected to do the calculation the fastest?**
fib.py, fib.sh, The fatest is fib.c

**How do I run a Java program?**
1. Compile: javac Fibonacci.java
2. Run: java Fibonacci

**How do I run a Python program?**
1. python3 fib.py

How do I run a C program?
1. Compile: gcc fib.c -o fib
2. Run: ./fib

How do I run a Bash script?
1. bash fib.sh

If I compile the above source code, will a new file be created? If so, which file?
C: Yes, If you run the gcc.fib it creates a default file named a.out. or a.exe on windows.
Java: Yes, The compiler makes an file with the extension .class
Python: No, You don’t have to compile python, it will run it.
Bash: No bash does not compiling.

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?

```
floris@florisdekstop: ~/Documents/code
floris@florisdekstop:~/Documents/code$ gcc fib.c -o fib
floris@florisdekstop:~/Documents/code$ ./fib
Fibonacci(18) = 2584
Execution time: 0.02 milliseconds
floris@florisdekstop:~/Documents/code$
```

```
floris@florisdekstop: ~/Documents/code
floris@florisdekstop:~/Documents/code$ javac Fibonacci.java
floris@florisdekstop:~/Documents/code$ java Fibonacci
Fibonacci(18) = 2584
Execution time: 0.37 milliseconds
floris@florisdekstop:~/Documents/code$
```

```
floris@florisdekstop: ~/Documents/code
floris@florisdekstop:~/Documents/code$ python3 fib.py
Fibonacci(18) = 2584
Execution time: 0.54 milliseconds
floris@florisdekstop:~/Documents/code$
```

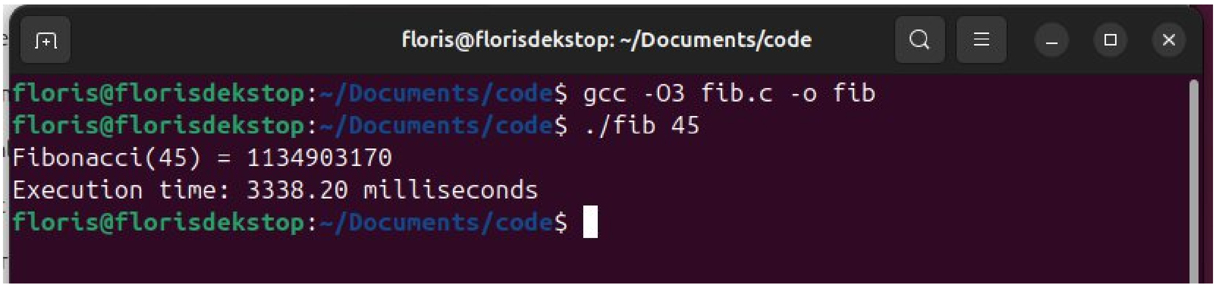
```
floris@florisdekstop: ~/Documents/code
floris@florisdekstop:~/Documents/code$ chmod +x fib.sh
floris@florisdekstop:~/Documents/code$ ./fib.sh

Fibonacci(18) = 2584
Exction time 12732 milliseconds
floris@florisdekstop:~/Documents/code$
floris@florisdekstop:~/Documents/code$
```

The first one (C) is 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.




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?

Yes

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.

```
========================================
1. COMPILING SOURCE CODES
========================================
[*] Compiling C (fib.c) with -O3...
[*] Compiling Java (Fibonacci.java)...

========================================
2. RUNNING PERFORMANCE TEST (N=40)
========================================
--- C Executable ---
Fibonacci(40) = 102334155
Execution time: 234.35 milliseconds

--- Java Bytecode ---
Fibonacci(40) = 102334155
Execution time: 668.18 milliseconds

--- Python Interpreter ---

Fibonacci(40) = 102334155
Execution time: 27276.00 milliseconds

--- Bash Script (Using N=15 for safety) ---
Fibonacci(15) = 610
Excution time 3017 milliseconds
```

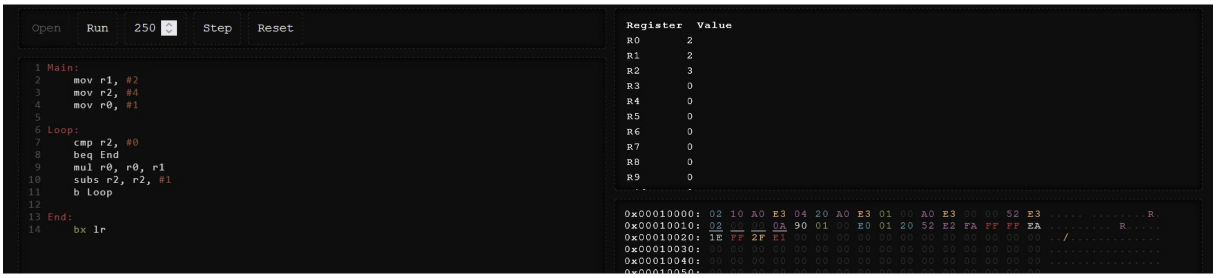
**Assignment 4.5: More ARM Assembly**

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**