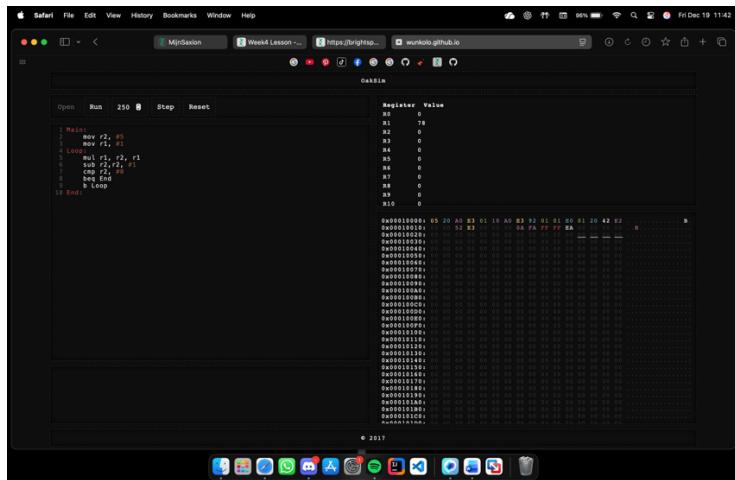


# Template Week 4 – Software

Student number: **592964**

## 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

```
dani@dani-ubuntu:~$ javac --version
javac 21.0.9
```

java –version

```
dani@dani-ubuntu:~$ 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)
```

gcc –version

```
dani@dani-ubuntu:~$ 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.
```

python3 –version

```
dani@dani-ubuntu:~$ python3 --version
Python 3.12.3
```

bash –version

```
dani@dani-ubuntu:~$ bash --version
GNU bash, version 5.2.21(1)-release (aarch64-unknown-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 and .c**

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

**.c**

Which source code files are compiled to byte code?

**.java**

Which source code files are interpreted by an interpreter?

**.py and .sh**

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

**.c**

How do I run a Java program?

**Commands - javac Fibonacci.java and then - java Fibonacci**

How do I run a Python program?

**python3 fib.py**

How do I run a C program?

**gcc fib.c -o fib and then - ./fib**

How do I run a Bash script?

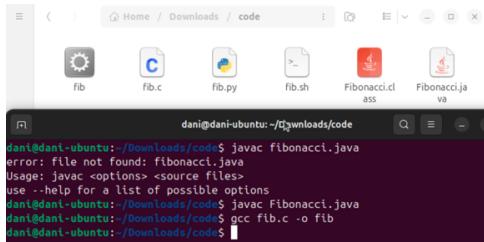
**bash fib.sh**

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

**.java and .c will create new file (.class and .out)**

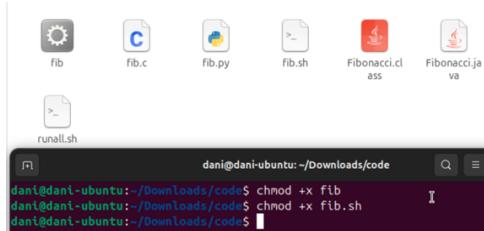
Take relevant screenshots of the following commands:

- Compile the source files where necessary



```
dani@dani-ubuntu:~/Downloads/code$ javac fibonacci.java
error: file not found: fibonacci.java
Usage: javac <options> <source files>
use --help for a list of possible options
dani@dani-ubuntu:~/Downloads/code$ javac Fibonacci.java
dani@dani-ubuntu:~/Downloads/code$ gcc fib.c -o fib
dani@dani-ubuntu:~/Downloads/code$
```

- Make them executable



```
dani@dani-ubuntu:~/Downloads/code$ chmod +x fib
dani@dani-ubuntu:~/Downloads/code$ chmod +x fib.sh
dani@dani-ubuntu:~/Downloads/code$
```

- Run them



```
dani@dani-ubuntu:~/Downloads/code$ ./fib
Fibonacci(18) = 2584
Execution time: 0.02 milliseconds
dani@dani-ubuntu:~/Downloads/code$ java Fibonacci
Fibonacci(18) = 2584
Execution time: 0.17 milliseconds
dani@dani-ubuntu:~/Downloads/code$ python3 fib.py
Fibonacci(18) = 2584
Execution time: 0.29 milliseconds
dani@dani-ubuntu:~/Downloads/code$ ./fib.sh
.fib.sh: command not found
dani@dani-ubuntu:~/Downloads/code$ ./fib.sh
Fibonacci(18) = 2584
Execution time 1463 milliseconds
dani@dani-ubuntu:~/Downloads/code$
```

- Which (compiled) source code file performs the calculation the fastest?

1. **C - 0.02ms**
2. Java – 0.17ms
3. Python – 0.29ms
4. Bash – 1463ms



```
dani@dani-ubuntu:~/Downloads/code$ ./fib
Fibonacci(18) = 2584
Execution time: 0.02 milliseconds
dani@dani-ubuntu:~/Downloads/code$ java Fibonacci
Fibonacci(18) = 2584
Execution time: 0.17 milliseconds
dani@dani-ubuntu:~/Downloads/code$ python3 fib.py
Fibonacci(18) = 2584
Execution time: 0.29 milliseconds
dani@dani-ubuntu:~/Downloads/code$ ./fib.sh
.fib.sh: command not found
dani@dani-ubuntu:~/Downloads/code$ ./fib.sh
Fibonacci(18) = 2584
Execution time 1463 milliseconds
dani@dani-ubuntu:~/Downloads/code$
```

#### 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.

**gcc -O2 fib.c -o fib**

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

**gcc -O2 fib.c -o fib\_opt**

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

**For me it still runs at the same speed – 0.02ms**

- 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.

```
Running C program:  
Fibonacci(19) = 4181  
Execution time: 0.03 milliseconds  
  
Running Java program:  
Fibonacci(19) = 4181  
Execution time: 0.12 milliseconds  
  
Running Python program:  
Fibonacci(19) = 4181  
Execution time: 0.26 milliseconds  
  
Running BASH Script  
Fibonacci(19) = 4181  
Excution time 2318 milliseconds  
  
dani@dani-ubuntu:~/Downloads/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:

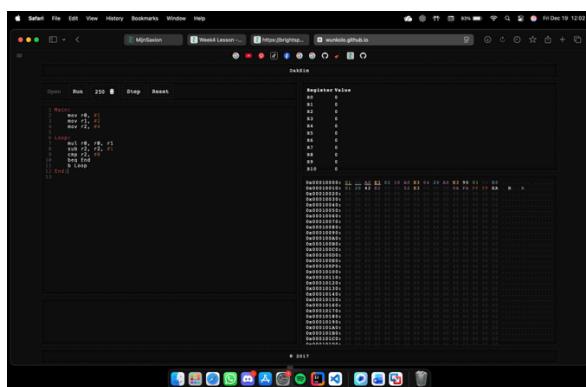
```
mov r1, #2  
mov r2, #4
```

Loop:

End:

Complete the code. See the PowerPoint slides of week 4.

Screenshot of the completed code here.



The screenshot shows a debugger interface with the following details:

- Registers:** Shows the state of various ARM registers (R0-R3, R12, PC, SP, LR, CPSR) across multiple frames. The R0 register shows values 0, 2, 4, 8, 16, 32, 64, 128, and finally 256.
- Memory:** A dump of memory starting at address 0x00000000, showing the assembly code and its execution results.
- Code View:** Displays the assembly code for the program, including the main loop logic.

Ready? Save this file and export it as a pdf file with the name: [week4.pdf](#)