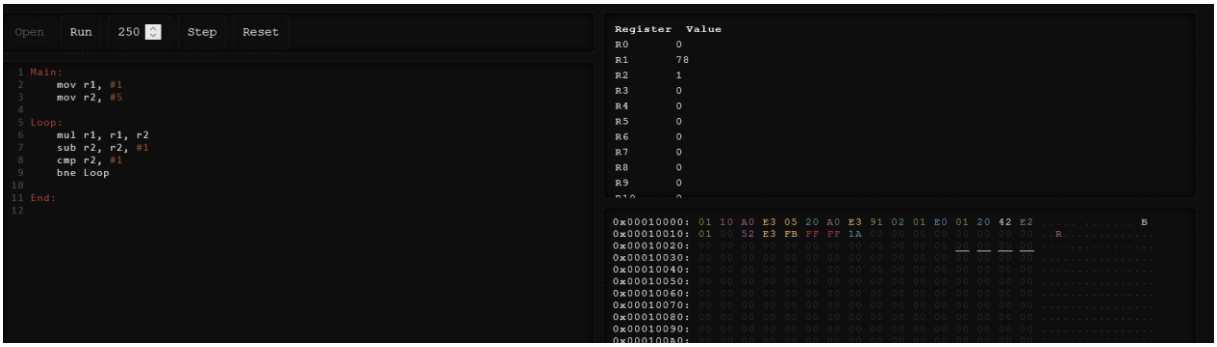
Template Week 4 – Software

Student number: 588991

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

```
yvar@yvar-VMware-Virtual-Platform:~$ javac --version
javac 21.0.8
```

java –version

```
yvar@yvar-VMware-Virtual-Platform:~$ java --version
openjdk 21.0.8 2025-07-15
OpenJDK Runtime Environment (build 21.0.8+9-Ubuntu-0ubuntu124.04.1)
OpenJDK 64-Bit Server VM (build 21.0.8+9-Ubuntu-0ubuntu124.04.1, mixed mode, sha
ring)
```

gcc –version

```
yvar@yvar-VMware-Virtual-Platform:~$ 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

```
yvar@yvar-VMware-Virtual-Platform:~$ python3 --version
Python 3.12.3
```

bash --version

```
yvar@yvar-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?
De Fibonacci.java en de fib.c

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

Which source code files are compiled to byte code?
Fibonacci.java

Which source code files are interpreted by an interpreter?
fib.py en fib.sh

These source code files will perform the same calculation after compilation/interpretation. Which one is expected to do the calculation the fastest?
De fib.c want deze wordt direct op de CPU uitgevoerd.

How do I run a Java program?
javac Fibonacci.java
java Fibonacci

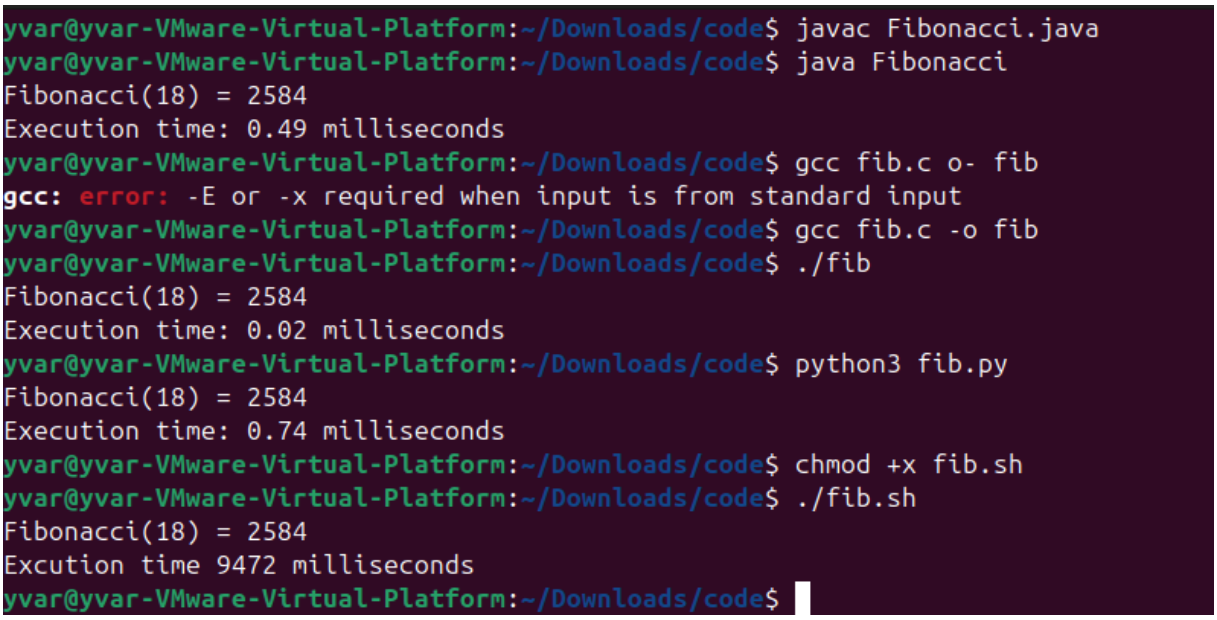
How do I run a Python program?
python3 fib.py

How do I run a C program?
gcc fib.c -o fib
./fib

How do I run a Bash script?
chmod +x fib.sh
./fib.sh

If I compile the above source code, will a new file be created? If so, which file?
Bij de .java en de .c wel en daar komen de volgende bestanden uit.
.Java : Fibonacci.class
.c : fib.exe

Take relevant screenshots of the following commands:



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

```
yvar@yvar-VMware-Virtual-Platform:~/Downloads/code$ gcc fib.c -o fib -o3
```

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

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

```
Running C program:
Fibonacci(19) = 4181
Execution time: 0.02 milliseconds


Running Java program:
Fibonacci(19) = 4181
Execution time: 0.30 milliseconds


Running Python program:
Fibonacci(19) = 4181
Execution time: 0.50 milliseconds


Running BASH Script
Fibonacci(19) = 4181
Excution time 11032 milliseconds


yvar@yvar-VMware-Virtual-Platform:~/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.

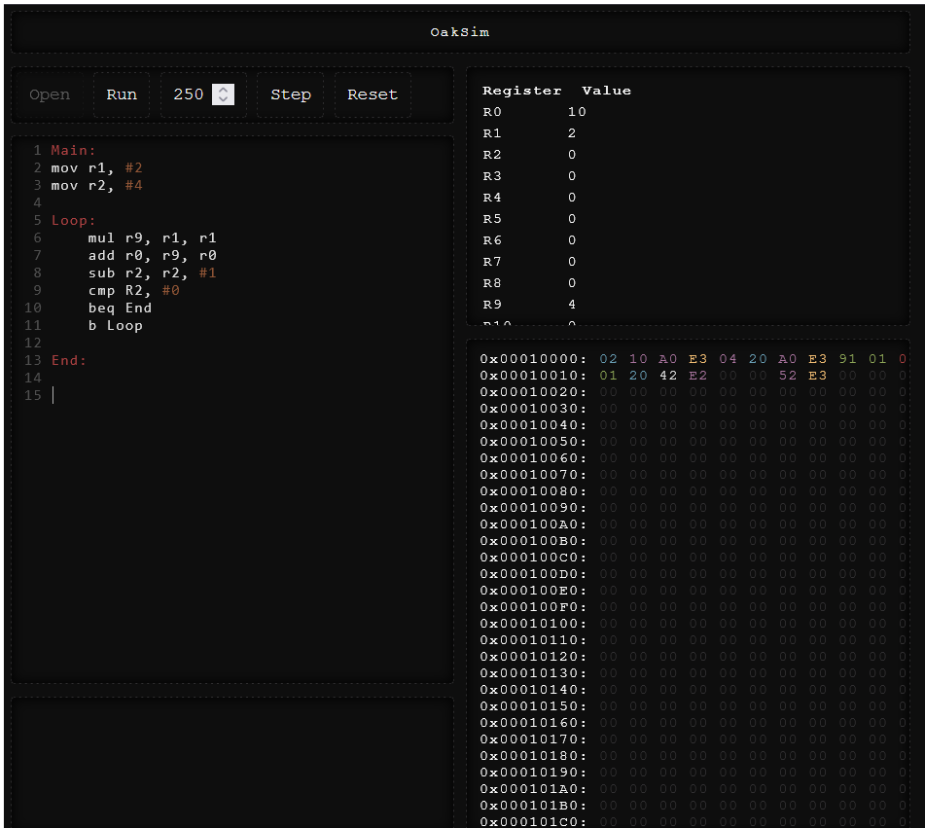
```
Main:
mov r1, #2
mov r2, #4

Loop:
      mul r9, r1, r1
      add r0, r9, r0
      sub r2, r2, #1
      cmp R2, #0
      beq End
      b 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**