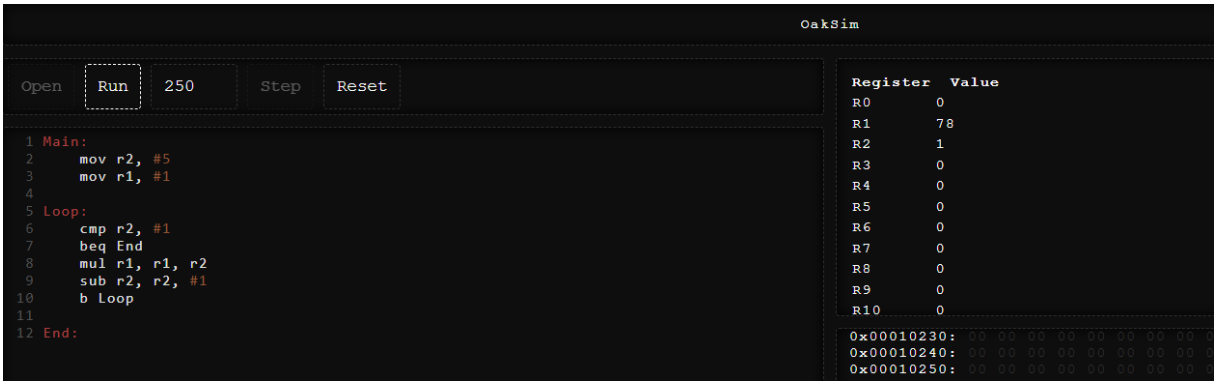
Template Week 4 – Software

Student number: 589020

Assignment 4.1: ARM assembly

Screenshot of working assembly code of factorial calculation:



Result: 78 in hex = 120 in decimal.

Assignment 4.2: Programming languages

Take screenshots that the following commands work:

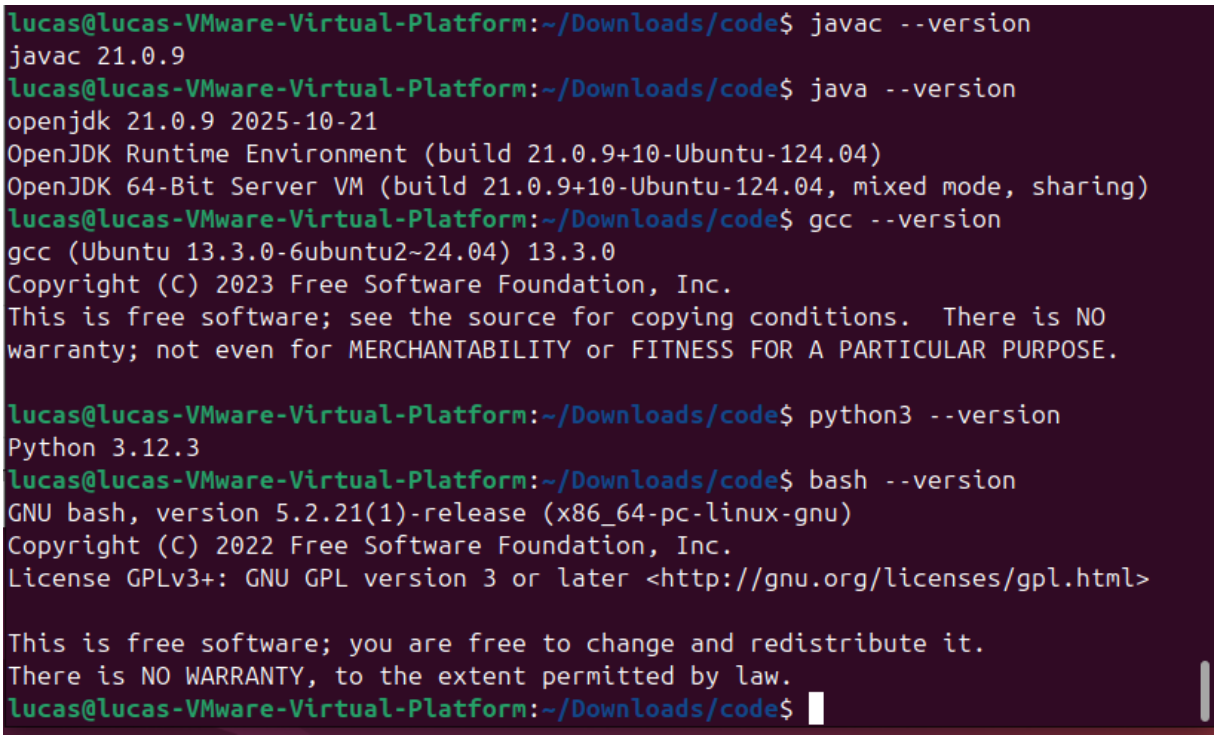
javac --version

java --version

gcc --version

python3 --version

bash –version


lucas@lucas-VMware-Virtual-Platform:~/Downloads/code$ javac --version
javac 21.0.9
lucas@lucas-VMware-Virtual-Platform:~/Downloads/code$ 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)
lucas@lucas-VMware-Virtual-Platform:~/Downloads/code$ 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.
lucas@lucas-VMware-Virtual-Platform:~/Downloads/code$ python3 --version
Python 3.12.3
lucas@lucas-VMware-Virtual-Platform:~/Downloads/code$ 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.
lucas@lucas-VMware-Virtual-Platform:~/Downloads/code$

Assignment 4.3: Compile

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

The C file and the java file.

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

The C file which is compiled by gcc into a executable.

Which source code files are compiled to byte code?

The java file which is compiled into a .class file.

Which source code files are interpreted by an interpreter?

Fib.py. Interpreted by the Python interpreter.

Fib.sh. Interpreted by the Bash shell.

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

Most likely the C file because it compiles directly to machine code and runs natively without any virtual machine or interpreter.

How do I run a Java program?

In terminal type *javac filename.java* this creates a file named *filename.class*

Then in terminal type *java filename*

How do I run a Python program?

In terminal type *python3 filename.py*

How do I run a C program?

In terminal type *gcc -o filename filename.c*

Then type in terminal *./filename* to run the program.

How do I run a Bash script?

In terminal type *bash filename.sh*

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

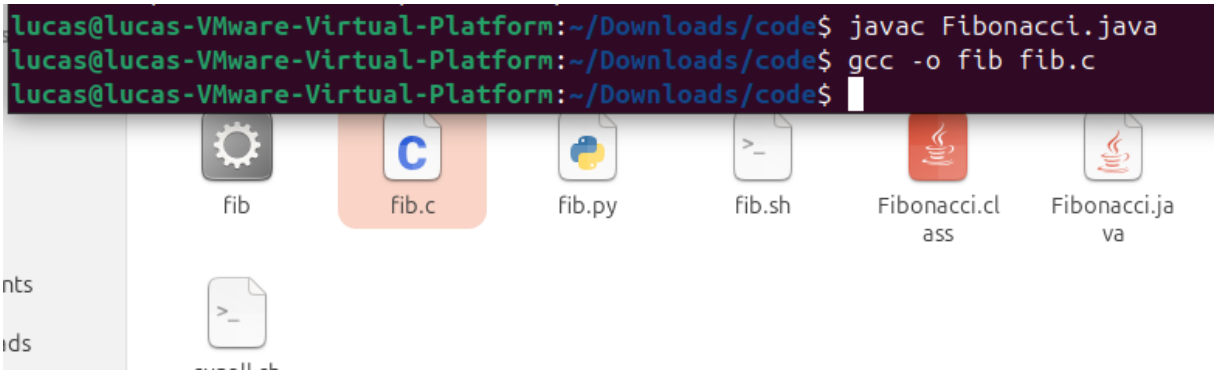
Fibonacci.java -> Fibonacci.class

Fib.c -> fib

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?**

Compiling java and c file into an executable:



Running them:

```
lucas@lucas-VMware-Virtual-Platform:~/Downloads/code$ java Fibonacci
Fibonacci(18) = 2584
Execution time: 0.22 milliseconds
lucas@lucas-VMware-Virtual-Platform:~/Downloads/code$ python3 fib.py
Fibonacci(18) = 2584
Execution time: 0.34 milliseconds
lucas@lucas-VMware-Virtual-Platform:~/Downloads/code$ ./fib
Fibonacci(18) = 2584
Execution time: 0.02 milliseconds
lucas@lucas-VMware-Virtual-Platform:~/Downloads/code$ bash fib.sh
Fibonacci(18) = 2584
Excution time 5674 milliseconds
lucas@lucas-VMware-Virtual-Platform:~/Downloads/code$ S
```

As you can see the C file is by far the fastest with 0.02 miliseconds. I don't know why the bash file takes more than 5 seconds.

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 pass -O1, -O2, -O3 and -Ofast to optimize for execution speed. I have tried -O3 with this command: gcc -O3 fib.c -o fib. And it had an execution time of 0.01 milliseconds!

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?**

```
lucas@lucas-VMware-Virtual-Platform:~/Downloads/code$ gcc -O3 fib.c -o fib
lucas@lucas-VMware-Virtual-Platform:~/Downloads/code$ ./fib
Fibonacci(18) = 2584
Execution time: 0.01 milliseconds
```

Yes it now is 0.01 milliseconds instead of 0.02 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: 0.01 milliseconds


Running Java program:
Fibonacci(19) = 4181
Execution time: 0.36 milliseconds


Running Python program:
Fibonacci(19) = 4181
Execution time: 0.63 milliseconds


Running BASH Script
Fibonacci(19) = 4181
Excution time 15612 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**