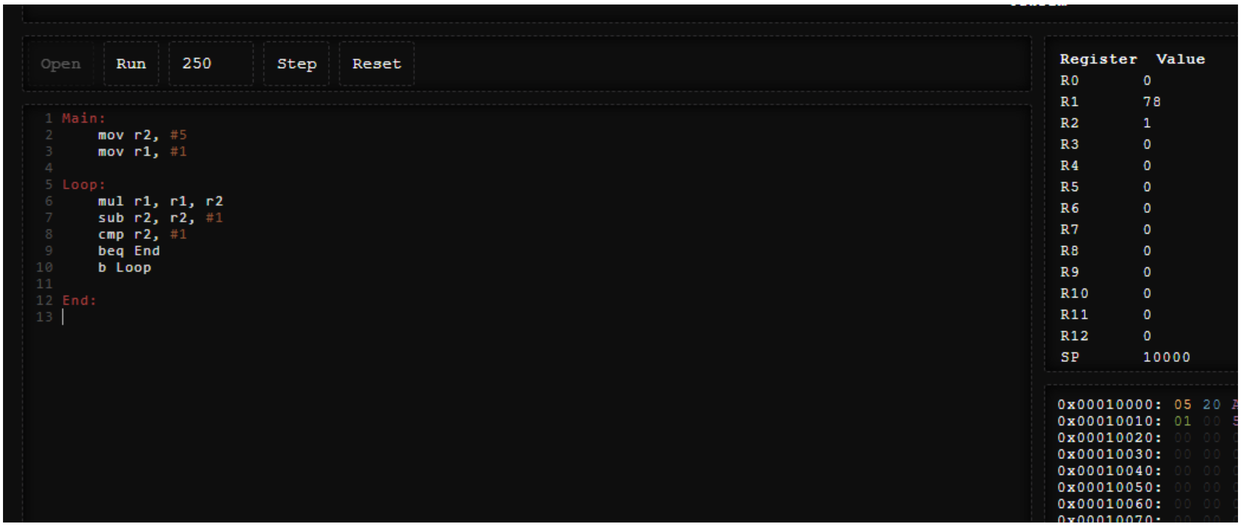
Template Week 4 – Software

Student number: 547518

Assignment 4.1: ARM assembly

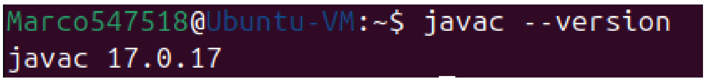
Screenshot of working assembly code of factorial calculation:



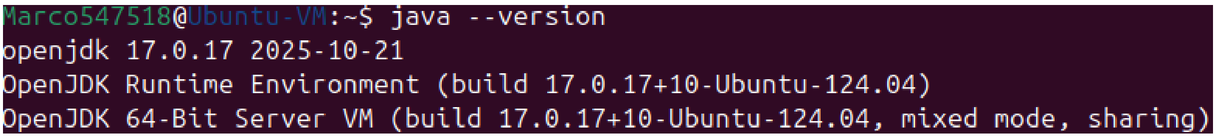
Assignment 4.2: Programming languages

Take screenshots that the following commands work:

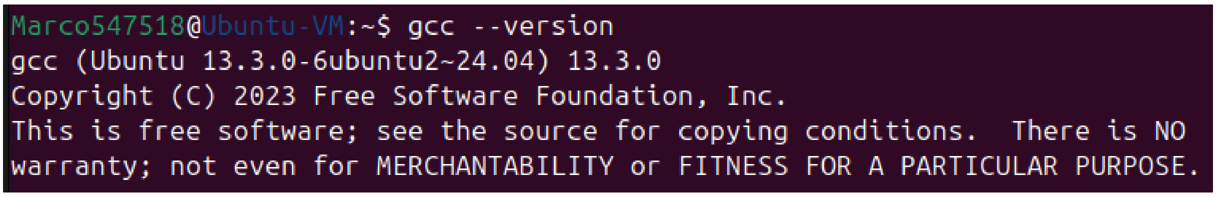
javac –version



java –version



gcc –version



python3 –version

bash –version

```
Marco547518@Ubuntu-VM:~$ 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? **Fibonacci.java and fib.c**

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

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

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

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

How do I run a Java program? **Run javac Fibonacci.java to compile it. Thus creating Fibonacci.class and then run java Fibonacci to run the program**

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

How do I run a C program? **First compile it with gcc like this: gcc fib.c -o fib Then it create fib then make it an executable with: chmod +x fib then run ./fib**

How do I run a Bash script? **Make the script a executable with sudo chmod a+x fib.sh then run sudo ./fib.sh**

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

Yes fib.c will create fib and Fibonnaci.java will create Fibonnaci.class

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?

```
Marco547518@Ubuntu-VM:~/Downloads/code$ javac Fibonacci.java
Marco547518@Ubuntu-VM:~/Downloads/code$ java Fibonacci
Fibonacci(18) = 2584
Execution time: 0.33 milliseconds
```

Example of java compilation for Fibonacci.java

```
Marco547518@Ubuntu-VM:~/Downloads/code$ gcc fib.c -o fib
Marco547518@Ubuntu-VM:~/Downloads/code$ chmod +x fib
Marco547518@Ubuntu-VM:~/Downloads/code$ ./fib
Fibonacci(18) = 2584
Execution time: 0.02 milliseconds
```

Example of C compilation for fib.c . C has the fastest execution time out of all the programs

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. There are flags like -O0 which is on my default that uses no optimizations and it goes until -O3 but with research it appears that -O2 is the best for performance as -O3 increases the file size

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

Ran the command gcc -O2 fib.c -o fib_opt

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

```
Marco547518@Ubuntu-VM:~/Downloads/code$ gcc -O2 fib.c -o fib_opt
Marco547518@Ubuntu-VM:~/Downloads/code$ ./fib_opt
Fibonacci(18) = 2584
Execution time: 0.00 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: 0.03 milliseconds


Running Java program:
Fibonacci(19) = 4181
Execution time: 0.33 milliseconds


Running Python program:
Fibonacci(19) = 4181
Execution time: 1.52 milliseconds


Running BASH Script
Fibonacci(19) = 4181
Excution time 10328 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**