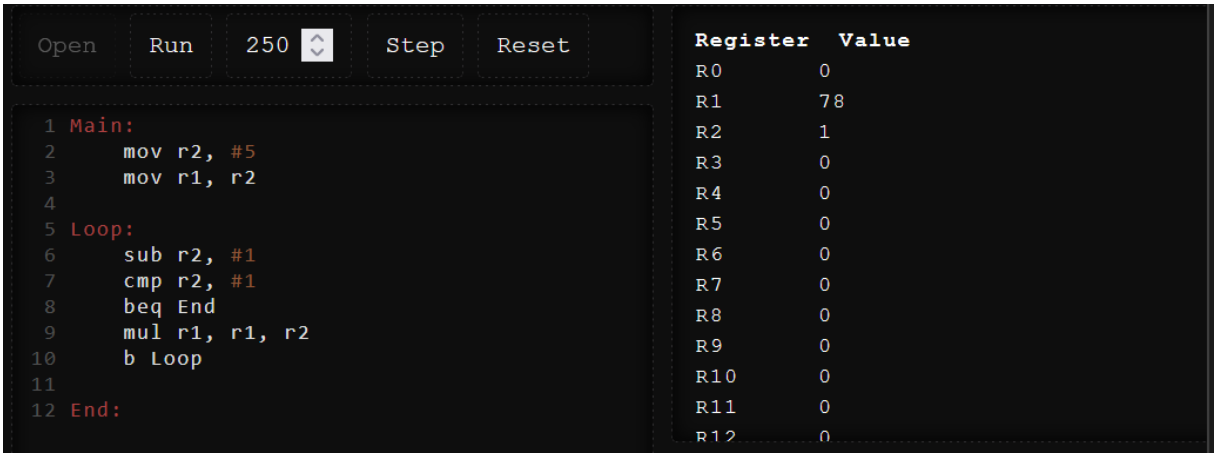
# Template Week 4 – Software

Student number: 522880

**Assignment 4.1: ARM assembly**

Screenshot of working assembly code of factorial calculation:




78 hexadecimal equals 120 decimal

**Assignment 4.2: Programming languages**

Take screenshots that the following commands work:

javac --version

java --version

gcc --version

python3 --version

bash --version

```
niels@NielsPC:~/Downloads/code$ javac -version
javac 21.0.5
niels@NielsPC:~/Downloads/code$ java -version
openjdk version "21.0.5" 2024-10-15
OpenJDK Runtime Environment (build 21.0.5+11-Ubuntu-1ubuntu124.04)
OpenJDK 64-Bit Server VM (build 21.0.5+11-Ubuntu-1ubuntu124.04, mixed mode, sha
ing)
niels@NielsPC:~/Downloads/code$ gcc --version
gcc (Ubuntu 13.2.0-23ubuntu4) 13.2.0
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.

niels@NielsPC:~/Downloads/code$ python3 --version
Python 3.12.3
niels@NielsPC:~/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.
```

**Assignment 4.3: Compile**

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

Java & C -> Fibonacci.java and fib.c

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

C -> fib.c

Which source code files are compiled to byte code?

Java -> Fibonacci.java

Which source code files are interpreted by an interpreter?

Python -> fib.py

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

C -> fib.c

How do I run a Java program?

javac [filename] -> compilation

java [compiled_filename] -> running the compiled file

How do I run a Python program?

python3 [filename]

How do I run a C program?

gcc [filename] ->compilation

./[compiled_filename] ->running the compiled code

How do I run a Bash script?

./[filename.sh]

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

Yes, Fibonacci.class for java, a fib.exe for C

Take relevant screenshots of the following commands:

- Compile the source files where necessary

```
niels@NielsPC:~/Downloads/code$ gcc fib.c -o fib
niels@NielsPC:~/Downloads/code$ javac Fibonacci.java
```

- Make them executable

```
niels@NielsPC:~/Downloads/code$ chmod +x fib.sh
```

- Run them

```
niels@NielsPC:~/Downloads/code$ ./fib
Fibonacci(18) = 2584
Execution time: 0.03 milliseconds
niels@NielsPC:~/Downloads/code$ java Fibonacci
Fibonacci(18) = 2584
Execution time: 0.39 milliseconds
niels@NielsPC:~/Downloads/code$ python3 fib.py
Fibonacci(18) = 2584
Execution time: 0.40 milliseconds
niels@NielsPC:~/Downloads/code$ ./
fib         fib.sh      runall.sh
niels@NielsPC:~/Downloads/code$ ./fib.sh
Fibonacci(18) = 2584
Excution time 4741 milliseconds
```

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

```
Running C program:
Fibonacci(19) = 4181
Execution time: 0.04 milliseconds


Running Java program:
Fibonacci(19) = 4181
Execution time: 0.73 milliseconds


Running Python program:
Fibonacci(19) = 4181
Execution time: 0.47 milliseconds


Running BASH Script
Fibonacci(19) = 4181
Excution time 7566 milliseconds
```

**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.
The flags of -O1 to -O3 should optimize more with higher numbers, the flag -Ofast should disregard standard compliance for maximum speed

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

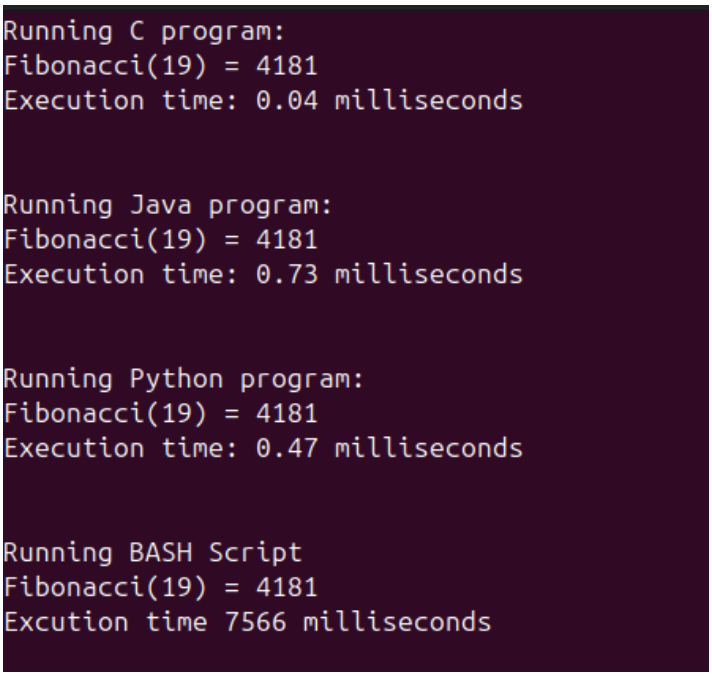
```
niels@NielsPC:~/Downloads/code$ gcc -O3 fib.c -o fibo3
niels@NielsPC:~/Downloads/code$ gcc -O2 fib.c -o fibo2
niels@NielsPC:~/Downloads/code$ gcc -O1 fib.c -o fibo1
```

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

```
niels@NielsPC:~/Downloads/code$ ./fibo1
Fibonacci(18) = 2584
Execution time: 0.02 milliseconds
niels@NielsPC:~/Downloads/code$ ./fibo2
Fibonacci(18) = 2584
Execution time: 0.01 milliseconds
niels@NielsPC:~/Downloads/code$ ./fibo3
Fibonacci(18) = 2584
Execution time: 0.01 milliseconds
```

The calculation seems to be the same speed but the execution time seems to be faster

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.



**Bonus point assignment – week 4**

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.

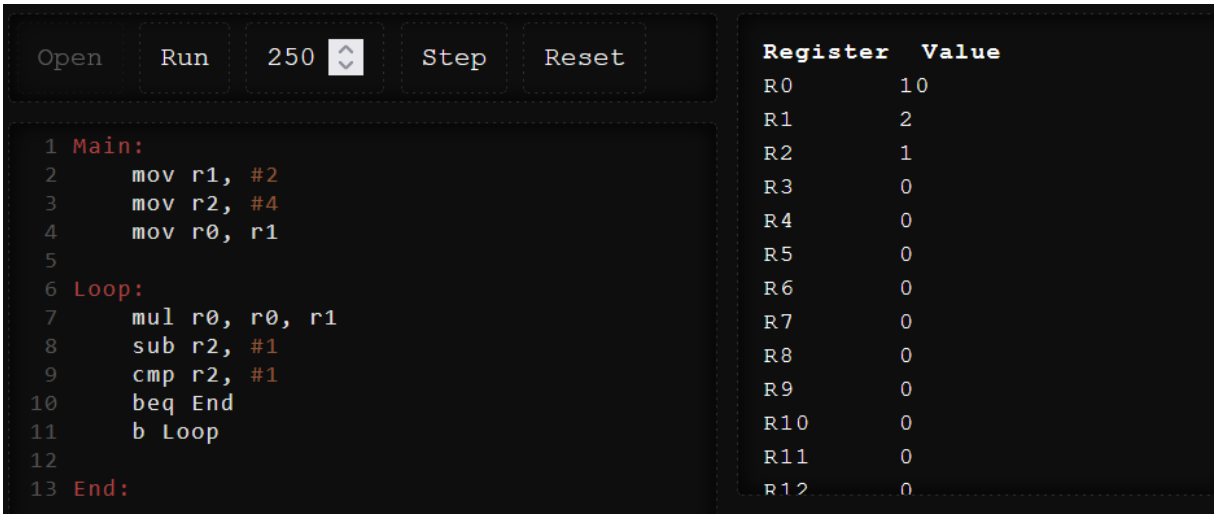
```
Main:
mov r1, #2
mov r2, #4

Loop:

End:
```

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

Screenshot of the completed code here.




10 hexadecimal equals 16 decimal

Ready? Save this file and export it as a pdf file with the name: **week4.pdf**