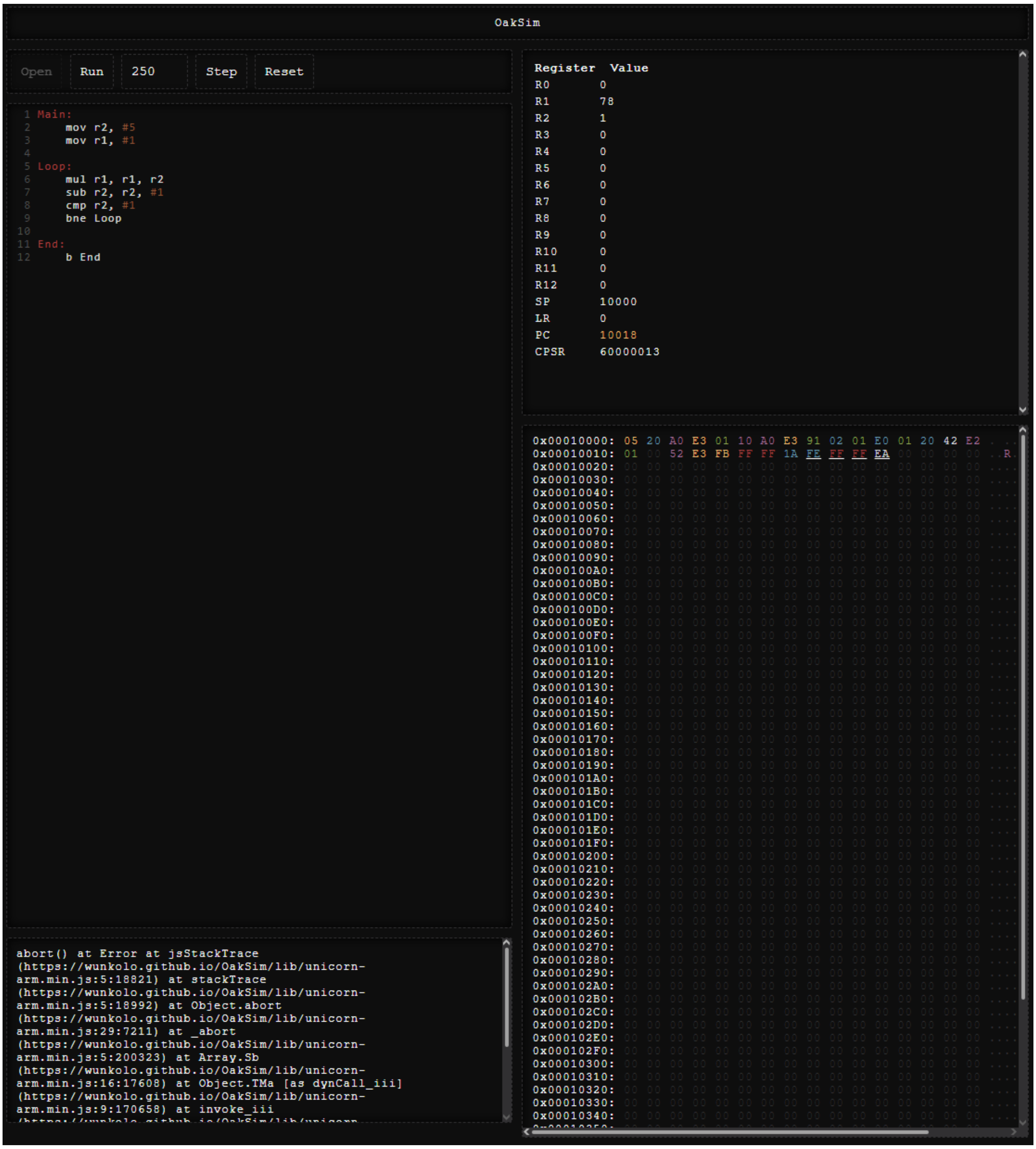
Template Week 4 – Software

Student number: 569091

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:

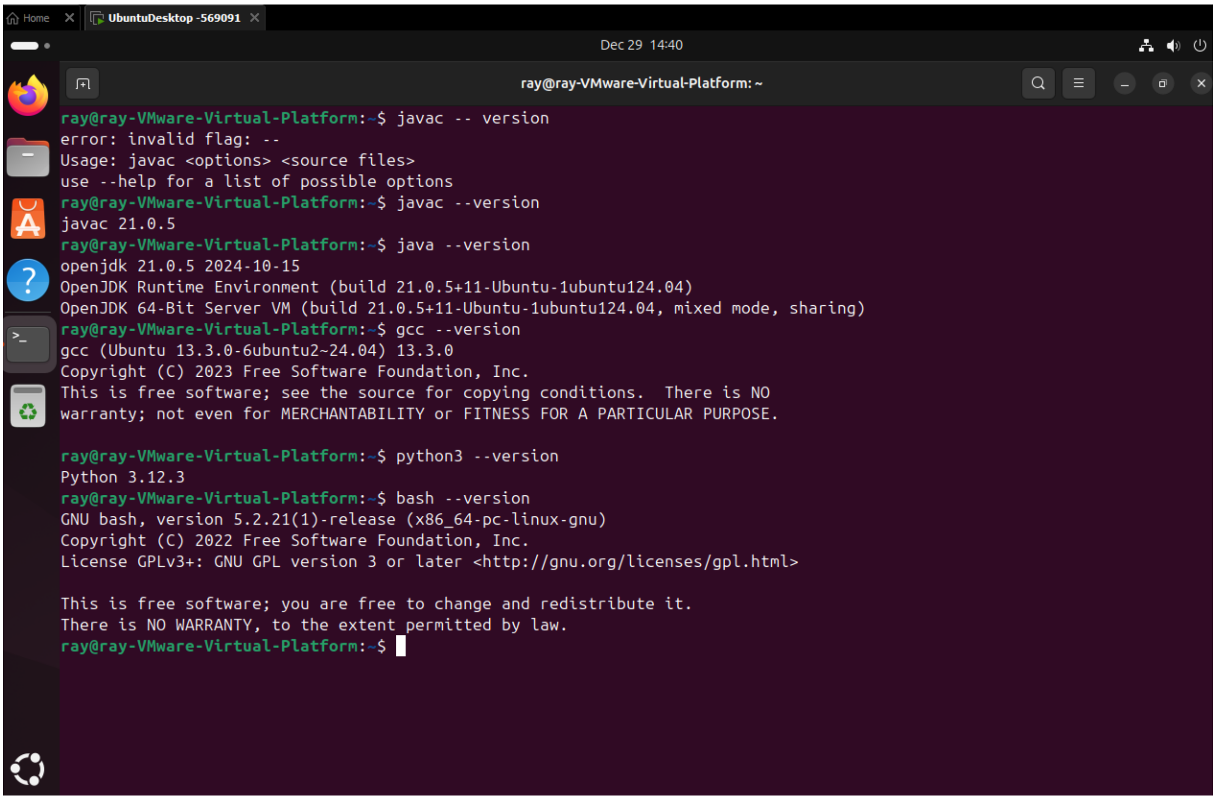
javac --version

java --version

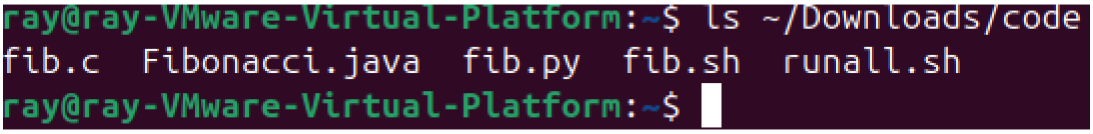
gcc --version

python3 --version

bash –version



Assignment 4.3: Compile



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

Files that require compiling:

- Fibonacci.java (Java source code).
- fib.c (C source code).

Files that do not need compilation:

- fib.py (Python source code is interpreted by an interpreter).
- fib.sh (Bash script executed by the shell interpreter).

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

- fib.c (compiled to a binary file, executable by the processor).

Which source code files are compiled to byte code?

- Fibonacci.java (compiled to Java bytecode, typically in a .class file, executed by the Java Virtual Machine).

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?

- fib.c (C source code) is expected to perform the fastest because it is compiled into machine code and executed directly by the processor without additional layers like an interpreter or virtual machine.

How do I run a Java program?

Compile the Java file: **javac Fibonacci.java**

This creates a Fibonacci.class file.

Run the compiled Java program: **Java Fibonacci**

How do I run a Python program?

Run the Python program using the Python interpreter: **python3 fib.py**

How do I run a C program?

Compile the C file using a compiler like gcc: **gcc fib.c -o fib**

Run the compiled program: **./fib**

How do I run a Bash script?

Make the script executable: **chmod a+x fib.sh**

Run the script: **./fib.sh**

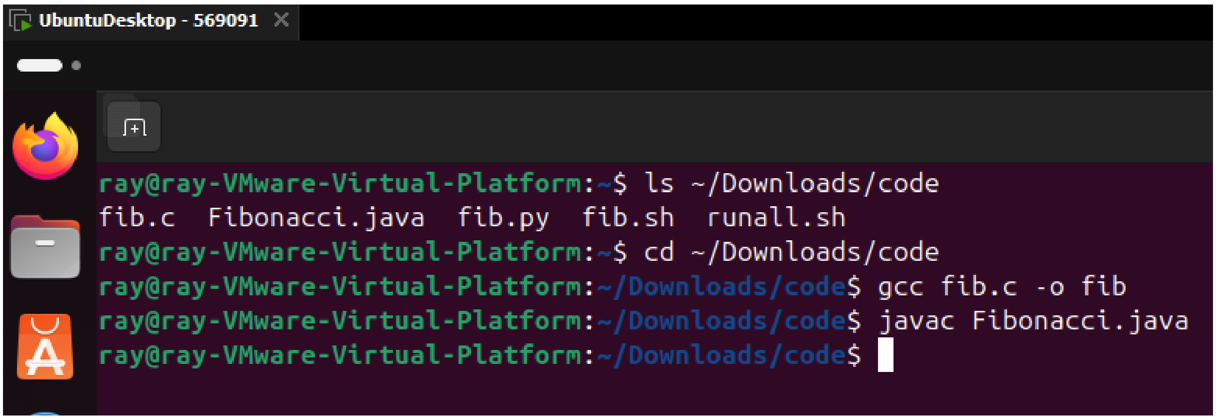
If I compile the above source code, will a new file be created? If so, which file?

Yes, compiling the source code creates new files:

- **Fibonacci.java**: Creates Fibonacci.class.
- **fib.c**: Creates fib (or another name you specify with -o during compilation).
- **fib.py and fib.sh**: Do not create compiled files as they are interpreted.

Take relevant screenshots of the following commands:

- Compile the source files where necessary



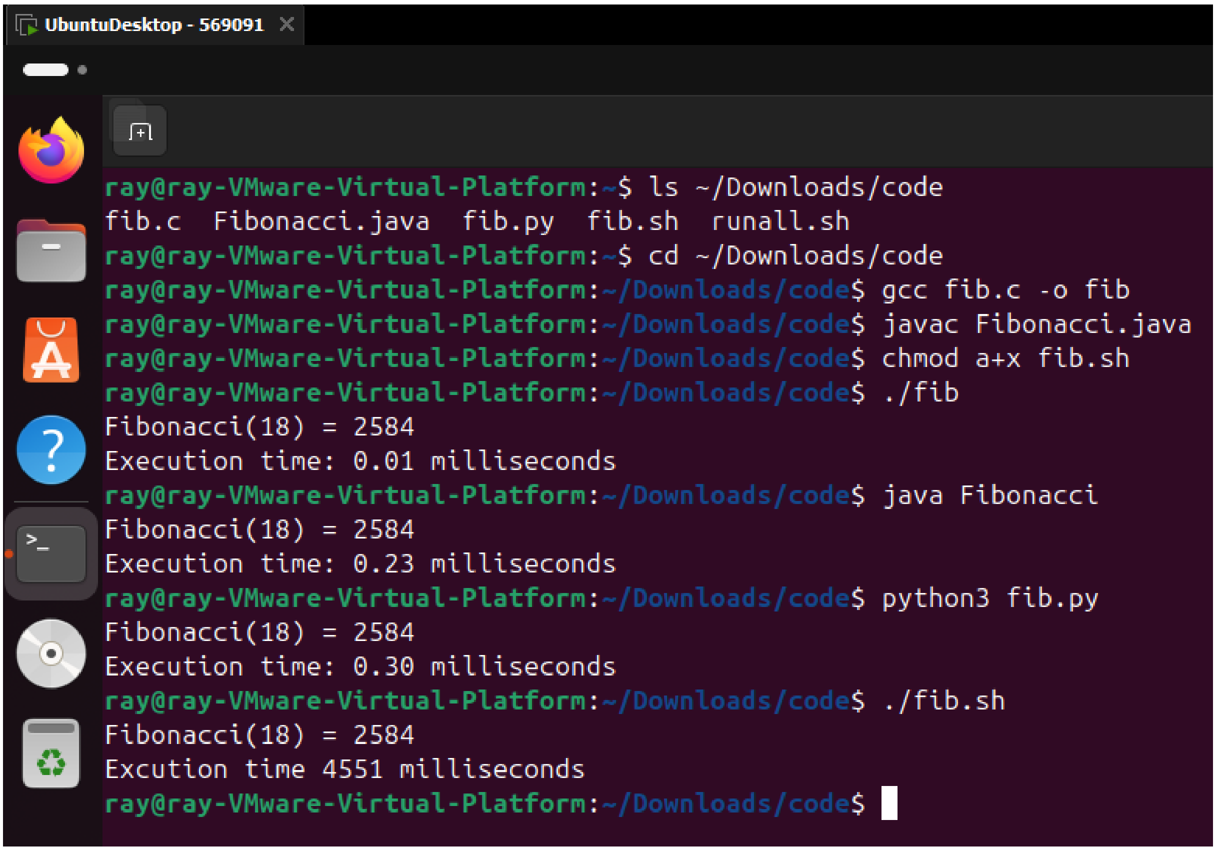
- Make them executable

```
ray@ray-VMware-Virtual-Platform:~/Downloads/code$ chmod a+x fib.sh
```

- Run them

```
ray@ray-VMware-Virtual-Platform:~/Downloads/code$ ./fib
Fibonacci(18) = 2584
Execution time: 0.01 milliseconds
ray@ray-VMware-Virtual-Platform:~/Downloads/code$ java Fibonacci
Fibonacci(18) = 2584
Execution time: 0.23 milliseconds
ray@ray-VMware-Virtual-Platform:~/Downloads/code$ python3 fib.py
Fibonacci(18) = 2584
Execution time: 0.30 milliseconds
ray@ray-VMware-Virtual-Platform:~/Downloads/code$ ./fib.sh
Fibonacci(18) = 2584
Excution time 4551 milliseconds
ray@ray-VMware-Virtual-Platform:~/Downloads/code$
```

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



Fib.c performs the calculation the fastest

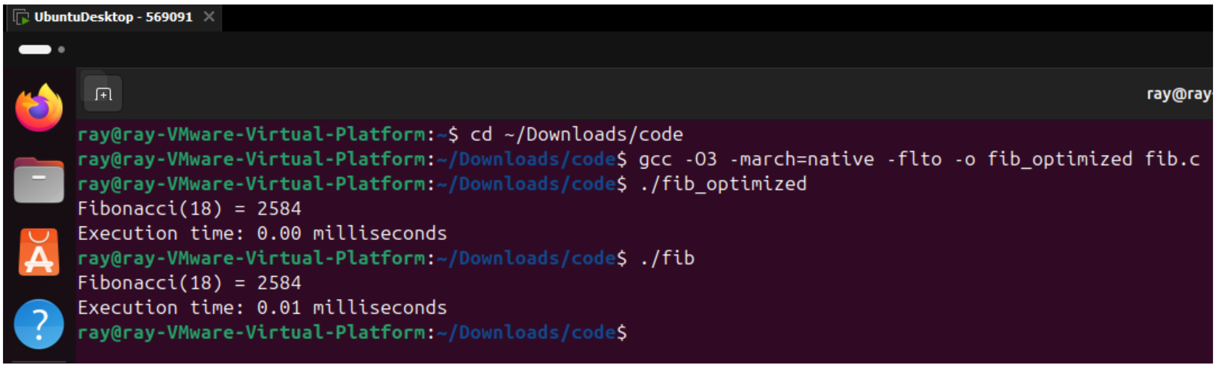
Assignment 4.4: Optimize

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.
O3: Enables aggressive optimizations to maximize performance, but may increase compile time and executable size.
march=native: Optimizes the program for the architecture of the machine it's compiling on
flto: Enables Link Time Optimization, which can provide even better performance.

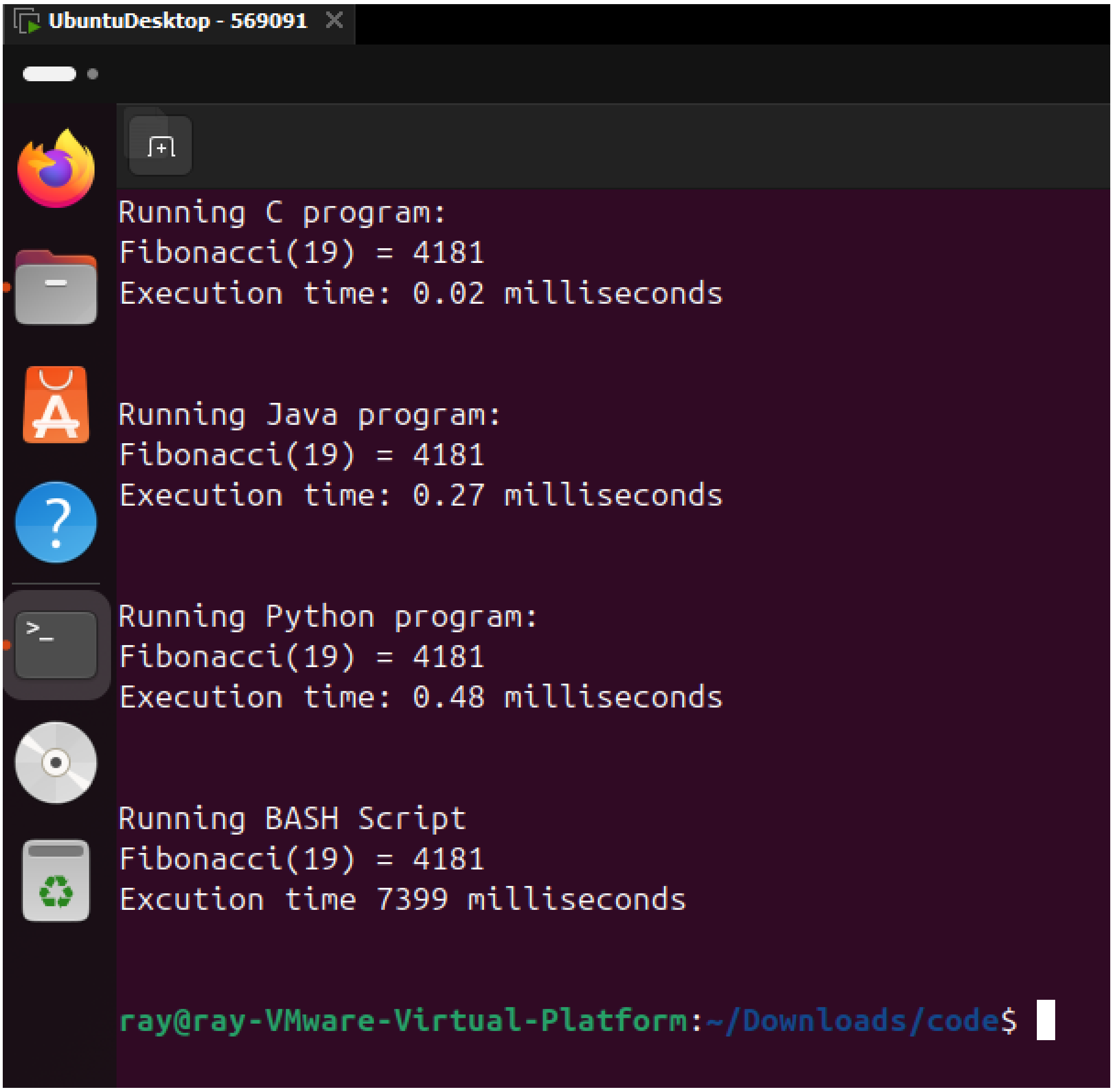
b) Compile **fib.c** again with the optimization parameters
gcc -O3 -march=native -flto -o fib_optimized fib.c

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



yes

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.
Swapped **./fib $n** with **./fib_optimized $n** within **runall.sh**
Provide **runall.sh** with permissions: **chmod +x runall.sh**
Run: **./runall.sh**


UbuntuDesktop - 569091
Running C program:
Fibonacci(19) = 4181
Execution time: 0.02 milliseconds
Running Java program:
Fibonacci(19) = 4181
Execution time: 0.27 milliseconds
Running Python program:
Fibonacci(19) = 4181
Execution time: 0.48 milliseconds
Running BASH Script
Fibonacci(19) = 4181
Excution time 7399 milliseconds
ray@ray-VMware-Virtual-Platform:~/Downloads/code$

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.

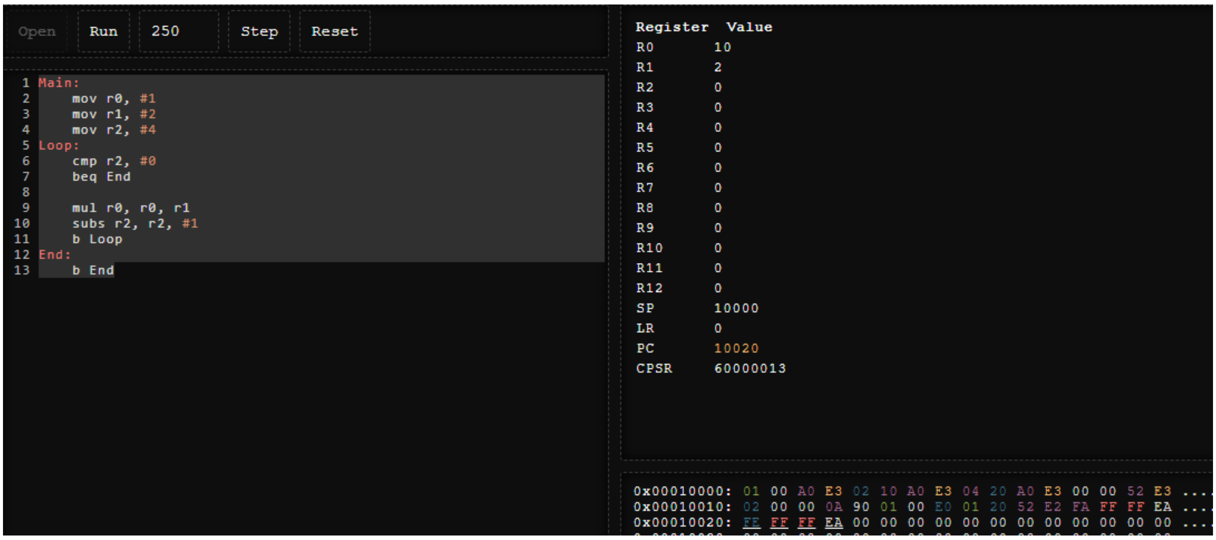
```
Main:

mov r1, #2

mov r2, #4

Loop:

End:
```

Completed code:

```
Main:

   mov r0, #1

   mov r1, #2

          mov r2, #4

Loop:

          cmp r2, #0

          beq End


          mul r0, r0, r1

          subs r2, r2, #1

          b Loop

End:

   b 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**