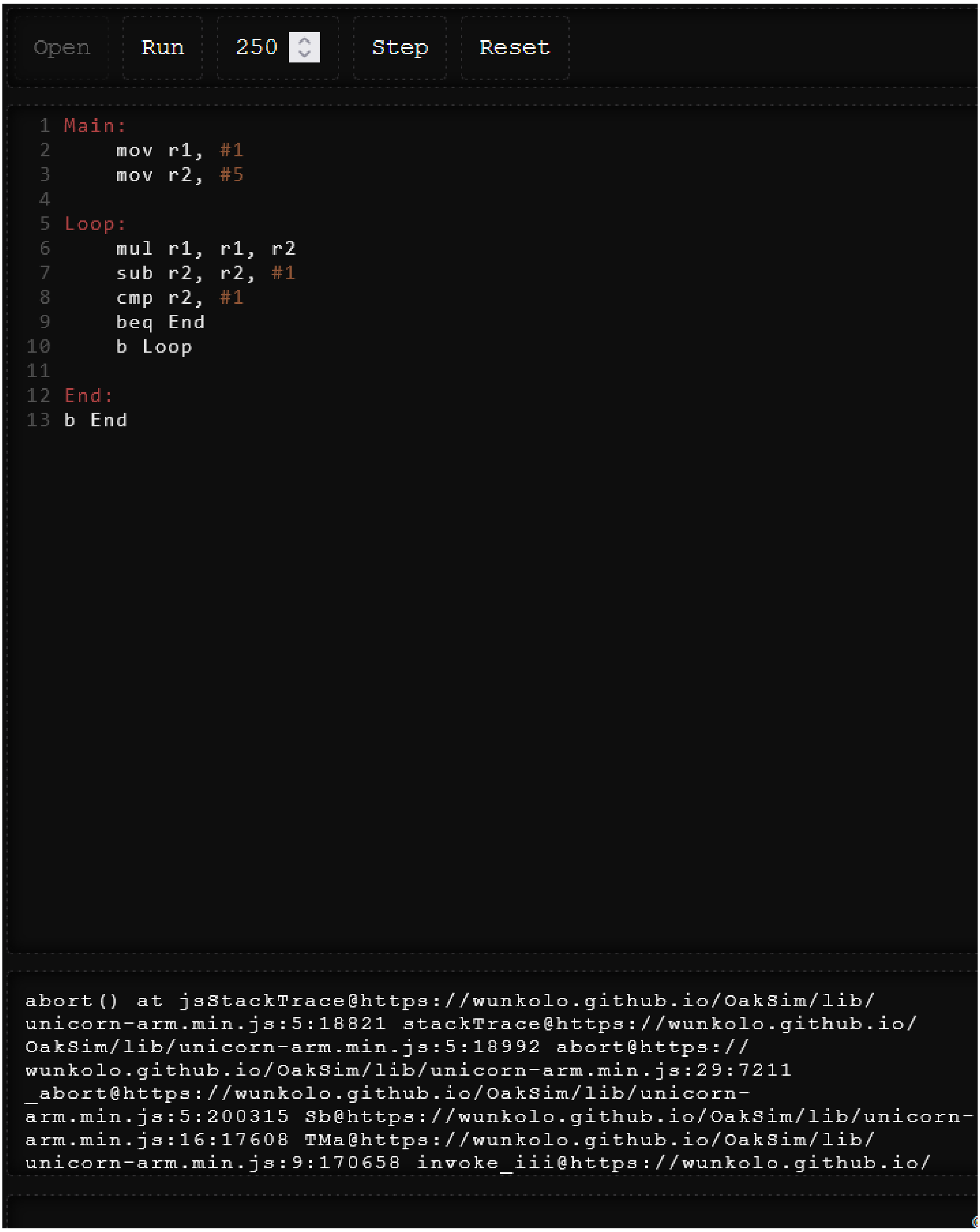
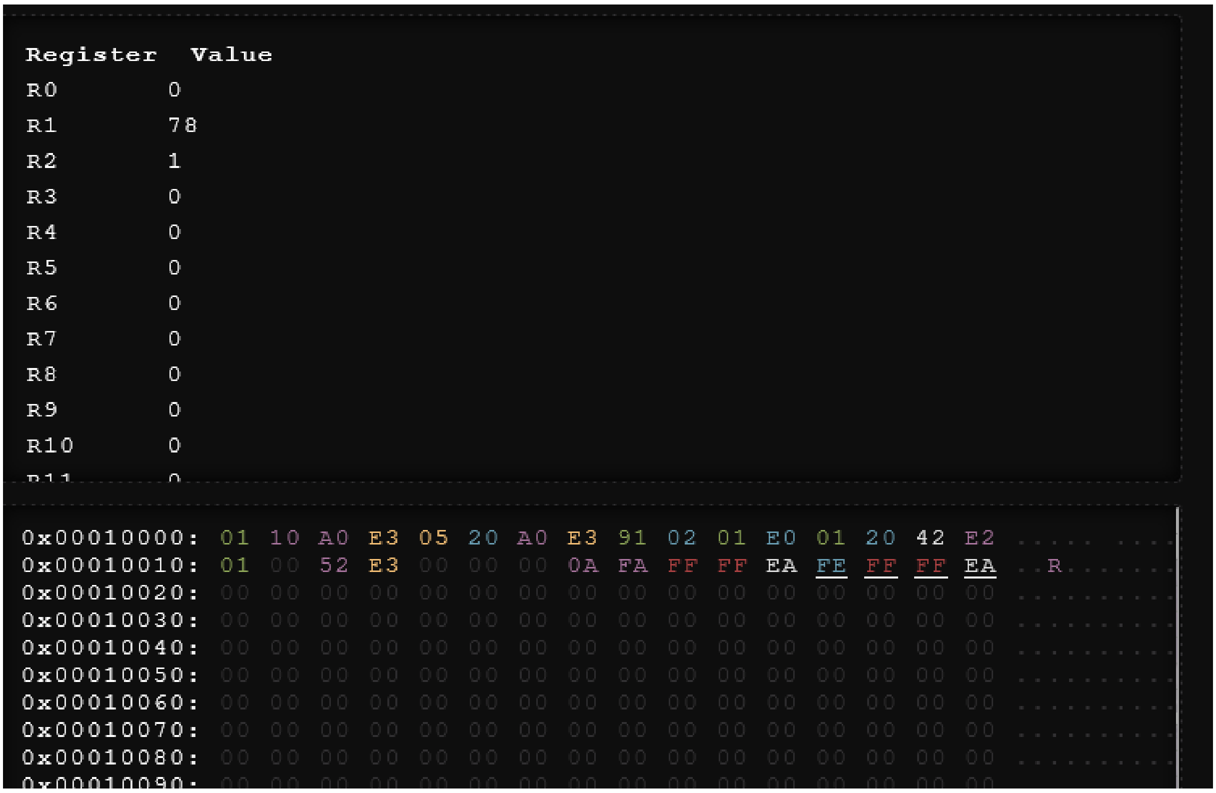
Template Week 4 – Software

Student number: 578634

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

konstantin@konstantin-VMware-Virtual-Platform: ~

```
konstantin@konstantin-VMware-Virtual-Platform:~$ javac --version
javac 21.0.8
konstantin@konstantin-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)
konstantin@konstantin-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.

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

Assignment 4.3: Compile

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

- Fibonnaci.java
- 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?

- Fibonnaci.java

Which source code files are interpreted by an interpreter?

- Fib.py
- Fib.sh

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

- Fastest code file will be fib.c due to it being translated into machine code by a compiler, which is makes it able to be executed directly by the processor.

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 ./fib.sh
- ./fib.sh

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

- Fibonacci.class for Fibonacci.java
- An executable file for fib.c named 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?

```
konstantin@konstantin-VMware-Virtual-Platform:~/Downloads/code$ javac Fibonacci.java

konstantin@konstantin-VMware-Virtual-Platform:~/Downloads/code$ java Fibonacci
Fibonacci(18) = 2584
Execution time: 0.22 milliseconds
konstantin@konstantin-VMware-Virtual-Platform:~/Downloads/code$ python3 fib.py
Fibonacci(18) = 2584
Execution time: 0.30 milliseconds
konstantin@konstantin-VMware-Virtual-Platform:~/Downloads/code$ gcc fib.c -o fib
konstantin@konstantin-VMware-Virtual-Platform:~/Downloads/code$ ./fib
Fibonacci(18) = 2584
Execution time: 0.02 milliseconds
konstantin@konstantin-VMware-Virtual-Platform:~/Downloads/code$ chmod a+x fib.py
konstantin@konstantin-VMware-Virtual-Platform:~/Downloads/code$ ./fib.py
./fib.py: line 1: import: command not found
./fib.py: line 2: import: command not found
./fib.py: line 4: syntax error near unexpected token `('
./fib.py: line 4: `def fibonacci(n):'
konstantin@konstantin-VMware-Virtual-Platform:~/Downloads/code$ sudo chmod a+x fib.sh
[sudo] password for konstantin:
konstantin@konstantin-VMware-Virtual-Platform:~/Downloads/code$ sudo ./fib.sh
Fibonacci(18) = 2584
Excution time 5894 milliseconds
konstantin@konstantin-VMware-Virtual-Platform:~/Downloads/code$
```

The fastest source code is the C file.

Assignment 4.4: Optimize

Take relevant screenshots of the following commands:

- 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.
    a. Parameter found in man gcc: -O2
    b. Explanation: enables a higher level of optimization, making the compiled program run faster.
    c. **gcc -O2 fib.c -o fib**

- Compile **fib.c** again with the optimization parameters

```
konstantin@konstantin-VMware-Virtual-Platform:~/Downloads/code$ man gcc
konstantin@konstantin-VMware-Virtual-Platform:~/Downloads/code$ gcc -O2 fib.c -o fib
konstantin@konstantin-VMware-Virtual-Platform:~/Downloads/code$ time ./fib
Fibonacci(18) = 2584
Execution time: 0.01 milliseconds

real    0m0.002s
user    0m0.001s
sys     0m0.001s
```

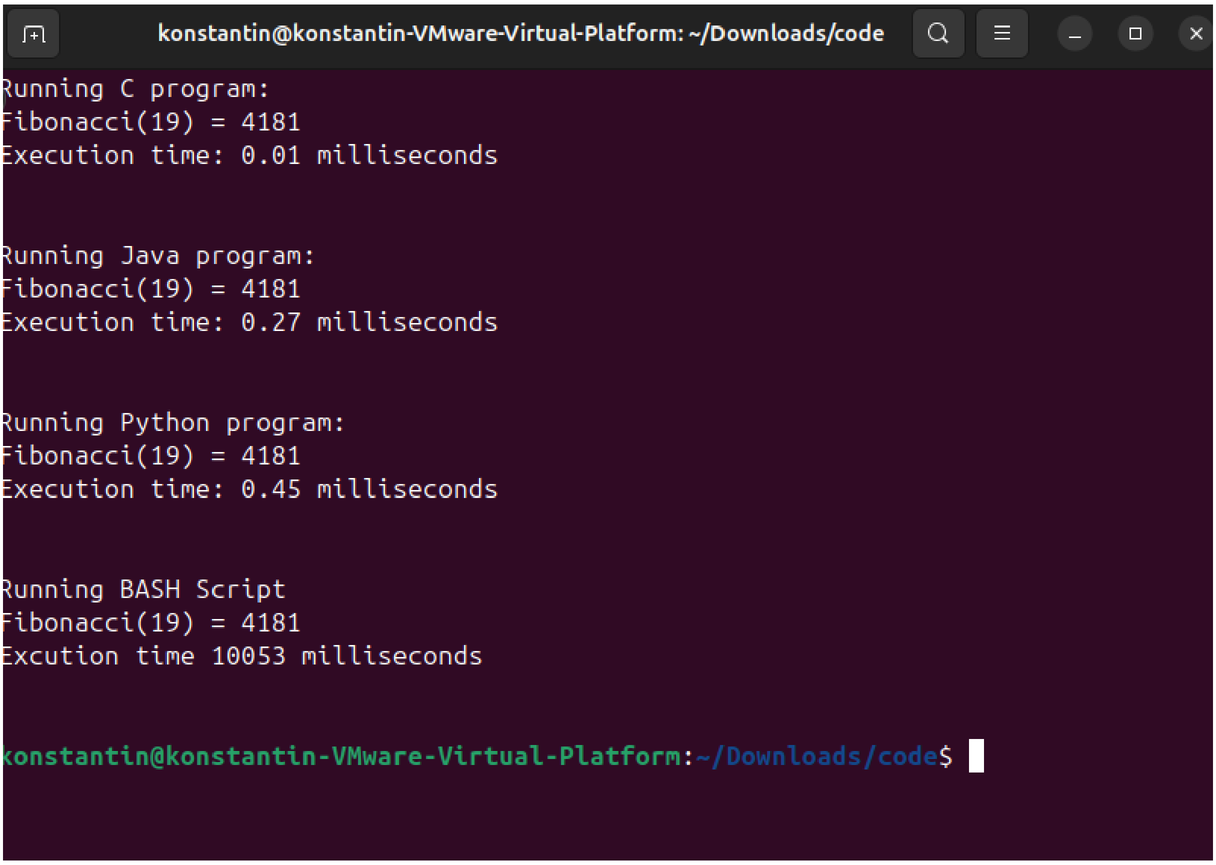
- Run the newly compiled program. Is it true that it now performs the calculation faster?

```
konstantin@konstantin-VMware-Virtual-Platform:~/Downloads/code$ man gcc
konstantin@konstantin-VMware-Virtual-Platform:~/Downloads/code$ gcc -O2 fib.c -o fib
konstantin@konstantin-VMware-Virtual-Platform:~/Downloads/code$ time ./fib
Fibonacci(18) = 2584
Execution time: 0.01 milliseconds

real    0m0.002s
user    0m0.001s
sys     0m0.001s
```

```
konstantin@konstantin-VMware-Virtual-Platform:~/Downloads/code$ gcc fib.c -o fib
konstantin@konstantin-VMware-Virtual-Platform:~/Downloads/code$ ./fib
Fibonacci(18) = 2584
Terminal
Execution time: 0.02 milliseconds
```

Using the parameters makes the execution time faster by 0.01 millisecond

- 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.



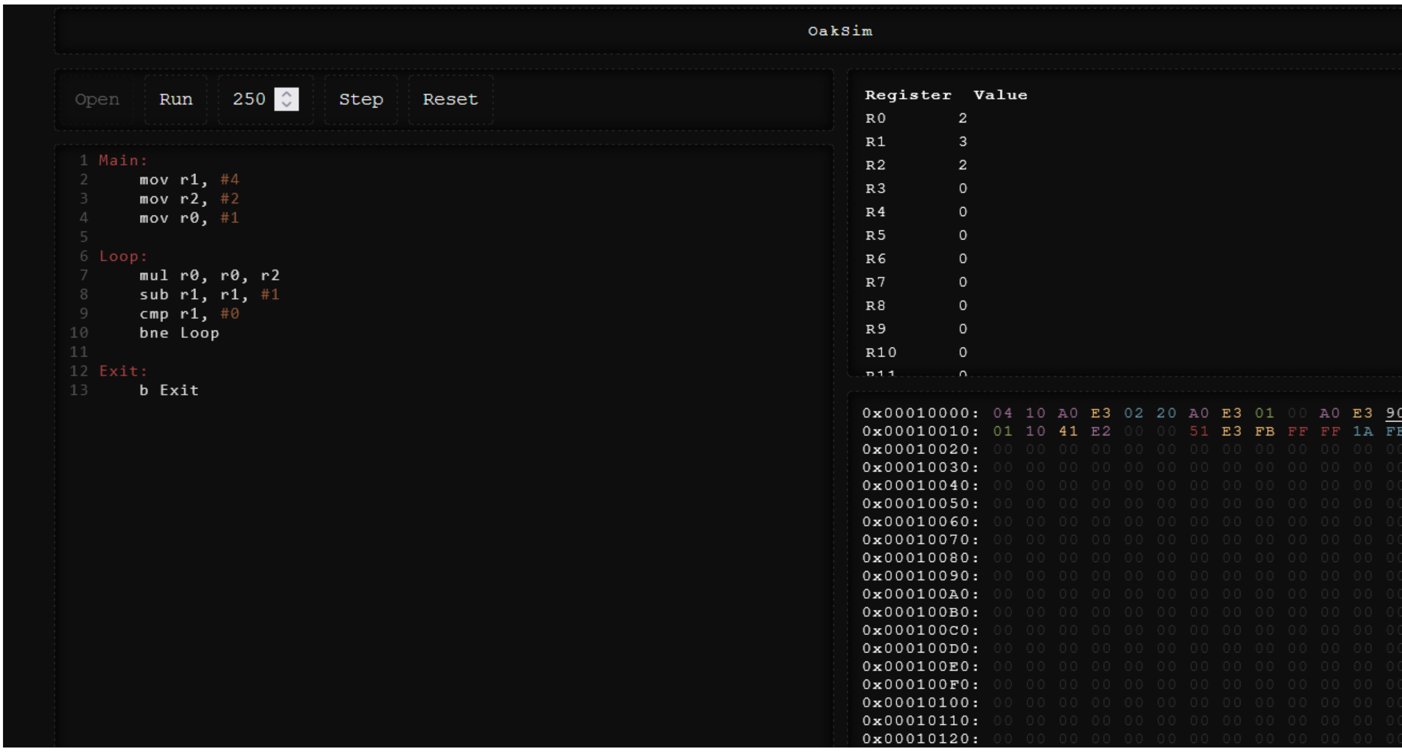
Assignment 4.5: More ARM Assembly

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**