

# Template Week 4 – Software

Student number: 569527

## Assignment 4.1: ARM assembly

Screenshot of working assembly code of factorial calculation:

The screenshot shows the OakSim ARM simulator interface. The top bar includes navigation icons and the URL `wunkolo.github.io`. The main window is titled "OakSim" and contains three panels:

- Assembly Code Panel:** Displays the following code:

```
1
2 _start:
3   mov r2, #5
4   mov r1, #1
5
6 Loop:
7   mul r1, r1, r2
8   sub r2, r2, #1
9   cmp r2, #0
10  bgt Loop
11
12 End:
13   b End
14
15 569527|
```
- Registers Panel:** A table showing the state of 11 registers:

Register	Value
R0	0
R1	78
R2	0
R3	0
R4	0
R5	0
R6	0
R7	0
R8	0
R9	0
R10	0
R11	0
- Memory Panel:** Displays a hex dump of memory starting from address `0x00010000`. The first few lines show:

```
0x00010000: 05 20 A0 E3 01 10 A0 E3 91 02 01 E0 01 20 42 E2 . . .
0x00010010: 00 00 52 E3 FB FF FF CA FE FF FF EA 00 00 00 00 . R
0x00010020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 . . .
```

At the bottom, a stack trace is visible, indicating an `abort()` error at `jsStackTrace` with a link to `https://wunkolo.github.io/OakSim/lib/unicorn-arm.min.js:5:18821`.

## Assignment 4.2: Programming languages

Take screenshots that the following commands work:

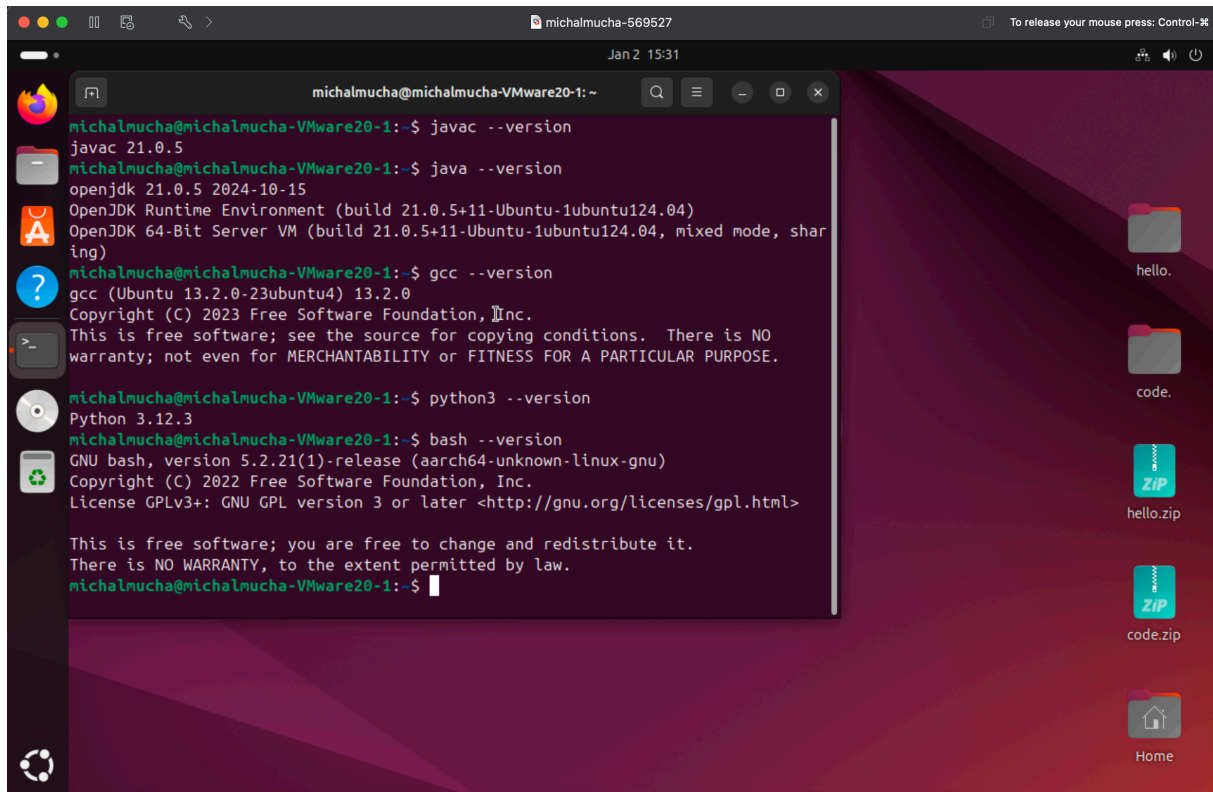
```
javac --version
```

```
java --version
```

```
gcc --version
```

```
python3 --version
```

```
bash --version
```



The screenshot shows a terminal window titled "michalmuchu@569527" with a date and time of "Jan 2 15:31". The terminal is running a series of commands to check the versions of various programming languages and tools. The output for each command is as follows:

```
michalmuchu@michalmuchu-VMware20-1:~$ javac --version
javac 21.0.5
michalmuchu@michalmuchu-VMware20-1:~$ java --version
openjdk 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, sharing)
michalmuchu@michalmuchu-VMware20-1:~$ 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.
michalmuchu@michalmuchu-VMware20-1:~$ python3 --version
Python 3.12.3
michalmuchu@michalmuchu-VMware20-1:~$ bash --version
GNU bash, version 5.2.21(1)-release (aarch64-unknown-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.
michalmuchu@michalmuchu-VMware20-1:~$
```

The terminal window is open on a desktop environment with a dark purple background. On the desktop, there are several icons: a folder named "hello.", a folder named "code.", a ZIP file named "hello.zip", a ZIP file named "code.zip", and a "Home" icon. The terminal window has a title bar with standard window controls (minimize, maximize, close) and a search icon.

### Assignment 4.3: Compile

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

**Fibonacci.java** (Java): Needs to be compiled to bytecode using javac.

**fib.c** (C): Needs to be compiled to machine code using gcc.

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**: 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**

How do I run a Java program?

Java Fibonacci

How do I run a Python program?

python3 fib.py

How do I run a C program?

./fib

How do I run a Bash script?

chmod +x fib.sh

./fib.sh

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

**Fibonacci.java**: Compilation creates a bytecode file named **Fibonacci.class**.

**fib.c**: Compilation creates a machine code executable, typically named **a.out** (default) or a custom name specified with **-o**, such as **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?  
Fastest is C code

The image consists of two screenshots of a terminal window on a Linux desktop. The desktop has a dark purple background with icons for 'hello.zip', '4pixels.bmp', 'code.zip', and 'IT Fundamentals 1.2'. A file manager window is open, showing the contents of the '/home/michalmucha/Desktop' directory, which includes folders 'code', 'hello', and 'IT', and files '4pixels.bmp' and 'code.zip'.

The terminal window shows the following commands and output:

```
michalmucha@michalmucha-VMware20-1: ~/Desktop/code
michalmucha@michalmucha-VMware20-1: ~/Desktop/code$ javac Fibonacci.java
michalmucha@michalmucha-VMware20-1: ~/Desktop/code$ gcc -o fib fib.c
michalmucha@michalmucha-VMware20-1: ~/Desktop/code$ sudo chmod +x fib.sh
[sudo] password for michalmucha:
michalmucha@michalmucha-VMware20-1: ~/Desktop/code$
```

The second screenshot shows the execution of the programs with timing information:

```
michalmucha@michalmucha-VMware20-1: ~/Desktop/code$ time ./fib
Fibonacci(18) = 2584
Execution time: 0.02 milliseconds
real    0m0.002s
user    0m0.000s
sys     0m0.002s
michalmucha@michalmucha-VMware20-1: ~/Desktop/code$ time java Fibonacci
Fibonacci(18) = 2584
Execution time: 0.15 milliseconds
real    0m0.077s
user    0m0.058s
sys     0m0.020s
michalmucha@michalmucha-VMware20-1: ~/Desktop/code$ time python3 fib.py
Fibonacci(18) = 2584
Execution time: 0.39 milliseconds
real    0m0.030s
user    0m0.019s
sys     0m0.007s
michalmucha@michalmucha-VMware20-1: ~/Desktop/code$ time ./fib.sh
Fibonacci(18) = 2584
Execution time: 2240 milliseconds
real    0m2.245s
user    0m1.587s
sys     0m0.481s
michalmucha@michalmucha-VMware20-1: ~/Desktop/code$
```

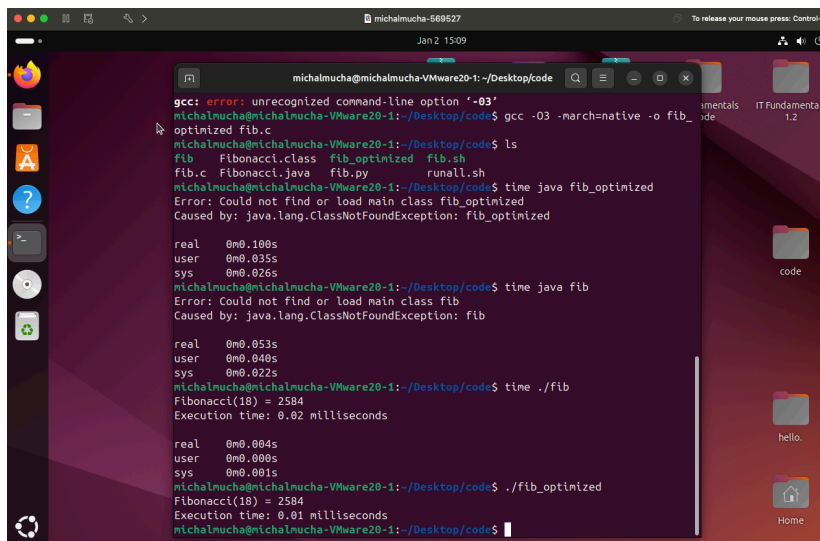
## Assignment 4.4: Optimize

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.

```
gcc -O3 -march=native -o fib_optimized fib.c
```

- Compile **fib.c** again with the optimization parameters
- Run the newly compiled program. Is it true that it now performs the calculation faster?



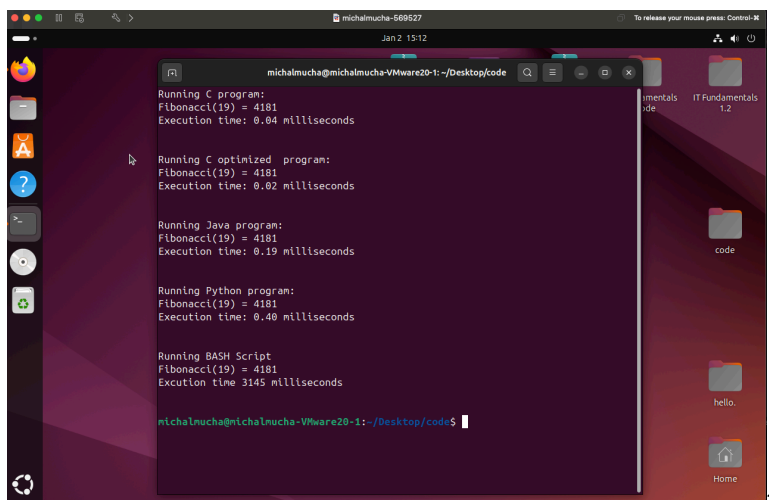
```
michalmuch@micahmuch-VirtualBox: ~/Desktop/code
gcc: error: unrecognized command-line option '-O3'
michalmuch@micahmuch-VirtualBox:~/Desktop/code$ gcc -O3 -march=native -o fib_optimized fib.c
michalmuch@micahmuch-VirtualBox:~/Desktop/code$ ls
fib      Fibonacci.class  fib_optimized  fib.sh
fib.c    Fibonacci.java   fib.py         runall.sh
michalmuch@micahmuch-VirtualBox:~/Desktop/code$ time java fib_optimized
Error: Could not find or load main class fib_optimized
Caused by: java.lang.ClassNotFoundException: fib_optimized

real    0m0.100s
user    0m0.035s
sys     0m0.026s
michalmuch@micahmuch-VirtualBox:~/Desktop/code$ time java fib
Error: Could not find or load main class fib
Caused by: java.lang.ClassNotFoundException: fib

real    0m0.053s
user    0m0.040s
sys     0m0.022s
michalmuch@micahmuch-VirtualBox:~/Desktop/code$ time ./fib
Fibonacci(18) = 2584
Execution time: 0.02 milliseconds

real    0m0.004s
user    0m0.000s
sys     0m0.001s
michalmuch@micahmuch-VirtualBox:~/Desktop/code$ time ./fib_optimized
Fibonacci(18) = 2584
Execution time: 0.01 milliseconds
michalmuch@micahmuch-VirtualBox:~/Desktop/code$
```

- 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.



```
michalmuch@micahmuch-VirtualBox:~/Desktop/code
Running C program:
Fibonacci(19) = 4181
Execution time: 0.04 milliseconds

Running C optimized program:
Fibonacci(19) = 4181
Execution time: 0.02 milliseconds

Running Java program:
Fibonacci(19) = 4181
Execution time: 0.19 milliseconds

Running Python program:
Fibonacci(19) = 4181
Execution time: 0.40 milliseconds

Running BASH Script
Fibonacci(19) = 4181
Execution time 3145 milliseconds

michalmuch@micahmuch-VirtualBox:~/Desktop/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
```

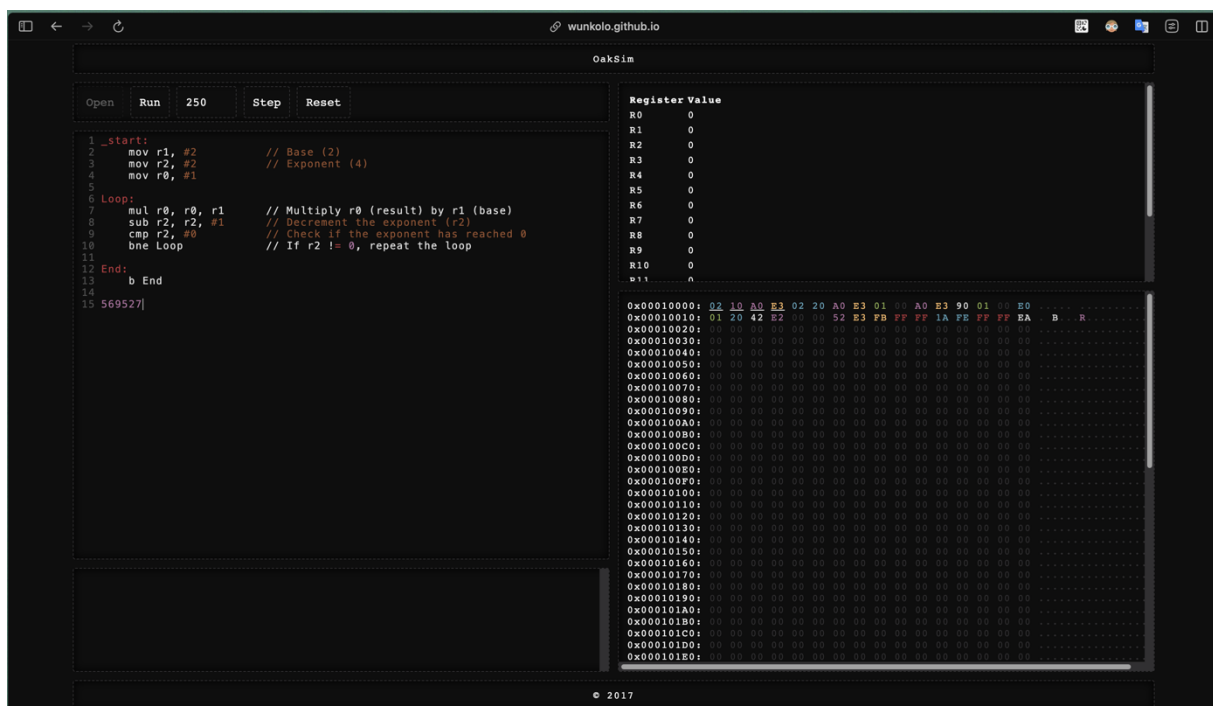
```
mov r2, #4
```

Loop:

End:

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](#)