

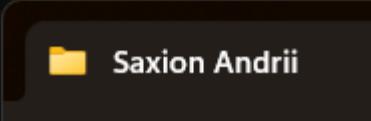
# Week 4 – Software.

Student number: 578856

## Assignment 4.1: ARM assembly

Screenshot of working assembly code of factorial calculation:

```
1 main:
2     mov r2, #5
3     mov r1, #1
4 loop:
5     mul r1, r1, r2
6     sub r2, r2, #1
7     cmp r1, #120
8     beq Exit
9     b Loop
10 exit:
```



## Assignment 4.2: Programming languages

Take screenshots that the following commands work:

```
javac --version
```

```
java --version
```

```
gcc --version
```

```
python3 --version
```

```
bash --version
```

```
andrii@andrii-VM:~$ java --version
openjdk 21.0.9 2025-10-21
OpenJDK Runtime Environment (build 21.0.9+10-Ubuntu-124.04)
OpenJDK 64-Bit Server VM (build 21.0.9+10-Ubuntu-124.04, mixed mode, sharing)
andrii@andrii-VM:~$ javac --version
javac 21.0.9
andrii@andrii-VM:~$ 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.

andrii@andrii-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.
andrii@andrii-VM:~$ python3 --version
Python 3.12.3
andrii@andrii-VM:~$
```

### **Assignment 4.3: Compile**

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

Fibonacci.java needs to be compiled via javac. Also fib.c needs to be compiled via gcc.

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

When you compile fib.c, you get pure machine code, which can be directly executable by a processor.

Which source code files are compiled to byte code?

After compiling Fibonacci.java using javac, you get bytecode with .class extension, those files can be executable only with the help of Java Virtual Machine.

Which source code files are interpreted by an interpreter?

fib.py and fib.sh are interpreted by interpreters, so every time you run them, interpreter translates it to CPU.

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

fib.c will perform calculations the fastest, because once you compile the file, it becomes a native machine code, which CPU can directly run, with no interpretation and no VMs.

How do I run a Java program?

To run java program, firstly you need to compile the file using javac Fibonacci.java in terminal, and you will get Fibonacci.class. After that, run Fibonacci.class using java Fibonacci in terminal.

How do I run a Python program?

To run python program, use python3 fib.py in terminal.

How do I run a C program?

To run C program, compile the file with gcc -o fib fib.c. Then run it using ./fib in terminal.

How do I run a Bash script?

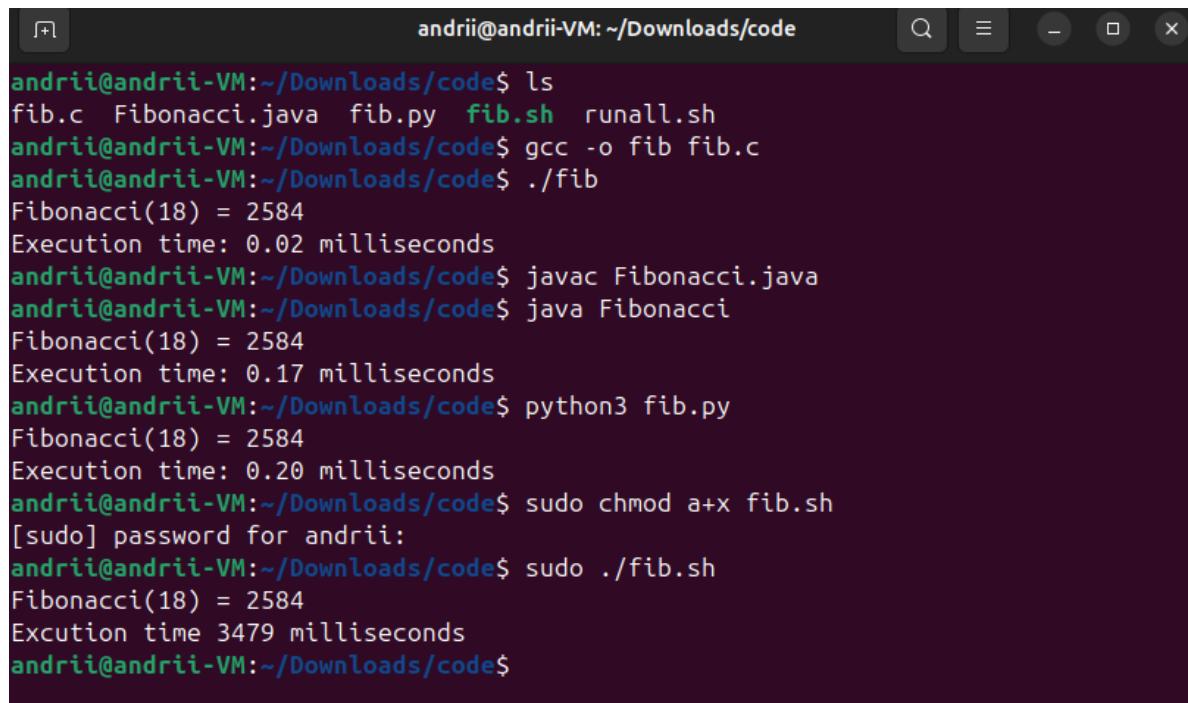
To run bash script, firstly you need to give a permission with sudo chmod a+x fib.sh, then run it with ./fib.sh.

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

If you compile fib.c, file fib will be created. If the Fibonacci.java is compiled, the Fibonacci.class will be created. You don't need to compile .py and .sh files so no additional files will be created.

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?



```
andrii@andrii-VM:~/Downloads/code$ ls
fib.c Fibonacci.java fib.py fib.sh runall.sh
andrii@andrii-VM:~/Downloads/code$ gcc -o fib fib.c
andrii@andrii-VM:~/Downloads/code$ ./fib
Fibonacci(18) = 2584
Execution time: 0.02 milliseconds
andrii@andrii-VM:~/Downloads/code$ javac Fibonacci.java
andrii@andrii-VM:~/Downloads/code$ java Fibonacci
Fibonacci(18) = 2584
Execution time: 0.17 milliseconds
andrii@andrii-VM:~/Downloads/code$ python3 fib.py
Fibonacci(18) = 2584
Execution time: 0.20 milliseconds
andrii@andrii-VM:~/Downloads/code$ sudo chmod a+x fib.sh
[sudo] password for andrii:
andrii@andrii-VM:~/Downloads/code$ sudo ./fib.sh
Fibonacci(18) = 2584
Execution time 3479 milliseconds
andrii@andrii-VM:~/Downloads/code$
```

Compiled C file performs the calculation the fastest because this file is directly executable for CPU.

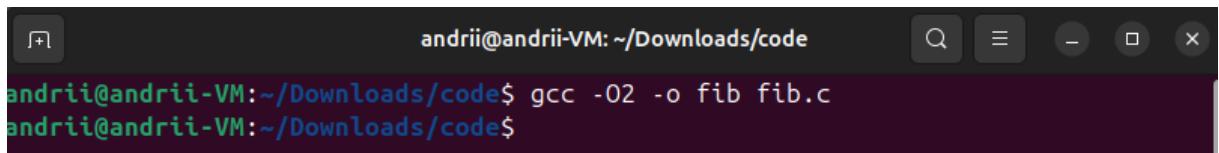
#### Assignment 4.4: Optimize

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.

To run the code faster you need to pass -O2 parameter to compiler and it optimize it.

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



```
andrii@andrii-VM:~/Downloads/code$ gcc -O2 -o fib fib.c  
andrii@andrii-VM:~/Downloads/code$
```

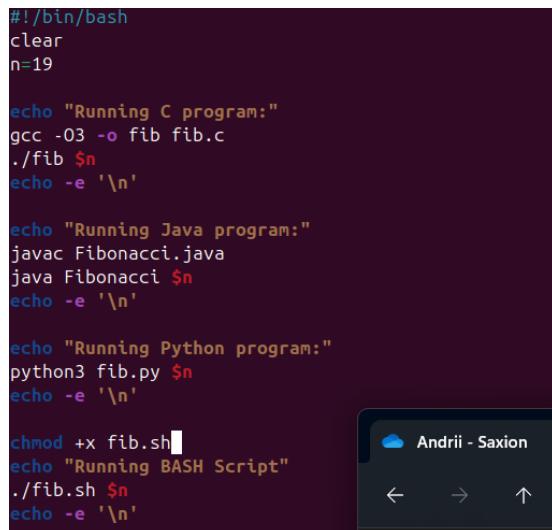
- Run the newly compiled program. Is it true that it now performs the calculation faster?



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

It runs 0.02 milliseconds faster.

- 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.



```
#!/bin/bash  
clear  
n=19  
  
echo "Running C program:"  
gcc -O3 -o fib fib.c  
./fib $n  
echo -e '\n'  
  
echo "Running Java program:"  
javac Fibonacci.java  
java Fibonacci $n  
echo -e '\n'  
  
echo "Running Python program:"  
python3 fib.py $n  
echo -e '\n'  
  
chmod +x fib.sh  
echo "Running BASH Script"  
./fib.sh $n  
echo -e '\n'
```

```
Running C program:  
Fibonacci(19) = 4181  
Execution time: 0.01 milliseconds  
  
Running Java program:  
Fibonacci(19) = 4181  
Execution time: 0.19 milliseconds  
  
Running Python program:  
Fibonacci(19) = 4181  
Execution time: 0.32 milliseconds  
  
Running BASH Script  
Fibonacci(19) = 4181  
Excution time 5589 milliseconds  
  
andrii@andrii-VM:~/Downloads/code$
```

#### 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.

```
1 main:  
2     mov r0, #1  
3     mov r1, #2  
4     mov r2, #4  
5 loop:  
6     cmp r2, #0  
7     beq end  
8     sub r2, r2, #1  
9     mul r0, r0, r1  
10    b loop  
11 end:
```

Saxion Andrii

