

# Template Week 4 – Software

Student number: 590173

## Assignment 4.1: ARM assembly

Screenshot of working assembly code of factorial calculation:

The screenshot shows a debugger interface with the following details:

- Registers:** Shows register values for R0 through R11, all initialized to 0.
- Memory:** A dump of memory starting at address 0x00010000, showing assembly instructions in hex and their corresponding binary representation.
- Code:** The assembly code for the factorial calculation:

```
1 main:
2 mov r2, #5
3 mov r1, #1 ; mohammed 590173
4
5 loop:
6 mul r1, r1, r2
7 add r1, r1, #1
8 cmp r2, #1
9 bgt loop
10 end:
11 halt
```

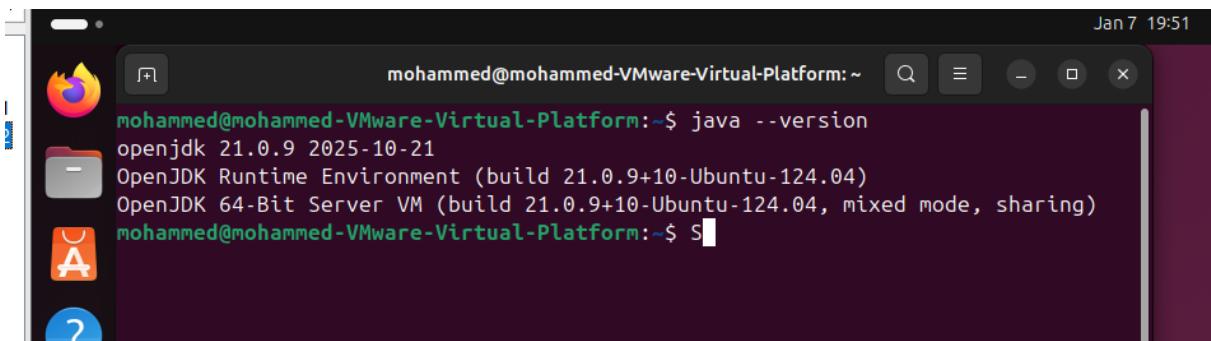
## Assignment 4.2: Programming languages

Take screenshots that the following commands work:

```
javac --version
```

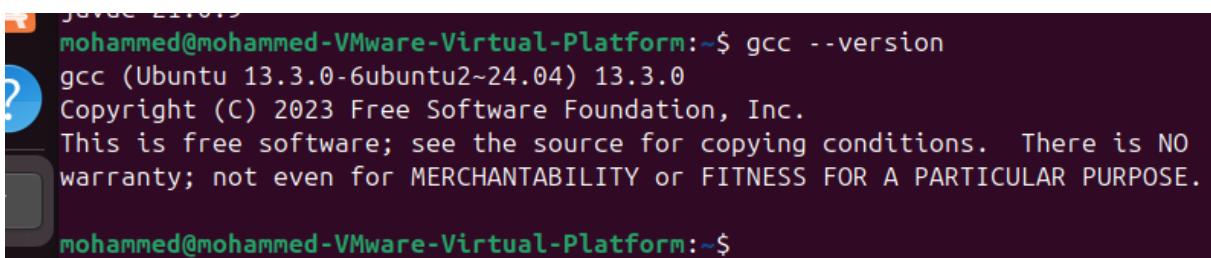
```
mohammed@mohammed-Virtual-Platform:~$ javac --version
javac 21.0.9
mohammed@mohammed-Virtual-Platform:~$
```

```
java --version
```



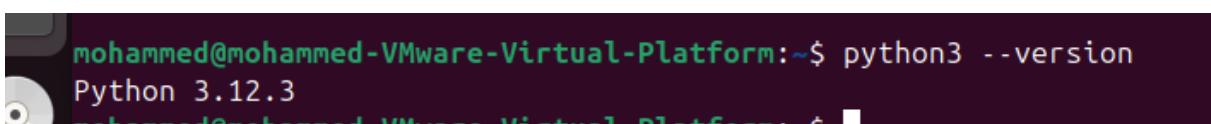
mohammed@mohammed-Virtual-Platform:~\$ java --version  
openjdk 21.0.9 2025-10-21  
OpenJDK Runtime Environment (build 21.0.9+10-Ubuntu-124.04)  
OpenJDK 64-Bit Server VM (build 21.0.9+10-Ubuntu-124.04, mixed mode, sharing)  
mohammed@mohammed-Virtual-Platform:~\$ S

gcc --version



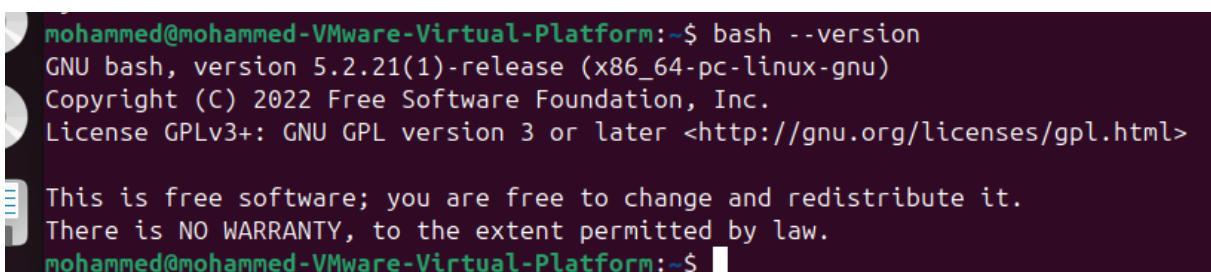
mohammed@mohammed-Virtual-Platform:~\$ gcc --version  
gcc (Ubuntu 13.3.0-6ubuntu2~24.04) 13.3.0  
Copyright (C) 2023 Free Software Foundation, Inc.  
This is free software; see the source for copying conditions. There is NO  
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  
mohammed@mohammed-Virtual-Platform:~\$

python3 --version



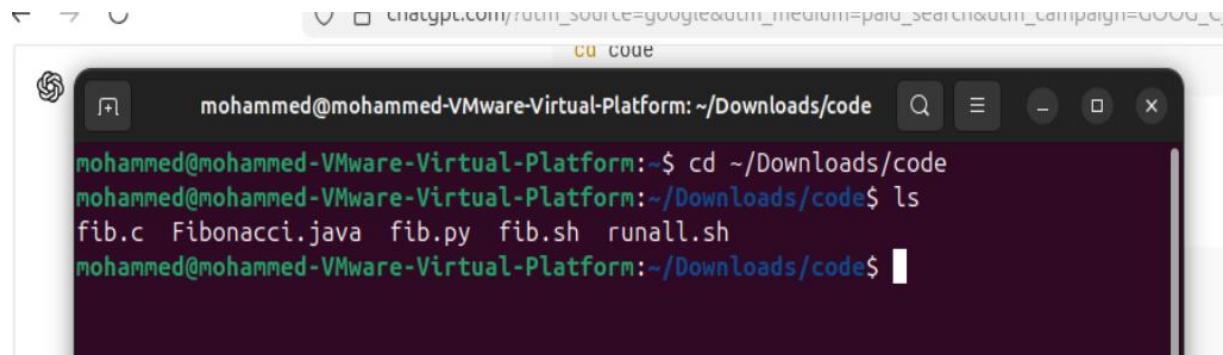
mohammed@mohammed-Virtual-Platform:~\$ python3 --version  
Python 3.12.3  
mohammed@mohammed-Virtual-Platform:~\$

bash --version



mohammed@mohammed-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.  
mohammed@mohammed-Virtual-Platform:~\$

### Assignment 4.3: Compile



```
mohammed@mohammed-Virtual-Platform:~/Downloads/code$ cd ~/Downloads/code
mohammed@mohammed-Virtual-Platform:~/Downloads/code$ ls
fib.c Fibonacci.java fib.py fib.sh runall.sh
mohammed@mohammed-Virtual-Platform:~/Downloads/code$
```

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

**fib.c → YES (must be compiled)**

**Fibonacci.java → YES (must be compiled)**

**fib.py → NO**

**fib.sh → NO**

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

**fib.c C compiles directly to CPU instructions.**

Which source code files are compiled to byte code?

**Fibonacci.java Java compiles to .class bytecode and runs on the JVM**

Which source code files are interpreted by an interpreter?

**fib.py (Python interpreter)**

**fib.sh (Bash interpreter)**

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

**The C program (fib.c) will be expected to perform the calculation the fastest.**

How do I run a Java program?

**To run a Java program, you first need to compile it using javac, and then run the compiled program with java.**

How do I run a Python program?

**Python is an interpreted language, so you don't need to compile it. You just run it directly with the python3 command.**

How do I run a C program?

**To run a C program, you need to compile it into an executable first and then run that file.**

How do I run a Bash script?

**To run a Bash script, you can either make it executable and then run it, or just run it directly using bash.**

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

**Yes, new files will be created when you compile the programs:**

**For fib.c (C program):**

**A new file called fib will be created after compiling. This is the executable file.**

**For Fibonacci.java (Java program):**

**A new file called Fibonacci.class will be created after compiling. This is the bytecode file.**

Take relevant screenshots of the following commands:

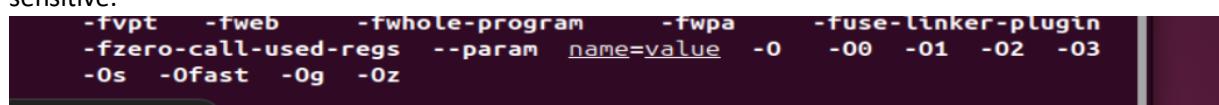
- Compile the source files where necessary
- Make them executable
- Run them
- Which (compiled) source code file performs the calculation the fastest?

```
mohammed@mohammed-VMware-Virtual-Platform:~/Downloads/code$ cd ~/Downloads/code
mohammed@mohammed-VMware-Virtual-Platform:~/Downloads/code$ ls
fib.c Fibonacci.java fib.py fib.sh runall.sh
mohammed@mohammed-VMware-Virtual-Platform:~/Downloads/code$ gcc fib.c -o fib
mohammed@mohammed-VMware-Virtual-Platform:~/Downloads/code$ javac Fibonacci.java
mohammed@mohammed-VMware-Virtual-Platform:~/Downloads/code$ ls
fib fib.c Fibonacci.class Fibonacci.java fib.py fib.sh runall.sh
mohammed@mohammed-VMware-Virtual-Platform:~/Downloads/code$ chmod +x fib.sh
mohammed@mohammed-VMware-Virtual-Platform:~/Downloads/code$ ls
fib fib.c Fibonacci.class Fibonacci.java fib.py fib.sh runall.sh
mohammed@mohammed-VMware-Virtual-Platform:~/Downloads/code$ ./fib
Fibonacci(18) = 2584
Execution time: 0.05 milliseconds
mohammed@mohammed-VMware-Virtual-Platform:~/Downloads/code$ java Fibonacci
Error: Could not find or load main class Fibnonacci
Caused by: java.lang.ClassNotFoundException: Fibnonacci
mohammed@mohammed-VMware-Virtual-Platform:~/Downloads/code$ java Fibonacci
Fibonacci(18) = 2584
Execution time: 0.71 milliseconds
mohammed@mohammed-VMware-Virtual-Platform:~/Downloads/code$ ./fib.sh
Fibonacci(18) = 2584
Excution time 11187 milliseconds
mohammed@mohammed-VMware-Virtual-Platform:~/Downloads/code$ python3 fib.py
Fibonacci(18) = 2584
Execution time: 1.15 milliseconds
mohammed@mohammed-VMware-Virtual-Platform:~/Downloads/code$
```

#### Assignment 4.4: Optimize

Take relevant screenshots of the following commands:

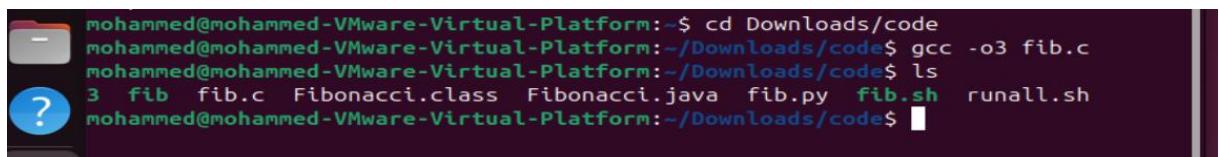
- Figure out which parameters you need to pass to **the gcc** compiler so that the compiler performs a number of optimizations that will ensure that the compiled source code will run faster. **Tip!** The parameters are usually a letter followed by a number. Also read **page 191** of your book, but find a better optimization in the man pages. Please note that Linux is case sensitive.



```
-fvpt -fweb -fwhole-program -fwpa -fuse-linker-plugin
-fzero-call-used-reg -param name=value -O -O0 -O1 -O2 -O3
-Os -Ofast -Og -Oz
```

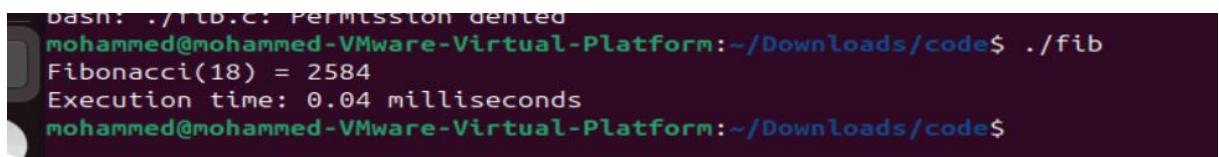
in the world of gcc, the higher the number, the harder the compiler works to make your code fast. While -O2 is good, -O3 turns on even more "speed tricks" to ensure your C program is the fastest in the race.

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



```
mohammed@mohammed-VMware-Virtual-Platform:~/Downloads/code$ cd Downloads/code
mohammed@mohammed-VMware-Virtual-Platform:~/Downloads/code$ gcc -O3 fib.c
mohammed@mohammed-VMware-Virtual-Platform:~/Downloads/code$ ls
3 fib fib.c Fibonacci.class Fibonacci.java fib.py fib.sh runall.sh
mohammed@mohammed-VMware-Virtual-Platform:~/Downloads/code$
```

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



```
dash: ./fib: Permission denied
mohammed@mohammed-VMware-Virtual-Platform:~/Downloads/code$ ./fib
Fibonacci(18) = 2584
Execution time: 0.04 milliseconds
mohammed@mohammed-VMware-Virtual-Platform:~/Downloads/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.

```

mohammed@mohammed-VMware-Virtual-Platform:~/Downloads/code$ time ./fib
Fibonacci(18) = 2584
Execution time: 0.04 milliseconds

real    0m0.003s
user    0m0.000s
sys     0m0.003s
mohammed@mohammed-VMware-Virtual-Platform:~/Downloads/code$ time java Fibonacci # java
Fibonacci(18) = 2584
Execution time: 0.26 milliseconds
Ubuntu 24.04.3 LTS amd64
user    0m0.020s
sys     0m0.123s
mohammed@mohammed-VMware-Virtual-Platform:~/Downloads/code$ time python3 fib.py
Fibonacci(18) = 2584
Execution time: 0.82 milliseconds

real    0m0.035s
user    0m0.017s
sys     0m0.011s
mohammed@mohammed-VMware-Virtual-Platform:~/Downloads/code$ time bash fib.sh
Fibonacci(18) = 2584
Execution time 4403 milliseconds

real    0m4.419s
user    0m2.529s
sys     0m2.267s

```

#### 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.

The screenshot shows a debugger interface with the following components:

- Top Bar:** Buttons for Open, Run (highlighted), Step, and Reset.
- Assembly Code:**

```
1 mov r0, #1
2 mov r1, #4
3 loop:
4 add r1, #0
5 beq end
6 lsl r0,r0,#1
7
8 sub r1, r1, #1
10 b loop
11
12 end:
13 : mohammed alsewari 590173
```
- Registers:** A table showing register values:

Register	Value
R0	10
R1	0
R2	0
R3	0
R4	0
R5	0
R6	0
R7	0
R8	0
R9	0
R10	0
R11	0
R12	0
- Memory Dump:** Hex dump of memory starting at address 0x00010000.

Address	Value
0x00010000	01 00 A0 E3 04 10 A0 E3 00 00 51 E3 02 00 00 0A
0x00010010	00 00 A0 E1 01 10 41 E2 FA FF EA 00 00 00 00 00 00
0x00010020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00010030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00010040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00010050	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00010060	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00010070	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00010080	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00010090	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000100A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000100B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Ready? Save this file and export it as a pdf file with the name: [week4.pdf](#)