

Template Week 4 – Software

Student number: 589531

Assignment 4.1: ARM assembly

Screenshot of working assembly code of factorial calculation:

The screenshot shows a debugger interface with the following details:

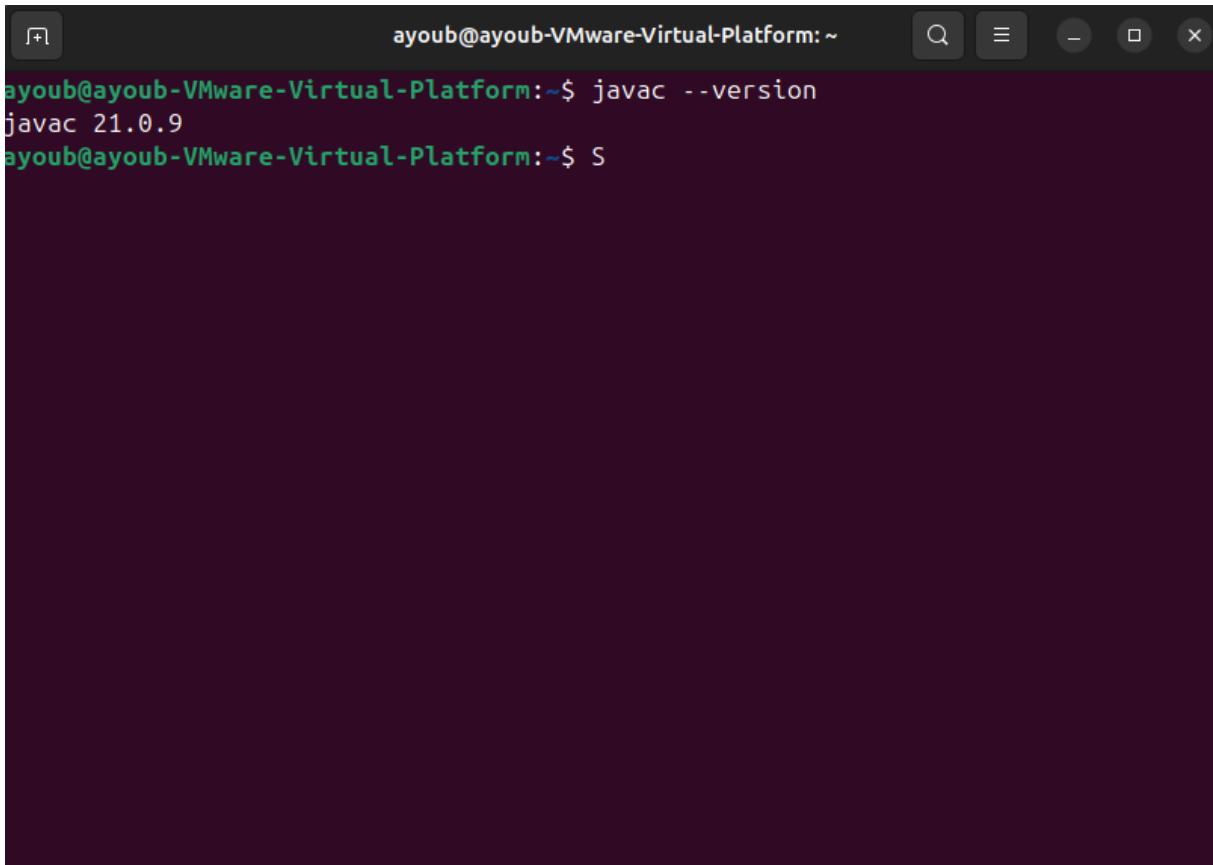
- Registers:** R0=0, R1=78, R2=1, R3=0, R4=0, R5=0, R6=0, R7=0, R8=0, R9=0, R10=0.
- Memory Dump:** A large dump of memory starting at address 0x000010000, showing mostly zeros with some scattered values like 78 and 1.
- Assembly Code:**

```
open Run 250 Step Reset
4 Main:
5     mov r2, #5
6     mov r1, #1
7
8 Loop:
9     mul r1, r1, r2
10    sub r2, r2, #1
11    cmp r2, #1
12    beq End
13    b Loop
14
15 End:
16 b End
17
```

Assignment 4.2: Programming languages

Take screenshots that the following commands work:

javac –version



A screenshot of a terminal window titled "ayoub@ayoub-VMware-Virtual-Platform:~". The window shows the command "javac --version" being run, which outputs "javac 21.0.9". The terminal has a dark background and light-colored text.

java –version

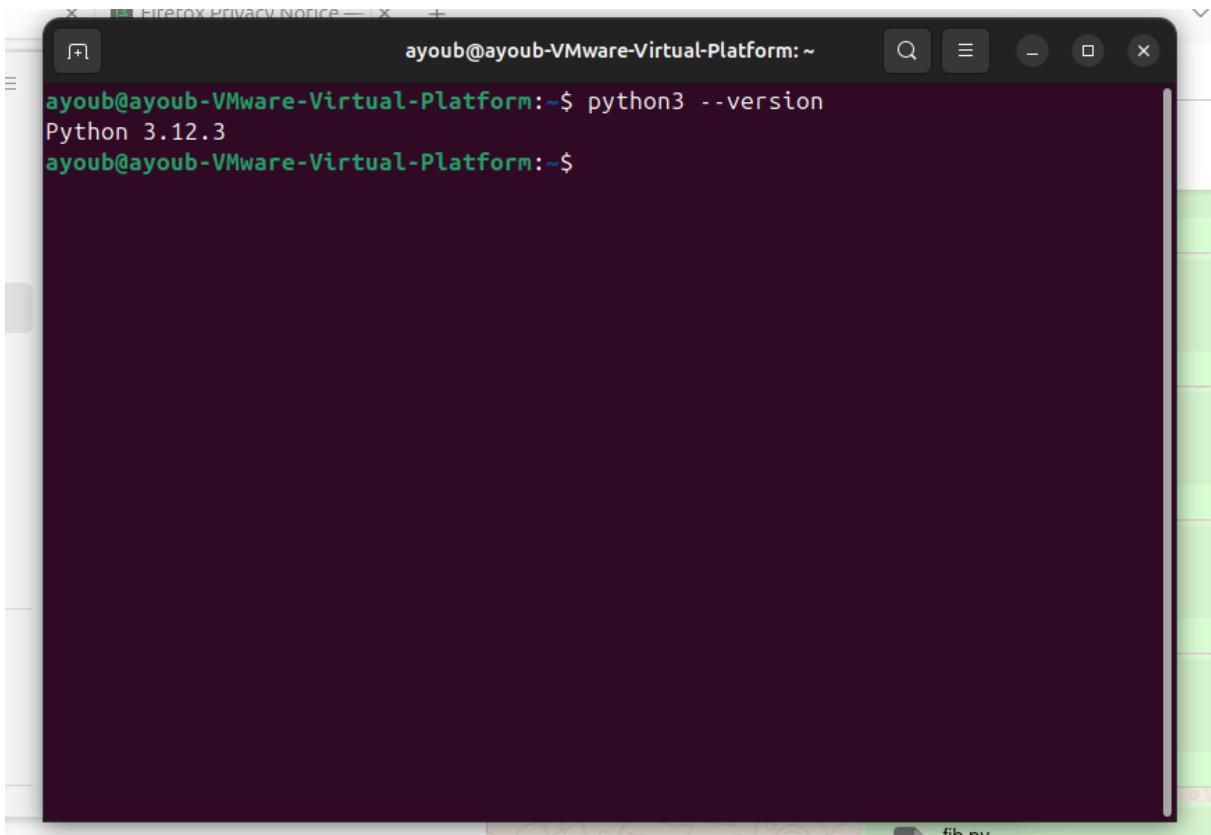
```
ayoub@ayoub-VMware-Virtual-Platform:~$ java --version
openjdk 21.0.9 2025-10-21
OpenJDK Runtime Environment (build 21.0.9+10-Ubuntu-124.04)
OpenJDK 64-Bit Server VM (build 21.0.9+10-Ubuntu-124.04, mixed mode, sharing)
ayoub@ayoub-VMware-Virtual-Platform:~$ S
```

gcc --version

```
ayoub@ayoub-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.

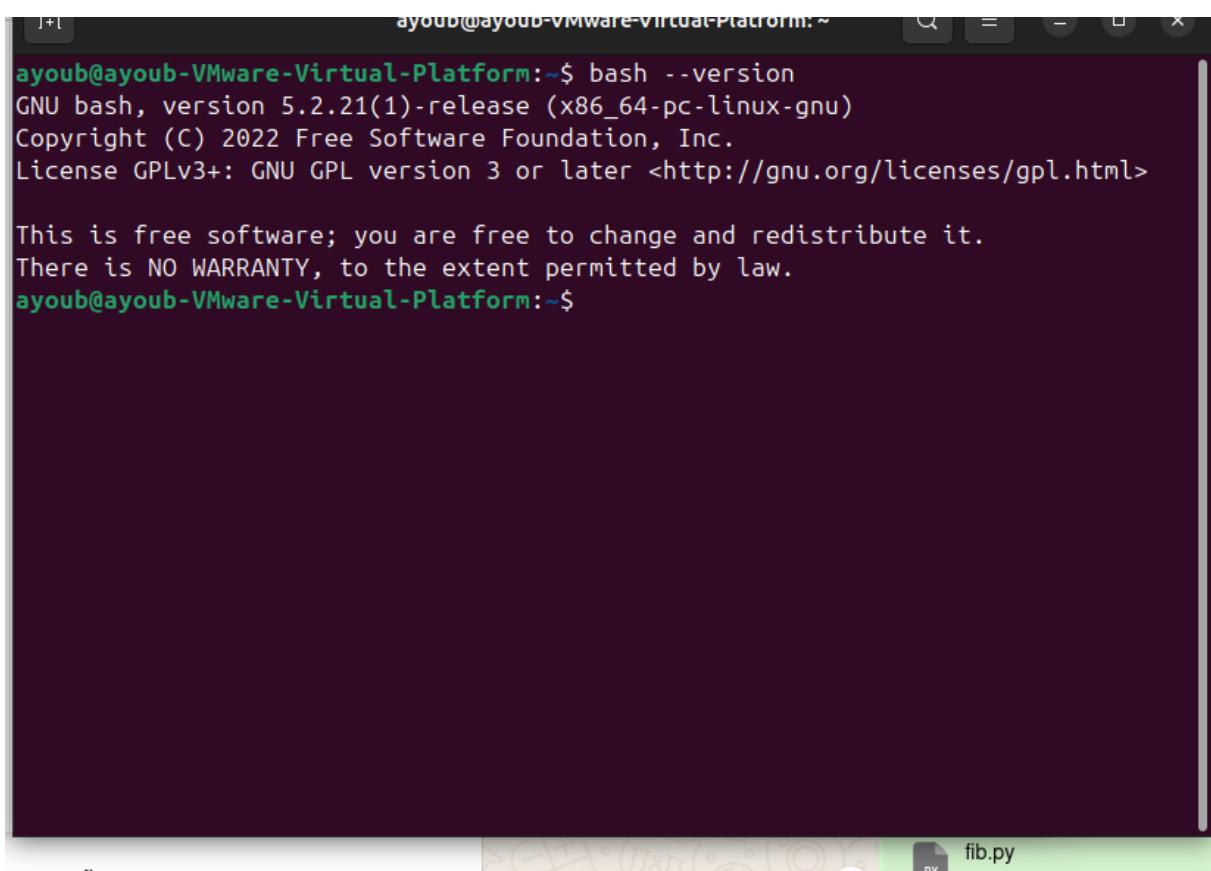
ayoub@ayoub-VMware-Virtual-Platform:~$
```

```
python3 --version
```



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

```
bash --version
```



```
ayoub@ayoub-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.
ayoub@ayoub-VMware-Virtual-Platform:~$
```


Assignment 4.3: Compile

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

Fib.c en Fibonacci.java

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

Fib.c, nadat die gecompiled is dan word die leesbaar voor de CPU

Which source code files are compiled to byte code?

Fibonacci.java

Which source code files are interpreted by an interpreter?

Fib.py (python interpreter) en 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?

Fib.c want hij heeft geen interpreter nodig om te runnen.

How do I run a Java program?

Eerst het programma compilen door javac Fibonacci.java, dat creeert Fibonacci.class en dat kun je runnen.

How do I run a Python program?

Die kun je gelijk runnen zonder te compilen.

How do I run a C program?

Door de file te compilen met GCC en dat maakt een executable file.

How do I run a Bash script?

Compilen met chmod en daarna gewoon runnen.

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

Ja, voor C komt een exe file, voor Java komt er een .class file dat je kan runnen, voor python en bash niet.

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?
De snelste bleek C te zijn met sys 0.001s echt snel snel

```
ayoub@ayoub-VMware-Virtual-Platform:~/Documents/code$ java Fibonacci
Fibonacci(18) = 2584
Execution time: 0.68 milliseconds
ayoub@ayoub-VMware-Virtual-Platform:~/Documents/code$
```

```
ayoub@ayoub-VMware-Virtual-Platform:~/Documents/code$ time python3 fib.py
Fibonacci(18) = 2584
Execution time: 0.99 milliseconds

real    0m0.041s
user    0m0.028s
sys     0m0.013s
ayoub@ayoub-VMware-Virtual-Platform:~/Documents/code$
```

A screenshot of a terminal window titled "ayoub@ayoub-VMware-Virtual-Platform: ~/Documents/code". The window shows the command "time ./fib.sh" being run. The output indicates that the Fibonacci number at index 18 is 2584, and the execution time is 16303 milliseconds. The terminal also displays the system performance metrics: real 0m16.318s, user 0m3.913s, and sys 0m13.782s.

```
ayoub@ayoub-VMware-Virtual-Platform:~/Documents/code$ time ./fib.sh
Fibonacci(18) = 2584
Excution time 16303 milliseconds

real    0m16.318s
user    0m3.913s
sys     0m13.782s
ayoub@ayoub-VMware-Virtual-Platform:~/Documents/code$
```

A screenshot of a terminal window titled "ayoub@ayoub-VMware-Virtual-Platform: ~/Documents/code". The window shows the command "time ./fib" being run. The output indicates that the Fibonacci number at index 18 is 2584, and the execution time is 0.14 milliseconds. The terminal also displays the system performance metrics: real 0m0.006s, user 0m0.005s, and sys 0m0.001s.

```
ayoub@ayoub-VMware-Virtual-Platform:~/Documents/code$ time ./fib
Fibonacci(18) = 2584
Execution time: 0.14 milliseconds

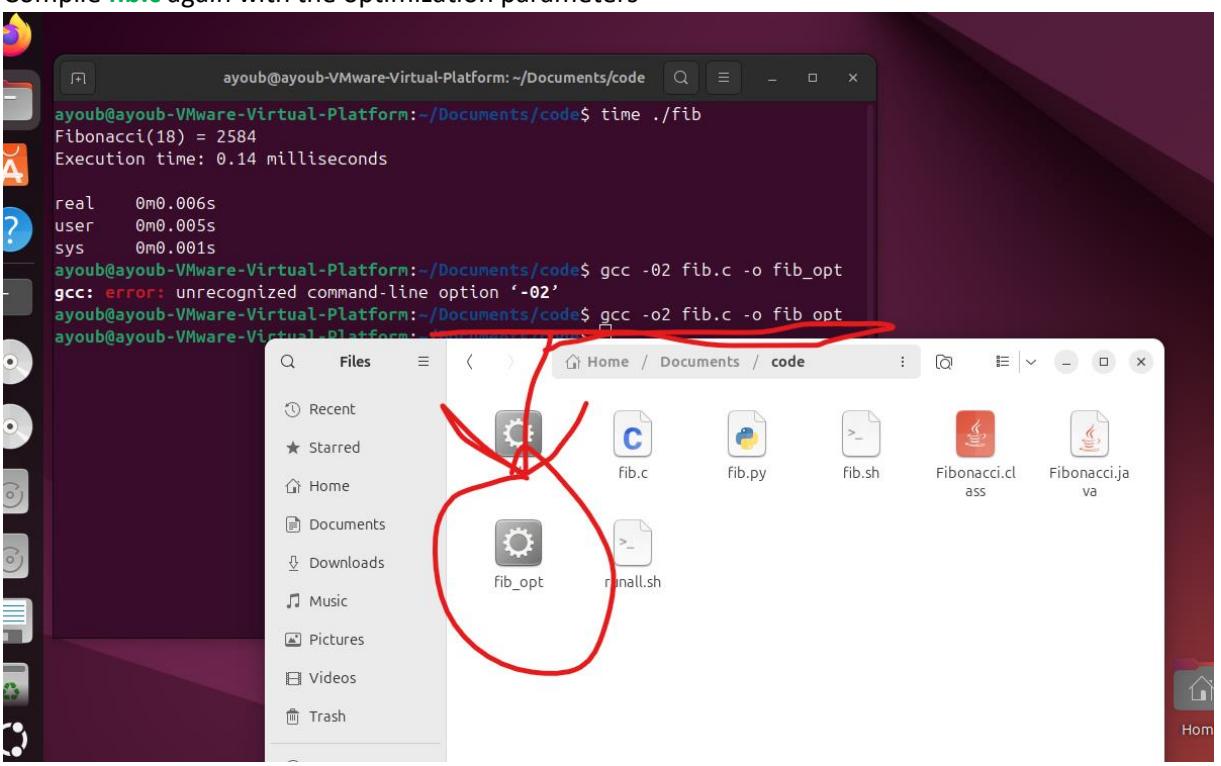
real    0m0.006s
user    0m0.005s
sys     0m0.001s
ayoub@ayoub-VMware-Virtual-Platform:~/Documents/code$
```

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.

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



The screenshot shows a Linux desktop environment with a terminal window and a file manager window.

In the terminal window (top), the user runs the command `time ./fib`. The output shows the calculation of Fibonacci(18) = 2584 and an execution time of 0.14 milliseconds. Then, the user runs `gcc -O2 fib.c -o fib_opt`, which results in an error message: "gcc: error: unrecognized command-line option '-O2'". Finally, the user runs `gcc -o2 fib.c -o fib_opt`, which succeeds.

In the file manager window (bottom), the user has navigated to the directory `~/Documents/code`. They have selected the file `fib_opt` and right-clicked it, opening a context menu. A red circle highlights the context menu icon on the left side of the menu.

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

The image shows a Linux desktop environment with a dark theme. Two terminal windows are open side-by-side, both titled "ayoub@ayoub-VMware-Virtual-Platform: ~/Documents/code\$".

The top terminal window displays the following output:

```
Fibonacci(18) = 2584
Execution time: 0.08 milliseconds

real    0m0.007s
user    0m0.005s
sys     0m0.002s
```

The bottom terminal window displays the following output:

```
Fibonacci(18) = 2584
Execution time: 0.08 milliseconds

real    0m0.007s
user    0m0.005s
sys     0m0.002s

ayoub@ayoub-VMware-Virtual-Platform:~/Documents/code$ time ./fib_opt
Fibonacci(18) = 2584
Execution time: 0.05 milliseconds

real    0m0.007s
user    0m0.003s
sys     0m0.004s
```

The desktop interface includes a vertical dock on the left containing icons for various applications like a browser, file manager, terminal, and system settings. Below the dock are icons for Pictures and Videos.

Zoals je kan zien runt die intotaal maar 0.03 miliseconden sneller dus in principe is die wel sneller ja.

- e) 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.

The screenshot shows a terminal window with a dark background. At the top, it displays the date and time: "Jan 8 23:52" and the user information: "ayoub@ayoub-VMware-Virtual-Platform: ~/Documents/code". The terminal window contains the following text:

```
Running C program:  
Fibonacci(19) = 4181  
Execution time: 0.08 milliseconds  
  
Running Java program:  
Fibonacci(19) = 4181  
Execution time: 0.90 milliseconds  
  
Running Python program:  
Fibonacci(19) = 4181  
Execution time: 1.60 milliseconds  
  
Running BASH Script  
Fibonacci(19) = 4181  
Execution time: 26436 milliseconds  
  
ayoub@ayoub-VMware-Virtual-Platform:~/Documents/code$
```

The terminal window has a standard Linux-style interface with icons on the left and a scroll bar on the right.

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 the OakSim debugger interface. The assembly code window displays the following code:

```
1 Main:  
2     mov r1, #2  
3     mov r2, #4  
4     mov r0, #1  
5  
6 Loop:  
7     mul r0, r0, r1  
8     sub r2, r2, #1  
9     cmp r2, #0  
10    beq End  
11    b Loop  
12  
13 End:  
14    b End  
15
```

The register window shows the following initial values:

Register	Value
R0	10
R1	2
R2	0
R3	0
R4	0
R5	0
R6	0
R7	0
R8	0
R9	0
R10	0

The memory dump window shows the memory starting at address 0x0000100000. The first few bytes are 02 10 40 E3 04 20 A0 E3 01 00 A0 E3 90 01 00 E0, followed by several FF FF EA bytes, and then more 00 00 00 00 bytes.

Ready? Save this file and export it as a pdf file with the name: **week4.pdf**