

# Template Week 4 – Software

Student number:

## Assignment 4.1: ARM assembly

Screenshot of working assembly code of factorial calculation:

The screenshot shows the OakSim software interface. On the left, there is a text editor containing ARM assembly code. The code calculates the factorial of 7, starting with r0 = 1, r2 = 7, and then looping until r2 = 1. On the right, there is a register dump window titled "Register Value". The registers R0 through R10 are listed with their current values: R0 = 78, R1 = 0, R2 = 0, R3 = 0, R4 = 0, R5 = 0, R6 = 0, R7 = 0, R8 = 0, R9 = 0, and R10 = 0.

```
1 Main:    mov    r0, #1
2          mov    r2, #5
3
4 Loop:    mul    r0, r2, r0
5          subs   r2, r2, #1
6          bne    Loop
7
8 End:    .
```

Register	Value
R0	78
R1	0
R2	0
R3	0
R4	0
R5	0
R6	0
R7	0
R8	0
R9	0
R10	0

## Assignment 4.2: Programming languages

Take screenshots that the following commands work:

javac –version

The screenshot shows a terminal window titled "Ubuntu 64-bit" running on a Linux desktop environment. The user "wessel" is at the prompt. The command "javac --version" is entered and executed, displaying the Java compiler version 21.0.9. The terminal window is part of a desktop environment with icons for the Dash, Home, Help, and others visible on the left.

```
wessel@helpdesk:~$ javac --version
javac 21.0.9
wessel@helpdesk:~$
```

java –version

```
wessel@helpdesk: ~$ javac --version
javac 21.0.9
wessel@helpdesk: ~$ java--version
java--version: command not found
wessel@helpdesk: ~$ 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)
```

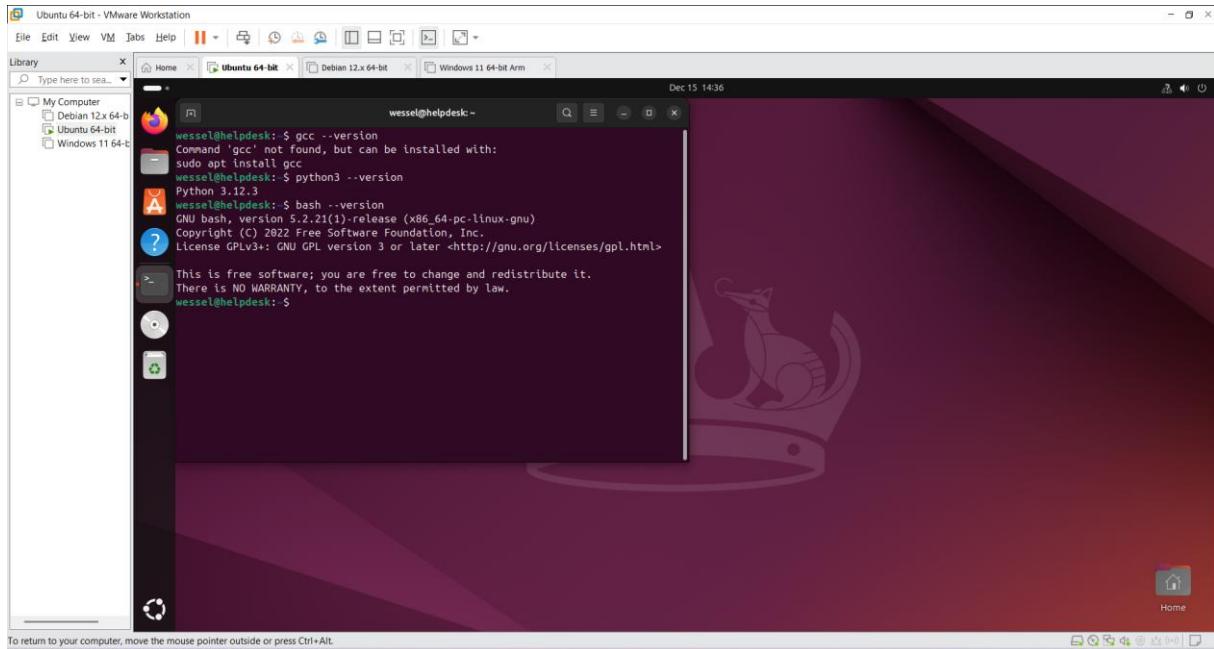
gcc –version

```
wessel@helpdesk: ~$ 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.
```

python3 –version

```
wessel@helpdesk: ~$ gcc --version
Command 'gcc' not found, but can be installed with:
sudo apt install gcc
wessel@helpdesk: ~$ python3 --version
Python 3.12.3
wessel@helpdesk: ~$
```

bash –version



### **Assignment 4.3: Compile**

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

Java en C

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

C C++, Assembly

Which source code files are compiled to byte code?

Java, C#

Which source code files are interpreted by an interpreter?

Py, Sh, Bash

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

C, omdat hij gelijk dan word gecompileerd naar machinecode

How do I run a Java program?

Om te compileren doe je javac hello.java

En dan uitvoeren doe je hello

How do I run a Python program?

Je kan hem direct uitvoeren met

Python3 hello.py

How do I run a C program?

Om te compileren doe je gcc hello.c -o hello

En dan uitvoeren doe je ./hello

How do I run a Bash script?

Om hem uitvoerbaar te maken doe je chmod +x hello.sh

En dan uitvoeren doe je ./hello.sh

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

Ja runall.sh

Take relevant screenshots of the following commands:

- Compile the source files where necessary
- Make them executable
- Run them

```
Ubuntu 22.04 LTS.0  
C:\Saxion\virtual machines\Ubuntu 64-bit\Ubuntu 64-bit.vmx  
Copyright © 2022 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.  
wessel@helpdesk:~$ 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.  
wessel@helpdesk:~$ cd code  
wessel@helpdesk:~/code$ ls  
fib.c Fibonacci.java fib.py fib.sh runall.sh  
wessel@helpdesk:~/code$
```

Fib.sh

```
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.  
wessel@helpdesk:~$ cd code  
wessel@helpdesk:~/code$ ls  
fib.c Fibonacci.java fib.py fib.sh runall.sh  
wessel@helpdesk:~/code$ sudo ./fib.sh  
Fibonacci(18) = 2584  
Execution time 3600 milliseconds  
wessel@helpdesk:~/code$
```

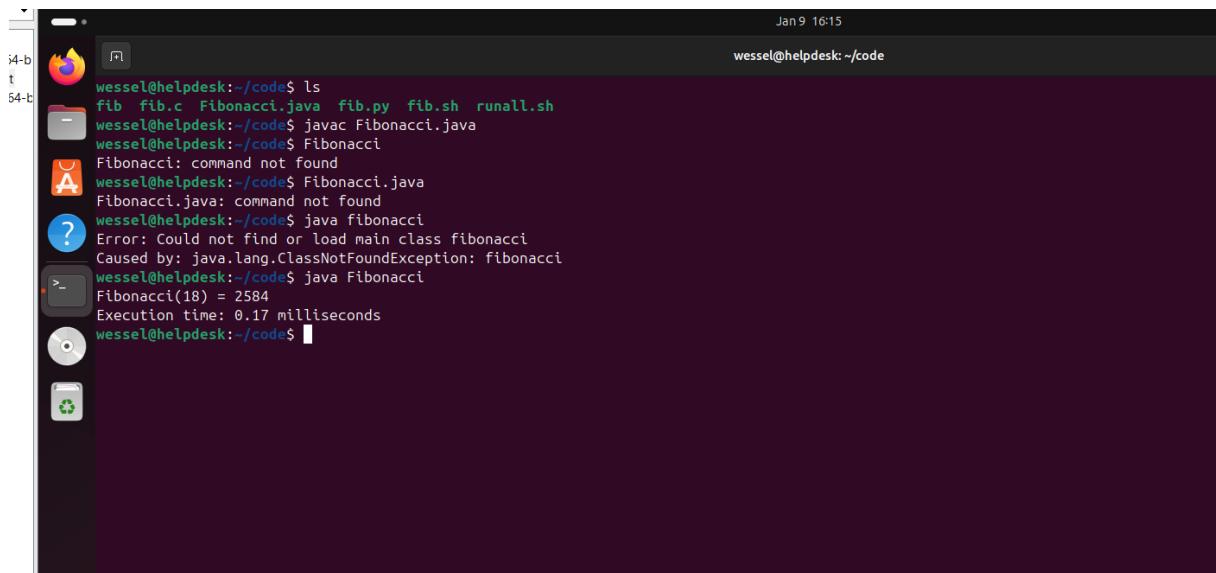
Python3

```
wessel@helpdesk:~/code$ python3 fib.py  
Fibonacci(18) = 2584  
Execution time 3600 milliseconds  
wessel@helpdesk:~/code$ python3 fib.py  
Fibonacci(18) = 2584  
Execution time: 0.20 milliseconds  
wessel@helpdesk:~/code$
```

C

```
Execution time 3600 milliseconds
wessel@helpdesk:~/code$ python3 fib.py
Fibonacci(18) = 2584
Execution time: 0.20 milliseconds
wessel@helpdesk:~/code$ gcc fib.c -o fib
wessel@helpdesk:~/code$ ./fib.c
./fib.c: line 5: syntax error near unexpected token `('
./fib.c: line 5: `int fibonacci(int n) {'
wessel@helpdesk:~/code$ ./fib
Fibonacci(18) = 2584
Execution time: 0.05 milliseconds
wessel@helpdesk:~/code$
```

Java



The screenshot shows a macOS desktop environment with a terminal window open. The terminal window has a dark background and displays the following command-line session:

```
wessel@helpdesk:~/code$ ls
fib fib.c Fibonacci.java fib.py fib.sh runall.sh
wessel@helpdesk:~/code$ javac Fibonacci.java
wessel@helpdesk:~/code$ Fibonacci
Fibonacci: command not found
wessel@helpdesk:~/code$ Fibonacci.java
Fibonacci.java: command not found
wessel@helpdesk:~/code$ java fibonacci
Error: Could not find or load main class fibonacci
Caused by: java.lang.ClassNotFoundException: fibonacci
wessel@helpdesk:~/code$ java Fibonacci
Fibonacci(18) = 2584
Execution time: 0.17 milliseconds
wessel@helpdesk:~/code$
```

- Which (compiled) source code file performs the calculation the fastest?

C omdat het direct naar machinecode wordt vertaald

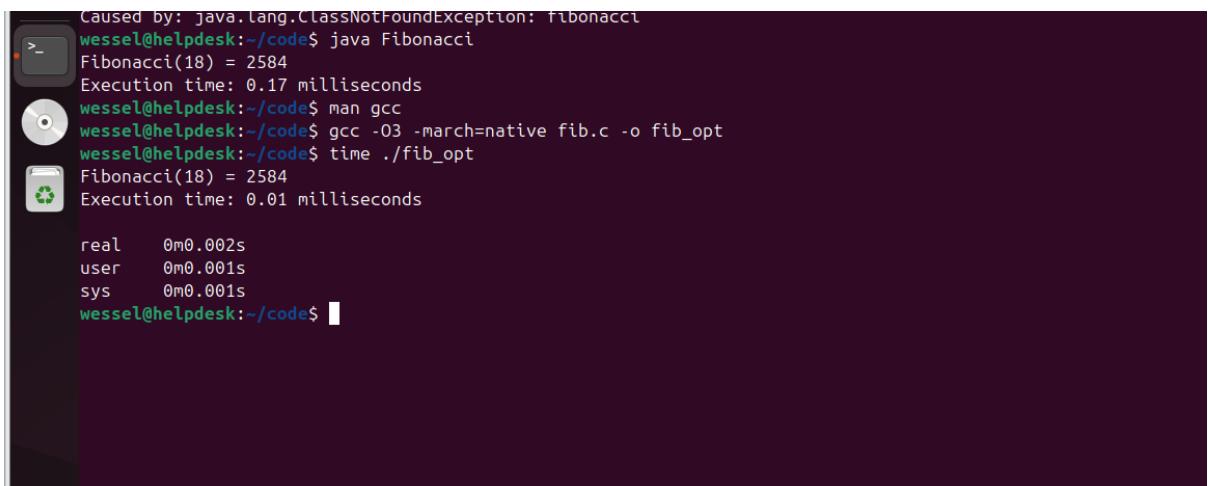
#### Assignment 4.4: Optimize

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.

O3 – march=native

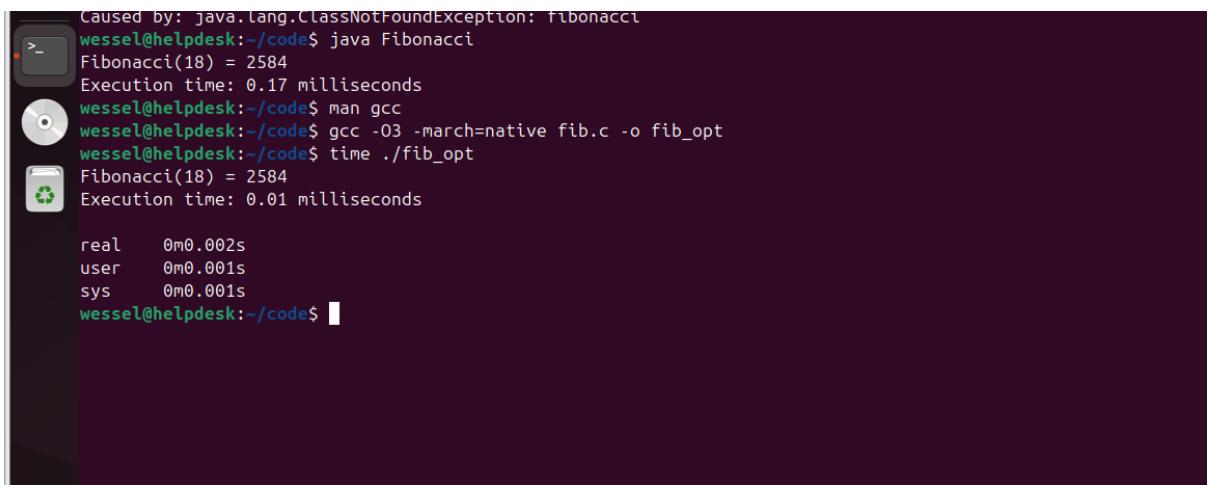
- b) Compile **fib.c** again with the optimization parameters



```
Caused by: java.lang.ClassNotFoundException: fibonacci
wessel@helpdesk:~/code$ java Fibonacci
Fibonacci(18) = 2584
Execution time: 0.17 milliseconds
wessel@helpdesk:~/code$ man gcc
wessel@helpdesk:~/code$ gcc -O3 -march=native fib.c -o fib_opt
wessel@helpdesk:~/code$ time ./fib_opt
Fibonacci(18) = 2584
Execution time: 0.01 milliseconds

real    0m0.002s
user    0m0.001s
sys     0m0.001s
wessel@helpdesk:~/code$
```

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



```
Caused by: java.lang.ClassNotFoundException: fibonacci
wessel@helpdesk:~/code$ java Fibonacci
Fibonacci(18) = 2584
Execution time: 0.17 milliseconds
wessel@helpdesk:~/code$ man gcc
wessel@helpdesk:~/code$ gcc -O3 -march=native fib.c -o fib_opt
wessel@helpdesk:~/code$ time ./fib_opt
Fibonacci(18) = 2584
Execution time: 0.01 milliseconds

real    0m0.002s
user    0m0.001s
sys     0m0.001s
wessel@helpdesk:~/code$
```

- d) 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.

```
-b
Running C program:
Fibonacci(19) = 4181
Execution time: 0.05 milliseconds

Running Java program:
Fibonacci(19) = 4181
Execution time: 0.20 milliseconds

Running Python program:
Fibonacci(19) = 4181
Execution time: 0.32 milliseconds

Running BASH Script
Fibonacci(19) = 4181
Excution time 5852 milliseconds

wessel@helpdesk:~/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 r0, #1
mov r1, #2
mov r2, #4
```

Loop:

```
cmp r2, #0
beq End
mul r0, r0, r1
```

```
sub r2, r2, #1
```

```
b 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 sections:

- Control Bar:** Open, Run (highlighted), 250, Step, Reset.
- Assembly View:** Shows the assembly code:

```
1 Main:
2   mov r0, #1
3   mov r1, #2
4   mov r2, #4
5
6 Loop:
7   cmp r2, #0
8   bne End
9   mul r0, r0, r1
10  sub r2, r2, #1
11  b Loop
12
13 End:
```
- Registers View:** Shows register 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
- Memory Dump View:** Shows memory dump starting at address 0x00010000.

Address	Value
0x00010000:	01 00 A0 E3 02 10 A0 E3 04 20 A0 E3 01 00 52 E3
0x00010010:	02 00 0A 90 01 E0 01 20 42 E2 F2 FF EA
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

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