

Template Week 4 – Software

Student number: 589530

Assignment 4.1: ARM assembly

Values of:

R0 = 1

R1 = 1

R2 = 0

R3 = f

Screenshot of working assembly code of factorial calculation:

```
1 Main:
2     mov r0, #4
3     mov r1, #1
4
5 Loop:
6     mul r1, r1, r0
7
8     sub r0, r0, #1
9
10    cmp r0, #0
11    BNE Loop
12
13 Exit:
14 |
```

Assignment 4.2: Programming languages

Take screenshots that the following commands work:

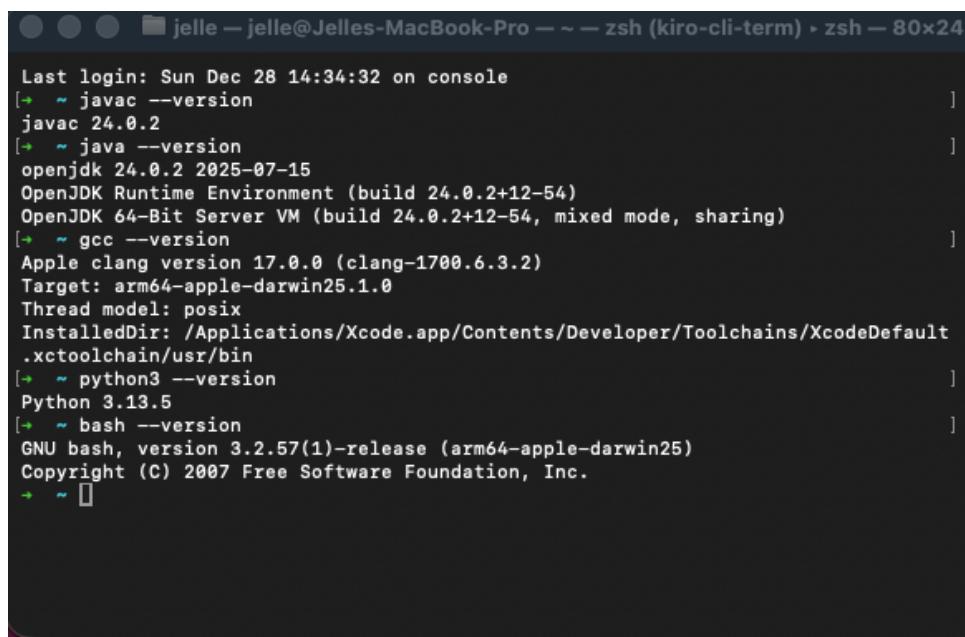
```
javac --version
```

```
java --version
```

```
gcc --version
```

```
python3 --version
```

```
bash --version
```



The screenshot shows a terminal window titled "jelle — jelle@Jelles-MacBook-Pro — ~ — zsh (kiro-cli-term) > zsh — 80x24". The window displays the output of several system and application version queries:

```
Last login: Sun Dec 28 14:34:32 on console
[~] ~ javac --version
javac 24.0.2
[~] ~ java --version
openjdk 24.0.2 2025-07-15
OpenJDK Runtime Environment (build 24.0.2+12-54)
OpenJDK 64-Bit Server VM (build 24.0.2+12-54, mixed mode, sharing)
[~] ~ gcc --version
Apple clang version 17.0.0 (clang-1700.6.3.2)
Target: arm64-apple-darwin25.1.0
Thread model: posix
InstalledDir: /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin
[~] ~ python3 --version
Python 3.13.5
[~] ~ bash --version
GNU bash, version 3.2.57(1)-release (arm64-apple-darwin25)
Copyright (C) 2007 Free Software Foundation, Inc.
[~] ~ [ ]
```

Assignment 4.3: Compile

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

Fibonacci.java en fib.c, moeten alle bij eerst compiled worden voordat je kan runnen, dit komt omdat .c en .java compiled talen zijn, de rest is interpreteert talen.

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

Alleen fib.c, de java bestand word compiled naar java byte code en uitgevoert door JVM. En niet de cpu.

Which source code files are compiled to byte code?

Fibonacci.java

Which source code files are interpreted by an interpreter?

Fib.sh en fib.py

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

Een compiled taal is sneller, dus zelfde code logica in .c zou sneller zijn dan in .py aangezien python een interpreter taal is.

How do I run a Java program?

Type “Javac Fibonacci.java” in de terminal

How do I run a Python program?

Type “Python fib.py” in de terminal

How do I run a C program?

gcc fib.c -o output_fib (to compile)

fib.c (to run it.)

How do I run a Bash script?

Chmod +x fib.sh (geef permissie om ‘t bestand te mogen uitvoeren.)

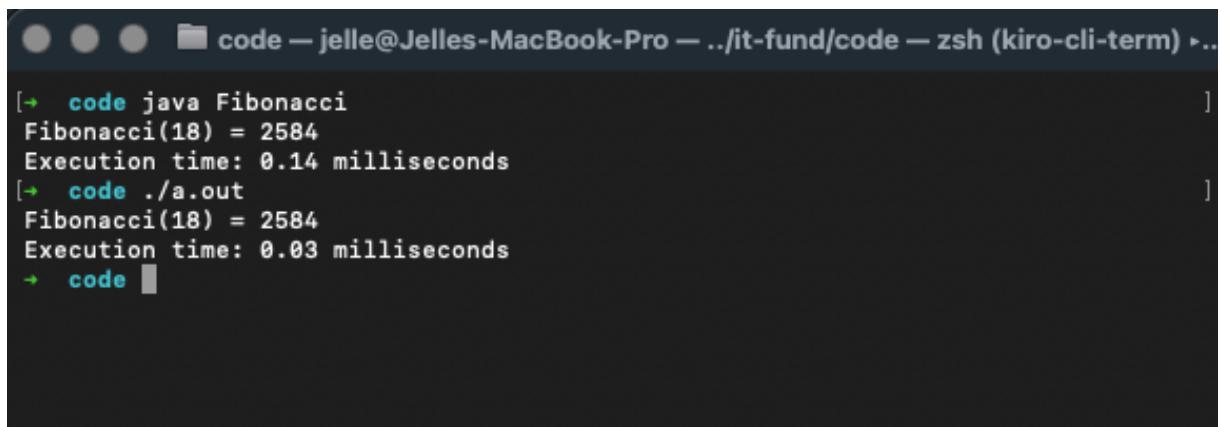
./fib.sh (om te runnen)

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

Ja, als je een taal compiled krijg je nog een bestand om hem uit te voeren, in windows een .exe

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?
 - Zoals in de screenshot staat is de java code sneller.



```
[+ code java Fibonacci
Fibonacci(18) = 2584
Execution time: 0.14 milliseconds
[+ code ./a.out
Fibonacci(18) = 2584
Execution time: 0.03 milliseconds
→ code ]
```

Name	Date Modified	Size	Kind
a.out	Today at 12:20	34 KB	Document
fib.c	9 Jun 2023 at 21:43	831 bytes	C Source
fib.py	9 Jun 2023 at 20:20	516 bytes	Python script
fib.sh	9 Jun 2023 at 21:40	668 bytes	shell script
Fibonacci.class	Today at 12:26	1 KB	Java class file
Fibonacci.java	9 Jun 2023 at 20:32	839 bytes	Java so...e code
runall.sh	9 Jun 2023 at 22:15	249 bytes	shell script

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.
 - a. Ik heb -O2 gebruik voor optimalisatie, dit geeft goede optimalisatie zonder gevaren.
- b) Compile **fib.c** again with the optimization parameters
gedaan
- c) Run the newly compiled program. Is it true that it now performs the calculation faster?

Ja, nu runt hij in 0.03 milliseconden.

```
[→ code gcc -O2 fib.c -o fib_O2
[→ code ./fib_O2
Fibonacci(18) = 2584
Execution time: 0.02 milliseconds
→ 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.

Gedaan.

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.

```
1 Main:  
2   mov r0, #4  
3   mov r1, #4  
4  
5  
6 Loop:  
7     add r2, r0, r1  
8     sub r3, r0, r1  
9  
10    cmp r2, r3  
11    BNE End  
12  
13    b Loop  
14  
15  
16 End:  
17
```

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](#)