

Template Week 4 – Software

Student number: 579864

Assignment 4.1: ARM assembly

Screenshot of working assembly code of factorial calculation:

The screenshot shows an ARM assembly simulator interface. At the top, there are buttons for 'Open', 'Run', '250', 'Step', and 'Reset'. Below these is a code editor with the following assembly code:

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

To the right of the code editor is a table showing the state of the registers:

Register	Value
R0	0
R1	78
R2	1
R3	0
R4	0
R5	0
R6	0
R7	0
R8	0
R9	0

Assignment 4.2: Programming languages

Take screenshots that the following commands work:

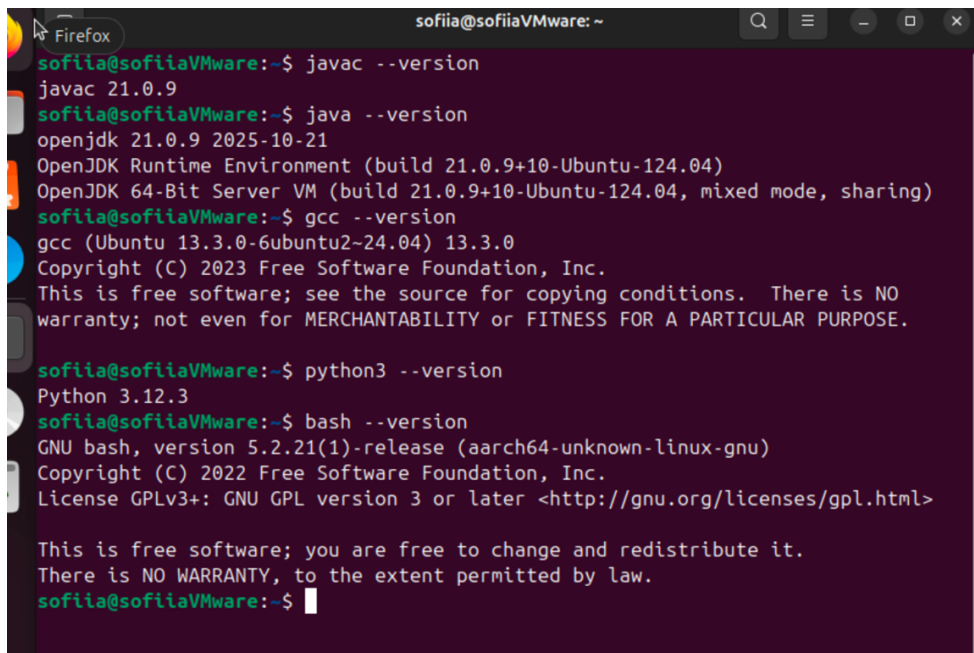
javac --version

java --version

gcc --version

python3 --version

bash --version

A screenshot of a terminal window titled 'sofiia@sofiiaVMware: ~'. The terminal shows the following commands and their outputs:
- `javac --version` outputs: `javac 21.0.9`
- `java --version` outputs: `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` outputs: `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` outputs: `Python 3.12.3`
- `bash --version` outputs: `GNU bash, version 5.2.21(1)-release (aarch64-unknown-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.`
The prompt `sofiia@sofiiaVMware:~$` is shown at the end.

Assignment 4.3: Compile

Which of the above files need to be compiled before you can run them? Fibonacci.java, fib.c

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

Which source code files are compiled to byte code? fib.c, Fibonacci.java

Which source code files are interpreted by an interpreter? fib.py

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

How do I run a Java program? Through a compiler that compiles java code to bytecode and translated by java virtual machine to machine code directly readable by CPU.

How do I run a Python program?

How do I run a C program? Compile the file with .c extension with a gcc -o ... command and run it.

How do I run a Bash script? it has to be made executable and then you can run it.

If I compile the above source code, will a new file be created? If so, which file? Yes, after fib.c compilation a new file fib was created and after java file compilation a file named Fibonacci.class was 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?

```
sofiia@sofiiaVMware:~/Downloads/code$ gcc -o fib fib.c
sofiia@sofiiaVMware:~/Downloads/code$ ls
fib fib.c Fibonacci.java fib.py fib.sh runall.sh
sofiia@sofiiaVMware:~/Downloads/code$ ./fib
Fibonacci(18) = 2584
Execution time: 0.01 milliseconds
sofiia@sofiiaVMware:~/Downloads/code$
```

```
sofiia@sofiiaVMware:~/Downloads/code$ javac Fibonacci.java
sofiia@sofiiaVMware:~/Downloads/code$ ls
fib fib.c Fibonacci.class Fibonacci.java fib.py fib.sh runall.sh
sofiia@sofiiaVMware:~/Downloads/code$ java Fibonacci
Fibonacci(18) = 2584
Execution time: 0.19 milliseconds
```

```
sofiia@sofiiaVMware:~/Downloads/code$ sudo chmod +x fib.sh
sofiia@sofiiaVMware:~/Downloads/code$ ls -l
total 40
-rwxrwxr-x 1 sofiia sofiia 70544 Dec  5 00:19 fib
-rw-rw-r-- 1 sofiia sofiia  831 Jun  9 2023 fib.c
-rw-rw-r-- 1 sofiia sofiia 1448 Dec  5 00:22 Fibonacci.class
-rw-rw-r-- 1 sofiia sofiia  839 Jun  9 2023 Fibonacci.java
-rw-rw-r-- 1 sofiia sofiia  516 Jun  9 2023 fib.py
-rwxrwxr-x 1 sofiia sofiia  668 Jun  9 2023 fib.sh
-rw-rw-r-- 1 sofiia sofiia  249 Jun  9 2023 runall.sh
```

```
sofiia@sofiiaVMware:~/Downloads/code$ ./fib.sh
Fibonacci(18) = 2584
Execution time 1913 milliseconds
```

```
sofia@sofiaVMware:~/Downloads/code$ python3 fib.py
Fibonacci(18) = 2584
Execution time: 0.42 milliseconds
```

```
Running C program:
Fibonacci(19) = 4181
Execution time: 0.03 milliseconds
```

```
Running Java program:
Fibonacci(19) = 4181
Execution time: 0.45 milliseconds
```

```
Running Python program:
Fibonacci(19) = 4181
Execution time: 0.39 milliseconds
```

```
Running BASH Script
Fibonacci(19) = 4181
Execution time 3135 milliseconds
```

```
sofia@sofiaVMware:~/Downloads/code$
```

The .c file runs the fastest.

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.
- Compile **fib.c** again with the optimization parameters
- Run the newly compiled program. Is it true that it now performs the calculation faster? Yes, it performs the calculation slightly faster (0.01 to 0.03 milliseconds) while the fib file sometimes takes around 0.01 to 0.06 milliseconds

```
sofia@sofiaVMware:~/Downloads/code$ gcc -O3 fib.c
sofia@sofiaVMware:~/Downloads/code$ ls
a.out  fib  fib.c  Fibonacci.class  Fibonacci.java  fib.py  fib.sh  runall.sh
sofia@sofiaVMware:~/Downloads/code$ ./a.out
Fibonacci(18) = 2584
Execution time: 0.01 milliseconds
```

- 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.

```
Running C program:
Fibonacci(19) = 4181
Execution time: 0.03 milliseconds

Running Java program:
Fibonacci(19) = 4181
Execution time: 0.17 milliseconds

Running Python program:
Fibonacci(19) = 4181
Execution time: 0.40 milliseconds

Running BASH Script
Fibonacci(19) = 4181
Execution time 3101 milliseconds
```

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.

OpenRun250StepReset

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

Register	Value
R0	10
R1	2
R2	1
R3	0
R4	0
R5	0
R6	0
R7	0
R8	0
R9	0

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