

Template Week 4 – Software

Student number: 589055

Assignment 4.1: ARM assembly

Screenshot of working assembly code of factorial calculation:

The screenshot shows an ARM assembly simulator interface. On the left, the assembly code is displayed:

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

In the center, a text box displays the name "Nick Stomps" and the student number "589055D". On the right, the Register window shows the current state of registers:

Register	Value
R0	0
R1	78
R2	2
R3	0
R4	0
R5	0
R6	0
R7	0
R8	0
R9	0

Below the registers, a memory dump shows the current state of memory, with the first few lines highlighted in red:

```
0x00010000: 05 20 A0 E3 02 10 81 E0 01 20 42 E2 91 02 01 E0 .....B.....
0x00010010: 01 52 E3 00 00 0A FA FE FE FA .....R.....
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 .....
0x000100C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0x000100D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0x000100E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0x000100F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0x00010100: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0x00010110: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0x00010120: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0x00010130: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
```

Assignment 4.2: Programming languages

Take screenshots that the following commands work:

`javac --version`

```
nick@helpdesk:~$ javac --version
Command 'javac' not found, but can be installed with:
sudo apt install default-jdk # version 2:1.17-75, or
sudo apt install openjdk-17-jdk-headless # version 17.0.16+8-us1-0ubuntu1~24.04.1
sudo apt install openjdk-21-jdk-headless # version 21.0.8+9-us1-0ubuntu1~24.04.1
sudo apt install ecj # version 3.32.0+eclipse4.26-2
sudo apt install openjdk-19-jdk-headless # version 19.0.2+7-4
sudo apt install openjdk-20-jdk-headless # version 20.0.2+9-1
sudo apt install openjdk-22-jdk-headless # version 22~22ea-1
sudo apt install openjdk-11-jdk-headless # version 11.0.28+6-1ubuntu1~24.04.1
sudo apt install openjdk-8-jdk-headless # version 8u462-ga-us1-0ubuntu2~24.04.2
sudo apt install openjdk-25-jdk-headless # version 25+36-1~24.04.2
```

`java --version`

```
nick@helpdesk:~$ java --version
Command 'java' not found, but can be installed with:
sudo apt install default-jre # version 2:1.17-75, or
sudo apt install openjdk-17-jre-headless # version 17.0.16+8-us1-0ubuntu1~24.04.1
sudo apt install openjdk-21-jre-headless # version 21.0.8+9-us1-0ubuntu1~24.04.1
sudo apt install openjdk-19-jre-headless # version 19.0.2+7-4
sudo apt install openjdk-20-jre-headless # version 20.0.2+9-1
sudo apt install openjdk-22-jre-headless # version 22~22ea-1
sudo apt install openjdk-11-jre-headless # version 11.0.28+6-1ubuntu1~24.04.1
sudo apt install openjdk-8-jre-headless # version 8u462-ga-us1-0ubuntu2~24.04.2
sudo apt install openjdk-25-jre-headless # version 25+36-1~24.04.2
```

`gcc --version`

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

`python3 --version`

```
nick@helpdesk:~$ python3 --version
Python 3.12.3
```

`bash --version`

```
nick@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.
```

Assignment 4.3: Compile

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

Fibonacci.java en 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?

Fibonacci.java

Which source code files are interpreted by an interpreter?

Fib.py en fib.sh

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?

Eerst `javac myProject.java` doen, daarna `java myProject`

How do I run a Python program?

Met de command `python3 myProject.py`

How do I run a C program?

Eerst `gcc myProject.c -o fib`, daarna `./myProject`

How do I run a Bash script?

Eerst `chmod +x myProject.sh`, daarna `./myProject.sh`

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

Ja, de file `Fibonacci.class` wordt aangemaakt en de executable voor C

Take relevant screenshots of the following commands:

- Compile the source files where necessary

```
nick@helpdesk:~/Downloads/code$ ls -a
.  ..  fib  fib.c  Fibonacci.class  Fibonacci.java  fib.py  fib.sh  runall.sh
nick@helpdesk:~/Downloads/code$
```

- Make them executable

```
nick@helpdesk:~/Downloads/code$ ls -a
.  ..  fib  fib.c  Fibonacci.class  Fibonacci.java  fib.py  fib.sh  runall.sh
nick@helpdesk:~/Downloads/code$
```

- Run them

```
Running C program:
Fibonacci(19) = 4181
Execution time: 0.02 milliseconds
```

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

```
Running Python program:
Fibonacci(19) = 4181
Execution time: 0.34 milliseconds
```

```
Running BASH Script
Fibonacci(19) = 4181
Execution time 5651 milliseconds
```

```
nick@helpdesk:~/Downloads/code$
```

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

De fib.c file

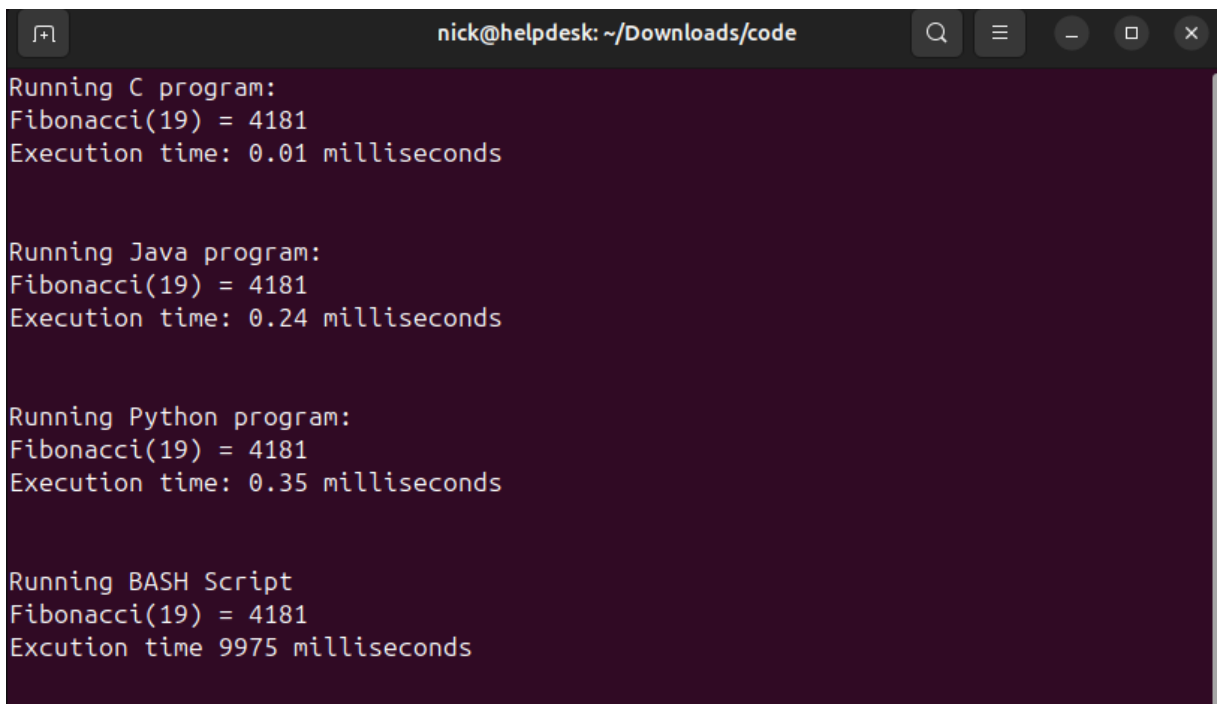
Assignment 4.4: Optimize

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.

Met -O3 kan je veel optimizations laten runnen

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



```
nick@helpdesk: ~/Downloads/code
Running C program:
Fibonacci(19) = 4181
Execution time: 0.01 milliseconds

Running Java program:
Fibonacci(19) = 4181
Execution time: 0.24 milliseconds

Running Python program:
Fibonacci(19) = 4181
Execution time: 0.35 milliseconds

Running BASH Script
Fibonacci(19) = 4181
Execution time 9975 milliseconds
```

- Run the newly compiled program. Is it true that it now performs the calculation faster?
- Hij runt het C programma sneller
- 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.

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
```

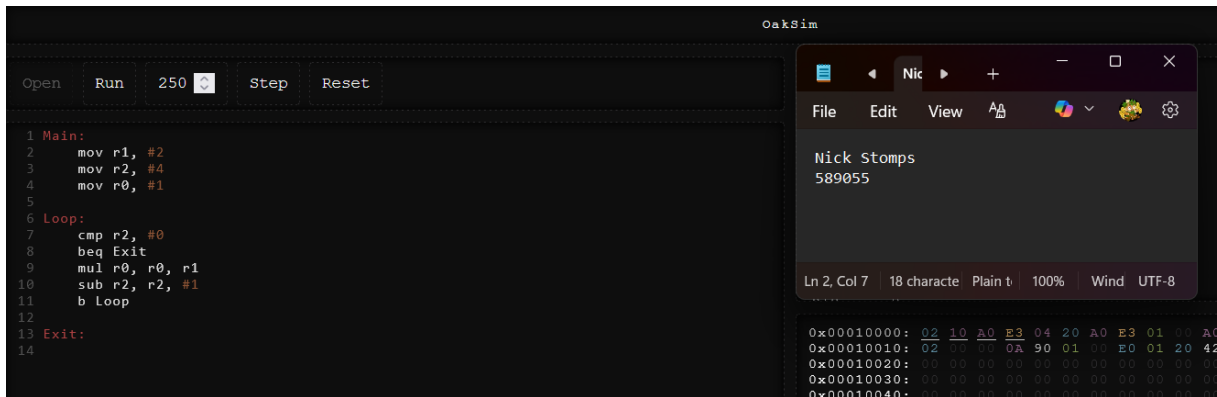
```
mov r2, #4
```

Loop:

End:

Complete the code. See the PowerPoint slides of week 4.

Screenshot of the completed code here.



The screenshot shows the OakSim assembly simulator interface. The main window displays the following assembly code:

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

On the right side, there is a window titled "Nick Stomps" showing the output "589055". Below this, a memory dump is visible, showing hexadecimal addresses and their corresponding values in hexadecimal and ASCII.

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