

# Template Week 4 – Software

Student number: 571755

## Assignment 4.1: ARM assembly

Screenshot of working assembly code of factorial calculation:

## Assignment 4.2: Programming languages

Take screenshots that the following commands work:

`javac --version`

`java --version`

`gcc --version`

`python3 --version`

`bash --version`

### **Assignment 4.3: Compile**

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

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

Which source code files are compiled to byte code?

Which source code files are interpreted by an interpreter?

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

How do I run a Java program?

How do I run a Python program?

How do I run a C program?

How do I run a Bash script?

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

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?

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

#### Bonus point assignment – week 4

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.

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

Main:

```
mov r0, #1    // Initialize r0 as 1 (start with 1, the identity for multiplication)
mov r1, #2    // Base = 2
mov r2, #4    // Exponent = 4
```

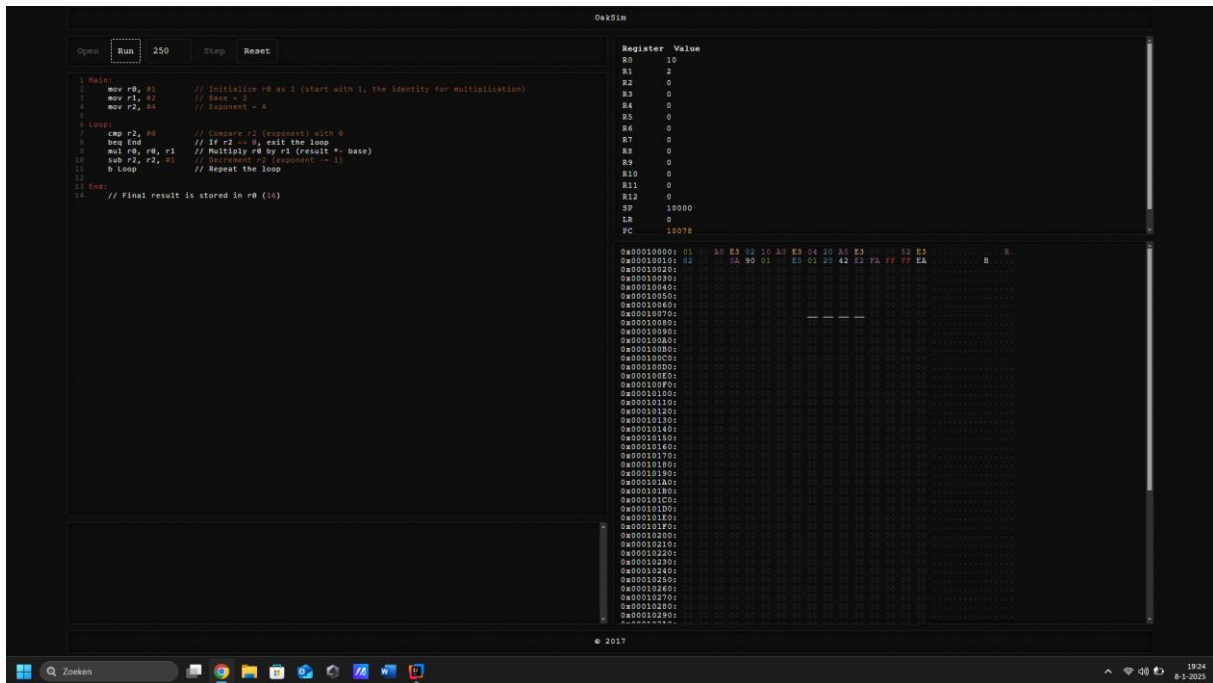
Loop:

```
cmp r2, #0    // Compare r2 (exponent) with 0
beq End      // If r2 == 0, exit the loop
mul r0, r0, r1 // Multiply r0 by r1 (result *= base)
sub r2, r2, #1 // Decrement r2 (exponent -= 1)
b Loop       // Repeat the loop
```

End:

```
// Result in r0
```

Screenshot of the completed code here.



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