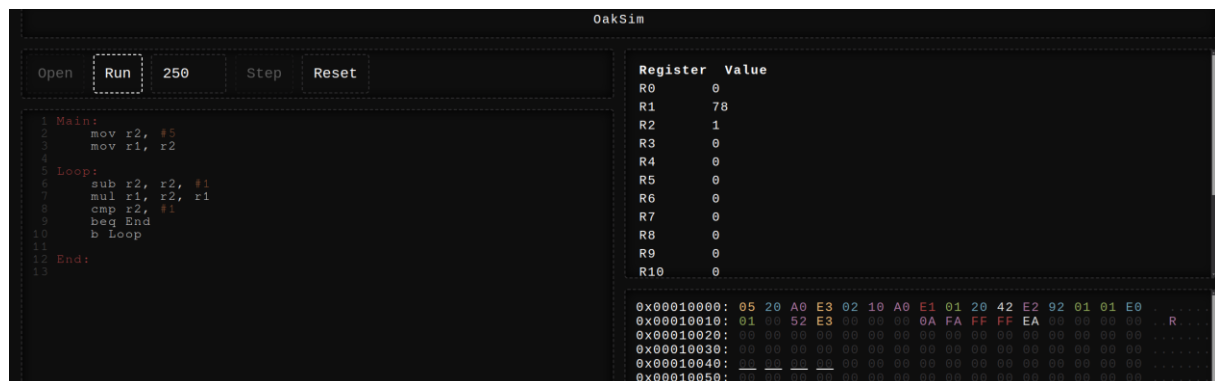


Template Week 4 – Software

Student number: 563064

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`

```
~ # javac --version
javac 21.0.5
~ # java --version
openjdk 21.0.5 2024-10-15
OpenJDK Runtime Environment (build 21.0.5+11)
OpenJDK 64-Bit Server VM (build 21.0.5+11, mixed mode, sharing)
~ # gcc --version
gcc (GCC) 14.2.1 20240910
Copyright (C) 2024 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
Python 3.12.7
~ # bash --version
GNU bash, version 5.2.37(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?

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.

Main:

```
mov r1, #2
```

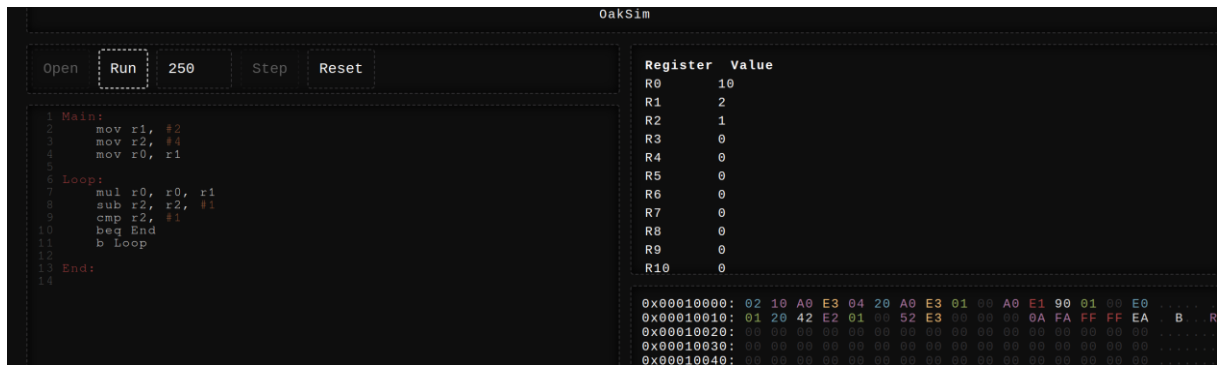
```
mov r2, #4
```

Loop:

End:

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

Screenshot of the completed code [here](#).



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