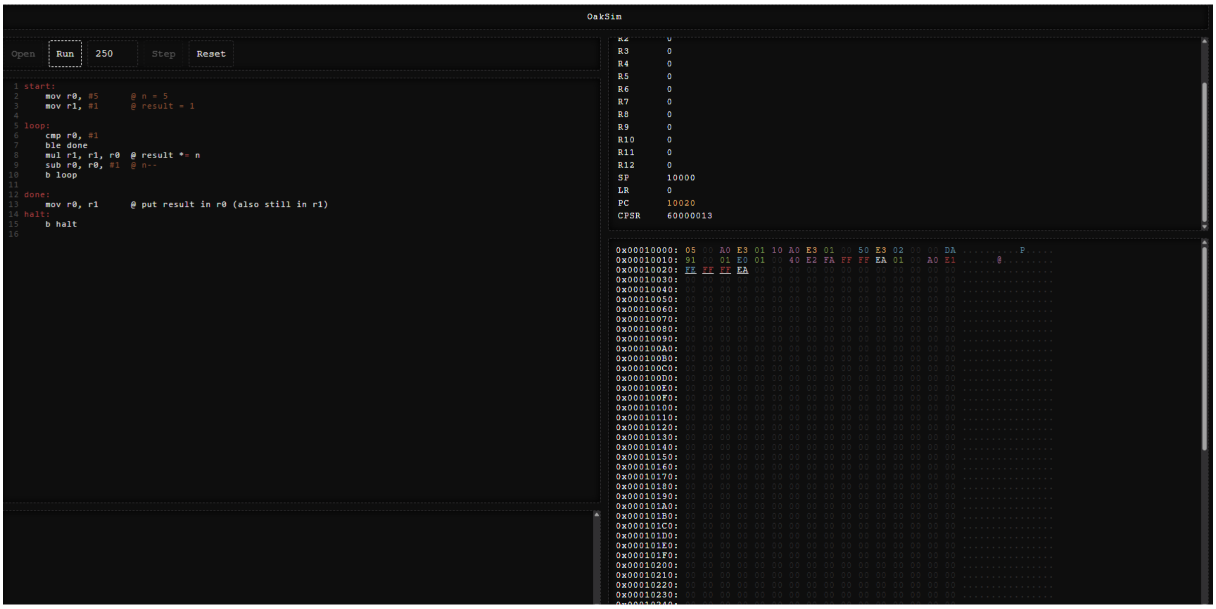
Template Week 4 – Software

Student number: 593201

Assignment 4.1: ARM assembly

Screenshot of working assembly code of factorial calculation:

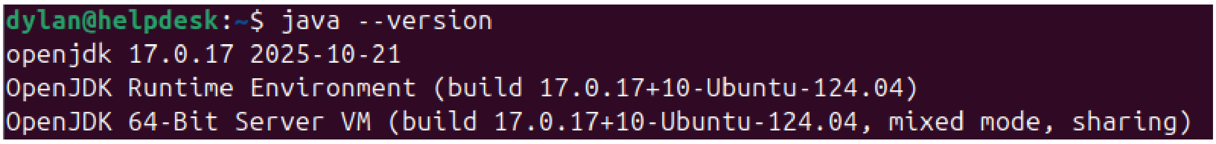


Assignment 4.2: Programming languages

Take screenshots that the following commands work:

javac –version

java –version



gcc –version

```
dylan@helpdesk:~$ gcc --version
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

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

bash --version

```
dylan@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?

- Java (.java): Yes, compile first (with javac) before you run it (with java).
- C (.c): Yes, compile first (with gcc) before you can run it.
- Python (.py): No, typically run directly with the interpreter.
- Bash (.sh): No, run directly with the shell.

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

- **C (.c)** → compiled by gcc into **native machine code** (a binary executable).

Which source code files are compiled to byte code?

- **Java (.java)** → compiled by javac into **bytecode** (.class), run by the JVM.

Which source code files are interpreted by an interpreter?

- P**ython (.py)** → interpreted by python3 (often compiled to bytecode internally, but you run it via the interpreter).
- **Bash (.sh)** → interpreted by bash.

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

Usually: C (native machine code) is fastest.
Then typically Java (JIT-compiled by the JVM) can be close.
Then Python, then Bash is usually slowest for computation-heavy work.

How do I run a Java program?

- javac Program.java
- java Program

How do I run a Python program?

- python3 program.py

How do I run a C program?

- gcc program.c -o program
- ./program

How do I run a Bash script?

- chmod +x script.sh
- ./script.sh

Or

- bash script.sh

If I compile the above source code, Java: yes → creates Program.class (and possibly multiple .class files)

- **Java**: yes → creates **Program.class** (and possibly multiple .class files)
- **C**: yes → creates an **executable binary** (e.g. program) and optionally intermediate .o files
- **Python**: not usually by you, but it may create **__pycache__/program.cpython-*.pyc**
- **Bash**: no compiled output file created (unless you explicitly generate something yourself)

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?

```
Running C program:
Fibonacci(19) = 4181
Execution time: 0.23 milliseconds


Running Java program:
Fibonacci(19) = 4181
Execution time: 0.47 milliseconds


Running Python program:
Fibonacci(19) = 4181
Execution time: 0.92 milliseconds


Running BASH Script
Fibonacci(19) = 4181
Excution time 13949 milliseconds


dylan@helpdesk:~/Desktop/code$
```

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.

```
Running C program:
Fibonacci(19) = 4181
Execution time: 0.000 milliseconds


Running Java program:
Fibonacci(19) = 4181
Execution time: 0.56 milliseconds


Running Python program:
Fibonacci(19) = 4181
Execution time: 0.95 milliseconds


Running BASH Script
Fibonacci(19) = 4181
Excution time 15689 milliseconds


dylan@helpdesk:~/Desktop/code$ ^C
dylan@helpdesk:~/Desktop/code$ ./runall.sh
```

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.

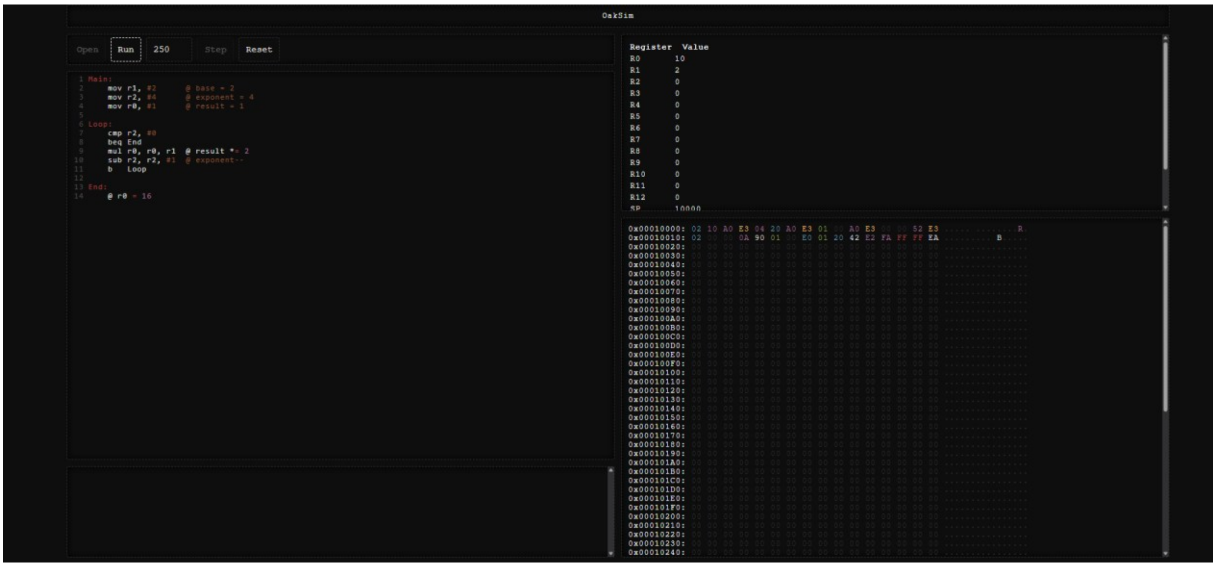
```
Main:
    mov r1, #2      @ base = 2
    mov r2, #4      @ exponent = 4
    mov r0, #1      @ result = 1

Loop:
    cmp r2, #0
    beq End
    mul r0, r0, r1  @ result *= 2
    sub r2, r2, #1  @ exponent--
    b   Loop

End:
    @ r0 = 16
```

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**