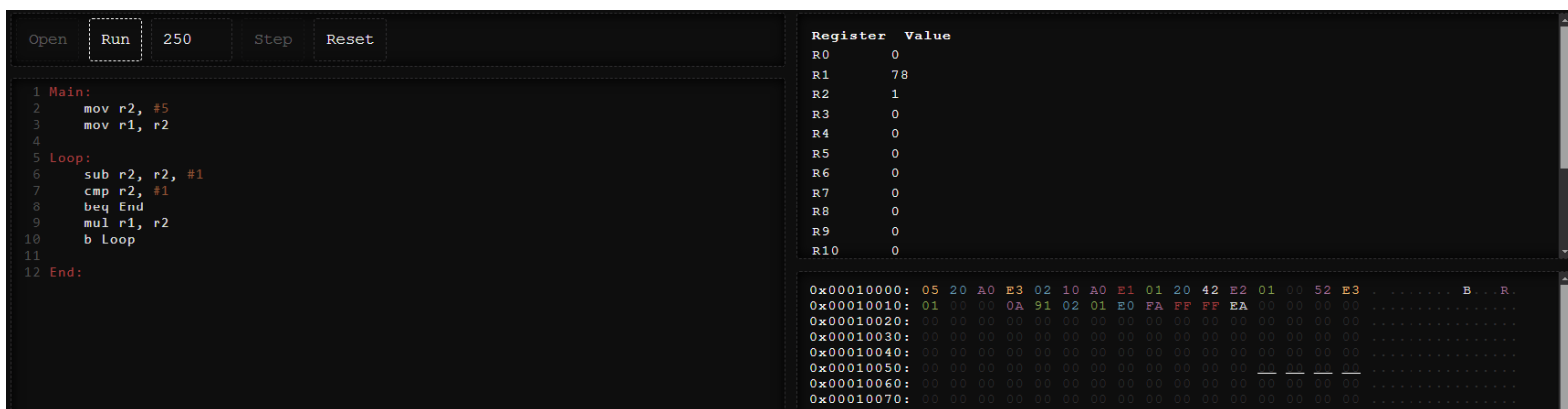


Template Week 4 – Software

Student number: 573512

Assignment 4.1: ARM assembly

Screenshot of working assembly code of factorial calculation:



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

Register	Value
R0	0
R1	78
R2	1
R3	0
R4	0
R5	0
R6	0
R7	0
R8	0
R9	0
R10	0

Memory dump (hex):

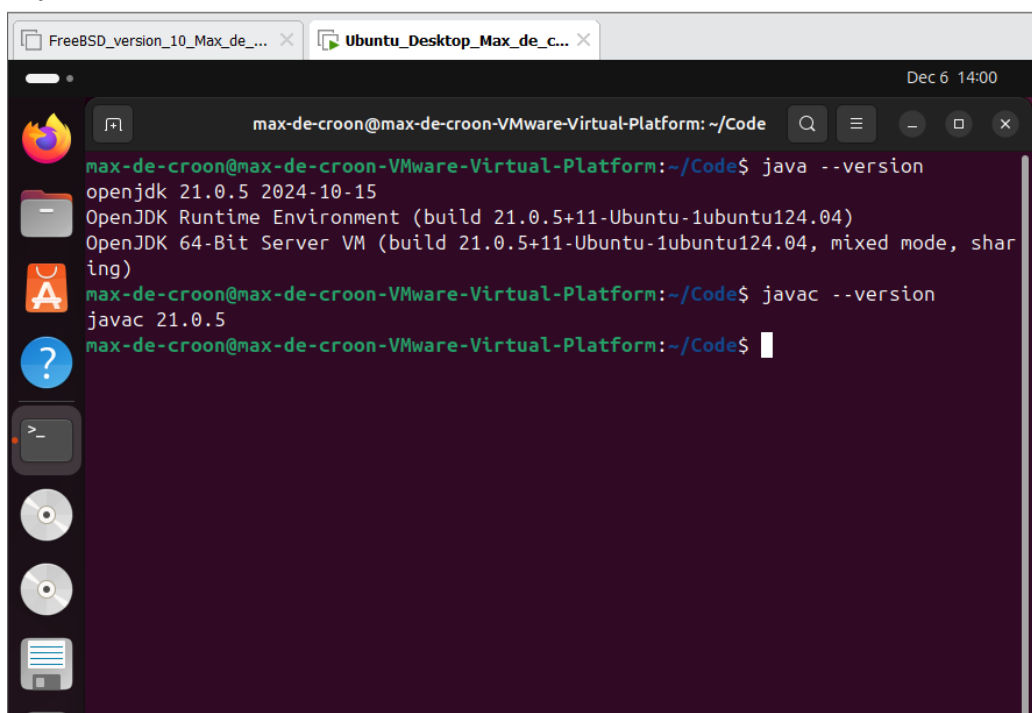
```
0x00010000: 05 20 A0 E3 02 10 A0 E1 01 20 42 E2 01 00 52 E3 ... B...R...
0x00010010: 01 00 00 0A 91 02 01 E0 FA FF FF EA 00 00 00 00 ...
0x00010020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...
0x00010030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...
0x00010040: 00 00 00 00 00 00 00 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 ...
```

Assignment 4.2: Programming languages

Take screenshots that the following commands work:

javac --version

java --version



```
max-de-croon@max-de-croon-VMware-Virtual-Platform: ~/Code
max-de-croon@max-de-croon-VMware-Virtual-Platform:~/Code$ java --version
openjdk 21.0.5 2024-10-15
OpenJDK Runtime Environment (build 21.0.5+11-Ubuntu-1ubuntu124.04)
OpenJDK 64-Bit Server VM (build 21.0.5+11-Ubuntu-1ubuntu124.04, mixed mode, sharing)
max-de-croon@max-de-croon-VMware-Virtual-Platform:~/Code$ javac --version
javac 21.0.5
max-de-croon@max-de-croon-VMware-Virtual-Platform:~/Code$
```

Java en Javac --version

gcc --version

```
max-de-croon@max-de-croon-VMware-Virtual-Platform:~/Code$ gcc --version
gcc (Ubuntu 13.2.0-23ubuntu4) 13.2.0
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.

max-de-croon@max-de-croon-VMware-Virtual-Platform:~/Code$
```

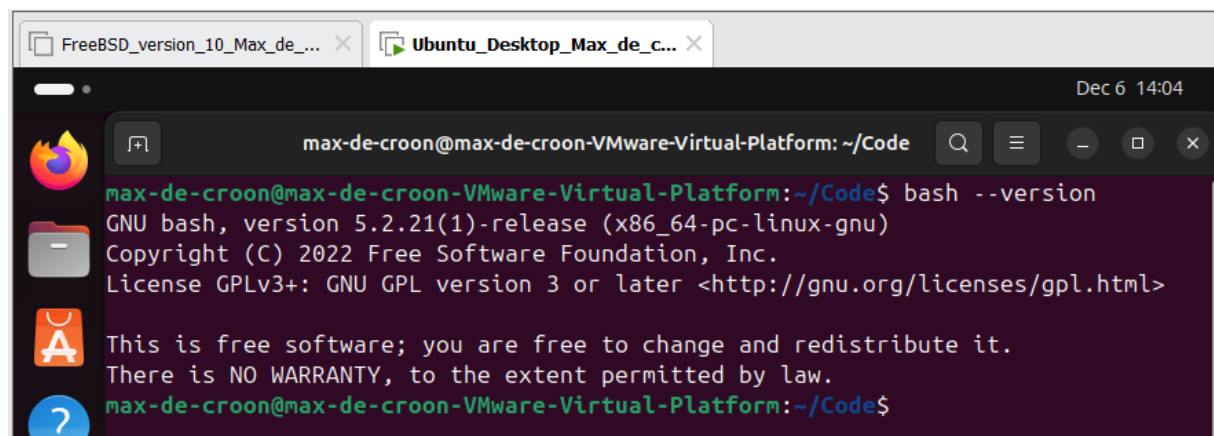
gcc --version

python3 --version

```
max-de-croon@max-de-croon-VMware-Virtual-Platform:~/Code$ python3 --version
Python 3.12.3
max-de-croon@max-de-croon-VMware-Virtual-Platform:~/Code$
```

Python3 --version

bash --version

A screenshot of a terminal window titled 'max-de-croon@max-de-croon-VMware-Virtual-Platform: ~/Code'. The terminal shows the command 'bash --version' and its output: '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.', and 'There is NO WARRANTY, to the extent permitted by law.' The prompt 'max-de-croon@max-de-croon-VMware-Virtual-Platform:~/Code\$' is visible at the bottom. The terminal window is part of a desktop environment with other windows visible in the background, including one titled 'FreeBSD_version_10_Max_de_...' and another titled 'Ubuntu_Desktop_Max_de_c...'. The system clock in the top right corner shows 'Dec 6 14:04'.

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.

Main:

```
mov r1, #2
mov r2, #4
mov r0, #1
```

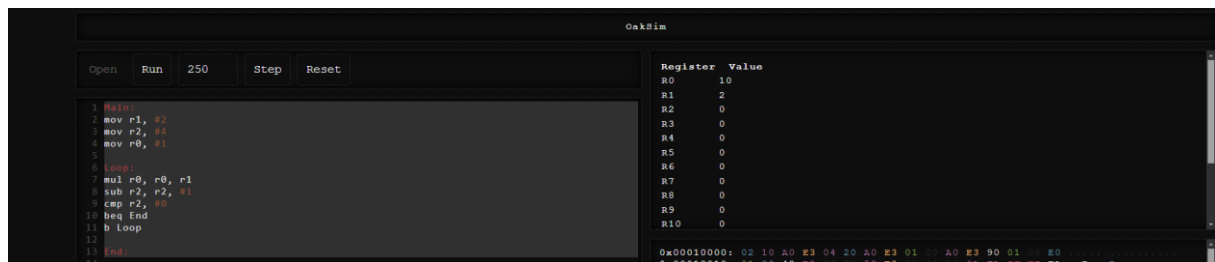
Loop:

```
mul r0, r0, r1
sub r2, r2, #1
cmp r2, #0
beq End
b 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](#)