

Template Week 4 – Software

Student number: 486707

Assignment 4.1: ARM assembly

Screenshot of working assembly code of factorial calculation:

The screenshot shows the OakSim software interface. At the top, there are buttons for 'Open', 'Run' (which is highlighted), '250', 'Step', and 'Reset'. Below these are the assembly instructions and register values.

Assembly Code:

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

Registers:

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

Memory Dump:

0x00010000:	05 20 A0 E3 02 10 A0 E1 01 20
0x00010010:	01 00 52 E3 00 00 00 0A FA FF
0x00010020:	00 00 00 00 00 00 00 00 00 00 00
0x00010030:	00 00 00 00 00 00 00 00 00 00 00
0x00010040:	00 00 00 00 00 00 00 00 00 00 00
0x00010050:	00 00 00 00 00 00 00 00 00 00 00
0x00010060:	00 00 00 00 00 00 00 00 00 00 00
0x00010070:	00 00 00 00 00 00 00 00 00 00 00
0x00010080:	00 00 00 00 00 00 00 00 00 00 00
0x00010090:	00 00 00 00 00 00 00 00 00 00 00
0x000100A0:	00 00 00 00 00 00 00 00 00 00 00
0x000100B0:	00 00 00 00 00 00 00 00 00 00 00
0x000100C0:	00 00 00 00 00 00 00 00 00 00 00
0x000100D0:	00 00 00 00 00 00 00 00 00 00 00
0x000100E0:	00 00 00 00 00 00 00 00 00 00 00
0x000100F0:	00 00 00 00 00 00 00 00 00 00 00
0x00010100:	00 00 00 00 00 00 00 00 00 00 00
0x00010110:	00 00 00 00 00 00 00 00 00 00 00
0x00010120:	00 00 00 00 00 00 00 00 00 00 00
0x00010130:	00 00 00 00 00 00 00 00 00 00 00
0x00010140:	00 00 00 00 00 00 00 00 00 00 00
0x00010150:	00 00 00 00 00 00 00 00 00 00 00
0x00010160:	00 00 00 00 00 00 00 00 00 00 00
0x00010170:	00 00 00 00 00 00 00 00 00 00 00
0x00010180:	00 00 00 00 00 00 00 00 00 00 00

Assignment 4.2: Programming languages

Take screenshots that the following commands work:

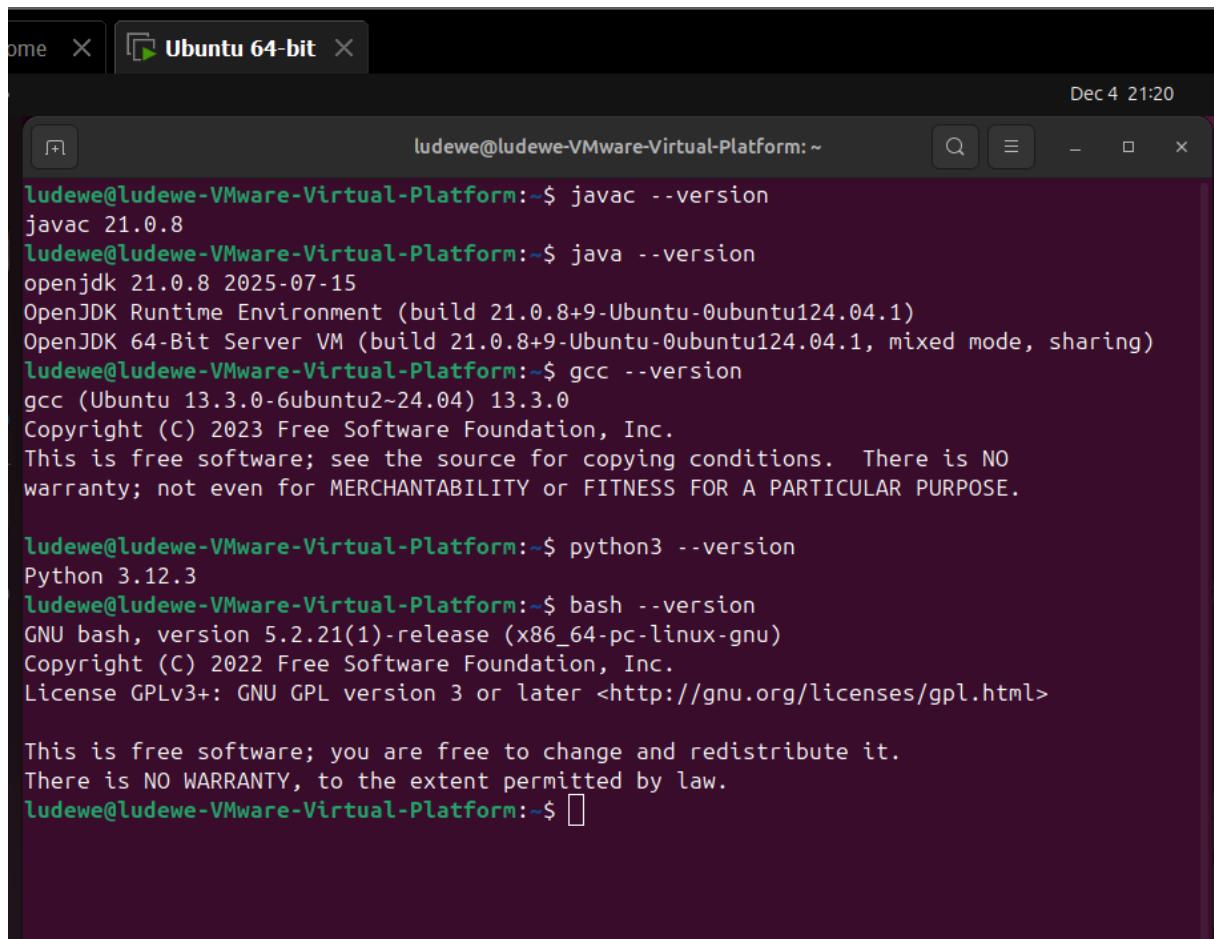
```
javac --version
```

```
java --version
```

```
gcc --version
```

```
python3 --version
```

```
bash --version
```



The screenshot shows a terminal window titled "Ubuntu 64-bit" running on an Ubuntu system. The terminal window has a dark background and light-colored text. It displays the following command-line session:

```
ludewe@ludewe-Virtual-Platform:~$ javac --version
javac 21.0.8
ludewe@ludewe-Virtual-Platform:~$ java --version
openjdk 21.0.8 2025-07-15
OpenJDK Runtime Environment (build 21.0.8+9-Ubuntu-0ubuntu124.04.1)
OpenJDK 64-Bit Server VM (build 21.0.8+9-Ubuntu-0ubuntu124.04.1, mixed mode, sharing)
ludewe@ludewe-Virtual-Platform:~$ 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.

ludewe@ludewe-Virtual-Platform:~$ python3 --version
Python 3.12.3
ludewe@ludewe-Virtual-Platform:~$ 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.
ludewe@ludewe-Virtual-Platform:~$
```

Assignment 4.3: Compile

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

The c- and Java files.

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, fib.py

Which source code files are interpreted by an interpreter?

Fibonacci.java, fib.py

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

C file

How do I run a Java program?

Java Fibonacci.java

How do I run a Python program?

Python3 fib.py

How do I run a C program?

./fib

How do I run a Bash script?

./scriptname.sh

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

No

Take relevant screenshots of the following commands:

- Compile the source files where necessary

```
ludewe@ludewe-Virtual-Platform:~/Downloads/code$ gcc -o fib fib.c
```

```
ludewe@ludewe-Virtual-Platform:~/Downloads/code$ javac Fibonacci.java
```

- Make them executable

```
ludewe@ludewe-VMware-Virtual-Platform:~/Downloads/code$ ls
fib fib.c Fibonacci.class Fibonacci.java fib.py fib.sh runall.sh
ludewe@ludewe-VMware-Virtual-Platform:~/Downloads/code$
```

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

The fastest is the c file.

```
I+I ludewe@ludewe-VMware-Virtual-Platform:~/Downloads/code Q = - x
Running C program:
Fibonacci(19) = 4181
Execution time: 0.05 milliseconds

Running Java program:
Fibonacci(19) = 4181
Execution time: 0.63 milliseconds

Running Python program:
Fibonacci(19) = 4181
Execution time: 0.83 milliseconds

Running BASH Script
Fibonacci(19) = 4181
Excution time 16252 milliseconds

ludewe@ludewe-VMware-Virtual-Platform:~/Downloads/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.

```
gcc -O3 fib.c -o fib
```

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

```
gcc -O3 fib.c -o fib
```

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

It does. It saved 0.04 milliseconds.

The screenshot shows a terminal window with a dark background and light-colored text. The title bar reads "ludewe@ludewe-VMware-Virtual-Platform: ~/Documents/code". The terminal displays four separate runs of different programs, each calculating the 19th Fibonacci number (4181) and reporting its execution time. The programs are: C (Execution time: 0.01 milliseconds), Java (Execution time: 0.28 milliseconds), Python (Execution time: 0.35 milliseconds), and BASH Script (Execution time: 6812 milliseconds).

```
ludewe@ludewe-VMware-Virtual-Platform: ~/Documents/code$ ./fib_c
Running C program:
Fibonacci(19) = 4181
Execution time: 0.01 milliseconds

ludewe@ludewe-VMware-Virtual-Platform: ~/Documents/code$ java fib.java
Running Java program:
Fibonacci(19) = 4181
Execution time: 0.28 milliseconds

ludewe@ludewe-VMware-Virtual-Platform: ~/Documents/code$ python fib.py
Running Python program:
Fibonacci(19) = 4181
Execution time: 0.35 milliseconds

ludewe@ludewe-VMware-Virtual-Platform: ~/Documents/code$ ./fib_bs
Running BASH Script
Fibonacci(19) = 4181
Execution time: 6812 milliseconds
```

- 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 interpreted:  
Fibonacci(19) = 4181  
Execution time: 0.06 milliseconds  
  
Running C program compiled/optimised:  
Fibonacci(19) = 4181  
Execution time: 0.01 milliseconds  
  
Running Java program:  
Fibonacci(19) = 4181  
Execution time: 0.26 milliseconds  
  
Running Python program:  
Fibonacci(19) = 4181  
Execution time: 0.33 milliseconds  
  
Running BASH Script  
Fibonacci(19) = 4181  
Excution time 6763 milliseconds
```

Ludewe@ludewe-VMware-Virtual-Platform:~/Documents/code\$ █

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.

OakSim

Open	Run	250	Step	Reset																																																										
<pre> 1 Main: 2 mov r1, #2 3 mov r2, #4 4 mov r0, r1 5 Loop: 6 7 mul r0, r1, r0 8 sub r2,r2, #1 9 cmp r2,#1 10 BEQ End 11 B Loop 12 13 End: </pre>																																																														
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">Register</th> <th style="width: 90%;">Value</th> </tr> </thead> <tbody> <tr><td>R0</td><td>10</td></tr> <tr><td>R1</td><td>2</td></tr> <tr><td>R2</td><td>1</td></tr> <tr><td>R3</td><td>0</td></tr> <tr><td>R4</td><td>0</td></tr> <tr><td>R5</td><td>0</td></tr> <tr><td>R6</td><td>0</td></tr> <tr><td>R7</td><td>0</td></tr> <tr><td>R8</td><td>0</td></tr> </tbody> </table> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">Address</th> <th style="width: 90%;">Value</th> </tr> </thead> <tbody> <tr><td>0x00010000:</td><td>02 10 A0 E3 04 20 A0 E3 01 00</td></tr> <tr><td>0x00010010:</td><td>01 20 42 E2 01 00 52 E3 00 00</td></tr> <tr><td>0x00010020:</td><td>00 00 00 00 00 00 00 00 00 00 00 00</td></tr> <tr><td>0x00010030:</td><td>00 00 00 00 00 00 00 00 00 00 00 00</td></tr> <tr><td>0x00010040:</td><td>00 00 00 00 00 00 00 00 00 00 00 00</td></tr> <tr><td>0x00010050:</td><td>00 00 00 00 00 00 00 00 00 00 00 00</td></tr> <tr><td>0x00010060:</td><td>00 00 00 00 00 00 00 00 00 00 00 00</td></tr> <tr><td>0x00010070:</td><td>00 00 00 00 00 00 00 00 00 00 00 00</td></tr> <tr><td>0x00010080:</td><td>00 00 00 00 00 00 00 00 00 00 00 00</td></tr> <tr><td>0x00010090:</td><td>00 00 00 00 00 00 00 00 00 00 00 00</td></tr> <tr><td>0x000100A0:</td><td>00 00 00 00 00 00 00 00 00 00 00 00</td></tr> <tr><td>0x000100B0:</td><td>00 00 00 00 00 00 00 00 00 00 00 00</td></tr> <tr><td>0x000100C0:</td><td>00 00 00 00 00 00 00 00 00 00 00 00</td></tr> <tr><td>0x000100D0:</td><td>00 00 00 00 00 00 00 00 00 00 00 00</td></tr> <tr><td>0x000100E0:</td><td>00 00 00 00 00 00 00 00 00 00 00 00</td></tr> <tr><td>0x000100F0:</td><td>00 00 00 00 00 00 00 00 00 00 00 00</td></tr> <tr><td>0x00010100:</td><td>00 00 00 00 00 00 00 00 00 00 00 00</td></tr> <tr><td>0x00010110:</td><td>00 00 00 00 00 00 00 00 00 00 00 00</td></tr> </tbody> </table>					Register	Value	R0	10	R1	2	R2	1	R3	0	R4	0	R5	0	R6	0	R7	0	R8	0	Address	Value	0x00010000:	02 10 A0 E3 04 20 A0 E3 01 00	0x00010010:	01 20 42 E2 01 00 52 E3 00 00	0x00010020:	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	0x00010040:	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	0x00010060:	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	0x00010080:	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	0x000100A0:	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	0x000100C0:	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	0x000100E0:	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	0x00010100:	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
Register	Value																																																													
R0	10																																																													
R1	2																																																													
R2	1																																																													
R3	0																																																													
R4	0																																																													
R5	0																																																													
R6	0																																																													
R7	0																																																													
R8	0																																																													
Address	Value																																																													
0x00010000:	02 10 A0 E3 04 20 A0 E3 01 00																																																													
0x00010010:	01 20 42 E2 01 00 52 E3 00 00																																																													
0x00010020:	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																																																													
0x00010040:	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																																																													
0x00010060:	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																																																													
0x00010080:	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																																																													
0x000100A0:	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																																																													
0x000100C0:	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																																																													
0x000100E0:	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																																																													
0x00010100:	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																																																													

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