

Template Week 4 – Software

Student number: 529471

Assignment 4.1: ARM assembly

Screenshot of working assembly code of factorial calculation:

The screenshot shows the OakSim ARM simulator interface. The left pane contains the assembly code for a factorial calculation. The right pane shows the register values and a memory dump.

```
1 Main:
2   mov r2, #5
3   mov r1, #1
4 Loop:
5   cmp r2, #0
6   beq End
7   mul r1, r1, r2
8   sub r2, r2, #1
9   b Loop
10
11 End:
```

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

Memory dump (hex):

```
0x00010000: 05 20 A0 E3 01 10 A0 E3 00 00 52 E3 02 00 00 0A
0x00010010: 91 02 01 E0 01 20 42 E2 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
0x00010090: 00 00 00 00 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 00 00 00 00
0x000100B0: 00 00 00 00 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 00 00 00 00
0x000100D0: 00 00 00 00 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 00 00 00 00
0x000100F0: 00 00 00 00 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 00 00 00 00
0x00010110: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00010120: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00010130: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00010140: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00010150: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00010160: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00010170: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00010180: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00010190: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000101A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000101B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000101C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000101D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Screenshot 1 - Factorial calculation in Assembly

Assignment 4.2: Programming languages

Take screenshots that the following commands work:

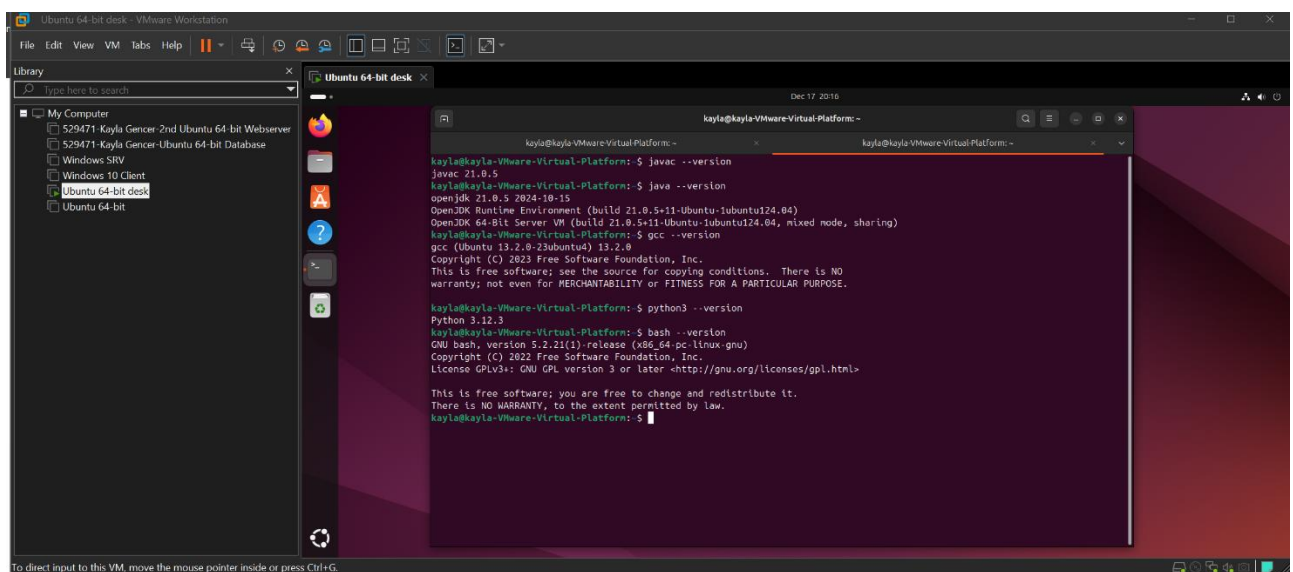
`javac --version`

`java --version`

`gcc --version`

`python3 --version`

`bash --version`



Screenshot 2 - Version commands

Assignment 4.3: Compile

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

Java: The Java source code must be compiled into bytecode for the JVM to interpret and run the bytecode during program execution (Strmecki, 2024).

C: C needs a compiler to transform its code into an executable form so that the program can run, as it is a mid-level language (GeeksforGeeks, 2024).

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

C

Which source code files are compiled to byte code?

Java

Which source code files are interpreted by an interpreter?

Python

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

C, as it is known to be the fastest programming language for low-level development (Joberty, 2024).

How do I run a Java program?

1. Make sure that the JDK is installed by typing the command **"javac --version"** in the terminal. If the JDK is not installed, type the command **"sudo apt install default-jdk"**.
2. In the terminal, access the folder where the file is in with the command **"cd directoryname"**.
3. Check for the file by typing the command **"ls"**.
4. Use the command **"javac"** then add the filename and **".java"** after it (**javac filename.java**). This will create a new file in the same directory as **"filename.class"**.
5. Open the terminal again and navigate to the directory of the Java program and run the command **"java filename"**. This command runs the program, generating the output in the terminal.

(Mannan, 2024)

How do I run a Python program?

1. Make sure that Python is installed by typing the command **"python3 --version"** in the terminal. If Python is not installed, type the command **"sudo apt install python3"**.
2. In the terminal, access the folder where the file is in with the command **"cd directoryname"**.
3. Check for the file by typing the command **"ls"**.
4. Type the command **"python3 filename.py"**. This command runs the program, generating the output in the terminal.

(Running Python File in Terminal, n.d.)

How do I run a C program?

1. Make sure that a C compiler (the most popular compiler is GNU Compiler Collection, aka gcc) is installed by typing the command **"gcc --version"** in the terminal. If the compiler is not installed, type the command **"sudo apt install gcc"**.
2. In the terminal, access the folder where the file is in with the command **"cd directoryname"**.
3. Check for the file by typing the command **"ls"**.
4. Generate the object file by compiling the program with the command **"gcc -o filename filename.c"**.
5. Run the generated object file by typing the command **"./filename"**. This will run the C program, generating the output in the terminal.

(Prakash, 2023)

How do I run a Bash script?

1. Make sure that Bash is installed by typing the command **"bash --version"** in the terminal. If it is not installed, type the command **"sudo apt-get install bash"**.
2. In the terminal, access the folder where the file is in with the command **"cd directoryname"**.
3. Check for the file by typing the command **"ls"**.
4. Type the command **"bash filename.sh"**. This will run the bash script, generating the output in the terminal.

(Dancuk, 2021)

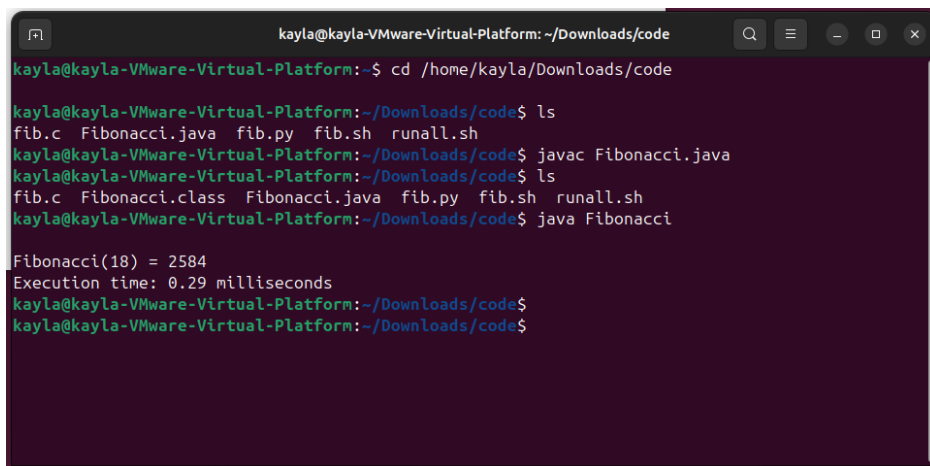
If I compile the above source code, will a new file be created? If so, which file?

Compiling the above source code will generate new files for Java and C.

Take relevant screenshots of the following commands:

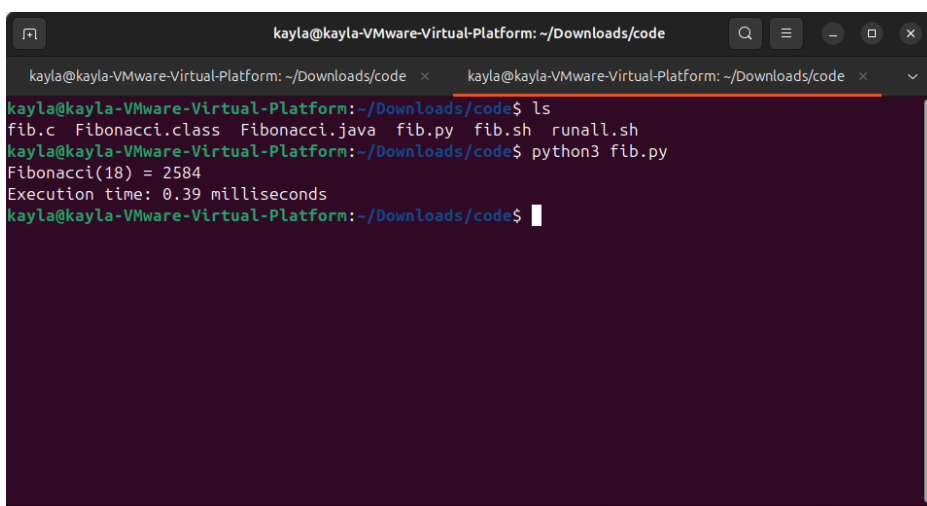
- Compile the source files where necessary
- Make them executable
- Run them
- Which (compiled) source code file performs the calculation the fastest?

C is the fastest.



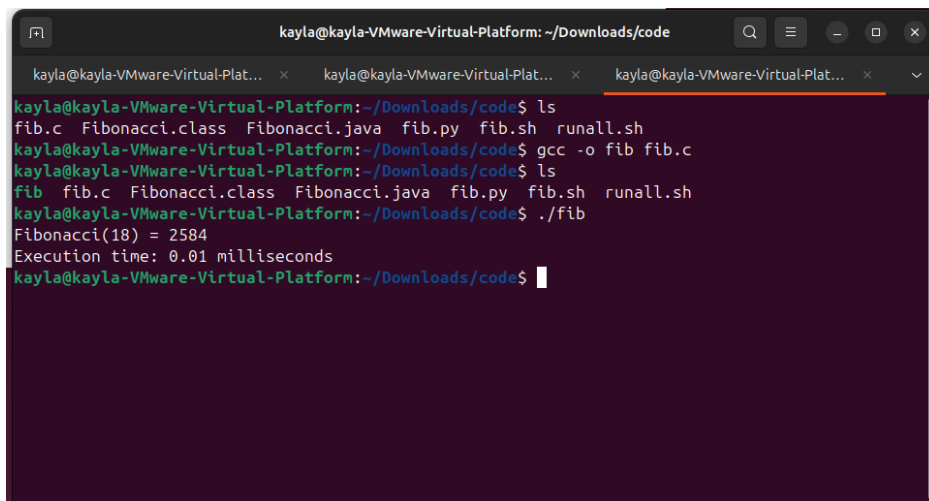
```
kayla@kayla-VMware-Virtual-Platform: ~/Downloads/code
kayla@kayla-VMware-Virtual-Platform:~$ cd /home/kayla/Downloads/code
kayla@kayla-VMware-Virtual-Platform:~/Downloads/code$ ls
fib.c Fibonacci.java fib.py fib.sh runall.sh
kayla@kayla-VMware-Virtual-Platform:~/Downloads/code$ javac Fibonacci.java
kayla@kayla-VMware-Virtual-Platform:~/Downloads/code$ ls
fib.c Fibonacci.class Fibonacci.java fib.py fib.sh runall.sh
kayla@kayla-VMware-Virtual-Platform:~/Downloads/code$ java Fibonacci
Fibonacci(18) = 2584
Execution time: 0.29 milliseconds
kayla@kayla-VMware-Virtual-Platform:~/Downloads/code$
kayla@kayla-VMware-Virtual-Platform:~/Downloads/code$
```

Screenshot 3 - Java



```
kayla@kayla-VMware-Virtual-Platform: ~/Downloads/code
kayla@kayla-VMware-Virtual-Platform:~/Downloads/code$ ls
fib.c Fibonacci.class Fibonacci.java fib.py fib.sh runall.sh
kayla@kayla-VMware-Virtual-Platform:~/Downloads/code$ python3 fib.py
Fibonacci(18) = 2584
Execution time: 0.39 milliseconds
kayla@kayla-VMware-Virtual-Platform:~/Downloads/code$
```

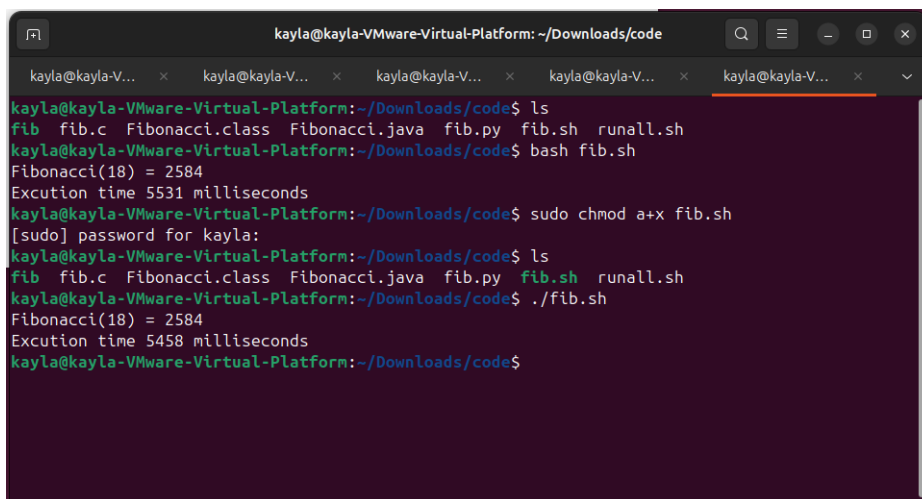
Screenshot 4 - Python



A terminal window titled 'kayla@kayla-VMware-Virtual-Platform: ~/Downloads/code'. The terminal shows the following commands and output:

```
kayla@kayla-VMware-Virtual-Platform:~/Downloads/code$ ls
fib.c Fibonacci.class Fibonacci.java fib.py fib.sh runall.sh
kayla@kayla-VMware-Virtual-Platform:~/Downloads/code$ gcc -o fib fib.c
kayla@kayla-VMware-Virtual-Platform:~/Downloads/code$ ls
fib fib.c Fibonacci.class Fibonacci.java fib.py fib.sh runall.sh
kayla@kayla-VMware-Virtual-Platform:~/Downloads/code$ ./fib
Fibonacci(18) = 2584
Execution time: 0.01 milliseconds
kayla@kayla-VMware-Virtual-Platform:~/Downloads/code$
```

Screenshot 5 - C



A terminal window titled 'kayla@kayla-VMware-Virtual-Platform: ~/Downloads/code'. The terminal shows the following commands and output:

```
kayla@kayla-VMware-Virtual-Platform:~/Downloads/code$ ls
fib fib.c Fibonacci.class Fibonacci.java fib.py fib.sh runall.sh
kayla@kayla-VMware-Virtual-Platform:~/Downloads/code$ bash fib.sh
Fibonacci(18) = 2584
Execution time 5531 milliseconds
kayla@kayla-VMware-Virtual-Platform:~/Downloads/code$ sudo chmod a+x fib.sh
[sudo] password for kayla:
kayla@kayla-VMware-Virtual-Platform:~/Downloads/code$ ls
fib fib.c Fibonacci.class Fibonacci.java fib.py fib.sh runall.sh
kayla@kayla-VMware-Virtual-Platform:~/Downloads/code$ ./fib.sh
Fibonacci(18) = 2584
Execution time 5458 milliseconds
kayla@kayla-VMware-Virtual-Platform:~/Downloads/code$
```

Screenshot 6 - Bash

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 r0, #1
```

```
mov r1, #2
```

```
mov r2, #4
```

Loop:

```
cmp r2, #0
```

```
beq End
```

```
mul r0, r0, r1
```

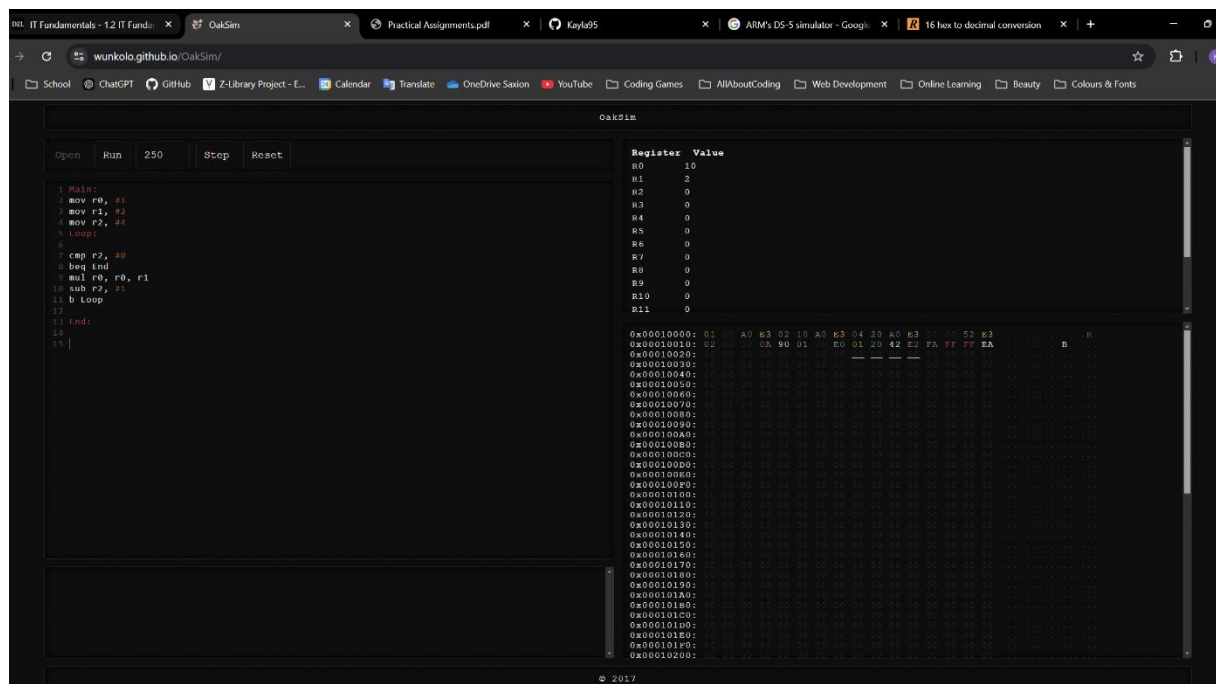
```
sub r2, #1
```

```
b Loop
```

End:

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

Screenshot of the completed code here.



Screenshot 7 - Power of 2 in Assembly

References

- Dancuk, M. (2021, December 9). *How To Run A Bash Script {7 Methods}*. Retrieved from phoenixNAP: <https://phoenixnap.com/kb/run-bash-script>
- GeeksforGeeks. (2024, October 11). *Compiling a C Program: Behind the Scenes*. Retrieved from GeeksforGeeks: <https://www.geeksforgeeks.org/compiling-a-c-program-behind-the-scenes/>
- Joberty. (2024, February 6). *The Fastest Programming Languages for Developers*. Retrieved from Joberty: <https://www.joberty.com/blog/the-fastest-programming-languages/>
- Mannan, A. (2024, June 30). *How to Run Java Programs in Linux Terminal?* Retrieved from Dracula Servers: <https://draculaservers.com/tutorials/run-java-programs-linux-terminal/#:~:text=If%20you%20are%20only%20looking,the%20program%20on%20the%20terminal.>
- Prakash, A. (2023, March 29). *How to Write, Compile and Run a C Program in Ubuntu and Other Linux Distributions [Beginner's Tip]*. Retrieved from It's Foss: <https://itsfoss.com/run-c-program-linux/>
- Running Python File in Terminal*. (n.d.). Retrieved from ask Ubuntu: <https://askubuntu.com/questions/244378/running-python-file-in-terminal>
- Strmecki, D. (2024, January 8). *Is Java a Compiled or Interpreted Language?* Retrieved from Baeldung: <https://www.baeldung.com/java-compiled-interpreted#:~:text=This%20might%20sound%20like%20a,executes%20this%20code%20at%20runtime.>

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