

# Week 4 – Software

Student number: 577029

## Assignment 4.1: ARM assembly

Screenshot of working assembly code of factorial calculation:

```
OakSim
```

Open Run 250 Step Reset

```
1 Main:
2     mov r1, #1
3     mov r2, #5
4
5 Loop:
6     mul r1, r1, r2
7     sub r2, r2, #1
8
9     cmp r2, #0
10    beq End
11    b Loop
12 End:
13
14 ---577029
```

Register Value	
R0	0
R1	0
R2	0
R3	0
R4	0
R5	0
R6	0
R7	0
R8	0
R9	0
R10	0
R11	0
R12	0
SP	10000
LR	0
PC	10000
CPSR	13

## Assignment 4.2: Programming languages

Take screenshots that the following commands work:

javac -version

java -version

gcc --version

python3 --version

bash --version

```

puja@puja-VMware-Virtual-Platform:~$ javac --version
javac 21.0.9
puja@puja-VMware-Virtual-Platform:~$ java --version
openjdk 21.0.9 2025-10-21
OpenJDK Runtime Environment (build 21.0.9+10-Ubuntu-124.04)
OpenJDK 64-Bit Server VM (build 21.0.9+10-Ubuntu-124.04, mixed mode, sharing)
puja@puja-VMware-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.

puja@puja-VMware-Virtual-Platform:~$ python3 --version
Python 3.12.3
puja@puja-VMware-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.
puja@puja-VMware-Virtual-Platform:~$

```

### Assignment 4.3: Compile

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

C and java required compilation steps

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

C, compile the code directly into native machine

Which source code files are compiled to byte code?

Java & Python

Which source code files are interpreted by an interpreter?

Python & bash

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

C. because its compiled directly to the machine code and has no runtime environment (VM) managing memory code.

How do I run a Java program?

```

puja@puja-VMware-Virtual-Platform:~$ cd Documents
puja@puja-VMware-Virtual-Platform:~/Documents$ nano Main.java
puja@puja-VMware-Virtual-Platform:~/Documents$ ls
backup.txt  Main.java  test.txt
puja@puja-VMware-Virtual-Platform:~/Documents$ javac Main.java
puja@puja-VMware-Virtual-Platform:~/Documents$ java Main
Hello from Ubuntu!
puja@puja-VMware-Virtual-Platform:~/Documents$

```

I need to make java file first then run it.

How do I run a Python program?

```
puja@puja-VMware-Virtual-Platform:~/Documents$ nano main.py
puja@puja-VMware-Virtual-Platform:~/Documents$ python3 main.py
Hello from Python on Ubuntu!
puja@puja-VMware-Virtual-Platform:~/Documents$
```

How do I run a C program?

```
puja@puja-VMware-Virtual-Platform:~/Documents$ nano main.c
puja@puja-VMware-Virtual-Platform:~/Documents$ gcc main.c -o my_c_app
```



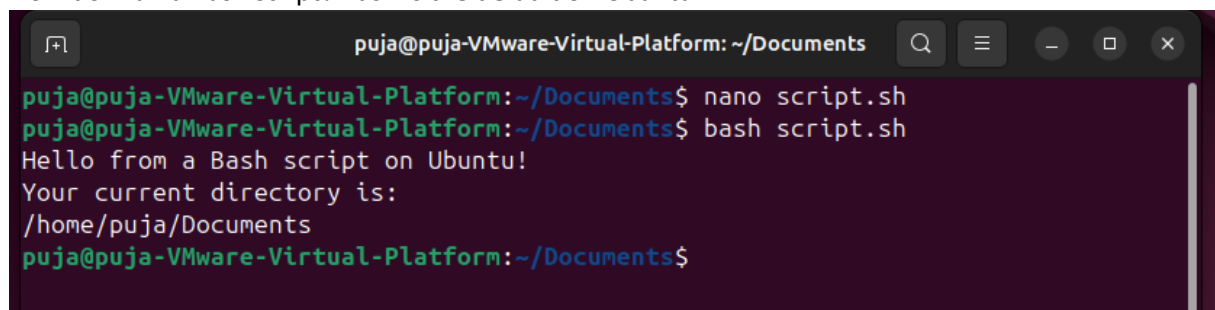
```
main.c
~/Documents

metadata_oldca | message.txt | test.txt | puja@192.168.1 | main.c x

#include <stdio.h>

int main() {
    printf("Hello from C on Ubuntu!\n");
    return 0;
}
```

How do I run a Bash script? Bash is the default on Ubuntu.



```
puja@puja-VMware-Virtual-Platform: ~/Documents

puja@puja-VMware-Virtual-Platform:~/Documents$ nano script.sh
puja@puja-VMware-Virtual-Platform:~/Documents$ bash script.sh
Hello from a Bash script on Ubuntu!
Your current directory is:
/home/puja/Documents
puja@puja-VMware-Virtual-Platform:~/Documents$
```

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

Only python and bash is not making 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.

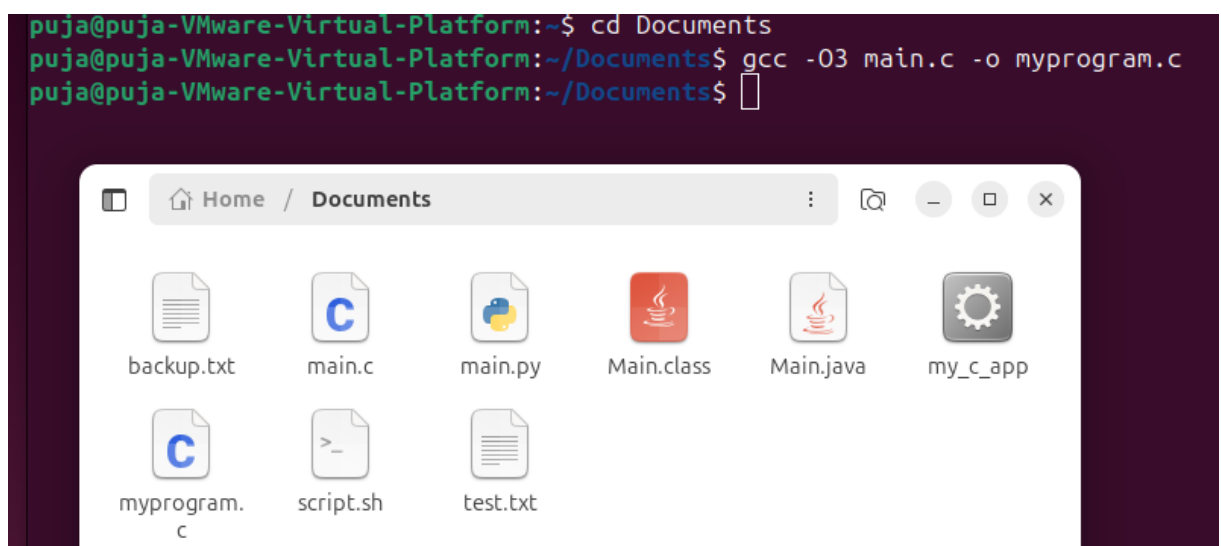
`gcc -Ofast`

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

`gcc -O3 fib.c -o fib_optimized`

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

`gcc [optimization_flag] [source_file] -o [output_file]`



- 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.

```

puja@puja-VMware-Virtual-Platform:~/Documents$ nano runall.sh
puja@puja-VMware-Virtual-Platform:~/Documents$ bash runall.sh
-----
1. C Version (Compiled with -O3)
Hello from C on Ubuntu!
-----
2. Java Version
Hello from Ubuntu!
-----
3. Python Version
Hello from Python on Ubuntu!
-----
4. Bash Version
Hello from a Bash script on Ubuntu!
Your current directory is:
/home/puja/Documents
-----
Done!
puja@puja-VMware-Virtual-Platform:~/Documents$

```

#### 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
mov r3, #1

```

Loop:

End:

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

Screenshot of the completed code here.

OakSim

Open Run 250 Step Reset

```
1 Main:
2   mov r1, #2
3   mov r2, #4
4   mov r0, #1
5
6 Loop:
7   mul r0, r0, r1
8   sub r2, r2, #1
9
10  cmp r2, #0
11  beq End
12  b Loop
13 End:
14
15 ---577029
```

Register	Value
R0	10
R1	2
R2	0
R3	0
R4	0
R5	0
R6	0
R7	0
R8	0
R9	0
R10	0
R11	0
R12	0
SP	10000
LR	0
PC	1038c
CPSR	60000013

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