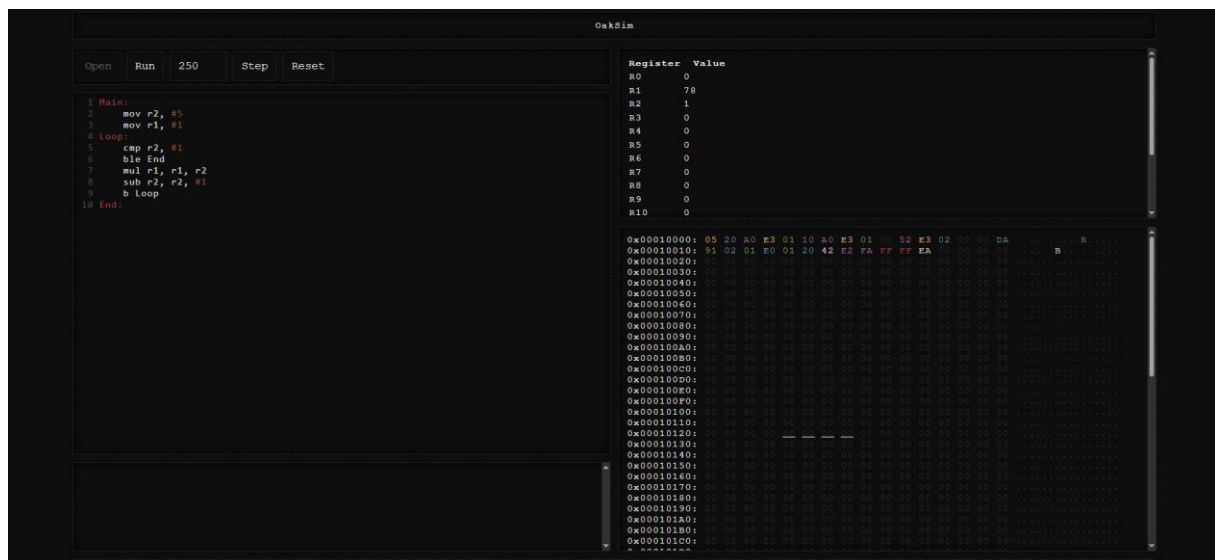


# Template Week 4 – Software

Student number:582777

## Assignment 4.1: ARM assembly

Screenshot of working assembly code of factorial calculation:



The screenshot shows an ARM assembly simulator interface. On the left, the assembly code is displayed:

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

On the right, the Register Value table is shown:

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

Below the register table, a memory dump is visible, showing hexadecimal addresses and their corresponding values in hexadecimal and ASCII. The memory dump starts at 0x00010000 and ends at 0x000101C0.

## Assignment 4.2: Programming languages

Take screenshots that the following commands work:

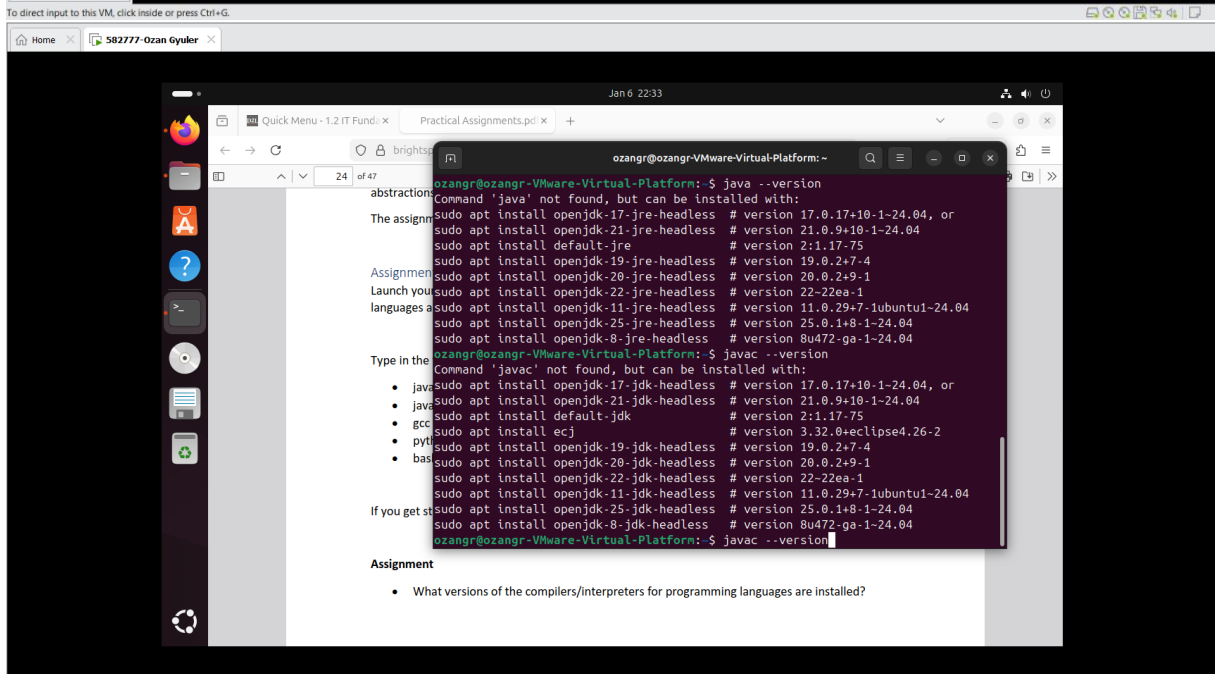
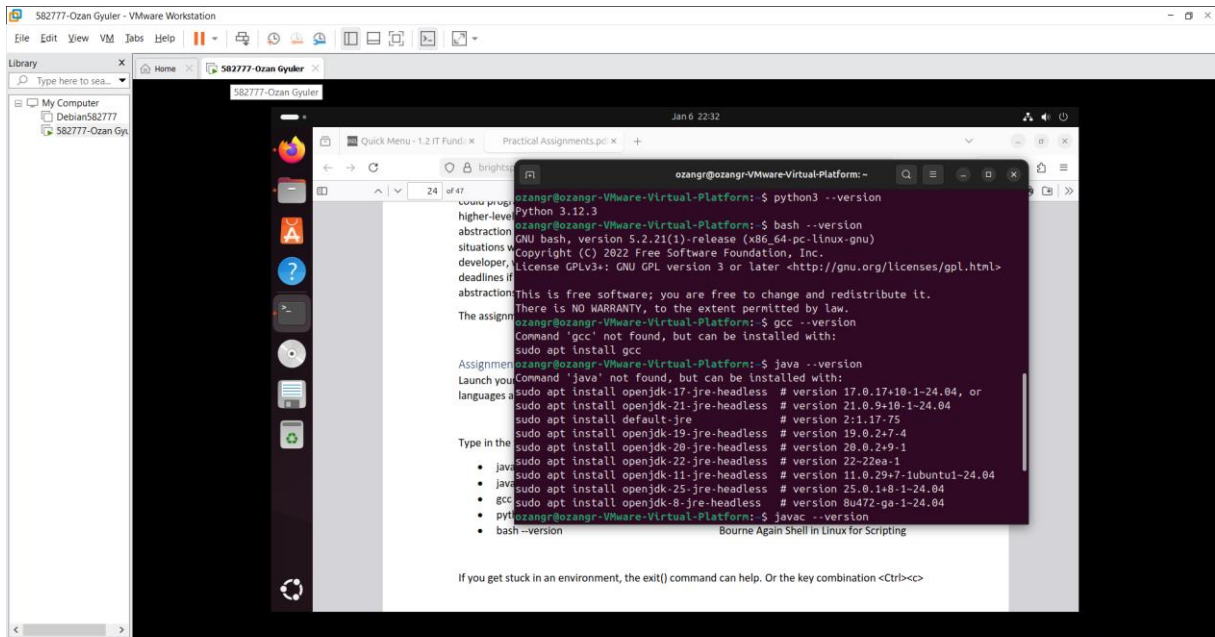
javac --version

java --version

gcc --version

python3 --version

bash --version



### Assignment 4.3: Compile

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

Fibonacci.java and fib.c needs compilation.

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

fib.c

Compiled by a C compiler (e.g., gcc) directly into native machine code.

The resulting executable runs directly on the CPU.

Which source code files are compiled to byte code?

Fibonacci.java compiled by javac into Java bytecode.

Which source code files are interpreted by an interpreter?

fib.py - Interpreted by the Python interpreter

fib.sh -Interpreted by a shell interpreter (e.g., bash)

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

C (fib.c) - Compiled to native machine code

How do I run a Java program?

java Fibonacci

How do I run a Python program?

python3 fib.py - python fib.py depends on the system

How do I run a C program?

./fib

How do I run a Bash script?

chmod +x fib.sh make executable (once)

./fib.sh run or bash fib.sh run

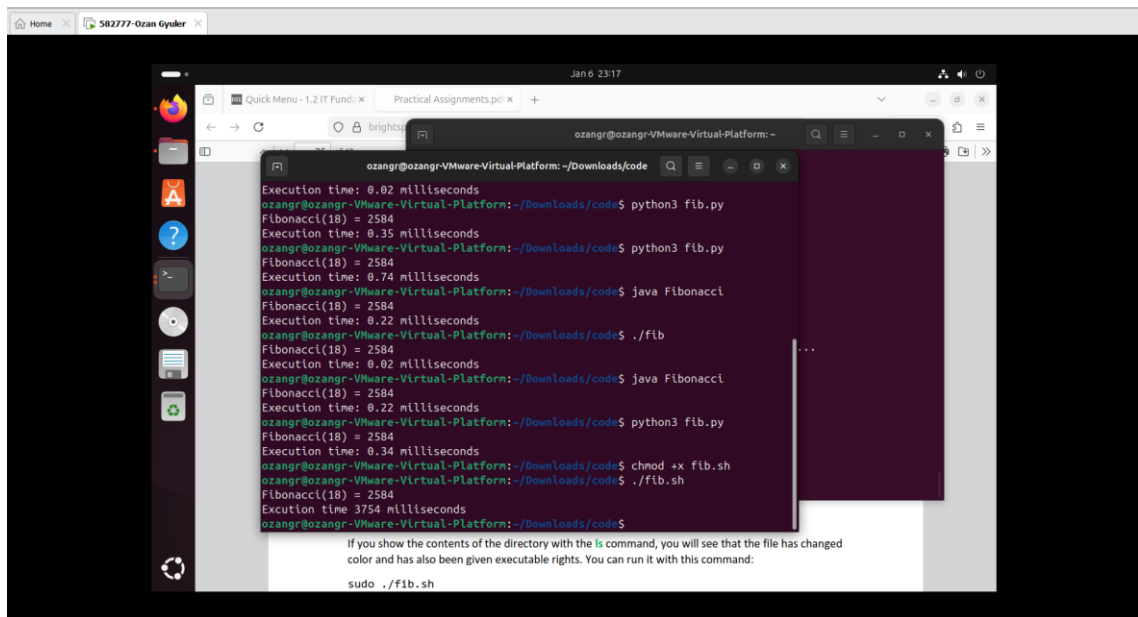
If I compile the above source code, will a new file be created? If so, which file?

fib.c -- fib (or fib.exe on Windows)

Fibonacci.java -- Fibonacci.class

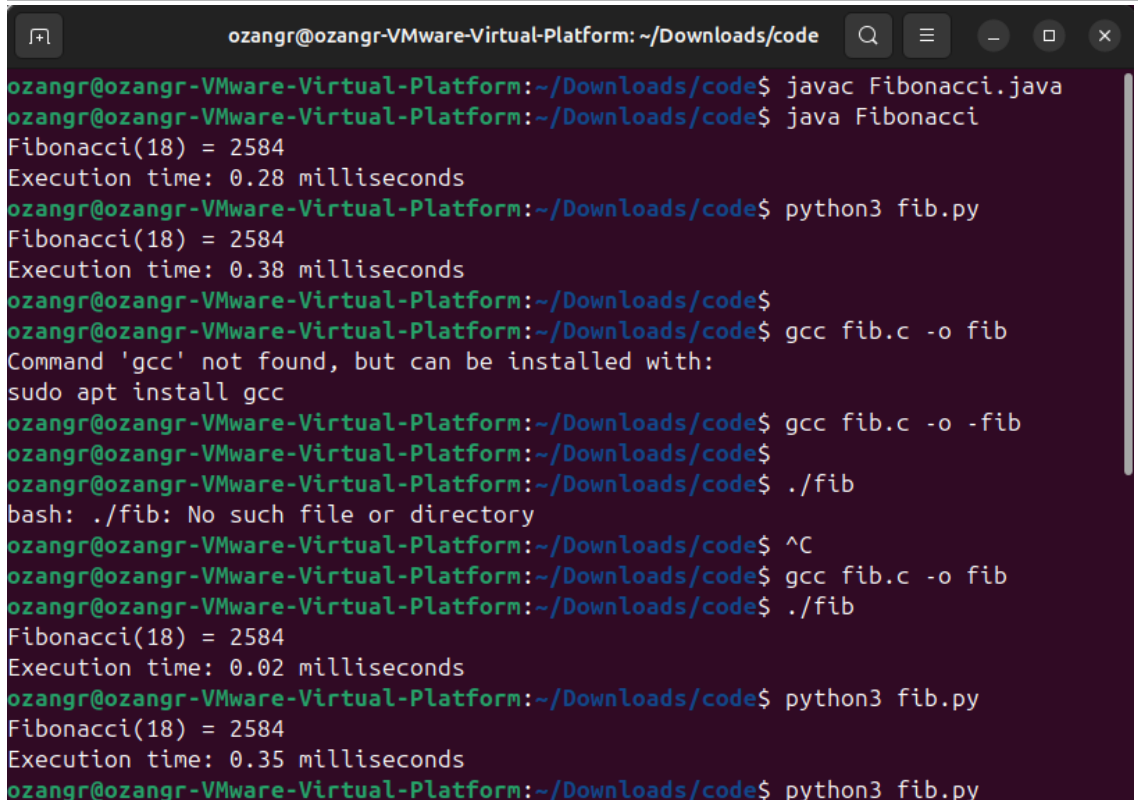
Take relevant screenshots of the following commands:

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



```
ozangr@ozangr-VMware-Virtual-Platform: ~/Downloads/code
Execution time: 0.02 milliseconds
ozangr@ozangr-VMware-Virtual-Platform:~/Downloads/code$ python3 fib.py
Fibonacci(18) = 2584
Execution time: 0.35 milliseconds
ozangr@ozangr-VMware-Virtual-Platform:~/Downloads/code$ python3 fib.py
Fibonacci(18) = 2584
Execution time: 0.74 milliseconds
ozangr@ozangr-VMware-Virtual-Platform:~/Downloads/code$ java Fibonacci
Fibonacci(18) = 2584
Execution time: 0.22 milliseconds
ozangr@ozangr-VMware-Virtual-Platform:~/Downloads/code$ ./fib
Fibonacci(18) = 2584
Execution time: 0.02 milliseconds
ozangr@ozangr-VMware-Virtual-Platform:~/Downloads/code$ java Fibonacci
Fibonacci(18) = 2584
Execution time: 0.22 milliseconds
ozangr@ozangr-VMware-Virtual-Platform:~/Downloads/code$ python3 fib.py
Fibonacci(18) = 2584
Execution time: 0.34 milliseconds
ozangr@ozangr-VMware-Virtual-Platform:~/Downloads/code$ chmod +x fib.sh
ozangr@ozangr-VMware-Virtual-Platform:~/Downloads/code$ ./fib.sh
Execution time 3754 milliseconds
ozangr@ozangr-VMware-Virtual-Platform:~/Downloads/code$
```

If you show the contents of the directory with the `ls` command, you will see that the file has changed color and has also been given executable rights. You can run it with this command:  
`sudo ./fib.sh`

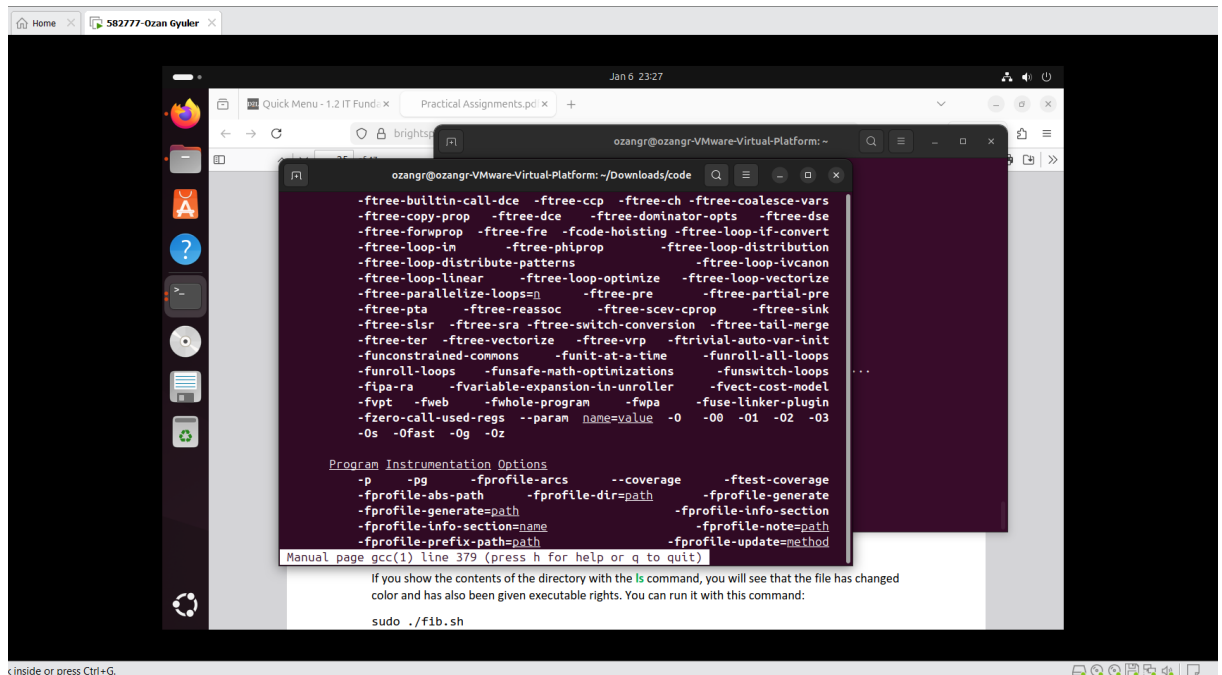


```
ozangr@ozangr-VMware-Virtual-Platform: ~/Downloads/code
ozangr@ozangr-VMware-Virtual-Platform:~/Downloads/code$ javac Fibonacci.java
ozangr@ozangr-VMware-Virtual-Platform:~/Downloads/code$ java Fibonacci
Fibonacci(18) = 2584
Execution time: 0.28 milliseconds
ozangr@ozangr-VMware-Virtual-Platform:~/Downloads/code$ python3 fib.py
Fibonacci(18) = 2584
Execution time: 0.38 milliseconds
ozangr@ozangr-VMware-Virtual-Platform:~/Downloads/code$
ozangr@ozangr-VMware-Virtual-Platform:~/Downloads/code$ gcc fib.c -o fib
Command 'gcc' not found, but can be installed with:
sudo apt install gcc
ozangr@ozangr-VMware-Virtual-Platform:~/Downloads/code$ gcc fib.c -o -fib
ozangr@ozangr-VMware-Virtual-Platform:~/Downloads/code$ ./fib
bash: ./fib: No such file or directory
ozangr@ozangr-VMware-Virtual-Platform:~/Downloads/code$ ^C
ozangr@ozangr-VMware-Virtual-Platform:~/Downloads/code$ gcc fib.c -o fib
ozangr@ozangr-VMware-Virtual-Platform:~/Downloads/code$ ./fib
Fibonacci(18) = 2584
Execution time: 0.02 milliseconds
ozangr@ozangr-VMware-Virtual-Platform:~/Downloads/code$ python3 fib.py
Fibonacci(18) = 2584
Execution time: 0.35 milliseconds
ozangr@ozangr-VMware-Virtual-Platform:~/Downloads/code$ python3 fib.py
```

## Assignment 4.4: Optimize

Take relevant screenshots of the following commands:

- 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.



```
ozangr@ozangr-VMware-Virtual-Platform: ~/Downloads/code
-ftree-builtin-call-dce -ftree-ccp -ftree-ch -ftree-coalesce-vars
-ftree-copy-prop -ftree-dce -ftree-dominator-opts -ftree-dse
-ftree-forwprop -ftree-fre -fcode-hoisting -ftree-loop-if-convert
-ftree-loop-in -ftree-phi-prop -ftree-loop-distribution
-ftree-loop-distribute-patterns -ftree-loop-ivcanon
-ftree-loop-linear -ftree-loop-optimize -ftree-loop-vectorize
-ftree-parallelize-loops=n -ftree-pre -ftree-partial-pre
-ftree-pta -ftree-reassoc -ftree-scev-cprop -ftree-sink
-ftree-slsr -ftree-sra -ftree-switch-conversion -ftree-tail-merge
-ftree-ter -ftree-vectorize -ftree-vrp -ftrivial-auto-var-init
-funconstrained-commons -funit-at-a-time -funroll-all-loops
-funroll-loops -funsafe-math-optimizations -funsafe-loops
-fipa-ra -fvariable-expansion-in-unroller -fvect-cost-model
-fvpt -fweb -fwhole-program -fipa -fuse-linker-plugin
-fzero-call-used-regs --param name=value -O -O0 -O1 -O2 -O3
-Os -Ofast -Og -Oz

Program Instrumentation Options
-p -pg -fprofile-arcs --coverage -ftest-coverage
-fprofile-abs-path -fprofile-dir=path -fprofile-generate
-fprofile-generate=path -fprofile-info-section
-fprofile-info-section=name -fprofile-note=path
-fprofile-prefix-path=path -fprofile-update=method

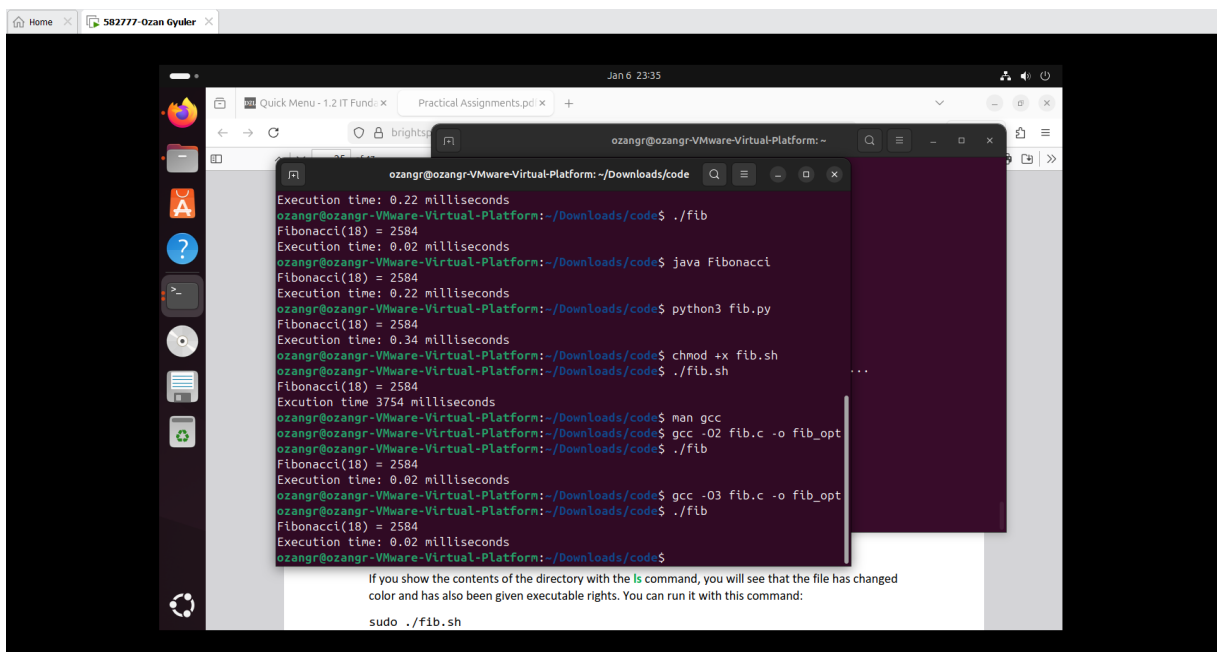
Manual page gcc(1) line 379 (press h for help or q to quit)

If you show the contents of the directory with the ls command, you will see that the file has changed
color and has also been given executable rights. You can run it with this command:

sudo ./fib.sh
```

Ofast is fast but not reliable and O2-O3 is more safe way to make it faster

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

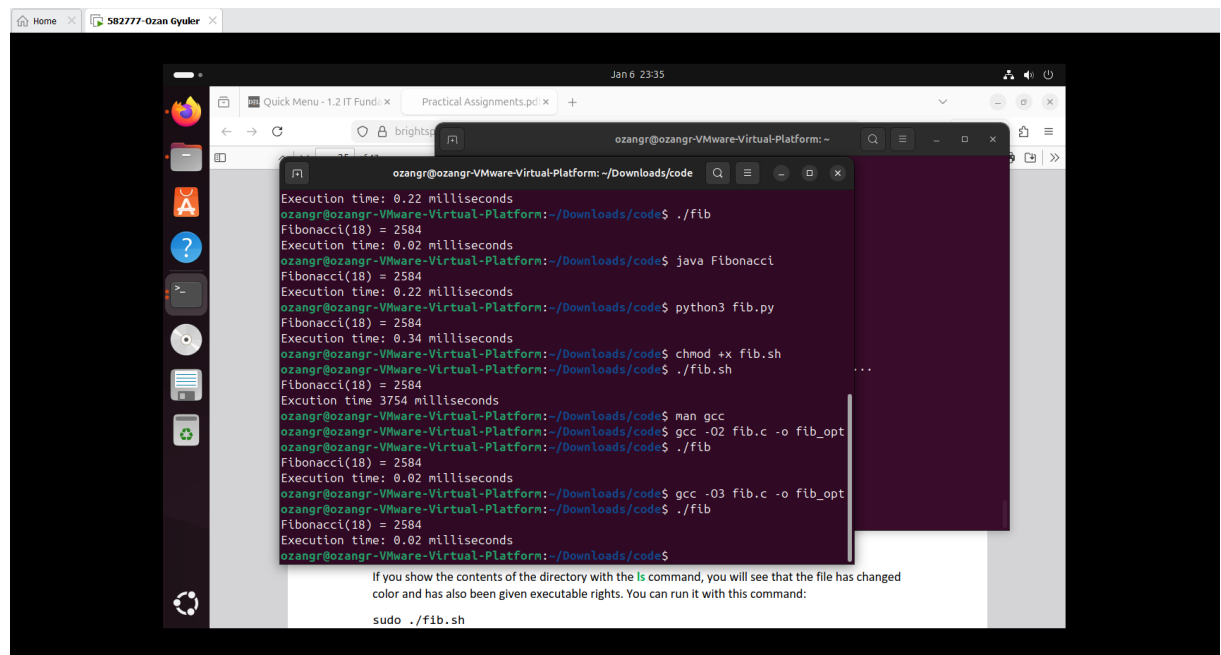


```
ozangr@ozangr-VMware-Virtual-Platform: ~/Downloads/code
Execution time: 0.22 milliseconds
ozangr@ozangr-VMware-Virtual-Platform: ~/Downloads/code$ ./fib
Fibonacci(18) = 2584
Execution time: 0.02 milliseconds
ozangr@ozangr-VMware-Virtual-Platform: ~/Downloads/code$ java Fibonacci
Fibonacci(18) = 2584
Execution time: 0.22 milliseconds
ozangr@ozangr-VMware-Virtual-Platform: ~/Downloads/code$ python3 fib.py
Fibonacci(18) = 2584
Execution time: 0.34 milliseconds
ozangr@ozangr-VMware-Virtual-Platform: ~/Downloads/code$ chmod +x fib.sh
ozangr@ozangr-VMware-Virtual-Platform: ~/Downloads/code$ ./fib.sh
Fibonacci(18) = 2584
Execution time 3754 milliseconds
ozangr@ozangr-VMware-Virtual-Platform: ~/Downloads/code$ man gcc
ozangr@ozangr-VMware-Virtual-Platform: ~/Downloads/code$ gcc -O2 fib.c -o fib_opt
ozangr@ozangr-VMware-Virtual-Platform: ~/Downloads/code$ ./fib
Fibonacci(18) = 2584
Execution time: 0.02 milliseconds
ozangr@ozangr-VMware-Virtual-Platform: ~/Downloads/code$ gcc -O3 fib.c -o fib_opt
ozangr@ozangr-VMware-Virtual-Platform: ~/Downloads/code$ ./fib
Fibonacci(18) = 2584
Execution time: 0.02 milliseconds
ozangr@ozangr-VMware-Virtual-Platform: ~/Downloads/code$

If you show the contents of the directory with the ls command, you will see that the file has changed
color and has also been given executable rights. You can run it with this command:

sudo ./fib.sh
```

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

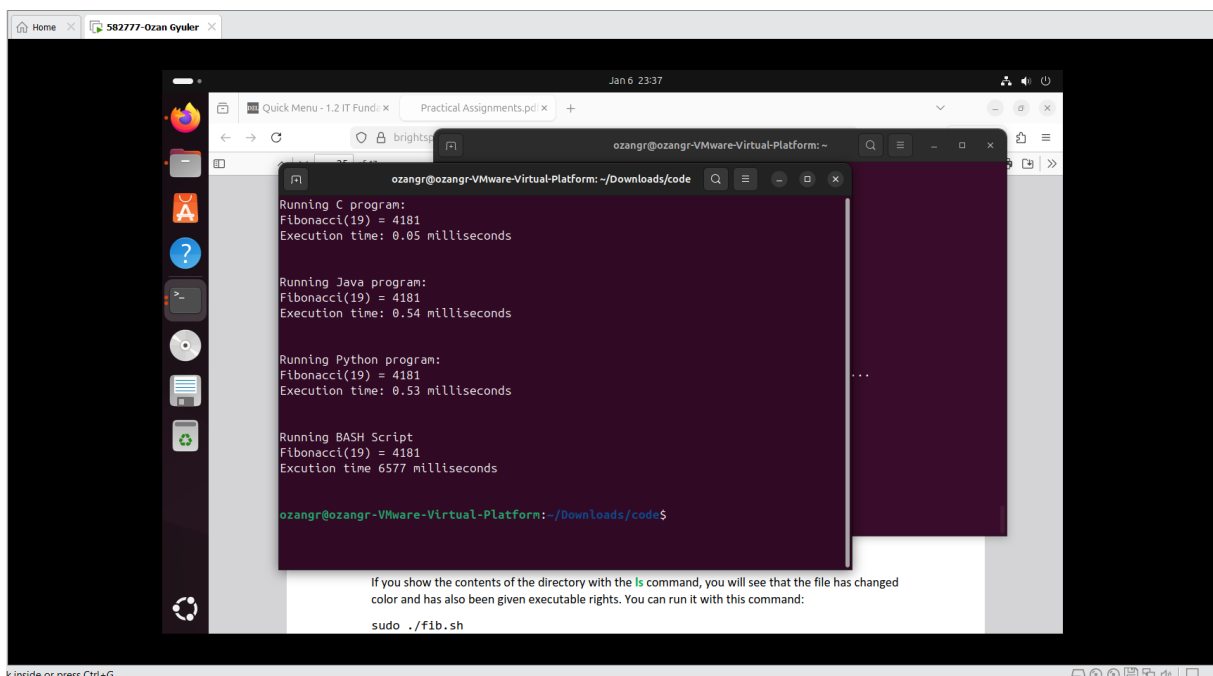


```
ozangr@ozangr-Virtual-Platform:~/Downloads/code$ ./fib
Execution time: 0.22 milliseconds
ozangr@ozangr-Virtual-Platform:~/Downloads/code$ ./fib
Fibonacci(18) = 2584
Execution time: 0.02 milliseconds
ozangr@ozangr-Virtual-Platform:~/Downloads/code$ java Fibonacci
Fibonacci(18) = 2584
Execution time: 0.22 milliseconds
ozangr@ozangr-Virtual-Platform:~/Downloads/code$ python3 fib.py
Fibonacci(18) = 2584
Execution time: 0.34 milliseconds
ozangr@ozangr-Virtual-Platform:~/Downloads/code$ chmod +x fib.sh
ozangr@ozangr-Virtual-Platform:~/Downloads/code$ ./fib.sh
Fibonacci(18) = 2584
Execution time 3754 milliseconds
ozangr@ozangr-Virtual-Platform:~/Downloads/code$ man gcc
ozangr@ozangr-Virtual-Platform:~/Downloads/code$ gcc -O2 fib.c -o fib_opt
ozangr@ozangr-Virtual-Platform:~/Downloads/code$ ./fib
Fibonacci(18) = 2584
Execution time: 0.02 milliseconds
ozangr@ozangr-Virtual-Platform:~/Downloads/code$ gcc -O3 fib.c -o fib_opt
ozangr@ozangr-Virtual-Platform:~/Downloads/code$ ./fib
Fibonacci(18) = 2584
Execution time: 0.02 milliseconds
ozangr@ozangr-Virtual-Platform:~/Downloads/code$
```

If you show the contents of the directory with the `ls` command, you will see that the file has changed color and has also been given executable rights. You can run it with this command:

```
sudo ./fib.sh
```

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.



```
ozangr@ozangr-Virtual-Platform:~/Downloads/code$
Running C program:
Fibonacci(19) = 4181
Execution time: 0.05 milliseconds

Running Java program:
Fibonacci(19) = 4181
Execution time: 0.54 milliseconds

Running Python program:
Fibonacci(19) = 4181
Execution time: 0.53 milliseconds

Running BASH Script
Fibonacci(19) = 4181
Execution time 6577 milliseconds

ozangr@ozangr-Virtual-Platform:~/Downloads/code$
```

If you show the contents of the directory with the `ls` command, you will see that the file has changed color and has also been given executable rights. You can run it with this command:

```
sudo ./fib.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
mov r2, #4
mov r0, #1
```

Loop:

```
cmp r2, #0
beq End
mul r0, r0, r1
sub r2, r2, #1
b Loop
```

End:

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

Screenshot of the completed code here.

```
OaKSim
-----
Open Run 250 Stop Reset
1 Main:
2 mov r1, #2
3 mov r2, #4
4 mov r0, #1
5 Loop:
6 cmp r2, #0
7 beq End
8 mul r0, r0, r1
9 sub r2, r2, #1
10 b Loop
11 End

Register Value
R0 10
R1 2
R2 0
R3 0
R4 0
R5 0
R6 0
R7 0
R8 0
R9 0
R10 0

0x00010000: 02 10 A0 E3 04 20 A0 E3 01 10 E3 12 E3
0x00010010: 02 10 A0 E3 04 20 A0 E3 01 10 E3 12 E3
0x00010020:
0x00010030:
0x00010040:
0x00010050:
0x00010060:
0x00010070:
0x00010080:
0x00010090:
0x000100A0:
0x000100B0:
0x000100C0:
0x000100D0:
0x000100E0:
0x000100F0:
0x00010100:
0x00010110:
0x00010120:
0x00010130:
0x00010140:
0x00010150:
0x00010160:
0x00010170:
0x00010180:
0x00010190:
0x000101A0:
0x000101B0:
0x000101C0:
0x000101D0:
0x000101E0:
0x000101F0:
0x00010200:
-----
© 2017
```

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