Template Week 4 – Software

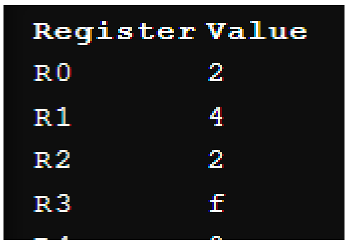
Student number:553809

Assignment 4.1: ARM assembly

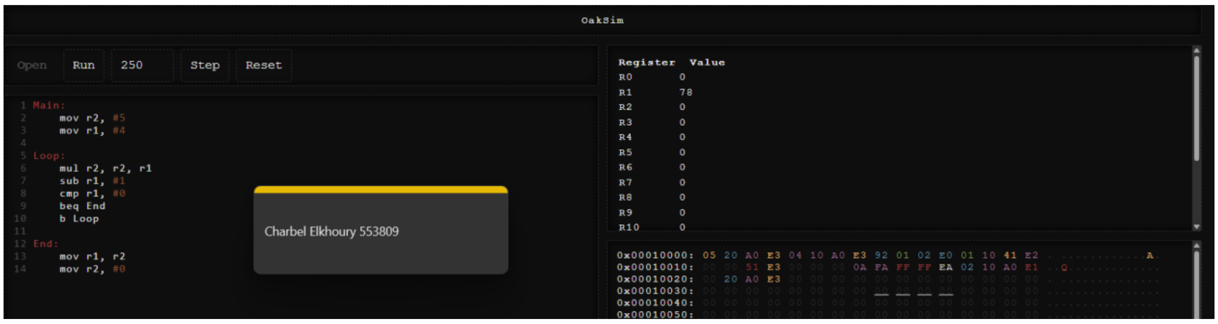
Screenshot of working assembly code of factorial calculation:

What is the value of:

- r0? = 2
- r1? = 4
- r2? = 2
- r3? = f



```
Register Value
R0       2
R1       4
R2       2
R3       f
```



```
Main:
    mov r2, #5
    mov r1, #4

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

End:
    mov r1, r2
    mov r2, #0
```

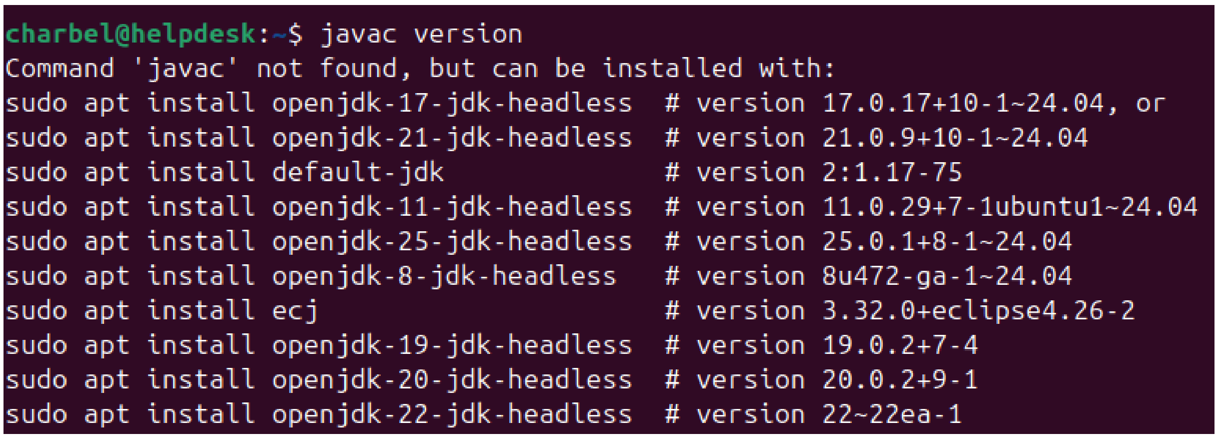
```
Register  Value
R0        0
R1        78
R2        0
R3        0
R4        0
R5        0
R6        0
R7        0
R8        0
R9        0
R10       0
```

```
0x00010000: 05 20 A0 E3 04 10 A0 E3 92 01 02 E0 01 10 41 E2 ..............A.
0x00010010: 00 00 51 E3 00 00 00 0A FA FF FF EA 02 10 A0 E1 ..Q.............
0x00010020: 00 20 A0 E3 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 ................
```

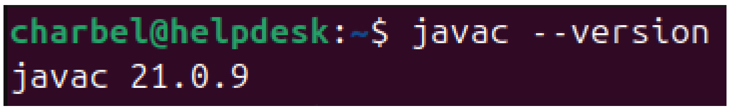
Assignment 4.2: Programming languages

Take screenshots that the following commands work:

javac --version

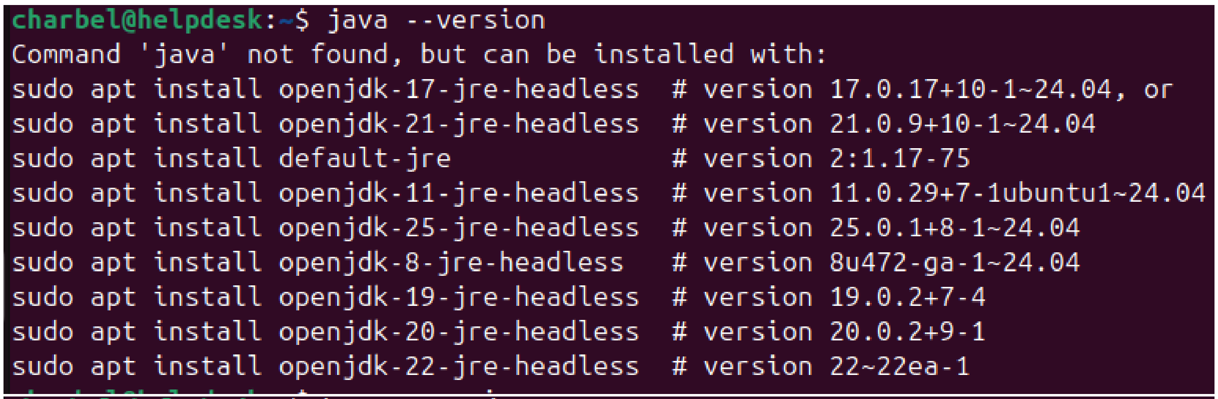


```
charbel@helpdesk:~$ javac version
Command 'javac' not found, but can be installed with:
sudo apt install openjdk-17-jdk-headless  # version 17.0.17+10-1~24.04, or
sudo apt install openjdk-21-jdk-headless  # version 21.0.9+10-1~24.04
sudo apt install default-jdk              # version 2:1.17-75
sudo apt install openjdk-11-jdk-headless  # version 11.0.29+7-1ubuntu1~24.04
sudo apt install openjdk-25-jdk-headless  # version 25.0.1+8-1~24.04
sudo apt install openjdk-8-jdk-headless   # version 8u472-ga-1~24.04
sudo apt install ecj                      # version 3.32.0+eclipse4.26-2
sudo apt install openjdk-19-jdk-headless  # version 19.0.2+7-4
sudo apt install openjdk-20-jdk-headless  # version 20.0.2+9-1
sudo apt install openjdk-22-jdk-headless  # version 22~22ea-1
```

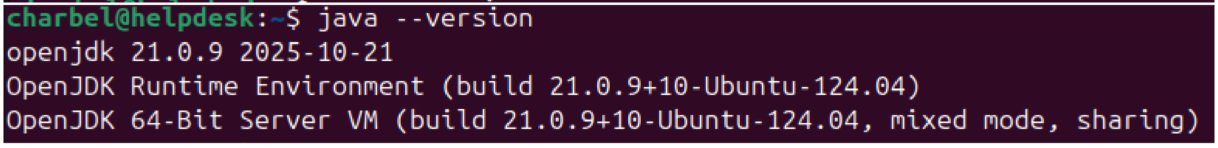


```
charbel@helpdesk:~$ javac --version
javac 21.0.9
```

java --version

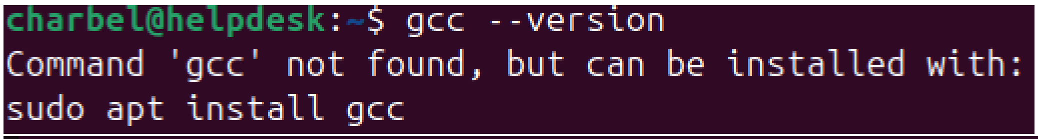


```
charbel@helpdesk:~$ java --version
Command 'java' not found, but can be installed with:
sudo apt install openjdk-17-jre-headless  # version 17.0.17+10-1~24.04, or
sudo apt install openjdk-21-jre-headless  # version 21.0.9+10-1~24.04
sudo apt install default-jre              # version 2:1.17-75
sudo apt install openjdk-11-jre-headless  # version 11.0.29+7-1ubuntu1~24.04
sudo apt install openjdk-25-jre-headless  # version 25.0.1+8-1~24.04
sudo apt install openjdk-8-jre-headless   # version 8u472-ga-1~24.04
sudo apt install openjdk-19-jre-headless  # version 19.0.2+7-4
sudo apt install openjdk-20-jre-headless  # version 20.0.2+9-1
sudo apt install openjdk-22-jre-headless  # version 22~22ea-1
```

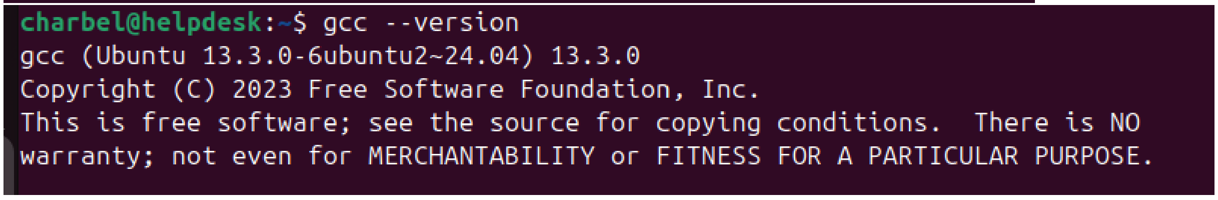


```
charbel@helpdesk:~$ java --version
openjdk 21.0.9 2025-10-21
OpenJDK Runtime Environment (build 21.0.9+10-Ubuntu-124.04)
OpenJDK 64-Bit Server VM (build 21.0.9+10-Ubuntu-124.04, mixed mode, sharing)
```

gcc --version

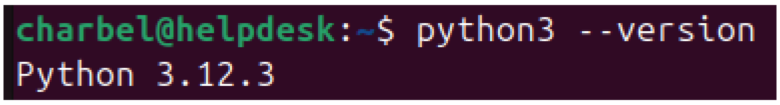


```
charbel@helpdesk:~$ gcc --version
Command 'gcc' not found, but can be installed with:
sudo apt install gcc
```



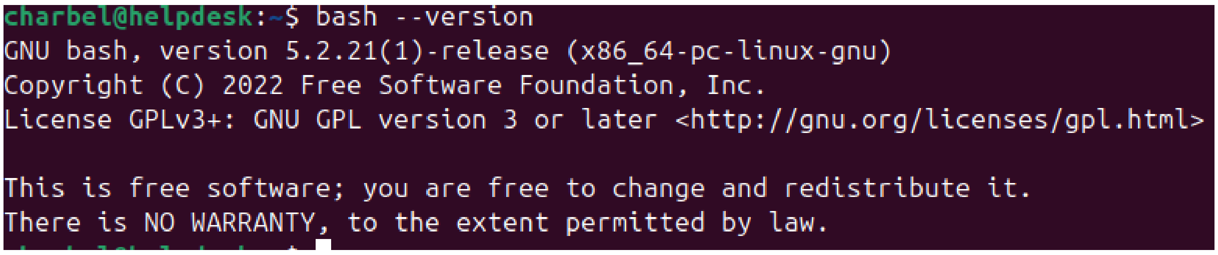
```
charbel@helpdesk:~$ gcc --version
gcc (Ubuntu 13.3.0-6ubuntu2~24.04) 13.3.0
Copyright (C) 2023 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

python3 --version



```
charbel@helpdesk:~$ python3 --version
Python 3.12.3
```

bash --version



```
charbel@helpdesk:~$ 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.
```

Assignment 4.3: Compile

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

Fibonacci.java en fib.c

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

Which source code files are interpreted by an interpreter?

Fib.py en fib.sh

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

Fib.c
Fibonacci.java
fib.py
fib.sh

How do I run a Java program?

java filename

How do I run a Python program?

Pyhton3 filename.py

How do I run a C program?

Gcc fib.c -o fib	compiles naar executable
./fib	run

How do I run a Bash script?

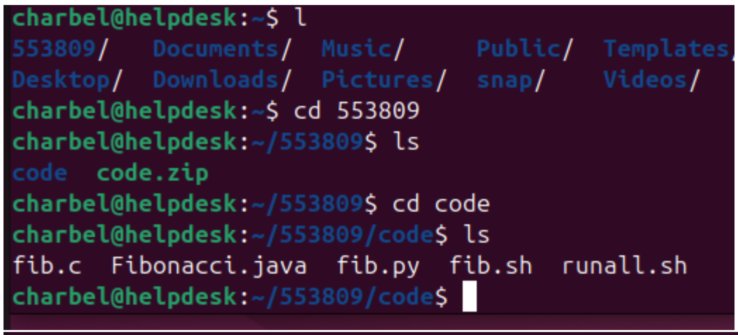
sudo chmod a+x fib.sh
sudo ./fib.sh

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

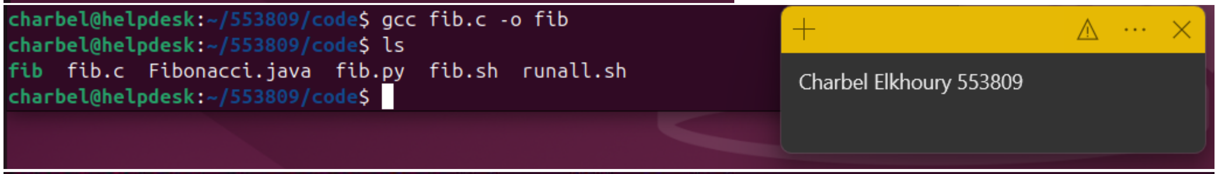
Fibonacci.java naar Fibonacci.class
fib.c naar fib.exe

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?

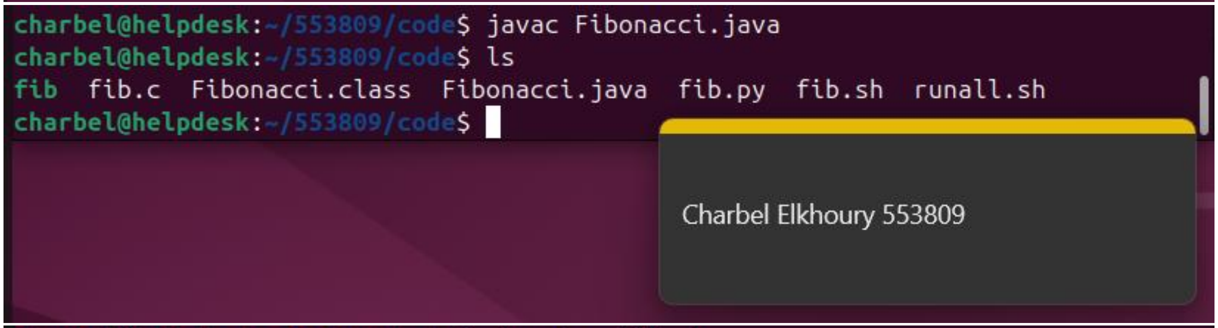


```
charbel@helpdesk:~$ l
553809/   Documents/  Music/     Public/   Templates/
Desktop/  Downloads/  Pictures/  snap/     Videos/
charbel@helpdesk:~$ cd 553809
charbel@helpdesk:~/553809$ ls
code  code.zip
charbel@helpdesk:~/553809$ cd code
charbel@helpdesk:~/553809/code$ ls
fib.c  Fibonacci.java  fib.py  fib.sh  runall.sh
charbel@helpdesk:~/553809/code$ 
```



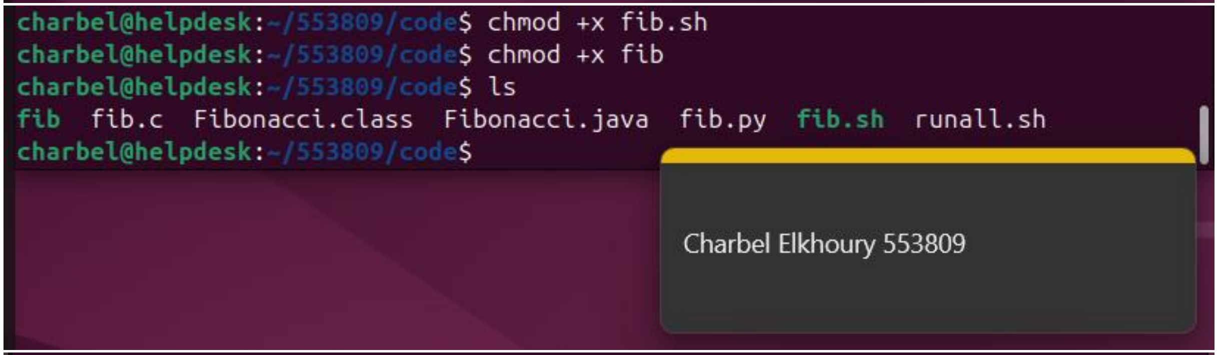
```
charbel@helpdesk:~/553809/code$ gcc fib.c -o fib
charbel@helpdesk:~/553809/code$ ls
fib  fib.c  Fibonacci.java  fib.py  fib.sh  runall.sh
charbel@helpdesk:~/553809/code$ 
```

Charbel Elkhoury 553809



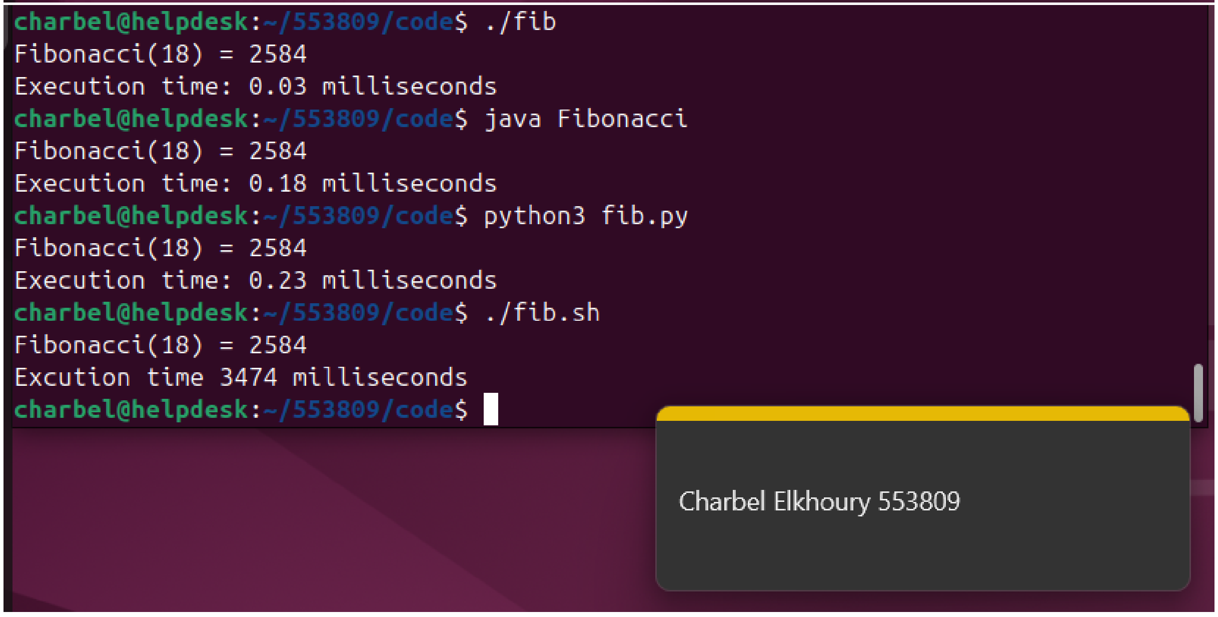
```
charbel@helpdesk:~/553809/code$ javac Fibonacci.java
charbel@helpdesk:~/553809/code$ ls
fib  fib.c  Fibonacci.class  Fibonacci.java  fib.py  fib.sh  runall.sh
charbel@helpdesk:~/553809/code$ 
```

Charbel Elkhoury 553809



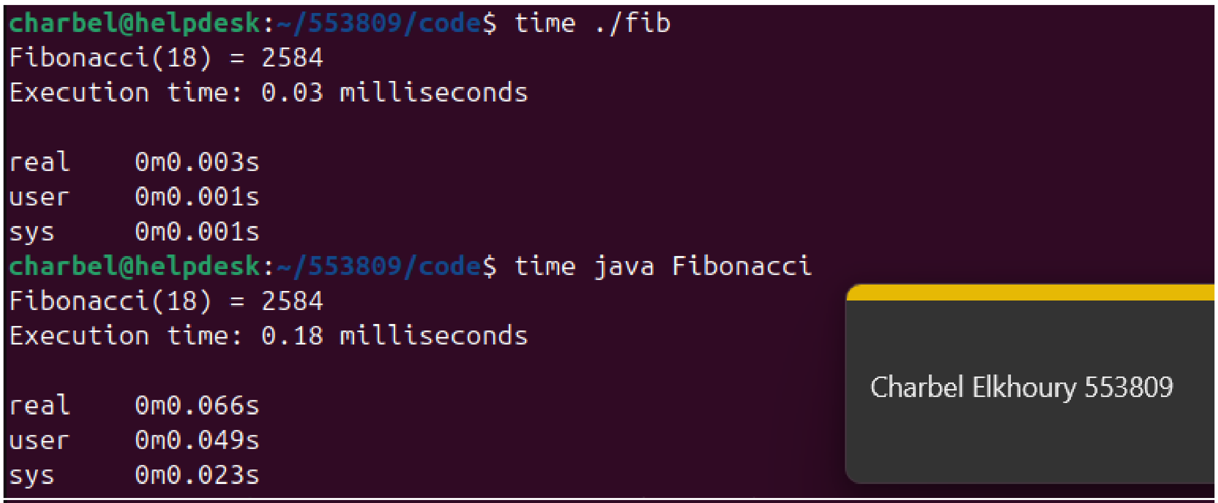
```
charbel@helpdesk:~/553809/code$ chmod +x fib.sh
charbel@helpdesk:~/553809/code$ chmod +x fib
charbel@helpdesk:~/553809/code$ ls
fib  fib.c  Fibonacci.class  Fibonacci.java  fib.py  fib.sh  runall.sh
charbel@helpdesk:~/553809/code$
```

Charbel Elkhoury 553809

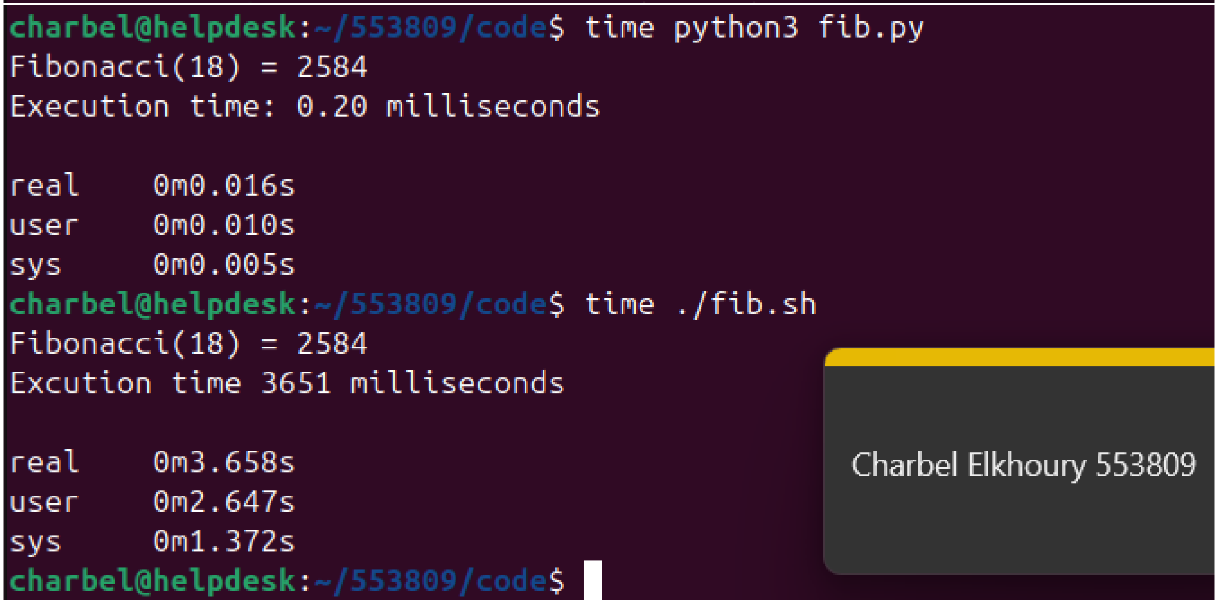


```
charbel@helpdesk:~/553809/code$ ./fib
Fibonacci(18) = 2584
Execution time: 0.03 milliseconds
charbel@helpdesk:~/553809/code$ java Fibonacci
Fibonacci(18) = 2584
Execution time: 0.18 milliseconds
charbel@helpdesk:~/553809/code$ python3 fib.py
Fibonacci(18) = 2584
Execution time: 0.23 milliseconds
charbel@helpdesk:~/553809/code$ ./fib.sh
Fibonacci(18) = 2584
Excution time 3474 milliseconds
charbel@helpdesk:~/553809/code$ 
```

Charbel Elkhoury 553809



```
charbel@helpdesk:~/553809/code$ time ./fib
Fibonacci(18) = 2584
Execution time: 0.03 milliseconds

real	0m0.003s
user	0m0.001s
sys	0m0.001s
charbel@helpdesk:~/553809/code$ time java Fibonacci
Fibonacci(18) = 2584
Execution time: 0.18 milliseconds

real	0m0.066s
user	0m0.049s
sys	0m0.023s
```



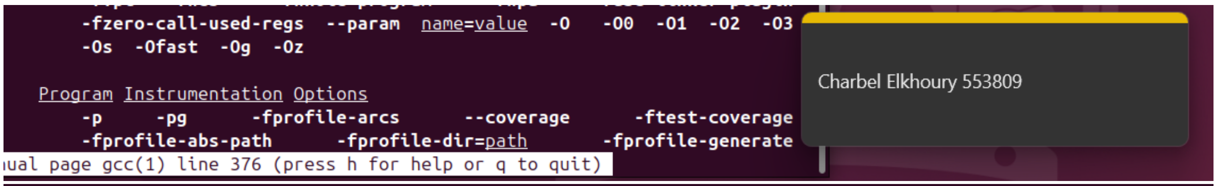
```
charbel@helpdesk:~/553809/code$ time python3 fib.py
Fibonacci(18) = 2584
Execution time: 0.20 milliseconds

real	0m0.016s
user	0m0.010s
sys	0m0.005s
charbel@helpdesk:~/553809/code$ time ./fib.sh
Fibonacci(18) = 2584
Excution time 3651 milliseconds

real	0m3.658s
user	0m2.647s
sys	0m1.372s
charbel@helpdesk:~/553809/code$
```

1. C
2. Java
3. Python
4. Bash

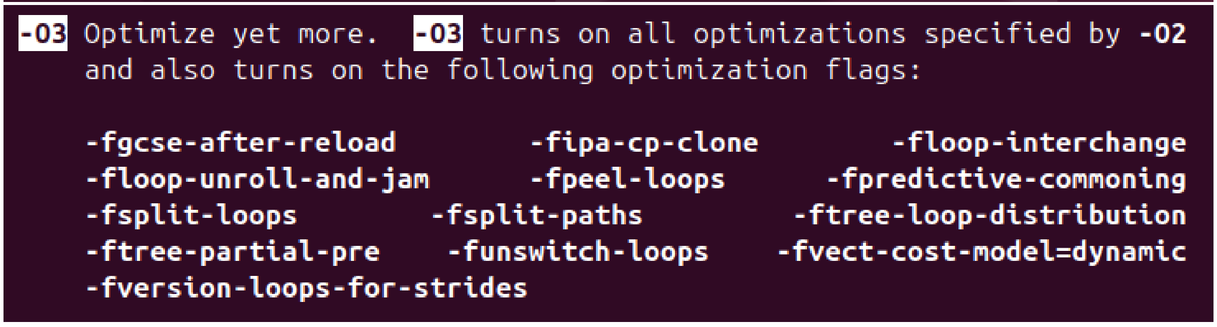
Assignment 4.4: Optimize

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.

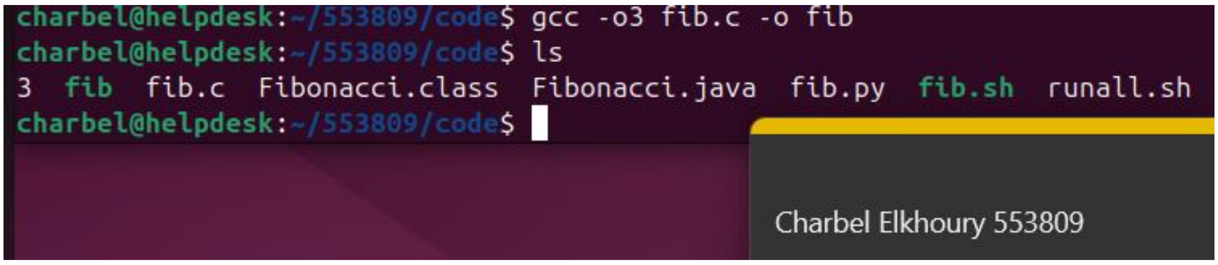


```
       -fzero-call-used-regs  --param  name=value  -O   -O0  -O1  -O2  -O3
       -Os  -Ofast  -Og  -Oz

   Program Instrumentation Options
       -p      -pg      -fprofile-arcs      --coverage      -ftest-coverage
       -fprofile-abs-path       -fprofile-dir=path       -fprofile-generate
ual page gcc(1) line 376 (press h for help or q to quit)
```



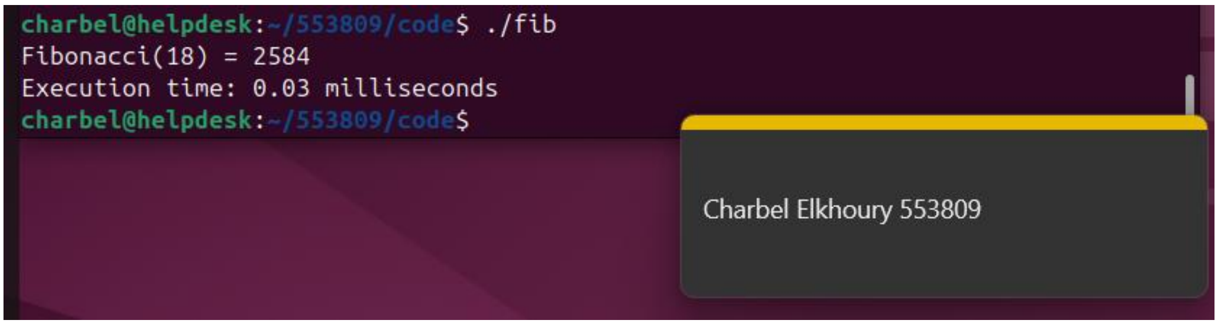
```
-O3 Optimize yet more.  -O3 turns on all optimizations specified by -O2
    and also turns on the following optimization flags:

    -fgcse-after-reload         -fipa-cp-clone         -floop-interchange
    -floop-unroll-and-jam       -fpeel-loops        -fpredictive-commoning
    -fsplit-loops            -fsplit-paths          -ftree-loop-distribution
    -ftree-partial-pre        -funswitch-loops     -fvect-cost-model=dynamic
    -fversion-loops-for-strides
```

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



```
charbel@helpdesk:~/553809/code$ gcc -o3 fib.c -o fib
charbel@helpdesk:~/553809/code$ ls
3  fib  fib.c  Fibonacci.class  Fibonacci.java  fib.py  fib.sh  runall.sh
charbel@helpdesk:~/553809/code$ 
```

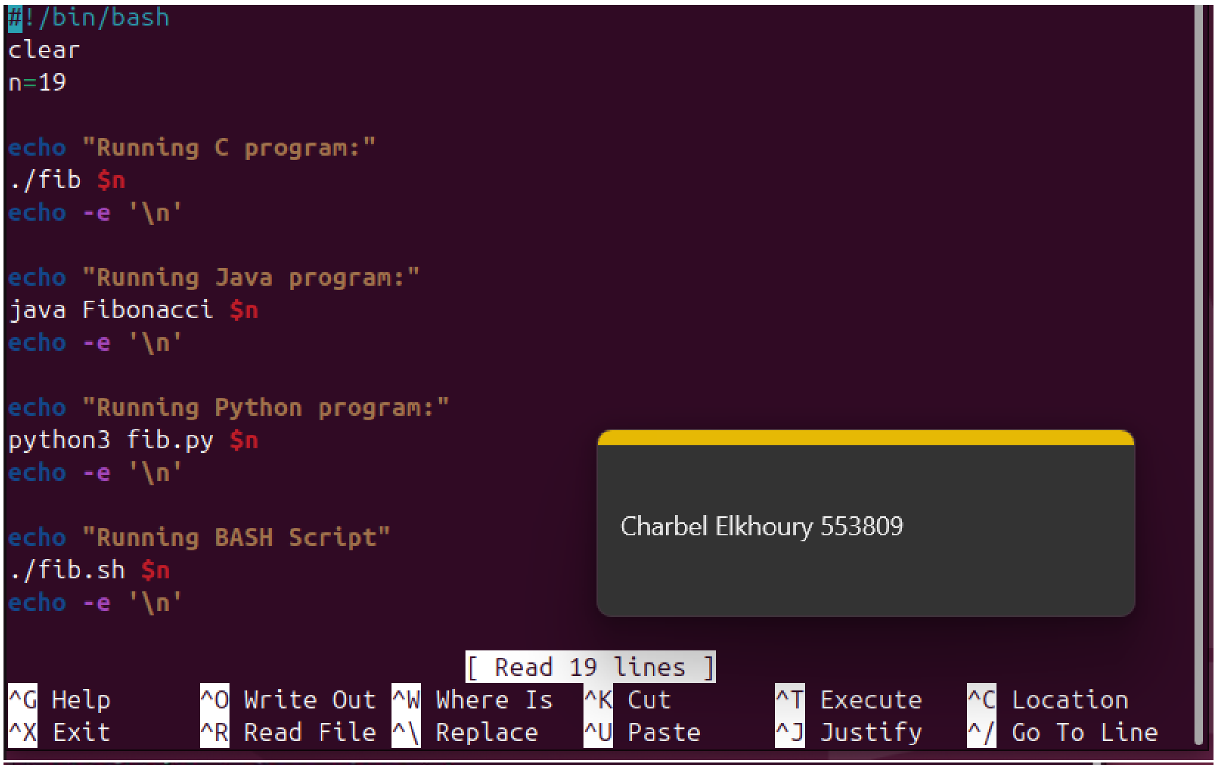
c) Run the newly compiled program. Is it true that it now performs the calculation faster?



```
charbel@helpdesk:~/553809/code$ ./fib
Fibonacci(18) = 2584
Execution time: 0.03 milliseconds
charbel@helpdesk:~/553809/code$
```

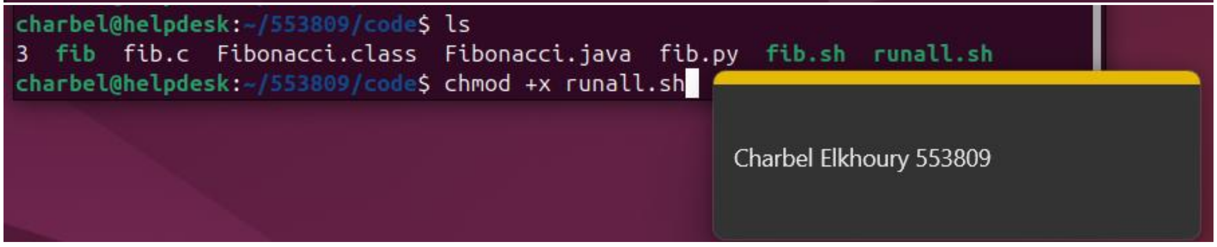
Nee is het zelfde

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.



```
#!/bin/bash
clear
n=19

echo "Running C program:"
./fib $n
echo -e '\n'

echo "Running Java program:"
java Fibonacci $n
echo -e '\n'

echo "Running Python program:"
python3 fib.py $n
echo -e '\n'

echo "Running BASH Script"
./fib.sh $n
echo -e '\n'

                    [ Read 19 lines ]
^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line
```

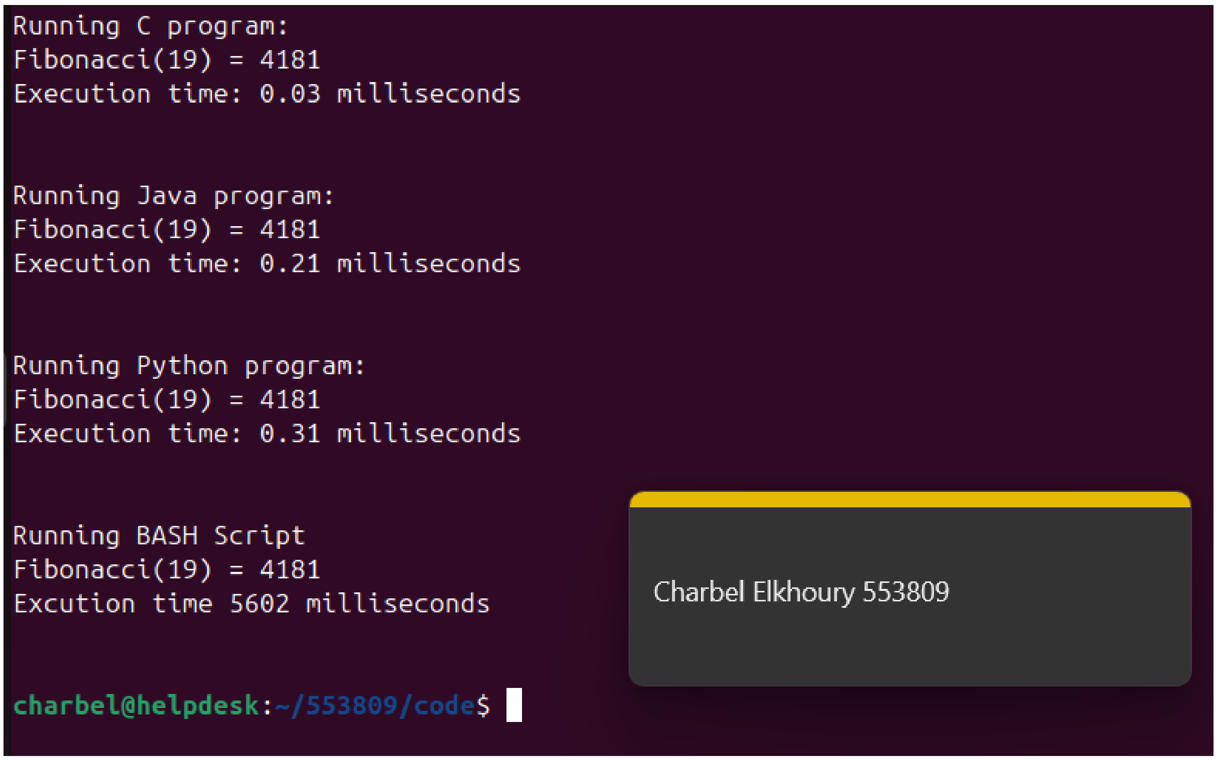
Charbel Elkhoury 553809



```
charbel@helpdesk:~/553809/code$ ls
3  fib  fib.c  Fibonacci.class  Fibonacci.java  fib.py  fib.sh  runall.sh
charbel@helpdesk:~/553809/code$ chmod +x runall.sh
```

Charbel Elkhoury 553809

chmod +x runall.sh als eerst gedaan zodat het executable is



```
Running C program:
Fibonacci(19) = 4181
Execution time: 0.03 milliseconds


Running Java program:
Fibonacci(19) = 4181
Execution time: 0.21 milliseconds


Running Python program:
Fibonacci(19) = 4181
Execution time: 0.31 milliseconds


Running BASH Script
Fibonacci(19) = 4181
Excution time 5602 milliseconds


charbel@helpdesk:~/553809/code$
```

Charbel Elkhoury 553809

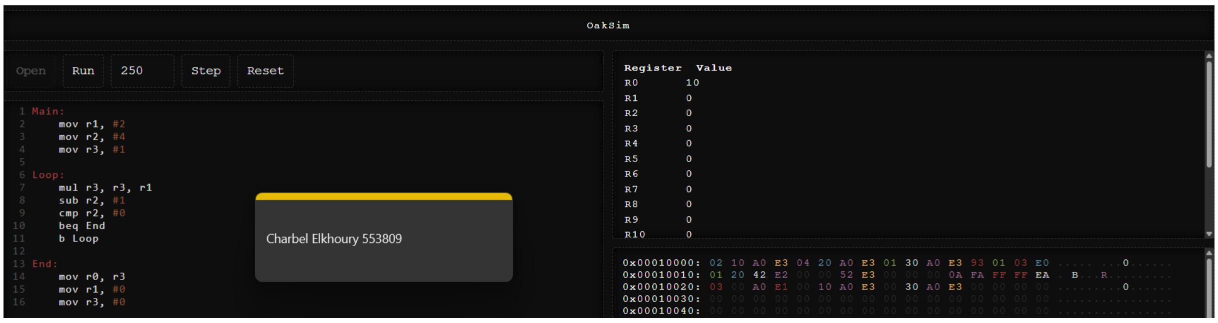
Assignment 4.5: More ARM Assembly

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.



```
Main:
    mov r1, #2
    mov r2, #4
    mov r3, #1

Loop:
    mul r3, r3, r1
    sub r2, #1
    cmp r2, #0
    beq End
    b Loop

End:
    mov r0, r3
    mov r1, #0
    mov r3, #0
```

Charbel Elkhoury 553809

```
Register  Value
R0        10
R1        0
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 30 A0 E3 93 01 03 E0 ..........0......
0x00010010: 01 20 42 E2 00 00 52 E3 00 00 00 0A FA FF FF EA . B...R..........
0x00010020: 03 00 A0 E1 00 10 A0 E3 00 30 A0 E3 00 00 00 00 ..........0......
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 ................
```

Het is 10 vanwege hexadecimaal zat hier een beetje vast mee want dacht dat het niet verder ging dan 10 maar HEX 10 is DEC 16

Ready? Save this file and export it as a pdf file with the name: **week4.pdf**