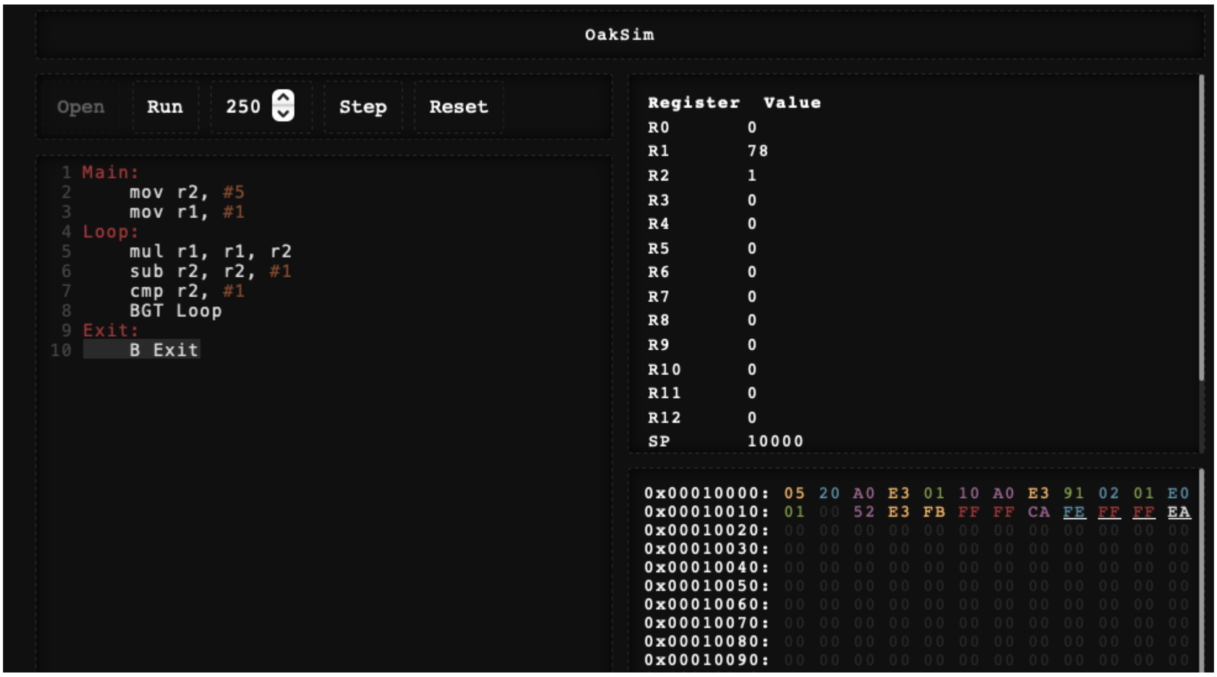
Week 4 – Software

Student number: 564530

Assignment 4.1: ARM assembly

Screenshot of working assembly code of factorial calculation:



```
Main:
      mov r2, #5
      mov r1, #1
Loop:
      mul r1, r1, r2
      sub r2, r2, #1
      cmp r2, #1
      BGT Loop
Exit:
      B Exit
```

```
Register  Value
R0        0
R1        78
R2        1
R3        0
R4        0
R5        0
R6        0
R7        0
R8        0
R9        0
R10       0
R11       0
R12       0
SP        10000
```

```
0x00010000: 05 20 A0 E3 01 10 A0 E3 91 02 01 E0
0x00010010: 01 00 52 E3 FB FF FF CA FE FF FF EA
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
```

Assignment 4.2: Programming languages

Take screenshots that the following commands work:

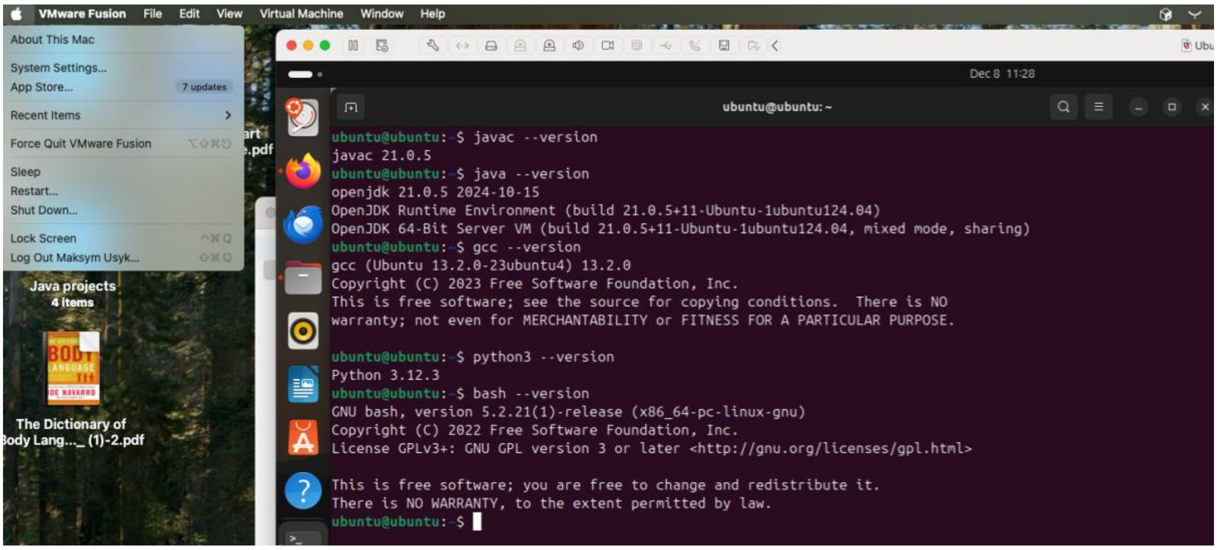
javac --version

java --version

gcc --version

python3 --version

bash --version



```
ubuntu@ubuntu:~$ javac --version
javac 21.0.5
ubuntu@ubuntu:~$ java --version
openjdk 21.0.5 2024-10-15
OpenJDK Runtime Environment (build 21.0.5+11-Ubuntu-1ubuntu124.04)
OpenJDK 64-Bit Server VM (build 21.0.5+11-Ubuntu-1ubuntu124.04, mixed mode, sharing)
ubuntu@ubuntu:~$ gcc --version
gcc (Ubuntu 13.2.0-23ubuntu4) 13.2.0
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.

ubuntu@ubuntu:~$ python3 --version
Python 3.12.3
ubuntu@ubuntu:~$ 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.
ubuntu@ubuntu:~$
```

Assignment 4.3: Compile

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

Fibonacci.java, 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, 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

How do I run a Java program?

javac and then java <file>

How do I run a Python program?

python3

How do I run a C program?

gcc -o <file name in future> <file.name> and then ./<filename>

How do I run a Bash script?

First make it executable

sudo chmod a+x <file.name>

Then run it

./<file.name>

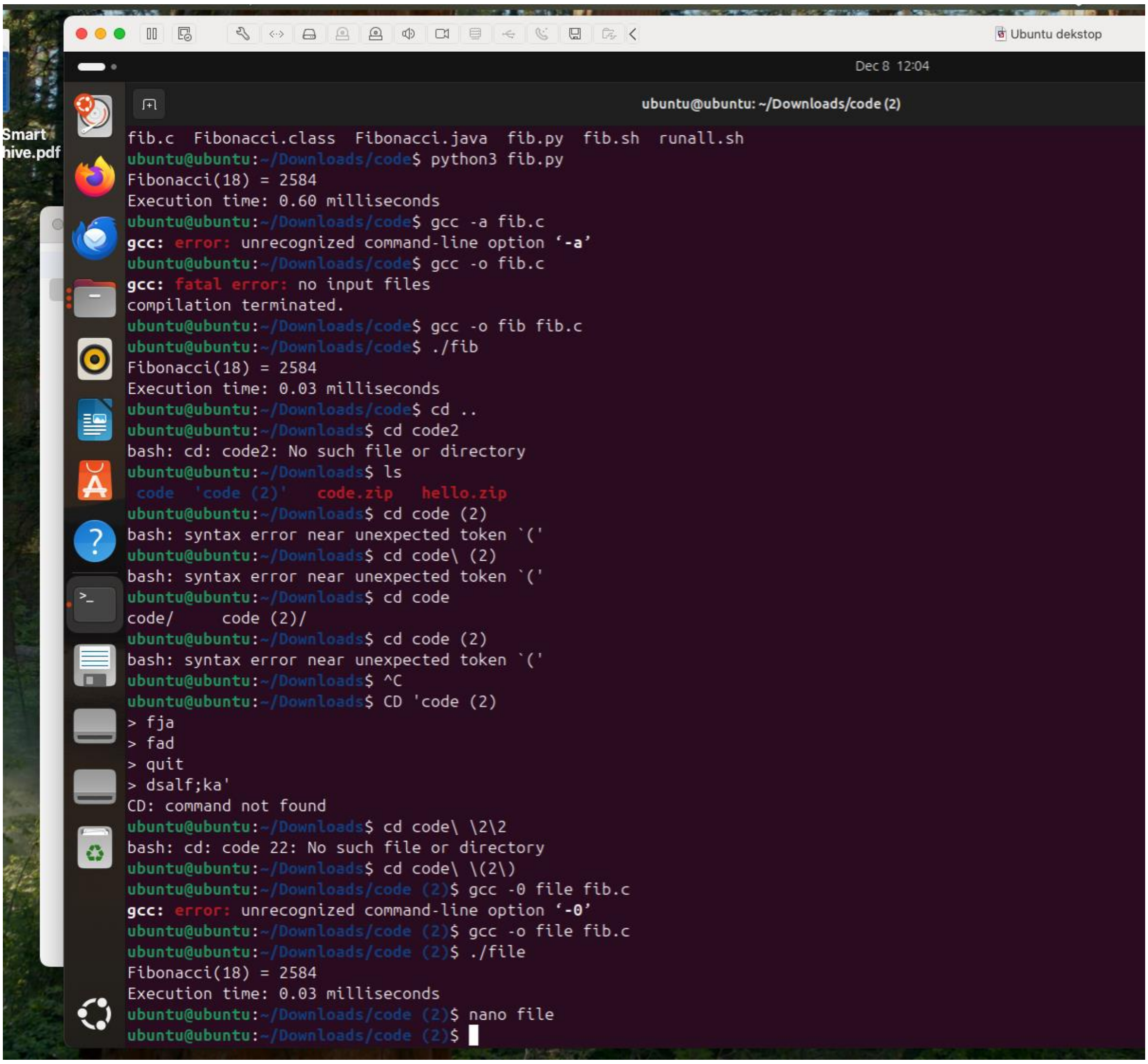
Will a new file be created if I compile the above source code? If so, which file?

When working with java, yes when you ran a java program for the first time, it is just a source code, you have to execute it and convert to bytecode by a command javac, and only then you can make use of a virtual machine JVM. In addition, with the c files will happen almost the same, however you will just compile c source code into a file that can be directly executed by a processor.

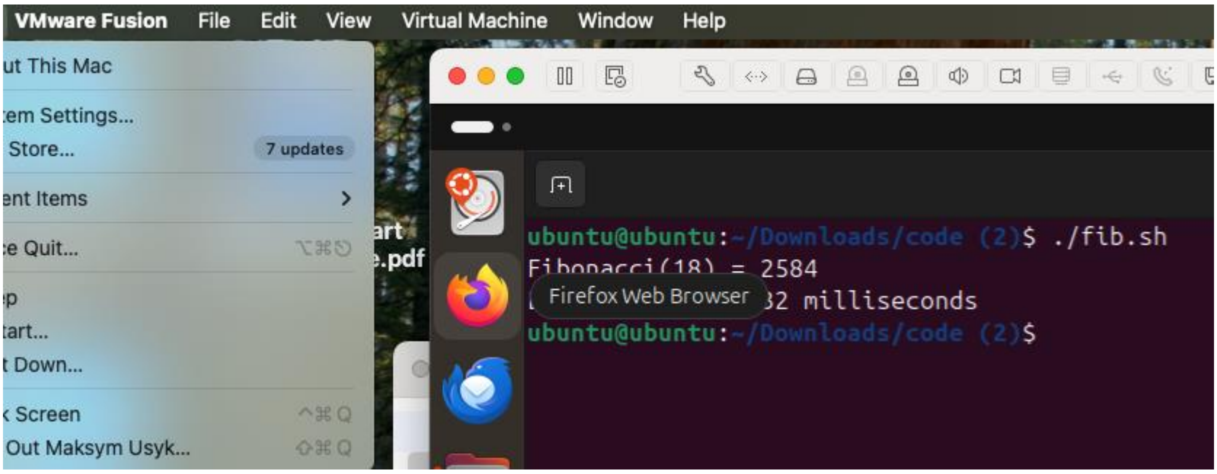
Take relevant screenshots of the following commands:

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

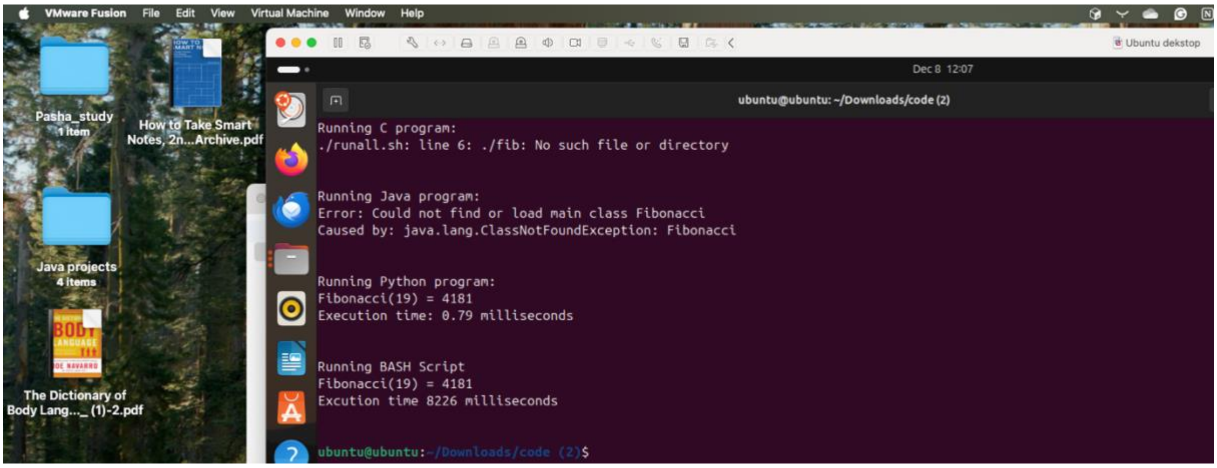
fib.c



```
fib.c  Fibonacci.class  Fibonacci.java  fib.py  fib.sh  runall.sh
ubuntu@ubuntu:~/Downloads/code$ python3 fib.py
Fibonacci(18) = 2584
Execution time: 0.60 milliseconds
ubuntu@ubuntu:~/Downloads/code$ gcc -a fib.c
gcc: error: unrecognized command-line option ‘-a’
ubuntu@ubuntu:~/Downloads/code$ gcc -o fib.c
gcc: fatal error: no input files
compilation terminated.
ubuntu@ubuntu:~/Downloads/code$ gcc -o fib fib.c
ubuntu@ubuntu:~/Downloads/code$ ./fib
Fibonacci(18) = 2584
Execution time: 0.03 milliseconds
ubuntu@ubuntu:~/Downloads/code$ cd ..
ubuntu@ubuntu:~/Downloads$ cd code2
bash: cd: code2: No such file or directory
ubuntu@ubuntu:~/Downloads$ ls
 code  'code (2)'   code.zip   hello.zip
ubuntu@ubuntu:~/Downloads$ cd code (2)
bash: syntax error near unexpected token `('
ubuntu@ubuntu:~/Downloads$ cd code\ (2)
bash: syntax error near unexpected token `('
ubuntu@ubuntu:~/Downloads$ cd code
code/      code (2)/
ubuntu@ubuntu:~/Downloads$ cd code (2)
bash: syntax error near unexpected token `('
ubuntu@ubuntu:~/Downloads$ ^C
ubuntu@ubuntu:~/Downloads$ CD 'code (2)
> fja
> fad
> quit
> dsalf;ka'
CD: command not found
ubuntu@ubuntu:~/Downloads$ cd code\ \2\2
bash: cd: code 22: No such file or directory
ubuntu@ubuntu:~/Downloads$ cd code\ \(2\)
ubuntu@ubuntu:~/Downloads/code (2)$ gcc -0 file fib.c
gcc: error: unrecognized command-line option ‘-0’
ubuntu@ubuntu:~/Downloads/code (2)$ gcc -o file fib.c
ubuntu@ubuntu:~/Downloads/code (2)$ ./file
Fibonacci(18) = 2584
Execution time: 0.03 milliseconds
ubuntu@ubuntu:~/Downloads/code (2)$ nano file
ubuntu@ubuntu:~/Downloads/code (2)$ 
```



```
ubuntu@ubuntu:~/Downloads/code (2)$ ./fib.sh
Fibonacci(18) = 2584
32 milliseconds
ubuntu@ubuntu:~/Downloads/code (2)$
```



VMware Fusion
File
Edit
View
Virtual Machine
Window
Help
Pasha_study
1 item
How to Take Smart Notes, 2n...Archive.pdf
Java projects
4 items
The Dictionary of Body Lang..._ (1)-2.pdf
Ubuntu dekstop
Dec 8 12:07
ubuntu@ubuntu: ~/Downloads/code (2)

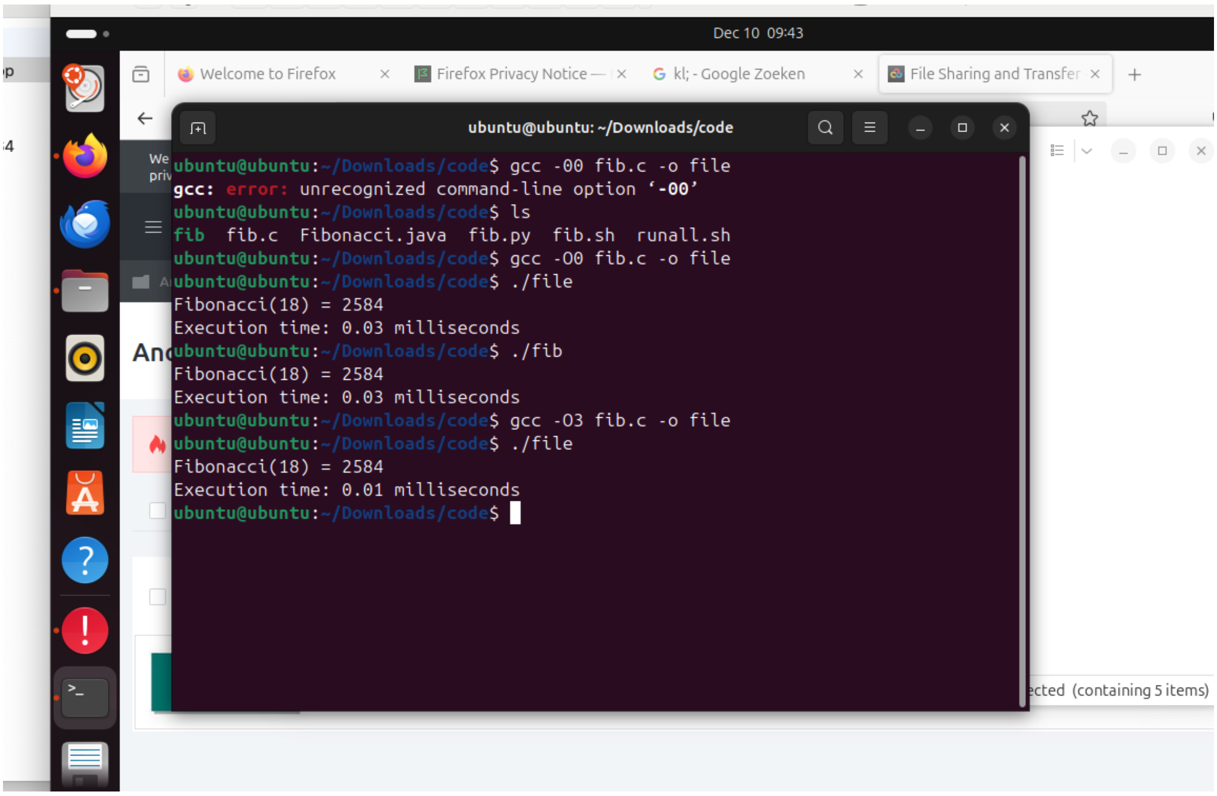
```
Running C program:
./runall.sh: line 6: ./fib: No such file or directory


Running Java program:
Error: Could not find or load main class Fibonacci
Caused by: java.lang.ClassNotFoundException: Fibonacci


Running Python program:
Fibonacci(19) = 4181
Execution time: 0.79 milliseconds


Running BASH Script
Fibonacci(19) = 4181
Excution time 8226 milliseconds


ubuntu@ubuntu:~/Downloads/code (2)$
```

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) ompile **fib.c** again with the optimization parameters



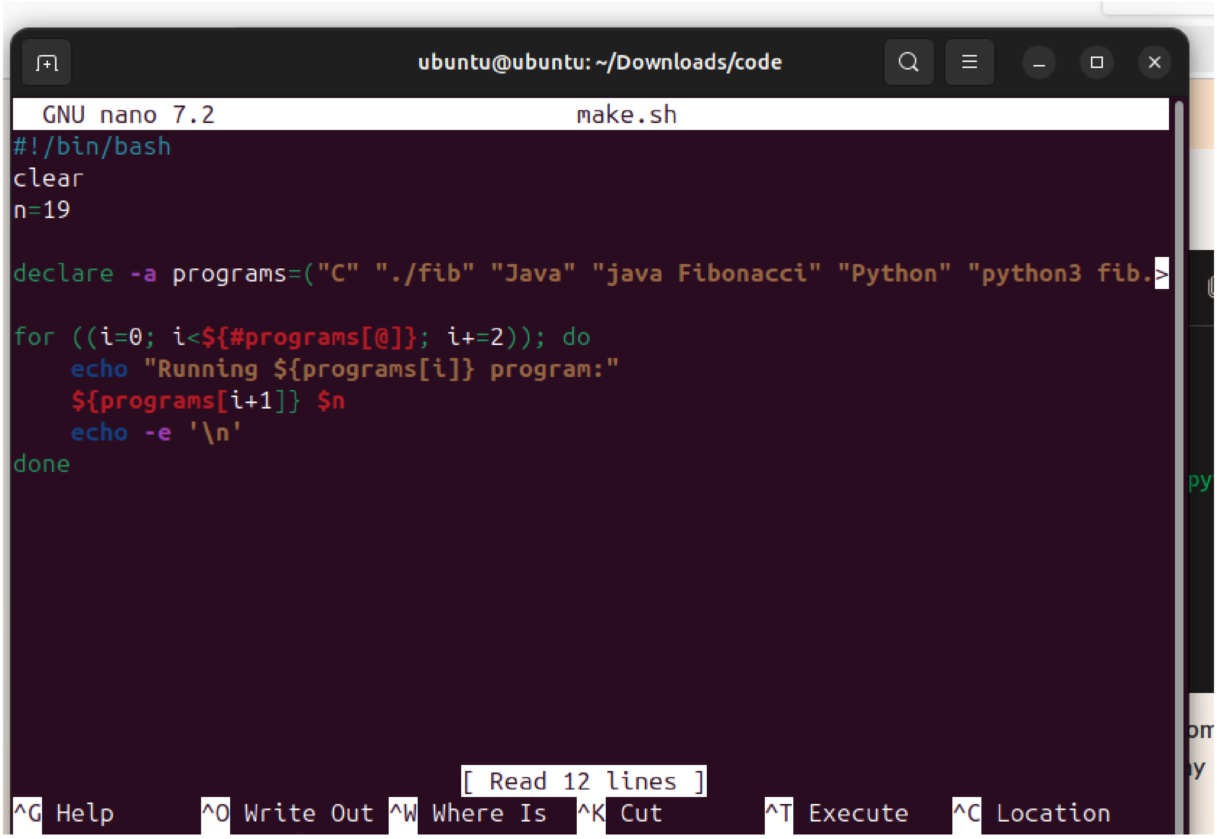
Dec 10 09:43
Welcome to Firefox
Firefox Privacy Notice —
kl; - Google Zoeken
File Sharing and Transfer
ubuntu@ubuntu: ~/Downloads/code
cted (containing 5 items)

```
ubuntu@ubuntu:~/Downloads/code$ gcc -O0 fib.c -o file
gcc: error: unrecognized command-line option ‘-00’
ubuntu@ubuntu:~/Downloads/code$ ls
fib  fib.c  Fibonacci.java  fib.py  fib.sh  runall.sh
ubuntu@ubuntu:~/Downloads/code$ gcc -O0 fib.c -o file
ubuntu@ubuntu:~/Downloads/code$ ./file
Fibonacci(18) = 2584
Execution time: 0.03 milliseconds
ubuntu@ubuntu:~/Downloads/code$ ./fib
Fibonacci(18) = 2584
Execution time: 0.03 milliseconds
ubuntu@ubuntu:~/Downloads/code$ gcc -O3 fib.c -o file
ubuntu@ubuntu:~/Downloads/code$ ./file
Fibonacci(18) = 2584
Execution time: 0.01 milliseconds
ubuntu@ubuntu:~/Downloads/code$
```

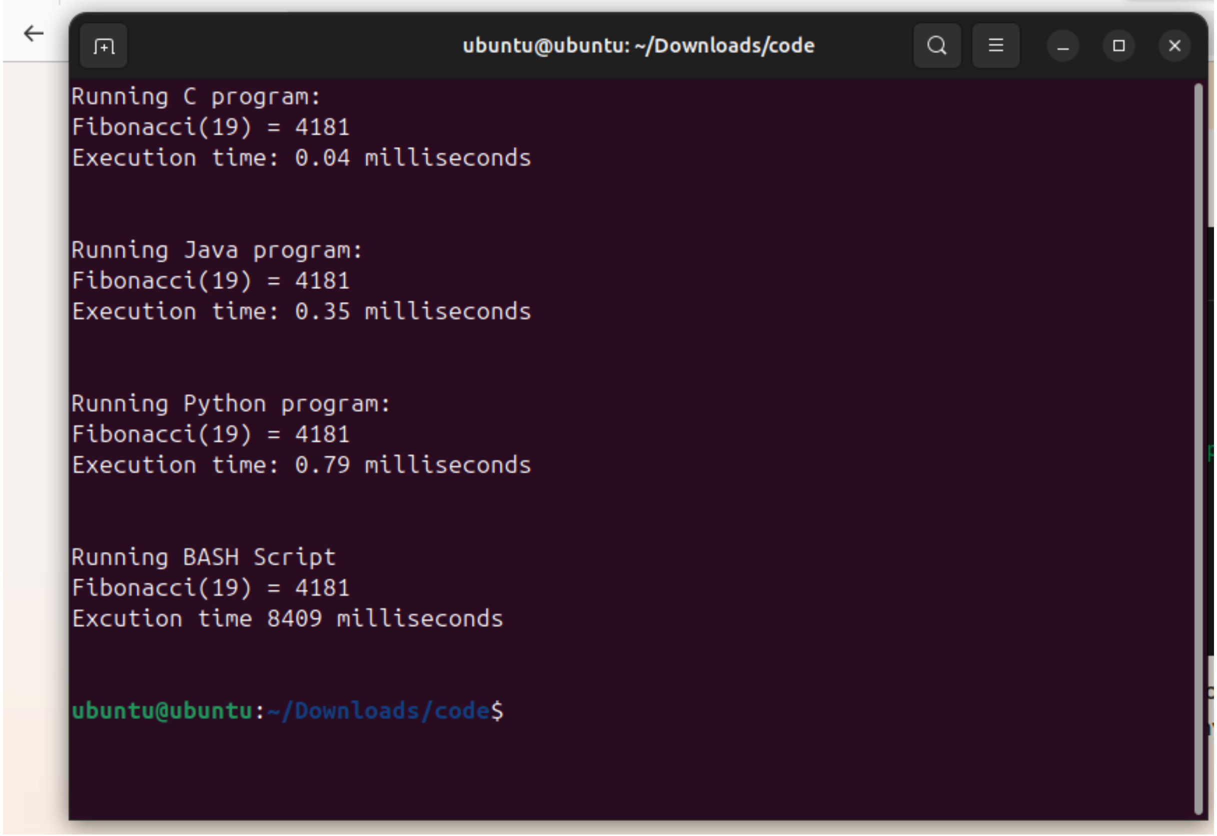
c) Run the newly compiled program. Is it true that it now performs the calculation faster?

Yes

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.



```
  GNU nano 7.2                          make.sh
#!/bin/bash
clear
n=19

declare -a programs=("C" "./fib" "Java" "java Fibonacci" "Python" "python3 fib.>

for ((i=0; i<${#programs[@]}; i+=2)); do
    echo "Running ${programs[i]} program:"
    ${programs[i+1]} $n
    echo -e '\n'
done




                                [ Read 12 lines ]
^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location
```



```
Running C program:
Fibonacci(19) = 4181
Execution time: 0.04 milliseconds


Running Java program:
Fibonacci(19) = 4181
Execution time: 0.35 milliseconds


Running Python program:
Fibonacci(19) = 4181
Execution time: 0.79 milliseconds


Running BASH Script
Fibonacci(19) = 4181
Excution time 8409 milliseconds


ubuntu@ubuntu:~/Downloads/code$
```

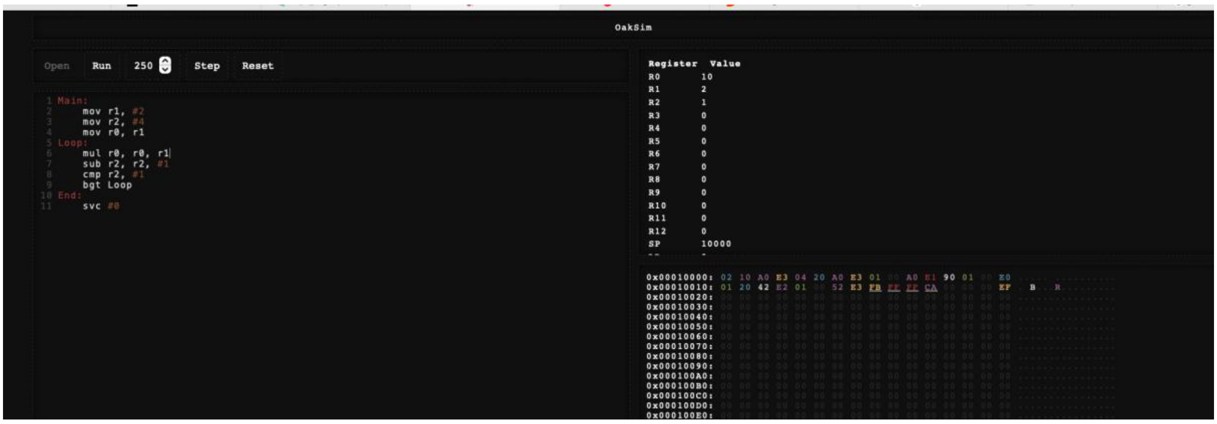
Bonus point assignment – week 4

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 r1, #2
        mov r2, #4
        mov r0, r1
Loop:
        mul r0, r0, r1
        sub r2, r2, #1
        cmp r2, #1
        bgt Loop
```

End:

    svc #0



```
Main:
    mov r1, #2
    mov r2, #4
    mov r0, r1
Loop:
    mul r0, r0, r1
    sub r2, r2, #1
    cmp r2, #1
    bgt Loop
End:
    svc #0
```

```
Register  Value
R0        10
R1        2
R2        1
R3        0
R4        0
R5        0
R6        0
R7        0
R8        0
R9        0
R10       0
R11       0
R12       0
SP        10000
```

```
0x00010000: 02 10 A0 E3 04 20 A0 E3 01 00 A0 E1 90 01 00 E0 ................
0x00010010: 01 20 42 E2 01 00 52 E3 FB FF FF CA 00 00 00 EF . B...R.........
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 ................
```