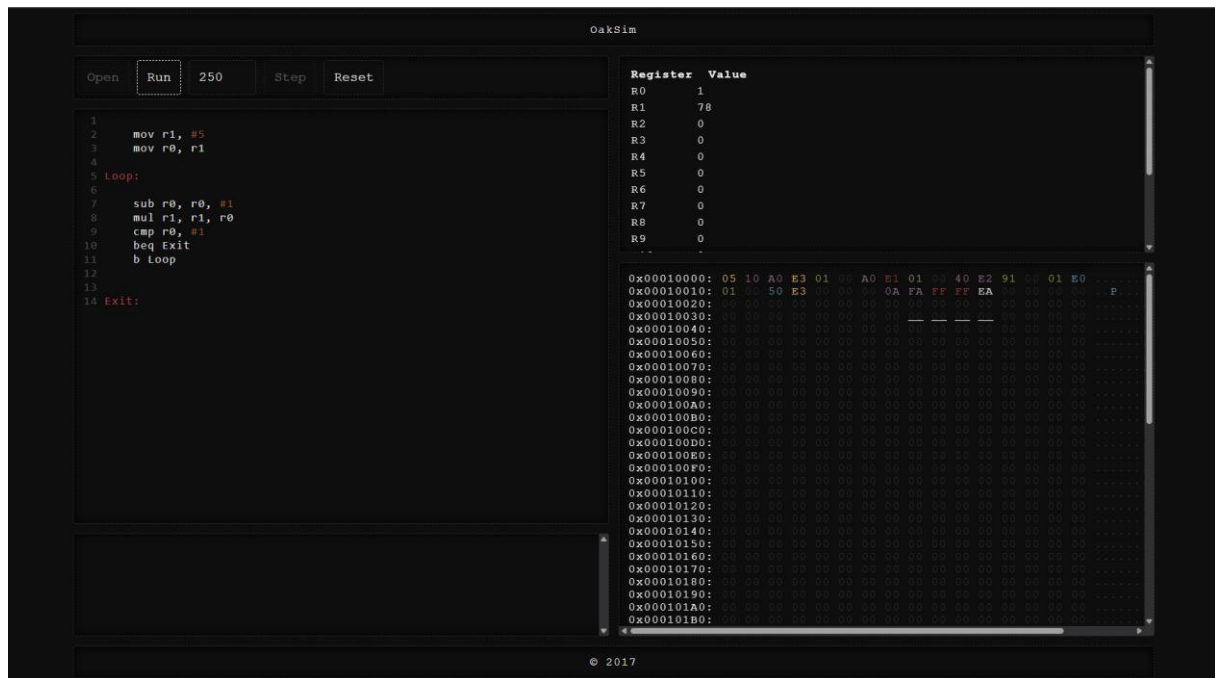


Template Week 4 – Software

Student number: 578688

Assignment 4.1: ARM assembly

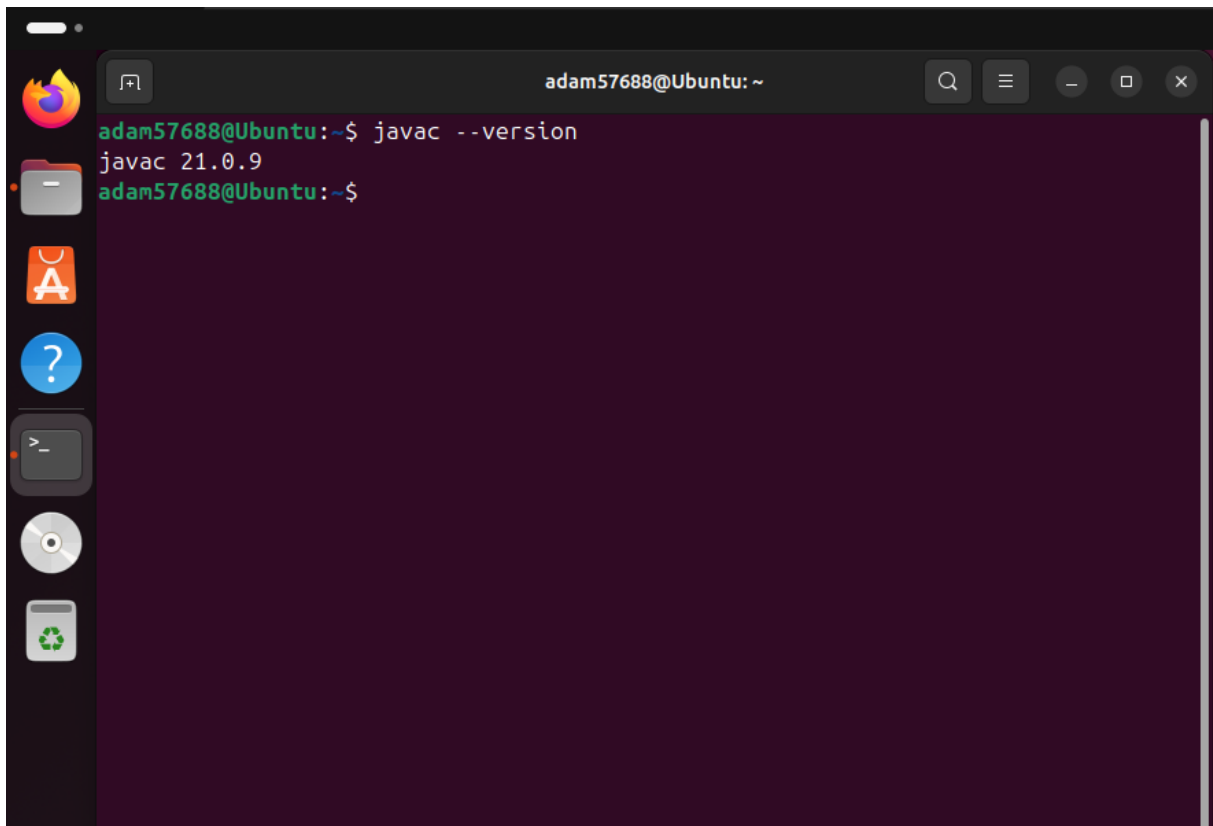
Screenshot of working assembly code of factorial calculation:



Assignment 4.2: Programming languages

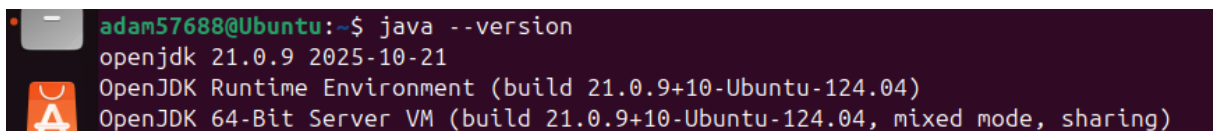
Take screenshots that the following commands work:

`javac --version`

A terminal window titled 'adam57688@Ubuntu: ~' with search, menu, and window control buttons. The terminal shows the command 'javac --version' being executed, resulting in the output 'javac 21.0.9'. The prompt 'adam57688@Ubuntu:~\$' is visible on the line following the output. The terminal has a dark purple background and a sidebar on the left with icons for applications, help, and system utilities.

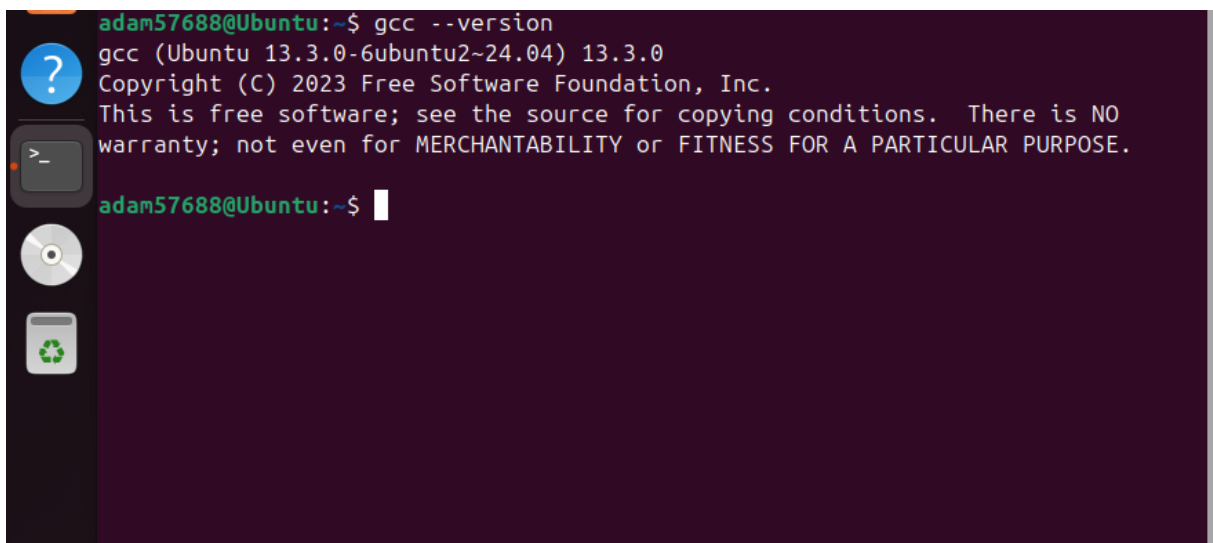
```
adam57688@Ubuntu:~$ javac --version
javac 21.0.9
adam57688@Ubuntu:~$
```

`java --version`

A terminal window showing the command 'java --version' being executed. The output displays the OpenJDK version and build information: 'openjdk 21.0.9 2025-10-21', 'OpenJDK Runtime Environment (build 21.0.9+10-Ubuntu-124.04)', and 'OpenJDK 64-Bit Server VM (build 21.0.9+10-Ubuntu-124.04, mixed mode, sharing)'. The prompt 'adam57688@Ubuntu:~\$' is visible at the end of the output line.

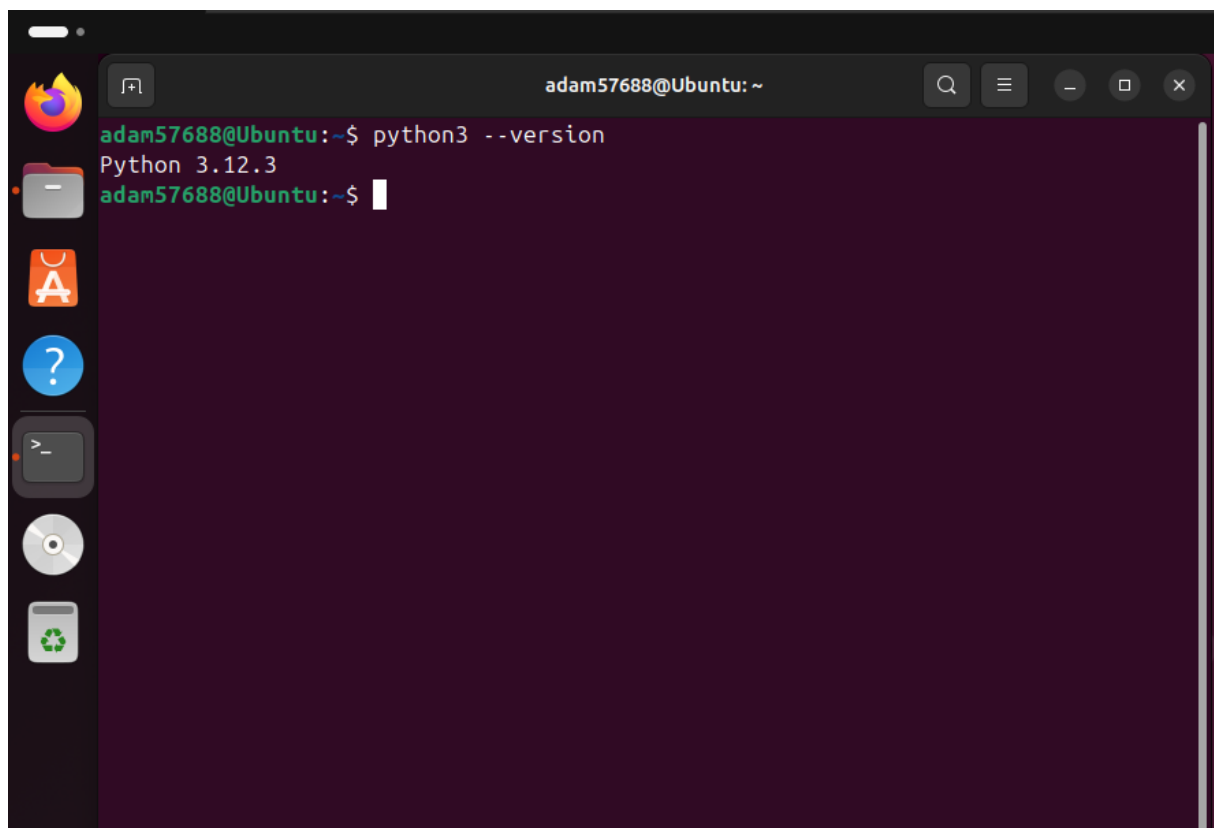
```
adam57688@Ubuntu:~$ java --version
openjdk 21.0.9 2025-10-21
OpenJDK Runtime Environment (build 21.0.9+10-Ubuntu-124.04)
OpenJDK 64-Bit Server VM (build 21.0.9+10-Ubuntu-124.04, mixed mode, sharing)
adam57688@Ubuntu:~$
```

`gcc --version`

A terminal window showing the command 'gcc --version' being executed. The output displays the GCC version and copyright information: 'gcc (Ubuntu 13.3.0-6ubuntu2~24.04) 13.3.0', 'Copyright (C) 2023 Free Software Foundation, Inc.', and a disclaimer: 'This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.' The prompt 'adam57688@Ubuntu:~\$' is visible at the end of the output line.

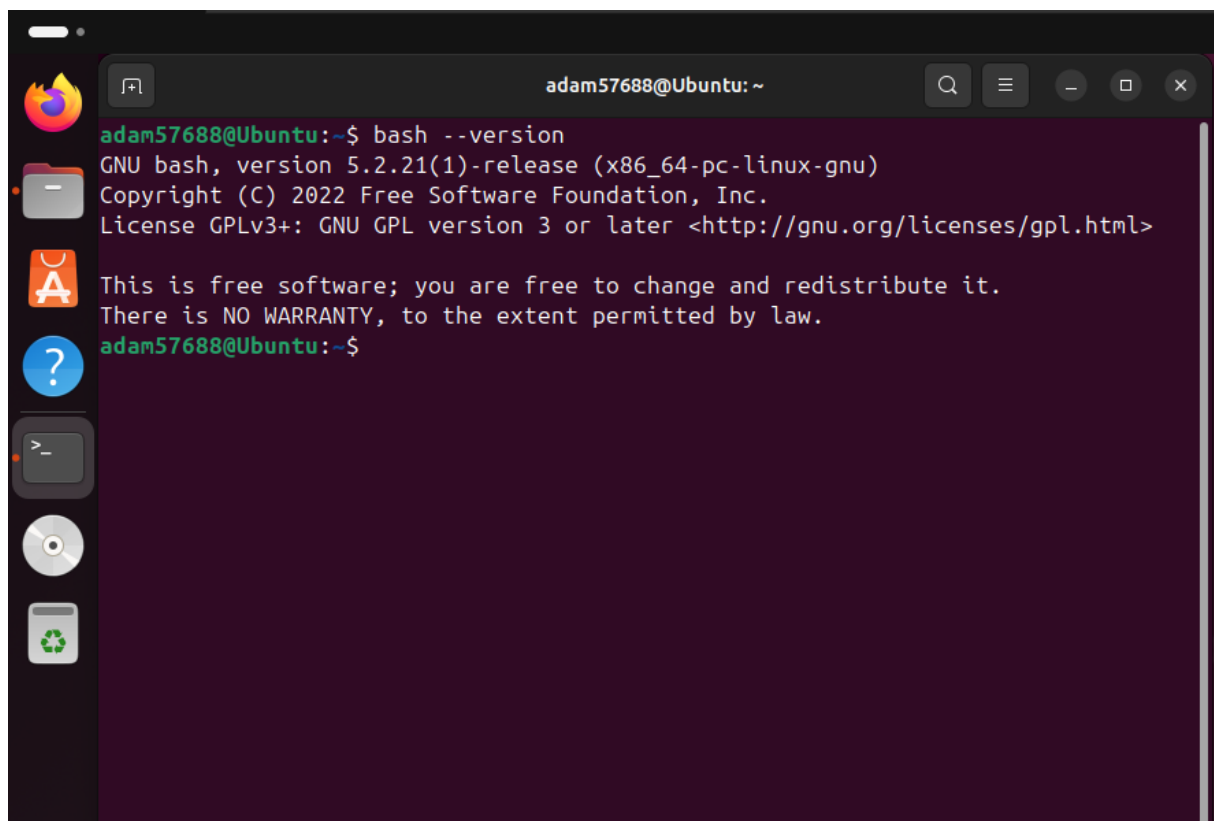
```
adam57688@Ubuntu:~$ gcc --version
gcc (Ubuntu 13.3.0-6ubuntu2~24.04) 13.3.0
Copyright (C) 2023 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
adam57688@Ubuntu:~$
```

python3 --version

A terminal window titled 'adam57688@Ubuntu: ~' with a dark purple background. The prompt is 'adam57688@Ubuntu:~\$'. The command 'python3 --version' has been entered and executed, resulting in the output 'Python 3.12.3'. The prompt is now 'adam57688@Ubuntu:~\$' with a cursor. On the left side of the terminal, there is a vertical dock with icons for the Dash application, Home folder, Applications, Help, Terminal, CD-ROM, and Recycle Bin.

```
adam57688@Ubuntu:~$ python3 --version
Python 3.12.3
adam57688@Ubuntu:~$
```

bash --version

A terminal window titled 'adam57688@Ubuntu: ~' with a dark purple background. The prompt is 'adam57688@Ubuntu:~\$'. The command 'bash --version' has been entered and executed, resulting in the output: '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.', and 'There is NO WARRANTY, to the extent permitted by law.' The prompt is now 'adam57688@Ubuntu:~\$' with a cursor. On the left side of the terminal, there is a vertical dock with icons for the Dash application, Home folder, Applications, Help, Terminal, CD-ROM, and Recycle Bin.

```
adam57688@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.
adam57688@Ubuntu:~$
```

Assignment 4.3: Compile

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

Java, C and python

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

How do I run a Java program?

java filename

How do I run a Python program?

python3 file name

How do I run a C program?

./filename

How do I run a Bash script?

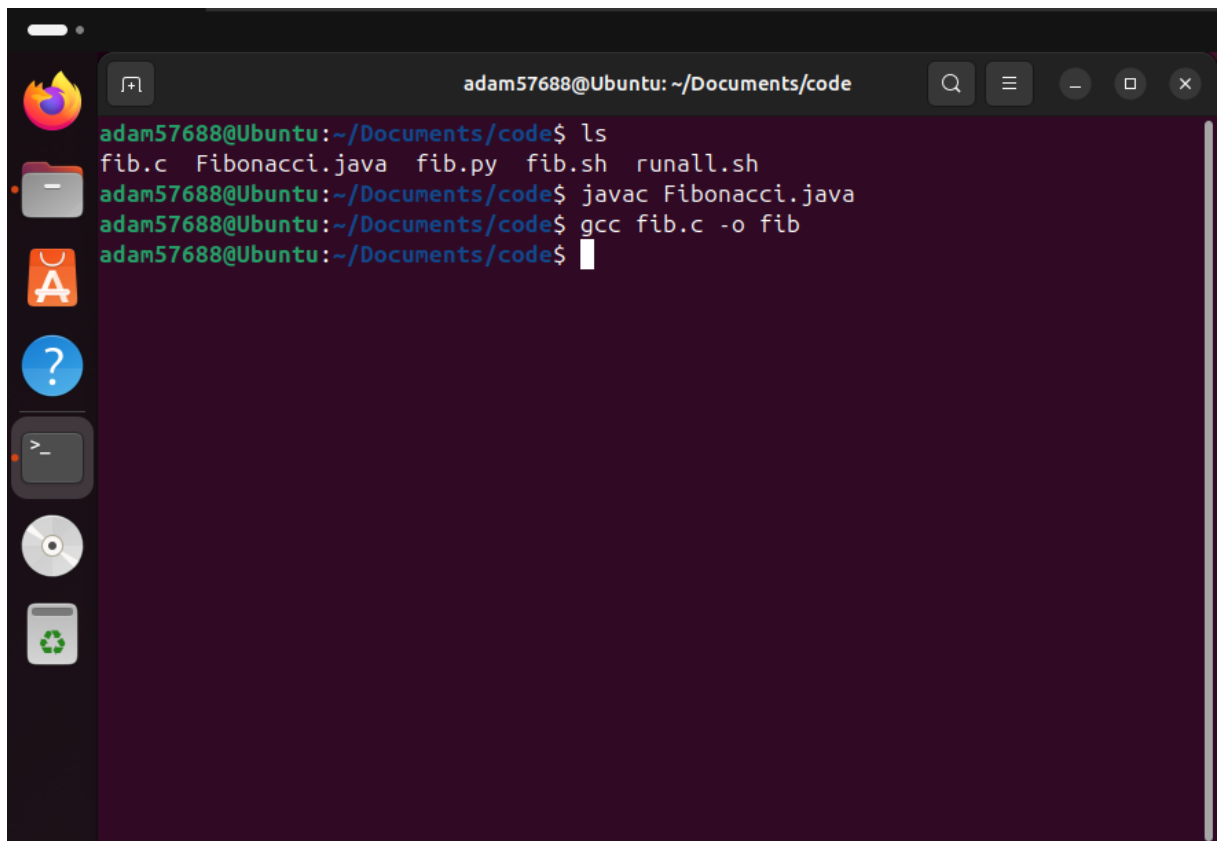
./filename

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

Yes, fib.c, Fibonacci.class, fib.py

Take relevant screenshots of the following commands:

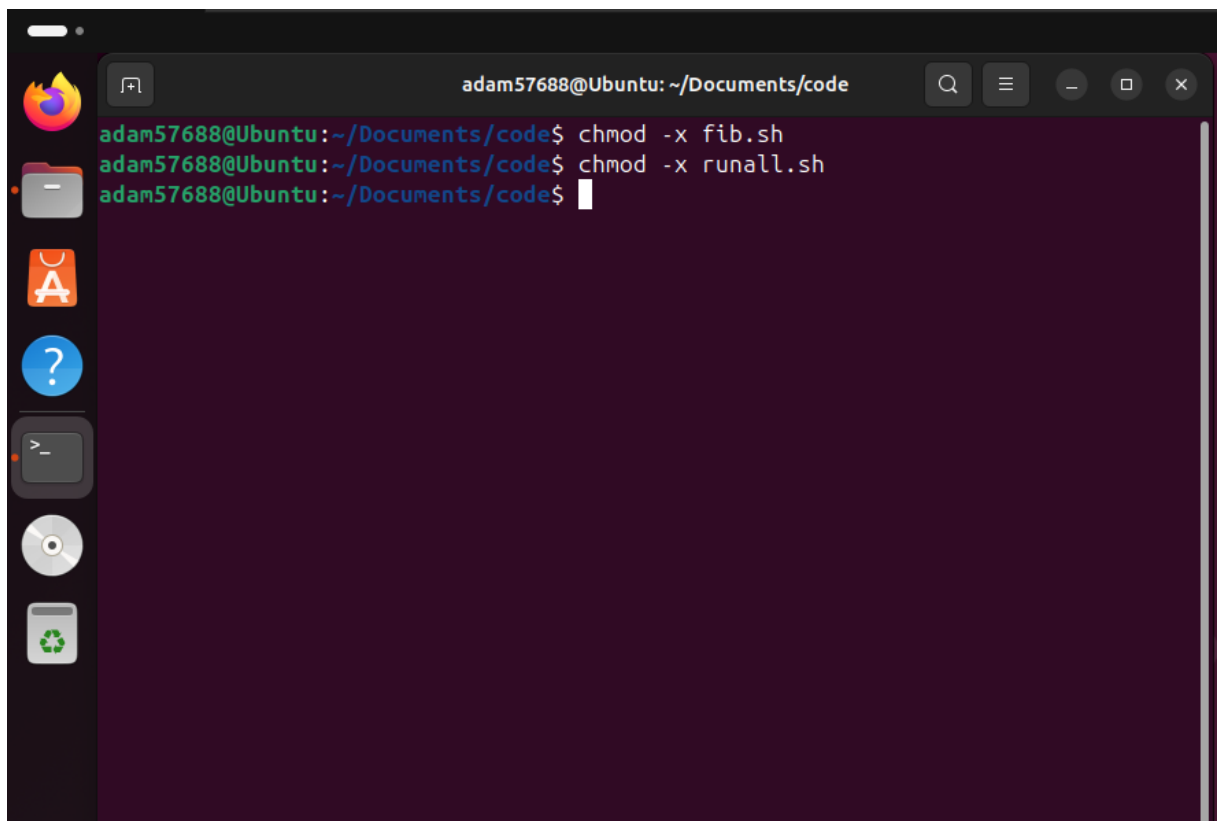
- Compile the source files where necessary



A terminal window titled 'adam57688@Ubuntu: ~/Documents/code' with search, menu, and window control buttons. The terminal shows the following commands and output:

```
adam57688@Ubuntu:~/Documents/code$ ls
fib.c  Fibonacci.java  fib.py  fib.sh  runall.sh
adam57688@Ubuntu:~/Documents/code$ javac Fibonacci.java
adam57688@Ubuntu:~/Documents/code$ gcc fib.c -o fib
adam57688@Ubuntu:~/Documents/code$
```

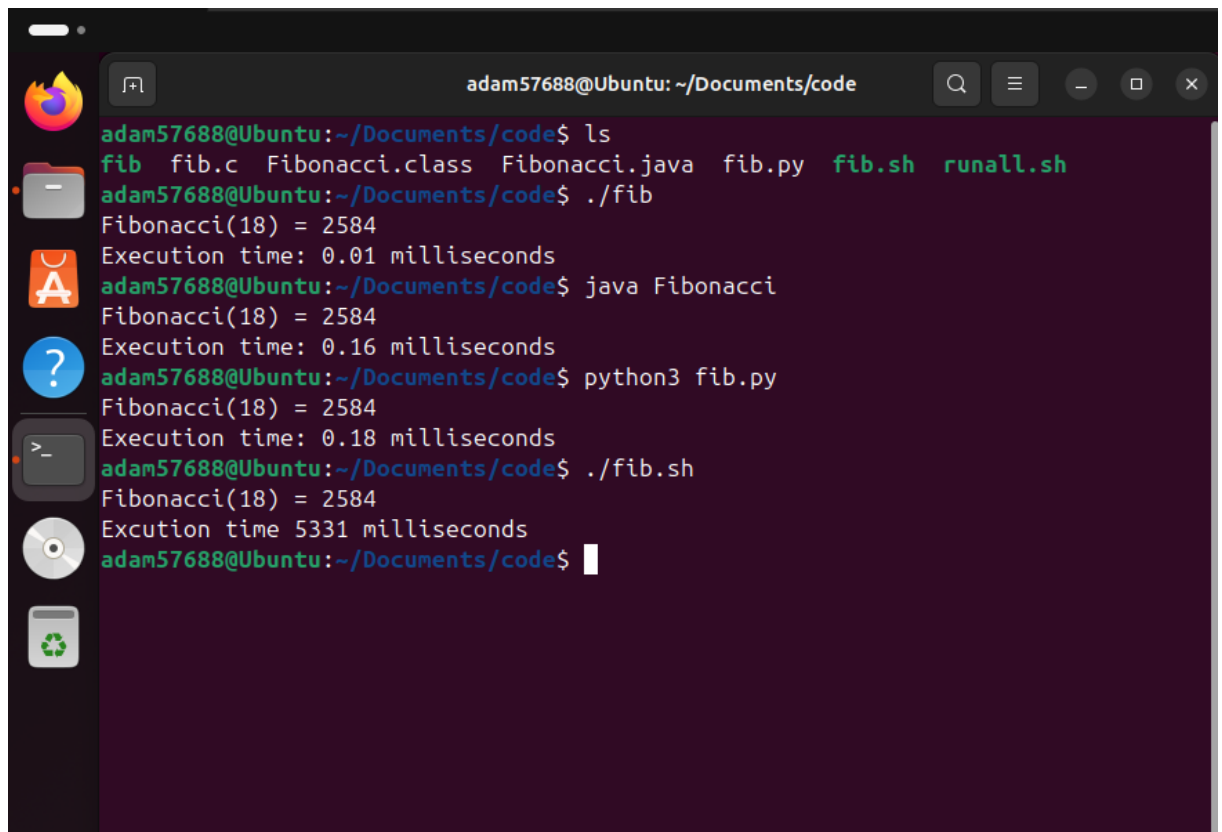
-
- Make them executable



A terminal window titled 'adam57688@Ubuntu: ~/Documents/code' with search, menu, and window control buttons. The terminal shows the following commands and output:

```
adam57688@Ubuntu:~/Documents/code$ chmod -x fib.sh
adam57688@Ubuntu:~/Documents/code$ chmod -x runall.sh
adam57688@Ubuntu:~/Documents/code$
```

-
- Run them



The image shows a terminal window titled 'adam57688@Ubuntu: ~/Documents/code'. The user has listed files: fib, fib.c, Fibonacci.class, Fibonacci.java, fib.py, fib.sh, and runall.sh. They then executed each file to calculate the 18th Fibonacci number (2584) and measured the execution time. The results are as follows:

Command	Execution Time
./fib	0.01 milliseconds
java Fibonacci	0.16 milliseconds
python3 fib.py	0.18 milliseconds
./fib.sh	5331 milliseconds

-
- Which (compiled) source code file performs the calculation the fastest?
Fib.c

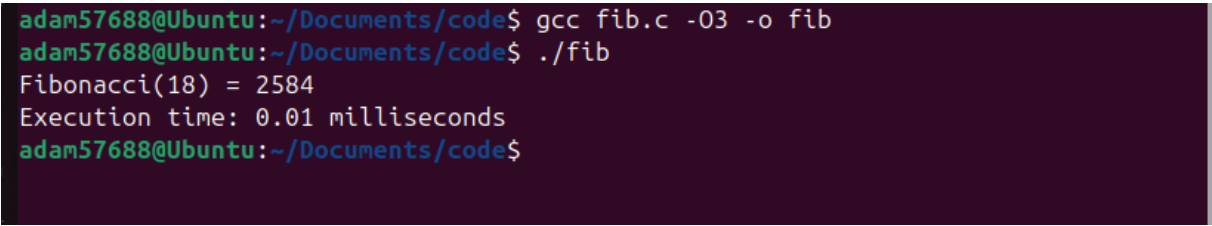
Assignment 4.4: Optimize

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.

```
gcc -O -o
```

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

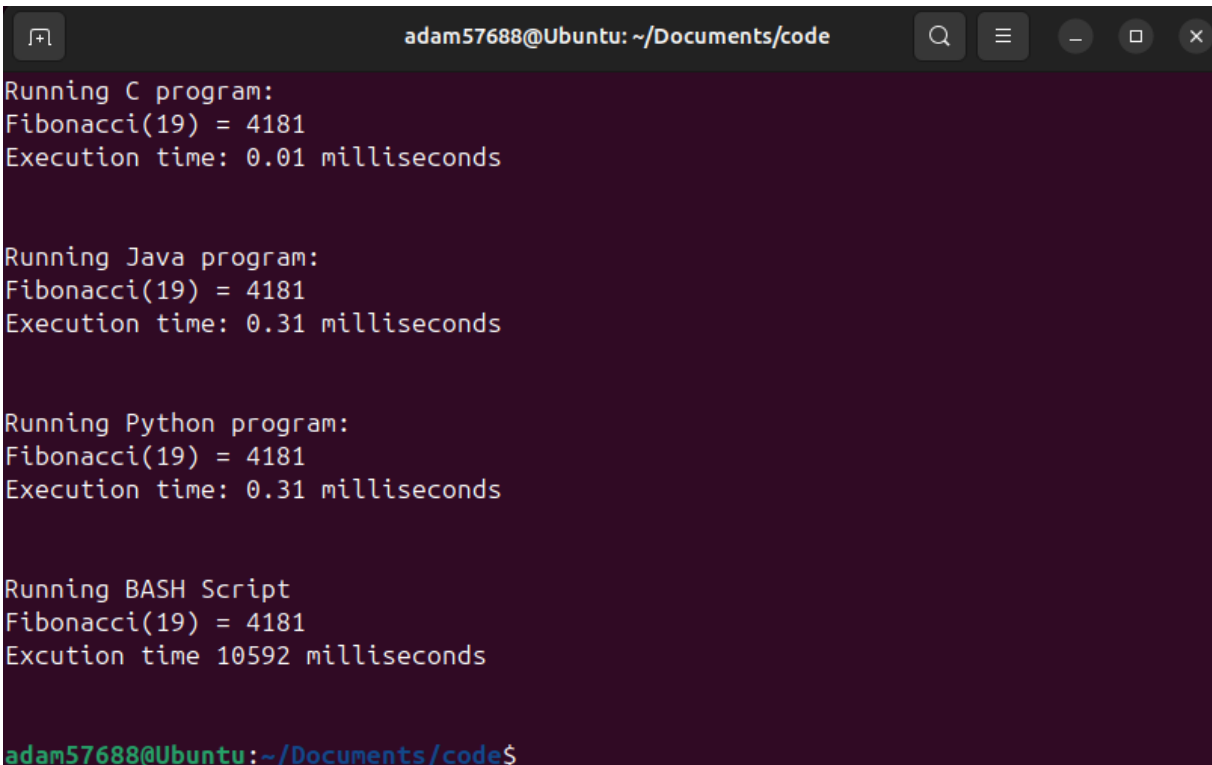


```
adam57688@Ubuntu:~/Documents/code$ gcc fib.c -O3 -o fib
adam57688@Ubuntu:~/Documents/code$ ./fib
Fibonacci(18) = 2584
Execution time: 0.01 milliseconds
adam57688@Ubuntu:~/Documents/code$
```

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

Yes.

- 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.



```
adam57688@Ubuntu: ~/Documents/code
Running C program:
Fibonacci(19) = 4181
Execution time: 0.01 milliseconds

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

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

Running BASH Script
Fibonacci(19) = 4181
Execution time 10592 milliseconds

adam57688@Ubuntu:~/Documents/code$
```

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
```

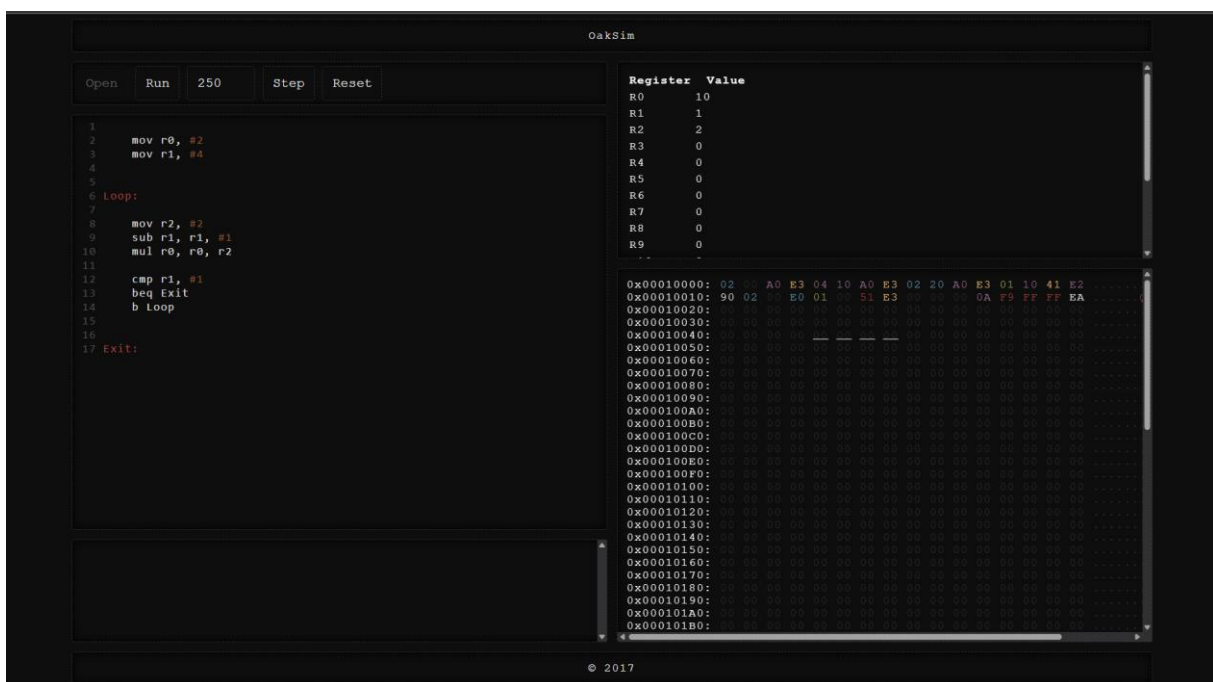
```
mov r2, #4
```

Loop:

End:

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

Screenshot of the completed code here.



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