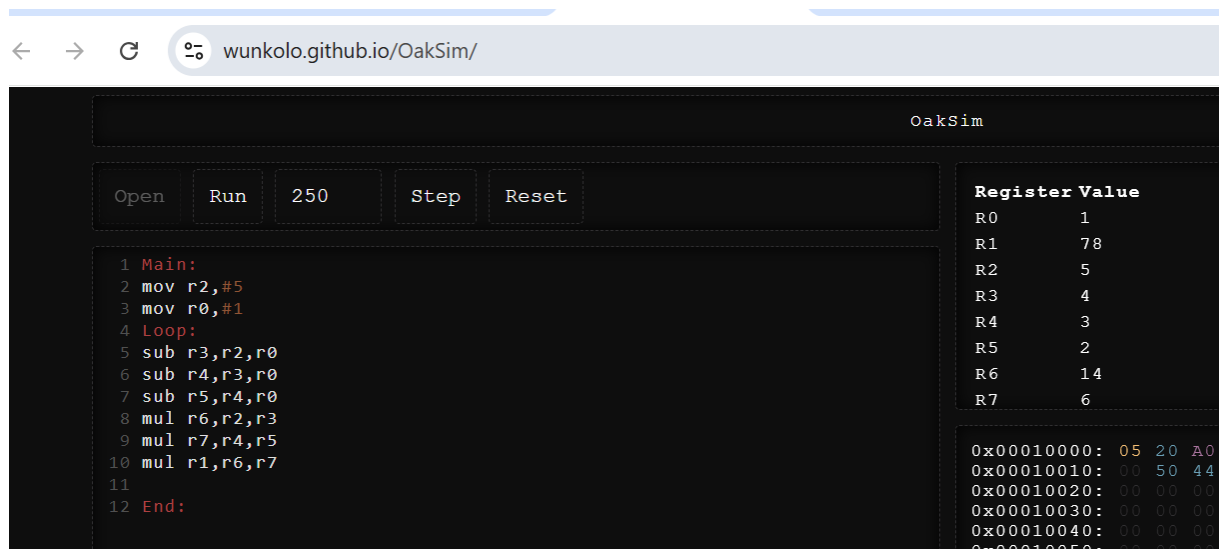


Template Week 4 – Software

Student number: 579185 – Nafsika Pagkali

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:

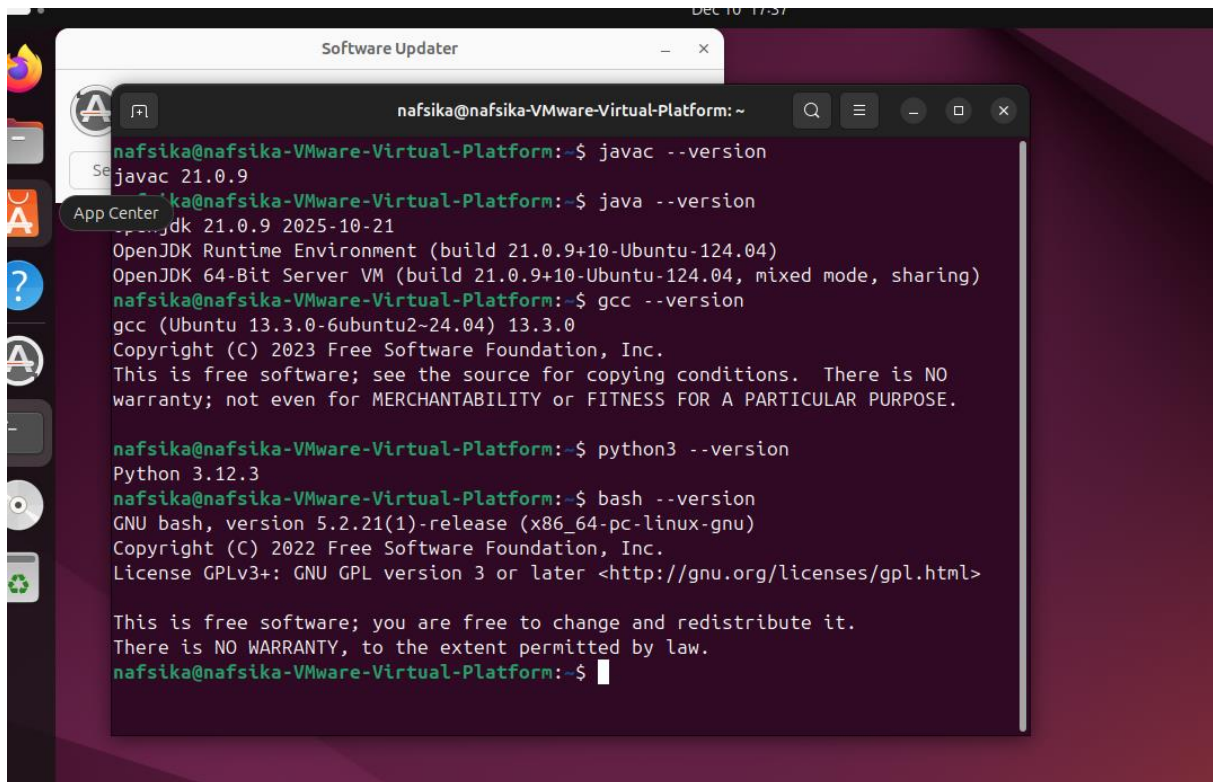
`javac --version`

`java --version`

`gcc --version`

`python3 --version`

`bash --version`



The screenshot shows a Linux desktop with a dark theme. A terminal window titled 'nafsika@nafsika-VMware-Virtual-Platform: ~' is open, displaying the results of several version-checking commands. Above the terminal, a 'Software Updater' window is partially visible. The terminal output is as follows:

```
nafsika@nafsika-VMware-Virtual-Platform:~$ javac --version
javac 21.0.9
nafsika@nafsika-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)
nafsika@nafsika-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.

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

Assignment 4.3: Compile

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

Java and C

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 and 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 machine code is executed by the processor

How do I run a Java program?

`javac Fibonacci.java (to compile)`

`java Fibonacci`

How do I run a Python program?

`python3 fib.py`

How do I run a C program?

`gcc fib.c -o fib (to compile)`

`./fib`

How do I run a Bash script?

`Bash fib.sh`

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

Java and C will create new files but python and bash no

Java : will create 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?
C and after that Java

```

nafsika@nafsika-VMware-Virtual-Platform:~/Downloads/code$ ls -l
total 20
-rw-rw-r-- 1 nafsika nafsika 831 Jun  9 2023 fib.c
-rw-rw-r-- 1 nafsika nafsika 839 Jun  9 2023 Fibonacci.java
-rw-rw-r-- 1 nafsika nafsika 516 Jun  9 2023 fib.py
-rw-rw-r-- 1 nafsika nafsika 668 Jun  9 2023 fib.sh
-rw-rw-r-- 1 nafsika nafsika 249 Jun  9 2023 runall.sh
nafsika@nafsika-VMware-Virtual-Platform:~/Downloads/code$ javac Fibonacci.java
nafsika@nafsika-VMware-Virtual-Platform:~/Downloads/code$ java Fibonacci
Fibonacci(18) = 2584
Execution time: 0.29 milliseconds
nafsika@nafsika-VMware-Virtual-Platform:~/Downloads/code$

nafsika@nafsika-VMware-Virtual-Platform:~/Downloads$ gcc fib.c -o fib
cc1: fatal error: fib.c: No such file or directory
compilation terminated.
nafsika@nafsika-VMware-Virtual-Platform:~/Downloads$ cd code
nafsika@nafsika-VMware-Virtual-Platform:~/Downloads/code$ gcc fib.c -o fib
nafsika@nafsika-VMware-Virtual-Platform:~/Downloads/code$ ./fib
Fibonacci(18) = 2584
Execution time: 0.07 milliseconds
nafsika@nafsika-VMware-Virtual-Platform:~/Downloads/code$

nafsika@nafsika-VMware-Virtual-Platform:~/Downloads/code$ ./fib.sh
Fibonacci(18) = 2584
Execution time 6107 milliseconds
nafsika@nafsika-VMware-Virtual-Platform:~/Downloads/code$ chmod +x fib.sh
nafsika@nafsika-VMware-Virtual-Platform:~/Downloads/code$ ./fib.sh
Fibonacci(18) = 2584
Execution time 6010 milliseconds
nafsika@nafsika-VMware-Virtual-Platform:~/Downloads/code$

```

```

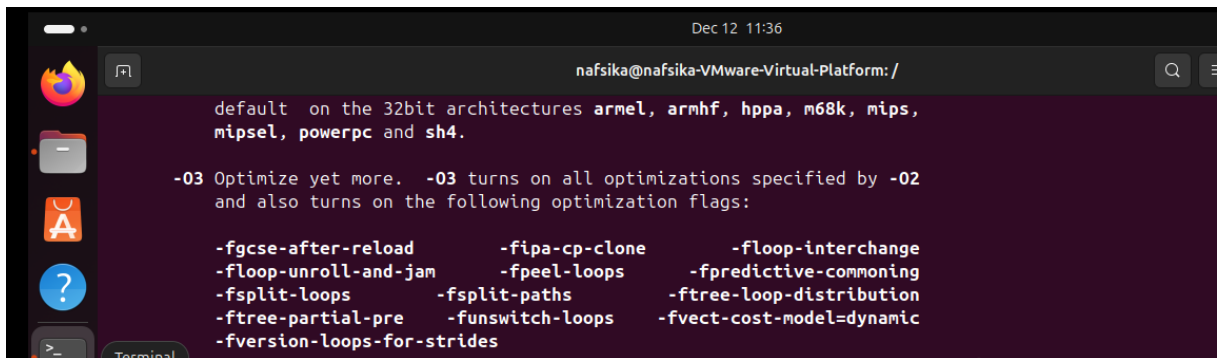
Execution time 6010 milliseconds
nafsika@nafsika-VMware-Virtual-Platform:~/Downloads/code$ python3 fib.py
Fibonacci(18) = 2584
Execution time: 0.68 milliseconds
nafsika@nafsika-VMware-Virtual-Platform:~/Downloads/code$

```

Assignment 4.4: Optimize

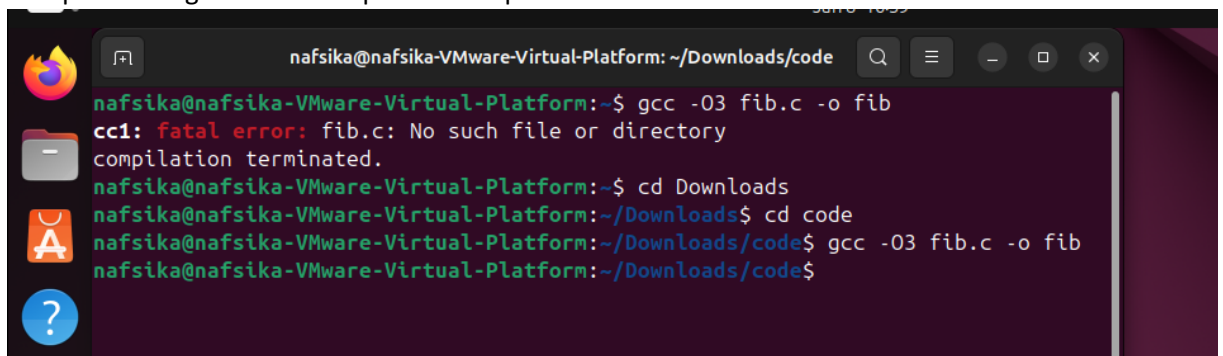
Take relevant screenshots of the following commands:

- 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 -O3`



```
nafsika@nafsika-VMware-Virtual-Platform: /  
default on the 32bit architectures armel, armhf, hppa, m68k, mips,  
mipsel, powerpc and sh4.  
  
-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
```

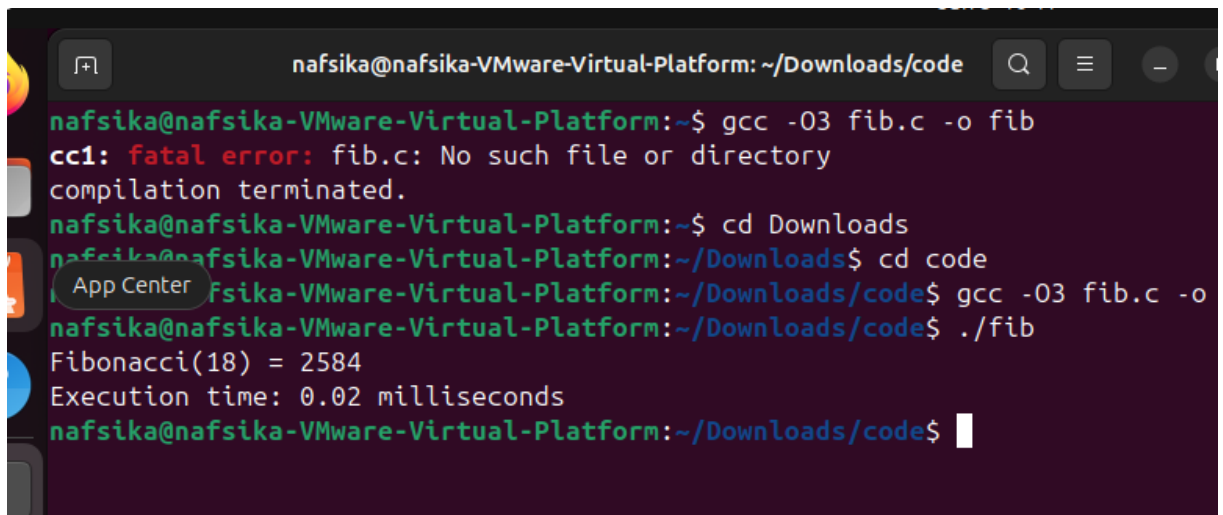
- Compile **fib.c** again with the optimization parameters



```
nafsika@nafsika-VMware-Virtual-Platform: ~/Downloads/code  
nafsika@nafsika-VMware-Virtual-Platform:~$ gcc -O3 fib.c -o fib  
cc1: fatal error: fib.c: No such file or directory  
compilation terminated.  
nafsika@nafsika-VMware-Virtual-Platform:~$ cd Downloads  
nafsika@nafsika-VMware-Virtual-Platform:~/Downloads$ cd code  
nafsika@nafsika-VMware-Virtual-Platform:~/Downloads/code$ gcc -O3 fib.c -o fib  
nafsika@nafsika-VMware-Virtual-Platform:~/Downloads/code$
```

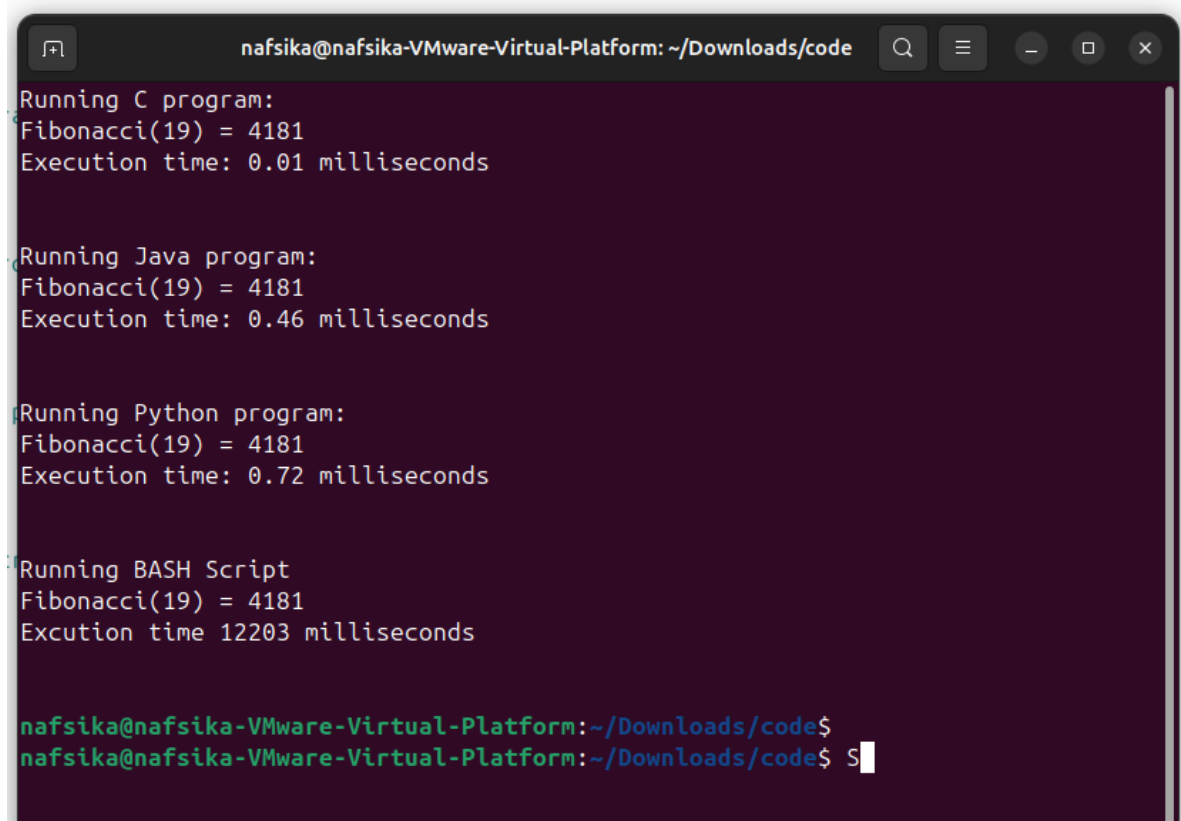
- Run the newly compiled program. Is it true that it now performs the calculation faster?
Indeed the calculation is faster

d)



```
nafsika@nafsika-VMware-Virtual-Platform: ~/Downloads/code
nafsika@nafsika-VMware-Virtual-Platform:~$ gcc -O3 fib.c -o fib
cc1: fatal error: fib.c: No such file or directory
compilation terminated.
nafsika@nafsika-VMware-Virtual-Platform:~$ cd Downloads
nafsika@nafsika-VMware-Virtual-Platform:~/Downloads$ cd code
nafsika@nafsika-VMware-Virtual-Platform:~/Downloads/code$ gcc -O3 fib.c -o
nafsika@nafsika-VMware-Virtual-Platform:~/Downloads/code$ ./fib
Fibonacci(18) = 2584
Execution time: 0.02 milliseconds
nafsika@nafsika-VMware-Virtual-Platform:~/Downloads/code$
```

- e) 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.



```
nafsika@nafsika-VMware-Virtual-Platform: ~/Downloads/code
Running C program:
Fibonacci(19) = 4181
Execution time: 0.01 milliseconds

Running Java program:
Fibonacci(19) = 4181
Execution time: 0.46 milliseconds

Running Python program:
Fibonacci(19) = 4181
Execution time: 0.72 milliseconds

Running BASH Script
Fibonacci(19) = 4181
Execution time 12203 milliseconds

nafsika@nafsika-VMware-Virtual-Platform:~/Downloads/code$
nafsika@nafsika-VMware-Virtual-Platform:~/Downloads/code$ S
```

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.

ain:

```
mov r0, #1    result = 1
mov r1, #2    base = 2
mov r2, #4    exponent = 4
```

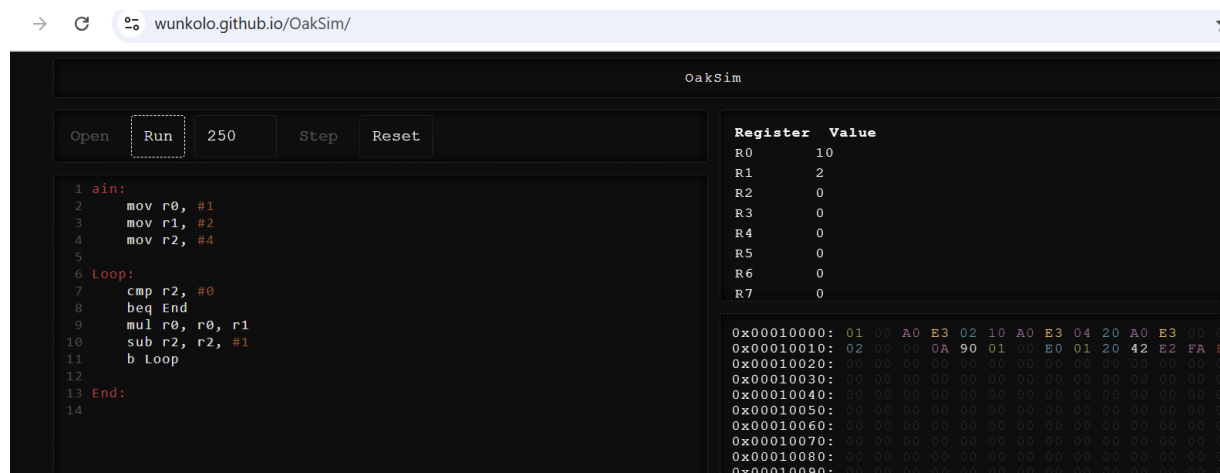
Loop:

```
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.



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