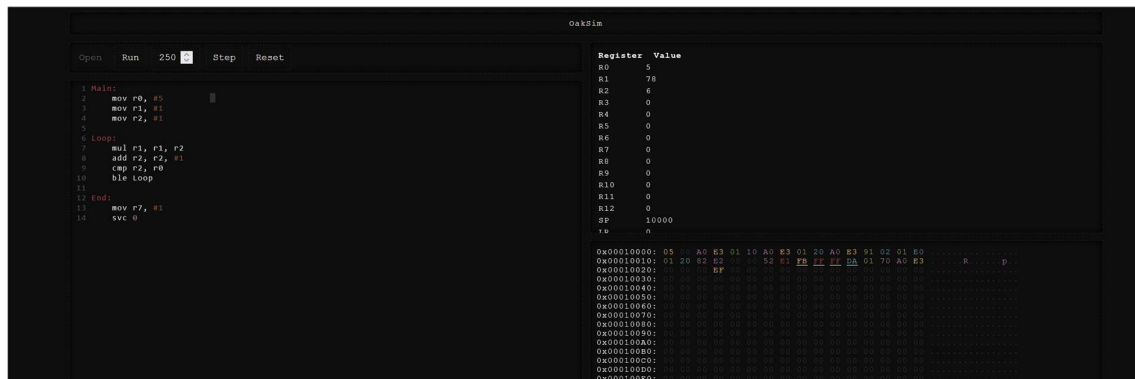


Template Week 4 – Software

Student number: 569681

Assignment 4.1: ARM assembly

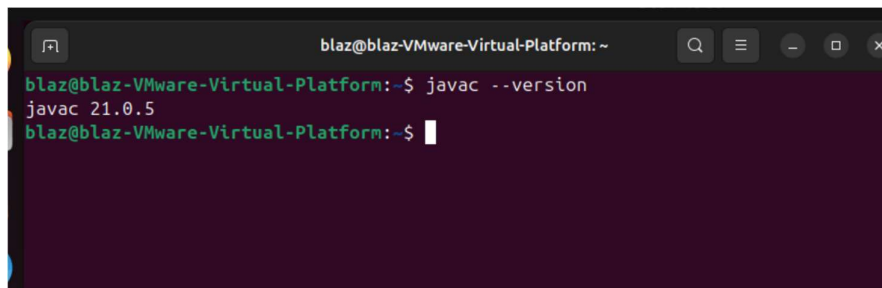
Screenshot of working assembly code of factorial calculation:



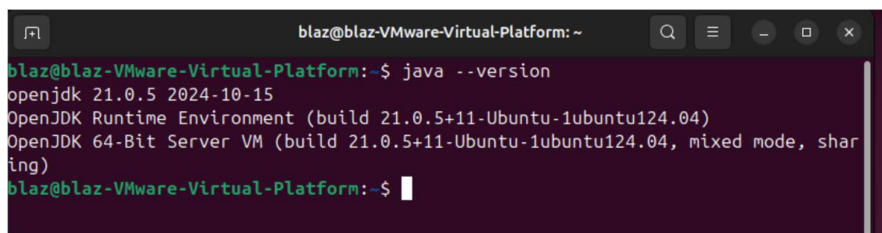
Assignment 4.2: Programming languages

Take screenshots that the following commands work:

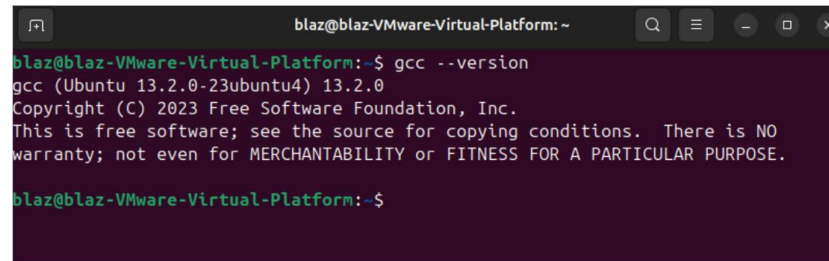
javac -version



java -version



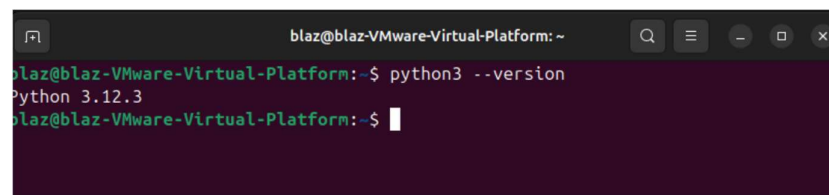
gcc --version

A terminal window titled 'blaz@blaz-VMware-Virtual-Platform: ~' with search, menu, and window control icons. The command 'gcc --version' has been executed, resulting in the following output:

```
blaz@blaz-VMware-Virtual-Platform:~$ 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.

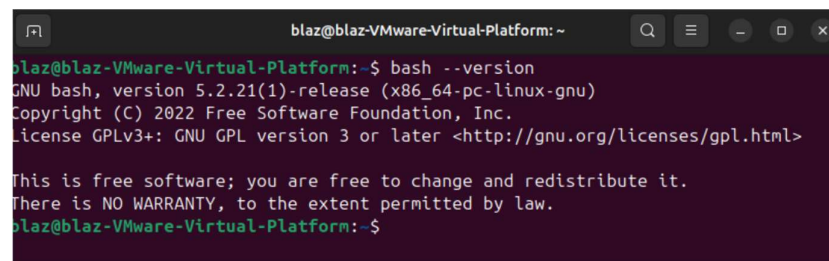
blaz@blaz-VMware-Virtual-Platform:~$
```

python3 --version

A terminal window titled 'blaz@blaz-VMware-Virtual-Platform: ~' with search, menu, and window control icons. The command 'python3 --version' has been executed, resulting in the following output:

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

bash --version

A terminal window titled 'blaz@blaz-VMware-Virtual-Platform: ~' with search, menu, and window control icons. The command 'bash --version' has been executed, resulting in the following output:

```
blaz@blaz-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.

blaz@blaz-VMware-Virtual-Platform:~$
```

Assignment 4.3: Compile

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

Fibonacci.java and 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 and 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?

Compile with `javac Fibonacci.java` and run with `java Fibonacci`

How do I run a Python program?

Run with `python3 fib.py`.

How do I run a C program?

Compile with `gcc fib.c -o fib` and run with `./fib`.

How do I run a Bash script?

Make it executable with `chmod +x fib.sh` and run with `./fib.sh` (or run directly with `bash fib.sh`).

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

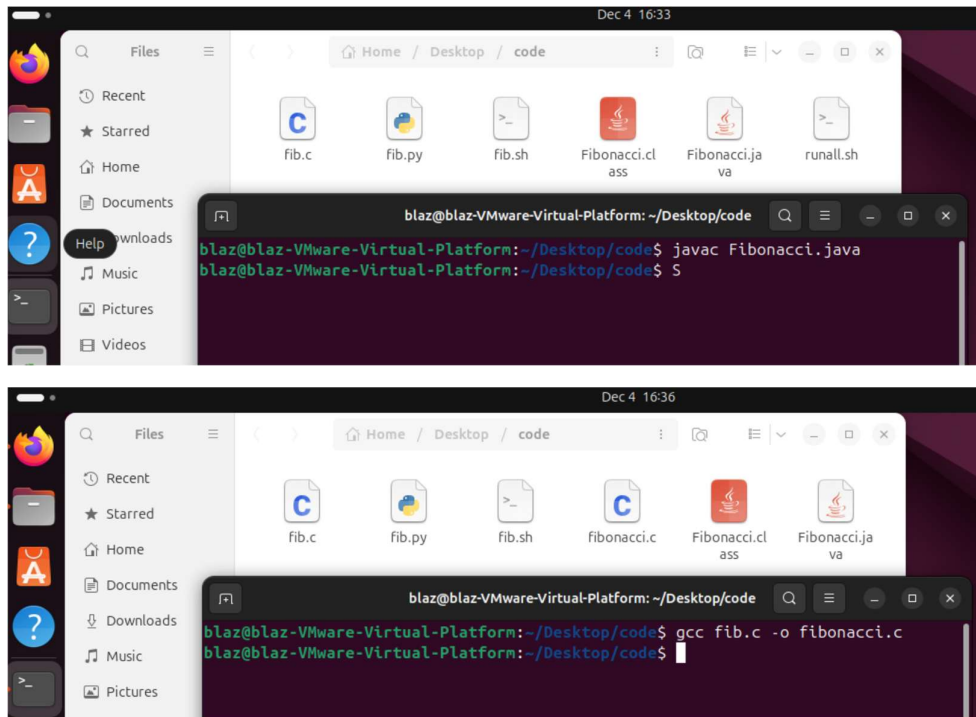
Java: Compiling creates Fibonacci.class.

C: Compiling creates an executable file, typically fib.

Python and Bash: No new file is created since they are interpreted.

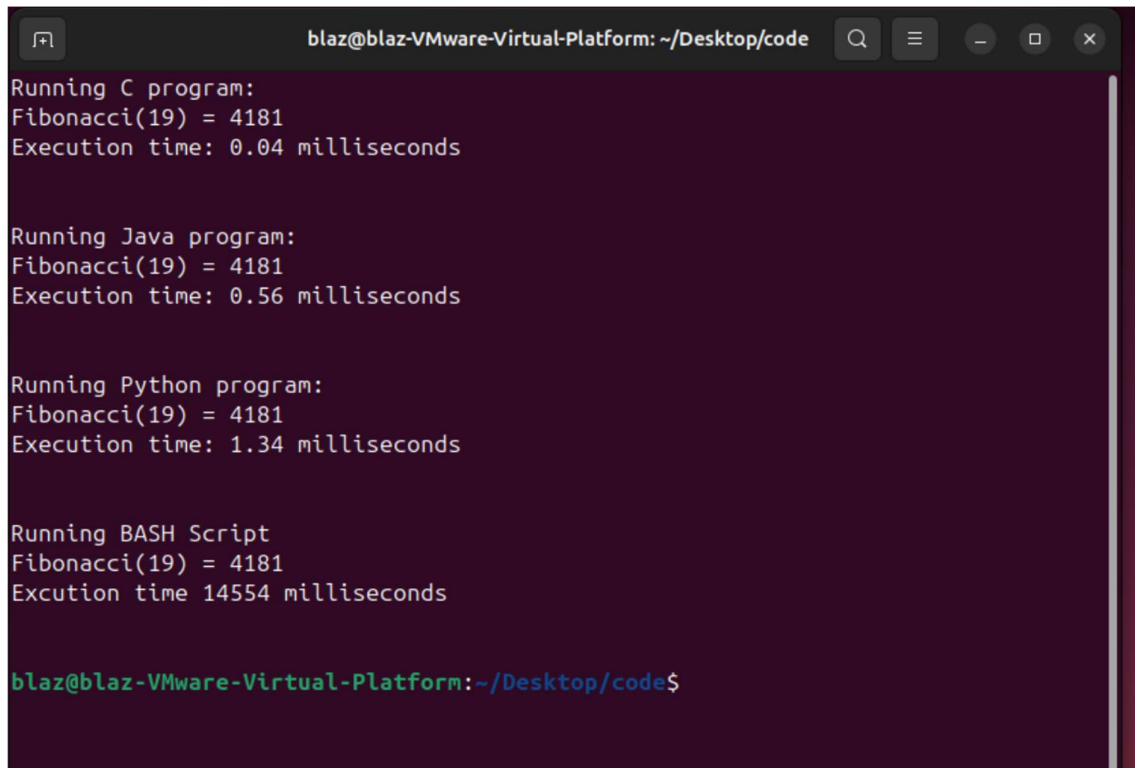
Take relevant screenshots of the following commands:

- Compile the source files where necessary



- Make them executable
- Run them

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



```
blaz@blaz-VMware-Virtual-Platform: ~/Desktop/code
Running C program:
Fibonacci(19) = 4181
Execution time: 0.04 milliseconds

Running Java program:
Fibonacci(19) = 4181
Execution time: 0.56 milliseconds

Running Python program:
Fibonacci(19) = 4181
Execution time: 1.34 milliseconds

Running BASH Script
Fibonacci(19) = 4181
Execution time 14554 milliseconds

blaz@blaz-VMware-Virtual-Platform:~/Desktop/code$
```

The image shows a terminal window with a dark background. The title bar at the top reads 'blaz@blaz-VMware-Virtual-Platform: ~/Desktop/code'. The terminal content shows four separate tests for calculating the 19th Fibonacci number (4181). The C program is the fastest at 0.04 ms, followed by Java at 0.56 ms and Python at 1.34 ms. The BASH script is significantly slower, taking 14554 ms. The prompt at the bottom is 'blaz@blaz-VMware-Virtual-Platform:~/Desktop/code\$'.

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.

-o turns on the optimization flags.

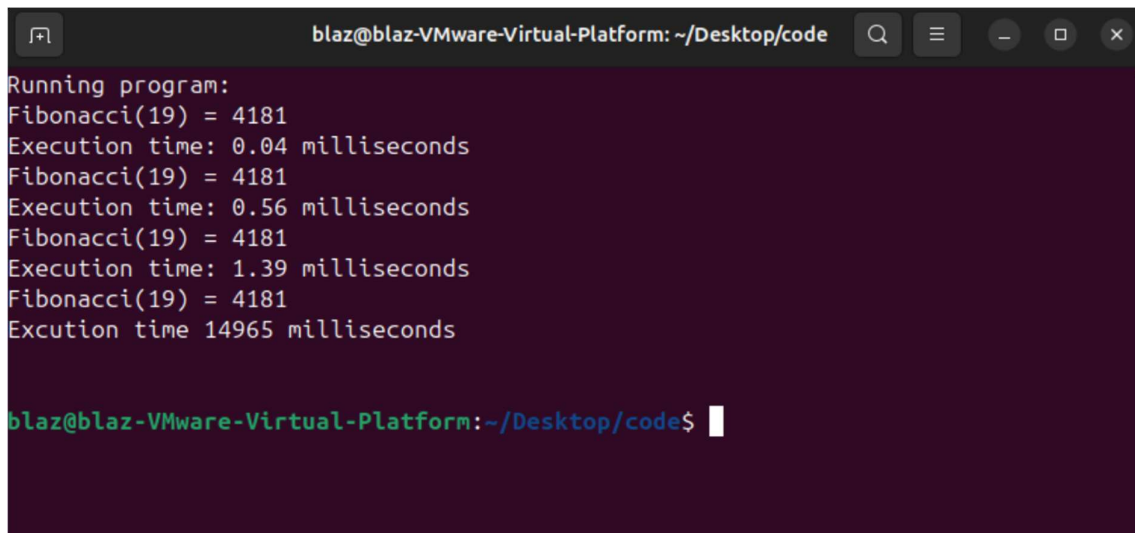
- b) Compile **fib.c** again with the optimization parameters

Already done, see task 4.3

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

The program runs a lot faster than the normally compiled c file.

- 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.



A terminal window titled 'blaz@blaz-VMware-Virtual-Platform: ~/Desktop/code' showing the execution of a program. The output displays four iterations of calculating the 19th Fibonacci number (4181) with decreasing execution times: 0.04, 0.56, 1.39, and 14965 milliseconds. The prompt at the bottom is 'blaz@blaz-VMware-Virtual-Platform:~/Desktop/code\$'.

```
Running program:
Fibonacci(19) = 4181
Execution time: 0.04 milliseconds
Fibonacci(19) = 4181
Execution time: 0.56 milliseconds
Fibonacci(19) = 4181
Execution time: 1.39 milliseconds
Fibonacci(19) = 4181
Execution time 14965 milliseconds

blaz@blaz-VMware-Virtual-Platform:~/Desktop/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 r0, #2
    mov r1, #2

Loop:
    mul r0, r0, r1

    cmp r2, #16

    beq End
    bne Loop

End:
    svc 0
```

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

Screenshot of the completed code here.

