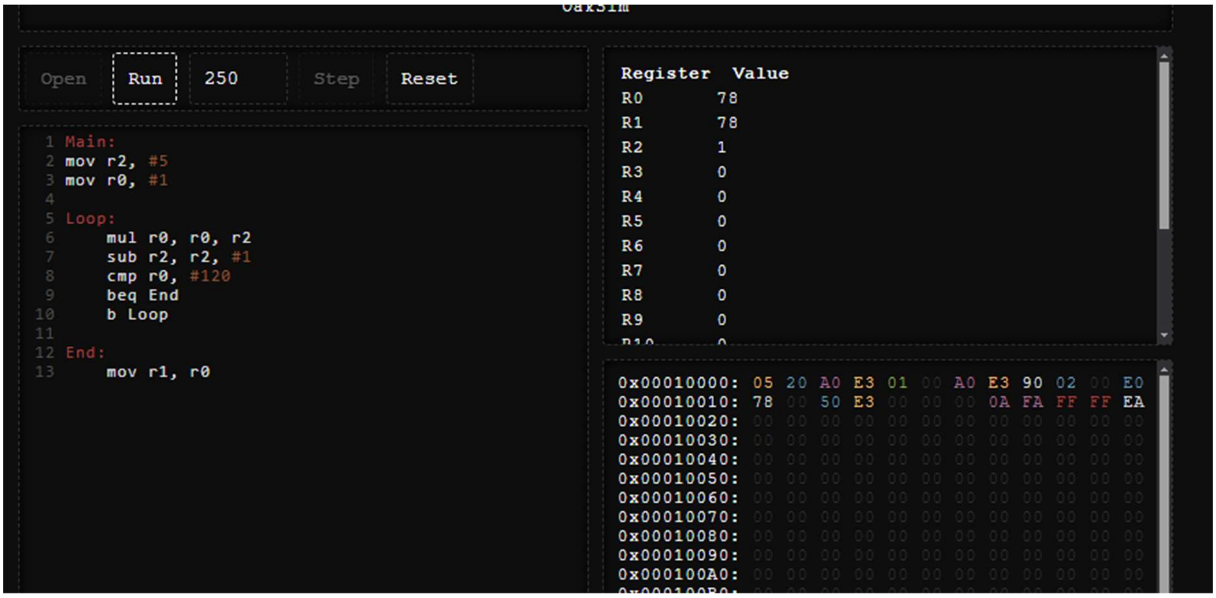
Template Week 4 – Software

Student number:

Assignment 4.1: ARM assembly

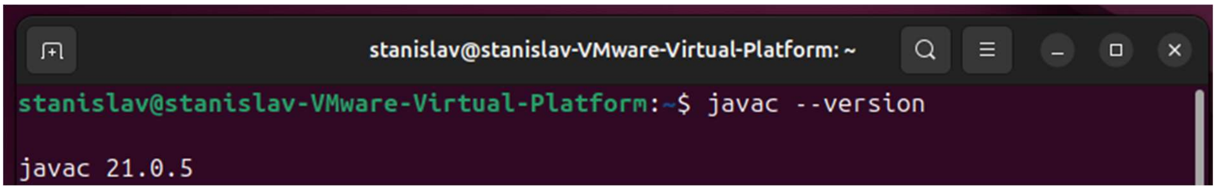
Screenshot of working assembly code of factorial calculation:



Assignment 4.2: Programming languages

Take screenshots that the following commands work:

javac --version



java --version

```
stanislav@stanislav-VMware-Virtual-Platform:~$ 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, shar
ing)
```

gcc --version

```
stanislav@stanislav-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.
```

python3 --version

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

bash --version

```
stanislav@stanislav-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.
```

Assignment 4.3: Compile

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

Needs to be compiled into bytecode using a Java compiler (javac).

Needs to be compiled into machine code using a C compiler (gcc).

Which source code files are compiled into machine code and then directly executable by a processor?

fib.c when compiled using gcc.

Which source code files are compiled to byte code?

Fibonacci.java when compiled using javac.

Which source code files are interpreted by an interpreter?

fib.py: Interpreted by the Python interpreter (python3).

fib.sh: Interpreted by the Bash shell.

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

fib.c is the fastest. With record speed of 0.03 miliseconds

How do I run a Java program?

First compile:

javac Fiboacci.java

Then execute:

Java Fibonacci

How do I run a Python program?

Python3 fib.py

How do I run a C program?

First compile:

gcc -o fib fib.c

Then execute:

./fib

How do I run a Bash script?

First get access:

chmod +x fib.sh

Then execute:

./fib.sh

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

Java:

Creates a class object Fibonacci.class

```
stanislav@stanislav-VMware-Virtual-Platform:~/code$ javac Fibonacci.java
stanislav@stanislav-VMware-Virtual-Platform:~/code$ java Fibonacci
Fibonacci(18) = 2584
Execution time: 0.53 milliseconds
```

Python:

Executes script straight away

```
stanislav@stanislav-VMware-Virtual-Platform:~/code$ python3 fib.py
Fibonacci(18) = 2584
Execution time: 0.48 milliseconds
```

C:

Requires compilation (in my case fib file was created)

```
stanislav@stanislav-VMware-Virtual-Platform:~/code$ gcc -o fib fib.c
stanislav@stanislav-VMware-Virtual-Platform:~/code$ ./fib
Fibonacci(18) = 2584
Execution time: 0.03 milliseconds
```

Bash:

Executes script straight away

```
stanislav@stanislav-VMware-Virtual-Platform:~/code$ chmod +x fib.sh
stanislav@stanislav-VMware-Virtual-Platform:~/code$ ./fib.sh
Fibonacci(18) = 2584
Excution time 10239 milliseconds
```

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?

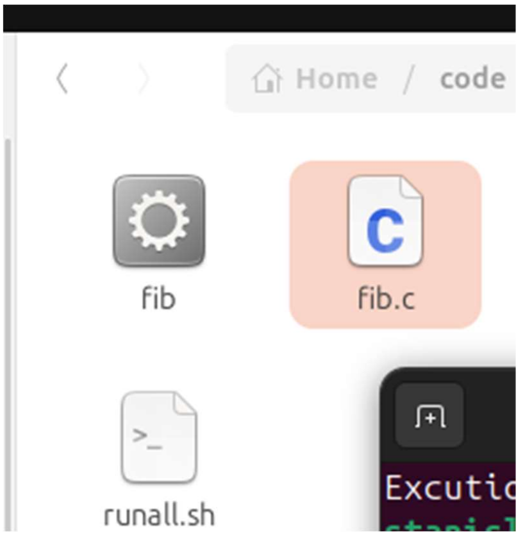
Assignment 4.4: Optimize

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.

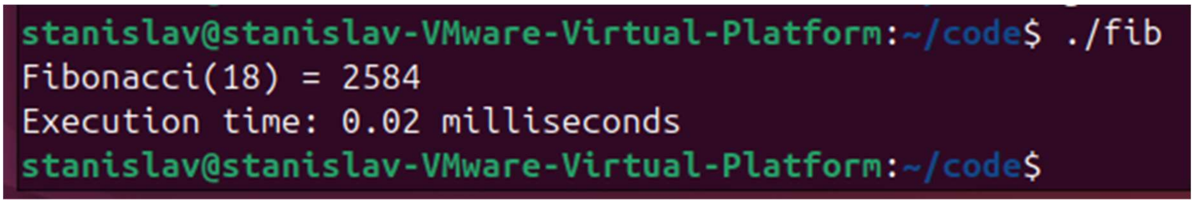
   gcc –O -o

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



```
stanislav@stanislav-VMware-Virtual-Platform:~/code$ gcc -O -o fib fib.c
```

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



```
stanislav@stanislav-VMware-Virtual-Platform:~/code$ ./fib
Fibonacci(18) = 2584
Execution time: 0.02 milliseconds
stanislav@stanislav-VMware-Virtual-Platform:~/code$
```

   Yes, it does! Previously 0.03 milliseconds, now – 0.02

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.

```
stanislav@stanislav-VMware-Virtu
Running C program:
Fibonacci(19) = 4181
Execution time: 0.03 milliseconds


Running Java program:
Fibonacci(19) = 4181
Execution time: 0.50 milliseconds


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


Running BASH Script
```

```
stanislav@stanislav-VMware-Virtual-Platform: ~/code
  GNU nano 7.2                    runall.sh *
S#!/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'

                    [ To suspend, type ^T^Z ]
^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
```

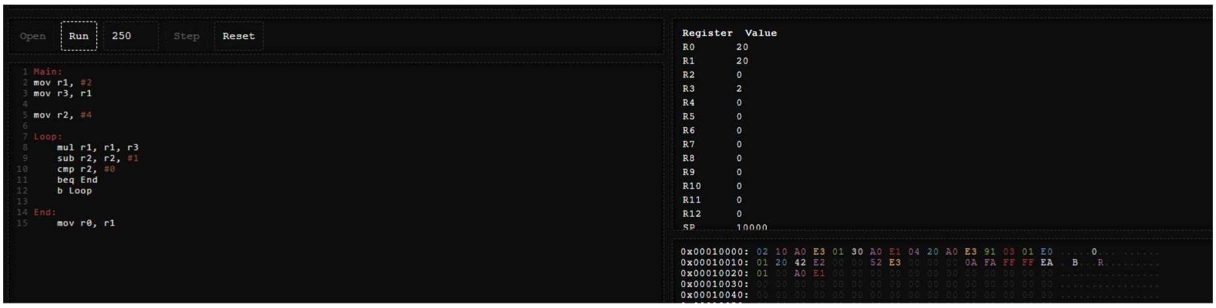
Bonus point assignment – week 4

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 r3, r1

mov r2, #4

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

End:
      mov r0, r1
```

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**