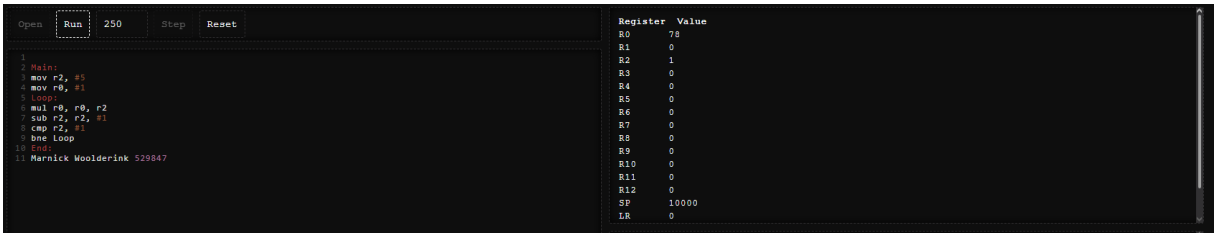
Template Week 4 – Software

Student number:

Assignment 4.1: ARM assembly

Screenshot of working assembly code of factorial calculation:

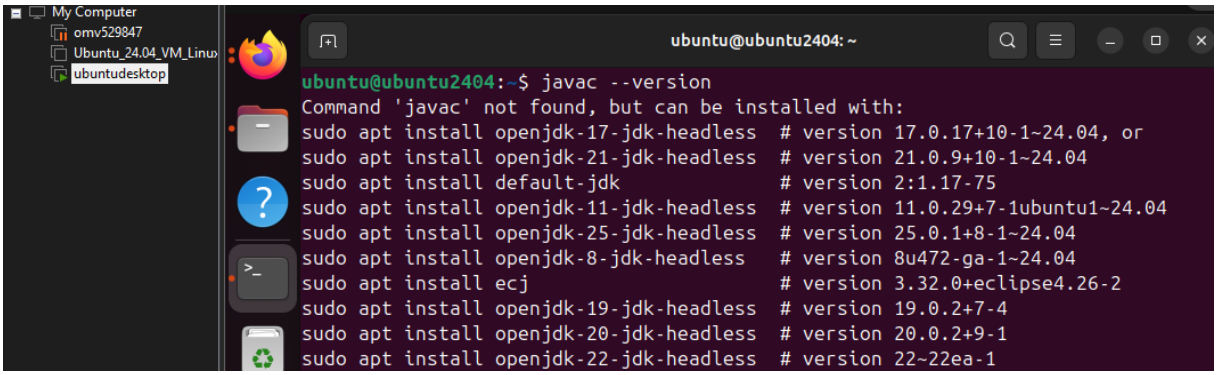


Antwoord is in hexadecimal maar is wel 120

Assignment 4.2: Programming languages

Take screenshots that the following commands work:

javac –version

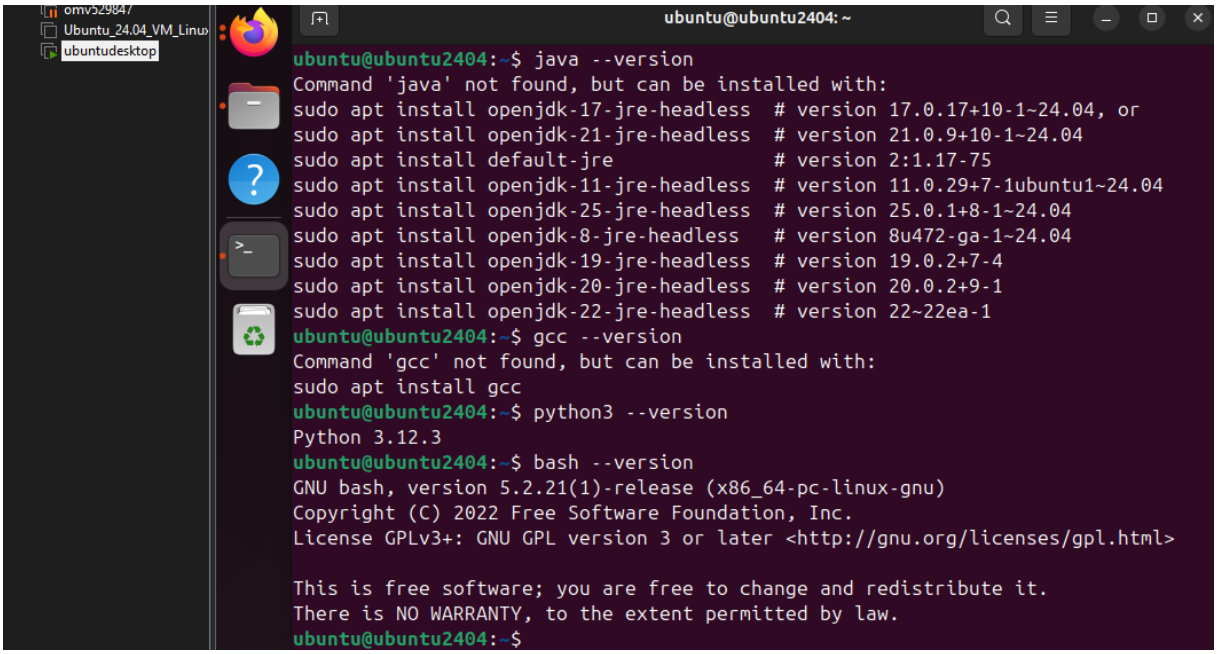


java --version

gcc --version

python3 --version

bash --version


ubuntu@ubuntu2404: ~
ubuntu@ubuntu2404:~$ java --version
Command 'java' not found, but can be installed with:
sudo apt install openjdk-17-jre-headless # version 17.0.17+10-1~24.04, or
sudo apt install openjdk-21-jre-headless # version 21.0.9+10-1~24.04
sudo apt install default-jre   # version 2:1.17-75
sudo apt install openjdk-11-jre-headless # version 11.0.29+7-1ubuntu1~24.04
sudo apt install openjdk-25-jre-headless # version 25.0.1+8-1~24.04
sudo apt install openjdk-8-jre-headless   # version 8u472-ga-1~24.04
sudo apt install openjdk-19-jre-headless # version 19.0.2+7-4
sudo apt install openjdk-20-jre-headless # version 20.0.2+9-1
sudo apt install openjdk-22-jre-headless # version 22~22ea-1
ubuntu@ubuntu2404:~$ gcc --version
Command 'gcc' not found, but can be installed with:
sudo apt install gcc
ubuntu@ubuntu2404:~$ python3 --version
Python 3.12.3
ubuntu@ubuntu2404:~$ 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@ubuntu2404:~$

Assignment 4.3: Compile

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

Fibonacci.java en 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?

Fibbonacci.java

Which source code files are interpreted by an interpreter?

Fib.py en 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 Fibonacci.java # compile

java Fibonacci # run (no .java or .class)

How do I run a Python program?

python3 fib.py

How do I run a C program?

gcc fib.c -o fib # compile

./fib # run

How do I run a Bash script?

chmod +x fib.sh # make executable (once)

./fib.sh

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

Fibbonacci.java ja: Fibbonacci.class

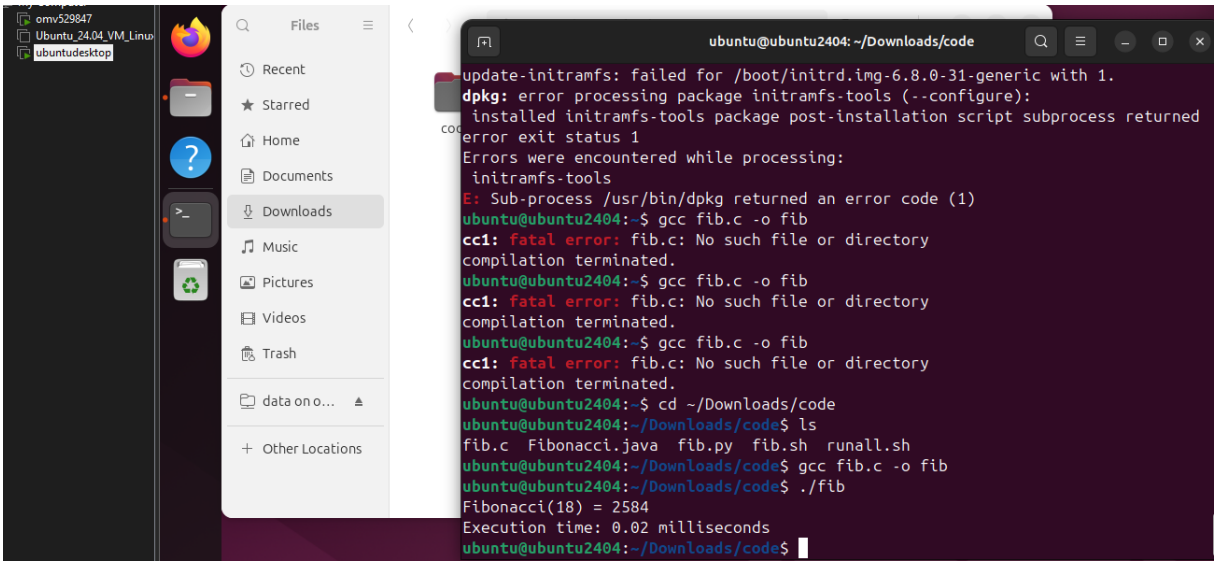
Fib.c ja: fib of fib.exe

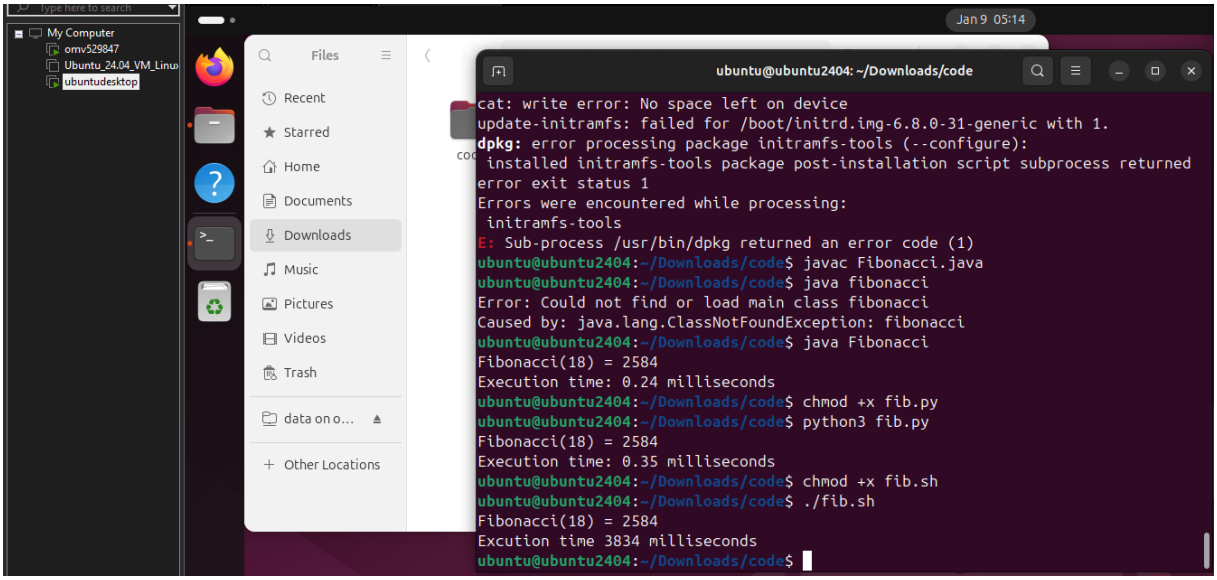
Fib.py mogelijk: eventueel .pyc

Fib.sh nee

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?





De fib.c file was de snelste met 0.02 milliseconden

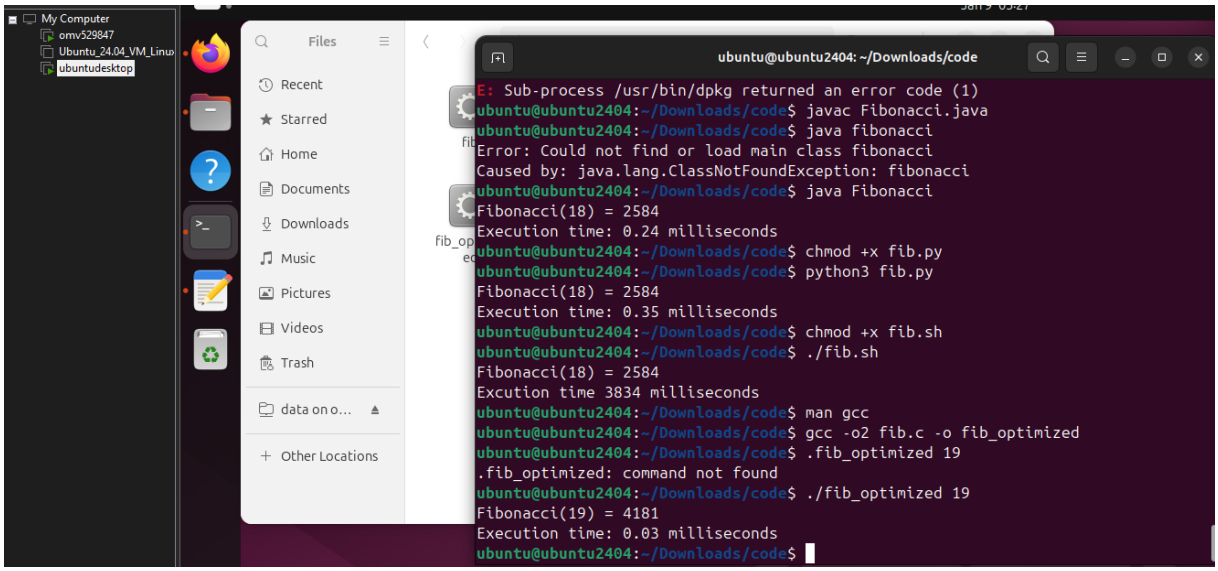
Assignment 4.4: Optimize

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.

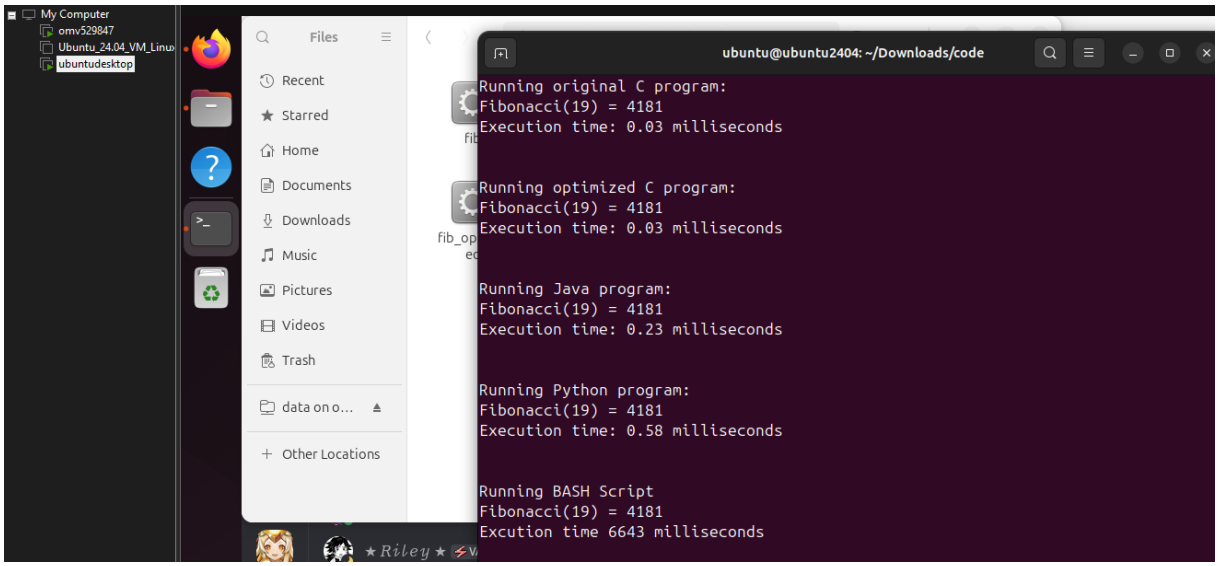
b) Compile **fib.c** again with the optimization parameters

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



It is not faster then the previous time

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.



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.

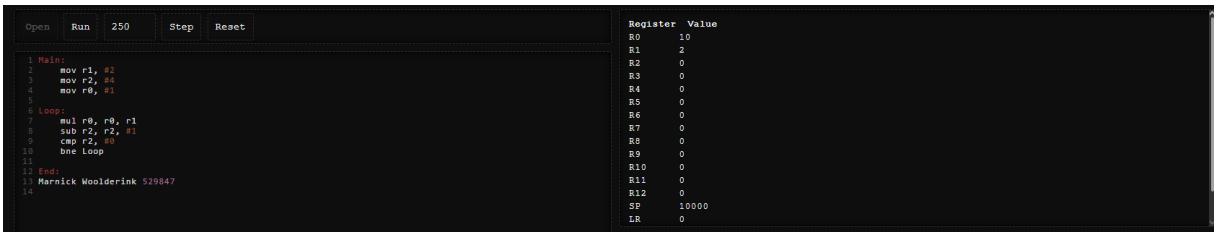
```
Main:
mov r1, #2
mov r2, #4

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**