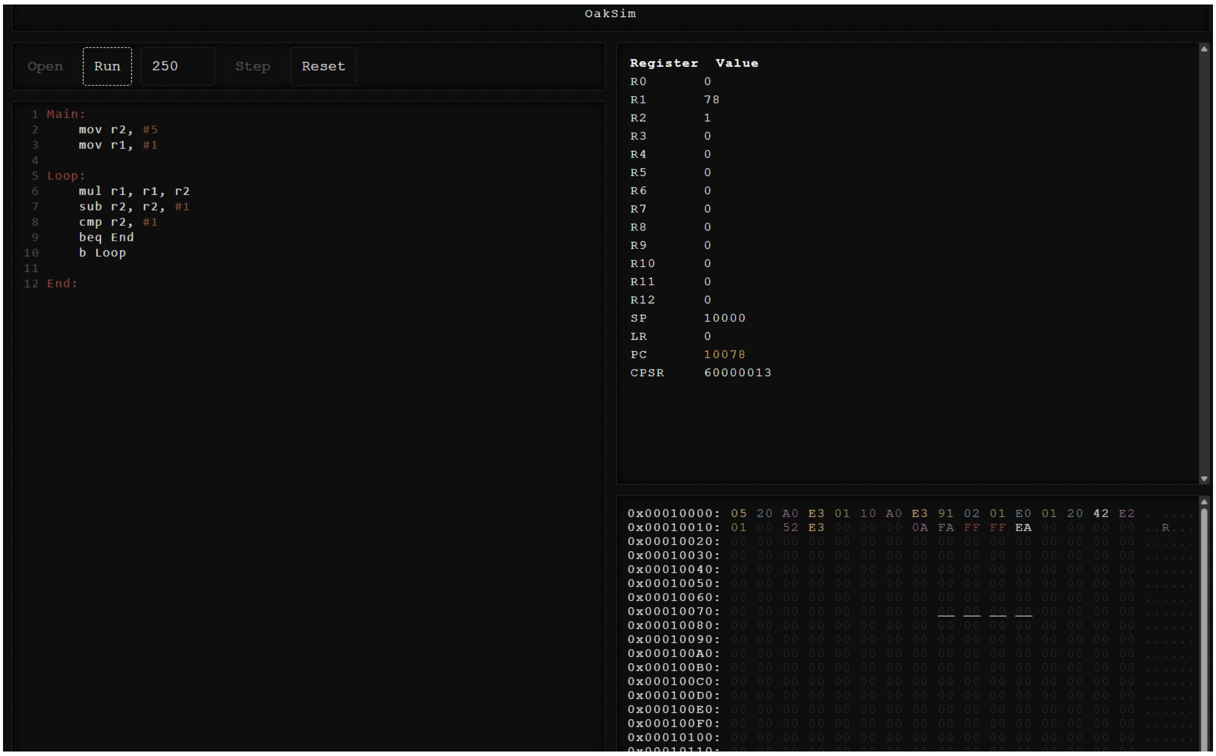
Template Week 4 – Software

Student number:582310

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

```
bash: syntax error near unexpected token `('
lucas@Helpdesk:~$ javac --version
Command 'javac' not found, but can be installed with:
sudo apt install default-jdk              # version 2:1.17-75, or
sudo apt install openjdk-17-jdk-headless  # version 17.0.17+10-1~24.04
sudo apt install openjdk-21-jdk-headless  # version 21.0.9+10-1~24.04
sudo apt install openjdk-11-jdk-headless  # version 11.0.29+7-1ubuntu1~24.04
sudo apt install openjdk-25-jdk-headless  # version 25.0.1+8-1~24.04
sudo apt install openjdk-8-jdk-headless   # version 8u472-ga-1~24.04
sudo apt install ecj                      # version 3.32.0+eclipse4.26-2
sudo apt install openjdk-19-jdk-headless  # version 19.0.2+7-4
sudo apt install openjdk-20-jdk-headless  # version 20.0.2+9-1
sudo apt install openjdk-22-jdk-headless  # version 22~22ea-1
lucas@Helpdesk:~$
```

java –version

```
lucas@Helpdesk:~$ java --version
Command 'java' not found, but can be installed with:
sudo apt install default-jre              # version 2:1.17-75, or
sudo apt install openjdk-17-jre-headless  # version 17.0.17+10-1~24.04
sudo apt install openjdk-21-jre-headless  # version 21.0.9+10-1~24.04
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
lucas@Helpdesk:~$
```

gcc –version

```
lucas@Helpdesk:~$ gcc --version
Command 'gcc' not found, but can be installed with:
sudo apt install gcc
```

python3 –version

```
lucas@Helpdesk:~$ python3 -- version
python3: can't open file '/home/lucas/version': [Errno 2] No such file or directory
```

bash –version

```
lucas@Helpdesk:~$ 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?
De bestanden die moeten worden gecompileerd zijn fib.c en Fibonacci.java

Which source code files are compiled into machine code and then directly executable by a processor?
Het bestand fib.c wordt gecompileerd naar machinecode en is daarna direct uitvoerbaar door de processor

Which source code files are compiled to byte code?
Het bestand Fibonacci.java wordt gecompileerd naar bytecode en uitgevoerd door de Java Virtual Machine

Which source code files are interpreted by an interpreter?
De bestanden fib.py en fib.sh worden geïnterpreteerd door respectievelijk de Python-interpreter en de Bash-interpreter

These source code files will perform the same calculation after compilation/interpretation. Which one is expected to do the calculation the fastest?
Het C-programma is het snelst omdat het wordt gecompileerd naar native machinecode

How do I run a Java program?
Je voert een Java-programma uit door het eerst te compileren met javac Bestandsnaam.java en daarna uit te voeren met java Bestandsnaam zonder extensie

How do I run a Python program?
Je voert een Python-programma uit met python3 fib.py

How do I run a C program?
Je compileert een C-programma met gcc fib.c -o fib en voert het uit met ./fib

How do I run a Bash script?
Je maakt een Bash-script uitvoerbaar met chmod a+x fib.sh en voert het uit met ./fib.sh

If I compile the above source code, will a new file be created? If so, which file?
Bij het compileren van fib.c wordt het bestand fib aangemaakt
Bij het compileren van Fibonacci.java wordt het bestand Fibonacci.class aangemaakt
fib.py en fib.sh maken geen nieuwe bestanden aan

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?

```
lucas@Helpdesk:~/Downloads/code$ cd ~/Downloads/code
ls
fib.c  Fibonacci.java  fib.py  fib.sh  runall.sh
lucas@Helpdesk:~/Downloads/code$ sudo chmod a+x fib.sh
ls
fib.c  Fibonacci.java  fib.py  fib.sh  runall.sh
lucas@Helpdesk:~/Downloads/code$ gcc fib.c -o fib
lucas@Helpdesk:~/Downloads/code$ ./fib
Fibonacci(18) = 2584
Execution time: 0.01 milliseconds
lucas@Helpdesk:~/Downloads/code$
```

```
lucas@Helpdesk:~/Downloads/code$ javac --version
java --version
javac 21.0.9
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)
lucas@Helpdesk:~/Downloads/code$ cd ~/Downloads/code
lucas@Helpdesk:~/Downloads/code$ javac Fibonacci.java
lucas@Helpdesk:~/Downloads/code$ java Fibonacci
Fibonacci(18) = 2584
Execution time: 0.16 milliseconds
lucas@Helpdesk:~/Downloads/code$
```

```
lucas@Helpdesk:~/Downloads/code$ sudo apt install python3
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
python3 is already the newest version (3.12.3-0ubuntu2.1).
python3 set to manually installed.
The following package was automatically installed and is no longer required:
  libllvm19
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 154 not upgraded.
lucas@Helpdesk:~/Downloads/code$ python3 fib.py
Fibonacci(18) = 2584
Execution time: 0.21 milliseconds
lucas@Helpdesk:~/Downloads/code$ S
```

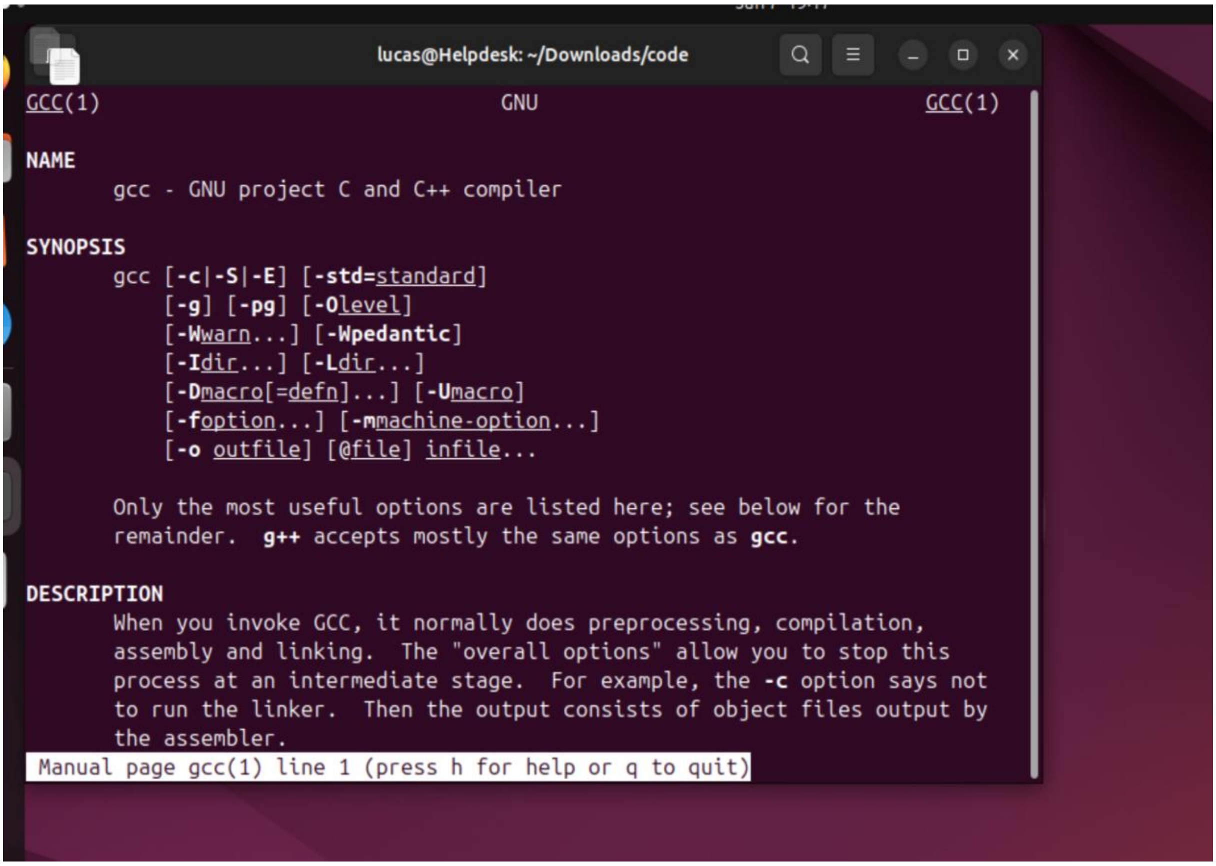
```
lucas@Helpdesk:~/Downloads/code$ ./fib.sh

Fibonacci(18) = 2584
Excution time 6064 milliseconds
lucas@Helpdesk:~/Downloads/code$
lucas@Helpdesk:~/Downloads/code$ S
```

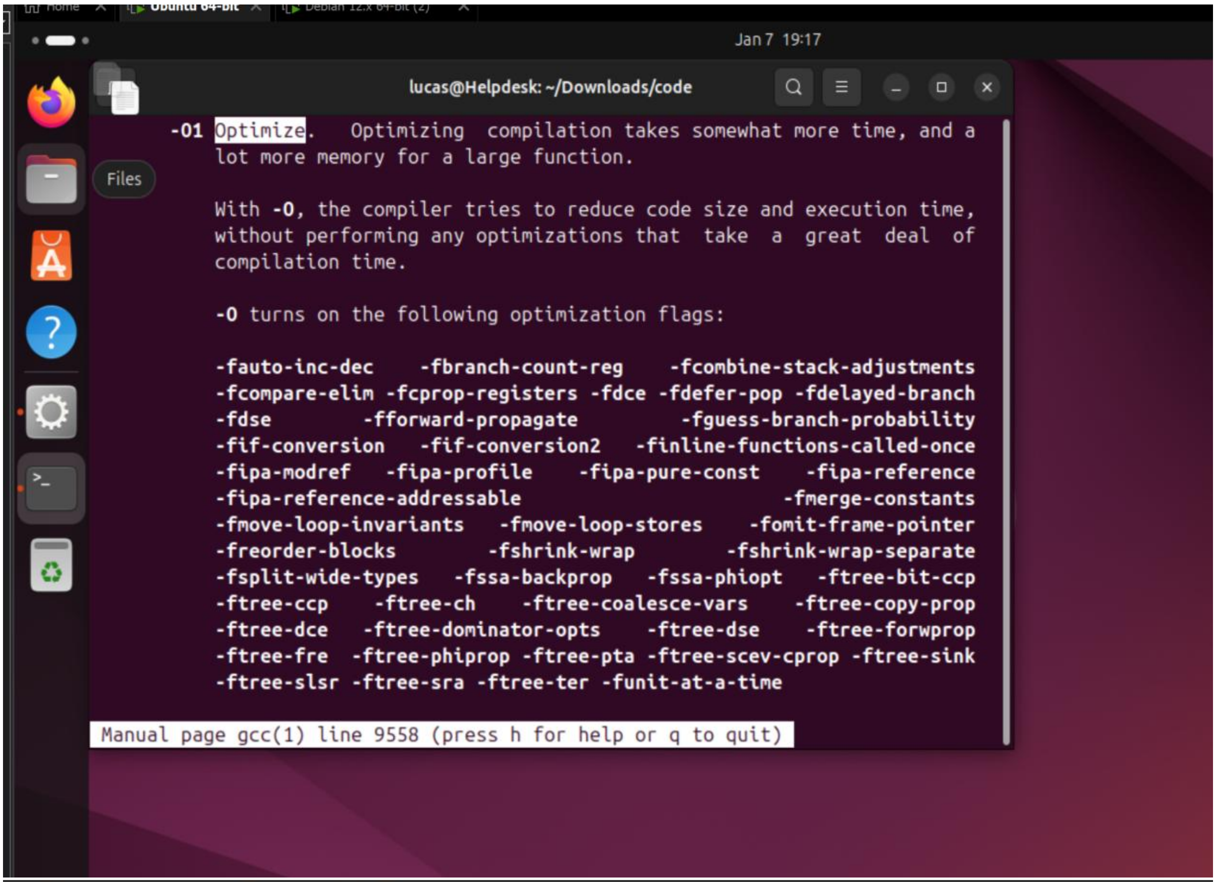
Assignment 4.4: Optimize

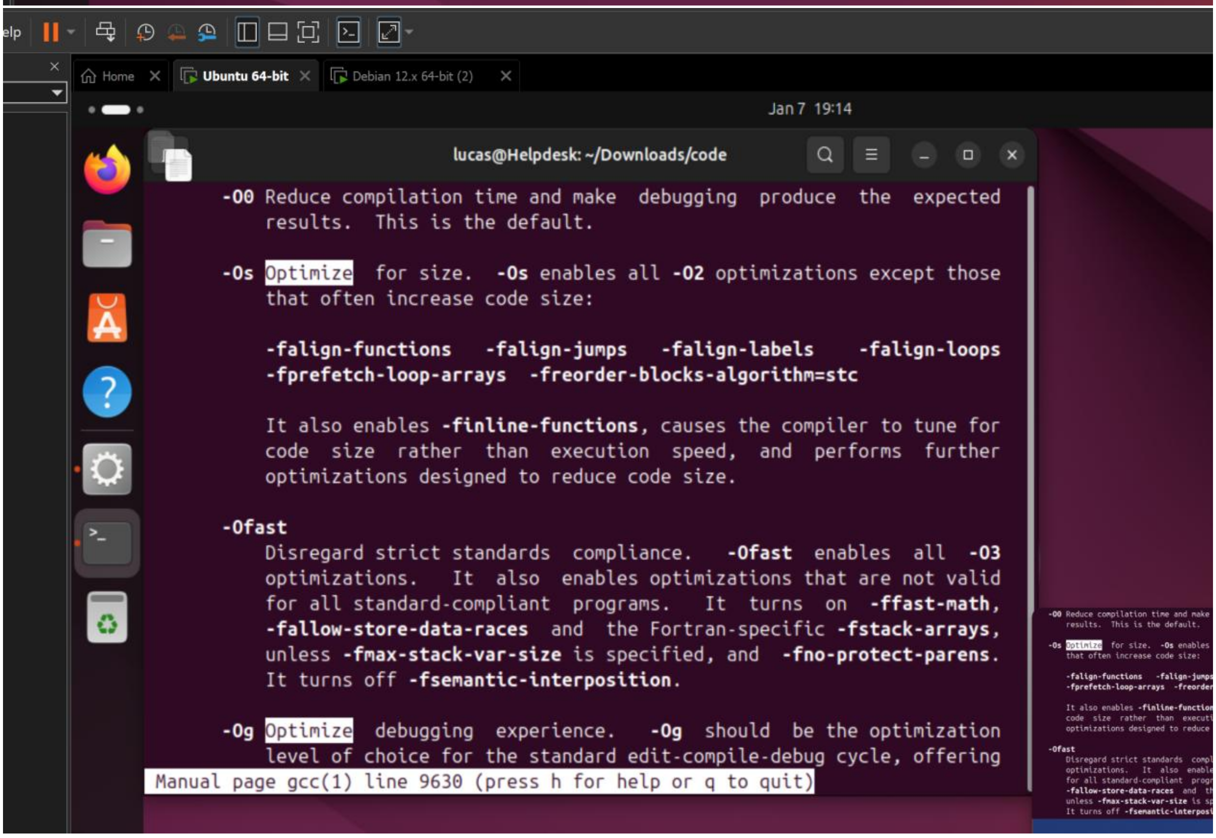
Take relevant screenshots of the following commands:

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.

   Ik heb in de man-page van gcc gezocht naar optimalisatie-opties. Daar vond ik de flags -O1, -O2, -O3 en -Ofast. Voor deze opdracht kies ik -O3, omdat dit zorgt voor maximale optimalisatie en betere uitvoersnelheid van het programma.

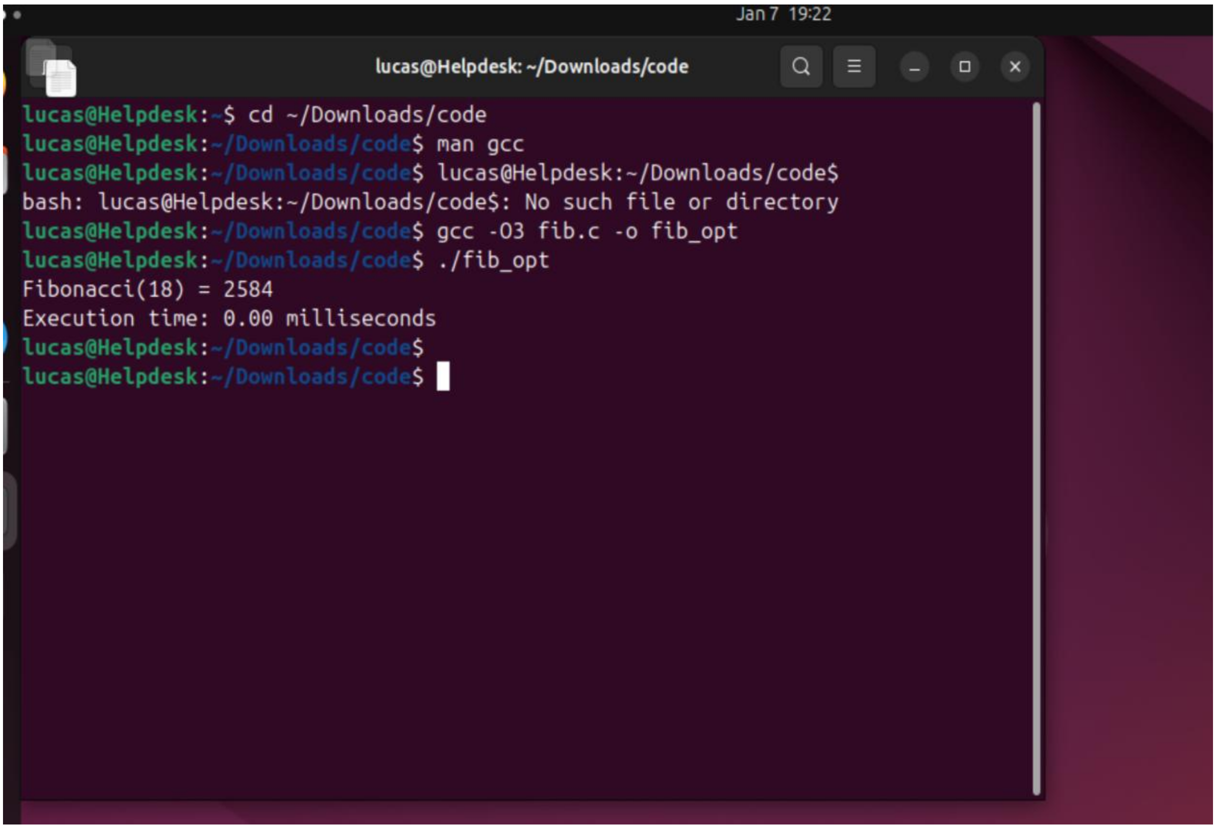



Jan 7 19:17
lucas@Helpdesk: ~/Downloads/code
Files
-O1 Optimize. Optimizing compilation takes somewhat more time, and a lot more memory for a large function.
With -O, the compiler tries to reduce code size and execution time, without performing any optimizations that take a great deal of compilation time.
-O turns on the following optimization flags:
-fauto-inc-dec -fbranch-count-reg -fcombine-stack-adjustments -fcompare-elim -fcprop-registers -fdce -fdefer-pop -fdelayed-branch -fdse -fforward-propagate -fguess-branch-probability -fif-conversion -fif-conversion2 -finline-functions-called-once -fipa-modref -fipa-profile -fipa-pure-const -fipa-reference -fipa-reference-addressable -fmerge-constants -fmove-loop-invariants -fmove-loop-stores -fomit-frame-pointer -freorder-blocks -fshrink-wrap -fshrink-wrap-separate -fsplit-wide-types -fssa-backprop -fssa-phiopt -ftree-bit-ccp -ftree-ccp -ftree-ch -ftree-coalesce-vars -ftree-copy-prop -ftree-dce -ftree-dominator-opts -ftree-dse -ftree-forwprop -ftree-fre -ftree-phiprop -ftree-pta -ftree-scev-cprop -ftree-sink -ftree-slsr -ftree-sra -ftree-ter -funit-at-a-time
Manual page gcc(1) line 9558 (press h for help or q to quit)


Home
Ubuntu 64-bit
Debian 12.x 64-bit (2)
Jan 7 19:14
lucas@Helpdesk: ~/Downloads/code
-O0 Reduce compilation time and make debugging produce the expected results. This is the default.
-Os Optimize for size. -Os enables all -O2 optimizations except those that often increase code size:
-falign-functions -falign-jumps -falign-labels -falign-loops -fprefetch-loop-arrays -freorder-blocks-algorithm=stc
It also enables -finline-functions, causes the compiler to tune for code size rather than execution speed, and performs further optimizations designed to reduce code size.
-Ofast
Disregard strict standards compliance. -Ofast enables all -O3 optimizations. It also enables optimizations that are not valid for all standard-compliant programs. It turns on -ffast-math, -fallow-store-data-races and the Fortran-specific -fstack-arrays, unless -fmax-stack-var-size is specified, and -fno-protect-parens. It turns off -fsemantic-interposition.
-Og Optimize debugging experience. -Og should be the optimization level of choice for the standard edit-compile-debug cycle, offering
Manual page gcc(1) line 9630 (press h for help or q to quit)

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

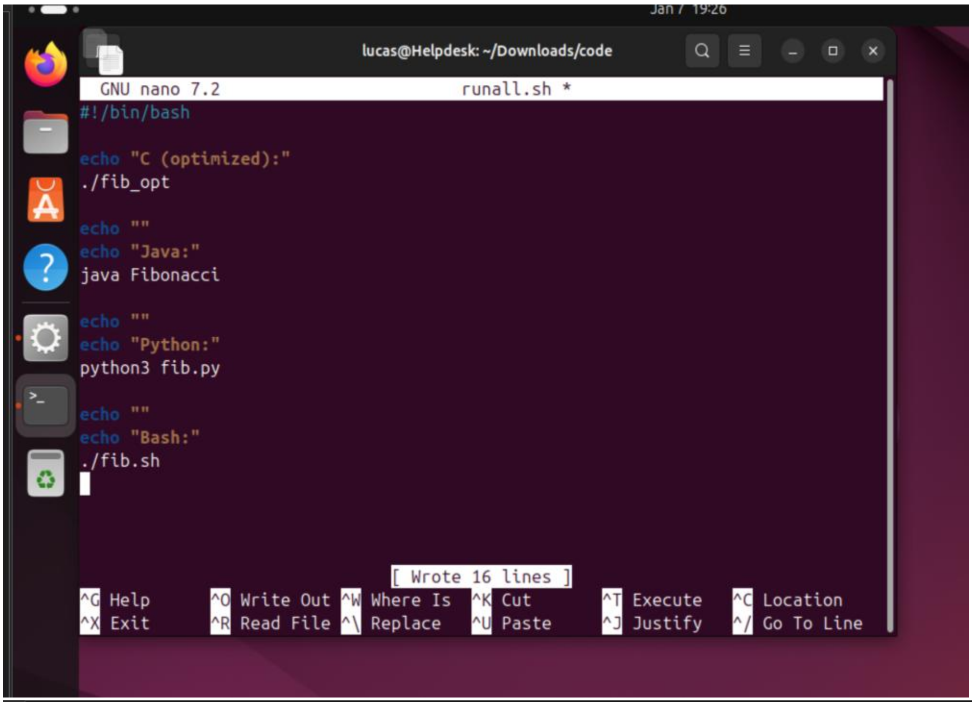
Ja, het geoptimaliseerde programma is sneller dan de niet-geoptimaliseerde versie. Door de optimalisatie voert de compiler extra verbeteringen uit waardoor de code efficiënter draait

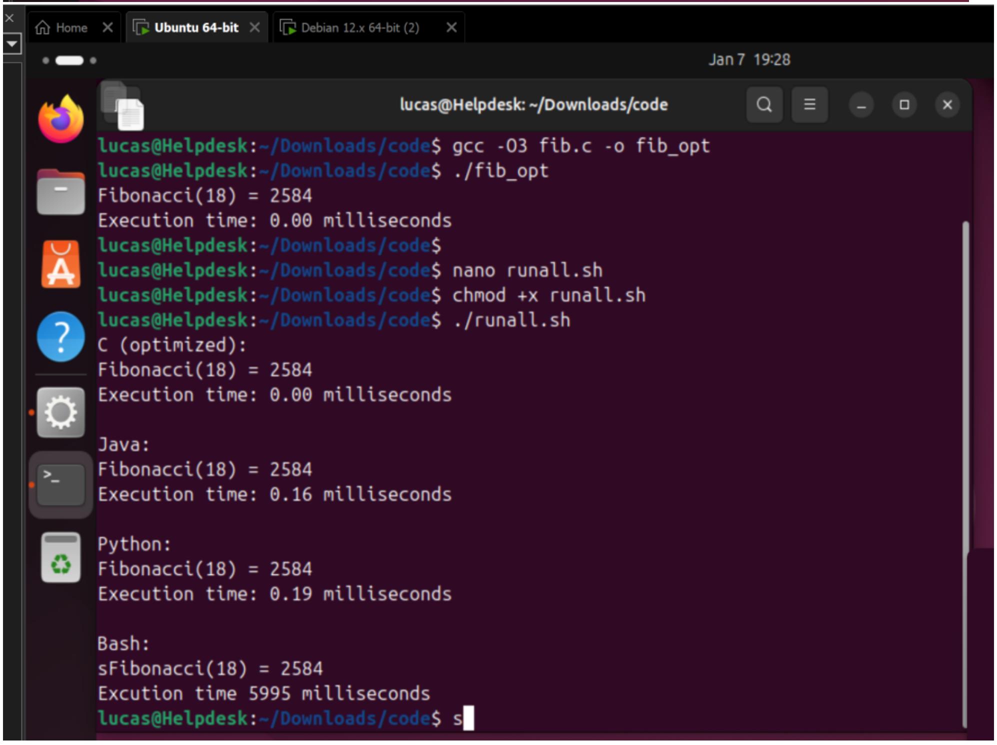


c) Run the newly compiled program. Is it true that it now performs the calculation faster?

```
lucas@Helpdesk:~$ cd ~/Downloads/code
lucas@Helpdesk:~/Downloads/code$ man gcc
lucas@Helpdesk:~/Downloads/code$ lucas@Helpdesk:~/Downloads/code$
bash: lucas@Helpdesk:~/Downloads/code$: No such file or directory
lucas@Helpdesk:~/Downloads/code$ gcc -O3 fib.c -o fib_opt
lucas@Helpdesk:~/Downloads/code$ ./fib_opt
Fibonacci(18) = 2584
Execution time: 0.00 milliseconds
lucas@Helpdesk:~/Downloads/code$
```

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.





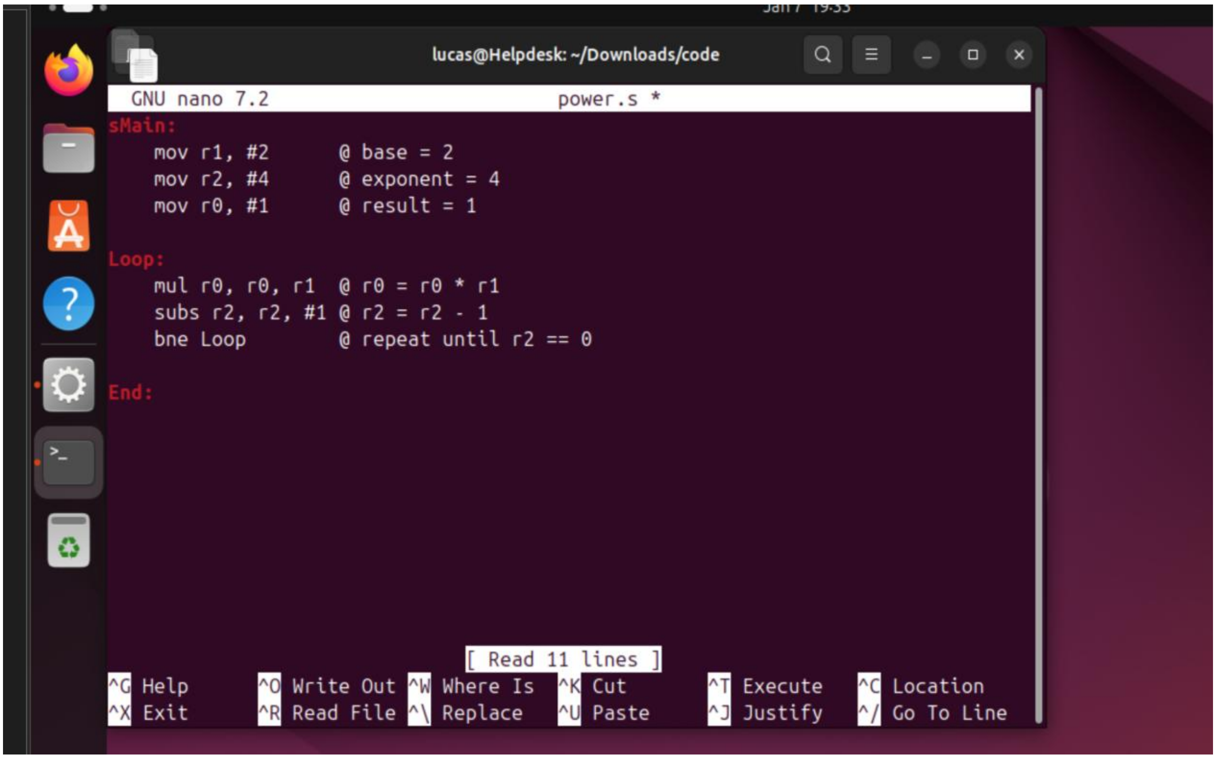
Assignment 4.5: More ARM Assembly

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**