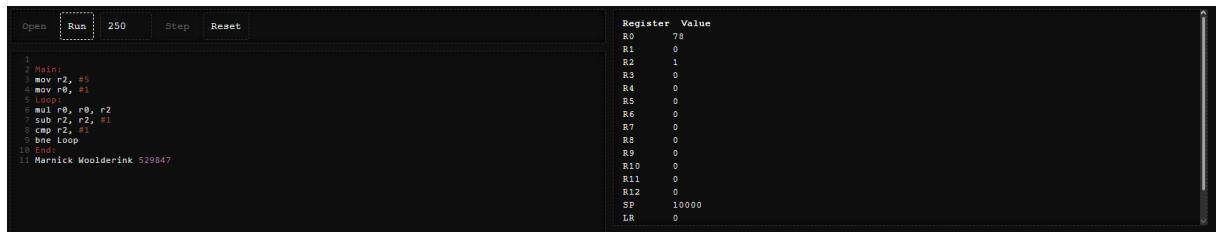


# Template Week 4 – Software

Student number:

## Assignment 4.1: ARM assembly

Screenshot of working assembly code of factorial calculation:



The screenshot shows a debugger interface with the following assembly code and register values:

```
Open Run 250 Step Reset

1 Main:
2     mov r2, #5
3     mov r0, #1
4     ldr r1, =Loop
5     mul r0, r0, r2
6     sub r2, r2, #1
7     cmp r2, #1
8     bne Loop
9     End:
10    Marnick Woolderink 529847
```

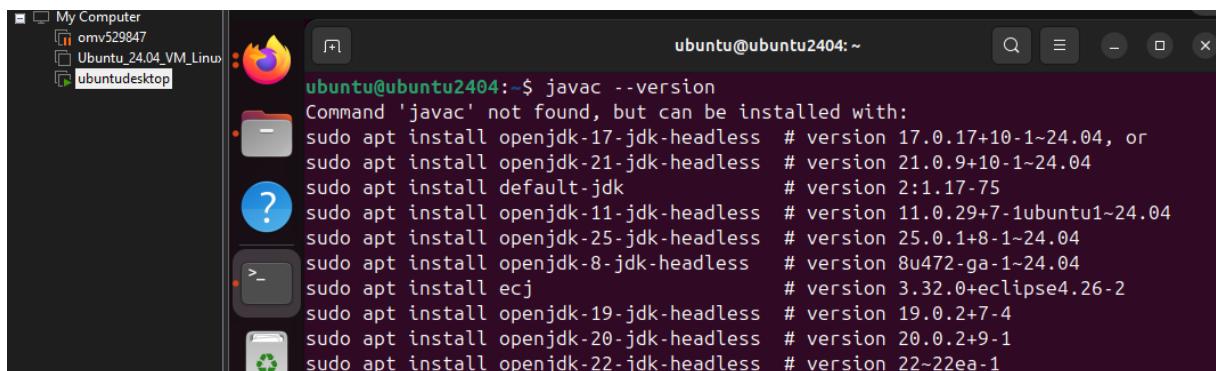
| Register | Value |
|----------|-------|
| R0       | 78    |
| R1       | 0     |
| R2       | 1     |
| R3       | 0     |
| R4       | 0     |
| R5       | 0     |
| R6       | 0     |
| R7       | 0     |
| R8       | 0     |
| R9       | 0     |
| R10      | 0     |
| R11      | 0     |
| R12      | 0     |
| SP       | 10000 |
| LR       | 0     |

Antwoord is in hexadecimal maar is wel 120

## Assignment 4.2: Programming languages

Take screenshots that the following commands work:

javac --version



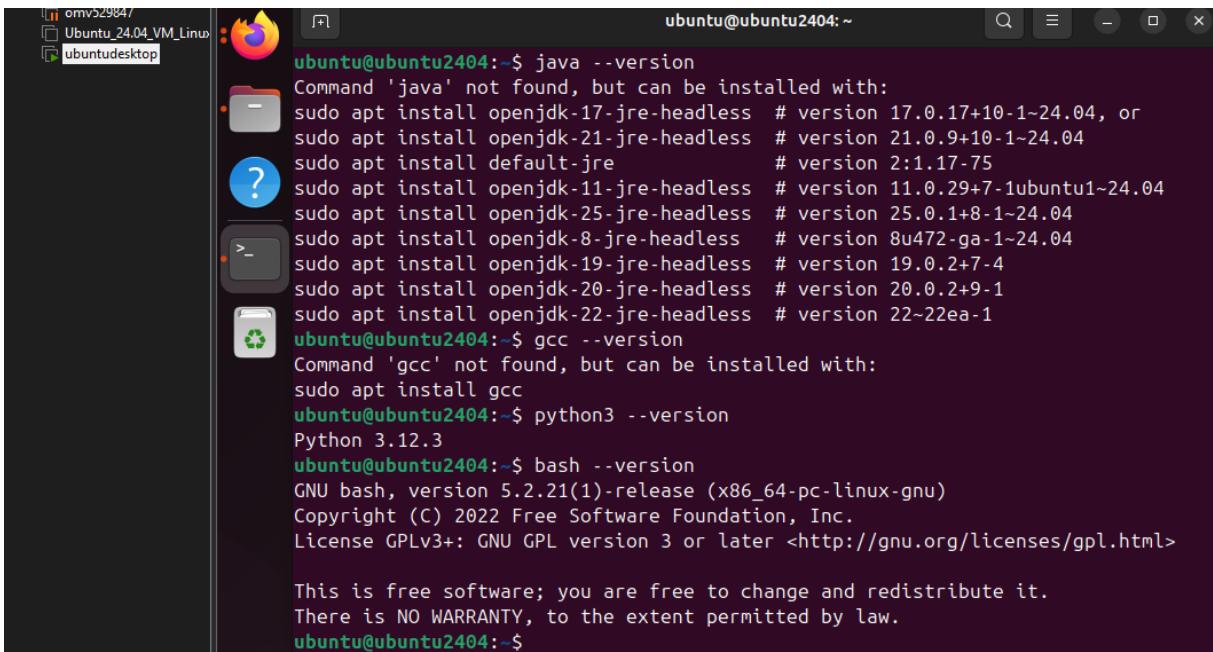
```
ubuntu@ubuntu2404:~$ javac --version
Command 'javac' not found, but can be installed with:
sudo apt install openjdk-17-jdk-headless  # version 17.0.17+10-1~24.04, or
sudo apt install openjdk-21-jdk-headless  # version 21.0.9+10-1~24.04
sudo apt install default-jdk           # version 2:1.17-75
sudo 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+eclipselink4.26-2
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
```

java --version

gcc --version

python3 --version

bash --version



The screenshot shows a Linux desktop environment with a terminal window open. The terminal window has a dark background and contains the following command-line session:

```
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?

Fibonacci.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?

Fibonacci.java ja: Fibonacci.class

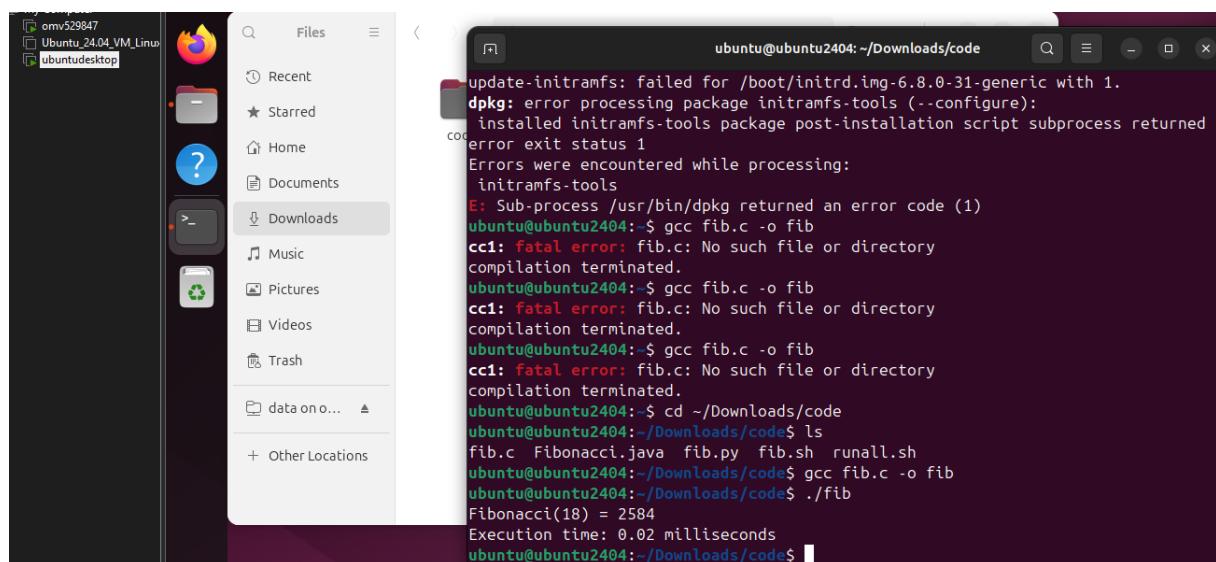
Fib.c ja: fib of fib.exe

Fib.py mogelijk: eventueel .pyc

Fib.sh nee

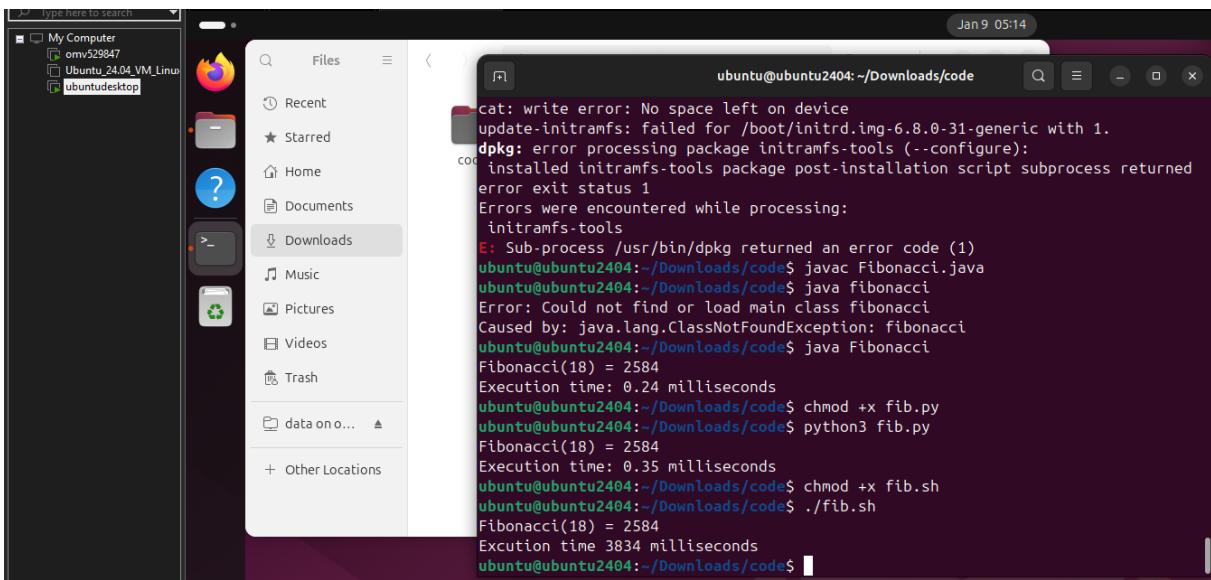
Take relevant screenshots of the following commands:

- Compile the source files where necessary
- Make them executable
- Run them
- Which (compiled) source code file performs the calculation the fastest?



The screenshot shows a Linux desktop environment with a dark theme. On the left is a file manager window titled 'Files' showing recent documents like 'Ubuntu 24.04.VM.Linux' and 'Ubuntudesktop'. On the right is a terminal window titled 'ubuntu@ubuntu2404: ~/Downloads/code'. The terminal output shows the following steps:

```
update-initramfs: failed for /boot/initrd.img-6.8.0-31-generic with 1.  
dpkg: error processing package initramfs-tools (--configure):  
  installed initramfs-tools package post-installation script subprocess returned  
  error exit status 1  
  Errors were encountered while processing:  
    initramfs-tools  
E: Sub-process /usr/bin/dpkg returned an error code (1)  
ubuntu@ubuntu2404:~$ gcc fib.c -o fib  
cc1: fatal error: fib.c: No such file or directory  
compilation terminated.  
ubuntu@ubuntu2404:~$ gcc fib.c -o fib  
cc1: fatal error: fib.c: No such file or directory  
compilation terminated.  
ubuntu@ubuntu2404:~$ gcc fib.c -o fib  
cc1: fatal error: fib.c: No such file or directory  
compilation terminated.  
ubuntu@ubuntu2404:~$ cd ~/Downloads/code  
ubuntu@ubuntu2404:~/Downloads/code$ ls  
fib.c Fibonacci.java fib.py fib.sh runall.sh  
ubuntu@ubuntu2404:~/Downloads/code$ gcc fib.c -o fib  
ubuntu@ubuntu2404:~/Downloads/code$ ./fib  
Fibonacci(18) = 2584  
Execution time: 0.02 milliseconds  
ubuntu@ubuntu2404:~/Downloads/code$
```



A screenshot of a Linux desktop environment. On the left, there's a file manager window titled "Files" showing a tree view of "My Computer" with items like "omv529847", "Ubuntu\_24.04\_VM\_Linux", and "ubuntudesktop". Below the tree view is a sidebar with links to "Recent", "Starred", "Home", "Documents", "Downloads" (which is currently selected), "Music", "Pictures", "Videos", and "Trash". To the right of the file manager is a terminal window titled "ubuntu@ubuntu2404: ~/Downloads/code". The terminal shows the following command-line session:

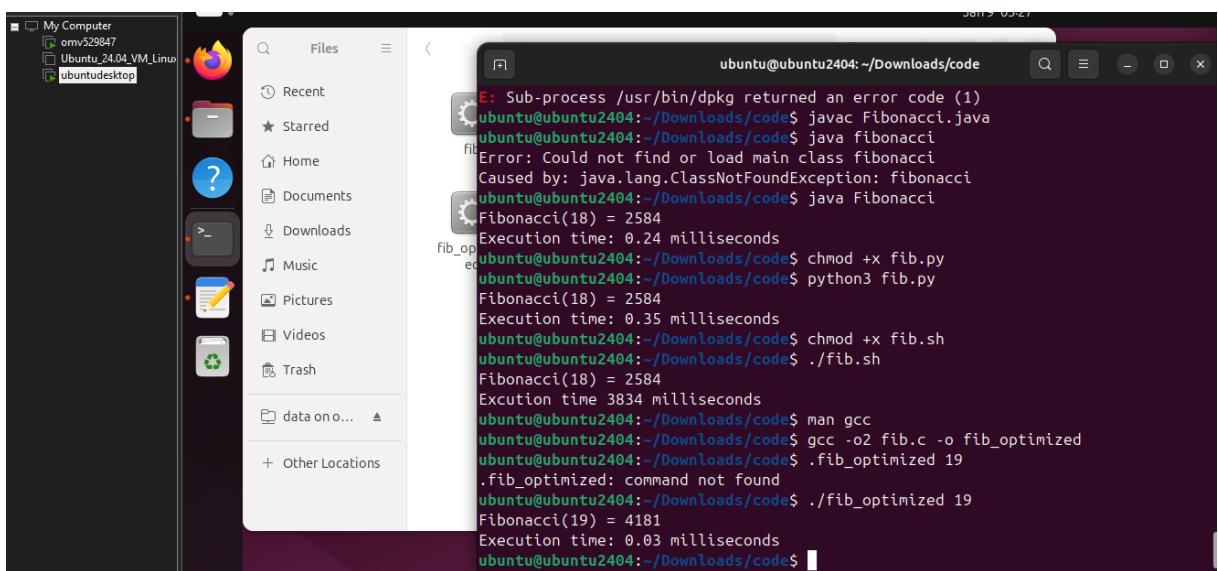
```
cat: write error: No space left on device
update-initramfs: failed for /boot/initrd.img-6.8.0-31-generic with 1.
dpkg: error processing package initramfs-tools (--configure):
  installed initramfs-tools package post-installation script subprocess returned
  error exit status 1
Errors were encountered while processing:
  initramfs-tools
E: Sub-process /usr/bin/dpkg returned an error code (1)
ubuntu@ubuntu2404:~/Downloads/code$ javac Fibonacci.java
ubuntu@ubuntu2404:~/Downloads/code$ java fibonacci
Error: Could not find or load main class fibonacci
Caused by: java.lang.ClassNotFoundException: fibonacci
ubuntu@ubuntu2404:~/Downloads/code$ java Fibonacci
Fibonacci(18) = 2584
Execution time: 0.24 milliseconds
ubuntu@ubuntu2404:~/Downloads/code$ chmod +x fib.py
ubuntu@ubuntu2404:~/Downloads/code$ python3 fib.py
Fibonacci(18) = 2584
Execution time: 0.35 milliseconds
ubuntu@ubuntu2404:~/Downloads/code$ chmod +x fib.sh
ubuntu@ubuntu2404:~/Downloads/code$ ./fib.sh
Fibonacci(18) = 2584
Execution time 3834 milliseconds
ubuntu@ubuntu2404:~/Downloads/code$
```

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?

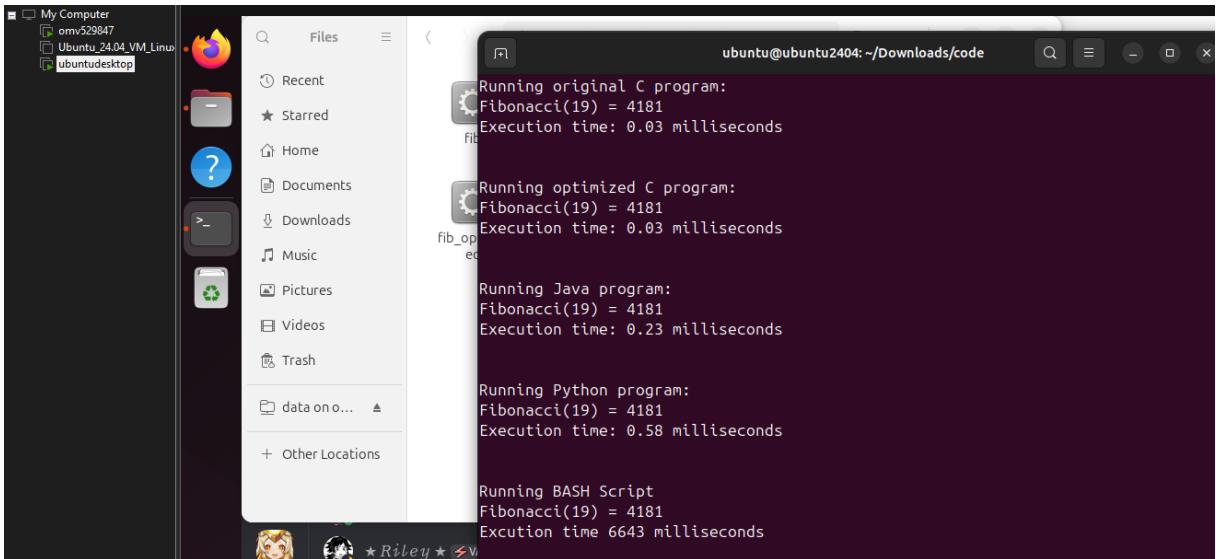


The screenshot shows a Linux desktop environment with a dark theme. On the left, there is a file manager window titled "Files" showing a list of locations including "My Computer", "Recent", and various folders like "Documents", "Downloads", "Music", "Pictures", "Videos", and "Trash". On the right, there is a terminal window titled "ubuntu@ubuntu2404: ~/Downloads/code". The terminal output shows the following steps and results:

```
E: Sub-process /usr/bin/dpkg returned an error code (1)
ubuntu@ubuntu2404:~/Downloads/code$ javac Fibonacci.java
ubuntu@ubuntu2404:~/Downloads/code$ java fibonacci
Error: Could not find or load main class fibonacci
Caused by: java.lang.ClassNotFoundException: fibonacci
ubuntu@ubuntu2404:~/Downloads/code$ java Fibonacci
Fibonacci(18) = 2584
Execution time: 0.24 milliseconds
fib_optimized$ chmod +x fib.py
ubuntu@ubuntu2404:~/Downloads/code$ python3 fib.py
Fibonacci(18) = 2584
Execution time: 0.35 milliseconds
ubuntu@ubuntu2404:~/Downloads/code$ chmod +x fib.sh
ubuntu@ubuntu2404:~/Downloads/code$ ./fib.sh
Fibonacci(18) = 2584
Execution time 3834 milliseconds
ubuntu@ubuntu2404:~/Downloads/code$ man gcc
ubuntu@ubuntu2404:~/Downloads/code$ gcc -O2 fib.c -o fib_optimized
ubuntu@ubuntu2404:~/Downloads/code$ ./fib_optimized 19
.fib_optimized: command not found
ubuntu@ubuntu2404:~/Downloads/code$ ./fib_optimized 19
Fibonacci(19) = 4181
Execution time: 0.03 milliseconds
ubuntu@ubuntu2404:~/Downloads/code$
```

It is not faster than 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.

| Register | Value |
|----------|-------|
| R0       | 10    |
| R1       | 2     |
| R2       | 0     |
| R3       | 0     |
| R4       | 0     |
| R5       | 0     |
| R6       | 0     |
| R7       | 0     |
| R8       | 0     |
| R9       | 0     |
| R10      | 0     |
| R11      | 0     |
| R12      | 0     |
| SP       | 10000 |
| LR       | 0     |

```

Open Run 250 Step Reset
1 Main:
2   mov r1, #2
3   mov r2, #4
4   mov r0, #1
5
6   Loop:
7   mul r0, r0, r1
8   sub r2, r2, #1
9   cmp r2, #0
10  bne Loop
11
12 End:
13 Marnick Woorderink 529847
14

```

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