

Template Week 4 – Software

Student number: 574642

Assignment 4.1: ARM assembly

Screenshot of working assembly code of factorial calculation:

Na 1 keer runnen krijg ik

R0 = 2

R1 = 4

R2 = 2

R3 = F

The screenshot shows a debugger interface with the following details:

- Assembly View:** Displays the assembly code for a factorial calculation. The code includes instructions for moving constants to registers (R2, R1, R0), performing a loop (Loop1) with multiplication and subtraction, and exiting the loop (End). The assembly code is:

```
1 Main:
2     mov r2, #5
3     mov r1, #1
4     mov r0, #0
5 Loop1:
6     mul r1, r1, r2
7     sub r2, r2, #1
8     cmn r2, r0
9     breq End
10    b Loop1
11 End:
```
- Registers View:** Shows the current values of all ARM registers (R0-R12, SP). All registers except R0, R1, and R2 have a value of 0.

Register	Value
R0	0
R1	78
R2	0
R3	0
R4	0
R5	0
R6	0
R7	0
R8	0
R9	0
R10	0
R11	0
R12	0
SP	10000
- Memory View:** Displays the memory dump starting at address 0x00001000. The memory contains the assembly code and some initial values.

Assignment 4.2: Programming languages

Take screenshots that the following commands work:

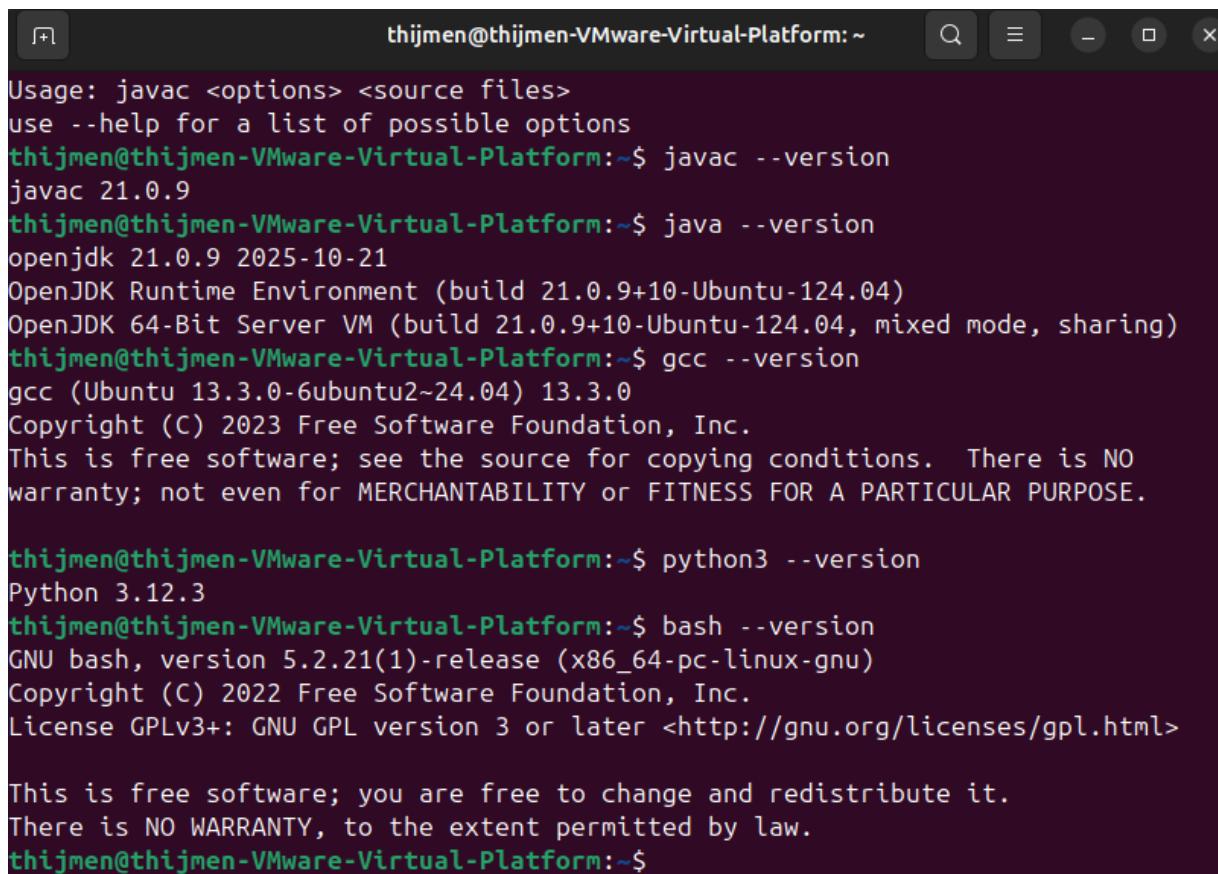
javac --version

java --version

gcc --version

python3 --version

```
bash --version
```



A screenshot of a terminal window titled "thijmen@thijmen-VMware-Virtual-Platform:~". The window contains the following text:

```
thijmen@thijmen-VMware-Virtual-Platform:~$ Usage: javac <options> <source files>
use --help for a list of possible options
thijmen@thijmen-VMware-Virtual-Platform:~$ javac --version
javac 21.0.9
thijmen@thijmen-VMware-Virtual-Platform:~$ java --version
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)
thijmen@thijmen-VMware-Virtual-Platform:~$ gcc --version
gcc (Ubuntu 13.3.0-6ubuntu2~24.04) 13.3.0
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.

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

Assignment 4.3: Compile

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

Fibonaci.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

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 omdat die gelijk uitvoerbaar is door de processor na het compilieren

How do I run a Java program?

Javac fibonacci.java compileren > java fibonacci runnen

How do I run a Python program?

Python3 fib.py

How do I run a C program?

Gcc fib.c -o fib compileren > ./fib runnen

How do I run a Bash script?

Chmod a+x fib.sh uitvoerbaar maken > ./fib.sh runnen

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

Fibonacci.java > fibonacci.class

Fib.c > fib

Fib.py > zelfde naam

Fib.sh > zelfde naam

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?

```
thijmen@thijmen-VMware-Virtual-Platform: ~/Documents/code$ ls -l
total 20
-rw-rw-r-- 1 thijmen thijmen 831 Jun  9  2023 fib.c
-rw-rw-r-- 1 thijmen thijmen 839 Jun  9  2023 Fibonacci.java
-rw-rw-r-- 1 thijmen thijmen 516 Jun  9  2023 fib.py
-rw-rw-r-- 1 thijmen thijmen 668 Jun  9  2023 fib.sh
-rw-rw-r-- 1 thijmen thijmen 249 Jun  9  2023 runall.sh
thijmen@thijmen-VMware-Virtual-Platform:~/Documents/code$ 

•
thijmen@thijmen-VMware-Virtual-Platform:~/Documents/code$ ls -l
total 20
-rw-rw-r-- 1 thijmen thijmen 831 Jun  9  2023 fib.c
-rw-rw-r-- 1 thijmen thijmen 839 Jun  9  2023 Fibonacci.java
-rw-rw-r-- 1 thijmen thijmen 516 Jun  9  2023 fib.py
-rw-rw-r-- 1 thijmen thijmen 668 Jun  9  2023 fib.sh
-rw-rw-r-- 1 thijmen thijmen 249 Jun  9  2023 runall.sh
thijmen@thijmen-VMware-Virtual-Platform:~/Documents/code$ javac Fibonacci.java
thijmen@thijmen-VMware-Virtual-Platform:~/Documents/code$ java Fibonacci
Fibonacci(18) = 2584
Execution time: 2.60 milliseconds
thijmen@thijmen-VMware-Virtual-Platform:~/Documents/code$ ls -l
total 24
-rw-rw-r-- 1 thijmen thijmen 831 Jun  9  2023 fib.c
-rw-rw-r-- 1 thijmen thijmen 1448 Jan  8 10:24 Fibonacci.class
-rw-rw-r-- 1 thijmen thijmen 839 Jun  9  2023 Fibonacci.java
-rw-rw-r-- 1 thijmen thijmen 516 Jun  9  2023 fib.py
-rw-rw-r-- 1 thijmen thijmen 668 Jun  9  2023 fib.sh
-rw-rw-r-- 1 thijmen thijmen 249 Jun  9  2023 runall.sh
thijmen@thijmen-VMware-Virtual-Platform:~/Documents/code$ 

thijmen@thijmen-VMware-Virtual-Platform:~/Documents/code$ python3 fib.py
Fibonacci(18) = 2584
Execution time: 1.61 milliseconds
thijmen@thijmen-VMware-Virtual-Platform:~/Documents/code$ ls -l
total 24
-rw-rw-r-- 1 thijmen thijmen 831 Jun  9  2023 fib.c
-rw-rw-r-- 1 thijmen thijmen 1448 Jan  8 10:24 Fibonacci.class
-rw-rw-r-- 1 thijmen thijmen 839 Jun  9  2023 Fibonacci.java
-rw-rw-r-- 1 thijmen thijmen 516 Jun  9  2023 fib.py
-rw-rw-r-- 1 thijmen thijmen 668 Jun  9  2023 fib.sh
-rw-rw-r-- 1 thijmen thijmen 249 Jun  9  2023 runall.sh
thijmen@thijmen-VMware-Virtual-Platform:~/Documents/code$ 
```

```

thijmen@thijmen-VMware-Virtual-Platform:~/Documents/code$ chmod a+x fib.sh
thijmen@thijmen-VMware-Virtual-Platform:~/Documents/code$ ./fib.sh
Fibonacci(18) = 2584
Excution time 17526 milliseconds
thijmen@thijmen-VMware-Virtual-Platform:~/Documents/code$ ls -l
total 40
-rwxrwxr-x 1 thijmen thijmen 16136 Jan  8 10:26 fib
-rw-rw-r-- 1 thijmen thijmen   831 Jun  9 2023 fib.c
-rw-rw-r-- 1 thijmen thijmen  1448 Jan  8 10:24 Fibonacci.class
-rw-rw-r-- 1 thijmen thijmen   839 Jun  9 2023 Fibonacci.java
-rw-rw-r-- 1 thijmen thijmen   516 Jun  9 2023 fib.py
-rwxrwxr-x 1 thijmen thijmen   668 Jun  9 2023 fib.sh
-rw-rw-r-- 1 thijmen thijmen   249 Jun  9 2023 runall.sh
thijmen@thijmen-VMware-Virtual-Platform:~/Documents/code$ 

-rw-rw-r-- 1 thijmen thijmen   249 Jun  9 2023 runall.sh
thijmen@thijmen-VMware-Virtual-Platform:~/Documents/code$ gcc fib.c -o fib
thijmen@thijmen-VMware-Virtual-Platform:~/Documents/code$ ./fib
Fibonacci(18) = 2584
Execution time: 0.04 milliseconds
thijmen@thijmen-VMware-Virtual-Platform:~/Documents/code$ ls -l
total 40
-rwxrwxr-x 1 thijmen thijmen 16136 Jan  8 10:26 fib
-rw-rw-r-- 1 thijmen thijmen   831 Jun  9 2023 fib.c
-rw-rw-r-- 1 thijmen thijmen  1448 Jan  8 10:24 Fibonacci.class
-rw-rw-r-- 1 thijmen thijmen   839 Jun  9 2023 Fibonacci.java
-rw-rw-r-- 1 thijmen thijmen   516 Jun  9 2023 fib.py
-rw-rw-r-- 1 thijmen thijmen   668 Jun  9 2023 fib.sh
-rw-rw-r-- 1 thijmen thijmen   249 Jun  9 2023 runall.sh

```

Hierboven alles gecompiled met bash + alle loadtimes

Assignment 4.4: Optimize

Take relevant screenshots of the following commands:

- 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(1)                               GNU                               GCC(1)

NAME
    gcc - GNU project C and C++ compiler

SYNOPSIS
    gcc [-c|-S|-E] [-std=standard]
        [-g] [-pg] [-Olevel]
        [-Wwarn...]
        [-Wpedantic]
        [-Idir...]
        [-Ldir...]
        [-Dmacro[=defn]...]
        [-Umacro]
        [-foption...]
        [-mmachine-option...]
        [-o outfile] [infile...]

    Only the most useful options are listed here; see below for the remainder. g++ accepts mostly the same
    options as gcc.

DESCRIPTION
    When you invoke GCC, it normally does preprocessing, compilation, assembly and linking. The "overall options"
    allow you to stop this process at an intermediate stage. For example, the -c option says not to run the
    linker. Then the output consists of object files output by the assembler.

    Other options are passed on to one or more stages of processing. Some options control the preprocessor and
    others the compiler itself. Yet other options control the assembler and linker; most of these are not
    documented here, since you rarely need to use any of them.

    Most of the command-line options that you can use with GCC are useful for C programs; when an option is only
    useful with another language (usually C++), the explanation says so explicitly. If the description for a
    particular option does not mention a source language, you can use that option with all supported languages.

```

Ik heb info gcc gedaan om te kijken welke paramiter ik moest hebben, dit kon ik zo snel niet vinden dus ik heb het opgezocht in dit geval is 03 het snelst.

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

```

thijmen@thijmen-Virtual-Platform:~/Documents/code$ gcc fib.c -o fib
thijmen@thijmen-Virtual-Platform:~/Documents/code$ ./fib
Fibonacci(18) = 2584
Execution time: 0.03 milliseconds
thijmen@thijmen-Virtual-Platform:~/Documents/code$ gcc -O3 fib.c -o fib_optimized
gcc: error: unrecognized command-line option '-O3'
thijmen@thijmen-Virtual-Platform:~/Documents/code$ gcc -O3 fib.c -o fib_optimized
thijmen@thijmen-Virtual-Platform:~/Documents/code$ ./fib optimized
Fibonacci(18) = 2584
Execution time: 0.03 milliseconds
thijmen@thijmen-Virtual-Platform:~/Documents/code$ 

```

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

Nee, het is bij mij niet sneller

- 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.

The screenshot shows the OakSim assembly debugger interface. The assembly code is as follows:

```
1 Main:  
2     mov r0, #1  
3     mov r1, #2  
4     mov r2, #4  
5 Loop:  
6     mul r0, r0, r1  
7     subs r2, r2, #1  
8     bne Loop  
9 End:
```

The register values are:

Register	Value
R0	1
R1	2
R2	4
R3	0
R4	0
R5	0
R6	0
R7	0
R8	0
R9	0

The memory dump shows the value 57464 at address 0x000010000:

```
0x000010000: 01 00 A0 E3 02 10 A0 E3 04 20 A0 E3 FF FF FF 1A  
0x000010010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
0x000010020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
0x000010030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
0x000010040: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
0x000010050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
0x000010060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
0x000010070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
0x000010080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
0x000010090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
0x0000100A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
0x0000100B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
0x0000100C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
0x0000100D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

The status bar shows the address 57464.

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