Week 4: Software (ch.8, ch.9)

Student number: 499193

Low level programming languages (ch.8)

Assignment 4.1: introduction to ARM assembly

We are going to use the following online arm assembly simulator:
https://wunkolo.github.io/OakSim/

Now for the assignment you need to perform. Empty the code editor again. You may only use the assembly instructions below. With your acquired knowledge about labels and the following ARM assembly instructions…

- mov
- sub
- mul
- cmp
- beq
- b

… it should be possible to make a loop that calculates the factorial from a number. What is a factorial? There's a button on your calculator with an exclamation mark on it!. 5! = 5 x 4 x 3 x 2 x 1 = 120. What is 120 in hexadecimal again? You start by placing the number 5 in a register. Complete the code below:

```
main:

mov r2, #5
mov r1, #1

loop:

cmp   r2,
beq   End
mul   r1, r1, r2
sub   r2, r2, #1
b     loop

end:

bx    lr
```

The result of 5!, you save in register r1. Show the result to your teacher.

Assignment 4.2: Programming languages

• What versions of the compilers/interpreters for programming languages are installed?

gcc (Ubuntu 13.3.0-6ubuntu2~24.04) 13.3.0

javac 21.0.9

openjdk 21.0.9 2025-10-21
OpenJDK Runtime Environment (build 21.0.9+10-Ubuntu-124.04)

Python 3.12.3

GNU bash, version 5.2.21(1)-release (x86_64-pc-linux-gnu)

Assignment 4.3: Compile

You should see 4 source code files:
• Fibonacci.java
• fib.c
• fib.py
• fib.sh

There is one thing that is different on Linux than on Windows. Compiled files or script files are not directly executable. To do this, you must first give them permissions. I can already tell you that fib.sh is a Bash script file that doesn't need to be compiled, but still needs to be made executable. You can do this with the command: sudo chmod a+x fib.sh If you show the contents of the directory with the ls command, you will see that the file has changed color and has also been given executable rights. You can run it with this command: sudo ./fib.sh

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

Java: 0.21 milliseconds
Python: 0.35 milliseconds
bash: 5766 milliseconds
c: 0.01 milliseconds

fib.c performs the calculation the fastest (0.01ms).

Assignment 4.4: Optimize The previous assignment should show that a program written in C runs faster than a program written in Java, Python or Bash. If that is not the case, then we must do something about it.

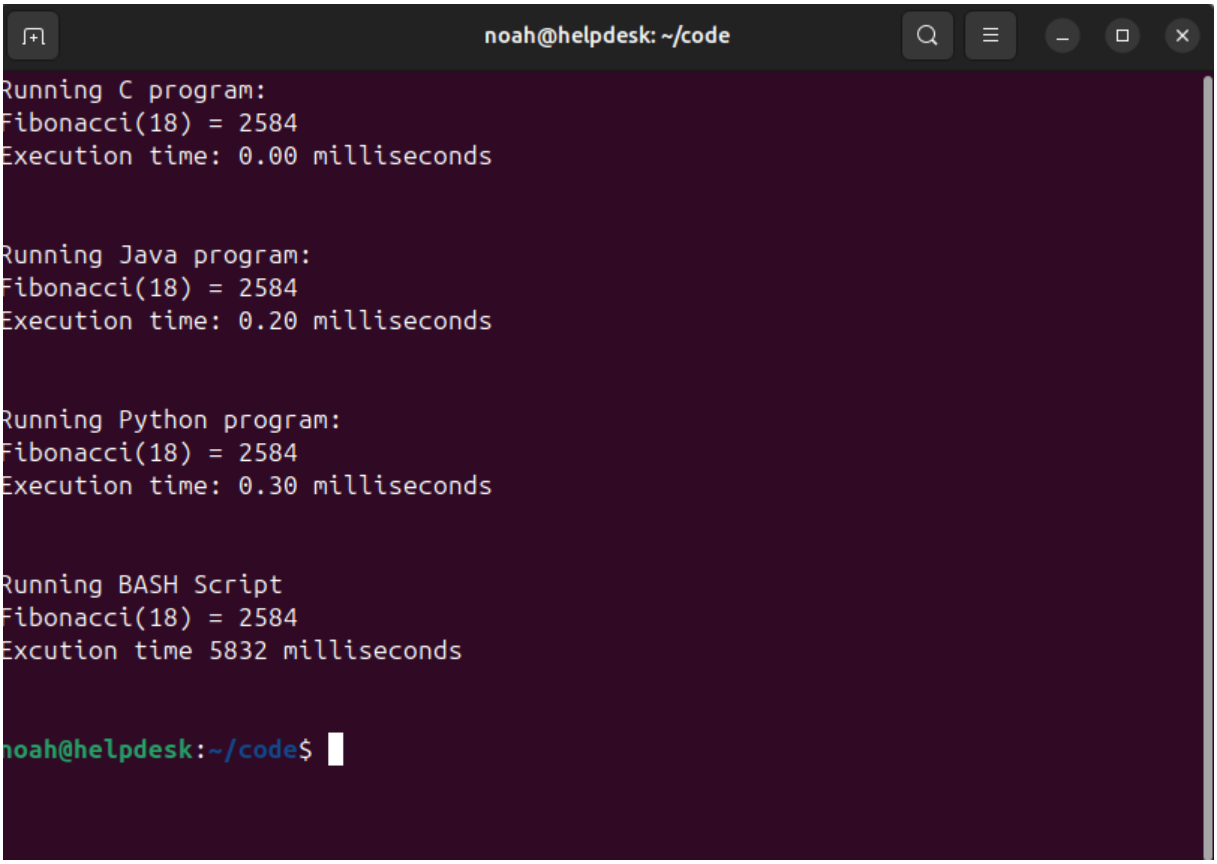
a) Figure out which parameters you need to pass to the gcc compiler so that the compiler performs a number of optimizations that will ensure that the compiled source code will run faster. Tip! The parameters are usually a letter followed by a number. Also read page 191 of your book, but find a better 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?

gcc -O3 fib.c -o fiboptimized
(0.00ms)

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 24 = 16. Use iteration to calculate the result. Store the result in r0.

```
main:
        mov     r1, #2      @ base = 2
        mov     r2, #4      @ exponent = 4
        mov     r0, #1      @ result starts at 1

loop:
        mul     r0, r0, r1  @ r0 = r0 * 2
        subs    r2, r2, #1  @ exponent--
        bne     loop        @ repeat until r2 == 0

end:
        b       end
```

Complete the code. See the PowerPoint slides of week 4.