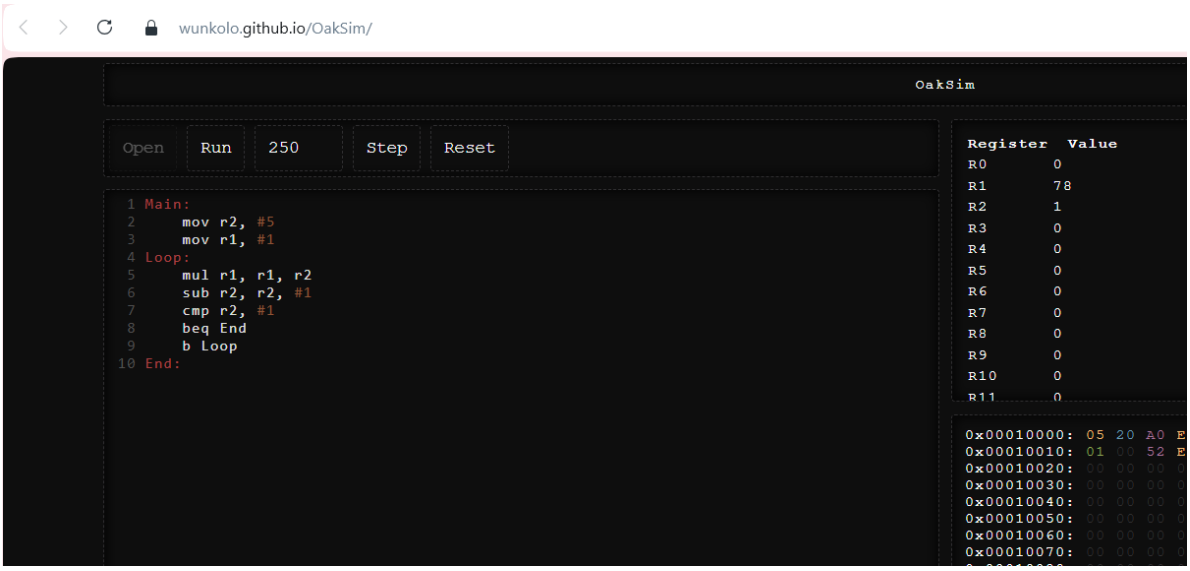
Template Week 4 – Software

Student number: 547201

Assignment 4.1: ARM assembly

Screenshot of working assembly code of factorial calculation:



mov r2, #5 - Put the value 5 into register r2.

mov r1, #1 - Put the value 1 into register r1.

mul r1, r1, r2 - Multiply r1 by r2, store the result in r1.

sub r2, r2, #1 - r2 = r2 – 1

cmp r2, #1 - Compare r2 with 1.

beq End - If r2 equals 1, jump to End

b Loop - jump back to Loop

Assignment 4.2: Programming languages

Take screenshots that the following commands work:

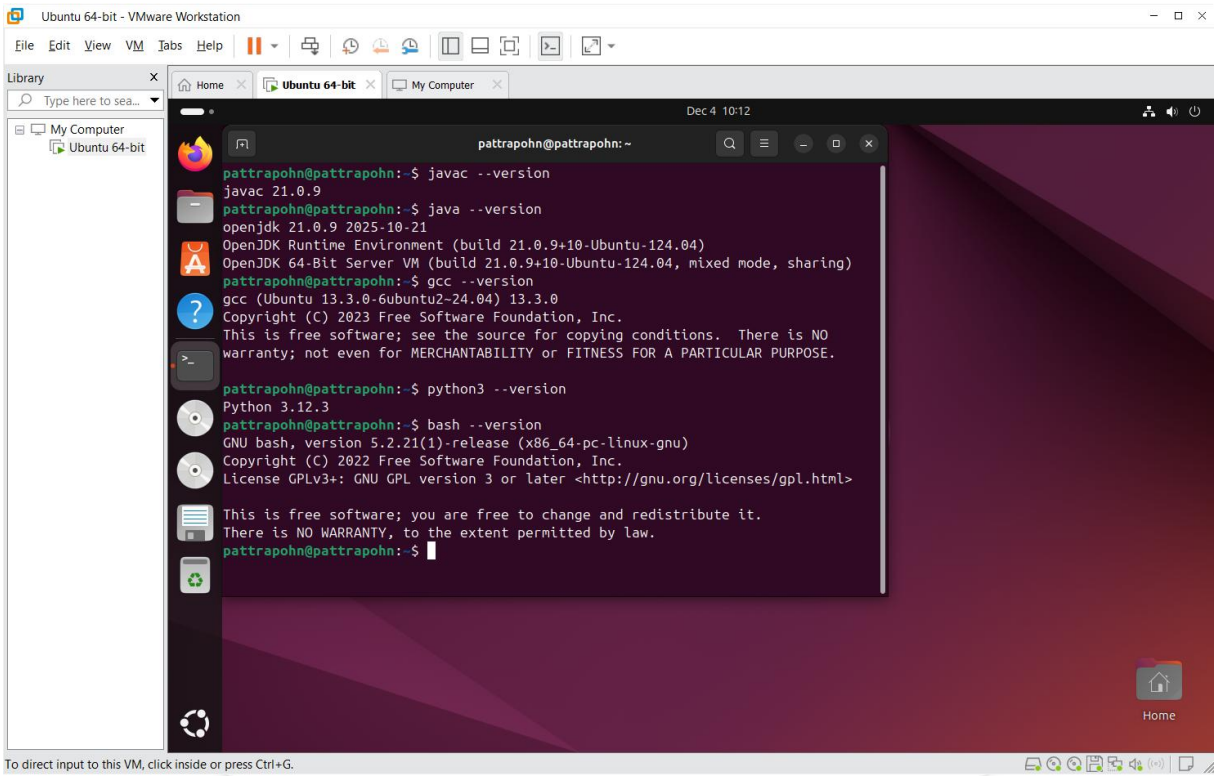
javac --version

java --version

gcc --version

python3 --version

bash --version


Ubuntu 64-bit - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to sea...
My Computer
Ubuntu 64-bit
Home
Ubuntu 64-bit
My Computer
Dec 4 10:12
pattrapohn@pattrapohn: ~
pattrapohn@pattrapohn:~$ javac --version
javac 21.0.9
pattrapohn@pattrapohn:~$ 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)
pattrapohn@pattrapohn:~$ 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.
pattrapohn@pattrapohn:~$ python3 --version
Python 3.12.3
pattrapohn@pattrapohn:~$ 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.
pattrapohn@pattrapohn:~$
Home
To direct input to this VM, click inside or press Ctrl+G.

Assignment 4.3: Compile

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

- fib.c
- Fibonacci.java

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 program

How do I run a Java program?

- java Fibonacci

How do I run a Python program?

- python3 fib.py

How do I run a C program?

- ./fib

How do I run a Bash script?

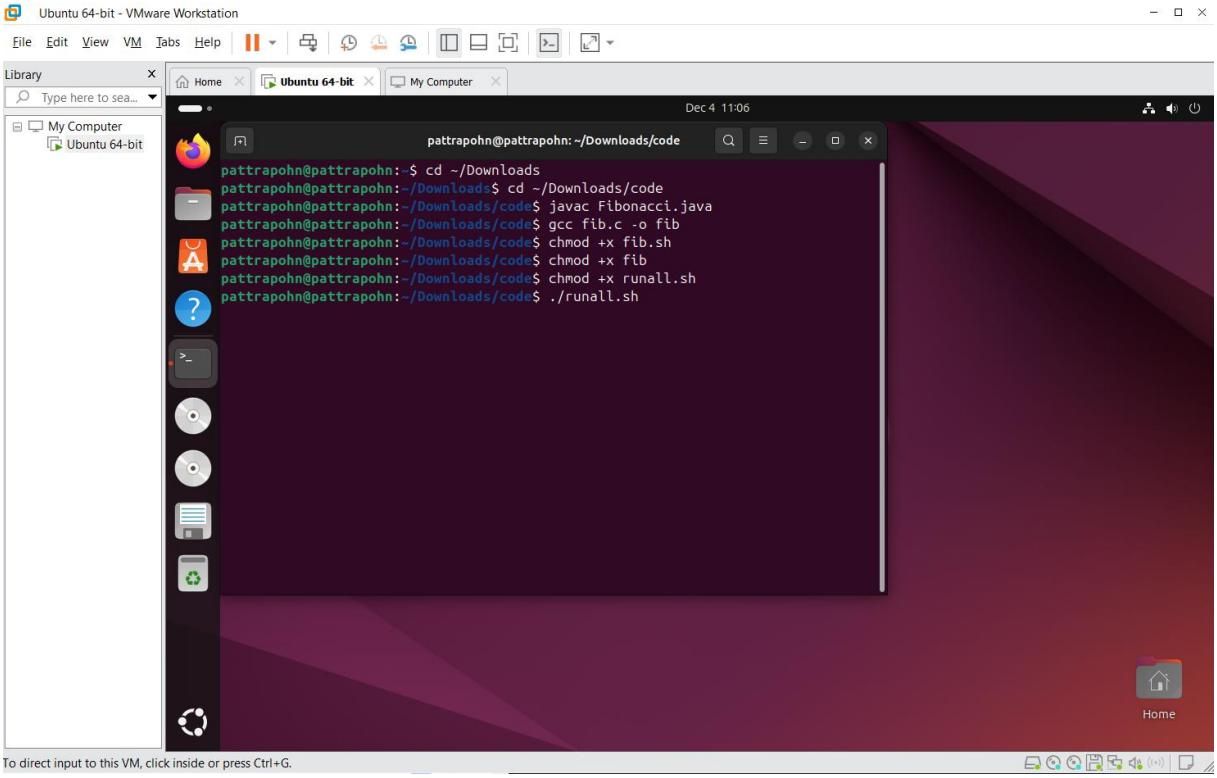
- ./fib.sh

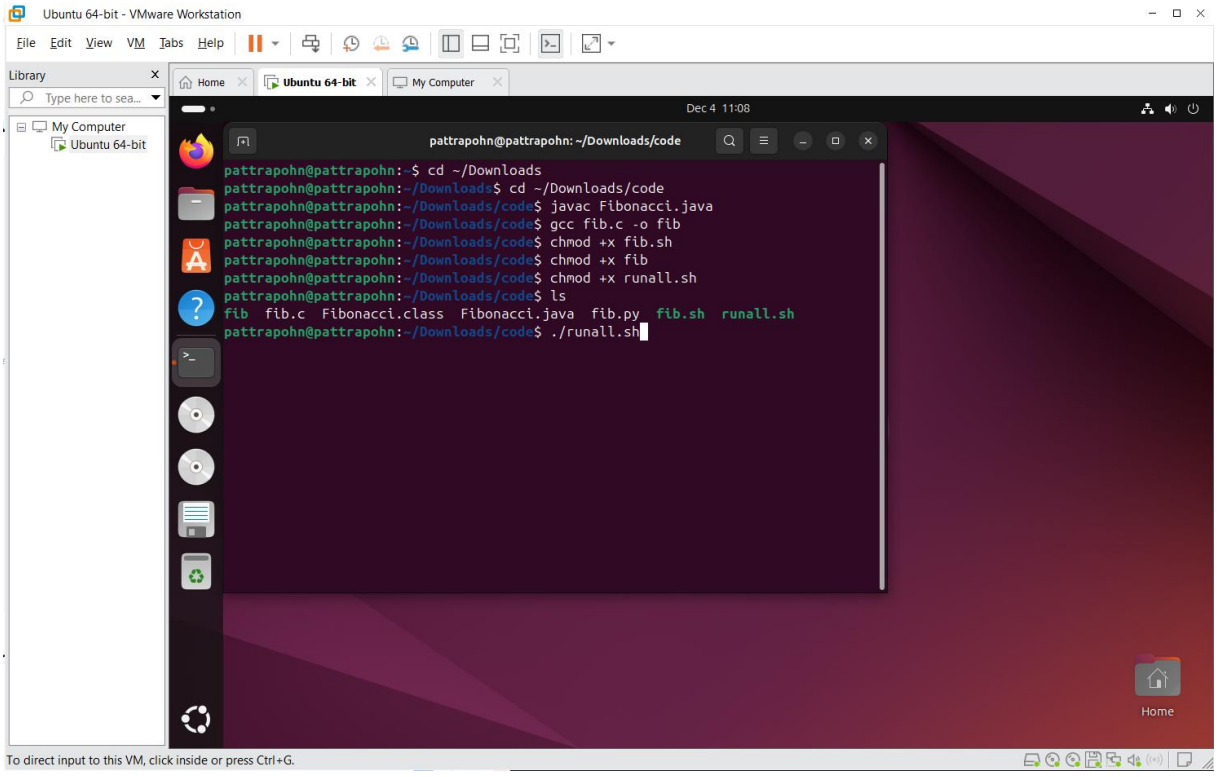
If I compile the above source code, will a new file be created? If so, which file?

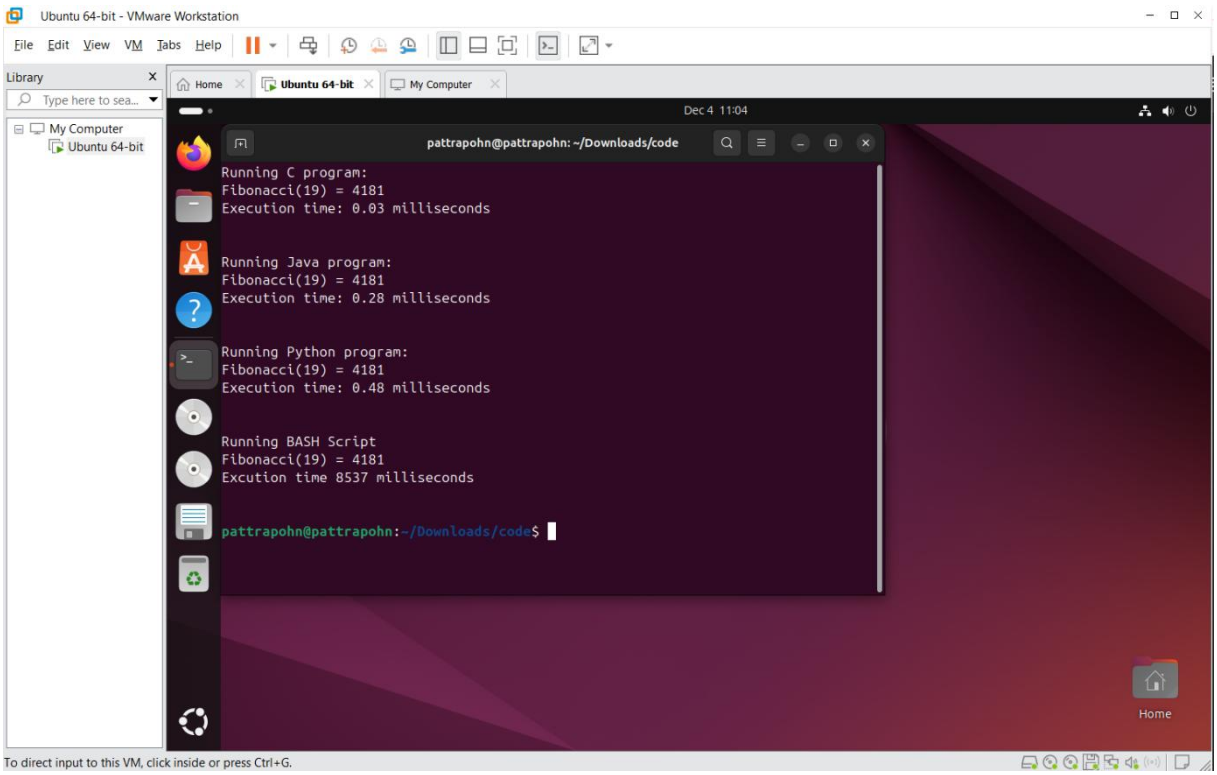
- Fibonacci.class from java compilation and fib from C compilation

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?


Ubuntu 64-bit - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to sea...
My Computer
Ubuntu 64-bit
Home
Ubuntu 64-bit
My Computer
Dec 4 11:06
pattrapohn@pattrapohn: ~/Downloads/code
pattrapohn@pattrapohn:~$ cd ~/Downloads
pattrapohn@pattrapohn:~/Downloads$ cd ~/Downloads/code
pattrapohn@pattrapohn:~/Downloads/code$ javac Fibonacci.java
pattrapohn@pattrapohn:~/Downloads/code$ gcc fib.c -o fib
pattrapohn@pattrapohn:~/Downloads/code$ chmod +x fib.sh
pattrapohn@pattrapohn:~/Downloads/code$ chmod +x fib
pattrapohn@pattrapohn:~/Downloads/code$ chmod +x runall.sh
pattrapohn@pattrapohn:~/Downloads/code$ ./runall.sh
Home
To direct input to this VM, click inside or press Ctrl+G.


Ubuntu 64-bit - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to sea...
My Computer
Ubuntu 64-bit
Home
Ubuntu 64-bit
My Computer
Dec 4 11:08
pattrapohn@pattrapohn: ~/Downloads/code
pattrapohn@pattrapohn:~$ cd ~/Downloads
pattrapohn@pattrapohn:~/Downloads$ cd ~/Downloads/code
pattrapohn@pattrapohn:~/Downloads/code$ javac Fibonacci.java
pattrapohn@pattrapohn:~/Downloads/code$ gcc fib.c -o fib
pattrapohn@pattrapohn:~/Downloads/code$ chmod +x fib.sh
pattrapohn@pattrapohn:~/Downloads/code$ chmod +x fib
pattrapohn@pattrapohn:~/Downloads/code$ chmod +x runall.sh
pattrapohn@pattrapohn:~/Downloads/code$ ls
fib fib.c Fibonacci.class Fibonacci.java fib.py fib.sh runall.sh
pattrapohn@pattrapohn:~/Downloads/code$ ./runall.sh
Home
To direct input to this VM, click inside or press Ctrl+G.



- ./fib or C program run the fastest

Assignment 4.4: Optimize

Take relevant screenshots of the following commands:

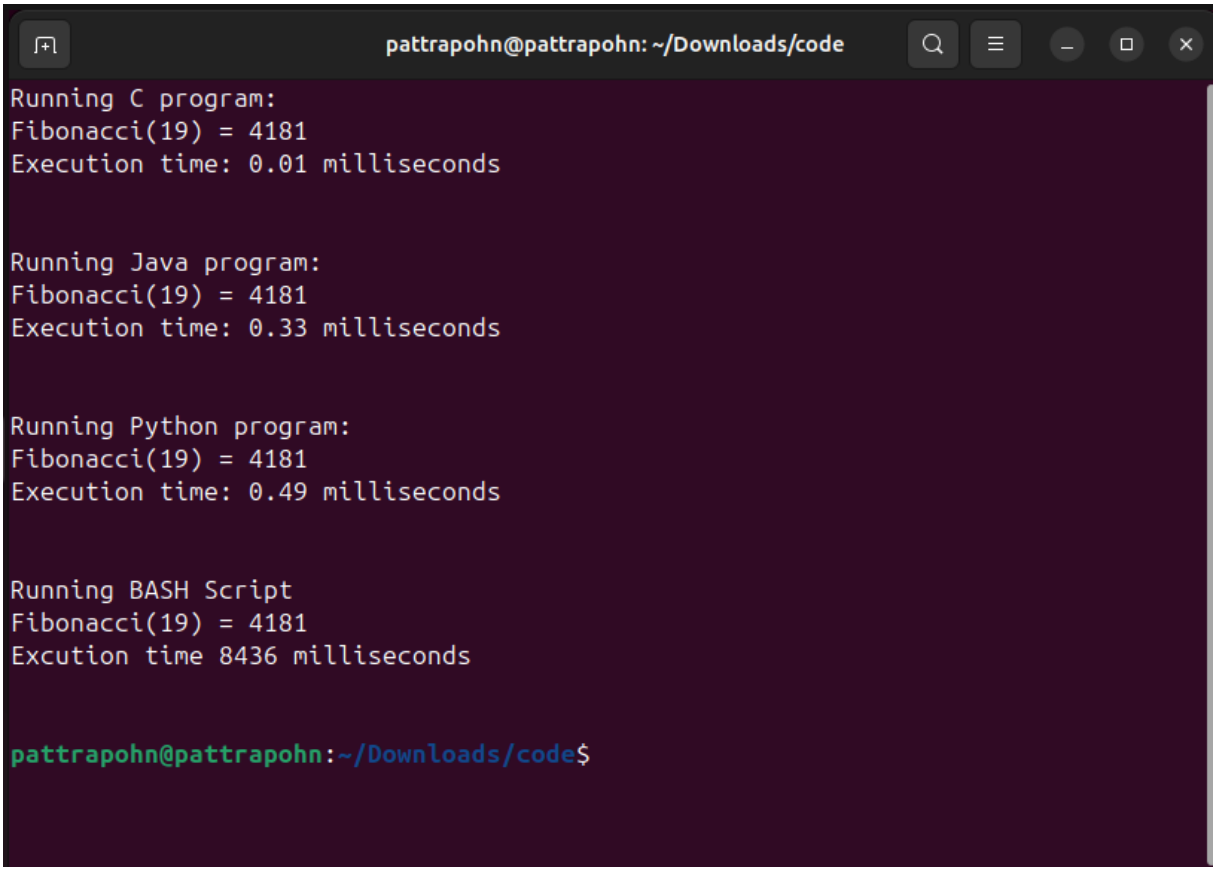
a) Figure out which parameters you need to pass to **the gcc** compiler so that the compiler performs a number of optimizations that will ensure that the compiled source code will run faster. **Tip!** The parameters are usually a letter followed by a number. Also read **page 191** of your book, but find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

```
pattrapohn@pattrapohn:~/Downloads/code$ gcc -O1 -o fib fib.c
pattrapohn@pattrapohn:~/Downloads/code$ ./fib
Fibonacci(18) = 2584
Execution time: 0.01 milliseconds
pattrapohn@pattrapohn:~/Downloads/code$ gcc -O3 -o fib fib.c
pattrapohn@pattrapohn:~/Downloads/code$ ./fib
Fibonacci(18) = 2584
Execution time: 0.00 milliseconds
pattrapohn@pattrapohn:~/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
	mov r0, #1
Loop:
	cmp r2, #0	if it's 0 it means it has mul 4 times
	beq End
	mul r0, r0, r1	mul 1 with 2 = 2, mul 2 with 2 = 4 . . .
	sub r2, r2, #1	4 - 1 = 3, 3-1 = 2 . . .
	b Loop
End:
```

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

Screenshot of the completed code here.