

## **Template Week 4 – Software**

Student number: 564595

## Assignment 4.1: ARM assembly

Screenshot of working assembly code of factorial calculation:

OakSim

Open	Run	250	Step	Reset																						
<pre>1 Main: 2     mov r2, #5 3     mov r1, r2 4 5 Loop: 6     sub r2, r2, #1 7     mul r1, r1, r2 8     cmp r2, #1 9     beq End 10    b Loop 11 End:</pre>																										
<table border="1"><thead><tr><th>Register</th><th>Value</th></tr></thead><tbody><tr><td>R0</td><td>0</td></tr><tr><td>R1</td><td>78</td></tr><tr><td>R2</td><td>1</td></tr><tr><td>R3</td><td>0</td></tr><tr><td>R4</td><td>0</td></tr><tr><td>R5</td><td>0</td></tr><tr><td>R6</td><td>0</td></tr><tr><td>R7</td><td>0</td></tr><tr><td>R8</td><td>0</td></tr><tr><td>-</td><td>-</td></tr></tbody></table>					Register	Value	R0	0	R1	78	R2	1	R3	0	R4	0	R5	0	R6	0	R7	0	R8	0	-	-
Register	Value																									
R0	0																									
R1	78																									
R2	1																									
R3	0																									
R4	0																									
R5	0																									
R6	0																									
R7	0																									
R8	0																									
-	-																									
<table border="1"><thead><tr><th>Address</th><th>Value</th></tr></thead><tbody><tr><td>0x000010000:</td><td>05 20 A0 E3 02 10 A0 E1 01 20 42 E2 91 02 01 E0 . B</td></tr><tr><td>0x000010010:</td><td>01 52 E3 0A FA FF FF EA .. R</td></tr><tr><td>0x000010020:</td><td>00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00</td></tr></tbody></table>					Address	Value	0x000010000:	05 20 A0 E3 02 10 A0 E1 01 20 42 E2 91 02 01 E0 . B	0x000010010:	01 52 E3 0A FA FF FF EA .. R	0x000010020:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00														
Address	Value																									
0x000010000:	05 20 A0 E3 02 10 A0 E1 01 20 42 E2 91 02 01 E0 . B																									
0x000010010:	01 52 E3 0A FA FF FF EA .. R																									
0x000010020:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00																									

## Assignment 4.2: Programming languages

Take screenshots that the following commands work:

```
javac --version
```

`java --version`

gcc --version

```
python3 --version
```

bash --version

reza-hekmatirad@re

```
reza-hekmatirad@reza-hekmatirad-VMware-Virtual-Platform:~$ python3 --version
| App Center 2.3
reza-hekmatirad@reza-hekmatirad-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.
reza-hekmatirad@reza-hekmatirad-VMware-Virtual-Platform:~$ gcc --version
gcc (Ubuntu 13.3.0-0ubuntu2-24.04) 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.

reza-hekmatirad@reza-hekmatirad-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)
reza-hekmatirad@reza-hekmatirad-VMware-Virtual-Platform:~$
```

### **Assignment 4.3: Compile**

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

-Fibonacci.java and 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 and fib.py

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?

-First compile it via javac (file name.java) then run it via java(file name)

How do I run a Python program?

-It does not need to be compiled since Python is interpreted so just do the python3 (file name.py)

How do I run a C program?

-First compile it via gcc -o (file name)(file name.c) then run it via ./(file name)

How do I run a Bash script?

-It does not need to be compiled since it is interpreted so first make it executable(chmod + x (file name)) and then just do the bash(file name.sh)

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

For .c files → Yes → fib.c turns into fib

For .java → Yes → Fibonacci.java turns into Fibonacci.class

For .py → No

For .sh or .bash → No

Take relevant screenshots of the following commands:

- Compile the source files where necessary:

The screenshot shows a Linux desktop interface. On the left is a file manager window titled 'Files' showing files in the directory '/code'. Files include 'fib', 'fib.c', 'fib.py', 'fib.sh', 'Fibonacci.class', 'Fibonacci.java', and 'runall.sh'. On the right is a terminal window titled 'reza-hekmatirad@hekmatirad: ~/code' with the following command history:

```
reza-hekmatirad@hekmatirad:~/code$ gcc -o fib fib.c
reza-hekmatirad@hekmatirad:~/code$ ls
fib fib.c Fibonacci.java fib.py fib.sh runall.sh
reza-hekmatirad@hekmatirad:~/code$ javac Fibonacci.java
reza-hekmatirad@hekmatirad:~/code$ ls
fib fib.c Fibonacci.class Fibonacci.java fib.py fib.sh runall.sh
reza-hekmatirad@hekmatirad:~/code$ S
```

- Make them executable

The screenshot shows a terminal window with the following command history:

```
reza-hekmatirad@hekmatirad:~/code$ ls
fib fib.c Fibonacci.class Fibonacci.java fib.py fib.sh runall.sh
reza-hekmatirad@hekmatirad:~/code$ chmod +x fib.sh
reza-hekmatirad@hekmatirad:~/code$ 
reza-hekmatirad@hekmatirad:~/code$ 
reza-hekmatirad@hekmatirad:~/code$ chmod +x runall.sh
reza-hekmatirad@hekmatirad:~/code$ ls -l
total 40
-rwxrwxr-x 1 reza-hekmatirad reza-hekmatirad 16136 Dec  5 13:56 fib
-rw-rw-r-- 1 reza-hekmatirad reza-hekmatirad   831 Jun  9 2023 fib.c
-rw-rw-r-- 1 reza-hekmatirad reza-hekmatirad  1448 Dec  5 13:57 Fibonacci.class
-rw-rw-r-- 1 reza-hekmatirad reza-hekmatirad   839 Jun  9 2023 Fibonacci.java
-rw-rw-r-- 1 reza-hekmatirad reza-hekmatirad   516 Jun  9 2023 fib.py
-rwxrwxr-x 1 reza-hekmatirad reza-hekmatirad   668 Jun  9 2023 fib.sh
-rwxrwxr-x 1 reza-hekmatirad reza-hekmatirad   249 Jun  9 2023 runall.sh
reza-hekmatirad@hekmatirad:~/code$
```

- Run them

The screenshot shows a terminal window with the following command history:

```
reza-hekmatirad@hekmatirad:~/code$ ./fib
Fibonacci(18) = 2584
Execution time: 0.02 milliseconds
reza-hekmatirad@hekmatirad:~/code$ java Fibonacci
Fibonacci(18) = 2584
Execution time: 0.36 milliseconds
reza-hekmatirad@hekmatirad:~/code$ ./fib.py
bash: ./fib.py: Permission denied
reza-hekmatirad@hekmatirad:~/code$ ./fib.sh
Fibonacci(18) = 2584
Execution time 11810 milliseconds
reza-hekmatirad@hekmatirad:~/code$
```

```
reza-hekmatirad@hekmatirad:~/code
Running C program:
Fibonacci(19) = 4181
Execution time: 0.03 milliseconds

Running Java program:
Fibonacci(19) = 4181
Execution time: 0.52 milliseconds

Running Python program:
Fibonacci(19) = 4181
Execution time: 1.02 milliseconds

Running BASH Script
Fibonacci(19) = 4181
Execution time: 18798 milliseconds

reza-hekmatirad@hekmatirad:~/code$
```

- Which (compiled) source code file performs the calculation the fastest?
  - Obviously .c source code

#### 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 -O3 -o fib fib.c`
- Compile **fib.c** again with the optimization parameters

```
reza-hekmatirad@hekmatirad:~/code
reza-hekmatirad@hekmatirad:~/code$ gcc -O3 -o fib fib.c
reza-hekmatirad@hekmatirad:~/code$ S
```

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

reza-hekmatirad@hekmatirad:~/code\$ gcc -O3 -o fib fib.c  
reza-hekmatirad@hekmatirad:~/code\$ ./fib  
Fibonacci(18) = 2584  
Execution time: 0.01 milliseconds  
reza-hekmatirad@hekmatirad:~/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.

```
reza-hekmatirad@hekmatirad:~/code$ ./runall.sh  
Running C program:  
Fibonacci(19) = 4181  
Execution time: 0.01 milliseconds  
  
Running Java program:  
Fibonacci(19) = 4181  
Execution time: 0.52 milliseconds  
  
Running Python program:  
Fibonacci(19) = 4181  
Execution time: 0.98 milliseconds  
  
Running BASH Script  
Fibonacci(19) = 4181  
Execution time: 20066 milliseconds  
  
reza-hekmatirad@hekmatirad:~/code$
```

#### 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 a debugger interface with the following components:

- Toolbar:** Open, Run (highlighted), 250, Step, Reset.
- Assembly Code:**

```
1 Main:
2     mov r1, #2
3     mov r2, #4
4     mov r0, r1
5
6 Loop:
7     mul r0, r0, r1
8     sub r2, r2, #1
9     cmp r2, #1
10    beq End
11    b Loop
12
13 End:
```
- Registers:** A table showing register values:

Register	Value
R0	10
R1	2
R2	1
R3	0
R4	0
R5	0
R6	0
R7	0
R8	0
-	-
- Memory Dump:** Hex dump of memory at addresses 0x000010000 and 0x000010010.

	0x000010000:	0x000010010:
02 10 A0 E3 04 20 A0 E3 01 00 A0 E1 90 01 00 E0	01 20 42 E2 01 00 52 E3 00 00 0A FA FF FF EA B R	

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