

# Lab Sheet 1 for CS F342 Computer Architecture

Semester 1 – 2017-18

Lab One will have two goals:

**Goal 1:** Using videos to understand the physical setup of computers.

**Goal 2:** Installing and launching SPIM – this tools will be used for bulk of the lab work in subsequent weeks.

## Understanding the insides of a Computer:

Watch the following four videos on YouTube

- Lenovo OEM PC Upgrades (PSU and GPU) - <https://www.youtube.com/watch?v=SACe4gLpprk>
- How to: Intel iMac Hard drive replacement - [https://www.youtube.com/watch?v=3w6E2\\_XqaBw](https://www.youtube.com/watch?v=3w6E2_XqaBw)
- Beginners Guide to Motherboards - <https://www.youtube.com/watch?v=ZnaQyGAg8Eg>
- Raspberry Pi Zero - <https://www.youtube.com/watch?v=WJdQ4rknBX0>

Videos also downloaded at <http://172.16.2.193/BigFiles/CompArch>

Some questions to discuss and ponder (non-evaluative):

- Tabulate the new terms and acronyms that you came across the three videos.
- Give some pros and cons for modularity.
- Why do we need slots for memory modules(DIMM slots)?
- Whys are different motherboards needed for Intel vs. AMD?
- Can multiple motherboard designs exist for an Intel i7 (say 6<sup>th</sup> generation)? What would be some of the reasons for such variety?
- Rank the following in terms of modularity – Mobile phones or Tablet devices, Desktop PC; All-in-Ones (e.g. iMac) ; Laptop.

## Installing and launching SPIM

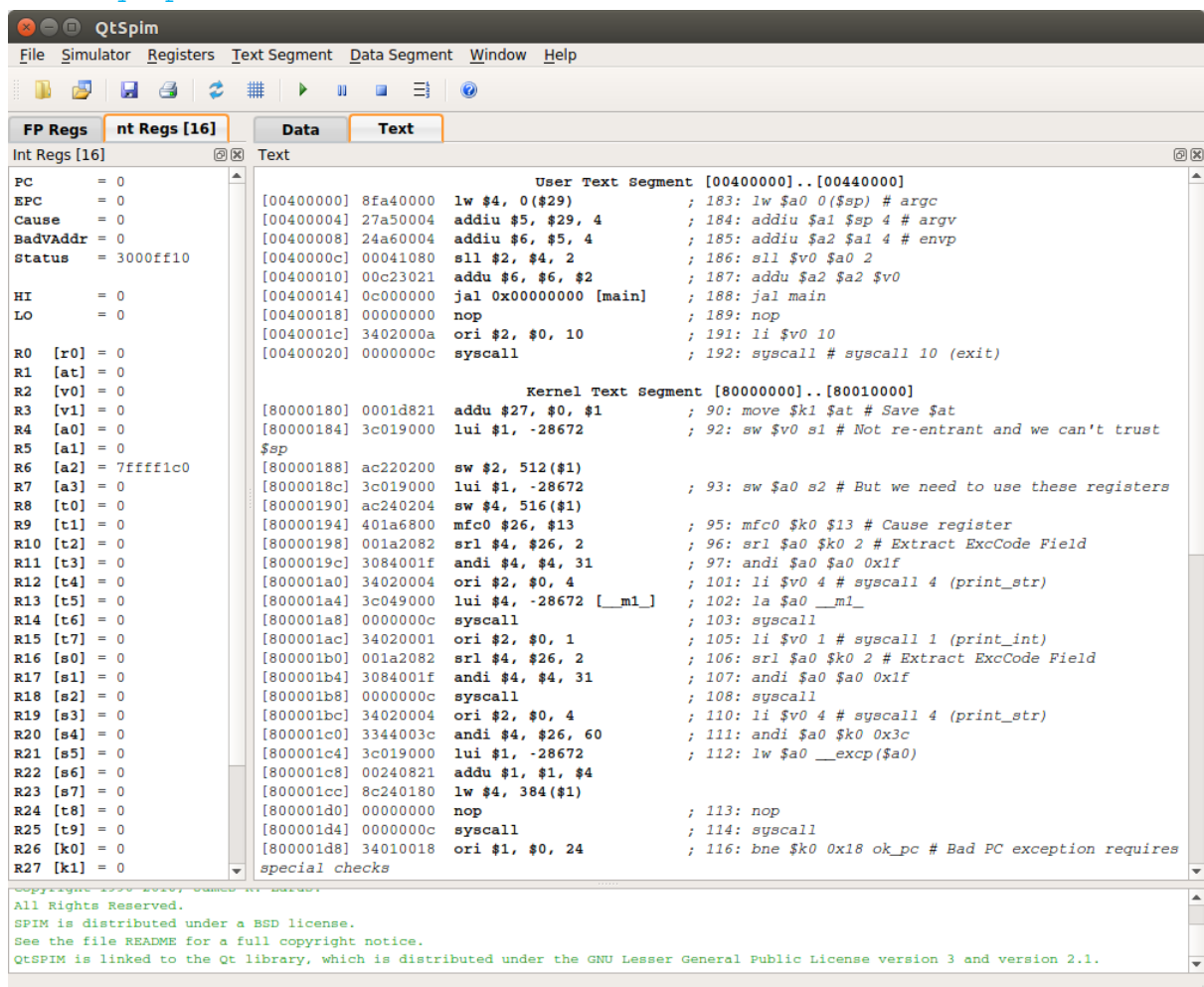
Open a terminal to run the following commands.

- Check the OS + processor support on your Linux machine – verify whether 32 bit or 64 bit. [Try commands like `uname -p` ; `uname -a` ; `man uname`]
- Download appropriate `qtspim` binary from internet or <http://172.16.2.193/BigFiles/CompArch>

- C. Install using `dpkg` tool. Do not forget to get “super user privilege” using `sudo`.

```
abhishek@hpslkbk: ~/Downloads
abhishek@hpslkbk:~/Downloads$ sudo dpkg -i qtspim_9.1.17_linux64.deb
[sudo] password for abhishek:
Selecting previously unselected package qtspim.
(Reading database ... 496661 files and directories currently installed.)
Preparing to unpack qtspim_9.1.17_linux64.deb ...
Unpacking qtspim (9.1.17) ...
Setting up qtspim (9.1.17) ...
Processing triggers for gnome-menus (3.10.1-0ubuntu2) ...
Processing triggers for desktop-file-utils (0.22-1ubuntu1) ...
Processing triggers for bamfdaemon (0.5.1+14.04.20140409-0ubuntu1) ...
Rebuilding /usr/share/applications/bamf-2.index...
Processing triggers for mime-support (3.54ubuntu1.1) ...
Processing triggers for man-db (2.6.7.1-1ubuntu1) ...
abhishek@hpslkbk:~/Downloads$
```

- D. Launch `qtspim`



The screenshot shows the QtSpim application window. The 'Text' tab is selected, displaying assembly code. The code is divided into two sections: 'User Text Segment' and 'Kernel Text Segment'. The 'User Text Segment' code starts at address 00400000 and ends at 00440000. It includes instructions like `lw $4, 0($29)`, `addiu $5, $29, 4`, `addiu $6, $5, 4`, `sll $2, $4, 2`, `addu $6, $6, $2`, `jal 0x00000000 [main]`, `nop`, `ori $2, $0, 10`, and `syscall`. The 'Kernel Text Segment' code starts at address 80000180 and ends at 80010000. It includes instructions like `addu $27, $0, $1`, `lui $1, -28672`, `sw $2, 512($1)`, `lui $1, -28672`, `sw $4, 516($1)`, `mfc0 $26, $13`, `srl $4, $26, 2`, `andi $4, $4, 31`, `ori $2, $0, 4`, `lui $4, -28672 [__m1]`, `syscall`, `ori $2, $0, 1`, `srl $4, $26, 2`, `andi $4, $4, 31`, `syscall`, `ori $2, $0, 4`, `andi $4, $26, 60`, `lui $1, -28672`, `addu $1, $1, $4`, `lw $4, 384($1)`, `nop`, `syscall`, `ori $1, $0, 24`, and `special checks`. The interface also shows the 'Registers' tab with various registers like PC, EPC, Cause, BadVAddr, Status, HI, LO, R0, R1, R2, R3, R4, R5, R6, R7, R8, R9, R10, R11, R12, R13, R14, R15, R16, R17, R18, R19, R20, R21, R22, R23, R24, R25, R26, R27, R28, R29, R30, R31, and their values.

Reference for QtSpim: Chapter 2 and Appendix A of Second Edition “Patterson and Hennessy”

Next Week: Getting started with Assembly on QtSpim - System Calls and User Input + Add/Sub + Disassembly -  
- reversing given byte code (binary) to assembly