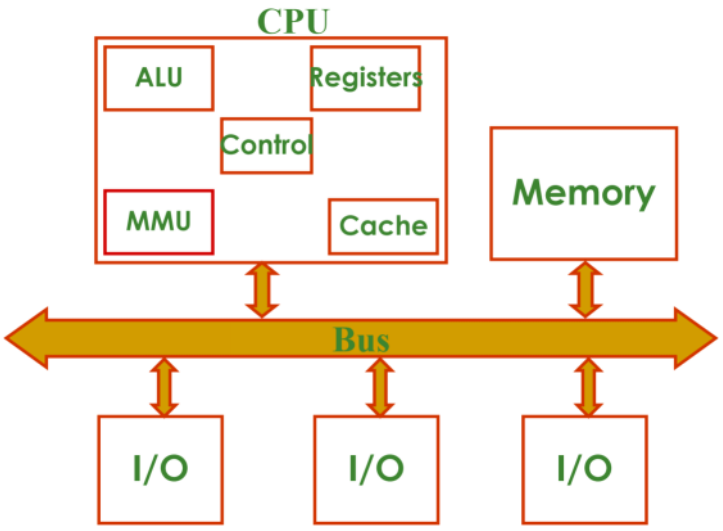
Software Architecture

- We are now moving towards a new set of topics
- Until now, we were talking about the hardware
  - ↳ we want to look at the software necessary to create programs
- Recall Hardware Architecture



- We can now move into software
  - ↳ There are several components of software that are always there

- There can always be more than one program running concurrently
- This means that the hardware resources are actually shared by programs in execution
- There must be some software helping to do this.
- One of the very special software that helps in managing of the hardware resources is the Operating System.
  - ↳ so we need to know about the operating system to understand what happens when programs run on a computer system.

Examples of Operating Systems

- Unix → AIX, HP-UX, Solaris

- Linux → Debian, Fedora, RedHat
- Mac OS →
- Windows → Window 10, 8, 7, Vista

- It is not our objective to talk about any specific OS
  - ↳ we will use a Unix like OS

Processes

- A fundamental concept to understand program execution
  - ↳ process is not a piece of software
    - ↳ it's an abstract entity
      - ↳ for now, we will think of it as a program in execution
        - ↳ ∴ a process must be present in memory and being executed

- We can get some practical understanding of the fact that there can be more than one program in execution by using a command called ps

% ps ⇒ Shell: a command interpreter, through which you interact with the computer system.

shell prompt

↳ there are many shells available
csh, bash

↳ Shell is just a program

- If you run ps...

"terminal"

```
% ps
  PID    TTY      TIME     CMD
15459  pts/10   00:00:00  bash
15491  pts/10   00:00:00  ps
```

- To get all the processes

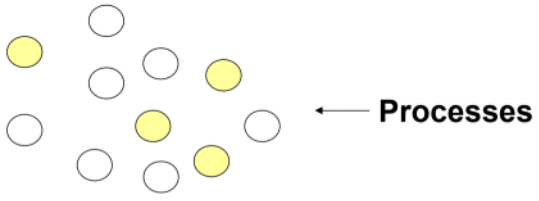
```
% ps  -a
  PID   TTY      TIME     CMD
 6358  pts/4    00:00:00  pine
15538 pts/10   00:00:00  ps
20252 pts/2    00:00:01  pine
31066 pts/5    00:00:01  emacs-x
31072 pts/5    00:00:00  xterm
31084 pts/5    00:00:00  xdvi-xaw3d.bin
```

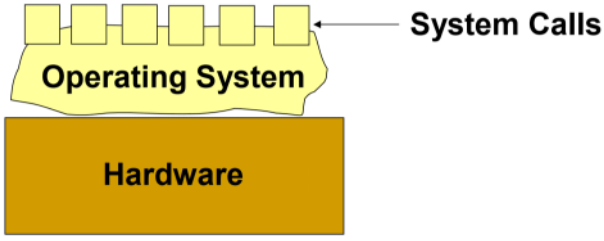
- There is another option

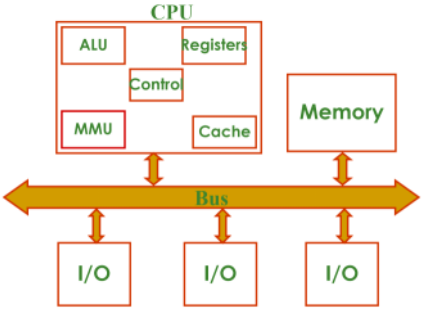
```
% ps  -l
F S  UID  PID  PPID  C PRI  NI ADDR  SZ WCHAN  TTY TIME  CMD
0 S  539 15459 15458  0  76   0 - 16517 wait   pts/10 00:00:00 bash
0 R  539 15539 15459  0  78   0 - 15876 -      pts/10 00:00:00 ps
```

Operating System, Processes, Hardware

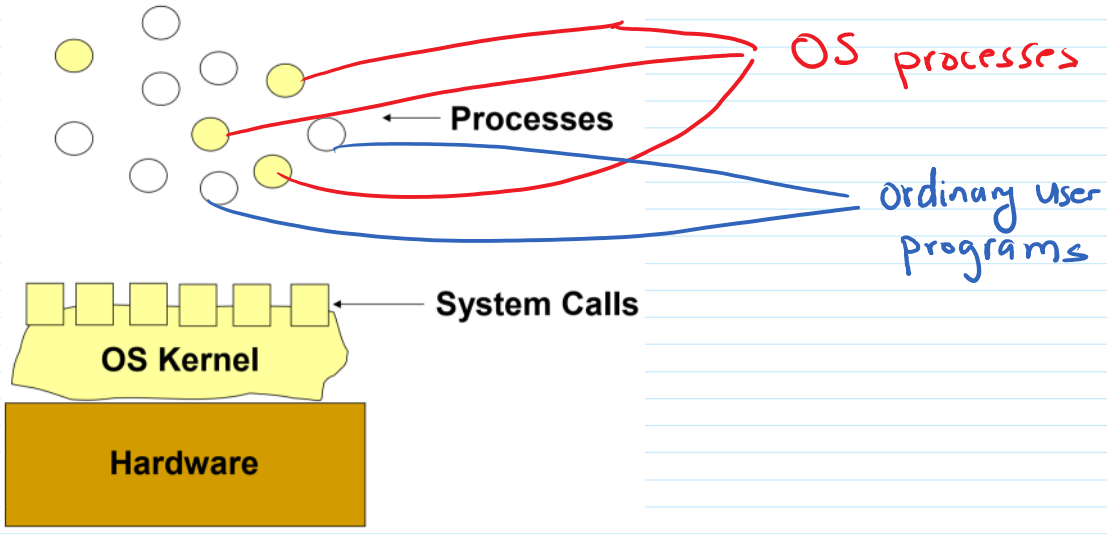
- Our objective is to understand the relationship between the operating system, the processes and the hardware.







- More accurate representation



- OS Kernel is the core of the operating system
  - ↳ some of the functionality of the operating system may be implemented elsewhere.
  - ↳ this is represented by the yellow processes above.

System Calls

- It is through the system calls that a process can get some functionality out of the operating system
  - ↳ Interface or API for interaction with the operating system
    - ↳ API is a specification on how a piece of software can can interact with another piece of software.

~ 200 - 300 system call functions in a typical OS.

Examples of System Calls

⇒ Operation on files

- create() ⇒ to create a new file
- unlink() ⇒ to remove a file
- open() → to open file for reading and/or writing

- read() ⇒ to read data from an open file into a variable
- write() ⇒ to write data into an open file
- lseek() ⇒ to change the current pointer into a file.

⇒ Operation on processes
↳ a process may need to operate on itself or other processes

- fork() ⇒ to create a new process
  ↳ lingo: parent process calls fork() which causes a child process to be created.
  ↳ Both parent and child processes continue to execute from that point in the program.
    ↳ the call to fork()
- exec() ⇒ to change the memory image of a process
  ↳ ie to change the program that a process is executing (odd concept!)
- exit() ⇒ gracefully terminate.
- wait() ⇒ to make parent sleep until child terminates.

OS Privileges

- A process must be allowed to do sensitive operations while it is executing a system call
- Special instructions will have been included in the instruction set Architecture for such purposes
  ↳ they are called privileged instructions

↳ meant for use only by special programs like the OS