High Performance Computing
Lecture 11

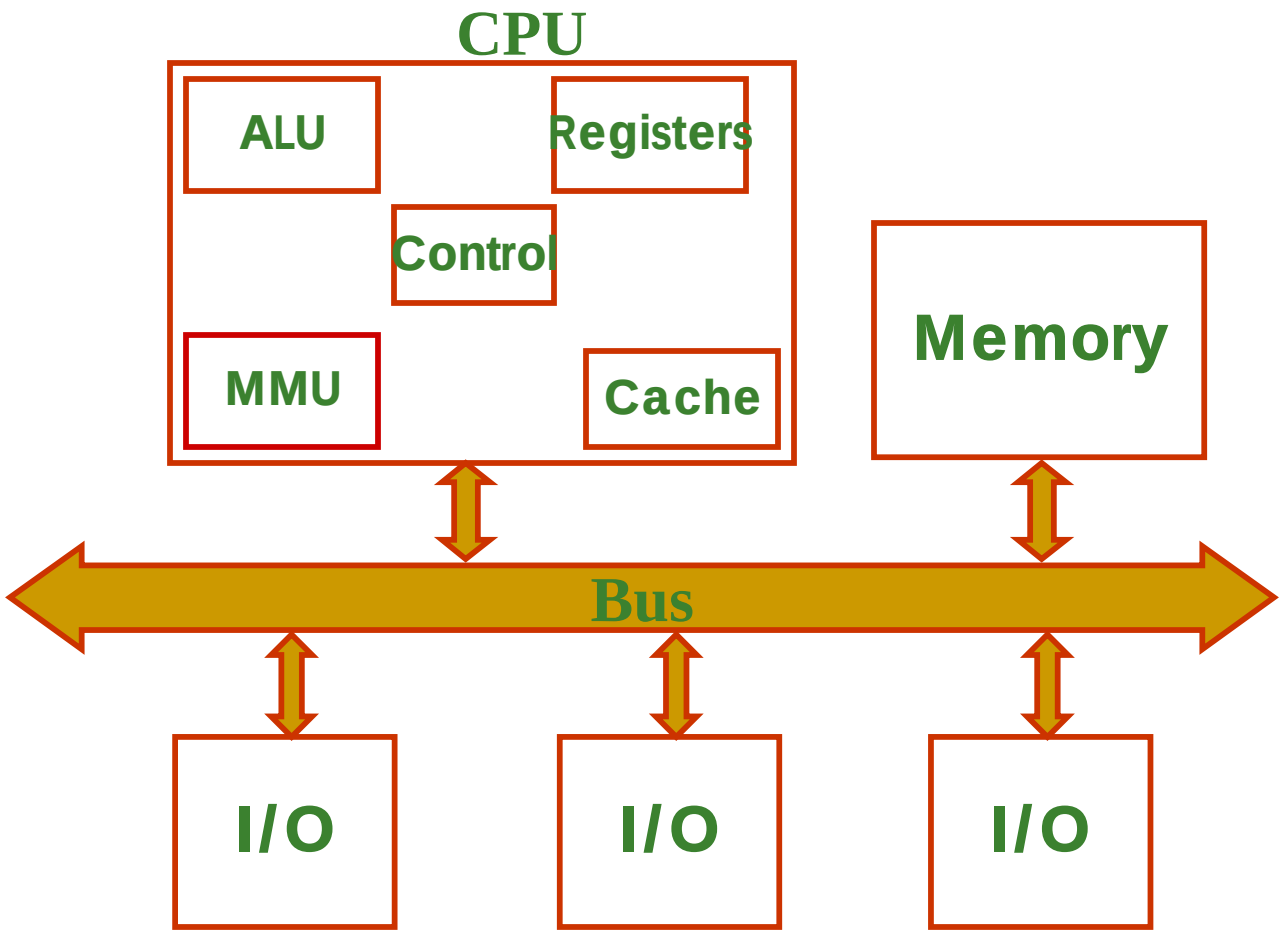
Matthew Jacob

Indian Institute of Science

Agenda

1. Program execution: Compilation, Object files, Function call and return, Address space, Data & its representation (4)
2. Computer organization: Memory, Registers, Instruction set architecture, Instruction processing (6)
3. Virtual memory: Address translation, Paging (4)
4. Operating system: Processes, System calls, Process management (6)
5. Pipelined processors: Structural, data and control hazards, impact on programming (4)
6. Cache memory: Organization, impact on programming (5)
7. Program profiling (2)
8. File systems: Disk management, Name management, Protection (4)
9. Parallel programming: Inter-process communication, Synchronization, Mutual exclusion, Parallel architecture, Programming with message passing using MPI (5)

Computer Organization: Hardware



Computer Organization: Software

- Hardware resources of computer system are shared by programs in execution
- Operating System: Special software that manages this sharing

Operating Systems (OS)

Examples

- Unix AIX, HP-UX, Solaris
- Linux Fedora, openSUSE, Ubuntu, Debian
- Apple Mac OS Mac OS X Snow Leopard
- Microsoft Windows Windows 7, Vista, XP
- Google Chrome OS

Computer Organization: Software

- Hardware resources of computer system are shared by programs in execution
- Operating System: Special software that manages this sharing
- Process: A program in execution
  - i.e., present in main memory and being executed
  - On Unix systems, you can use ps to get information about the current status of processes

```
%   ps
```

Shell prompt

Computer Organization: Software

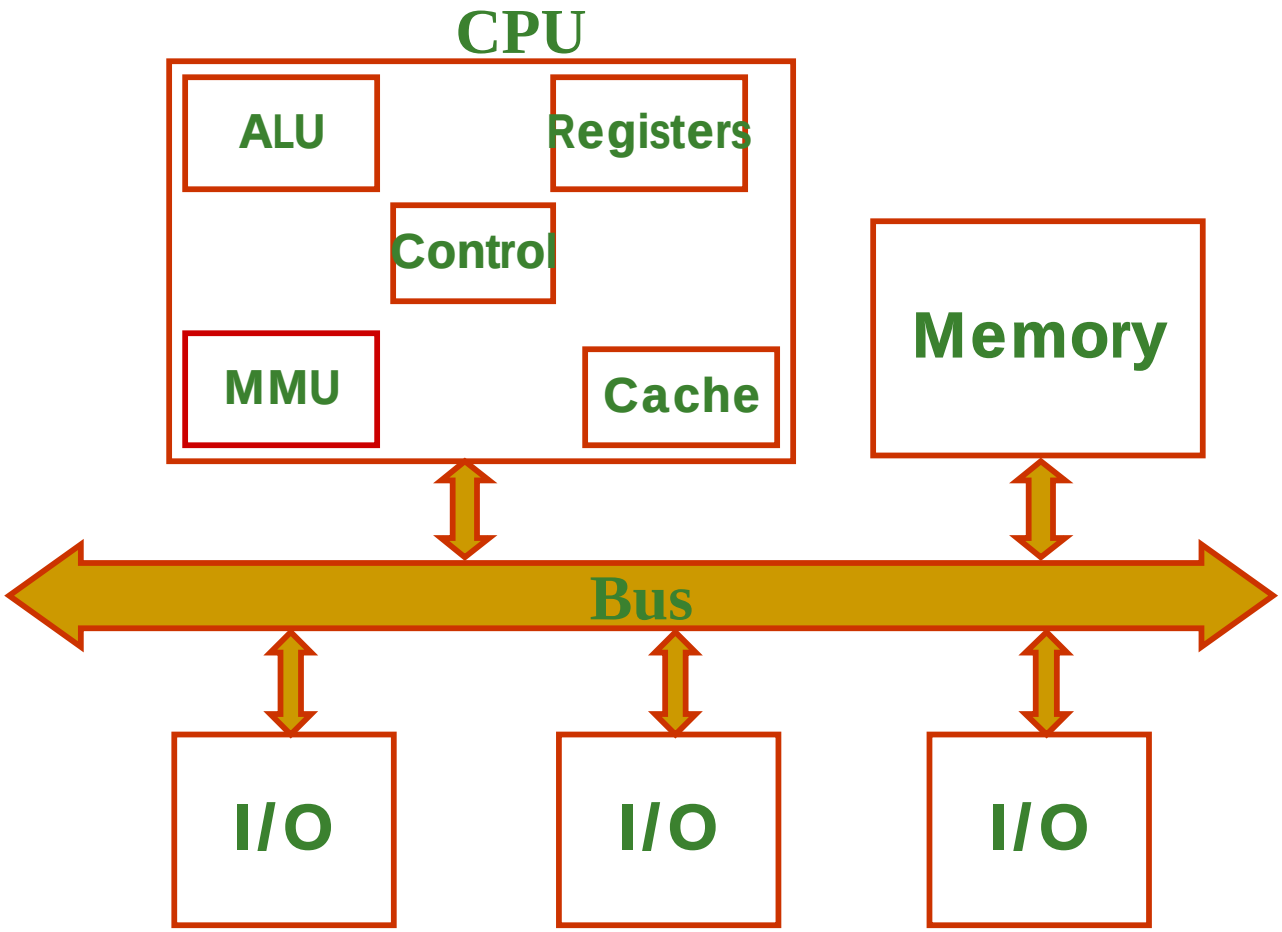
- Hardware resources of computer system are shared by programs in execution
- Operating System: Special software that manages this sharing
- Process: A program in execution
- Shell: A command interpreter, through which you interact with the computer system
  - Examples of Unix shells: csh, bash
  - A program; just another program

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

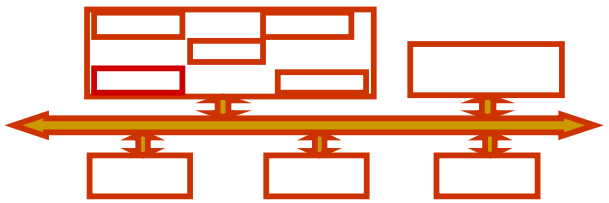
```
% ps  -a
  PID    TTY      TIME     CMD
 6358  pts/4    00:00:00  pine
15538 pts/10   00:00:00  ps
20252 pts/2    00:00:01  pine
31066 pts/5    00:00:01  emacs-x
31072 pts/5    00:00:00  xterm
31084 pts/5    00:00:00  xdvi-xaw3d.bin

% 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
```

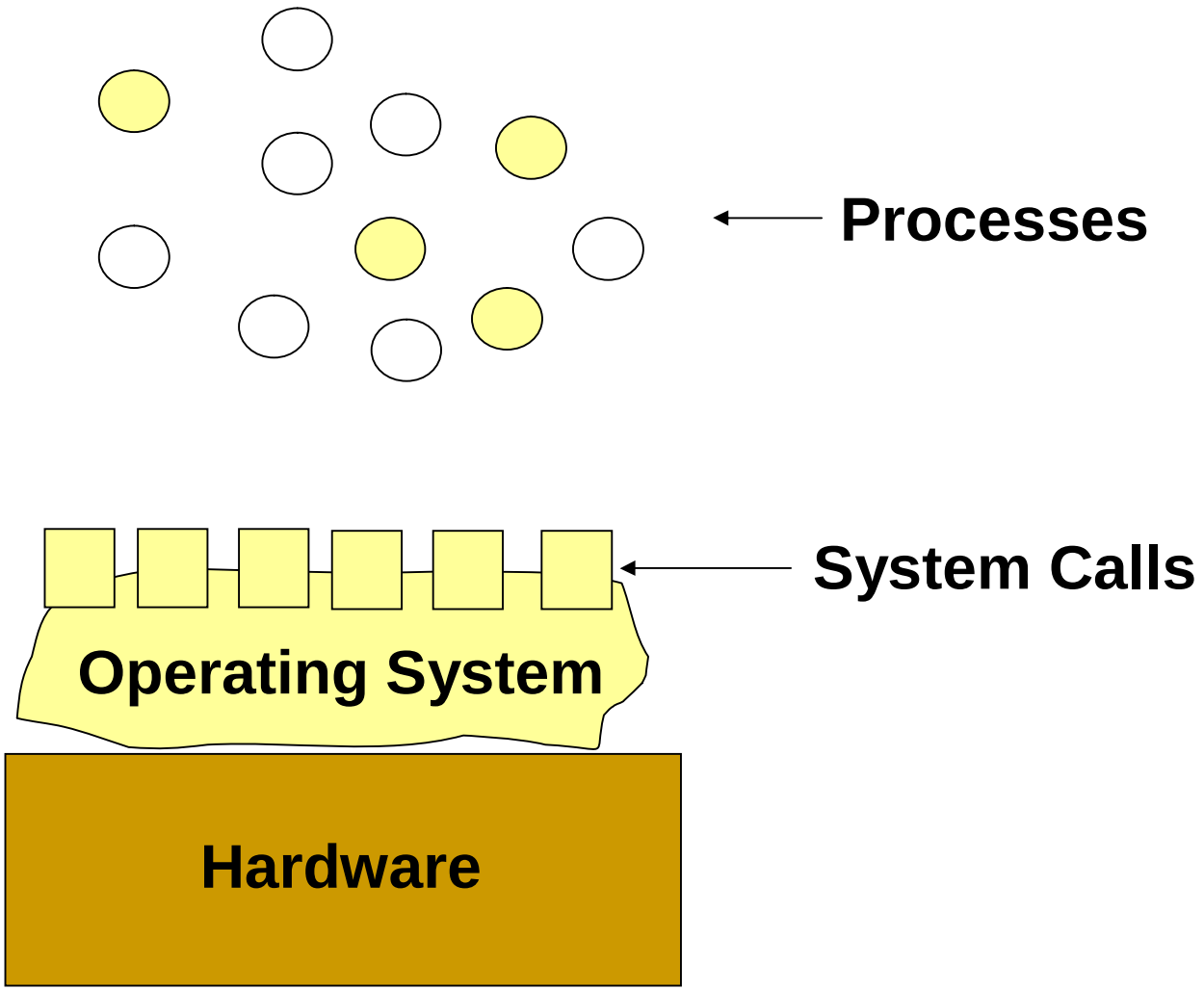
Operating System, Processes, Hardware



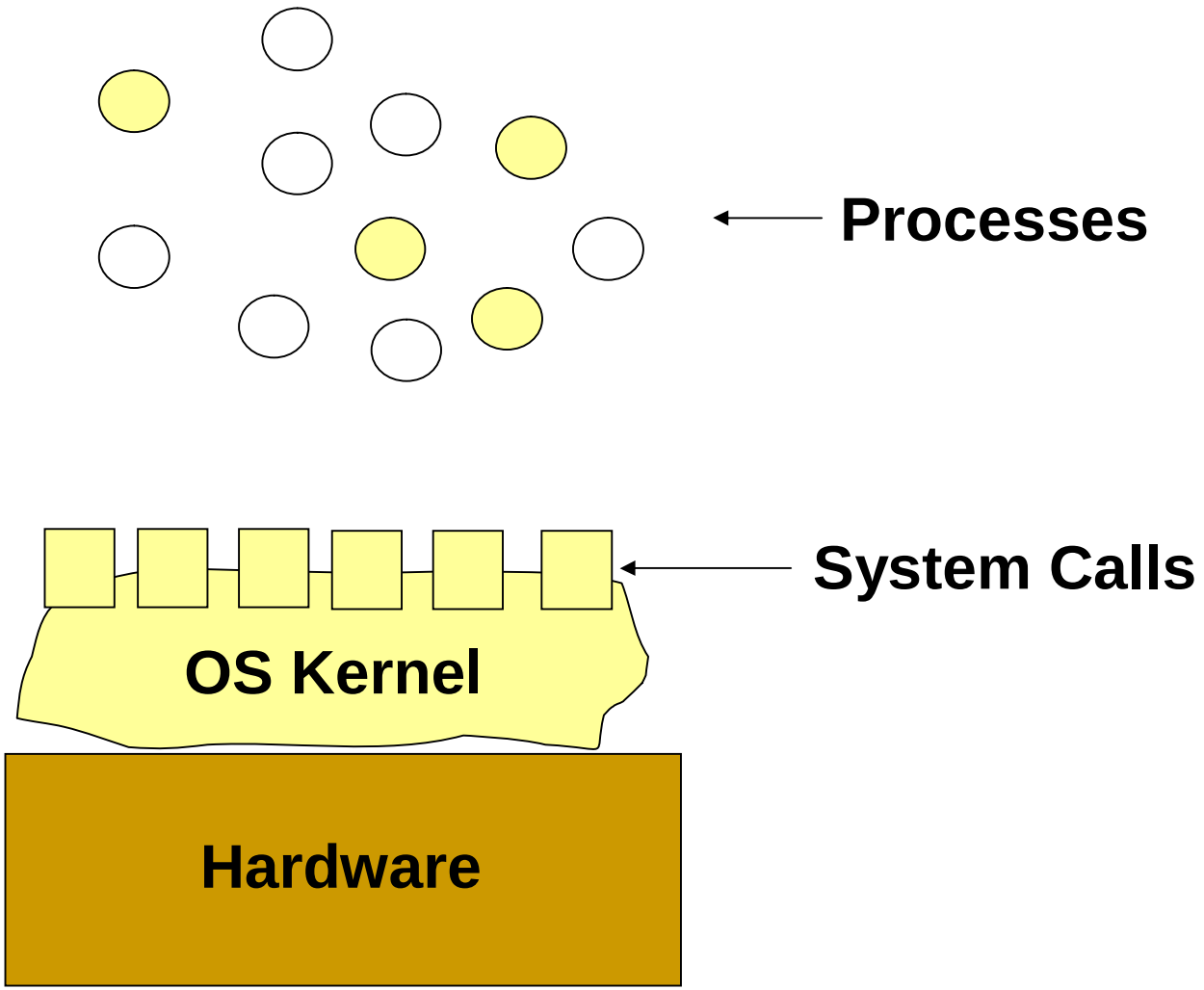
Operating System, Processes, Hardware



Operating System, Processes, Hardware



Operating System, Processes, Hardware



System Calls

- How a process gets the operating system to do something for it
  - Interface or API (Application Programming Interface) for interaction with the operating system

System Calls

- How a process gets the operating system to do something for it
  - Interface or API (Application Programming Interface) for interaction with the operating system
- Examples: Operations on files
  - creat(): to create a new file
  - unlink(): to remove a file
  - open(): to open a 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 the open file

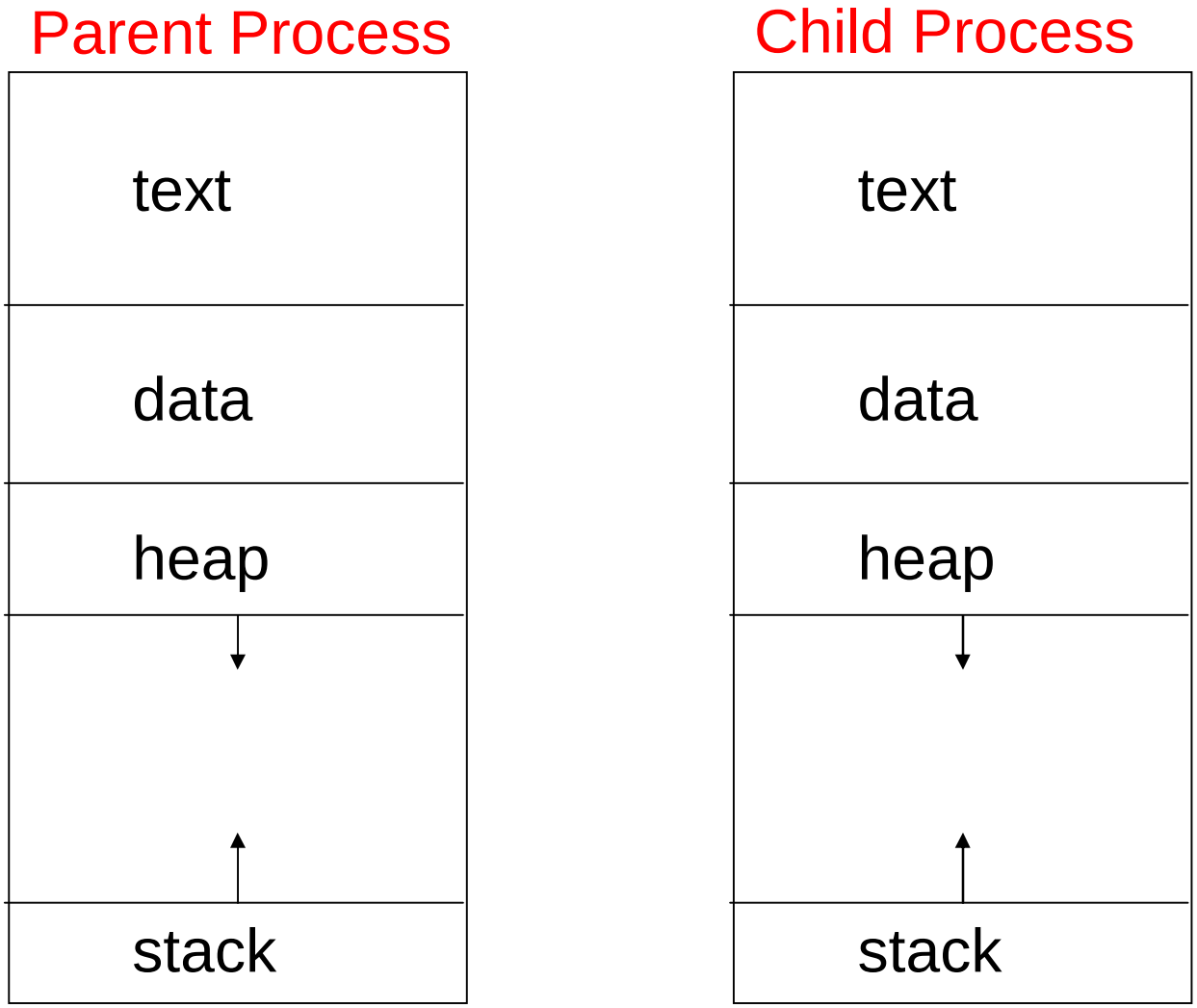
System Calls

- How a process gets the operating system to do something for it
  - Interface or API (Application Programming Interface) for interaction with the operating system
- Examples: Operations on processes
  - fork(): to create a new process
    - Terminology: 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

System Calls

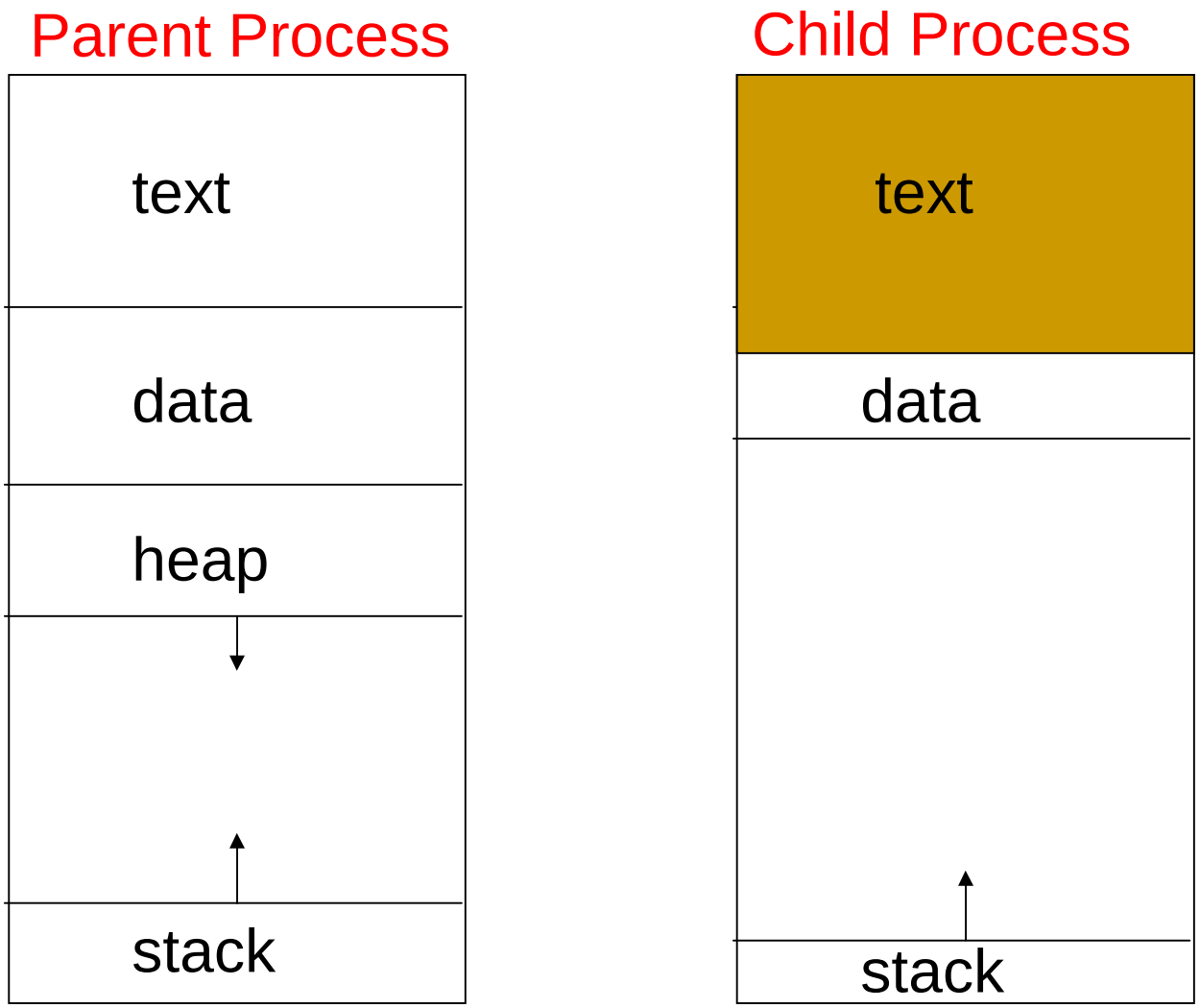
- How a process gets the operating system to do something for it
  - Interface or API (Application Programming Interface) for interaction with the operating system
- Examples: Operations on processes
  - fork(): to create a new process
  - exec(): to change the memory image of a process
    - e,g, to change the program that a process is executing

fork( ) and exec( )




```
retval = fork();
if (retval == 0) {
    /* child */
    exec(…);
} else {
    /* parent */
}
```

fork( ) and exec( )



System Calls

- How a process gets the operating system to do something for it
  - Interface or API (Application Programming Interface) for interaction with the operating system
- Examples: Operations on processes
  - fork(): to create a new process
  - exec(): to change the memory image of a process
  - exit(): to terminate
  - wait(): to make parent sleep until child terminates

System Calls

- How a process gets the operating system to do something for it
  - Interface or API (Application Programming Interface) for interaction with the operating system
- Examples: Operations on memory
  - sbrk: can be used by malloc() to increase size of the heap

System Calls

- How a process gets the operating system to do something for it
  - Interface or API (Application Programming Interface) for interaction with the operating system
- Examples: Operations on files, processes, memory, etc
- When a process is executing in a system call, it is actually executing Operating System code

Operating System, Processes, Hardware