CS:3620 Operating Systems

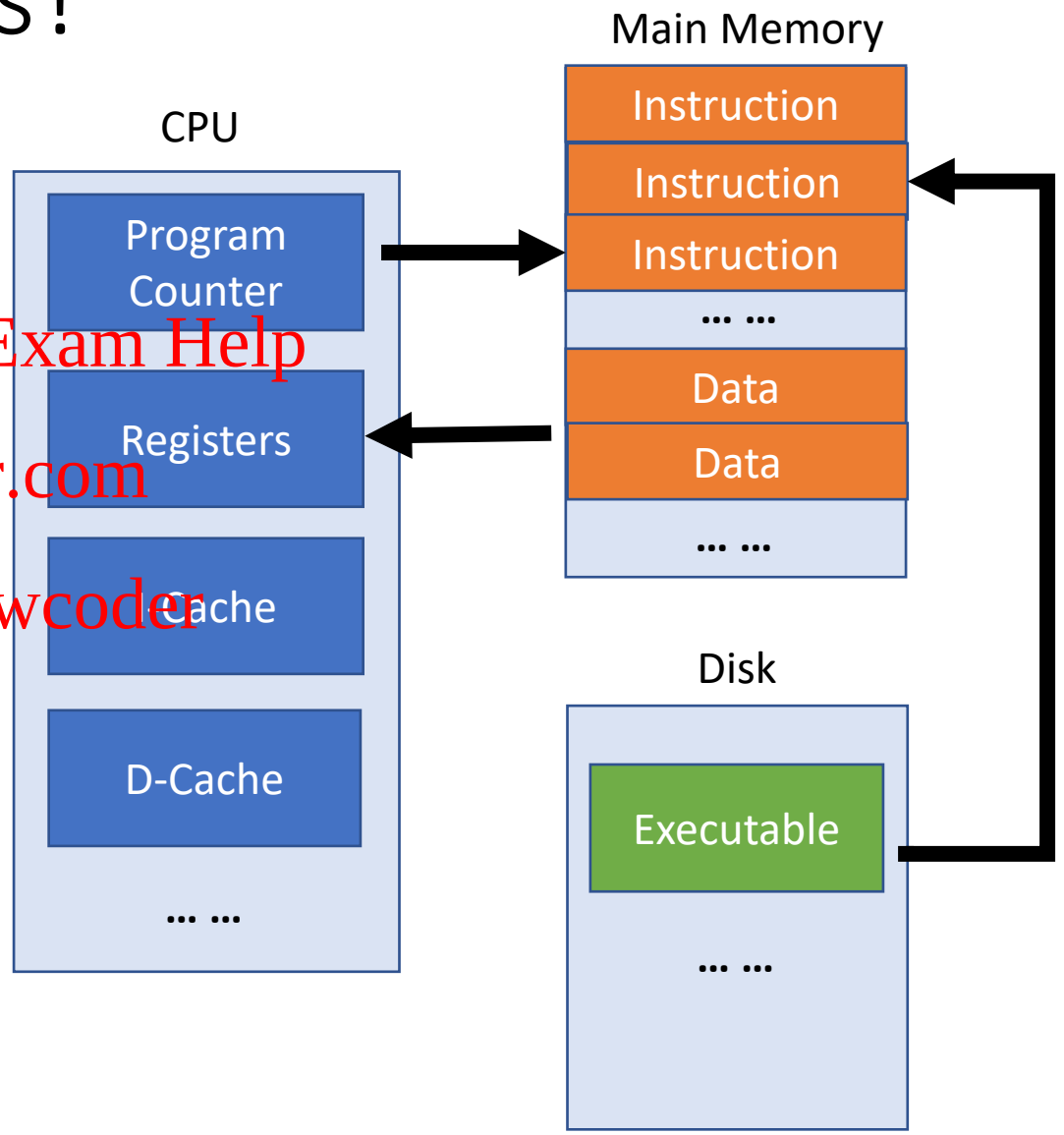
Process Abstraction

OS provides process abstraction

- When you run an exe file, the OS creates a process = a running program
- OS timeshares CPU across multiple processes: virtualizes CPU
  - No. of running processes > No. of physical CPU cores
  - Programs are responsive to users, even when the CPU is busy with other tasks
- OS has a CPU scheduler that picks one of the many active processes to execute on a CPU
  - Policy: which process to run
  - Mechanism: how to “context switch” between processes

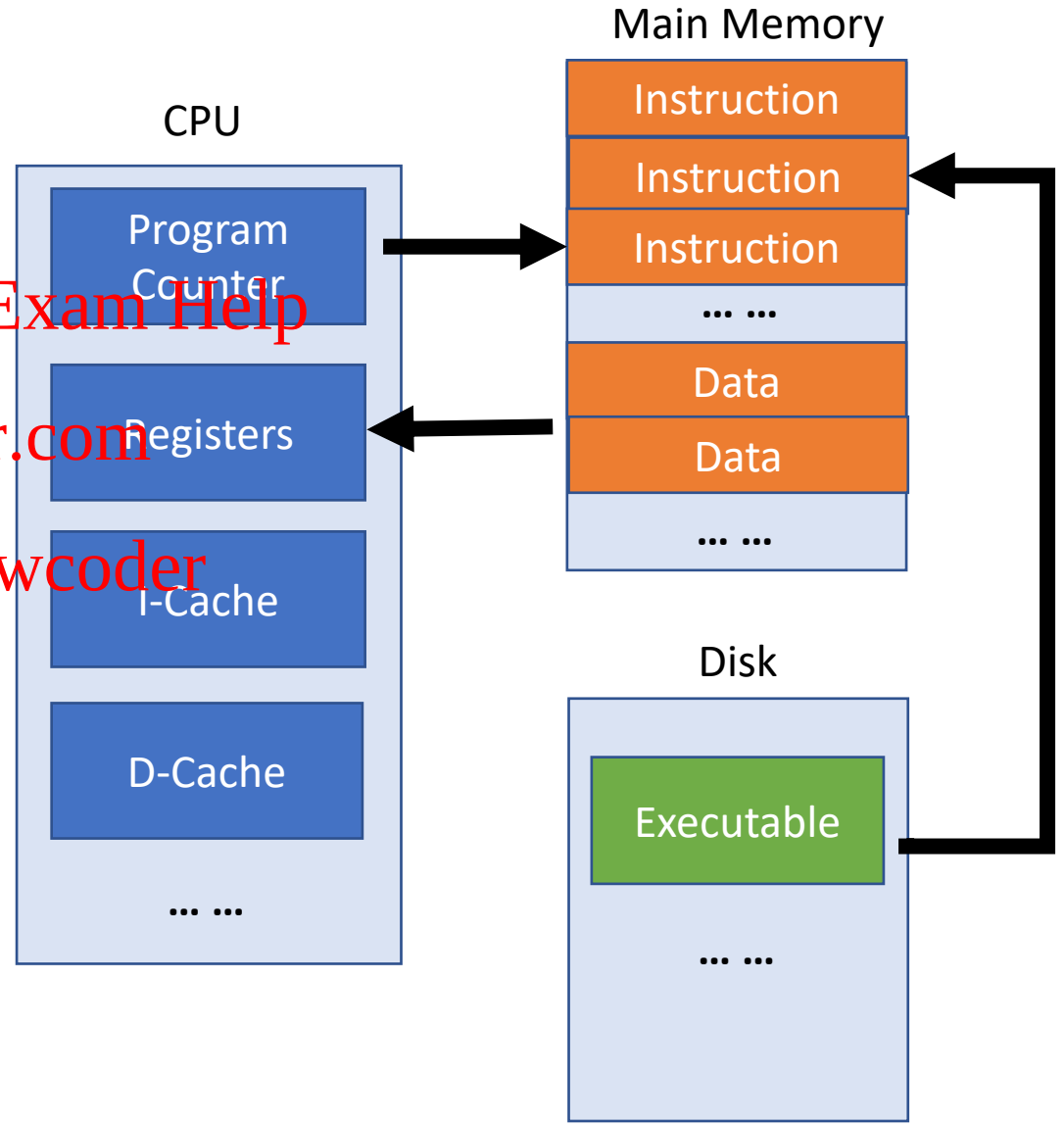
What constitutes a process?

- A unique identifier (PID)
- Memory image
  - Code & data (static)
  - Stack and heap (dynamic)
- CPU context: registers
  - Program counter
  - Current operands
  - Stack pointer
- File descriptors
  - Pointers to open files and devices
  - E.g., STDOUT, STDIN, STDERR



How does OS create a process?

- Allocates memory and creates memory image
  - Loads code, data from disk
  - Creates runtime stack, heap
- Opens basic files
  - STD IN, OUT, ERR
- Initializes CPU registers
  - PC points to first instruction



Process Information

- In Linux, you can access a lot of information about a processes in: /proc/<pid>/

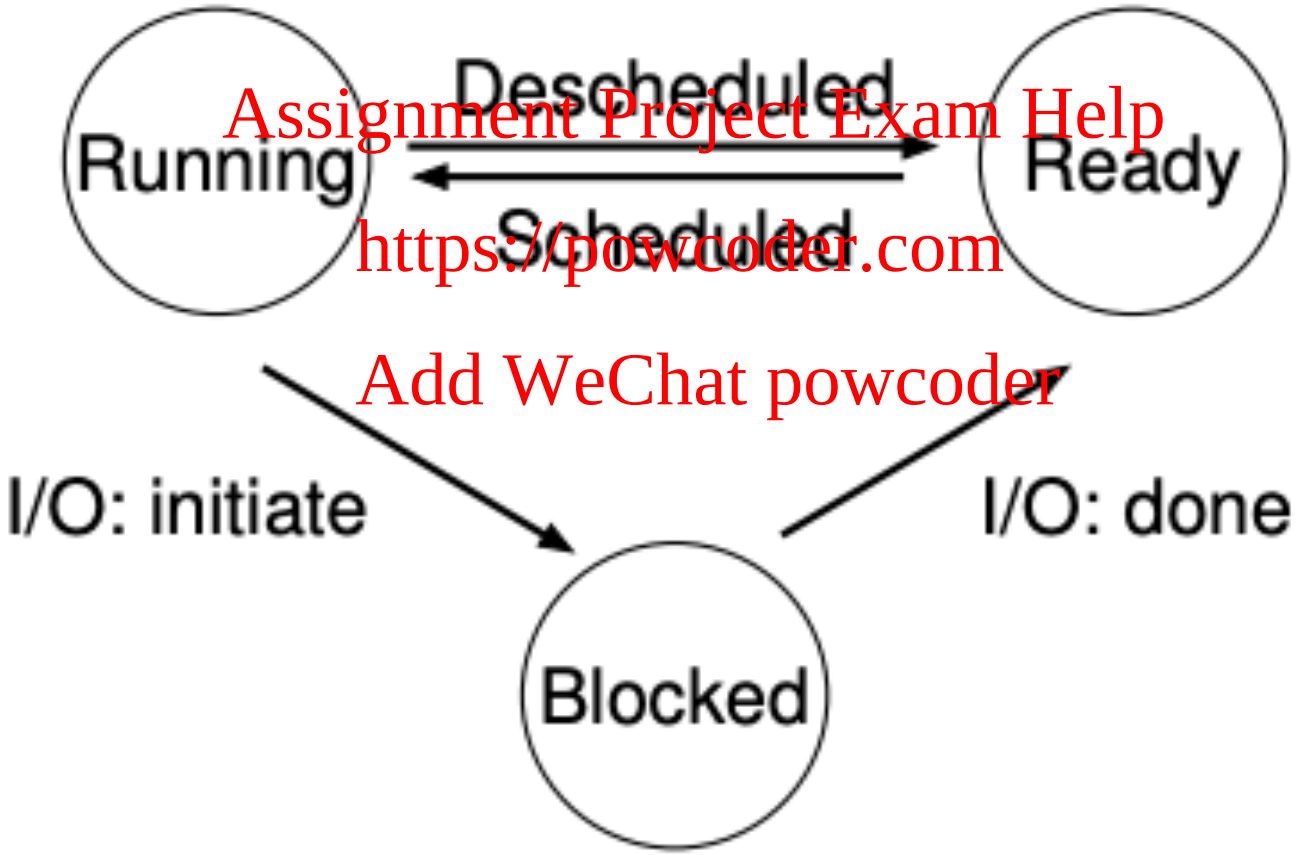
maps: memory information
cwd: link to current working directory
exe: link to executable file
fd: links to open files
cmdline: command to run the executable
environ: environment variables
…

Tools like ps and htop, parse /proc

States of a process

- Running: currently executing on CPU
- Ready: waiting to be scheduled
- Blocked: suspended, not ready to run
  - Why? Waiting for some event, e.g., process issues a read from disk
  - When is it unblocked? Disk issues an interrupt when data is ready
- New: being created, yet to run
- Dead: terminated

Process State Transitions



Example: Process State - CPU Only

Time	$Process_0$	$Process_1$	Notes
1	Running	Ready	
2	Running	Ready	
3	Running	Ready	
4	Running	Ready	$Process_0$ now done
5	–	Running	
6	–	Running	
7	–	Running	
8	–	Running	$Process_1$ now done

Example: Process State - CPU and I/O

Time	$Process_0$	$Process_1$	Notes
1	Running	Ready	
2	Running	Ready	
3	Running	Ready	$Process_0$ initiates I/O
4	Blocked	Running	$Process_0$ is blocked,
5	Blocked	Running	so $Process_1$ runs
6	Blocked	Running	
7	Ready	Running	I/O done
8	Ready	Running	$Process_1$ now done
9	Running	–	
10	Running	–	$Process_0$ now done

OS data structures

- OS maintains a data structure (e.g., list) of all active processes
- Information about each process is stored in a process control block (PCB)
  - Process identifier
  - Process state
  - Pointers to other related processes (parent)
  - CPU context of the process (saved when the process is suspended)
  - Pointers to memory locations
  - Pointers to open files

Exercise

- Exercise 1: Simulation of Process scheduler

Disclaimer

- *These lecture slides are based on a slide set by Youjip Won (Hanyang University) and Mythili Vutukuru (IIT Bombay)*