CS343: Operating System

Process Management

Lect09 : 17th Aug 2023

Dr. A. Sahu

Dept of Comp. Sc. & Engg.

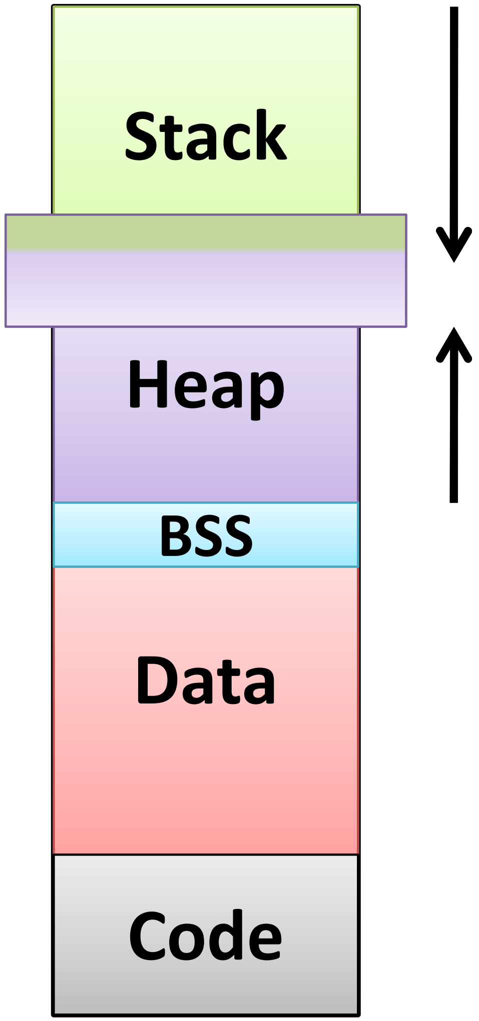
Indian Institute of Technology Guwahati

Outline

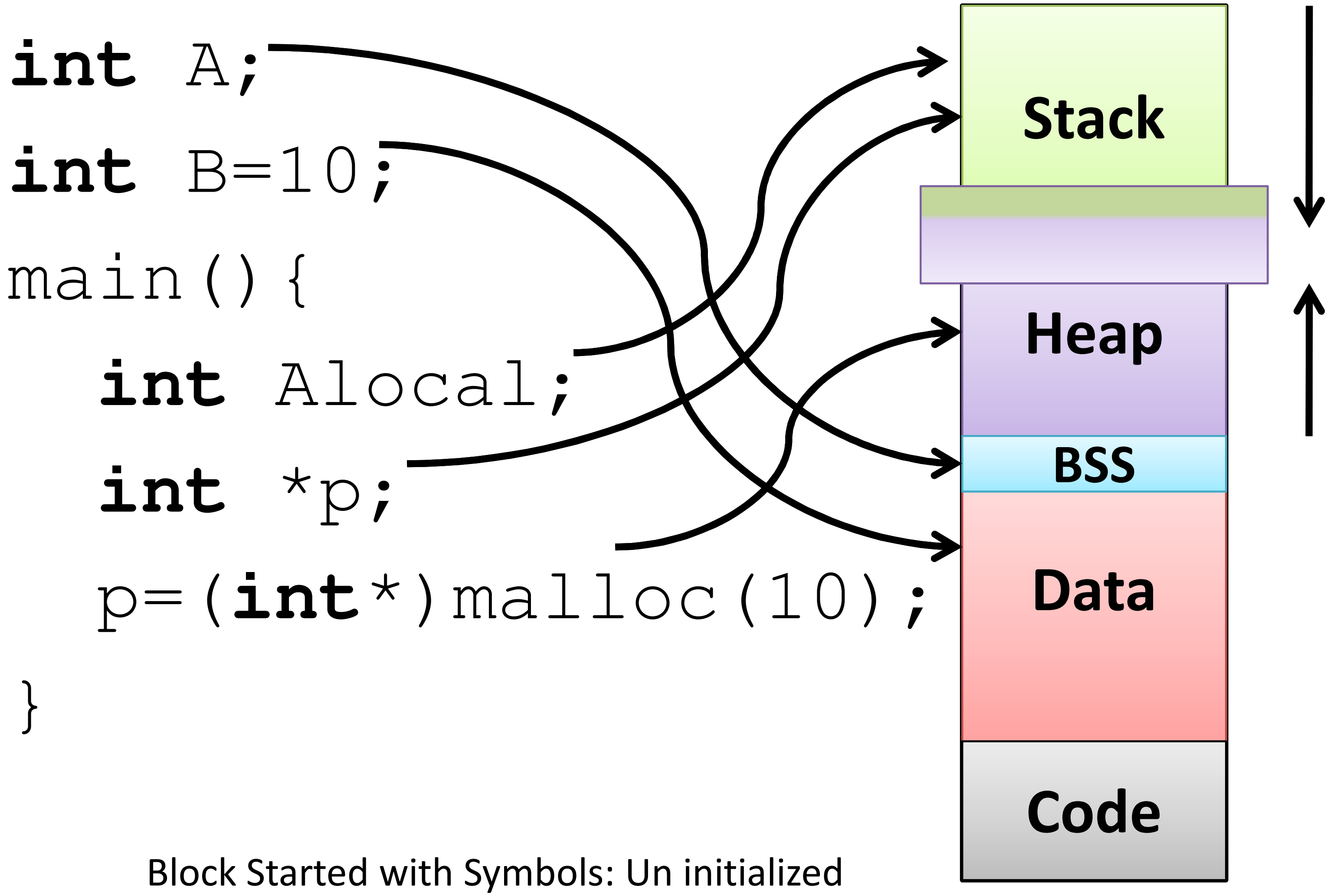
- Memory Layout of C Program
- Process Concepts
- Process States
- Process Control Block (PCB)
- IPC (Inter Process Communication)
- Threads ()
- Scheduling: **Theoretical Analysis**

Process in Mmeory: Memory layout of C program

- Stack
  - automatic (default), local
  - Initialized/uninitialized
- Data
  - Global, static, extern
  - BSS: Block Started by Symbol
  - BBS: Uninitialized Data Seg.
- Code: program instructions
- Heap : malloc, calloc



Memory layout of C program



```
int A;
int B=10;
main(){
    int Alocal;
    int *p;
    p=(int*)malloc(10);
}
```

Block Started with Symbols: Un initialized

Process Concept

- **Process** – a program in execution; process execution must progress in sequential fashion
- Multiple parts
  - The program code, also called **text section**
  - Current activity including **PC**, processor registers
  - **Stack** containing temporary data
    - Function parameters, return addresses, local variables
  - **Data section** containing global variables
  - **Heap** containing memory dynamically allocated during run time

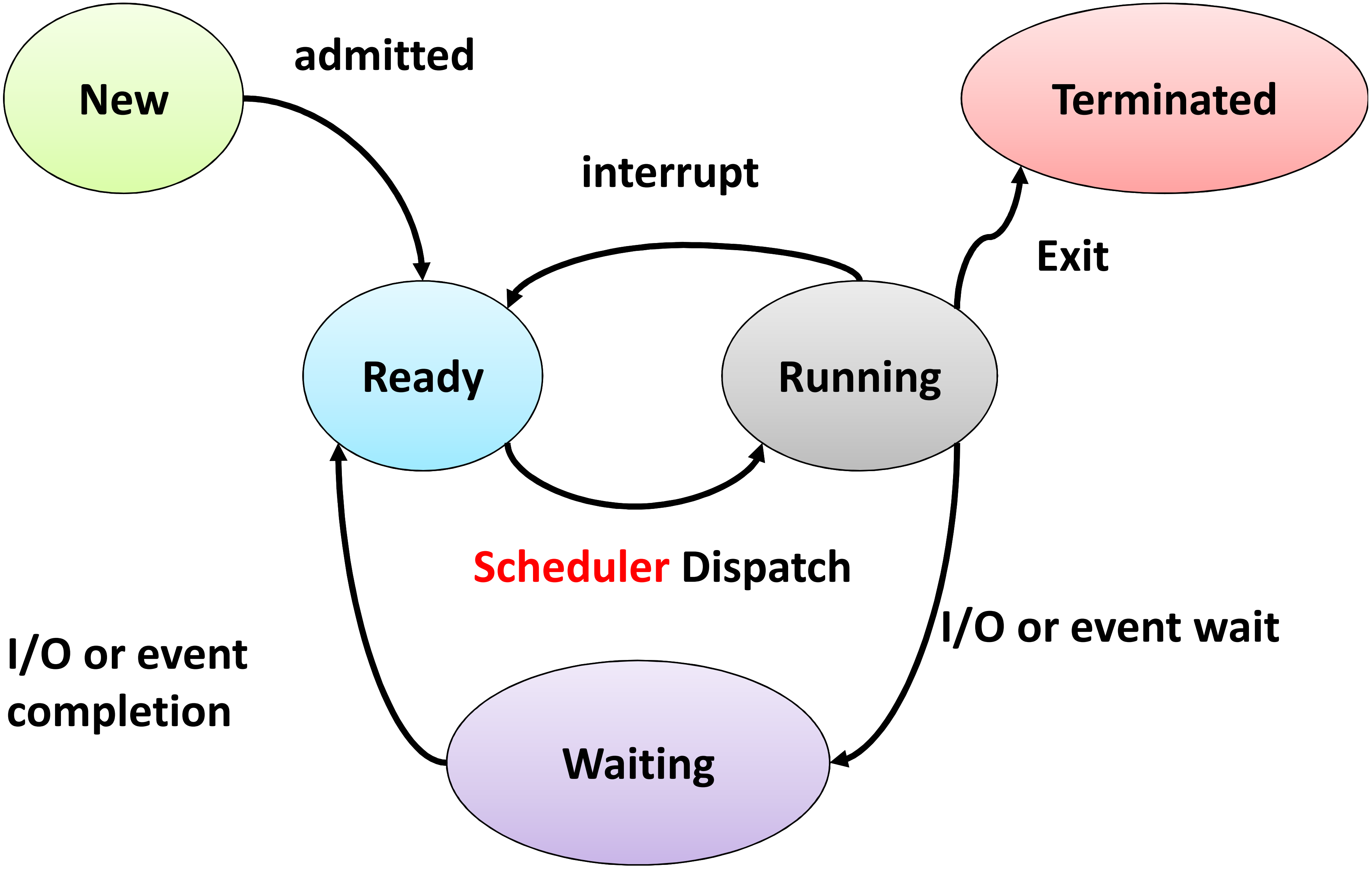
Process Concept (Cont.)

- Program is ***passive*** entity stored on disk (**executable file**), process is ***active***
  - Program becomes process when executable file loaded into memory
- Execution of program started via GUI mouse clicks, command line entry of its name, etc
- One program can be several processes
  - Consider multiple users executing the same program

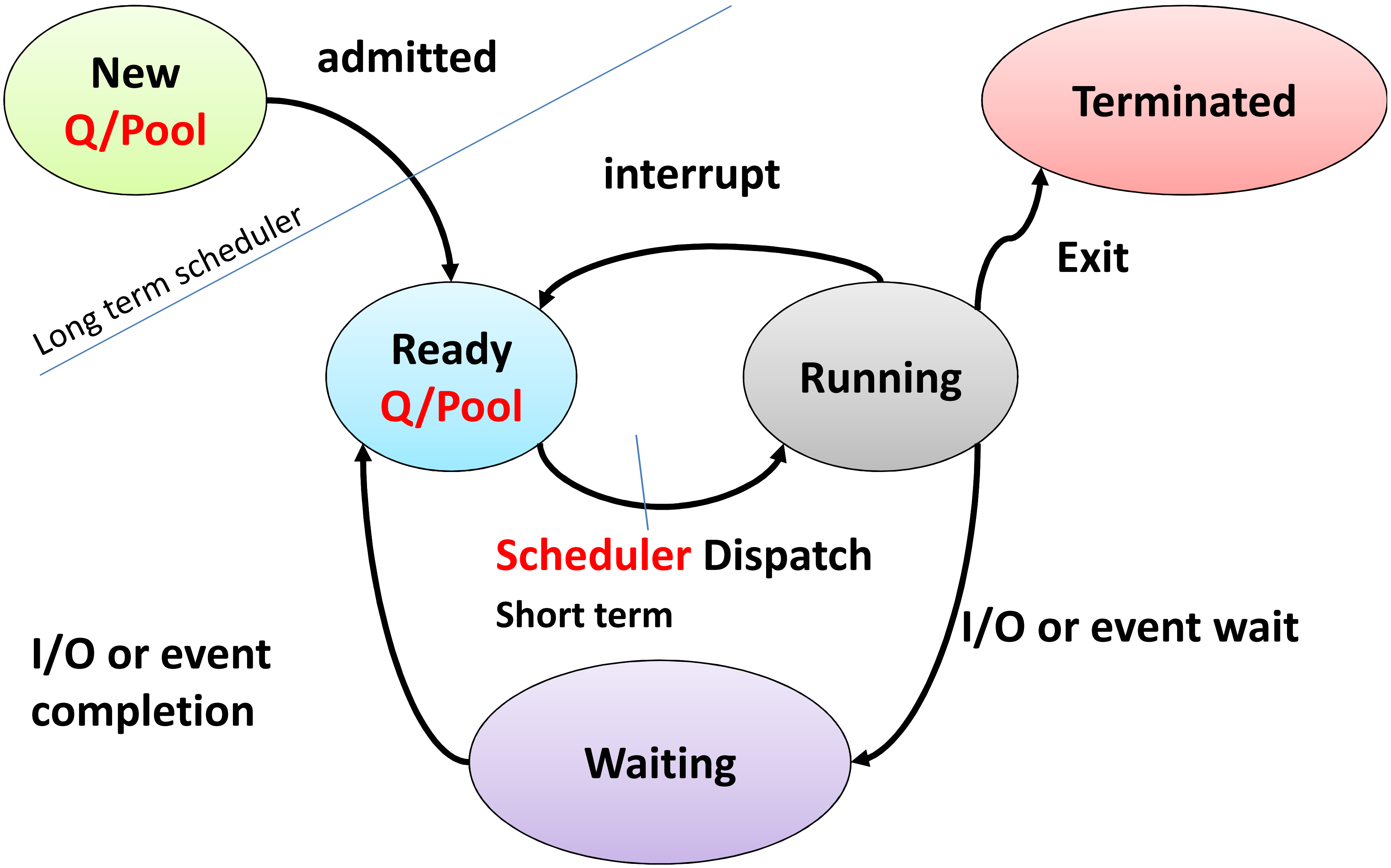
Process State

- As a process executes, it changes **state**
  - **new**: The process is being created
  - **running**: Instructions are being executed
  - **waiting**: The process is waiting for some event to occur
  - **ready**: The process is waiting to be assigned to a processor
  - **terminated**: The process has finished execution

Process State: State Diagram



Process State: State Diagram



Process Control Block (PCB)

Information associated with each process

(also called **task control block**)

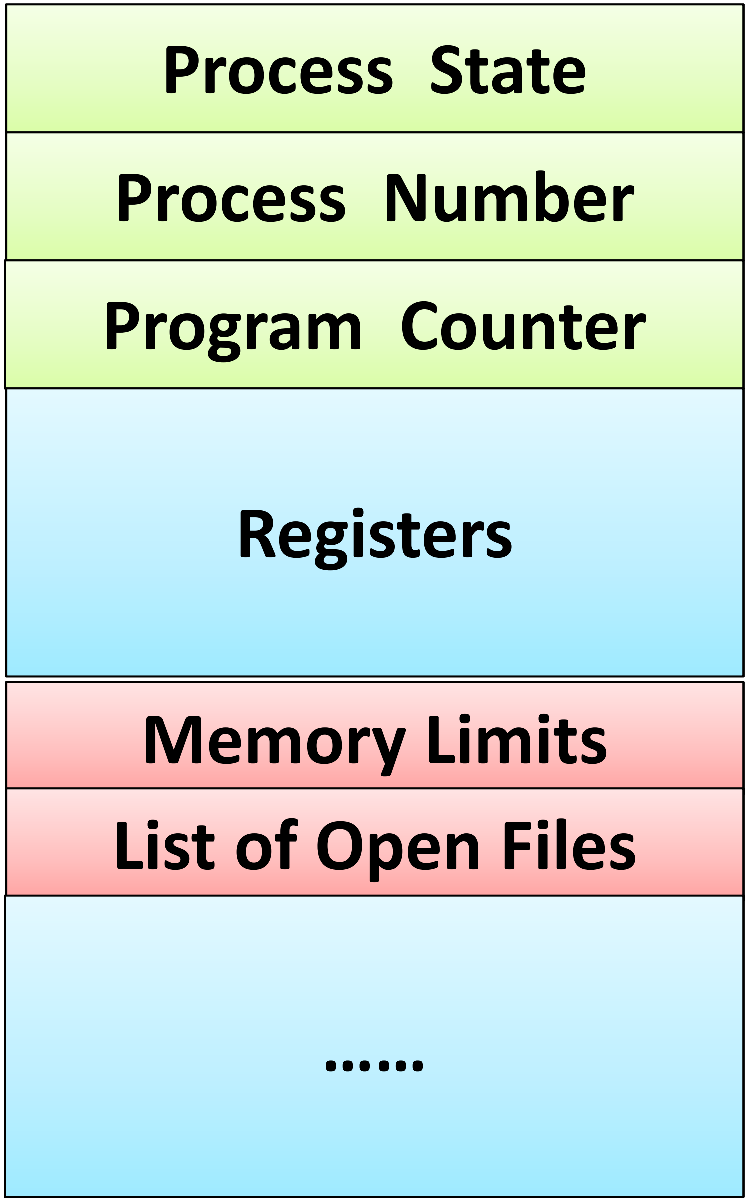
- Process state – running, waiting, etc
- Program counter – location of instruction to next execute
- CPU registers – contents of all process-centric registers
- CPU scheduling information- priorities, scheduling queue pointers

Process Control Block (PCB)

Information associated with each process Cntd..

- Memory-management information – memory allocated to the process
- Accounting information – CPU used, clock time elapsed since start, time limits
- I/O status information – I/O devices allocated to process, list of open files

Process Control Block (PCB)



Process State

Process Number

Program Counter

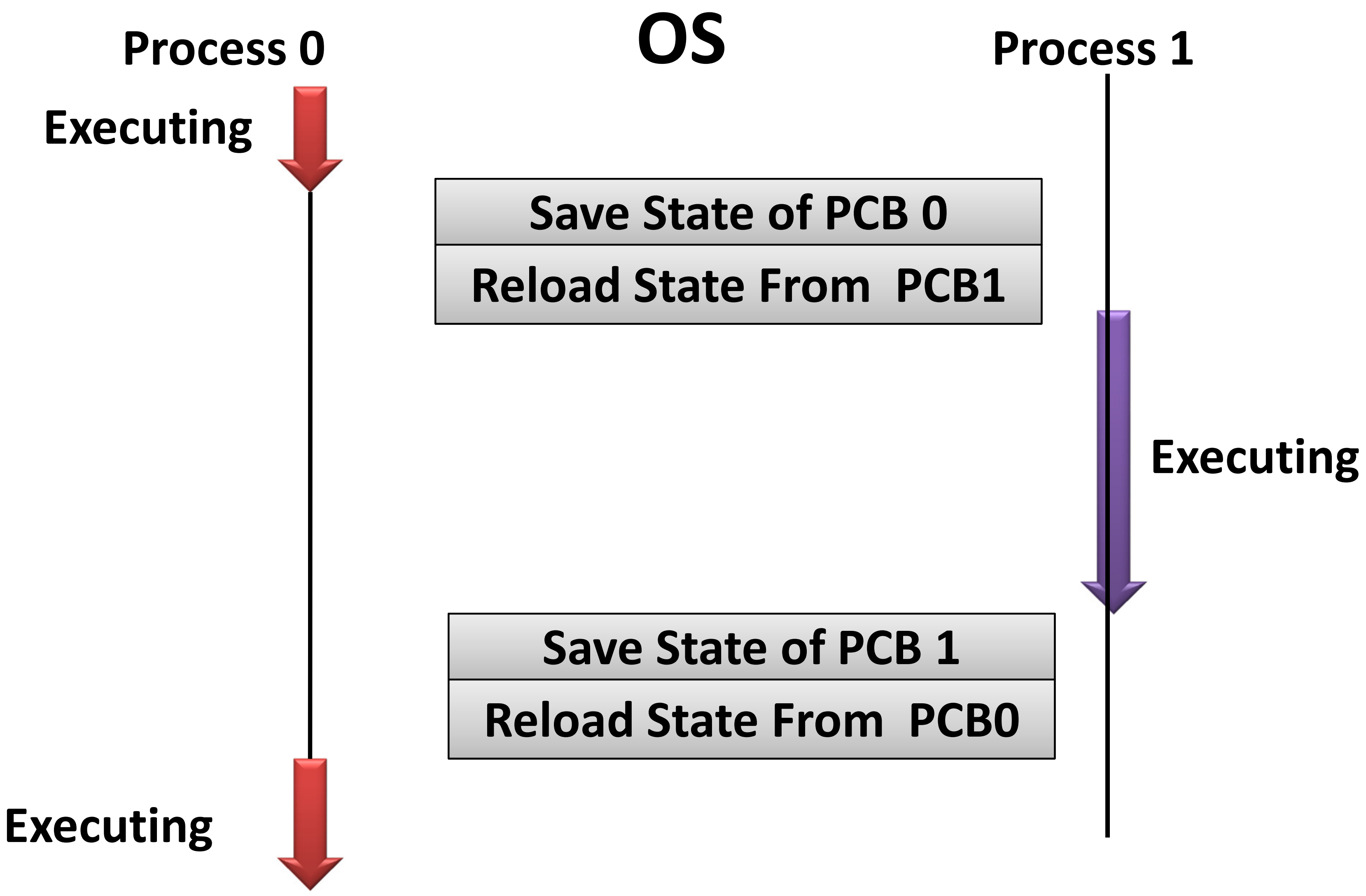
Registers

Memory Limits

List of Open Files

……

CPU Switch From Process to Process



Process 0

OS

Process 1

Executing

Save State of PCB 0

Reload State From PCB1

Executing

Save State of PCB 1

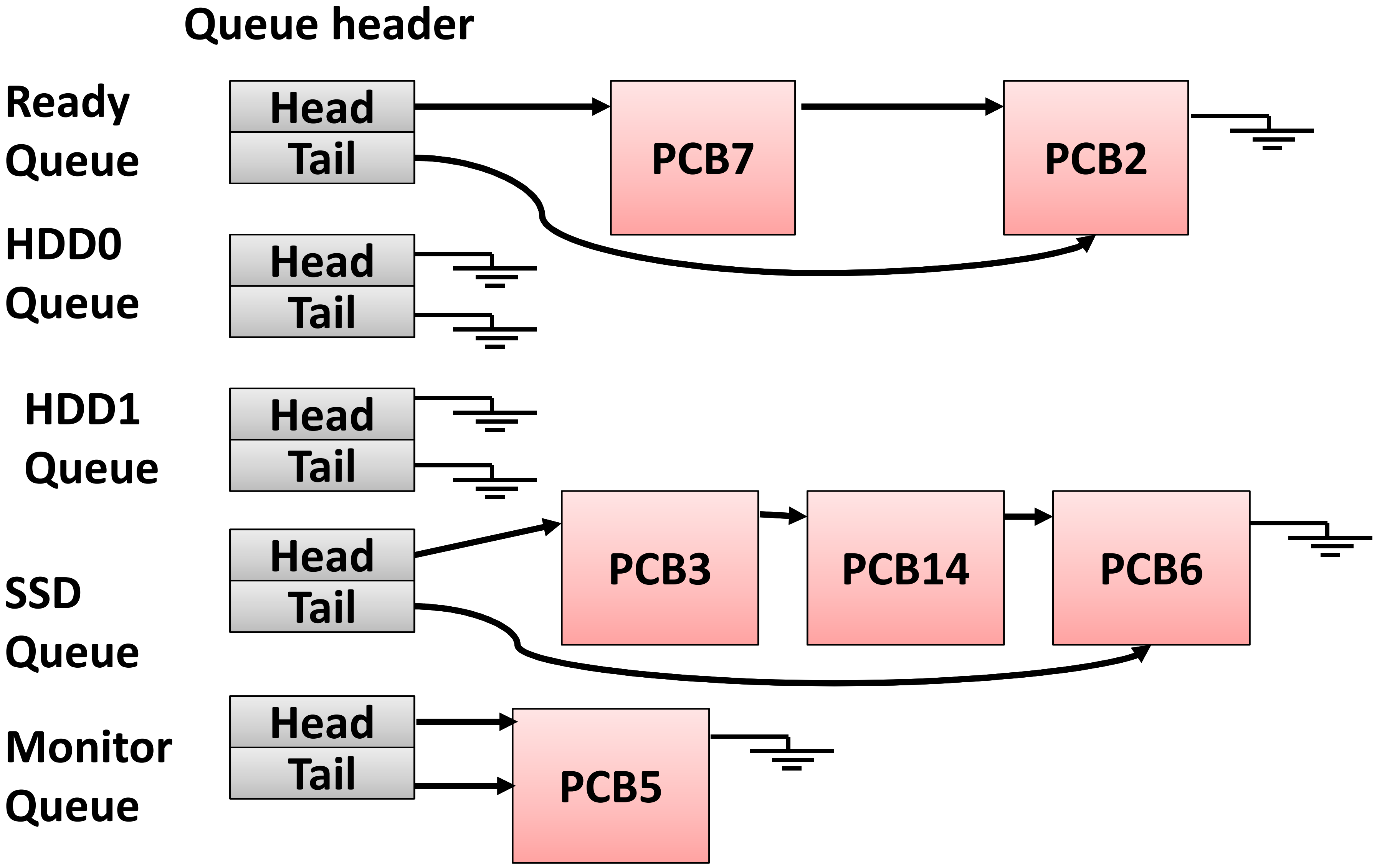
Reload State From PCB0

Executing

Process Scheduling

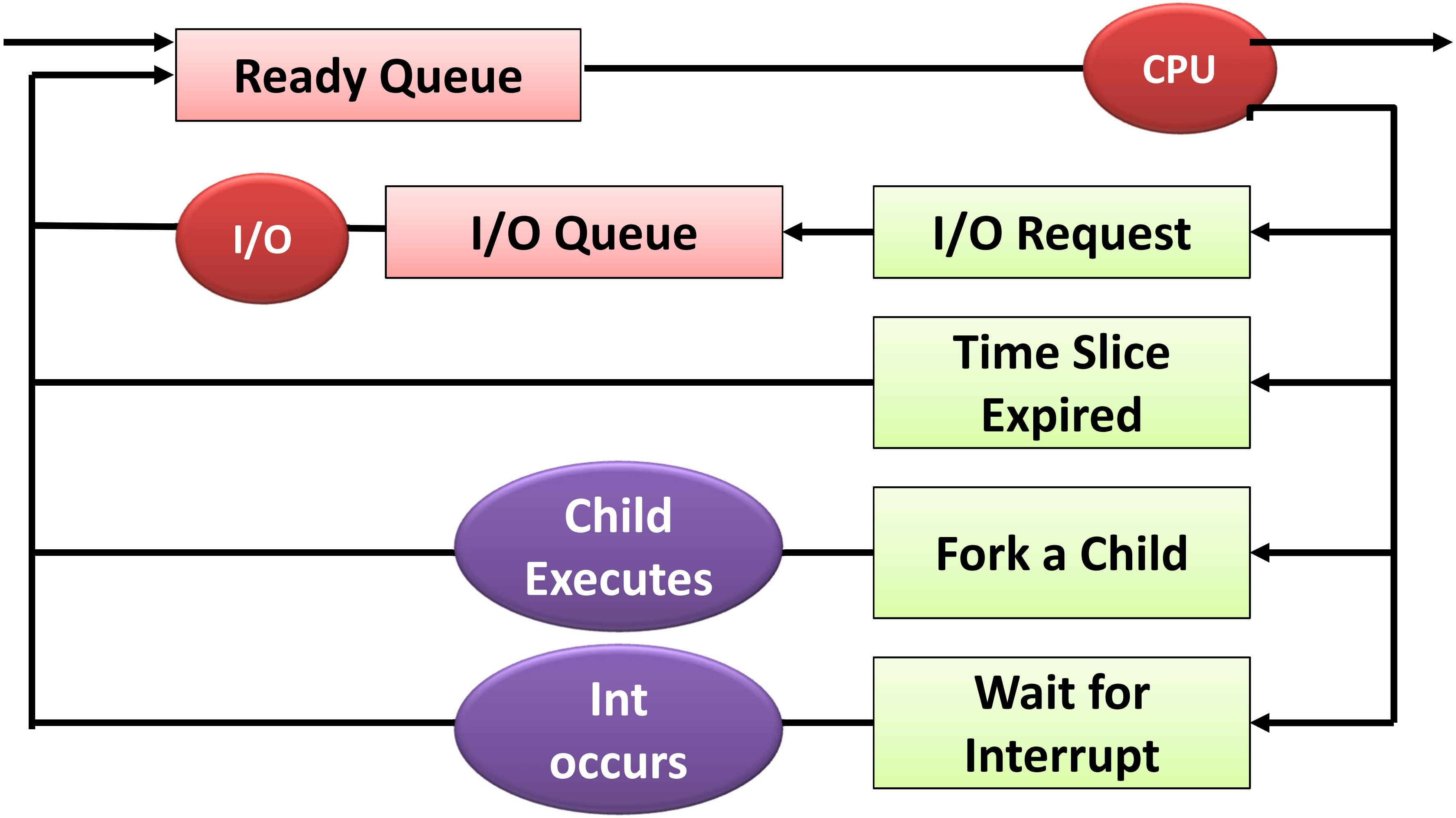
- Maximize CPU use, quickly switch processes onto CPU for time sharing
- **Process scheduler** selects among available processes for next execution on CPU
- Maintains **scheduling queues** of processes
  - **Job queue** – set of all processes in the system
  - **Ready queue** – set of all processes residing in main memory, ready and waiting to execute
  - **Device queues** – set of processes waiting for an I/O device
  - Processes migrate among the various queues

Ready Queue & I/O Device Queues



Representation of Process Scheduling

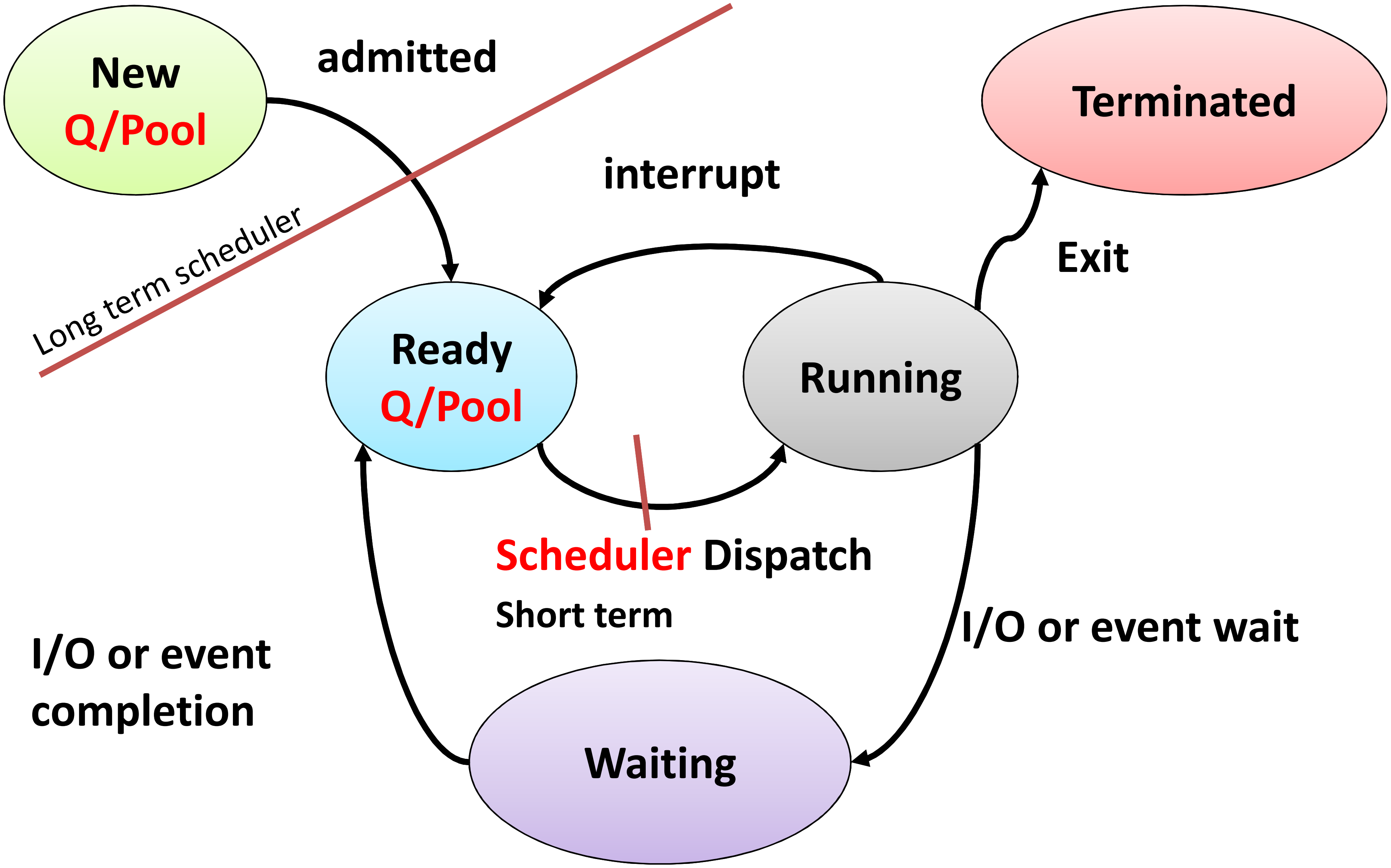
- **Queueing diagram** represents queues, resources, flows



Schedulers

- **Short-term scheduler** (or **CPU scheduler**) – selects which process should be executed next and allocates CPU
  - Sometimes the only scheduler in a system
  - Short-term scheduler is invoked frequently (milliseconds) ⇒ (must be fast)
- **Long-term scheduler** (or **job scheduler**) – selects which processes should be brought into the ready queue
  - Long-term scheduler is invoked infrequently (seconds, minutes) ⇒ (may be slow)
  - The long-term scheduler controls the **degree of multiprogramming**

Process State: State Diagram



Schedulers

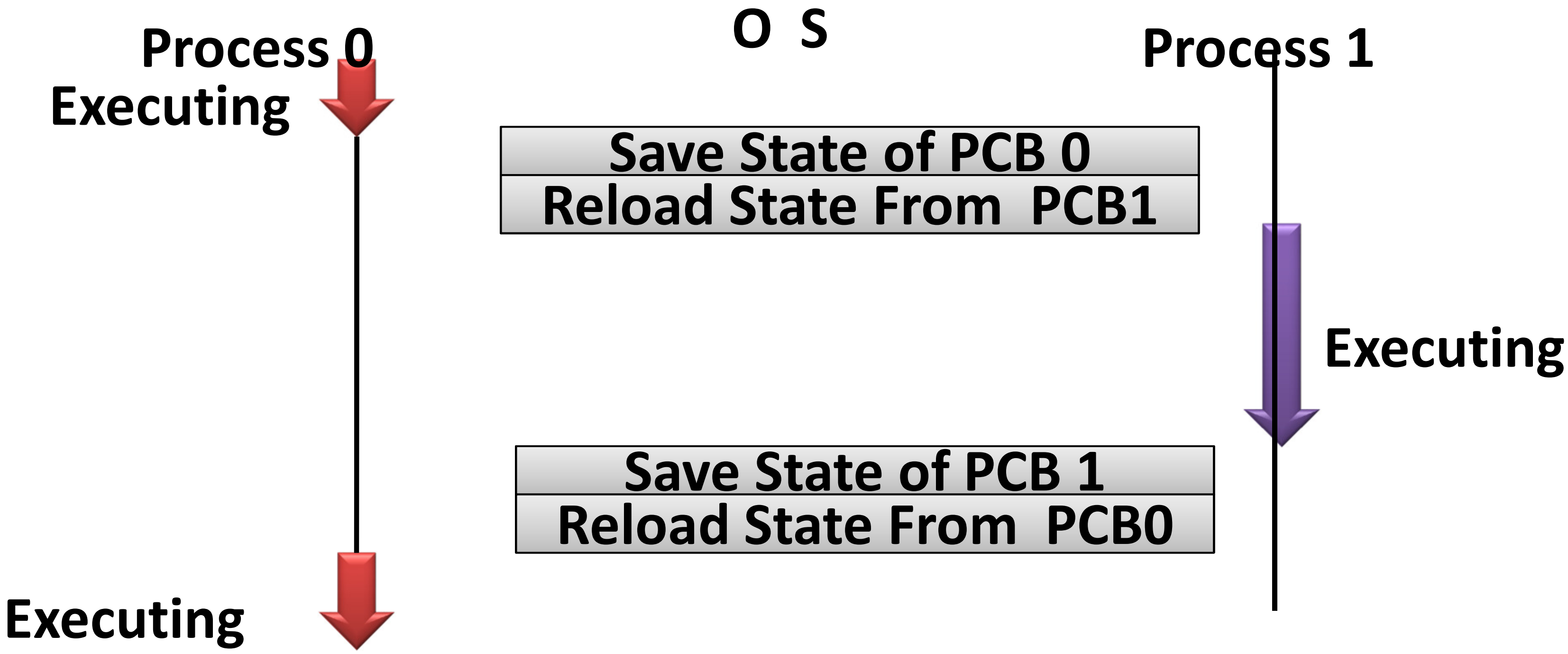
- Processes can be described as either:
  - **I/O-bound process** – spends more time doing I/O than computations, many short CPU bursts
    - Example $cp file1 file2
  - **CPU-bound process** – spends more time doing computations; few very long CPU bursts
    - Example $./fib 100 *// fib(n)=fib(n-1)+fib(n-2)*
- Long-term scheduler strives for good ***process mix***

Addition of Medium Term Scheduling

- **Medium-term scheduler** can be added if degree of multiple programming needs to decrease
  - Remove process from memory, store on disk, bring back in from disk to continue execution: **swapping**

Context Switch

- When CPU switches to another process, the system must **save the state** of the old process and load the **saved state** for the new process via a **context switch**
- **Context** of a process represented in the PCB



Context Switch

- Context-switch time is overhead; the system does no useful work while switching
  - The more complex the OS and the PCB → the longer the context switch
- Time dependent on hardware support
  - Some hardware provides multiple sets of registers per CPU → multiple contexts loaded at once

Many time Context switch code written Manually.
Compiler Generated code may not be efficient

Thanks