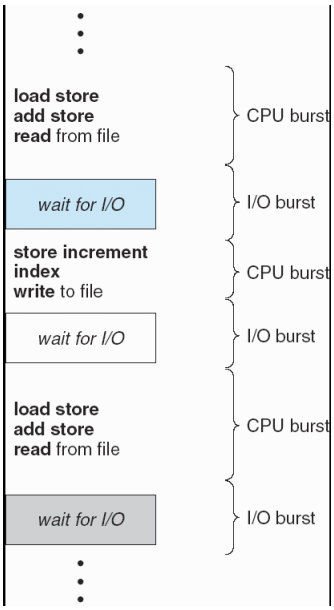
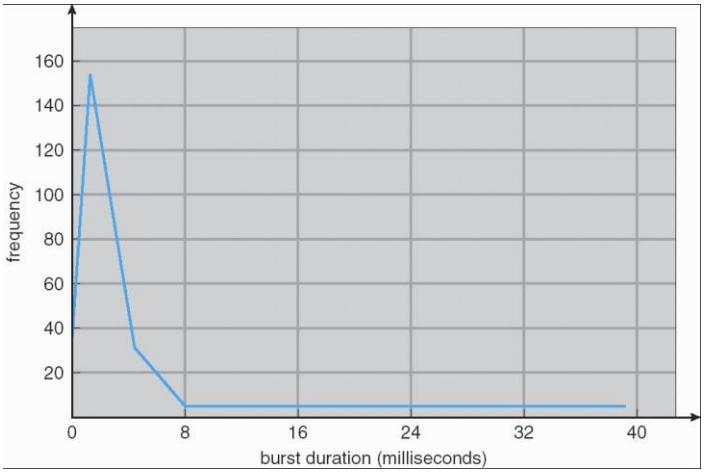
CPU-I/O Burst Cycle

- Process execution comprises a cycle of CPU execution & I/O wait
- Process execution begins with a CPU burst, followed by an I/O burst, then another CPU burst, etc…
- Finally, a CPU burst ends with a request to terminate execution



Histogram of CPU-burst times:



- An I/O-bound program typically has many short CPU bursts
- A CPU-bound program might have a few long CPU bursts
- These are important points to keep in mind for the selection of an appropriate CPU-scheduling algorithm

CPU Scheduler

- Selects from among the processes in memory that are ready to execute, and allocates the CPU to one of them
- The short-term scheduler selects a process in the ready queue when the CPU becomes idle
- The ready queue could be a FIFO / priority queue, tree, list…
- The records in the queues are generally process control blocks (PCBs) of the processes

Preemptive Scheduling

- Circumstances under which CPU scheduling decisions take place:
  - When a process switches from the running state to the waiting state (E.g. I/O request) (1)
  - When a process switches from the running state to the ready state (E.g. when an interrupt occurs) (2)
  - When a process switches from the waiting state to the ready state (E.g. completion of I/O) (3)

- When a process terminates (4)

- Non-preemptive/cooperative scheduling
  - Processes are allowed to **run to completion**
  - When scheduling takes place under circumstances 1 & 4
  - There is no choice in terms of scheduling
- Preemptive scheduling
  - Processes that are runnable may be **temporarily suspended**
  - There is a scheduling choice in circumstances 2 & 3
  - Problem: if one process is busy updating data and it is preempted for the second process to run, if the second process reads that data, it could be inconsistent

Dispatcher

- A component involved in the CPU scheduling function
- The dispatcher is the module that **gives control** of the CPU to the process selected by the short-term scheduler
- This function involves:
  - Switching context
  - Switching user mode
  - Jumping to the proper location in the user program to restart that program
- The dispatcher should be as fast as possible, given that it is invoked during every process switch
- Dispatch latency = the time it takes for the dispatcher to stop one process and start another running

Scheduling Criteria

- Different CPU-scheduling algorithms have different properties and the choice of a particular algorithm may favor one class of process over another
- **Criteria** to compare CPU-scheduling algorithms:
  - **CPU utilization**
    - CPU utilization should range from 40% - 90%
  - **Throughput**
    - The number of processes completed per time unit
  - **Turnaround time**
    - The time interval from process submission to completion
    - Formula: Time of completion – Time of submission
    - Formula: CPU burst time + Waiting time (includes I/O)
  - **Waiting time**
    - The sum of the periods spent waiting in the ready queue
    - Formula: Turnaround time – CPU burst time
  - **Response time**

        - The amount of time it takes to start responding, but not the time it takes to output that response
- We want to maximize CPU utilization, and minimize turnaround, waiting & response time