CHAPTER 3

MEDIUM-TERM SCHEDULER

Dr. Soha S. Zaghloul

LAYOUT

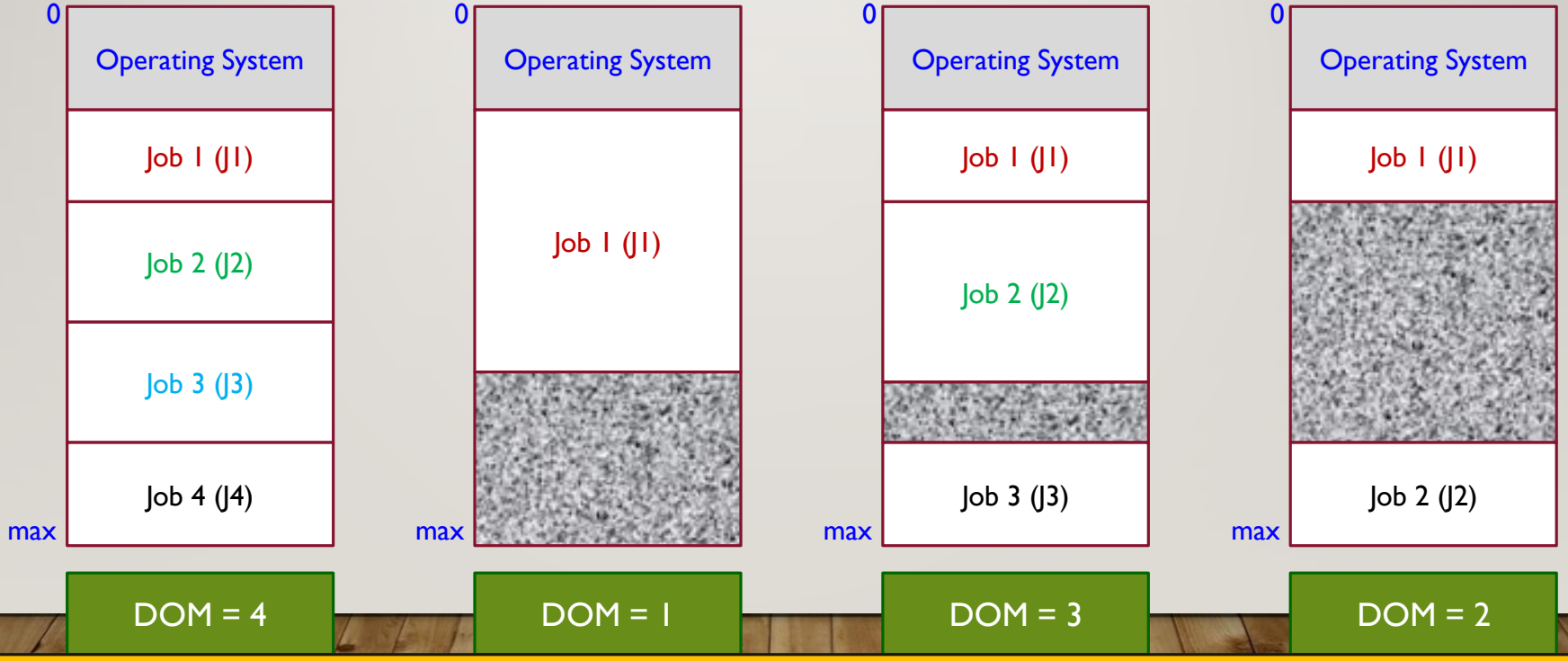
- Degree of Multiprogramming
- CPU-bound & I/O-bound Processes
- Swapping
- Medium-term Scheduler
- Context Switching

Degree of Multiprogramming (1)

Degree of Multiprogramming (DOM) designates the number of processes available in the memory.

The term "multiprogramming" in this context designates a multiprocessing environment. In other words, the term "Degree of Multiprogramming" applies on both multiprogramming and multitasking.

Consider the following figure:



DEGREE OF MULTIPROGRAMMING (2)

As the DOM increases, as there is less probability that the CPU stays idle.

When the DOM becomes stable, then the average rate of memory allocation (*process creation*) is equal to the average rate of memory deallocation (*processes termination*).

The long-term scheduler is invoked only when a process deallocates the memory.

CPU-Bound & I/O Bound Processes (1)

A burst is the time spent by a process allocating the CPU *continuously* without being stopped for any reason.

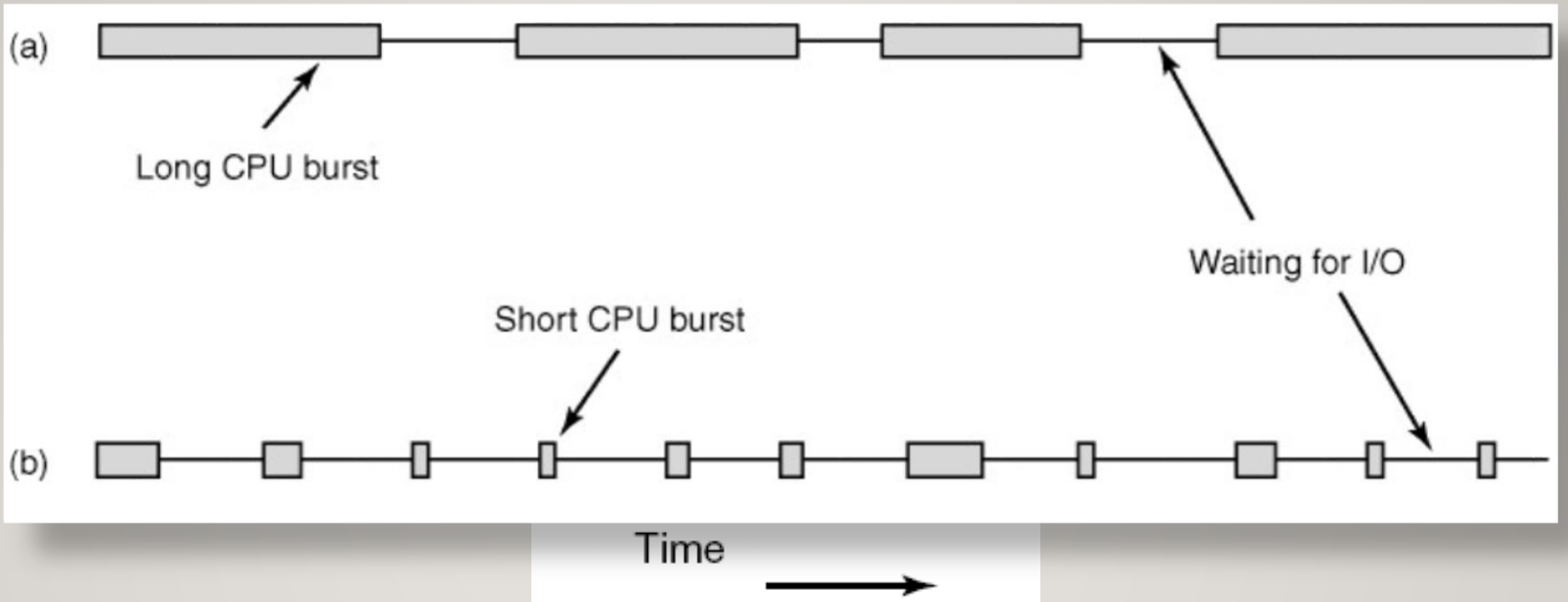
In other words, a burst is the time spent by a process to be continuously in the Running state.

A CPU-bound process is a process that spends most of its lifetime executing instructions on the CPU; ie. In the Running State.

An I/O process is a process that spends most of its lifetime processing input and/or output instructions; ie. In the Waiting State.

CPU-Bound & I/O Bound Processes (2)

Consider the following figure:



The process in figure (a) spends most of its time allocating the CPU for computations ➔ this is a CPU-bound process.

The process in figure (b) spends most of its time waiting for an I/O ➔ this is an I/O-bound process.

CPU-bound processes have long CPU bursts, and infrequent I/O waits.
I/O-bound processes have short CPU burst, and frequent I/O waits.

Note that the key factor is the length of the CPU bursts rather than that of the I/O bursts.

As CPUs get faster, processes tend to get more I/O-bound: CPU's speed is improving too much faster than disks' speed.

CPU-Bound & I/O Bound Processes (3)

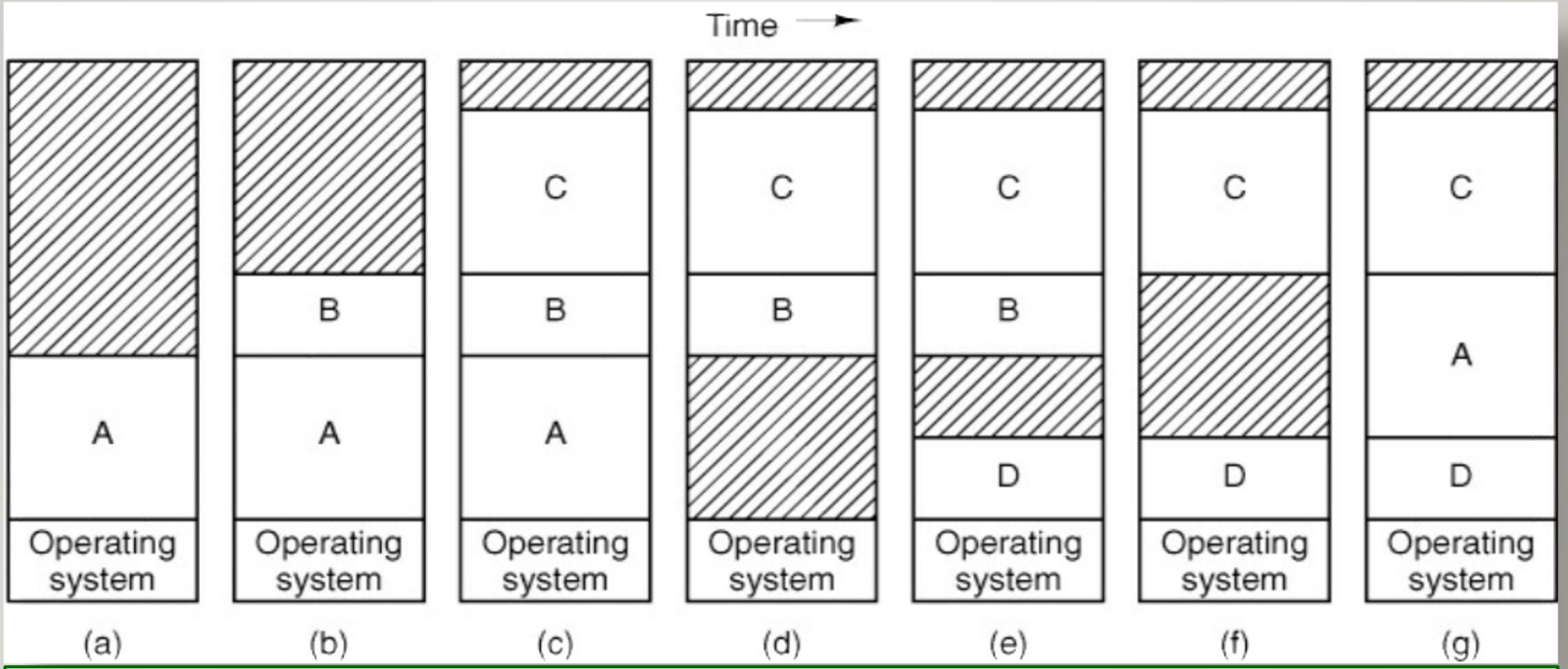
An efficient OS is designed to have a long-term scheduler that selects a balanced mix of CPU-bound and I/O-bound processes.

- If all processes are I/O-bound, then the ready queue will almost be empty ➔ the short-term scheduler will have little to do ➔ the CPU will stay idle.
- If all processes are CPU-bound, then I/O waiting queue will almost be empty ➔ devices will stay unused ➔ the system will get unbalanced.

SWAPPING

Swapping is the technique of bringing a process into memory, running it for a while, then putting it back on the disk.

In fact, memory allocation changes as processes come into memory and leave it:



(a)-(c): Processes A, B & C are created and swapped in from disk to memory.

(d): Process A is swapped out from memory to disk.

(e): Process D is swapped in from disk to memory.

(f): Process B terminates execution and deallocates the memory.

(g): Process A returns back (swapped in) in a different location in memory.

MEDIUM-TERM SCHEDULER (1)

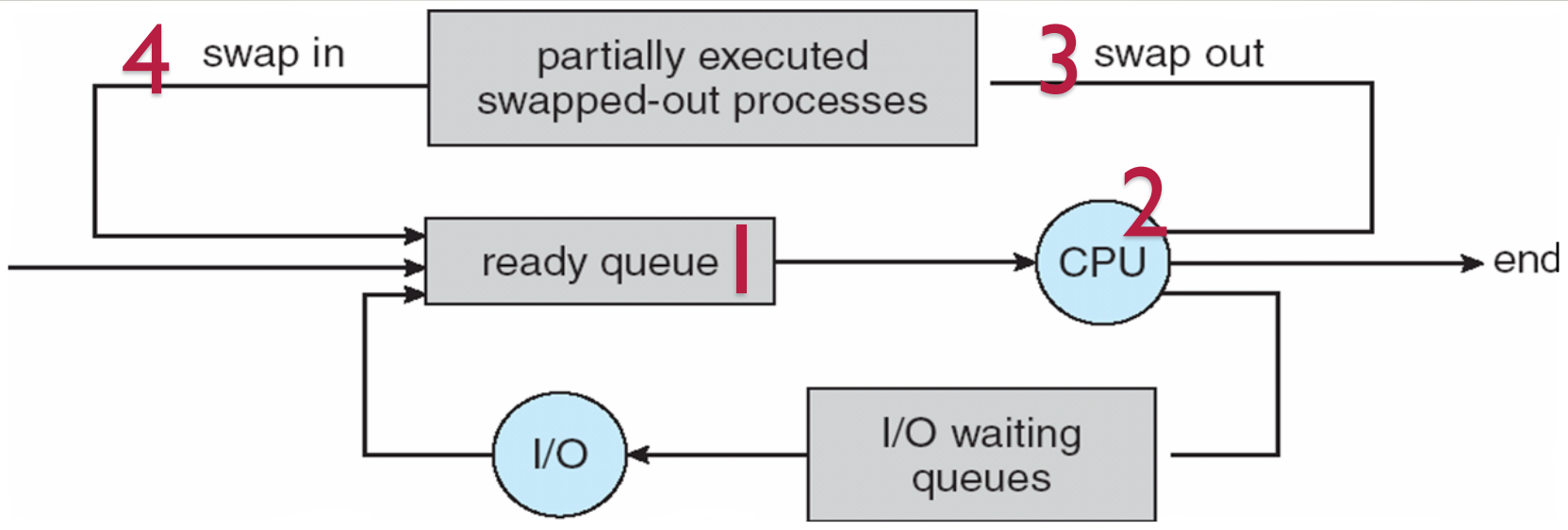
Modern OSs introduce an additional level of scheduling that is responsible for swapping: this is the medium-term scheduler.

Sometimes, it is a good practice to perform swapping: remove a process from memory and from active contention for the CPU, and thus reduce the DOM. Later, the process is reintroduced into memory, and resumes execution where it left off.

Swapping may be needed to improve the process mix, or because a change in memory requirements has overcommitted available memory, requiring some space to be freed up *(this will be discussed in more details later in this course)*.

MEDIUM-TERM SCHEDULER (2)

Consider the following figure:



(1) The long-term scheduler selects P1 to be added to the ready queue.

(2) The short-term scheduler selects P1 to be dispatched.

(3) The medium-term scheduler selects P1 to swap out (from memory to disk).

(4) Later on, the medium-term scheduler selects P1 to swap in (from disk to memory).

Obviously, swapped out processes wait in a dedicated queue (swap queue) in order to be re-assigned to the CPU by the medium-term scheduler.

CONTEXT SWITCHING (1)

The context of a process is its complete information as stored in the Process Control Block (PCB).

This includes the values stored in the CPU registers generated by the executing process, memory management information, etc…

As a process - say P1 - is swapped out (or interrupted for any reason), another process – say P2 – is dispatched. Therefore, the context of P2 should replace that of P1 ➔ this is known as context switching.

Later on, when P1 resumes execution, another context switching is necessary to provide the CPU with the final values of P1 before being swapped out/interrupted.

When a context switch occurs, the kernel saves the context of the old process in its PCB and restores the previously saved context of the new process that is scheduled to run.

Context switching imposes an overhead on the CPU since the latter stays idle till the context switching takes place.

CONTEXT SWITCHING (2)

The speed of context switching depends on many parameters such as:

- The hardware
- The memory speed
- Number of registers to be saved/restored
- The existence of special instructions that help in the context switching process; such as load/store registers.

Some processors, such as UltraSPARC, provide multiple sets of registers. A context switching simple requires changing the pointer to the current register set.