Operating Systems
CS2006

Lecture 2

Interrupts

30th January 2023

Dr. Rana Asif Rehman

A Question

- The operating system
  - Gets an input
  - Performs a computation
  - Produces an output
  - And Quits
  - Yes or no?

The answer: No

- The operating system is a *Reactive Program*

Operating System

- Modern Operating Systems are ***Interrupt driven***
- If
  - No process to execute
  - No I/O device to service
  - No user to whom to respond
- Then
- OS will sit quietly, waiting for something to happen
- This something is ***Interrupt***
- At a particular time either a user program is running or operating system is running

Interrupts

- **Interrupts** are signals by external devices, normally I/O devices. They tell the CPU to stop its current activities and execute the appropriate part of the operating system.

Types of Interrupts

- There are three (or broadly two) types of interrupts:
  - Hardware Interrupts
  - Software Interrupts
  - Traps

Hardware Interrupts

- **Hardware Interrupts** - generated by hardware devices to signal that they need some attention from the OS.
  - They may have just received some data (e.g., keystrokes on the keyboard or an data on the Ethernet card);
  - or they have just completed a task which the operating system previously requested, such as transferring data between the hard drive and memory.

Software Interrupts

- **Software Interrupts** are generated by programs when they want to request a system call to be performed by the operating system.

Traps

- **Traps** are generated by the CPU itself to indicate that some error or condition occurred for which assistance from the operating system is needed.

Interrupts and Traps

Occurrence	Cause	OS Mode
Trap	Unexpected occurrence within the program	Kernel
Interrupt	Event outside of program	Kernel
Subroutine call	Subroutine call branch in the program	User
System call	System call branch in the program	Kernel

Simple Instruction Cycle

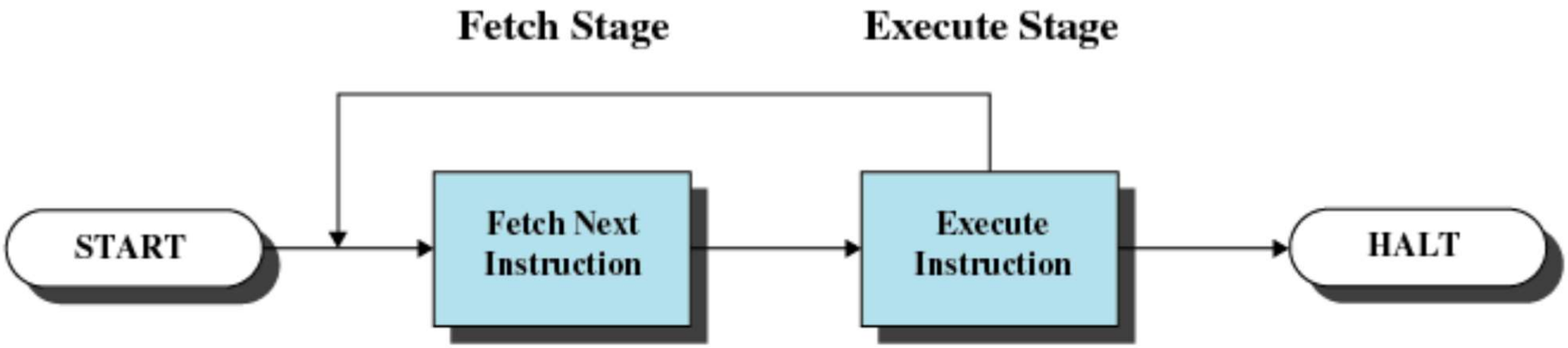


Figure 1.2 Basic Instruction Cycle

Interrupts

- Suspends the normal sequence of execution
- Used to improve processor utilization

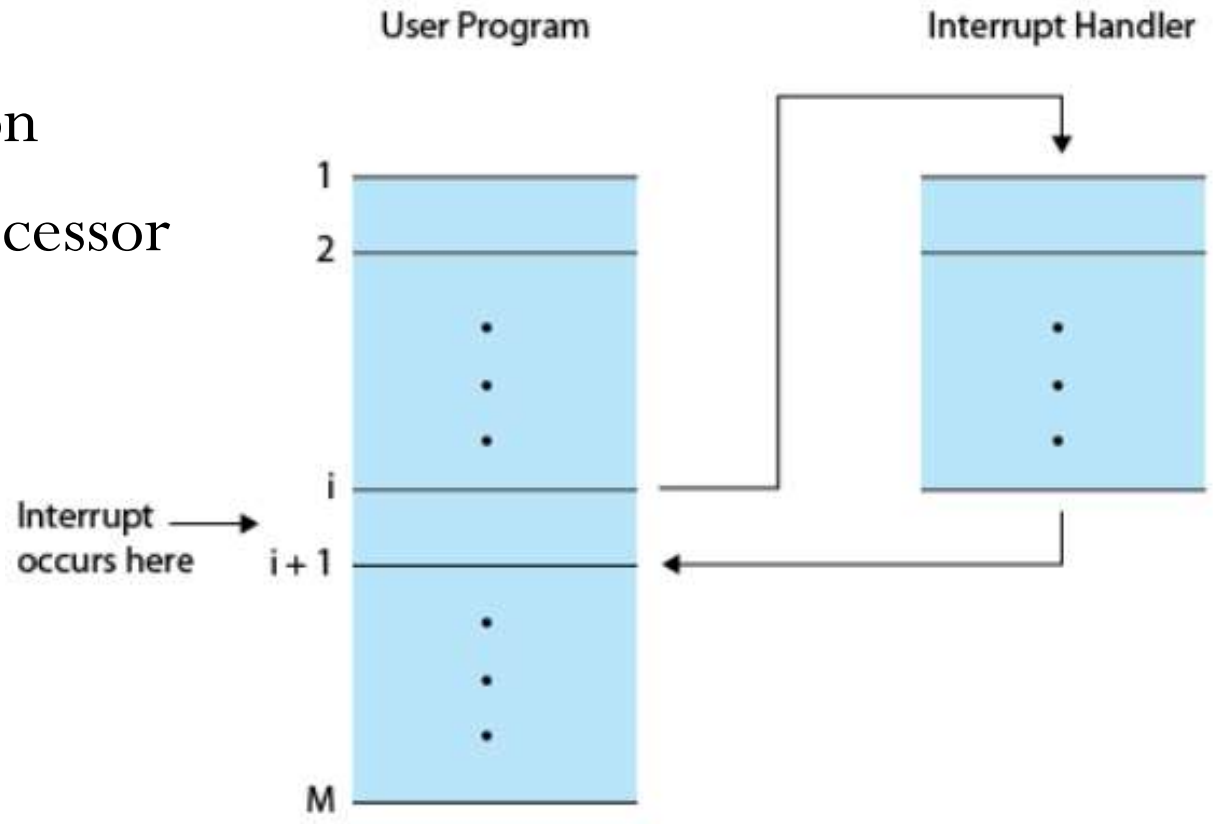


Figure 1.6 Transfer of Control via Interrupts

Interrupt Handling

- Different routines handle different type of interrupts.
- Called ***Interrupt Service Routine (ISR)***
- When the CPU is interrupted it stops what it is doing
- The address of the interrupted instruction is saved
- A generic routine is run
  - This routine examines the nature of interrupt
  - Calls the corresponding ISR
- The ISR's are usually stored in the lower part of the memory
- After the interrupt is serviced, the saved address is loaded to the Program Counter
- The Interrupted computation can resume as though the interrupt had not occurred.

Interrupt Cycle

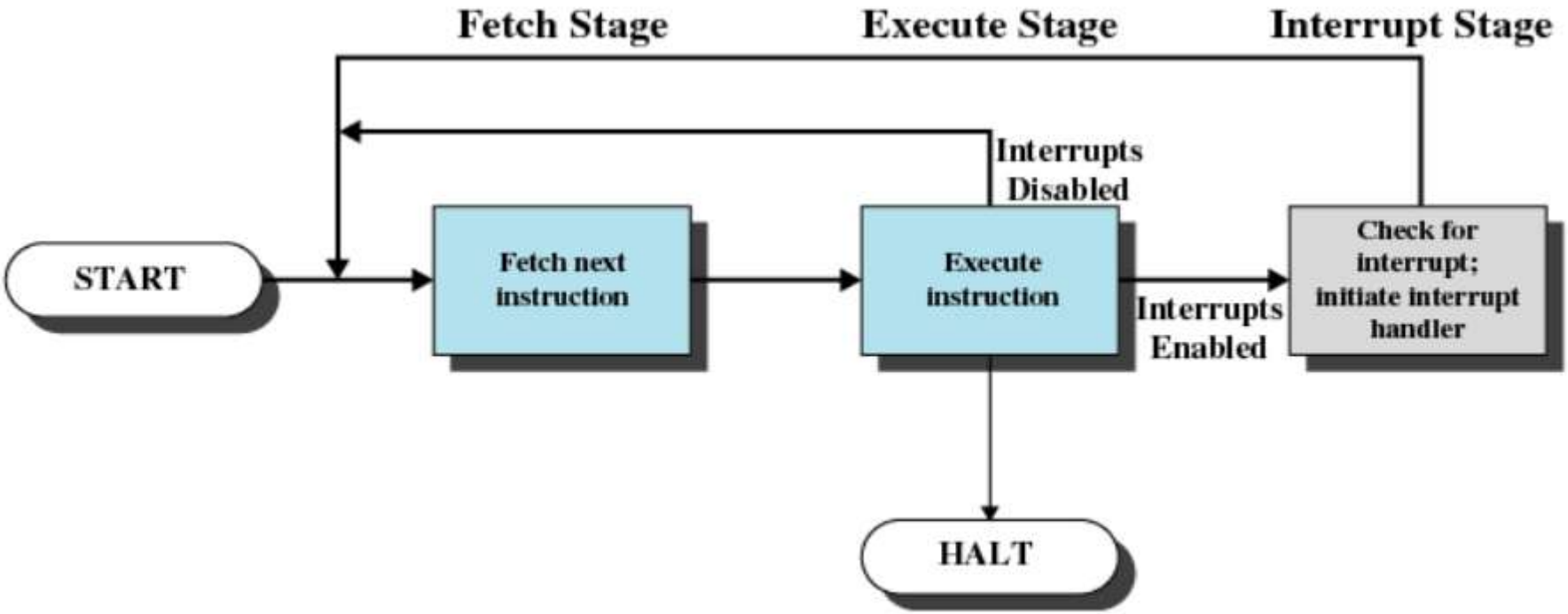
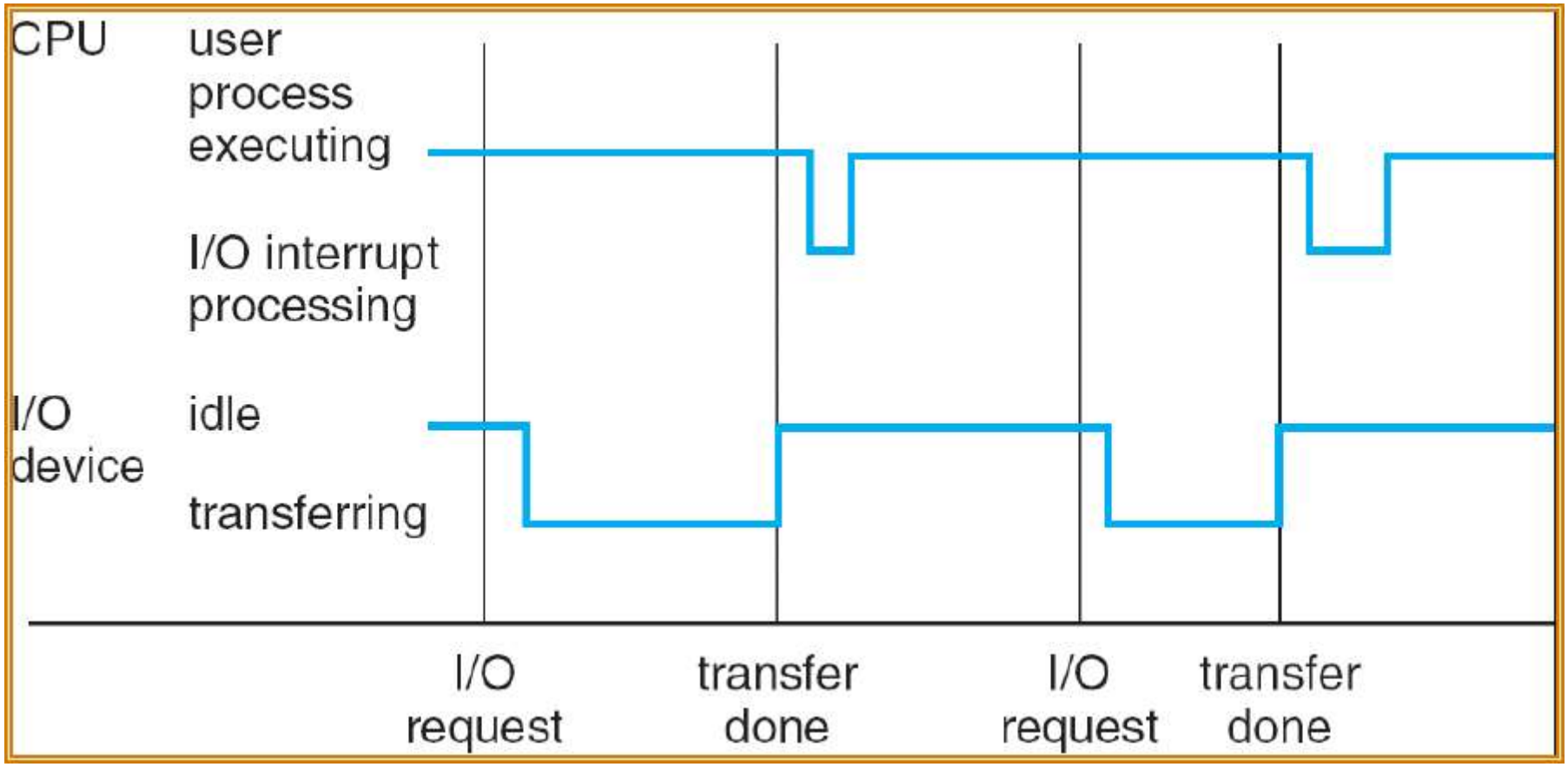


Figure 1.7 Instruction Cycle with Interrupts

Interrupt Timeline



Simple Interrupt Processing

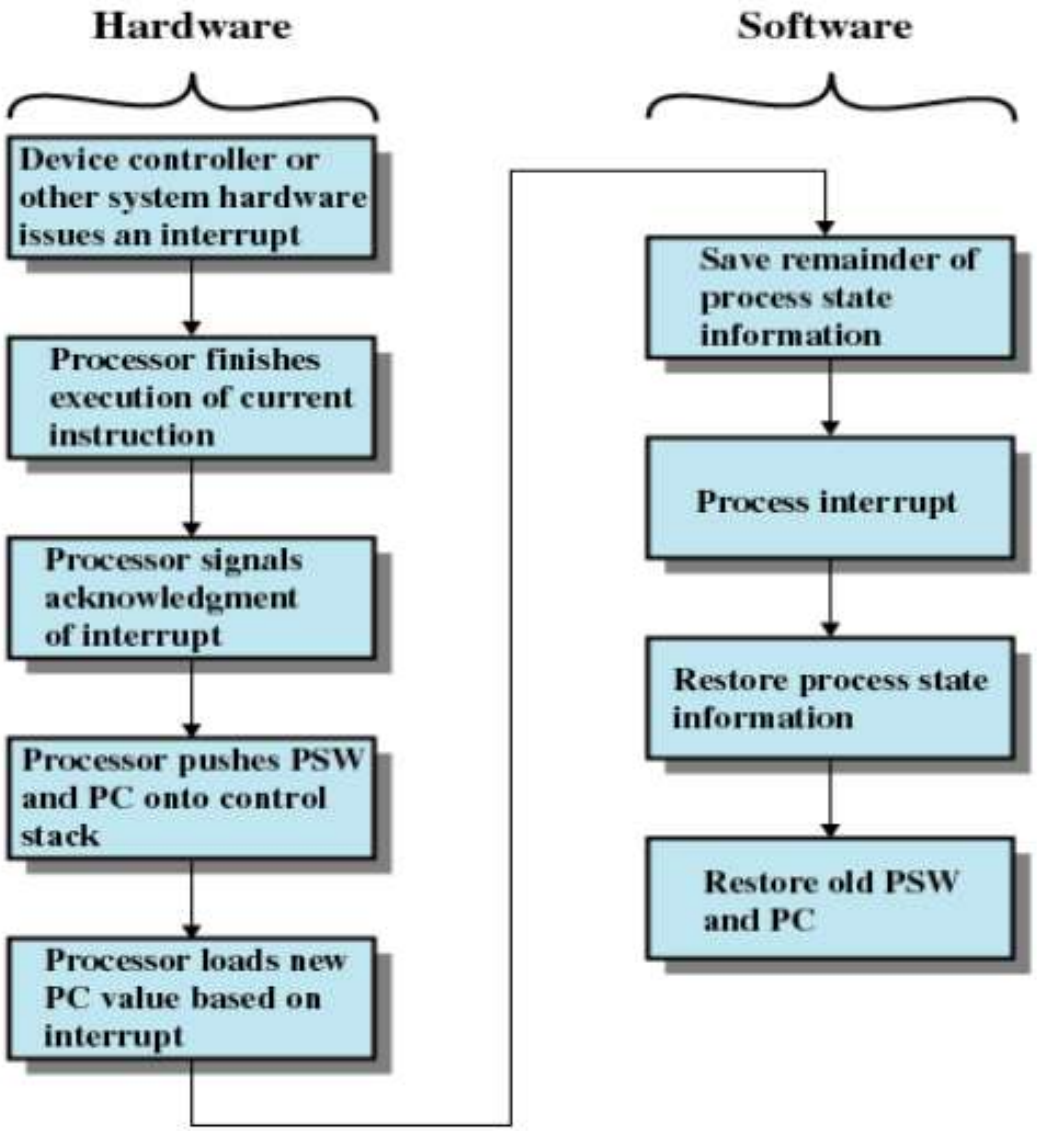
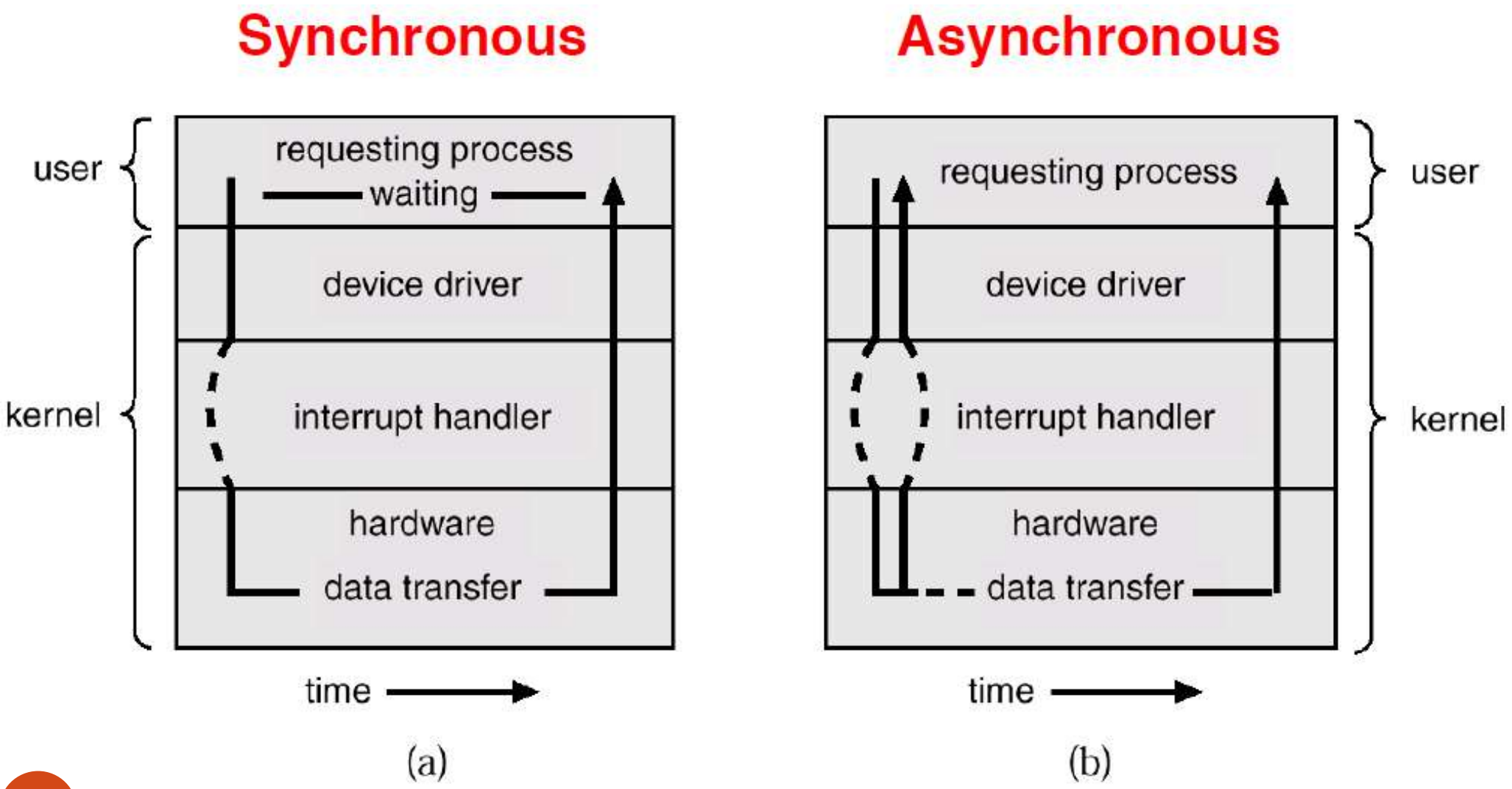


Figure 1.10 Simple Interrupt Processing

Two I/O methods



1. Synchronous I/O

- The OS makes the CPU wait till the I/O complete interrupt is generated
- How can a CPU wait, somehow it has to do F->D->E ??
- Wait can be accomplished by jumping to an infinite loop code in OS:

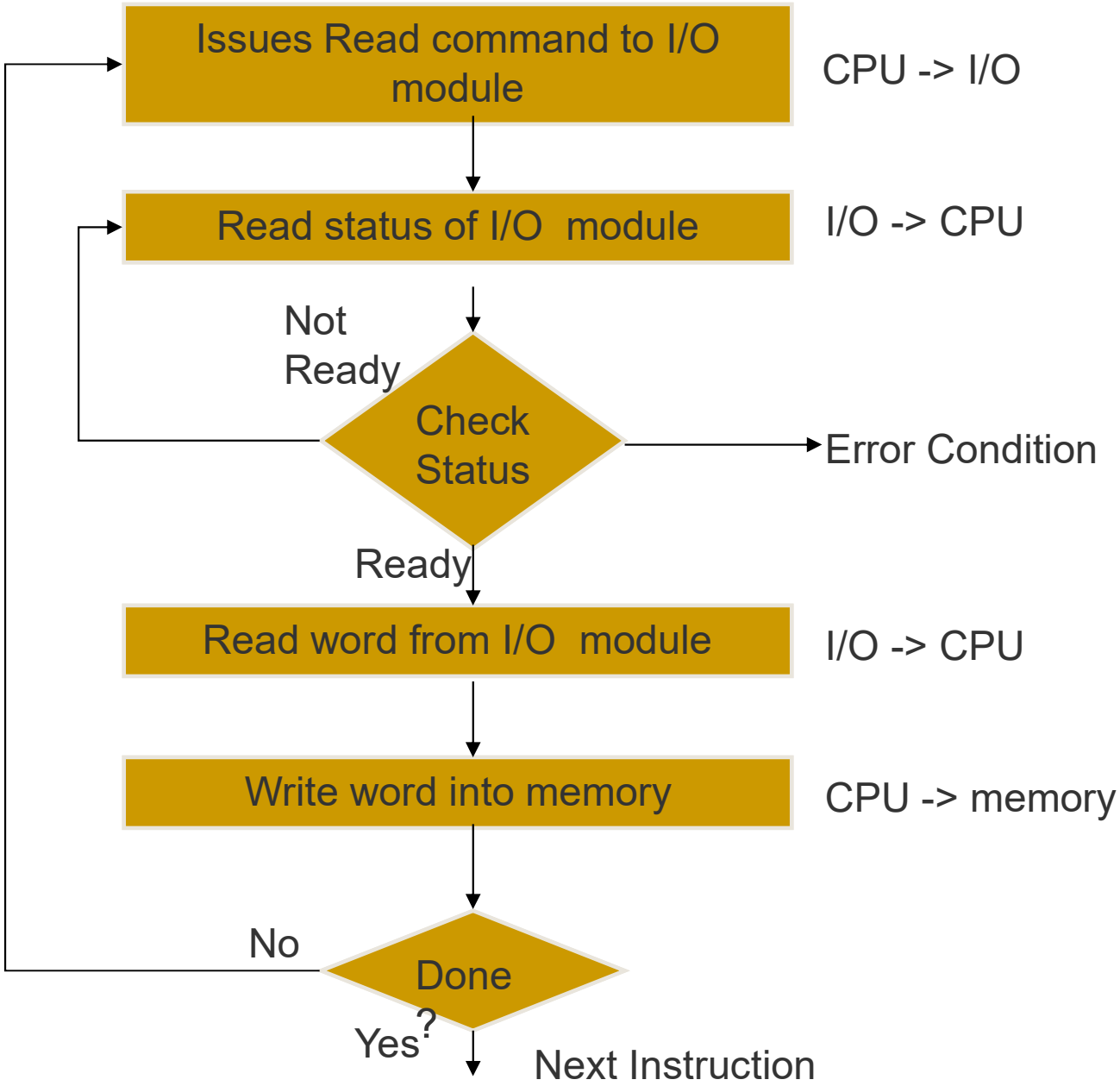
```
Loop: jump Loop
```

- If the devices does not support interrupts
- They will just set a flag in their register to indicate I/O complete
- The above loop would require to include ***Polling*** such devices as well

1. Synchronous I/O

- If the CPU always has to wait for the I/O completion then the number of outstanding I/O requests is?
  - Exactly One
- So, whenever the I/O complete interrupt is generated the OS knows the source device
- Drawback?
- Excludes concurrent I/O operations to several devices

1. Synchronous I/O



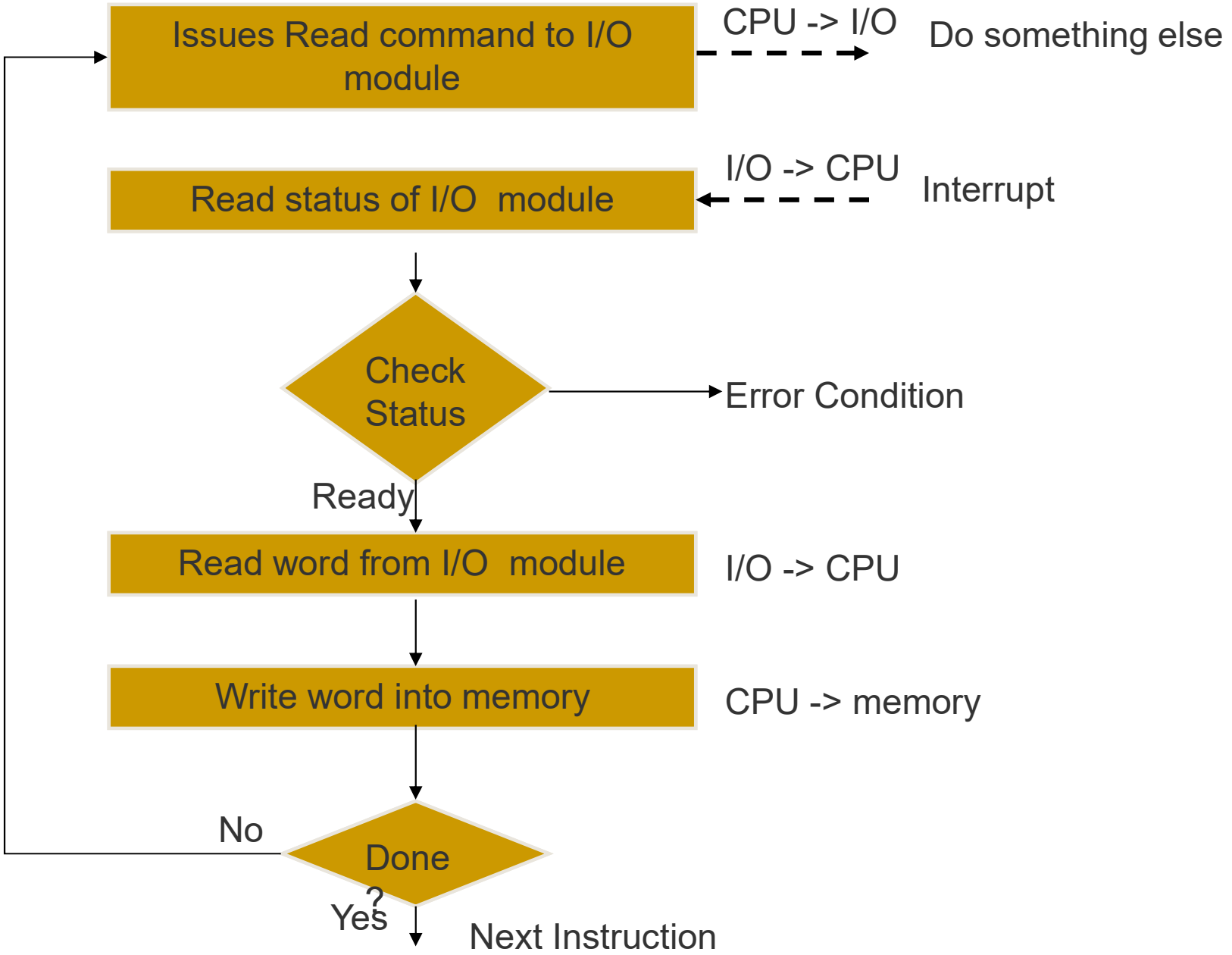
2. Asynchronous I/O

- Start the I/O request
- Return control immediately to the user process
- If the current user program cant run without the I/O, let the current process wait
- Schedule some other user program or operating system code
- If no process ready to run, then still need the idle loop

2. Asynchronous I/O

- The number of outstanding I/O requests is?
  - More then One
- Need to keep track of multiple I/O requests

2. Asynchronous I/O



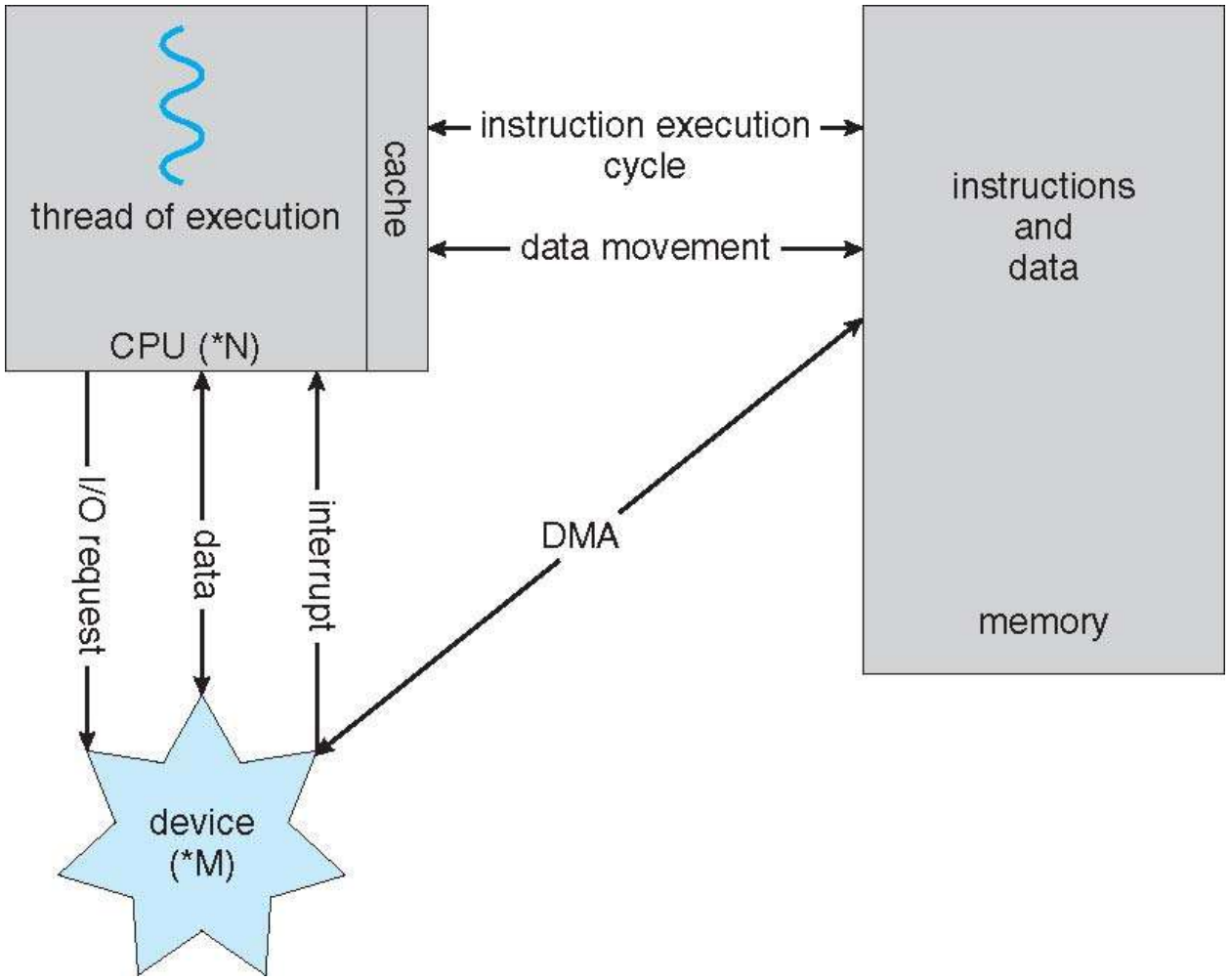
2. Asynchronous I/O

- The I/O controller interrupts when it needs to be serviced
- The OS determines the I/O controller
- Processor reads the word of data from I/O controller to the memory
- Updates the status in the Device Status table
- In case of I/O complete interrupt, checks the additional request
- The control is returned to the user program

3. Direct Memory Access (DMA)

- Drawback of Asynchronous I/O?
- Any data transfer must traverse a path through the processor
- In case of DMA, the following info to the DMA module:
  - Read or Write request
  - Address of the I/O device
  - Starting location in memory to read from or write to
  - Number of words
- The DMA module is then asked to start data transfer
- The above operations are performed by Device Driver

How a Modern Computer Works



A von Neumann architecture

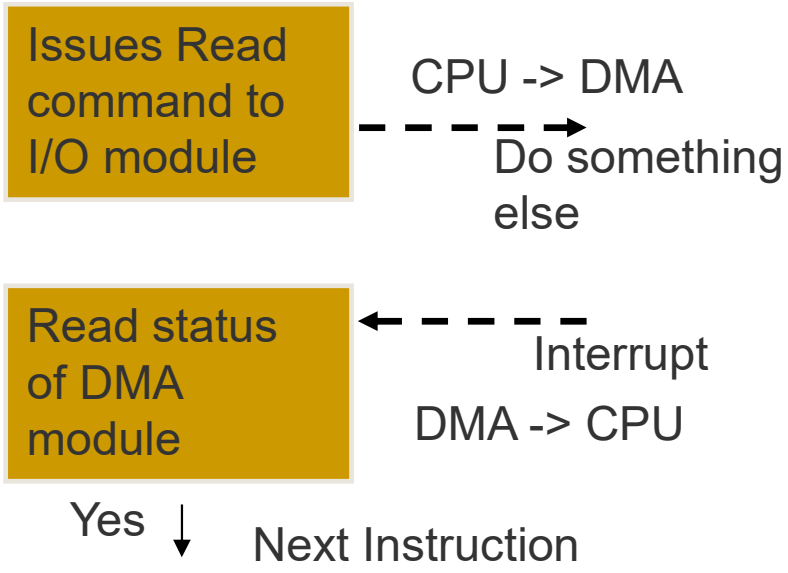
3. Direct Memory Access

- The processor can continue with its work
- The DMA module transfer data directly from the device to the main memory
- But how many accesses to memory are possible at a time
- In order to do so, it has to "steal" cycles from processor…***Cycle Stealing***
- Since, only one access to memory is possible at a time

3. Direct Memory Access

- The processor has to wait in the mean while
- This would slow down the processor
- But still more efficient than the previous two cases
- After the block transfer is completed, a single I/O interrupt will be generated by the DMA module.

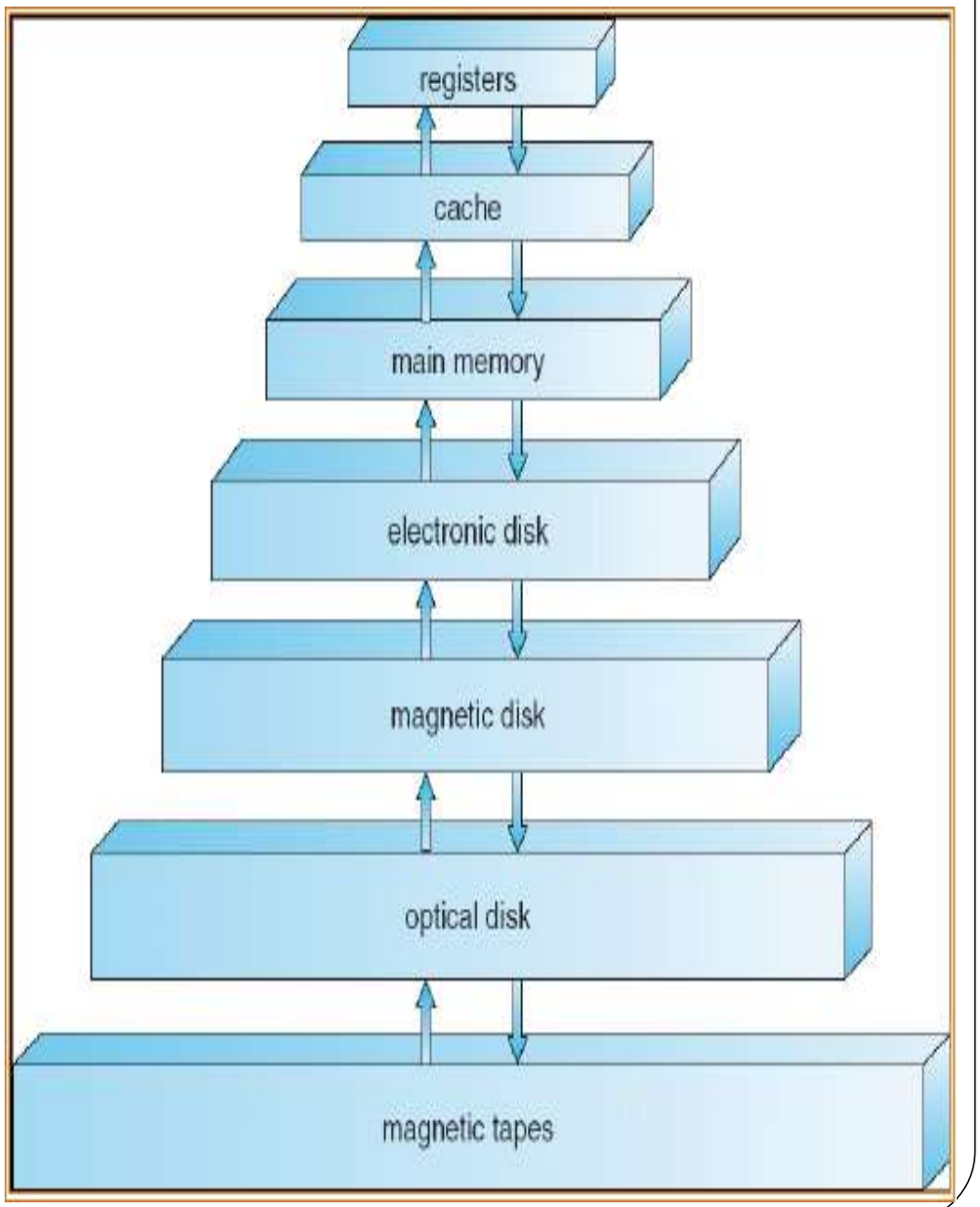
3. Direct Memory Access



Direct Memory Access

Storage Device Hierarchy

- Decreasing cost per bit
- Increasing capacity
- Increasing access time
- Decreasing frequency of access of the memory by the processor
  - Locality of reference
- Increasing size of the transfer unit



Storage hierarchy

- Storage systems organized in a hierarchy
  - Speed
  - Cost
  - Volatility
- Caching: copying information into faster storage system; main memory can be viewed as a last cache for secondary storage.

Performance of Various Levels of Storage

- Movement between levels of storage hierarchy can be explicit or implicit

Level	1	2	3	4
Name	registers	cache	main memory	disk storage
Typical size	< 1 KB	> 16 MB	> 16 GB	> 100 GB
Implementation technology	custom memory with multiple ports, CMOS	on-chip or off-chip CMOS SRAM	CMOS DRAM	magnetic disk
Access time (ns)	0.25 – 0.5	0.5 – 25	80 – 250	5,000.000
Bandwidth (MB/sec)	20,000 – 100,000	5000 – 10,000	1000 – 5000	20 – 150
Managed by	compiler	hardware	operating system	operating system
Backed by	cache	main memory	disk	CD or tape

Reference

- Chapter 1, Operating Systems by William Stallings
- http://www.cs.toronto.edu/~demke/469F.06/Lectures/Lecture6.pdf
- http://www.tldp.org/LDP/tlk/dd/interrupts.html
- http://en.wikibooks.org/wiki/Operating_System_Design/Processes/Interrupt
- http://www.slideshare.net/guest2e9811e/interrupts
- https://amser.org/g11619
- http://www.eecs.harvard.edu/~mdw/course/cs161/notes/osstructure.pdf
- http://www.sci.brooklyn.cuny.edu/~jniu/teaching/csc33200/files/0910-ComputerSystemOverview02.pdf
- http://siber.cankaya.edu.tr/OperatingSystems/ceng328/node87.html
- http://www.personal.kent.edu/~rmuhamma/OpSystems/Myos/processControl.htm
- http://www.tutorialspoint.com/operating_system/os_processes.htm
- http://wiki.answers.com/Q/Explain_process_control_block
- https://msdn.microsoft.com/en-us/library/windows/desktop/aa365683(v=vs.85).aspx