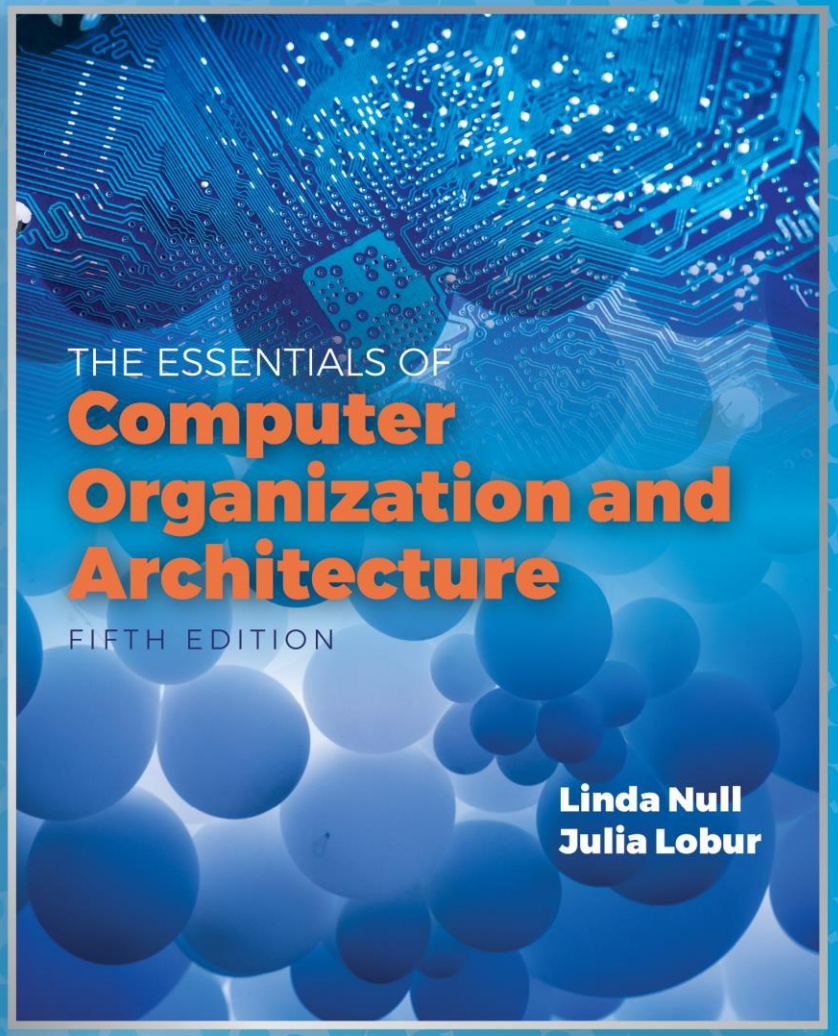
This is the first lecture of Chapter 7

Chapter 7

Input/Output Systems (A)



Objectives

- Understand how I/O systems work, including I/O methods and architectures.
- Become familiar with storage media, and the differences in their respective formats.
- Understand how RAID improves disk performance and reliability, and which RAID systems are most useful today.
- Be familiar with emerging data storage technologies and the barriers that remain to be overcome.

7.1 Introduction

- Data storage and retrieval is one of the primary functions of computer systems.
  - One could easily make the argument that computers are more useful to us as data storage and retrieval devices than they are as computational machines.
- All computers have I/O devices connected to them, and to achieve good performance I/O should be kept to a minimum!
- In studying I/O, we seek to understand the different types of I/O devices as well as how they work.

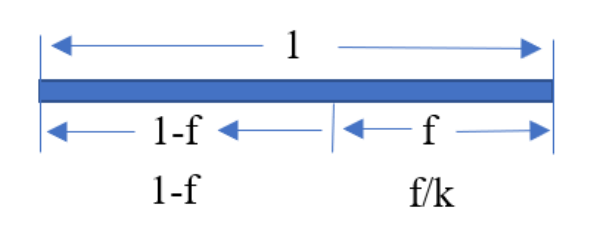
7.2 I/O and Performance

- Sluggish I/O throughput can have a ripple effect, dragging down overall system performance.
  - This is especially true when virtual memory is involved.
- The fastest processor in the world is of little use if it spends most of its time waiting for data.
- If we really understand what's happening in a computer system we can make the best possible use of its resources.

7.3 Amdahl's Law (1 of 3)

- The overall performance of a system is a result of the interaction of all of its components.
- System performance is most effectively improved when the performance of the most heavily used components is improved.
- This idea is quantified by Amdahl's Law:

$$S = \frac{1}{(1 - f) + (f/k)}$$



- *S* is the overall speedup;
- *f* is the fraction of work performed by a faster component; and
- *k* is the speedup of the faster component.

Speedup vs Increase

$$Speedup = \frac{Time_{old}}{Time_{new}}$$

$$Speedup = \frac{Speed_{new}}{Speed_{old}}$$

Speed increases 50% $\Longleftrightarrow$ $\frac{Speed_{new} - Speed_{old}}{Speed_{old}} = 0.5$

$\Updownarrow$

$$\frac{Speed_{new}}{Speed_{old}} - 1 = 0.5$$

Speedup is 1.5 $\Longleftrightarrow$ $S = \frac{Speed_{new}}{Speed_{old}} = 1.5$

7.3 Amdahl's Law (2 of 3)

$$S = \frac{1}{(1 - f) + (f/k)}$$

- Amdahl's Law gives us a handy way to estimate the performance improvement we can expect when we upgrade a system component.
- On a large system, suppose we can upgrade a CPU to make it 50% faster for $10,000 or upgrade its disk drives for $7,000 to make them 150% faster.
  - **K = 1.5 for CPU and K = 2.5 for Disk**
- Processes spend 70% of their time running in the CPU and 30% of their time waiting for disk service.
  - **f = 0.7 for CPU and f = 0.3 for Disk**
- An upgrade of which component would offer the greater benefit for the lesser cost?

7.3 Amdahl's Law (3 of 3)

$$S = \frac{1}{(1 - f) + (f/k)}$$

- The processor option offers a 30% speedup:

$$f = 0.70, k = 1.5, \text{so } S = \frac{1}{(1 - 0.7) + (0.7/5)} = 1.30$$

- And the disk drive option gives a 22% speedup:

$$f = 0.30, k = 2.5, \text{so } S = \frac{1}{(1 - 0.3) + (0.3/2.5)} \approx 1.22$$

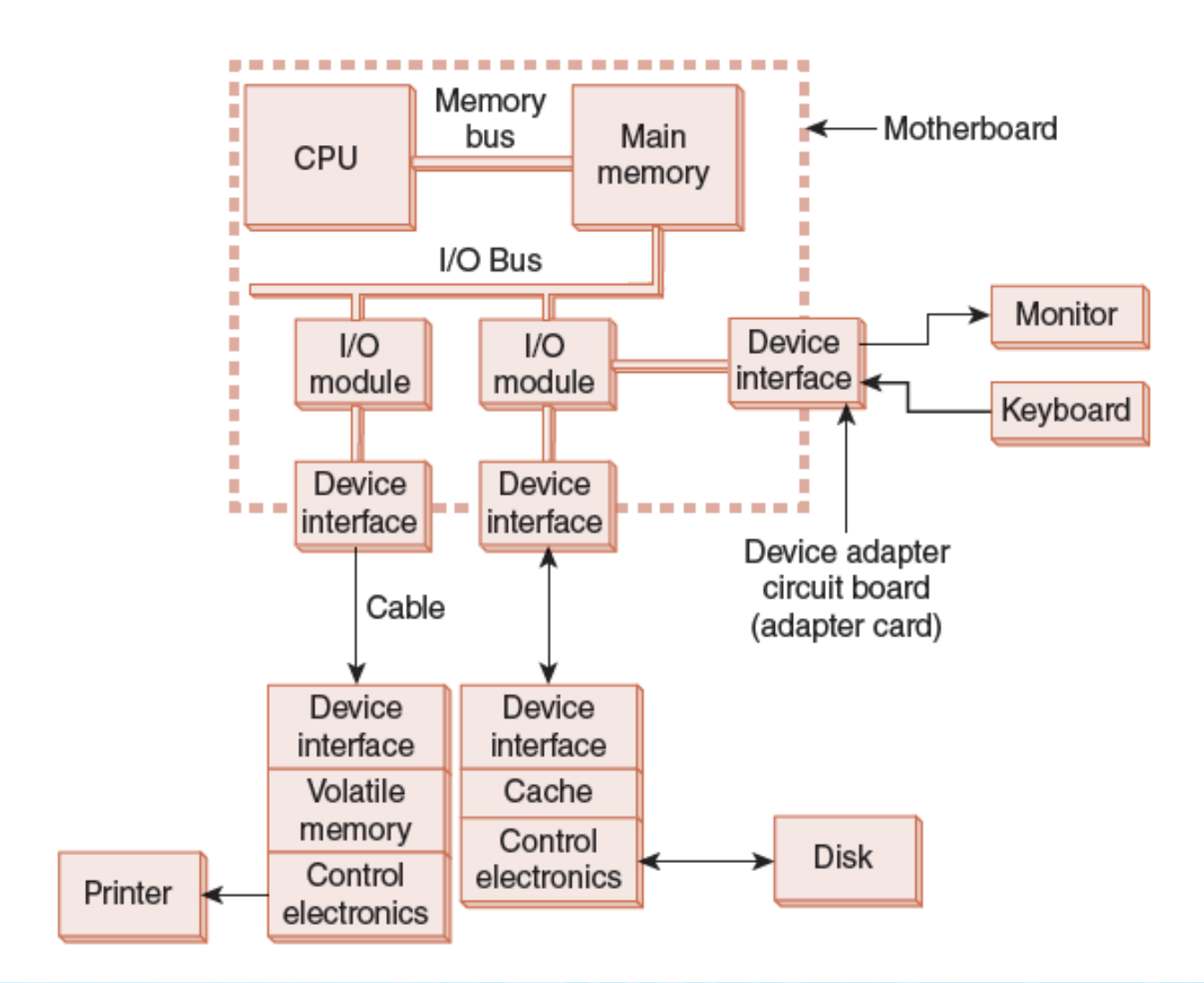
- Each 1% of improvement for the processor costs $333 (=10,000/30), and for the disk a 1% improvement costs $318 (=7,000/22).

Should price/performance be your only concern?

7.4 I/O Architectures (1 of 16)

- We define input/output as a subsystem of components that moves coded data between external devices and a host system.
- I/O subsystems include:
  - Blocks of main memory that are devoted to I/O functions.
  - Buses that move data into and out of the system.
  - Control modules in the host and in peripheral devices
  - Interfaces to external components such as keyboards and disks.
  - Cabling or communications links between the host system and its peripherals.

7.4 I/O Architectures (2 of 16)
A model I/O configuration



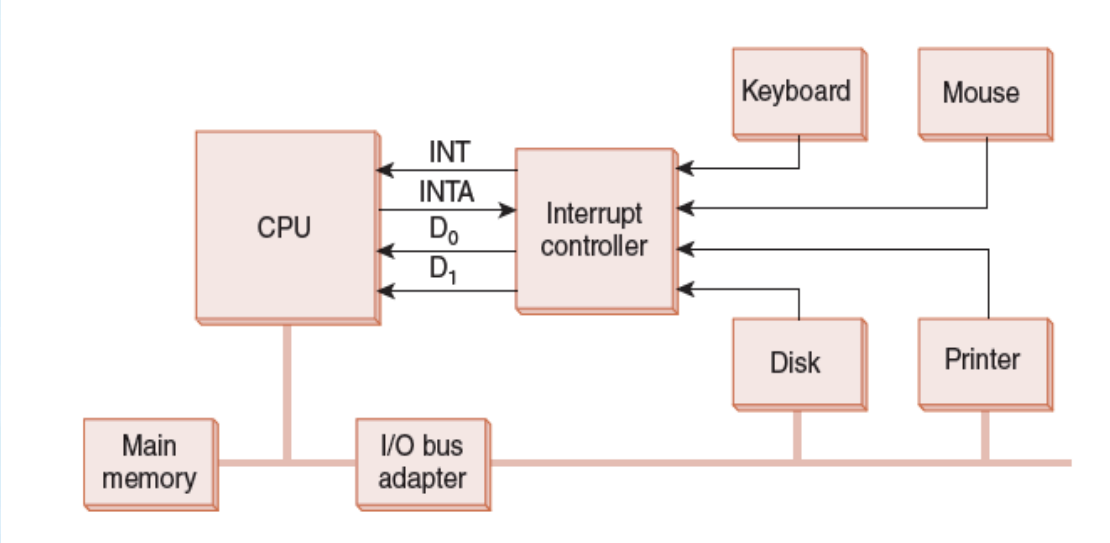
7.4 I/O Architectures (3 of 16)

- I/O can be controlled in five general ways.
  - *Programmed I/O* reserves a register for each I/O device. Each register is continually polled to detect data arrival.
  - *Interrupt-Driven I/O* allows the CPU to do other things until I/O is requested.
  - *Memory-Mapped I/O* shares memory address space between I/O devices and program memory.
  - *Direct Memory Access* (DMA) offloads I/O processing to a special-purpose chip that takes care of the details.
  - Channel I/O uses dedicated I/O processors.

7.4 I/O Architectures (4 of 16)

- This is an idealized I/O subsystem that uses interrupts.
- Each device connects its interrupt line to the interrupt controller.

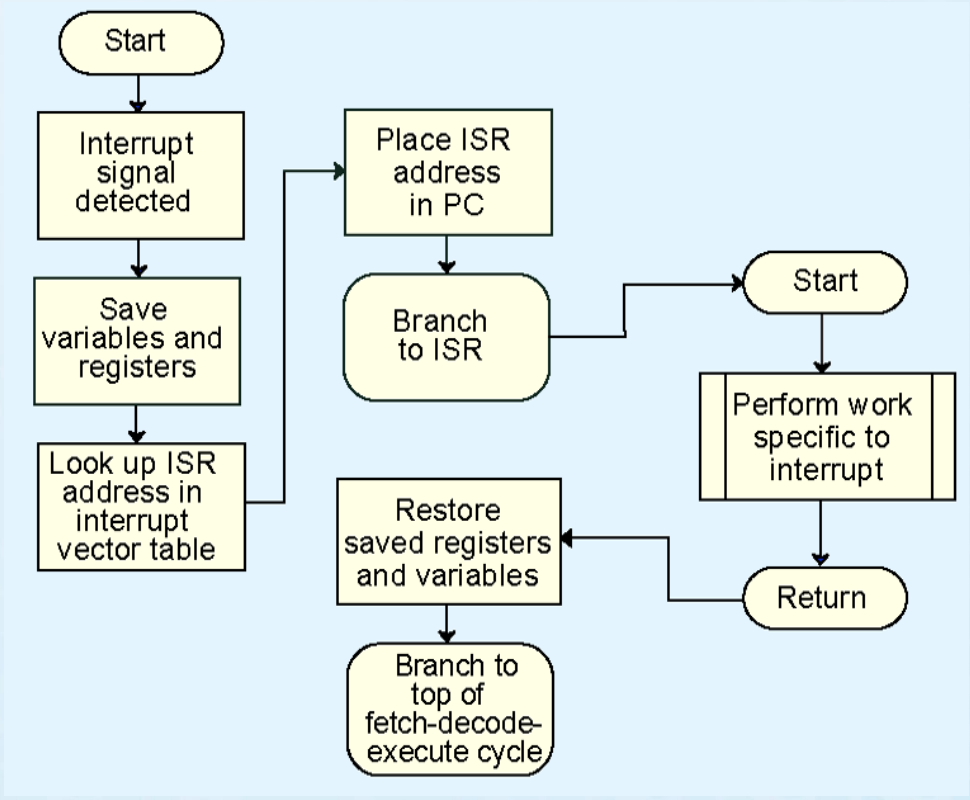
The controller signals the CPU when any of the interrupt lines are asserted.

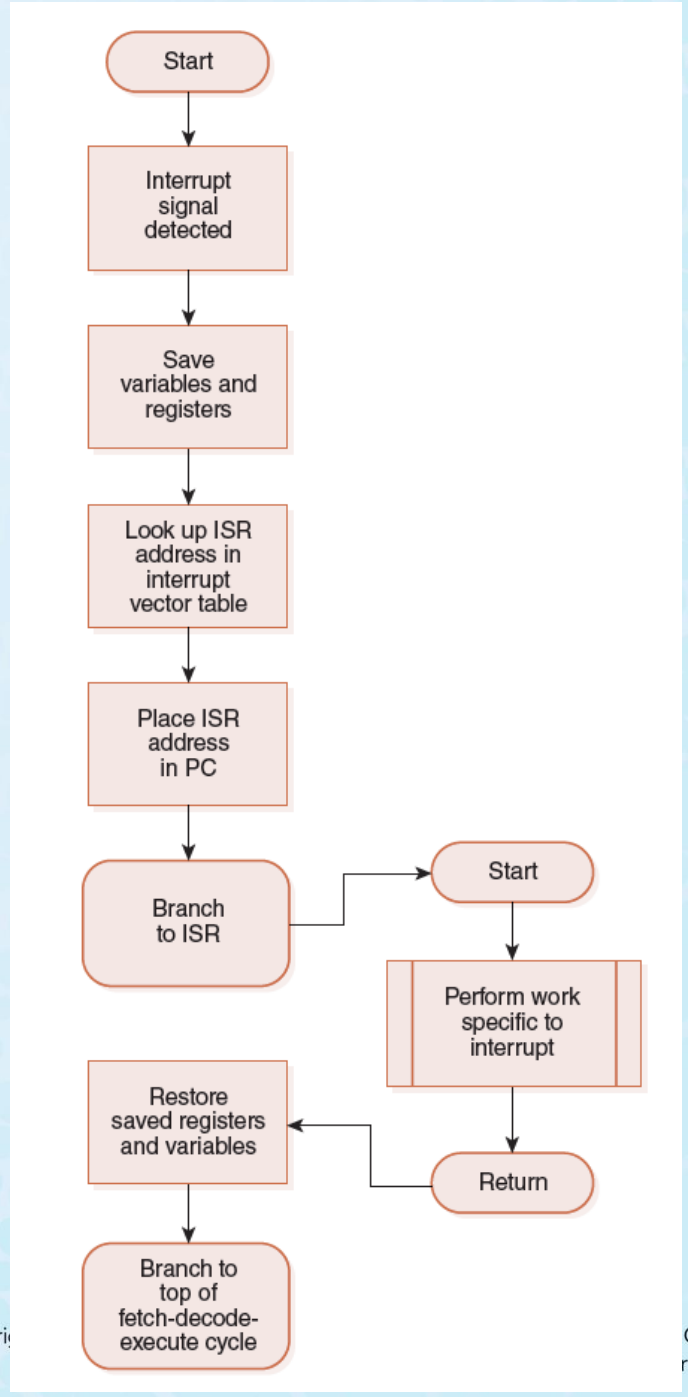


7.4 I/O Architectures (5 of 16)

- Recall from Chapter 4 that in a system that uses interrupts, the status of the interrupt signal is checked at the top of the fetch-decode-execute cycle.
- The particular code that is executed whenever an interrupt occurs is determined by a set of addresses called ***interrupt vectors*** that are stored in low memory.
- The system state is saved before the interrupt service routine is executed and is restored afterward.
- We provide a flowchart on the next slide.

7.4 I/O Architectures (6 of 16)



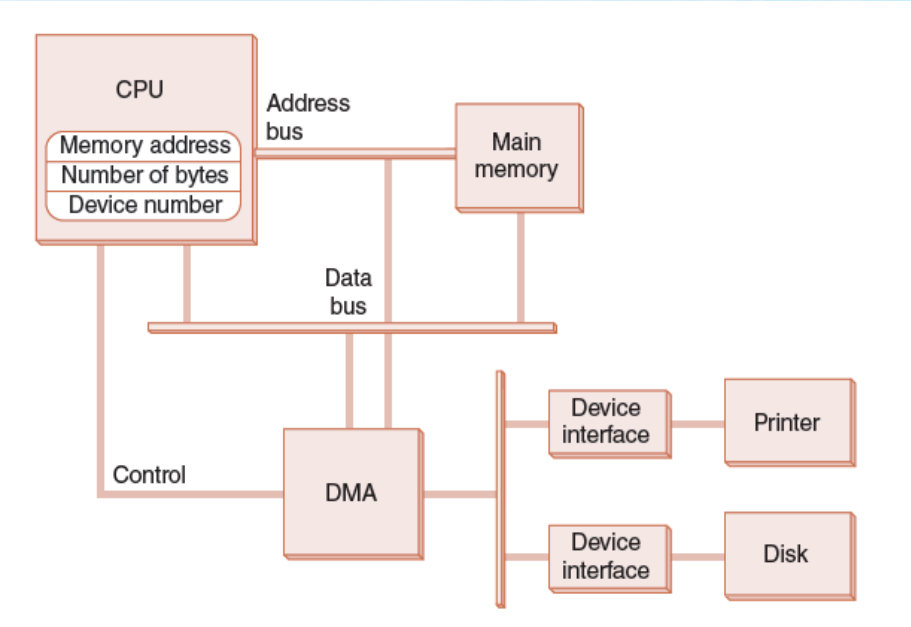


7.4 I/O Architectures (7 of 16)

- In memory-mapped I/O devices and main memory share the same address space.
  - Each I/O device has its own reserved block of memory.
  - Memory-mapped I/O therefore looks just like a memory access from the point of view of the CPU.
  - Thus the same instructions to move data to and from both I/O and memory, greatly simplifying system design.
- In small systems the low-level details of the data transfers are offloaded to the I/O controllers built into the I/O devices.

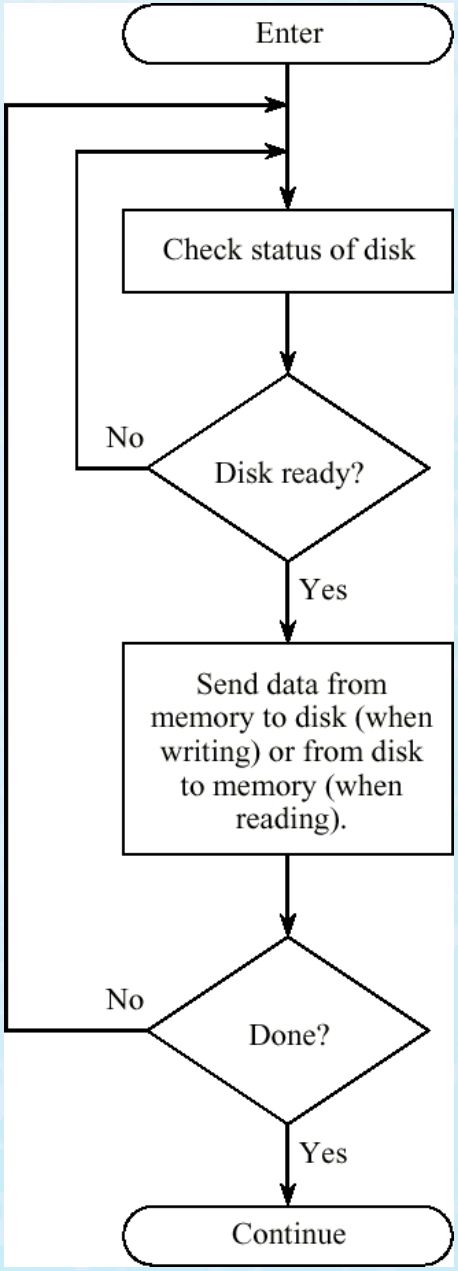
7.4 I/O Architectures (8 of 16)

- This is a DMA configuration.
- Notice that the DMA and the CPU share the bus.
- The DMA runs at a higher priority and steals memory cycles from the CPU.

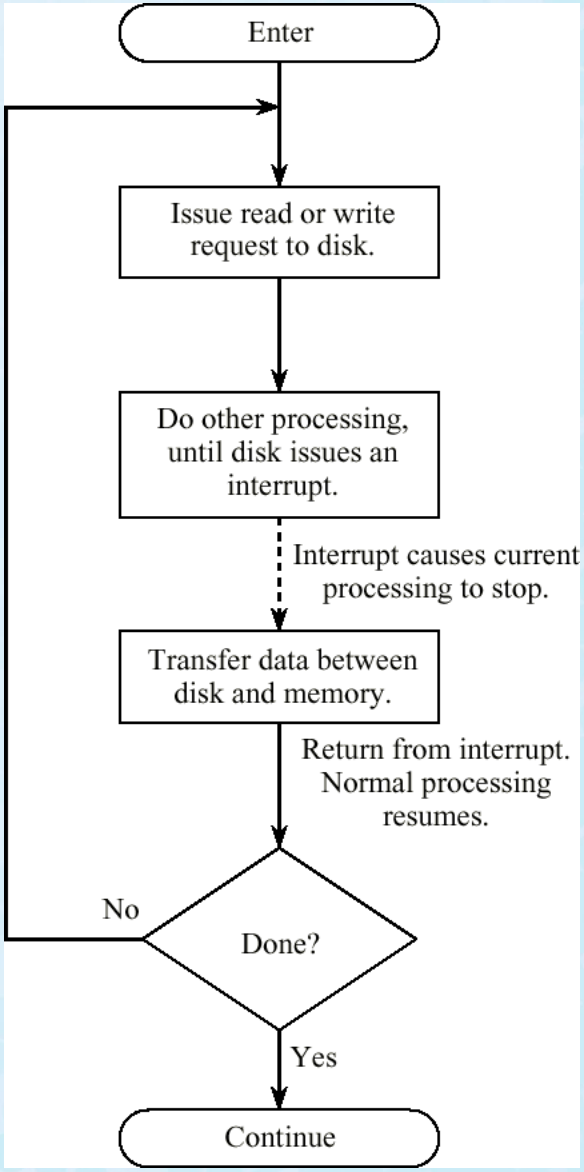


Programmed I/O Flowchart for a Disk Transfer

If the disk is not ready for read or write then the process loops back and checks the status continuously until the disk is ready. This is referred as a busy-wait.



Interrupt Driven I/O Flowchart for a Disk Transfer



DMA Flowchart for a Disk Transfer