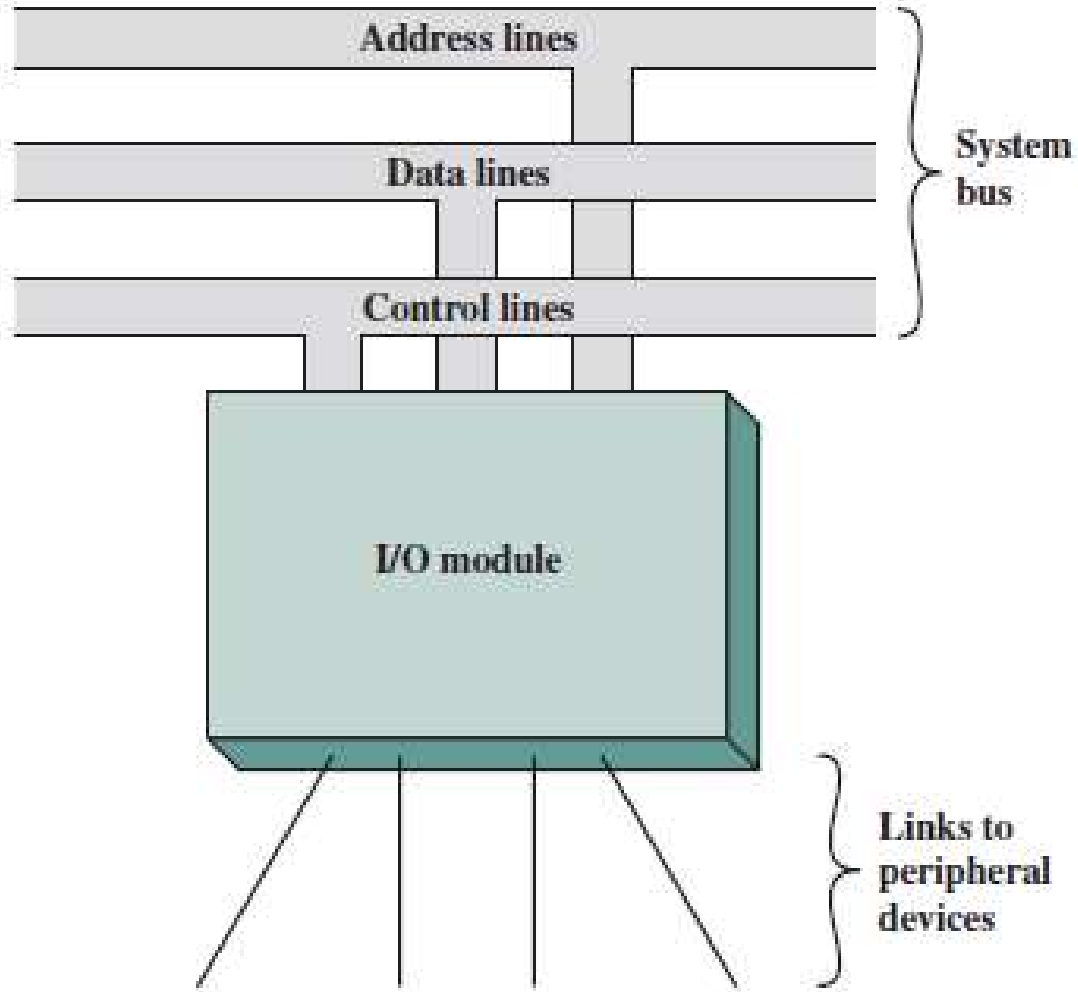
Input/Output and Operating System Support

Lecture 7 & 8

External Devices

- External devices provide a means of exchanging data between the external environment and the computer.
- An external device attaches to the computer by a link to an I/O module.
- The link is used to exchange control, status, and data between the I/O module and the external device.

External Devices



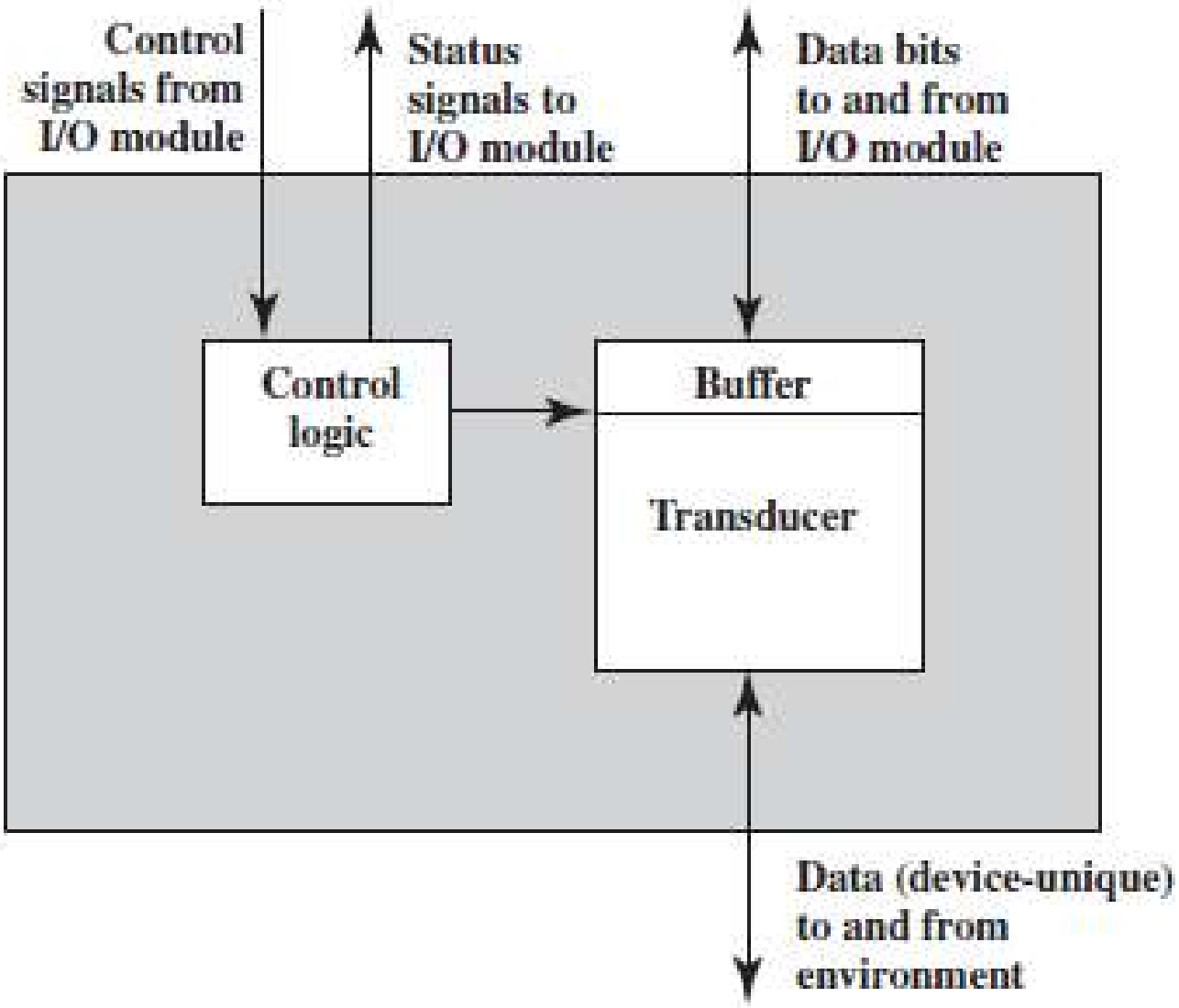
External Devices

- An external device connected to an I/O module is often referred to as a peripheral device or, simply, a peripheral.
- **Human readable**: Suitable for communicating with the computer user;
- **Machine readable**: Suitable for communicating with equipment;
- **Communication**: Suitable for communicating with remote devices.

External Devices

- Examples of human-readable devices are video display terminals (VDTs) and printers.
- Examples of machine-readable devices are magnetic disk and tape systems, and sensors and actuators, such as are used in a robotics application.
- Communication devices allow a computer to exchange data with a remote device, which may be a human readable device, such as a terminal, a machine-readable device, or even another computer.

External Devices



I/O Modules

- The major functions of an I/O module:
  - Control and timing
  - Processor communication
  - Device communication
  - Data buffering
  - Error detection

I/O Modules

- During any period of time, the processor may communicate with one or more external devices in unpredictable patterns.
- The internal resources, such as main memory and the system bus, must be shared among a number of activities, including data I/O.
- For example, the control of the transfer of data from an external device to the processor might involve the following sequence of steps:

I/O Modules

- The processor interrogates the I/O module to check the status of the attached device.
- The I/O module returns the device status.
- If the device is operational and ready to transmit, the processor requests the transfer of data, by means of a command to the I/O module.
- The I/O module obtains a unit of data (e.g., 8 or 16 bits) from the external device.
- The data are transferred from the I/O module to the processor.

Programmed I/O

- Three techniques are possible for I/O operations.
- With *programmed I/O*, data are exchanged between the processor and the I/O module.
- The processor executes a program that gives it direct control of the I/O operation, including sensing device status, sending a read or write command, and transferring the data.

Programmed I/O

- When the processor issues a command to the I/O module, it must wait until the I/O operation is complete.
- If the processor is faster than the I/O module, this is waste of processor time.
- With *interrupt-driven I/O*, the processor issues an I/O command, continues to execute other instructions, and is interrupted by the I/O module when the latter has completed its work.

Programmed I/O

- With both programmed and interrupt I/O, the processor is responsible for extracting data from main memory for output and storing data in main memory for input.

Interrupt- Driven I/O

- The problem with programmed I/O is that the processor has to wait a long time for the I/O module of concern to be ready for either reception or transmission of data.
- The processor, while waiting, must repeatedly interrogate the status of the I/O module.
- As a result, the level of the performance of the entire system is severely degraded.

Interrupt- Driven I/O

- An alternative is for the processor to issue an I/O command to a module and then go on to do some other useful work.
- The I/O module will then interrupt the processor to request service when it is ready to exchange data with the processor.
- The processor then executes the data transfer, as before, and then resumes its former processing.

Interrupt- Driven I/O

- **Design Issues**
- Two design issues arise in implementing interrupt I/O.
- First, because there will almost invariably be multiple I/O modules, how does the processor determine which *device* issued the interrupt?
- And second, if multiple interrupts have occurred, how does the processor decide which one to process?

Interrupt- Driven I/O

- Let us consider device identification first. Four general categories of techniques are in common use:
  - Multiple interrupt lines
  - Software poll
  - Daisy chain (hardware poll, vectored)
  - Bus arbitration (vectored)

Direct Memory Access

- Interrupt-driven I/O, though more efficient than simple programmed I/O, still requires the active intervention of the processor to transfer data between memory and an I/O module, and any data transfer must traverse a path through the processor.
- Thus, both these forms of I/O suffer from two inherent drawbacks:

Direct Memory Access

- The I/O transfer rate is limited by the speed with which the processor can test and service a device.
- The processor is tied up in managing an I/O transfer; a number of instructions must be executed for each I/O transfer.

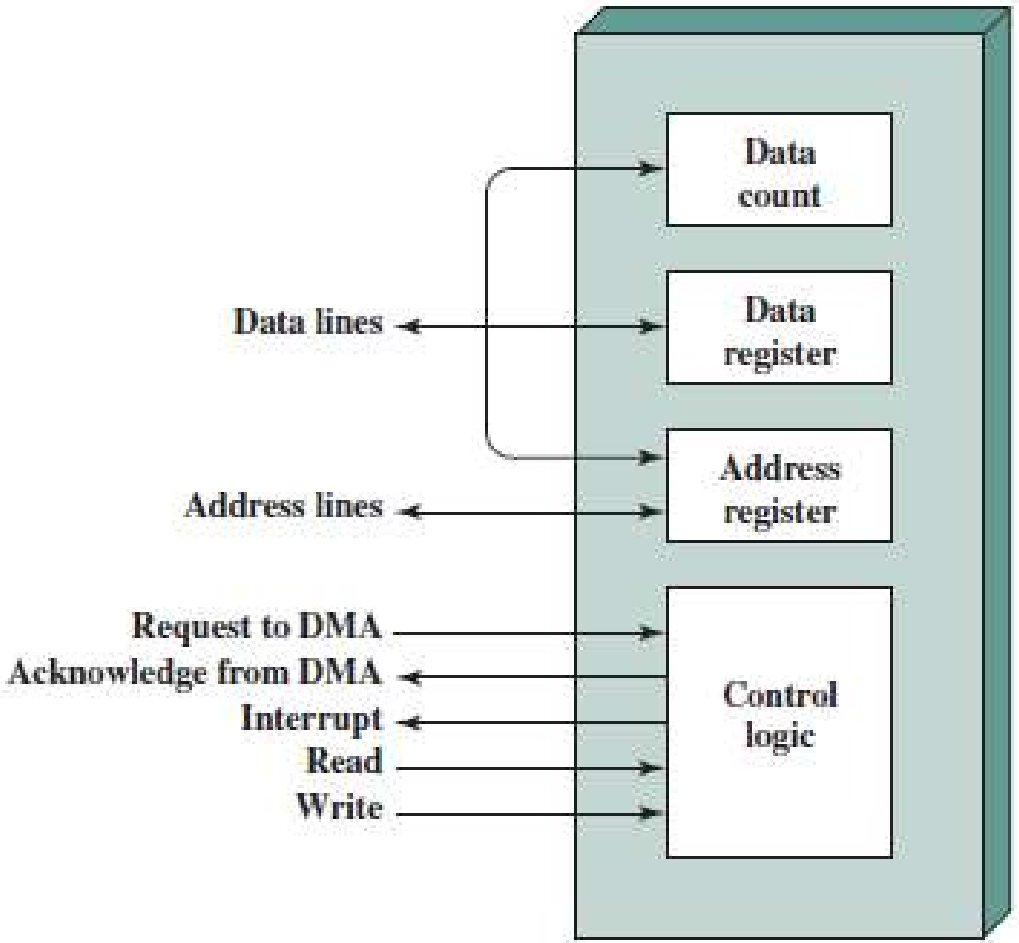
Direct Memory Access

- DMA involves an additional module on the system bus.
- The DMA module is capable of mimicking the processor and, indeed, of taking over control of the system from the processor.
- It needs to do this to transfer data to and from memory over the system bus.

Direct Memory Access

- For this purpose, the DMA module must use the bus only when the processor does not need it, or it must force the processor to suspend operation temporarily.
- The latter technique is more common and is referred to as *cycle stealing*, because the DMA module in effect steals a bus cycle.

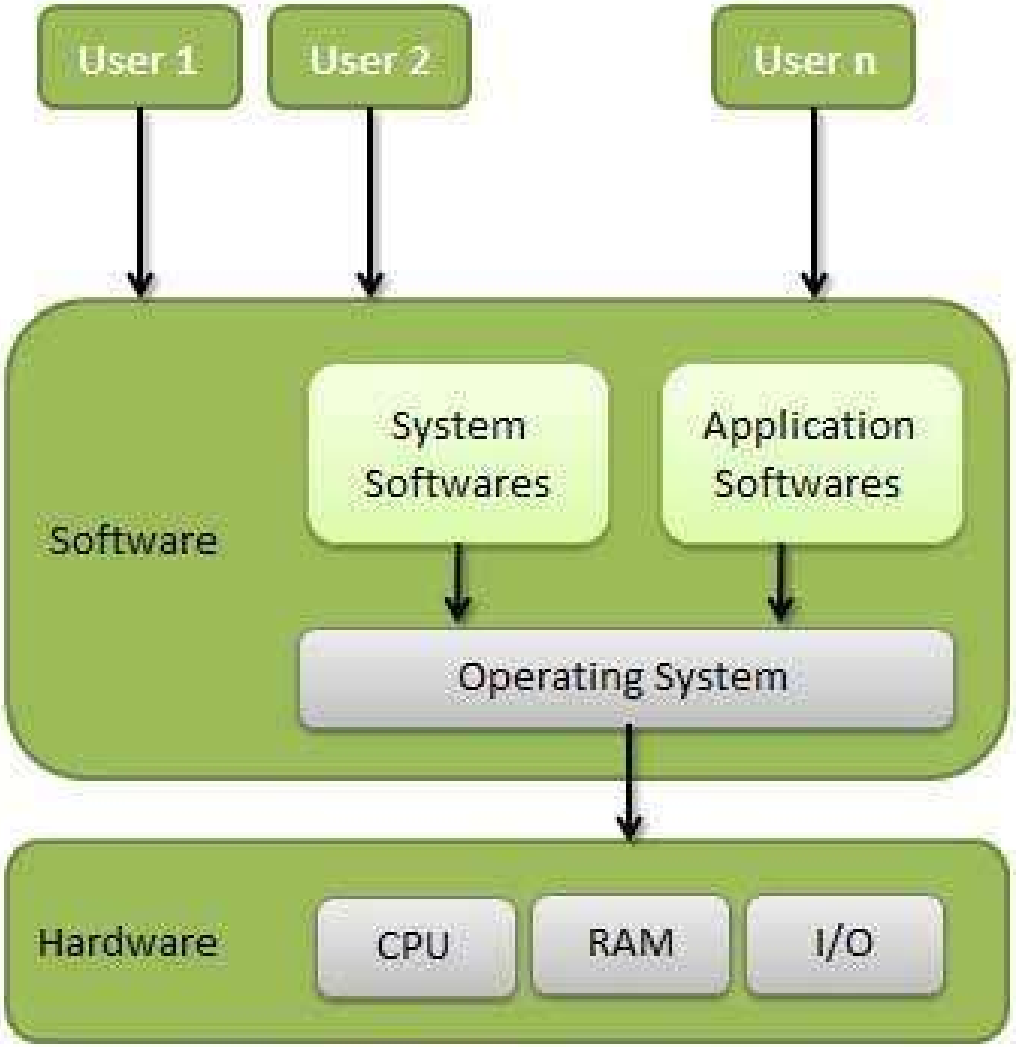
Direct Memory Access



Operating System Support

- An operating system is a program that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs.
- It is a software which performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers.
- Is a collection of software that manages computer hardware resources and provides common services for computer programs.
- The operating system is a vital component of the system software in a computer system.

Operating System Support



Examples of Operating Systems

- Windows (7, 8, 10, Sever)
- Mac OS
- Linux
- Solaris
- Android
- Chrome OS
- iOS
- Ubuntu, etc

Functions of OS

- Memory Management
- Processor Management
- Device Management
- File Management
- Security
- Control over system performance
- Job accounting
- Error detecting aids
- Coordination between other software and users

Advantages of OS

- Easy to use
- User friendly
- Intermediate between hardwares and softwares of the system
- No need to know any technical languages

Scheduling Algorithms

- A Process Scheduler schedules different processes to be assigned to the CPU based on particular scheduling algorithms . There are six popular process scheduling algorithms.

– First-Come, First-Served (FCFS) Scheduling
– Shortest-Job-Next (SJN) Scheduling
– Priority Scheduling
– Shortest Remaining Time
– Round Robin(RR) Scheduling
– Multiple-Level Queues Scheduling

Non-preemptive or Preemptive

- Non-preemptive algorithms are designed so that once a process enters the running state, it cannot be preempted until it completes its allotted time, whereas the preemptive scheduling is based on priority where a scheduler may preempt a low priority running process anytime when a high priority process enters into a ready state

First Come First Serve (FCFS)

- Jobs are executed on first come, first serve basis.
- It is a non-preemptive, pre-emptive scheduling algorithm.
- Its implementation is based on FIFO queue.
- Poor in performance as average wait time is high
- Easy to understand and implement

First Come First Serve (FCFS)

Process	Arrival Time	Execute Time	Service Time
P0	0	5	0
P1	1	3	5
P2	2	8	8
P3	3	6	16

First Come First Serve (FCFS)

Wait time of each process is as follows −

Process	Wait Time : Service Time - Arrival Time
P0	0 - 0 = 0
P1	5 - 1 = 4
P2	8 - 2 = 6
P3	16 - 3 = 13

Average Wait Time: (0+4+6+13) / 4 = 5.75

Shortest Job Next (SJN)

- This is also known as shortest job first, or SJF
- This is a non-preemptive, pre-emptive scheduling algorithm.
- Best approach to minimize waiting time.

Shortest Job Next (SJN)

Given: Table of processes, and their Arrival time, Execution time

Process	Arrival Time	Execution Time	Service Time
P0	0	5	0
P1	1	3	5
P2	2	8	14
P3	3	6	8

Shortest Job Next (SJN)

Waiting time of each process is as follows −

Process	Waiting Time
P0	0 - 0 = 0
P1	5 - 1 = 4
P2	14 - 2 = 12
P3	8 - 3 = 5

Average Wait Time: (0 + 4 + 12 + 5)/4 = 21 / 4 = 5.25

Priority Based Scheduling

- Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems.
- Each process is assigned a priority. Process with highest priority is to be executed first and so on.

Priority Based Scheduling

Process	Arrival Time	Execution Time	Priority	Service Time
P0	0	5	1	0
P1	1	3	2	11
P2	2	8	1	14
P3	3	6	3	5

Priority Based Scheduling

Waiting time of each process is as follows −

Process	Waiting Time
P0	0 - 0 = 0
P1	11 - 1 = 10
P2	14 - 2 = 12
P3	5 - 3 = 2

Average Wait Time: (0 + 10 + 12 + 2)/4 = 24 / 4 = 6