INPUT-OUTPUT ORGANIZATION

- **Peripheral Devices**
- **Input-Output Interface**
- **Modes of Transfer**
- **Priority Interrupt**
- **Direct Memory Access**
- **Input-Output Processor**

PERIPHERAL DEVICES

- The input –output subsystem of a computer referred as I/O, provides an efficient mode of communication b/w central system and outside environment.
- A computer serves no useful purpose without the ability to receive information from the outside world and to transmit results in meaningful form.
- Devices that are designed to read information into or out of the memory unit upon command from CPU are considered to be the part of total computer system.
- Hence Input or Output devices attached to the computer are called PERIPHERALS.

PERIPHERAL DEVICES

Input Devices

- Keyboard
- Optical input devices
  - Card Reader
  - Paper Tape Reader
  - Bar code reader
  - Digitizer
  - Optical Mark Reader
- Magnetic Input Devices
  - Magnetic Stripe Reader
- Screen Input Devices
  - Touch Screen
  - Light Pen
  - Mouse
- Analog Input Devices

Output Devices

- Card Puncher,
- Paper Tape Puncher
- CRT
- Printer (Impact, Ink Jet, Laser, Dot Matrix)
- Plotter

General Description of I/O

Wide variety of peripherals

- Delivering different amounts of data
- At different speeds
- In different formats (bit depth, etc.)

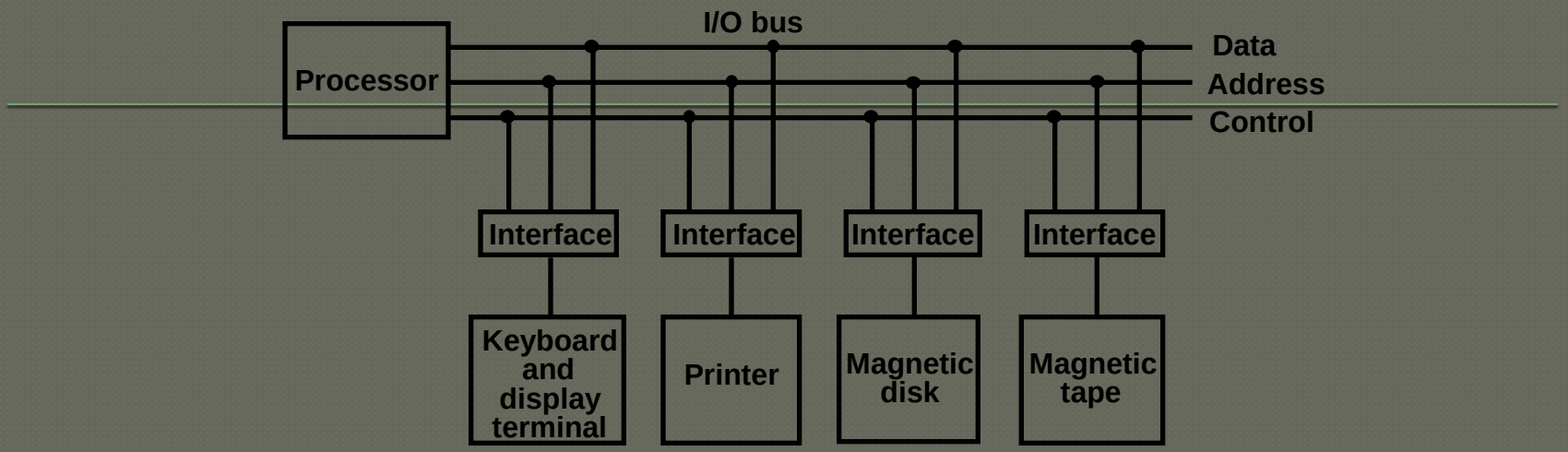
INPUT/OUTPUT INTERFACE

- Provides a method for transferring information between internal storage (such as memory and CPU registers) and external I/O devices
- Resolves the *differences* between the computer and peripheral devices
  - Peripherals - Electromechanical Devices
  - CPU or Memory - Electronic Device
  - Data Transfer Rate
    - Peripherals - Usually slower
    - CPU or Memory - Usually faster than peripherals
      - Some kinds of Synchronization mechanism may be needed
  - Unit of Information
    - Peripherals – Byte, Block, …
    - CPU or Memory – Word
  - Data representations may differ

Closing the Gap

To resolve these differences, computer system include special hardware components between CPU and peripherals to supervise and synchronize all input and output transfers. These components are called INTERFACE UNIT.

I/O BUS AND INTERFACE MODULES



Each peripheral has an interface module associated with it

Interface

- **Decodes the device address (device code)**
- **Decodes the commands (operation)**
- **Provides signals for the peripheral controller**
- **Synchronizes the data flow and supervises the transfer rate between peripheral and CPU or Memory**

Typical I/O instruction

Op. code	Device address	Function code

(Command)

INPUT OUTPUT COMMANDS

- CONTROL is issued to activate the peripheral and to inform it what to do.
- STATUS is used to test various conditions in the interface and the peripherals.
- DATA OUTPUT causes the interface to respond by transferring data from bus into one of its registers.
- DATA INPUT is opposite of data output.

I/O BUS AND MEMORY BUS

Functions of Buses

*** *MEMORY BUS* is for information transfers between CPU and the MM**

•*I/O BUS* is for information transfers between CPU and I/O devices through their I/O interface

Physical Organizations

•Many computers use a common single bus system for both memory and I/O interface units

- Use one common bus but separate control lines for each function
- Use one common bus with common control lines for both functions

*** Some computer systems use two separate buses, one to communicate with memory and the other with I/O interfaces**

I/O Bus

- Communication between CPU and all interface units is via a common I/O Bus
- An interface connected to a peripheral device may have a number of *data registers* , a *control register*, and a *status register*
- A command is passed to the peripheral by sending to the appropriate interface register
- Function code and sense lines are not needed (Transfer of data, control, and status information is always via the common I/O Bus)

ISOLATED vs MEMORY MAPPED I/O

Isolated I/O

- Separate I/O read/write control lines in addition to memory read/write control lines
- Separate (isolated) memory and I/O address spaces
- Distinct input and output instructions

Memory-mapped I/O

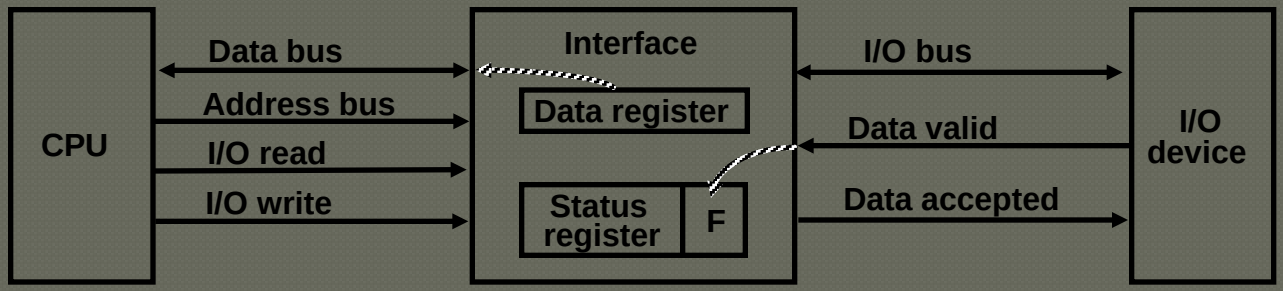
- A single set of read/write control lines
  (no distinction between memory and I/O transfer)
- Memory and I/O addresses share the common address space
  -> reduces memory address range available
- No specific input or output instruction
  -> The same memory reference instructions can be used for I/O transfers
- Considerable flexibility in handling I/O operations

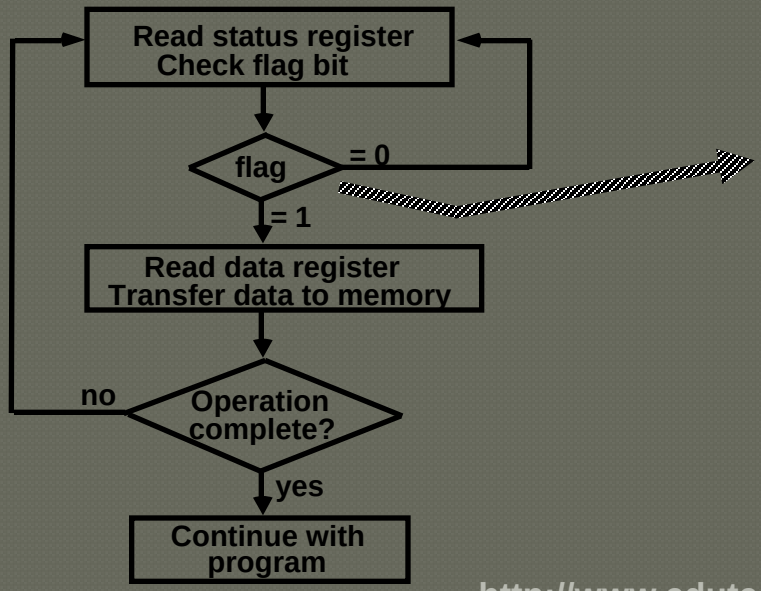
MODES OF TRANSFER - PROGRAM-CONTROLLED I/O -

3 different Data Transfer Modes between the central computer(CPU or Memory) and peripherals;

Program-Controlled I/O
Interrupt-Initiated I/O
Direct Memory Access (DMA)

Program-Controlled I/O(Input Dev to CPU)





Polling or Status Checking

- **Continuous CPU involvement**
- **CPU slowed down to I/O speed**
- **Simple**
- **Least hardware**

MODES OF TRANSFER - INTERRUPT INITIATED I/O & DMA

Interrupt Initiated I/O

- Polling takes valuable CPU time
- Open communication only when some data has to be passed -> *Interrupt*.
- I/O interface, instead of the CPU, monitors the I/O device
- When the interface determines that the I/O device is ready for data transfer, it generates an *Interrupt Request* to the CPU
- Upon detecting an interrupt, CPU stops momentarily the task it is doing, branches to the service routine to process the data transfer, and then returns to the task it was performing

DMA (Direct Memory Access)

- Large blocks of data transferred at a high speed to or from high speed devices, magnetic drums, disks, tapes, etc.
- DMA controller
  Interface that provides I/O transfer of data directly to and from the memory and the I/O device
- CPU initializes the DMA controller by sending a memory address and the number of words to be transferred
- Actual transfer of data is done directly between the device and memory through DMA controller
  -> Freeing CPU for other tasks

PRIORITY INTERRUPT

Priority

- Determines which interrupt is to be served first when two or more requests are made simultaneously
- Also determines which interrupts are permitted to interrupt the computer while another is being serviced
- Higher priority interrupts can make requests while servicing a lower priority interrupt

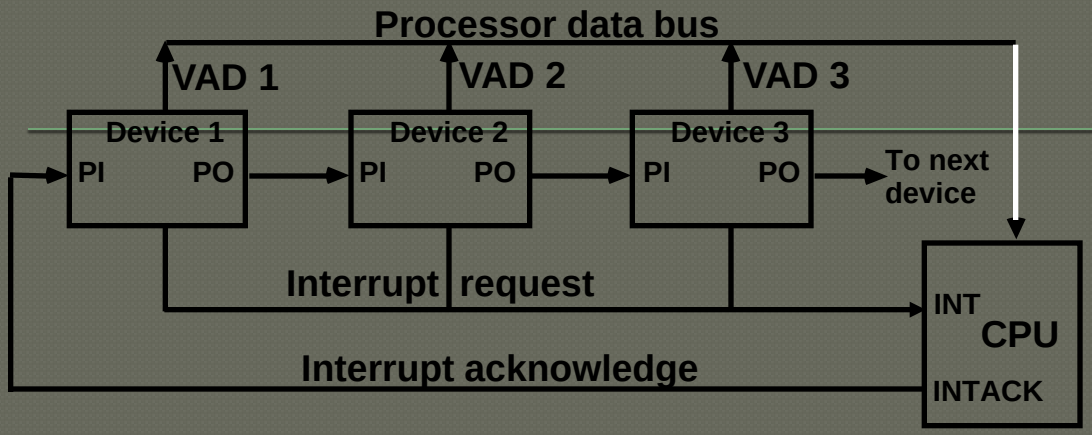
Priority Interrupt by Software(Polling)

- Priority is established by the order of polling the devices(interrupt sources)
- Flexible since it is established by software
- Low cost since it needs a very little hardware
- Very slow

Priority Interrupt by Hardware

- Require a priority interrupt manager which accepts all the interrupt requests to determine the highest priority request
- Fast since identification of the highest priority interrupt request is identified by the hardware
- Fast since each interrupt source has its own interrupt vector to access directly to its own service routine

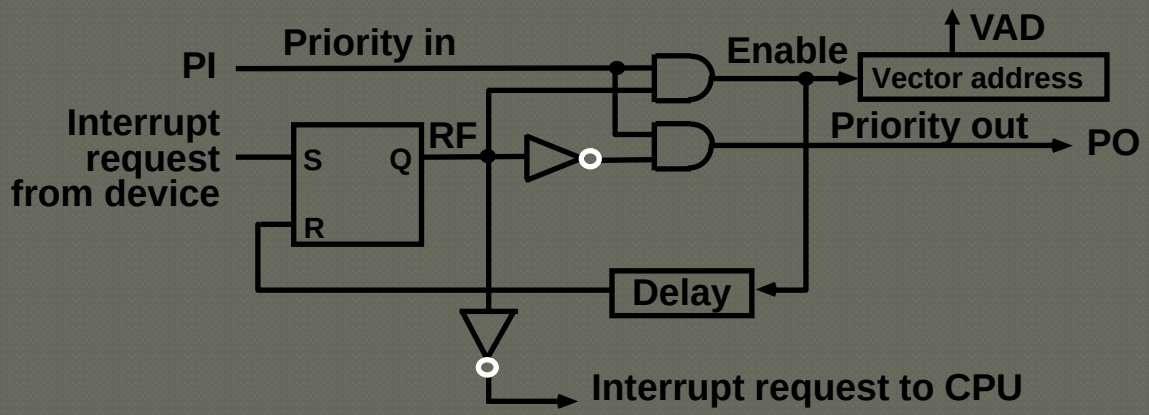
HARDWARE PRIORITY INTERRUPT - DAISY-CHAIN -



* Serial hardware priority function
* Interrupt Request Line
  - Single common line
* Interrupt Acknowledge Line
  - Daisy-Chain

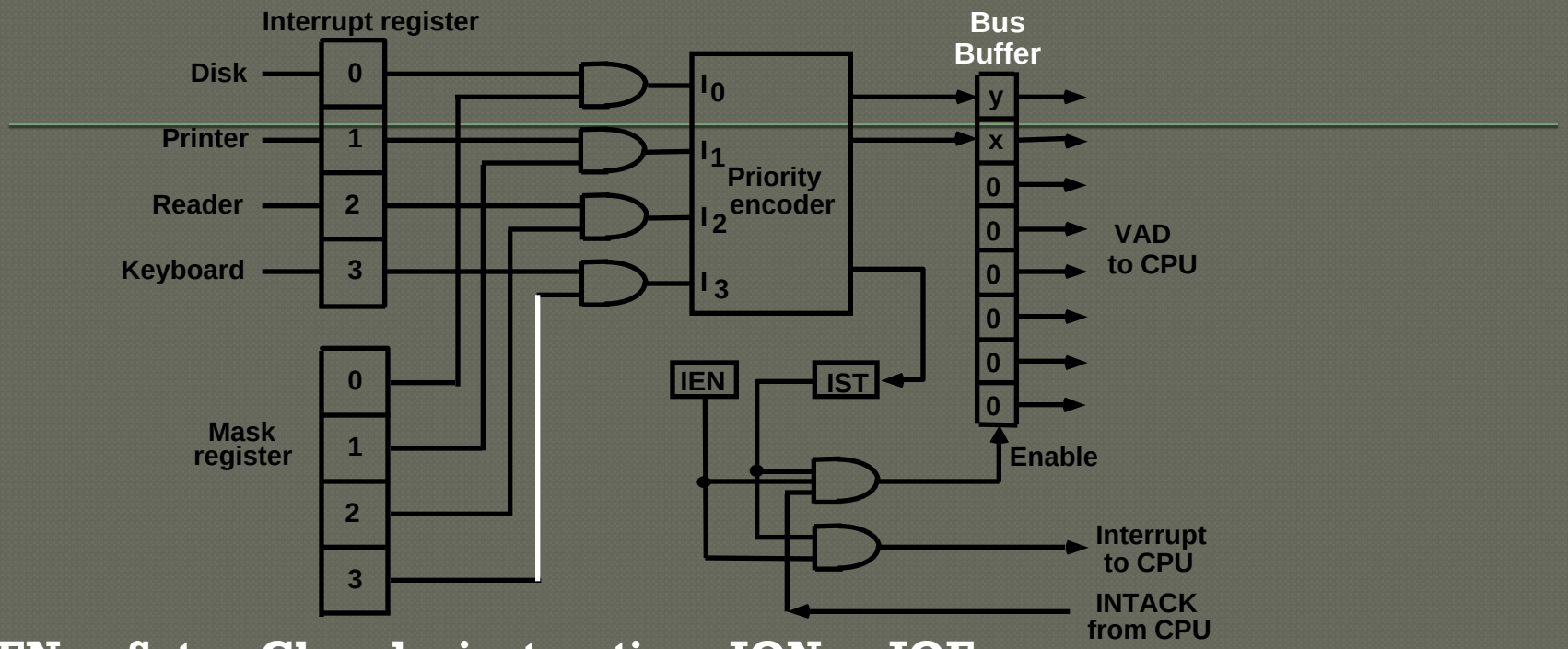
Interrupt Request from any device(>=1)
-> CPU responds by INTACK <- 1
-> Any device receives signal(INTACK) 1 at PI puts the VAD on the bus
Among interrupt requesting devices the only device which is physically closes to CPU gets INTACK=1, and it blocks INTACK to propagate to the next device

One stage of the daisy chain priority arrangement



PI	RF	PO	Enable
0	0	0	0
0	1	0	0
1	0	1	0
1	1	1	1

PARALLEL PRIORITY INTERRUPT



IEN: Set or Clear by instructions ION or IOF

IST: Represents an unmasked interrupt has occurred. INTACK enables tristate Bus Buffer to load VAD generated by the Priority Logic

Interrupt Register:

- Each bit is associated with an Interrupt Request from different Interrupt Source - different priority level
- Each bit can be cleared by a program instruction

Mask Register:

- Mask Register is associated with Interrupt Register
- Each bit can be set or cleared by an Instruction

INTERRUPT PRIORITY ENCODER

Determines the highest priority interrupt when more than one interrupts take place

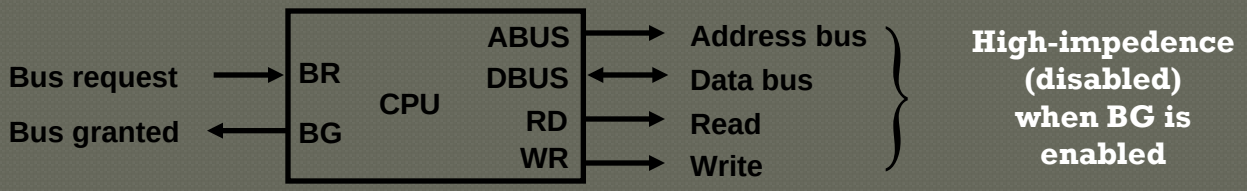
Priority Encoder Truth table

Inputs				Outputs			
I_0	I_1	I_2	I_3	x	y	IST	Boolean functions
1	d	d	d	0	0	1	
0	1	d	d	0	1	1	
0	0	1	d	1	0	1	$x = I_0' \; I_1'$
0	0	0	1	1	1	1	$y = I_0' \; I_1 + I_0' \; I_2'$
0	0	0	0	d	d	0	$(IST) = I_0 + I_1 + I_2 + I_3$

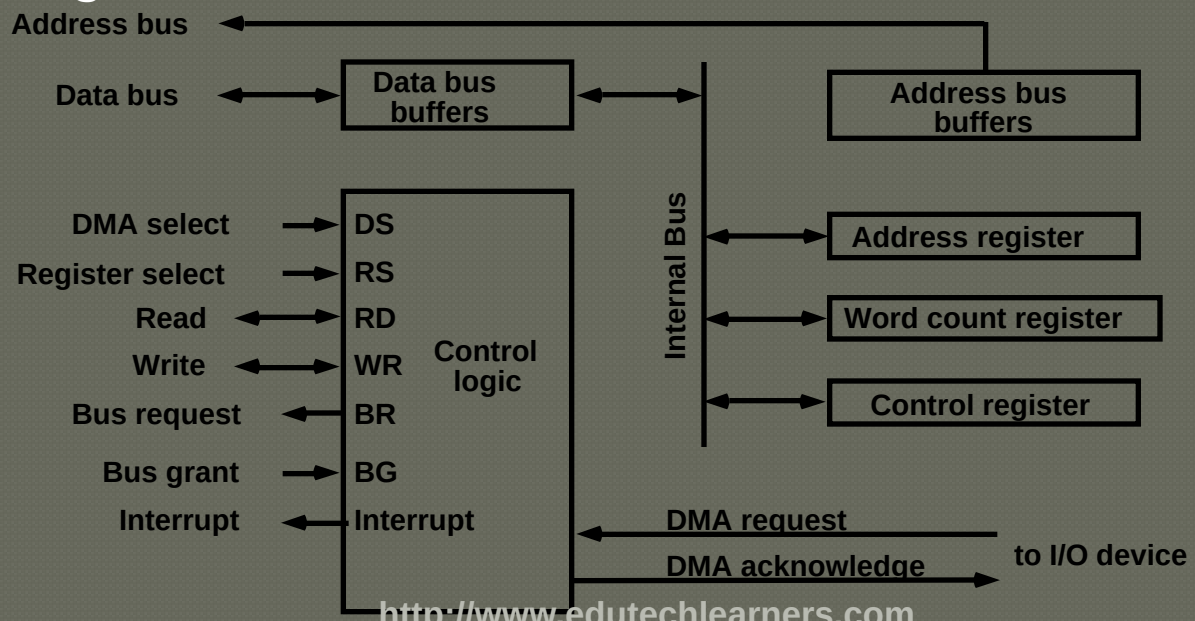
DIRECT MEMORY ACCESS

* Block of data transfer from high speed devices, Drum, Disk, Tape
* DMA controller - Interface which allows I/O transfer directly between Memory and Device, freeing CPU for other tasks
* CPU initializes DMA Controller by sending memory address and the block size(number of words)

CPU bus signals for DMA transfer



Block diagram of DMA controller



DMA I/O OPERATION

Starting an I/O

- CPU executes instruction to

Load Memory Address Register
Load Word Counter
Load Function(Read or Write) to be performed
Issue a GO command

Upon receiving a GO Command DMA performs I/O operation as follows independently from CPU

Input

[1] Input Device <- R (Read control signal)
[2] Buffer(DMA Controller) <- Input Byte; and assembles the byte into a word until word is full
[4] M <- memory address, W(Write control signal)
[5] Address Reg <- Address Reg +1; WC(Word Counter) <- WC - 1
[6] If WC = 0, then Interrupt to acknowledge done, else go to [1]

Output

[1] M <- M Address, R
M Address R <- M Address R + 1, WC <- WC - 1
[2] Disassemble the word
[3] Buffer <- One byte; Output Device <- W, for all disassembled bytes
[4] If WC = 0, then Interrupt to acknowledge done, else go to [1]

CYCLE STEALING

While DMA I/O takes place, CPU is also executing instructions
DMA Controller and CPU both access Memory -> Memory Access Conflict

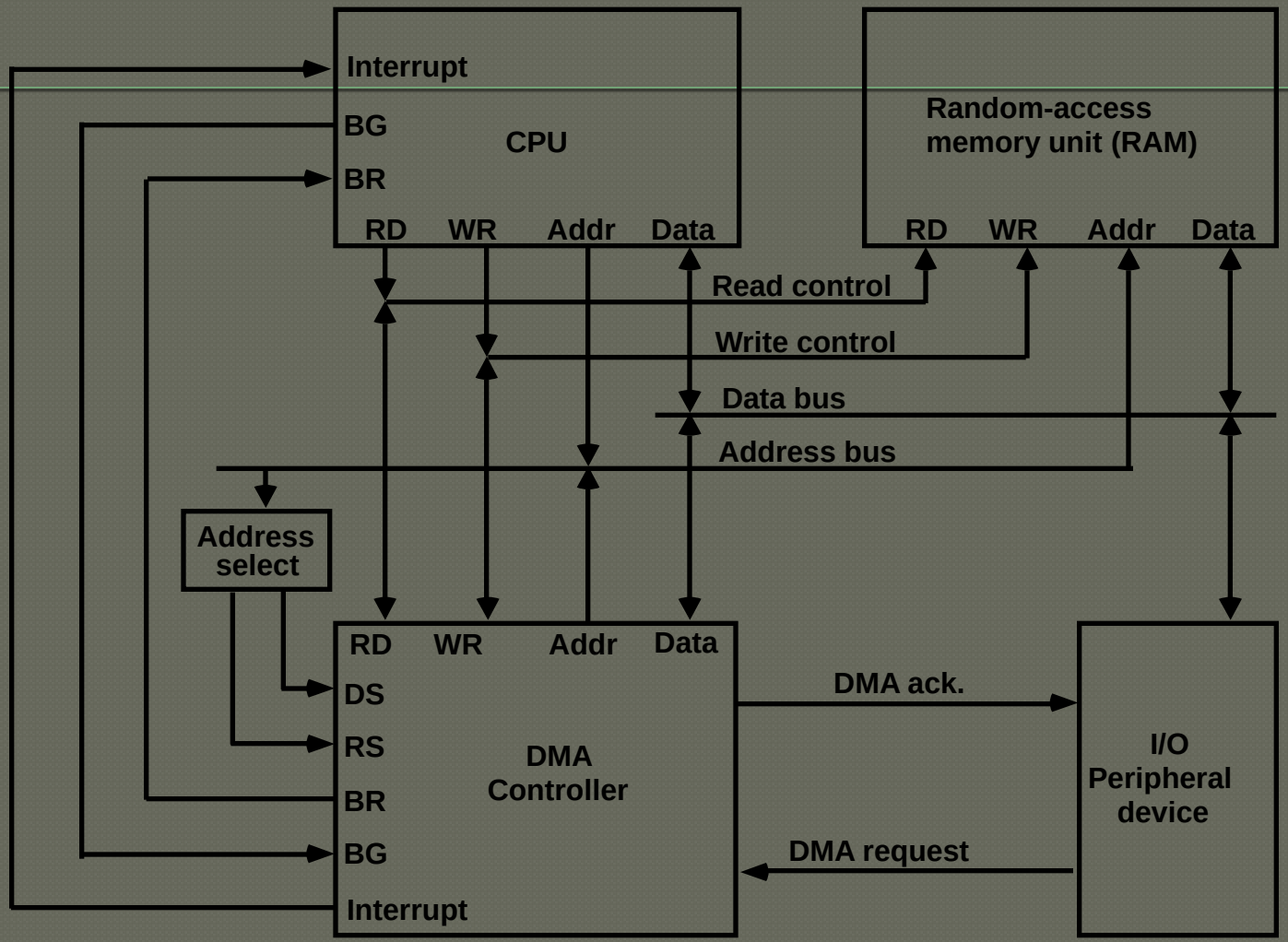
Memory Bus Controller

- **Coordinating the activities of all devices requesting memory access**
- **Priority System**
  Memory accesses by CPU and DMA Controller are interwoven, with the top priority given to DMA Controller
  -> Cycle Stealing

Cycle Steal

- **CPU is usually much faster than I/O(DMA), thus CPU uses the most of the memory cycles**
- **DMA Controller steals the memory cycles from CPU**
- **For those stolen cycles, CPU remains idle**
- **For those slow CPU, DMA Controller may steal most of the memory cycles which may cause CPU remain idle long time**

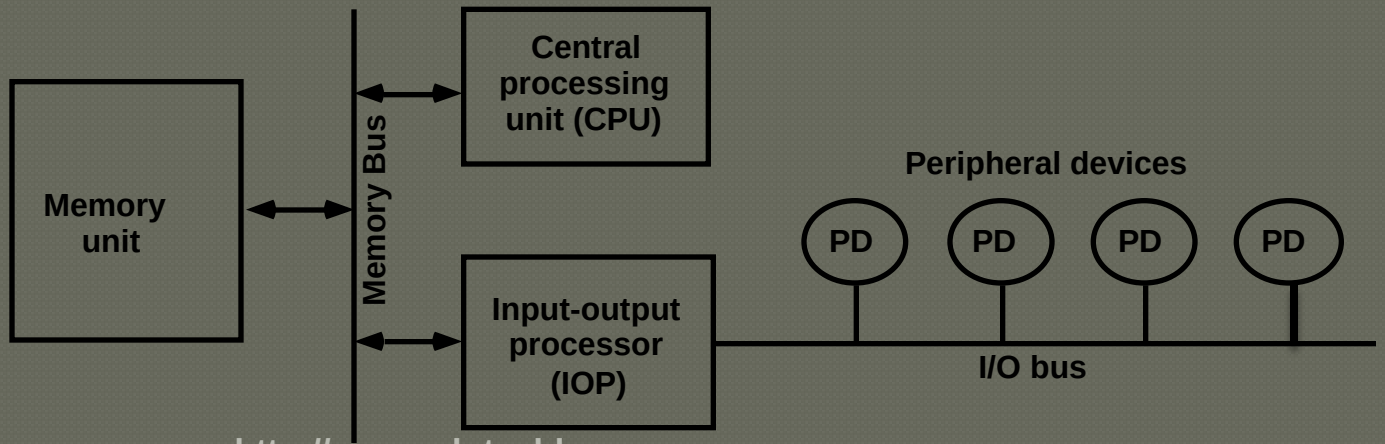
DMA TRANSFER



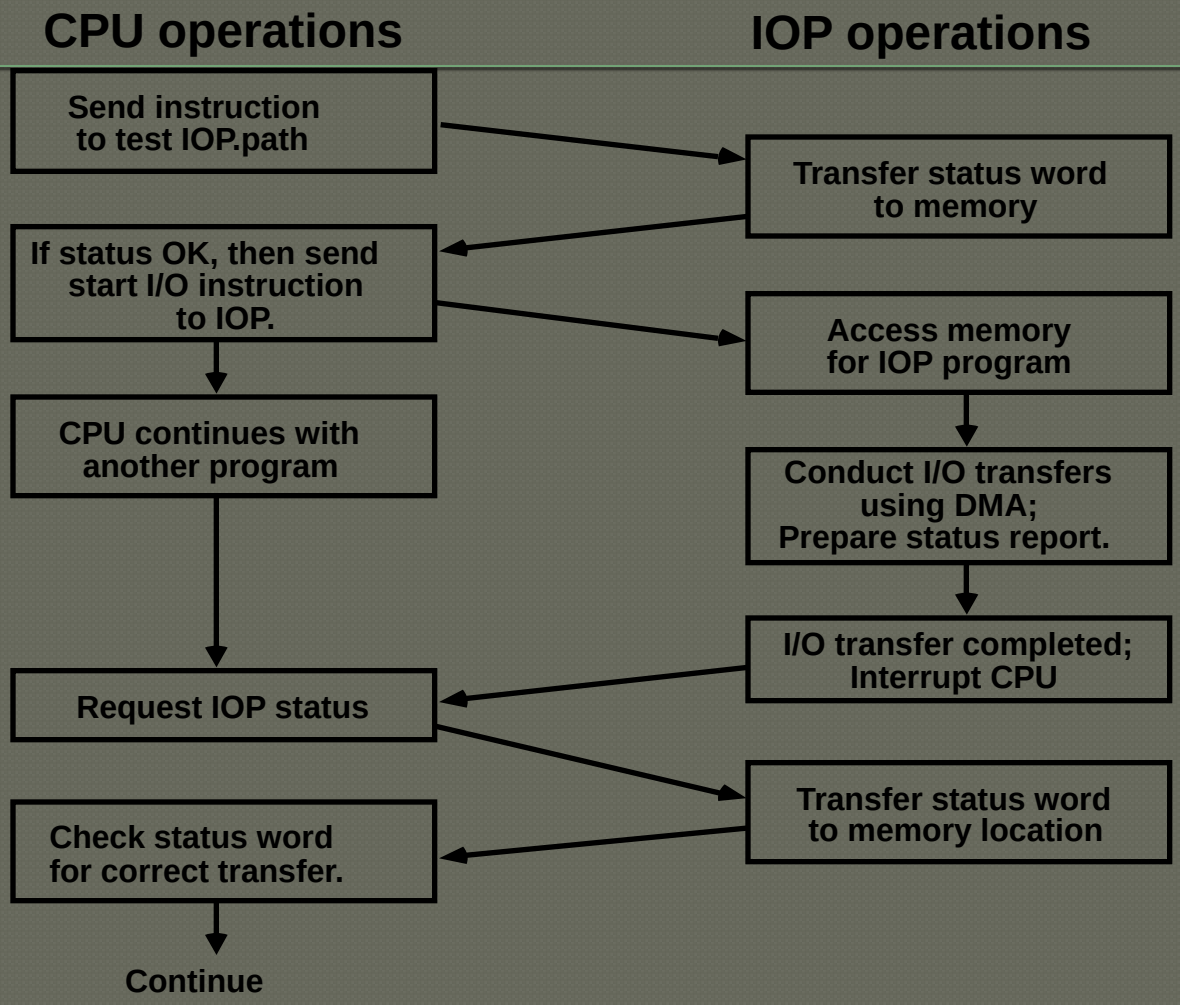
INPUT/OUTPUT PROCESSOR - CHANNEL -

Channel

- **Processor with direct memory access capability that communicates with I/O devices**
- **Channel accesses memory by cycle stealing**
- **Channel can execute a Channel Program**
  - **Stored in the main memory**
  - **Consists of Channel Command Word(CCW)**
  - **Each CCW specifies the parameters needed by the channel to control the I/O devices and perform data transfer operations**
- **CPU initiates the channel by executing an channel I/O class instruction and once initiated, channel operates independently of the CPU**



CHANNEL / CPU COMMUNICATION



NETWORK TOPOLOGIES

There are some basic configurations used to connect computers they are as:-

- Ring
- Star
- Mesh
- Tree

Ring Topology

- In Ring topology each node is connected to the two nearest nodes so the entire network forms a circle
- Data only travels in one direction on a Ring network

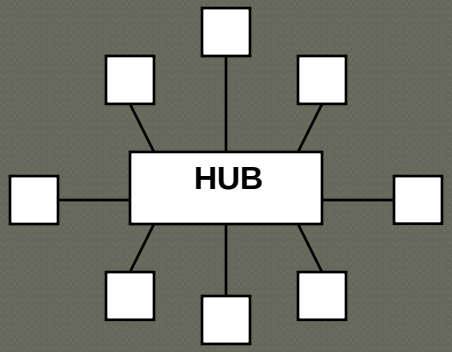
How this Topology works

- a node has information to send to another computer on the network so it sends the information out on the network to the PC it is connected to, if the information is for this PC (the recipients NIC address is attached to the message, which is like putting an address on an envelope) then the PC accepts the data
- otherwise it passes the information on to the next PC by repeating the data back out on the line
- This method of repeating the data helps keep the integrity of the data readable by other computers

Problems and Solutions

- The drawback to this type of topology is that a single malfunctioning workstation can disable the whole network
- To make sure all the information is sent the receiving PC sends the token back to the sending PC after it has received all the data
- If the sending PC is finished sending it passes the token to the next PC
- This type of network was also widely used in the 1980's

Star topology



- In a Star topology every node is connected through a central device such as a Hub, Switch or Router
- Compared to a Ring or Bus topology a Star topology requires that more thought be put into its setup

The Good and Bad of a Star Network

- The upside of a star network is that if any one cable fails then only the node connected on that cable would be affected
- Another positive point to this type of network is that it is very simple to join two star networks together by connecting their central devices to each other

The Good and Bad of a Star Network

- As each computer is connected to a central device (Hub) the location of the Hub must be made as central as possible, so as to reduce cable lengths
- The drawback to this type of topology is if a central device was to fail then all computers connected to that device would not be able to see the network

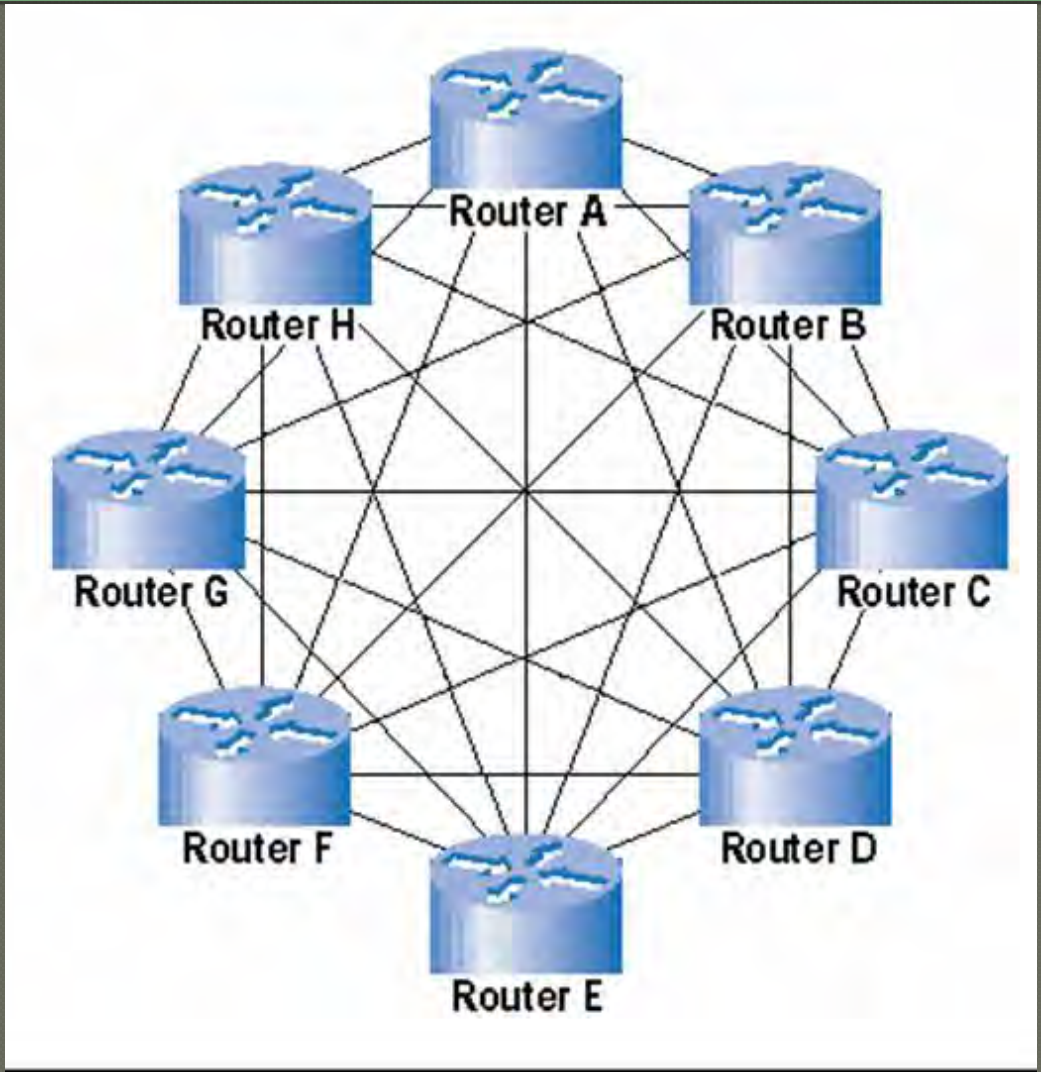
Mesh Network

- Natural way of interconnecting a large no. of nods by means of mesh.
- A local area network can employs either a full mesh topology or partial mesh topology
- Full mesh topology- each node is connected directly to each of the others
- Partial mesh topology- some nodes are connected to all the others, but some of them are only connected to nodes with which they exchange the most data

Full Mesh Toplology

- Every node has a circuit connecting it to every other node in the network
- Yields greatest redundancy, so if one node fails, network traffic can be redirected to any of the other nodes
- Usually reserved for backbone networks since it is very expensive

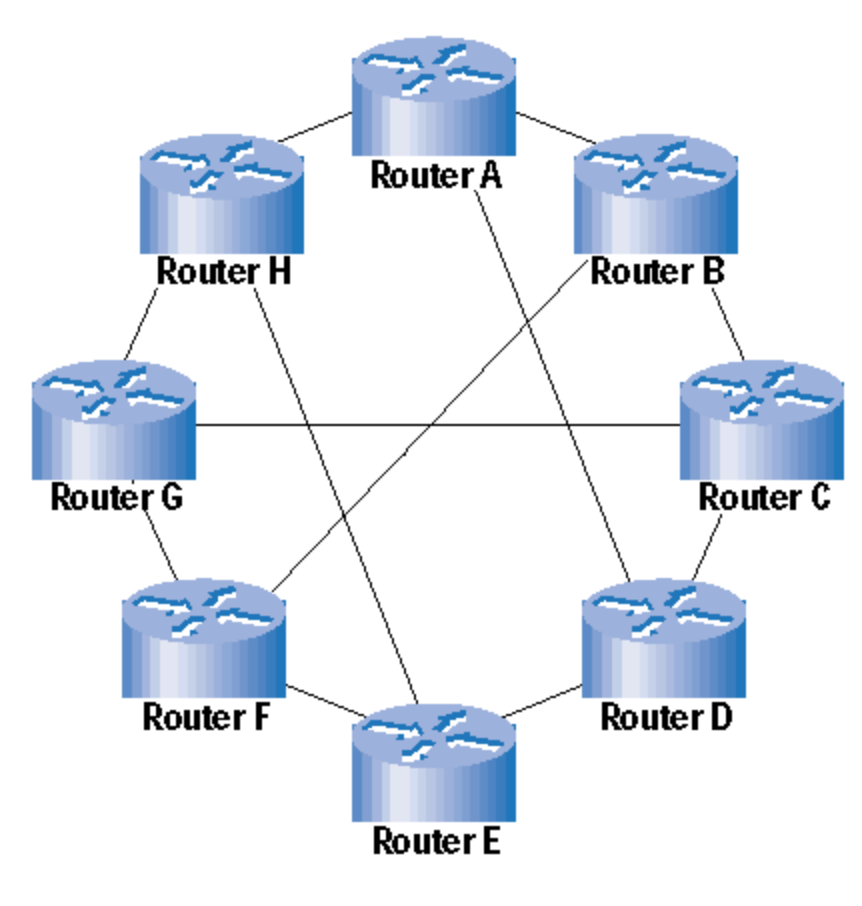
Full Mesh Toplology



Partial Mesh Toplology

- Some nodes are organized in a full mesh scheme but others are only connected to 1 or 2 in the network
- Common in peripheral networks connected to a full meshed backbone
- Less expensive to implement
- Yields less redundancy

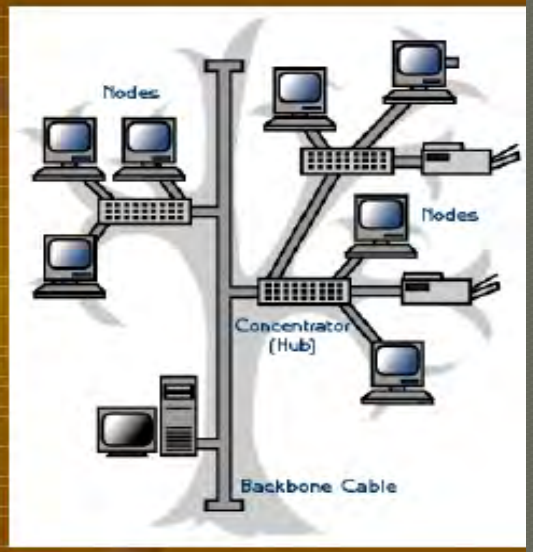
Partial Mesh Toplology



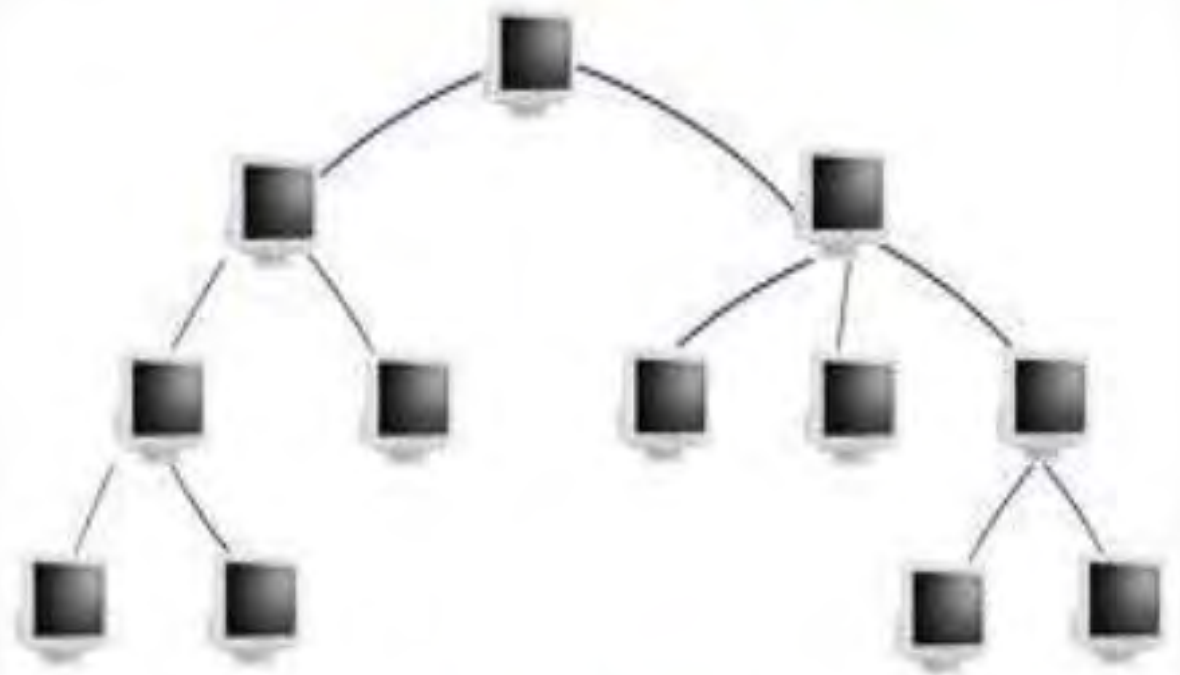
Tree Networks

Tree Topology

- A tree topology combines characteristics of linear bus and star topologies.
- It consists of groups of star-configured workstations connected to a linear bus backbone cable.
- Tree topologies allow for the expansion of an existing network, and enable schools to configure a network to meet their needs



Tree Topology



Digram - Tree Topology

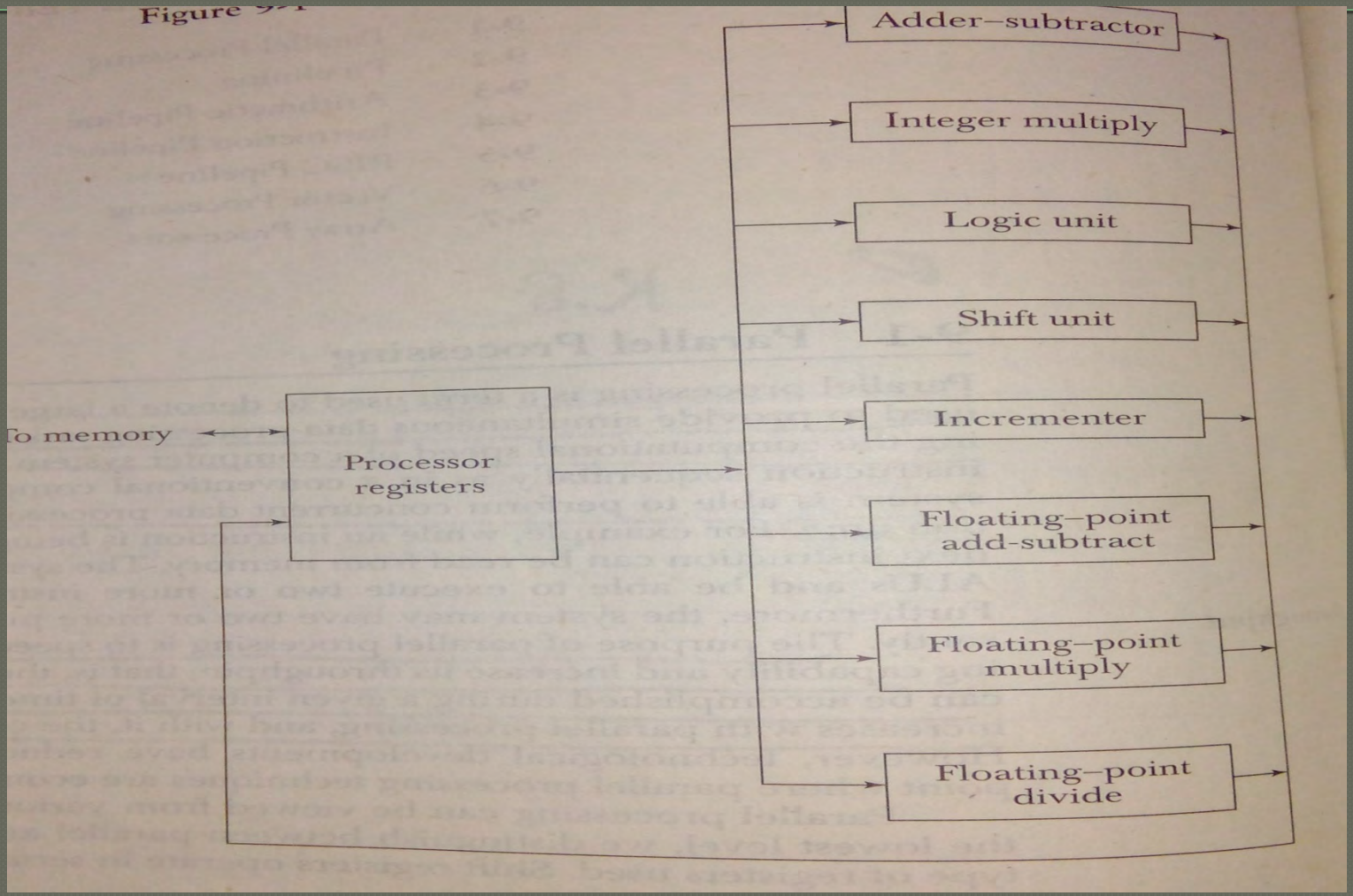
PARALLEL PROCESSING

Execution of *Concurrent Events* in the computing process to achieve faster *Computational Speed*

Levels of Parallel Processing

- **Job or Program level**
- **Task or Procedure level**
- **Inter-Instruction level**
- **Intra-Instruction level**

Processors with multiple functional units



PARALLEL COMPUTERS

Architectural Classification

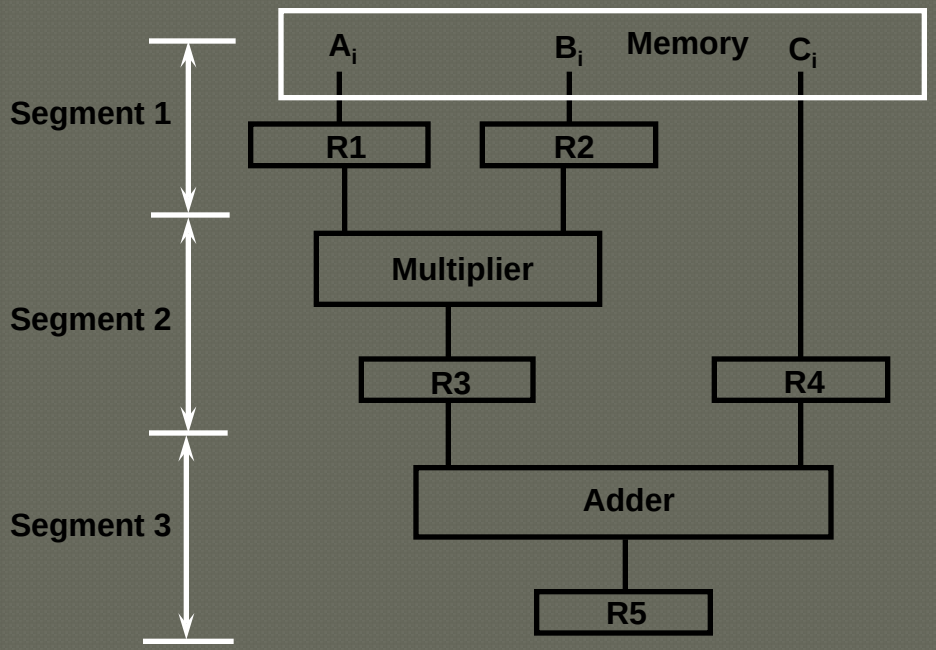
- Flynn's classification
  - Based on the multiplicity of *Instruction Streams* and *Data Streams*
  - Instruction Stream
    - Sequence of Instructions read from memory
  - Data Stream
    - Operations performed on the data in the processor

		Number of *Data Streams*	
		Single	**Multiple**
Number of *Instruction Streams*	**Single**	**SISD**	**SIMD**
	Multiple	**MISD**	**MIMD**

PIPELINING

A technique of decomposing a sequential process into suboperations, with each subprocess being executed in a partial dedicated segment that operates concurrently with all other segments.

$A_i * B_i + C_i$ for i = 1, 2, 3, ... , 7



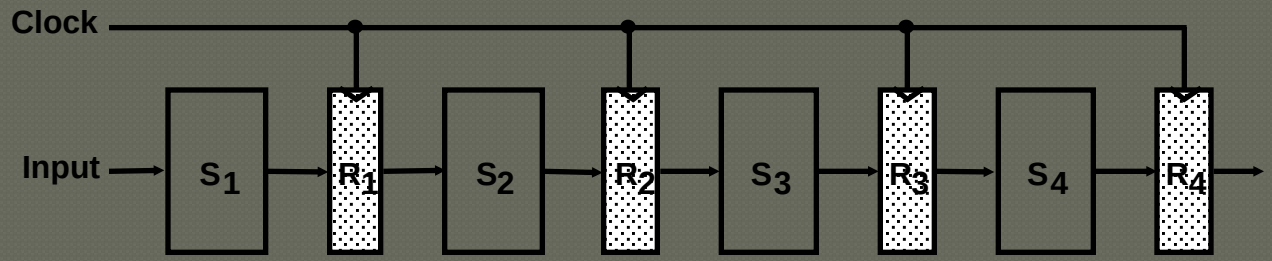
$R1 \leftarrow A_i$, $R2 \leftarrow B_i$	Load A_i and B_i
$R3 \leftarrow R1 * R2$, $R4 \leftarrow C_i$	Multiply and load C_i
$R5 \leftarrow R3 + R4$	Add

OPERATIONS IN EACH PIPELINE STAGE

Clock Pulse Number	Segment 1		Segment 2		Segment 3
	R1	R2	R3	R4	R5
1	A1	B1			
2	A2	B2	A1 * B1	C1	
3	A3	B3	A2 * B2	C2	A1 * B1 + C1
4	A4	B4	A3 * B3	C3	A2 * B2 + C2
5	A5	B5	A4 * B4	C4	A3 * B3 + C3
6	A6	B6	A5 * B5	C5	A4 * B4 + C4
7	A7	B7	A6 * B6	C6	A5 * B5 + C5
8			A7 * B7	C7	A6 * B6 + C6
9					A7 * B7 + C7

GENERAL PIPELINE

General Structure of a 4-Segment Pipeline



Space-Time Diagram

Segment	1	2	3	4	5	6	7	8	9
1	T1	T2	T3	T4	T5	T6			
2		T1	T2	T3	T4	T5	T6		
3			T1	T2	T3	T4	T5	T6	
4				T1	T2	T3	T4	T5	T6

Clock cycles