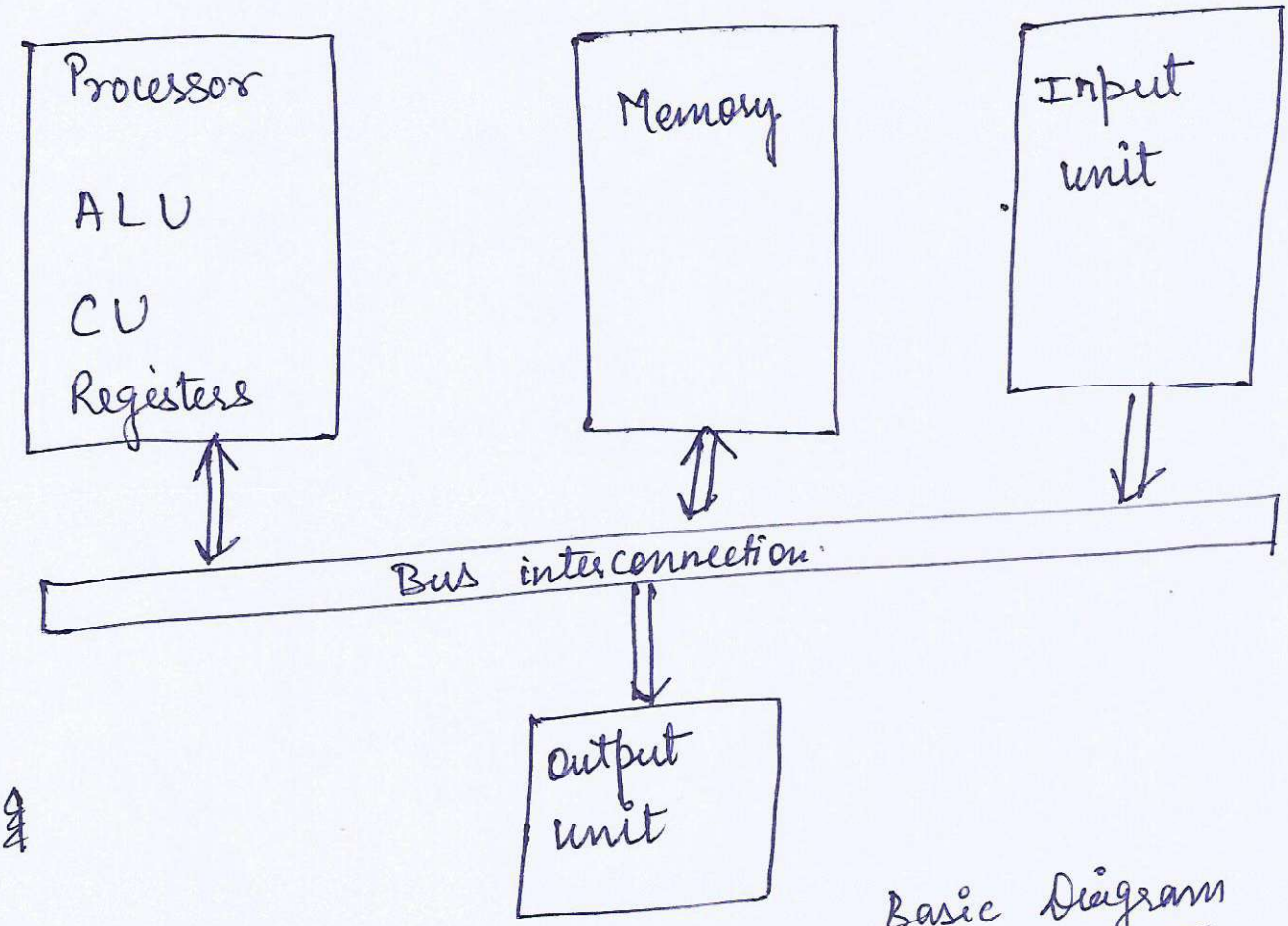
COA Unit-1 Notes

Minakshi

Functional units and their interconnections:

→ Functional units of a compiler / digital system are:

- → Input unit
- → Output Unit
- → Central Processing Unit (CPU)
- → Memory
- → Bus structure.



Basic Diagram

Note: Theory about these components are given in "Fundamental of COA" Notes and Unit 1 Notes (about bus)

Buses, bus Architecture, types of buses and bus arbitration

Bus:

→ Bus is a communication system that helps data transfer between different modules of the computer.

→ A bus is a communication pathway connecting two or more devices.

→ A bus is a group of electrical lines/wires that carry computer signals/bits.

→ A bus consists of multiple lines. Each line is capable of transmitting signals representing 1 or binary 0.

> A bus that connects major computer components (processor, memory, I/O) is called a system bus.

→ On any bus lines can be classified into three function groups:-

1. Data Bus
2. Address Bus
3. Control Bus.

Data Bus:- Data lines provides path for moving data between components.

- Bidirectional

→ Width of databus is number of lines in data bus.

→ Each line can carry only one bit at a time.

Address Bus: (Unidirectional)

→ The address lines are used to designate (know) the source or destination of the data on the data bus.

For example: if the processor wishes to read or write a memory word, it puts the address of desired word on the address lines.

The width of address bus determines the maximum possible memory capacity of the system.

control Bus: (Bidirectional)

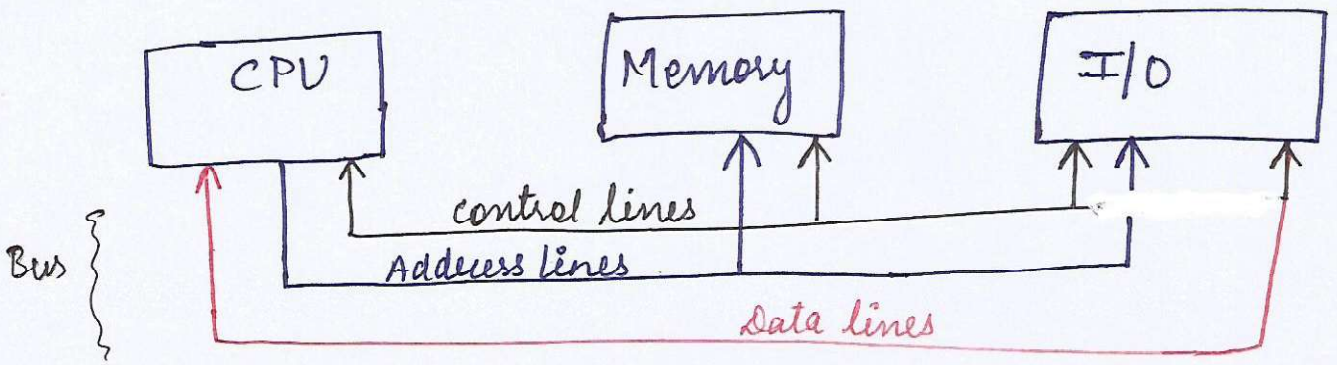
- The control lines are used to control the access to and the use of the data and address lines.
- control signals transmit both command and timming information

Timing signals ⇒ indicate the validity of data and address information

command signals ⇒ specify operations to be performed.

- Typical control lines include:-
  - Memory write
  - Memory Read
  - I/O write
  - I/O Read
  - Transfer Ack
  - Bus Request
  - Bus Grant
  - Interrupt Request
  - Interrupt Ack
  - clock
  - etc.

4.



Bus interconnection scheme.

Types of Buses:

- Dedicated
- Multiplexed

- Dedicated bus line is permanently assigned either to one function or to a physical subset of computer components.
- Example of functional dedication :- the use of separate dedicated address and data lines.
- Multiplexed: Address and data information may be transmitted over the same set of lines using an Address Valid control line.

For example:

At the beginning of a data transfer, the address is placed on the bus and Address valid line is activated. ~~At this point~~ The address is then removed from the bus and the same bus are used for subsequent read or write data transfer. (this known as time multiplexing)

5/

Bus Arbitration:

→ More than one module may need control of the bus e.g CPU and DMA controller.

e.g. I/O module may need to read or write directly to the memory without sending the data to the processor.

The process by which multiple requests are recognized and given priority given to one of them is called arbitration

- Arbitration can be — Centralized / localized
  — Decentralized / Distributed

→ In a centralized scheme, a single hardware device, referred to as a bus controller or arbiter, is responsible for allocating time on the bus. The device may be a separate module or part of the processor.

- In distributed scheme, there is no central controller. Rather, each module contains access control logic and the modules act together to share the bus.

- With both methods of arbitration, the purpose is to designate one device, either the processor or an I/O module, as master. The master may initiate a data transfer (e.g. read or write) with some other device, which acts as slave for this particular exchange.

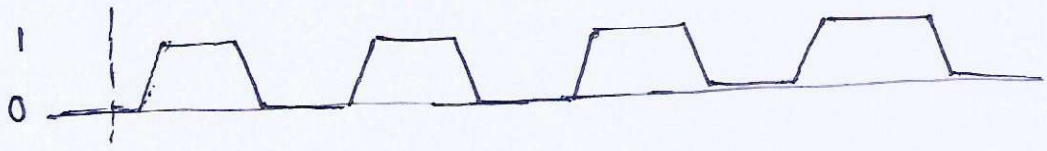
Timing

Timing Timing Refers to the way in which events are coordinated on the bus.

- Buses use — ① Synchronous Timing
  — ② Asynchronous Timing

Synchronous Timing:

- With synchronous timing the occurrence of the events on the bus is determined by a clock.
- The bus includes a clock line upon which a clock transmits a regular sequence of alternative 0's and 1's of equal duration. A single 1-0 transmission is referred to as a clock cycle or bus cycle and defines a time slot.
- All events starts at the beginning of a clock cycle.



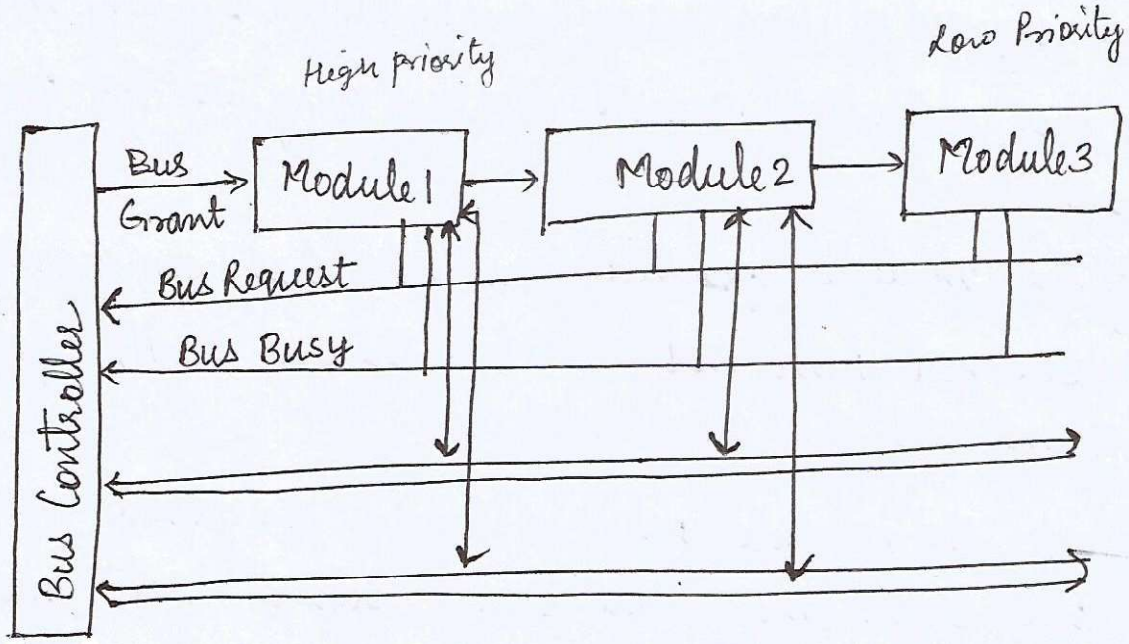
Asynchronous Timing:

→ with asynchronous timing, the occurrence of one event on a bus follows and depends on the occurrence of a previous event.

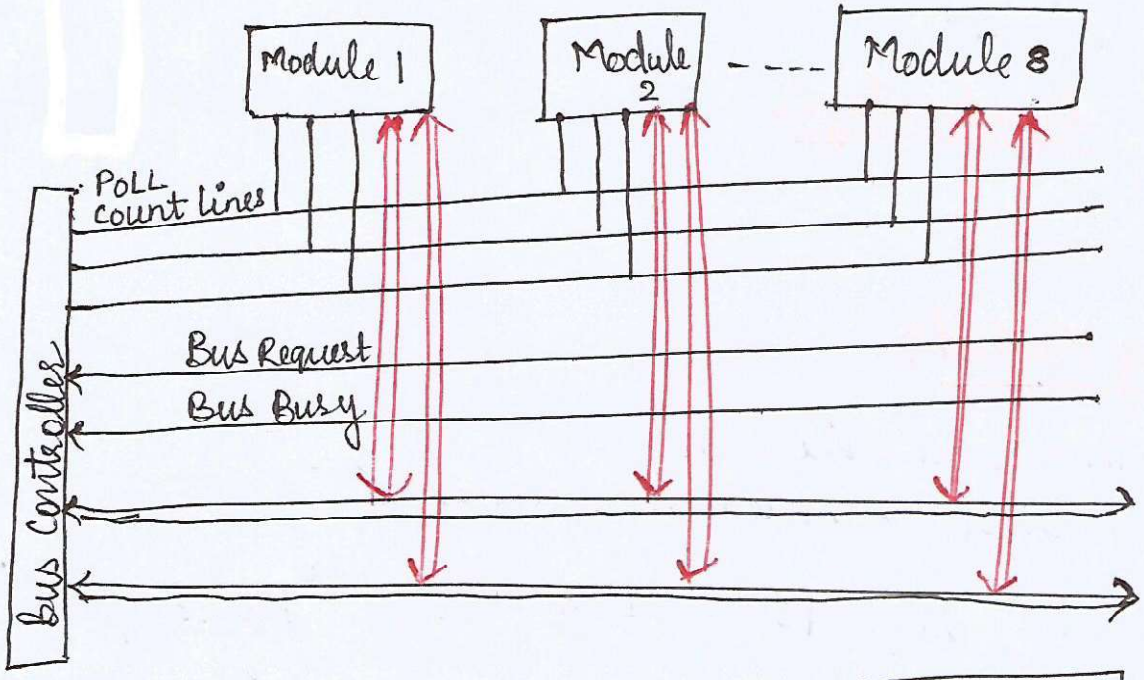
→ No clock.

Bus Arbitration Techniques:

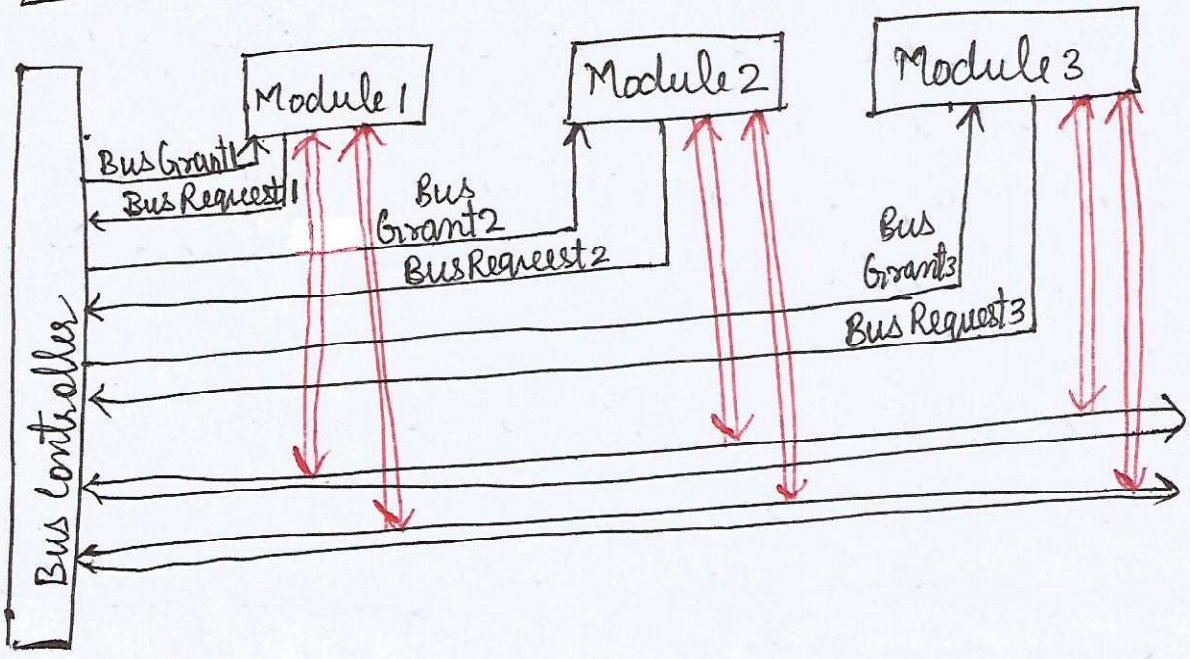
Daisy chain



Polling



Independent Requesting



Daisy-chain Method or Serial Method:

⇒ The request of bus usage can be placed by any module connected.

⇒ The bus controller grants the bus only to the first module connected.

⇒ ⇒ if the module ~~granted bus~~, who has been granted bus to, ~~does not~~ have not requested the bus, will forward the grant to the next module.

⇒ if the module who is having the grant wants to use bus, makes the busy signal 'set' and use the bus. After using bus, the module unset the busy signal.

Disadvantage: Modules has the fixed Priority.

Polling Method:

Polling count line = n

Max Number of connected modules ⇒ 2^n.

⇒ Modules requests the bus using 'Bus Request' line.

⇒ Bus controller places any one of 2^n sequences on the poll count lines.

⇒ The module, for which polling sequence is allocated by the bus controller, can set busy signal and use the bus.

⇒ Priority is set by the bus controller.

⇒ It is time consuming method.

⇒ Non-productive Polling ⇒ Bus controller generated the polling sequence for the module which did not request the bus

10/

Independent Request Arbitration Technique:

→ In this method, every module has connected to the bus controller by separate 'Bus grant' and 'Bus Request'

⇒ Individual module can request to the bus controller by its own 'Bus Request' line. And, Bus controller can grant the bus to the module which has requested for the bus.

⇒ But this method increases the cost of the system.

⇒ Better Performance.

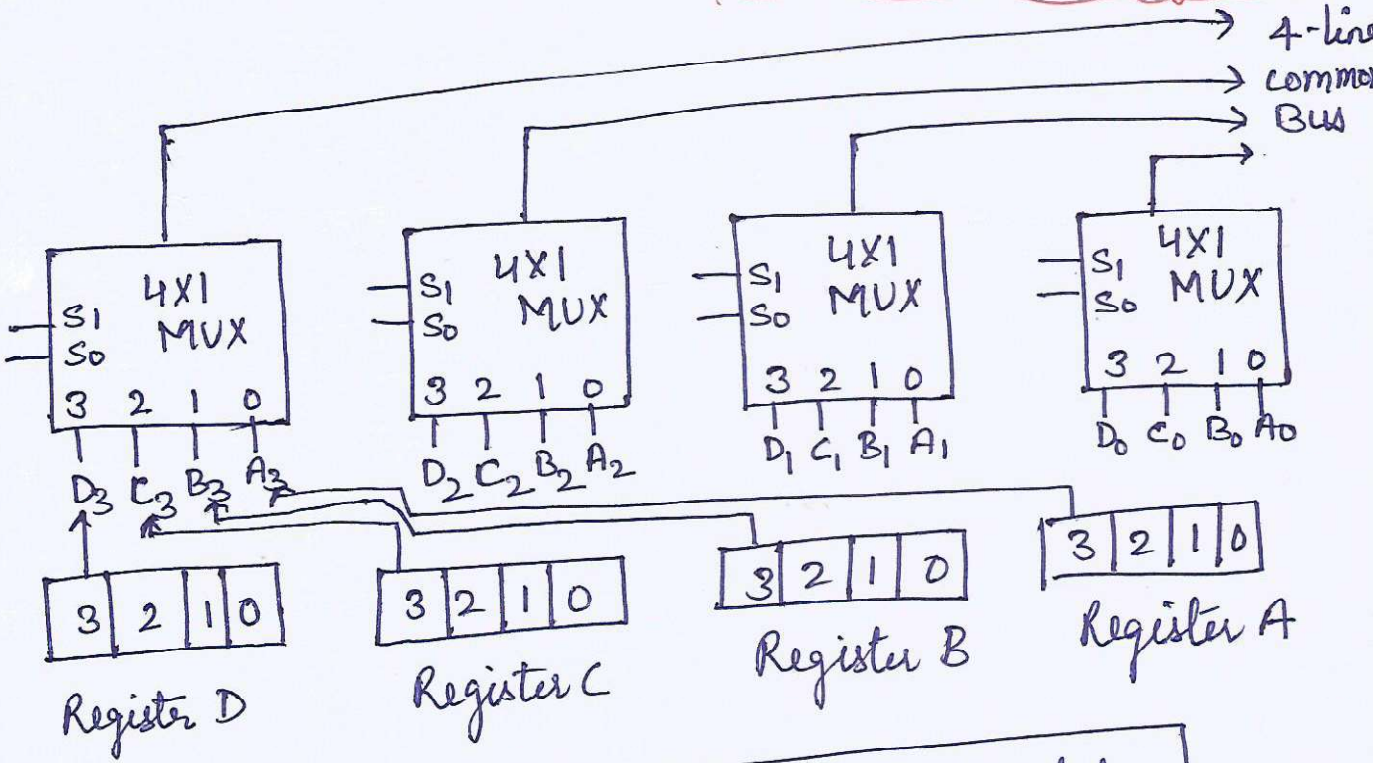
Bus and Memory Transfer:

→ A typical digital computer has many registers.
→ A path must be provided to transfer information from one register to another.
→ common bus structure
  → using Multiplexer.
  → Using Tristate Buffer.

Bus system using Multiplexer:

Example ⇒ For four registers.

Number of multiplexers = Size of register.
Size of multiplexer = Number of registers



Function Table:

S_1	S_0	Register selected
0	0	A
0	1	B
1	0	C
1	1	D

Memory Transfer:

~~Memory Read: The transfer of information to be stored into memory~~

MemoryRead: The transfer of information from memory to the outside environment is called a read operation

Read: $DR \leftarrow M[AR]$

DR ⇒ Data Register

M[AR] ⇒ memory location specified by Address Register AR.

Memory Write: The transfer of new information to be stored into the memory is called a write operation

Write: $M[AR] \leftarrow DR$

Registers

- Registers are used to store data temporarily

→ The register in CPU performs two roles

- User Visible Register
  - General Purpose Registers
  - Data Register
  - Address Register
  - Condition codes
- control and Status Register
  - Program Counter
  - Instruction Register
  - MAR
  - MDR

User Visible Register:

- A user visibe register may be referenced by means of machine language that the processor executes.
- categories – General Purpose
  - Data Register
  - Address Register
  - Condition Codes

General Purpose Registers:

- can be used by the programmer.
- contains operands for any opcode.
- can be used for addressing functions (e.g. Register indirect, displacement).

Data Register: Data Register may be used only to hold the data

Address Register:

Address Register may be devoted to a particular addressing modes. Examples
- Segment Pointer
- Index Register
- Stack Pointer.

- Segment Pointer holds the address of the base of the segment in segmented addressing.
- Index Registers are used for indexed addressing and may be auto indexed.
- Stack Pointer points to the top of the stack. This allows implicit addressing that is push, pop and other stack instructions need not ~~the~~ contain an explicit stack operand.

Condition codes (Also referred to as Flags):

condition codes are the bits set by the processor hardware as the result of operation.

Example:
- positive or negative result
- Zero Result
- Overflow
- etc.

control and status Register:

→ Registers employed to control the operation of processor.
→ Not visible to the user.
→ Four Registers are essential to instruction execution
  ① Program counter
  ② Instruction Register
  ③ Memory Address Register (MAR)
  ④ Memory Data Register (MDR)

- Program counter contains the address of the instruction to be fetched
- Instruction Register contains the instruction most recently fetched.

→ MAR contains the address of a location in the memory.
→ MDR or MBR (Memory Buffer Register) contains a word of the data to be written to memory or the word most recently read.

PSW : Program Status Word :

Many processors include a set of registers known as PSW, that contain the status information
- condition code and other status information

Sign: contains the sign bit of the result.
Zero: set when the result is zero.
Carry: set if an operation resulted in a carry into or borrow out of a higher order bit.

Equal: set if a logical compare result is equal.

Overflow: used to indicate arithmetic overflow.

Interrupt Enable/disable: used to enable or disable interrupts

Supervisor:- indicates whether the processor is executing supervisor or user mode.
Certain previledged instructions can be executed only in supervised mode, and certain memory areas can be accessed only in supervisor mode.

Topic

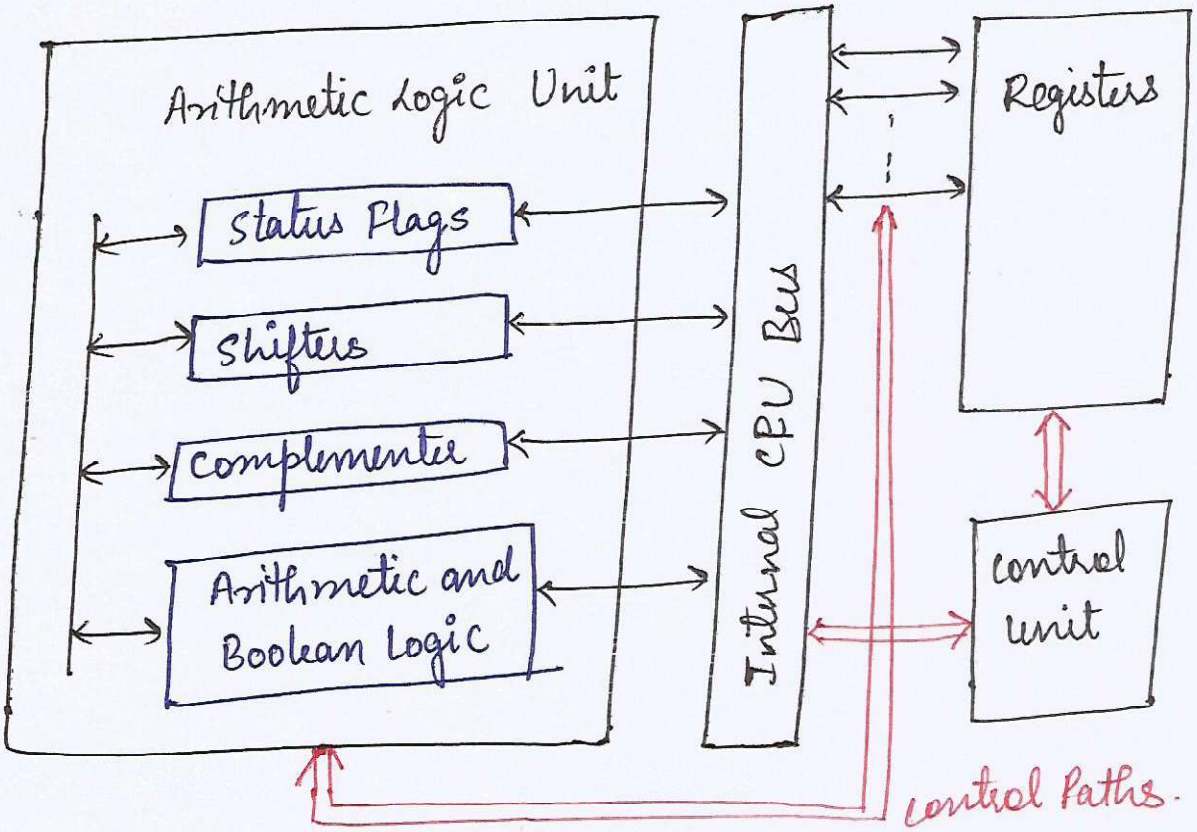
Processor Organization:

Processor contains ALU, CU and Registers

→ ALU ⇒ The ALU (Arithmetic Logic Unit) does the actual computation or processing of the data

→ CU: control unit:- CU controls the movement of data and instruction into and out of the processor and controls the operation of ALU.

→ Registers: minimal internal memory.



Internal structure of CPU

- Processor Organization means how the components of processor are connected and accomplish their tasks.

A processor does the following things.

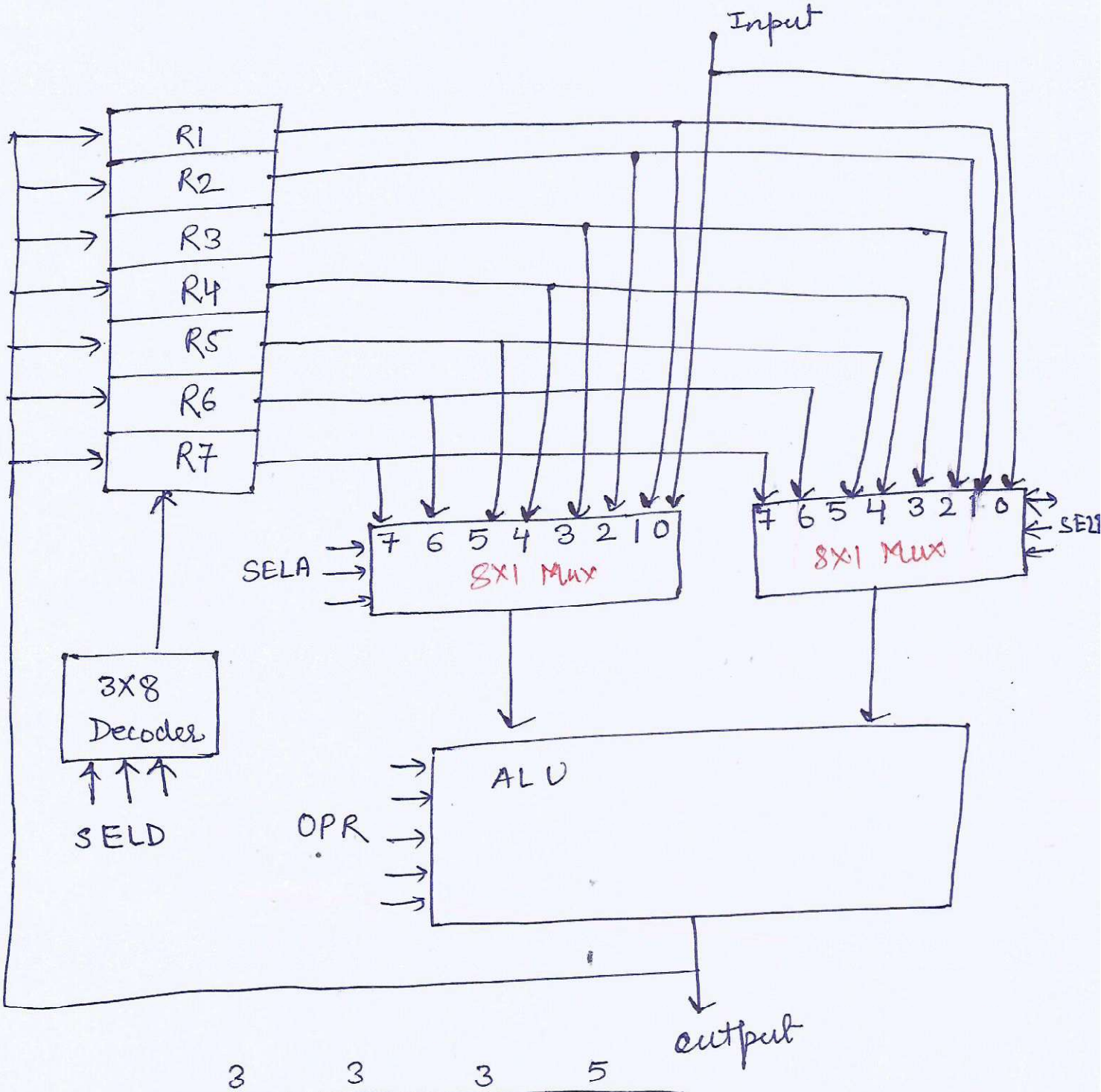
- Fetch Instruction
- Interpret instruction
- Fetch Data
- Process Data
- Write Data

- **Fetch Instruction:** The processor reads an instruction from memory.
- **Interpret Instruction:** The instruction is decoded to determine what action is required.
- **Fetch data:** The execution of an instruction may require reading data from memory or an I/O module
- **Process Data:** The execution of an instruction may require performing some arithmetic or logical operation on the data.
- **Write Data:** The result of an execution may require writing data to memory or I/O module.

Topic

General Registes Organization :

→ A bus organization for Seven CPU registers.



control word.

Register set with common ALU

- The output of each Register is connected to two multiplexers to form the buses A and B.
- The selection lines in each multiplexer selects one register or input for the particular bus.
- → The buses A and B form the inputs to a common ALU.
- → The operation selected in the ALU determines the arithmetic or logic microoperation that is to be performed.
- → The result is available for output data and goes into the inputs of all registers. Register for output is selected by a decoder.
- → The decoder activates one of the register load inputs.

Register Selection:

Binary Codes	SEL A	SEL B	SEL D
000	input	input	None
001	R1	R1	R1
010	R2	R2	R2
011	R3	R3	R3
100	R4	R4	R4
101	R5	R5	R5
110	R6	R6	R6
111	R7	R7	R7

Table of operations performed by ALU

	OPR select	Operation
0	00000	Transfer A
1	00001	Increment A
2	00010	Add A+B
5	00101	Subtract A-B
6	00110	Decrement A
8	01000	AND A, B
10	01010	OR A, B
12	01100	XOR A, B
14	01110	complement A
16	10000	shift Right A
24	11000	shift Left A

This table specifies some selected operants only ⇒ 11

five bits can represent total 32 operation

Topic.

Stack Organization:

→ stores information is LIFO order (Last In First Out)

→ The register that holds the address of the stack is called Stack Pointer.

→ Stack Pointer's value always points to the top element.

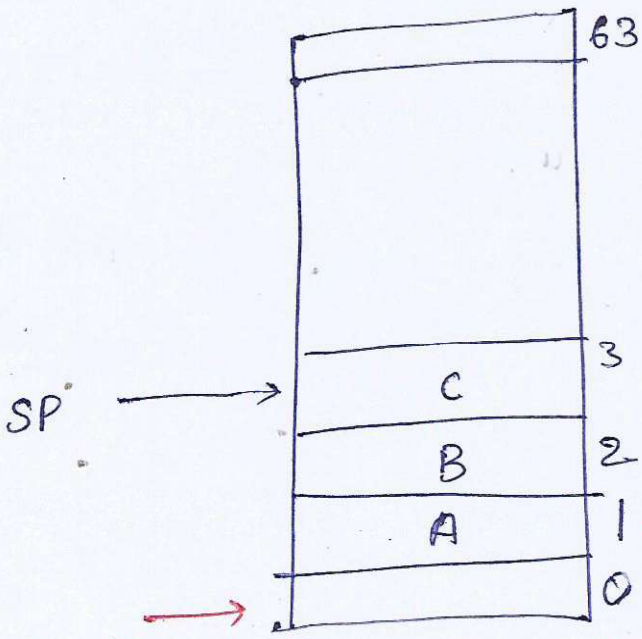
→ Two operations on stack ① Push
② Pop

→ Push ⇒ to insert an element in the stack
Pop ⇒ to delete an element from the stack

⇒ Two status Full ⇒ set to 1 when stack is full
Empty ⇒ set to 1 when stack is empty

⇒ Stack can be ⇒ collection of memory words or registers

⇒ Example 64 - word register stack



Full
Empty

Full → 1 ⇒ Full
0 ⇒ not full

Empty → 1 Empty
0 not empty

initially SP ← 0
EMPTY ← 1
FULL ← 0

if (Full = 0) ⇒ new item is inserted with push operation
if (EMPTY = 0) ⇒ an item can be deleted from stack with popoperation

Push operation :-

SP ← SP + 1 increment stack Pointer
M[SP] ← DR write item on the top of stack.
if (SP = 0) then FULL ← 1
EMPTY ← 0

M[SP] denotes the memory word specified by the address presently available in the SP.

⇒ The first item is stored in stack at address 1.
The last item is stored in the stack at address 0.

Pop Operation :

A new item is deleted from the stack if stack is not empty

DR ← M[SP] ⇒ Read the data from the top of stack.
SP ← SP - 1 ⇒ Decrement SP
if (SP = 0) then EMPTY ← 1 Check if the stack is empty
FULL ← 0 ⇒ Mark the stack not full.

ADDRESSING MODES:

Two issues: ① How is the address of an operand specified?
② How the bits of an instruction organized to define the operand addresses and operation of that instruction.

The operation field of an instruction specifies the operation to be performed. This operation must be executed on some data stored in the computer registers or memory words.
The way the operands are choosen during the program execution depends on the addressing mode of the instruction

The addressing mode specifies a rule for interpreting or modifying the address field of the instruction before the operand is actually referenced.

Most Common addressing techniques:

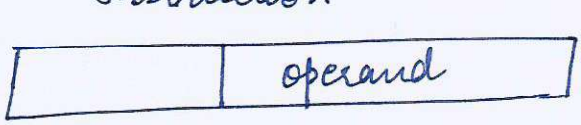
- Immediate Addressing
- Direct Addressing
- Indirect addressing
- Register Addressing
- Register Indirect Addressing
- Displacement → Relative Addressing
  → Base Register Addressing
  → Indexing
- Stack addressing

① Immediate Addressing:

→ In this addressing mode, the operand value is present in the instruction.

→ Advantage: no memory reference other than the instruction fetch is required to obtain the operand.

→ Disadvantage: Limited operand magnitude.

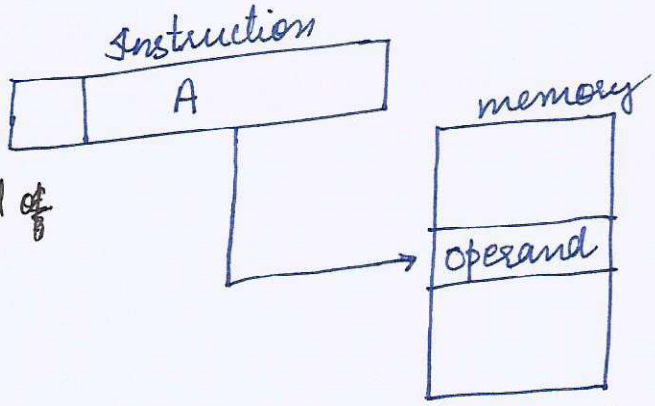


② Direct Addressing:

In this addressing mode, the address field contains the actual/effective address of the operand.

$EA = A$

A ⇒ content of an address field in the instruction



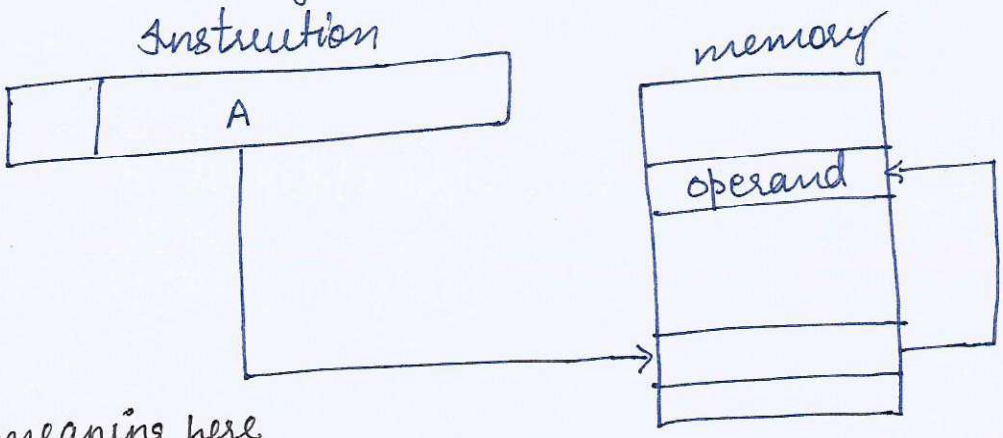
Advantage:

- It requires only one memory reference and no special calculation.
- simple

Disadvantage: Limited address space.

(3) Indirect Addressing:

In this addressing mode, address field refers to the address of a word in the memory.



EA = (A)

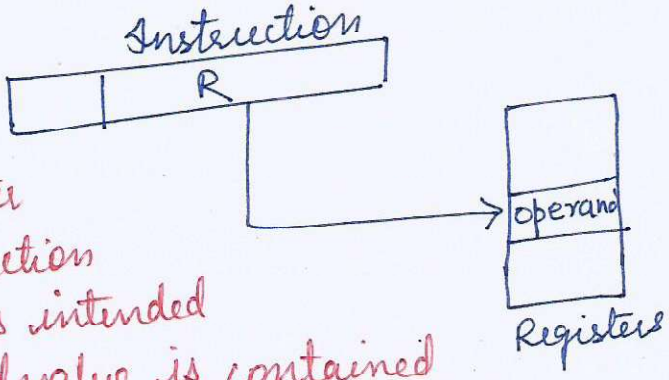
* parantheses meaning here
  ⇒ "content of"
* EA = Actual / effective address of the location containing the referenced operand.

Advantage: Large address space:
→ for a word length of N, an address space of 2^N is now available.

Disadvantage: requires two memory reference
→ one for address of operand.
→ one for to get its value (operand's)

(4) Register Addressing:

Register addressing is ~~similar~~ similar to direct addressing. The only difference is that the address field refers to a register rather than a main memory address.



EA = R

if the content of a register address field in an instruction is 5, then register R5 is intended address, and the operand value is contained in R5.

26/ Advantages:

→ only a small address field in instruction is needed
→ no time consuming memory references.

Disadvantages:

→ very limited register address space.

"Content of the register will be operand" in Register Addressing

⑤ Register Indirect Addressing:

- similar to indirect addressing
- But here, the address field of instruction specifies a register whose contents give the address of the operand in the memory.

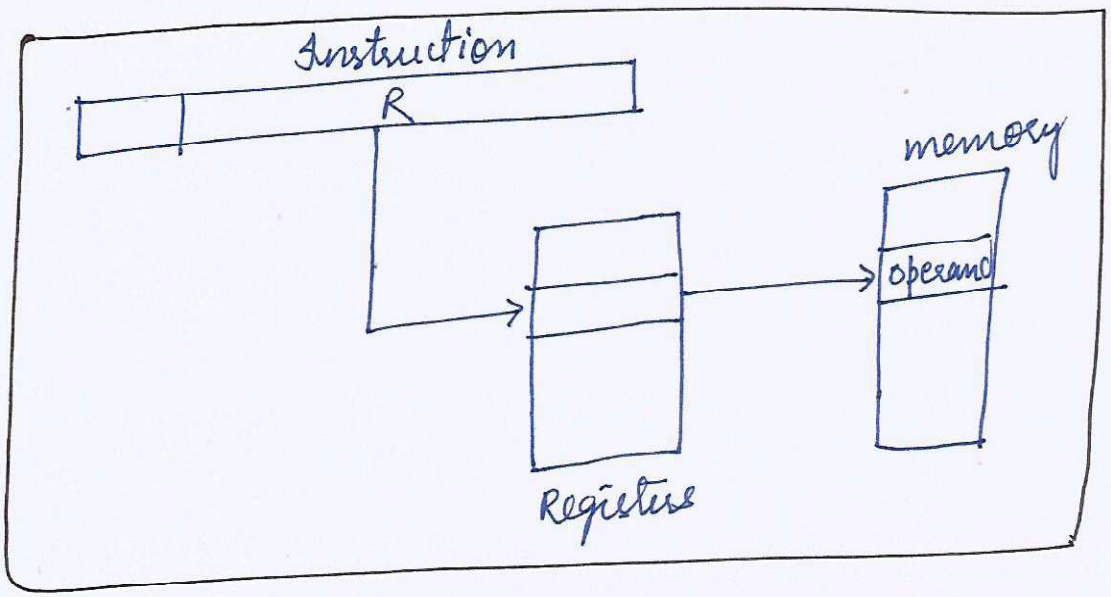
$$EA = (R)$$

Advantage:

→ fewer bits to select a register than a memory address
→ Large address space.

Disadvantage:

- Extra memory Reference.



COA Unit-5 Notes

Peripheral Devices:-

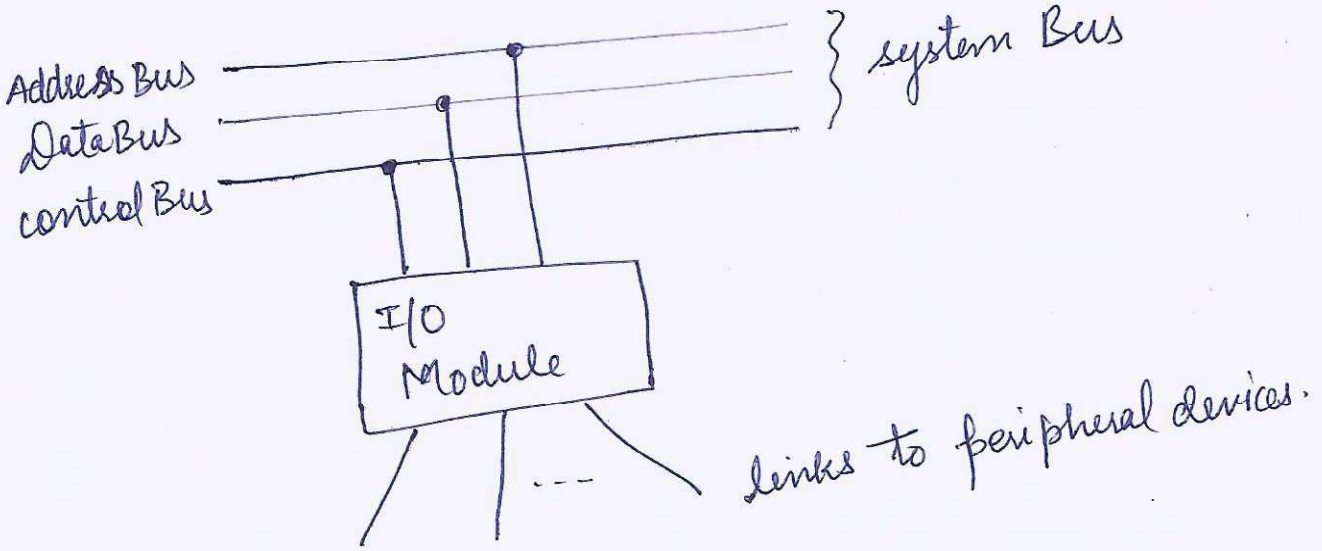
An external device connected to an I/O module is often referred to as a peripheral device or simply a peripheral.

3 categories of external devices:-

(1) Human Readable Device:
suitable for communicating with the computer user (e.g. VDT (Video Display Terminal), Printer, Keyboard, etc).

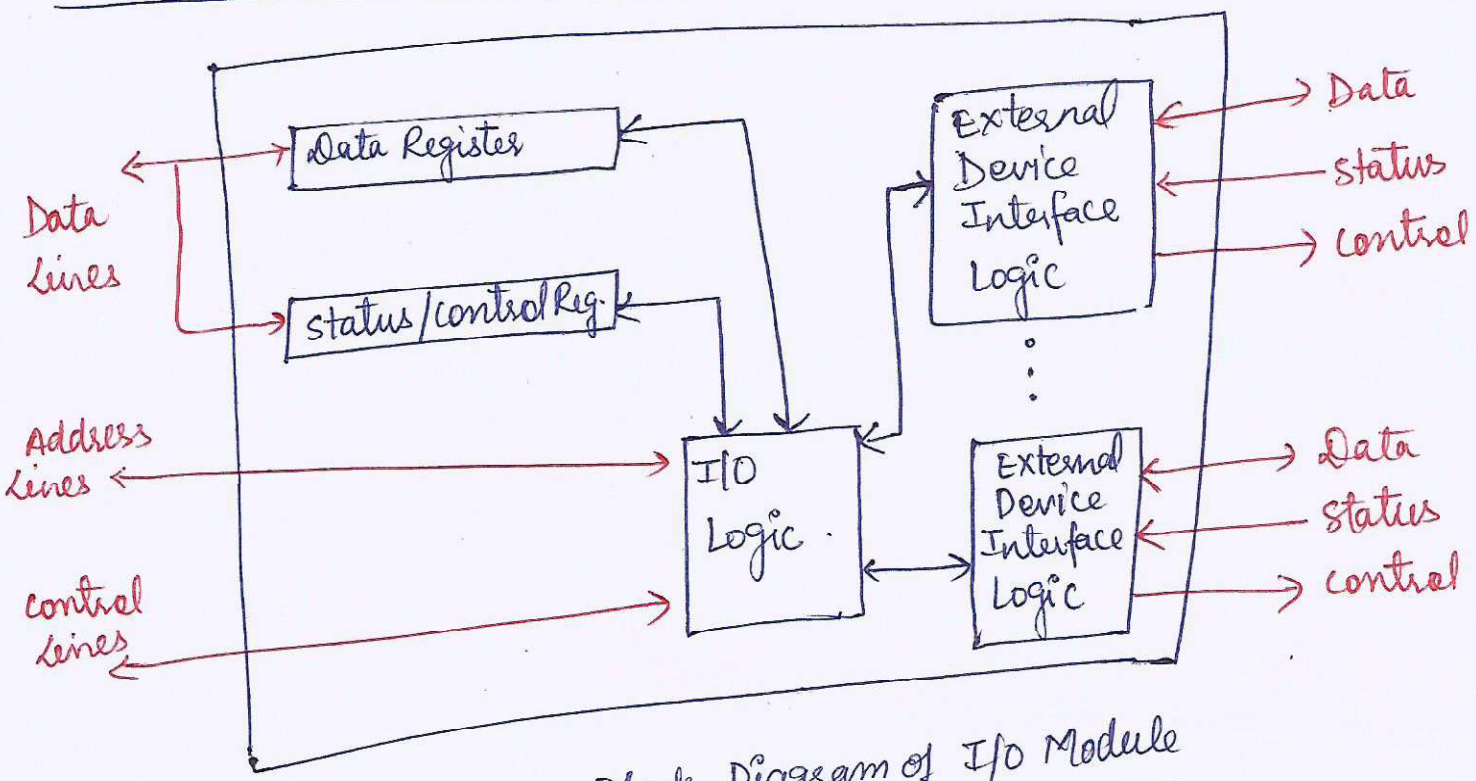
(2) Machine Readable Devices:-
suitable for communicating with equipment. eg. magnetic disk, sensor etc.

(3) communication Device:
suitable for communicating remote devices.
Eg- The remote device can be the human readable device such as terminal, a machine readable device or another computer.



Generic Model of an I/O Module

Block Diagram of I/O Module



Block Diagram of I/O Module

I/O Port: A connection Point that acts as an interface b/w the computer and external devices like mouse, keyboard, printer, moden etc.

Ports are of two types:-

Internal Port: It connects motherboard to internal device like Hard disk drive, internal modem etc.

External Port: It connects the motherboard to external device like, mouse, printer, and keyboard etc.

I/O Systems:

User programs never directly interact with I/O devices. All I/O operations are performed with the help of system calls.

There are 4 types of I/O commands:-

- → Control
- → Test Status
- → Read
- → Write

Modes of Data Transfer (I/O Techniques)

- Programmed I/O
- Interrupt Driven I/O
- Direct Memory Access

Programmed I/O

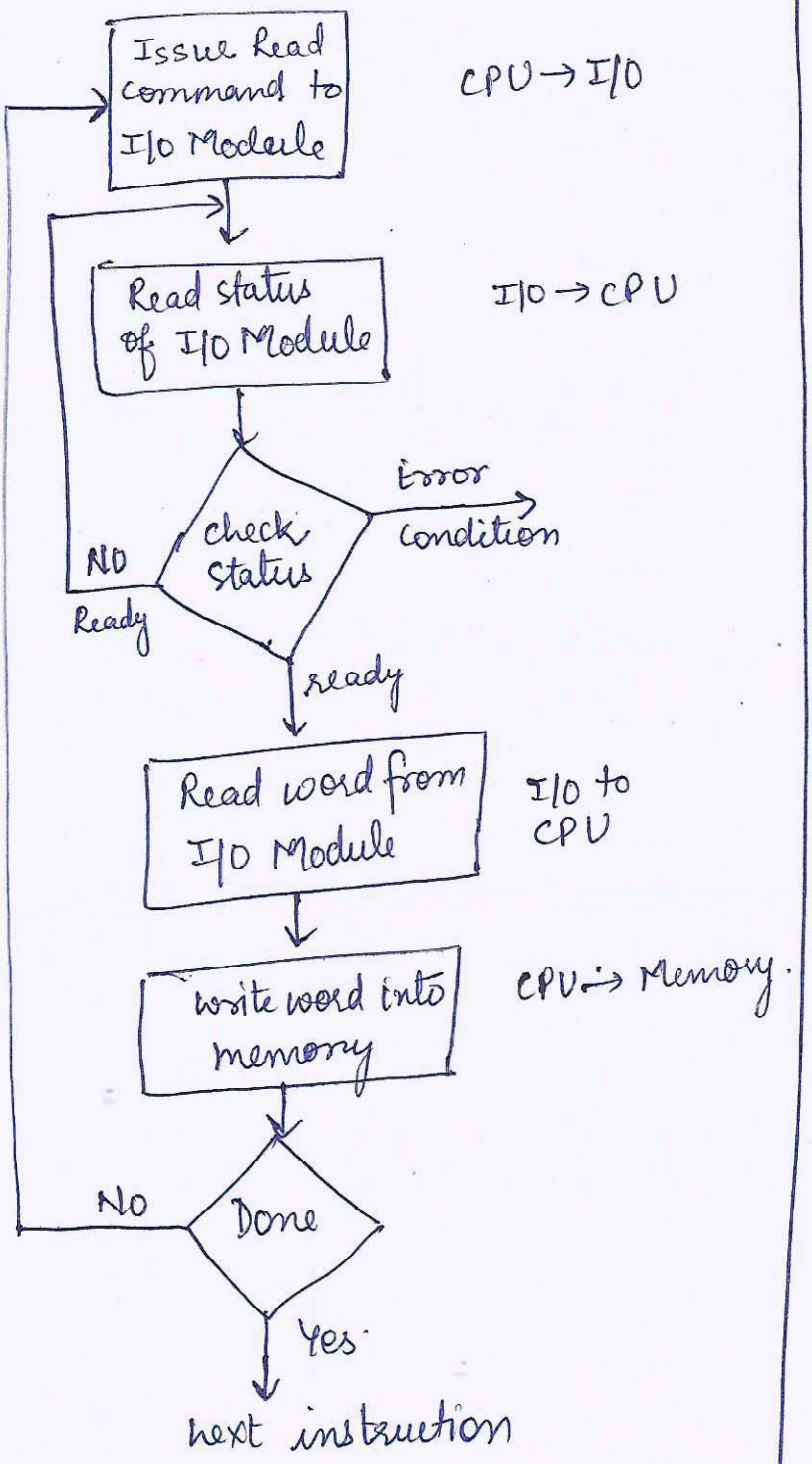
- Input-output instructions written in computer program.
- Each data item transfer is initiated by an instruction in the program
- Transferring data under program control requires a constant monitoring of peripherals by CPU. (once initiated when to transfer)

⇒ ~~Once initiated~~

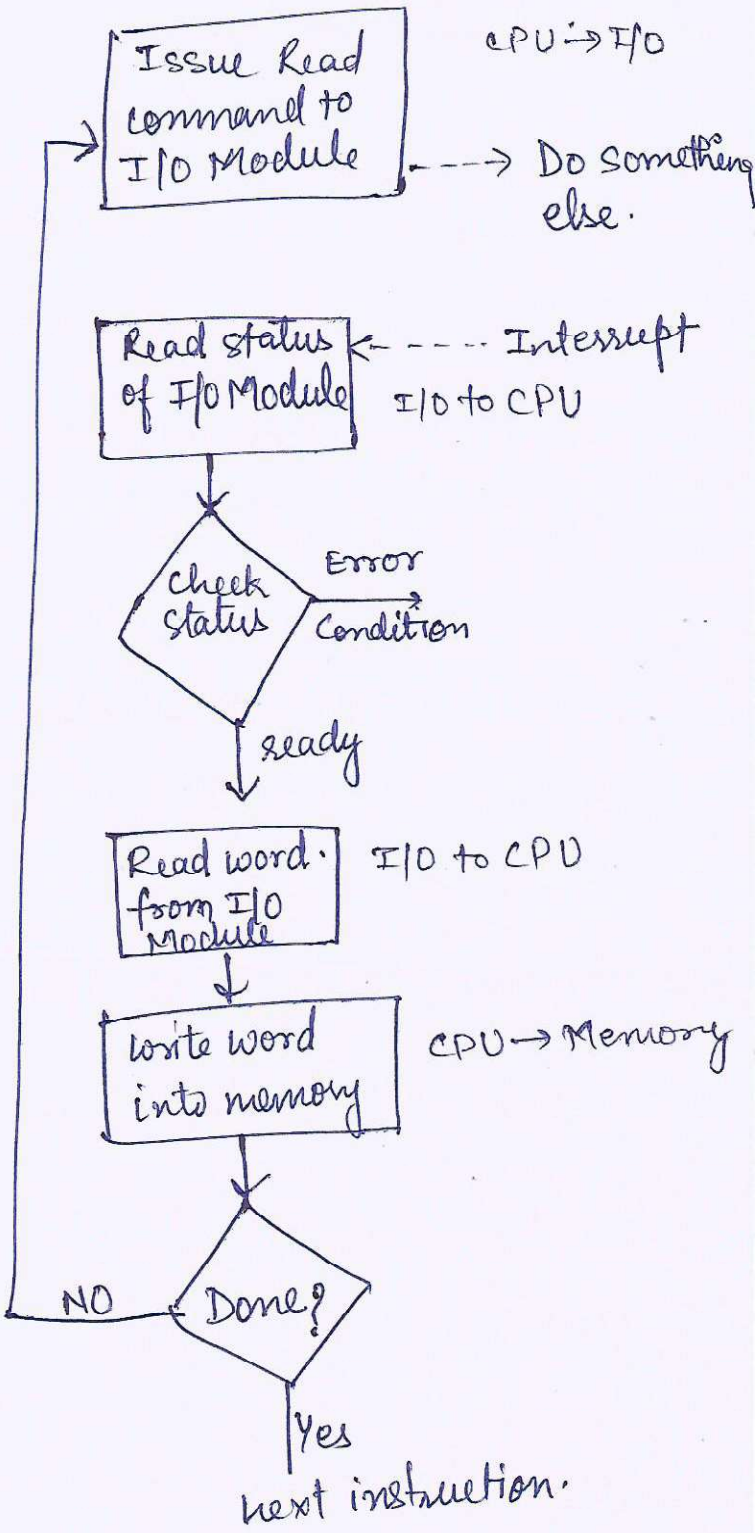
⇒ In this method, the CPU stays in a program loop until I/O unit indicates that it is ready for the data transfer.

⇒ Time consuming

⇒ ~~Less efficient method~~ Less efficiency.



Programmed I/O



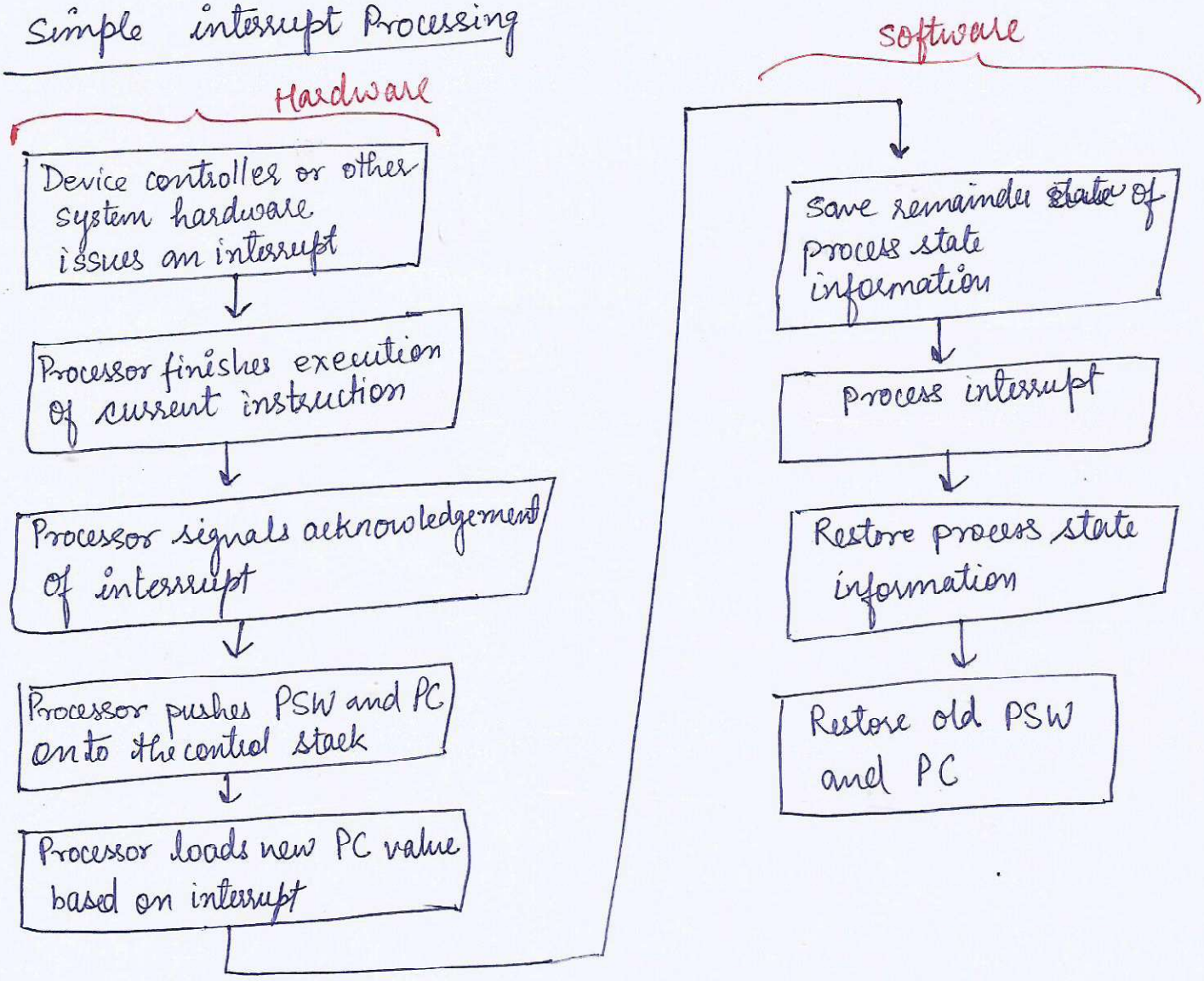
Interrupt Driven I/O

Interrupt Processing:

The basic method of interrupting the CPU is done by activating a control line that connects the interrupt source to the CPU.

CPU ⇒ * recognizes the presence of interrupt
* executes a specific interrupt handling program.
* determines the source of interrupt
* determines the address (branch address) of the interrupt handling program.

Simple interrupt Processing



Design issues:

Two design issues in implementing I/O Interrupt I/O.

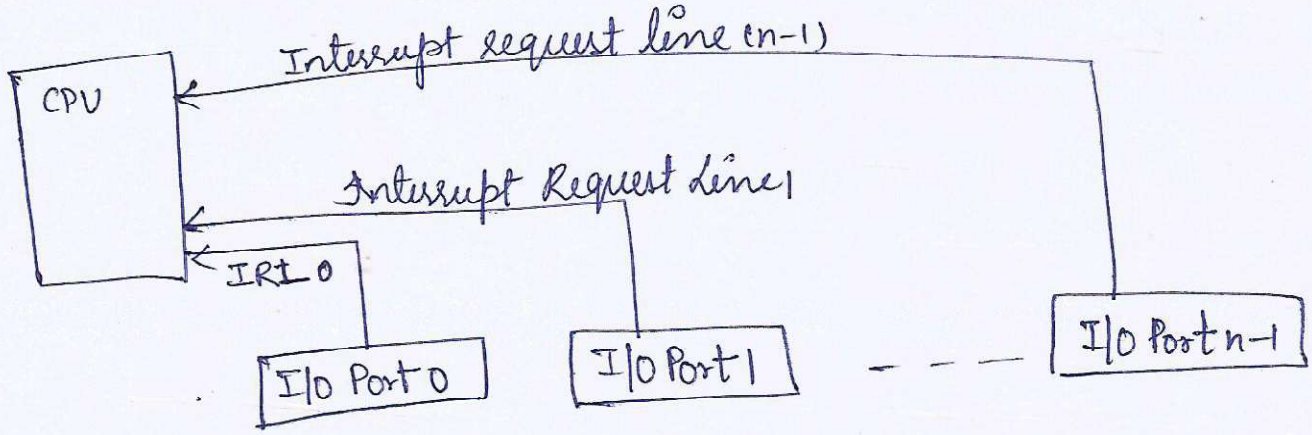
(1) → To determine which device issued the interrupt from multiple devices.

(2) if multiple devices interrupts have occurred, how does the the processor decide which one to process.

Issue: Device identification

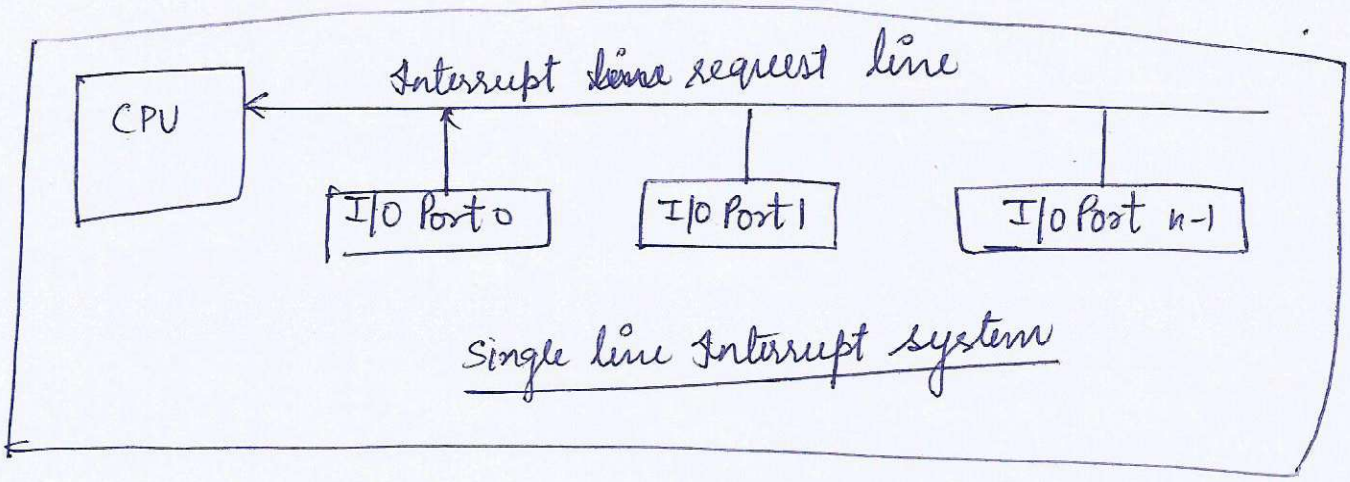
4 Techniques exist.
- → Multiple interrupt lines
- → Software Poll
- → Daisy chain (Hardware Poll, vectored)
- → Bus Arbitration (vectored)

Multiple Interrupt Lines:-



⇒ immediate recognition of device.
⇒ separate Interrupt request line
⇒ impractical approach.

multiple Interrupt line system



Single line Interrupt system

Software Poll:

⇒ when processor detects the interrupt, it branches to an interrupt service-routine whose job is to poll each I/O module to determine which module caused the interrupt.

⇒ Time consuming

⇒ Priority is can be implemented by defining polling sequence.

Vectored Interrupt using Daisy chaining: (Hardware Poll)

⇒ All I/O module shares common interrupt request line

→ The interrupt acknowledgement line is daisychained through the modules.

– when a processor senses the interrupt, it sends out an interrupt acknowledge propagating through a series of I/O module until it gets a requesting module.

– The requesting module responds by placing a word on data lines. This word is ~~address~~ referred to as vector.

Bus Arbitration Technique:- (vectored Interrupts)

→ with bus arbitration technique, an I/O module first gain control of bus before it can raise the interrupt request line.

→ Thus only one module can raise the line at a time.

Types of Interrupts:

Program interrupts

These are generated by some condition that occurs as a result of an instruction execution such as:

- → Arithmetic Overflow
- → Division by zero
- → Execution of illegal machine instruction
- → Segment limit violation
- → Execution of privileged instruction

Timer Interrupts

These are generated within the processor. This allows operating system to perform certain operations on regular basis.

Input-output interrupts:

These are generated for initiation or completion of I/O operations I/O failure or I/O error too can generate an interrupt.

Hardware Failure Interrupts

These are generated by a failure; such as power failure or memory parity error.

Interrupts v.s. Exceptions:

An interrupt is generated by a signal from hardware, and it may occur at random times during execution of a program.

An exception is generated from software, and it is provoked by the execution of an instruction.

Hardware and Software interrupts

→ When microprocessor receive interrupts signals through pins (hardware) of microprocessor. These are known as Hardware interrupts

→ Software interrupts are those which are inserted in the between the program which means these are mnemonics of the microprocessor.

Vectored and non vectored interrupts

- In vectored interrupts, the source (or device) that interrupts supplies the branch information to the computer.
- In a non-vectored interrupt, the branch address is assigned to a fixed location in the memory.

Maskable and non-maskable:

- Maskable interrupts are those which can be disabled or ignored by the microprocessor.
- Non-maskable: which can not be disabled or ignored by the microprocessor.

Types of interrupts:

4 categories
- → Program interrupts
- → Timer interrupts
- → Input/output interrupts
- → Hardware failure

Direct memory Access :-

→ Both interrupt driven and programmed I/O require involvement of CPU for data transfer.

→ CPU can be better utilized for program execution.

→ I/O activities are slow and involvement of CPU will not have a desired effect on the system performance.

→ DMA increases the speed of I/O transfer by eliminating the role played by CPU in I/O operations.

→ when large amount of data is stored/transferred from CPU, a DMA module can be used.

DMA operates in the following ways:

when I/O is requested, the CPU instructs the DMA module about the operation, by providing information:

- → Source address
- → Target address
- → Read / write
- → Number of words to be read or written.

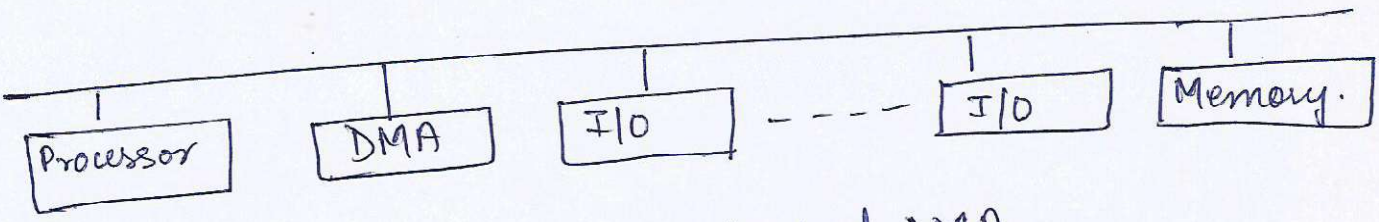
⇒ CPU loads the DMA registers 'Data count' and 'Address Register' with number of bytes to be transferred and starting address memory location respectively.

when the DMA controller is ready to transmit or receive data, it activates the DMA request line to CPU. CPU wait for the next break point. CPU now leaves the control of system bus and activates DMA acknowledge.

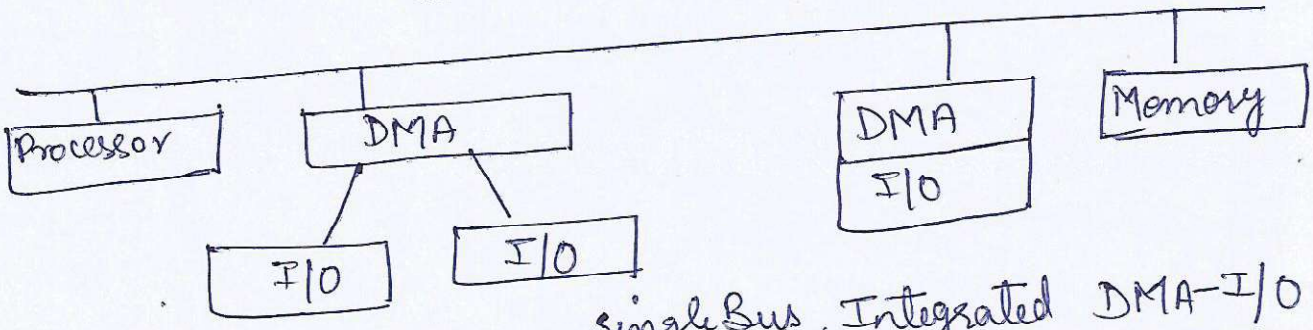
→ The DMA controller transfers the data directly from or to main memory. After, a word is transferred. Data count (or word count register) and Address register are updated.

→ If the data count register has not reached zero but the I/O device is not ready to send or receive the data, the DMA controller releases the system bus to CPU by deactivating DMA request line.

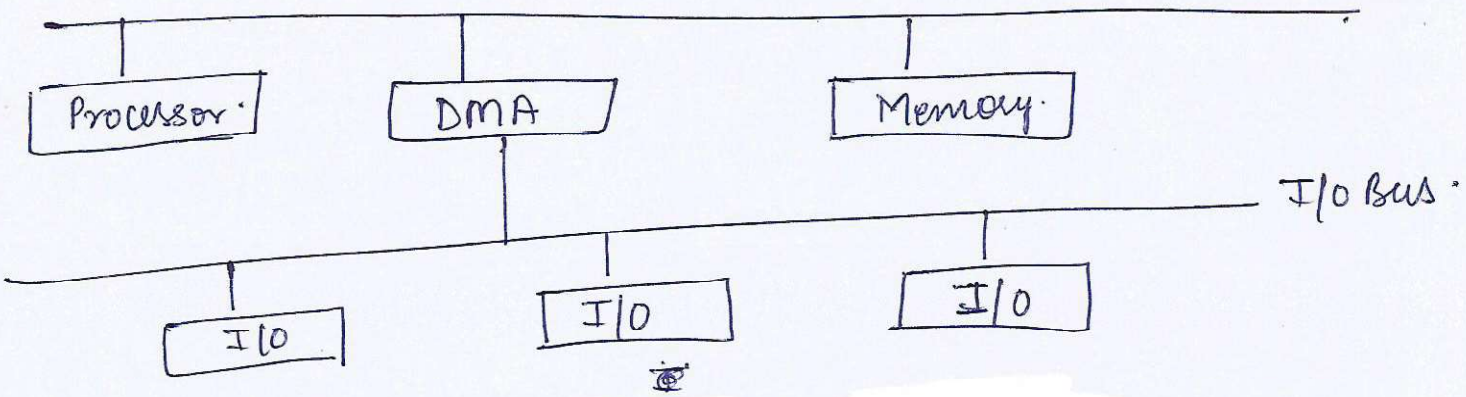
- if data count register is decremented to zero, DMA controller finally relinquishs the control of system bus. It may also send an interrupt signal to CPU to indicate the end of data transfer.



Single Bus Detached DMA

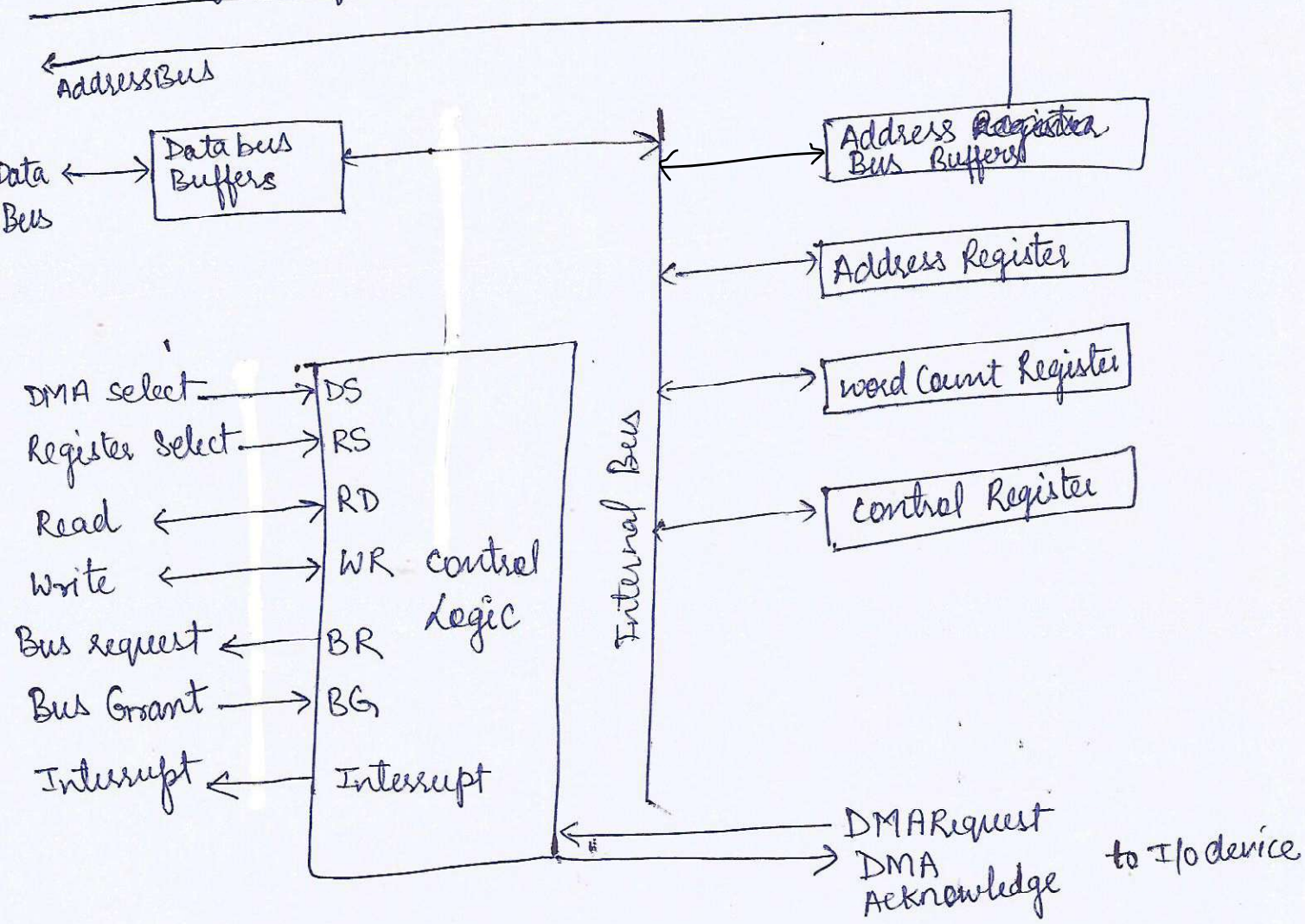


Single Bus, Integrated DMA-I/O



I/O Bus

Block Diagram of DMA



→ Bus Request (BR) input is used ~~to~~ by the DMA controller to request to CPU to leave the control of system bus.

→ The CPU activates the 'Bus Grant' (BG) output to inform the DMA controller.

→ When DMA terminates the transfer, it disable the bus request line. CPU disables the bus grant; takes control of buses and returns to its normal operation.

DMA Data Transfer modes:

→ DMA block transfer
→ Cycle stealing mode
→ Transparent DMA.

(1) DMA Block Transfer:-

→ In DMA block transfer technique, a block of data of arbitrary length can be transfered in a single burst.
→ This DMA mode is needed for secondary memories like disk drivers, where data transmission can not be stopped or ~~stoved~~ slowed without loss of data.

(2) cycle steading Mode:

In cycle stealing mode, the DMA controller is allowed to transfer one word at a time, after which it must return the control of the bus to the CPU.

* DMA module must use the bus only when the processor does not need it, or it must force the processor to suspend operation temporarily.

DMA module transfers a word and returns the control to the processor Note that this is not an interrupt; the processor does not save a context and do something else.
Rather, processor pauses for one bus cycle. The overall effect is to cause the processor to execute more slowly. Nevertheless, for a multiple-word I/O transfer, DMA is far more efficient than interrupt driven or programmed I/O.

(3) Transparent DMA:

In transparent DMA, DMA is allowed to steal only those cycles when the CPU is not using the system bus.
CPU does not require to have control of system bus during Decode instruction or execute instruction phase.

* DMA transfessing data during transparent DMA does not have any adverse effect on CPU performance.

Input - output channels:

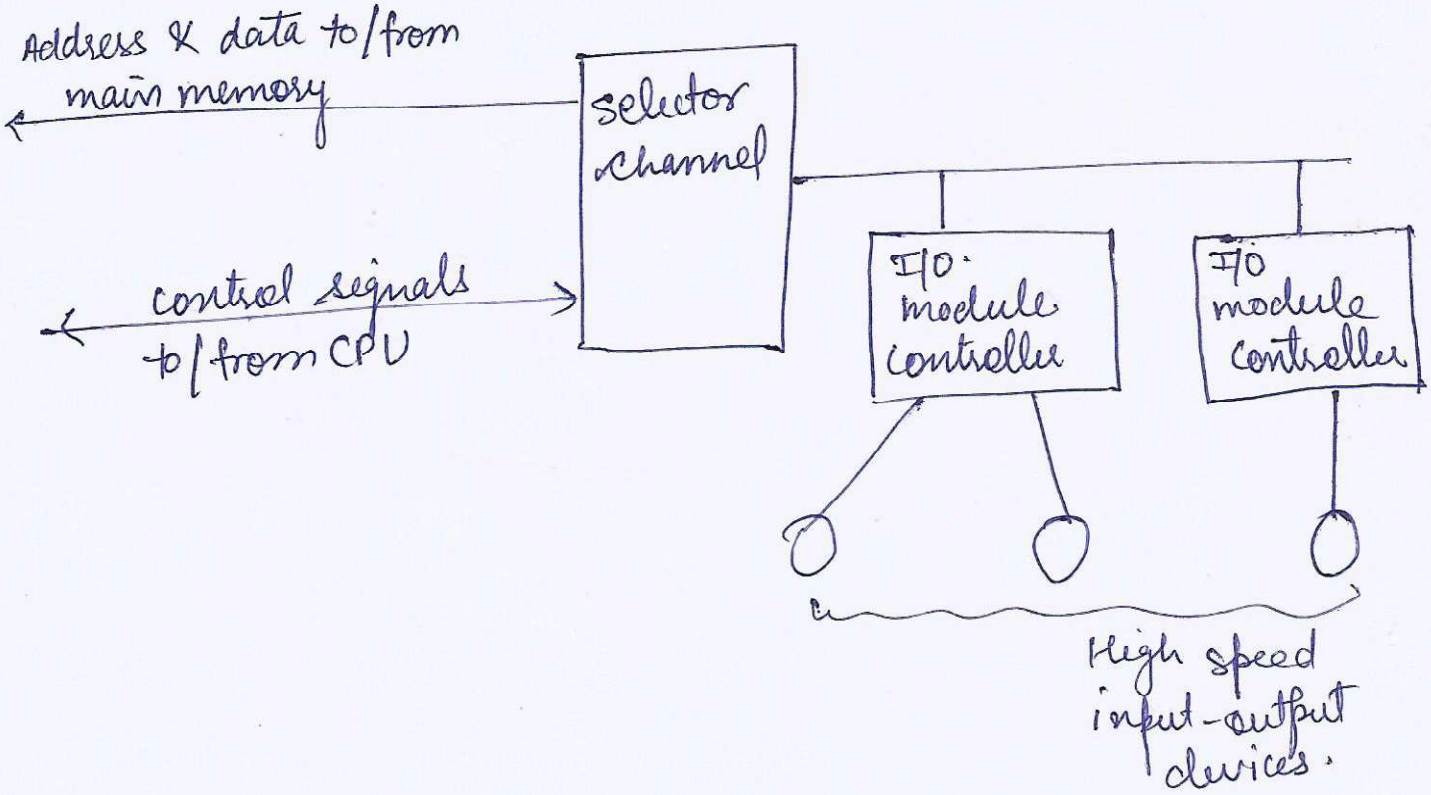
Input-output channels or input-output processors, virtually retrieves CPU from all I/O related activities. It has complete control over device. It can also communicate with CPU.

* I/O processor is a processor like CPU with limited number of instructions. An I/O processor has the ability to execute the I/O instructions It can also carry out the work of data conversion required by the input-output device.

There are two types of I/O channels:

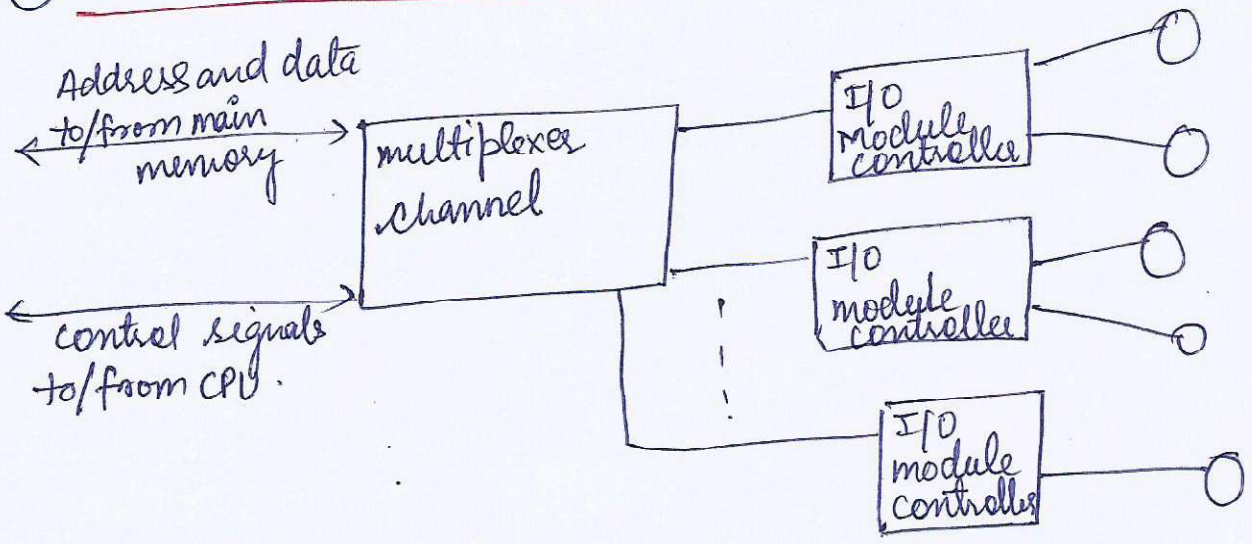
1. Selector channel
2. Multiplexer channel.

(1) Selector channel:



- A selector channel can handle multiple high speed devices. Only one device is selected at a time for communication.
- A selector channel is useful in high speed transfer of data in burst mode.

(2) A multiplexer channel:



A multiplexer channel can handle I/O with a number of devices at the same time. If the device is slow then byte, multiplexing is used.

Example There are three devices which need to send individual bytes are

B1, B2, ----- B5 → data from 1st device.
X1, X2, ----- X5 → data from 2nd device.
Y1, Y2 - --- Y5 → data from 3rd device.

Then on a byte multiplexer channel may send the ~~data~~ bytes as B1 X1 Y1 B2 X2 Y2 B3 X3 Y3 B4 X4 Y4 - ----.

Data Transfer:

- Synchronous
- Asynchronous

Synchronous: If registers in the interface share a common clock with CPU registers, the transfer between two units is said to be synchronous.

Asynchronous: The internal timing in each unit is independent from the other in that each uses its own private clock for internal registers, in this case the two units are said to be asynchronous.

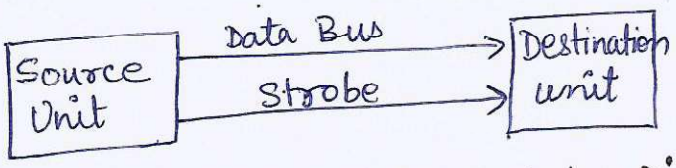
Asynchronous Data Transfer:

→ two independent units transmit control signals to indicate the time at which data is being transmitted.

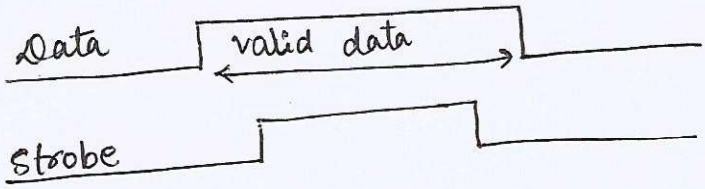
→ Two methods
- Strobe
- Handshaking

Strobed Method (one way) One of communicating device supplies all the control signals.

- Strobe can be activated by either source or destination



Source iniated Strobe Diagram



Timing Diagram

The strobe is a single line that informs the destination unit when a valid data word is available in the bus.

* The source unit first places the data on the data bus. After a brief delay to ensure that the data settle a steady value, the source activates the strobe pulse.

* In destination initiated data transfer, destination unit activates the strobe pulse, informing the source to provide the data.

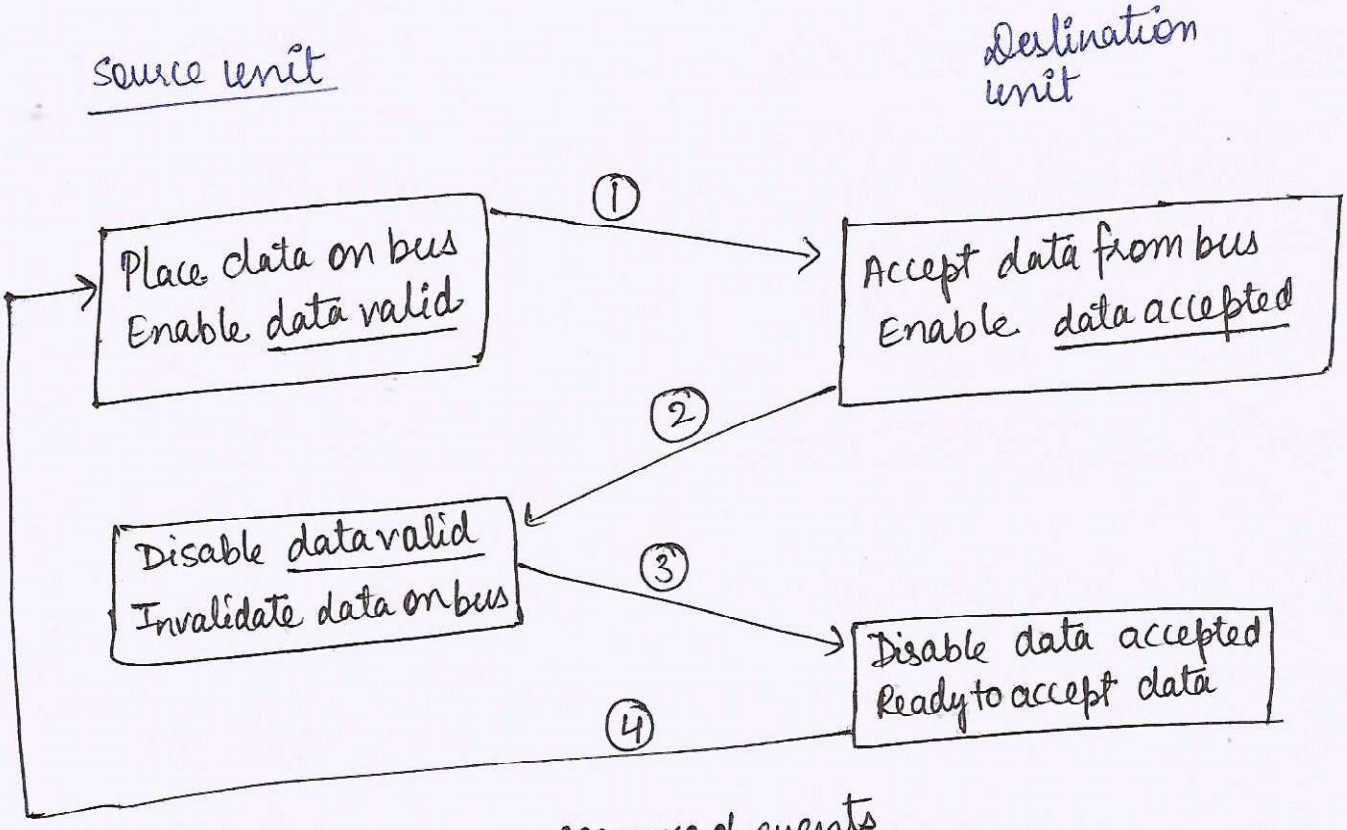
Disadvantage: In strobe method, the source unit that initiates the transfer has no way of knowing whether the destination unit has actually received the data item.

Handshaking: (Two way)

* The disadvantage of oneway control is that it does not verify that the data transfer has been completed successfully. Data may be lost.

* Control of data transfer is based on the use of handshake between processor and the device selected for input-output.

Block Diagram



sequence of events

Serial communication

or

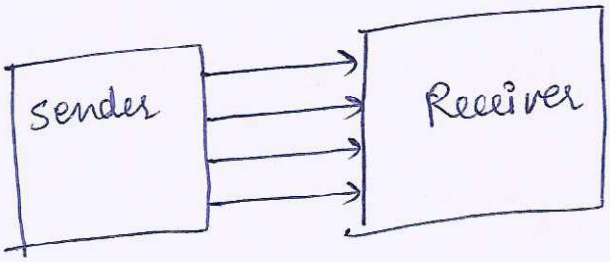
Serial Transmission

⇒ Serial data transmission occurs by transmitting data one bit at a time in sequential order over a bus.

start bit data bits stop bits ⇒ To know the start and end of transmission.

Parallel Transmission

In parallel transmission several bits can be transmitted at the same time.



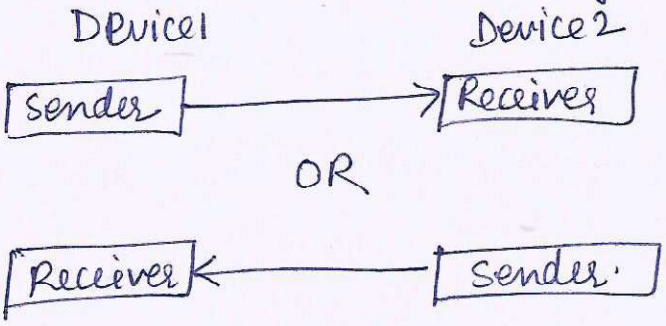
Serial Communication can be
- Simplex
- Half Duplex
- Full Duplex

Simplex Transmission: Signals travels in one direction only.
eg. Keyboard

Half Duplex:

→ capable of sending signals in both the directions, but in only one direction at a time

eg ~~Telephone~~ Walkie Talkie.



Full Duplex: (Bidirectional Transmission)

→ This method allows signal transmission in both the directions simultaneously.

eg Telephone IP service.