Unit 5

• Peripheral Devices

→ connected to comp & controlled by processor Eg. Keyboard, Printer

→ under direct control of computer

→ designed to read info into or out of the memory unit

→ part of total computer system

→ Types (i) Input (iii) Input-Output
(ii) Output

• Input / Output Interface

→ provides method for transferring information b/w internal storage and external I/O devices

→ A hardware unit that plays an imp role b/w CPU and peripheral devices to supervise and synchronise input and output data or tasks.

→ Why we need it?

(i) Data transfer rate of peripheral are usually slower than CPU

(ii) Data codes and formats of peripheral devices differ from the word format of CPU & memory

(iii) The operating modes of peripheral are different from other peripherals and each Peripherals must be controlled such that others ~~do~~ ~~not~~ connected to CPU do not get disturbed.

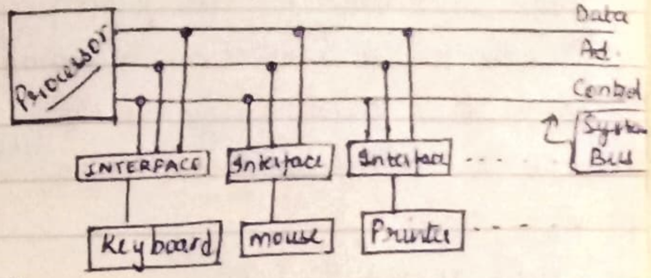
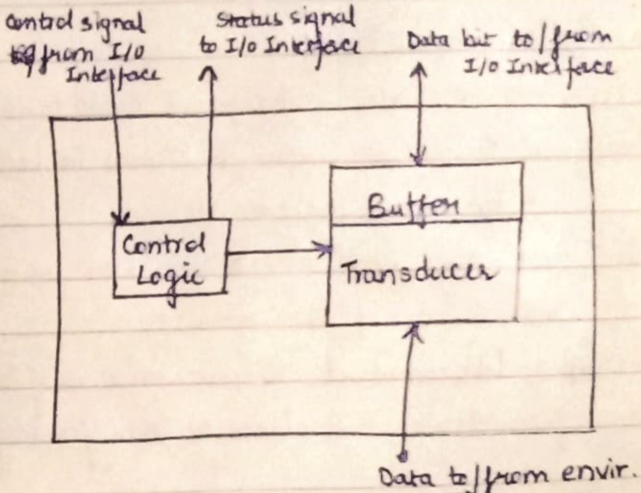


fig Connection of I/O bus to I/O Device
or
General model of I/O Interface


control signal ~~to~~ from I/O Interface
Status signal to I/O Interface
Data bit to/from I/O Interface
Buffer
Control Logic
Transducer
Data to/from envir.

Block Diagram
External Device

→ Functions of I/O Interface

(i) Control and timing

(ii) Processor communication

(iii) Device communication

(iv) Data buffering

(v) Error Detection

• During period of time, the processor may communicate with one/more devices in unpredictable pattern, depending on program's I/O needs.

- **Modes of Data Transfer**

→ Binary Info received from a device is stored in memory for later processing

→ CPU executes the instructions and may accept data temporarily, but the ultimate destination of data is memory unit.

→ Types

(i) Programmed I/O

(ii) Interrupt initiated I/O

(iii) DMA (Direct Memory acess)

- **I/O Command**

(i) Control : Activates and informs what to do

(ii) Status : Test various status conditions in both interface and peripherals

(iii) Output data Command Transfer data from bus to one of its registers

(iv) Input data command : receives an item of data from device and places it in buffer reg.

- **I/O Bus**

→ Consists of

(i) Data lines

(ii) Address lines

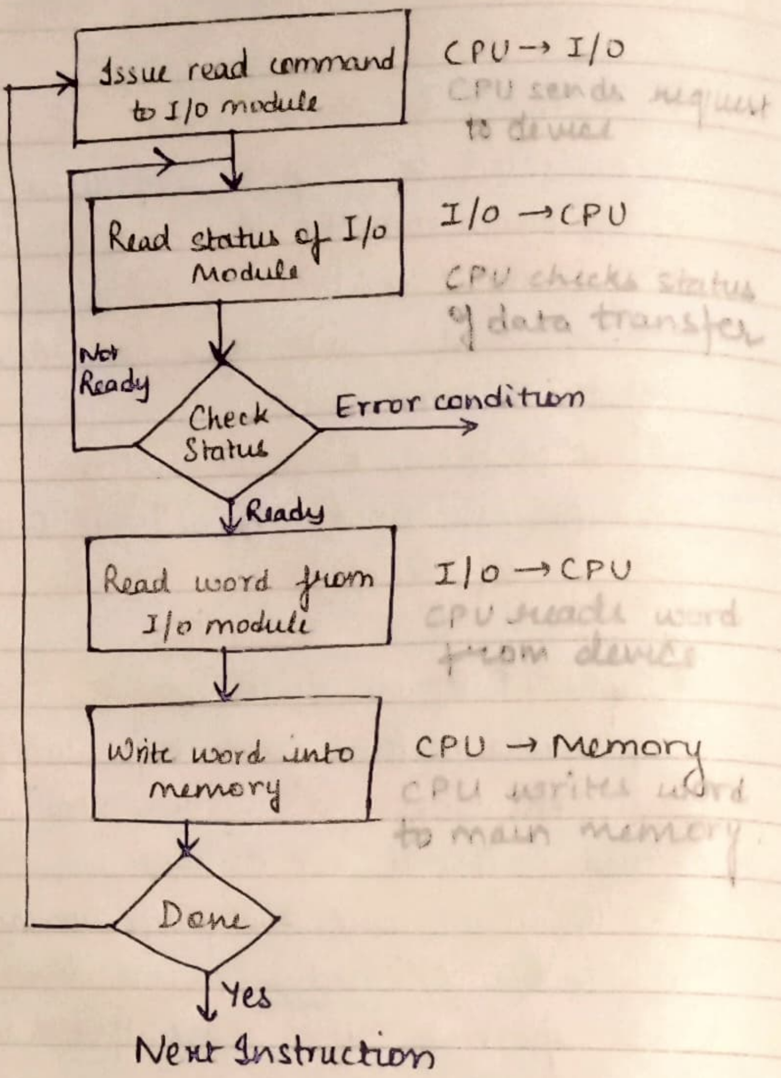
(iii) Control lines

→ communication link b/w processor & device

→ I/O bus from processor is attached to all peripheral interface.

→ Comprises of mag. tapes, disks, printers & terminals.

(i) Programmed I/O


Issue read command to I/O module
CPU → I/O
CPU sends request to device
Read status of I/O Module
I/O → CPU
CPU checks status of data transfer
Not Ready
Check Status
Error condition
Ready
Read word from I/O module
I/O → CPU
CPU reads word from device
Write word into memory
CPU → Memory
CPU writes word to main memory
Done
Yes
Next Instruction

- Used when data is to be transferred b/w CPU and I/O.
- Explanation → I/O devices : no direct access to memory

→ In Programmed I/O, CPU makes a request and then CPU stays in program loop (called Polling) untill the I/O device indicates that it is ready for data transfer

→ Since, I/O device takes no further action to alert CPU (ie it doesn't interrupt CPU) to Tell that data transfer can now be done.

→ Thus, transferring data under program

control requires constant monitoring of I/O device by the CPU.

→ This is a time consuming process & keeps the processor busy needlessly.

⊛ Programmed I/O operations are result of I/O ~~p~~ instructions written in form of computer program. Each data transfer is initiated by an I/O instruction in the program (to access register or memory on a device)

→ The instructions written in program basically initiates the data transfer to & from CPU register & I/O devices

→ Prog. I/O requires that all I/O operatns to be executed under direct control of CPU.

→ Useful in small, low speed systems where hardware costs is minimized.

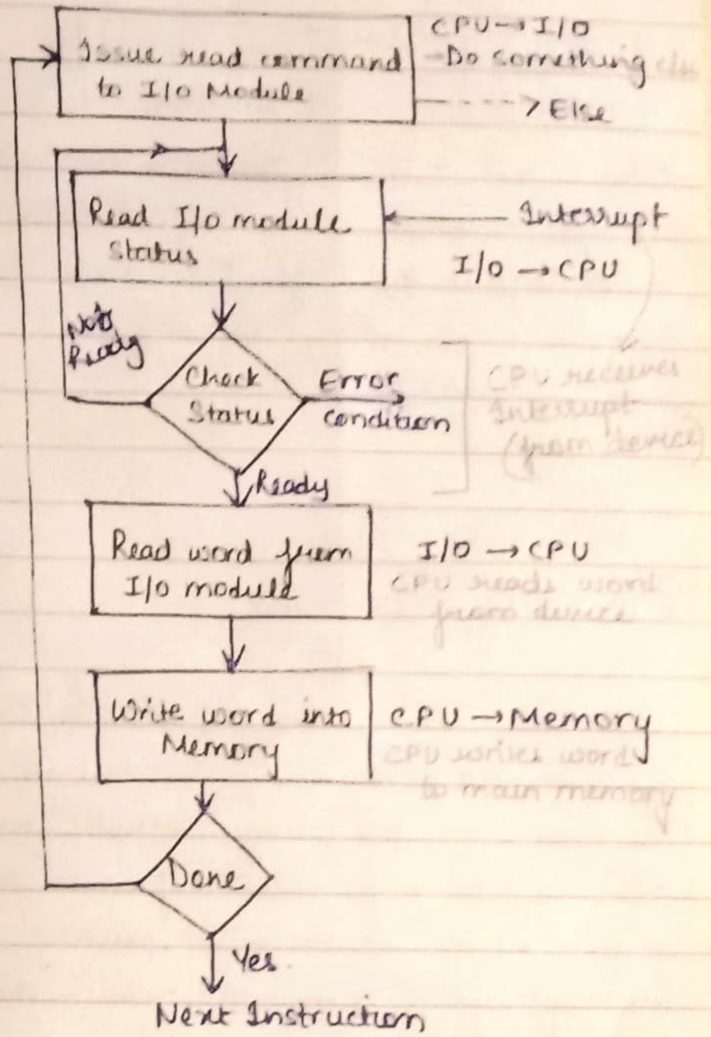
(ii) Interrupted-Initiated I/O

- Why → Prog. I/O: The processor had to wait a long time for I/O Module of concern to be ready for data transfer.
  - → The processor also had to wait & had to repeatedly interrogate the status of data transfer.
  - → This degraded the performance of entire system.

Thus, we use interrupted-Initiated I/O as it eliminates needless waiting

- **Drawback**: still consumes a lot of processor time as every word from I/O to memory or from memory to I/O must pass through processor

- Flowchart



Explanation

- In Interrupted I/O, instead of continuous monitoring of CPU, interface will be informed to issue an Interrupt request signal, when data are available from device.
- Meanwhile CPU proceeds to execute another program & interface keeps monitoring the device
- When device is ready for data transfer it generates interupt signal / request.
- Upon detecting the external interrupt signal, CPU stops the task it is performing, process the I/O data-transfer & the resumes the original task it was performing

- Types of Interrupt

Vector Interrupt

→ The source that interrupts, suppliesthe branch info to the computer. This info is called interrupt Vector.

Non Vector Interrupt

→ The branch address is assigned to a fixed location in memory.

(*) Priority Interrupt

→ Since, there are number of I/o device that are connected to the computer And all are capable of producing interrupt signal

→ When the interrupt is generated from more than 1 device simultaneously, priority interrupt system is used to determine which device is to be serviced first

"A priority interrupt is a system that establishes a priority over various sources (Interrupt) to determine which condition is to be serviced first when 2 or more requests arrives simultaneously"

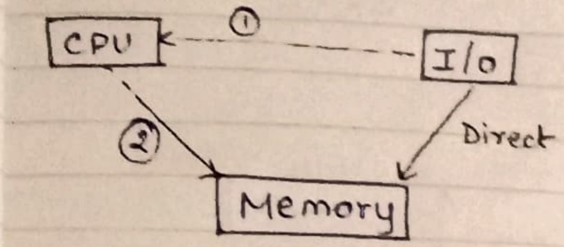
→ Device with high speed → high priority

Eg Hard disks

→ Devices (Slow devices) with low speed → low priority

Eg Keyboard

ii) DMA (Direct Memory Access)



- for I/o device to access memory
  → 2 cycles ① I/o to CPU
  ② CPU to memory.
- To make it less time consuming
  → I/o to Memory
  we use DMA

So to transfer large blocks of data at high speed b/w I/o device & main-memory we use DMA aproach

DMA allows data transfer directly b/w I/o device & main memory with minimal intervention of CPU.

- CPU grants I/o interface authority to read from or write to memory without CPU's involvement.
- DMA itself controls data transfer b/w main memory & I/o device.
- CPU is only involved in beginning & end of data transfer b/w main memory & I/o device, and interrupted only after ~~the~~ entire block has been transfered.

① Trasfer of data b/w a fast storage devic such as mag. tapes and memory is limited by speed of CPU

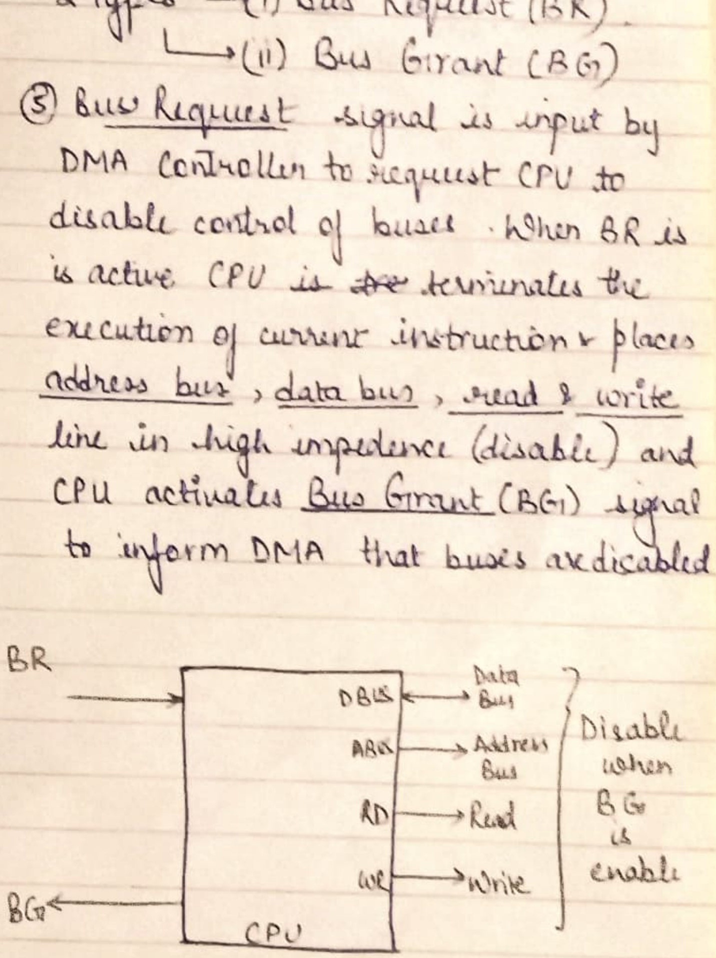
② Removing CPU from the path the device manage the memory bus directly, thus increase data transfer speed.
This technique is DMA.

③ During DMA transfer, CPU is idle & has no control of memory bus. A DMA controller takes over the buses to manage the transfer directly b/w the I/O device & memory.

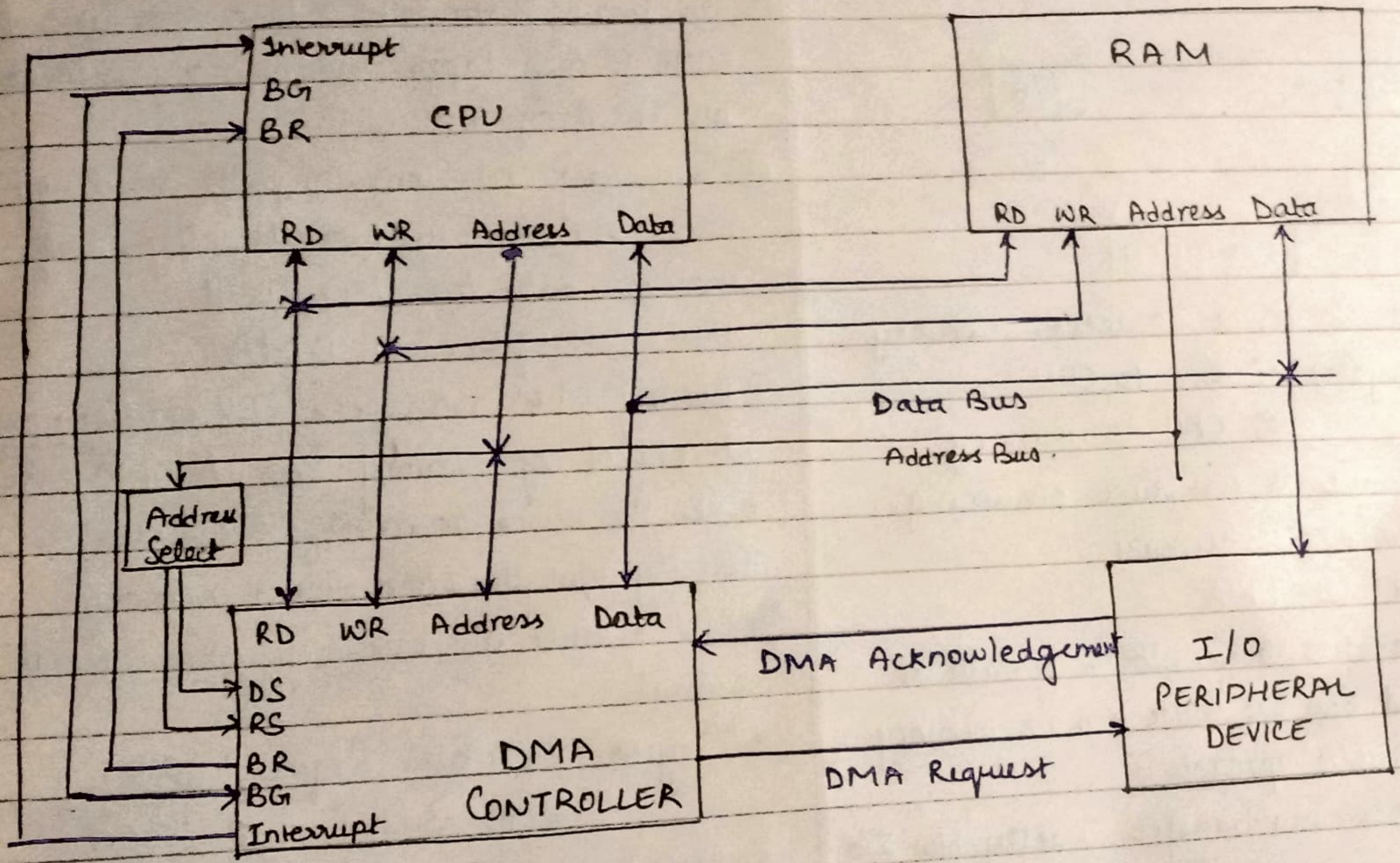
④ CPU disable the bus by some special signal.

- 2 types → (i) Bus Request (BR).
  → (ii) Bus Grant (BG)

⑤ Bus Request signal is input by DMA Controller to request CPU to disable control of buses. When BR is is active CPU is ~~are~~ terminates the execution of current instruction & places address bus, data bus, read & write line in high impedence (disable) and CPU activates Bus Grant (BG) signal to inform DMA that buses are dicabled



(CPU bus Signal for ~~Bus~~ DMA Transfer)



DMA Transfer in Computer System.

- **Serial Communication**

→ refers to device that cannot handle more than 1 bit of data at a given time.

→ requires only 1 wire and is very slow

- Usually CPU use parallel communication

→ If device is serial then data is converted to use parallel communication, by attaching more than on wire parallely.

- Types

① **Synchronous serial communication**

- 2 units share a common clock frequency
- Bits are transmitted continuously at a rate dictated by the clock pulse
- Long distance → each unit is driven by separate clock of same frequency.
- Sync. signals are transmitted periodically b/w the two units to keep their clocks in step with each other
- Data transfer takes place in blocks.

② **Asynchronous Serial Communication**

- Transmitter & receiver are not sync. by clock ie frequency differs.
- Bits are transmitted at a constant rate
- Data transfer is character oriented.
- A special bit is transmitted at both ends.

1) Start bit → always 0
   ↳ Indicates beginning of character

2) Character bit → Stores data

3) End bit → always 1
   ↳ Indicates ending of character