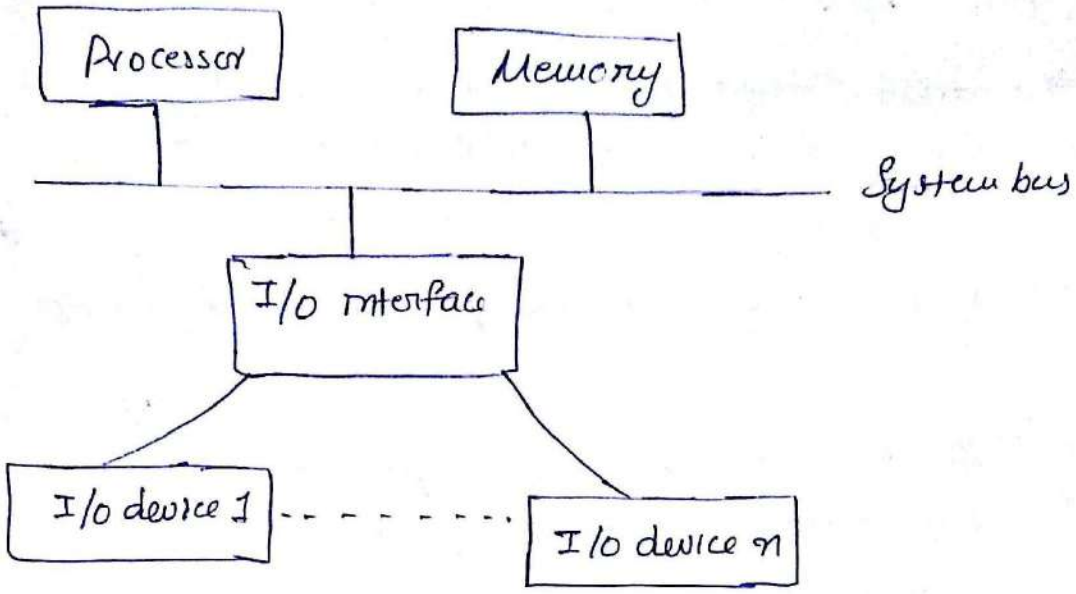


Requirements of I/O interface :-

The I/O interface is nothing but the hardware required to connect an I/O device to the bus. It is also called I/O system. The requirements of an I/O interface are :

- → Control and timing
- → Processor communication
- → Device communication
- → Data buffering
- → Error detection.

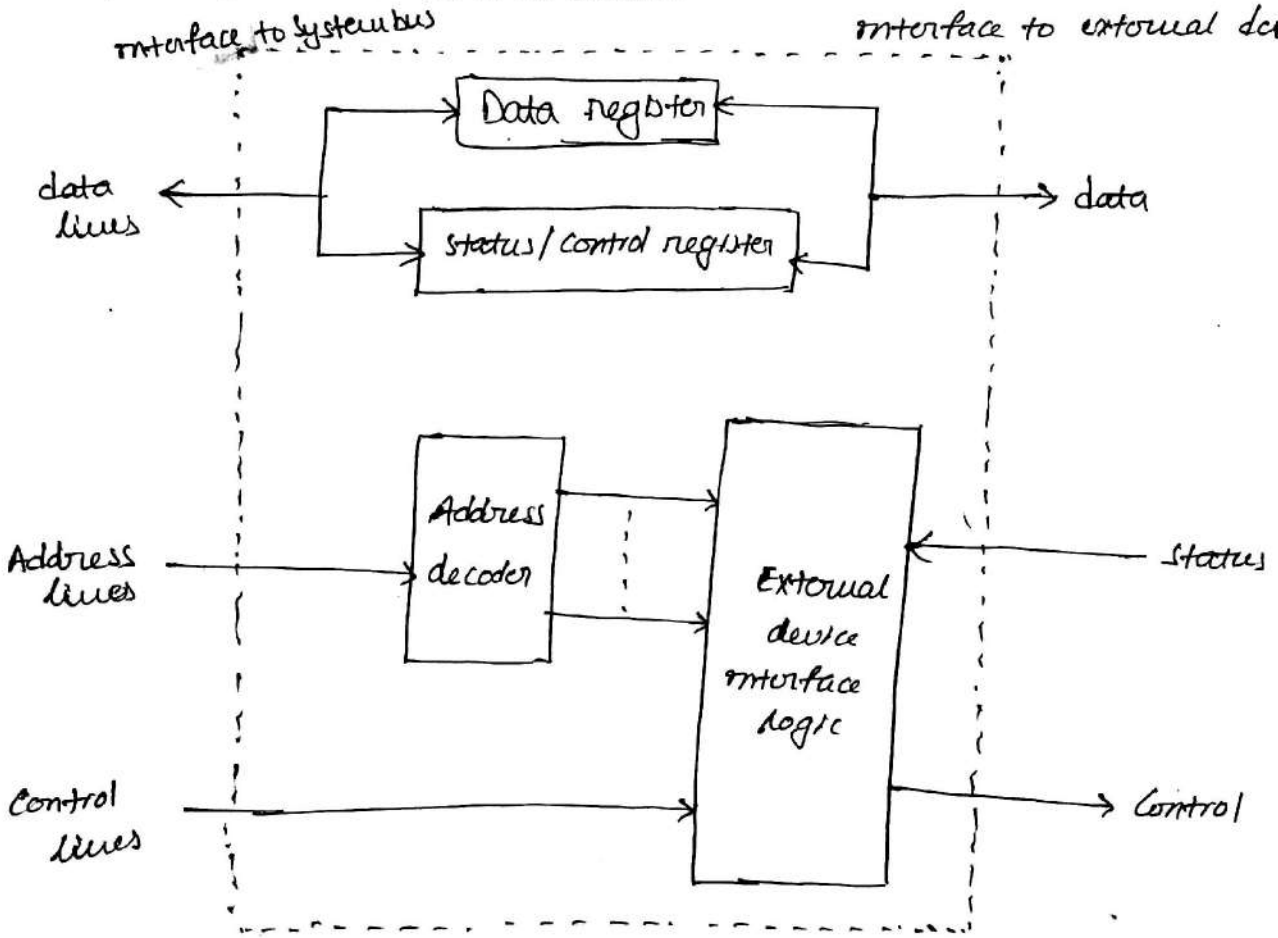
The I/O interface includes a control and timing requirements to coordinate the flow of traffic between internal resources and external devices. Processor communication involves different types of signals transfer such as :

- → Processor sends commands to the I/O system which are generally the control signals on the control bus.
- → Exchange of data between the processor and I/O interface over the data bus.

The I/O interface must able to perform device communication which involves commands, status information and data

Data buffering is also an essential task of an I/O interface. Data transfer rates of peripheral devices are quite high than that of processor and memory. The data coming from memory or processor are sent to an I/O interface, buffered and then sent to the peripheral device at it's data rate.

I/O interface is also responsible for error detection and for reporting errors to the processor. The different types of errors are mechanical, electrical malfunctions reported by the device such as bad disk track, unintentional changes to the bit patterns, transmission errors.

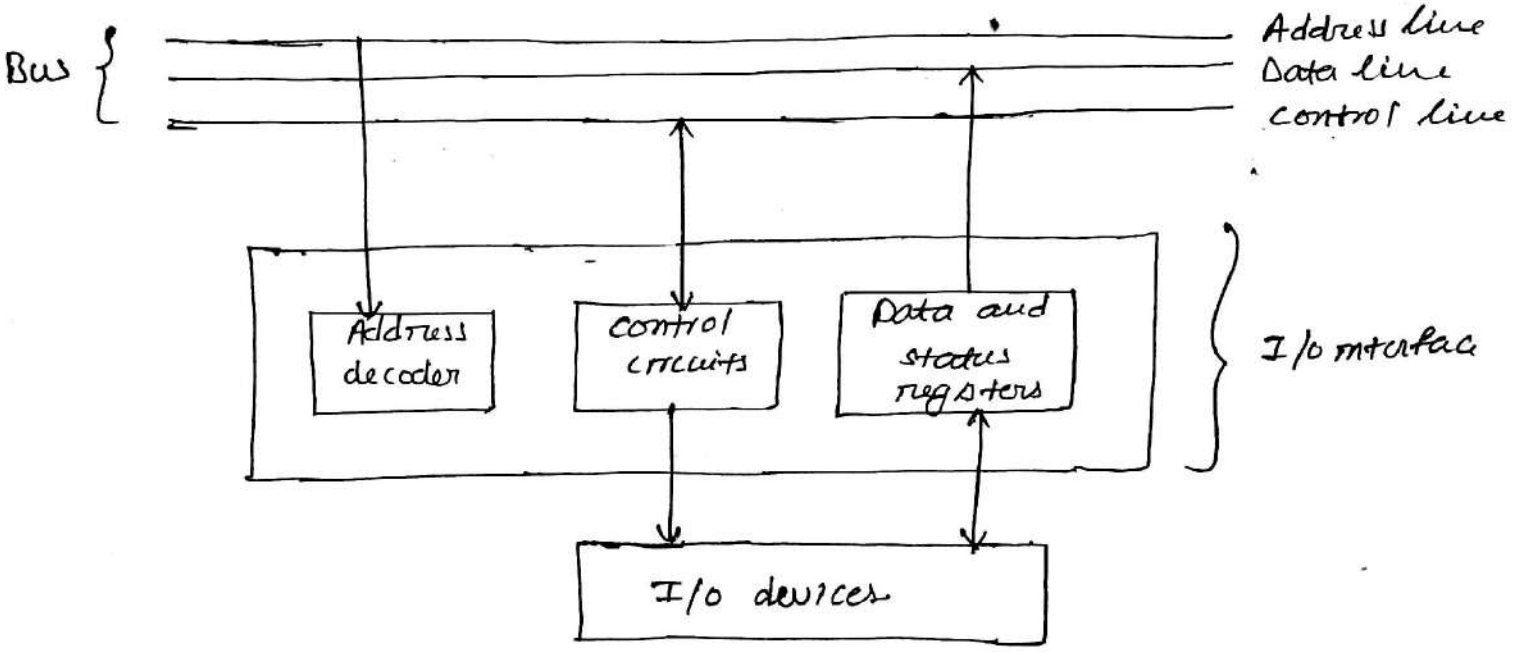


I/O interface

I/O interface consists of data register, status / control register, address decoder and external device interface logic. The data registers holds the data being transferred to or from the processor.

The status / control register contains information related to the operation of the I/O device. Both data and status / control registers are connected to the data bus. Address line is connected to address decoder. The address decoder enables the device to recognize it's address when address appears on the address line.

We can draw the I/O interface for input device and output device like below.



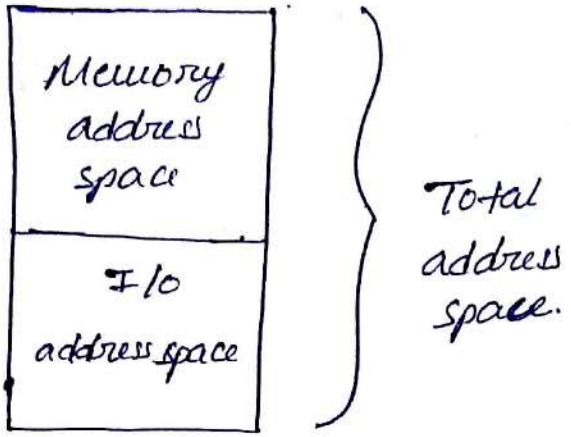
I/O interfacing Techniques :-

The most of the processors support isolated I/O system. It partitions memory from I/O, via software, by having instructions that specifically access memory and others that specifically access I/O.

I/O devices can be interfaced to a computer system I/O in two ways, which are called interfacing techniques. They are

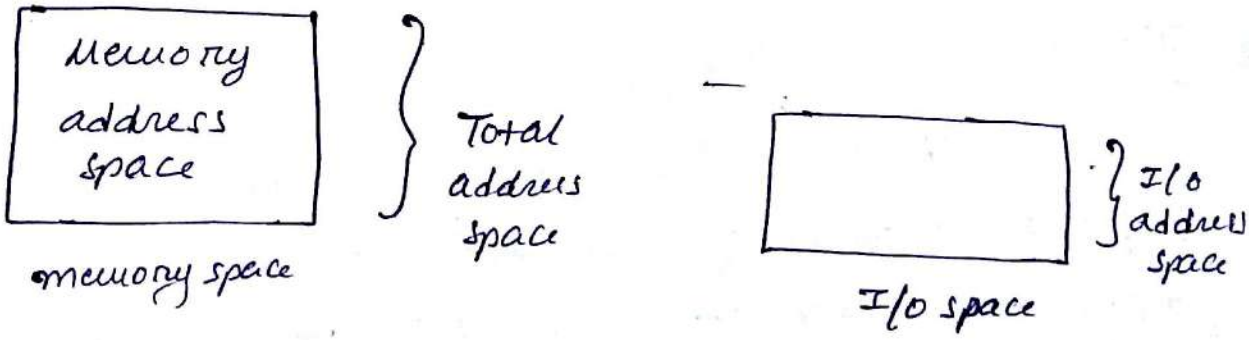
- Memory mapped I/O
- I/O mapped I/O (Isolated I/O).

→ In memory mapped I/O the total memory address space is partitioned and part of this space is given to I/O addressing as.



When this technique is used, a memory reference instruction that causes data to be fetched from or stored at address specified automatically becomes an I/O instruction if that address is made the address of an I/O port.

→ If we don't want to reduce the memory address space we allot a different I/O address space, apart from total memory space which is called I/O mapped I/O or Isolated I/O



Here the advantage is that the full memory address space is available. Processor can only use this mode if it has special instructions for I/O related operations such as I/O read, I/O write.

Memory mapped I/O	I/O mapped I/O
① Memory and I/O share the same address pace of processor.	① processor provides separate address space for memory and I/O.
② Usually, processor provides more address lines for accessing memory. So more decoding is required.	② Usually processor provides less address lines for accessing I/O. So less decoding is required.
③ Memory control signals are used to control read and write I/O operations.	③ I/O control signals are used to control read and write I/O operations

Two basic schemas are used for communication on the bus between processor and I/O devices. They are

- Synchronous.
- Asynchronous.

→ If both CPU and I/O interface share common clock b/w them, the data transfer between them is said to be "Synchronous."

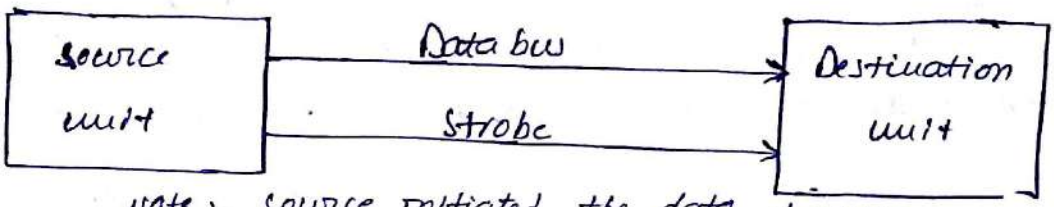
→ If both CPU and I/O interface uses it's own private clock, then the data transfer between them is "asynchronous."

In asynchronous data transfer control signals are used to assist the data transfer. Two methods are used to transfer data between two independent units. In asynchronous way. They are.

① Strobe
② Handshake

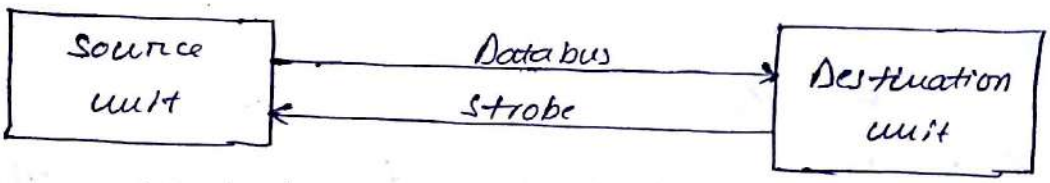
Strobe Control :-

In the strobe control asynchronous data transfer, a single control line is used to indicate to the other unit when the transfer has to occur. The strobe signal may be activated by either the source or the data destination unit.



Note :- Source initiated the data transfer.

This strobe signal informs the destination unit that the valid data is available on the data bus from the source unit.



Note: Destination initiated the data transfer

Hand shaking :-

The disadvantage of strobe method is that the source unit that initiates the transfer has no idea whether the destination unit has actually accepted the data sent by it. Similarly, a destination unit that initiates the transfer has no idea whether the source unit has actually placed the data on the bus.

The handshake method solves this problem by introducing one more signal called acknowledge signal. Three special control lines are used for handshaking method. They are Read Request (Read Req), Data Ready (Data Rdy), Acknowledgement (Ack).

Read Req :- Used to indicate a read request for memory. The address is put on data lines at the same time

Data Rdy :- Used to indicate that the data word is now ready on the datalines, asserted by output/memory and input/I/O devices.

Ack :- Used to acknowledge the Read req or DataRdy signal of the other party.

Modes of Transfer :-

Data transfer to and from peripherals may be handled in one of three possible modes :

① Programmed I/O

② Interrupt driven I/O

③ Direct Memory Access (DMA)

	No interrupts	Use of interrupts
I/O to memory transfer through processor	Programmed I/O.	Interrupt – driven I/O
Direct I/O to memory transfer		Direct memory access (DMA)

Programmed I/O :-

With programmed I/O data are exchanged between the processor and I/O module. The ~~program~~ 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. With programmed I/O, the I/O module will perform the requested action and then set the appropriate bits in the I/O status register.

To execute an I/O related instruction, the processor issues an address, specifying the particular I/O module and external device, and an I/O command. There are four types of I/O commands that an I/O module may receive.

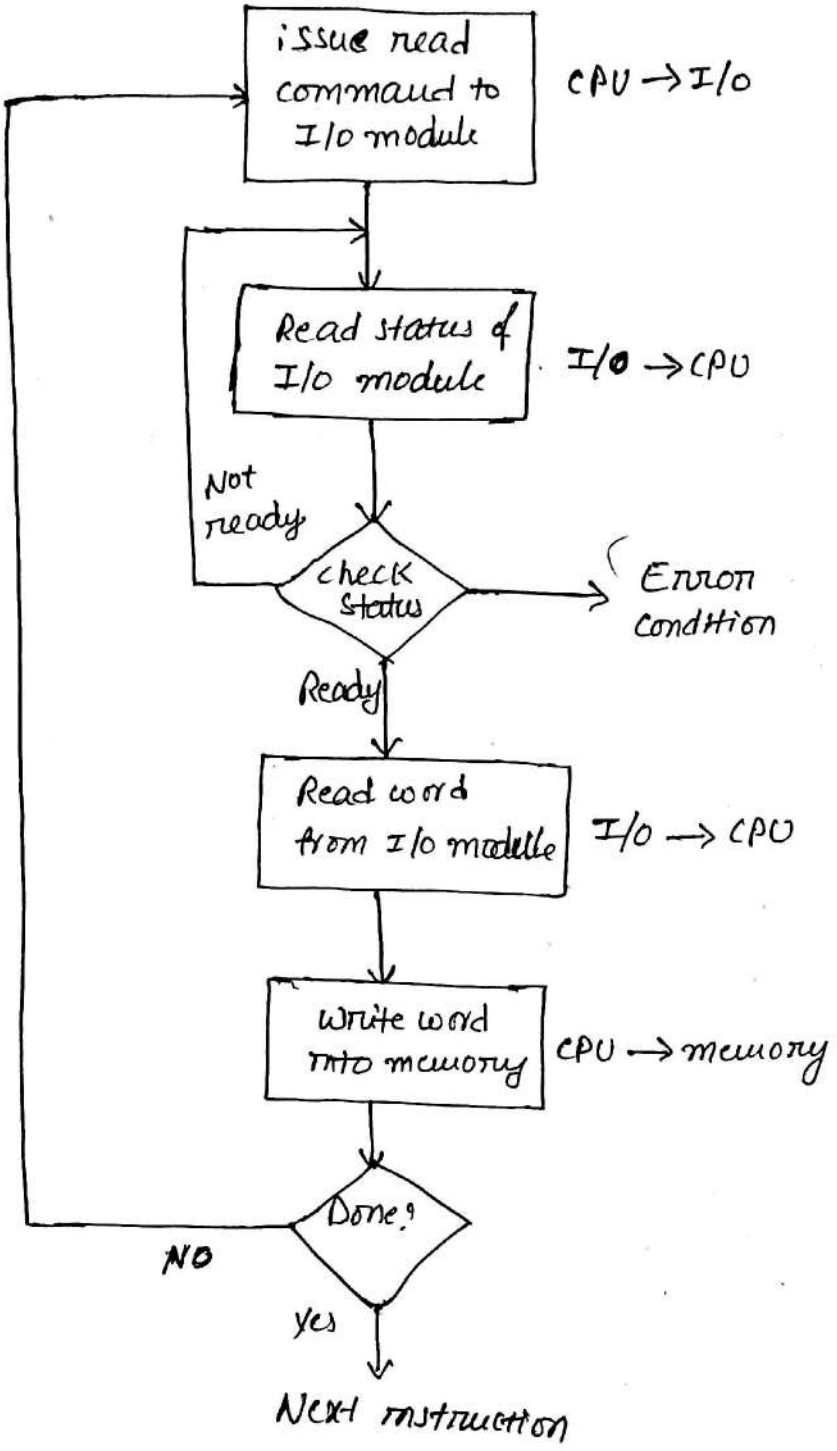
① Control:- Used to activate a peripheral and tell it what to do. For example magnetic-tape unit may be instructed to rewind, or to move forward, or to record.

② Test:- Used to test various status conditions associated with an I/O module and its peripherals. The processor will want to know that the peripherals are available for use or not.

③ Read:- Causes the I/O module to obtain an item of data from the peripherals and place it in an internal buffer. The processor can then obtain the data item by requesting that the I/O module place it on the data bus.

④ Write:- Causes the I/O module to take an item of data from the data bus and subsequently transmit that data item to the peripheral.

With programmed I/O the instructions are easily mapped into I/O commands, and there is often a simple one-to-one relationship. The form of the instruction depends on the way in which external devices are addressed.

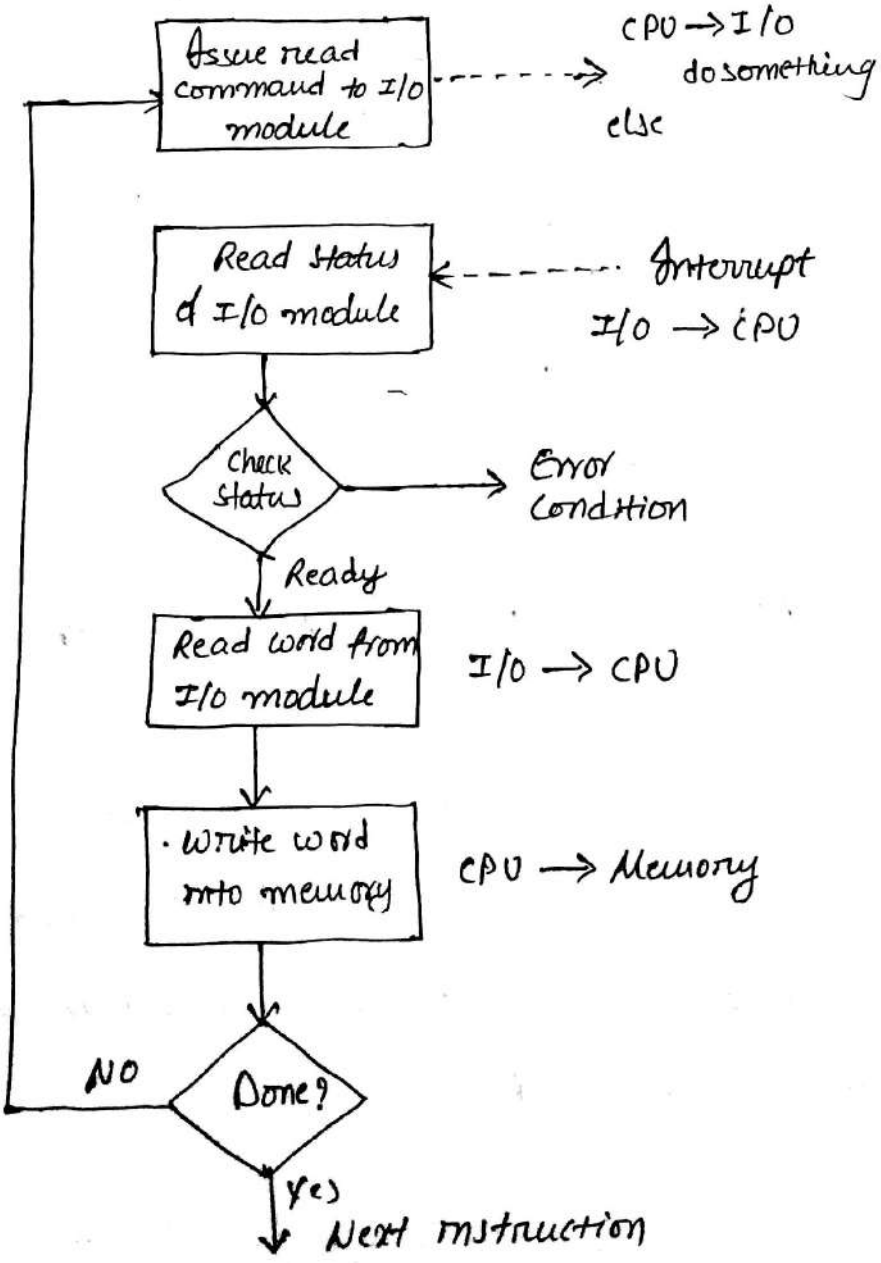


In programmed I/O the processor sends a signal to access a peripheral device, and need to wait until it gets a response from the device. Meanwhile the processor is going to be in waiting state. The flow chart is an example of the use of programmed I/O to read in a block of data from a ~~data~~ peripheral device.

Interrupt Driven I/O :-

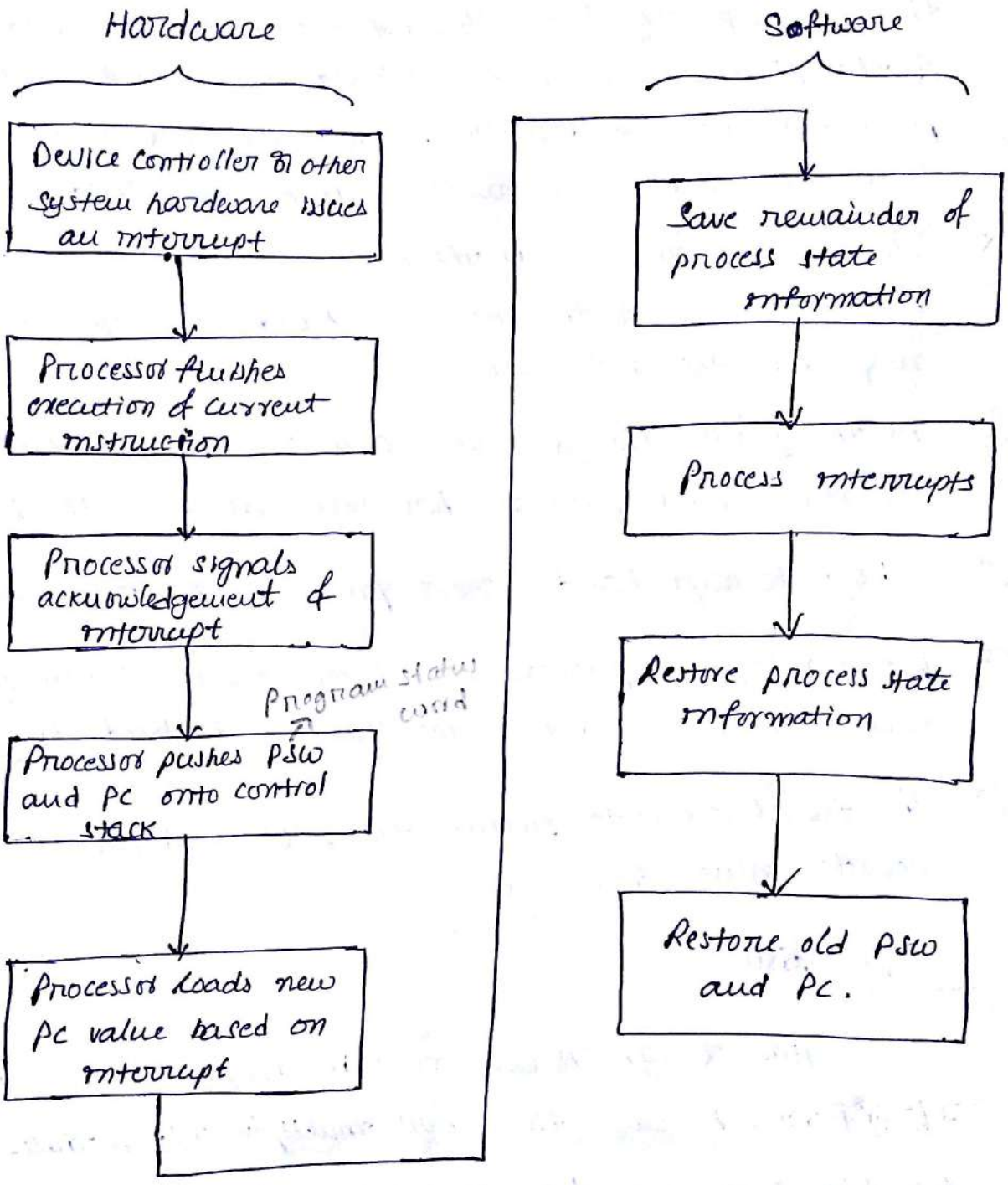
The problem with programmed I/O is that the processor has to wait a long time for the I/O module of concern while waiting the processor must repeatedly interrogate the status of the I/O module. As a result the level of the performance of the entire system is degraded.

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 then interrupt the processor to request service when it is ready to exchange data with the processor. Then the processor executes the data transfer first and then resumes its previous execution.



Interrupt Processing :

The occurance of an interrupt triggers a number of events both in the processor hardware and in software



① The device issues an interrupt signal to the processor.

② The processor finishes execution of the current instruction before responding to the interrupt.

(3) The processor tests for an interrupt, determines that there is one, and sends an acknowledgment signal to the device that issued the interrupt.

(4) The processor now needs to prepare to transfer control to the interrupt routine. The information required is the status of the processor, which is contained in a register called the program status word (PSW), the location of next instruction to be executed which available in PC will stored in a stack.

(5) The processor now loads the program counter with the entry location of the interrupt-handling program that will respond to this interrupt.

(6) At this point the program counter and PSW relating to the interrupted program have been saved on the system stack.

(7) The interrupt handler next processes the interrupt.

(8) When interrupt processing is complete, the saved registered values are retrieved from the stack and stored to the registers.

(9) The final act is to restore the PSW and program counter values from the stack.

Design Issues :-

Two design issues arise in implementing interrupt I/O (*) First, because there will multiple I/O modules, how does the processor determine which device issued the interrupt?

(*) Second, if multiple interrupts have occured, how does the processor decide which one to process?

First we will see device identification. Four general categories of techniques are in common use.

→ Multiple interrupt lines

→ Software poll

→ Daisy chain (hardware poll, vectored)

→ Bus arbitration (vectored).

The most straightforward approach to the problem is to provide multiple interrupt lines between the processor and the I/O modules. It is impractical to dedicate more than a few lines of processor pins to interrupt lines. Consequently even if multiple lines are used, it is likely that each line will have multiple I/O modules attached to it.

Another technique is software poll. When processor detects an interrupt, it branches to an interrupt-service routine whose job it is to poll each I/o module to determine which module caused the interrupt. The poll could be in the form of a separate command line. For example the processor raises TEST I/O and places the address of a particular I/o module on the address lines. The I/o module responds positively if it set the interrupt. Otherwise each I/o module could contain an addressable status register. Then the processor reads the status register of each I/o module to identify the interrupting module. Once the correct module is identified, the processor branches to a device-service routine specific to that device.

The disadvantage of the software poll is that it is time consuming. A more efficient technique is to use a "daisy chain" which provides a hardware poll. The interrupt acknowledge line is daisy chained to all the modules. Whenever there is an interrupt, the processor send out an interrupt acknowledge which

will propagate throughout the series of I/O modules. This process will continue until it reaches a requesting module. The module will respond by placing a word on the datalines. The word is known as vector. This vector can either be the address of the module or a specific identifier. The processor subsequently directs the module to it's specific device-service routine based on it's vector. This technique is also known as the "vectored interrupt". It completely removes the need for interrupt - service routine.

The last method which also utilises vectored interrupts is bus arbitration. This method involves the I/O module gaining control over the bus before requesting for the interrupt. This is limited to only one module at a time. The processor sends an acknowledge signal whenever it detects an interrupt. The requesting module then places its vector on the datalines.

However, when there are multiple interrupts at a single time, there will be a need to assign priorities. These 4 methods have their own way of assigning priorities.

① Multiple interrupt lines :- The processor picks the interrupt line with highest priority.

② Software poll :- The priority is determined by the order in which the modules are polled.

③ Daisy chain :- The priority is determined by the order in which the modules are polled.

④ Bus Arbitration :- Employs it's own priority scheme.

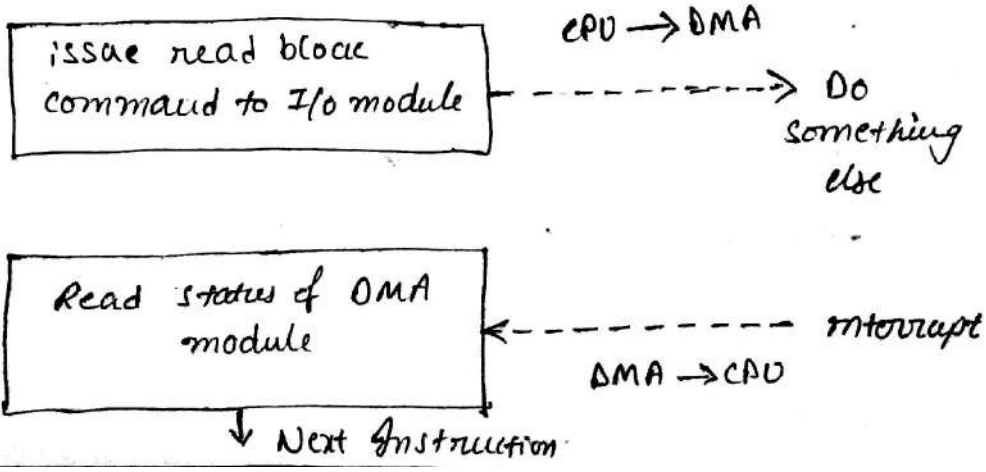
The advantages of interrupt driven I/O is fast and efficient. The disadvantages are can be tricky to write if using a low level language. It can be tough to get various pieces to work well together. It usually done by the hardware manufacturer / OS maker.

Direct Memory Access (DMA) :-

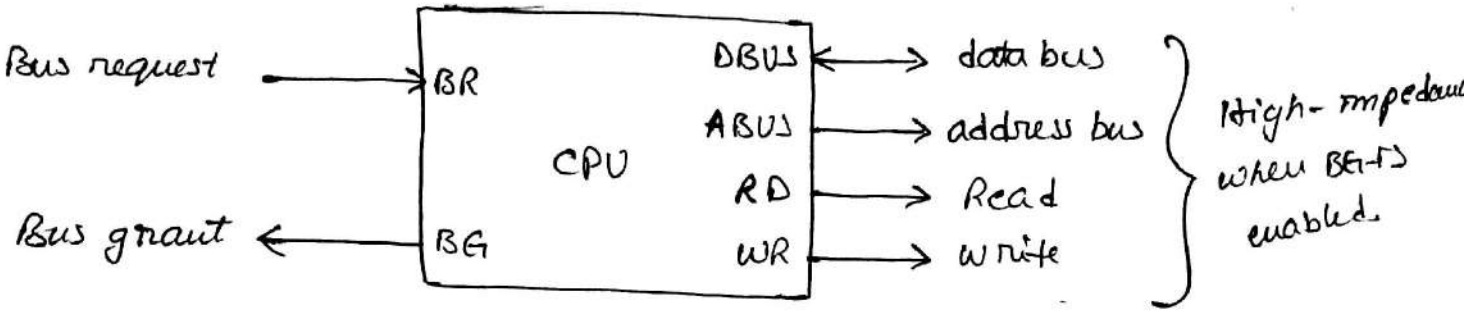
Direct memory access allows devices to transfer data without subjecting the processor. Generally the processor would have to copy each piece of data from source to destination. This is typically slower than copying normal blocks of memory since access to I/O devices over peripheral bus is generally slower than normal system RAM. During the time the processor would be unavailable for any other task involving processor bus access. DMA transfers are essential for high performance embedded systems where large chunks of data need to be ~~transferred~~ transferred from the input/output devices to or from the primary memory.

The copy of data can be performed from

- → I/o device to memory
- → memory to I/o device
- → memory to memory
- → I/o device to I/o device



The following figure shows two control signals in the CPU that facilitate the DMA transfer.

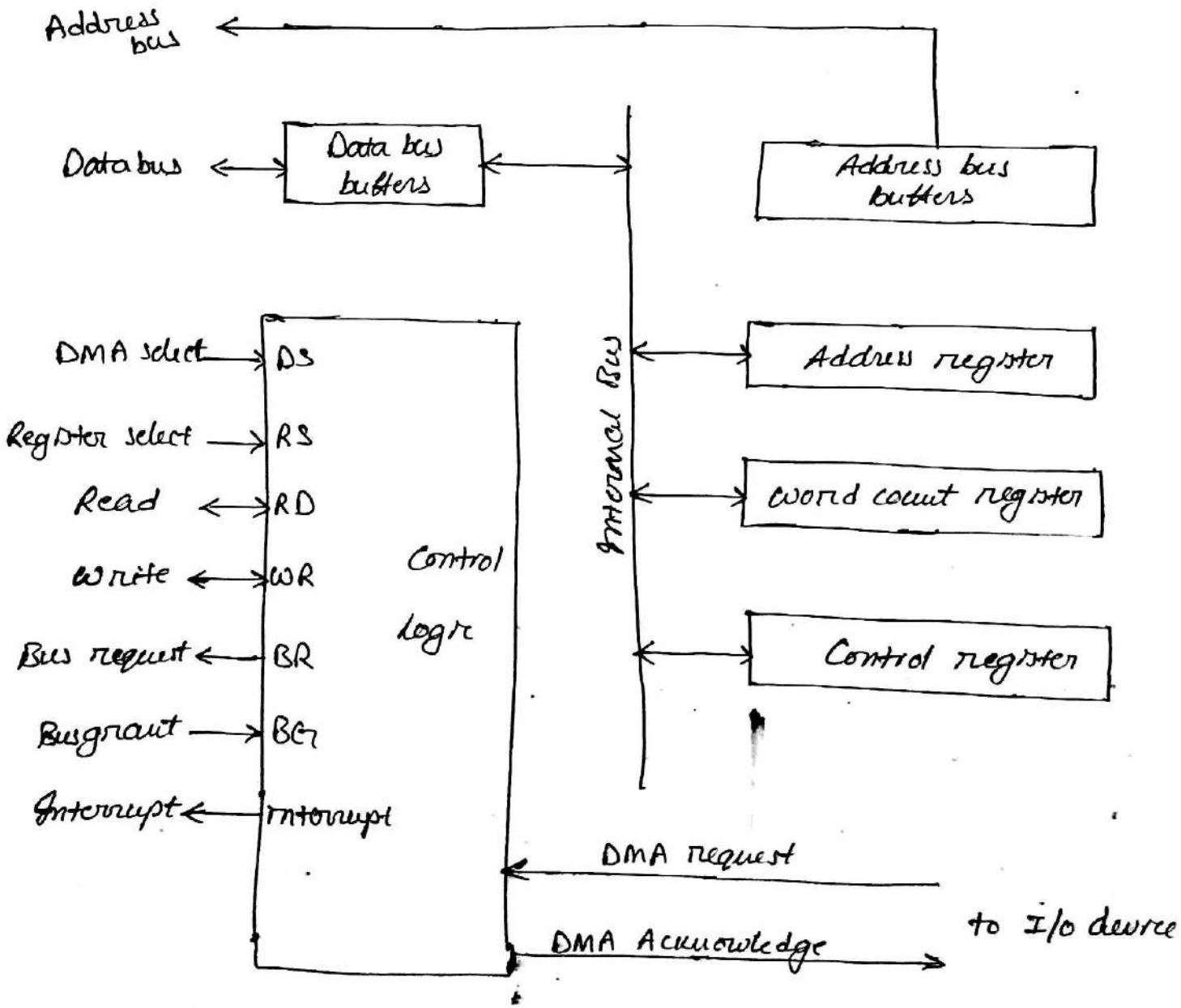


The "bus request" (BR) is used by the DMA controller as input to request the CPU to get control of the buses. When this input is active, the CPU terminates the execution of current instruction and places the address and data bus, read, write lines into a high-impedance state. The CPU activates the bus grant (BG) output to inform the external DMA that the buses are in the high-impedance state.

When the DMA takes control of the bus system, it communicates directly with the memory. The transfer can be made in several ways. In DMA "burst transfer", a block sequence consisting of a number of memory words is transferred in a continuous burst while the DMA controller is master of the memory buses. This mode of transfer is needed for fast devices such as magnetic disks, where an entire block is transferred. An alternative technique called "cycle stealing" allows the DMA controller to transfer one data word at a time, after which it must return control of the buses to the CPU. The CPU delays its operation for one memory cycle to allow the direct memory I/O transfer to "steal" one memory cycle.

DMA Controller :-

The DMA controller needs the usual circuits of an interface to communicate with the CPU and I/o devices. It needs an address register, a word count register, and a set of address lines. The address register and address lines are used for direct communication with the memory. The word count register specifies the number of words that must be transferred. The following figure is the block diagram of a typical DMA controller.

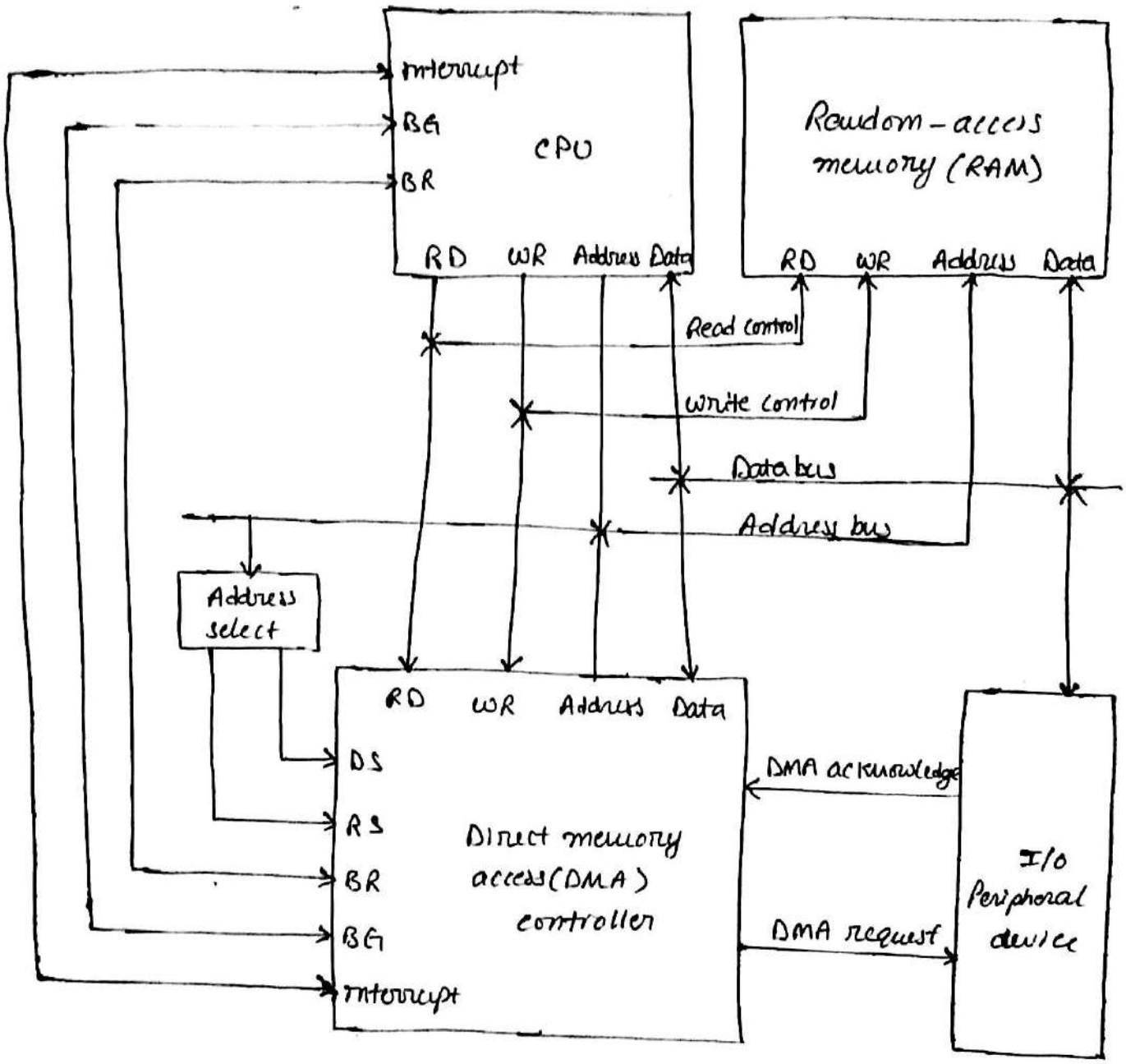


The unit communicates with the CPU via the data bus and control lines. The registers in the DMA are selected by the CPU through the address bus by enabling the DS and RS inputs. When BG=0 the CPU can communicate with the DMA registers through the data bus to read from or write to the DMA registers. When BG=1 the DMA can communicate directly with the memory by specifying an address in the address bus and activating the RD or WR control. The DMA communicates with the external peripheral through the handshaking procedure

The DMA controller has three registers: an address register, a word count register, and a control register. The address register contains an address to specify the desired location in memory. The address register is incremented after each word that is transferred to memory. The word count register holds the number of words to be transferred. This register is decremented by one after each word transfer.

DMA Transfer:-

The position of the DMA controller among the other components in a computer system is illustrated in following figure. The CPU communicates with the DMA through the address and data buses as with any interface unit. The DMA has it's own address, which activates the DS and RS lines. Once the DMA receives the start control command, it can start the transfer between the peripheral device and the memory.



When the peripheral device sends a DMA request, the DMA controller activates the BR line, informing the CPU to relinquish the buses. The DMA puts the current value of its address register into the address bus, initiates the RD and WR signal, and sends a DMA acknowledge to the peripheral device. The RD and WR lines in the DMA controller are bidirectional. The transfer will follows the procedure which discussed on above concept.

DMA operations:- A lot of different operating modes exist for DMACs. They are

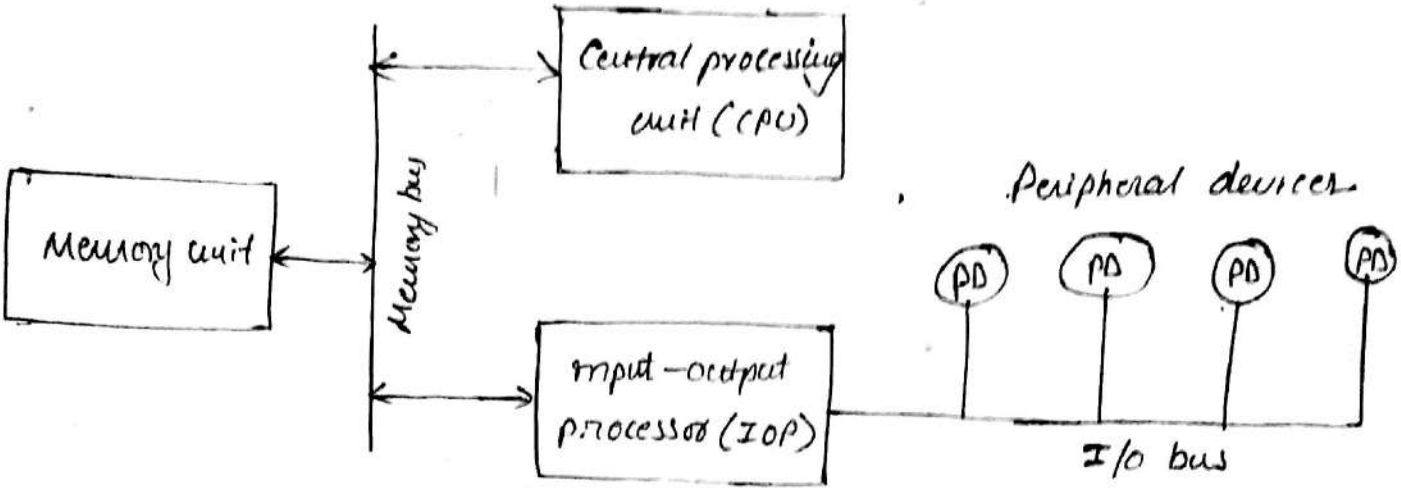
→ Single block transfer
→ Chained block transfer
→ Linked block transfer
→ Fly-by transfers.

Input-Output Processor (IOP)

Instead of each I/O device communicate with the CPU, a computer may incorporate one or more external processors and assign them the task of communicating directly with all I/O devices. An IO (input-output) processor is a processor with direct memory access capability that communicates with I/O devices. In this way the system will have memory unit, a number of processors comprised of the CPU and one or more IOPs. Each IOP takes care of input and output tasks, relieving the CPU from the housekeeping chores involved in I/O transfers.

The IOP is similar to CPU except it designed for only IO operations. The IOP can fetch and decode its own instructions. IOP instructions are specifically designed to facilitate I/O transfers. In addition IOP can perform other processing tasks such as arithmetic, logic, branching, and code translation.

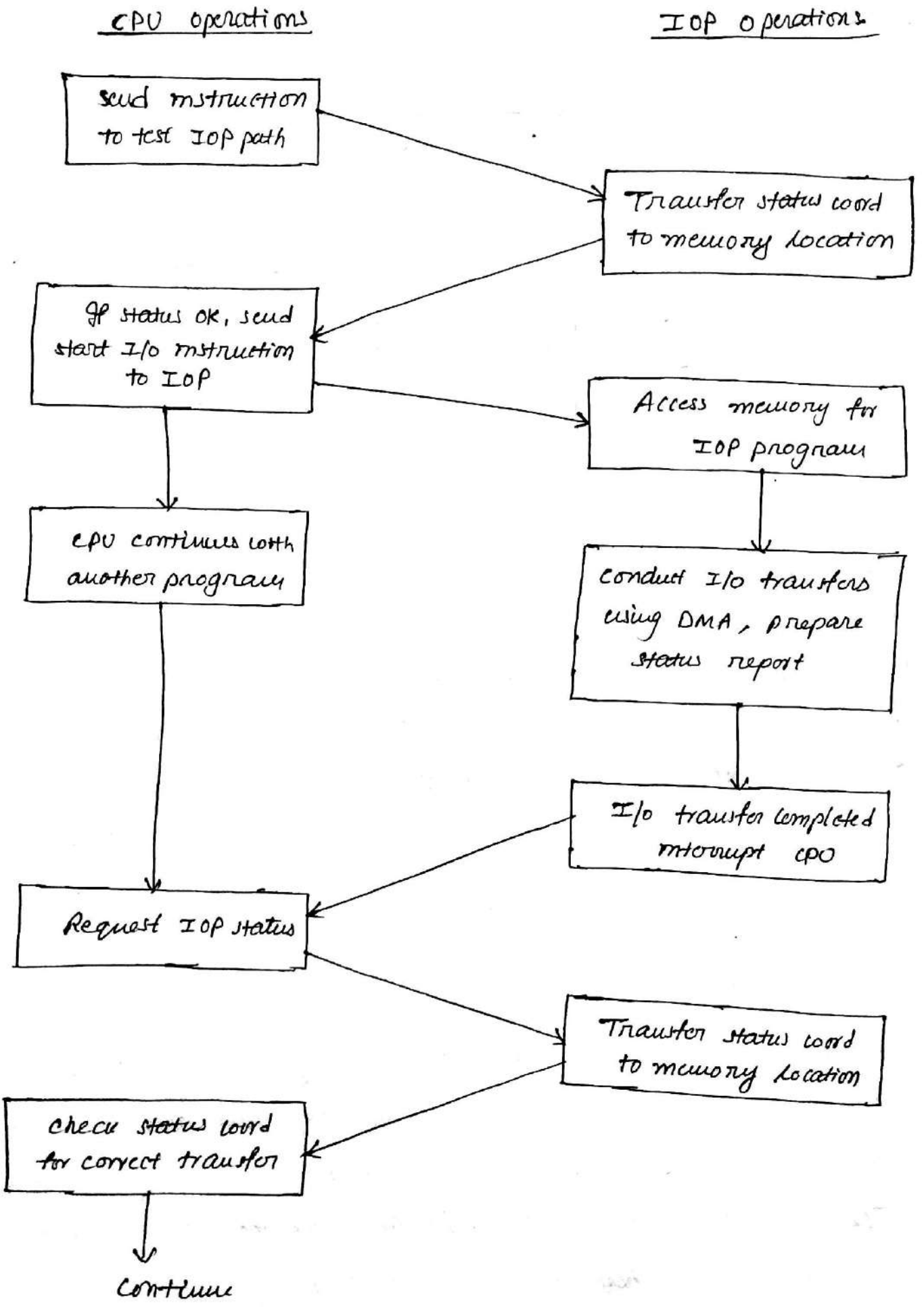
The block diagram of a computer with two processors is shown below.



The data formats of peripheral devices differ from memory and CPU data formats. The IOP must structure data words from many different sources. The communication between IOP and the devices attached to it is similar to the program control method of transfer. Communication with the memory is similar to the direct memory access method. The CPU is the "master" while the IOP is a "slave" processor. Instructions that are read from memory by an IOP are sometimes called "commands".

CPU-IOP communication:

The communication between CPU and IOP may take in different forms, depending upon the particular computer considered. In most cases the memory unit acts as a message center where each processor leaves information for the other. The following flowchart indicates how the communication is carried out between CPU and IOP.



The CPU sends an instruction to test the IOP path. The IOP responds by inserting a status word in memory for the CPU to check. The CPU refers to the status word in memory to decide what to do next. If all is OK, the CPU sends the instruction to start I/O transfer. Now the CPU continue with another program while the IOP is busy with the I/O program. Both programs refer to memory by using DMA transfer.

When the IOP terminates the execution of it's program, it sends an interrupt request to the CPU. Now the CPU responds to the interrupt by issuing an instruction to read the status from the IOP. The status word indicates whether the transfer has been completed or if any occurred during the transfer. The IOP takes care of all data transfers between several I/O units and the memory while the CPU is processing another program.

Processor Examples :-

IBM 370 I/O Channel :-

The I/O processor in the IBM 370 computer is called a "channel". A computer system configuration includes a number of channels with each channel attached to one or more I/O devices. There are three types of channels

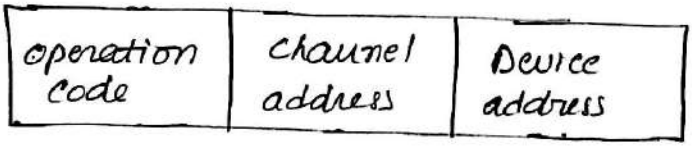
1. Multiplexer
2. Selector
3. block - multiplexer

→ The multiplexer channel will be connected with no. of slow- and medium speed devices and is capable of operating with a number of I/O devices simultaneously.

→ The selector channel is designed to handle one I/O operation at a time and is normally used to interconnect with high-speed device.

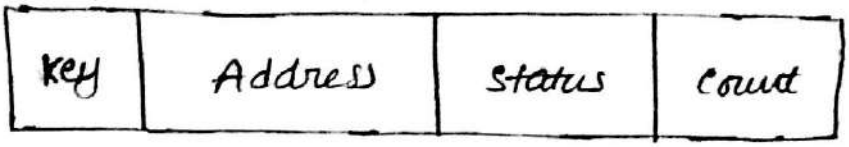
→ The block-multiplexer channel combines the features of both the multiplexer and selector channel.

The CPU communicates directly with the channels through dedicated control lines and indirectly through reserved storage areas in memory. The following are the word formats associated with the channel operation.



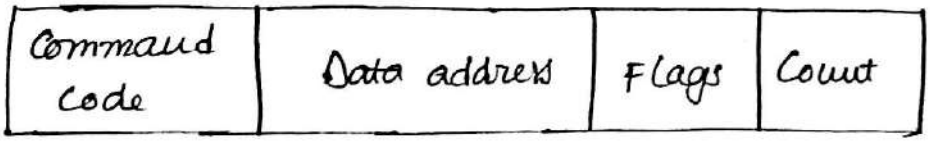
I/O instruction format

The I/O instruction format has three fields: operation code, channel address, and Device address. The computer system may have different channels each associated with an address. This address is specified in channel address field. Similarly each channel is interconnected with different devices which are having individual addresses, specified in device address field. The operation code specifies one of eight I/O instructions: start I/O, start I/O fast release, test I/O, clear I/O, halt I/O, halt device, test channel, and store channel identification.



Channel status word format

The key field is a protection mechanism used to prevent unauthorized access. The address field consists the address of the last command word used by the channel. The count field gives the residual count when the transfer was terminated. At the end of the transfer the count will become zero. The status field identifies the conditions in the device and the channel and any errors that occured during the transfer



Channel command word format.

The data address field specfies the first address of a memory buffer and the count field gives the number of bytes involved in the transfer. The command field specfies ~~the~~ an I/O operation and flag bits provide additional information about the channel.

The Command field corresponds to an operation code that specfies one of six basic I/o operations.

1) write : Transfer data from memory to I/o devices
2) Read : Transfer data from I/o device to memory
3) Read backwards : Read magnetic tape with tape moving backward.

4) Control : Used to initiate an operation not involving transfer of data, such as rewinding of tape & positioning a disk-access mechanism.

5) Sense : Informs the channel to transfer its channel status word to memory location 6H.

6) Transfer in channel : Used instead of a jump instruction. Here the data address field specifies the address of the next command word to be executed by the channel.

Intel 8089 IOP :-

The Intel 8089 I/O processor is contained in a 40-pin integrated circuit package. Within the 8089 are two independent units called channels. Each channel combines the general characteristics of a processor unit, with those of a direct memory access controller. The 8089 is designed to function as an IOP in a microcomputer system where the Intel 8086 microprocessor is used as the CPU. The 8086 CPU initiates an I/O operation by building a message in memory that describes the function to be performed. The 8089 IOP reads the message from memory, carries out the operation, and notifies the CPU when it has finished.

In contrast to the IBM 370 channel, which has only six basic I/O commands, the 8089 IOP has 50 basic instructions that can operate on individual bits, on bytes, or 16bit words.

A microcomputer system using the Intel 8086/8089 pair of integrated circuits shown below.

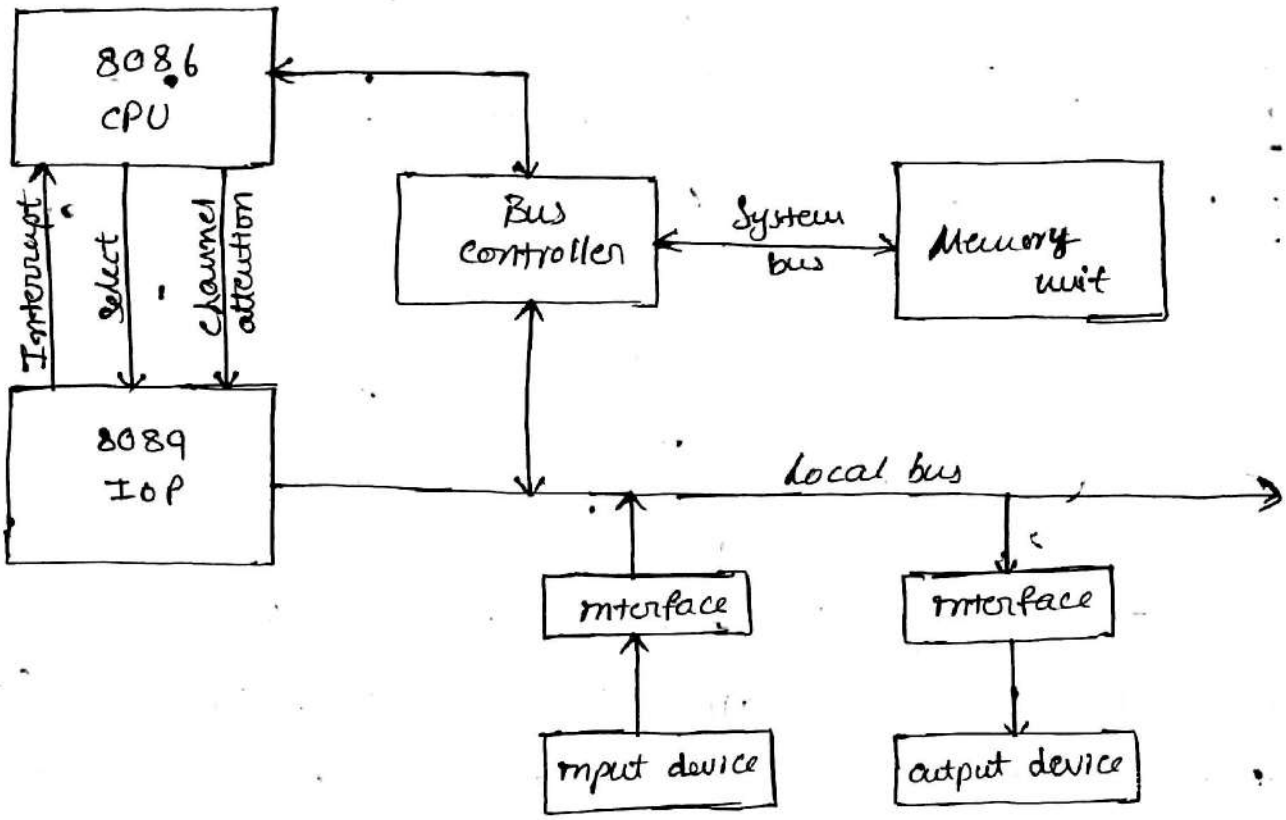
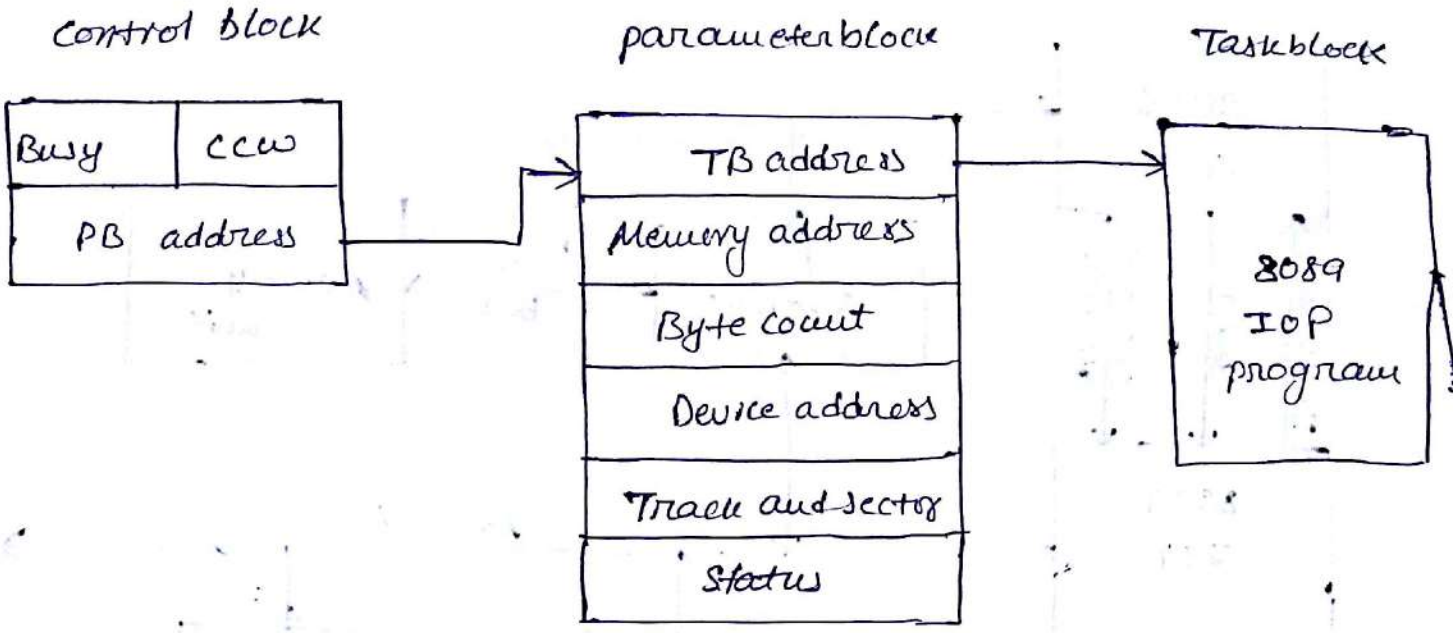


Fig: Intel 8086/8089 microcomputer system block diagram

The 8086 functions as CPU and the 8089 as the IOP. The two units share a common memory through a bus controller connected to a system bus, which is called "Multibus" by Intel. The IOP uses local bus to communicate with various interface units connected to I/O devices. The CPU communicates with the IOP by enabling the "channel attention" line. The select line is used by the CPU to select one of two channel in the 8089. The IOP gets the attention of the CPU by sending an interrupt request, when the channel has completed it's program.

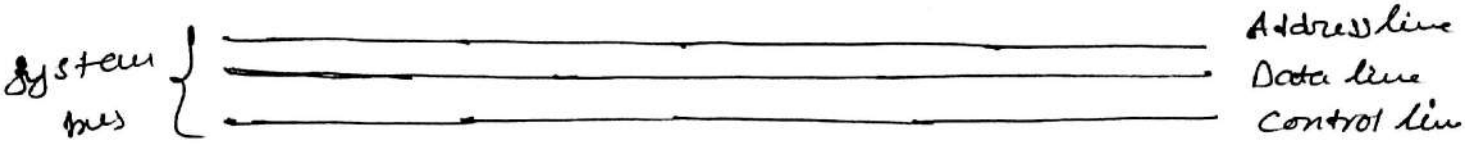
The communication scheme consists of a program selection called "blocks", which are stored in memory as shown below.



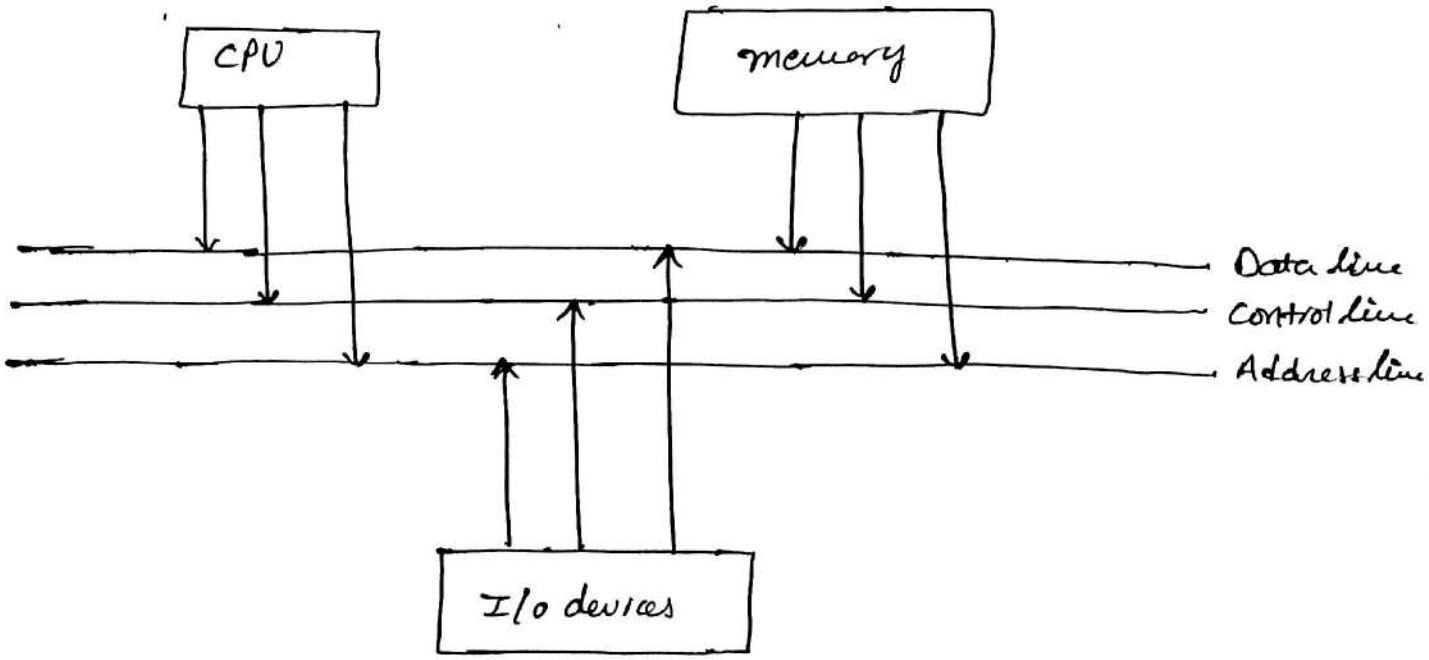
The busy flag indicates whether the IOP is busy or ready to perform a new I/O operation. The CCW ~~(control word)~~ (channel command word) is specified by the CPU to indicate the type of operation required from the IOP. The CCW here is more like an I/O instruction that specifies an operation for the IOP, such as start operation, suspend operation, resume operation, and halt I/O program. The memory address specifies the beginning address of a memory buffer. The Byte count gives the number of bytes to be transferred. The device address specifies the particular I/O device to be used. The track and sector number locate the data on the disk. When the I/O operation is completed the IOP stores its status bits in the status word location of the parameter block.

Buses

A Bus is a ~~memory~~ shared line which can be used to interconnect different components in the system. A bus consists of three lines namely address line, data line and control line.



A Single bus organization which is interconnecting different components of a system shown below.



Here control line and data lines are ~~bi~~ bidirectional and address line is a uni directional. The buses are categorized into two types they are

① internal bus

② External bus

A bus which is used to interconnect the internal components of a system is called internal bus. A bus which is used to interconnect the external components to a system is called external bus. The communication in a bus will be done in two ways.

① Synchronus

② Asynchronus

which we discussed in previous topics.

Scanned by CamScanner