Unit-8

Input Output Organization:

The input-output subsystem of a computer, referred as I/O, provides an efficient mode of communication between the central system and outside environment. Data and programs must be entered into the computer's memory for processing and result of computations must be displayed to user. This is done with the help of different peripheral devices.

Peripheral device → Input or output devices attached to computer are called peripheral devices. These devices provide an efficient mode of communication between the central system and the outside environment.

Types

Input devices → Keyboard, mouse etc.
Output devices → Printer, monitor etc.
Input-Output device → Memory

⊛ Input Output Interface / IO Interface:-

→ Input Output interface provides a method for transfering information between internal storage and external I/O devices.
→ It resolves the differences between the computer and peripheral devices which are as follows:

i) Data transfer rate of peripherals is slower than that of CPU. So some synchronization mechanism may be needed.
ii) Operating modes of peripherals are different from each other and each must be controlled so as not to disturb others.
iii) Data codes and formats in peripherals differ from the word format in CPU and memory.
iv) Peripherals are electromechanical and electromagnetic devices and manner of operation is different from that of CPU which is electronic component.

⊛. I/O Bus and Interface Modules:-

Peripherals connected to a computer need special communication link to interface with CPU. This special link is called I/O bus.

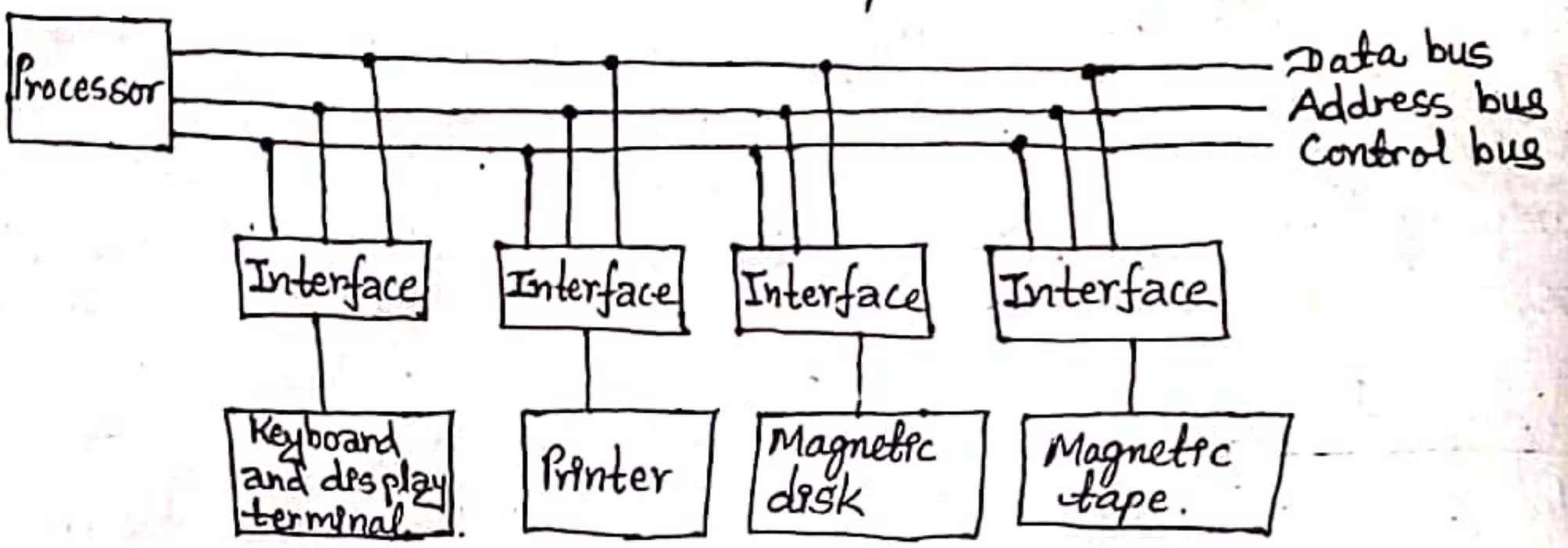


Fig. Connection of I/O bus to I/O devices

→ I/O bus from the processor is attached to all peripheral interfaces.
→ I/O bus consists of data lines, address lines and control lines.
→ To communicate with a particular device, the processor places a device address on the address lines. Each peripheral has an interface module associated with its interface.

Functions:

→ decodes the device address,
→ decodes the I/O commands.
→ Provides signal for peripheral controller.
→ Synchronizes the data flow.
→ Supervises the transfer rate between peripheral and CPU.

⊛ I/O Commands:-

The function code provided by processor in control line is called I/O command. There are 4 types of commands that an interface may receive:

a) Control command → Issued to active the peripheral and to inform it what to do.

b) Status command → Used to test various status conditions in the interface and peripherals. e.g, error during data transfer completion successfully.

c) Data input command → Causes the interface to read the data from peripheral and places it into the interface buffer.

d) Data output command → Causes the interface to read the data from the bus and saves it into the interface buffer.

⊛. I/O vs Memory Bus:

Following are the three ways that computer buses can be used to communicate with memory and I/O.

a). Use two seperate buses, one for memory and another for I/O.

b). Use one common bus for both memory and I/O but have seperate control lines for each.

c). Use one common bus for memory and I/O with common control line.

⊛ Isolated I/O vs Memory-Mapped I/O:

Isolated I/O

→ CPU has distinct input and output as well as memory instruction.
→ Distinct address space for I/O and memory operation.

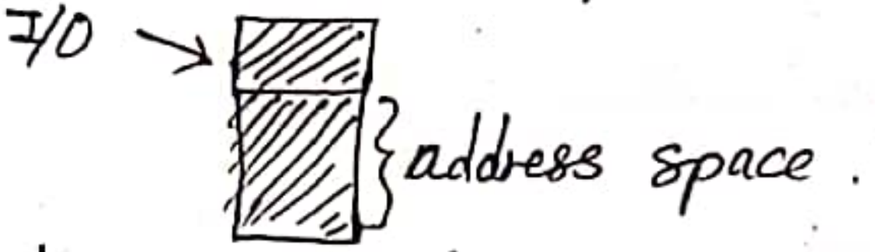
Address space for I/O

Address space for memory.

→ The ~~register~~ device register is 8-bit.
→ For application address space complete 1MB memory is allowed.
→ Maximum number of I/O devices are 256.

Memory-Mapped I/O

→ No distinct I/O and memory instruction.
→ No distinct address space.



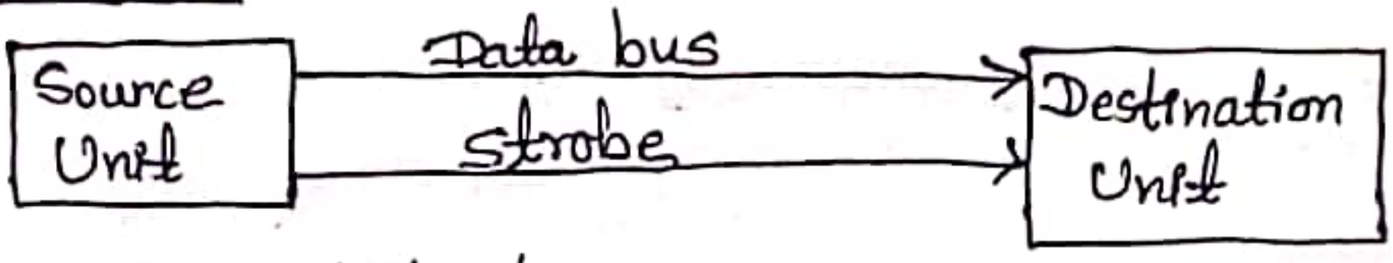
→ The device register is 16-bit.
→ It takes only some part of memory not complete 1MB memory.
→ Maximum number of I/O devices are 65536.

⊛. Asynchronous Data Transfer:-

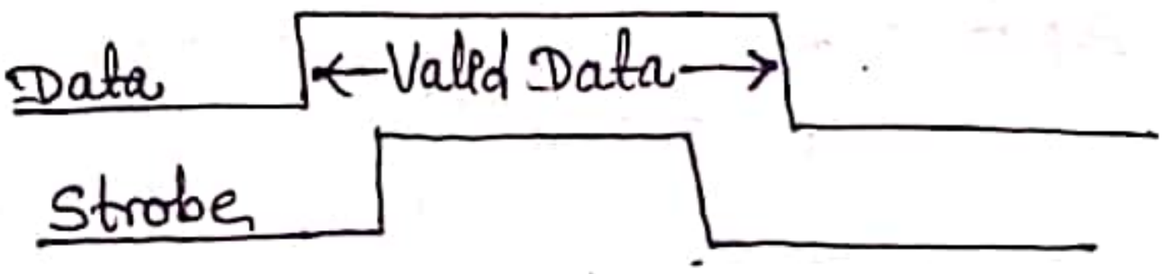
Generally communication in between CPU and peripherals is performed asynchronously. In this mode of transfer, there is need of control signals to be transmitted between the communicating unit to indicate the time at which data is being transmitted.

- For this we have two approach:
  - → Strobe method
  - → Handshaking method.

Strobe Method:

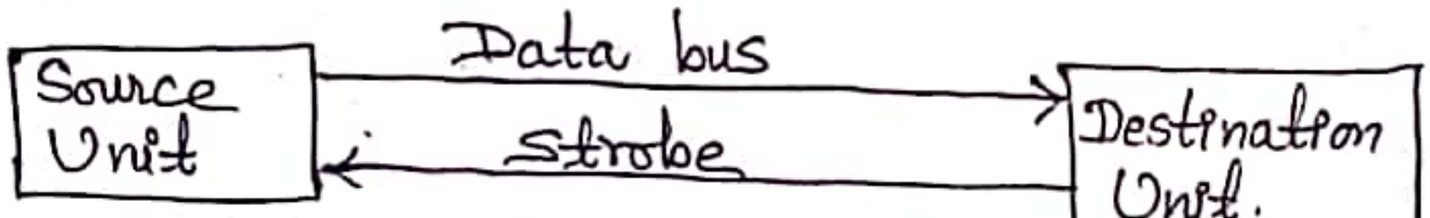


Block diagram

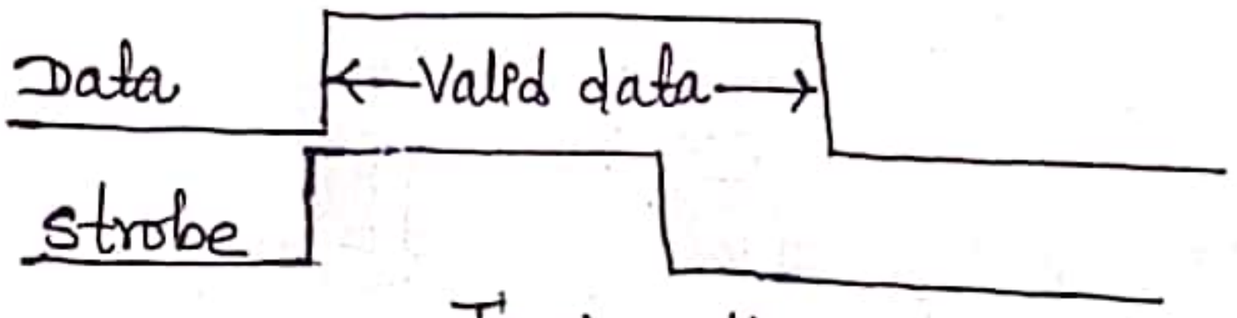


Timing diagram

fig. Source-initiated strobe for data transfer.



Block diagram



Timing diagram

fig. Destination-initiated strobe for data transfer.

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

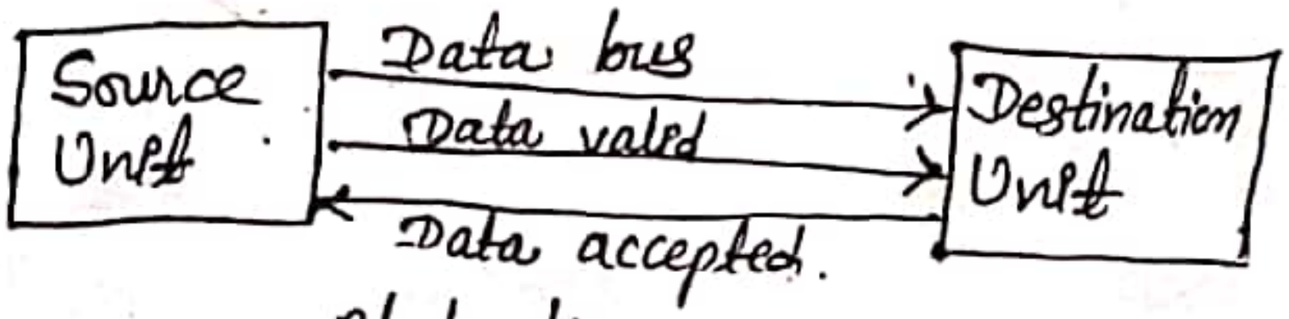
drawback

(Source-initiated) ⇒ No way of knowing whether the destination unit has actually received whether data item that was placed in the bus.

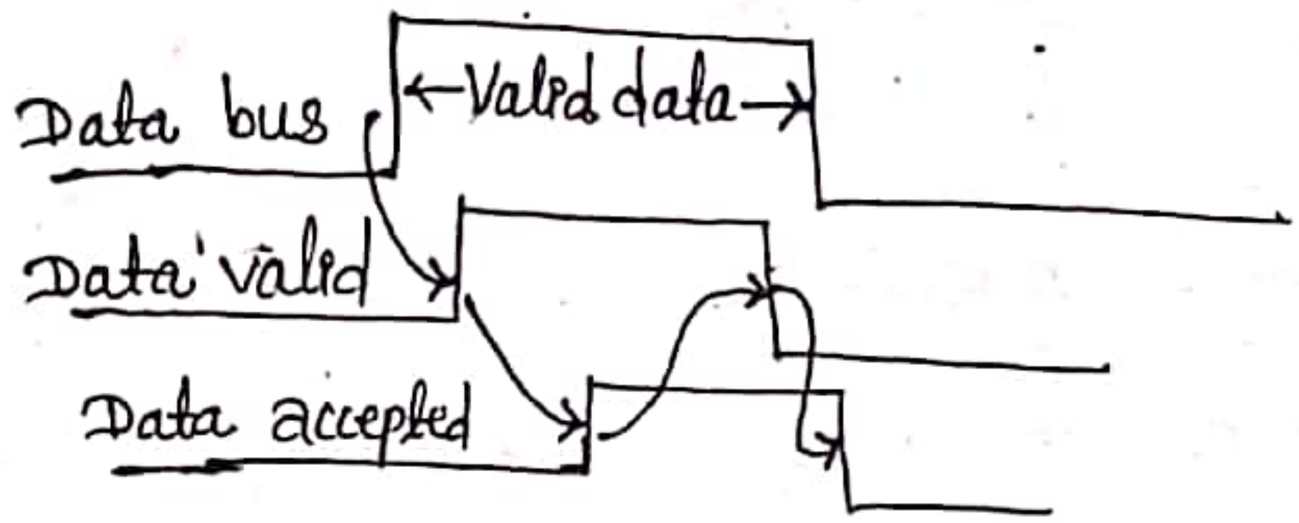
(Destination-initiated) ⇒ No way of knowing whether the source unit has actually placed data on data bus.

Handshaking Method:

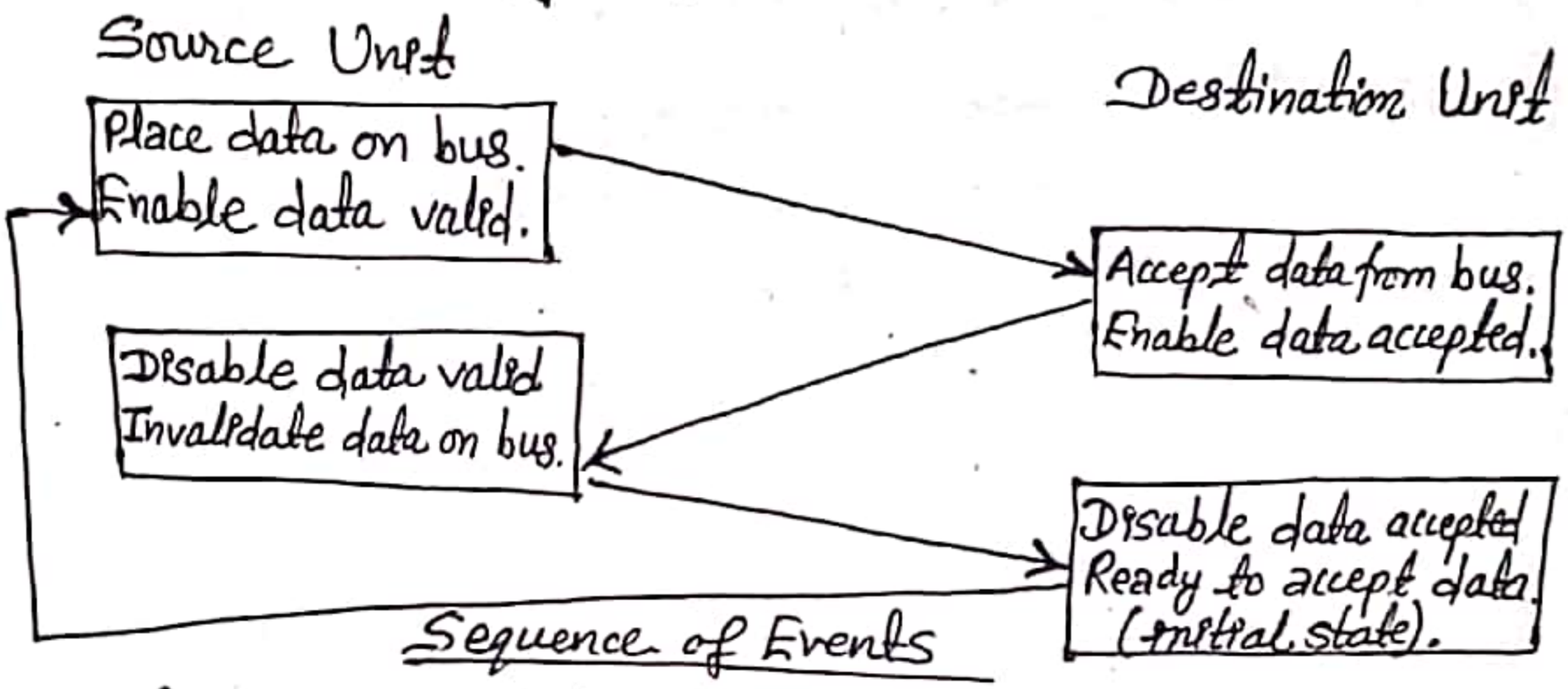
Solves the limitation of strobe method.



Block diagram

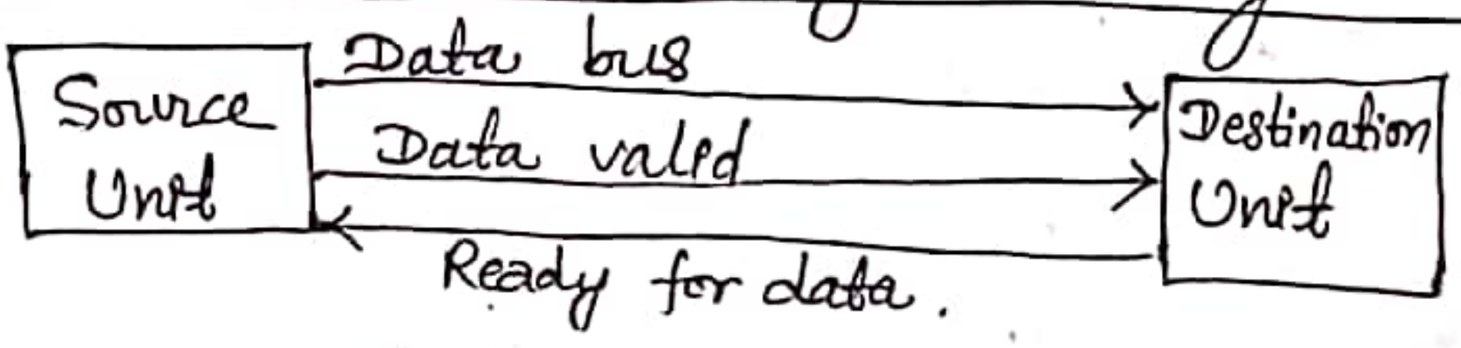


Timing Diagram

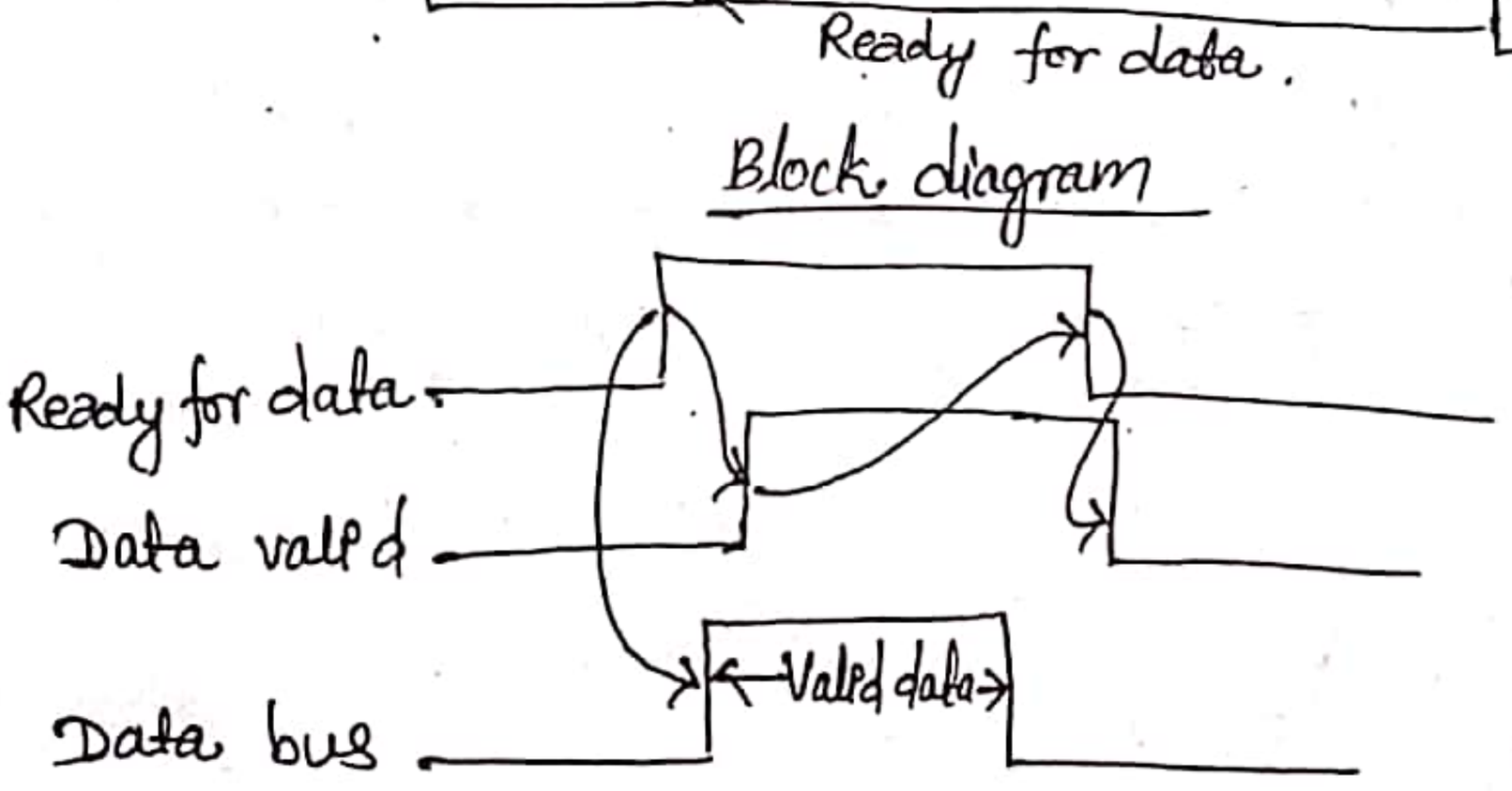


Sequence of Events

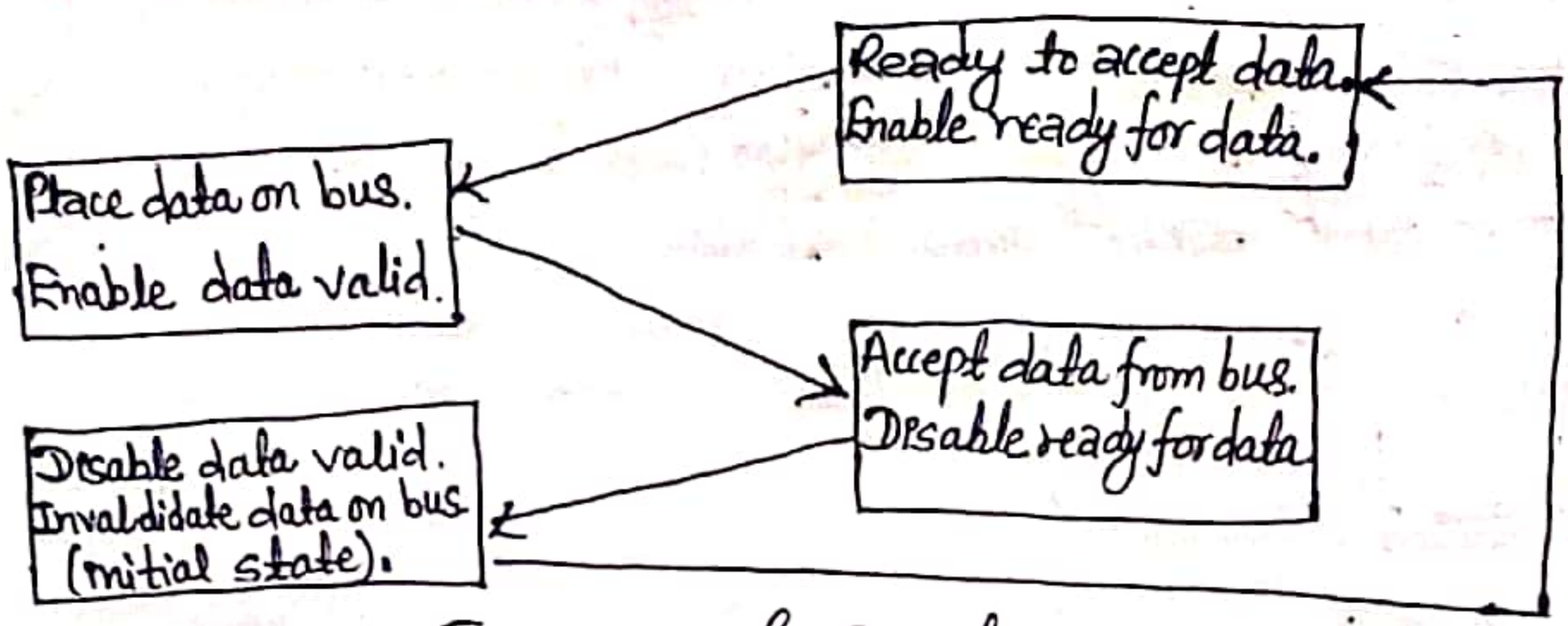
fig. Source-initiated transfer using handshaking



Block diagram



Timing Diagram



Sequence of Events

Fig Destination-initiated transfer using handshaking.

⊛. Modes of Transfer:-

Data transfer to and from peripherals can be handled in one of the following three possible modes.

a) Programmed I/O:- Programmed I/O operations are the result of I/O instructions written in computer program. In this method I/O device does not have direct access to memory. It requires constantly monitoring of I/O device. It is inefficient use of CPU.

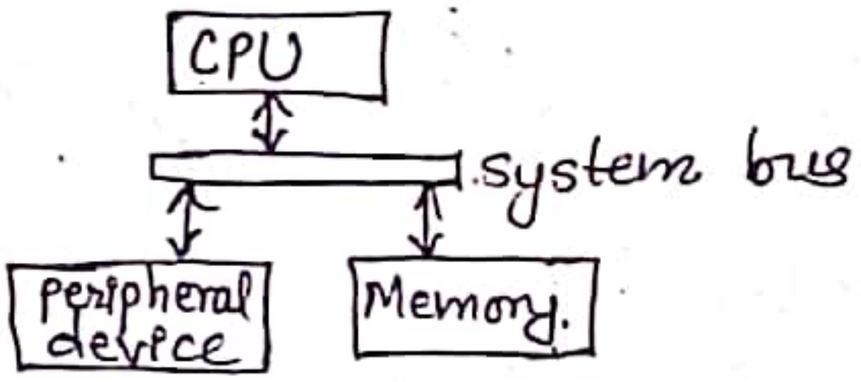


fig. programmed I/O.

b) Interrupt-Initiated I/O → This mode of transfer uses the interrupt facility. In this method CPU doesnot monitor peripherals. I/O devices generate interrupt signal to CPU when it is ready for data transfer. One interrupt signal is received by CPU, then it stops its ongoing task and branch to ISR to serve I/O devices. After serving I/O device, CPU resumes its previous job. It is efficient use of CPU.

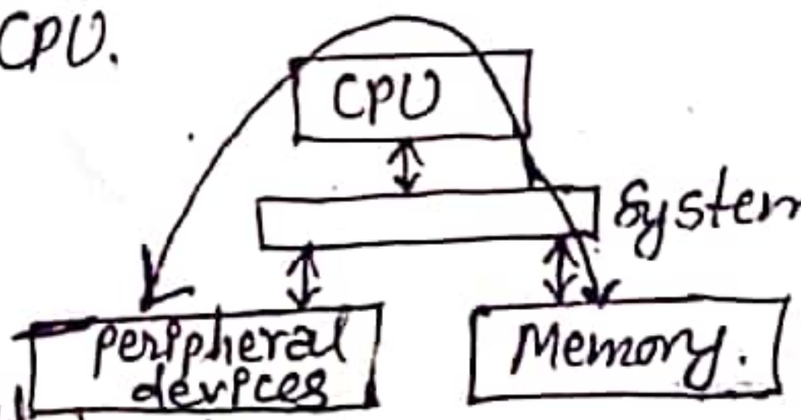


fig. Interrupt-initiated I/O.

c) Direct Memory Access (DMA) → It is very useful for bulk data transfer. e.g. data transfer in between memory and hard disk. Data transfer between peripherals and memory takes place without the involvement of CPU.

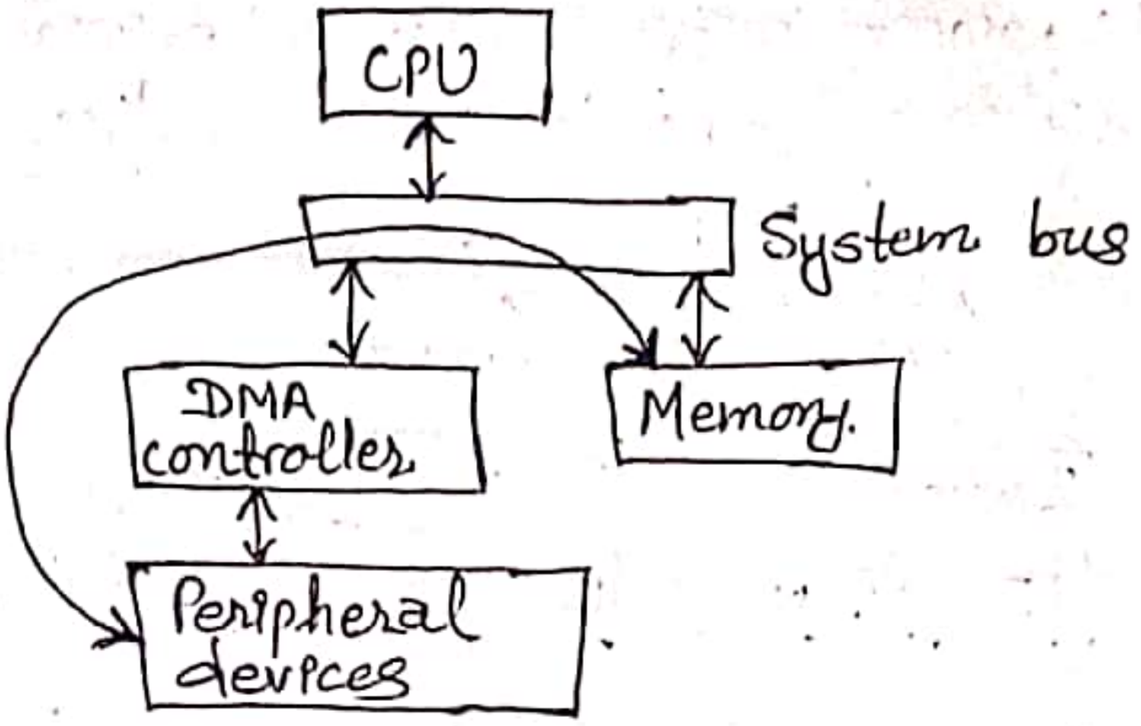


fig. DMA (Direct Memory Access).

⊛. Priority Interrupt:

⊛. How can CPU handle simultaneous interrupt from different interrupt source? [Question may be asked in this way].

Ans: There are two apporaches to handle simultaneous interrupt from different interrupt source.

a) Software approach:

→ Polling procedure.

b). Hardware approach:

→ Daisy-Chaining method

→ Parallel Priority interrupt method.

Polling procedure/Method → A polling procedure is used to identify the highest-priority source by software means. In this method there is common branch address for all interrupts. The program that takes care of interrupts begins at the branch address and polls the interrupt sources in sequence. The order in which they are tested determines the priority of each interrupt. The highest-priority source is tested first, and if its signal is on, control branches to a service routine for this source. Otherwise, the next-lower-priority source is tested and so on.

Daisy-Chaining Method: The daisy-chaining method of establishing priority consists of a serial connection of all devices that request an interrupt. The device with the highest priority is placed in the first position, followed by lower-priority devices up to the device with the lowest priority, which is placed last in the chain. This method of connection between three devices and the CPU is as shown below:

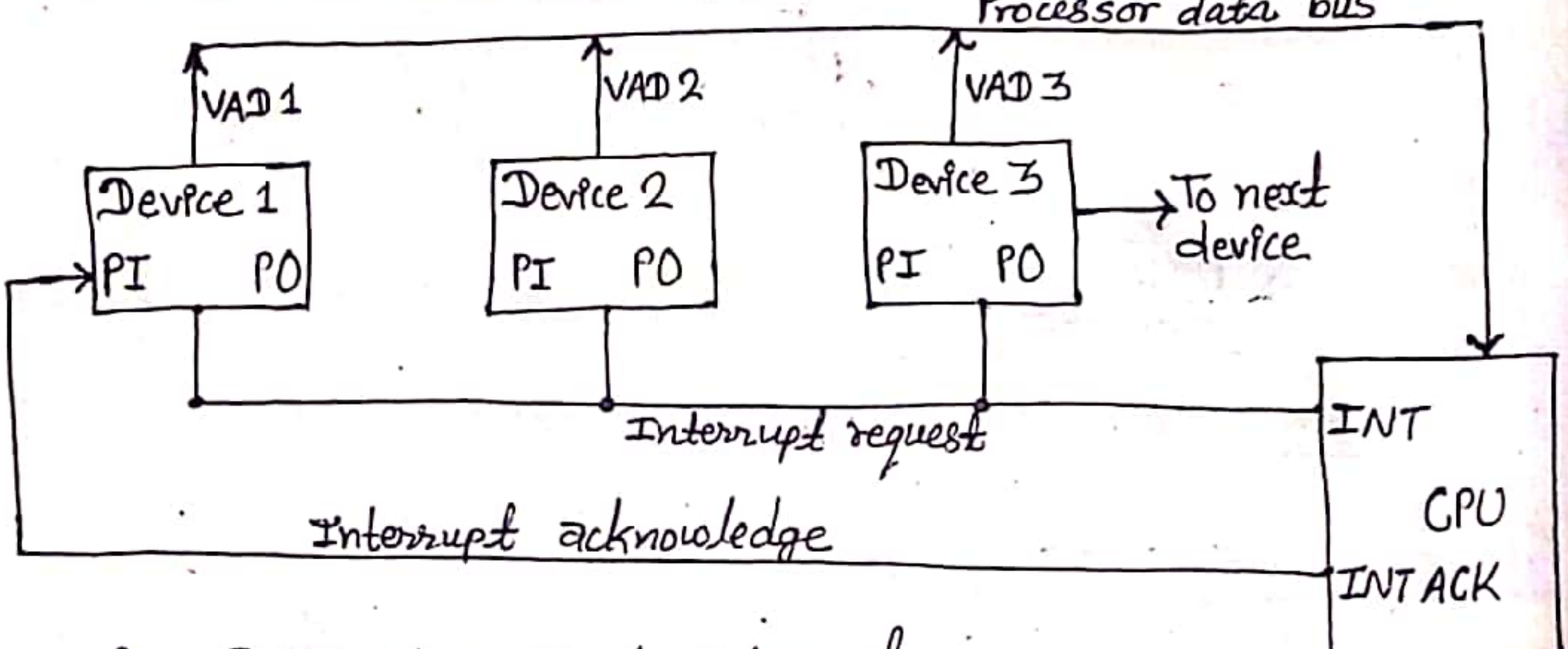
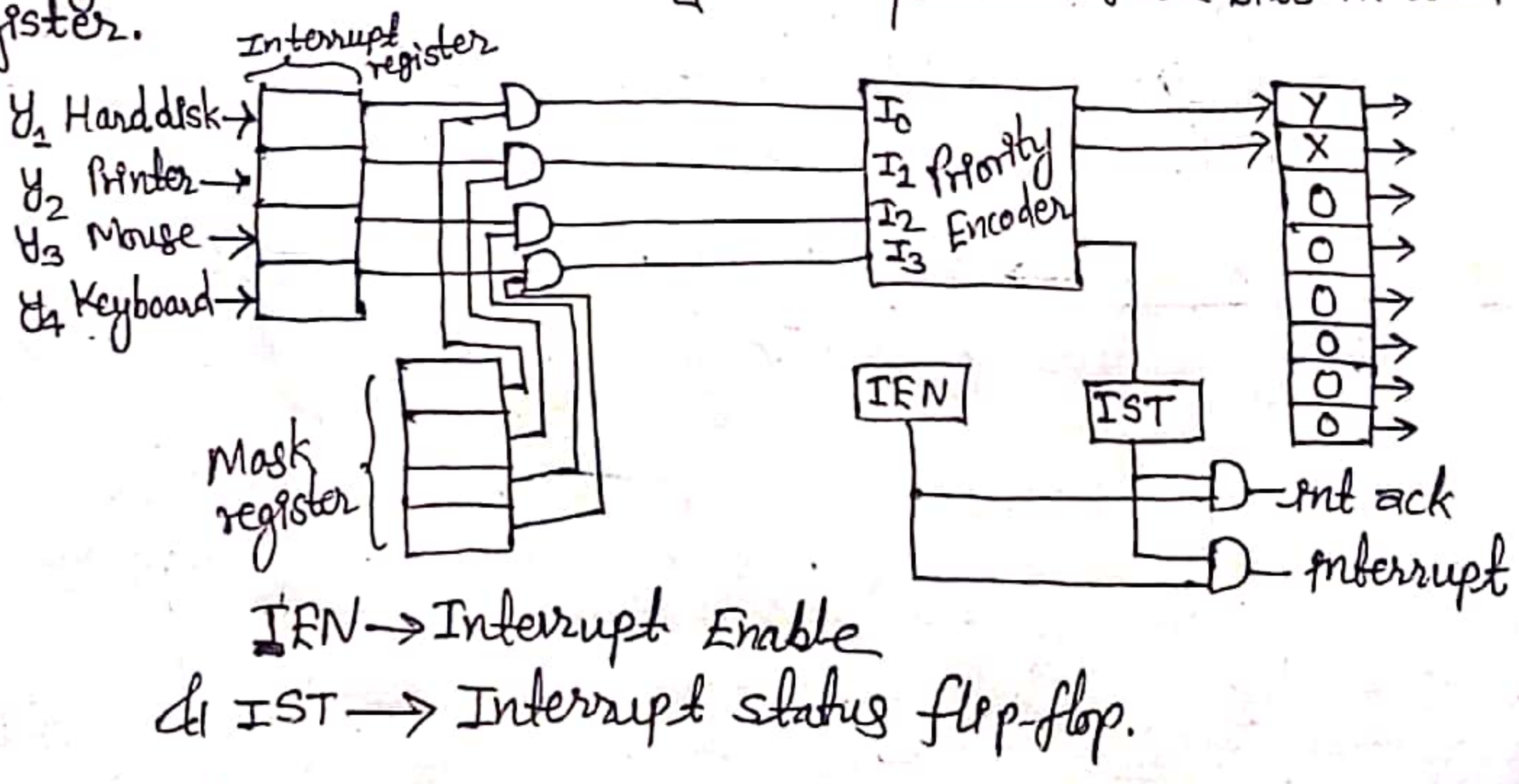


fig. Daisy-Chain Priority Interrupt.

Each device has priority in (PI) and priority out (PO). Also it provide interrupt vector address (VAD) to processor data bus.

Parallel Priority Interrupt Method:

The parallel priority interrupt method uses a register whose bits are set separately by the interrupt signal from each device. Priority is established according to the position of the bits in the register.



IEN → Interrupt Enable

IST → Interrupt status flip-flop.

⊛. DMA and IOP:

1> Direct Memory Access (DMA): Direct Memory Access (DMA) is a process for data transfer between memory and I/O, controlled by an external circuit called DMA controller, without the involvement of CPU.

Most of the data that is input or output from computer is processed by the CPU, but some data does not require processing or can be processed by another device. In these situations DMA can save processing time and is a more efficient way to move data from the computer's memory to other devices. For example: A PCI controller and a hard drive controller each have their own set of DMA channels.

⊛ Sequence of events that occur during DMA operation:-

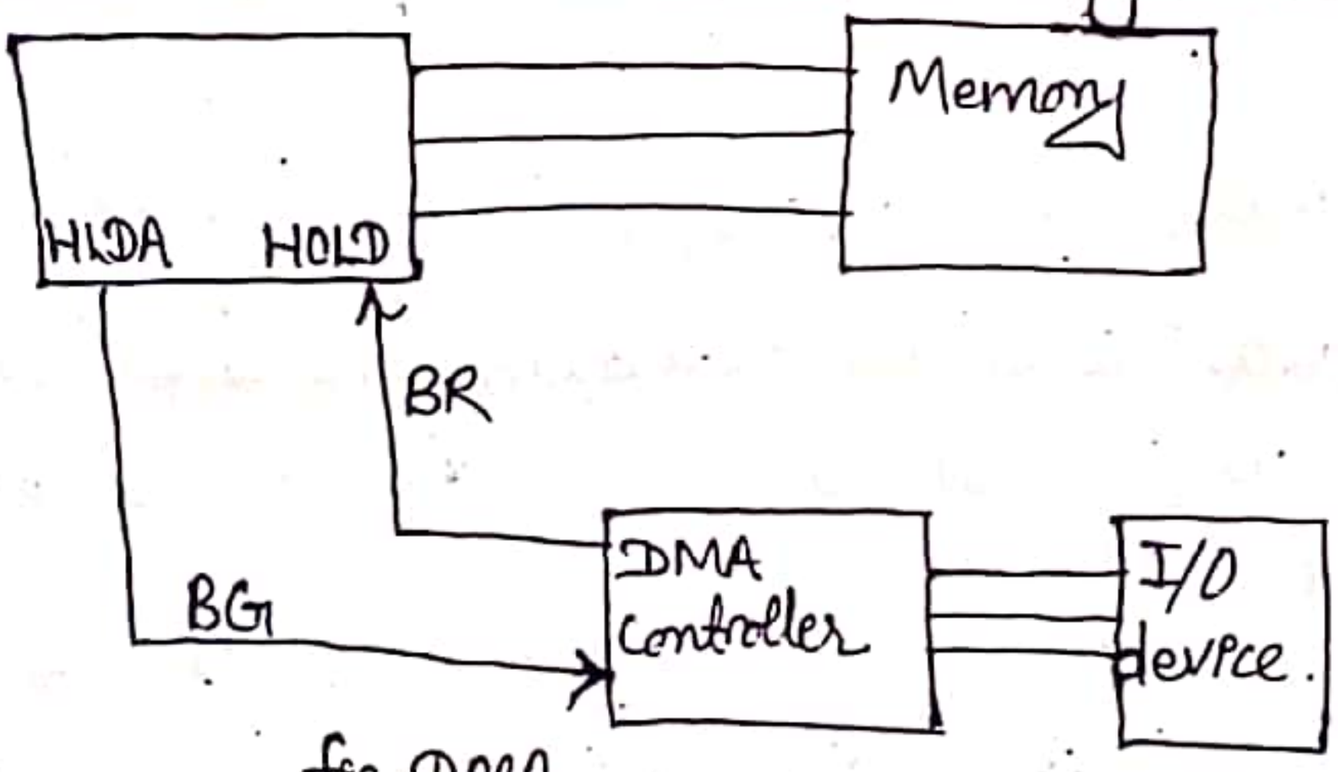


fig. DMA

The CPU/Processor has two pins HOLD and HLDA which are used for DMA operation. It works with following two control signals.

i) Bus Request (BR) → It is used by DMA controller to request CPU for buses. When this input is active, CPU terminates the execution of the current instruction and places the address bus, data bus and read & write lines into high impedance state.

ii) Bus Grant (BG) → CPU activates BG output to inform DMA that buses are available. DMA now take control over buses to conduct memory transfers without the involvement of CPU. When DMA terminates the transfer, it disables the BR line and CPU disables BG and returns to normal operation.

When DMA takes control of bus system, the transfer with memory can be made with Burst transfer and cycle stealing.

Q. What is DMA transfer? Explain.

Ans:-

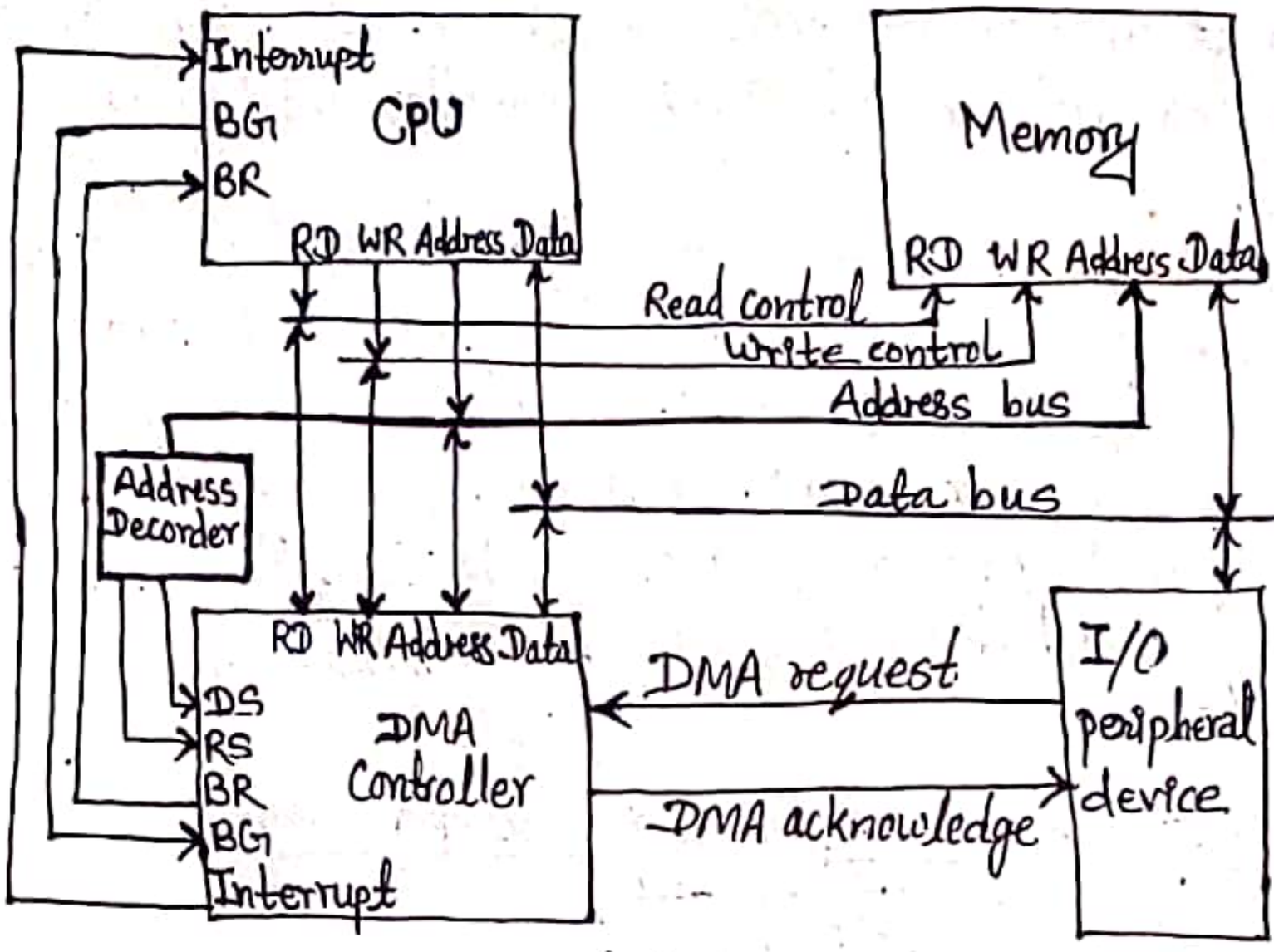


Fig. DMA transfer in a computer system.

→ CPU communicates with the DMA through address and data buses.

→ DMA has its own address which activates RS (Register select) and DS (DMA select) lines.

→ When a peripheral device sends a DMA request, the DMA controller activates the BR line, informing CPU to leave buses. The CPU responds with its BG line.

→ DMA then puts current value of its address register into the address bus, initiates RD and WR signals, and sends a DMA acknowledge to the peripheral devices.

→ When BG=0, RD & WR signals allow CPU to communicate with internal DMA registers. When BG=1, DMA communicates with RAM through RD & WR lines.

2) Input-Output Processor (IOP):- Processor with direct memory access capability that communicates with I/O device is called I/O Processor. Processor accesses memory by cycle stealing. Processor can execute a channel program stored in main memory. CPU initiates the channel by executing a channel I/O class instruction and once initiated, channel operates independently of the CPU.

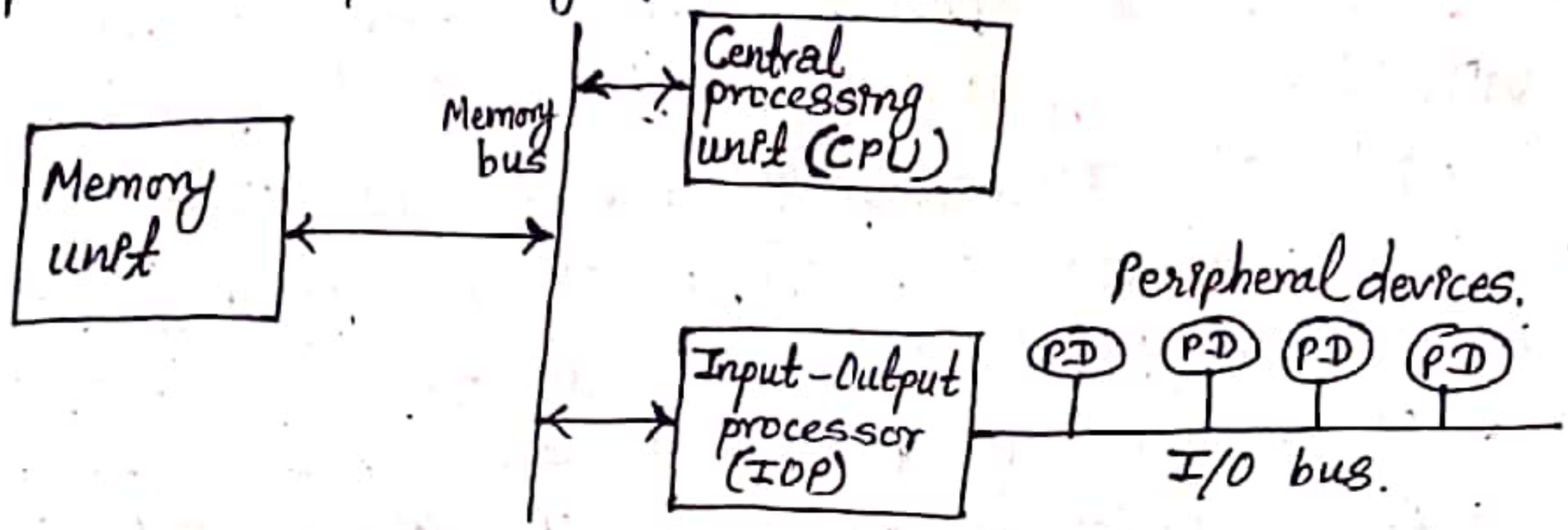


fig. Block diagram of computer with I/O processor.

⊛ CPU-IOP communication:

Communication between CPU and IOP may take different forms depending on the particular computer used. Mostly, memory unit acts as a memory center where each processor leaves information for the other.

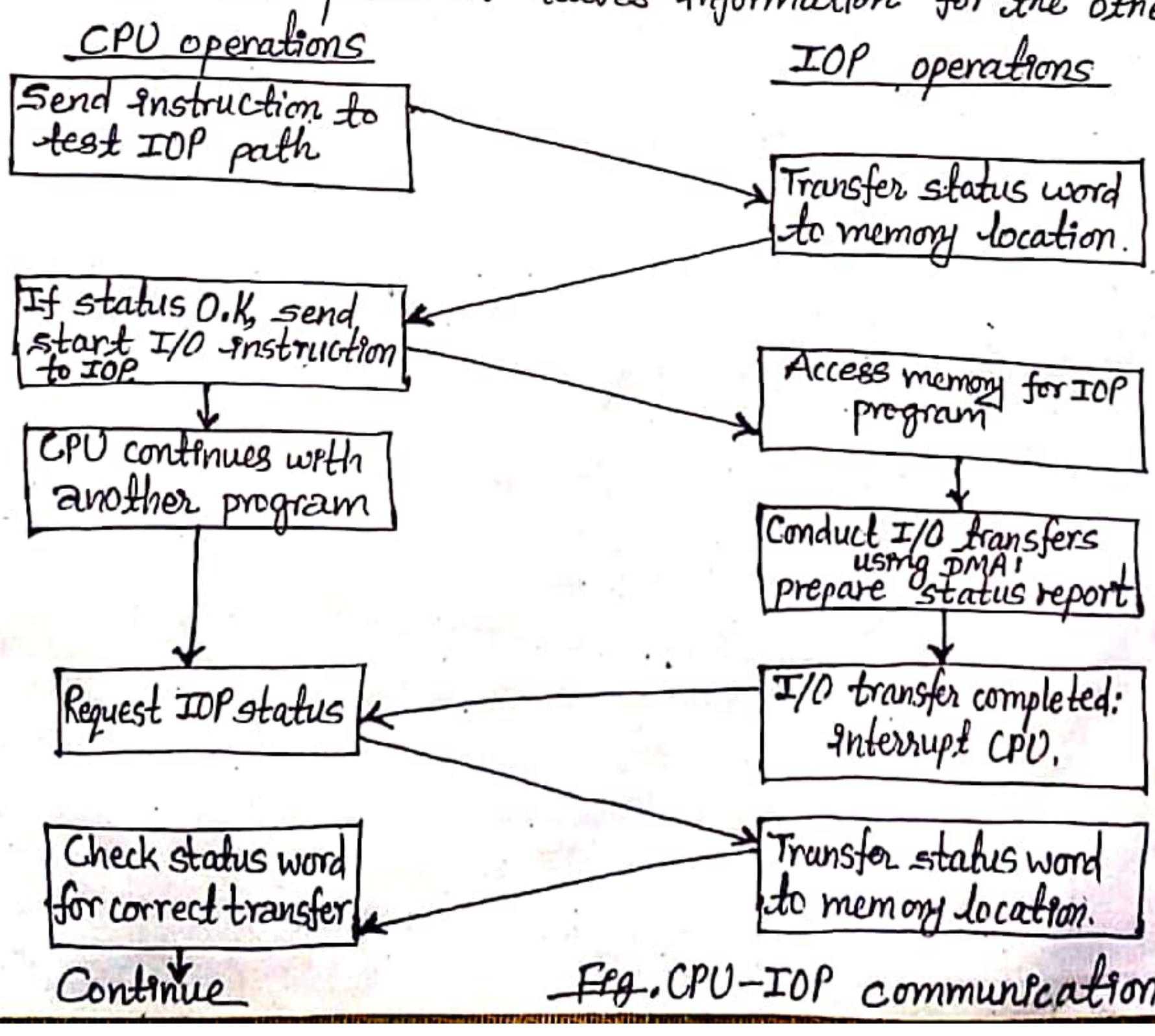


fig. CPU-IOP communication.

Mechanism: 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 bits of the status word indicate the condition of IOP and I/O device. CPU then checks status word to decide what to do next. If all is in order, CPU sends the instruction to start the I/O transfer. The memory address received with this instruction tells the IOP where to find its program. When IOP terminates the transfer using DMA, it sends an interrupt request to CPU. The CPU responds by issuing an instruction to read the status from the IOP and then IOP answers by placing the status report into specified memory location. By inspecting the bits in status word, CPU determines whether the I/O operation was completed satisfactorily and the process is repeated again.