Input / Output Subsystem
There are three basic forms of input and output systems –

- **Programmed I/O**
- **Interrupt driven I/O**
- **Direct Memory Access(DMA)**

With programmed I/O, the processor executes a program that gives its direct control of the I/O operation, including sensing device status, sending a read or write command, and transferring the data.
With interrupt driven I/O, the processor issues an I/O command, continues to execute other instructions, and is interrupted by the I/O module when the I/O module completes its work.
In Direct Memory Access (DMA), the I/O module and main memory exchange data directly without processor involvement.

With both programmed I/O and Interrupt driven I/O, the processor is responsible for extracting data from main memory for output operation and storing data in main memory for input operation.
To send data to an output device, the CPU simply moves that data to a *special memory location* in the I/O address space if I/O mapped input/output is used or to an address in the memory address space if memory mapped I/O is used.

Data	I/O Address Space (in memory)	if I/O mapped input/output is used
	memory address space	if memory mapped I/O is used

To read data from an input device, the CPU simply moves data from the address (I/O or memory) of that device into the CPU.

Input/Output Operation: The input and output operation looks very similar to a memory read or write operation except it usually takes *more time* since peripheral devices are slow in speed than main memory modules.

The working principle of the three methods for input of a Block of Data is shown in the Figure.

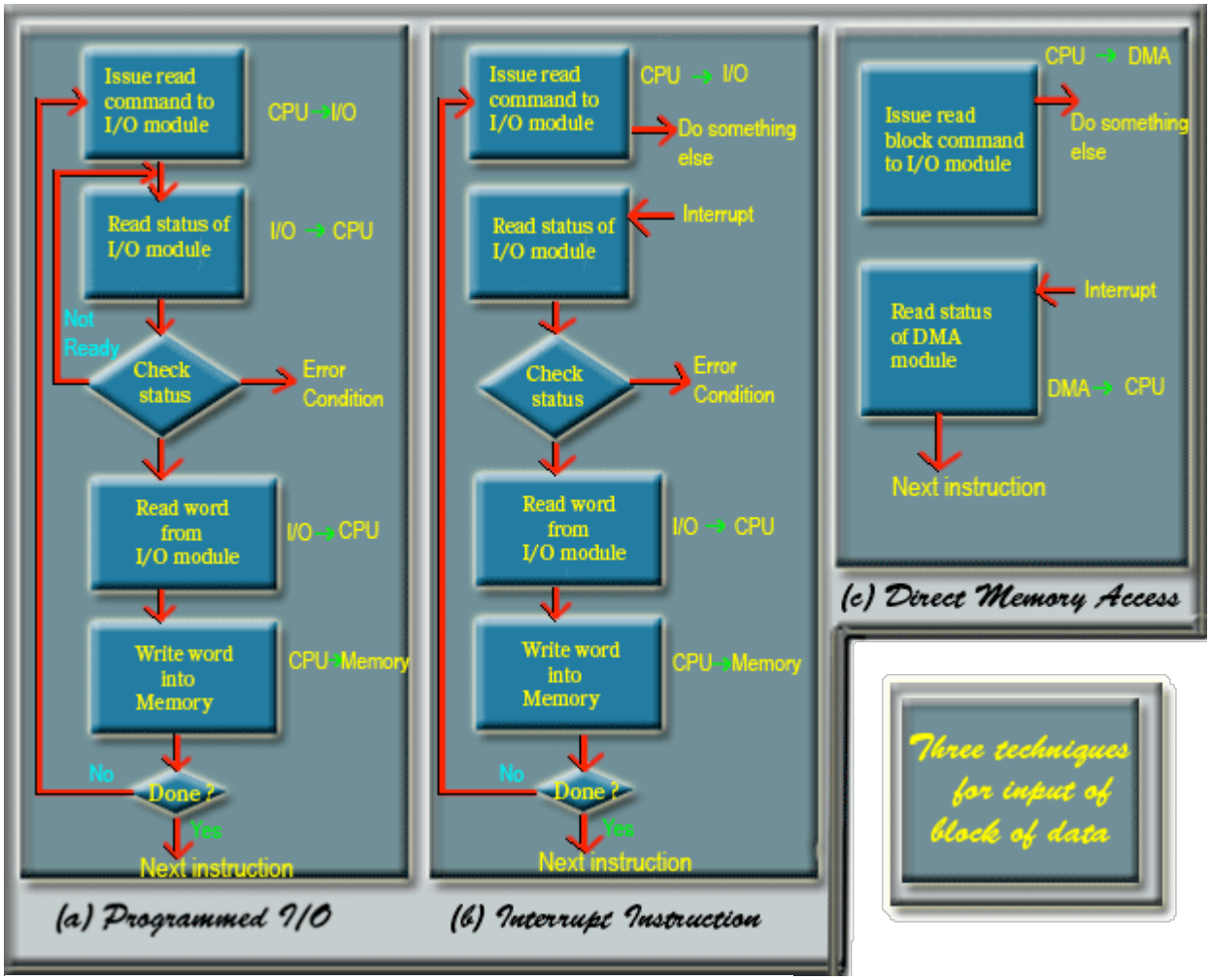


Figure: Working of three techniques for input of block of data.

Input/Output Port

An *I/O port* is a device that looks like a memory cell to the computer but contains connection to the outside world.

An *I/O port* typically uses a *latch*. When the CPU writes to the address associated with the latch, the latch device captures the data and makes it available on a set of wires external to the CPU and memory system.

The *I/O ports* can be *read-only*, *write-only*, or *read/write*. The *write-only* port is shown in the Figure.

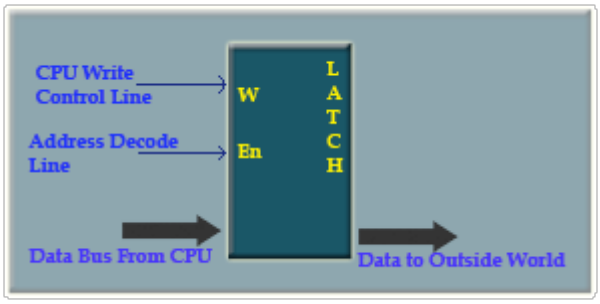


Figure : The write only port

Programmed I/O

Programmed I/O (PIO) refers to data transfers initiated by a CPU under driver software control to access registers or memory on a device.

The CPU issues a command then waits for I/O operations to be complete. As the CPU is faster than the I/O module, the problem with programmed I/O is that the CPU has to wait a long time for the I/O module of concern to be ready for either reception or transmission of data. The CPU, while waiting, must repeatedly check the status of the I/O module, and this process is known as Polling. As a result, the level of the performance of the entire system is severely degraded.

Programmed I/O basically works in these ways:

- CPU requests I/O operation
- I/O module performs operation
- I/O module sets status bits
- CPU checks status bits periodically
- I/O module does not inform CPU directly
- I/O module does not interrupt CPU
- CPU may wait or come back later

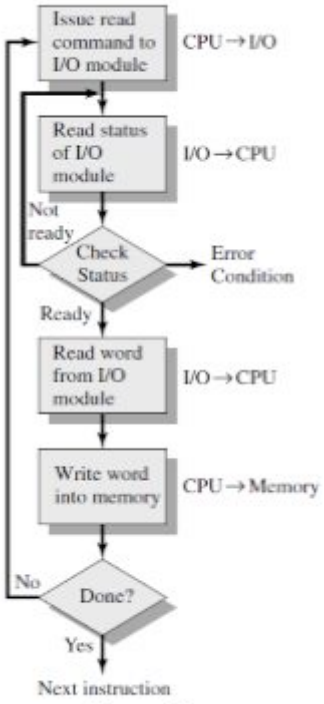


Figure 6

- As the CPU is faster than the I/O module, the problem with programmed I/O is that the CPU has to wait a long time for the I/O module of concern to be ready for either reception or transmission of data.

  i. The CPU, while waiting, must repeatedly check the status of the I/O module, and this process is known as Polling.

  ii. Therefore, the level of the performance of the entire system is severely degraded.

Programmed I/O

* It used only in some low-end microcomputers.
* It has single input and single output instruction.
* Each instructions selects one I/O device (by number) and transfers a single character (byte)
* Four registers: input status and character, output status and character.

Programmed-driven I/O means the program is polling or checking some hardware item e.g. mouse within a loop.
For Interrupt driven I/O, the same mouse will trigger a signal to the program to process the mouse event.

Advantage of Programmed Driven: easy to program and understand

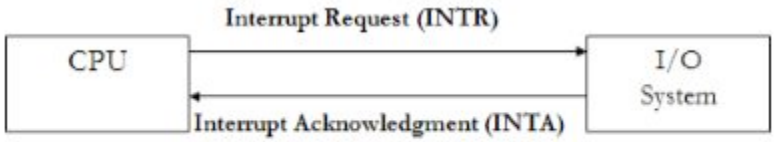
Disadvantages: slow and inefficient

Question: Write short note on Interrupt driven I/O.

- This technique is used to overcome the limitation of programmed I/O.
- In interrupt driven I/O, instead of making the processor to verify the status of I/O module. It is the responsibility of I/O module to intimate the processor by interrupt signal.
- CPU responds to interrupt signals and stores the return address from the program counter (PC) into the memory stack and then the control branches to a interrupt service routine (ISR).
- ISR processes the required I/O Transfer.
- After completion of executing interrupt routine CPU returns to previous program and continue what it was doing before.

Interrupt:

- An interrupt or exception causes CPU to transfer the control temporarily from its current program to another program i.e. interrupt handler.
- Block Diagram for Interrupt Driven I/O



Transfer control from main program to Interrupt Handler.

Interrupt-driven I/O

Although Interrupt relieves the CPU of having to wait for the devices, but it is still inefficient in data transfer of large amount because the CPU has to transfer the data word by word between I/O module and memory.

Below are the basic operations of Interrupt:

- CPU issues read command
- I/O module gets data from peripheral whilst CPU does other work
- I/O module interrupts CPU
- CPU requests data
- I/O module transfers data

For Interrupt driven I/O, the same mouse will trigger a signal to the program to process the mouse event.

Advantage of Interrupt Driven: fast and efficient
Disadvantage: Can be tricky to write if you are using a low level language. Can be tough to get the various pieces to work well together. Usually done by the hardware manufacturer or the OS maker e.g. Microsoft.

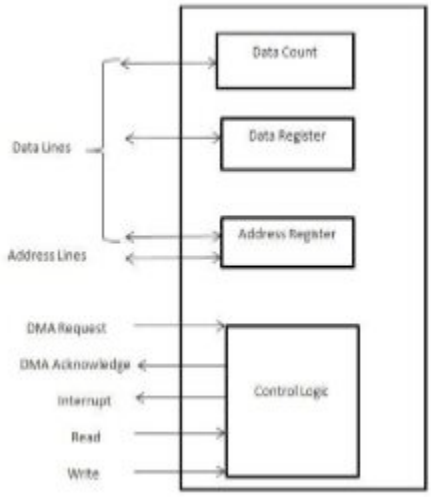
* Primary disadvantage of programmed I/O is that CPU spends most of its time in a tight loop waiting for the device to become ready. This is called busy waiting.
* With interrupt-driven I/O, the CPU starts the device and tells it to generate an interrupt when it is finished.

Question: Write short note on DMA.

Direct Memory Access (DMA)
Direct Memory Access (DMA) means CPU grants I/O module authority to read from or write to memory without involvement. DMA module controls exchange of data between main memory and the I/O device. Because of DMA device can transfer data directly to and from memory, rather than using the CPU as an intermediary, and can thus relieve congestion on the bus. CPU is only involved at the beginning and end of the transfer and interrupted only after entire block has been transferred.
Direct Memory Access needs a special hardware called DMA controller (DMAC) that manages the data transfers and arbitrates access to the system bus. The controllers are programmed with source and destination pointers (where to read/write the data), counters to track the number of transferred bytes, and settings, which includes I/O and memory types, interrupts and states for the CPU cycles.
DMA increases system concurrency by allowing the CPU to perform tasks while the DMA system transfers data via the system and memory busses. Hardware design is complicated because the DMA controller must be integrated into the system, and the system must allow the DMA controller to be a bus master. Cycle stealing may also be necessary to allow the CPU and DMA controller to share use of the memory bus.

> When large volumes of data are to be moved, a more efficient technique is required: direct memory access (DMA).

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



- It is also referred to as cycle stealing, because the DMA module in effect steals a bus cycle. When the processor wishes to read or write a block of data, it issues a command to the DMA module, by sending to the DMA module the following information:

i. Whether a read or write is requested, using the read or write control line between the processor and the DMA module

ii. The address of the I/O device involved, communicated on the data lines

iii. The starting location in memory to read from or write to, communicated on the data lines and stored by the DMA module in its address register

iv. The number of words to be read or written, again communicated via the data lines and stored in the data count register

1. The Control Logic in the DMA module is responsible for the generation of control signals.
2. The processor then continues with other work. It has delegated this I/O operation to the DMA module. The DMA module transfers the entire block of data, one word at a time, directly to or from memory, without going through the processor. When the transfer is complete, the DMA module sends an interrupt signal to the processor.
3. Thus, the processor is involved only at the beginning and end of the transfer. In the instruction cycle the processor may be suspended. In each case, the processor is suspended just before it needs to use the bus. The DMA module then transfers one word and returns control to the processor.

4. Note that this is not an interrupt; the processor does not save a context and do something else. Rather, the 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.**

With DMA, the CPU can process other tasks while data transfer is being performed. The transfer of data is first initiated by the CPU. During the transfer of data between the DMA channel and I/O device, the CPU performs other tasks. When the data transfer is complete, the CPU receives an interrupt request from the DMA controller.

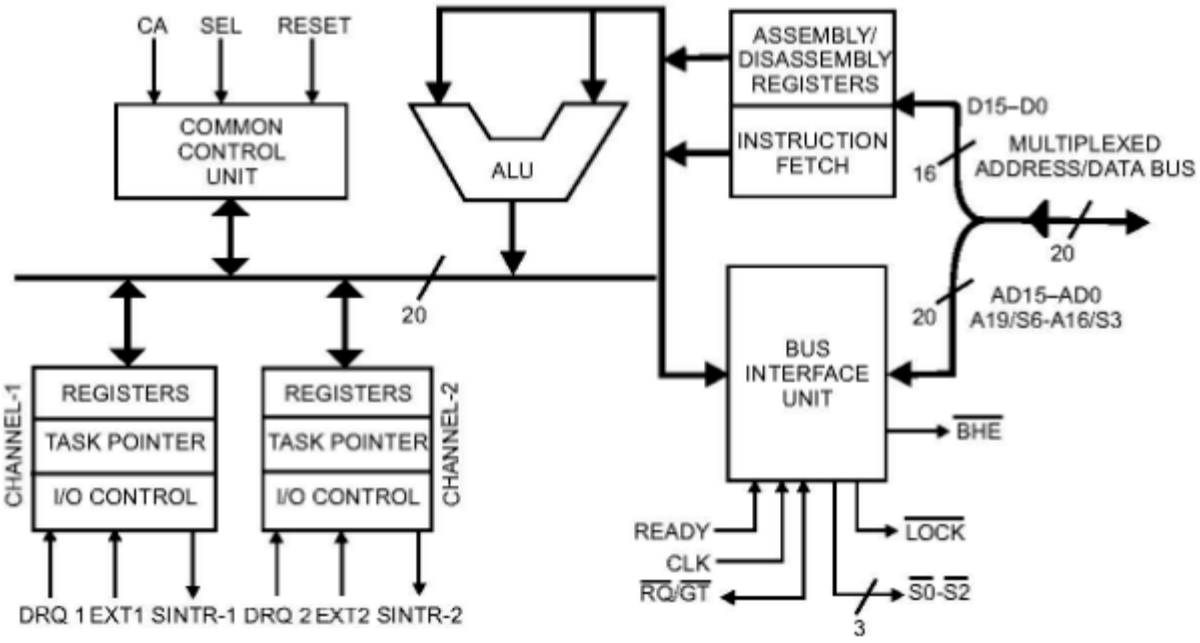
Question: 8089 I/O Processor

1. 8089 is an I/O processor.
2. It was available for use with the 8086/8088 central processor.
3. It uses the same programming technique as 8087 for I/O Operations, such as transfer of data from memory to a peripheral device.

Features:

1. 8089 has very high speed DMA capability.
2. It has 1 MB address capability.
3. It is compatible with iAPX 86, 88.
4. It supports local mode and remote mode I/O processing.
5. 8089 allows mixed interface of 8-and 16-bit peripherals, to 8-and 16-bit processor buses.
6. It supports two I/O channels.
7. Multibus compatible system interface.
8. Memory based communications with CPU.

Block Diagram:



I) Common Control Unit (CCU):

8089 I/O Processor has two channels.¬

The activities of these two channels are controlled by CCU.¬

CCU determines which channel—1 or 2 will execute the next cycle.¬

In a particular case where both the channels have equal priority, an interleave procedure is adopted in which each alternate cycle is assigned to channels 1 and 2.¬

II) Arithmetic & Logic Unit (ALU):

ALU is used to perform the Arithmetic¬ & Logical operations.

It performs Arithmetic Operations like Addition, Subtraction¬ & Logical Operations like AND, OR, EX-OR etc.

ALU looks after the branching decisions.¬

III) Assembly/Disassembly registers:

This registers permits 8089 to deal with 8-or 16-bit data width devices or a mix of both.¬

In a particular case of an 8–bit width I/O device inputting data to a 16-bit memory interface, 8089 capture two bytes from the device and then write it into the assigned memory locations with the help of assembly/disassembly register.¬

IV) Bus Interface Unit (BIU):

Fetch the instruction or data from primary memory.¬

Read / Write of data from / to primary memory.¬

I/O of data from / to peripheral ports.¬

Address generation for memory reference.¬

V) Instruction Fetch:

It is used to fetches the instructions from the external memory and stores them in the Queue to be executed further.¬