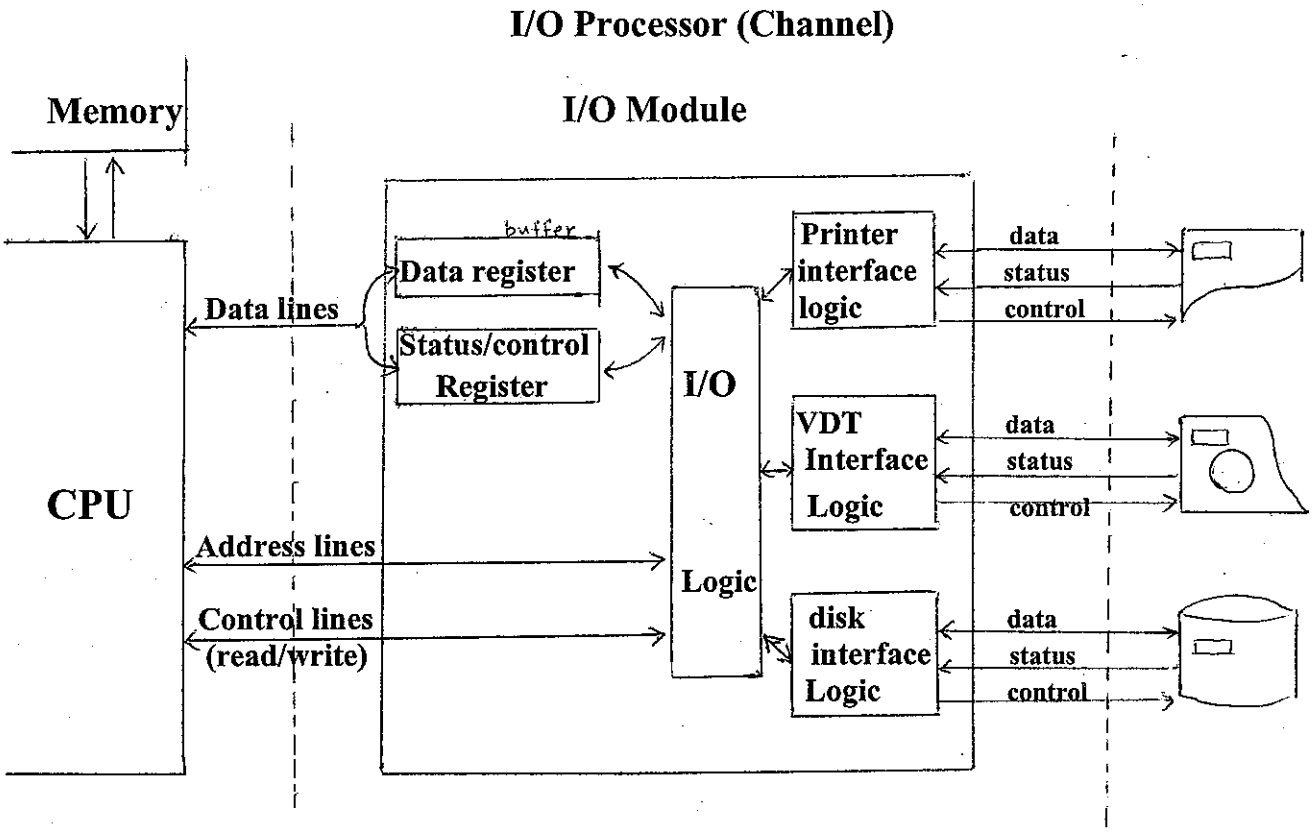
7. INPUT/OUTOUT



Functions of I/O module

1. **control and timing** – to share system bus
2. **communication to CPU**
3. **communication to devices**
4. **data buffering** –speed difference between memory/cpu and peripherals
5. **error detection** – parity check

I/O Methods

method \ interrupt?	No	Yes
I/O to memory transfer through CPU	Programmed I/O (Polled I/O)	Interrupt-driven I/O
Direct I/O to memory transfer		Direct Memory Access

Interrupts

- Events that alter (interrupt) the normal flow of execution in the system

When:

- I/O requests
- Arithmetic errors, such as division by zero
- Overflow / underflow
- Hardware errors, such as memory parity error
- User-defined break points (in debugger)
- Page faults
- Invalid instruction

:

Note. An interrupt is not a process.

➔ Need special way to handle interrupts.

ATM, Embedded system for monitoring

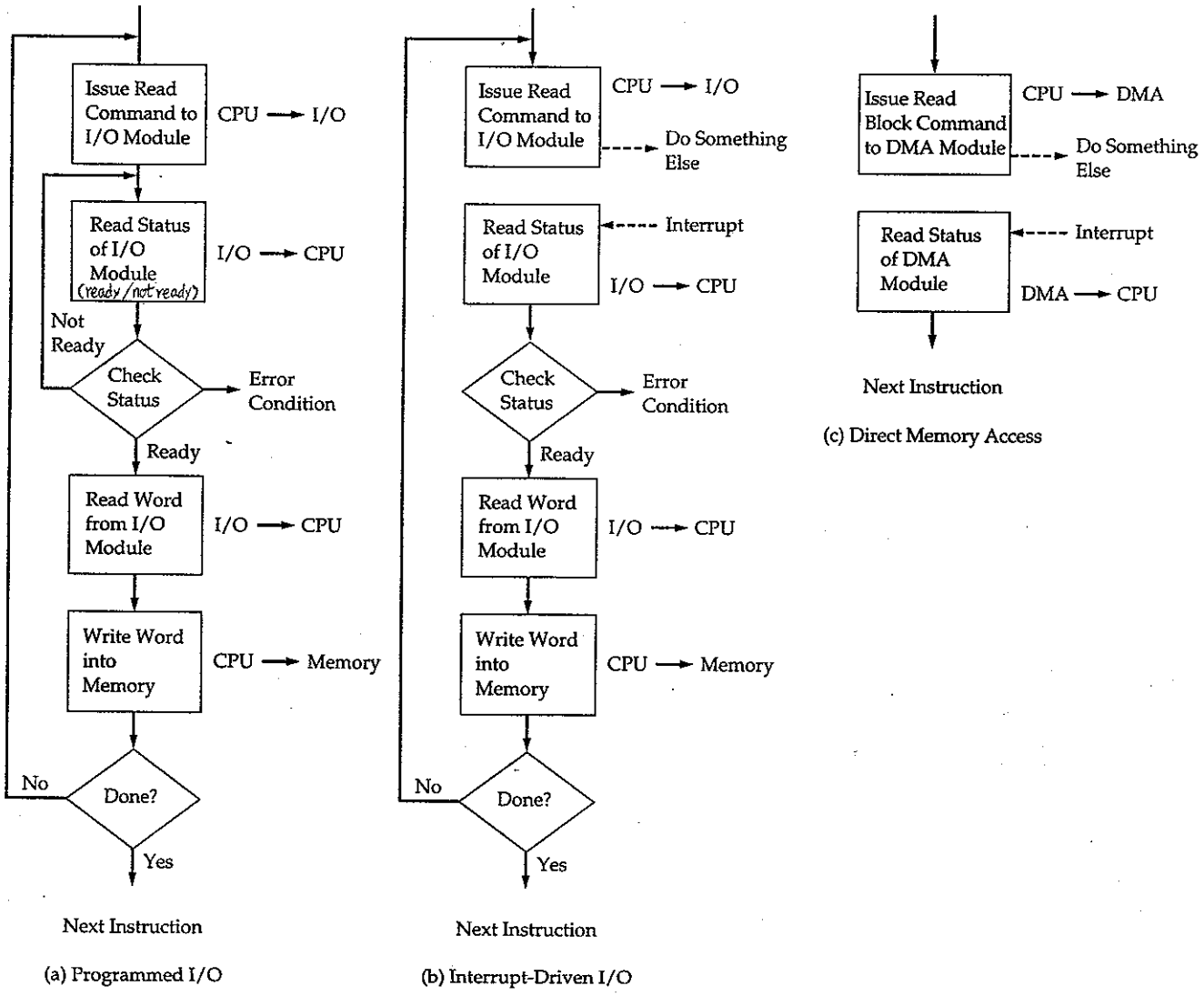


FIGURE 7.4. Three techniques for input of a block of data

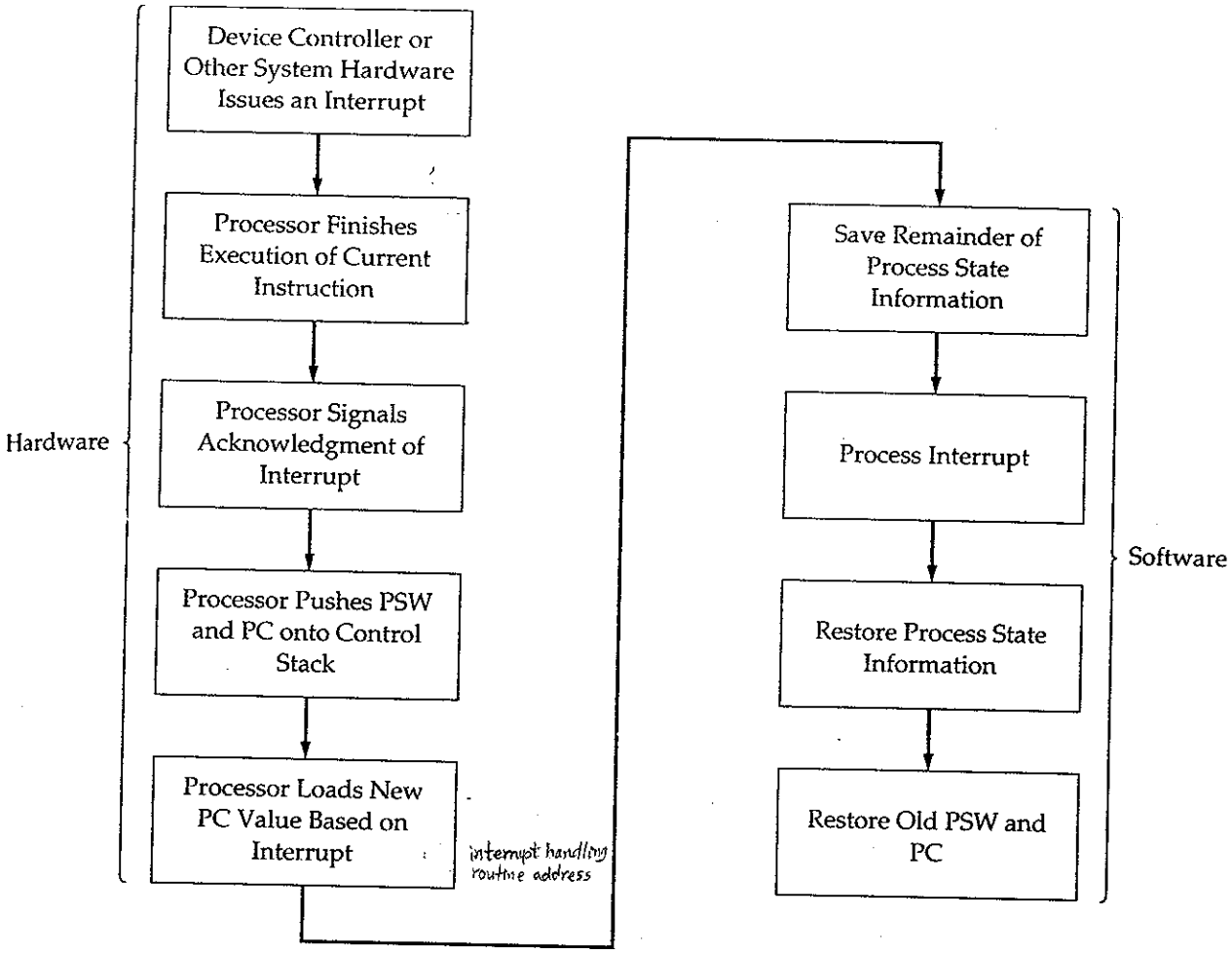
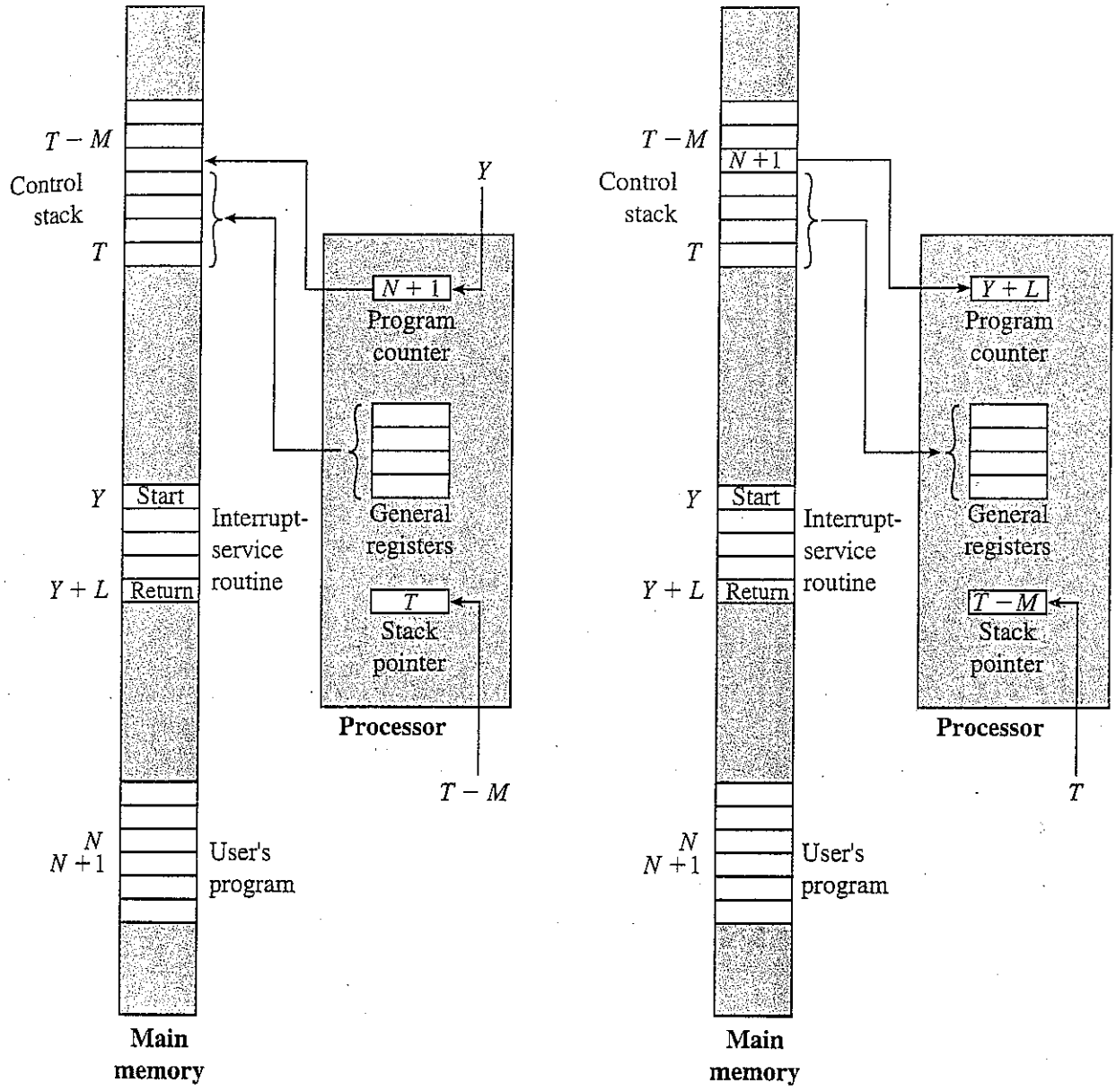


FIGURE 7.6. Simple interrupt processing



(a) Interrupt occurs after instruction at location N

(b) Return from interrupt

Figure 7.7 Changes in Memory and Registers for an Interrupt

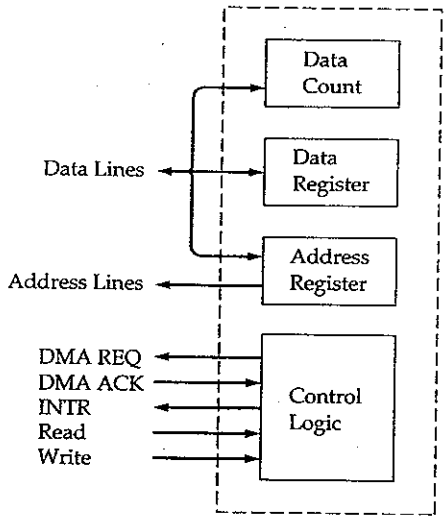


FIGURE 7.11. Typical DMA block diagram

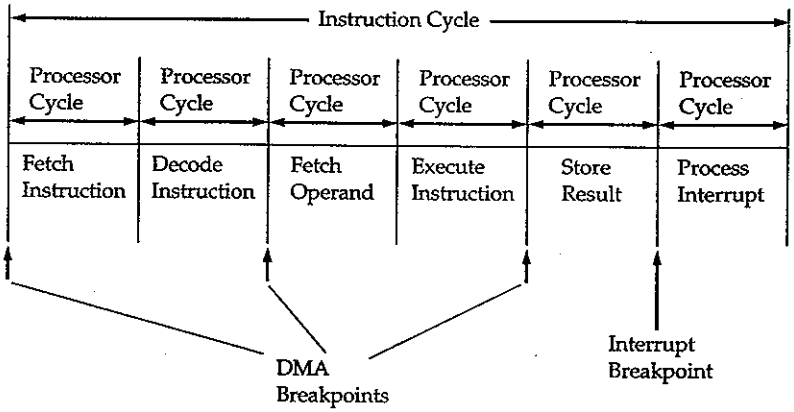
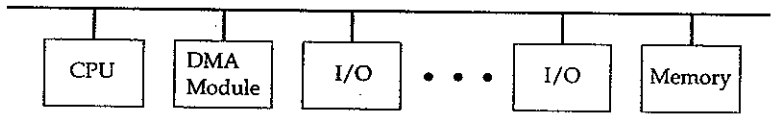
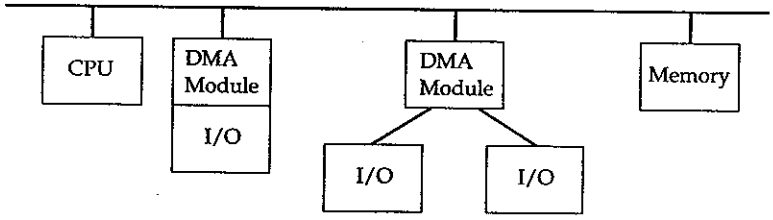


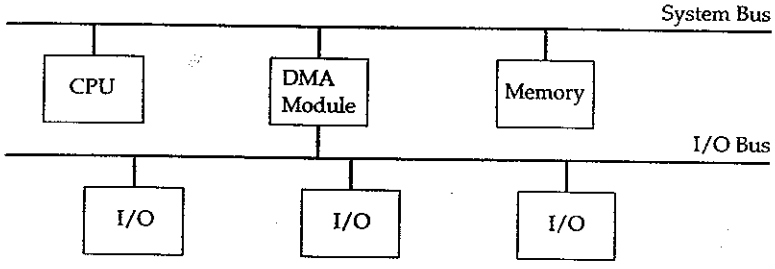
FIGURE 7.12. DMA and interrupt breakpoints during an instruction cycle



(a) Single-Bus, Detached DMA



(b) Single-Bus, Integrated DMA-I/O



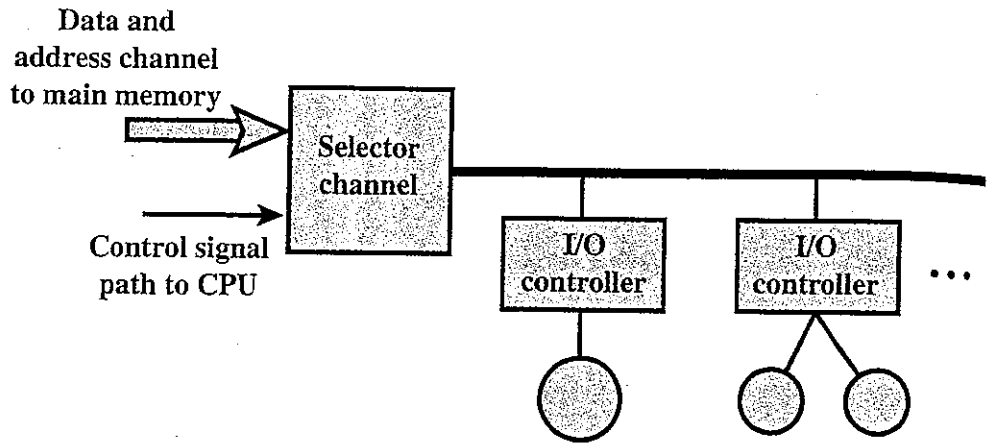
(c) I/O Bus

FIGURE 7.13. Possible DMA configuration

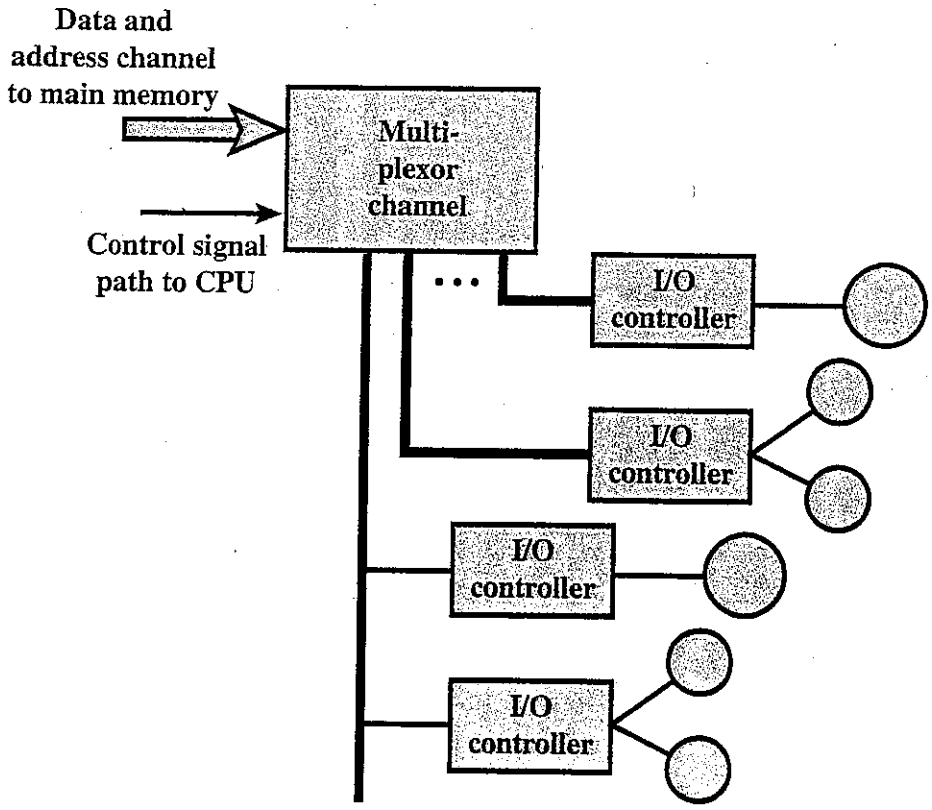
7.6 I/O CHANNELS AND PROCESSORS

The Evolution of the I/O Function

1. The CPU directly controls a peripheral device. This is seen in simple microprocessor-controlled devices.
2. A controller or I/O module is added. The CPU uses programmed I/O without interrupts. With this step, the CPU becomes somewhat divorced from the specific details of external device interfaces.
3. The same configuration as in step 2 is used, but now interrupts are employed. The CPU need not spend time waiting for an I/O operation to be performed, increasing efficiency.
4. The I/O module is given direct access to memory via DMA. It can now move a block of data to or from memory without involving the CPU, except at the beginning and end of the transfer.
5. The I/O module is enhanced to become a processor in its own right, with a specialized instruction set tailored for I/O. The CPU directs the I/O processor to execute an I/O program in memory. The I/O processor fetches and executes these instructions without CPU intervention. This allows the CPU to specify a sequence of I/O activities and to be interrupted only when the entire sequence has been performed.
6. The I/O module has a local memory of its own and is, in fact, a computer in its own right. With this architecture, a large set of I/O devices can be controlled, with minimal CPU involvement. A common use for such an architecture has been to control communication with interactive terminals. The I/O processor takes care of most of the tasks involved in controlling the terminals.



(a) Selector



(b) Multiplexor

Figure 7.15 I/O Channel Architecture