Input Output Organization

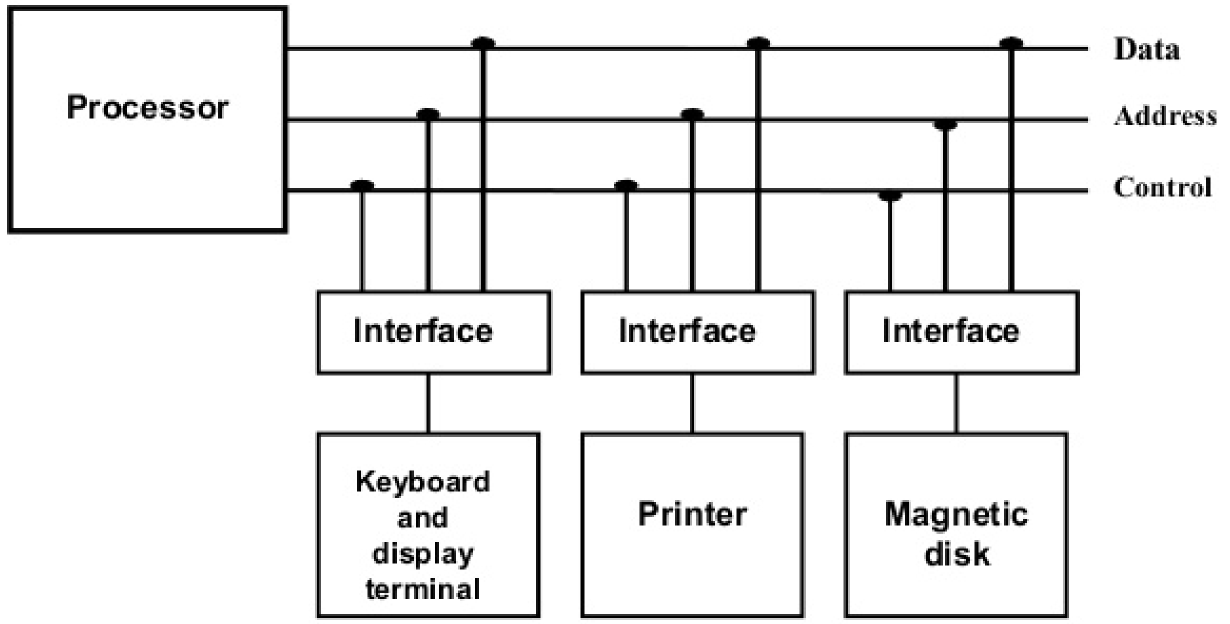
Input-output Subsystem(I/O)

- Provides the efficient mode of communication between system and the outside environment.
- I/O devices attached to the computer are also called ***peripherals.***

  Ex. Keyboard, display unit, printer, magnetic disk, magnetic tape.
- Peripherals may be analog/digital, serial/parallel, electromechanical, electromagnetic devices.

Input-Output Interface

- Provides a method for transferring information between internal storage and external I/O devices.
- Peripherals need special communication links to resolve the differences that exist between the central computer and each peripheral.



Connection of I/O bus to input-output devices

- Conversion of signal values may be required.
- Synchronization mechanisms are needed to compensate the speed mismatch.
- Data code and formats in peripherals may be different from the word format of CPU and memory.
- Operating modes of each peripherals are different and must be controlled so as not to disturb operation of other peripherals.

Interface Units

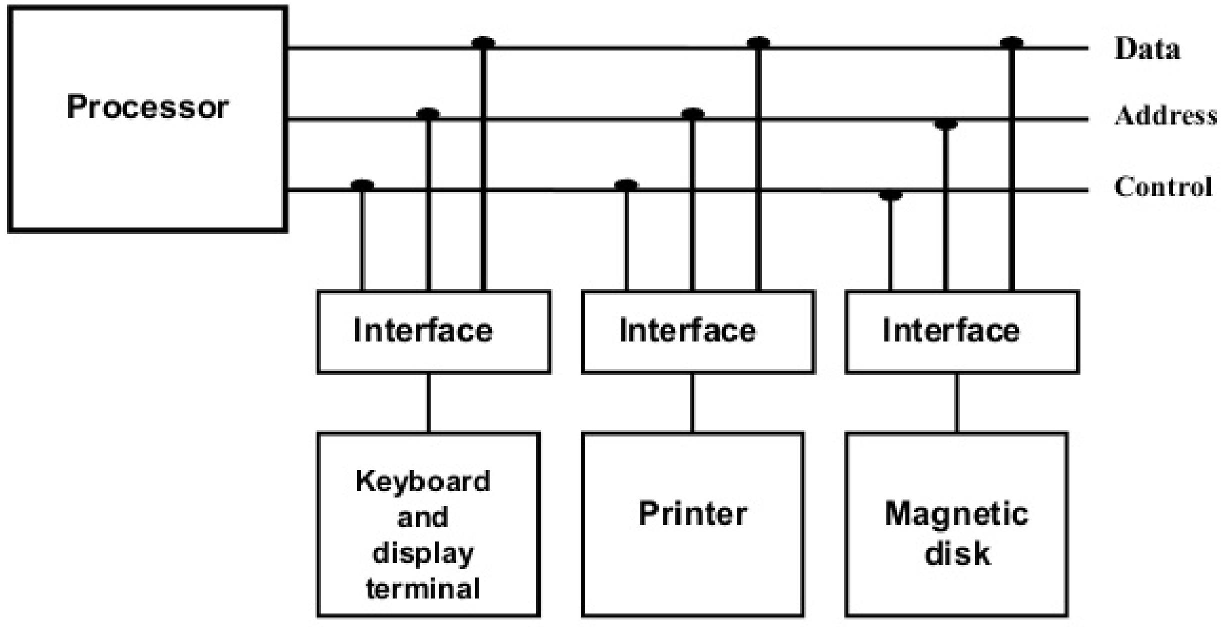
- Special hardware components used between CPU and peripherals to resolve the differences and to supervise and synchronize all I/O transfers.
- Interface: point of contact
- “To interface” means to attach components via their respective interface points for data exchange.

I/O Interface Functions

- **Data conversion** (digital and analog signals, Parallel and serial formats)
- **Synchronization** (matching speeds)
- **Device selection**

- In addition, each device may have its own <u>controller</u> that supervises the operation of the particular mechanism in the peripheral.
- Ex. Printer controller controls paper motion, print timing..

I/O Bus and Interface Modules



Connection of I/O bus to input-output devices

- Interface decodes the address and control received from the I/O bus and provides signals for the peripheral controller.

- Four types of I/O command:
  1. **Control command:** Activate the peripheral and inform what to do.
  2. **Status command:** to test various status conditions in the interface and the peripherals.
  3. **Data output command:** processor checks status and issues data output command - Interface transfer data from bus to its buffer register – then to the device.
  4. **Data input command:** interface receives data from peripheral, places it in its buffer register - Processors checks if data is available by using status command and issues data input command- The interface places data on data lines.

I/O Vs. Memory Bus

- In addition to I/O, the processor must communicate with the memory unit.
- Memory bus contains data, address and R/W control lines.
- Three ways that computer buses can be used to communicate with memory and I/O:

1. Separate buses for memory and I/O.
2. One common bus for both with separate control lines for each.
3. Common bus for both.

- Option 1: used in computers that provide separate I/O processors in addition to the CPU.
- Memory communicates with both CPU and I/O processor through a memory bus.
- I/O processor takes care of input-output tasks.
- It can fetch and execute its own instructions.

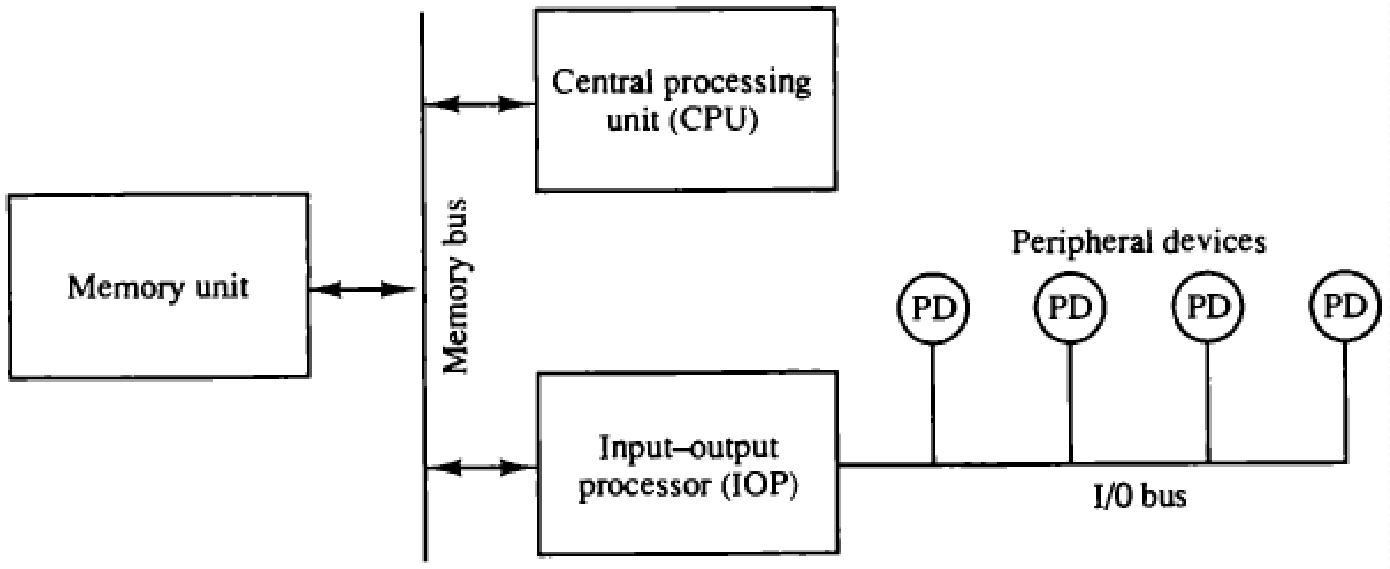


Figure 11-19 Block diagram of a computer with I/O processor.

Isolated I/O

- Common bus to transfer information between memory or I/O and CPU.
- Distinction between a memory transfer and I/O transfer is made through separate R/W lines
- **I/O transfer:** I/O read or write control lines are enabled.
- **Memory transfer:** Memory read/write control lines are enabled.

- This configuration isolates all I/O interface addresses from the addresses assigned to memory.
- CPU has distinct input and output instructions.
- Each instruction is associated with the address of an interface register.

Memory-Mapped I/O

- There is only one set of read and write signals and do not distinguish between memory and I/O addresses.
- No specific I/O instructions.
- Same address space for both memory and I/O, treats interface register as being part of memory system.
- The assigned addresses for interface registers cannot be used for memory words, which reduces the memory address range available.

Asynchronous Data Transfer

- Internal operations in a digital system are synchronized by means of clock pulses supplied by a common pulse generator.
- Clock pulses are applied to all registers within a unit.
- Different units such as a CPU and an I/O interface are designed independently.

Asynchronous Data Transfer

- **Synchronous:** If the registers in the interface share a common clock with the CPU registers, the transfer is said to be synchronous.
- **Asynchronous:** timing in each unit is independent from the other.
- In asynchronous, the units communicate via control signals:

  - Strobe pulse

  - Handshaking

Strobe Control

- The sequence of control during an asynchronous transfer depends on whether the transfer is initiated by source or destination unit.

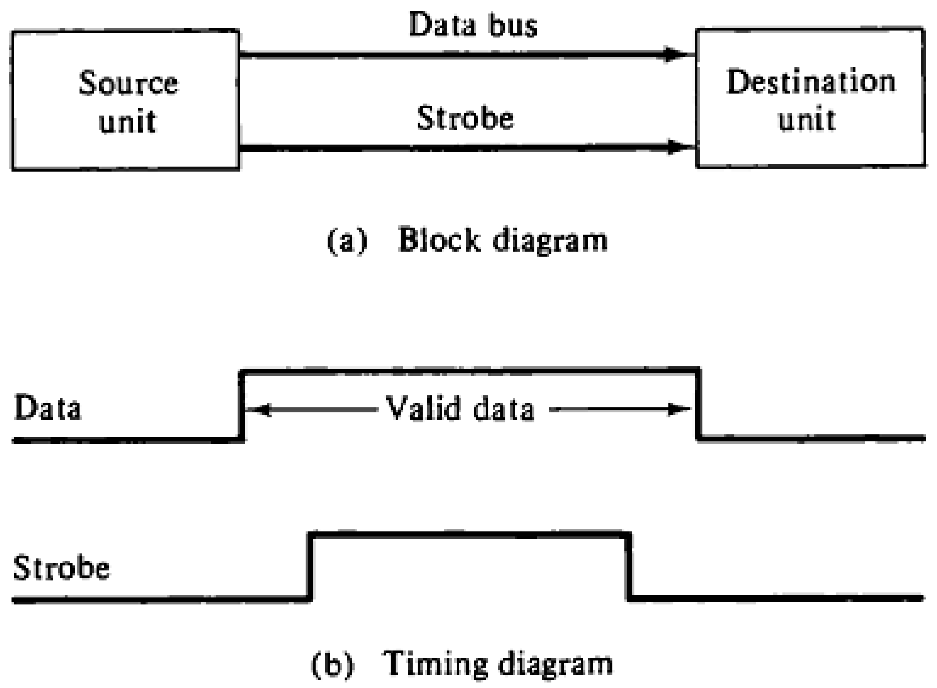


Figure 11-3 Source-initiated strobe for data transfer.

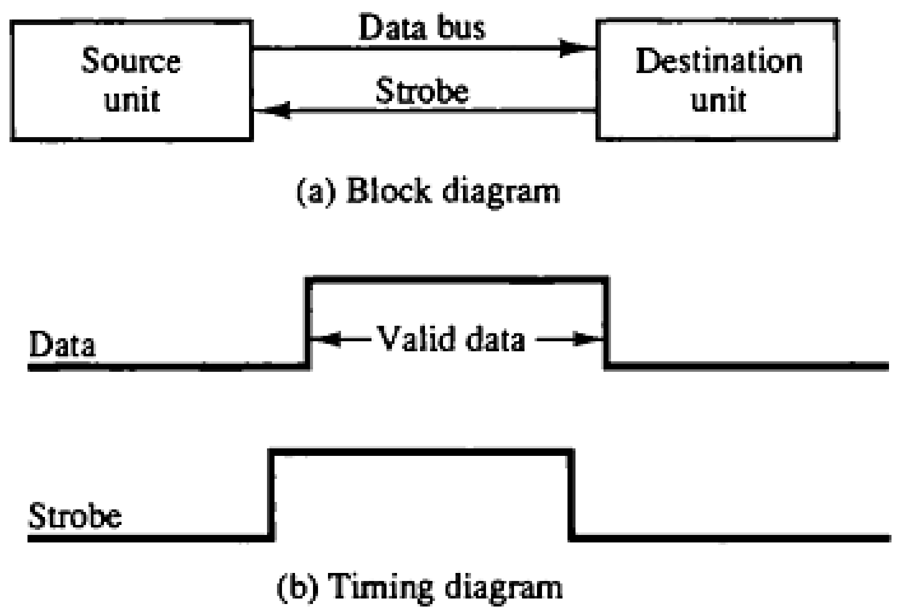
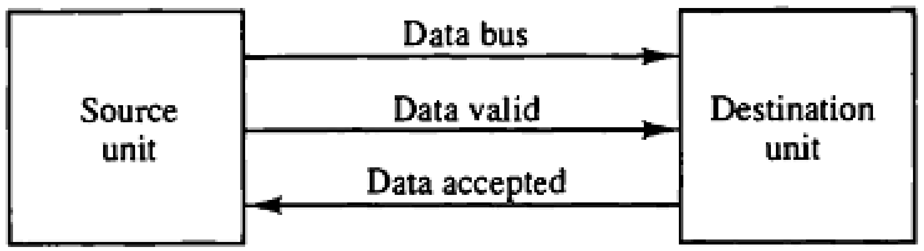


Figure 11-4 Destination-initiated strobe for data transfer.

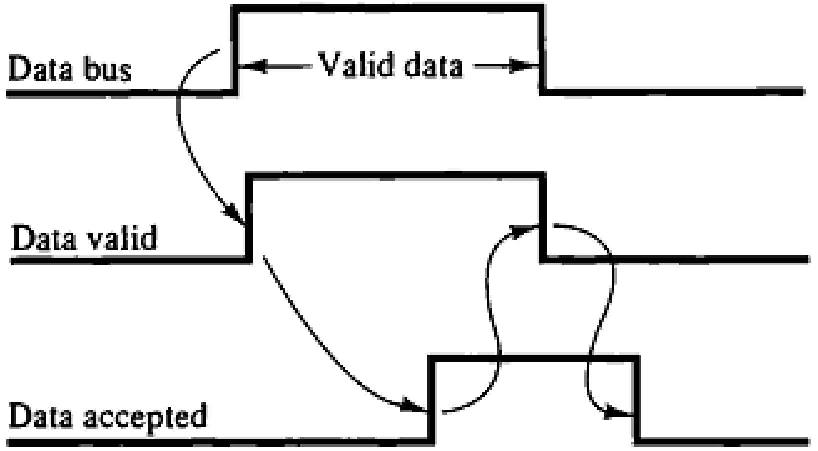
- Drawback of strobe method: no way to know if destination has actually received the data placed on the bus or whether the source has actually placed the data on data bus.

Handshaking

- Introduces second control signal to provide acknowledgement.
- Two-wire handshaking method.



(a) Block diagram



(b) Timing diagram

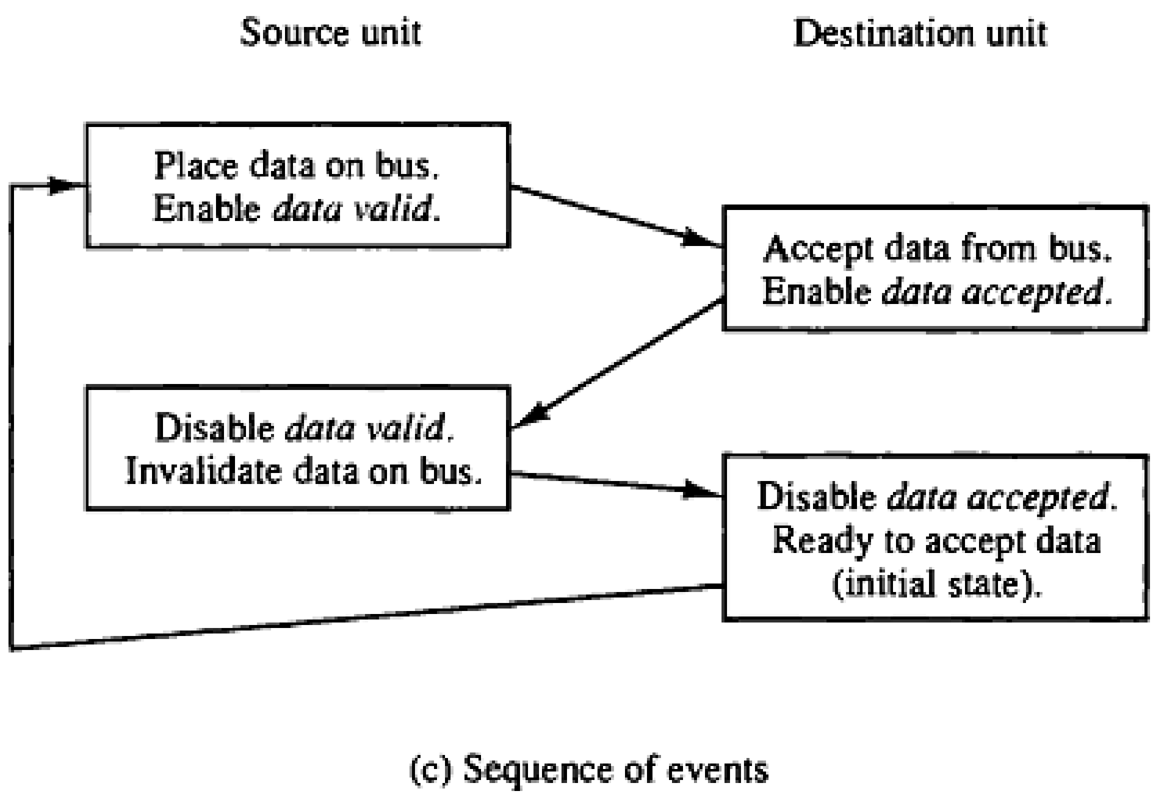
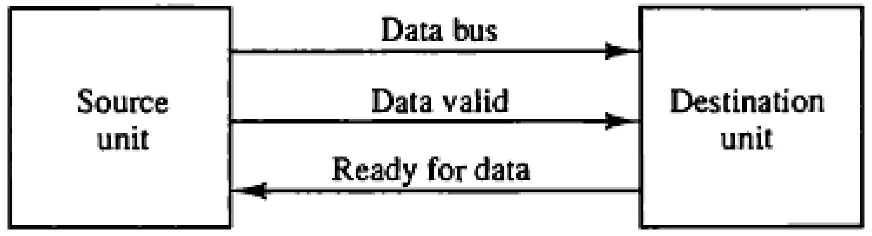
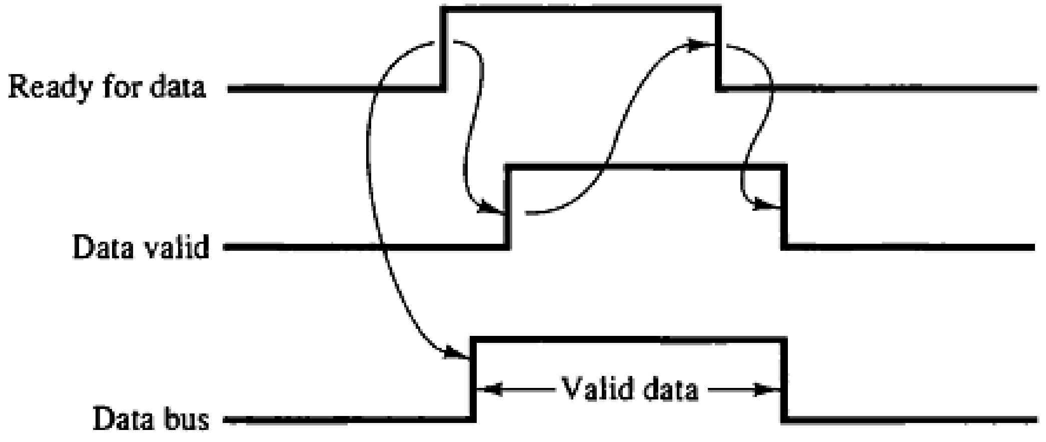


Figure 11-5 Source-initiated transfer using handshaking.

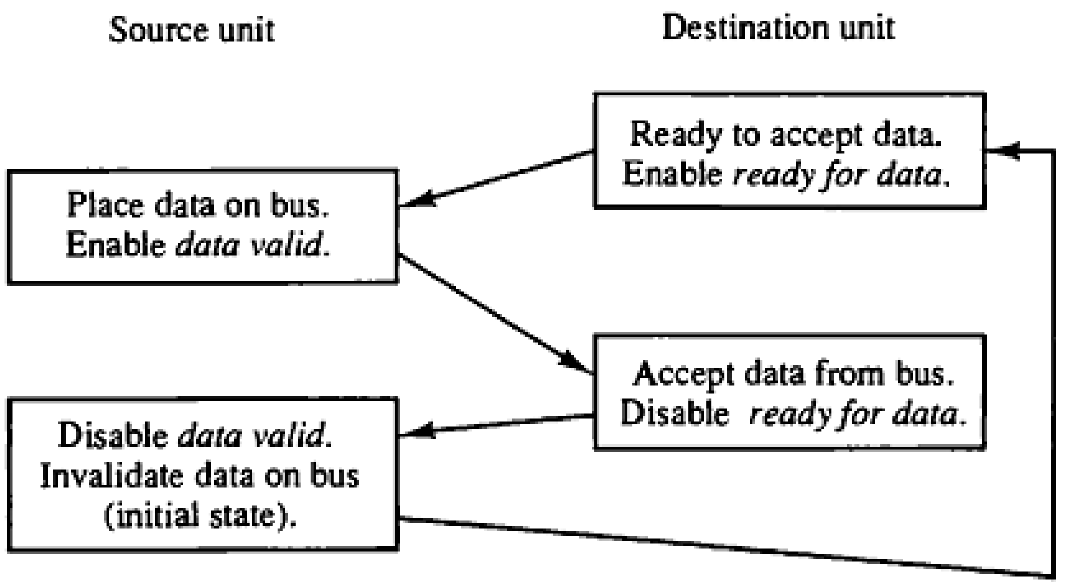
Figure 11-6 Destination-initiated transfer using handshaking.



(a) Block diagram



(b) Timing diagram


Source unit
Destination unit
Ready to accept data. Enable *ready for data*.
Place data on bus. Enable *data valid*.
Accept data from bus. Disable *ready for data*.
Disable *data valid*. Invalidate data on bus (initial state).

(c) Sequence of events

- Handshaking scheme provides a high degree of reliability.
- Timeout mechanism to detect if data transfer is not complete. (if the return handshake signal does not respond within a given amount of time, error is assumed)

Modes of Transfer

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

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

Programmed I/O

- Requires constant monitoring of the peripheral by the CPU.
- Once data transfer is initiated, the CPU is required to monitor the interface to see when a transfer a can be made again.
- Each byte is read into CPU register then transferred to memory.
- Time-consuming and keeps processor needlessly busy.

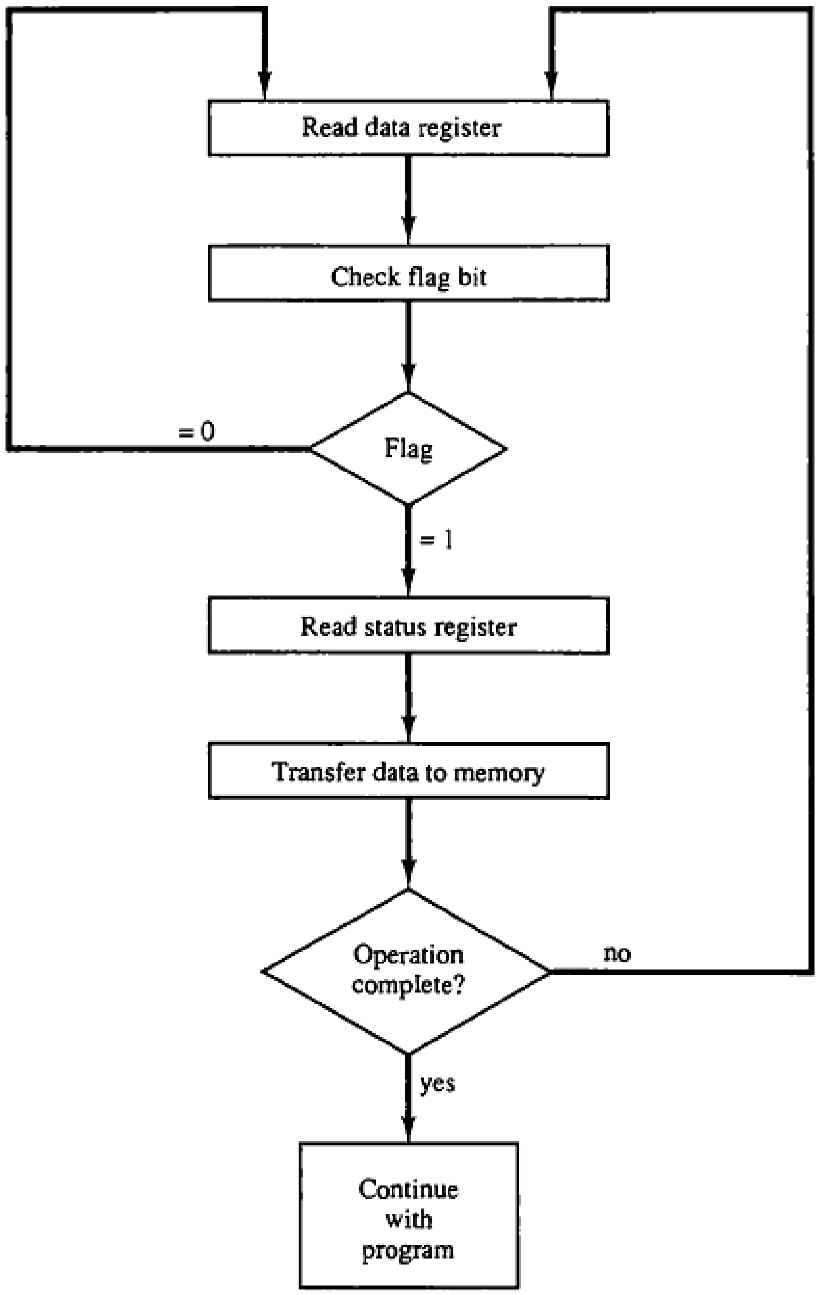


Figure 11-11 Flowchart for CPU program to input data.

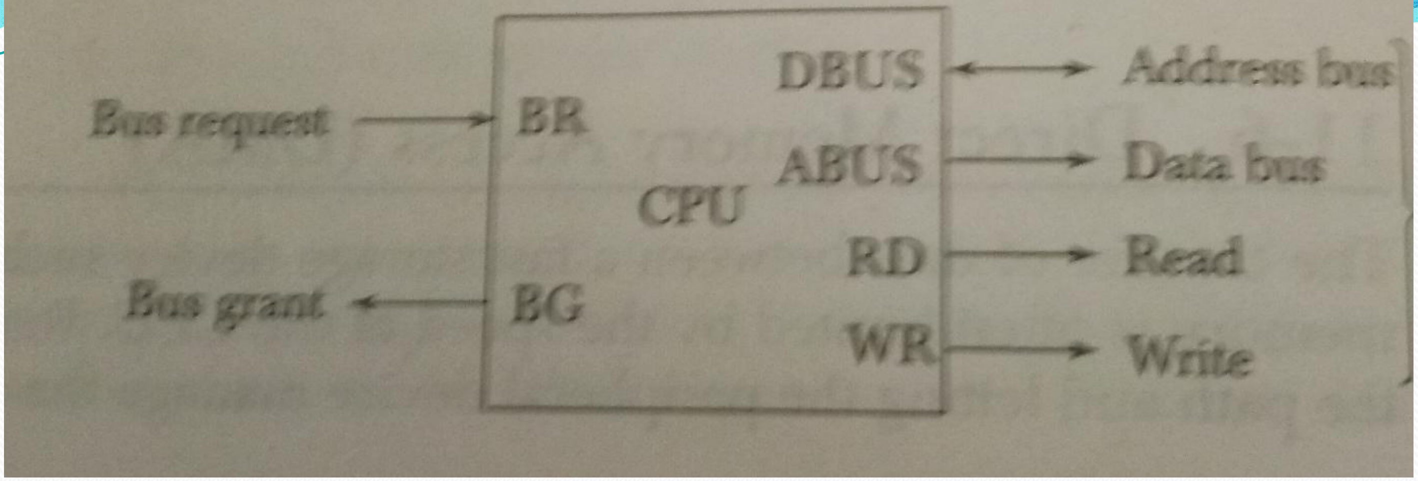
Interrupt-initiated I/O

- Let the interface inform the processor when it is ready to transfer data.
- Peripheral interface uses interrupt facility to inform CPU if device is ready for data transfer.
- CPU can execute another program in the meanwhile.
- When the flag is set, the CPU deviates from what it is doing to take care of the I/O transfer. (Branch to ISR)
- Then, return to the task.

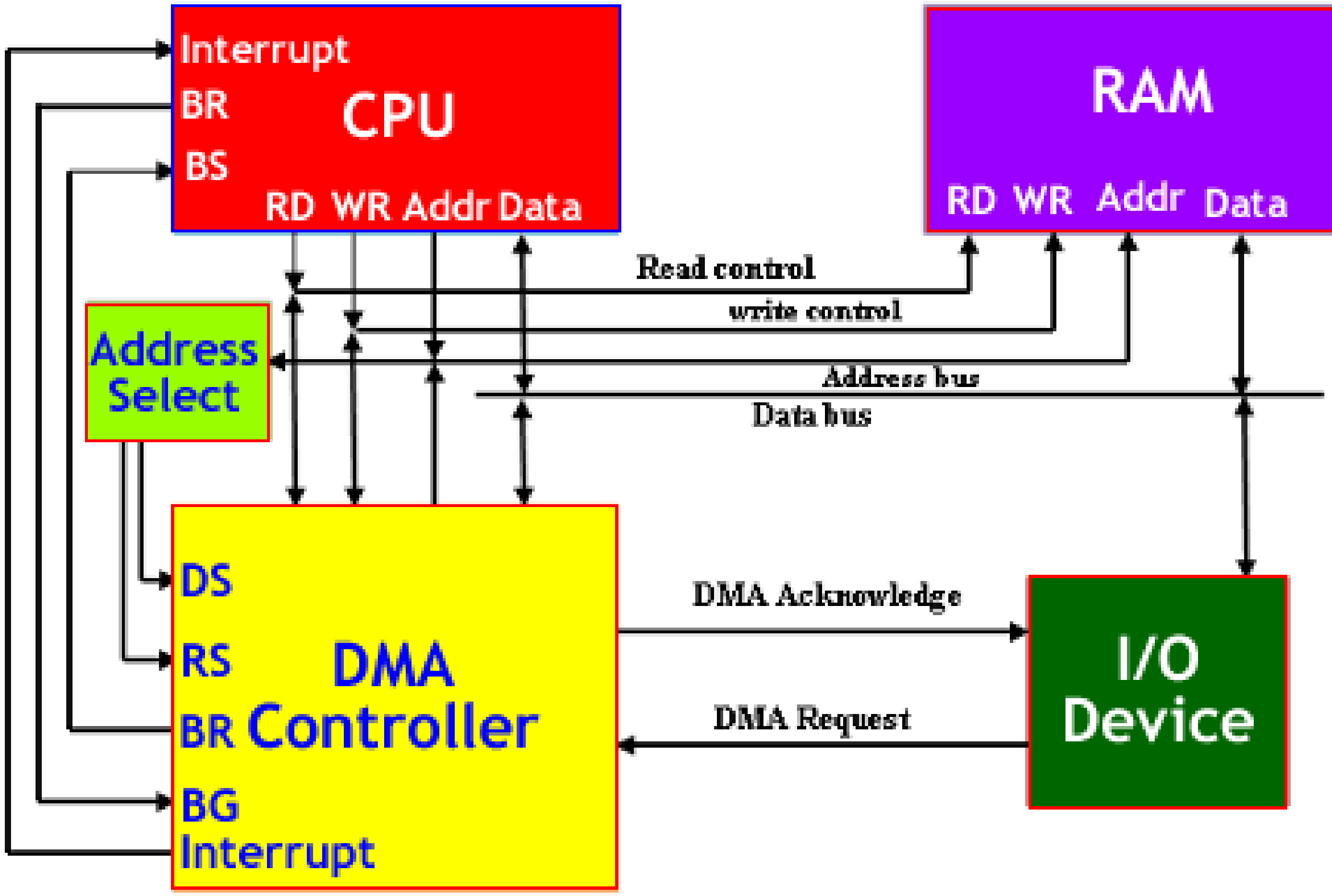
Direct Memory Access(DMA)

- A DMA controller manages the transfer directly between I/O device and memory.
- The DMA is first initialized by the CPU.
- CPU provides:
  - Starting address of the memory block
  - Word count
  - R/W
  - Control to start the DMA transfer
- Data transfer done directly between the device and memory under the control of the DMA.

Figure 11-16 CPU bus signals for DMA tra



- The DMA controller has : Address register, Word count register and control register.
- Once DMA receives start control, it starts transfer between peripheral and memory.(till wc =0)


Interrupt
BR
CPU
BS
RD WR Addr Data
RAM
RD WR Addr Data
Read control
write control
Address Select
Address bus
Data bus
DS
RS
DMA Controller
BR
BG
Interrupt
DMA Acknowledge
DMA Request
I/O Device

Direct Memory Access(DMA)

- DMA controller requests memory cycles through the memory bus.
- CPU delays its memory access operation to allow the direct memory I/O transfer.
- Ways of transfer:
  - Burst transfer
  - Cycle stealing

- DMA transfer is very useful for fast transfer of information
- Ex. between magnetic disk and memory.

4 X 4 FIFO Buffer

- **FIFO buffer:** Memory unit that stores information such that the item first in is the item first out.
- Output data is always in the same order in which the data entered the buffer.
- When placed between two units , it can input data at one transfer rate and output data at a different rate.
- Useful for synchronization when there is speed mismatch also when data are transferred asynchronously.

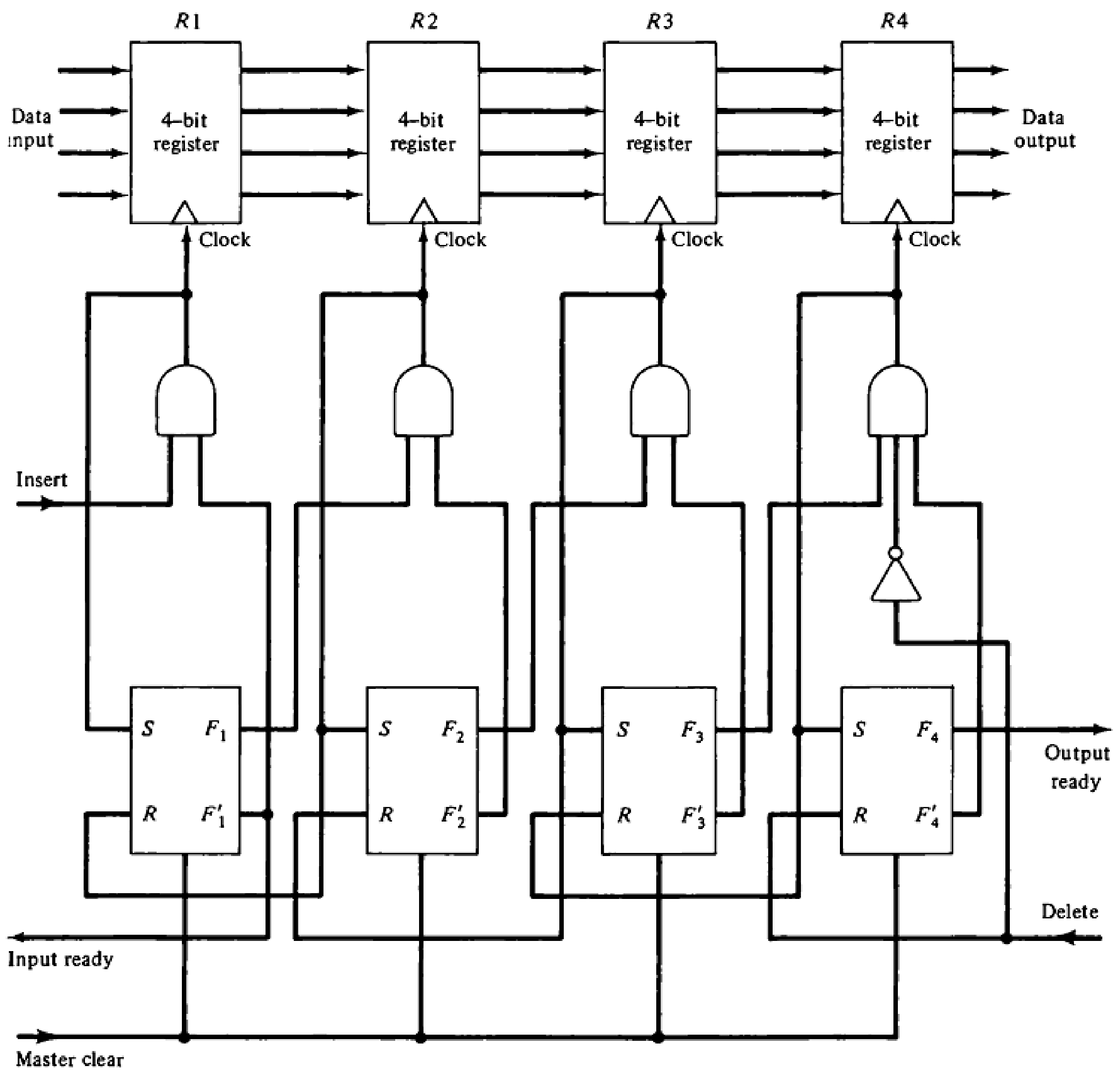


Figure 11-9 Circuit diagram of 4 × 4 FIFO buffer.

- F_i = 1 indicates that a 4-bit word is stored in the register RI.
- F_i=0, indicates the corresponding register does not contain valid data.
- F_i=1 and F_{i+1}=0 (F'_{i+1}=1) causes R(I+1) to accept data from RI.
- Data are inserted into the buffer provided that the *input ready* signal is enabled.
- The *output ready* control line is enabled when the last control flip-flop F4 is set, indicating there are valid data in R4.