Unit-7
Input-Output Organization

Marwadi University

Department of Computer Engineering

Computer Organization and Architecture 01CE1402

Asst.Prof. Kishan Makadiya

Outline

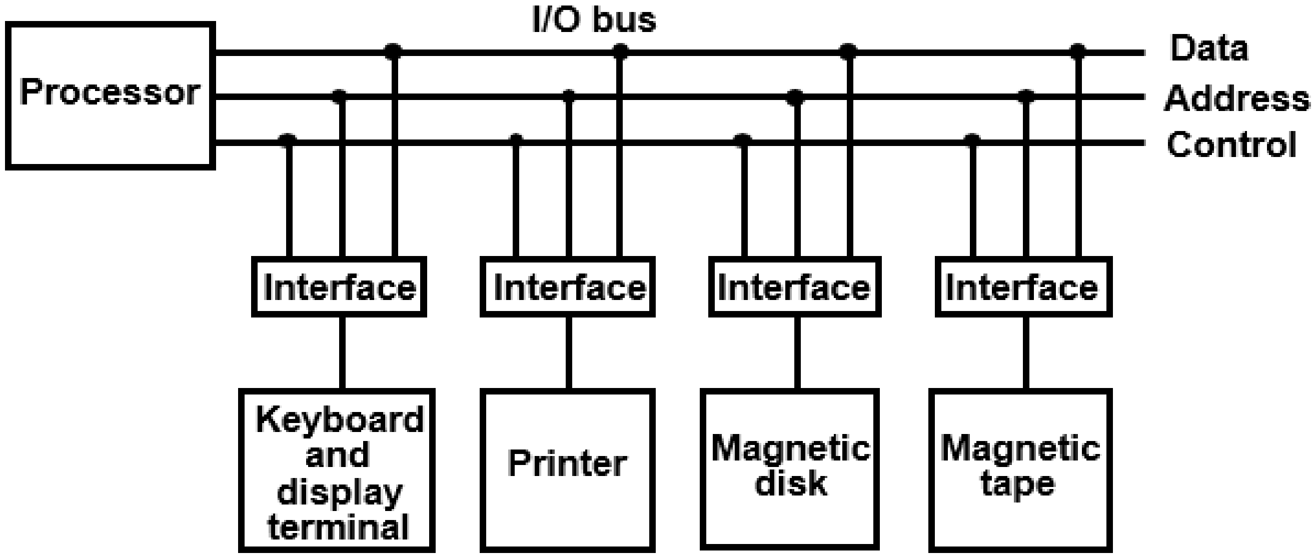
- Input-Output Interface
- Asynchronous Data Transfer
- Modes of Transfer
- Priority Interrupt
- Direct Memory Access
- Input-Output Processor
- CPU-IOP Communication

Input/Output Interface

- Provides a method for transferring information between internal storage (such as memory and CPU registers) and external I/O devices
- Resolves the *differences* between the computer and peripheral devices
  - Peripherals - Electromechanical Devices : Input or output devices are attached to the computer are called as a peripherals.
  - Most common peripheral devices are keyboard, display unit and printer.
  - Peripherals provide auxiliary storage for Magnetic tap and Magnetic disk.

I/O Bus And Interface Modules

- Input output interface provides a method for transferring information between internal storage and external I/O devices.
- The I/O bus consist of data lines, address lines and control lines.
- Each peripheral device has associated with its interface unit.



I/O Bus And Interface Modules

Working of interfaces:

- Decodes the device address (device code)
- Decodes the commands (operation)
- Provides signals for the peripheral controller
- Synchronizes the data flow and supervises the transfer rate between peripheral and CPU or Memory
- The I/O bus from the processor is attached to all peripheral interfaces.
- To communicated with a particular device, the processor places a device address on the address lines.
- Each interface attached to the I/O bus contain ad address decoder that monitor the address and activate path between bus line and sender device.

I/O Bus And Interface Modules

There are 4 types of command that an interface may receive/ interpret.

1. Control Command:
   - Use to activate the peripheral and to inform it what to do.
   - Rewind the magnetic tap, when it want to backspace the tape by one record OR To start moving magnetic tap in forward direction.
2. Status command:,
   - Used to test various status condition in interface and peripheral.
3. Data output command:
   - Cause the interface to respond by transferring data from bus into one of its register.
4. Data Input command:
   - Opposite of Data output command. In this case the interface receive an item of data from the peripheral and places it in its buffer register.

Isolated I/O Vs Memory Mapped I/O

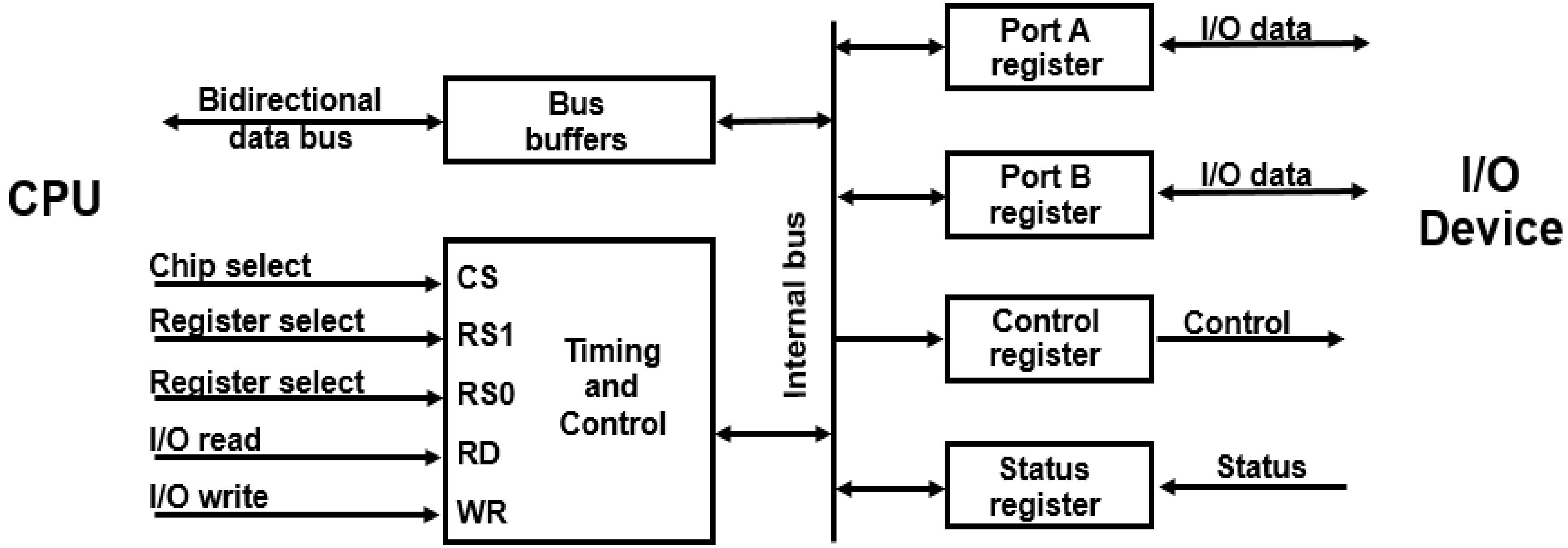
Isolated I/O

- Separate I/O read/write control lines in addition to memory read/write Control lines.
- Separate (isolated) memory and I/O address spaces
- Distinct input and output instructions

Memory mapped I/O

- A single set of read/write control lines
  (no distinction between memory and I/O transfer)
- Memory and I/O addresses share the common address space
  - reduces memory address range available
- No specific input or output instruction
  - The same memory reference instructions can be used for I/O transfers
- Considerable flexibility in handling I/O operations

Example I/O INTERFACE



CS	RS1	RS0	Register selected
0	x	x	None - data bus in high-impedance
1	0	0	Port A register
1	0	1	Port B register
1	1	0	Control register
1	1	1	Status register

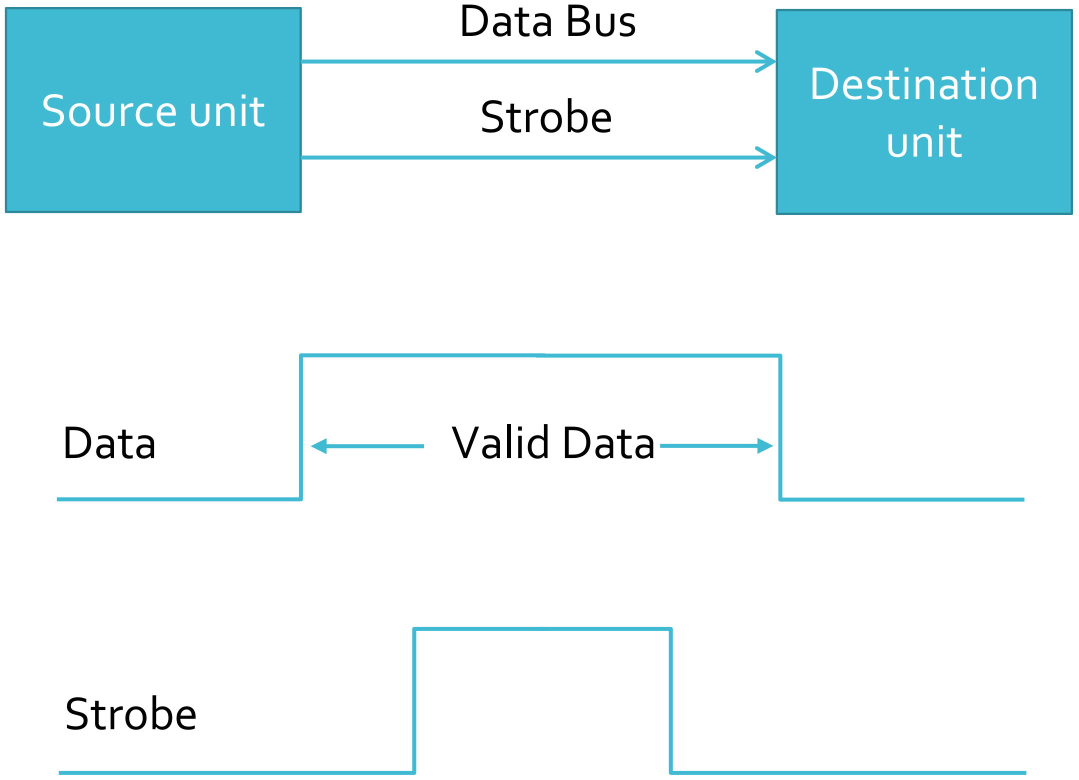
Example I/O INTERFACE

- It consist two data registers called ports, a control register, a status register, bus buffer and timing and control circuit.
- The interface communicate with the CPU through the data bus.
- The chip select and register select input determine the address which can be assign to the interface.
- The I/O read and write are two control lines that specify an input or output respectively.
- The four register communicate directly with the I/O device attached to the interface.
- If the interface connected to a printer only output data is active and if it service as a reader data, input data is active.
- A magnetic disk unit transfer data in both directions but not at the same time, so the interface can use bidirectional bus.
- The control register receive control information from the CPU.

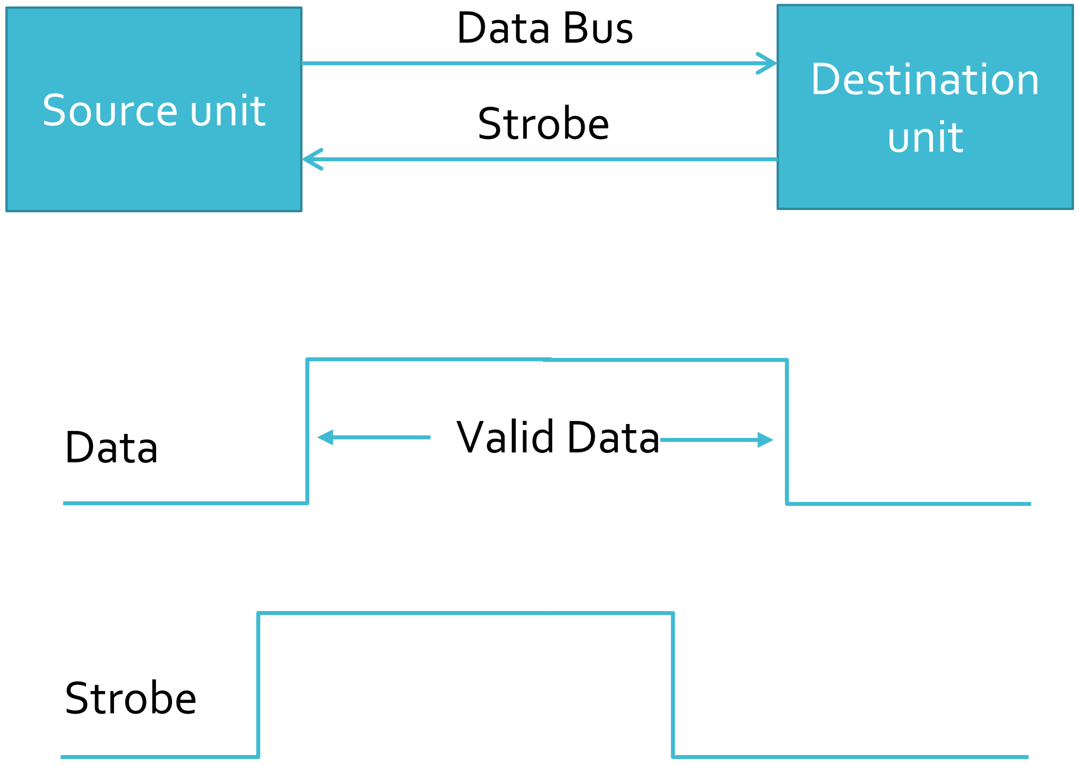
Asynchronous Data Transfer

- Asynchronous data transfer between two independent units requires that control signals be transmitted between the communicating units to indicate the time at which data is being transmitted.
- Two ways of achieving
  1. Strobe
  2. Handshaking

- 1.1 Source initiated Strobe

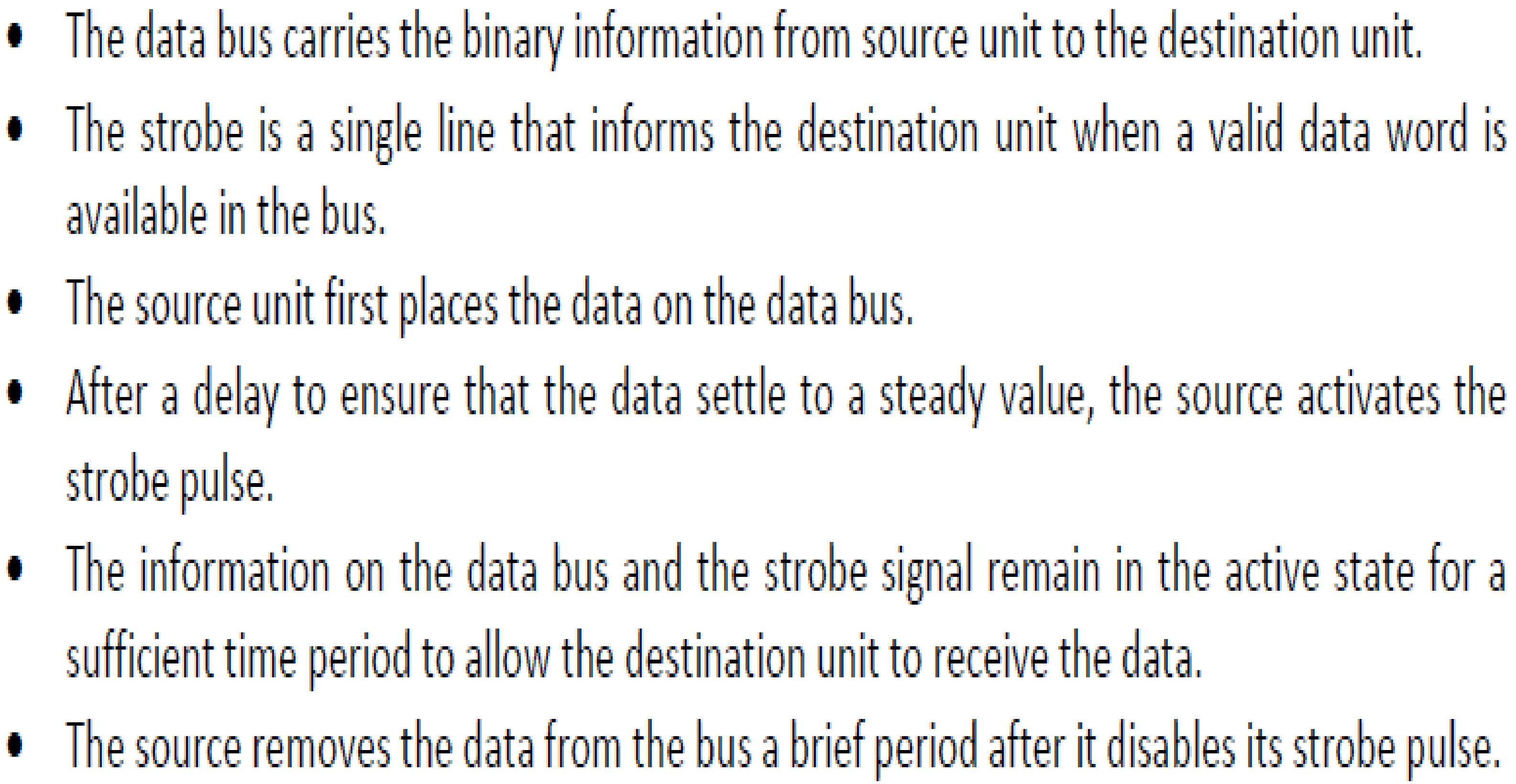


1.2 Destination initiated Strobe



Source initiated strobe

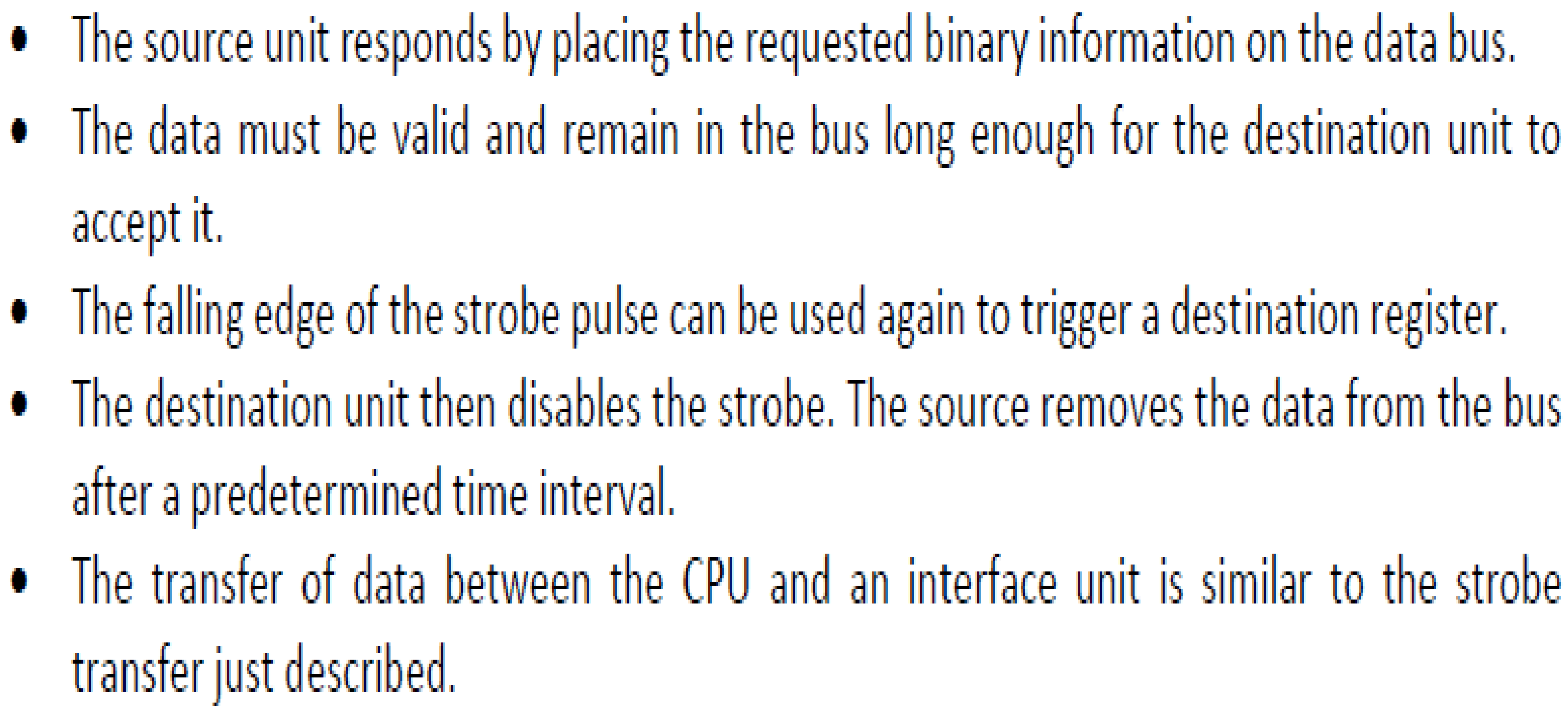
Source initiated strobe:



- The data bus carries the binary information from source unit to the destination unit.
- The strobe is a single line that informs the destination unit when a valid data word is available in the bus.
- The source unit first places the data on the data bus.
- After a delay to ensure that the data settle to a steady value, the source activates the strobe pulse.
- The information on the data bus and the strobe signal remain in the active state for a sufficient time period to allow the destination unit to receive the data.
- The source removes the data from the bus a brief period after it disables its strobe pulse.

Destination initiated strobe

Destination initiated strobe:

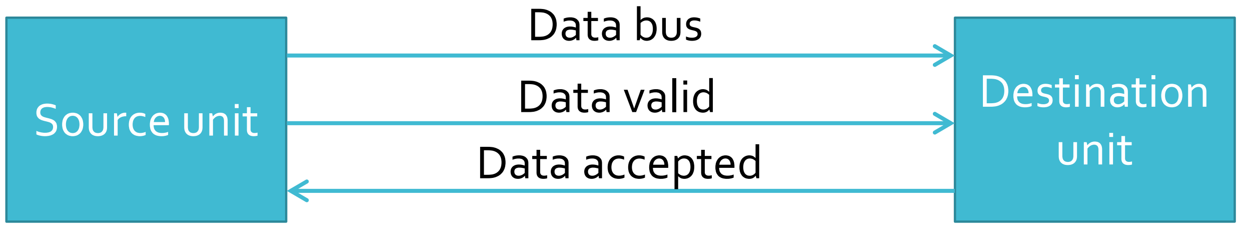


- The source unit responds by placing the requested binary information on the data bus.
- The data must be valid and remain in the bus long enough for the destination unit to accept it.
- The falling edge of the strobe pulse can be used again to trigger a destination register.
- The destination unit then disables the strobe. The source removes the data from the bus after a predetermined time interval.
- The transfer of data between the CPU and an interface unit is similar to the strobe transfer just described.

Handshaking

- Disadvantage of Strobe Method:
- Source-Initiated: The source unit that initiates the transfer has no way of knowing whether the destination unit has actually received data.
- Destination-Initiated: The destination unit that initiates the transfer no way of knowing whether the source has actually placed the data on the bus.
- To solve this problem, the HANDSHAKE method introduces a one more signal i.e acknowledge signal to provide a Reply to the unit that initiates the transfer

Source initiated Handshake initiated



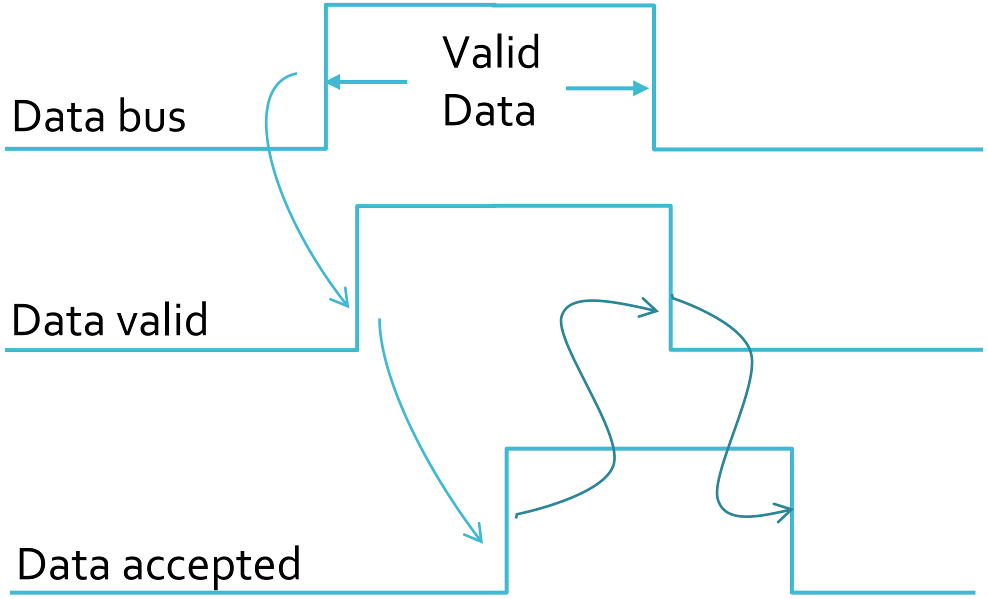
Source unit

Data bus

Data valid

Data accepted

Destination unit

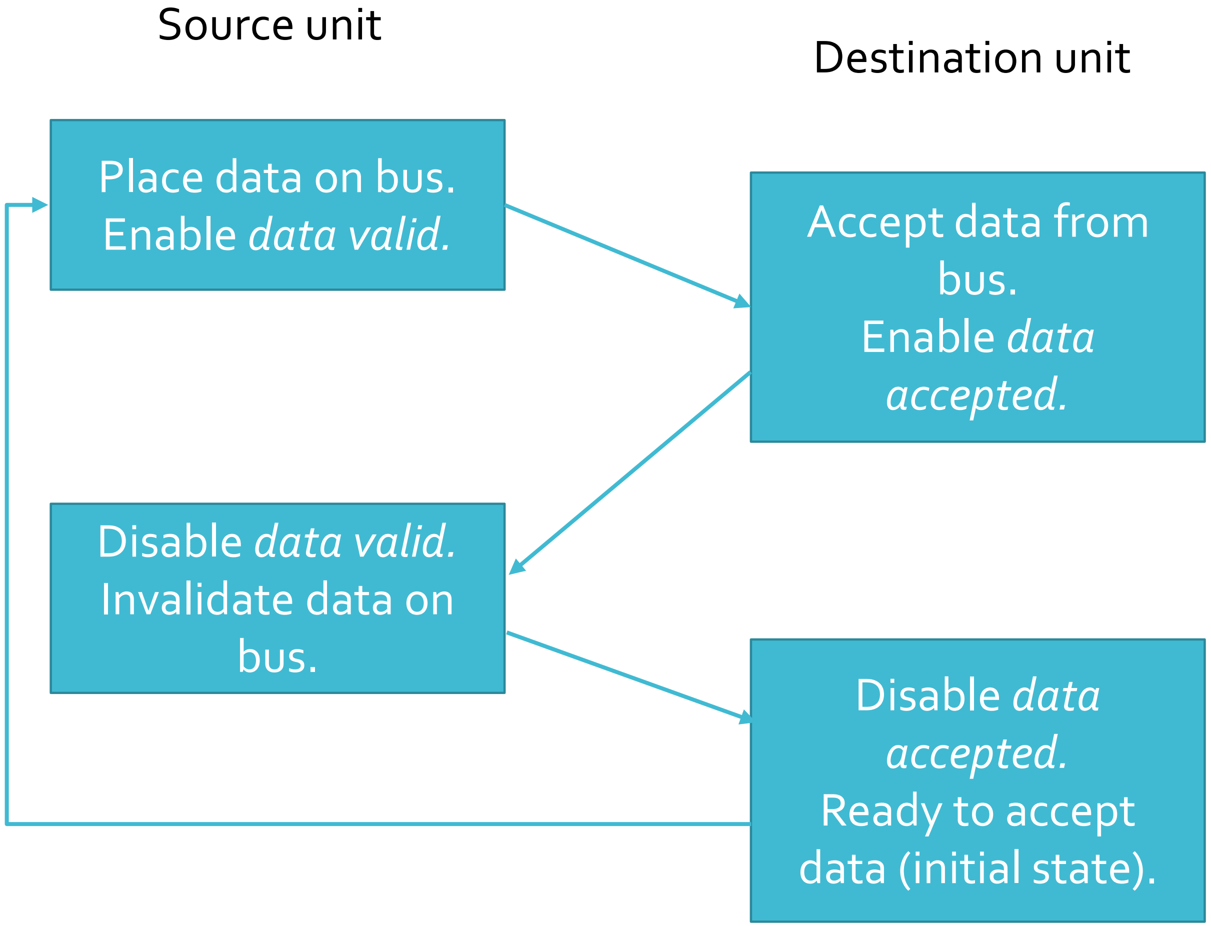


Data bus

Valid Data

Data valid

Data accepted



Source unit

Destination unit

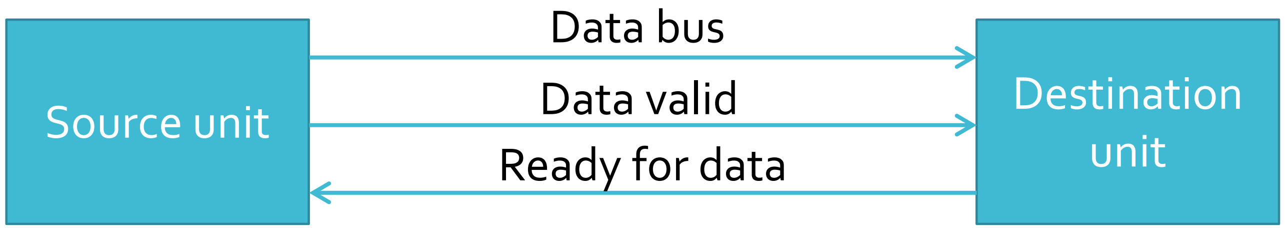
Place data on bus.
Enable *data valid.*

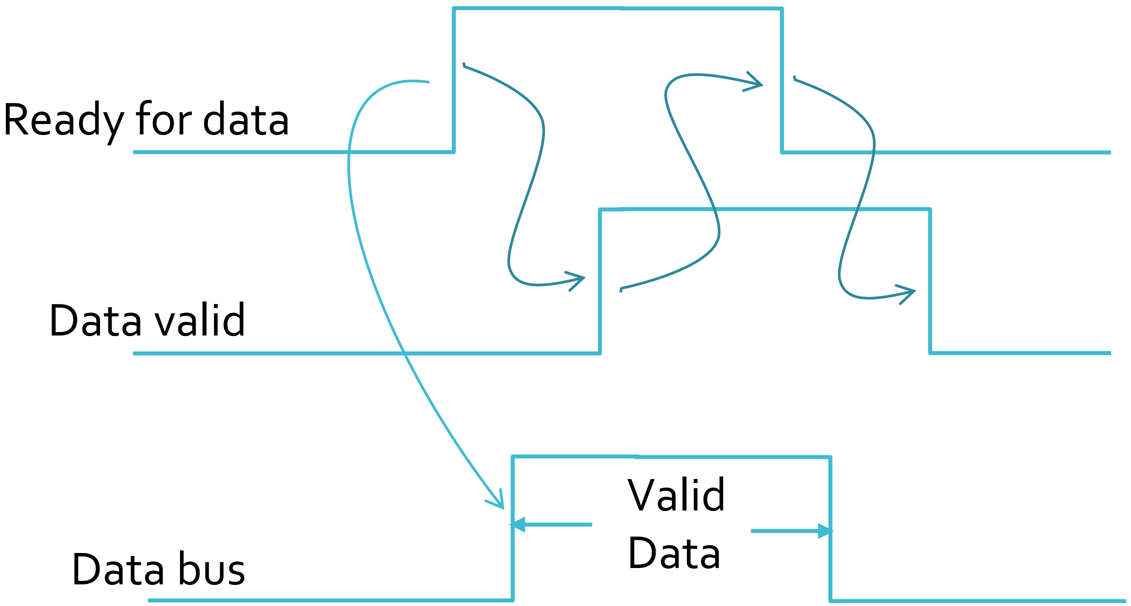
Accept data from bus.
Enable *data accepted.*

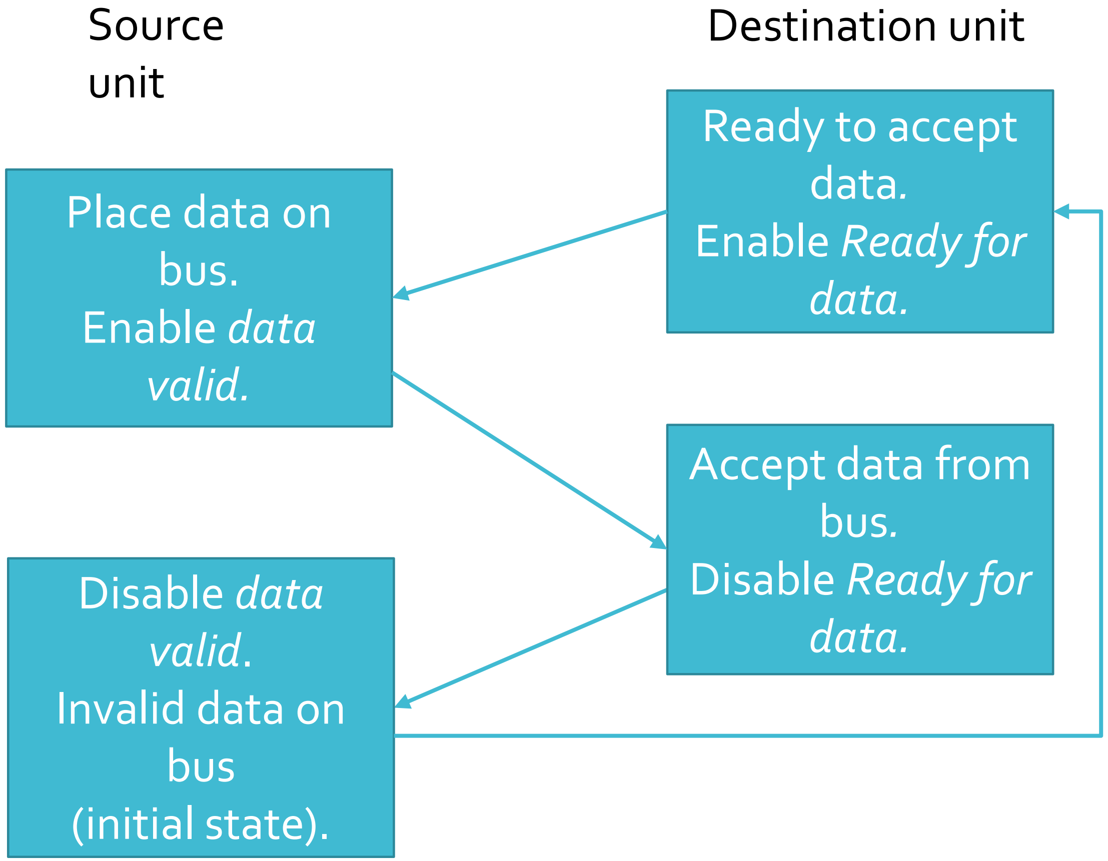
Disable *data valid.*
Invalidate data on bus.

Disable *data accepted.*
Ready to accept data (initial state).

Destination initiated Hand shake







Ready to accept data. Enable *Ready for data.*

Place data on bus. Enable *data valid.*

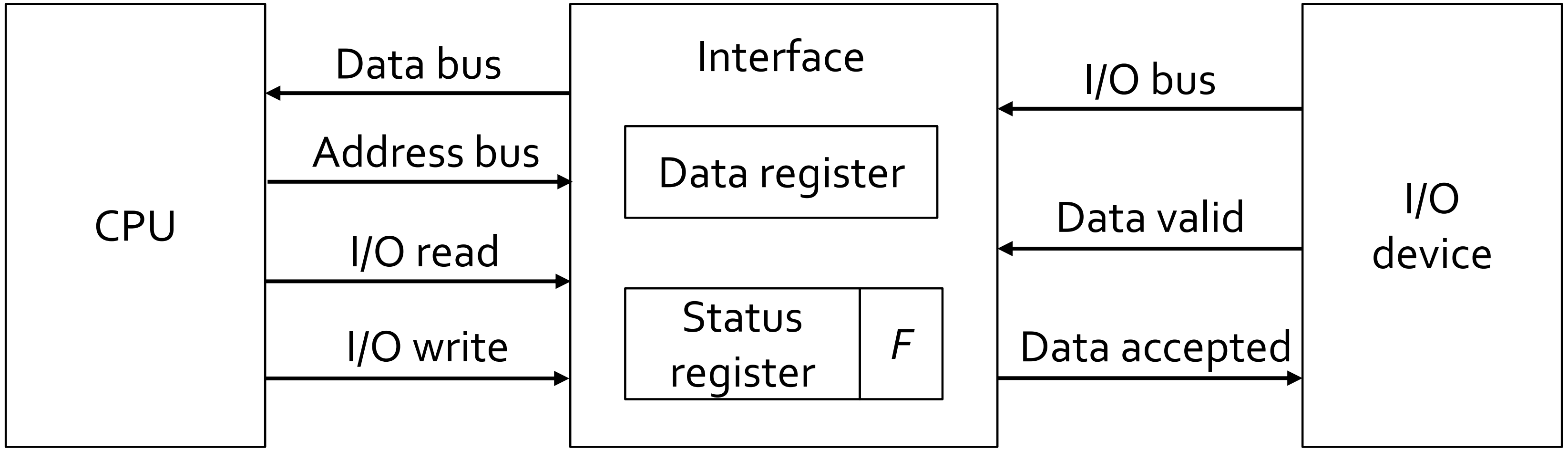
Accept data from bus. Disable *Ready for data.*

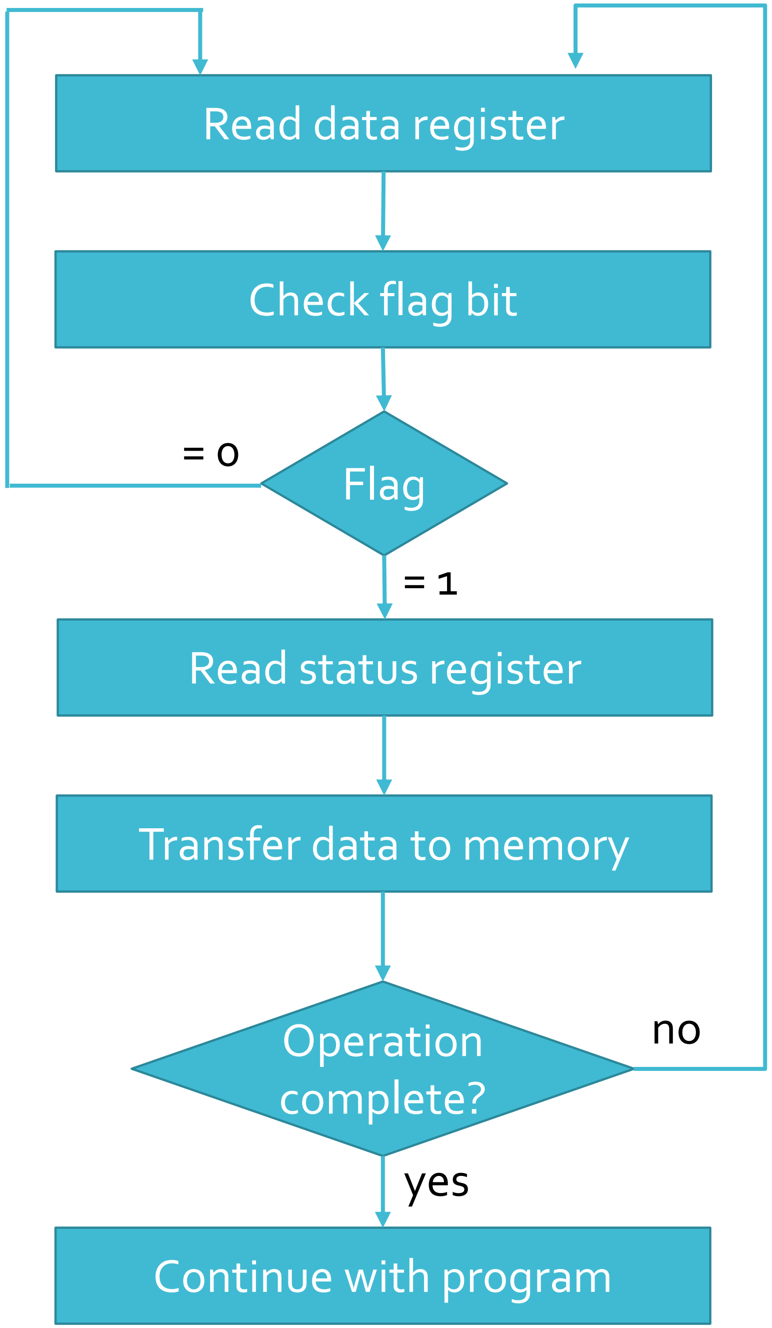
Disable *data valid*. Invalid data on bus (initial state).

Modes of Transfer

- Data transfer between the central computer and I/O devices may be handled in a variety of modes.
- Some modes use the CPU as an intermediate path; others transfer the data directly to and from the memory unit.
- Data transfer to and from peripherals may be handled in one of three possible modes:
  1. Programmed I/O
  2. Interrupt-initiated I/O
  3. Direct Memory Access (DMA)

Programmed I/O





Interrupt initiated I/O

- An alternative to the CPU constantly monitoring the flag is to let the interface inform the computer when it is ready to transfer data.
- While the CPU is running a program, it does not check the flag.
- However, when the flag is set, the computer is momentarily interrupted from proceeding with current program and is informed of the fact that the flag has been set.
- The CPU deviates from what it is doing to take care of the input or output transfer.
- After the transfer is completed, the computer returns to the previous program to continue what it was doing before the interrupt.

PRIORITY INTERRUPT

- Determines which interrupt is to be served first when two or more requests are made simultaneously.
- Also determines which interrupts are permitted to interrupt the computer while another is being serviced.
- When two devices interrupt the computer at the same time , the computer services the device with the higher priority first.
- Priority can be established using software and hardware.
- Two method to check priority for interrupt: **Polling** and **Daisy Chaining**
- A polling procedure is used to identify the highest priority source by software. Whereas daisy chaining method used by hardware.

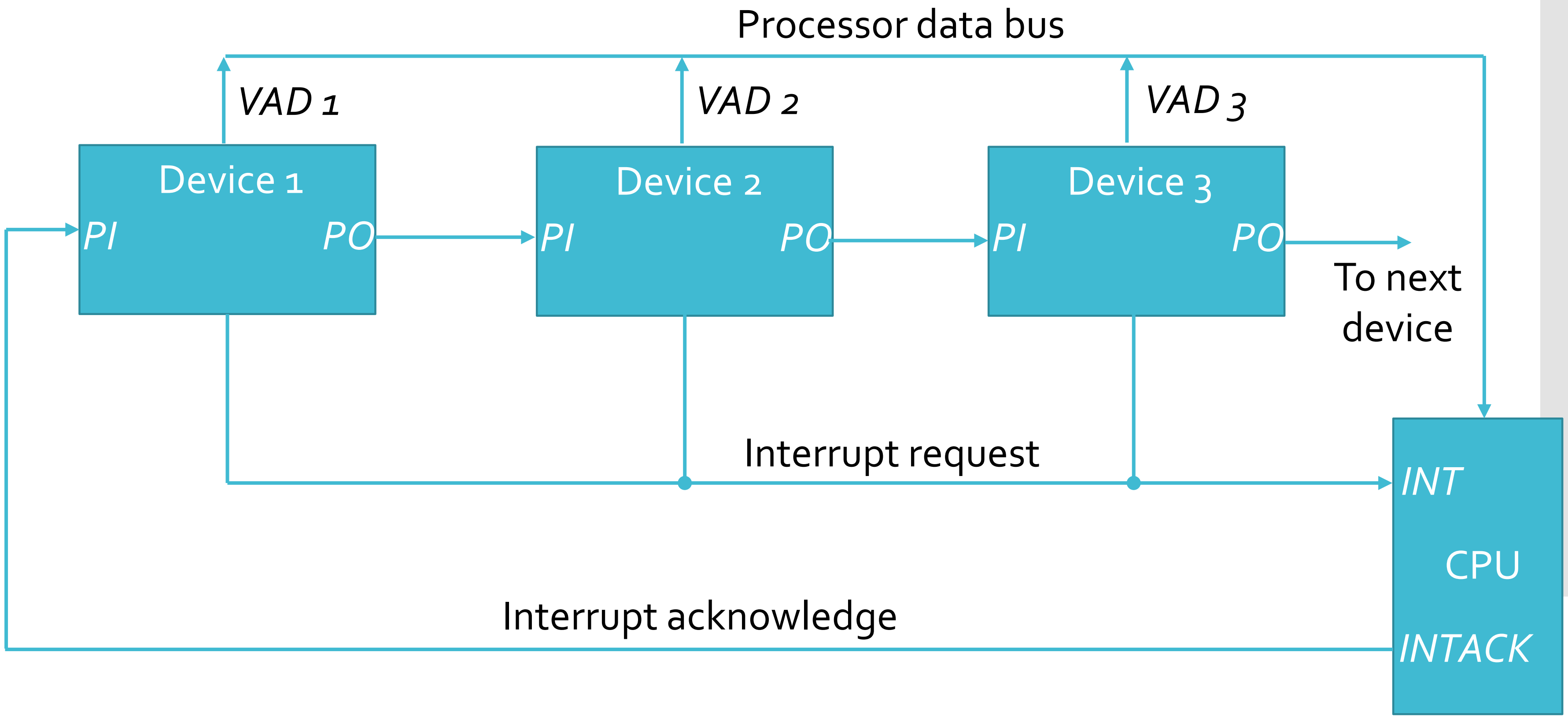
Polling

Polling:

- Polling is a protocol that notifies CPU that a device needs its attention. Unlike in interrupt, where device tells CPU that it needs CPU processing, in polling CPU keeps asking the I/O device whether it needs CPU processing.
- In this method there is one common branch address for all interrupts.
- The program that take 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.

Priority Interrupt (Daisy-Chaining Technique)

- Determines which interrupt is to be served first when two or more requests are made simultaneously
- Also determines which interrupts are permitted to interrupt the computer while another is being serviced.
- Higher priority interrupts can make requests while servicing a lower priority interrupt.



Direct Memory Access(DMA)

Definition: The data transfer technique in which peripherals manage the memory buses for direct interaction with main memory without involving the CPU is called DMA.

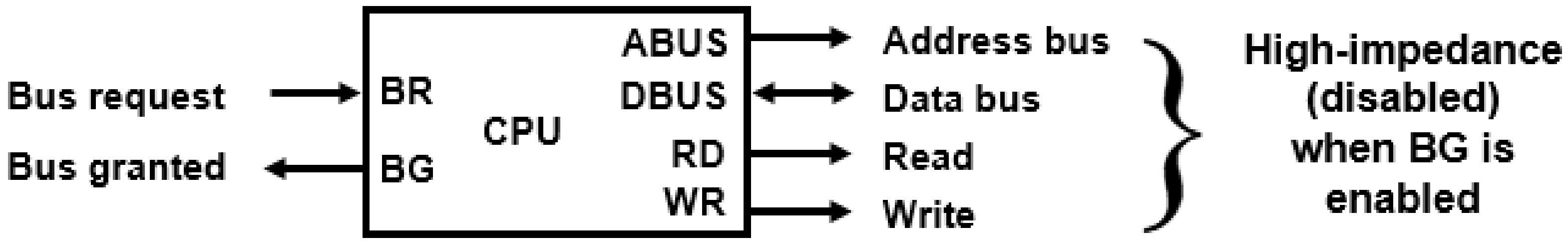
Direct Memory Access (DMA)

- Large blocks of data transferred at a high speed to or from high speed devices, disks, tapes, etc.
- DMA controller: removing CPU from the path and letting the peripheral device manage the memory buses directly to transfer the content.
- CPU initializes the DMA controller by sending a memory address and the number of words to be transferred.
- Actual transfer of data is done directly between the device and memory through DMA controller

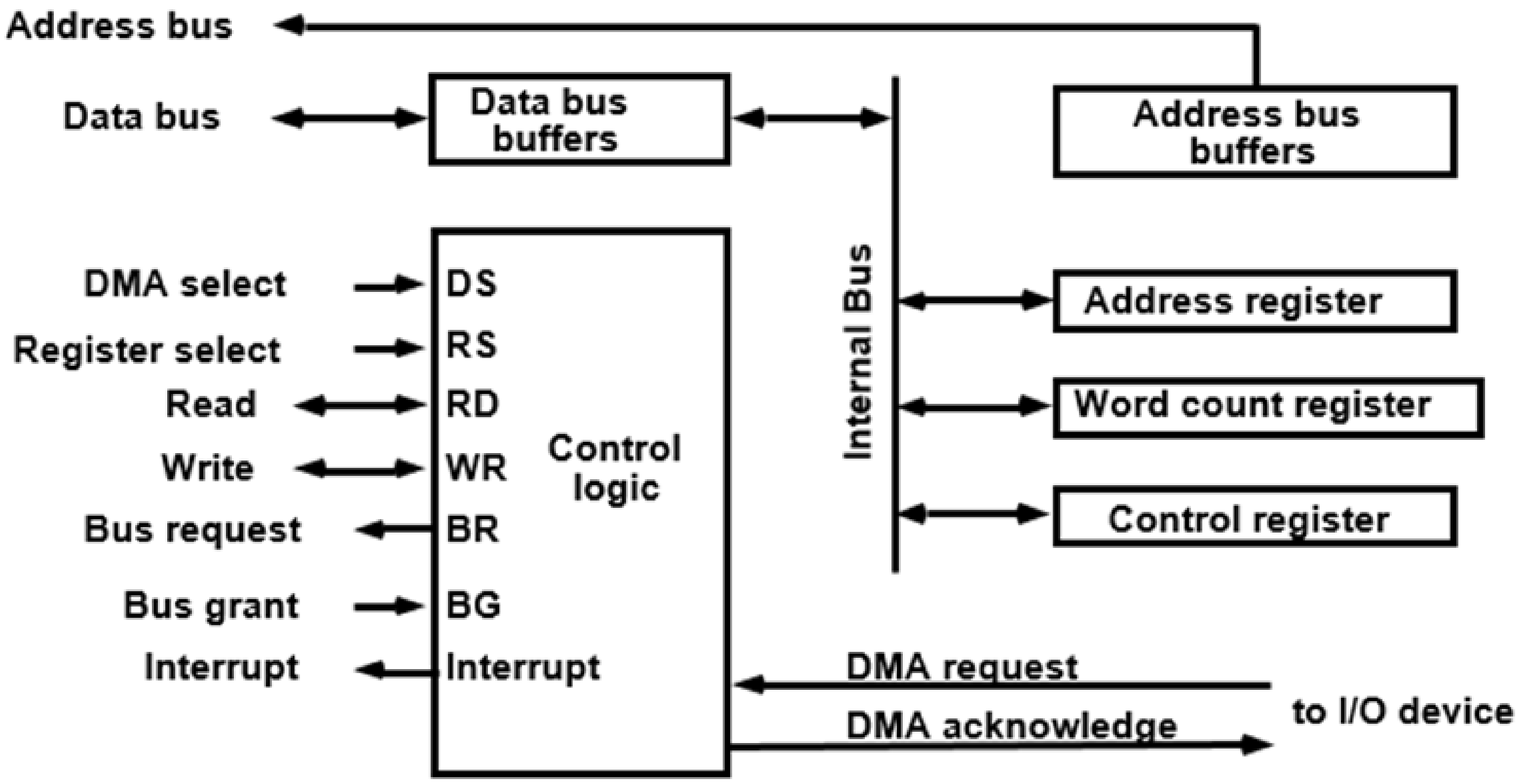
  -> Freeing CPU for other tasks
- DMA controller need the usual circuits of an interface to communicate with the CPU and I/O device.

Direct Memory Access(DMA)

CPU bus signals for DMA transfer



Block diagram of DMA controller



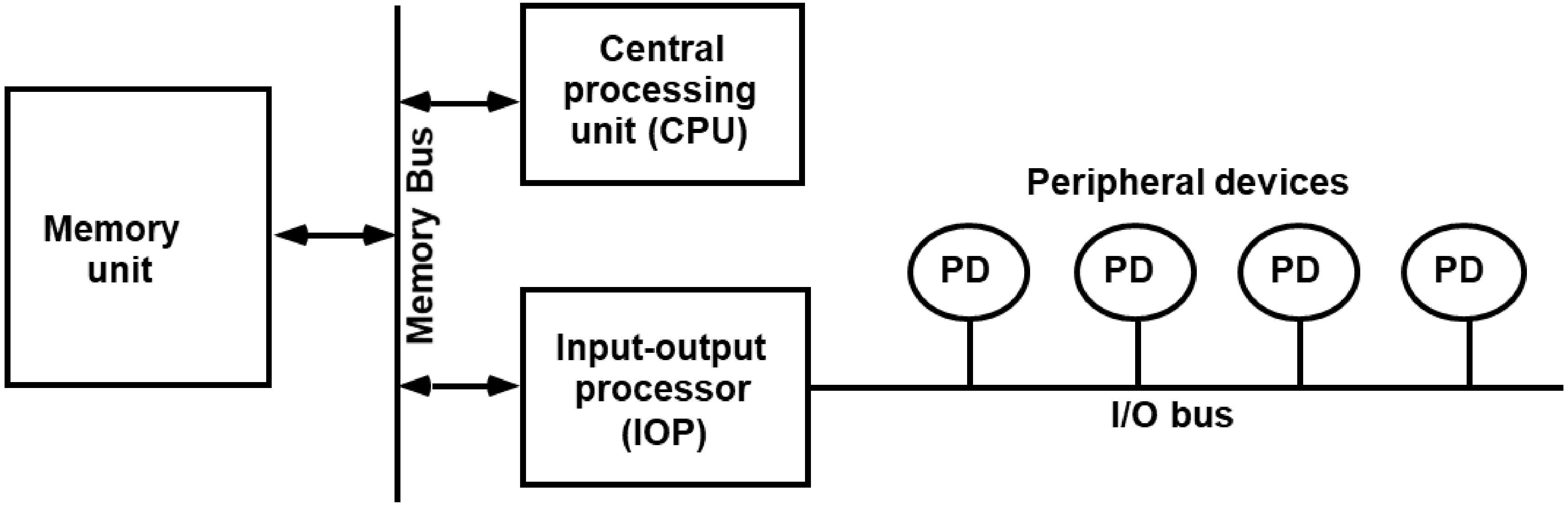
Direct Memory Access(DMA)

- It is bypassing the CPU to speedup memory operations.
- The process managed by a chip known as a DMA controller.
- During DMA transfer the CPU is idle and has no control of the memory buses.
- The control unit communicates the CPU via data bus control lines.
- The DMA controls the system bus using BR(bus request) and BG(bus grant)signals.
- DMA operates read and write operations via RD and WR signals.
- DMA send request and acknowledge to I/O devices via DMA request and DMA acknowledge signals.

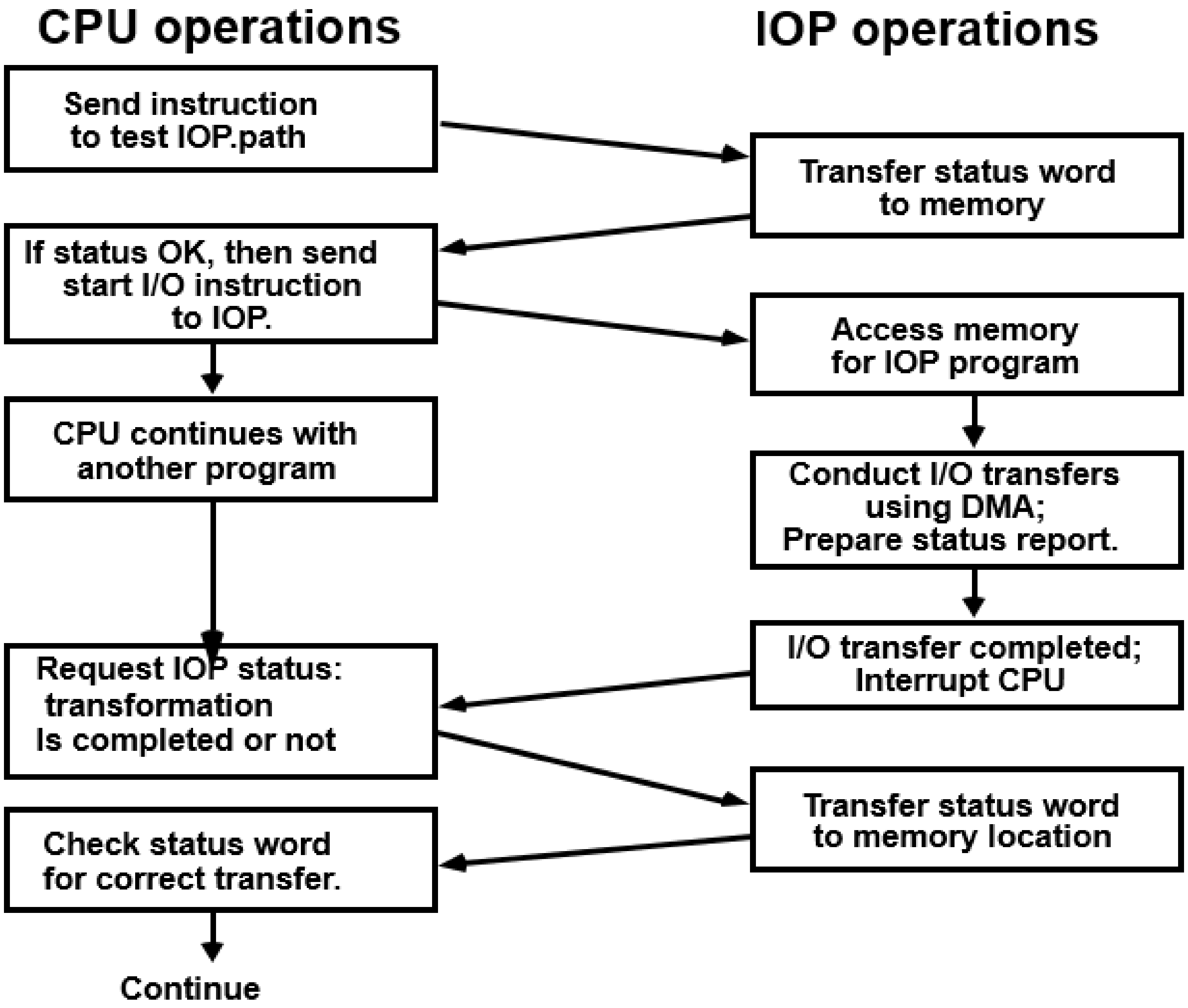
Direct Memory Access (DMA)

- The register selection process in DMA are done by the CPU through the address bus by enabling DS(DMA select) and RS(Register select) inputs.
- The Address register contains an address to specify the desired location in memory. It is incremented after each word that is transferred to the memory.
- The word count register holds the number of words to be transferred. It is decremented by one after each word transfer and internally tested for zero.
- The control register specifies the mode of transfer.

Input/Output Processor (IOP)



CPU-IOP Communication


CPU operations
IOP operations
Send instruction to test IOP.path
Transfer status word to memory
If status OK, then send start I/O instruction to IOP.
Access memory for IOP program
CPU continues with another program
Conduct I/O transfers using DMA; Prepare status report.
I/O transfer completed; Interrupt CPU
Request IOP status: transformation Is completed or not
Transfer status word to memory location
Check status word for correct transfer.
Continue