Input-Output Organization

Peripheral Devices

- Devices that are under the direct control of the computer are said to be connected on-line. These devices are designed to read information into or out of the memory unit upon command from the CPU and are considered to be part of the total computer system. Input or output devices attached to the computer are also called peripherals.
- Among the most common peripherals are keyboards, display units, and printers. Peripherals that provide auxiliary storage for the system are magnetic disks and tapes. Peripherals are electromechanical and electromagnetic devices of some complexity.

ASCII CHARACTERS

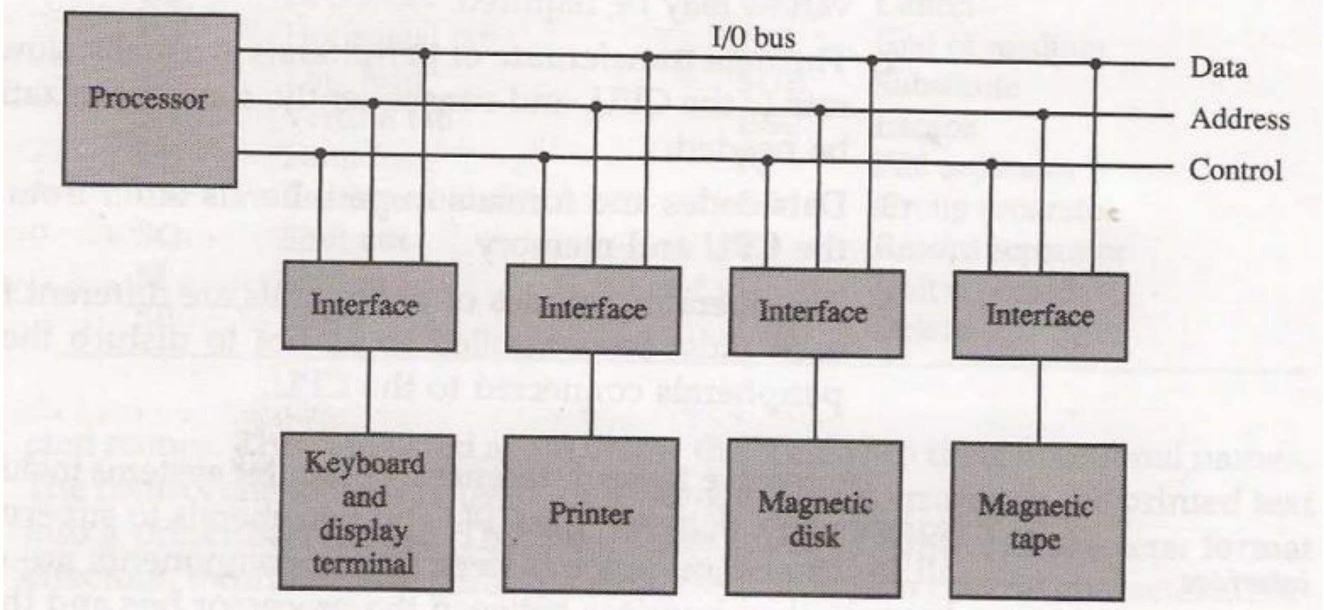
- Input and output devices that communicate with people and the computer are usually involved in the transfer of alphanumeric information to and from the device and the computer. The standard binary code for the alphanumeric characters is ASCII (American Standard Code for Information Interchange). It uses seven bits to code 128 characters

Input-Output Interface

- Input-output interface provides a method for transferring information between internal storage and external I/0 devices. Peripherals connected to a computer need special communication links for interfacing them with the central processing unit. The purpose of the communication link is to resolve the differences that exist between the central computer and each peripheral. The major differences are:
- 1. Peripherals are electromechanical and electromagnetic devices and their manner of operation is different from the operation of the CPU and memory, which are electronic devices. Therefore, a conversion of signal values may be required.
- 2. The data transfer rate of peripherals is usually slower than the transfer rate of the CPU, and consequently, a synchronization mechanism may be needed.
- 3. Data codes and formats in peripherals differ from the word format in the CPU and memory.
- 4. The operating modes of peripherals are different from each other and each must be controlled so as not to disturb the operation of other peripherals connected to the CPU.

- To resolve these differences, computer systems include special hardware components between the CPU and peripherals to supervise and synchronize all input and output transfers. These components are called interface units because they interface between the processor bus and the peripheral device.
- In addition, each device may have its own controller that supervises the operations of the particular mechanism in the peripheral.

Figure 11-1 Connection of I/O bus to input-output devices.



- The I/O bus from the processor is attached to all peripheral interfaces. To communicate with a particular device, the processor places a device address on the address lines. Each interface attached to the I/O bus contains an address decoder that monitors the address lines. When the interface detects its own address, it activates the path between the bus lines and the device that it controls. All peripherals whose address does not correspond to the address in the bus are disabled by their interface.
- **I/O command**: At the same time that the address is made available In the address lines, the processor provides a function code in the control lines. The interface selected responds to the function code and proceeds to execute it. The function code is referred to as an I/O command and is in essence an instruction that is executed in the interface and its attached peripheral unit.
- **Control command** is issued to activate the peripheral and to inform it what to do.
- **Status command** is used to test various status conditions in the interface and the peripheral.
- **output data:** A data output command causes the interface to respond by transferring data from the bus into one of its registers.
- **input data:** The data input command is the opposite of the data output. In this case the interface receives an item of data from the peripheral and places it in its buffer register

I/O versus Memory Bus

- There are three ways that computer buses can be used to communicate with memory and I/O:
- 1. Use two separate buses, one for memory and the other for I/O.
- 2. Use one common bus for both memory and I/O but have separate control lines for each.
- 3. Use one common bus for memory and I/O with common control lines.

Isolated I/0

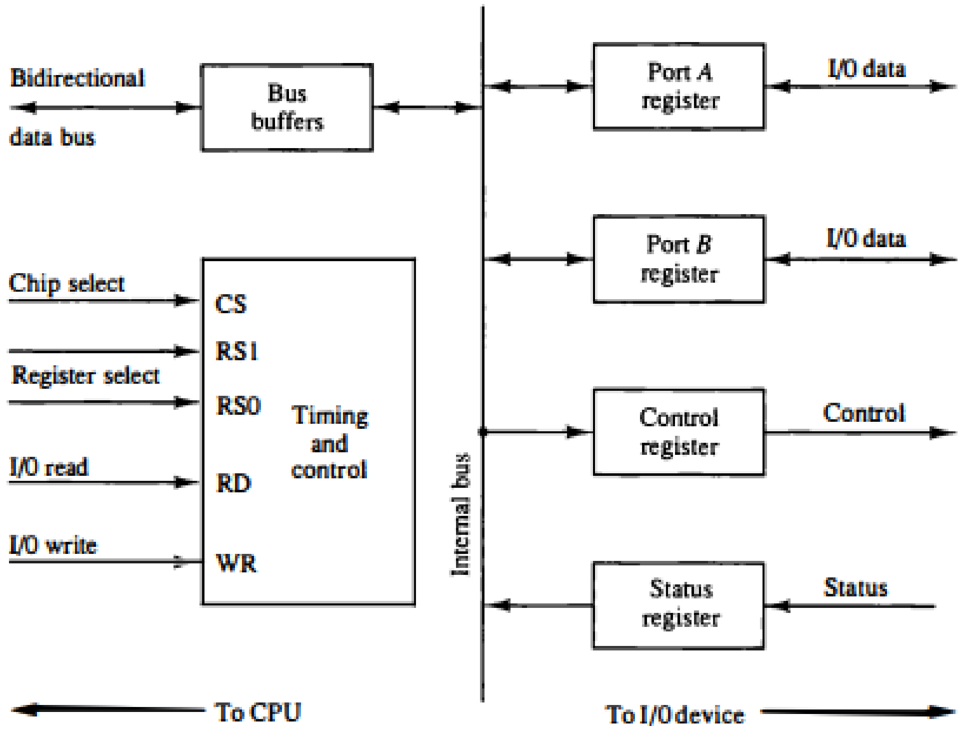
- In the isolated I/0 configuration, the CPU has distinct input and output instructions, and each of these instructions is associated with the address of an interface register. When the CPU fetches and decodes the operation code of an input or output instruction, it places the address associated with the instruction into the common address lines. At the same time, it enables the I/0 read (for input) or I/0 write (for output) control line. This informs the external components that are attached to the common bus that the address in the address lines is for an interface register and not for a memory word.
- On the other hand, when the CPU is fetching an instruction or an operand from memory, it places the memory address on the address lines and enables the memory read or memory write control line. This informs the external components that the address is for a memory word and not for an I/0 interface

Memory mapped I/O

- The isolated I/O method isolates memory and I/O addresses so that memory address values are not affected by interface address assignment since each has its own address space. The other alternative is to use the same address space for both memory and I/O. This is the case in computers that employ only one set of read and write signals and do not distinguish between memory and I/O addresses. This configuration is referred to as memory-mapped I/O. The computer treats an interface register as being part of the memory system. The assigned addresses for interface registers cannot be used for memory words, which reduces the memory address range available.
- In a memory-mapped l/0 organization there are no specific input or output instructions. The CPU can manipulate l/0 data residing in interface registers with the same instructions that are used to manipulate memory words. Each interface is organized as a set of registers that respond to read and write requests in the normal address space.

UART

- The terminal has a keyboard and a printer. Every time a key is depressed, the terminal sends 11 bits serially along a wire. To print a character in the printer, an 11-bit message must be received along another wire. The terminal interface consists of a transmitter and a receiver. The transmitter accepts an 8-bit character from the computer and proceeds to send a serial 11-bit message into the printer line. The receiver accepts a serial 11-bit message from the keyboard line and forwards the 8-bit character code into the computer. Integrated circuits are available which are specifically designed to provide the interface between computer and similar interactive terminals. Such a circuit is called an asynchronous communication interface or a universal asynchronous receivertransmitter (UART).



CS	RS1	RS0	Register selected
0	×	×	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

Figure 11-2 Example of I/O interface unit.

- The block diagram of an asynchronous communication interface is shown in Fig. It functions as both a transmitter and a receiver. The interface is initialized for a particular mode of transfer by means of a control byte that is loaded into its control register.
- The transmitter register accepts a data byte from the CPU through the data bus. This byte is transferred to a shift register for serial transmission. The receiver portion receives serial information into another shift register, and when a complete data byte is accumulated, it is transferred to the receiver register.
- The CPU can select the receiver register to read the byte through the data bus. The bits in the status register are used for input and output flags and for recording certain errors that may occur during the transmission. The CPU can read the status register to check the status of the flag bits and to determine if any errors have occurred.
- The chip select and the read and write control lines communicate with the CPU. The chip select (CS) input is used to select the interface through the address bus. The register select (RS) is associated with the read (RD) and write (WR) controls. Two registers are write-only and two are read-only. The register selected is a function of the RS value and the RD and WR status.

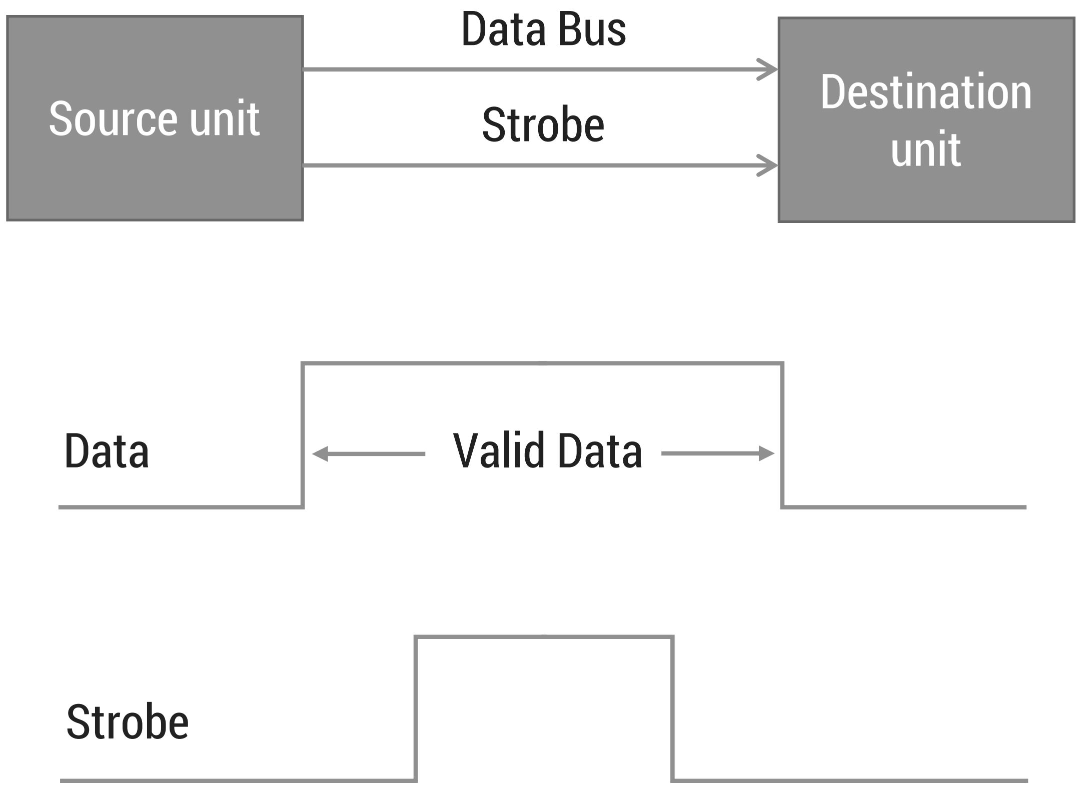
Asynchronous Data Transfer

- 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.

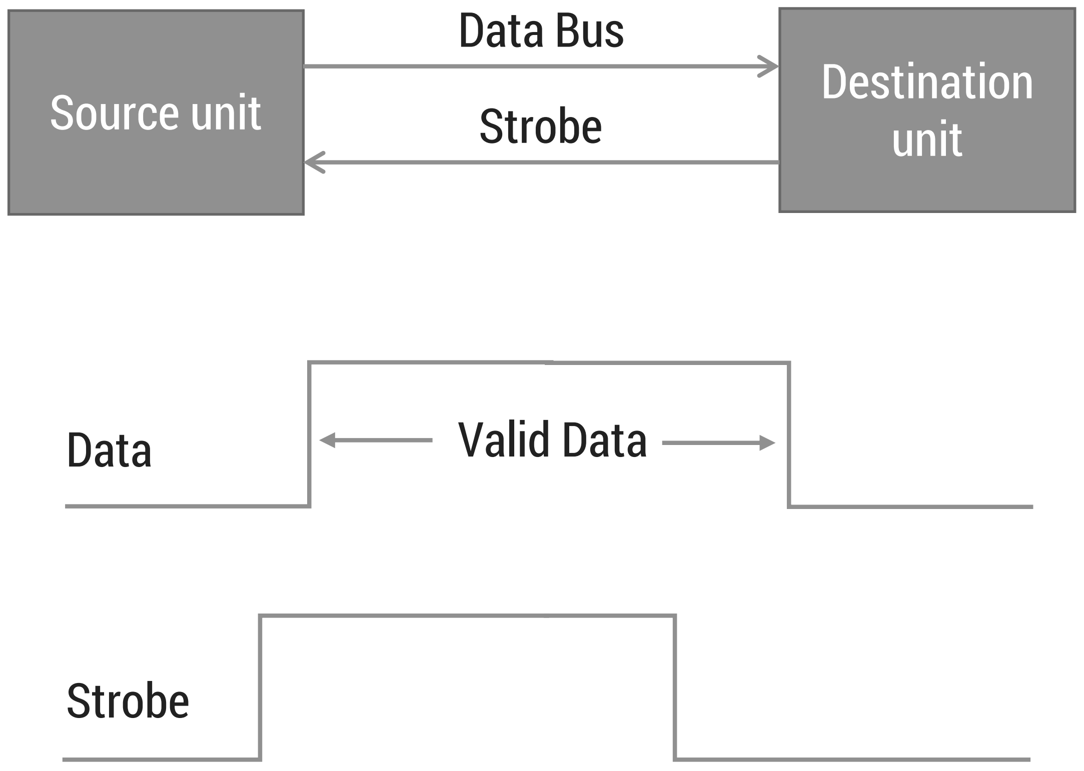
- Two ways of achieving
  1. Strobe
  2. Handshaking

Strobe Method

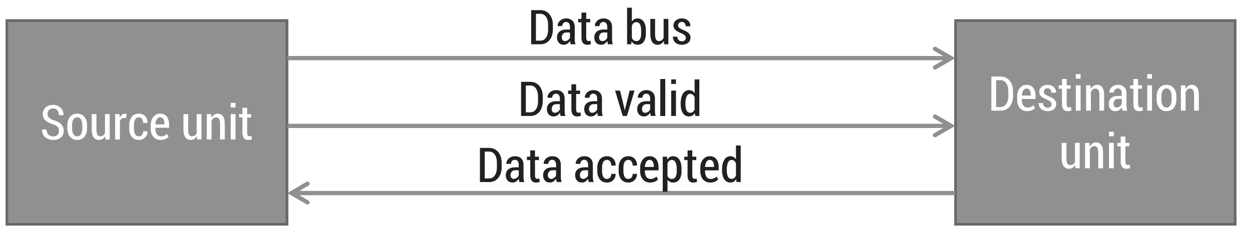
1.1 Source initiated Strobe

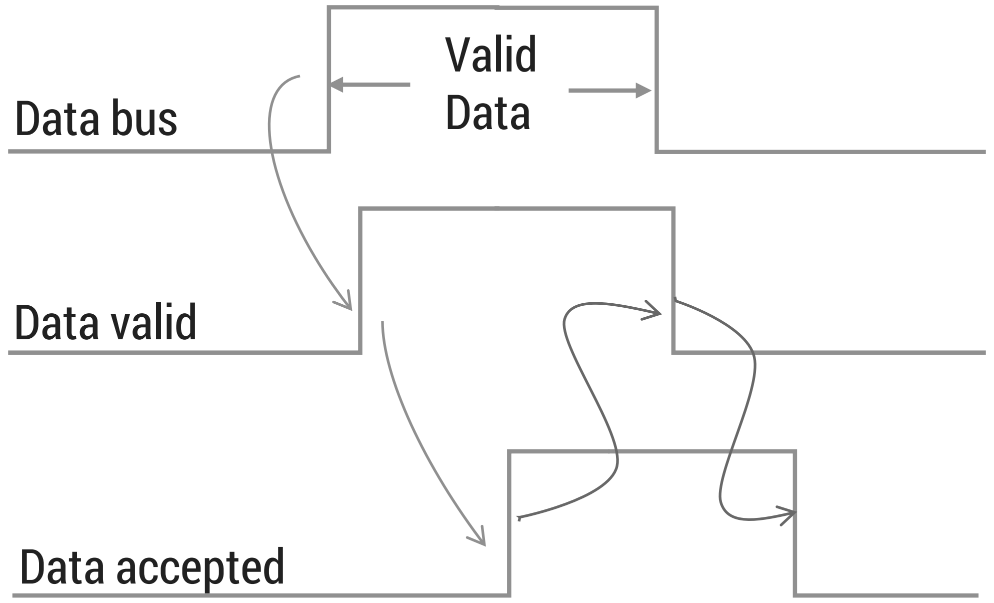


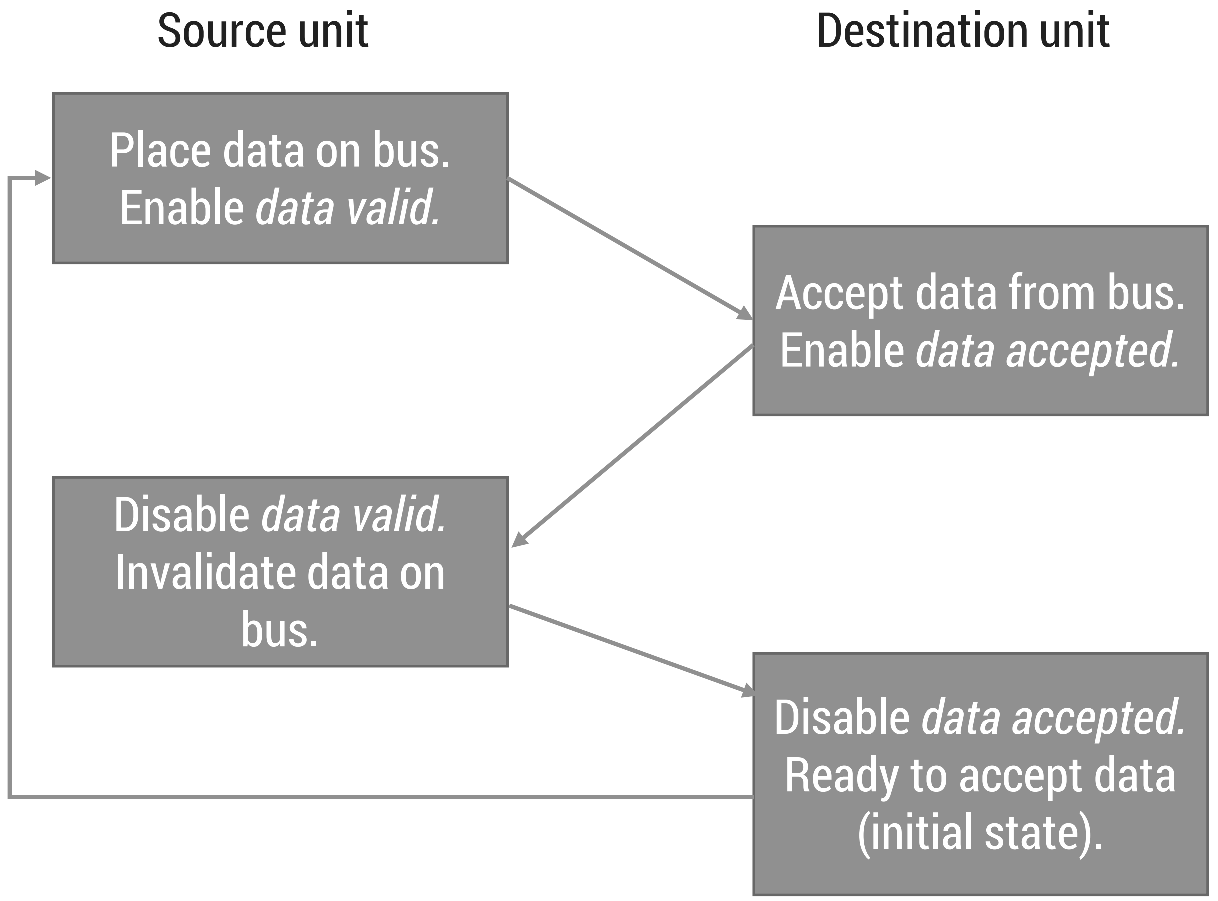
1.2 Destination initiated Strobe



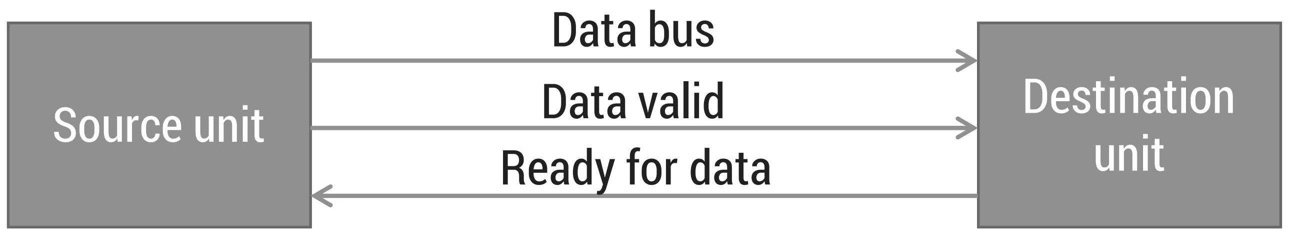
2.1 Source initiated Handshake

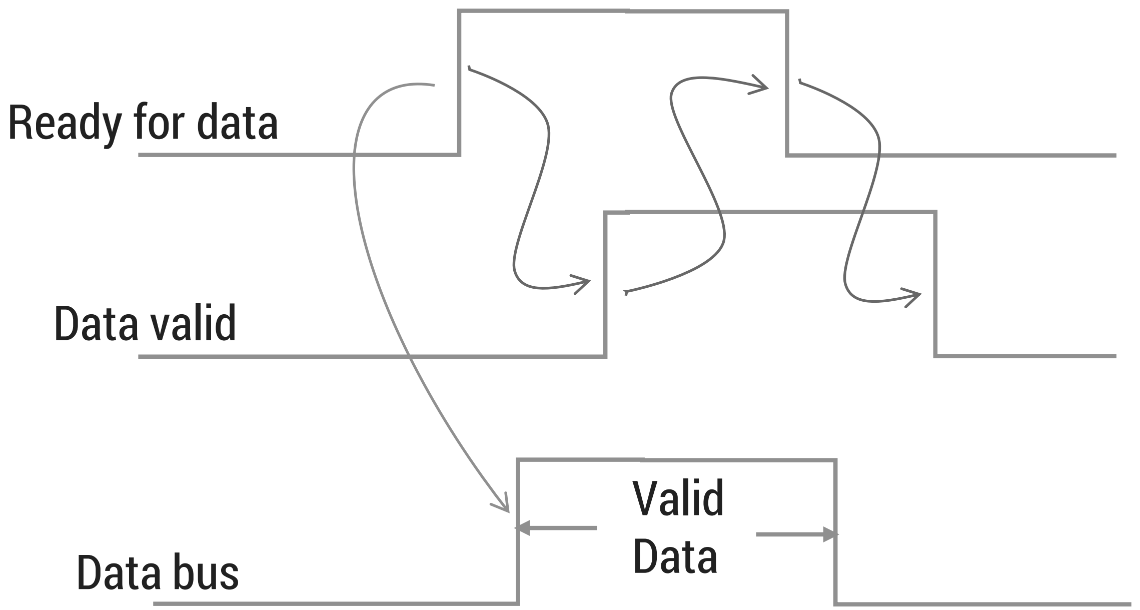


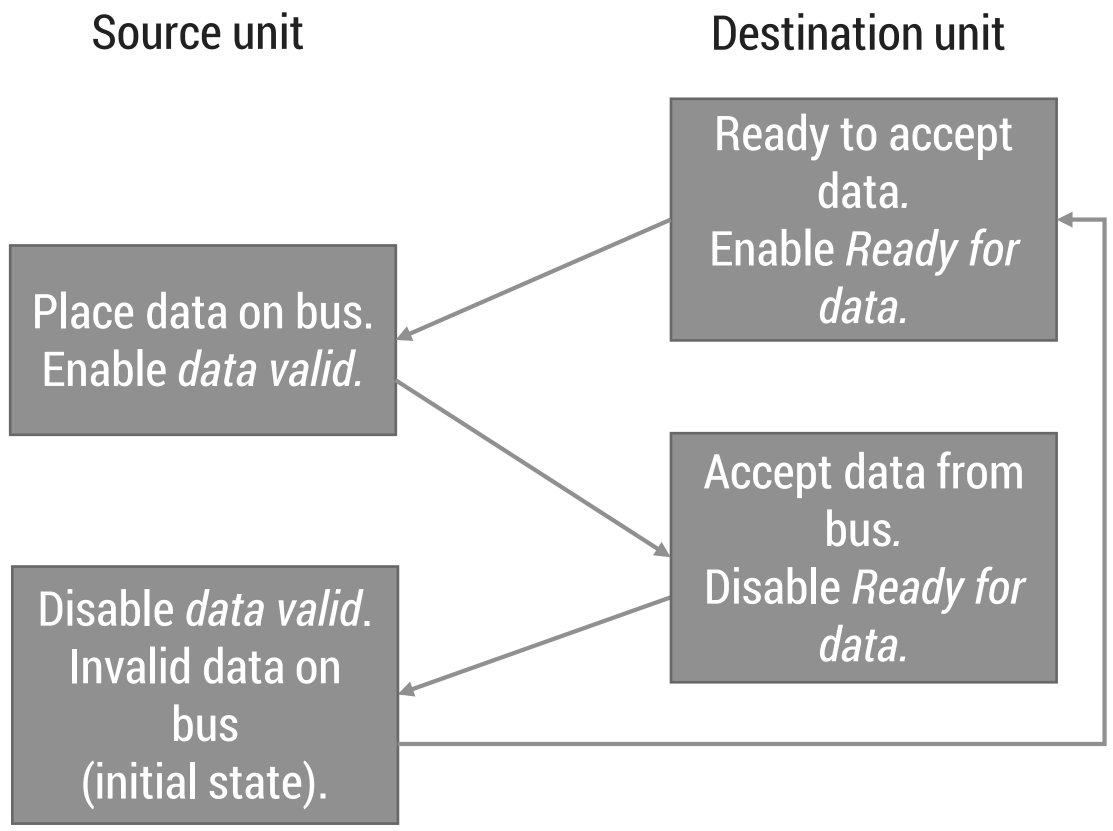



Source unit
Destination unit
Place data on bus. Enable *data valid.*
Accept data from bus. Enable *data accepted.*
Disable *data valid.* Invalidate data on bus.
Disable *data accepted.* Ready to accept data (initial state).

2.2 Destination initiated Handshake





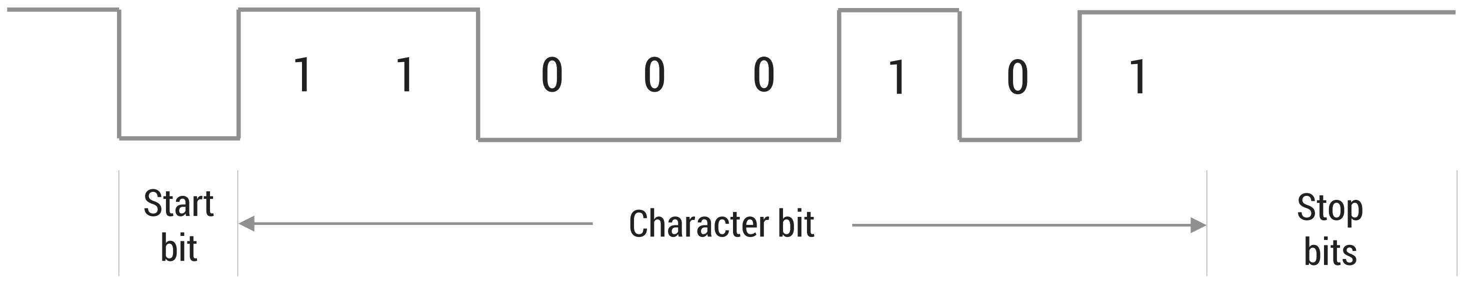


Synchronous vs Asynchronus serial transfer

- In long distant serial transmission, each unit is driven by a separate clock of the same frequency. Synchronization signals are transmitted periodically between the two units to keep their clocks in step with each other.
- In asynchronous transmission, binary information is sent only when it is available and the line remains idle when there is no information to be transmitted. This is in contrast to synchronous transmission, where bits must be transmitted continuously to keep the clock frequency in both units synchronized with each other.

Asynchronous Serial Transfer

- Rules for transmission
  1. When a character is not being sent, the line is kept in the 1-state.
  2. The initiation of a character transmission is detected from the start bit, which is always 0.
  3. The character bits always follow the start bit.
  4. After the last bit of the character is transmitted, a stop bit is detected when the line returns to the 1-state for at least one bit time.

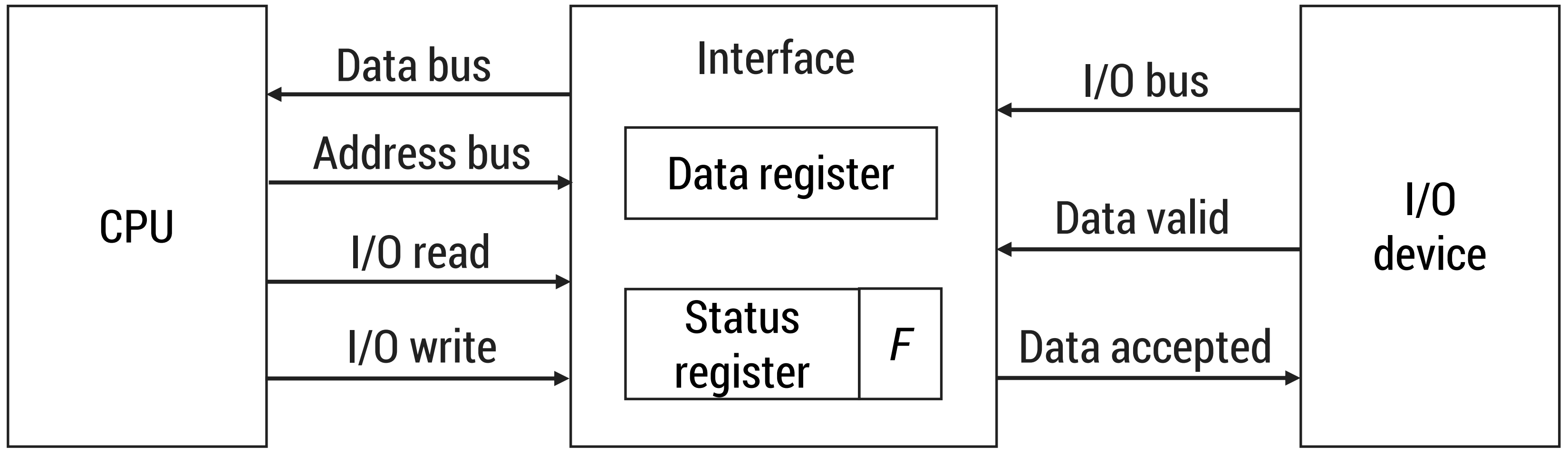


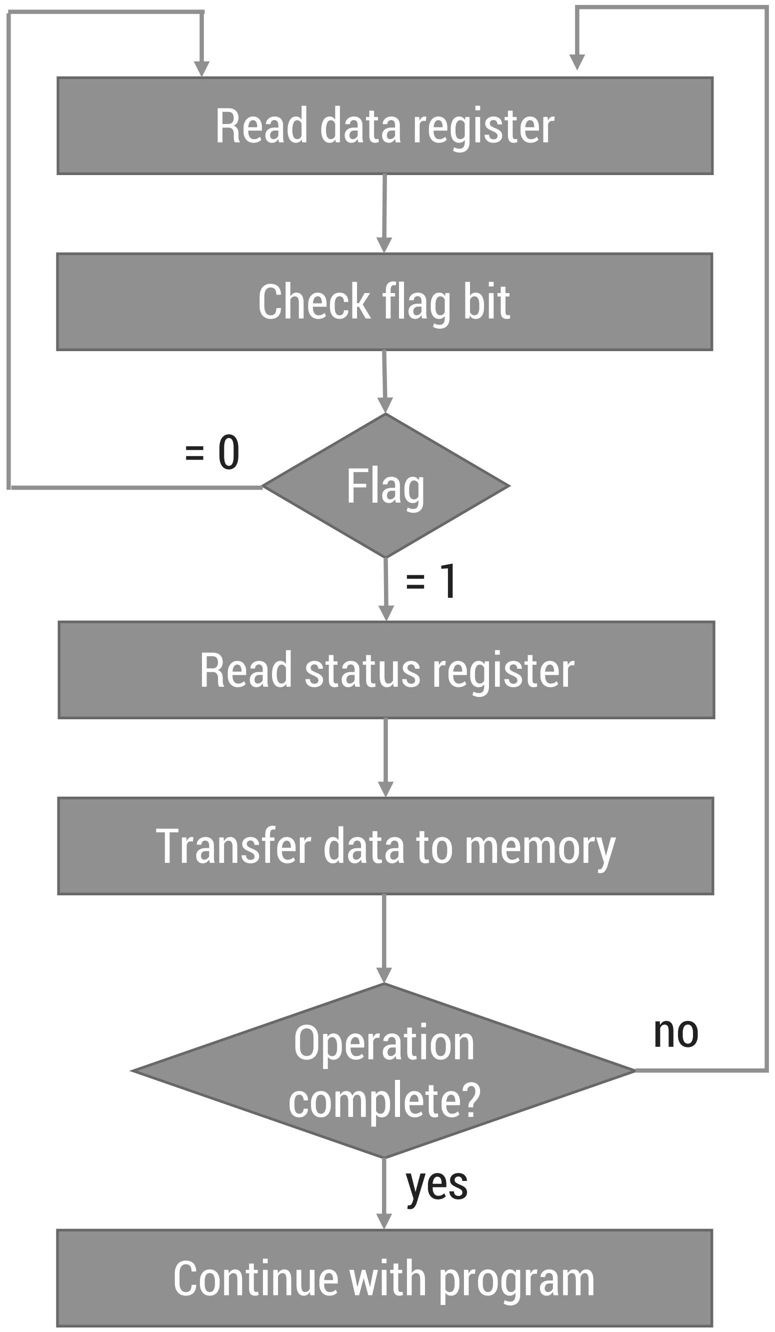
baud rate: The baud rate is defined as the rate at which serial information is transmitted and is equivalent to the data transfer in bits per second.

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



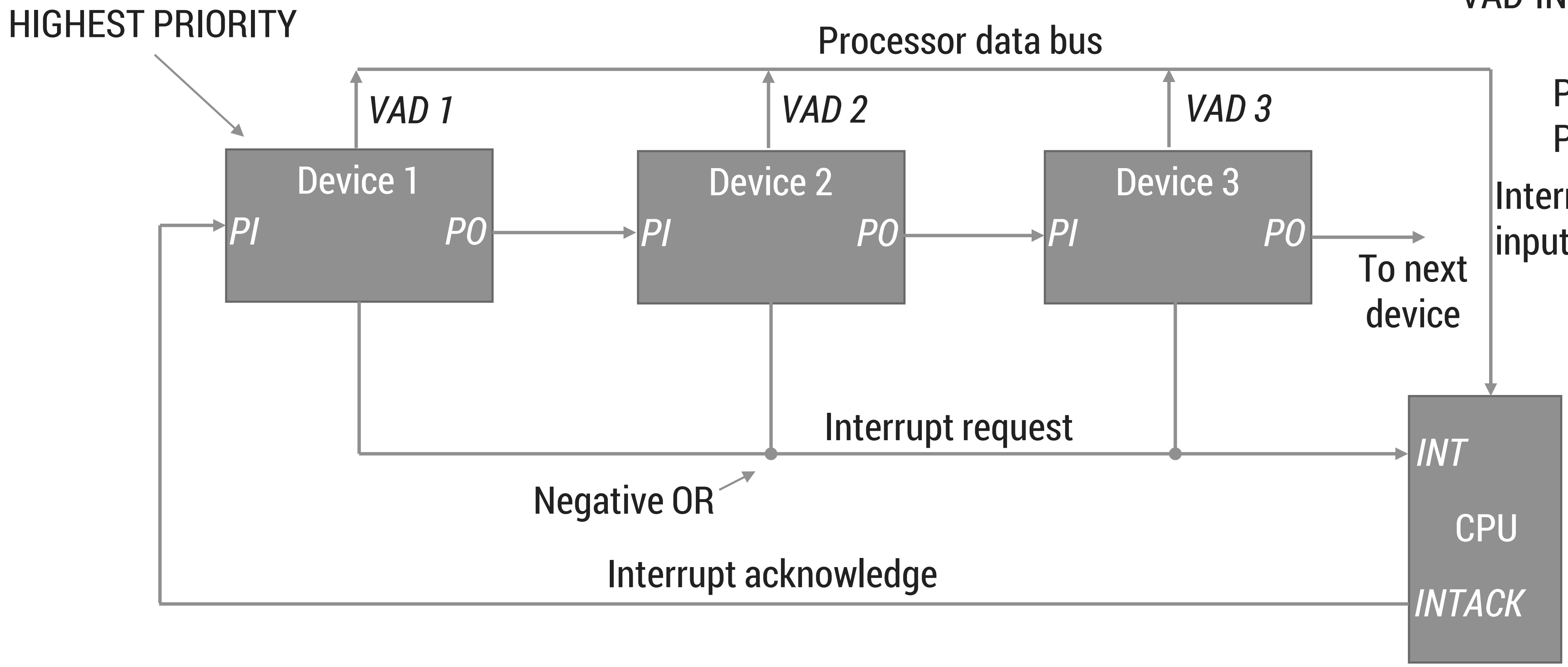


Interrupt-initiated I/O

- 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.
- The way that the processor chooses the branch address of the service routine varies from one unit to another. In principle, there are two methods for accomplishing this. One is called vectored interrupt and the other, nonvectored interrupt.
- In a non vectored interrupt, the branch address is assigned to a fixed location in memory.
- In a vectored interrupt, the source that interrupts supplies the branch information to the computer. This information is called the interrupt vector. In some computers the interrupt vector is the first address of the VO service routine.
- In other computers the interrupt vector is an address that points to a location in memory where the beginning address of the VO service routine is stored.

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.



VAD-INTRUPT VECTOR ADDRESS

PI-PRIORITY IN
PO-PRIORITY OUT

Interrupt request line low-> interrupt
input to CPU

Parallel Priority Interrupt

- 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.
- In addition to the interrupt register, the circuit may include a mask register whose purpose is to control the status of each interrupt request. The mask register can be programmed to disable lower-priority interrupts while a higher-priority device is being serviced. It can also provide a facility that allows a high-priority device to interrupt the CPU while a lower-priority device is being serviced.
- interrupt register whose individual bits are set by external conditions and cleared by program instructions. the encoder sets an interrupt status flip-flop IST when an interrupt that is not masked occurs.
- The interrupt enable flip-flop IEN can be set or cleared by the program to provide an overall control over the interrupt system.
- The outputs of IST ANDed with IEN provide a common interrupt signal for the CPU. The interrupt acknowledge IN'TACK signal from the CPU enables the bus buffers in the output register and a vector address VAD is placed into the data bus

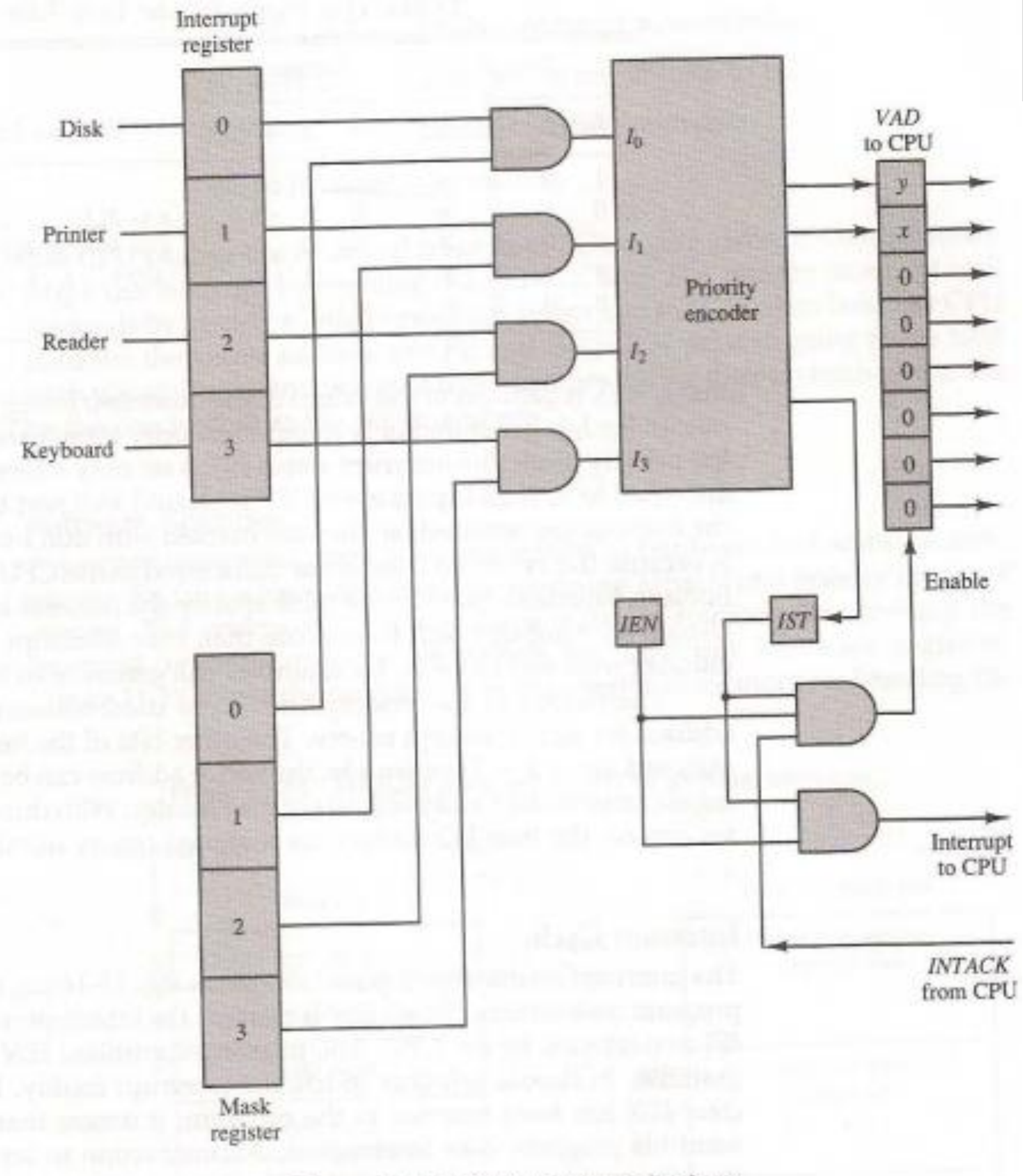
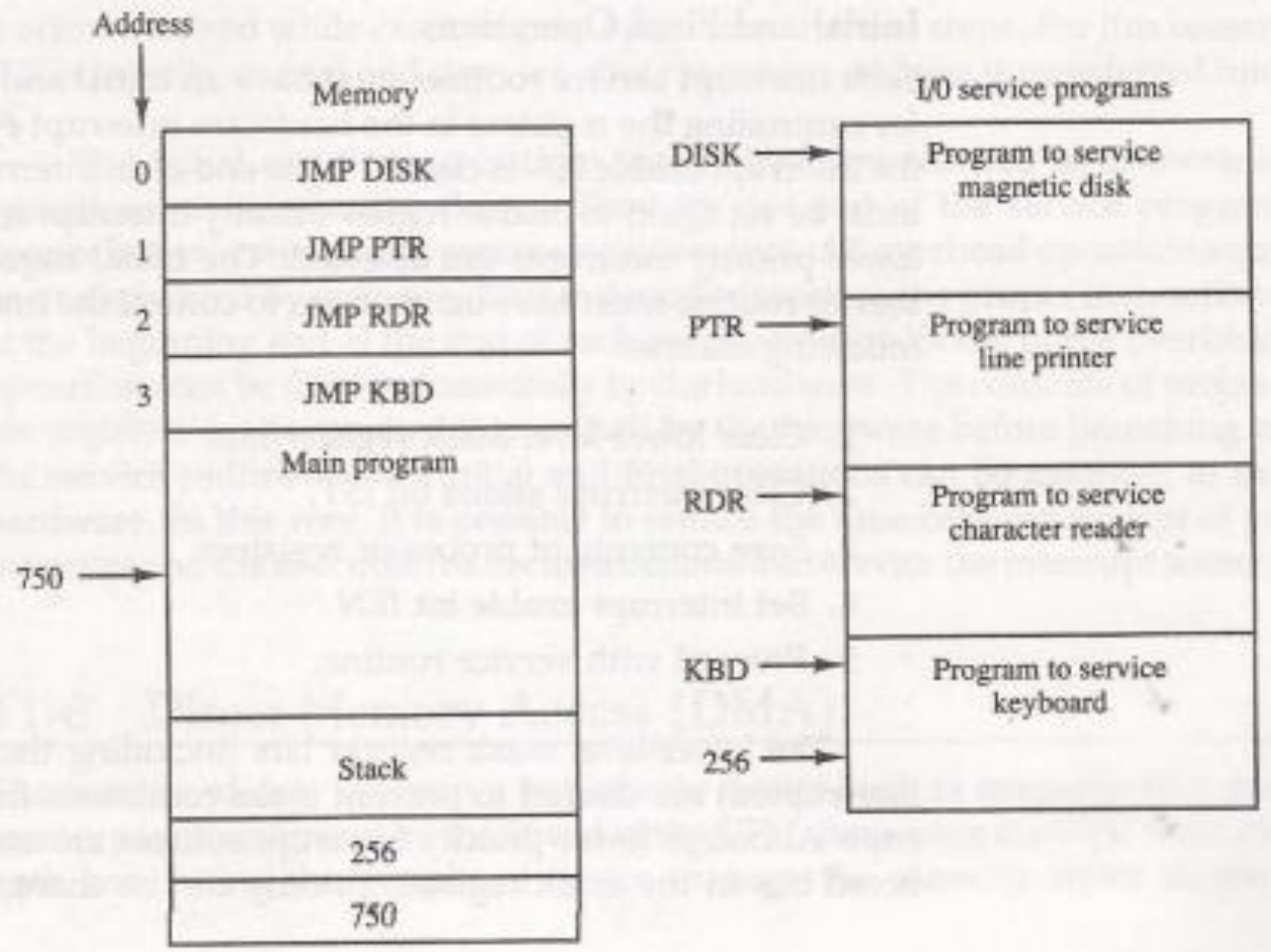


Figure 11-14 Priority interrupt hardware.

Figure 11-15 Programs stored in memory for servicing interrupts.



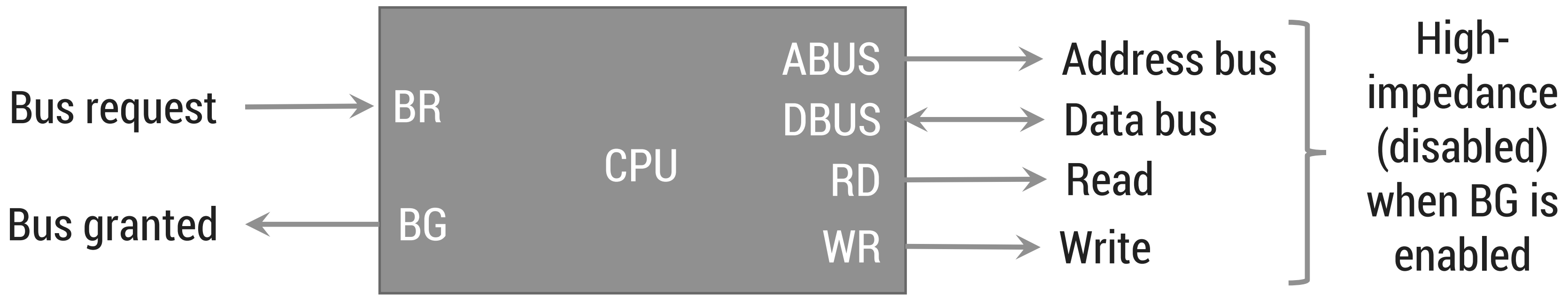
Address
Memory
0
JMP DISK
1
JMP PTR
2
JMP RDR
3
JMP KBD
Main program
750
Stack
256
750
I/0 service programs
DISK
Program to service magnetic disk
PTR
Program to service line printer
RDR
Program to service character reader
KBD
Program to service keyboard
256

- The initial sequence of each interrupt service routine must have instructions to control the interrupt hardware in the following manner:
- 1. Clear lower-level mask register bits.
- 2. Clear interrupt status bit IST.
- 3. Save contents of processor registers.
- 4. Set interrupt enable bit IEN.
- 5. Proceed with service routine.

- The final sequence in each interrupt service routine must have instrutions to control the interrupt hardware in the following manner:
- 1. Gear interrupt enable bit IEN.
- 2. Restore contents of processor registers.
- 3. Clear the bit in the interrupt register belonging to the source that has been serviced.
- 4. Set lower-level priority bits in the mask register.
- 5. Restore return address into PC and set IEN.

DMA (Direct Memory Access)

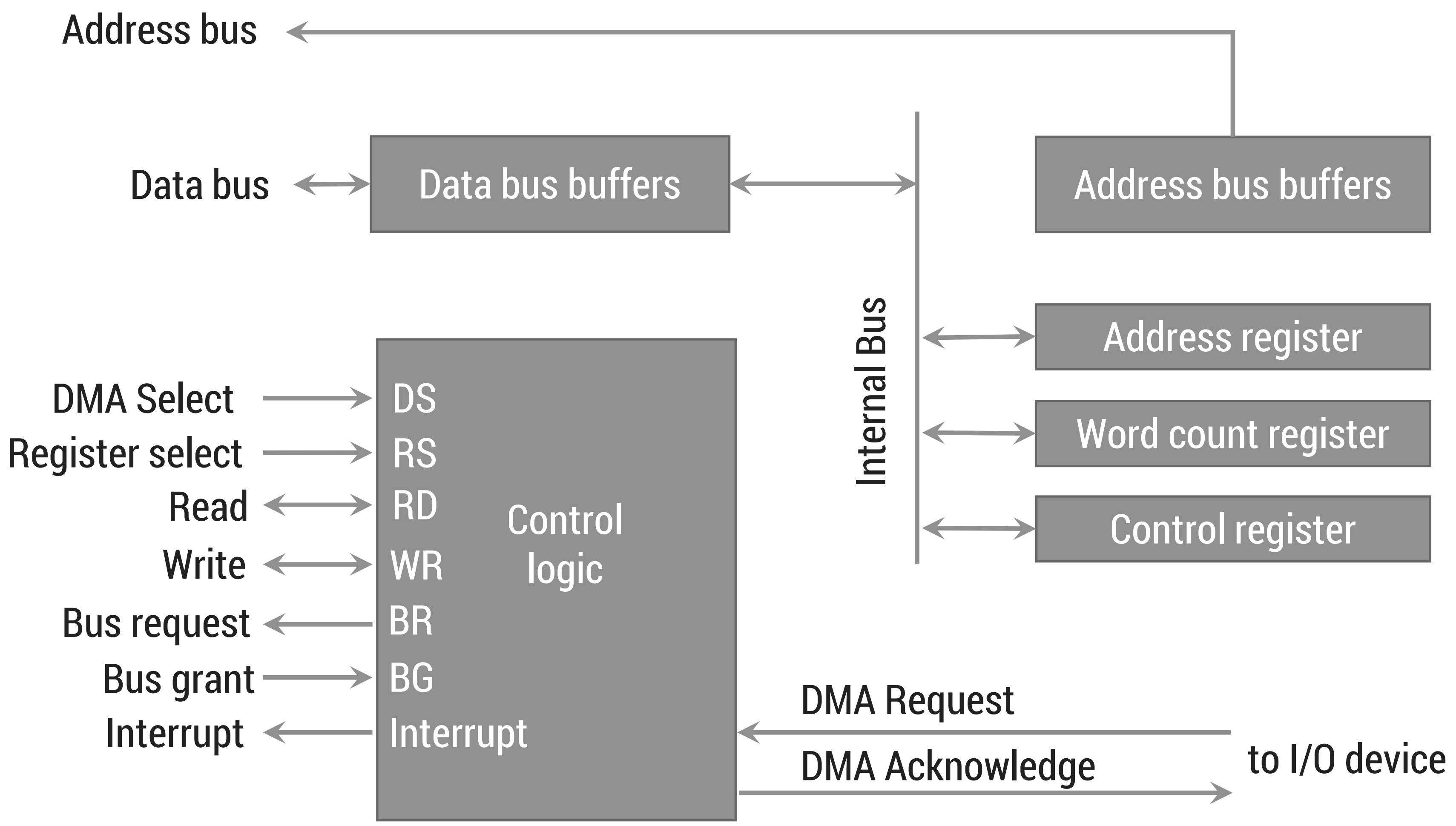
- The transfer of data between a fast storage device such as magnetic disk and memory is often limited by the speed of the CPU.
- Removing the CPU from the path and letting the peripheral device manage the memory buses directly would improve the speed of transfer.
- This transfer technique is called direct memory access (DMA).
- During DMA, CPU is idle and has no control of the memory buses.
- A DMA controller takes over the buses to manage the transfer directly between the I/O device and memory.



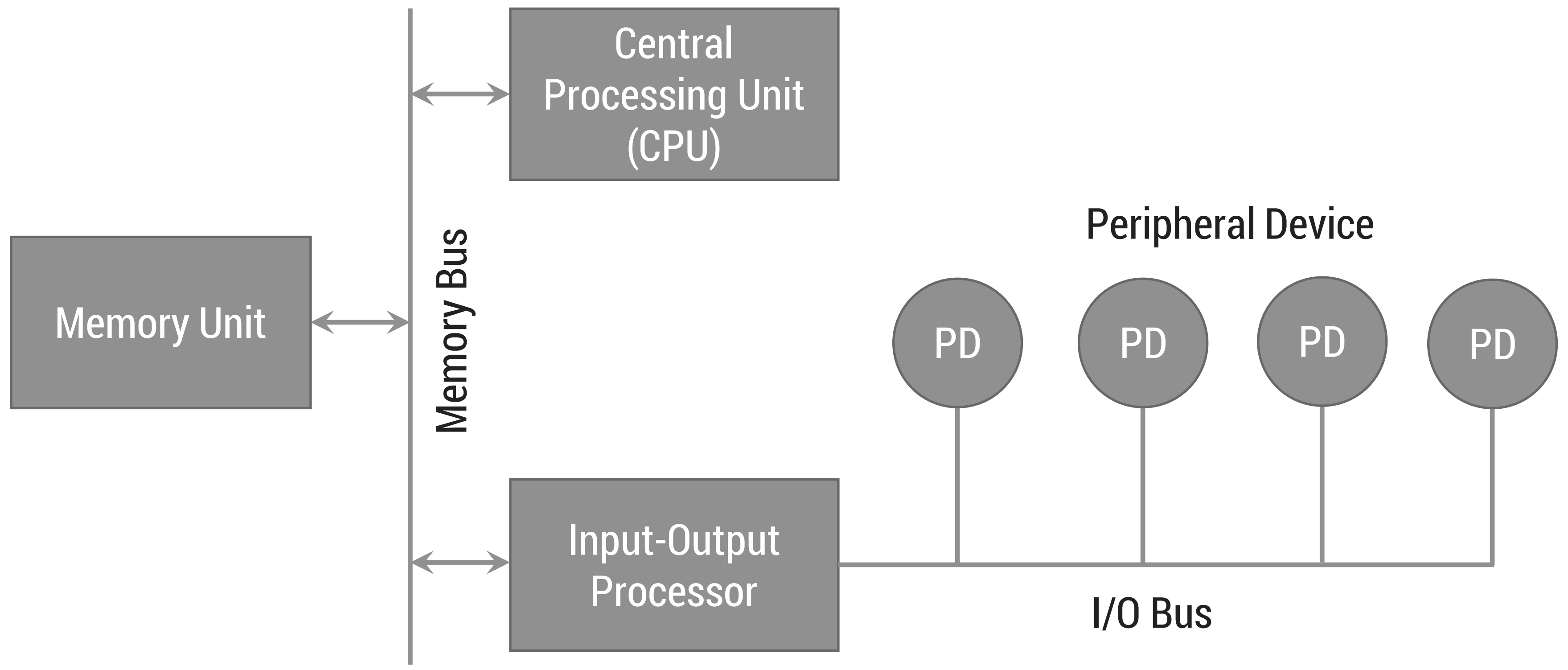
DMA Controller

DMA Controller

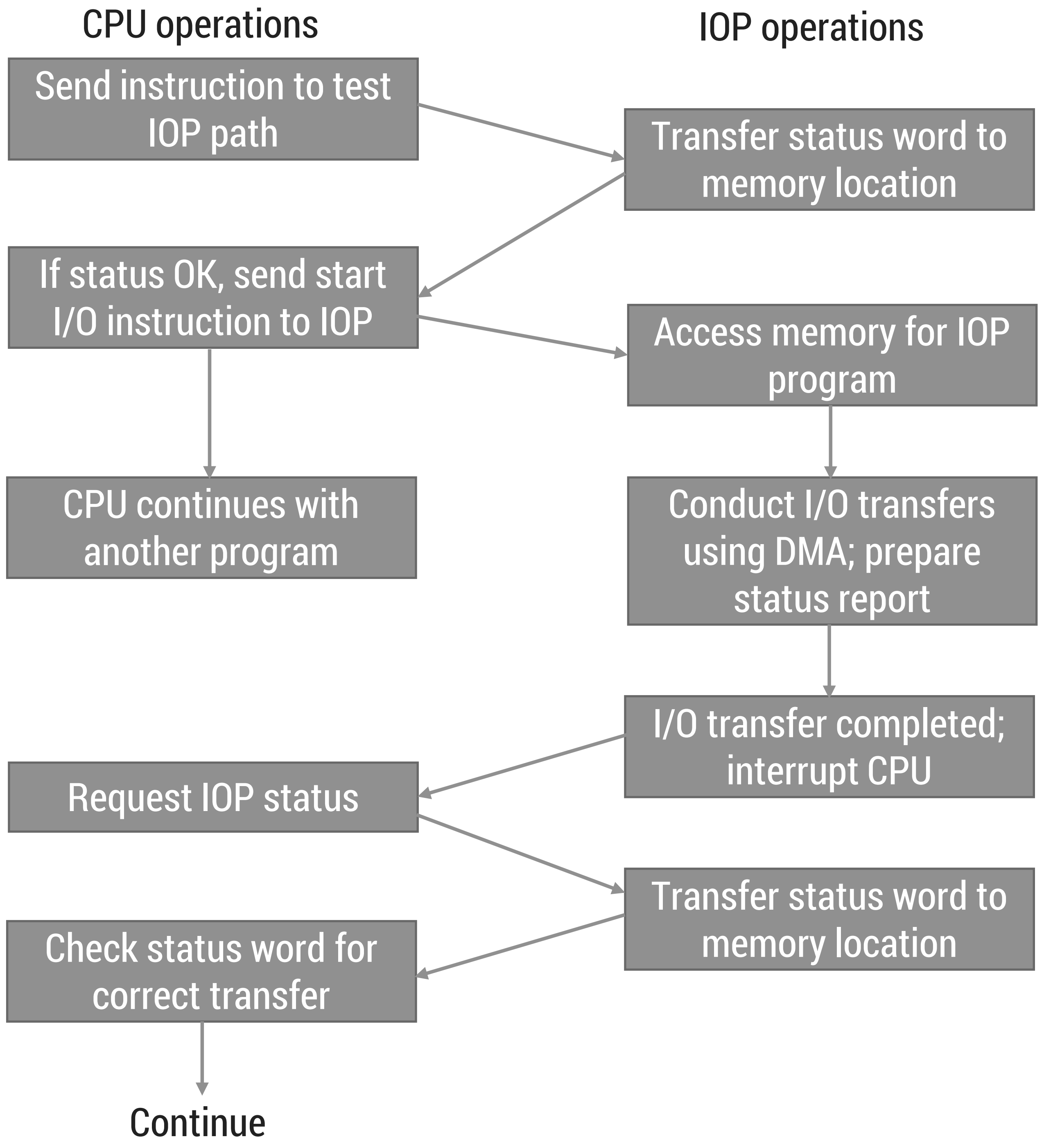
- DMA controller - Interface which allows I/O transfer directly between Memory and Device, freeing CPU for other tasks
- CPU initializes DMA Controller by sending memory address and the block size (number of words).



Input-Output Processor (IOP)



CPU – IOP Communication


CPU operations
IOP operations
Send instruction to test IOP path
Transfer status word to memory location
If status OK, 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
Transfer status word to memory location
Check status word for correct transfer
Continue