MODES OF TRANSFER

Binary information received from an external device is usually stored in memory for later processing.

Information transferred from the central computer into an external device originates in the memory unit.

Data transfer between the central computer and I/O devices may be handled in a variety of modes.

A. Programmed I/O
B. Interrupt- Initiated I/O
C. Direct Memory Access (DMA)

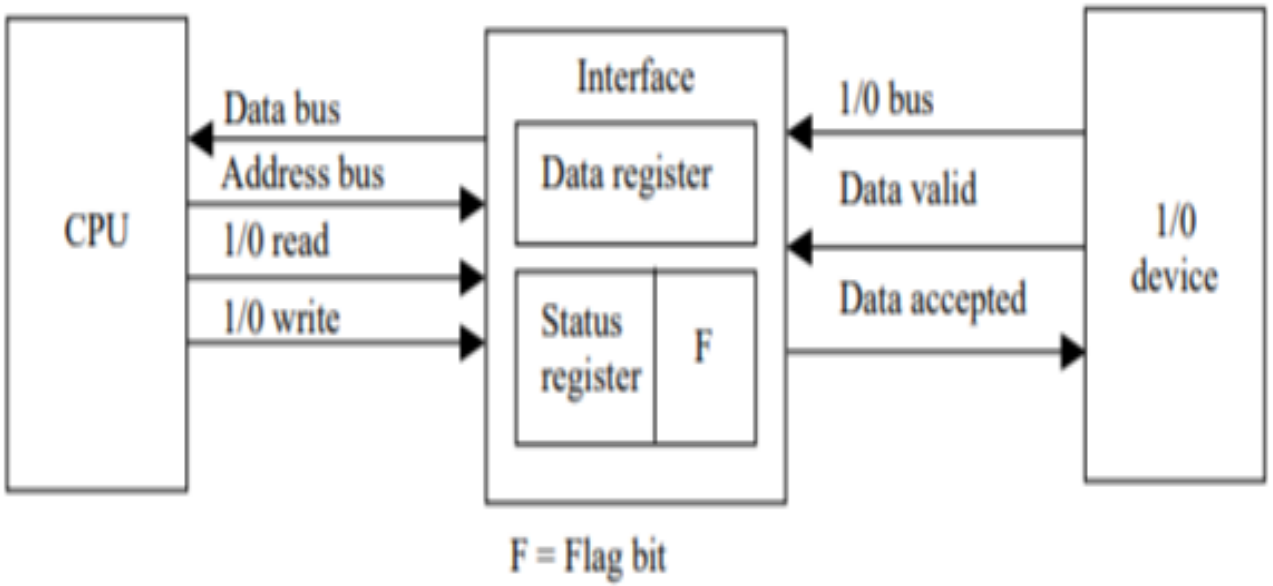
Programmed I/O

Programmed I/O operations are the result of I/O instructions written in the computer program. Each data item transfer is initiated by an instruction in the program.

Usually, the transfer is to and from a CPU register and peripheral. Other instructions are needed to transfer the data to and from CPU and memory.

Once a data transfer is initiated, the CPU is required to monitor the interface to see when a transfer can again be made.

In the programmed I/O method, the CPU stays in a program loop until the I/O unit indicates that it is ready for data transfer. This is a time-consuming process since it keeps the processor busy needlessly.



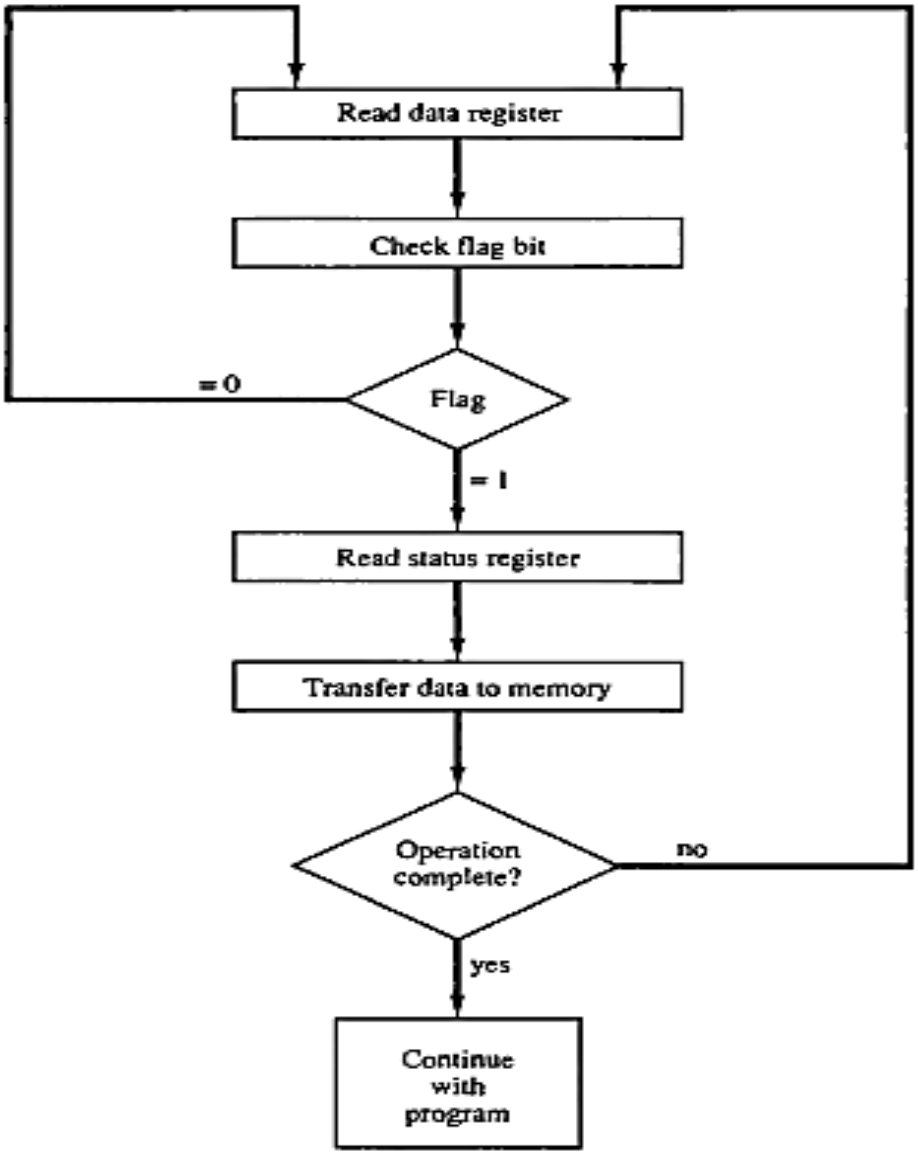


Figure 11-11 Flowchart for CPU program to input data.

A program is written for the computer to check the flag in the status register to determine if a byte has been placed in the data register by the I/O device.

This is done by reading the status register into a CPU register and checking the value of the flag bit.

If the flag is equal to 1, the CPU reads the data from the data register. The flag bit is then cleared to 0 by either the CPU or the interface, depending on how the interface circuits are designed.

Once the flag is cleared, the interface disables the data accepted line and the device can then transfer the next data byte.

The transfer of each byte requires three instructions:

1. Read the status register.
2. Check the status of the flag bit and branch to step 1 if not set or to step 3 if set.
3. Read the data register.

Disadvantage of Programmed I/O:

The problem with programmed I/O is that the processor has to wait a long time for the I/O module of concern to be ready for either reception or transmission of data.

The processor, while waiting, must repeatedly interrogate the status of the I/O module.

Performance of the entire system is severely degraded.

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.

This mode of transfer uses the interrupt facility. While the CPU is running a program, it does not check the flag.

However, when the flag is set, the computer is interrupted from proceeding with the 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 CPU responds to the interrupt signal by storing the return address from the program counter into a memory stack and then control branches to a service routine that processes the required I/O transfer. The way that the processor chooses the branch address of the service routine varies from tone unit to another.

In principle, there are **two methods** for accomplishing this. One is called **vectored interrupt** and the other, **no vectored 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.

SOFTWARE METHOD – POLLING

Polling process is used to identify the highest priority source by software. In this method, all interrupts are serviced by branching to the same service program.

This program then checks with each device if it is the one generating the interrupt. The order of checking is determined by the priority that has to be set.

The device having the highest priority is checked first and then devices are checked in descending order of priority.

If the device is checked to be generating the interrupt, another service program is called which works specifically for that particular device.

The major disadvantage of this method is that it is quite slow. To overcome this, we can use hardware solution, one of which involves connecting the devices in series. This is called Daisy-chaining method.

Hardware method- Daisy-Chain Interrupt

The daisy-chaining method of establishing priority consists of serial connection of all devices that request an interrupt.

The device with the highest priority is placed is the first position followed by the lower priority devices up to the device with the lowest priority, which is placed last in the chain.

This method of connection between three devices and the CPU

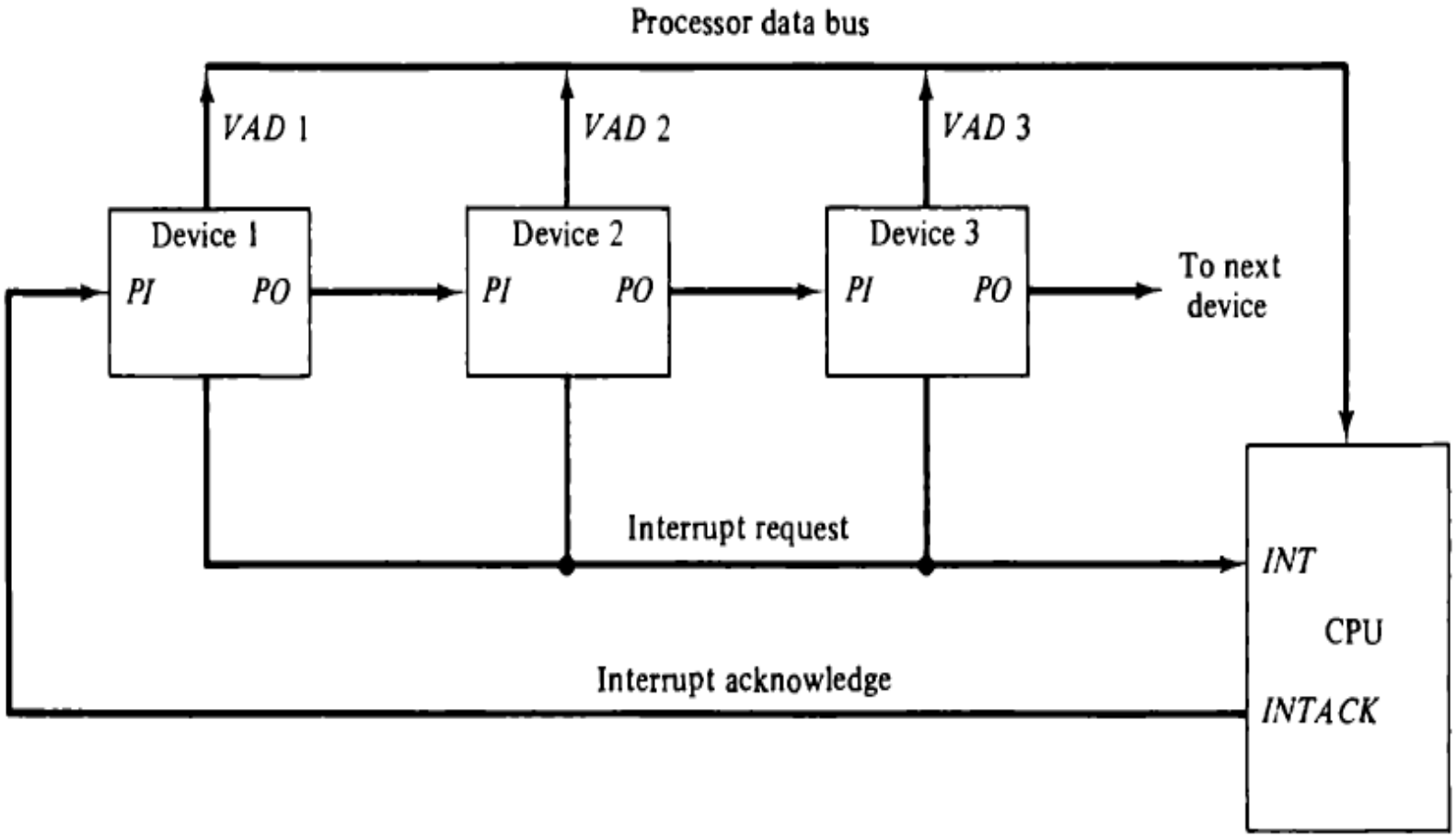


Figure 11-12 Daisy-chain priority interrupt.

The interrupt request line is common to all devices and form a wired logical connection.

Working

When no interrupts are pending, the line is in HIGH state. But if any of the devices raises an interrupt, it places the interrupt request line in the LOW state.

The CPU acknowledges this interrupt request from the line and then enables the interrupt acknowledge line in response to the request.

- This signal is received at the PI(Priority in) input of device 1.
- If the device has not requested the interrupt, it passes this signal to the next device through its PO(priority out) output. (PI = 1 & PO = 1)
- However, if the device had requested the interrupt, (PI =1 & PO = 0)
  - The device consumes the acknowledge signal and block its further use by placing 0 at its PO(priority out) output.
  - The device then proceeds to place its interrupt vector address(VAD) into the data bus of CPU.
  - The device puts its interrupt request signal in HIGH state to indicate its interrupt has been taken care of.

Interrupt signal state- High –No interrupt

Interrupt signal state- Low –interrupt

- Interrupt Acknowledge – PI=1 , PO=1 (No interrupt request by device)
- Interrupt Acknowledge – PI=1 , PO=0 (interrupt request by device) (High Priority)
- Interrupt Acknowledge – PI=0 , PO=0 (interrupt request by device)(Low priority)

PO=0 (Block the acknowledge signal)

Direct Memory Access(DMA)

Direct Memory Access (DMA) transfers the block of data between the memory and peripheral devices of the system, without the participation of the processor. The unit that controls the activity of accessing memory directly is called a DMA controller.

Removing the CPU from the path and letting the peripheral devices manage the memory bus directly.

During the DMA transfer, the 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.

The CPU may be placed in an idle state in a variety of ways. One common method extensively used in microprocessor is to disable the buses through special control signals such as:

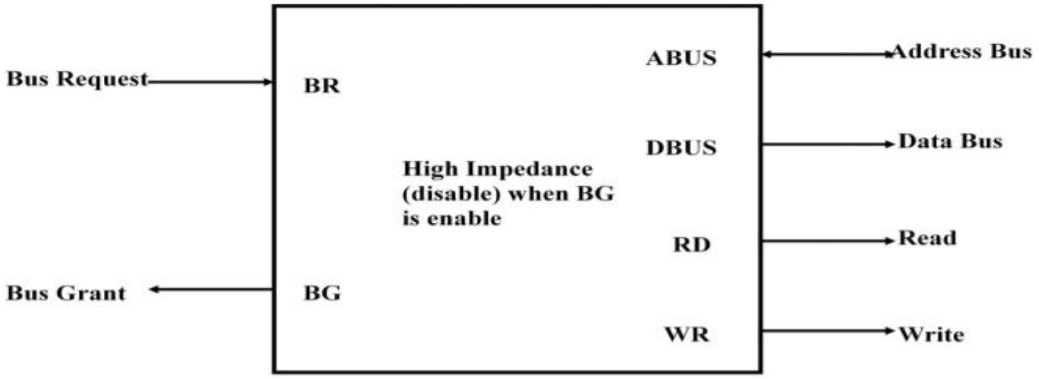
- **Bus Request (BR)**
- **Bus Grant (BG)**

These two control signals in the CPU that facilitates the DMA transfer.

The Bus Request (BR) input is used by the DMA controller to request the CPU.

When this input is active, the CPU terminates the execution of the current instruction and places the address bus, data bus and read write lines into a high Impedance state.

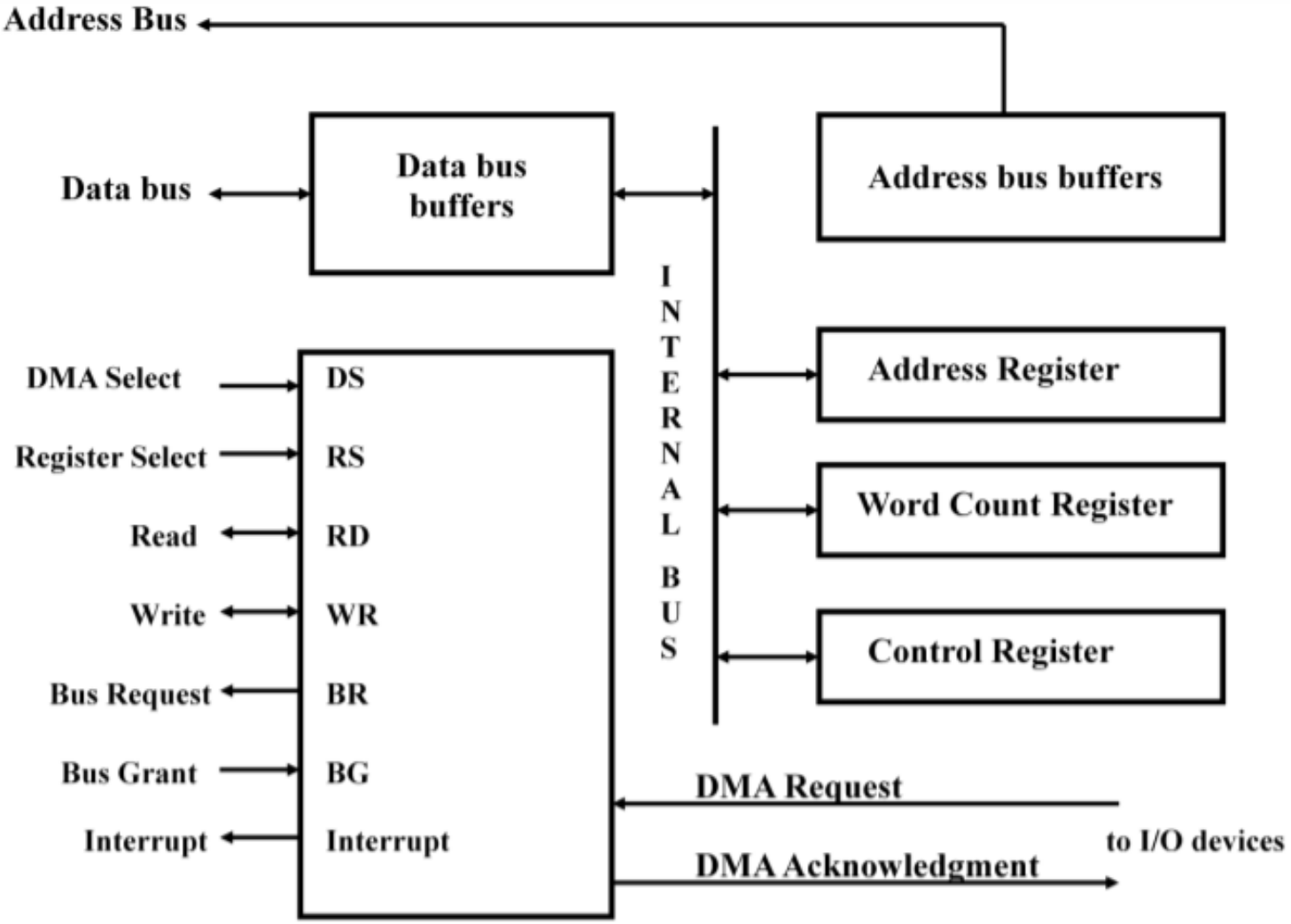
High Impedance state means that the output is disconnected.



CPU bus Signals for DMA Transfer

The CPU activates the Bus Grant (BG) output to inform the external DMA that the Bus Request (BR) can now take control of the buses to conduct memory transfer without processor.

When the DMA terminates the transfer, it disables the Bus Request (BR) line. The CPU disables the Bus Grant (BG), takes control of the buses and return to its normal operation.



Block Diagram of DMA Controller

The transfer can be made in several ways that are:

- i) DMA Burst ii. Cycle Stealing

i) DMA Burst :- In DMA Burst transfer, a block sequence consisting of a number of memory words is transferred in continuous burst while the DMA controller is master of the memory buses.

ii) Cycle Stealing :- Cycle stealing allows the DMA controller to transfer one data word at a time, after which it must returns control of the buses to the CPU.

DMA Controller: The DMA controller needs the usual circuits of an interface to communicate with the CPU and I/O device. The DMA controller has three registers:

i. Address Register :- Address Register contains an address to specify the desired location in memory.
ii. Word Count Register :- WC holds the number of words to be transferred. The register is increase/decrease by one after each word transfer and internally tested for zero.
iii. Control Register :- Control Register specifies the mode of transfer

The unit communicates with the CPU via the data bus and control lines.

The registers in the DMA are selected by the CPU through the address bus by enabling the DS (DMA select) and RS (Register select) inputs.

The RD (read) and WR (write) inputs are bidirectional. When the BG (Bus Grant) input is 0, the CPU can communicate with the DMA registers through the data bus to read from or write to the DMA registers.

When BG =1, the DMA can communicate directly with the memory by specifying an address in the address bus and activating the RD or WR control.