35

COA

CH-4

I/O Organization

BY

Dr. Raghad Samir Al Najim

SPU@ 2017

I/O Organization

The Input / output organization of computer depends upon the size of computer and the peripherals connected to it.

The most common input output devices are:

- Monitor
- Keyboard
- Mouse
- Printer
- Magnetic tapes

Input - Output Interface

Input Output Interface provides a method for transferring information between internal storage and external I/O devices.

Peripherals connected to a computer need special communication links for interfacing them with the CPU.

The purpose of communication link is to resolve the **differences** that exist between the central computer and each peripheral.

The Major Differences are:-

1. Peripherals are **electromechnical** and **electromagnetic** devices and their manner of operation is different of the CPU and memory, which are electronic devices. Therefore, a conversion of signal values may be needed.
2. The **data transfer rate** of peripherals is usually slower than the transfer rate of CPU , so synchronization mechanism may be needed.

The Major Differences are:-

3. **Data codes and formats** in the peripherals differ from the word format in the CPU and memory.
4. **The operating modes** of peripherals are different from each other and must be controlled so as not to disturb the operation of other peripherals connected to the CPU.

Input - Output Interface

To Resolve these differences, computer systems include special hardware components between the CPU and Peripherals to supervises and synchronizes all input and out transfers. These components are called **Interface Units** because they interface between the processor bus and the peripheral devices.

I/O BUS & Interface Module

It defines the typical link between the processor and several peripherals. The I/O Bus consists of **data lines, address lines** and **control lines**. The I/O bus from the processor is attached to all peripherals interface. To communicate with a particular device, the processor places a device address on address lines.

I/O BUS & Interface Module

Each **Interface decodes** the address and control received from the I/O bus, interprets them for peripherals and provides signals for the **peripheral controller**. It is also synchronizes the data flow and supervises the transfer between peripheral and processor. **Each peripheral has its own controller**. **For example**, the printer controller controls the paper motion, the print timing.

I/O BUS & Interface - I/0 commands

The control lines are referred as an **I/O commands**. The commands are as following:

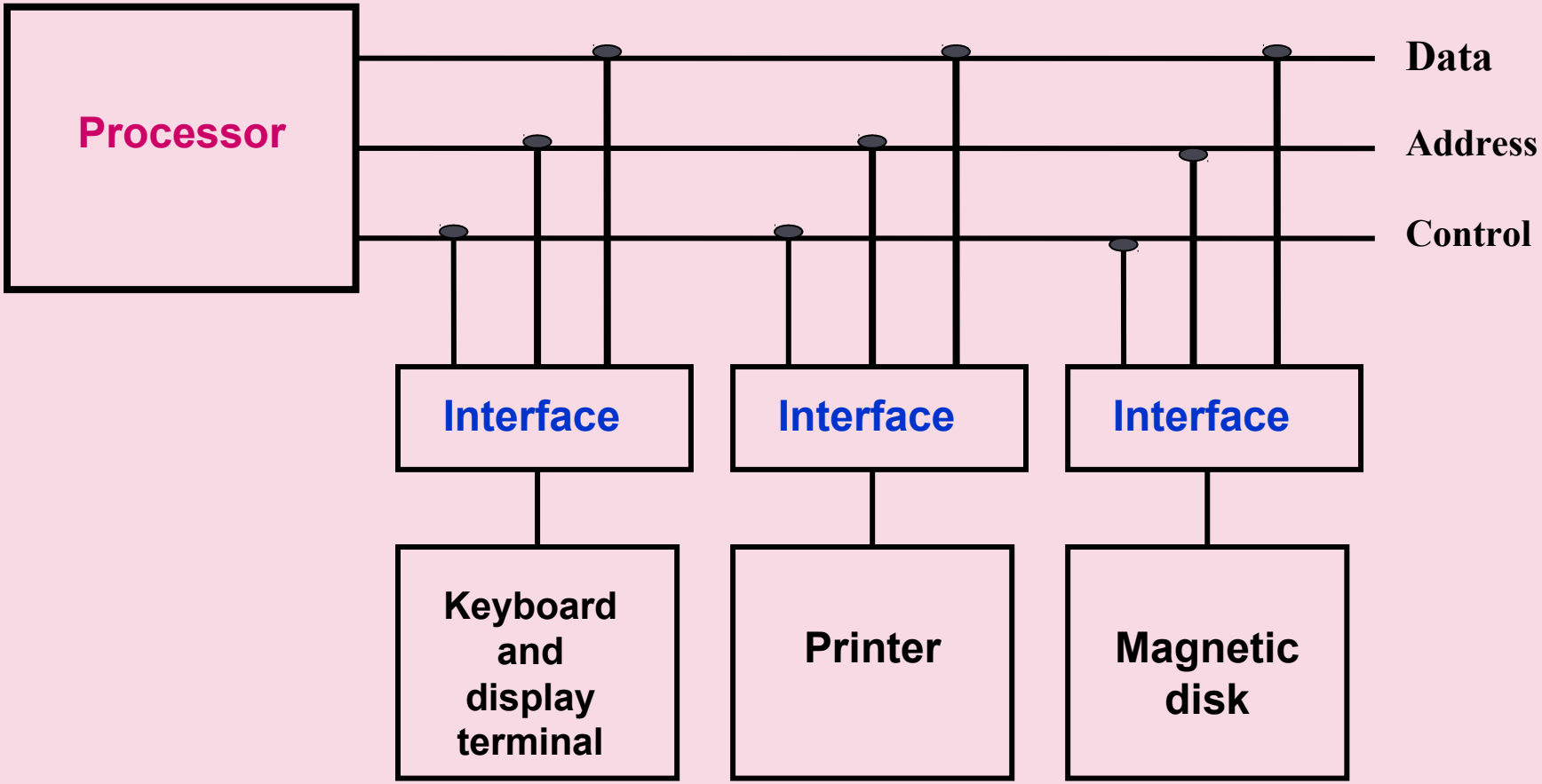
Control command- A control command is issued to activate the peripheral and to inform it what to do.

Status command- A status command is used to test various status conditions in the interface and the peripheral.

Output data command- A data output command causes the interface to respond by transferring data from the bus into one of its registers.

Input data command- In this case the interface receives one item of data from the peripheral and places it in its buffer register.

I/O BUS and Interface Module



Connection of I/O bus to input-output devices

I/O Versus Memory Bus

CPU communicates with I/O, and it communicates with the memory unit too. Like the I/O bus, the memory bus contains data, address and read/write control lines.

There are 3 ways that computer buses can be used to communicate with memory and I/O:

- Use two Separate buses , one for memory and other for I/O.
- Use one common bus for both memory and I/O but with separate control lines for each.
- Use one common bus for memory and I/O with common control lines.

I/O Versus Memory Bus

In the first method, the computer has independent sets of data, address and control buses one for accessing memory and other for I/O.

The purpose of this type is to provide an independent pathway for the transfer of information between external device and internal memory.

- **In second method** computers use common bus to transfer data between IO or memory and CPU. And separate read and write lines.
- **IO read/write is enabled during IO** transfer and **memory read/write is enabled during memory transfer.** This configuration isolates all I/O interface address with the memory address and is referred to as **Isolated IO method.**

- **In third method** same address space is used for both memory and IO interface. They have only one set of read and write signals.
- Computer treats interface as being part of the memory system.
- It assign address for each interface which cannot be used for memory words.
- This configuration is referred to as **memory-mapped IO**.

Modes of Data Transfer

Transfer of data is required between CPU and peripherals or memory or sometimes between any two devices or units of your computer system.

To transfer a data from one unit to another one should be sure that both units have proper connection and at the time of data transfer the receiving unit is not busy. This data transfer with the computer is **Internal Operation.**

Modes of Data transfer

Steps for transferring data between CPU and I/O devices:

- ✓ **CPU checks I/O module status**
- ✓ **I/O module returns status**
- ✓ **If ready, CPU requests data transfer**
- ✓ **I/O interface gets data from device**
- ✓ **I/O interface transfers data to CPU**

Data transfer between computer and I/O device can be handle in the following three ways:

1) Programmed I/O Mode

In this mode of data transfer the operations are the results in I/O instructions which is a part of computer program. Each data transfer is initiated by a instruction in the program. Normally the transfer is from a CPU register to peripheral device or vice-versa.

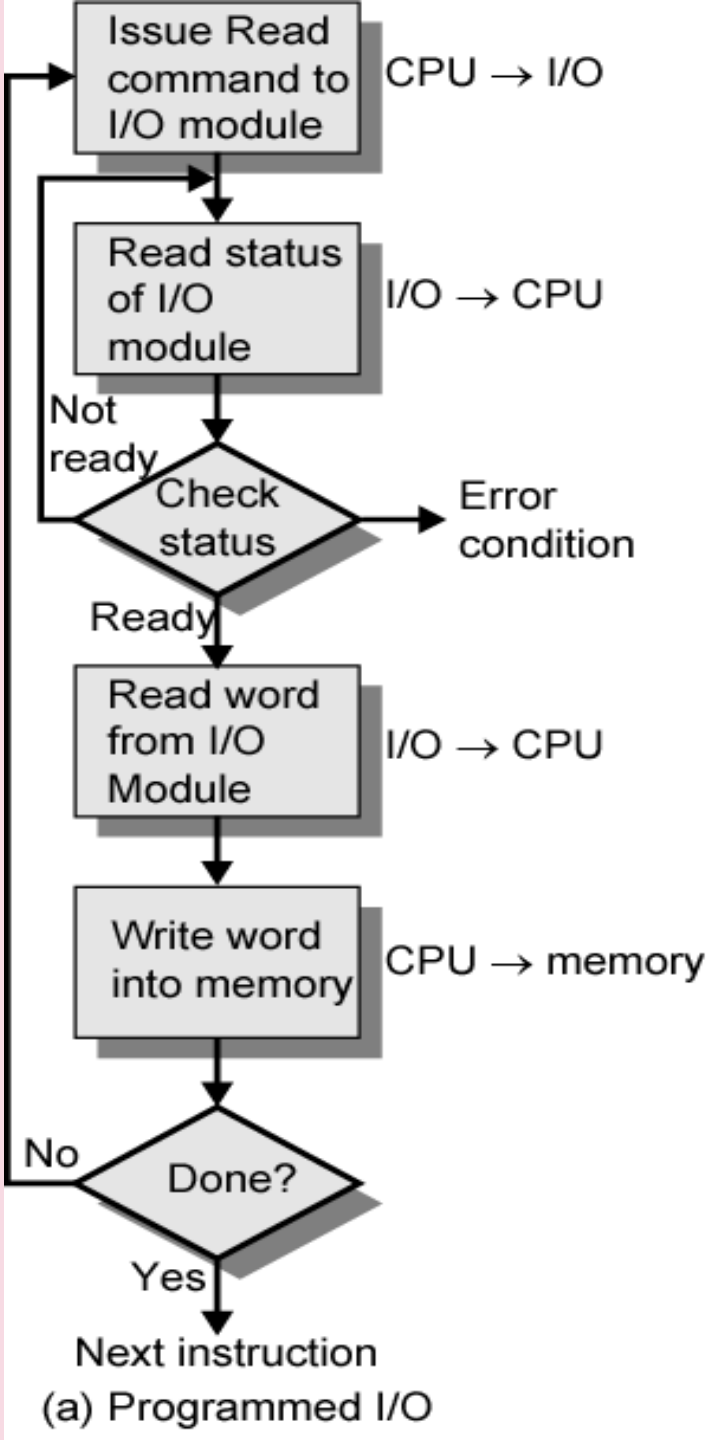
- **Operation:-**

1. **Address the Device.**
2. **Check its status.**
3. **IF ready then perform operation else check another device**

1) Programmed I/O Mode

Once the data is initiated, the CPU starts monitoring the interface to see when next transfer can made.

In this technique CPU is responsible for executing data from the memory for output and storing data in memory for executing of Programmed I/O as shown in Flowchart-:



(a) Programmed I/O

Disadvantage of the Programmed I/O

The main drawback of the Program Initiated I/O was that the CPU has to monitor the units all the times when the program is executing. Thus 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 and the CPU time is wasted a lot in keeping an eye to the executing of program.

To remove this problem an Interrupt facility and special commands are used.

2) Interrupt-Initiated I/O

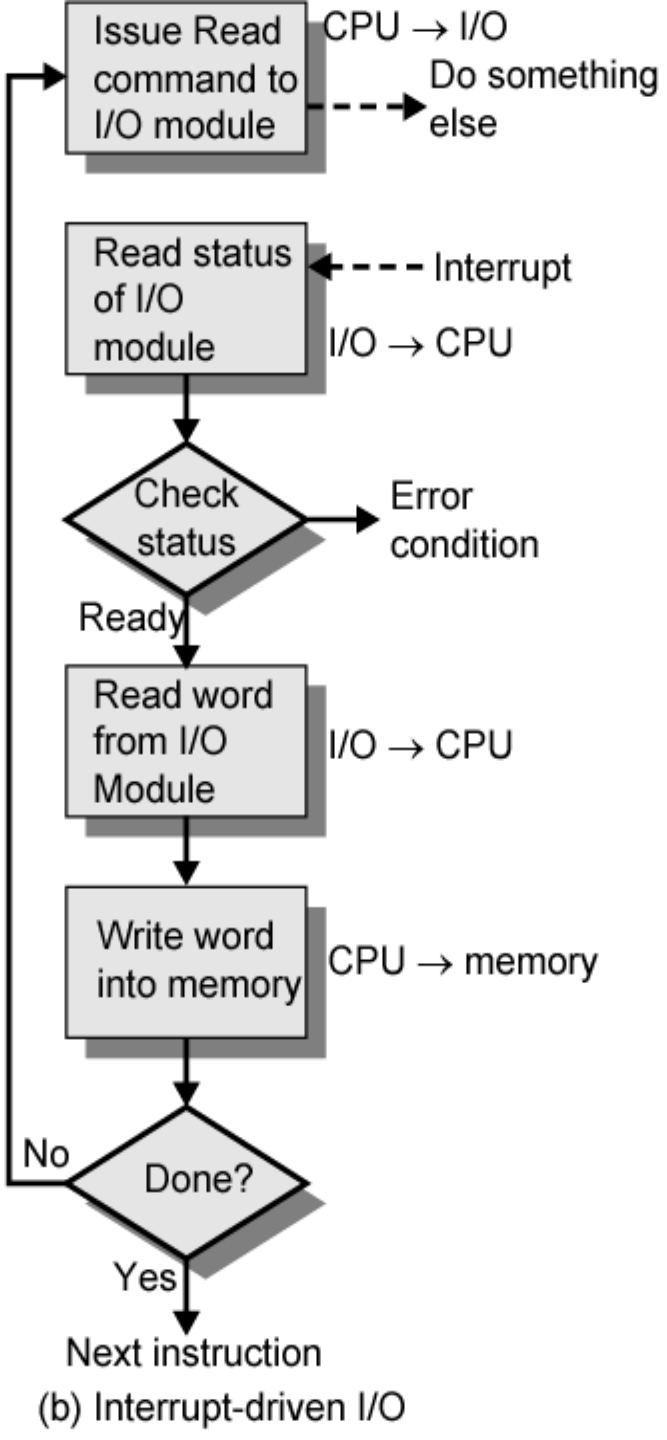
In this method an interrupt facility is used, an **interrupt command** is used to inform the device about the start and end of transfer. In the meantime the CPU executes other program. When the interface determines that the device is ready for data transfer it generates an **Interrupt Request** and sends it to the computer.

2) Interrupt-Initiated I/O

When the CPU receives an interrupt signal, it temporarily stops the execution of the program and branches to a service program(routine) to process the I/O transfer and after completing it returns back to task, what it was originally performing.

The Execution process of Interrupt–Initiated I/O is represented in the flowchart:

- **Operation:-**
- **Device interrupt's CPU**
- **CPU acknowledge the I/O device.**
- **CPU perform operation**

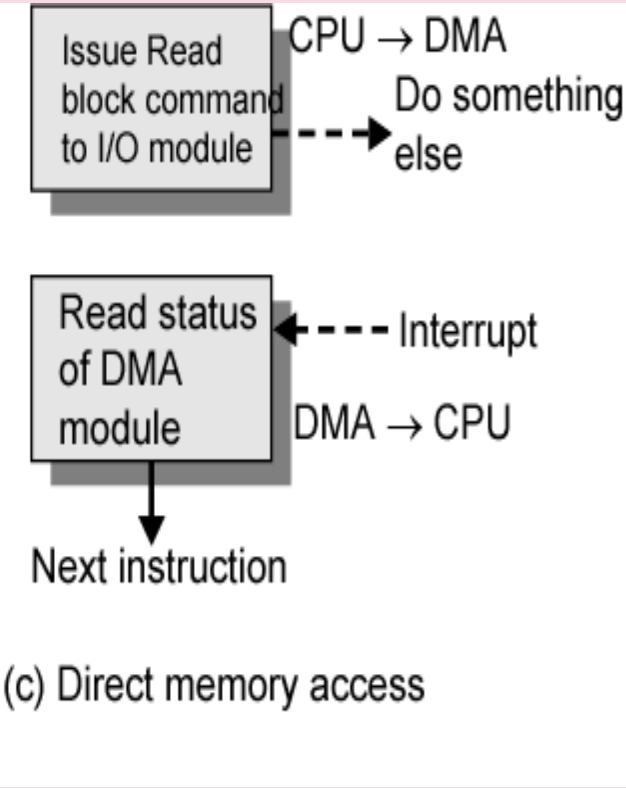


(b) Interrupt-driven I/O

3) Direct Memory Access (DMA)

In the Direct Memory Access (DMA) the interface transfer the data into and out of the memory unit through the system(memory) bus.

The transfer of data between a 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).**



(c) Direct memory access

DMA

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 , this is done by **disabling the buses** using special control signals such as:

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

DMA

These two control signals are:

The HOLD Signal is a Bus Request (BR)

input to CPU as a request from ***DMA controller*** to 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.

DMA

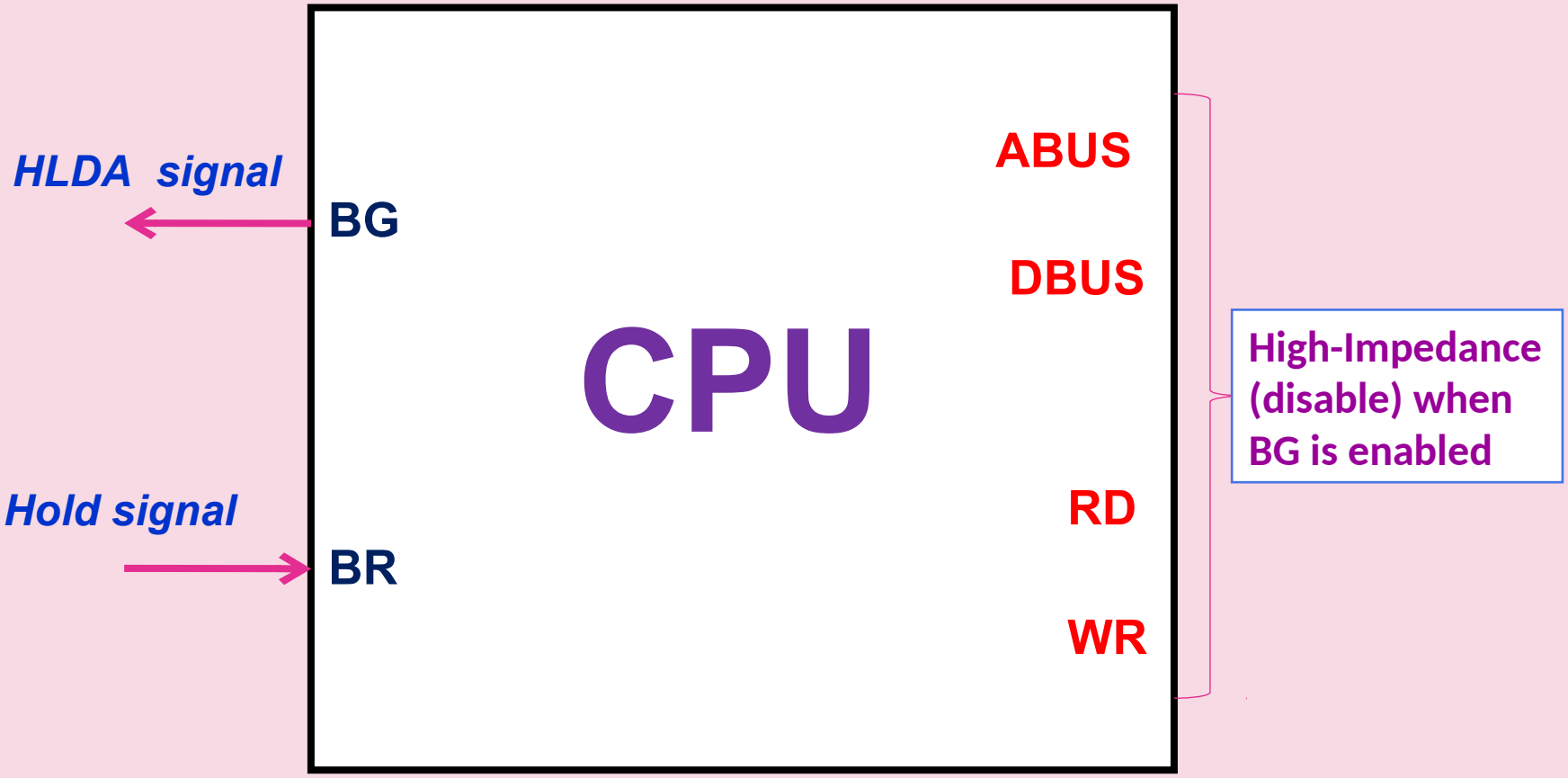
These two control signals are:

The HLDA signal is a Bus Grant (BG)

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

Or that the buses can now be used without the processor control.

CPU bus signals for DMA transfer

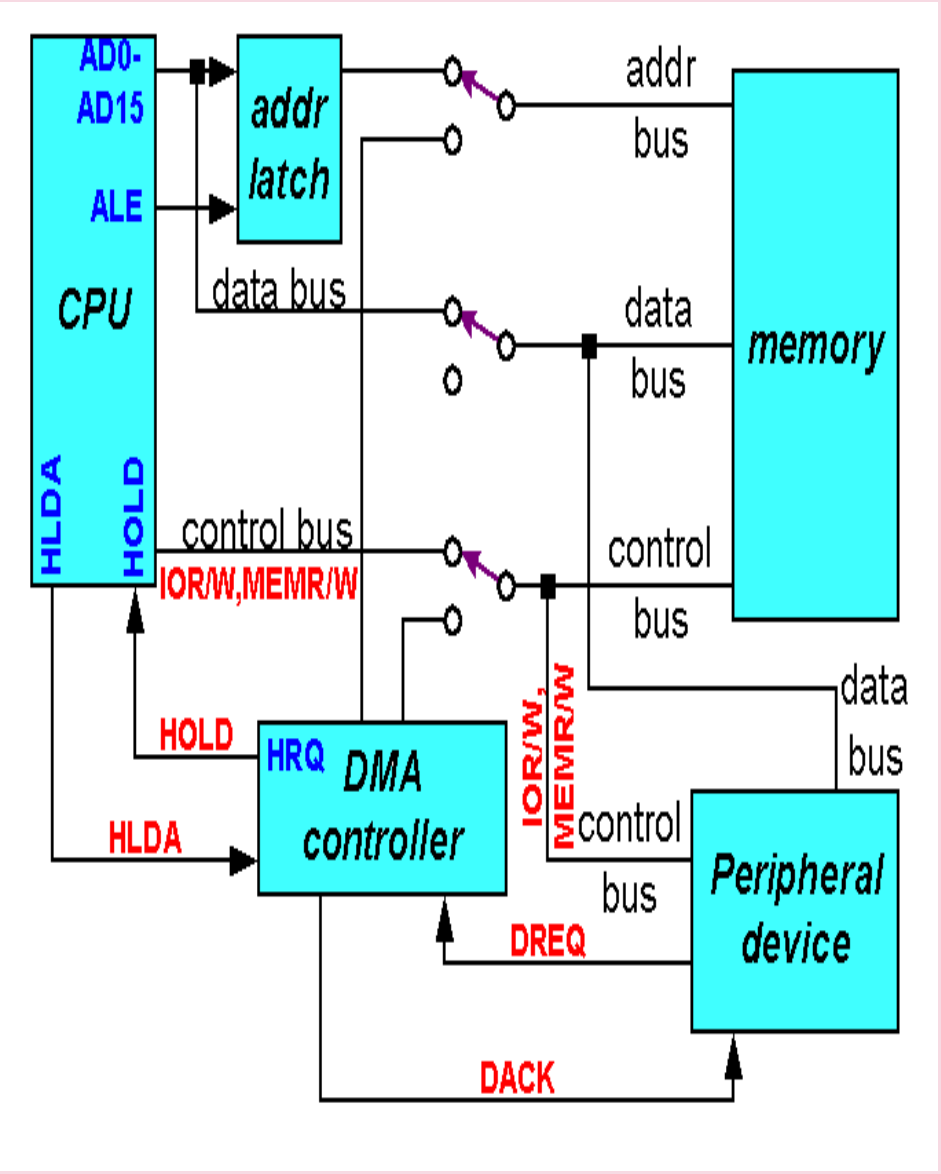


BG ⊟ BUS GRANT ; BR ⊟ BUS REQUEST

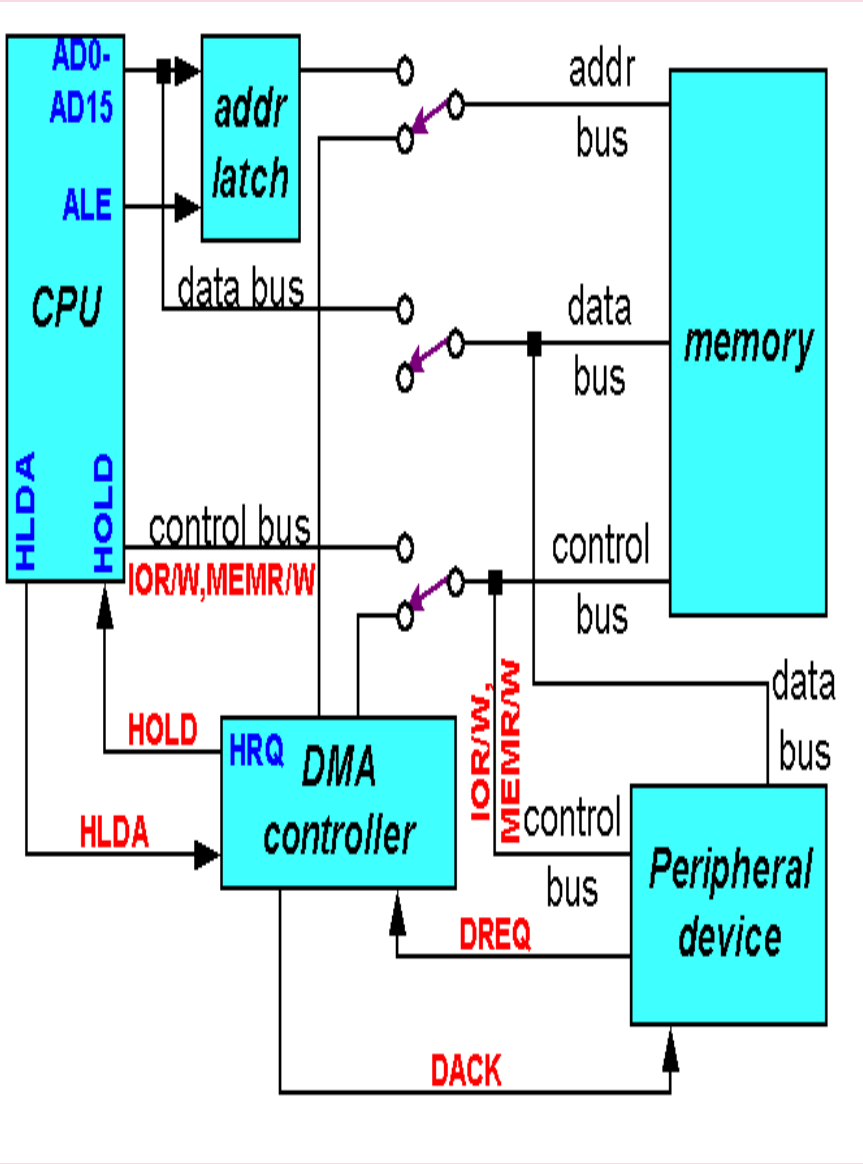
DMA vs. NO DMA

Without DMA	With DMA
When the CPU is using programmed I/O, it is typically fully occupied for the entire duration of the read or write operation, and is thus unavailable to perform other work .	The CPU initiates the transfer, does other operations while the transfer is in progress, and receives an Interrupt from the DMA controller when the operation is done.

This feature is useful any time the CPU cannot keep up with the rate of data transfer, or where the CPU needs to perform useful work while waiting for a relatively slow I/O data transfer.



Without DMA



With DMA

DMA

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.

The transfer can be made in several ways that are:

A) DMA Burst

B) Cycle Stealing

DMA

1) 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.

2) 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

It is sometimes referred to as a channel.

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:

1. **Address Register**
2. **Word Count Register**
3. **Control Register**

DMA Controller

1. Address Register :- Address Register contains an address to specify the desired location in memory.
2. Word Count Register :- WC holds the number of words to be transferred. The register is decreased by one after each word transfer and internally tested for zero.
3. Control Register accepts commands from the CPU. It is also treated as an O/P port by the CPU.
4. Data Register are used to store intermediate data values and result when any arithmetic operation is performed.

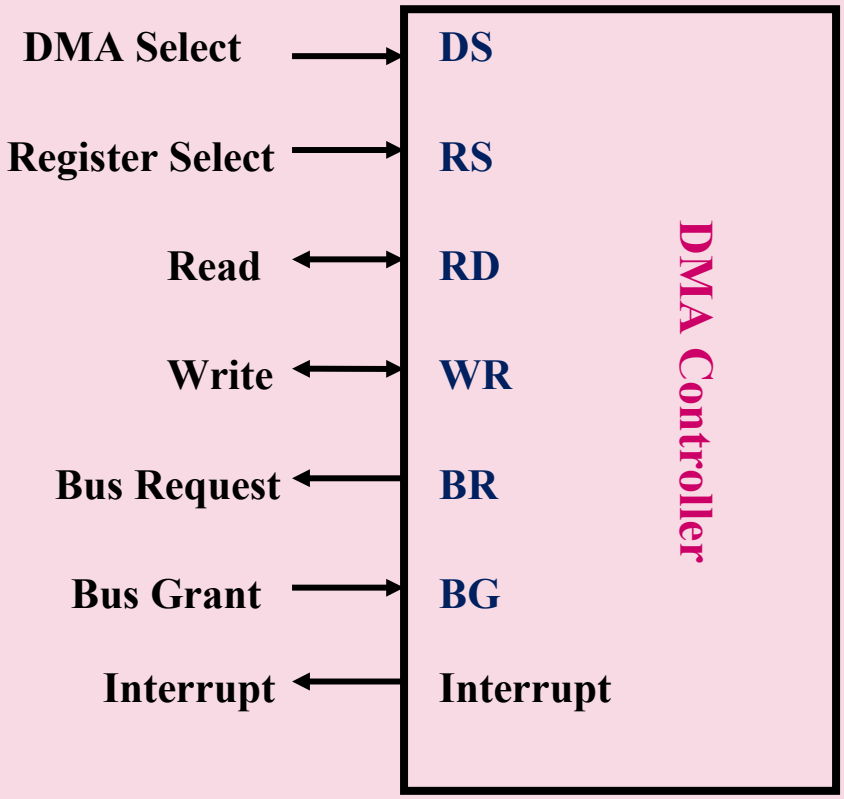
CPU tells DMA controller:-

- Read/Write
- Device address
- Starting address of memory block for data
- Amount of data to be transferred
- CPU carries on with other work

- DMA controller deals with transfer
- DMA controller sends interrupt when finished

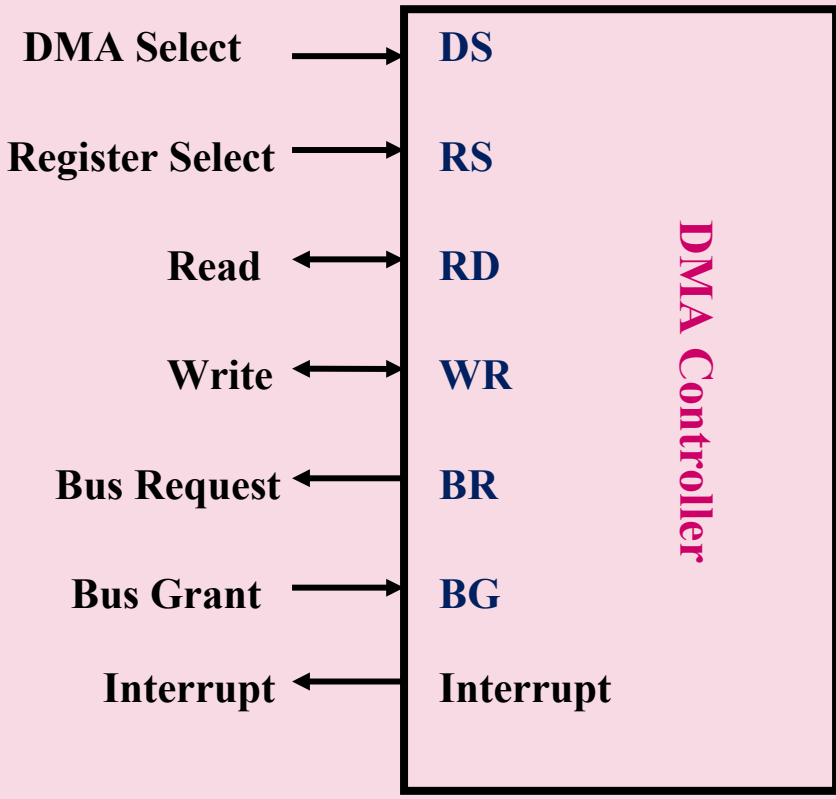
DMA Controller

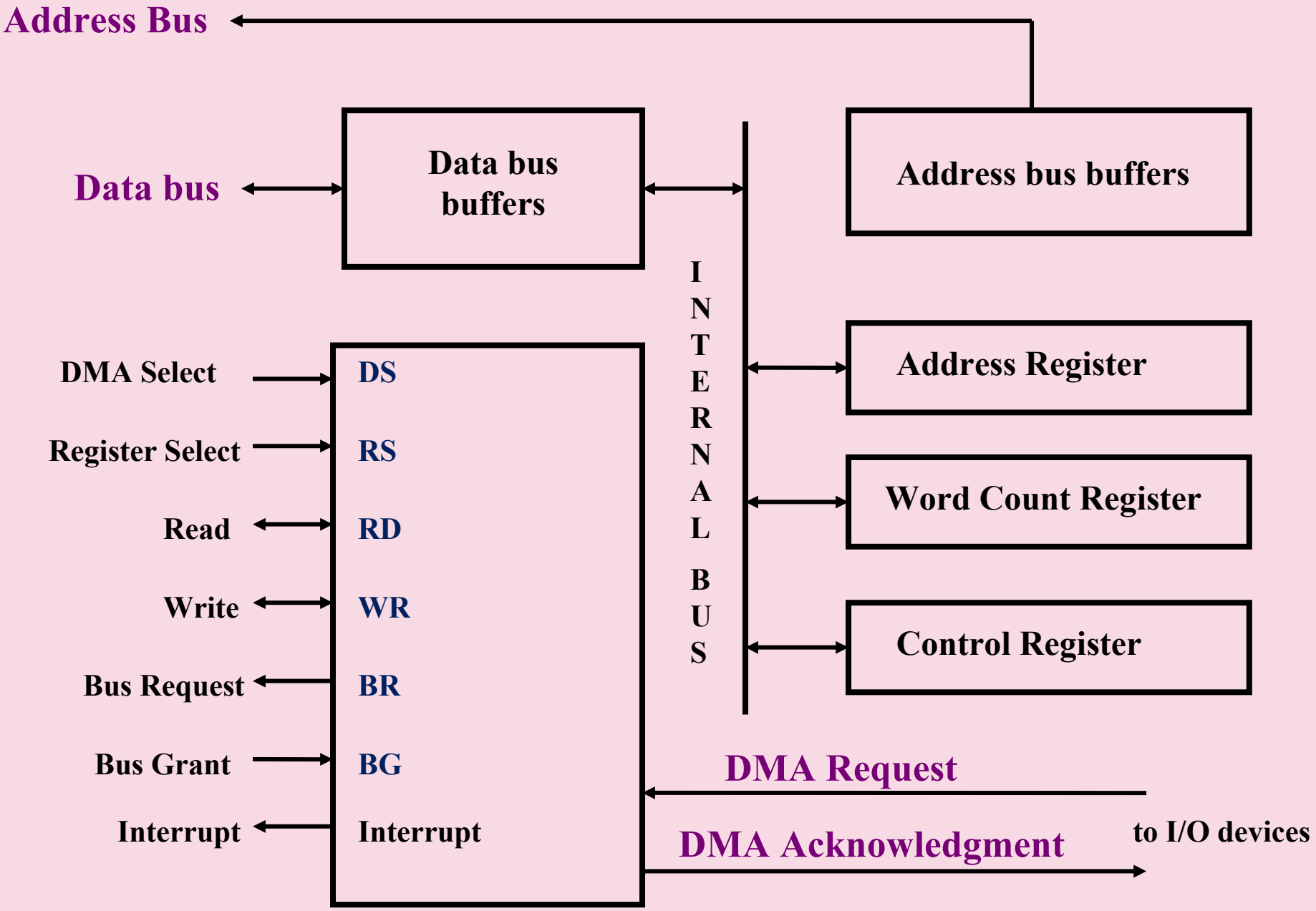
The registers in the DMA are selected by the CPU through the address bus by enabling:



- 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.





Block Diagram of DMA Controller

Ex:. transfer data between Disk & Memory

1. Device requests service from DMA by putting DREQ (DMA request) high
2. DMA puts high on HRQ (hold request),
3. CPU finishes its present bus cycle , puts high on HDLA (hold acknowledge).
4. DMA activates DACK (DMA acknowledge), telling device to start transfer
5. DMA starts transfer by putting address of first byte on address bus and activating MEMR; it then activates IOW to write to peripheral. DMA decrements counter and increments address pointer. Repeat until count reaches zero
6. DMA deactivates HRQ, giving buses under the control of CPU

Ex:. transfer data between Disk & Memory