Input Output Interfacing

I/O Interfacing

One of the essential features of a computer is its ability to send and receive data to and from other devices.

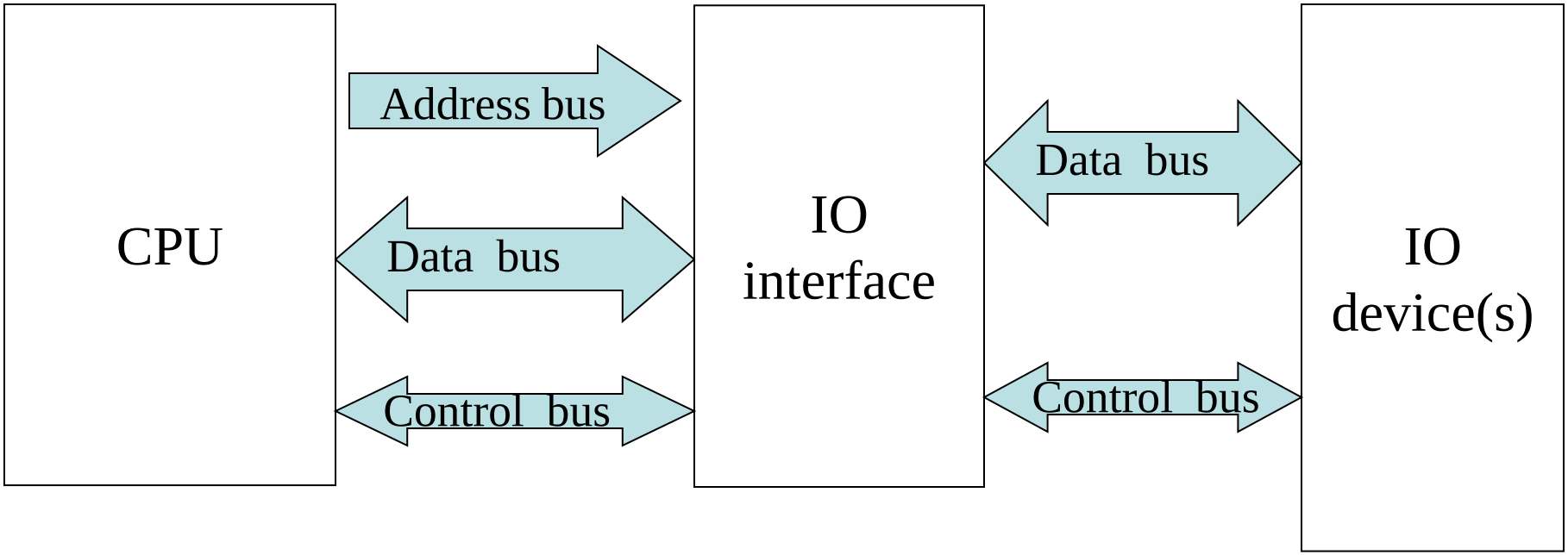
E.g. Keyboards, Display units, Printers, Disks, Tapes etc..

Further Voltmeters, Temperature sensors, Fire alarms, and digitally controlled activators.

A general purpose computer should have the ability to deal with a wide range of device characteristics. These can be analog or digital devices.

I/O device characteristics and the CPU characteristics are matched by a matching circuit **an interface/IO controller**.

A Simplified Block Diagram



CPU select a particular IO interface by sending an address through the address bus. Then the IO device(s) (sometimes may be more than one device) which is (are) connected to the interface, can communicate with the CPU (when there are more than one device interface do the selection).

Functions performed by an IO interface

1. **Device Selection**

   Multiple IO devices are generally connected to a CPU. Therefore each device interface must be assigned to a unique address to identify it from the rest.

2. **Device Status Storage**

   IO devices are generally much slower than the CPU. Therefore when CPU requests the service of an IO device the interface should be able to provide the status information of the device. For this function some storage must be provided within the interface.

3. **Data Storage**

   IO interface should provide some storage buffer, for the data being transferred between the CPU an IO device.

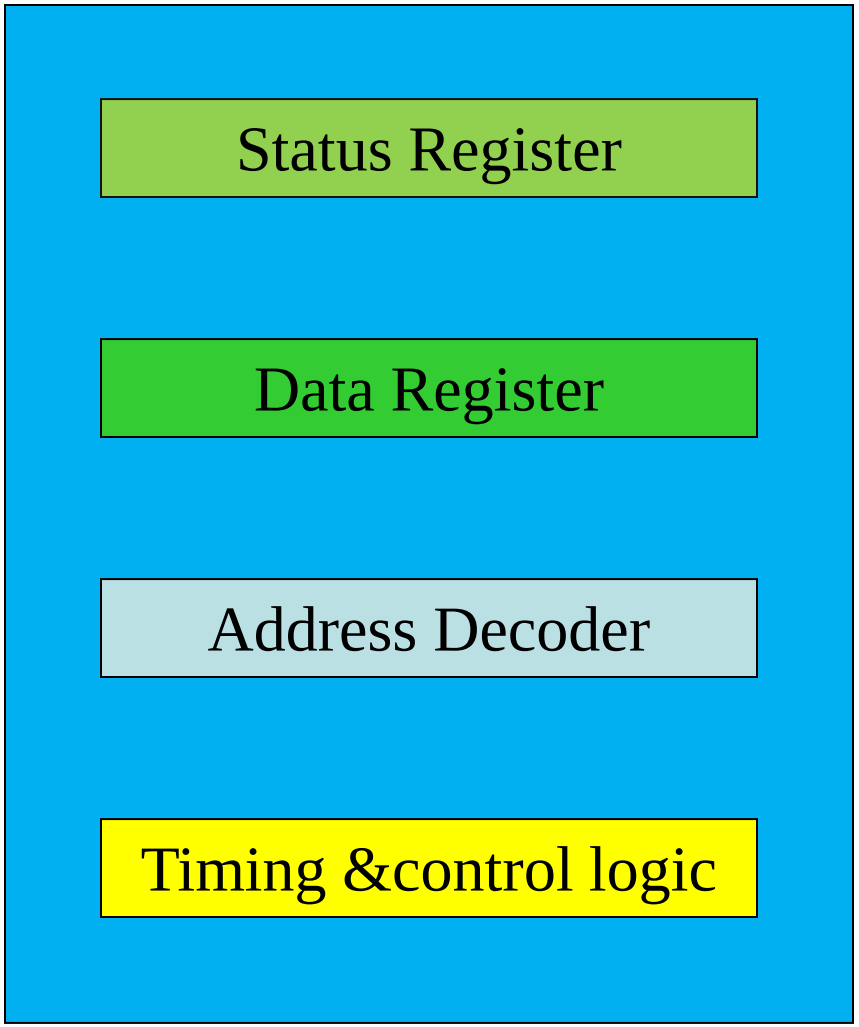
4. **Data conditioning**

   The CPU handles digital data in parallel form, where the device may handle digital or analog data parallely or serially. Therefore the IO interface should carry out necessary data conversions.

5. **Controlling**

   The IO interface should provide the necessary control signals to the CPU and to the IO device, to carry out the data transfer successfully.

A Typical IO Interface



Address decoder

Decodes the address provided by the CPU and selects the required register within the interface for a READ or WRITE operation.

Data register

Provides a temporary store for the data being transferred.

Status register

This holds the status information of the IO device

Ex. Ready or Busy condition of a printer.

Timing & control logic

This generates all the required control signals to carry out the data transfer and does the data conditioning.

IO Addressing Methods

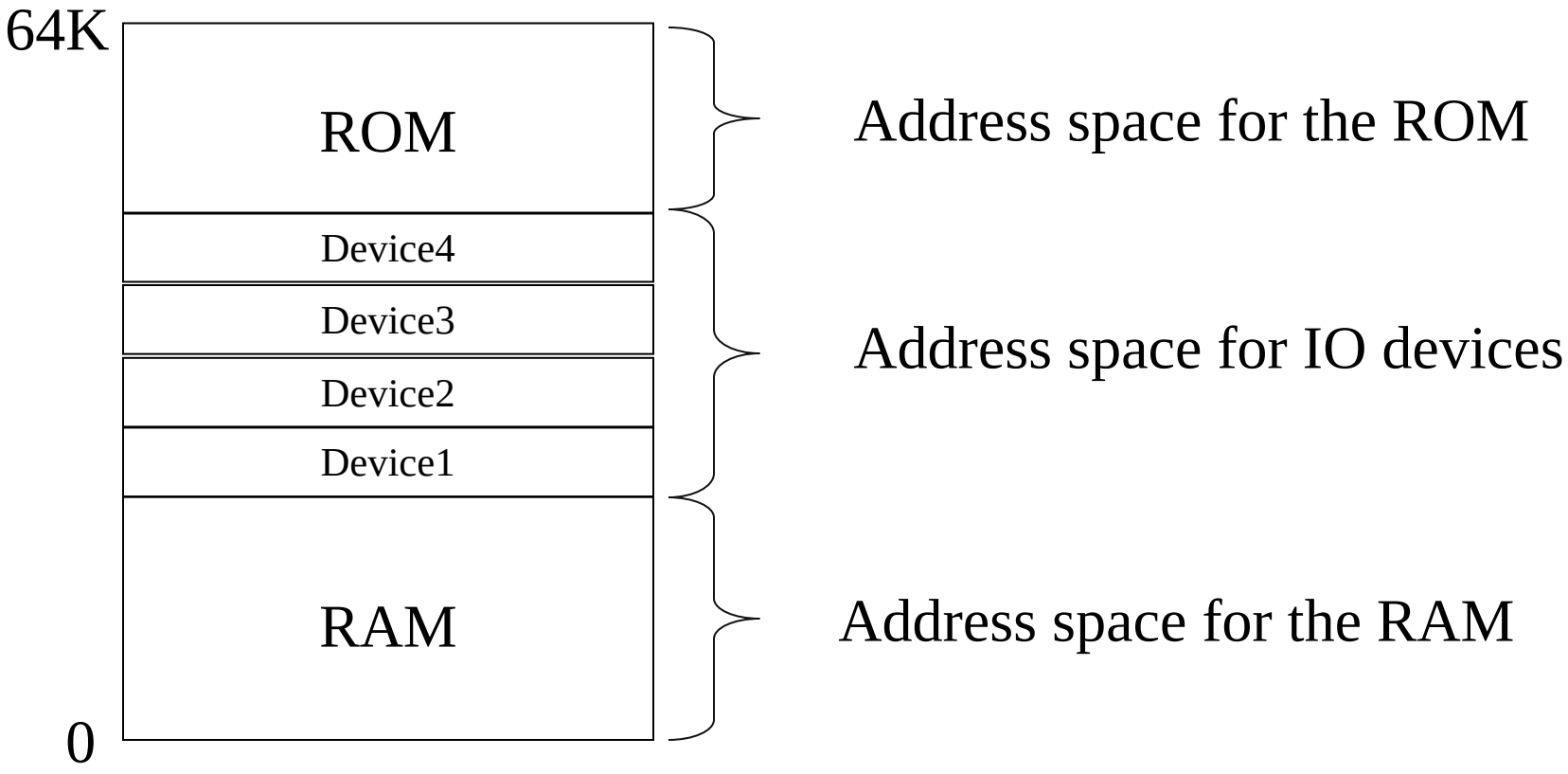
There are two methods of addressing IO devices.

1. Memory mapping/memory mapped IO
2. IO mapping/IO mapped IO

Memory mapping

In this method both IO devices and memory are assigned addresses in the same address space.

Therefore data transfer between CPU and IO devices takes place similar to the memory READ and WRITE operations. The machine instructions for IO data transfer could be the same, those used for memory READ & WRITE.

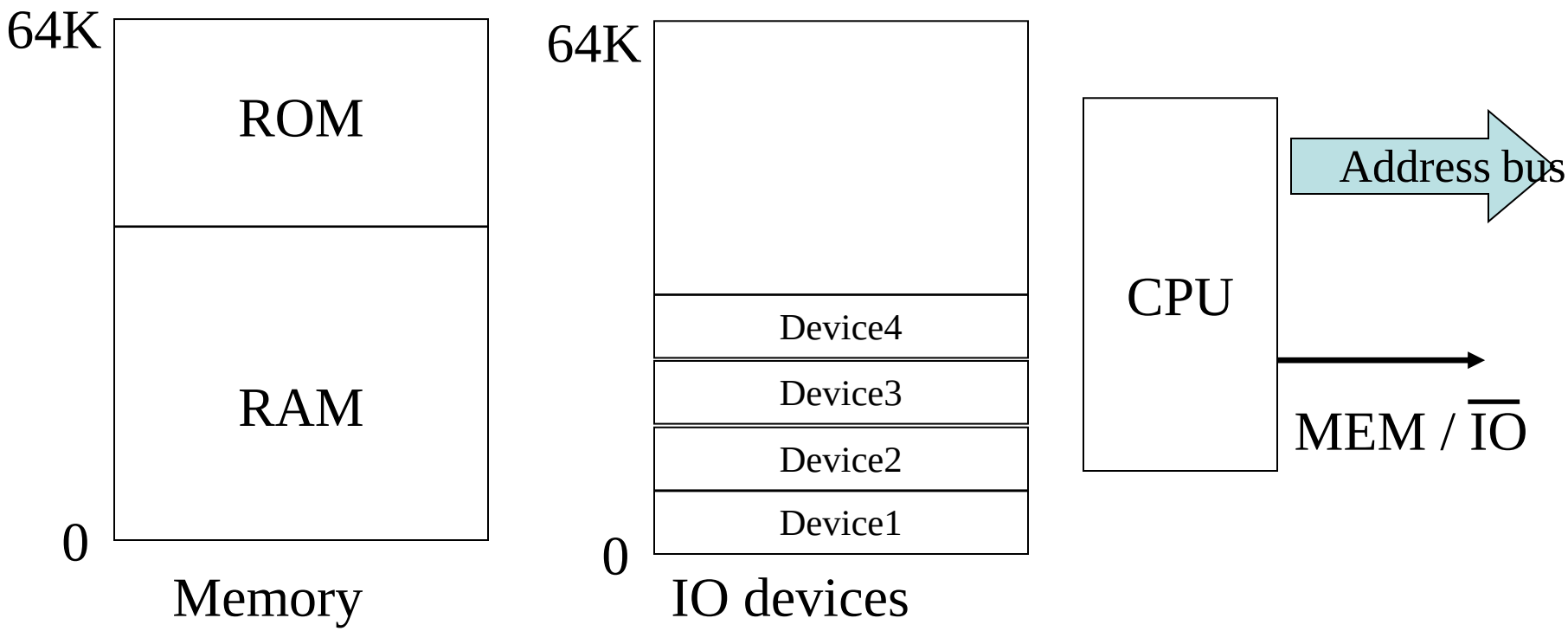


Consider a CPU with a 16 bit address bus. The available address space will be about 2^{16} or 64K locations. Part of that is allocated for IO devices and ROM which holds the BIOS program.

The disadvantage of memory mapping is that, part of the available address space should be set aside for IO devices.

IO mapping

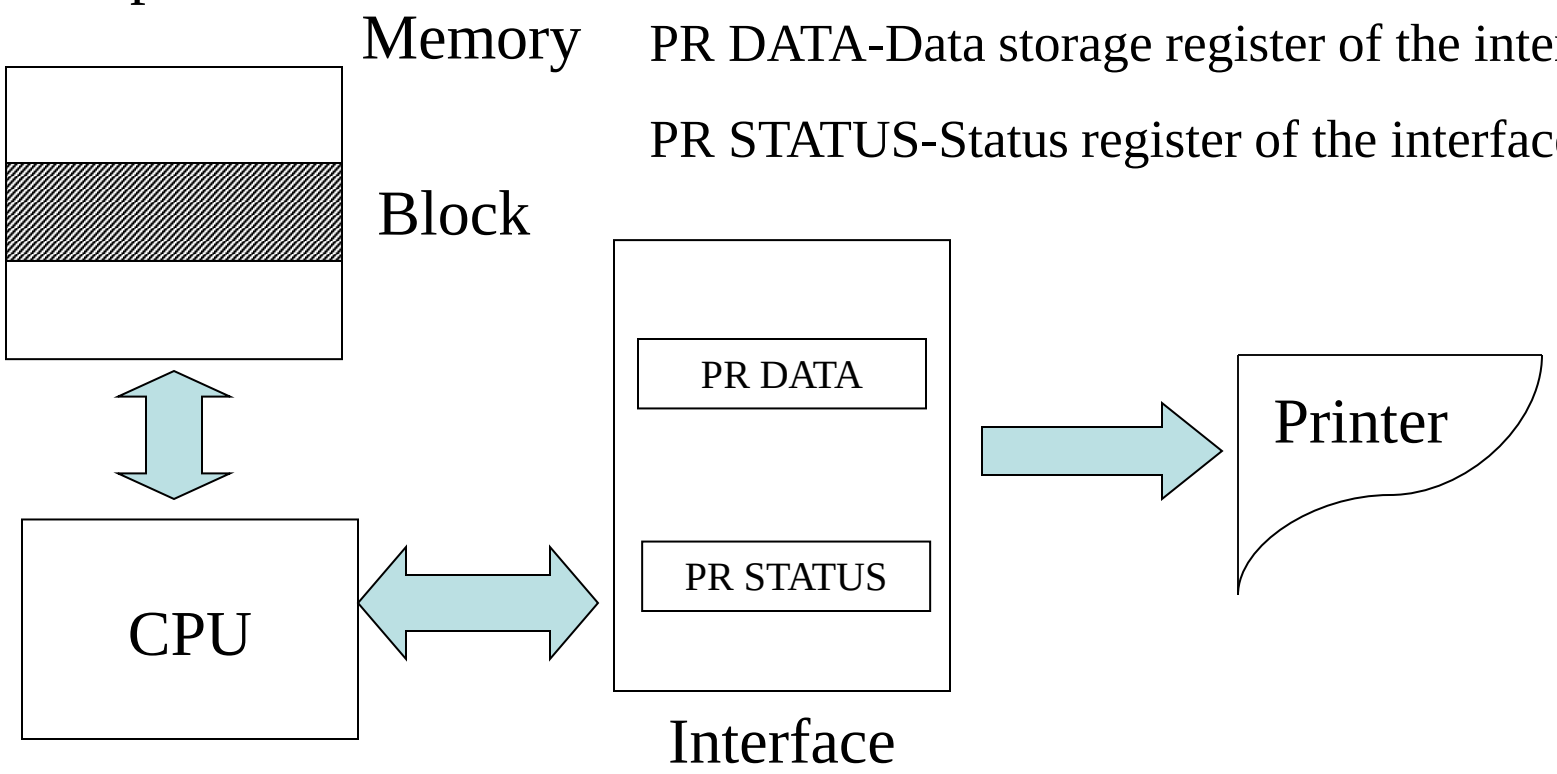
In this method CPU has a special control line to indicate whether it is addressing the memory or IO devices. Separate machine code instructions are available for the two classes of data transfer, memory & IO.



IO Data Transfer Control Methods

Program controlled IO

The data transfer is implemented by a program executed by the CPU. Consider a block of data transfer between a main memory and a printer.



PR DATA keeps the data available for the printer for a sufficient time so that the printer could read it.

PR STATUS keeps the printer status so that the CPU could read it whenever required. These two registers are assigned to two different addresses.

The disadvantage of program controlled IO is that the CPU has to waste a considerable time just to check the status of the device.

Interrupt controlled IO

When the CPU initializes an IO operation to a slow device the CPU has to wait a considerable time on busy wait. This affects the deficiency of the CPU specially in a multitasking environment.

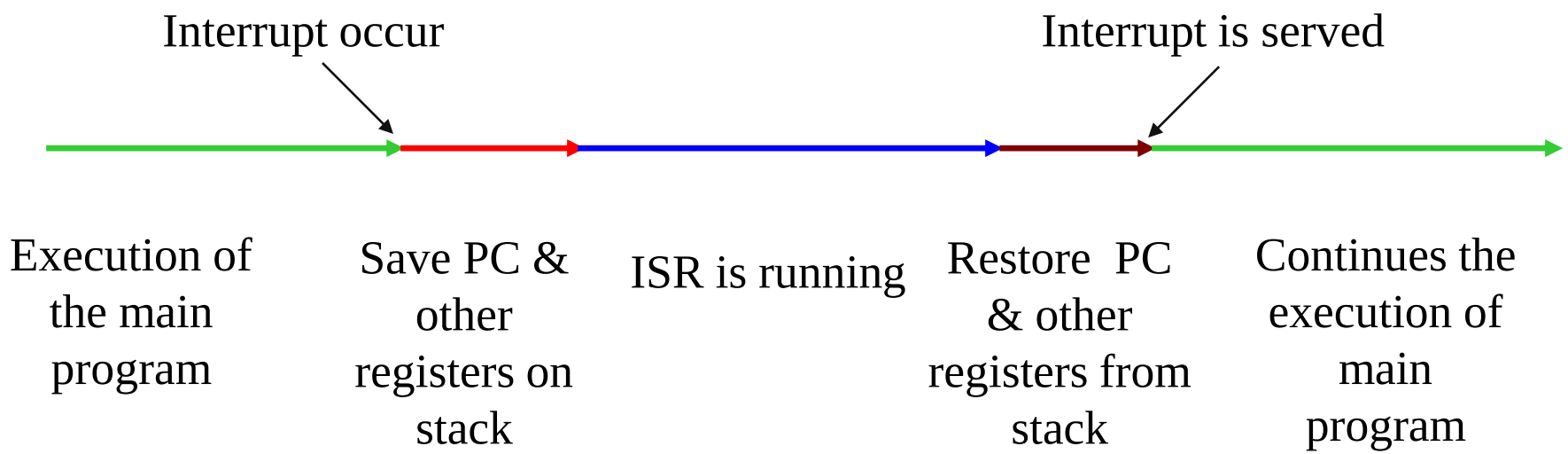
As a solution to this problem, the CPU can initiate the IO operation and thereafter proceed with the remaining tasks without waiting for the completion of the IO operation. The device whenever ready to proceed with IO operation, will interrupt the CPU.

However this requires a dedicated hardware control wire called the Interrupt line

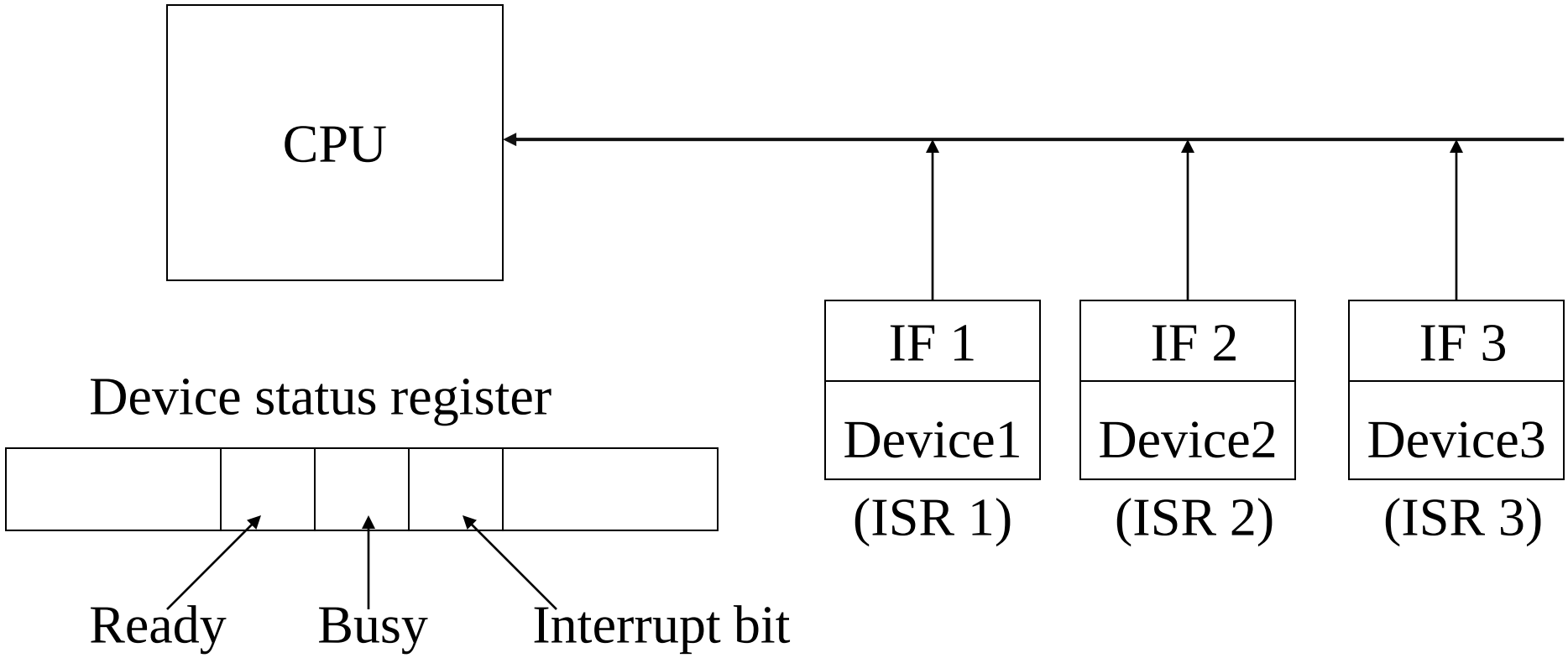
The interrupt line can also be used to initiate an IO operation by an IO device.

When CPU receives an interrupt

1. The current program that is being executed is suspended, after completing the current instruction.
2. The control is transferred to an interrupt service routine (ISR) (ISR is a program that serves the IO device. ISR may be stored at a fixed location in memory).
3. After the interrupt is serviced, the control is transferred to the suspended program.



Multiple IO devices can be connected to the same interrupt line



Status register of each device has a reserved bit to indicate whether it has generated the interrupt or not. CPU polls this bit of each device to identify the device which has generated the interrupt.

Different Interrupt Types in Computer Systems

There are basically three classes of interrupts in a computer system

1. Internal interrupts (Traps)
2. Software interrupts (System Calls)
3. External interrupts(Hardware Interrupts)

1. Internal Interrupts (Traps)

Traps are generated within the CPU, as a result of internal processor event such as page faults, protection violations etc.

On the occurrence of a Trap, CPU saves the state of the current process (program) and transfers the control to a trap vector location in memory

2. Software Interrupts

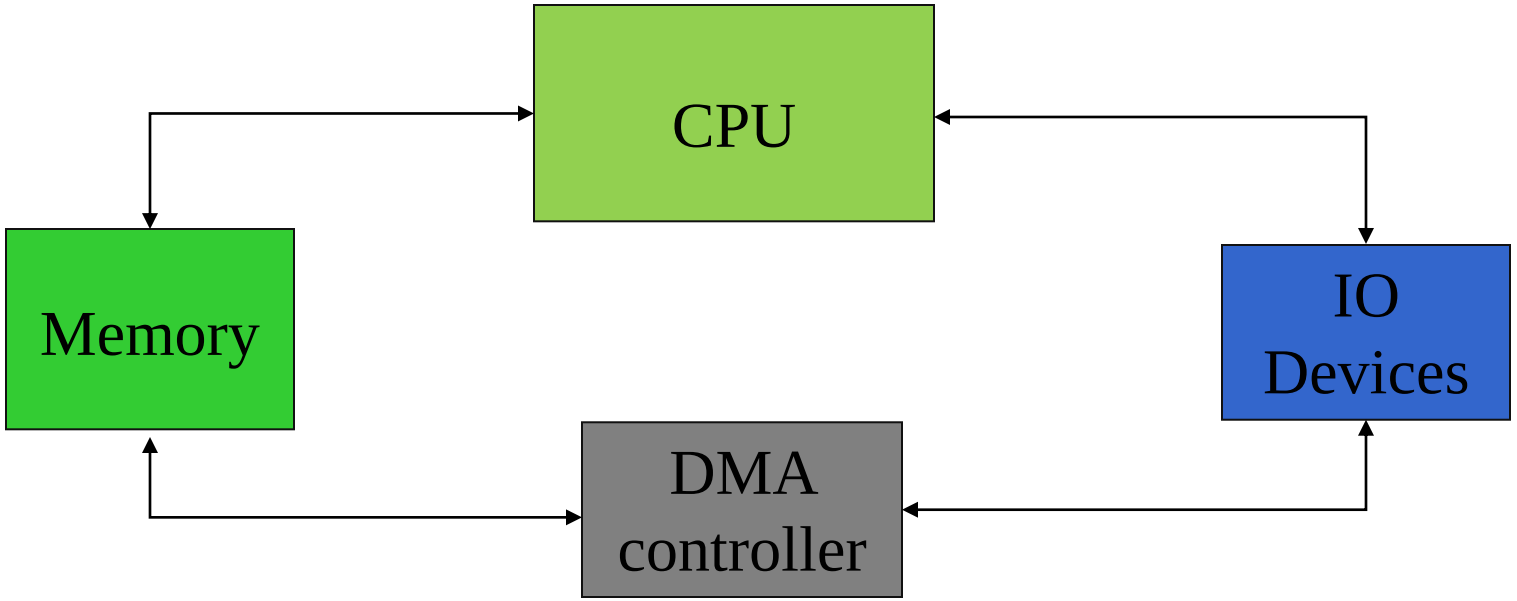
This affects the processor state, in much the same way as an internal interrupts. Software interrupts are generated by executing special instructions.

3. External Interrupts

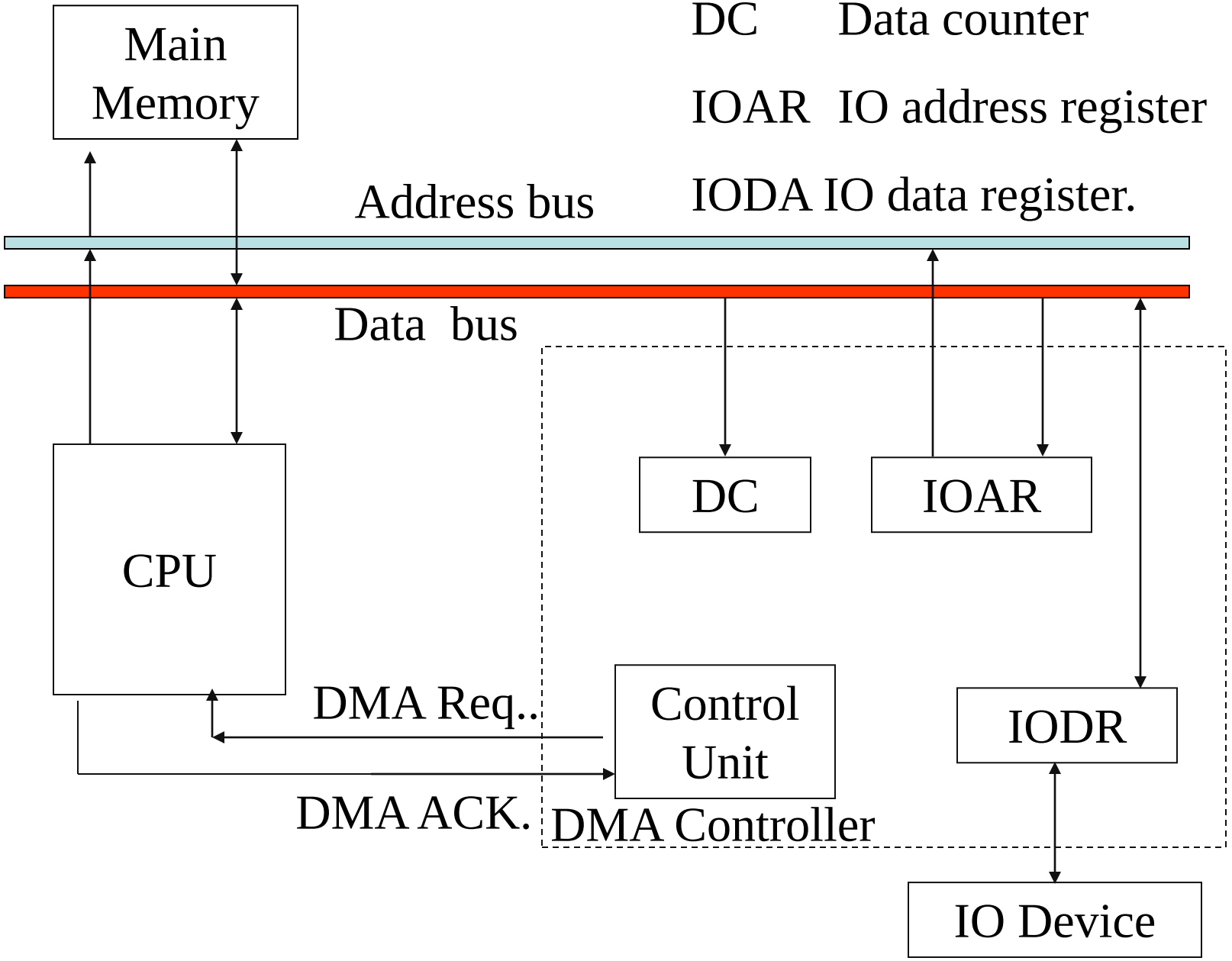
These are generated by external devices by activating an interrupt request line. Hardware interrupts can be either maskable interrupts (MI) or non maskable interrupts (NMI).

NMI's cannot be ignored by the CPU. NMI's are served as soon as they arrive at the CPU. However MI's can be disabled (or masked) by setting an interrupt mask flag or resetting an interrupt enable flag.

Direct Memory Access (DMA)



DMA involves transfer of data, between an IO device and main memory, bypassing the CPU. Without the use of DMA, data transfer from memory to an IO device (or vice versa) has to take place through the CPU. It is an inefficient method specially when large blocks of data is to be transferred.


Main Memory
DC Data counter
IOAR IO address register
IODA IO data register.
Address bus
Data bus
CPU
DC
IOAR
DMA Req..
Control Unit
IODR
DMA ACK.
DMA Controller
IO Device

DMA block data transfer proceeds as follows for the above system.

1. CPU loads the DMA registers IOAR and DC of their initial values. IOAR is loaded with the starting address of the data block and DC with number of data bytes to be transferred.
2. When the DMA controller (DMAC) is ready it asserts DMA request line (Bus request line)
3. When CPU receives the DMA request it releases the system bus, and activates the DMA acknowledgement line (Bus ack)
4. DMA controller(DMAC) now initiates the data transfer directly from the main memory (or to the main memory). After transferring each byte, IOAR is incremented and DC is decremented.
5. Once DC is decremented to zero, the DMAC returns the bus control to the CPU, by deactivating DMA request line. CPU responds to this by deactivating the DMA acknowledgement line

DMA Data Transfer Modes

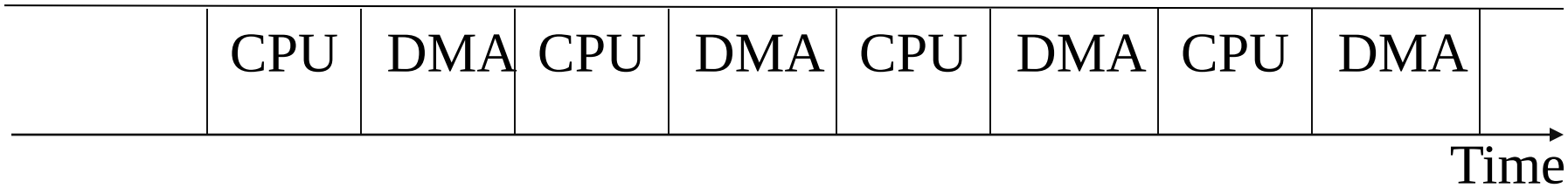
Block transfer DMA

The data word sequence of arbitrary length is transferred as a single contiguous burst while the DMA is the bus master. This mode of data transfer is used where the data transfer cannot be stopped or slowed down without a loss of data.

Ex. Reading from a magnetic disk, Writing to a CD-R etc.

Cycle Stealing DMA

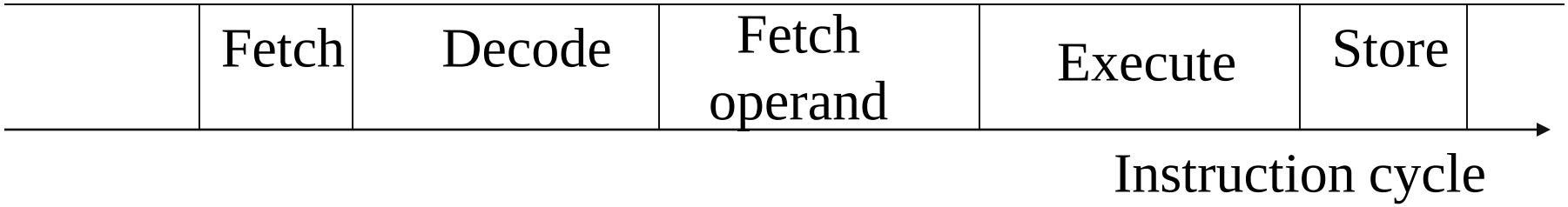
CPU and DMA bus transactions are interleaved, therefore large blocks of IO data is transferred by a sequence of DMA bus transactions, inter spaced with CPU bus transactions.



Transparent DMA

In this mode, bus cycles are stolen only when CPU is not actually using the system bus. This completely eliminates the DMA interference to the CPU activity.

Ex.



During instruction decoding and execution of any arithmetic or logical operation, CPU does not need the system bus, and DMAC can be the bus master.

Rate of data transfer increases ↑

Block DMA

Cycle Stealing DMA

Transparent DMA

↓ Interference to the CPU decreasing