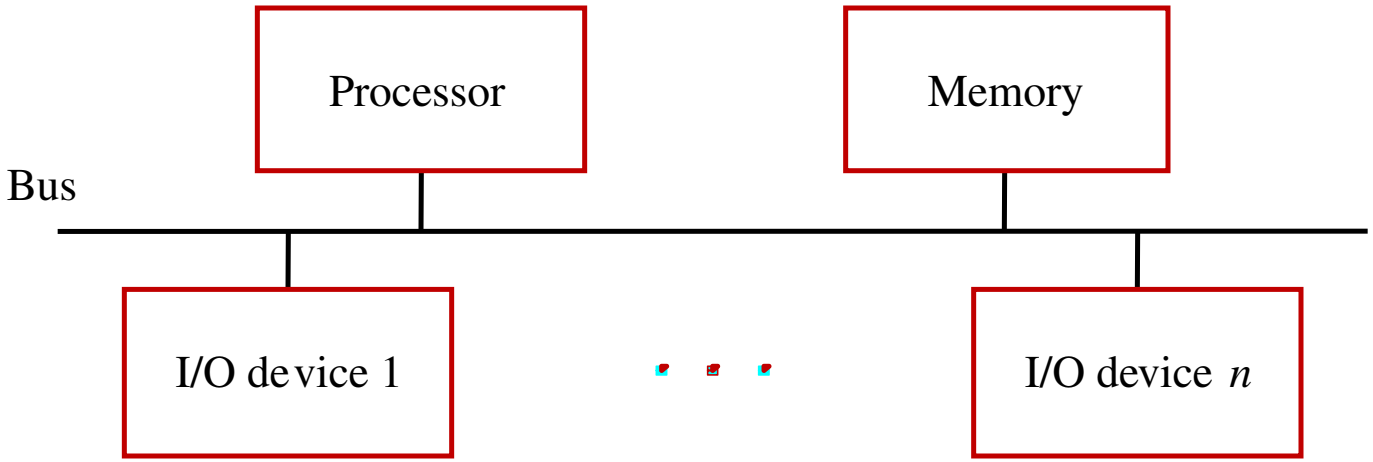
INPUT/OUTPUT ORGANIZATION

MODULE 4

Accessing I/O devices



- *Multiple I/O devices may be connected to the processor and the memory via a bus.*
- *Bus consists of three sets of lines to carry address, data and control signals.*
- *Each I/O device is assigned an unique address.*
- *To access an I/O device, the processor places the address on the address lines.*
- *The device recognizes the address, and responds to the control signals.*

Accessing I/O devices (contd..)

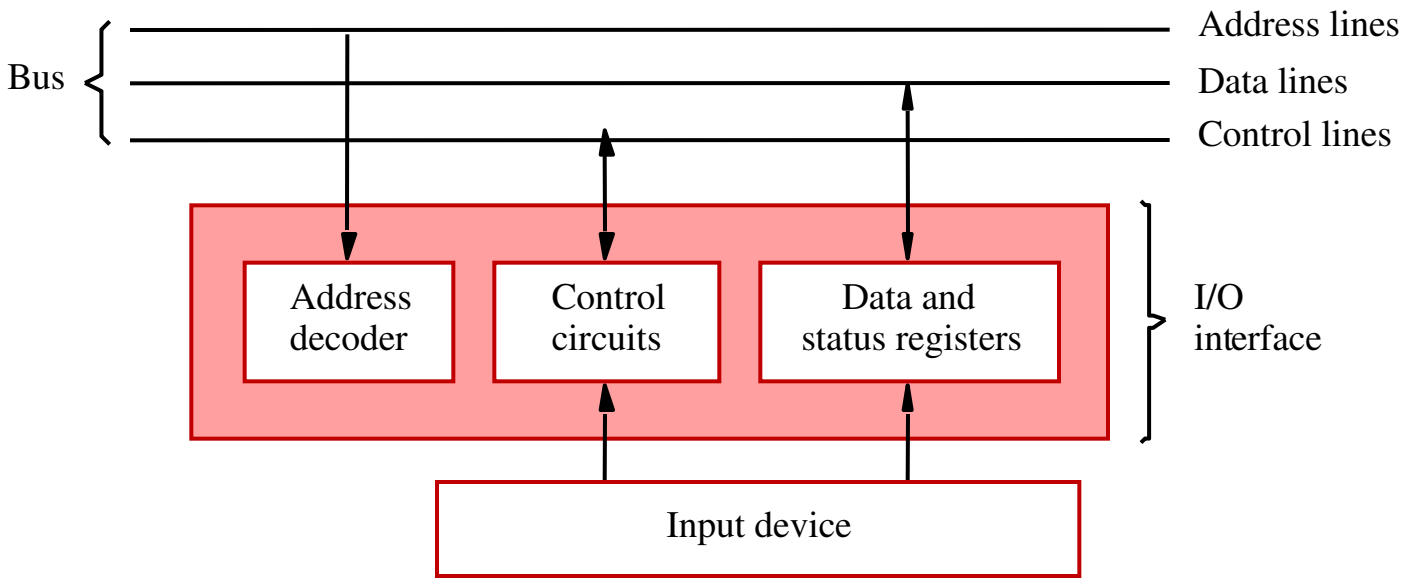
I/O devices and the memory may share the same address space:

- Memory-mapped I/O.
- Any machine instruction that can access memory can be used to transfer data to or from an I/O device.
- Simpler software.

I/O devices and the memory may have different address spaces:

- Special instructions to transfer data to and from I/O devices.
- I/O devices may have to deal with fewer address lines.
- I/O address lines need not be physically separate from memory address lines.
- In fact, address lines may be shared between I/O devices and memory, with a control signal to indicate whether it is a memory address or an I/O address.

Accessing I/O devices (contd..)



- *I/O device is connected to the bus using an I/O interface circuit which has:*
  - *Address decoder, control circuit, and data and status registers.*
- *Address decoder decodes the address placed on the address lines thus enabling device to recognize its address.*
- *Data register holds the data being transferred to or from the processor.*
- *Status register holds information necessary for the operation of the I/O device.*
- *Data and status registers are connected to the data lines, and have unique addres*
- *I/O interface circuit coordinates I/O transfers.*

Accessing I/O devices (contd..)

Recall that the rate of transfer to and from I/O devices is slower than the speed of the processor. This creates the need for mechanisms to synchronize data transfers between them.

Program-controlled I/O:

- Processor repeatedly monitors a status flag to achieve the necessary synchronization.
- Processor polls the I/O device.

Two other mechanisms used for synchronizing data transfers between the processor and memory:

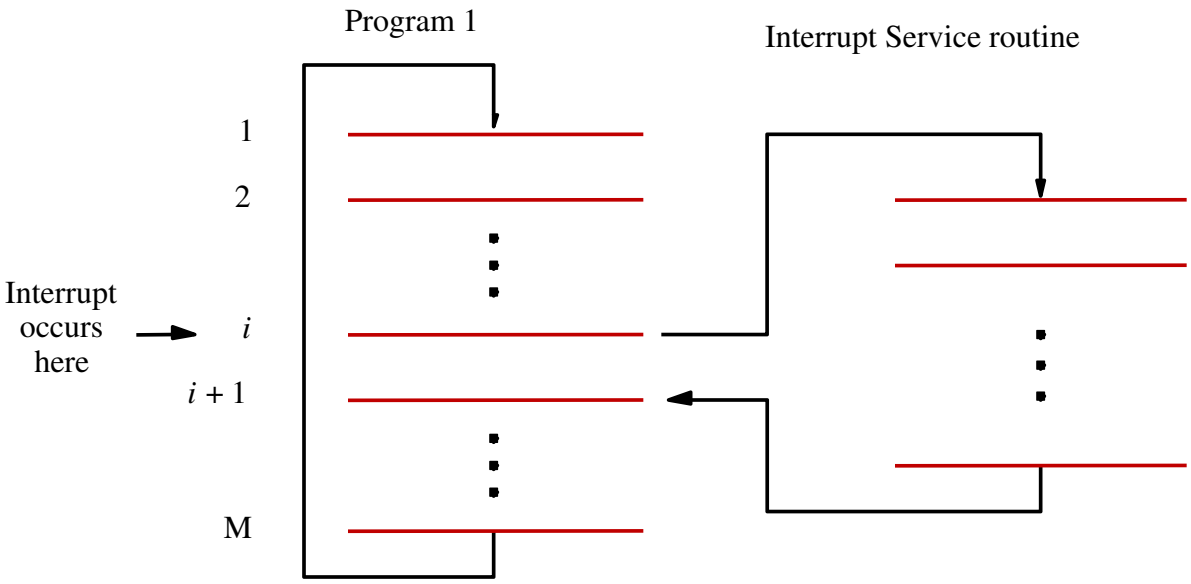
- Interrupts.
- Direct Memory Access.

Interrupts

Interrupts

- In program-controlled I/O, when the processor continuously monitors the status of the device, it does not perform any useful tasks.
- An alternate approach would be for the I/O device to alert the processor when it becomes ready.
  - Do so by sending a hardware signal called an interrupt to the processor.
  - At least one of the bus control lines, called an interrupt-request line is dedicated for this purpose.

Interrupts (contd..)



- *Processor is executing the instruction located at address i when an interrupt occurs.*
- *Routine executed in response to an interrupt request is called the interrupt-service routine.*
- *When an interrupt occurs, control must be transferred to the interrupt service routine.*
- *But before transferring control, the current contents of the PC (i+1), must be saved in a known location.*
- *This will enable the return-from-interrupt instruction to resume execution at i+1.*
- *Return address, or the contents of the PC are usually stored on the processor stack.*

Interrupts (contd..)

Treatment of an interrupt-service routine is very similar to that of a subroutine.

However there are significant differences:

- A subroutine performs a task that is required by the calling program.
- Interrupt-service routine may not have anything in common with the program it interrupts.
- Interrupt-service routine and the program that it interrupts may belong to different users.
- As a result, before branching to the interrupt-service routine, not only the PC, but other information such as condition code flags, and processor registers used by both the interrupted program and the interrupt service routine must be stored.
- This will enable the interrupted program to resume execution upon return from interrupt service routine.

Interrupts (contd..)

- Saving and restoring information can be done automatically by the processor or explicitly by program instructions.
- Saving and restoring registers involves memory transfers:
  - Increases the total execution time.
  - Increases the delay between the time an interrupt request is received, and the start of execution of the interrupt-service routine. This delay is called interrupt latency.
- In order to reduce the interrupt latency, most processors save only the minimal amount of information:
  - This minimal amount of information includes Program Counter and processor status registers.
- Any additional information that must be saved, must be saved explicitly by the program instructions at the beginning of the interrupt service routine.

Interrupts (contd..)

- When a processor receives an interrupt-request, it must branch to the interrupt service routine.
- It must also inform the device that it has recognized the interrupt request.
- This can be accomplished in two ways:
  - Some processors have an explicit interrupt-acknowledge control signal for this purpose.
  - In other cases, the data transfer that takes place between the device and the processor can be used to inform the device.

Interrupts (contd..)

Interrupt-requests interrupt the execution of a program, and may alter the intended sequence of events:

- Sometimes such alterations may be undesirable, and must not be allowed.
- For example, the processor may not want to be interrupted by the same device while executing its interrupt-service routine.

Processors generally provide the ability to enable and disable such interruptions as desired.

One simple way is to provide machine instructions such as *Interrupt-enable* and *Interrupt-disable* for this purpose.

Interrupts (contd..)

To avoid interruption by the same device during the execution of an interrupt service routine:

1. First instruction of an interrupt service routine can be Interrupt-disable. Last instruction of an interrupt service routine can be Interrupt-enable.
2. Interrupt Mask in status register
3. Interrupt request line edge triggered

Vectored Interrupts

- The device requesting an interrupt may identify itself directly to the processor.
  - Device can do so by sending a special code (4 to 8 bits) the processor over the bus.
  - Code supplied by the device may represent a part of the starting address of the interrupt-service routine.
  - The remainder of the starting address is obtained by the processor based on other information such as the range of memory addresses where interrupt service routines are located.

Vectored Interrupts

- Usually the location pointed to by the interrupting device is used to store the starting address of the interrupt-service routine.
- Contents of this location which constitutes a new value for the PC ie address of a memory location that contains the required starting address called Interrupt vector.
- The processor may not respond immediately.
- When it is ready it gives a interrupt acknowledge signal to interface which places the interrupt vector code on the data lines of bus and causes turn off of the INTR signal

Interrupts (contd..)

Previously, before the processor started executing the interrupt service routine for a device, it disabled the interrupts from the device.

In general, same arrangement is used when multiple devices can send interrupt requests to the processor.

- During the execution of an interrupt service routine of device, the processor does not accept interrupt requests from any other device.
- Since the interrupt service routines are usually short, the delay that this causes is generally acceptable.

However, for certain devices this delay may not be acceptable.

- Which devices can be allowed to interrupt a processor when it is executing an interrupt service routine of another device?

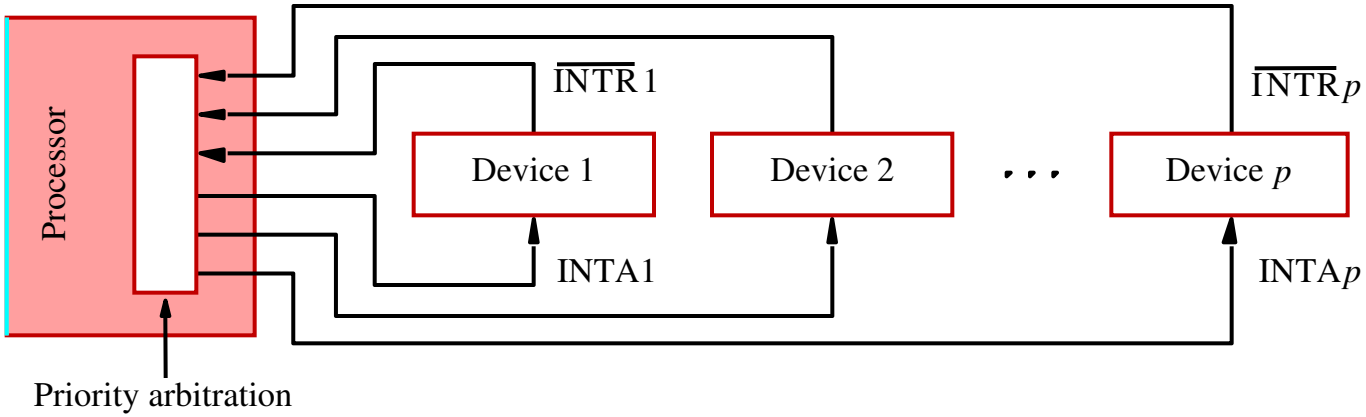
Interrupts (contd..)

- I/O devices are organized in a priority structure:
  - An interrupt request from a high-priority device is accepted while the processor is executing the interrupt service routine of a low priority device.
- A priority level is assigned to a processor that can be changed under program control.
  - Priority level of a processor is the priority of the program that is currently being executed.
  - When the processor starts executing the interrupt service routine of a device, its priority is raised to that of the device.
  - If the device sending an interrupt request has a higher priority than the processor, the processor accepts the interrupt request.

Interrupts (contd..)

- Processor's priority is encoded in a few bits of the processor status register.
  - Priority can be changed by instructions that write into the processor status register.
  - Usually, these are privileged instructions, or instructions that can be executed only in the supervisor mode.
  - Privileged instructions cannot be executed in the user mode.
  - Prevents a user program from accidentally or intentionally changing the priority of the processor.
- If there is an attempt to execute a privileged instruction in the user mode, it causes a special type of interrupt called as privilege exception.

Interrupts (contd..)



- *Each device has a separate interrupt-request and interrupt-acknowledge line.*
- *Each interrupt-request line is assigned a different priority level.*
- *Interrupt requests received over these lines are sent to a priority arbitration circuit in the processor.*
- *If the interrupt request has a higher priority level than the priority of the processor, then the request is accepted.*

Interrupts (contd..)

Which interrupt request does the processor accept if it receives interrupt requests from two or more devices simultaneously?.

If the I/O devices are organized in a priority structure, the processor accepts the interrupt request from a device with higher priority.

- Each device has its own interrupt request and interrupt acknowledge line.
- A different priority level is assigned to the interrupt request line of each device.

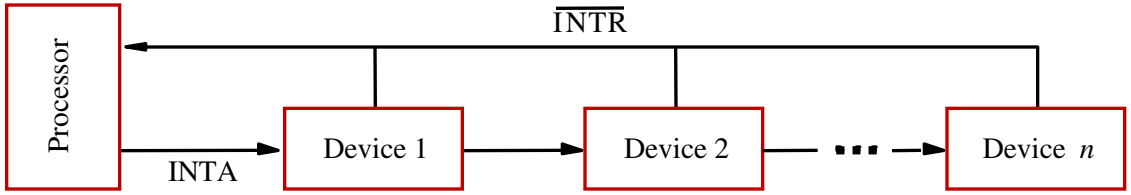
However, if the devices share an interrupt request line, then how does the processor decide which interrupt request to accept?

Interrupts (contd..)

Polling scheme:

- *If the processor uses a polling mechanism to poll the status registers of I/O device to determine which device is requesting an interrupt.*
- *In this case the priority is determined by the order in which the devices are polled*
- *The first device with status bit set to 1 is the device whose interrupt request is accepted.*

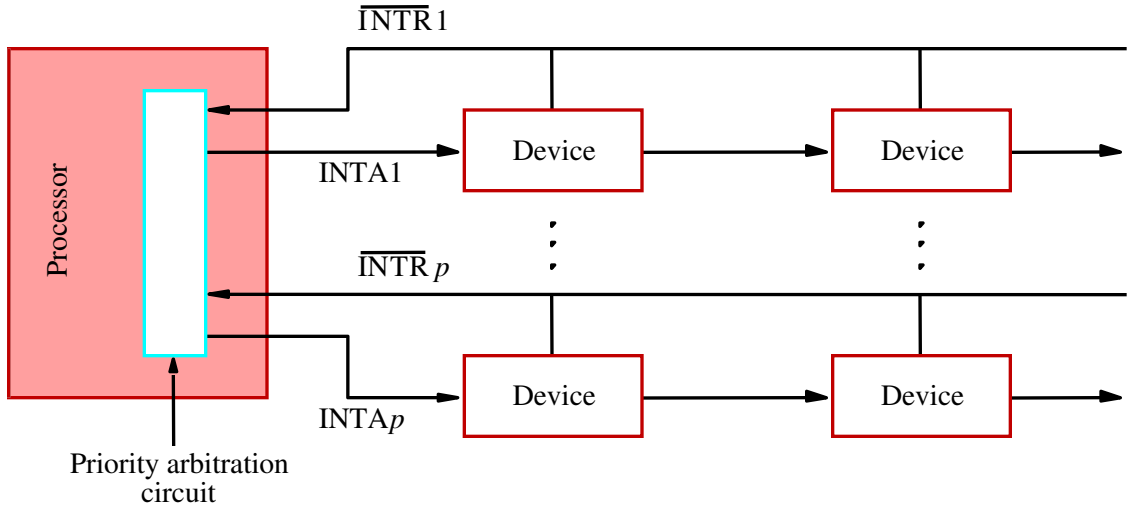
Daisy chain scheme:



- *Devices are connected to form a daisy chain.*
- *Devices share the interrupt-request line, and interrupt-acknowledge line is connec to form a daisy chain.*
- *When devices raise an interrupt request, the interrupt-request line is activated.*
- *The processor in response activates interrupt-acknowledge.*
- *Received by device 1, if device 1 does not need service, it passes the signal to d*
- *Device that is electrically closest to the processor has the highest priority.*

Interrupts (contd..)

- *When I/O devices were organized into a priority structure, each device had its own interrupt-request and interrupt-acknowledge line.*
- *When I/O devices were organized in a daisy chain fashion, the devices shared an interrupt-request line, and the interrupt-acknowledge propagated through the devic*
- *A combination of priority structure and daisy chain scheme can also used.*



- *Devices are organized into groups.*
- *Each group is assigned a different priority level.*
- *All the devices within a single group share an interrupt-request line, and are connected to form a daisy chain.*

Interrupts (contd..)

Only those devices that are being used in a program should be allowed to generate interrupt requests.

To control which devices are allowed to generate interrupt requests, the interface circuit of each I/O device has an interrupt-enable bit.

> If the interrupt-enable bit in the device interface is set to 1, then the device is allowed to generate an interrupt-request.

Interrupt-enable bit in the device's interface circuit determines whether the device is allowed to generate an interrupt request.

Interrupt-enable bit in the processor status register or the priority structure of the interrupts determines whether a given interrupt will be accepted.

Direct Memory Access

Direct Memory Access

Direct Memory Access (DMA):

A special control unit may be provided to transfer a block of data in high speed directly between an I/O device and the main memory, without continuous intervention by the processor.

Control unit which performs these transfers is a part of the I/O device's interface circuit. This control unit is called as a DMA controller.

DMA controller performs functions that would be normally carried out by the processor:

For each word, it provides the memory address and all the control signals.

To transfer a block of data, it increments the memory addresses and keeps track of the number of transfers.

Direct Memory Access (contd..)

DMA controller can transfer a block of data from an external device to the processor, without any intervention from the processor.

However, the operation of the DMA controller must be under the control of a program executed by the processor. That is, the processor must initiate the DMA transfer.

To initiate the DMA transfer, the processor informs the DMA controller of:

Starting address,
Number of words in the block.
Direction of transfer (I/O device to the memory, or memory to the I/O device).

Once the DMA controller completes the DMA transfer, it informs the processor by raising an interrupt signal.

- While DMA transfer is taking place, the program that requested the transfer cannot continue, and the processor can be used to execute another program.
- After the DMA transfer is completed, the processor can return to the program that requested the transfer.
- I/O operations are always performed by the operating system of the computer in response to a request from an application program.
- The OS is also responsible for suspending the execution of one program to blocked state, initiates the DMA operation and starts the execution of another.

- When the transfer is completed, DMA controller informs the processor by sending an interrupt request.
- In response OS puts the suspended program to runnable state.