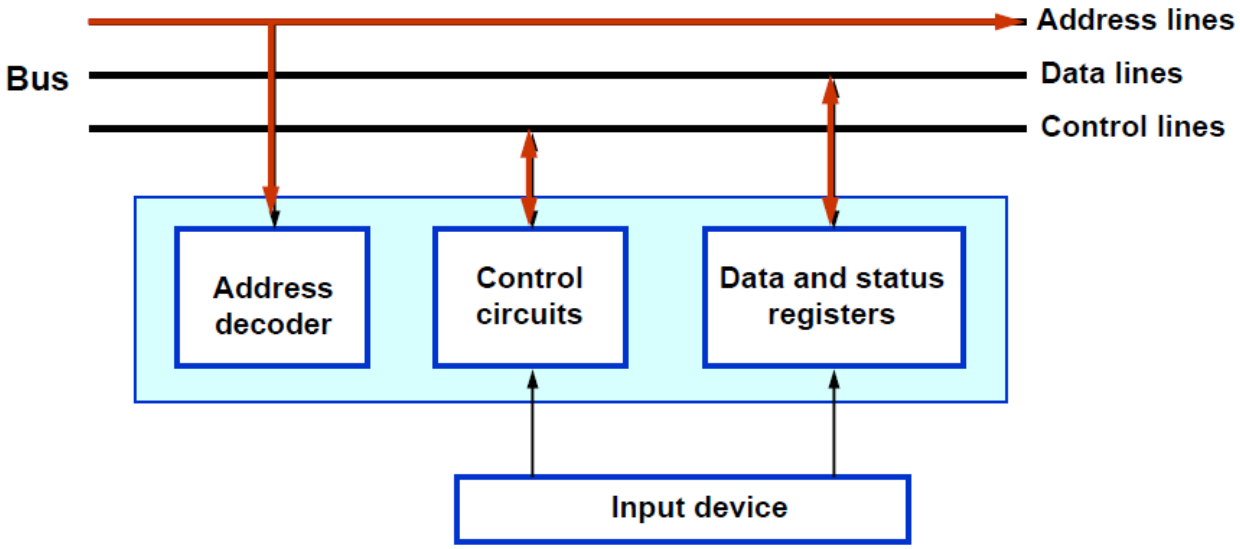
Part(5)

Input/output system(IO)

IO Interface for an Input Device

The address decoder, the data and status registers, and the control circuitry required to coordinate IO transfers constitute the device's interface circuit



address decoder : is used for IO device identification.

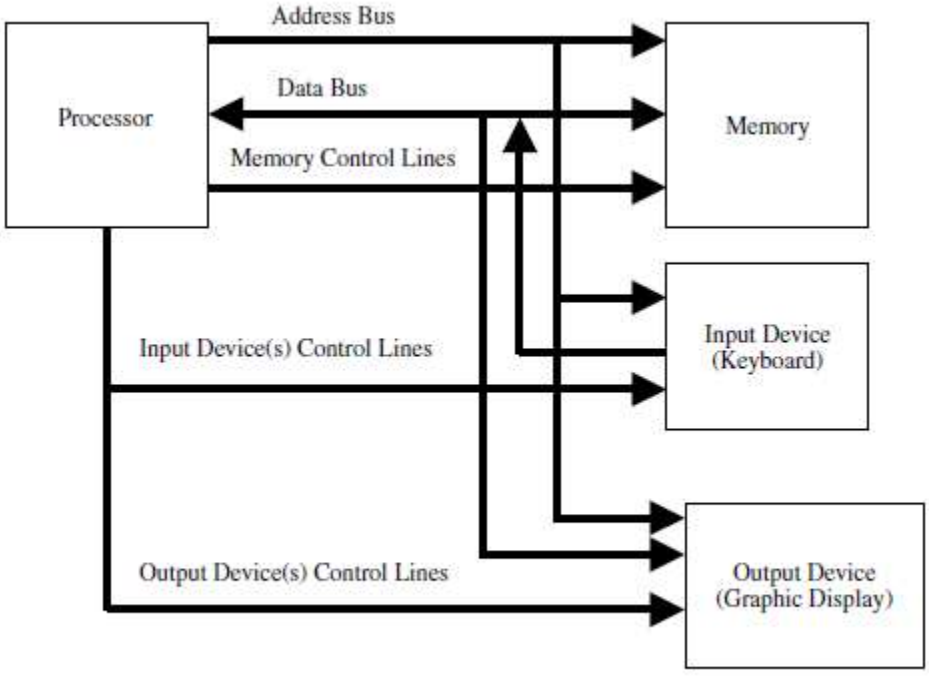
status registers: It is used to determine the status for each IO device, whether it is ready to transfer data to the processor

There are two methods in which the processor can **address** the input/output devices:

1-Shared IO:

• IO devices are assigned particular addresses, isolated from the address space assigned to the memory.
• An **input** and **output** instructions are used to input and output data.
• The address and data lines from the CPU can be shared between the memory and the IO devices.
• A separate control line will have to be used. This is because of the need for executing input and output instructions.

•The main **advantage** of the shared IO arrangement is the *separation* between the memory address space and that of the IO devices. Its main **disadvantage** is the need to have special input and output instructions in the processor instruction set.

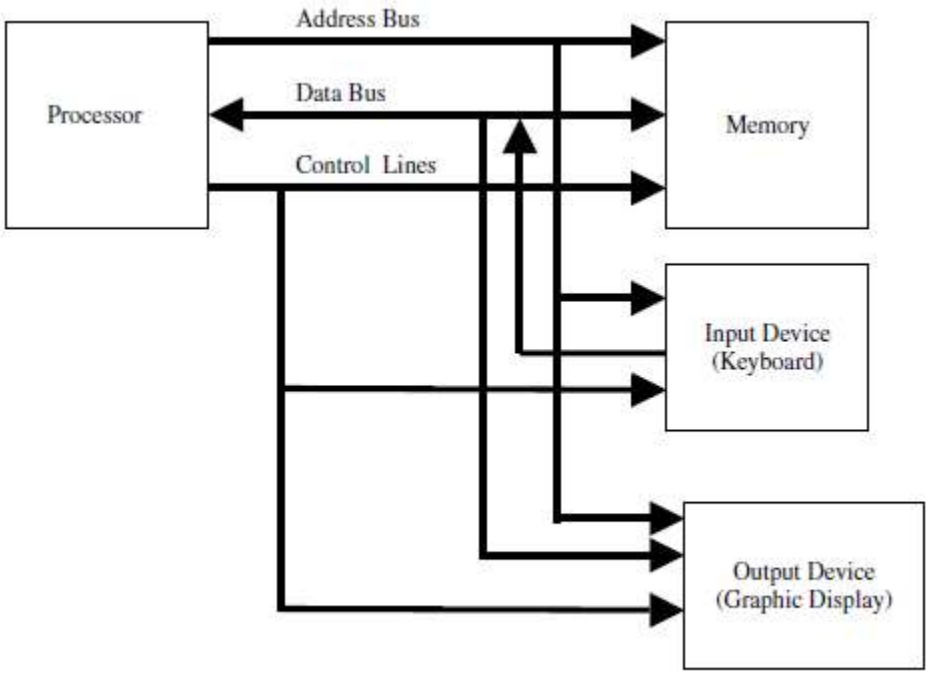


Shared I/O arrangement

2-memory-mapped IO:

• It deal with input and output registers as if they are regular memory locations
•It uses **Read** and **Write** (memory instruction).
• The main **advantage** of the memory-mapped IO is the use of the read and write instructions of the processor to perform the input and output operations, respectively. It eliminates the need for introducing special IO instructions.

• The main **disadvantage** of the memory-mapped IO is the need to reserve a certain part of the memory address space for addressing IO devices, that is, a reduction in the available memory address space.



Memory-mapped I/O arrangement

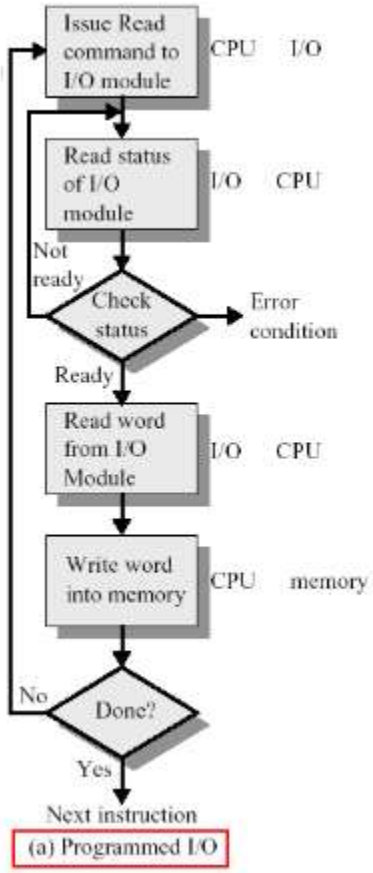
Data transfer to and from IO devices may be handled in one of the following techniques:

1-PROGRAMMED IO :

• The processor repeatedly checks a status flag to achieve the required synchronization between the processor and an input or output device.

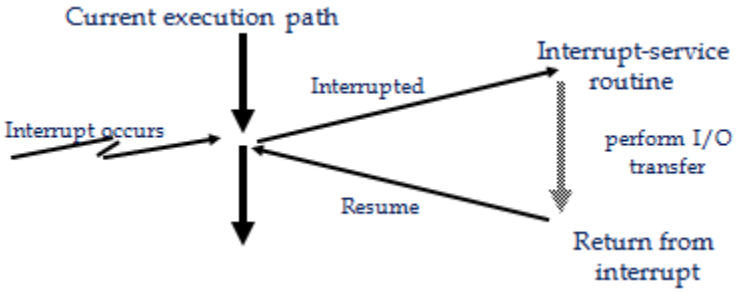
• CPU stays in the program loop until IO indicates it is ready for data transfer

• It is time consuming process since it keeps processor busy needlessly.



2- Interrupt:

• When an IO device is ready to send (receive) data to (from) the CPU, it **interrupt**s the CPU for its attention.
• No need to poll device status.
• As soon as the CPU finishes the current instruction, it transfers its execution to an interrupt-service routine(**ISR**) which responds to the external interrupt.



• Before executing the **ISR**, any information that may be altered during the execution of that routine must be saved. This information must be restored before the interrupted program is resumed.

There are two ways of choosing the branch address:
1- vector interrupt where the source that interrupt the CPU provides the branch address.

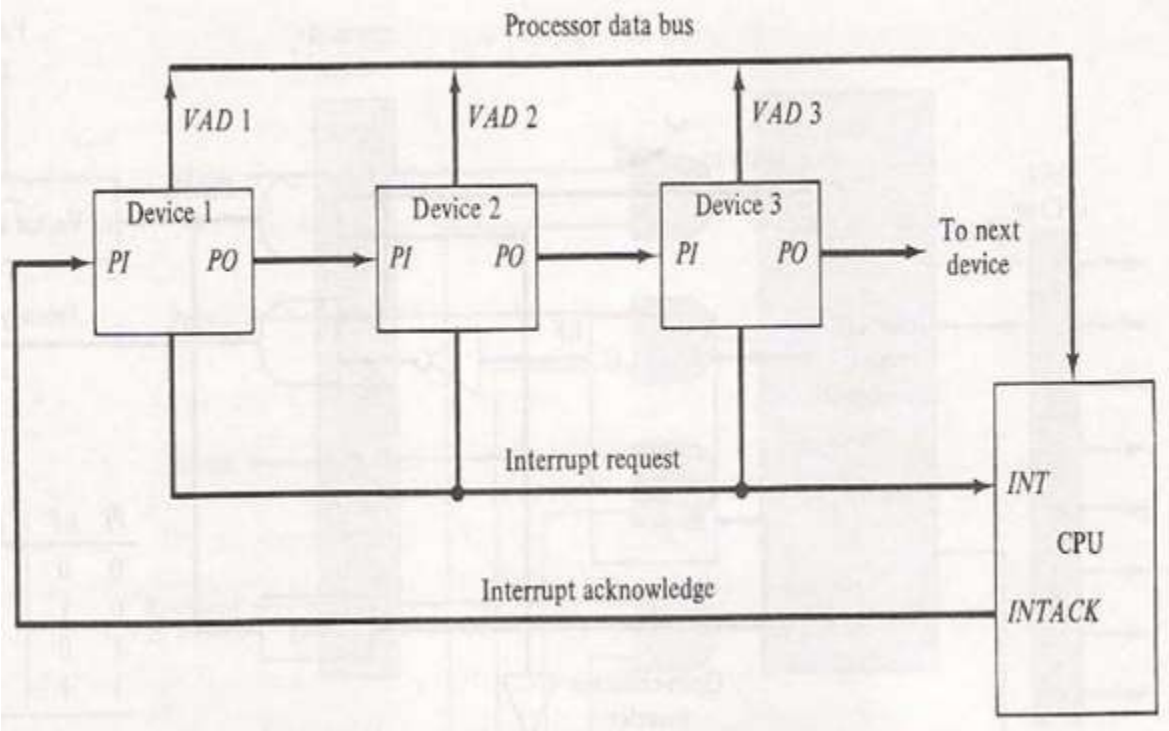
2- non vector interrupt where the branch address is assigned to a fixed address in the memory.

Priority interrupt:

System that assigned priority over the different IO devices when two or more devices are requested services at the same time. Devices with high transfer speed are given higher priority and slow devices given lower priority. There are two types of interrupt priority:

1-Polling: Used to identify the highest priority device with software means.

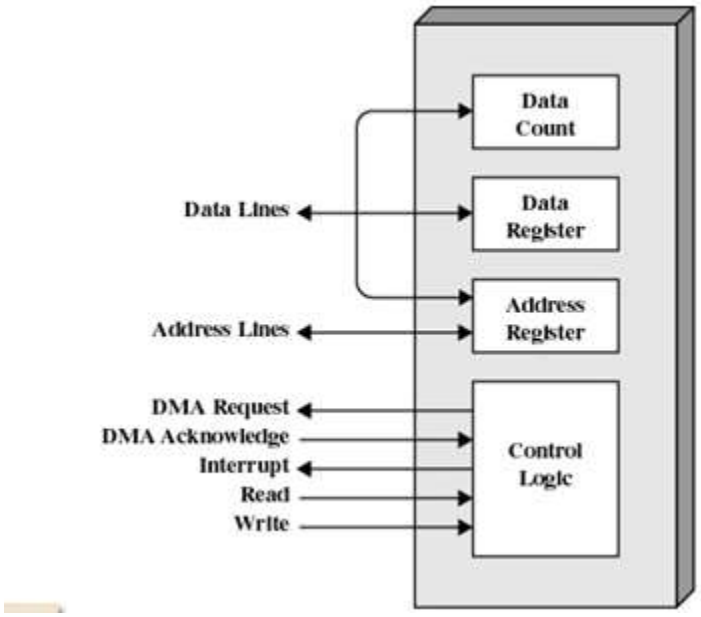
2-Daisy chain: Used to identify the highest priority device by hardware means. the device with the highest priority is placed first



- Device with highest priority is placed first.
- Device that wants the attention send the interrupt request to the CPU.
- CPU then sends the INTACK signal which is applied to PI(priority in) of the first device.
- If it had requested the attention, it place its VAD(vector address) on the bus. And it block the signal by placing 0 in PO(priority out)
- If not it pass the signal to next device through PO(priority out) by placing 1.
- This process is continued until appropriate device is found.
- The device whose PI is 1 and PO is 0 is the device that send the interrupt request.

3-DIRECT MEMORY ACCESS (DMA):

The DMA controller is a piece of hardware that controls one or more peripheral devices. It allows devices to transfer data to or from the system's memory without the help of the processor.
Having peripheral devices access memory directly would allow the CPU to do other work, which would lead to improved performance, especially in the cases of large transfers.



A DMA controller has an **address register**, a **Data count register**, and a **control logic**:

•**The address register**: contains an address that specifies the memory location of the data to be transferred. The DMA controller automatically increment the address register after each word transfer, so that the next transfer will be from the next memory location.

•**The Data count register**: Holds the number of words to be transferred. The word count is decremented by one after each word transfer.

•**The control logic**: Specifies the transfer mode (number of DMA channels they support).

The following steps summarize the **DMA operations**:
1. DMA request.
2. DMA accept.
3. Release buses.
4. DMA controller initiates data transfer.
5. Data is moved (increasing the address in memory, and reducing the count of words to be moved).
6. When word count reaches zero, the DMA informs the CPU of the termination by means of an interrupt.
7. The CPU regains access to the memory bus.

DMA transfer types

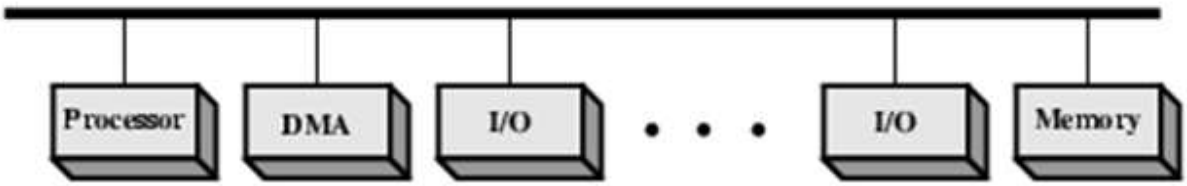
1- Burst mode: The DMA controller keeps control of the bus until all the data has been transferred to(from) memory from(to) the peripheral device. This mode of transfer is needed for fast devices where data transfer cannot be stopped until the entire transfer is done.

2- Single-cycle mode (**cycle stealing or flyby**): the DMA controller releasing the bus after each transfer of one data word. This minimizes the amount of time that the CPU is not controlling the bus. The single-cycle mode is preferred if IO devices can store very large amounts of data.

DMA configurations:

The DMA mechanism can be configured in a variety of ways. Some possibilities are shown in Figure below:

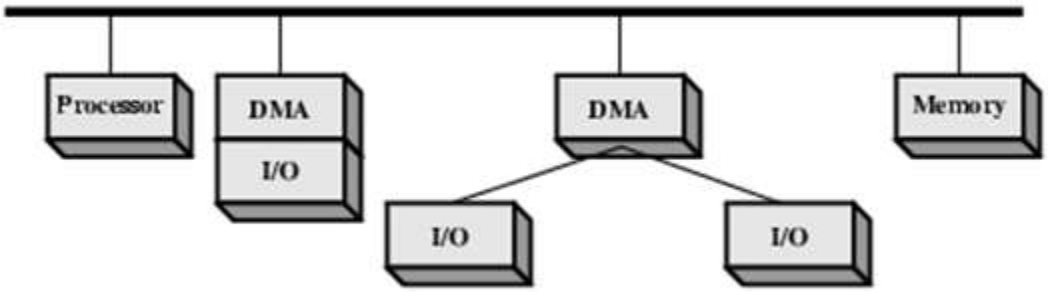
DMA configuration(1)



(a) Single Bus, Detached DMA controller

- Each transfer uses bus twice(IO to DMA then DMA to memory)
- CPU is suspended twice .

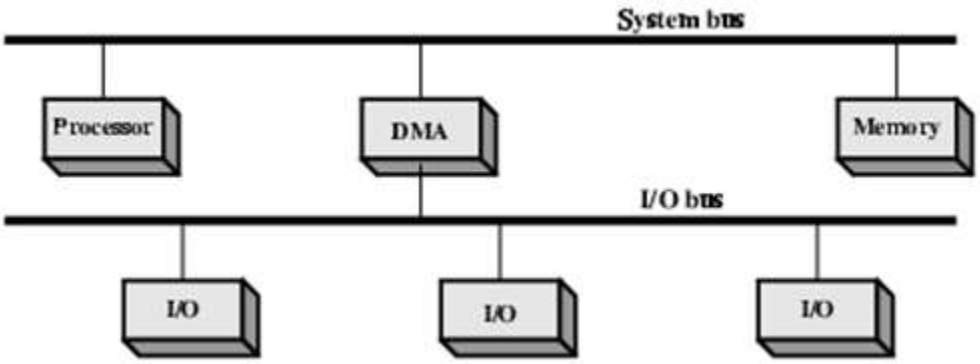
DMA configuration (2)



(b) Single-bus, Integrated DMA-I/O

- Single Bus, **Integrated** DMA controller(support one IO device or more)
- Each transfer uses bus once(DMA to memory) so CPU may be suspended only once.

DMA configuration(3)



(c) I/O bus

- Separate IO Bus
- IO Bus connect between DMA and all enabled devices
- Each transfer uses bus once (DMA to memory) so CPU suspended once.