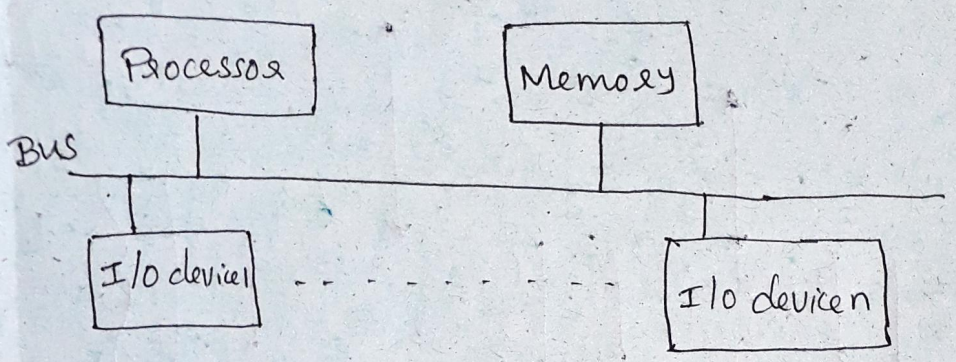
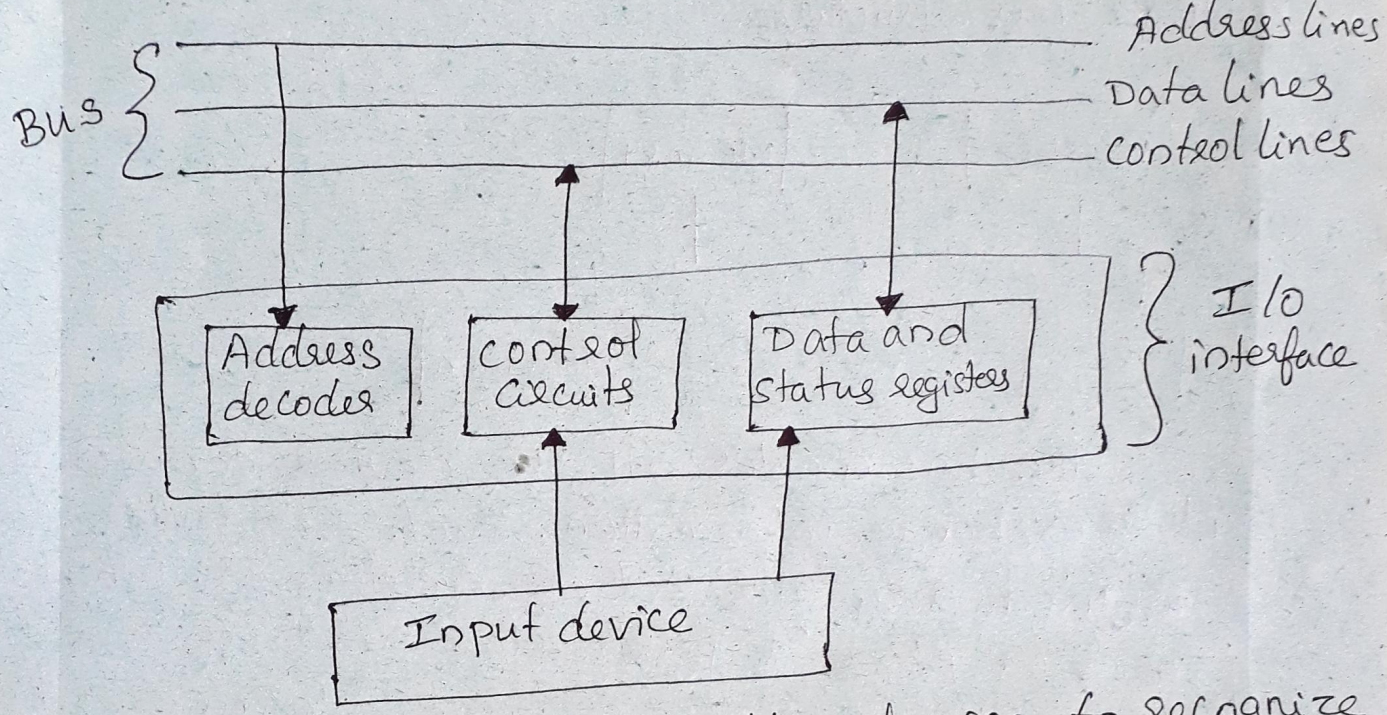
Accessing I/O Devices

* Single-bus structure



The bus enables all the devices connected to it to exchange information. Each I/O device is assigned a unique set of addresses. When the processor places a particular address on the address lines, the device that recognizes this address responds to the commands issued on the control lines. The processor requests either a read or a write operation, and the requested data are transferred over the data lines. When I/O devices and the memory share the same address space, the arrangement is called memory-mapped I/O. With memory mapped I/O, any machine instruction that can access memory can be used to transfer data to or from an I/O device.

* I/O interface for an input device



Address decoder: enables the device to recognize its address when this address appears on the address lines.

Data Register: holds the data being transferred to or from the processor.

Status Register: contains information relevant to the operation of the I/O device.

NOTE: Both the data and status registers are connected to the data bus and assigned unique addresses.

I/O devices operate at speeds that are vastly different from that of the processor.

An instruction that reads a character from the keyboard should be executed only when a character is available in the input buffer of the keyboard interface. Also, we must make sure that an input character is read only once.

For an input device such as a keyboard a status flag, SIN, is included in the interface circuit as part of the status register. This flag is set to '1' when a character is entered at the keyboard and cleared to '0' once this character is read by the processor. Hence, by checking the SIN flag, the software can ensure that it is always reading valid data. This is often accomplished in a program loop that repeatedly reads the status register and checks the state of SIN. When SIN becomes equal to 1, the program reads the input data register. A similar procedure can be used to control ouput operation using an output status flag, SOUT.

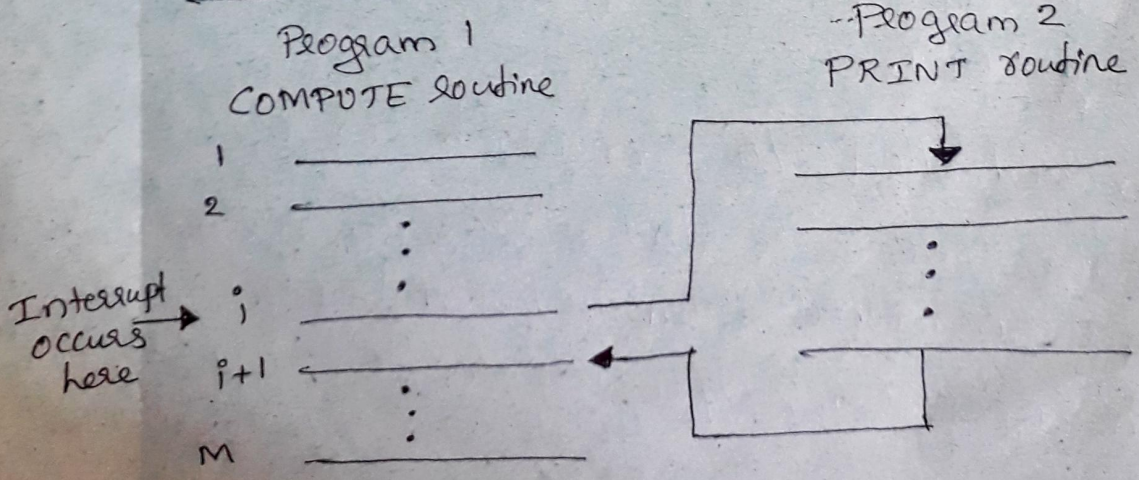
Program controlled I/O, in which the processor repeatedly checks a status flag to achieve the required synchronization between the processor and an input or output device.

There are 2 other commonly used mechanisms for implementing I/O operations:
(1) Interrupts (2) Direct Memory Access (DMA)

Interrupts

Synchronization is achieved by having the I/O device send a special signal, called interrupts, over the bus whenever it is ready for a data transfer operation. Atleast one of the bus control lines, called an interrupt-request line, is usually dedicated for this purpose. Since the processor is no longer required to continuously check the status of external devices, it can use the waiting period to perform other useful functions. The routine executed in response to an interrupt request is called the interrupt-service routine.

Transfer of control through the use of interrupts



Assume that an interrupt request arrives during execution of instruction 'i' in above figure. The processor first completes execution of instruction i. Then, it loads the Program Counter with the address of the first instruction of the interrupt-service routine. After execution of the interrupt-service routine, the processor has to come back to instruction i+1. Therefore, when an interrupt occurs, the current contents of the PC, which point to instruction i+1, must be put in temporary storage in a known location. A Return from interrupt instruction at the end of the interrupt-service routine reloads the PC from that temporary storage location, causing execution to resume at instruction i+1.

The interrupt service routine may not have anything in common with the program being executed at the time the interrupt request is received. In fact, the two programs often belong to different users. Therefore, before starting execution of the interrupt-service routine, any information that may be altered during the execution of that routine must be saved. this information must be restored before

execution of the interrupted program is resumed. The information that needs to be saved and restored typically includes the condition code flags and the contents of any registers used by both the interrupted program and the interrupt-service routine.