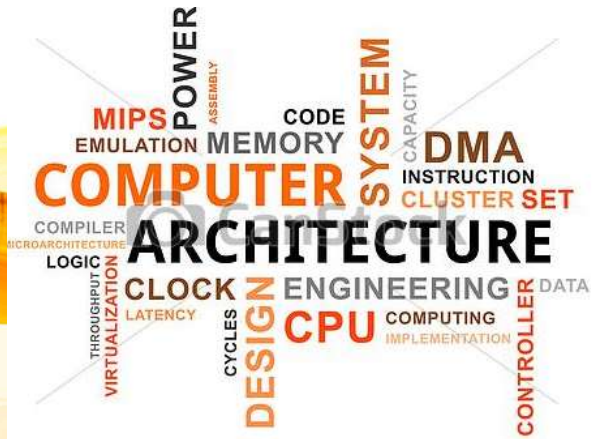
UNIT V
I/O ORGANIZATION AND PARALLELISM

Accessing I/O devices – Interrupts – Direct Memory Access - Buses– Interface circuits - Standard I/O Interfaces (PCI, SCSI, USB)–Instruction Level Parallelism : Concepts and Challenges – Introduction to multicore processor Graphics Processing Unit.



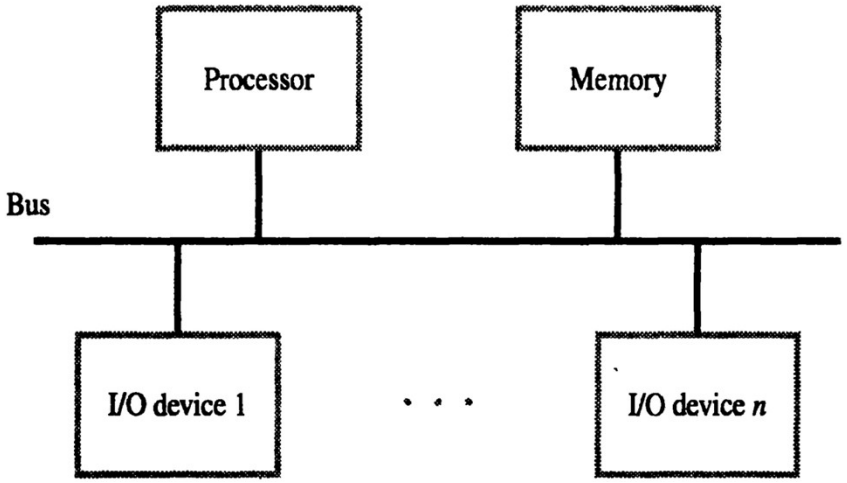
Recap the previous Class

I/O in CO and O/S

- Programmed I/O
- Interrupts
- DMA (Direct memory Access)

Device	Behavior	Partner	Data rate (Mbit/sec)
Keyboard	input	human	0.0001
Mouse	input	human	0.0038
Voice input	input	human	0.2640
Sound input	input	machine	3.0000
Scanner	input	human	3.2000
Voice output	output	human	0.2640
Sound output	output	human	8.0000
Laser printer	output	human	3.2000
Graphics display	output	human	800.0000–8000.0000
Modem	input or output	machine	0.0160–0.0640
Network/LAN	input or output	machine	100.0000–1000.0000
Network/wireless LAN	input or output	machine	11.0000–54.0000
Optical disk	storage	machine	80.0000
Magnetic tape	storage	machine	32.0000
Magnetic disk	storage	machine	240.0000–2560.0000

- A bus is a shared communication link, which uses one set of wires to connect multiple subsystems.
- The two major advantages of the bus organization are versatility and low cost.



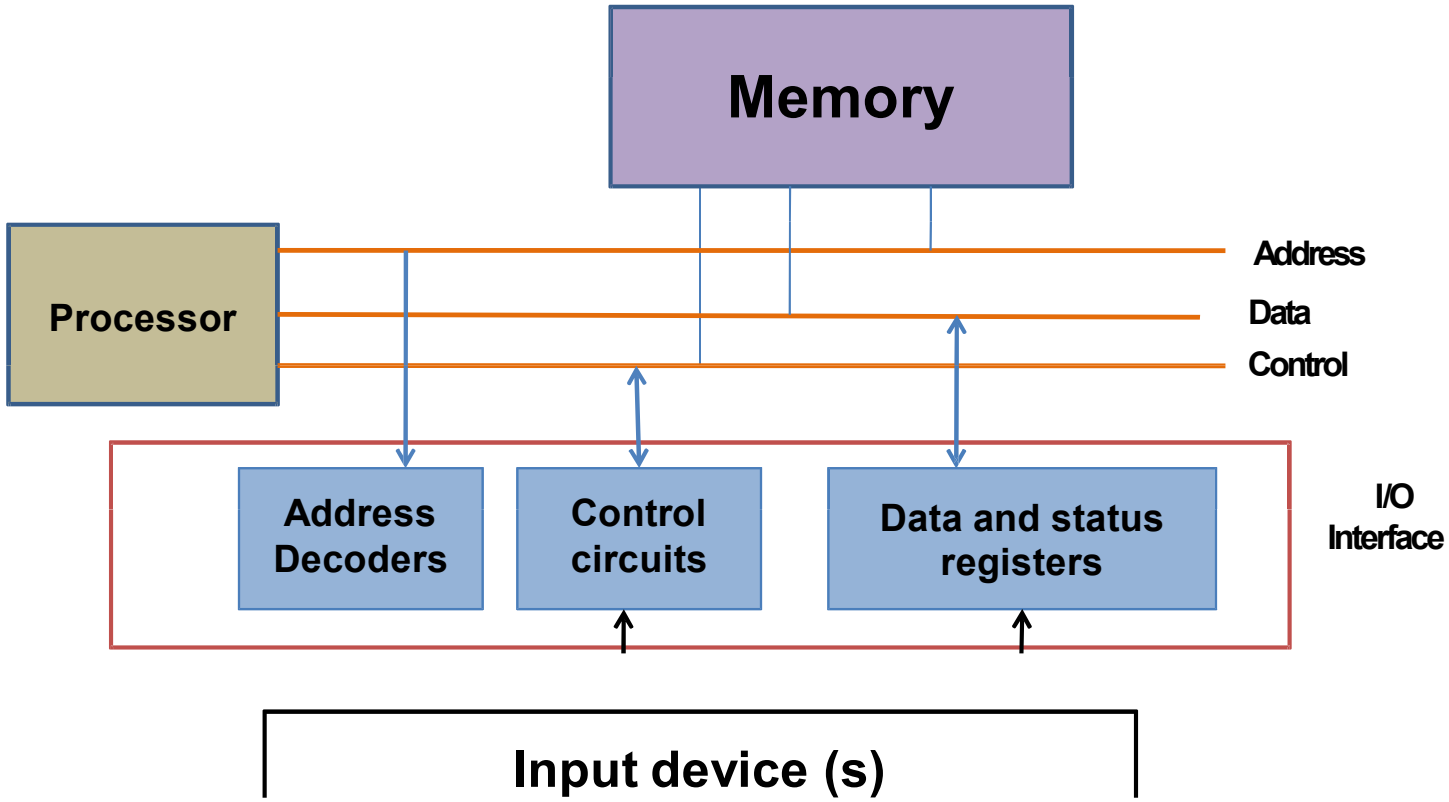
Accessing I/O Devices

- Most modern computers use single bus arrangement for connecting I/O devices to CPU & Memory
- The bus enables all the devices connected to it to exchange information
- Bus consists of 3 set of lines : *Address, Data, Control*
- Processor places a particular address (unique for an I/O Dev.) on address lines
- Device which recognizes this address responds to the
- Processor requests for either Read / Write
- The data will be placed on Data lines

Hardware to connect I/O devices to bus

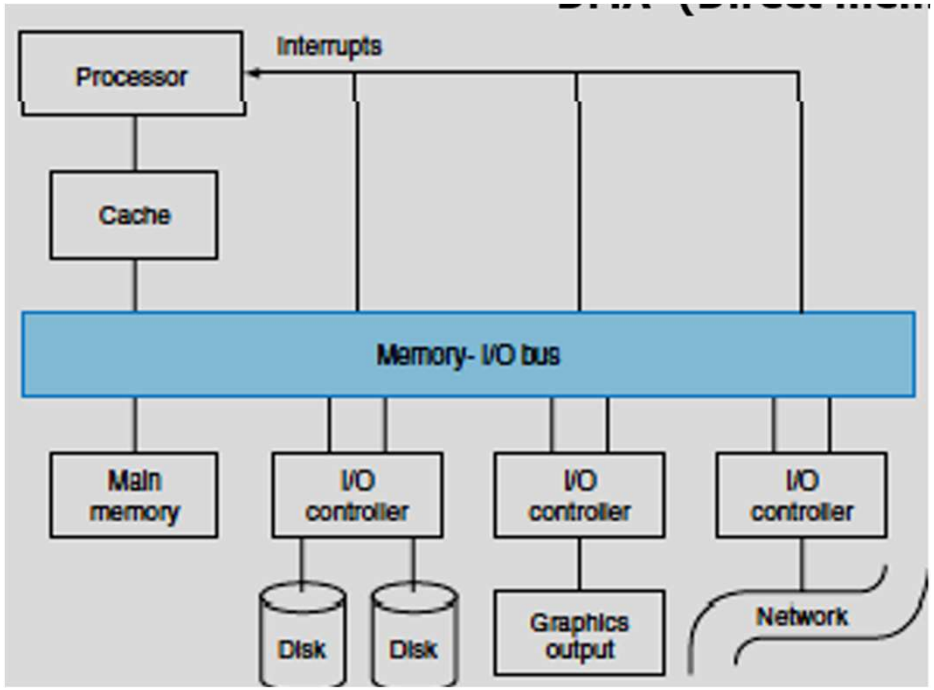
- Interface Circuit
  - Address Decoder
  - Control Circuits
  - Data registers
  - Status registers
- The Registers in I/O Interface – buffer and control
- Flags in Status Registers, like SIN, SOUT
- Data Registers, like Data-IN, Data-OUT

I/O interface for an input device



Input Output mechanism

- Memory mapped I/O
- Programmed I/O
- Interrupts
- DMA(Direct memory Access)



Input Output mechanism

- A bus generally contains a set of control lines and a set of data lines.
- The control lines are used to signal requests and acknowledgments, and to indicate what type of information is on the data lines. The control lines are used to indicate what the bus contains and to implement the bus protocol.
- The data lines of the bus carry information between the source and the destination. This information may consist of data, complex commands, or addresses.
- Buses are traditionally classified as processor-memory buses or I/O buses or special purposed buses (Graphics, etc.). Processor memory buses are short, generally high speed, and matched to the memory system so as to maximize memory- processor bandwidth.
- I/O buses, by contrast, can be lengthy, can have many types of devices connected to them, and often have a wide range in the data bandwidth of the devices connected to them. I/O buses do not typically interface directly to the memory but use either a processor-memory or a backplane bus to connect to memory.

- The major disadvantage of a bus is that it creates a communication bottleneck, possibly limiting the maximum I/O throughput.
- When I/O must pass through a single bus, the bandwidth of that bus limits the maximum I/O throughput.

Reason why bus design is so difficult :

- the maximum bus speed is largely limited by physical factors: the length of the bus and the number of devices. These physical limits prevent us from running the bus arbitrarily fast.
- In addition, the need to support a range of devices with widely varying latencies and data transfer rates also makes bus design challenging.
- it becomes difficult to run many parallel wires at high speed due to clock skew and reflection.

- The two basic schemes for communication on the bus are synchronous and asynchronous.
- If a bus is synchronous (e.g. Processor-memory), it includes a clock in the control lines and a fixed protocol for communicating that is relative to the clock.
- This type of protocol can be implemented easily in a small finite state machine. Because the protocol is predetermined and involves little logic, the bus can run very fast and the interface logic will be small.

Synchronous buses have two major disadvantages:

- First, every device on the bus must run at the same clock rate.
- Second, because of clock skew problems, synchronous buses cannot be long if they are fast.

An **asynchronous bus** is not clocked. It can accommodate a wide variety of devices, and the bus can be lengthened without worrying about clock skew or synchronization problems.

To coordinate the transmission of data between sender and receiver, an asynchronous bus uses a handshaking protocol.

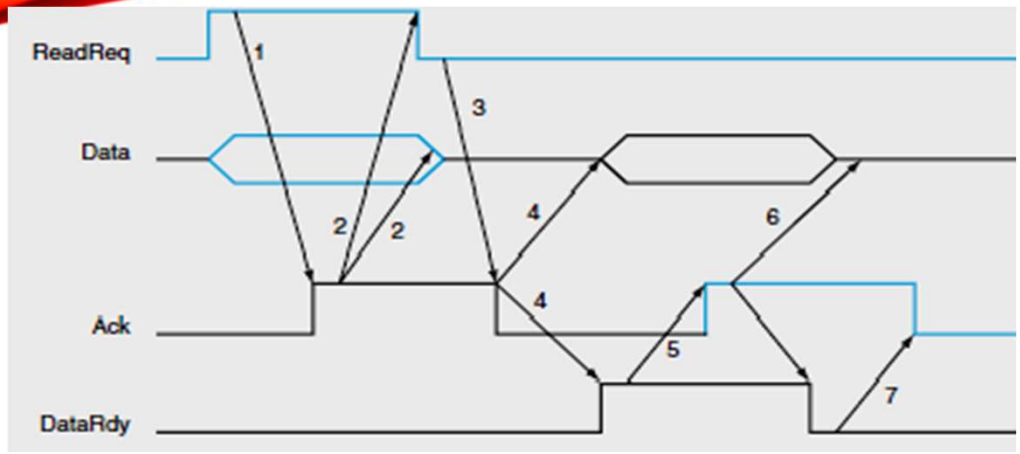
Characteristic	Firewire (1394)	USB 2.0
Bus type	I/O	I/O
Basic data bus width (signals)	4	2
Clocking	asynchronous	asynchronous
Theoretical peak bandwidth	50 MB/sec (Firewire 400) or 100 MB/sec (Firewire 800)	0.2 MB/sec (low speed), 1.5 MB/sec (full speed), or 60 MB/sec (high speed)
Hot plugable	yes	yes
Maximum number of devices	63	127
Maximum bus length (copper wire)	4.5 meters	5 meters
Standard name	IEEE 1394, 1394b	USB Implementors Forum

Three special control lines required for hand-shaking:

- ReadReq: Used to indicate a read request for memory. The address is put on the data lines at the same time.
- DataRdy: Used to indicate that the data word is now ready on the data lines; asserted by: Output/Memory and Input/I_O Device.
- Ack: Used to acknowledge the ReadReq or the DataRdy signal of the other party.

I/O Dev.

Memory



Steps after the device signals a request by raising ReadReq and putting the address on the Data lines:

1.When memory sees the ReadReq line, it reads the address from the data bus and raises Ack to indicate it has been seen.

2. As the Ack line is high - I/O releases the ReadReq and data lines.

3.Memory sees that ReadReq is low and drops the Ack line to acknowledge the ReadReq signal (*Mem. Reading in progress now*).

4.This step starts when the memory has the data ready. It places the data from the read request on the data lines and raises DataRdy.

5.The I/O device sees DataRdy, reads the data from the bus, and signals that it has the data by raising Ack.

6. On the Ack signal, M/M drops DataRdy, and releases the data lines.

7.Finally, the I/O device, seeing DataRdy go low, drops the Ack line, which indicates that the transmission is completed.

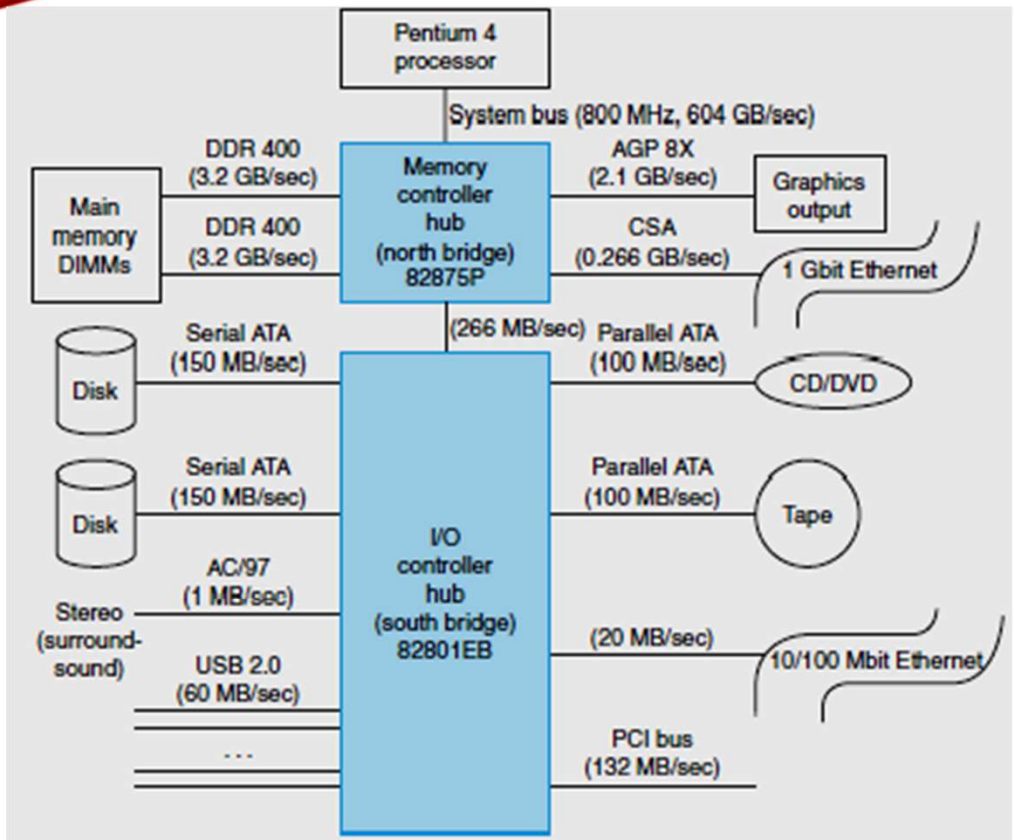


FIGURE 8.11 Organization of the I/O system on a Pentium 4 PC using the Intel 875 chip set. Note that the maximum transfer rate between the north bridge (memory hub) and south bridge (I/O hub) is 266 MB/sec, which is why Intel put the AGP bus and Gigabit Ethernet on the north bridge.

Memory mapped I/O

- I/O devices and the memory share the same address space, the arrangement is called *Memory-mapped I/O.*

- In Memory-mapped I/O portions of address space are assigned to I/O devices and reads and writes to those addresses are interpreted as commands to the I/O device.

"DATAIN" is the address of the input buffer associated with the keyboard.

- Move DATAIN, R0

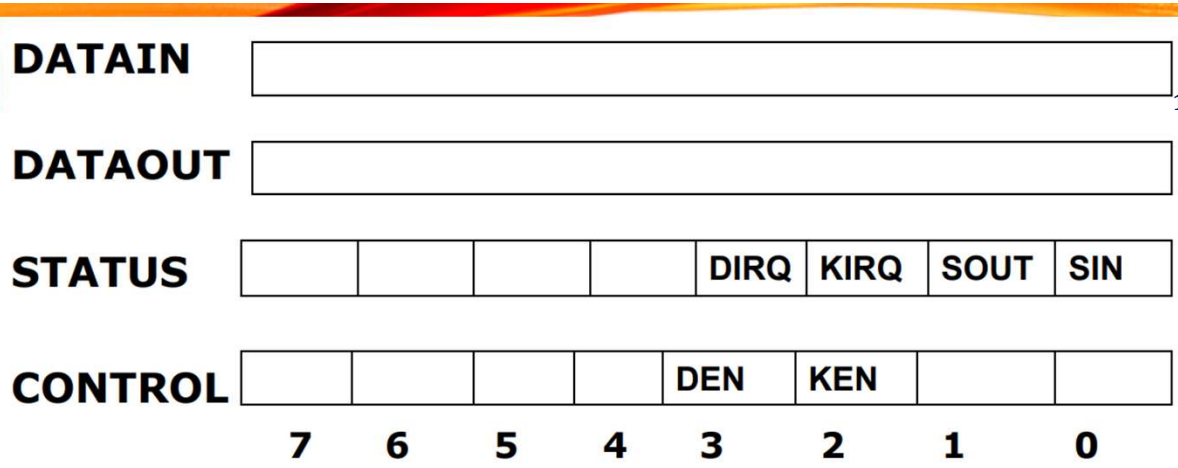
reads the data from DATAIN and stores them into processor register R0;

- Move R0, DATAOUT

sends the contents of register R0 to location DATAOUT

Option of special I/O address space or incorporate as a part of memory address space (address bus is same always).

- When the processor places the address and data on the memory bus, the memory system ignores the operation because the address indicates a portion of the memory space used for I/O.
- The device controller, however, sees the operation, records the data, and transmits it to the device as a command.
- User programs are prevented from issuing I/O operations directly because the OS does not provide access to the address space assigned to the I/O devices and thus the addresses are protected by the address translation.
- Memory mapped I/O can also be used to transmit data by writing or reading to select addresses. The device uses the address to determine the type of command, and the data may be provided by a write or obtained by a read.
- A program request usually requires several separate I/O operations. Furthermore, the processor may have to interrogate the status of the device between individual commands to determine whether the command completed successfully.



I/O operation involving keyboard and display devices

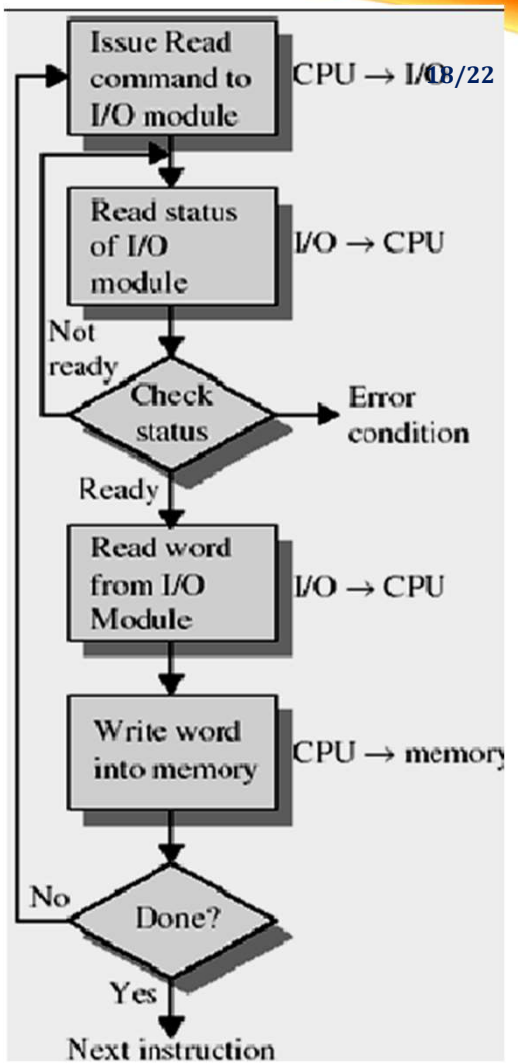
Registers: DATAIN, DATAOUT, STATUS, CONTROL

Flags: SIN, SOUT - Provides status information for keyboard and display unit

KIRQ, DIRQ – Keyboard, Display Interrupt request bits DEN, KEN –Keyboard, Display Enable bits

Programmed I/O

1. CPU has direct control over I/O
    1. Sensing status
    2. Read/write commands
    3. Transferring data
2. CPU waits for I/O module to complete operation
3. Wastes CPU time



- In this case, use dedicated I/O instructions in the processor. These I/O instructions can specify both the device number and the command word (or the location of the command word in memory).
- The processor communicates the device address via a set of wires normally included as part of the I/O bus. The actual command can be transmitted over the data lines in the bus. (example - Intel IA-32).
- By making the I/O instructions illegal to execute when not in kernel or supervisor mode, user programs can be prevented from accessing the devices directly.
- The process of periodically checking status bits to see if it is time for the next I/O operation, is called polling. Polling is the simplest way for an I/O device to communicate with the processor.
- The I/O device simply puts the information in a Status register, and the processor must come and get the information. The processor is totally in control and does all the work.

A ISA program to read one line from the keyboard, store it in memory buffer, and echo it back to the display

```
        Move        #LINE,R0          Initialize memory pointer.
WAITK   TestBit     #0,STATUS         Test SIN.
        Branch=0    WAITK             Wait for character to be entered.
        Move        DATAIN,R1         Read character.
WAITD   TestBit     #1,STATUS         Test SOUT.
        Branch=0    WAITD             Wait for display to become ready.
        Move        R1,DATAOUT        Send character to display.
        Move        R1,(R0)+          Store charater and advance pointer.
        Compare     #$0D,R1           Check if Carriage Return.
        Branch≠0    WAITK             If not, get another character.
        Move        #$0A,DATAOUT      Otherwise, send Line Feed.
        Call        PROCESS           Call a subroutine to process the
                                        the input line.
```

- The disadvantage of polling is that it can waste a lot of processor time because processors are so much faster than I/O devices.
- The processor may read the Status register many times, only to find that the device has not yet completed a comparatively slow I/O operation, or that the mouse has not budged since the last time it was polled.
- When the device completes an operation, we must still read the status to determine whether it (I/O) was successful.
    - Overhead in a polling interface lead to the
- invention of interrupts to notify the processor when an I/O device requires attention from the processor.
- Interrupt-driven I/O, employs I/O interrupts to indicate to the processor that an I/O device needs attention.
- When a device wants to notify the processor that it has completed some operation or needs attention, it causes the processor to be interrupted.


Thank You