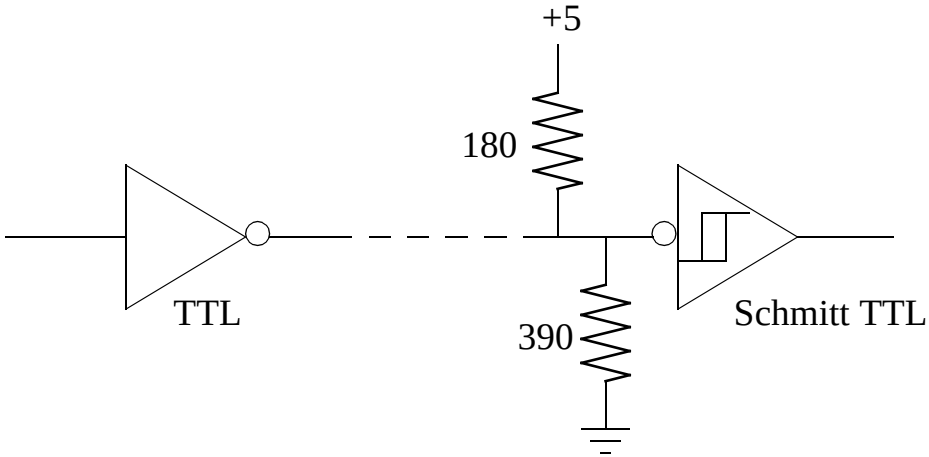
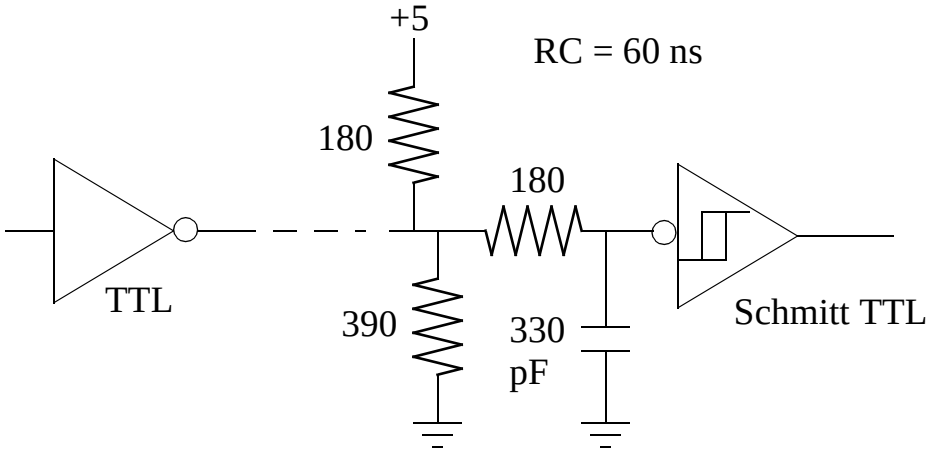
Data Cables

- Data cables link one instrument to another.
- Signals can attenuate or disperse on long wires.
- A direct wire works best for short cables of less than 10 ft.
- A TTL cable connection can use a Schmitt input to negate noise.



Signal Slowdown

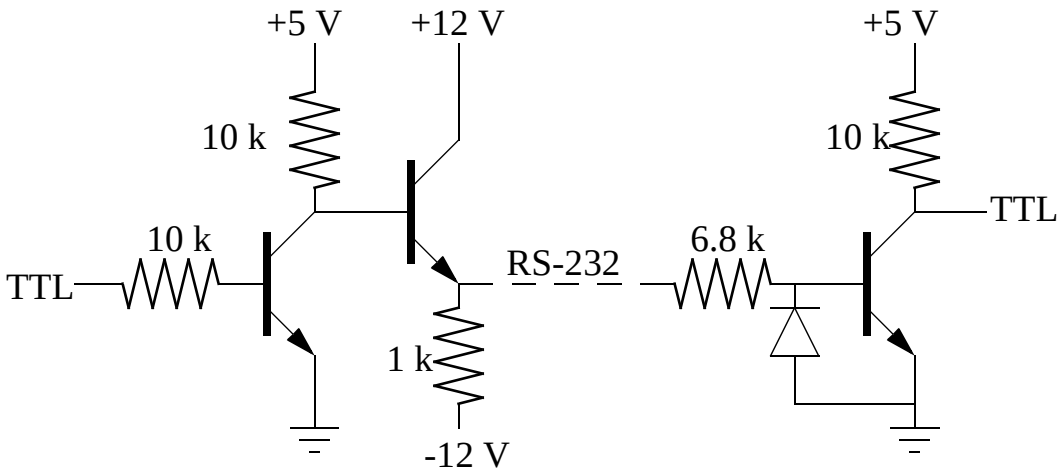
- If a digital signal is not too fast, an RC circuit can integrate noise.
- The cost is a slower signal response.



- The RC time constant here is 60 ns, and would not have much effect on signals at 1 MHz.

Bipolar Drive: RS-232

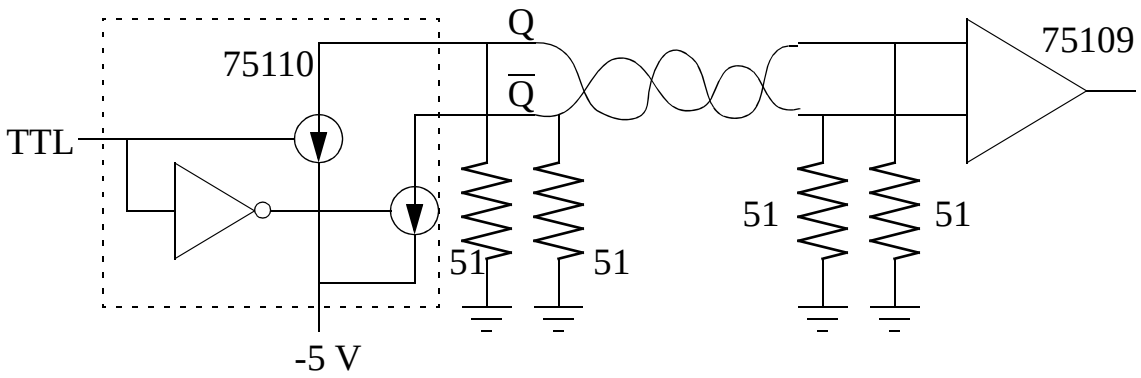
- The RS-232 standard increases the voltage scale to provide a clearer difference between logic HIGH and LOW.
- This standard can use voltages that ranges from -5 to +5 V up to -15 to +15 V including asymmetric combinations like +12 V/-5 V, with voltage HIGH corresponding to FALSE.



- RS-232 forces additional power supply requirements in addition to +5 V.
- RS-232 is used for many computer connections of up to a few tens of feet in a noisy environment.
- Drivers and receivers for RS-232 can be made from discrete components, op-amps/logic gates, or dedicated RS-232 drivers.
- The maximum slew rate is 30V/µs and the system can handle rates of up to 20-40 kb/s.

Differential Drive

- Differential drive cables improve on noise immunity by using two wires per signal, one with Q and one with $\overline{Q}$.
- Differential Current Drive



- The 75110/75109 combination can drive 1 Mbit/s over 2000 ft and up to 10 Mb/s for a few hundred feet.
- Like RS-232 this system relies on at least one additional power supply (-5 V).

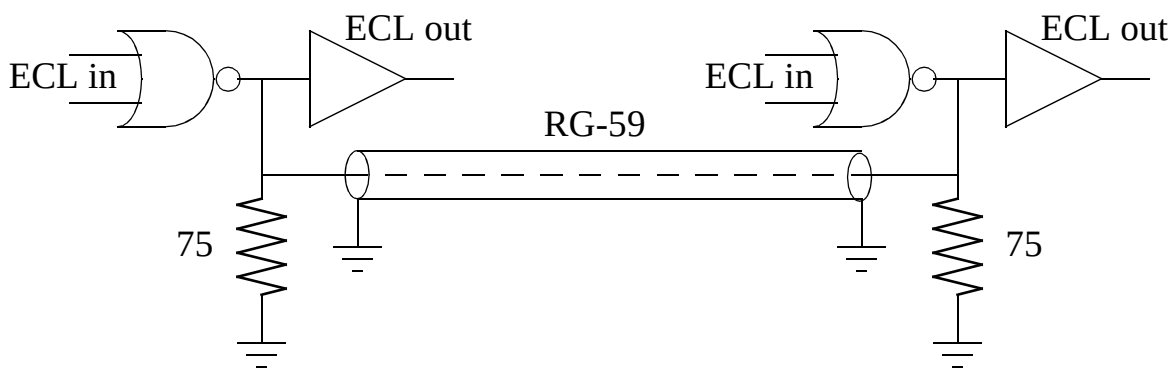
RS-422

- This differential standard uses just a +5V supply to ground and the Q and $\overline{Q}$ are basically TTL.
- There is no slew rate limit like RS-232.
- Data rates of up to 10 Mb/s are possible on 40 ft of cable and 100 kb/s are possible even on 4000 ft of cable.

Coaxial Cable Driver

- Coaxial cable will suppress noise by jacketing the signal line in a ground sheet.
- Each cable has a characteristic impedance that must be matched to avoid reflections of the signal.
- For example RG-58 is a 50 Ω cable, and RG-59 is a 75 Ω cable.
- Many drivers and receivers are available for TTL or other logic levels.

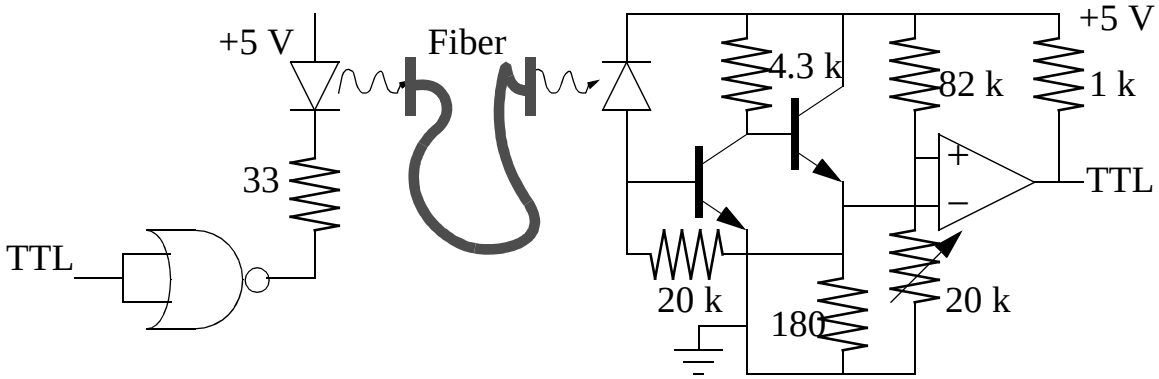
- One example is the full-duplex transceivers for ECL (10194)



- Simultaneous asynchronous bidirectional transmission of 100 Mb/s.
- High speed transmission of 1 Gb/s over 100 ft.

Fiber-optic Drive

- Fiber-optics are used for long high-speed digital transmission.
- The copper transmission cable is replaced with a fiber-optic cable.



- Fiber-optic cable is capable many Gb/s over many miles.

RS-232 Serial Interface Protocol

- RS-232 specifies a 25-pin D-connector that is reduced to 9 functional lines, often on a 9-pin D-connector.
- The 9 lines are in 4 pairs plus a ground line:

1. Data Pair - transmit data (TD) and receive data (RD).
2. Handshake Pair - request to send (RTS) and clear to send (CTS).
3. Handshake Pair - data terminal ready (DTR) and data set ready (DSR).
4. Enable Input Pair - data carrier detect (DCD) and ring indicator (RI).

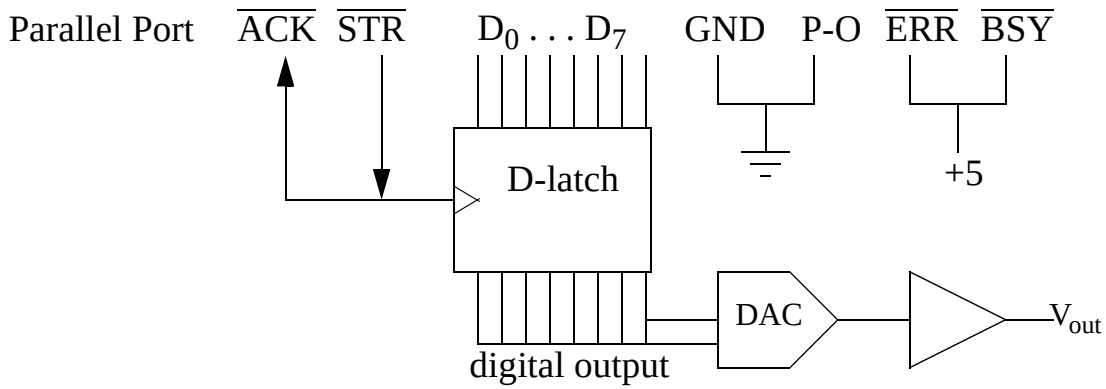
The RS-232 is an old protocol that is based on data terminal equipment (DTE) communicating with data communication equipment (DCE). Historically this was a dumb terminal and a modem, and all the wires directly connect if that is the connection. The above lines are all descibed from the DTE perspective.

The modern use is far more complicated since microprocessors and some other smart peripherals are ambiguous as to DTE/DCE status. For instance the IBM-PC is DTE but most other computers are DCE. The result is that one needs to know the status of the device and cable it correctly. A frequently encountered object is a *null-modem* cable which reverses wires so that DTE-DTE and DCE-DCE communication is possible.

Also, not all handshakes are used by all devices, so there are more cable types involved. Typically a handshaking object is attached to a non-handshaking object, so the handshake pairs are directly connected back to the sender.

Parallel Interface Protocol

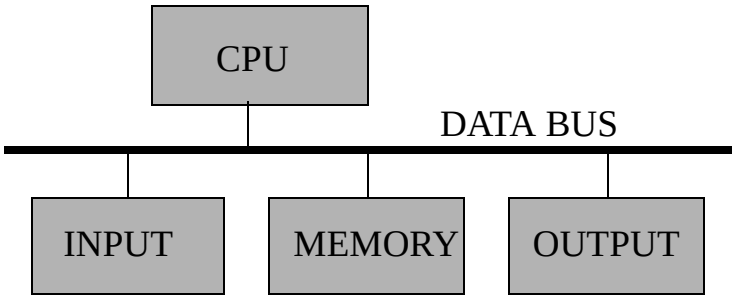
- This is also called the Centronics protocol and also uses a 25-pin D-connector.



Pin 1	$\overline{STROBE}$	Pin 10	$\overline{ACK}$ (input)
Pin 2	D_0	Pin 11	$\overline{BUSY}$ (input)
Pin 3	D_1	Pin 12	PAPER EMPTY (input)
Pin 4	D_2	Pin 13	SELECT (input)
Pin 5	D_3	Pin 14	AUTO FEED
Pin 6	D_4	Pin 15	$\overline{ERROR}$ (input)
Pin 7	D_5	Pin 16	$\overline{INIT}$
Pin 8	D_6	Pin 17	SLCTIN
Pin 9	D_7	Pin 18-25	Ground

Computer Buses

- All microcomputer buses have well defined lines to connect between the CPU and other devices.



- By convention signals go out or in with respect to the CPU.
- All buses include some of the following signals:

<u>Power supply</u> lines carry power and ground to all the devices onnected to the bus.

<u>Address lines</u> indicate what device is being communicated with, and what location is specified within a device. This includes addresses for both memory and i/o devices.

<u>Data lines</u> carry the actual data in or out of the CPU. On some buses data can also go between devices not involving the CPU.

<u>Data strobes</u> indicate to devices when data is valid.

<u>Timing</u> lines carry a standard clock pulse to all devices.

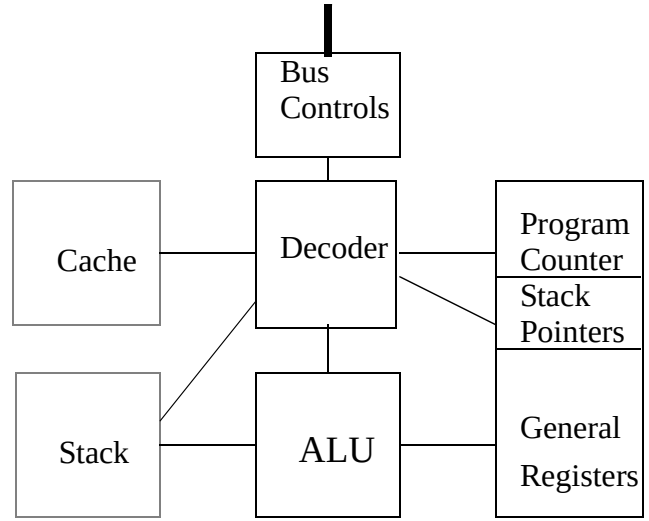
<u>Interrupt requests</u> allow a device to request action from the CPU.

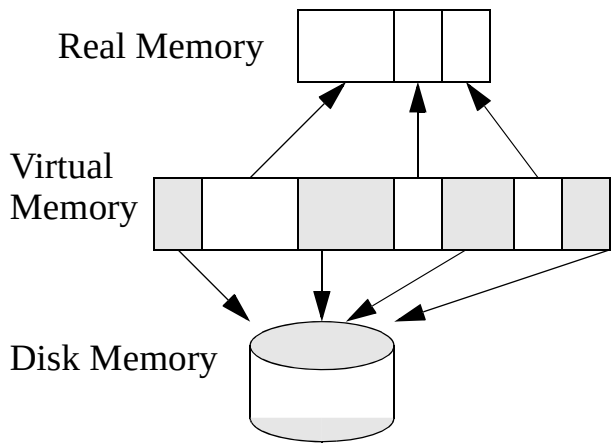
Microprocessor Architecture

- The CPU is the microprocessor that provides the control and arithmetic functions for the computer.

- Limited memory is within the CPU. Most of the memory is connected by the bus.

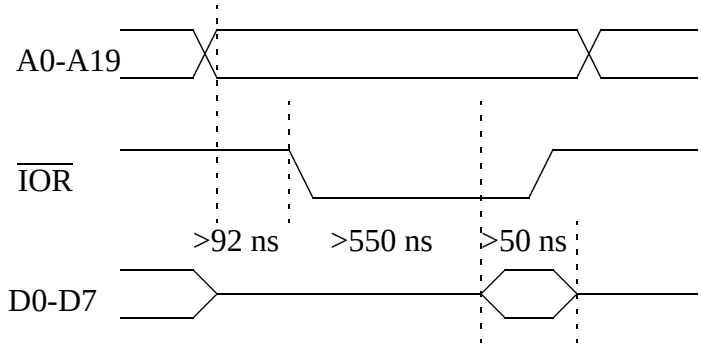
- Some machines have more addressing capacity than available memory (eg. a 1 GB limit though 32 bits can address 4 GB.) This is called virtual memory and uses part of the disk storage to extend the memory.





Data Timing

- Data input and data output have similar timing sequences.
- The primary difference is that input requires a read strobe instead of a write strobe, and the data is valid after a longer period of time since a response is required by the input device.
- Input timing for the original IBM-PC is sequences address lines, data lines and the read strobe.



Input / Output Ports

- Data is moved in and out of a CPU through data ports.
- The basic assembly instructions move data in and out of internal registers.

Input Port Instructions (8088)

IN

`IN AL,`*portno* ; replace AL with one byte from location *portno*.

`IN AX,`*portno* ; replace AX with one word from location *portno*.

`IN AL,DX` ; replace AL with one byte from the location in DX.

`IN AX,DX` ; replace AX with one byte from the location in DX.

Output Port Instructions (8088)

OUT

`OUT` *portno***`,AL`** ; send one byte from AL to location *portno*.

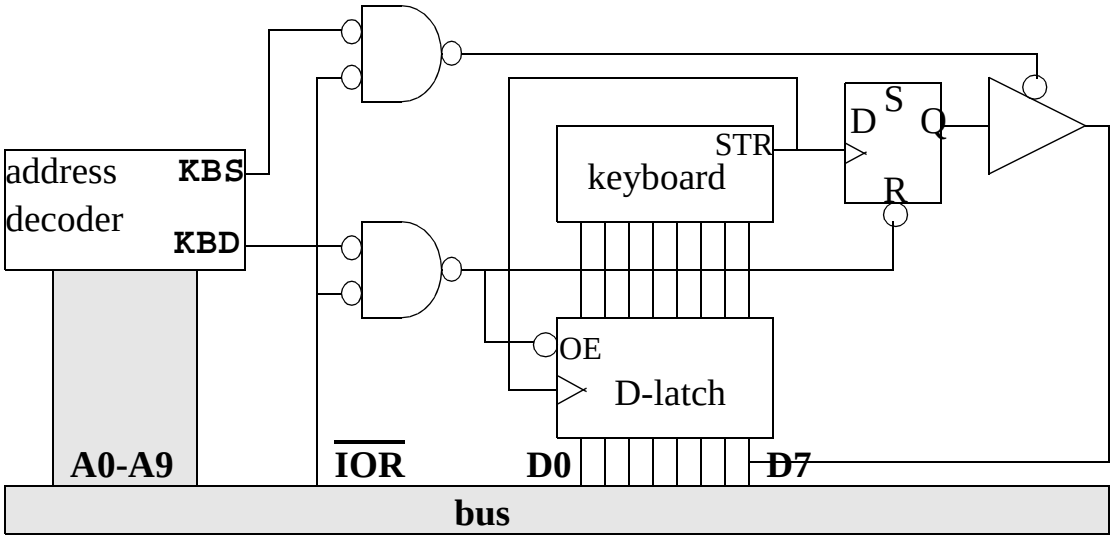
`OUT` *portno***`,AX`** ; send one byte from AX to location *portno*.

`OUT DX,AL` ; send one byte from AL to the location in DX.

`OUT DX,AX` ; send one byte from AX to the location in DX.

Register Transfers

- The keyboard sends an 8-bit character plus a strobe when a key is hit.
- An address decoder allows the CPU to read the strobe or the data.




```
        MOV BP,OFFSET(BUF)    ; set buffer pointer to initial location.
GETKB   IN AL.KBS             ; get status word from keyboard.
        AND AL,80H            ; bit 7 indicates a new character.
        JZ GETKB              ; no data, try again.
        IN AL,KBD             ; get data from keyboard.
        MOV [BP],AL           ; store character in buffer.
        INC BP                ; move buffer pointer to next location.
        CMP AL,0DH            ; test for carriage return.
        JNZ GETKB             ; not a carriage return, get another character.
```

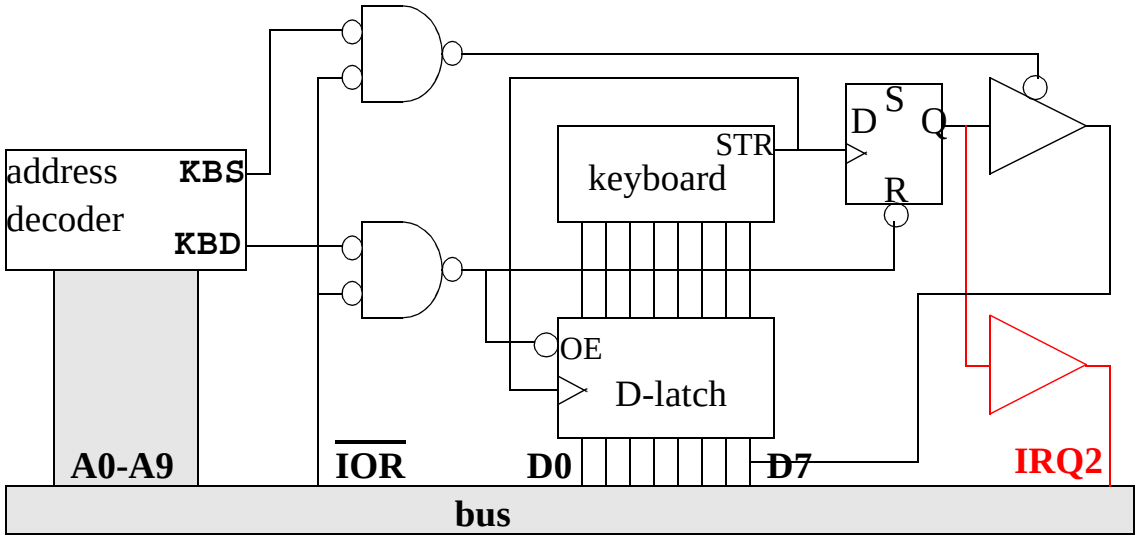
Interrupts

- Status registers require the CPU to actively check a device.
- Interrupts allow the device to force action from the CPU.
- Interrupts are edge triggered and each can be attached to only one interrupting device.
- A computer will have an IC to prioritize interrupts and generate the correct pointers.

- If more than some number of interrupts are required, then 2 or more devices will have to share an interrupt line. The interrupt controller together with the software will have to determine which device requested action.
- *Autovectored* interrupts require devices to have a status register that is set when an interrupt is made. When an interrupt is detected, the handler software polls each status register that is associated with the interrupt line. When a set register is found, the correct action is taken.
- *Acknowledged* interrupts require devices to respond to a special iterrupt acknowledge signal and then send placing the correct vector for that device on the data lines. This method eliminates the need to poll devices independently since the acknowledge goes to all possible devices at once.
- An *interrupt mask* can turn off unwanted interrupts from within the CPU program. For instance, printer interrupts for more data should be off when there is nothing to print.
- *Software* interrupts perform the same function in software as a hardware device making an interrupt. For example a software interrupt might be used to load a string of characters or perform a screen capture.

Keyboard Input with Interrupts

- This circuit adds an additional buffer to drive IRQ2.




```
; Beginning of program
MOV BP,OFFSET(BUF)
; set buffer pointer to initial location.
MOV 28H,OFFSET(GETKB)
; store address of interrupt subroutine.
; Main program.
```

```
GETKB   PUSH AX         ; save register while in the subroutine.
        IN AL,KBD       ; get data from keyboard.
        MOV [BP],AL     ; store character in buffer.
        INC BP          ; move buffer pointer to next location.
        CMP AL,0DH      ; test for carriage return.
        JNZ GETKB       ; not a carriage return, get another character.
        POP AX          ; restore register after the subroutine.
        IRET            ; return to main program.
```