CS 320
Ch 7 – I/O

An I/O module **Provides an interface to the processor and memory via the system bus and to provide an interface to one or more I/O devices via a tailored link.**

A keyboard I/O device works by using **the IRA (International Reference Alphabet) code (also called ASCII in U.S.) which has alphabetic plus control characters. Is typically a 7-bit code with an 8th bit added as a parity bit.**

A transducer is a device which **Converts data from electrical to other energy forms and vice-versa. A disk head is a transducer which converts electrical signals to magnetic signals.**

I/O modules are often responsible for data buffering. This is n**ecessary because of the disparity in time between the I/O device and the system bus. Most disk systems buffer data that is read to and from the disk.**

The term I/O module is a generic term that refers to *I/O Channels, I/O Processors, I/O Controllers, or Device Controllers.* **I/O Channel and I/O Processor is the same. It handles the details of doing I/O and is actually another specialized computer. I/O Controllers and Device Controllers are the same and refer to an interface in which the CPU does most of the I/O detail work. I/O processors are used on mainframes and I/O Controllers are used on microcontrollers.**

When a processor, main memory, and I/O share a common bus, there are two types of I/O possible: *memory mapped I/O and Isolated I/O.* **Memory mapped is in the memory address space and has no explicit I/O instructions. Isolated is in its own address space and has explicit I/O instructions.**

In order to have isolated I/O a processor needs at least one additional line usually called the $I/O/\overline{M}$ **line which allows the address to be enable for either I/O or Memory.**

For a system which has isolated I/O, it is possible to do memory mapped I/O also by decoding a memory address for an I/O device.

Interrupt Processing
Interrupt driven I/O. **Solves the wait problem caused by the disparity in time between the CPU and an I/O device. CPU starts I/O and device interrupts when it needs attention.**

The author lists 9 operations that are needed to process an interrupt. The figure below summarizes these processes.

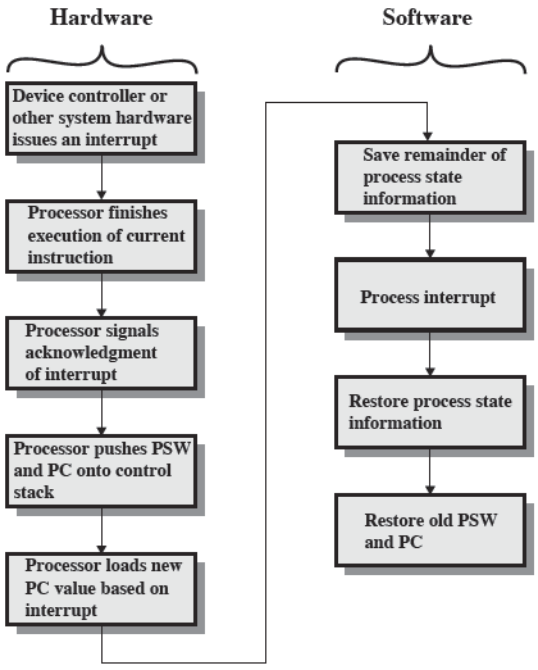

Figure 7.6 Simple Interrupt Processing

There are 4 generic methods of handling multiple interrupts. **Multiple interrupt lines, software polling, daisy chain hardware polled, and bus arbitration or vectored.**

Multiple interrupt lines method has the following advantages and disadvantages: **CPU as a separate line for each interrupt. Unusual to have more than 4. Must be expanded using one of the other three methods. Can be fast.**

Software polling method has the following advantages and disadvantages: **When an int occurs software reads a status byte from each I/O device in turn. Has a built in natural priority order. Slow but cheap.**

The Daisy Chain method has the following advantages and disadvantages. **I/O modules are chained together on a common signal line. An int ack is passed down the line until it reaches the interrupting device. Device then causes interrupt via a vector. Natural priorities, relatively cheap, can be slow. Easily expandable.**

For the bus arbitration method an e**xternal arbitrator decides on priorities and allows only one device at a time to issue an int. Interrupting device puts a vector on the bus when it is acknowledged. Fast, easy to expand, requires external hardware.**

The Figure below is that of an 8259A interrupt controller.

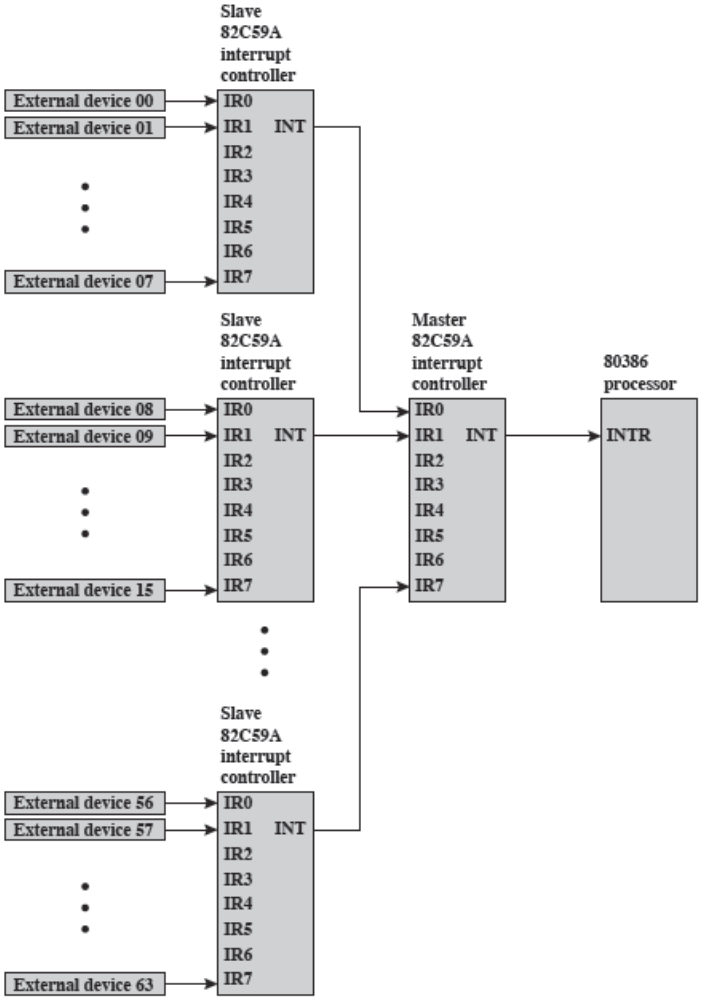

Figure 7.8 Use of the 82C59A Interrupt Controller

The figure below is that of the 8255 programmable I/O device.

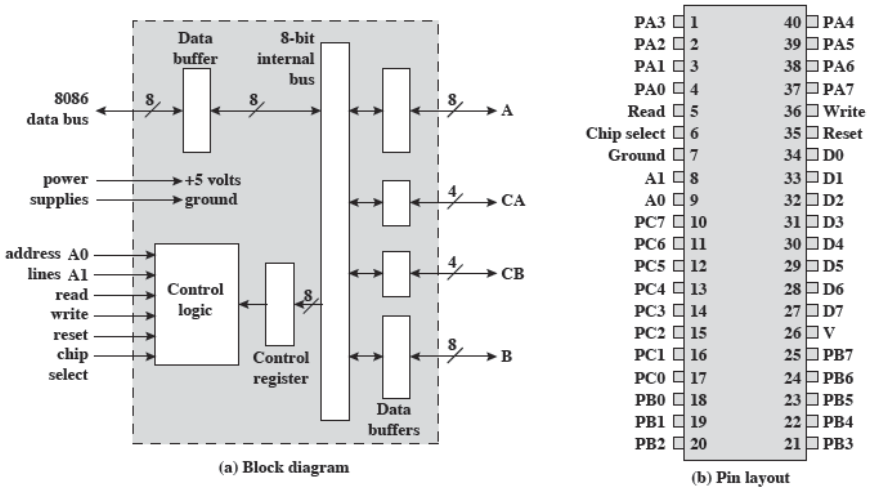

Figure 7.9 The Intel 82C55A Programmable Peripheral Interface

The disadvantages of interrupt driven I/O are t**ransfer speed is limited by processor speed and ties up processor doing I/O to Memory transfers.**

DMA
DMA is **Direct Memory Access. Processor relinquishes control of the address and data bus and DMA controller transfers data directly between and I/O device and memory.**

In cycle stealing **DMA controller gets busses and effectively steals a bus cycle.**

The processor uses the DMA module by **assigning an I/O job which includes the addresses involved and the number of bytes to be transferred. The processor then goes about its business and the DMA controller uses cycle stealing or simply takes over the bus directly to transfer the data directly into memory. When the job is complete the DMA controller signals the CPU usually by way of an interrupt.**

I/O transfers using DMA are considerably faster but CPU's now can use a bus nearly 100% of the time so there is some degradation in performance. But still it is much better (usually) than other forms of I/O.
Some examples where DMA transfers might be useful include: **Disk I/O, A/D converter, data to image processor.**

Direct Cache Access
Direct cache access a**llows DMA to access the last level of cache memory – usually L3.**

WiFi and gigabit Ethernet are fast enough that direct access to cache provides a significant speed increase.

To output data that is in the last level cache the processor establishes DCA and the last level cache transfers the data directly to the output port and simultaneously erases the data in the cache. If the data is needed in the future it must be read from memory back into the cache. For data coming into the processor the data can be written to the cache using DCA. The data can then be used by the CPU and written to main memory before it is erased. The cache then acts like an input buffer.

I/O Channel
The I/O channel and was invented by **IBM in the 60's with the mainframe.**

An I/O channel is a**n I/O module with its own CPU which may or may not have its own local memory.**

When using an I/O channel the **CPU does all I/O between itself and memory. The I/O processor transfers all data between the I/O devices and the memory. Thus, the CPU has no I/O instructions.**

There are two types of I/O Channel: **Selector Channel and a multiplexor channel. The selector channel is connected to multiple devices and it selects one for which it handles I/O transfers. The multiplexor multiplexes several devices and can do byte transfers to several devices in turn.**

External Interface
There are two types of external interfaces: **Parallel and serial. Parallel for high speed and serial for slower speed peripherals.**

The trend in recent years with regard to types of external interfaces has been to h**igh speed serial – mostly driven by convenience of use.**

When information is sent to an I/O module we must do the following: **1) IO Module requests permission to send/receive data. 2) Peripheral acknowledges request 3) Data is transferred 4) Peripheral acknowledges receipt of data. This is called *handshaking.***

Almost all I/O modules have buffers. **These allow data to be stored to compensate for the difference in timing between the peripheral device the CPU.**

USB
Universal Serial Bus – for slower speed devices and used by some for higher speed devices like external disk drives. USB 3.0 can run as high as 10 Gbps. This is a four wire connection that includes a 5 volt bus that defaults to half an amp. Information is transmitted in serial packets.

Thunderbolt is an example of a **multi-point peripheral connection developed by Intel/Apple. Combines data, video, audio, and power into a single high speed connection.**

SATA is **Serial Advanced Technology Attachment – a high speed serial bus used for disk system in desktop computers. Runs at 300 Mbps.**

Standard connector, data segment[42]

Pin #	Mating	Function
1	1st	Ground
2	2nd	A+ (transmit)
3	2nd	A− (transmit)
4	1st	Ground
5	2nd	B− (receive)
6	2nd	B+ (receive)
7	1st	Ground
—		Coding notch

WiFi and Ethernet are both serial communication channels. **Wifi tops out at 3.2 Gbps while Ethernet runs between 40 Gbps and 100 Gbps**