I. External devices
   A. External devices are not connected directly to the system bus because they have a wide range of control logics, as well as data transfer speeds and formats.
   B. Virtually all external devices have buffers, control signals, status signals, and data bits.
   C. Those that deal with other forms of energy have transducers that converts from non-electrical data to electrical data, (e.g. key press to ASCII in a keyboard), or electrical data to non-electrical data (e.g. bytes to light in a monitor).
   D. In the past, intra-system (less than 1 meter) connections were usually parallel, and inter-system were serial. Now almost all are serial.
      1. To convert from parallel to serial use a shift register.
      2. USB (universal serial bus) is a common standard for serial transmission with USB 3.0 transferring at 5Gb/s.

II. I/O Modules (south bridge and north bridge on a PC) on the motherboard provide the logic, buffers, error detection, and ports to communicate with the external devices on one side, and a system-type bus on the other. For external device interfaces, the modules have data, status, and control lines. For the system bus they have data, address, and control lines. The south bridge handles slower I/O devices and is connected to the north bridge rather than the system bus directly.

III. Programmed I/O
   A. Overview of Programmed I/O waits for the processor to query it.
   B. Four types of I/O commands: 1) control (e.g. rewind tape or clear screen), 2) test (i.e. reading status register), 3) read, and 4) write.

IV. Two modes of addressing are possible for I/O:
   A. Memory mapped, used by current chips including ARM, has the addresses for I/O within the main memory address space. Eliminates the need for special I/O instructions, but you lose part of the memory space.
   B. I/O mapped, used by Intel, has a separate memory space for I/O which require special I/O instructions, but no memory space lost (64K is minor nowadays though).

V. Interrupt-Driven I/O
   A. Interrupt processing proceeds in nine steps with the first five hardwired:
      1. Device controller or other system hardware issues an interrupt.
      2. Processor finishes execution of current instruction
      3. Processor signals acknowledgment of interrupt.
      4. Processor pushes PSW and PC onto stack.
      5. Processor loads new PC value from jump table based on interrupt.
      6. Interrupt service routine (ISR) saves remainder of process state information.
      7. ISR processes the interrupt.
      8. ISR restores the process state information.
      9. RTI instruction at the end of the ISR restores the old PSW and PC.
   B. Two design issues in dealing with multiple interrupts at the same time.
      1. How does the processor determine which device issued the interrupt?
      2. How does the processor determine which interrupt to process?
      3. Four solutions.
         a. Multiple interrupt lines into the processor. Number of lines must be limited, and will be fewer than devices. Order of checking the lines determines priority.
         b. Software poll by testing the status of each I/O module. Time consuming. Order of polling determines priority.
         c. Daisy chain (hardware poll, vectored). Devices share an interrupt request line, but granting is daisy chained through them. When a device receives the grant signal it requested, it places its unique ID or address on the data lines, called its vector, so this is called a vectored interrupt. Daisy chain order determines priority.
         d. Bus arbitration (vectored) is similar to daisy chain, except it relies on the arbitration system of the particular bus of the devices. Bus arbitration method determines priority.
   C. Intel 82C59A Interrupt Controller
      1. The Intel 80386 had a single Interrupt Request (INTR) line, and a single Interrupt Acknowledge (INTA) line. It relied on separate chip(s) to handle multiple interrupt requests.
      2. The 82C59A had 8 IRQ input lines and one output.
      3. Priority was set using a control register in the chip.
         a. Fully nested: based on pin number, IR0 first, IR7 last.
b. Rotating for devices of equal priority.
c. Special mask to inhibit interrupts from certain devices.
4. Can cascade chips to allow for 64 devices.

VI. Direct Memory Access (DMA)
A. Drawbacks of Programmed and Interrupt-Driven I/O arise because they involve the processor.
   1. I/O transfer limited by the speed with which the CPU tests and services the device.
   2. CPU has devote time to mundane I/O code for each transfer.
B. DMA Function is administered by a separate DMA module, usually in the South Bridge.
   1. It can transfer data between memory and devices without the CPU being involved.
   2. Cycle stealing = the DMA preempts the CPU to use the system bus for its own transfer needs. Done just before fetch instruction, fetch operand, and store result in the instruction cycle. Slows CPU access to RAM, but is more efficient than having a context switch.
   3. CPU uses command lines to DMA module to indicate: type of request (read or write), I/O device, start location in memory, number of bytes to be transferred. When DMA is done with a request it sends an interrupt to the CPU.

VII. The Evolution of the I/O Function
A. CPU directly controls a peripheral device.
B. CPU uses programmed I/O to interact with a separate I/O module.
C. CPU use interrupts to interact with a separate I/O module.
D. I/O module now has DMA.
E. I/O module becomes a processor in its own right with its own specialized instruction set (south bridge).
F. I/O module has local memory so it can interact with a large set of I/O devices with minimal CPU involvement.