Preview

- Input / Output
  - I/O Devices (categories, components)
  - Device Controllers
  - How CPU communicate with Device controller
    - Memory-Mapped I/O (Indirect)
    - Non-Memory-Mapped I/O (Direct)
    - Hybrid
  - Direct Memory Access (DMA)
  - Interrupt Revisited

Input / Output

- Operating System controls all I/O devices by
  - Issue commands to Devices (read/write)
  - Catch interrupts from Devices (when device is ready to read or write)
  - Handle error
- OS also provide interface between the devices and the rest of the system (through device driver)

Input / Output Devices

- I/O Devices can be roughly divided into two categories: **block devices**, **character devices, Others**
- **Block devices**
  - Block devices stores information in fixed-size blocks, each block has its own address.
  - Common block size range is from 512 Byte to 32 KB.
  - The essential property of a block device is possible to read or write each block independently. Need seek operation before read or write operation.
  - Ex) HDD, CD ROM, SSD, USB, …

Input /Output Devices

- **The Character devices**
  - A character device delivers (or accepts) a stream of characters, without regard to any block structure.
  - It is not addressable and does not have any seek operation.
  - ex) Printers, network interfaces, mouse
- Others
  - Clock – only cause interrupt at well-defined interval
  - Memory Mapped Screen

Device Controllers

- Most of I/O units consist of a mechanical components, an electronic component and device driver
  - **Mechanical Components** - Device itself
  - **Electrical components**
    - Device controller (adapter).
    - On personal computer, it takes the **form of a chip on the** parent board or a printed circuit card that can be inserted into a PCI expansion slot.
  - **Device Driver (software)** – interface between OS

Device Controllers

- A **device controller** is a part of a computer system.
- A device's controller plays an important role in the operation of that device; it functions as a bridge between the device and the operating system.
- Any device connected to the computer is connected by a plug and socket, and the socket is connected to a device controller.
- Each device controller has a local buffer and a command register. It communicates with the CPU by interrupts.

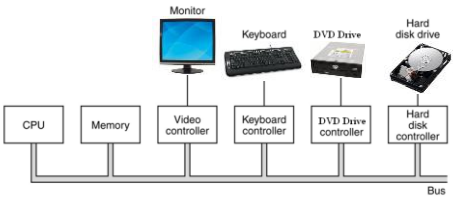
Device Controllers

Hard disk and controller

Device Controllers

- The **Device Controller** receives the data from a connected device and stores it in local buffer inside the controller before sending through bus.
- Then it communicates the data with a **Device Driver**.
- For each **device controller** there is an equivalent **device driver** which is the standard interface through which the device controller communicates with the Operating Systems through interrupts.

Device Controllers



Components of Personal Computer

Device Controllers

- The interface between the controller and device is a very low-level interface.
  - Ex) Disk Controller (512 byte per block)
    - Disk Drive send a serial bit streams start with
      - Preamble -cylinder and sector number, sector size,…
      - 4096 bit (512 × 8) (data)
      - Check sum for error checking
    - Disk Controller's job
      - Convert the serial bit stream into a block of bytes
      - Error checking and error correction
      - Send to main memory.

How CPU communicate with Controller

- Each controller has a few registers used for communicating with CPU.
- By writing into these registers, OS can command the device
  - To deliver data (write)
  - To accept data (read)
  - To switch itself on or off (its ready to receive or send)
- By reading from these registers, OS can learn
  - What is the device state?
    - Whether it is ready to accept new command or not

How CPU communicate with Controller

- Devices have a data buffer in device controller for OS can read from and write to.
  - Ex) the common way for computers to display pixels on the screen is to have a video RAM, which is basically just a data buffer available for programs or OS to write into.
- The issue is how the CPU communicates with control registers and the device buffers!!!
  - Non Memory-Mapped I/O
  - Memory-Mapped I/O
  - Hybrid scheme

How CPU communicate with Controller
(Non Memory Mapped I/O)

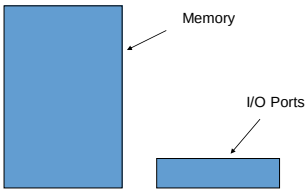
Non Memory-Mapped I/O

- Communicate directly through controller's register and buffer.
- Each control register is assigned an I/O port number (8 bit or 16 bit number)
- The set of all the I/O ports form the I/O port space, only the OS can access it.
- By using special I/O instruction, the CPU can read in control register and can write the content of a register

Ex)
IN REG, PORT; the CPU read in control register PORT and store the result in CPU REG.
OUT PORT, REG; the CPU write the contents of REG to a control register

How CPU communicate with Controller
(Non Memory Mapped I/O)

- In non-mapped I/O scheme, the address spaces for memory and I/O are different



How CPU communicate with Controller
(Non Memory Mapped I/O)

- In Non Memory-Mapped Scheme, different instruction sets are used for communication between the CPU and memory and between the CPU and I/O device controller registers

Ex)

Port number

IN R0, 4 ; reads the contents of I/O port 4 and puts it in R0

MOV R0, 4; reads the contents of memory word 4 and put in R0

Memory address

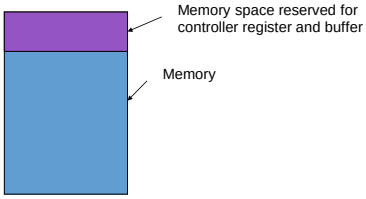
How CPU communicate with Controller
(Memory Mapped I/O)

Memory-Mapped I/O

- Indirect communication through part of memory!
- Each control register is assigned a unique memory address to which no memory is assigned. (for buffer and register)
- The assigned addresses are usually at the top of the address space.

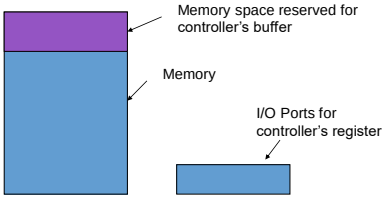
- **A hybrid scheme**, with memory-mapped I/O for data buffers and separate ports for the control registers is used in the **Pentium**.

How CPU communicate with Controller
(Memory Mapped I/O)



Pure Memory-Mapped I/O

How CPU communicate with Controller
(Hybrid Scheme)



Hybrid Scheme

How CPU communicate with Controller
(Hybrid Scheme)

- **Hybrid scheme**
  When the CPU wants to read a word, either from memory (Data from buffer) or from an I/O port (command or status from register),
  - The CPU puts the address on the bus's address line
  - Then, asserts a read signal on the bus's control line.
  - The second signal line is used to tell whether I/O space or memory space is needed.
  - Either I/O device or memory response to the request based on the second signal line.

How CPU communicate with Controller
(Memory Mapped I/O)

Pure memory-mapped I/O – Memory spaces are used for data buffer and registers of controller.

- Every memory module and every I/O device compares the address lines to the range of addresses that it service.
- If the address falls in its range, it responds to the request.

How CPU communicate with Controller
(Memory Mapped I/O)

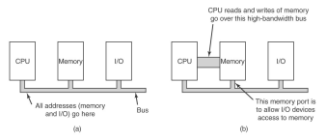
Advantage with Memory-Mapped I/O

- Since, Non-Memory Mapped I/O system need a special instruction set, device driver cannot be written by a high level language such as C or C++.
- But with Memory-Mapped I/O, device driver can totally written by a high-level language.
- No special protection mechanism is needed to user processes from performing I/O. – Operating system just check whether user process try to use controller's space or not.
- Every instruction (that can reference memory) can also reference control register

How CPU communicate with Controller
(Memory Mapped I/O)

Disadvantage with Memory-Mapped I/O

- The trend in modern personal computer is to have a dedicated high-speed memory bus between memory and CPU.
- I/O devices have no way of seeing memory addresses as they go by on the memory bus, so they have no way of responding – need extra mechanism to solve this problem.



Direct Memory Access (DMA)

- Whether CPU have memory-mapped I/O or not, it needs to address the device controllers to exchange data with them.
- The CPU can request data from an I/O controller one byte at a time.
- If CPU need wait for completion of data transfer, it waste the CPU time.
- Direct Memory Access (DMA) controller can handle I/O independent from the CPU.
- DMA is a capability provided by some computer bus architectures that allows data to be sent directly from an attached device (such as a disk drive) to the memory.
- The CPU is freed from involvement with the data transfer, thus speeding up overall computer operation.

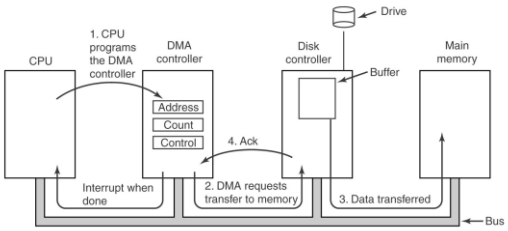
Direct Memory Access (DMA)

Ex) Disk read from Disk to Memory without DMA controller

1. The disk controller reads data bit by bit from the disk until an entire block is in the controller's buffer.
2. It check checksum to verify any error.
3. If there is no error, controller causes an interrupt to get a service from operating system.
4. The service transfer data from controller's buffer to main memory through bus line

Direct Memory Access (DMA)

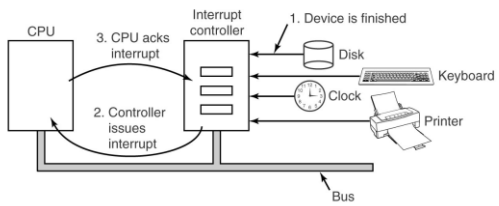
- How to reads a block from the disk with DMA?



Interrupt Revisited

- When a I/O device finish its job, it cause interrupt.
- The I/O device send a signal for interrupt service through bus line
- This signal is detected by the interrupt controller chip on the parentboard.
- If no other interrupt are pending, the interrupt controller processes the interrupt immediately. If interrupt controller is busy to handle other interrupt, the device is just ignored for the moment. And continues to assert an interrupt signal on the bus until it is serviced by the CPU.

Interrupt Revisited



Interrupt Revisited

- Interrupt controller send a number to the CPU which is used for index of interrupt vector.
- The signal from the Interrupt controller cause the CPU to stop what it is doing and it start to handle the interrupt.
- Interrupt vector (table) has pointers where interrupt service procedures are located.
- Program counter save the location where the interrupt service procedure is located.
- Shortly after it starts running, the interrupt service procedure acknowledges the interrupt by writing a certain value to one of the interrupt controller's I/O ports which tells the controller that it is free to issue another interrupt.