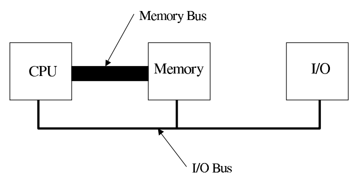
Input / Output

CS 537 – Introduction to Operating Systems

Basic I/O Hardware

- Many modern system have multiple busses
  - memory bus
    - direct connection between CPU and memory only
    - high speed
  - I/O bus
    - connection between CPU, memory, and I/O
    - low speed

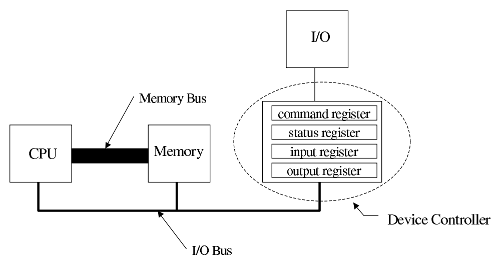
Basic I/O Hardware



Device Controllers

- I/O devices have controllers
  - disk controller, monitor controller, etc.
- Controller manipulates/interprets electrical signals to/from the device
- Controller accepts commands from CPU or provides data to CPU
- Controller and CPU communicate over I/O ports
  - control, status, input, and output ports

Adding the Device Controller



Communicating with I/O

- Two major techniques for communicating with I/O ports
  1. Special Instructions
  2. Memory Mapped I/O

Special Instructions

- Each port is given a special address
- Use an assembly instruction to read/write a port
- Examples:
  - OUT *port*, *reg*
    - writes the value in CPU register *reg* to I/O port *port*
  - IN *reg*, *port*
    - reads the value at I/O port *port* into CPU register *reg*

Special Instructions

- Major Issues
  1. communication requires the programmer to use assembly code
     - C/C++/Java and other high-level languages only work with main memory
  2. how to do protection?
     - users should have access to some I/O devices but not to others

Memory Mapped I/O

- I/O ports are mapped directly into memory space
- Use standard load and store instructions
- Examples:
  - ST *port*, *reg*
    - stores the value at CPU register *reg* to I/O port *port*
  - LD *reg*, *port*
    - reads the value at I/O port *port* into CPU register *reg*
- Can now use high-level language to communicate directly with I/O devices

Memory Mapped I/O

- Requires special hardware to map specific addresses to I/O controllers instead of memory
  - this can be tricky with 2 busses
- Part of the address space is now unusable
  - this does not mean part of physical memory is unusable

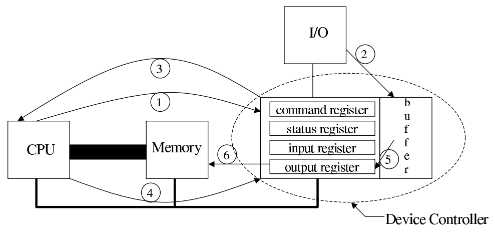
Memory Mapped I/O

- Major Issues
  1. caching of memory values in CPU
     - do not want to cache I/O port values in memory because they may change
     - imagine a busy wait – the value it is checking will never change if it is cached
     - hardware must allow some values to not be cached
  2. how to do mapping of addresses to ports
     - with two busses this can be tricky
     - need to do some kind of "snooping" on address bus

Transferring Data (read request)

1. CPU issues command to I/O device controller to retrieve data
   - this is slow
2. Device places data into controller buffer
3. Device controller interrupts the CPU
4. CPU issues command to read a single byte (or word) of data from controller buffer
5. Controller places data into data register
6. CPU reads data into memory
7. Repeat steps 4, 5 and 6 until all data is read

Transferring Data (read request)



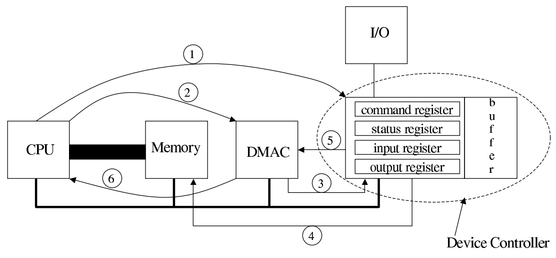
Direct Memory Access

- Problem with previous approach:
  - if CPU must issue all commands to I/O devices, it can't do anything else
  - it may take a long time to transfer all data
- Solution:
  - allow I/O to communicate directly with memory (Direct Memory Access – DMA)

DMA Controller

- Need a special hardware device to communicate between CPU and I/O
- DMAC has several registers
  1. memory address to read/write from/to
  2. I/O port to communicate with
  3. number of bytes to transfer
  4. direction of transfer (read or write)

Basic Hardware with DMA



Basic Hardware with DMA

1. CPU instructs device controller to read
   - device controller informs CPU when the read is complete – data in controller buffer
2. CPU programs the DMAC
3. DMAC asks for byte (or word) of data to be written to memory
4. Device controller writes data to memory
5. Device gives acknowledgement to DMAC
   - repeat steps 3, 4, and 5 until all data is transferred
6. DMAC interrupts CPU when transfer complete

More on DMAC's

- Some DMA controllers have multiple "channels"
  - can allow communication with more than one device simultaneously
- Some systems choose not to use DMA at all
  - why?