I/O Management
Intro

Chapter 5

I/O Devices

- There exists a large variety of I/O devices:
  - Many of the with different properties
  - They seem to require different interfaces to manipulate and manage them
    - We don’t want a new interface for every device
    - Diverse, but similar interfaces leads to code duplication
- Challenge:
  - Uniform and efficient approach to I/O

Categories of I/O Devices (by usage)

- Human readable
  - Used to communicate with the user
  - Printers, Video Display, Keyboard, Mouse
- Machine readable
  - Used to communicate with electronic equipment
  - Disk and tape drives, Sensors, Controllers, Actuators
- Communication
  - Used to communicate with remote devices
  - Ethernet, Modems, Wireless

Differences that Impact I/O Device Handling

- Data rate
  - May be differences of several orders of magnitude between the data transfer rates

Sample Data Rates

Device	Data rate
Keyboard	10 bytes/sec
Mouse	100 bytes/sec
56K modem	7 KB/sec
Telephone channel	8 KB/sec
Dual ISDN lines	16 KB/sec
Laser printer	100 KB/sec
Scanner	400 KB/sec
Classic Ethernet	1.25 MB/sec
USB (Universal Serial Bus)	1.5 MB/sec
Digital camcorder	4 MB/sec
IDE disk	5 MB/sec
40x CD-ROM	6 MB/sec
Fast Ethernet	12.5 MB/sec
ISA bus	16.7 MB/sec
EIDE (ATA-2) disk	16.7 MB/sec
FireWire (IEEE 1394)	50 MB/sec
XGA Monitor	60 MB/sec
SONET OC-12 network	78 MB/sec
SCSI Ultra 2 disk	80 MB/sec
Gigabit Ethernet	125 MB/sec
Ultrium tape	320 MB/sec
PCI bus	528 MB/sec
Sun Gigaplane XB backplane	20 GB/sec

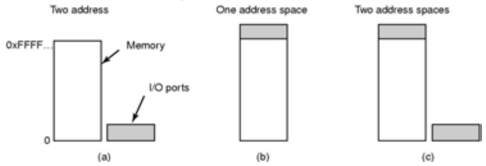
Differences that Impact I/O Device Handling

- Application
  - Disk used to store files requires file-management software
    - May provide feature specific to function, e.g. non-volatile RAM.
  - Disk used to store virtual memory pages needs special hardware and software to support it
  - Terminal used by system administrator may have a higher priority

Differences that Impact I/O Device Handling

- Complexity of control
- Unit of transfer
  - Data may be transferred as a stream of bytes for a terminal or in larger blocks for a disk
- Data representation
  - Encoding schemes
- Error conditions
  - Devices respond to errors differently

Accessing I/O Controllers

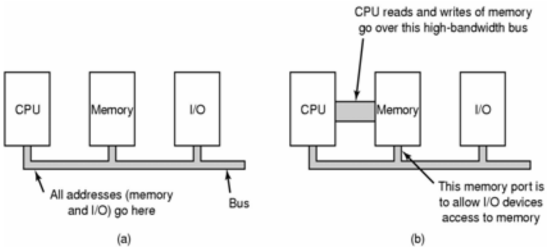


a) Separate I/O and memory space
  - I/O controller registers appear as I/O ports
  - Accessed with special I/O instructions

b) Memory-mapped I/O
  - Controller registers appear as memory
  - Use normal load/store instructions to access

c) Hybrid
  - x86 has both ports and memory mapped I/O

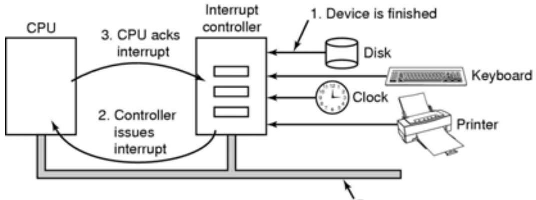
Bus Architectures



(a) A single-bus architecture

(b) A dual-bus memory architecture

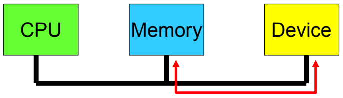
Interrupts Revisited



- Devices connected to an *Interrupt Controller* via lines on an I/O bus (e.g. PCI)
- Interrupt Controller signals interrupt to CPU and is eventually acknowledged.
- Exact details are architecture specific.

Direct Memory Access

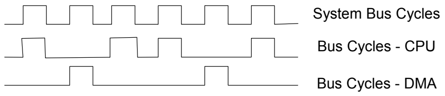
- Takes control of the bus from the CPU to transfer data to and from memory over the system bus
- Reduced number of interrupts occur
  - No expensive context switches



DMA

- Cycle stealing is used to transfer data on the system bus
  - The instruction cycle is suspended so data can be transferred
  - The CPU pauses one bus cycle
    - CPU Cache can hopefully avoid such pauses by hide DMA bus transactions
  - Cycle stealing causes the CPU to execute more slowly
    - Still more efficient than CPU doing transfer itself

Very Simplified Model of Cycle Stealing



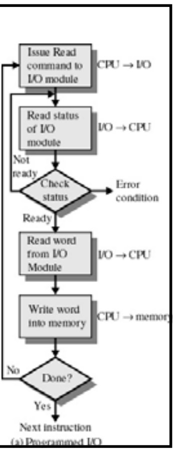
DMA

- Commonly *burst-mode* is used
  - CPU uses several consecutive cycles to load entire cache line
  - DMA writes (or reads) a similar sized burst
  - Reason: More efficient (less cycles overall) to transfer a sequence of words than a word at a time.
    - No bus arbitration, read/write setup, or addressing cycles.
- Number of required busy cycles can be cut by
  - Path between DMA module and I/O module that does not include the system bus

---

Programmed I/O

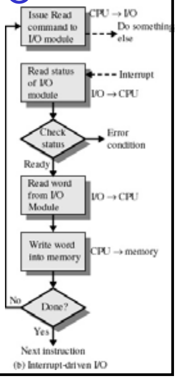
- Also called *polling*, or *busy waiting*
- I/O module (controller) performs the action, not the processor
- Sets appropriate bits in the I/O status register
- No interrupts occur
- Processor checks status until operation is complete
  - Wastes CPU cycles



(a) Programmed I/O

Interrupt-Driven I/O

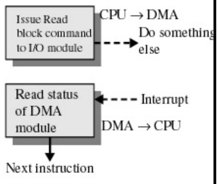
- Processor is interrupted when I/O module (controller) ready to exchange data
- Processor is free to do other work
- No needless waiting
- Consumes a lot of processor time because every word read or written passes through the processor



(b) Interrupt-driven I/O

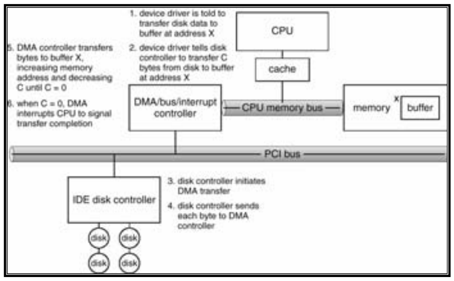
Direct Memory Access

- Transfers a block of data directly to or from memory
- An interrupt is sent when the task is complete
- The processor is only involved at the beginning and end of the transfer



(c) Direct memory access

The Process to Perform DMA Transfer

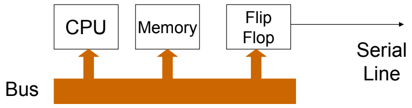


1. device driver is told to transfer disk data to buffer at address X
2. device driver tells disk controller to transfer C bytes from disk to buffer at address X
3. disk controller initiates DMA transfer
4. disk controller sends each byte to DMA controller
5. DMA controller transfers bytes to buffer X, increasing memory address and decreasing C until C = 0
6. when C = 0, DMA interrupts CPU to signal transfer completion

---

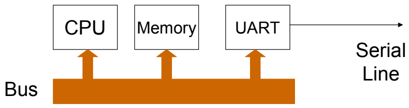
Evolution of the I/O Function

- Processor directly controls a peripheral device
  - Example: CPU controls a flip-flop to implement a serial line



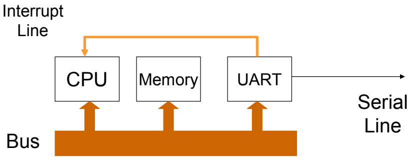
Evolution of the I/O Function

- Controller or I/O module is added
  - Processor uses programmed I/O without interrupts
  - Processor does not need to handle details of external devices
  - Example: A Univeral Asynchronous Receiver Transmitter
    - CPU simply reads and writes bytes to I/O controller
    - I/O controller responsible for managing the signalling



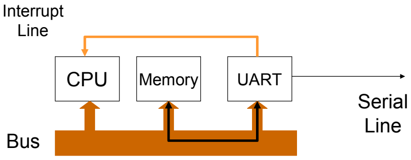
Evolution of the I/O Function

- Controller or I/O module with interrupts
  - Processor does not spend time waiting for an I/O operation to be performed



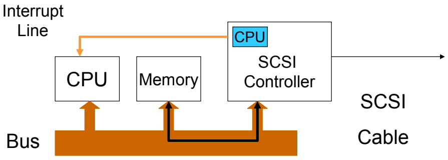
Evolution of the I/O Function

- Direct Memory Access
  - Blocks of data are moved into memory without involving the processor
  - Processor involved at beginning and end only



Evolution of the I/O Function

- I/O module has a separate processor
  - Example: SCSI controller
    - Controller CPU executes SCSI program code out of main memory



Evolution of the I/O Function

- I/O processor
  - I/O module has its own local memory, internal bus, etc.
  - Its a computer in its own right
  - Example: Myrinet Multi-gigabit Network Controller