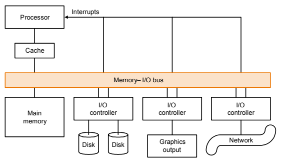
Interfacing Processors and Peripherals

- I/O Design affected by many factors (expandability, resilience)
- Performance:
  - — access latency
  - — throughput
  - — connection between devices and the system
  - — the memory hierarchy
  - — the operating system
- A variety of different users (e.g., banks, supercomputers, engineers)



1

I/O Importance

- What would happen if we have
  - PC with terminal output instead of flat panel display
  - Fast 3.06GHz Pentium-4 machine with 20MB hard drive
  - Digital color pictures with only black-white printers to print them
  - Internet without enough storage for image, audio, or video
- Important but neglected

  "The difficulties in assessing and designing I/O systems have often relegated I/O to second class status"

  - Design considerations are dramatically different among different types of I/O device
  - R&D efforts are very specialized
- We won't be looking at I/O in much detail
  - Read Chapter 8
  - Keep I/O issues in mind for future courses (OS, networking, etc.) or applications you may meet

2

I/O Devices

- **Very diverse devices**
  - **— behavior (i.e., input vs. output vs. both)**
  - **— partners (who is at the other end?)**
  - **— data rates**

Device	Behavior	Partner	Data rate (KB/sec)
Keyboard	input	human	0.01
Mouse	input	human	0.02
Voice input	input	human	0.02
Scanner	input	human	400.00
Voice output	output	human	0.60
Line printer	output	human	1.00
Laser printer	output	human	200.00
Graphics display	output	human	60,000.00
Modem	input or output	machine	2.00-8.00
Network/LAN	input or output	machine	500.00-6000.00
Floppy disk	storage	machine	100.00
Optical disk	storage	machine	1000.00
Magnetic tape	storage	machine	2000.00
Magnetic disk	storage	machine	2000.00-10,000.00

3

Communication with I/O Devices

- Talking with device: Memory-mapped I/O vs. special I/O instructions
  - Reading and writing to devices usually requires several steps
- Status Registers
  - Hold information pertaining to the state of the device
  - Done bit or Error bit, etc.
  - May also be written to for notifying device when the data input is ready
- Data Registers
  - Buffers for information
  - Examples: character to be printed, Ethernet packet, etc.
  - Some are only readable, others are only writeable. Sometimes they are both R/W.
- User I/O is managed by the supervisor (kernel) level, since the device address space is not usually available to a user

How can I/O Interact with processor, program, and OS?

4

I/O Polling

CPU checks and reads and writes device buffers:

- Processor must check whether or not I/O device has new meaningful information
- Large overhead costs
- Still sees some use though with very slow devices that are routinely used (e.g. mouse)

Example: A 500MHz processor with the following I/O devices
- Mouse that must be polled 30 times per second
- Floppy disk transferring data to processor in 2-byte units with transfer rate 50KB/sec
- Hard disk transferring data in 16-byte chunks at 4MB/sec, with 5% busy time
- Polling overhead is 400 cycles

How frequent is the polling? How much time does CPU spend on I/O?

5

Interrupt-driven I/O

CPU asks devices to let it know when they are read

- I/O device will notify processor by way of interrupt to request services
- Not synchronous with instructions
- Vectored Interrupts or EPC, so CPU knows what's happening
- Can have various interrupt levels to show priority

Review the example: A 500MHz processor with the following I/O devices
- Floppy disk transferring data to processor in 2-byte units with transfer rate 50KB/sec
- Hard disk transferring data in 16-byte chunks at 4MB/sec, with 5% busy time
- Each interrupt costs 500 cycles

How much time does CPU spend on I/O? How about mouse?

6

Direct Memory Access (DMA)

Memory/Device data transfers without constant use of the processor:

- Processor must inform DMA of operation to perform along with the various parameters (e.g. device address, source address, destination address, bytes to transfer, …)
- DMA starts the transfer and controls the bus, performing the requested operation
- When the operation is done, the DMA controller sends an interrupt to the CPU to let it know the status of the transfer

Review the example: A 500MHz processor with the following I/O devices

- – Hard disk transferring data in 16-byte chunks at 4MB/sec, with 100% busy time
- – Setting up DMA takes 1000 cycles, finishing up takes 500 cycles

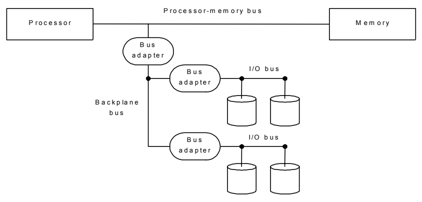
There are more complexities:

- There can be many DMA's attached to the same bus
- Difficulties with virtual address translation
  - – Coherency problem

I/O Example: Buses

- Shared communication link (one or more wires)
- Difficult design:
  - — may be bottleneck
  - — length of the bus
  - — number of devices
  - — tradeoffs (buffers for higher bandwidth increases latency)
  - — support for many different devices
  - — cost
- Types of buses:
  - — processor-memory (short high speed, custom design)
  - — backplane (high speed, often standardized, e.g., PCI)
  - — I/O (lengthy, different devices, standardized, e.g., SCSI)
- Synchronous vs. Asynchronous
  - — use a clock and a synchronous protocol, fast and small but every device must operate at same rate and clock skew requires the bus to be short
  - — don't use a clock and instead use handshaking

Some Example Problems



Bus Arbitration: Who will get the bus?

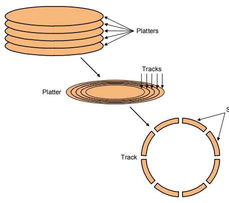
- – bus masters vs. bus slaves
- – daisy chain arbitration (not very fair)
- – centralized arbitration (requires an arbiter), e.g., PCI
- – self selection, e.g., NuBus used in Macintosh, HPIB
- – collision detection, e.g., Ethernet *mouse?*

Some performance examples

Let's look at some examples from the text

- We are talking about synchronous bus, and are not going to include bus acquisition time
- Parameters:
  - – Processor runs at 200MHz, clock cycle time 5 nsec
  - – Address transfer takes 1 cycles
  - – Memory reads first 4 word block in 200 nsec (40 cycles)
  - – Reads successive 4 word blocks in 20 nsec (4 cycles)
  - – transfer of a 4 word block takes 10 nsec (2 cycles)
  - – Successive read and transfers can be overlapped
  - – There should be at least two cycles delay between two transfers
- Consider two design alternatives
  - – Transfer 4-word unit: 1 + 40 + 2 + 2 = 45 cycles per unit
  - – Transfer 16-word unit: 1 + 40 + 4 + 4 + 4 + 2 + 2= 57 cycles per unit
- To transfer 256 words using
  - – 4 words at a time will take 64 * 45 = 2880 cycles
  - – 16 words at a time will take 16 * 57 = 912 cycles

I/O Example: Disk Drives



To access data we need the following steps

- **– Set DMA controller with address and count of words to be read**
- **– Ask disk to be on the right track and right position**
- **– Transfer Data to/from memory**

DISK I/O Time

- – **Preparation time: OS prepares/delivers the transaction controller**
  - 1-2 ms.
- – **Seek: position head over the proper track**
  - On average takes 8 to 10 ms.
- – **Rotational latency: wait for desired sector (.5 / RPM)**
  - At 7200 RPM, time for 0.5 revolution is 4.2 ms.
- – **Transfer time: grab the data and transfer to memory**
  - At 4MB/sec, 4Kb takes 4KB/(4MB/sec) = 1 ms.
- – **Total time: 14.2 ms to 17.2ms (min to max)**
  - Say 15ms on average
- – **Keeps bus busy for 1 ms out of 15 ms**

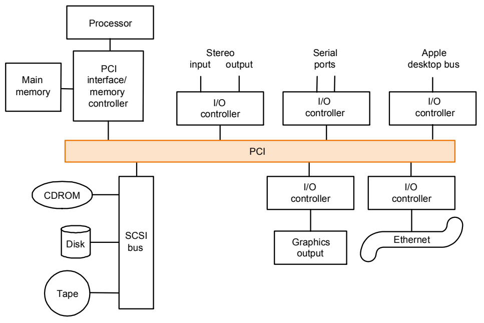
I/O Design Example

We are designing a small server with intensive I/O:

- CPU can run up to 300M instructions per second
- The server program generates disk reads every 100K instructions
- Memory backplane bus can sustain 100MB/s transfer rate
- SCSI-2 controllers can sustain 20MB/s transfer rate and support up to seven disks
- Disk drives can sustain 5MB/s transfer rate with 10ms latency for seek plus rotational latency
- We can use multiple SCSI-2 controllers and drives to maximize performance

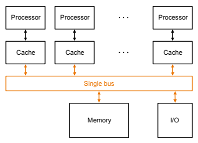
Where is the bottleneck? How many disks and controllers are needed?

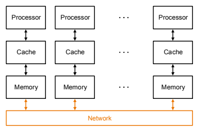
Real Stuff: Desktop I/O System on Macintosh 7200



What's Not Covered: Multiprocessors

- Idea: create powerful computers by connecting many smaller ones
  - good: works for timesharing (better than supercomputer)
  - bad: its really hard to write good parallel programs
- Read Chapter 9





Concluding Remarks

- **Evolution vs. Revolution**

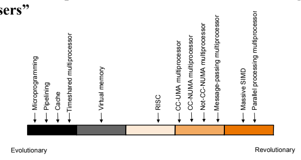
  "More often the expense of innovation comes from being too disruptive to computer users"

  "Acceptance of hardware ideas requires acceptance by software people; therefore hardware people should learn about software. And if software people want good machines, they must learn more about hardware to be able to communicate with and thereby influence hardware engineers."

Concluding Remarks

- **Evolution vs. Revolution**

  "More often the expense of innovation comes from being too disruptive to computer users"



  "Acceptance of hardware ideas requires acceptance by software people; therefore hardware people should learn about software. And if software people want good machines, they must learn more about hardware to be able to communicate with and thereby influence hardware engineers."