Review:
Break:
Vector Processing
Evening Midterm?, no final
HW assignment:
7.3, 7.7

I just joted down the notes and took it from there

First lecture
Importance of I/O
Charactersistics of I/O
How to trade off latency and bandwidth
(short lecture due to fire alarm)

Second lecture
Formulas
Disk I/O & disk arrays
Disk pitfall

Third Lecture
Graphics
Networks (multiprocessor vs network)
Buses

Today:
1/2 lecture

OS is to I/O as compiler is to ISA

IBM I/O system
Can divide into control and data path
Data Fig 9.33 on page 548
Top down: memory connected to channels via speed matching buffers (two 8 bytes buffers/channel)

Storage dirtector connected to channels
(allows many channels to talk to many devices)

2 or 4 path strings to connect to disks
(top is Head of string, all commnuication passes through it)

Multiple paths allow multiple transfers simultaneously

12 step Fig 5.34 on page 550

1. The user sets up a data structure in memory containing the operations that should occur during this I/O event. This data structure is termed an *I/O control block*, or IOCB, which also points to a list of channel control words (CCWs). This list is called a *channel program*.
2. The CPU executes a `START SUBCHANNEL` instruction. The actual request is defined in the channel program. A channel program to read a record might look like Figure 9.35.
3. The I/O processor uses the control wires of one of the channels to tell the storage director which disk is to be accessed and the disk address to be read. The channel is then released.
4. The storage director sends a SEEK command to the head-of-string controller and the head-of-string controller connects to the desired disk, telling it to seek to the appropriate track, and then disconnects. The disconnect occurs between CCW2 and CCW3 in Figure 9.35.

Arm starts seeking for correct track
During disconnect/seek hierachy can do other things

Reconnect: when head on right track and data nearly under head (RPS)

5. When the disk completes the seek and rotates to the correct position, it contacts the head-of-string controller.
6. The head-of-string controller looks for a free storage director to send the signal that the disk is on the right track.
7. The storage director looks for a free channel so that it can use the control wires to tell the I/O processor that the disk is on the right track.
8. The I/O processor simultaneously contacts the storage director and I/O device (the IBM 3380 disk) to give the OK to transfer data, and tells the channel controller where to put the information in main memory when it arrives at the channel.

Path established

9. When the disk is ready to transfer, it sends the data at 3 megabytes per second over a bit-serial line to the storage director.
10. The storage director collects 16 bytes in one of two buffers and sends the information on to the channel controller.
11. The channel controller has a pair of 16-byte buffers per storage director and sends 16 bytes over a 3-MB or 4.5-MB per second, 8-bit-wide datapath to the speed-matching buffers.

12. The speed-matching buffers take the information coming in from all channels. There are two 8-byte buffers per channel that send 8 bytes at a time to the appropriate locations in main memory.

Data in memory!

Fig 9.36 shows improvements over time
370 XA: different path to disk (1-4) and back (5-12): Dynamic Path Reconnect: -17% reduction in response time
    savings: waiting for channel and waiting for disks (queue)
Device Level Select
    better microcode and use of internal paths : -9%
Cache
    32 MB write through cache: -41%
        reduced rotational latency, seek, queuing delays

Pitfalls
    I/O not important for supercomputers
p. 555 9.37
    I/O processor vs CPU
    Media vs drive

Vector

Even higher performance for special category
Purchasing on peak MFLOPS on a program vs sustained

What if data much larger than caches?
If operation is on an array, why issue an instruction every cycle to say its the same operation?
If operation is on an array, why not tell the computer so that it doesn't have to worry about data harzards between vector elements?
If operation is on an array, why not tell the memory system so that it can use interleaving to achieve higher memory bandwdith?
If operation is an array, why include a branch?

2 choices: mem-mem or vec-reg

DLXV (like Cray 1)
p. 354, Fig 7.1
8 vector registers, each 64x64b
5 vector functional units
vector load/store unit
p. 356 7.3 DLXV instruction set

p. 357: SAXPY in DLX vs DLXV

Startup time: latency of a fauncitonal unit (get first result)
initiation rate: time per result from a function unit (usually 1, can be 2)

vector operation time of n = startuptime + n*initiation rate

Fig 7.4 on 339 shows variations

DLXV Fig 7.5 on page 361

Memory: DLXV assumes 12 clock cycles
add extra 4 clock systle if there is a dependenence
multiple memory banks
Similar to interleaving, except get full memory bandwidth
Fig 7.6 p. 363

Vector length and stride
VLR: vectors shorter than 64
Strip mining if greater; see p. 364

Vector stride:

What if 2D array?

2 Choices for layout in memory

7.10 p. 367 (Only Fortran uses column major order)

stride: distance between vector elements in memory to be merged into a single vectgor

unit stride works best with memory; nono-unit stride can be a problem