CS350 Final Review

I/O

- A device driver interacts with a device by reading/writing to the device’s command, status and data registers

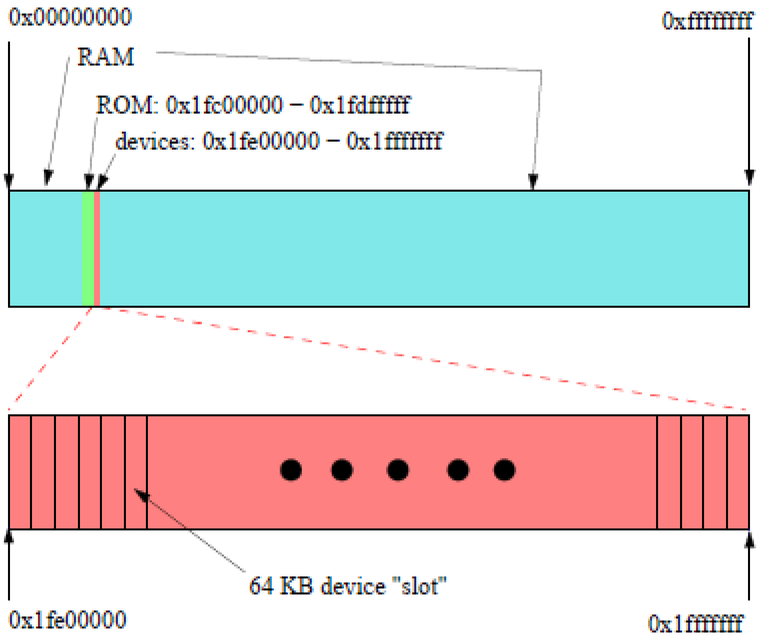
- For example: Sending data to an output device
  - Write data to the devince’s data register
  - Write “output” command to the device’s command register
  - Keep reading the device’s status register until it is “completed”
  - Reset the status register so it can send the next piece of data

Interrupts

- Polling is not always desirable (why?)
- Device can instead raise an interrupt when it is finished with the command
- Device driver can block on a semaphore/condition variable while waiting for the interrupt

Accessing Devices

- Special I/O instructions (x86)
- Memory-mapped I/O
  - Device registers have a memory address
  - Use standard "load" to read data from a device, and "store" to write data to the device



Direct Memory Access

- The "load" instruction read data one word at a time
  - This can be slow if we are trying to transfer a large amount of data
  - Also requires the CPU to perform the transfer

- Direct Memory Access (DMA) is a method to allow a separate controller to transfer data between a device and memory
  - DMA controller becomes bus master and performs the data transfer on behalf of the CPU
  - Instead of writing data into the data register, write the source/destination memory address into the address register

Disk

- Service time is the sum of
  - Seek time: Moving the disk head to the right track/cylinder
  - Rotational latency: Waiting until the sector spins to disk head
  - Transfer time: Wait until all of the desired sectors spin past the disk head

- Know how to calculate these values given disk specifications
  - Edge-to-edge seek latency: 20 ms
  - Number of tracks/cylinders = 10
  - 7200 RPM = 120 RPS = 8.3 ms per rotation
  - What is the average service time to read 1 sector that is 2 tracks away?

Disk Head Scheduling

- Given a queue of requests, create a schedule that minimizes seeks
- FCFS
  - Simple and fair
- Shortest Seek Time First (SSTF)
  - Lower seek time than FCFS
  - May cause starvation
- Elevator algorithm
  - Moves only in one direction until it reaches the edge of the disk
  - No starvation
  - Fewer seeks than FCFS

Scheduling

- Want to minimize turnaround time
  - Arrival time
  - Run time (length of the job)
  - Start time
  - Finish time
- Turnaround time: f – a
- Response time: s - a

Schedulers

- FCFS
- Round-robin
- Shortest Job First (SJF)
- Shortest Remaining Time First (SRTF)

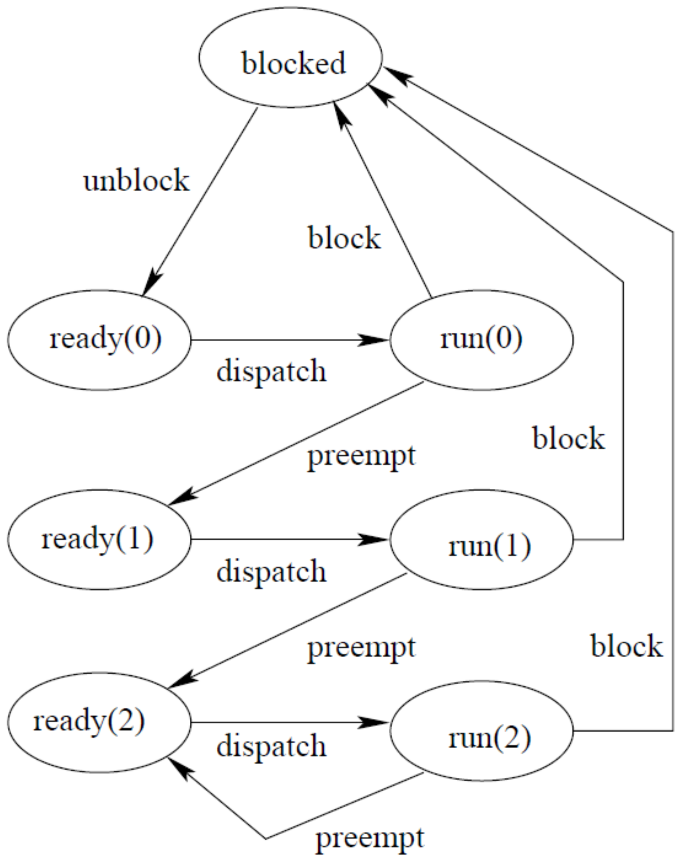
CPU Scheduling

- Thread arrives when it is ready
- Runtime is the amount of time a thread runs until it blocks or finishes

- We can estimate a threads runtime by determining its tendency to block
  - A thread that blocks often is likely an interactive thread and has short runtime
  - A thread that doesn’t block is likely a computationally intensive thread with a long runtime
  - Should we assign higher priority to interactive or non-interactive thread?

Multi-Level Feedback Queue

- Assigns a priority level to a thread based on its previous behaviour



Multi-Core Scheduling

- Queue per core first single ready queue
- Contention and Scalability
- Cache affinity
- Load imbalance

File Systems

- File
  - Identified by an i-number
  - An i-number is the index to an i-node array
- i-node
  - Stores meta-information about the file
    - File type, permissions, length, last modified, etc.
  - Stores block pointers
    - Why do we need indirect block pointers?
    - What does an indirect block pointer point to?
- Bitmap nodes
  - Stores the availability of inodes and data blocks
- Supernode
  - Meta-information of the entire file system

    Number of inodes, where the inode blocks begin, etc.

File Systems

- Given a file offset, know how to determine which block pointers must be accessed to determine the block location
- Understand design decisions for a particular file system
  - Why not just have indirect pointers?
  - In what situations is an indexed file system more appropriate than a chained file system?
- How are directories implemented?
  - What is a hard link? Soft link?
  - What is referential integrity?

File Systems

- Given a file (/foo/bar), know all of the inodes/data blocks/bitmap nodes that need to be accessed

Assignments

- Study your assignments!
- Example question:
  - Given a panic output, can you determine the cause of the error?

Questions from Students

- Can we go through an example of how we can communicate with devices by writing to their command, status, and data registers?

Device Register Example: Sys/161 disk controller

Offset	Size	Type	Description
0	4	status	number of sectors
4	4	status and command	status
8	4	command	sector number
12	4	status	rotational speed (RPM)
32768	512	data	transfer buffer

- Example 1: We want to write one sector of data to the disk controller
- The device driver for the disk controller will do the following:
  - Write the target sector into the sector number command register (using a store operation). The address of the data register is the base address of the device's slot plus the register's offset (8).
  - Write data to the 512-byte data register using multiple store operations (offset 32768).
  - Write "write" command to the status and command register (offset 4)
  - Keep reading the status register until it returns "completed" or error
  - Clear the status register

- Example 2: We want to read one sector of data from the disk controller using DMA
  - Write the target sector into the sector number command register (using a store operation).
  - Write the destination address into the address register (not in the previous table)
  - Write “read” command into the status and command register
  - Block until the device generates a completion interrupt
  - Read status register to check for errors
  - Clear status register

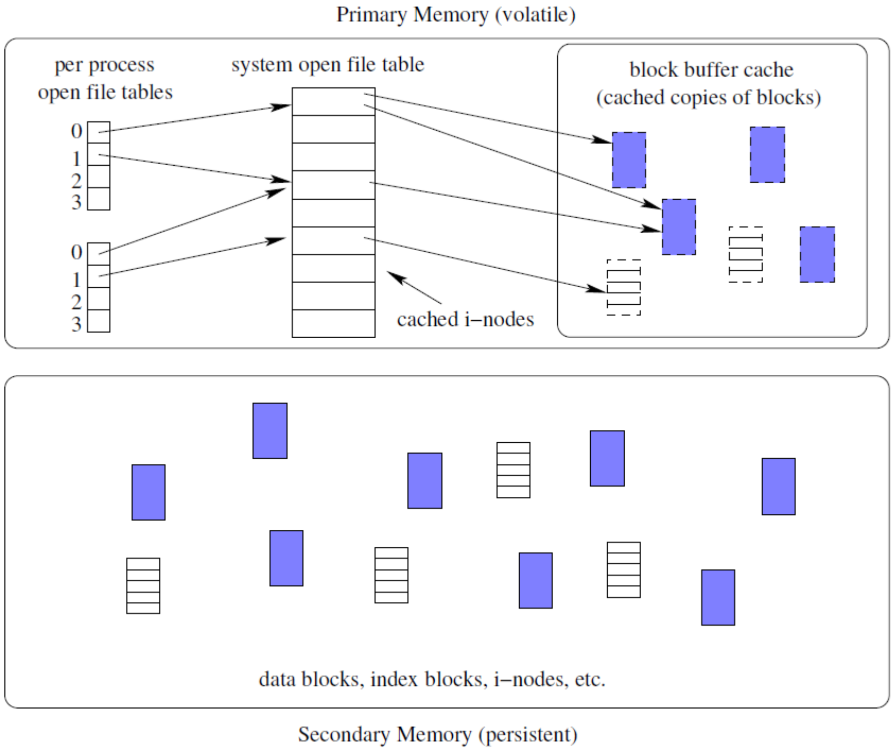
- Does it make sense to use polling with DMA?

- Can we have more examples of navigating inodes?
- Question: Assume that an inode has 12 direct pointers, 1 single indirect pointer and 1 double indirect pointer, and block size is 4 KB and a pointer is 32 bits
- Which pointer is used to fetch a block from offset 2^23 + 10 (8 MB + 10)?
  - The extra + 10 offset is just to eliminate any possible confusion

- We know that this will require a double indirect pointer because of its size
- (12 + 1024 + x) * 2^12 = 2^23 (ignoring the offset for now)
- 12 + 1024 + x = 2^11
- x = 1012
- For 2^23 + 10, we would need to add 1:
  - x = 1013. This assumes that the first pointer in an indirect block is pointer 1. You can also assume that the first pointer is pointer 0, in which case x = 1012.

- How many disk accesses are required?
  - inode + first-level indirect block + second-level indirect block + data block

- Per process open file table entries must include an offset into the system open file table, and the file position for the open file
  - It can include different information if it is for a socket or a pipe
- Each system open file table entry has a cached copy of the inode
- Recently accessed data blocks and indirect blocks can be cached



34. Suppose that a particular program goes through two execution phases. During the first phase, which lasts for t_1 time units, the program makes frequent and heavy use of p_1 pages in its address space. During the second phase, which begins immediately after the first, the program makes frequent and heavy use of p_2 of the pages from its address space. These pages are distinct from those used during the first phase.

    Let $|WS(t, \Delta)|$ represent the size of the program's working set at time t, for a window size of Δ. In other words $|WS(t, \Delta)|$ represents the number of distinct pages referenced during the interval from $t - \Delta$ to t. Sketch a plot of $|WS(t, \Delta)|$ versus t. Assume that $0 < \Delta < t_1$, and that the number of pages referenced during a window of length Δ is much larger than p_1 or p_2. Be sure to mark the axes to indicate the location(s) of any "interesting" points in your sketch.