Announcements

- **Project #1 grades were returned on Monday**
  - Requests for re-grades due by Tuesday
- **Midterm #1**
  - Re-grade requests due by Monday
- **Project #2**
  - Due 10 AM Monday
Page State (hardware view)

- **Page frame number** (location in memory or on disk)
- **Valid Bit**
  - indicates if a page is present in memory or stored on disk
- **A modify or dirty bit**
  - set by hardware on write to a page
  - indicates whether the contents of a page have been modified since the page was last loaded into main memory
  - if a page has not been modified, the page does not have to be written to disk before the page frame can be reused
- **Reference bit**
  - set by the hardware on read/write
  - cleared by OS
  - can be used to approximate LRU page replacement
- **Protection attributes**
  - read, write, execute
What happens when we fault and there are no more physical pages?

- **Need to remove a page from main memory**
  - if it is “dirty” we must store it to disk first.
    - dirty pages have been modified since they were last stored on disk.

- **How to we pick a page?**
  - Need to choose an appropriate algorithm
    - should it be global?
    - should it be local (one owned by the faulting process)
Page Replacement Algorithms

- **FIFO**
  - Replace the page that was brought in longest ago
  - However
    - old pages may be great pages (frequently used)
    - number of page faults may increase when one increases number of page frames (discouraging!)
      - called belady’s anomaly
        - 1,2,3,4,1,2,5,1,2,3,4,5 (consider 3 vs. 4 frames)

- **Optimal**
  - Replace the page that will be used furthest in the future
  - Good algorithm(!) but requires knowledge of the future
  - With good compiler assistance, knowledge of the future is sometimes possible
Page Replacement Algorithms

- **LRU**
  - Replace the page that was actually used longest ago
  - Implementation of LRU can be a bit expensive
    - e.g. maintain a stack of nodes representing pages and put page on top of stack when the page is accessed
    - maintain a time stamp associated with each page

- **Approximate LRU algorithms**
  - maintain reference bit(s) which are set whenever a page is used
  - at the end of a given time period, reference bits are cleared
FIFO Example (3 frames)

- Reference string: 1,2,3,4,1,2,5,1,2,3,4,5
  - access 1 - (1) fault
  - access 2 - (1,2) fault
  - access 3- (1,2,3) fault
  - access 4 - (2,3,4) fault, replacement
  - access 1 - (3,4,1) fault, replacement
  - access 2 - (4,1,2) fault, replacement
  - access 5 - (1,2,5) fault, replacement
  - access 1- (1,2,5)
  - access 2 - (1,2,5)
  - access 3 - (2,5,3) fault, replacement
  - access 4 - (5,3,4) fault, replacement
  - access 5 - (5,3,4)

- 9 page faults
LRU Example (3 frames)

- Reference string: 1,2,3,4,1,2,5,1,2,3,4,5
  - access 1 - (1) fault
  - access 2 - (1,2) fault
  - access 3- (1,2,3) fault
  - access 4 - (2,3,4) fault, replacement
  - access 1 - (3,4,1) fault, replacement
  - access 2 - (4,1,2) fault, replacement
  - access 5 - (1,2,5) fault, replacement
  - access 1- (2,5,1)
  - access 2 - (5,1,2)
  - access 3 - (1,2,3) fault, replacement
  - access 4 - (2,3,4) fault, replacement
  - access 5 - (3,4,5) fault, replacement

- 10 page faults
LRU Example (4 frames)

- Reference string: 1,2,3,4,1,2,5,1,2,3,4,5
  - access 1 - (1) fault
  - access 2 - (1,2) fault
  - access 3- (1,2,3) fault
  - access 4 - (1,2,3,4) fault, replacement
  - access 1 - (2,3,4,1)
  - access 2 - (3,4,1,2)
  - access 5 - (4,1,2,5) fault, replacement
  - access 1- (4,2,5,1)
  - access 2 - (4,5,1,2)
  - access 3 - (5,1,2,3) fault, replacement
  - access 4 - (1,2,3,4) fault, replacement
  - access 5 - (2,3,4,5) fault, replacement

- 8 faults
FIFO Example (4 frames)

- Reference string: 1,2,3,4,1,2,5,1,2,3,4,5
  - access 1 - (1) fault
  - access 2 - (1,2) fault
  - access 3 - (1,2,3) fault
  - access 4 - (1,2,3,4) fault, replacement
  - access 1 - (1,2,3,4)
  - access 2 - (1,2,3,4)
  - access 5 - (2,3,4,5) fault, replacement
  - access 1 - (3,4,5,1) fault, replacement
  - access 2 - (4,5,1,2) fault, replacement
  - access 3 - (5,1,2,3) fault, replacement
  - access 4 - (1,2,3,4) fault, replacement
  - access 5 - (2,3,4,5) fault, replacement

- 10 Page faults
Thrashing

- Virtual memory is not “free”
  - can allocate so much virtual memory that the system spends all its time getting pages
  - the situation is called thrashing
  - need to select one or more processes to swap out

- Swapping
  - write all of the memory of a process out to disk
  - don’t run the process for a period of time
  - part of medium term scheduling

- How do we know when we are thrashing?
  - check CPU utilization?
  - check paging rate?
  - Answer: need to look at both
    - low CPU utilization plus high paging rate --> thrashing