

Midterm Announcements

- **Midterms will be returned at the end of class**
- Solutions for the midterm will be posted on the course website very soon.
- Regrade policy:
 - o **If we made a math mistake, let us know ASAP and we will correct it.**
 - o However, if you believe you deserve more points, turn your midterm back in for a regrade ASAP.
 - o Keep in mind that if we do complete a regrade on your exam, we will regrade your ENTIRE exam, and likely make a profit on the deal.
 - So be very sure that we made a mistake before trying for more.
- Stats on the midterm:
 - o 74 mean
 - o 77 median
 - o 38 low
 - o 96 high
 - o 11 std dev

Project Announcements

- Code due Thursday
- Design docs due Friday
- You know the drill

Today's Menu:

- Cache miss types
- Demand paging
- Page replacement algorithms
- Transparent Page Faulting
- Quiz

Cache Miss Types

- Compulsory
 - o First reference to data will always miss, even with an infinite sized cache
- Capacity
 - o Non-compulsory misses due to limited cache size, assuming fully associative and optimal replacement policy
- Conflict
 - o Non-compulsory, non-capacity misses, due to limited associativity, assuming optimal replacement policy.
- Policy

- Misses due to non-optimal replacement policy

Demand Paging

- Recall two-level page table structure:
 - Each page-table entry contains a 'present' bit
 - On x86 is low-order bit; if 0, present, if 1, not present

QUESTION: Why do we need a 'valid' bit?

ANSWER: So that we can indicate that that portion of the virtual address space is not present in main memory. (i.e. the page is on disk somewhere)

QUESTION: If a page table entry's valid bit is off, what is stored in that page table entry?

ANSWER: If a page is not present, page table entry holds some OS-specific information to maintain location of page on disk

- Typically, this will be a disk block number
- Now, on every page translation:
 - Check valid bit.
 - !valid = PAGE FAULT.
 - Allocate new page
 - read from disk
 - update page table
 - restart faulting instruction
- Note that page tables themselves can be paged!
 - Top-level table always present
 - Second-level tables may be swapped out...
- Key part of the process above:
 - How do we 'allocate a new page'?
 - If one free, allocate it (Nachos Phase II)
 - If not, we need to select a victim and kick it out! (Nachos Phase III)

Page Replacement Algorithms

- How do we select which page to toss?
 - Random
 - FIFO - throw out oldest page
 - MIN - Replace page that won't be used for the longest time in the future
 - (Ideal, but how can we tell?)
 - LRU - Throw out page that hasn't been used for the longest time
 - Often approximated by the CLOCK algorithm.
- Examples of each policy:
 - Assume that we 3 pages of physical mem, 5 pages of virtual mem from different processes: ABCDE
 - The access pattern for these pages is ABCDDEECBADECBA

- *FIFO policy:*

Phys addr	A	B	C	D	D	E	E	C	B	A	D	E	C	B	A
1	A			D						A			C		
2		B				E					D			B	
3			C						B			E			A

- How many page faults?
 - o 12!
- What is hit rate?
 - o Well, only 3 out of 15 of our accesses did not necessitate a page fault, so it's 3/15.
- How many compulsory misses? 5, since we had to bring in each page at least one time during the string of accesses.
- How many capacity and conflict misses?
- Capacity and conflict misses assume that we had an optimal replacement strategy to begin with, and that we COULD NOT AVOID those misses regardless.
 - o FIFO is hardly optimal
 - o Probably best to characterize all of these misses as "policy" misses.
- How many policy misses?
 - o In this case, we have 7 -- all of the misses except for the first 5.

- *MIN policy:*

Phys addr	A	B	C	D	D	E	E	C	B	A	D	E	C	B	A
1	A			D									C	<---	slot 1 or 2 poss.
2		B				E							B	<-	slot 1 or 2 poss.
3			C						B	A					

- How many page faults?
 - o 9
- How many compulsory misses?
 - o 5, of course
- How many capacity misses?
 - o Well, since this is an "optimal" policy we can talk about that.
 - If we had an infinite physical memory then we would have no capacity misses.
 - Since we have only 3 pages, we have to miss for B, A, C, and B -- 4 misses.
- How many conflict misses?
 - o None, really, since the reason we miss the cache has to do with capacity rather than limited associativity.
- How many policy misses?
 - o None - see above.

- *LRU policy:*

Phys addr	A	B	C	D	D	E	E	C	B	A	D	E	C	B	A
-----------	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

1	A	D	A	C
2	B	E	D	B
3	C	B	E	A

- Note that this is the same as FIFO!
 - o Not surprising, since LRU looks at the page that was referenced the longest time in the past.
- This is not going to NECESSARILY be the same as FIFO
 - o depends on the access pattern.

Transparent Page Faulting

QUESTION: Once we have handled the page fault, how do we transparently restart faulting instruction?

ANSWER: Hardware must help us out.

- Must save:
 - o Faulting instruction
 - o Processor state!
- There are lots of complex hardware issues here
 - o What if the next instruction faults while a load is still going on?
 - Again, must save enough processor state in order to be able to restart.

Quiz!