16.482 / 16.561: Computer Architecture and Design

Summer 2015

Lecture 9: Key Questions
June 18, 2015

1. Example: Given a 4-way set associative cache, and five blocks (A, B, C, D, E) that all map to the same set, determine which block is evicted on the access to block E in each of the access sequences below if we use LRU replacement:

a. A, B, C, D, E

b. A, B, C, D, B, C, A, D, A, C, D, B, A, E

c. A, B, C, D, C, B, A, C, A, C, B, E

2. Example: Given the final set-associative cache and memory state shown (after walking through the first five instructions in the set-associative cache example):

MEMORY:

Address		Address	
0	78	8	18
1	29	9	21
2	120	10	33
3	123	11	28
4	18	12	19
5	150	13	200
6	162	14	210
7	173	15	225

CACHE:

V	D	MRU	Tag	Data	
1	0	1	10	18	21
1	1	0	11	19	29
0	0	0	00	0	0
0	0	0	00	0	0

Determine the new cache state, as well as any modified registers and/or memory blocks, after each access listed below.

Access	Modified register	Cache state						Modified mem. block
		V	D	MRU	Tag	Data		
`lb $t1,3($zero)`								
`lb $t0,11($zero)`								
`sb $t0, 2($zero)`								

3. What are the benefits of virtual memory?

4. Four questions for virtual memory:

a. Where can a page be placed in main memory?

b. How is a page found if it is in main memory?

c. Which page should be replaced on a page fault?

d. What happens on a write?

5. Describe the purpose and operation of a translation lookaside buffer (TLB).

6. **Example:** Assume the current process uses the page table below:

Virtual page #	Valid bit	Reference bit	Dirty bit	Frame #
0	1	1	0	4
1	1	1	1	7
2	0	0	0	--
3	1	0	0	2
4	0	0	0	--
5	1	0	1	0

a. Which virtual pages are present in physical memory?

b. Assuming 1 KB pages and 16-bit addresses, what physical addresses would the virtual addresses below map to?

i. 0x041C

ii. 0x08AD

iii. 0x157B

7. Explain each of the following advanced cache optimizations:
a. Way prediction

b. Trace caches

c. Non-blocking caches

d. Multi-banked caches

e. Critical word first and early restart

f. Merging write buffers

g. Software optimizations: array merging, loop interchange, loop fusion, blocking

h. Prefetching (both hardware and software)