CSE 351 Section 8 – More Caches, Processes & Concurrency

Hi there! Welcome back to section, we're happy that you're here ☺

Write Policies

Write Hit

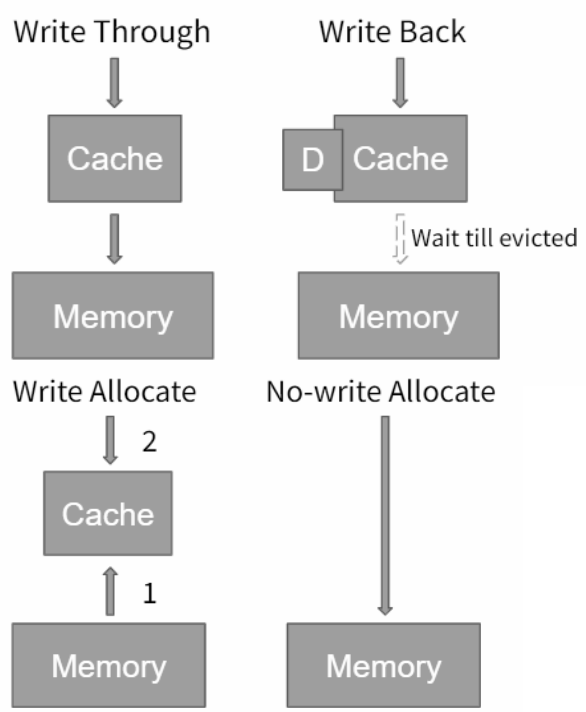
Write Through
- Write to "next level" directly

Write Back
- Defer writing until cache line we wrote to is evicted
- Requires a "dirty bit" that keeps track of modifications
- Only write on eviction if "dirty bit" is set

Write Miss

Write Allocate (fetch on write)
- Load data into cache first (akin to a read)
- Then write to cache
- Good for locality if adjacent writes or reads follow

No-write Allocate (write around)
- Write to "next level" directly



Practice Cache Exam Problem

We have a 64 KiB address space. The cache is a 1 KiB, direct-mapped cache using 256-byte blocks with write-back and write-allocate policies.

a) Calculate the TIO address breakdown:

Tag	Index	Offset

b) During some part of a running program, the cache's management bits are as shown below. Four options for the next two memory accesses are given (R = read, W = write). Circle the option that results in data from the cache being *written to memory*.

Set	Valid	Dirty	Tag
00	0	0	1000 01
01	1	1	0101 01
10	1	0	1110 00
11	0	0	0000 11

(1) `R 0x4C00, W 0x5C00`

(2) `W 0x5500, W 0x7A00`

(3) `W 0x2300, R 0x0F00`

(4) `R 0x3000, R 0x3000`

c) The code snippet below loops through a character array. Give the value of LEAP that results in a Hit Rate of 15/16.

```
#define ARRAY_SIZE 8192
char string[ARRAY_SIZE];              // &string = 0x8000
for(i = 0; i < ARRAY_SIZE; i += LEAP) {
     string[i] |= 0x20;               // to lower
}
```

d) For the loop shown in part (c), let LEAP = 64. Circle ONE of the following changes that increases the hit rate

Increase Block Size Increase Cache Size Add an L2 Cache Increase LEAP

e) What are the three kinds of cache misses? When do they occur? Circle the kind of miss that happens in part (c).

Benedict Cumbercache

Given the following sequence of access results (addresses are given in decimal) on a cold/empty cache of size 16 bytes, what can we *deduce* about its properties? Assume an LRU replacement policy.

```
(0, Miss),(8, Miss),(0, Hit),(16, Miss),(8, Miss)
```

1) What can we say about the block size?

2) Assuming that the block size is 8 bytes, can this cache be… (Hint: draw the cache and simulate it)
   a. Direct-mapped?

   b. 2-way set associative?

   c. 4-way set associative?

Fork and Concurrency

Consider this code using Linux's `fork`:

```
int x = 7;
if( fork() ) {
    x++;
    printf(" %d ", x);
    fork();
    x++;
    printf(" %d ", x);
} else {
    printf(" %d ", x);
}
```

Tip: try drawing a process graph for this program

Write all four of the different possible outputs (i.e. order of things printed) for this code?