Tutorial: Caches

Caches: 2

2/2 points (ungraded)

Two otherwise-identical Beta systems have 1024-line two-way set-associative caches using LRU and FIFO replacement, respectively. For what programs does the **FIFO** cache show a higher hit ratio than the LRU cache?

FIFO outperforms on:

- ○ **ALL**
- ○ **MOST**
- ◉ **A FEW**
- ○ **NONE**

✔

Two otherwise-identical Beta systems have fully-associative LRU caches. One cache has four lines, each caching a single 32-bit word; the other cache has eight lines, each also holding a single 32-bit word. For what programs does the **smaller** cache show a higher hit ratio?

***Small* cache outperforms on (circle one):**

- ○ **ALL**
- ○ **MOST**
- ○ **A FEW**
- ◉ **NONE**

✔

Submit

Caches: 1

0.5/1.0 point (ungraded)

1. You are evaluating two tiny 4-line caches, each with a total storage of four data words. Model DM is direct-mapped, and model FA is fully associative (LRU). Each uses **word addressing** (hence consecutive addresses differ by 1, not 4 as in the Beta). The benchmark involves just six memory reads, starting with an empty (invalidated) cache. Finish the reference string given below to yield an access pattern that will give a **better hit rate on DM than on FA**. Fill in the final two additional memory addresses. **Use single-digit decimal addresses.**

   0, 1, 2, 3, 3 ✖ , 3 ✖

2. Same setup as above; this time give references that make FA look better. Again, **use single-digit decimal addresses.**

   0, 1, 2, 3, 5 ✔ , 1 ✔

Submit

Caches: 2

9 points possible (ungraded)

You are considering three possible caches for use in a Beta-like computer system that uses 24-bit byte addresses to access 32-bit (4-byte) words in main memory – hence each word address is divisible by four (as in

the Beta).

Each of the caches you are considering has a total of 64 cache lines, each holding one 32-bit data word. The three cache models are as follows:

Cache	Description
DM	Direct-mapped, 64 lines
4Way	4-way set associative: 16 sets of 4 lines each.
FA	Fully-associative: one set of 64 lines.

To compare caches, you have collected some additional parameters about each model and have run a tiny benchmark program to measure the performance of each. Your benchmark program is a tight loop that repeatedly accesses the **4** locations given by the **hex byte addresses**:

0×100, 0×200, 0×104, 0×108, 0×100, 0×200, 0×104, 0×108, 0×100, ...

You have been asked to complete the following table. Fill in blank entries; use “NA” (Not Applicable) for inapplicable parameters.

Assume that the incoming address from the CPU is A[23:0], that each cache stores the minimum number of address bits in the tag field for each line. The “# Lines for Mem[0]” column lists the number of lines in each cache that might hold the contents of main memory location zero, and the Hit Rate column reflects each cache’s steady-state performance on the little benchmark described above.

Cache Model	Bits of A[23:0] used to address cache	# Tag bits Per line	Replacement Strategy	# Lines for Mem[0]	Hit Rate
DM	A[7:2]	16 Answer: 16	NA Answer: NA	1	2/4 Answer: 50%
4Way	**A[** 5 Answer: 5 **:** 2 Answer: 2 **]**	18 Answer: 18	LRU	4 Answer: 4	1 Answer: 100%
FA	NA	22	LRU	64 Answer: 64	100%

Explanation

The DM cache has 64 lines that are indexed via the address bits. Since each cache line is 32 bits (4 bytes) and addresses are byte addresses, that means that A[1:0] are used as byte offset. So in order to distinguish between the 64 lines of the direct mapped cache, 6 bits of address must be used. To benefit from cache locality, the bits used are the lower address bits, hence A[7:2] are used to select a line in the cache. The remaining bits are used for the tag. For the 4 Way set associative cache there are 16 distinct cache lines so only 4 bits are required to select one of the cache lines, hence A[5:2].

Since we started with 24 bits of address, that means that for the DM cache, there are 16 bits of tag, and for the 4Way set associative cache, there are 18 bits of tag.

In a direct mapped cache there is no replacement strategy, therefore, that answer is NA.

The number of lines that memory location 0 could be mapped to corresponds to the associativity of the cache. In a direct mapped cache, there is exactly one line it can be mapped to, whereas in a 4Way set associative with 16 sets of size 4, there are 4 possible locations that address 0 could end up in. Finally, for the fully associative cache with 64 lines, address 0 can go in any one of the 64 lines.

To analyze the hit rate of the benchmark program on each cache, we see that there are 4 instructions that are repeated over and over. This means that in the fully associative cache, there is enough room for the entire loop so there will be no conflicts and a 100% hit rate. In the 4Way set associative cache, even for addresses that map to the same set, there are 4 possible lines to choose from and therefore, the entire loop will fit in this cache as well and result in a 100% hit rate. In the direct mapped cache, however, addresses 0×100, 0×104, and 0×108 map to different cache lines but 0×200 maps to the same cache line as 0×100. This means that 0×100 and 0×200 will always miss and 0×104 and 0×108 will always hit in steady state. So the hit rate is 50%.

Submit

Answers are displayed within the problem

Discussion

Hide Discussion

Topic: 14. Caches and the Memory Hierarchy / Tutorial: Caches

Add a Post

Show all posts by recent activity

Problem 1 issue 4
Hi All, Regarding the first problem, I have some difficulties to understand what is requested, even with the answer! What are exactly...

Entering the hit rates. 2

Previous Next