LE14.3.1 Set-associative cache

0.0/1.0 point (ungraded)

Consider a **2-way set-associative** cache that has **16 cache lines** in each of the two direct-mapped subcaches and a **block size of 2 words** (i.e., a miss brings in an even-odd word pair from main memory).

1. The Beta produces 32-bit byte addresses, A[31:0]. To ensure the best cache performance for the 2-way set associative cache described above, which address bits should be used for selecting the cache line? For matching with the tag field?

   Address bits used for selecting cache line: A[

   □ Answer: 6

   :

   □ Answer: 3

   ]

   Address bits used for matching with tag field: A[

   □ Answer: 31

   :

   □ Answer: 7

   ]

   Explanation

   Looking at the address bits from right to left, the bottom 2 bits of the address are 00 for word alignment. After that we want to have the block offset bits so that you can benefit from locality by bringing more than one word in per block. Next comes the cache line selector, and finally, the tag.

   Having a block size of 2 words means that 1 bit is used for the block offset, so bit 2 is used for the block offset. To address 16 lines of cache, we needs 4 bits to select the line, so the next 4 bits A[6:3] are used to select the cache line. The remaining bits are all used for the tag, A[31:7].

2. Using the 2-way set-associative cache with a block size of 2 described above, estimate the approximate long-term hit ratio for the following program. Assume that the cache is empty before execution (all the valid bits are 0) and that an LRU replacement strategy is used. Remember the cache is used for both instruction and data (LD) accesses.

```
      . = 0x2000
      CMOVE(0x1000,R0)
loop: LD(R0,0,R1)
      SUBC(R0,4,R0)
      BNE(R0,loop)
      HALT()
```

Approximate long-term hit ratio:

Answer: 7/8

Explanation

In steady state, the loop has 4 memory accesses: 3 instruction fetches and 1 data fetch. The instruction fetches will all hit. If the block size was 1 word, then the data fetches would all be misses. However, because the block is of size 2. That means that every other data fetch will be a hit. So you have 1 miss per 8 memory accesses, so your hit ratio is 7/8.

Submit

ℹ Answers are displayed within the problem

LE14.3.2 Fully-associative cache

0.0/1.0 point (ungraded)

A Beta processor (generating 32-bit byte addresses) is connected to a fully associative cache having four cache lines each containing one 32-bit data word as well as dirty and valid bits. The cache uses Least Recently Used (LRU) replacement. Access times are 5 ns on a hit and a total of 50ns on a miss (including the 5ns cache access time).

1. What hit ratio is necessary to yield an average access time of 14ns?

   Hit ratio for 14ns t_{avg}:

   Answer: 0.8

   Explanation

   average access time = (hit rate * hit time) + (miss rate * miss time)

   14 = hit_rate * 5 + (1 - hit_rate) * 50 = -45 hit_rate + 50

   45 hit_rate = 36

   hit_rate = 36/45 = 4/5 = 0.8

2. The Beta produces 32-bit byte addresses, **A[31:0]**. Which of these bits are compared with tags stored in the cache?

   Bits compared with stored tags: A[

   Answer: 31

   :

   Answer: 2

   ]

   Explanation

   The bottom two bits of the address are used for byte addressing and are always treated as 00 when looking up 32 bit words, so these bits are not part of the tag. All other bits are used in the tag because the type of cache in this beta is a fully associative cache so no bits are needed to pick a set. Furthermore, there is exactly 1 word per cache line so no bits are needed to identify the block within a line.

3. How many such comparisons are performed simultaneously for each memory read?

   number of simultaneous comparisons with stored tags:

   Answer: 4

   Explanation

   In a fully associative cache, the number of comparisons made is equal to the number of cache lines because the data could be located in any of them. So in a 4 line fully associative cache, there are 4 comparisons of the tag to the given address.

Submit

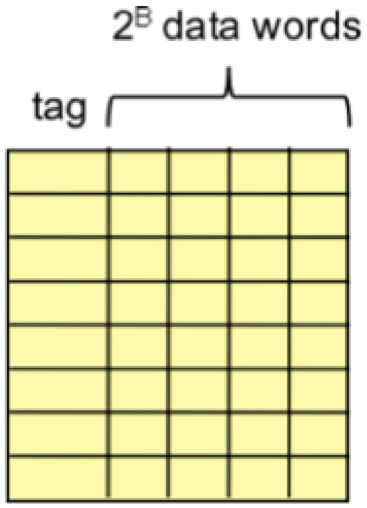
ℹ Answers are displayed within the problem

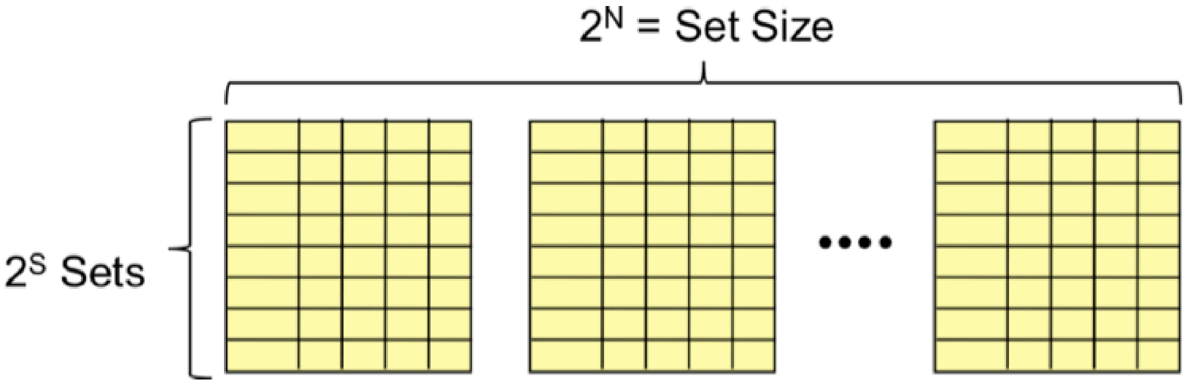
LE14.3.3 Set-associative cache parameters

1/1 point (ungraded)

Consider the universe of caches constructed from one or more static RAM lookup tables organized as shown on the right, i.e. as table allowing access to a single entry (line) at a time, each line having a tag portion as well as 2^B data entries.

Combining multiple of these devices along with necessary logic, comparators, and replacement strategy, we can build a parameterized family of set-associative caches. Each cache in this family implements 2^S sets, where each set comprises 2^N lines and each line has a tag as well as 2^B consecutive words of data from main memory as depicted in the “big picture” view below:





For each of the following questions, assume a machine that uses **A** bit addresses (to keep it simple, we avoid the byte addressing of the Beta; thus consecutive addresses differ by one). You may answer each question by a number, or a formula involving the parameters A, B, N, and S.

1. What is the total number of data words that can be held in the cache?
   Formula for total cache size, in words:

   2^(B+S+N) ✓ Answer: 2^(S+N+B)

   2^{B+S+N}

   Explanation
   The total number of words stored in the cache is the:
   number of sets × *set size* × *data words per cache line*
   $= 2^S * 2^N * 2^B = 2^{S+N+B}$.

2. What constraint on the above parameters characterizes a **direct-mapped** cache?

   ○ $A = 0$

   ○ $B = 0$

● $N = 0$

○ $S = 0$

✓

Explanation
In a direct mapped cache, any element can be mapped to exactly one location within a set. This occurs when N = 0, so the size of the set = 1.

3. What constraint on the above parameters characterizes a **fully-associative** cache?

○ $A = 0$

○ $B = 0$

○ $N = 0$

● $S = 0$

✓

Explanation
In a fully associative cache, there is only 1 set, so any element can be mapped to any location in the cache. This occurs when S = 0.

4. What is the minimum number of bits required in the tag portion of each cache line?

Formula for size of each tag:

A-S-B ✓ Answer: A-S-B

$A - S - B$

Explanation
The tag portion of each cache line must uniquely identify the address of the word stored in that cache location. Since the block that a word is mapped to depends on B bits of the address, and the set that a word is mapped to depends on S bits of the address, the remaining bits of tag are $A - S - B$. Note that the selection of line within a set is based on the cache replacement strategy, not on the address, so N is not part of this equation.

Submit

ℹ Answers are displayed within the problem

LE14.3.4 Cache comparisons

1/1 point (ungraded)

Three otherwise identical Beta systems have slightly different cache configurations. Recall that the Beta supplies 32-bit byte addresses when accessing memory. Each cache has a total of 8 lines caching a single 32-bit data word; a single cache is used when responding to both instruction and data fetches. However, the caches differ in their associativity as follows:

- **Cache C1:** 8-line direct mapped.
- **Cache C2:** 2-way set associative (4 sets of 2 lines), LRU replacement.
- **Cache C3:** 8-line fully associative, LRU replacement.

1. How many bits are there in the TAG field of each line in cache C1? Which address bits from the Beta are used by the cache's comparator to determine if there is a cache hit?

   Number of tag bits in C1 cache line:

   27 ✓ Answer: 27

   Beta address bits used in hit logic: A[

   31 ✓ Answer: 31

   :

   5 ✓ Answer: 5

   ]

   Explanation

   In cache C1, the bottom two address bits (A[1:0]) would be used for the byte offset, the next 3 lowest bits (A[4:2]) would be used to select a cache line. The remaining bits would be used for the tag. So bits A[31:5] are used for the tag for a total of 27 tag bits.

2. After the Beta with the C2 cache runs for a while the tag and data fields of the cache are as shown in the table below. Assume that any unspecified bits are 0 and that all cache entries are valid. For each of the given Beta read requests to memory, indicate whether they would hit or miss in the cache and the data returned on a hit (enter CAN'T TELL for the data returned if there is a cache miss).

Line #	Tag	Data	Tag	Data
3	0×40	0×739F0083	0×00	0×73FF0121
2	0×00	0×73FF012C	0×40	0×619F0044
1	0×00	0×73FF029E	0×40	0×515F003C
0	0×41	0×627F0060	0×00	0×73FF02C3

Hit for address 0×00C?

● HIT

○ MISS

✓

Data at location 0×00C or "CAN'T TELL" for "MISS": 0x

73FF0121 ✓ Answer: 73FF0121

Hit for address: 0×400?

○ HIT

● MISS

✓

Data at location 0×400 or "CAN'T TELL" for "MISS": 0x

CAN'T TELL ✓ Answer: CAN'T TELL

Explanation

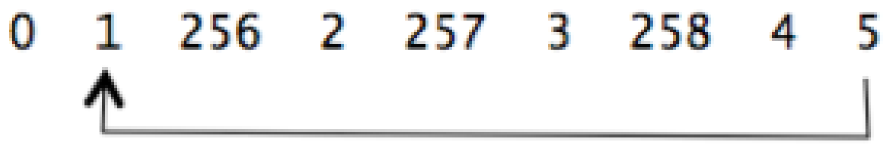
In the 2-way set-associative cache, the bottom 2 bits of the address are used for the byte offset, the next 2 bits are used to select the cache line. The remaining bits are used for the tag.

For address 0×00C, the cache line is cache 0b11 = 3 and the tag = 0×00. Since one of the caches holds tag = 0×00 in line 3, this results in a hit, and the data returned is 0×73FF0121.

For address 0×400, the cache line is 0b00 = 0 and the tag is 0×40. Since this tag is not present in cache line 0, this results in a cache miss and we therefore CAN'T TELL the contents of that address.

3. Consider the short benchmark program given below. The reference string of word addresses generated as the benchmark runs is shown below the benchmark with an arrow indicating how the access pattern repeats. Please indicate the hit ratio delivered by the each of the caches after the benchmark has run for a while, e.g., during the 10th iteration of the loop.

```
      .=0
      CMOVE(100, R1)
LOOP: LD(R31, 1024, R0)
      LD(R31, 1024+4, R0)
      LD(R31, 1024+8, R0)
      SUBC(R1, 1, R1)
      BNE(R1, LOOP)
```



Hit Ratio? for C1:

4/8

✓ Answer: 4/8

Hit Ratio? for C2:

5/8

✓ Answer: 5/8

Hit Ratio? for C3:

1

✓ Answer: 8/8

Explanation

The benchmark has a loop containing 5 instructions and 3 data fetches. In cache C1, the direct mapped cache, the fetch of word 256 maps to line 0, 257 to line 1, and 258 to line 2. The instructions in the loop are at lines 1-5. This means that there is a conflict between 2 of the data fetches and 2 of the instruction fetches. Therefore the hit rate is 4/8.

In cache C2, the two-way set associative cache, the data fetches map to lines 0-2, while the instruction fetches map to lines 1,2,3,0,1. Because the cache is 2-way, there is room in the cache for two fetches that map to each line. However, there are three fetches that map to line 1 so those 3 fetches will conflict with each other resulting in a hit ratio of 5/8.

In cache C3, the fully associative cache, there is enough room for the 5 instructions and 3 data fetches so the resulting hit ratio is 8/8 or 100%.

Submit

Answers are displayed within the problem

Discussion

Hide Discussion

Topic: 14. Caches and the Memory Hierarchy / LE14.3

Add a Post

Show all posts by recent activity

- ☑ LE14.3.4 CACHE COMPARISONS Part C: Why maps - in cache C1 - the fetch of word 256 to line 0, etc.? 3
  The explanation states: *The benchmark has a loop containing 5 instructions and 3 data fetches. I...
- Question about execise B of LE14.3.1 SET-ASSOCIATIVE CACHE 3
  The cache description is as the following. ------------ Consider a 2-way set-associative cache tha...
- About the hit rate 2
  "If the block size was 1 word, then the data fetches would all be misses. However, because the blo...
- ☑ LE14.3.1 2
  What is "word alignment" at the bottom 2 bits? and why it is 00?
- ☑ Memory access time solution? 4
  In the LE14.3.2 Problem A, the solution states that **average access time = (hit rate * hit time) + (...
- CMOVE(0×1000,R0) 2
  I have no idea how I got through the rest of the course without knowing this, but: ADDC(R31, c , Rc...
- LE14.3.1.b Does cache load memory addresses in ascending order? 3
  As I understand over the long term, the cache will retain the instruction that are in Memory locatio...
- CACHE COMPARISONS 3
  In the last problem, shouldn't the description of cache c2 be "2 sets of 4 lines"?
- 2-way SA 2
  During a miss, which way does the program overwrite? Is this a relevant question?
- Implementing "waits" on CPU 2
  The Beta design seems predicated on an assumption that we will always have the next instruction ...