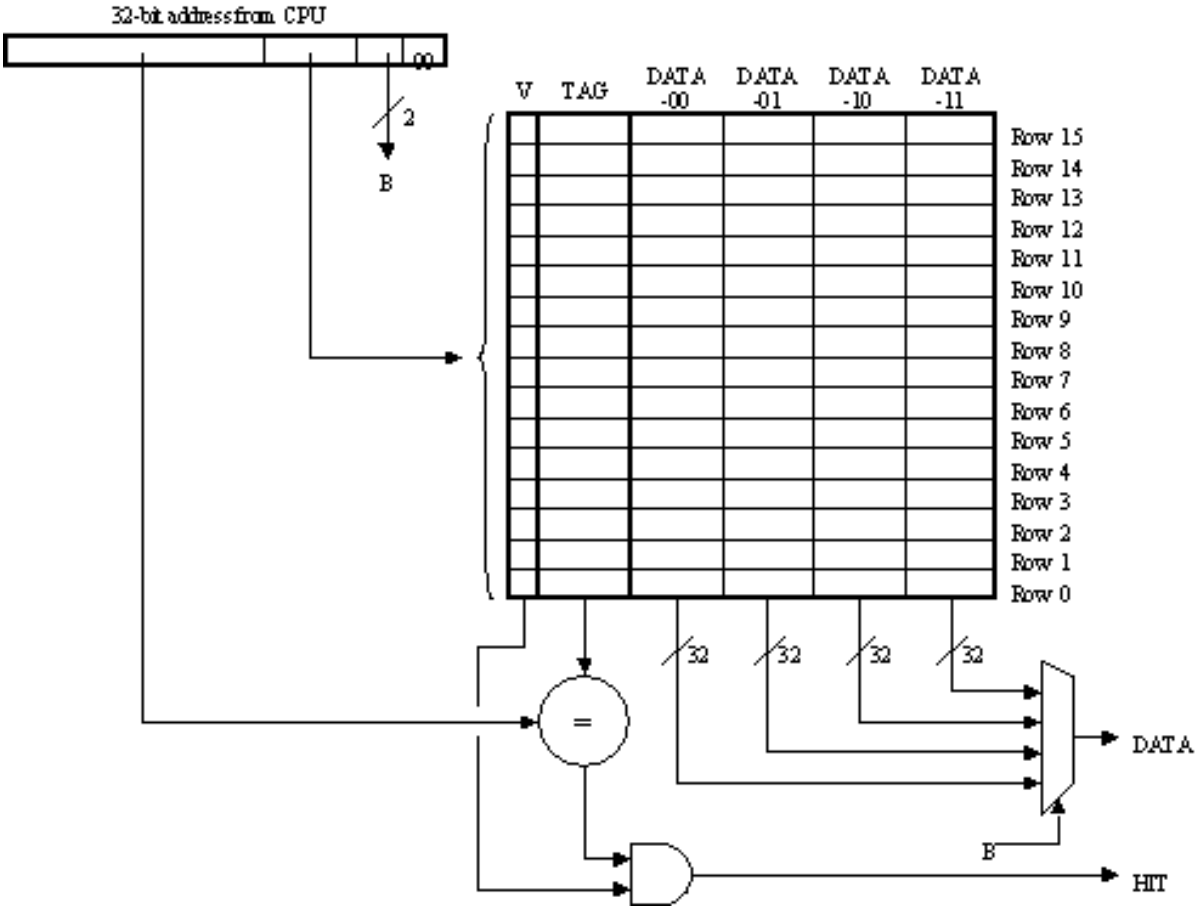
6.004 Computation Structures
Spring 2009

Caches

Problem 1.



The diagram above illustrates a blocked, direct-mapped cache for a computer that uses 32-bit data words and 32-bit byte addresses.

A. ★ What is the maximum number words of data from main memory that can be stored in the cache at any one time?

Maximum number of data words from main memory = (16 lines)(4 words/line) = 64 words

B. ★ How many bits of the address are used to select which line of the cache is accessed?

With 16 cache lines, 4 bits of the address are required to select which line of the cache is accessed.

C. ★ How many bits wide is the tag field?

Bits in the tag field = (32 address bits) - (4 bits to select line) - (4 bits to select word/byte) = 24 bits

D. ★ Briefly explain the purpose of the one-bit V field associated with each cache line.

The tag and data fields of the cache will always have value in them, so the V bit is used to denote whether these value are consistent (valid) with what is in memory. Typically the V bit for each line in the cache is set to "0" when the machine is reset or the cache is flushed.

E. ★ Assume that memory location 0x2045C was present in the cache. Using the row and column labels from the figure, in what cache location(s) could we find the data from that memory location? What would the value(s) of the tag field(s) have to be for the cache row(s) in which the data appears?

The cache uses ADDR[7:4] to determine where data from a particular address will be stored in the cache. Thus, location 0x0002045C will be stored in line 5 of cache. The tag field should contain the upper 24 bits of the address, i.e., 0x000204. Note that the bottom 4 bits of the address (0xC) determine which word and byte of the cache line is being referenced.

F. ★ Can data from locations 0x12368 and 0x322FF8 be present in the cache at the same time? What about data from locations 0x2536038 and 0x1034? Explain.

Location 0x12368 will be stored in line 6 of the cache. Location 0x322F68 will be stored in line F of the cache. Since the lines differ, both locations can be cached at the same time. However, locations 0x2536038 and 0x1034 both would be stored in line 3 of cache, so they both could not be present in the cache at the same time.

G. ★ When an access causes a cache miss, how many words need to be fetched from memory to fill the appropriate cache location(s) and satisfy the request?

There are 4 words in each line of the cache and since we only have one valid bit for the whole line, all 4 words have to have valid values. So to fill a cache line on a cache miss all 4 words would have to be fetched from main memory.

Problem 2. The following four cache designs C1 through C4, are proposed for the Beta. All use LRU replacement where applicable (e.g. within each set of a set associative cache).

Cache	C1	C2	C3	C4
Total Data words	8K	4K	8K	16K
Total Lines	8K	4K	4K	8K
Associativity	Fully	2-way S.A.	Direct Mapped	fully
Block size, words/line	1	1	2	2

A. ★ Which cache would you expect to take the most chip area (hence cost) ?

Cache C4 would most likely take up the most chip area because it is fully associative, thereby requiring a comparator for each cache line, and because it has the most data word capacity.

B. ★ Which cache is likely to perform worst in a benchmark involving repeated cycling through an array of 6K integers ? Explain.

C2 would likely have the worst performance on a benchmark involving repeated cycling through an array of 6K integers since it is the only cache listed with less than 6K data word capacity.

C. ★ It is observed that one of the caches performs very poorly in a particular benchmark which repeatedly copies one 1000-word array to another. Moving one of the arrays seems to cure the problem. Which cache is most likely to exhibit this behavior ? Explain.

We are told that one of the caches performs poorly in a particular benchmark which repeatedly copies one 1000-word array to another and that if one of the arrays is moved, the problem seems to be cured. This behavior is most likely exhibited by cache C3 because it is a direct mapped cache which only has one location to put any particular address. If the lower bits (used to choose the cache line) for the addresses of the array overlap, poor performance could be observed. Moving the array so that the lower bits of the array addresses don't overlap could solve this problem.

D. ★ Recall that we say cache A dominates cache B if for every input pattern, A caches every location cached by B. Identify every pair (A, B) of caches from the above list where A dominates B. Explain your reasoning.

So long as we are not using a random replacement strategy, it is always possible to come up with a benchmark that will make a particular type of cache have a miss on every data access. Thus, we cannot say that one particular type of cache always dominates another type of cache. However, we can compare two caches of the same type. Both C4 and C1 are fully associative caches with the same replacement strategy. We can say that C4 dominates C1 since C4 has a greater data word capacity.

Problem 3. The data-sheet for a particular byte-addressable 32-bit microprocessor reads as follows:

The CPU produces a 32-bit virtual address for both data and instruction fetches. There are two caches: one is used when fetching instructions; the other is used for data accesses. Both caches are virtually addressed. The instruction cache is two-way set-associative with a total of 2^{12} bytes of data storage, with 32-byte blocks. The data cache is two-way set-associative with a total of 2^{13} bytes of data storage, with 32-byte blocks

A. How many bits long is the tag field in each line of the instruction cache?

There are $32 = 2^5$ bytes per block. The cache has 2^{12} total bytes and is 2-way set associative, so each set has 2^{11} bytes and thus $2^{11}/2^5 = 2^6$ cache lines. So the address is partitioned by the cache as follows:

[4:0] = 5 address bits for selecting byte/word within a block
[10:5] = 6 address bits for selecting the cache line
[31:11] = 21 address bit of tag field

B. How many address bits are used to choose which line is accessed in the data cache?

There are $32 = 2^5$ bytes per block. The cache has 2^{13} total bytes and is 2-way set associative, so each set has 2^{12} bytes and thus $2^{12}/2^5 = 2^7$ cache lines. So the address is partitioned by the cache as follows:

[4:0] = 5 address bits for selecting byte/word within a block
[11:5] = 7 address bits for selecting the cache line
[31:12] = 20 address bit of tag field

C. Which of the following instruction addresses would never collide in the instruction cache with an instruction stored at location 0x0ACE6004?

(A) 0x0BAD6004 (D) 0x0ACE6838
(B) 0x0C81C81C (E) 0xFACE6004
(C) 0x00000004 (F) 0x0CEDE008

Collisions happen when instruction addresses map to the same cache line. Referring to the answer for (A), address bits [10:5] are used to determine the cache line, so location 0x0ACE6004 is mapped to cache line 0.

Only (D) 0x0ACE6838 maps to a different cache line and hence could never collide in the instruction cache with location 0x0ACE6004.

D. What is the number of instructions in the largest instruction loop that could be executed with a 100% instruction cache hit rate during all but the first time through the loop?

The instruction cache hold 2^{12} bytes or $2^{10} = 1024$ instructions. So if the loop had 1024 instructions it would just fit into the cache.

Problem 4. The following questions ask you to evaluate alternative cache designs using patterns of memory references taken from running programs. **Each of the caches under consideration has a total capacity of 8 (4-byte) words**, with one word stored in each cache line. The cache designs under consideration are:

DM: a direct-mapped cache.

S2: a 2-way set-associative cache with a least-recently-used replacement policy.

FA: a fully-associative cache with a least-recently-used replacement policy.

The questions below present a sequence of addresses for memory reads. **You should assume the sequences repeat from the start whenever you see "...".** Keep in mind that byte addressing is used; addresses of consecutive words

in memory differ by 4. Each question asks which cache(s) give the best hit rate for the sequence. Answer by considering the steady-state hit rate, i.e., the percentage of memory references that hit in the cache after the sequence has been repeated many times.

A. ★ Which cache(s) have the best hit rate for the sequence 0, 16, 4, 36, ...

DM: locations 4 and 36 collide, so each iteration has 2 hits, 2 misses.

S2: 100% hit rate. 0 and 16 map to the same cache line, as do 4 and 36, but since the cache is 2-way associative they don't collide.

FA: 100% hit rate. The cache is only half filled by this loop.

B. ★ Which cache(s) have the best hit rate for the sequence 0, 4, 8, 12, 16, 20, 24, 28, 32, ...

DM: locations 0 and 32 collide, so each iteration has 7 hits, 2 misses.

S2: locations 0, 16 and 32 all map to the same cache line. The LRU replacement strategy replaces 0 when accessing 32, 16 when accesing 0, 32 when accessing 16, etc., so each iteration has 6 hits, 3 misses.

FA: has 0% hit rate in the steady state since the LRU replacement strategy throws out each location just before it's accessed by the loop!

C. ★ Which cache(s) have the best hit rate for the sequence 0, 4, 8, 12, 16, 20, 24, 28, 32, 28, 24, 20, 16, 12, 8, 4, ...

All caches perform the same -- locations 0 and 32 trade places in the caches, so each iteration has 14 hits and 2 misses.

D. ★ Which cache(s) have the best hit rate for the sequence 0, 4, 8, 12, 32, 36, 40, 44, 16, ..

DM: 32 collides with 0, 36 with 4, 40 with 8, 44 with 12, so each itreation has only 1 hit and 8 misses.

S2: locations 0, 16 and 32 trade places in the cache, so each iteration has 6 hits and 3 misses.

FA: 0 hits since LRU throws out each location just before it's accessed by the loop.

E. ★ Assume that a cache access takes 1 cycle and a memory access takes 4 cycles. If a memory access is initiated only after the cache has missed, what is the maximum miss rate we can tolerate before use of the cache actually slows down accesses?

If accesses always go to memory, it takes 4 cycles per access. Using the cache the average number of cycles per access is

1 + (miss rate)*4

So if the miss rate is larger than 75% the average number of cycles per access is more than 4.

Problem 5. Ben Bitdiddle has been exploring various cache designs for use with the Beta processor. He is considering only caches with one word (4 bytes) per line. He is interested in the following cache designs:

C1: 2-way set associative, LRU replacement, 256 total data words (128 sets of 2 words each).

C2: 2-way set associative, random replacement, 256 total data words (128 sets of 2 words each).

C3: 2-way set associative, LRU replacement, 512 total data words (256 sets of 2 words each).

C4: 4-way set associative, LRU replacement, 512 total data words (128 sets of 4 words each).

C5: Fully associative, LRU replacement, 512 total data words.

In order to help her analysis, Ben is trying to identify cases where one cache dominates another in terms of cache hits. Ben considers that cache A dominates cache B if, given identical strings of memory references, every memory reference that gives a cache hit using B is also a hit using A. Thus if A dominates B, A will give at least as high a hit rate as B for every program.

In each of the following pairs of caches, deduce whether the first dominates the second:

A. C1 dominates C2

False. C1 has a 0% hit rate for 0, 256, 512, 0, 256, 512, ..., but C2 might do slightly better because it chooses the replacement set at random.

B. C2 dominates C1

No. C1 has a 100% hit rate for 0, 256, 0, 256, ..., but C2 might have an occasional miss.

C. C3 dominates C1

Yes. C3 differs only in having a higher capacity than C1.

D. C3 dominates C2

No. As we saw in (A) there are programs where LRU gets 0% hit rate and random may do slightly better, independently of the sizes of the caches.

E. C4 dominates C3

No. C4 has 0% hit rate on 0, 128, 256, 384, 512, 0, ... since all accesses map to the same cache line and LRU throws out the location just about to be accessed. In C3, 128 and 384 map to a different cache line than 0, 256 and 512, so manages a 40% hit rate in the steady state.

F. C4 dominates C2

No, for the same reason as (A) and (D).

G. C5 dominates C1

No. Consider the following access pattern: 0, accesses to 512 uncached locations whose addresses don't map to cache line 0 for cache C1, 0, ...

C5 will replace location 0 on the 513th access and hence miss when 0 is accessed in the following cycle. C1 will have location 0 still in the cache when it's accessed again by the loop.

H. Averaged over a wide range of typical application programs, which of the above caches would you expect to yield the highest hit rate?

In general larger caches are better and fully associative caches are better than set associative caches, so C5 should have the highest hit rate.

Problem 6. Adverbs Unlimited is considering a computer system based loosely on the Beta. Five designs have been proposed, each of them similar to the Beta except for a cache between the 32-bit processor data bus and the main-memory subsystem. Like the Beta, each machine deals with 32-bit main-memory addresses, for a total address space of 2^{32} bytes. The machines' caches differ only in the parameters of associativity, size, and writeback. The block size for each cache 1 word (4 bytes).

Model	Associativity	Total data size (bytes)	Write-
DEFINATELY	four-way	2^{16}	back
CERTAINLY	direct-mapped	2^{16}	back
HOPEFULLY	4-way	2^{10}	through
PERHAPS	2-way	2^{10}	back
DOUBTFULLY	direct-mapped	2^{10}	back

A. How many bits are required for the *tag* portion of each cache line for each of the architectures? How bits of comparitor are needed? How many bits of SRAM altogether (including tag fields, valid and dirty bits).

DEFINIATELY: 2^{16}/4-way = 2^{14} bytes/subcache
=> 2^{12} cache lines/subcache => 32 - 14 = 18 tag bits

=> 18 * 4 = 76 bits of comparator
=> total SRAM bits = 4*(8*2^{14} data bits + 2^{12}*(18 tag + 1 valid + 1 dirty))

CERTAINLY: 2^{16}/1-way = 2^{16} bytes/subcache
=> 2^{14} cache lines => 32 - 16 = 16 tag bits
=> 16 bits of comparator
=> total SRAM bits = 8*2^{16} data bits + 2^{14}*(16 tag + 1 valid + 1 dirty)

HOPEFULLY: 2^{10}/4-way = 2^{8} bytes/subcache
=> 2^{6} cache lines/subcach => 32 - 8 = 24 tag bits
=> 24 * 4 = 96 bits of comparator
=> total SRAM bits = 4*(8*2^{8} data bits + 2^{6}*(24 tag + 1 valid))

PERHAPS: 2^{10}/2-way = 2^{9} bytes/subcache
=> 2^{7} cache lines/subcach => 32 - 9 = 23 tag bits
=> 23 * 2 = 46 bits of comparator
=> total SRAM bits = 2*(8*2^{9} data bits + 2^{7}*(23 tag + 1 valid + 1 dirty))

DOUBTFULLY: 2^{10}/1-way = 2^{10} bytes/subcache
=> 2^{8} cache lines => 32 - 10 = 22 tag bits
=> 22 bits of comparator
=> total SRAM bits = 8*2^{10} data bits + 2^{8}*(22 tag + 1 valid + 1 dirty)

B. Address lines from the CPU are designated A31, ..., A1, A0, where A0 is the low-order address bit. Which of these CPU address lines are used as address inputs to the SRAMs of the cache in the PERHAPS model?

PERHAPS is a 2-way set-associative cache with a total of 2^{10} bytes, so each direct-mapped subcache contains 2^{9} bytes. With a block size of 1 word (4 bytes), address bits [8:2] would be used as the index into the 32-bit-wide SRAM.

C. Suppose that address lines A2 and A9 were inadvertently interchanged in the cable between the DOUBTFULLY CPU and its cache. Which, if any, of the following statements best describes the effect(s) of this change, assuming that other hardware and software remain unmodified?
   A. The machine would no longer work.
   B. The machine would continue to work as before.
   C. The machine would continue to work, but at a reduced performance level.
   D. The machine would continue to work, at an improved performance level.

(B). Address bits A2 through A9 are used as the cache index, interchanging them has no effect other than to change where in SRAM each cache line is stored, i.e., all the same locations are cached, they just happen to be stored in different cache SRAM locations than one might have expected.