In C, mulit-dimensional arrays are stored in row-major order: elements in a particular row are adjacent in memory. For example, given the declaration for an (n+1)-row by (m+1)-column array, "int A[n+1][m+1]", the array is layed out in memory as follows (lower memory addresses to the left),

A[0][0], A[0][1], ... A[0][m], A[1][0], A[1][1], ..., A[1][m], ..., A[n][0], A[n][1], ..., A[n][m]

That is, "A[k]" is a reference to a linear array of m+1 elements. Here is some C code:

```
int A[8000][8000], B[8000][8000], x, y;

for (x = 0; x < 8000; x++) {
    for (y = 0; y < 8; y++) {
        A[x][y] = B[y][0] + A[y][x];
    }
}
```

**Q.** Suppose we have 16B cache blocks and INT word size is 32-bit. How many words per cache line?

**Q.** How many cache blocks are needed to hold all data items referenced on the LHS in the inner loop?

**Q.** Show the referenced items for the first term on the RHS. How many blocks are needed to hold these?

**Q.** What items are referenced by the second term on the RHS? How many blocks needed? For a cache to hold all the items referenced, how many blocks would it need to store at once?

Q. In the above code, which memory references have temporal locality? Which have spatial locality?

Following is a sequence of word-sized memory references (32-bit addresses, 32-bit words, word-addressability). Only the lower 16 bits of each address is shown: assume the upper 16 bits are 0x0040.

0x0001, 0x0134, 0x0212, 0x0001, 0x0135, 0x0213, 0x0162, 0x0161, 0x0002, 0x0044, 0x0041, 0x0221

Suppose we have the choice of either of three direct-mapped cache designs:

(C1) 8 1-word blocks, miss penalty = 25 cycles, hit access time = 2 cycles.
(C2) 4 2-word blocks, miss penalty = 25 cycles, hit access time = 3 cycles.
(C3) 2 4-word blocks, miss penalty = 25 cycles, hit access time = 5 cycles.

**Q.** For each cache, show which references are cache hits and which are misses for a system using that cache. What is the total number of words transmitted between cache and memory?

**Q.** What is the total access time in cycles for a system using each these caches? Which is better?

Suppose a direct-mapped cache uses its address bits in the following way:

| ADDRESS[31:10] | ADDRESS[9:4] | ADDRESS[3:0] |
|---|---|---|
| tag | index | offset |

The memory is byte-addressable.

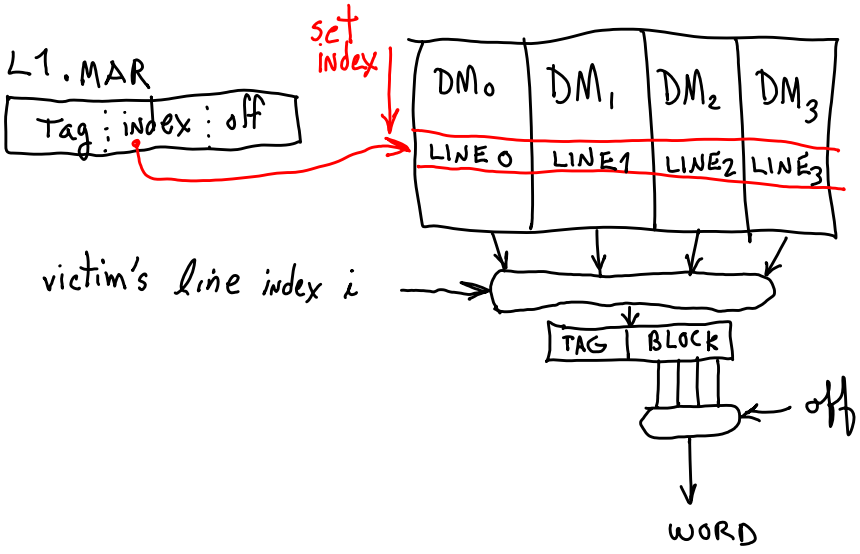
**Q.** What is the cache block size in 32b words? How many entries does the cache have (that is, how many cache blocks does the cache hold)?

**Q.** In total, how many data bits does the cache hold (assuming all entries are valid)? In total, how many bits of storage does the cache require? What is the VLSI area overhead with respect to the storage area for data bits?

**Q.** Suppose we alter the cache to be 8-way set associative without changing its overall size or the size of its cache block. Show the usage of address bits.

Our L1 cache is (write-back, allocate), our L2 cache is (write-through, no-allocate). Both have their own write buffers. L1's write buffer simply passes its writes on to L2 as writes. Since L2 is write-through with a write buffer, it is alternatively called "write-behind". Both use the same block sizes. L1 is 4-way set associative; L2 is 8-way set associative.

**Q.** Write a cache-controller algorithm for handling an L1 write miss. A cache controller is a finite-state machine, which can be specified as an algorithm. Both L1 and L2 have separate controllers that communicate via signals (R/W request, Ready, Address, Data, ...) This algorithm for a write miss would be the specification for a combination of the write-miss portion of the L1 controller and part of the L2 controller.




L1.MAR.index = set index ⇒ which set
i = which line in set
off = which word in block

Note: Not shown is that tags go directly to comparators in parallel without passing through the line-select MUX for determining whether there is a hit or not.

Suppose program P has the following behavior per 1000 instructions executed: data reads = 180, data writes = 120. Suppose P running on system S has 0.2% instruction cache misses and 2% data cache misses. S executes one instruction per cycle, except for memory stalls. All instruction and data accesses are 32b words, and cache blocks are 16B.

**Q.** If S has a (write-through, allocate)-cache without write buffering, what minimum memory bandwidth (bytes per cycle) is needed to guarantee S has an overall CPI of 2? If S's clock rate is 2 GHz, what is the required memory bandwidth in B/sec?

Q. Suppose S is modified to have write buffering. Can this change the required minimum memory bandwidth to get an average CPI = 2?

Q. Suppose system S2 has a (write-back, allocate)-cache with write buffering. Assume 30% of evicted cache blocks are dirty (modified). For the same program P and miss rates, what memory bandwidth is needed to achieve an average CPI of 2?

A "streaming" system typically does many sequential data reads with very little reuse of the same memory location. Prefetching brings in data speculatively, based on memory access patterns, before it is referenced. A stream buffer prefetches data that is sequentially adjacent to the most recent cache block referenced. The stream buffer is logically part of the cache; it is accessed in parallel with the cache. If there is a hit in the buffer, the cache block is moved to the cache proper. If a cache block is accessed that is not adjacent to a block in the buffer, that block in the buffer will be overwritten by the next speculative prefetch.

**Q.** Suppose system S has a 2-line prefetch buffer and that the amount of computation required per cache line takes long enough so that prefetches always complete before the next cache line is requested. Suppose a process running on S accesses a 1/2 MB block in a loop that runs for 100 iterations. The sequence of memory addresses accessed has these offsets from the start of the block,

0, 4, 8, 12, 16, 20, 24, ..., 0, 4, 8, 12, ...

and so on. S has a 64kB DM cache, cache lines are 8 32b words, and the cache is initially cold (all entries are invalid). The memory is byte-addressable. Calculate the miss rate for this job. What would be the miss rate if there were no prefetching?

Q. Suppose S's miss penalty is BS (B) X 20 (cycles/B), for a cache block size of BS bytes. For instance, if BS = 16, S's miss penalty would be 16 B X 20 (cycles/B) = 320 cycles. Suppose job J running on S has a memory access pattern that results in the following miss rates, depending on the cache block size (MR_i means the Miss Rate for BS = i bytes):

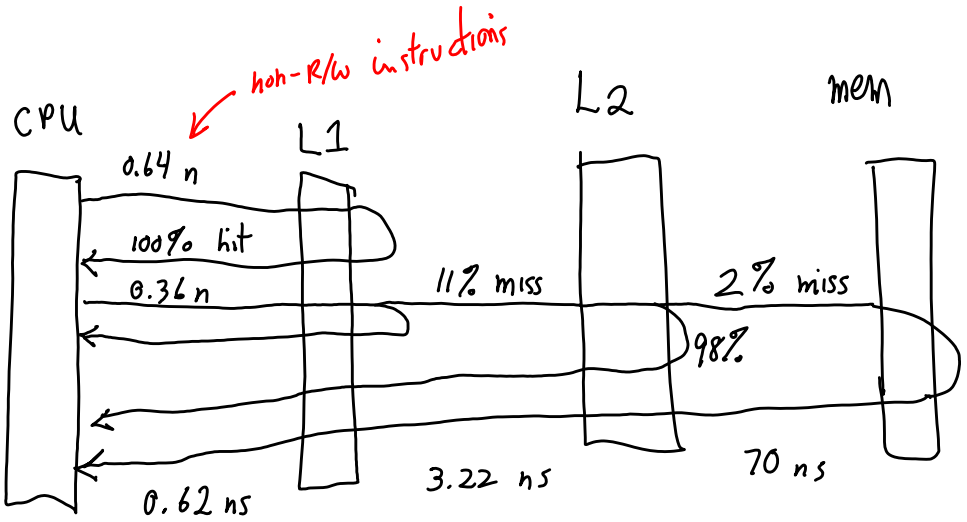
MR_8 = 8%, MR_16 = 3%, MR_32 = 1.8%, MR_64 = 1.5%, MR_128 = 2%

J runs with an average CPI of 1, except for memory stalls, and has an average memory reference rate of 1.35 (mem refs / instr.), which includes both instruction fetches and data read/writes. Find the block size that gives J its best performance. What is J's average CPI? NB--The miss rates include misses that apply to the combined cache and prefetch buffer.

Systems S1 and S2 have a memory access time of 70 ns. Job J has 36% memory accesses for data. Miss rates for J are MR_1k = 11%, MR_2k = 8%. S1's L1 cache is 1kB and access time is 0.62 ns; S2's L1 cache is 2kB with access time of 0.66 ns.

Q. What are the clock rates for the two systems? What is the average memory access time for each? Suppose both have average CPIs of 1 ignoring memory stalls, what are their CPIs?

Q. Suppose each system S1 is given a 512B L2 cache with the following characteristics for J: MR_512k = 2%, access time = 3.22 ns. Which processor is faster?



System S has these characteristics: CPI = 2 (w/o memory stalls), CR = 2 GHz, memory access time = 125 ns, L1 cache MR = 5%.

Q. We are considering two different L2 caches for S: L2a is direct-mapped, has an access time of 15 cycles, and results in a global MR= 3%. L2b is 8-way set associative, has a 25 cycle access time, and results in a 1.8% global MR. (A global MR is the percentage of the total memory references that result in a read or write to main memory.) What are S's CPIs for (1) only an L1 cache, (2) both L1 and L2a, (3) both L1 and L2b?

$$CPI = \left[ 2 \left(\frac{\text{cycle}}{\text{instr}}\right) n_{\text{instr}} + (5\%)\, n \left(125\,\text{ns} \left(\frac{2\text{G cycle}}{\text{sec}}\right)\right)\right]\frac{1}{n} = \frac{(2 + 12.5)\,\text{cycles}}{\text{instr.}} = 14.5$$

Q. If within the next few years we can expect memory speeds to double, which configuration is best? What if processor speed also quadruples?

System S has virtual memory with 4kB pages, 4-entry fully-associative TLB, true LRU replacement. Physical memory that holds is single page is called a page "frame".

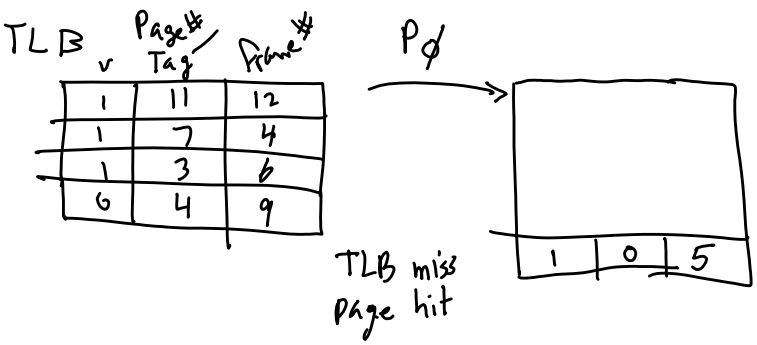
Q. For the sequence of memory references below, the initial TLB content, and the initial page table content, show the final page table, TLB, and for each reference whether it is a TLB hit, a page table hit (page in memory), or a page fault. TLB entries are [valid, tag, #frame]; PT entries are [1, #frame] or [0, d], depending on whether the page is in a physical frame or not (if not, d is some disk address).

Memory references: 4095, 31272, 15789, 15000, 7193, 8912

TLB: [1, 11, 12], [1, 7, 4], [1, 3, 6], [0, 4, 9]
PT: [1, 5], [0, d], [0, d], [1, 6], [1, 9], [1, 11], [0, d], [1, 4], [0, d], [0, d], [1, 3], [1, 12]

If a page must be brought in from disk, assume the free frame with the smallest frame number is used.

4kB pages $\rightarrow 2^{12}$ B pages $\rightarrow$ 12 bits offset, 4096 B per page




System S has 64b virtual addresses, 16GB physical memory, 8kB pages, 8B PT entries.

Q. For a single-level page table scheme, how many page table entries in a page table? How much physical memory would the page table require? What would be the minimum page size to make a single-level page table scheme practical?

$\left(\left(2^{64}\ B/mem\right)/\left(2^{13}\ B/page\right) = 2^{51}\ page\ table\ entries\right) @ 8B = 2^{54}\ B\ table$

$= 16\ Peta\ B$

Q. If we instead use a multi-level page table, with each an 8kB page directory and 8kB sub-directories and sub-tables. That is, each piece of the multi-level page table data structure fits into an 8kB frame. How many levels would be required?

$\left(8kB\ page / 8\ B/entry\right) = 1k\ entries/page$

$\left(2^{10}\ 1^{st}\text{-level entries}\right)\left(2^{10}\ 2^{nd}\text{-level}\right) \times \left(2^{10}\ 3^{rd}\text{-level}\right) \times \left(2^{10}\ 4^{th}\right) \times \left(2^{10}\ 5^{th}\right) \times \left(2^{10}\ 6^{th}\right) \geq 2^{51}$

$\Rightarrow$ 6-levels

System S has a direct-mapped cache write buffer. The cache is write-back. If there is a cache miss and the evicted cache line is modified, the cache will immediately issue a memory read request for the missed cache block and sends the evicted cache block to the write buffer to be handled by a memory bus interface unit whenever the memory bus is free.

Q. Suppose the evicted cache block is being written from the write buffer when another instruction makes a memory reference that hits in the cache. What action should the cache take? What would happen if an instruction referenced the evicted cache block?