Reading: Patterson & Hennessy, Chapter 5.

(1.) C multi-dimensional arrays are stored in row-major order. For example, the n-row by m-column array A would be stored in memory as (smaller memory addresses to the left),

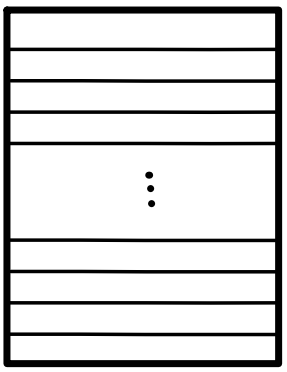
A[0][0], A[0][1], ... A[0][m-1], A[1][0], A[1][1], ... A[1][m-1], ... A[n-1][0], A[n-1][1], ... A[n-1][m-1]

Thus A[k] references an m-element linear array. Here is a bit of C code ("8k" is shorthand for 8192):

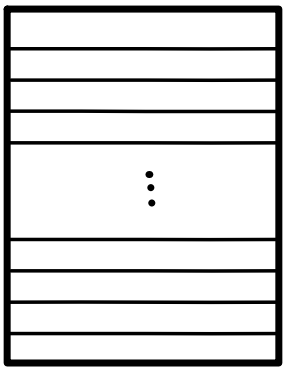
```
int A[8k][8k], B[8k][8k], x, y;

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

Q.1.a. Suppose our cache has 16B blocks and 32-bit ints. How many words (ints) per cache line?

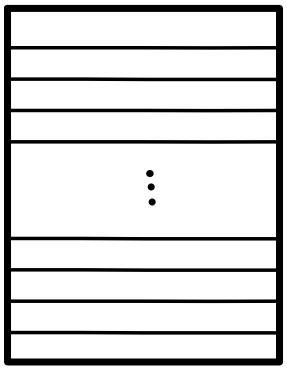
Q.1.b. Suppose the code above executed. How many cache blocks were accessed in writing to A[][] (the LHS of the inner loop)? To get started, a diagram of A[][] is shown below as a block of rows and columns. Show the dimensions of A and the part accessed by the LHS. Indicate the access order.



Q.1.c. Do the same for B. How many blocks were accessed?



Q.1.d. Show the part of A accessed on the RHS. How many blocks were accessed? Indicate the access order.



Q.1.e. Of all memory references for A and B, which have temporal locality when the time window is 8 iterations of the inner loop? When the time window is the entire execution? Which have spatial locality?

(2.) Following is a sequence of word-sized memory references (32-bit addresses, 32-bit words, **word-addressable**). Only the low-order 16 bits of each address is shown: assume the most-significant16 bits are 0x0040.

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

Suppose we have 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.2.a. For each cache, show which references are hits and which are misses, how many words were transfered between cache and memory in total, and the total access time. Which is better?

(3.) 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.

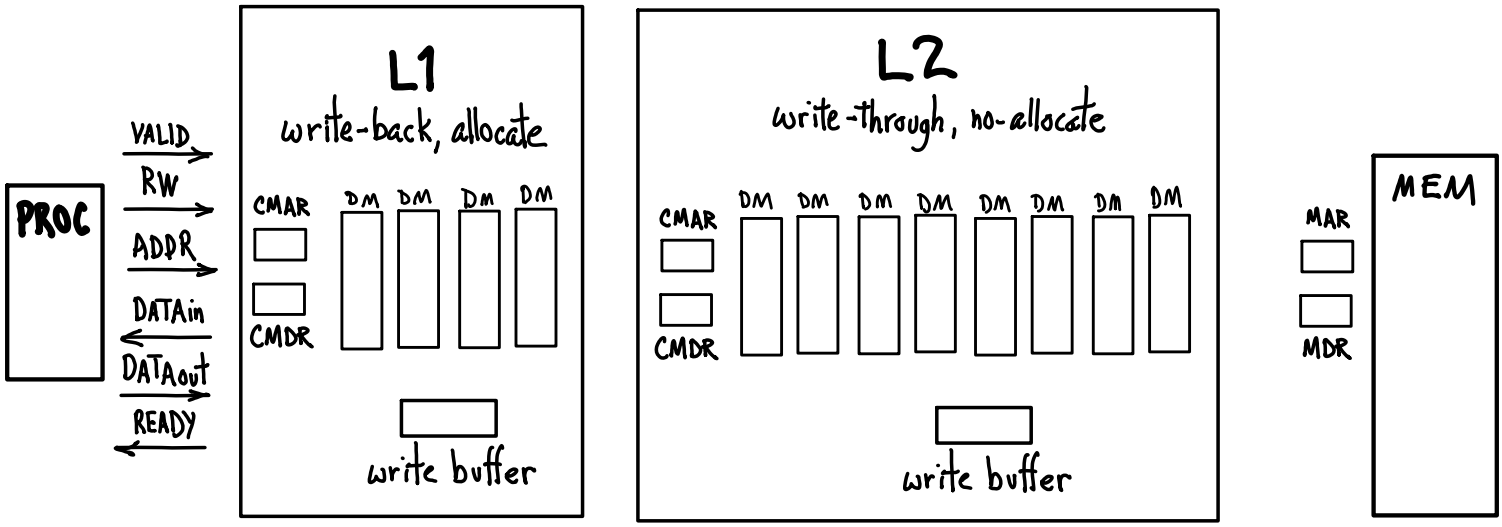
Q.3.a. What is the cache block size in 32-bit words? How many lines does the cache have?

Q.3.b. 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 overhead for tag storage as a percent of data storage?

Q.3.c. Suppose we alter the cache to be 8-way set associative without changing its overall data storage size or the size of cache blocks. Show the usage of address bits.

(4.) Suppose we have a 4-way set-associative L1 cache (write-back, allocate) and an 8-way set-associative L2 cache (write-through, no-allocate). Both have write buffers. L1's write buffer passes its writes to L2. Both have the same block size. The interfaces between both caches, memory, and the processor communicate using the same set of signals, VALID, ADDR, DATAin, DATAout, RW, and READY; e..g., shown below between PROC and L1.

Q.4.a. Write the portions of the cache controllers' algorithms that handle an L1 write miss. This code specifies a portion of both L1's and L2's FSM controllers. Use dot notation for signals, e.g., L1.READY, L2.READY, and so on, with the left-side indicating the signal's sender (e.g., L1 sends READY to PROC and L2 sends READY to L1). Describe your controllers as high-level psuedo-code, or as a logic flow chart, or as a FSM state transition diagram, or any combination of these. Assume L2 hits.



(5.) Suppose program P running on system S has the following behavior: per 1000 instructions executed, data reads = 180, data writes = 120, and cache miss rates are $MR_{instuctions} = 0.2\%$ and $MR_{data} = 2\%$ for both read and write misses. CPI = 1 (which includes cache hit time) except when there is a stall to access memory. To transfer a cache block to or from memory requires t cycles per word plus a latency of $10t$ cycles. A single word access takes $10t + t$ cycles. Instruction and data are 32b words; cache blocks are 16B. S's cache is write-through, allocate on write miss, and does not have a write buffer.

Q.5.a. What is the size of a single data write operation to memory (in words)? How many cycles does this require? What is the size of a single data or instruction read operation from memory (in words)? How many cycles does this require (in terms of t)?

Q.5.b. On average per 1,000 instructions executed, how many words of data are written to memory? How many cycles are needed for these memory accesses?

Q.5.c. Data-write misses must allocate and fetch a block. What is the average data-read traffic for this (measured in blocks and words)? How many cycles needed?

Q.5.d. Data-read misses must allocate and fetch a block. What is the average data-read traffic for this (measured in blocks and words)? How many cycles needed?

Q.5.e. On average per 1,000 instructions, how many instruction fetches miss? What is the average instruction-fetch read traffic to memory (measured in blocks and words)? How many cycles does this contribute to program execution time?

Q.5.f. We want S to have an overall CPI of 2. Show an upper bound on t. S's clock rate is 2 GHz. What is the required memory bandwidth in B/sec?

Q.5.g. A modified version of S, S1, adds a write buffer. Does S1 have more or less or the same total traffic to memory as S?

Q.5.h. Suppose all writes are overlapped with instruction execution: Instructions always execute without stalling whenever the write buffer is busy writing data to memory. Put another way, a cache miss never occurs while the write buffer is busy. Of course, write misses still incur the same penalty as before to read in a block. Per 1,000 instructions executed, how many cycles are now required to execute the program.

Q.5.i. What is the speedup of S1 relative to S?

Q.5.j. To have a CPI = 2, what memory bandwith is required for S1?

Q.5.k. A modified S1, S2, has a write-back cache instead of write-through. For program P running on S2, 25% of evicted cache blocks are dirty (modified). Write misses that have to evict a dirty block write the evicted block to memory. What is the data-write traffic for this new system?

Q.5.l. What memory bandwidth does S2 need to have an average CPI of 2?

(6.) Systems S1 and S2 both have a memory access time of 70 ns. Job J has 36% memory accesses for data. The miss rate for J with a 1kB cache is 11%, with a 2kB cache it is 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.6.a. What is the fastest possible clock rate for S1? for S2?

Q.6.b. What is the average memory access time for S1 when running job J? for S2?

Q.6.c. S1 and S2 both have an average CPI = 1, ignoring memory stalls. What are their CPIs when running job J?

Q.6.d. A modified S1, S1', adds a 1/2 kB L2 cache with access time = 3.22 ns. Running Job J, the L2 miss rate is 2%. For job J, what is the speedup of S1' relative to S2?

(7.) System S running a particular job has these characteristics:
CPI = 2 (w/o memory stalls), CR = 2 GHz, memory access time = 125 ns, MR_{L1} = 7%.

We are considering two different L2 caches for S,
L2a: direct-mapped, access time =15 cycles, MR_{global} = 3%.
L2b: 8-way set associative, access time = 25 cycles, MR_{global} = 1.8%.

Recall that MR_{global} is the percentage of total memory references that result in a main memory access.

Q.7.a. What is S's CPI with only an L1 cache?

Q.7.b. What is the CPI with L1 and L2a?

Q.7.c. With L1 and L2b?

Q.7.d. Suppose S's new memory is twice as fast, which configuration is best?

Q.7.e. What if its CR also quadruples?

(8.) 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.8.a. 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]

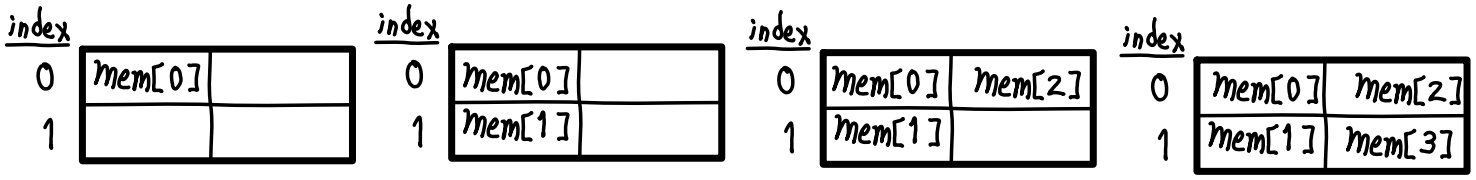
If a page must be brought in from disk, assume the free frame with the smallest frame number is used.

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

Q.9.a. 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?

Q.9.b. 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?

(10.) In the following, a series of memory references are sent to a 2-way set-associative cache. The cache is initially empty (all entries invalid). The initial sequence of block addresses is {0, 1, 2, 3}. They all miss and have the corresponding cache block brought from memory {mem[0], mem[1], mem[2], mem[3]. The cache content after these accesses is shown. Block addresses are in hex.



Q.10.a. The following sequence of block addresses (hex) is sent to the cache. Replacement is LRU: The block whose time of last access is oldest is replaced. Show which are hits.
{4, 0, 2, 8, 10, 12, 14, 16, 0, 1, 3, 5, 1, 3, 1, 3, 5, 3}.

Q.10.b. Suppose you knew ahead of time the entire sequence of addresses sent to the cache. Which block would you replace each time? That is, what blocks would be evicted and replaced to give the best performance?

(11.) Consider a job J running on a machine G has these characteristics:

I/O traps: 30 (per 10k instructions executed)
Traps to priviliged OS mode for non-I/O: 90 (per 10k instructions executed)
Cycles to switch to OS mode per trap: 15 cycles per trap
Average CPI w/o mode switching overhead: 1.5 cycles per instruction
(that is, the CPI for all instructions executed if no mode switches occurred.)
I/O access delay time (w/o trap overhead): 1k cycles per I/O

Q.11.a. What is the CPI for G running job J?

Q.11.b. Suppose we run G virtually on a Virtual Machine Monitor (VMM). G executes its instructions natively on the host hardware (instructions are not simulated/emulated). G runs in user mode, so each trap causes a mode exception trap to VMM. VMM handles the priviliged instruction execution itself, then returns control to G, which adds 175 cycle per mode switch. I/O traps also incur this switching overhead, but the I/O delay time is the same as if G were running w/o VMM. What is the CPI now?