Homework Assignment 10

CS/ECE 3810: Computer Organization
Nov 16, 2020

Memory system and caches

Due Date: Nov 23, 2020
(100 points)

Important Notes:

- Solutions turned in must be your own. Please, mention references (if any) at the end of each question. *Please refrain from cheating.*
- All solutions must be accompanied by the equations used/logic/intermediate steps. Writing only the final answer will receive **zero** credits.
- Partial score of every question is dedicated to each correct final answer provided by you. Please ensure both your equation/logic and final answer are correct. Moreover, you are expected to provide explanation for your solutions.
- All units must be mentioned wherever required.
- Late submissions **(after 11:59PM on 11/23/2020)** will not be accepted.
- We encourage all solutions to be typed in for which you could use software programs like LaTeX, Microsoft Word etc. If you submit handwritten solutions, they must be readable by the TAs to receive credits.

Memory System. A memory system is crucial to the system performance and energy-efficiency of modern computers. Memory technologies, such as DRAM, SRAM, and the emerging resistive RAMs, offer trade-offs among capacity, speed, and cost of the memory systems.

Question 1. Explain the following concepts involved in understanding computer performance. [Refer to Slide #5, #14, and #15 of the Memory system lecture, Section 5.1 of the textbook, Week-11 of the IVC meetings.]
i) Memory wall. **(6 points)**

The memory wall refers to the processor-memory performance gap. This gap increases by about 50% per year.
Processor performance historically improves by about 60% per year while main memory access time improves by about 5% per year.

ii) Temporal locality in a cache with an example. **(7 points)**

Temporal locality: the probability of accessing $A + \epsilon$ at time $t + \delta$ is highest when $\delta \to 0$.

For example, a cache will use temporal locality in an application when executing a for loop. Temporal locality will be used to access the index variable of the for loop much faster.

iii) Spatial locality in a cache with an example. **(7 points)**

Spacial locality: the probability of accessing $A + \delta$ at time $t + \epsilon$ is highest when $\delta \to 0$.

For example, a cache will use spatial locality when iterating through an array. Spatial locality will be used to access neighboring elements in the array much faster.

Average memory access time (AMAT). It is a common metric used for analyzing the performance of memory systems. As it was mentioned in the class, a simplified version of AMAT uses hit time (t_h), miss penalty (t_p), and miss rate (r_m) to measure memory performance.

Question 2. Consider a processor comprising an L1 Cache and main memory. The hit rate for the L1 cache is 85%, and the hit time is 2 cycles. It takes 200 cycles to access the main memory. Compute the average memory access time (AMAT) for this processor.
(20 points)

$AMAT = t_h + r_m t_p$

$AMAT = 2 + 0.15 * 200 = 32$ cycles is the average memory access time.

Caches Modern CPU cores from ultra-low power chips like the ARM Cortex-A5 to the highest-end Intel Core i9 employ caches. Most CPUs have different independent caches, including instruction and data caches, where the data cache is usually organized as a hierarchy of more cache levels (L1, L2, L3, L4, etc.). The operation of a particular cache can be specified entirely by the cache size, the cache block size, the number of blocks in a set, the cache block replacement policy, and the cache write policy (write-through or write-back).

Question 3. Consider a processor using a 2MB direct-mapped cache with 32B cache blocks. Assume that the tag size is 14 bits. Compute the size of the main memory of this processor.
(15 points)

main memory size = tag bits + index bits + offset bits

main memory size = $14 + 19 + 5 = 38$ where the offset bits (5) come from the 32B cache blocks and the index bits (19) come from the direct-mapped cache size divided by the size of a cache block.

Thus, main memory size is $2^{38} = 274877906944$ bytes or 34.359738368 GB.

Question 4. Consider a processor using a 32KB direct-mapped cache with 32B cache blocks. Assume that the processor can address up to 4GB of main memory. Compute the size of all address fields: byte offset, index, and tag.
(15 points)

4 GB = 2^{32} ⇒ 32 address bits.

32 B = 2^5 ⇒ 5 offset bits.

32 MB /32 B = 2^{17} ⇒ 17 index bits.

32 - 5 - 17 = 10 tag bits.

Cache terminologies.
i) Cache block - The basic unit for cache storage. May contain multiple bytes/words of data.
ii) Cache set - A "row" in the cache with multiple blocks. The number of blocks per set is determined by the layout of the cache (e.g. direct mapped, set-associative, or fully associative).
iii) Tag - A unique identifier for a group of data. Because different regions of memory may be mapped into a block, the tag is used to differentiate between them.
iv) Valid bit - A bit flag that indicates whether the data in a block is valid (1) or not (0).

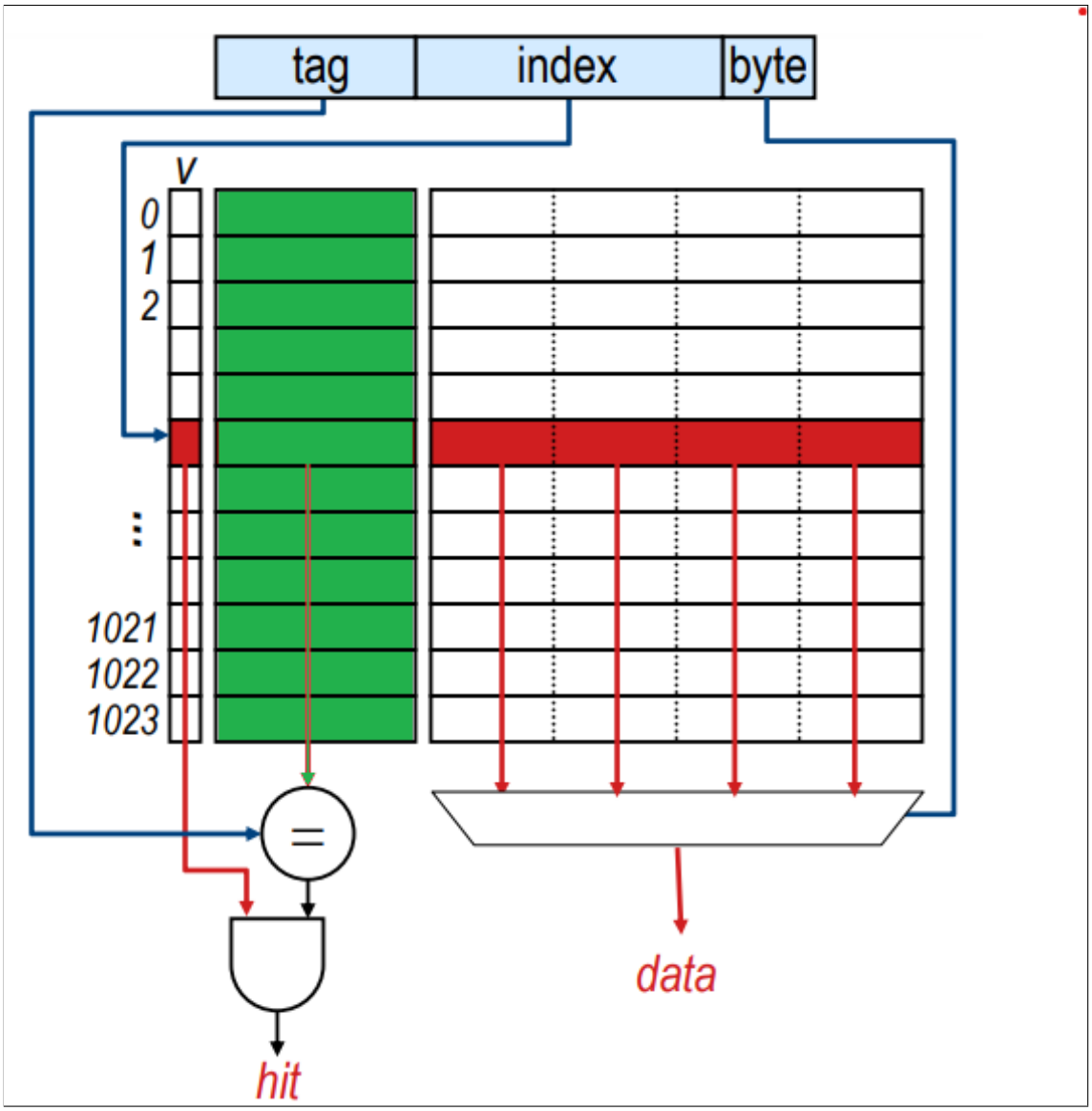

Figure 1: Cache

In this figure, assuming we have 10 bits for the index field, we have 2^{10} cache rows, with indices running from 0 to 1023. Each row of the cache has 3 different parts, data, tag and valid bit flag. Assume that we have 8 bits for the tag field, then we have 8 times 2^{10} bits = 2^{10} bytes = 1KB as the size of the tag array. [Filled with green color in the figure.]

Question 5. Consider an 8 megabyte (MB), 8-way set associative cache array with 64-byte cache blocks.
(a) How many sets are present in this cache?
(b) Assuming that the CPU provides 32-bit addresses, what is the size of all address fields: byte offset, index, and tag.?
(c) How large is the tag array?
(30 points)

(a) number of sets $= \dfrac{32 * 1024}{8 * 64} = 64$ sets.

(b) 32 bit addresses is given.

64 B = 2^6 $\Rightarrow$ 6 offset bits.

8 MB /64 B = 2^{17} $\Rightarrow$ 17 index bits.

32 - 6 - 17 = 9 tag bits.

(c) tag array size = number of sets x number of ways x tag bits:

$64 * 8 * 9 = 4$ KB