Michael Lankford

Homework 8

1. Definitions

* Locality principles and its types
  + Temporal Locality
    - If a data location is referenced, then it will tend to be referenced again soon.
    - Locality in time.
  + Spatial Locality
    - If a data location is referenced, data locations with nearby addresses will tend to be referenced soon.
    - Locality in space.
* Hit rate
  + The fraction of memory accesses found in a level of the memory hierarchy.
* Miss rate
  + The fraction of memory accesses not found in a level of the memory hierarchy.
  + The equation is (1-hit rate)
* Hit time
  + The time to access a level of the memory hierarchy.
  + Includes the time needed to determine whether the access is a hit or miss.
* Miss penalty
  + The time to replace a block in a level with the corresponding block from the lower level, plus the time to deliver this block to the processor.
* Page fault
  + Occurs when an accessed page is not present in main memory.
* Cache topologies
  + Direct-mapped
    - Cache structure where every memory location is mapped to exactly one location in the cache.
  + Set associative
    - Cache structure where every memory location is mapped to a fixed number of locations where each block can be placed.
  + Fully associative
    - Cache structure where every memory location is mapped to any location in the cache.
* Write-through and write-back policies
  + Write-through
    - Scheme where writes always update both the cache and the next lower level of the memory hierarchy, ensuring data is consistent between the two.
  + Write-back
    - Scheme where writes update values only to the block in the cache, then writing the modified block to the lower level of the hierarchy when the block is replaced.
* Cache replacement policies
  + LRU
    - Least recently used.
    - The block replaced is the one that has been unused for the longest time.
  + FIFO
    - First-In, First-Out
    - The block replaced is the one that was first in.
  + Random
    - The block replaced is randomly chosen.
* Virtual address and physical address
  + Virtual address
    - Address that corresponds to a location in virtual space and is translated by address mapping to a physical address when memory is accessed.
  + Physical address
    - Address in main memory.
* Page table
  + Table containing the virtual to physical address translations in a virtual memory system
* TLB
  + Translation-lookaside buffer
  + Cache that keeps track of recently used address mappings to try to avoid an access to the page table.
* Types(sources) of cache misses
  + Compulsory misses
    - Misses caused by the first access to a block that has never been in the cache.
  + Capacity misses
    - Misses caused when the cache cannot contain all the blocks needed during execution of a program.
    - Occur when blocks are replaced and then later retrieved.
  + Conflict misses
    - Misses that occur in set-associative or direct-mapped caches when multiple blocks compete for the same set.

1. Exercise 5.2 - Caches are important to providing a high-performance memory hierarchy to processors. Below is a list of 32-bit memory address references, given as word addresses. 0x03, 0xb4, 0x2b, 0x02, 0xbf, 0x58, 0xbe, 0x0e, 0xb5, 0x2c, 0xba, 0xfd
   1. Exercise 5.2.1 - For each of these references, identify the binary address, the tag, and the index given a direct-mapped cache with 16 one-word blocks. Also list if each reference is a hit or a miss, assuming the cache is initially empty.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Word Address | Binary Address | Tag | Index | Hit/Miss |
| 0x03 | 0000 0000 | 0 | 3 | M |
| 0xb4 | 1011 0100 | B | 4 | M |
| 0x2b | 0010 1011 | 2 | B | M |
| 0x02 | 0000 0010 | 0 | 2 | M |
| 0xbf | 1011 1111 | B | F | M |
| 0x58 | 0101 1000 | 5 | 8 | M |
| 0xbe | 1011 1110 | B | E | M |
| 0x0e | 0000 1110 | 0 | E | M |
| 0xb5 | 1011 0101 | B | 5 | M |
| 0x2c | 0010 1100 | 2 | C | M |
| 0xba | 1011 1010 | B | A | M |
| 0xfd | 1111 1101 | f | D | M |

* 1. Exercise 5.2.2 - For each of these references, identify the binary address, the tag, and the index given a direct-mapped cache with two-word blocks and a total size of 8 blocks. Also list if each reference is a hit or a miss, assuming the cache is initially empty.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Word Address | Binary Address | Tag | Index | Hit/Miss | |
| 0x03 | 0000 0000 | 0 | 3 | 1 | M |
| 0xb4 | 1011 0100 | B | 4 | 0 | M |
| 0x2b | 0010 1011 | 2 | B | 1 | M |
| 0x02 | 0000 0010 | 0 | 2 | 0 | H |
| 0xbf | 1011 1111 | B | F | 1 | M |
| 0x58 | 0101 1000 | 5 | 8 | 0 | M |
| 0xbe | 1011 1110 | B | E | 0 | H |
| 0x0e | 0000 1110 | 0 | E | 0 | M |
| 0xb5 | 1011 0101 | B | 5 | 1 | H |
| 0x2c | 0010 1100 | 2 | C | 0 | M |
| 0xba | 1011 1010 | B | A | 0 | M |
| 0xfd | 1111 1101 | f | D | 1 | M |

1. Exercise 5.3.1 - By convention, a cache is named according to the amount of data it contains (i.e., a 4 KiB cache can hold 4 KiB of data); however, caches also require SRAM to store metadata such as tags and valid bits. For this exercise, you will examine how a cache’s configuration affects the total amount of SRAM needed to implement it, as well as the performance of the cache. For all parts, assume that the caches are byte addressable, and that addresses and words are 64 bits. Calculate the total number of bits required to implement a 32 KiB cache with two-word blocks.

1 word = 8 bytes

2 words = 16 bytes = 2^4 bytes

32 KiB = 2^15 bytes 🡪 2^15 / 2^4 = 2^11 lines of data

2^15 \* 8 bits of data + 2^11 \* 49 bits of tag + 2^11 \* 1 valid bits = 364,544 bits

1. Exercise 5.5 - For a direct-mapped cache design with a 32-bit address, the following bits of the address are used to access the cache.

|  |  |  |
| --- | --- | --- |
| Tag | Index | Offset |
| 31-10 | 9-5 | 4-0 |

* 1. Exercise 5.5.1 - What is the cache block size (in words)?

32 bytes / 8 = 4 words

* 1. Exercise 5.5.2 - How many entries does the cache have?

5 index bits so 2^5 = 32 lines in the cache

* 1. Exercise 5.5.3 - What is the ratio between total bits required for such a cache implementation over the data storage bits?

32 lines \* 4 words/block \* 8 bytes/word = 1024 bytes = 8192 bits

8192 + 54 tag bits \* 32 + 1 valid bit \* 32 = 9952 bits

9952 / 8192 = 1.21 ratio

1. Exercise 5.10
   1. Exercise 5.10.1
   2. Exercise 5.10.2
   3. Exercise 5.10.3
2. Exercise 5.18.1
3. Exercise 5.20.1