**Department of Computer Science**

**National Tsing Hua University**

**CS4100 Computer Architecture**

**Spring 2023 Homework 6**

Deadline: 2023/06/09 23:59

1. (15 points) Please run the simulation to find the following cases. For (a)~(c) case, please show the screenshots of (i) the load/store instruction at which you find the case, (ii) accessed memory address, and (iii) cache states before and after that load/store instruction. Also, please give a simple explanation of what happened for each case. You may follow the example question we made to answer the following cases.
2. (3 points) One case of cache insertion at an index (a set) with exactly one way already occupied. Don’t forget to explain the change of LRU bits of both ways in that set.
3. (3 points) One case of a hit that makes a block become dirty. (Not dirty before that hit)
4. (3 points) One case of a replacement with write-back needed.
5. (6 points) Improve the hit rate by designing your own cache. You may adjust the associativity or the cache size. Please give some screenshots to show (i) the cache you design and (ii) the improvement of the hit rate. A brief discussion about why the cache you design makes the hit rate higher is also needed.
6. (12 points) Consider a 10-bit data, 𝐷 = {𝑑1, 𝑑2, 𝑑3, 𝑑4, 𝑑5, 𝑑6, 𝑑7, 𝑑8, 𝑑9, 𝑑10}, to be protected by using a Hamming single error correcting code with an additional parity bit, 𝑝11, so that double errors can be detected.
7. (2 points) Let the encoded codeword 𝐶 = {𝑝1, 𝑝2, 𝑑1, 𝑝4, 𝑑2, 𝑑3, 𝑑4, 𝑝8, 𝑑5, 𝑑6, 𝑑7, 𝑑8, 𝑑9, 𝑑10, 𝑝11}. Show the encoding process of 𝐷 =1001011001 to obtain the encoded codeword 𝐶.
8. (2 points) Assume no error occurs. Show the decoding process of 𝐶 (i.e., locating/correcting the Screenshots Explanation Instruction: Cache states before and after the write miss 3 single error, detecting double errors, or confirming that there is no error; you have to verify if the result is right or wrong) from (a).
9. (2 points) From (a), suppose 𝑑5 of 𝐶 is inverted. Show the decoding process.

d5 is bit 9.

1. (2 points) From (a), suppose 𝑝1 and 𝑑8 of 𝐶 are inverted. Show the decoding process.

p1 is bit 1 and d8 is bit 12.

1. (2 points) From (a), suppose 𝑝1, 𝑑8, and 𝑝11 of 𝐶 are inverted after the encoding. Show the decoding process.
2. (2 points) What is the minimum number of parity bits required to protect 128-bit data using the same method to construct a SECDED code?
3. (12 points) For a direct-mapped cache design with a 14-bit memory address, the following bit fields of the address are used to access the cache.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Tag | Index | Block Offset | Byte Offset |
| Bits | 13:9 | 8:4 | 3:2 | 1:0 |

1. (2 points) What is the cache block size (in bytes)?
2. (2 points) How many blocks does the cache have?
3. (2 points) What is the total cache size (in bytes)?
4. (2 points) What is the actual SRAM memory size used to implement this cache in bytes? Please also consider the valid bit, dirty bit, and tag bits for each block.
5. (2 points) With the same cache size in (c), derive the bit fields of the address for a 2-way set associative cache design.
6. (2 points) From (e), what is the actual SRAM memory size used to implement this cache in bytes? Please also consider the valid bit, dirty bit, and tag bits for each block; an extra reference bit for each set.
7. (8 points) Consider the cache architecture of a specific processor that its base CPI without memory stalls is 1. Suppose that the L-1 miss rate is 5%. The main memory access delay is 200 cycles. There are several options of L-2 and L-3 caches in the following table.

|  |  |  |  |
| --- | --- | --- | --- |
| Level | Name (Placement) | Access Cycle | Local Miss Rate |
| L-2 | L2-DM (Direct Mapped) | L2 access cycle = 16 | L2 Miss Rate = 4.0% |
| L-2 | L2-4WAY (4-way Set Associative) | L2 access cycle = 20 | L2 Miss Rate = 3.5% |
| L-3 | L3-8WAY (8-way Set Associative) | L3 access cycle = 50 | L3 Miss Rate = 2.0% |

Please note the following definitions:

* Local Miss Rate: the fraction of references to one level of a cache that miss. That is, the number of misses in a cache divided by the total number of memory accesses to that cache.
* Global Miss Rate: the fraction of references that miss in all levels of a multilevel cache. That is, the number of misses in the cache divided by the total number of memory accesses generated by the CPU.

1. (5 points) Calculate the effective CPI for each possible multilevel (i.e., 2-level or 3-level) cache design. Which one provides the best performance based on the effective CPI?

|  |  |
| --- | --- |
| L2-DM |  |
| L2-4WAY |  |
| L2-DM + L3-8WAY |  |
| L2-4WAY + L3-8WAY |  |

1. (3 points) From (a), consider the multilevel cache design with the best performance. Suppose you want to further reduce the effective CPI to 1.8 or lower by improving the L-2 cache with fewer access cycles (assuming the local miss rate remains the same). What are the maximum access cycles of this improved L-2 cache to meet this requirement?
2. (23 points) Consider that a computer P uses a 128-byte 2-way set associative cache as a primary cache. The cache adopts the write-back and write-allocate policies. Each block in the cache has 16 bytes. The physical byte address of the computer has 12 bits. The block replacement policy is LRU (Least-Recently Used).

Suppose the cache receives the following memory accesses. Show the content of the cache after each memory access. Indicate if it is a read/write miss/hit for each access. Also, indicate if there occurs a block replacement, write allocate, or write back.

|  |  |  |
| --- | --- | --- |
| No. | Read or Write | Address |
| 1 | Read | 0x340 |
| 2 | Read | 0x000 |
| 3 | Read | 0x1d8 |
| 4 | Write | 0x354 |
| 5 | Read | 0xa61 |
| 6 | Write | 0xa61 |
| 7 | Read | 0x3ec |
| 8 | Read | 0xa62 |
| 9 | Read | 0x3ea |
| 10 | Read | 0x422 |

Suppose initially the cache has one entry as shown in the following table. For each block (way), the **V** denotes the valid bit (V=1 indicates a valid block; V=0 or a blank field indicates an invalid block), and the **D** is the dirty bit (D=1 indicates a dirty block). For each set, the **R** denotes the reference bit (R=0 indicates that Way 0 is the least recently used between the two ways in the same set; R=1 indicates that Way 1 is the least recently used).Additionally, **Mem[0x000-00f]** represents a data block of 16 bytes from **Mem[0x000]** to **Mem[0x00f]**. Note that the empty tag for the existing entry in Set 0 is also for you to fill in.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Way 0 | | | | Way 1 | | | |
|  | R | V | D | Tag | Data | V | D | Tag | Data |
| Set 0 | 1 | 1 | 0 | 0000002 | Mem[0x000-00f] |  |  |  |  |
| Set 1 |  |  |  |  |  |  |  |  |  |
| Set 2 |  |  |  |  |  |  |  |  |  |
| Set 3 |  |  |  |  |  |  |  |  |  |

Read 0x340:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Way 0 | | | | Way 1 | | | |
|  | R | V | D | Tag | Data | V | D | Tag | Data |
| Set 0 |  |  |  |  |  |  |  |  |  |
| Set 1 |  |  |  |  |  |  |  |  |  |
| Set 2 |  |  |  |  |  |  |  |  |  |
| Set 3 |  |  |  |  |  |  |  |  |  |

Read 0x000:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Way 0 | | | | Way 1 | | | |
|  | R | V | D | Tag | Data | V | D | Tag | Data |
| Set 0 |  |  |  |  |  |  |  |  |  |
| Set 1 |  |  |  |  |  |  |  |  |  |
| Set 2 |  |  |  |  |  |  |  |  |  |
| Set 3 |  |  |  |  |  |  |  |  |  |

Read 0x1d8:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Way 0 | | | | Way 1 | | | |
|  | R | V | D | Tag | Data | V | D | Tag | Data |
| Set 0 |  |  |  |  |  |  |  |  |  |
| Set 1 |  |  |  |  |  |  |  |  |  |
| Set 2 |  |  |  |  |  |  |  |  |  |
| Set 3 |  |  |  |  |  |  |  |  |  |

Write 0x354:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Way 0 | | | | Way 1 | | | |
|  | R | V | D | Tag | Data | V | D | Tag | Data |
| Set 0 |  |  |  |  |  |  |  |  |  |
| Set 1 |  |  |  |  |  |  |  |  |  |
| Set 2 |  |  |  |  |  |  |  |  |  |
| Set 3 |  |  |  |  |  |  |  |  |  |

Read 0xa61:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Way 0 | | | | Way 1 | | | |
|  | R | V | D | Tag | Data | V | D | Tag | Data |
| Set 0 |  |  |  |  |  |  |  |  |  |
| Set 1 |  |  |  |  |  |  |  |  |  |
| Set 2 |  |  |  |  |  |  |  |  |  |
| Set 3 |  |  |  |  |  |  |  |  |  |

Write 0xa61:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Way 0 | | | | Way 1 | | | |
|  | R | V | D | Tag | Data | V | D | Tag | Data |
| Set 0 |  |  |  |  |  |  |  |  |  |
| Set 1 |  |  |  |  |  |  |  |  |  |
| Set 2 |  |  |  |  |  |  |  |  |  |
| Set 3 |  |  |  |  |  |  |  |  |  |

Read 0x3ec:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Way 0 | | | | Way 1 | | | |
|  | R | V | D | Tag | Data | V | D | Tag | Data |
| Set 0 |  |  |  |  |  |  |  |  |  |
| Set 1 |  |  |  |  |  |  |  |  |  |
| Set 2 |  |  |  |  |  |  |  |  |  |
| Set 3 |  |  |  |  |  |  |  |  |  |

Read 0xa62:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Way 0 | | | | Way 1 | | | |
|  | R | V | D | Tag | Data | V | D | Tag | Data |
| Set 0 |  |  |  |  |  |  |  |  |  |
| Set 1 |  |  |  |  |  |  |  |  |  |
| Set 2 |  |  |  |  |  |  |  |  |  |
| Set 3 |  |  |  |  |  |  |  |  |  |

Read 0x3ea:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Way 0 | | | | Way 1 | | | |
|  | R | V | D | Tag | Data | V | D | Tag | Data |
| Set 0 |  |  |  |  |  |  |  |  |  |
| Set 1 |  |  |  |  |  |  |  |  |  |
| Set 2 |  |  |  |  |  |  |  |  |  |
| Set 3 |  |  |  |  |  |  |  |  |  |

Read 0x422:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Way 0 | | | | Way 1 | | | |
|  | R | V | D | Tag | Data | V | D | Tag | Data |
| Set 0 |  |  |  |  |  |  |  |  |  |
| Set 1 |  |  |  |  |  |  |  |  |  |
| Set 2 |  |  |  |  |  |  |  |  |  |
| Set 3 |  |  |  |  |  |  |  |  |  |

1. (10 points) A byte-addressable virtual memory system has the following characteristics:

* Each virtual address is 16 bits.
* Each physical address is 14 bits.
* The page size is 512 bytes.
* Each process has a one-level page table, and each page table entry (PTE) is 4 bytes.
* The TLB is directed-mapped and has 16 entries

1. (2 points) How many bits are required for the page offset?
2. (2 points) What is the maximum number of virtual pages a process could have?
3. (2 points) How many physical pages are there?
4. (2 points) What is the maximum page table size for a process?
5. (2 points) Assuming each entry of the TLB has a valid bit, a dirty bit, a reference bit, a tag, and a physical page number, what is the total number of bits in the TLB?
6. (20 points) The following addresses constitute a stream of virtual byte addresses generated by a processor.

0x5368, 0x02c3, 0x434b, 0x6812, 0xaf50

Assume that the main memory has adequate space to accommodate the requested physical page brought from the disk without page replacement. Therefore, you can choose any page number as long as it is not used in the page table. Suppose that a page has 4KB. A four-entry fully associative TLB with the approximate LRU replacement policy (see the appendix) is utilized.

The initial TLB and page table states are listed below. Suppose the ***replacement pointer*** points to the first block.

|  |  |  |  |
| --- | --- | --- | --- |
| Valid | Tag | Physical Page Number | Reference (Used)  ← Replacement Pointer |
| 1 | 0x4 | 6 | 1 |
| 1 | 0x1 | 2 | 0 |
| 1 | 0xa | 3 | 1 |
| 0 | 0x3 | 5 | 0 |

|  |  |  |
| --- | --- | --- |
| Index | Valid | Physical Page Number or in Disk |
| 0 | 0 | Disk |
| 1 | 1 | 2 |
| 2 | 0 | Disk |
| 3 | 1 | 5 |
| 4 | 1 | 6 |
| 5 | 1 | 11 |
| 6 | 1 | 7 |
| 7 | 0 | Disk |
| 8 | 0 | Disk |
| 9 | 0 | Disk |
| a | 1 | 3 |
| b | 0 | Disk |

For each access shown above, list whether the access is a hit or miss in the TLB, whether the access

causes a page fault, and the updated state of the TLB.

Appendix: An approximate LRU can be implemented as follows:

* A ***used bit*** (initial 0) is associated with every block.
* When a block is accessed (hit or miss), its used bit is set to 1.
* On a replacement, a ***replacement pointer*** circularly scans through the blocks to find a block with its used bit = 0 to replace.
  + The replacement pointer points to the first block initially, and scans circularly through the blocks in a set during the operation.
* Along the way, the replacement pointer also resets the encountered used bits from 1 to 0.
  + Alternatively, if on an access, all other used bits in a set are 1, they are reset to 0 except the bit of the block that is accessed.

0x5368:

← Replacement Pointer

|  |  |  |  |
| --- | --- | --- | --- |
| Valid | Tag | Physical Page Number | Reference (Used) |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

0x02c3:

|  |  |  |  |
| --- | --- | --- | --- |
| Valid | Tag | Physical Page Number | Reference (Used)  ← Replacement Pointer |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

0x434b:

|  |  |  |  |
| --- | --- | --- | --- |
| Valid | Tag | Physical Page Number | Reference (Used)  ← Replacement Pointer |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

0x6812:

|  |  |  |  |
| --- | --- | --- | --- |
| Valid | Tag | Physical Page Number | Reference (Used)  ← Replacement Pointer |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

0xaf50:

|  |  |  |  |
| --- | --- | --- | --- |
| Valid | Tag | Physical Page Number | Reference (Used)  ← Replacement Pointer |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |