Homework Problem Set #3

Joshua Landron

CSS422

**Q1. Cache and Memory mapping (6 points)**

Suppose a byte-addressable memory has a total memory capacity of 2M bytes and the cache consists of 64 blocks, where each block contains 32 bytes.

1. Direct Mapping

  1) Divide the bits into tag, block and offset bits.

2M -> 2^21, so 21 bits needed to address all bytes

2M bytes/(64 blocks \* 32 bytes/block) = 1K blocks in main memory

->2^10 -> 10 bits for page#

64 blocks per page->2^6 blocks -> 6 bits for block

32 bytes per block -> 2^5 bytes -> 5 bits for offset

Answer:

Tag block offset

|  |  |  |
| --- | --- | --- |
| 10 bits | 6 bits | 5 bits |

  2) What is the tag, line and offset for the address $123A63, in hexadecimal?

123A63->000|1 0010 0011 1010 0110 0011

|  |  |  |
| --- | --- | --- |
| 1 0010 0011 1 | 010 011 | 0 0011 |

Into Hex:

|  |  |  |
| --- | --- | --- |
| 0010 0100 0111 -> 247 | 0001 0011 -> 13 | 0000 0011 -> 03 |

Answer:

          tag: 0x247

line: 0x13

offset:   0x03

2. Fully Associative Mapping

  1) Divide the bits into tag and offset bits.

 Offset is the same as above -> 32bytes per block -> 5 bits

Tag needs to be able to reference all 2M/32 blocks -> 64K blocks in main memory -> 64K -> 2^16 -> 16 bits

Answer:

Tag offset

|  |  |
| --- | --- |
| 16 bits | 5 bits |

  2) What is the tag and offset for the address $123A63, in hexadecimal?

123A63->000|1 0010 0011 1010 0110 0011

|  |  |
| --- | --- |
| 1 0010 0011 1010 011 | 0 0011 |

Into Hex:

|  |  |
| --- | --- |
| 1001 0001 1101 0011 -> 91D3 | 0000 0011 -> 03 |

Answer:

          tag: 0x913D

offset:   0x03

3. 4-way set associative mapping

  1) Divide the bits into tag, set and offset bits

 Sets direct mapping in pages of the size of each set in cache

4-way -> 4 sets in cache -> 64 / 4 = 16 blocks per set -> 2^4 -> 4 bits set

Offset is the same as above -> 5 bit offset

21 bits total – 4 bits set – 5 bit offset = 12 bit tag

Tag block offset

|  |  |  |
| --- | --- | --- |
| 12 bits | 4 bits | 5 bits |

  2) What is the tag, set and offset for the address $123A63, in hexadecimal?

123A63->000|1 0010 0011 1010 0110 0011

|  |  |  |
| --- | --- | --- |
| 1 0010 0011 101 | 0 011 | 0 0011 |

Into Hex:

|  |  |  |
| --- | --- | --- |
| 1001 0001 1101 ->91D | 0011 -> 3 | 0000 0011 -> 03 |

Answer:

              tag:       0x91D

              set:       0x3

              offset:   0x03

**Q2. Cache hit and miss (3 points)**

Suppose we have a computer that uses a memory with a total memory capacity of 256 bytes. The computer has a 16-byte direct-mapped cache with 4 bytes per block. The computer accesses a number of memory locations throughout the course of running a program. Here is the memory addresses in this exact order: **0x91, 0xA8, 0xA9, 0xAB, 0xAD, 0x93, 0x6E, 0xB9, 0x17, 0xE2, 0x4E, 0x4F, 0x50, and 0xA4.** The cache Tag and Block information has been filled out as shown below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tag (binary) | Block # | offset 0 | offset 1 | offset 2 | offset 3 |
| 1110 | 0 |  |  |  |  |
| 0001 | 1 |  |  |  |  |
| 1011 | 2 |  |  |  |  |
| 0110 | 3 |  |  |  |  |

1. What is the hit ratio for the entire memory reference sequence (given in bold)?

Going through all memory accesses and replacing the block as needed:

0x91 -> block 0, page 9, miss -> replace

0xA8 -> block 2, page A, miss -> replace

0xA9 -> block 2, page A, hit -> no action

0xAB -> block 2, page A, hit -> no action

0xAD -> block 3, page A, miss -> replace

0x93 -> block 0, page 9, hit -> no action

0x6E -> block 3, page 6, miss -> replace

0xB9 -> block 2, page B, miss -> replace

0x17 -> block 1, page 1, hit -> no action

0xE2 -> block 0, page E, miss -> replace

0x4E -> block 3, page 4, miss -> replace

0x4F -> block 3, page 4, hit -> no action

0x50 -> block 0, page 5, miss -> replace

0xA4 -> block 1, page A, miss -> replace

Hit Ratio = 5 / 14 = 0.36

2. What memory blocks will be in the cache after the last address has been assessed? Please fill in the Tag and Block first. Then, fill the actual address value for each offset location in the corresponding cell.

|  |  |  |
| --- | --- | --- |
| Tag (binary) | Block # | Offset for address |
| 0101 | 0 -> 00 | 00 |
| 1010 | 1 -> 01 | 00 |
| 1010 | 2 -> 10 | 01 |
| 0100 | 3 -> 11 | 11 |

**Q3.  Virtual memory and cache (6 points)**

Consider a processor with the following memory hierarchy:

256K virtual address space (byte addressable)

128K physical address space, each page (frame) has 32K bytes (byte addressable)

2Kbyte direct-mapped cache, a block (refill line) has 256 bytes

The machine uses a two entry TLB.

All replacement policies are LRU. There are two LRU stack.

The entry of these stacks are the page number of a virtual memory.

Note that all the values are represented as hexadecimal.

**TLB**

|  |  |  |
| --- | --- | --- |
| Virtual page # | Physical page # | Valid |
| 5 | 3 | 1 |
| 0 | 2 | 1 |

**TLB LRU stack**

|  |
| --- |
| 0 |
| 5 |

**Page Table**

|  |  |  |
| --- | --- | --- |
| Virtual page # | Physical page # | Valid |
| 0 | 2 | 1 |
| 1 | 1 | 1 |
| 2 | --- | 0 |
| 3 | --- | 0 |
| 4 | 0 | 1 |
| 5 | 3 | 1 |
| 6 | --- | 0 |
| 7 | --- | 0 |

**Mem LRU stack**

|  |
| --- |
| 0 |
| 5 |
| 4 |
| 1 |

**cache**

|  |  |  |
| --- | --- | --- |
| Line # | Tag | Data block |
| 0 | 10 | \* |
| 1 | 0A | \* |
| 2 | 3C | \* |
| 3 | 14 | \* |
| 4 | 28 | \* |
| 5 | 04 | \* |
| 6 | 37 | \* |
| 7 | 1D | \* |

1. Split the bits of virtual address and physical address.

Frame size = 32K -> 4 frames in PM (2 bits), 8 frames in VM (3 bits)

Both have 15 bits left as offset

Physical address -> 18 bits

|  |  |
| --- | --- |
| 3 bits | 15 bits |

Virtual address -> 17 bits

|  |  |
| --- | --- |
| 2 bits | 15 bits |

2. Split the bits in memory address based on the cache.

2K DM cache, 256Bytes/block -> 8 blocks/page -> 64 pages in PM

256Bytes/page -> 8 bits offset

8 blocks/page -> 3 bit row

64 pages -> 6 bits page#

Which totals 17 bits, which is the amount needed to map all 128K bytes in PM

|  |  |  |
| --- | --- | --- |
| 6 bits | 3 bits | 8 bits |

3. Suppose the processor has requested to access a memory in 0x32764 (which is virtual address)

0x32764 -> 00|11 0|010 0111 0110 0100

Page# 110 -> 6

1) Is it a page fault? Explain.

Yes, page 6 is not stored in Physical memory.

2) Show the changes of TLB, TLB LRU, page table and Mem LRU

**TLB**

|  |  |  |
| --- | --- | --- |
| Virtual page # | Physical page # | Valid |
| 6 | 1 | 1 |
| 0 | 2 | 1 |

**TLB LRU stack**

virtual page 6 is added to the top, 5 is pushed out

|  |
| --- |
| 6 |
| 0 |

**Page Table**

Physical memory page 1 now stores the contents of virtual page 6

|  |  |  |
| --- | --- | --- |
| Virtual page # | Physical page # | Valid |
| 0 | 2 | 1 |
| 1 | --- | 0 |
| 2 | --- | 0 |
| 3 | --- | 0 |
| 4 | 0 | 1 |
| 5 | 3 | 1 |
| 6 | 1 | 0 |
| 7 | --- | 0 |

**Mem LRU stack**

page 6 is added to the top of LRU, page 1 is removes from memory

Initial final

|  |
| --- |
| 6 |
| 0 |
| 5 |
| 4 |

**cache**

Virtual address -> 110 | 010 0111 0110 0100

Into Physical address -> 01 | 010 0111 0110 0100

Address for cache -> 01 0100 | 111 | 0110 0100

Set: 111 -> 0x7

Tag: 01 0100 -> 0x14

Offset: 0110 0100 -> 0x64

Directly replaces Line#(set) 0x7

|  |  |  |
| --- | --- | --- |
| Line # | Tag | Data block |
| 0 | 10 | \* |
| 1 | 0A | \* |
| 2 | 3C | \* |
| 3 | 14 | \* |
| 4 | 28 | \* |
| 5 | 04 | \* |
| 6 | 37 | \* |
| 7 | 14 | \* |