**vHomework Problem Set #3**

Tolaesh Mengeste

CSS 422 Hardware

**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.

Total memory capacity of 2MB = 221B

cache consists of 64 blocks = 26  from here the block line is 6

each block contains 32 bytes = 25 from here the offset is 5

Tag = 21 – 5 - 6 = 10

|  |  |  |
| --- | --- | --- |
| Tag (10) | Block (6) | Offset (5) |

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

$123A63 in binary 1 0010 0011 1010 0110 0011

              tag:       0x\_\_\_\_247\_\_\_\_\_\_

              line:      0x\_\_\_\_\_13\_\_\_\_\_

              offset:   0x\_\_\_\_\_3\_\_\_\_\_

2. Fully Associative Mapping

  1) Divide the bits into tag and offset bits.

Total memory capacity of 2MB = 221B

each block contains 32 bytes = 25 from here the offset is 5

Tag = 21 – 5 = 16

|  |  |
| --- | --- |
| Tag (16) | Offset (5) |

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

$123A63 in binary 1 0010 0011 1010 0110 0011 then the

              tag:       0x\_\_\_91D3\_\_\_\_\_\_\_

              offset:   0x\_\_\_\_3\_\_\_\_\_\_

3. 4-way set associative mapping

Total memory capacity of 2MB = 221B

cache consists of 64 blocks = 26  from here the block line is 26/22  = 24 so that the set is 4.

each block contains 32 bytes = 25 from here the offset is 5

Tag = 21 – 5 - 4 = 12

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

|  |  |  |
| --- | --- | --- |
| Tag (12) | Set (4) | Offset (5) |

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

              tag:       0x\_\_\_\_\_91D3\_\_\_\_\_

              set:       0x\_\_\_\_3\_\_\_\_\_\_

              offset:   0x\_\_\_3\_\_\_\_\_\_\_

**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 | E0 | E1 | E2 | E3 |
| 0001 | 1 | 14 | 15 | 16 | 17 |
| 1011 | 2 | B8 | B9 | BA | B8 |
| 0110 | 3 | 6C | 6D | 5E | 6F |

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

The total memory capacity of 256 bytes = 28

16-byte direct-mapped cache = 24 in this case the offset is 4.

cache with 4 bytes per block

There is 5 hit memory address that are 93,17, A8, A9, and 4F

  The total memory address is 14 to get the hit ratio

Hit Ratio =HIT/Total memory address

= 5/14 \*100%

= 35.7%. ~ 35.5%

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 0 | offset 1 | offset 2 | offset 3 |
| 0101 | 0 | 50 | 51 | 52 | 53 |
| 1010 | 1 | A4 | A5 | A6 | A7 |
| 1011 | 2 | B8 | B9 | BA | BB |
| 0100 | 3 | 4C | 4D | 4E | 4F |

**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 is 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.

Virtual address

256K virtual address = 218

each page (frame) has 32K bytes (byte addressable) = 215  the offset is 15

From here the page is 18-15 = 3.

|  |  |
| --- | --- |
| Page (3) | Offset (15) |

Physical address

128K physical address space = 217

each page (frame) has 32K bytes (byte addressable) = 215  the offset is 15

From here the page is 17-15 = 2

|  |  |
| --- | --- |
| Page (2) | Offset (15) |

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

 Block size 256 = 28 from here we get 8 bits offset

128K physical address space = 217

2Kbyte direct-mapped cache = 211

The block line = 11 – 8 = 3

Tag = 17 – 3 – 8 = 6 bits.

|  |  |  |
| --- | --- | --- |
| Tag (6) | Block (3) | Offset (8) |

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

1) Is it a page fault? Explain.

Yes. Because the pages are not mapped with frames in the page table therefore the page fault is occurred.

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

 0x32764 change this to binary 11 0010 0111 0110 0100

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

**TLB LRU stack**

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

**Page Table**

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

**Mem LRU stack**

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

3) Show the changes in Cache.

**Cache**

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