Lidya Habteselassie

Homework Problem Set #3

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

   1**.    Direct Mapping**

* 1. Divide the bits in main memory into tag, block and offset bits.
* Main Memory 2M byte = 2^21
* 64 block

Each block has 32 bytes,

* offset = 5

Each page has 64 blocks,

* 2^6 = 64
* Row = 6

Tag = 21 – 6 – 5

Tag = 10

**Answer, tag = 10, lines = 6 and offset = 5**

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

$123A63 = 1 0010 0011 1010 0110 0011

              tag:       0x247

              line:       0x13

              offset:   0x3

**2.   Fully associative mapping**

  1) Divide the bits in main memory into tag and offset bits.

Main Memory 2M byte = 2^21

* + 1. block

Each block has 32 bytes

Tag and offset Each block has 32 bytes. Offset = 5

* Tag = 21 – 5
* Tag = 16

**Answer, tag 16, offset 5**

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

        First, convert the hex to binary

$123A63 = 0001 0010 0011 1010 0110 0011

tag:   0x91d3

              offset:   0x3

**3.    4-way set associative mapping**

 1) Divide the bits in main memory into tag, set and offset bits

Offset = 5

Set = 4

Tag = 21 – 5 – 4 , tag = 12

**Answer: Tag : 12, Set: 4, Offset = 5**

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

0x123A63 = 1 0010 0011 1010 0110 0011

  tag:       0x91d

              set:       0x3

              offset:   0x3

**Q2. (6 pts) Cache hit and miss**

Suppose we have a computer that uses a memory with 256 byte capacity. 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 is already filled out as shown below. (The contents of the tag are shown in binary and the cache “contents” are simply the address stored at that cache location)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 | BB |
| 0110 | 3 | 6C | 6D | 6E | 6F |

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

Memory 2^8 byte = 256 byte

Direct-mapped cache: 2^4, 4 bytes per block (offset is 2)

14 accesses total. hit ration is 5/14 ~ 35.7%

5 hits on address A9, AB, 93, 17 and 4F

**Answer: 35.5%**

**2.    What memory blocks will be in the cache after the last address has been assessed?**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 |

**Q6. (6 pts)  Virtual memory and cache**

Consider a processor with the following memory hierarchy:

256K byte virtual address space

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

2K byte direct-mapped cache, a block (refill line) has 256 byte

The machine uses a two entry TLB.

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

TLB LRU is used whenever an entry of TLB is replaced.

Mem LRU is used when a mapping is replaced when page fault happens.

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**

**Virtual Address**

256K = 2^8x2^10= 2^18= has a total of 18 bits

Size of each page: 32K = 2^15 = 15 bit is offset

|  |  |
| --- | --- |
| Page=3 | Offset= 15 |

**Physical address**

128K = 2^7x2^10= 2^17= has a total of 18 bits

Size of each page: 32K = 2^15 = 15 bit is offset

**17-15=2 bits for page**

|  |  |
| --- | --- |
| Page=2 | Offset= 15 |

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

 Memory size 128k= 2^17

Cache size 2k = 2^11

Block size 256 =2^8

Tag =17-(3+8)=6bits

Block =11-8 bits = 3 bits

Offset = 8 bits

|  |  |  |
| --- | --- | --- |
| Tag=6 | Line =3 | Offset =8 |

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

1) Show the changes of TLB, TLB LRU, page table and Mem LRU. Also state whether page fault happen or not.

 0x32764 = 11 0010 0111 0110 0100 in virtual address.

Fist three bits are the page number, $6

|  |  |  |  |
| --- | --- | --- | --- |
| TL | 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 |

**2) Show the changes in Cache.**

**Cache**

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