**[NOTE: ALL HANDWRITTEN WORK IS AT END OF FILE. FILLED IN ALL ANSWERS IN WORD FOR EASE OF READING AS I HAVE PRETTY TERRIBLE HANDWRITING]**

# Q1. Cache and Memory mapping (6 points)

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

**1.    Direct Mapping**

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

|  |  |  |
| --- | --- | --- |
| TAG | ROW/LINE | OFFSET |
| 10 | 6 | 5 |

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

              Tag:       0x247

              Line:      0x13

              Offset:   0x03

**2.   Fully associative mapping**

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

|  |  |
| --- | --- |
| TAG | OFFSET |
| 16 | 5 |

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

              Tag:       0x91D3

              Offset:   0x03

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

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

|  |  |  |
| --- | --- | --- |
| TAG | SET | OFFSET |
| 12 | 4 | 5 |

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

              Tag:       0x91D

              Set:       0x3

              Offset:   0x03

# Q2. Cache hit and miss (3 points) Slide 60

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 | 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)?**

 14 Total Address. 5 hit, 9 miss. 35.7% Hit Ratio

**2. What memory 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 | 91 | E2 | 93 |
| 1010 | 1 | A4 | 15 | 16 | 17 |
| 1011 | 2 | A8 | B9 | BA | AB |
| 0100 | 3 | 6C | AD | 4E | 4F |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ADDRESS | TAG | BLOCK | OFFSET | HIT/MISS |
| 91 | 1001 | 00 | 01 | MISS |
| A8 | 1010 | 10 | 00 | MISS |
| A9 | 1010 | 10 | 01 | HIT |
| AB | 1010 | 10 | 11 | HIT |
| AD | 1010 | 11 | 01 | MISS |
| 93 | 1001 | 00 | 11 | HIT |
| 6E | 0110 | 11 | 10 | MISS |
| B9 | 1011 | 10 | 01 | MISS |
| 17 | 0001 | 01 | 11 | HIT |
| E2 | 1110 | 00 | 10 | MISS |
| 4E | 0100 | 11 | 10 | MISS |
| 4F | 0100 | 11 | 11 | HIT |
| 50 | 0101 | 00 | 00 | MISS |
| A4 | 1010 | 01 | 00 | MISS |

# 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.
* 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 | **Page #** | **Offset** |
|  | **3** | **15** |
| Physical | **Frame #** | **Offset** |
|  | **2** | **15** |

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

|  |  |  |
| --- | --- | --- |
| TAG | ROW/LINE | OFFSET |
| 6 | 3 | 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, page fault occurs, 6 was not in TLB (miss) so went to page table for lookup. 6 was not mapped so PAGE\_FAULT occurred

**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**

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

**Page Table**

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

**Mem LRU stack**

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

**3) Show the changes in Cache. 0x32764(Virtual) -> 0x0A764**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **0** | **1** | **0** | **0** | **1** | **1** | **1** | **0** | **1** | **1** | **0** | **0** | **1** | **0** | **0** |
| **TAG** | | | | | | **ROW/LINE** | | | **OFFSET** | | | | | | | |
| **0x14** | | | | | | **7** | | | **b** | | | | | | | |

**Cache**

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







