**CS 572: Computer Architecture** **(Due: April 27th, 2021)**

# Submission Homework 2

*Instructor:* Prof. Justin Goins *Name:* Rashmi Jadhav, *Student ID:* 934-069-574

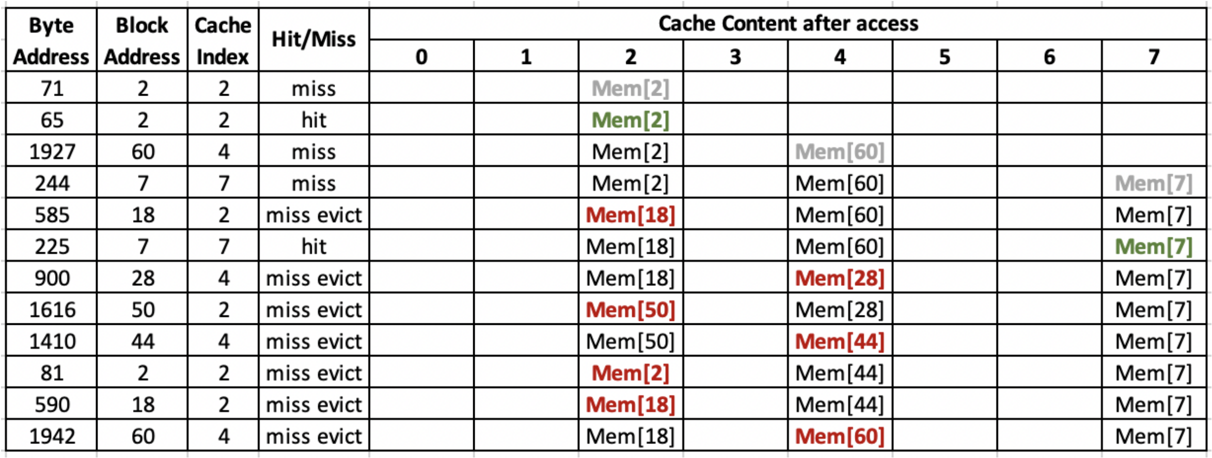
1. Direct-mapped cache utilizing 26-bit address:
```
Offset bits - 0, 1, ..., 4 (total 5)
Index bits - 5, 6, 7 (total 3)
Tag bits - 8, 9, ..., 25 (total 18)
```
   (a) The byte offset has a total of 5 bits and thus it can represent $2^5$ bytes within a block. Thus, the **block size** for the given direct-mapped cache is $2^5$ = **32 bytes**.

   (b) The address has 3 bits for storing indexes which means that it can represent a total of $2^3$ indexes. Thus, the **cache can contain** $2^3 = 8$ **blocks.**

   (c) One entry in the cache would occupy **1 valid bit, 18 tag bits, and 256 data bits**(since block size is 32 bytes = 32 x 8 bits).
   Thus, **one cache entry takes a total of 1 + 18 + 256 = 275 bits.**
   Since the cache can contain 8 such entries, **the total number of bits required by the entire hypothetical cache would be 275 x 8 = 2200 bits.**

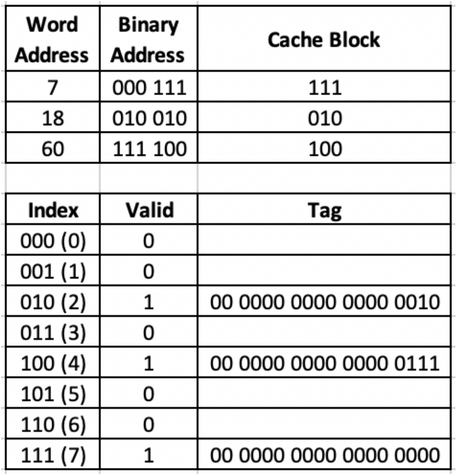
2. Direct-mapped cache memory accesses:

   (a) Following are the cache accesses for the given byte addresses (the convention for depicting hit: green, miss: gray, and evict:red is same as per class):



| Byte Address | Block Address | Cache Index | Hit/Miss | Cache Content after access | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 71 | 2 | 2 | miss | | | Mem[2] | | | | | |
| 65 | 2 | 2 | hit | | | Mem[2] | | | | | |
| 1927 | 60 | 4 | miss | | | Mem[2] | | Mem[60] | | | |
| 244 | 7 | 7 | miss | | | Mem[2] | | Mem[60] | | | Mem[7] |
| 585 | 18 | 2 | miss evict | | | Mem[18] | | Mem[60] | | | Mem[7] |
| 225 | 7 | 7 | hit | | | Mem[18] | | Mem[60] | | | Mem[7] |
| 900 | 28 | 4 | miss evict | | | Mem[18] | | Mem[28] | | | Mem[7] |
| 1616 | 50 | 2 | miss evict | | | Mem[50] | | Mem[28] | | | Mem[7] |
| 1410 | 44 | 4 | miss evict | | | Mem[50] | | Mem[44] | | | Mem[7] |
| 81 | 2 | 2 | miss evict | | | Mem[2] | | Mem[44] | | | Mem[7] |
| 590 | 18 | 2 | miss evict | | | Mem[18] | | Mem[44] | | | Mem[7] |
| 1942 | 60 | 4 | miss evict | | | Mem[18] | | Mem[60] | | | Mem[7] |

   (b) The final state of the cache is as follows:



| Word Address | Binary Address | Cache Block |
|---|---|---|
| 7 | 000 111 | 111 |
| 18 | 010 010 | 010 |
| 60 | 111 100 | 100 |

| Index | Valid | Tag |
|---|---|---|
| 000 (0) | 0 | |
| 001 (1) | 0 | |
| 010 (2) | 1 | 00 0000 0000 0000 0010 |
| 011 (3) | 0 | |
| 100 (4) | 1 | 00 0000 0000 0000 0111 |
| 101 (5) | 0 | |
| 110 (6) | 0 | |
| 111 (7) | 1 | 00 0000 0000 0000 0000 |

3. (a) The byte offset has a total of 5 bits and thus it can represent $2^5$ bytes within a block. Thus, the **block size** for the given 4-way set associative cache is $2^5$ = **32 bytes**.
   (b) The address has 3 bits for storing indexes which means that it can represent a total of $2^3 = 8$ indexes. For a 4-way set associative cache, each index can contain 4 blocks and thus this **cache can contain 8 x 4 = 32 blocks.**
   (c) One entry in the cache would occupy **1 valid bit, 18 tag bits, and 256 data bits**(since block size is 32 bytes = 32 x 8 bits).
   Thus, **one cache entry takes a total of 1 + 18 + 256 = 275 bits.**
   Since the cache can contain 32 such entries, **the total number of bits required by the entire hypothetical cache would be 275 x 32 = 8800 bits.**

4. 4-way set associative cache memory accesses:
   (a) Following are the cache accesses for the given byte addresses (the convention for depicting hit: green, miss: gray, and evict:red is same as per class):

| Byte Address | Block Addre | Cache Index | Hit/Miss | Cache Content after access | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Set 0 | | | | Set 1 | | | | Set 2 | | | | Set 3 | | | |
| 71 | 2 | 2 | Miss | | | | | | | | | Mem [2] | | | | | | | |
| 65 | 2 | 2 | Hit | | | | | | | | | Mem [2] | | | | | | | |
| 1927 | 60 | 0 | Miss | Mem [60] | | | | | | | | Mem [2] | | | | | | | |
| 244 | 7 | 3 | Miss | Mem [60] | | | | | | | | Mem [2] | | | | Mem [7] | | | |
| 585 | 18 | 2 | Miss | Mem [60] | | | | | | | | Mem [2] | Mem [18] | | | Mem [7] | | | |
| 225 | 7 | 3 | Hit | Mem [60] | | | | | | | | Mem [2] | Mem [18] | | | Mem [7] | | | |
| 900 | 28 | 0 | Miss | Mem [60] | Mem [28] | | | | | | | Mem [2] | Mem [18] | | | Mem [7] | | | |
| 1616 | 50 | 2 | Miss | Mem [60] | Mem [28] | | | | | | | Mem [2] | Mem [18] | Mem [50] | | Mem [7] | | | |
| 1410 | 44 | 0 | Miss | Mem [60] | Mem [28] | Mem [44] | | | | | | Mem [2] | Mem [18] | Mem [50] | | Mem [7] | | | |
| 81 | 2 | 2 | Hit | Mem [60] | Mem [28] | Mem [44] | | | | | | Mem [2] | Mem [18] | Mem [50] | | Mem [7] | | | |
| 590 | 18 | 2 | Hit | Mem [60] | Mem [28] | Mem [44] | | | | | | Mem [2] | Mem [18] | Mem [50] | | Mem [7] | | | |
| 1942 | 60 | 0 | Hit | Mem [60] | Mem [28] | Mem [44] | | | | | | Mem [2] | Mem [18] | Mem [50] | | Mem [7] | | | |

   (b) The final state of the cache is as follows:

| Word Address | Binary Address | Cache Block |
|---|---|---|
| 2 | 000 010 | 010 |
| 7 | 000 111 | 111 |
| 18 | 010 010 | 010 |
| 28 | 011 100 | 100 |
| 44 | 101 100 | 100 |
| 50 | 110 010 | 010 |
| 60 | 111 100 | 100 |

| Index | Valid 0 | Tag 0 | Valid 1 | Tag 1 | Valid 2 | Tag 2 | Valid 3 | Tag 3 |
|---|---|---|---|---|---|---|---|---|
| 000 (0) | 0 | | 0 | | 0 | | 0 | |
| 001 (1) | 0 | | 0 | | 0 | | 0 | |
| 010 (2) | 1 | 00 0000 0000 0000 0000 | 1 | 00 0000 0000 0000 0010 | 1 | 00 0000 0000 0000 0110 | 0 | |
| 011 (3) | 0 | | 0 | | 0 | | 0 | |
| 100 (4) | 1 | 00 0000 0000 0000 0111 | 1 | 00 0000 0000 0000 0011 | 1 | 00 0000 0000 0000 0101 | 0 | |
| 101 (5) | 0 | | 0 | | 0 | | 0 | |
| 110 (6) | 0 | | 0 | | 0 | | 0 | |
| 111 (7) | 1 | 00 0000 0000 0000 0000 | 0 | | 0 | | 0 | |

5. Given:
   `average clocks per instruction (CPI) = 1.2;`
   `clock speed = 3GHz;`
   `main memory access time = 120 cycles`
   `miss rate per instruction at the primary cache = 3.5%.`

   `CPU time per instruction = CPI / clock speed = 1.2 / 3 = 0.4 ns.`
   `Average Memory Access Time (AMAT) = Hit Time + Miss Rate x Miss Penalty`
   $\Longrightarrow AMAT_1$ `= (0.4) x 96.5% + (0.4 + 120 / 3) x 3.5% = 1.8 ns`

   `Upon adding secondary cache,`
   $AMAT_2$ `= (0.4) x 96.5% + 3.5% ((0.4 + 5) x 99.2% + (0.4 + 5 + 120 / 3) x 0.8%)`
   $\Longrightarrow AMAT_2$ `= 0.5862 ns`

   `Old CPI / New CPI = 1.8 / 0.5862 = 3.07062`

   The system thus operates approximately **3x faster** when we add a secondary cache with given parameters.