****

**Computer Organization and Assembly Language**

**Dr. Mohamed Shalan**

**Project 2 Report**

**Cache Simulator**

**Name: Ahmed Mohamed Ahmed**

**ID: 900141806**

**Name: Hossam El Samanody**

**ID: 900143231**

**12/14/2016**

**Objectives:**

A cache simulator, using the C/C++ language, to simulate a set-associative cache.

The simulator does 3 experiments:

1. Direct Mapped Cache (one way): varying line size and cache size.
2. Fully Associative Cache (n-way): varying cache size and replacement policy.
3. Set Associative Cache (n > number of ways > 1): varying number of ways.

**Tools:**

C++ programming language & IDE. also: iostream, vector, and iomanip libraries.

**Inputs:**

No inputs.

**Outputs:**

Miss ratios for all the experiments’ cases.

**Procedures:**

1. Reading the requirements and rubrics.
2. Studying Cache logic.
3. Trying the given skeleton and understanding it.
4. Adding the needed functions.
5. Debugging the code.
6. Trying a 12-address test case.
7. Making a conclusion in the report

**Code’s Description:**

Language: C++, using Microsoft Visual Studio.

The code is modular by using functions and commented.

Functions: main, 6 memGen functions, 6 show\_miss\_ratio functions, and 25 functions for all included cases.

Global Data: Sizes, random generator parameters, and number of accesses.

**Main:**

Calls the 6 show\_miss\_ratio functions.

**Show\_miss\_ratio functions:**

Each function generates the needed data structures for the used caches, and counters for the percentage of hits. Then loops by number of accesses, each iteration: generates the needed address by the related memGen function, calls all 25 functions for experiments’ cases, calculates hit ratios, and displays them.

**MemGen functions:**

Each one of them generates addresses in a different way.

memGen1: sequential 0,1,…, (real DRAM size = 64 MB)-1, 0, 1, …

memGen2: random in range 0 to (pseudo DRAM size = 128 KB)-1.

memGen3: random in range 0 to (real DRAM size = 64 MB)-1.

memGen4: sequential 0,1,…, (pseudo DRAM size = 1 KB)-1, 0, 1, …

memGen5: sequential 0,1,…, (pseudo DRAM size = 64 KB)-1, 0, 1, …

memGen6: sequential 256,512,…, (real DRAM size = 64 MB)-256, 256, 512, …

**Experiments’ cases functions:**

Takes address and related cache data structure. Calculates index (if needed) and tag. Using the index and tag: it checks for compulsory miss (if so, a miss and adds the address), then checks for a hit, and last, checks for a conflict miss. All replacements are done after the cache is full for full and set associative. The replacement way depends on the experiment’s case.

**General conclusion:**

As approximations:

For memGen: 1, 4, 5: if sequential accesses: 0, 1, 2, …, pseudo RAM size-1, 0, 1,

{If (Cache size < pseudo RAM size)

Miss ratio = 1 / line size;

Else miss ratio = (pseudo RAM size / line size) / total number of accesses.}

For memGen: 2, 3: if random: Miss ratio = 1- cache size / pseudo RAM size;

For memGen 6: since line size < 512 B (least possible line size to get a miss followed by a hit) and sequential in steps of 256, therefore: miss percentage = 100%.

**Direct mapping conclusion:**

Data:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| mem1 | mem2 | mem3 | mem4 | mem5 | mem6 |
| 0.25 | 0.507661 | 0.999034 | 0.000256 | 0.016384 | 1 |
| 0.125 | 0.50353 | 0.999017 | 0.000128 | 0.008192 | 1 |
| 0.0625 | 0.501895 | 0.999046 | 6.40E-05 | 0.004096 | 1 |
| 0.03125 | 0.500991 | 0.998998 | 3.20E-05 | 0.002048 | 1 |
| 0.015625 | 0.500397 | 0.998986 | 1.60E-05 | 0.001024 | 1 |
| 0.007813 | 0.499906 | 0.999012 | 8.00E-06 | 0.000512 | 1 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| mem1 | mem2 | mem3 | mem4 | mem5 | mem6 |
| 0.0625 | 0.992021 | 0.999988 | 6.40E-05 | 0.0625 | 1 |
| 0.0625 | 0.984218 | 0.999973 | 6.40E-05 | 0.0625 | 1 |
| 0.0625 | 0.968532 | 0.999945 | 6.40E-05 | 0.0625 | 1 |
| 0.0625 | 0.937495 | 0.999878 | 6.40E-05 | 0.0625 | 1 |
| 0.0625 | 0.875427 | 0.999762 | 6.40E-05 | 0.0625 | 1 |
| 0.0625 | 0.750348 | 0.99952 | 6.40E-05 | 0.0625 | 1 |
| 0.0625 | 0.501895 | 0.999046 | 6.40E-05 | 0.004096 | 1 |

**memGen1**: a) because addresses are sequential till RAM size, the first of a 32-byte group will make a miss, while the other 31 will make hits, therefore for 1 M iterations: miss ratio = 1/line size. Therefore, negative slope.

b) because addresses are sequential till RAM size, the first of a 32-byte group will make a miss, while the other 31 will make hits, therefore for 1 M iterations: miss ratio = 1/line size = 1/16 = 0.0625. Therefore, constant slope.

**memGen2**: a) because addresses are random in range (0 to 128KB-1), therefore, changing the number of bytes per line is not effective because of random addresses and only changing the size will change the number of RAM lines mapped to the same cache line and will have an effect. Therefore, miss ratio = 1-hit ratio = 1- cache size / RAM size = 1- 64 KB / 128 KB = 0.5. Constant slope.

b) Therefore, miss ratio = 1-hit ratio = 1- cache size / RAM size = 1- cache size / 128 KB. Negative slope.

**memGen3**: a) because addresses are random in range (0 to 64MB), therefore, changing the size will change the number of RAM lines mapped to the same cache line, and the number of bytes per line is not effective because of random addresses.

Therefore, miss ratio = 1-hit ratio = 1- cache size / RAM size = 1- 64 KB/ 64MB = 0.999023. Constant slope.

b) Therefore, miss ratio = 1-hit ratio = 1- cache size / RAM size = 1- cache size / 64 MB. Nearly constant negative slope.

**memGen4**: a) addresses are sequential till RAM size (0 to 1024 B), therefore, RAM size <= Cache size. Therefore, all misses are compulsory. Therefore, miss ratio = (RAM size / cache line size) / iterations = (1 KB / cache line size) / 1000000. Nearly constant negative slope.

b) Therefore, miss ratio = (RAM size / cache line size) / iterations = (1 KB / 16 B) / 1000000 = 6.4e-005. Constant slope.

**memGen5**: a) because addresses are sequential till RAM size (0 to 64 KB), therefore, RAM size <= Cache size. Therefore, all misses are compulsory. Therefore, miss ratio = (RAM size / cache line size) / iterations = (64 KB / cache line size) / 1000000. Nearly constant negative slope.

b) This combines memGen1 and memGen4 cases: all except last input: RAM size > Cache size. Therefore, miss ratio = 1/line size = 0.0625. Last input: RAM size <= Cache size. Therefore, miss ratio = (RAM size / cache line size) / iterations = (64 KB / 16 B) / 1000000 = 0.004096. Changing constant slope.

**memGen6**: since line size < 512 (least possible line size to get a miss followed by a hit), and replacement rotates. Therefore: miss percentage = 100%. Constant slope.

**Full associative conclusion:**

**Data:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| mem1 | mem2 | mem3 | mem4 | mem5 | mem6 |
| 0.03125 | 0.992187 | 0.999983 | 3.20E-05 | 0.03125 | 1 |
| 0.03125 | 0.984455 | 0.999967 | 3.20E-05 | 0.03125 | 1 |
| 0.03125 | 0.968809 | 0.999939 | 3.20E-05 | 0.03125 | 1 |
| 0.03125 | 0.93766 | 0.999866 | 3.20E-05 | 0.031243 | 1 |
| 0.03125 | 0.874896 | 0.999762 | 3.20E-05 | 0.030668 | 1 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| mem1 | mem2 | mem3 | mem4 | mem5 | mem6 |  |
| 0.03125 | 0.968776 | 0.999943 | 3.20E-05 | 0.03125 | 1 | FIFO |
| 0.03125 | 0.968789 | 0.999943 | 3.20E-05 | 0.03125 | 1 | LRU |
| 0.03125 | 0.968739 | 0.999944 | 3.20E-05 | 0.029345 | 0.999619 | LFU |
| 0.03125 | 0.968809 | 0.999939 | 3.20E-05 | 0.03125 | 1 | Random |

**memGen1**: because addresses are sequential till RAM size, the first of a 32-byte group will make a miss, while the other 31 will make hits, therefore for 1 M iterations: miss ratio = 1/line size = 1/32 = 0.03125. Constant slope.

**memGen2**: a) because addresses are random in range (0 to 128KB-1), therefore, changing the size will change the number of map lines mapped to the same cache line, and the number of bytes per line is not effective because of random addresses.

Therefore, miss ratio = 1-hit ratio = 1- cache size / RAM size = 1- cache size / 128 KB. Negative slope.

b) because addresses are random in range (0 to 128KB-1), therefore, changing the way of replacement and the number of bytes per line will not be effective because of random addresses. Therefore, miss ratio = 1-hit ratio = 1- cache size / RAM size = 1- 4KB / 128 KB approximately = 0.96875. Constant slope.

**memGen3**: a) because addresses are random in range (0 to 64MB), therefore, changing the size will change the number of RAM lines mapped to the same cache line, and the number of bytes per line is not effective because of random addresses.

Therefore, miss ratio = 1-hit ratio = 1- cache size / RAM size = 1- cache size / 64MB. Nearly constant negative slope.

b) because addresses are random in range (0 to 64 MB -1), therefore, changing the way of replacement and the number of bytes per line will not be effective because of random addresses. Therefore, miss ratio = 1-hit ratio = 1- cache size / RAM size = 1- 4KB / 64 MB approximately = 0.99994. Constant slope.

**memGen4**: because addresses are sequential till RAM size (0 to 1024 B), therefore, RAM size <= Cache size. Therefore, all misses are compulsory. Therefore, miss ratio = (RAM size / cache line size) / iterations = (1 KB / 32 B) / 1000000 = 3.2e-005. Constant slope.

**memGen5**: because addresses are sequential till RAM size (0 to 64 KB), therefore, the first of a 32-byte group will make a miss, while the other 31 will make hits. Therefore, miss ratio = 1 / line size = 1/32 = 0.03125. Constant slope.

**memGen6**: a) random replacement: percentage for a hit = (1/262144) \* (1/1000000). therefore: miss percentage = 100%. Constant slope.

b) Only LFU != 100%. Since the first 128 (Cache size 4 KB / Line Size 32 bytes) addresses are placed in the cache when it’s empty and then each following address is a conflict miss that is placed in the first location in the cache, the first element will always be a miss, while the 127 other addresses that were stored in the first circulation will make hits. Since no. of iterations = 1000000 and 64 MB / 256 = 262144, therefore: no. of times to hit each of the stored 127 locations = 3. Therefore: no. of hits = 3\*127 = 381. Therefore: miss ratio for LFU = 1 - 381/1000000 = 0.999619. Somewhat constant slope.

**Set associative conclusion:**

**Data:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| mem1 | mem2 | mem3 | mem4 | mem5 | mem6 |
| 0.03125 | 0.50066 | 0.998993 | 3.20E-05 | 0.002048 | 1 |
| 0.03125 | 0.500569 | 0.998989 | 3.20E-05 | 0.002048 | 1 |
| 0.03125 | 0.501335 | 0.999027 | 3.20E-05 | 0.002048 | 1 |
| 0.03125 | 0.499913 | 0.999005 | 3.20E-05 | 0.002048 | 1 |

**memGen1**: because addresses are sequential till RAM size, the first of a 32-byte group will make a miss, while the other 31 will make hits, therefore for 1 M iterations: miss ratio = 1/line size = 1/32 = 0.03125. Number of ways and the random policy have no effect because each miss will be followed by 31 hits. Constant slope.

**memGen2**: because addresses are random in range (0 to 128KB-1), therefore, changing the number of bytes per line and number of ways are not effective because of random addresses and only changing the size will change the number of RAM lines mapped to the same cache set and will have an effect. Therefore, miss ratio = 1-hit ratio = 1- cache size / RAM size = 1- 64 KB / 128 KB = 0.5. Constant slope.

**memGen3**: because addresses are random in range (0 to 64 MB -1), therefore, changing the way of replacement and the number of bytes per line will not be effective because of random addresses. Therefore, miss ratio = 1-hit ratio = 1- cache size / RAM size = 1- 64KB / 64 MB approximately = 0.999023. Constant slope.

**memGen4**: because addresses are sequential till RAM size (0 to 1024 B), therefore, RAM size <= Cache size. Therefore, all misses are compulsory. Therefore, miss ratio = (RAM size / cache line size) / iterations = (1 KB / 32 B) / 1000000 = 3.2e-005. Constant slope.

**memGen5**: because addresses are sequential till RAM size (0 to 64 KB), therefore, RAM size <= Cache size. Therefore, all misses are compulsory. Therefore, miss ratio = (RAM size / cache line size) / iterations = (64 KB / 32 B) / 1000000 = 0.002048. Constant slope.

**memGen6:** For set associative, it combines the percentages for direct mapped and full associative caches. Therefore: miss percentage = 100%. Constant slope.

**Challenges faced:**

1. Test cases: a 50-page 12-address analysis for each memGen function was done by hand and will be included in the online submission.
2. Making conclusions.
3. Constructing formulas.