Max Shi  
CS 383 C  
Professor Barbalace  
November 18, 2019  
I pledge my honor that I have abided by the Stevens Honor System

Homework Assignment 4

5.1.1.  
64 bit integers take 8 bytes to store, so two 64 bit integers can be stored in a 16 byte cache block.

5.1.2.  
The values of I and J are used many times, so they have temporal locality, as well as B[I][0] because it is in the for loop that changes J but does not change I, so this is the same statement 8000 times in a row.

5.1.3.  
Only A[I][J] has spatial locality, as these are stored sequentially in memory. A[J][I] as J increases is not next to each other in memory.

5.2.1.

16 one word blocks –  
index bits => 16 = 2n -> n = 4  
tag bits => 1 word means 2m = 1, m = 0, thus 64 – (4+0) = 60 bits, offset is 0 bits.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Address | Binary word address | Index | Tag | Hit/Miss |
| 0x03 | 00000011 | 0011 | 0000 | Miss |
| 0xb4 | 10110100 | 0100 | 1011 | Miss |
| 0x2b | 00101011 | 1011 | 0010 | Miss |
| 0x02 | 00000010 | 0010 | 0000 | Miss |
| 0xbf | 10111111 | 1111 | 1011 | Miss |
| 0x58 | 01011000 | 1000 | 0101 | Miss |
| 0xbe | 10111110 | 1110 | 1011 | Miss |
| 0x0e | 00001110 | 1110 | 0000 | Miss |
| 0xb5 | 10110101 | 0101 | 1011 | Miss |
| 0x2c | 00101100 | 1100 | 0010 | Miss |
| 0xba | 10111010 | 1010 | 1011 | Miss |
| 0xfd | 11111101 | 1101 | 1111 | Miss |

5.2.2.  
two word blocks with 8 blocks total --  
index bits => 8 = 2n -> n = 3  
tag bits => 2 words means m = 1, thus 64 – (3+1) = 60 bits, meaning offset is 1 bit.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Address | Binary word address | Index | Tag | Offset | Hit/Miss |
| 0x03 | 00000011 | 001 | 0000 | 1 | Miss |
| 0xb4 | 10110100 | 010 | 1011 | 0 | Miss |
| 0x2b | 00101011 | 101 | 0010 | 1 | Miss |
| 0x02 | 00000010 | 001 | 0000 | 0 | Hit |
| 0xbf | 10111111 | 111 | 1011 | 1 | Miss |
| 0x58 | 01011000 | 100 | 0101 | 0 | Miss |
| 0xbe | 10111110 | 111 | 1011 | 0 | Hit |
| 0x0e | 00001110 | 111 | 0000 | 0 | Miss |
| 0xb5 | 10110101 | 010 | 1011 | 1 | Hit |
| 0x2c | 00101100 | 110 | 0010 | 0 | Miss |
| 0xba | 10111010 | 101 | 1011 | 0 | Miss |
| 0xfd | 11111101 | 110 | 1111 | 1 | Miss |

5.2.3.

For C1, index bits are 2-0 as there are 8 blocks

For C2, index bits are 2-1 as there are 4 blocks

For C3, index bits = are 2-2 as there is one block

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Binary word address | Tag | Index C1 | H/M | Index C2 | H/M | Index C3 | H/M |
| 0x03 | 00000011 | 0000 | 011 | Miss | 01 | Miss | 0 | Miss |
| 0xb4 | 10110100 | 1011 | 100 | Miss | 10 | Miss | 1 | Miss |
| 0x2b | 00101011 | 0010 | 011 | Miss | 01 | Miss | 0 | Miss |
| 0x02 | 00000010 | 0000 | 010 | Miss | 01 | Miss | 0 | Miss |
| 0xbf | 10111111 | 1011 | 111 | Miss | 11 | Miss | 1 | Miss |
| 0x58 | 01011000 | 0101 | 000 | Miss | 00 | Miss | 0 | Miss |
| 0xbe | 10111110 | 1011 | 110 | Miss | 11 | Hit | 1 | Hit |
| 0x0e | 00001110 | 0000 | 110 | Miss | 11 | Miss | 1 | Miss |
| 0xb5 | 10110101 | 1011 | 101 | Miss | 10 | Hit | 1 | Miss |
| 0x2c | 00101100 | 0010 | 110 | Miss | 10 | Miss | 1 | Miss |
| 0xba | 10111010 | 1011 | 010 | Miss | 01 | Miss | 0 | Miss |
| 0xfd | 11111101 | 1111 | 101 | Miss | 10 | Miss | 1 | Miss |

We then choose the one with the lowest miss rate. Cache 1 has 100%, Cache 2 has 10/12 = 83%, Cache 3 has 11/12 = 92%. Thus, we choose cache 2.

5.5.1.

The cache block size is 2m words, where tag size = 64 – (index bits + m + offset).  
Solving for m where index bits = 5 and offset bits = 5 and tag size = 54, 54 = 64 – (5 + m).  
54 = 64 – 5 – m gets m = 5. Because a word is 2 bytes, we have 3 bits left. Therefore, there are 3 bits for the block offset. Thus, the cache block size is 8 words.

5.5.2.  
The cache has 2n blocks, so it has 25 blocks, which is 32.

5.5.3.  
The cache stores 32 blocks of 8 words, so the size in bits is 32 blocks \* 8 words/block \* 32 bits/word = 8192 bits. Then we add on the tag size and index bits for each block, which is (54) \* 32. Then we add on a valid bit for each block. So the total is 8192 + 1728 + 32 = 9952 bits. The data storage is without the extra data, so just the blocks themselves. This quantity is 8192 bits. 9952/8192 is 1.214, which is the ratio.

5.5.4.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Hex Addr | Binary | Index (9-5) | Tag (63-10) | Offset (4-0) | Hit/Miss | Replaced |
| 00 | 00000000 | 0x0 | 0x0 | 0x0 | Miss |  |
| 04 | 00000100 | 0x0 | 0x0 | 0x4 | Hit |  |
| 10 | 00010000 | 0x0 | 0x0 | 0x10 | Hit |  |
| 84 | 10000100 | 0x4 | 0x0 | 0x4 | Miss |  |
| E8 | 11101000 | 0x7 | 0x0 | 0x8 | Miss |  |
| A0 | 10100000 | 0x5 | 0x0 | 0x0 | Miss |  |
| 400 | 010000000000 | 0x0 | 0x1 | 0x0 | Miss | 0x00-0x1f |
| 1E | 00011110 | 0x0 | 0x0 | 0x1E | Miss | 0x00-0x1f |
| 8C | 10001100 | 0x4 | 0x0 | 0xC | Miss |  |
| C1C | 110000011100 | 0x0 | 0x3 | 0x1C | Miss | 0x00-0x1f |
| B4 | 10110010 | 0x5 | 0x0 | 0x12 | Hit |  |
| 884 | 100010000100 | 0x4 | 0x2 | 0x4 | Miss | 0x80-0x9f |

5.5.5.  
3/12 = 0.25 hit rate.

5.5.6.  
<0,3, Mem[0x00]-Mem[0x1f]>

<4,2, Mem[0x80]-Mem[0x9f]>

<5,0, Mem[0xa0]-Mem[0xbf]>

<7,0, Mem[0xc0]-Mem[0xdf]>

5.9.1.  
(0.04) \* 20\*8 = 6.4  
(0.03) \* 20\*16 = 9.6  
(0.02) \* 20\*32 = 12.8  
(0.015) \* 20\*64 = 19.2  
(0.01) \* 20\*128 = 25.6  
8 byte cache block is optimal

5.9.2.  
(0.04) \* (20+8) = 1.12  
(0.03) \* (20+16) = 1.08  
(0.02) \* (20+32) = 1.04  
(0.015) \* (20+64) = 1.26  
(0.01) \* (20+128) = 1.48  
32 byte cache block is optimal.

5.9.3.  
For consistent miss latency, we just choose the lowest miss rate, which is 128 byte cache blocks.

5.11.1.

Address:

|  |  |  |
| --- | --- | --- |
| Tag (63-4) | Index(3-1) | Offset (0) |

Tag is 60 bits, index is 3 bits.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| C1 | Valid | Tag | Index | Data | C2 | V | Tag | Index | Data | C3 | V | Tag | Index | Data |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

5.11.2.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Binary Address | Tag | Index | Offset | H/M | C1 | C2 | C3 |
| 0x03 | 00000011 | 0x0 | 0x1 | 1 | M | 1,0 |  |  |
| 0xb4 | 10110100 | 0xb | 0x2 | 0 | M | 1,0  2,b |  |  |
| 0x2b | 00101011 | 0x2 | 0x5 | 1 | M | 1,0  2,b  5,2 |  |  |
| 0x02 | 00000010 | 0x0 | 0x1 | 0 | H | 1,0  2,b  5,2 |  |  |
| 0xbe | 10111110 | 0xb | 0x7 | 0 | M | 1,0  2,b  5,2  7,b |  |  |
| 0x58 | 01011000 | 0x5 | 0x4 | 0 | M | 1,0  2,b  5,2  4,5  7,b |  |  |
| 0xbf | 10111111 | 0xb | 0x7 | 1 | H | 1,0  2,b  5,2  4,5  7,b |  |  |
| 0x0e | 00001110 | 0x0 | 0x7 | 0 | M | 1,0  2,b  5,2  4,5  7,b | 7,0 |  |
| 0x1f | 00011111 | 0x1 | 0x7 | 1 | M | 1,0  2,b  5,2  4,5  7,b | 7,0 | 7,1 |
| 0xb5 | 10110101 | 0xb | 0x2 | 1 | H | 1,0  2,b  5,2  4,5  7,b | 7,0 | 7,1 |
| 0xbf | 10111111 | 0xb | 0x7 | 1 | H | 1,0  2,b  5,2  4,5  7,b | 7,0 | 7,1 |
| 0xba | 10111010 | 0xb | 0x5 | 0 | M | 1,0  2,b  5,2  4,5  7,b | 5,b  7,0 | 7,1 |
| 0x2e | 00101110 | 0x2 | 0x7 | 0 | M | 1,0  2,b  5,2  4,5  7,b | 5,b  7,2 | 7,1 |
| 0xce | 11001110 | 0xc | 0x7 | 0 | M | 1,0  2,b  5,2  4,5  7,b | 5,b  7,2 | 7,c |

Cache mapping is index,tag notation.

5.11.3.

Fully associative means that there is only one block, thus 0 bits for index.

1 word/block = 20 which means offset is 0.

Address: Tag is all 64 bits of the address.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Valid | Tag | Data | Valid | Tag | Data | Valid | Tag | Data | Valid | Tag | Data | Valid | Tag | Data | Valid | Tag | Data |

+ 2 more repetitions of valid,tag,data.

5.13.1.  
MTBF = MTTF + MTTR = 365\*3 + 1 = 1096 days.

5.13.2  
Availability = MTTF / MTBF = 1095/1096 = 0.99908

5.13.3  
Availability will approach 1 as MTTR approaches 0. However, this is unrealistic, as nothing can really be replaced instantly upon failure.

5.13.4.  
Availability will approach 0 as MTTR gets very high. If MTTR is high enough, availability should be almost zero, signifying that the device is not available a lot of the time, as it is bring replaced. Thus, high MTTR does imply low availability.

5.17.1.  
To map to each page address, the size of 8 KiB requires 13 bits to encode. Thus, the tag size can be 32 -13 = 19 bits. Then, there can be five page tables for each of the five processes, giving 2^19 \* 4 \* 5 bytes = 10485760 bytes.

5.17.2.

If we have 256 segments, using half the memory would mean we are using 128 segments out of the 256. On the second level, each page takes 8 KiB and requires 5 for the 5 processes. Therefore, on the second level, 8KiB\*128 segments\*5 processes = 5 MiB. For the first level cache, this is the same setup, except we use 6KiB per page. Therefore, 6KiB \* 128 segments \* 5 processes = 3840 KiB.

Maximum usage occurs at full usage of all 256 segments. Multiplying both of these by 2 gets 7680 KiB and 10 MiB for the first and second levels respectively.

5.17.3.

Two 64 bit words per block would result in 16 byte blocks. Thus, 16 KiB / 16 byte blocks = 1024 blocks. Therefore, this requires 10 index bits for direct associativity. 2 words per block requires one offset bit, and 8 bytes per word requires 3 byte bits, totaling 4 offset bits. However, from the earlier part of this problem, we saw that we need 13 bits to encode addresses for the virtual memory. Therefore, the offset and the index bits cannot fit in the page index. We can increase associativity to reduce the amount of indexes there are in the memory while maintaining the amount of data we can hold, or even increasing it.