5.2.1

Index length = 4 bit

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Address | Binary | Tag | Index | Hit / Miss |
| 3 | 00000011 | 0000 | 0011 | Miss |
| 180 | 10110100 | 1011 | 0100 | Miss |
| 43 | 00101011 | 0010 | 1011 | Miss |
| 2 | 00000010 | 0000 | 0010 | Miss |
| 191 | 10111111 | 1011 | 1111 | Miss |
| 88 | 01011000 | 0101 | 1000 | Miss |
| 190 | 10111110 | 1011 | 1110 | Miss |
| 14 | 00001110 | 0000 | 1110 | Miss |
| 181 | 10110101 | 1011 | 0101 | Miss |
| 44 | 00101100 | 0010 | 1100 | Miss |
| 186 | 10111010 | 1011 | 1010 | Miss |
| 253 | 11111101 | 1111 | 1101 | Miss |

5.2.2

Index length = 3 bit

Offset length = 1 bit

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Address | Binary | Tag | Index | Offset | Hit / Miss |
| 3 | 00000011 | 0000 | 001 | 1 | Miss |
| 180 | 10110100 | 1011 | 010 | 0 | Miss |
| 43 | 00101011 | 0010 | 101 | 1 | Miss |
| 2 | 00000010 | 0000 | 001 | 0 | Hit |
| 191 | 10111111 | 1011 | 111 | 1 | Miss |
| 88 | 01011000 | 0101 | 100 | 0 | Miss |
| 190 | 10111110 | 1011 | 111 | 0 | Hit |
| 14 | 00001110 | 0000 | 111 | 0 | Miss |
| 181 | 10110101 | 1011 | 010 | 1 | Hit |
| 44 | 00101100 | 0010 | 110 | 0 | Miss |
| 186 | 10111010 | 1011 | 101 | 0 | Miss |
| 253 | 11111101 | 1111 | 110 | 1 | Miss |

5.2.3

8 words of data =

Miss rate: 3% for current block size

C1 miss rate = 11/12

C2 miss rate = 9/12

C3 miss rate = 10/12

**In terms of miss rate, C2 design is the best among three.**

C1 stall time: 2 \* 12 + 25 \* 11 = 299

C2 stall time: 2 \* 12 + 25 \* 9 = 261

C3 stall time: 2 \* 12 + 25 \* 10 = 310

**C2 is the best design since it has smallest stall time**

5.2.4

Total bits = (2^index bits) \* (valid bits + tag bits + (data bits \* 2^offset bits))

**First cache:**

32 KiB =

Index bit: 15

Offset bit: 1

Address bit: 32

Tag bit: 14

Valid bit: 1

**Total bits = 2^ 15 \* (1 + 14 + (32 \* 2 ^ 1)) = 2588672 bits**

**Second cache (16-word blocks):**

Index bit: 13

Offset bit: 4

Address bit: 32

Tag bit: 13

Valid bit: 1

**Total bits = 2^ 13 \* (1 + 13 + (32 \* 2 ^ 4)) = 4308992 bits**

**It might have a slower performance because the number of bits for 16-word cache is more than the first cache**

5.2.5

Data size = 32 KB = KB

Size of cache block = 2 words

Number of blocks = 16KB

Number of Cache lines = 14

Number of sets in 2-way set associative memory = 8KB

Increase the block size

There are more data stored in the cache, thus it will have a less miss rate

Advantage:

Less miss rate and mapping will become easier

Disadvantage:

Take a longer time to access, and larger miss penalty

5.2.6

XOR is possible. If we use direct mapping cache, the tag bits will still point to the correct location.

Using mapping function: set – (block address % set), values will be stored in the blocks in a reverse way.

5.7.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Address | Binary | Tag | Index | offset | Hit / Miss |
| 3 | 00000011 | 00000 | 01 | 1 | Miss |
| 180 | 10110100 | 10110 | 10 | 0 | Miss |
| 43 | 00101011 | 00101 | 01 | 1 | Miss |
| 2 | 00000010 | 00000 | 01 | 0 | Hit |
| 191 | 10111111 | 10111 | 11 | 1 | Miss |
| 88 | 01011000 | 01011 | 00 | 0 | Miss |
| 190 | 10111110 | 10111 | 11 | 0 | Hit |
| 14 | 00001110 | 00001 | 11 | 0 | Miss |
| 181 | 10110101 | 10110 | 10 | 1 | Hit |
| 44 | 00101100 | 00101 | 10 | 0 | Miss |
| 186 | 10111010 | 10111 | 01 | 0 | Miss |
| 253 | 11111101 | 11111 | 10 | 1 | Miss |

5.7.2

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address of memory block accessed | Binary address | Hit/  Miss | Contents of cache blocks after reference | | | | | | | |
| B0 | B1 | B2 | B3 | B4 | B5 | B6 | B7 |
| 3 | 0000 0011 | Miss | **M[3]** |  |  |  |  |  |  |  |
| 180 | 1011 0100 | Miss | M[3] | **M[180]** |  |  |  |  |  |  |
| 43 | 0010 1011 | Miss | M[3] | M[180] | **M[83]** |  |  |  |  |  |
| 2 | 0000 0010 | Miss | M[3] | M[180] | M[83] | **M[2]** |  |  |  |  |
| 191 | 1011 1111 | Miss | M[3] | M[180] | M[83] | M[2] | **M[191]** |  |  |  |
| 88 | 0101 1000 | Miss | M[3] | M[180] | M[83] | M[2] | M[191] | **M[88]** |  |  |
| 190 | 1011 1110 | Miss | M[3] | M[180] | M[83] | M[2] | M[191] | M[88] | **M[190]** |  |
| 14 | 0000 1110 | Miss | M[3] | M[180] | M[83] | M[2] | M[191] | M[88] | M[190] | **M[14]** |
| 181 | 1011 0101 | Miss | **M[181]** | M[180] | M[83] | M[2] | M[191] | M[88] | M[190] | M[14] |
| 44 | 0010 1100 | Miss | M[181] | **M[44]** | M[83] | M[2] | M[191] | M[88] | M[190] | M[14] |
| 186 | 1011 1010 | Miss | M[181] | M[44] | **M[186]** | M[2] | M[191] | M[88] | M[190] | M[14] |
| 253 | 1111 1101 | Miss | M[181] | M[44] | M[186] | **M[253]** | M[191] | M[88] | M[190] | M[14] |

We don’t need index for this problem, offset will be 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Address | Tag | Index | offset | Hit / Miss |
| 3 | 00000011 | 000 | 0 | Miss |
| 180 | 10110100 | 001 | 0 | Miss |
| 43 | 00101011 | 010 | 0 | Miss |
| 2 | 00000010 | 011 | 0 | Miss |
| 191 | 10111111 | 100 | 0 | Miss |
| 88 | 01011000 | 101 | 0 | Miss |
| 190 | 10111110 | 110 | 0 | Miss |
| 14 | 00001110 | 111 | 0 | Miss |
| 181 | 10110101 | 000 | 0 | Miss |
| 44 | 00101100 | 001 | 0 | Miss |
| 186 | 10111010 | 010 | 0 | Miss |
| 253 | 11111101 | 011 | 0 | Miss |

5.7.3

LRU:

Page numbers = 12

page hits = 2

miss = 10

miss rate = 10 / 12 = 0.83

MRU:

Page numbers = 12

page hits = 3

miss = 9

**miss rate = 9 / 12 = 0.75**

**Best miss rate: 0.75**

5.7.4

1)

Miss cycles = 100 / 0.5 = 200 clock cycles

CPI = 1.5 + 200 \* 0.07 = 15.5

Access time double:

Miss cycles = 100 \*2 / 0.5 = 400 clock cycles

CPI = 1.5 + 400 \* 0.07 = 29.5

Access time half:

Miss cycles = 100 / 1 = 100 clock cycles

CPI = 1.5 + 100 \* 0.07 = 8.5

2)

Miss cycles = 100 / 0.5 = 200 clock cycles

CPI = 1.5 + 200 \* 0.035 + 12 \* 0.07 = 9.34

Access time double:

Miss cycles = 100 \* 2 / 0.5 = 400 clock cycles

CPI = 1.5 + 400 \* 0.035 + 12 \* 0.07 = 16.34

Access time half:

Miss cycles = 100 / 1 = 100 clock cycles

CPI = 1.5 + 100 \* 0.035 + 12 \* 0.07 = 5.84

3)

Miss cycles = 100 / 0.5 = 200 clock cycles

CPI = 1.5 + 200 \* 0.015 + 28 \* 0.07 = 6.46

Access time double:

Miss cycles = 100 \* 2 / 0.5 = 400 clock cycles

CPI = 1.5 + 400 \* 0.015 + 28 \* 0.07 = 9.46

Access time half:

Miss cycles = 100 / 1 = 100 clock cycles

CPI = 1.5 + 100 \* 0.015 + 12 \* 0.07 = 4.96

5.7.5

Second level: CPI = 1.5 + 200 \* 0.035 +12 \* 0.07 = 9.34

Third level: CPI = 1.5 + 200 \* 0.013 + 12 \* 0.07 + 50 \* 0.035 = 6.69

From the value above, third level ache will provide a better performance in this case

Advantage:

Miss penalty is smaller and have a better performance

Disadvantage:

Hard to design and implement third level cache, cycle time will increase

5.7.6

Miss cycle = 100 /0.5 = 200 clock cycles

Second level CPI of direct mapping = 9.34

2 levels of 512KB cache

Second level CPI of 8-way associative cache = 6.46

4 levels of 512 KB cache