| PART 1 | | | | | | PART 2: Index is now 3 bits | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Word Address | Bin Address | Tag | Index | Hit? | | Word Address | Bin Address | Tag | Index | Hit? |
| 3 | 0000 0011 | 0 | 3 | Miss | | 3 | 0000 0011 | 0 | 1 | Miss |
| 180 | 1011 0100 | 11 | 4 | Miss | | 180 | 1011 0100 | 11 | 2 | Miss |
| 43 | 0010 1011 | 2 | 11 | Miss | | 43 | 0010 1011 | 2 | 5 | Miss |
| 2 | 0000 0010 | 0 | 2 | Miss | | 2 | 0000 0010 | 0 | 1 | Hit |
| 191 | 1011 1111 | 11 | 15 | Miss | | 191 | 1011 1111 | 11 | 7 | Miss |
| 88 | 0101 1000 | 5 | 8 | Miss | | 88 | 0101 1000 | 5 | 4 | Miss |
| 190 | 1011 1110 | 11 | 14 | Miss | | 190 | 1011 1110 | 11 | 7 | Hit |
| 14 | 0000 1110 | 0 | 14 | Miss | | 14 | 0000 1110 | 0 | 7 | Miss |
| 181 | 1011 0101 | 11 | 5 | Miss | | 181 | 1011 0101 | 11 | 2 | Hit |
| 44 | 0010 1100 | 2 | 12 | Miss | | 44 | 0010 1100 | 2 | 6 | Miss |
| 186 | 1011 1010 | 11 | 10 | Miss | | 186 | 1011 1010 | 11 | 5 | Miss |
| 253 | 1111 1101 | 15 | 13 | Miss | | 253 | 1111 1101 | 15 | 6 | Miss |

**PART 3:**

| | | Cache 3 | | |
|---|---|---|---|---|
| Word Address | Bin Address | Tag | Index | |
| 3 | 0000 0011 | 0 | 0 | Hit? |
| 180 | 1011 0100 | 5 | 5 | Miss |
| 43 | 0010 1011 | 1 | 2 | Miss |
| 2 | 0000 0010 | 0 | 0 | Miss |
| 191 | 1011 1111 | 5 | 7 | Hit |
| 88 | 0101 1000 | 2 | 6 | Miss |
| 190 | 1011 1110 | 5 | 7 | Miss |
| 14 | 0000 1110 | 0 | 3 | Hit |
| 181 | 1011 0101 | 3 | 5 | Miss |
| 44 | 0010 1100 | 1 | 3 | Miss |
| 186 | 1011 1010 | 5 | 6 | Miss |
| 253 | 1111 1101 | 7 | 7 | Miss |

Miss Rates:

1.) 100%
2.) 75%
3.) 83.33%

Total Cycles:

1.) 324
2.) 261
3.) 310

AMAT:

1.) 27
2.) 18.75
3.) 20.83

Therefore Cache 2 has the best performance of the bunch.

**PART 4:**

a.) 41984 Bytes
b.) 1024 Block Cache
c.) Hit time and miss penalty will increase with a larger block size. However, miss rates will increase with a smaller number of blocks.

**PART 5:**

0.07 * (12 + 0.035 * (50 + 0.013 * 100)) = 1.0ns

As a result of adding the L3 cache, memory access time will decrease, however this reduces the available space for other computational components.

PART 1: 1 Bit Offset

3: miss

180: miss

43: miss

2: hit

191: miss

88: miss

190: hit

14: miss

181: hit

44: miss

186: miss

253: miss

**PART 2:**

**Cache Content**

3: 3

180: 3, 180

43: 3, 180, 43

2: 3, 180, 43, 2

191: 3, 180, 43, 2, 191

88: 3, 180, 43, 2, 191, 88

190: 3, 180, 43, 2, 191, 88, 190

14: 3, 180, 43, 2, 191, 88, 190, 14

181: 181, 180, 43, 2, 191, 88, 190, 14

44: 81, 44, 43, 2, 191, 88, 190, 14

186: 181, 44, 186, 2, 191, 88, 190, 14

253:181, 44, 186, 253, 191, 88, 190, 14

**PART 3:**

**Cache Content (LRU)**

3: 1

180: 90

43: 1, 90, 21

2: 1, 90, 21

191: 1, 90, 21, 95

88: 1, 90, 21, 95, 44

190: 1, 90, 21, 95, 44

14: 1, 90, 21, 95, 44, 7

181: 1, 90, 21, 95, 44, 7

44: 1, 90, 21, 95, 44, 7, 22

186: 1, 90, 21, 95, 44, 7, 22, 93

253: 1, 90, 126, 95, 44, 7, 22, 93

PART 4:

**1**

.07 × 100 = 7

CPI = 7/1.5 = 4.667 → 5

*DOUBLE:*

.07 × 200 = 14

CPI = 14/1.5 = 9.334 → 10

*HALF:*

.07 × 50 = 3.5

CPI = 3.5/1.5 = 2.334 → 3

---

**2**

2.4.2 5.7.4.2

.07 × (12 + 0.035 × 100) = 1.085

CPI = 1.085/1.5 = 0.723 → 1

*DOUBLE:*

.07 × (12 + 0.035 × 200) = 1.33

CPI = 1.33/1.5 = 0.887 → 1

*HALF:*

.07 × (12 + 0.035 × 50) = 1.75

CPI = 1.75/1.5 = 1.167 → 2

---

**3**

.07 × (28 + 0.015 × 100) = 2.065

CPI = 2.065/1.5 = 1.377 → 2

*DOUBLE:*

.07 × (28 + 0.015 × 200) = 2.17

CPI = 2.17/1.5 = 1.447 → 2

*HALF:*

.07 × (28 + 0.015 × 50) = 2.71

CPI = 2.71/1.5 = 1.807 → 2

Regardless of the replacement algorithm, the miss rate is 9/12 or 3/4

**PART 5:**

Read Request List: 1, 7, 1, 7, 1, 7, 1, 7, 1, 7, 1, 7

Once the first two requests are processed, all others will be hits

PART 6:

This function can be used to index the address, but the data loss caused by XOR would require a larger bit field for the tag.

**PART 6:**

This performance goal is unreachable because of the 50ns access time.