# Solution for Chapter 8

8.2. Each column address strobe causes 8×4 = 32 bytes o be transferred.

(*a*) Latency = 5 clock cycles or 12.5 ns

Total time = 5 + 8 = 13 clock cycles, or 32.5 ns.

(*b*) A second column strobe is needed to transfer the second burst of 32 bytes. Therefore:

Latency = 5 clock cycles or 12.5 ns

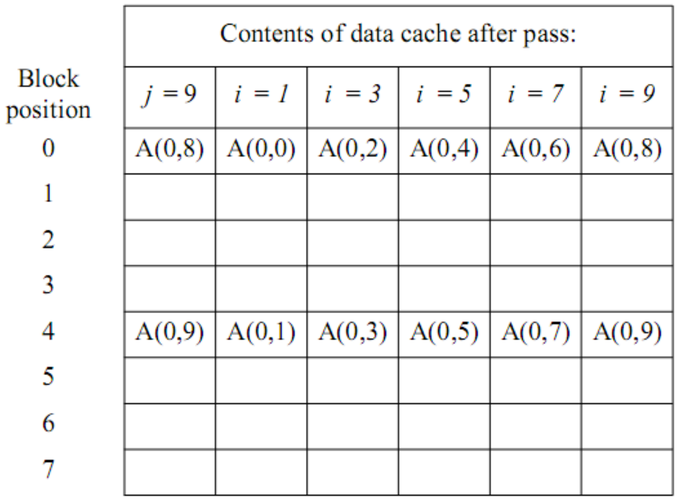
Total time = 5 + 8 + 2 + 8 = 23 clock cycles, or 57.5 ns.

8.3. A 16M module can be structured as 16 rows, each containing eight 1M×4 chips. A 24-bit address is required. Address lines $A_{19-0}$ should be connected to all chips. Address lines $A_{23-20}$ should be connected to a 4-bit decoder to select one of the 16 rows.

8.8. Each block contains 128 bytes, thus requiring a 7-bit Word field. There are 16 sets, requiring a 4-bit Set field. The remaining 21 bits of the address constitute the tag field.

8.10. For the first loop, the contents of the cache are as indicated in Figures 8.21 through 8.23. For the second loop, they are as follows.

(a) Direct-mapped cache



| Block position | Contents of data cache after pass: | | | | | |
|---|---|---|---|---|---|---|
| | *j* = 9 | *i* = *1* | *i* = *3* | *i* = *5* | *i* = *7* | *i* = *9* |
| 0 | A(0,8) | A(0,0) | A(0,2) | A(0,4) | A(0,6) | A(0,8) |
| 1 | | | | | | |
| 2 | | | | | | |
| 3 | | | | | | |
| 4 | A(0,9) | A(0,1) | A(0,3) | A(0,5) | A(0,7) | A(0,9) |
| 5 | | | | | | |
| 6 | | | | | | |
| 7 | | | | | | |

(b) Associative-mapped cache

| Block position | Contents of data cache after pass: | | | |
|---|---|---|---|---|
| | $j = 9$ | $i = 0$ | $i = 5$ | $i = 9$ |
| 0 | A(0,8) | A(0,8) | A(0,8) | A(0,6) |
| 1 | A(0,9) | A(0,9) | A(0,9) | A(0,7) |
| 2 | A(0,2) | A(0,0) | A(0,0) | A(0,8) |
| 3 | A(0,3) | A(0,3) | A(0,1) | A(0,9) |
| 4 | A(0,4) | A(0,4) | A(0,2) | A(0,2) |
| 5 | A(0,5) | A(0,5) | A(0,3) | A(0,3) |
| 6 | A(0,6) | A(0,6) | A(0,4) | A(0,4) |
| 7 | A(0,7) | A(0,7) | A(0,5) | A(0,5) |

(c) Set-associative-mapped cache

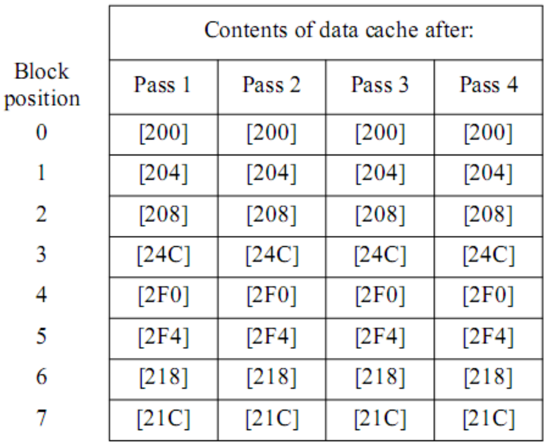
| | Block position | Contents of data cache after pass: | | | |
|---|---|---|---|---|---|
| | | $j = 9$ | $i = 3$ | $i = 7$ | $i = 9$ |
| Set 0 | 0 | A(0,8) | A(0,2) | A(0,6) | A(0,6) |
| | 1 | A(0,9) | A(0,3) | A(0,7) | A(0,7) |
| | 2 | A(0,6) | A(0,0) | A(0,4) | A(0,8) |
| | 3 | A(0,7) | A(0,1) | A(0,5) | A(0,9) |
| Set 1 | 0 | | | | |
| | 1 | | | | |
| | 2 | | | | |
| | 3 | | | | |

In all 3 cases, all elements are overwritten before they are used in the second loop. This suggests that the LRU algorithm may not lead to good performance if used with arrays that do not fit into the cache. Performance can be improved by introducing some randomness in the replacement algorithm.

8.11.The two least-significant bits of an address, $A_{1-0}$, specify a byte within a 32-bit word. For a direct-mapped cache, bits $A_{4-2}$ specify the block position in the cache. For a set-associative-mapped cache, bit $A_2$ specifies the set.

(a) Direct-mapped cache

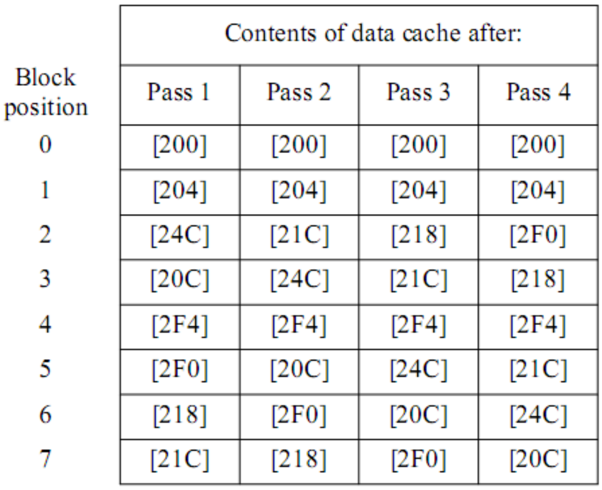
200: 0010 0000 0000
204: 0010 0000 0100
208: 0010 0000 1000
20C:0010 0000 1100
2F4: 0010 1111 0100
2F0: 0010 1111 0000
200
204
218: 0010 0001 1000
21C:0010 0001 1100
24C:0010 0100 1100
2F4



| Block position | Contents of data cache after: | | | |
|---|---|---|---|---|
| | Pass 1 | Pass 2 | Pass 3 | Pass 4 |
| 0 | [200] | [200] | [200] | [200] |
| 1 | [204] | [204] | [204] | [204] |
| 2 | [208] | [208] | [208] | [208] |
| 3 | [24C] | [24C] | [24C] | [24C] |
| 4 | [2F0] | [2F0] | [2F0] | [2F0] |
| 5 | [2F4] | [2F4] | [2F4] | [2F4] |
| 6 | [218] | [218] | [218] | [218] |
| 7 | [21C] | [21C] | [21C] | [21C] |

Hit rate = 33/48 = 0.69

(b) Associative-mapped cache

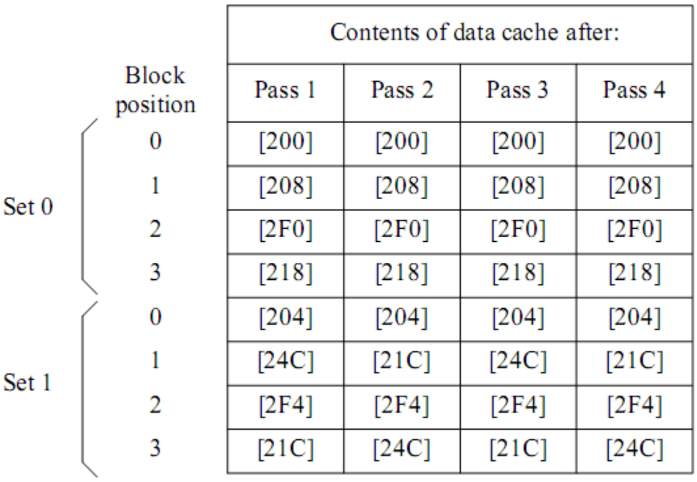


| Block position | Contents of data cache after: | | | |
|---|---|---|---|---|
| | Pass 1 | Pass 2 | Pass 3 | Pass 4 |
| 0 | [200] | [200] | [200] | [200] |
| 1 | [204] | [204] | [204] | [204] |
| 2 | [24C] | [21C] | [218] | [2F0] |
| 3 | [20C] | [24C] | [21C] | [218] |
| 4 | [2F4] | [2F4] | [2F4] | [2F4] |
| 5 | [2F0] | [20C] | [24C] | [21C] |
| 6 | [218] | [2F0] | [20C] | [24C] |
| 7 | [21C] | [218] | [2F0] | [20C] |

Hit rate = 21/48 = 0.44

(c) Set-associative-mapped cache

200: 0010 0000 0000
204: 0010 0000 0100
208: 0010 0000 1000
20C:0010 0000 1100
2F4: 0010 1111 0100
2F0: 0010 1111 0000
200
204
218: 0010 0001 1000
21C:0010 0001 1100
24C:0010 0100 1100
2F4



| | Block position | Contents of data cache after: | | | |
|---|---|---|---|---|---|
| | | Pass 1 | Pass 2 | Pass 3 | Pass 4 |
| Set 0 | 0 | [200] | [200] | [200] | [200] |
| | 1 | [208] | [208] | [208] | [208] |
| | 2 | [2F0] | [2F0] | [2F0] | [2F0] |
| | 3 | [218] | [218] | [218] | [218] |
| Set 1 | 0 | [204] | [204] | [204] | [204] |
| | 1 | [24C] | [21C] | [24C] | [21C] |
| | 2 | [2F4] | [2F4] | [2F4] | [2F4] |
| | 3 | [21C] | [24C] | [21C] | [24C] |

Hit rate = 30/48 = 0.63

8.12.The three least-significant bits of an address, $A_{2\sim0}$, specify a byte within a block (two 32-bit words). For a direct-mapped cache, bits $A_{4\sim3}$ specify the block position. For a set-associative-mapped cache, bit $A_3$ specifies the set.

(a) Direct-mapped cache

200: 0010 0000 0000
204: 0010 0000 0100
208: 0010 0000 1000
20C: 0010 0000 1100
2F4: 0010 1111 0100
2F0: 0010 1111 0000
200
204
218: 0010 0001 1000
21C: 0010 0001 1100
24C: 0010 0100 1100
2F4

| Block position | Contents of data cache after: | | | |
|---|---|---|---|---|
| | Pass 1 | Pass 2 | Pass 3 | Pass 4 |
| 0 | [200] | [200] | [200] | [200] |
| | [204] | [204] | [204] | [204] |
| 1 | [248] | [248] | [248] | [248] |
| | [24C] | [24C] | [24C] | [24C] |
| 2 | [2F0] | [2F0] | [2F0] | [2F0] |
| | [2F4] | [2F4] | [2F4] | [2F4] |
| 3 | [218] | [218] | [218] | [218] |
| | [21C] | [21C] | [21C] | [21C] |

Hit rate = 37/48 = 0.77

(b) Associative-mapped cache

| Block position | Contents of data cache after: | | | |
|---|---|---|---|---|
| | Pass 1 | Pass 2 | Pass 3 | Pass 4 |
| 0 | [200] | [200] | [200] | [200] |
| | [204] | [204] | [204] | [204] |
| 1 | [248] | [218] | [248] | [218] |
| | [24C] | [21C] | [24C] | [21C] |
| 2 | [2F0] | [2F0] | [2F0] | [2F0] |
| | [2F4] | [2F4] | [2F4] | [2F4] |
| 3 | [218] | [248] | [218] | [248] |
| | [21C] | [24C] | [21C] | [24C] |

Hit rate = 34/48 = 0.71

(c) Set-associative-mapped cache

200: 0010 0000 0000
204: 0010 0000 0100
208: 0010 0000 1000
20C:0010 0000 1100
2F4: 0010 1111 0100
2F0: 0010 1111 0000
200
204
218: 0010 0001 1000
21C:0010 0001 1100
24C:0010 0100 1100
2F4

| | Block position | Contents of data cache after: | | | |
|---|---|---|---|---|---|
| | | Pass 1 | Pass 2 | Pass 3 | Pass 4 |
| Set 0 | 0 | [200] | [200] | [200] | [200] |
| | | [204] | [204] | [204] | [204] |
| | 1 | [2F0] | [2F0] | [2F0] | [2F0] |
| | | [2F4] | [2F4] | [2F4] | [2F4] |
| Set 1 | 0 | [248] | [218] | [248] | [218] |
| | | [24C] | [21C] | [24C] | [21C] |
| | 1 | [218] | [248] | [218] | [248] |
| | | [21C] | [24C] | [21C] | [24C] |

Hit rate = 34/48 = 0.71

8.13. Larger size

- Fewer misses if most of the data in the block are actually used
- Wasteful if much of the data are not used before the cache block is ejected from the cache
- Larger miss penalty

Smaller size

- More misses
- Smaller miss penalty

8.22. (a) The maximum number of bytes that can be stored on this disk is $24 \times 14000 \times 400 \times 512 = 68.8 \times 10^9$ bytes.

(b) The data transfer rate is $(400 \times 512 \times 7200)/60 = 24.58 \times 10^6$ bytes/s.

(c) We need 9 bits to identify a sector, 14 bits for a track, and 5 bits for a surface. Thus, using a 32 bit word $b_{31} \cdots b_0$, a possible scheme is to use bits $b_{8-0}$ for sector, $b_{22-9}$ for track, and $b_{27-23}$ for surface identification. Bits $b_{31-28}$ would not be used.