**CS224**

**Lab 06**

**Section 003**

**Furkan Mert Aksakal**

**22003191**

**11.12.2024**

1.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **No.** | **Cache Size**  **KB** | **N**  **way cache** | **Word Size** | **Block size (no. of words)** | **No. of Sets** | **Tag Size in bits** | **Index Size (Set No.) in bits** | **Word Block Offset Size in bits** | **Byte Offset Size in bits** | **Block Replacement Policy Needed (Yes/No)** |
| 1 | 64 | 1 | 32 bits | 4 | 4096 | 16 bits | 12 bits | 2 bits | 2 bits | No |
| 2 | 64 | 2 | 32 bits | 4 | 2048 | 17 bits | 11 bits | 2 bits | 2 bits | Yes (2-way) |
| 3 | 64 | 4 | 32 bits | 8 | 512 | 19 bits | 9 bits | 3 bits | 2 bits | Yes (4-way) |
| 4 | 64 | Full | 32 bits | 8 | 1 | 27 bits | 0 bits | 3 bits | 2 bits | Yes (Fully Associative) |
| 9 | 128 | 1 | 16 bits | 4 | 8192 | 15 bits | 13 bits | 2 bits | 2 bits | No |
| 10 | 128 | 2 | 16 bits | 4 | 4096 | 16 bits | 12 bits | 2 bits | 2 bits | Yes (2-way) |
| 11 | 128 | 4 | 16 bits | 16 | 512 | 19 bits | 9 bits | 4 bits | 2 bits | Yes (4-way) |
| 12 | 128 | Full | 16 bits | 16 | 1 | 26 bits | 0 bits | 4 bits | 2 bits | Yes (Fully Associative) |

2.

a)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Instruction** | **Iteration No.** | | | | |
| **1** | **2** | **3** | **4** | **5** |
| lw $t1, 0x4($0) | Compulsory miss (first access to block 2) | Conflict miss (block 6 replaced block 2) | Conflict miss | Conflict miss | Conflict miss |
| lw $t2, 0xC($0) | Compulsory miss (first access to block 6) | Conflict miss (block 4 replaced block 6) | Conflict miss | Conflict miss | Conflict miss |
| lw $t3, 0x8($0) | Compulsory miss (first access to block 4) | Conflict miss (block 2 replaced block 4) | Conflict miss | Conflict miss | Conflict miss |

b)

Number of blocks = Cache capacity / Block size 🡺 8/2 = 4 blocks

Data bits in each block = Block size × Word size 🡺 2 words × 32 bits = 64 bits

Tag bits = 32 - (index + block offset + byte offset) 🡺 32 - (2 + 1 + 2) = 27 bits

Data bits = 64 bits // Tag bits = 27 bits // V bit = 1 bit 🡺 Total per block = 64 + 27 + 1 = 92 bits

Total = Number of blocks × Bits per block 🡺 4 × 92 = 368 bits

c)  
AND gates: 8 (4 for tag comparison with valid bits + 4 for decoder)

OR gates: 3 (inside the multiplexer for final data selection)

Equality Comparators: 4 (one 27-bit comparator per block)

Multiplexers: 1 (64-bit 4-to-1 MUX)

3.

a)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Instruction** | **Iteration No.** | | | | |
| **1** | **2** | **3** | **4** | **5** |
| lw $t1, 0x4($0) | Compulsory miss (cache: [4, -]) | Capacity miss (replaces addr 12, cache: [8, 4]) | Capacity miss (replaces addr 12, cache: [8, 4]) | Capacity miss (replaces addr 12, cache: [8, 4]) | Capacity miss (replaces addr 12, cache: [8, 4]) |
| lw $t2, 0xC($0) | Compulsory miss (cache: [4, 12]) | Capacity miss (replaces addr 8, cache: [4, 12]) | Capacity miss (replaces addr 8, cache: [4, 12]) | Capacity miss (replaces addr 8, cache: [4, 12]) | Capacity miss (replaces addr 8, cache: [4, 12]) |
| lw $t3, 0x8($0 | Capacity miss (replaces addr 4, cache: [12, 8]) | Capacity miss (replaces addr 4, cache: [12, 8]) | Capacity miss (replaces addr 4, cache: [12, 8]) | Capacity miss (replaces addr 4, cache: [12, 8]) | Capacity miss (replaces addr 4, cache: [12, 8]) |

b)

Block size = 1 word = 32 bits

Number of blocks = 2

Main memory address = 32 bits

Byte offset = log₂(4) = 2 bits

Tag bits = 32 - 2 = 30 bits

Data bits = 32 bits (1 word) // Tag bits = 30 bits // V bit = 1 bit

Per block = 32 + 30 + 1 = 63 bits

Data + Tag + V bits = 63 bits × 2 blocks = 126 bits

LRU bits = 1 bit

Total cache memory size = 126 + 1 = 127 bits

c)

AND gates: 3 (2 for tag comparison with valid bits + 1 for LRU)

OR gates: 2 (1 for multiplexer data selection + 1 for LRU)

Equality Comparators: 2 (one 30-bit comparator per block)

Multiplexers: 1 (32-bit 2-to-1 MUX)

4.

L1 Access:

Hit time = 1 cycle // Miss rate = 20% = 0.2 // On miss, goes to L2

L2 Access:

Hit time = 4 cycles // Miss rate = 5% = 0.05 // On miss, goes to main memory // When accessed from L1 miss: 0.2 × (4 + 0.05 × main\_memory\_time)

Main Memory Access:

Access time = 10 × L2 time = 10 × 4 = 40 cycles

AMAT Calculation:

Clock rate = 4 GHz = 4 × 10⁹ cycles/second

Time per instruction = 2.2 cycles

Total cycles needed = 10¹² × 2.2 = 2.2 × 10¹² cycles

Total time = (2.2 × 10¹²) ÷ (4 × 10⁹)

Total time = 550 seconds = 9.17 minutes