ICS 233 Homework Assignment 05 Key

**Question 1**: In a 2-level **non-strict hierarchy** memory system, the average access time without Level 1 is 110 nSeconds and with Level 1 it is 35 nSeconds.

If level 1 access time is 20 nSeconds, then

1. What is the Hit ratio ?
2. If the average access time is increased by 18%, what will be the % change in the hit ratio?

**Note**: Show all computations

1. T2 = 110 nSeconds

Tavg = 35 nSeconds

T1 = 20 nSeconds

Tavg = H1T1 + (1 – H1) T2

35 = H1T1 + (1 – H1) T2

35 = H1 (T1  - T2) + T2

35 = -90H1 + 110

90 H1 = 110 – 35

 H1 = 75 / 90 = 0.8333

🡪 H1 = 83.33 %

1. Tavg = 35 nSeconds

Increase = 18/100 \* 35 = 6.3 nSeconds

🡪 Tavg = 41.3 nSec

But Tavg = H1T1 + (1 – H1) T2

 🡪 41.3 = 20 H1 + (1 – H1) 110

 90 H1 = 110 – 41.3

 H1 = 68.7 / 90 = 0.7633

🡪 H1 = 76.33 %

Percent change in hit ratio = 83.33 % - 76.33 % = 7 % [Decrease]

**Question 2**: In a three-level **strict hierarchy** system:

L1 cache has access time T1 = 4 nanoseconds and hit-ratio h1 = 0.92
L2 cache has access time T2 = 9 nanoseconds and hit-ratio h2 = 0.94
Main memory has access time T3 = 80.0 nanoseconds

What is the average access time?

**Note**: Show all computations

Tavg = H1 T1 + (1 – H1) H2(T1 + T2) + (1 – H1)(1 – H2)(T1 + T2 + T3)

 = 0.92 \* 4 + (1 – 0.92)\*0.94\*(4 + 9) + (1 – 0.92)(1 – 0.94)(4 + 9 + 80)

 = 3.68 + 0.08 \* 0.94 \* 13 + 0.08 \* 0.06 \* 93 = **5.104 nanoseconds**

**Question 3:** Consider a 32-bit physical address memory system with block size 16 bytes and a 32 blocks direct-mapped cache. The cache is initially empty.

The following decimal memory addresses are referenced:

 994, 1022, 2019, 5106, 1006 and 1020

Map the addresses to cache blocks and indicate whether hit or miss.

**Note**: You must use the hexadecimal approach in solving this question.

 You must also show the computations of dividing the memory address into tag bits, cache index bits, and block offset bits.

Solution:

Number of address bits = 32 bits

Number of cache blocks = 32 🡺Cache index bits = log2 32 = log2 25 = 5 bits

Block size = 16 bytes 🡺Block offset bits = log2 16 = log2 24 = 4 bits

🡺Tag bits = 32 – 5 – 4 = 23

 

**Hexadecimal approach:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Access times | Address | Tag (23 bits) | Cache index (5 bits) | Status |
| T1 | 994 = 000003E2h | 0. . .01 | 11110  | Cold miss, Memory block 000003EhCopied to cache block 11110B |
| T2 | 1022 = 000003FEh | 0. . .01 | 11111  | Cold miss, Memory block 000003Fh copied to cache block 11111B  |
| T3 | 2019 = 000007E3h | 0. . .011 | 11110  | Conflict miss (different tag to block 000003E), memory block 000003Eh ejected from cache. Memory block 000007Eh copied to cache block 11110B |
| T4 | 5106 = 000013F2h | 0. . 01001 | 11111  | Conflict miss (different tag to block 000003Fh),memory block 000003Fh ejected from cache block 11111B, memory block 000013Fh copied to cache block 11111B |
| T5 | 1006 = 000003EEh | 0. . .01 | 11110  | Conflict miss, memory block 0000007Eh is ejected. Memory block 000003Eh is copied to cache block 11110B |
| T6 | 1020 = 000003FCh | 0. . .01 | 11111  | Conflict miss. memory block 000013Fh ejected. Memory block 000003Fh copied to cache block 11111B |

Cache:

|  |  |  |  |
| --- | --- | --- | --- |
| Cache Block# | Valid Bit | Memory block in Cache | Access times |
| 11110B | 0 🡪 1 (T1)  | ~~000003Eh~~ ~~000007Eh~~, 000003Eh | T1, T3, T5 |
| 11111B | 0 🡪 1 (T2) | ~~000003Fh~~ ~~000013Fh~~, 000003Fh | T2, T4,T6 |

**Question 4:** Consider a 2-way set associative memory system in which:

 Main memory size m = **128** words

 Cache memory size c = 32 words

 Block size b = 4 words

 Replacement algorithm: FIFO

Write policy: write-back + write-allocate

**Note: Within a set, assume that main memory blocks are copied sequentially**

Given that the cache is initially empty and that the following decimal memory addresses are referenced:

 90 [Write], 86 [Read], 70 [Read], 116 [Write], 95 [Read], 84 [Write], 100 [Read]

Map the addresses to cache blocks and indicate whether hit or miss.

**Note: You must use the decimal approach in solving this question and you must show all computations.**

**Decimal approach:**

Mapping: CacheSetNumber = MemoryBlockNumber mod NumberOfSets

 = MemoryBlockNumber mod 4

MemoryBlockNumber = $\left⌊\frac{ByteAddress}{BlockSize}\right⌋$

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Access Time | Decimal address | Block number | Cache Set | Comment |
| T1 | 90 | $$\left⌊\frac{90}{4}\right⌋=22$$ | 22 mod 4 = 2 | Write cold miss. Memory block 22 copied in first block of set 2. Dirt bit set. |
| T2 | 86 | $$\left⌊\frac{86}{4}\right⌋=21$$ | 21 mod 4 = 1 | Read cold miss. Memory lock 21 copied in first block of set 1 |
| T3 | 70 | $$\left⌊\frac{70}{4}\right⌋=17$$ | 17 mod 4 = 1 | Read cold miss. Memory block 17 copied to second block of set 1. |
| T4 | 116 | $$\left⌊\frac{116}{4}\right⌋=29$$ | 29 mod 4 = 1 | Write conflict miss. Memory block 21 ejected from cache. Memory block 29 copied to cache. Dirty bit set to 1 |
| T5 | 95 | $$\left⌊\frac{95}{4}\right⌋=23$$ | 23 mod 4 = 3 | Read cold miss. Block 23 copied to first block of set 3 |
| T6 | 84 | $$\left⌊\frac{84}{4}\right⌋=21$$ | 21 mod 4 = 1 | Write conflict miss. Memory block 17 ejected from cache memory. Block 21 copied to second block of set 1. Dirty set 1. |
| T7 | 100 | $$\left⌊\frac{100}{4}\right⌋=25$$ | 25 mod 4 = 1 | Read conflict miss. Memory block 29 ejected from cache memory. Memory block 25 copied to block 1 of set 1. Dirt bit cleared. Memory block 29 updated in main memory. |

Cache:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Set# | Valid Bit | Dirty bit | Memory block in Cache | Access times |
| 0 | 0  | 0  |   |  |
| 0 | 0 |  |  |
| 1 | 0 🡪 1 (T2)  | 0 🡪 1 (T4), 1 🡪 0 (T7) | ~~21~~, ~~29~~, 25  | T2, T4, T7 |
| 0 🡪 1 (T3) | 0 🡪 1 (T6) | ~~17~~, 21 | T3, T6 |
| 2 | 0 🡪 1 (T1)  | 0 🡪 1 (T1) | 22 | T1 |
| 0 | 0 |  |  |
| 3 | 0 🡪 1 (T5)  | 0  | 23 | T5 |
| 0  | 0  |  |  |

**Question 05**: For each of the following questions show detailed steps and provide instruction format diagrams:

1. What is the machine code for: **MOV DX, ES: [2532h]**

|  |  |  |  |
| --- | --- | --- | --- |
| BYTE 1 | BYTE 2 | BYTE 3 | BYTE 4 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | **00110010** | **00100101** |
| Opcode | D | W | MOD | REG | R/M | Dir Addr LB | Dir Addr HB |

Opcode for MOV r16, r/m16 = 100010B

D = 1 (DX destination operand)

W bit = 1 (16-bits)

Therefore byte 1 is 10001011B = 8BH

• [disp16] 🡺 MOD = 00

• REG = 010 (code for DX)

• [disp16] 🡺 R/M = 110

Therefore Byte 2 is 00000101B = 16H

* Direct address low byte = 32h = **00110010B**
* Direct address high byte = 25h = **00100101B**
* Segment register override ES: is 26H

Hence the machine code for **MOV DX, ES:[3532H] is 268B163225H**

1. What is the machine code for: **MOV EBX, [EDX + 452H]**

|  |  |  |  |
| --- | --- | --- | --- |
| BYTE 1 | BYTE 2 | BYTE 3 | BYTE 4 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | **01010010** | **00000100** |
| Opcode | D | W | MOD | REG | R/M | Low Disp | High Disp |

Opcode for MOV r32, r/m32 = 100010B

D = 1 (EBX destination operand)

W bit = 1(32-bits)

Therefore byte 1 is 10001011B = 8BH

• [EDX + disp32] 🡺 MOD = 10

• REG = 011 (code for EBX)

• [EDX + disp32] 🡺 R/M = 010

Therefore Byte 2 is 10000001B = 9AH

* Low displacement byte = 52H = **01010010B**
* High displacement byte = 04H = **00000100B**

Hence the machine code for **MOV EBX, [EDX + 452H] is 8B9A5204H**

1. What is the machine code for: **MOV ECX, [EAX + 8\*EBX]**

|  |  |  |
| --- | --- | --- |
| BYTE 1 | BYTE 2 | SIB |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | **1** | **1** | **0** | **1** | **1** | 0 | 0 | 0 |
| Opcode | D | W | MOD | REG | R/M | scale | index | base |

Opcode for MOV r32, r/m32 = 100010B

D = 1 (ECX destination operand)

W bit = 1 (32-bits)

Therefore byte 1 is 10001011B = 8BH

• [Scaled index] 🡺 MOD = 00

• REG = 001 (code for ECX)

• [Scaled index] 🡺 R/M = 100

Therefore byte 2 is 00001100B = 0CH

* Scale = 8 🡺 **11**
* Index register = EBX = **011**
* Base register = EAX = 000

Therefore SIB is 11011000B = D8H

Hence the machine code for **MOV ECX, [EAX + 8\*EBX]**  is 8B0CD8H

1. Given **100010** is an opcode for **MOV**. What is the x86 assembly language instruction for the machine language instruction **8903H** in 32-bit mode?

Solution:

* Convert 8903H to binary 🡺 **1000 1001** 00**00** **0**011
* Opcode = 100010 🡺 MOV
* D = **0** 🡺 data flows from REG to R/M
* **W = 1** 🡺 32-bit data
* MOD = **00** and **R/M = 011 🡺 [EBX] i.e., indexed addressing mode**
* **REG = 000** 🡺 EAX register

Hence, the assembly language instruction is: **MOV [EBX], EAX**