# Habib University

**Dhanani School of Science and Engineering**

**Computer Architecture**
**EE 371 / CS 330 / CE 321 (Spring 2024)**

# Solution of Homework 4

| | |
|---|---|
| **Release Date:** April 5, 2024 | **Due by:** April 18, 2024 11:59 PM |

| | |
|---|---|
| **Total marks:** 100 | **Marks obtained:** |

**Purpose:**
This assignment hopefully helped you understand the performance of cache operations.

**Instructions:**
1. This assignment should be done in pairs.
2. All questions should be answered in **black ink only**.
3. Scan your answer sheet and upload it on HU LMS before the due date.

**Grading Criteria:**
1. Your assignments will be checked by instructor/TA.
2. You can also be asked to give a viva where you will be judged whether you understood the question yourself or not. If you are unable to correctly answer the question you have attempted right, you may lose your marks.
3. Zero will be given if the assignment is found to be plagiarized.
4. Untidy work will result in a reduction of your points.

**Submission policy:**
1. No submission will be accepted after the instructor releases the solution on HU LMS.

**CLO Assessment:**
This assignment assesses students for the following course learning outcomes.

| Course Learning Outcomes | | CLO Assessed |
|---|---|---|
| **CLO 1** | ***Explain*** the role of ISA in modern processors and instruction encodings and assembly language programming | |
| **CLO 2** | ***Explain*** the architecture and working of a single cycle processor | |
| **CLO 3** | ***Design*** the architecture to mitigate issues of a pipelined processor | |
| **CLO 4** | ***Analyze the*** performance of cache operations | |

**Question 1 [10 Marks]**

A cache is named according to the amount of data it contains (i.e., a 4 KiB cache can hold 4 KiB of data); however, caches are also required to store metadata such as tags and valid bits. We will see how block size affects the size of metadata needed for a cache. For all parts, assume that addresses are 64 bits, and we are dealing with words of 4 bytes.

**a)** Calculate the total number of bits required to implement a 128KiB cache with two-word per block. **[3 Marks]**

128KiB 128/4K word = 32K words

2 words/block $2^m = 2$ **m = 1**

2 words/block 32K words/2 16K blocks $2^{14}$ blocks

16K blocks $2^{14}$ blocks **index = 14**

Tag = 64 − (index + m + 2) **Tag = 47 bits**

single line size = Tag + valid bit + 2*word_size

single line size = 47 + 1 + 2*32 = 112 bits

Total blocks/lines = 214 cache size = $112 * 2^{14}$

**cache size = 224KiB**

**b)** Now repeat the calculation in part a with 8, 16 and 32 words per block. **[3 Marks]**

| Block size | **8 words/block** | **16 words/ block** | **32 words/block** |
|---|---|---|---|
| **m (using $2^m$)** | 3 | 4 | 5 |
| **Blocks** | $2^{12}$ | $2^{11}$ | $2^{10}$ |
| **Index** | 12 | 11 | 10 |
| **Tag** | 47 | 47 | 47 |
| **Line Size** | 304 | 560 | 1004 |
| **Total Size** | 152 KiB | 140 KiB | 134 KiB |

**c)** Find the ratio of total cache size to the amount of data (i.e. data size) for all the cases in part ‘a’ and ‘b’. Also, plot (roughly) words per block vs the calculated ratio. **[4 Marks]**

| Block Size | **Ratio** |
|---|---|
| **2 words/blocks** | 224/128 = 1.75 |
| **8 words/block** | 152/128 = 1.1875 |
| **16 words/block** | 140/128 = 1.0937 |
| **32 words/block** | 134/128 = 1.04 |

## Question 2 [10 Marks]

Applications that engage in the playback of audio or video content are categorized under a group of tasks known as "streaming" workloads. These tasks involve the retrieval of substantial data volumes but typically involve minimal data reuse. Let us examine a scenario involving video streaming, where a workload accesses a 512 KiB working set in a sequential manner. The access pattern can be represented by the following sequence of word address stream:

*Address stream: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 ...*

Assume a 64 KiB direct-mapped cache with a 32-byte block for the following

**a.** What is the miss rate for the address stream above? How is this miss rate sensitive to the size of the cache or the working set? How would you categorize the misses this workload is experiencing, based on the 3C (Compulsory Misses, Capacity Misses, Conflict Misses) Model? **[4 Marks]**

The addresses are given as word addresses; each 32-byte block contains eight words. Thus, every eight access will be a miss (i.e., a miss rate of 1/8). All misses are compulsory misses. The miss rate is not sensitive to the size of the cache or the size of the working set. It is, however, sensitive to the access pattern and block size

**b.** Re-compute the miss rate when the cache block size is 16 bytes, 64 bytes, and 128 bytes. What kind of locality is this workload exploiting? **[3 Marks]**

The miss rates are 1/4 , 1/16 , and 1/32, respectively, The workload is exploiting spatial locality.

**c.** “Prefetching” is a technique that leverages predictable address patterns to speculatively bring in additional cache blocks when a particular cache block is accessed. One example of prefetching is a stream buffer that prefetches sequentially adjacent cache blocks into a separate buffer when a particular cache block is brought in. If the data are found in the

prefetch buffer, it is considered as a hit, moved into the cache, and the next cache block is prefetched. Assume a two-entry stream buffer; and assume that the cache latency is such that a cache block can be loaded before the computation on the previous cache block is completed.
What is the miss rate for the address stream above? **[3 Marks]**

In this case the miss rate is 0: The prefetch buffer always has the next request ready.

**Question 3 [10 Marks]**

<u>Considering the address size of 64 or 32-bits,</u> fill in the data for different types of caches.

| | Blocks | Data per block | Sets | Associativity -ways | Tag Bits (Depends on what student chose) | | Index Bits | Offset Bits |
|---|---|---|---|---|---|---|---|---|
| | | | | | 64-bits | 32-bits | | |
| Fully Associative Cache | 32 | 4 words | -- | 32-ways | 60 | 28 | - | 4 |
| Direct Mapped Cache | 32 | 8 words | -- | 1-way | 54 | 22 | 5 | 5 |
| Set Associative Cache | 32 | 8 words | 4 | 8-ways | 57 | 25 | 2 | 5 |
| Direct Mapped Cache | 64 | 4 words | -- | 1-way | 54 | 22 | 6 | 4 |
| Set Associative Cache | 128 | 8 words | 32 | 4-ways | 54 | 22 | 5 | 5 |
| Set Associative Cache | 256 | 8 words | 32 | 8-ways | 54 | 22 | 5 | 5 |
| Fully Associative Cache | 512 | 8 words | -- | 51-ways | 59 | 27 | - | 5 |
| Direct Mapped Cache | 1024 | 8 words | -- | 1-way | 49 | 17 | 10 | 5 |
| Set Associative Cache | 2048 | 4 words | 64 | 32-ways | 54 | 22 | 6 | 4 |
| Direct Mapped Cache | 4096 | 8 words | -- | 1-way | 47 | 15 | 12 | 5 |

**Calculation of Offset-bits:**
(Data of blocks in words) * 4 = X
Offset Bits = $\log_2(X)$
**Calculation of Index-bits:**
*<u>For Direct-Mapped:</u>*
$\log_2(Blocks)$
*<u>For Set-Associative:</u>*
Associativity = Blocks or Number of Cache Lines / Sets
Index Bits= $\log_2(Associativity)$
*<u>Fully Association:</u>*
Index Bits = 0
**Calculation of Tag Bits =** address size[1] - Index Bits - Offset Bits

[1]Depends whether you chose 32 or 64-bit system

**Question 4 [05 Marks]**

Cache block size (B) can affect both miss rate and miss latency. Assuming a machine with a base CPI of 1, and an average of 1.35 references (both instruction and data) per instruction, find the block size that minimizes the total miss latency given the following miss rates for various block sizes. Cache block size is assumed to be in bytes.

| | | | | |
|---|---|---|---|---|
| 8: 4% | 16: 3% | 32: 2% | 64: 1.5% | 128: 1% |

**a)** What is the optimal block size for a miss latency of 20 × B cycles? **[2.5 Marks]**

AMAT for B = 8: 0.040 × (20 × 8) = 6.40
AMAT for B = 16: 0.030 × (20 × 16) = 9.60
AMAT for B = 32: 0.020 × (20 × 32) = 12.80
AMAT for B = 64: 0.015 × (20 × 64) = 19.20
AMAT for B = 128: 0.010 × (20 × 128) = 25.60
B = 8 is optimal

**b)** What is the optimal block size for a miss latency of 24 + B cycles? **[2.5 Marks]**

AMAT for B = 8: 0.040 × (24 + 8) = 1.28
AMAT for B = 16: 0.030 × (24 + 16) = 1.20
AMAT for B = 32: 0.020 × (24 + 32) = 1.12
AMAT for B = 64: 0.015 × (24 + 64) = 1.32
AMAT for B = 128: 0.010 × (24 + 128) = 1.52
B = 32 is optimal

**Question 5 [10 Marks]**

For a direct-mapped cache design with a 32-bit address, the following bits of the address are used to access the cache.

| **Tag** | **Index** | **Offset** |
|---|---|---|
| 32-10 | 9-3 | 2-0 |

Offset = 3 bits
Index = 7 bits
Tag = 22 bits

a) **[03 Marks]** What is the cache block size (in words)?

Offset = 3 bits = 23 bytes in a block = 8 bytes in a block

*As a single word is 4 bytes.*

8/4 = **2 words in a block.**

b) **[03 Marks]** How many blocks does the cache have?

There are 7 bits for the index which means there are 27 blocks = 128 blocks

c) **[04 Marks]** What is the ratio between total bits required for such a cache implementation over the data storage bits?

The total number of bits in a direct-mapped cache are:
$2^n$ * (block size + tag size + valid field size)
where;

| | | |
|---|---|---|
| n = 7 (index bits) | valid = 1 bit | block size = 2 words or 2*32 bits |

$2^7$ * ((2*32) + 22 + 1) = 11136 bits

Nominal cache size i.e. specified in specifications is:
Number of blocks * bytes per block * bits in a byte = 128*2*8 bits = 2048

Ratio of actual cache size (with tag and valid bits) to that of nominal cache size = 11136/2048 = 5.4375

## Question 6 [10 Marks]

We are given 4 arrays of size 6. Each element in an array is of 32 bytes i.e., one word. Following is the data stored in the array:

A = (27, 11, 23, 10, 28, 12)
B = (20, 35, 33, 48, 10, 32)
C = (40, 11, 9, 42, 49, 20)
D = (11, 31, 17, 37, 13, 2)

The array data is arranged in main memory as follows:

| | |
|---|---|
| 00000 | A[0] |
| 00001 | A[1] |
| 00010 | A[2] |
| 00011 | A[3] |
| 00100 | A[4] |
| 00101 | A[5] |
| 00110 | |
| 00111 | |
| 01000 | B[0] |
| 01001 | B[1] |

| | |
|---|---|
| 01010 | B[2] |
| 01011 | B[3] |
| 01100 | B[4] |
| 01101 | B[5] |
| 01110 | |
| 01111 | |
| 10000 | C[0] |
| 10001 | C[1] |
| 10010 | C[2] |
| 10011 | C[3] |
| 10100 | C[4] |
| 10101 | C[5] |
| 10110 | |
| 10111 | |
| 11000 | D[0] |
| 11001 | D[1] |
| 11010 | D[2] |
| 11011 | D[3] |
| 11100 | D[4] |
| 11101 | D[5] |
| 11110 | |
| 11111 | |

We are given a direct mapped cache which contains 8 blocks (each block will contain one word). Insert the following elements in cache one by one and mention whether it was a hit or a miss. Assume that the first block of the cache will be populated by the first element of the array and so on. First insertion is already done so that you may get the idea.

| **Data to be Inserted** | **Hit/Miss** | **Cache Index** | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| A[0] | M | A[0] | | | | | | | |
| A[1] | M | A[0] | A[1] | | | | | | |
| A[2] | M | A[0] | A[1] | A[2] | | | | | |
| A[1] | H | A[0] | A[1] | A[2] | | | | | |
| A[5] | M | A[0] | A[1] | A[2] | | | A[5] | | |
| B[5] | M | A[0] | A[1] | A[2] | | | B[5] | | |
| B[4] | M | A[0] | A[1] | A[2] | | B[4] | B[5] | | |
| B[3] | M | A[0] | A[1] | A[2] | B[3] | B[4] | B[5] | | |
| B[3] | H | A[0] | A[1] | A[2] | B[3] | B[4] | B[5] | | |
| B[4] | H | A[0] | A[1] | A[2] | B[3] | B[4] | B[5] | | |
| D[1] | M | A[0] | D[1] | A[2] | B[3] | B[4] | B[5] | | |
| D[2] | M | A[0] | D[1] | D[2] | B[3] | B[4] | B[5] | | |
| D[3] | M | A[0] | D[1] | D[2] | D[3] | B[4] | B[5] | | |
| D[4] | M | A[0] | D[1] | D[2] | D[3] | D[4] | B[5] | | |
| C[3] | M | A[0] | D[1] | D[2] | C[3] | D[4] | B[5] | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| C[2] | M | A[0] | D[1] | C[2] | C[3] | D[4] | B[5] | | |
| C[4] | M | A[0] | D[1] | C[2] | C[3] | C[4] | B[5] | | |
| C[2] | H | A[0] | D[1] | C[2] | C[3] | C[4] | B[5] | | |

**What is the Hit Ratio and the Miss Ratio in the above case?**

Hit ratio = number of hits / number of accesses = 4/18 = 0.22 = 22%

Miss ratio = number of misses / number of accesses = 14/18 = 0.78 = 78%

**Question 7 [10 Marks]**

Whenever an element from an array is accessed, it is most probable that some other remaining elements of the array are also accessed. Repeat the same task as in Question 6 but this time design a cache with 4 blocks in which each block can accommodate 2 words. The first insertion is done again so that you may get the idea.

| **Data to be Inserted** | **Hit/Miss** | **Cache Index** | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | **0** | | **1** | | **2** | | **3** | |
| A[0] | M | A[0] | A[1] | | | | | | |
| A[1] | H | A[0] | A[1] | | | | | | |
| A[2] | M | A[0] | A[1] | A[2] | A[3] | | | | |
| A[1] | H | A[0] | A[1] | A[2] | A[3] | | | | |
| A[5] | M | A[0] | A[1] | A[2] | A[3] | A[4] | A[5] | | |
| B[5] | M | A[0] | A[1] | A[2] | A[3] | B[4] | B[5] | | |
| B[4] | H | A[0] | A[1] | A[2] | A[3] | B[4] | B[5] | | |
| B[3] | M | A[0] | A[1] | B[2] | B[3] | B[4] | B[5] | | |
| B[3] | H | A[0] | A[1] | B[2] | B[3] | B[4] | B[5] | | |
| B[4] | H | A[0] | A[1] | B[2] | B[3] | B[4] | B[5] | | |
| D[1] | M | D[0] | D[1] | B[2] | B[3] | B[4] | B[5] | | |
| D[2] | M | D[0] | D[1] | D[2] | D[3] | B[4] | B[5] | | |
| D[3] | H | D[0] | D[1] | D[2] | D[3] | B[4] | B[5] | | |
| D[4] | M | D[0] | D[1] | D[2] | D[3] | D[4] | D[5] | | |
| C[3] | M | D[0] | D[1] | C[2] | C[3] | D[4] | D[5] | | |
| C[2] | H | D[0] | D[1] | C[2] | C[3] | D[4] | D[5] | | |
| C[4] | M | D[0] | D[1] | C[2] | C[3] | C[4] | C[5] | | |
| C[2] | H | D[0] | D[1] | C[2] | C[3] | C[4] | C[5] | | |

What is the Hit Ratio and the Miss Ratio in this case? Is it better than the previous? Does loading the whole array into the cache help us in accessing the elements fast?

Hit ratio = number of hits / number of accesses = 8/18 = 0.44 = 44%
Miss ratio = number of misses / number of accesses = 10/18 = 0.56 = 56%

The hit rate has improved in this case. Yes, loading two array elements into the cache helps in faster access time.
Loading the entire array into cache will improve the spatial locality.

**Question 8 [20 Marks]**

Assume that main memory accesses take 70 ns and that 36% of all instructions access data memory. The following table shows data for L1 caches attached to each of two processors, P1 and P2.

| | L1 size | L1 miss rate | L1 hit time |
|---|---|---|---|
| P1 | 2 KiB | 8% | .66 nsec |

|  |  |  |  |
|---|---|---|---|
| P2 | 4 KiB | 6% | .9 nsec |

**a)** If the L1 hit time determines the cycle times for P1 and P2, what are their respective clock rates? **[3 Marks]**

|  |  |
|---|---|
| P1 | 1.515 Ghz |
| P2 | 1.11 Ghz |

**b)** What is the Average Memory Access Time for P1 and P2 (in cycles) **[2 Marks]**

|  |  |  |
|---|---|---|
| P1 | 6.31 ns | 9.56 cycles |
| **P2** | 5.11 ns | 5.68 cycles |

## For P1

all memory accesses require at least one cycle (to access L1).
8% of memory accesses additionally require a 70 ns access to main memory.
This is 70/0.66 = 106.06 cycles. However, we can't divide cycles; therefore, we must round up to 107 cycles.
Thus, the Average Memory Access time is **1 + 0.08*107 = 9.56 cycles, or 6.31 ps.**

## For P2

main memory access takes 70 ns.
This is 70/0.66 = 77.78 cycles.
Because we can't divide cycles, we must round up to 78 cycles.
Thus the Average Memory Access time is **1 + 0.06*78 = 5.68 cycles, or 6.11 ps.**

**c)** Assuming a base CPI of 1.0 without any memory stalls, what is the total CPI for P1 and P2? Which processor is faster? (When we say a "base CPI of 1.0", we mean that instructions complete in one cycle, unless either the instruction access or the data access causes a cache miss.) **[3 Marks]**

|  |  |  |
|---|---|---|
| P1 | 12.64 CPI | 8.34ns per inst |
| P2 | 7.36 CPI | 6.63ns per inst |

## For P1

Every instruction requires at least one cycle. In addition, 8% of all instructions miss in the instruction cache and incur a 107-cycle delay. Furthermore, 36% of the instructions are data accesses. 8% of these 36% are cache misses, which adds an additional 107 cycles.

1 + .08*107 + .36*.08*107 = 12.64

With a clock cycle of 0.66 ps, each instruction requires 8.34 ns.

**For P2**

**Using the same logic,** we can see that P2 has a CPI of 7.36 and an average of only 6.63 ns/instruction.

Now consider the addition of an L2 cache to P1. Use the L1 cache capacities and hit times from the previous table when solving these problems. The L2 miss rate indicated is its local miss rate.

| L2 size | L2 miss rate | L2 hit time |
|---|---|---|
| 1 MiB | 95% | 5.62 nsec |

**d)** What is the AMAT for P1 with the addition of an L2 cache? Is the AMAT better or worse with the L2 cache? **[3 Marks]**

An L2 access requires nine cycles (5.62/0.66 rounded up to the next integer).
All memory accesses require at least one cycle. 8% of memory accesses miss in the L1 cache and make an L2 access, which takes nine cycles. 95% of all L2 access are misses and require a 107 cycle memory lookup.

***1 + .08[9 + 0.95*107] = 9.85 cycles (It's worse)***

**e)** Assuming a base CPI of 1.0 without any memory stalls, what is the total CPI for P1 with the addition of an L2 cache? **[3 Marks]**

Notice that we can compute the answer to 5.6.3 as follows: AMAT + %memory * (AMAT-1).
Using this formula, we see that the CPI for P1 with an L2 cache is

***9.85 * 0.36*8.85 = 13.04***

**f)** What would the L2 miss rate be for P1 with an L2 cache to be faster than P1 without one? **[3 Marks]**

Because the clock cycle time and percentage of memory instructions is the same for both versions of P1, it is sufficient to focus on AMAT. We want
AMAT with L2 < AMAT with L1 only
***1 + 0.08[9 + m*107] < 9.56***
This happens when m< .916

**g)** What would the L2 miss rate be for P1 with an L2 cache to be faster than P2 without one? **[3 Marks]**

We want P1's average time per instruction to be less than 6.63 ns.
This means that we want

***$(CPI\_P1 * 0.66) < 6.63$.*** Thus, we need *$CPI\_P1 < 10.05$*
***$CPI\_P1 = AMAT\_P1 + 0.36(AMAT\_P1 - 1)$***
Thus, we want

***$AMAT\_P1 + 0.36(AMAT\_P1 - 1) < 10.05$***
This happens when *$AMAT\_P1 < 7.65$.*
Finally, we solve for

***$1 + 0.08[9 + m*107] < 7.65$***
and find that ***$m < 0.693$***

**This miss rate can be at most 69.3%.**

**Question 9 [15 Marks]**

In this exercise, we will examine how replacement schemes affect miss rates. Assume a three-way set associative cache with four one-word blocks. Consider the following word address sequence: 0, 1, 2, 3, 4, 2, 3, 4, 5, 6, 7, 0, 1, 2, 3, 4, 5, 6, 7, 0.

a) **[05 Marks]** Assuming an LRU replacement scheme, which accesses are hits?

| **Word Address** | **Cache Index** | **Hit/ Miss** | **Cache Index** | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | **Set 0** | | **Set 1** | | **Set 2** | |
| | | | **Block 0** | **Block 1** | **Block 0** | **Block 1** | **Block 0** | **Block 1** |
| 0 | 0 | M | **MEM[0]** | | | | | |
| 1 | 1 | M | MEM[0] | | **MEM[1]** | | | |
| 2 | 2 | M | MEM[0] | | MEM[1] | | **MEM[2]** | |
| 3 | 0 | M | MEM[0] | **MEM[3]** | MEM[1] | | MEM[2] | |
| 4 | 1 | M | MEM[0] | MEM[3] | MEM[1] | **MEM[4]** | MEM[2] | |
| 2 | 2 | ***H*** | MEM[0] | MEM[3] | MEM[1] | MEM[4] | **MEM[2]** | |
| 3 | 0 | ***H*** | MEM[0] | **MEM[3]** | MEM[1] | MEM[4] | MEM[2] | |
| 4 | 1 | ***H*** | MEM[0] | MEM[3] | **MEM[1]** | MEM[4] | MEM[2] | |
| 5 | 2 | M | MEM[0] | MEM[3] | MEM[1] | MEM[4] | MEM[2] | **MEM[5]** |
| 6 | 0 | M | **MEM[6]** | MEM[3] | MEM[1] | MEM[4] | MEM[2] | MEM[5] |
| 7 | 1 | M | MEM[6] | MEM[3] | **MEM[7]** | MEM[4] | MEM[2] | MEM[5] |
| 0 | 0 | M | MEM[6] | **MEM[0]** | MEM[7] | MEM[4] | MEM[2] | MEM[5] |
| 1 | 1 | M | MEM[6] | MEM[0] | MEM[7] | **MEM[1]** | MEM[2] | MEM[5] |
| 2 | 2 | ***H*** | MEM[6] | MEM[0] | MEM[7] | MEM[1] | **MEM[2]** | MEM[5] |
| 3 | 0 | M | **MEM[3]** | MEM[0] | MEM[7] | MEM[1] | MEM[2] | MEM[5] |
| 4 | 1 | M | MEM[3] | MEM[0] | **MEM[4]** | MEM[1] | MEM[2] | MEM[5] |
| 5 | 2 | ***H*** | MEM[3] | MEM[0] | MEM[4] | MEM[1] | MEM[2] | **MEM[5]** |
| 6 | 0 | M | MEM[3] | **MEM[6]** | MEM[4] | MEM[1] | MEM[2] | MEM[5] |
| 7 | 1 | M | MEM[3] | MEM[6] | MEM[4] | **MEM[7]** | MEM[2] | MEM[5] |
| 0 | 0 | M | **MEM[0]** | MEM[6] | MEM[4] | MEM[7] | MEM[2] | MEM[5] |

**Hit Rate = 5 / 20 = 25%**
**Miss Rate = 15/ 20 = 75 %**

| Block Address | **Cache Set** |
|---|---|
| 0 | (0 modulo 3) = 0 |
| 1 | (1 modulo 3) = 1 |
| 2 | (2 modulo 3) = 2 |
| 3 | (3 modulo 3) = 0 |
| 4 | (4 modulo 3) = 1 |
| 5 | (5 modulo 3) = 2 |
| 6 | (6 modulo 3) = 0 |
| 7 | (7 modulo 3) = 1 |

**b)** Most Recently Used (MRU) is a cache replacement scheme which removes the most recently used items first. A MRU scheme is good in situations in which the older an item is, the more likely it is to be accessed. Assuming an MRU (most recently used) replacement scheme, which accesses are hits? **[05 Marks]**

| **Word Address** | **Cache Index** | **Hit/ Miss** | **Cache Index** | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | **Set 0** | | **Set 1** | | **Set 2** | |
| | | | **Block 0** | **Block 1** | **Block 0** | **Block 1** | **Block 0** | **Block 1** |
| 0 | 0 | M | **MEM[0]** | | | | | |
| 1 | 1 | M | MEM[0] | | **MEM[1]** | | | |
| 2 | 2 | M | MEM[0] | | MEM[1] | | **MEM[2]** | |
| 3 | 0 | M | MEM[0] | **MEM[3]** | MEM[1] | | MEM[2] | |
| 4 | 1 | M | MEM[0] | MEM[3] | MEM[1] | **MEM[4]** | MEM[2] | |
| 2 | 2 | ***H*** | MEM[0] | MEM[3] | MEM[1] | MEM[4] | **MEM[2]** | |
| 3 | 0 | ***H*** | MEM[0] | **MEM[3]** | MEM[1] | MEM[4] | MEM[2] | |
| 4 | 1 | ***H*** | MEM[0] | MEM[3] | MEM[1] | **MEM[4]** | MEM[2] | |
| 5 | 2 | M | MEM[0] | MEM[3] | MEM[1] | MEM[4] | MEM[2] | **MEM[5]** |
| 6 | 0 | M | MEM[0] | **MEM[6]** | MEM[1] | MEM[4] | MEM[2] | MEM[5] |
| 7 | 1 | M | MEM[0] | MEM[6] | MEM[1] | **MEM[7]** | MEM[2] | MEM[5] |
| 0 | 0 | ***H*** | **MEM[0]** | MEM[6] | MEM[1] | MEM[7] | MEM[2] | MEM[5] |
| 1 | 1 | ***H*** | MEM[0] | MEM[6] | **MEM[1]** | MEM[7] | MEM[2] | MEM[5] |
| 2 | 2 | ***H*** | MEM[0] | MEM[6] | MEM[1] | MEM[7] | **MEM[2]** | MEM[5] |
| 3 | 0 | M | **MEM[3]** | MEM[6] | MEM[1] | MEM[7] | MEM[2] | MEM[5] |
| 4 | 1 | M | MEM[3] | MEM[6] | **MEM[1]** | MEM[7] | MEM[2] | MEM[5] |
| 5 | 2 | ***H*** | MEM[3] | MEM[6] | MEM[1] | MEM[7] | **MEM[2]** | MEM[5] |
| 6 | 0 | ***H*** | MEM[3] | **MEM[6]** | MEM[1] | MEM[7] | MEM[2] | MEM[5] |
| 7 | 1 | ***H*** | MEM[3] | MEM[6] | MEM[1] | **MEM[7]** | MEM[2] | MEM[5] |
| 0 | 0 | M | MEM[3] | **MEM[0]** | MEM[1] | MEM[7] | MEM[2] | MEM[5] |

**Hit Rate = 9 / 20 = 45%**
**Miss Rate = 11 / 20 = 55 %**

c) Describe an optimal replacement scheme for this sequence. Which accesses are hits using this policy? **[2.5 Marks]**

MRU is the most optimal.
It has 45% hits as compared to 25% hits in LRU.

d) Describe why it is difficult to implement a cache replacement scheme that is optimal for all address sequences. **[2.5 Marks]**

The best block to evict is the one that will cause the fewest misses in the future.
Unfortunately, a cache controller cannot know the future! Our best alternative is to make a good prediction.