National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department of Electrical Engineering

Faculty Member:____________________ **Dated: ____________________**

Course/Section:____________________ **Semester: __________________**

Computer Organization and Achitecture (EE321)

Lab # Cache Performance Analysis

Name	Reg. no.	Report Marks / 10	Viva Marks / 5	Total/15
Arifullah Jan	186943			
Bilal Khalid	128608			
Waqas Yaseen	196819			

Lab # Cache Performance Analysis

Objectives: The aim of this lab is to understand how different design parameters of Cache design process are selected to maximize cache performance.

Lab Instructions

- ✓ This lab activity comprises two parts, namely Lab tasks, and Post-Lab Viva session.
- ✓ Each group to upload completed lab report on LMS for grading.
- ✓ The students will start lab task and demonstrate each lab task separately for step-wise evaluation
- ✓ There are related questions in this activity give complete answers. Also provide complete code and results.

1. Introduction

The goal of an effective memory system is that the effective access time that the processor sees is very close to t_o, the access time of the cache. Most accesses that the processor makes to the cache are contained within this level. The achievement of this goal depends on many factors: the architecture of the processor, the behavioral properties of the programs being executed, and the size and organization of the cache. Caches work on the basis of the locality of program behavior. There are three principles involved:

1. **Spatial Locality -** Given an access to a particular location in memory, there is a high probability that other accesses will be made to either that or neighboring locations within the lifetime of the program.
2. **Temporal Locality -** This is complementary to spatial locality. Given a sequence of references to *n* locations, there is a high probability that references following this sequence will be made into the sequence. Elements of the sequence will again be referenced during the lifetime of the program.
3. **Sequentiality-** Given that a reference has been made to a particular location *s* it is likely that within the next several references a reference to the location of *s* + 1 will be made. Sequentiality is a restricted type of spatial locality and can be regarded as a subset of it.

Some common terms

Processor reference that are found in the cache are called cache hits. References not found in the cache are called cache misses. On a cache miss, the cache control mechanism must fetch the missing data from memory and place it in the cache. Usually the cache fetches a spatial locality called the *line* from memory. The physical word is the basic unit of access in the memory. The processor-cache interface can be characterized by a number of parameters. Those that directly affect processor performance include:

1. Access time for a reference found in the cache (a hit) - property of the cache size and organization.
2. Access time for a reference not found in the cache (a miss) - property of the memory organization.
3. Time to initially compute a real address given a virtual address (not-in-TLB-time) - property of the address translation facility, which, though strictly speaking, is not part of the cache, resembles the cache in most aspects and is discussed in this chapter.

Cache Organization

Within the cache, there are three basic types of organization:

1. Direct Mapped
2. Fully Associative
3. Set Associative

In **fully associative mapping**, when a request is made to the cahce, the requested address is compared in a directory against all entries in the directory. If the requested address is found (*a directory hit*), the corresponding location in the cache is fetched and returned to the processor; otherwise, a *miss* occurs.

In a **direct mapped** cache, lower order line address bits are used to access the directory. Since multiple line addresses map into the same location in the cache directory, the upper line address bits (tag bits) must be compared with the directory address to ensure a hit. If a comparison is not valid, the result is a cache miss, or simply a miss. The address given to the cache by the processor actually is subdivided into several pieces, each of which has a different role in accessing data.

The **set associative** cache operates in a fashion somewhat similar to the direct-mapped cache. Bits from the line address are used to address a cache directory. However, now there are multiple choices: two, four, or more complete line addresses may be present in the directory. Each of these line addresses corresponds to a location in a sub-cache. The collection of these sub-caches forms the total cache array. In a set associative cache, as in the direct-maped cache, all of these sub-arrays can be accessed simultaneously, together with the cache directory. If any of the entries in the cache directory match the reference address, and there is a hit, the particular sub-cache array is selected and outgated back to the processor.

2. **Cache Simulator** (address mapping)

Link for Simulator:
http://www.ecs.umass.edu/ece/koren/architecture/Cache/frame0.htm
http://www.ecs.umass.edu/ece/koren/architecture/Cache/frame1.html

About: The simulator emulates small sized caches based on a user-input cache model and displays the cache contents at the end of the simulation cycle based on an input sequence which is entered by the user, or randomly generated if so selected. Cache content is treated as one word per block, and the range of randomly generated addresses is the physical address space which is set to span 4 times the cache size in blocks. Output includes Cache content display, input query sequence event trace, simulation statistics, and also Multi-Tasking simulation if selected. Hexadecimal format, user-input or random query sequences, sequence repeat cycle control (helpful for viewing cache contents a few random queries at a time), show tags, and sequence length limiting are all optional. Cache size, set associativity, and replacement policy are menu selectable. You may simply click the 'SHOW CACHE' button on startup default settings after entering a query sequence, or selecting "Use Random", to begin.

HELP reference table of Control Elements	
Cache Size	Select **Cache Size** in blocks from menu. Cache size cannot be set less than #Sets in cache (associativity level). BlockSize is fixed at one word.
# Sets in cache	Select **Associativity** level from menu, It is the number of sets in the cache. The Associativity level must be less than or equal to the Cache Size#
if (#Sets<SizeBlocks)	- (#Sets) = N-way Set Associative cache
if (#Sets=SizeBlocks)	- Direct Mapped cache
if (#Sets=1;)	- Fully Associative cache
Replacement Policy (for non-compulsory cache misses)	
Random	Select destination block in set for **replacement at random**
Least **R**ecently **U**sed	:Select destination block in set for **replacement by age since most**

	recent access (replace oldest read/write)
First **I**n - **F**irst **O**ut	Select destination block in set for **replacement by age since most recent update** (replace oldest write)
Enter Query Sequence	If not Use Random, then manually **enter your query sequence here**. Your values should be entered space-delimited. Input will be interpreted as hexadecimal if **Use Hex** is selected.
Use Random	Select checkbox **Random generation of sequence** of #(Limit Query) addresses as cache query. Disregards any entries in input-box
Use Hex	Select checkbox to change input/output format to **hexadecimal.** Note that the # Cycles and Length fields are always decimal inputs, otherwise **Use Hex** causes all Input / Output to be interpreted / given in hexadecimal format.
Limit Query	Select checkbox to limit length of random sequence or user-input sequence to the length specified by the value in the Limit Query Length field.
Limit Query # Length	Sets length of random query sequence, or **limits length** of user - input query sequence in input window. The Limit Query checkbox must be selected also.
Show Cache	**View cache** as configured, shows effect of query sequence on cache, outputs query event sequence with resulting actions with the cache, with analysis.
Set Repeat	**To repeat an input query sequence** through N iterations, Set Repeat N Cycles. Will show cache content output at end of each cycle.
(Set Repeat) **# Cycles**	**# Cycles** sets the number of times the cache will run through Query Sequences set by **Limit Query and Length**, where **Use Random** changes the query sequence each **cycle**, or **if not Use Random** then each cycle repeats the user input sequence from the Enter Query Sequence input textfield area.
Use Multi-Tasking	Select **Multi-Tasking** to simulate one cache as it is switched between two processes, one for each cycle in **# Cycles**. If you Select Multi Tasking, it is best used with **Use Random** selected, with **Limit Query #Length** set.
Show Tags	Select **show tags** to show the tags in the cache contents output table. Tags are shown in red binary, with index field omitted for clarity. No offset field shown either because this simulator uses one word per block

Tips and Tricks

- **First time users** should start by simply clicking on the "SHOW CACHE" button after entering a query sequence, or selecting "Use Random", and proceed from there.
- **Single stepping** through a query sequence can be done by entering entire sequence in the "**Enter a query sequence**" input field. Next select **Limit Query** and set **Limit Query Length** to 1. Next click on the SHOW CACHE button, The output window will contain results for the first Query. (*) Change **Limit Query Length** to the next value (add 1 to length) and Click on **SHOW CACHE**. Repeat from (*) until done.
- **Sequence-Stepping**, with **Multi-Tasking not** selected, and #Cycles set to greater than 1, with **Use Random** selected (preferred), the simulator will show cache contents and results for each Cycle, with each cycle having **#Length** entries. If Random not selected, each cycle repeats on the Entered Query Sequence. (up to #Length).
- **Multi - Tasking** Simulates a multiprocessing context in which the processor switches between two tasks A and B, and the Cache Content, output each cycle, reflects the most recent query sequence on the cache. By noticing that **Task A** and Task B are represented by **color** andgrayscale respectively, you can visualize the two tasks overwriting each other's cache entries with each cycle. Cache Sequence Results are cumulative with each cycle of Multi-Tasking, and are representative of a multi-tasking environment in which two tasks share the same address and data space. For your first try; With Multi-Tasking selected, you should also select Set Repeat to some small number, Use Random and Limit Query should both be selected, and set the limit # to some value, preferably about or under the Cache Size in Blocks.

Note: click on help button for more help

Lab Task 1

1) Consider the input sequence of length 120 given below.

220 66 131 35 94 172 126 217 73 176 250 84 114 187 201 116 4 102 84 22 44 87 114 82 144 28 211 131 25 192 12 134 176 157 197 211 223 67 199 203 30 154 51 123 140 172 218 249 27 91 5 51 202 59 196 240 238 71 100 217 49 231 226 12 118 233 204 222 220 31 220 66 173 5 6 94 62 126 124 250 21 81 74 116 233 9 167 62 20 4 161 35 152 102 79 73 86 84 182 22 92 44 66 159 187 240 167 100 169 201 174 114 232 82 187 87 175 131 156 301

	2- way			4 – way			8 - way		
	LRU	FIFO	RAND	LRU	FIFO	RAND	LRU	FIFO	RAND
8 blocks									
16 blocks									
32 blocks									
64 blocks									

(a) Analyze the effectiveness of different block replacement techniques by listing down the miss rate in each case.

(b) What other block replacement technique can be used and is proved to be the ideal? Explain.

2) In a N-way set-associative cache, blocks are mapped to different sets when N changes. Also, for a particular sequence, the number of compulsory and conflict misses change with the cache type. Consider the following sequence 4 0 9 7 8 11 7 5 2 1 12 6 8.

(a) List the compulsory and conflict misses for different replacement techniques for the caches below.

MISSES	**LRU**		**FIFO**		**RANDOM**	
	Comp	***Conflict***	***Comp***	***Conflict***	***Comp***	***Conflict***
4blocks 2sets						
8blocks 2sets						
16blocks 2sets						

(b) Define compulsory, capacity and conflict misses. Explain the difference between them.

(c) What is the best way to reduce conflict misses? Can it be used?

(d) List which set in the given cache will the following blocks be mapped

BLOCK	**CACHE**	**#SET**
0	***8block,2sets***	
9		
11		
4		
2	***8blocks,4sets***	
9		
10		

4		
7	***16blocks,2sets***	
1		
12		
3		

3. **Cache Simulator** (Time Analysis)

Simulator Link:http://www.ecs.umass.edu/ece/koren/architecture/Cache/frame2.htm

About: This Cache time analyzer demonstrates Average Memory Access Time analysis for the cache parameters you specify. Miss rates are provided for your cache specifications based on data from the course textbook derived from actual tests using the SPEC benchmark and LRU replacement algorithm. The analysis is performed for the general case, which means that the analysis does not accommodate hardware optimizations such as second-level cache, victim buffer, or prefetching, for example. **Optional graphing** is available by selecting a graphing parameter option from the plot menu at the bottom of the control panel.

Replacement Policy Assumptions: It is assumed, in the case of the Write Through policy, that on a **write**, the memory write (in this case a word write) begins in parallel with the cache write, thus the time-cost incurred is equal to the greater of Hit Time and Memory Word-Write Time. Since in the real world the Memory Word-Write Time is always greater than the Cache Hit Time, the analysis uses your setting for **Clocks.Mem.Write** regardless of what you have set for hit time This applies to both write hits and write misses for Write-Through only. All other equations for all other policies and instances are clear and need no such explanation.

HELP reference table of Control Elements	
Cache Size	Select **Cache Size** in KBytes from menu. Used to determine miss rate.
Associativity	Select **Associativity** level from menu, It is the number of sets in the cache. Used to determine miss rate.
Block Size	Select **Block Size** from the menu. It specifies the size of each block in words. Used to determine miss rate.
Write Policy	Select **Write Policy**. Choose Write Back or Write Through, and No Write Allocate or Allocate on Miss
% Writes	Enter the **%** of memory accesses that are **Writes.** Valid inputs are integers 0-99
% Dirty Data	Enter the **%** of **Dirty Data** representing the chance a block in cache

	has been modified but not written back to memory. This number is not used if you select **Write-Through.** Valid inputs are integers 0-99
Clocks Miss Penalty	Enter the **Miss Penalty** in clock cycles. Valid inputs are integers 1-99
Clocks Hit Time	Enter the **Hit Time in clock cycles.** Valid inputs are integers 1-9
Clocks Mem. Write	Enter the Memory Write time in clock cycles for a single write (word write) to memory.
ANALYZE	Click on this button to submit your cache configuration for analysis.
Plot Options Menu	Select option to graph parameter vs. average memory access time (Tavg). Displays graph below analysis table.

Tips and Tricks

- **First time users** should start by simply clicking on the "**ANALYZE**" button to view results, and proceed from there.
- **Graphs**: If you select a plot option. note that you can move the cursor over the plotted points to see exact values displayed.
- **Graphs**: **If you re-size, reload.** If you selected a plot option and clicked "ANALYZE" , your plot will be displayed (below analysis table). If you re-size your browser window at this point the plot output window will probably become distorted. If this happens, you may click the "ANALYZE" button again, or choose 'RELOAD' from your browser's menu, either one will reload the plot correctly into the current browser window frame-size.

Note: click on help button for more help

Lab Task 2

1) Consider a 4-way set associative cache of size 4KB and block size of 64bytes. Consider 25% writes, miss penalty of 40 cycles, hit time of 1 cycle and mem write time of 6 cycles. How can the following happen under the condition that the *memory miss penalty, % write* and *hit time* remain the same for both the cases? Reason your findings in a few sentences.

(a) Avg. Memory access for Write Back cache > Write through cache - No write allocate

(b) Avg. Memory access for Write Back cache > Write through cache - Write allocate

2) Assume that the % of dirty blocks in the cache is 99%, if the total number of blocks in the cache is very high, the write back based cache should have a high memory access time. Is this case true for both small and large cache sizes from this animation program? Fill the given table

and explain the situation for the 4 caches. The rest of the simulation term values are listed in the table.

Simulation terms	**Value**
Cache type	4-way set associative
% writes	25 %
Miss Penalty	40 cycle
Hit Time	1 cycle
Memory Write time	6 cycles

Cache size	**Block size**	**Hit rate**	**Memory access time**
1KB	256bytes		
	16bytes		
4KB	256bytes		
	16bytes		
32KB	256bytes		
	16bytes		
256KB	256bytes		
	16bytes		