CS224 – Spring 2019 - Lab #6 (ver. 1, 29 April 2019, 1:03 am)

Examining the Effect of Cache Parameters and Program Factors on Cache Hit Rate

Dates: Section 1, Monday, 6 May, 13:40-17:30
Section 3, Tuesday, 7 May, 13:40-17:30
Section 2, Wednesday, 8 May, 13:40-17:30
Section 4, Thursday, 9 May, 13:40-17:30
Section 5, Friday, 10 May, 8:40-12:30
Section 6, Friday, 10 May, 13:40-17:30
Lab Location: EA-Z04

Purpose: Studying the effect of various cache design parameters. The first part includes problem solving and writing a program. The second part, the lab part, involves execution of the program (with possible extensions etc. if suggested by your TA) and preparing a report.

In your solutions and report make sure that you have proper tables, page numbering and understandable explanation with good writing. All tables must have a subtitle and table number. In tables columns must have meaning names/headers. Try to do everything before coming to the lab but make sure that you demonstrate your work to your TA.

Summary
Part 1 (50 points): Preliminary Report/Preliminary Design Report: Involves problem solving related to cache memory design and a program written for cache testing.
Part 2 (50 points): Experiments with caching and experiment report.

DUE DATE OF PART 1: SAME FOR ALL SECTIONS:

1. Please bring and drop your printed preliminary work into the box(es) provided in front of TA office room number EA-407 by 13:30 on Monday May 6.
2. Please **upload your programs of Part 1 (PRELIMINARY WORK)** to the Unilica by 13:30 on Monday May 6. Use filename **StudentID_FirstName_LastName_SecNo_PRELIM_LabNo.txt** Upload only the programs. Only a NOTEPAD FILE (txt file) is accepted. Your paper submission for preliminary work must match MOSS submission.

   We use MOSS testing only for code submissions. So, you are not supposed to submit the whole preliminary report online. You will ONLY submit the code online. This time lab part, Part 2, involves two submissions: code and the lab experiments report you showed to your TA in the lab.

No late submission will be accepted. No late submission will be accepted. No late submission will be accepted. No late submission will be accepted. No late submission will be accepted.

DUE TIME OF PART 2—DIFFERENT FOR EACH SECTION:
You have to demonstrate your lab work to the TA for grade by **12:15** in the morning lab and by **17:15** in the afternoon lab. Your TAs may give further instructions on this. If you wait idly and show your work last minute, 20 points may be taken off from your grade.

At the conclusion of the demo for getting your grade, you will **upload your lab program work** (only text) to the Unilica Assignment, for similarity testing by MOSS. See Part 3 below for details.

You will also submit **lab experiment report separately (pdf is prefered: follow your TAs suggestions). Two separate files no confusion: No excuses will be accepted.**

For further possible instructions etc. please follow the possible updates on the Unilica course web page. In the lab your TAs may also give further instructions.

Part 1. Preliminary Work / Preliminary Design Report (50 points)

You have to provide a neat presentation prepared by Word or a word processor with similar output quality. Handwritten answers will not be accepted. At the top of the paper on left provide the following information and staple all papers. Please make sure that this info is there for proper grading of your work, otherwise some points will be taken off.

CS224
Section No.: ...
Spring 2019
Lab No.:
Your Full Name/Bilkent ID:

Solve the problems of this part and print and leave your report in the box provided in the lab as described above. Number the pages, staple your submission and use a word processor for its preparation. Only the program will be uploaded as described above.

1. (5 points: With 3 or more errors you get 0 points. Otherwise full point.) Fill in the empty cells of the following table. Assume that main memory size is 0.5 GB. **Index Size**: No. of bits needed to express the set number in an address, **Block Offset**: No. of bits needed to indicate the word offset in a block, **Byte Offset**: No. of bits needed to indicate the byte offset in a word. **Block Replacement Policy Needed**: Indicate if a block replacement policy such as FIFO, LRU, LFU (Least Frequently Used) etc. is needed (yes) or not (no). If some combinations are not possible mark them.

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[1]	Byte Offset Size in bits[2]	Block Replacement Policy Needed (Yes/No)
1	32	1	32 bits	4						
2	32	2	32 bits	4						
3	32	4	32 bits	8						
4	32	Full	32 bits	8						
9	256	1	16 bits	4						
10	256	2	16 bits	4						
11	256	4	8 bits	16						
12	256	Full	8 bits	16						

[1] **Word Block Offset Size in bits:** Log_2(No. of words in a block)

[2] **Byte Offset Size in bits:** Log_2(No. of bytes in a word)

2. (5 points: With 3 or more errors you get 0 points. Otherwise full point.) Consider the following MIPS code segment. (Remember MIPS memory size is 4 GB.) Cache capacity is 8 words, Block size: 2 words, N= 1.

```
        addi    $t0, $0, 5
loop:   beq     $t0, $0, done
```

```
        lw      $t1, 0x24($0)
        lw      $t2, 0x2C($0)
        lw      $t3, 0x28($0)
        addi    $t0, $t0, -1
        j       loop
done:
```

a. In the following table indicate the type of miss, if any: Compulsory, Conflict, Capacity.

Instruction	Iteration No.				
	1	**2**	**3**	**4**	**5**
lw $t1, 0x24($0)					
lw $t2, 0x2C($0)					
lw $t3, 0x28($0)					

b. What is the total cache memory size in number of bits? Include the V bit your calculations. Show the details of your calculation.

c. State the number of AND and OR gates, EQUALITY COMPARATORs and MULTIPLEXERs needed to implement the cache memory. No drawing is needed.

3. **(5 points: With 3 or more errors you get 0 points. Otherwise full point.)** Consider the above MIPS code segment. The cache capacity is 2 words , block size is 1 word. There is only 1 set. The block replacement policy is LRU.

a. In the following table indicate the type of miss, if any: Compulsory, Conflict, Capacity.

Instruction	Iteration No.				
	1	**2**	**3**	**4**	**5**
lw $t1, 0x24($0)					
lw $t2, 0x2C($0)					
lw $t3, 0x28($0)					

b. How many bits are needed for the implementation of LRU policy? What is the total cache memory size in number of bits? Include the V bit and the bit(s) used for LRU in your calculations. Show the details of your calculation.

c. State the number of AND and OR gates, EQUALITY COMPARATORs and MULTIPLEXERs needed to implement the cache memory. No drawing is needed.

4. **(5 points)** Consider a three level memory: L1 and L2 are for cache memory and the third level is for the main memory. Access time for L1 is 1 clock cycle, the access time for L2 is 4 times as much as L1 and main memory access time is 20 times as much as L2. The miss rate for L1 is 5% and the miss rate for L2 is 25%. What is the effective clock cycle for memory access (AMAT in number of clock cycles)?

With 2 GHz clock rate how much time is needed for a program with 10^{12} instructions to execute using the cache conditions defined above?

Show your work briefly.

5. **(30 points)** Write a program to find the summation of the elements of a square matrix, trace of the matrix, and trace like summation using the other diagonal (shaped like: / from right upper corner to left bottom corner) of the matrix. Provide a user interface for user interaction to demonstrate that your program is working properly. See below for the description of the user interface menu. Assume that in the main memory matrix elements are placed row by row. For example a 3 by 3 (N= 3) matrix would have the following values

the following values.

1	2	3
4	5	6
7	8	9

The row by row placement means that you will have the values of the above 3 x 3 matrix are stored as follows in the memory.

Matrix Index (Row No., Col. No.)	(1, 1)	(1, 2)	(1, 3)	(2, 1)	(2, 2)	(2, 3)	(3, 1)	(3, 2)	(3, 3)
Displacement With respect the beginning of the array containing the matrix	0	4	8	12	16	20	24	28	32
Value stored	1	2	3	4	5	6	7	8	9

In this configuration accessing the matrix element (i, j) simply involves computation of its displacement from the beginning of the array that stores the matrix elements. For example, the displacement of the matrix element with the index (i, j) with respect to the beginning of the array is (i - 1) x N x 4 + (j - 1) x 4, for a matrix of size N x N.

In your program for the matrix allocate an array with proper size using syscall code 9.

Your user interface must provide at least the following menu options:
1. Ask the user the matrix size in terms of its dimensions (N),
and then ask the user enter matrix elements row by row.
2. Ask the user the matrix size in terms of its dimensions (N),
and initialize the matrix entries with consecutive values (1, 2, 3 ...),
3. Display a desired element of the matrix by specifying its row and column number,
4. Display entire matrix row by row,
5. Obtain trace of the matrix and display,
6. Obtain trace like summation using the other diagonal of the matrix and display,
7. Obtain sum of matrix elements by row-major (row by row) summation,
8. Obtain sum of matrix elements by column-major (column by column) summation.

2. [50 pts] Experiments

2.A. [10 pts] Experiments with Trace & Trace-Like Summation

Run your program with matrices of different sizes as asked by your TA for
trace calculation
and trace-like calculation
and show that trace and trace-like menu options work properly.

2.B. [40 pts] Experiments with Data Cache Parameters Using Summations

Run your program for summation of matrix elements as suggested below with two reasonably large different matrix sizes that would provide meaningful observations. Modify your original program if needed. For large matrix initialization use the second menu option and initialize matrix entries with consecutive values.

Report for Matrix Size 1: 20 Points
Report for Matrix Size 2: 20 Points

Make sure that you have a easy to follow presentation by numbering tables and by providing proper table titles. Make sure that your tables have proper headings etc.

Make sure that you find the summation of matrix elements by performing row-major and column-major

Make sure that you find the summation of matrix elements by performing row-major and column-major addition.

a) **Direct Mapped Caches**: For the matrix sizes you have chosen, conduct tests with various cache sizes and block sizes, to determine the hit rate, miss rate and number of misses. Use at least 5 different cache sizes and 5 different block sizes (make sure your values are reasonable) in order to obtain curves like those of Figure 8.18 (see below) in the textbook. The figure shows the cache memory size in the x axis. Make a 5 x 5 table with your values, with miss rate and # of misses as the data at each row-column location. Make a graph of miss rate versus block size, parameterized by cache size, like Figure 8.18.

b) **Fully Associative Caches**: Pick 3 of your parameter points obtained in part for column-major addition a), one with good hit rate, one with medium hit rate, and one with poor hit rate. For these 3 results, there were 3 configuration pairs of cache size and block size that resulted in the data. Take the same 3 configuration pairs, but this time run the simulation with a fully associate cache, using LRU replacement policy. Compare the results obtained: the Direct Mapped good result versus the Fully Associative good result, the Direct Mapped medium result versus the Fully Associative medium result, and the Direct Mapped poor result versus the Fully Associative poor result. How much difference did the change to fully associative architecture make? Now change the replacement policy to Random replacement, and run the 3 tests again (using the same 3 configuration pairs). Does replacement policy make a significant difference? Record these 9 values in a new table, with 3 lines: for Direct Mapped, for Fully Associative-LRU and for Fully Associative-Random.

c) **N-way Set Associative Caches**: to save on hardware costs, fully set-associative caches are rarely used. Instead, most of the benefit can be obtained with an N-way set associative cache. Pick the medium hit rate configuration that you found in a) and used again in b), and change the architecture to N-way set associative. For several different set sizes (at least 4) and LRU replacement policy, run the program and record the hit rate, miss rate and number of misses. What set size gives the best result? How much improvement is gained as N (the number of blocks in a set) increases each step? Now repeat the tests, but for the good hit rate configuration from a) and b). Record these data and answer the same question again. Finally, repeat for the poor hit rate configuration.

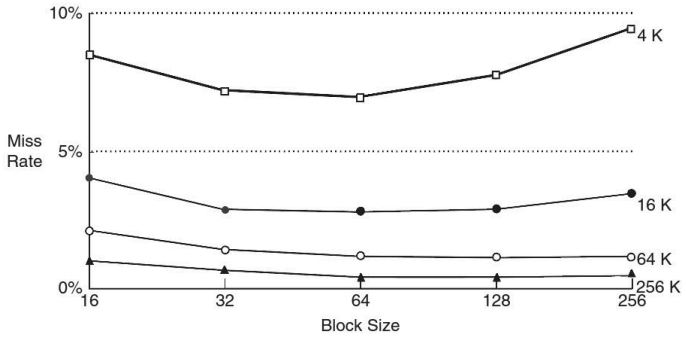

Figure 8.18 **Miss rate versus block size and cache size on SPEC92 benchmark**
Adapted with permission from Hennessy and Patterson, *Computer Architecture: A Quantitative Approach*, 5th ed., Morgan Kaufmann, 2012.

Oral Interview with TA and Submission of your Data

Get ready for the interview with your TA, by gathering and analyzing your data, having it ready to submit in a clean understandable format. Be sure that you understand what you have done, and can interpret your data to the TA. Then call him over, and answer the questions he asks you.

Part 3. Submissions

A. Submit your code for MOSS similarity testing

Combine all the new and modified codes into a file called **StudentID_FirstName_LastName_SecNo_LabNo_LAB.txt**. You will then upload this file to the Unilica > CS224 > Assignment for your section. While the TA or Tutor is watching, you will upload this file. *Even if you didn't completely finish, you must submit the* **StudentID_FirstName_LastName_SecNo_LabNo_LAB.txt** *file to the Unilica Assignment for similarity checking*. If you don't submit your code, your grade for the lab will be 0 (see Lab Policies section, below NOTES.) Your codes will be compared against all the other codes in the class, by the MOSS program, to determine how similar it is (as an indication of plagiarism). So be sure that you only

submit code that you actually wrote yourself ! All students must upload their code to Unilica > Assignment while the TA or Tutor is watching, and before the deadline. NOTE: you are allowed to upload only ONE file to Unilica, so be sure it contains exactly and only the codes required.

B. Submit your experimental report for grading

Submit your experimental report separately as instructed by your TA. For this submission pdf is preferred, please follow your TAs suggestions on this. Name your file as **StudentID_FirstName_LastName_SecNo_LabNo_REPORT.pdf**
Note that this file will be submitted into a separate Unilica folder.

Part 4. Cleanup

1. After saving any files that you might want to have in the future to your own storage device, erase all the files you created from the computer in the lab.
2. When applicable put back all the hardware, boards, wires, tools, etc where they came from.
3. Clean up your lab desk, to leave it completely clean and ready for the next group who will come.

Part 5. Lab Policies

1. You can do the lab only in your section. Missing your section time and doing in another day is not allowed.

2. Students will earn their own individual lab grade. The questions asked by the TA will have an effect on your individual lab score.

3. Lab score will be reduced to 0 if the code is not submitted for similarity testing, or if it is plagiarized. MOSS-testing will be done, to determine similarity rates. Trivial changes to code will not hide plagiarism from MOSS—the algorithm is quite sophisticated and powerful. Please also note that obviously you should not use any program available on the web, or in a book, etc. since MOSS will find it. The use of the ideas we discussed in the classroom is not a problem.

4. You must be in lab, working on the lab, from the time lab starts until your work is finished and you leave.

5. No cell phone usage during lab.

6. Internet usage is permitted only to lab-related technical sites.

7. For labs that involve hardware for design you will always use the same board provided to you by the lab engineer.