CS224 – Spring 2023 - Lab #6 (Version 1, May 12, 7:46 pm)

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

Dates (TAs) - (Tutor(s)):
Section 1: Mon, 22 May, 8:30-12:20 in EA-Z04 (Deniz, Ece) - (Berkan)
Section 2: Wed, 24 May, 13:30-17:20 in EA-Z04 (Sepehr, Pouya) - (Berkan, Ege)
Section 3: Tue, 23 May, , 13:30-17:20 in EA-Z04 (Onur, Utku) - (Talay)
Section 4: Fri, 26 May, Fri 8:30-12:20 in EA-Z04 (Sanaz, Soheil) - (Eren)
Section 5: Wed, 24 May Feb, 8:30-12:20 in EA-Z04 (Navid, Onur) - (Yarkın)
Section 6: Fri, 26 May, 13:30-17:20 in EA-Z04 (Sanaz, Soheil) - (Yarkın)

TA Name (email address: @bilkent.edu.tr): Section (hours)
Ece Kunduracıoğlu (e.kunduracioglu@): Section 1 (Mon 08:30-10:30)
Deniz Uzel (deniz.uzel@): : Section 1 (Mon 08:30-10:30)
Navid Ghamari (navid.ghamari@): Section 2 (Wed 13:30-15:30)
Onur Yıldırım (o.yildirim@): Section 3 (Tue 13:30-17:30)
Pouya Ghahramanian (ghahramanian@): Section 2 (Wed 13:30-17:30)
Sanaz Gheibuni (sanaz.gheibuni@): Section 4 (Fri 08:30-12:30), Section 6 (Fri 13:30-15:30)
Sepehr Bakhshi (sepehr.bakhshi@): Section 2 (Wed 13:30-17:30)
Soheil Abadifard (soheil.abadifard@): Section 4 (Fri 08:30-12:30), Section 6 (Fri 13:30-15:30)
Utku Gülgeç (utku.gulgec@): Section 3 (Tue 13:30-17:30)

Tutor Name (email address: @ug.bilkent.edu.tr): Section (hours)
Berkan Şahin (berkan.sahin@): Section 1 (Mon 08:30-10:20) & Section 2 (Wed 13:30-15:20)
Hasan Ege Tunç (hasan.tunc@): Section 2 (Wed 13:30-17:20)
Atak Talay Yücel (talay.yucel@): Section 3 (Tue 13:30-17:20)
Mehmet Eren Balasar (eren.balasar@): Section 4 (Fri 08:30-12:20)
Hasan Yarkın Kurt (yarkin.kurt@): Section 5 (Wed 08:30-10:20) & Section 6 (Fri 13:30-15:20

Purpose: In this lab you will study 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, numbering etc. All tables must have subtitle and table number furthermore columns must have column names, etc. In your report make sure that you have a nice presentation. Make sure that you number the pages. Try to do everything before coming to the lab but make sure that you demonstrate your work to your TA.

Summary

Preliminary Work (50 points)

Involves problem solving related to cache memory design and a program written for cache testing.

Lab Work (50 points)

Experiments with caching and experiment report.

Important Notes for All Labs

1. Try to complete the lab part at home before coming to the lab. Make sure that you show your work to your TA and answer his questions to show that you know what you are doing before uploading your lab work and follow the instructions of your TAs. In all labs if you are not told you may assume that inputs are corrects. For all works when needed please provide a simple user interface for inputs and outputs.
2. You are obliged to read this document word by word and are responsible for the mistakes you make by not following the rules. Your programs should be reasonably documented (purpose etc.) and must have a neat presentation in terms of variable names, subprogram names. indentation, comments, blank lines etc.
3. **If we suspect that there is cheating we will send the work with the names of the students to the university disciplinary committee.**

DUE DATE/TIME OF PART 1 --SAME FOR ALL SECTIONS (MONDAY MAY 22nd, 8:30 AM)

No late submission will be accepted.

a. Please upload your problem solutions and programs of preliminary work to Moodle by 8:30 on Monday May 22 for similarity testing by MOSS. We plan to use MOSS both for problems solutions and program.

b. **Submission of your work**

Problems: Use the file name

c. **StudentID_FirstName_LastName_SecNo_PRELIMproblem_LabNo.pdf** [A pdf file as its extension suggests, which contains your solutions to the Preliminary Part]. Only a pdf file is accepted. Any other form of submission receives 0 (zero).

Code: For the program part use the filename **StudentID_FirstName_LastName_SecNo_PRELIMcode_LabNo.txt** [A NOTEPAD FILE as its extension suggests, which contains only the program part. Only a NOTEPAD FILE (txt file) is accepted. Any other form of submission receives 0 (zero).

d. Note that the Moodle submission closes sharp at 8:30 and no late submissions will be accepted. You can make resubmissions before the system closes, so do not wait for the last moment. Submit your work earlier and change your submitted work if necessary. Note that only the last submission will be graded.

e. Do not send your work by email attachment, they will not be processed. They have to be in the Moodle system to be processed.

f. At the beginning of your submission files include the following make sure that each of them is in a separate line:

Course No.: CS224
Lab No.
Section No.
Your Full Name
Bilkent ID.

DUE TIME OF PART 2—DIFFERENT FOR EACH SECTION:

a. You have to demonstrate your lab work to your TA for grading. Do this by **12:00** in the morning lab and by **17:00** in the afternoon lab. Your TAs may give further instructions on this and they may make changes. If you wait idly and show your work last minute, your work may not be graded. Make sure that you follow your TA's instructions.

b. At the conclusion of the demo for getting your grade, you will **upload your Lab Work** to the Moodle Assignment, for similarity testing by MOSS. See lab part submission details below.

c. Aim to finish all of your lab work before coming to the lab, but make sure that you upload your work after making sure that your work is analyzed by your TA and/or you are given the permission by your TA to upload.

d. At the beginning of your submission files include the following make sure that each of them is in a separate line: Course No.: CS224, Lab No., Section No., Your Full Name, and your Bilkent ID.

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.: ...
Lab No.:
Your Full Name/Bilkent ID:

Solve the problems of this part.

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 4GB. **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, Random 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	64	1	32 bits	4						
2	64	2	32 bits	4						
3	64	4	32 bits	8						
4	64	Full	32 bits	8						
9	128	1	16 bits	4						
10	128	2	16 bits	4						
11	128	4	16 bits	16						
12	128	Full	16 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. Cache capacity is 8 words, Block size: 2 words, N= 1.

```
        addi   $t0, $0, 5
loop:   beq    $t0, $0, done
        lw     $t1, 0x4($0)
        lw     $t2, 0xC($0)
        lw     $t3, 0x8($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, 0x4($0)					
lw $t2, 0xC($0)					
lw $t3, 0x8($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.

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, 0x4($0)					
lw $t2, 0xC($0)					
lw $t3, 0x8($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.

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 more than L1 and main memory access time is 10 times more than L2. The miss rate for L1 is 20% and the miss rate for L2 is 5%. What is the effective clock cycle for memory access (AMAT in number of clock cycles)?

With 4 GHz clock rate how much time is needed for a program with 10^{12} instructions to execute?

Show your work briefly.

5. **(30 points)** Write a program to find the summation of the elements of a square matrix. Provide a user interface for user interaction to demonstrate that your program is working properly. Assume that in the main memory matrix elements are placed column by column. Create an array for the matrix elements and initialize them column by column with consecutive values. For example, a 3 by 3 (N= 3) matrix would have the following values.

1	4	7
2	5	8
3	6	9

The column by column 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)	(2, 1)	(3, 1)	(1, 2)	(2, 2)	(3, 2)	(1, 3)	(2, 3)	(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 (j - 1) x N x 4 + (i - 1) x 4, for a matrix of size N x N.

Your user interface must provide at least the following functionalities,

1. Ask the user the matrix size in terms of its dimensions (N),
2. Allocate an array with proper size using syscall code 9,
3. Obtain summation of matrix elements row-major (row by row) summation,
4. Obtain summation of matrix elements column-major (column by column) summation,
5. Display desired elements of the matrix by specifying its row and column member

Part 2. Experiments with Data Cache Parameters (50 pts)

Run your program with two reasonably large different matrix sizes that would provide meaningful observations. Modify your original program if needed. For large matrix initialization you may use a loop rather than user interface.

Report for Matrix Size 1: 25 Points

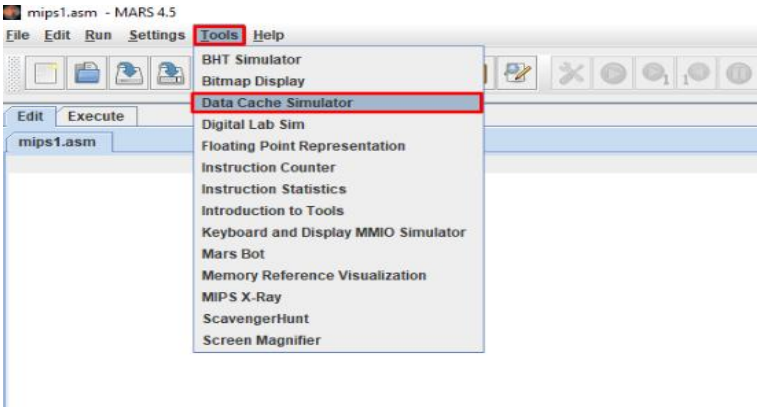
Report for Matrix Size 2: 25 Points

As described above make sure that you have an easy to follow presentation with numbered tables having proper heading etc.

Make sure that you find the summation of matrix elements by performing row-major and column-major addition. Note that the column-major addition is a simple array addition from the beginning to the end; however, the row-major addition is somewhat tricky.

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. Make a 5 x 5 table with your values, with miss rate and # of misses as the data at each row-column location. Make the graph of miss rate versus block size, parameterized by cache size, like Figure 8.18 both for row-major and column-major additions.

   Hint: You can reach the Cache Simulator from MARS/Tools/Data Cache Simulator as shown in the following image:



b) **Fully Associative Caches**: Pick 3 of your parameter points obtained in part for row-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 graph, with 3 lines: for Direct Mapped, for Fully Associative-LRU and for Fully Associative-Random. Note that this step is only for row-major addition so that by the end of the lab you'll have (2+1)x2 = 6 graphs.

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.

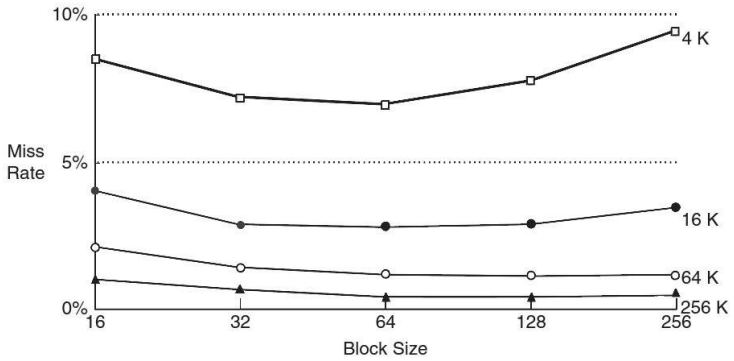


Figure 8.18 **Miss rate versus block size and cache size on SPEC92 benchmark**
Adapted with permission from Hennessy and Patterson, *Computer Architecture: A Quantitative Approach*, 5th ed., Morgan Kaufmann, 2012.

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/her over, and answer the questions he/she asks you.

Part 3. Submit Experiment Report and Your Code for MOSS similarity testing

Submit your MIPS codes for similarity testing to the Unilica > Assignment specific for your section.

As described above you have two files to upload.

Use filename **StudentID_FirstName_LastName_SecNo_Lab6_report.doc** [A DOC FILE as its extension suggests, which contains all the work done for the Lab Experiment Report Part].

For the program part Use filename **StudentID_FirstName_LastName_SecNo_Lab6_code.txt** [A NOTEPAD FILE as its extension suggests, which contains the Experiment Code Part]: this part may involve deviation, as suggested by your TA, from the preliminary work program you submitted. No matter what submit the program you used in the lab experiments.

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 the code you submit is code that you actually wrote yourself! All students must upload their code to Unilica > Assignment while the TA watches. Submissions made without the TA observing will be deleted, resulting in a lab score of 0. The same type of comparison is also planned for the reports.

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.

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.