CS224 - Fall 2022- Lab #6 (V1: Dec 5, 19:02)

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

Dates:
Section 1: Wed, 14 Dec, 13:30-17:20 in EA-Z04
Section 2: Thu, 15 Dec, 13:30-17:20 in EA-Z04
Section 3: Thu, 15 Dec, 8:30-12:20 in EA-Z04
Section 4: Fri, 16 Dec, 13:30-17:20 in EA-Z04

TA name (x No of labs): email address
Kenan Çağrı Hırlak (x2): cagri.hirlak@bilkent.edu.tr
Pouya Ghahramanian (x2): ghahramanian@bilkent.edu.tr
Sepehr Bakhshi (x2): sepehr.bakhshi@bilkent.edu.tr
Soheil Abadifard (x1): soheil.abadifard@bilkent.edu.tr

TAs; Tutor:
Section 1: Pouya Ghharamanian, Sepehr Bakhshi
Section 2: Kenan Çağrı Hırlak
Section 3: Pouya Ghharamanian, Sepehr Bakhshi
Section 4: Kenan Çağrı Hırlak, Soheil Abadifard

Tutor: Yarkın Kurt (x1): yarkin.kurt@ug.bilkent.edu.tr, Lab Section 3 (Thu morning)
Tutor: Efe Yakar (x2): efe.yakar@ug.bilkent.edu.tr, Lab Section 2 (Thu afternoon),
Section 4 (Fri afternoon 2 hours, may change based on his course meetings)

Recitation: For all sections, Monday 17:30-19:30 (by Yarkın Kurt), Place: EA-Z04

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.

In the Moodle Lab6 folder there is a simple program that shows the use of lh (load half word) and sh (store half word) instruction. There is also a pdf file that explains how many LRU (u) bits are needed for block replacement.

Summary

Preliminary Work: 50 points

Problem solving and writing a matrix addition program with a simple user interface.

Lab Work: 50 points

Experimental observations using a program that finds the summation of the elements of a matrix with different cache conditions.

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 OF PRELIMINARY WORK: SAME FOR ALL SECTIONS

No late submission will be accepted.

a. Please upload your problem solutions and programs of preliminary work to Moodle by 13:30 on Wednesday December 14 for similarity testing by MOSS. We plan to use MOSS both for problems solutions and program.

b. **Problems**: Use the filename **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).

c. Note that the Moodle submission closes sharp at 13:30 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.

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

e. 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.

DUE DATE OF LAB WORK): (different for each section) YOUR LAB DAY

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.

1. (5 points: With 2 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, LRU etc. is needed (yes) or not (no). If some combinations are not possible mark them.

No.	Cache Size KB	N way cache	Word Size in bits	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	128	1	32	4						
2	128	4	32	16						
3	128	Full	32	16						
4	256	2	64	8						
5	256	4	64	32						
6	256	Full	16	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 2 or more incorrect answers you get 0 points. Otherwise full point.)** Consider the following memory configuration: Assume that main memory size is 4GB. N= 1 (direct mapped), Block size= 8 bytes, No. of sets= 4.

Consider the following consecutive memory accesses and indicate which set is selected and indicate if it is a hit or miss.

Memory Address Accessed (hex)	**Set No.**	**Hit (yes/no)**
00 00 20 24		
00 00 20 42		
00 00 20 68		
00 00 20 04		
00 00 20 0C		
00 00 20 4C		

3. **(5 points: With 2 or more incorrect answers you get 0 points. Otherwise full point.)** Consider the following memory configuration: The same as Section 2 above only difference is N= 2. Block replacement policy is FIFO.

Consider the following memory accesses and indicate which set is selected and indicate if it is a hit or miss. **(With 2 or more errors you get 0 points. Otherwise full point.)**

Memory Address Accessed (hex)	**Set No.**	**Hit (yes/no)**
00 00 00 2C		
00 00 00 48		
00 00 00 44		
00 00 00 0C		
00 00 00 04		
00 00 00 0C		

4. **(5 points, With 1 or more incorrect answers you get 0 points. Otherwise full point.)**
Physical memory size is 4 GB.
Word size is 2 bytes.
Block size is 64 words.
Cache memory data area size= 1KB.
N= 16. Assume that LRU is used for block replacement.
D (dirty bit) is used to keep track of the changes in the cache blocks.

a. Show the structure of the physical address structure when it is used to access the cache memory: show its sub fields, their names, their sizes in number of bits.
b. What is the size of a block in terms of number of bits (it includes both data area and the overheads like tag etc.)? Include D bit in your calculations. Show the components of your calculation (its name like tag and its size in bits).
c. What is the size of a set in bits? What is the total SRAM size in bits?

d. If you use random replacement for blocks of a set what will be its effect on the SRAM size. Is it going to make it smaller, how many bits? If it makes it larger, how many bits? Explain 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 row by row. Create an array for the matrix elements and initialize them row by row with consecutive values. For example a 3 by 3 (N= 3) matrix would have the following values. Note that matrix elements are half words. This means that when you load them to a register you will use lh (load half word). Use sh (store half word) when you store a half word value to a memory location. See MARS help menu to understand how lh and sh works.

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	2	4	6	8	10	12	14	16
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 2 + (j - 1) x 2, for a matrix of size N x N.

Your user interface/program 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. Display desired elements of the matrix by specifying its row and column member,
4. Obtain summation of matrix elements row-major (row by row) summation,
5. Obtain summation of matrix elements column-major (column by column) summation,

When appropriate; such as items no. 3, 4, 5 above; use a subprogram and make sure that you follow the MIPS assembly language programming conventions (use of $a registers etc.).

2. [50 pts] Experiments with Data Cache Parameters

Run your program with two reasonably large different matrix sizes that would provide meaningful observations.

Report for Matrix Size 1: 25 Points

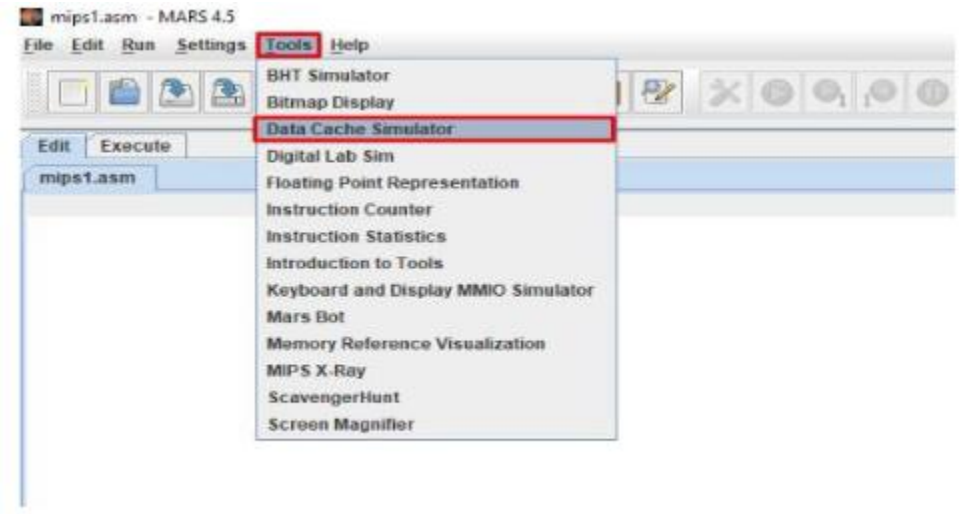
Report for Matrix Size 2: 25 Points

As described above make sure that you have a 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 row-major addition is a simple array addition from the beginning to the end; however, the column-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 in the textbook (see below). 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.

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 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.

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.

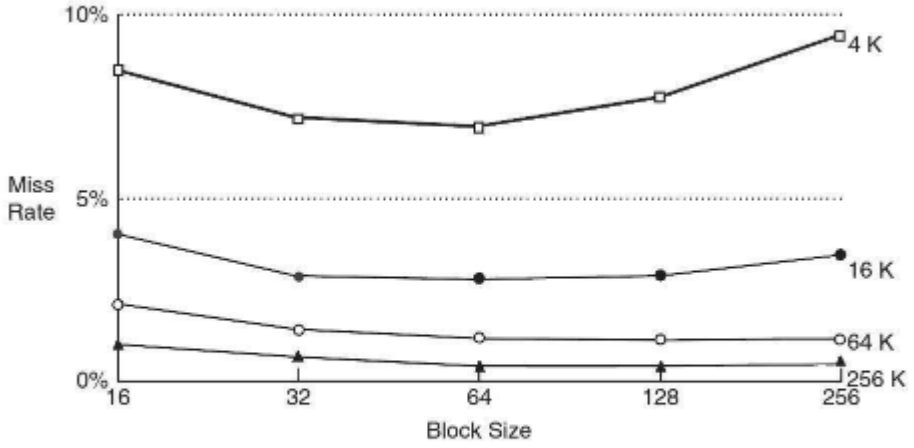


Figure 8.18 **Miss rate versus block size and cache size on SPEC92 benchmark**
Adapted with permission from Hennessy and Patterson, *Computer Architecture: A Quantitative Approach*, 5th ed., Morgan Kaufmann, 2012.

Part 3. Submit Lab Work for MOSS Similarity Testing

1. Submit your Lab Work MIPS codes for similarity testing to Moodle. See instructions below.
2. **Report**: Put your experiment results for Part 2 (including the tables and graphs) in a single PDF file. Use filename **StudentID_FirstName_LastName_SecNo_Lab6_lab_report.pdf** [pdf FILE as its extension suggests, which contains all the work done for the Lab Experiment Report Part].

   Code: Put your MIPS code for Part 1.5 into a .txt file. Use filename . Note that as you do the experiments your program may or may not involve changes. Whichever the case upload. **StudentID_FirstName_LastName_SecNo_Lab6_lab_code.txt** [A NOTEPAD FILE as its extension suggests, which contains the Program Code Part]
   Your program (code) will be compared against all the other programs 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! The same type of comparison is also planned for the reports.
3. *Even if you didn't finish, or didn't get the MIPS codes working, you must submit your code to the Moodle Assignment for similarity checking.*
4. 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!
5. At the beginning of your submission files include the following make sure that each of them is in a separate line: CS224, Lab No., Section No., Your Full Name, Bilkent ID.
6. For your preliminary and lab works to be graded you must attend the lab.

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. As indicated earlier: Attendance is mandatory and the preliminary work is graded only if you submit your lab work with the observation/permission of your TA.
3. The questions asked by the TA will have an effect on your lab score.
4. Lab score <u>will be reduced to 0</u> 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.

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