# Caches: A Primer on Data Locality

## Introduction

In this lab you will learn how the layout of data affects the locality in cache and cache performance. You will mainly learn:

1. How a program behavior affects the number of cache misses.
2. How cache size affects the total number of cache misses.
3. How block size affects the total number of cache misses.
4. How cache associativity affects the total number cache misses.

### What to turn in

You are expected to hand in the answers to all the listed questions. You will be required to show how you compute the answers and also give the reason for your answers/assumptions whenever required.

You are required to implement and submit at least one program (Part 2 – good loop order). Your computation of cache statistics will be based on your implemented solution.

## Getting Started

### Read the Background Material

Before you get started, it is strongly recommended that you read Chapter 5.2 in the 4th Edition of the book. This will explain in details the concepts behind caching. Specifically, you "should" know what is a cold/compulsory, conflict and capacity miss.

To know capacity misses at a specific cache size with a fix block size: simulate a fully associative cache of that size with the same cache block size, the misses encountered are the capacity misses.

To know conflict misses at a specific cache size with a fix block size: simulate a direct mapped cache of that size with the same cache block size, the misses encountered are the conflict misses.

### Files and programs

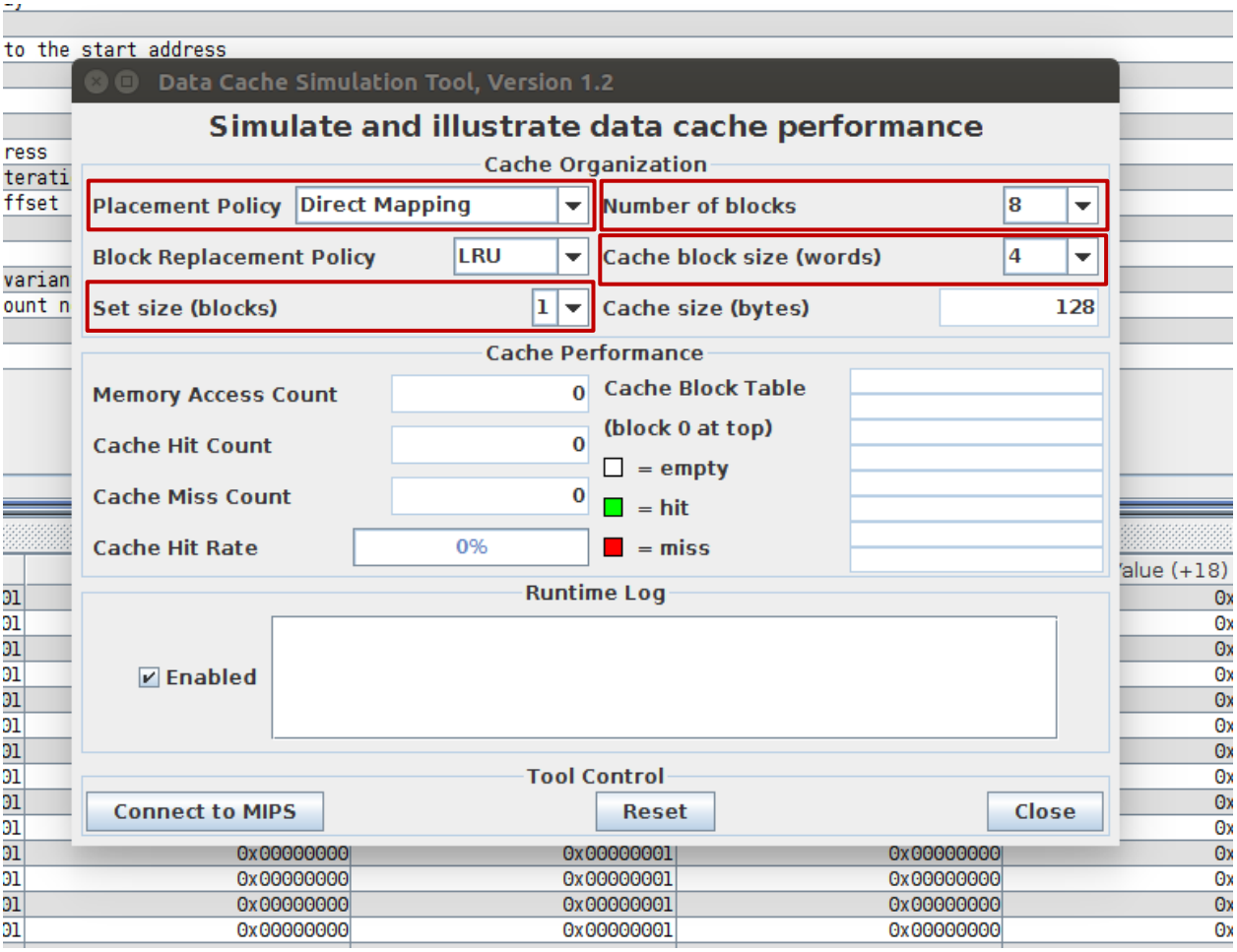
You are provided with the following files in the dark-cache-lab.zip archive:

- iteration-array-bad-loop.s
- conflicts.s

For this lab, we will use MARS Data cache simulator and experiment with a two-dimensional matrix to affect its data locality and reason about the cache hit/miss statistics.

## Step 1: Setting Up MARS Data cache simulator:

Open MARS, Settings -> Data cache simulator.

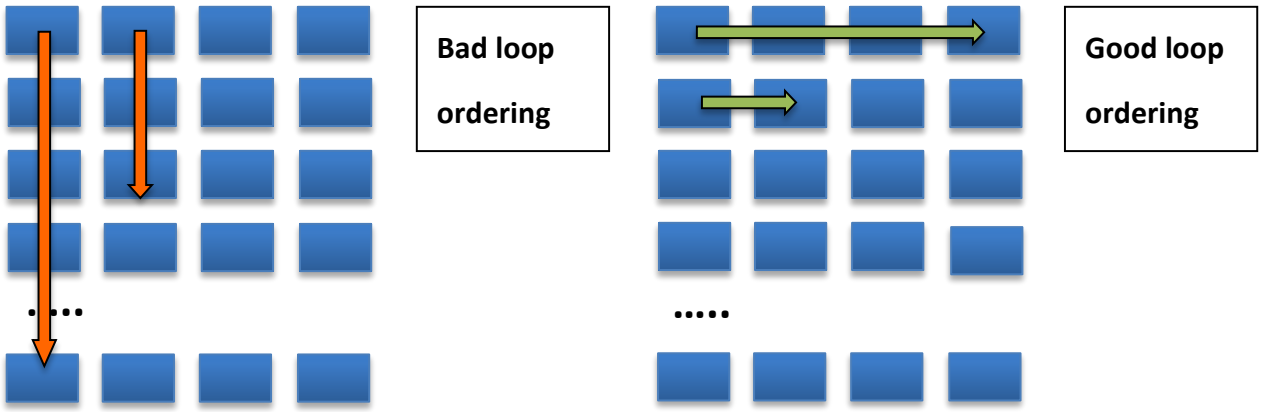



Click 'connect to MIPS', so that the program running in MARS will be connected to the data cache simulator. Now, whenever you run a program in MARS, it's data access addresses will be fed to the data cache simulator to simulate the data cache. You can change Placement policy to Direct Mapped or Fully associative or N-way associative. If you choose N-way associative cache, you have to set the Set size (blocks) parameter to whatever number of ways you want to have in the cache. Number of blocks parameter lets you select the total number of cache lines you want to have in the cache. You select the cacheline size in the cache by Cache block size (words) parameter. Note that a word is 4 bytes, so if you want to have a cache-line size of 4 bytes, choose 1 in Cache block size parameter. You cannot change the Cache size (bytes) parameter.

## Step 2: Loop Ordering

The way a program accesses data in the cache impacts the overall performance of the program by affecting the number of data cache misses.

The program in the file **iteration-array-bad-loop.s** traverses a matrix of dimensions 4x64. The matrix is laid out as a one-dimensional array such that the 4 elements of the 1st row are followed by the 4 elements of the 2nd row and so on. Each element in the array is 4 bytes. The program goes through each element in the matrix in the bad loop order (column-major ordering as illustrated below) and adds them to a variable sum. This loop is repeated 10 times. **Study the source code until you are convinced that the description above is correct.**




Now, load the file **iteration-array-bad-loop.s** in MARS.

### Part 1

Configure the **Cache Size** to 512 bytes, the **Block Size** to 4 bytes, the **Mapping** to "direct mapping" and run the program. Look at the statistics and answer the following questions

1. What is the hit rate? Why is it that much?
2. How many compulsory misses? Why is it that much?
3. How are the compulsory misses affected when Block Size is changed from 4 bytes to 8 bytes? Why does it change?
4. Change the block size back to 4 again, and change the mapping from "direct mapping" to "2-ways set associative"? Explain why the hit rate does or doesn't change.
5. What cache size improves the hit rate? Why is the hit rate improved? Try with both "direct mapping and "2-ways set associative".
6. At this "optimal" cache size (question 5), and "direct mapping", what hit rates do you obtain by changing the Block size? Try with the sizes 4 bytes, 8 bytes and 16 bytes.
7. Show how the achieved hit rate (as shown by simulator) can be computed (by hand) for different block sizes for the "optimal" cache size you found in question 5. *Hint: what is the size of a single element in the array?*

### Part 2

Rewrite the program in such a way that all the elements in the array are accessed sequentially i.e. in a good loop ordering or row major ordering. Configure the **Cache Size** to 512 bytes, the **Block Size** to 4 bytes, the **Mapping** to "direct mapping" and run the program. Look at the statistics and answer the following questions.

1. Does the hit rate improve? Why (show computation)?
2. Change the Block Size to 8 bytes? Does the hit rate improve? Why (show computation)?
3. Change the Block Size to 16 bytes? Does the hit rate improve? Why (show computation)?
4. Change the Cache Size to 1024 bytes and Reset the Block Size to 4 bytes. What is the hit rate? Compute this hit rate by hand
5. Change Block Size to 8 bytes. What is the hit rate? Compute this hit rate by hand
6. Change Block Size to 16 bytes. What is the hit rate? Compute this hit rate by hand
7. Can a perfect hit rate of 1.0 be achieved without changing the program? Why?

## Bonus: Conflicts

In this exercise we will examine the effects of conflicts in data caches. The file **conflicts.s** contains a program that traverses the elements of two equal sized arrays (A and B). There are 64 4-byte elements in each of these arrays. The program accesses the elements sequentially from both arrays and adds them to a variable sum. The loop is repeated 10 times.

Configure the **Cache Size** to 512 bytes, the **Block Size** to 4 bytes, the **Mapping** to "direct mapping" and run the program. Look at the statistics and answer the following questions

1. What is the hit rate? Explain why. (*Hint: look at conflict misses)*
2. Does changing the block size from 4 bytes to 8 bytes change the hit rate? Explain why it does or does not.
3. Reset the Block size to 4 bytes and change the Mapping from "direct mapping" to "2-way set associative". Does the hit rate change? Explain why it does or does not.
4. Change the Mapping from "2-way set associative" to "4-way set associative". Does the hit rate change compared to 2-way set associative? Explain why it does or does not.
5. What is the main purpose of increasing the associativity of a cache?

## That's it.

Turn in your answers to the written questions in an open standard text files (txt, rtf, pdf). Also submit the good array loop code.

## Revisions

$120801$ version 1.0 – Fall 2012 – Muneeb Khan
$131201$ version 1.1 – Fall 2013 – Ricardo Alves, Germán Ceballos
$141001$ version 2.0 – Fall 2014 – Germán Ceballos
$160818$ version 2.1 – Fall 2016 – Johan Janzén
$171004$ version 2.2 – Fall 2017 – Ricardo Alves, Germán Ceballos
$181002$ version 2.3 – Fall 2018 – Ricardo Alves
$200221$ version 2.4 – Spring 2020 – Muhammad Hassan