CS2323 Lab Assignment -7

Kartik Srinivas - ES20BTECH11015

Department of Computer Science and Engineering Science, IIT Hyderabad

November 21, 2022

Contents

1 Analysis of Reads

1.0.1 Program P1

Say that we are analyzing the case with four blocks and direct mapped (non assoscative case) P1 is a program where the register `x12` is read and is incremented by 8 every single time, the main idea is to see once every four times there is a cache miss. Therefore the average cache perofmance should converge to $3/4 \approx 0.75$. **This won't change if we change the number of Line's in the cache**, simply because our memory access pattern is Sequential in nature, i.e. the next memory location accessed for read is always **new**. There are no instances where I access a previously accessed memory location with a different Tag The only way the performance of the cache will change is if we change the number of blocks that are addressed by a particular Tag will change. i.e increase in blocks as illustrated by the following Table

L1	Blocks	Lines	Ways	Hit rate
8,8	2	32	1	0.5
8,8	2	16	1	0.5
8,8	2	8	1	0.5
8,8	4	32	1	0.7424
8,8	8	32	1	0.8636
8,8	16	32	1	0.9242
8,16	2	32	1	0.5
8,16	4	32	1	0.7424
8,16	8	32	1	0.8636
8,16	16	32	1	0.9242

1.0.2 Program P2

The main difference as compared to P1 is that `x12` is incremented by `x15` at every iteration, and it is **reset to the value of the register** `x15` which falls back to previously accessed memory locations. So, larger line sizes(up to some extent) will help use imporve the hit rate. The number or cache acesses will be the same at the value of 64. But the hit rate will be higher(because the cache conflicts will decrease) If I increase the number of ways then the performance will increase(when you have a smaller number of lines), again, as the cache will start having more space and the conflicts will decrease. Usually in terms of the cycles of the outer loop of the program, we get Loops 1, 3 Have cache misses and Loops 2, 4, 5, 6, 7, 8 have cache hits, so the performance of the cache will converge to $6/8 \approx 3/4 \approx 0.75$ again(in the case of four blocks) **Increasing the number of blocks again improves the performance**. Excessive lines (after a certain point) will not help us improve the performance, because the cache will not be utilized that much anyway!

L1	Blocks	Lines	Ways	Hit rate
8,8	32	8	1	0.9545
8,8	16	8	1	0.9242
8,8	8	8	1	0.8636
8,8	4	8	1	0.0454
8,8	32	32	1	0.9545

2 Analysis of Write's

2.0.1 Write-Back

In this section of the code we can see the Dirty bits etc. During **Write Allocate** the memory access patters will remain the same!, because WA behaves just like read miss when the block is brough to the cache. So the hit rates will remain the same in Write allocate scenario. The only difference is that the Dirty bit will be set to 1. Writebacks only occur when there is an **evicition** from the cache.

2.1 Problem -1

In this problem the memory acesses are very sequential in nature and therefore unless the size of the cache is very small there will not be many write-backs

2.2 Problem -2

In this case, if you are given enough blocks or lines, the number of Writebacks will be 0!. This is because the memory access pattern of the second program conforms to that of hits on prevously accessed locations or just new locations (which are not in the cache) , very rarley do we see a case where the tag matches but the index does not.

L1	Blocks	Lines	Writebacks
8,8	2	32	0
8,8	32	2	0
16, 16	32	2	126
8, 8	2	8	248

The red numbers show that if the space in the cache is too small or if the number of accesses is too large, then the number of conflicts = Writebacks are quite large and the performance is poor.