CSE 30321 – Computer Architecture I – Fall 2010

Lab 04: Cache Organization and Design
Total Points: 100 points
Assigned: November 2, 2010
Due: November 18, 2010

Goals and Description

The purpose of this lab is to see – for real benchmarks and real processor configurations – how both the presence of caches, and cache organization can impact benchmark performance. You will also be asked to use simulation data to make recommendations about how a cache should be organized given a specific benchmark suite. In this regard, while I *will* show you how you can use the SimpleScalar simulator to modify a given cache configuration, *you* will need to decide what simulations to perform to make a convincing argument as to why your chosen design is best.

Suggestion

It will be much easier to explain/answer the questions in this lab with neatly formatted graphs and/or tables!

Part A: Quantifying how the presence of cache(s) impact performance

Background:

As you have seen in lecture, the presence of caches can significantly impact microprocessor performance. For example, in Lecture 18, we considered what would happen if – for a pipelined processor – 10% of the time we had to wait 75 clock cycles instead of just 1 clock cycle to get data from memory. The net effect was that average CPI increased by a factor of 10!

In Part A of this lab, you will need to determine how the execution time of the *lame* and *mad* benchmarks is affected assuming that:

1. There is no cache on chip.
2. There is just one level of cache on chip
3. There is a first and a second level of cache on chip

To do this, you will need to modify a processor configuration file. To reduce the number of simulations that you need to do, we will just work with the Xscale configuration that was used in previous labs. A baseline configuration file can be found at: /afs/nd.edu/coursefa.10/cse/cse30321.01/Labs/04/xscale_partA.cfg

To get started, download and open this file so that you can make edits to it.

- Once the file is open, scroll down until you see the entry:

```
# l1 data cache config, i.e., {<config>|none}
-cache:dl1          dl1:32:32:32:f
```

- This line in the configuration file allows you to specify how the cache is organized. More specifically, the different entries refer to the following:

  <cache name> : <# of sets in cache> : <# of bytes/cache block> : <associativity> : <replacement policy>

- Thus, for the example entry given above:
  - "dl1" is used to note that the cache we have added is a Level 1 Data Cache (<cache name>).
  - There are 32 sets in the cache (<# of sets in cache>).
  - Each cache block has 32 bytes associated with it (<# of bytes/cache block>)
  - The cache is 32-way set associative (<associativity>)
  - A "first in, first out" replacement policy is used (<replacement policy>).

- We can use this information to discern some other information about our cache too…
  - i.e. we can calculate the total size of the cache:
    - The fact that the cache is 32-way set associative means that there are 32 blocks per set.
    - The fact that there are 32 sets means that there are 32 x 32 = 1024 blocks in the cache.
    - As there are 32 bytes per block, the cache can hold 32 x 1024 = 32,768 bytes of data (i.e. this is a 32 Kbyte cache).
    - Assuming that data words are 4-bytes (i.e. 32 bits) long, the cache can hold 8,192 data words.

- Note that you will also see the following entry in your configuration file:

```
# l1 instruction cache config, i.e., {<config>|none}
-cache:il1          il1:32:32:32:f
```

- This entry specifies the organization of the *instruction cache*.
  - In the Xscale microprocessor, there are separate caches for instruction encodings and data words. The caches generally have the same organization.
  - *For this lab, you can always assume that the instruction and data cache will have the same organization, AND that there will always be an instruction and a data cache.*

To Do:

For the *mad* and *lame* benchmarks, with large and small data sets, determine how benchmark performance is affected assuming:

1. There are no on-chip caches.
2. There are just level 1 caches.
3. There is are level 1 and level 2 caches on chip.

Essentially, you need to complete the following table:

	No L1, No L2	L1, no L2	L1 and L2
Lame (small)			
Lame (large)			
Mad (small)			
Mad (large)			

Some notes:

- If you would like, you may assume that the processor's clock rate is 233 MHz.

- The easiest way to simulate a design with no cache, is to modify the configuration file so that the level 1 cache has just one block.
    - Per the specifications of the simulator, a cache block must contain at least 8 bytes of data.
    - Thus, for design number 1 (no on-chip caches), your cache configurations should be:

```
-cache:dl1          dl1:1:8:1:f
-cache:il1          il1:1:8:1:f
```

- When you simulate a processor configuration with a level 2 cache, you will need to update the following lines in the base configuration file:

```
# l2 data cache config, i.e., {<config>|none}
-cache:dl2          none

# l2 instruction cache config, i.e., {<config>|none}
-cache:il2          none
```

    - More specifically, you will need to change the entry from "none" to something like:

```
dl2: <# of sets in cache> : <# of bytes/cache block> : <associativity> :f
```

    - Your level 2 caches should have 256 sets, 32 bytes per block, and be 32-way set associative.
    - Your level 1 caches should have the same configuration as given in the base configuration file.

Part B: Quantifying how cache organization impacts performance (part 1)

Background:

As seen in Lecture 19, it is also possible to organize a cache in different ways. More specifically, as seen in Fig. 1 below, a cache that stores the same amount of information can be organized such that it is "long and skinny" or "short and fat":

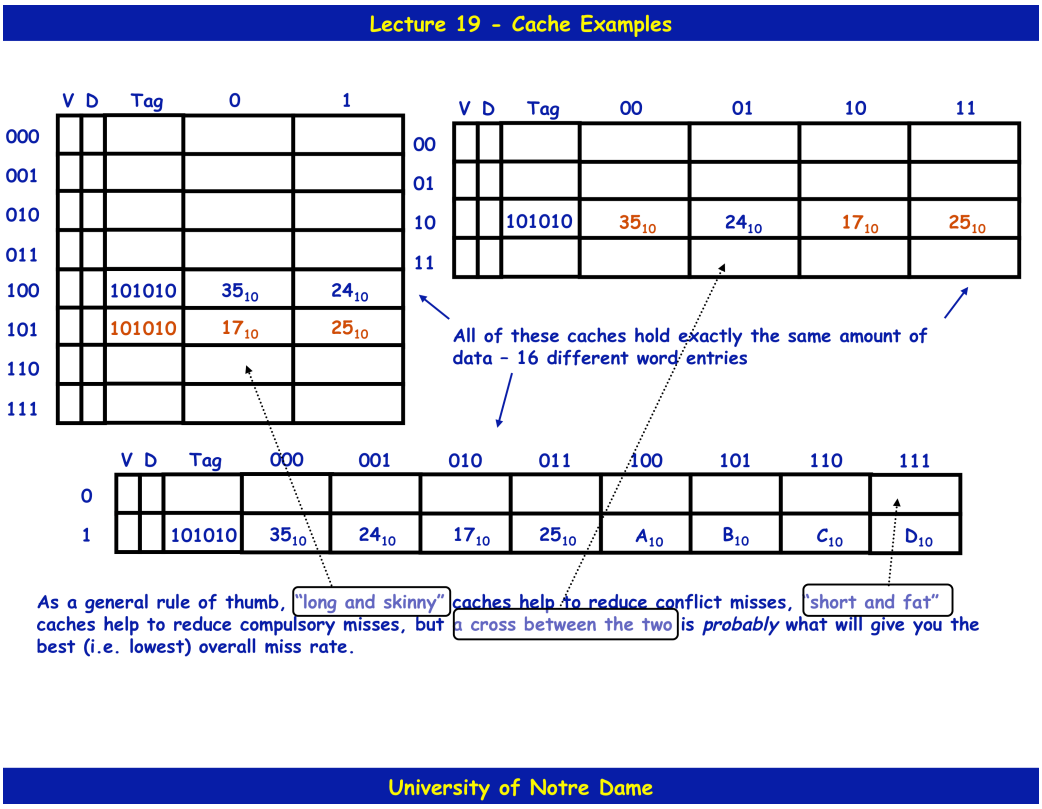

Figure 1: Caches can be "long and skinny". A "long and skinny" cache has a large number of blocks (but just a few data words may be associated with each block). A "short and fat" cache may have many data words per block, but the number of blocks is reduced. Generally, the best performance comes from a cache design that is neither "long and skinny" or "short and fat", but somewhere in between.

For this part of the lab, you will be asked to determine the ideal cache configuration for a suite of benchmarks. More specifically (and to simplify the amount of simulation work you have to do), we will assume the following:

- Your cache organizations are direct mapped (and not N-way set associative)
- The benchmark suite to consider will involve just the large data sets for the *ispell*, *jpeg-compress*, *jpeg-decompress*, *lame*, *mad*, and *patricia* benchmarks.

To begin, download the configuration file at: /afs/nd.edu/coursefa.10/cse/cse30321.01/Labs/04/xscale_partB.cfg

Note that this configuration file is the same as that for Part A with one exception: the cache organization has been changed from 32-way set associative to direct mapped:

Old configuration syntax:	-cache:dl1	dl1:32:32:32:f
New configuration syntax:	-cache:dl1	dl1:256:128:1:f

Before continuing, be sure that you understand how (i) the new syntax specifies a direct mapped cache and (ii) that the cache size specified with the new syntax is the same as that specified by the old syntax:

Old configuration size:	(32 sets) x (32 bytes / block) x (32 blocks / set)	= 32 Kbytes
New configuration size:	(256 sets) x (128 bytes / block) x (1 block / set)	= 32 Kbytes

To Do:

Question 1:

For this part of the lab, you will need to explore how different cache organizations impact the performance of the benchmark suite described above. Your job is to (a) find the cache organization that provides the best performance for the suite, and (b) justify your answer with simulation data. My suggestion would be to fill in multiple rows of the table below (where the benchmark entries represent a measure of performance for a given cache configuration):

Cache configuration	*ispell*	*jpeg-compress*	*jpeg-decompress*	*lame*	*mad*	*patricia*
4096 blocks, 8 bytes/block						
…						
256 blocks, 128 bytes/block						
…						

A few notes:

- Remember – you should run the benchmarks with just the large datasets.
- You only need to consider direct mapped designs (so the number of sets should always be 1).
- Your cache size should always remain the same – i.e. the number of blocks and bytes per block chosen should always lead to a 32 Kbyte cache.
- Be sure to update the instruction cache accordingly too.
- You should not expect *one* design to be the absolute best for *every* benchmark!
- There is no single right answer to this question. Credit will heavily depend on your justification, data to support your justification, and your simulation methodology. Thus, in addition to testing your knowledge about caches, this question also tests your ability to make convincing design decisions.
- For this part (and the remainder of) the lab, there you should assume that there are NO L2 caches.

Hint:

- To expedite your work for this problem, I would suggest that you identify a general trend by varying cache configurations for 1 or 2 of the benchmarks (the *ispell* and *jpeg* benchmarks generally take the least amount of time to run). This might allow you to narrow your simulation space for the longer running benchmarks.

Question 2:

Qualitatively, why do you think one cache organization might lead to the best performance for benchmark A, while another might lead to better performance for benchmark B? Your answer does not need to be longer than 4-6 sentences.

Part C: Quantifying how cache organization impacts performance (part 2)

Background:

In Part B, we considered different ways to organize *a direct-mapped cache*. As you saw in Lecture 18, you can also vary the associativity to help improve performance. In Part C of this lab, we will quantify how changing associativity can impact benchmark performance.

To Do:

Question 1:

Using the direct mapped cache organization that you decided was optimal (from Part B), determine how increased associativity can impact performance. Based on your simulation results, what associativity would you recommend? (i.e. for an *N*-way set associative cache, what number should *N* be?)

Notes:

- You only need to do this for *one* of the 6 benchmarks.
- You should consider a fully-associative cache as part of your study.

Question 2:

Why does increasing the associativity help improve performance?

Part D: Revisiting Part B and Part C

To Do:

Question 1:

Using your recommended cache configuration from Part B, and the associativity determined in Part C, re-run all 6 benchmarks with this new configuration. (Again, the cache should still be 32 Kbytes.) Compare the performance of all benchmarks with this new design to that determined in Part B. What is the average speedup?

Note: basically, you need to complete a table similar to that shown below:

Cache configuration	*ispell*	*jpeg-compress*	*jpeg-decompress*	*lame*	*mad*	*patricia*
Part B (direct mapped config.)						
Part D (new config.)						

Question 2:

Using your answers from Parts A-D, what do these results tell you about how to best organize a cache?