CSE 30321 – Computer Architecture I – Fall 2009

Homework 08 – Pipelined Processors and Multi-core Programming

Assigned: November 17, 2009 **– Due:** December 1, 2009

This assignment can be done in groups of 1, 2, or 3. The assignment is worth 85 points

Problem 1: (10 points)

Background:

Memory hierarchies can improve performance for more than one reason. Often, reasons are related to a cache improving either temporal locality (i.e. data or an instruction that was referenced once will be referenced again soon) or spatial locality (i.e. data or an instruction nearby data or an instruction that was referenced will probably be needed soon).

Question:

Describe the general characteristics of a program that would exhibit very high amounts of temporal locality but very little spatial locality with regard to data accesses. Provide an example program (pseudocode is fine).

Problem 2: (30 points)

Background:

One possible organization of the memory hierarchy on an Intel Pentium 4 chip is:

- An 8KB L1 instruction cache
  - (e.g. the L1 instruction cache contains *only* instruction encodings)
- An 8KB L1 data cache
  - (e.g. the L1 instruction cache contains *only* data words)
- A 256KB unified L2 cache
  - (e.g. the L2 cache can contain both data and instruction encodings)
- An 8 MB unified L3 cache
  - (e.g. the L3 cache can contain both data and instruction encodings)

Question A: (5 points)

What might be a benefit of splitting up the L1 cache into separate instruction and data caches?

Question B: (5 points)

The L1 data cache in the P4 holds 8 KBytes of data. Each block holds 64 bytes of data and the cache is 4-way set associative. How many blocks are in the cache? How many sets are in the cache?

Question C: (5 points)

The L2 cache in the P4 holds 256 KBytes of data. The cache is 8-way set associative. Each block holds 128 bytes of data. If physical addresses are 32 bits long, each data word is 32 bits, and entries are word addressable, what bits of the 32 bit physical address comprise the tag, index and offset?

Question D: (5 points)

The L3 cache in the P4 holds 8 MBytes of data. It too is 8-way set associative and each block holds 128 bytes of data. If physical addresses are 32 bits long, each data word is 32 bits, and entries are word addressable, what bits of the 32 bit physical address comprise the tag, index and offset?

Question E: (10 points)

For the P4 architecture discussed above, we know the following:

- The hit time for either L1 cache is 2 CCs
- The time required to find data in the L2 cache is 7 CCs
  - Thus, the time to find data in the L2 cache is really 9 CCs – 2 CCs to look in L1 and 7 more CCs to get the data from L2
- The time required to find data in the L3 cache is 10 CCs
  - Thus, the time to find data in the L3 cache is 9 CCs + 10 CCs – because we look in L1 and L2 first
- If data is not found in the L3 cache, we will need to get data from main memory.
  - The bus between main memory and the L3 cache can transfer 64 bytes of data at a time
    - Thus, 2 main memory references are required to fill up the 128 byte cache block
  - It takes 50 CCs to access main memory once and accesses cannot overlap.

For the benchmark that we are running:

- The L1 cache hit rate is 90%
- The L2 cache hit rate is 95%
- The L3 cache hit rate is 99%

What is the average memory access time?

Problem 3: (5 points)

Background:

A certain computer has the following characteristics:

- A physical address that is 20 bits long
    - **(not all addresses are/have to be 32 bits!)**
- A cache with each block holding 8 data words
- Addresses that are to the word.
- The cache has 1024 ***blocks***.
- The cache is direct mapped

Question:

How may bits make up the index, tag, and offset of this physical address?

Problem 4: (10 points)

Background:

You are to design a cache:

- …that holds 2 KB (note BYTES) of data.
- …where each data word is just 8 bits long (and addresses are to the word).
- …that is 4-way set associative
- …that has 64 total ***blocks***
- …that uses a write back policy

Question:

Draw the cache. [Note, you don't have to draw every last thing. Just provide enough information so that we know you go the problem right. I.e. if there are 19 words in a cache block, write block 0, block 1, …. block 18

Problem 5 (20 points):

Consider a cache with the following specs:

- It is 4-way set associative
- It holds 64 KB of *data*
- Data words are 32 bits each
- Data words are *byte* addressed
- Physical addresses are 32 bits
- There are 64 words per cache block
- A First-In, First-Out replacement policy is used for each set
- All cache entries are initially empty (i.e. their valid bits are not set)

At startup the following addresses (in hex) are supplied to this cache in the order below:

1.	AA	AB	BB	BC
2.	AA	AB	BB	BF
3.	FC	CB	BB	BD
4.	AA	AB	BB	BF
5.	FF	FB	BB	BC
6.	FF	FB	BB	BB
7.	FC	BD	BB	AC
8.	AA	AB	CC	CF
9.	AA	AB	BB	FF
10.	FF	FF	AA	BC

Question A: (5 points)

- How many *sets* of the cache has this pattern of accesses touched?

Question B: (5 points)

- How many compulsory misses are there for this pattern of accesses?

Question C: (5 points)

- How many conflict misses are there for this pattern of accesses?

Question D: (5 points)

- What is the overall miss rate for this pattern of accesses?

Problem 6 (10 points):

Background:
There are many ways to reduce cache misses by changing processor hardware – e.g. making cache blocks larger, making larger caches, increasing associativity, etc. However, cache performance can be improved by optimizing software too.

Question:
If the arrays in the for loop below are stored in memory, there could be many cache misses. Explain why.

```
for (j = 0; j < 100; j = j + 1) {
        for (i = 0; i < 5000; i = i + 1) {
                x [i] [j] = 2 * x [i] [j]
```

Extra Credit:

Background:
There are many ways to reduce cache misses by changing processor hardware – e.g. making cache blocks larger, making larger caches, increasing associativity, etc. However, cache performance can be improved by optimizing software too.

Question:
Another technique that improves cache performance at the software level is called “Blocking”. By doing your own research, explain how blocking improves cache performance.