CSE 30321 – Computer Architecture I – Fall 2010

Homework 07 – Memory Hierarchies

Assigned: November 9, 2010**, Due:** November 23, 2010, **Total Points:** 100

Problem 1: (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 2: (10 points)

Background:

You are to design a cache:

- …that holds 4 KB (note BYTES) of data.
- …where each data word is just 16 bits long (and addresses are to the word).
- …that is 2-way set associative
- …that has128 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 3 (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	10	11-B	BC	-	compulsory miss (block 1)
2.	AA	AB	10	11-B	BF	-	hit (block 1)
3.	FC	CB	10	11-B	BD	-	compulsory miss (block 2)
4.	AA	AB	10	11-B	BF	-	hit (block 2)
5.	FF	FB	10	11-B	BC	-	compulsory miss (block 3)
6.	FF	FB	10	11-B	BB	-	hit (block 3)
7.	FC	BD	10	11-B	AC	-	compulsory miss (block 4)
8.	AA	AB	11	00-C	CF	-	compulsory miss (block 1)
9.	AA	AB	10	11-B	FF	-	hit (block 1)
10.	FF	FF	10	10-A	BC	-	compulsory miss (block 1)

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 4 (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]
```

Problem 5 (30 points):

As discussed in class, virtual memory uses a page table and TLB to track the mapping of virtual addresses to physical addresses. This exercise shows how the TLB must be updated as addresses are accessed.

For this problem, you should assume the following:

1. Pages have 4 KB of addressable locations
2. There is a 4-entry, fully-associative TLB
3. The TLB uses a true, least-recently-used replacement policy

For the pattern of virtual addresses shown below, comment on whether the entry is:

a. Found in the TLB
b. Not found in the TLB but is found in the Page Table
c. Not found in the TLB or the Page Table (thus, there is a page fault)

The initial TLB and page table state is shown below.

If there is a page fault, and you need to replace a page from disk, assume the page number is one higher than the current highest page in the page table. (This is currently 1100_2 / 12_{10}. Thus, the next page would be 1101_2 / 13_{10}, the next would be 1110_2 / 14_{10}, etc.)

Stream of Virtual Addresses:

	MSB			LSB
1	0000	1111	1111	1111
2	0111	1010	0010	1000
3	0011	1101	1010	1101
4	0011	1010	1001	1000
5	0001	1100	0001	1001
6	0001	0000	0000	0000
7	0010	0010	1101	0000

Initial TLB State:
(Note that ‘1’ = “Most Recently Used and ‘4’ = “Least Recently Used”)

Valid	LRU	Tag	Physical Page #
1	3	1011	1100
1	2	0111	0100
1	1	1001	0110
0	4	0000	-----

Initial Page Table State:

	Valid	Physical Page #
0000	1	0101
0001	0	Disk
0010	0	Disk
0011	1	0110
0100	1	1001
0101	1	1011
0110	0	Disk
0111	1	0100
1000	0	Disk
1001	0	Disk
1010	1	0011
1011	1	1100

Question A: (15 points) – **You must use these tables to answer this question!**
Given the virtual addresses above, show how the state of the TLB changes by filling in the tables:

Address 1:

1	0000	1111	1111	1111

Valid	LRU	Tag	Physical Page #

This reference is a:

Address 2:

2	0111	1010	0010	1000

Valid	LRU	Tag	Physical Page #

This reference is a:

Address 3:

3	0011	1101	1010	1101

Valid	LRU	Tag	Physical Page #

This reference is a:

Address 4:

4	0011	1010	1001	1000

Valid	LRU	Tag	Physical Page #

This reference is a:

Address 5:

5	0001	1100	0001	1001

Valid	LRU	Tag	Physical Page #

This reference is a:

Address 6:

6	0001	0000	0000	0000

Valid	LRU	Tag	Physical Page #

This reference is a:

Address 7:

7	0010	0010	1101	0000

Valid	LRU	Tag	Physical Page #

This reference is a:

Question B: (10 points)
What would some of the advantages of having a larger page size be? What are some of the disadvantages?

Question C: (5 points)
Given the parameters in the table above, calculate the total page table size in a system running 5 applications.

Virtual Address Size	Page Size	Page Table Entry Size
64 bits	16 KB	8 bytes