Homework Assignment 3

CS/ECE 6810: Computer Architecture
October 31,2018
Name: Jake Pitkin
UID: u0891770

OoO and Cache Optimization

Due Date: November 11, 2018.
120 points

1. **OoO Processor and Cache Parameters.** Please specify each of the following statement is True or False and explain why.

   i Instructions read all source operands from the register file in the Tomasulos algorithm.**(2 points)**

   False. A source operand could also be read from a reservation station as Tomasulo's algorithm uses these for register renaming.

   ii In a processor with hardware speculation, a store can always be committed as soon as its effective address and source value are available in the load-store queue.**(2 points)**

   True. Effective address is required for dependence check and we check availability of operands every cycle to determine if we can execute the store operation.

   iii If cache block size remains same, but the number of cache blocks doubles, the compulsory misses remain the same.**(2 points)**

   False. A compulsory miss is the first access to a block. With twice as many cache blocks, we increase the number of possible compulsory misses. Increasing the cache block size instead could decrease the number of compulsory misses.

   iv If the cache capacity is fixed but the block size is increased, the miss rate will not change.**(2 points)**

   False. The number of compulsory misses will decrease as the block size is increasing. Additionally, changing the cache design will most likely change the conflict miss rate (depending on the access pattern of a given program) compared to the miss rate of the original design.

   v Write-through caches use write allocate mechanism to prevent writing dirty blocks to lower memory levels.**(2 points)**

2. **Memory Hierarchy.** Consider a processor with the following memory organization: L1 Cache, L2 Cache, L3 Cache and main memory. Each cache stores both tags and data. The data and tag arrays are going to be accessed with the same index value. Assume that the processor does serial tag/data look-up (first tag lookup and then data access) for L2 and L3 caches and parallel tag/data look-up for L1 cache. The table given below provides the time take to access the tag and data arrays in CPU cycles. Find the number of cycles required to complete 2000 load instructions accessing this hierarchy.**(15 points)**

Level	Tag Access	Data Access	Hit Rate
L1 (32 KB)	1	-	40%
L2 (1 MB)	4	12	50%
L3 (8 MB)	25	90	80%
Main Memory	-	500	-

3. **Cache Hit Rate.** Consider a 256K main memory system with a direct mapped cache that stores up to 4 blocks. Each cache block comprises 4 words. The replacement policy is MRU and at the beginning cache is empty. Compute the hit rate for the following stream of addresses generated by the processor.(All memory addresses are in decimal format and each address refer to a word).**(10 points)**

   `170 , 257, 168, 246, 176, 175, 176, 177, 175, 176, 177, 175, 176, 177, 176, 175, 174, 173, 172, 171, 170, 169, 168, 167, 168, 165, 164`

4. **Cache Performance.** Consider the following pseudo code; suppose that **X** and **Y** are allocated as contiguous integer arrays in memory and are aligned to the 4KB boundaries. Assume **k** and **l** are allocated in the register file with no need for memory accesses. We execute the code on a machine where the integer type (**int**) is 4 bytes wide.

```
#define M 1024
int X[M*M];
int Y[M*M];
int k, l;
  for (k = 0; k < M; k++) do
    for (l = 0; l < M; l++) do
      Y[l*M+k] = X[k*M+l];
    end for
  end for
```

   i Consider a 4KB 2-way set-associative cache architecture while cache block size is 32-byte (8-word) and the replacement policy is LRU. Compute the hit rate of data accesses generated by `load`s and `store`s to **X** and **Y**?**(15 points)**

   ii Consider a 4KB fully associative cache architecture with 32-byte blocks. The replacement policy is LRU. Rewrite the code to remove all of the non-compulsory misses. (You need to ensure the new code generate the exact same output in the main memory.You are allowed to add a nested for loop to the code if necessary.) Please provide explanation on how the new code can remove those misses.**(15 points)**

5. **Cache Addressing.** Consider a processor using 3 cache levels. Level-1 cache is a 32KB direct mapped cache with 16B blocks used for both instructions and data. Level-2 is a 256KB, 2-way set associative cache with 64B cache lines. Level-3 is a 1MB, 4-way set associative cache with 64B cache lines. Assume that the processor can address up to 16GB of main memory. Compute the size of the tag arrays in KB for each level.**(15 points)**

6. **Cache and Memory Model using CACTI**

   CACTI (`http://www.cs.utah.edu/~rajeev/cacti7/`) is an integrated cache and memory model for access time, cycle time, area, leakage, and dynamic power. You are asked to use CACTI 7 for investigating the impact of small and simple caches using a 22nm CMOS technology node. Consider a processor using 2 cache levels. Level-1 cache is a 32 KB direct mapped cache with 16B blocks used for both instructions and data. Level-2 is

a 1MB, 4-way set associative cache with 64B cache lines. Assume that the processor can address up to 2GB of main memory. Assume that both of the Level-1 and Level-2 caches are single bank.

i Compare the access time, energy, leakage power, and area of the caches mentioned above.**(10 points)**

ii Investigate the impact of varying associativity for 1, 2, and 8 on the access time, energy, leakage, and area of the Level-2 cache.**(10 points)**

7. **Bonus Question.** The table below lists a sequence of loads and stores in the load-store queue (LSQ), when their one/two input operands are made available, and their computed effective addresses. Find when the address calculation happens for each load/store and when each load/store accesses the data memory. The processor assumes that loads do not depend on prior stores and issues loads speculatively. If the speculation is incorrect, the load must be re-issued. Assume that there are enough memory ports that you don't have to worry about structural hazards.**(20 points)**

LD/ST	The register for the address calculation is made available	The register that must be stored into memory is made available	The calculated effective addr (hexadecimal)	Addr calculation happens	Data memory accessed
LD	6	-	12345678		
ST	9	8	87654321		
LD	1	-	87654321		
LD	5	-	12345678		
ST	23	30	87654321		
LD	4	-	87654321		