Department of Computer Science
National Tsing Hua University
CS5100 Advanced Computer Architecture
Spring, 2017

Homework 2

Deadline: 2017/03/29

In this homework, you will work on the *Dinero IV* cache simulator and *Gem5* simulator to do trace-driven cache simulation to evaluate the performance of different cache policies and configurations.

1. Please follow the instructions in the accompany slide file "Trace-driven cache simulation" to install Dinero IV and to learn how to obtain memory access trace using Gem5.

2. Do the trace-driven cache simulation with the baseline configuration, which is given the in the "Trace-driven cache simulation" slide. Next, compare the performance of different enhancement strategies and policies with the baseline:
   (1) Double the L1 I-cache and D-cache size of the baseline
   (2) Use 4-way set-associative I-cache and D-cache
   (3) Use the LRU replacement policy
   (4) Enable write allocate policy
   (5) Add L2 D-cache with cache size 16k bytes and block size 16 bytes
   To compare the performance, you can use cache miss rate, number of capacity misses and conflict misses. Report the results and comment on your findings.

3. Let us compare the write bandwidth requirements of write-through versus write-back caches using a concrete example.
   Assume that we have a 64 KB cache with a block size of 32 bytes. The cache will allocate an empty block on a write miss. If configured as a write-back cache, it will write back the whole dirty block if the block needs to be replaced. Assume also that the cache is connected to the lower level in the hierarchy through a 4-byte-wide bus. The number of CPU cycles for a B-bytes write access on this bus is

   $$15 + 5\left(\left[\frac{B}{4}\right] + 1\right)$$

   For example, a 12-byte transfer would take 35 cycles. Answer the following questions while referring to the C code snippet below, where **data[j]** has 4 bytes and the write to **data[base]** causes a write miss:

```
#define PORTION 1
base = 8*i; //point to the start of a block
for(unsigned int j = base; j < base+PORTION; j++) {
    //assume j is stored in a register
    data[j] = j;
}
```

(1) For a write-through cache, how many CPU cycles are spent on transferring data between the two levels of the caches for all the combined iterations of the j-loop?

(2) If the cache is configured as a write-back cache, how many CPU cycles are spent on transferring data between the two levels of the caches for all the combined iterations of the j-loop, assuming the cache block to be replaced by that of **data[base]** is dirty.

(3) Change PORTION to 8 and repeat part (1) and (2).

(4) What is the minimum number of array updates to the same cache block (before replacing it) that would render the write-back cache superior?