EECE 7352: Computer Architecture
HW4: Due at 11:59pm April 21, 2022

As always, goals of these assignments are to help you: (1) practice knowledge gained during the lectures, (2) learn new technical material (esp. experimental learning), (3) apply course concepts and principles to new problems, (4) develop additional skills in expressing yourself in writing and orally, (5) analyze and evaluate ideas, arguments, and points of view.

You will not receive any points if you simply copy paste from the suggested readings or online resources. Please express your ideas, understanding and viewpoints in your own words (why steal others' words to convey what should be completely your point!).

Question 1. (30 points)

Download and set up the Dinero IV cache simulator on a COE machine. The Dinero IV cache simulator is available here: http://pages.cs.wisc.edu/~markhill/DineroIV/

We will use the traditional din input format for the trace (-informat d). You can find more details on using dineroIV by issuing dineroIV –help. Dinero allows you to configure a cache using command line. The simulator takes as input an address trace. We will use the "din" input format, which is an ASCII file with one LABEL and one ADDRESS per line.

The rest of the line is ignored so that it can be used for comments.
LABEL
= 0 read data
= 1 write data
= 2 instruction fetch
= 3 escape record (treated as unknown access type)
= 4 escape record (causes cache flush)
0 <= ADDRESS <= ffffffff where the hexadecimal addresses are NOT preceded by "0x."

Here are some examples:
2 0 This is an instruction fetch at hex address 0.
0 1000 This is a data read at hex address 1000.
1 70f60888 This is a data write at hex address 70f60888.

Given a trace file (e.g., trace.txt), you run Dinero as follows:

./dineroIV –l1-isize 32K –l1-ibsize 32 –l1-dsize 16K –l1-dbsize 32 -informat d < trace.txt

This will simulate a 32 KB instruction cache with a 32 byte cache block size and a 16 KB data cache with a 32 byte block size.

(A) [18 points] Using the trace provided on the Canvas as input to DineroIV (trace.txt), model an instruction cache and a data cache with a combined total cache space of 32KB (for a split cache, assume a 16KB instruction cache and a 16KB data cache. The block size should be varied (32B, 64B and 128B) and the associativity should be varied (direct mapped and 8-way). Model both split (separate instruction and data caches) and shared (all accesses go to a single cache that holds both instructions and data) caches. These are a total of 12 simulations. No other parameters should be varied. Graph the results you get from these experiments and analyze in detail the trends you observe inyour graphs. Copy the trace file provided to the COE Linux system /local system to complete your homework.

(B) [12 points] In this interesting part, you will generate reference streams in Dinero's din format that achieve a specific purpose (you do not need to use the trace given in part A). When turning in your report, submit enough of the trace so that we can understand the reasoning and provide some evidence that you have been successful in achieving the desired goal whenever possible.

1. Assume that you have a 2-way set associative 16KB instruction cache with 32Byte block size and the replacement policy is LRU. Generate an address stream that touches every cache set exactly 5 times, producing 3 misses and 2 hits, but only 3 unique addresses access each set. Describe your approach in detail.

2. Given a n-way set associative 16KB instruction cache with 32Byte block size and the replacement policy as LRU (when n is great than 1), how would you generate one or multiple address stream to decode the associativity of the cache (i.e., value of n)? Of course, given an access stream, you can observe the hit and miss events from this n-way set associative cache.

Q2. (10 points)

"COZ: Finding Code that Counts with Causal Profiling" (SOSP 2015)
https://sigops.org/s/conferences/sosp/2015/current/2015-Monterey/printable/090-curtsinger.pdf

This paper is surprisingly novel and insightful. Most programmers focus on optimizing parts of their code that take up most of the execution time. This paper (surprisingly) shows why such an intuitive and long-standing practice might not be optimal. It introduces the concept of "virtual speed ups" where effect of speeding up a certain part of the program on the whole execution time is estimated. Please read this paper and list three major findings and why do you think they are important and correct!

Q3. (20 points)

ISCA 1990: "Improving Direct Mapped Cache Performance by the Addition of a Small Fully Associative Cache and Prefetch Buffers"
https://www.hpl.hp.com/techreports/Compaq-DEC/WRL-TN-14.pdf

Please read this seminal paper on "victim cache" and answer the following questions:

What is the basic idea behind victim caches? Why does this idea work?

Q4. (20 points)

ISCA 2007: Adaptive Insertion Policies for High Performance Caching (published in ISCA 2007)
https://people.csail.mit.edu/emer/papers/2007.06.isca.dip.pdf

Please read this seminal paper on dynamic insertion policy that is a small, yet very effective, tweak to the LRU cache replacement policy.

(a) What is the basic idea behind dynamic insertion policy? When and why does it work?

(b) What is "set-dueling"? What problem does it solve?

Q5. (10 points)

ISCA 2012: FLEXclusion: balancing cache capacity and on-chip bandwidth via flexible exclusion
https://dl.acm.org/doi/10.5555/2337159.2337196

Discuss the key ideas presented in this paper. What are the key benefits and downsides of the proposed approach?