EEC 483 Computer Organization, Spring 2020
Project #3: Cache simulator
Due: April 16 (Thursday)

In this programming assignment, you will need to develop a cache simulator and to measure the cache performance (miss ratio, etc.).

Input to the cache simulator is a list of memory addresses accessed. Outputs from the cache simulator are cache performance metrics such as cache miss ratio. It is preferred that you write this project in C or C++ but any other language is also acceptable. Please document your source code as you develop it. On the due day, a short video is desirable that briefly explains your code, parameters, as well as execution results.

Detailed description:

- The cache simulator will run based on a memory trace file that is a list of memory addresses referenced during the execution of a program. The cache simulator should output cache statistics such as miss ratio, etc.
- Assume the following cache parameters as the basis.
  - 64KB cache and 64B cache block
  - Single-level cache
  - Direct-mapped cache
  - Write-through policy
  - Data/instruction integrated cache
- However, your simulator will run the simulation a couple more times to test different cache configurations (a total of 12 cases):
  - Block size varies 16B~128B (4 cases)
  - Direct-mapped cache, 4-set associative cache, and 8-set associative cache (3 cases)
- Your job is to build a simulator that reads a memory trace and simulates the action of a cache. Your simulator should keep track of which memory blocks are loaded into cache. As it processes each memory reference from the trace, it should check to see if the corresponding memory address is in the cache or not.
- Of course, this is just a simulation of cache, so **you do not actually need to read and write data from memory**. **Just keep track of which blocks are in the cache**.
- Write-through policy is assumed in this assignment. **Since we do not actually read or write, write policy does not affect the cache performance such as miss ratio**.

Memory trace file:

- As an input to the cache simulator, you will use four memory traces – two integer programs and two floating point programs. You can access the four trace files (each about 15MB) from the class webpage:
  - 085.gcc.din (ref: https://www.spec.org/cpu92/cint92.html)
  - 022.li.din (ref: https://www.spec.org/cpu92/cint92.html)
  - 078.swm256.din (ref: https://www.spec.org/cpu92/cfp92.html)
  - 047.tomcatv.din (ref: https://www.spec.org/cpu92/cfp92.html)

- Each trace is a real recording of a running program, taken from the SPEC benchmarks. Each trace only consists of one million memory accesses taken from the beginning of each program.
- Each trace is a series of lines, each listing three fields per line: "accesstype address size/data." Address and size/data are hexadecimal. Accesstype is
  - **0 read**
  - **1 write**
  - **2 instruction fetch**
- Addresses are byte addresses. In other words, memory address 0000000c mean the 13th byte from the top of the memory. Every memory reference accesses a word data (4 bytes) regardless the size/data field. For example, "1 0000000c 0" denotes that a word data is written on memory address 0000000c. In other words, data at memory address 0000000c, 0000000d, 0000000e and 0000000f will be written. For this reason, all memory addresses in the trace files are word-aligned.
- For more information about the trace files, please visit SPEC (Standard Performance Evaluation Corporation) webpage: http://www.spec.org/benchmarks.html#cpu

Simulator requirements:

- You are free to structure and write your simulator in any reasonable manner. It is not mandatory but you may want to make your simulator flexible to accommodate other input parameters such as cache size and instruction/data integrated or separate cache as an argument.
- At the end of the simulation, the cache simulator should print out a few simple statistics like this:

```
Memory trace: 085.gcc.din
Total memory references in trace: 1002050 (read: 400000, write: 300000, fetch: 302050)
<Cache parameters>
Cache size: 64KB
Cache block size: 64 bytes
<Cache statistics>
Number of memory write events: 300000
Number of block placements: 1751
Number of block replacements: 983
…..
Cache miss ratio: 4.5%
```