Inf2C Computer Systems

Coursework 2: Cache Simulation

Deadline: Wed 23 Nov (Week 10) 16:00

Boris Grot, Priyank Faldu

The aim of this assignment is to write a parameterized cache simulator that reads a memory address trace and provides statistics about cache accesses and hits. The simulator is to be written in C and to get you started an outline of the simulator is provided in cache_sim.c file. You are strongly advised to read up on caches in the lecture notes/course textbook and commence work as soon as possible.

This is the second of the two assignments for the Inf2C Computer Systems course. It is worth 50% of the coursework marks and 20% of the overall course marks.

Please bear in mind the guidelines on plagiarism which are linked to from the online Undergraduate Year 2 Handbook.

1. Cache Simulator

Cache is a special memory that is used to store instructions and/or data so that the processor can access them at a very high speed. A detailed understanding of the cache dynamic behavior is usually obtained by software simulations rather than monitoring of hardware. In this exercise you are asked to investigate the memory access behavior of a benchmark. To do this, you have to write a reasonably flexible cache simulator. Assume the following specifications in designing the cache simulator:

Fixed parameters: The following parameters always stay the same:

(i) Memory addresses are 32 bits
(ii) Cache Organization: Unified (data and instruction cache blocks are kept in the same cache)

Variable parameters: The following parameters are passed as command-line arguments to the cache simulator:

(i) Cache Size: 256B to 8KB; powers of 2 only
(ii) Cache Block Size: 16, 32, 64, 128, or 256 bytes
(iii) Cache Mapping: Direct Mapped or Fully Associative
(iv) Cache Replacement Policy if Fully Associative Mapping: FIFO or LRU

For example, if the command-line parameters specify following options:
Cache size: 2KB, Block Size: 64B, Cache Mapping: Direct Mapped then the cache will have the following properties:

Parameters	**Values**
Cache Mapping	Direct Mapped
Replacement Policy	Not Applicable
Cache Size	2KB
Block Size	64B
Number of blocks	32 (2KB/64B)
Num bits for block offset	6 (64B block)
Num bits for index	5 (32 blocks)
Num bits for tag	21 (32-6-5)

2. Trace and Code files

You are provided with a memory trace file *mem_trace.txt* and a skeleton for the cache simulator in *cache_sim.c.*

The memory trace file consists of a sequence of memory accesses and it is an input to the cache simulator. For each memory access the trace file provides: a) whether it is an instruction or data access and b) 32-bit memory address in hex. The addresses in trace file are byte addresses and not the block addresses. An example of the trace file is as follows:

```
I  8cda3fa8
I  8158bf94
D 8cd94c50
I  8cd94d64
D 8cd94c54
```

where the letter "I" indicates that it is an instruction access, "D" denotes a data access, and is followed by a 32-bit address.

The file cache_sim.c already contains the code for:
1) reading the command-line parameters and initializing the corresponding variables
2) reading the trace file

For your convinience the output of the program is already provided in print_statistics(…) function. You need to populate the parameters of the function with appropriate values. Following is the full list of all variables you need to populate.

1. ***num_blocks***: total number of blocks in a cache
2. ***num_bits_for_block_offset***: number of bits required to access offset within a block
3. ***num_bits_for_index***: number of bits required to represent an index
4. ***num_bits_for_tag***: number of bits required for tag
5. ***result.instruction_accesses***: total number of cache accesses made by instructions
6. ***result.instruction_hits***: total number of hits incurred for instruction accesses
7. ***result.data_accesses***: total number of cache accesses made by data
8. ***result.data_hits***: total number of hits incurred for data accesses

Summary: In this assignment, you are required to write a cache simulator which reads a trace of memory references and reports cache statistics. The cache configuration is determined by parameters that are passed as command line arguments. Following parameters may be changed via command line parameters. cache size, block size, cache mapping, replacement policy if fully associative cache.

Hint: You can write your own trace files with easily predictable hit rates (e.g. think of a four access trace which uses four different addresses and has a 50% hit rate) to verify your simulator before using the trace (*mem_trace.txt*) provided to you. Your simulator will be tested with trace files different form the one that is already provided to you.

3. Compiling and Running the simulator

Compile the simulator on the DICE machines with the command:

```
gcc -o cache_sim  cache_sim.c
```

This creates an executable cache_sim. To run the simulator you need to pass command-line parameters as follows:

```
./cache_sim 256 64 LRU
```

where cache_sim is the executable name. 256 is cache size in bytes, 64 is block size in bytes and LRU tells that the cache is fully associative and replacement policy is LRU.

The parameters can take following values:
cache size : 256 – 8192 (has to be a power of 2)
block size : 16, 32, 64, 128 or 256
cache mapping : DM or FIFO or LRU
DM → Direct Mapped Cache
FIFO → Fully Associative Cache & FIFO Replacement Policy
LRU → Fully Associative Cache & LRU Replacement Policy

Make sure that your simulator compiles and runs on a DICE machine without errors and warnings.

Potential output looks as below for the following input

```
./cache_sim 1024 64 LRU
Num_Blocks:16
Bits_BlockOffset:6
Bits_Index:4
Bits_Tag:22
Total_Accesses:1465707
Total_Hits:1200000
Total_HitRate:81.87%
Instruction_Accesses:1104614
Instruction_Hits:1000000
Instruction_HitRate:90.53%
Data_Accesses:361093
Data_Hits:200000
Data_HitRate:55.39%
```

Note that the hit rate and hits provided in the sample output is arbitrary chosen and may be completely different than what you should get under the same parameters.

4. Submission

You should submit a copy of your simulator before 4pm on 23 Nov 2016, using the command

```
submit inf2c-cs 2 cache_sim.c
```

at a command prompt on a DICE machine. Unless there are special circumstances, late submissions are not allowed. Please consult the online Undergraduate Year 2 Handbook for further information on this.

5. Similarity Checking and Plagiarism

You must submit your own work. Any code that is not written by you must be clearly identified and explained through comments at the top of your files. Failure to do so is plagiarism. Detailed guidelines on what constitutes plagiarism can be found at:

http://web.inf.ed.ac.uk/infweb/admin/policies/guidelines-plagiarism

All submitted code is checked for similarity with other submissions using the MOSS system. MOSS has been effective in the past