CSCI 113 (Spring 2021)

Memory Heirarchy Simulation Programming (40 pts) due: 05/04 (T)

Write a simulation program for tracing memory operations described below. Please use either C or C++ for coding.

Hardware description:

- Memory hierarchy consists of higher-level (cache) and lower-level (main memory) modules, and we consider only data cache and data memory;
- Cache is 2way-set-associative, total cache data size is 16 words, one word per block, and the replacement policy used is LRU (Least Recently Used);
  Each cache block has 3 fields, i.e., valid bit, tag and data;
- When write_hit and write_miss, cache uses WB and no-write-allocate policies, respectively;
- Main memory size is 128 words and there are 128 memory blocks (1 word per block);
- Physical address is 7 bits wide and it points a block (a word) in the memory (0^{th} block ~ 127^{th} block);
  Physical address is divided into index and tag fileds;
- For the sake of simplicity, register file contains only 8 registers ($s0~$s7).

Input and output:

- Input is given as an object file (see attached), which consists of a series of MIPS load/store machine instructions;
- After executing each instruction, your program should display either "hit" or "miss" message;
  At the end of executing all the instructions in the object file, your program should display (in binary) the final contents of register file, cache and main memory.

Submission:

- Source code (.c or .cpp file); Please include a good global documentation and function head documentations in your source code (before compilation).
- Runtime output, e.g., a typescript file or screenshot (png, jpg or pdf);
  Output should be readable.

Details will be discussed in class and lab.

Initialization

Initially, all cache entries are empty and their valid bits are 0’s;

Initialize registers ($s0~$s7) in the register file with value 0’s;

Initialize memory contents with the following data:
mem[block_address] ← block_address + 5; e.g., mem[0] ← 5, … mem[127] ← 132

Testing instruction sequence:
you have to use the followng series of MIPS machine codes to submit output.

MIPS Assembly code	MIPS Machine code
`lw $s1,   4($zero)`	`100011 00000 10001 0000000000000100`
`lw $s2,  16($zero)`	`100011 00000 10010 0000000000010000`
`lw $s3,  32($zero)`	`100011 00000 10011 0000000000100000`
`lw $s4,  20($zero)`	`100011 00000 10100 0000000000010100`
`sw $s1,  80($zero)`	`101011 00000 10001 0000000001010000`
`sw $s2,  68($zero)`	`101011 00000 10010 0000000001000100`
`sw $s3,  76($zero)`	`101011 00000 10011 0000000001001100`
`sw $s4, 224($zero)`	`101011 00000 10100 0000000011100000`
`lw $s1,  36($zero)`	`100011 00000 10001 0000000000100100`
`lw $s2,  44($zero)`	`100011 00000 10010 0000000000101100`
`lw $s3,  16($zero)`	`100011 00000 10011 0000000000010000`
`lw $s4, 172($zero)`	`100011 00000 10100 0000000010101100`
`sw $s1,  20($zero)`	`101011 00000 10001 0000000000010100`
`sw $s2,  24($zero)`	`101011 00000 10010 0000000000011000`
`sw $s3,  36($zero)`	`101011 00000 10011 0000000000100100`
`sw $s4,  68($zero)`	`101011 00000 10100 0000000001000100`
`lw $s1,  36($zero)`	`100011 00000 10001 0000000000100100`
`lw $s2,  44($zero)`	`100011 00000 10010 0000000000101100`
`lw $s3,  16($zero)`	`100011 00000 10011 0000000000010000`
`lw $s4, 172($zero)`	`100011 00000 10100 0000000010101100`
`sw $s1,  96($zero)`	`101011 00000 10001 0000000001100000`
`sw $s2,  84($zero)`	`101011 00000 10010 0000000001010100`
`sw $s3,  92($zero)`	`101011 00000 10011 0000000001011100`
`sw $s4, 240($zero)`	`101011 00000 10100 0000000011110000`

* Macnie code file is placed in the Canvas.