CSCI 246 (Computer Arch.) Fall 2019 HW #5 (20 pts) Due: Nov. 07 (Th)

Consider the directory-based cache coherence protocol used in the cc-NUMA (DASH) machine.

A simulated system is shown in the diagram attached and the cache read/write policy follows.

Description of the simulated system:

- System consists of 4 MIPS-based SMP nodes, each of which consists of the following components: 2 scalar processors with a local cache each, 1 memory module, 1 directory
- Cache is direct-mapped and uses WB when write hit and no-write-allocate when write miss; Cache size (data only) is 4 words and a cache line (and memory line) size is 1 word (32 bits).
- Memory is globally addressed and the total memory size in the system is 64 words (16 words/node); Physical address is 6 bits (2 bits for index, 4 bits for tag), and the memory address shown in the diagram is word address and we ignore byte level addressing.
- Each directory also consists of 16 entries, one for each line (1 word) in the node memory.

Initial configuration:

- Initially, all caches are empty and their valid bits are 0's (invalid);
- Initially, local registers ($s1, $s2) in each processor have value 0's;
- Initially, memory contents are: word_address + 5; e.g., M[0] ← 5, M[1] ← 6, …, M[63] ← 68
- Initially, the status filed of each entry in the directory is "00" (uncached);

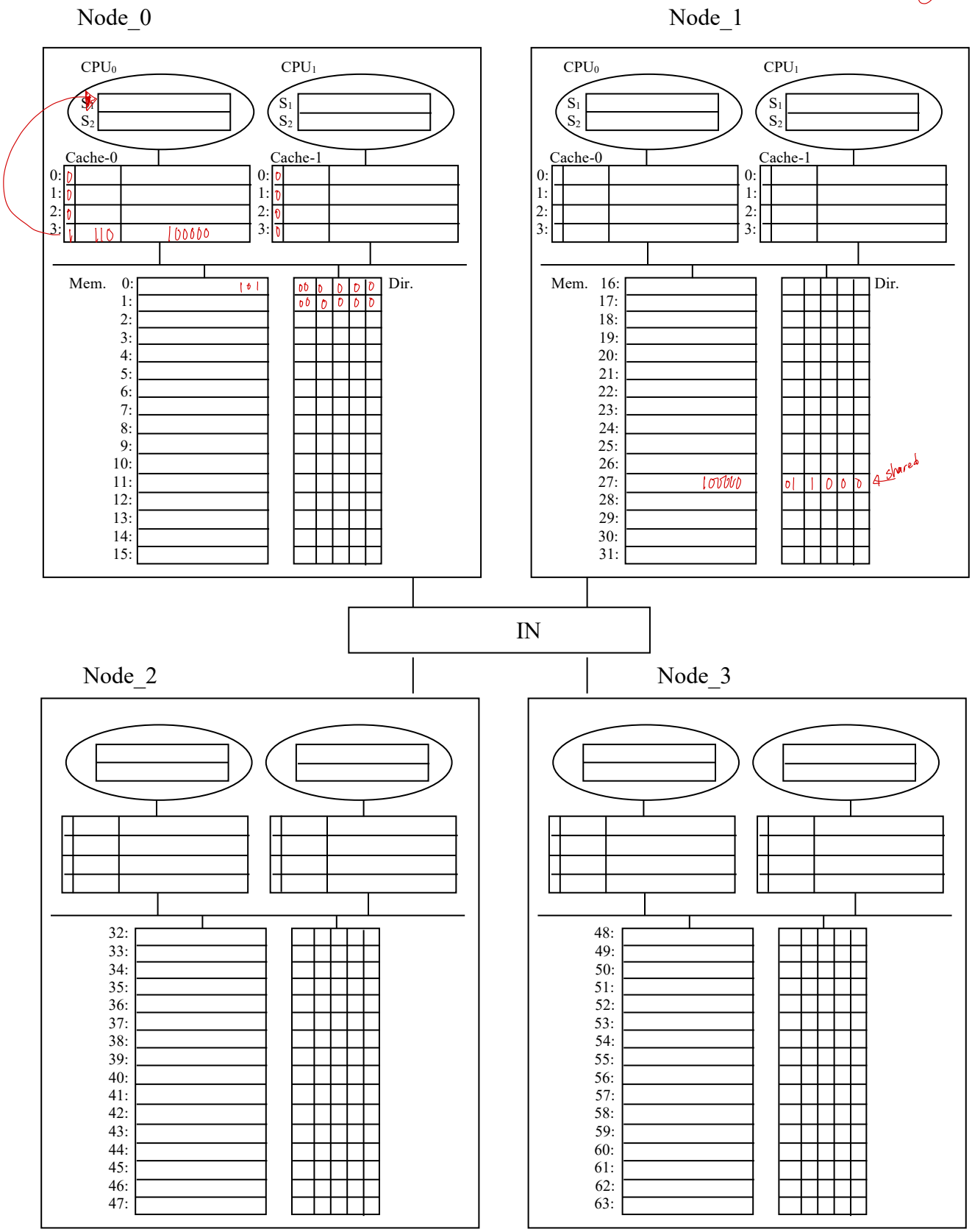
Show the contents of the simulated components (registers/caches/memory_modules/directories) and the total execution cycles after executing the following 8 consecutive instructions, which are in the MIPS 32bit ISA format. Please show only affected areas.

000 (node 0, CPU 0): lw $s1, 108($zero)
001 (node 0, CPU 1): lw $s2, 108($zero)
000 (node 0, CPU 0): sw $s1, 72($zero)
010 (node 1, CPU 0): lw $s1, 108($zero)
100 (node 2, CPU 0): lw $s1, 108($zero)
010 (node 1, CPU 0): lw $s1, 228($zero)
010 (node 1, CPU 0): sw $s1, 108($zero)
110 (node 3, CPU 0): lw $s1, 108($zero)

Corresponding machine codes (program input file)

000: 10001100000100010000000001101100
001: 10001100000100100000000001101100
000: 10101100000100010000000001001000
010: 10001100000100010000000001101100
100: 10001100000100010000000001101100
010: 10001100000100010000000011100100
010: 10101100000100010000000001101100
110: 10001100000100010000000001101100

Simulated cc-NUMA (DASH)



memory read (read-hit/read-miss)

read hit (direct-mapped)

1. Search local cache (local to each processor), i.e.,
   use MOD function for computing the cache address, and check valid bit (1) and tag;
   if found, access it (load it into the register); // this consumes 1 clock cycle.
   else, goto step2.

read miss

2. Search another cache in the local node;
   Do the same as in step1;
   if found, access it (load it into the cache & register); //this consumes 30 clock cycles.
   else, goto step3.
3. Search the home node's memory/directory;
   if directory indicates "uncached" or "shared" (means home memory has most recent or clean data), access it (load it into the local cache and register);
   //this consumes 100 clock cycles.
   else ("dirty"), goto step4.
4. Search all caches in the dirty node (use MOD function for the cache address);
   Do the operations described in Fig(a); //This consumes 135 clock cycles.
   "dirty → "shared"

In all steps above, manage the directories and valid/invalid bits of caches appropriately.

memory write (write-hit/write-miss)

write hit (WB)

1. Search local cache (local to each processor), i.e.,
   use MOD function for computing the cache address, and check valid bit (1) and tag;
   if found, get the exclusive right to access it first (see Fig(b) – using home directory, invalidate all cache copies if shared), and then update it with the content of the register; //this consumes 1 clock cycle; we ignore all other overheads.
   Home directory is updated with "dirty"; //possibly, dirty → dirty if local node is dirty node.
   else, goto step2.

write miss (no-W-alloc)

2. Update the home node memory with the content of the register; //this consumes 100 clock cycles; we ignore all other overheads.
   If directory indicates "uncached" → again, "uncached"
   "shared" → invalidate all shared cache copies; keep "shared"
   "dirty" → invalidate the dirty cache copy, "shared" now

"shared", but content is outdated (invalid)

Note that the above writing steps are based on the no-write-allocate when write miss.