CSCE614 HW4: Implementing Pseudo-LRU Cache Replacement Policy

11/3/2011

Baik Song An

Pseudo-LRU (PLRU)

- Finds an item that most likely has not been accessed very recently, given a set of items and a sequence of access events to the items
- Lower overheads compared to true LRU
  - 4-way cache set
    - True LRU: 6 bits needed per set
    - Pseudo-LRU : 3 bits needed per set

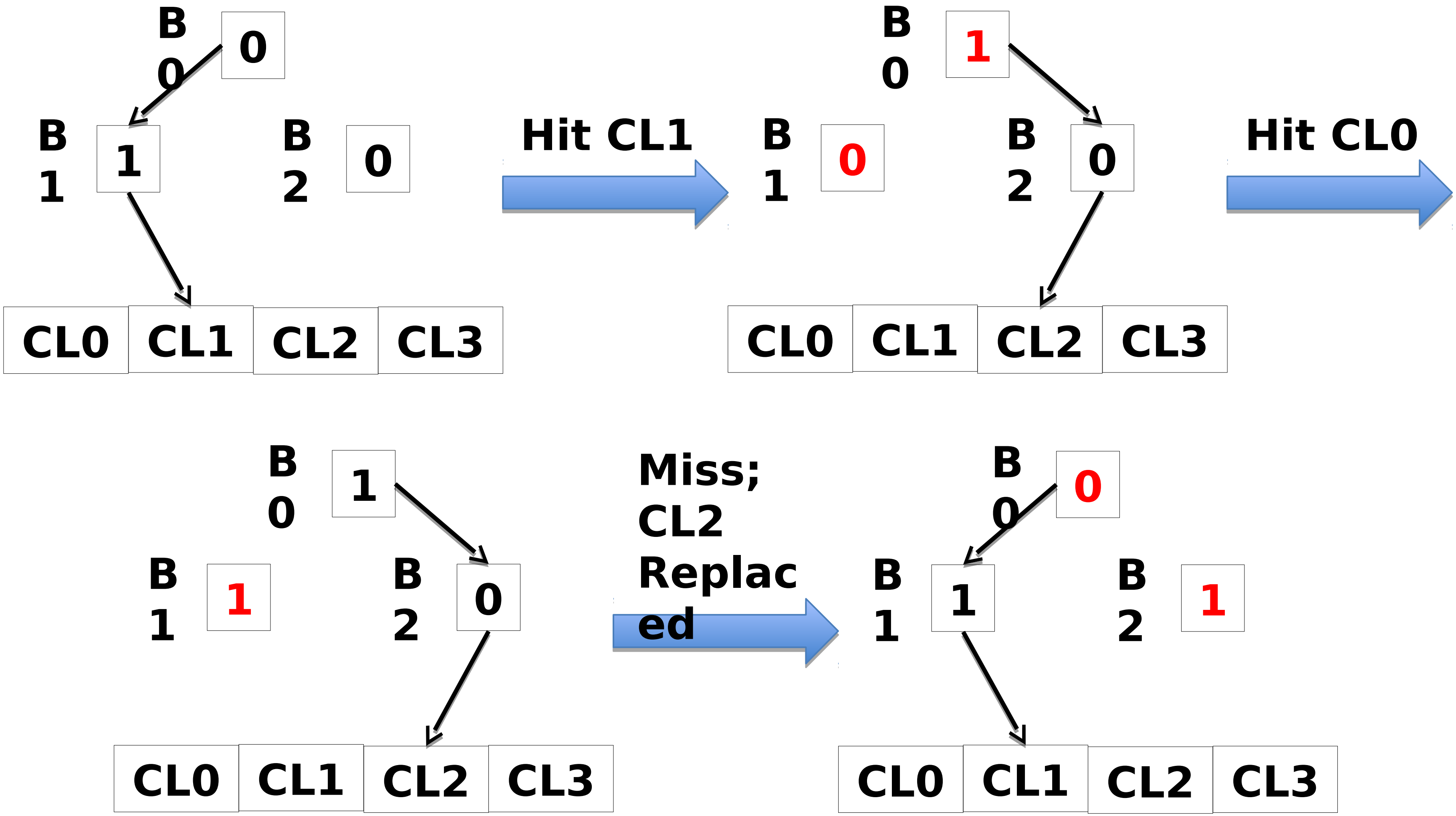
Algorithm

- Use of a binary search tree for each set
  - Represented by bit flags
- Each bit represents one branch point in a binary decision tree
  - 1 represents
    - The left side has been referenced more recently than the right side
    - “Go right to find a pseudo-LRU element.”
  - 0 vice-versa

Algorithm (Cont’d)

- Finding a pseudo-LRU element
  - Traverse the tree according to the values of the flags
- Updating the tree with an access to an item N
  - Traverse the tree to find N
  - During the traversal, set the node flags to denote the direction that is opposite to the direction taken

Example


B0
0
B1
1
B2
0
CL0
CL1
CL2
CL3
Hit CL1
B0
1
B1
0
B2
0
CL0
CL1
CL2
CL3
Hit CL0
B0
1
B1
1
B2
0
CL0
CL1
CL2
CL3
Miss; CL2 Replaced
B0
0
B1
1
B2
1
CL0
CL1
CL2
CL3

Rules

State	Replace
00x	CL0
01x	CL1
1x0	CL2
1x1	CL3

(x: Don’t Care)

Reference	Next State
CL0	11_
CL1	10_
CL2	0_1
CL3	0_0

(_: Unchanged)

Implementation

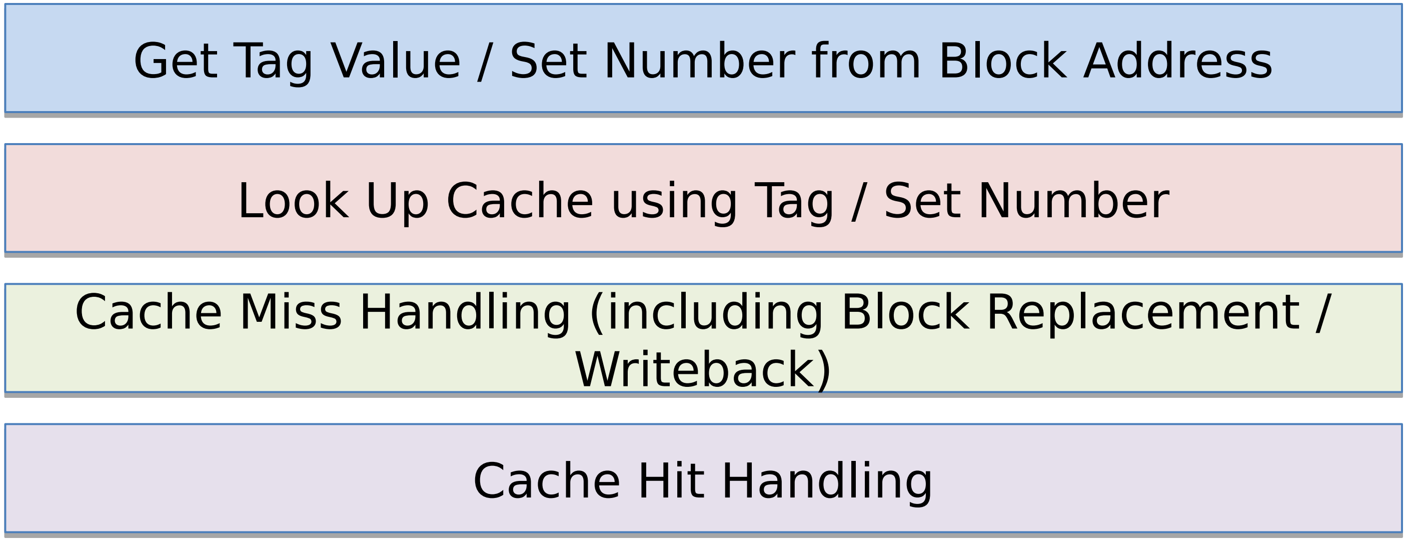
- Adding a variable in a cache set structure definition
  - Unsigned long: 32 bits in x86 Linux
  - Enough for handling 4-way cache sets

```
struct cache_set_t {

...

unsigned int PLRU_bits;
};
```

cache_access()

- Access a cache to perform load or store operations

```
cache_access() {
```



Get Tag Value / Set Number from Block Address

Look Up Cache using Tag / Set Number

Cache Miss Handling (including Block Replacement / Writeback)

Cache Hit Handling

```
}
```

Miss Handling in PLRU

- Add a new `case` statement in `switch()` for choosing a new policy
- Figure out which block should be evicted
  - Need to determine `repl` (pointer to the replaced block)
  - First, find if there is an invalid block
    - blk->status & CACHE_BLK_VALID
  - If all blocks are valid, choose one by traversing the binary search tree
- Update PLRU_bits after miss handling

Miss Handling in PLRU (Cont’d)

```
switch(cp->policy) {
case LRU:
case FIFO:
case Random:
case PLRU:
    {
      Determine repl (invalid block / tree traversal)
    }
}
…

      Update PLRU_bits
```

Cache Hit in PLRU

- Update PLRU_bits at the end
  - The same as in miss handling
- Do nothing for cache_fast_hit

Tip(s)

Start as early as possible!

Questions: baiksong@cse.tamu.edu

Good luck!