Fully Associative Write-Back
(Cache Simulation)

Ulises Méndez Martínez

Universidad Autónoma de Guadalajara

ulisesmdzmtz@gmail.com

August 13, 2016

Overview

1. Cache Simulation
   - Fully Associative
   - Write Back
2. Code Implementation
   - Helper Macros
   - Debug Simulation
   - Search in cache operation
   - Utils operations
   - Read operation
   - Write operation
   - Update operation

Fully Associative

Fully Associative Cache

A cache where data from any address can be stored in any cache location. The whole address must be used as the tag. All tags must be compared simultaneously (associatively) with the requested address and if one matches then its associated data is accessed.

Write Back

Write Back

Write back is a storage method in which data is written into the cache every time a change occurs, but is written into the corresponding location in main memory only at specified intervals or under certain conditions.

Starting code

Example (Helpers Macros for bit manipulation)

```
#define shiftL(n, k) ((n)<<(k))
#define shiftR(n, k) ((n)>>(k))
#define BITS_MEM    (20)
#define BITS_CACHE  (8)
#define BITS_DATA   (8)
```

Commands Simulation

Example (Debug Simulation)

```
//helper function to get bits, len shold be at least 1
inline int get_bits(int n, int len, int from) {
  int bits = (shiftL(1,len)) - 1;
  return (shiftR(n,from)&bits);
}
... // within the main function
    // Each command is 32 bits,
    rnd = rand();
    // 0 - bit indicates operation type 0:read 1:write
    cmd = get_bits(rnd,1,0);
    // 20...1 bits for address
    address = get_bits(rnd,BITS_MEM,1);
    // 28...21 bits for data
    dat = get_bits(rnd,BITS_DATA,BITS_MEM + 1);
```

Search in cache operation

Example (Search in cache)

```
// Inline function to search address within cachedA array
// In real life all operations are done in parallel
inline int cache_search(int address) {
  int sz = shiftL(1,BITS_CACHE);
  for(int i=0; i<sz; ++i)
    if(cachedAddr[i] == address) return i;
  return -1;
}
... // Within the main function
    //Perform operation according
    // search in cache
    pos = cache_search(address);
```

Utils operations

Example (Utils operations)

```
// We know the position read data cached
inline int read_cache(int index) {
  return cachedData[index];
}
// Read from an specific address
inline int read_raw(int address) {
  return rawMemory[address];
}
// We know the position write data in cache
inline void write_cache(int index, int data) {
  cachedData[index] = data;
}
// Write to an specific address
inline void write_raw(int address, int data) {
  rawMemory[address] = data;
}
```

Read operation

Example (Read operation)

```
if(cmd == read) {
  puts("Operation: [Read]");
  // Was found in cache, just read this value
  if(pos >= 0) {
    printf("Address 0x%x found ... index(%d)\n",address,pos);
    printf("Data in cache = 0x%x\n",read_cache(pos));
  }else { // Read from raw memory
    printf("Address 0x%x not Found!\n",address);
    printf("Raw data read = 0x%x\n",rawMemory[address]);
    // And insert new data into cache
    update_cache(address, read_raw(address));
  }
}
```

Write operation

Example (Write operation)

```
... { // write
  puts("Operation: [Write]");
  // Was found in cache, just update this value
  if(pos >= 0) {
    Log("Address 0x%x found ... index(%d)\n",address,pos);
    Log("Update ... New[0x%x]\n",read_cache(pos),dat);
    write_cache(pos, dat);
  }else{
    // Write directly to raw
    Log("Address 0x%x not Found!\n",address);
    Log("Write raw data, New[0x%x]\n",dat);
    write_raw(address, dat);
    // And insert new data into cache
    update_cache(address, read_raw(address));
  }
}
```

Update operation

Example (Update operation)

```
// Check if write - back is needed
inline void update_cache(int address, int data) {
    // get next position in the circular queue
    tail = (tail + 1) % shiftL(1,BITS_CACHE);
    // The cached was already used need to write back
    // the values to the raw memory
    if(cachedAddr[tail] != -1) {
      printf("Address␣0x%x␣left␣cache\n",cachedAddr[tail]);
      printf("Backup␣Data[0x%x]␣to␣Raw␣memory\n",read_cache(tai
      write_raw(cachedAddr[tail], read_cache(tail));
    }
    // Update the cache address and data
    printf("New␣address␣0x%x␣stored␣in␣cache␣at␣pos␣%d\n",addre
    cachedAddr[tail] = address;
    write_cache(tail, data);
}
```

Q & A