SWE3005

SUNGKYUNKWAN UNIVERSITY

Introduction to Computer Architectures (Fall 2019)

PA3: Cache Simulator

Prof. Jinkyu Jeong (jinkyu@skku.edu)

TA – Minwoo Ahn (minwoo.ahn@csi.skku.edu)

TA – Sunghwan Kim (sunghwan.kim@csi.skku.edu)

Computers Systems and Intelligence Laboratory (http://csi.skku.edu)

2019. 12. 03.

Overview: What to do?

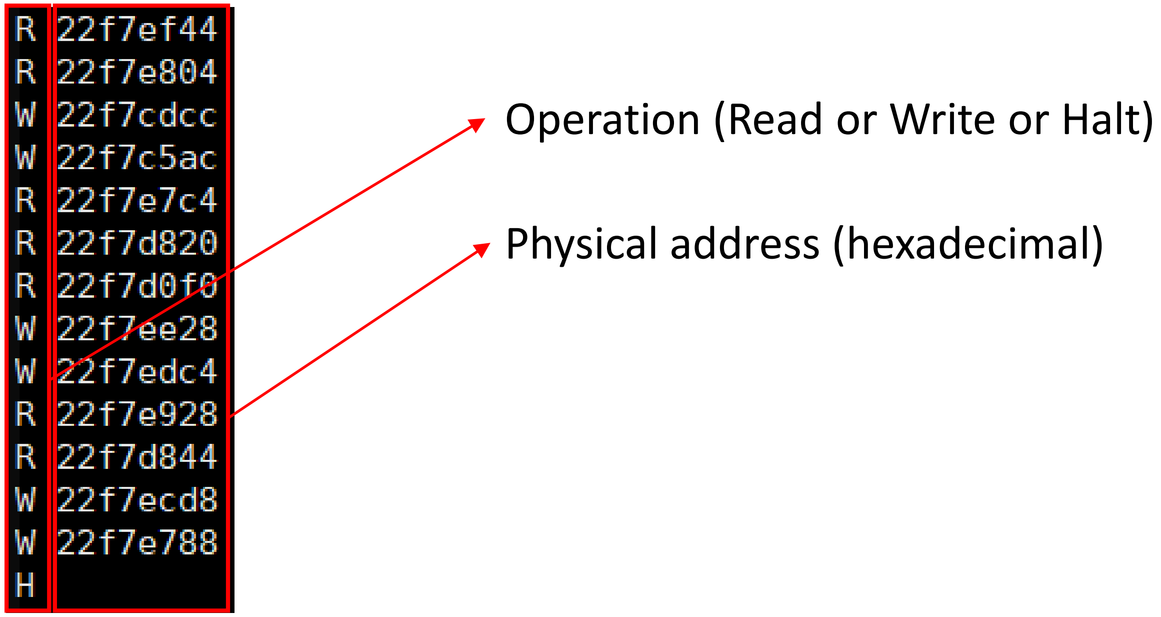
- Goal
  - To help you understand the internal of cache
- Implement cache simulator in C language
  - Skelaton code is provided
- Environment
  - Recommend to implement your code on Linux

Overview: Given Files

- Configuration (cache.config)
  - Configuration of cache is listed
  - level, size, way, entry, replacement, write policy, access cycle
- Input (cache.input)
  - Random (or sequential) read and write request stream
- Skelaton (main.c, mem.h)
  - You have to implement your cache in these files
  - Do not modify original code

Input Files

- Input files are set of [read | write] operation with 32bit physical address (file must end with ‘H’, halt)
  - Read example (R 22f7c744)
  - Write example (W 22f7c54c)
  - Every read and write operation requests 4 bytes word data

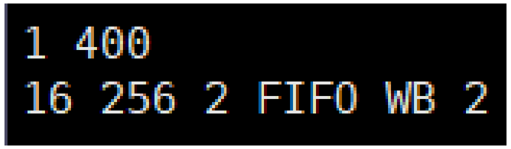


```
R 22f7ef44
R 22f7e804
W 22f7cdcc
W 22f7c5ac
R 22f7e7c4
R 22f7d820
R 22f7d0f0
W 22f7ee28
W 22f7edc4
R 22f7e928
R 22f7d844
W 22f7ecd8
W 22f7e788
H
```

Operation (Read or Write or Halt)
Physical address (hexadecimal)

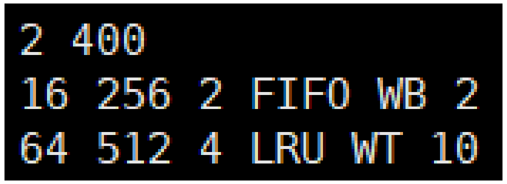
Configuration File

- 1st line: # of cache level, memory access time (cycle)
- 2nd line (level 1 cache): cache size (byte), # of entries, # of ways, replacement policy, write policy, cache access time (cycle)
- 3rd line (level 2 cache): same configuration as the second line
- Replacement policy
  - LRU: Least Recently Used
  - FIFO: First-in-First-out
- Write policy
  - WB: Write Back
  - WT: Write Through



```
1 400
16 256 2 FIFO WB 2
```

(a) 1 level cache config

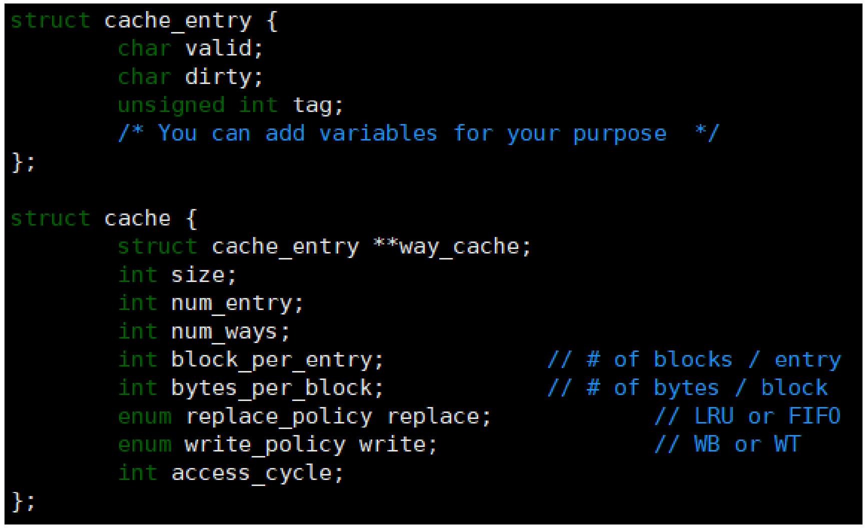


```
2 400
16 256 2 FIFO WB 2
64 512 4 LRU WT 10
```

(b) 2 level cache config

Skelaton Code

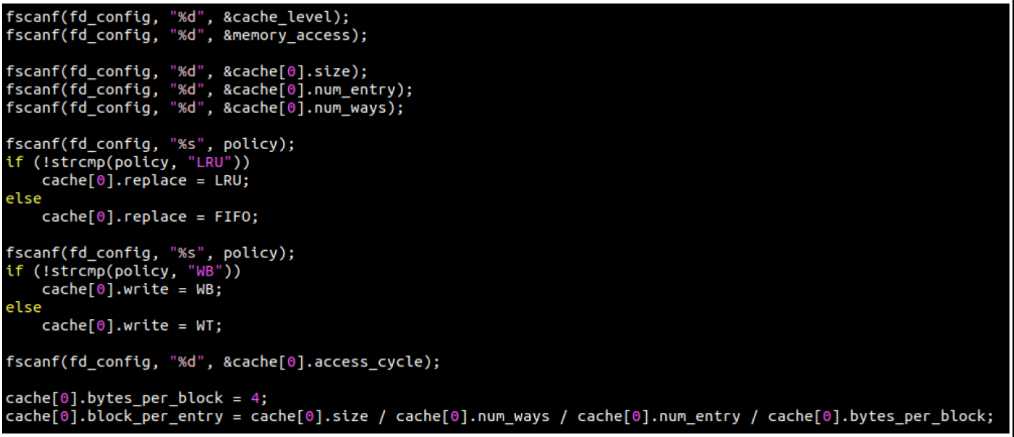
- mem.h
  - struct cache_entry
    - Valid bit, dirty bit, tag
    - You can add new variables for your own purpose
  - struct cache
    - Variable: size, num_entry, num_ways, replace_policy, write_policy, access_cycle
    - Usage: way_cache[way #][entry #]



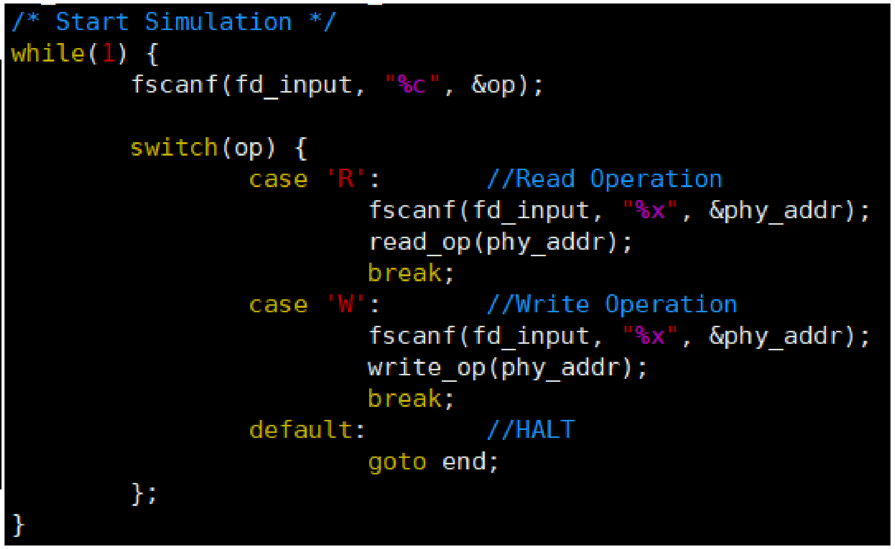
```
struct cache_entry {
        char valid;
        char dirty;
        unsigned int tag;
        /* You can add variables for your purpose  */
};

struct cache {
        struct cache_entry **way_cache;
        int size;
        int num_entry;
        int num_ways;
        int block_per_entry;            // # of blocks / entry
        int bytes_per_block;            // # of bytes / block
        enum replace_policy replace;            // LRU or FIFO
        enum write_policy write;                // WB or WT
        int access_cycle;
};
```

main.c – main()

- Read configuration file
- Read input file
- Cache operation
- Print result

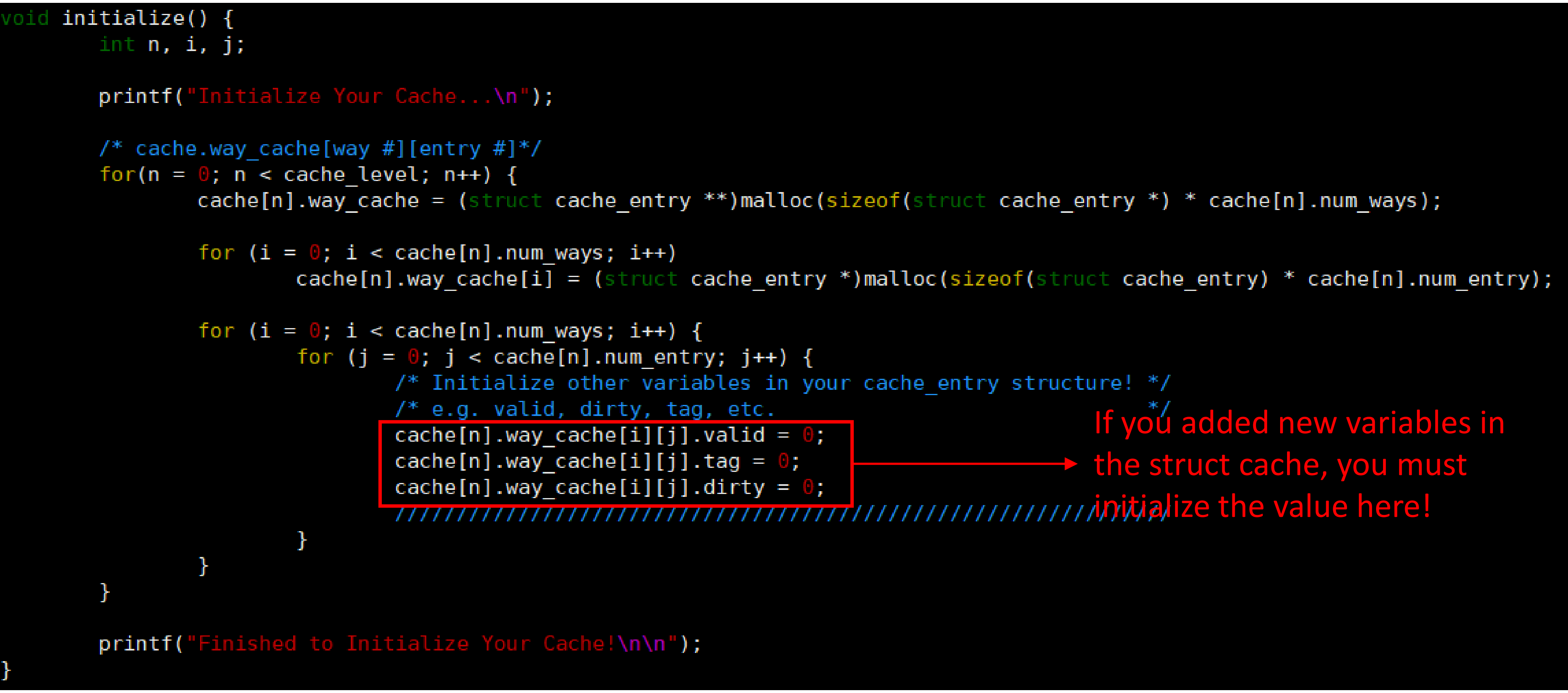


```
fscanf(fd_config, "%d", &cache_level);
fscanf(fd_config, "%d", &memory_access);

fscanf(fd_config, "%d", &cache[0].size);
fscanf(fd_config, "%d", &cache[0].num_entry);
fscanf(fd_config, "%d", &cache[0].num_ways);

fscanf(fd_config, "%s", policy);
if (!strcmp(policy, "LRU"))
    cache[0].replace = LRU;
else
    cache[0].replace = FIFO;

fscanf(fd_config, "%s", policy);
if (!strcmp(policy, "WB"))
    cache[0].write = WB;
else
    cache[0].write = WT;

fscanf(fd_config, "%d", &cache[0].access_cycle);

cache[0].bytes_per_block = 4;
cache[0].block_per_entry = cache[0].size / cache[0].num_ways / cache[0].num_entry / cache[0].bytes_per_block;
```



```
/* Start Simulation */
while(1) {
        fscanf(fd_input, "%c", &op);

        switch(op) {
                case 'R':       //Read Operation
                        fscanf(fd_input, "%x", &phy_addr);
                        read_op(phy_addr);
                        break;
                case 'W':       //Write Operation
                        fscanf(fd_input, "%x", &phy_addr);
                        write_op(phy_addr);
                        break;
                default:        //HALT
                        goto end;
        };
}
```

main.c – initialize()

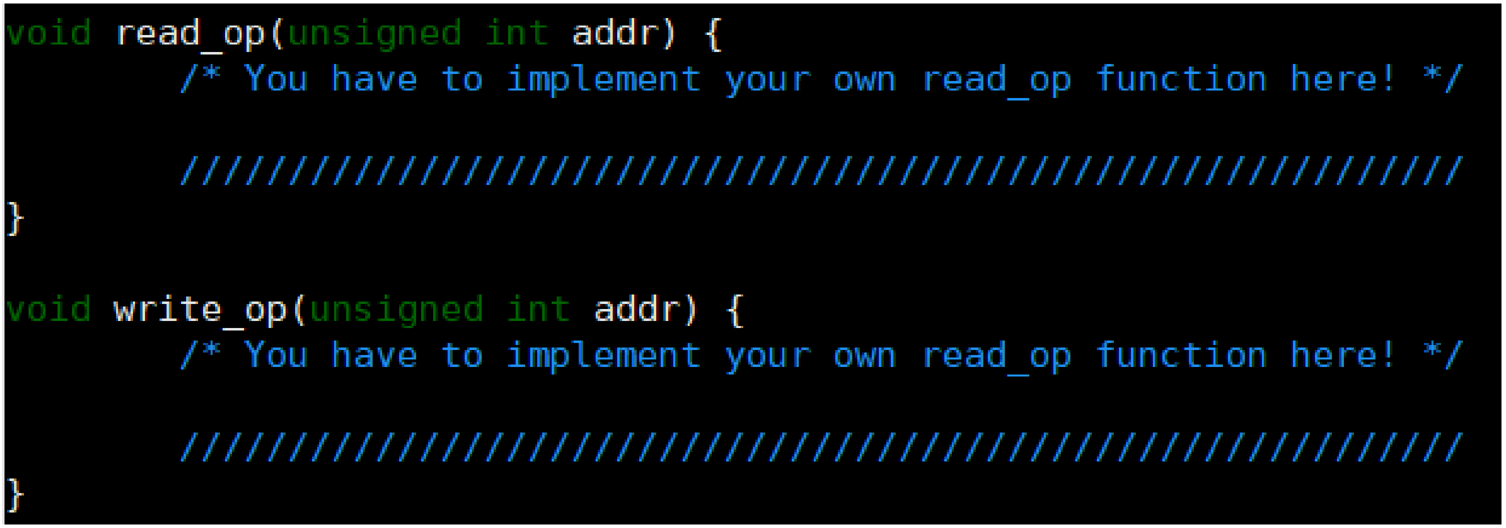


```
void initialize() {
        int n, i, j;

        printf("Initialize Your Cache...\n");

        /* cache.way_cache[way #][entry #]*/
        for(n = 0; n < cache_level; n++) {
                cache[n].way_cache = (struct cache_entry **)malloc(sizeof(struct cache_entry *) * cache[n].num_ways);

                for (i = 0; i < cache[n].num_ways; i++)
                        cache[n].way_cache[i] = (struct cache_entry *)malloc(sizeof(struct cache_entry) * cache[n].num_entry);

                for (i = 0; i < cache[n].num_ways; i++) {
                        for (j = 0; j < cache[n].num_entry; j++) {
                                /* Initialize other variables in your cache_entry structure! */
                                /* e.g. valid, dirty, tag, etc.                             */
                                cache[n].way_cache[i][j].valid = 0;
                                cache[n].way_cache[i][j].tag = 0;
                                cache[n].way_cache[i][j].dirty = 0;
                                ///////////////////////////////////////////////////////////////
                        }
                }
        }

        printf("Finished to Initialize Your Cache!\n\n");
}
```

If you added new variables in the struct cache, you must initialize the value here!

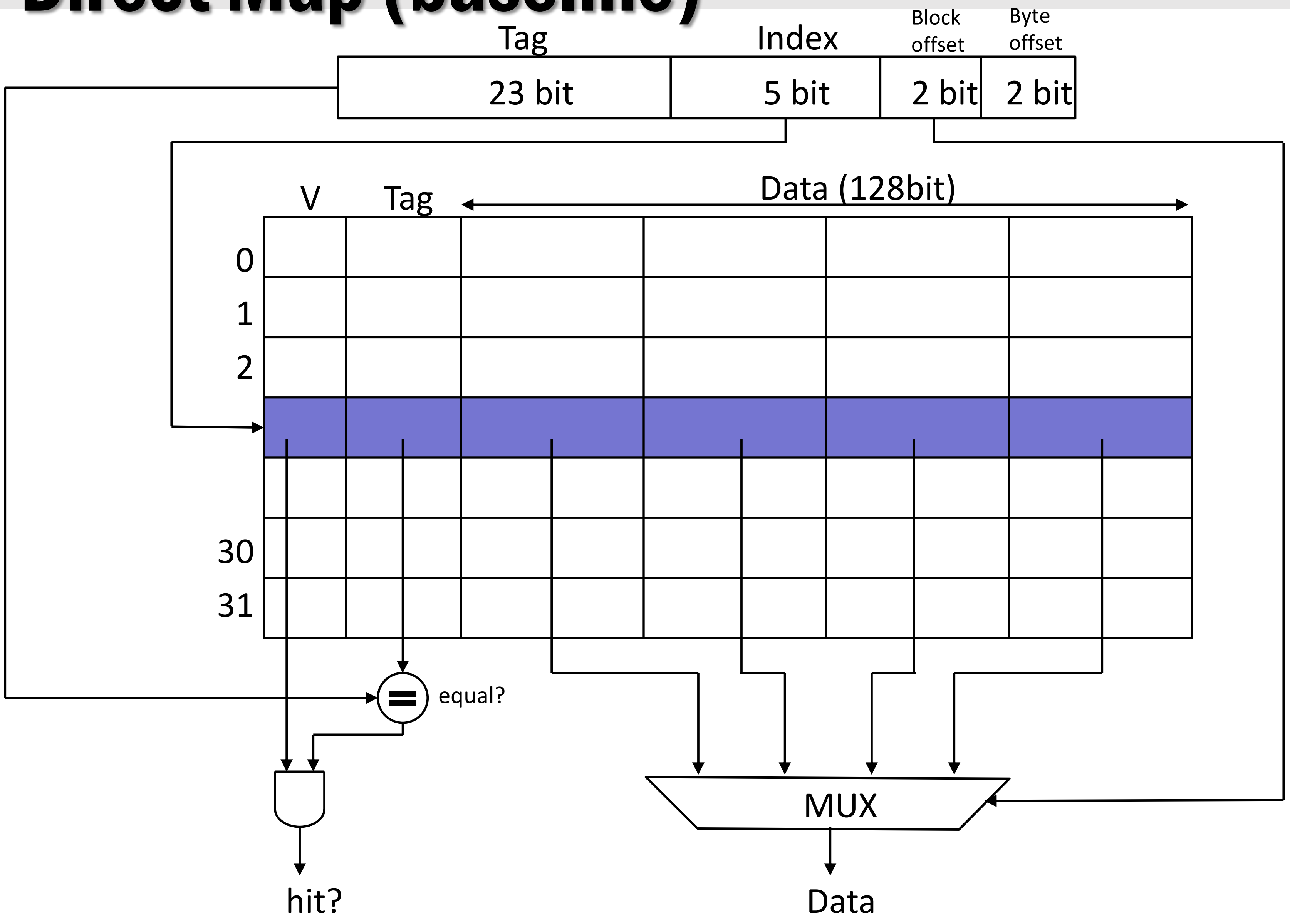
main.c – read_op(), write_op()

- You must implement these three functions
  - read_op(), write_op()
    - You should count the hit and miss each level of cache
    - At the miss, you should implement cache load and eviction
    - If evicted entry is dirty, write back or write through the entry
    - You should count the total access time (cycle)

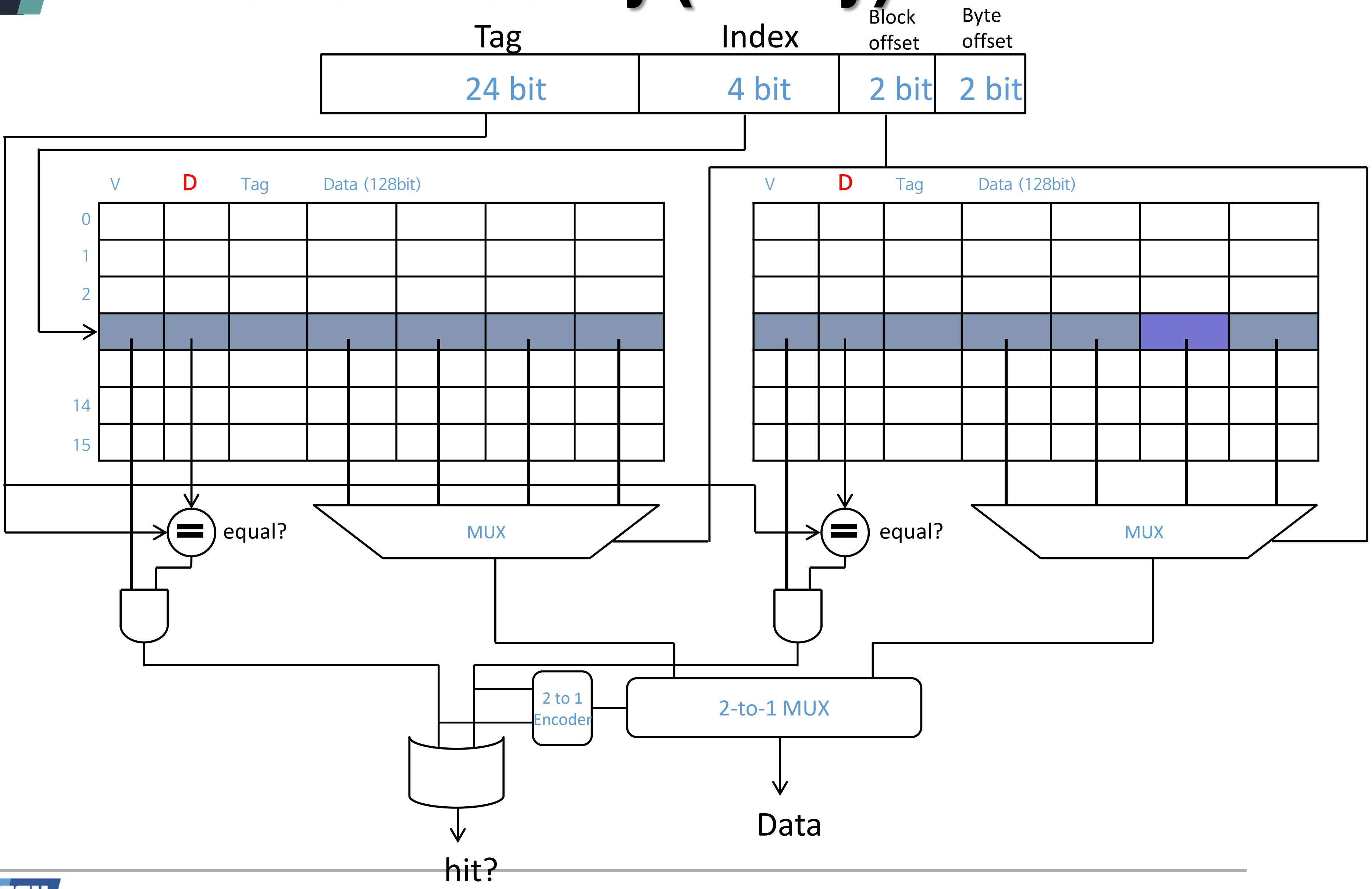


```
void read_op(unsigned int addr) {
        /* You have to implement your own read_op function here! */

        ///////////////////////////////////////////////////////////
}

void write_op(unsigned int addr) {
        /* You have to implement your own read_op function here! */

        ///////////////////////////////////////////////////////////
}
```

Direct Map (baseline)



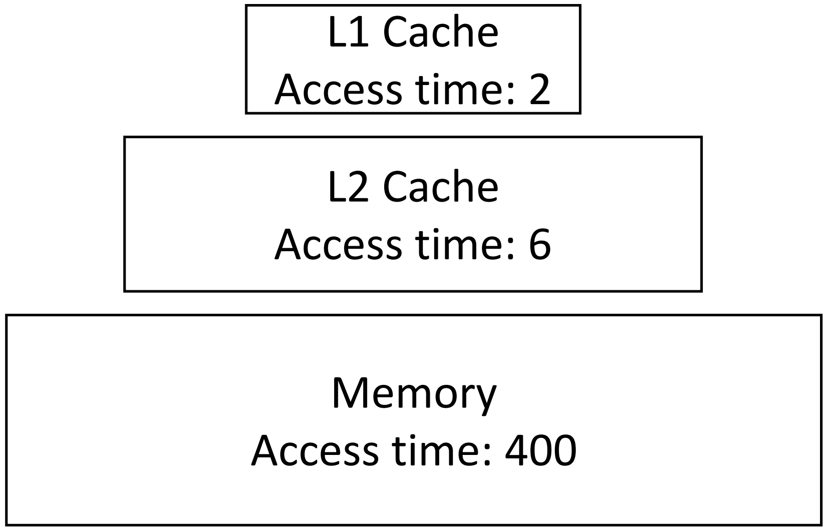
Cache Associativity (2-way)



PA3 – Cache Simulator

- Implement cache simulator in C language
  - read_op(), write_op()
- L1 cache & L2 cache must satisfy the inclusion property
  - No addition cycle
- L1 cache & L2 cache have same number of data blocks
- Write policy with write-allocation
- No write buffer

2 Level Cache



- L1 hit: 2 cycles
- L1 miss, L2 hit: 2+6 cycles
- L1 miss, L2 miss: 2+6+400 cycles
- Write through or dirty bit entry eviction need cycle addition

Submission

- Compress your main.c, and mem.h files only
  - Do not change file name
  - Without subdirectories
  - Submitted file name should be YourStudentID.zip
    - ex) 2018000000.zip
  - Please follow this format
- Upload your zip file to I-Campus *Assignments* bulletin
- DO NOT COPY
  - If you have any questions please use your TAs
- Due date: TBD
  - -20% per day for delayed submission

Tips

- Execution
  - To run the code, type $make command under skeleton directory
  - “simulator” executable file will be made
  - Run the executable without argument
    - $./simulator
- Lecture notes will help your assignment

Example

cache.config

```
2 400
16 2 1 LRU WT 2
32 2 2 LRU WB 10
```

Tag	Index	Block offset	Byte offset
28 bit	1 bit	1 bit	2 bit

Index	Valid	Dirty	Tag	Cache line	
				Data1	Data2
0	0	0	0	0	0
1	0	0	0	0	0

WT L1 cache

Tag	Index	Block offset	Byte offset
28 bit	1 bit	1 bit	2 bit

I	V	D	T	Cache line		I	V	D	T	Cache line	
				Data1	Data2					Data1	Data2
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	1	0	0	0	0	0

WB L2 cache

Example

cache.config

```
2 400
16 2 1 LRU WT 2
32 2 2 LRU WB 10
```

1. W 22F7CF44

Index	Valid	Dirty	Tag	Cache line	
				Data1	Data2
0	1	0	22F7CF4	xx	aa
1	0	0	0	0	0

WT L1 cache

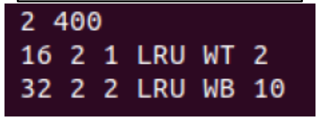
L1 miss + L2 miss = 2 + 10 + 400 cycles

I	V	D	T	Cache line		I	V	D	T	Cache line	
				Data1	Data2					Data1	Data2
0	1	0	22F7CF4	xx	xx	0	0	0	0	0	0
1	0	0	0	0	0	1	0	0	0	0	0

WB L2 cache

Example

cache.config



```
2 400
16 2 1 LRU WT 2
32 2 2 LRU WB 10
```

1. W 22F7CF44

Index	Valid	Dirty	Tag	Cache line	
				Data1	Data2
0	1	0	22F7CF4	xx	aa
1	0	0	0	0	0

WT L1 cache

Write Through = 10 cycles

I	V	D	T	Cache line		I	V	D	T	Cache line	
				Data1	Data2					Data1	Data2
0	1	0	22F7CF4	xx	aa	0	0	0	0	0	0
1	0	0	0	0	0	1	0	0	0	0	0

WB L2 cache

Example

cache.config

```
2 400
16 2 1 LRU WT 2
32 2 2 LRU WB 10
```

2. R 22F7CF40

WT L1 cache

Index	Valid	Dirty	Tag	Cache line	
				Data1	Data2
0	1	0	22F7CF4	xx	aa
1	0	0	0	0	0

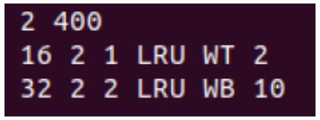
L1 hit = 2 cycles

I	V	D	T	Cache line		I	V	D	T	Cache line	
				Data1	Data2					Data1	Data2
0	1	1	22F7CF4	xx	aa	0	0	0	0	0	0
1	0	0	0	0	0	1	0	0	0	0	0

WB L2 cache

Example

3. R 22F7CA3C



```
2 400
16 2 1 LRU WT 2
32 2 2 LRU WB 10
```

Index	Valid	Dirty	Tag	Cache line	
				Data1	Data2
0	1	0	22F7CF4	xx	aa
1	1	0	22F7CA3	xx	xx

WT L1 cache

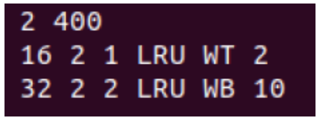
L1 miss + L2 miss = 2 + 10 + 400 cycles

I	V	D	T	Cache line		I	V	D	T	Cache line	
				Data1	Data2					Data1	Data2
0	1	1	22F7CF4	xx	aa	0	0	0	0	0	0
1	1	0	22F7CA3	xx	xx	1	0	0	0	0	0

WB L2 cache

Example

4. W 22F7CA30



```
2 400
16 2 1 LRU WT 2
32 2 2 LRU WB 10
```

Index	Valid	Dirty	Tag	Cache line	
				Data1	Data2
0	1	0	22F7CA3	bb	xx
1	1	0	22F7CA3	xx	xx

WT L1 cache

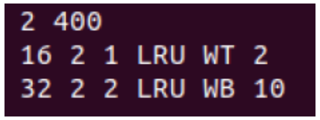
L1 miss + L2 miss = 2 + 10 + 400 cycles

I	V	D	T	Cache line		I	V	D	T	Cache line	
				Data1	Data2					Data1	Data2
0	1	1	22F7CF4	xx	aa	0	1	0	22F7CA3	xx	xx
1	1	0	22F7CA3	xx	xx	1	0	0	0	0	0

WB L2 cache

Example

4. W 22F7CA30



```
2 400
16 2 1 LRU WT 2
32 2 2 LRU WB 10
```

Index	Valid	Dirty	Tag	Cache line	
				Data1	Data2
0	1	0	22F7CA3	bb	xx
1	1	0	22F7CA3	xx	xx

WT L1 cache

Write Through = 10 cycles

I	V	D	T	Cache line		I	V	D	T	Cache line	
				Data1	Data2					Data1	Data2
0	1	1	22F7CF4	xx	aa	0	1	0	22F7CA3	bb	xx
1	1	0	22F7CA3	xx	xx	1	0	0	0	0	0

WB L2 cache

Example

5. R 22F7CF40

```
2 400
16 2 1 LRU WT 2
32 2 2 LRU WB 10
```

Index	Valid	Dirty	Tag	Cache line	
				Data1	Data2
0	1	0	22F7CF4	xx	aa
1	1	0	22F7CA3	xx	xx

WT L1 cache

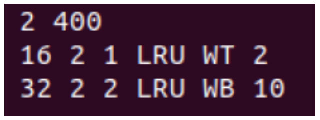
L1 miss + L2 hit = 2 + 10 cycles

I	V	D	T	Cache line		I	V	D	T	Cache line	
				Data1	Data2					Data1	Data2
0	1	1	22F7CF4	xx	aa	0	1	1	22F7CA3	bb	xx
1	1	0	22F7CA3	xx	xx	1	0	0	0	0	0

WB L2 cache

Example

6. W 22F7C420



```
2 400
16 2 1 LRU WT 2
32 2 2 LRU WB 10
```

Index	Valid	Dirty	Tag	Cache line	
				Data1	Data2
0	1	0	22F7CF4	xx	aa
1	1	0	22F7CA3	xx	xx

WT L1 cache

L1 miss + L2 miss + L2 replacement = 2 + 10 + 400 + 400 cycles

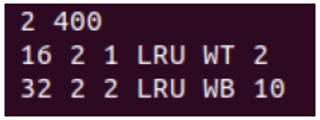
I	V	D	T	Cache line		I	V	D	T	Cache line	
				Data1	Data2					Data1	Data2
0	1	1	22F7CF4	xx	aa	0	1	1	22F7CA3	bb	xx
1	1	0	22F7CA3	xx	xx	1	0	0	0	0	0

Write back

WB L2 cache

Example

6. W 22F7C420


```
2 400
16 2 1 LRU WT 2
32 2 2 LRU WB 10
```

Index	Valid	Dirty	Tag	Cache line	
				Data1	Data2
0	1	0	22F7C42	aa	xx
1	1	0	22F7CA3	xx	xx

WT L1 cache

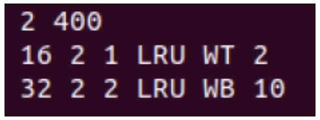
L1 miss + L2 miss + L2 replacement = 2 + 10 + 400 + 400 cycles

I	V	D	T	Cache line		I	V	D	T	Cache line	
				Data1	Data2					Data1	Data2
0	1	1	22F7CF4	xx	aa	0	1	0	22F7C42	xx	xx
1	1	0	22F7CA3	xx	xx	1	0	0	0	0	0

WB L2 cache

Example

6. W 22F7C420



```
2 400
16 2 1 LRU WT 2
32 2 2 LRU WB 10
```

Index	Valid	Dirty	Tag	Cache line	
				Data1	Data2
0	1	0	22F7C42	aa	xx
1	1	0	22F7CA3	xx	xx

WT L1 cache

Write through = 10 cycles

I	V	D	T	Cache line		I	V	D	T	Cache line	
				Data1	Data2					Data1	Data2
0	1	1	22F7CF4	xx	aa	0	1	0	22F7C42	aa	xx
1	1	0	22F7CA3	xx	xx	1	0	0	0	0	0

WB L2 cache

Example

7. W 22F7C420

```
2 400
16 2 1 LRU WT 2
32 2 2 LRU WB 10
```

Index	Valid	Dirty	Tag	Cache line	
				Data1	Data2
0	1	0	22F7C42	bb	xx
1	1	0	22F7CA3	xx	xx

WT L1 cache

L1 hit + Write through= 2 + 10 cycles

I	V	D	T	Cache line		I	V	D	T	Cache line	
				Data1	Data2					Data1	Data2
0	1	1	22F7CF4	xx	aa	0	1	1	22F7C42	bb	xx
1	1	0	22F7CA3	xx	xx	1	0	0	0	0	0

WB L2 cache

Example

8. R 22F7C424

```
2 400
16 2 1 LRU WT 2
32 2 2 LRU WB 10
```

Index	Valid	Dirty	Tag	Cache line	
				Data1	Data2
0	1	0	22F7C42	bb	xx
1	1	0	22F7CA3	xx	xx

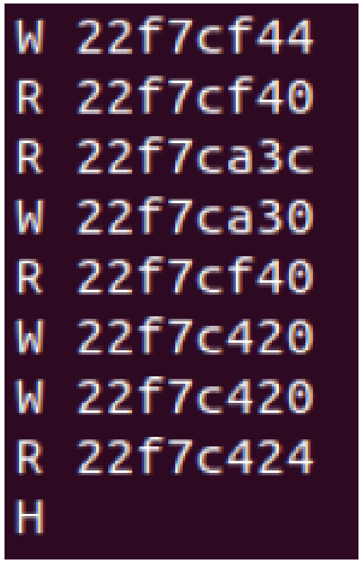
WT L1 cache

L1 hit = 2 cycles

I	V	D	T	Cache line		I	V	D	T	Cache line	
				Data1	Data2					Data1	Data2
0	1	1	22F7CF4	xx	aa	0	1	1	22F7C42	bb	xx
1	1	0	22F7CA3	xx	xx	1	0	0	0	0	0

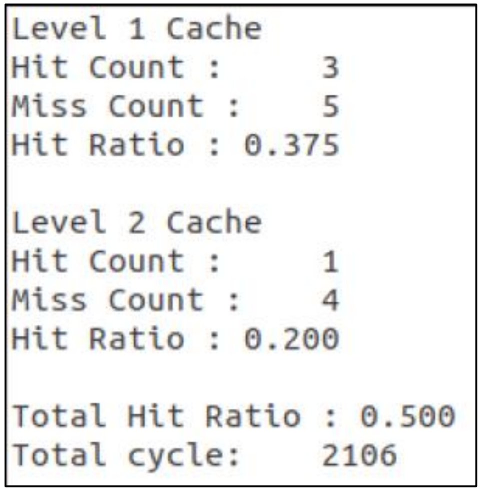
WB L2 cache

Example – result



```
W 22f7cf44
R 22f7cf40
R 22f7ca3c
W 22f7ca30
R 22f7cf40
W 22f7c420
W 22f7c420
R 22f7c424
H
```

cache.input



```
Level 1 Cache
Hit Count :      3
Miss Count :     5
Hit Ratio : 0.375

Level 2 Cache
Hit Count :      1
Miss Count :     4
Hit Ratio : 0.200

Total Hit Ratio : 0.500
Total cycle:     2106
```

cache.output

Questions

- You are free to ask questions to TAs
  - Please send email before visit the office
  - Email: minwoo.ahn@csi.skku.edu / sunghwan.kim@csi.skku.edu
  - Office: Semiconductor Building #400509