Discussion 10 : Cache

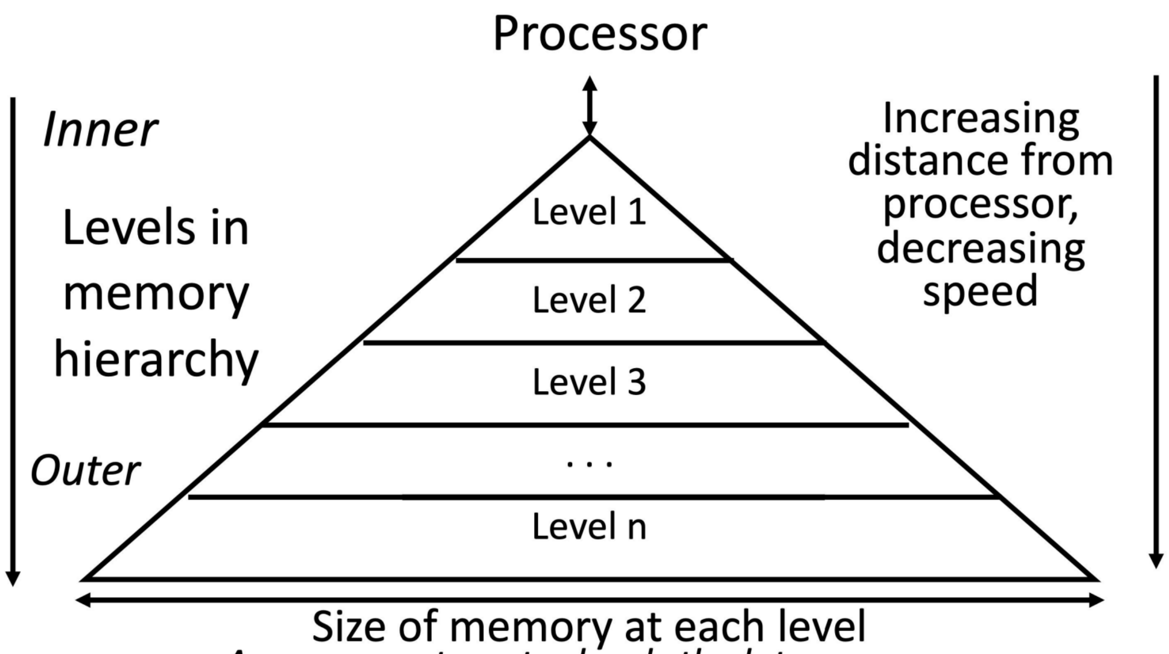
Linjie Ma

Memory Hierarchy

上海科技大学 ShanghaiTech University

Why Memory Hierarchy?

- Huge memory works slow
- Small memory works fast
- Use memory hierarchy can make CPU get most data faster



As we move to outer levels the latency goes up and price per bit goes down.

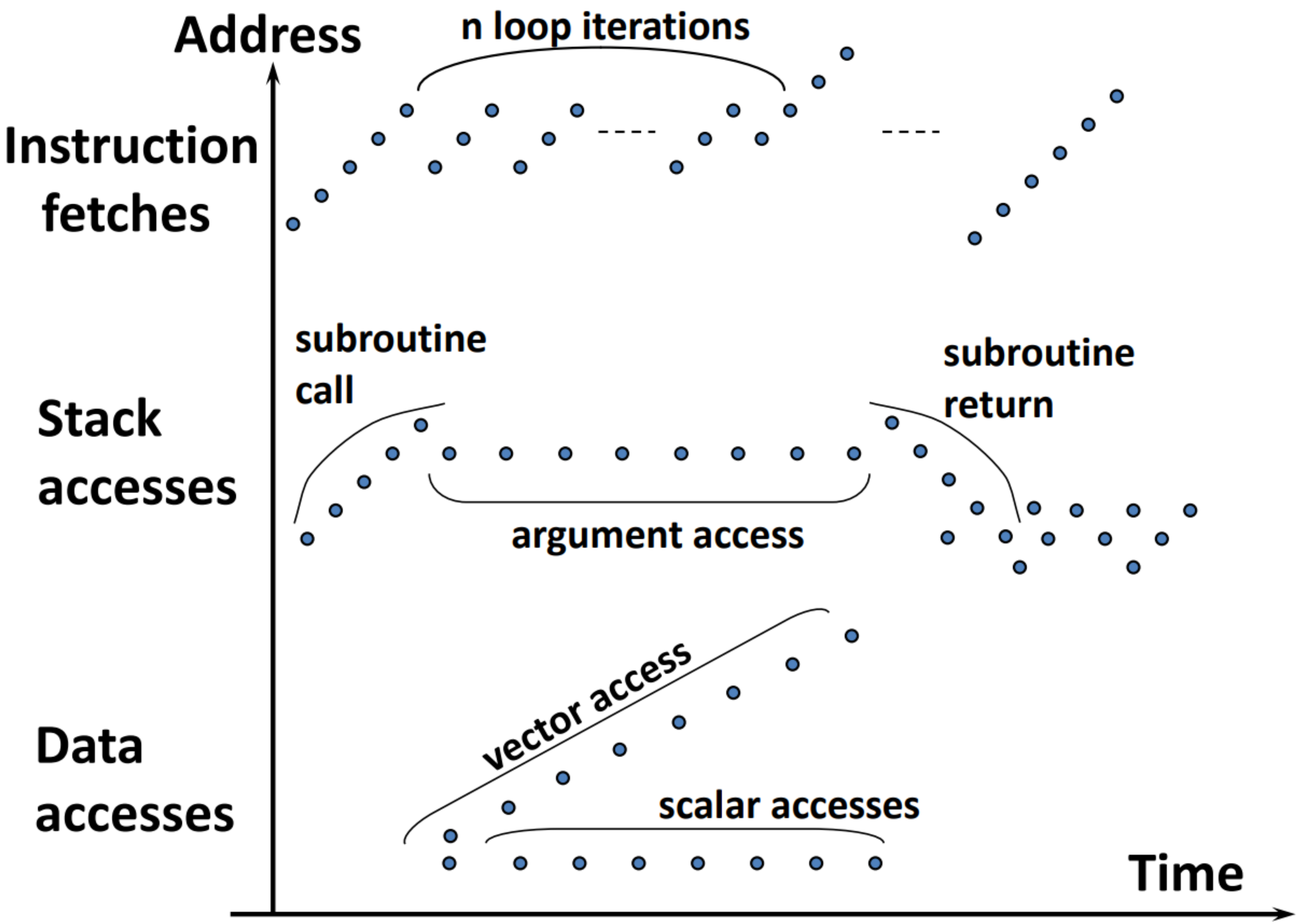
立志成才报国裕民

Locality

- *Temporal Locality* (locality in time)
  – If a memory location is referenced, then it will tend to be referenced again soon

- *Spatial Locality* (locality in space)
  – If a memory location is referenced, the locations with nearby addresses will tend to be referenced soon

Memory Reference Patterns



Cache Design

Processor Address (32-bits total)

Tag	Set Index	Block offset

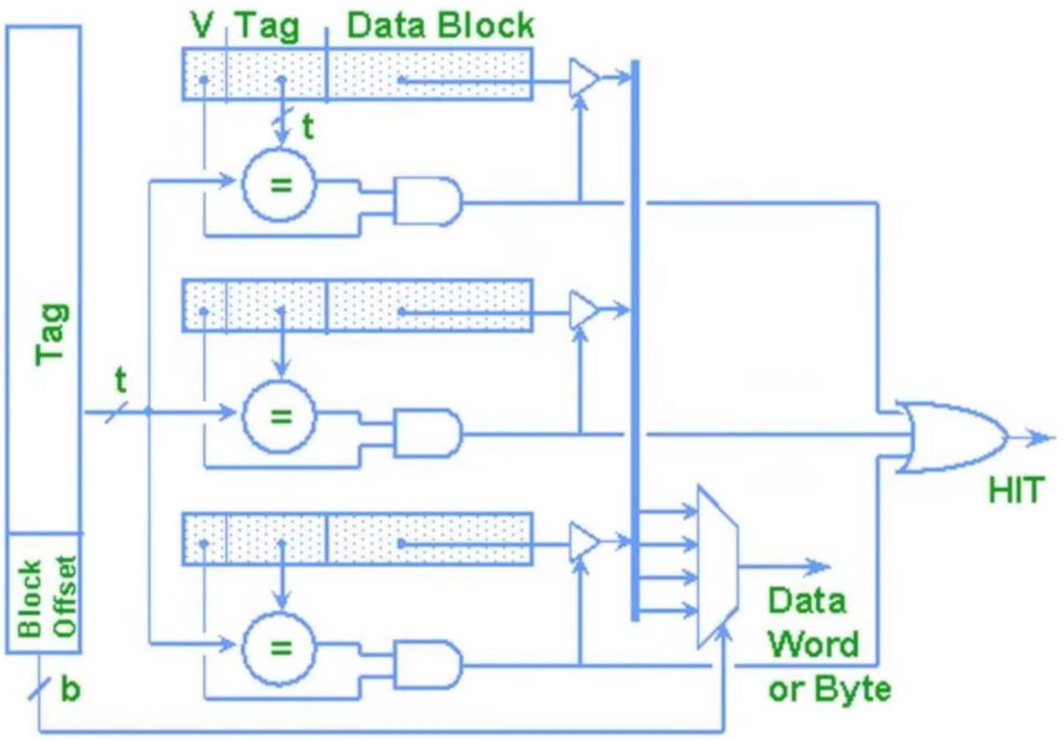
- Size of Cache Block(or Cache Line) → Block offset
- Number of sets → Set index
- Remain bits → Tag
- Associativity(or Way) → Number of cache blocks within a set

Cache size = # of sets * Associativity * size-of-block

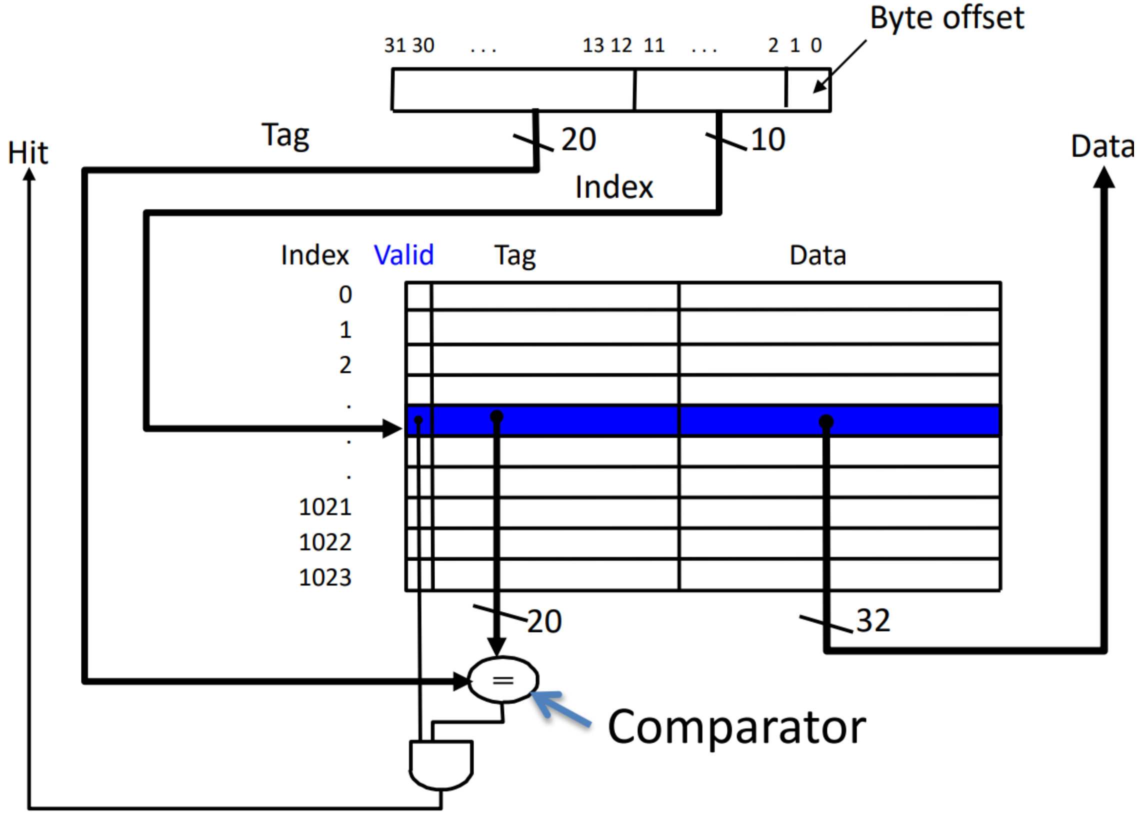
Cache Design

上海科技大学
ShanghaiTech University

Fully Associative Cache



Direct Mapped Cache



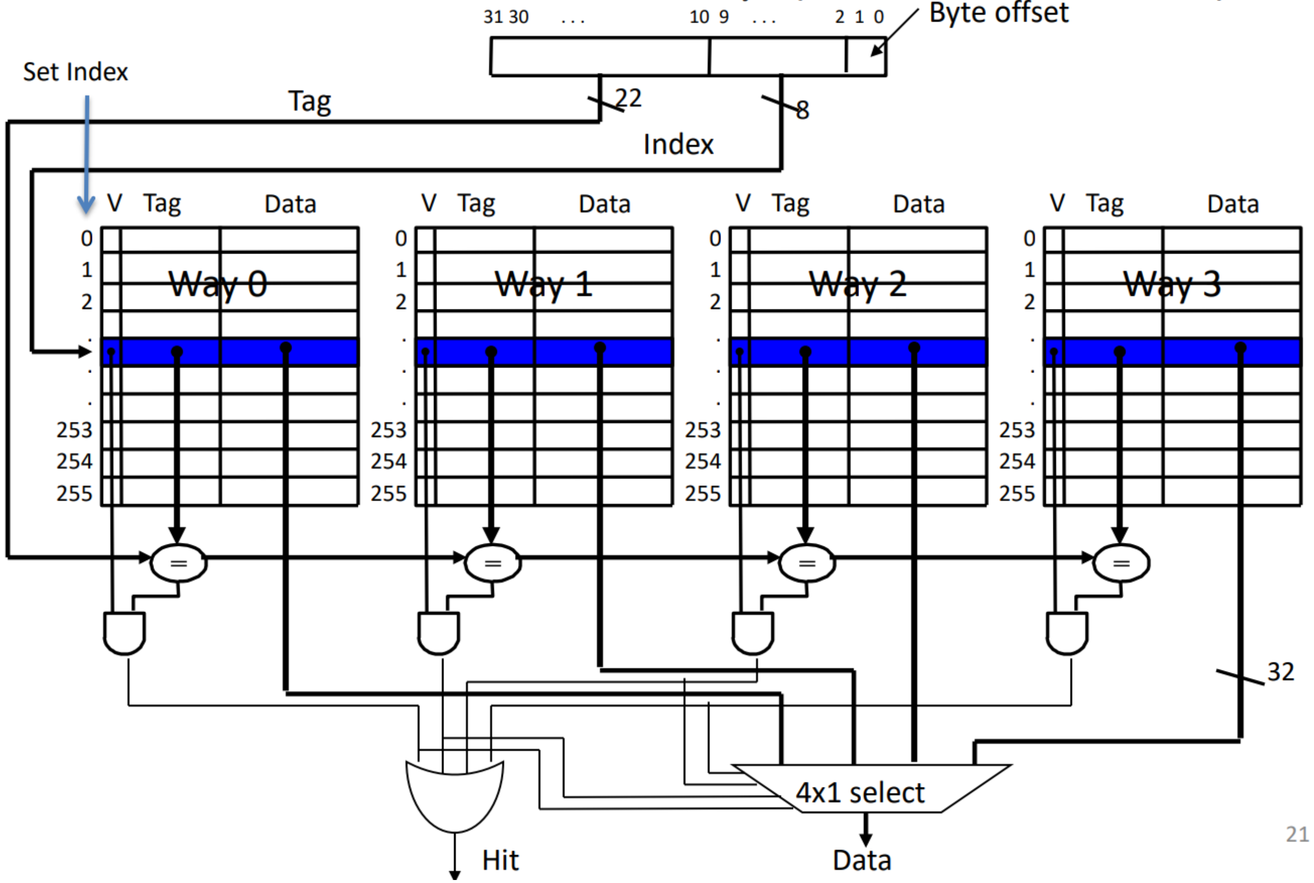
立志成才报国裕民

Cache Design

N-way Set Associative

Four-Way Set-Associative Cache

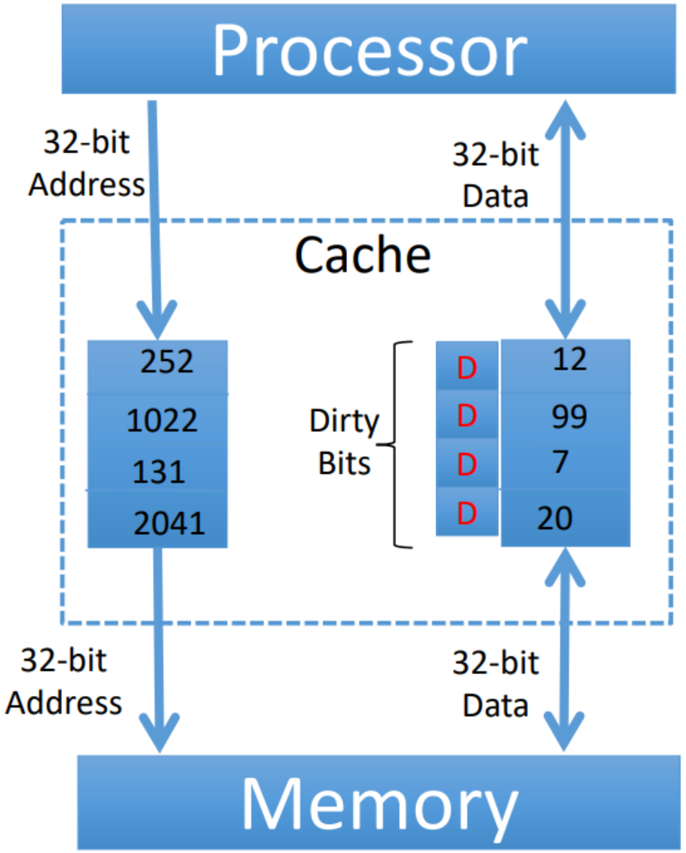
- 2^8 = 256 sets each with four ways (each with one block)



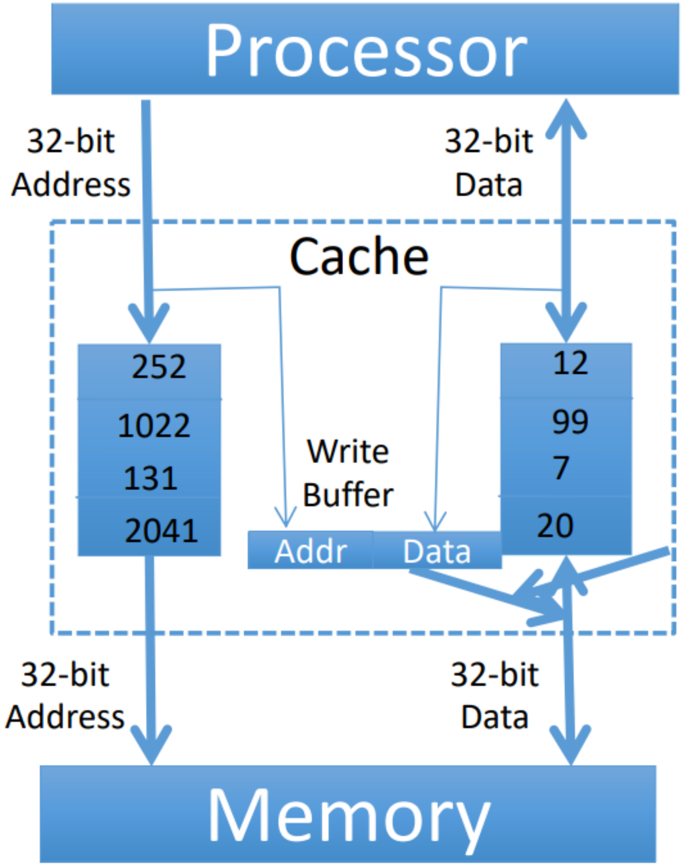
Cache Design

上海科技大学 ShanghaiTech University

Write Back Policy



Write Through Policy



立志成才报国裕民

Real exam problem(1)

(a) We have an 10-bit address space and a n-way set associative cache. After a while, the entire cache has the following state:

Index	Tag 1	Valid 1	Tag 2	Valid 2
0b00	0b1011	1	0b1101	1
0b01	0b0011	1	0b0010	1
0b10	0b1110	1	0b0111	0
0b11	0b1111	0	0b0001	0

Calculate the bit width of tag, index, and block offset.

Tag	Index	Offset

Solution: 4, 2, 4.

(b) Calculate the following parameters of the cache in (b):

Associativity: __________

Block size (in Bytes): __________

Cache size (in Bytes): __________

Solution:

1. Associativity is 2
2. Block size is 16 Bytes;
3. Cache size is 128 Bytes;

Cache performance

上海科技大学
ShanghaiTech University

- Hit rate : fraction of accesses that hit in the cache
- Miss rate : 1- Hit rate
- Miss penalty : time to replace a cache block from lower level in memory hierarchy to cache
- Hit time : time to access cache memory

- AMAT : Average Memory Access Time (AMAT) is the average time to access memory considering both hits and misses in the cache

- AMAT = Hit time + Miss rate * Miss penalty

立志成才报国裕民

Cache replacement

上海科技大学
ShanghaiTech University

- LRU : Replace the cache block which is accessed least recently
- FIFO : Replace the cache block which is loaded into cache earliest
- Random : Replace a random cache block

Cache Misses:

- Compulsory : First access to block impossible to avoid
  - Compulsory misses are misses that will occur no matter how you change the cache
- Capacity : Cache cannot contain all blocks accessed by the program
  - Capacity misses are misses that will still occur even if the cache were fully associative with LRU replacement
- Conflict : Multiple memory locations mapped to the same cache location
  - Conflict misses are misses that would not occur if the cache were fully associative with LRU replacement

Real exam problem(2)

上海科技大学
ShanghaiTech University

(a) We have an 8-bit address space and a 2-way set associative cache with properties as follows:

1. Cache size is 32 Bytes;
2. Block size is 8 Bytes;

Calculate the bit width of tag, index, and offset bits.

TAG	Set Index	Block Offset

Solution: 4, 1, 3.

(b) We will access the data of addresses as follows. Fill in the blanks. It is about T/I/O (tag/index/offset, write down the value in decimal), classify the access as a Hit, Miss or Replace. (each line worth 1 pt.)

Address	T/I/O	Hit, Miss or Replace
0b00000100		
0b00000101		
0b01101000		
0b11001000		
0b01101000		
0b11011101		

Solution:

Address	T/I/O	Hit, Miss or Replace
0b00000100	0/0/4	Miss
0b00000101	0/0/5	Hit
0b01101000	6/1/0	Miss
0b11001000	12/1/0	Miss
0b01101000	6/1/0	Hit
0b11011101	13/1/5	Replace

立志成才报国裕民

Real exam problem(2)

(c) Assume we have a single-level, 1 KiB direct-mapped L1 cache, whose bit width of tag, index, and offset bits are 22, 6, 4 separately. An integer is 4 bytes. The array is block-aligned. Given the following C source code, what is the hit rate?

```c
#define LEN 512

int array[LEN];
int main() {
    for (int i = 0; i < LEN; i += 128) {
        array[i] = 0;
    }
    for (int i = LEN - 128; i >= 0; i -= 128) {
        array[i] = 0;
    }
    return 0;
}
```

Solution: $1/4$

Real exam problem(3)

(b) Suppose the cache has the following settings:

Cache levels	1
Block size	16 bytes
Number of sets	4
Cache size	128 bytes
Block replacement policy	LRU

1. Is this cache direct-mapped, set-associative, or fully-associative? If it is associative, also write down its associativity.

____

____

2. Suppose the memory addresses are 32-bit long, what is the length of the tag field?

____

____

Solution: set-associative, 2 way; 26.

Real exam problem(3)

(c) Suppose the following code is running on a system with the above cache, where `sizeof(int) == 4`.

```
1  #define array_size   64
2  #define repeat_times 1
3  #define step_size    2
4
5  int main() {
6      int array[array_size] = { };
7
8      for (int r = 0; r < repeat_times; r++) {
9          for (int i = 0; i < array_size; i += step_size) {
10             array[i] = array[i] + 2333;
11         }
12     }
13
14     return 0;
15 }
```

1. What is the total number of accesses to the cache?

2. What is the hit rate?

3. Which type(s) of miss occur(s)?

4. Suppose `repeat_times` **goes to infinity** (only for this question), what number will the hit rate converge to?

5. If `repeat_times` **is changed to 2** (only for this question), try to swap two lines of the above code to maximize the hit rate without disturbing the results. Which two lines will you choose and what is the maximized hit rate?

6. **For the modified code in the previous question**, suppose again `repeat_times` **goes to infinity** (only for this question), what number will the hit rate converge to?

Solution: 64; 0.75; compulsory miss; 0.75; line 8 and 9, 0.875; 1.