通用映射策略是直接映射、组关联映射、关联映射的大一统。

首先我们有

$$\text{组数} \times \text{每组行数} = cache\text{总行数}$$
$$SETS \cdot setSize = cnt\_cacheline$$

然后对于 memory 中的 32 位地址，取前 26 位为块号，后 6 位为块内地址，共 2^{26} 个块，每个块有 $2^6 = 64$ 个字。

容易取出块号 $blockNO$

首先把 $blockNO$ 映射到某个组，这个组的求法是块号对组数取模，即

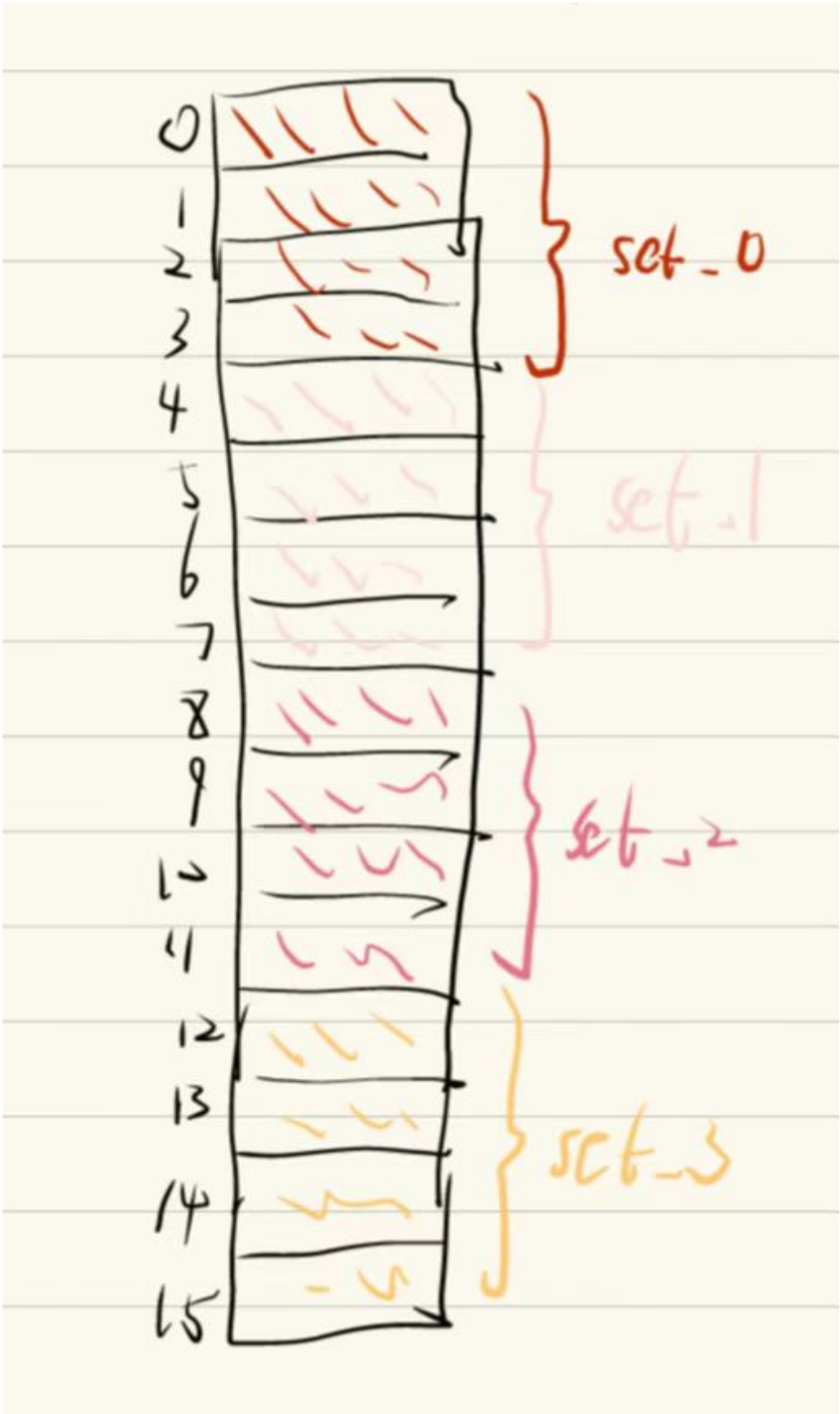
$$set\_id = blockNO \% SETS$$

然后属于设这个组的行(hang)为 $[from, to)$，则有

$$from = set\_id * setSize$$
$$to = (set\_id + 1) * setSize$$

所以我们的 map 方法就是要在这个区间里面找对应的 $blockNO$ 是否存在。

这张丑图是把 16 行的 cache 分成四组，每组四行的情况。然后 $blockNO$ 就是先投射到组，然后在组内找行。


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
set_0
set_1
set_2
set_3

```
private int map(String addr){
    int blockNO = get_block(addr);  //求出块号
    int set_id = ...;   //求组号
    int from = ..., to = ...;   //求属于该组的行区间
    while(from<to) {
        //挨个比较,看看这一行是不是blockNO对应的行
        ++from;
    }
    return -1;//没找到
}
```

那么问题来了，怎么判断该行是否与 $blockNO$ 相对应呢?

tag!

映射到同一个组（例如组号 id）的块号都有一个特点，它们对组数取模都余 id。

还有一个性质，这些块对组数作除法，所得商为 $0, 1, 2, \ldots$。

这个商就是区分不同 block 的关键所在，它们的二进制码，以 26 位 tag 的形式存储在 cache 中。

由此可以写一个 calc_tag 方法，根据块号得出 26 位 tag。

```
private char[] calc_tag(int blockNO){

}
```

然后 cache 行与 block 的编号之间就做好了对应

$$calc\_tag(blockNO) == cache[rowNO].tag$$

然后就到 fetch 方法了。

```
private int fetch(String addr) {
    int mp_rowNO = map(addr);
    if (mp_rowNO != -1) return mp_rowNO;    //映射成功
    //TODO
    return -1;
}
```

问题就是这个 TODO 该怎么写。

首先确定，cache 里面找不到我们要的这个块，所以我们需要把这个块以及块的数据替换到cache里面。

所以就 TODO 就是确定哪一行来替换的问题，这个交给 Replacement 策略。暂时谈谈直接映射的替换。

直接映射，琢磨琢磨就是每组一行，区间 $[from, to)$ 长为 1，把 $from$ 行替换掉就好啦。

```
private int fetch(String addr) {
    int mp_rowNO = map(addr);
    if (mp_rowNO != -1) return mp_rowNO;    //映射成功
    //TODO
    if (replacementStrategy == null) {
        update(from, tag, data);
    }
    return from;
}
```