CS 631-02 Cache Simulation

analysis

~~quips~~

jal rd, offset

↑

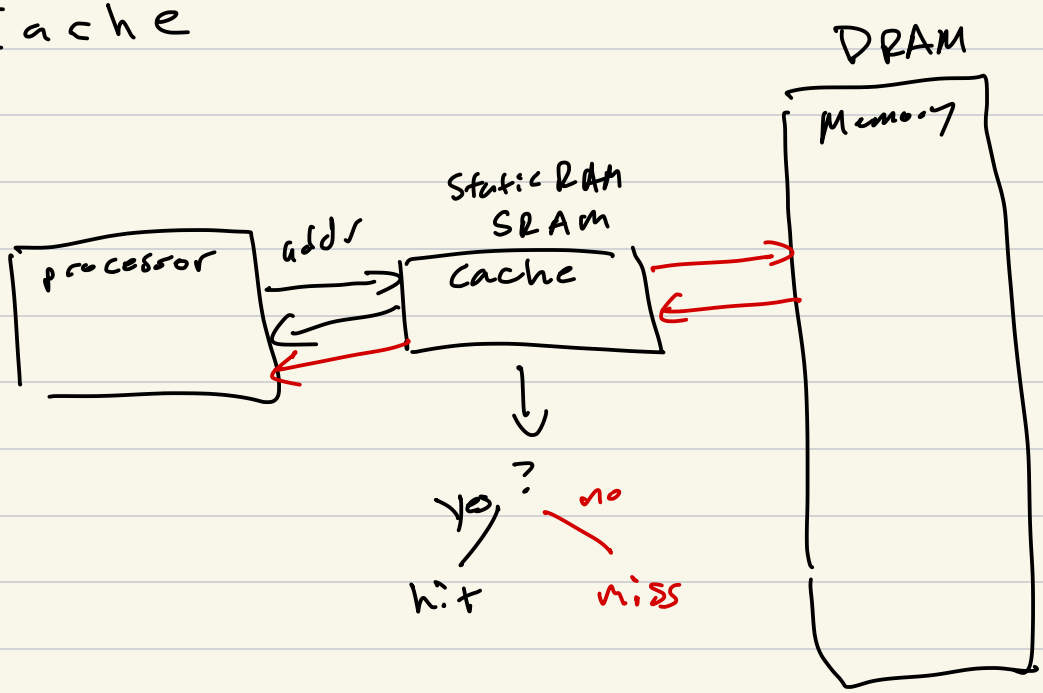
call jal x1, offset
ra

↓

j jal x0, offset

```
if (rd != 0) {
    regs[rd] = pc + 4
}
```

Cache

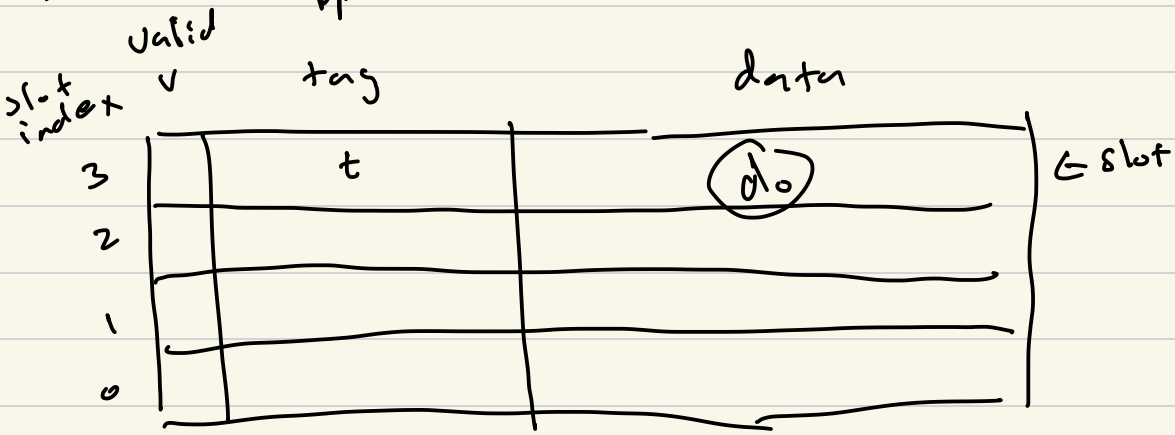


memory requests

$$\text{hit rate} = \frac{\#\text{ hits}}{\#\text{ reqs}}$$

$$\text{miss rate} = \frac{\#\text{ misses}}{\#\text{ reqs}}$$

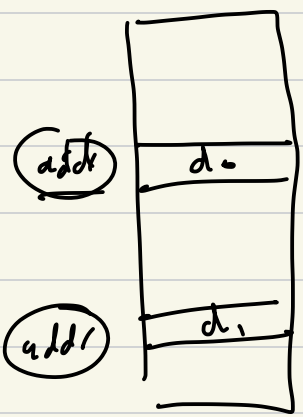
Direct Mapped



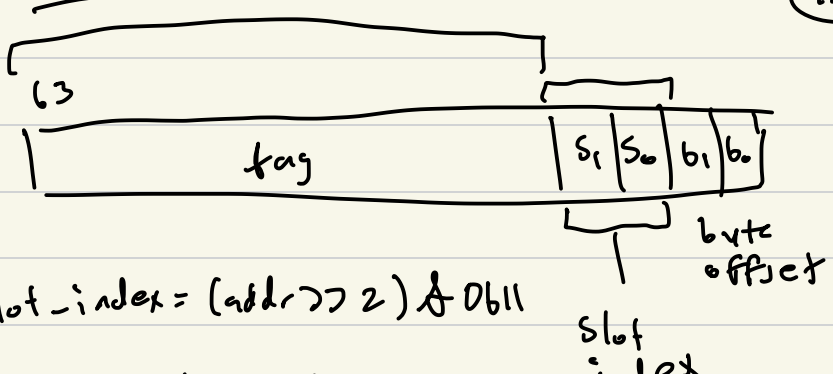
addr assume addr is word aligned

addr_word = addr / 4

slot_index = addr_word % 4

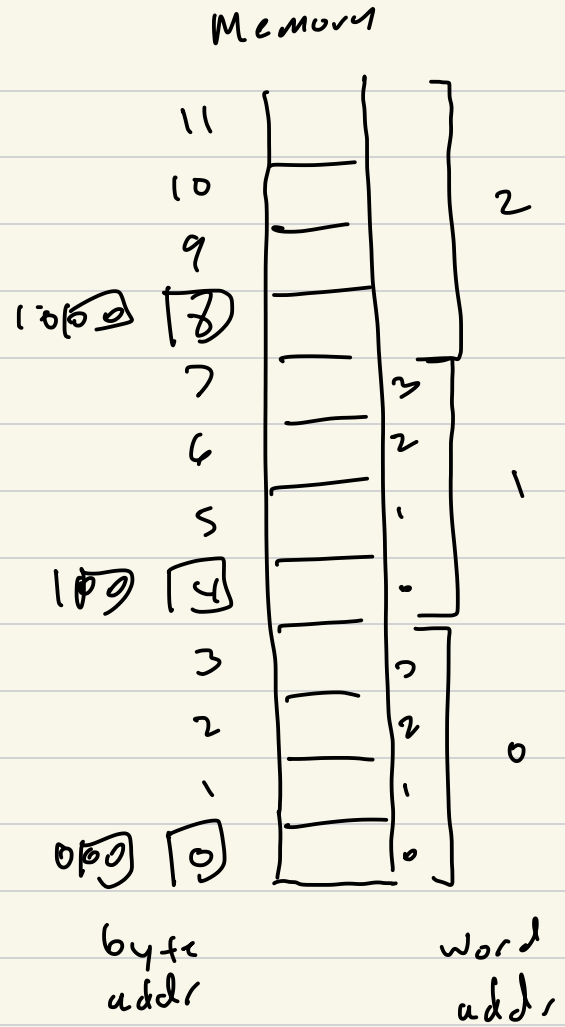


addr



slot_index = (addr >> 2) & 0b11

tag = addr >> 4


Memory
11
10
9
8
7
6
5
4
3
2
1
0
2
1
0
3
2
1
0
byte addr
word addr

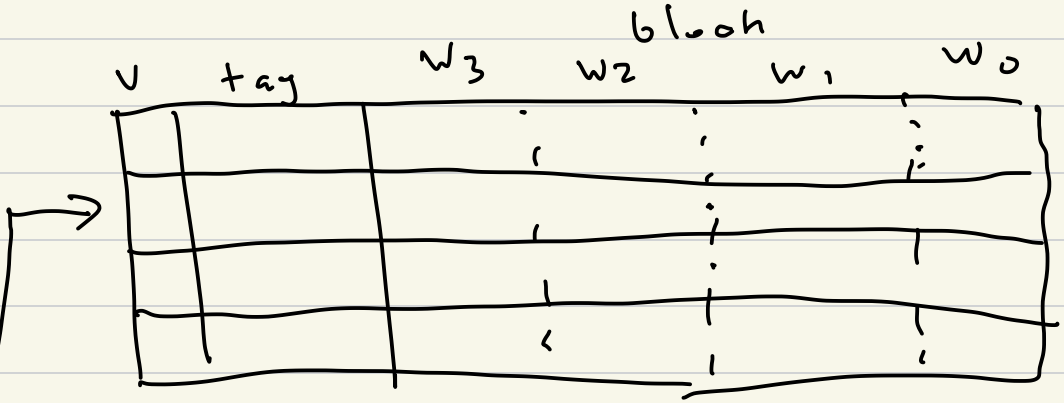
Direct Mapped Pseudo Code

```
tag = addr >> 4;
index_mask = 0b11
slot_index = (addr >> 2) & index_mask
slot = cache[slot_index]
if (slot.valid == 1 && slot.tag == tag) {
    // hit
    return slot.data
} else {
    // miss
    slot.data = *((uint32_t *) addr)
    slot.tag = tag
    slot.valid = 1
}
```

Principles of locality

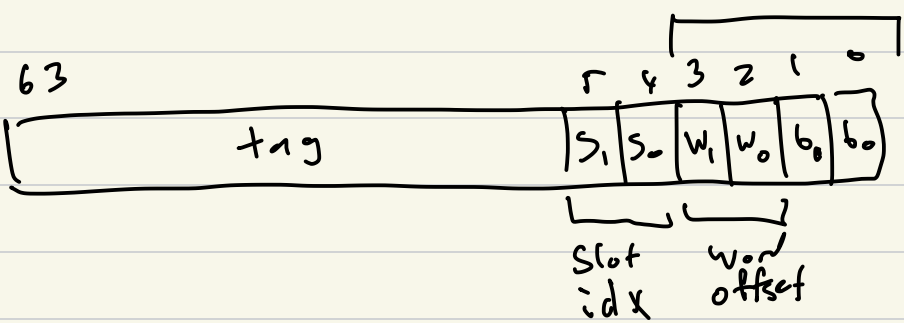
- temporal
- spatial

Block Size



addr

addr_word = addr / 4



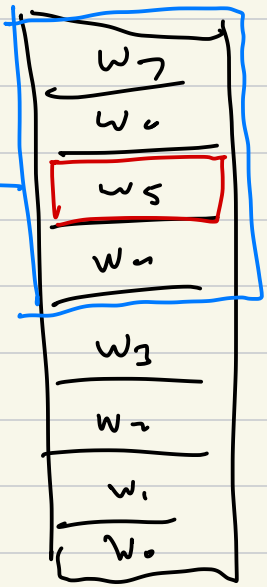
slot_idx = addr_word % 16

slot_idx = addr >> 4

Block Size

miss

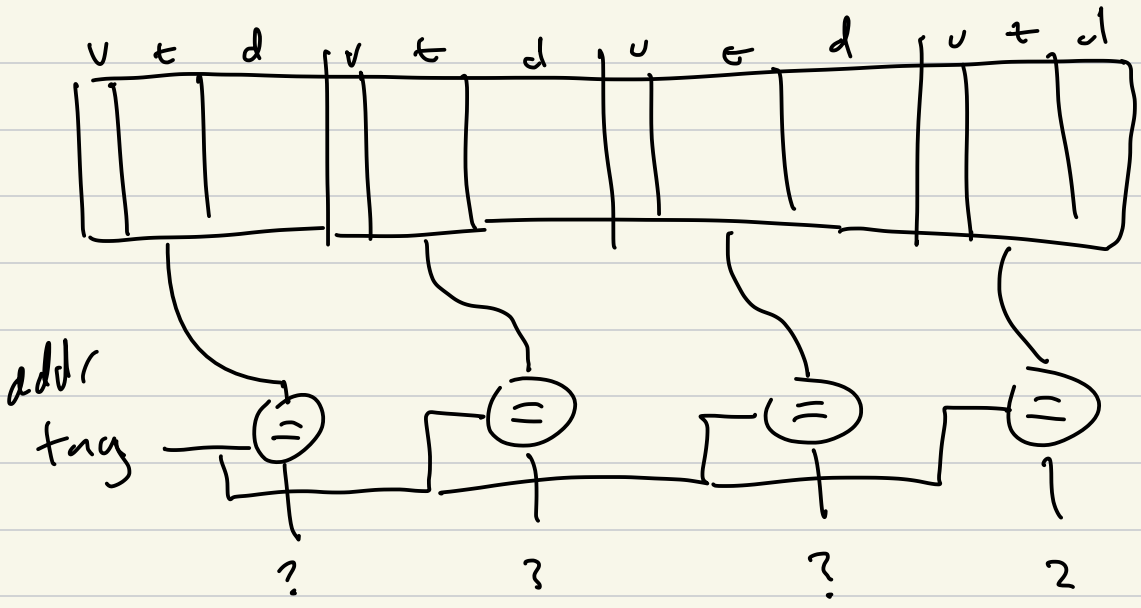
addr



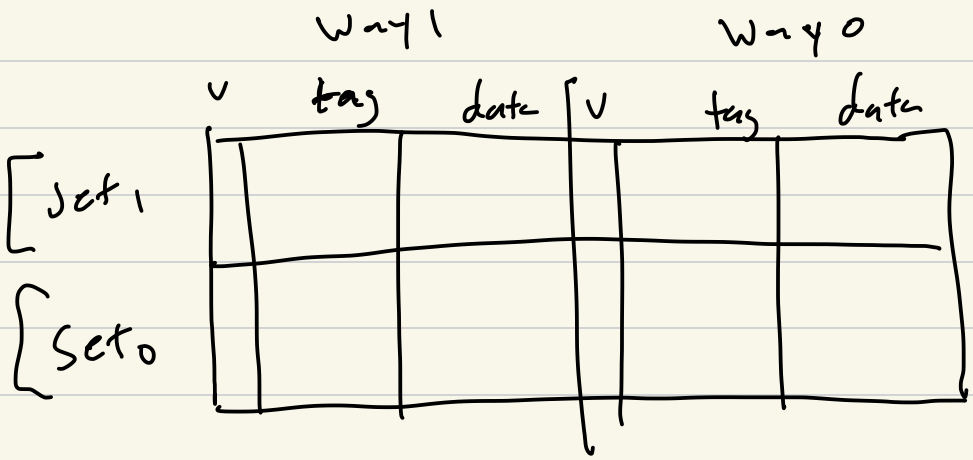
cache slot

block array

Fully Associative Cache

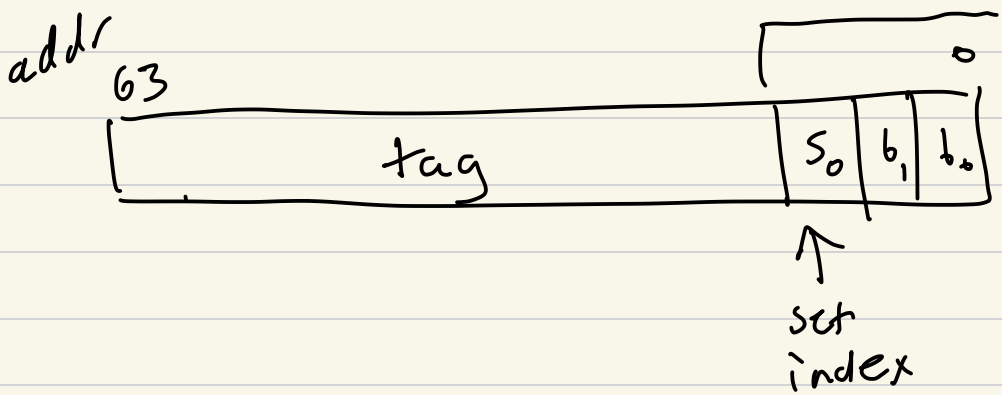


Set Associative Cache

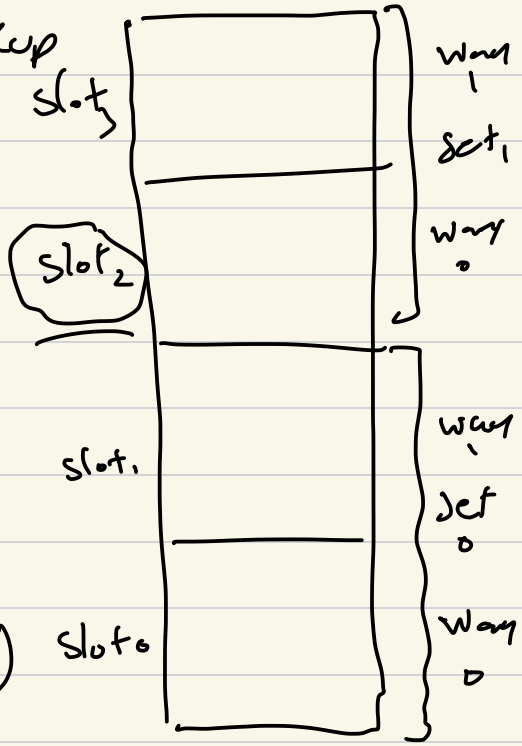


n-way set associative

2-way



SA Pseudo Code Lookup




```
num_refs += 1;
num_ways = 2
addr_tag = addr >> 3
set_index = (addr >> 2) & 0b1
set_base = set_index * 2
for (i=0; i<2; i++) {
    slot = cache[set_base+i]
    if (slot.valid && slot.tag == tag)
        // hit
        slot.timestamp = num_refs;
        return slot.data
}
// miss
slot = find_lru_in_set(cache, set_base)
slot.data = *((uint32_t *) addr);
slot.tag = tag
slot.timestamp = num_refs
return slot.data;
```