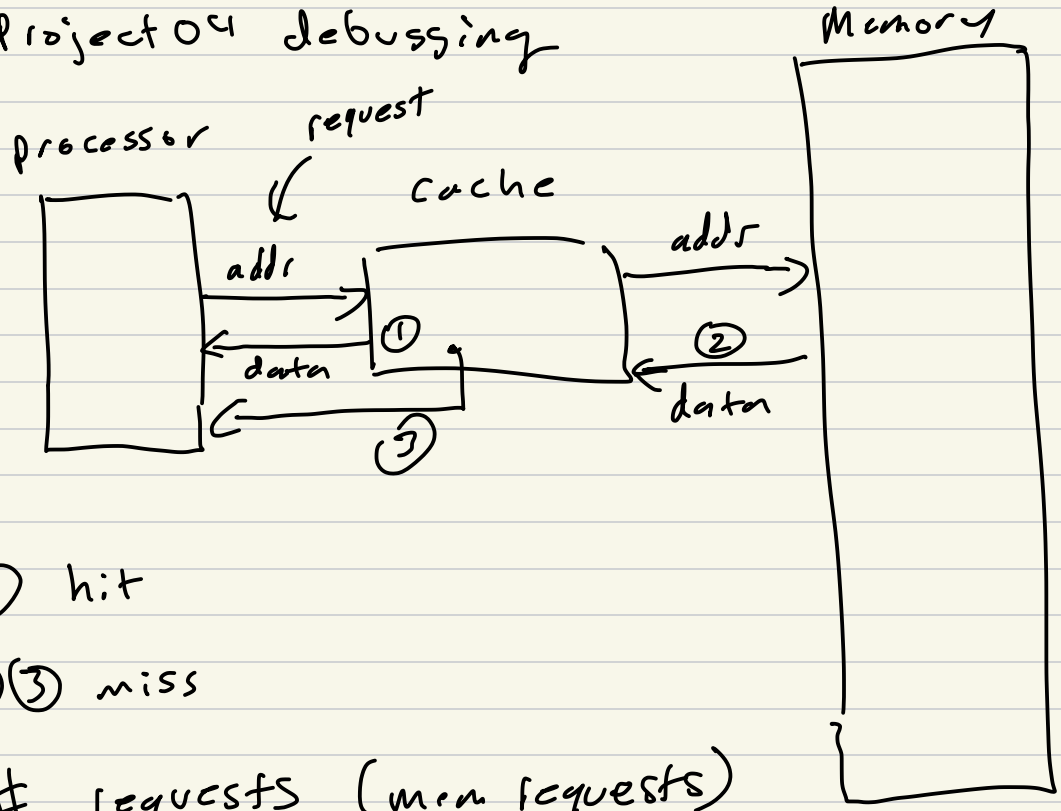
CS 315-01 Cache Simulation

Project 04 debugging



(1) hit

(2)(3) miss

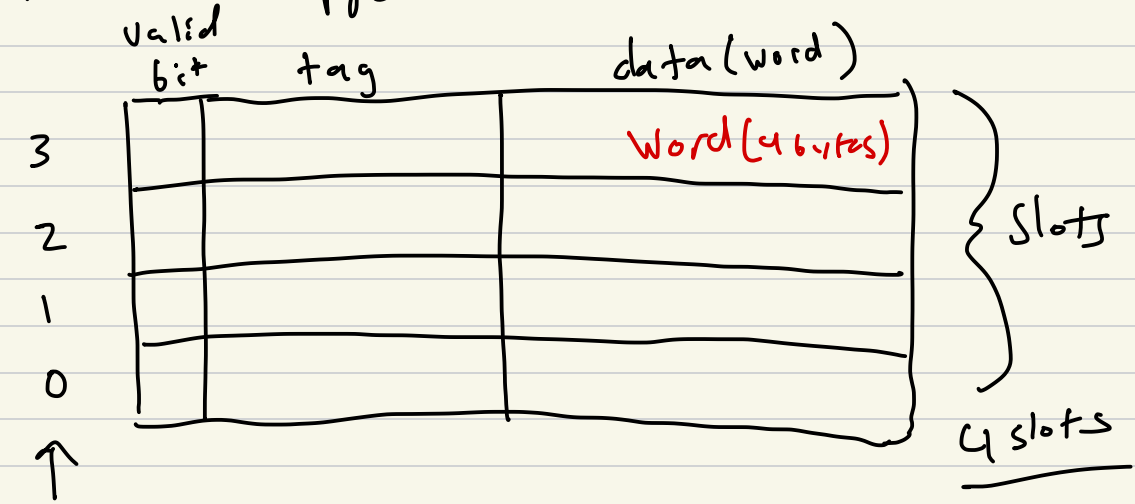
requests (mem requests)
reqs

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

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

$$\text{hit rate} = 1 - \text{miss rate}$$

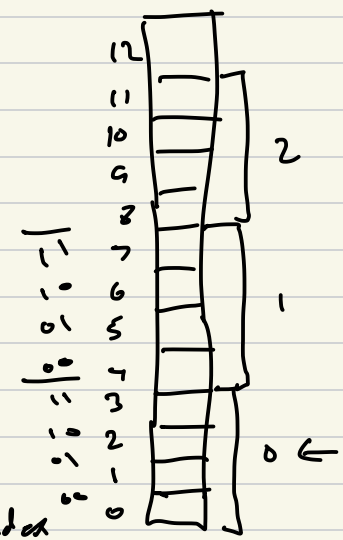
Direct Mapped Cache



addr assume addr is word aligned

addr_word = addr_byte / 4

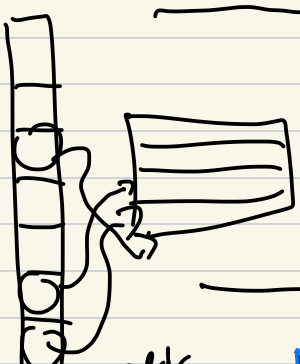
slot_index = addr_word % 4



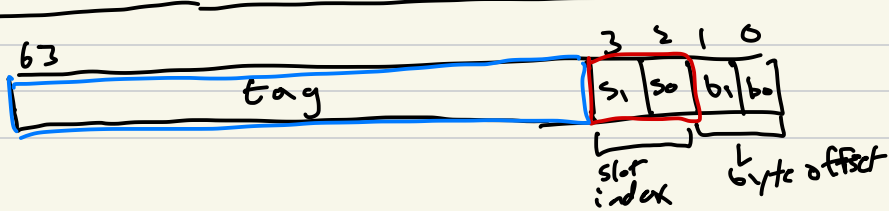
addr_word slot_index

4 4%4 = 0

17 17%4 = 1



addr (byte) 64 bit



4 = 000000

17 = 010001

17 x 4 = 68 = 1000100

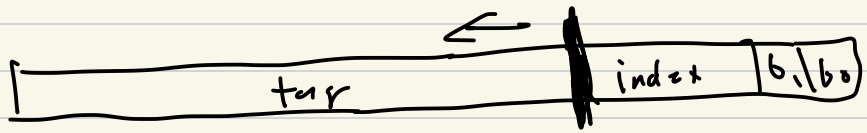
slot_index = (addr >> 2) & 0b11

Direct Mapped Cache Pseudo code

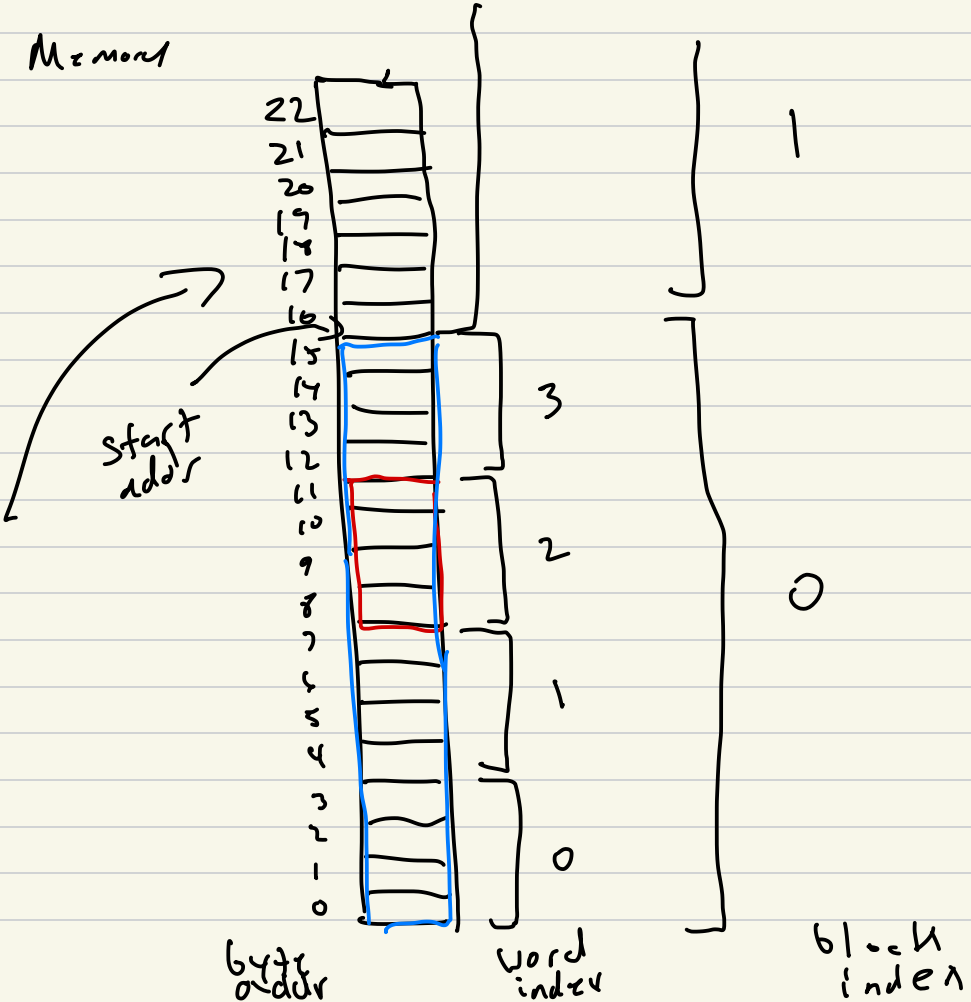
```
Lookup (addr)
    tag = addr >> 4;
    index_mask = 0b11;
    slot_index = (addr >> 2) & index_mask
    slot = cache[slot_index]
    if (slot.valid == 1 && slot.tag == tag) {
        // hit
        return slot.data
    } else {
        // miss
        slot.data = *((uint32_t *)addr);
        slot.tag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

# of slots

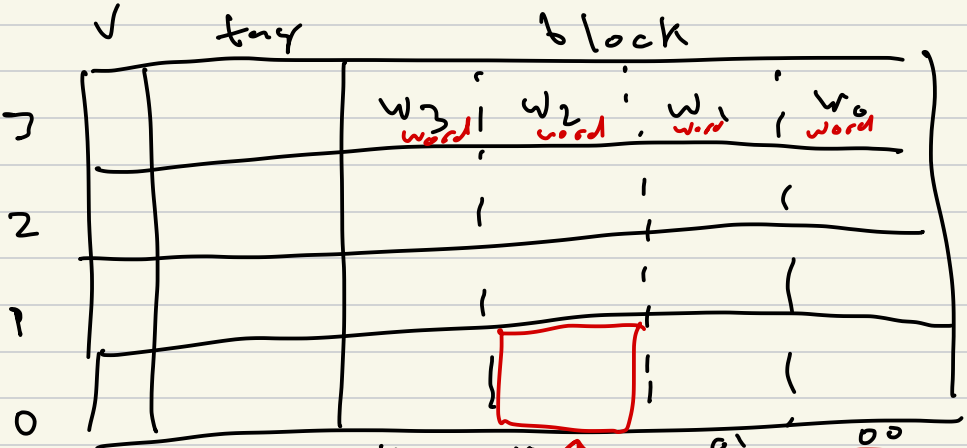
4 slots (2 bits)
8 slots (3 bits)
16 slots (4 bits)
128 slots (7 bits)

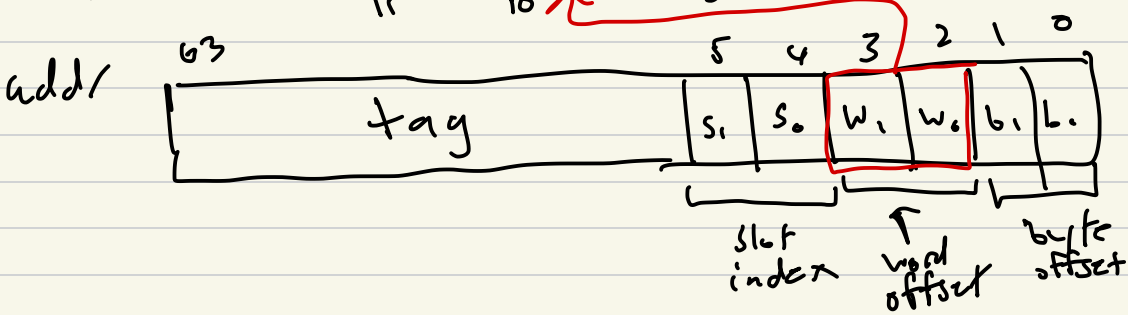


Mask



Block size (size of 4 words)





With block size > 1

1) on hit

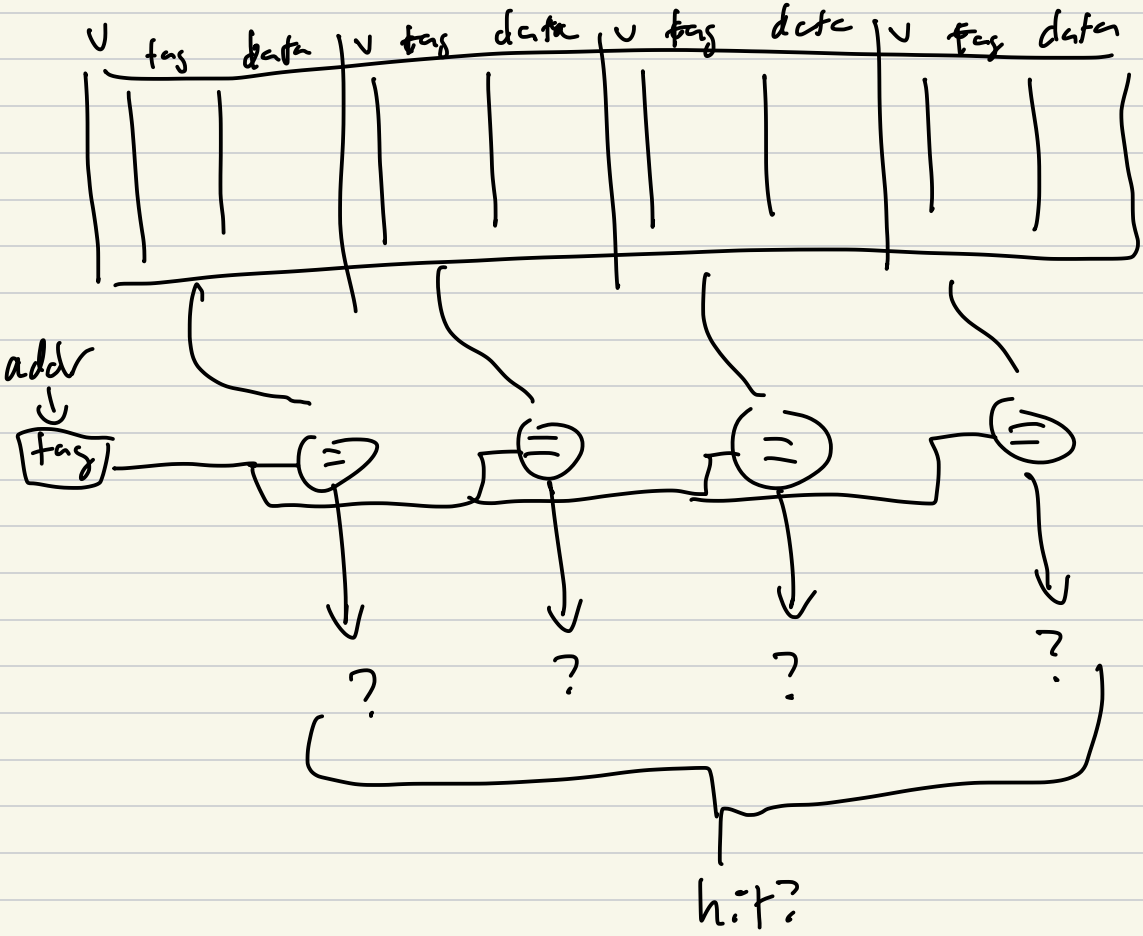
get word from slot using the word offset

2) on miss

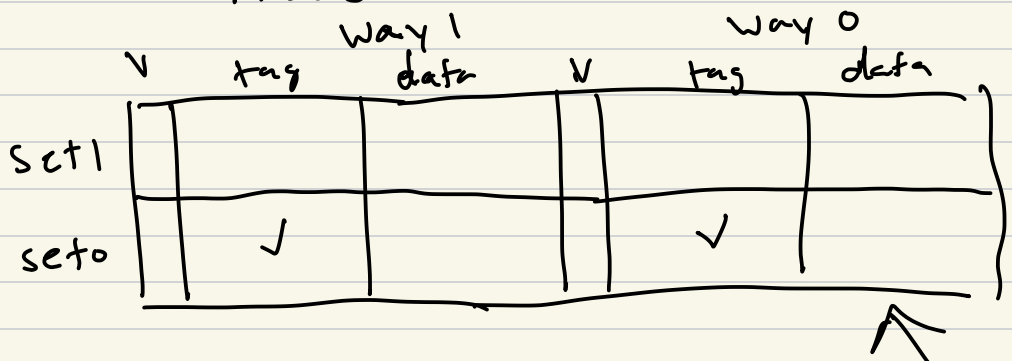
find block_start_addr from addr

get N words from memory

Fully Associative Cache



Set Associative Cache



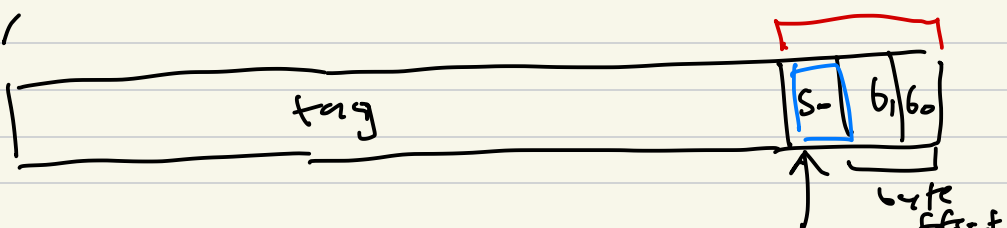
n-way set associative

4-way set associative

block size

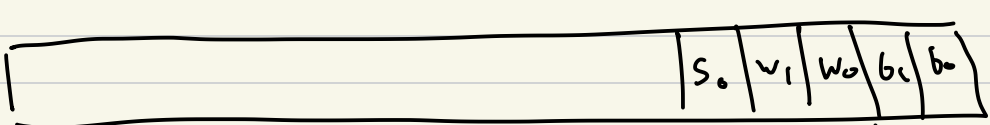
2-way

addr



set index

addr block size = 4



set index

word offset

Set Associative Pseudo Code

```
sa_lookup(addr)

    num_refs += 1;
    num_ways = 2
     tag = addr >> 3;
     set_index = (addr >> 2) & 0b1;
    set_base = set_index * 2
    for (i=0; i<num_ways; 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) slots
            // favor unused
                slots

slot.data = *((uint32_t *) addr);
slot.tag = tag
slot.timestamp = num_refs;
return slot.data
```