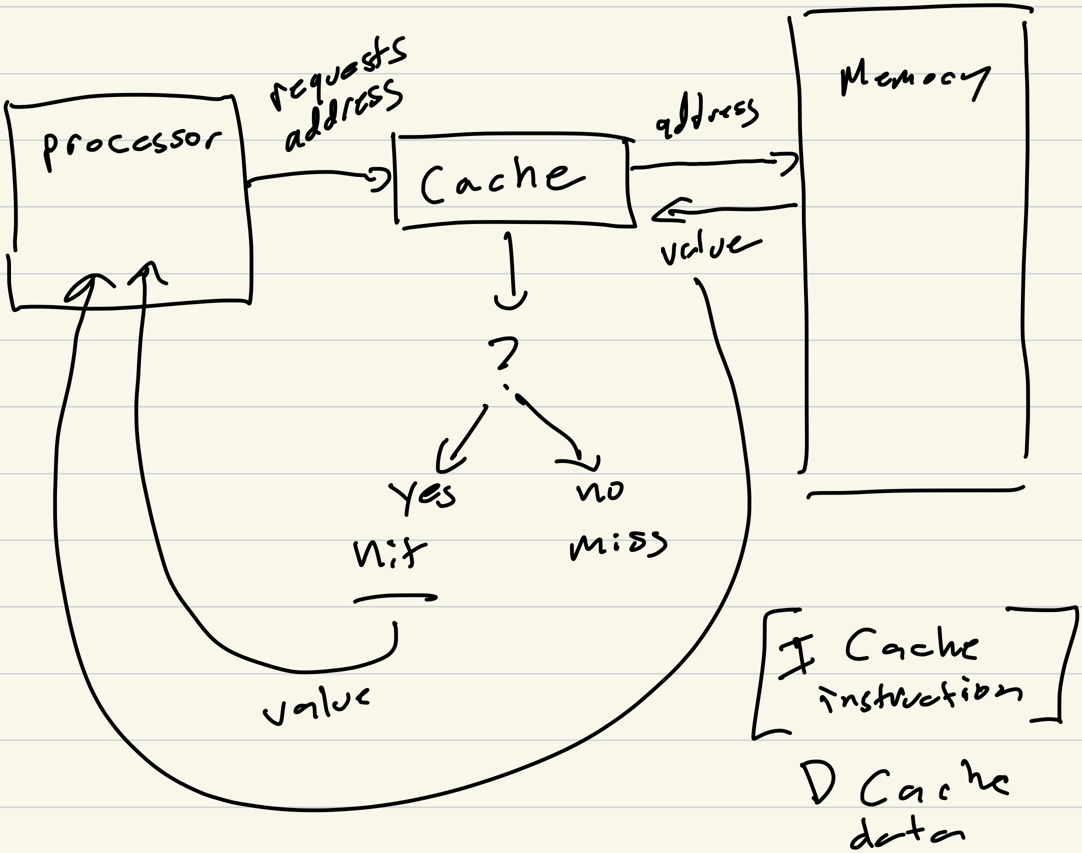
CS315-02 Cache Simulation



I Cache
instruction

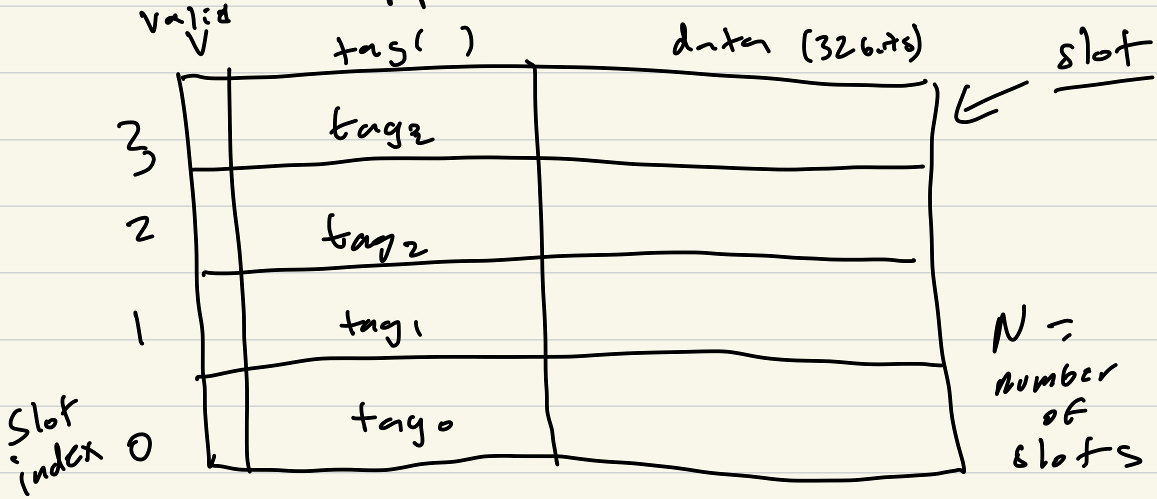
D Cache
data

# memory requests / references
reqs refs

hit rate $\frac{\#\text{hits}}{\#\text{refs}}$

miss rate $\frac{\#\text{misses}}{\#\text{refs}}$

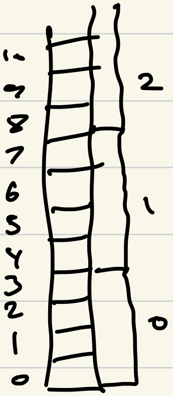
Direct Mapped Cache

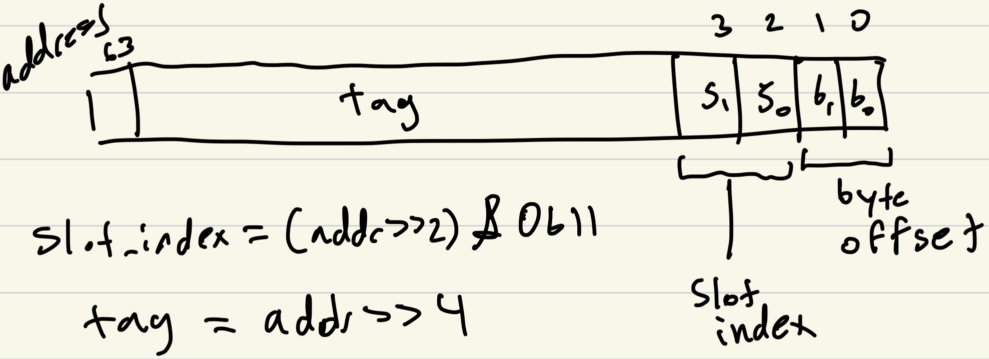


addr assume addr is word aligned

addr_word = addr_byte / 4;

slot_index = addr_word % $\overset{N}{\underline{4}}$;





slot_index = (addr>>2) & 0b11

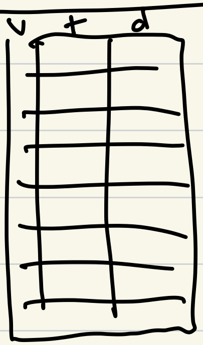
tag = addr>>4

Direct Mapped Pseudo code

```
tag = addr >> 4;
index_mask = 0b11;
slot_index = (addr >> 2) & index_mask;
slot = cache[slot_index];

if ( slot.valid && slot.tg = tag) {
    // hit
    return slot.data
} else {
    // miss
    slot.data = *((uint32_t *) addr);
    slot.tag = tag;
    slot.valid = 1;
    return slot.data;
}
```

8 Slots



```
addr_word = addr_byte / 4
slot_index = addr_word % 8
```

```
tag = addr >> 5;
index_mask = 0b111
slot_index = (addr >> 2) & index_mask
```

Block Size

block

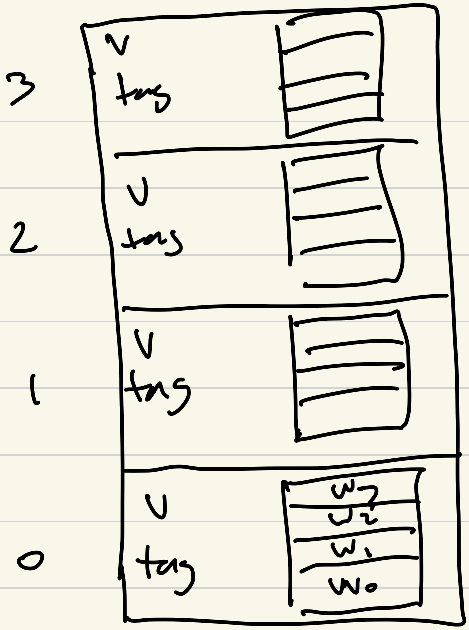
32 bits

	V	tag	W_3	W_2	W_1	W_0
3						
2						
1						
0						

addr

63		5	4	3	2	1	0
	tag	s_1	s_0	w_1	w_0	b_1	b_0
		slot index		word offset		byte offset	

Slots



slot_index

hit

data = slot.block[$\underset{x}{0}$]

block_index = addr_word % 4

data = slot.block[block_index];

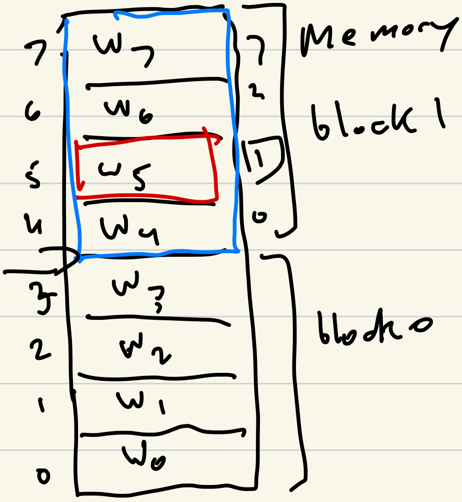
miss

addr

block_base = addr_word - block_index

5 - 1

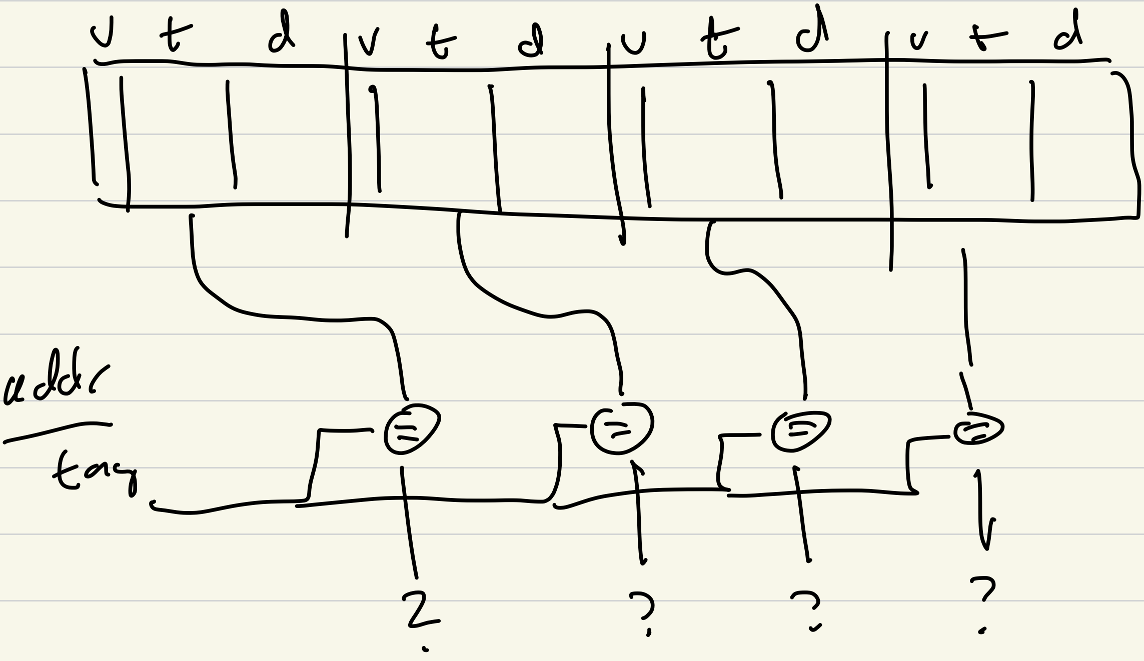
= $\boxed{4}$



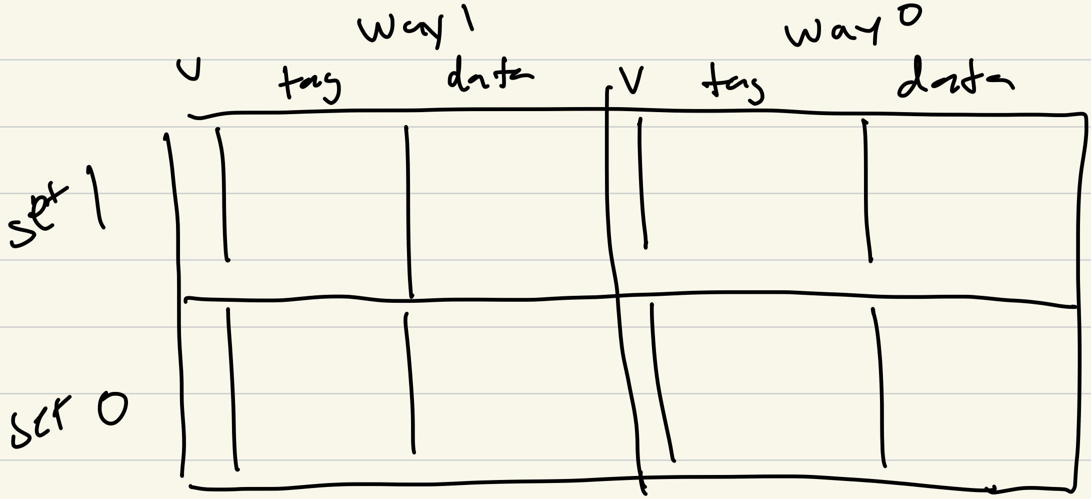
Read in entire block (1 word at a time)

return the word needed

Fully Associative Cache



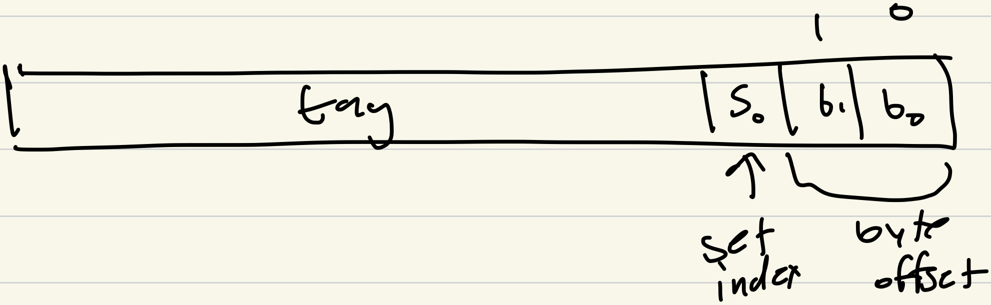
Set Associative Cache



n-way set associative

↘ how many ways

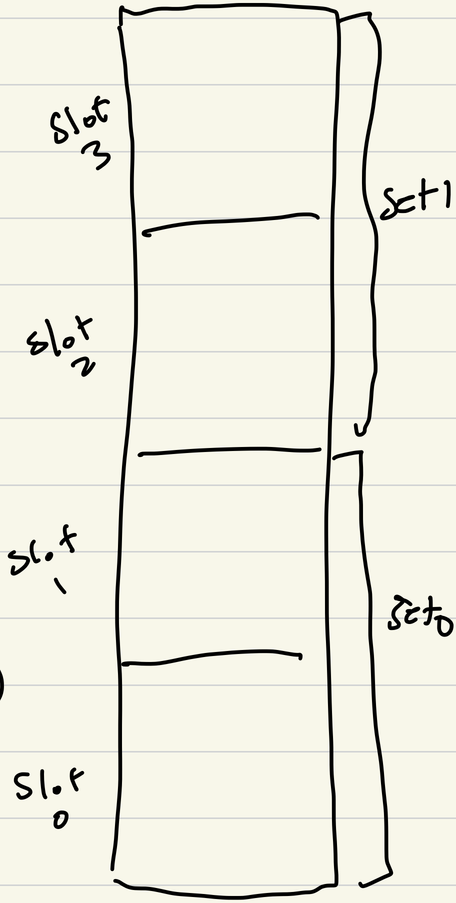
2-way SA Cache



SA Pseudo Code Lookup

```
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)

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


slot 3
set1
slot 2
slot 1
set0
slot 0