CS354: Machine Organization and Programming

Lecture 21
Wednesday the October 21th 2015

Section 2
Instructor: Leo Arulraj

Class Announcements

1. Programming Assignment 2 was due by 9 AM today. You can submit it upto 48 hours after the deadline with penalties.

2. Email me if you will have conflicts with the CS354 Midterm Exam 2:

   Nov 10th Tues 5:30 PM to 7:00 PM at

   Van Vleck Room B130(Section 2)

   (Come to the Location about 15 mins earlier)

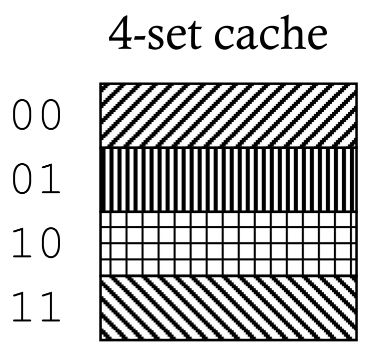
Lecture Overview

1. Types of Cache misses
2. Looking up the cache contents in Set Associative Caches
3. Tracing through an example Set Associative Cache

On a miss

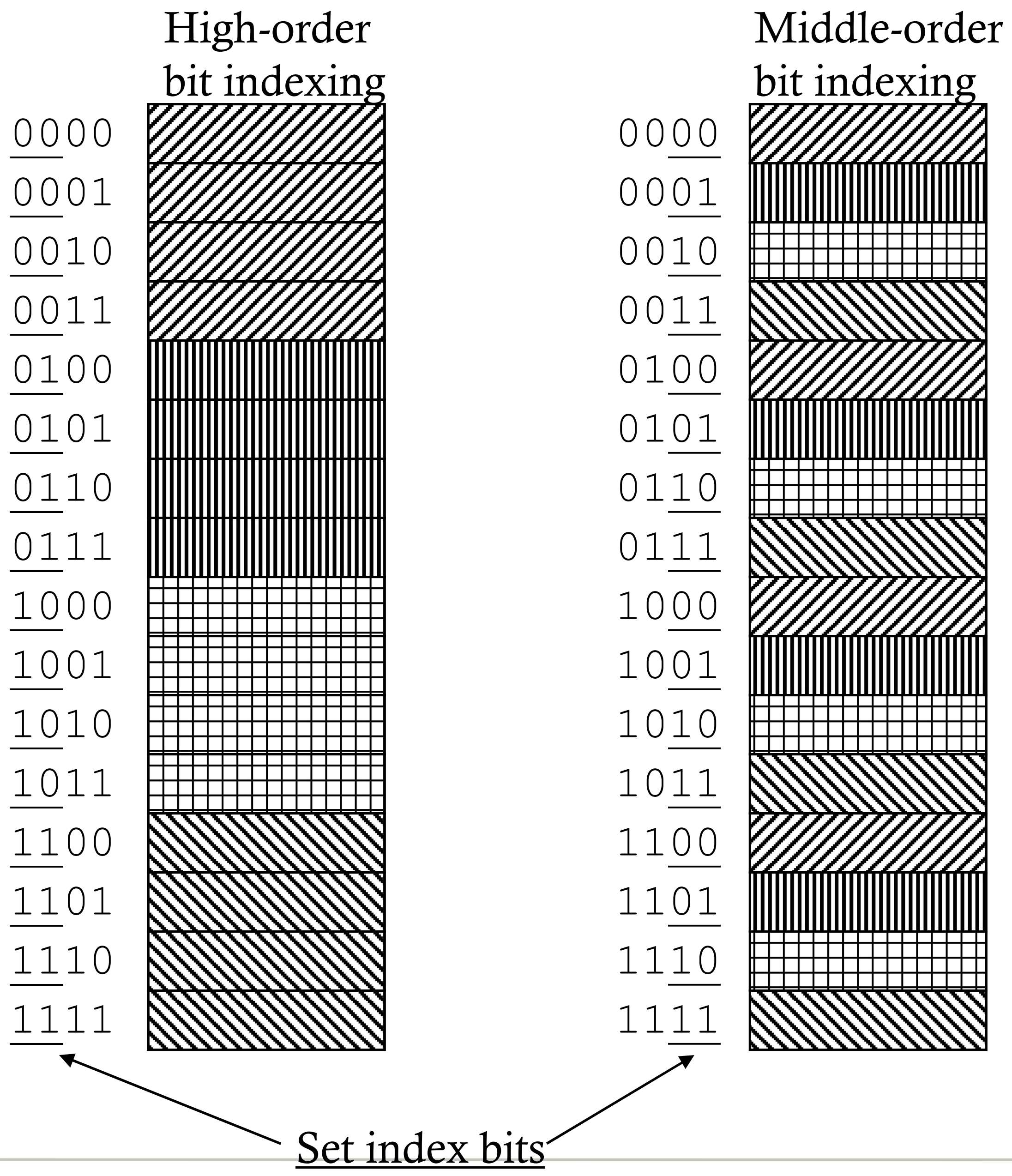
- Send the memory request to main memory.
- Memory returns the entire block containing the needed byte/word.
- Place the block into the frame.
  - Set the tag bits
  - mark the frame valid.

And, while doing this, extract the byte/word + return it to the processor, completing the memory access.



<span style="color:red">Why middle order bits are used as Set Index?</span>

Higher order bits lead to consecutive addresses mapping to same set.



Types of misses:

1) compulsory
2) conflict
3) capacity

Types of Misses

- **Compulsory or cold misses:** Cache is empty to start with and will miss.
- **Conflict misses:** Cache has space but because objects map to the same cache block they keep missing.
- **Capacity misses:** Cache does not have space because size of the working set exceeds the size of the cache.

Conflict misses are common

- **Consider:**

```
float dotprod(float x[8], float y[8])
{
        float sum = 0.0; register int i;
        for(i=0;i<8;i++)
                sum += x[i] * y[i];
        return sum;
}
```

Analyze for (S,E,B,m) = (2,1,16,6)

Conflict misses are common

- **It causes thrashing: repeatedly loading and evicting same cache blocks**

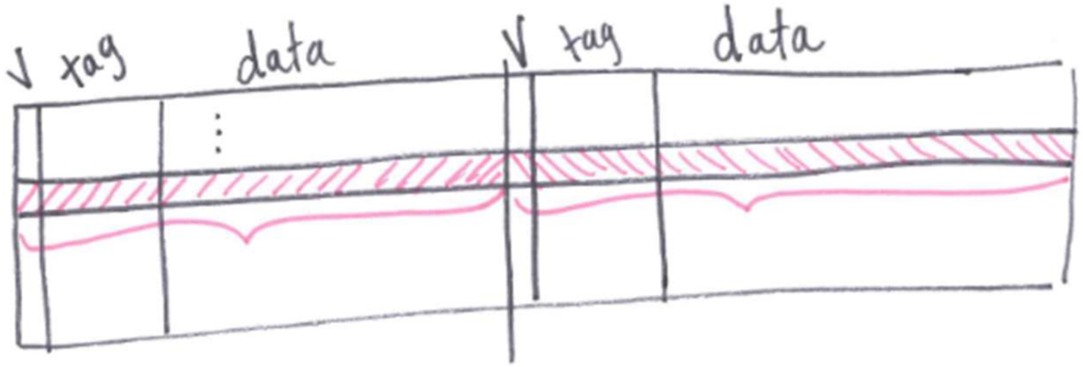
- **Thrashing is easy to avoid once you know it is going on: Use padded arrays so that the accessed elements are mapped to different cache sets**

To reduce conflict misses

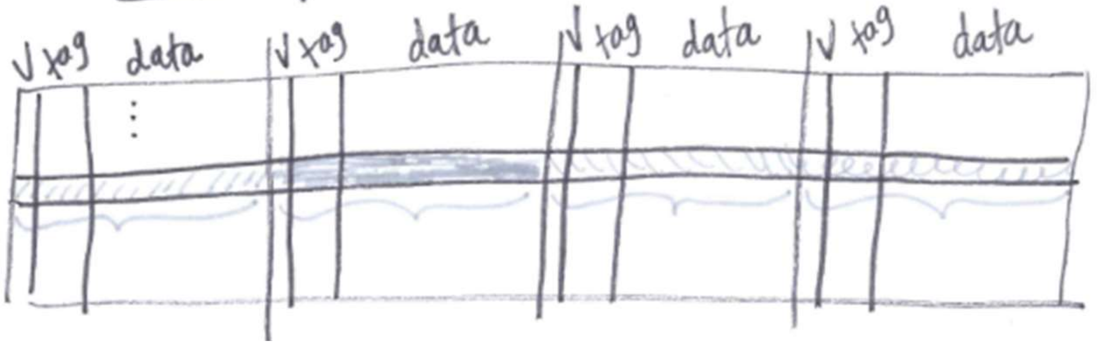
increase set associativity

2-way set associative

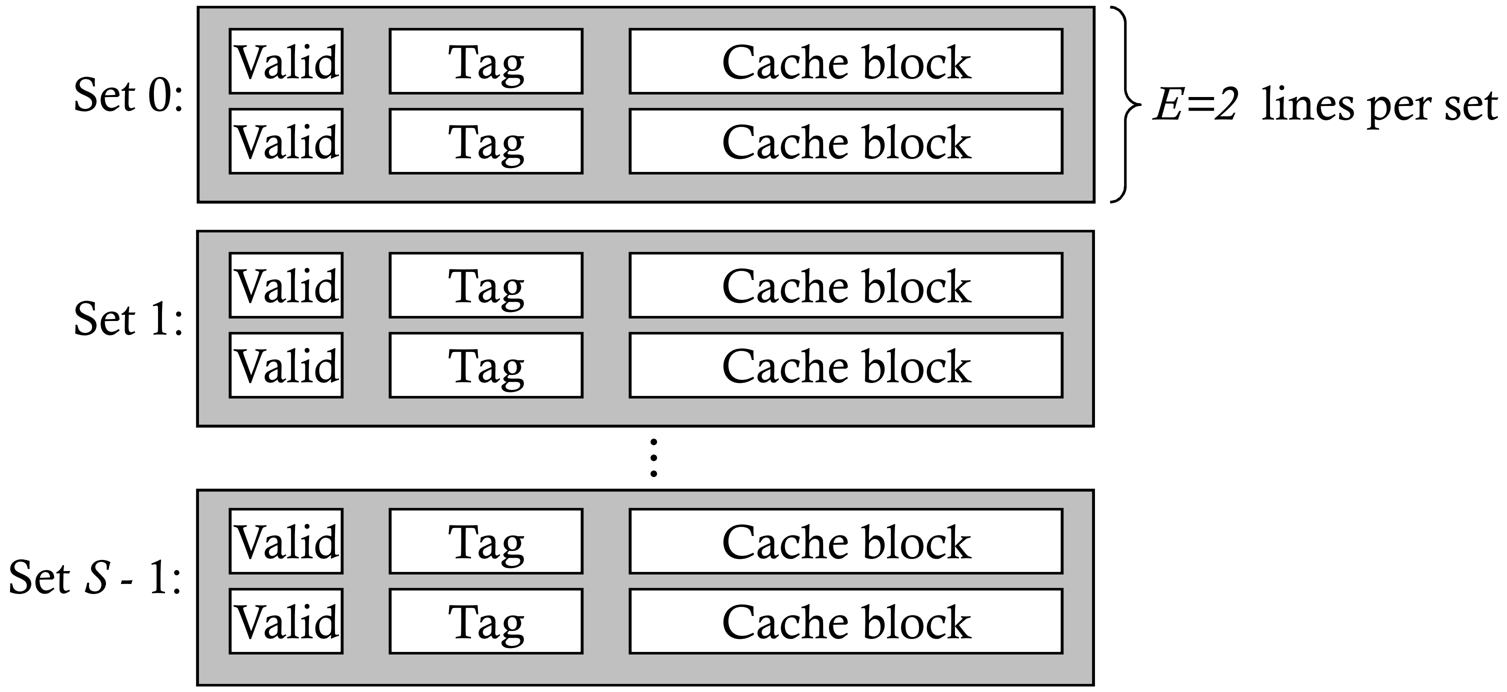
2 blocks per set (line)



4-way set associative



Set Associative Cache Organization

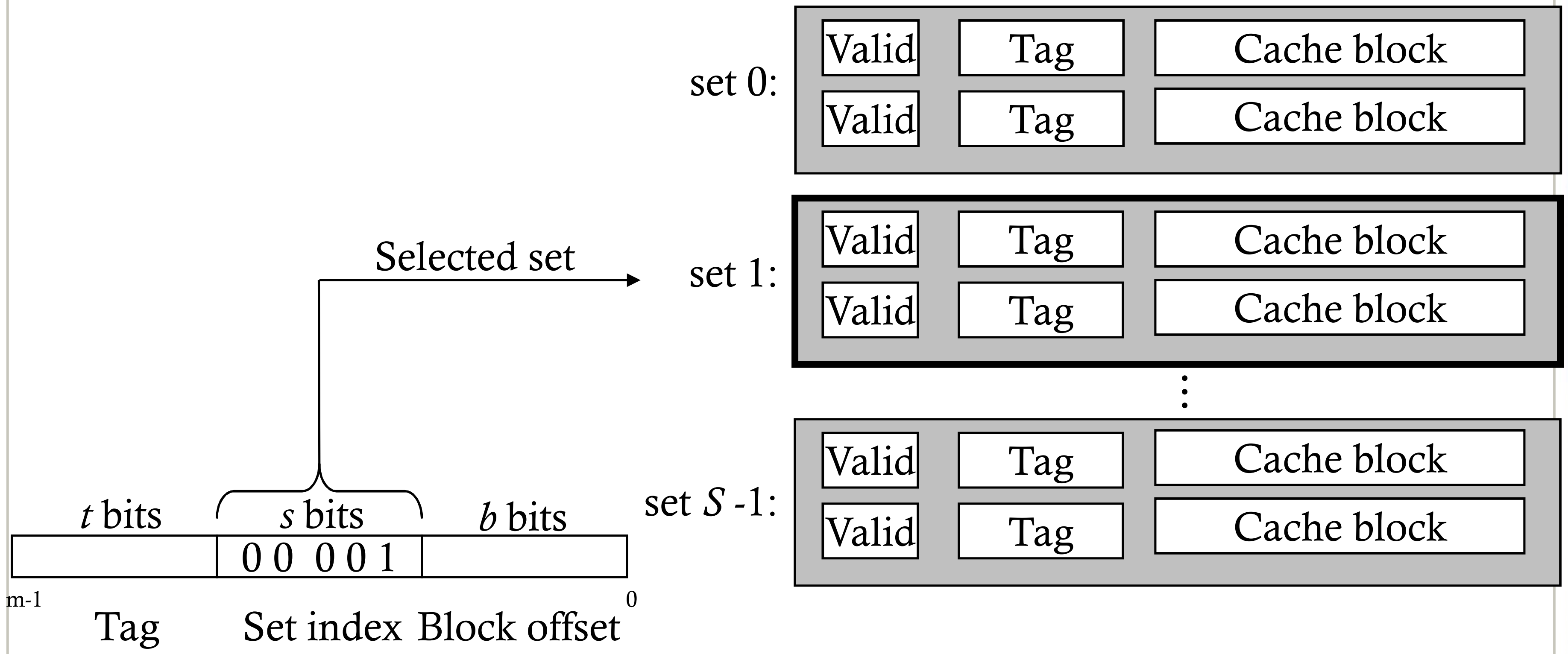


Larger set size

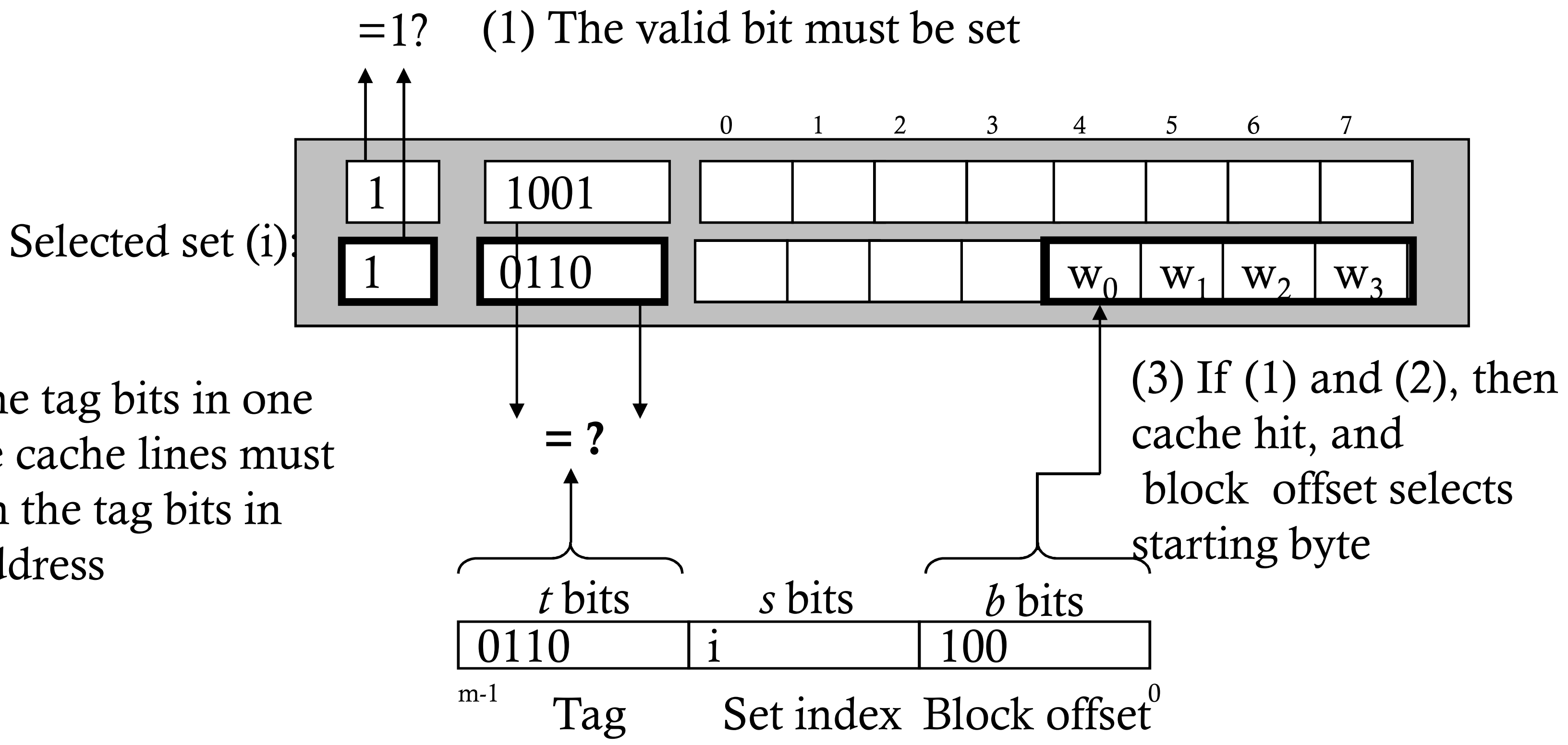
🙂 tends to lead to higher hit ratio (due to fewer conflic misses)

🙁 amount of circuitry goes up, leading to increase in T_c

Looking up a memory address in Set Associative Cache



Looking up a memory address in Set Associative Cache



(2) The tag bits in one of the cache lines must match the tag bits in the address

Tracing through a sample Set Associative Cache from CSAPP textbook practice problem 6.13

Set Associative Cache Practice Proble 6.13-6.16

- Consider a cache with: (S,E,B,m) = (8,2,4,13)

- Analyze memory references to :

- 0x0E34
- 0x0DD5
- 0x1FE4

The memory layout is shown in in next slide.

2-way set associative cache

	Line 0						Line 1					
Set index	Tag	Valid	Byte 0	Byte 1	Byte 2	Byte 3	Tag	Valid	Byte 0	Byte 1	Byte 2	Byte 3
0	09	1	86	30	3F	10	00	0	—	—	—	—
1	45	1	60	4F	E0	23	38	1	00	BC	0B	37
2	EB	0	—	—	—	—	0B	0	—	—	—	—
3	06	0	—	—	—	—	32	1	12	08	7B	AD
4	C7	1	06	78	07	C5	05	1	40	67	C2	3B
5	71	1	0B	DE	18	4B	6E	0	—	—	—	—
6	91	1	A0	B7	26	2D	F0	0	—	—	—	—
7	46	0	—	—	—	—	DE	1	12	C0	88	37

The following figure shows the format of an address (one bit per box). Indicate (by labeling the diagram) the fields that would be used to determine the following:

CO	The cache block offset
CI	The cache set index
CT	The cache tag

12	11	10	9	8	7	6	5	4	3	2	1	0

Cache Replacement Policies

- Which block to replace or evict to make space for new blocks?
    - Random Replacement Policy: chooses a random victim block.
    - Least Recently Used (LRU) Policy: chooses the block that was last accessed furthest in the past.
    - Least Frequently Used (LFU) Policy: chooses the block that was least frequently accessed in the past.