

CSE 351

Caches

Cache Associativity

- Determines the number of different locations a given address can map to in the cache
- Ex. Cache associativity = 1 (direct-mapped)
 - This means that every address has only one possible line it can map to
- Ex. Cache associativity = Cache size / Block size (fully-associative)
 - This means that any address can map to any line of the cache

Cache Mapping

- To determine where addresses map into a cache, you need to break the address space up into TAG, SET (a.k.a. index), and OFFSET bits
 - Work from right to left [TAG][SET][OFFSET]
 - The $\log(B)$ -most bits are used to express block offset
 - The next $\log(S)$ -most bits are used to express the set number
 - The remaining bits represent the tag

Cache Mapping

- Why do we order the bits TAG|INDEX|OFFSET?
 - This allows caches to capitalize on locality
- OFFSET bits are the lowest-order bits
 - Adjacent addresses are thus in the same block
 - Takes advantage of spatial locality
- INDEX bits are next
 - Adjacent blocks are in different sets
 - Takes advantage of temporal locality
- The location of the TAG bits is a result of the placement of the first two sets of bits

Example Cache: direct-mapped

Set	Valid	Tag	B0	B1	B2	B3	Set	Valid	Tag	B0	B1	B2	B3
0	1	15	63	B4	C1	A4	8	0	-	-	-	-	-
1	0	-	-	-	-	-	9	1	00	01	12	23	34
2	0	-	-	-	-	-	10	1	01	98	89	CB	BC
3	1	0D	DE	AF	BA	DE	11	0	1E	4B	33	10	54
4	0	-	-	-	-	-	12	0	-	-	-	-	-
5	0	-	-	-	-	-	13	1	11	C0	04	39	AA
6	1	13	31	14	15	93	14	0	-	-	-	-	-
7	0	-	-	-	-	-	15	1	0F	FF	6F	30	00

Example Cache: 2-way set associative

Set	Valid	Tag	B0	B1	B2	B3	Set	Valid	Tag	B0	B1	B2	B3
0	0	-	-	-	-	-	0	0	-	-	-	-	-
1	0	-	-	-	-	-	1	1	2F	01	20	40	03
2	1	03	4F	D4	A1	3B	2	1	0E	99	09	87	56
3	0	-	-	-	-	-	3	0	-	-	-	-	-
4	0	06	11	23	6A	42	4	0	-	-	-	-	-
5	1	21	DE	AD	BE	EF	5	0	-	-	-	-	-
6	0	-	-	-	-	-	6	1	37	22	B6	DB	AA
7	1	11	00	12	51	55	7	0	-	-	-	-	-

Example Cache: fully-associative

Set	Valid	Tag	B0	B1	B2	B3	Set	Valid	Tag	B0	B1	B2	B3
0	1	1F4	00	01	02	03	0	0	-	-	-	-	-
0	0	-	-	-	-	-	0	1	AB	02	30	44	67
0	1	100	F4	4D	EE	11	0	1	34	FD	EC	BA	23
0	0	77	12	23	34	45	0	0	-	-	-	-	-
0	0	-	-	-	-	-	0	1	1C6	00	11	22	33
0	1	101	DA	14	EE	22	0	1	45	67	78	89	9A
0	0	-	-	-	-	-	0	1	01	70	00	44	A6
0	1	16	90	32	AC	24	0	0	-	-	-	-	-

Example Problems

- The caches on the previous slides have the following properties:
 - 64 byte cache size
 - 4-byte block size
 - The associativity varies
- Assume an 11-bit address space
- Data is 1-byte addressable

Example Problems

- Direct-mapped cache:
 - READ 0x7AC
 - READ 0x24
 - READ 0x99F
- 2-way set associative cache:
 - READ 0x435
 - READ 0x388
 - READ 0x0D3
- Fully-associative cache:
 - READ 0x1DD
 - READ 0x719
 - READ 0x2AA

Example Problems

- What is the miss rate for the following code?
 - Assume cache size 1 KB, direct-mapped, 16B block size

```
for (int i = 0; i < 64; i++) {  
    for (int j = 0; j < 64; j++) {  
        array[i][j] = 0;  
    }  
}
```

Example Problems

- In the previous example, what code modifications can change the miss rate?
- What cache changes can change the miss rate?
 - Changing the cache size?
 - Changing the associativity?
 - Changing the block size?

Cache Experiments

- Assume we have some way of querying the cache to see whether certain addresses hit or miss
- What sequences of accesses can help us find more about the cache?
 - Block size
 - Associativity
 - Cache size