15-213 Recitation 6

Introduction to Computer Systems

Hasan Al-Jawaheri

2 October, 2013

Today

- Memory Hierarchy
- Caches
- Exam Review
- Cachelab

Caching

- Temporal Locality (Time)
  - A memory location accessed is likely to be accessed again multiple times in the future
  - After accessing address X in memory, save the bytes in cache for future access
- Spatial Locality (Space)
  - If a location is accessed, then nearby locations are likely to be accessed in the future.
  - After accessing address X, save the block of memory around X in cache for future access

Memory Addresses

- A *cache* is a set of 2^S cache sets
- A *cache set* is a set of E cache lines
  - E = 1 → Direct-mapped
  - E-way associative
- Each cache line stores a block
  - Each block has 2^b bytes

Miss Rate

- Fraction of memory references not found in the cache

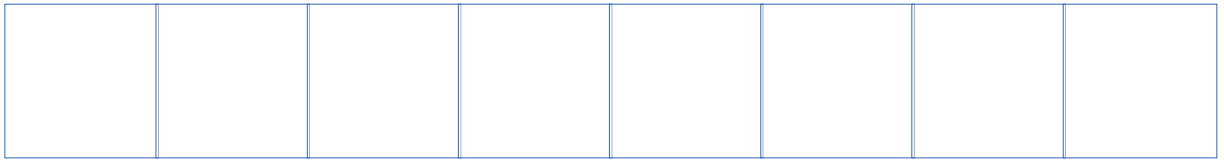
$$\text{Missrate} = 1 - \text{Hitrate}$$

Real Example!

- 8 bit-address space
- Direct-mapped 32-byte cache
  - 4 byte cache blocks

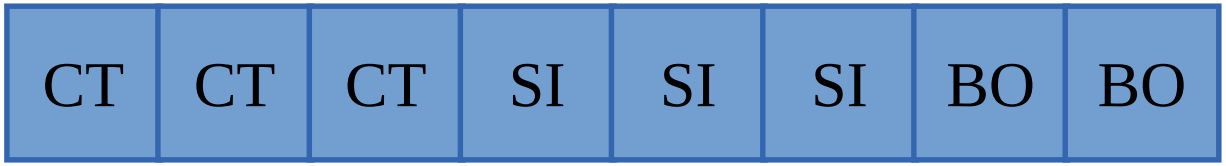
Real Example!

- Indicate which fields do the bits represent
  - “SI” → Set index
  - “BO” → Block offset
  - “CT” → Cache tag



Real Example!

- Indicate which fields do the bits represent
  - “SI” → Set index
  - “BO” → Block offset
  - “CT” → Cache tag



Real Example!

Load No.	Hex Address	Binary Address	Set Number?	Hit or Miss?
1	c7	110 001 11		
2	55	010 101 01		
3	1a	000 110 10		
4	c5	110 001 01		
5	e6	111 001 10		
6	56	010 101 10		
7	77	011 101 11		
8	28	001 010 00		
9	75	011 101 01		
10	94	100 101 00		

Real Example!

Load No.	Hex Address	Binary Address	Set Number?	Hit or Miss?
1	c7	110 001 11	1	M
2	55	010 101 01	5	M
3	1a	000 110 10	6	M
4	c5	110 001 01	1	H
5	e6	111 001 10	1	M
6	56	010 101 10	5	H
7	77	011 101 11	5	M
8	28	001 010 00	2	M
9	75	011 101 01	5	H
10	94	100 101 00	5	M

Real Example!

- Which of the following is a possible final state for the cache:

0	1	2	3	4	5	6	7
X	7	1	X	X	4	1	X

0	1	2	3	4	5	6	7
1	7	X	X	1	4	0	X

0	1	2	3	4	5	6	7
X	1	1	X	0	2	0	X

0	1	2	3	4	5	6	7
X	1	7	X	X	4	4	0

0	1	2	3	4	5	6	7
X	1	7	X	4	4	0	X

0	1	2	3	4	5	6	7
X	7	1	X	X	4	0	X

0	1	2	3	4	5	6	7
7	X	1	0	4	4	0	X

Real Example!

- Which of the following is a possible final state for the cache:

0	1	2	3	4	5	6	7
X	7	1	X	X	4	1	X

0	1	2	3	4	5	6	7
1	7	X	X	1	4	0	X

0	1	2	3	4	5	6	7
X	1	1	X	0	2	0	X

0	1	2	3	4	5	6	7
X	1	7	X	X	4	4	0

0	1	2	3	4	5	6	7
X	1	7	X	4	4	0	X

0	1	2	3	4	5	6	7
X	7	1	X	X	4	0	X

0	1	2	3	4	5	6	7
7	X	1	0	4	4	0	X

Another Example

This problem requires you to analyze the cache behavior of a function that sums the elements of an array A:

```
int A[2][4];

int sum()
{
    int i, j, sum=0;
    for (j=0; j<4; j++) {
        for (i=0; i<2; i++) {
            sum += A[i][j];
        }
    }
    return sum;
}
```

Assume the following:

- The memory system consists of registers, a single L1 cache, and main memory.
- The cache is cold when the function is called and the array has been initialized elsewhere.
- Variables i, j, and sum are all stored in registers.
- The array A is aligned in memory such that the first two array elements map to the same cache block.
- sizeof(int) == 4.
- The cache is direct mapped, with a block size of 8 bytes.

Another Example

CACHE CONSISTS OF 2 SETS

Another Example

```
int A[2][4];
int sum()
{
        int i, j, sum=0;
        for (j=0; j<4; j++) {
                for (i=0; i<2; i++) {
                        sum += A[i][j];
                }
        }
        return sum;
}
```

Address: 00 0 000
Block offset: 000
Set: 0
Tag: 00

1	**3**	**5**	**7**
2	4	6	8

Cold Miss

00	**0**	**000**

Another Example

```
int A[2][4];
int sum()
{
        int i, j, sum=0;
        for (j=0; j<4; j++) {
                for (i=0; i<2; i++) {
                        sum += A[i][j];
                }
        }
        return sum;
}
```

Address: 01 0 000
Block offset: 000
Set: 0
Tag: 01

1	**3**	**5**	**7**
2	4	6	8

Miss

01	**0**	**000**

Another Example

```
int A[2][4];
int sum()
{
        int i, j, sum=0;
        for (j=0; j<4; j++) {
                for (i=0; i<2; i++) {
                        sum += A[i][j];
                }
        }
        return sum;
}
```

Address: 00 0 100
Block offset: 100
Set: 0
Tag: 00

1	3	5	7
2	4	6	8

Miss

00	0	100

Another Example

```
int A[2][4];
int sum()
{
        int i, j, sum=0;
        for (j=0; j<4; j++) {
                for (i=0; i<2; i++) {
                        sum += A[i][j];
                }
        }
        return sum;
}
```

Address: 01 0 100
Block offset: 100
Set: 0
Tag: 01

1	**3**	**5**	**7**
2	4	6	8

Miss

01	**0**	**100**

Another Example

```
int A[2][4];
int sum()
{
        int i, j, sum=0;
        for (j=0; j<4; j++) {
                for (i=0; i<2; i++) {
                        sum += A[i][j];
                }
        }
        return sum;
}
```

Address: 00 1 000
Block offset: 000
Set: 1
Tag: 00

1	**3**	**5**	**7**
2	4	6	8

Cold Miss

01	**0**	**100**
00	1	000

Another Example

```
int A[2][4];
int sum()
{
        int i, j, sum=0;
        for (j=0; j<4; j++) {
                for (i=0; i<2; i++) {
                        sum += A[i][j];
                }
        }
        return sum;
}
```

Address: 01 1 000
Block offset: 000
Set: 1
Tag: 01

1	**3**	**5**	**7**
2	4	6	8

Miss

01	**0**	**100**
01	1	000

Another Example

```
int A[2][4];
int sum()
{
        int i, j, sum=0;
        for (j=0; j<4; j++) {
                for (i=0; i<2; i++) {
                        sum += A[i][j];
                }
        }
        return sum;
}
```

Address: 00 1 100
Block offset: 100
Set: 1
Tag: 00

1	**3**	**5**	**7**
2	4	6	8

Miss

01	**0**	**100**
00	1	100

Another Example

```
int A[2][4];
int sum()
{
        int i, j, sum=0;
        for (j=0; j<4; j++) {
                for (i=0; i<2; i++) {
                        sum += A[i][j];
                }
        }
        return sum;
}
```

Address: 01 1 100
Block offset: 100
Set: 1
Tag: 01

1	**3**	**5**	**7**
2	4	6	8

Miss

01	**0**	**100**
01	1	100

Another Example

```
int A[2][4];
int sum()
{
        int i, j, sum=0;
        for (j=0; j<4; j++) {
                for (i=0; i<2; i++) {
                        sum += A[i][j];
                }
        }
        return sum;
}
```

Everything missed!

Another Example

CACHE CONSISTS OF 4 SETS

Another Example

- Problem in previous cache:
  - Every time i changes, the new address has the same set but a different tag
  - Then we will miss every time

1	**3**	**5**	**7**
2	4	6	8

- New cache:
  - Every time i changes, the new address has a different set
  - But notice, for 3, 4, 7 and 8 we won't miss because 1, 2, 5 and 6 (respectively) loaded them.
  - Miss rate is only half!

1	**3**	**5**	**7**
2	4	6	8

Cachelab

- Due: Sometime after Eid \o/
- Office Hours/Piazza

Final

- Integers
  - Edge cases
  - Casting
  - Endianness
- Floating Points
- Assembly
  - How loops work
  - How addressing and dereferencing work
  - Typical address calculation scheme
  - How to use the book for quick function lookup
- Structs and unions
- Stack
  - Frames
  - Stack management
  - Caller/Callee roles
- Caches

FOCUS ON THE DETAILS!