15-213 Recitation 7
Caches and Blocking

<TA Names>

26th February 2018

Agenda

- Caching Review
- Blocking to reduce cache misses
- Cache alignment

Reminders

- Cache Lab is due Thursday!
- Exam 1 is next week!! (Week of March 5th)
- Start doing practice problems.
- Come to the review session.

What Type of Locality?

- The following function exhibits which type of locality? Consider *only* array accesses.

```
void who(int *arr, int size) {
  for (int i = 0; i < size-1; ++i)
    arr[i] = arr[i+1];
}
```

A.	Spatial
B.	Temporal
C	Both A and B
D.	Neither A nor B

What Type of Locality?

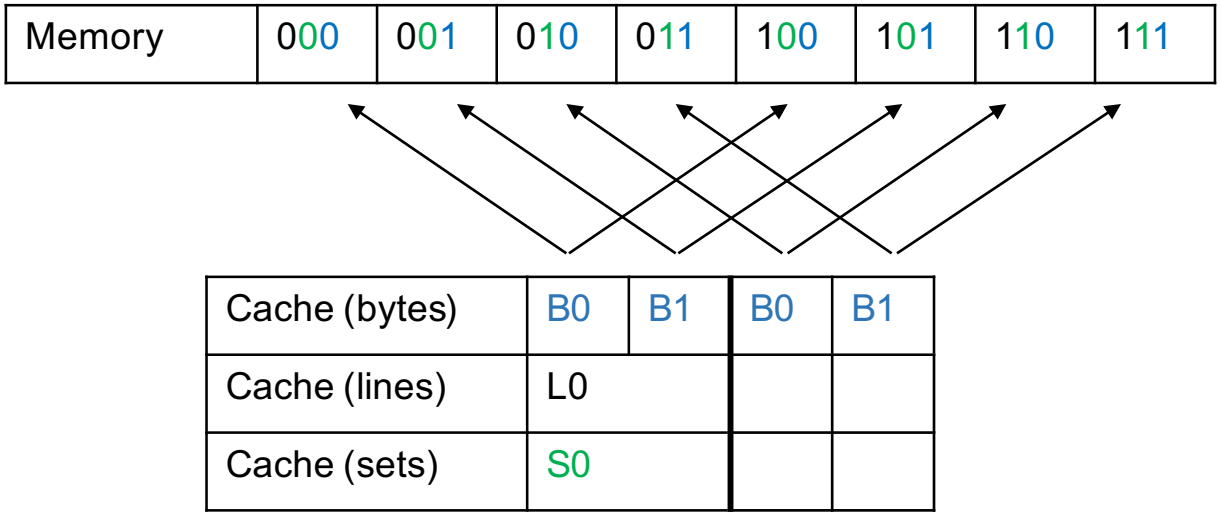
- The following function exhibits which type of locality? Consider *only* array accesses.

```
void coo(int *arr, int size) {
  for (int i = size-2; i >= 0; --i)
    arr[i] = arr[i+1];
}
```

A.	Spatial
B.	Temporal
C	Both A and B
D.	Neither A nor B

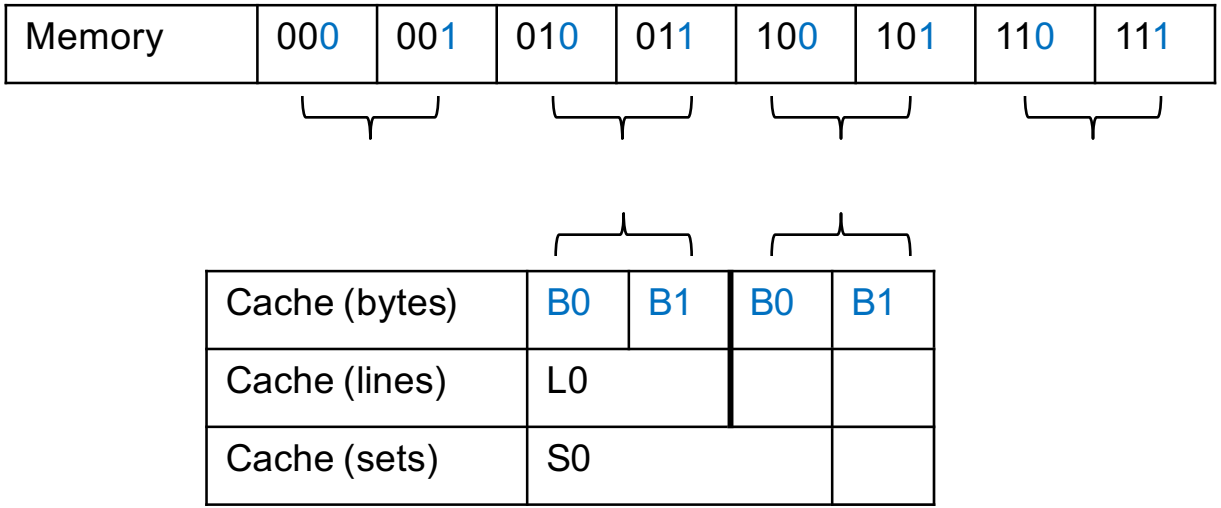
Interlude: terminology

- A **direct-mapped** cache only contains one line per set. This means $E = 2^e = 1$.



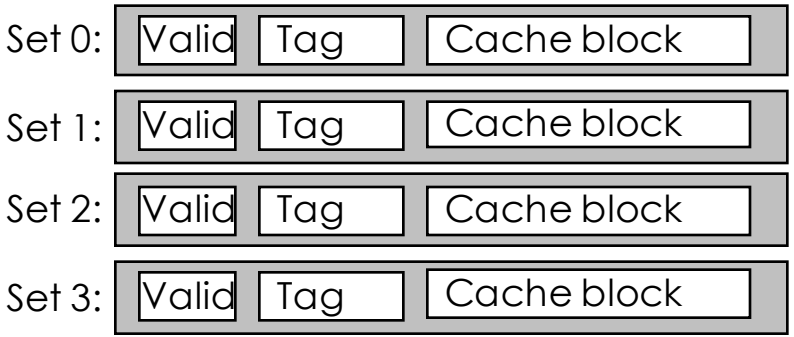
Interlude: terminology

- A **fully associative** cache has 1 set, and many lines for that one set. This means S = 2^s = 1.



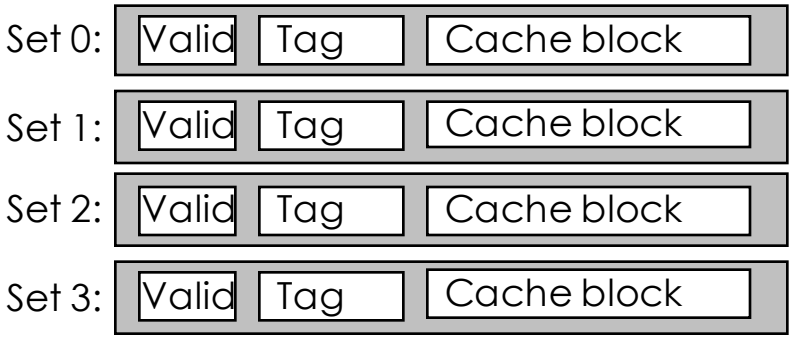
Direct-Mapped Cache Example

- Assuming a 32-bit address (i.e. m=32):
- Assume the cache is direct-mapped, and each block stores 8 bytes, and there are 4 sets
- how many bits are used for tag (t), set index (s), and block offset (b).



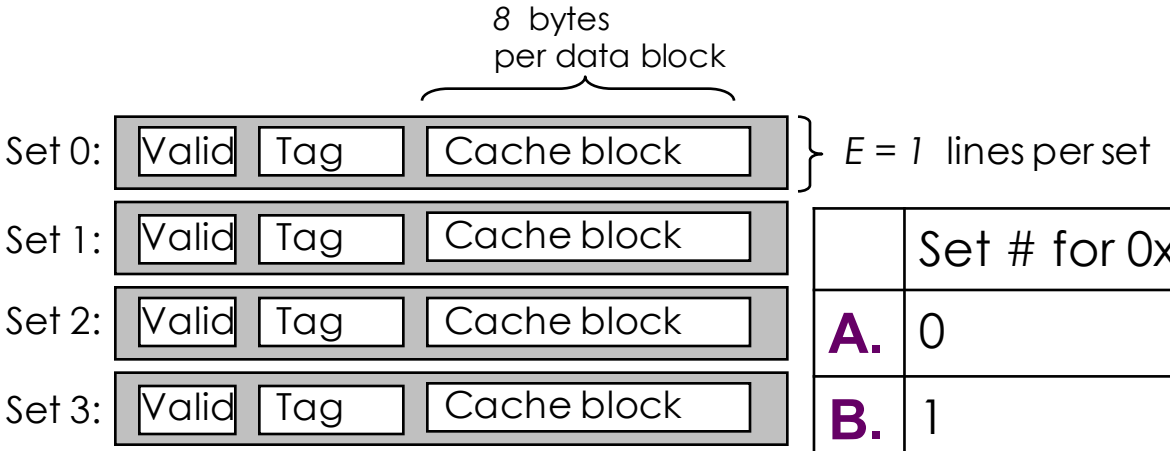
Direct-Mapped Cache Example

- Assuming a 32-bit address (i.e. m=32):
- Assume the cache is direct-mapped, and each block stores 8 bytes, and there are 4 sets
- how many bits are used for tag (t), set index (s), and block offset (b).
- t = 27, s = 2, b = 3



Which Set Is it?

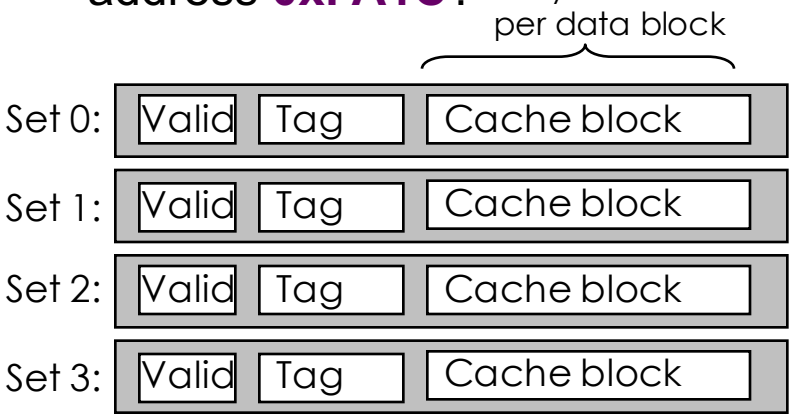
- Which set may the address **0xFA1C** be located in?

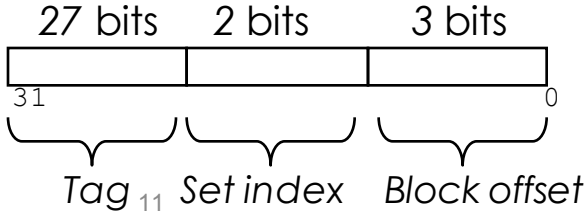


	Set # for 0xFA1C
A.	0
B.	1
C.	2
D.	3
E.	More than one of the above

Cache Block Range

- What range of addresses will be in the same block as address **0xFA1C**?





	Addr. Range
A.	0xFA1C
B.	0xFA1C – 0xFA23
C.	0xFA1C – 0xFA1F
D	0xFA18 – 0xFA1F
E.	It depends on the access size (byte, word, etc)

Cache Misses

If N = 16, how many bytes does the loop access of A?

```
int foo(int* a, int N)
{
    int i, sum = 0;
    for(i = 0; i < N; i++)
        sum += a[i];
    return sum;
}
```

	Accessed Bytes
A	4
B	16
C	64
D	256

Cache Misses

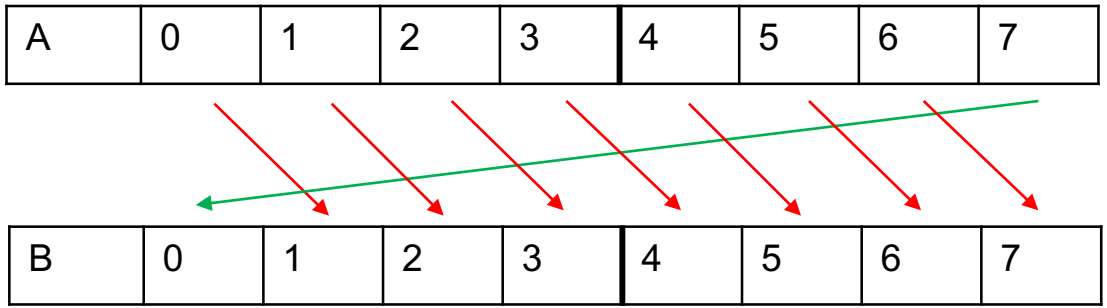
If there is a 48B cache with 8 bytes per block and 3 cache lines per set, how many misses if foo is called twice?
N still equals 16

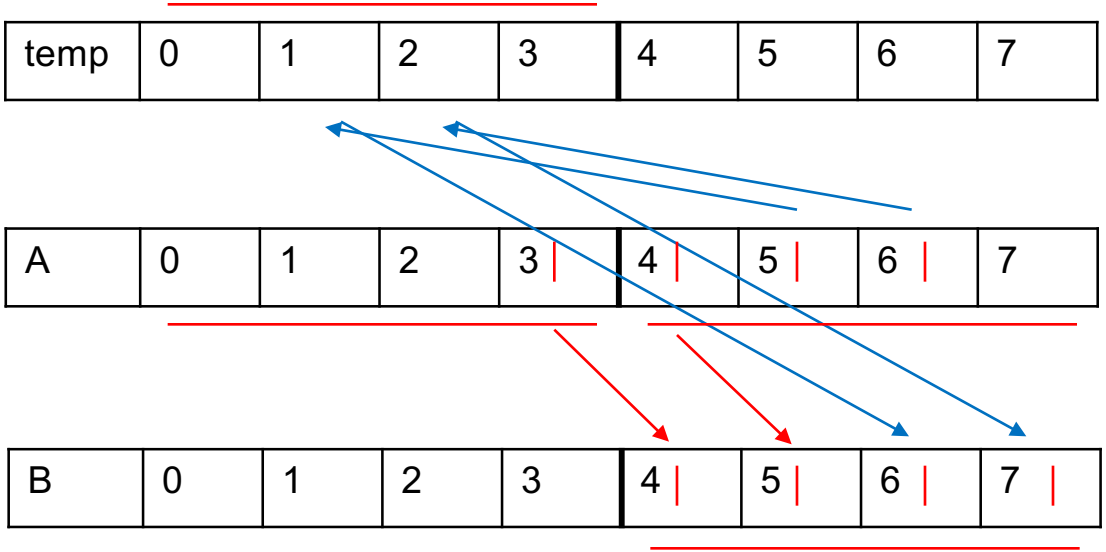
```
int foo(int* a, int N)
{
    int i, sum = 0;
    for(i = 0; i < N; i++)
        sum += a[i];
    return sum;
}
```

	Misses
A	0
B	8
C	12
D	14
E	16

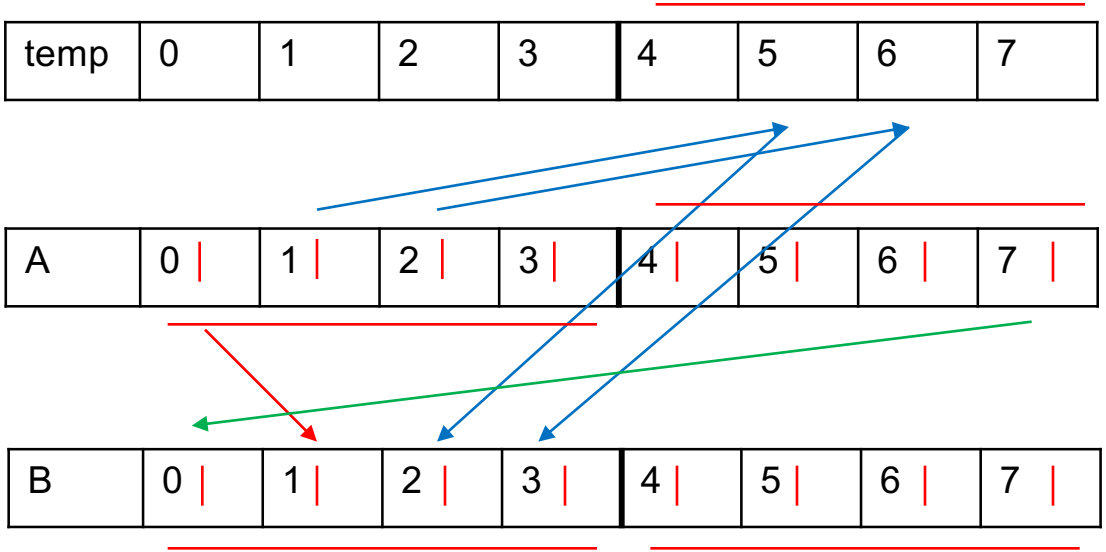
Very Hard Cache Problem

- We will use a direct-mapped cache with 2 sets, which each can hold up to 4 `int`'s.
- How can we copy A into B, shifted over by 1 position?
  - The most efficient way? (Use `temp`!)





Number of misses:



Number of misses:

← Could've been 16 misses otherwise!
We would save even more if the block size were larger, or if `temp` were already cached

If You Get Stuck

Please read the writeup

Read it again after doing ~25% of the lab

- CS:APP Chapter 6
- View lecture notes and course FAQ at

http://www.cs.cmu.edu/~213

- Office hours Sunday through Friday (Generally) 5:00-9:00pm in WeH 5207
- Post a **private** question on Piazza
- `man malloc`, `man gdb`, gdb's `help` command
- http://csapp.cs.cmu.edu/public/waside/waside-blocking.pdf

Appendix: C Programming Style

- Properly document your code
  - Header comments, overall operation of large blocks, any tricky bits
- Write robust code – check error and failure conditions
- Write modular code
  - Use interfaces for data structures, e.g. create/insert/remove/free functions for a linked list
  - No magic numbers – use #define
- Formatting
  - 80 characters per line
  - Consistent braces and whitespace
- No memory or file descriptor leaks