CSE 351: Week 7

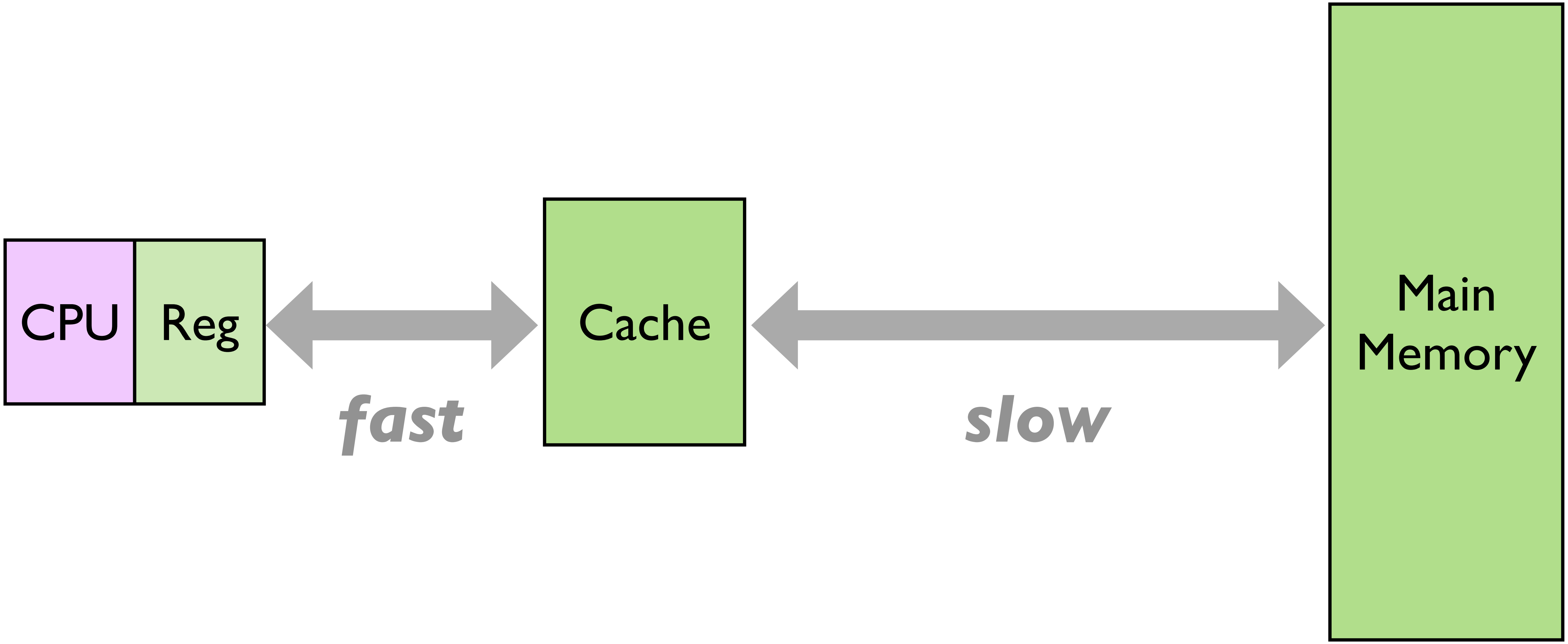
Tom Bergan, TA

Today

- Cache geometries
- Lab 4

Caches

they make memory faster



Tradeoff:
caches are smaller than main memory

Why do caches work?

Temporal locality:

```
int global;
...
for (...) {
   global++;
```

same variable accessed in each loop iteration

Spatial locality:

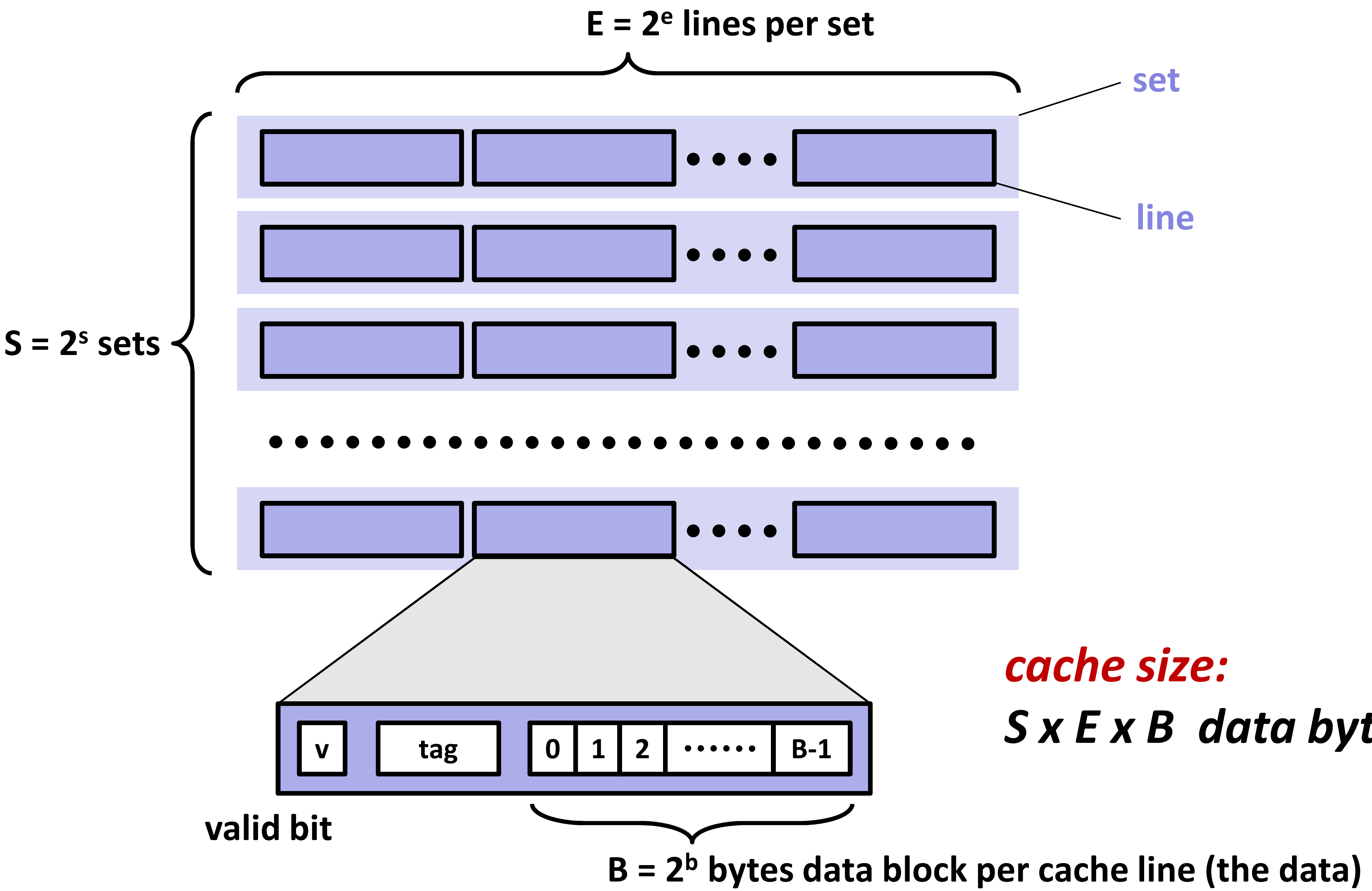
```
struct Point p;
p.x = 5;
p.y = 6;
```

fields of same struct accessed together

```
int a[10];
for (i=0; i<10; ++i)
  a[i] = i*2;
```

adjacent elements accessed consecutively

What a cache looks like



cache size:

S x E x B data bytes

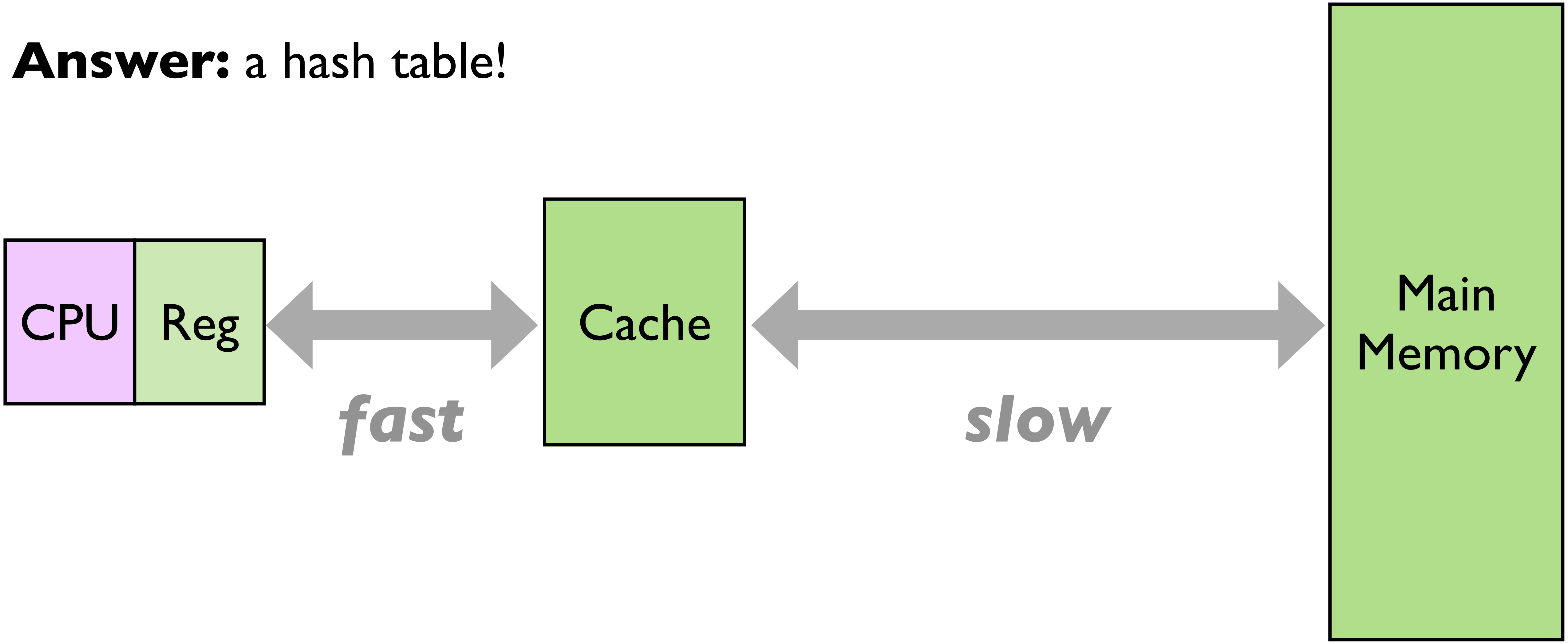
(figure from lecture slides)

Let’s backup and see how they came up with that ...

What data structure should we use for a cache?

Important: caches must be *fast*

Answer: a hash table!



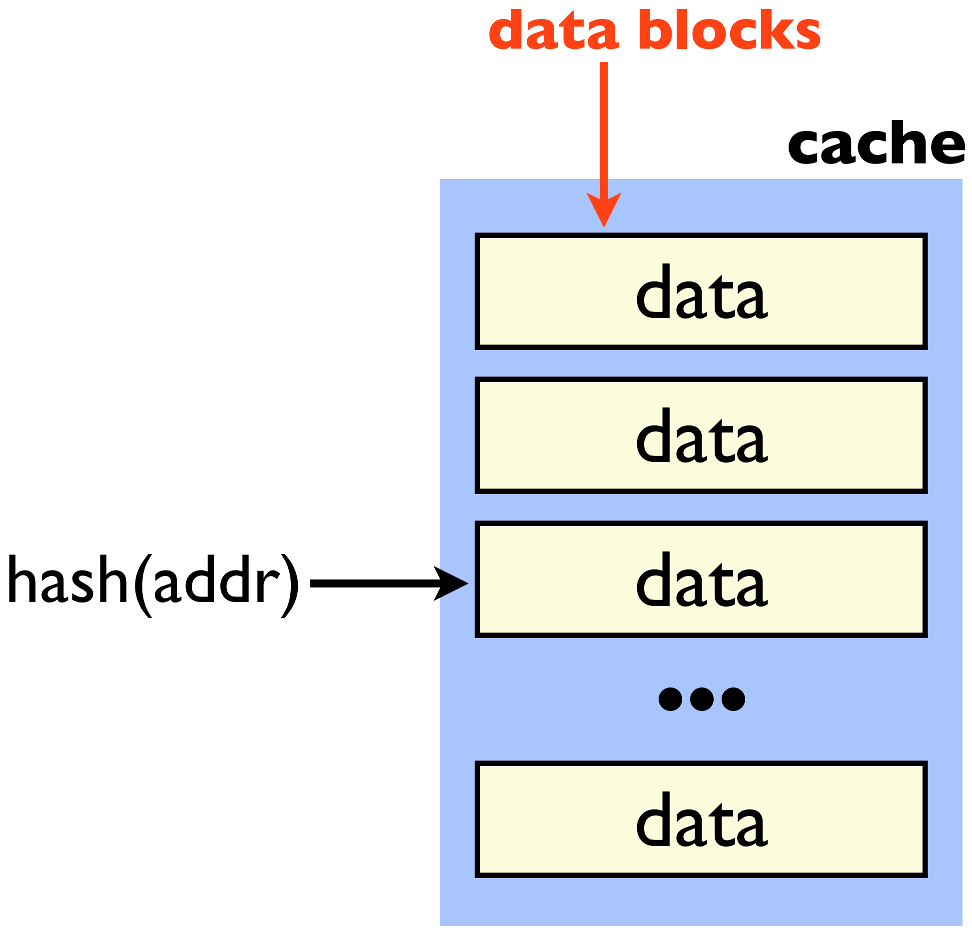
What a cache looks like

A cache is just a *fixed-size* hash table!

key: address

value: data at that address

Size of a data block is configurable

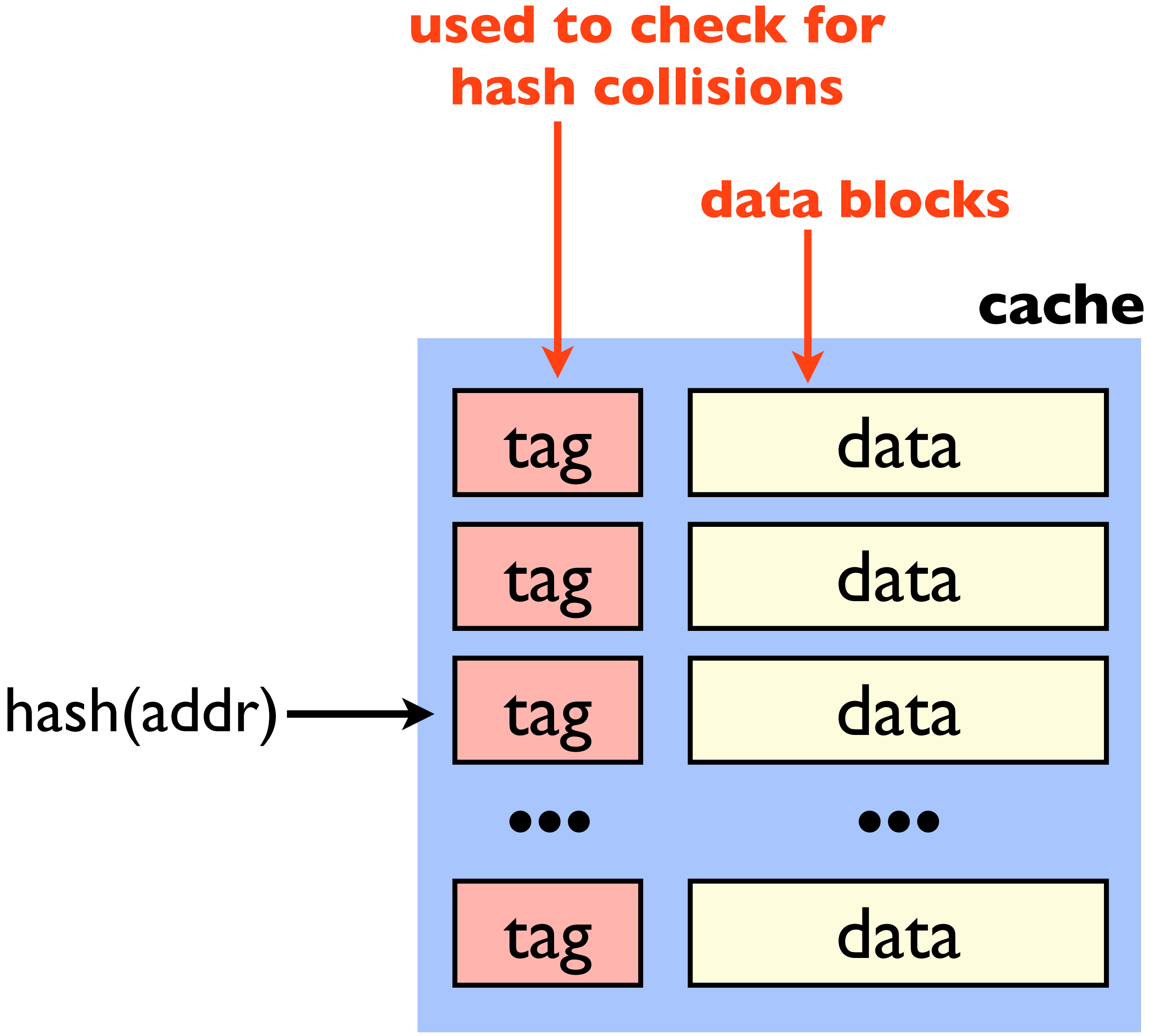


What a cache looks like

A cache is just a *fixed-size* hash table!

key: address

value: data at that address



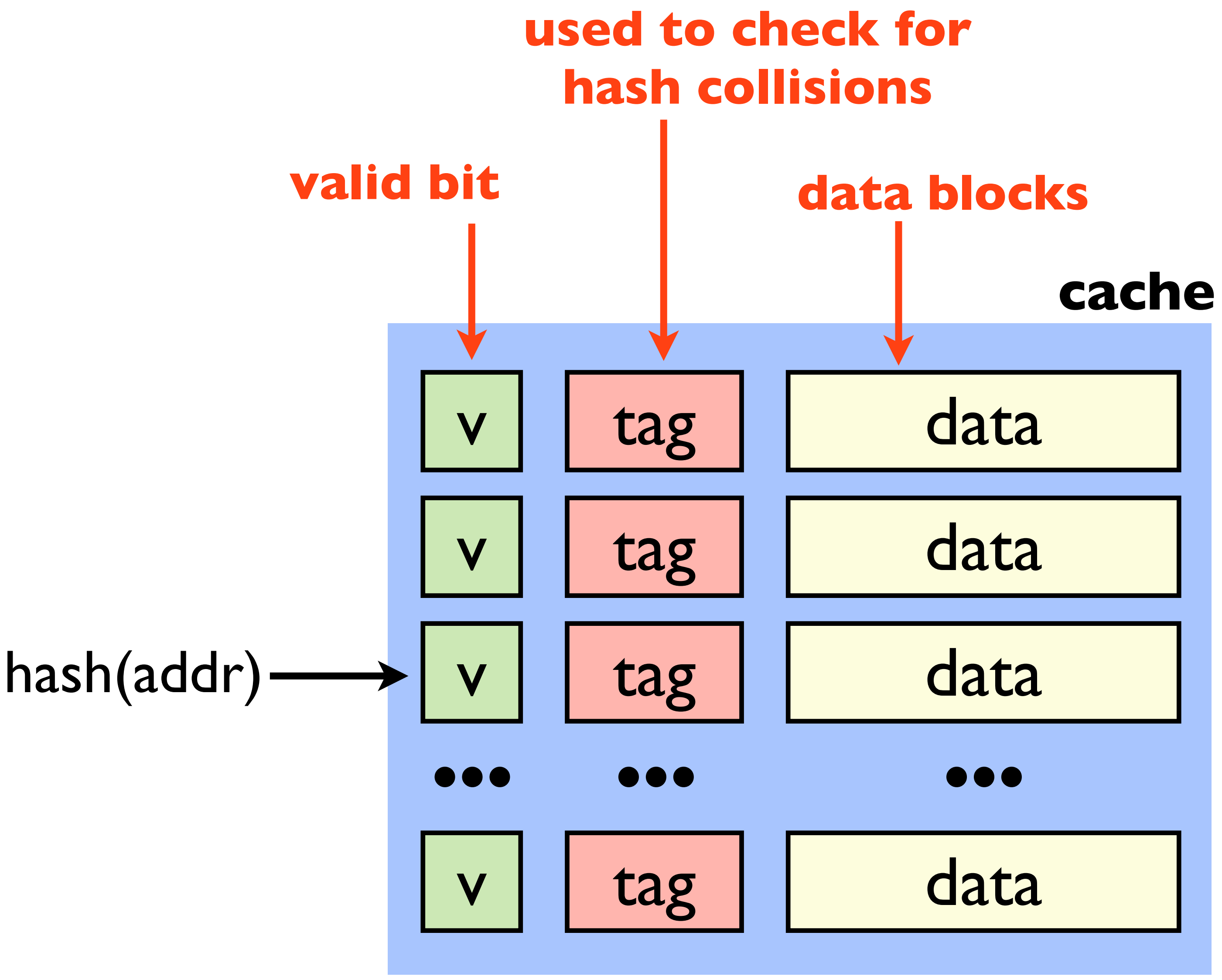
What a cache looks like

A cache is just a *fixed-size* hash table!

key: address

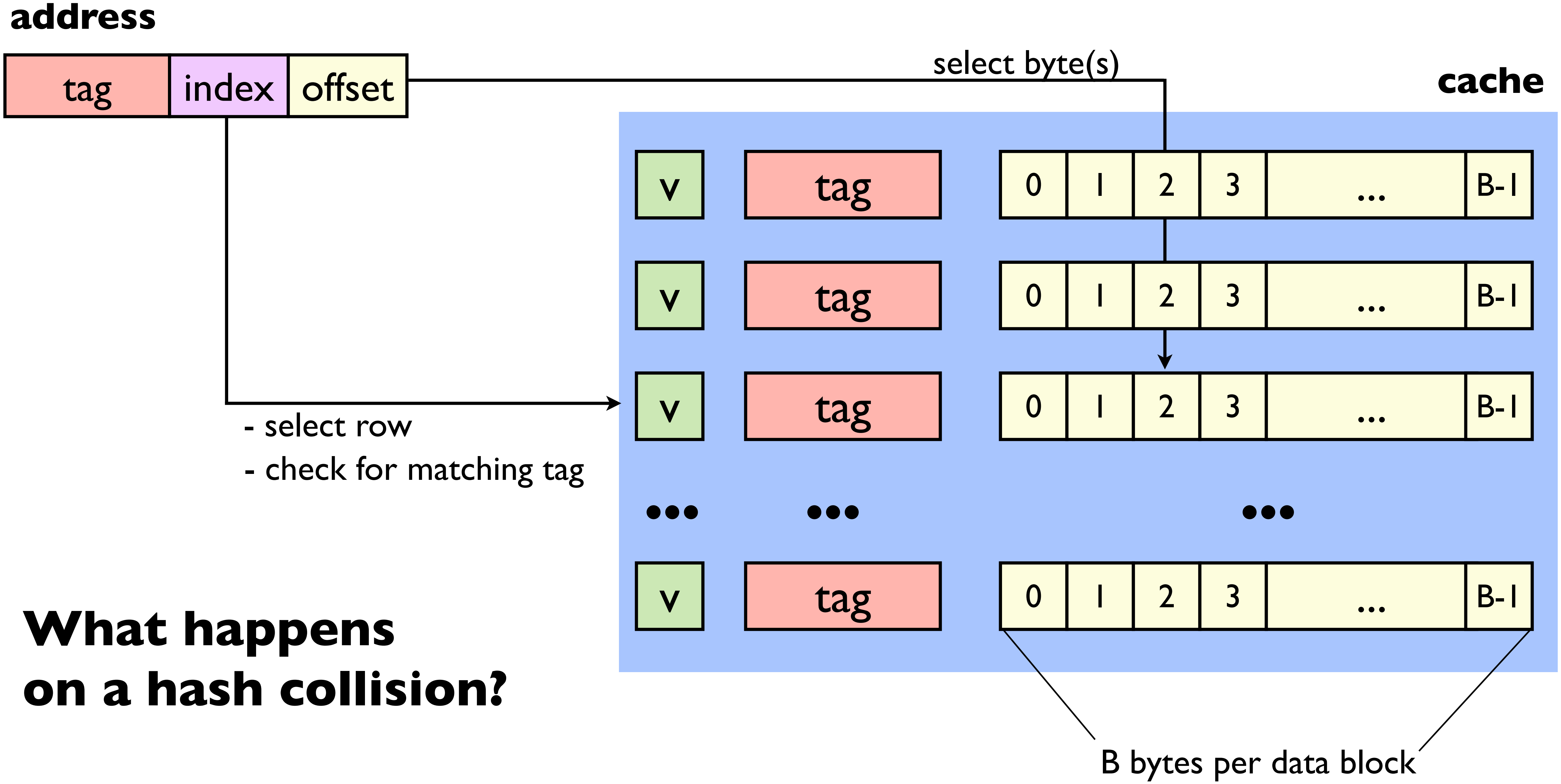
value: data at that address

What hash function should we use?



What a cache looks like

(direct mapped cache)



- select row
- check for matching tag

B bytes per data block

What happens on a hash collision?

Pathological Case

(direct mapped cache)

A simple program:

```
int a[64];
int b[64];
for (i=0; i<64; ++i)
   b[i] = a[i];
```

What if:

	tag	index	offset
&a = 0x	000A	020	00
&b = 0x	000B	020	00

There will be a cache miss on every access!

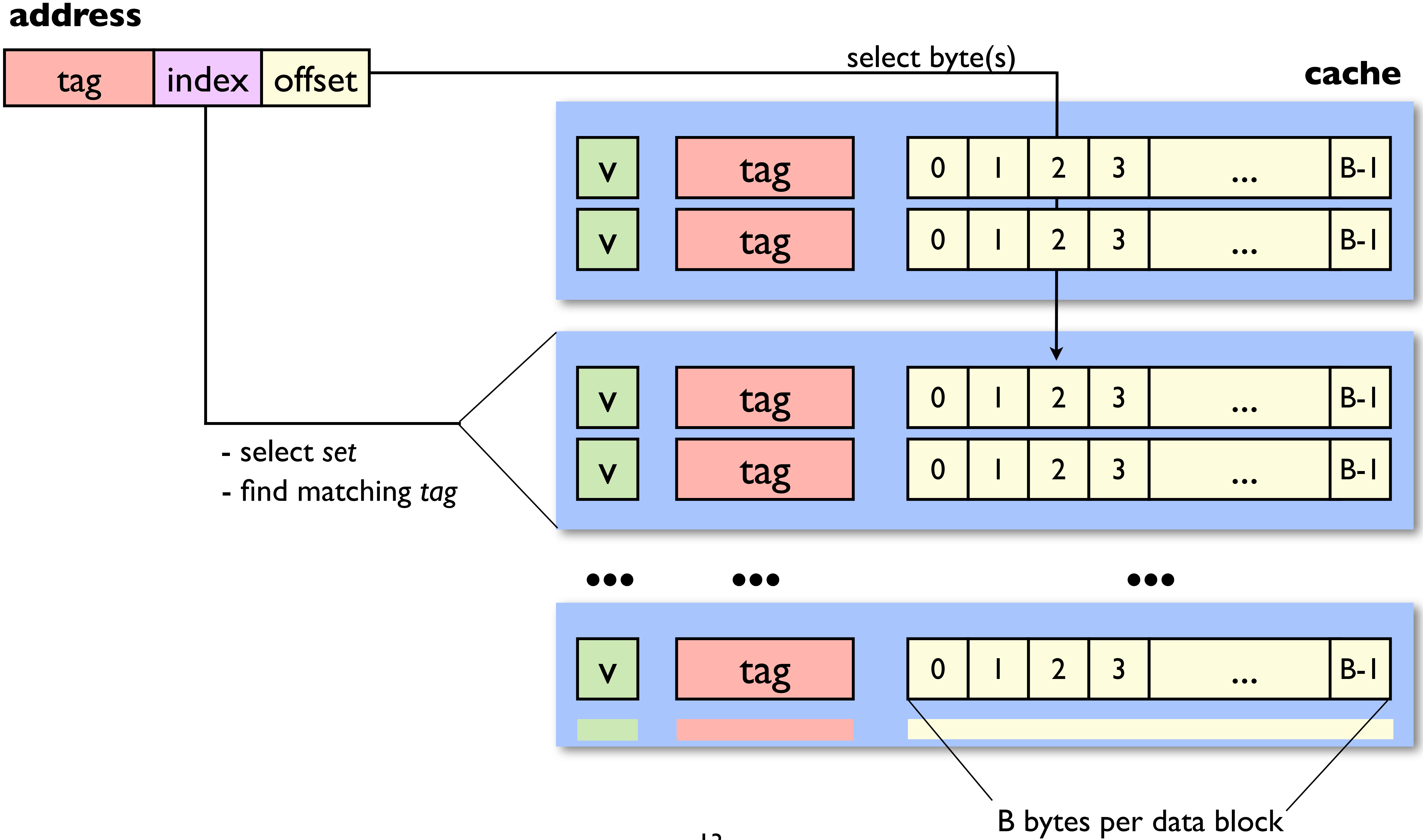
&a[0] = 0x	**000A**	020	00
&b[0] = 0x	**000B**	020	00
&a[1] = 0x	**000A**	020	01
&b[1] = 0x	**000B**	020	01

← **Note the alternating tags**

Solution: *associative sets*

What a cache looks like

(set associative cache)



address

select byte(s)

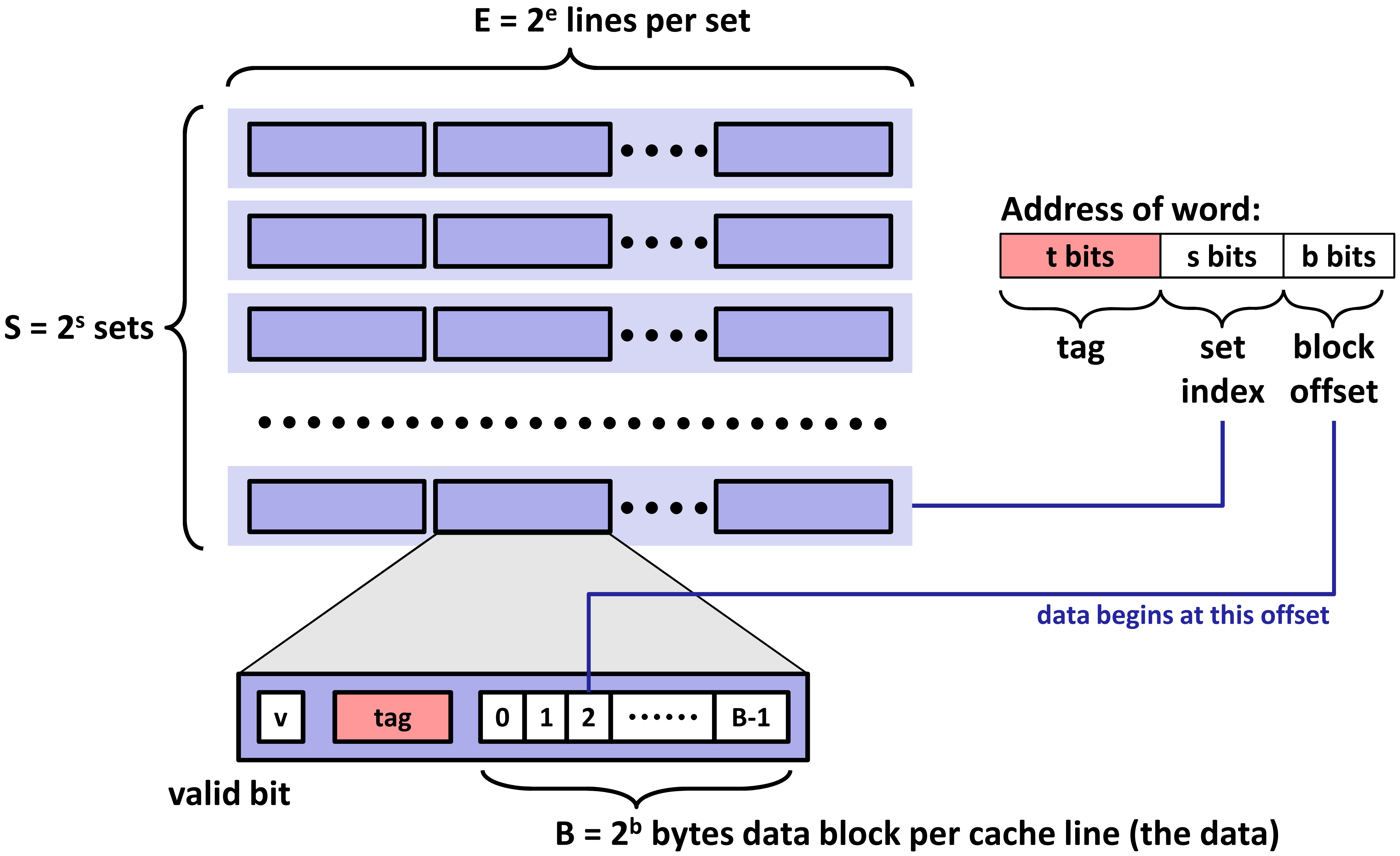
cache

- select *set*
- find matching *tag*

B bytes per data block

What a cache looks like

(set associative cache)



(figure from lecture slides)

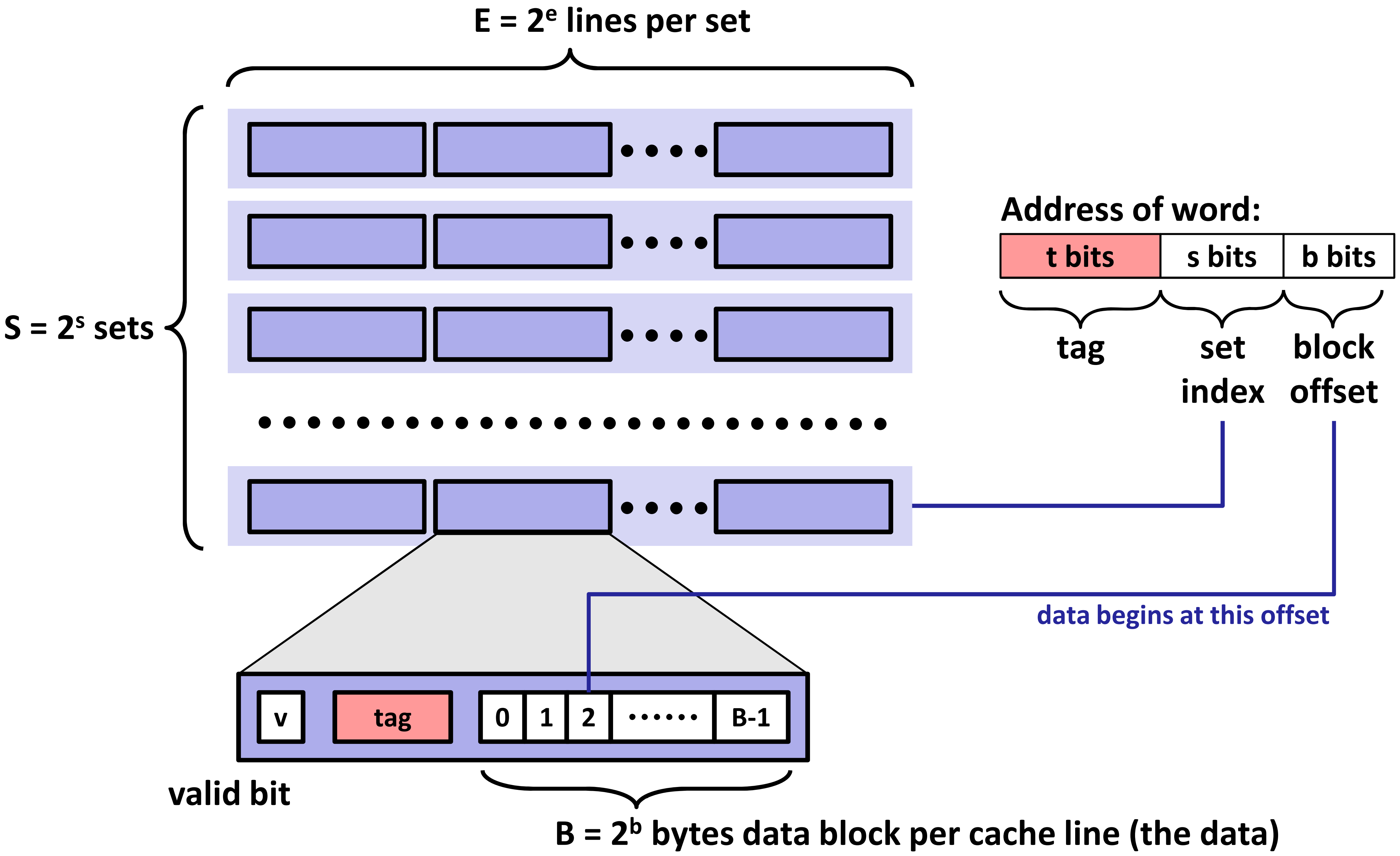
Lab 4: You measure the geometry

- We give you:
  - `flush_cache()`
  - `access_cache(addr)`

- You measure:

  `B:` bytes per block
  `E:` lines per set
  `S×E×B:` total size of the cache

Lab 4: You measure B, E, and S×E×B



(figure from lecture slides)

What does a cache-unfriendly program look like?

Friendly:

```
int a[64][64];

for (i=0; i<64; ++i)
  for (k=0; k<64; ++k)
    a[i][k]++;
```

Unfriendly:

```
int a[64][64];

for (i=0; i<64; ++i)
  for (k=0; k<64; ++k)
    a[k][i]++;
```

What does a cache-unfriendly program look like?

Friendly:

```
struct Point {
  int x;
  int y;
};
struct Point a[64];

for (i=0; i<64; ++i)
  a[i].x += a[i].y;
```

Unfriendly:

```
struct Points {
  int x[64];
  int y[64];
};
struct Points a;

for (i=0; i<64; ++i)
  a.x[i] += a.y[i];
```

When might this be better?

What does a cache-unfriendly program look like?

Which one is friendly depends on access patterns

```
struct Stuff {
  int x;
  char str[64];
  int y;
};
```

Good when str accessed frequently

```
struct Stuff {
  int x;
  char *str;
  int y;
};
```

Good when str accessed rarely