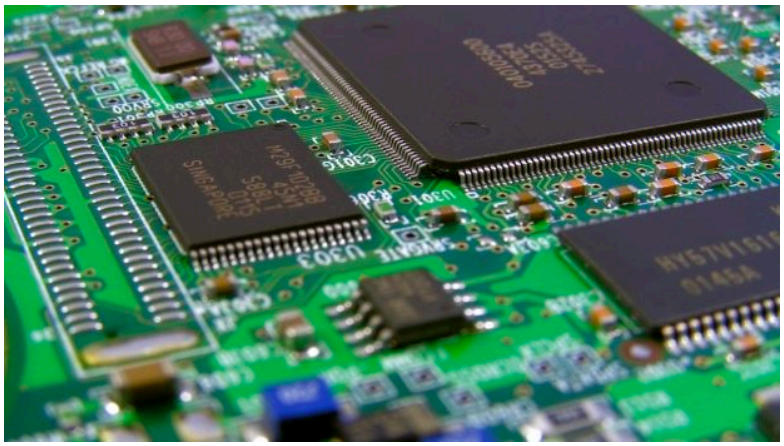
Caching



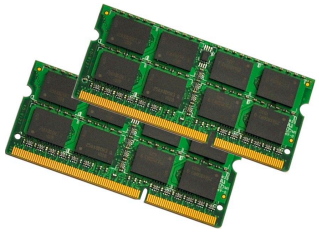
Data Storage

Disks

- Hard disk (HDD)
- Solid state drive (SSD)

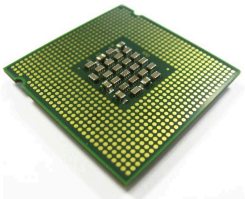
Random Access Memory

- Dynamic RAM (DRAM)
- Static RAM (SRAM)

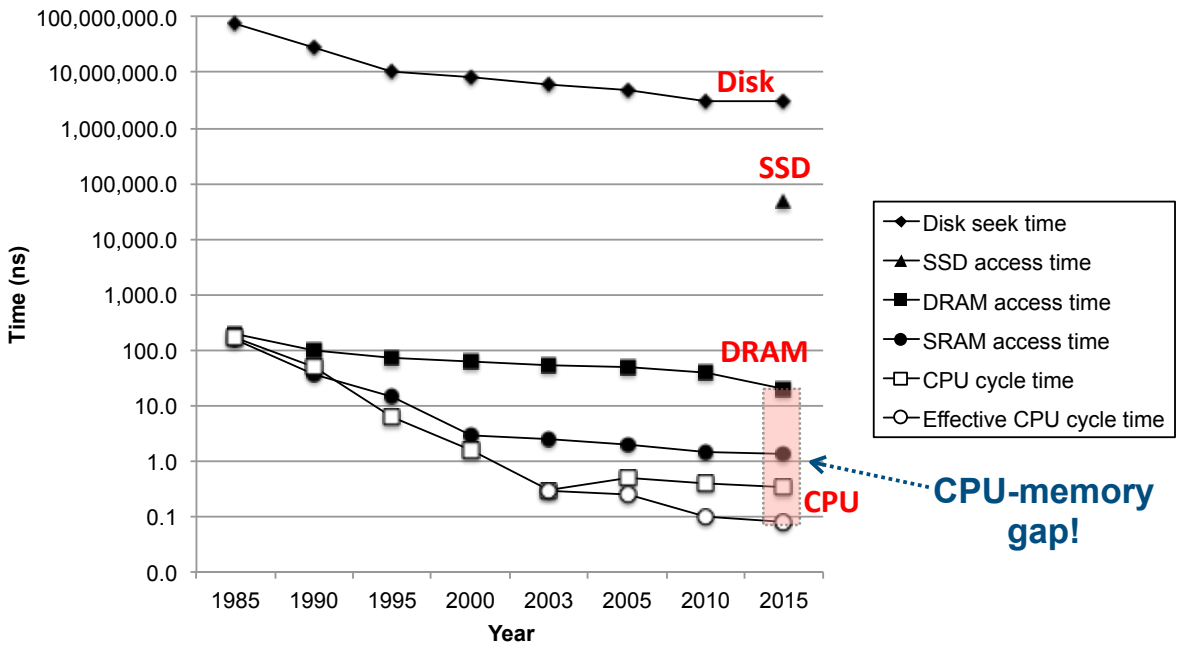


Registers

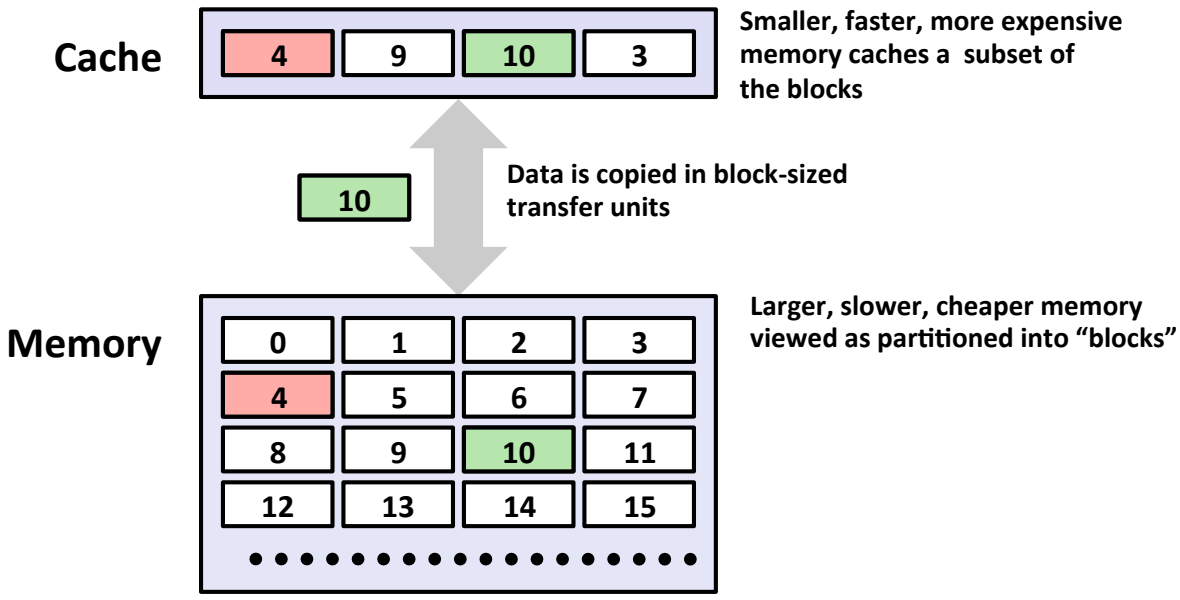
- `%rax`, `%rbx`, ...



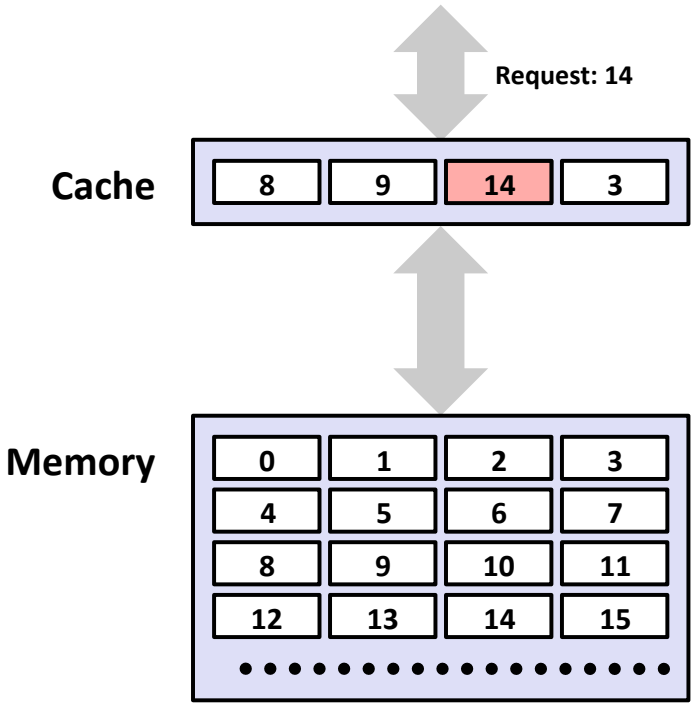
Storage Access Times



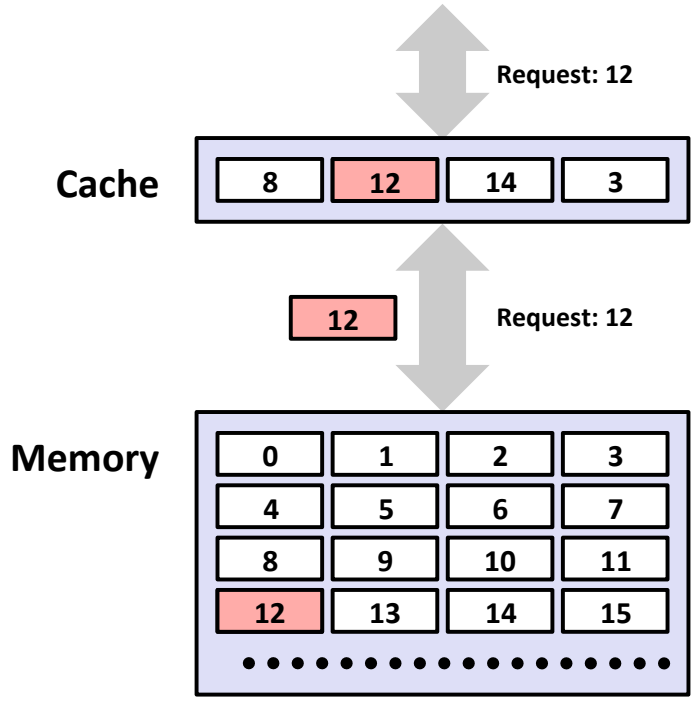
Caching Overview



Cache Hit



Cache Miss



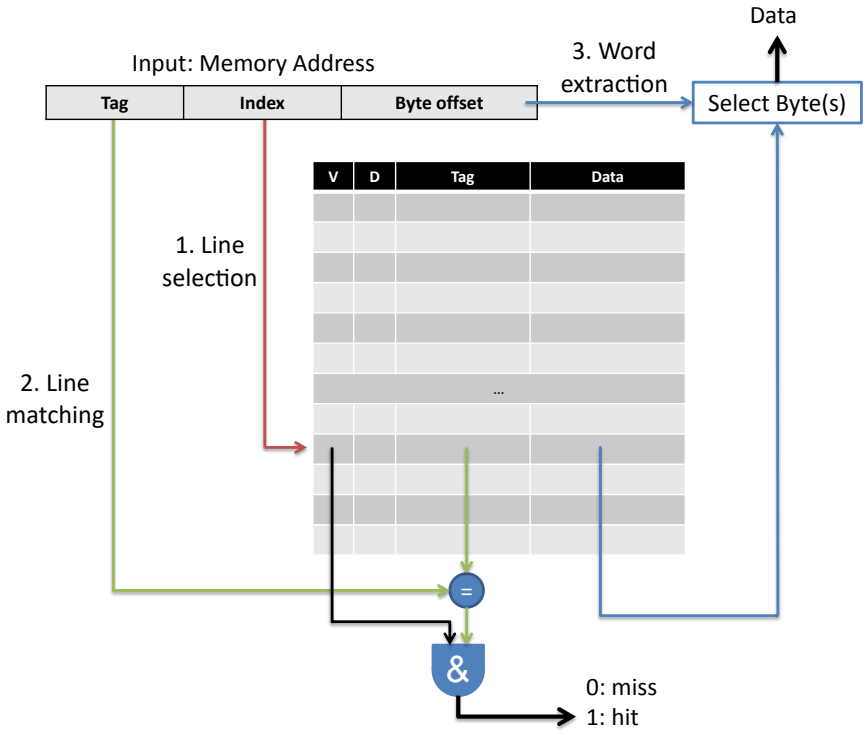
Direct-Mapped Cache

1024 cache lines
× 8-byte blocks
= 8 KB cache size

valid bit **dirty bit** (assumes write-back)

Line	V	D	Tag	Data (8 Bytes)
0				
1				
2				
3				
4				
…			…	
1020				
1021				
1022				
1023				

Direct-Mapped Cache Lookup



Direct-Mapped Address Bits

(assumes 32-bit addresses)

Tag (19 bits)	Index (10 bits)	Byte offset (3 bits)

Line	V	D	Tag	Data (8 Bytes)
0				
1				
2				
3				
4				
…			…	
1020				
1021				
1022				
1023				

Address Bits Example

Machine Properties:
4-bit addresses

Cache Properties:
2 cache lines
4-byte blocks

Line	V	D	Tag	Data (4 Bytes)
0				
1				

	Address	Address Bits (tag index offset)
Block 0	0	000000
	1	000001
	2	000100
	3	000111
Block 1	4	001000
	5	001001
	6	001100
	7	001111
Block 2	8	110000
	9	110001
	10	110100
	11	110111
Block 3	12	111000
	13	111001
	14	111100
	15	111111

Address Bits Example (cont.)

5-bit addresses

(same cache)

	Address	Address Bits (tag index offset)
Block 0	0	00 0 00
	1	00 0 01
	2	00 0 10
	3	00 0 11
Block 1	4	00 1 00
	5	00 1 01
	6	00 1 10
	7	00 1 11
Block 2	8	01 0 00
	9	01 0 01
	10	01 0 10
	11	01 0 11
Block 3	12	01 1 00
	13	01 1 01
	14	01 1 10
	15	01 1 11

	Address	Address Bits (tag index offset)
Block 4	16	10 0 00
	17	10 0 01
	18	10 0 10
	19	10 0 11
Block 5	20	10 1 00
	21	10 1 01
	22	10 1 10
	23	10 1 11
Block 6	24	11 0 00
	25	11 0 01
	26	11 0 10
	27	11 0 11
Block 7	28	11 1 00
	29	11 1 01
	30	11 1 10
	31	11 1 11

Direct-Mapped Address Components

Address **0x20F2022**

(assumes 32-bit addresses)

Tag (19 bits)	Index (10 bits)	Byte offset (3 bits)
4217	4	2

Line	V	D	Tag	Data (8 Bytes)
0				
1				
2				
3				
4	1		4217	
...			...	
1020				
1021				
1022				
1023				

Caching Exercises

tag | index | offset

Line	V	D	Tag	Data (4 Bytes)
0	1	0	111	17
1	1	0	011	9
2	0	0	101	15
3	1	1	001	8
4	1	0	011	4
5	0	0	111	6
6	0	0	101	32
7	1	0	110	3

Direct-Mapped Cache (redux)

(assumes 32-bit addresses)

Tag (19 bits)	Index (10 bits)	Byte offset (3 bits)

1024 cache lines
× 8-byte blocks
= 8 KB cache size

Line	V	D	Tag	Data (8 Bytes)
0				
1				
2				
3				
4				
…	…			
1020				
1021				
1022				
1023				

2-Way Set Associative Cache

512 **cache sets** × 2 lines per set
× 8-byte blocks
= same 8 KB cache size

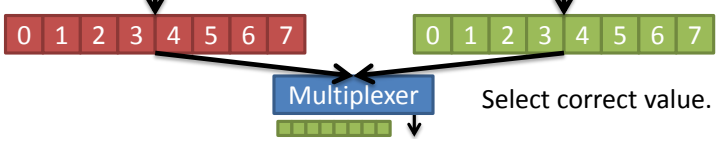
Tag (20 bits)	Set (9 bits)	Byte offset (3 bits)
3941	4	

Set #	V	D	Tag	Data (8 Bytes)	V	D	Tag	Data (8 Bytes)
0								
1								
2								
3								
4	1	1	4063		1	0	3941	
…			…				…	
508								
509								
510								
511								

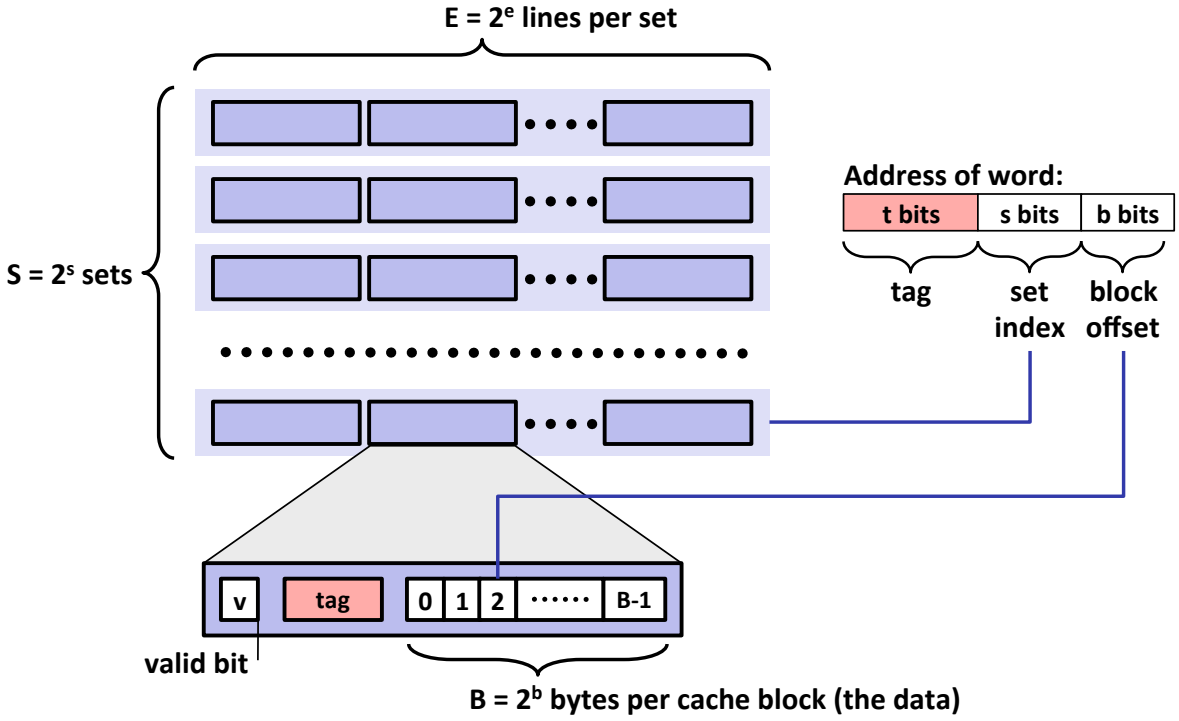
2-Way Set Associative Line Matching

Tag (20 bits)	Set (9 bits)	Byte offset (3 bits)
3941	4	

Set #	V	D	Tag	Data (8 Bytes)	V	D	Tag	Data (8 Bytes)
0								
1								
2								
3								
4	1	1	4063		1	0	3941	
…			…				…	
508								
509								
510								
511								

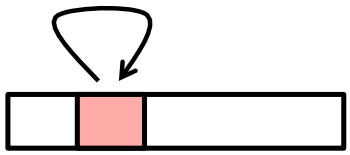


General Cache Model (S, E, B)

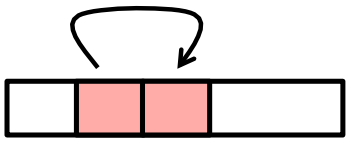


Locality

- **Temporal locality:**



- **Spatial locality:**



Locality Example

```
sum = 0;
for (i = 0; i < n; i++)
    sum += a[i];
return sum;
```


Locality Design

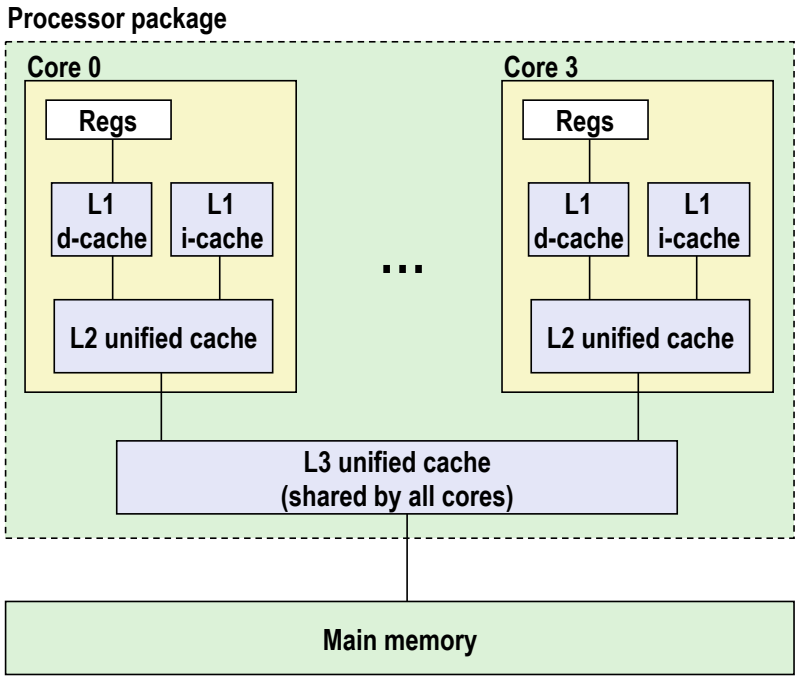
(v1)

```
int sum_array_rows(int a[M][N]) {
    int i, j, sum = 0;

    for (i = 0; i < M; i++)
        for (j = 0; j < N; j++)
            sum += a[i][j];
    return sum;
}
```

(v2)

```
int sum_array_cols(int a[M][N]) {
    int i, j, sum = 0;

    for (j = 0; j < N; j++)
        for (i = 0; i < M; i++)
            sum += a[i][j];
    return sum;
}
```

Intel Core i7 Cache Hierarchy (2015)



L1 i-cache and d-cache:
32 KB, 8-way,
Access: 4 cycles

L2 unified cache:
256 KB, 8-way,
Access: 10 cycles

L3 unified cache:
8 MB, 16-way,
Access: 40-75 cycles

Block size: 64 bytes for all caches.

The Memory Hierarchy