A sequência destes slides foi alterada

15-213

"The course that gives CMU its Zip!"

Cache Memories
Oct. 10, 2002

Topics

- Generic cache memory organization
- Direct mapped caches
- Set associative caches
- Impact of caches on performance

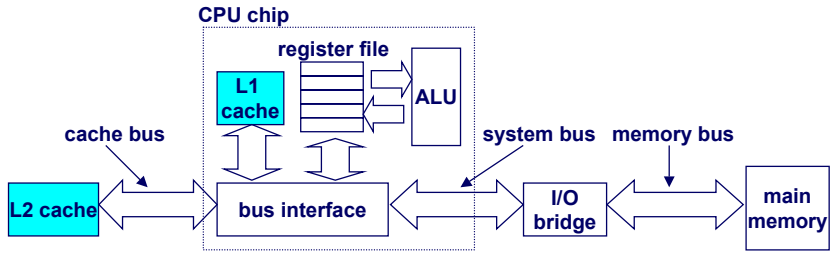
Cache Memories

Cache memories are small, fast SRAM-based memories managed automatically in hardware.

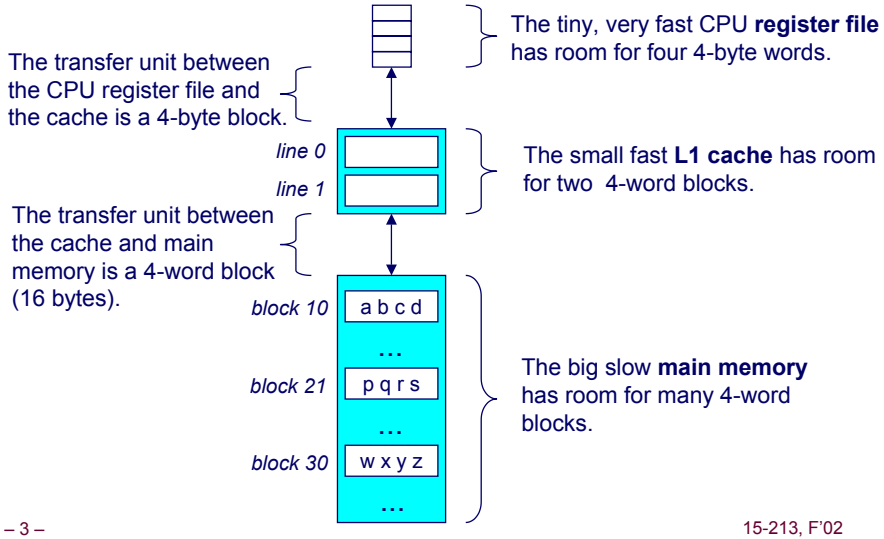
- Hold frequently accessed blocks of main memory

CPU looks first for data in L1, then in L2, then in main memory.

Typical bus structure:



Inserting an L1 Cache Between the CPU and Main Memory

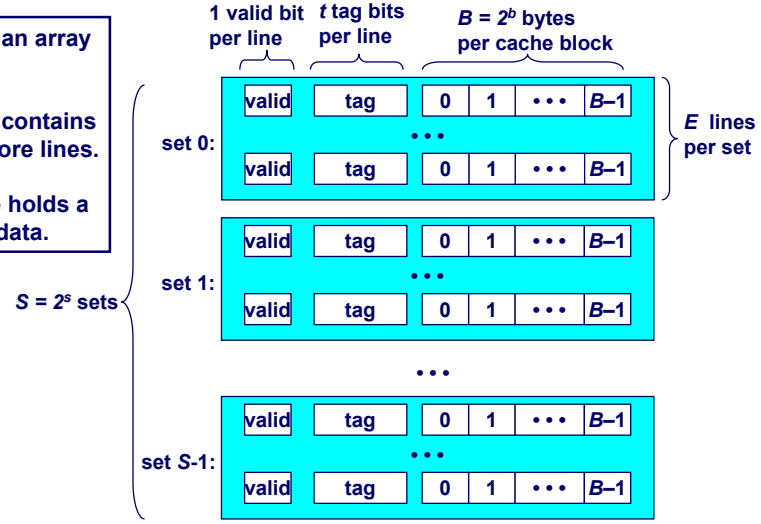


General Org of a Cache Memory

Cache is an array of sets.

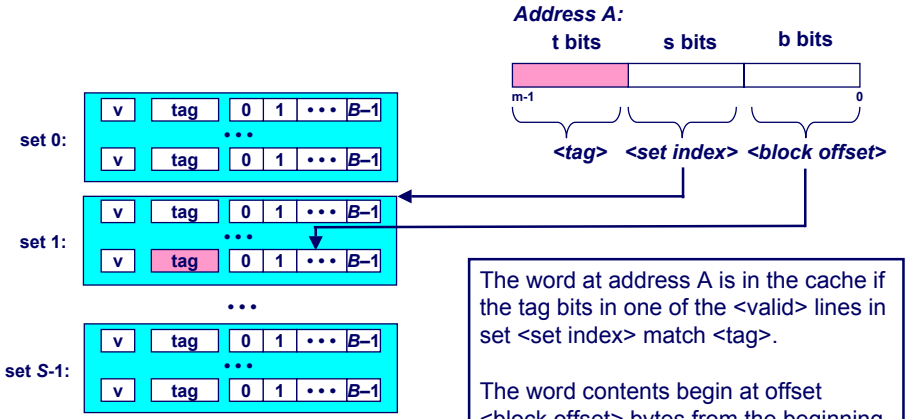
Each set contains one or more lines.

Each line holds a block of data.



Cache size: $C = B \times E \times S$ data bytes

Addressing Caches



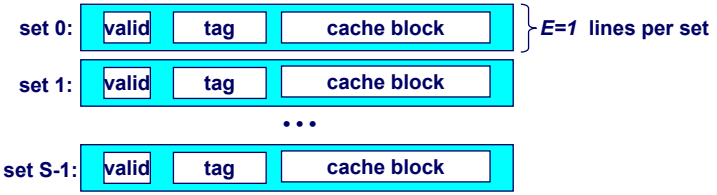
The word at address A is in the cache if the tag bits in one of the <valid> lines in set <set index> match <tag>.

The word contents begin at offset <block offset> bytes from the beginning of the block.

Direct-Mapped Cache

Simplest kind of cache

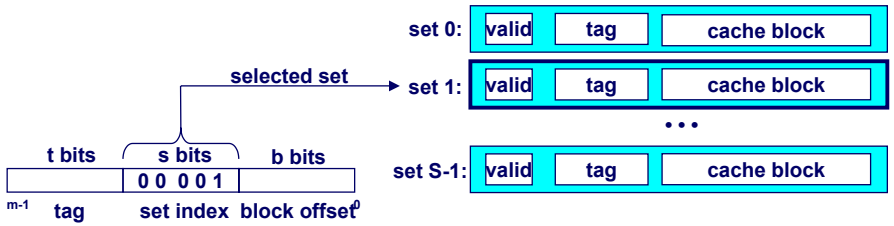
Characterized by exactly one line per set.



Accessing Direct-Mapped Caches

Set selection

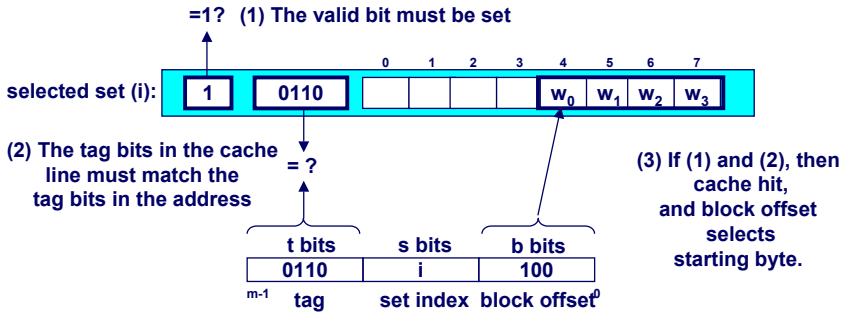
- Use the set index bits to determine the set of interest.



Accessing Direct-Mapped Caches

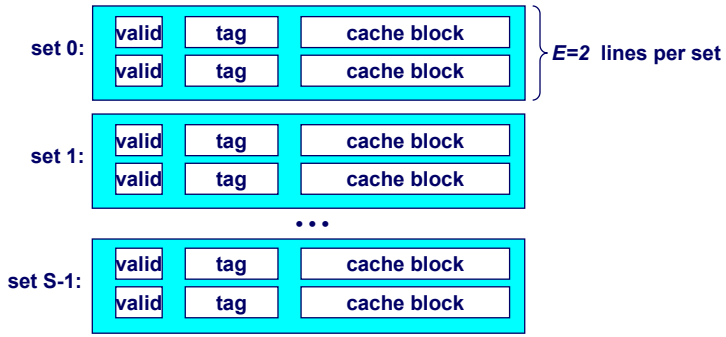
Line matching and word selection

- **Line matching**: Find a valid line in the selected set with a matching tag
- **Word selection**: Then extract the word



Set Associative Caches

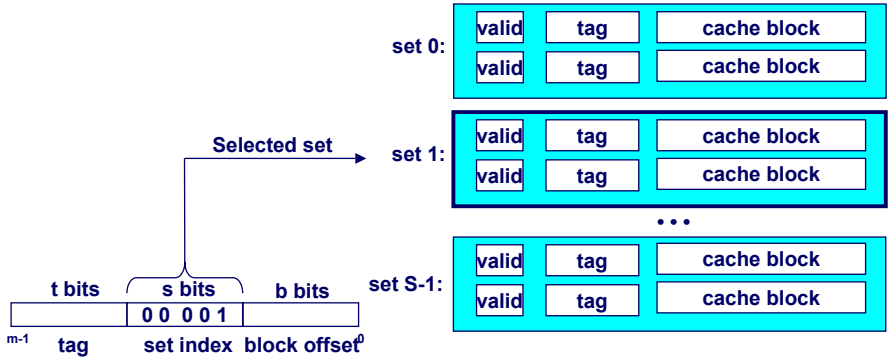
Characterized by more than one line per set



Accessing Set Associative Caches

Set selection

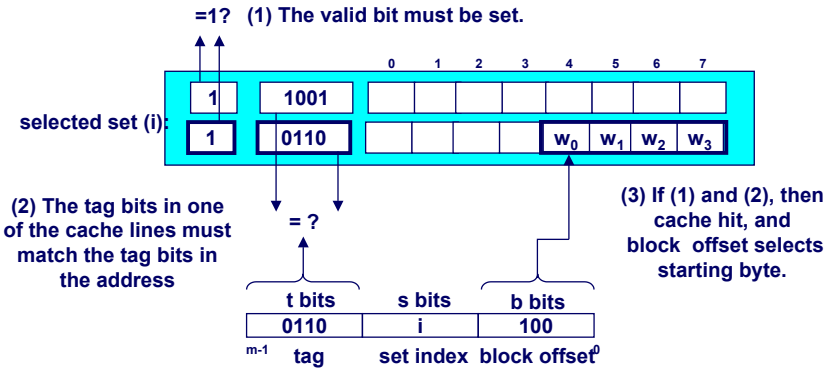
- identical to direct-mapped cache



Accessing Set Associative Caches

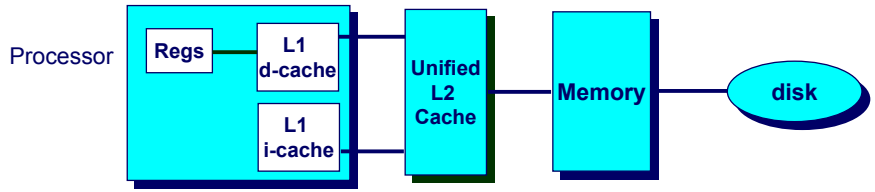
Line matching and word selection

- must compare the tag in each valid line in the selected set.



Multi-Level Caches

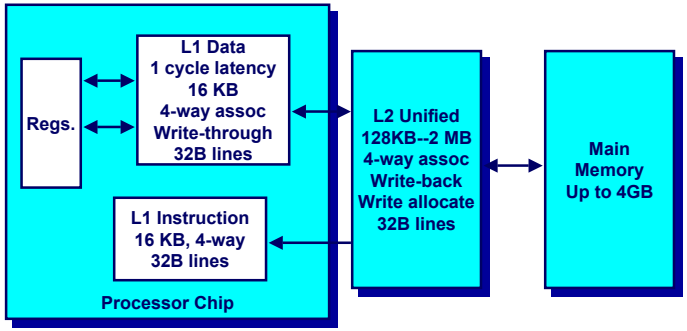
Options: separate data and instruction caches, or a unified cache



size:	200 B	8-64 KB	1-4MB SRAM	128 MB DRAM	30 GB
speed:	3 ns	3 ns	6 ns	60 ns	8 ms
$/Mbyte:			$100/MB	$1.50/MB	$0.05/MB
line size:	8 B	32 B	32 B	8 KB	

larger, slower, cheaper

Intel Pentium Cache Hierarchy



Cache Performance Metrics

Miss Rate

- Fraction of memory references not found in cache (misses/references)
- Typical numbers:
  - 3-10% for L1
  - can be quite small (e.g., < 1%) for L2, depending on size, etc.

Hit Time

- Time to deliver a line in the cache to the processor (includes time to determine whether the line is in the cache)
- Typical numbers:
  - 1 clock cycle for L1
  - 3-8 clock cycles for L2

Miss Penalty

- Additional time required because of a miss
  - Typically 25-100 cycles for main memory

Writing Cache Friendly Code

Repeated references to variables are good (temporal locality)

Stride-1 reference patterns are good (spatial locality)

Examples:

- cold cache, 4-byte words, 4-word cache blocks

```
int sumarrayrows(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;
}
```

Miss rate = 1/4 = 25%

```
int sumarraycols(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;
}
```

Miss rate = 100%

Concluding Observations

Programmer can optimize for cache performance

- How data structures are organized
- How data are accessed
  - Nested loop structure
  - Blocking is a general technique

All systems favor "cache friendly code"

- Getting absolute optimum performance is very platform specific
  - Cache sizes, line sizes, associativities, etc.
- Can get most of the advantage with generic code
  - Keep working set reasonably small (temporal locality)
  - Use small strides (spatial locality)