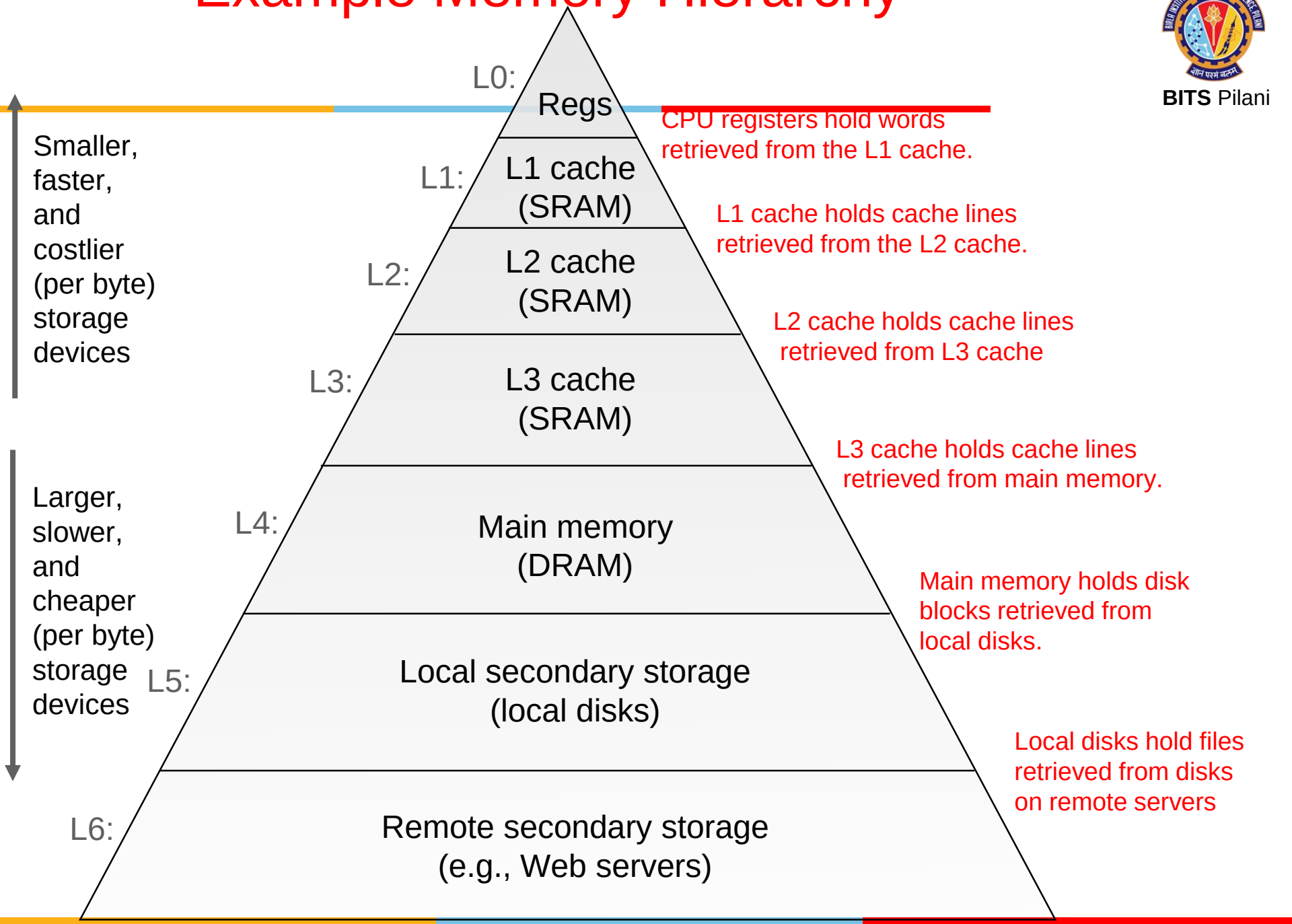
Memory Hierarchy

BITS Pilani
Pilani Campus

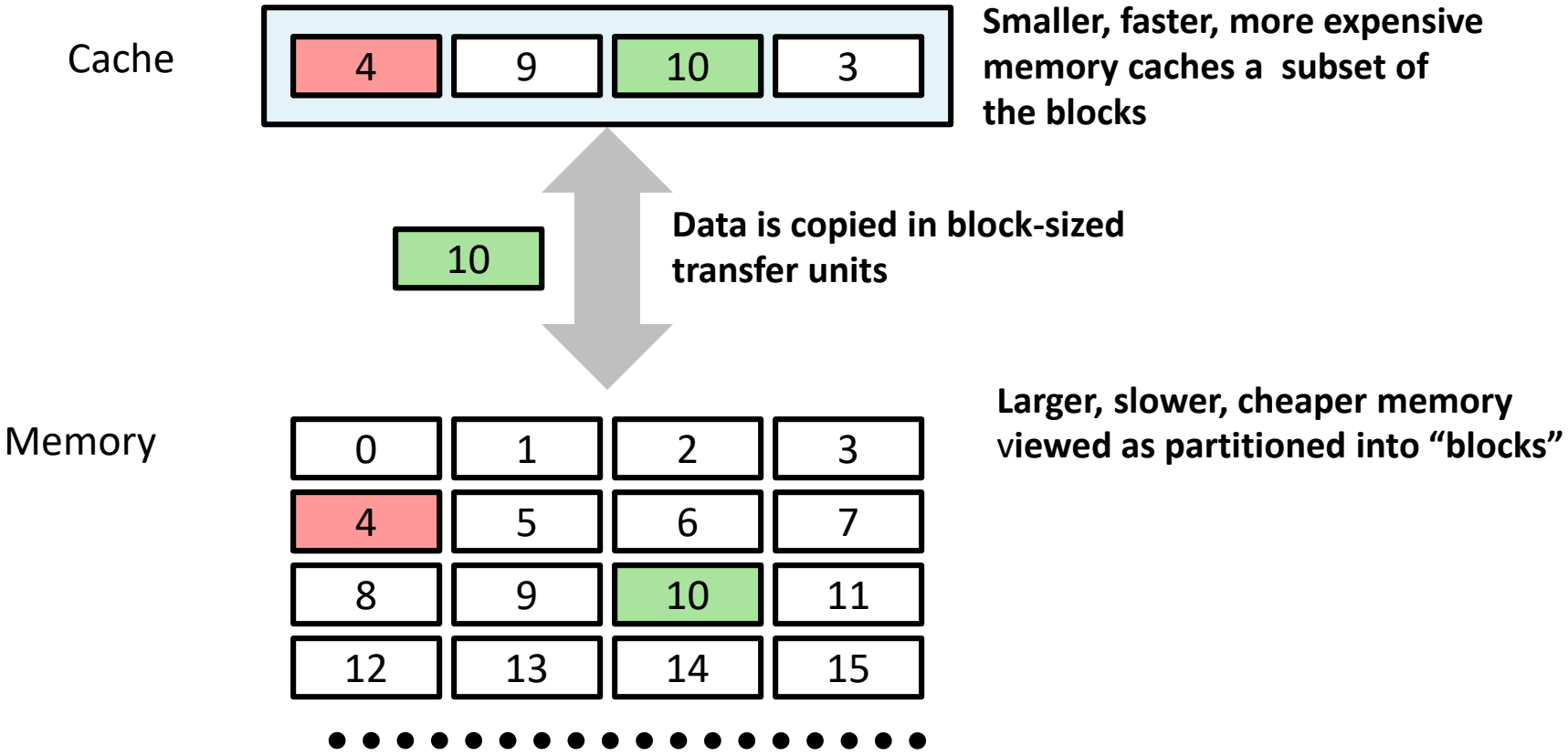
K Hari Babu
Department of Computer Science & Information Systems

Cache Memory Organization

Example Memory Hierarchy



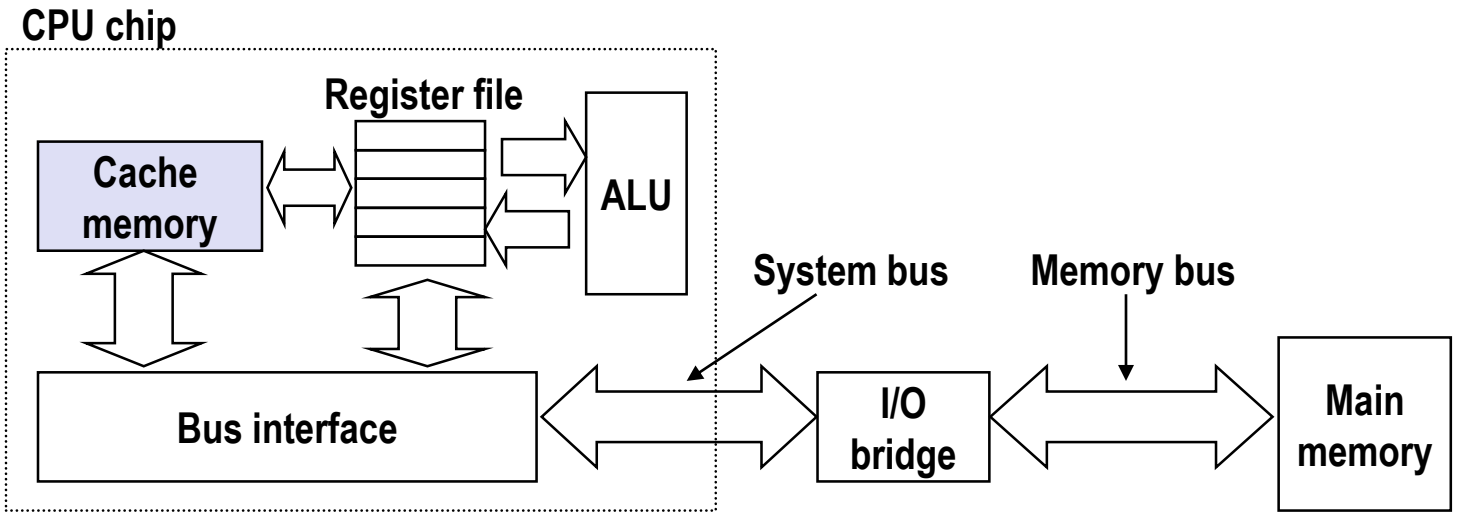
General Cache Concepts



Cache Memories

BITS Pilani

- 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 cache
- Typical system structure:



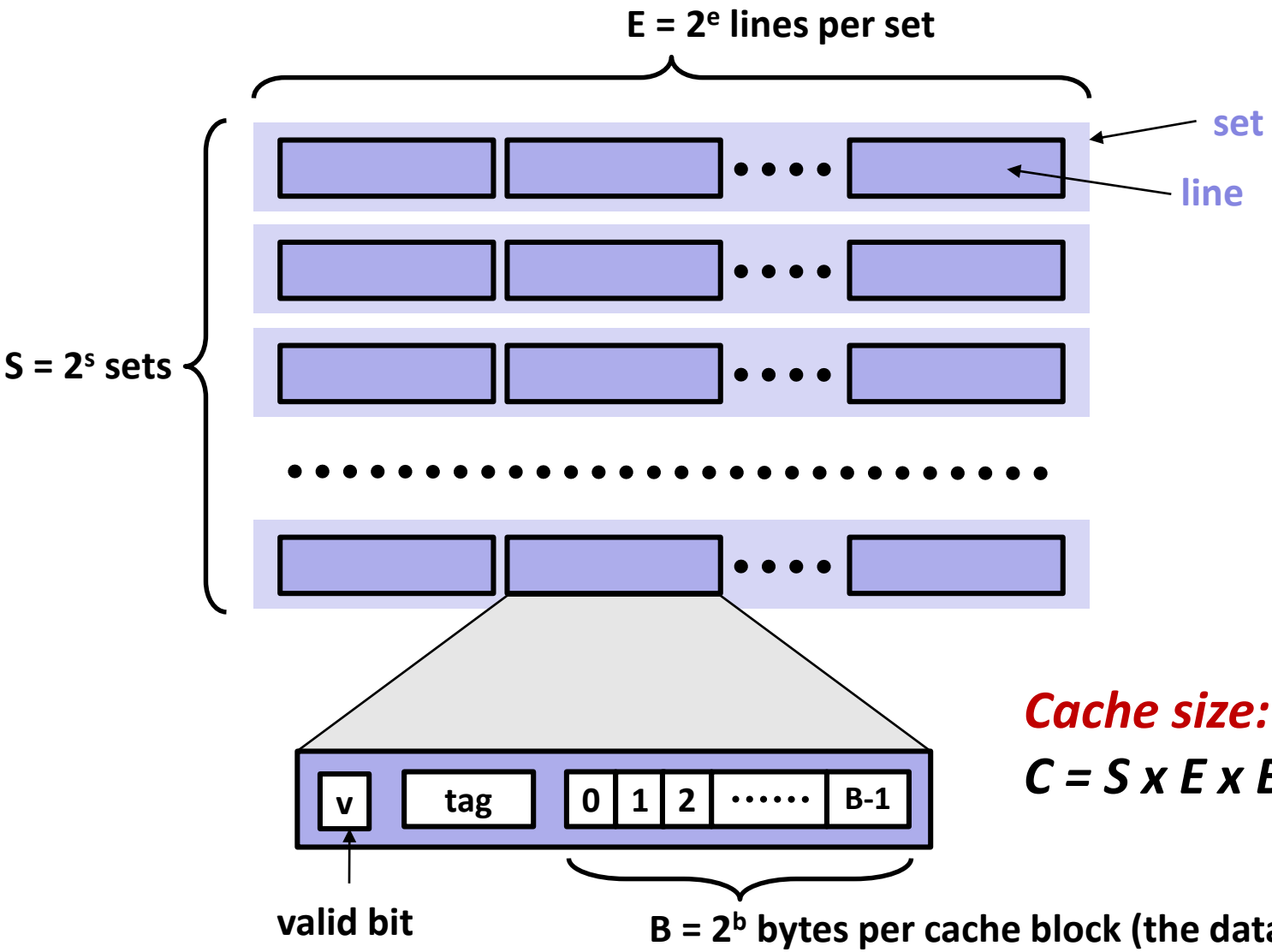
Generic Cache Memory Organization

- A computer system having memory address with m bits forms $M = 2^m$ unique addresses
- A cache for such a machine is organized as an array of $S = 2^s$ cache sets
- Each set consists of E cache lines
- Each line consists of a data block of $B = 2^b$ bytes, a valid bit that indicates whether or not the line contains meaningful information
  - $t = m - (b + s)$ tag bits
- A cache can be characterized by the tuple (S, E, B, m)
  - The size (or capacity) of a cache, C, is stated in terms of the aggregate size of all the blocks. The tag bits and valid bit are not included. Thus, $C = S \times E \times B$.

Cache Placement Policies

- The placement policy decides where in the cache a copy of a particular entry of main memory will go
  - If the placement policy is free to choose any entry in the cache to hold the copy, the cache is called fully associative
  - At the other extreme, if each entry in main memory can go in just one place in the cache, the cache is direct mapped
  - Many caches implement a compromise in which each entry in main memory can go to any one of N places in the cache, and are described as N-way set associative

General Cache Organization (S, E, B)

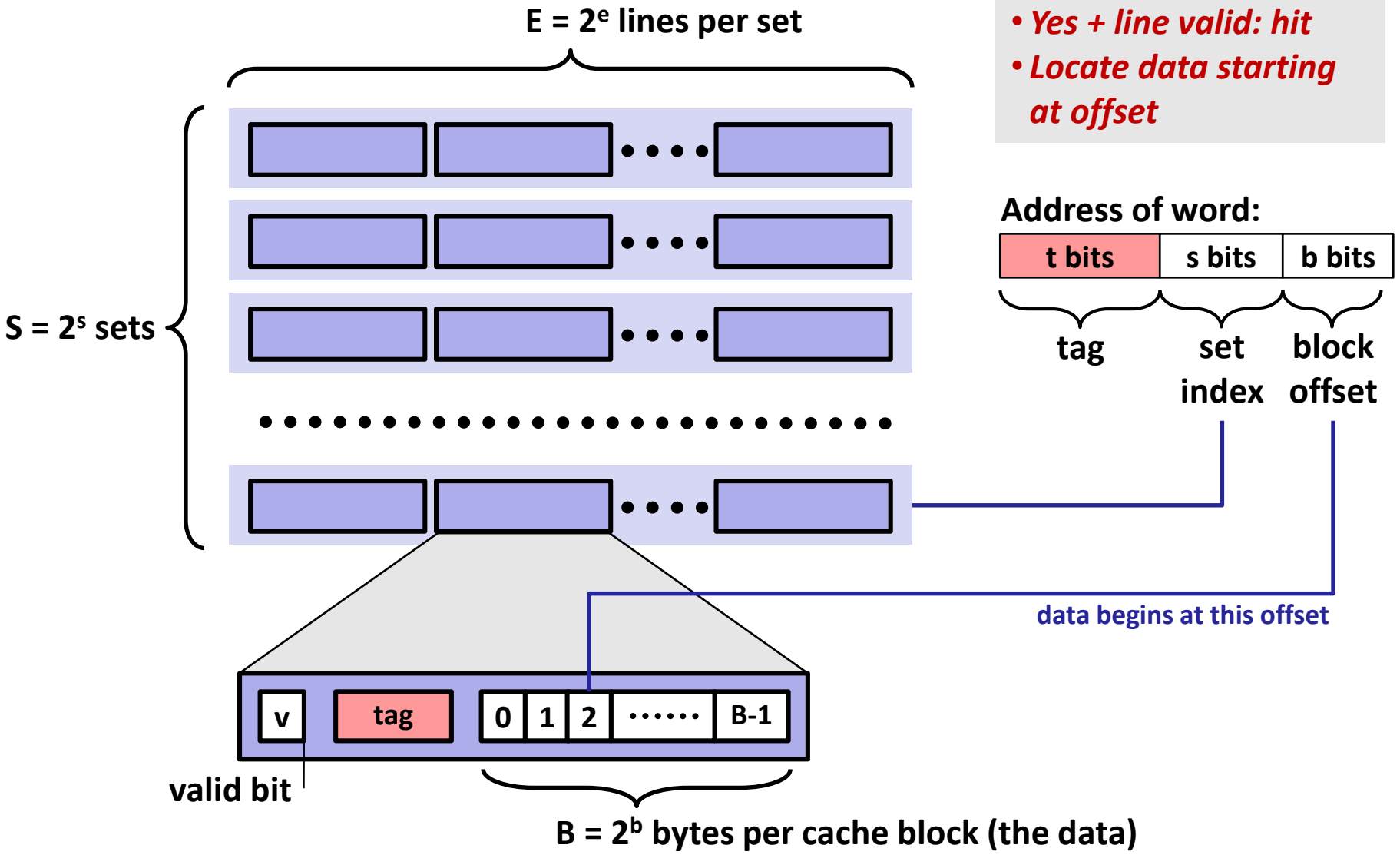


Cache size:

C = S x E x B data bytes

Cache Read

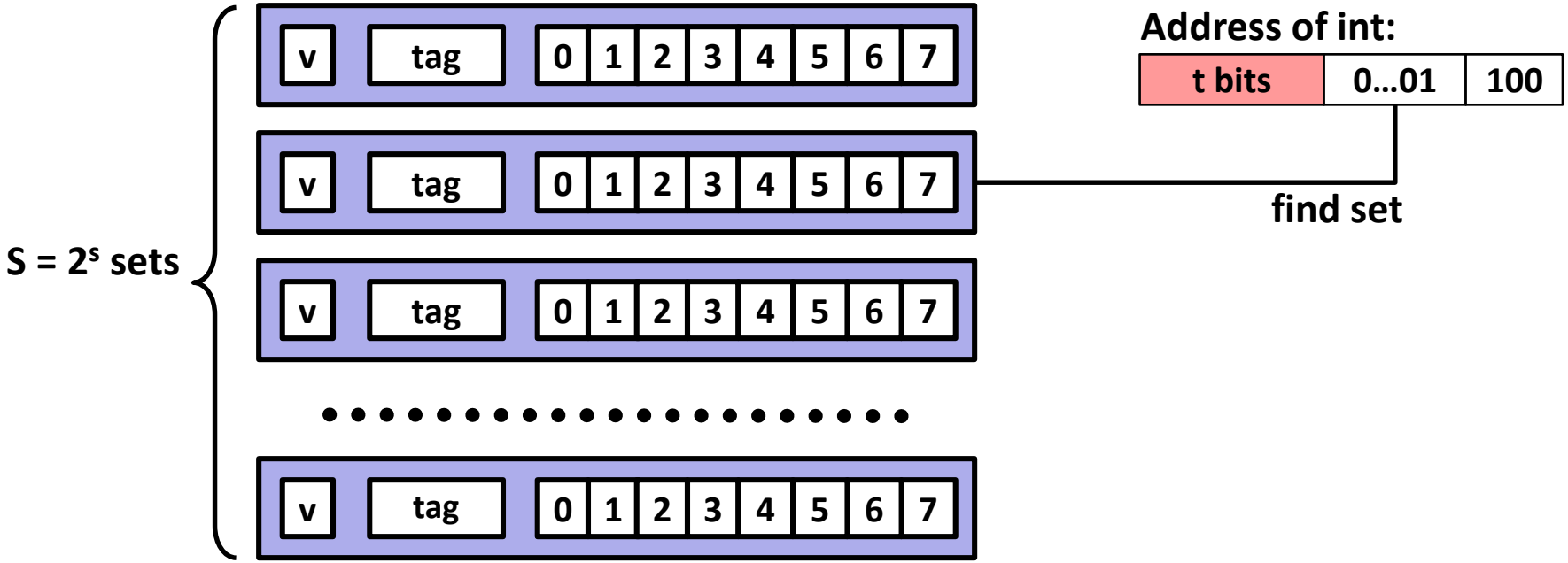
- *Locate set*
- *Check if any line in set has matching tag*
- *Yes + line valid: hit*
- *Locate data starting at offset*



Example: Direct Mapped Cache (E = 1)

BITS Pilani

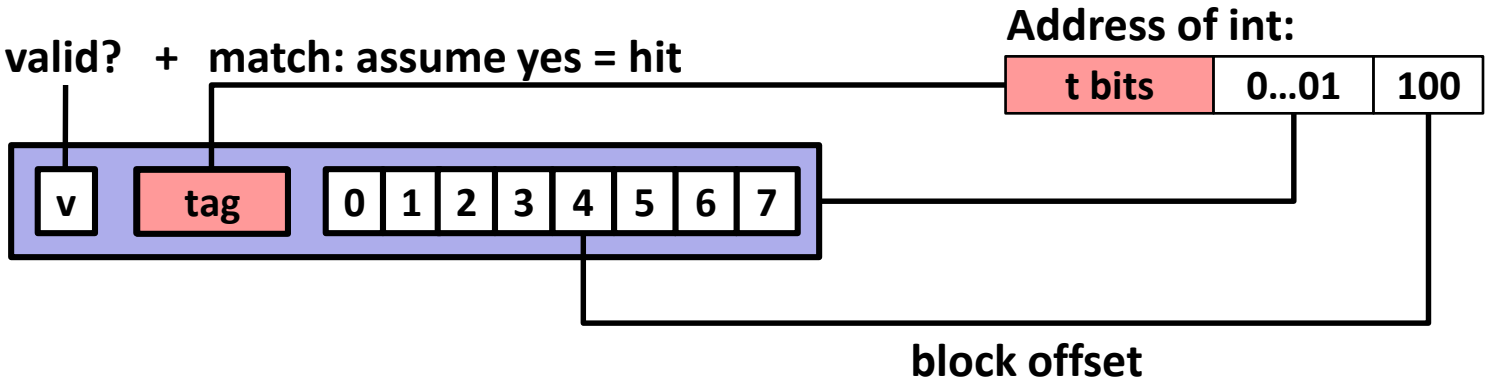
- Direct mapped: One line per set
- Assume: cache block size 8 bytes



Example: Direct Mapped Cache (E = 1)

BITS Pilani

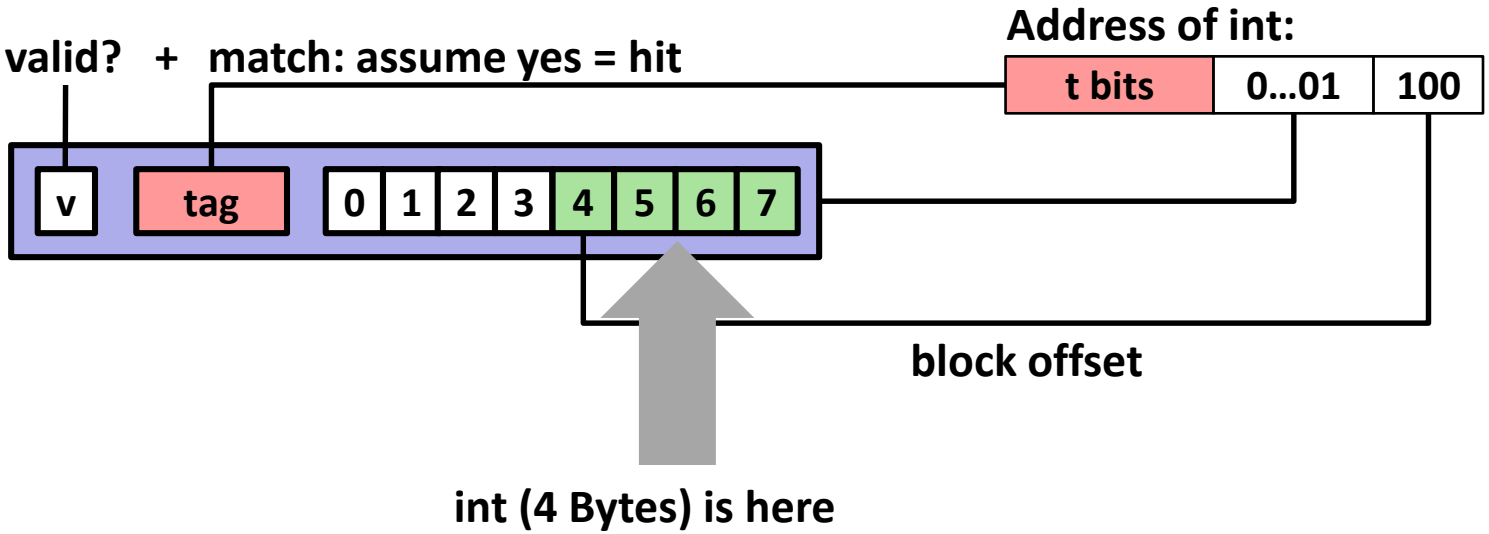
- Direct mapped: One line per set
- Assume: cache block size 8 bytes



Example: Direct Mapped Cache (E = 1)

BITS Pilani

- Direct mapped: One line per set
- Assume: cache block size 8 bytes



- If tag doesn't match: old line is evicted and replaced

Direct Mapped Cache Example

- This function has good spatial locality with respect to x and y, and so we might expect it to enjoy a good number of cache hits. Unfortunately, this is not always true
  - Suppose that floats are 4 bytes, that x is loaded into the 32 bytes of contiguous memory starting at address 0, and that y starts immediately after x at address 32.
  - For simplicity, suppose that a block is 16 bytes (big enough to hold four floats) and that the cache consists of two sets, for a total cache size of 32 bytes

```
1 float dotprod(float x[8], float y[8])
2 {
3 float sum = 0.0;
4 int i;
5
6 for (i = 0; i < 8; i++)
7 sum += x[i] * y[i];
8 return sum;
9 }
```

Direct Mapped Cache Example – Conflict Misses

BITS Pilani

- This function has good spatial locality with respect to x and y, and so we might expect it to enjoy a good number of cache hits. Unfortunately, this is not always true
  - Each x[i] and y[i] will map to the identical set in the cache

Element	Address	Set index	Element	Address	Set index
x[0]	0	0	y[0]	32	0
x[1]	4	0	y[1]	36	0
x[2]	8	0	y[2]	40	0
x[3]	12	0	y[3]	44	0
x[4]	16	1	y[4]	48	1
x[5]	20	1	y[5]	52	1
x[6]	24	1	y[6]	56	1
x[7]	28	1	y[7]	60	1

```
1 float dotprod(float x[8], float y[8])
2 {
3 float sum = 0.0;
4 int i;
5
6 for (i = 0; i < 8; i++)
7 sum += x[i] * y[i];
8 return sum;
9 }
```

  - Each subsequent reference to x and y will result in a conflict miss as we thrash back and forth between blocks of x and y.

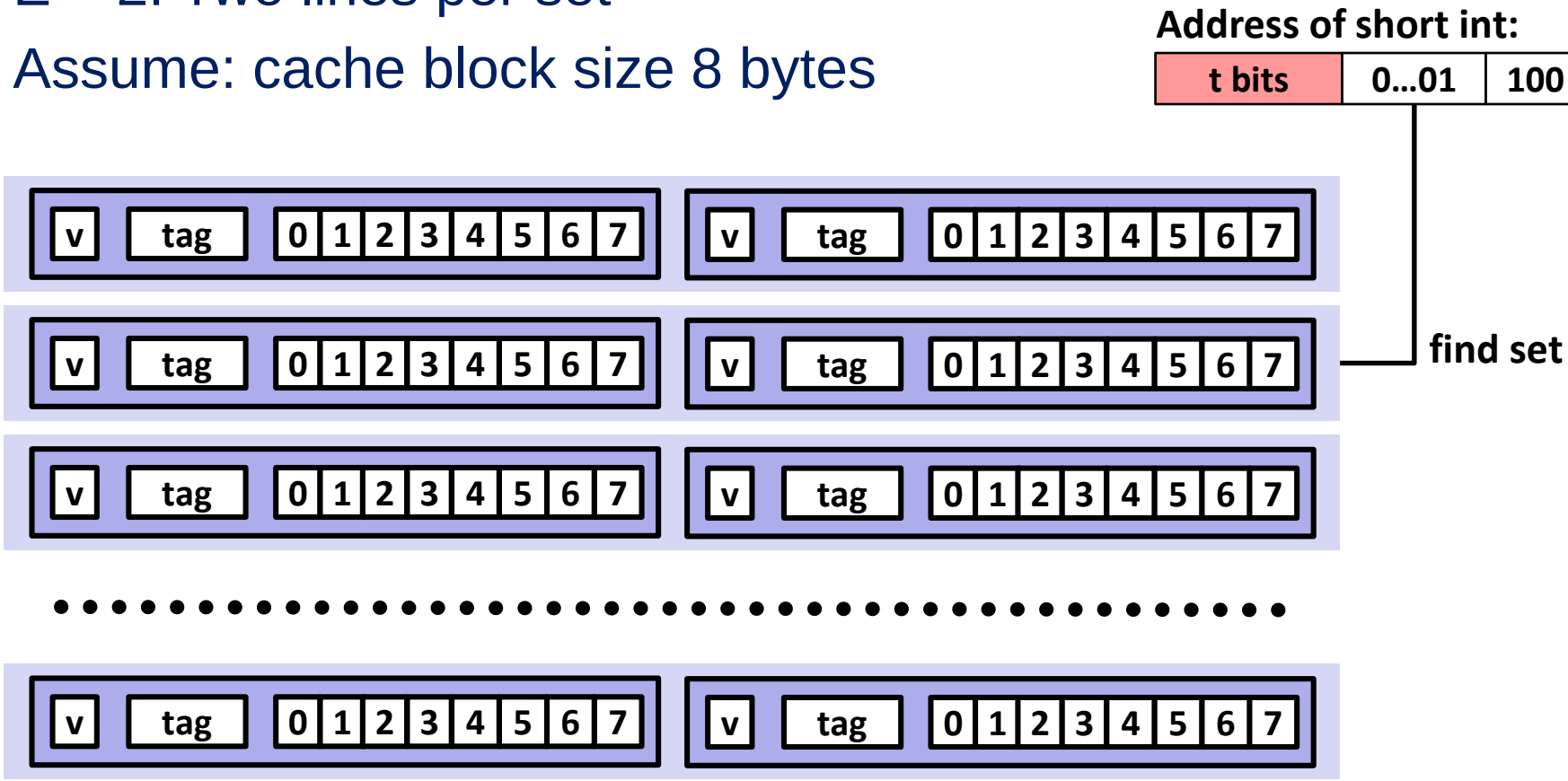
Set Associative Caches

- The problem with conflict misses in direct-mapped caches stems from the constraint that each set has exactly one line (or in our terminology, E = 1)
  - A set associative cache relaxes this constraint so that each set holds more than one cache line. A cache with 1 < E < C/B is often called an E-way set associative cache
  - A fully associative cache consists of a single set (i.e., E = C/B) that contains all of the cache lines
- A block is mapped to a set, and it can be placed in any of the lines inside the set along with a tag
  - If no line is free, line is replaced.

E-way Set Associative Cache (Here: E = 2)

BITS Pilani

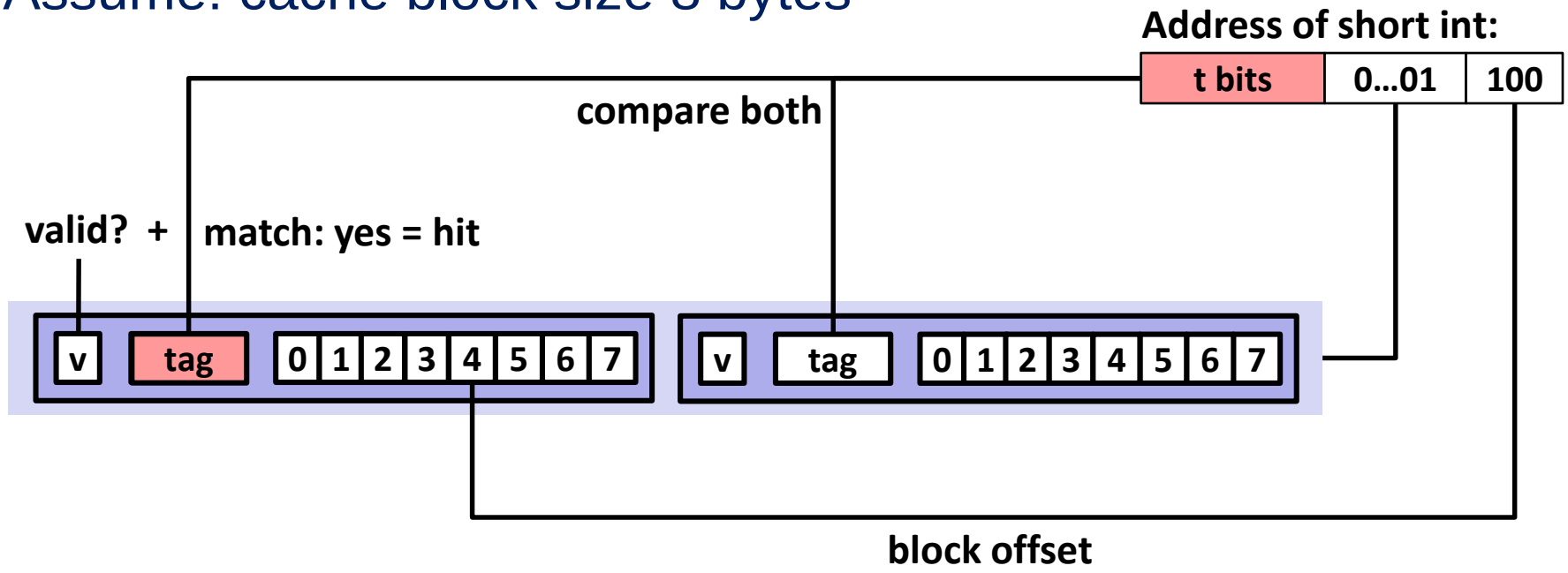
- E = 2: Two lines per set
- Assume: cache block size 8 bytes



E-way Set Associative Cache (Here: E = 2)

BITS Pilani

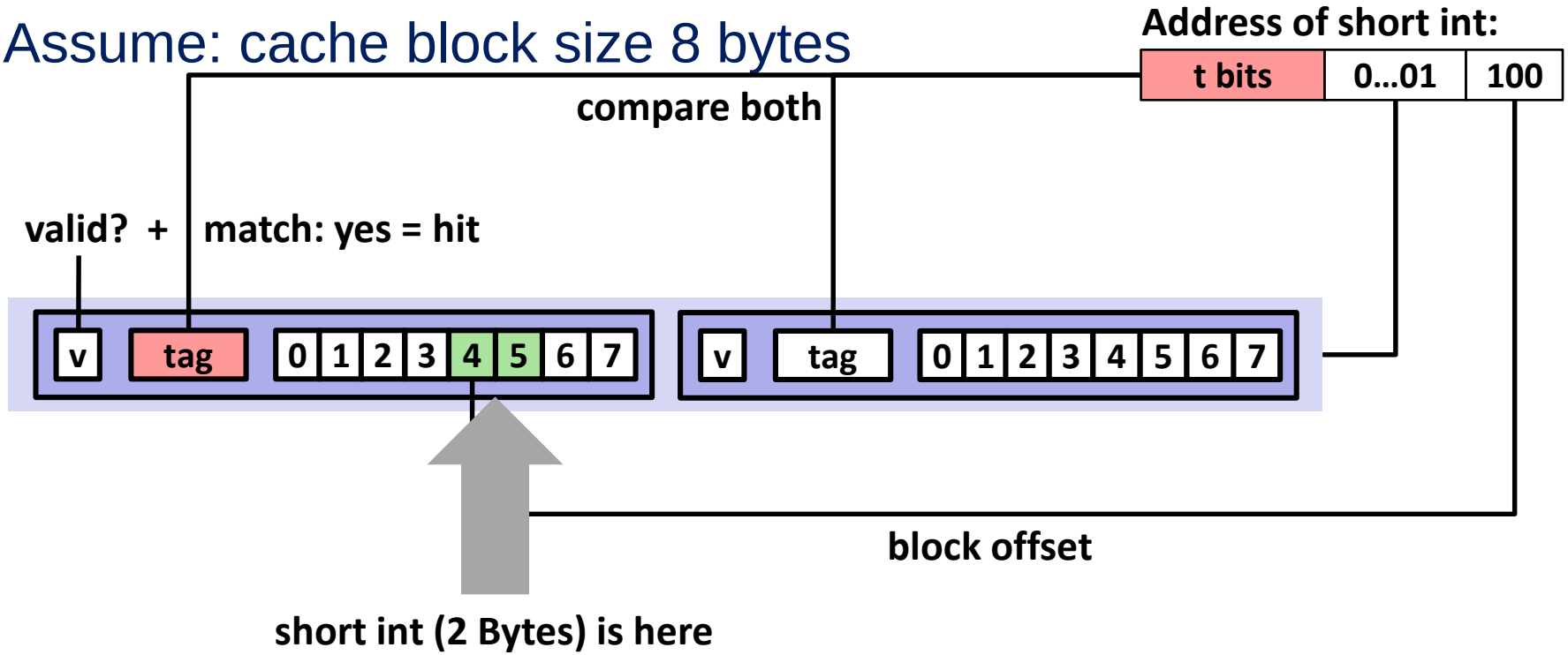
- E = 2: Two lines per set
- Assume: cache block size 8 bytes



E-way Set Associative Cache (Here: E = 2)

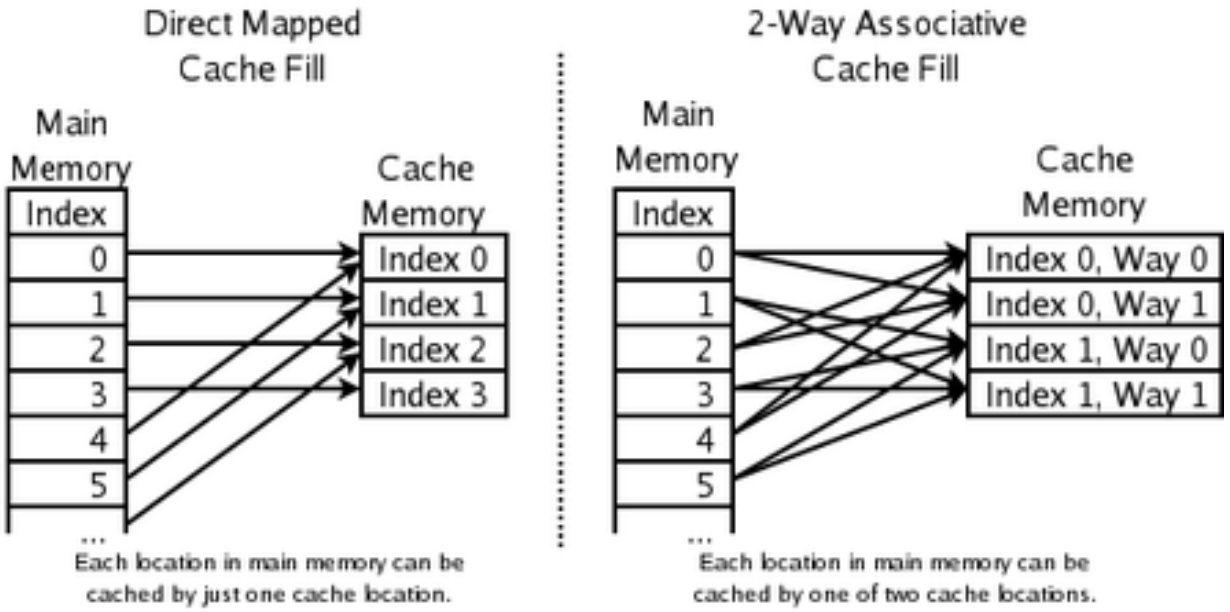
BITS Pilani

- E = 2: Two lines per set
- Assume: cache block size 8 bytes



- If No match:
  - One line in set is selected for eviction and replacement
  - Replacement policies: random, least recently used (LRU), …

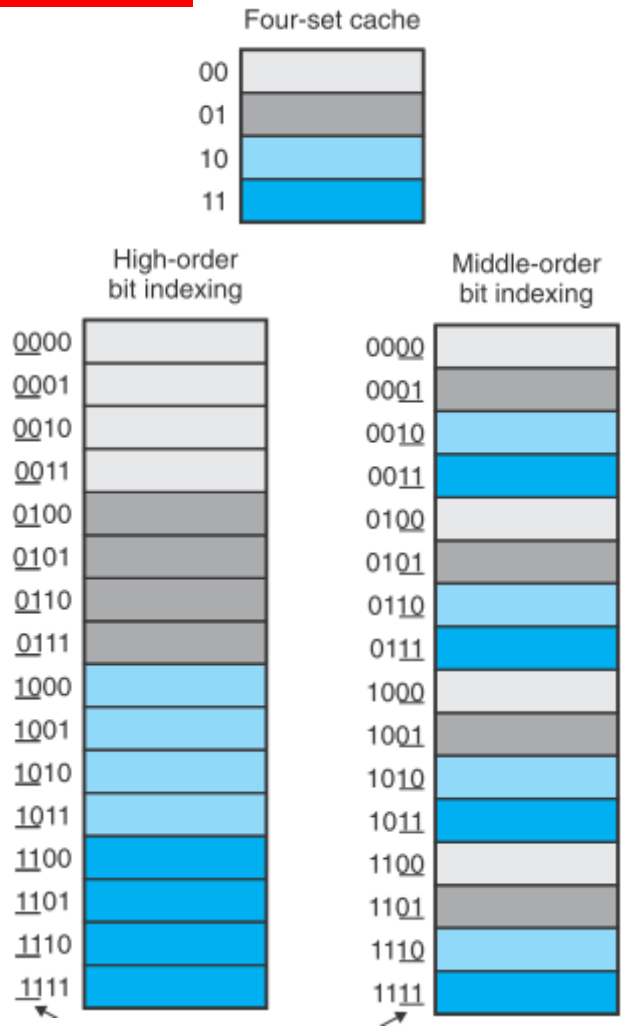
Direct Mapped Vs 2-Way Associative



Why index with the middle bits?

BITS Pilani

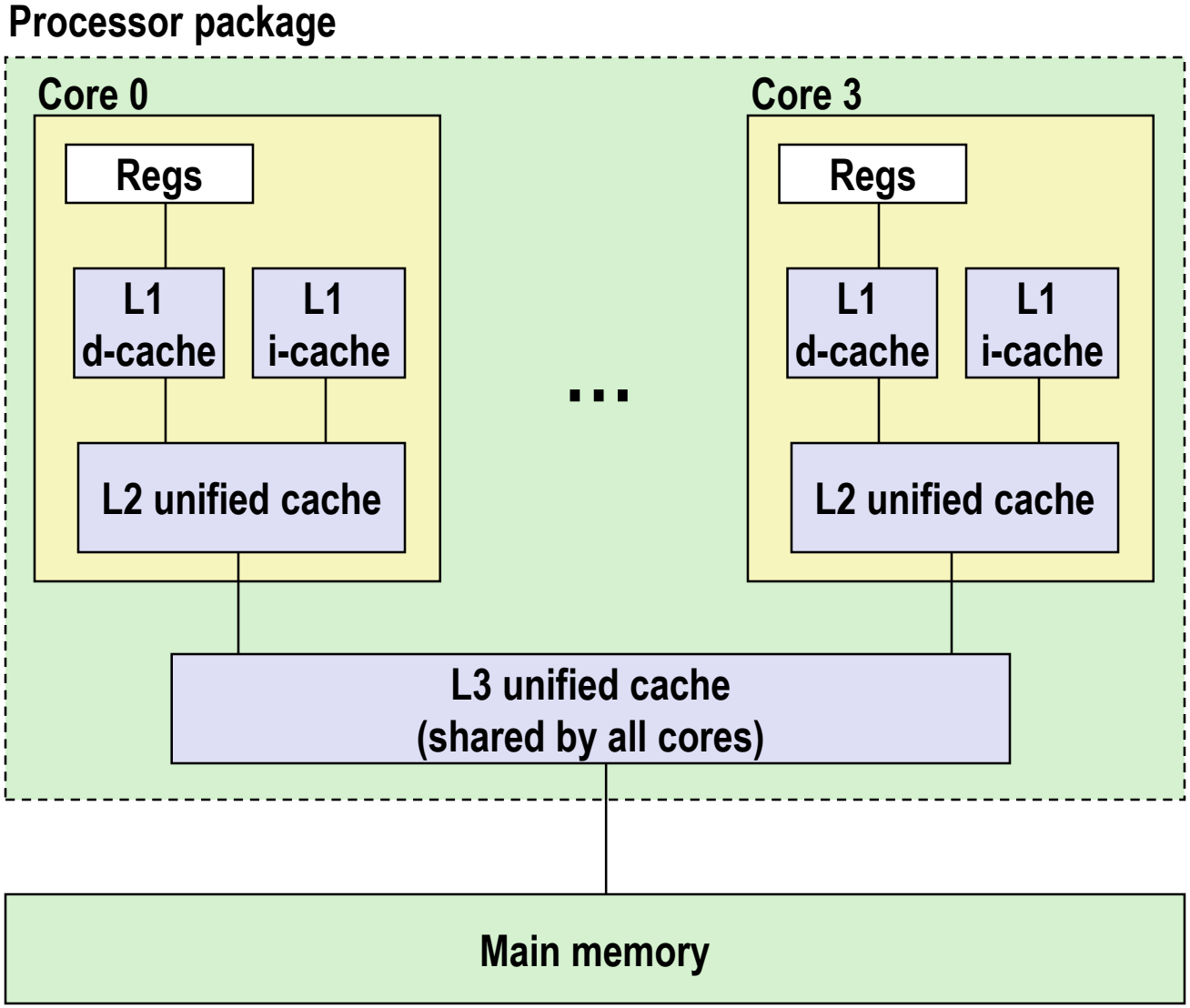
- Why caches use the middle bits for the set index instead of the high-order bits?
  - If the high-order bits are used as an index, then some contiguous memory blocks will map to the same cache set
  - If a program scans the elements of an array sequentially, then the cache can only hold a block-size chunk of the array at any point in time. This is an inefficient use of the cache
  - Contrast this with middle-bit indexing, where adjacent blocks always map to different cache sets
    - In this case, the cache can hold an entire C-size chunk of the array, where C is the cache size.



What about writes?

- Multiple copies of data exist:
  - L1, L2, L3, Main Memory, Disk
- What to do on a write-hit?
  - Write-through (write immediately to memory)
  - Write-back (defer write to memory until replacement of line)
    - Need a dirty bit (line different from memory or not)
- What to do on a write-miss?
  - Write-allocate (load into cache, update line in cache)
    - Good if more writes to the location follow
  - No-write-allocate (writes straight to memory, does not load into cache)
- Typical
  - Write-through + No-write-allocate
  - **Write-back + Write-allocate**

Intel Core i7 Cache Hierarchy



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.

Cache Performance Metrics

- Miss Rate
  - Fraction of memory references not found in cache (misses / accesses) = 1 – hit rate
  - Typical numbers (in percentages):
    - 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:
    - 4 clock cycle for L1
    - 10 clock cycles for L2
- Miss Penalty
  - Additional time required because of a miss
    - typically 50-200 cycles for main memory (Trend: increasing!)

References

- Section 6.4 from Computer Systems: A Programmer's Perspective, 3rd Edition by Randal E. Bryant and David R. O'Hallaron, Pearson, 2016

Q&A

innovate achieve lead

BITS Pilani

Pilani Campus

Thank You