CMPE 110: Computer Architecture

Week 6
Cache

Jishen Zhao (http://users.soe.ucsc.edu/~jzhao/)

[Adapted in part from Jose Renau, Mary Jane Irwin, Joe Devietti, Onur Mutlu, and others]

Reminder

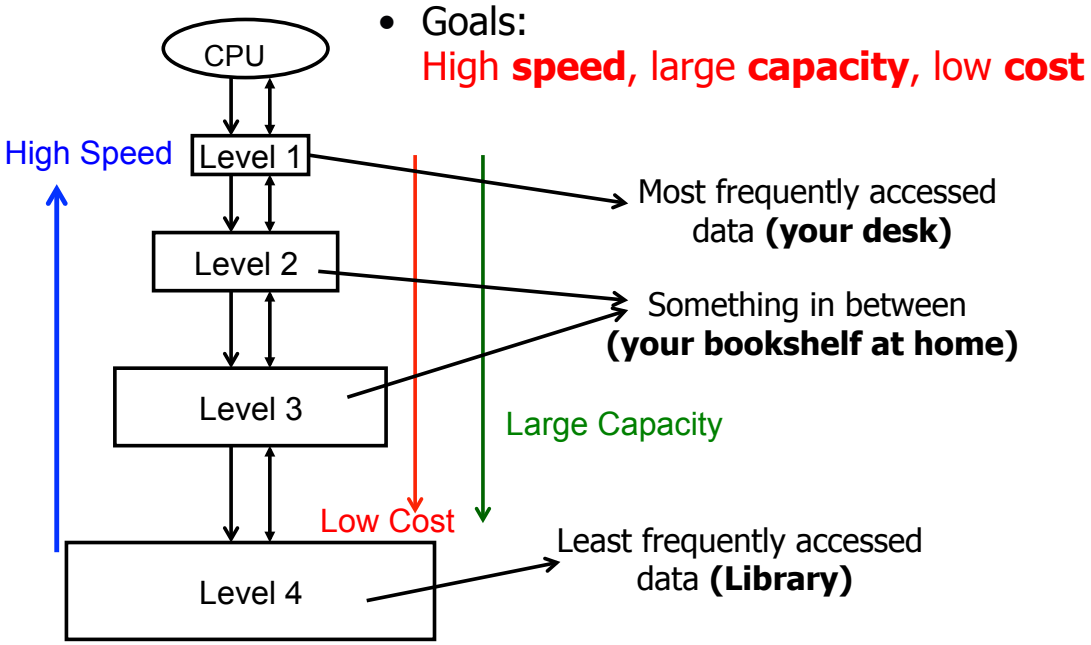
- Quiz 2 is due today midnight

Review: Temporal and Spatial Locality

- Which memory accesses demonstrate spatial locality?
- Which memory accesses demonstrate temporal locality?

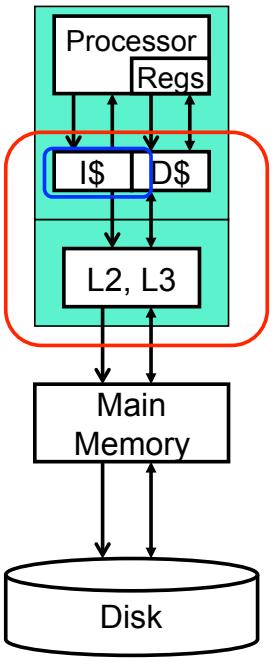
```
int sum = 0;
int x[1000];

for(int c = 0; c < 1000; c++) {
  sum += c;  // Temporal locality: A program tends to access
                the same memory location many times and all
                within a small window of time

  x[c] = 0;  // Spatial locality: A program tends to reference a
                cluster of memory locations at a time
}
```


Review: basic idea of memory hierarchy



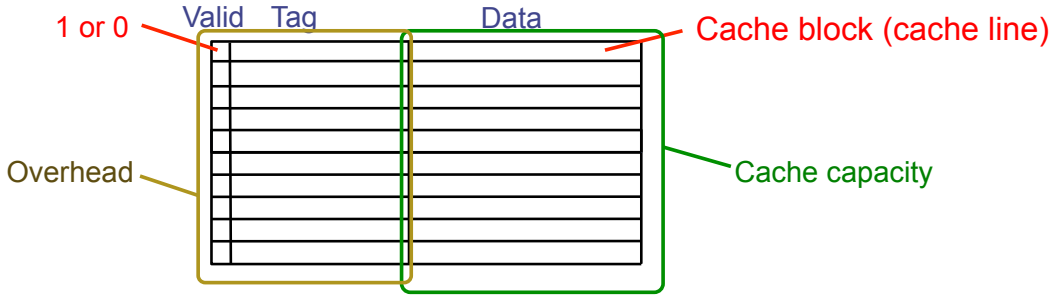
- Goals:
  High **speed**, large **capacity**, low **cost**

Review: Cache basics



- I$ (instruction cache): read only
- D$ (data cache): read/write
- L2, L3 are lower-level caches for both instruction and data
- Inclusive vs. exclusive vs. non-inclusive cache

Review: Cache basis



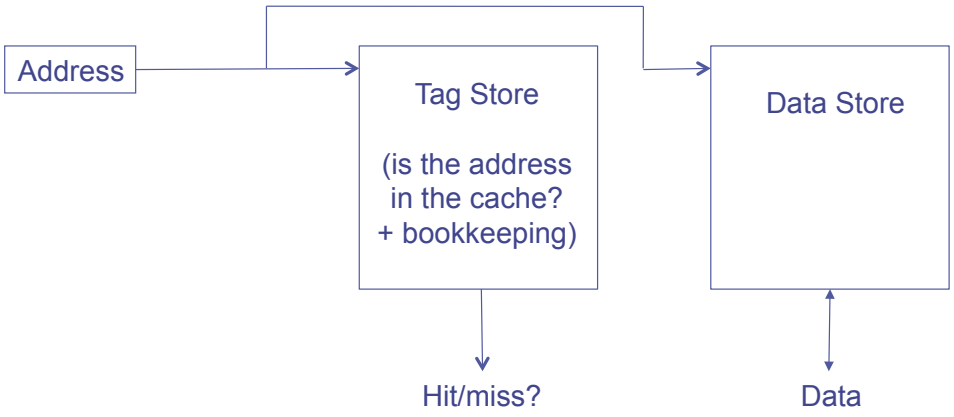
- When data referenced
  - HIT: If in cache, use cached data instead of accessing memory
  - MISS: If not in cache, bring block into cache (invalid → miss)
    - Go to the next level of cache to bring this data up
    - Have to kick something else out to do it, if it is full

Today: Cache (cont.)

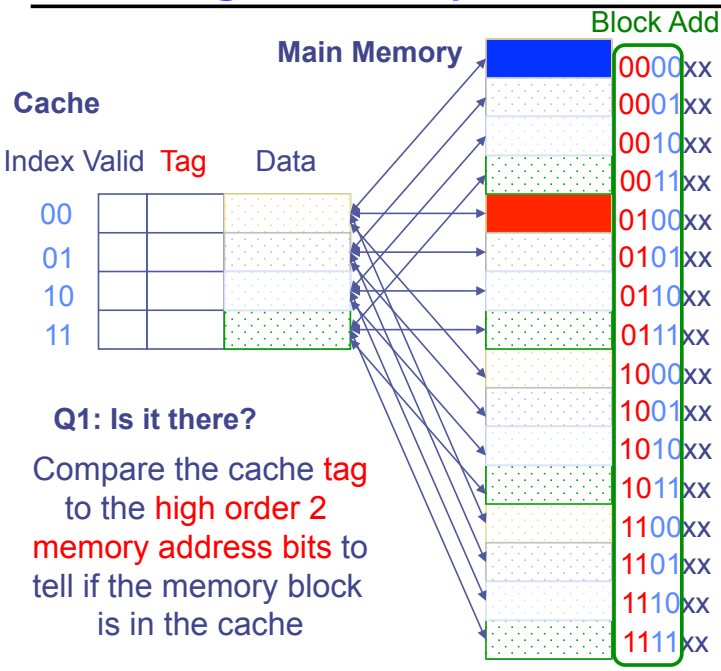
- How to access cache with a memory address?
- How to calculate tag overhead?
- Cache and the pipeline

How is cache accessed?

- Two questions to answer (in hardware):
  - Q1: How do we know if a data item is in the cache?
  - Q2: If it is, how do we find it?



Caching: A Simple First Example



32-bit (4-byte) data (1 word)
6-bit address

- Need 2 bits (xx) to identify the 4 bytes of data
- The rest 4 bits used as "block address"

Q1: Is it there?

Compare the cache tag to the high order 2 memory address bits to tell if the memory block is in the cache

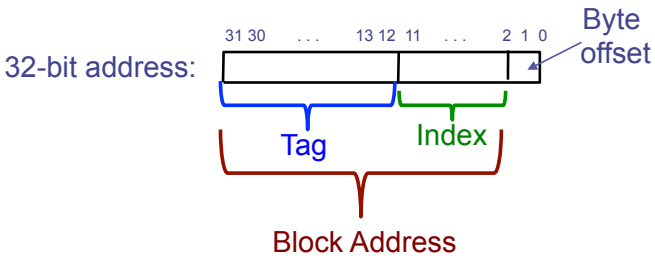
Q2: How do we find it?

Use next 2 low order memory address bits – the index – to determine which cache block (i.e., modulo the number of blocks in the cache)

Index = (block address) modulo (# of blocks in the cache)

Three regions in memory address

- Address is a unique pointer to a data block in main memory



How is cache accessed?

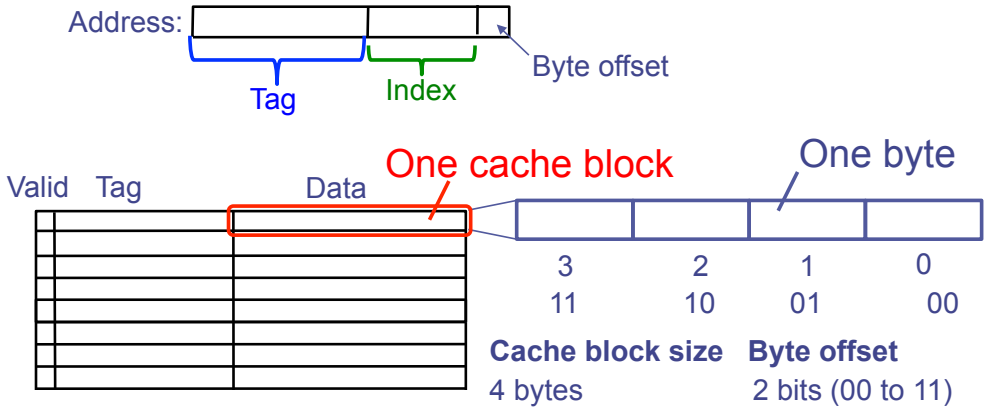
- **Input:** the address of data to be accessed
- **Output:**
  - Hit or miss?
  - If hit → we get the data from cache
  - If miss → we don't get the data from cache, have to access the next level in the memory hierarchy

Cache access with three steps

- Step 1: Identify "byte offset" and "block address"
- Step 2: Calculate "cache index"
- Step 3: Compare "cache tag"

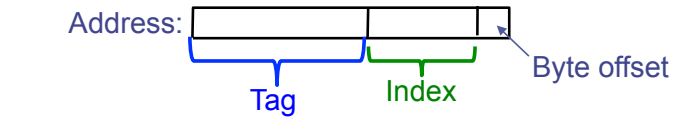
Step 1: Byte offset and block address

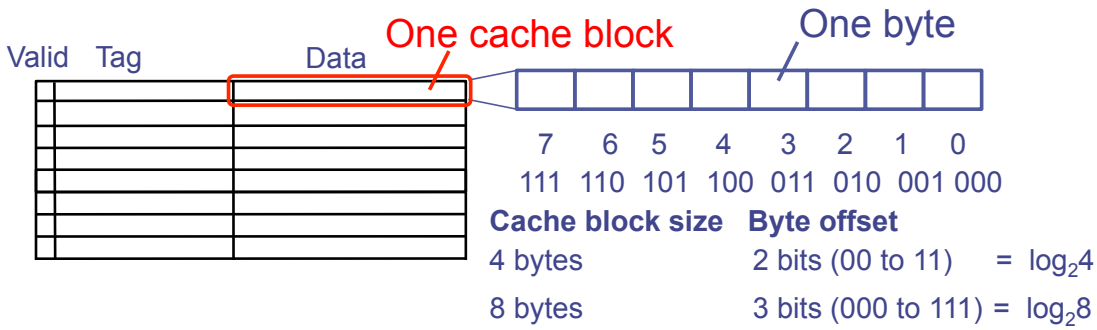
- Byte offset: determined by the size of one cache block
- Index: determined by the total number of cache blocks the the cache
- Tag: the rest of bits in the address



Step 1: Byte offset and block address

- Byte offset = $\log_2$(Bytes in one cache block)
- Index: determined by the total number of cache blocks the the cache
- Tag: the rest of bits in the address

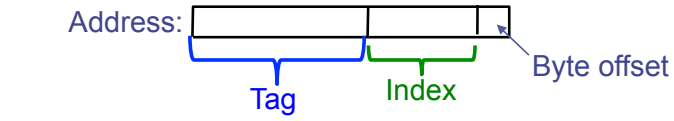


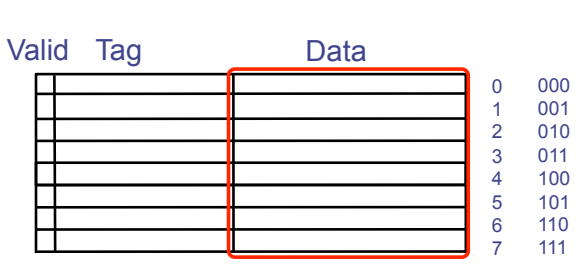


Cache block size	Byte offset
4 bytes	2 bits (00 to 11) = $\log_2 4$
8 bytes	3 bits (000 to 111) = $\log_2 8$

Step 2: Index region

- Byte offset = $\log_2$(Bytes in one cache block)
- Index: determined by the total number of cache blocks the the cache
- Tag: the rest of bits in the address

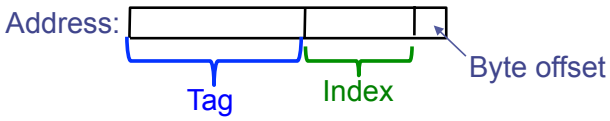


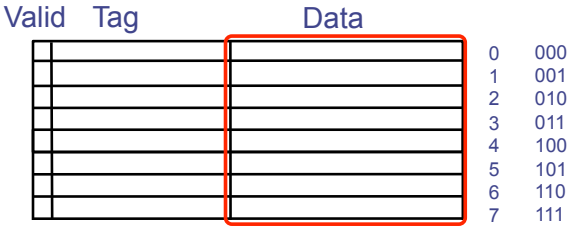


Number of cache blocks	Index
8	3 bits (000 to 111)
16	4 bits (0000 to 1111)
32	5 bits (00000 to 11111)
64	6 bits (000000 to 111111)
.....	

Step 2: Index region

- Byte offset = $\log_2$(Bytes in one cache block)
- Index = $\log_2$(Number of cache blocks in the cache)
- Tag: the rest of bits in the address

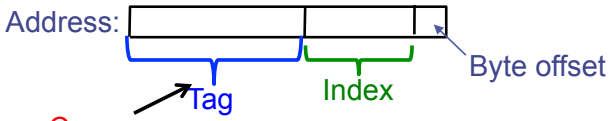


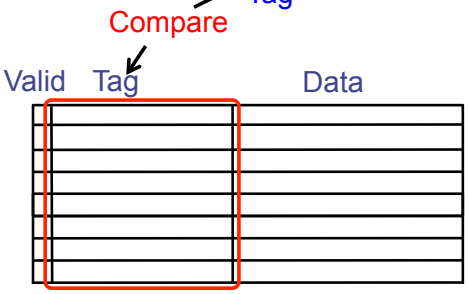


Number of cache blocks	Index
8	3 bits (000 to 111)
16	4 bits (0000 to 1111)
32	5 bits (00000 to 11111)
64	6 bits (000000 to 111111)
.....	

Step 3: tag region

- Byte offset = $\log_2$(Bytes in one cache block)
- Index = $\log_2$(Number of cache blocks in the cache)
- Tag: the rest of bits in the address





Put it all together

- Cache block = 1 word (4 bytes),
- cache capacity = 1K words (4K bytes)