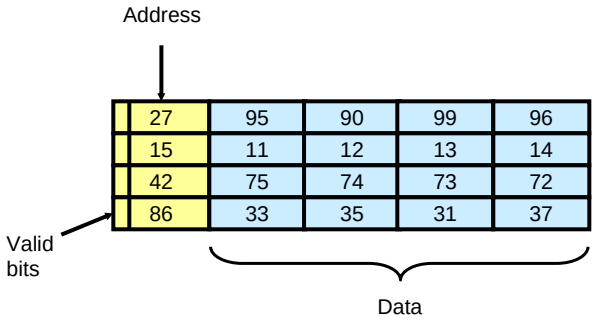
Lecture 16: Improving Cache Performance

- Last Time:
  - Cache introduction
    - Average Memory Access Time (AMAT)
    - Set associativity
- Today
  - Replacement and write policies
  - Cache performance optimizations

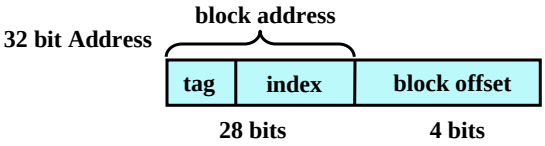
Cache Organization



- **Where does a block get placed?**
- **How do we find it?**
- **Which one do we replace when a new one is brought in?**
- **What happens on a write?**

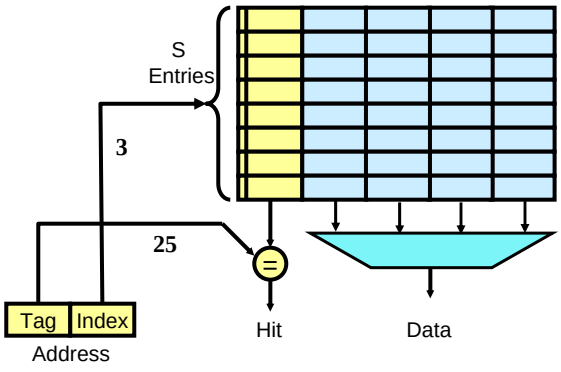
How Do We Find a Block in The Cache?

- Our Example:
  - Main memory address space = 32 bits (= 4GBytes)
  - Block size = 4 words = 16 bytes
  - Cache capacity = 8 blocks = 128 bytes



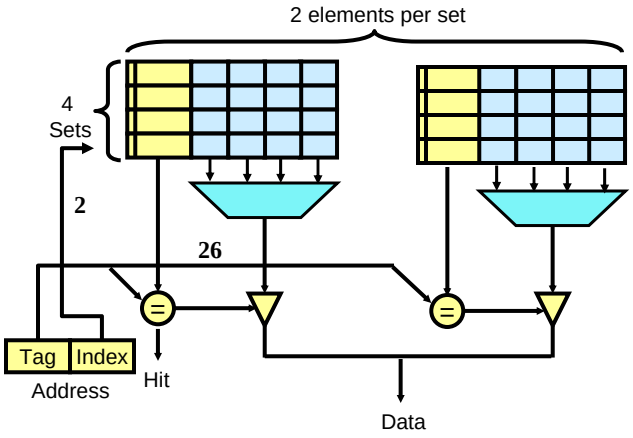
- Valid bit ⇒ is cache block good?
- index ⇒ which set
- tag ⇒ which data/instruction in block
- block offset ⇒ which word in block
- # tag/index bits determine the associativity
- tag/index bits can come from anywhere in block address

Finding a Block: Direct-Mapped



With cache capacity = 8 blocks

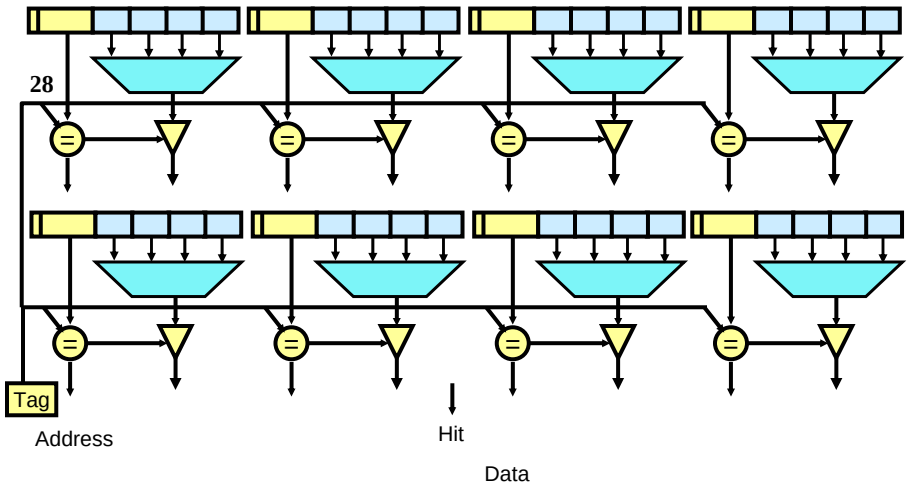
Finding A Block: 2-Way Set-Associative



Set Associative Cache

- S - sets
- A - elements in each set
  - A-way associative
- In the example, S=4, A=2
  - 4-way associative 8-entry cache
- All of main memory is divided into S sets
  - All addresses in set N map to same set of the cache
    - Addr = N mod S
    - A locations available
- Shares costly comparators across sets
- Low address bits select set
  - 2 in example
- High address bits are *tag*, used to associatively search the selected set
- Extreme cases
  - A=1: Direct mapped cache
  - S=1: Fully associative
- A need not be a power of 2

Finding A Block: Fully Associative



Which Block Should Be Replaced on Miss?

- Direct Mapped
  - Choice is easy - only one option
- Associative
  - Randomly select block in set to replace
  - Least-Recently used (LRU)
- Implementing LRU
  - 2-way set-associative
  - >2 way set-associative

What Happens on a Store?

- Need to keep cache consistent with main memory
  - Reads are easy - no modifications
  - Writes are harder - when do we update main memory?
- Write-Through
  - On cache write - always update main memory as well
  - Use a write buffer to stockpile writes to main memory for speed
- Write-Back
  - On cache write - remember that block is modified (dirty bit)
  - Update main memory when dirty block is replaced
  - Sometimes need to flush cache (I/O, multiprocessing)

BUT: What if Store Causes Miss!

- Write-Allocate
  - Bring written block into cache
  - Update word in block
  - Anticipate further use of block
- No-write Allocate
  - Main memory is updated
  - Cache contents unmodified