Outline of Lecture

- **Designing Cache Memory**

Types of Cache Memory

➔ ***Direct mapped*: Each block has only one place to appear in the cache. The mapping is (*block address MOD number of blocks in cache*).**

✣ **It is easy to locate blocks in the cache (only one possibility)**

- **Certain blocks cannot be simultaneously present in the cache (they can only have the same cache location)**

➔ ***Fully associative*: A block can be placed anywhere in the cache.**

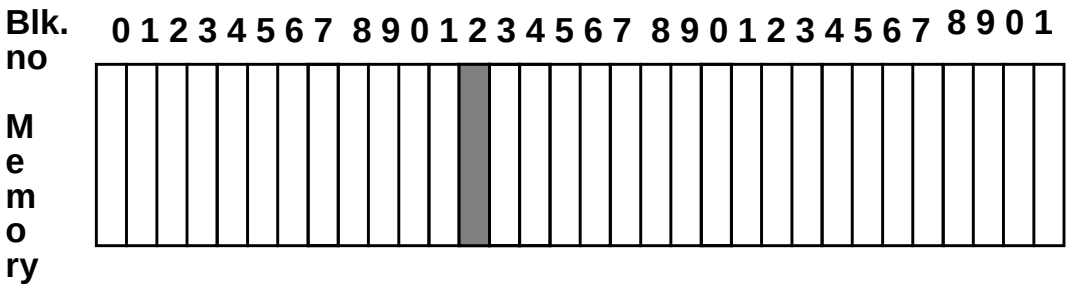
✣ **No restriction on the placement of blocks. any combination of blocks can be simultaneously present on the cache.**

- **Quite costly (hardware and time) to search for a block.**

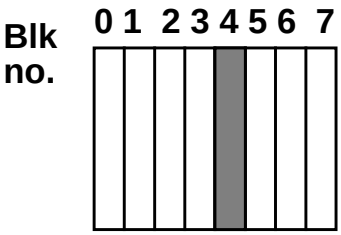
➔ ***Set associative*: Each block has only a certain number of places to appear in the cache. The mapping is (*block address MOD number of sets in cache*).**

✣ **A good compromise between direct mapped and fully associative caches.**

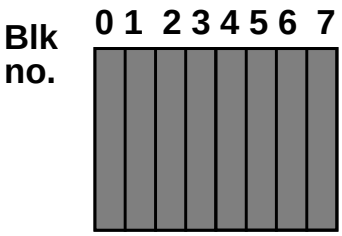
Comparison



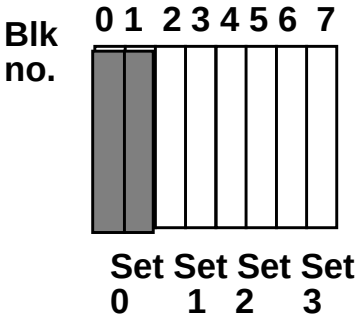
Direct mapped
(Blk 12 can go only into Blk. 4 12 mod 8)



Fully associative
(blk. 12 can go anywhere)



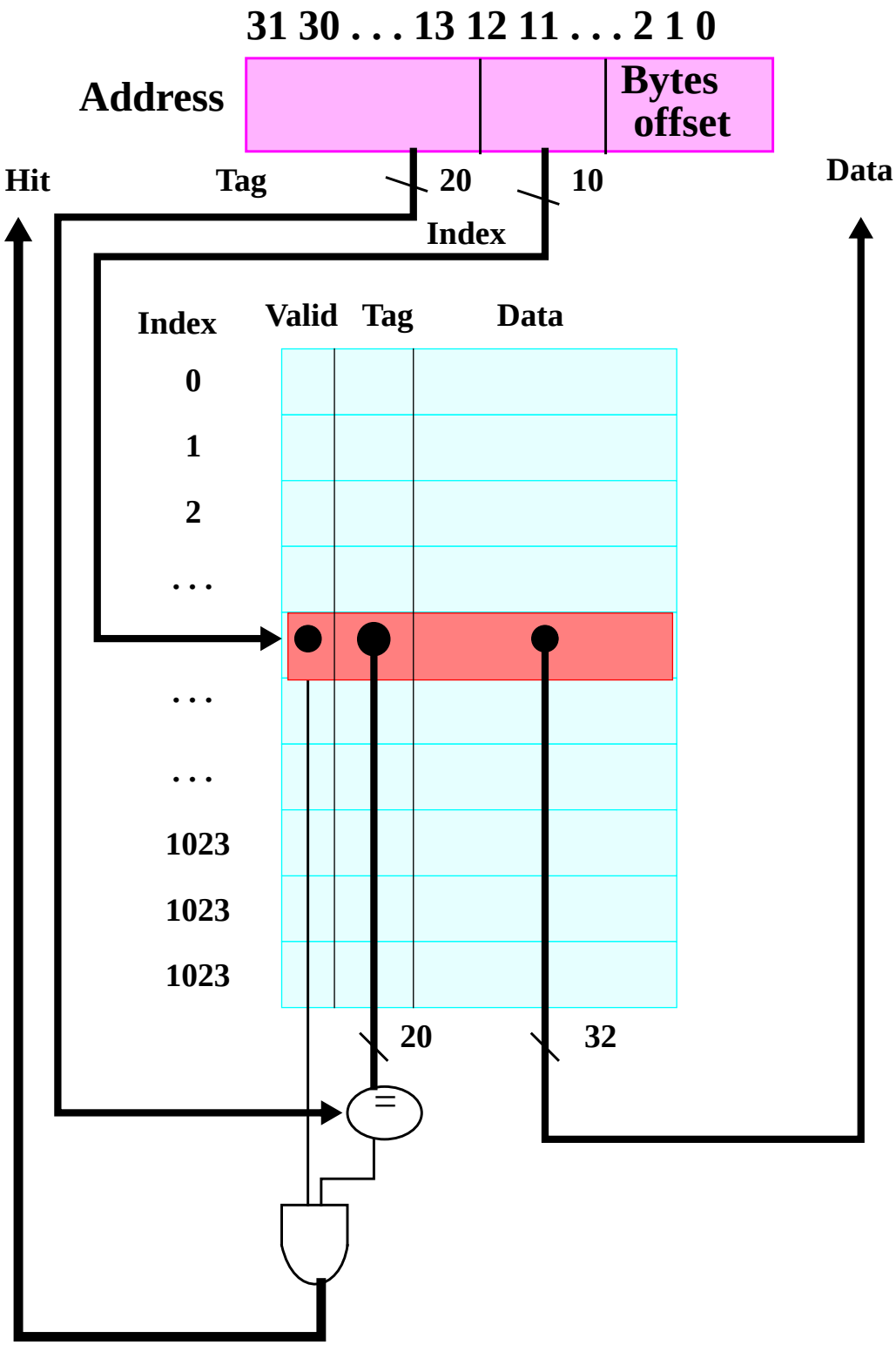
Set associative
(Blk 12 can go anywhere in set 0 (12 mod 4))



Finding a Block in Cache

- Each block in the cache has an address ***tag*** that gives the block address.
- We further need an ***index*** to point to the appropriate location within the cache.
- We also need a ***valid*** bit to indicate whether the item in the cache is valid or not.
- ***Example***: Direct mapped cache:
  - ➔ Cache index selects a location from cache
  - ➔ The tag is compared to that in the cache location.
  - ➔ Data is selected based on the tag result.

Direct Mapped Cache



Cache Size

Question:

How many total bits are required for a direct-mapped cache with 64KB of data, and one word block, assuming a 32-bit address?

Answer:

64KB means 16 K words = 2^{14} words.

Since the block size is one word, we have 2^{14} blocks.

Each block has a 32 bit data plus a tag, which is 32 – 14 – 2 bits, plus a valid bit.

The total cache size is

2^{14} * (32 + (32 – 14 – 2) + 1)

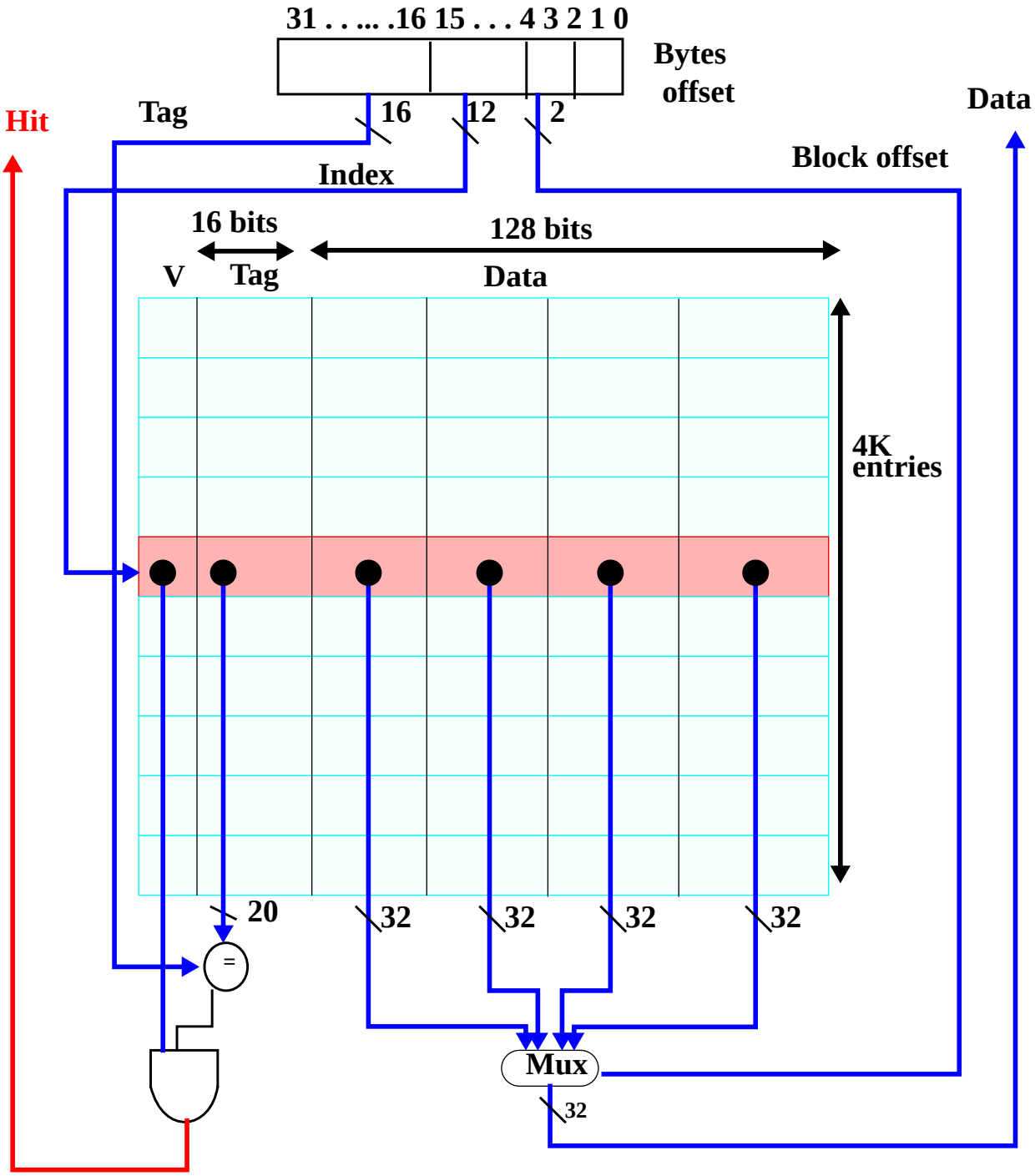
= 2^{14} * 49

= 784 * 2^{10}

= 784 bits

Direct Mapped Cache

Taking advantage of spatial locality:



Bock Replacement

- When a miss occurs, the cache controller must select a block to be replaced with the desired data (with direct mapped there is only one choice).

  - ***Random:*** To spread allocation uniformly, candidate blocks are randomly selected.

  - ***Least-recently used (LSU):*** The block replaced is the one that has been unused for the longest time.

The Write Strategy

- **There are two basic options when writing to the cache:**

  - ***Write through (or store through):*** **The information is written to both the block in the cache *and* to the block in the lower-level memory.**

  - ***Write back (store back):*** **The information is written only to the block in the cache. The modified cache block is written to main memory only when it is replaced.**

Further Reading

Chapter 7. **David A. Patterson and John L. Hennessy. *Computer Organization & Design: The Hardware / Software Interface*. Morgan Kaufman (page 540-555).**