CS209 Computer Architecture

Q2 Cache Memory-II

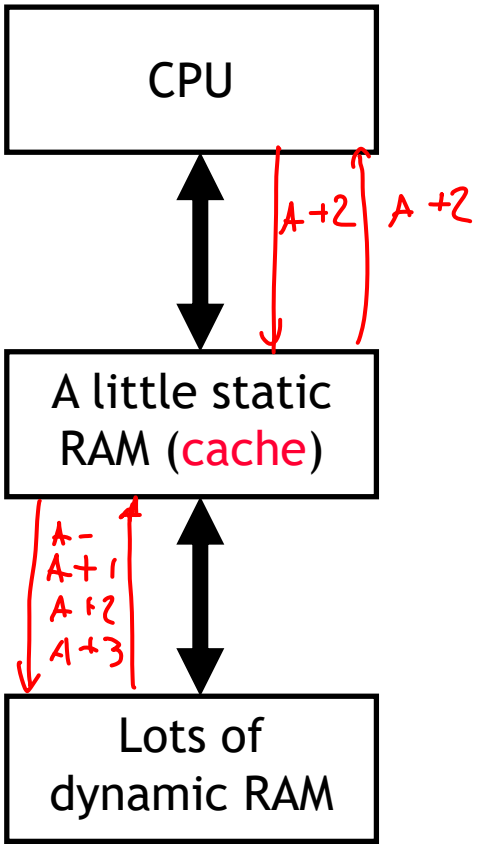
Somanath Tripathy
IIT Patna

Introduction

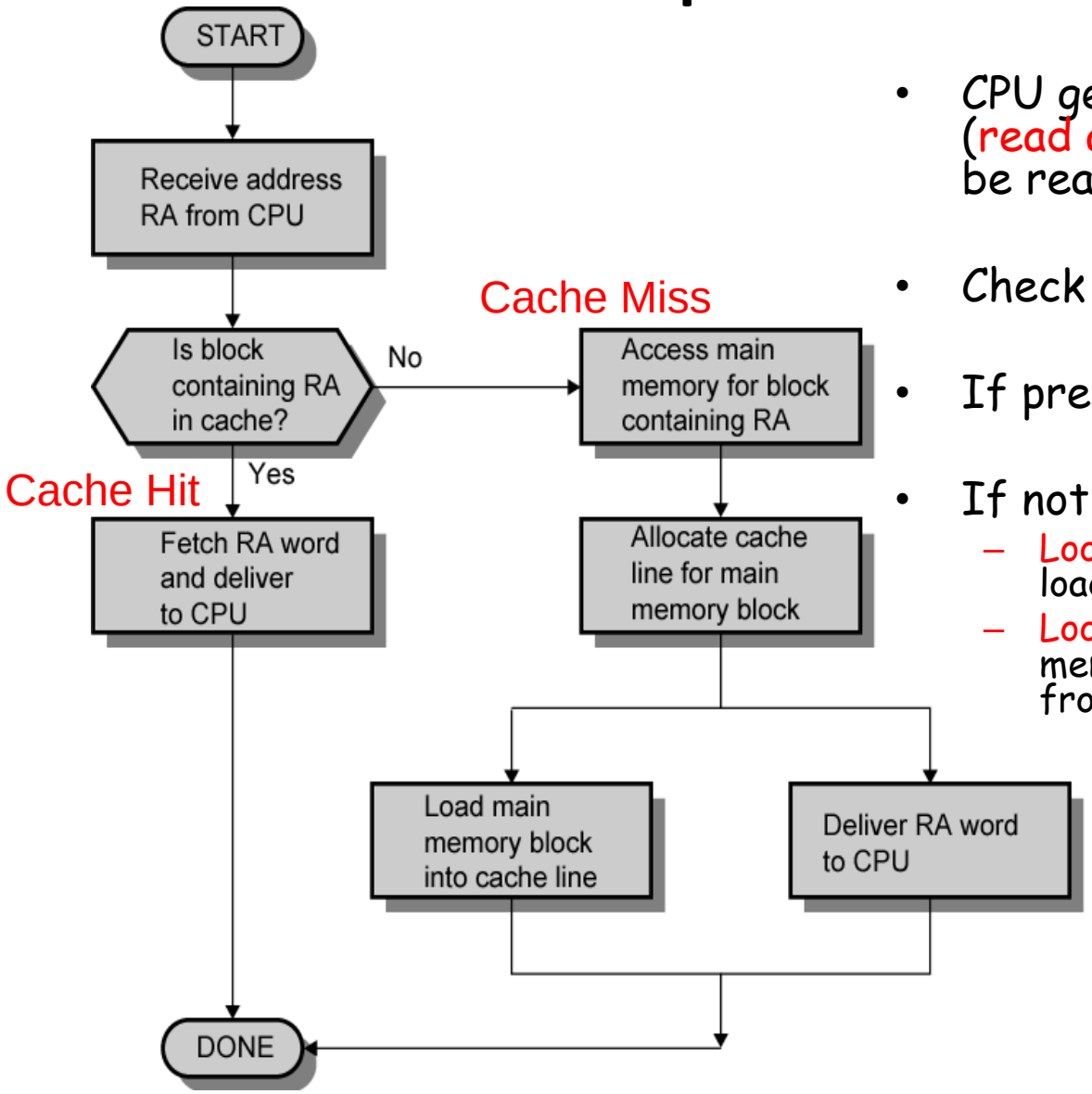
- Cache Memory

```
int array[SIZE];
int A = 0;

for (int i = 0 ; i < 20000 ; ++ i) {
   for (int j = 0 ; j < SIZE ; ++ j)
   {
        A += array[j];
   }
}
```

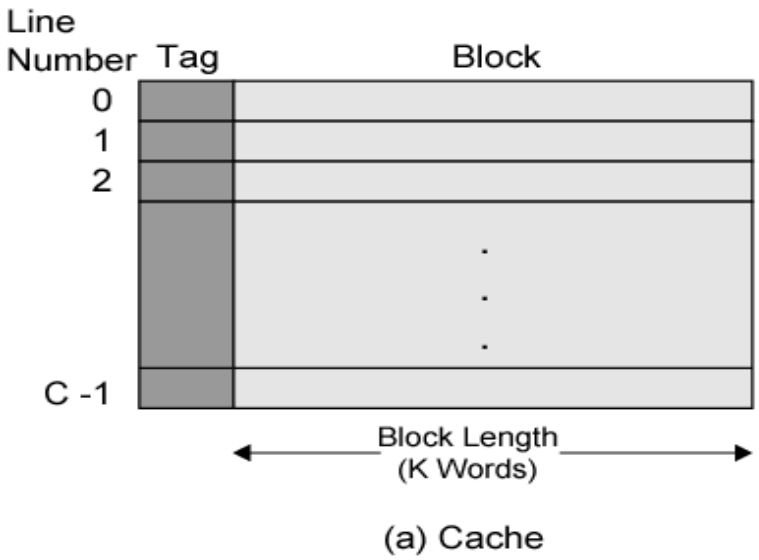


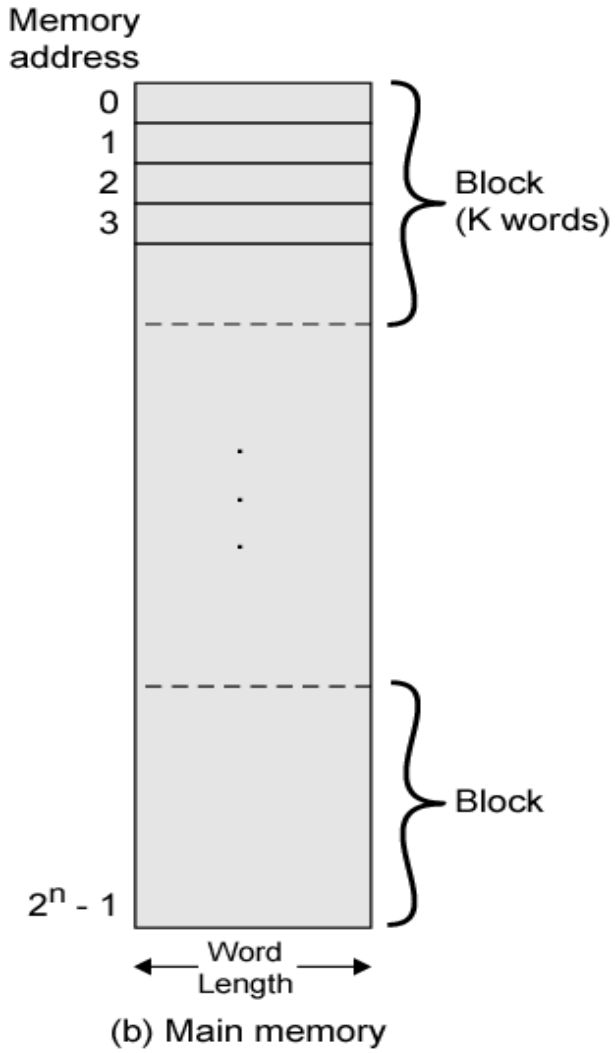
Cache Read Operation - Flowchart



- CPU generates the address (read address RA) of a word to be read
- Check cache for this data word
- If present, get from cache (fast)
- If not present,
  - Look aside: processor AND cache loaded from memory in parallel
  - Look through: cache loaded from memory, then processor loaded from cache.

Cache/Main Memory Structure





- Cache includes tags to identify which block of main memory is in each cache slot

Cache Design parameters

- Size
  - Cost
    - More cache is expensive
  - Speed
    - More cache is faster (up to a point)
    - Checking cache for data takes time
- Mapping Function
- Adding Tags
- Replacement Algorithm
- Write Policy

4 Questions:

1. When we copy a block of data from main memory to the cache, where exactly should we put it?
2. How can we tell if a word is already in the cache, or if it has to be fetched from main memory first?
3. Eventually, the small cache memory might fill up. To load a new block from main RAM, we'd have to replace one of the existing blocks in the cache... which one?
4. How can *write* operations be handled by the memory system?

Mapping Function

- How to Map (How the cache is organized)
  - Direct
  - Associative
  - Set associative

- Assume:
  - Cache of 64kByte
  - Cache block of 4 bytes
    - i.e. cache has 16k (2^{14}) lines of 4 bytes
  - 16MBytes main memory
    - 24 bit address

Direct Mapping

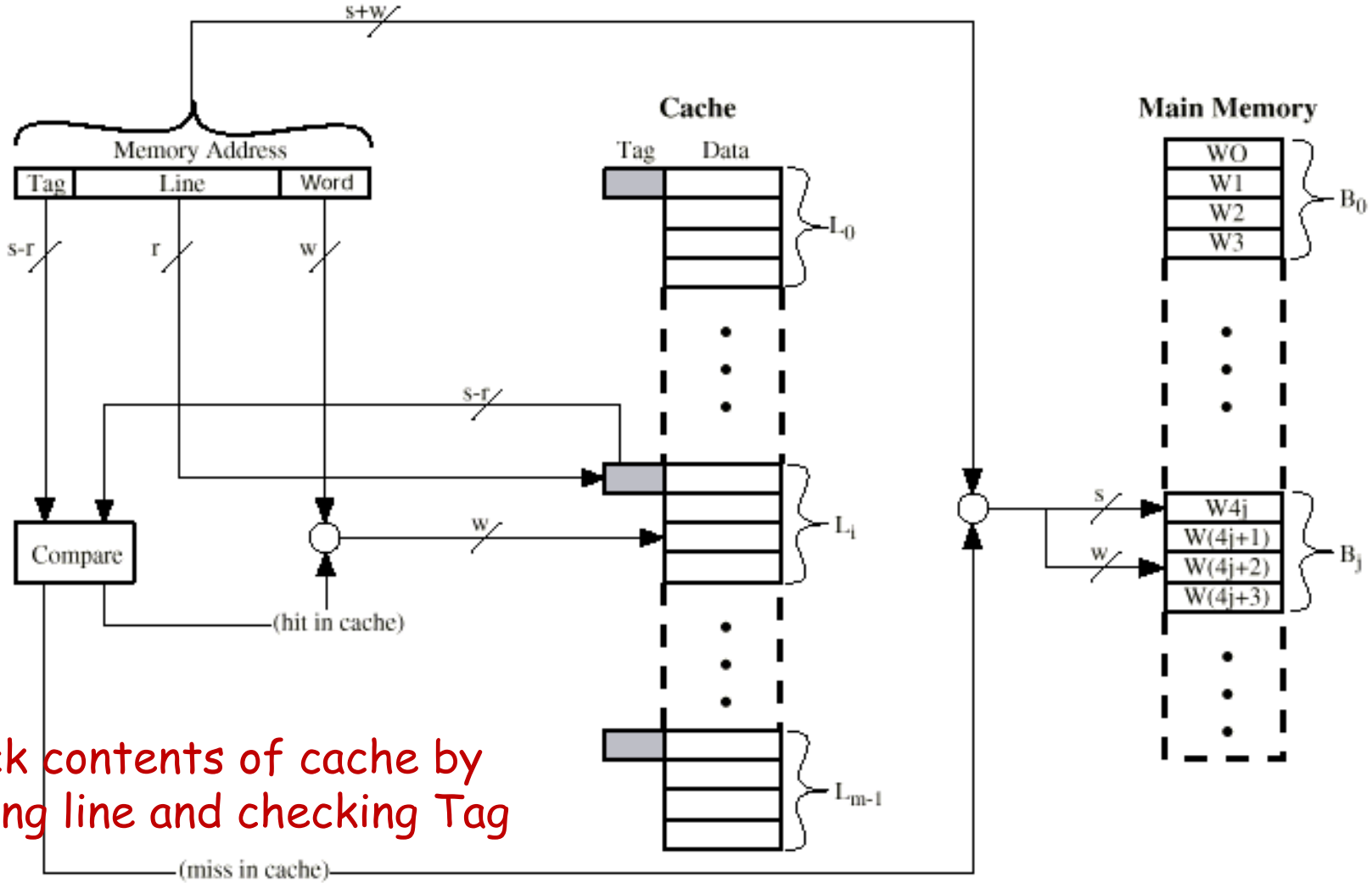
– Cache of 64kByte: Cache block of 4 bytes
  • i.e. cache has 16k (2^{14}) lines of 4 bytes
– 16MBytes main memory: 24 bit address

Address Structure

Tag s-r	Line or Slot r	Word w
8	14	2

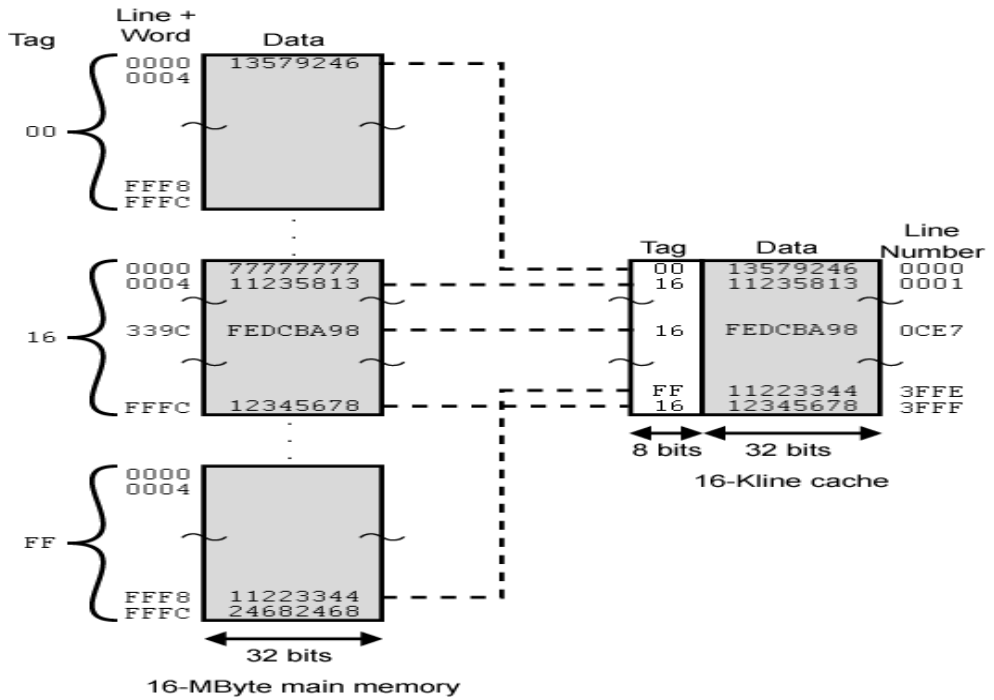
- Each block of main memory maps to only one cache line
  - i.e. if a block is in cache, it must be in one specific place
- Address is in two parts
  - Least Significant w bits identify unique word
  - Most Significant s bits specify one memory block
  - The MSBs are split into a cache line field r and a tag of s-r (most significant)

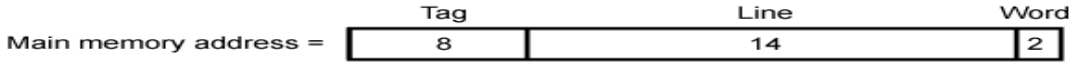
Direct Mapping Cache Organization



Check contents of cache by finding line and checking Tag

Direct Mapping Example



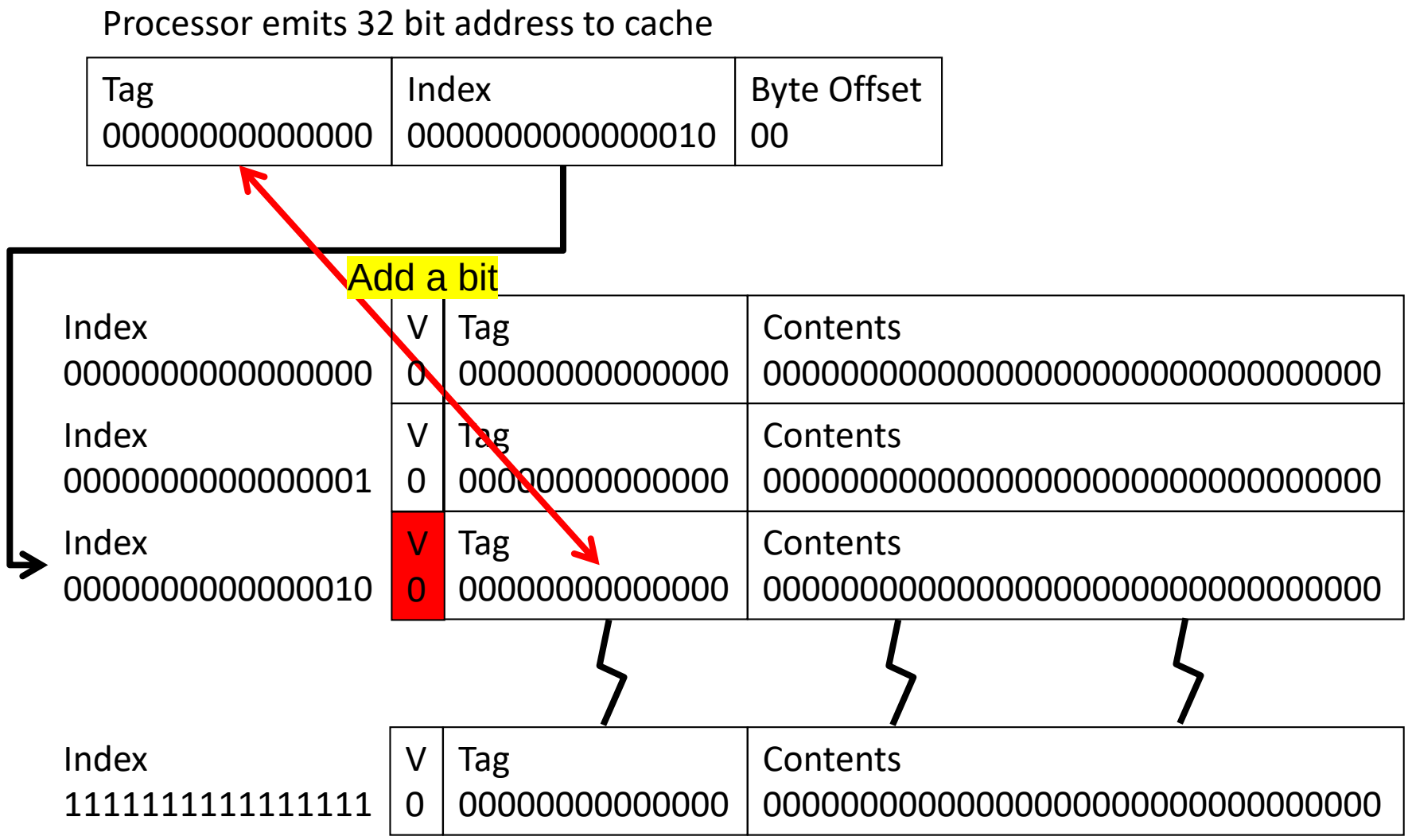


e.g

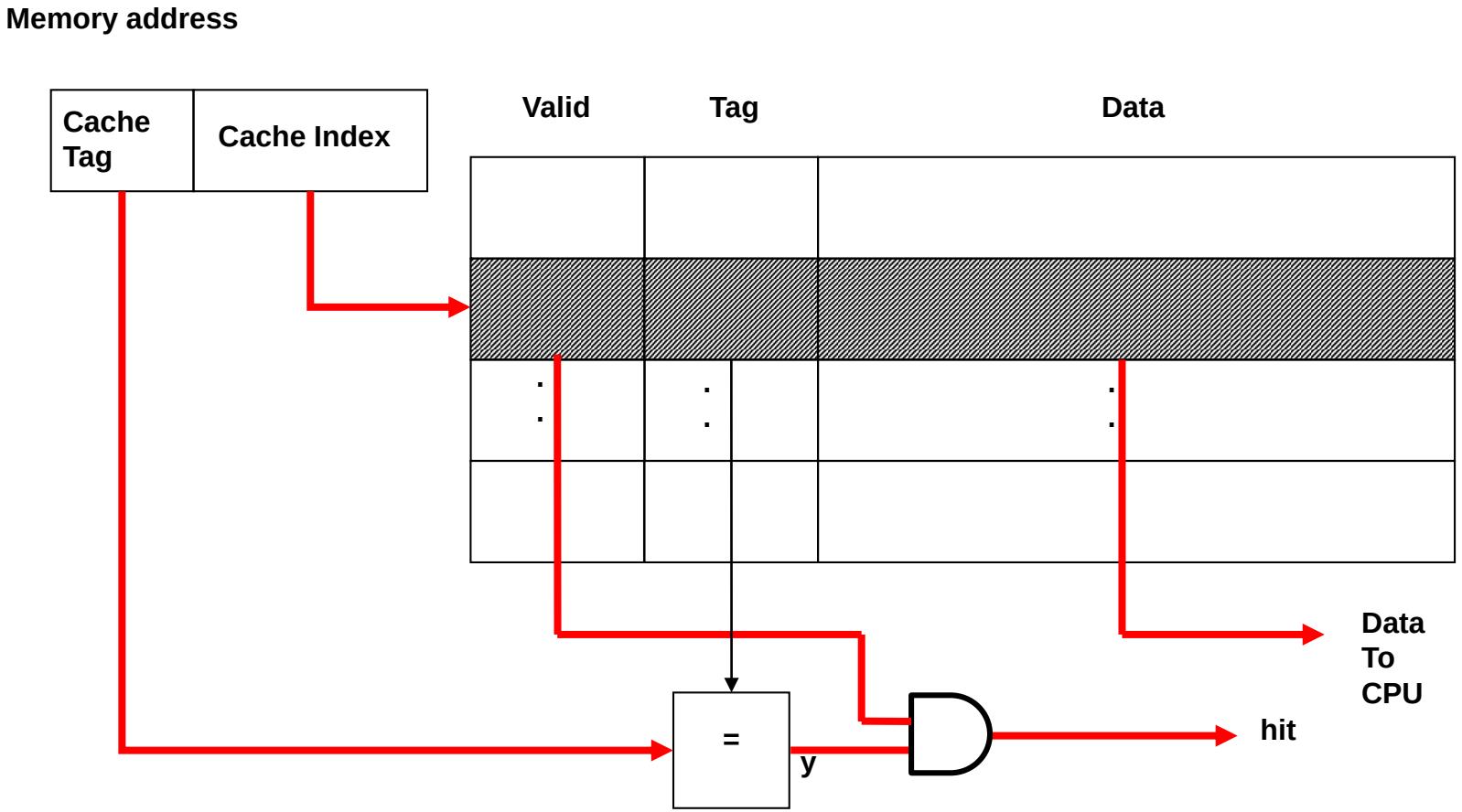
Address	Tag	Data	Set number
FFFFF8	FF	11223344	3FFE

Thought Question

Each cache entry contains a bit indicating if the line is valid or not. Initialized to invalid



Hardware for direct mapped cache



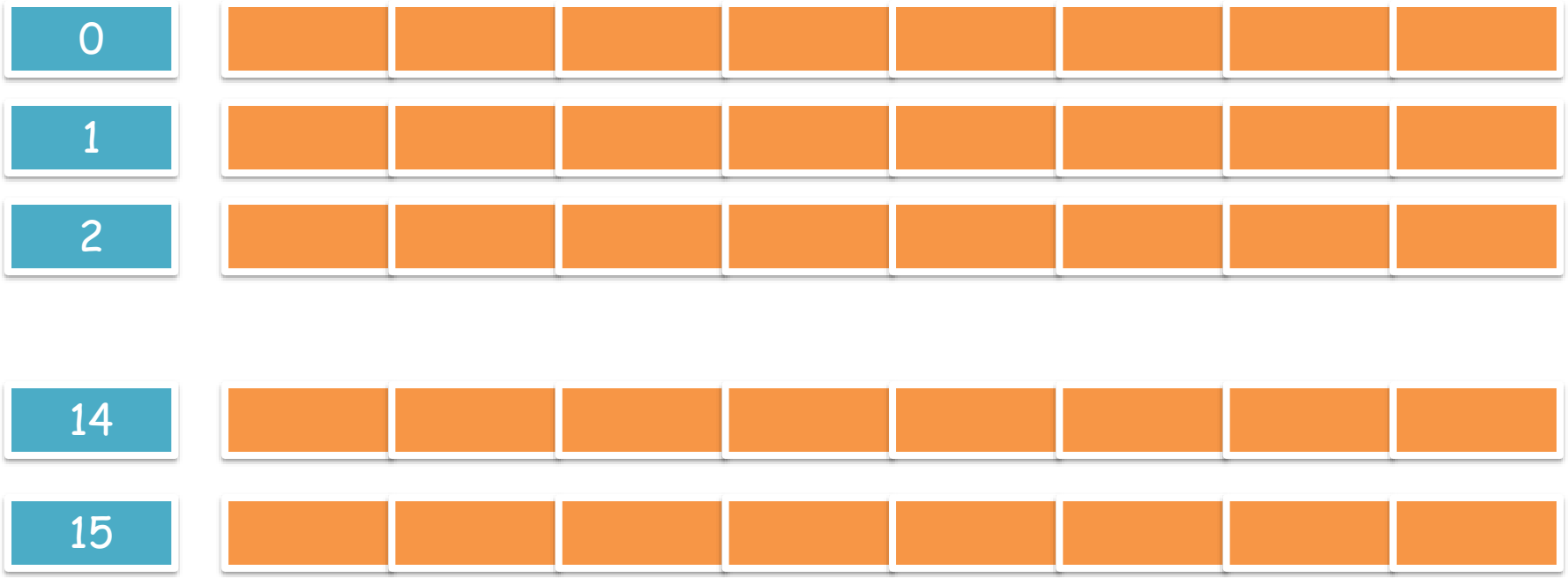
Direct Mapping pros & cons

- Pros
  - Simple
  - Inexpensive
- Cons
  - Fixed location for given block
    - If a program accesses 2 blocks that map to the same line repeatedly, cache misses are very high
      - Thrashing

Program Cache Hit/Miss

Cache model

- Direct mapped 8 word per line



Program

```
int A[128];
for(i=0;i<128;i++){
    A[i]=i;
}
```

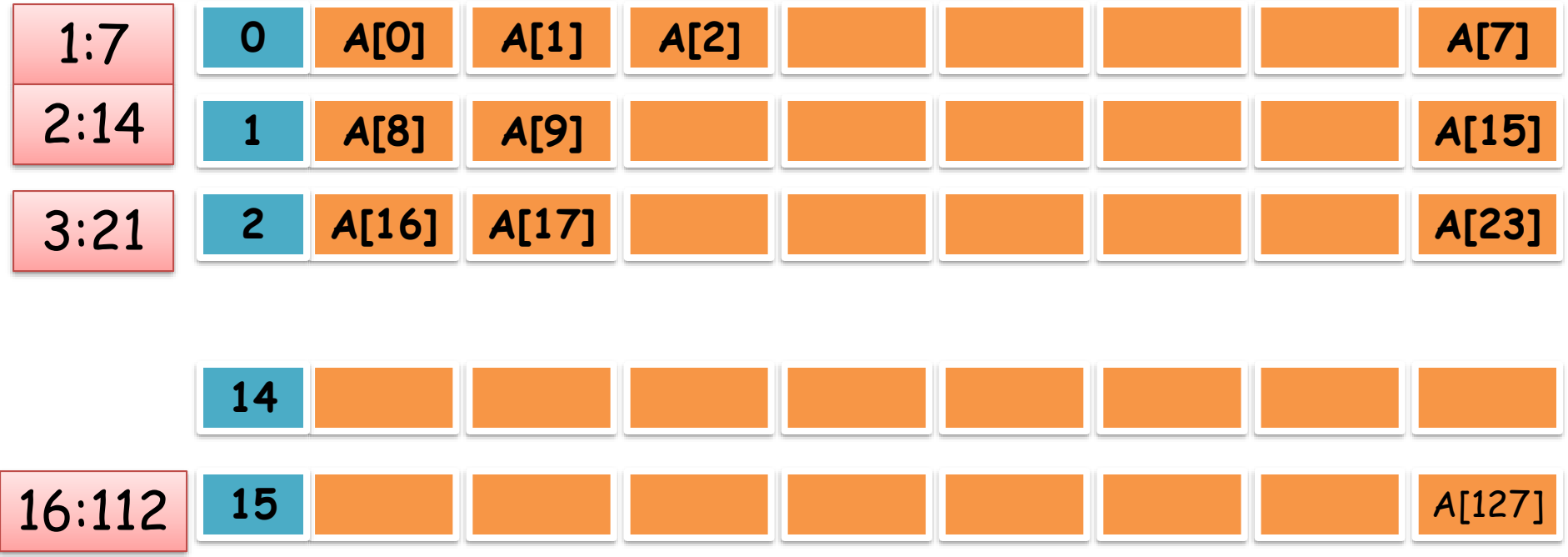
- Assume &A=000000, **Behavior of only Data**
- Scalar variable {i} mapped to register
- Data have to moved from cache/memory

Cache perf. : Data Size <= Cache Size

```
int A[128];
for(i=0;i<128;i++){
    A[i]=i;
}
```

Scalar mapped to register
Vector mapped to memory

1:7= 1miss:7heat

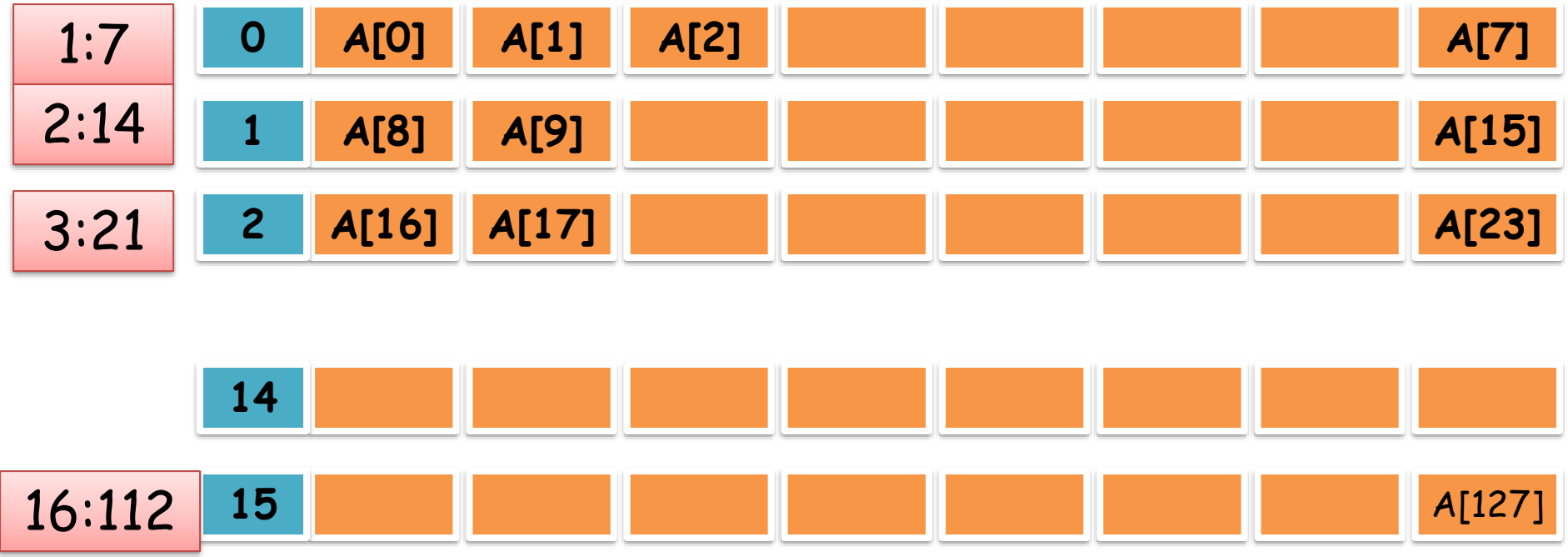


Cache perf. : Data Size > Cache Size

```
int A[512];
for(i=0;i<512;i++){
    A[i]=i;
}
```

Scalar mapped to register
Vector mapped to memory

1:7= 1miss:7heat

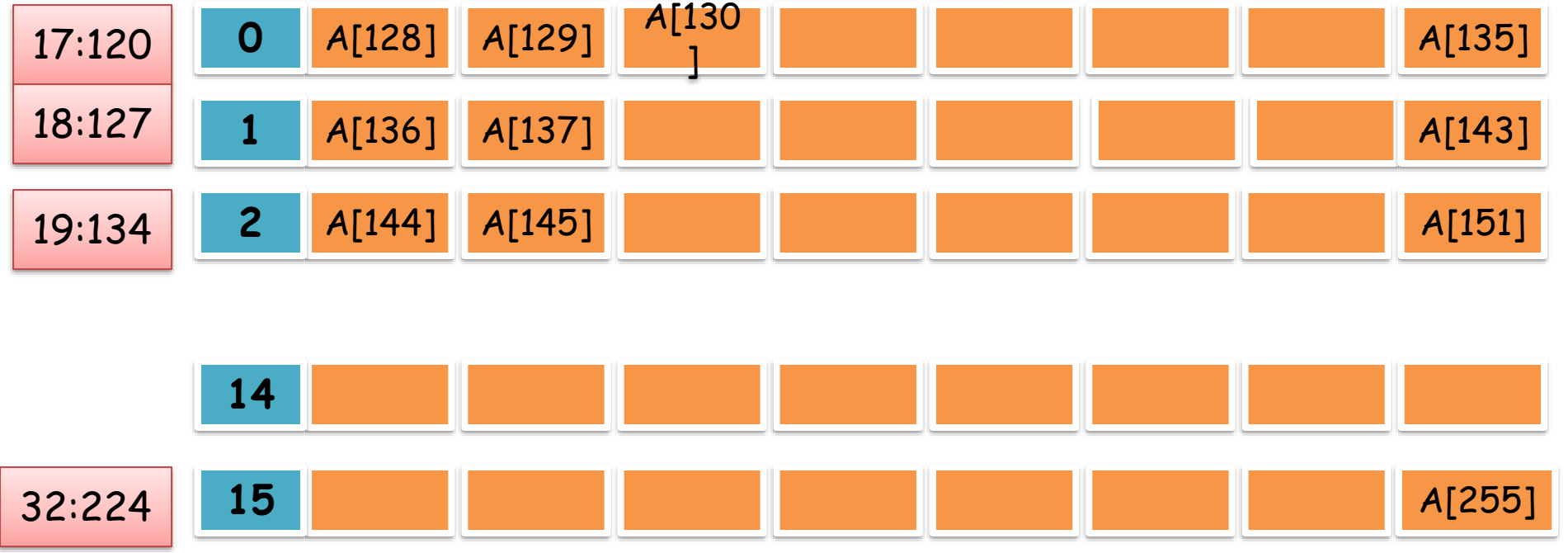


Cache perf. : Data Size <= Cache Size

```
int A[512];
for(i=0;i<512;i++){
    A[i]=i;
}
```

Scalar mapped to register
Vector mapped to memory

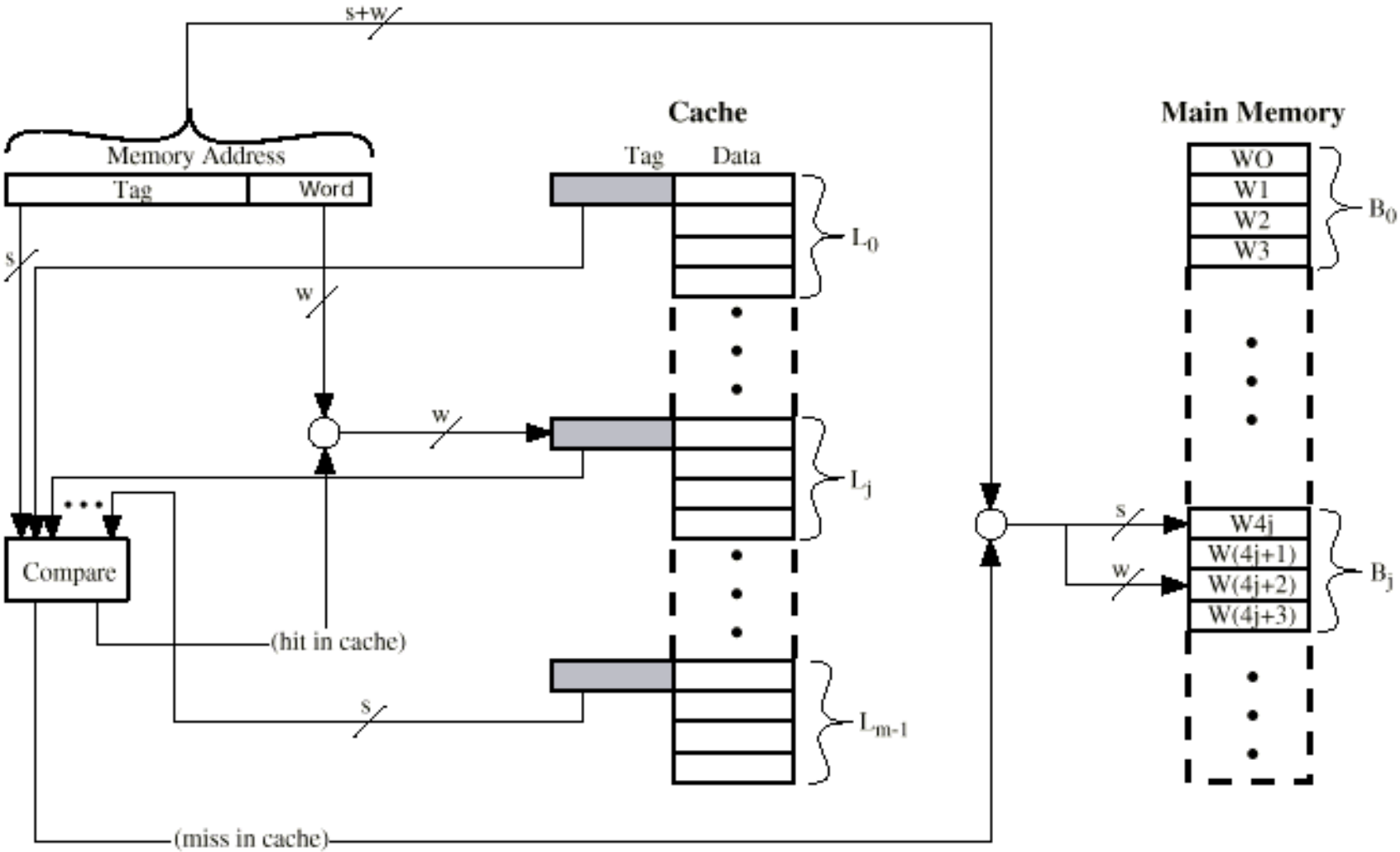
32:224= ==➔
64m:448h



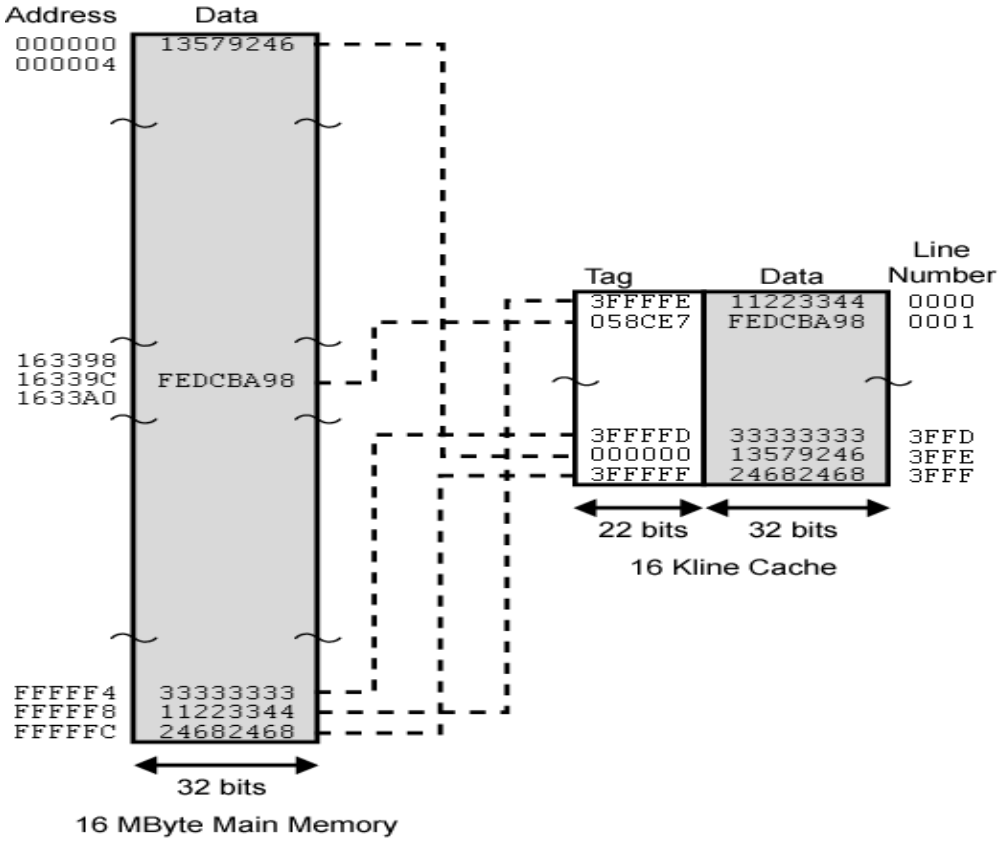
Associative Mapping

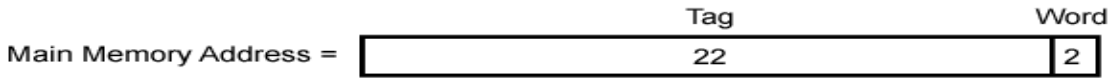
- A main memory block can load into any line of cache
- Memory address is interpreted as tag and word
- Tag uniquely identifies block of memory
- Every line's tag is examined for a match
- Cache searching gets expensive

Fully Associative Cache Organization



Associative Mapping Example





e.g.

Address	Tag	Data	Cache line
FFFFFC	3FFFFF	24682468	3FFF

Associative Mapping Address Structure

Tag 22 bit	Word 2 bit

- 22 bit tag stored with each 32 bit block of data
- Compare tag field with tag entry in cache to check for hit
- Least significant 2 bits of address identify which word is required from the data block
- Disadvantage: Complex circuitry required to examine the tags of all the cache lines in parallel.

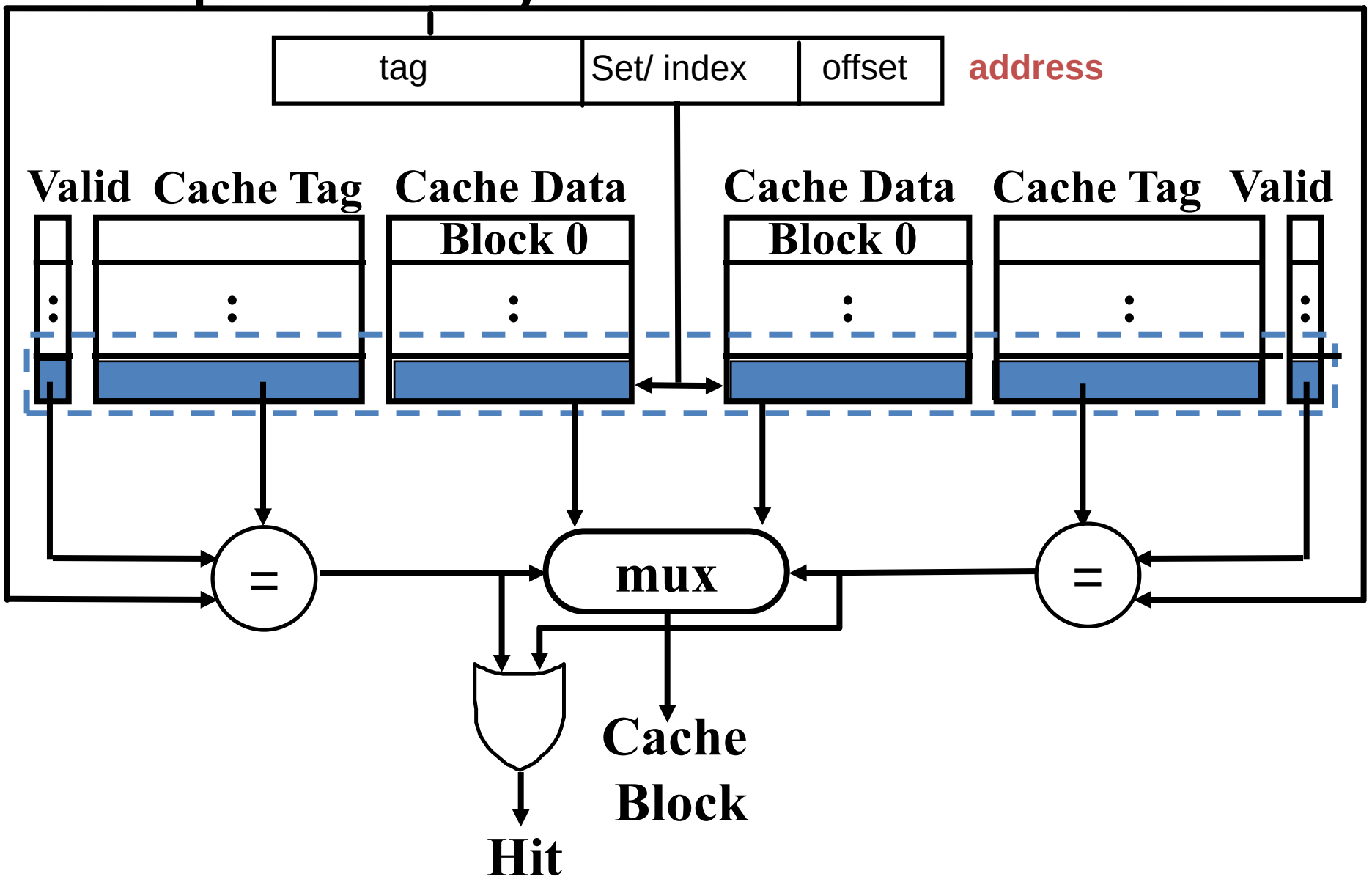
Set Associative Mapping

- To exhibit the strength of both direct and associative mapping while reducing their disadvantages

- Cache is divided into a number of sets
- Each set contains a number of lines
- A given block maps to any line in a given set
  - e.g. Block B can be in any line of set i

- e.g. 2 lines per set
  - 2 way associative mapping
  - A given block can be in one of 2 lines in only one set

Example: 2-way Set Associative Cache



- Thanks