# Q.3

### What is a cache?

* A cache is a hardware or software component that stores data from memory closer to the processor to remove long delays when fetching data from main memory

### How does a cache reduce the effective memory address access time?

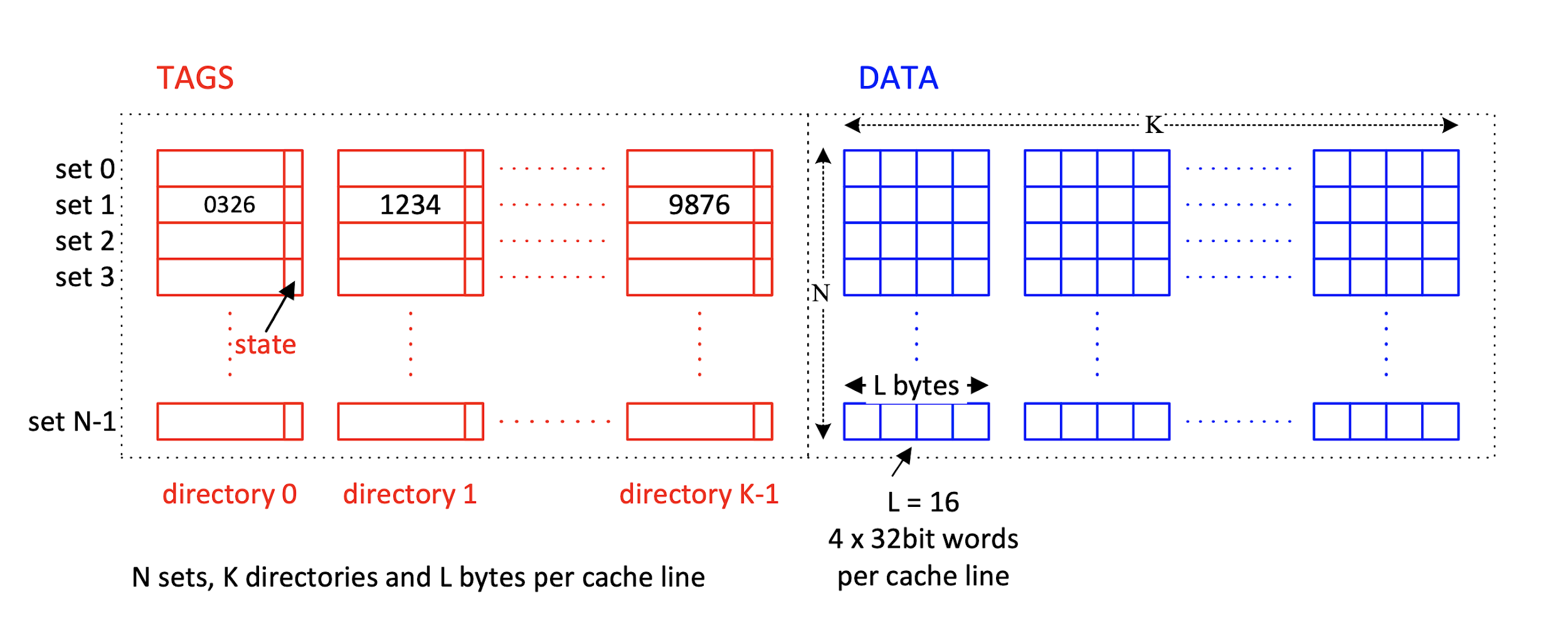
* By using the Tag RAM associated with blocks of data to look for data stored in the cache
* A CPU typically has a much higher clock speed than the system bus. This means that CPU needs to read data from the system memory, it must conform to the system bus speed, even though the bus speed is ridiculous slower compared to the CPU’s clock speed. A CPU cache places a small amount of memory directly in the CPU. This cache is much faster than the system memory because it operates at the CPU’s speed rather than the system bus speed.The idea behind this is that if the data has been requested once, it’s likely to be accessed again.Placing the data on the cache makes it accessible faster.

### Cache organisation LKN

L = number of bytes per cache line

K = number of cache lines (directories) per set

N = number of sets



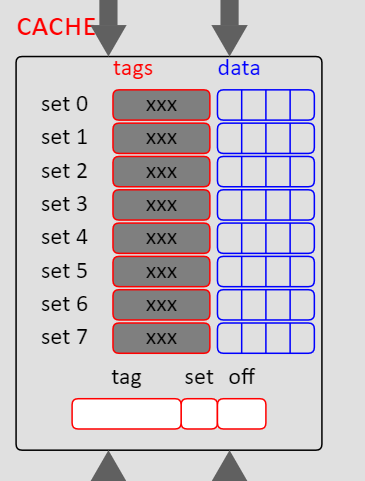
### 

### How a data item is accessed in a LKN cache

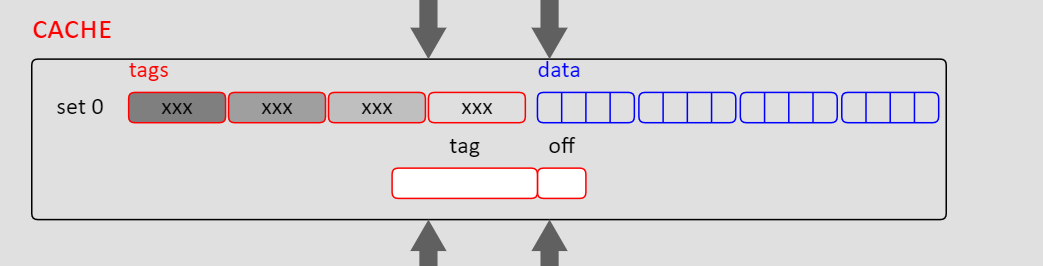
* Address is mapped onto a particular set (set #) by extracting bits from incoming address
* Consider an address that maps to Set 1
  + The Set 1 tag of each K directory is compared w/ the incoming address tag simultaneously
  + If a match is found (hit), corresponding data returned from the offset index within cache line
  + The K data lines in the set are accessed concurrently with the directory entries so that a hit from the directory can be routed to the output buffer quickly
  + If a match is not found (miss), data is read from memory and placed in the cache line within the set and the corresponding tag is updated

### Compute no of hits and misses

Direct Mapped Diagram (N = 8, K =1 , L = 16)



Fully Associative Diagram (N = 1, K = 4, L =16)



* Get the address, change from Hex to Binary
* Last 4 bits indicate the offset
* Next Log2(N) bits give the set
* Tag is the rest
* Keep tabs of the last used tag for each set(For direct mapped).
* No need to calculate set for Fully associative

### 

### Explaining pseudo LRU in IA32 xaiu

* Pseudo-LRU selects the cache line to replace based on an approximation measure of age
* It selects the line that is most likely to not have been accessed recently
* In the algorithm to select the pseudo-LRU, the bits and cache line are organised in a Binary Search Tree
* Each node of the tree has a one-bit flag.
* If the flag = 0, go left to find the pseudo-LRU element
* If the flag = 1, go right
* On access, set bits in the tree to point away from accessed cache line

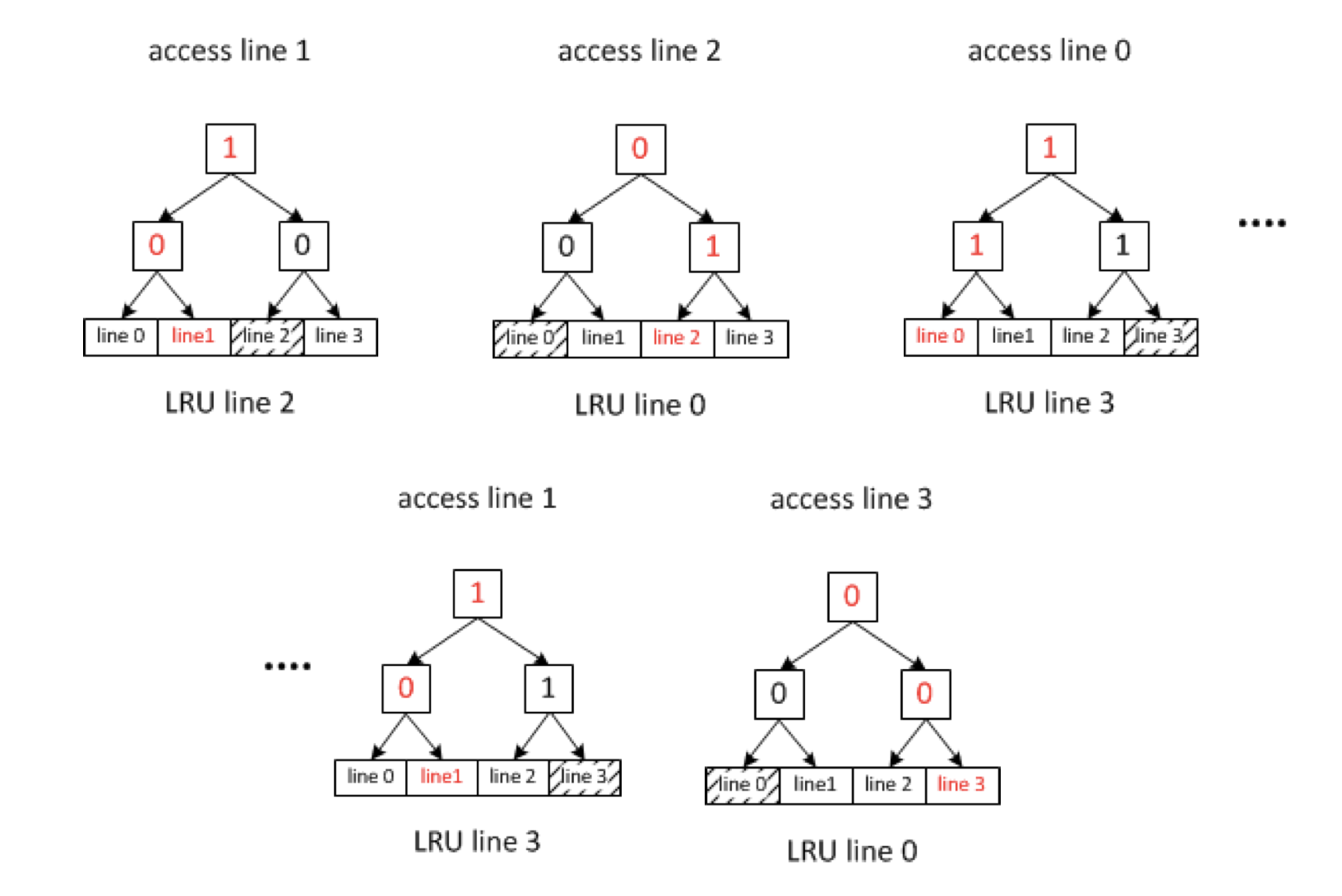
// I don’t know how it achieves this with K-1 bits

### 

### 

### Which tag is considered to be pseudo-LRU

* Access lines in the following order: 1, 2, 0, 1, 3
* Initially all bits are 0



* Line 0 is considered to be pseudo-LRU cause that’s where the tree is pointing to