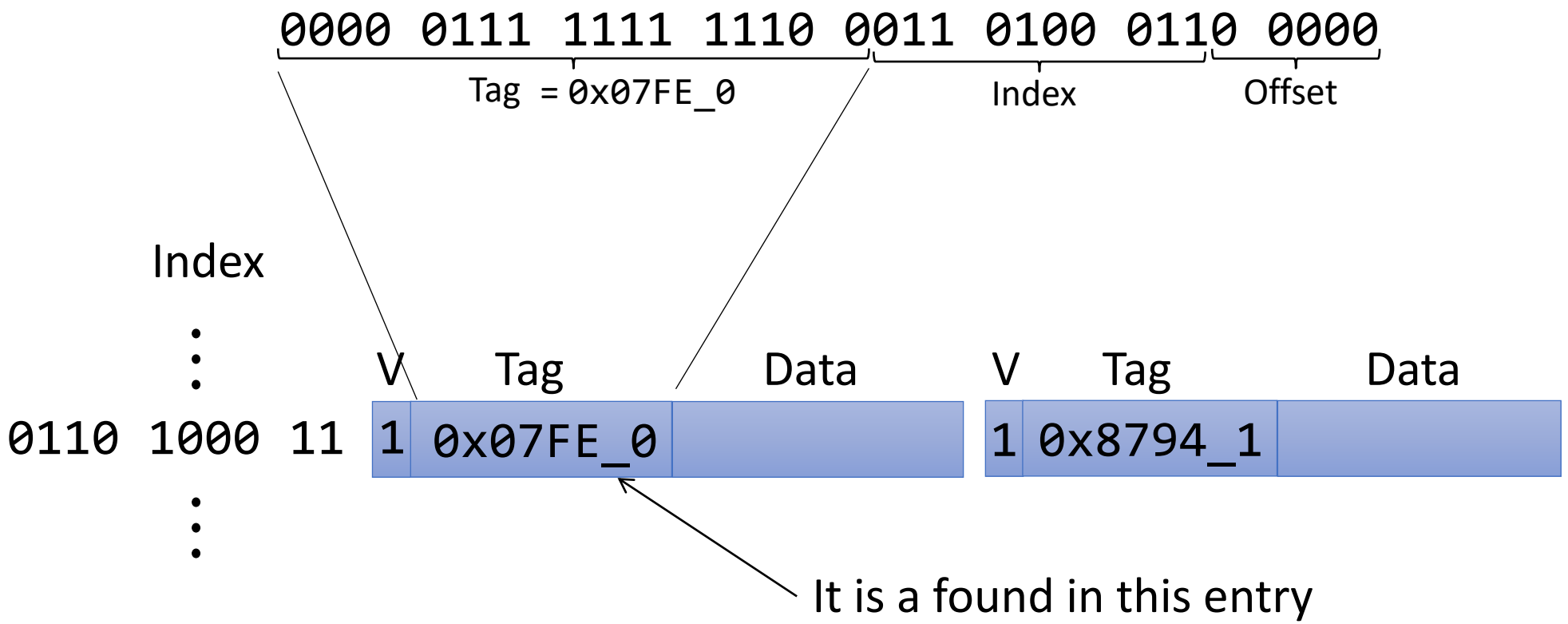
Tree-based Least-Recently Used (LRU) Approximation

José Nelson Amaral

2-way set associative caches

1st Memory reference: `0x07FE 3460`

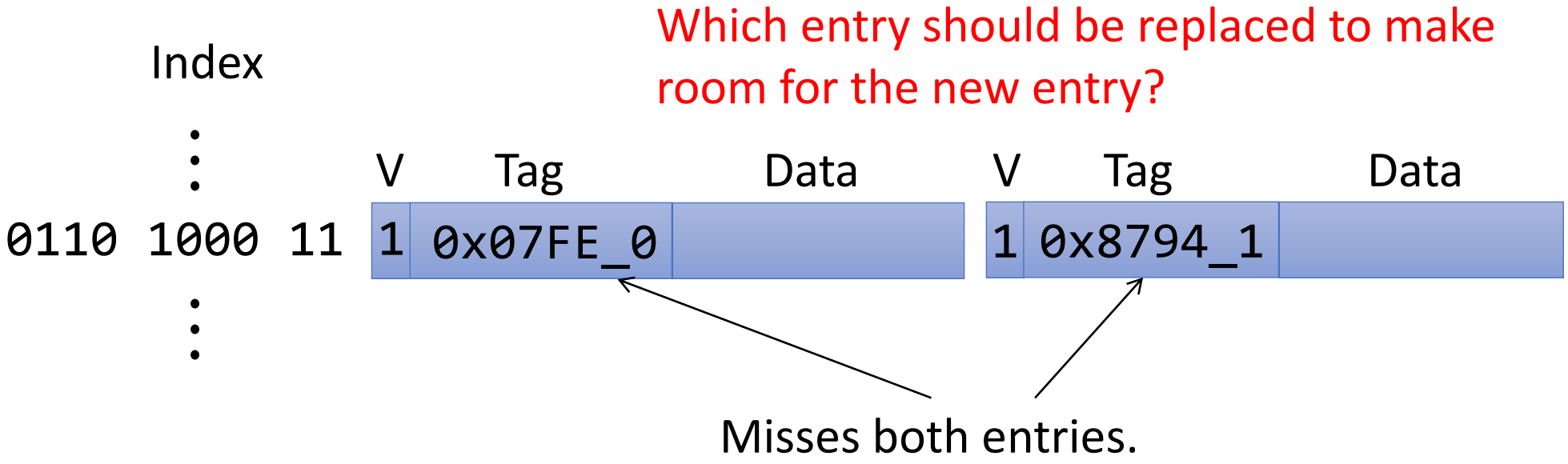


2-way set associative caches

New Memory reference: `0x6541 B468`

`0110 0101 0100 0001 1011 0100 0110 1000`

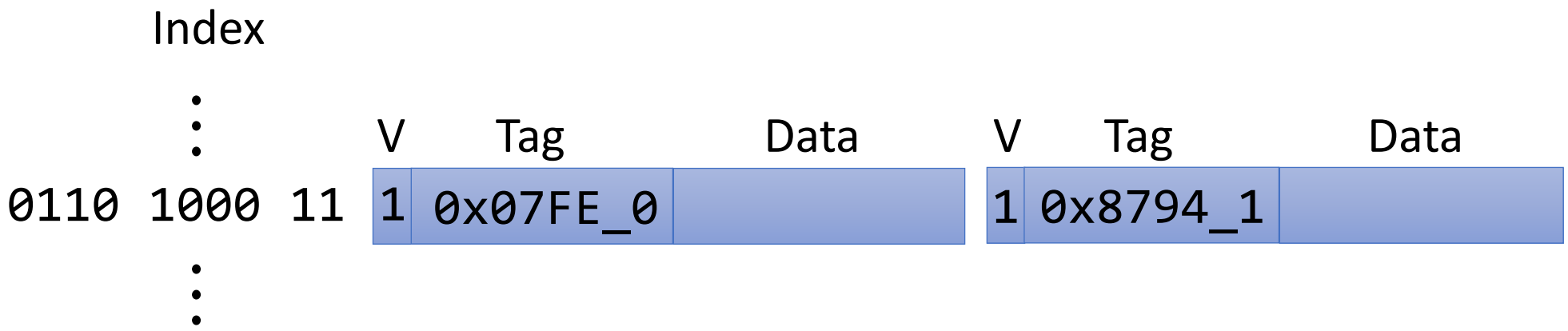
Tag = `0x6541_1` | Index | Offset



2-way set associative caches

LRU: **L**east-**R**ecently **U**sed entry.

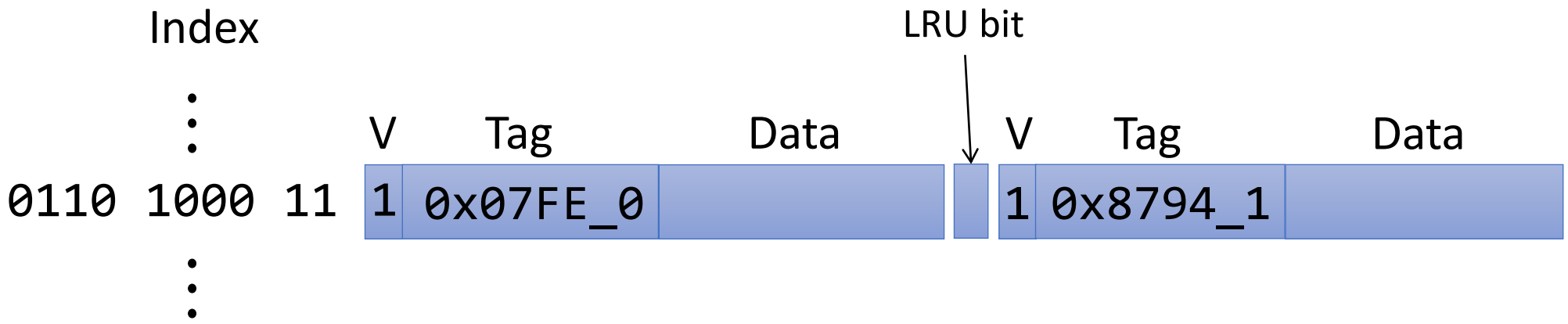
MRU: **M**ost-**R**ecently **U**sed entry.



2-way set associative caches

LRU: **L**east-**R**ecently **U**sed entry.

MRU: **M**ost-**R**ecently **U**sed entry.

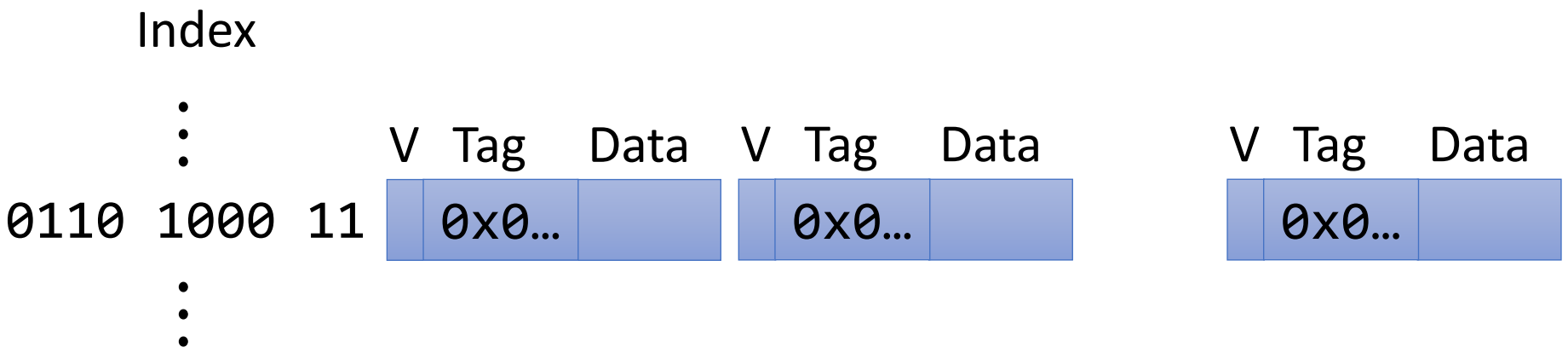


n-way set associative caches

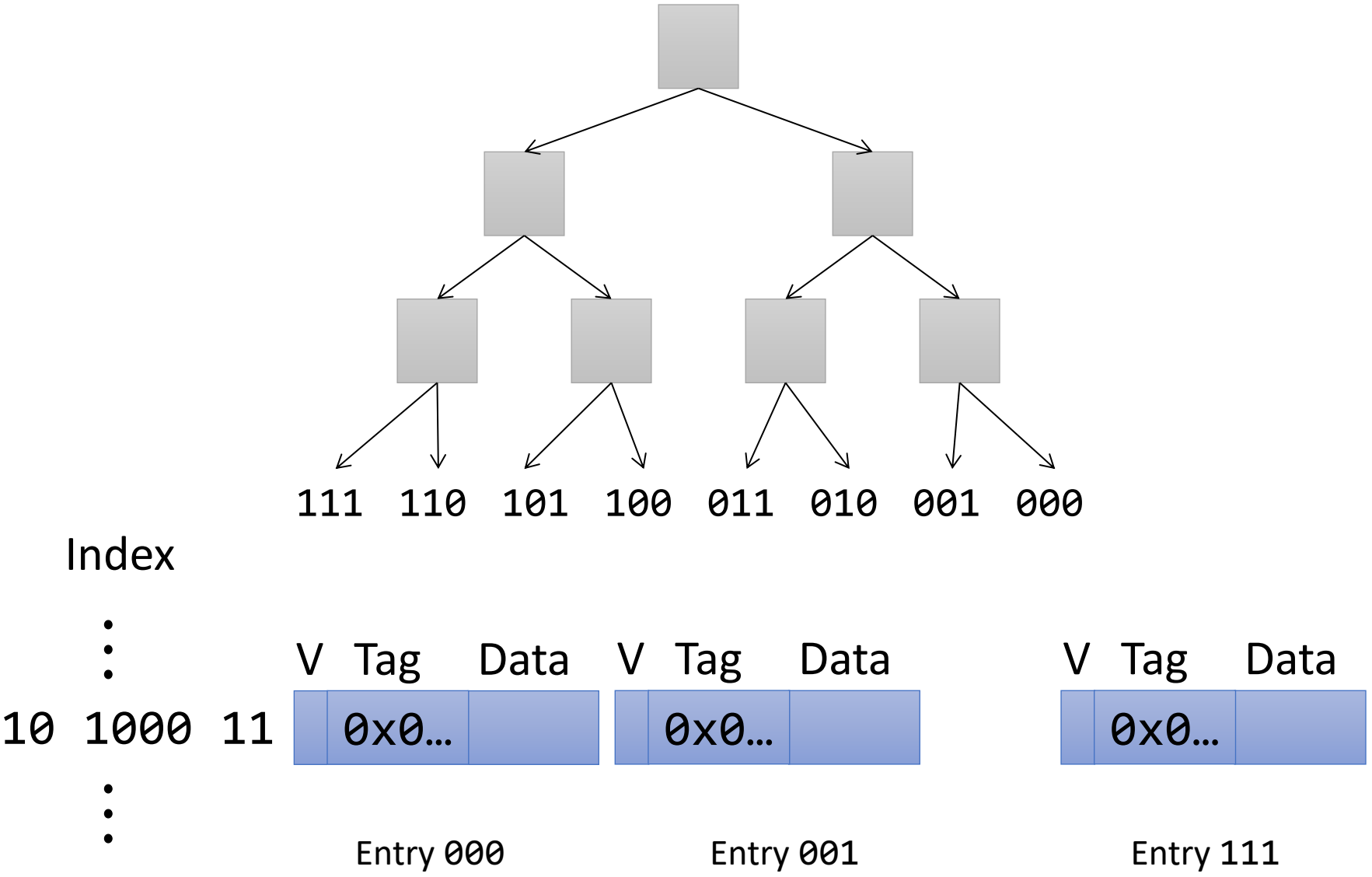
An **LRU** bit is not a solution.

➡ Computing true **LRU** entry is too expensive.

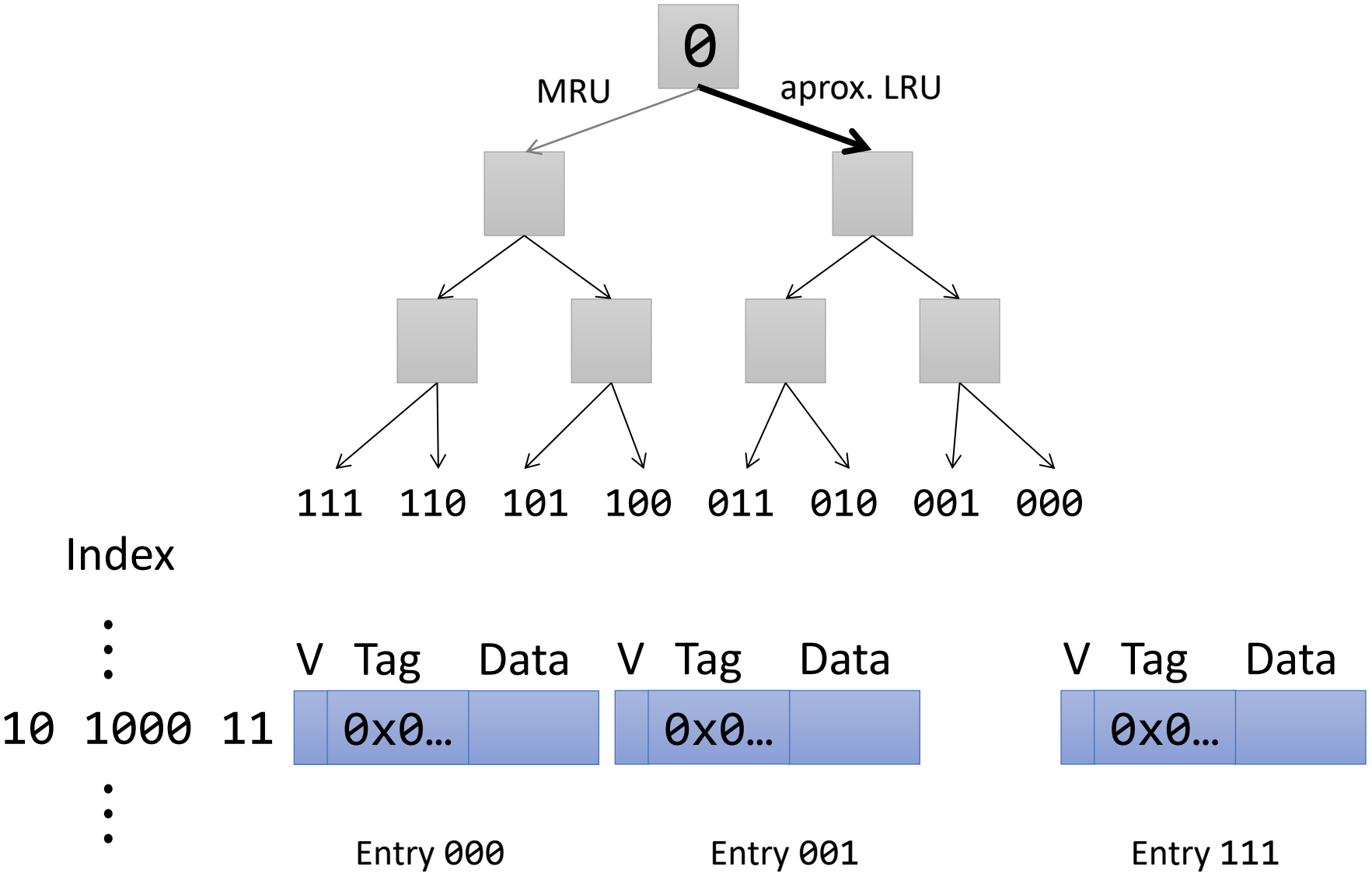
➡ Solution: use an **LRU** approximation.



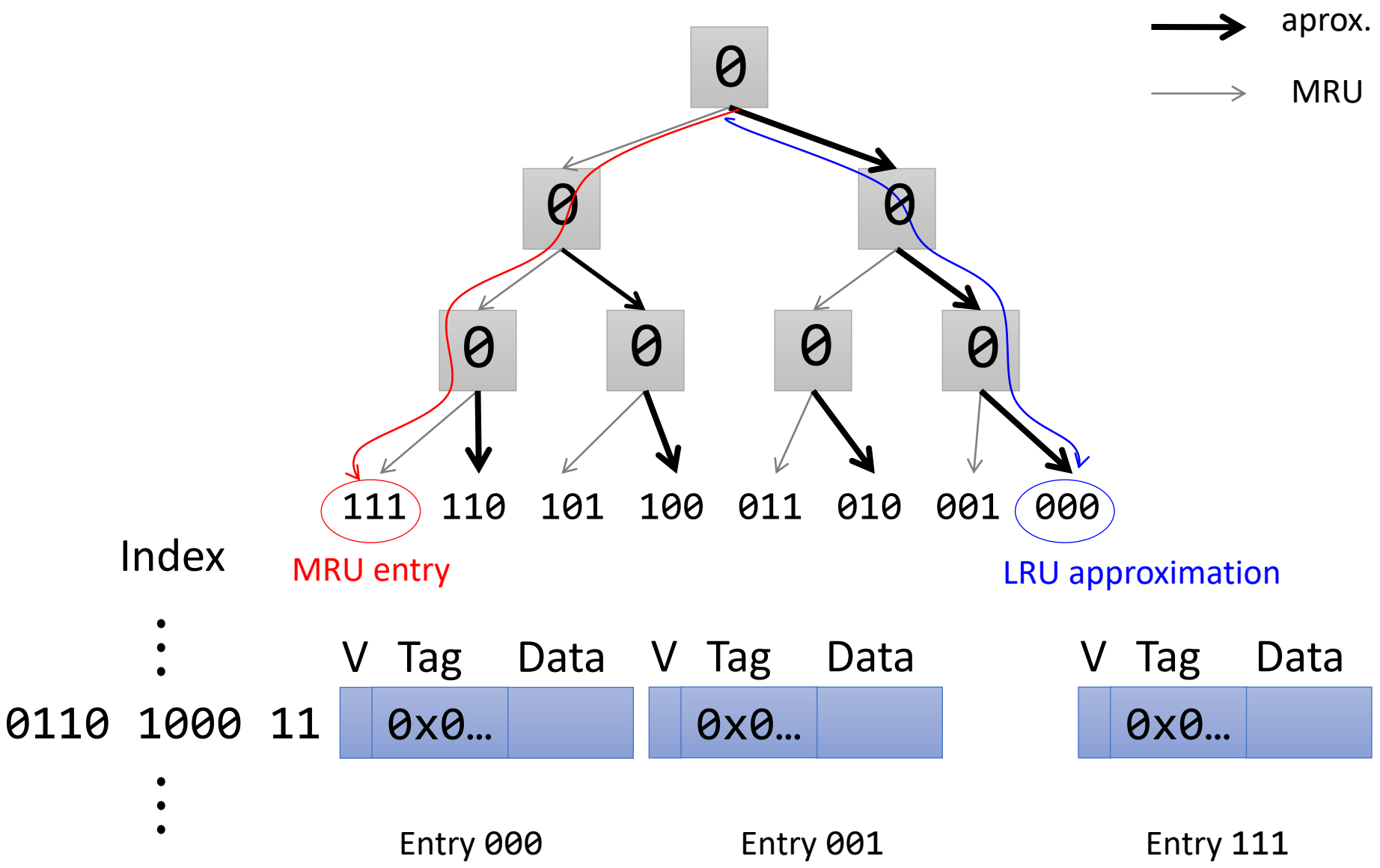
LRU Approximation



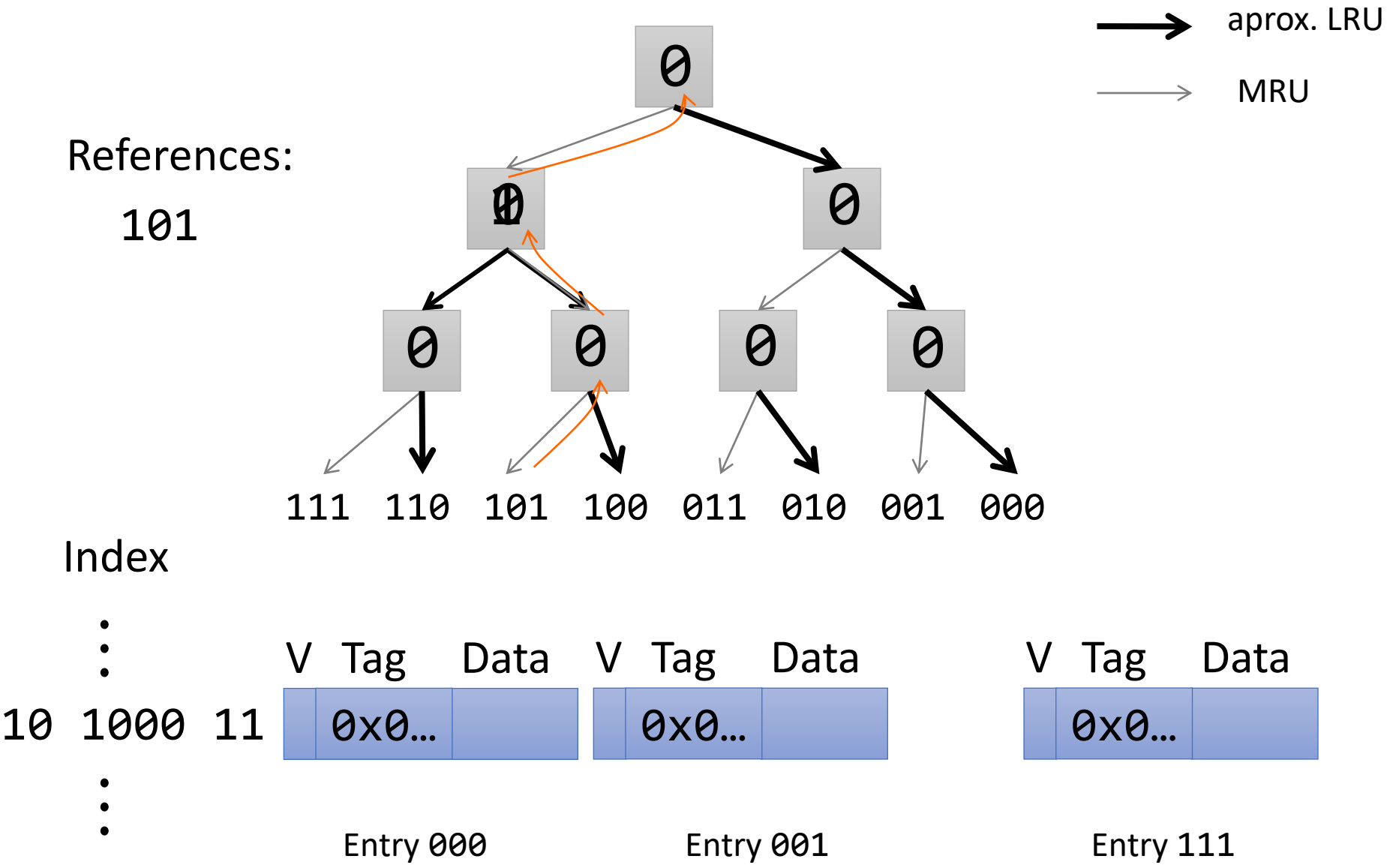
LRU Approximation



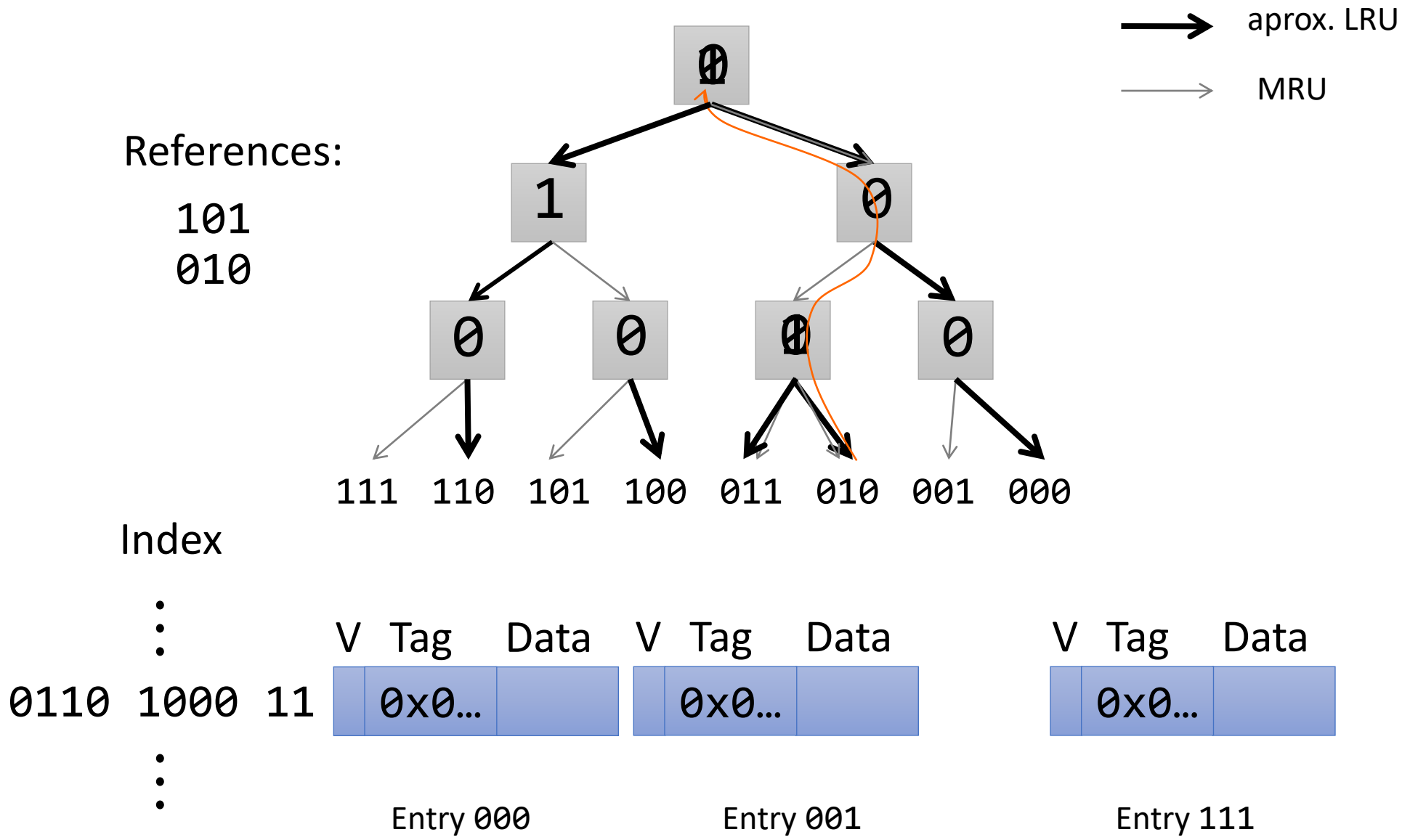
LRU Approximation



LRU Approximation



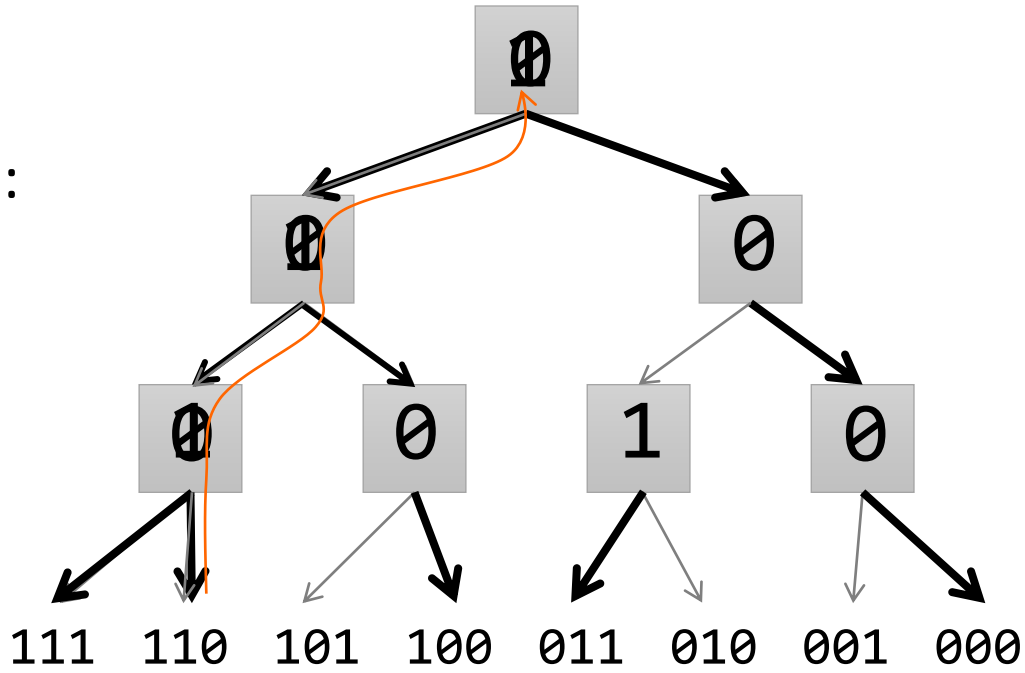
LRU Approximation



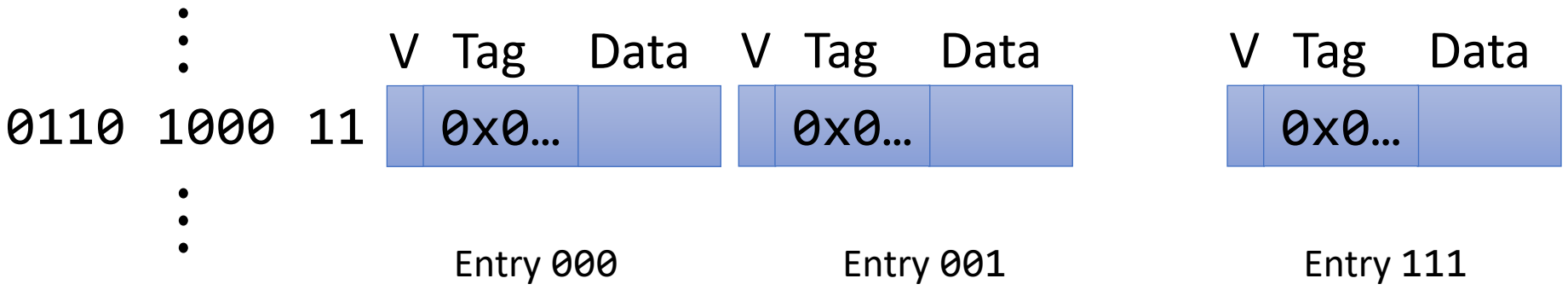
LRU Approximation

References:

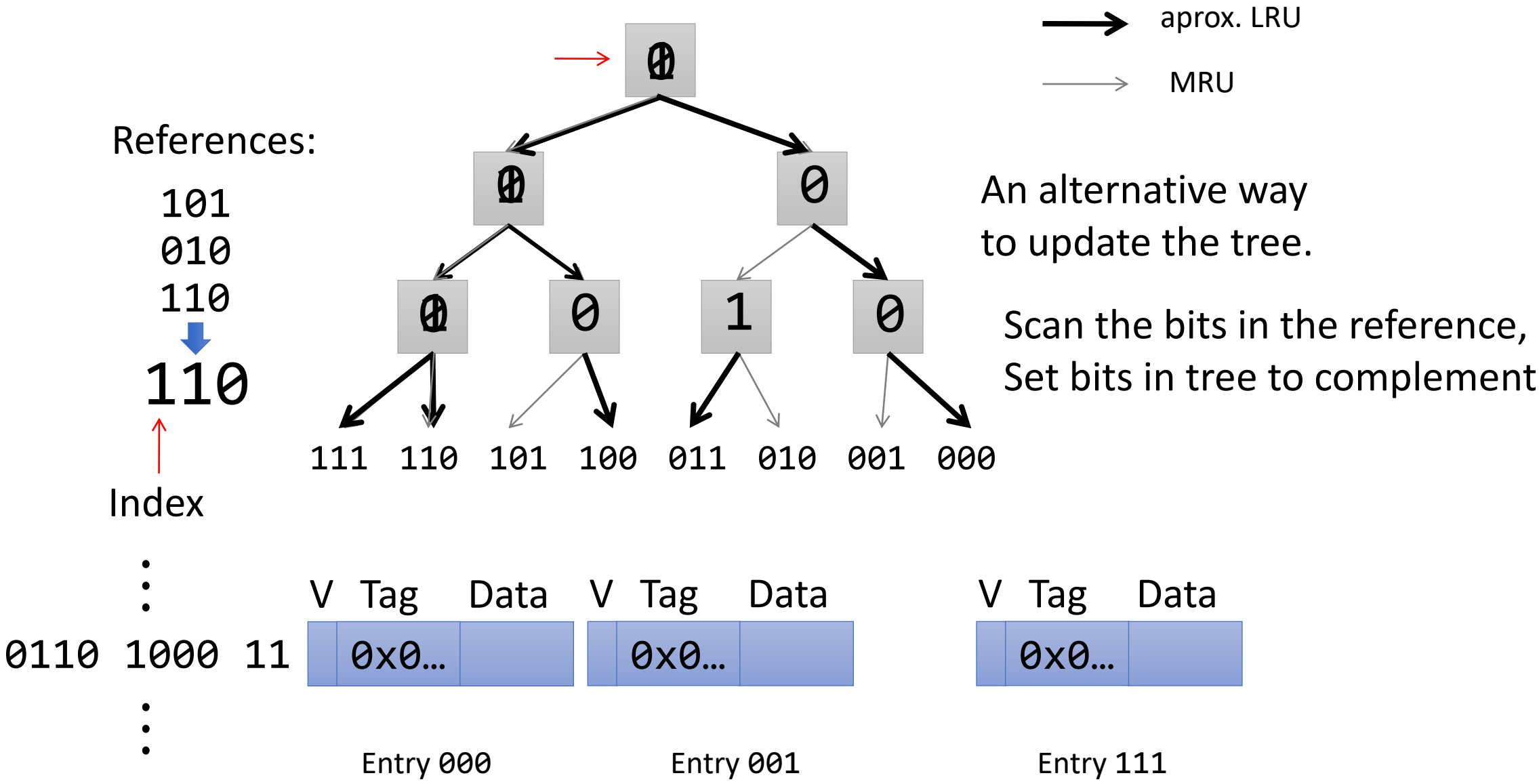
101
010
110



Index



LRU Approximation



Assignment

- startCache
  - Arguments:
    - `a0` = the associativity of the cache
  - Return Values: None
  - Execution:
    - Initialize cache set with all nodes equal `0`.

Assignment (cont.)

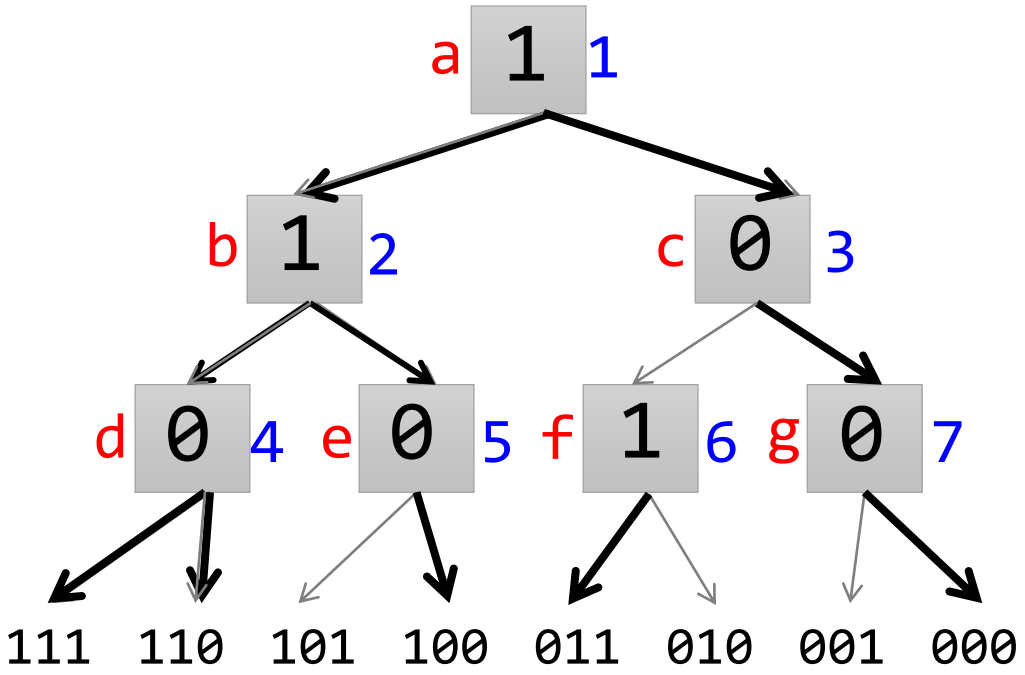
- getLRU
  - Arguments:
    - `a0` = Pointer to a stream of bits
    - `a1` = the number of references to be read from stream
  - Return Values:
    - `a0` = identifies the approximate LRU entry. For example, if the entry `10011` is the LRU approximation, then return:
      - `a0 = 0000 0000 0000 0000 0000 0000 0001 0011`

Example

- `a0 = 0x10010044, a1 = 7`

Address	Value
0x 1001 0044	0101 0101 1011 0010 0111 0110 1110 0000
0x 1001 0048	0000 0000 0000 0000 0000 0000 1111 1101

Storing a binary tree in memory



left child = 2 × parent

right child = 2 × parent + 1

$$\text{parent} = \left\lfloor \frac{\text{child}}{2} \right\rfloor$$

Address	Value hgfe dcba
0x 1000 1000	0000 0000 0000 0000 0000 0000 0010 0011
0x 1000 1004	0000 0000 0000 0000 0000 0000 0000 0000