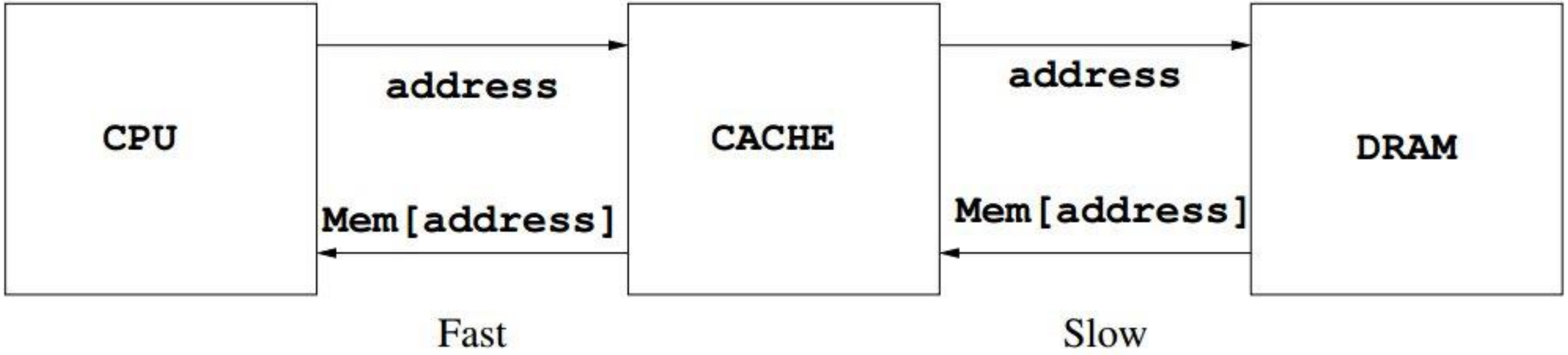
Tutorial 9 : cache memory

Why use a cache ?

- **Main memory** (VRAM/DRAM) **is slow** !
- To deal with this, the β-machine speed is reduced to match the memory read and write speed
- To make the machine faster, one can use a intermediate smaller and faster memory between the processor and the main memory: **a cache**.
- The cache associates memory addresses with their values (taken from the main memory)



Basic working principle

- Reading a value from memory in presence of a cache is simple:
    1. Check whether the cache memory contains the address
    2. If it does, read the associated value from the cache
    3. Otherwise, save the value in the cache and return it

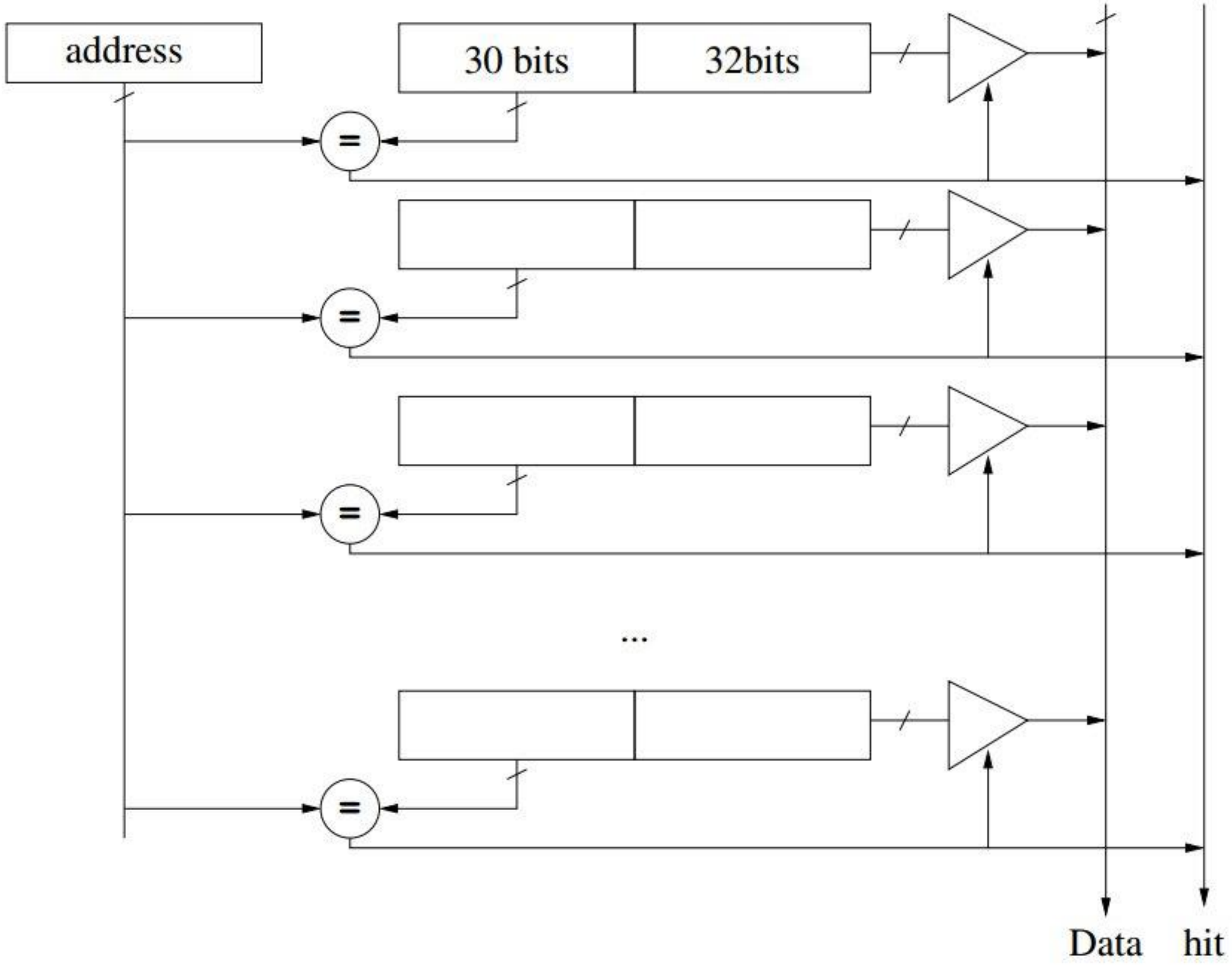
- This usually works because memory accesses are not random. They follow the subsequent principles:
    - **Temporal locality principle**
    - **Spatial locality principle**

Cache memory variants

- Totally associative cache
- Totally associative cache in blocks
- Direct mapped cache
- Set associative cache

Totally associative cache

For each memory address *A*, **store its** corresponding **word**. Select a location using a **replacement policy**.



Pros:

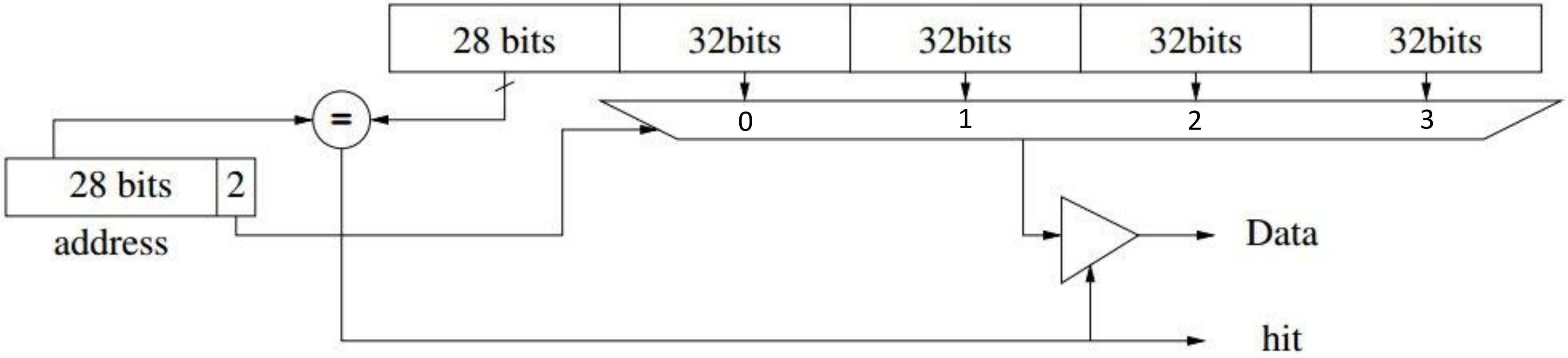
- Simple

Cons:

- One comparator and one address per stored word
- Does not exploit fully the locality principle
- Need for a replacement policy (can be costly to implement)

Associative cache in blocks

For each address *A*, it stores the **N consecutive words** starting with the one stored at *A*.



Pros:

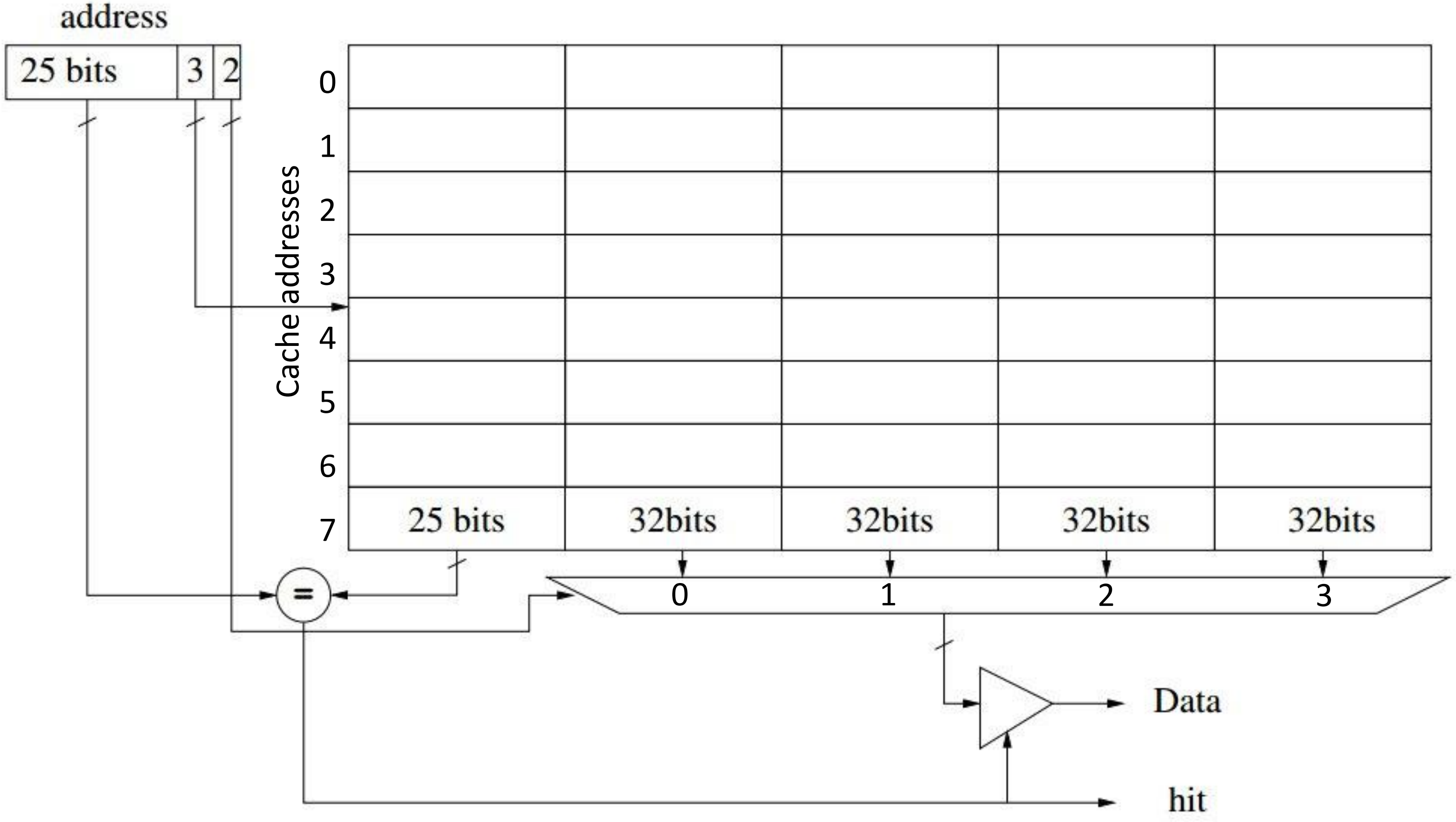
- Exploit the locality principle better
- Better capacity: one comparator for N stored words

Cons:

- Need for a replacement policy which can be costly

Direct mapped cache (in blocks)

Uses a part of the (memory) address as cache address !



Pros:

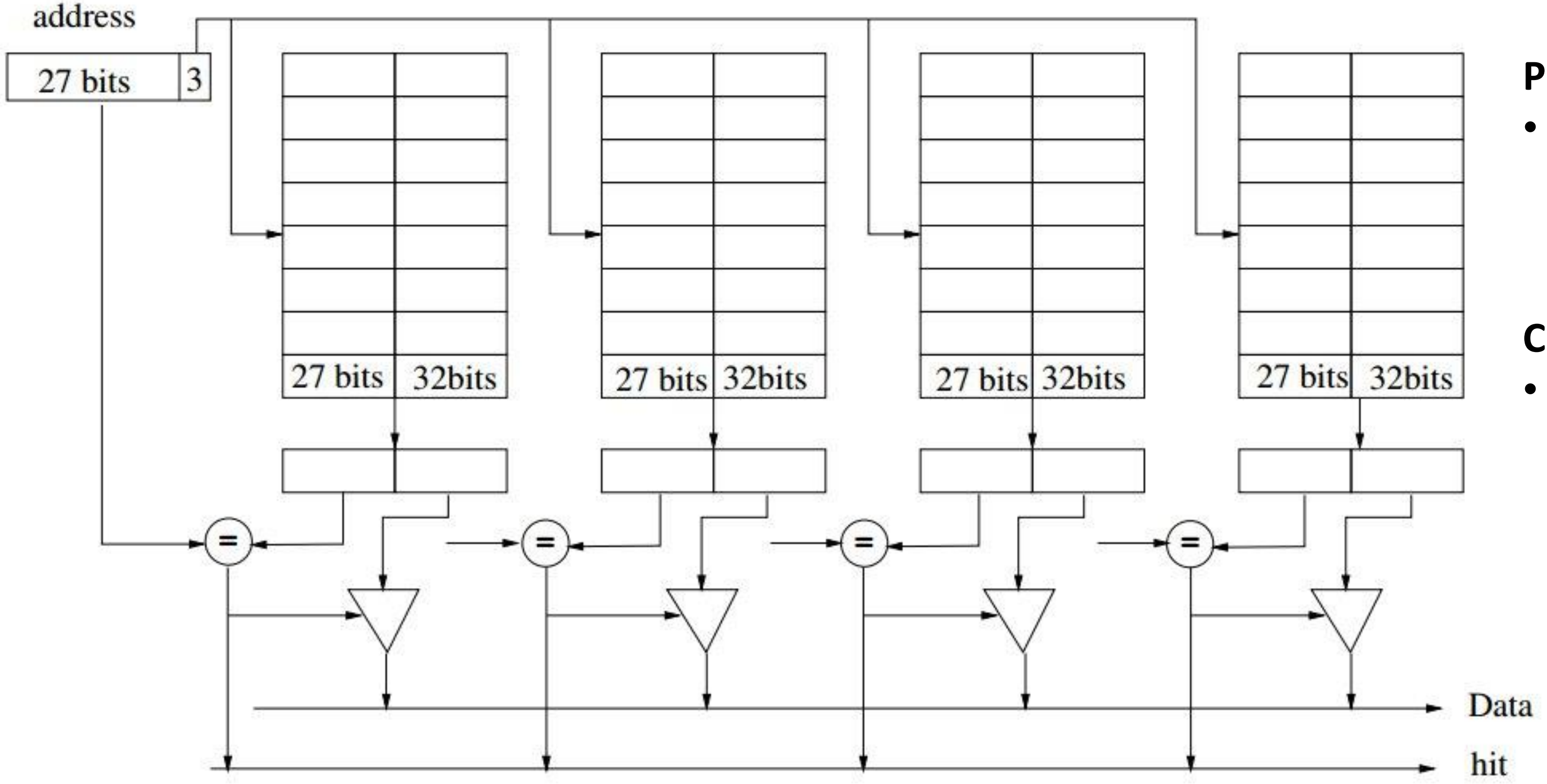
- No need for a replacement policy
- Only one comparator is needed

Cons:

- Not possible to store simultaneously the content of different memory addresses sharing the same cache address

Set associative cache

Compromise between associative cache and direct mapped cache
N direct mapped caches (in blocks or not).
Selection of cache using a replacement policy.



Pros:

- Can store the content of memory addresses having the same cache address

Cons:

- Need for a replacement policy but for large enough cache, random selection yields results almost as good as LRU