How do caches even work?

Key terms:

Temporal Locality:
- data that was recently accessed is likely to be accessed again in the near future.

Spacial Locality:
- Data that is physically close in memory tends to be accessed around the same time.

Cache memory structure

- cache organizes data into LINES, smallest chunk of memory a cache can transfer

TAG | DATA

Lines are 32/64*/128 bytes

- data has tags with a number representing where the data came from

- then, its divided into SETS. Each SET has multiple cache lines

etc...

n-way set associative cache

PROJECT 3: CACHE

7月11日

Tag	set index	offset

cache line

- offset bits are $\log_2$(block size)
- set index is $\log_2$(# sets)
- tag bits are whatever's left
  tag bits = address width − offset_bits − index bits

1 or 64 — n-way

- Cache is a 2D array
- Cacheline cache[numsets][numways]

(total cache size)★ = (#sets)★ · (# ways) · block size★

or the other way, since we know cache size at execution

$$\#\text{sets} = \frac{\text{(total cache size)}}{\text{(associativity} \cdot \text{block size)}}$$

instructions are fixed at 8 bytes.

num lines = 64
block size = 16 bytes

$$\text{ways} = \frac{\text{total lines}}{\#\text{sets}}$$ → we know sets now and lines

no fractional ways, check for that

ways = $\frac{64}{64 \text{ sets}}$ → direct mapped, 1 way

ways = $\frac{64}{1 \text{ set}}$ → fully associative, 64 ways

Addr width — 32 bits

offset bits: $\log_2(16)$ = 4 bits

set index: $\log_2$(# sets) → will be known at runtime

tag bits = $32 - (4 + \log_2(\#\text{sets}))$

so direct mapped:

64 cache lines
16 byte blocks

64 "slots"

cache[0][0] → { valid, tag(22b), data[0...15] }
⋮
cache[63] → { valid, tag(22b), data[0,...15] }

each slot has:

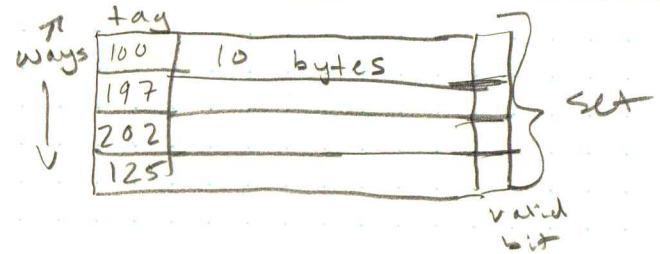
Valid bit	1 bit
tag	22 bits
data block	128 bits (16 bytes)

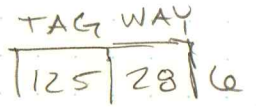
addresses are split as

[31...10] [9...4] [3...0]

tag	index	offset
(22b)	(6b)	(4b)

offset picks 1 of 16 blocks
index picks which way
tag must match





TAG → Is my line the one you want?
INDEX → Which row of the 2D array do we look in?
WAY → Which column in that row has the winner?
OFFSET → Which Byte/Word inside the 16-byte line?

Important to note: The cache is a 2D array of Cache Lines, where the Size of the total cache is

↳ [Number of sets] [Associativity]

---

keeping track of LRU (counter, update when last used)