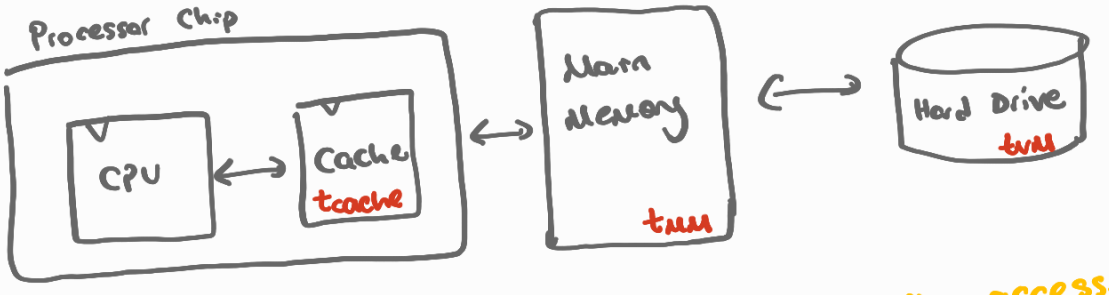
MEMORY SYSTEMS & ORGANIZATION

From Ece Gjran Schmidt's Notes

Overview

In reality, memory is not as fast as the processor.



Locality → Temporal → keep recently accessed data in higher levels of memory hierarchy

Locality → Spatial → when access data, bring nearby data into higher levels of memory hierarchy too

$$\text{Miss Rate} = \frac{\#\text{ of Misses}}{\#\text{ of total memory accesses}}$$

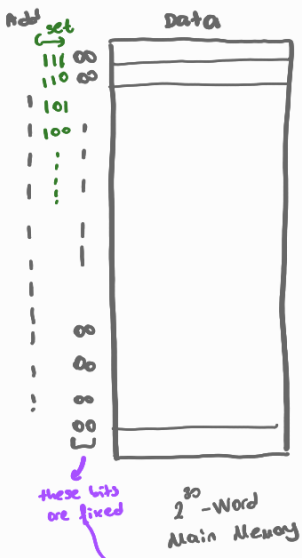
$$\text{Hit Rate} = 1 - \text{Miss Rate}$$

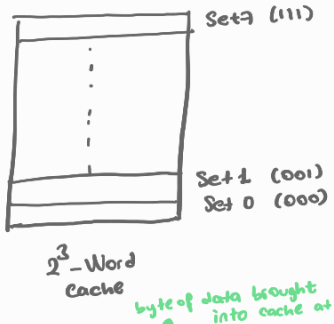
Memory Performance

$$\text{Average memory access time (AMAT)} = t_{cache} + MR_{cache}(t_{MM} + MR_{MM}\, t_{VM})$$

Cache

1) Direct mapped cache ($N=1$, $S=B$) (set, # of blocks)





byte of data brought into cache at once

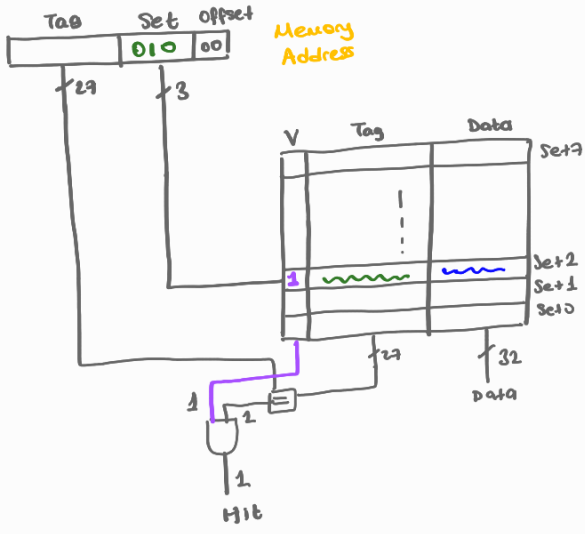
$C = 8$ words, $b = 1$ words

$S = B = 8$ where $B = C/b = 8$, $S = B/N$ (N=1)

$\log_2 S = 3$ bits for set

Tag = $32 - (3+2) = 27$ bits

Hardware

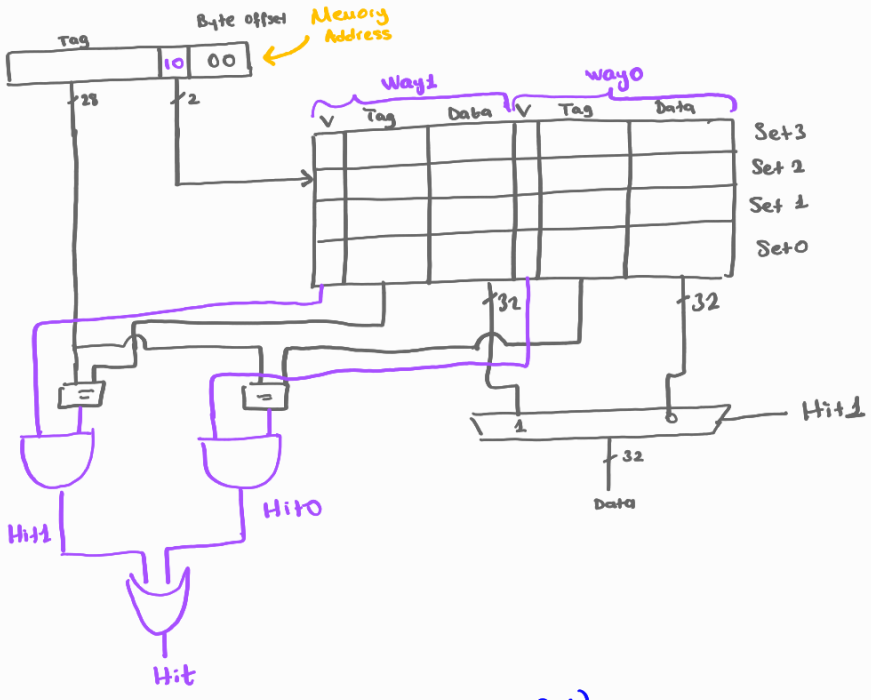


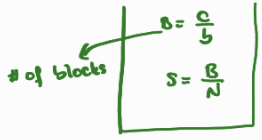
* If "No Match!" occurs → Cache Miss → Access Main Memory

* Conflicts and evicts may occur

2) Associative caches N>1

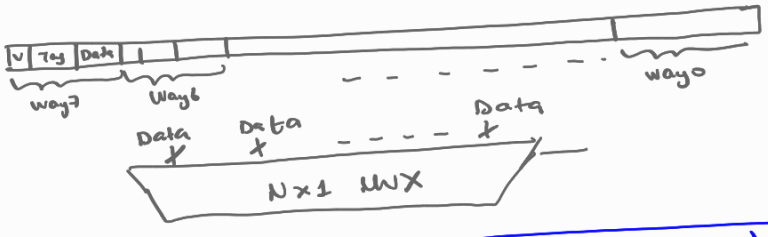
2.1) N-way set associative cache : 1<N<B , S= B/N





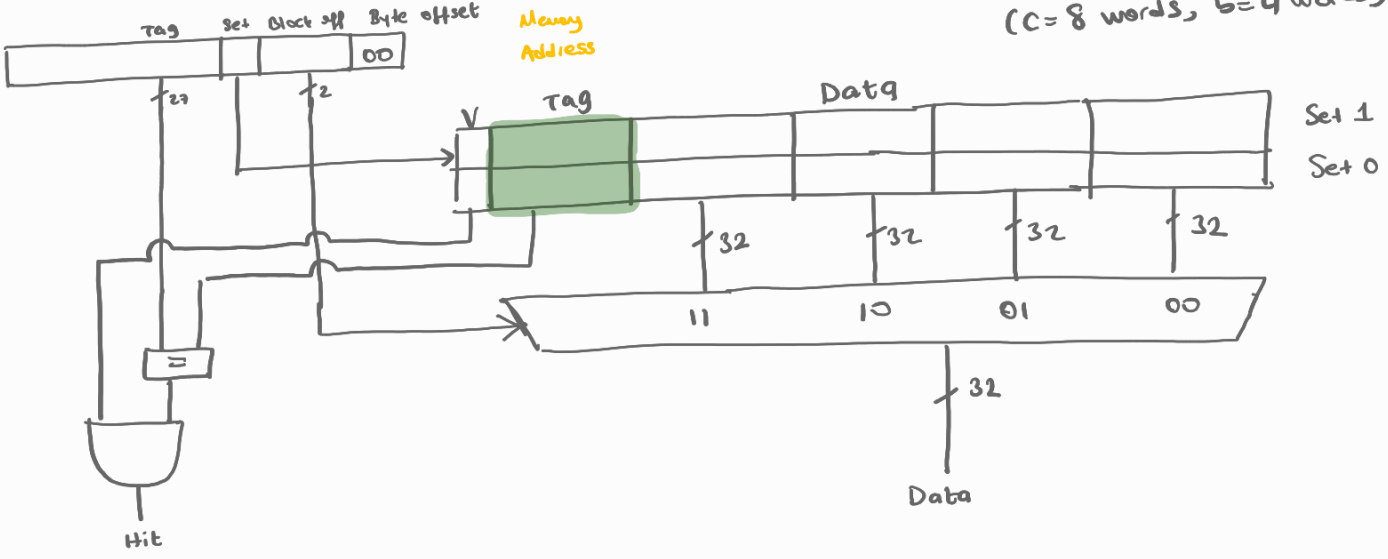
C = 8 (capacity), S = 4 (# set), N = 2 (degree of associativity (W))

2.2) Fully associative cache (N=B, S=1)



Increase block size (more than one word / block)

Direct Mapped Cache
(C = 8 words, b = 4 words, B = 8/4 = 2)



Selecting the Data to Replace

- Set associative cache, N=2, LRU:

  Each time one of the way is used, U is adjusted to indicate the other way.

  * Use additional U bit for it.

Example

W = 2 Byte (words) — S = ?

C = 64 Bytes

b = 2 words

A = 16 bits long

N = 2

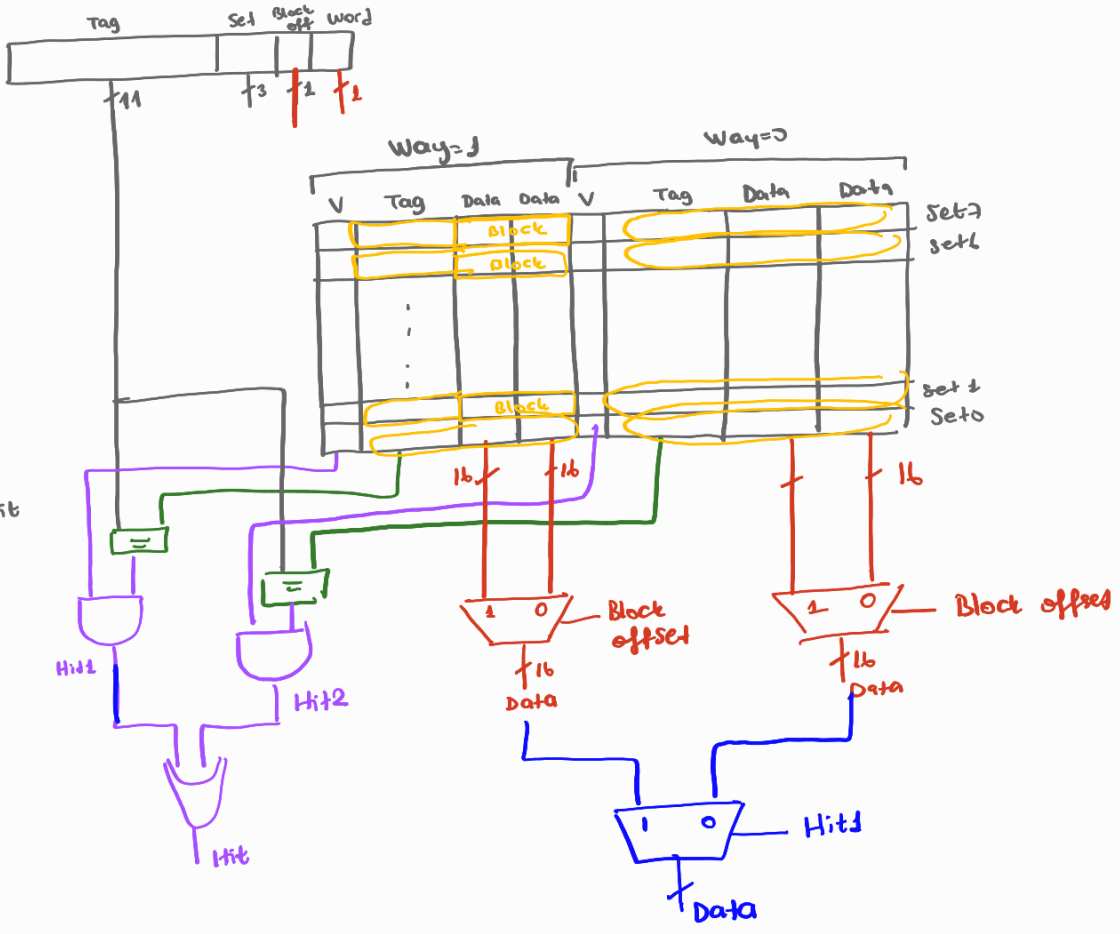
$\frac{64\ Byte}{2\ Byte} = 2^5$ word Cache

$B = \frac{C}{b} = \frac{32\ word}{2\ word} = 16$

$S = \frac{B}{N} = \frac{16}{2} = 8 \rightarrow \log_2 8 = 3 bit$

3+2=5

16-5= 11 (Tag bits)



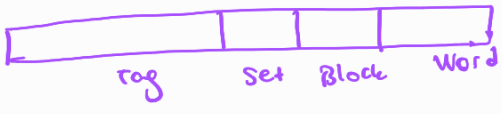
Review

→ Capacity = # words in the cache

→ Cache block = cache line = fixed size unit of data transfered in one shot

→ Cap = C = $\underbrace{S \times N}_{B} \times b \times W$

A bits (Memory) = # tag bits + # bits for set + # bits for block + word offset



$C = S \times N \times b \times W$

$64 = S \times 2 \times 2 \times 2 \Rightarrow S = 8 \Rightarrow \boxed{\log_2 S = 3}$

$\#\ tag\ bits = \underbrace{16}_{\text{memory address}} - (\underbrace{\log_2 8}_{\text{set}} + \underbrace{\log_2 2}_{\text{word}} + \underbrace{\log_2 2}_{\text{block}}) = 11\ bits$

$\#\ of\ blocks = \frac{C}{b} = \frac{64\ bytes}{\underbrace{2\ words}_{2\ byte}} = 16\ blocks$

Each block (Tag + Data) = 16 × (11 bits (tag) + 32 bits (data)) = 16 × 43 //

Example

- Given C = 16 words
- Construct the cache and map the following addresses : 7C, 9C in :

a- Direct Mapped, b = 1 (N=1)

b- 2 way, b = 1 (N=2)

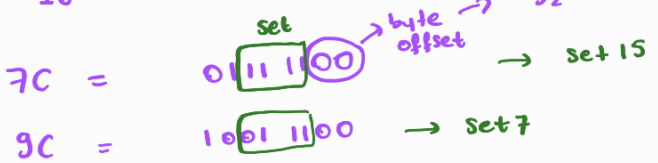
c- Direct Mapped, b = 2 (N=1)

Then Map 78, 74, 70, 20, 1C, 98, ...

a) $C = S \times N \times b \times W$ (N = 1, b = 1, W = 1 word) → S = 16 → 4 bits to encode

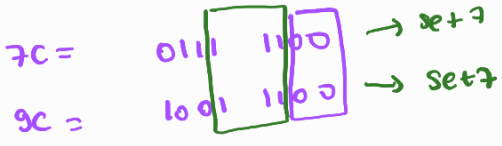
$C = 16$, $S \times N = 16$

(word = 4 byte)



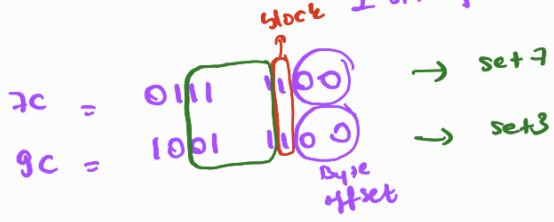
b) 2 way b = 1

$C = S \times N \times b \times W$ (N = 2, b = 1, W = 1) → S = 8 → $\log_2 8 = 3$ bits



c) $C = S \times N \times b \times W$ (N = 1, b = 2, W = 1) → S = 8 → 3 bits for set

1 bit for block



Cache Misses

→ Compulsory Miss: The first request to a cache block

→ Capacity Miss: The cache is too small to hold all concurrently used data

→ Conflict Miss:

Cache Writes

Write-back Cache Writes

Dirty bit (D)

Write-through Caches

No dirty bit

Handling Write Misses:

→ Write Allocate

→ No-Write Allocate

No-write allocate → no update in cache

STR R1, #100 → Not in cache, write Miss

STR R2, #100 → Still not in cache, write miss

LDR R3, #200 → Not in cache, read Miss, loaded to cache

STR R1, #200 → In cache, Hit

STR R2, #100 → Not in cache, write Miss

Write-Allocate

STR R1, #100 → Not in cache, write Miss, written in cache

STR R2, #100 → In cache, hit

LDR R3, #200 → Not in cache, read Miss, loaded in cache

STR R1, #200 → In cache, hit

STR R2, #100 → In cache, hit