14/04/2025.

→ Appendix B

* MEMORY HIERARCHY:

NOTE: please refer to previous lectures.

CACHE: → (frequently accessed memory)

* located b/w CPU & main memory

CPU → cache → main memory

↳ contains internal registers, but their size is small.

L1 L2 L3 divisions of cache

L1 L2: attached with CPU (no distance)

L3 → resides in motherboard

→ to synchronize i.e combine the speed of CPU & L1/L2 cache.

* in order to avoid the delay, CPU doesn't accesses the main memory directly.

CPU ~~consults~~ → Cache

↳ consults cache first.

if data is not found, then it accesses the main memory.

CPU — cache — main memory

* After retrieving the data from main memory, a copy of the data is also saved in the cache before processing the data.

Algorithms: spatial & temporal locality.

Case study:

if data is of 100MB & cache is of 50MB, then how to copy data?

CPU → cache → main mem.

cache → MB

main mem. → GB

Sol: several algorithms are used.

SHAHEER PAPER PRODUCTS

CamScanner

Dated:

CACHE PERFORMANCE:

CACHE HIT: If required word is found in cache.

CACHE MISS: If required word is not found in cache.

HIT RATIO: "(h)" $\frac{\text{Total no. of hits}}{\text{Hits + Miss}}$

"(1-h)" MISS ~~MISS~~ RATIO: $\frac{\text{Total no. of Miss}}{\text{Hits + Miss}}$

Total no. of CPU references

hit = pass, miss = fail.

(in sec) CACHE ACCESS TIME: Time required to access word from cache.

MISS PENALTY: If cache is missed by CPU, it is miss penalty. Also, it is time required to ~~fetch block~~ fetch blocks from main memory.

T_M = cache access time + main memory access

* CPU accesses the data from cache in the form of word.
* Data copied from main memory into cache is called fetch block.

Average Access Time of CPU:

= Hit ratio × CAT + MR × MP

$$= h \times T_c + (1-h) \times T_M$$

21/04/2025

CACHE MAPPING:

There are some methods.

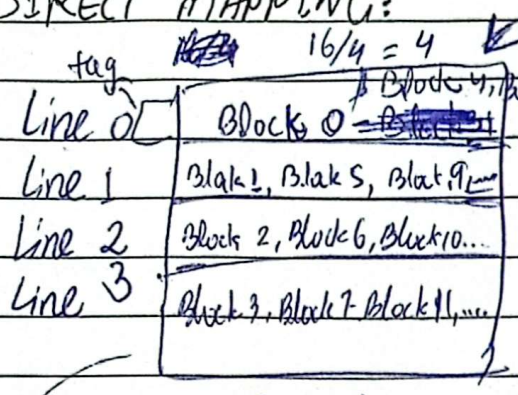
1. Direct mapping
2. Fully associative
3. K-way fully associative

$\frac{128}{4} = 32$ blocks

Dated:

DIRECT MAPPING:

$16/4 = 4$



16 word

Fig 2 Cache memory

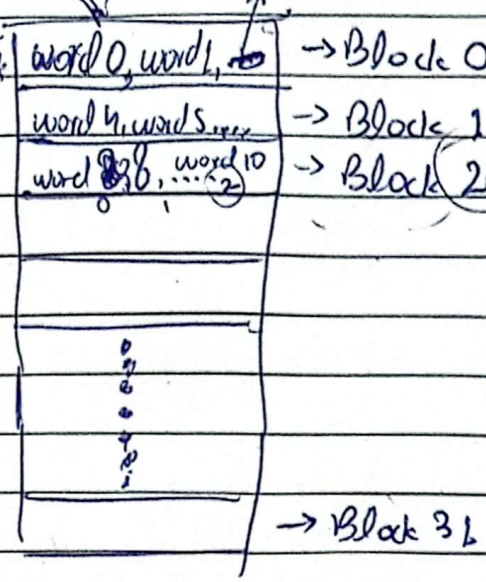


Fig 1 128 word Main memory

~~Line size = Block size~~

↓

words

Formula: $\boxed{k \bmod n}$

where, k = Block number

n = line number. (total lines)

example

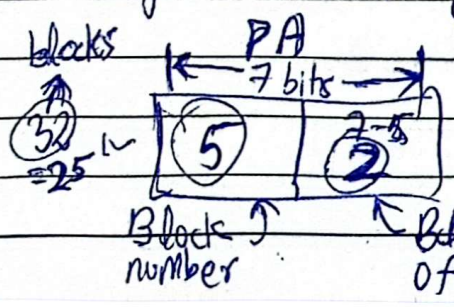
Block no		Total lines	Line no.
0	mod	4	= 0
1	mod	4	= 1
2	mod	4	= 2
3	mod	4	= 3
4	mod	4	= 0
5	mod	4	= 1

... and so on.

Example:

how does a CPU generate address of a required data?

range: 0 – 127 (for 128 word) = 2^7 (to represent the value in bits)



Dated:

For example,

CPU generates

000 10	10
2	2
↳ Block number	↳ Block offset.

word 10 from fig 1

In order to map data from Main memory to cache, consider PA again as

PA (7 bits)

3	2	2
↳ Tag (associated with particular line)	↳ Line number (2^2)	↳ Block offset.

Example:

Tag = 010

Tag	010	00	
	↓ any value.		↳ all combinations.

010	00	00	=	32
010	00	01	=	33
010	00	10	=	34
010	00	11	=	35

word #32

Block 8 is in Line 0

Fig 2

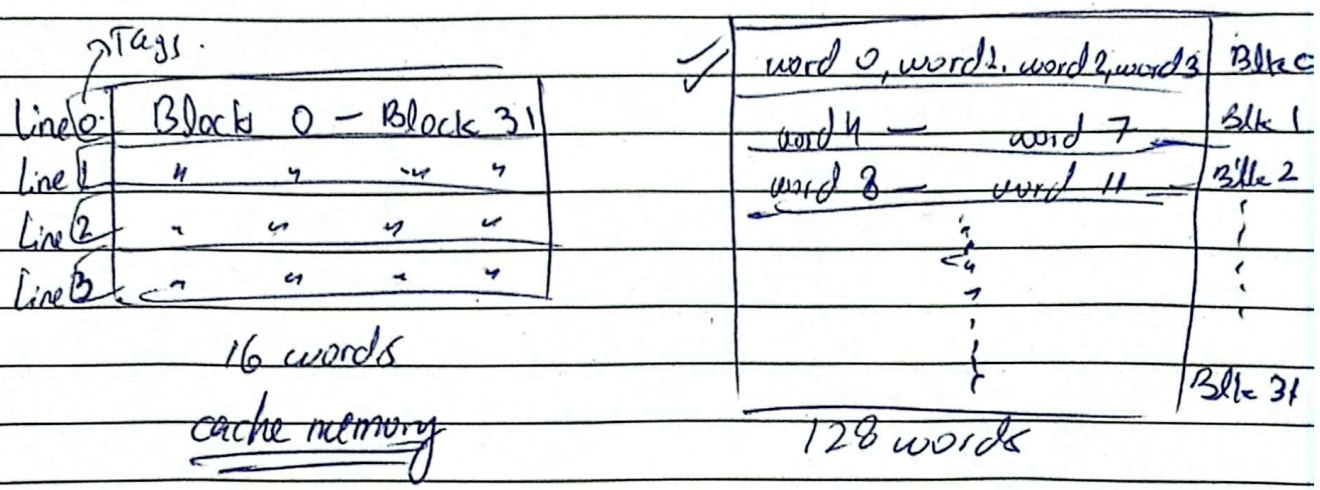
W0 W1 W2 W3	
	0 → Block no.
	1
	2
	⋮
W35	
W32, W33, W34, W35	8

Dated: 24/04/2025

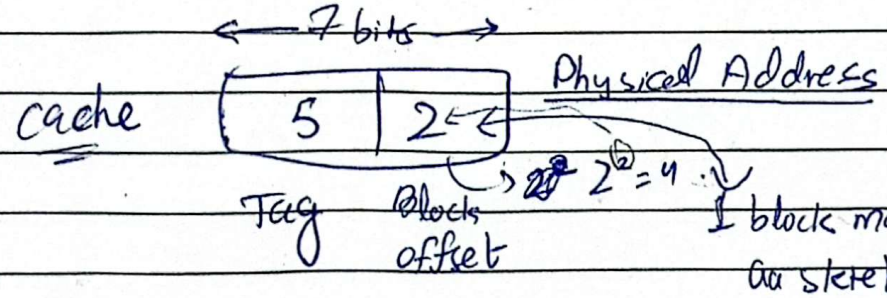
② FULLY ASSOCIATING MAPPING:

METHOD:

① Transferring data from main memory to cache.
② Physical address of both memories are same because of associativity regarding blocks.
③ ~~Complete~~ Calculate the repetitive bits considering given conditions.



Tags are associated with every line of cache it is expressed as indexing of cache.



Dated:

Example: Consider a request generated by CPU containing tag bits = 00100, calculate which block number accomodates in Line 0.

PA

5	2
Tag	offset

	All combo of offset	
00100	00	= 16
00100	01	= 17
00100	10	= 18
00100	11	= 19

} words → refer to figure

⇒ Block 4 accomodates in line 0 of cache.

(3) K-WAY SET ASSOCIATIVE MAPPING:

METHOD:

1. K is given (in 2's power).
2. Calculate the set value.
3. To decide which block lies in cache using direct mapping.
4. To decide which block sustain in cache, use fully associative mapping.
5. PA formats are different for main & cache.

Consider: k = 2.

∵ set = $\frac{\text{no. of lines in cache}}{K}$

PA

tag	set no.	Block offset
4	1	2
	$2^1 = 2$	

Set = $\frac{4}{2}$ = 2

cache

Line 0	
L1	
L2	
L3	

} S_0

} S_1

0 & 2.

divide

cont...

Contd...

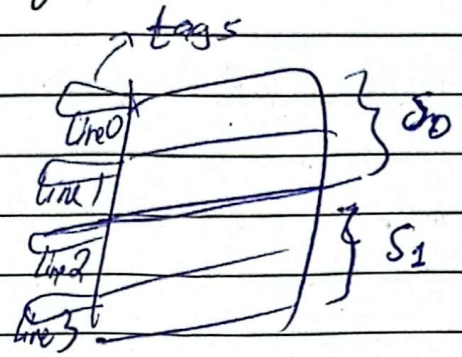
Dated: 24/04/2025

Cache Mapping:

25/04/2025

(3) ~~Fully~~ k-way set associative mapping.

$k \bmod n$ — k: block no., n: no. of (sets)



$0 \bmod 2 = 0$ → set no.

$1 \bmod 2 = 1$

$2 \bmod 2 = 0$

$3 \bmod 2 = 1$

differences ??? in all cache mapping.

Example: A system incorporates 4-way set associative memory with a volume of 16KB & using a block size of 8 words. Considering the word length of 32 bits, calculate the no. of tag bits if the size of physical address space is 4GB.

← 32 bits →

20	7	5
Tag	Set	Block offset (total no. of words)

$32-7-5=$

$2^5 = 32$

$4GB = 2^{32}$, $2^2 \cdot 2^{30}$

Words	
1 2 3 4 5 6 7 8	Block

$* \frac{32 \text{ bits}}{8} = 4$ byte

$\boxed{\text{set} = \frac{2^9}{4} = 2^7}$ (2^2)

∴ block size = line size

$8 \times 4 = 32$ bytes (size of 1 block)

SW SHAHEER PAPER PRODUCTS

CamScanner

Dated:

Size of cache: $\rightarrow \frac{16\,KB}{32\,B} = \frac{2^4 \cdot 2^{10}}{2^5} = 2^9$

lines in cache

CACHE REPLACEMENT POLICY:

Algorithm

1. LRU (Least Recently Used)
2. FIFO

Example: Considering a system follow fully associative memory with (0-7) lines & has following sequence:
4, 3, 25, 8, 19, 6, 25, 8, 16, 35, 45, 22, 8, 3, 16, 25, 7
CPU requests
which cache block will accomodate block 7 if the system follows LRU?

Sol

Diagram is required.

CA

Line	
0	4 45 (LRU)
1	LRU 3 22
2	25
3	8
4	~~19~~ 3
5	~~6~~ 7
6	~~25~~ 16
7	~~8~~ 35

duplicated, do not replace

repeated values

cache full

repeated values

* Block 7 will accomodate Line # 5 of the cache.

Dated: 28/04/2025.

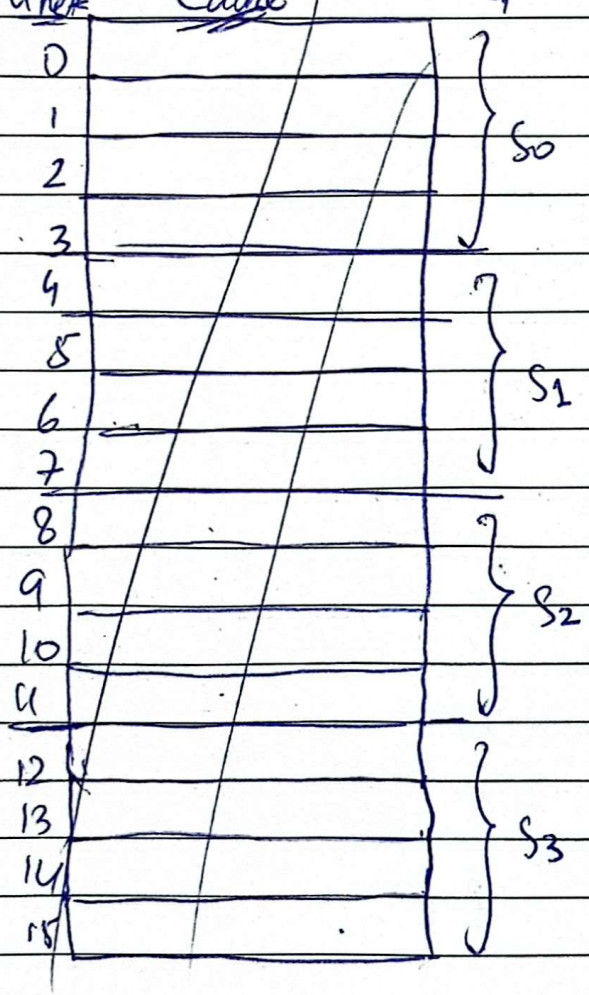
Example: Consider a 4-way set associative Cache with total of 16 cache blocks main memory block requests are:
0, 255, 1, 4, 3, 8, 133, 159, 216, 129, 63, 8, 48, 32, 73, 92, 155

Draw a cache memory consider cache replacement policy.

Sol: k-way set associative

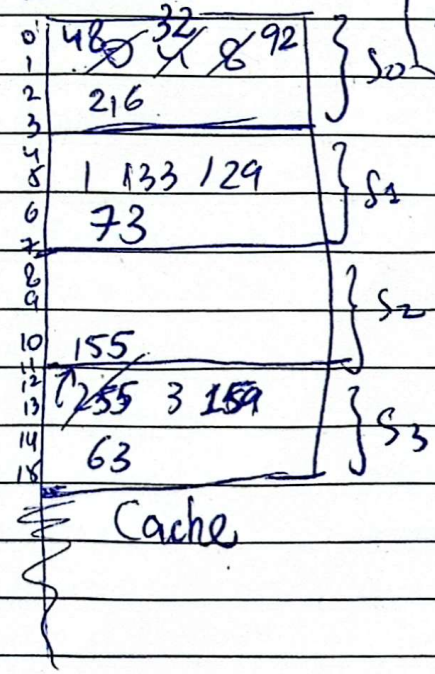
$K = 4$, set → $\frac{\text{no of lines}}{k}$

$= \frac{16}{4} = 4$ (sets)



∵ k mod n.

∴ k mod. 4 → total sets



(in k-way associative, we're not concerned with the line no. of cache.)

Dated:

CACHE: (Topics)

1. Cache performance
2. " Mapping
3. " Replacement Policy
4. " Optimization → (Appendix B.3)
   (Software based)

optimization:
Acc to someone's requirement, modifying the system.
i.e A person is concerned with the high security and can compromise on time, so design the system a/c to this.

[in cache: frequently used data is placed]

CATEGORIES OF MISSES:

1. Compulsory: (Cold-start misses / first-reference misses).
   A miss is compulsory.