CSA (Computer System and Architecture).

Associative Memory

* A memory unit accessed by content is called an associative memory or Content Addressable Memory (CAM).

* Associative memory is accessed simultaneously and is parallel on the basis of data content rather by specific address or location.

* When a word is written in an associative memory, no address is given.

* This memory is capable of finding an empty unused location to store the word.

* When a word is to be "read" from an associative memory, the content of word is specified or part of word is specified.

* The memory locates all words which match the specified content and marks them for reading.

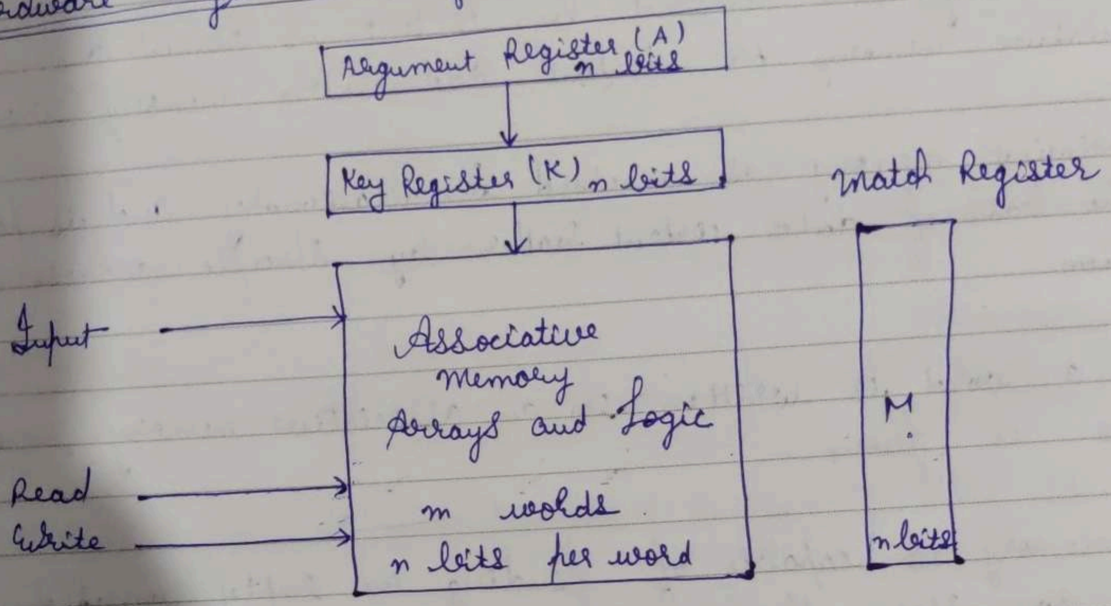
* Because of it's organisation, the associative memory is uniquely suited to do parallel searches by data association.

* An associative memory is more expensive than Random Access Memory because ~~much it~~ must have storage capacity as well as logic ~~ctt~~ circuits for matching it's content with an external argument.

* That's why, associative memory is useful in application: where search time is very critical and must be very short.

Date

Hardware Organisation of Associative Memory :-



Block diagram of Associative Memory hardware organisation.

* Argument Register (A) :- It contains words to be searched.

* Key register (K) :- It provides a mask for chosing a particular field / key in argument word. or It specifies which part of the argument word needs to be compared with words in memory.
If all bits in key's register ~~are~~ having 1's ~~are~~ in their corresponding position are compared.

* Associative memory array :- It contains the words that are to be compared with the argument word in parallel.
It consists of m words with n bits per word.

Date

* **match logic (M) :-** It has m bits, one bit corresponding to each word in the memory array.
After the matching ~~processes~~, the bits corresponding to matching words in ~~no match~~ match register are set to 1.

* Reading is ~~accessed~~ accomplished by Sequential access in memory for those words match bits are set (or 1).

e.g. :-

A	101	111100
K	111	000000
	unmask	masked

word 1	100	111100	$\boxed{0}$
word 2	101	000001	$\boxed{1}$
word 3	101	111100	$\boxed{1}$
word 4	110	001010	$\boxed{0}$
word 5	111	000100	$\boxed{0}$

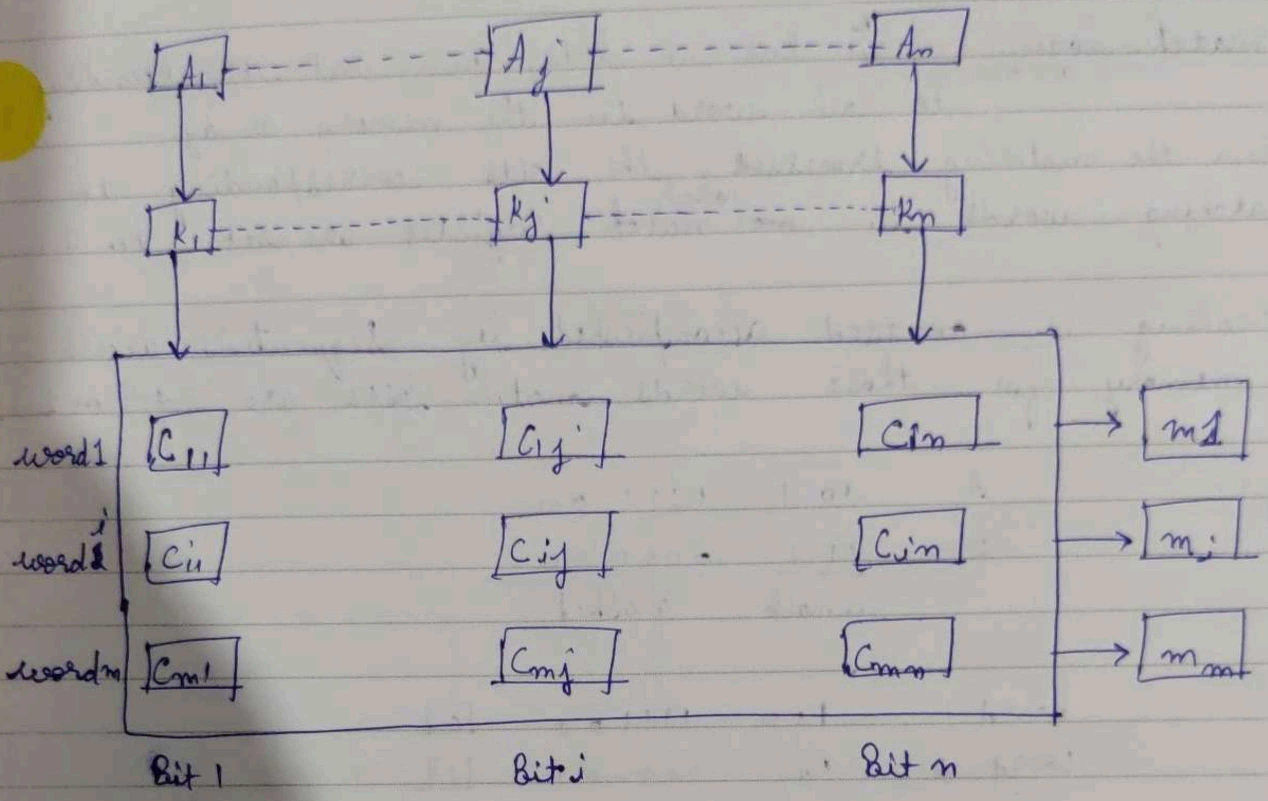
⇑

0 → no match
1 → match.

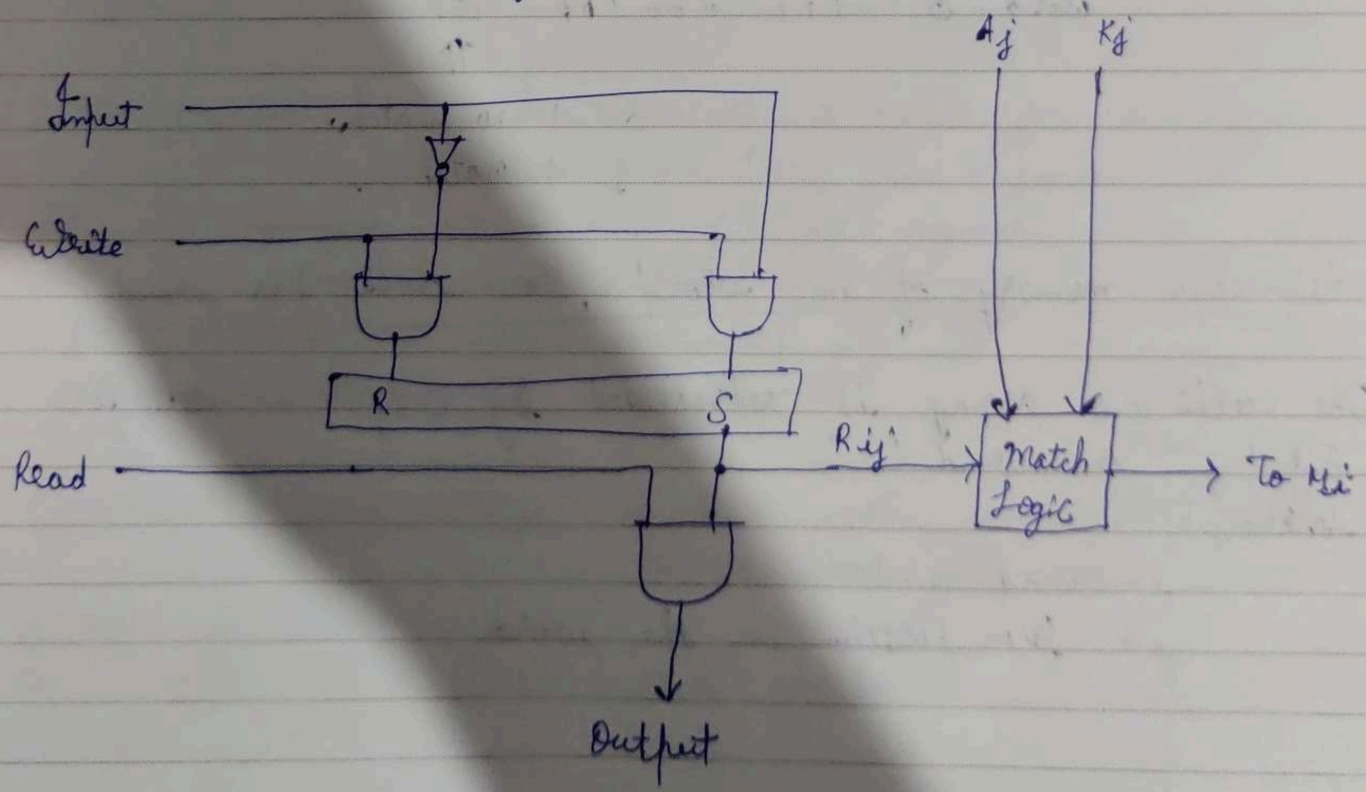
Associative memory of m word, n cells per word :-

The cells in array is represented by C_{ij} : a cell for bit "j" in word "i".
where,
i = word number
j = bit position in the word.

Date



Internal Organisation of cell C_{ij} :—



Date

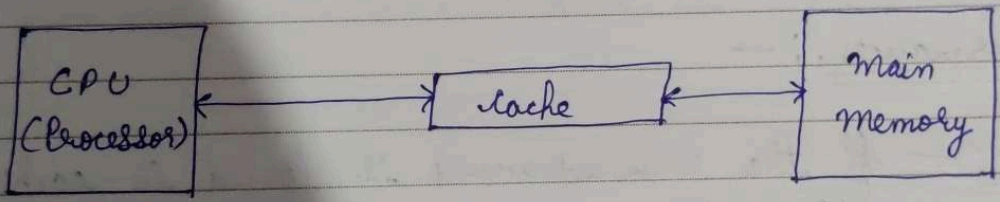
* It consists of flip-flop storage element Fij & circuits for reading, writing & matching the cell.

* The input bit is transferred into storage cell during a write operation.

* The bit stored is read out using a read operation.

* The match logic compares the content of the storage cell with the corresponding unmasked bit of the argument & set the bit in Mi.

<u>Cache Memory.</u>



* Small sized fast memory.
* Placed between main memory & CPU.
* High speed volatile memory.
* Contains most frequently accessed instructions & data.
* Located inside the CPU chip or motherboard.
    (internal cache)   (External cache - L3)
    L1 and L2

<u>Working of cache :-</u>

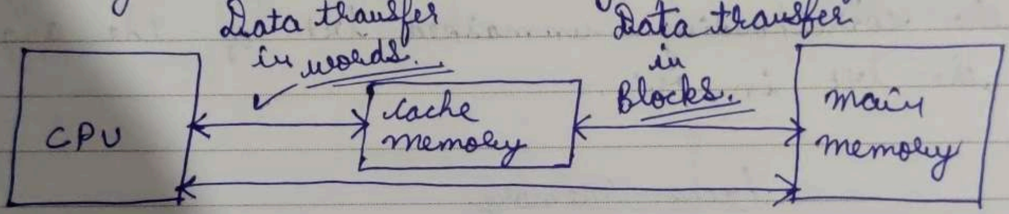
* The CPU initially looks in the cache cache for the data it needs.

Date

* If the data there, it will retrieve it and process it.

* If the data is not there, then the CPU access the system main memory & then puts a copy of the new data in the cache before processing it.

* Next time, if the CPU needs to access the same data again, it will just retrieve the data from the cache instead of going through the whole loading process again.



<u>Cache Performance</u>.:—

<u>Cache hit</u>.:— If the required word is found in cache is called Hit.

$$\text{Hit Relation }(h) = \frac{\text{Hits}}{\text{Hits} + \text{miss}}$$

$$= \frac{\text{no. of hits}}{\text{Total no. of CPU } \sout{\text{performance}} \text{ references}}$$

Cache Access Time.:— Time required to access.
(Cache Hit Time).:— Word from the Cache.

<u>Cache miss</u>.:— If the required word is not found in cache is called cache miss.

Date

$$\text{miss ratio } (1-h) = \frac{\text{miss}}{\text{Hits + miss}}$$

$$= \frac{\text{no. of miss}}{\text{Total no. of CPU references}}$$

miss Penalty (cache miss time penalty) :—
The time required to fetch the required block from main memory.

$$\text{Average access time of CPU} = \text{HitRatio} \times \text{Cache access Time} + (1-\text{Hit Ratio}) \times \underline{\text{miss Penalty}}$$

cache + main memory access time

$$\boxed{\text{Average access time of CPU} = h \times T_c + (1-h) \times T_m}$$

Cache Mapping.

* Cache Mapping is a technique by which content of main memory is brought into the cache memory.

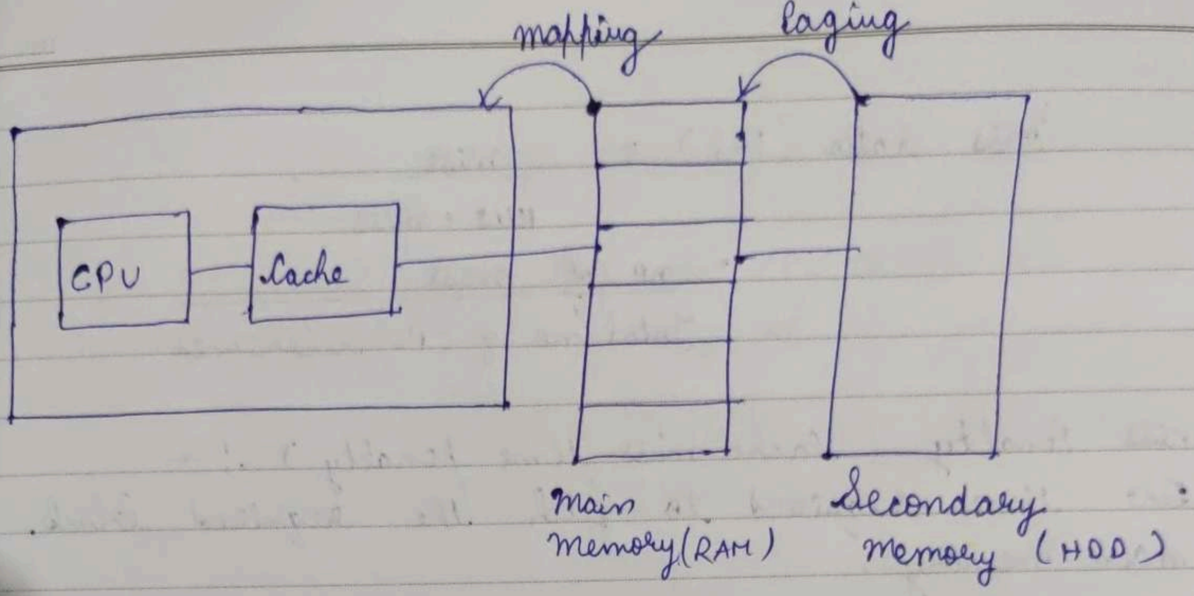
* It is a transformation of data from main memory to cache memory.

* 3 types of cache mapping are the following :—

1. Associative mapping.
2. Direct mapping.
3. Set- Associative mapping.

Shipra

Date



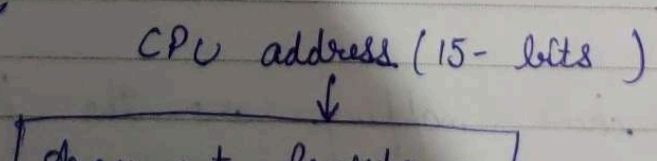
1. <u>Associative mapping</u> :-

* Fastest & most flexible cache organisation ~~&~~ uses associative memory.

* In associative mapping, caches are made up of associative memory. Associative memory is used to store both the address and content data of the memory word.

* It permits any location in cache to store any word from main memory, i.e. it enables any word from memory at any place in the cache memory (which does not happen in other mappings).

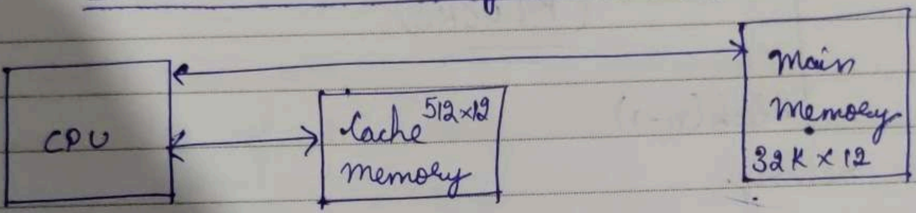
~~* Fastest & most~~



Date

← address (15 bits) →	← Data (12 bits) →
01000	3456
02777	6710
22345	1234

Associative mapping cache (All numbers are in octal)



32K words.
$2^5 \times 2^{10} = 2^{15}$ words.
CPU address = 15 bits.
Data = 12 bits.

* To replace any data pair, it uses a replacement policy,
  * FIFO.

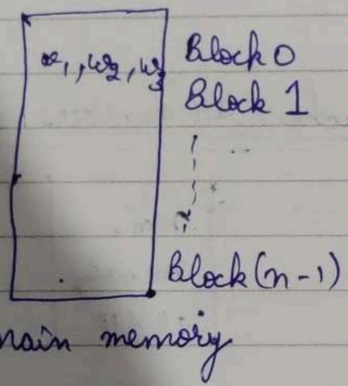
Direct mapping.

* Associative memories are expensive compared to Random Access memories (because of added matching logic attached with each cell)

* For random - access memory, direct mapping can be used.

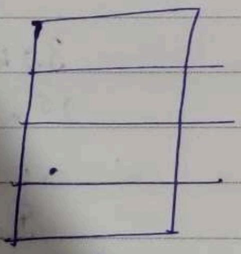
Date

* Simplest technique :— Direct mapping ∃ it maps each block of main memory into one possible cache line.
or it assigns each memory block to a specific line in the cache.



(BLOCKS)

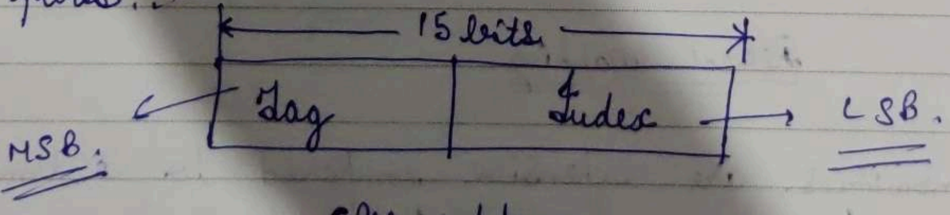
1 block may contain 1 word to 16 words.



(LINES)

Cache memory

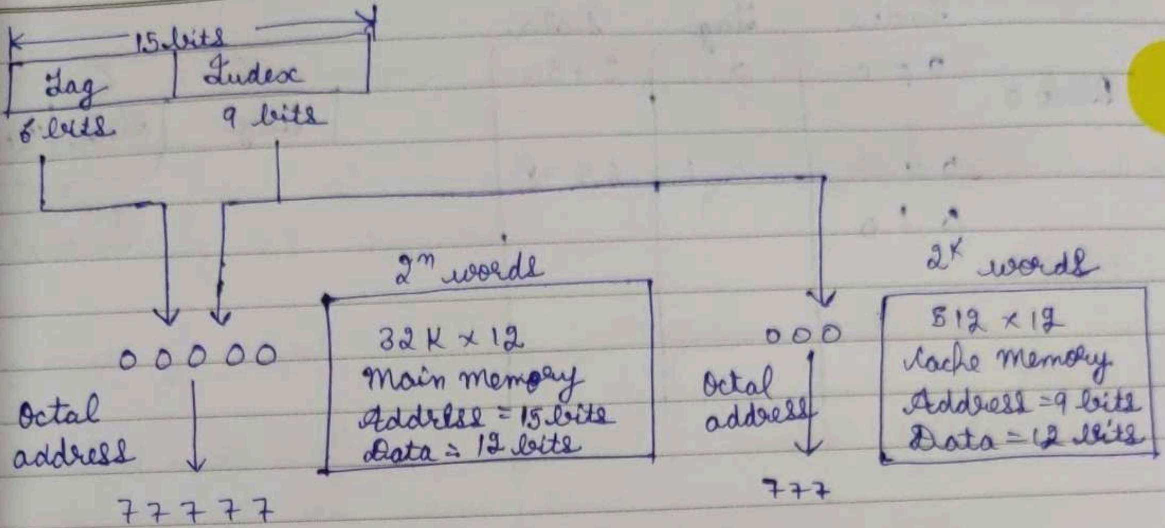
* CPU address as address space is divided into 2 parts :—



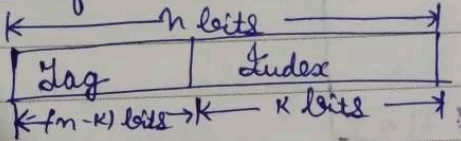
CPU address

Shipra

Date



If, MM (main memory) is of 2^n words & CM (cache memory) is of 2^K words.



* n-bits address is used to access the main memory.
* K-bits index is used to access the cache memory.

(a) main memory (b) cache memory

	memory data
00000	1220
00777	2340
01000	3450
01777	4560
02000	5670
02777	6710

Index	Tag	Data
000	00	1220
777	02	6710

1 Block = 1 word

E.g.:- $\underbrace{00}_{\text{Tag}}\ \underbrace{000}_{\text{Index}}$ hit (Data is in cache)

$\underbrace{01}_{\text{Tag}}\ \underbrace{000}_{\text{Index}}$ ✓ miss

$\underbrace{02}_{\text{Tag}}\ \underbrace{000}_{\text{Index}}$ ✓ miss

Date

	Index	Tag	Data
Block 0	000	01	3450
	007	01	6578
Block 1	010		
	017		
Block63	770	02	
	777	02	6710

$$\boxed{\overset{6}{\text{Tag}}} \boxed{\underbrace{\overset{6}{\text{Block}} \mid \overset{3}{\text{Word}}}_{\text{Index}}}$$

Cache memory = 512 words
Size

If $\boxed{1 \text{ Block} = 8 \text{ words}}$.

Total no. of blocks = $\frac{512}{8}$ = 64 blocks.

= 2^6 blocks,

{ Block = 6 bits.

1 block = 8 words = 2^3 words

{ word = 3 bits,

Date

Set - Associative Mapping.

* Improved form of direct mapping, where drawback of direct mapping is removed.

* Drawback of direct mapping.:— Two words with the same index in their address but with different tag values cannot reside in cache memory at the same time.

e.g.:— 0 1 000
0 2 000

Index	Tag	data
000	~~01~~	~~1250~~
	02	3950

In set-associative mapping, each word of a cache can store two or more words of memory under the same index address, creating a set.

→ Each data word is stored together with it's tag.
→ The number of tag -data items in one word is said to form a set.

Set - Associative memory combines direct mapping & Associative mapping.

Shipra