10) Clock :- It is used to synchronize operations.

11) Reset :- Initializes all modules

* Word is collection of bit. 8086 → 8 bit. 80x86 → 16/32/64
No. of bits at a time can transferred from one

CHAPTER 4 - Cache Memory

4.1 Completer Memory system Overview

A memory is characterised by following parameters

1) Location :- It refers to wether memory is internal to or external to the computer.

* Internal memory include main memory, registers, cache memory, whereas external memory refers to peripheral devices, disc, magnetic tape etc

2) Capacity :- for internal memory the capacity is defined in terms of bytes / words. whereas for external memory the capacity is always expressed in terms of bytes only.

3) Unit of transfer :- for internal memory the unit of transfer is equal to the no. of electrical line into and out of the memory module i.e it deals with the data bus width.

Word → It is a natural unit of organisation, a size of a word is typically equal to the no. of bits used to represent an integer and instruction length.

Addressable unit - It is the smallest location where can be addressed uniquely let 'A' is the length of an address in bits then the no. of addressable unit is given as $N = 2^A$ ——①

ex- 8086 → 20 no. of address line

Addressable unit = 2^{20} = 1MB.

4) Access Method - There are four types :-

(i) Sequencial access → It starts at the beginning and proceeds in sequencial order hence the access time depends on the location of data and previous location. ex- magnetic tape.

(ii) Direct access → Individual blocks have unique address hence here access is given by jumping into the respective block plus the sequencial search. Here also, access time depends on the location & previous location ex 3000 2000 2020

(iii) Random access → Individual access can exactly identify different location. Here the access time is independent of the location or previous access

ex - RAM ex MOV Ax, [3000H]

(iv) Associative access → Here data is located as a portion of the main memory. The access time is independent of the location and the previous location. ex - cache memory

5) Performance -

(i) Access time (Latency) → It is the time b/w presenting the address and getting the valid data

ex - MOV Ax, [5000H] time required to access data.

Imp (ii) Memory cycle time → It refers to the access time + any additional time required before the next access

Imp (iii) Transfer rate → It is the rate at which data can be moved into and out of a memory unit.

for random access, the transfer rate is $\frac{1}{\text{cycle time}}$ ——②

for non-random access, $T_n = T_A + \frac{n}{R}$

where T_n ← Average time to read or write n bits
T_A ← Is the average access time
n ← No. of bits
R ← Is the transfer rate in bits per second (bps)

6) Physical types of memory -
(i) Semiconductor memory (RAM, ROM)
(ii) Optical memory (CD, DVD) Compact disc
↳ faster, write on both side, more storage
(iii) Magnetic memory (magnetic disc, magnetic tape)

7) Physical characteristics -
(i) Volatile -
(ii) Non-volatile -

8) Organisation - It refers to the physical arrangement of bits into words. It is not frequently used.

Memory Hierarchy

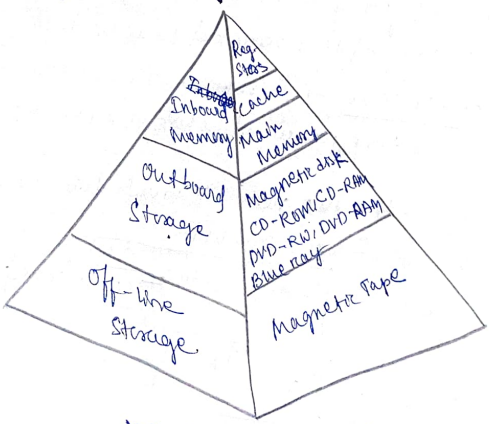


fig:- memory Hierarchy

The following relationship holds w.r.t capacity, time and cost
1) Faster access time → Greater cost per bit
2) Greater capacity → smaller cost per bit
3) Greater capacity → slower access time

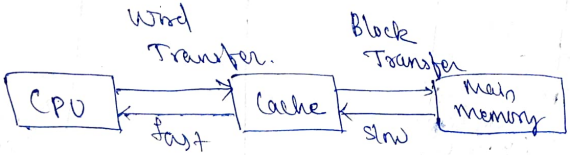
To remove this dilema we are using a memory hierarchy as we move down the hierarchy the following occurs.
1) Decreasing cost per bit.
2) Increasing capacity
3) Increasing access time
4) Decreasing frequency of access of the memory by the processor (this principle is known as locality of reference) nearby to the processor.

Main memory is the principle internal memory system of the computer. Each location in main memory has one unique address. Main memory is usually extended with a higher speed and smaller cache.

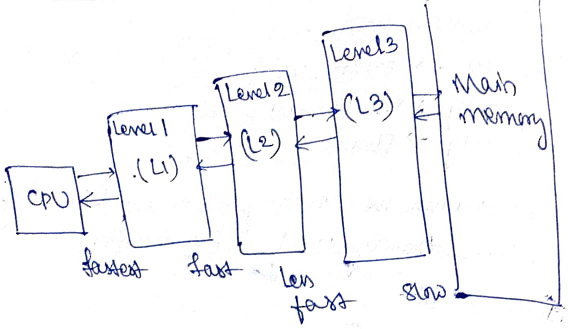
27/3/24
4.2 Cache memory principle.

Cache memory is designed to combine the memory access time of expensive, high speed memory combined with large memory size of less expensive and lower speed memory.

Cache is small amount of fast memory located on CPU Chip or module. It usually lies between main memory and CPU.



(a) Single cache

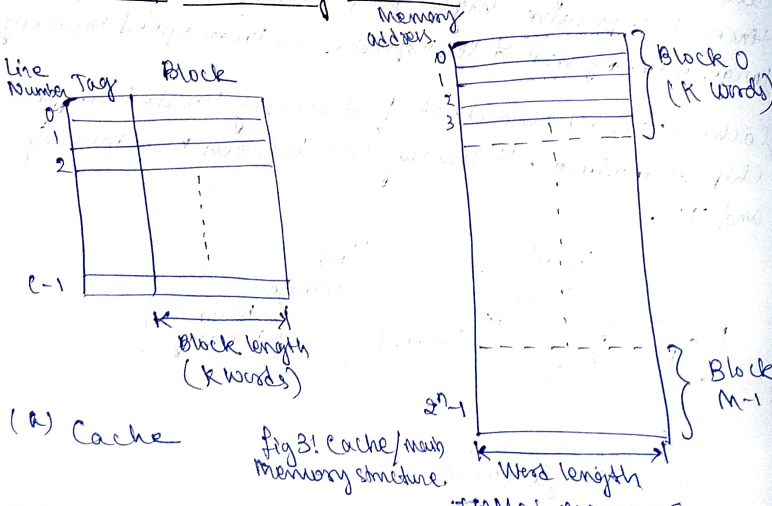


(b) 3-level cache Organization

fig 2:- Cache and Main memory

When the processor attempts to read a word of memory a check is made to determine if the word is present in the cache. If so the word is delivered to the processor. If not a block of main memory consisting of some fixed number of words is read into the cache and then delivered to the processor.

Cache/Main memory Structure



(a) Cache

(b) Main memory

fig 3: Cache/main memory structure.

Main memory consists of upto 2^n addressible words with each word having one unique n bit address. The memory consists of a number of fixed length blocks of K words each. There are $M = \frac{2^n}{K}$ blocks in main memory.

The cache consists of small m blocks called lines (0, 1, 2 ... c-1). each lines contains K words along with few tag bits and control bits. The length of a line not including the tag and control bits is known as the line size.

Line = Tag + Control + Block

Line size = Block

The number of lines is very less than the number of main memory block ie $m << M$.

At any time, some subset of the block of memory resides in lines in the cache. if a word in a block of memory is read, that block is transfered to one of the lines of the cache since there are more blocks than lines and individual line cannot be uniquely and permanently dedicated to a particular block hence each line includes a tag that identify which particular block is currently being stored.

Cache read Operation

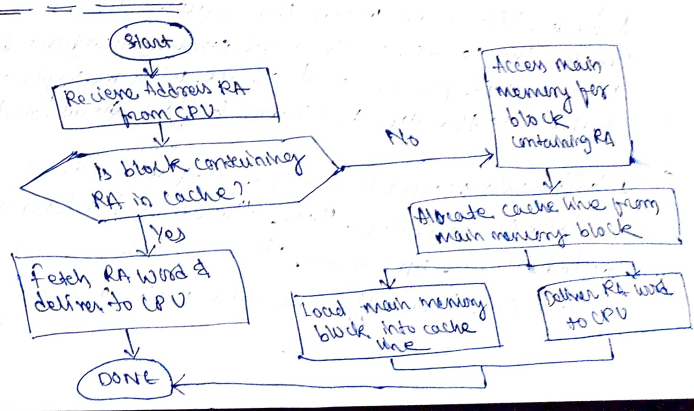


fig 4: Cache Read Operation

The processor generates Read address RA to read the word. If the word is contained in the cache then it is delivered to the processor otherwise the block containing the word is located in the cache and the same word is delivered to the processor simultaneously.

Cache organization

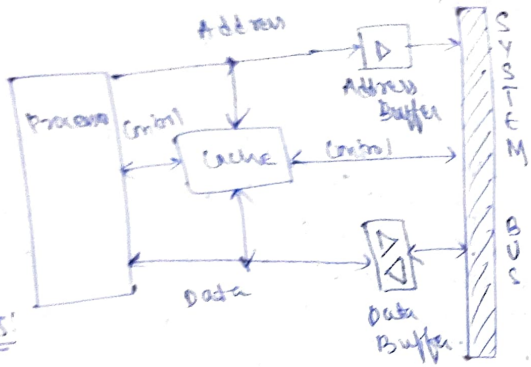


Fig 5

The cache is connected to the processor by means of data, control and address lines. The data and address lines are attached to the data and address buffers through the system bus to reach the main memory.

A 'cache hit' occurs when the required data is available in the the cache. In this case communication is done between the processor and cache with no system bus transfer. Also the data and address buffers are disabled.

A 'cache miss' occurs when the requested data is not available in the cache. Here the desired address is loaded into the system bus. Then the data are returned through the data buffer to both cache and the processor.

4.3 Elements of cache design

a) Cache Address - logical cache, Physical cache.
b) Cache Size -
c) Mapping function - direct mapping, associative, set associative
d) Replacement Algorithm - LRU (least recently used)
- FIFO (first in first out)
- LFU (Least frequently used)
- Random
e) Write Policy - write through, write back
f) Line Size
g) Number of caches - Single/unified / Splitted
- 2-level

Cache Address

MMU - memory management unit

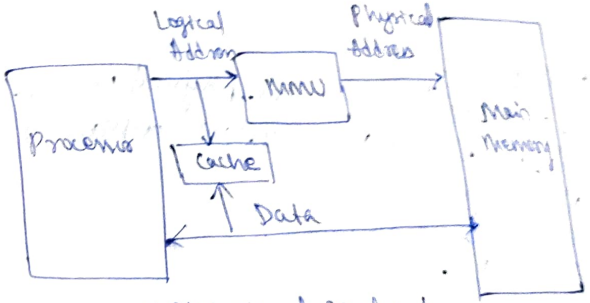
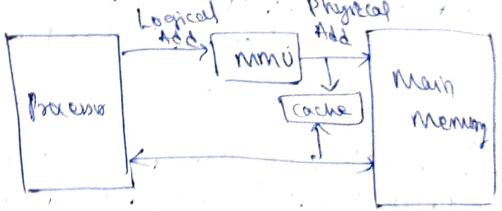


fig 6: (a) Logical cache / Virtual cache



(b) Physical cache

Virtual memory is a facility that allow the programs to address memory from a logical point of view irrespective of the actual amount of main memory physically available. When virtual memory is used, the address field of machine instructions contains the virtual addresses.

For reads to or writes from main memory, a hardware memory management unit (MMU) translates each virtual address into a physical address of main memory.

A logical cache also known as virtual cache stores data using virtual addresses. The processor then access the cache directly without going through the MMU.

A physical cache stores data using main memory physical addresses.

Advantage:
Logical cache is faster than the physical cache because the cache can respond before the MMU performs address translation

V.V. Imp
Mapping function

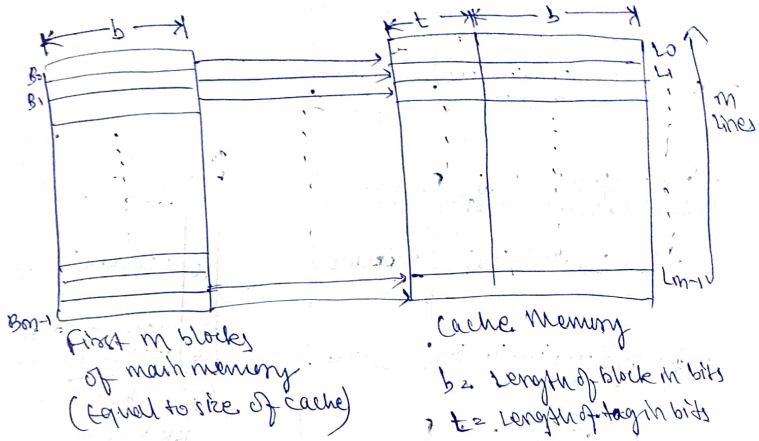
Since there are few number of cache lines than the available main memory blocks hence an algorithm is needed for mapping main memory blocks into cache lines

1) Direct Mapping:

In this mapping each block of main memory maps into only one possible cache line. The mapping is expressed as

$i = j \bmod m$

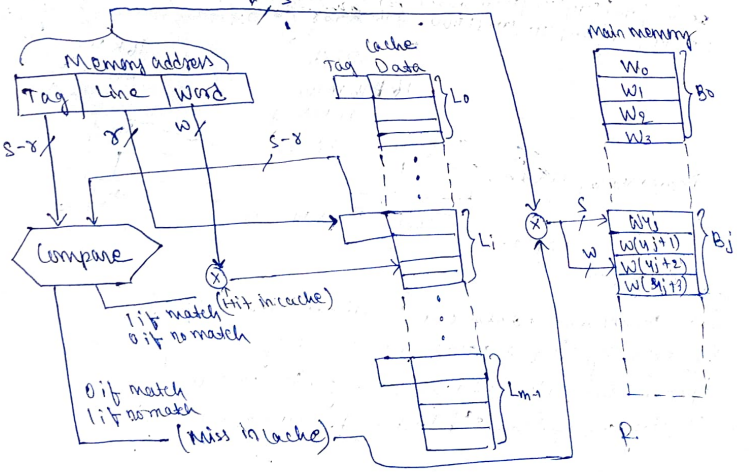
where
- $i \leftarrow$ cache line no.
- $j \leftarrow$ main memory block number
- $m \leftarrow$ number of lines in the cache



b = length of block in bits
t = length of tag in bits

fig 7: (a) Direct Mapping

Fig 8 Direct mapping cache organization



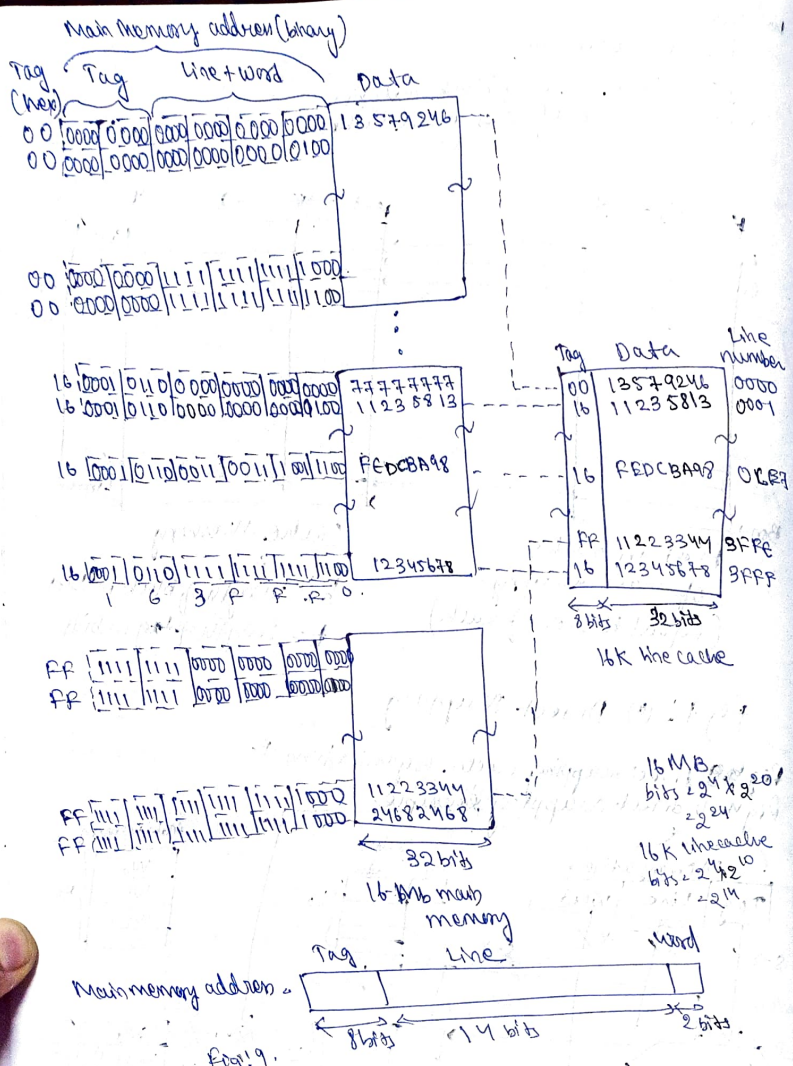


Fig: 9.

4.9 diagram

Each memory address can be divided into 3 fields.
The least significant w bit can uniquely identify a word or a byte within a block of main memory. The remaining s bits specify one of the 2^s blocks of main memory.

The s bits are divided into two fields i.e. $(s-r)$ bits of Tag field and r bits of line field.

1) Address length = $(s+w)$ bits.
2) No. of addressable units = 2^{s+w} words/bytes
3) Block size = Line size = 2^w words
4) Number of blocks in main memory = $\frac{2^{s+w}}{2^w} = 2^s$
5) No. of lines in cache = 2^r = m lines
6) Size of cache = 2^{r+w} words/bytes [Excluding the tag field]
7) Size of tag = $(s-r)$ bits

Cache Memory

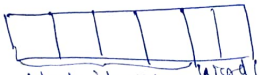
Q) Let, cache size is 16 words
Line size = block size = 4 words
Determine the ~~bits~~ no. of lines in the cache.

No. of bits used $2^{4\text{ bits}} = 16$ No = 4 bits
No. of bits used to represent 4 words = $2^2 = 4$ = 2 bit
No. of lines in cache = $\frac{16 \text{ words}}{4 \text{ words}} = 4$

Q) Ex-1 Let us consider a 64 words of main memory is mapped into 16 word of cache memory using direct mapping function, assume a block size of 4 words

Main memory = 64 words.
No of addressable line = $\log_2 64 = \log_2 2^6 = 6 \log_2 2^1 = 6$.
Physical address of data =

given block size of 4 words
No. of bits used to represent/identify each word = 2 bit.



Main memory structure

	0	1	2	3
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15
4	16	17	18	19
5	20	21	22	23
6	24	25	26	27
7	28	29	30	31
8	32	33	34	35
9				
10				
11				
12				
13				
14				
15				

Cache structure

0				
1				
2				
3				15

0 4 8 12
1 5 9 13
2 6 10 14
3 7 11 15

16 blks

32 → 1 0 0 0 0 0

Block (3) | word (0)

15 → 0 0 1 1 1 1

3 | 3

Fig 10.1

8) Consider a memory system that uses a 32 bit address to address at the byte level, & a cache that uses 64 byte line size. Assume a direct mapped cache with a tag field in the address of 20 bits, Show the address format and determine the no. of addressable units, no. of blocks in main memory, no of lines in cache and size of the tag

Given - The address length = 32 bits ⇒ s+w = 32 bits

Cache size = Block size = 64 byte

w = 6

Given tag s = 20 bits

s-r	r	w
20	6	6

32

(s-r) = 32 - (20+6) = 6

9) Consider a direct mapped cache of size 32 KB, with block size of 32 bytes. The CPU generates 32 bit addresses. Identify the no. of bits needed for cache indexing and the no. of tag bits required

Given cache size = 32 KB

Block size = 32 Bytes

No. of Blocks = $\frac{32KB}{32B} = 2^{10}$

Bits for cache indexing / No. of lines (r) = 10

w = 5

s-r	r	w
17	10	5

32

4.10 explanation

2/4/24

Disadvantage of direct mapping

In case of direct mapping multiple blocks of main memory are assigned to lines of cache as follows.

cache	Main memory blocks assigned			
0	0	m	2m+1	0 → 0 4 8 12
1	1	m+1	2m+1	1 →
2	2	m+2	2m+2	
.	.	.	.	
.	.	.	.	
m-1	2m-1	2m-1	3m-1	

0,1,2,3 | 4,5,6,7 | 8,9,10,11

Line no

0	4	5	6	7
1				
2				
3				

Accentime more

Cache hit

In direct mapping there is a fixed cache location for ~~Thus~~ any given block. Thus if a prog. repeatedly refers two different blocks that map into the same line then the blocks will be continuously swapped in the cache and the hit ratio will be low.

This phenomenon is also known as thrashing. also the miss penalty will be high.

To reduce the miss penalty we need to store the discarded data. since the discarded data has already been fetched, it can be used again at a small cost. such recycling is possible using a victim cache. for this we are using the associative mapping.

2) Associative mapping

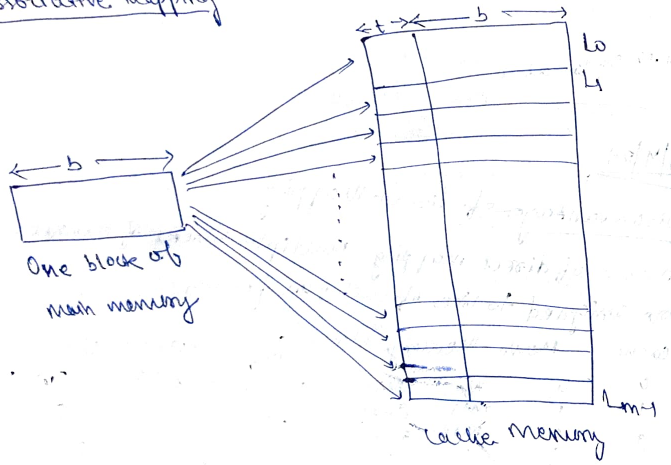


fig 4: Associative mapping.

4.12 Associative mapping maps each main memory block
4.13 to any line of the cache as shown in fig 4. Thus
4.14 4.15 case a memory address consists only a tag and a word

field. The Tag field uniquely identifying a block of main memory. To determine whether a block is in the cache, the cache control logic must examine the every lines tag for a match as shown in fig 4.12

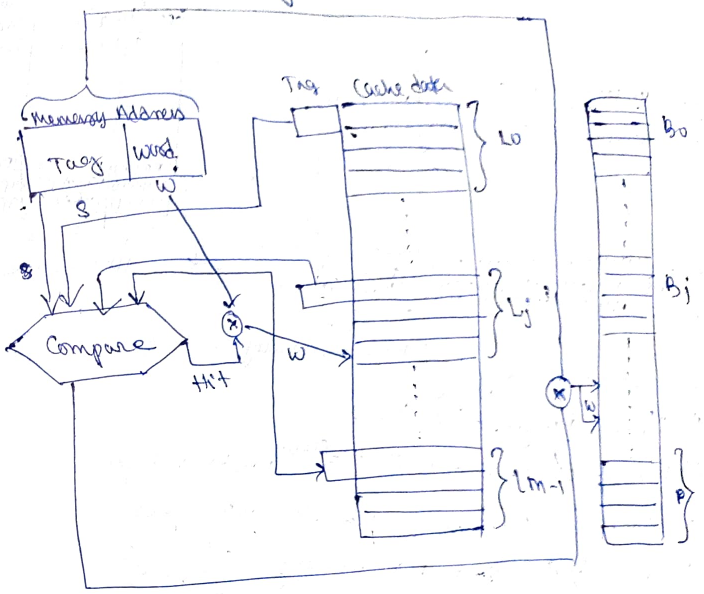


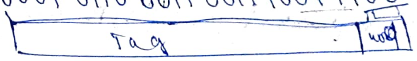
fig 12: Associative mapping

Let us consider the example of associative mapping

A main memory address consist of 22-bit tag and a 2-bit byte number. The 22-bit tag must be stored with the 32-bit block of data for each line in the cache.

Given a 24-bit hexadecimal address 16339C, identify the tag to be stored

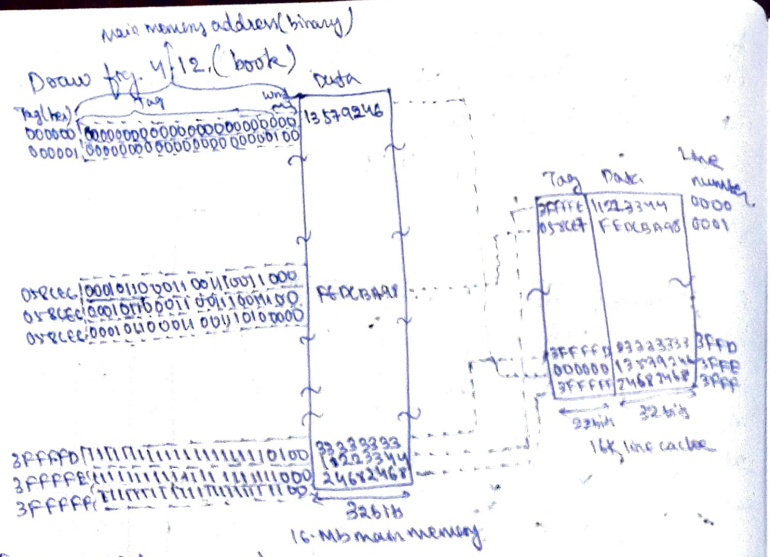
0001 0110 0011 0011 1001 1100



Tag
0001011000110011100111

058CE7

16 8 8 C E 7



for associative mapping

1) Address length = (s + w) bits
2) No of addressable unit = $2^{(s+w)}$ words or bytes
3) Block size = Line size = 2^w words or bytes
4) No of Blocks in main memory = $\frac{2^{s+w}}{2^w} = 2^s$
5) No of lines in cache = Not Defined
6) Size of tag = s bits

5/4/24

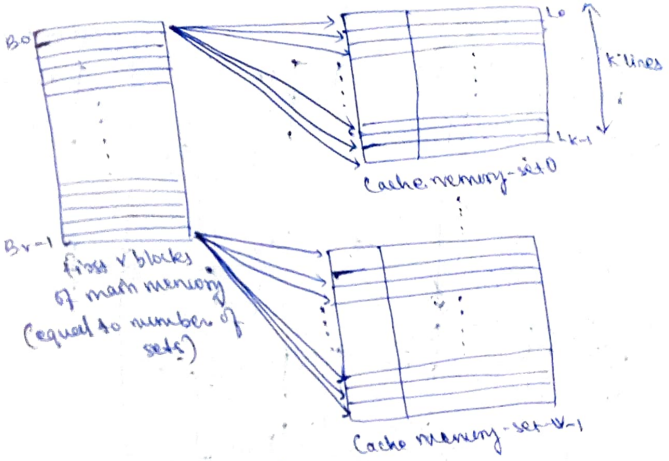
3) Set Associative Mapping

fig 4.13, 4.14 and 4.15 to be drawn

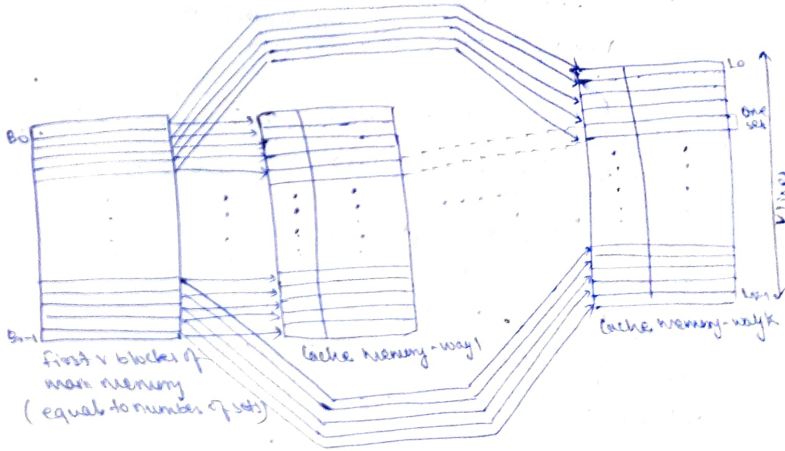
It is a combination of both the direct and associative mapping. In this case the cache consists of no. of sets each of which consists of no. of lines.

The relationship is given as $m = v \times k$ and $i = j \bmod v$

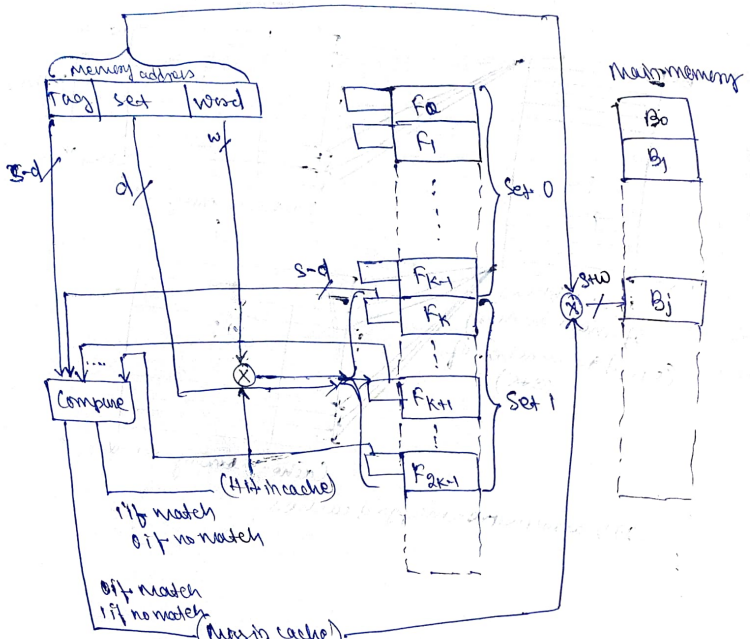
where, i = cache set number
j = main memory block no
m = no of lines in the cache
v = no of sets
k = no of lines in each set



(a) v associative-mapped caches.



(b) k direct-mapped caches.



K-way set - Associative Cache Organization

* Here the memory address is refered as a combination of 3-fields i.e Tag, set and word.

Tag	Set	Word
s-d	d	w

1) Address length = $s-d+d+w = (s+w)$ bits
2) No. of addressable units = $2^{(s+w)}$ words or bytes
3) Block size = line size = 2^w words or bytes
4) No. of blocks in main memory = $\frac{2^{s+w}}{2^w} = 2^s$
5) No. of lines in ~~cache~~ set = K
6) ~~Size of tag~~ Number of set = $v = 2^d$
7) No. of line in cache = $m = K * v = K \times 2^d$

8) Size of cache = $K \times 2^{d+w}$ words or bytes
~~Size of cache = K*~~
9) Size of tag = $(s-d)$ bits.

Q.4.2)

sol^n :- Address : 111111, 666666, BBBBBB

(a)

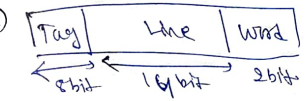
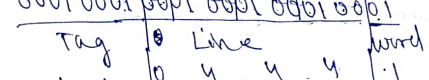


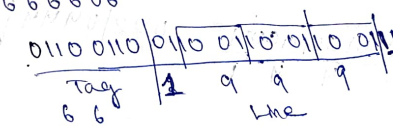
Fig 4.9

Direct Mapping



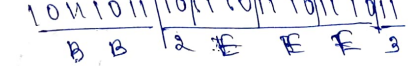
11/04444/1

666666



66/19991/2

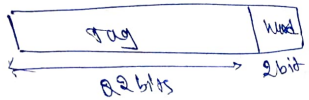
BBBBBB



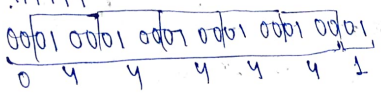
BB/2EEE/3

Associative ~~cache~~ Mapping.

(b)

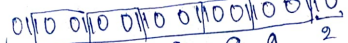


111111



04444441

666666



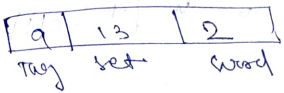
199999|2

BBBBBB

2EEEEE|3

(c) Tag | set | word

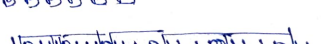
Set Associative mapping



111111

022|0444|1

BBBBBB



177|0EEE|3

666666

CC|1999|2

Q28) (a) Given Address length 16 bits

⇒ S+W = 16

Block size = 8 bytes = line size = $2^W = 8$ ⇒ W = 3

⇒ S = 16 − 3 = 13 ⇒ S = 13

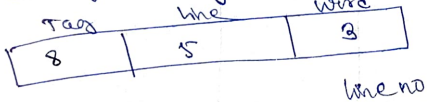
No. of lines in cache = 32 lines

⇒ $2^r = 32$

⇒ r = 5 lines

Size of the tag = s − r = 8

(a)



(b)

		line no
0001 0001	0001 1011	→ 3
1100 0011	0011 0100	→ 6
1101 0000	0001 1101	→ 3
1010 1010	1010 1010	→ 21 ← binary or 15 in hexa

(c)

0001 1010 0001 1010

The three byte addresses are

0001 1010 0001 1010
0001 1010 0001 1000
0001 1010 0001 1001
0001 1010 0001 1011
0001 1010 0001 1100
0001 1010 0001 1101
0001 1010 0001 1110
0001 1010 0001 1111

3 bit: 000, 001, 010, 011, 100, 101, 110, 111

(d) Because the two data or lines with two different memory locations (addresses) can be stored in the same place or in the same line number of the cache hence the tag is used in the cache to identify or distinguish the data

(d) g^{16} ←

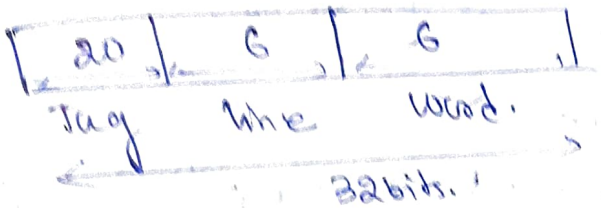
4.19)

a) Given, Tag = 20 bits

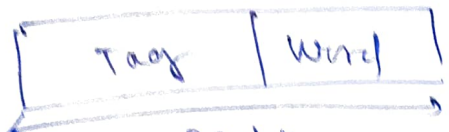
cache = 64 byte

Line size = 64 = 2^6

No. of addressable unit = 2^{32} bytes.



b)



Tag size = 32 - 6 = 26

No. of addressable units = 2^{32} bytes

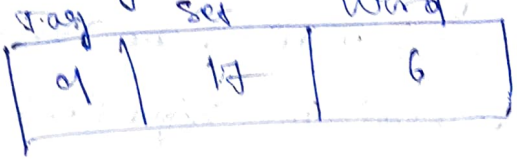
No. of line in the cache = No defined.

No. of Tag bits = 26

c) Using set-associative Mapping.

K = 4

Tag = 9 bits.



32 - (9+6) = 17

No. of line in the set = K = 4

No. of sets in the cache = $2^9 \cdot 2^{17} = 2^7 \cdot 2^{10} = 128$ K

No. of tag bits = 9 bits.

S-d = Tag

d = set

w = word

$K = \frac{64}{4} = 16$

$64 = K \times V = 16 \times 4$

$V = 4$

$V = 2^d \Rightarrow 4 = 2^2$

$\Rightarrow d = 2$

4.1
The cache is divided into 16 set of 4 line each

No. of bits per set = $\log_2 16 = 4$

Main memory = $4K \times 4B = 2^2 \times 2^{10} = 2^{12}$

Tag + set = 12 bits

Tag = 12 - 4 = 8 bits

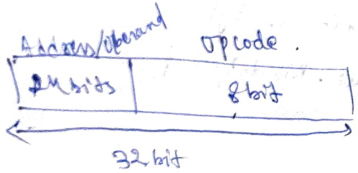
Given, 128 words

No. of bits to identify word = $\log_2 128$

$= \log_2 2^7 = 7$

Chapter-3

3.3



a) Maximum addressable capacity = 2^{24} = 16MB.

b) (i) 32-bit local address bus & 16-bit local data bus.

1 + 2 = 3 cycles

Instruction and data transfer would take 3 bus cycles each 1 for the address and 2 for the data.

If the address bus is 32 bits, the whole address can be transferred to memory at once however since the data bus is only 16 bits, it will require 2 bus cycles to fetch the 32 bit instruction or operand.

(ii) 16 bit local address bus & 16-bit data bus

1 + 2 = 4

store instruction and data transfer would take 4 bus cycles each 2 for the address and 2 for the data.

c) PC = 24

The pc need 24 bits for 24 bit address and the IR needs 32 bits for 32 bit addresses.

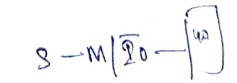
3.4. processor size is 16 bits

so addressable bit = 2^{16}

(a) The max memory address is $2^{16} = 2^6 \times 2^{10}$ = 64KB.

(b) 2^{16}.

(c) Separate I/O instructions are needed because during its execution it will generate separate I/O signals these signals will be different from the memory signals which is generated during the execution of memory instructions. Therefore one more o/p pin will be needed to carry the I/O signals.



S=1 M data

S=0 $\overline{IO} = \bar{0} = 1 - IO$ data.

d) No. of IO ports = 8 bits

Hence it can support 2^8 IO Ports.

3.5 microprocessor size = 32 bit

data bus size = 16 bit