Page No.
Date :

Unit-04 Computer organization & Architecture

Main memory & its types

Main memory is situated b/w the CPU & secondary memory hard disk.

Main Memory

RAM (Random access memory)
→ volatile (till power supply is ON)
→ R/W memory
→ Access method Random
→ writing data is faster

ROM (Read only memory)
→ NON-volatile
→ only for read
→ Access method Random
→ writing data is slower

RAM are of two types :-

① SRAM :→ Static RAM
→ called static RAM because data retain (does not change) till the power is ON.
→ uses flip-flop as memory cell. (1 bit storage)
→ size of memory cell is larger.
→ flip-flop stores "0" & "1"
→ Memory density (memory cell/area) is low. so access time is less.
→ faster
→ costlier
→ used in cache memory

② DRAM :- → Dynamic RAM
→ called dynamic because data changes in the presence of "power" also.
→ uses capacitor as memory cell.
→ size of memory cell is smaller.
→ presence of electric charge on capacitor show "1" and absence shows "0".

→ Due to natural tendency of charge decay on capacitor, the data "1" because "0" after electric charge delay below threshold level.
→ Memory density is high so access time is high.
→ slowest & cheaper.
→ used as primary memory.

ROM types :- chip is used in ROM to write data.

① PROM :- ① Programmable ROM.
② One time programmable by user.
③ Data once written cannot be erased.

② EPROM :- ① Erasable ROM.
② Multitime programmable by user.
③ Data once written can be erased by UV ray.
④ Whole chip is erased ~~&~~ ~~to~~ written again.
⑤ If error, complete chip head to be rewrite again.
⑥ More time in reprogramming.

③ EEPROM :- ① Electrical EPROM.
② Multitime programmable by user.
③ Data once written can be erased using electrical signal.
④ Byte/ block level erasing & written again.
⑤ Less time in reprogramming.

④ Flash memory :- ① similar to EEPROM.
② Multitime programmable by user.
③ Data once written can be erased using electrical signal.
④ Bit level erasing & written again.
⑤ ~~Data~~ Superfast in reprogramming.

(10) → address lines (8) → every address bits

Page No.

A computer employs RAM chips of 256×8 (memory location → 8 bit data at every location) and ROM chips of 1024×8. The computer system needs 2K bytes of RAM, 4K bytes of ROM & four Interface units, each with 4 registers. A memory mapped I/O configuration is used. The 2 highest order bits of the address bus are assigned 00 for RAM, 01 for ROM & 10 for Interface register.

(i) How many RAM & ROM chips are needed.
(ii) Draw the memory address map for the system.
(iii) Give the address range in hexadecimal for RAM, ROM & I/O Interface.

(i) No. of RAM chips = $\frac{\text{Total RAM size}}{\text{One RAM size}}$

$= \frac{2K \text{ bytes}}{256 \times 8}$ 1 byte = 8 bits

$= \frac{2K \times 8}{256 \times 8} = \frac{2K}{256} = \frac{2^1 \times 2^{10}}{256}$

$= \frac{2^{11}}{2^8} = 2^{11-8} = 2^3$

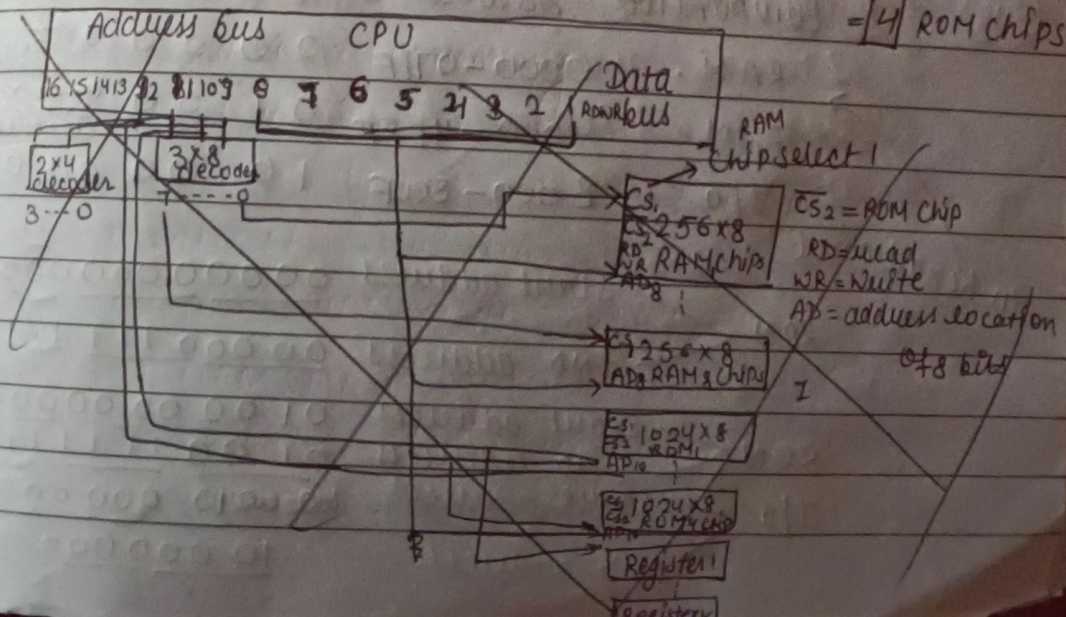
= 8 RAM chips

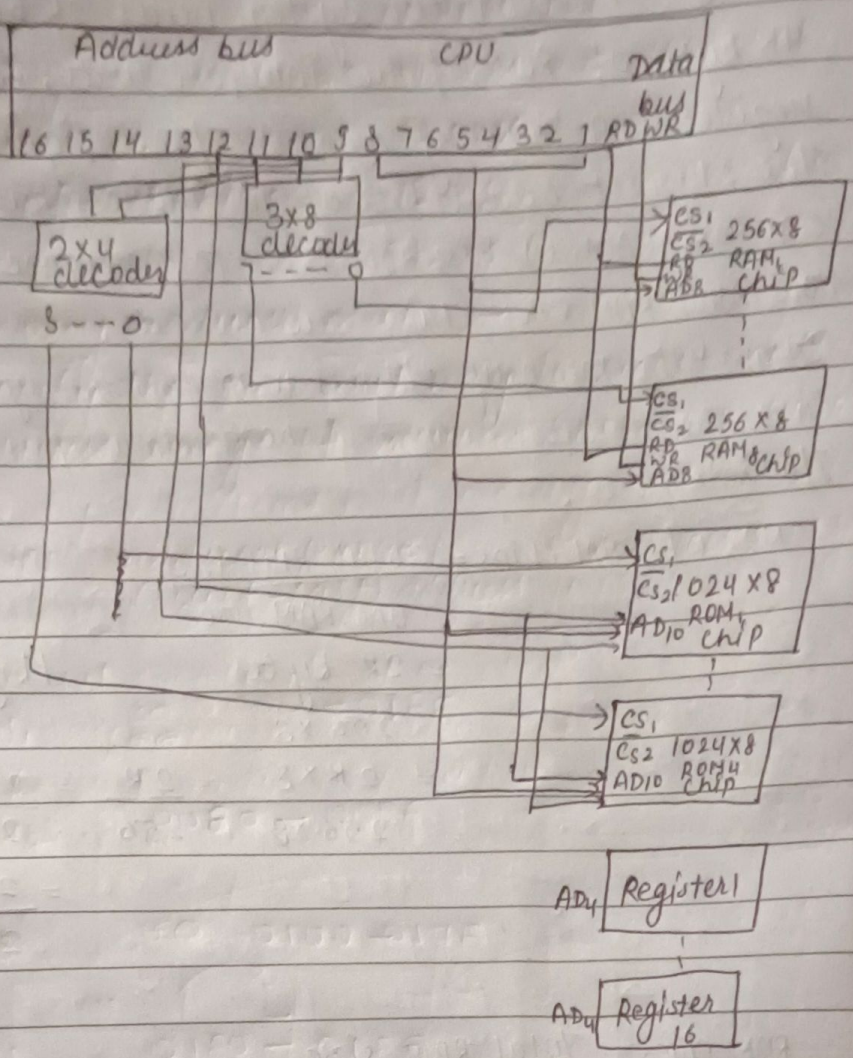
ROM chips = $\frac{\text{Total ROM size}}{\text{One ROM size}}$

$= \frac{4K \text{ bytes}}{1024 \times 8} = \frac{4K \times 8}{1024 \times 8} = \frac{4K}{1024} = \frac{4 \times 2^{10}}{2^{10}}$

= 4 ROM chips

(ii)





No. of I/O interface register = 4×4 = 16 registers

(ii) & (iii) Memory address map

Component	Address	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
RAM	0000-07FF	0	0	0	0	0	← 3×8 decoder →			x	x	x	x	x	x	x	x
ROM	4000-4FFF	0	1	0	0	← 2×4 decoder →				x	x	x	x	x	x	x	x
I/O	8000-800F	1	0	0	0	0	0	0	0	0	0	0	0	x	x	x	x

In hexadecimal

RAM
- Start address: 0000 0000 0000 0000
- END address: 0000 1111 1111 1111

ROM
- Start address: 0100 0000 0000 0000
- END address: 0100 1111 1111 1111

I/O
- Start address: 1000 0000 0000 0000
- END address: 1000 0000 0000 1111

Page No.
Date :

Ques (a) How many 128×8 RAM chips are needed to provide a memory capacity of 2048 bytes.

(b) How many lines of the address bus must be used to access 2048 bytes of memory? How many of these lines will be common to all chips?

(c) How many lines must be decoded for chip select? Specify the size of decoder?

(a) No. of RAM chips = $\frac{\text{Total RAM chips}}{\text{One RAM size}}$

$= \frac{2048 \times 8}{128 \times 8} = \frac{2048}{128} = \frac{2^{11}}{2^7}$

$= 2^4 = \boxed{16 \text{ chips}}$

(b) No. of address lines to access 2048 bytes of memory?

= 2048 bytes

= 2048 × (8) → data lines

= ~~20~~ $2^{(11)}$ → No. of address lines (bits)

= 11

(c) No. of common address lines

= No. of address lines in one chip

= 128 × 8

= $2^{(7)}$ → No. of common address lines

(d) 4 for decoder = 11 − 7 = 4 decoder lines

(c) decoder size

No. of RAM chips = 16

decoder size = (4) × 16

↓

No. of decoded lines = 4

Ques A computer system employs 128×8 of RAM chips, 512×8 chips of ROM. The system needs total of 512 bytes of RAM & 512 bytes of ROM. RAM, ROM chips & draw the address map?

Page No.
Date :

Ans (i) NO. of RAM chips = $\frac{\text{Total RAM size}}{\text{one RAM size}}$

$= \frac{512 \times 8}{128 \times 8}$ = 4 RAM chips

NO. of ROM chips = $\frac{512 \times 8}{512 \times 8}$ = 1 ROM chip

(ii) Memory address map :-
Specify the pictorial way to represent address range of various RAM, ROM chips

128×8 ② → 7 address lines are used for RAM chips

512×8 ③ → 9 address lines are used for ROM chips

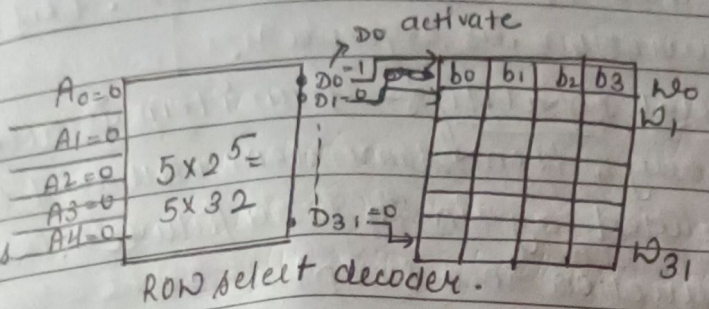
Address Range (First - last) address	Address lines 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
RAM1 0000 - 01FF	S 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 (decoder) E 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1
RAM2 0080 - 00FF	S 0 0 0 0 0 0 0 0 1 x x x x x x x x E 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1
RAM3 0100 - 017F	S 0 0 0 0 0 0 0 1 0 x x x x x x x x E 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1
RAM4 0180 - 01FF	S 0 0 0 0 0 0 0 1 1 x x x x x x x x E 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1
ROM 0200 - 03FF	S 0 0 0 0 0 0 1 x x x x x x x x x x E 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1

Ques A computer employs RAM chips of 256×8 and ROM chips of 1024×8. The computer system needs 2K bytes of RAM, 4K bytes of ROM and four interface units, each with 4 reg

Page No.
Date :

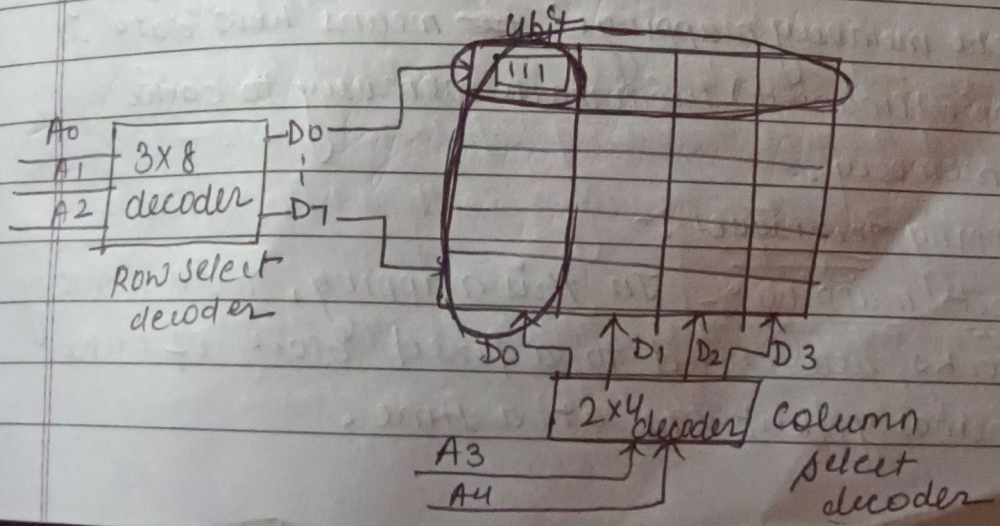
2D RAM :- ① In 2D RAM organization, a RAM of $2^n \times m$ consisting 2^n memory words of m bits each.
② It has a row select decoder of size $n \times 2^n$ to select 1 out of 2^n memory words.
③ Each row (memory words) have m columns (1 column for each individual bit.
④ In 2D RAM organization, hardware is fixed.
⑤ It requires more no of logic gates.
⑥ 2D RAM is more complex.

Organization of 32×4 2D RAM
$= 2^5 \times 4$



2.5 RAM :- ① 2.5D RAM organization, the no. of address lines are divided into approximately equal parts, one for row select select decoder & another for column select decoder.
② In 2.5D RAM, hardware is variable.
③ It requires less logic gates.
④ 2.5D RAM is less complex.

$(32) \times 4$ 2.5D RAM
2^5



Locality of reference in cache memory

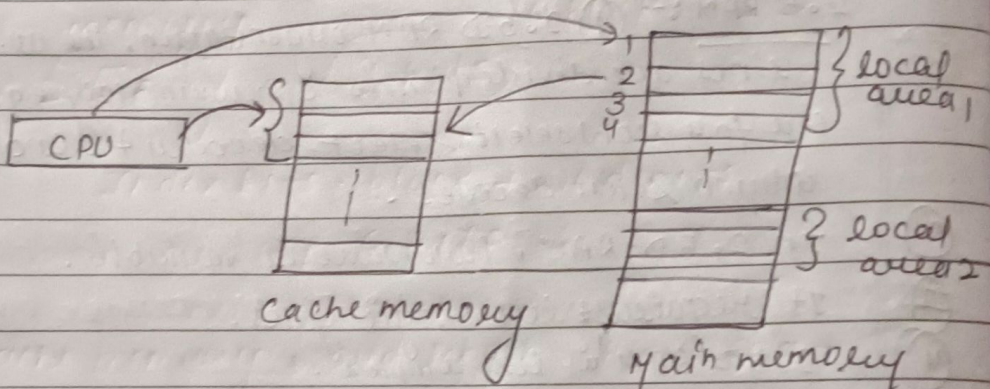
Locality of reference :- also known as principle of locality.

→ locality of reference is the tendancy of a processor to access data and instruction from a certain localized area of ~~local~~ memory.

The locality of reference is also of two types :-

① **Spatial locality :-** If an item is accessed in memory then items with nearby address will tend to be referred next. (Referencing instruction in sequence).

② **Temporal locality :-** If an item is accessed in memory, then some item will tend to be referred again in near future. (Referring instruction in a loop).



Cache memory Mapping Techniques

"cache memory mappings" ~~is~~ means how data is copied (mapped) from main memory to cache memory.

Mapping Techniques :-

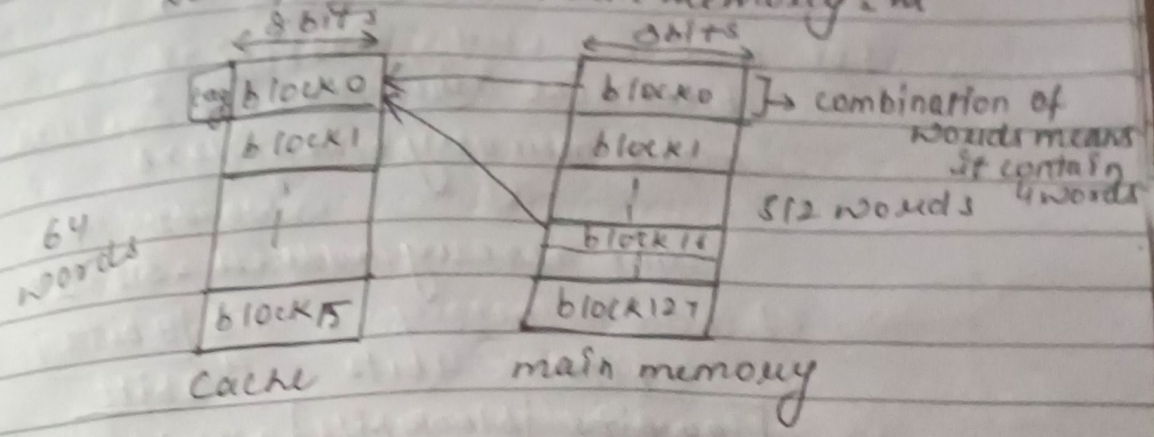
→ **Direct mapping :-** In this mapping, main memory blocks are copied to a fixed block of cache memory, but one at a time.

Let cache memory block no. = C_b

Main memory block no. = M_b

No. of block in cache = C

No. of block in main memory = M



Cache memory block no. = (main memory block no.) mod (No. of cache m/m)

$C_b = M_b \bmod C$

Main memory size = 512 x 8

cache memory size = 64 x 8

Block size = 4 words

No. of main memory blocks = $\frac{\text{Total main memory words}}{\text{block size}}$

$= \frac{512}{4} = 128$ blocks

No. of cache memory blocks = $\frac{\text{Total cache memory words}}{\text{block size}}$

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

Cache memory block no. (C_b) = 0 mod 16

= 0

48 ÷ 16 = 0 (0 gives remainder)

= 16 mod 16 = 0

= 32 mod 16 = 0

= 48 mod 16 = 0

= 17 mod 16 = 1

= 33 mod 16 = 1

No. of main memory blocks in one block of cache = No. of main memory blocks / No. of cache memory blocks

$= \frac{128}{16} = 8$

block size = (4) words $\rightarrow 2^{(2)} \rightarrow$ 2 bits

main memory size = 512 words $\times 8$

main memory address = $512 = 2^9$

= 9 bits

3 bits	4 bits	2 bits
tag bits	block bits	word bits

block size = $4 = 2^{(2)}$ → 2 bits

tag bits → No. of main memory blocks in one block of cache memory → $2^{(3)} = 8$ → 3 bits

block bits → index bits → Main memory address how many blocks are there in cache memory → 16 blocks = $2^{(4)}$ → 4 bits

9 bits

Index bits = block + word bits

= 6 bits

cache memory word length = block size bits + tag bits

= 8 bits + 3 bits

= 11 bits

Ques A digital computer has a memory unit of 64K x16 and a cache memory of 1K words. The cache uses "Direct memory" with a block size of 4 words.

(a) How many bits are there in the tag, index, block and word field of address format.

(b) How many bits are there in each words of cache, and how are they divided into functions? ~~and~~ include a valid bit.

(c) How many blocks can the cache accomodate?

(a)

Index bits

tag	block	word

block size = 4 words = 2^{2} → 2 bits of word bits

block size of cache memory = $\frac{\text{Total cache memory word}}{\text{block size}}$

$= \frac{1K}{4} = \frac{1 \times 1024}{4} = 256$ blocks

block size of main memory = $\frac{64K}{4} = 16K = 16 \times 1024 = 16384$

tag = $\frac{\text{No. of main memory block}}{\text{No. of cache memory block}}$

$= \frac{16 \times 1024}{256} = 64 = 2^{6}$ → 6 bits

No. of tag bits = 6

256 blocks in cache = 2^{8} → 8 bits

block bits = 8 bits

Index bits = 8 bits + 2 bits ⇒ Address bits = 16 bits

= **10 bits**

verify main memory

$= 64K$

$= 2^{6} \times 2^{10}$

$= 2^{16}$ → bits

(b)

1 bit	6 bits	16 bits
valid bit	tag	data

16 + 6 + 1 = 23 bits

(c) 256 blocks

Page No.
Date :

* **Associative Technique of :-**

In associative mapping, main memory blocks are copied into any block of cache memory.

Let main memory size = $\frac{512 \times 8}{\downarrow \text{word} \quad \downarrow \text{word size}}$

cache memory size = $\frac{64 \times 8}{\downarrow \text{word} \quad \downarrow \text{word size}}$

block size = 4 words

No. of main memory blocks = $\frac{512}{4}$ = 128 blocks

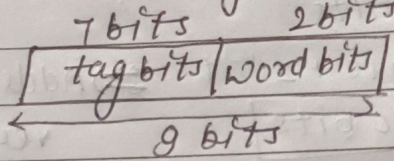
No. of cache memory blocks = $\frac{64}{4}$ = 16 blocks

No. of main memory blocks in one block of cache = 128 = 2^{7}

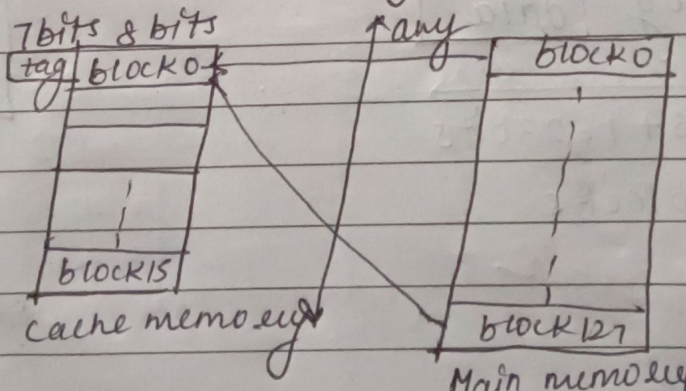
Main memory size = $\frac{512 \times 8}{\downarrow \text{words}}$

= 2^{9}

main memory address bits = 9 bits



Main memory address



Associative mapping

Associative memory

① An associative memory is also called content addressable memory (CAM).

② This type of memory is accessed simultaneously and in parallel of data content rather than by specific address or location.

③ (i) When a word is written in associative memory, no address is given.

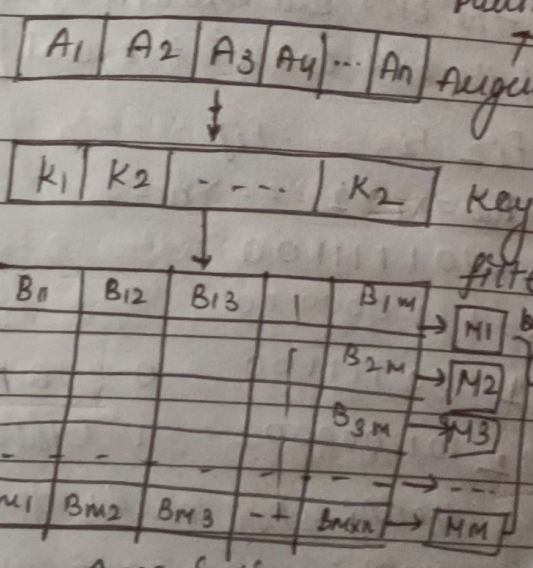
(ii) The memory is capable of finding an empty unused location to store the word.

④ When a word is to be read from an associative memory, then the content of the word or part of the word is specified.

(ii) The memory locates all words that match the specified content and marks them for reading.

⑤ It takes relatively less time to find an item based on content rather than by an address & Expensive

Block diagram



A = 101 1111 00

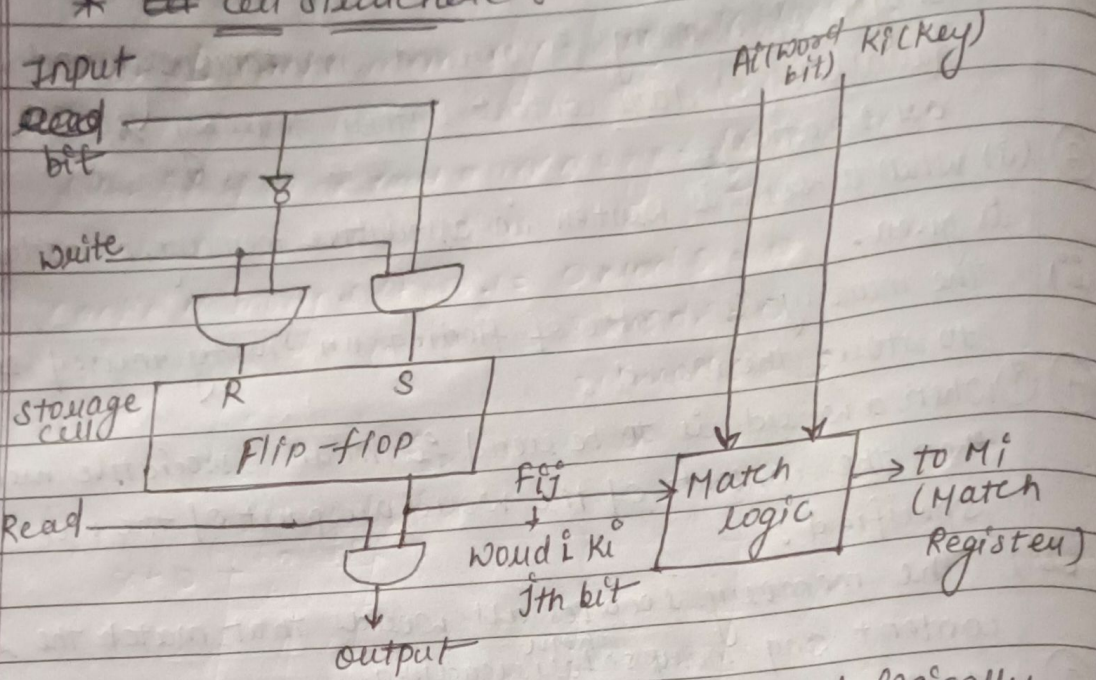
K = 111 000000

W1 = 10011100 (NO match)

W2 = 10100001 (Match)

Cell structure & Match logic in Associative memory

* ~~Cell~~ Cell structure :-



(1) The equality of two bits can be expressed logically using :-

$$x_j = A_j F_{ij} + A'_j F'_{ij}$$

If the bits are equal, $x_j = 1$

If the bits are not equal $x_j = 0$

(2) $W_i = F_{i1}\ F_{i2}\ F_{i3} \cdots F_{ij} \cdots F_{in}$

$M_i = x_1\ x_2 \cdots x_n = 1$

(3) A = 101111100

K = 111 0000 00

Word 1 100 111100 0 (NO Match)

Word 2 101 000001 1 (Match)

(4) If $K_j = 0$, then A_j & F_{ij} needs on comparison

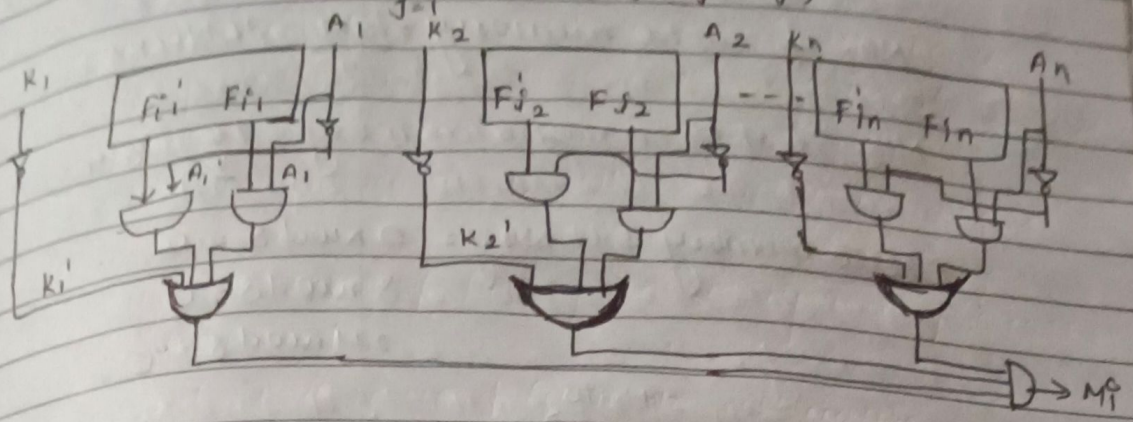
If $K_j = 1$, then A_j and F_{ij} ~~needs~~ are compared

(5) So, $M_i = (x_1 + K_1')\ (x_2 + K_2') \cdots (x_n + K_n')$

so the general function for match logic is

$$M_i = \sum_{j=1}^{n} (x_j \cdot K_j')$$

$$M_i = \sum_{j=1}^{n} (A_j \cdot A_{1S} + A_j' \cdot F_{ij}' + K_j')$$



Match logic of one word of associative memory

Mapping Technique :-

(3) Set associative Mapping :- It is a direct mapping & associative mapping.

Set associative mapping = direct mapping + Associative mapping

In this cache memory is divided into set :-

set = group of blocks

block = group of words.

Cache memory set NO. = (Main memory block NO.) mod (NO. of set in cache memory)

NO. Of sets in cache memory = NO. Of cache memory blocks / set size

Set size = 2 (Two way set of associative)

Block size = 4 words

Page No.
Date :

Main memory size = 512 X 8

Cache memory size = 64 X 8

No. of main memory blocks = $\frac{512}{4}$ = 128 blocks

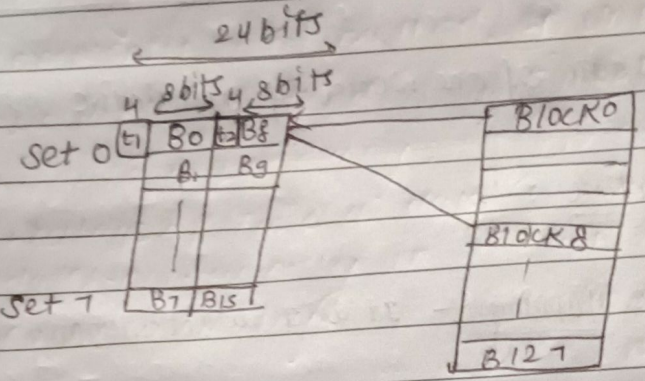
No. of cache memory = $\frac{64}{4}$ = 16 blocks

No. of sets in cache memory = $\frac{16}{2}$ = 8 sets = 2^{3} → 3 bits of sets

Cache memory set NO. = 0 mod 8 = 0

= 8 mod 8 = 0

= 24 mod 8 = 0



No. of main memory blocks in one set of cache (2 at a time) = $\frac{\text{Main memory block}}{\text{cache memory block}}$ = $\frac{128}{8}$ = 16 = 2^{4}

index

4 bits | 3 bit | 2 bits

| tag | set | word |

4 word = 2^{2} → 2 bit

Main memory address bits = 9 bits

$512 = 2^{9}$ → 9 bits

Page No.
Date

Ques A two-way set-associative cache memory uses blocks of four words. The cache can accomodate a total of 2048 words of main memory. The main memory size is 128K×32.
(a) formulate all pertinent info. required to construct the cache memory.
(b) what is the size of cache memory = No. of words × Word length
= $2^{\text{index bits}}$

Soln set size = 2
Block size = 4

No. of cache memory block = $\frac{2048}{4}$ = 512 block

No. of main memory block = $\frac{128K}{4}$ = 32K

No. of cache memory sets = $\frac{512}{2}$ = 256 sets

(a) tag = $\frac{\text{main memory block}}{\text{Cache memory sets}}$ = $\frac{32K}{256}$ = 128
= 2^7 = 7 bits

sets = No. of sets in cache
= 256 = 2^8 = 8 bits
words = 4 = 2^2 → 2 bits
Index = 8+2 bits = 10 bits

(b) 7bits 32bits 7bits 32bits = 78 bits = word length

| tag | data | tag | data |

Cache size = 1024 × 78

32
32
7
-7
78

Before	Write back	After
x = 100	CPU	x = x+200
↑		↓
x = 100	cache m/m	x = 300
↑		↓
x = 100	Main memory	x = 100

When use of x is over in memory than main memory variable x is updated

Write through	After
CPU	x = x+200
	↓
Cache m/m	x = 300
	↓
main memory	x = 300

consistent but waste of time of changing the value of main memory

Write back	Write through
(1) Data is updated in cache memory.	(1) Data is updated in both cache & main memory.
(2) simple method to implement.	(2) complex method to implement.
(3) When a cache block is not required then selected cache block is written back to main memory to update the main memory.	(3) When cache block is not required, there is no need to written back to main memory.
(4) commonly used method.	(4) Unpopular method.
(5) Unreliable method	(5) Reliable method.
(6) No data redundancy.	(6) Data redundancy.
(7) No wastage of time as data updates occur only in cache memory.	(7) Wastage of time as each data updates both in cache & main memory.

Virtual Memory

It is a concept that gives an illusion to the user that he has sufficient memory to execute any application / program of any size.

(1) Virtual memory allows a no. of applications having total size more than the main memory size, to run at the same time.

(2) Virtual memory is a simulated memory that is written to a file on the hard drive. This file is called page file or swap file. It is used by operating system to simulate physical RAM by using hard disk space.

Page No.
Date

(3) In windows 1.0, 2.0 versions there was no virtual memory, so we were not able to run a no. of applications due to run out of RAM space.

(4) However from windows 3.0 ~~on~~ onwards, concept of virtual memory was introduced.

(5) To implement it, a portion of hard drive is reserved by the system. This portion can be either a file or a seperate partition. In windows, it uses a file called pagefile.sys. In linux, a separate partition is used for virtual memory.

(6) When the system needs more memory, it maps some of its memory address out to the hard disk drive. This extra memory does not actually exist in the RAM, it is the storage space on the disk drive.

(7) The more RAM (main memory) your computer has, the faster your programs will run.

(8) If lack of RAM is slowing our computer, then we can increase virtual memory to compensate. However, adding more RAM gives high speed rather than virtual memory because it takes more time to swap data from hard disk than from RAM.

(9) for windows 10
initial virtual memory size = 1.5 × RAM size
= 1.5 × 4GB = 6GB
MAX. = 3 × initial size = 6GB × 3 = 18GB

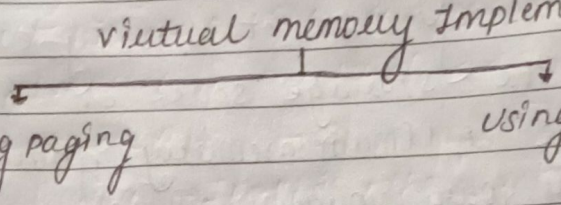
* Virtual address (logical address) :- Each address in virtual memory is called virtual address.
* Address space :- set of all virtual addresses.
* Memory address (physical address) :- Each address in main memory.

Page No
Date :

Memory space :- The set of all memory address.

Swapping :- Swapping is a mechanism in which a process temporarily moved out from main memory to secondary storage (disk) and another process moved in from secondary storage (disk) to main memory. After some time, the first process again brough back to main memory.

Address mapping :- address mapping specify how to convert virtual address to memory address.



Page :- virtual memory is divided into equal size of groups. Each group is called page.

① virtual memory (address space) is a group of page.

② Each page consists of a no. of words called page size.

let virtual memory size (address space) = 8K words

page size = 2K words = block size

Block (frame) :- Main memory is divided into equal size of groups. Each group is called block (frame).

① Main memory (memory space) is a group of blocks.

② Each block (frame) consists of a no. of words called block size.

page size = block size

(a) No. of pages = $\frac{\text{address space size}}{\text{page size}}$

(b) No of blocks = $\frac{\text{memory space size}}{\text{block size}}$

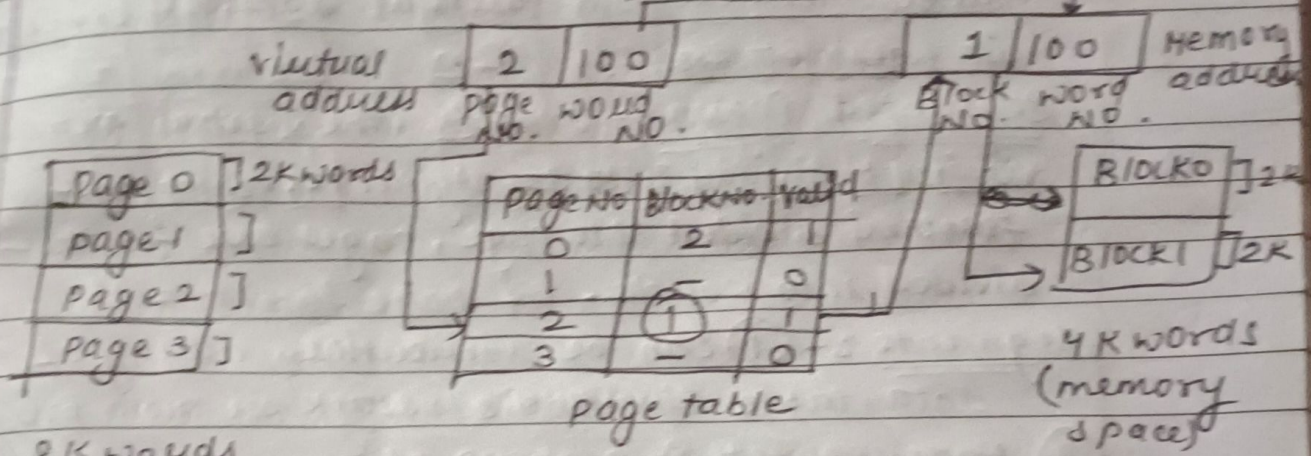
(c) Virtual address bits = n
address space = 2^n

(d) memory address bits = m
memory space = 2^m

Address mapping using paging

Page size = 2K words = block size

No. of pages = $\frac{8K}{2K} = 4$



A virtual memory system has a page size of 1K words. There are 8 pages and 4 blocks. The associative memory page table contains the following entries :-

Page	Block
0	3
1	1
4	2
6	0

Make a list of all virtual addresses (in decimal) that will cause a page fault if used by the processor.

Ans

[0, 1023]	Page 0 ✓	7 1K words
+1023 [1024, 2047]	Page 1 ✓	
+1023 [2047, 3071]	page 2 ✗	
+1023 [3072, 4095]	page 3 ✗	
+1023 [4096, 5119]	page 4 ✓	
+1023 [5120, 6143]	page 5 ✗	
+1023 [6144, 7167]	page 6 ✓	
+1023 [7168, 8191]	page 7 ✗	

virtual memory

Page	Virtual address (decimal)
page 2	2048 - 3071
page 3	3072 - 4095
page 5	5120 - 6143
page 7	7168 - 8191

Ans

~~Segmented Paging :-~~

Page fault occurs when a process attempts to access memory in virtual address space that it does not own.

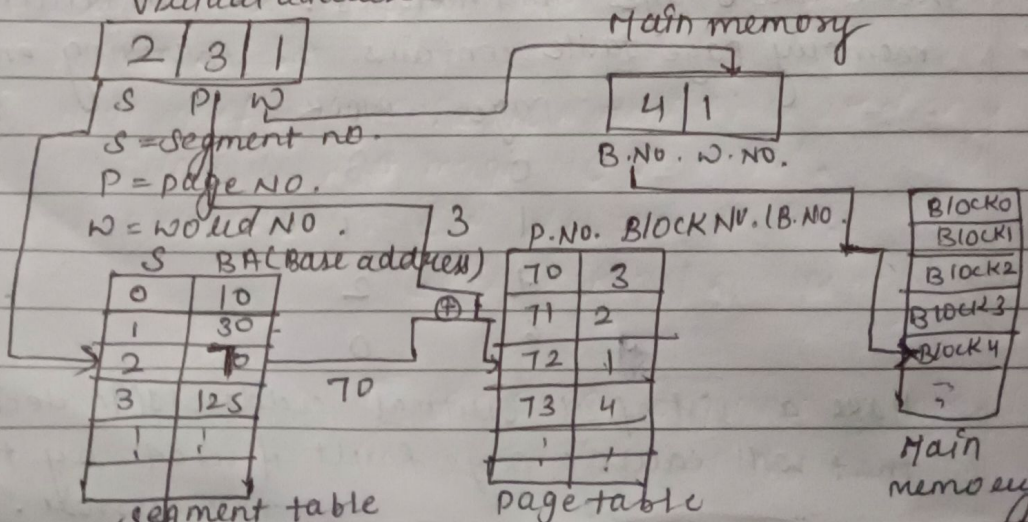
Segmented Paging :- In this, the virtual memory is first divided into ~~pa~~ segments, which are further divided into page, pages are further divided into words.

Segments :- (1) can be subroutine, array of data, table of symbols, stack or main program.

(2) Each segment is a group of pages.

(3) Each page is a group of words.

Address Mapping



page fault :- when we require a particular page & that particular page is not available in main m/m.

BA → Base address (starting address)

Ques The logical address space in a computer system consists of 128 segments. Each segment consists of 32 pages of 4K words in each. Physical memory consists of 4K blocks of 4K words in each. Formulate the logical & physical address formats.

Sol No. of segments = 128
$= 2^7$

No. of pages in each segments = 32
$= 2^5$

No. of words in each pages = 4K
$= 2^2 \times 2^{10}$
$= 2^{12}$

7 bits	5 bits	12 bits
segment	page	word

Logical address format

12 bits	12 bits
Block	word

Physical address

No. of blocks = $\frac{4K}{} = 4 \times 2^{10} = 2^{12}$

No. of words = $\frac{4K}{} = 2^{12}$

Page replacement algorithm in memory organization :-

① FIFO (first in first out) :- page comes first in main memory will be moved out first.

page reference :-

Main memory is empty initially ∴ page fault occurs

	4	2	0	1	2	6	1	4	0	1	0	2	3	5	7
	4	4	4	4		6		6				6	6	5	5
		2	2	2		2		4				4	4	4	7
			0	0		0		0				2	2	2	2
				1		1		1				1	3	3	3

No. of page fault = 10

Assume :-

Virtual address space = 8K words

No. of pages = $\frac{8K}{1K}$ = 8 pages

Main memory space = 4K words

page size = block size = 1K

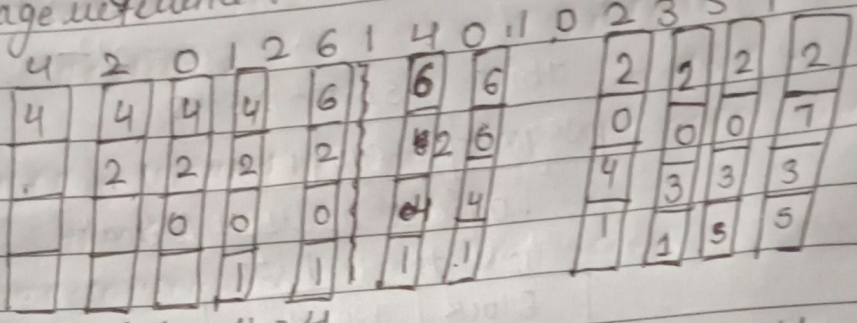
No. of blocks = $\frac{4K}{1K}$ = 4 blocks

(2) LRU :- Least recently used algorithm

Replace the page that has been used least recently.

Page reference :-

4 2 0 1 2 6 1 4 0 1 0 2 3 5 7



No. of block = 4

No. of page fault = 11

Auxillary memory

* Magnetic Tape :-

(1) coated strip of plastic on which data can be recorded.

(2) Data is stored permanently & non-volatile in nature.

(3) Made up of mylar (plastic material) coated only one side side with magnetic material.

characteristics :- (1) Read/write head.

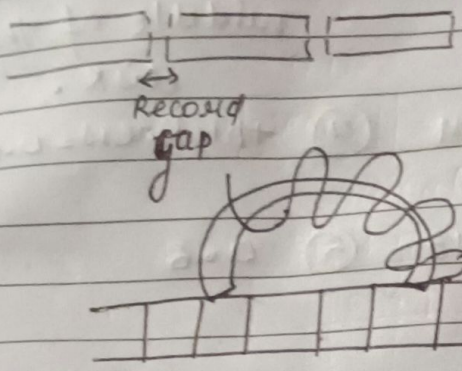
(2) Erase head erases previous data.

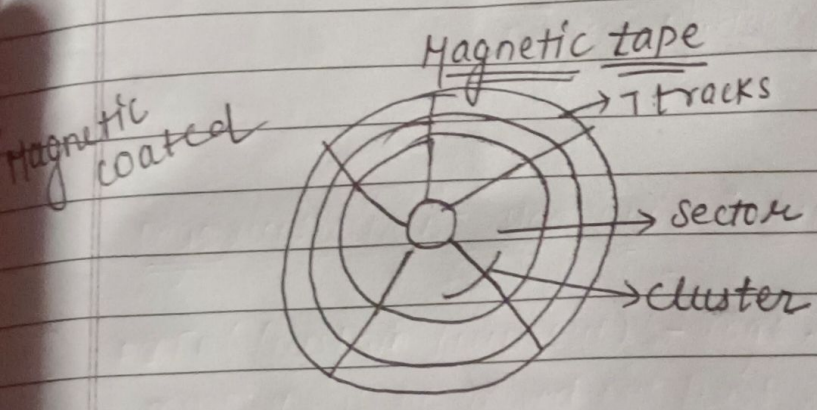
(3) ~~Essential~~ sequence access of data

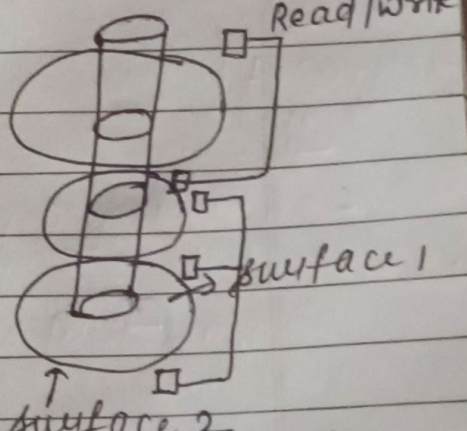
(4) primarily used for back up

Disadvantage :- ① Magnetic material is coated on only one side therefore when magnetic material reaches it does not able to read/write head.
② As it is made up of plastic it is fragile and hence it can easily be damaged.
③ ~~It is~~ sequence access of data.

* Advantage :- ① Strength & durability
② ~~Rugged~~ High capacity storage
③ Data security
④ Inexpensive.
⑤ Simple.







① Each track has sectors, the data is first store in first track sector then move to the another track. Two adjacent sectors are known as cluster.

Page No.
Date :

Optical disk

CD / DVD's

CD (compact disk)	DVD (digital versatile disk)
(1) ~~used for storage~~ consists of one reflective layer.	(1) Data can be changed in it. consists of two reflective layers.
(2) Has less capacity of data storage.	(2) Have more capacity of data storage.
(3) The CD's metal or recording layer is located on the top of the disk.	(3) The metal layer or recording layer is located in the middle of the disk.
(4) The CD is around 1.2mm thick.	(4) thickness = 0.6mm
(5) Numeric aperture = 0.45	(5) 0.6