MEMORY ORGANIZATION

MCA-05/MSC(IT)-05/PGDCA-05/BCA-II

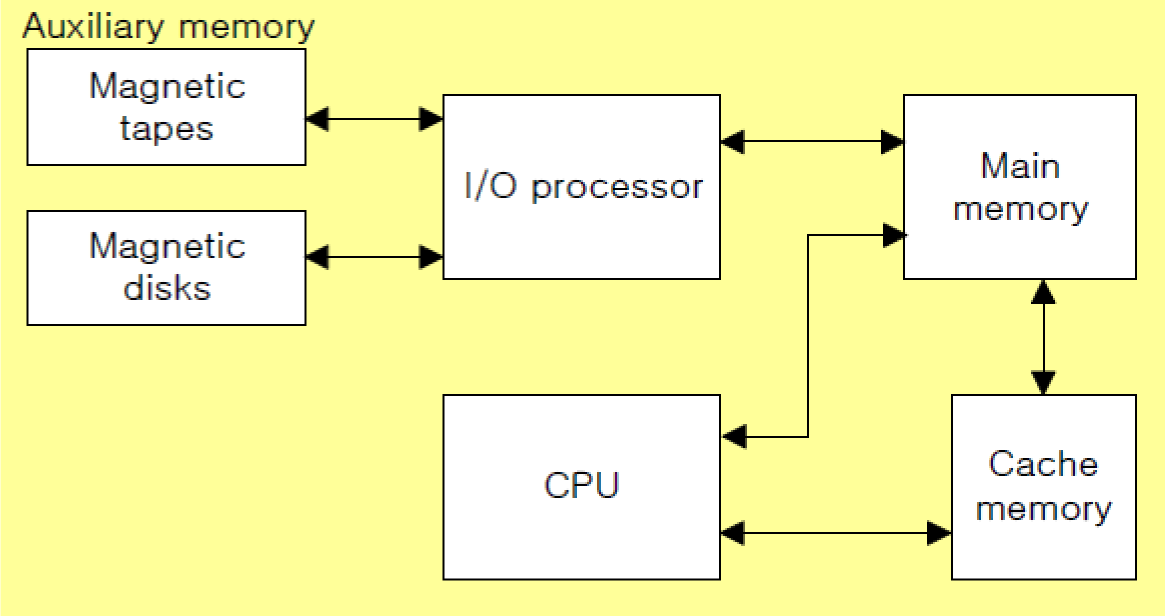
12-1 Memory Hierarchy

- Memory hierarchy in a computer system :

  Main Memory : memory unit that communicates directly with the CPU (RAM)

  Auxiliary Memory : device that provide backup storage (Disk Drives)

  Cache Memory : special very-high-speed memory to increase the processing speed (Cache RAM)

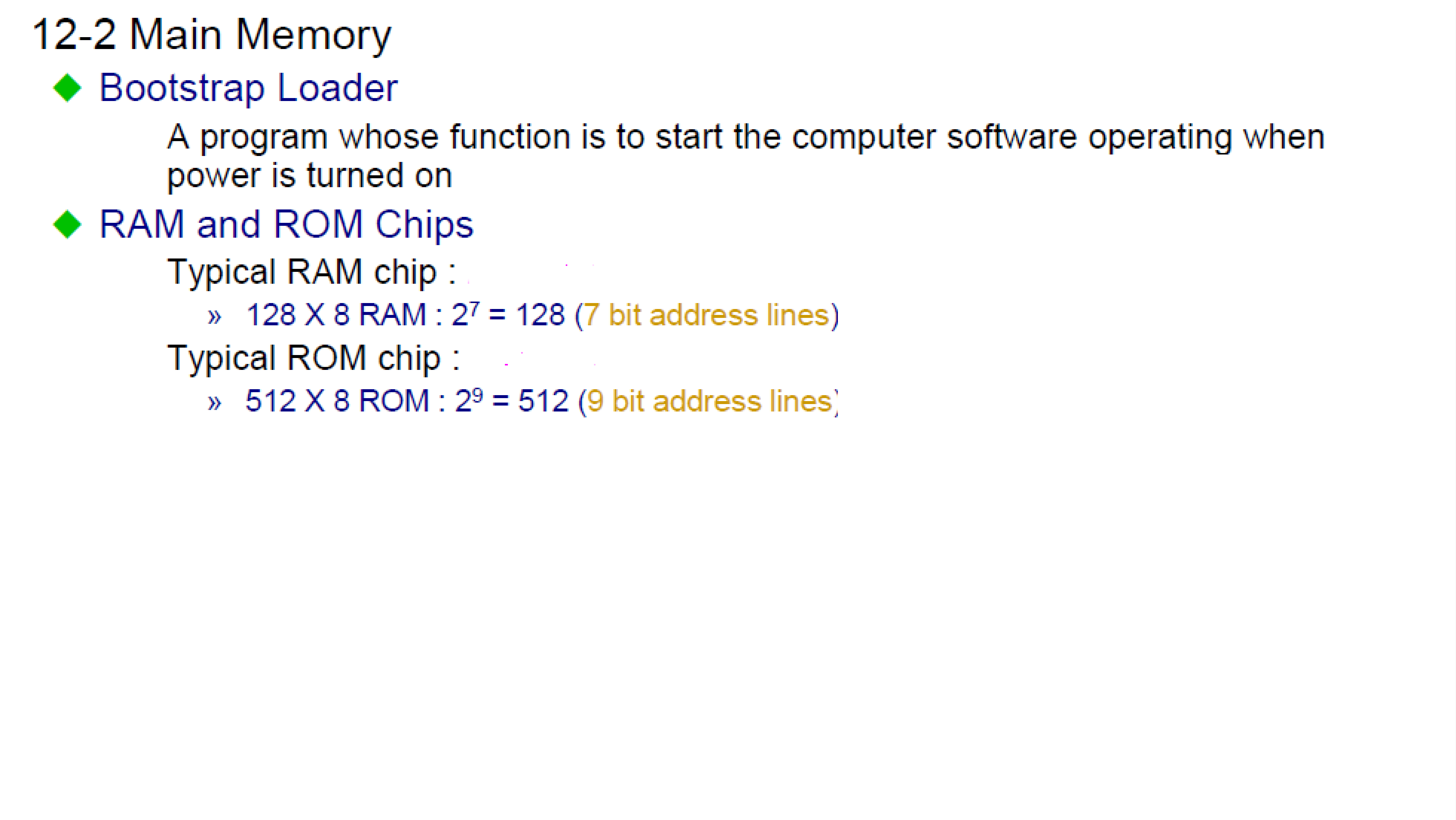


- Multiprogramming

  enable the CPU to process a number of independent program concurrently

- Memory Management System :

  supervise the flow of information between auxiliary memory and main memory



12-2 Main Memory

- Bootstrap Loader

  A program whose function is to start the computer software operating when power is turned on

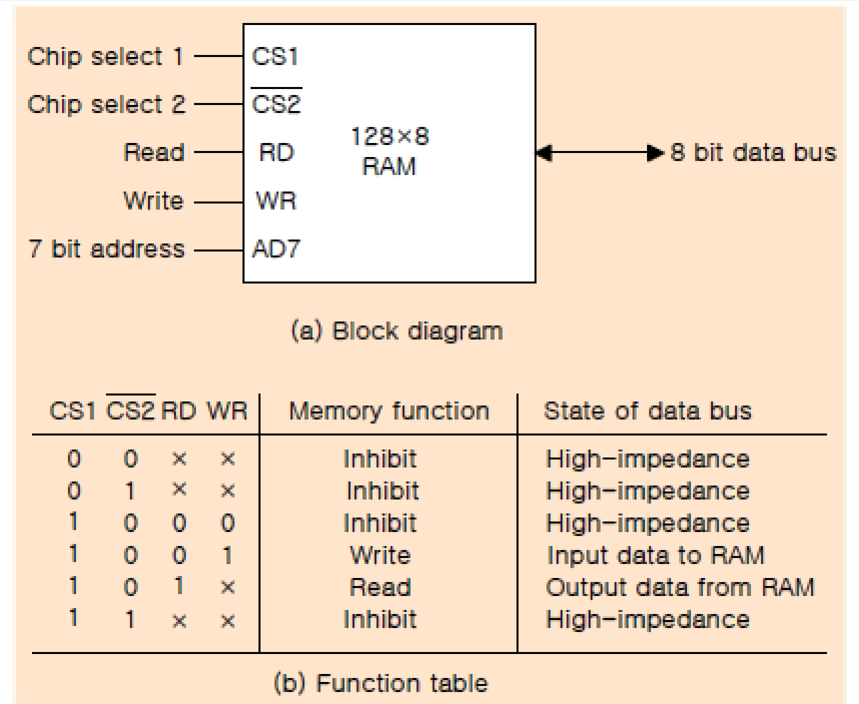
- RAM and ROM Chips

  Typical RAM chip :

  » 128 X 8 RAM : $2^7 = 128$ (7 bit address lines)

  Typical ROM chip :

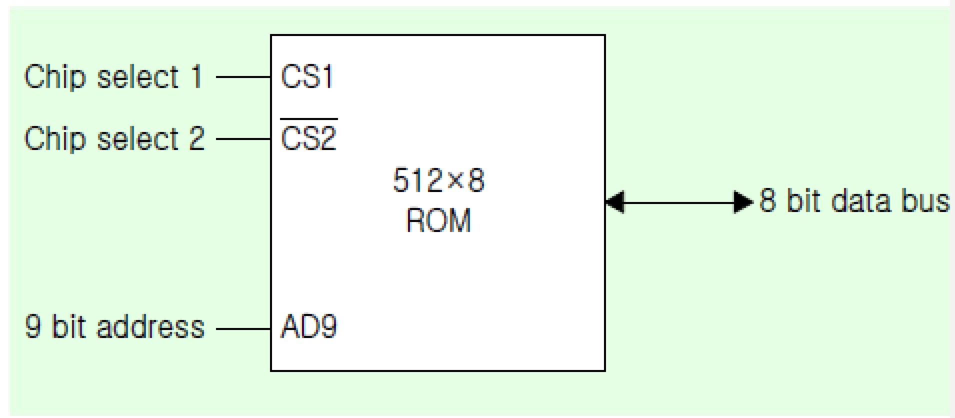
  » 512 X 8 ROM : $2^9 = 512$ (9 bit address lines)

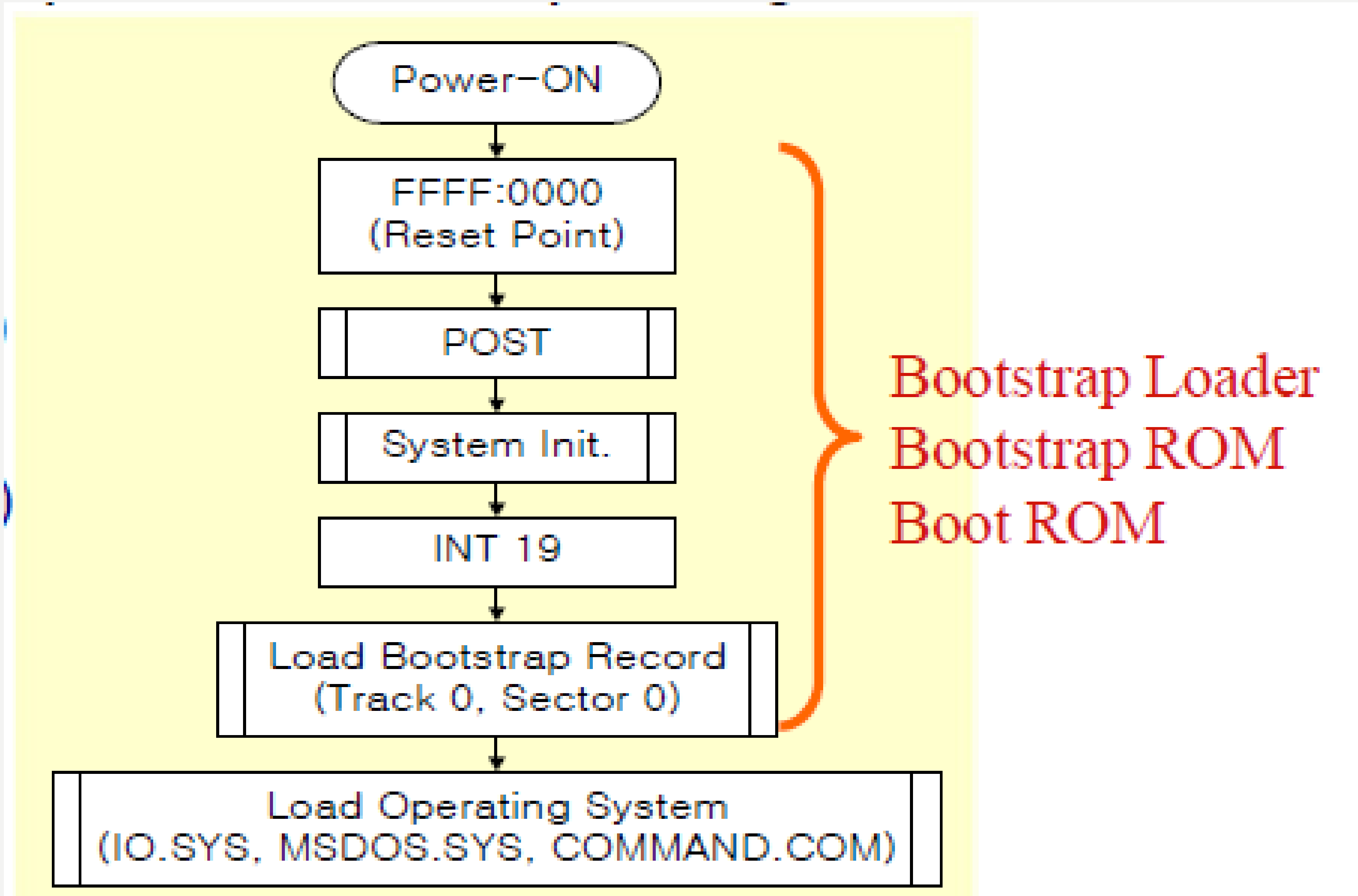


(a) Block diagram

CS1	$\overline{CS2}$	RD	WR	Memory function	State of data bus
0	0	×	×	Inhibit	High-impedance
0	1	×	×	Inhibit	High-impedance
1	0	0	0	Inhibit	High-impedance
1	0	0	1	Write	Input data to RAM
1	0	1	×	Read	Output data from RAM
1	1	×	×	Inhibit	High-impedance

(b) Function table


Chip select 1
Chip select 2
9 bit address
CS1
$\overline{CS2}$
AD9
512×8
ROM
8 bit data bus



Power-ON

FFFF:0000
(Reset Point)

POST

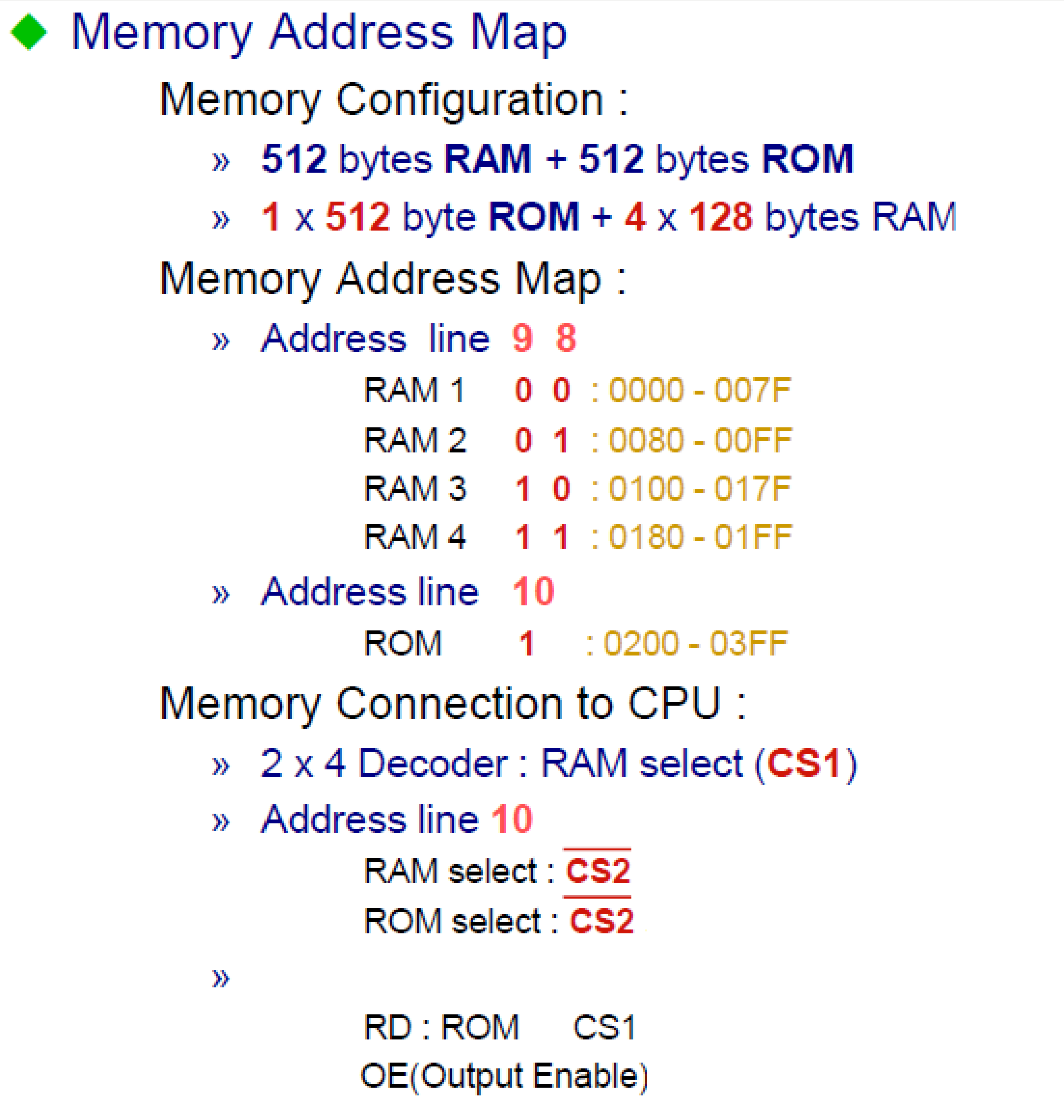
System Init.

INT 19

Load Bootstrap Record
(Track 0, Sector 0)

Load Operating System
(IO.SYS, MSDOS.SYS, COMMAND.COM)

Bootstrap Loader
Bootstrap ROM
Boot ROM



Memory Address Map

Memory Configuration :

» **512** bytes **RAM** + **512** bytes **ROM**

» **1** x **512** byte **ROM** + **4** x **128** bytes RAM

Memory Address Map :

» Address line 9 8

RAM 1	**0**	**0**	: 0000 - 007F
RAM 2	**0**	**1**	: 0080 - 00FF
RAM 3	**1**	**0**	: 0100 - 017F
RAM 4	**1**	**1**	: 0180 - 01FF

» Address line 10

ROM	**1**	: 0200 - 03FF

Memory Connection to CPU :

» 2 x 4 Decoder : RAM select (**CS1**)

» Address line 10

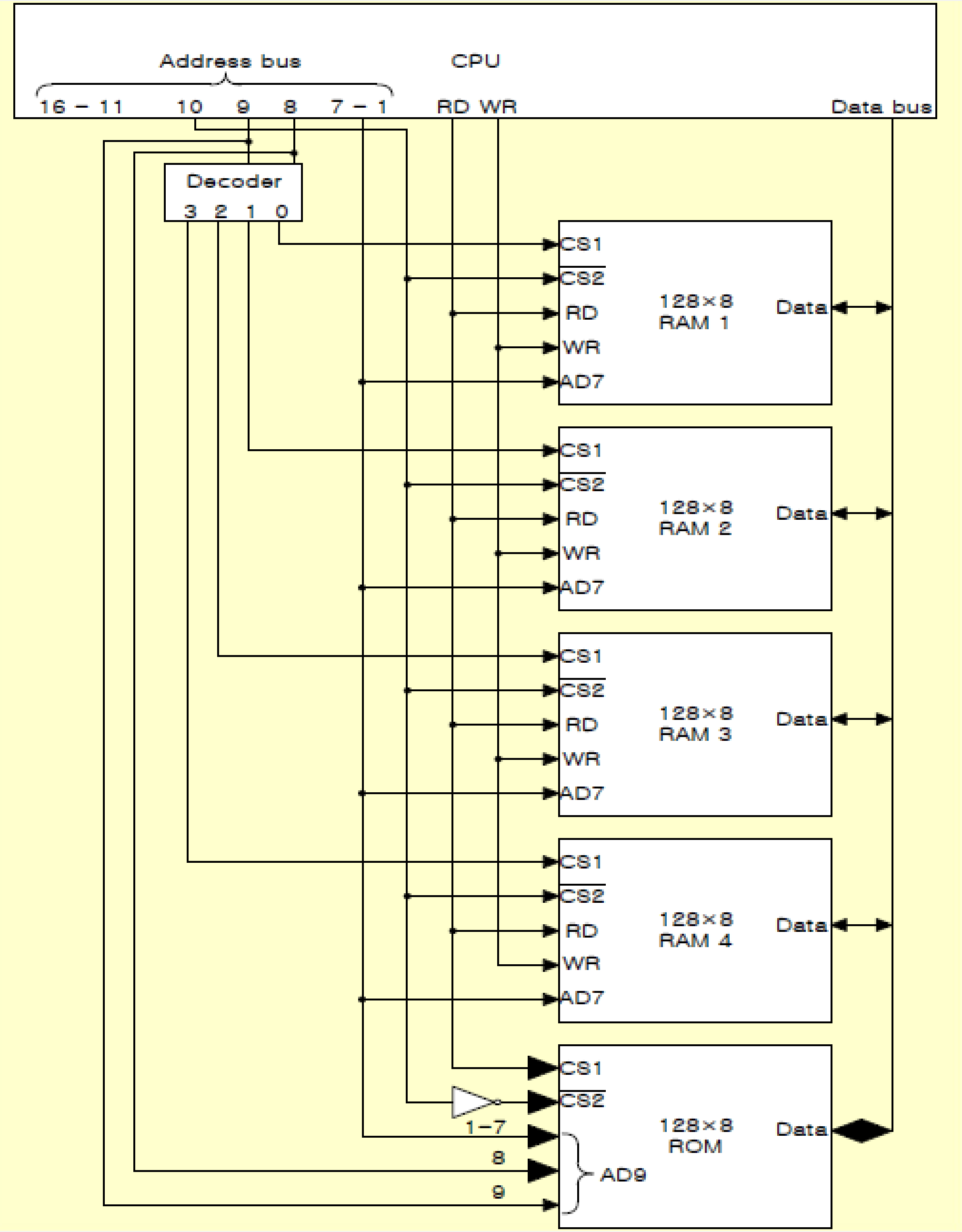
RAM select : $\overline{\textbf{CS2}}$

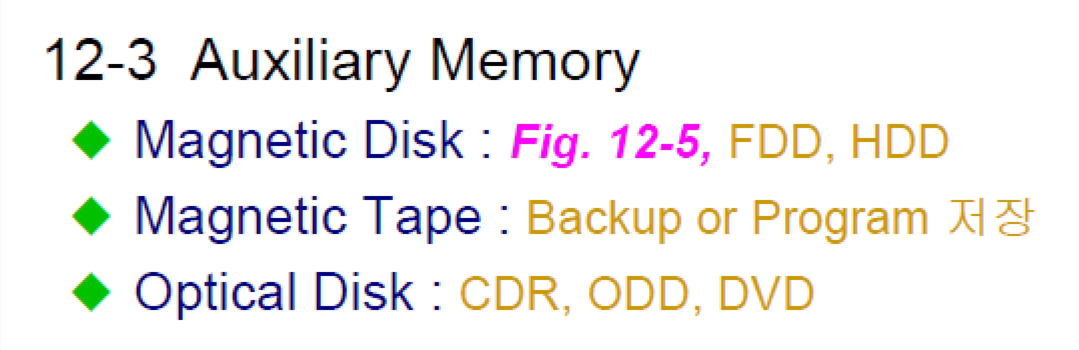
ROM select : $\overline{\textbf{CS2}}$

»

RD : ROM CS1

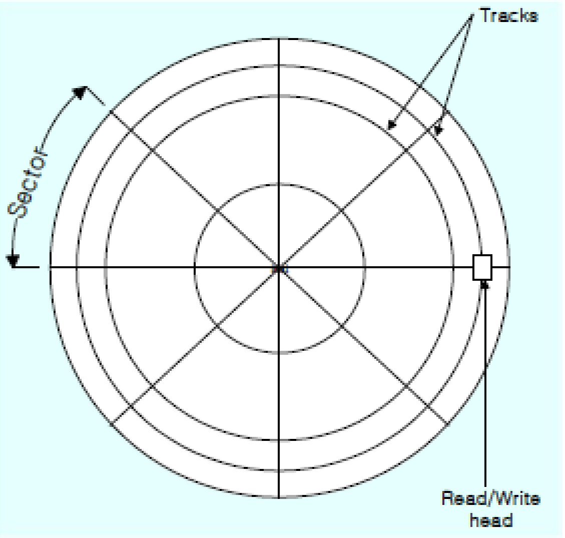
OE(Output Enable)

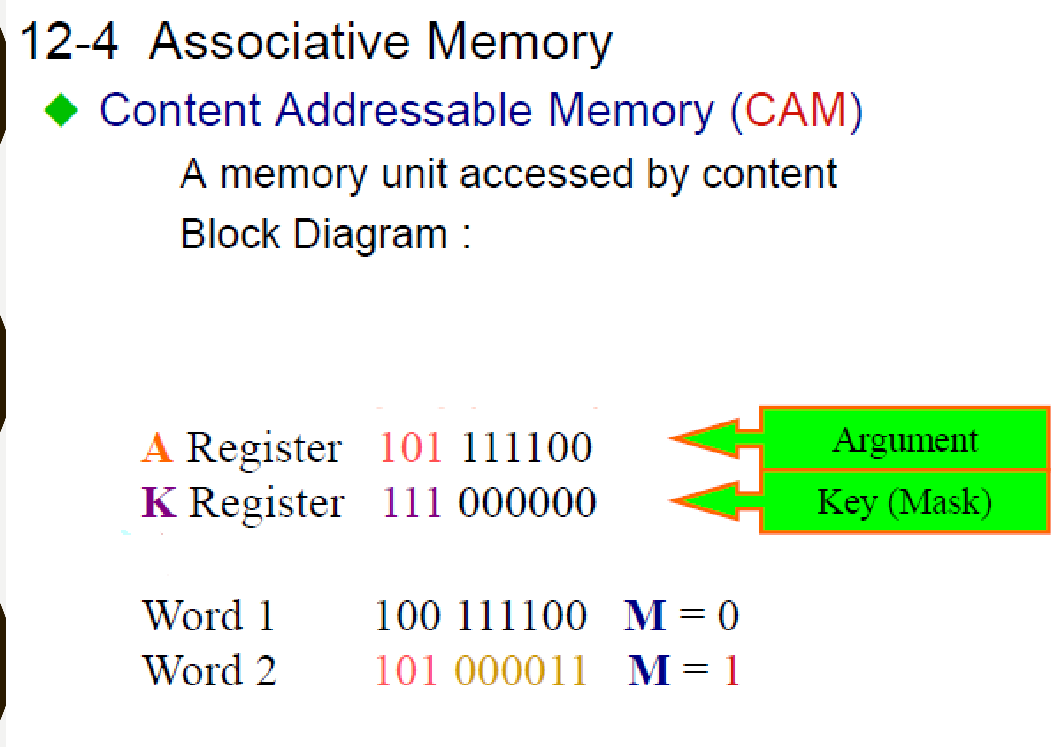

Address bus
CPU
16 – 11
10
9
8
7 – 1
RD
WR
Data bus
Decoder
3 2 1 0
CS1
CS2
RD
WR
AD7
128×8
RAM 1
Data
128×8
RAM 2
128×8
RAM 3
128×8
RAM 4
128×8
ROM
1–7
8
9
AD9



12-3 Auxiliary Memory

- Magnetic Disk : ***Fig. 12-5,*** FDD, HDD
- Magnetic Tape : Backup or Program 저장
- Optical Disk : CDR, ODD, DVD





12-4 Associative Memory

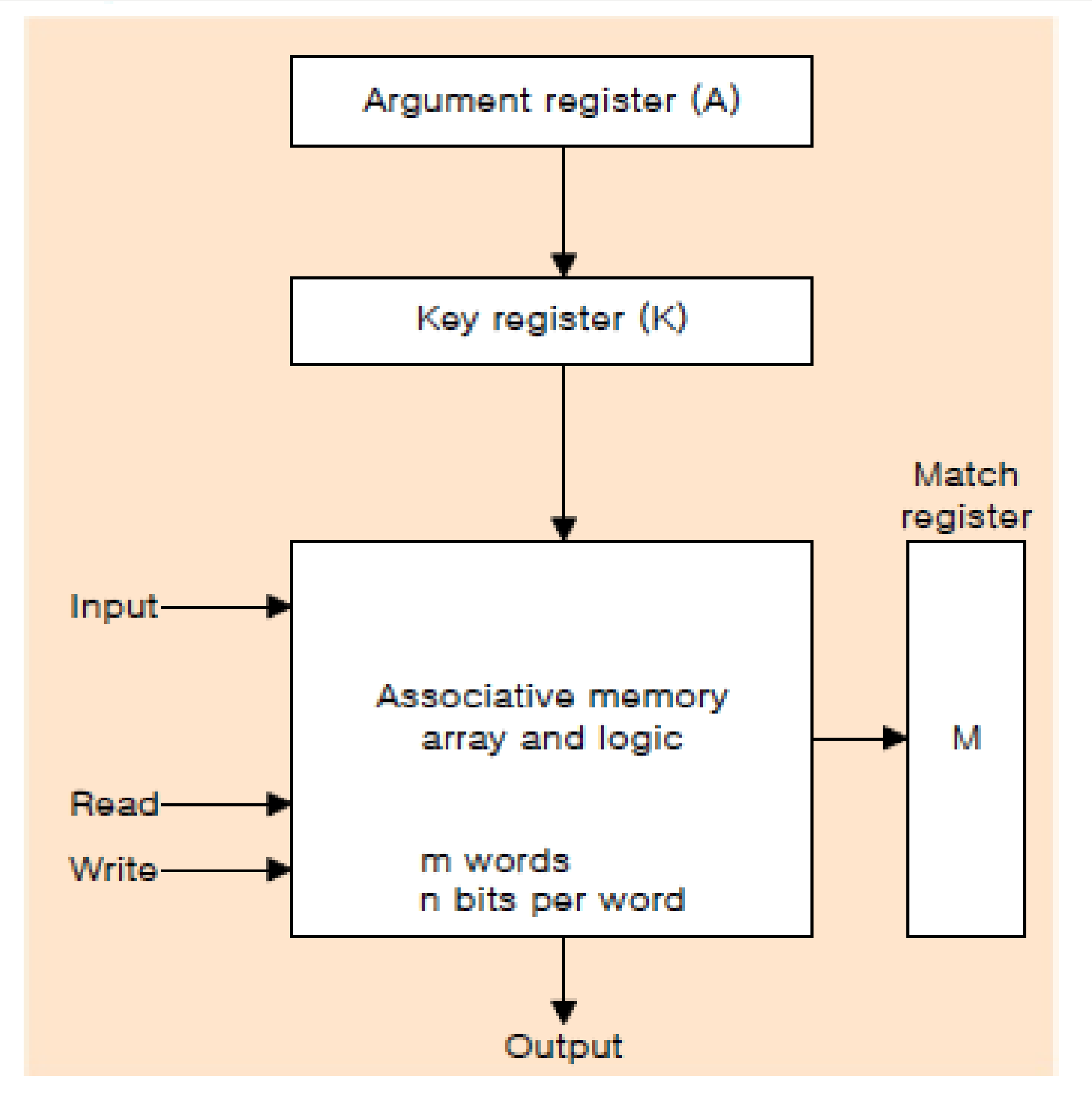
- Content Addressable Memory (CAM)

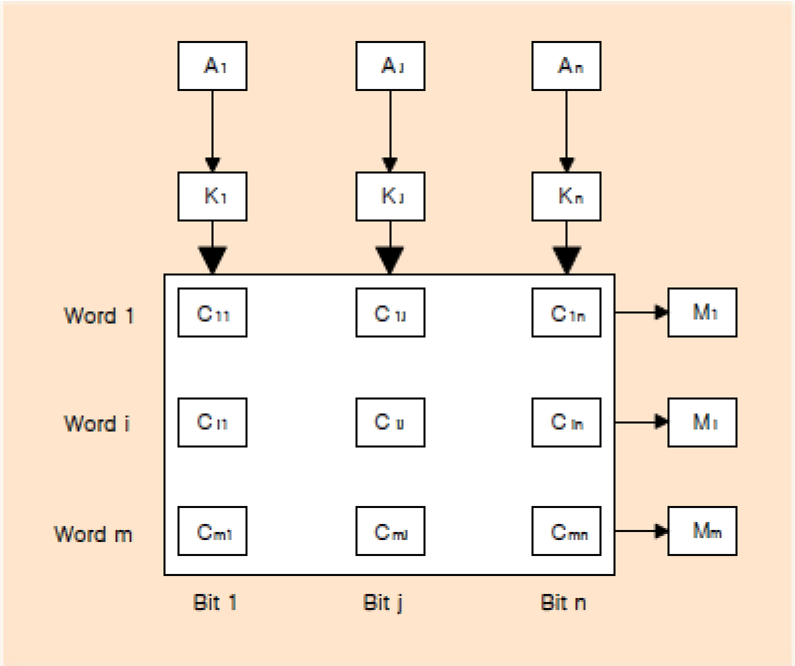
A memory unit accessed by content

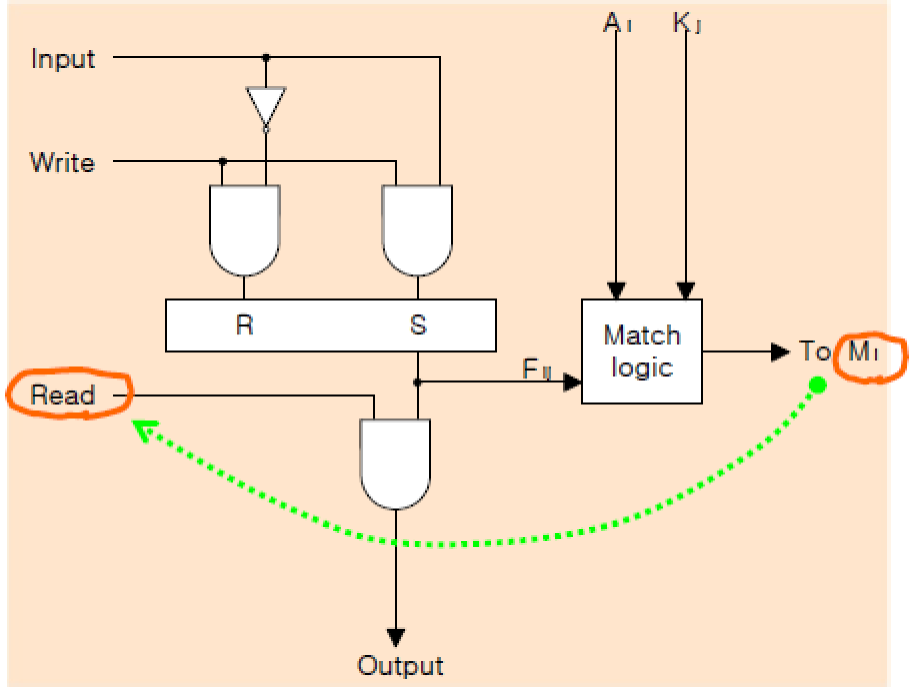
Block Diagram :

A Register	101 111100	Argument
K Register	111 000000	Key (Mask)

Word 1	100 111100	M = 0
Word 2	101 000011	M = 1




A_1
A_j
A_n
K_1
K_j
K_n
Word 1
C_{11}
C_{1j}
C_{1n}
M_1
Word i
C_{i1}
C_{ij}
C_{in}
M_i
Word m
C_{m1}
C_{mj}
C_{mn}
M_m
Bit 1
Bit j
Bit n


A_j
K_j
Input
Write
R
S
Match logic
F_{ij}
To M_i
Read
Output

12-5 Cache Memory

- **Locality of Reference**

  the references to memory *tend to be confined within a few localized areas* in memory

- **Cache Memory : a fast small memory**

  keeping the most frequently accessed instructions and data in the fast cache memory

- **Cache**

  cache size : 256 K byte (512 K byte)

  mapping method : 1) associative, 2) direct, 3) set-associative

  replace algorithm : 1) LRU, 2) LFU, 3) FIFO

  write policy : 1) write-through, 2) write-back

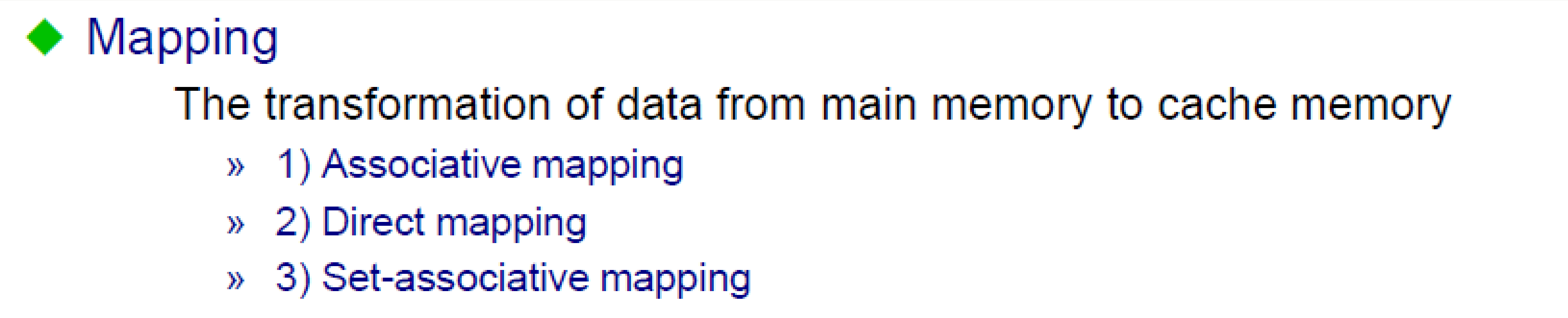
- **Hit Ratio**

  the ratio of the number of hits divided by the total CPU references (hits + misses) to memory

  - ***hit*** : the CPU finds the word in the cache (0.9)
  - ***miss*** : the word is not found in cache (CPU must read main memory)

  cache memory access time = 100 ns, main memory access time = 1000 ns, hit ratio = 0.9

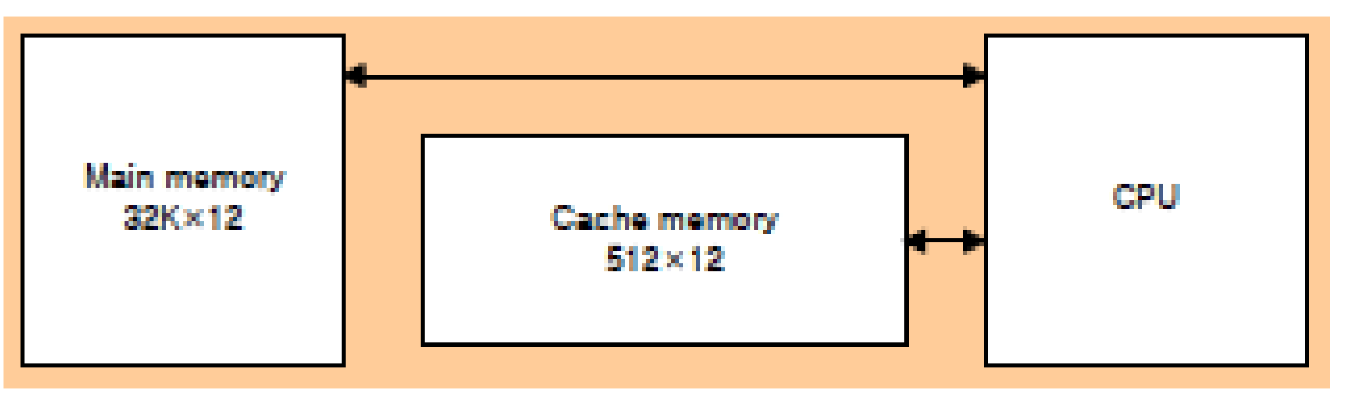
  - 1 miss : 1 x 1000 ns
  - 9 hit : 9 x 100 ns



◆ Mapping

The transformation of data from main memory to cache memory

- » 1) Associative mapping
- » 2) Direct mapping
- » 3) Set-associative mapping



- Example of cache memory :

  main memory : **32 K** x 12 bit word (15 bit address lines)
  cache memory : **512** x 12 bit word

  » CPU sends a 15-bit address to cache

  Hit : CPU accepts the 12-bit data from cache
  Miss : CPU reads the data from main memory

- Associative mapping :

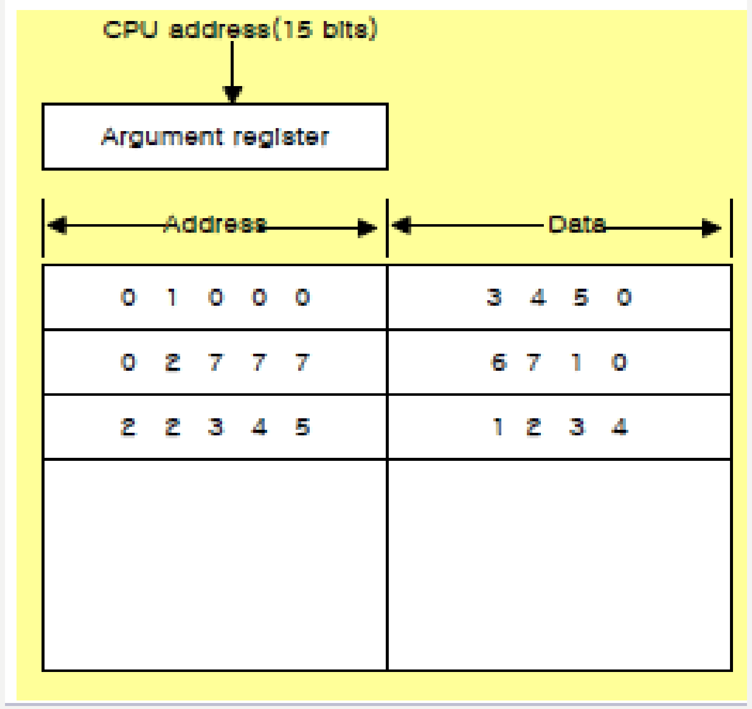
  Cache memory associative memory
  Address Data Cache memory

- Direct mapping :

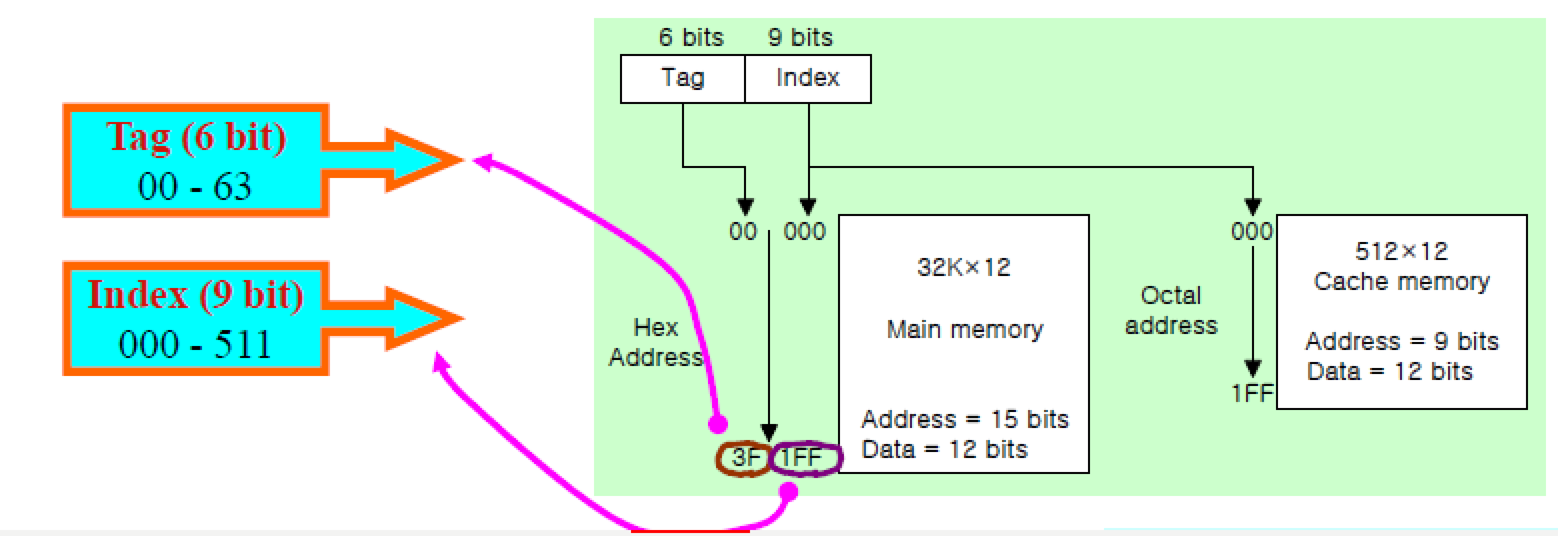
  Cache memory
  Tag field ($n - k$) Index field (k)

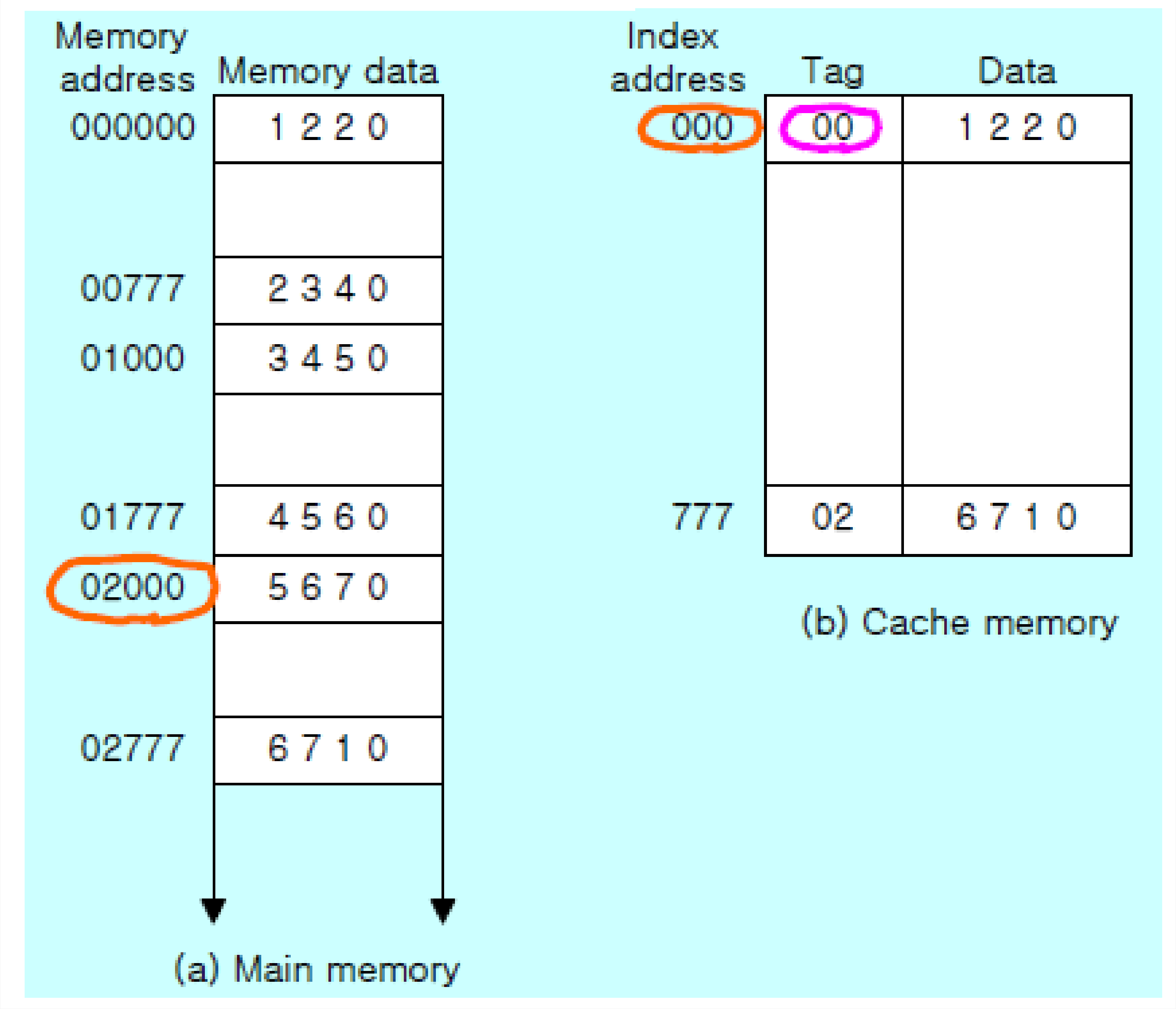
  » 2^k words cache memory + 2^n words main memory

  Tag = 6 bit (15 - 9), **Index** = 9 bit



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


Tag (6 bit)
00 - 63
Index (9 bit)
000 - 511
6 bits
9 bits
Tag
Index
00
000
3F
1FF
Hex
Address
32K×12
Main memory
Address = 15 bits
Data = 12 bits
Octal
address
000
1FF
512×12
Cache memory
Address = 9 bits
Data = 12 bits



Memory address	Memory data
000000	1 2 2 0
00777	2 3 4 0
01000	3 4 5 0
01777	4 5 6 0
02000	5 6 7 0
02777	6 7 1 0

(a) Main memory

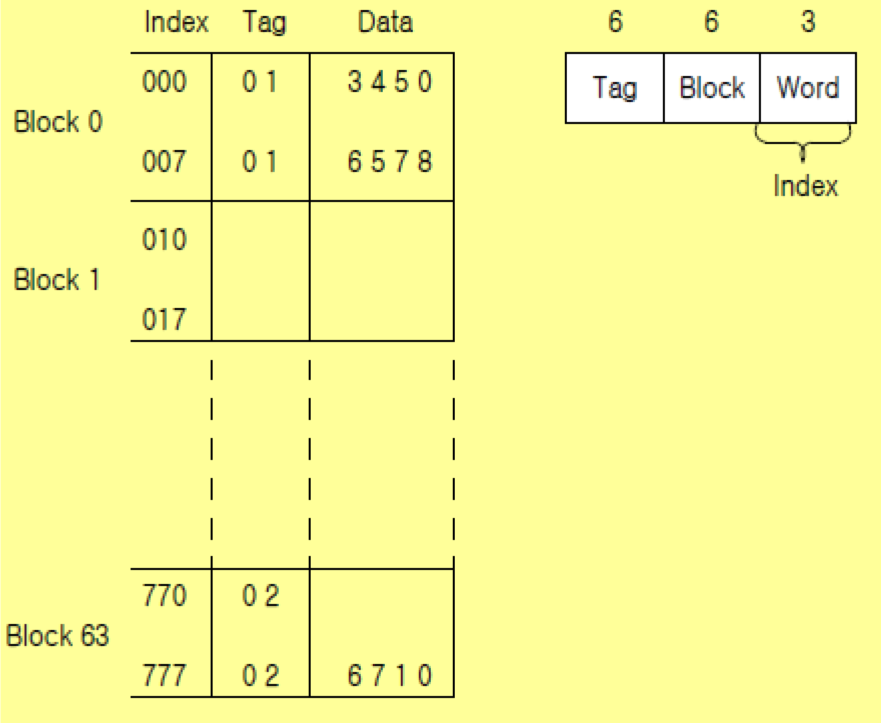
Index address	Tag	Data
000	00	1 2 2 0
777	02	6 7 1 0

(b) Cache memory

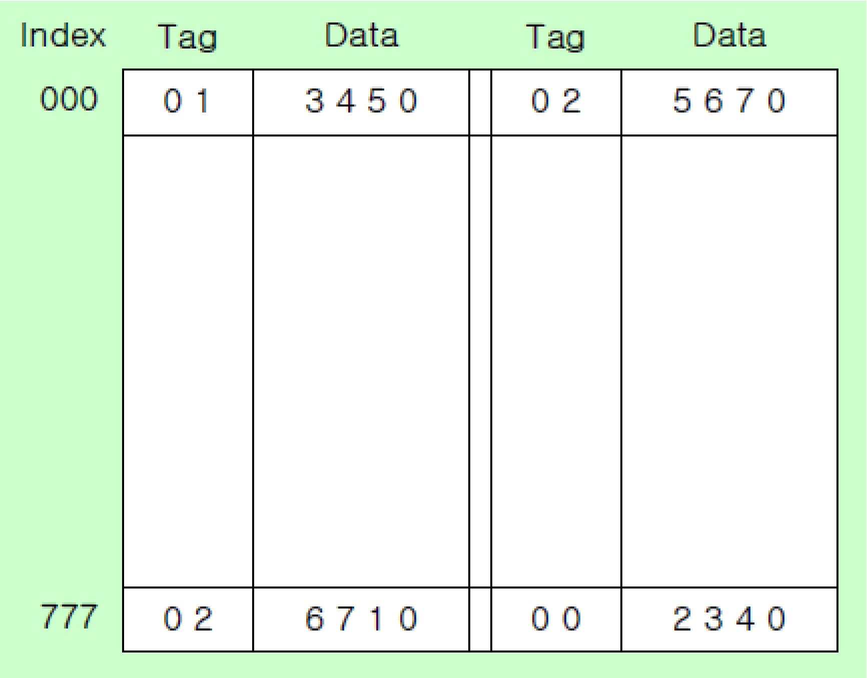
Direct mapping cache with block size of 8 words :

- 64 block x 8 word = 512 cache words size

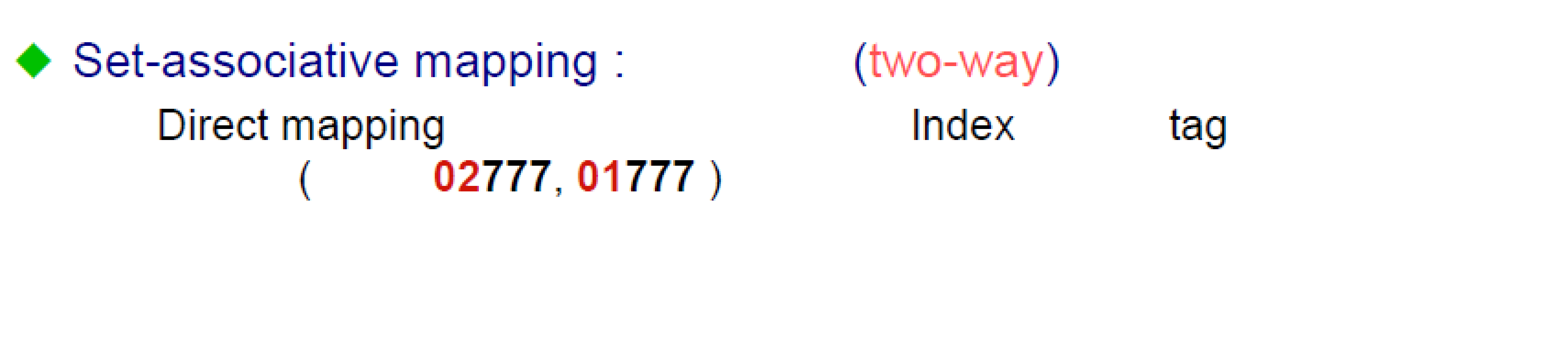
8 word block · . . update



	Index	Tag	Data
Block 0	000	01	3450
	007	01	6578
Block 1	010		
	017		
Block 63	770	02	
	777	02	6710



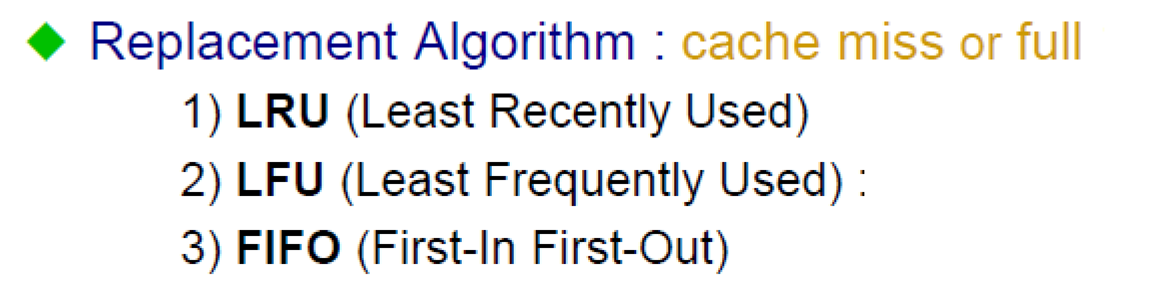
Index	Tag	Data	Tag	Data
000	01	3450	02	5670
777	02	6710	00	2340



- Set-associative mapping : (two-way)

  Direct mapping Index tag

  (**02777**, **01777**)

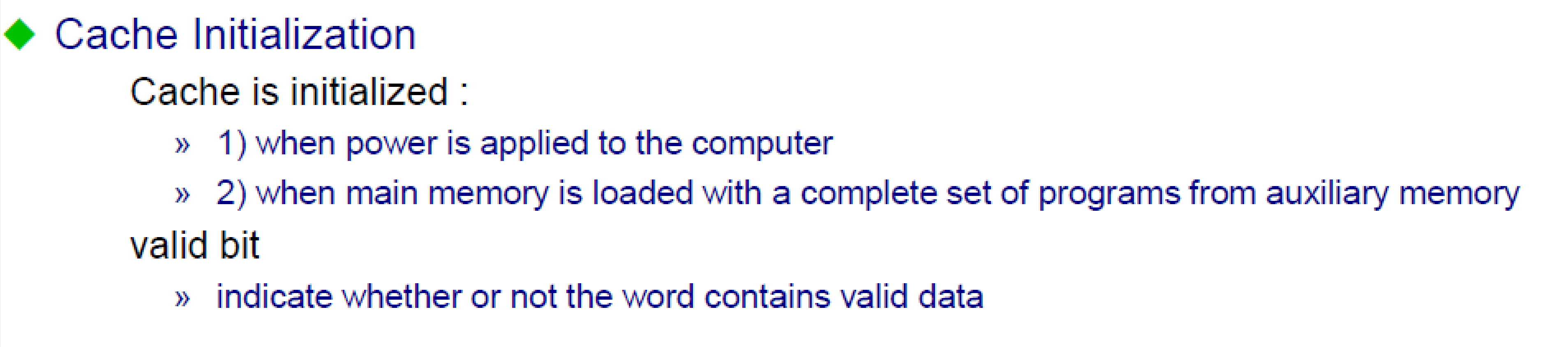


- Replacement Algorithm : cache miss or full

  1) **LRU** (Least Recently Used)

  2) **LFU** (Least Frequently Used) :

  3) **FIFO** (First-In First-Out)



- Cache Initialization

  Cache is initialized :

  » 1) when power is applied to the computer

  » 2) when main memory is loaded with a complete set of programs from auxiliary memory

  valid bit

  » indicate whether or not the word contains valid data

12-6 Virtual Memory

- Virtual Memory : Auxiliary memory ⇒ Main memory

Translate program-generated (Aux. Memory) address into main memory location

» Give programmers the illusion that they have a very large memory, even though the computer actually has a relatively small main memory

Intel Pentium Processor

» Physical Address Lines = A_0 - A_{31} : $2^{32} = 2^{30} \times 2^2$ = 4 Giga

» Logical Address = 46 bits address : $2^{46} = 2^{40} \times 2^6$ = 64 Tera

- Address Space & Memory Space

Address Space : ***Virtual Address***

» Address used by a programmer

Memory Space : ***Physical Address***(Location)

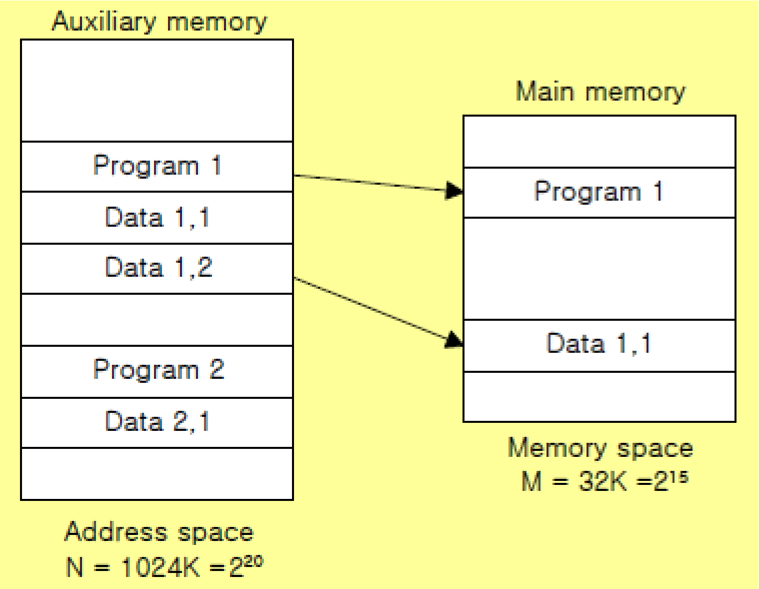
» Address in main memory

address space (**N**) = 1024 K = 2^{20}

» Auxiliary Memory

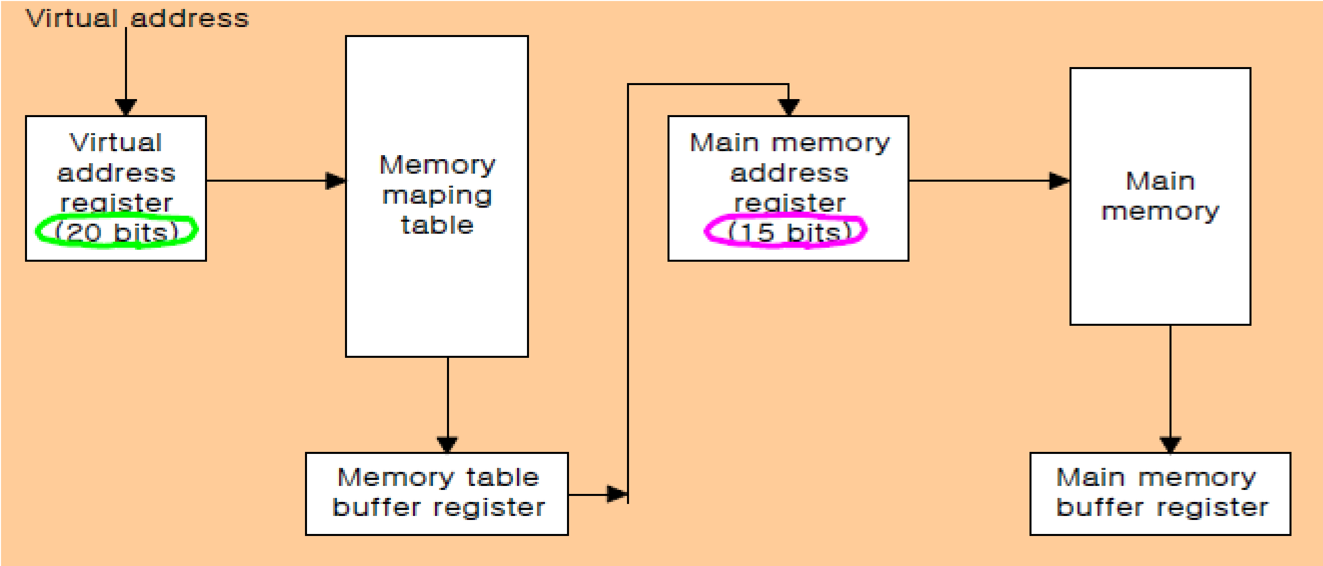
memory space (**M**) = 32 K = 2^{15}

» main Memory



- Memory table for mapping a virtual address :

  Translate the *20 bits Virtual address* into the *15 bits Physical address*


Virtual address
Virtual address register (20 bits)
Memory maping table
Memory table buffer register
Main memory address register (15 bits)
Main memory
Main memory buffer register

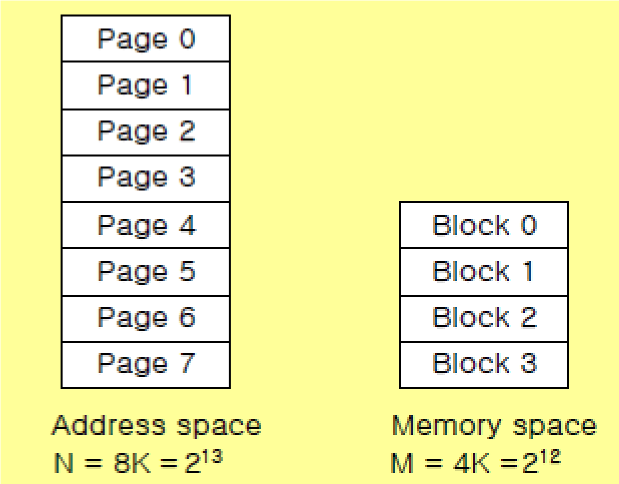
- Address Mapping Using Pages :

  Address mapping

  » Address space memory space fixed size

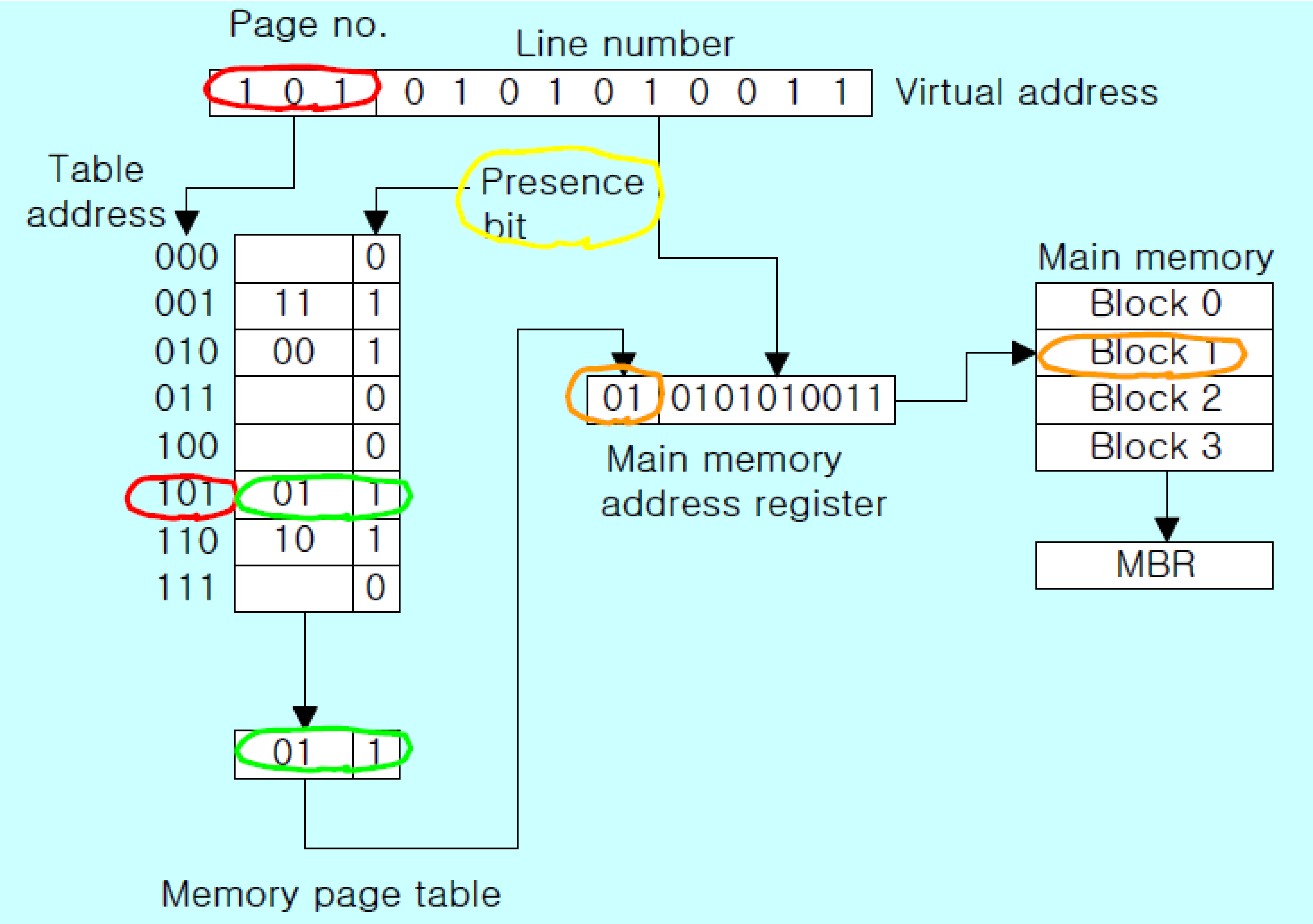
  Address space : 1 K page
  Memory space : 1 k block

  » Address space 4 page memory space block



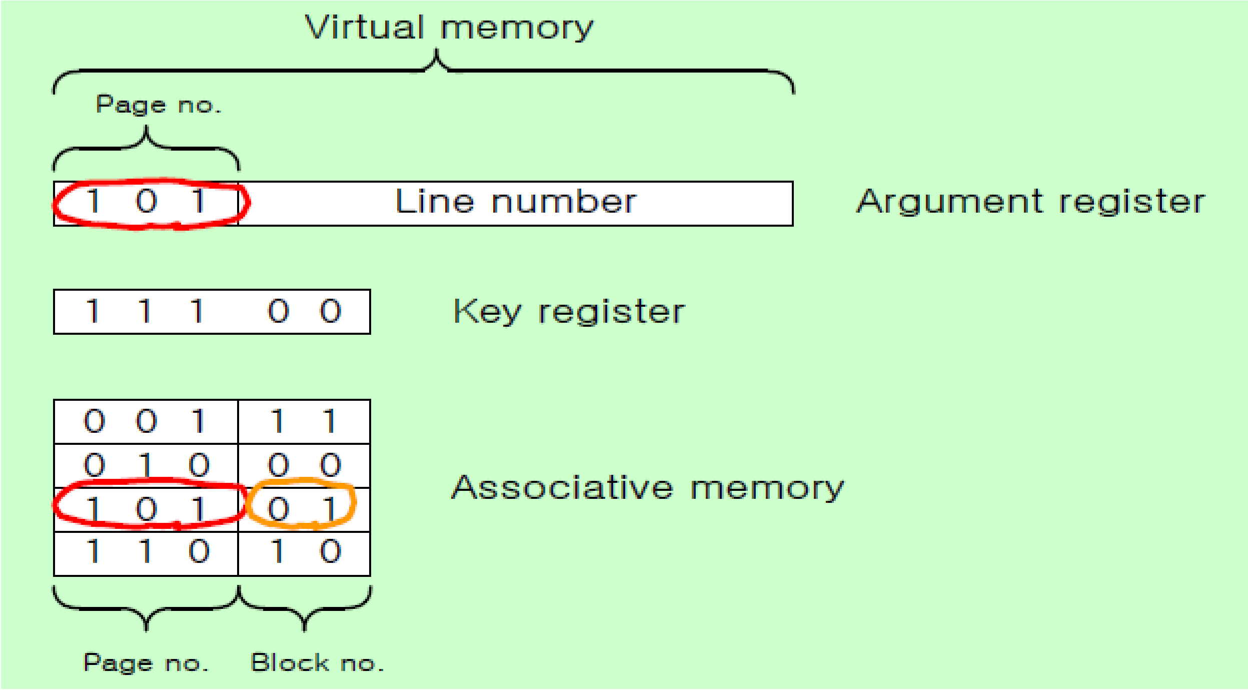
◆ Memory table in a paged system :

Virtual address → Memory address



◆ Associative memory page table :

Associative memory   block number(**01**)



Argument register: 1 0 1 | Line number

Key register: 1 1 1 0 0

Page no.	Block no.
0 0 1	1 1
0 1 0	0 0
1 0 1	0 1
1 1 0	1 0

◆ Page(Block) Replacement

Page Fault : the page referenced by the CPU is not in main memory

- » a new page should be transferred from auxiliary memory to main memory

Replacement algorithm : FIFO   LRU

REFERENCE

- *Mano, M. Morris (October 1992). Computer System Architecture (3rd ed.). Prentice-Hall. ISBN 0-13-175563-3*
- *Lecture notes of* Dept. of Info. & Comm., Korea Univ. of Tech. & Edu., Korea