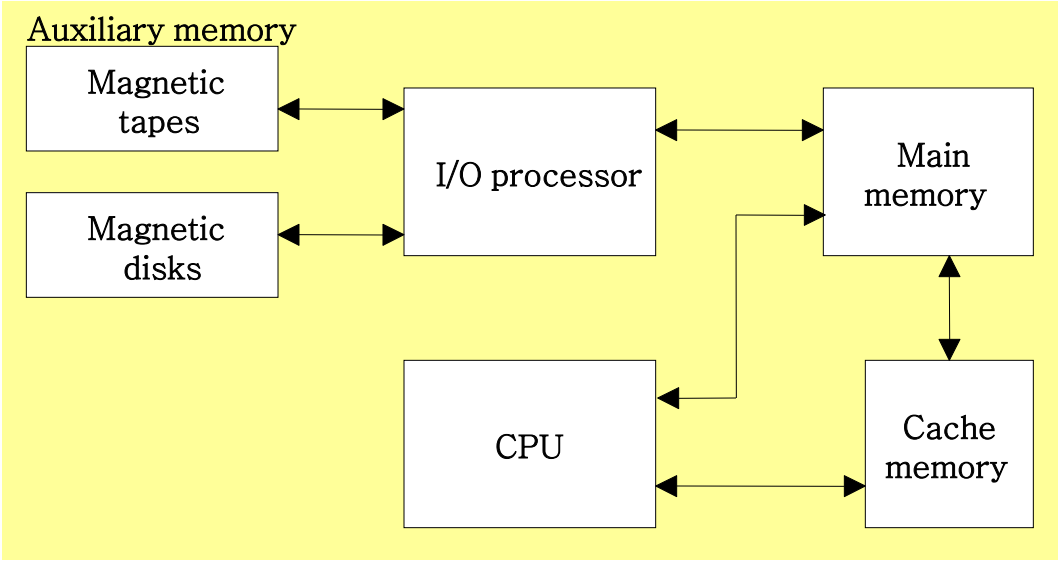
Introduction: Memory Organization (Memory Hierarchy)

- Memory hierarchy in a computer system : ***Fig. 12-1***
  - **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 by making current programs and data available to the CPU at a very fast rate .
    - » It is used to compensate the speed difference between main memory access time and processor logic.
    - » While the I/O processor manages data transfer between auxiliary memory and main memory, the cache memory is concerned with the data transfer between main memory and CPU.



◆Multiprogramming : Enables the CPU to process a number of independent program concurrently.

- ✓Multiprogramming refers to the existence of two or more programs in different parts of the memory hierarchy at the same time.
- ✓It is possible to keep all parts of CPU busy by working with several programs in sequence.
- ✓In multiprogramming environment when one program is waiting for input or output transfer there is another program ready to utilize the CPU.

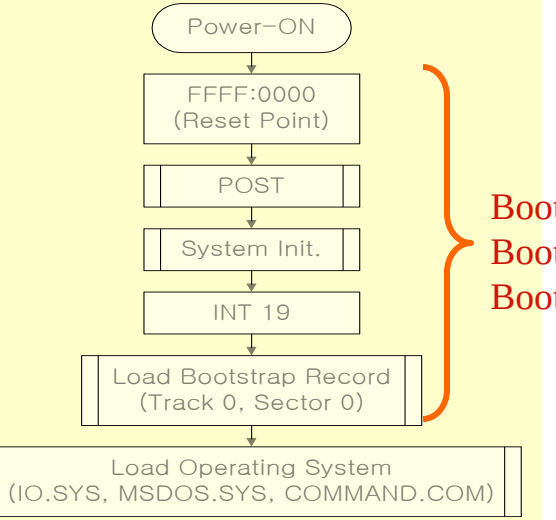
12-2 Main Memory

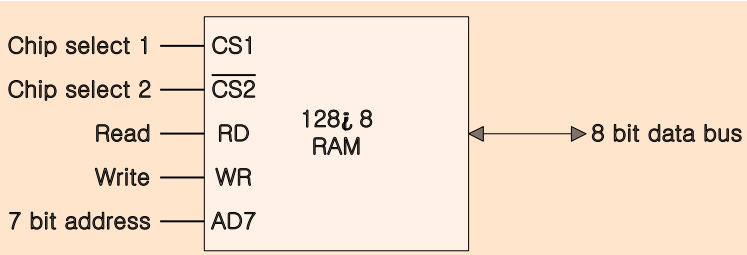
Bootstrap Loader

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

RAM and ROM Chips

- Typical RAM chip : ***Fig. 12-2***
  - » 128 X 8 RAM : 2^7 = 128 (7 bit address lines)
- Typical ROM chip : ***Fig. 12-3***
  - » 512 X 8 ROM : 2^9 = 512 (9 bit address lines)

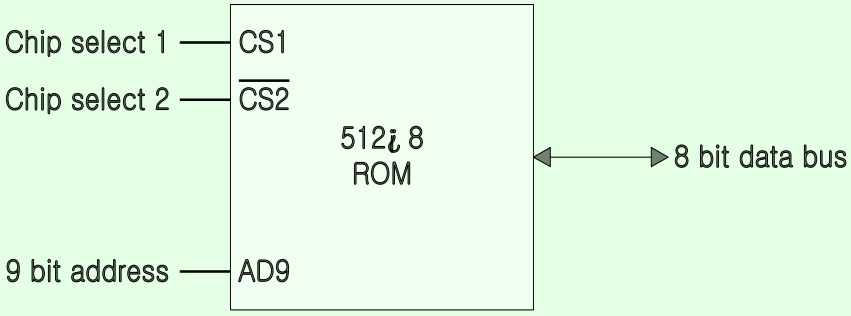




(a) Block diagram

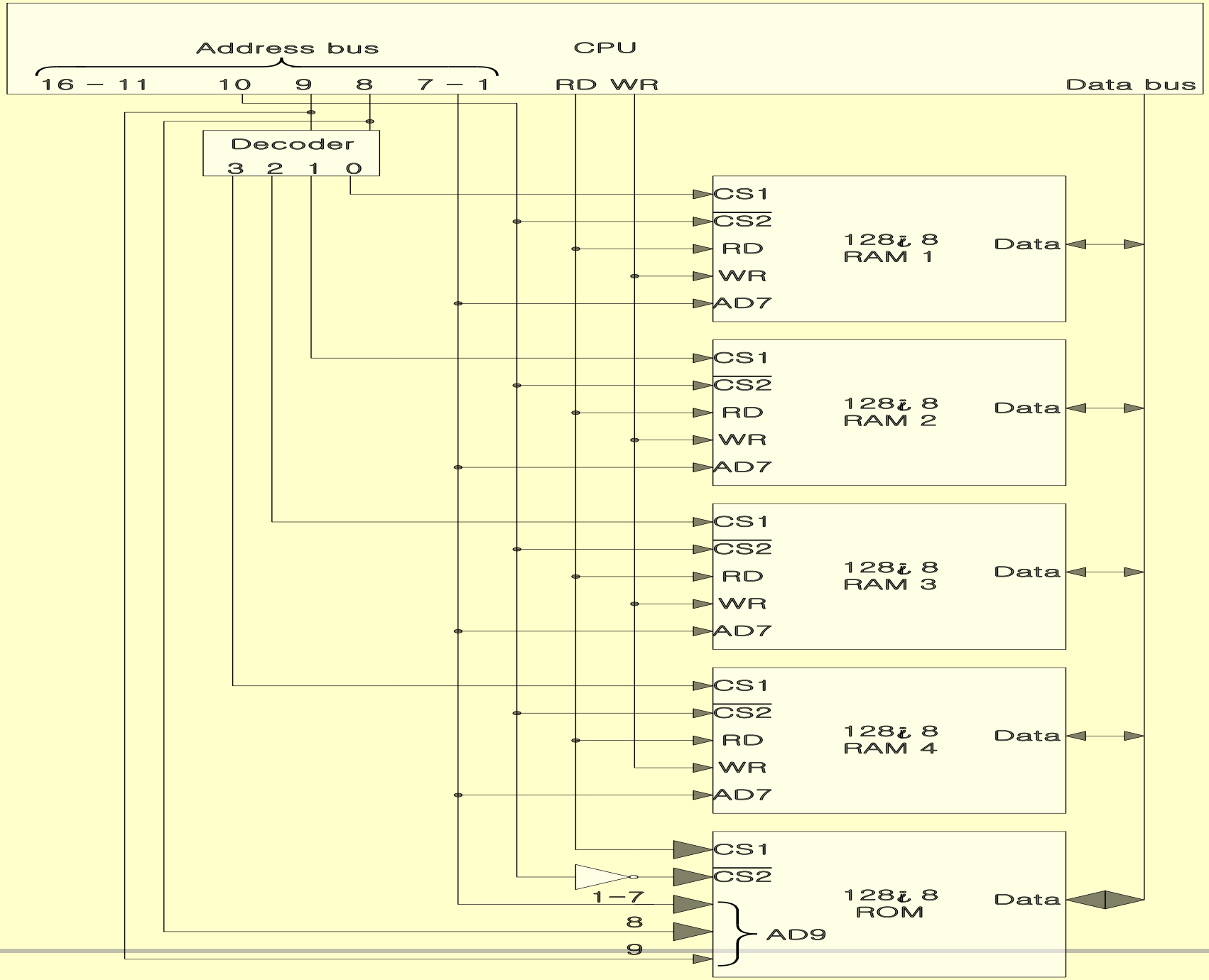
CS1	CS2	RD	WR	Memory function	State of data bus
0	0	¿	¿	Inhibit	High-impedance
0	1	¿	¿	Inhibit	High-impedance
1	0	0	0	Inhibit	High-impedance
1	0	0	1	Write	Input data to RAM
1	0	1	¿	Read	Output data from RAM
1	1	¿	¿	Inhibit	High-impedance

(b) Function table

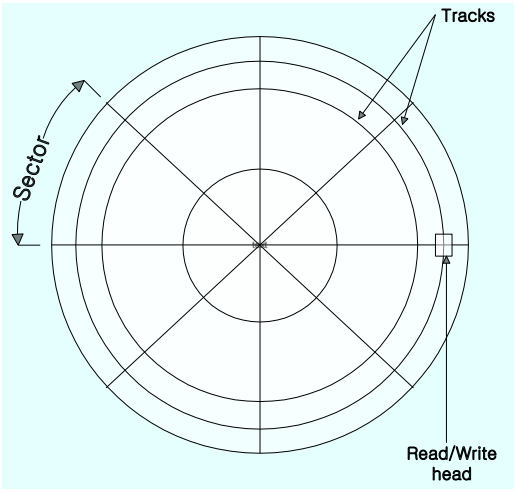


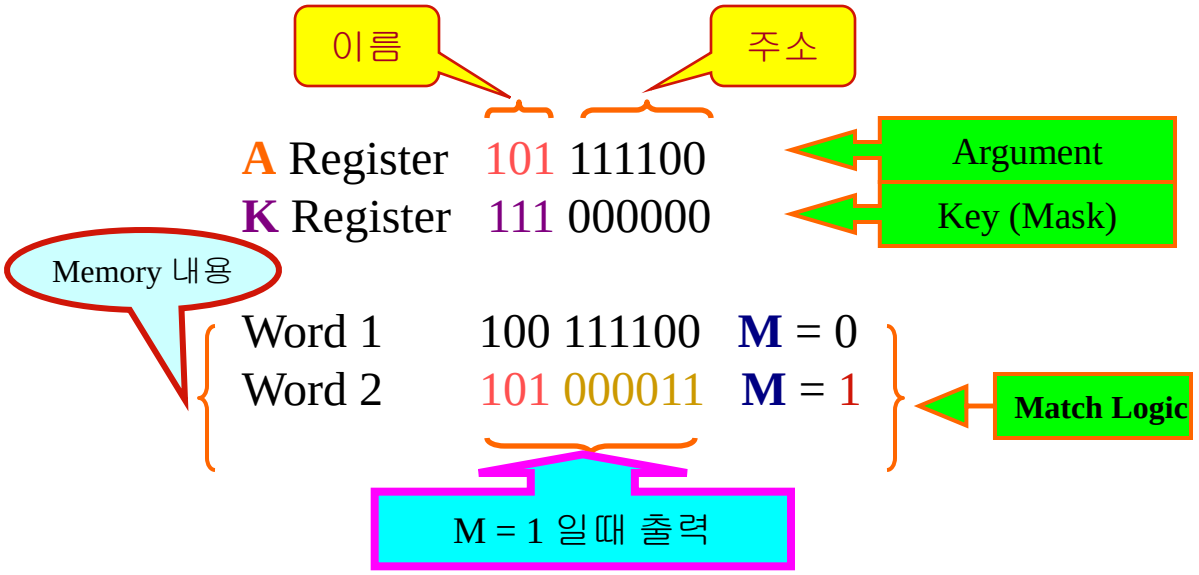
Memory Address Map

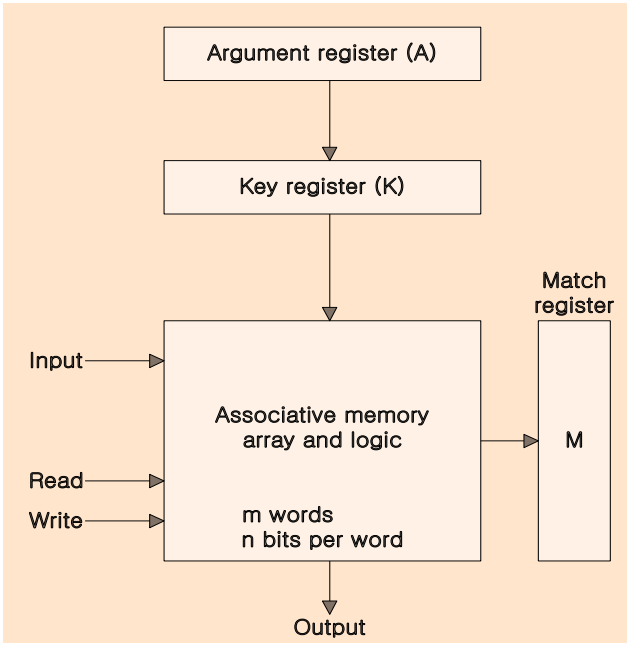
- Memory Configuration : **512** bytes **RAM** + **512** bytes **ROM**
  - » **1** x **512** byte **ROM** + **4** x **128** bytes RAM
- **Memory Address Map :** it is a pictorial representation of assigned address space for each chip in the system.
  - » Address line **10 9 8 7 6 5 4 3 2 1**
    - RAM 1 0 **0** **0 x x x** **x x x x** : 0000 - 007F
    - RAM 1 0 **0** **1 x x x** **x x x x** : 0080 - 00FF
    - RAM 1 0 **1** **0 x x x** **x x x x** : 0100 - 017F
    - RAM 1 0 **1** **1 x x x** **x x x x** : 0180 - 01FF
    - ROM **1** **x** **x x x x** **x x x x** : 0200 - 03FF
    - The small x's under the address bus lines, designate those lines that must be connected to the address input in each chip.
    - RAM chips have 128 bytes and needs 7 address lines.
    - ROM chip have 512 bytes and needs 9 address lines
- Memory Connection to CPU :
- 2 x 4 Decoder : RAM select (**CS1**)
  - » Address line **10**
    - RAM select : ~~**CS2**~~
    - ROM select : ~~**CS2**~~ 의 **Invert**
    - RD : ROM CS1
    - OE(Output Enable)


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

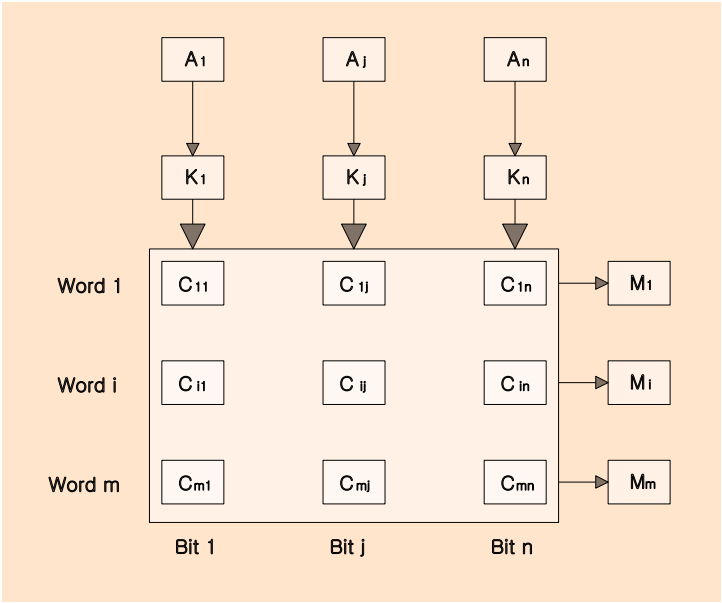
- 12-3 Auxiliary Memory
  - Magnetic Disk : FDD, HDD
  - Magnetic Tape : Backup or Program
  - Optical Disk : CDR, ODD, DVD
- 12-4 Associative Memory
  - Content Addressable Memory (CAM)
    - A memory unit accessed by content
    - Block Diagram : ***Fig. 12-6***

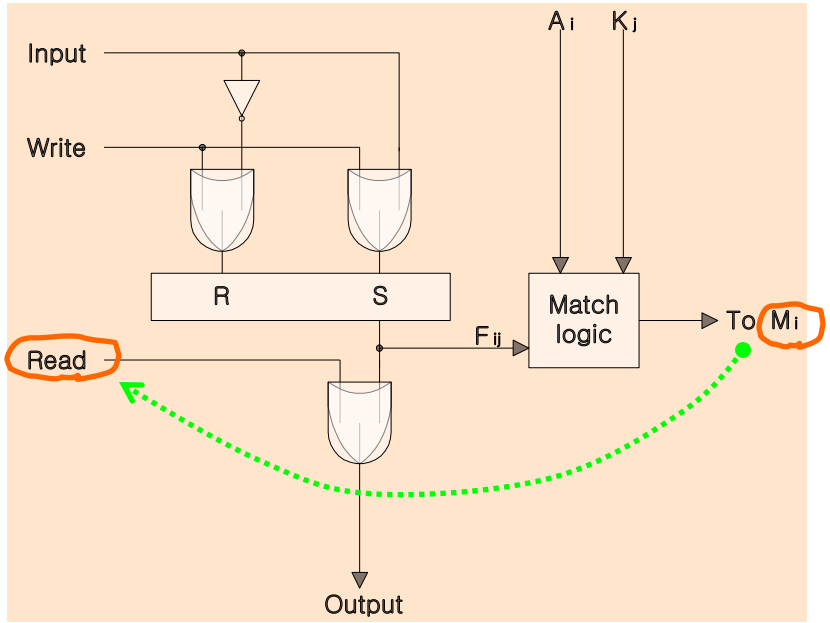






- m word x n cells per word : ***Fig. 12-7***





- Match Logic
  - One cell of associative memory : ***Fig. 12-8***
    - » Input = 1 or 0 Write F/F
    - » A 와 K 에 의해 Match Logic 에서 M=1 이면 (M을 READ에 직접 연결 가능함)
    - » Read 신호에 따라 F/F에서 데이터를 읽는다

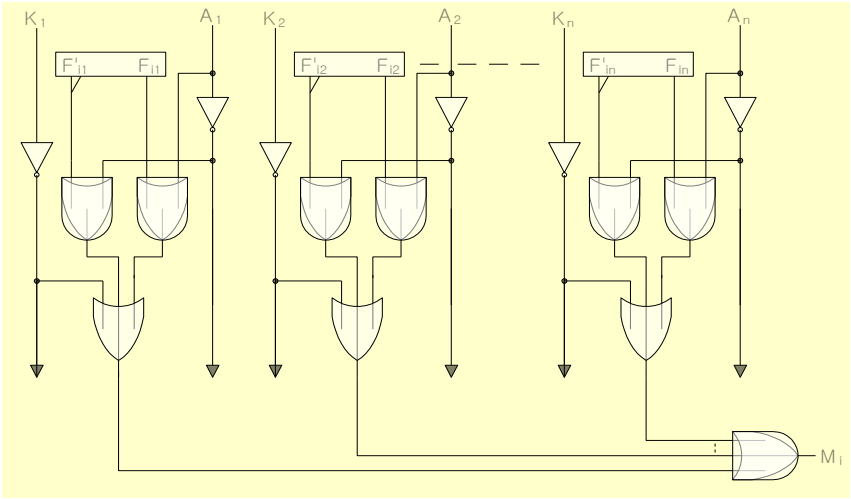
- Match Logic : ***Fig. 12-9***
  - » A_j = Argument, F_{ij} = Cell ij bit
  - » j 1 bit match

    $x_j = A_j F_{ij}$ (1 AND 1)+ $A_j' F_{ij}'$ (0 AND 0)
  - » 1 - n , n bits match $M_i = x_1 x_2 \ldots . x_n$
  - » Key bit K_j : $x_j + K_j'$
    - $K_j = 0$: A_j 와 F_{ij} 는 **no comparison** ($K_j : x_j + 1 = 1$)
    - $K_j = 1$: A_j 와 F_{ij} 는 **comparison** ($K_j : x_j + 0 = x_j$)
  - » Match Logic for word I :

    $M_i = (x_1 + K_1')(x_2 + K_2') \ldots . (x_n + K_n')$

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

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



12-5 Cache Memory

Locality of Reference

- the references to memory *tend to be confined within a few localized areas* in memory. Ex. Program loops or subroutine. It states that over a short interval of time the addresses generated by a typical program refers to a few localized areas of memory repeatedly.

Cache Memory : **a fast small memory**

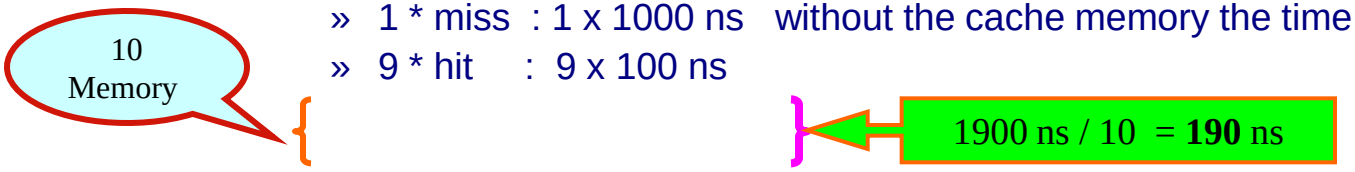
- keeping the most frequently accessed instructions and data in the fast cache memory

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
  - » ***miss*** : the word is not found in cache (CPU must read main memory)
- An example where cache memory access time = 100 ns, main memory access time = 1000 ns, hit ratio = 0.9 produces an average access time of 200 ns.
  - » 1 * miss : 1 x 1000 ns without the cache memory the time is 1000ns
  - » 9 * hit : 9 x 100 ns



Mapping

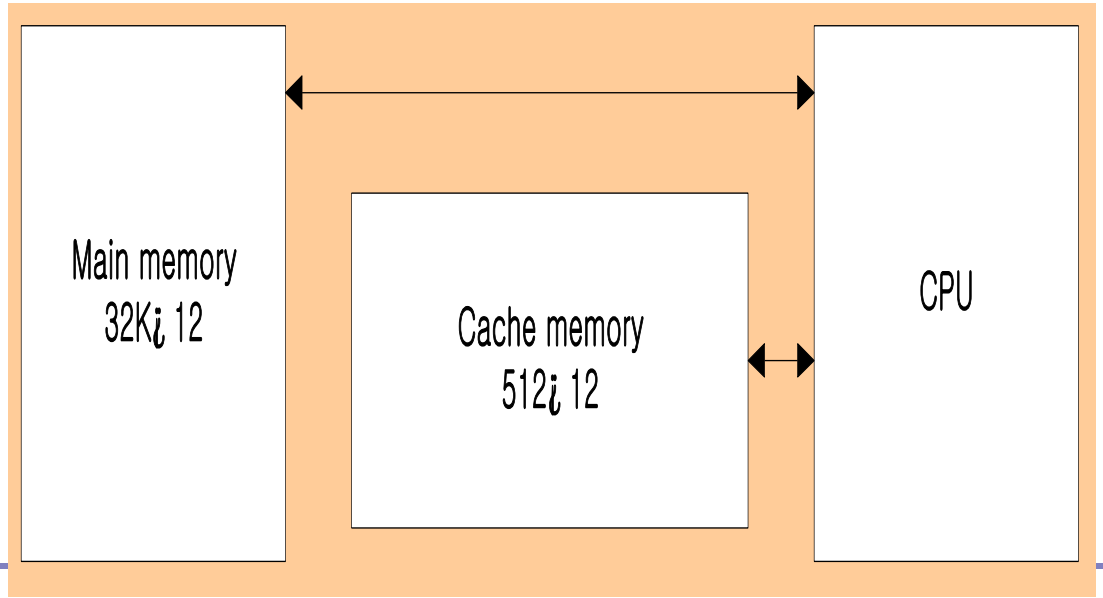
- The transformation of data from main memory to cache memory
  - » 1) Associative mapping
  - » 2) Direct mapping
  - » 3) Set-associative mapping

Example of cache memory :

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

cache memory : **512** x 12 bit word

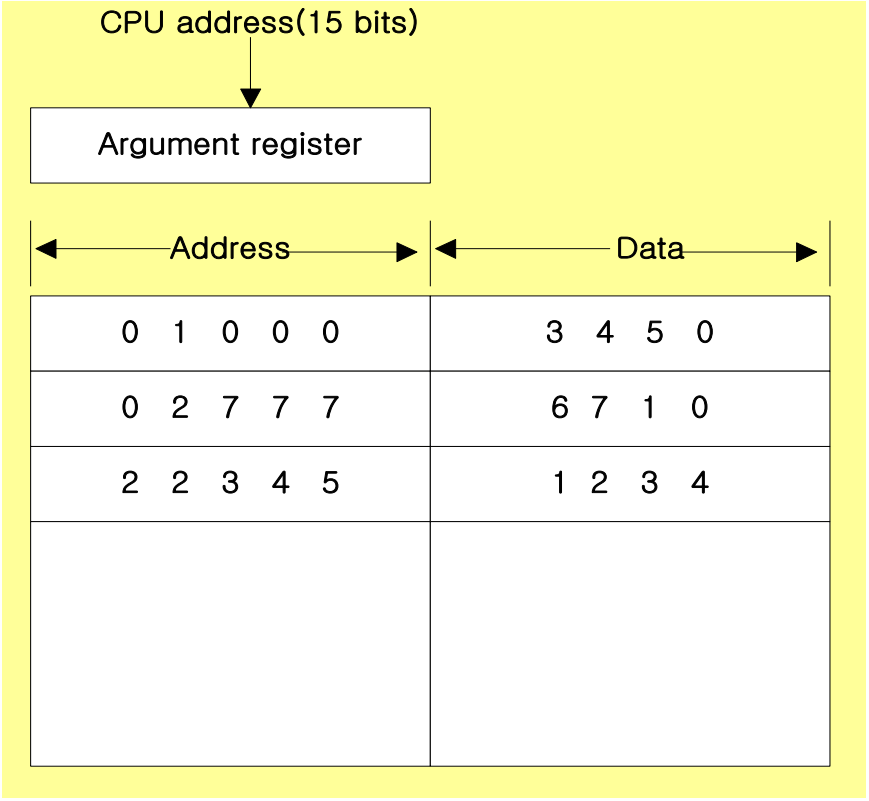
- » CPU sends a 15-bit address to cache
  - **Hit** : CPU accepts the 12-bit data from cache
  - **Miss** : CPU reads the data from main memory (then data is written to cache)



◆Associative mapping : associative memory stores both address and data of the memory word.

If the address is found, the corresponding 12-bit data is read and send to the CPU. IF NO MATCH OCCURS, then main memory is accessed for the word. The address pair is then transferred to the associative memory. If the cache is full, an address-data pair must be displaced to make room for a pair that is needed and not presently is in cache.

This is done with replacement algorithm.



◆Direct mapping : *Fig. 12-12*

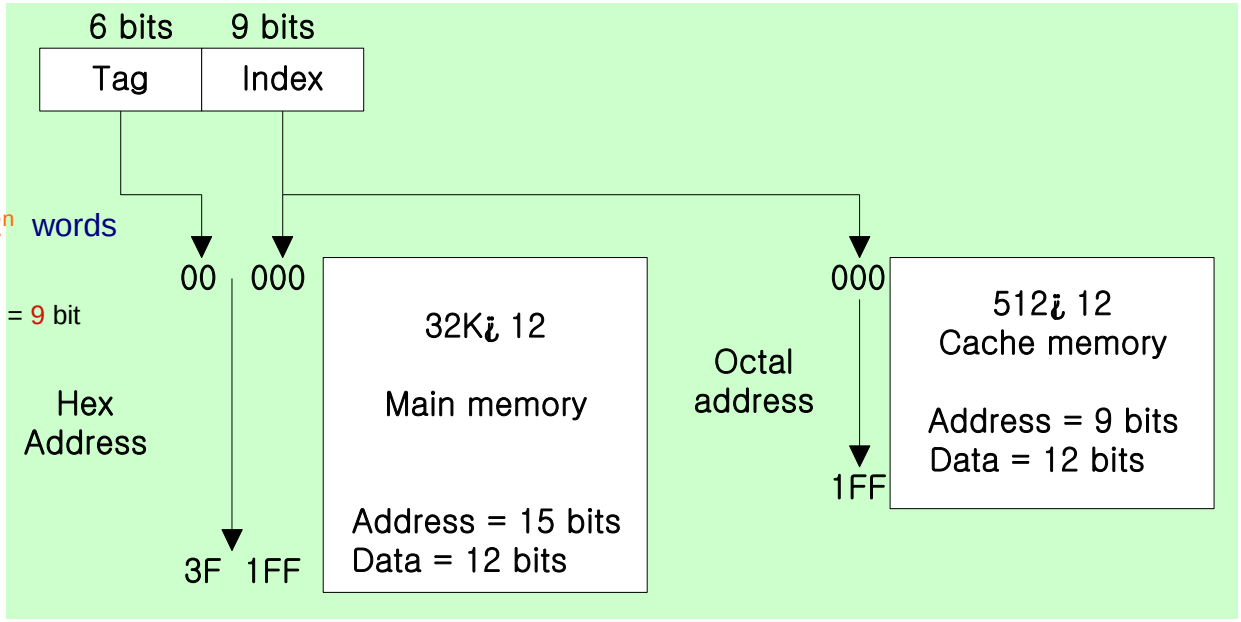
- ●Cache memory
- ●Tag field (**n - k**)
- ●Index field (**k**)

»2^k words cache memory **and** 2^n words main memory

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

Tag (6 bit)
00 - 63

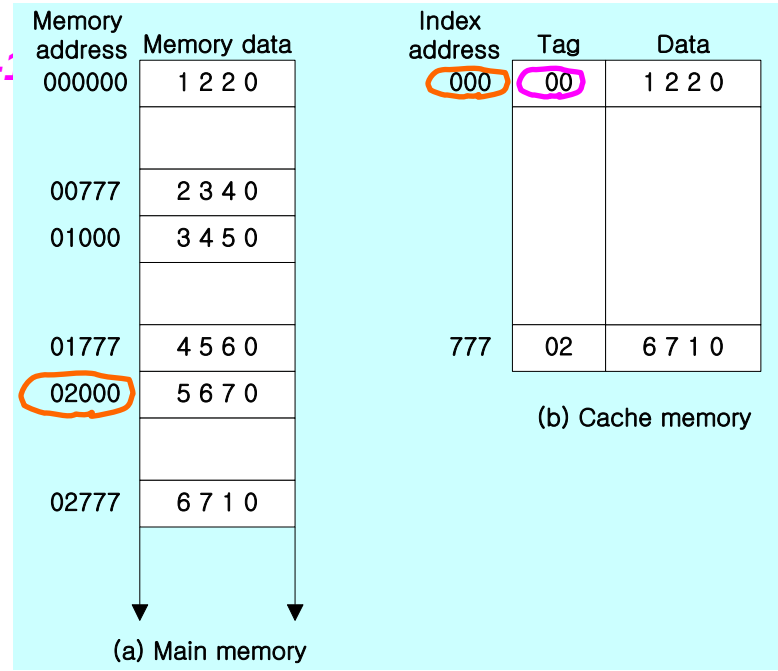
Index (9 bit)
000 - 511



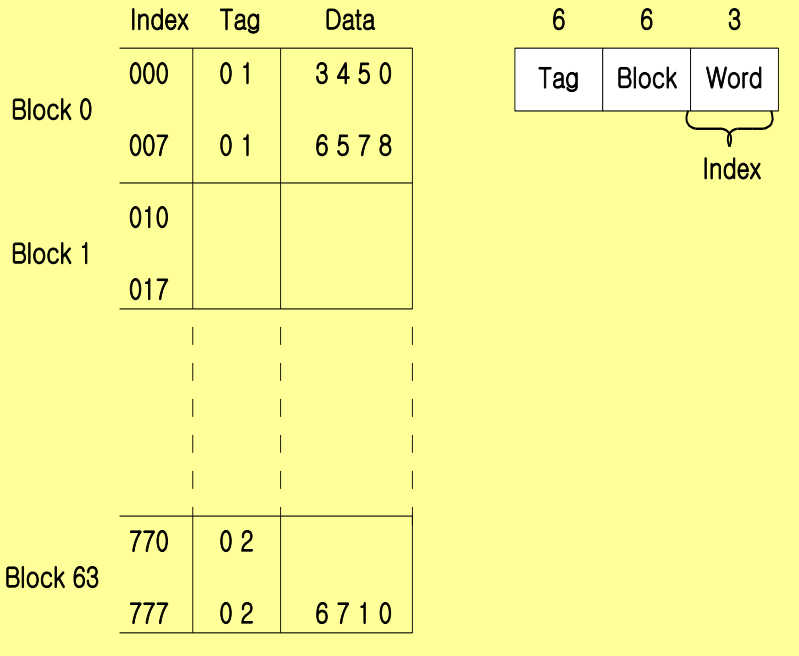
- Direct mapping cache organization : *Fig. 12-1*

For address **02000**

1) Index 000 cache , tag 00 and data 1220
2) Suppose CPU wants to access the word at address 02000.
3) The index address is 000 so it is used to Access cache. Two tags then compared.
4)Cache tag 00 but address tag 02, not match
5)Main m/m accessed & data word 5670 is Transferred to CPU.
6)Now 000 is replaced with tag 02 & data 5670.



- Direct mapping cache with block size of 8 words : ***Fig. 12-14***
  - » 64 block x 8 word = 512 cache words size



Set-associative mapping :

Disadvantage of direct mapping: two words with the same index in their address but with different tag values can not reside in cache memory at the same time.

Each data word is stored together with its tag and the number of tag-data item in one word of cache is said to form a set.

Index	Tag	Data	Tag	Data
000	0 1	3 4 5 0	0 2	5 6 7 0
777	0 2	6 7 1 0	0 0	2 3 4 0

- Replacement Algorithm : cache miss or full
  - 1) **LRU** (Least Recently Used) :
  - 2) **LFU** (Least Frequently Used)
  - 3) **FIFO** (First-In First-Out) :
- Writing to Cache:

Cache **READ**

Write Through: Advantage that main m/m always contain same data as cache.

Write back: only the cache location is updated during a Write operation.

- 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

Assignment

- Explain memory hierarchy with the help of an example.
- Differentiate between cache memory and main memory.