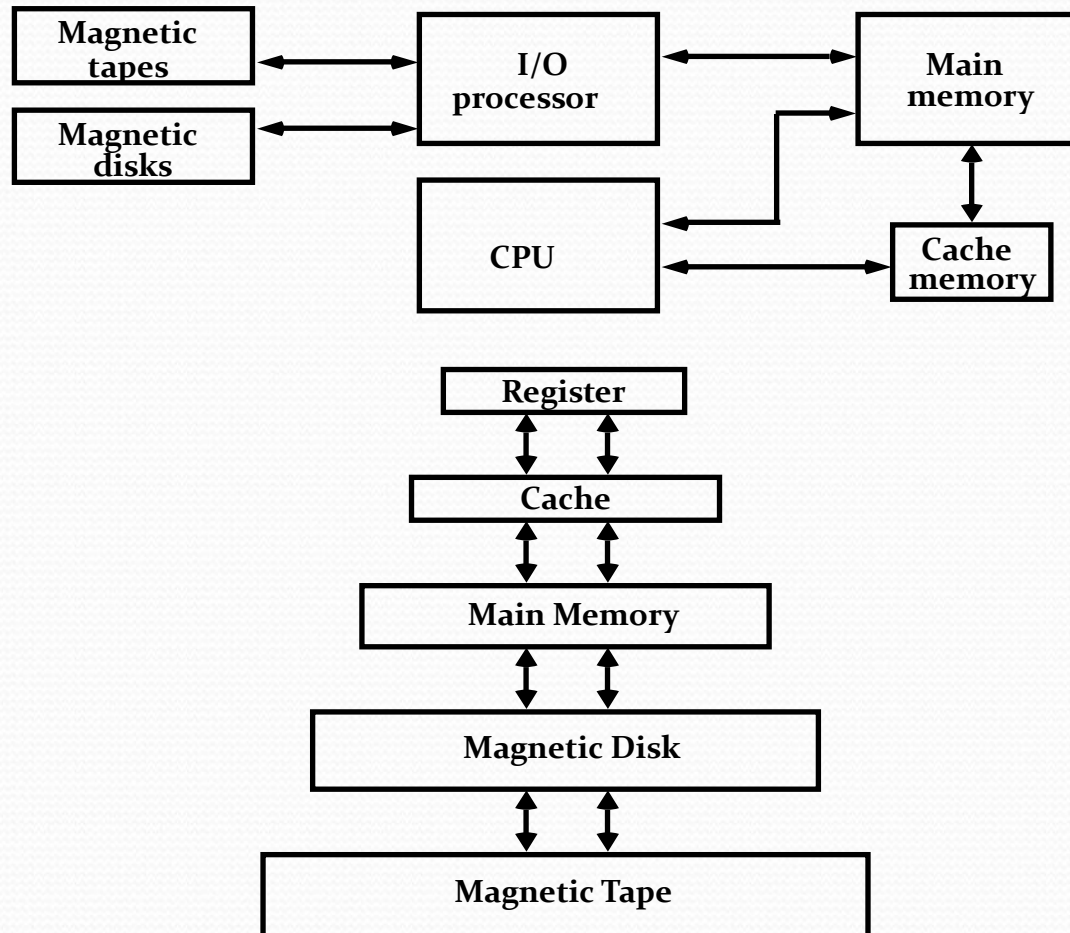


# Overview

- Memory Hierarchy
- Main Memory
- Auxiliary Memory
- Associative Memory
- Cache Memory
- Virtual Memory

# Memory Hierarchy

**Memory Hierarchy is to obtain the highest possible access speed while minimizing the total cost of the memory system**

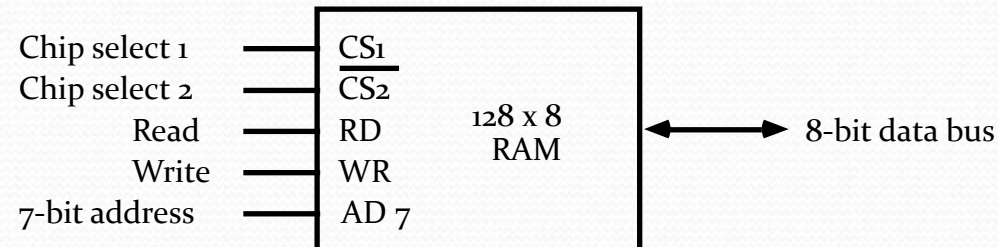




# Main Memory

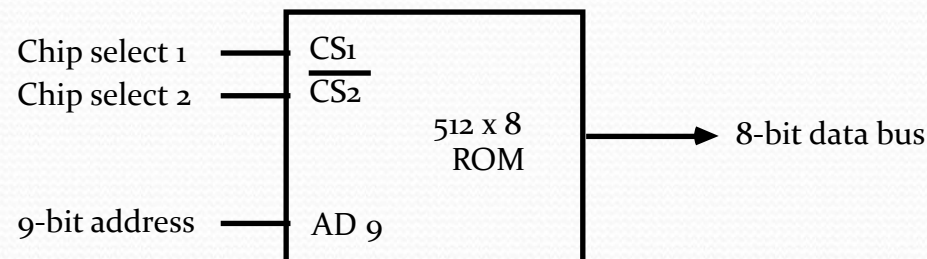
## RAM and ROM Chips

### Typical RAM chip



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

### Typical ROM chip



# Memory Address Map

## Address space assignment to each memory chip

**Example: 512 bytes RAM and 512 bytes ROM**

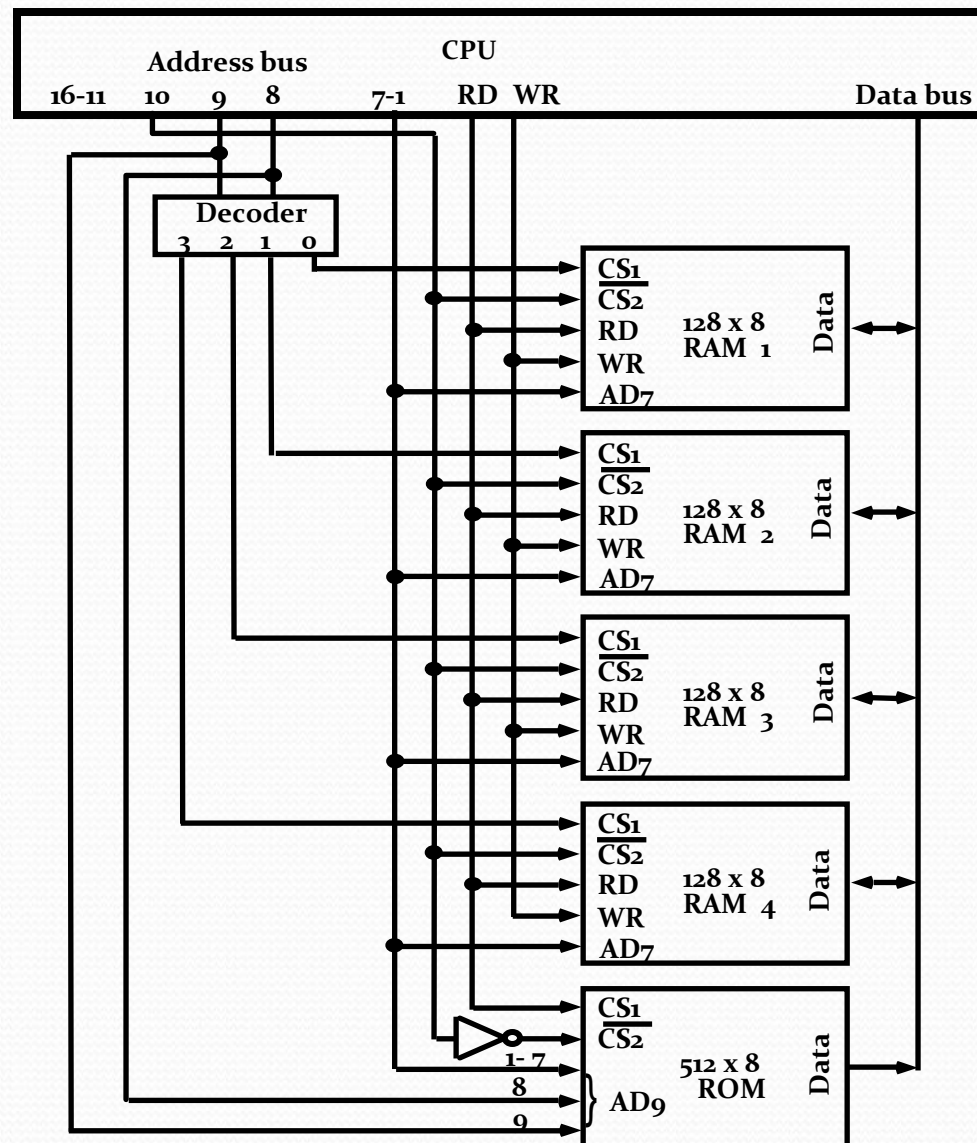
Component	Hexa address	Address bus									
		10	9	8	7	6	5	4	3	2	1
RAM 1	0000 - 007F	0	0	0	x	x	x	x	x	x	x
RAM 2	0080 - 00FF	0	0	1	x	x	x	x	x	x	x
RAM 3	0100 - 017F	0	1	0	x	x	x	x	x	x	x
RAM 4	0180 - 01FF	0	1	1	x	x	x	x	x	x	x
ROM	0200 - 03FF	1	x	x	x	x	x	x	x	x	x

## Memory Connection to CPU

- RAM and ROM chips are connected to a CPU through the data and address buses
- The low-order lines in the address bus select the byte within the chips and other lines in the address bus select a particular chip through its chip select inputs

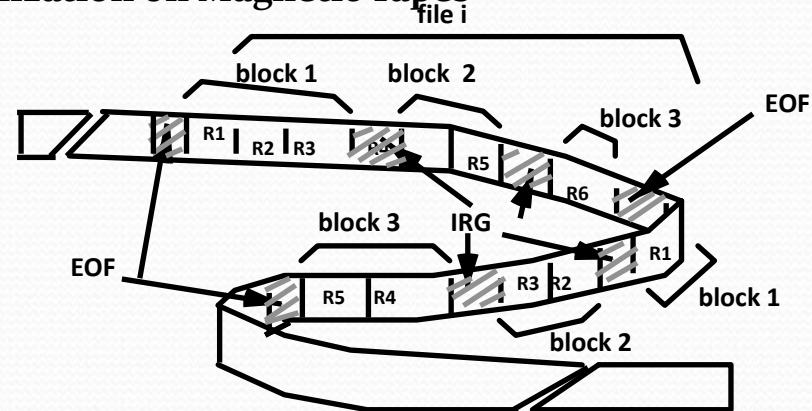


# Connection of Memory to CPU



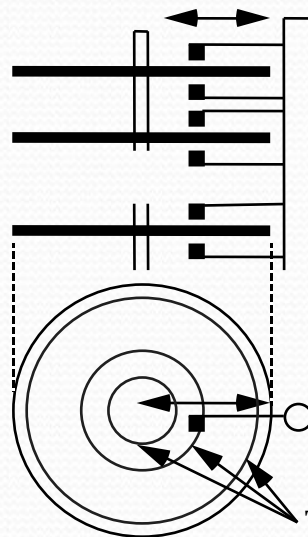
# Auxiliary Memory

## Information Organization on Magnetic Tapes

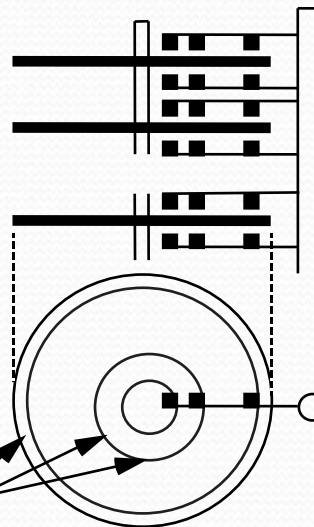


## Organization of Disk Hardware

### Moving Head Disk



### Fixed Head Disk

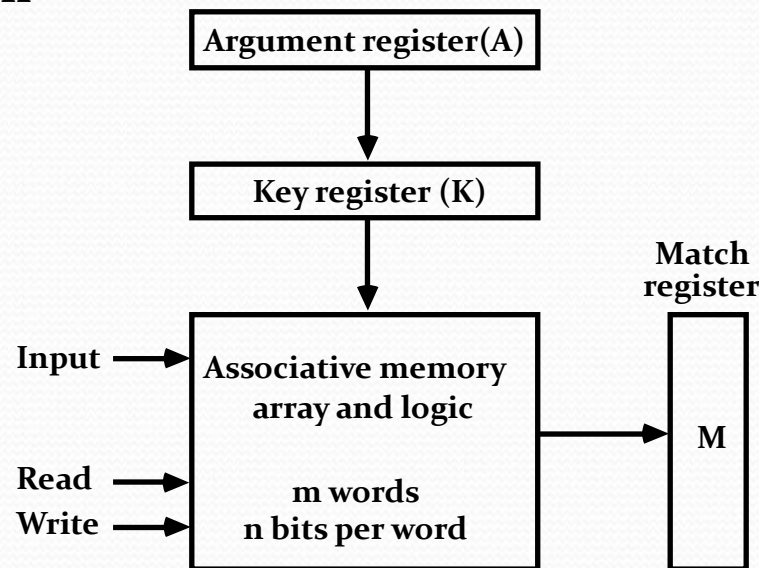




# Associative Memory

- Accessed by the content of the data rather than by an address
- Also called Content Addressable Memory (CAM)

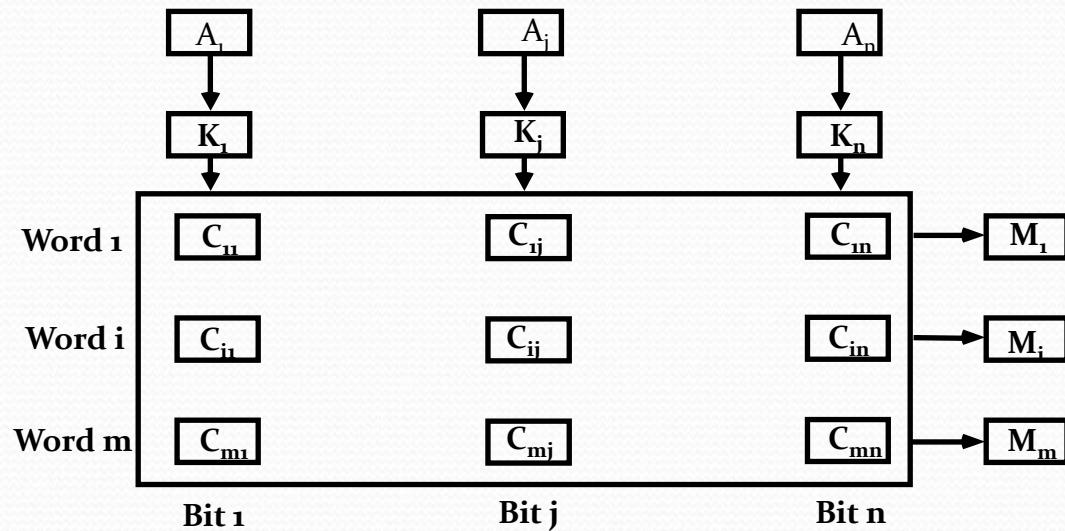
## Hardware Organization



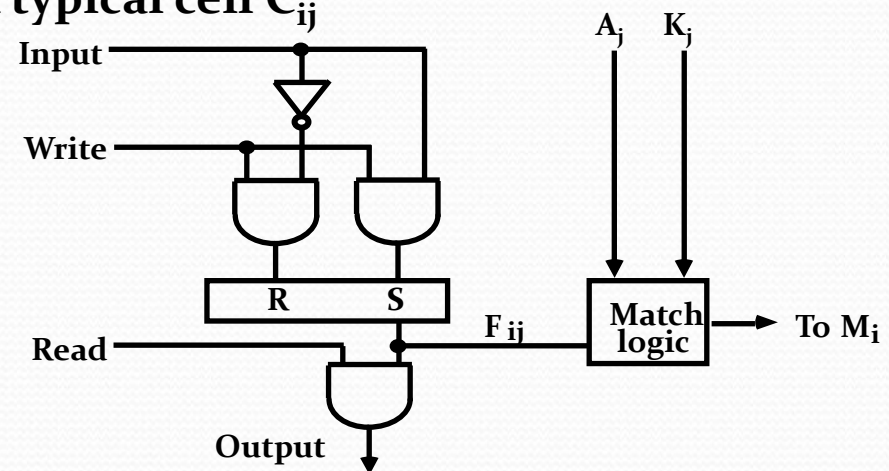
A	101 111100	
K	111 000000	
Word 1	100 111100	no match
Word 2	101 000001	match

# Organization of CAM

$i$  = word  
 $j$  = bit in word



## Internal organization of a typical cell $C_{ij}$





# Cache Memory

## Locality of Reference

- The references to memory at any given time interval tend to be confined within a localized areas
- This area contains a set of information and the membership changes gradually as time goes by
- *Temporal Locality*  
The information which will be used in near future is likely to be in use already( e.g. Reuse of information in loops)
- *Spatial Locality*  
If a word is accessed, adjacent(near) words are likely accessed soon (e.g. Related data items (arrays) are usually stored together; instructions are executed sequentially)

## Cache

- The property of Locality of Reference makes the cache memory systems work
- Cache is a fast small capacity memory that should hold those information which are most likely to be accessed



# Performance of Cache

## Memory Access

All the memory accesses are directed first to Cache

If the word is in Cache; Access cache to provide it to CPU

If the word is not in Cache; Bring a block (or a line) including that word to replace a block now in Cache

- How can we know if the word that is required is there ?
- If a new block is to replace one of the old blocks, which one should we choose ?

## Performance of Cache Memory System

Hit Ratio - % of memory accesses satisfied by Cache memory system

$T_e$ : Effective memory access time in Cache memory system

$T_c$ : Cache access time

$T_m$ : Main memory access time

$$T_e = T_c + (1 - h) T_m$$

Example:  $T_c = 0.4 \mu s$ ,  $T_m = 1.2 \mu s$ ,  $h = 0.85$

$$T_e = 0.4 + (1 - 0.85) * 1.2 = 0.58 \mu s$$



# Memory and Cache Mapping – (Associative Mapping)

## Mapping Function

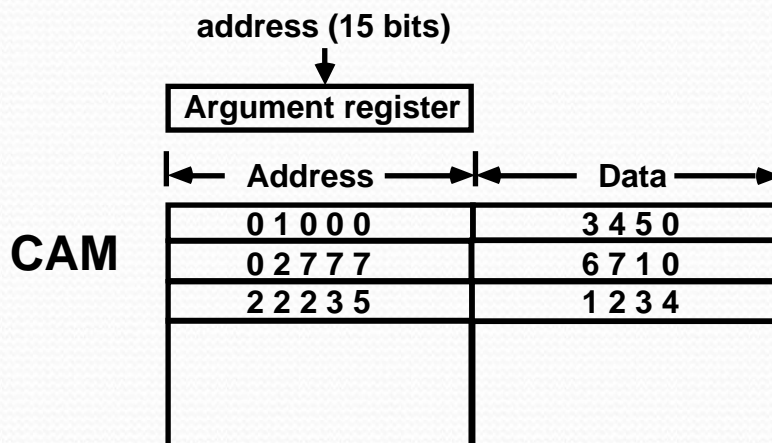
Specification of correspondence between main memory blocks and cache blocks

Associative mapping  
Direct mapping  
Set-associative mapping

## Associative Mapping

- Any block location in Cache can store any block in memory
- > Most flexible
- Mapping Table is implemented in an associative memory
- > Fast, very Expensive
- Mapping Table

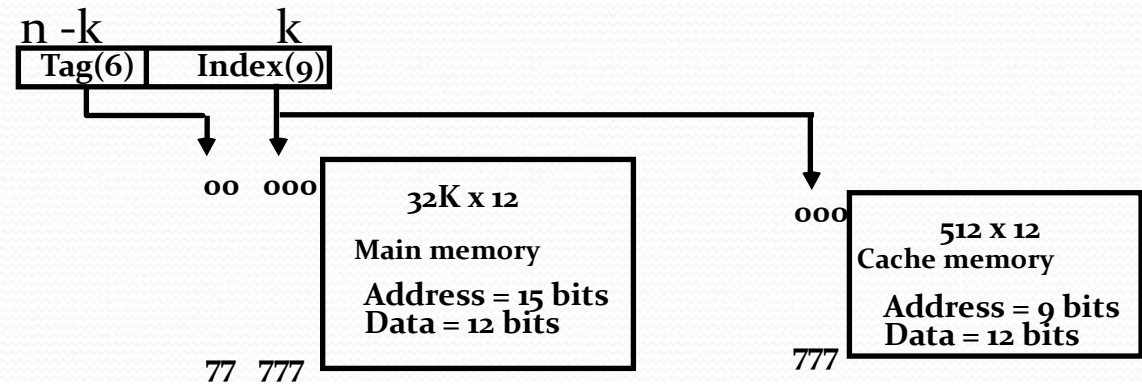
Stores both address and the content of the memory word



# Cache Mapping – direct mapping

- Each memory block has only one place to load in Cache
- Mapping Table is made of RAM instead of CAM
- n-bit memory address consists of 2 parts; k bits of Index field and n-k bits of Tag field
- n-bit addresses are used to access main memory and k-bit Index is used to access the Cache

## Addressing Relationships



## Direct Mapping Cache Organization

Memory address	Memory data
00000	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

Cache memory		
Index address	Tag	Data
000	0 0	1 2 2 0
777	0 2	6 7 1 0



# Cache Mapping – direct mapping

## Operation

- CPU generates a memory request with (TAG;INDEX)
- Access Cache using INDEX ; (tag; data)  
Compare TAG and tag
- If matches -> Hit  
Provide Cache(data) to CPU
- If not match -> Miss  
Search main memory and replace the block from cache memory

## Direct Mapping with block size of 8 words

Block 0	000	0 1	3 4 5 0
	007	0 1	6 5 7 8
Block 1	010		
	017		
Block 63	770	0 2	
	777	0 2	6 7 1 0

Tag	Block	Word
-----	-------	------

INDEX

# Cache Mapping – Set Associative Mapping

- Each memory block has a set of locations in the Cache to load

**Set Associative Mapping Cache with set size of two**

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





# Cache Write

## Write Through

When writing into memory

If Hit, both Cache and memory is written in parallel

If Miss, Memory is written

For a read miss, missing block may be overloaded onto a cache block

Memory is always updated

-> Important when CPU and DMA I/O are both executing

Slow, due to the memory access time

## Write-Back (Copy-Back)

When writing into memory

If Hit, only Cache is written

If Miss, missing block is brought to Cache and write into Cache

For a read miss, candidate block must be written back to the memory

Memory is not up-to-date, i.e., the same item in

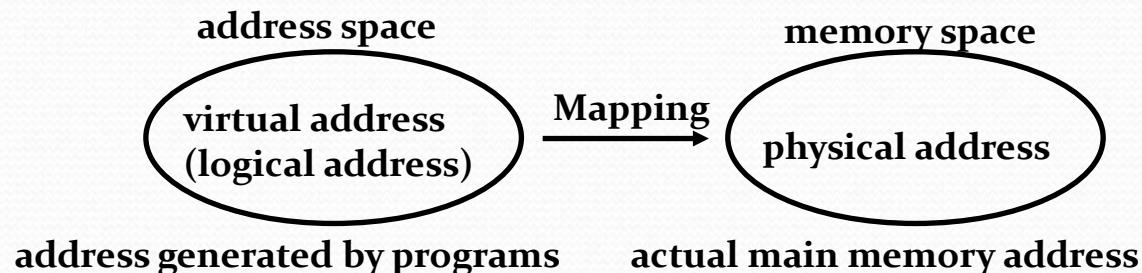
Cache and memory may have different value



# Virtual Memory

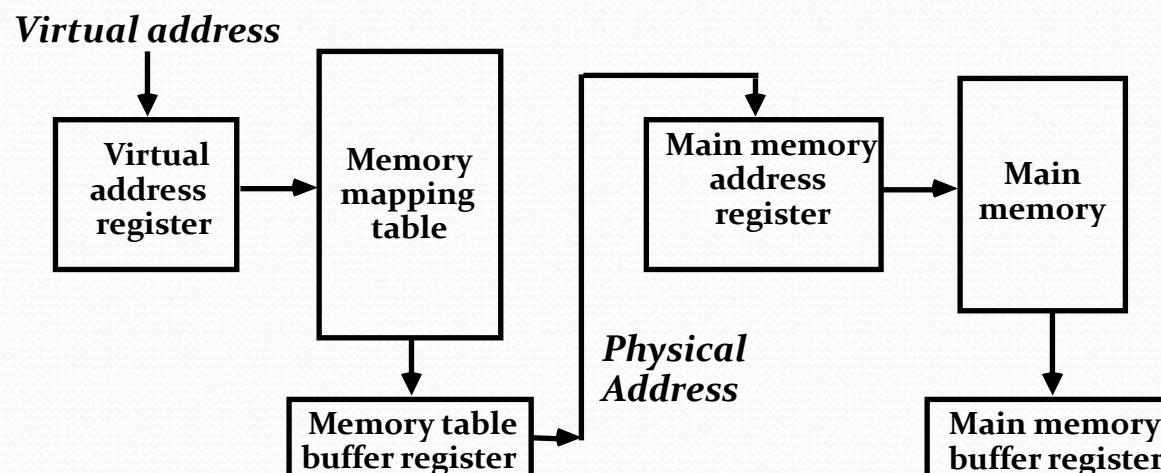
Give the programmer the illusion that the system has a very large memory, even though the computer actually has a relatively small main memory

## Address Space(Logical) and Memory Space(Physical)



## Address Mapping

### Memory Mapping Table for Virtual Address -> Physical Address



# Address Mapping

Address Space and Memory Space are each divided into fixed size group of words called *blocks* or *pages*

1K words group

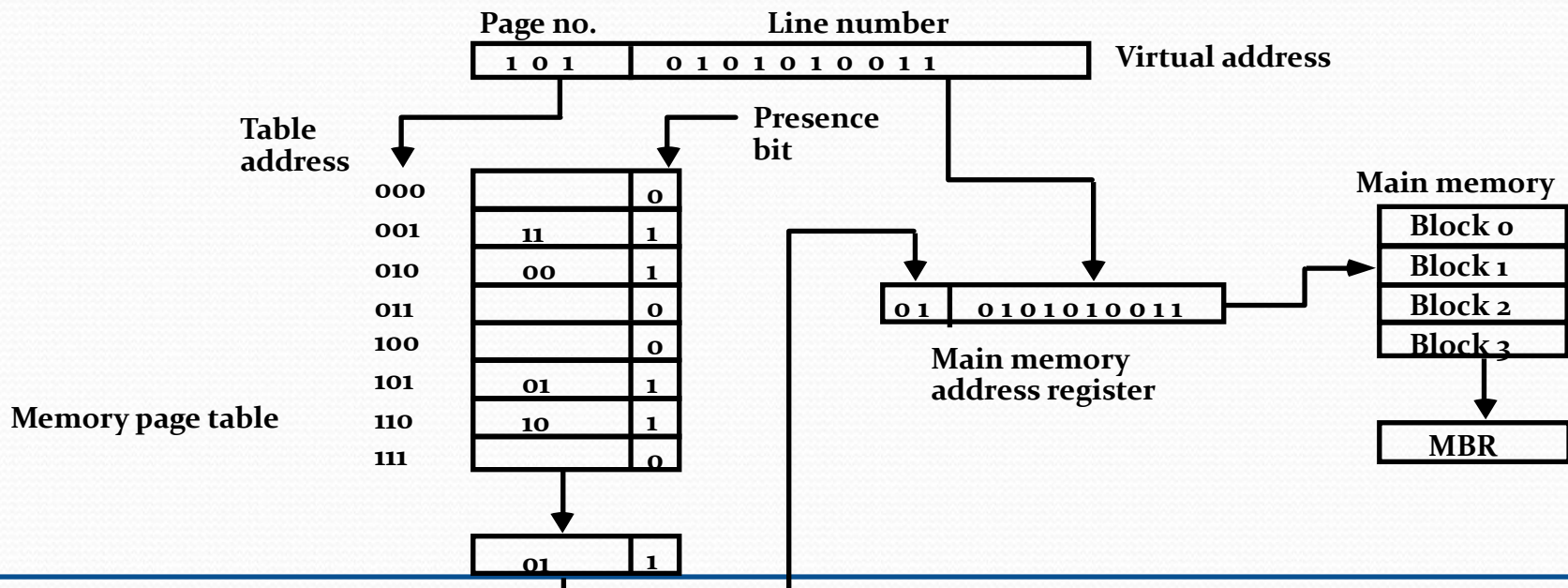
Address space  
 $N = 8K = 2^{13}$

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

Memory space  
 $M = 4K = 2^{12}$

Block 0
Block 1
Block 2
Block 3

## Organization of memory Mapping Table in a paged system





# Associative Memory Page Table

Assume that

Number of Blocks in memory =  $m$

Number of Pages in Virtual Address Space =  $n$

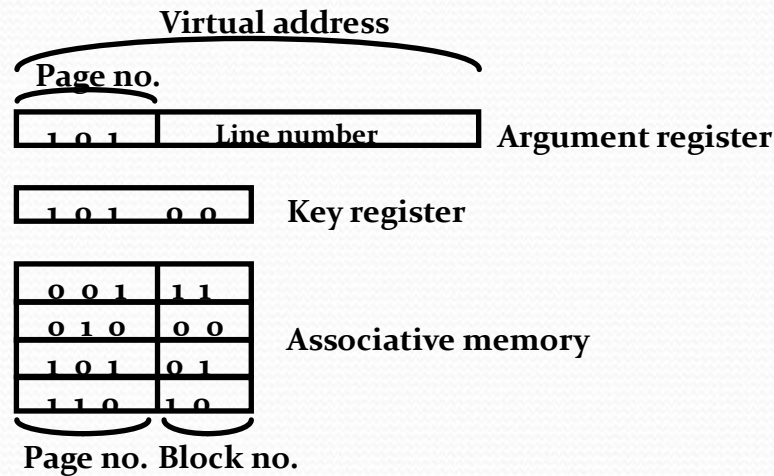
## Page Table

- Straight forward design  $\rightarrow$   $n$  entry table in memory Inefficient storage space utilization

$\leftarrow$   $n-m$  entries of the table is empty

- More efficient method is  $m$ -entry Page Table

Page Table made of an Associative Memory  
 $m$  words; (Page Number: Block Number)



## Page Fault

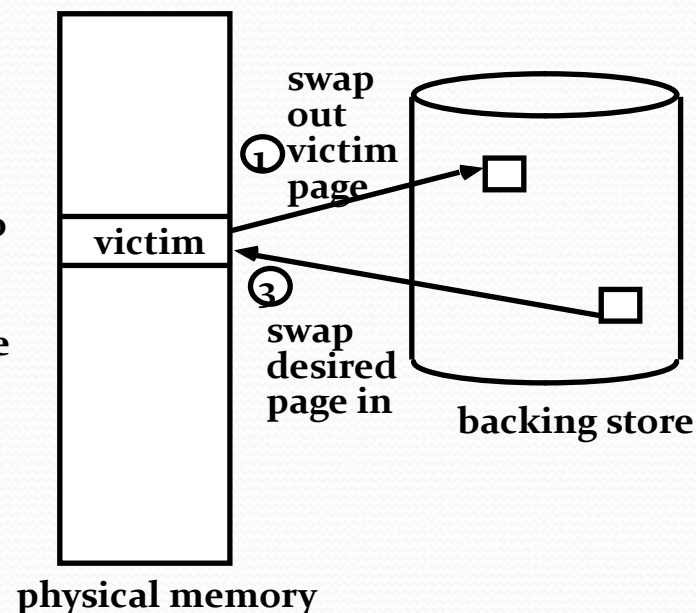
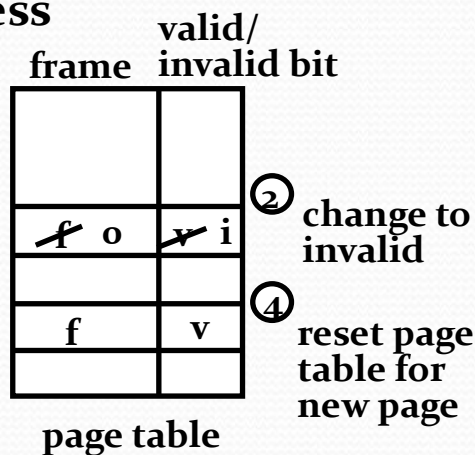
Page number cannot be found in the Page Table

# Page Replacement

Decision on which page to displace to make room for an incoming page when no free frame is available

Modified page fault service routine

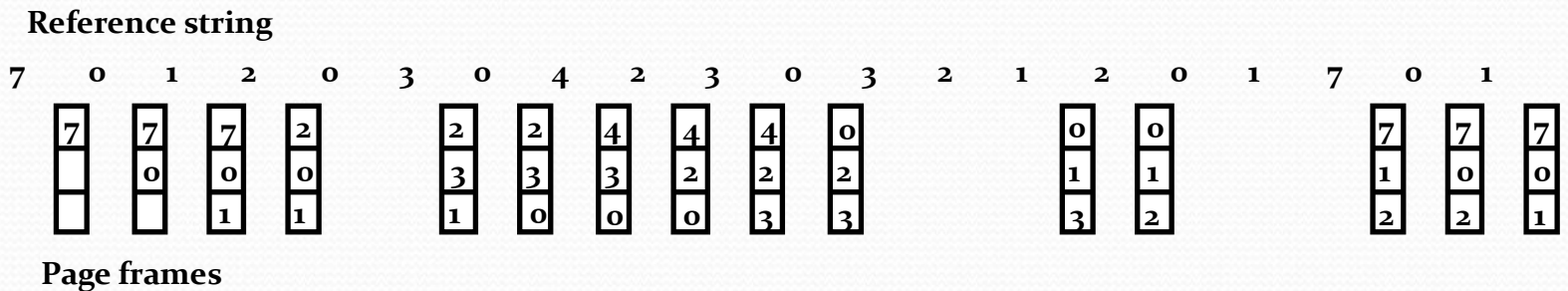
1. Find the location of the desired page on the backing store
2. Find a free frame
  - If there is a free frame, use it
  - Otherwise, use a page-replacement algorithm to select a *victim* frame
  - Write the victim page to the backing store
3. Read the desired page into the (newly) free frame
4. Restart the user process





# Page Replacement Algorithms

## FIFO



FIFO algorithm selects the page that has been in memory the longest time  
 Using a queue - every time a page is loaded, its  
 identification is inserted in the queue

Easy to implement

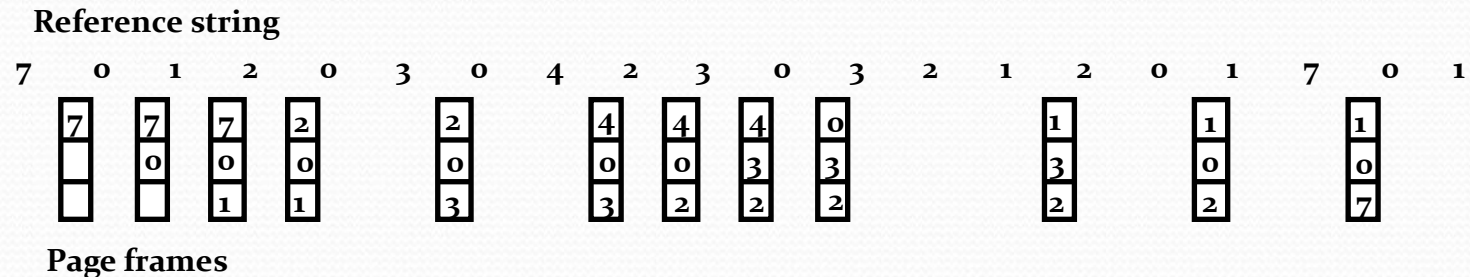
May result in a frequent page fault

# Page Replacement Algorithms

## LRU

- LRU uses the recent past as an approximation of near future.

Replace that page which has not been used for the longest period of time



- LRU may require substantial hardware assistance
- The problem is to determine an order for the frames defined by the time of last use