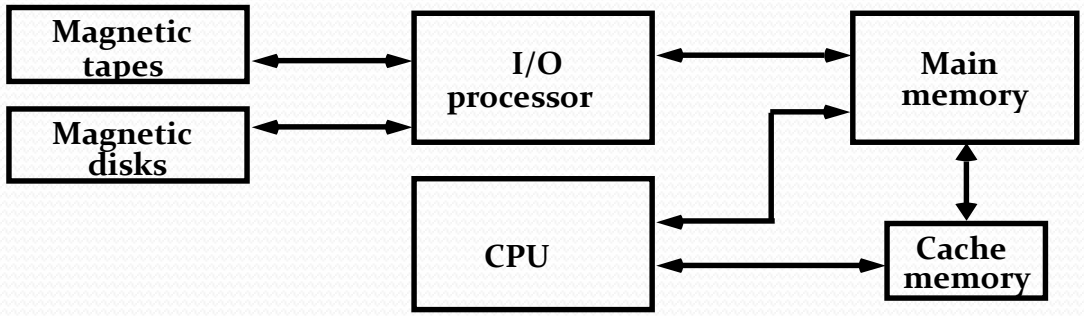
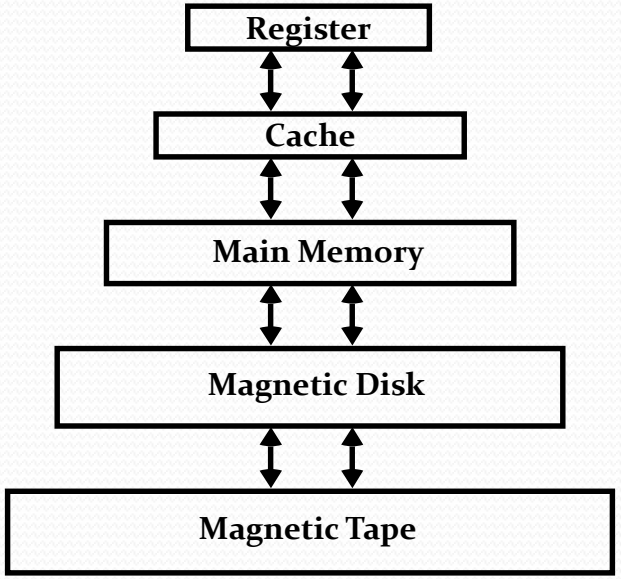
Overview

- 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





Quiz Time

Which of the following is the fastest memory?

A. Magnetic Disk
B. Optical Disk
C. Tape Disk
D. ROM

Quiz Time

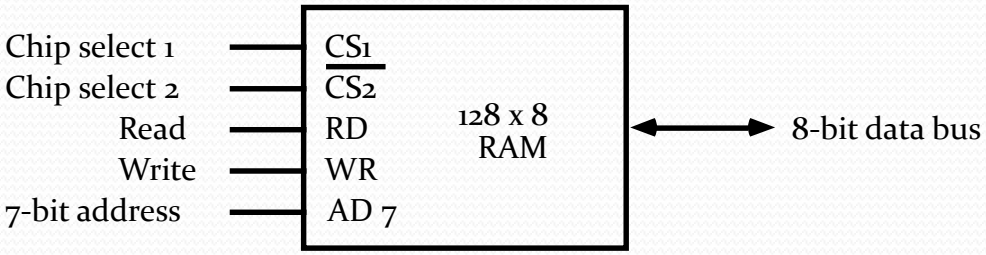
Which of the following can be used to store large size of data?

A. Cache Memory
B. Random Access Memory
C. Magnetic Disk

Main Memory

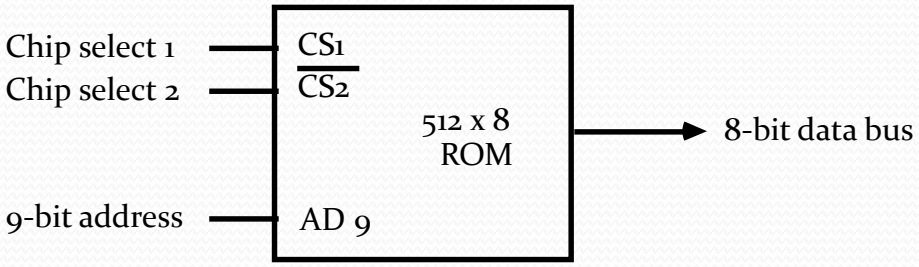
RAM and ROM Chips

Typical RAM chip



CS1	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



Quiz Time

How many 128 x 8 RAM chips are needed to create 2048 Byte of memory?

A. 16 Chips
B. 14 Chips
C. 10 Chips
D. 8 Chips

Memory Address Map

Address space assignment to each memory chip

Example: 512 bytes RAM and 512 bytes ROM

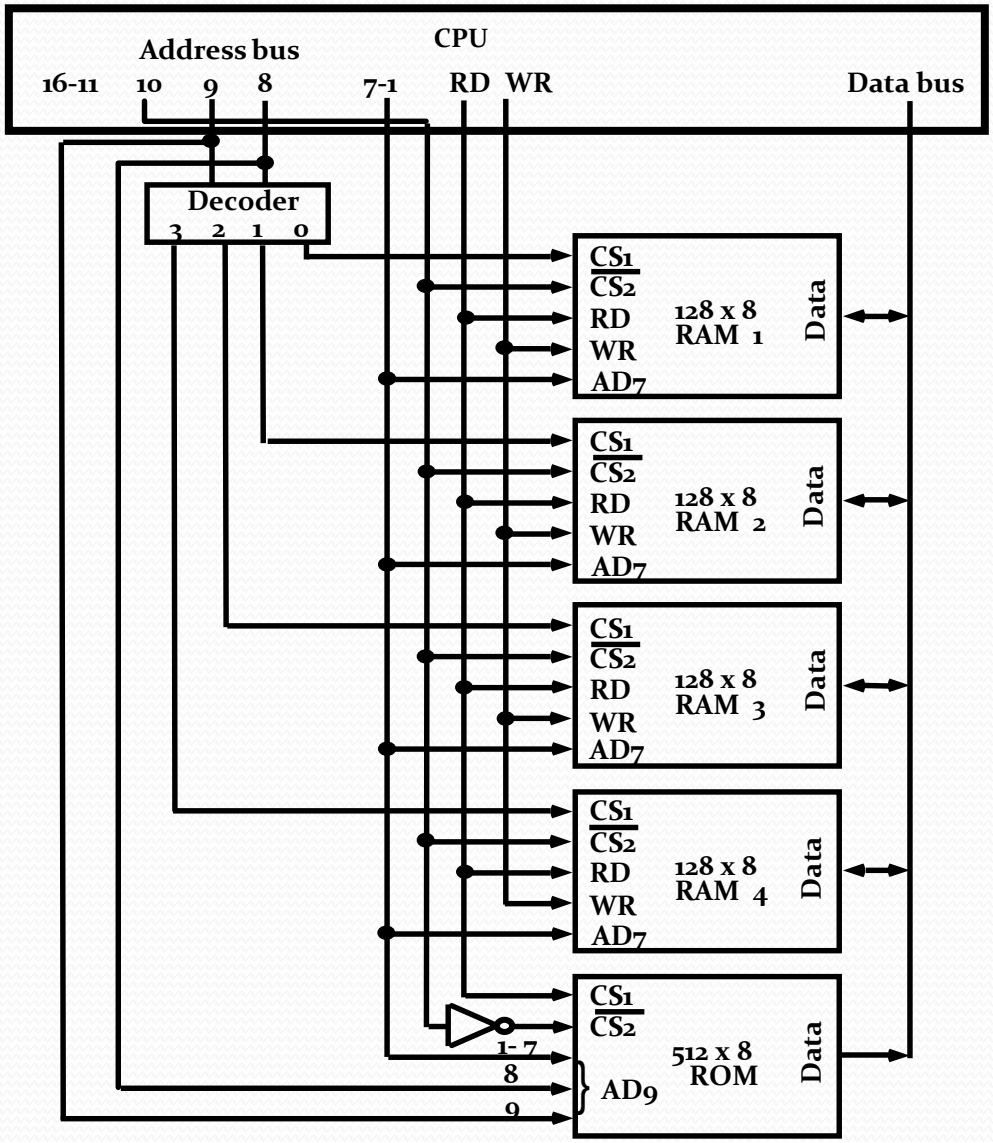
Component	Hexa address	Address bus
		10 9 8 7 6 5 4 3 2 1
RAM 1	0000 - 007F	0 0 0 x x x x x x x
RAM 2	0080 - 00FF	0 0 1 x x x x x x x
RAM 3	0100 - 017F	0 1 0 x x x x x x x
RAM 4	0180 - 01FF	0 1 1 x x x x x x x
ROM	0200 - 03FF	1 x x x x x x x x x

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



Quiz Time

How many lines of the address bus Configuration Size of each chip is 128 X 8 (Total 16 chips = 2048 bytes)must be used to access 2048 bytes of memory? How many of these lines will be common to all chips?

A. **11 bit address line, 7 lines common for each chip**
B. **11 bit address line, 8 lines common for each chip**
C. **12 bit address line, 7 lines common for each chip**

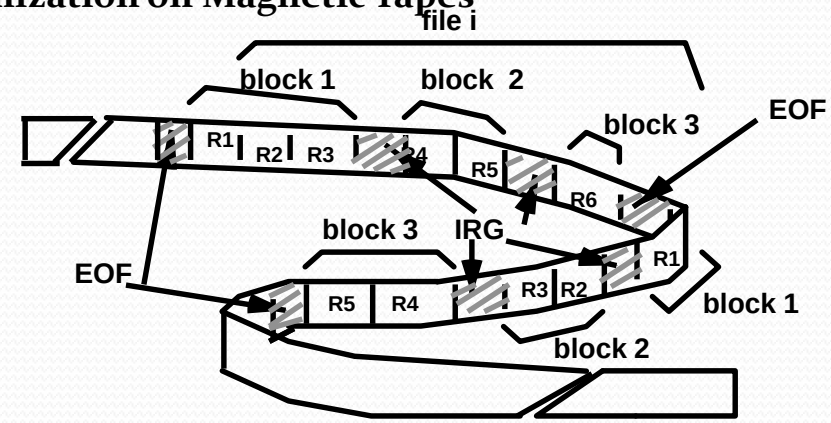
Quiz Time

How many lines must be decoded for chip select? Specify the size of the decoders. Configuration Size of each chip is 128 X 8 (Total 16 chips = 2048 bytes)

A. **4-lines ; Size 4 x16 Decoder**
B. **16-lines ; Size 16 x 4 Decoder**
C. **4-lines ; Size 16 x 4 Decoder**

Auxiliary Memory

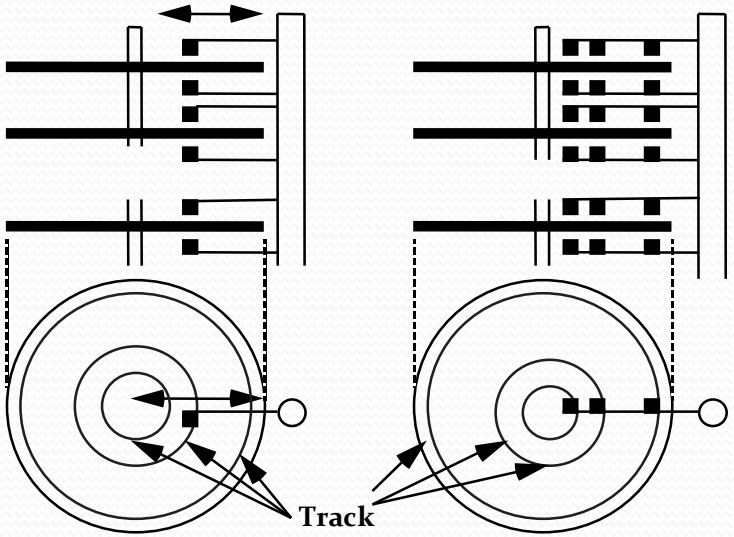
Information Organization on Magnetic Tapes



Organization of Disk Hardware

Moving Head Disk

Fixed Head Disk



https://youtu.be/wIoupu9eVcw

A magnetic disk system has the following parameters:

T_s = average time to position the magnetic head over a track

R = rotation speed of disk in revolutions per second

N_t = number of bits per track

N_s = number of bits per sector

Calculate the average time T_a that it will take to read one sector.

Average time = T_s + time for half revolution + time to read a sector.

$$T_a = T_s + \frac{1}{2R} + \frac{N_s}{N_t} \times \frac{1}{R}$$

Q.2

What is the transfer rate of an eight-track magnetic tape whose speed is 120 inches per second and whose density is 1600 bits per inch?

Solution .2

An eight-track tape reads 8 bits (one character) at the same time.

Transfer rate = 1600 x 120 = 192,000 characters/s

Associative Memory

The time required to find an item stored in memory can be reduced considerably if stored data can be identified for access by the content of the data itself rather than by an address.

A memory unit access by content is called an associative memory or Content Addressable Memory (CAM). This type of memory is accessed simultaneously and in parallel on the basis of data content rather than specific address or location.

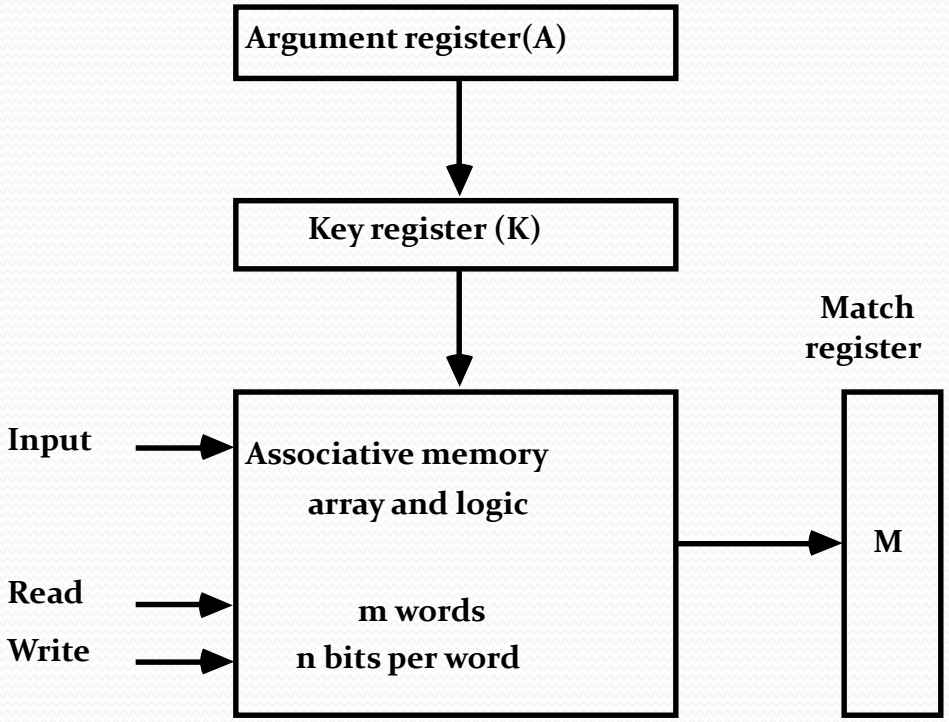
When a word is written in an associative memory, no address is given. The memory is capable of finding an empty unused location to store the word. When a word is to be read from an associative memory, the content of the word or part of the word is specified.

The associative memory is uniquely suited to do parallel searches by data association. Moreover, searches can be done on an entire word or on a specific field within a word. Associative memories are used in applications where the search time is very critical and must be very short.

Associative Memory

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

Hardware Organization



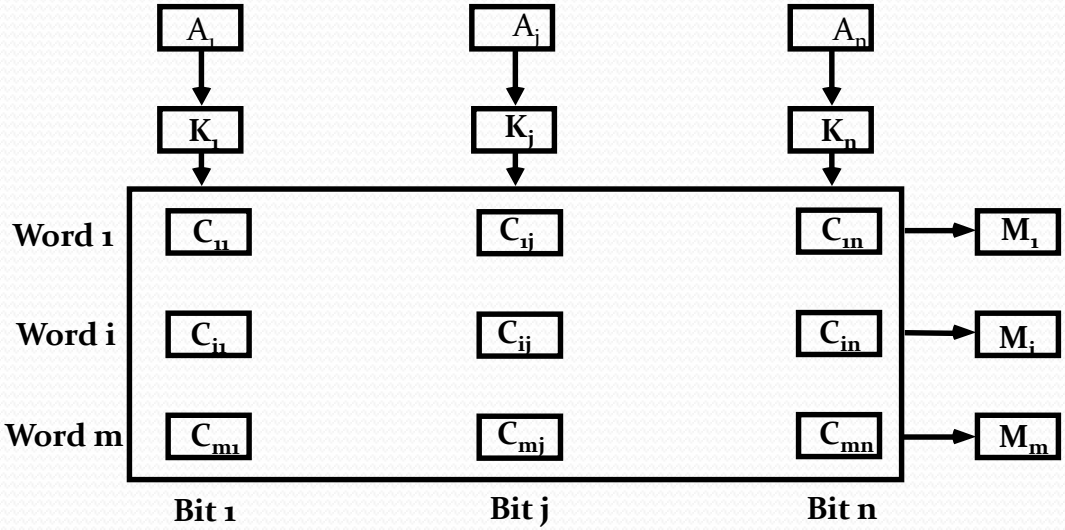
Example

Find The Match Word Using Associative Memory

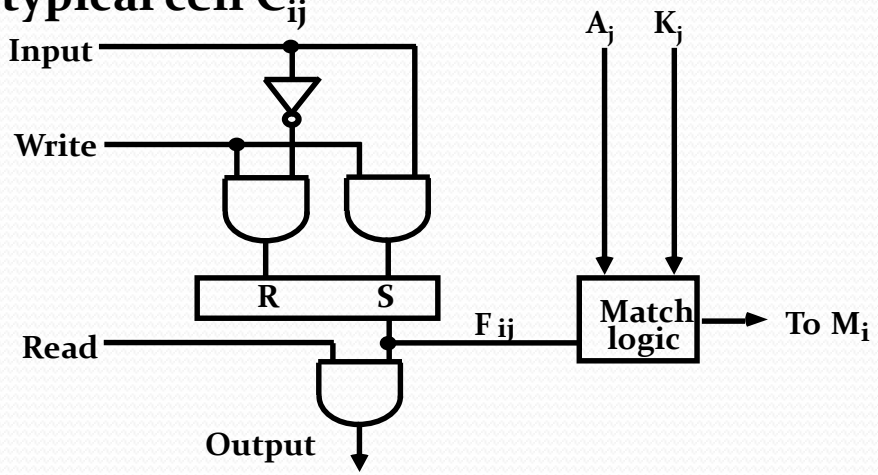
A	101 111100	
K	111 000000	
Word 1	100 111100	no match
Word 2	101 000001	match

Organization of CAM

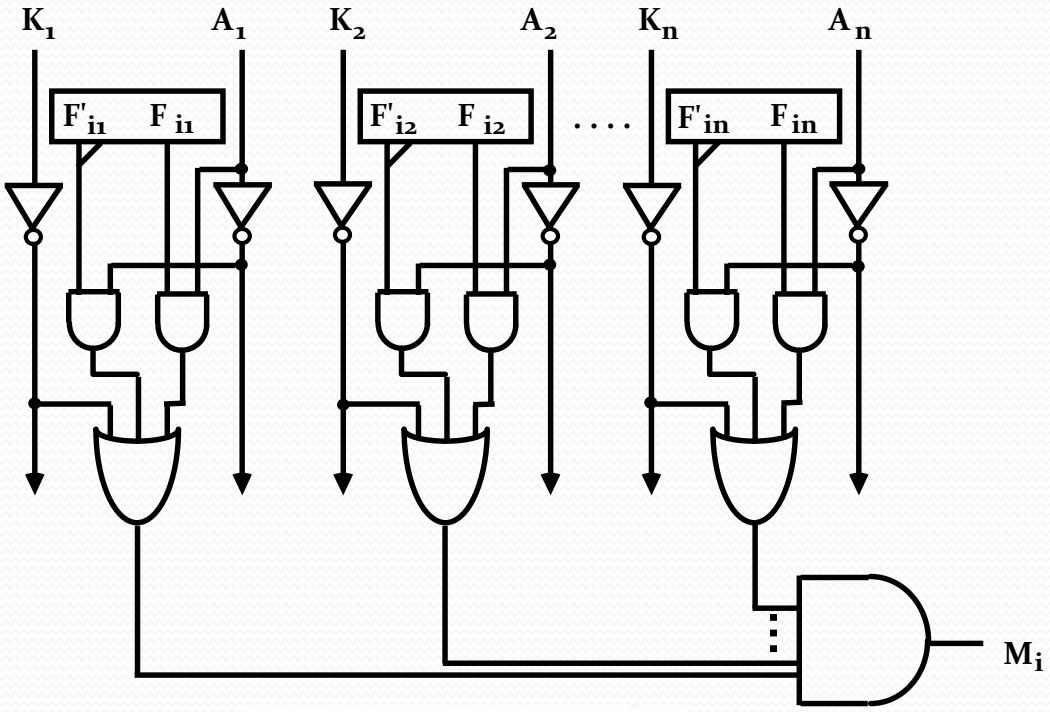
i= word
j-= bit in word



Internal organization of a typical cell C_{ij}



Match Logic



$$x_j = A_j F_{ij} + A_j' F_{ij}'$$

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

Read Operation

If more than one word in memory matches the unmasked argument field, all the matched words will have 1's in the corresponding bit position of the match register.

It is then necessary to scan the bits of the match register one at a time. The matched words are read in sequence by applying a read signal to each word line whose corresponding M_i bit is a 1.

If only one word may match the unmasked argument field, then connect output M_i directly to the read line in the same word position,

The content of the matched word will be presented automatically at the output lines and no special read command signal is needed.

If we exclude words having zero content, then all zero output will indicate that no match occurred and that the searched item is not available in memory.

Write Operation

If the entire memory is loaded with new information at once, then the writing can be done by addressing each location in sequence.

The information is loaded prior to a search operation.

If unwanted words have to be deleted and new words inserted one at a time, there is a need for a special register to distinguish between active an inactive words.

This register is called “Tag Register”.

A word is deleted from memory by clearing its tag bit to 0.

Quiz Time

Find The Match Word Using Associative Memory

Q.no.2	A	1 0 1 0 1 0 1 0
	K	0 0 0 1 1 0 1 0
	word 1	1 0 0 0 1 1 1 0
	Word 2	1 0 0 1 1 1 1 0

A. Word-1
B. Word-2

Quiz Time

Find The Match Word Using Associative Memory

Q.no.1

A	101 111100
K	111 000000
Word 1	100 111100
Word 2	101 000001

A. Word-1
B. Word-2

$$x_j = A_j F_{ij} + A_j' F_{ij}'$$

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

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***

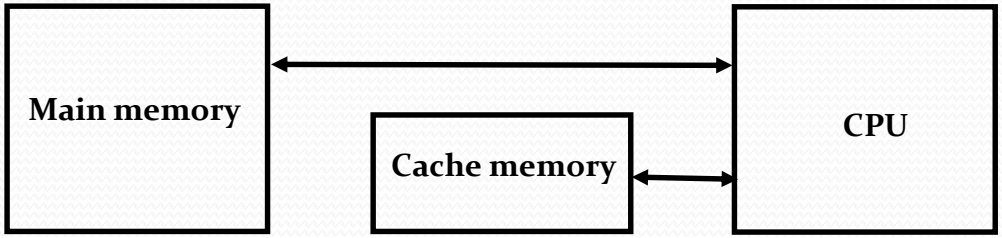
  If a word is accessed, adjacent(near) words are likely accessed soon (e.g. Related data items (arrays) are usually stored together; instructions are executed sequentially)

Cache Memory

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
Te: Effective memory access time in Cache memory system
Tc: Cache access time
Tm: Main memory access time

$$Te = Tc + (1 - h)\, Tm$$

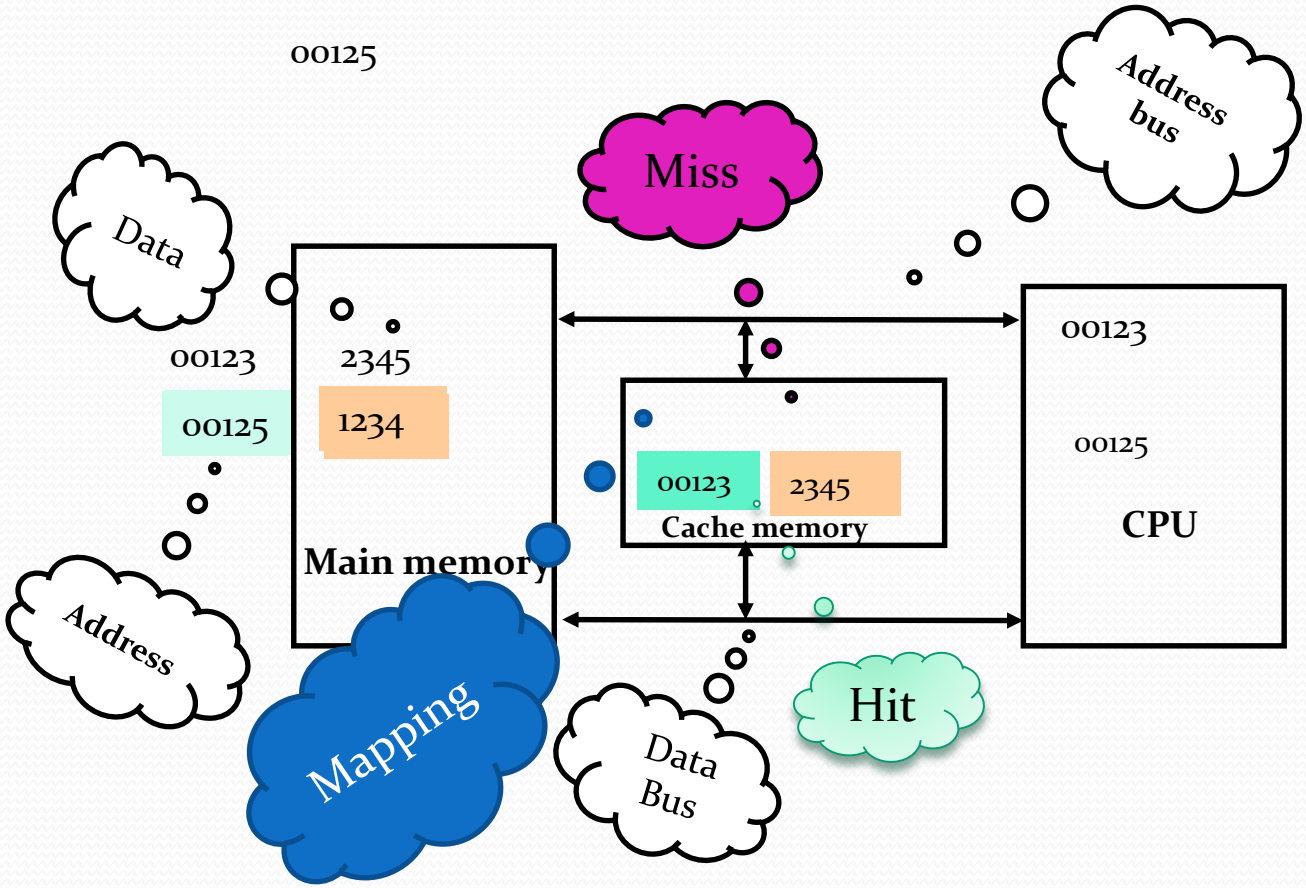
Example: Tc = 0.4 μs, Tm = 1.2μs, h = 0.85%

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

CPU wants to access two addresses:

00123

00125



Effective Access Time example: A computer has a single cache (off-chip) with a 2 ns hit time and a 98% hit rate. Main memory has a 40 ns access time. What is the computer's effective access time? If we add an on-chip cache with a .5 ns hit time and a 94% hit rate, what is the computer's effective access time? How much of a speedup does the on-chip cache give the computer?

Answers:

2 ns + .02 * 40 ns = 2.8 ns.

With the on-chip cache, we have .5 ns + .06 * (2 ns + .02 * 40 ns) = .668 ns. The speedup is 2.8 / .668 = 4.2.

Memory and Cache Mapping – (Associative Mapping)

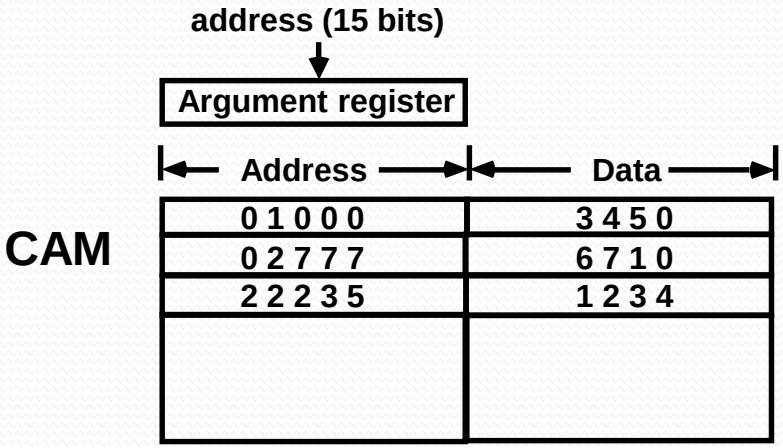
Mapping Function

Specification of correspondence between main memory blocks and cache blocks

Associative mapping
Direct mapping
Set-associative mapping

Associative Mapping

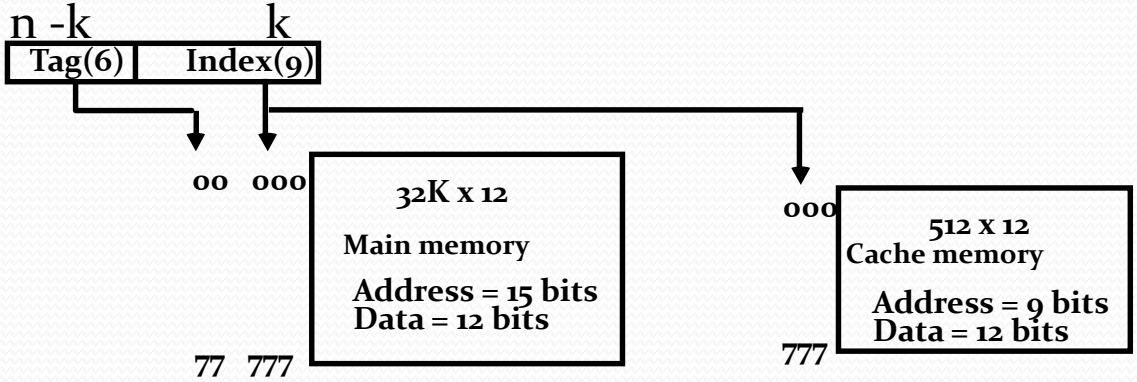
- Any block location in Cache can store any block in memory
  -> Most flexible
- Mapping Table is implemented in an associative memory
  -> Fast, very Expensive
- Mapping Table
  Stores both address and the content of the memory word



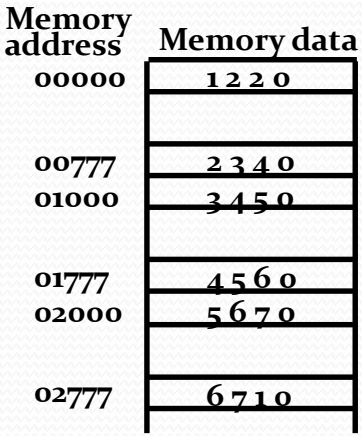
Cache Mapping – direct mapping

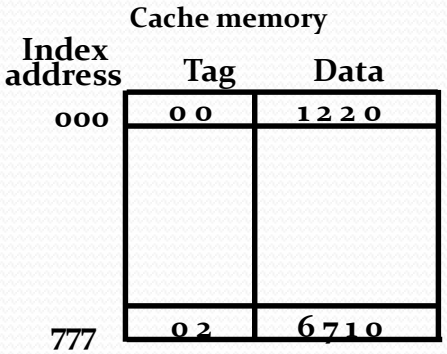
- Each memory block has only one place to load in Cache
- Mapping Table is made of RAM instead of CAM
- n-bit memory address consists of 2 parts; k bits of Index field and n-k bits of Tag field
- n-bit addresses are used to access main memory and k-bit Index is used to access the Cache

Addressing Relationships



Direct Mapping Cache Organization



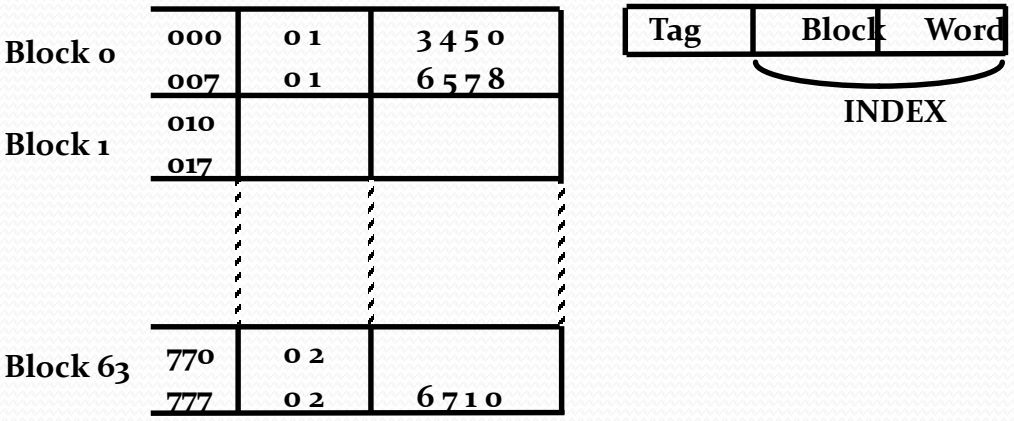


Cache Mapping – direct mapping

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



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	01	3450	02	5670
777	02	6710	00	2340

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

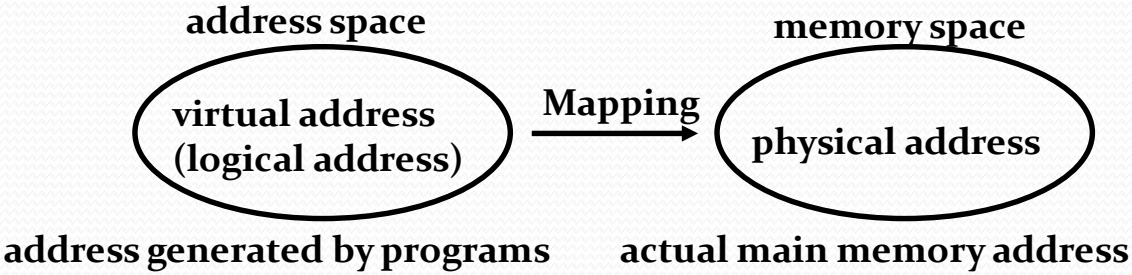
If Hit, only Cache is written
If Miss, missing block is brought to Cache and write into Cache
For a read miss, candidate block must be written back to the memory

Memory is not up-to-date, i.e., the same item in Cache and memory may have different value

Virtual Memory

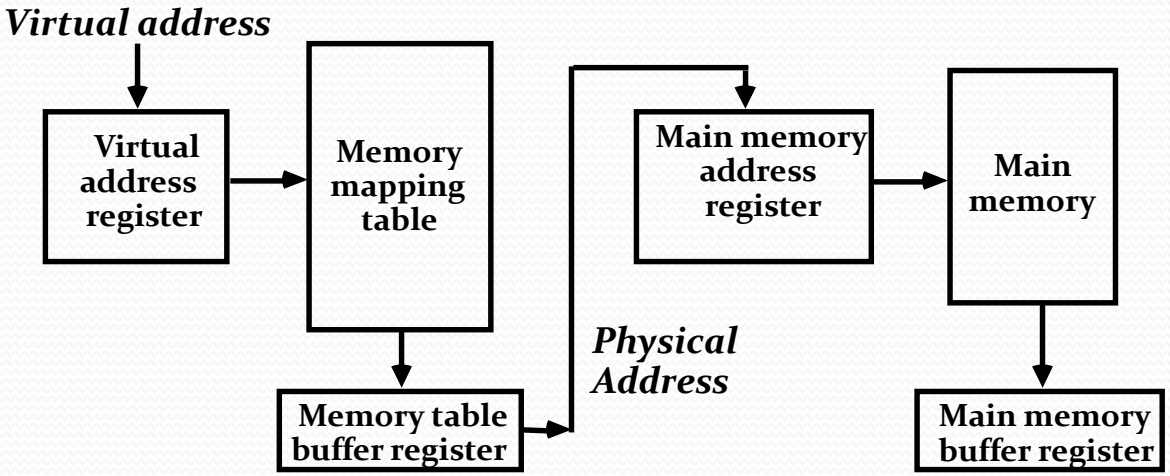
Give the programmer the illusion that the system has a very large memory, even though the computer actually has a relatively small main memory

Address Space(Logical) and Memory Space(Physical)



Address Mapping

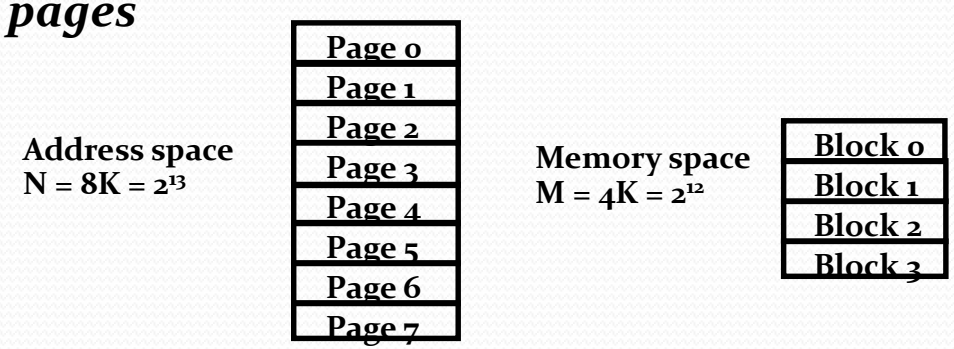
Memory *Mapping Table* for *Virtual Address* -> *Physical Address*



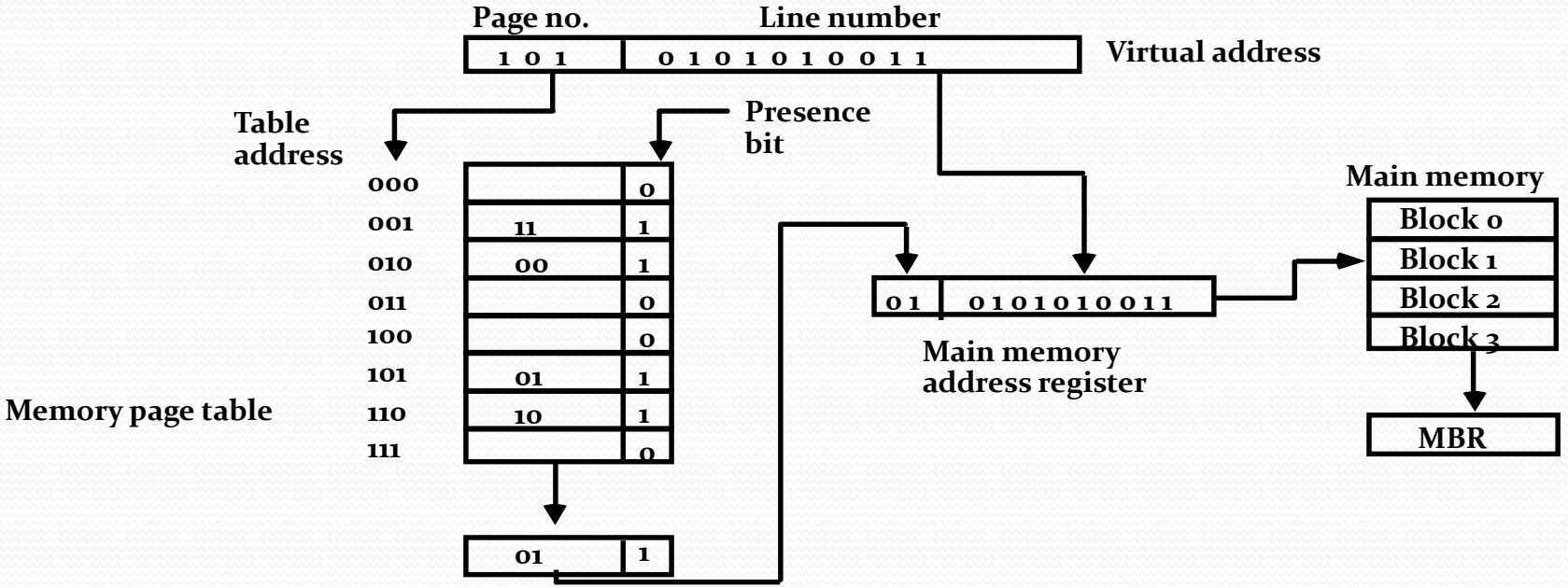
Address Mapping

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

1K words group



Organization of memory Mapping Table in a paged system



Example:-

a. Address space = 24 bits, find number of words in address space.

b. Memory Space= 16 bits , find number of words in memory space.

c. If a page consists of 2K words, how many pages and blocks are there in the system.

Solution -

12.19

(a) Address space = 24 bits 2^{24} = 16 M words

(b) Memory space = 16 bits 2^{16} = 64 K words

(c) $\frac{16M}{2K} = 8\,K\,pages$ $\frac{64K}{2K} = 32\,blocks$

Associative Memory Page Table

Assume that

Number of Blocks in memory = m

Number of Pages in Virtual Address Space = n

Page Table

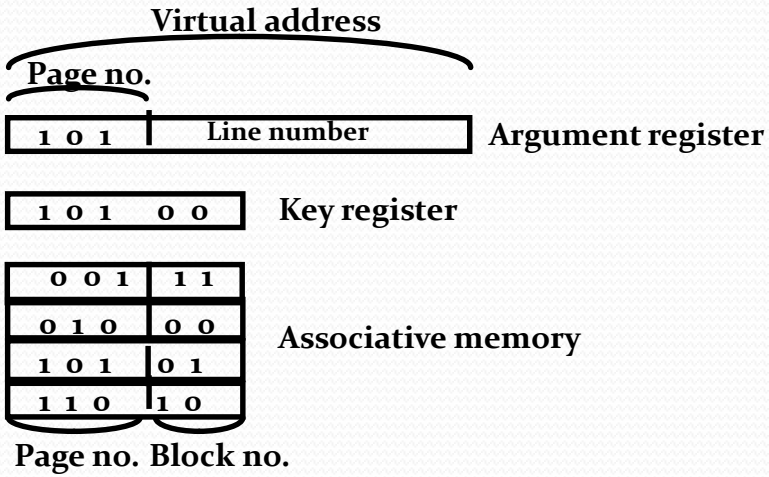
- Straight forward design -> n entry table in memory Inefficient storage space utilization

<- 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

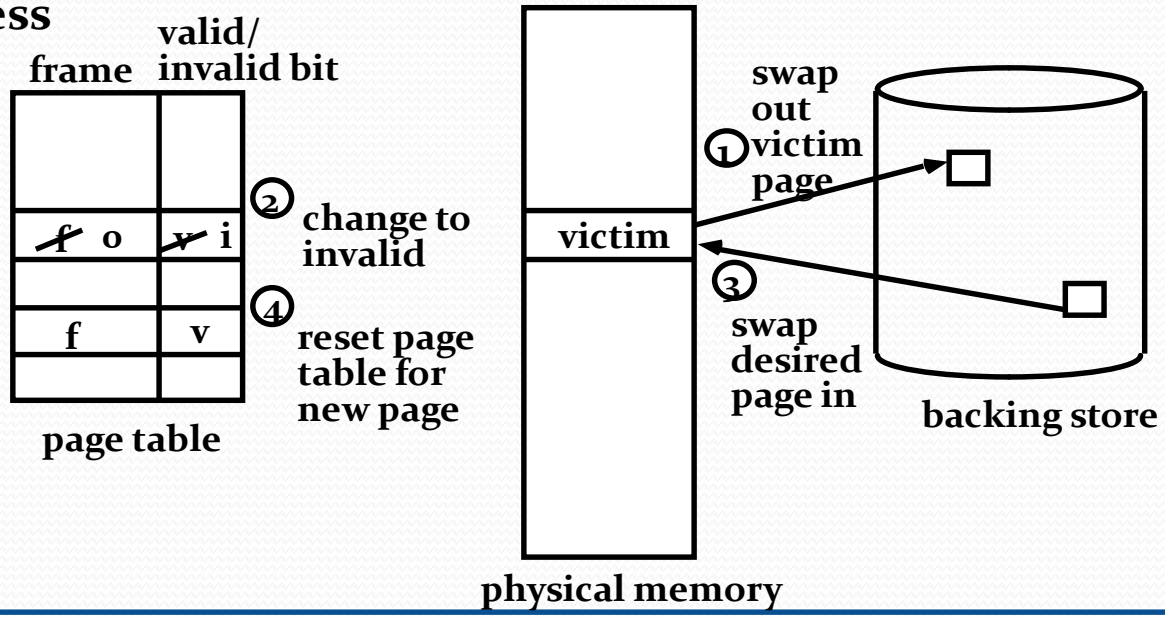
Page number cannot be found in the Page Table

Page Replacement

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

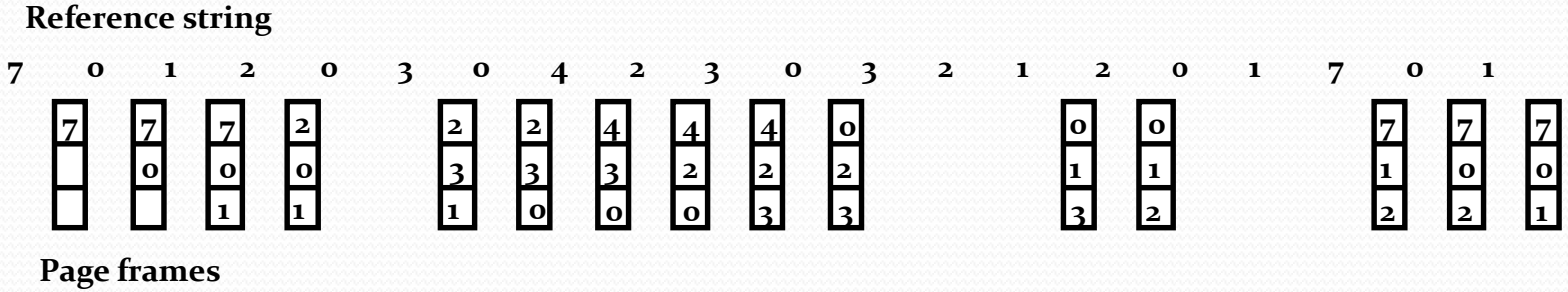
Modified page fault service routine

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



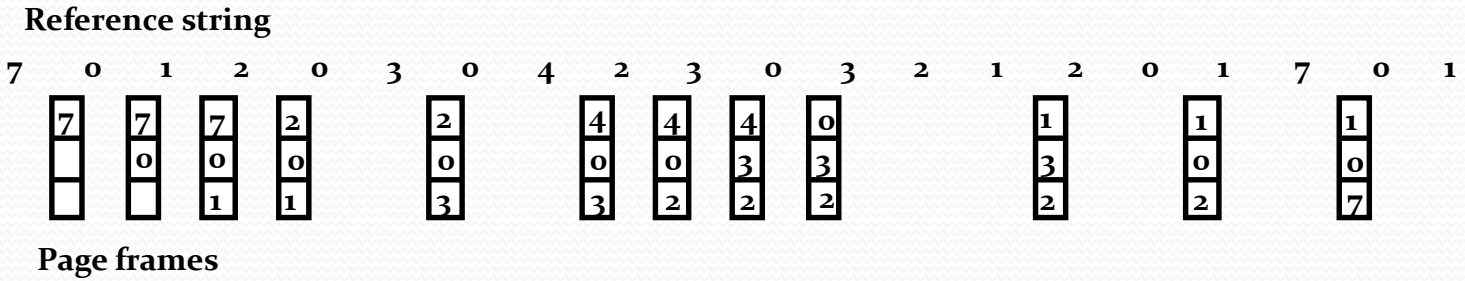
FIFO algorithm selects the page that has been in memory the longest time
Using a queue - every time a page is loaded, its identification is inserted in the queue
Easy to implement
May result in a frequent page fault

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**

A virtual memory system has an address space of 8K words, a memory space of 4K words, and page and block sizes of 1K words (see Fig. 12-18). The following page reference changes occur during a given time interval. (Only page changes are listed. If the same page is referenced again, it is not listed twice.)

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

Determine the four pages that are resident in main memory after each page reference change if the replacement algorithm used is (a) FIFO; (b) LRU.

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 four registers. A memory-mapped I/O configuration is used. The two highest-order bits of the address bus are assigned 00 for RAM, 01 for ROM, and 10 for interface registers.

a. How many RAM and ROM chips are needed?
b. Draw a memory-address map for the system.
c. Give the address range in hexadecimal for RAM, ROM, and interface.

A computer uses RAM chips of 1024 × 1 capacity.

a. How many chips are needed, and how should their address lines be connected to provide a memory capacity of 1024 bytes?

b. How many chips are needed to provide a memory capacity of 16K bytes? Explain in words how the chips are to be connected to the address bus.

(a) 8 chips are needed with address lines connected in parallel.

(b) $16 \times 8 = 128$ chips. Use 14 address lines ($16\ k = 2^{14}$)

10 lines specify the chip address

4 lines are decoded into 16 chip-select inputs.