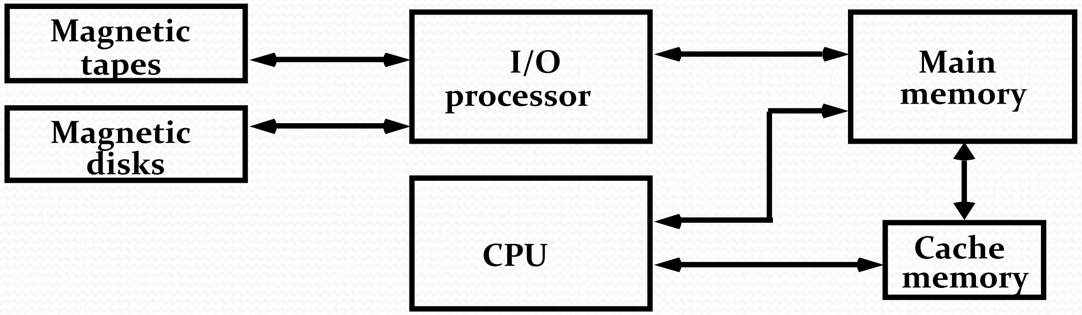
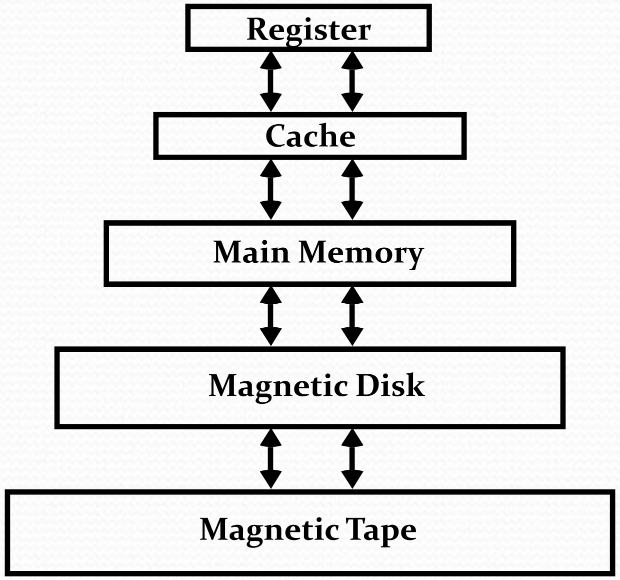
Overview

- 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

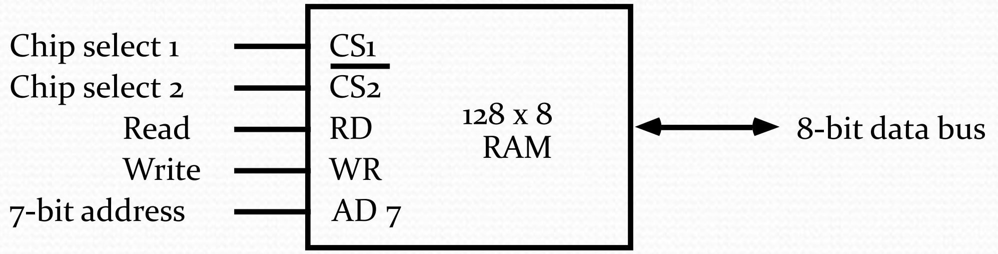




Main Memory

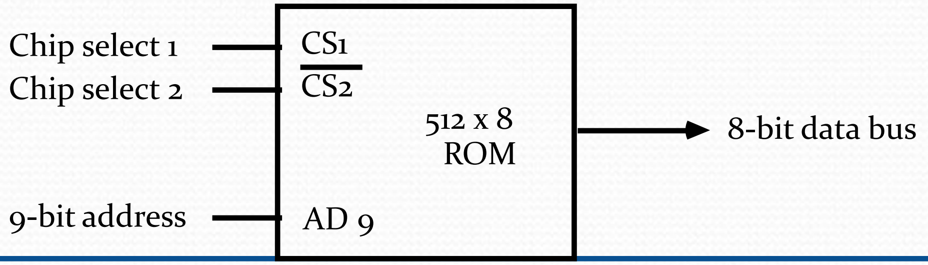
RAM and ROM Chips

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

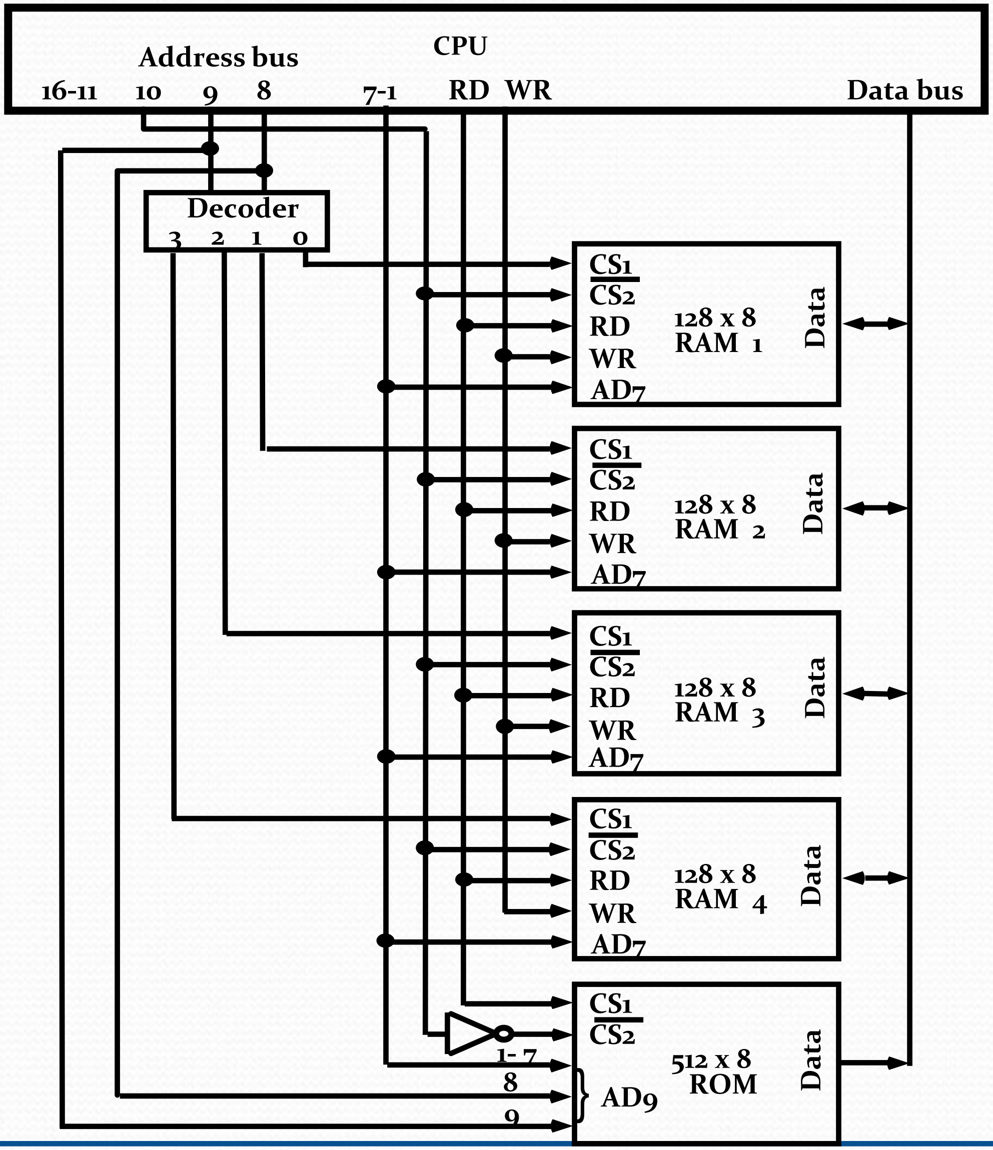
Example: 512 bytes RAM and 512 bytes ROM

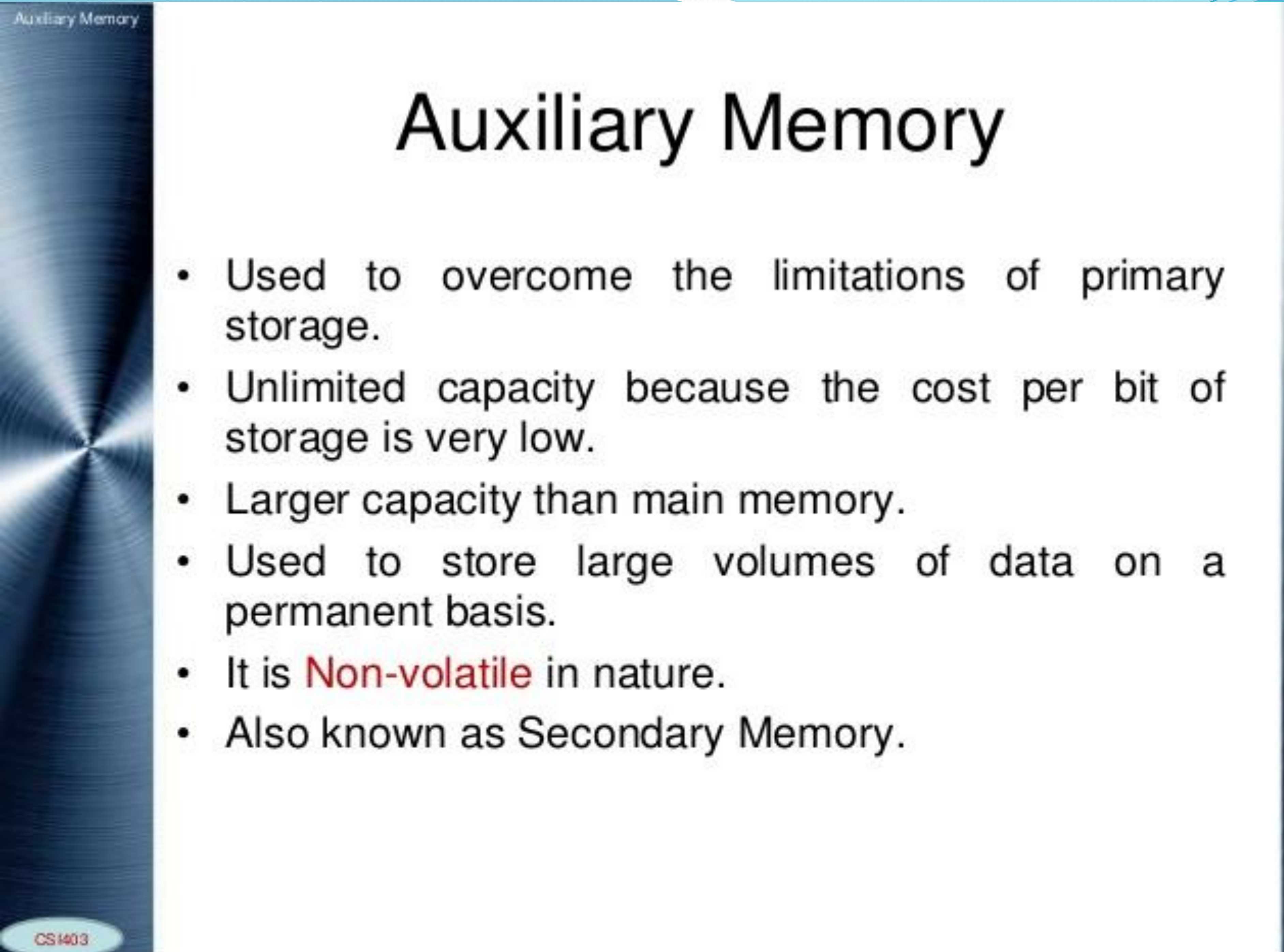
Component	Hexa address	Address bus									
		10	9	8	7	6	5	4	3	2	1
RAM 1	0000 - 007F	0	0	0	x	x	x	x	x	x	x
RAM 2	0080 - 00FF	0	0	1	x	x	x	x	x	x	x
RAM 3	0100 - 017F	0	1	0	x	x	x	x	x	x	x
RAM 4	0180 - 01FF	0	1	1	x	x	x	x	x	x	x
ROM	0200 - 03FF	1	x	x	x	x	x	x	x	x	x

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



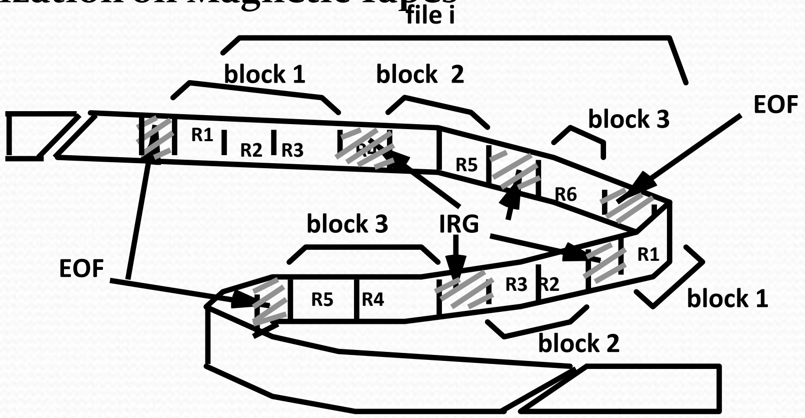


Auxiliary Memory

- Used to overcome the limitations of primary storage.
- Unlimited capacity because the cost per bit of storage is very low.
- Larger capacity than main memory.
- Used to store large volumes of data on a permanent basis.
- It is Non-volatile in nature.
- Also known as Secondary Memory.

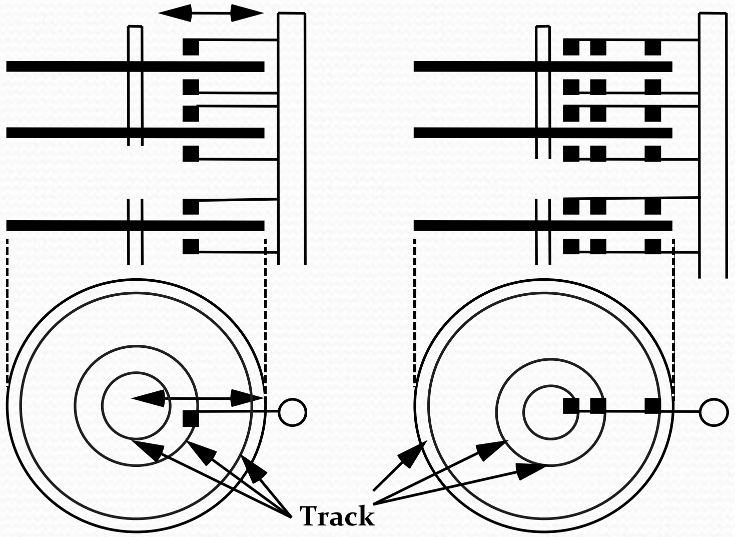
Auxiliary Memory

Information Organization on Magnetic Tapes



Organization of Disk Hardware

Moving Head Disk **Fixed Head Disk**

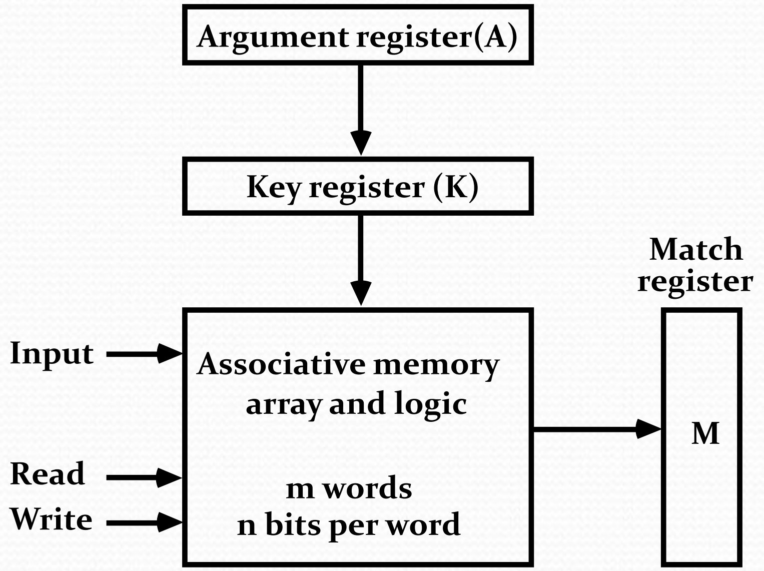


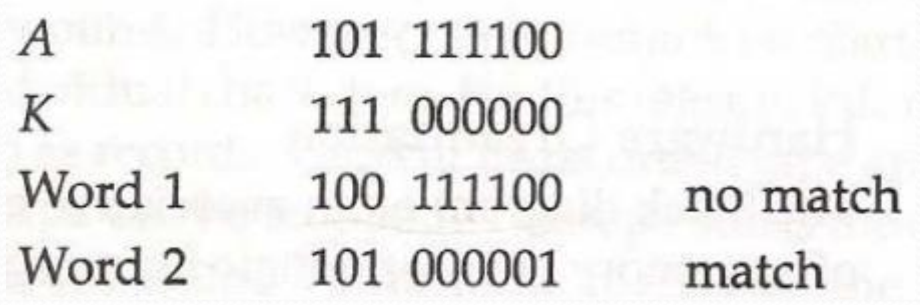
Devices that provide backup storage are called **auxiliary memory**. E.g. Magnetic disks and tapes.

Associative Memory

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

Hardware Organization

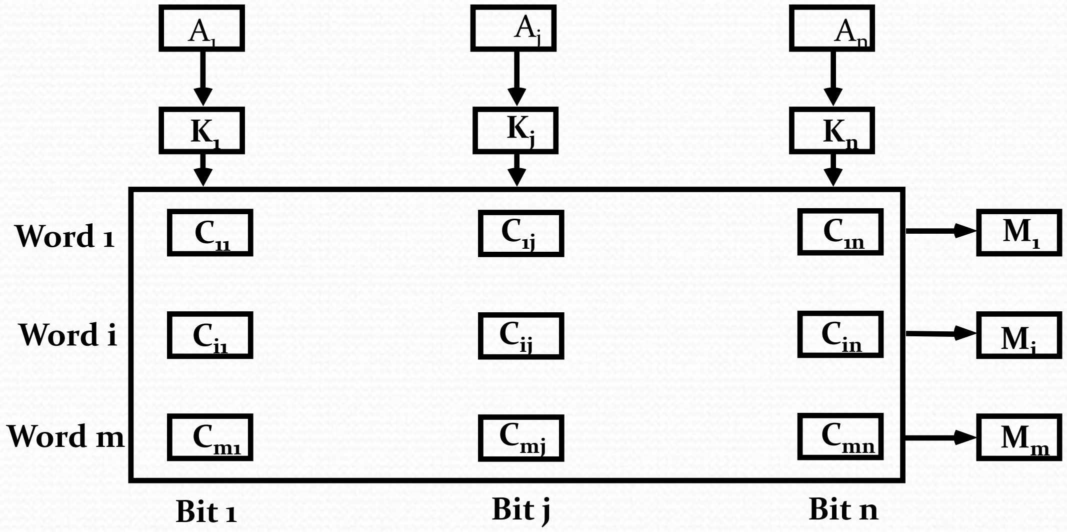




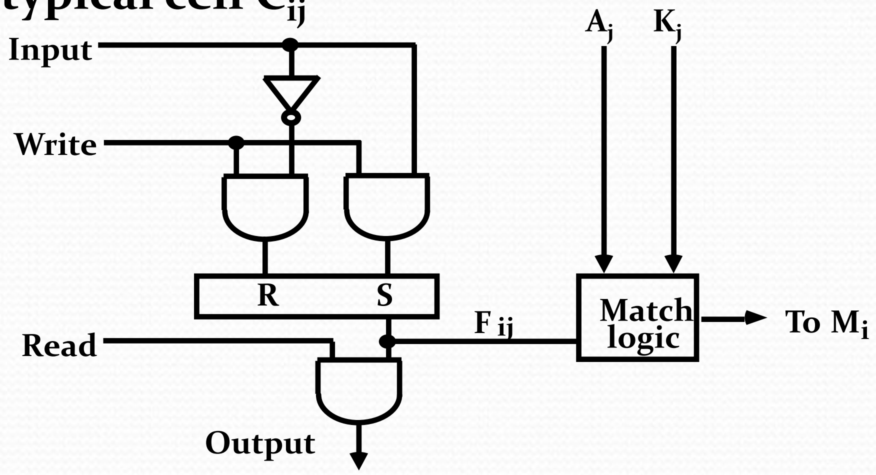
A	101 111100	
K	111 000000	
Word 1	100 111100	no match
Word 2	101 000001	match

Organization of CAM

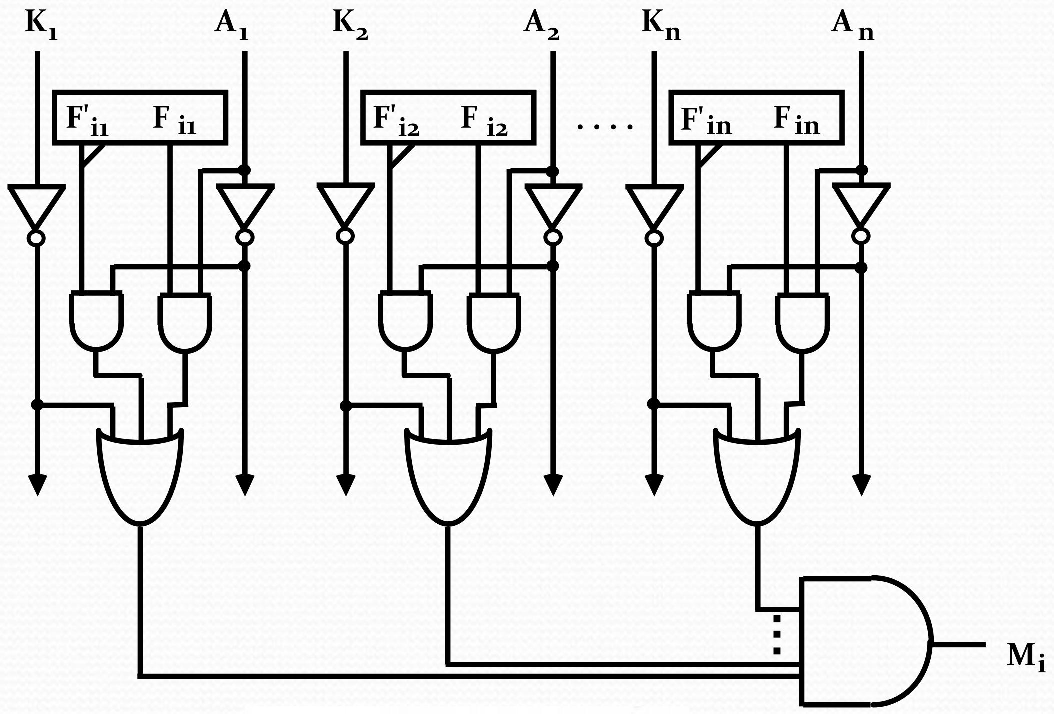
i= word
j-= bit in word



Internal organization of a typical cell C_{ij}



Match Logic



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

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

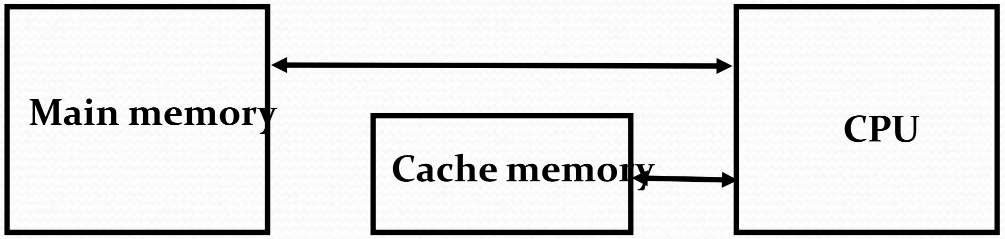
Cache Memory

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

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

Example: $Tc = 0.4\ \mu s$, $Tm = 1.2 \mu s$, $h = 0.85\%$
$Te = 0.4 + (1 - 0.85) * 1.2 = 0.58 \mu s$

Memory and Cache Mapping – (Associative Mapping)

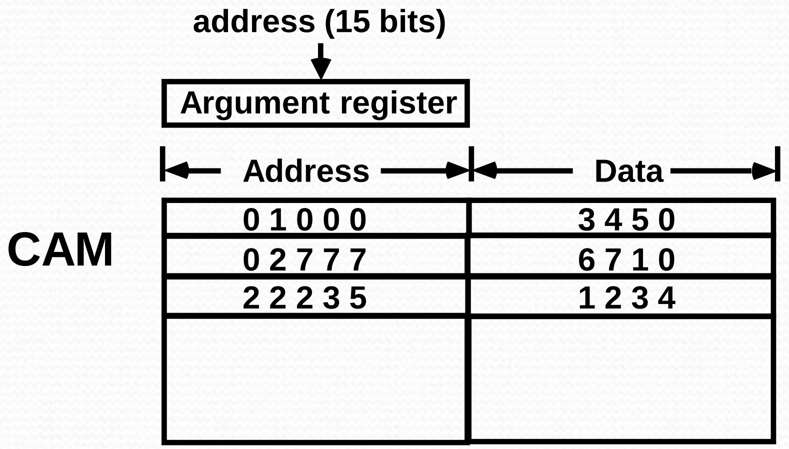
Mapping Function

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

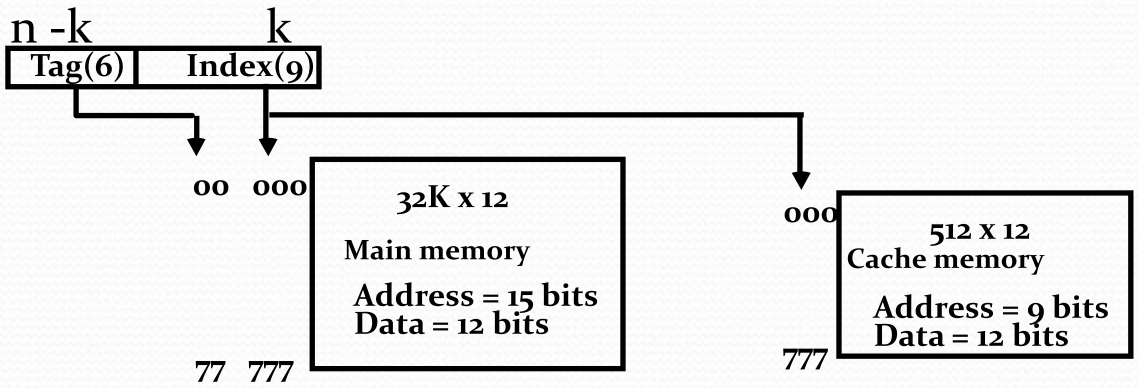


Address	Data
0 1 0 0 0	3 4 5 0
0 2 7 7 7	6 7 1 0
2 2 2 3 5	1 2 3 4

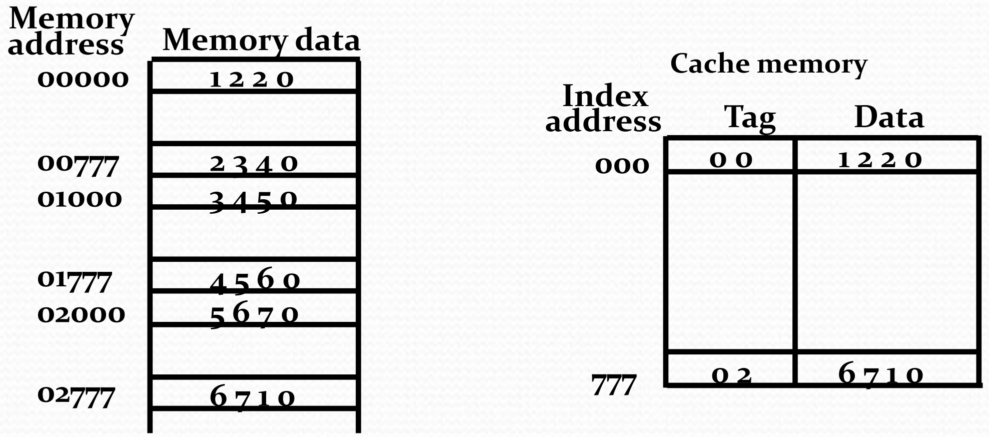
Cache Mapping – direct mapping

- 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

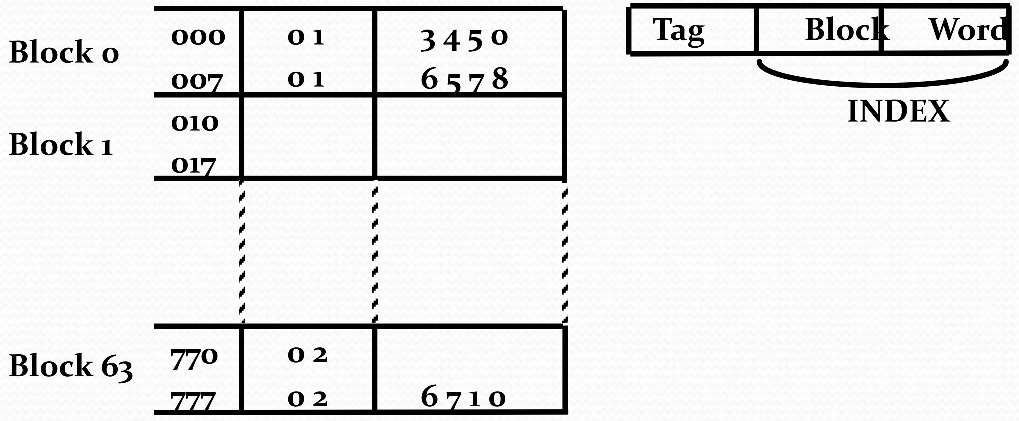


Cache Mapping – direct mapping

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	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

<u>**Write Through**</u>

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

<u>**Write-Back (Copy-Back)**</u>

When writing into memory

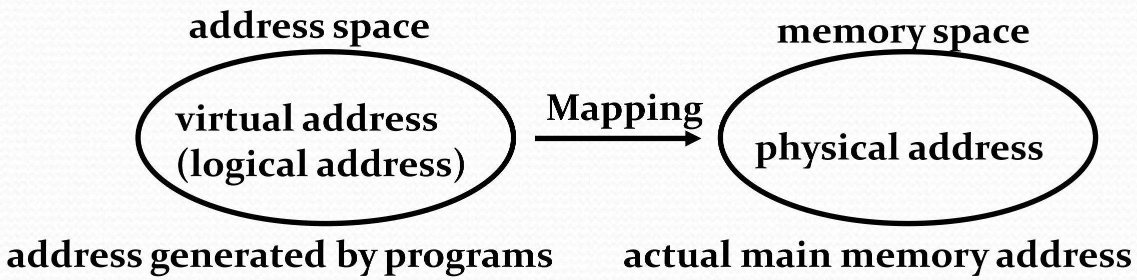
If Hit, only Cache is written
If Miss, missing block is brought to Cache and write into Cache
For a read miss, candidate block must be written back to the memory

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

Virtual Memory

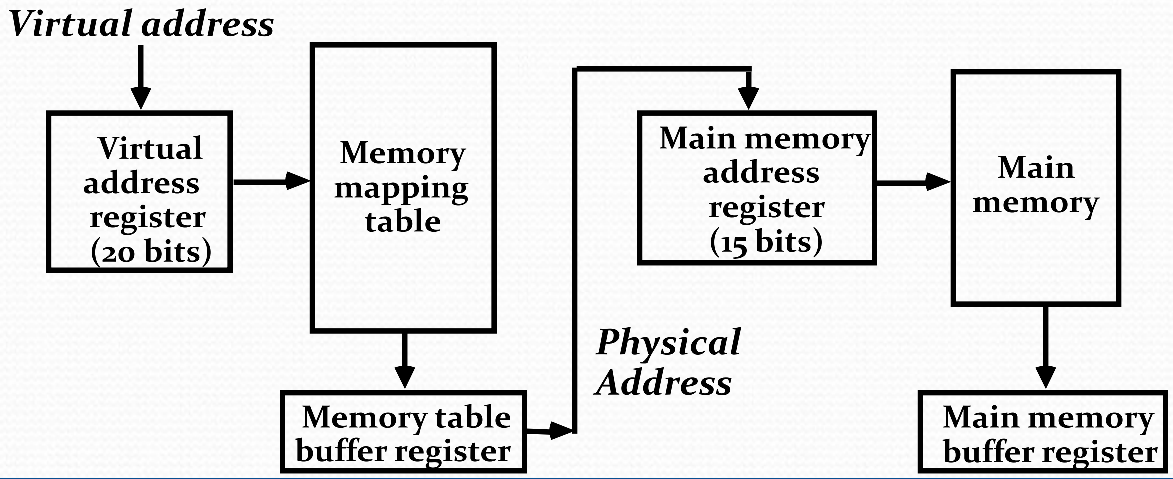
Give the programmer the illusion that the system has a very large memory, even though the computer actually has a relatively small main memory

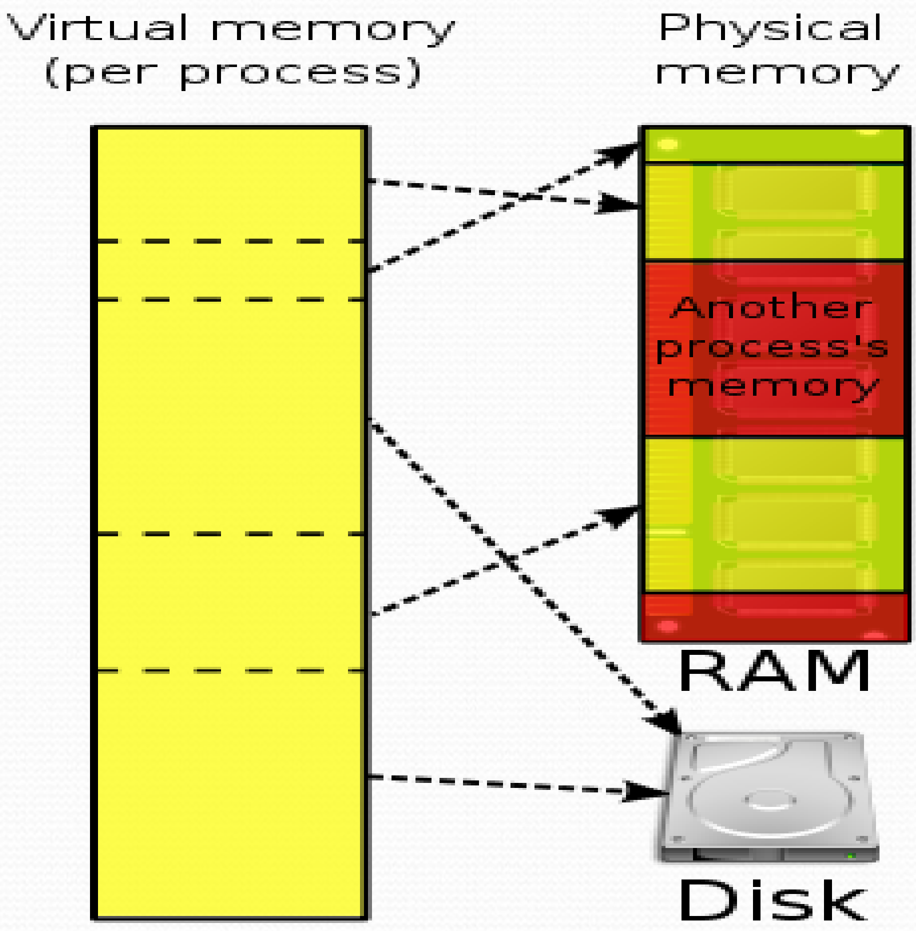
Address Space(Logical) and Memory Space(Physical)



Address Mapping

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


Virtual address
Virtual address register (20 bits)
Memory mapping table
Memory table buffer register
Physical Address
Main memory address register (15 bits)
Main memory
Main memory buffer register

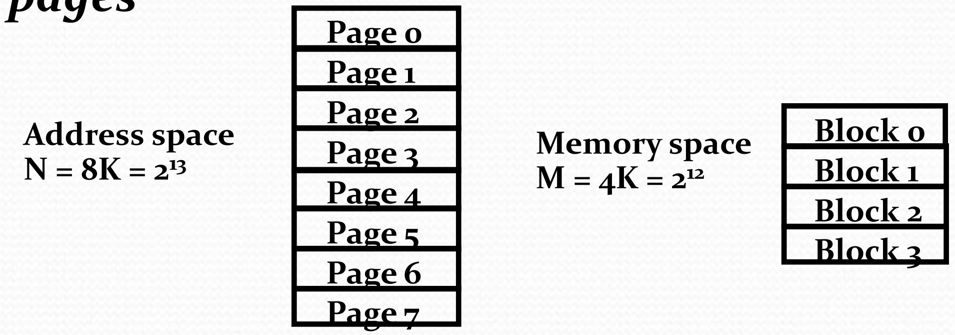


Virtual memory to form a large range of contiguous addresses.

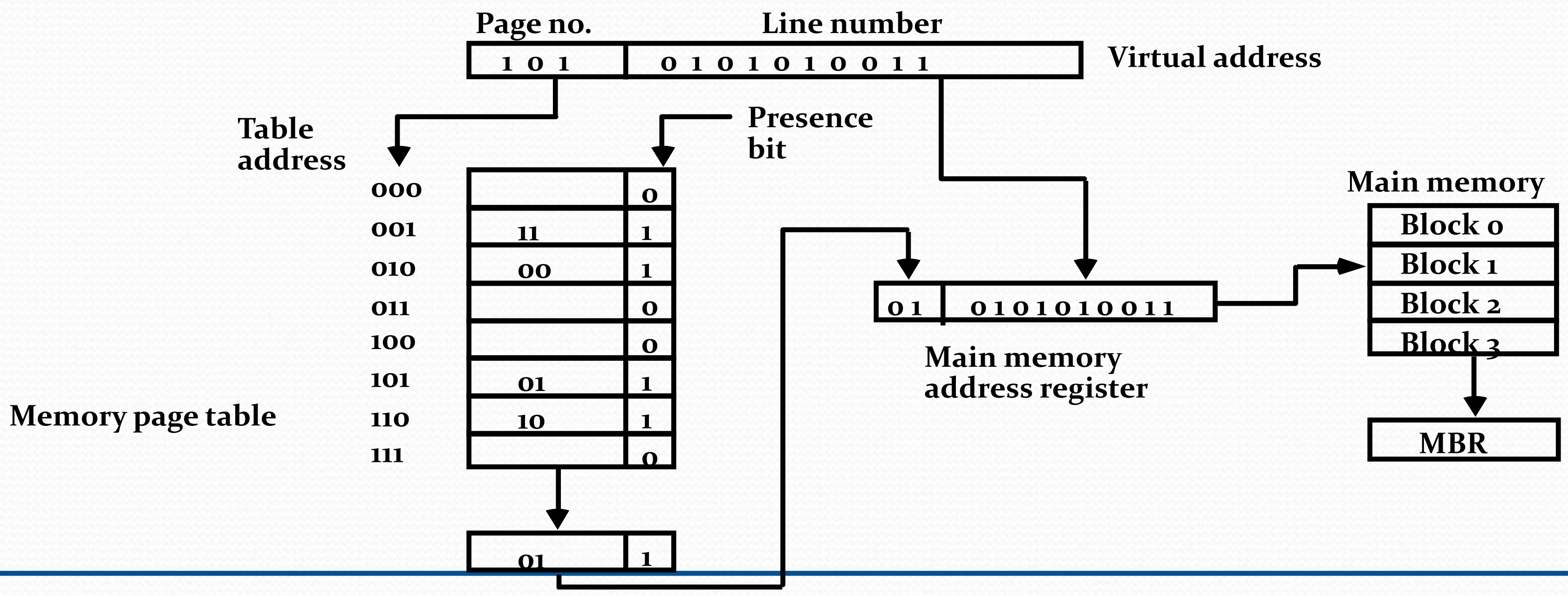
Address Mapping

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

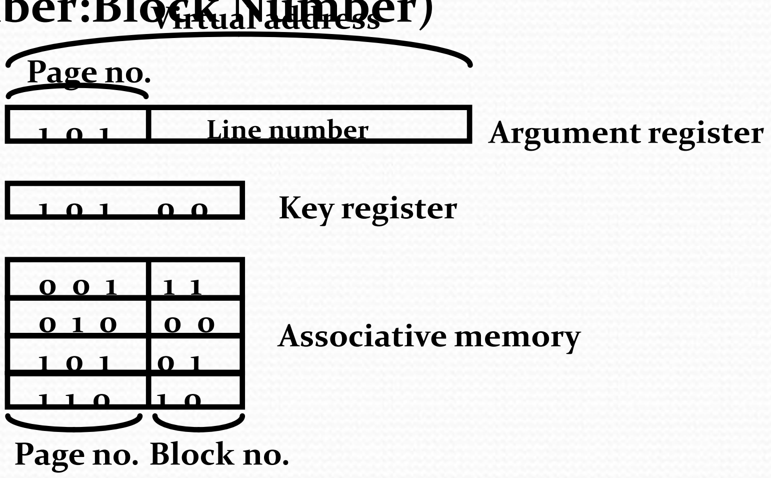


Associative Memory Page Table

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

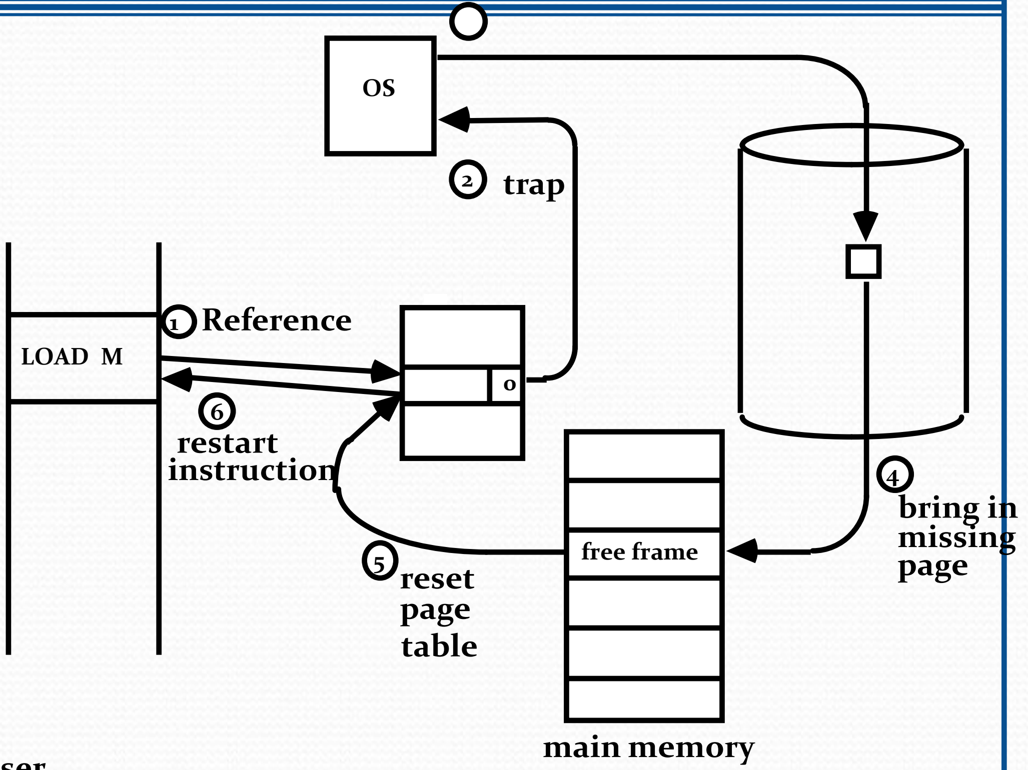
- More efficient method is m-entry Page Table
Page Table made of an Associative Memory
m words; (Page Number:Block Number)



Page Fault
Page number cannot be found in the Page Table

Page Fault

1. Trap to the OS
2. Save the user registers and program state
3. Determine that the interrupt was a page fault
4. Check that the page reference was legal and determine the location of the page on the backing store(disk)
5. Issue a read from the backing store to a free frame
   a. Wait in a queue for this device until serviced
   b. Wait for the device seek and/or latency time
   c. Begin the transfer of the page to a free frame
6. While waiting, the CPU may be allocated to some other process
7. Interrupt from the backing store (I/O completed)
8. Save the registers and program state for the other user
9. Determine that the interrupt was from the backing store
10. Correct the page tables (the desired page is now in memory)
11. Wait for the CPU to be allocated to this process again
12. Restore the user registers, program state, and new page table, then resume the interrupted instruction.



Processor architecture should provide the ability to restart any instruction after a page fault.

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

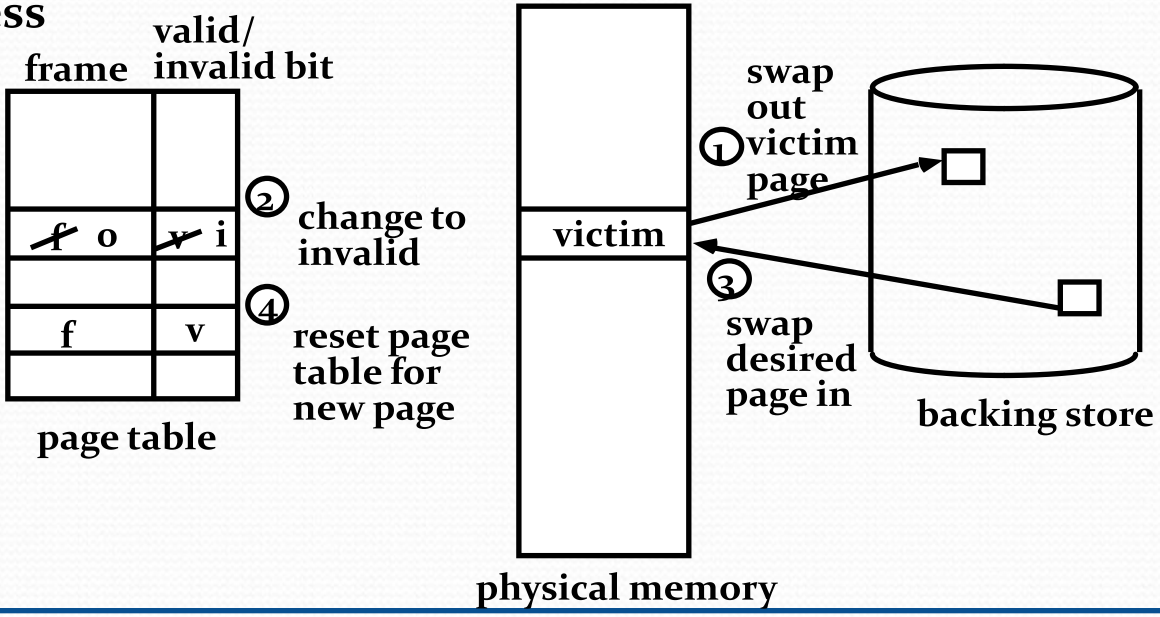
- If there is a free frame, use it

- Otherwise, use a page-replacement algorithm to select a *victim* frame

- Write the victim page to the backing store

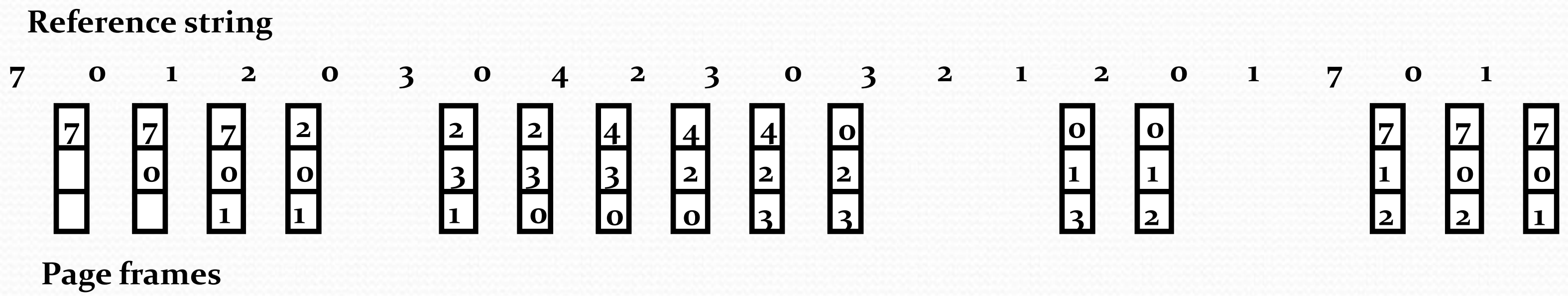
3. Read the desired page into the (newly) free frame

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

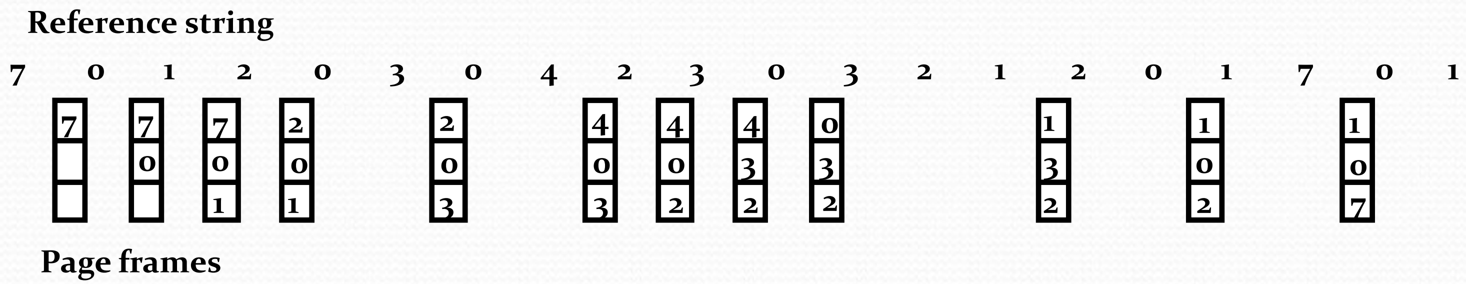
Page frames

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

MICROARCHITECTURE

COMPUTER ORGANIZATION AND DESIGN

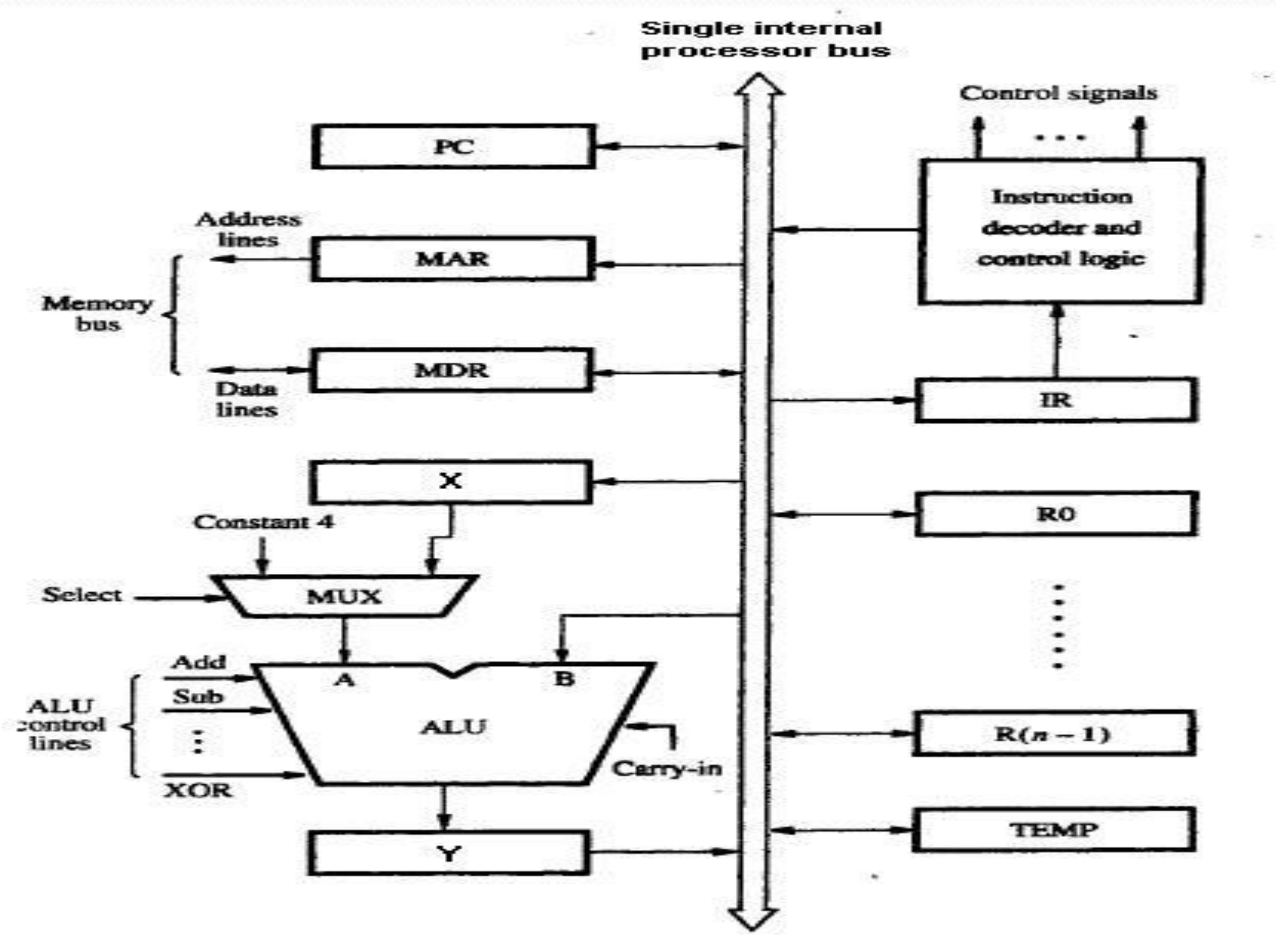
COURSE CODE- CSE211

- Microarchitecture or µarch is the way a given instruction set architecture (ISA) is implemented in a particular processor.
- A given ISA may be implemented with different microarchitectures; implementations may vary due to different goals of a given design or due to shifts in technology.
- Computer architecture is the combination of microarchitecture and instruction set.

Execution units in Microprocessor:

- Execution units are essential to microarchitecture.
- Execution units include arithmetic logic units (ALU), floating point units (FPU), load/store units, branch prediction, and SIMD.
- These units perform the operations or calculations of the processor. The choice of the number of execution units, their latency and throughput is a central micro architectural design task.
- The size, latency, throughput and connectivity of memories within the system are also micro architectural decisions.

A microarchitecture organized around a single bus

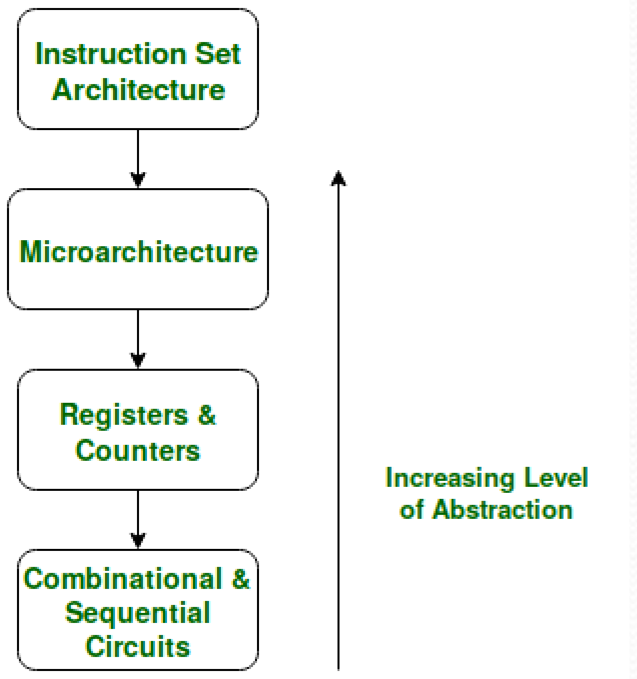


Single Bus Organization

Micro-architecture

An **ISA** describes the **design of a Computer** in terms of the **basic operations** it must support. The ISA is not concerned with the implementation specific details of a computer. It is only concerned with the set or collection of basic operations the computer must support. For example the AMD Athlon and the Core 2 Duo processors have entirely different implementations but they support more or less the same set of basic operations as defined in the x86 Instruction Set.

Micro architectural level lies just below the **ISA** level and hence is concerned with the implementation of the basic operations to be supported by the Computer as defined by the **ISA**. Therefore we can say that the AMD Athlon and Core 2 Duo processors are based on the same ISA but have different microarchitectures with different performance and efficiencies.


Instruction Set Architecture
Microarchitecture
Registers & Counters
Combinational & Sequential Circuits
Increasing Level of Abstraction