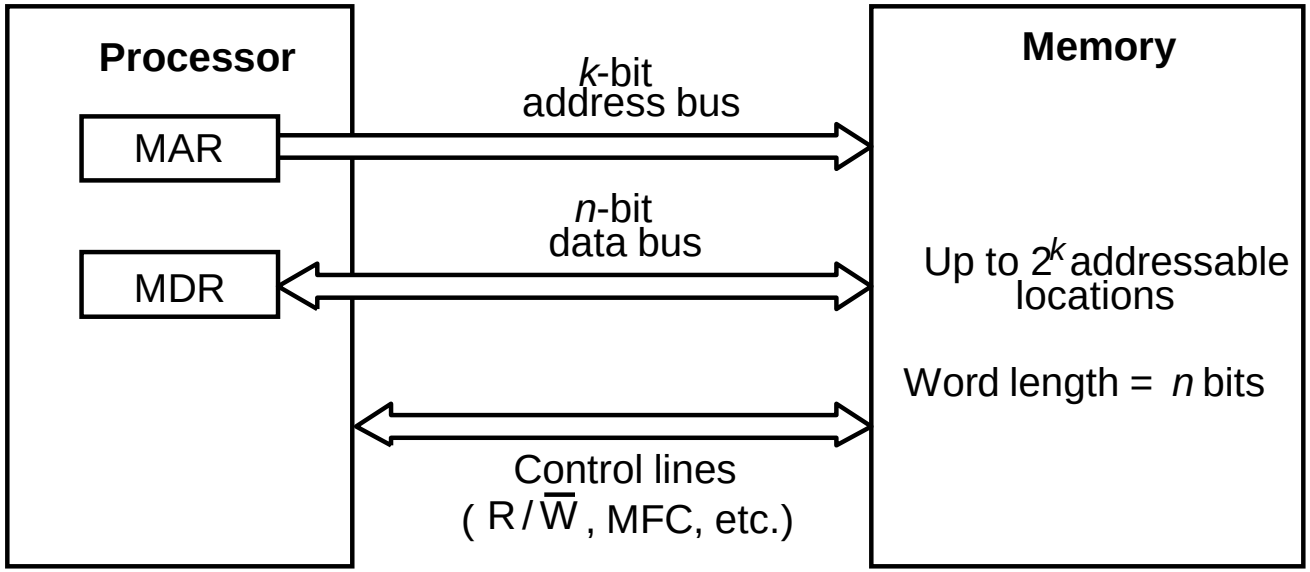
Fundamental Concepts

The Memory System

Some basic concepts

- Maximum size of the Main Memory
- byte-addressable
- CPU-Main Memory Connection



Some basic concepts(Contd.,)

- Measures for the speed of a memory:
  - memory access time.
  - memory cycle time.
- An important design issue is to provide a computer system with as large and fast a memory as possible, within a given cost target.
- Several techniques to increase the effective size and speed of the memory:
  - Cache memory (to increase the effective speed).
  - Virtual memory (to increase the effective size).

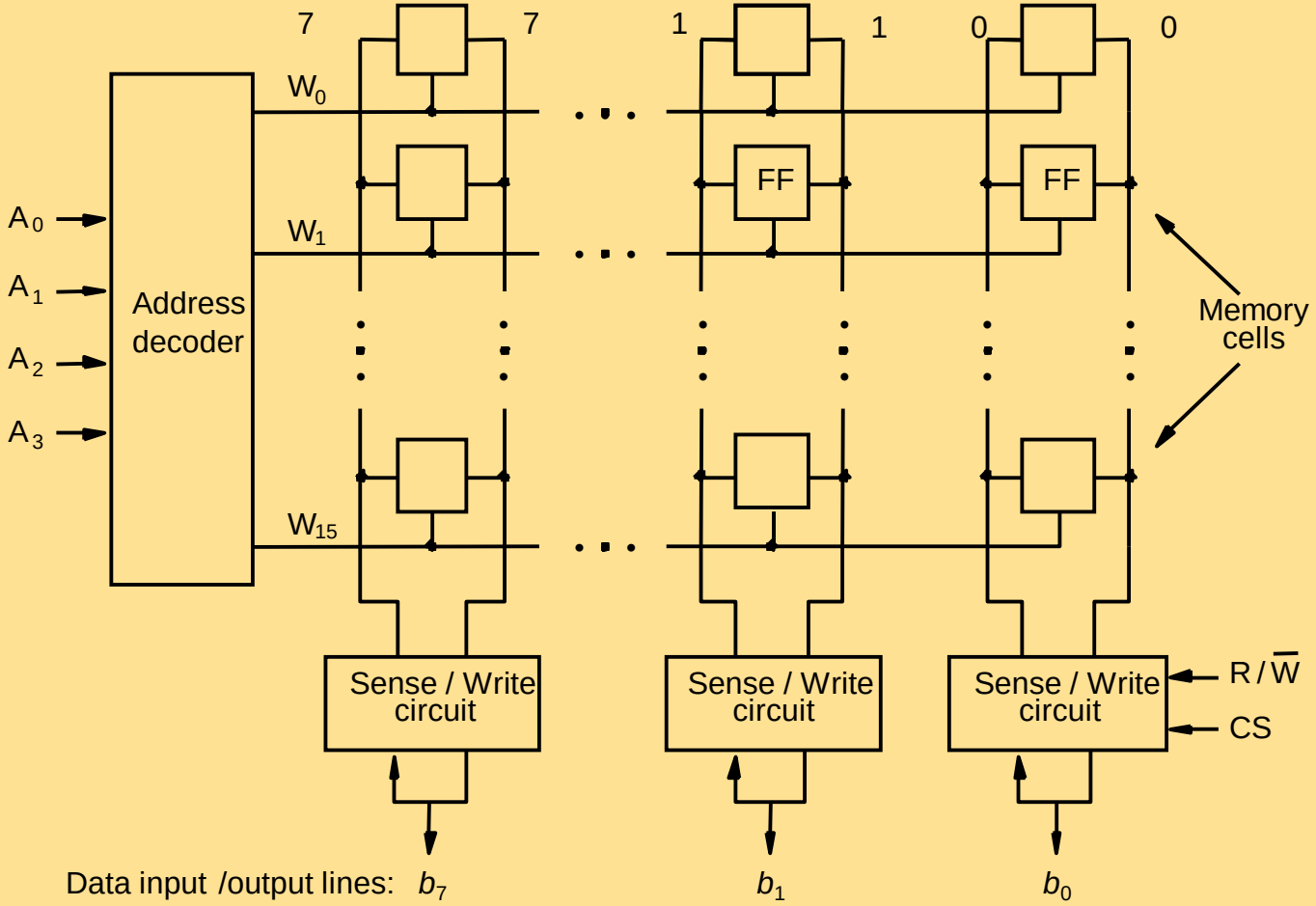
Semiconductor RAM memories

The Memory System

Internal organization of memory chips

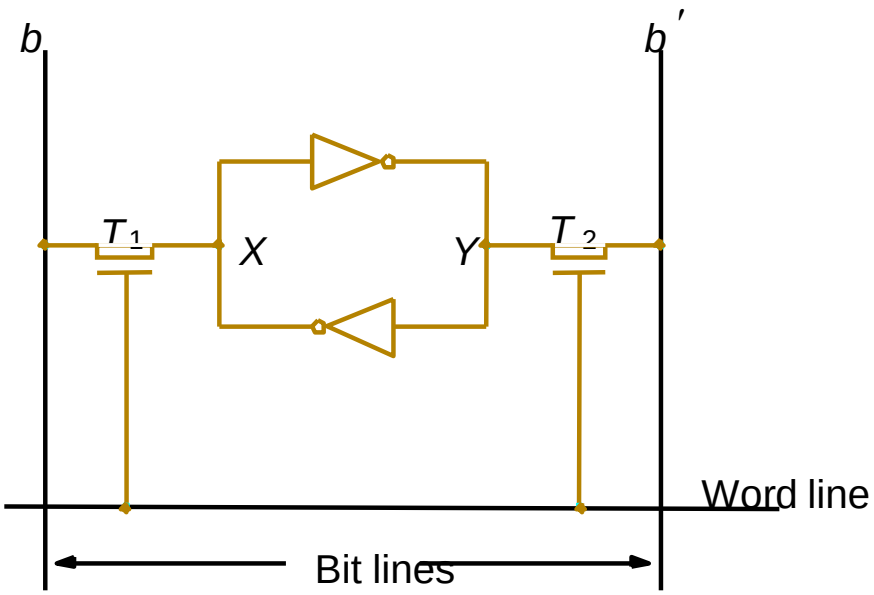
- Each memory cell can hold one bit of information.
- Memory cells are organized in the form of an array.
- One row is one memory word.
- All cells of a row are connected to a common line, known as the "word line".
- Word line is connected to the address decoder.
- Sense/write circuits are connected to the data input/output lines of the memory chip.

Internal organization of memory chips (Contd.,)



SRAM Cell

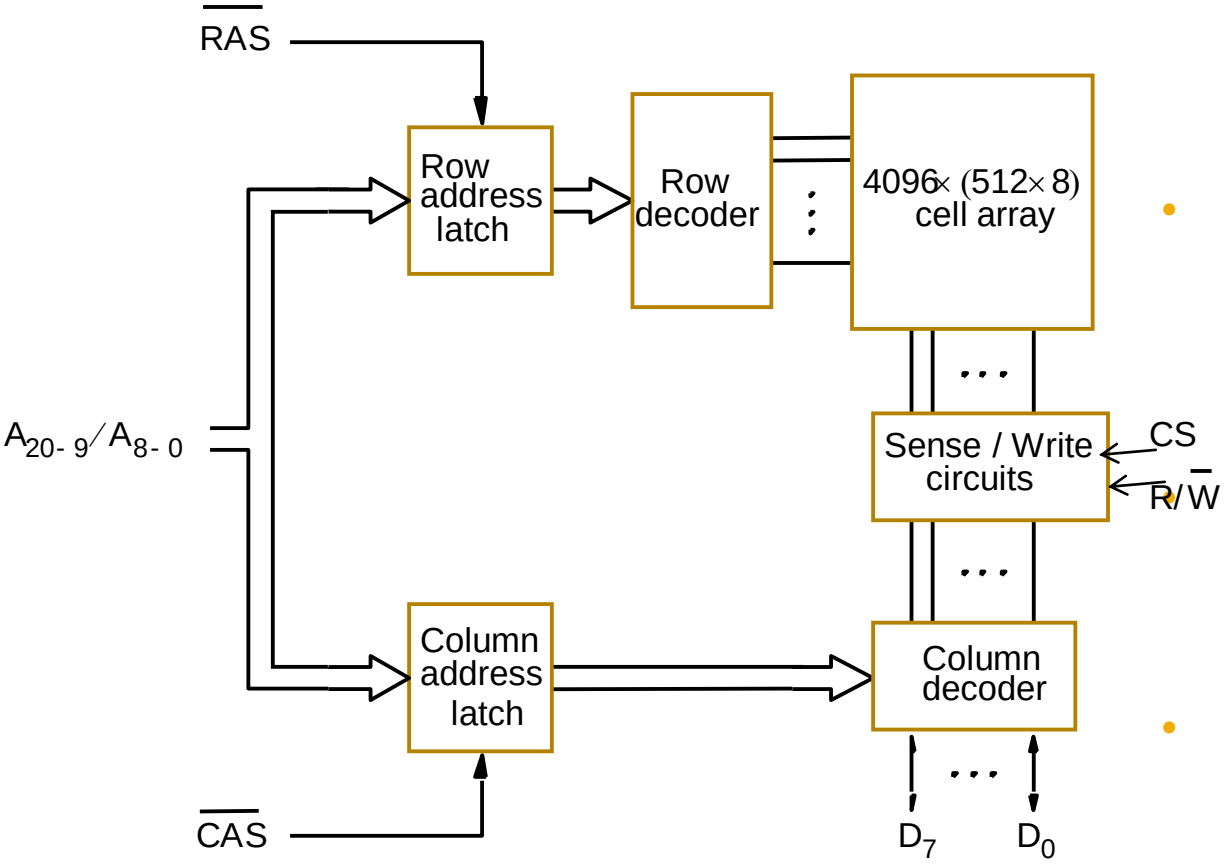
- Two transistor inverters are cross connected to implement a basic flip-flop.
- The cell is connected to one word line and two bits lines by transistors T1 and T2
- When word line is at ground level, the transistors are turned off and the latch retains its state
- Read operation: In order to read state of SRAM cell, the word line is activated to close switches T1 and T2. Sense/Write circuits at the bottom monitor the state of b and b'



Asynchronous DRAMs

- Static RAMs (SRAMs):
  - Consist of circuits that are capable of retaining their state as long as the power is applied.
  - Volatile memories, because their contents are lost when power is interrupted.
  - Access times of static RAMs are in the range of few nanoseconds.
  - However, the cost is usually high.
- Dynamic RAMs (DRAMs):
  - Do not retain their state indefinitely.
  - Contents must be periodically refreshed.
  - Contents may be refreshed while accessing them for reading.

Asynchronous DRAMs

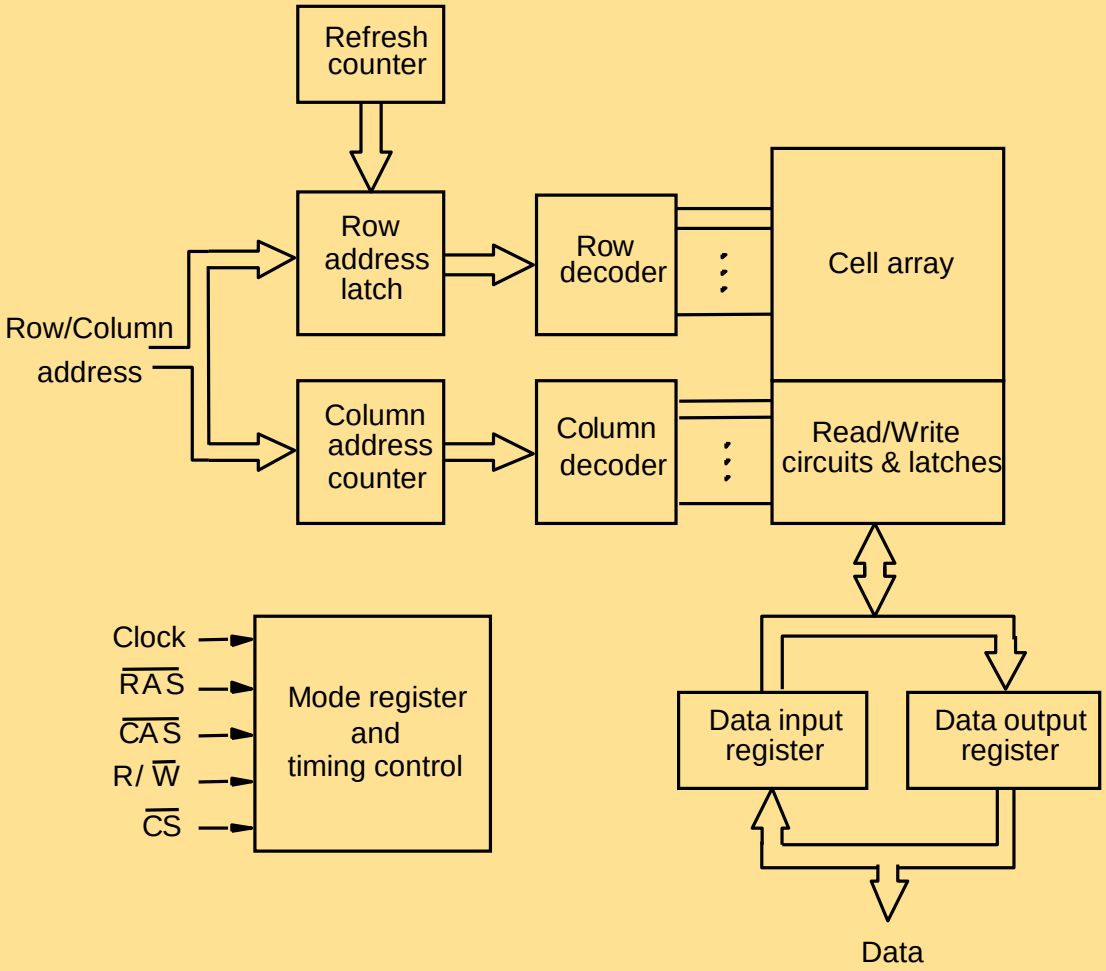


- *Each row can store 512 bytes. 12 bits to select a row, and 9 bits to select a group in a row. Total of 21 bits.*
- *First apply the row address, RAS signal latches the row address. Then apply the column address, CAS signal latches the address.*
- *Timing of the memory unit is controlled by a specialized unit which generates RAS and CAS.*
- *This is asynchronous DRAM*

Fast Page Mode

- Suppose if we want to access the consecutive bytes in the selected row.
- This can be done without having to reselect the row.
  - Add a latch at the output of the sense circuits in each row.
  - All the latches are loaded when the row is selected.
  - Different column addresses can be applied to select and place different bytes on the data lines.
- Consecutive sequence of column addresses can be applied under the control signal CAS, without reselecting the row.
  - Allows a block of data to be transferred at a much faster rate than random accesses.
  - A small collection/group of bytes is usually referred to as a block.
- This transfer capability is referred to as the fast page mode feature.

Synchronous DRAMs

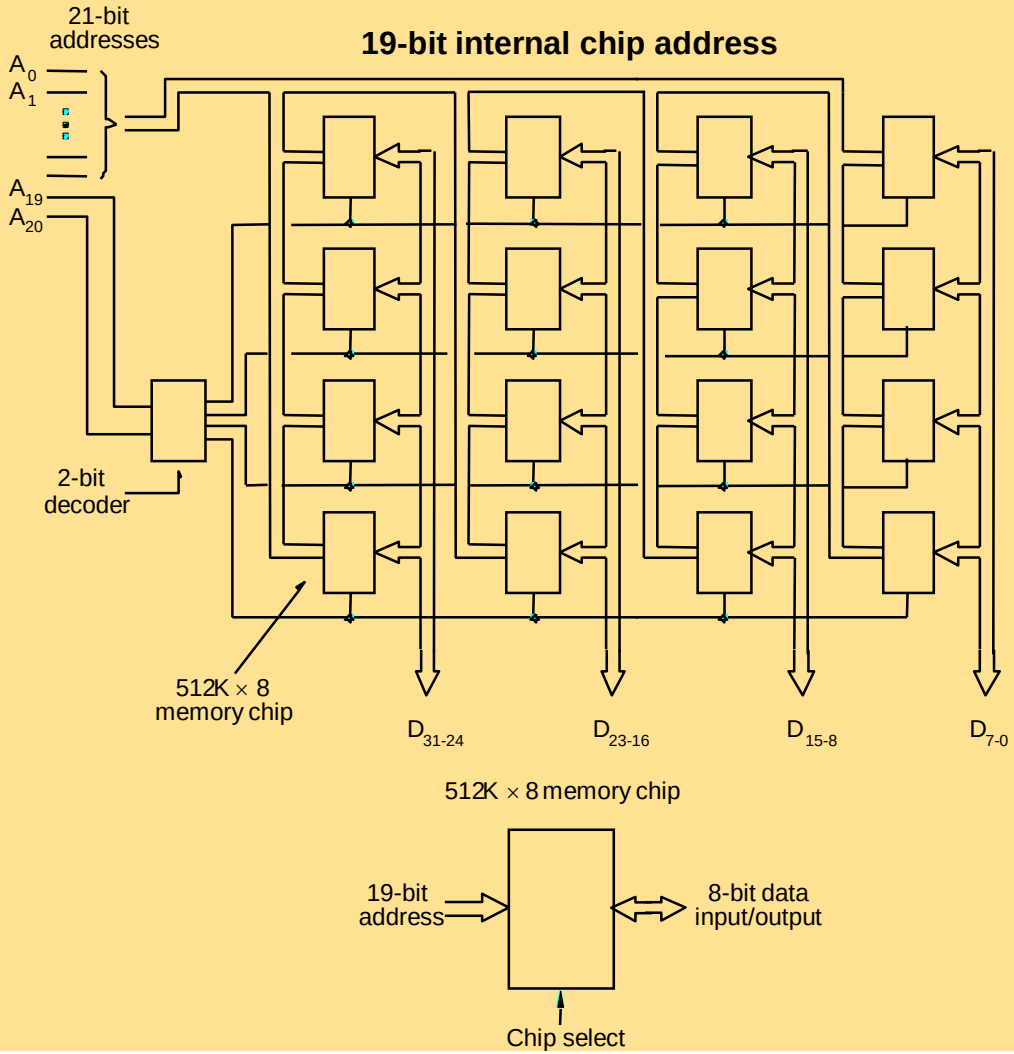


- *Operation is directly synchronized with processor clock signal.*
- *The outputs of the sense circuits are connected to a latch.*
- *During a Read operation, the contents of the cells in a row are loaded onto the latches.*
- *During a refresh operation, the contents of the cells are refreshed without changing the contents of the latches.*
- *Data held in the latches correspond to the selected columns are transferred to the output.*
- *For a burst mode of operation, successive columns are selected using column address counter and clock. $\overline{CAS}$ signal need not be generated externally. A new data is placed during raising edge of the clock*

Latency, Bandwidth, and DDRSDRAMs

- Memory latency is the time it takes to transfer a word of data to or from memory
- Memory bandwidth is the number of bits or bytes that can be transferred in one second.
- DDRSDRAMs
  - Cell array is organized in two banks

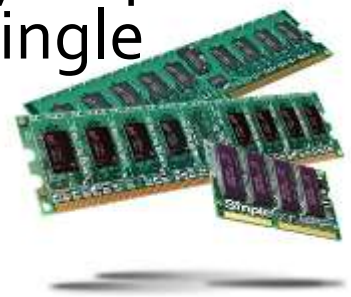
Static memories



Implement a memory unit of 2M words of 32 bits each.
Use 512x8 static memory chips.
Each column consists of 4 chips.
Each chip implements one byte position.
A chip is selected by setting its chip select control line to 1.
Selected chip places its data on the data output line, outputs of other chips are in high impedance state.
21 bits to address a 32-bit word.
High order 2 bits are needed to select the row, by activating the four Chip Select signals.
19 bits are used to access specific byte locations inside the selected chip.

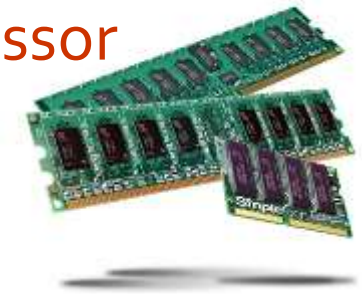
Dynamic memories

- Large dynamic memory systems can be implemented using DRAM chips in a similar way to static memory systems.
- Placing large memory systems directly on the motherboard will occupy a large amount of space.
  - Also, this arrangement is inflexible since the memory system cannot be expanded easily.
- Packaging considerations have led to the development of larger memory units known as SIMMs (Single In-line Memory Modules) and DIMMs (Dual In-line Memory Modules).
- Memory modules are an assembly of memory chips on a small board that plugs vertically onto a single socket on the motherboard.
  - Occupy less space on the motherboard.
  - Allows for easy expansion by replacement.

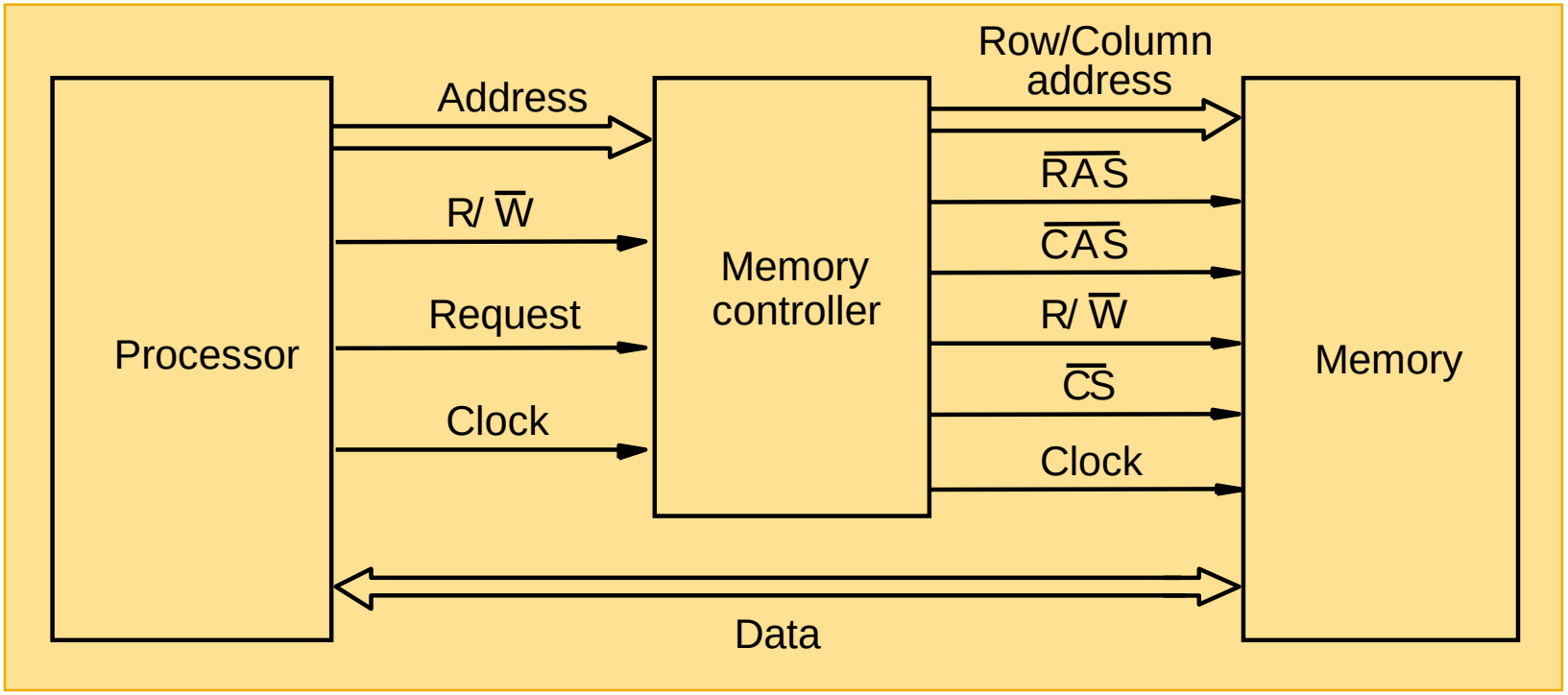


Memory controller

- Recall that in a dynamic memory chip, to reduce the number of pins, multiplexed addresses are used.
- Address is divided into two parts:
  - High-order address bits select a row in the array.
  - They are provided first, and latched using RAS signal.
  - Low-order address bits select a column in the row.
  - They are provided later, and latched using CAS signal.
- However, a processor issues all address bits at the same time.
- In order to achieve the multiplexing, memory controller circuit is inserted between the processor and memory.



Memory controller (contd..)



Read-Only Memories (ROMs)

The Memory System

Read-Only Memories (ROMs)

- SRAM and SDRAM chips are volatile:
  - Lose the contents when the power is turned off.
- Many applications need memory devices to retain contents after the power is turned off.
  - For example, computer is turned on, the operating system must be loaded from the disk into the memory.
  - Store instructions which would load the OS from the disk.
  - Need to store these instructions so that they will not be lost after the power is turned off.
  - We need to store the instructions into a non-volatile memory.
- Non-volatile memory is read in the same manner as volatile memory.
  - Separate writing process is needed to place information in this memory.
  - Normal operation involves only reading of data, this type of memory is called Read-Only memory (ROM).

Read-Only Memories (Contd.,)

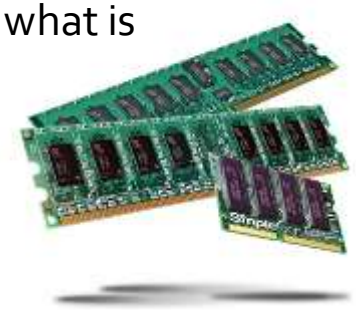
- Read-Only Memory:
  - Data are written into a ROM when it is manufactured.
- Programmable Read-Only Memory (PROM):
  - Allow the data to be loaded by a user.
  - Process of inserting the data is irreversible.
  - Storing information specific to a user in a ROM is expensive.
  - Providing programming capability to a user may be better.
- Erasable Programmable Read-Only Memory (EPROM):
  - Stored data to be erased and new data to be loaded.
  - Flexibility, useful during the development phase of digital systems.
  - Erasable, reprogrammable ROM.
  - Erasure requires exposing the ROM to UV light.

Read-Only Memories (Contd.,)

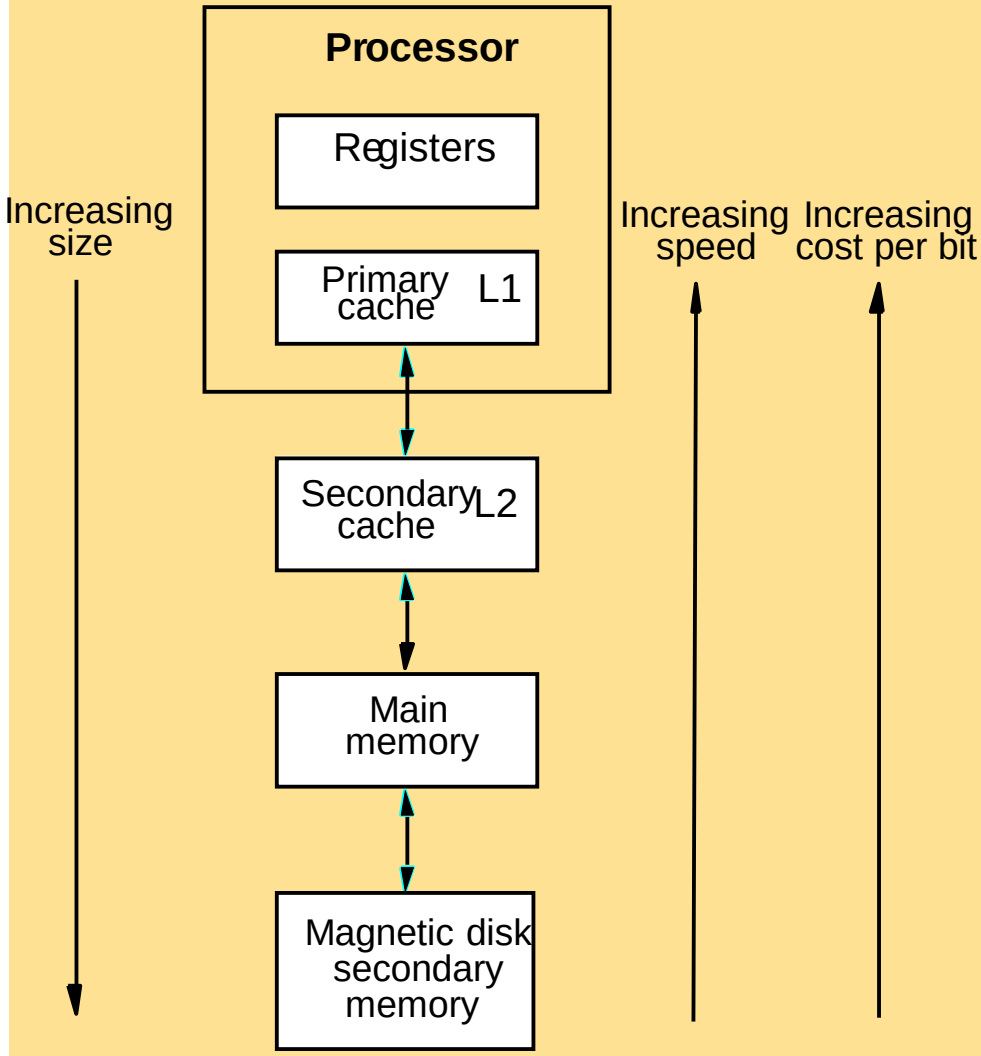
- Electrically Erasable Programmable Read-Only Memory (EEPROM):
  - To erase the contents of EPROMs, they have to be exposed to ultraviolet light.
  - Physically removed from the circuit.
  - EEPROMs the contents can be stored and erased electrically.
- Flash memory:
  - Has similar approach to EEPROM.
  - Read the contents of a single cell, but write the contents of an entire block of cells.
  - Flash devices have greater density.
    - Higher capacity and low storage cost per bit.
  - Power consumption of flash memory is very low, making it attractive for use in equipment that is battery-driven.
  - Single flash chips are not sufficiently large, so larger memory modules are implemented using flash cards and flash drives.

Speed, Size, and Cost

- A big challenge in the design of a computer system is to provide a sufficiently large memory, with a reasonable speed at an affordable cost.
- Static RAM:
  - Very fast, but expensive, because a basic SRAM cell has a complex circuit making it impossible to pack a large number of cells onto a single chip.
- Dynamic RAM:
  - Simpler basic cell circuit, hence are much less expensive, but significantly slower than SRAMs.
- Magnetic disks:
  - Storage provided by DRAMs is higher than SRAMs, but is still less than what is necessary.
  - Secondary storage such as magnetic disks provide a large amount of storage, but is much slower than DRAMs.



Memory Hierarchy



- *Fastest access is to the data held in processor registers. Registers are at the top of the memory hierarchy.*
- *Relatively small amount of memory that can be implemented on the processor chip. This is processor cache.*
- *Two levels of cache. Level 1 (L1) cache is on the processor chip. Level 2 (L2) cache is in between main memory and processor.*
- *Next level is main memory, implemented as SIMMs. Much larger, but much slower than cache memory.*
- *Next level is magnetic disks. Huge amount of inexepensive storage.*
- *Speed of memory access is critical, the idea is to bring instructions and data that will be used in the near future as close to the processor as possible.*

Cache Memories

The Memory System

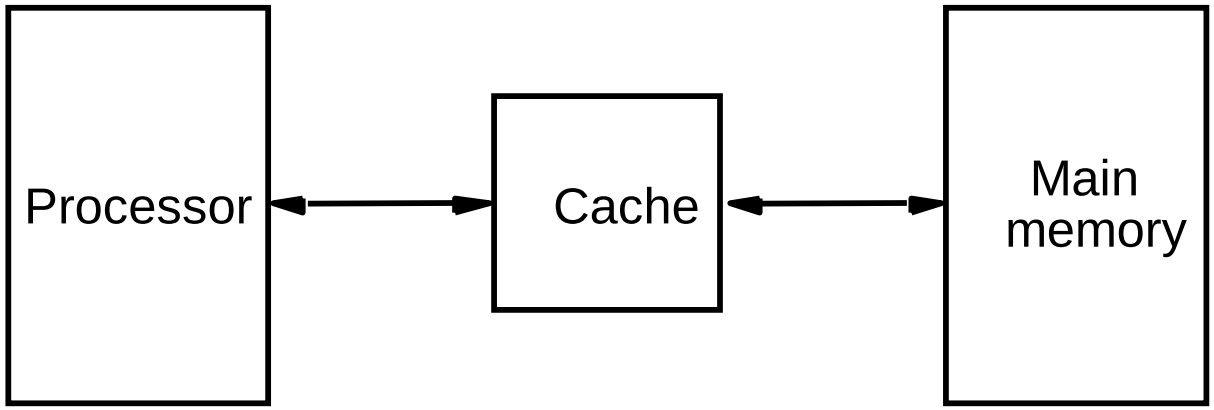
Cache Memories

- Processor is much faster than the main memory.
  - As a result, the processor has to spend much of its time waiting while instructions and data are being fetched from the main memory.
  - Major obstacle towards achieving good performance.
- Speed of the main memory cannot be increased beyond a certain point.
- Cache memory is an architectural arrangement which makes the main memory appear faster to the processor than it really is.
- Cache memory is based on the property of computer programs known as "locality of reference".

Locality of Reference

- Analysis of programs indicates that many instructions in localized areas of a program are executed repeatedly during some period of time, while the others are accessed relatively less frequently.
  - These instructions may be the ones in a loop, nested loop or few procedures calling each other repeatedly.
  - This is called "locality of reference".
- Temporal locality of reference:
  - Recently executed instruction is likely to be executed again very soon.
- Spatial locality of reference:
  - Instructions with addresses close to a recently instruction are likely to be executed soon.

Cache memories



- *Processor issues a Read request, a block of words is transferred from the main memory to the cache, one word at a time.*
- *Subsequent references to the data in this block of words are found in the cache.*
- *At any given time, only some blocks in the main memory are held in the cache. Which blocks in the main memory are in the cache is determined by a "mapping function".*
- *When the cache is full, and a block of words needs to be transferred from the main memory, some block of words in the cache must be replaced. This is determined by a "replacement algorithm".*

Cache hit

- *Existence of a cache is transparent to the processor. The processor issues Read and Write requests in the same manner.*
- *If the data is in the cache it is called a Read or Write hit.*
- *Read hit:*
  - *The data is obtained from the cache.*
- *Write hit:*
  - *Cache has a replica of the contents of the main memory.*
  - *Contents of the cache and the main memory may be updated simultaneously. This is the write-through protocol.*
  - *Update the contents of the cache, and mark it as updated by setting a bit known as the dirty bit or modified bit. The contents of the main memory are updated when this block is replaced. This is write-back or copy-back protocol.*

Cache miss

- *If the data is not present in the cache, then a Read miss or Write miss occurs.*
- *Read miss:*
  - *Block of words containing this requested word is transferred from the memory.*
  - *After the block is transferred, the desired word is forwarded to the processor.*
  - *The desired word may also be forwarded to the processor as soon as it is transferred without waiting for the entire block to be transferred. This is called load-through or early-restart.*
- *Write-miss:*
  - *Write-through protocol is used, then the contents of the main memory are updated directly.*
  - *If write-back protocol is used, the block containing the addressed word is first brought into the cache. The desired word is overwritten with new information.*

Cache Coherence Problem

- *A bit called as "valid bit" is provided for each block.*
- *If the block contains valid data, then the bit is set to 1, else it is 0.*
- *Valid bits are set to 0, when the power is just turned on.*
- *When a block is loaded into the cache for the first time, the valid bit is set to 1.*

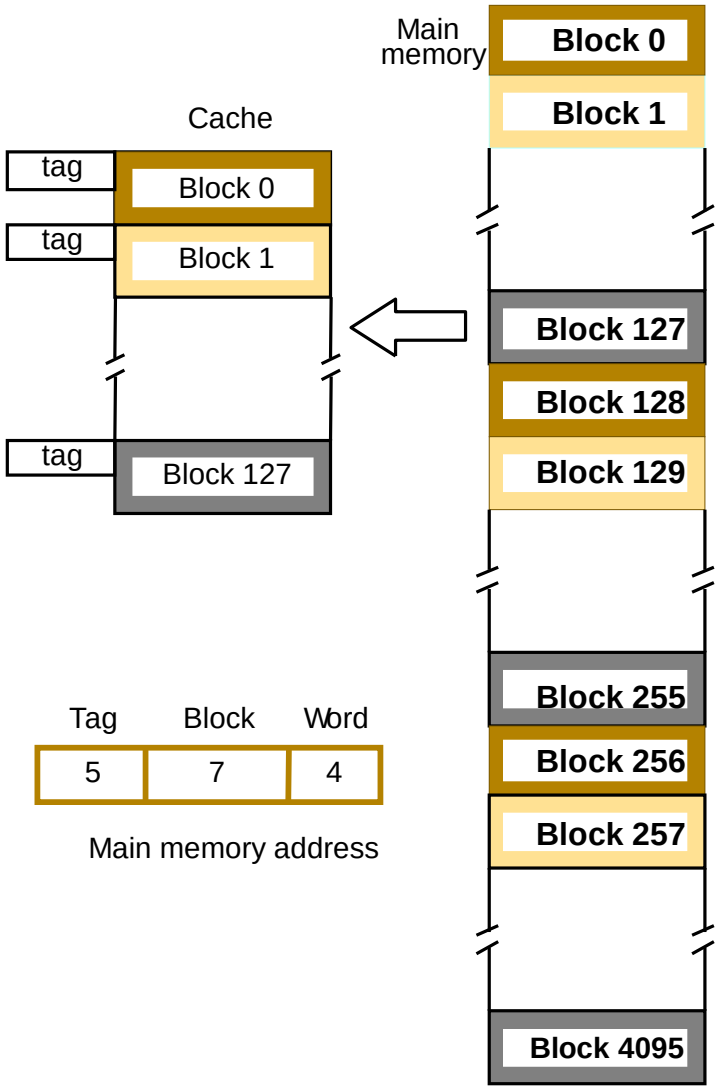
- *Data transfers between main memory and disk occur directly bypassing the cache.*
- *When the data on a disk changes, the main memory block is also updated.*
- *However, if the data is also resident in the cache, then the valid bit is set to 0.*

- *What happens if the data in the disk and main memory changes and the write-back protocol is being used?*
- *In this case, the data in the cache may also have changed and is indicated by the dirty bit.*
- *The copies of the data in the cache, and the main memory are different. This is called the cache coherence problem.*
- *One option is to force a write-back before the main memory is updated from the disk.*

Mapping functions

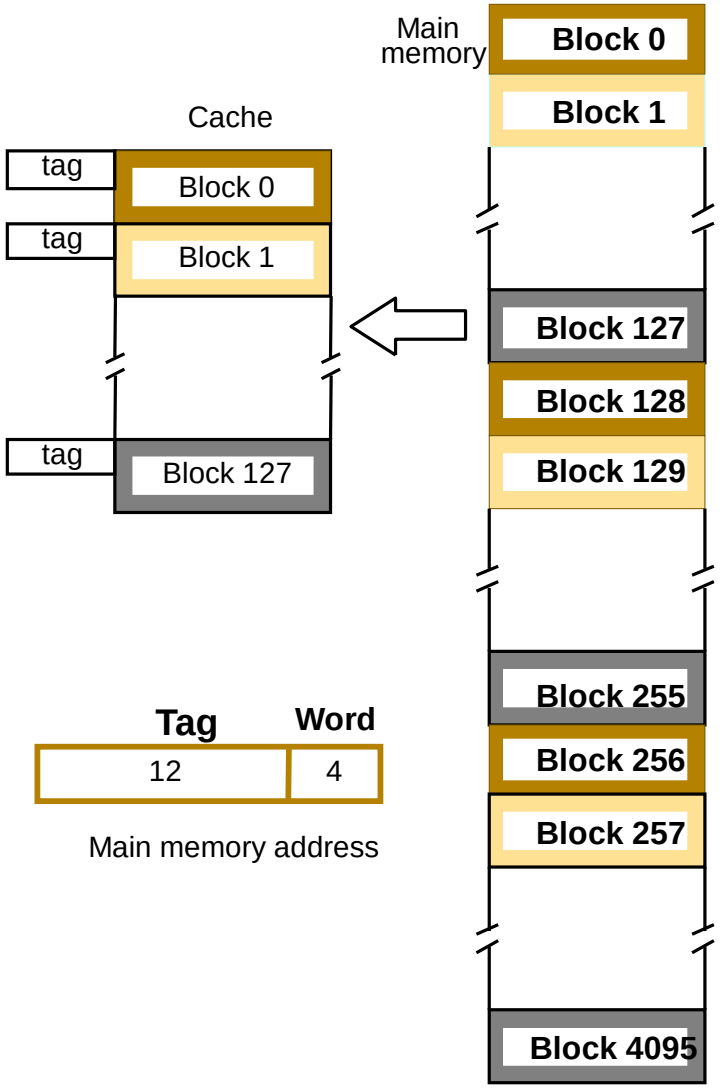
- Mapping functions determine how memory blocks are placed in the cache.
- A simple processor example:
  - Cache consisting of 128 blocks of 16 words each.
  - Total size of cache is 2048 (2K) words.
  - Main memory is addressable by a 16-bit address.
  - Main memory has 64K words.
  - Main memory has 4K blocks of 16 words each.
- Three mapping functions:
  - Direct mapping
  - Associative mapping
  - Set-associative mapping.

Direct mapping



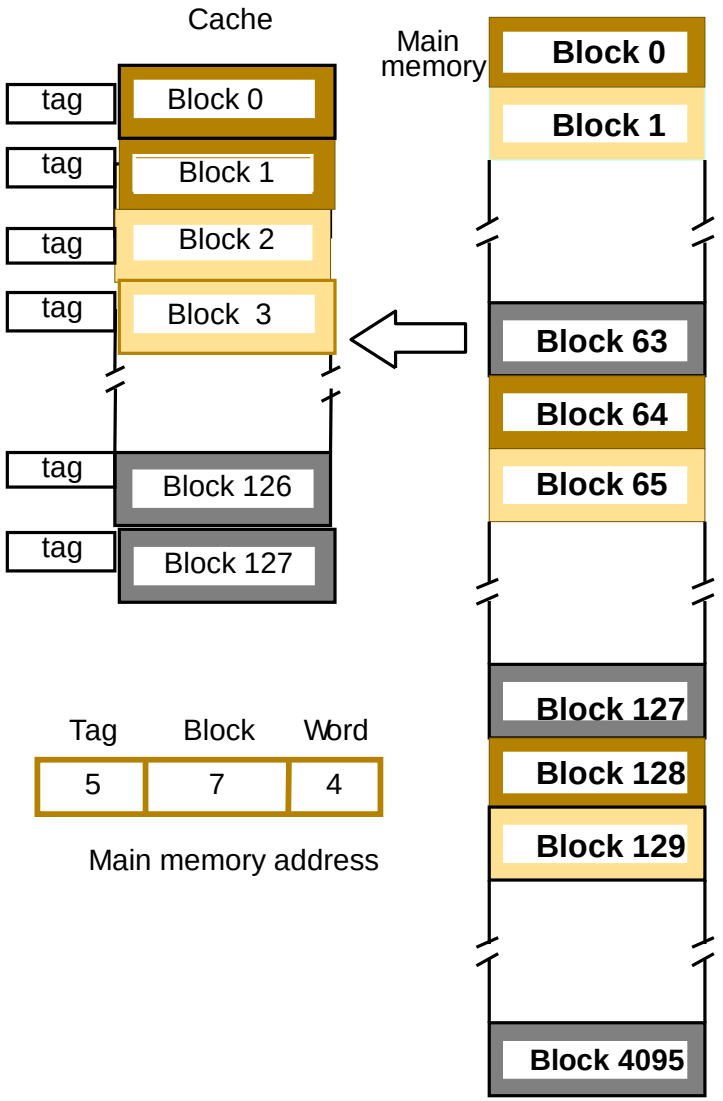
- *Block j of the main memory maps to j modulo 128 of the cache. 0 maps to 0, 129 maps to 1.*
- *More than one memory block is mapped onto the same position in the cache.*
- *May lead to contention for cache blocks even if the cache is not full.*
- *Resolve the contention by allowing new block to replace the old block, leading to a trivial replacement algorithm.*
- *Memory address is divided into three fields:*
  - *Low order 4 bits determine one of the 16 words in a block.*
  - *When a new block is brought into the cache, the the next 7 bits determine which cache block this new block is placed in.*
  - *High order 5 bits determine which of the possible 32 blocks is currently present in the cache. These are tag bits.*
- *Simple to implement but not very flexible.*

Associative mapping



- *Main memory block can be placed into any cache position.*
- *Memory address is divided into two fields:*
  - *Low order 4 bits identify the word within a block.*
  - *High order 12 bits or tag bits identify a memory block when it is resident in the cache.*
- *Flexible, and uses cache space efficiently.*
- *Replacement algorithms can be used to replace an existing block in the cache when the cache is full.*
- *Cost is higher than direct-mapped cache because of the need to search all 128 patterns to determine whether a given block is in the cache.*

Set-Associative mapping



Blocks of cache are grouped into sets.
Mapping function allows a block of the main memory to reside in any block of a specific set.
Divide the cache into 64 sets, with two blocks per set.
Memory block 0, 64, 128 etc. map to block 0, and they can occupy either of the two positions.
Memory address is divided into three fields:

- *6 bit field determines the set number.*
- *High order 6 bit fields are compared to the tag fields of the two blocks in a set.*

Set-associative mapping combination of direct and associative mapping.
Number of blocks per set is a design parameter.

- *One extreme is to have all the blocks in one set, requiring no set bits (fully associative mapping).*
- *Other extreme is to have one block per set, is the same as direct mapping.*

Performance considerations

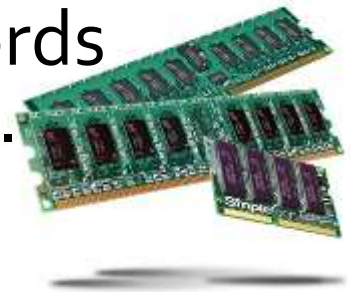
The Memory System

Performance considerations

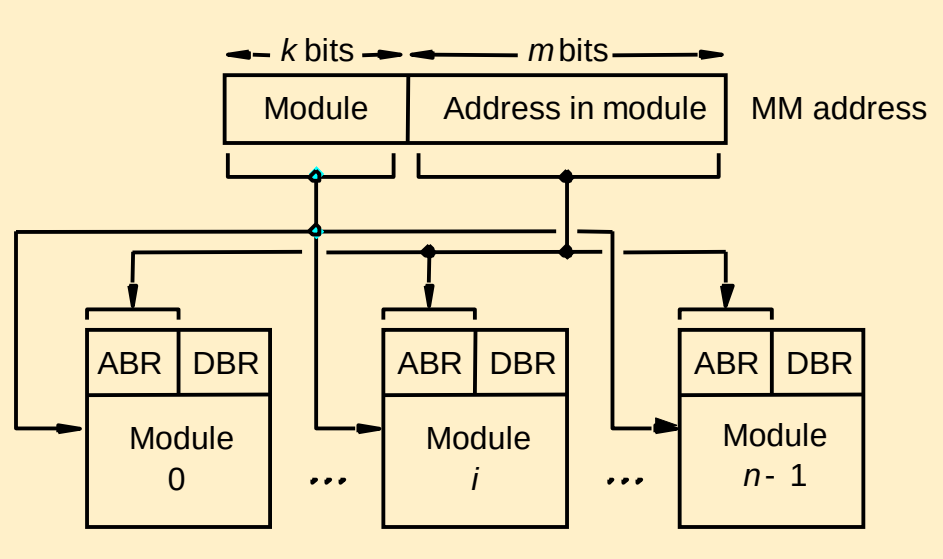
- A key design objective of a computer system is to achieve the best possible performance at the lowest possible cost.
  - Price/performance ratio is a common measure of success.
- Performance of a processor depends on:
  - How fast machine instructions can be brought into the processor for execution.
  - How fast the instructions can be executed.

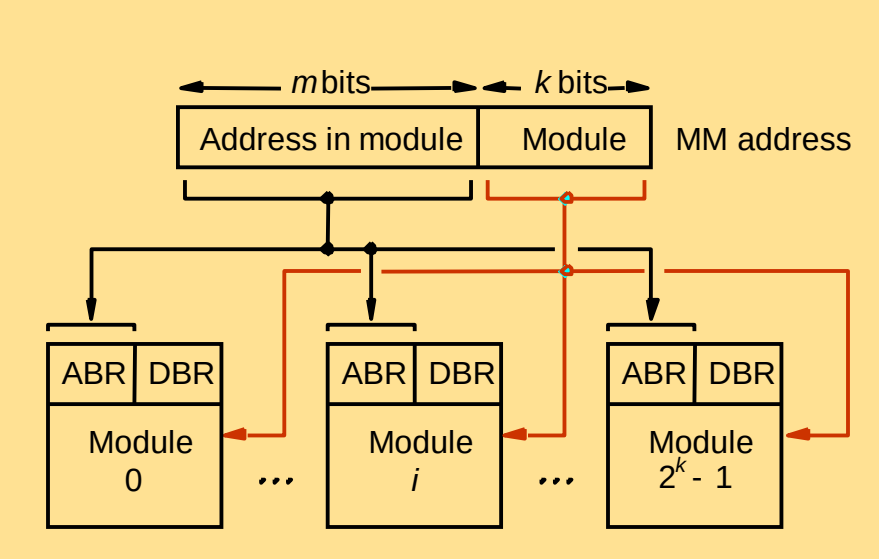
Interleaving

- Divides the memory system into a number of memory modules. Each module has its own address buffer register (ABR) and data buffer register (DBR).
- Arranges addressing so that successive words in the address space are placed in different modules.
- When requests for memory access involve consecutive addresses, the access will be to different modules.
- Since parallel access to these modules is possible, the average rate of fetching words from the Main Memory can be increased.



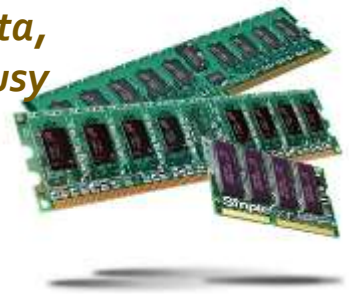
Methods of address layouts





- *Consecutive words are placed in a module.*
- *High-order k bits of a memory address determine the module.*
- *Low-order m bits of a memory address determine the word within a module.*
- *When a block of words is transferred from main memory to cache, only one module is busy at a time.*

- *Consecutive words are located in consecutive modules.*
- *Consecutive addresses can be located in consecutive modules.*
- *While transferring a block of data, several memory modules can be kept busy at the same time.*



Hit Rate and Miss Penalty

- Hit rate
- Miss penalty
- Hit rate can be improved by increasing block size, while keeping cache size constant
- Block sizes that are neither very small nor very large give best results.
- Miss penalty can be reduced if load-through approach is used when loading new blocks into cache.

Caches on the processor chip

- In high performance processors 2 levels of caches are normally used.
- Avg access time in a system with 2 levels of caches is

$$T_{ave} = h1c1+(1-h1)h2c2+(1-h1)(1-h2)M$$

Other Performance Enhancements

Write buffer

- *Write-through:*
- *Each write operation involves writing to the main memory.*
- *If the processor has to wait for the write operation to be complete, it slows down the processor.*
- *Processor does not depend on the results of the write operation.*
- *Write buffer can be included for temporary storage of write requests.*
- *Processor places each write request into the buffer and continues execution.*
- *If a subsequent Read request references data which is still in the write buffer, then this data is referenced in the write buffer.*

- *Write-back:*
- *Block is written back to the main memory when it is replaced.*
- *If the processor waits for this write to complete, before reading the new block, it is slowed down.*
- *Fast write buffer can hold the block to be written, and the new block can be read first.*

Other Performance Enhancements (Contd.,)

Prefetching

- *New data are brought into the processor when they are first needed.*
- *Processor has to wait before the data transfer is complete.*
- *Prefetch the data into the cache before they are actually needed, or a before a Read miss occurs.*
- *Prefetching can be accomplished through software by including a special instruction in the machine language of the processor.*
  - ***Inclusion of prefetch instructions increases the length of the programs.***
- *Prefetching can also be accomplished using hardware:*
  - ***Circuitry that attempts to discover patterns in memory references and then prefetches according to this pattern.***

Other Performance Enhancements (Contd.,)

Lockup-Free Cache

- *Prefetching scheme does not work if it stops other accesses to the cache until the prefetch is completed.*
- *A cache of this type is said to be "locked" while it services a miss.*
- *Cache structure which supports multiple outstanding misses is called a lockup free cache.*
- *Since only one miss can be serviced at a time, a lockup free cache must include circuits that keep track of all the outstanding misses.*
- *Special registers may hold the necessary information about these misses.*

Virtual Memory

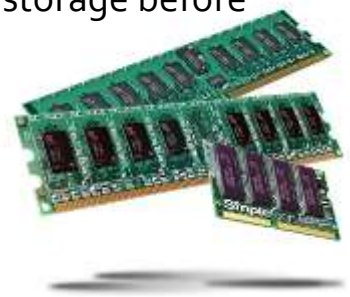
The Memory System

Virtual memories

- Recall that an important challenge in the design of a computer system is to provide a large, fast memory system at an affordable cost.
- Architectural solutions to increase the effective speed and size of the memory system.
- Cache memories were developed to increase the effective speed of the memory system.
- Virtual memory is an architectural solution to increase the effective size of the memory system.

Virtual memories (contd..)

- Recall that the addressable memory space depends on the number of address bits in a computer.
  - For example, if a computer issues 32-bit addresses, the addressable memory space is 4G bytes.
- Physical main memory in a computer is generally not as large as the entire possible addressable space.
  - Physical memory typically ranges from a few hundred megabytes to 1G bytes.
- Large programs that cannot fit completely into the main memory have their parts stored on secondary storage devices such as magnetic disks.
  - Pieces of programs must be transferred to the main memory from secondary storage before they can be executed.



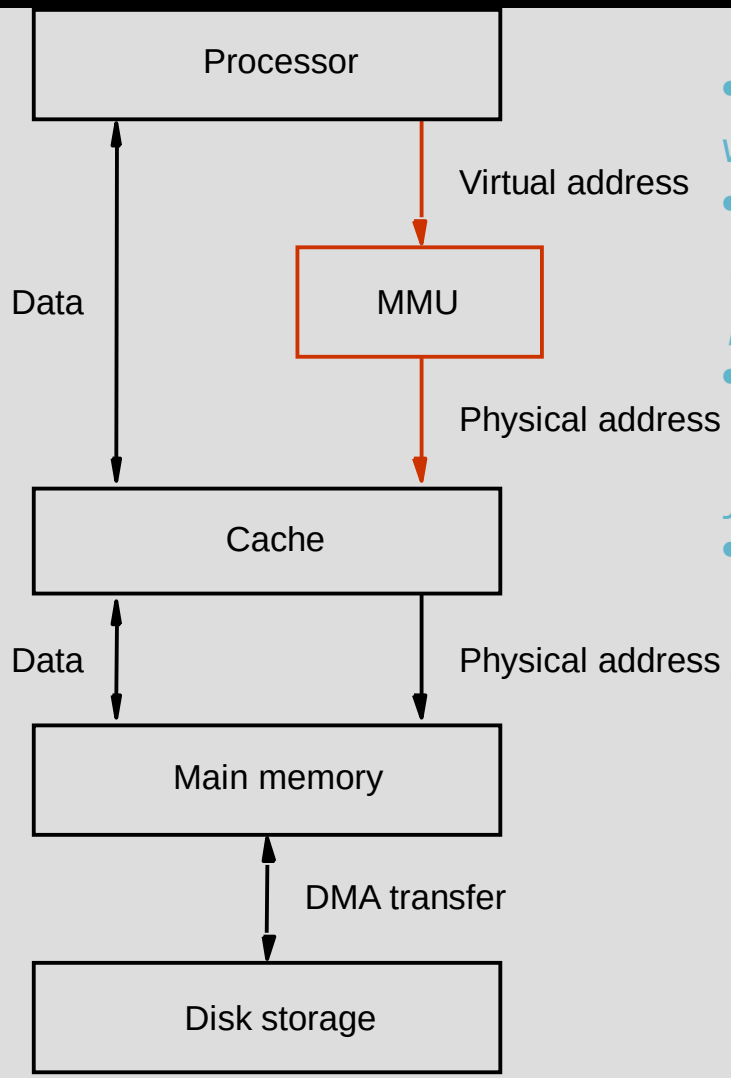
Virtual memories (contd..)

- When a new piece of a program is to be transferred to the main memory, and the main memory is full, then some other piece in the main memory must be replaced.
  - Recall this is very similar to what we studied in case of cache memories.
- Operating system automatically transfers data between the main memory and secondary storage.
  - Application programmer need not be concerned with this transfer.
  - Also, application programmer does not need to be aware of the limitations imposed by the available physical memory.

Virtual memories (contd..)

- Techniques that automatically move program and data between main memory and secondary storage when they are required for execution are called virtual-memory techniques.
- Programs and processors reference an instruction or data independent of the size of the main memory.
- Processor issues binary addresses for instructions and data.
  - These binary addresses are called logical or virtual addresses.
- Virtual addresses are translated into physical addresses by a combination of hardware and software subsystems.
  - If virtual address refers to a part of the program that is currently in the main memory, it is accessed immediately.
  - If the address refers to a part of the program that is not currently in the main memory, it is first transferred to the main memory before it can be used.

Virtual memory organization



- *Memory management unit (MMU) translates virtual addresses into physical addresses.*
- *If the desired data or instructions are in the main memory they are fetched as described previously.*
- *If the desired data or instructions are not in the main memory, they must be transferred from secondary storage to the main memory.*
- *MMU causes the operating system to bring the data from the secondary storage into the main memory.*

Address translation

- Assume that program and data are composed of fixed-length units called pages.
- A page consists of a block of words that occupy contiguous locations in the main memory.
- Page is a basic unit of information that is transferred between secondary storage and main memory.
- Size of a page commonly ranges from 2K to 16K bytes.
  - Pages should not be too small, because the access time of a secondary storage device is much larger than the main memory.
  - Pages should not be too large, else a large portion of the page may not be used, and it will occupy valuable space in the main memory.

Address translation (contd..)

- Concepts of virtual memory are similar to the concepts of cache memory.
- Cache memory:
  - Introduced to bridge the speed gap between the processor and the main memory.
  - Implemented in hardware.
- Virtual memory:
  - Introduced to bridge the speed gap between the main memory and secondary storage.
  - Implemented in part by software.

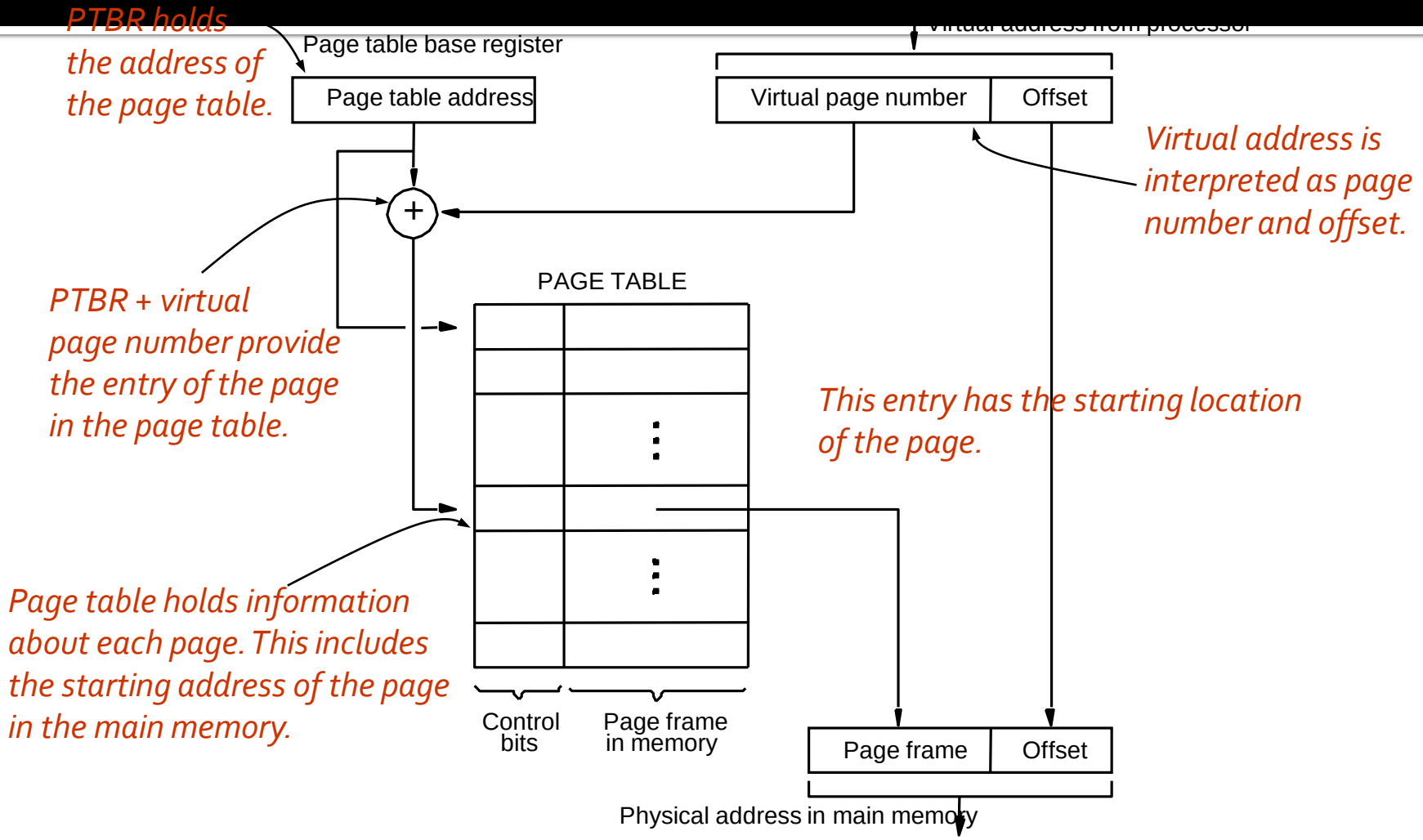
Address translation (contd..)

- Each virtual or logical address generated by a processor is interpreted as a virtual page number (high-order bits) plus an offset (low-order bits) that specifies the location of a particular byte within that page.
- Information about the main memory location of each page is kept in the page table.
  - Main memory address where the page is stored.
  - Current status of the page.
- Area of the main memory that can hold a page is called as page frame.
- Starting address of the page table is kept in a page table base register.

Address translation (contd..)

- Virtual page number generated by the processor is added to the contents of the page table base register.
  - This provides the address of the corresponding entry in the page table.
- The contents of this location in the page table give the starting address of the page if the page is currently in the main memory.

Address translation (contd..)



Address translation (contd..)

- Page table entry for a page also includes some control bits which describe the status of the page while it is in the main memory.
- One bit indicates the validity of the page.
  - Indicates whether the page is actually loaded into the main memory.
  - Allows the operating system to invalidate the page without actually removing it.
- One bit indicates whether the page has been modified during its residency in the main memory.
  - This bit determines whether the page should be written back to the disk when it is removed from the main memory.
  - Similar to the dirty or modified bit in case of cache memory.

Address translation (contd..)

- Other control bits for various other types of restrictions that may be imposed.
  - For example, a program may only have read permission for a page, but not write or modify permissions.

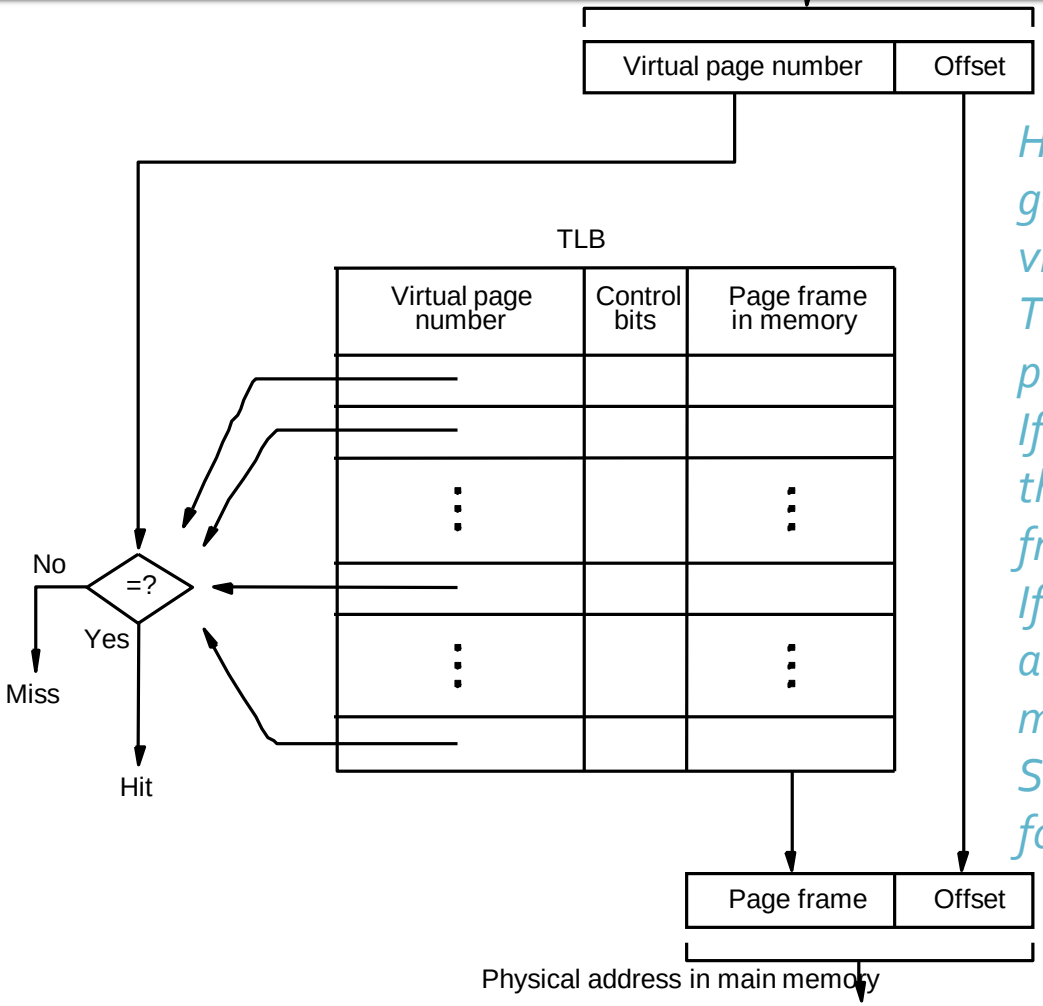
Address translation (contd..)

- Where should the page table be located?
- Recall that the page table is used by the MMU for every read and write access to the memory.
  - Ideal location for the page table is within the MMU.
- Page table is quite large.
- MMU is implemented as part of the processor chip.
- Impossible to include a complete page table on the chip.
- Page table is kept in the main memory.
- A copy of a small portion of the page table can be accommodated within the MMU.
  - Portion consists of page table entries that correspond to the most recently accessed pages.

Address translation (contd..)

- A small cache called as Translation Lookaside Buffer (TLB) is included in the MMU.
  - TLB holds page table entries of the most recently accessed pages.
- Recall that cache memory holds most recently accessed blocks from the main memory.
  - Operation of the TLB and page table in the main memory is similar to the operation of the cache and main memory.
- Page table entry for a page includes:
  - Address of the page frame where the page resides in the main memory.
  - Some control bits.
- In addition to the above for each page, TLB must hold the virtual page number for each page.

Address translation (contd..)



Associative-mapped TLB

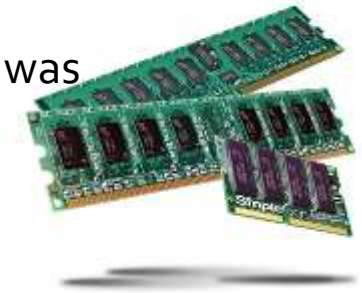
High-order bits of the virtual address generated by the processor select the virtual page.
These bits are compared to the virtual page numbers in the TLB.
If there is a match, a hit occurs and the corresponding address of the page frame is read.
If there is no match, a miss occurs and the page table within the main memory must be consulted.
Set-associative mapped TLBs are found in commercial processors.

Address translation (contd..)

- How to keep the entries of the TLB coherent with the contents of the page table in the main memory?
- Operating system may change the contents of the page table in the main memory.
  - Simultaneously it must also invalidate the corresponding entries in the TLB.
- A control bit is provided in the TLB to invalidate an entry.
- If an entry is invalidated, then the TLB gets the information for that entry from the page table.
  - Follows the same process that it would follow if the entry is not found in the TLB or if a "miss" occurs.

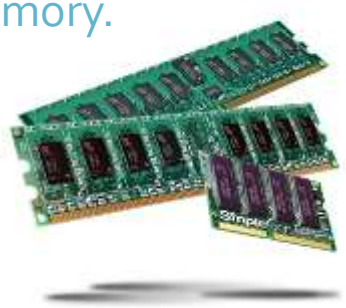
Address translation (contd..)

- What happens if a program generates an access to a page that is not in the main memory?
- In this case, a page fault is said to occur.
  - Whole page must be brought into the main memory from the disk, before the execution can proceed.
- Upon detecting a page fault by the MMU, following actions occur:
  - MMU asks the operating system to intervene by raising an exception.
  - Processing of the active task which caused the page fault is interrupted.
  - Control is transferred to the operating system.
  - Operating system copies the requested page from secondary storage to the main memory.
  - Once the page is copied, control is returned to the task which was interrupted.



Address translation (contd..)

- Servicing of a page fault requires transferring the requested page from secondary storage to the main memory.
- This transfer may incur a long delay.
- While the page is being transferred the operating system may:
  - Suspend the execution of the task that caused the page fault.
  - Begin execution of another task whose pages are in the main memory.
- Enables efficient use of the processor.

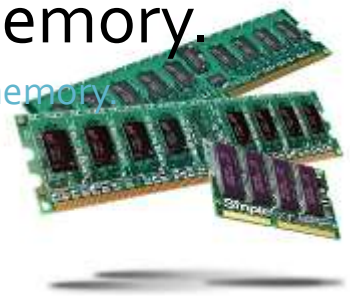


Address translation (contd..)

- How to ensure that the interrupted task can continue correctly when it resumes execution?
- There are two possibilities:
  - Execution of the interrupted task must continue from the point where it was interrupted.
  - The instruction must be restarted.
- Which specific option is followed depends on the design of the processor.

Address translation (contd..)

- When a new page is to be brought into the main memory from secondary storage, the main memory may be full.
  - Some page from the main memory must be replaced with this new page.
- How to choose which page to replace?
  - This is similar to the replacement that occurs when the cache is full.
  - The principle of locality of reference (?) can also be applied here.
  - A replacement strategy similar to LRU can be applied.
- Since the size of the main memory is relatively larger compared to cache, a relatively large amount of programs and data can be held in the main memory.
  - Minimizes the frequency of transfers between secondary storage and main memory.



Address translation (contd..)

- A page may be modified during its residency in the main memory.
- When should the page be written back to the secondary storage?
- Recall that we encountered a similar problem in the context of cache and main memory:
  - Write-through protocol(?)
  - Write-back protocol(?)
- Write-through protocol cannot be used, since it will incur a long delay each time a small amount of data is written to the disk.

Memory Management

The Memory System

Memory management

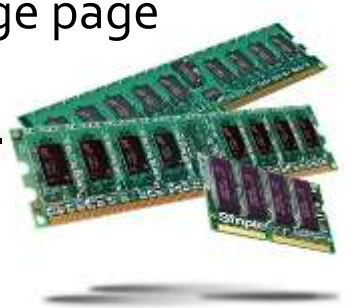
- Operating system is concerned with transferring programs and data between secondary storage and main memory.
- Operating system needs memory routines in addition to the other routines.
- Operating system routines are assembled into a virtual address space called system space.
- System space is separate from the space in which user application programs reside.
  - This is user space.
- Virtual address space is divided into one system space + several user spaces.

Memory management (contd..)

- Recall that the Memory Management Unit (MMU) translates logical or virtual addresses into physical addresses.
- MMU uses the contents of the page table base register to determine the address of the page table to be used in the translation.
  - Changing the contents of the page table base register can enable us to use a different page table, and switch from one space to another.
- At any given time, the page table base register can point to one page table.
  - Thus, only one page table can be used in the translation process at a given time.
  - Pages belonging to only one space are accessible at any given time.

Memory management (contd..)

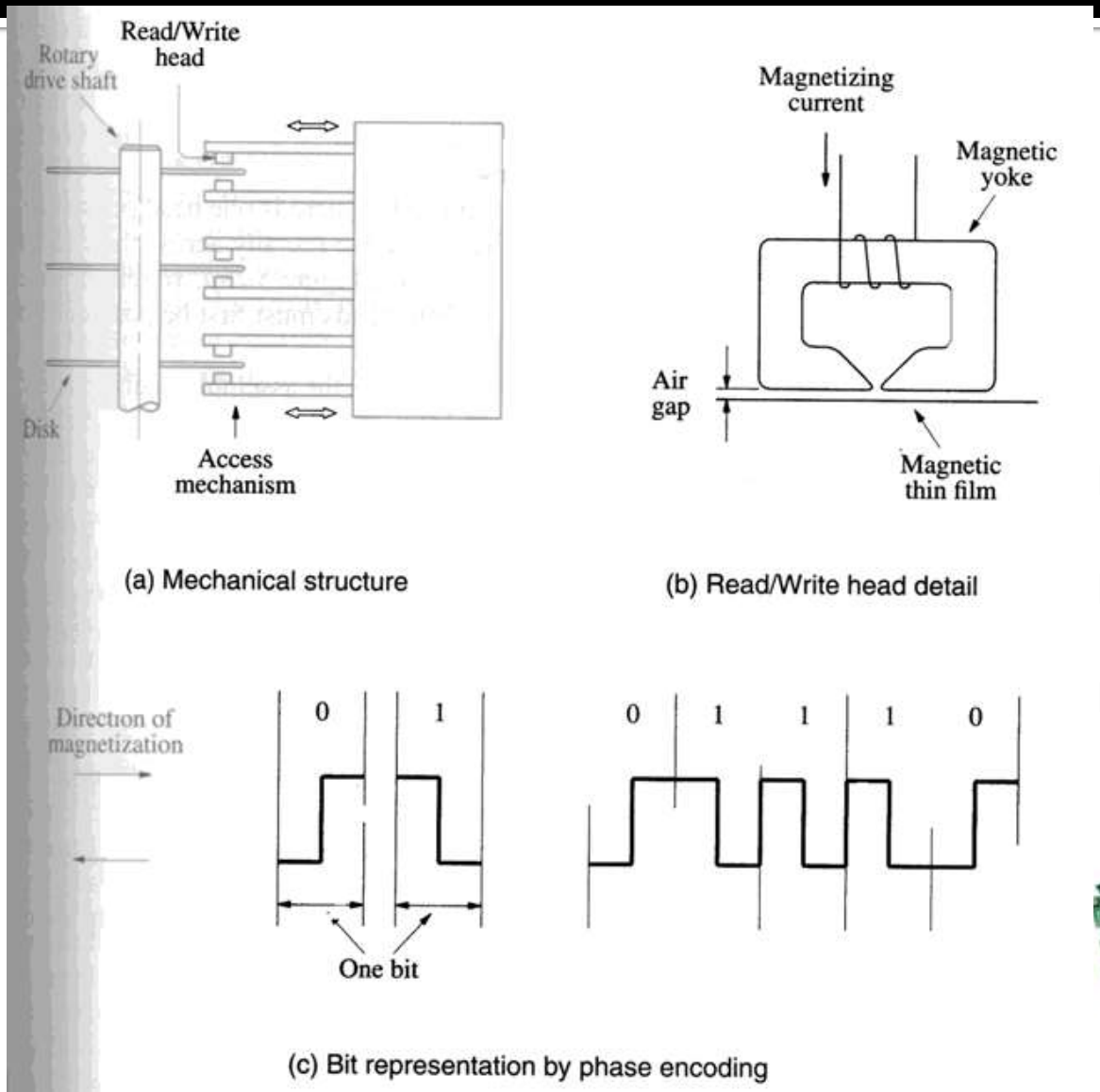
- When multiple, independent user programs coexist in the main memory, how to ensure that one program does not modify/destroy the contents of the other?
- Processor usually has two states of operation:
  - Supervisor state.
  - User state.
- Supervisor state:
  - Operating system routines are executed.
- User state:
  - User programs are executed.
  - Certain privileged instructions cannot be executed in user state.
  - These privileged instructions include the ones which change page table base register.
  - Prevents one user from accessing the space of other users.



Secondary Storage

The Memory System

Magnetic Hard Disks



Disk

Disk drive

Disk controller

Organization of Data on a Disk

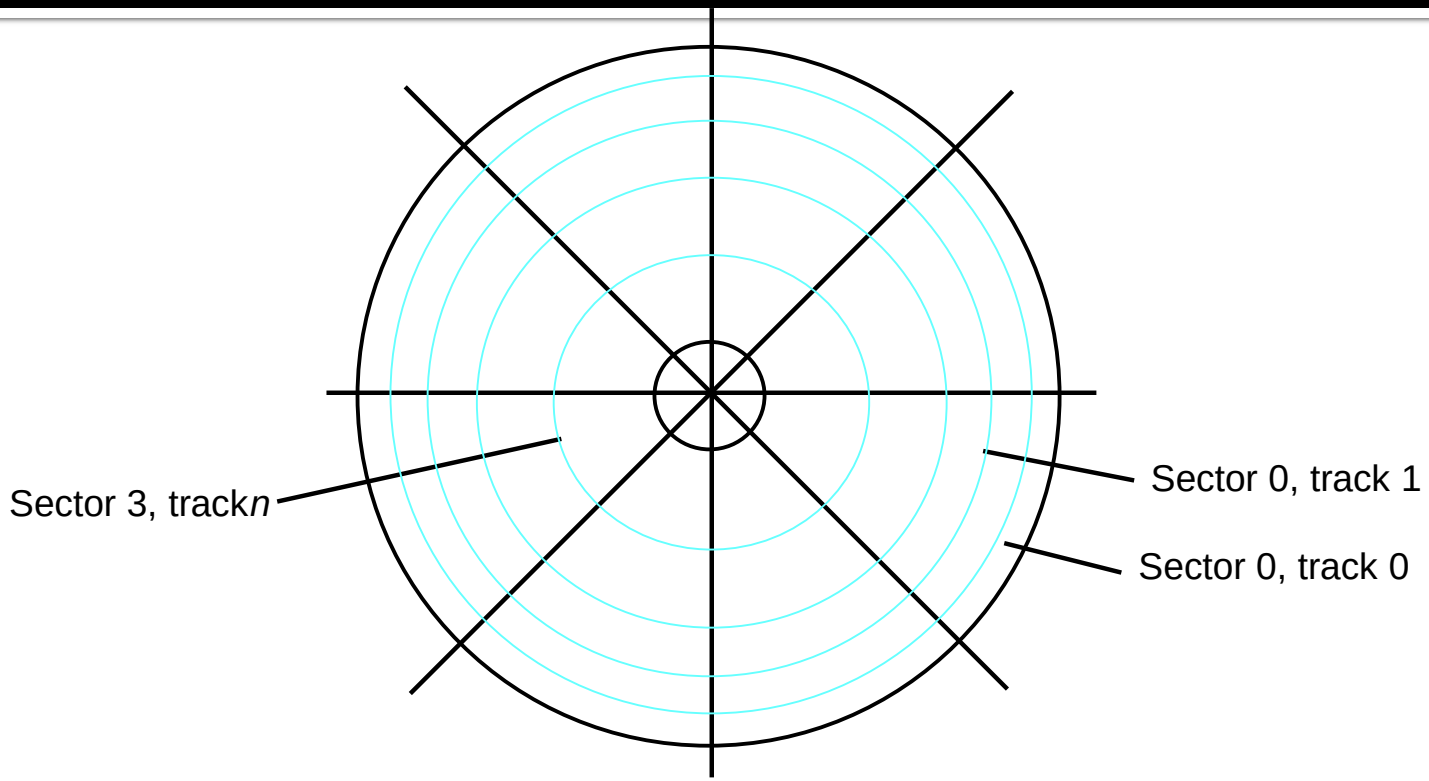


Figure 5.30. Organization of one surface of a disk.

Access Data on a Disk

- Sector header
- Following the data, there is an error-correction code (ECC).
- Formatting process
- Difference between inner tracks and outer tracks
- Access time – seek time / rotational delay (latency time)
- Data buffer/cache

Disk Controller

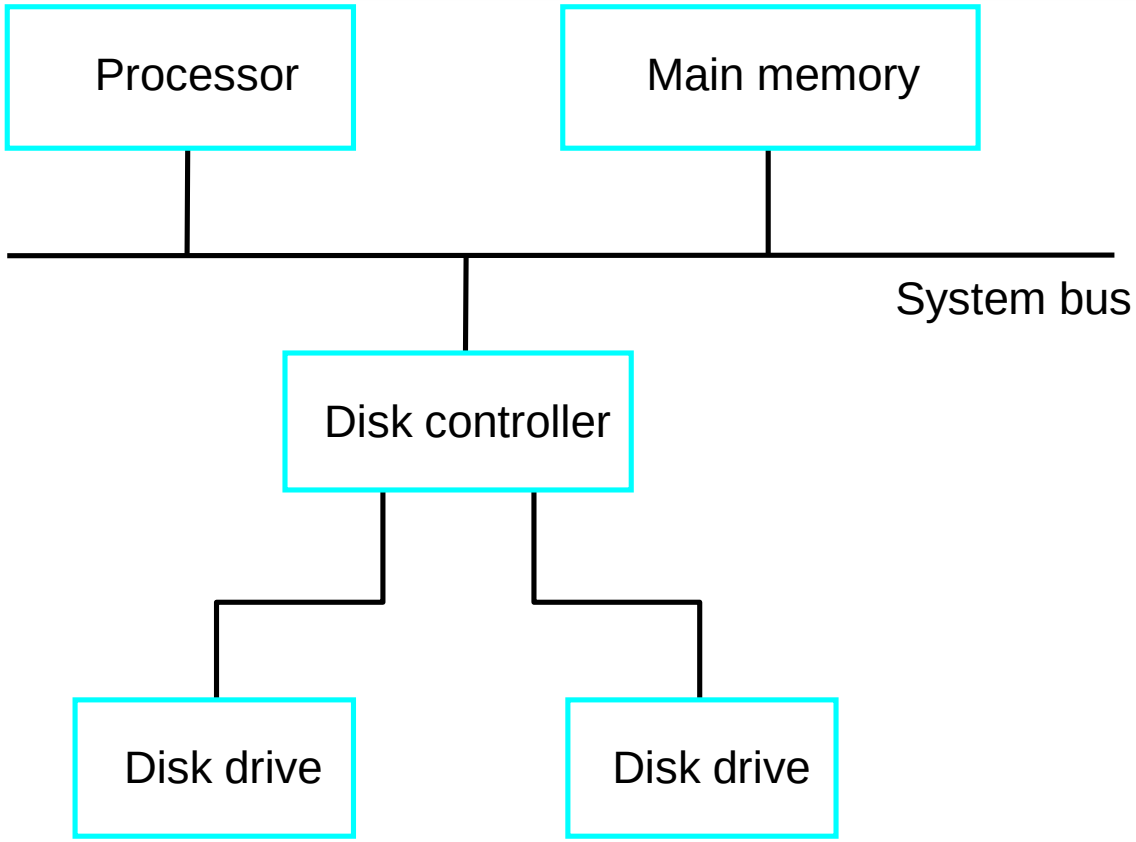


Figure 5.31. Disks connected to the system bus.

Disk Controller

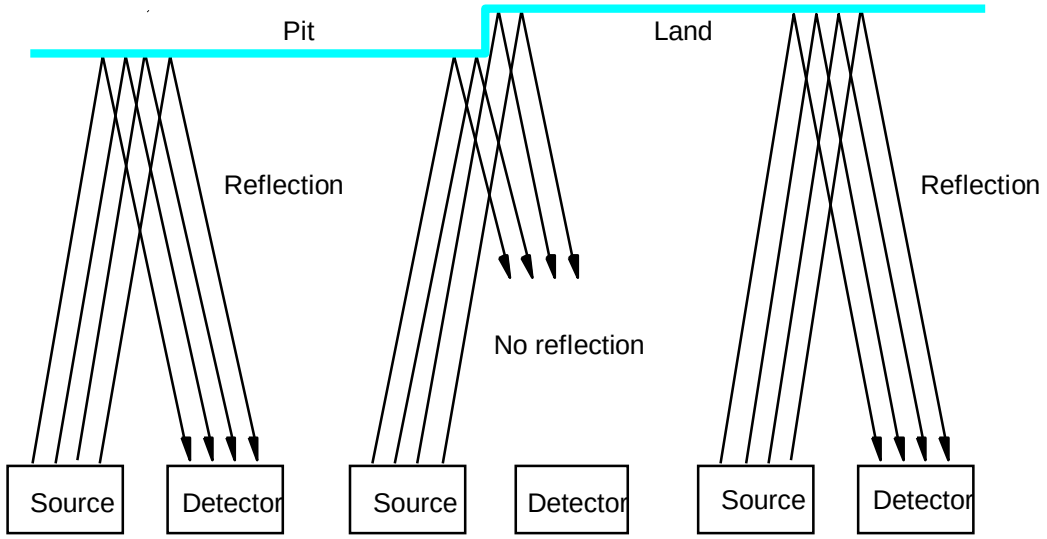
- Seek
- Read
- Write
- Error checking

RAID Disk Arrays

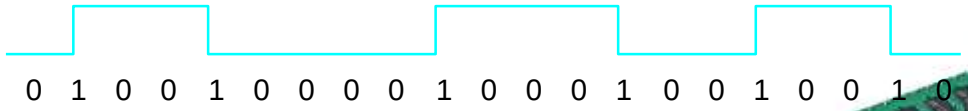
- Redundant Array of Inexpensive Disks
- Using multiple disks makes it cheaper for huge storage, and also possible to improve the reliability of the overall system.
- RAID0 – data striping
- RAID1 – identical copies of data on two disks
- RAID2, 3, 4 – increased reliability
- RAID5 – parity-based error-recovery

Optical Disks

(a) Cross-section



(b) Transition from pit to land



(c) Stored binary pattern

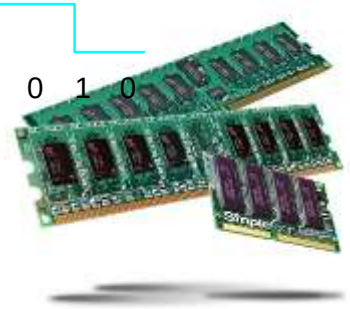


Figure 5.32. Optical disk.

Optical Disks

- CD-ROM
- CD-Recordable (CD-R)
- CD-ReWritable (CD-RW)
- DVD
- DVD-RAM

Magnetic Tape Systems

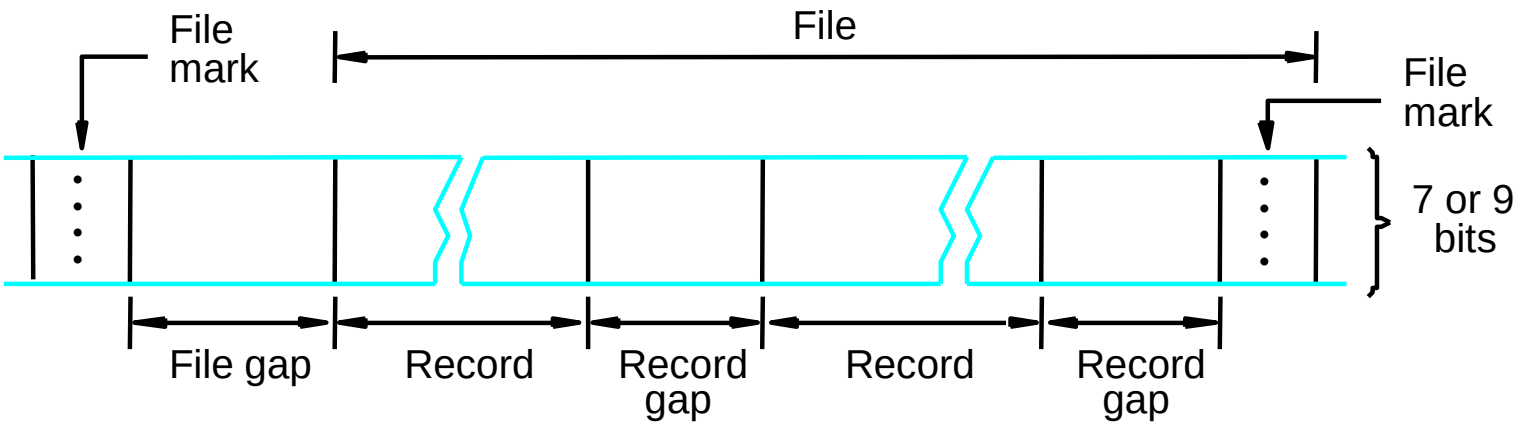


Figure 5.33. Organization of data on magnetic tape.

MODULE_4 : MEMORY SYSTEM

5.1 BASIC CONCEPTS:

The maximum size of the Main Memory (MM) that can be used in any computer is determined by its addressing scheme. For example, a 16-bit computer that generates 16-bit addresses is capable of addressing upto 2_{16} =64K memory locations. If a machine generates 32-bit addresses, it can access upto 2_{32} = 4G memory locations. This number represents the size of address space of the computer.

If the smallest addressable unit of information is a memory word, the machine is called word-addressable. If individual memory bytes are assigned distinct addresses, the computer is called byte-addressable. Most of the commercial machines are byte-addressable. For example in a byte-addressable 32-bit computer, each memory word contains 4 bytes. A possible word-address assignment would be:

Word Address	Byte Address			
0	0	1	2	3
4	4	5	6	7
8	8	9	10	11
.		…..		
.		…..		
.		…..		

With the above structure a READ or WRITE may involve an entire memory word or it may involve only a byte. In the case of byte read, other bytes can also be read but ignored by the CPU. However, during a write cycle, the control circuitry of the MM must ensure that only the specified byte is altered. In this case, the higher-order 30 bits can specify the word and the lower-order 2 bits can specify the byte within the word.

CPU-Main Memory Connection – A block schematic: -

From the system standpoint, the Main Memory (MM) unit can be viewed as a ―block box‖. Data transfer between CPU and MM takes place through the use of two CPU registers, usually called MAR (Memory Address Register) and MDR (Memory Data Register). If MAR is K bits long and MDR is _n' bits long, then the MM unit may contain upto 2^k addressable locations and each location will be _n' bits wide, while the word length is equal to _n' bits. During a ―memory cycle‖, n bits of data may be transferred between the MM and CPU. This transfer takes place over the processor bus, which has k address lines (address bus), n data lines (data bus) and control lines like Read, Write, Memory Function completed (MFC), Bytes specifiers etc (control bus). For a read operation, the CPU loads the address into MAR, set READ to 1 and sets other control signals if required. The data from the MM is loaded into MDR and MFC is set to 1. For a write operation, MAR, MDR are suitably loaded by the CPU, write is set to 1 and other control signals are set suitably. The MM control circuitry loads the data into appropriate locations and sets MFC to 1. This organization is shown in the following block schematic

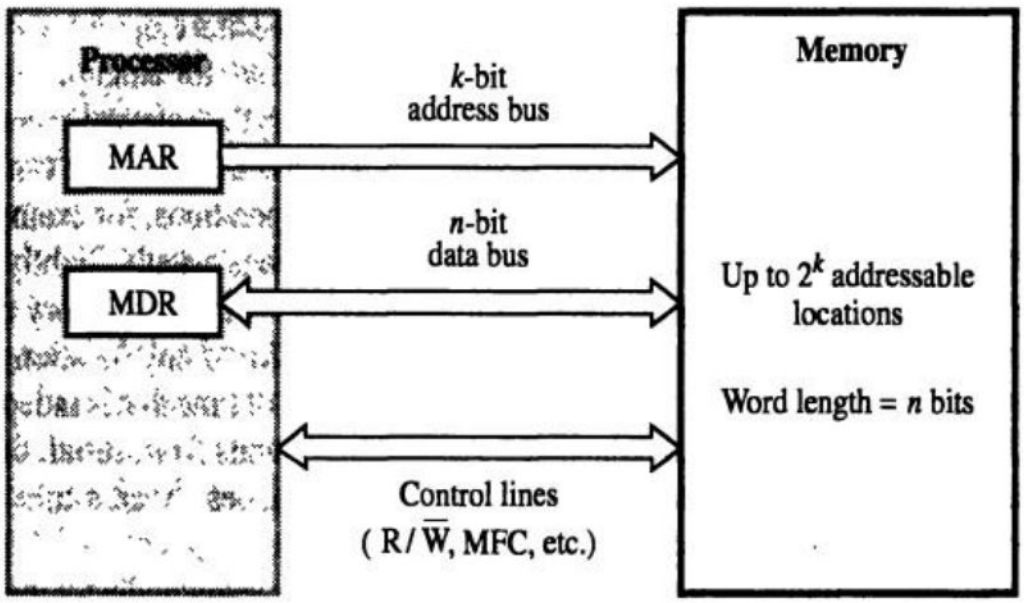


Figure 5.1 Connection of the memory to the processor.

Some Basic Concepts

Memory Access Times: -

It is a useful measure of the speed of the memory unit. It is the time that elapses between the initiation of an operation and the completion of that operation (for example, the time between READ and MFC).

Memory Cycle Time :-

It is an important measure of the memory system. It is the minimum time delay required between the initiations of two successive memory operations (for example, the time between two successive READ operations). The cycle time is usually slightly longer than the access time.

5.2 RANDOM ACCESS MEMORY (RAM):

A memory unit is called a Random Access Memory if any location can be accessed for a READ or WRITE operation in some fixed amount of time that is independent of the location‘s address. Main memory units are of this type. This distinguishes them from serial or partly serial access storage devices such as magnetic tapes and disks which are used as the secondary storage device.

Cache Memory:-

The CPU of a computer can usually process instructions and data faster than they can be fetched from compatibly priced main memory unit. Thus the memory cycle time become the bottleneck in the system. One way to reduce the memory access time is to use cache memory. This is a small and fast memory that is inserted between the larger, slower main memory and the CPU. This holds the currently active segments of a program and its data. Because of the locality of address references, the CPU can, most of the time, find the relevant information in the cache memory itself (cache hit) and infrequently needs access to the main memory (cache miss) with suitable size of the cache memory, cache hit rates of over 90% are possible leading to a cost-effective increase in the performance of the system.

Memory Interleaving: -

This technique divides the memory system into a number of memory modules and arranges addressing so that successive words in the address space are placed in different modules. When requests for memory access involve consecutive addresses, the access will be to different modules. Since parallel access to these modules is possible, the average rate of fetching words from the Main Memory can be increased.

Virtual Memory: -

In a virtual memory System, the address generated by the CPU is referred to as a virtual or logical address. The corresponding physical address can be different and the required mapping is implemented by a special memory control unit, often called the memory management unit. The mapping function itself may be changed during program execution according to system requirements.

Because of the distinction made between the logical (virtual) address space and the physical address space; while the former can be as large as the addressing capability of the CPU, the actual physical memory can be much smaller. Only the active portion of the virtual address space is mapped onto the physical memory and the rest of the virtual address space is mapped onto the bulk storage device used. If the addressed information is in the Main Memory (MM), it is accessed and execution proceeds. Otherwise, an exception is generated, in response to which the memory management unit transfers a contigious block of words containing the desired word from the bulk storage unit to the MM, displacing some block that is currently inactive. If the memory is managed in such a way that, such transfers are required relatively infrequency (ie the CPU will generally find the required information in the MM), the virtual memory system can provide a reasonably good performance and succeed in creating an illusion of a large memory with a small, in expensive MM.

5.3 INTERNAL ORGANIZATION OF SEMICONDUCTOR MEMORY CHIPS:

Memory chips are usually organized in the form of an array of cells, in which each cell is capable of storing one bit of information. A row of cells constitutes a memory word, and the cells of a row are connected to a common line referred to as the word line, and this line is driven by the address decoder on the chip. The cells in each column are connected to a sense/write circuit by two lines known as bit lines. The sense/write circuits are connected to the data input/output lines of the chip. During a READ operation, the Sense/Write circuits sense, or read, the information stored in the cells selected by a word line and transmit this information to the output lines. During a write operation, they receive input information and store it in the cells of the selected word.

The following figure shows such an organization of a memory chip consisting of 16 words of 8 bits each, which is usually referred to as a 16 x 8 organization.

The data input and the data output of each Sense/Write circuit are connected to a single bi-directional data line in order to reduce the number of pins required. One control line, the R/W (Read/Write) input is used a specify the required operation and another control line, the CS (Chip Select) input is used to select a given chip in a multichip memory system. This circuit requires 14 external connections, and allowing 2 pins for power supply and ground connections, can be manufactured in the form of a 16-pin chip. It can store 16 x 8 = 128 bits.

Another type of organization for 1k x 1 format is shown below:

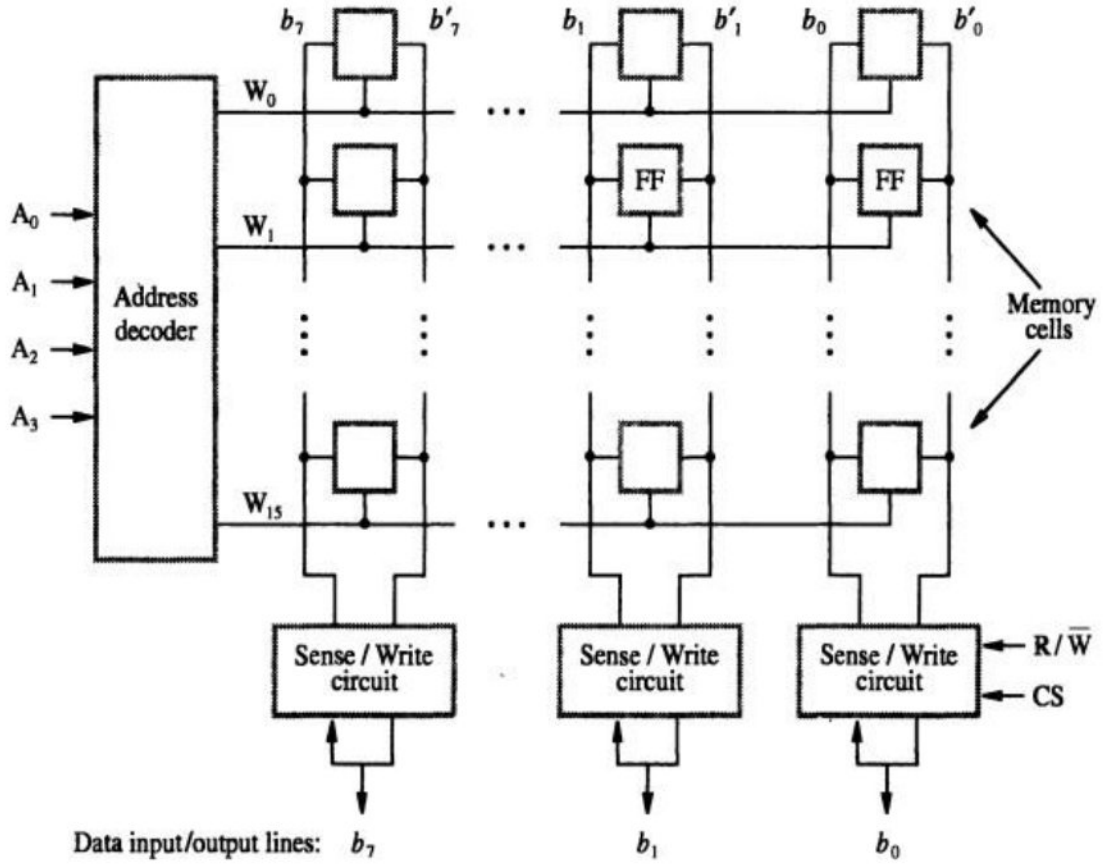


Figure 5.2 Organization of bit cells in a memory chip.

The 10-bit address is divided into two groups of 5 bits each to form the row and column addresses for the cell array. A row address selects a row of 32 cells, all of which are accessed in parallel. One of these, selected by the column address, is connected to the external data lines by the input and output multiplexers. This structure can store 1024

bits, can be implemented in a 16-
pin chip.

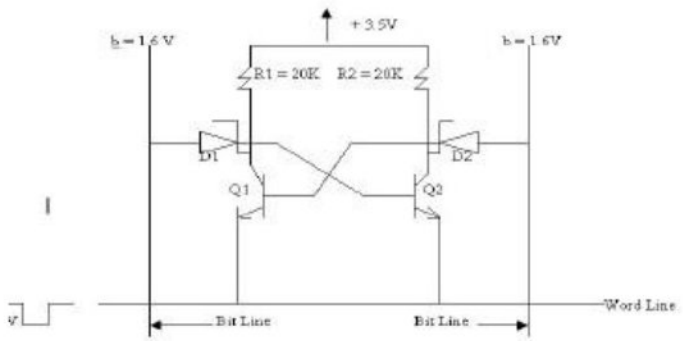
A Typical Memory Cell

Semiconductor memories may be divided into bipolar and MOS types. They may be compared as follows:

Characteristic	Bipolar	MOS
Power Dissipation	More	Less
Bit Density	Less	More
Impedance	Lower	Higher

Bipolar Memory Cell

A typical bipolar storage cell is shown below:



Two transistor inverters connected to implement a basic flip-flop. The cell is connected to one word line and two bits lines as shown. Normally, the bit lines are kept at about 1.6V, and the word line is kept at a slightly higher voltage of about 2.5V. Under these conditions, the two diodes D1 and D2 are reverse biased. Thus, because no current flows through the diodes, the cell is isolated from the bit lines.

Read Operation:-

Let us assume the Q1 on and Q2 off represents a 1 to read the contents of a given cell, the voltage on the corresponding word line is reduced from 2.5 V to approximately 0.3 V. This causes one of the diodes D1 or D2 to become forward-biased, depending on whether the transistor Q1 or Q2 is conducting. As a result, current flows from bit line b when the cell is in the 1 state and from bit line b when the cell is in the 0 state. The Sense/Write circuit at the end of each pair of bit lines monitors the current on lines b and b‘ and sets the output bit line accordingly.

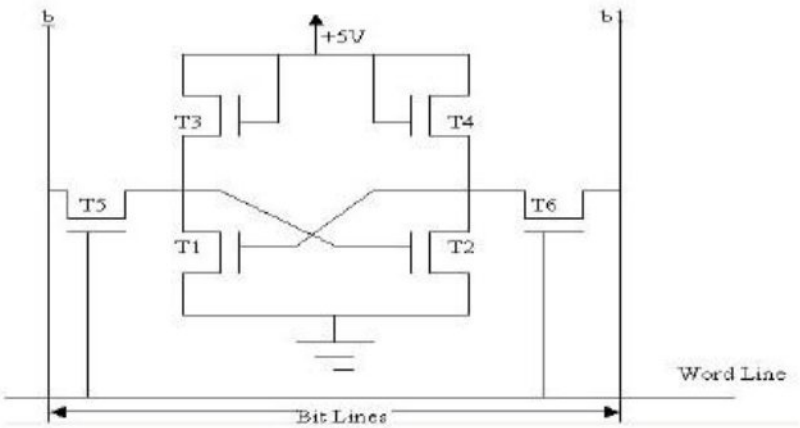
Write Operation: -

While a given row of bits is selected, that is, while the voltage on the corresponding word line is 0.3V, the cells can be individually forced to either the 1 state

by applying a positive voltage of about 3V to line b‘ or to the 0 state by driving line b. This function is performed by the Sense/Write circuit.

MOS Memory Cell: -

MOS technology is used extensively in Main Memory Units. As in the case of bipolar memories, many MOS cell configurations are possible. The simplest of these is a flip-flop circuit. Two transistors T1 and T2 are connected to implement a flip-flop. Active pull-up to VCC is provided through T3 and T4. Transistors T5 and T6 act as switches that can be opened or closed under control of the word line. For a read operation, when the cell is selected, T5 or T6 is closed and the corresponding flow of current through b or b‘ is sensed by the sense/write circuits to set the output bit line accordingly. For a write operation, the bit is selected and a positive voltage is applied on the appropriate bit line, to store a 0 or 1. This configuration is shown below:



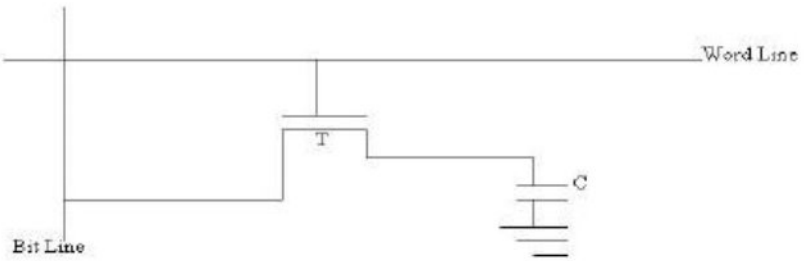
Static Memories Vs Dynamic Memories:-

Bipolar as well as MOS memory cells using a flip-flop like structure to store information can maintain the information as long as current flow to the cell is maintained. Such memories are called static memories. In contracts, Dynamic memories require not only the maintaining of a power supply, but also a periodic ―refresh‖ to maintain the information stored in them. Dynamic memories can have very high bit densities and very lower power consumption relative to static memories and are thus generally used to realize the main memory unit.

Dynamic Memories:-

The basic idea of dynamic memory is that information is stored in the form of a charge on the capacitor. An example of a dynamic memory cell is shown below:

When the transistor T is turned on and an appropriate voltage is applied to the bit line, information is stored in the cell, in the form of a known amount of charge stored on the capacitor. After the transistor is turned off, the capacitor begins to discharge. This is caused by the capacitor‘s own leakage resistance and the very small amount of current that still flows through the transistor. Hence the data is read correctly only if is read before the charge on the capacitor drops below some threshold value. During a Read



operation, the bit line is placed in a high-impendence state, the transistor is turned on and a sense circuit connected to the bit line is used to determine whether the charge on the capacitor is above or below the threshold value. During such a Read, the charge on the capacitor is restored to its original value and thus the cell is refreshed with every read operation.

Typical Organization of a Dynamic Memory Chip:-

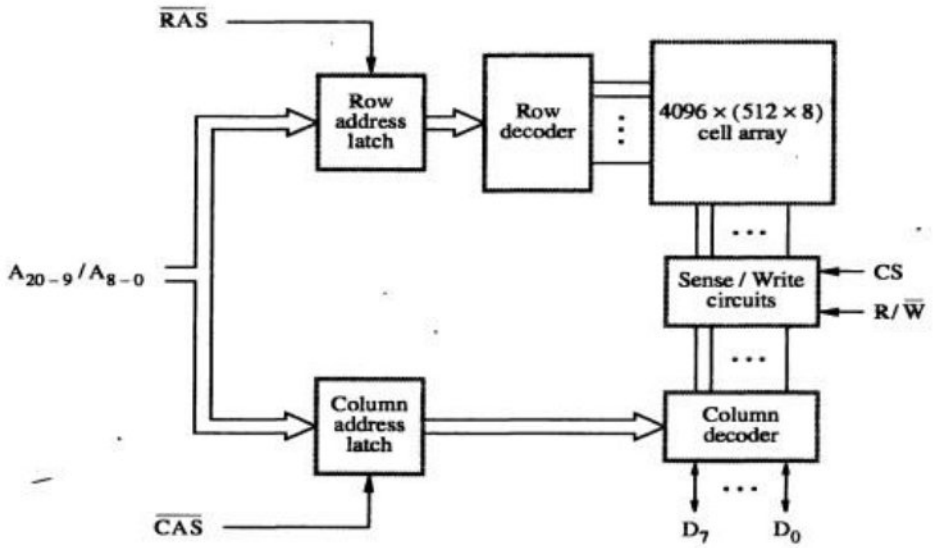


Figure 5.7 Internal organization of a 2M × 8 dynamic memory chip.

:

The cells are organized in the form of a square array such that the high-and lower-order 8 bits of the 16-bit address constitute the row and column addresses of a cell, respectively. In order to reduce the number of pins needed for external connections, the row and column address are multiplexed on 8 pins. To access a cell, the row address is applied first. It is loaded into the row address latch in response to a single pulse on the Row Address Strobe (RAS) input. This selects a row of cells. Now, the column address is applied to the address pins and is loaded into the column address latch under the control of the Column Address Strobe (CAS) input and this address selects the appropriate sense/write circuit. If the R/W signal indicates a Read operation, the output of the selected circuit is transferred to the data output. Do. For a write operation, the data on the DI line is used to overwrite the cell selected.

It is important to note that the application of a row address causes all the cells on the corresponding row to be read and refreshed during both Read and Write operations. To ensure that the contents of a dynamic memory are maintained, each row of cells must

be addressed periodically, typically once every two milliseconds. A Refresh circuit performs this function. Some dynamic memory chips incorportate a refresh facility the chips themselves and hence they appear as static memories to the user! such chips are often referred to as Pseudostatic.

Another feature available on many dynamic memory chips is that once the row address is loaded, successive locations can be accessed by loading only column addresses. Such blocktransfers can be carried out typically at a rate that is double that for transfers involving random addresses. Such a feature is useful when memory access follow a regular pattern, for example, in a graphics terminal.

Because of their high density and low cost, dynamic memories are widely used in the main memory units of computers. Commercially available chips range in size from 1k to 4M bits or more, and are available in various organizations like 64k x 1, 16k x 4, 1MB x 1 etc.

DESIGN CONSIDERATION FOR MEMORY SYSTEMS:-

The choice of a RAM chip for a given application depends on several factors like speed, power dissipation, size of the chip, availability of block transfer feature etc.

Bipolar memories are generally used when very fast operation is the primary requirement. High power dissipation in bipolar circuits makes it difficult to realize high bit densities.

Dynamic MOS memory is the predominant technology used in the main memories of computer, because their high bit-density makes it possible to implement large memories economically.

StaticMOSmemory chips have higher densities and slightly longer access times compared to bipolar chips. They have lower densities than dynamic memories but are easier to use because they do not require refreshing.

Design using static Memory Chips: -

Consider the design of a memory system of 64k x 16 using 16k x 1 static memory chips. We can use a column of 4 chips to implement one bit position. Sixteen such sets provide the required 64k x 16 memories. Each chip has a control input called chip select, which should be set to 1 to enable the chip to participate in a read or write operation. When this signal is 0, the chip is electrically isolated from the rest of the system. The high-order 2 bits of the 16-bit address bus are decoded to obtain the four chip select control signals, and the remaining 14address bits are used to access specific bit locations inside each chip of the selected row. The W input of all chips are fed together to provide a common Read/ *Write* control. This organization is shown in the following figure.

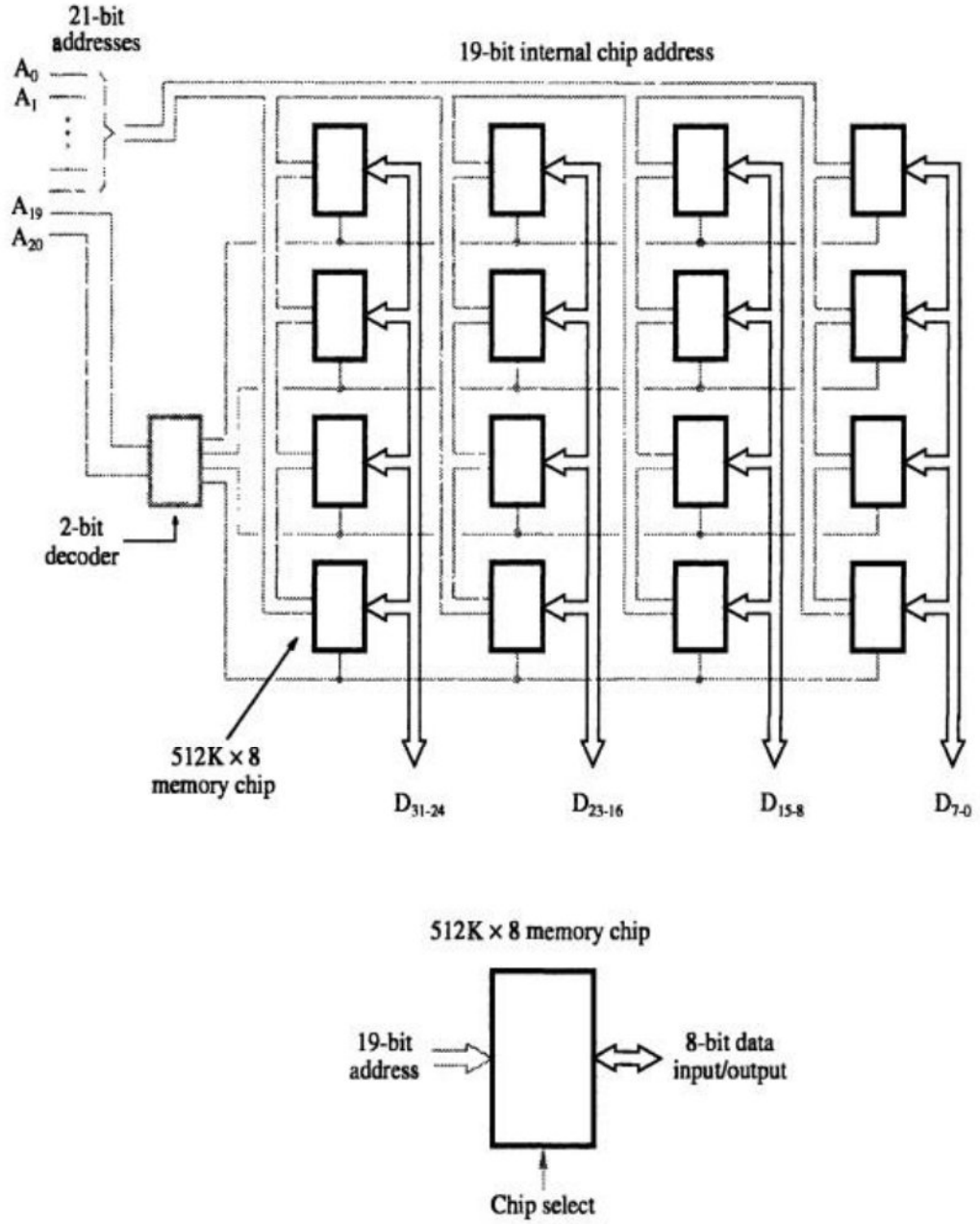


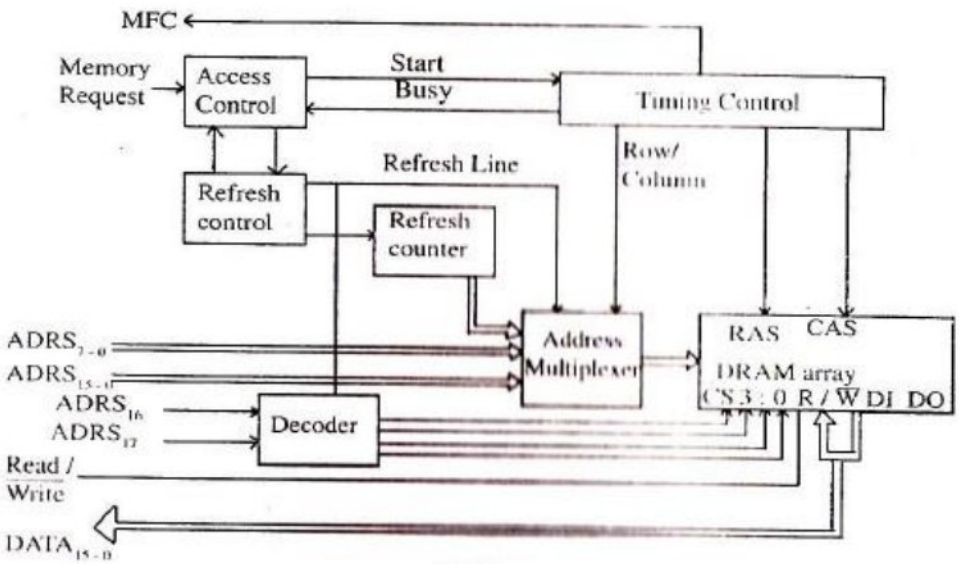
Figure 5.10 Organization of a 2M × 32 memory module using 512K × 8 static memory chips.

Design using Dynamic Memory Chips: -

The design of a memory system using dynamic memory chips is similar to the design using static memory chips, but with the following important differences in the control circry.

- The row and column parts of the address of each chip have to be multiplexed;
- A refresh circuit is needed; and
- The timing of various steps of a memory cycle must be carefully controlled.

A memory system of 256k x 16 designed using 64k x 1 DRAM chips, is shown below;



The memory unit is assumed to be connected to an asynchronous memory bus that has 18 address lines (ADRS 17-0), 16 data lines (DATA15-0), two handshake signals (Memory request and MFC), and a Read/ *Write* line to specify the operation (read to Write).

The memory chips are organized into 4 rows, with each row having 16 chips. Thus each column of the 16 columns implements one bit position. The higher order 12 bits of the address are decoded to get four chip select control signals which are used to

select one of the four rows. The remaining 16 bits, after multiplexing, are used to access specific bit locations inside each chip of the selected row. The R/W inputs of all chips are tied together to provide a common Read/Write control. The DI and DO lines, together, provide $D_{15} - D_0$ i.e. the data bus $DATA_{15-0}$.

The operation of the control circuit can be described by considering a normal memory read cycle. The cycle begins when the CPU activates the address, the Read/ $\overline{Write}$ and the Memory Request lines. When the memory request line becomes active, the Access control block sets the start signal to 1. The timing control block responds by setting Busy lines to 1, in order to prevent the access control block from accepting new requests until the current cycle ends. The timing control block then loads the row and column addresses into the memory chips by activating the RAS and CAS lines. During this time, it uses the Row/ $\overline{Column}$ line to select first the row address, $ADRS_{15-8}$, followed by the column address, $ADRS_{(7-0)}$.

Refresh Operation:-

The Refresh control block periodically generates Refresh, requests, causing the access control block to start a memory cycle in the normal way. This block allows the refresh operation by activating the Refresh Grant line. The access control block arbitrates between Memory Access requests and Refresh requests, with priority to Refresh requests in the case of a tie to ensure the integrity of the stored data.

As soon as the Refresh control block receives the Refresh Grant signal, it activates the Refresh line. This causes the address multiplexer to select the Refresh counter as the source and its contents are thus loaded into the row address latches of all memory chips when the RAS signal is activated. During this time the R/ $\overline{W}$ line may be low, causing an inadvertent write operation. One way to prevent this is to use the Refresh line to control the decoder block to deactivate all the chip select lines. The rest of the refresh cycle is the same as in a normal cycle. At the end, the Refresh control block increments the refresh counter in preparation for the next Refresh cycle.

Even though the row address has 8 bits, the Refresh counter need only be 7 bits wide because of the cell organization inside the memory chips. In a 64k x 1 memory chip, the 256x256 cell array actually consists of two 128x256 arrays. The low order 7 bits of the row address select a row from both arrays and thus the row from both arrays is refreshed!

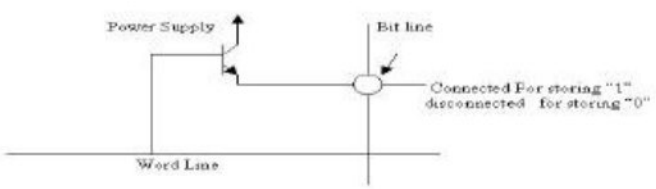
Ideally, the refresh operation should be transparent to the CPU. However, the response of the memory to a request from the CPU or from a DMA device may be delayed if a Refresh operation is in progress. Further, in the case of a tie, Refresh operation is given priority. Thus the normal access may be delayed. This delay will be more if all memory rows are refreshed before the memory is returned to normal use. A more common scheme, however, interleaves Refresh operations on successive rows with accesses from the memory bus. In either case, Refresh operations generally use less than 5% of the available memory cycles, so the time penalty caused by refreshing is very small.

The variability in the access times resulting from refresh can be easily accommodated in an asynchronous bus scheme. With synchronous buses, it may be necessary for the Refresh circuit to request bus cycles as a DMA device would!

Semi Conductor Rom Memories: -

Semiconductor read-only memory (ROM) units are well suited as the control store components in micro programmed processors and also as the parts of the main memory that contain fixed programs or data.

The following figure shows a possible configuration for a bipolar ROM cell.



The word line is normally held at a low voltage. If a word is to be selected, the voltage of the corresponding word line is momentarily raised, which causes all transistors whose

emitters are connected to their corresponding bit lines to be turned on. The current that flows from the voltage supply to the bit line can be detected by a sense circuit. The bit positions in which current is detected are read as 1s, and the remaining bits are read as Os. Therefore, the contents of a given word are determined by the pattern of emitter to bit-line connections similar configurations are possible in MOS technology.

Data are written into a ROM at the time of manufacture programmable ROM (PROM) devices allow the data to be loaded by the user. Programmability is achieved by connecting a fuse between the emitter and the bit line. Thus, prior to programming, the memory contains all 1s. The user can inserts Os at the required locations by burning out the fuses at these locations using high-current pulses. This process is irreversible.

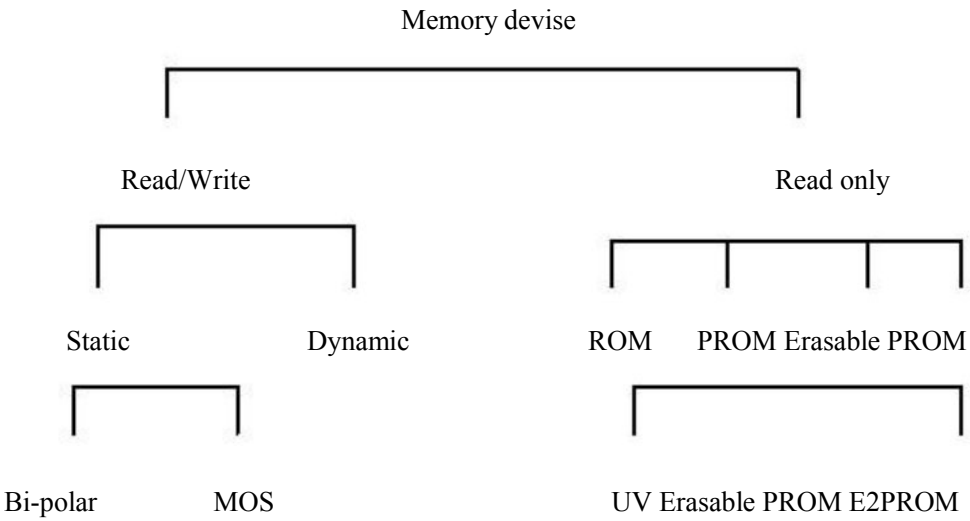
ROMs are attractive when high production volumes are involved. For smaller numbers, PROMs provide a faster and considerably less expensive approach.

Chips which allow the stored data to be erased and new data to be loaded. Such a chip is an erasable, programmable ROM, usually called an EPROM. It provides considerable flexibility during the development phase. An EPROM cell bears considerable resemblance to the dynamic memory cell. As in the case of dynamic memory, information is stored in the form of a charge on a capacitor. The main difference is that the capacitor in an EPROM cell is very well insulated. Its rate of discharge is so low that it retains the stored information for very long periods. To write information, allowing charge to be stored on the capacitor.

The contents of EPROM cells can be erased by increasing the discharge rate of the storage capacitor by several orders of magnitude. This can be accomplished by allowing ultraviolet light into the chip through a window provided for that purpose, or by the application of a high voltage similar to that used in a write operation. If ultraviolet light is used, all cells in the chip are erased at the same time. When electrical erasure is used, however, the process can be made selective. An electrically erasable EPROM, often referred to as EEPROM. However, the circuit must now include high voltage generation.

Some E2PROM chips incorporate the circuitry for generating these voltages o the chip itself. Depending on the requirements, suitable device can be selected.

Classification of memory devices



5.5 CACHE MEMORY:

Analysis of a large number of typical programs has shown that most of their execution time is spent on a few main row lines in which a number of instructions are executed repeatedly. These instructions may constitute a simple loop, nested loops or few procedure that repeatedly call each other. The main observation is that many instructions in a few localized are as of the program are repeatedly executed and that the remainder of the program is accessed relatively infrequently. This phenomenan is referred to as locality of reference.

If the active segments of a program can be placed in a fast memory, then the total execution time can be significantly reduced, such a memory is referred as a cache memory which is in served between the CPU and the main memory as shown in fig.1

Fig.1 cache memory between main memory & CPU.

Two Level memory Hierarchy: We will adopt the terms Primary level for the smaller, faster memory and the secondary level for larger, slower memory, we will also allow cache to be a primary level with slower semiconductor memory as the corresponding secondary level. At a different point in the hierarchy, the same S.C memory could be the primary level with disk as the secondary level.

Primary and Secondary addresses:-

A two level hierarchy and its addressing are illustrated in fig.2. A system address is applied to the memory management unit (MMU) that handles the mapping function for the particular pair in the hierarchy. If the MMU finds the address in the Primary level, it provides Primary address, which selects the item from the Primary memory. This translation must be fast, because every time memory is accessed, the system address must be translated. The translation may fail to produce a Primary address because the requested items is not found, so that information can be retrieved from the secondary level and transferred to the Primary level.

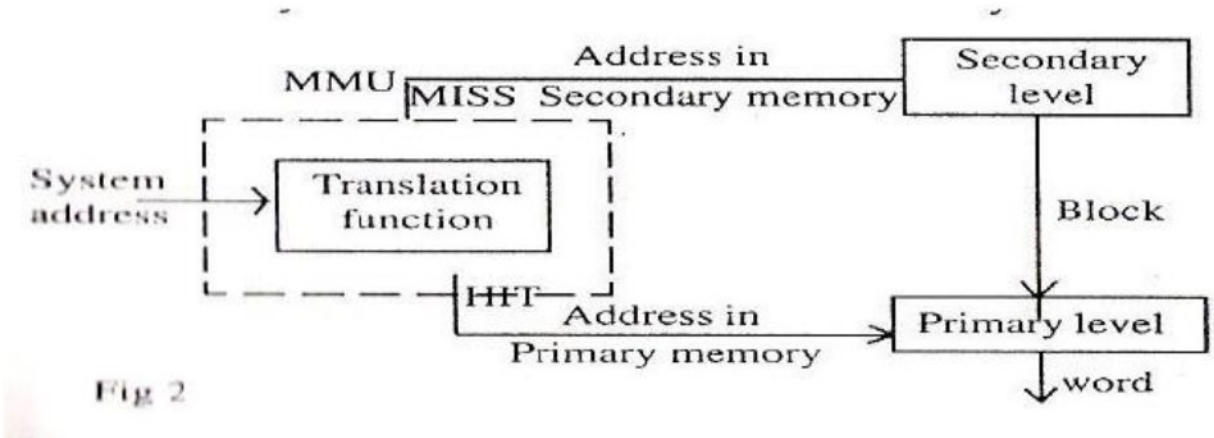


Fig 2

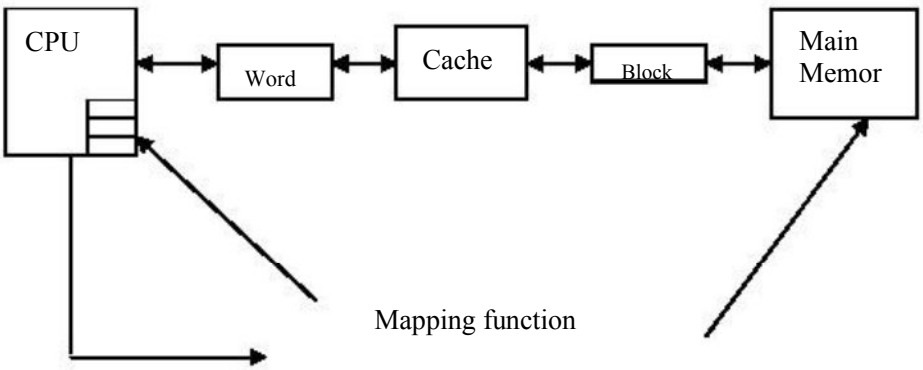
Hits and Misses:- Successful translation of reference into Primary address is called a hit, and failure is a miss. The hit ratio is (1-miss ratio). If t_p is the Primary memory access time and t_s is the secondary access time, the average access time for the two level hierarchy is

$$t_a = h\ t_p + (1-h)t_s$$

The Cache:- The Mapping Function

Fig. Shows a schematic view of the cache mapping function. The mapping function is responsible for all cache operations. It is implemented in hardware, because of the required high speed operation. The mapping function determines the following.

- Placement strategies – Where to place an incoming block in the cache
- Replacement strategies – Which block to replace when a cache miss occurs
- How to handle Read and Writes up as cache hit and mises



Three different types of mapping functions are in common use

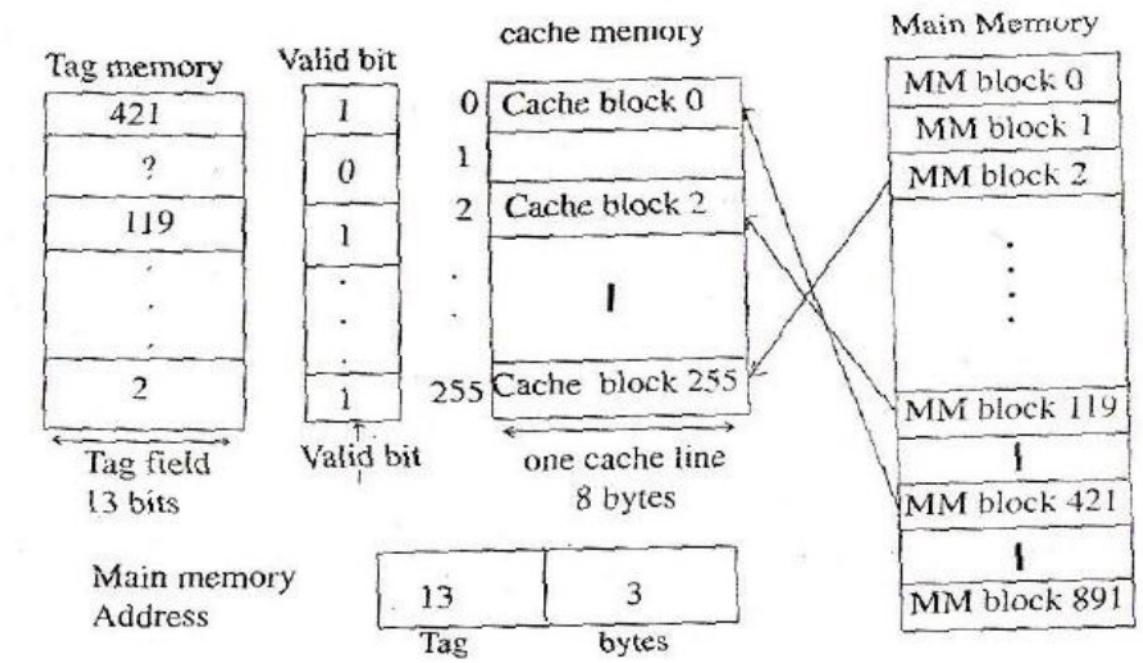
1. Associative mapping cache
2. Direct mapping cache
3. Block-set-associative mapping cache

1. Associative mapped caches:- In this any block from main memory can be placed any where in the cache. After being placed in the cache, a given block is identified uniquely by its main memory block number, referred to as the tag, which is stored inside a separate tag memory in the cache.

Regardless of the kind of cache, a given block in the cache may or may not contain valid information. For example, when the system has just been powered up add before the cache has had any blocks loaded into it, all the information there is invalid. The cache maintains a valid bit for each block to keep track of whether the information in the corresponding block is valid.

Fig4. shows the various memory structures in an associative cache, The cache itself contain 256, 8byte blocks, a 256 x 13 bit tag memory for holding the tags of the blocks currently stored in the cache, and a 256 x 1 bit memory for storing the valid bits, Main memory contains 8192, 8 byte blocks. The figure indicates that main memory address references are partition into two fields, a 3 bit word field describing the location of the desired word in the cache line, and a 13 bit tag field describing the main memory block number desired. The 3 bit word field becomes essentially a ―cache address‖ specifying where to find the word if indeed it is in the cache.

The remaining 13 bits must be compared against every 13 bit tag in the tag memory to see if the desired word is present.



In the fig, above, main memory block 2 has been stored in the 256 cache block and so the 256th tag entry is 2 mm block 119 has been stored in the second cache block

corresponding entry in tag memory is 119 mm block 421 has been stored in cache block 0 and tag memory location 0 has been set to 421. Three valid bits have also been set, indicating valid information in these locations.

The associative cache makes the most flexible and complete use of its capacity, storing blocks wherever it needs to, but there is a penalty to be paid for this flexibility the tag memory must be searched in for each memory reference.

Fig.5 shows the mechanism of operation of the associative cache. The process begins with the main memory address being placed in the argument register of the (associative) tag memory (1) if there is a match (hit), (2) and if the ratio bit for the block is set (3), then the block is gated out of the cache (4), and the 3 bit offset field is used to select the byte corresponding to the block offset field of the main memory address (5) That byte is forwarded to the CPU, (6)

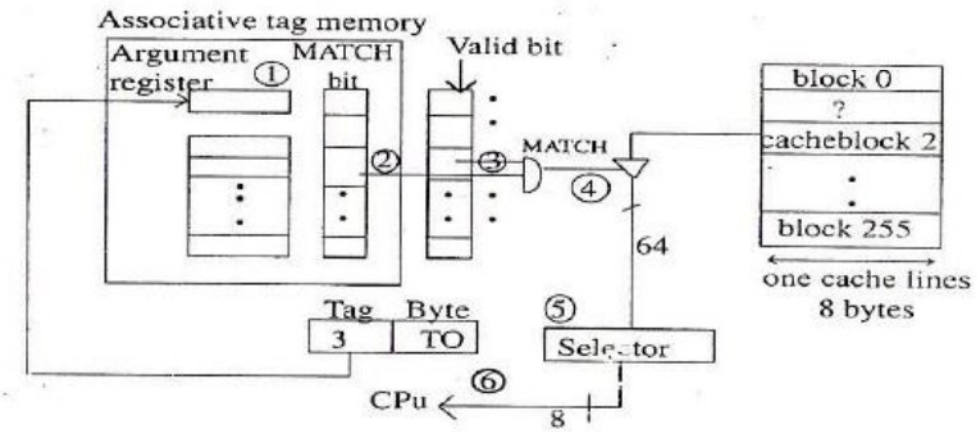


Fig. 5 Associative cache mechanism:

2. Direct mapped caches:- In this a given main memory block can be placed in one and only one place in the cache. Fig6. Shows an example of a direct – mapped cache. For simplicity, the example again uses a 256 block x 8 byte cache and a 16 bit main memory address. The main memory in the fig. has 256 rows x 32 columns, still fielding 256 x 32 = 8192 = 2_{13} total blocks as before. Notice that the main memory address is partitioned into 3 fields. The word field still specifies the word in the block. The group field specifies

which of the 256 cache locations the block will be in, if it is indeed in the cache. The tag field specifies which of the 32 blocks from main memory is actually present in the cache. Now the cache address is composed of the group field, which specifies the address of the block location in the cache and the word field, which specifies the address of the word in the block. There is also valid bit specifying whether the information in the selected block in valid.

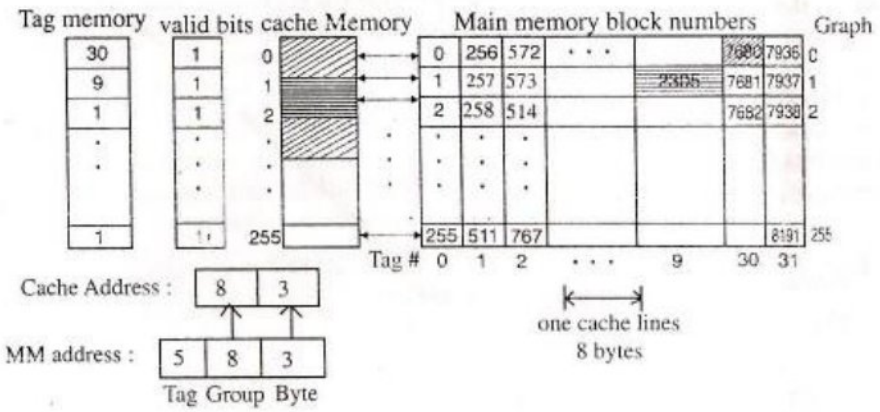
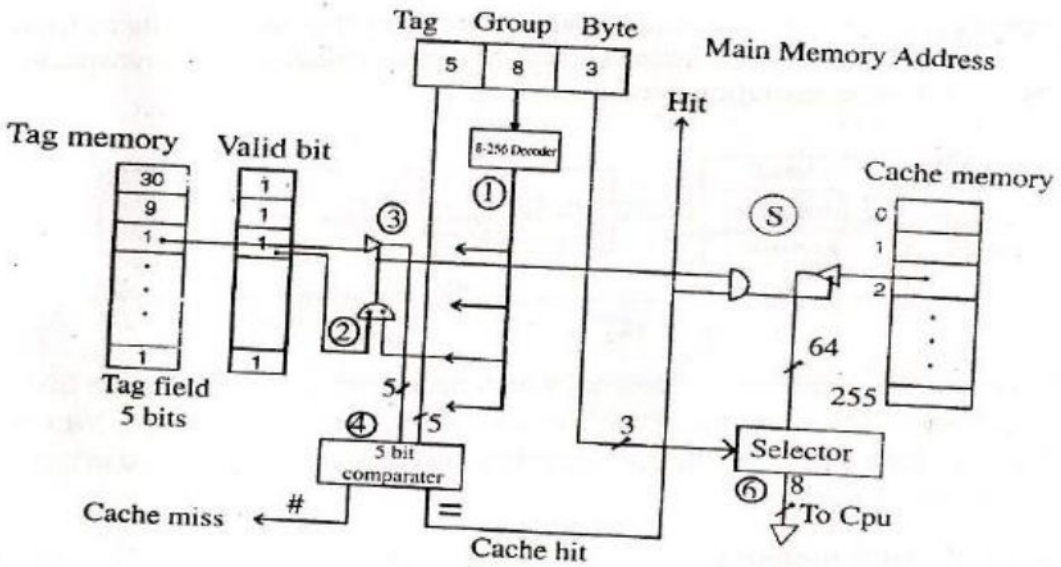


Fig 6. Direct mapped cache.

The fig.6 shows block 7680, from group 0 of MM placed in block location 0 of the cache and the corresponding tag set to 30. Similarly MM block 259 is in MM group 2, column 1, it is placed in block location 2 of the cache and the corresponding tag memory entry is 1.

The tasks required of the direct – mapped cache in servicing a memory request are shown in fig7.

The fig. shows the group field of the memory address being decoded 1) and used to select the tag of the one cache block location in which the block must be stored if it is the cache. If the valid bit for that block location is gated (2), then that tag is gated out, (3) and compared with the tag of the incoming memory address (4), A cache hit gates the cache block out (5) and the word field selects the specified word from the block (6), only one tag needs to be compared, resulting in considerably less hardware than in the associative memory case.



The direct mapped cache has the advantage of simplicity, but the obvious disadvantage that only a single block from a given group can be present in the cache at any given time.

3. Block-set-Associative cache:-

Block-set-Associative caches share properties of both of the previous mapping functions. It is similar to the direct-mapped cache, but now more than one block from a given group in main memory can occupy the same group in the cache. Assume the same main memory and block structure as before, but with the cache being twice as large, so that a set of two main memory blocks from the same group can occupy a given cache group.

Fig 8 shows a 2 way set associative cache that is similar to the direct mapped cache in the previous example, but with twice as many blocks in the cache, arranged so that a set of any two blocks from each main memory group can be stored in the cache. MM is still partitioned into an 8 bit set field and a 5 bit tag field, but now there are two

possible places in which a given block can reside and both must be searched associatively.

The cache group address is the same as that of the direct matched cache, an 8 bit block location and a 3 bit word location. Fig 8 shows that the cache entries corresponding to the second group contains blocks 513 and 2304. the group field now called the set field, is again decoded and directs the search to the correct group and new only the tags in the selected group must be searched. So instead of 256 compares, the cache only needs to do 2.

For simplicity, the valid bits are not shown, but they must be present. The cache hard ware would be similar so that shown in fig 7. but there would be two simultaneous comparisons of the two blocks in the set.

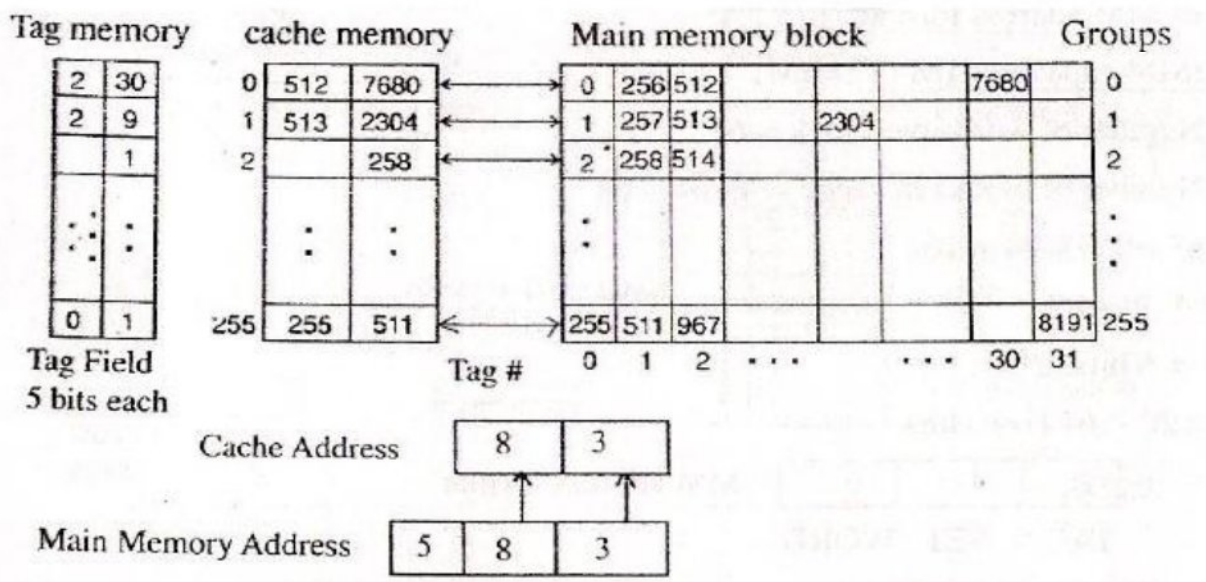


Fig 8. Two-way-set Associative cache.

Solved Problems:-

1. A block set associative cache consists of a total of 64 blocks divided into 4 block sets. The MM contains 4096 blocks each containing 128 words.

a) How many bits are there in MM address?

b) How many bits are there in each of the TAG, SET & word fields

Solution:- Number of sets = 64/4 = 16

Set bits = 4(2^4 = 16)

Number of words = 128

Word bits = 7 bits (2^7 = 128)

MM capacity : 4096 x 128 (2^{12} x 2^7 = 2^{19})

a) Number of bits in memory address = 19 bits

b)

8	4	7

TAG SET WORD

TAG bits = 19 – (7+4) = 8 bits.

2. A computer system has a MM capacity of a total of 1M 16 bits words. It also has a 4K words cache organized in the block set associative manner, with 4 blocks per set & 64 words per block. Calculate the number of bits in each of the TAG, SET & WORD fields of MM address format.

Solution: Capacity: 1M (2^{20} = 1M)

Number of words per block = 64

Number of blocks in cache = 4k/64 = 64

Number of sets = 64/4 = 16

Set bits = 4 (2^4 = 16)

Word bits = 6 bits (2^6 = 64)

Tag bits = 20-(6+4) = 10 bits

MM address format

10	4	6

Virtual Memory:-

Virtual memory is the technique of using secondary storage such as disks to enter the apparent size of accessible memory beyond its actual physical size. Virtual memory is implemented by employing a memory-management unit (MMU) to translate every logical address reference into a physical address reference as shown in fig 1. The MMU is imposed between the CPU and the physical memory where it performs these translations under the control of the operating system. Each memory reference is sued by the CPU is translated from the logical address space to the physical address space. Mapping tables guide the translation, again under the control of the operating system.

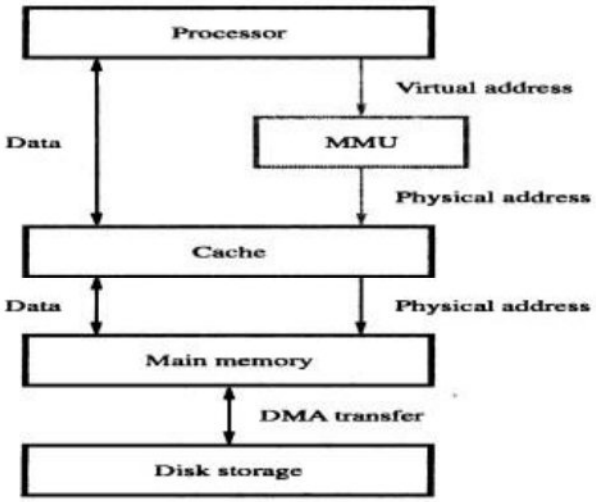


Figure 5.26 Virtual memory organization.

Virtual memory usually demand paging, which means that a Page is moved from disk into main memory only when the processor accesses a word on that page. Virtual memory pages always have a place on the disk once they are created, but are copied to main memory only on a miss or page fault.

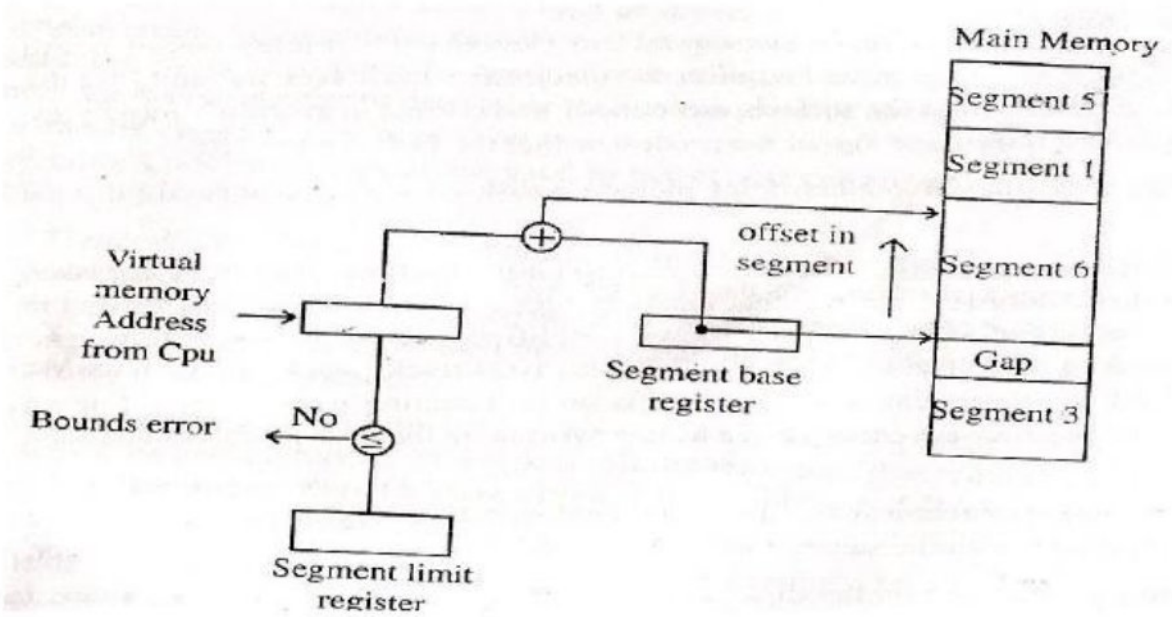
Advantages of Virtual memory:-

1. **Simplified addressing:-** Each program unit can be compiled into its own memory space, beginning at address O and extending far beyond the limits of physical memory. Programs and data structures do not require address relocation

at load time, nor must they be broken into fragments merely to accommodate memory limitations.

2. **Cost effective use of memory:** - Less expensive disk storage can replace more expensive RAM memory, since the entire program does not need to occupy physical memory at one time.
3. **Access control:** - Since each memory reference must be translated, it can be simultaneously checked for read, write and execute privileges. This allows hardward level control of access to system resources and also prevents and also prevents buggy programs or intruders from causing damage to the resources of other users or the system.

a) Memory management by segmentation:- Segmentation allows memory to be divided into segments of varying sizes depending upon requirements. Fig 2. Shows a main memory containing five segments identified by segment numbers. Each segment beings at a virtual address 0, regardless of where it is located in physical memory.



Each virtual address arriving from the CPU is added to the contents of the segment base register in the MMU to form the physical address. The virtual address may also optionally be compared to a segment limit register to trap reference beyond a specified limit.

b) Memory management by paging:- Fig 3 shows a simplified mechanism for virtual address translation in a paged MMU. The process begins in a manner similar to the segmentation process. The virtual address composed of a high order page number and a low order word number is applied to MMU. The virtual page number is limit checked to be certain that the page is within the page table, and if it is, it is added to the page table base to yield the page table entry. The page table entry contains several control fields in addition to the page field. The control fields may include access control bits, a presence bit, a dirty bit and one or more use bits, typically the access control field will include bits specifying read, write and perhaps execute permission. The presence bit indicates whether the page is currently in main memory. The use bit is set upon a read or write to the specified page, as an indication to the replaced algorithm in case a page must be replaced.

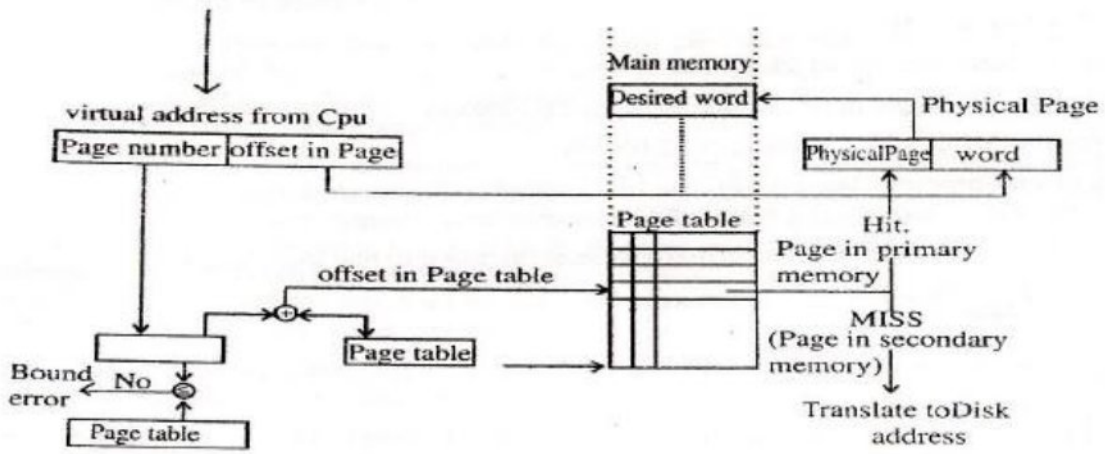


Fig. 12. Virtuel address translation in a paged MMU.

If the presence bit indicates a hit, then the page field of the page table entry will contain the physical page number. If the presence bit is a miss, which is page fault, then the page field of the page table entry which contains an address is secondary memory where the page is stored. This miss condition also generates an interrupt. The interrupt service routine will initiate the page fetch from secondary memory and with also suspended the requesting process until the page has been bought into main memory. If

the CPU operation is a write hit, then the dirty bit is set. If the CPU operation is a write miss, then the MMU with begin a write allocate process.

Problems:-

1. An address space is specified by 24 bits & the corresponding memory space is 16 bits.

   a) How many words are there in address space?

   b) How many words are there in memory space?

   c) If a page has 2k words, how many pages & blocks are in the system?

Solution:-

a) Address space = 24 bits

$2^{24} = 2^4.2^{20} = 16M$ words

b) Memory space: 16 bits

$2^{16} = 64k$ words

c) page consists of 2k words

Number of pages in add space = 16M/2K = 8000

Number of blocks = 64k/2k = 32 blocks

2. A virtual memory has a page size of 1k words. There are 8 pages & 4 blocks. The associative memory page table has the following entries.

Page	block
0	3 by cpu
1	1
4	2
6	0

Make a list of virtual addresses (in decimal) that will cause a page fault of used

Memory System Design

@ Dayanand Sagar University, Bengaluru

Presented By:
PROF. KRISHNANANDA L
DEPARTMENT OF ECE,
GOVT SKSJ TECHNOLOGICAL INSTITUTE,
BENGALURU.


THERE ARE TWO WAYS OF SPREADING LIGHT
TO BE THE CANDLE OR THE MIRROR THAT REFLECTS IT
Edith Wharton

Outline of the Talk

- Introduction
- Basic Concepts
- Memory Terminology
- Primary Memory
- Memory Expansion
- Flash Memory
- Cache Memory
- Cache Operation
- Cache –Mapping Techniques
- Cache –Performance
- Introduction to Virtual Memory
- Virtual Memory using Paging
- Conclusion
- References

Introduction

- Basic structure of a Computing System (Fig.1)
- "Memory" – an important and integral component to deliver better performance.
- Ideal Characteristics –High Speed, Large capacity & Low cost
- Memory Design: Very critical & challenging

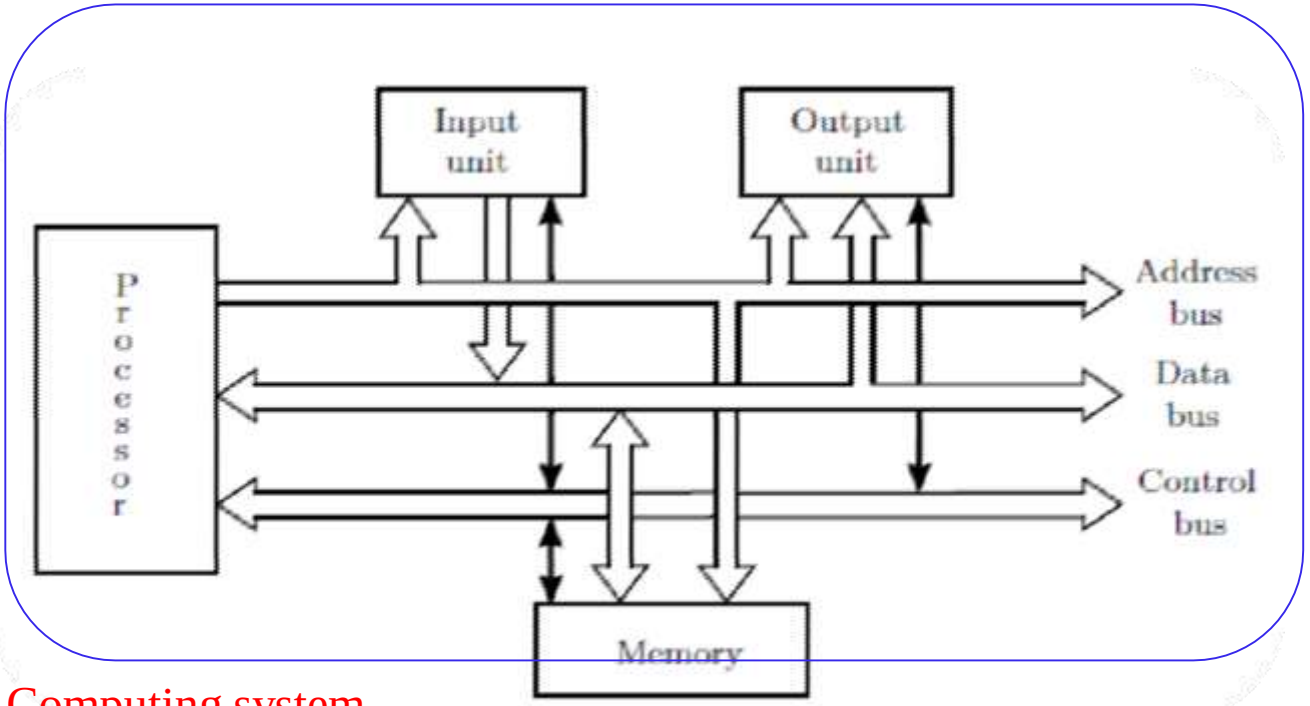
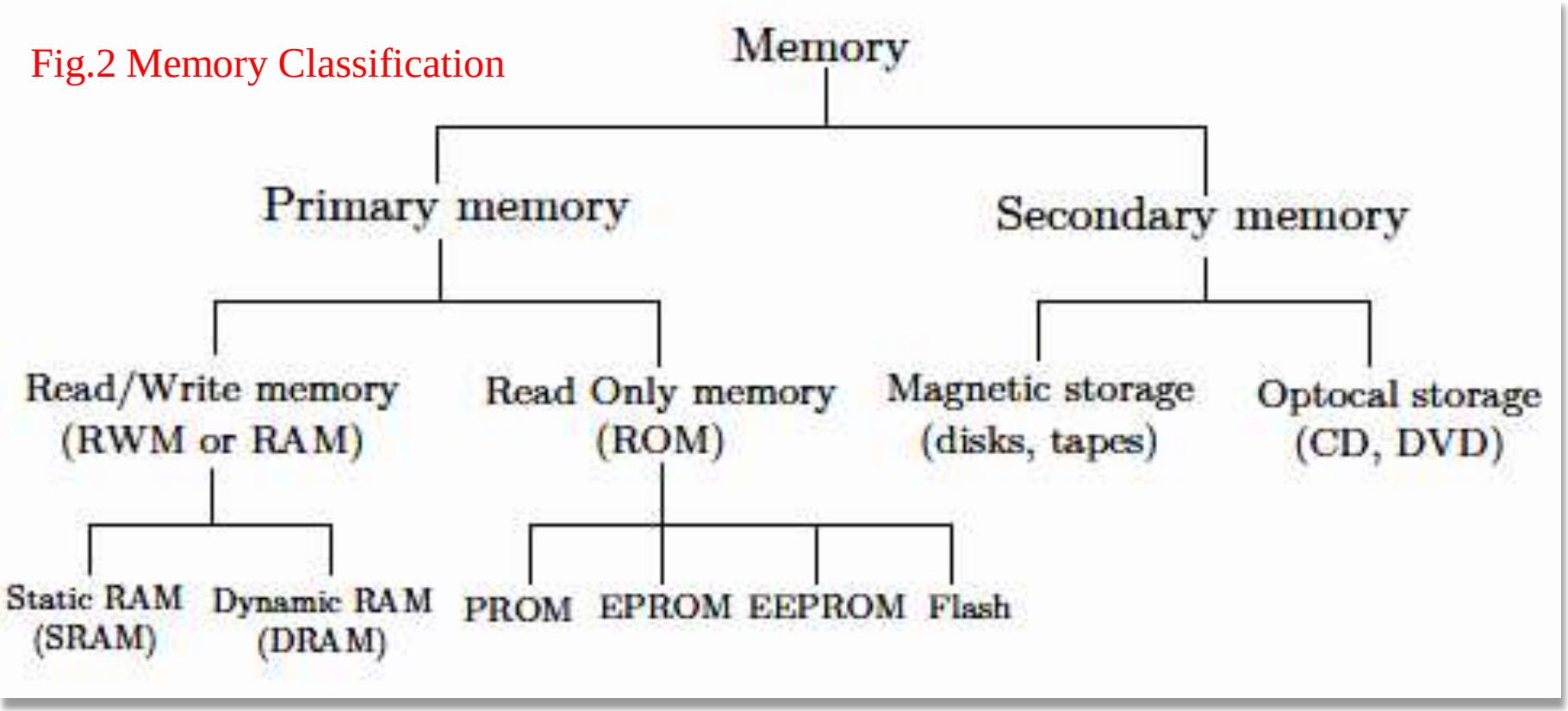


Fig.1 Structure of Computing system

Basic Concepts

- Memory Classification (Fig.2)
- a) Primary / Main/ Semiconductor memory
  b) Secondary/ Auxiliary memory /Mass storage

Fig.2 Memory Classification



Memory Terminology

- **Main/Primary Memory**: Semiconductor memory to which the CPU is directly connected. It stores instructions and data required by the CPU in "binary" form. It is faster but has less capacity. Ex: RAM, ROM
- **Auxiliary/Secondary Memory**: Also known as 'mass storage'. Used to store large volumes of data. It is slower, but economical. It's a permanent storage. Ex: Magnetic disks, CDs, DVDs etc
- **Read (Fetch) Operation**: Here, data stored in a specific memory location is accessed by the CPU (Processor) using the address.
- **Write (Store) Operation**: A new data (from CPU) is stored into a particular memory location using the address.
- **Memory Access time**: It's the total time elapsed between two events: i.e., the memory receiving a new address input and the data becoming available at the data bus. Access time is the speed with which memory responds.
- **Volatile memory**: If the power is switched-off, all information stored in the memory is lost.
- **Primary Memory :** Read-only or Read-Write
- **Read-only Memory (ROM):** The information stored can only be read, but not written into. A kind of primary memory but, *Non-volatile*

Memory Terminology contd..

- **Random-Access Memory (RAM)**: the access time is the same for any address in memory. i.e., the time taken to access any location is constant and independent of its physical address (location). RAM is a type of Primary memory & is *volatile.* The term RAM stands for *Random-access memory*; however, in literature, it means Read-write memory (RWM).
- **Memory cell**: An electrical circuit or a device to store a *single bit* (binary 0 or 1) Ex: Flip-flop, Charge on a capacitor etc
- **Memory Word**: A group of bits that represent some kind of "information" i.e., data or instruction. Word size may be 8-bits, 16-bits, 32- bits and so on [power of 2].
- **Byte**: Group of 8-bits is known as Byte. So, 16-bits is taken as 2 bytes, 32- bits as 4 bytes etc., Different type of data have different lengths. Ex: Integer is 2 or 4 bytes, Gray scale pixel is 1 byte, an ASCII character is 1 byte etc.,
- **Address**: Each location in the memory is identified with a unique "address" which is a numerical value (in decimal or Hex)
- **Address Bus**: The number of pins (on the CPU) used to carry address bits. If 'm' pins are used, total memory that can be connected to CPU is $L = 2^m$ bytes.
- **Data Bus**: The number of pins used to carry data between memory and CPU.

Primary Memory

- RAMs initially classified as: Static RAM (SRAM) and Dynamic RAM (DRAM). Recent variations are: SDRAM, DDRRAM, DDRRAM-2, 3, 4, etc
- Primary Memory is generally "byte addressable", i.e., each "location" of memory stores 8-bit of information

- **Memory Capacity/Density/Size**: Denotes the total number of bits that can be stored in memory. Capacity (L) = No. of Words (locations) x No. of bits/word
  = No. of locations (M) x 8-bits
  L = M bytes
- For 'k' bit address bus, memory capacity is $L = 2^k$ (M) bytes. If k=10, size is 2^{10}= 1KB (1 Kilo Byte). Similarly, 2^{20} = 1MB (1 Mega byte) and 2^{30} = 1GB (1 Giga Byte)

Fig.3 32x4 memory organization

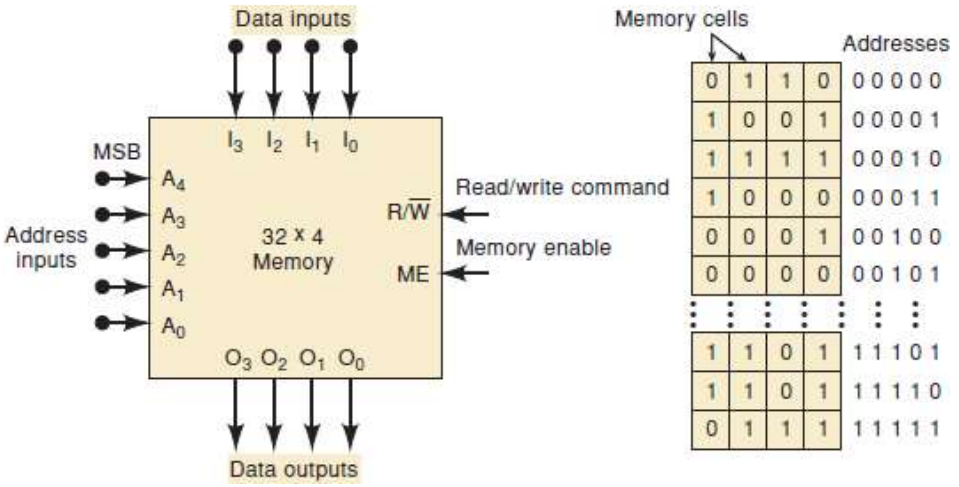
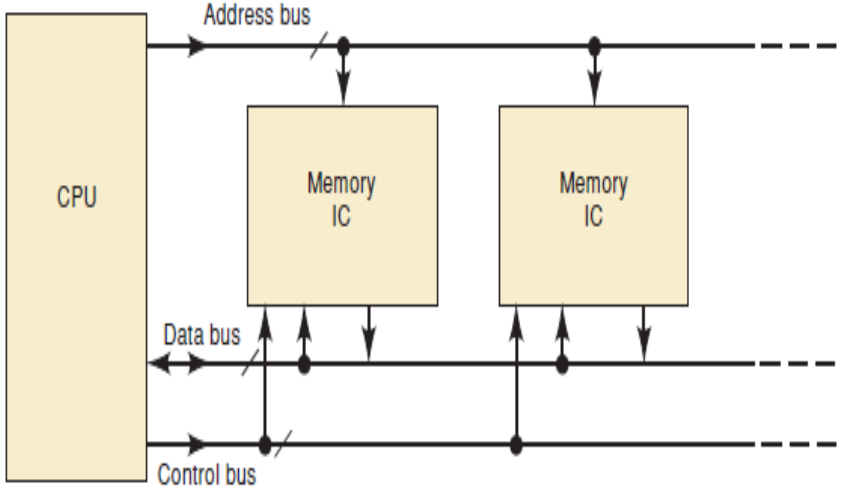


Fig.4 Buses connecting memory to CPU

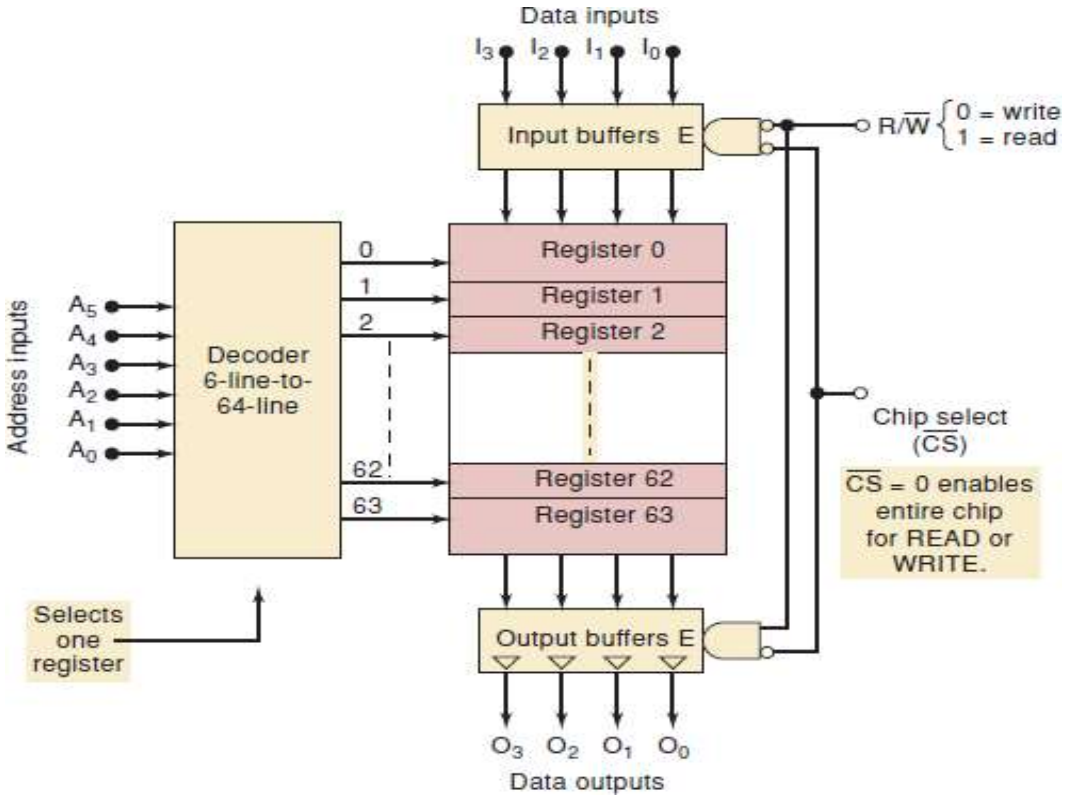


Primary memory - SRAM

- Static RAM (SRAM) – faster but, less capacity.
- Fig. 5 shows the structure of 64 x 4 SRAM.

- The basic blocks are: address decoder, 64 Registers (locations) each storing a single data word of 4 bits input & output buffers and Control signals (CS*, R/W*)
  - CS* is used to enable/disable RAM chip itself (to reduce power consumption) and also to cascade many memory chips.
- R/W* =1/0 for Read/Write operation
- Needs 18 external pin connections. To reduce the IC size, i/p and o/p pins can be combined as I/O pins which can be connected to the **"data bus"** of the CPU for data transfer.

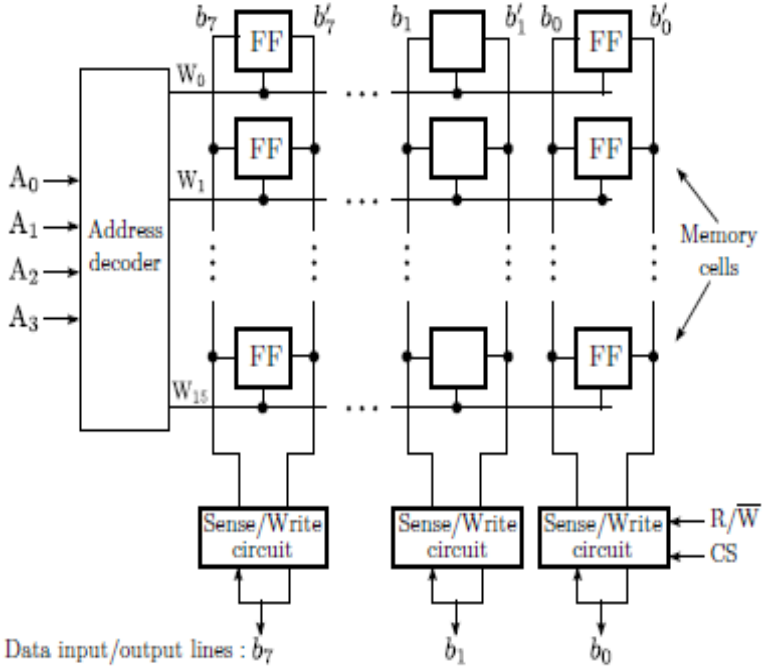
Fig.5 Architecture of 64x4 SRAM

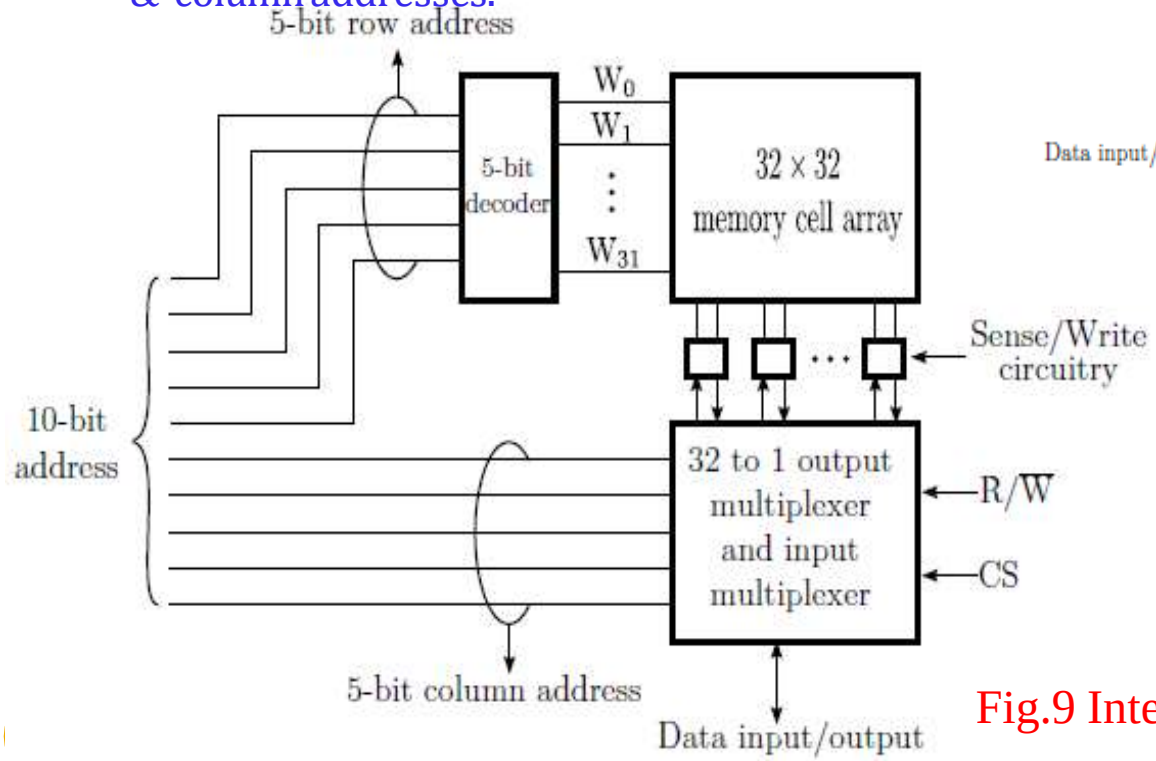


SRAM contd…

- SRAM: Flip-flop concept for storage.
- Fig. 8 shows the internal organization of 16 x 8 SRAM. It has a 4:16 decoder, 16 rows with 8 FFs in each row, Sense/Write Circuit, R/W* and CS signal, Data lines. This chip needs 16 pins.
- Fig.9 depicts the internal organization of 1Kx1 SRAM. Here, 1K (1024) cells are arranged in 1Kx1 format, with address lines split into row & column addresses.

Fig.8 Internal organization of 16x8 SRAM





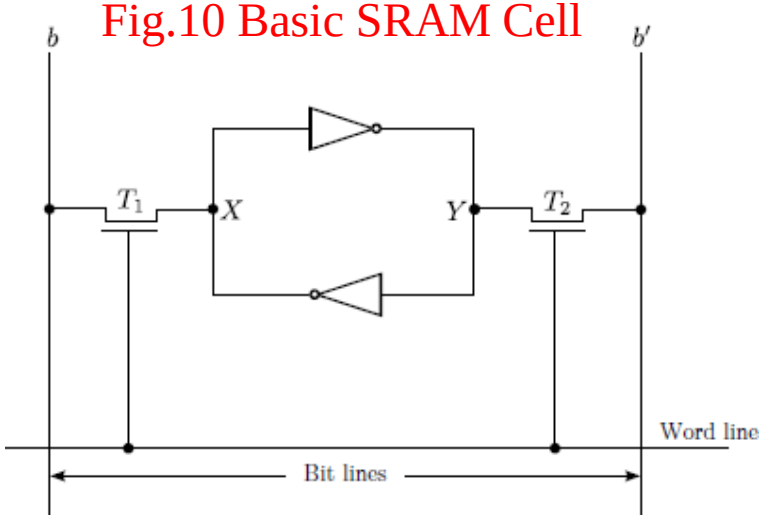
A row address selects a row of 32 cells, but column address connects only one of them to the external data line. This is done with 32:1 MUX and 1:32 DEMUX. The column address lines act as "select" lines for MUX/DEMUX

Fig.9 Internal organization of 1K x1 SRAM

SRAM Cell

- Fig.10 shows a basic Static RAM cell constructed using a latch. When Word line (W) =0, T1 and T2 are OFF and bit value (b) is retained. When W=1, T1 and T2 are ON. If the cell is in state 1, (X = 1, Y = 0), its available on bit line b and b'.
- The circuit in Fig.10 can be realized using CMOS logic (Fig.11) which has very low power consumption & low access time [faster]. An inverter is made of two transistors (pMOS and nMOS) together known as C-MOS (Complementary Metal Oxide Semiconductor). So, a 1-bit storage needs 6 transistors.
- The drawbacks of SRAM are: more transistors per bit of storage (not suitable to design large size mem) & Volatile.

Fig.10 Basic SRAM Cell



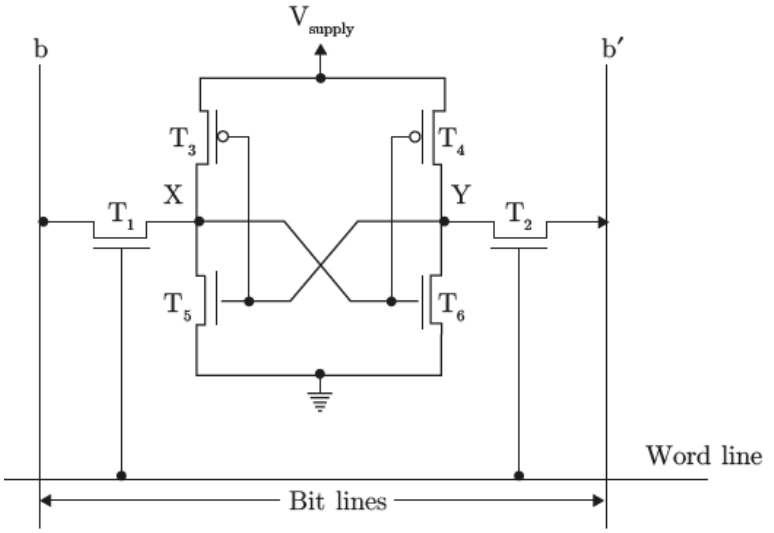
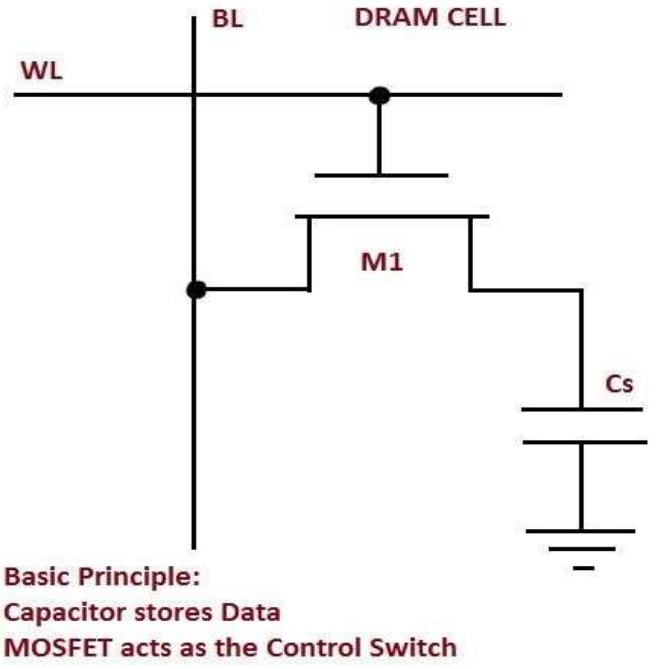


Fig.11 1-bit CMOS SRAM Cell

Asynchronous Dynamic RAM (DRAM)

- Store 1s and 0s as charges on a small capacitor. So, 1-bit storage needs just one capacitor instead of 6 transistors.
- Since capacitor discharges even though the power is on, DRAM needs a "refresh" circuitry. Refresh circuitry built on DRAM chip itself.
- Fig.12 is the structure of 1-bit DRAM Cell. To store information in this cell, transistor M1 is turned ON, appropriate voltage is applied to the bit line and the capacitor gets charged.
- DRAM offers high density @ low cost.
- DRAMs are preferred when cost, space and power requirements are critical. However, they are slower & complex compared to SRAMs

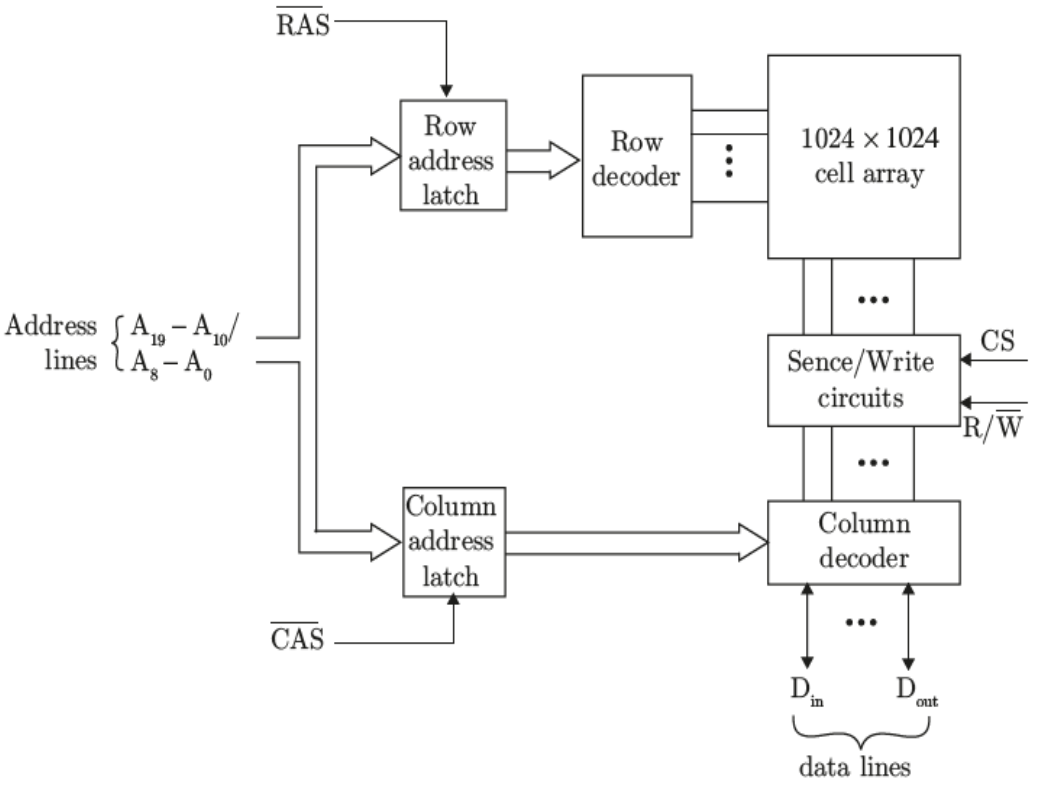
Fig.12 1-bit DRAM Cell



DRAM Contd..

- Fig.13 shows the organization of 1M x 1 DRAMchip
- 1M chip needs 20 address lines, split into row and column addresses.
- Two control signals RAS* (Row Address Strobe) and CAS* (Column Address Strobe) are used to enable the corresponding latches.
- The Row decoder selects one row out of 1024 rows.
- To reduce the total pins on the chip, address and data lines are multiplexed.

Fig.13 1M x 1 DRAM



- Initially, the row address is loaded into row address latch. All the cells of the selected row are refreshed. Then column address is loaded into latch, decoded and appropriate group of Sense/Write circuits are selected. If R/W* =1, the output of the selected circuit is put on data lines. Else, the info. on Data lines is transferred to the selected circuits.

Synchronous DRAM (SDRAM)

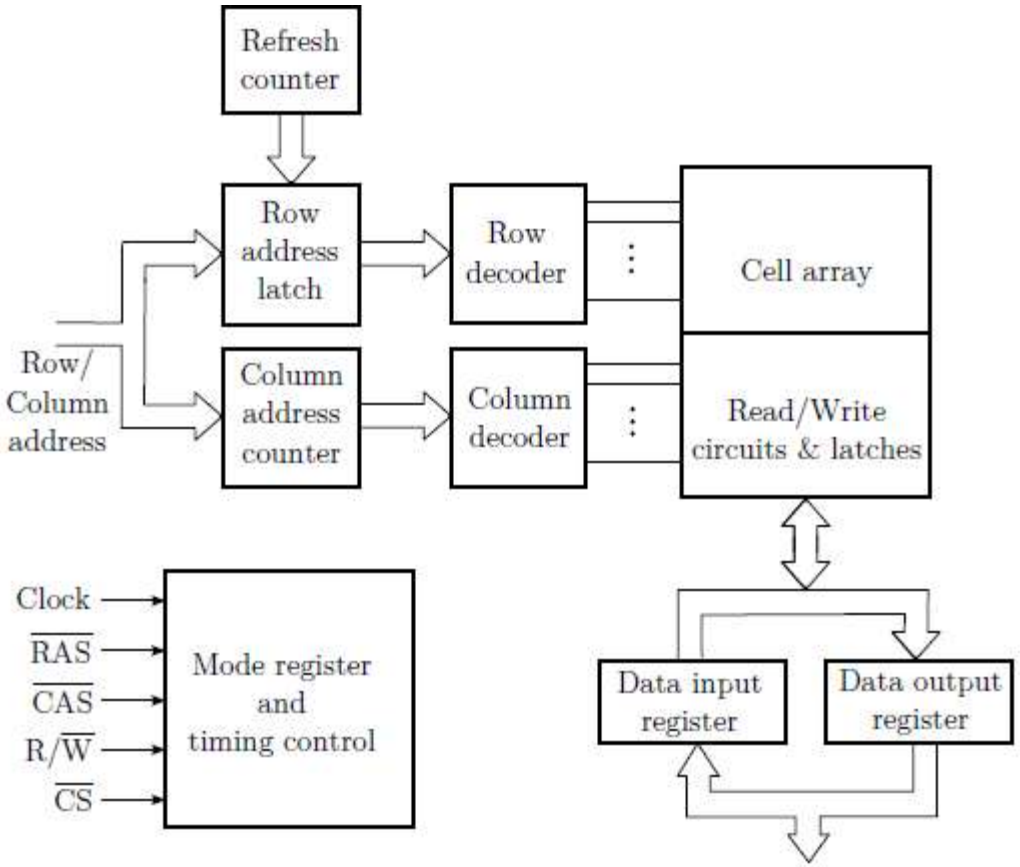
- Legacy SRAM and DRAM do not have a CLK signal.
- SDRAMs operate with a clock. A Read operation causes the contents of all cells in the selected row to be loaded into the latches. (Fig.14)

The row address is latched under the control of RAS signal. The memory activates the selected row after 2 or 3 clock cycles. Then, the column address is latched using CAS signal.

Data transfer happens only once in a clock cycle.

Latency: the amount of time taken to transfer the first word of a block of data to or from the memory

Fig. 14 Structure of SDRAM



SDRAM: Improvement & Recent Trends

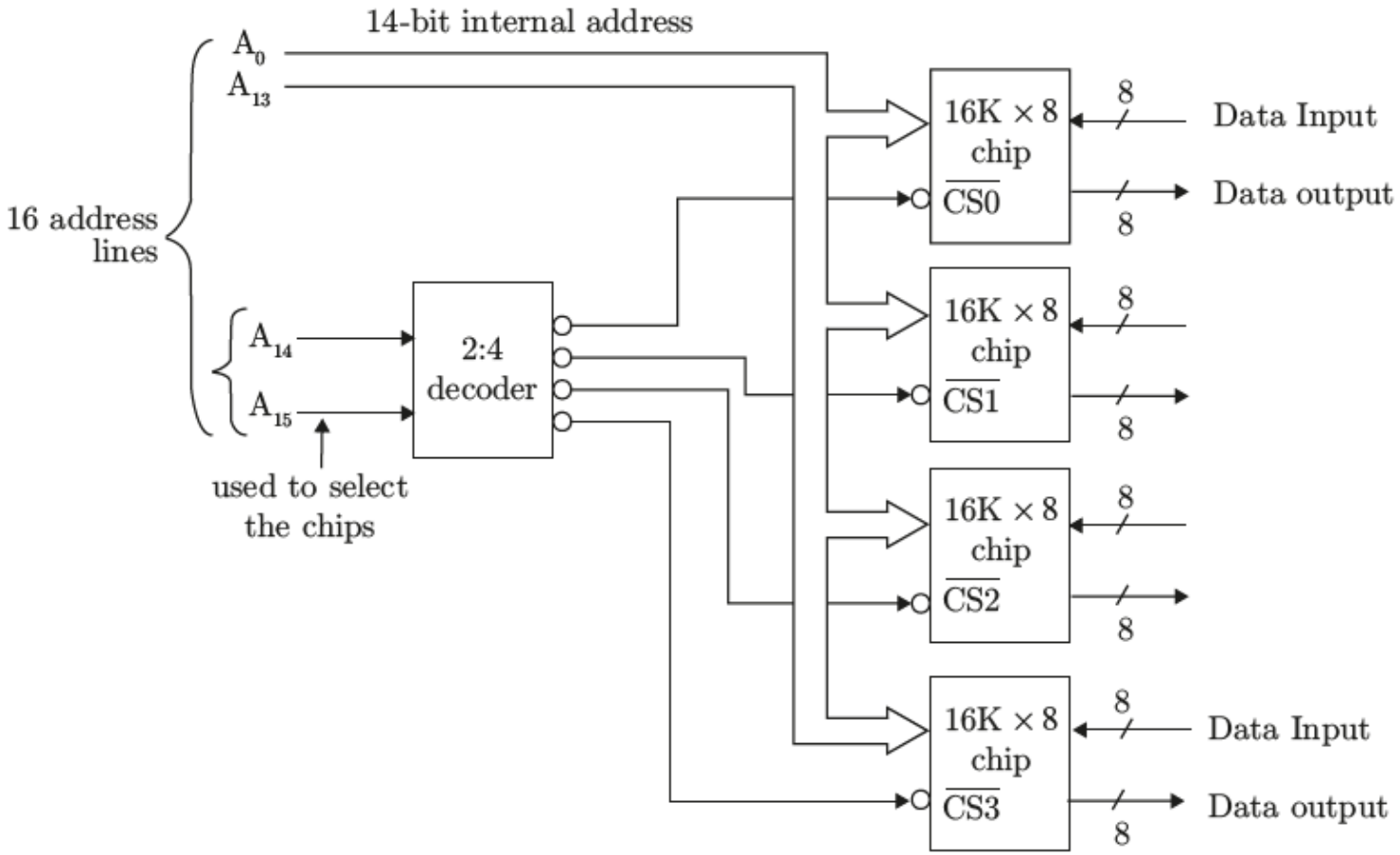
- DDRSDRAM: (Double Data Rate SDRAM): Transfer data on both dock edges. So, bandwidth and speed is doubled.
- DDR2 SDRAM: DDR2 allows higher bus speed and requires lower power by running the internal clock at half the speed of the data bus. (four data transfers per internal clock cycle). However, DDR2 DIMMs are neither forward compatible with DDR3 nor backward compatible with DDR.
- DDR3 SDRAM:DDR3 memory uses 30% less power. Another benefit is its **prefetch buffer**, which is 8-burst-deep. DDR3 DIMMs are electrically incompatible with DDR2. [notch positions and shape are different]
- DDR4 SDRAM: Very low power consumption & high clock rates.

Parameter Type	Data Transfer Rate	Supply Voltage	I/O Clock Frequency
DDR2	400–1066 MT/s	1.8V or 2.5V	200–533 MHz
DDR3	800–2133 MT/s	1.2–1.65V	400–1066 MHz
DDR4	2133–4266 MT/s	1.05–1.2V	> 1000MHz

Memory Expansion

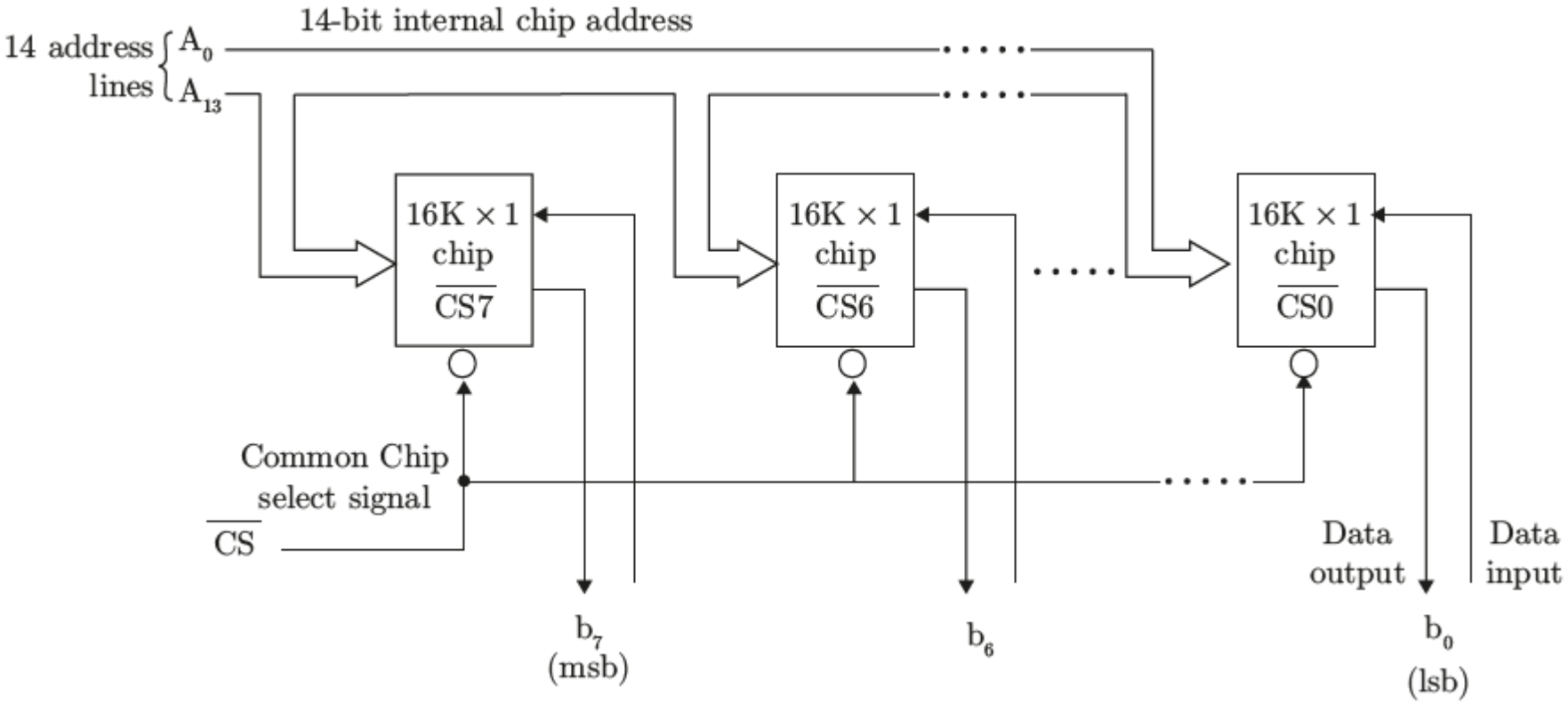
- Two types: Capacity Expansion and Word length expansion.

Fig.15 Memory Capacity expansion Example: 64Kx8 RAM using four 16Kx8 chips



Memory Expansion Contd…

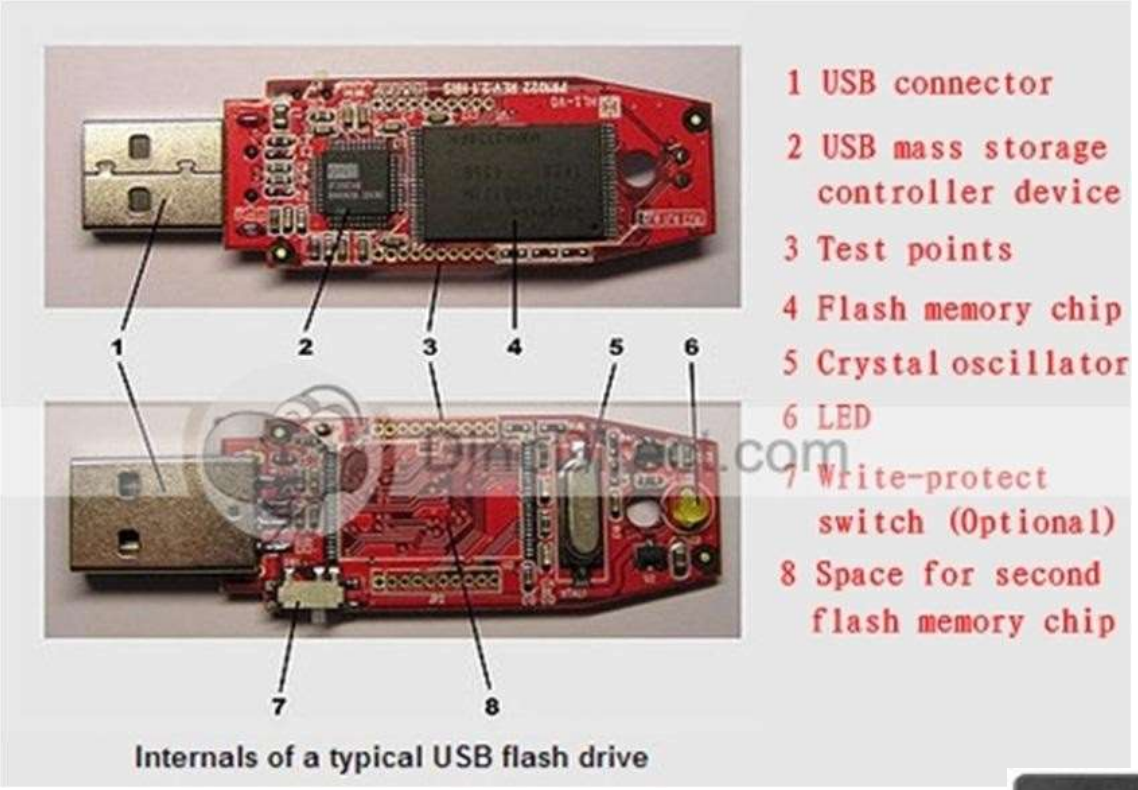
Fig.16 Word-length expansion Example: Paralleling Eight 16Kx1 chips to get 16Kx8 memory



Read-Only Memory (ROM)

- ROMs are used to store information that do not change during the normal operation of a system. Ex: Storing "Boot programs" of a computer, Mathematical tables etc
- ROMs are non-volatile; contents are not lost when electrical power is switched off. So, useful in many applications like Security systems, Electronic Appliances, Microcontroller-based Embedded systems, etc.
- The process of entering data into ROM is known as "burning" or "programming" the ROM. Done during fabrication. User can only read the information.
- ROM constructed with "fusing" gates/switches. (Fig.17) A logic 0 is stored if the transistor (T) is connected to ground at point P; otherwise, logic 1 value is stored.
- Variants: Different types of ROMs are: Mask-Programmed ROM (MROM), Programmable ROM (PROM), Erasable-PROM (EPROM), Electrically Erasable-PROM (EEPROM), CD-ROM, Flash, NVRAM etc
- **Flash Memory**: Widely used in all electronic gadgets like PDAs, Cell phones, Digital Camera, Gaming console, Audio/Video players, Tablets, Ereaders, etc., Advantages are: Low power consumption, Low cost, high capacity (Giga bytes of storage) portability, small size, light-weight, and available in many forms as Flash cell, Flash card, Flash drive etc.,
- Flash cards commonly known as SD (Secure Digital) cards available with different types and speeds like: SD, SDHC, SDXC, miniSD, microSD, memory stick, USB drive etc.

Flash devices Example



Internals of a typical USB flash drive

Memory Hierarchy

- An ideal memory must be fast, large and inexpensive.
- However, a single type of memory can't meet all the requirements.
- A computing system employs many different types of memory as shown in Fig.17.
- As seen, Disk memory has the highest capacity but its slower compared to primary memory.
- Earlier systems had only primary and secondary memory. To improve the system performance, another type of fast but small size memory known as "cache" was designed.
- "Cache" can be on-chip (L1 cache) as well as off-chip (L2, L3 cache).

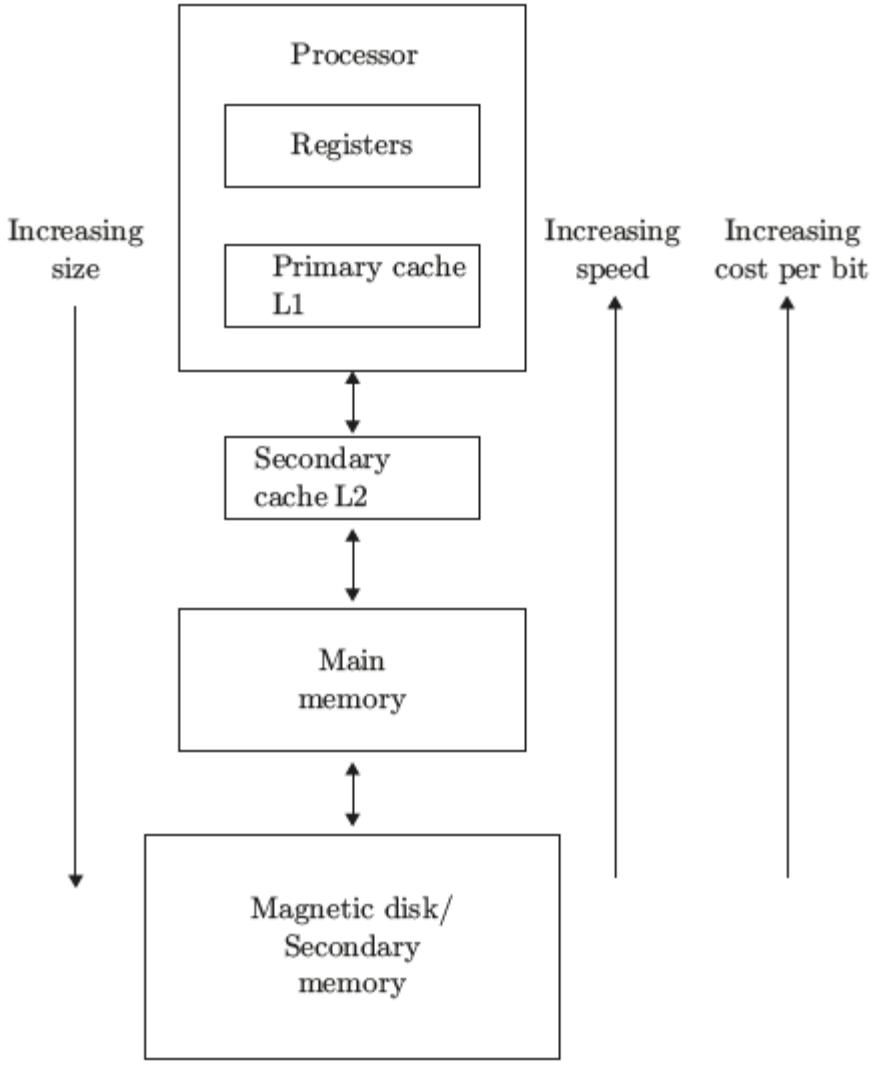


Fig.17 Memory Hierarchy

Cache Memory

- Processor Speed at least 10 times that of Primary memory. So, construction of smaller size but faster "Cache" memory.
- L1 (on-chip), L2 (On/Off-chip) and L3 (Off-chip) cache hierarchy.
- Operation based on principle of "**Locality of Reference".** Helps to reduce the main memory access every time and to speed up the execution.
- TEMPORAL LOCALITY: A memory location that is **referenced once** is likely to be **referenced multiple times** in near future. i.e., a recently executed instructions is more likely to be executed again.
- SPATIAL LOCALITY: If a memory location referenced, **more possibility that nearby memory locations are also accessed**. i.e., an instruction close by a recently executed instruction is likely to be executed soon.

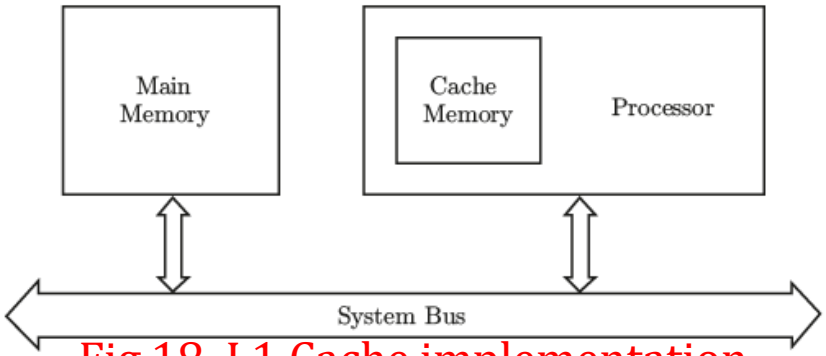


Fig.18 L1 Cache implementation

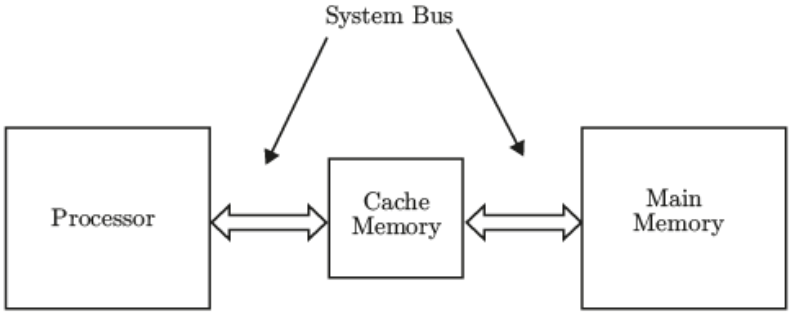


Fig.19 L2/L3 Cache implementation

Cache Memory: Terminologies

- The smallest unit of data that can be exchanged between main memory and cache is called "cache line". Collection of two or more lines is called a "set".
- Cache Write Operation: a) Write-Through method: the cache location and the main memory location are updated simultaneously. b) Write-back (copy- back) method: Update cache location first and main memory later only when the data block in the cache is to be replaced. However, 'cache coherence' problem exists in this case.
- During a write operation, a 'write-miss' occurs if the addressed location does not refer to cache memory.
- During a Read operation, if the referenced word is not present in the cache, a '*read miss*' occurs.
- Main memory is large compared to the cache. So, all data cannot be copied to cache at once. So, need a method for mapping the blocks of data from main memory into the cache.
- The correspondence between main memory blocks and cache blocks is specified by a **mapping function**
- When there are no available slots in the cache to place a block brought from main memory, a '**replacement policy**' is implemented

Cache Memory: Terminologies contd ..

- **Cache Lines / Blocks**: The smallest unit of memory that can be transferred between the main memory and the cache. The term 'block' is used to refer to a set of contiguous address locations of some size.
- **Set:** Collection of one or more cache lines.
- **Valid bit**: of a particular cache block is set to 1 the first time this block is loaded from Main memory.
- **Dirty-bit**: indicates whether the block in the cache is accessed and modified. A block that is modified must be written back to the main memory before its is reused to store another data block.
- **Cache Hit:** The referenced word is available in Cache.
- **Cache Miss**: The referenced word is not available in Cache
- **Hit ratio/Rate:** Ratio of Number of times the data is present in cache to the total number of cache references (value 0 to 1)
- **Miss Penalty:** When a cache miss occurs, the extra time needed to bring the required information from main memory.
- **Cache Coherence:** Copies of shared data have the same value at all times.

Cache Operation

Where cache can be used?

```
for (i=0; i<M; i++)
for(j=0; j<N; j++)
Z[i][j] = X[i][j] +
Y[i[[j];
```

Loop is executed (M*N) times

» Placing the code in cache avoids access to main memory

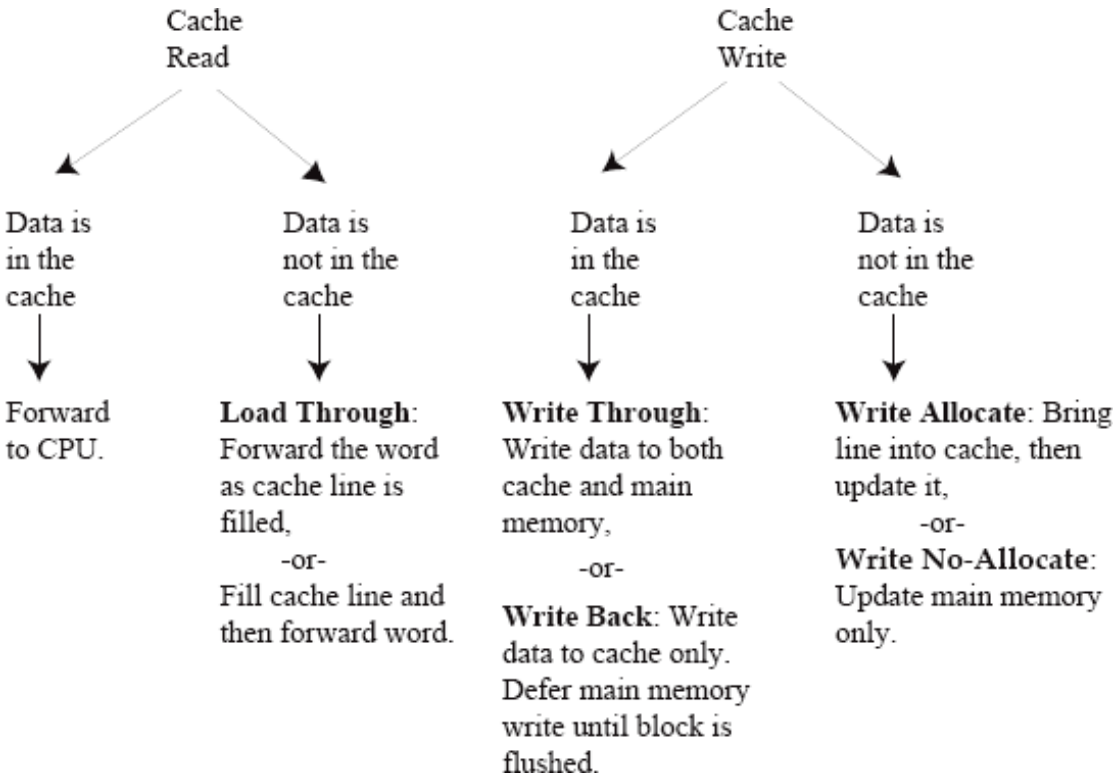
– Repetitive use

– Temporal locality

» Prefetching data

– Spatial locality

– When a program executes, the CPU looks into cache memory first.



Cache Mapping - Direct Mapping

There are three commonly used methods to translate main memory addresses to cache memory addresses. Direct-Mapped, Associate Mapped and Set-Associative Mapped Cache

- **Direct Mapping:** Simple method. A 'block' in main memory is "mapped" to a specific block in Cache. i.e., block j of the main memory is mapped onto block **j modulo N** of the cache, where 'N' is the total blocks in the cache.
- Ex: Consider a Cache with 128 (2^7) blocks of 16 words each (size 2K) and Main memory with 4096 blocks of 16 words each (size 64K). So, a total of 64K/2K = 32 main memory blocks can be mapped onto each Cache block (slot).
- Ex: Blocks 0, 128, 256 etc of main memory mapped to "block 0" of cache memory., Blocks 1, 129, 257 etc of main memory mapped to "block 1" of cache memory, and so on.
- Main memory address divided into 3 fields namely: Tag, Block and Word. The low order **4 bits** select one of 16 words in a block. When a new block enters the cache, the **7-bit Block** field identifies in which of the 2^7 slots, this block need to be stored. The high-order **5-bit Tag** field identifies which of the 32 blocks of main memory mapped into cache, is currently resident in Cache.
- The high-order 5 bits of the address are compared with the tag bits associated with that cache location. If they match, the desired word is in that cache block. If no match, then the block containing the required word must be read from main memory and loaded into cache.

Direct Mapping contd..

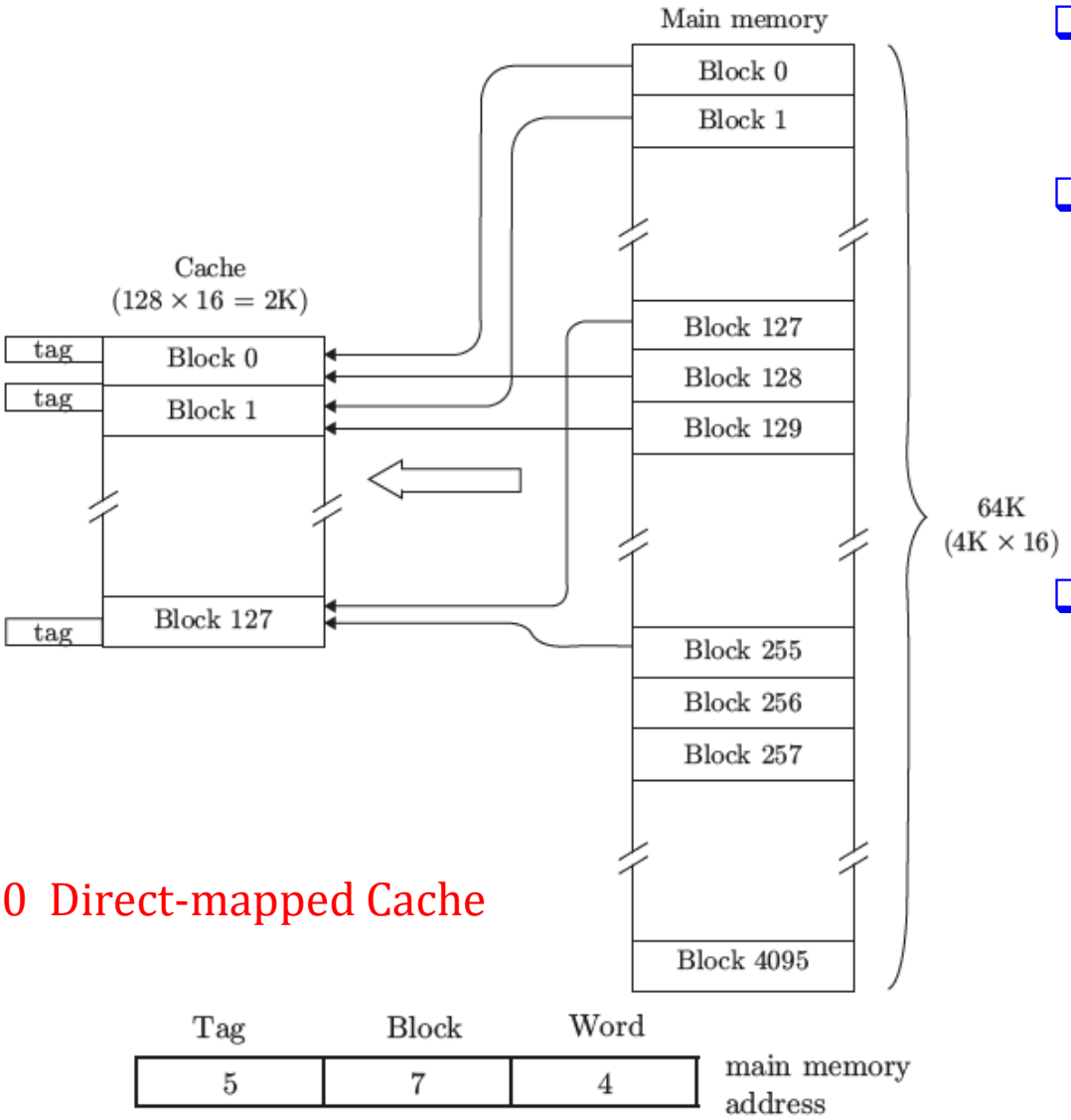


Fig.20 Direct-mapped Cache

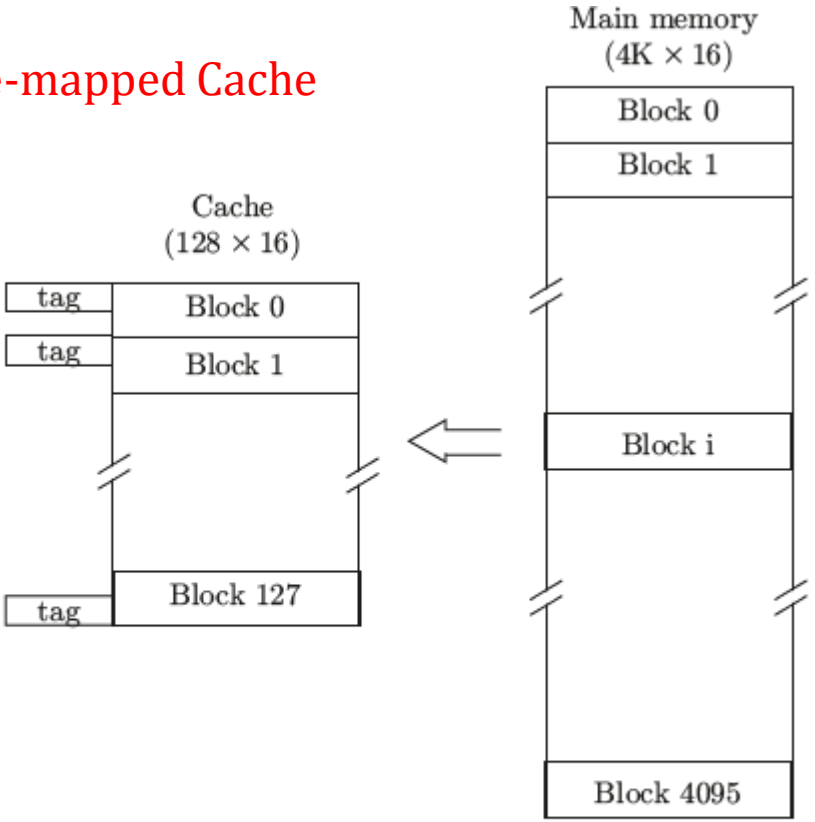
- Direct mapping is simple and easy to implement
- However, no flexibility. i.e., a main memory block cannot be put into a cache block which is not designated for it, even though its available.
- When a new main memory block is to be loaded & if cache is full, new block will overwrite the currently resident block. No replacement algorithm is needed

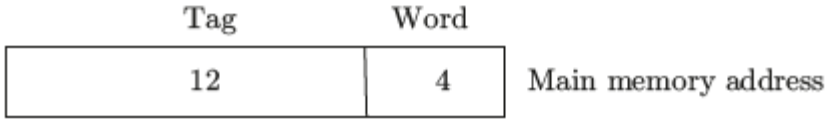
Cache Mapping – Associative Mapping

- More flexible – each main memory block is fully *associated* with any free cache block. i.e., A main memory block can be put into any "free" block in cache
- Ex: Consider a Cache with 128 (2^7) blocks of 16 words each (size 2K) and Main memory with 4096 blocks of 16 words each (size 64K). So, a total of 64K/2K = 32 main memory blocks can be mapped onto each Cache block (slot)
- No need to identify a specific "empty" cache block as in Direct mapping. Each main memory block is 'associated' with any free cache block.
- Main memory address divided into 2 fields namely: Tag and Word. The low order **4 bits** select one of 16 words in a block. 12-bit Tag field to identify a memory block when it is resident in the cache.
- Cache space used more efficiently, No contention, easy implementation
- When cache is full, to bring in a new block, an existing block has to be replaced. So, a 'replacement algorithm' is needed.
- Considerable hardware overhead. Need to search all 128 tag patterns to find whether a given memory block exists in the cache.
- Search time & cost involved is high.

Associate Mapping contd..

Fig.21 Associative-mapped Cache



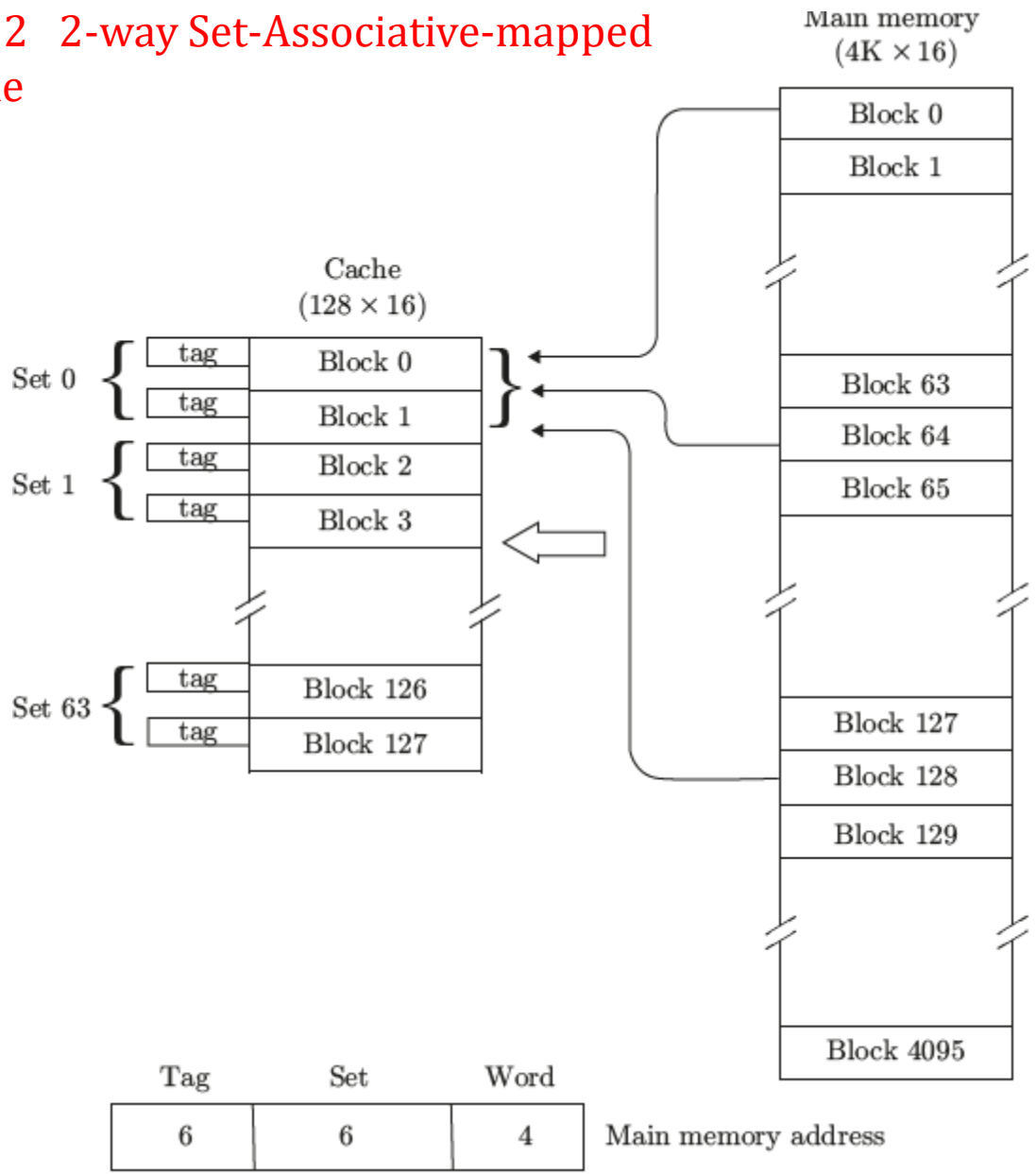


Cache Mapping – Set Associative Mapping

- Combines the simplicity of Direct mapping with the flexibility of Associative mapping
- Here, blocks of the cache are grouped into **sets**, and the mapping allows a block of main memory to reside in any block of a specific set in the cache.
- When an address is mapped to a set, the direct mapping scheme is used, and then associative mapping is used within a set.
- Ex: Consider a Cache with 128 (2^7) blocks of 16 words each (size 2K) and Main memory with 4096 blocks of 16 words each (size 64K). So, a total of 64K/2K = 32 main memory blocks can be mapped onto each Cache block
- Ex: 2-way Set-associative mapping. i.e., The blocks in cache are divided into 64 sets and there are two blocks in each set. Main memory blocks 0, 64, 128, 4032 maps to set 0 and can occupy either of the two positions.
- Main memory address divided into 3 fields namely: Tag, Set and Word. The low order **4 bits** select one of 16 words in a block. 6-bit Set field identifies one of 64 sets of the cache. The 'tag' field of the address is then compared to the tags of the two blocks of the set to check if the desired block is present.
- A set could have one block => direct mapping ; 128 blocks => Associative mapping
- 'k' blocks per set is referred to as k-way set-associative mapping
- Compared to Associative mapping, search size & time reduced. The bigger the "set" is, the better the performance, but complex and expensive.

Fig.22 2-way Set-Associative-mapped Cache

contd..



Cache Performance

- Given:
  - h = Hit ratio
  - T_a = Average effective memory access time by CPU
  - T_c = Cache access time
  - T_m = Main memory access time after cache miss

  Effective memory time is: $\mathbf{T_a = hT_c + (1-h)T_m}$
- Example:

Assume main memory access time of 100ns, cache access time of 10ns and hit ratio of 0.9. Then,

$T_a = 0.9(10ns) + (1 - 0.9)(100ns) = 19ns$

Speed up due to Cache is: $S_c = 100ns / 19ns = 5.2$

If hit ratio is now 0.95, then:

$T_a = 0.95(10ns) + (1 - 0.95)(100ns) = 14.5ns$

If LI and L2 caches are used, the average access time is:

$$\mathbf{T_{ave} = h_1 Tc_1 + (1-h_1) h_2 Tc_2 + (1-h_1)(1-h_2) T_M}$$

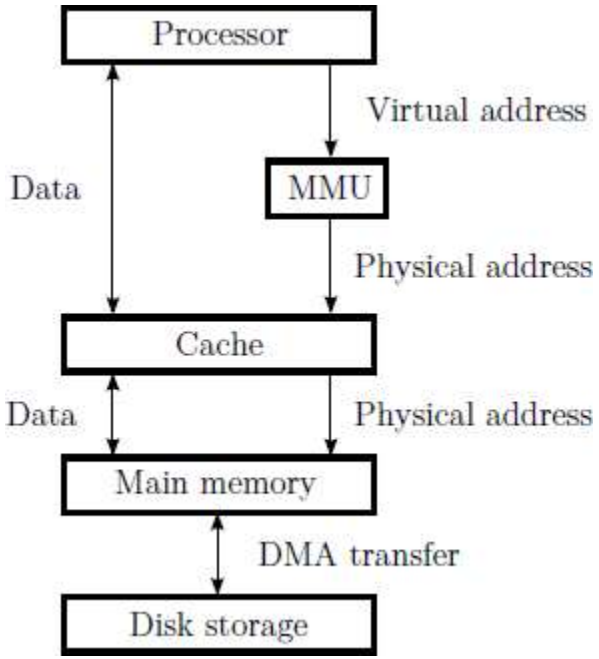
Where, h_1 – hit rate for L1 cache, h_2 - hit rate for L2 cache, Tc_1 – Time to access L1 cache, Tc_2 - Time to access L2 cache

Virtual Memory (VM)

- **Virtual memory** – separation of user's logical memory from physical memory
- Only part of the program needs to be in memory for execution. Physical memory constraint need not limit the number of applications running.
- Logical address space can therefore be much larger than physical address space.
- Allows address spaces to be shared by several processes.
- A special hardware unit called Memory Management Unit (MMU), translates the virtual addresses into physical addresses. If the data are not in the main memory, the MMU causes the operating system to bring the data into main memory from the disk.
- The binary addresses that the processor issues are called as **logical or virtual address.** The logical address is translated into physical address using hardware and/or software approach

❑The goal of cache is to improve speed.
❑The goal of VM is to increase memory space required for user applications

Fig. 23 Concept of Virtual Memory



Virtual Memory with Paging

- Virtual memory can be implemented via:
- ✦ Demand Paging
- ✦ Demand Segmentation
  - Physical memory is divided into chunks of fixed size called "page frames". Virtual memory is divided into chunks called "pages". The size of a page is same as that of a page frame.
  - The Operating System keeps track of mapping pages to page frames.

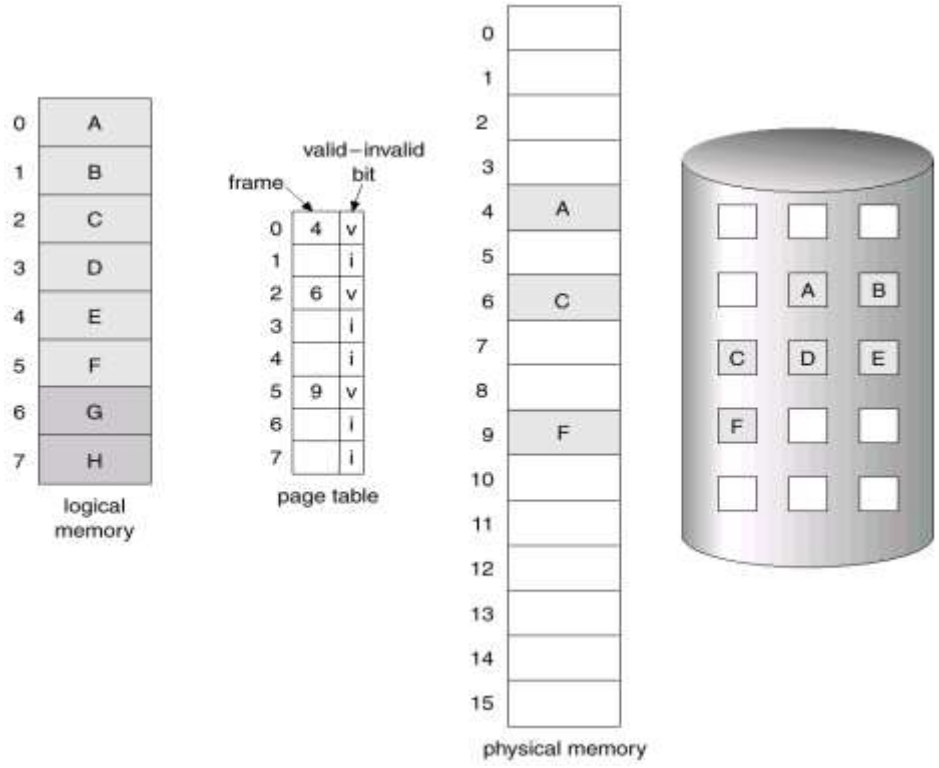
- Keep track of all free frames.
- To run a program of size *n* pages, need to find *n* free frames and load program.
- Set up a **Page Table (PT)** to translate logical to physical addresses.

•The PT contains info. like which page is free in main memory, 'valid' bit , 'dirty bit' etc,.

•**Valid bit** indicates whether the referred page is actually loaded in main memory.

•modified or **Dirty bit** shows whether the page has been modified/accessed when it is residing in main memory

Fig. 24 Concept of Paging

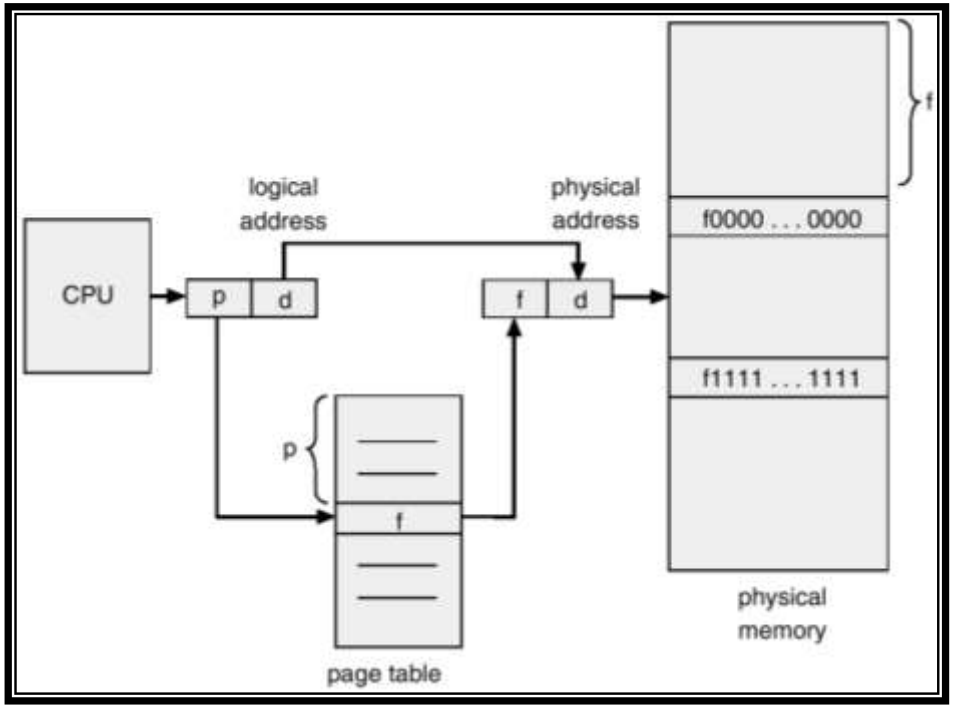


VM with Paging Contd..

- Fig.25 depicts Address Translation using Paging
- If the page table entries are less, then it can be placed inside the MMU and the MMU can use the information in the page table for every read and write access
- If entries are more, PT is stored in main memory and a copy of PT with latest entries is stored in a small cache known as "translation look aside buffer" (TLB)

Fig.25 Address translation using Paging

- If 'valid' bit = 0, then the page is not in main memory indicating a "page fault" error. In that case, the page has to be brought from secondary memory.
- **Demand Paging:** Bring a page into memory only when it is needed.
  - Less I/O needed
  - Less memory needed
  - Faster response



VM with Paging – Page Table and TLB

- Fig.22 shows the operation of VM with TLB.
- For a given virtual address, the MMU looks in the TLB for the referenced page. If the entry for this page is found, (TLB hit) the physical address is obtained immediately.
- If there is a TLB miss, then the entry is obtained from the page table in main memory and the TLB is updated

Translation Lookaside Buffers (TLB)

- Problem: page tables are stored in main memory
- Access to main memory is slow compared to clock cycle on CPU (factor of about 10 -- 10ns vs 1 ns)
  - An instruction such as **MOVE REG, ADDR** has to decode **ADDR** and thus go through page tables
  - This would make things very, very slow
- Standard solution: TLB in MMU
  - TLB stores small cache (say, 64) of page table entries to avoid the usual page-table lookup
  - TLB is *associative memory (content addressable memory)* that contains, basically, pairs of the form (page-no, page-frame)
  - Special hardware compares incoming page number in parallel with all entries in TLB to retrieve page-frame
  - If no match found in TLB, we do standard lookup via main memory
- TLB is like a cache for MMU

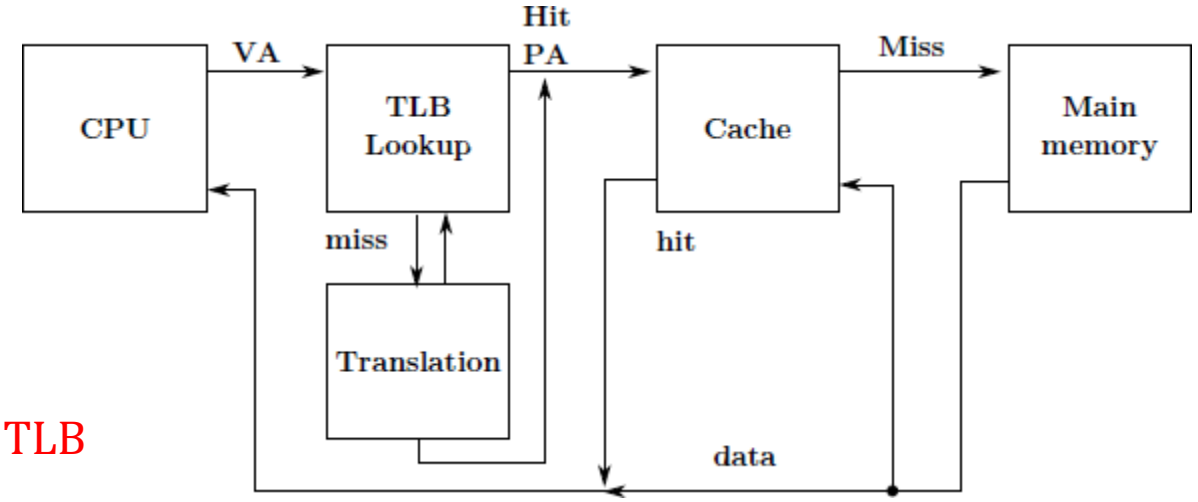


Fig.26 VM with TLB

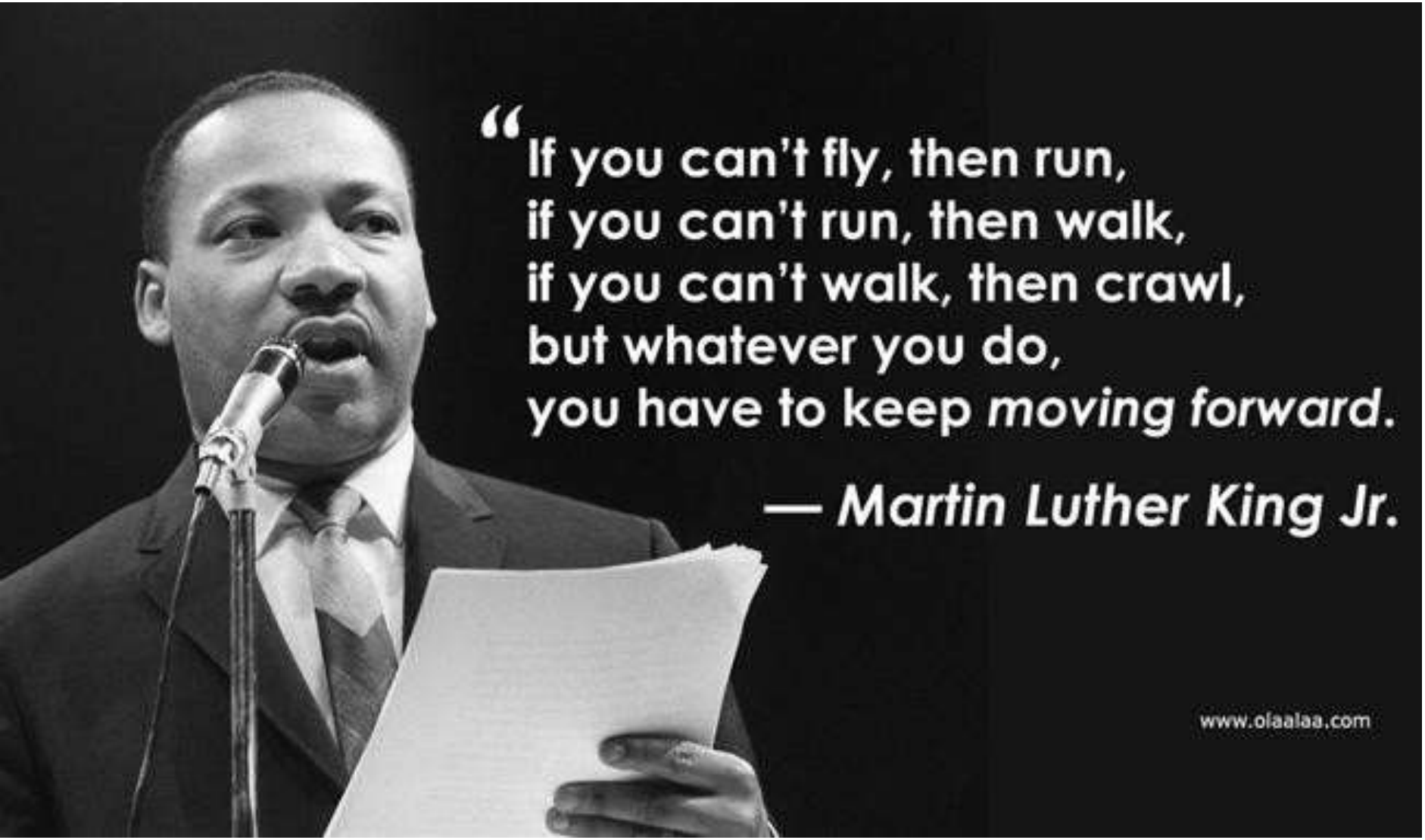
Conclusion

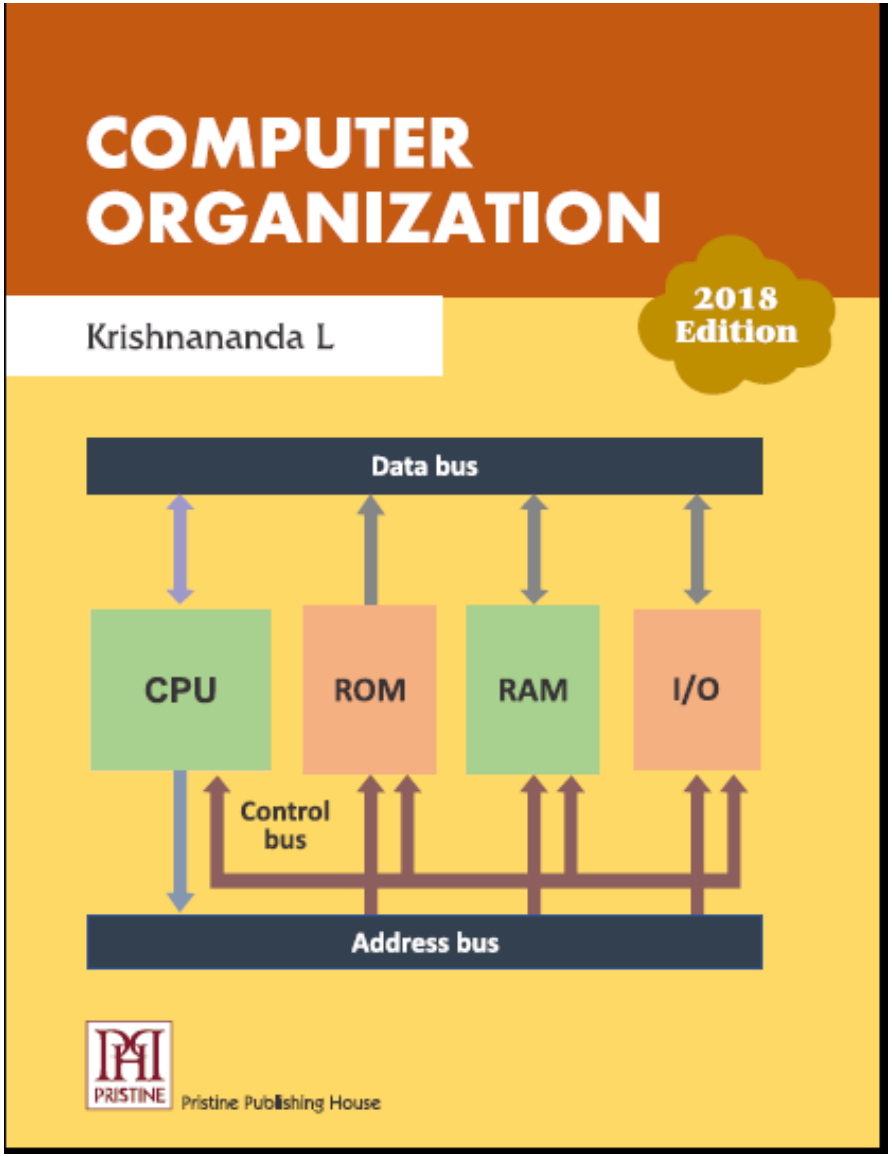
- Memory design is one of the most challenging issue in computing system. The goal is: low-cost, high density, small size chip with low-power consumption
- The memory system is classified as: Primary and Secondary
- Primary memory is faster but less capacity. Classified as: Read/Write memory and Read-only memory.
- Read/Write memory types are: SRAM, DRAM, SDRAM, DDRRAM, etc.,
- Read-only memory types are: ROM, PROM, EPROM, EEPROM, Flash etc.,
- Special memory: Cache memory - L1 cache, L2/L3 cache. Used to bridge the speed gap between CPU and main memory. Operates on the principle of "locality of reference".
- Cache mapping techniques: How to transfer data blocks between main memory and Cache? Three methods: Direct, Associative and Set-Associative
- Replacement methods: If cache is full and a new block is to be brought into the cache, one of the cache blocks is to be replaced. To find the 'victim' block, use "replacement algorithms". Ex: FIFO, LRU, Optimal etc
- Virtual Memory: Concept used to bridge the gap between main memory and secondary memory allowing the user to write programs without looking into memory size constraint..

References

The contents for this presentation are selected from the following books and web resources available. I take this opportunity to gratefully acknowledge the sources.

- **Carl Hamacher**, Zvonko Vranesic, Safwat Zaky, " *Computer Organization*", 5th ed, McGraw Hill Education
- **David A. Patterson**, John L. Hennessy, *Computer Organization and Design: The Hardware/Software Interface-ARM edition, 4/e,* Morgan Kaufmann Publishers/Elsevier
- **Miles J. Murdocca** and Vincent P. Heuring, *Computer Architecture and Organization: An Integrated Approach,* JohnWiley & Sons
- **Vincent P. Heuring** andHarry F.Jordan, *Computer Systems Design and Architecture,* 2/e, Pearson Education
- Web links:
- *www. cs.uwec. edu (Univeristy of Wisconsin- Eau Clarie)*
- *www. cs.siu. edu (Southern Illinois University)*
- *www. cs. ucla. edu (University* of California, Los Angeles)


“If you can’t fly, then run,
if you can’t run, then walk,
if you can’t walk, then crawl,
but whatever you do,
you have to keep moving forward.
— Martin Luther King Jr.
www.olaalaa.com



www.pristinebooksonline.com

Any Questions????