CO-UNIT V-MEMORY SYSTEM

dt 27.2.18

updated on dt.6/3/18

Text/Reference Books

The following sources are used for preparing these slides

1. **Computer Organization ,** Carl Hamacher, Zvonko Vranesic, Safwat Zaky McGraw Hill Publ.

2. **Computer Organization and Architecture: Designing for Performance,** William Stallings , Prentice-Hall India,Publ.

3. **Computer Architecture A Quantitative Approach** ,John L Hennessy and David Patterson , Morgan Kaufman Publ.

4. **Structured Computer Organization** ,Andrew S. Tanenbaum , Prentice-Hall India Publ.

5. **Computer Organization and Design,** P. Paul Choudhury ,Prentice-Hall India,Publ.

Websites:

- ❑ In this Unit Module, we will be studying the following Topics

i) Basic Concept of Memory

ii) Semi Conductor RAM memories

iii) Semi conductor Read-Only memories

iv) Cache Memory

Topic beyond The syllabus

Performance Considerations

v) Secondary Storage

Topics beyond The syllabus

Concept of Virtual memory

Memory Management

Introduction

- ❑ Digital computer works on stored programmed concept introduced by Von Neumann.
- ❑ Memory is used to store the information, which includes both program and data.
- ❑ Due to several reasons, we have different kind of memories. We use different kind of memory at different level.
- ❑ The memory of computer is broadly categories into two categories:

  ❍ Internal and ❍ external
- ❑ Internal memory is used by CPU to perform task and external memory is used to store bulk information, which includes large software and data.
- ❑ Memory is used to store the information in digital form. The memory hierarchy is given by:
- ❑ ❍ Register
- ❑ ❍ Cache Memory
- ❑ ❍ Main Memory ❍ Magnetic Disk
- ❑ ❍ Removable media (Magnetic tape)
- ❑

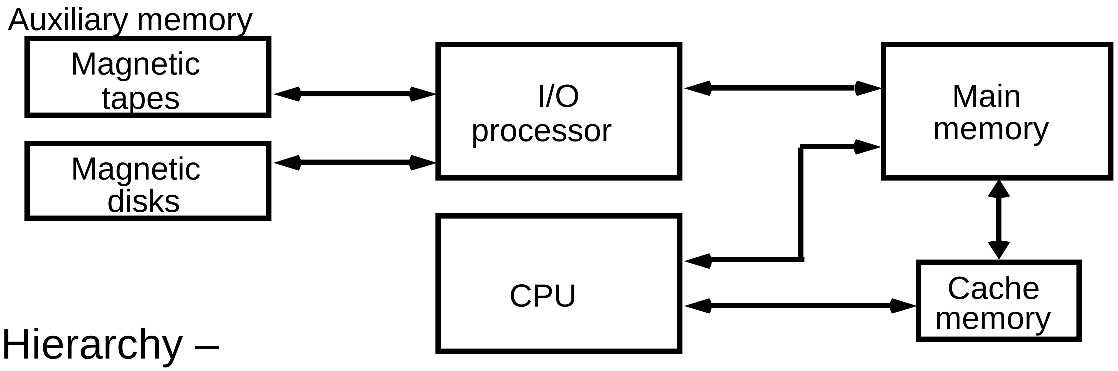
Memory Hierachy

- Programmers want unlimited amounts of memory with low latency
- Fast memory technology is more expensive per bit than slower memory
- Solution: organize memory system into a hierarchy
  - Entire addressable memory space available in largest, slowest memory
  - Incrementally smaller and faster memories, each containing a subset of the memory below it, proceed in steps up toward the processor
- Temporal and spatial locality insures that nearly all references can be found in smaller memories
  - Gives the allusion of a large, fast memory being presented to the processor

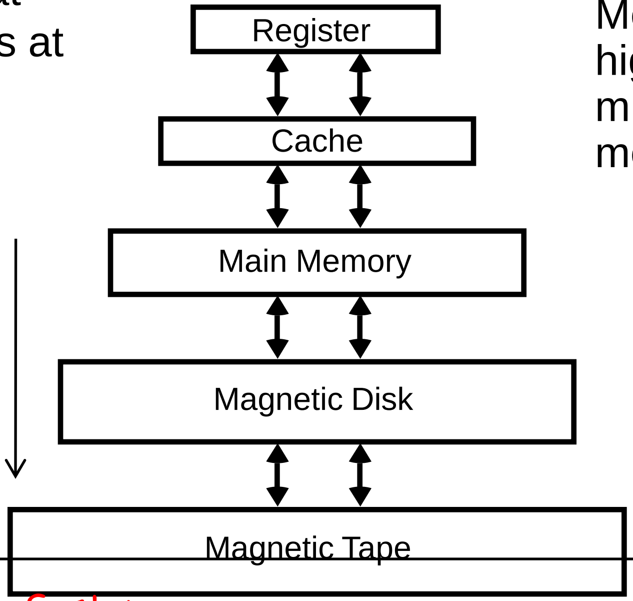
Exploiting the Memory Hierarchy

- Locality
  - Spatial Locality:
    - Data is more likely to be accessed if neighboring data is accessed.
      (e.g., data in a sequentially access array)
  - Temporal Locality:
    - Data is more likely to be accessed if it has been recently accessed.
      (e.g. code within a loop)

MEMORY HIERARCHY



Storage Hierarchy – fastest CPU registers at top, slowest tape drives at bottom



Decreasing frequency of access of the memory by the processor

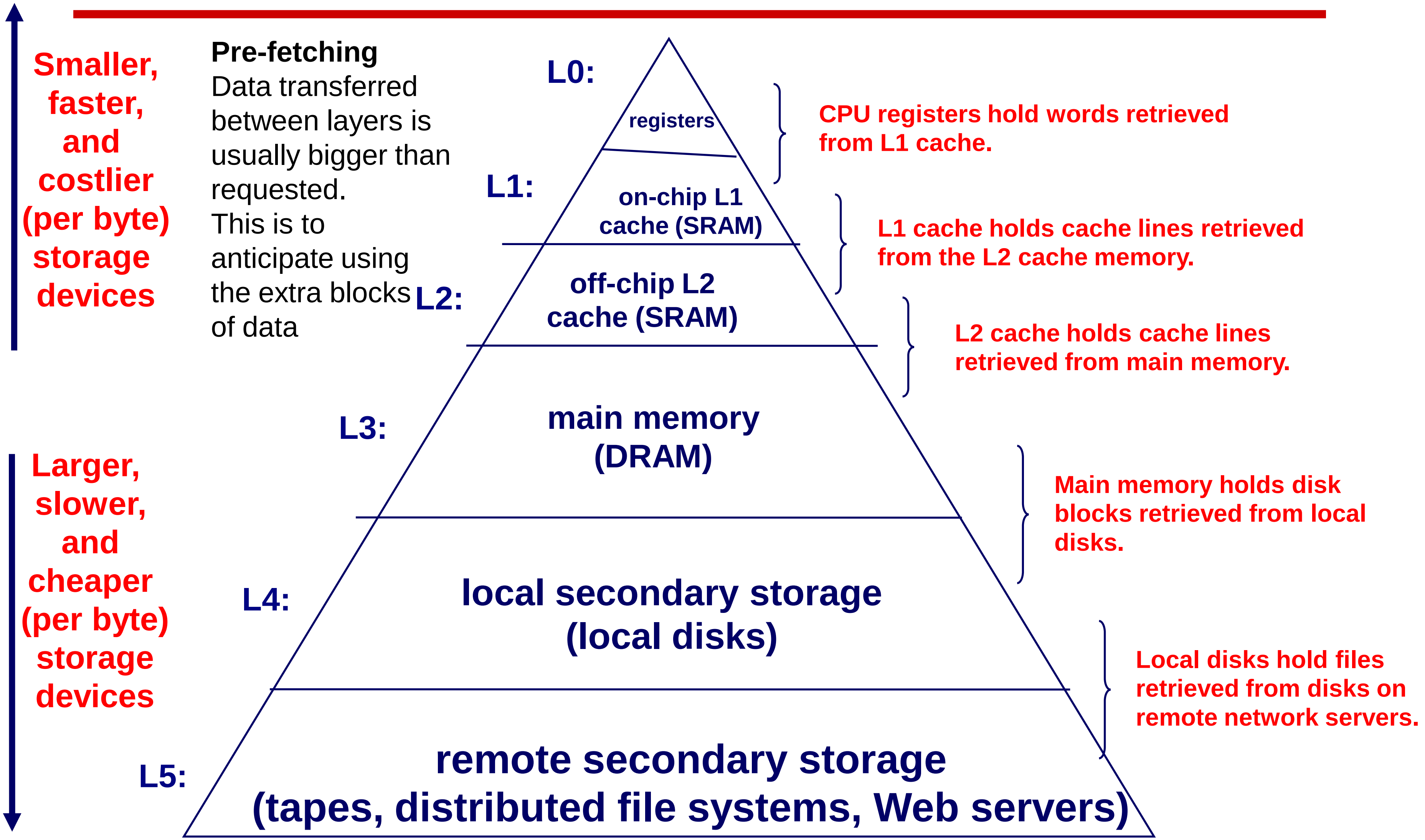
Memory Hierarchy is to obtain the highest possible access speed while minimizing the total cost of the memory system

- 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.

An Example Memory Hierarchy

Smaller, faster, and costlier (per byte) storage devices

Pre-fetching
Data transferred between layers is usually bigger than requested.
This is to anticipate using the extra blocks of data



L0: registers

L1: on-chip L1 cache (SRAM)

L2: off-chip L2 cache (SRAM)

L3: main memory (DRAM)

L4: local secondary storage (local disks)

L5: remote secondary storage (tapes, distributed file systems, Web servers)

CPU registers hold words retrieved from L1 cache.

L1 cache holds cache lines retrieved from the L2 cache memory.

L2 cache holds cache lines retrieved from main memory.

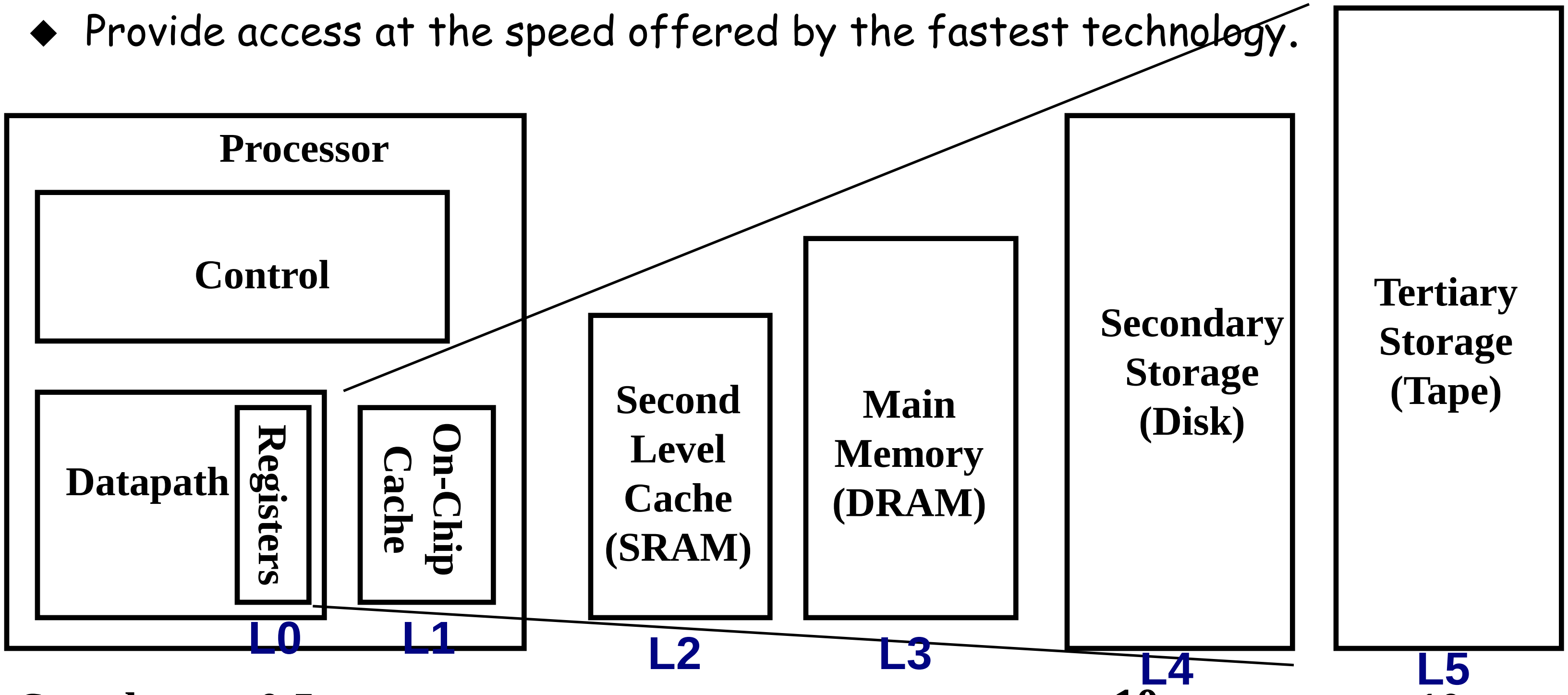
Main memory holds disk blocks retrieved from local disks.

Local disks hold files retrieved from disks on remote network servers.

Larger, slower, and cheaper (per byte) storage devices

Memory Hierarchy of a Modern Computer System

- By taking advantage of the principle of locality:
  - Present the user with as much memory as is available in the cheapest technology.
  - Provide access at the speed offered by the fastest technology.



Processor

Control

Datapath

Registers

On-Chip Cache

Second Level Cache (SRAM)

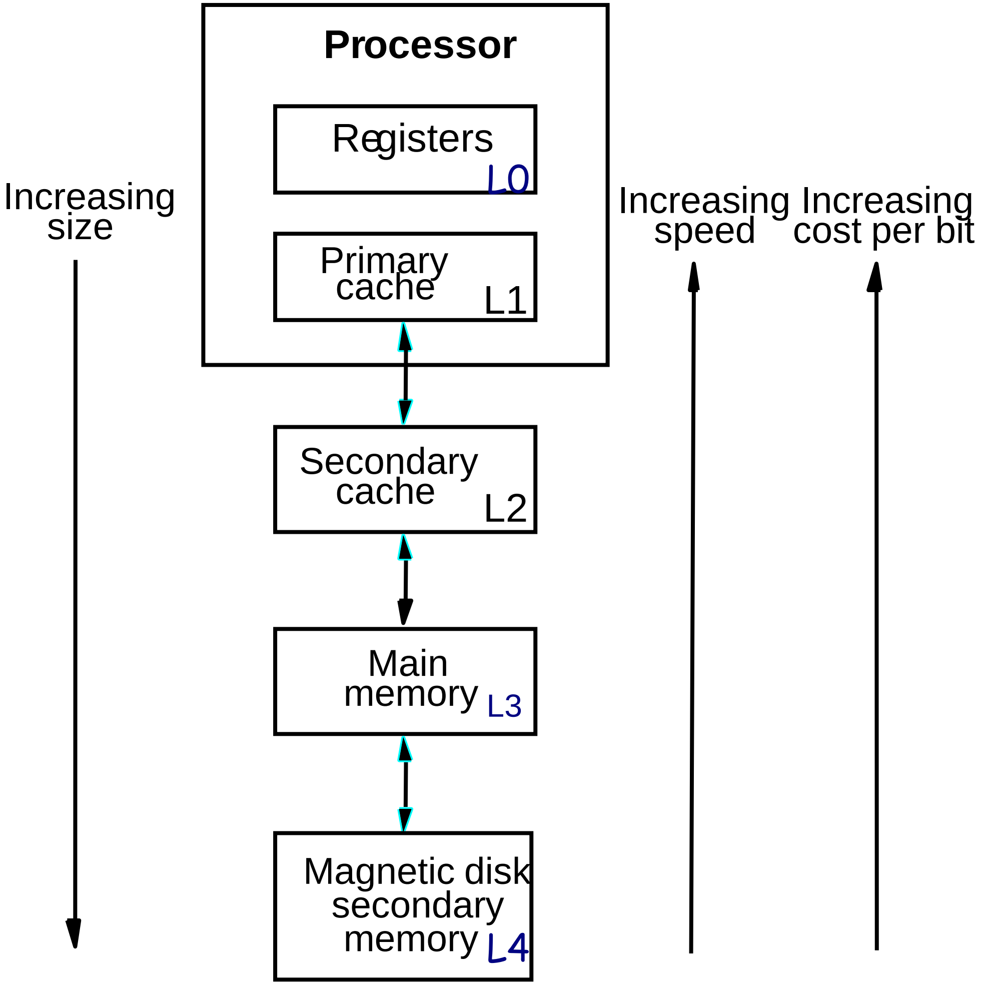
Main Memory (DRAM)

Secondary Storage (Disk)

Tertiary Storage (Tape)

	L0	L1	L2	L3	L4	L5
Speed (ns):	0.5ns	1.0ns	2.5ns	10ns	10ms	10s
Size (bytes):	256B	16KB	256KB	512MB	80GB	20TB

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.

Main Memory

- ❑ The main memory of a computer is semiconductor memory. The main memory unit is basically consists of two kinds of memory:

  RAM (RWM): Random access memory; which is volatile in nature.

  ROM: Read only memory; which is non-volatile.

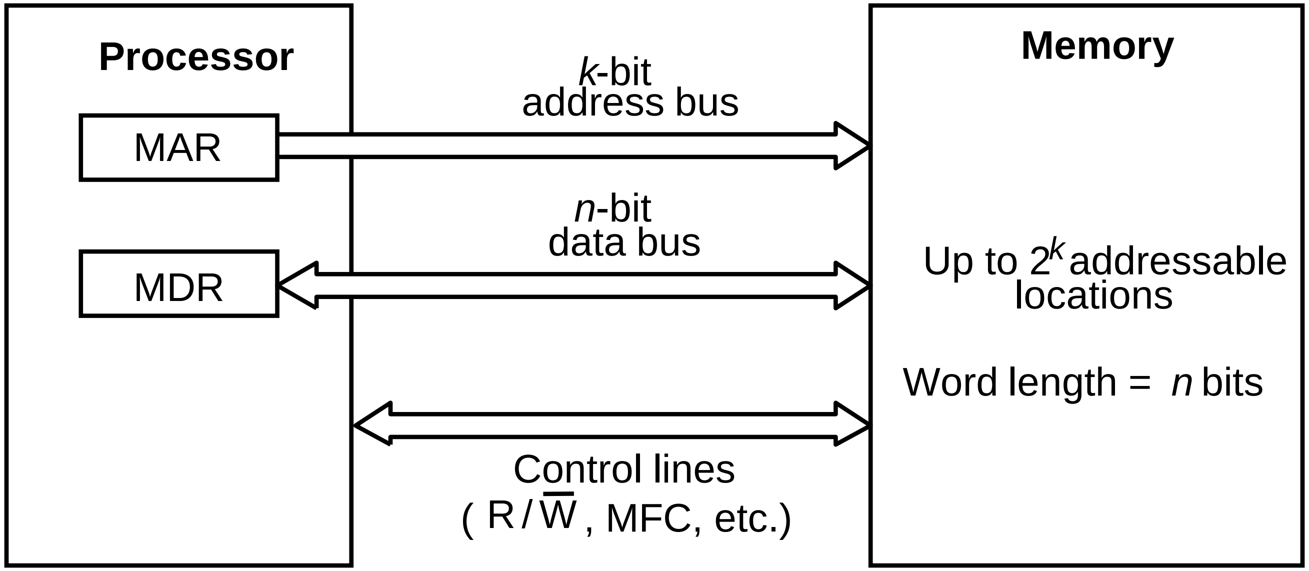
- ❑ Permanent information is kept in ROM and the user space is basically in RAM.
- ❑ The smallest unit of information is known as bit (binary digit)
- ❑ in one memory cell one bit of information can be stored.

  8 bit together is termed as a byte.

- ❑ The maximum size of main memory that can be used in any computer is determined by the addressing scheme.
- ❑ A computer that generates 16-bit address is capable of addressing upto 2^{16} which is equal to 64K memory location.
- ❑ Similarly, for 32 bit addresses, the total capacity will be 2^{32} which is equal to 4G memory location.
- ❑ In some computer, the smallest addressable unit of information is a memory word and the machine is called word-addressable .

Some basic concepts

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



Organization of the main memory

Byte-address

Word-address

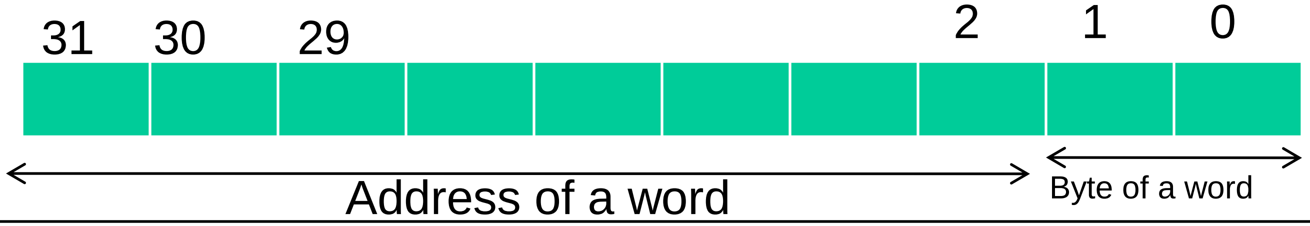
3	2	1	0
7	6	5	4
11	10	9	8
15	14	13	12

Big –endian Assignment

0	2	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Little –endian Asignment

Fig. Organization of the main memory in a 32 bit- byte addressable computer



Organization of the main memory

- In some computer, individual address is assigned for each byte of information, and it is called **byte-addressable computer**.
  In this computer, one memory word contains one or more memory bytes which can be addressed individually.
- A byte addressable 32-bit computer, each memory word contains 4 bytes. A possible way of address assignment is shown in figure.
  The address of a word is always integer multiple of 4.
- The main memory is usually designed to store and retrieve data in word length quantities.
  The word length of a computer is generally defined by the number of bits actually stored or retrieved in one main memory access.
- Consider a machine with 32 bit address bus. If the word size is 32 bit, then the high order 30 bit will specify the address of a word. If we want to access any byte of the word, then it can be specified by the lower two bit of the address bus.

Some basic concepts(Contd.,)

- 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).
- The data transfer between main memory and the CPU takes place through two CPU registers.

  MAR : Memory Address Register and MDR : Memory Data Register.
- If the MAR is k-bits, then the total addressable memory location will be 2^k.
- If the MDR is n-bits, then the n bit of data is transferred in one memory cycle.
- In the above example, size of data bus is n-bit and size of address bus is k bit.
- It also includes control lines like Read, Write and Memory Function Complete (MFC) for coordinating data transfer.
- In the case of byte addressable computer, another control line to be added to indicate the byte transfer instead of the whole word.

Memory Operation

CPU initiates a memory operation by loading the appropriate data i.e., address to MAR.

- ❑ Memory read operation,

CPU sets the read memory control line to 1.

Then the contents of the memory location is brought to MDR.

The memory control circuitry indicates this to the CPU by setting MFC to 1.

- ❑ Memory write operation

CPU places the data into MDR .

Sets the write memory control line to 1.

Once the contents of MDR are stored in specified memory location, then the memory control circuitry indicates the end of operation by setting MFC to 1.

Measures for the speed of a memory:

- ❑ Memory Access Time

  A useful measure of the speed of memory unit is the time that elapses between the initiation of an operation and the completion of the operation .

  (Ex. The time between Read and MFC)

- ❑ Memory cycle time

  This is the minimum time delay between the initiation two independent memory operations

  (Ex. two successive memory read operation).

- ❑ Memory cycle time is slightly larger than memory access time.

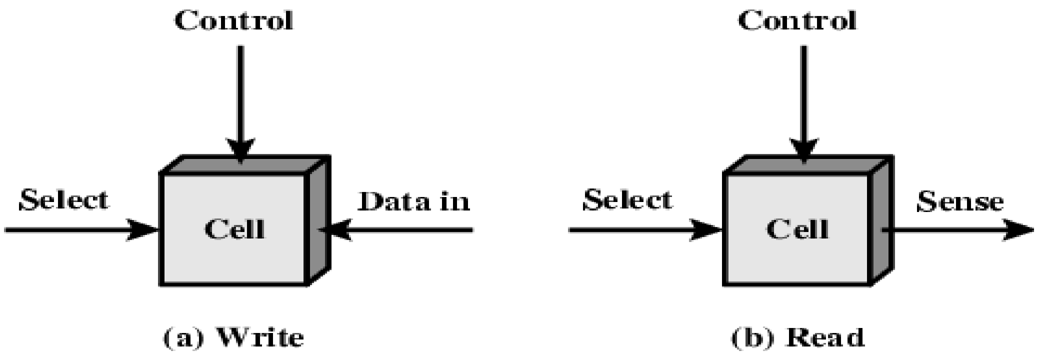
Semiconductor RAM memories

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.
- ❑ The storage part is modelled here with SR-latch, but in reality it is an electronics circuit made up of transistors.
- ❑ The memory constructed with the help of transistors is known as semiconductor memory.
- ❑ The semiconductor memories are termed as Random Access Memory(RAM), because it is possible to access any memory location in random.
- ❑ Depending on the technology used to construct a RAM, there are two types of RAM : **SRAM:** Static Random Access Memory.

  DRAM: Dynamic Random Access Memory.

Memory Cell Operation



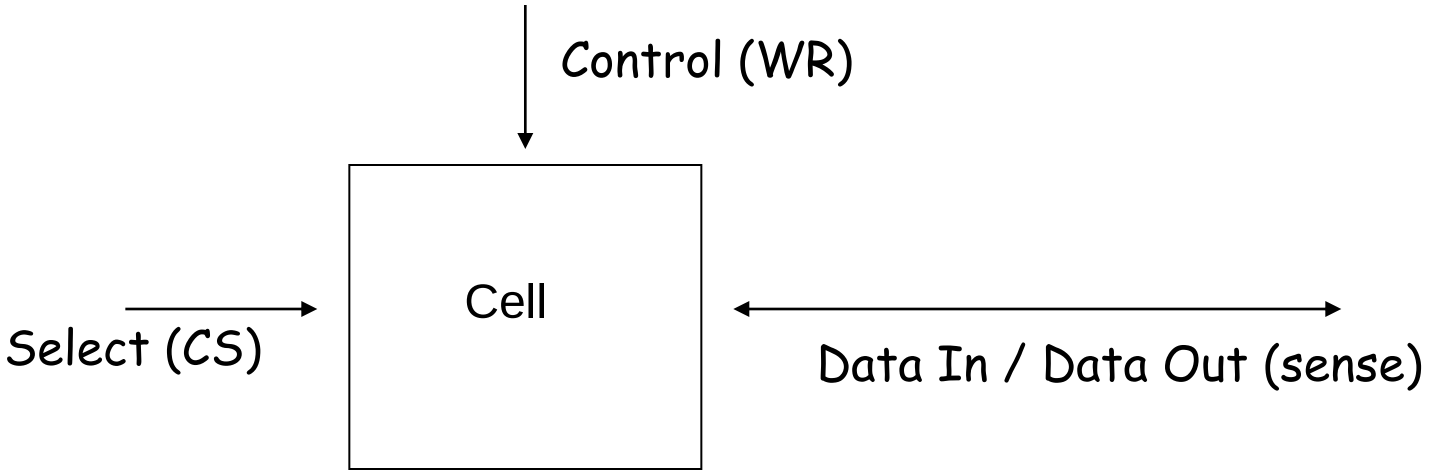
Control input to binary cell

Select	Read/Write	Memory Operation
0	x	None
1	0	Read
1	1	Write

Chip Organization

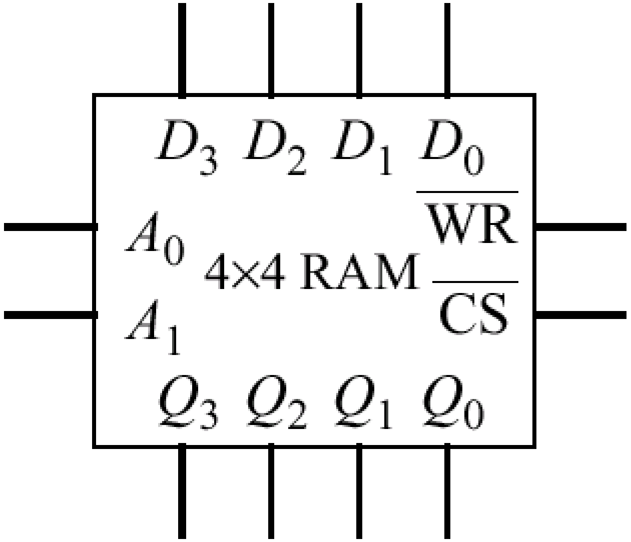
- ❑ Consider an individual memory cell. Select line indicates if active, Control line indicates read or write.

Let's say that each cell outputs 4 bits (i.e. word size=4 bits), and we would like to hook four of these together for a 4 word memory…



Simplified Representation-4 bit memory

- What one would see if this was packaged together?



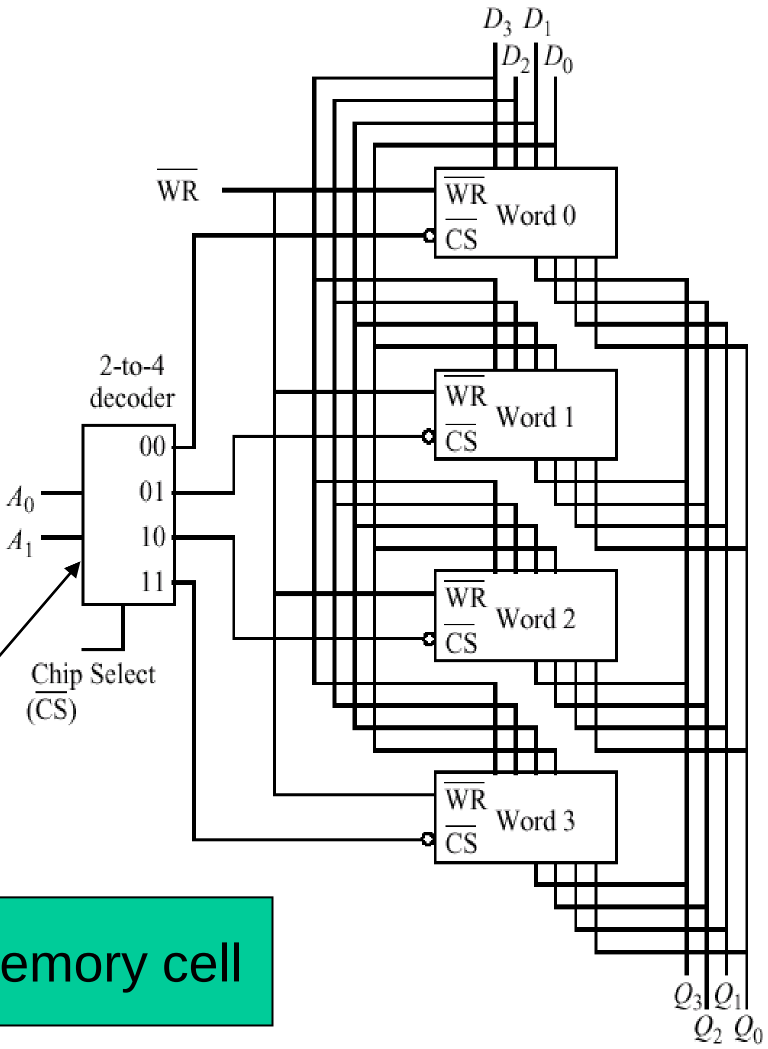
Four Word Memory, 4 bits per word

Memory addresses:

0	$A_1=0, A_0=0$
1	$A_1=0, A_0=1$
2	$A_1=1, A_0=0$
3	$A_1=1, A_0=1$

Datain : D3 D2 D1 D0

Dataout: Q_3, Q_2, Q_1, Q_0

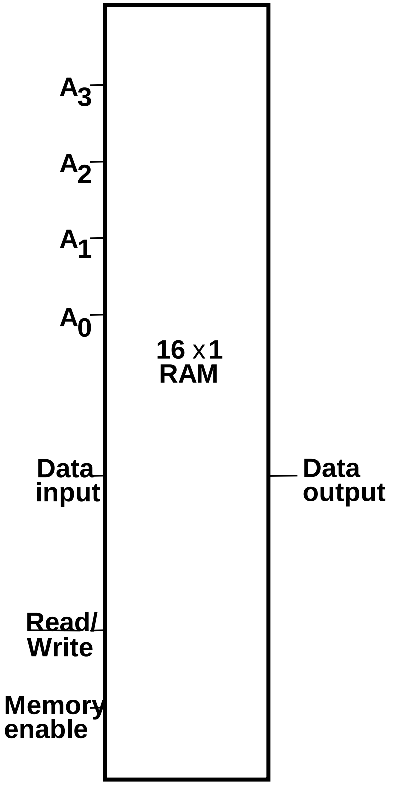


D_3 D_1
D_2 D_0
$\overline{WR}$
$\overline{WR}$ Word 0 $\overline{CS}$
$\overline{WR}$ Word 1 $\overline{CS}$
$\overline{WR}$ Word 2 $\overline{CS}$
$\overline{WR}$ Word 3 $\overline{CS}$
2-to-4 decoder
00
01
10
11
A_0
A_1
Chip Select ($\overline{CS}$)
Q_3 Q_1
Q_2 Q_0

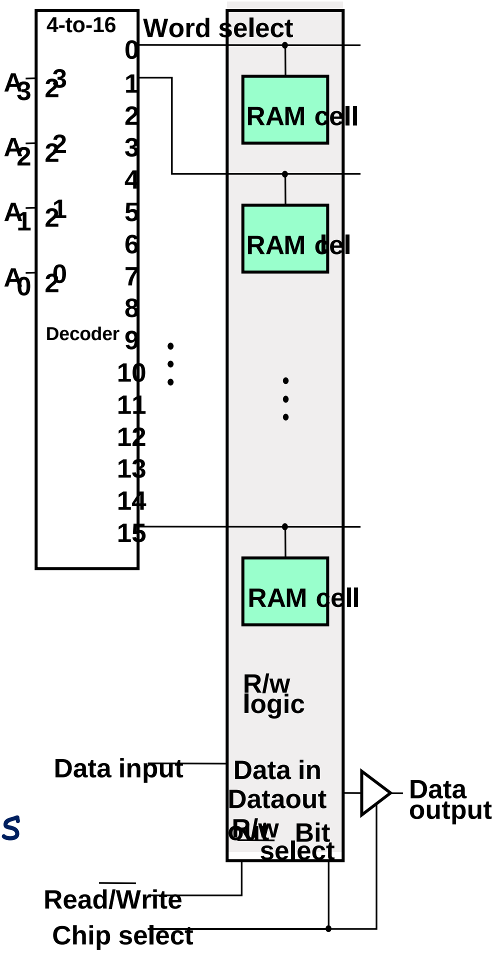
Decoder selects only one memory cell

2^n-Word × 1-Bit RAM IC

- To build a RAM IC from a RAM slice, we need:
  - Decoder decodes the n address lines to 2^n word select lines
  - A 3-state buffer on the data output.
- As memory arrays can be very large ,We need large decoders
- The decoder size and fanouts can be reduced by approximately by $\sqrt{n}$ using a **coincident selection in a 2-dimensional array**.
  - Uses two decoders, one for words and one for bits
  - Word select becomes Row select
  - Bit select becomes Column select

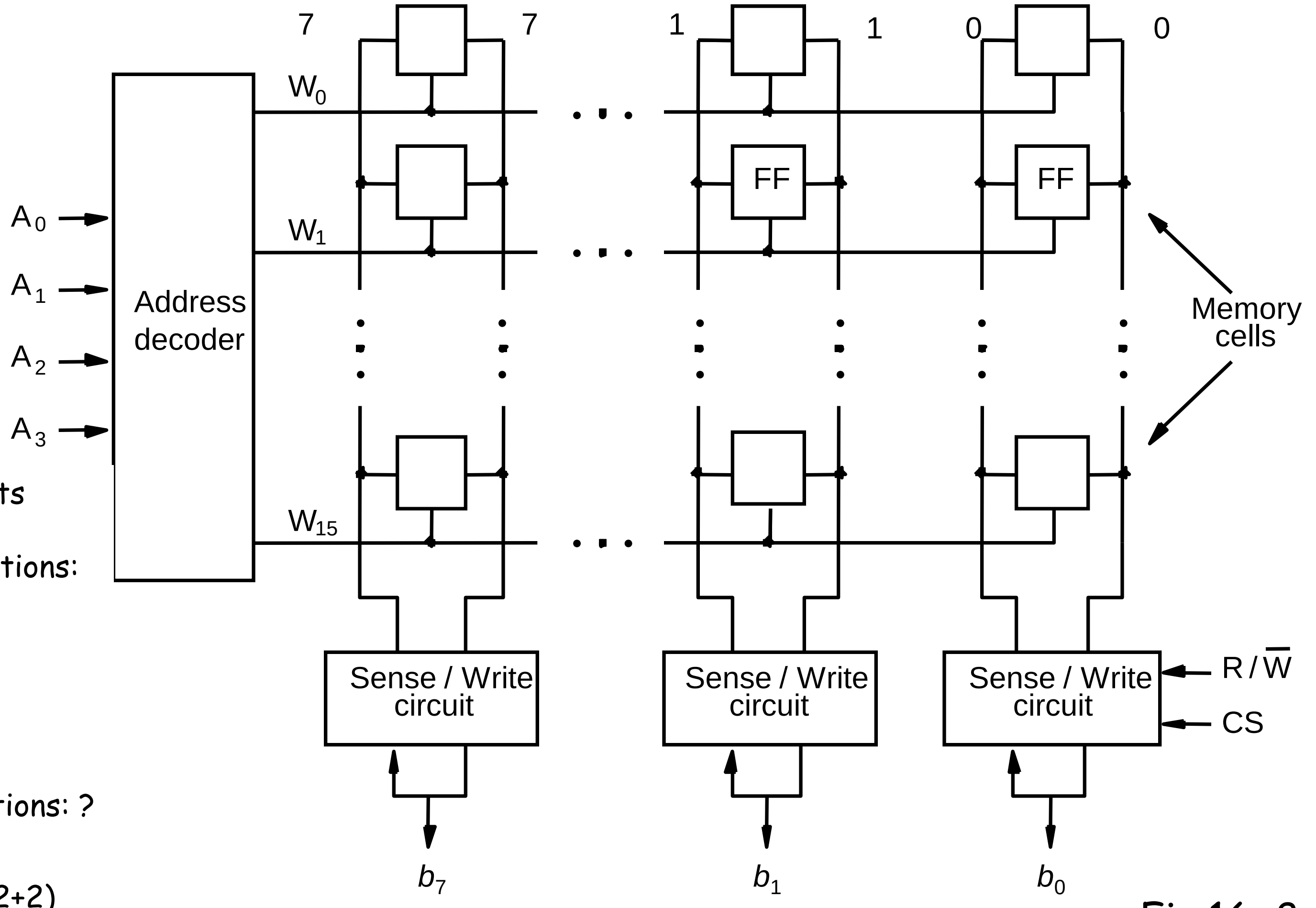


(a) Symbol



(b) Block diagram

Internal organization of memory chips (Contd.,)



16 words of 8 bits each: It has 16 external connections: addr. 4, data 8, control: 2, power/ground: 2

1K memory cells: 128x8 memory, external connections: ? 19 (7+8+2+2)

1Kx1:? 15 (10+1+2+2)

Data input/output lines:

Fig.16x8 memory org.

Cell Arrays and Coincident Selection (continued)

Example
For address 1001:
10→ selects row 2
01→ selects column 1

Cell 9 is accessed.
Address Lines :$A_3A_2A_1A_0$

Row select: A_3A_2

Column select: A_1A_0

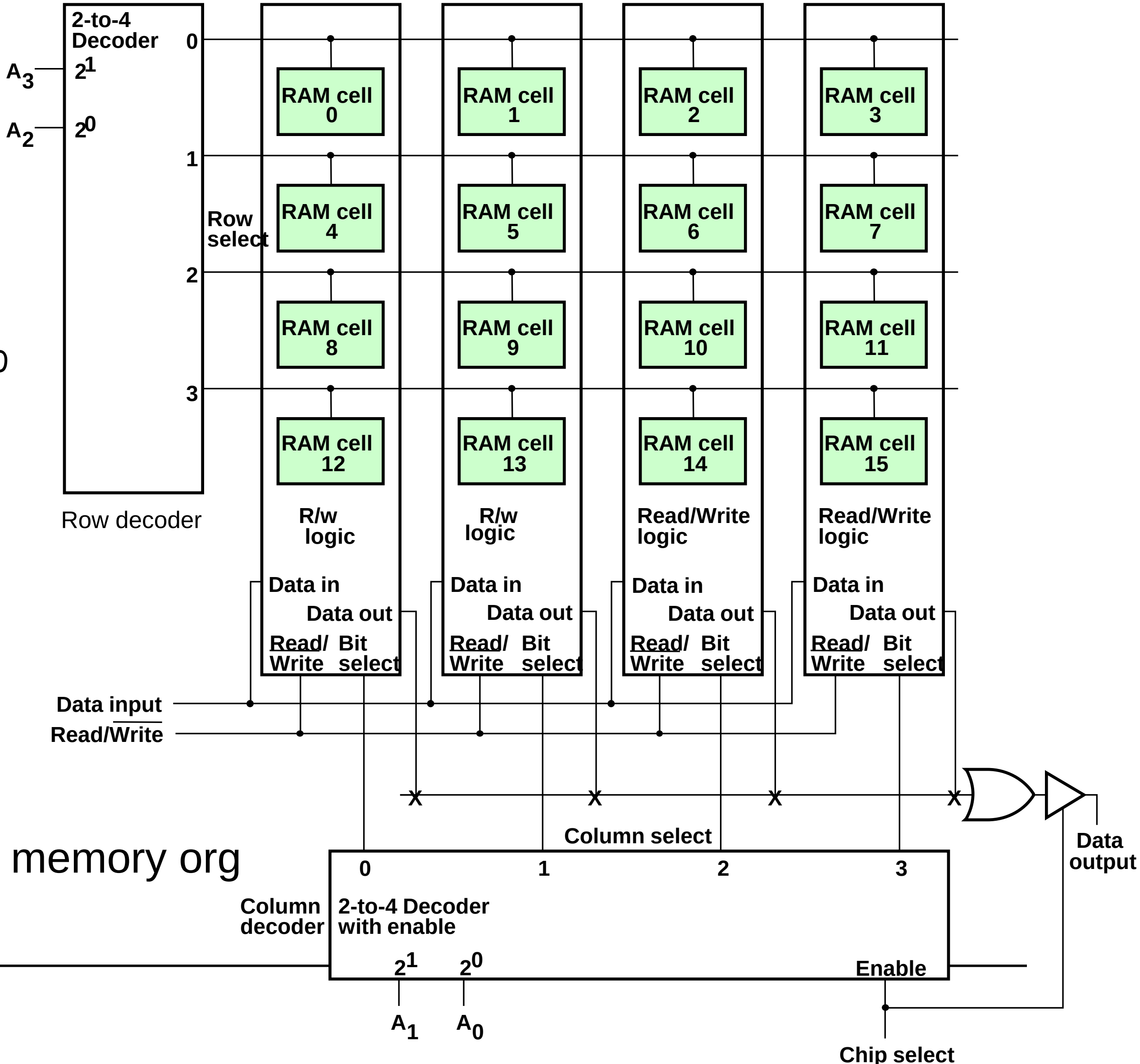


Fig.16x1 memory org

A Memory Chip

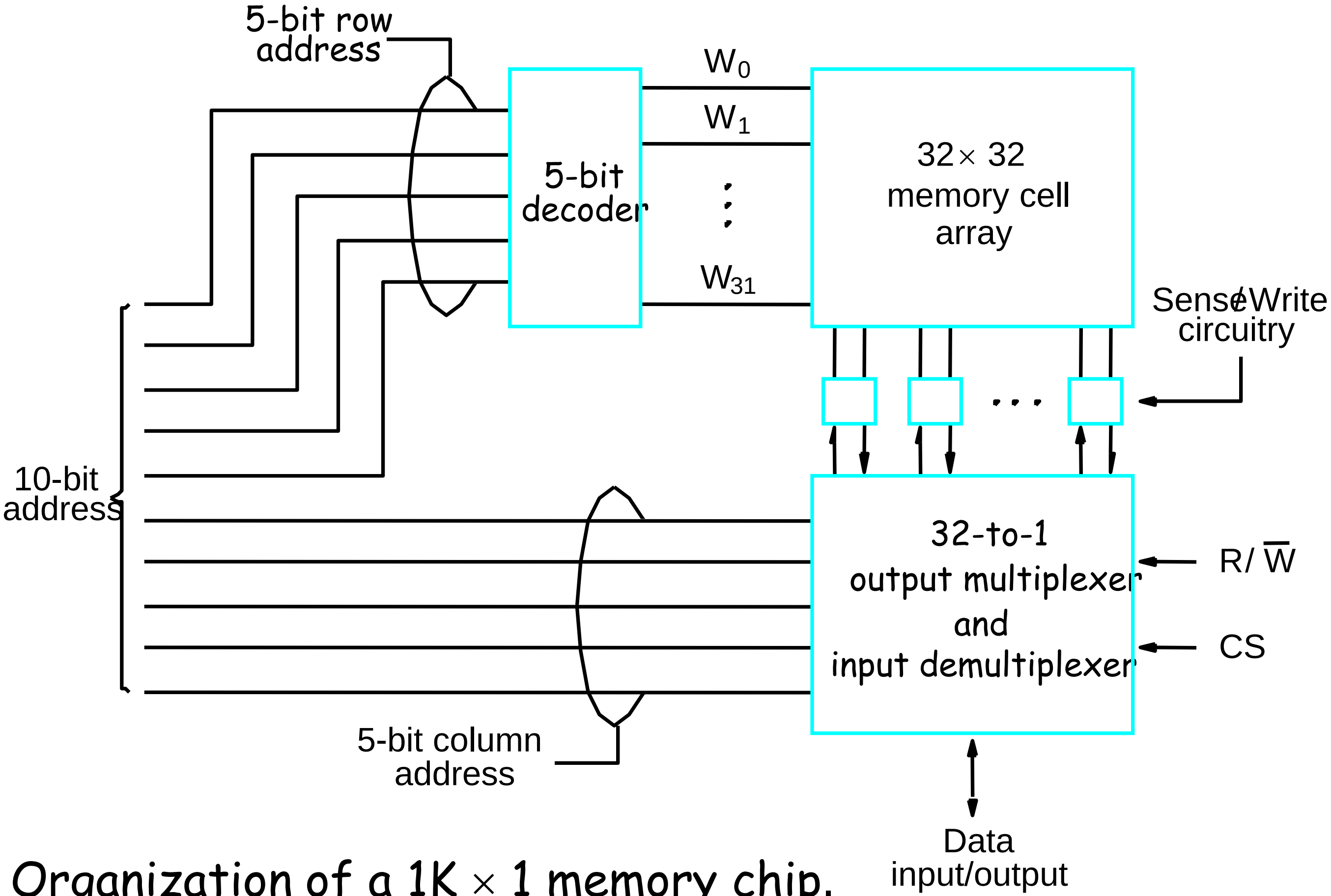
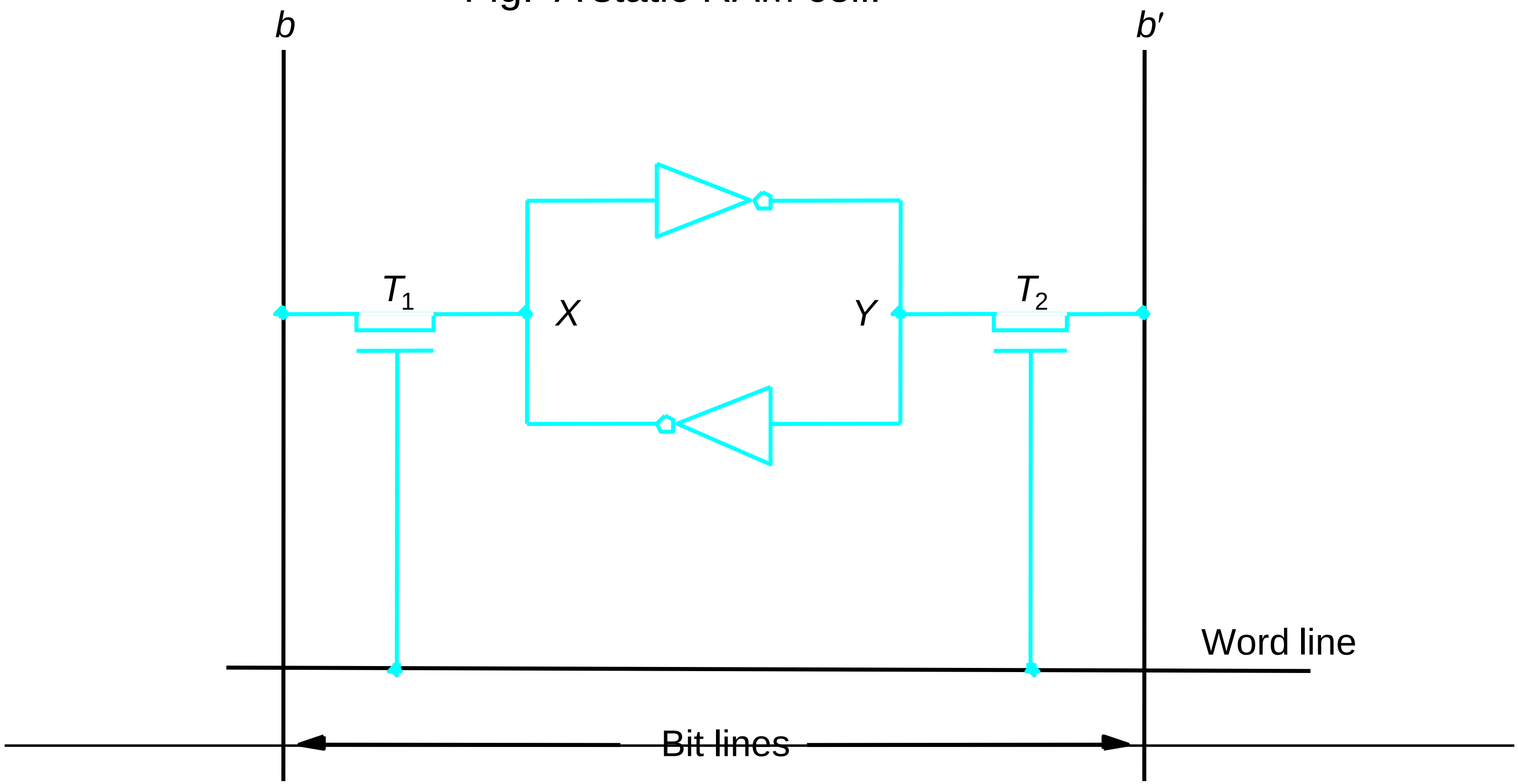


Fig. Organization of a 1K × 1 memory chip.

Static Memories

- The circuits are capable of retaining their state as long as power is applied.

Fig. A static RAM cell.



SRAM Cell

A static RAM cell (Transisitor Latch)

- ❑ Two transistor inverters are cross connected to implement a basic flip-flop (latch).
- ❑ The latch is connected to one word line and two bits lines by transistors T1 and T2.
- ❑ T1 and T2 act as swithes that can be opened or closed under the control of the word line.
- ❑ When word line is at ground level, the transistors are turned off and the latch retains its state.

Ex.The cell is in state 1 if the logic value at X is 1 and at Y is 0.

- ❑ How to read state of SRAM cell:

The word line is activated to close switches T1 and T2.

If the cell is in state 1, the signal on bit line b is high and the signal at bit line b' is low. The opposite is true if the cell is in state 0 Thus b and b' are complements of each other.

Sense/Write circuits at the end of the bit lines monitor the state of b and b' and set the output accordingly.

SRAM cell.

- Write operation:

The state of the cell is set by placing the appropriate value on bit b and b’, and then activating the word line.

This forces the cell into the corresponding state.

The required signals on the bit lines are generated by sense /write circuit.

Static Memories-CMOS Cell

- CMOS cell: low power consumption

Transistor pairs (T3,T5) and (T4,T6) form inverters.

In state 1,the voltage at X is maintained high by having T3 and T6 on ,while T4 and T5 are off,

Thus, if T1 and T2 are turned on (closed), bit lines b and b' will have high and low signals respectively.

Advantage of using CMOS: Low power consumption

Advantage of using Static RAM cell : Access time is less

Asynchronous DRAMs vs SRAMs

- 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. (fast)
  - However, the cost is usually high.
  - Cache memory
- Dynamic RAMs (DRAMs):
  - Do not retain their state indefinitely.
  - Contents must be periodically refreshed.
  - Contents may be refreshed while accessing them for reading.
  - Main memory

Asynchronous DRAMs vs SRAMs

- Both static and dynamic RAMs are volatile, that is, it will retain the information as long as power supply is applied.
- A dynamic memory cell is simpler and smaller than a static memory cell.
  Thus a DRAM is more dense, i.e., packing density is high (more cell per unit area).
  DRAM is less expensive than corresponding SRAM.
- DRAM requires the supporting refresh circuitry.
  For larger memories, the fixed cost of the refresh circuitry is more than compensated for by the less cost of DRAM cells.
- SRAM cells are generally faster than the DRAM cells.
  Therefore, to construct faster memory modules (like cache memory) SRAM is used.

Asynchronous DRAMs

A dynamic memory cell consists of a capacitor and a transistor. In order to store Information in this cell ,transisitor T is turned on and appropriate voltage is applied to the bit line.This caused know amount of charge to be stored in the capacitor. Asfter the transistor is turned off the capacitor begins to discharge. Hence the information stored in the cell need to be retrieved.

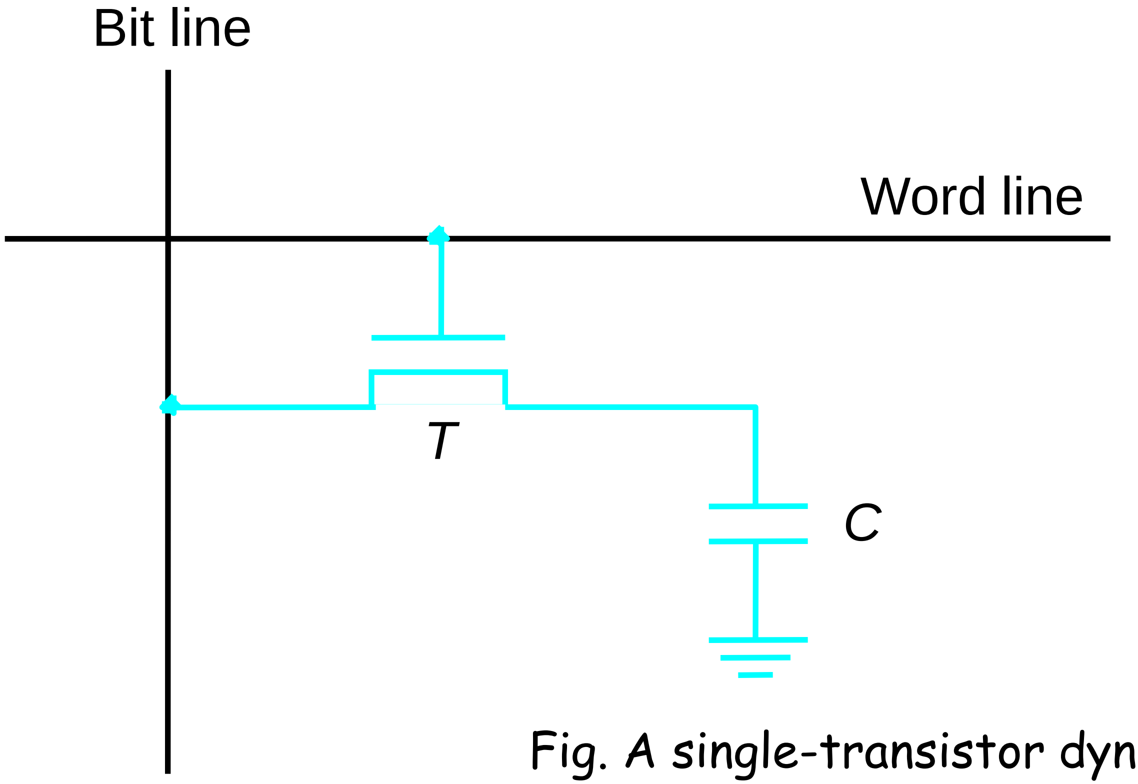


Fig. A single-transistor dynamic memory cell

Asynchronous DRAMs

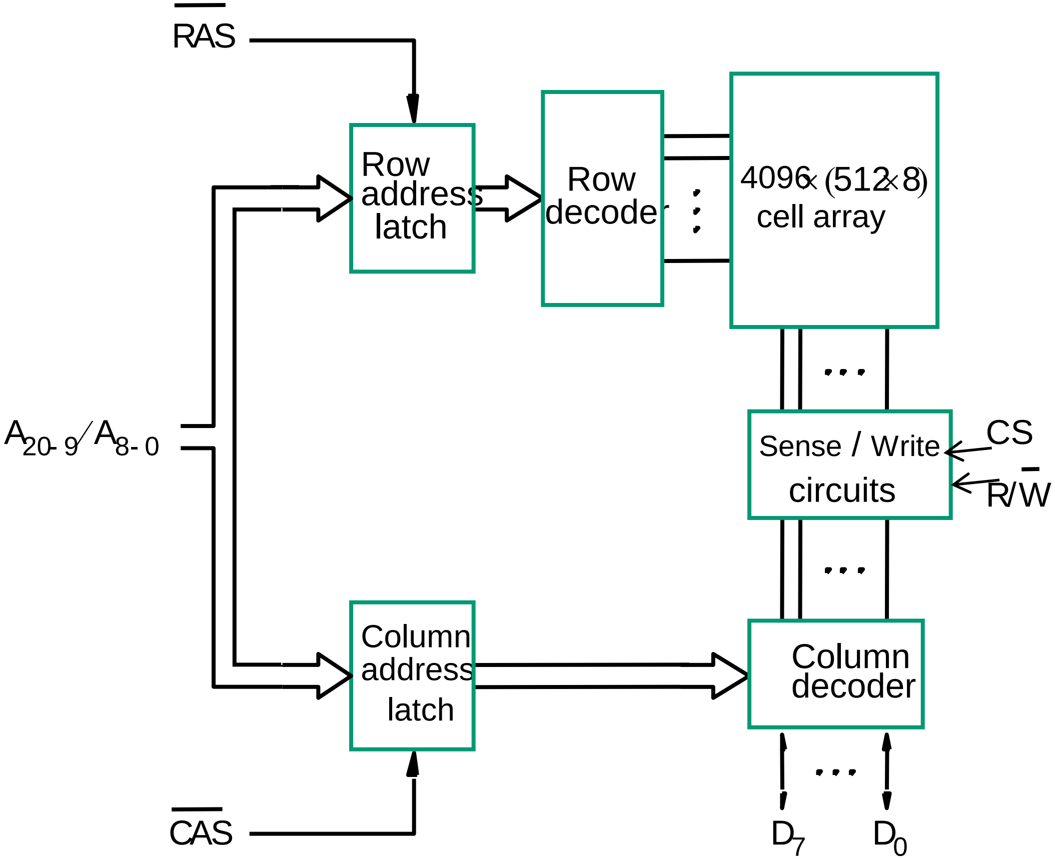


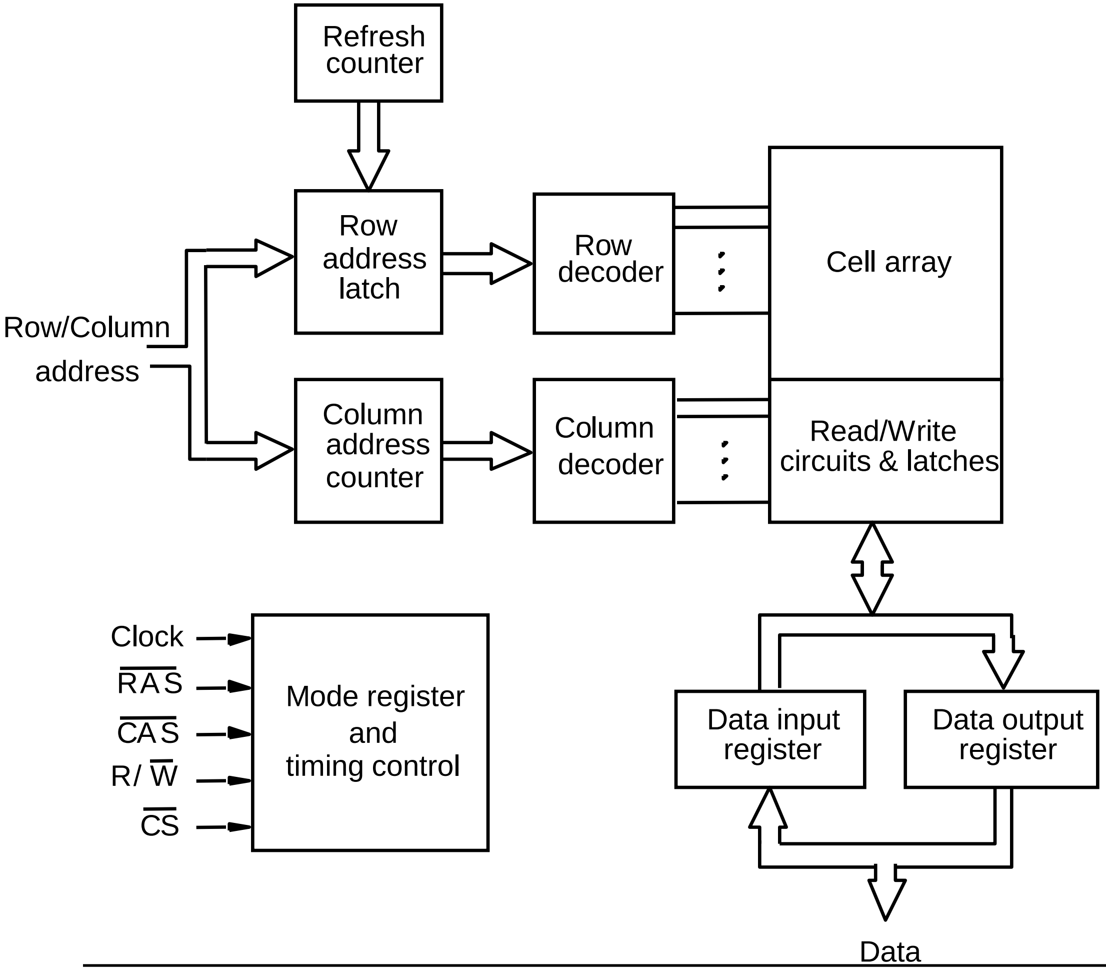
Fig. Internal organization of a 2M x 8 dynamic memory chip.

- ❑ 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. CAS signal need not be generated externally. A new data is placed during raising edge of the clock

Latency and Bandwidth

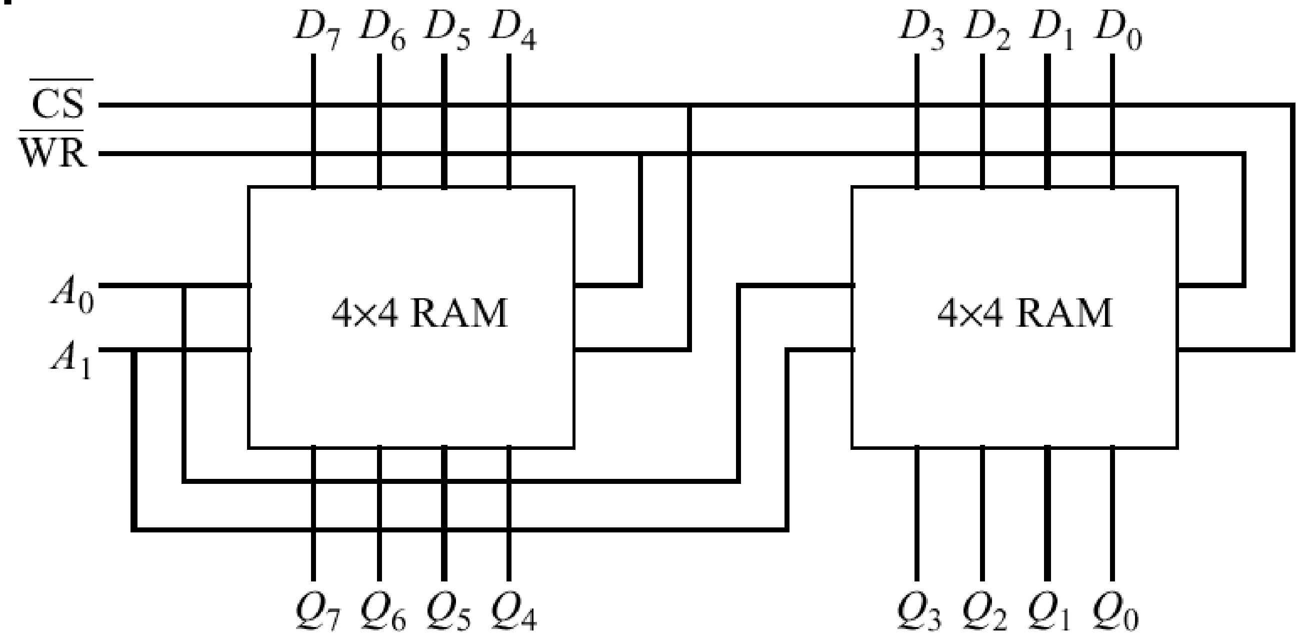
- The speed and efficiency of data transfers among memory, processor, and disk have a large impact on the performance of a computer system.
- Memory latency – the amount of time it takes to transfer a word of data to or from the memory.
- Memory bandwidth – the number of bits or bytes that can be transferred in one second. It is used to measure how much time is needed to transfer an entire block of data.
- Bandwidth- is not determined solely by memory. It is the product of the rate at which data are transferred (and accessed) and the width of the data bus.

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

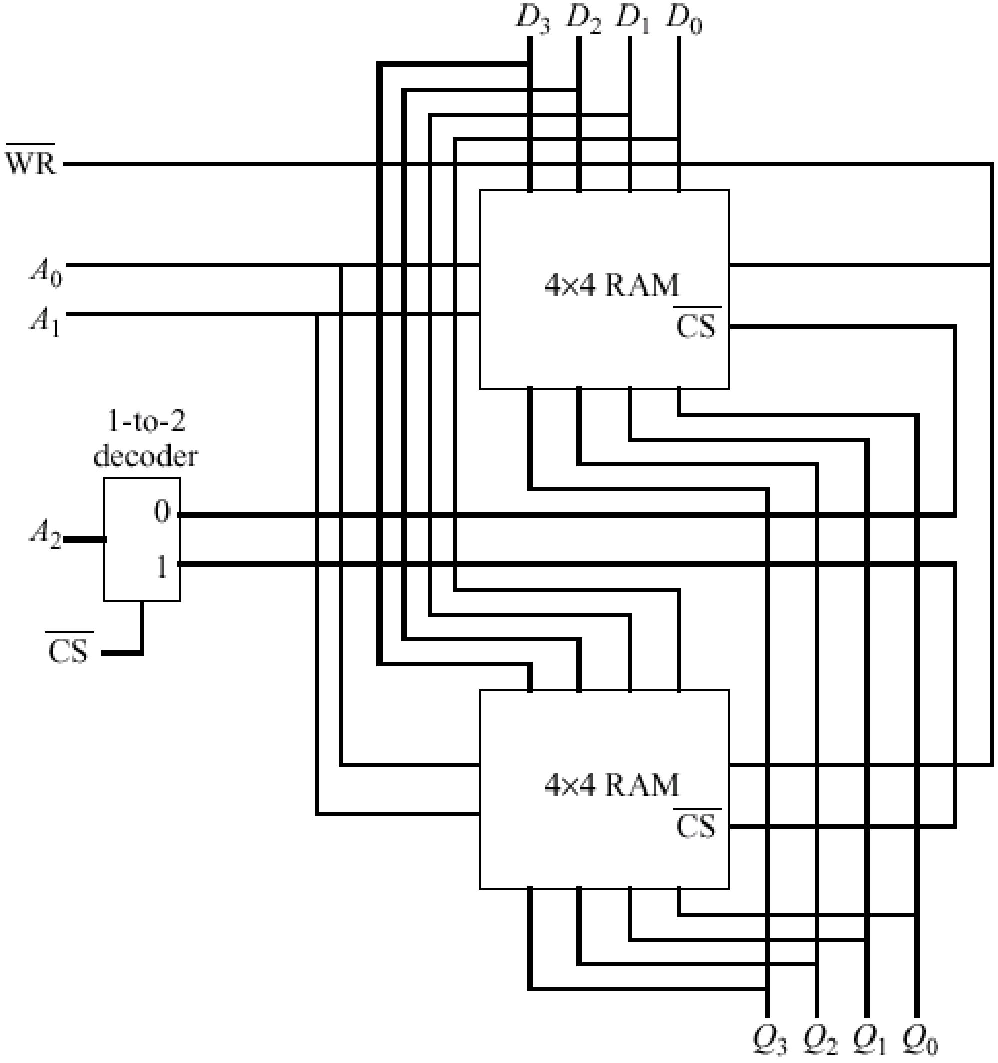
Constructing Wider Memory

- Can pair two of our 4 word x 4 bit chips to make a 4 word x 8 bit chip : Use both in parallel

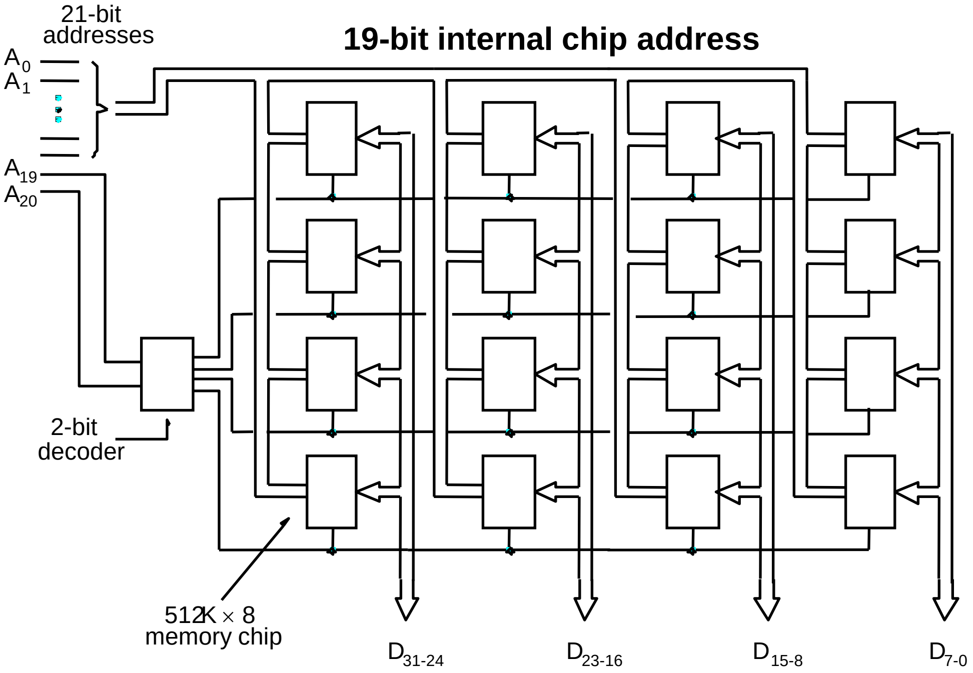


Constructing Longer Memory

- We can combine chips to create a 8 word x 4 bit memory. Third address bit goes to a decoder to select only one of the two chips.



Structure of large memories: Static memories



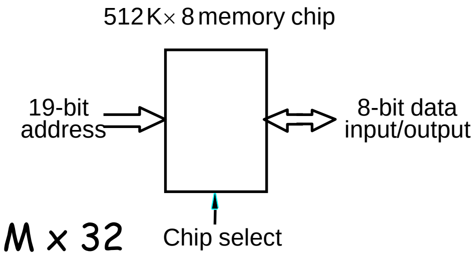


Fig. Organization of 2M x 32

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.

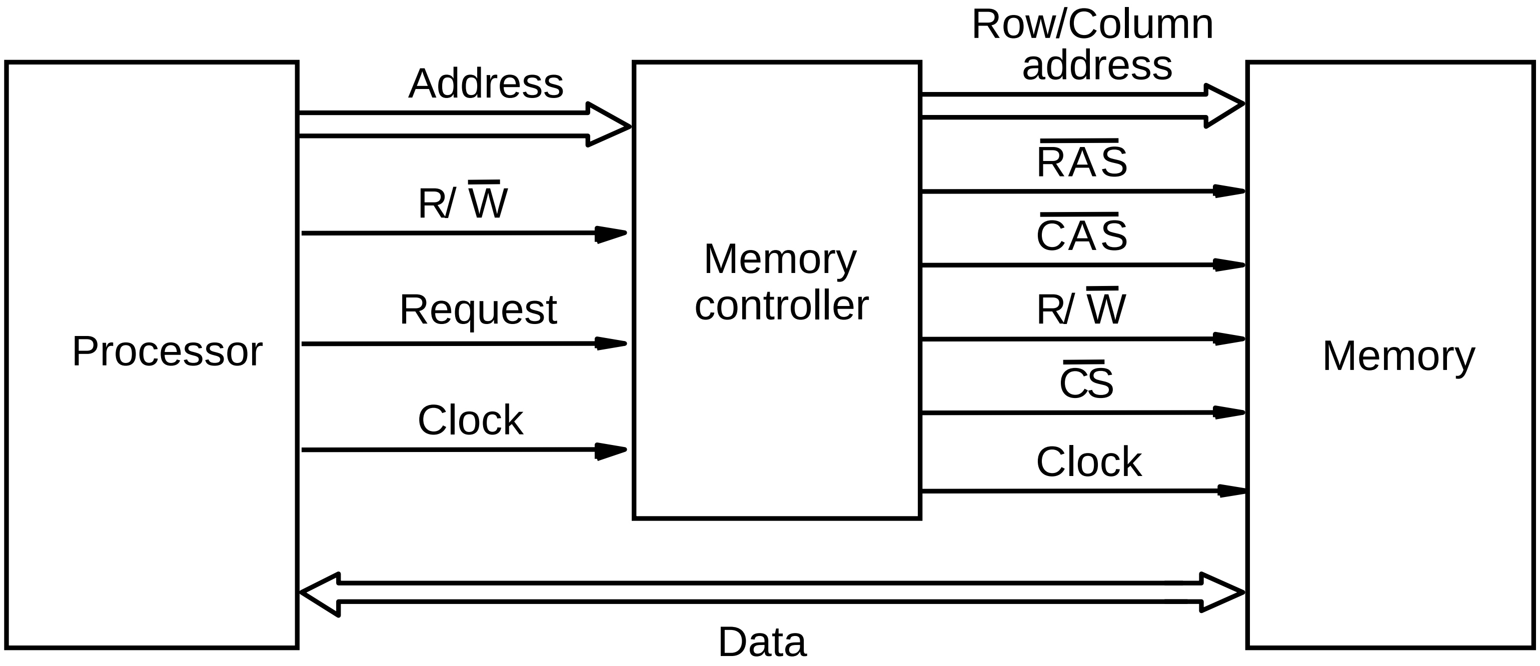
Large memories: Dynamic memories

- 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

- To reduce the number of pins, the dynamic memory chips use multiplexed address inputs.
- 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)

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-Memory

- ❑ Volatile / non-volatile memory
- ❑ ROM
- ❑ PROM: programmable ROM

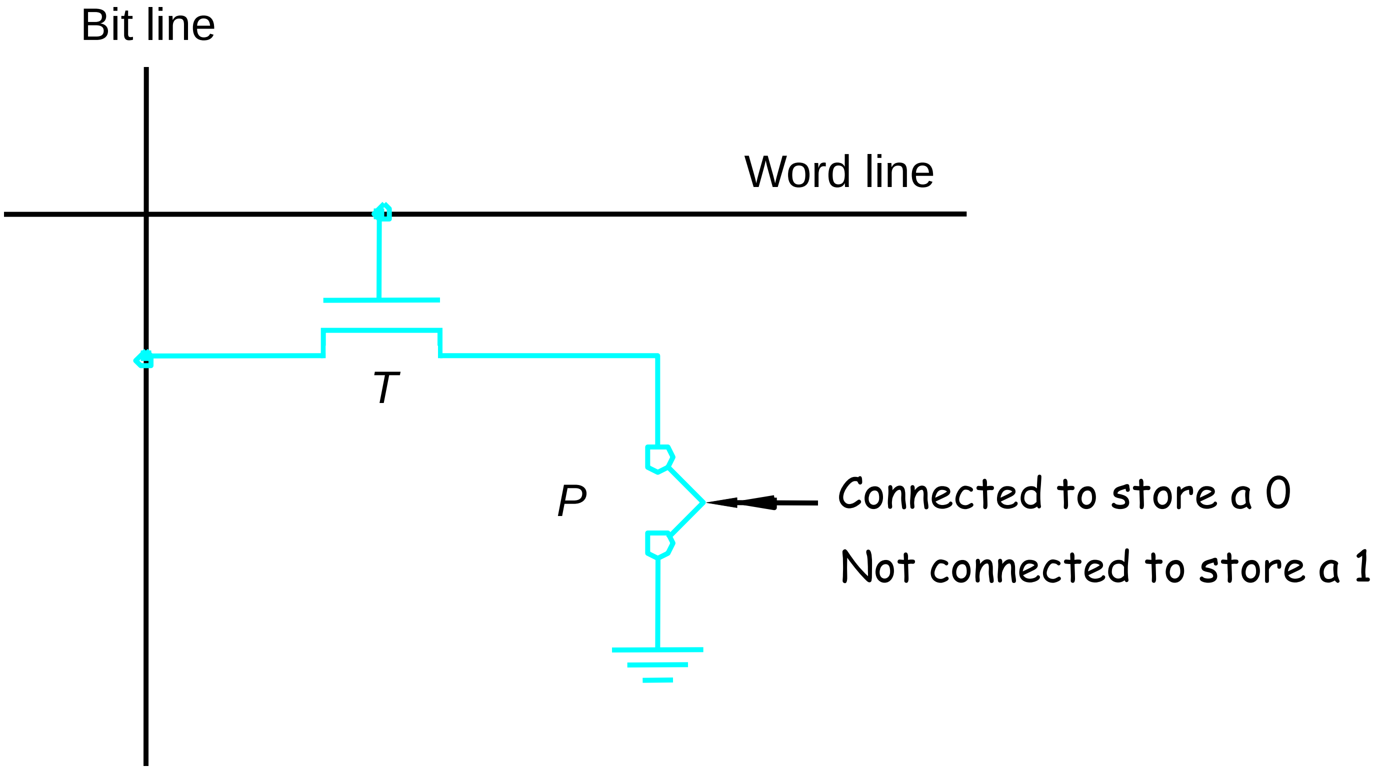


Fig. A ROM cell.

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.

Cache Memories

Cache Memories

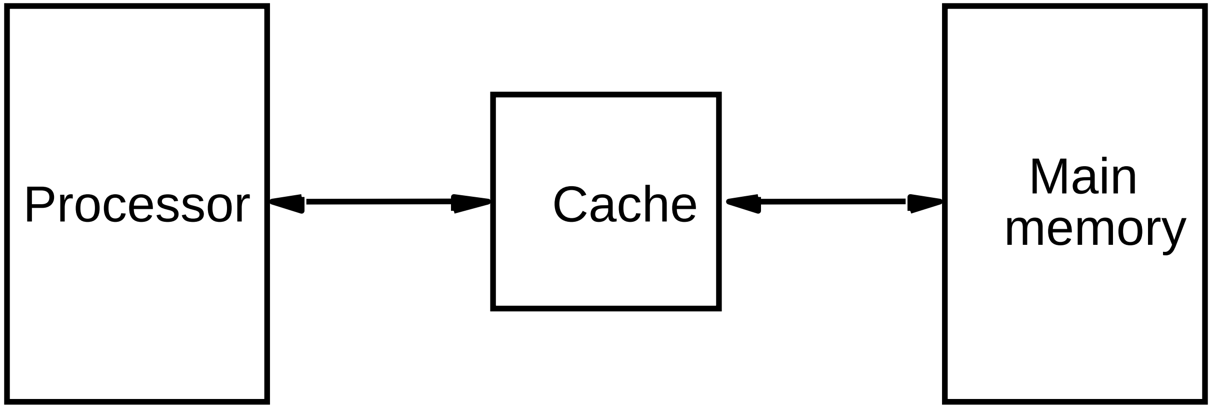
- Processor is much faster than the main memory.
  - As a result, the processor has to spend much of its time waiting while instructions and data are being fetched from the main memory.
  - Major obstacle towards achieving good performance.
- Speed of the main memory cannot be increased beyond a certain 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.

Cache Design

- Size
- Mapping Function
- Replacement Algorithm
- Write Policy
- Block Size
- Number of Caches

Tag Fields

- A cache line contains two fields

– Data from RAM

– The address of the block currently in the cache.

- The part of the cache line that stores the address of the block is called the tag field.

Many different addresses can be in any cache line.

The tag specifies the address currently in the cache line.

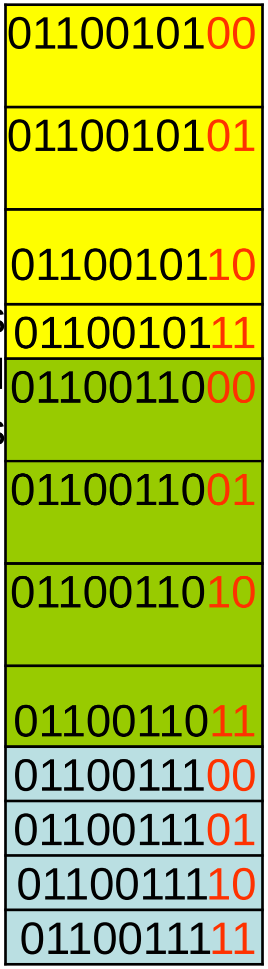
Only the upper address bits are needed.

Cache Lines

- The cache memory is divided into **blocks** or **lines**. Currently lines can range from 16 to 64 bytes.
- Data is copied to and from the cache one line at a time.
- The lower log2(line size) bits of an address specify a particular byte in line.

Line Example

These boxes represent RAM addresses



01100101**00**
01100101**01**
01100101**10**
01100101**11**
01100110**00**
01100110**01**
01100110**10**
01100110**11**
01100111**00**
01100111**01**
01100111**10**
01100111**11**

With a line size of 4,

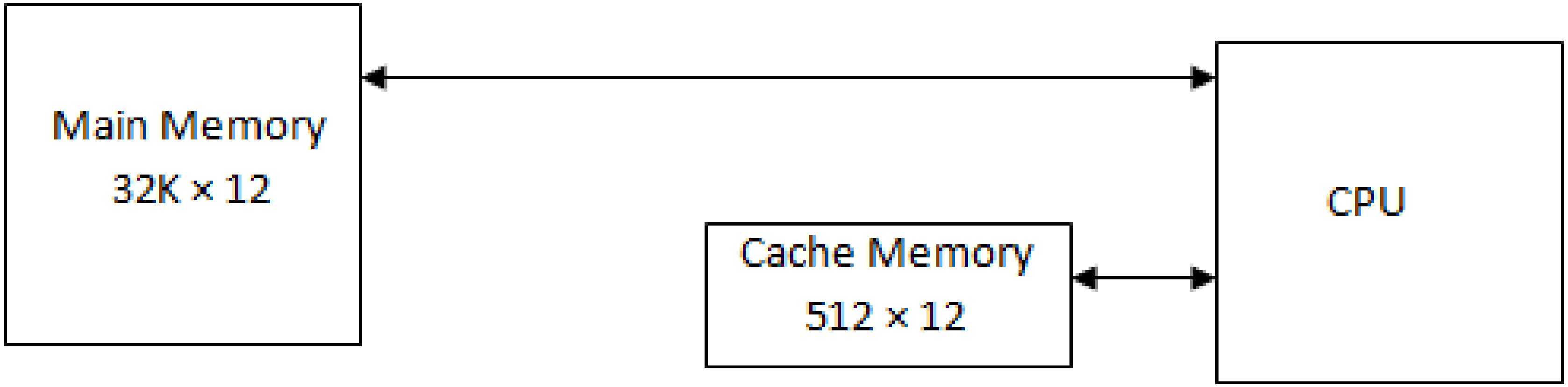
the offset is the log2(4) = 2 bits. The lower 2 bits specify which byte in the line

Mapping functions

- Mapping functions determine how memory blocks are placed in the cache.
- Three mapping functions:
  - Direct mapping
  - Associative mapping
  - Set-associative mapping.

Example

A system with 512 X 12 Cache and 32k X 12 of main memory



Direct Mapping

- Simplest mapping technique - each block of main memory maps to only one cache line

  i.e. if a block is in cache, it must be in one specific place

  Main memory locations can only be copied into one location in the cache, accomplished by dividing main memory into blocks that correspond in size with the cache.

- Formula to map a memory block to a cache line:
  - i = j mod c
    - i=Cache Line Number (block address of cache)
    - j=Main Memory Block Number(block address of main memory)
    - c=Number of Lines in Cache(no.of cache blocks)
  - i.e. we divide the memory block by the number of cache lines and the remainder is the cache line address

1.Direct mapping

The direct mapping technique is simple and inexpensive to implement.

When the CPU wants to access data from memory, it places a address. The index field of CPU address is used to access address.

The tag field of CPU address is compared with the associated tag in the word read from the cache.

If the tag-bits of CPU address is matched with the tag-bits of cache, then there is a *hit* and the required data word is read from cache.

If there is no match, then there is a *miss* and the required data word is stored in main memory.

It is then transferred from main memory to cache memory with the new tag.

Direct Mapping-with word transfers

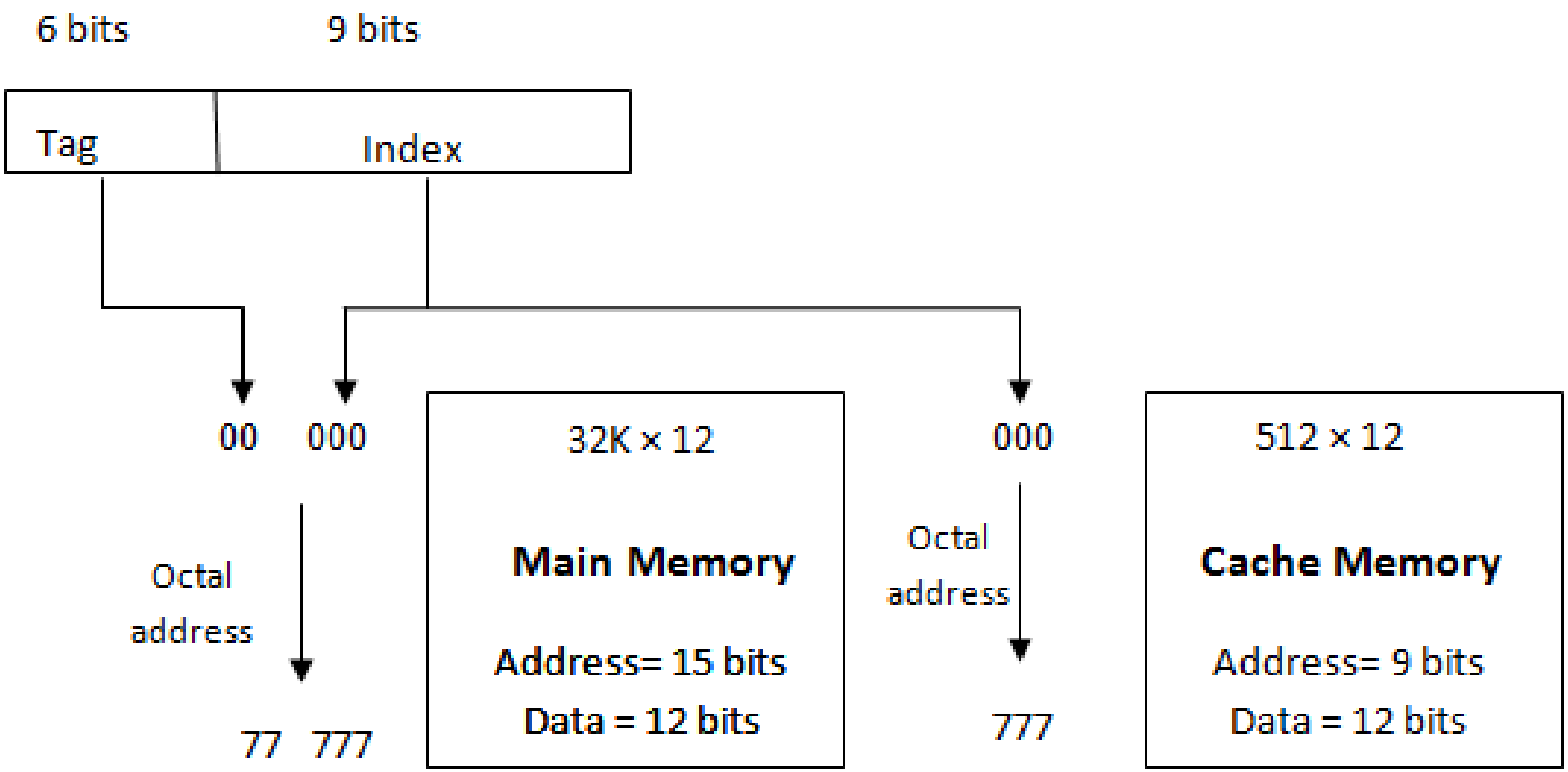
- Direct Cache Addressing
- The lower $\log_2$(line size) bits define which byte in the block
- The next $\log_2$(number of lines) bits defines which line of the cache
- The remaining upper bits are the tag field.

Cache Constants

- cache size / line size = number of lines
- $\log_2$(line size) = bits for offset
- $\log_2$(number of lines) = bits for cache index
- remaining upper bits = tag address bits

Given problem:
w=4 words=2^2
Divide the remaining main memory block size into two parts.(s=22)
r and (s-r) fields.
One part should be equal to cache block of cache called Index field (r=14)
And other is (s-r) =8,called as tag field.

1.Direct mapping



6 bits

9 bits

Tag

Index

00 000

Octal address

77 777

32K × 12

Main Memory

Address= 15 bits

Data = 12 bits

000

Octal address

777

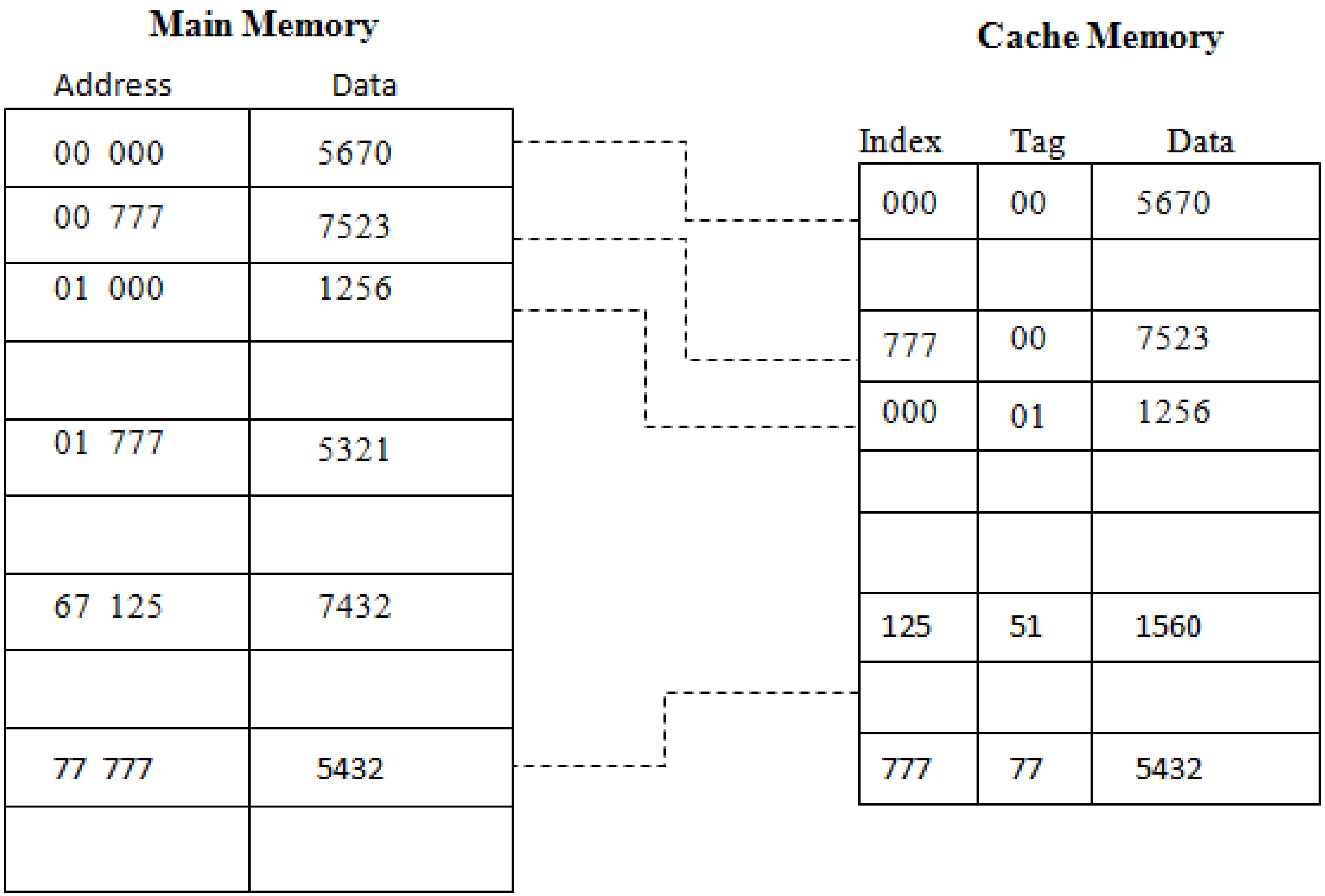
512 × 12

Cache Memory

Address= 9 bits

Data = 12 bits

1.Direct mapping



Main Memory

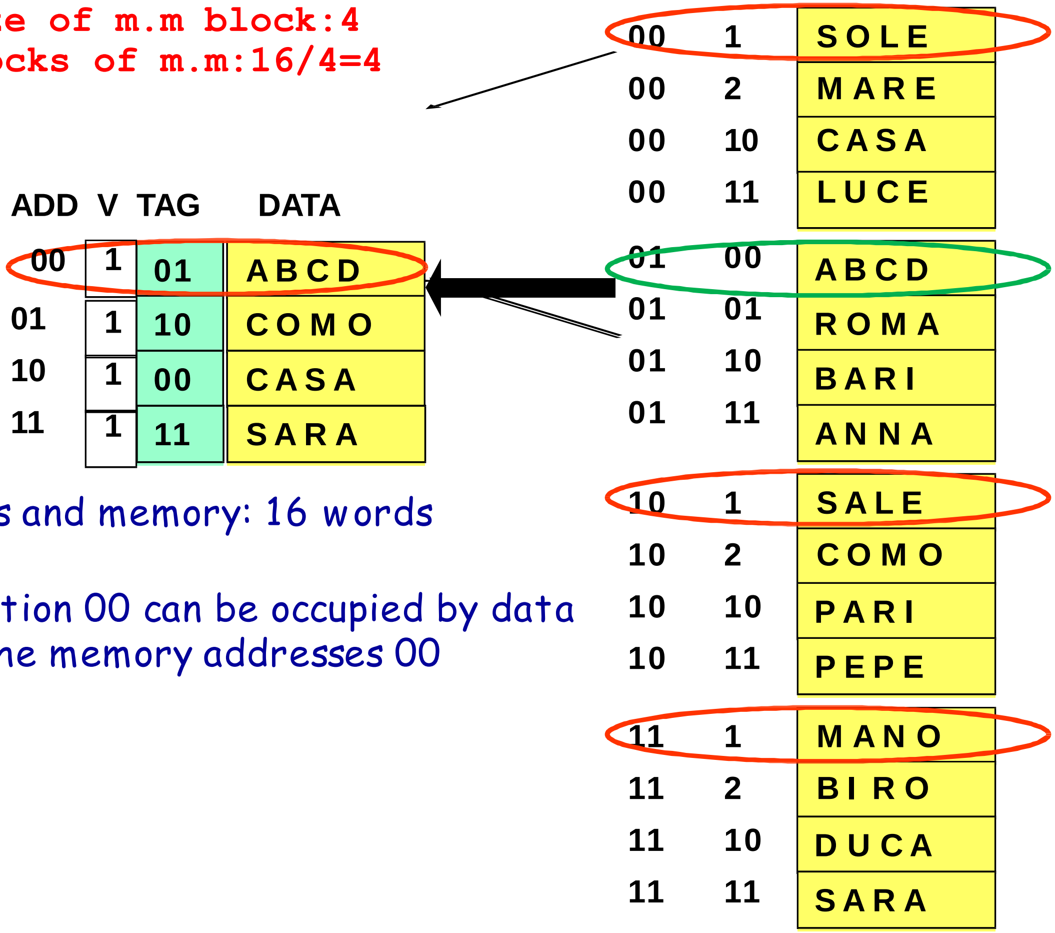
Address	Data
00 000	5670
00 777	7523
01 000	1256
01 777	5321
67 125	7432
77 777	5432

Cache Memory

Index	Tag	Data
000	00	5670
777	00	7523
000	01	1256
125	51	1560
777	77	5432

Direct Mapping Example

Cache line:4
(index:S)Size of m.m block:4
(t)No.of blocks of m.m:16/4=4
W=1



ADD	V	TAG	DATA
00	1	01	ABCD
01	1	10	COMO
10	1	00	CASA
11	1	11	SARA

ADDRESS

TAG	INDEX	DATA
00	1	SOLE
00	2	MARE
00	10	CASA
00	11	LUCE
01	00	ABCD
01	01	ROMA
01	10	BARI
01	11	ANNA
10	1	SALE
10	2	COMO
10	10	PARI
10	11	PEPE
11	1	MANO
11	2	BIRO
11	10	DUCA
11	11	SARA

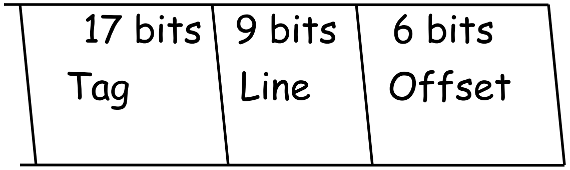
Cache: 4 words and memory: 16 words

The cache location 00 can be occupied by data coming from the memory addresses 00

How many bits are in the tag, line and offset fields?

Example direct address

- Assume you have 32 bit addresses (m.m can address 4 GB) $=2^{32}$
- 64 byte lines (2^6: offset is 6 bits)
- 32 KB of cache
- Number of lines = 32 KB / 64 = 512 $=2^9$
- Bits to specify which line = $\log_2(512) = 9$



17 bits | 9 bits | 6 bits

Tag | Line | Offset

How many bits are in the tag, line and offset fields?

24 bit addresses

64K bytes of cache

16 byte cache lines

- tag=4, line=16, offset=4
- tag=4, line=14, offset=6
- tag=8, line=12, offset=4
- tag=6, line=12, offset=6

Direct Mapping -Address Structure

Main memory address

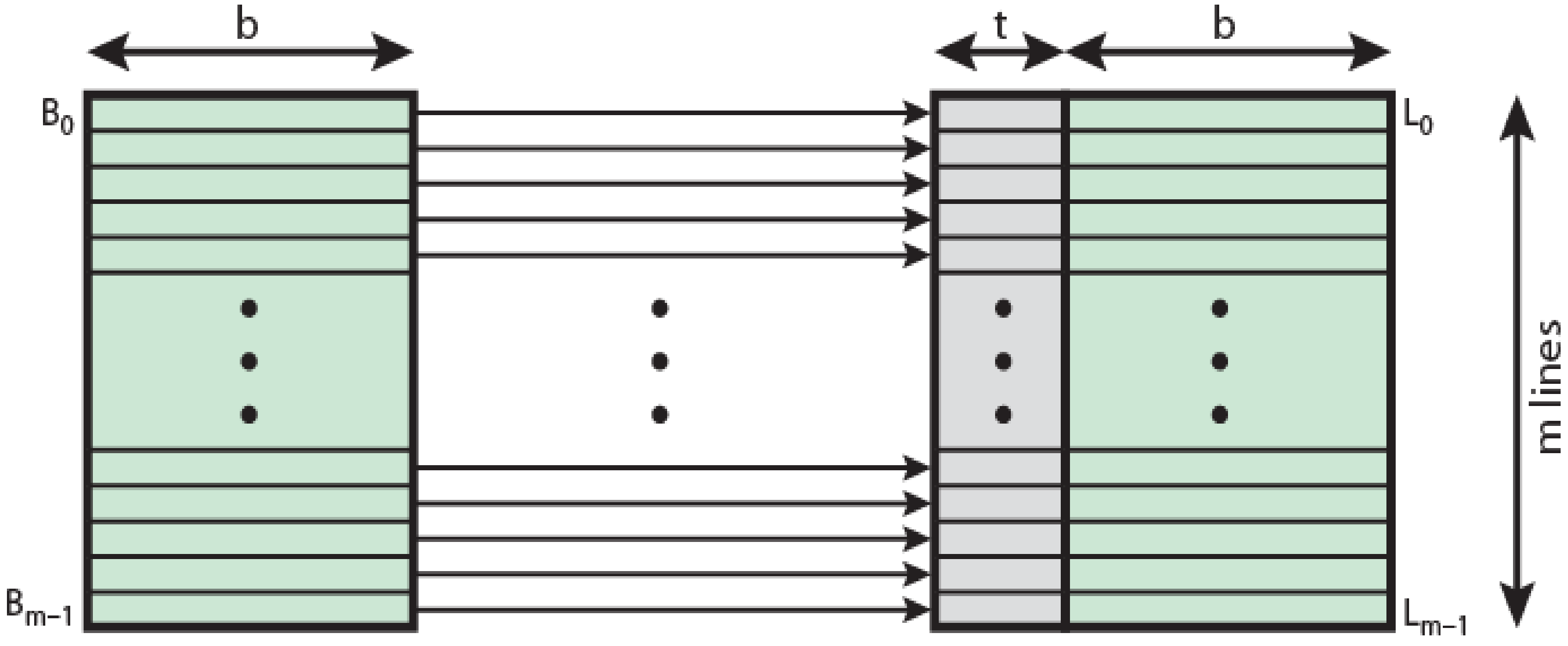
Tag	Index/ line	word

- 24 bit address
- w=2 bit word identifier (4 byte block)

Address field:

- 22 bit block identifier s=14,
  - 8 bit tag (=22-14)
  - 14 bit slot or line
- No two blocks in the same line have the same Tag field
- Check contents of cache by finding line and checking Tag

Direct Mapping from Cache to Main Memory



First m blocks of main memory (equal to size of cache)

cache memory

b = length of block in bits
t = length of tag in bits

(a) Direct mapping

Direct Mapping Cache Line Table

Cache line	Main Memory blocks held
0	0, m, 2m, 3m…2s-m
1	1,m+1, 2m+1…2s-m+1
…	
m-1	m-1, 2m-1,3m-1…2s-1

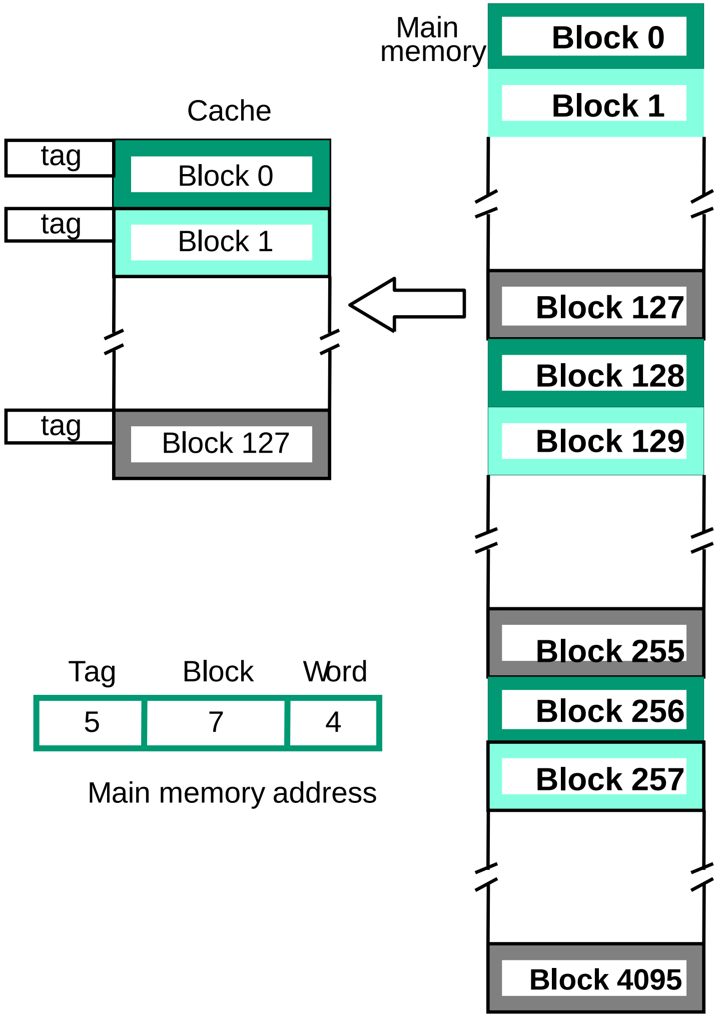
Direct Mapping pros & cons

- Simple
- Inexpensive
- Fixed location for given block
  - If a program accesses 2 blocks that map to the same line repeatedly, cache misses are very high

Direct mapping-example2

- 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.

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.

Mapping Function – 64K Cache Example

- Suppose we have the following configuration
  - Word size of 1 byte
  - Cache of 64 KByte
  - Cache line / Block size is 4 bytes
    - i.e. cache is 64 Kb / 4 bytes = 16,384 (2^{14}) lines of 4 bytes
  - Main memory of 16MBytes
    - 24 bit address
    - (2^{24}=16M)
    - 16Mb / 4bytes-per-block → 4 Meg of Memory Blocks
  - Somehow we have to map the 4 Meg of blocks in memory onto the 16K of lines in the cache.
  - Multiple memory blocks will have to map to the same line in the cache!

2.Fully Associative Mapping

An associative mapping uses an associative memory.

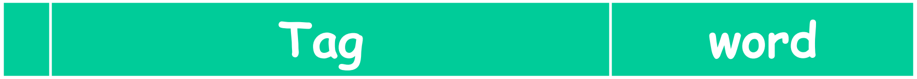
This memory is being accessed using its contents.

Each line of cache memory will accommodate the address (main memory) and the contents of that address from the main memory.

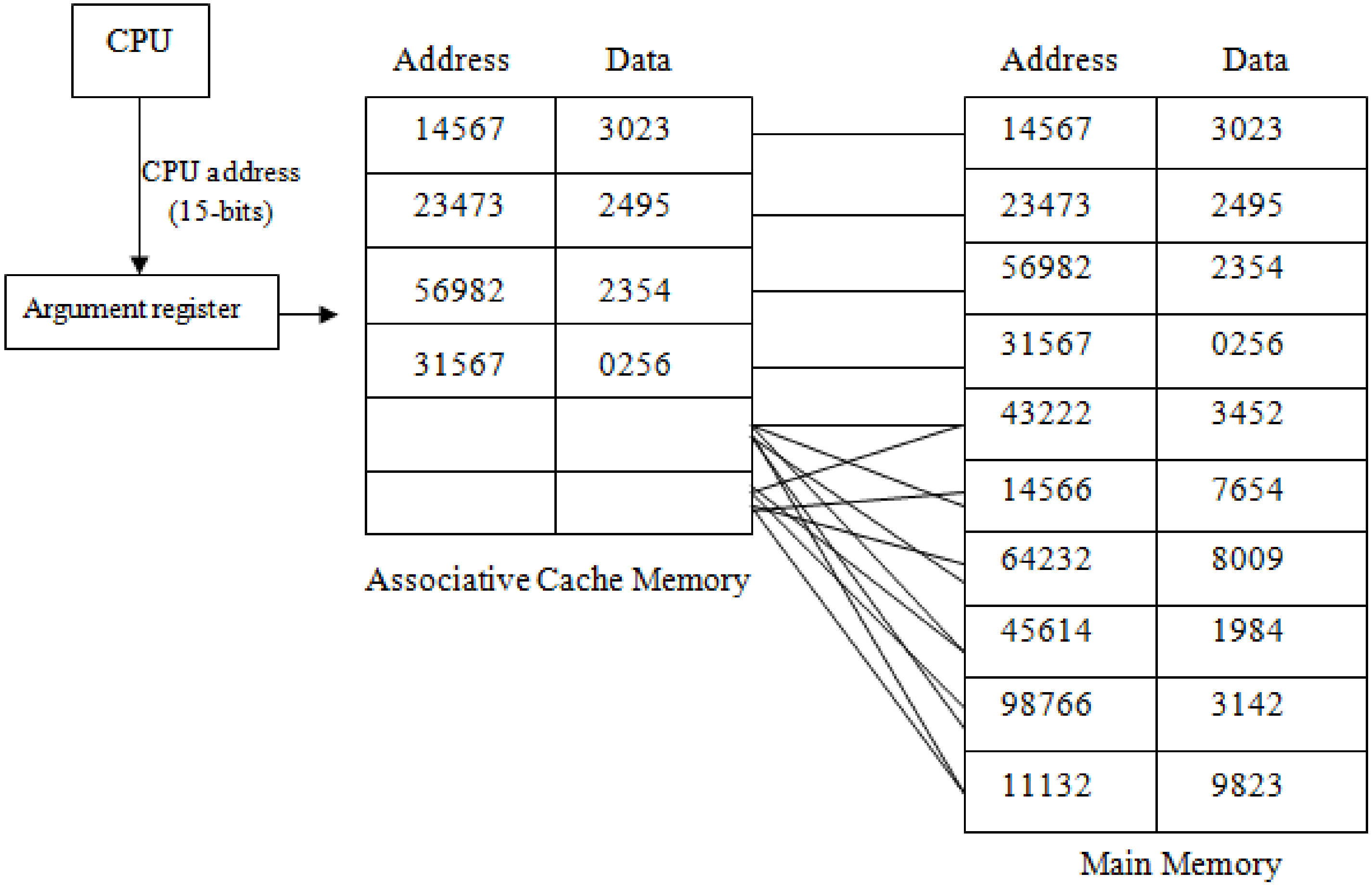
That is why this memory is also called content addressable Memory (CAM). It allows each block of main memory to be stored in the cache.

2. Fully Associative Mapping

- A fully associative mapping scheme can overcome the problems of the direct mapping scheme
  - A main memory block can load into any line of cache
  - Memory address is interpreted as tag and word
  - Tag uniquely identifies block of memory
  - Every line's tag is examined for a match
  - Also need a Dirty and Valid bit
- But Cache searching gets expensive!
  - Ideally need circuitry that can simultaneously examine all tags for a match
  - Lots of circuitry needed, high cost
- Need replacement policies now that anything can get thrown out of the cache (will look at this shortly)



2. Fully Associative Mapping



Address	Data
14567	3023
23473	2495
56982	2354
31567	0256

Associative Cache Memory

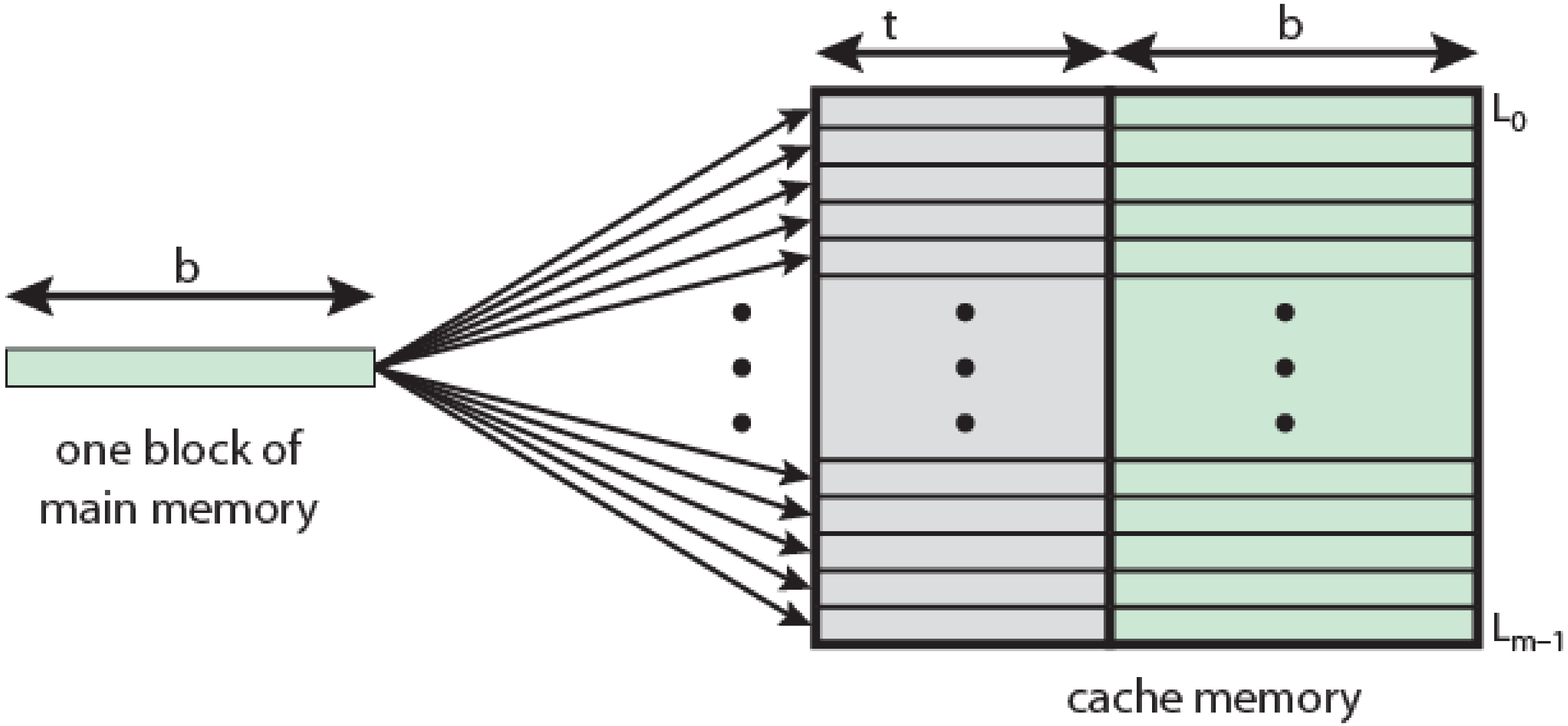
Address	Data
14567	3023
23473	2495
56982	2354
31567	0256
43222	3452
14566	7654
64232	8009
45614	1984
98766	3142
11132	9823

Main Memory

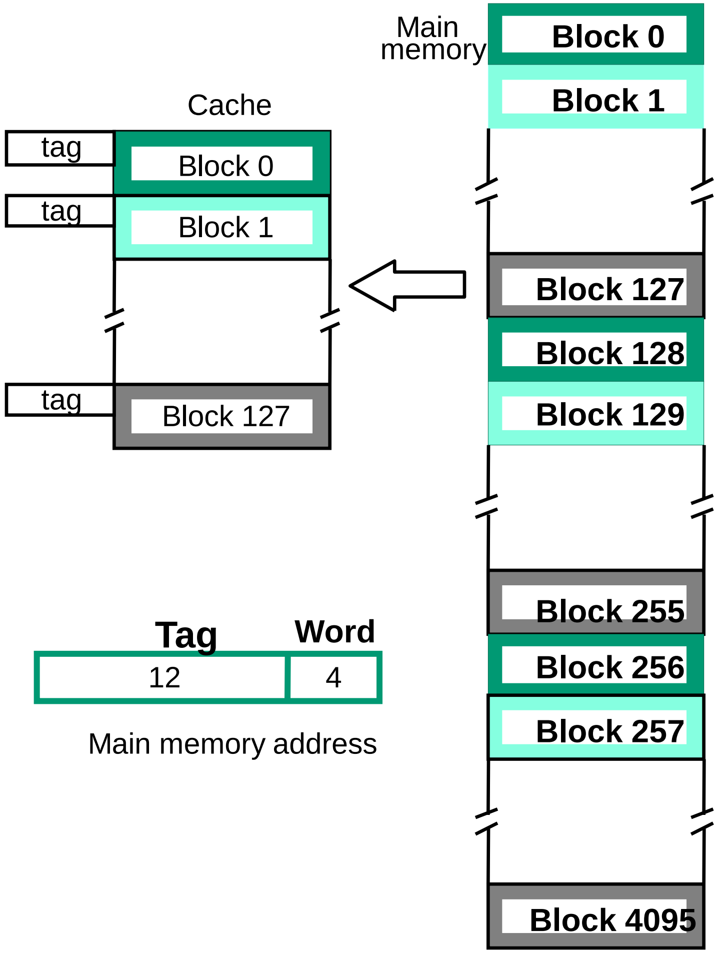
Associative Mapping

- A main memory block can load into any line of cache
- Memory address is interpreted as tag and word
- Tag uniquely identifies block of memory
- Every line’s tag is examined for a match
- Cache searching gets expensive

Associative Mapping from Cache to Main Memory



Associative mapping



- Main memory block can be placed into any cache position.
- Memory address is divided into 2 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.

3.Set Associative Mapping

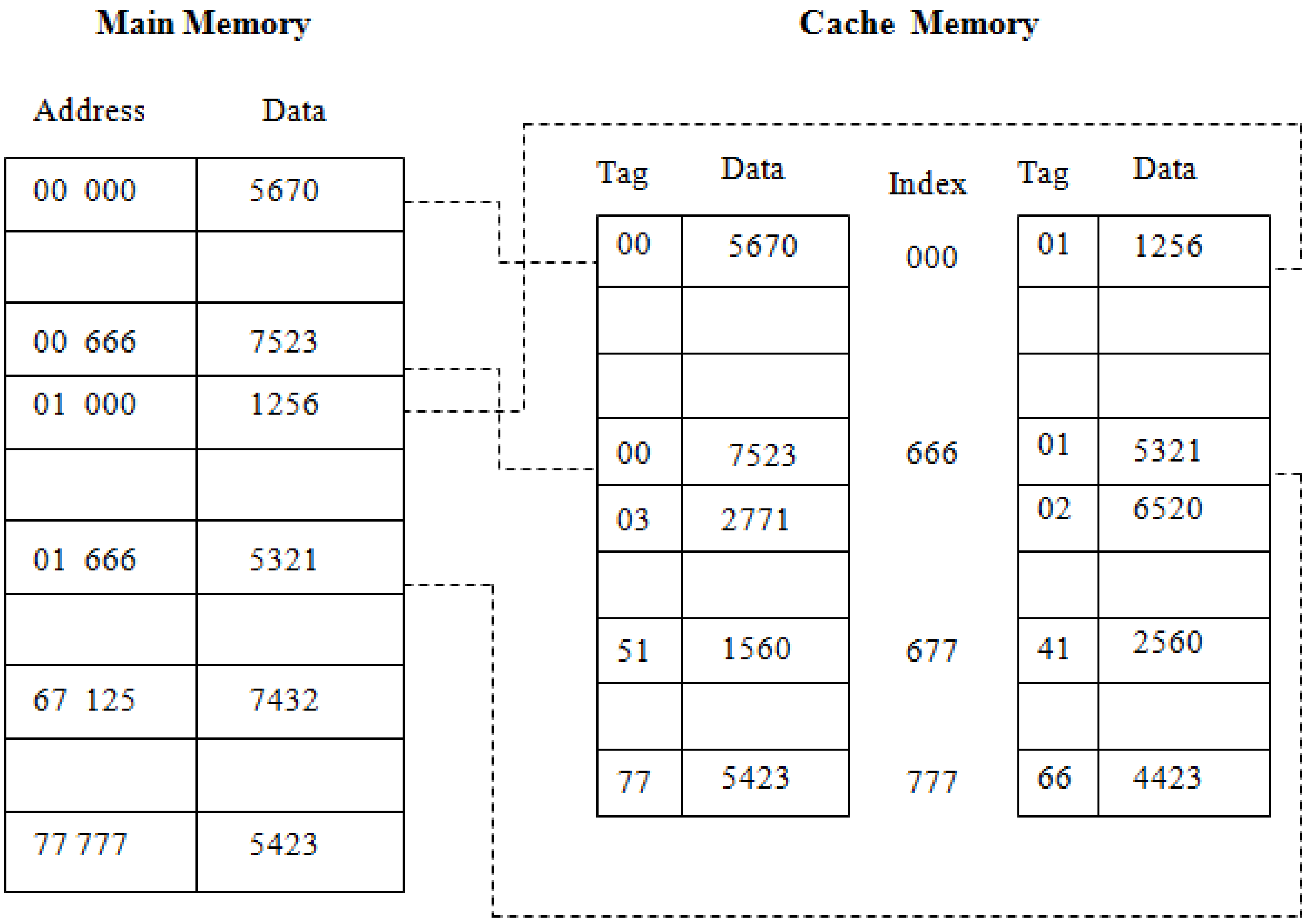
That is the easy control of the direct mapping cache and the more flexible mapping of the fully associative cache.

In set associative mapping, each cache location can have more than one pair of tag + data items.

That is more than one pair of tag and data are residing the at same location of cache memory. If one cache location is holding two pair of tag + data items, that is called *2-way set associative mapping.*

Tag	Set	word

2-way Set Associative Mapping



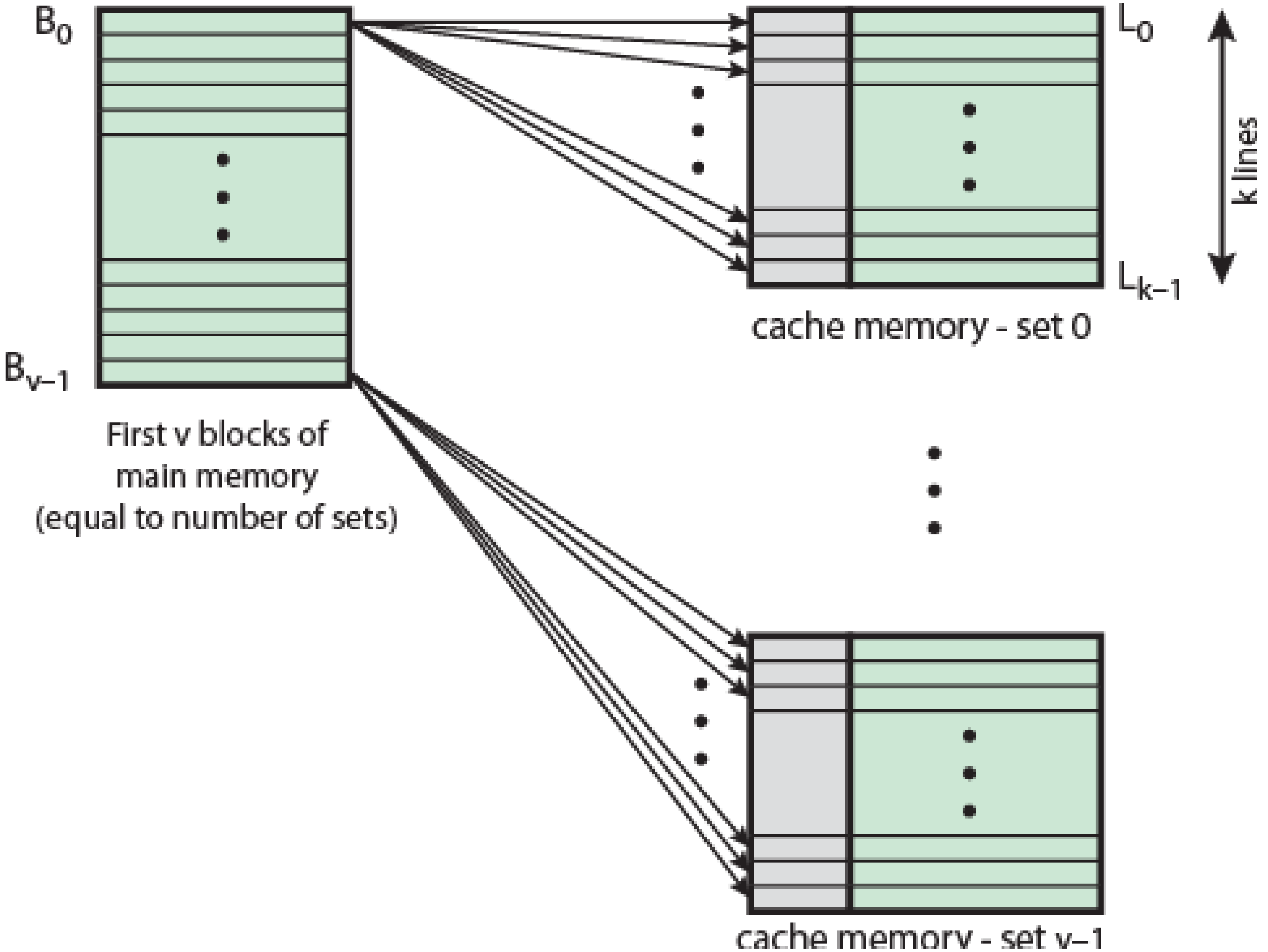
Main Memory

Address	Data
00 000	5670
00 666	7523
01 000	1256
01 666	5321
67 125	7432
77 777	5423

Cache Memory

Tag	Data	Index	Tag	Data
00	5670	000	01	1256
00	7523	666	01	5321
03	2771		02	6520
51	1560	677	41	2560
77	5423	777	66	4423

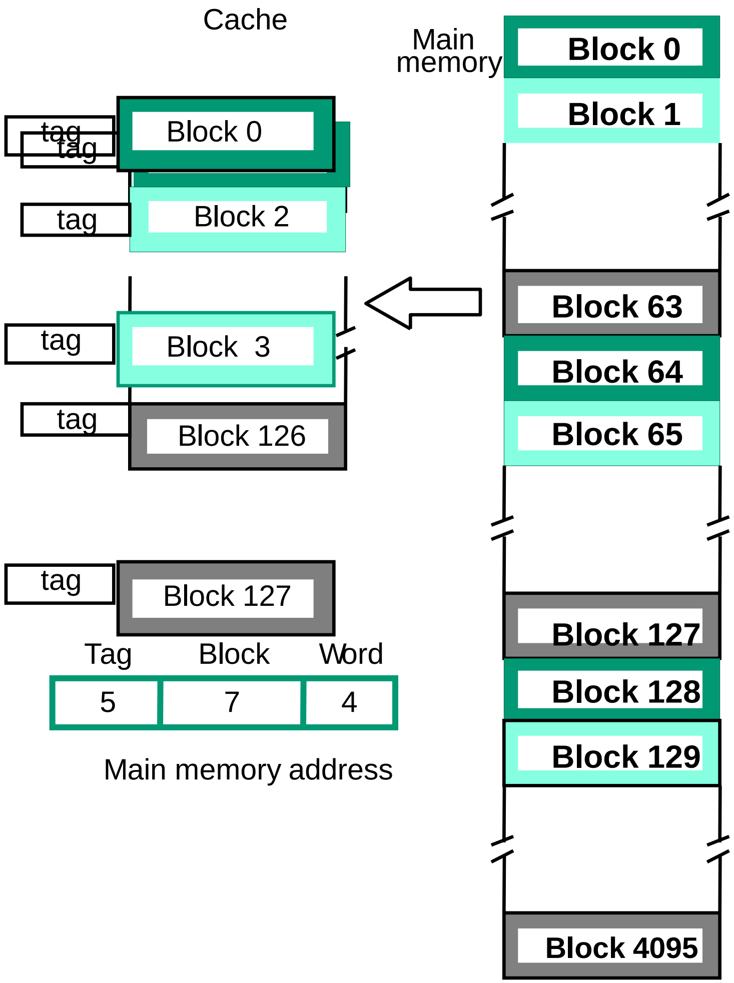
Set Associative



3.Set Associative Mapping

- ❑ Compromise between fully-associative and direct-mapped cache
  - ◆ Cache is divided into a number of sets
  - ◆ Each set contains a number of lines
  - ◆ A given block maps to any line in a specific set
    - Use direct-mapping to determine which set in the cache corresponds to a set in memory
    - Memory block could then be in any line of that set
  - ◆ e.g. 2 lines per set
    - 2 way associative mapping
    - A given block can be in either of 2 lines in a specific set
  - ◆ e.g. K lines per set
    - K way associative mapping
    - A given block can be in one of K lines in a specific set
    - Much easier to simultaneously search one set than all lines

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

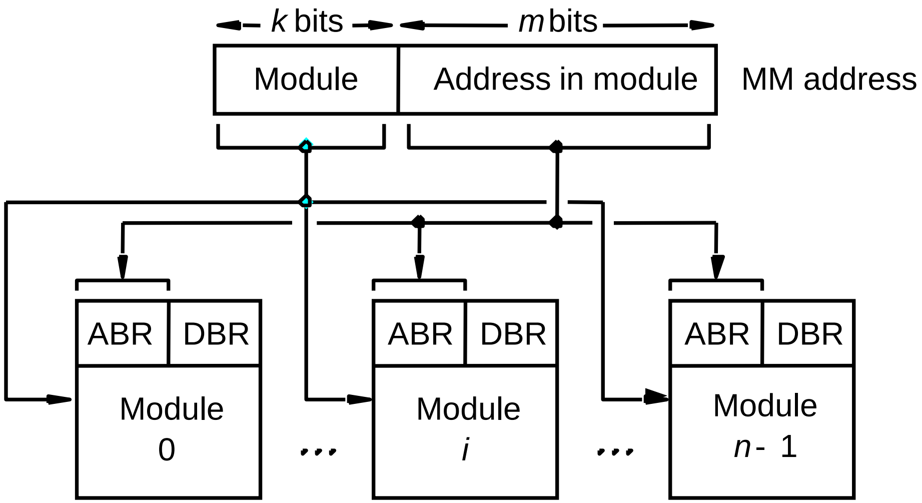
Performance considerations

- 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.

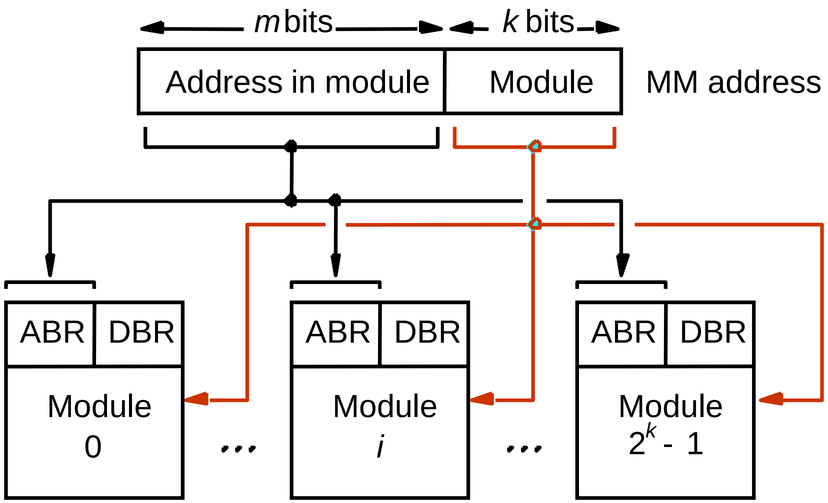
Interleaving

- 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

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

Writing into the cache

When memory write operations are performed, CPU first writes into the cache memory. These modifications made by CPU during a write operations, on the data saved in cache, need to be written back to main memory or to auxiliary memory.

The two popular cache write policies are:

Write –through and Write back

Other Performance Enhancements

- 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 due to a miss.
- 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.
- Dirty bit is set when we write to the cache, this indicates the cache is now inconsistent with main memory.
- Dirty bit for cache slot is cleared when update occurs.

Main advantages:
Write- Back vs Write- Through

- Write- Back:
  - The block can be written by the processor at the frequency at which the cache, and not the main memory, can accept it.
  - Multiple writes to the same block require only a single write to the main memory.
- Write- Through:
  - Simpler to be implemented, but to be effective it requires a **w rite buffer** to do , not to wait for the lower level of the memory hierarchy (to avoid write stalls)
  - The read misses are cheaper because they do not require any write to the lower level of the memory hierarchy
  - Memory always up to date.

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.

Cache replacement policies

Cache algorithms (Page replacement policies)

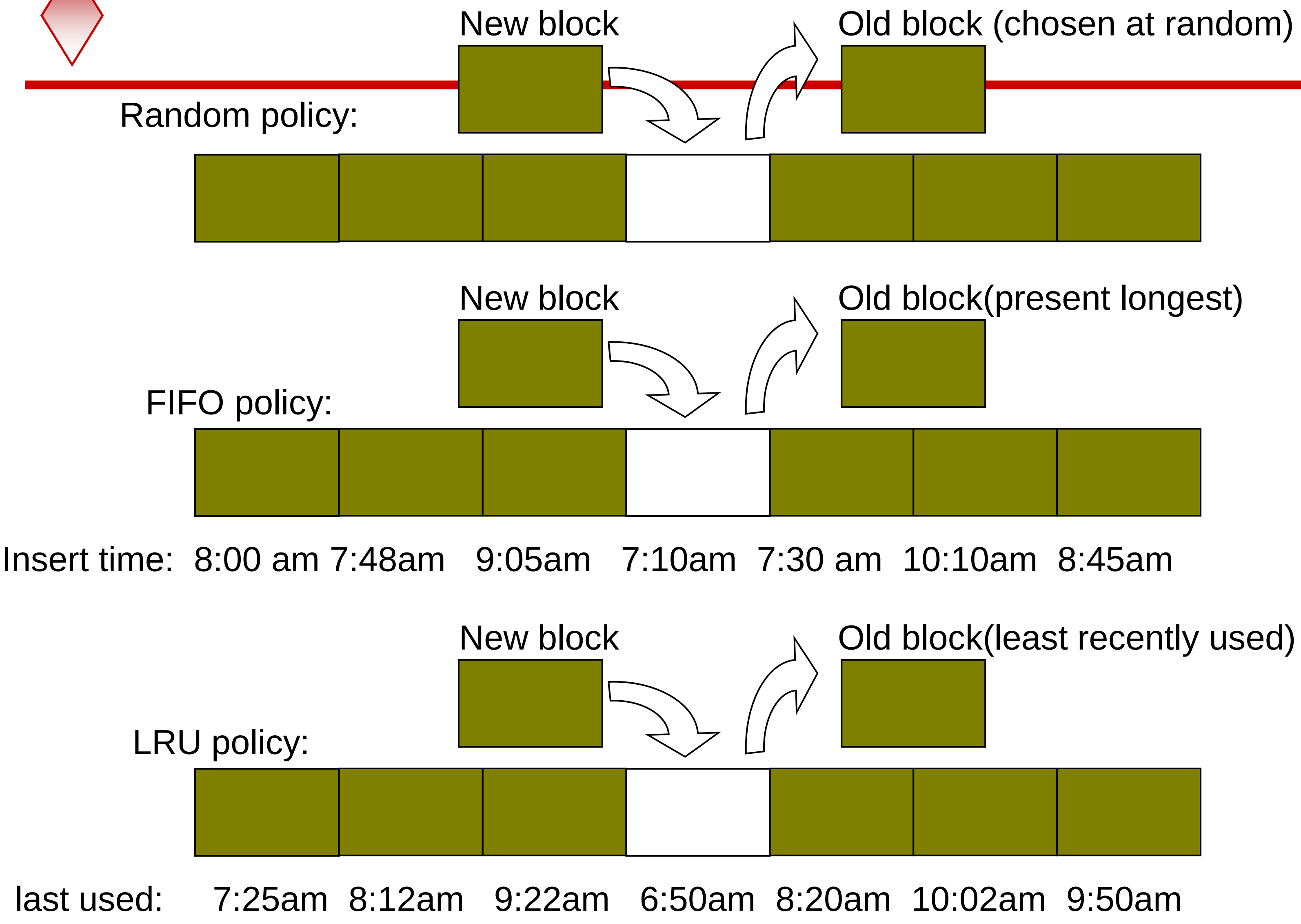
❑ Replacement algorithms are used when there are no available space in a cache in which to place a data. Four of the most common cache replacement algorithms are described

❑ First-in, First-out(FIFO): Evict the page that has been in the cache the longest time.

❑ Least recently used (LRU): Evict the page whose last request occurred furthest in the past.(least recently used page)

❑ Least Frequently Used (LRU): The LRU algorithm selects for replacement the item that has been least frequently used by the CPU.

❑ Random: Choose a page at random to evict from the cache.



Random policy:

New block

Old block (chosen at random)

FIFO policy:

New block

Old block(present longest)

Insert time: 8:00 am 7:48am 9:05am 7:10am 7:30 am 10:10am 8:45am

LRU policy:

New block

Old block(least recently used)

last used: 7:25am 8:12am 9:22am 6:50am 8:20am 10:02am 9:50am

The Random,FIFO,and LRU page replacement policies

Virtual Memory

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.

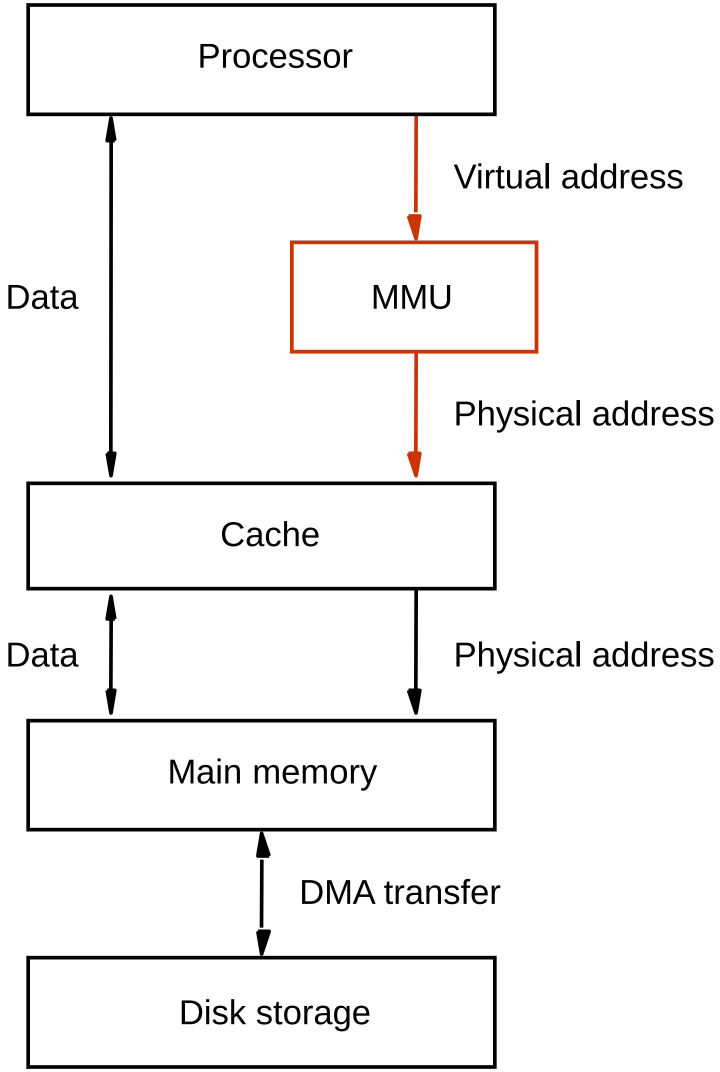
Virtual memories (contd..)

- 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..)

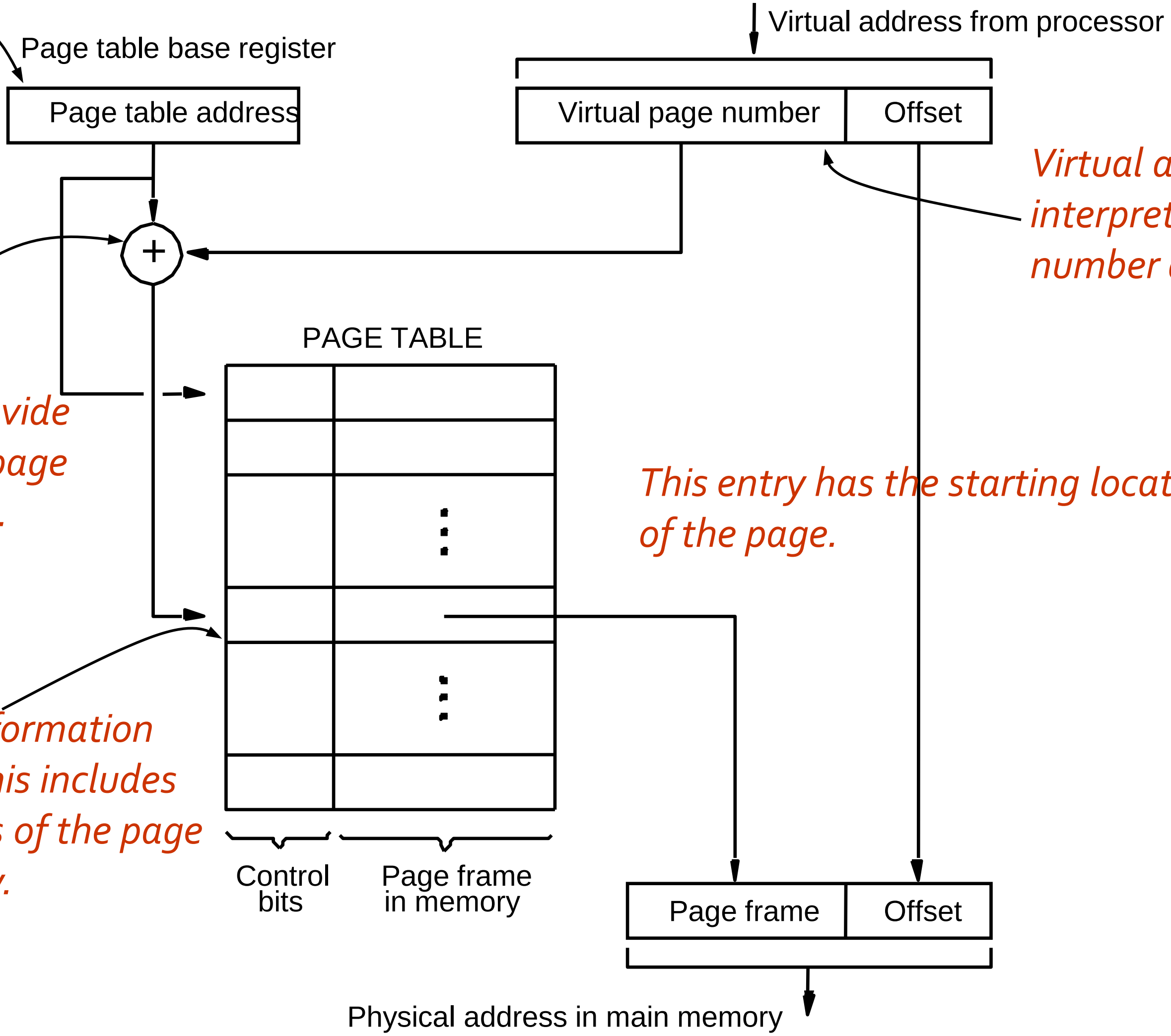
- Each virtual or logical address generated by a processor is interpreted as a virtual page number (high-order bits) plus an offset (low-order bits) that specifies the location of a particular byte within that page.
- Information about the main memory location of each page is kept in the page table.
  - 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..)

PTBR holds the address of the page table.



Virtual address is interpreted as page number and offset.

PTBR + virtual page number provide the entry of the page in the page table.

This entry has the starting location of the page.

Page table holds information about each page. This includes the starting address of the page in the main memory.

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.

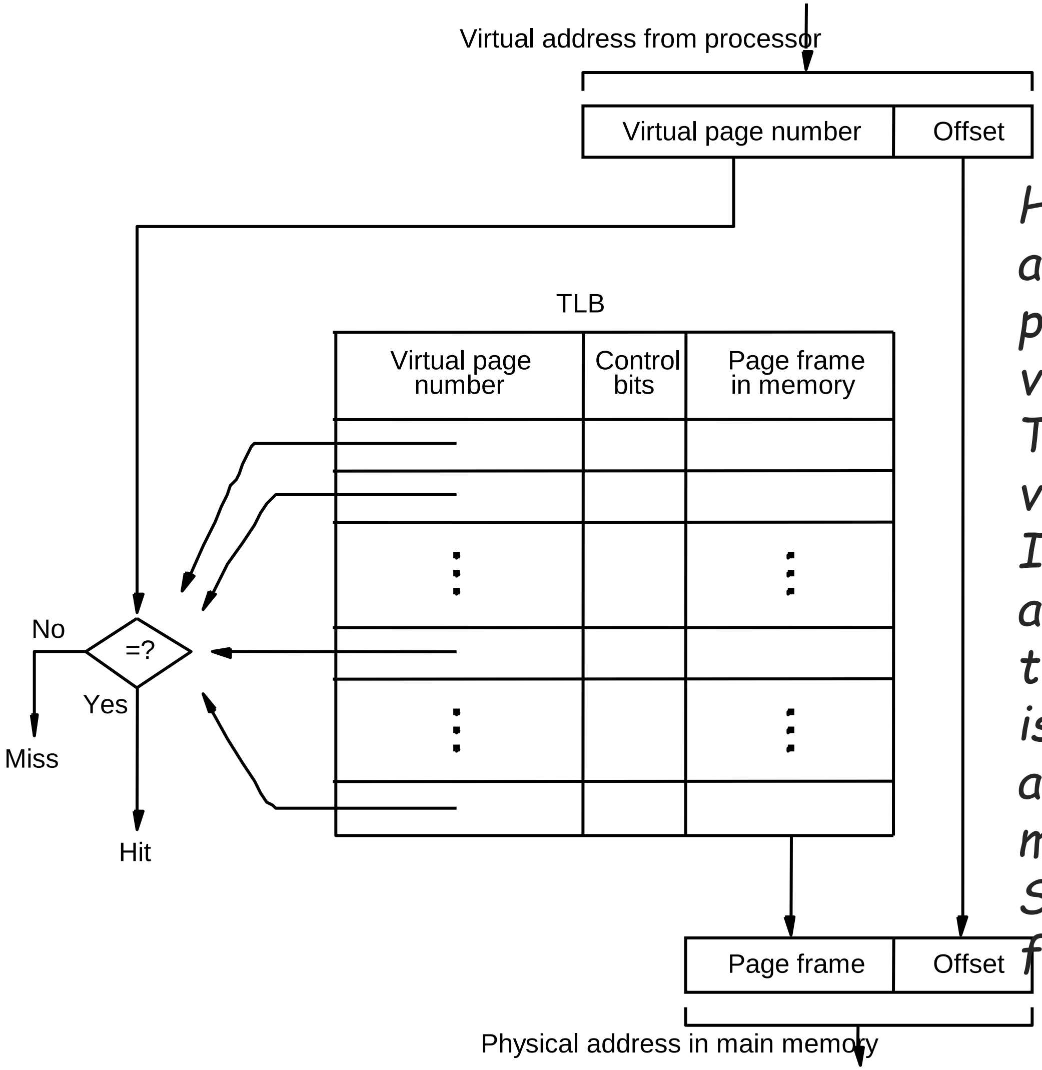
Address translation (contd..)

- 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 Look aside 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

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..)

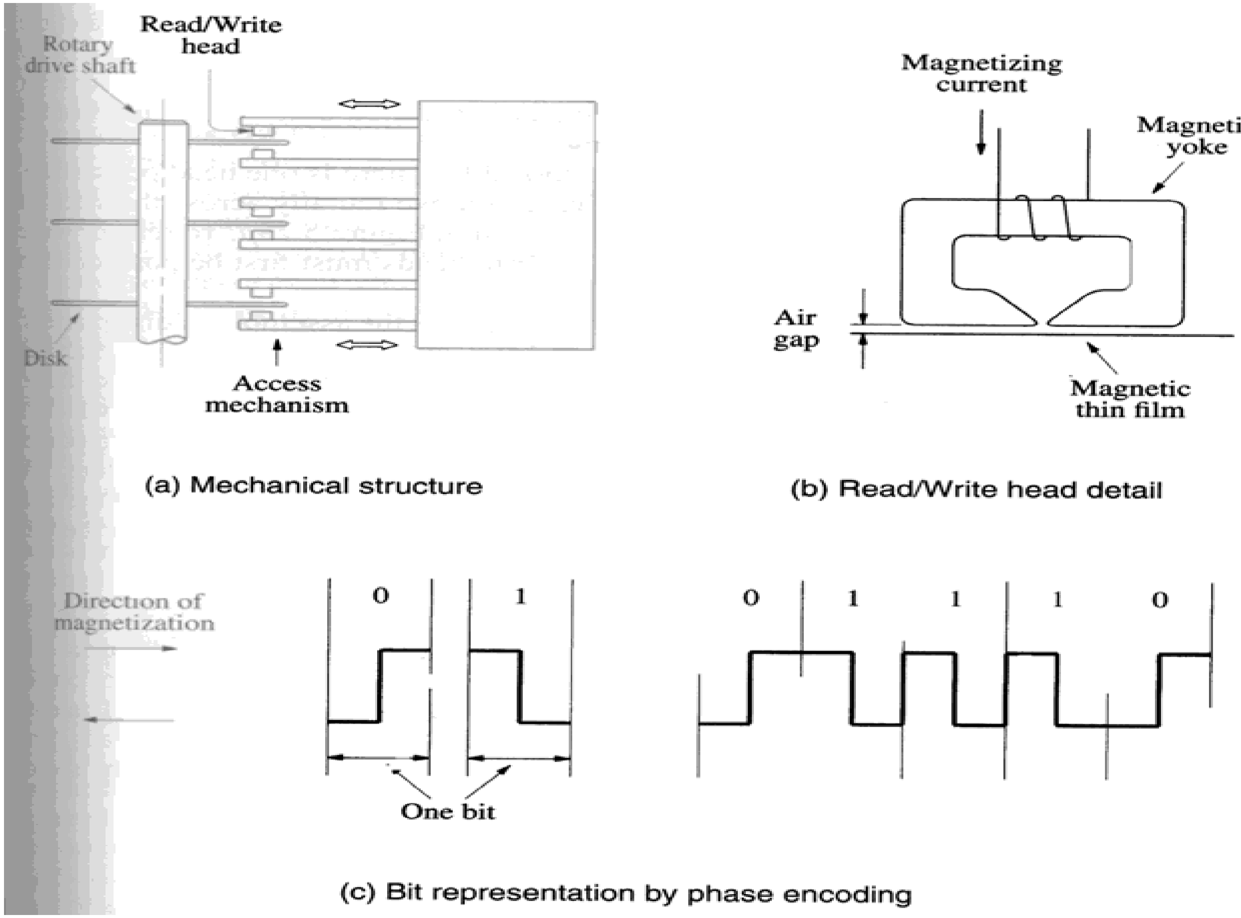
- Recall that the Memory Management Unit (MMU) translates logical or virtual addresses into physical addresses.
- MMU uses the contents of the page table base register to determine the address of the page table to be used in the translation.
    - Changing the contents of the page table base register 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

Magnetic Hard Disks



(a) Mechanical structure

(b) Read/Write head detail

(c) Bit representation by phase encoding

Disk

Disk drive

Disk controller

Organization of Data on a Disk

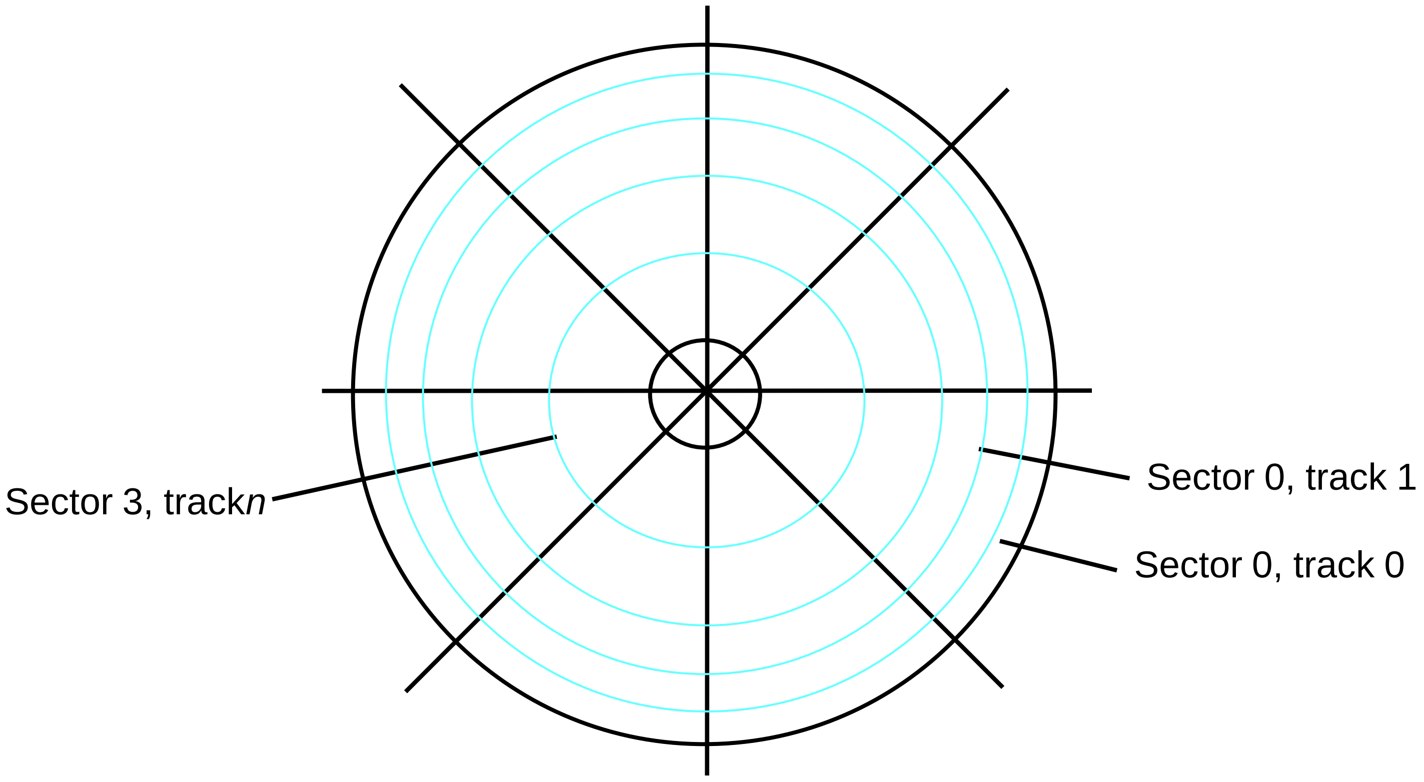


Fig.Organization of one surface of a disk.

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

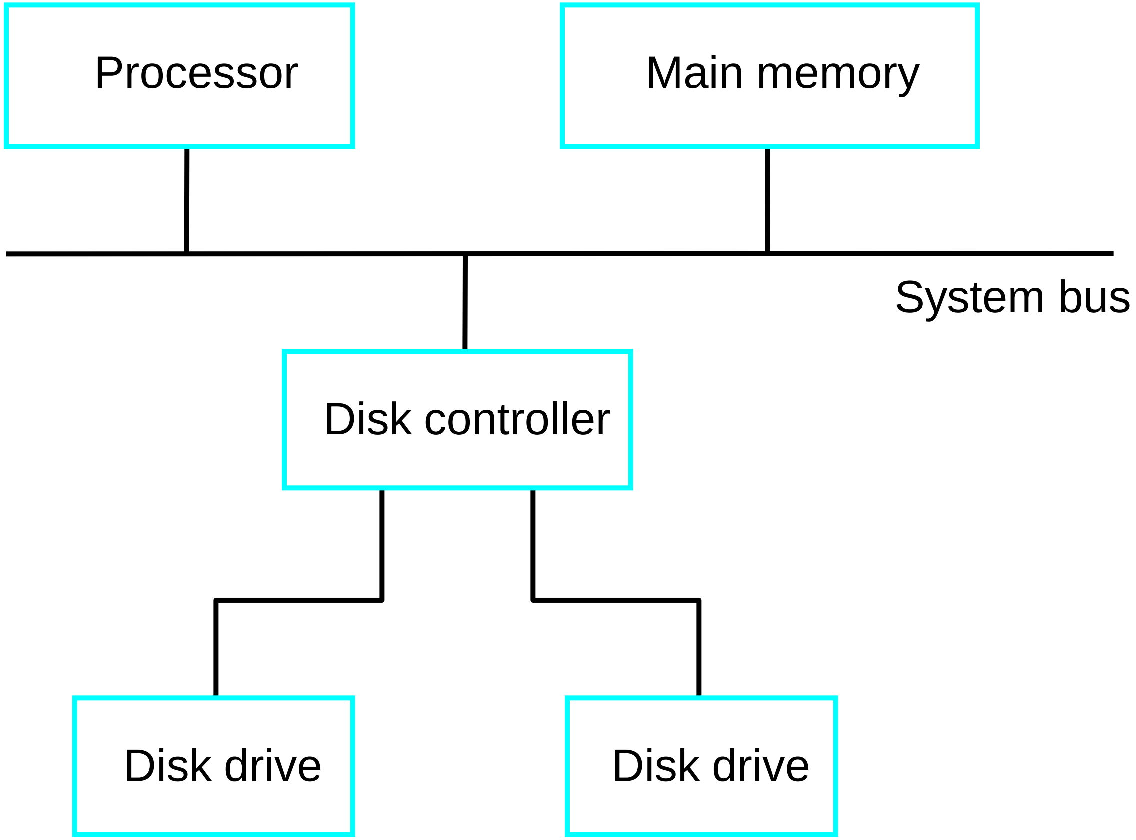


Fig. Disks connected to the system bus.

Disk Controller

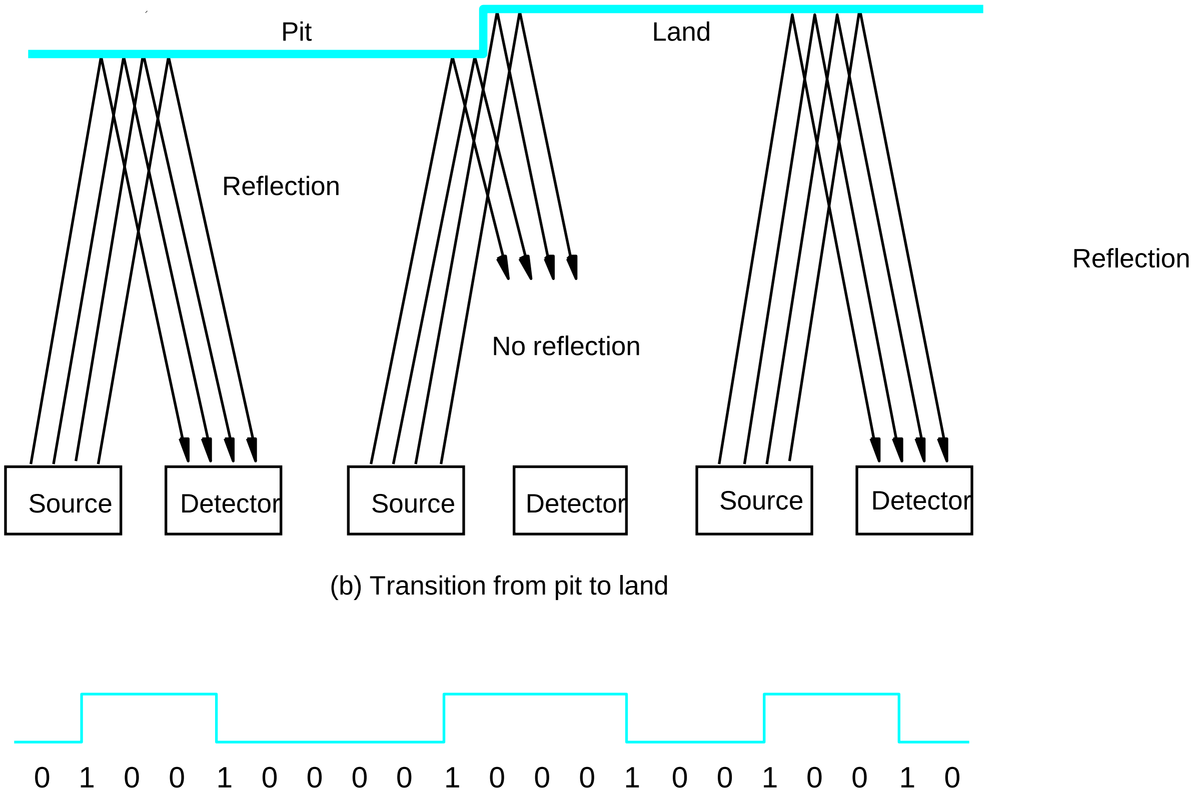
- ✓ Seek
- ✓ Read
- ✓ Write
- ✓ Error checking

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

Fig. Optical disk.

Optical Disks

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

Magnetic Tape Systems

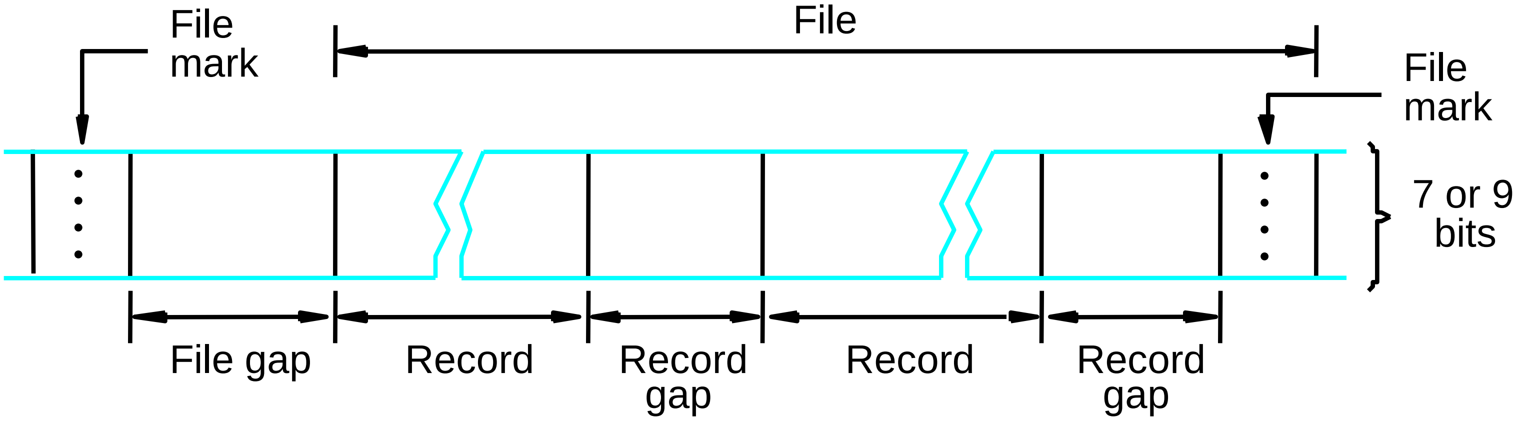


Fig.Organization of data on magnetic tape.