Prepared by Dr K Badari Nath, Associate Professor,
Computer Science & Engineering, RV College Of Engineering

FCSD – Foundations Of Computer System Design

Unit 5 : Memory System

The Memory System: Basic Concepts, Semiconductor RAM Memories, Cache Memories- Mapping Functions, Examples of Mapping Techniques, Performance Considerations.

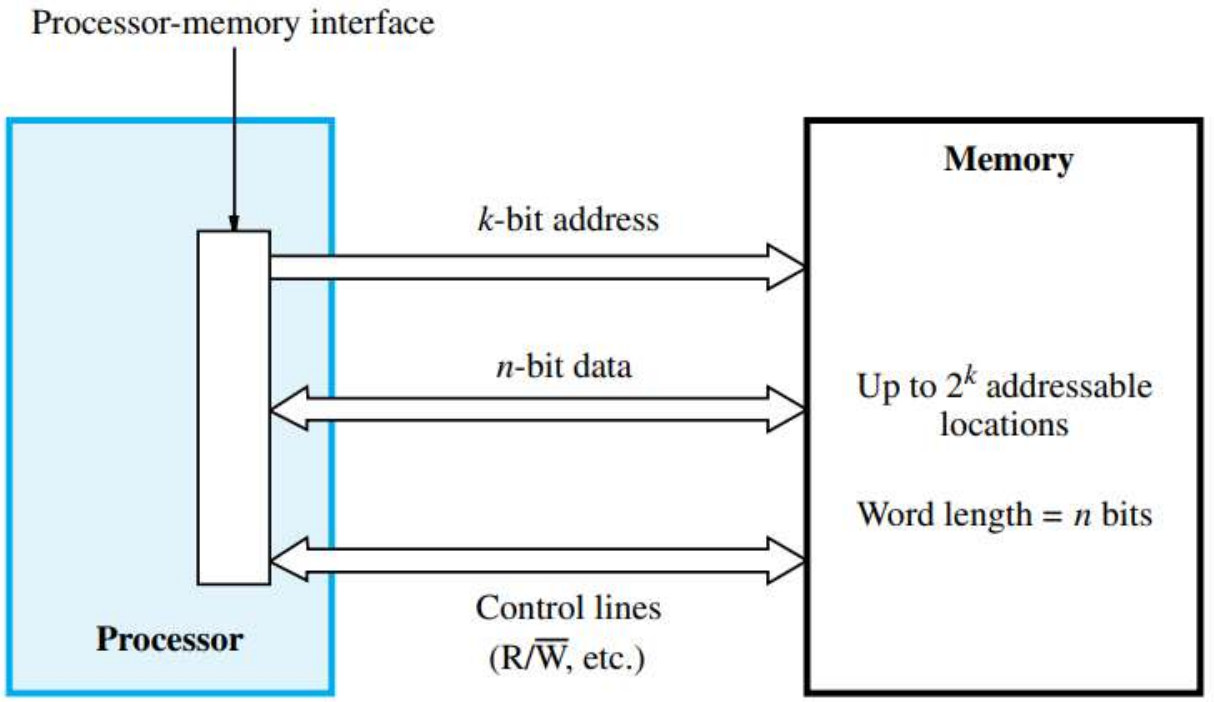
- CIE/SEE Question solutions are included

- Reference: Chapter – The Memory System
  (Book:"Computer Organization and Embedded Systems" by Carl Hamacher)

*https://www.youtube.com/@drkbadarinath4636
(for lecture videos)

Processor & Memory Interface

The maximum size of the memory that can be used in any computer is determined by the addressing scheme. For example, a computer that generates 16-bit addresses is capable of addressing up to 2power16 = 64K (kilo) memory locations. Machines whose instructions generate 32-bit addresses can utilize a memory that contains up to 2power32 = 4G (giga) locations, whereas machines with 64-bit addresses can access up to 2power64 = 16E (exa) ≈ 16 × 1018 locations. The number of locations represents the size of the address space of the computer.



Connection of the memory to the processor.

The connection between the processor and its memory consists of address, data, and control lines. The processor uses the address lines to specify the memory location involved in a data transfer operation, and uses the data lines to transfer the data. At the same time, the control lines carry the command indicating a Read or a Write operation and whether a byte or a word is to be transferred. The control lines also provide the necessary timing information and are used by the memory to indicate when it has completed the requested operation.

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. It also includes control lines like Read, Write and Memory Function Complete (MFC) for coordinating data transfer.

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:

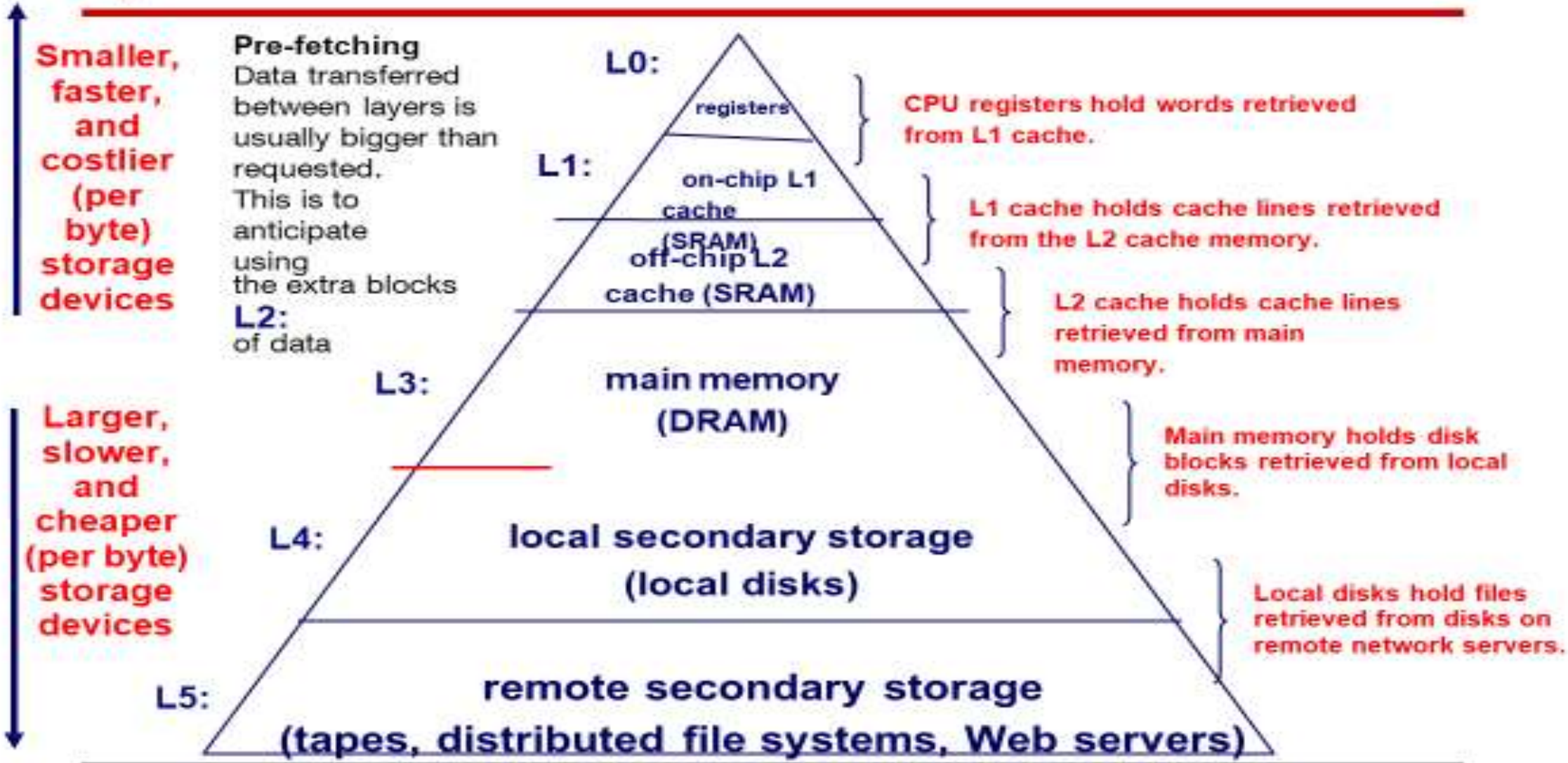
- Cache memory (to increase the effective speed).
- Virtual memory (to increase the effective size).

The 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.

An Example Memory Hierarchy



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.

Memory cells are usually organized in the form of an array, in which each cell is capable of storing one bit of information. A possible organization is illustrated in Figure. Each row of cells constitutes a memory word, and all cells of a row are connected to a common line referred to as the word line, which is driven by the address decoder on the chip. The cells in each column are connected to a Sense/Write circuit by two bit lines, and the Sense/Write circuits are connected to the data input/output lines of the chip. During a Read operation, these circuits sense, or read, the information stored in the cells selected by a word line and place this information on the output data lines. During a Write operation, the Sense/Write circuits receive input data and store them in the cells of the selected word. (refer the figure in the next slide)

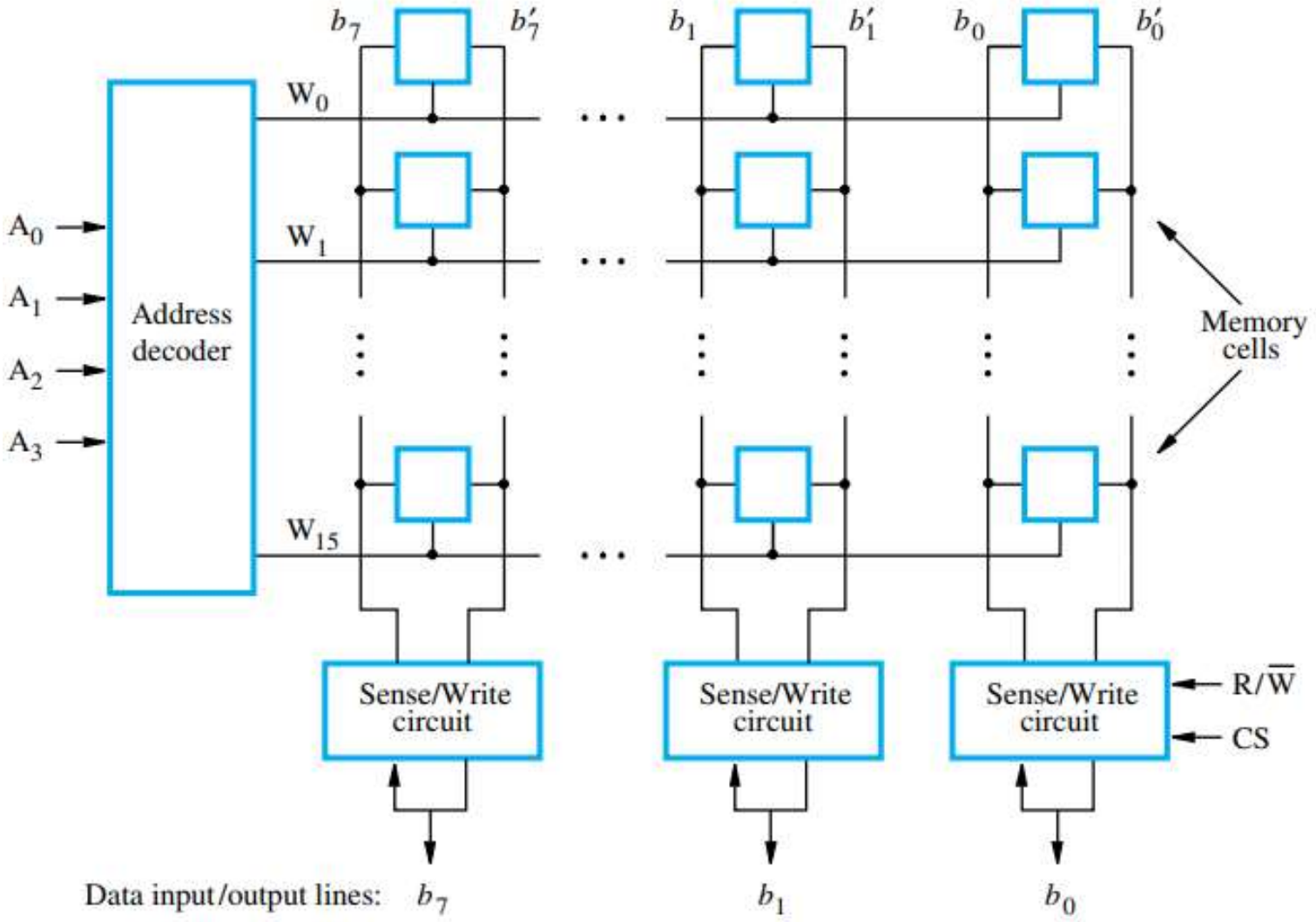
The storage part is modelled here with SR-latch, but in reality it is an electronics circuit made up of transistors. The memory constructed with the help of transistors is known as semiconductor memory. The semiconductor memories are termed as Random Access Memory(RAM), because it is possible to access any memory location in random.

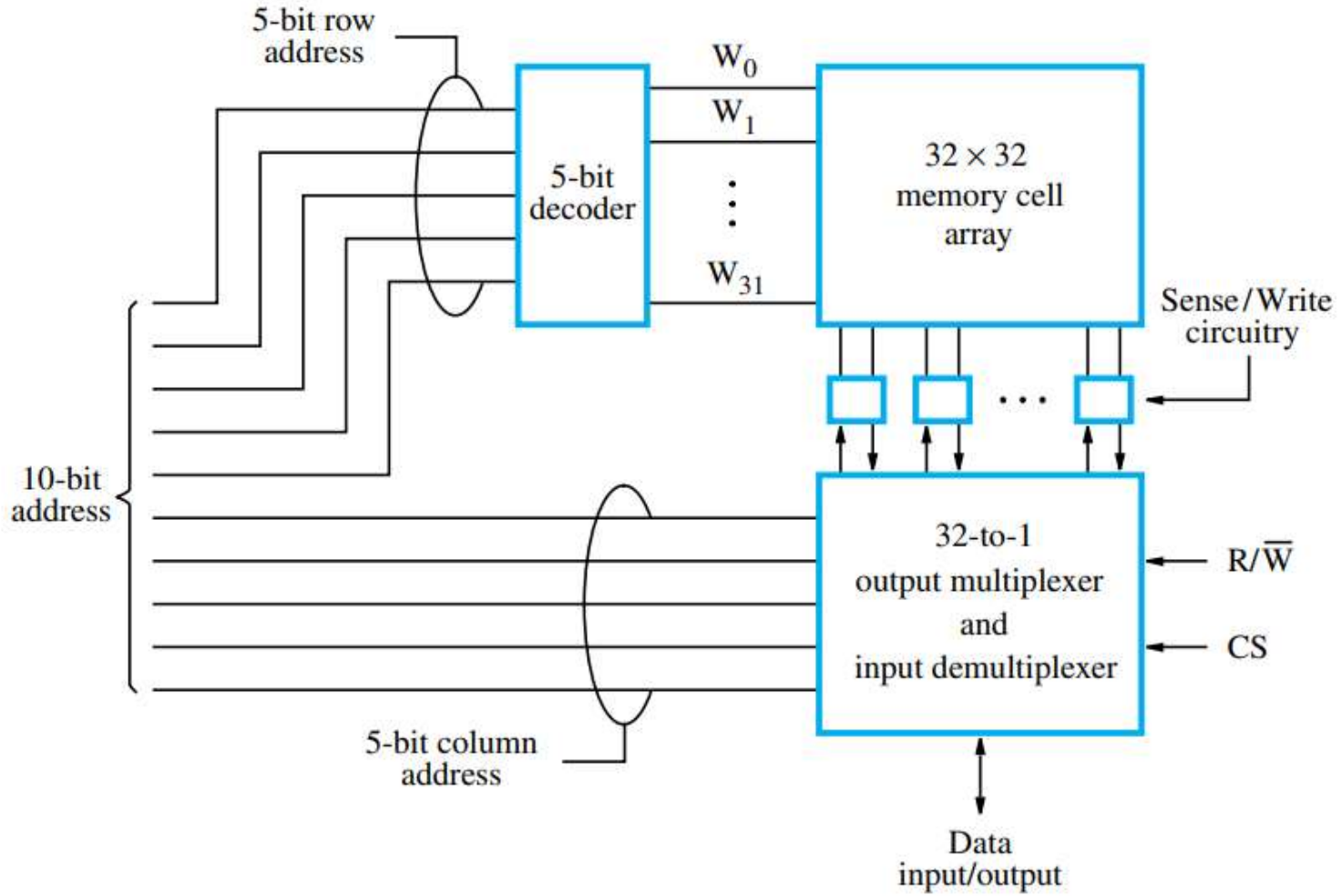
Depending on the technology used to construct a RAM, there are two types of RAM :

SRAM: Static Random Access Memory. **DRAM:** Dynamic Random Access Memory.

Organization of bit cells in a memory chip

Circuit in Figure is an example of a very small memory circuit consisting of 16 words of 8 bits each. This is referred to as a 16 × 8 organization. The data input and the data output of each Sense/Write circuit are connected to a single bidirectional data line that can be connected to the data lines of a computer. Two control lines, R/W and CS, are provided. The R/W (Read/Write) input specifies the required operation, and the CS (Chip Select) input selects a given chip in a multichip memory system. The memory circuit in Figure stores 128 bits and requires 14 external connections for address, data, and control lines. It also needs two lines for power supply and ground connections. Consider now a slightly larger memory circuit, one that has 1K (1024) memory cells. This circuit can be organized as a 128 × 8 memory, requiring a total of 19 external connections.

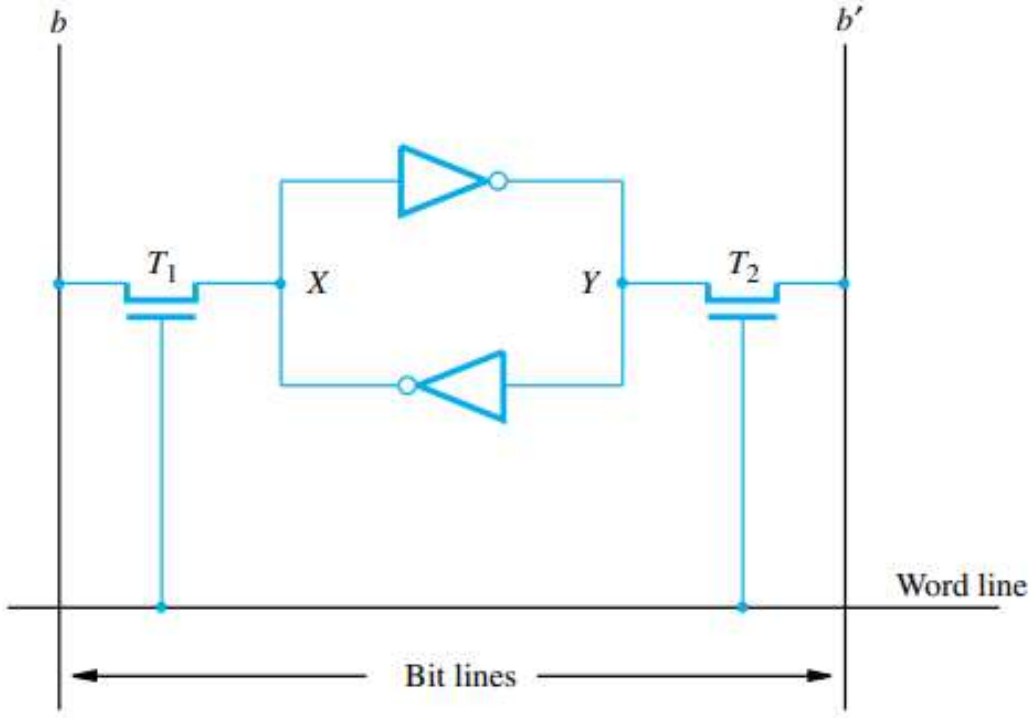




Organization of a 1K × 1 memory chip.

In this case, a 10-bit address is needed, but there is only one data line, resulting in 15 external connections. The required 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. But, only one of these cells is connected to the external data line, based on the column address.

Static Memories :



A static RAM cell.

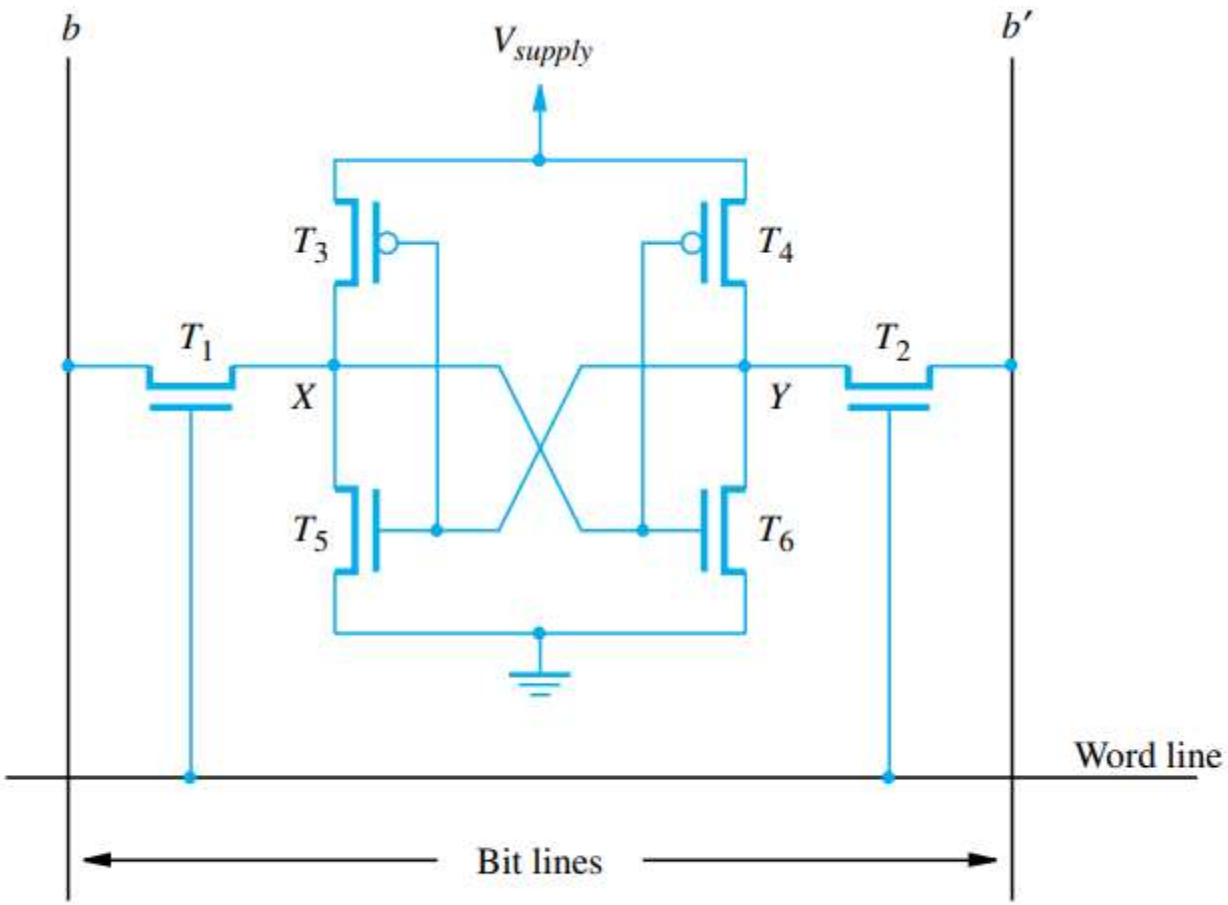
The circuits which are capable of retaining their state as long as power is applied are static memories

Figure illustrates how a static RAM (SRAM) cell may be implemented. Two inverters are cross-connected to form a latch. The latch is connected to two bit lines by transistors T1 and T2. These transistors act as switches that can be opened or closed under control of the word line. When the word line is at ground level, the transistors are turned off and the latch retains its state. For example, if the logic value at point X is 1 and at point Y is 0, this state is maintained as long as the signal on the word line is at ground level. Assume that this state represents the value 1.

Read Operation : In order to read the state of the 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 on bit line b is low. The opposite is true if the cell is in state 0. Thus, b and b are always complements of each other. The Sense/Write circuit at the end of the two bit lines monitors their state and sets the corresponding output accordingly.

Write Operation: During a Write operation, the Sense/Write circuit drives bit lines b and b , instead of sensing their state. It places the appropriate value on bit line b and its complement on b and activates the word line. This forces the cell into the corresponding state, which the cell retains when the word line is deactivated

CMOS Cell



An example of a CMOS memory cell.

A CMOS realization of the cell i is given in Figure. Transistor pairs (T3, T5) and (T4, T6) form the inverters in the latch. For example, in state 1, the voltage at point X is maintained high by having transistors T3 and T6 on, while T4 and T5 are off. If T1 and T2 are turned on, bit lines b and b will have high and low signals, respectively.

Continuous power is needed for the cell to retain its state. If power is interrupted, the cell's contents are lost. When power is restored, the latch settles into a stable state, but not necessarily the same state the cell was in before the interruption. Hence, SRAMs are said to be volatile memories because their contents are lost when power is interrupted.

A major advantage of CMOS SRAMs is their very low power consumption, because current flows in the cell only when the cell is being accessed. Otherwise, T1, T2, and one transistor in each inverter are turned off, ensuring that there is no continuous electrical path between Vsupply and ground. Static RAMs can be accessed very quickly. Access times on the order of a few nanoseconds are found in commercially available chips. SRAMs are used in applications where speed is of critical concern.

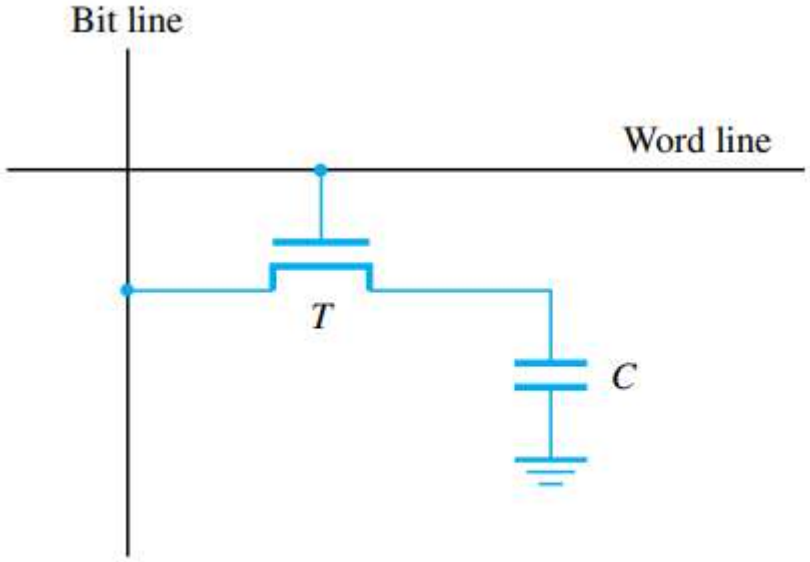
Asynchronous DRAMs vs SRAMs

- 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

- 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

A single-transistor dynamic memory cell

Static RAMs are fast, but their cells require several transistors. Less expensive and higher density RAMs can be implemented with simpler cells. But, these simpler cells do not retain their state for a long period, unless they are accessed frequently for Read or Write operations. Memories that use such cells are called dynamic RAMs (DRAMs).



A single-transistor dynamic memory cell.

Since the cell is required to store information for a much longer time, its contents must be periodically refreshed by restoring the capacitor charge to its full value. This occurs when the contents of the cell are read or when new information is written into it.

A dynamic memory cell consists of a capacitor and a transistor. In order to store Information in this cell , transistor T is turned on and appropriate voltage is applied to the bit line. This caused known amount of charge to be stored in the capacitor.

After the transistor is turned off, the charge remains stored in the capacitor, but not for long. The capacitor begins to discharge. This is because the transistor continues to conduct a tiny amount of current, measured in picoamperes, after it is turned off. Hence, the information stored in the cell can be retrieved correctly only if it is read before the charge in the capacitor drops below some threshold value.

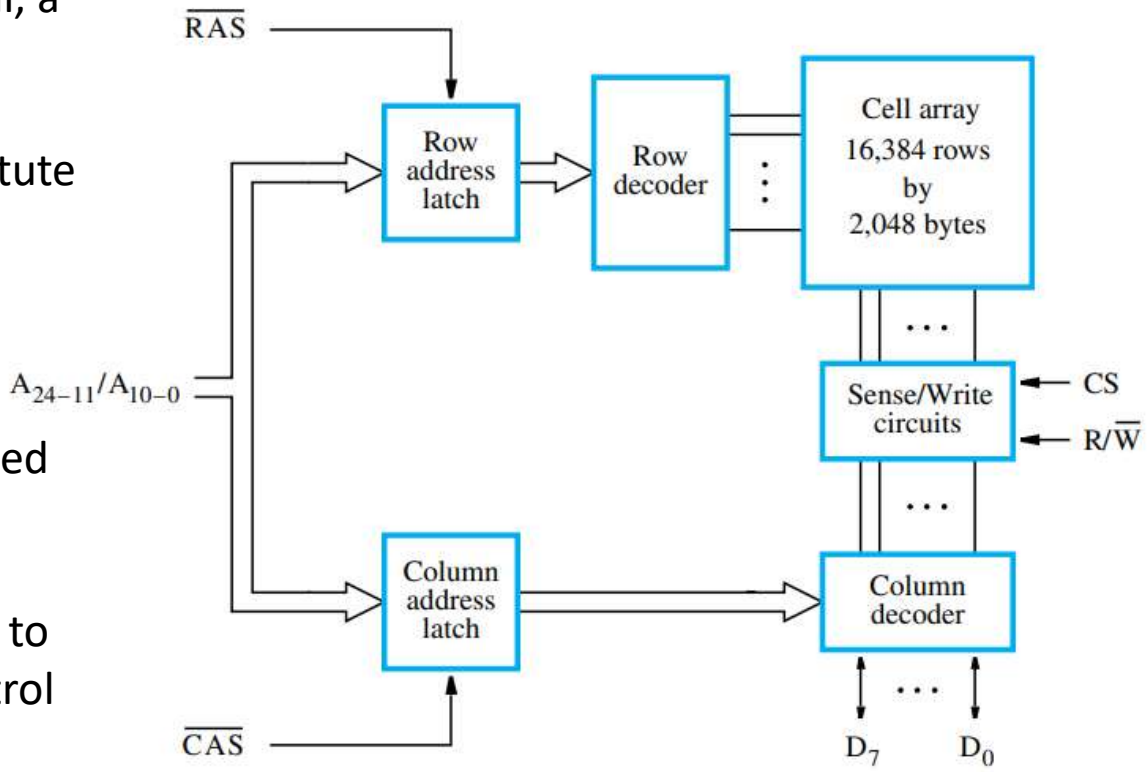
Reading the contents of a cell automatically refreshes its contents. Since the word line is common to all cells in a row, all cells in a selected row are read and refreshed at the same time

The cells are organized in the form of a 16K × 16K array. The 16,384 cells in each row are divided into 2,048 groups of 8, forming 2,048 bytes of data.
Therefore, 14 address bits are needed to select a row, and another 11 bits are needed to specify a group of 8 bits in the selected row. In total, a 25-bit address is needed to access a byte in this memory.

The high-order 14 bits and the low-order 11 bits of the address constitute the row and column addresses of a byte, respectively. To reduce the number of pins needed for external connections, the row and column addresses are multiplexed on 14 pins. During a Read or a Write operation, the row address is applied first. It is loaded into the row address latch in response to a signal pulse on an input control line called the Row Address Strobe (RAS). This causes a Read operation to be initiated, in which all cells in the selected row are read and refreshed. Shortly after the row address is loaded, the column address is applied to the address pins and loaded into the column address latch under control of a second control line called the Column Address Strobe (CAS).

The information in this latch is decoded and the appropriate group of 8 Sense/Write circuits is selected. If the R/W control signal indicates a Read operation, the output values of the selected circuits are transferred to the data lines, D7–0. For a Write operation, the information on the D7–0 lines is transferred to the selected circuits, then used to overwrite the contents of the selected cells in the corresponding 8 columns.

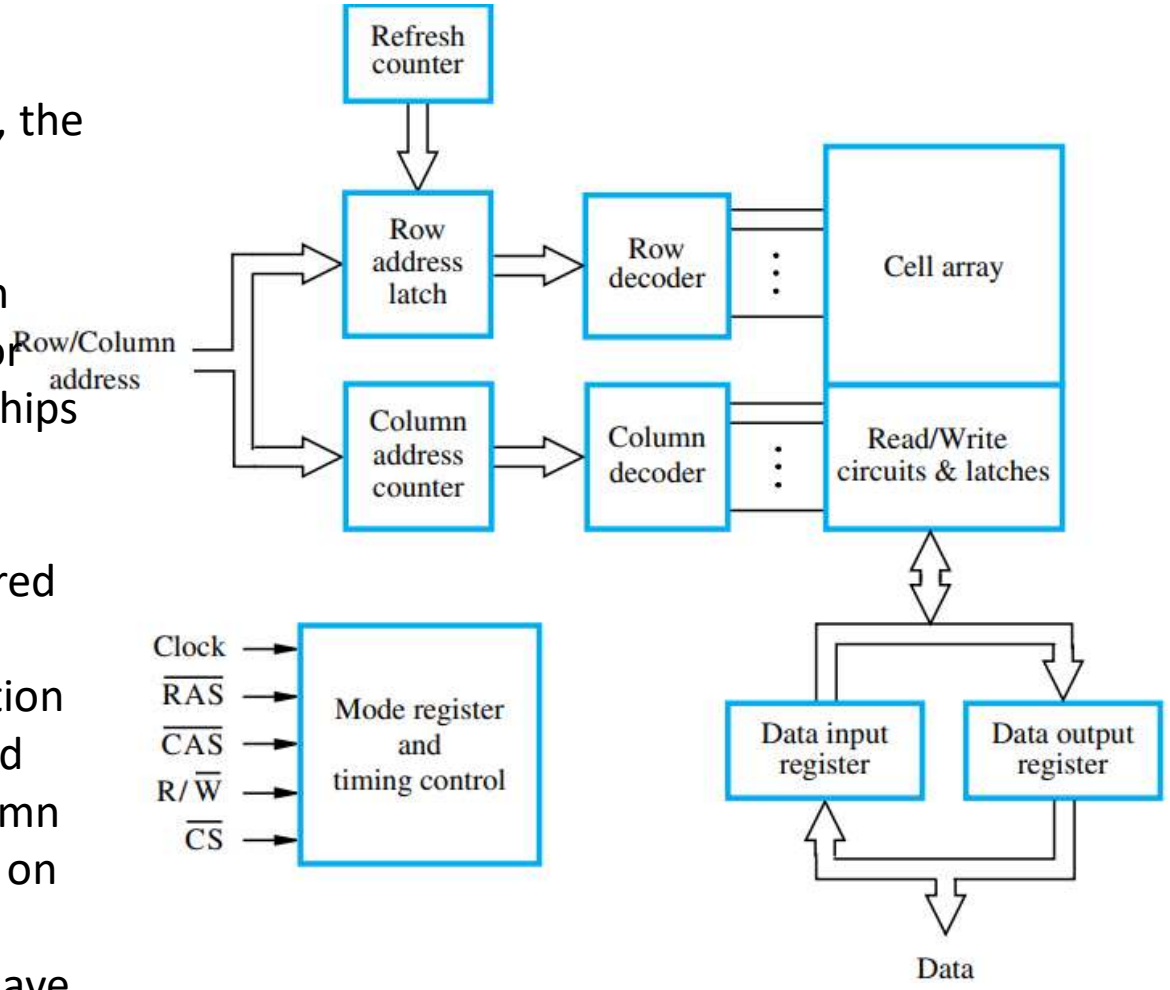
256-Megabit asynchronous DRAM chip, configured as 32M × 8



Internal organization of a 32M × 8 dynamic memory chip.

Synchronous DRAM

In the early 1990s, developments in memory technology resulted in DRAMs whose operation is synchronized with a clock signal. Such memories are known as synchronous DRAMs (SDRAMs). The cell array is the same as in asynchronous DRAMs. The distinguishing feature of an SDRAM is the use of a clock signal, the availability of which makes it possible to incorporate control circuitry on the chip that provides many useful features. For example, SDRAMs have built-in refresh circuitry, with a refresh counter to provide the addresses of the rows to be selected for refreshing. As a result, the dynamic nature of these memory chips is almost invisible to the user



The address and data connections of an SDRAM may be buffered by means of registers. Internally, the Sense/Write amplifiers function as latches, as in asynchronous DRAMs. A Read operation causes the contents of all cells in the selected row to be loaded into these latches. The data in the latches of the selected column are transferred into the data register, thus becoming available on the data output pins. The buffer registers are useful when transferring large blocks of data at very high speed. SDRAMs have several different modes of operation, which can be selected by writing control information into a mode register. For example, burst operations of different lengths can be specified.

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.
- 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

Double-Data-Rate SDRAM

In the continuous quest for improved performance, faster versions of SDRAMs have been developed. In addition to faster circuits, new organizational and operational features make it possible to achieve high data rates during block transfers. The key idea is to take advantage of the fact that a large number of bits are accessed at the same time inside the chip
when a row address is applied.

Various techniques are used to transfer these bits quickly to
the pins of the chip. To make the best use of the available clock speed, data are transferred externally on both the rising and falling edges of the clock. For this reason, memories that use this technique are called double-data-rate SDRAMs (DDR SDRAMs).

Several versions of DDR chips have been developed. The earliest version is known as DDR. Later versions, called DDR2, DDR3, and DDR4, have enhanced capabilities. They offer increased storage capacity, lower power, and faster clock speeds. For example, DDR2 and DDR3 can operate at clock frequencies of 400 and 800 MHz, respectively. Therefore, they transfer data using the effective clock speeds of 800 and 1600 MHz, respectively.

Design a 1024 x 8 RAM using 256 x 8 RAM CHIPS

Number of Chips =

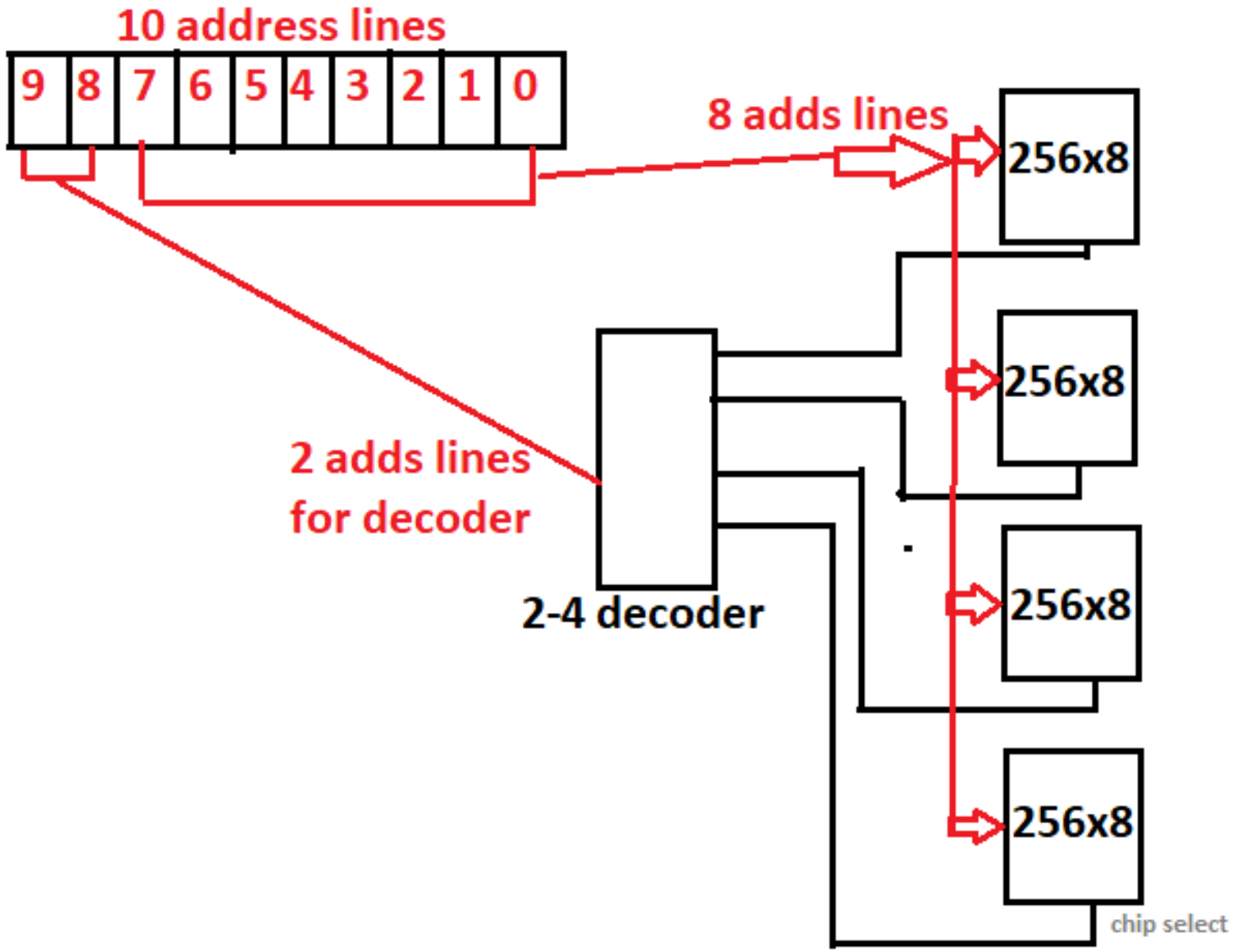
(1024 x 8) / (256x8)

2^10 x 8 / 2^8 x 8

2^2 = 4 chips of 256x8 are required, they are organized as follows, shown in figure ->

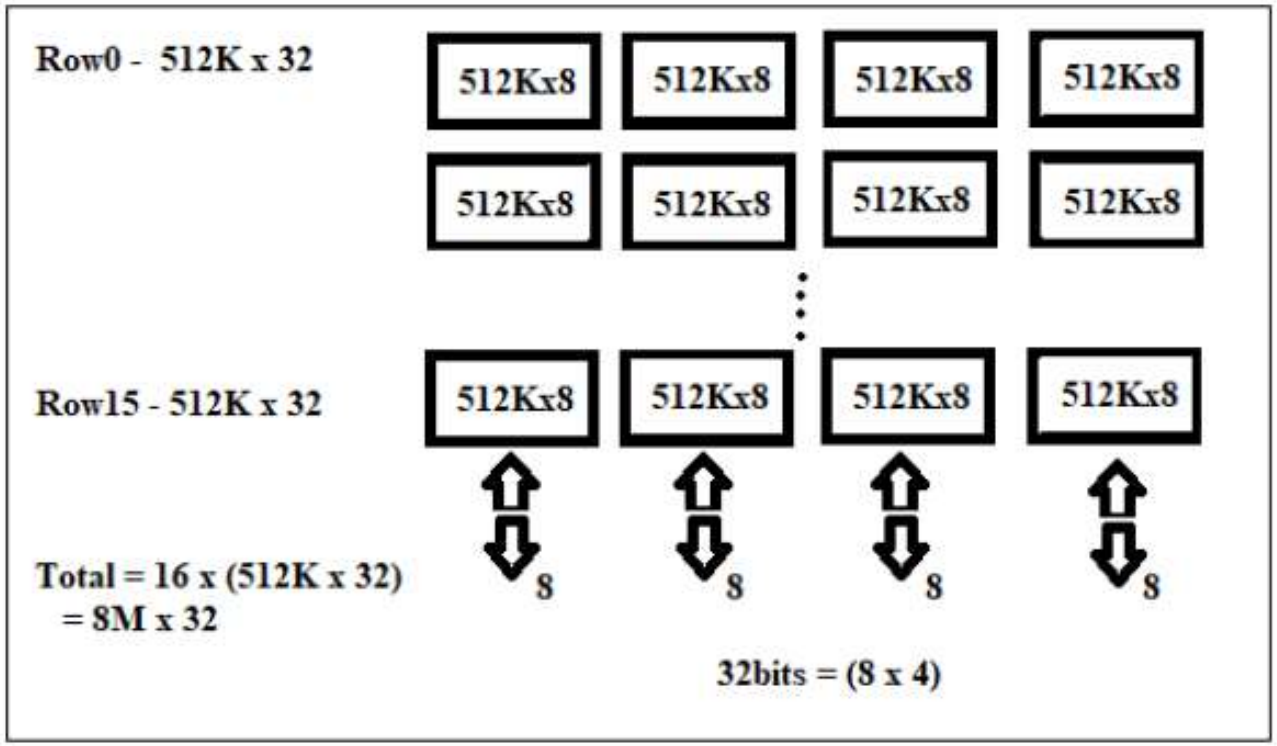
Decoder = 2^2 = 4, hence 2 to 4 line decoder required, to select one of the four chips, at a time.

No. of address lines =10bits, because 2^10 = 1024



Draw the block diagram for 8M x 32 memory using 512K x 8 memory chips. Show the calculations and all the necessary signals.

8 M / 512 K = 16 ROWS
32 / 8 = 4 COLUMNS
HENCE,
WE REQUIRE 16 X 4 =
64 UNITS/CHIPS OF 512 K X 8 MEMORY



For designing 1M x 8 size RAM chip using 256Kx8 RAM chip, how many 256Kx8 RAM chips are required and also specify the number of address bits.

1M – 1024 x 1024, 1k = 1024

1x1024x1024 / 256 x 1024 = 4 chips

No. address bits for each chip = 18 (2^18 = 256K)

No. address bits for full size = 20 (2^20 = 1M)

Intel 8086 Microprocessor supports 1Mbyte memory (byte addressable), indicate how many address lines are provided by the chip.

2^20 = 1Mbyte, Hence Intel 8086 Provides 20 address lines

Design 256x16 RAM Memory using 16x4 RAM chips by showing all the steps and draw the organization of the memory chips.

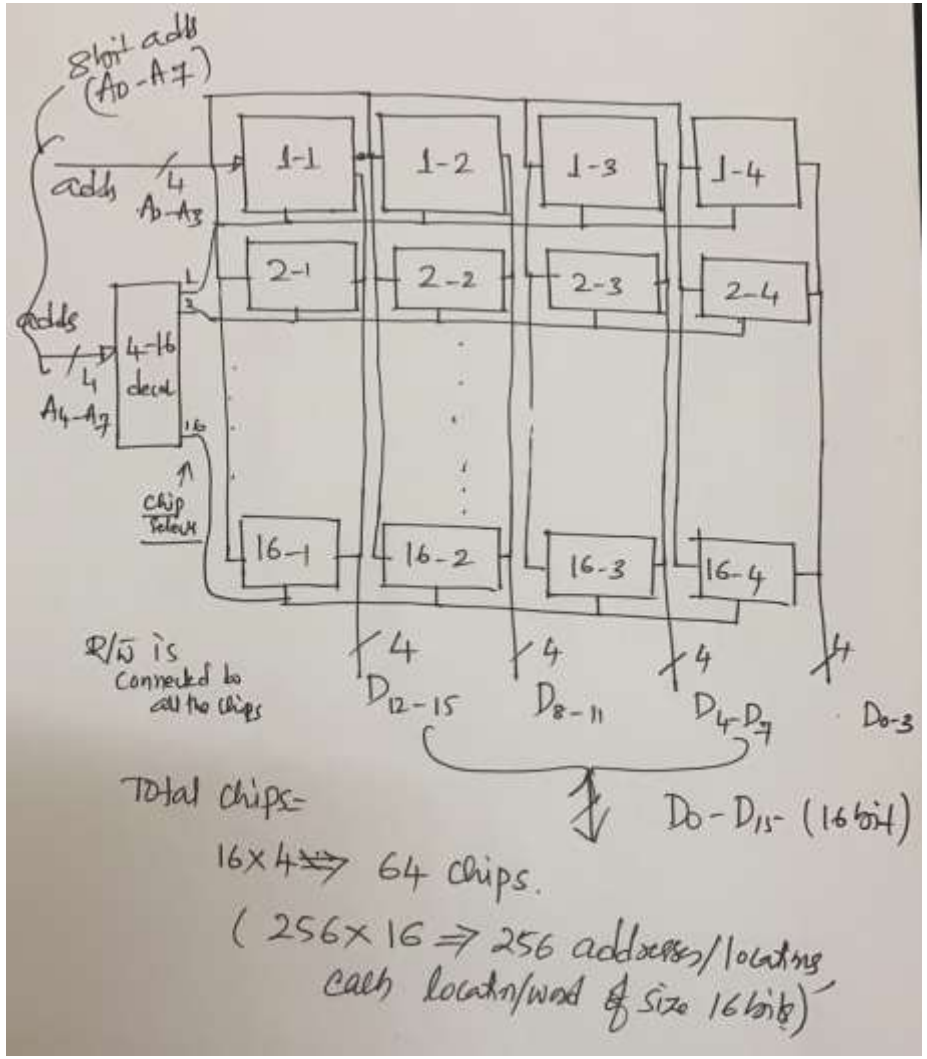
No. of Rows = 256/16 = 16, hence 16 decoder outputs to select one row at a time(hence 4 input lines for decoder)

No. of Columns = 16/4 = 4, each having common address lines, but each chip in a row, contributes 4 data lines, so total=16.

No. of address lines required for individual chip 16x4 = 4

No. of address lines required to be connected to decoder inputs = 4

Total address lines = 4 + 4 =8



Structure of large memories: Static memories

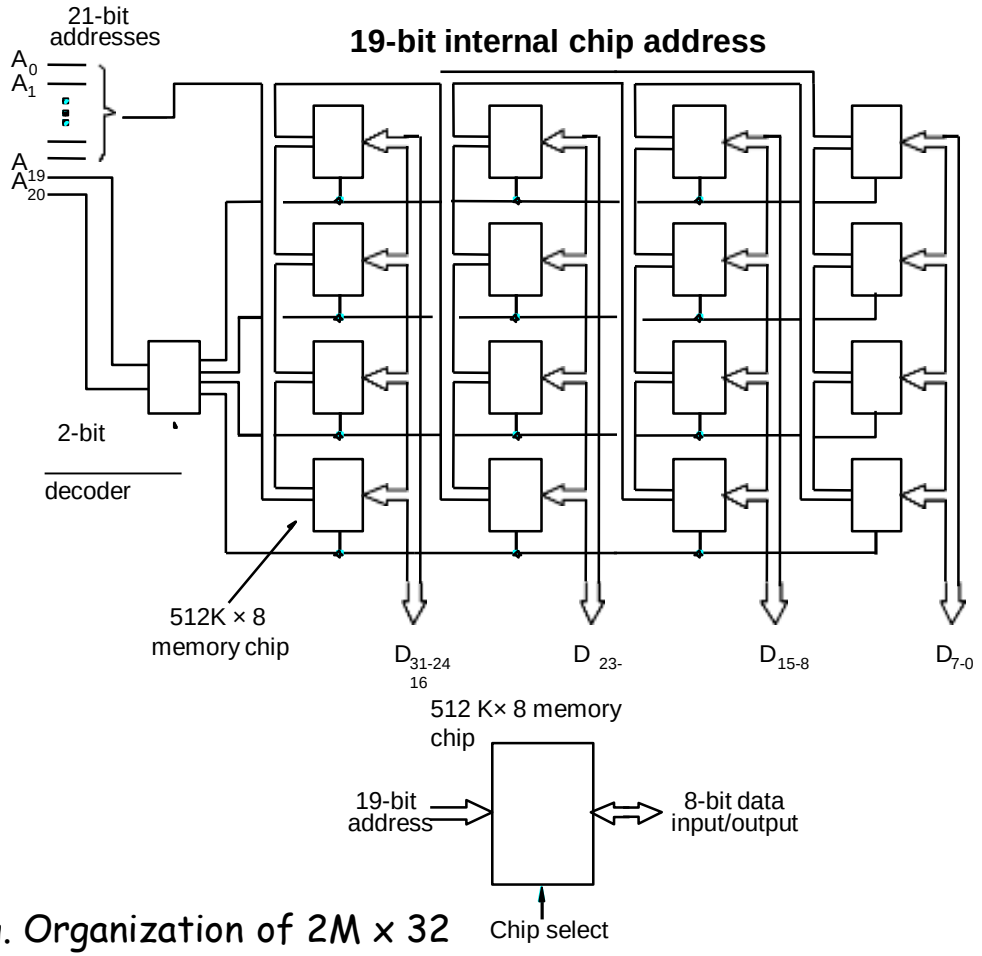


Fig. Organization of 2M x 32

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.

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.

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

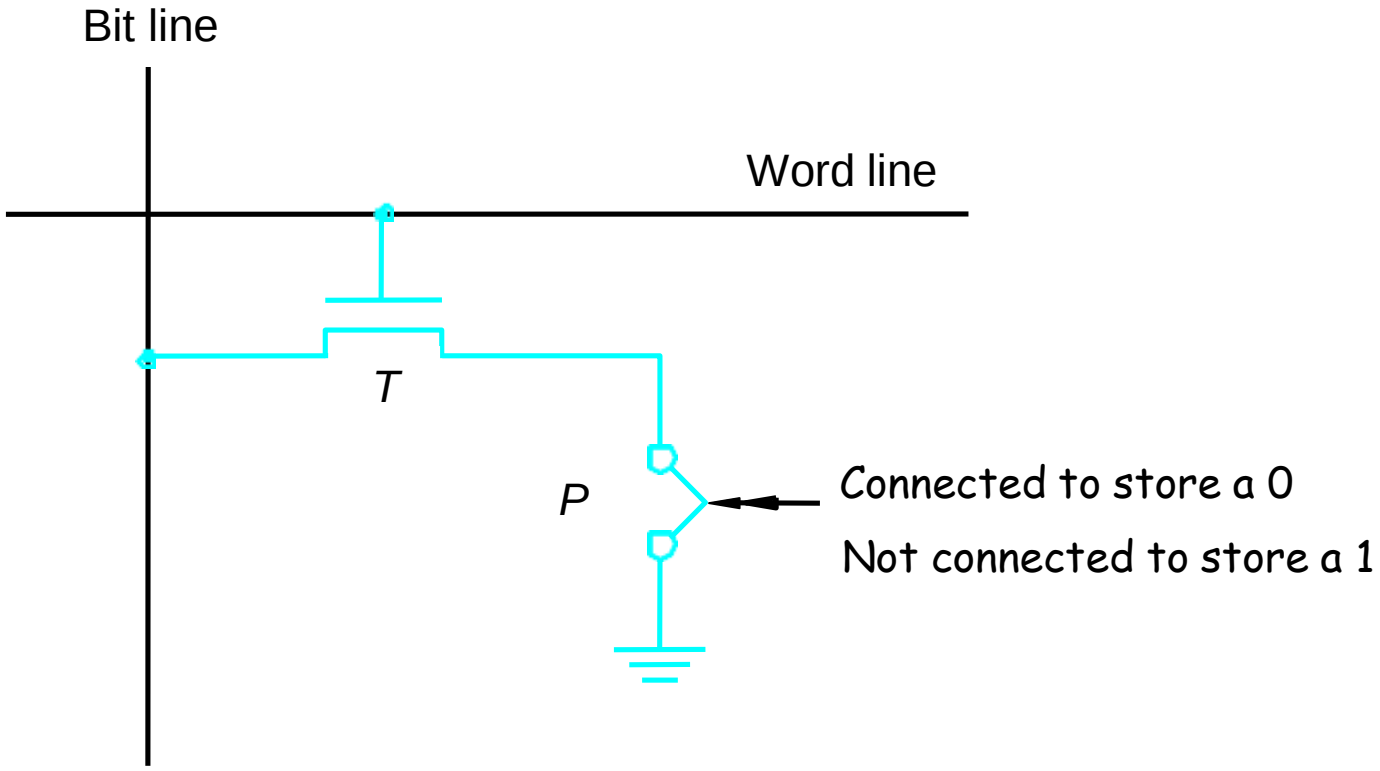


Fig. A ROM cell.

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.

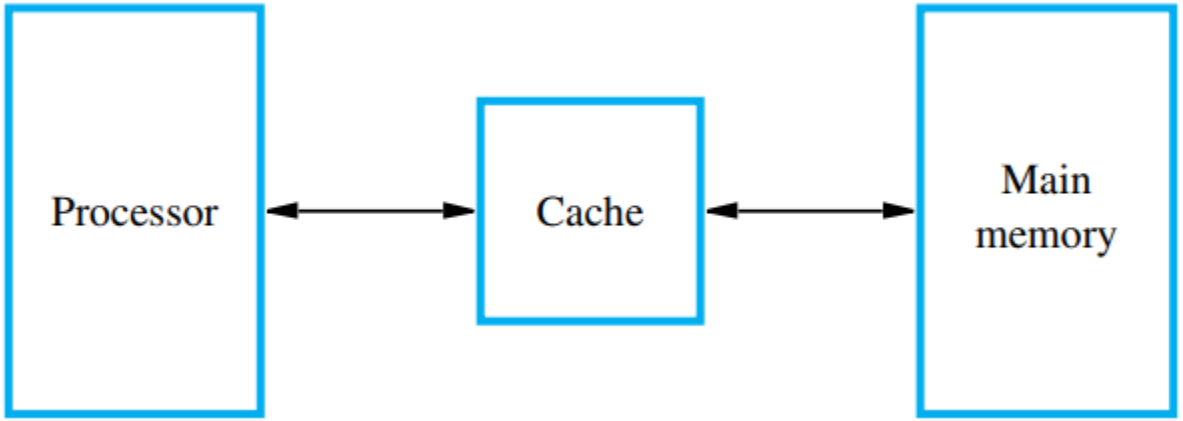
Cache Memory

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. This is Major obstacle towards achieving good performance. Speed of the main memory cannot be increased beyond a certain point, because of cost,size

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".

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". Two type of locality of reference are there,

1. Temporal locality of reference: Recently executed instruction is likely to be executed again very soon.
2. Spatial locality of reference: Instructions with addresses close to a recently instruction are likely to be executed soon.



- 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 Mapping Functions

There are several possible methods for determining where memory blocks are placed in the cache.

1. Direct Mapping
2. Associative Mapping
3. Set Associative Mapping

Let us assume, (for describing different methods, in the following slides)

Cache memory 2K, [128 blocks, 16 words/block]

Main Memory 64K [4024/4K blocks,16words/block]

Note: For simplicity, we have assumed that consecutive addresses refer to consecutive words.

Direct Mapping

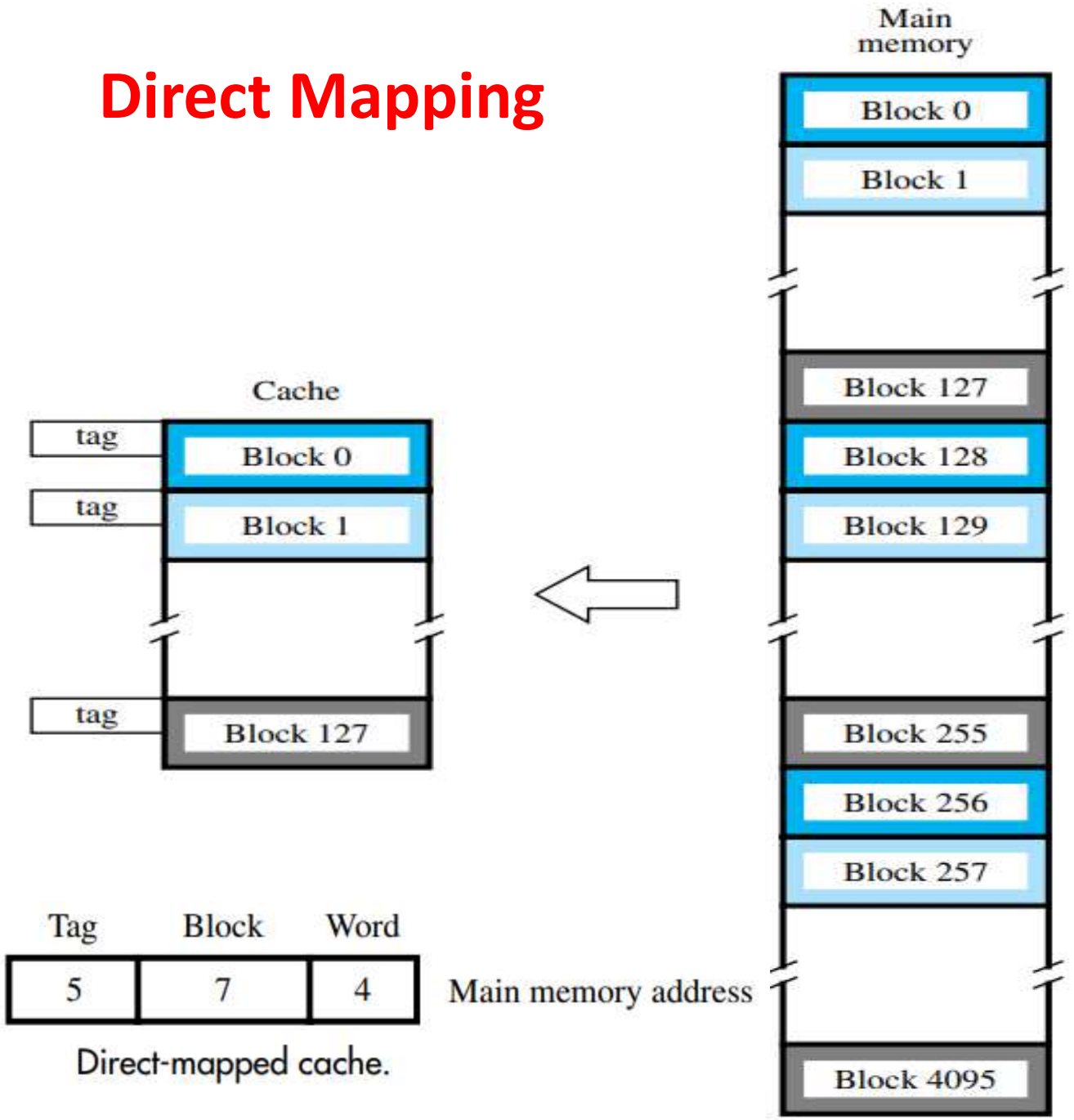
The simplest way to determine cache locations in which to store memory blocks is the direct-mapping technique. In this technique, block j of the main memory maps onto block j modulo 128 of the cache, as depicted in Figure. Thus, whenever one of the main memory blocks 0, 128, 256,... is loaded into the cache, it is stored in cache block 0. Blocks 1, 129, 257,... are stored in cache block 1, and so on. Since more than one memory block is mapped onto a given cache block position, contention may arise for that position even when the cache is not full.

Processor generates memory address of 16 bits, which has 3 fields (in direct mapping technique),

1. 4bits, used to select a **WORD** in the cache block (2power4= 16)
2. 7bits, used to select a **BLOCK** in cache memory (2power7 = 128)
3. 5bits, **TAG field** - used to identify which block of main memory (4Kblocks/128blocks=32)is currently sitting in the selected cache block (2power5=32)

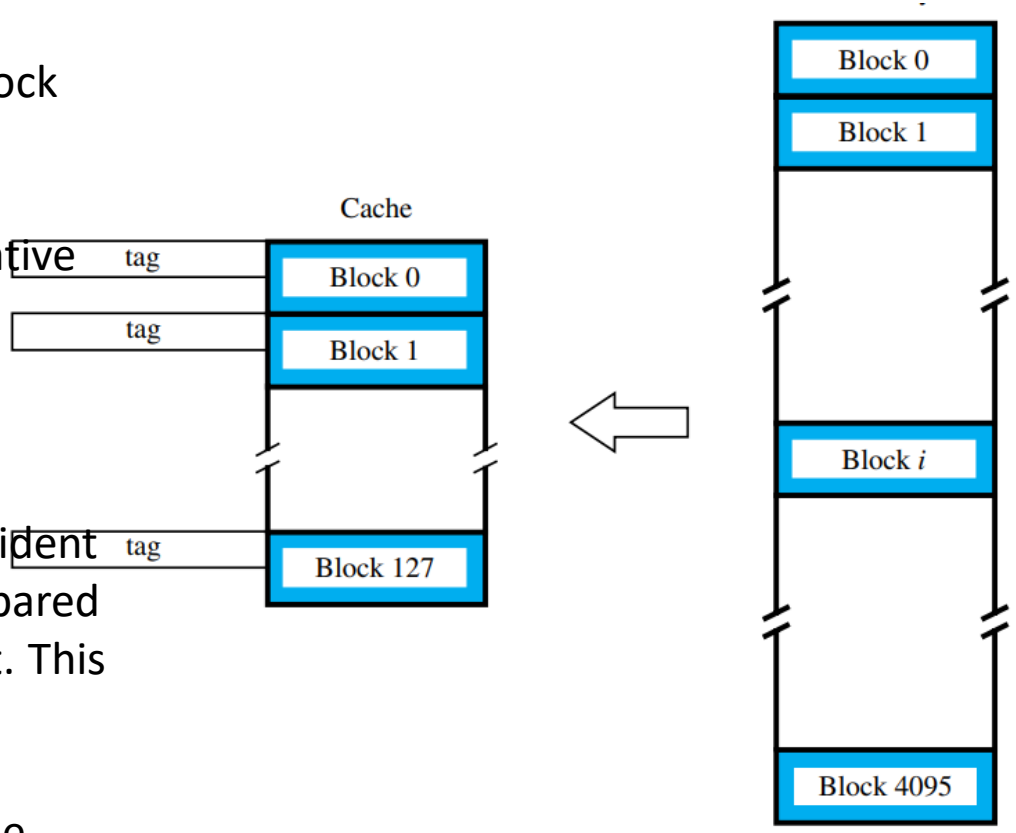
If the tag field is not matched, then it is a cache-miss, then required block from main memory is copied to cache memory and tag filed is suitably set.

The direct-mapping technique is easy to implement, but it is not very flexible



Associative Mapping

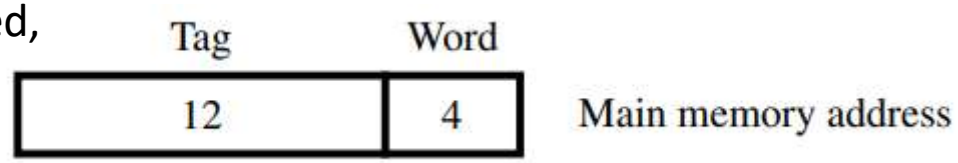
The disadvantage of direct mapping, each main memory block can be placed only in a particular cache block, this is overcome here.
Figure shows the most flexible mapping method, in which a main memory block can be placed into any cache block position.



Processor generates memory address of 16 bits, which has 2 fields (in associative mapping technique),
1. 4bits, used to select a **WORD** in the cache block (2power4= 16)
2. 12 bits, TAG field, used to identify a memory block(2power12=4096)

In this case, 12 tag bits are required to identify a memory block when it is resident in the cache. The tag bits of an address received from the processor are compared to the tag bits of each block of the cache to see if the desired block is present. This is called the associative-mapping technique.

It gives complete freedom in choosing the cache location in which to place the memory block, resulting in a more efficient use of the space in the cache. When a new block is brought into the cache, it replaces (ejects) an existing block only if the cache is full. In this case, we need an algorithm to select the block to be replaced, like LRU algorithm.
The complexity of an associative cache is higher than that of a direct-mapped cache, because of the need to search all 128 tag patterns to determine whether a given block is in the cache. To avoid a long delay, the tags must be searched in parallel. A search of this kind is called an associative search.



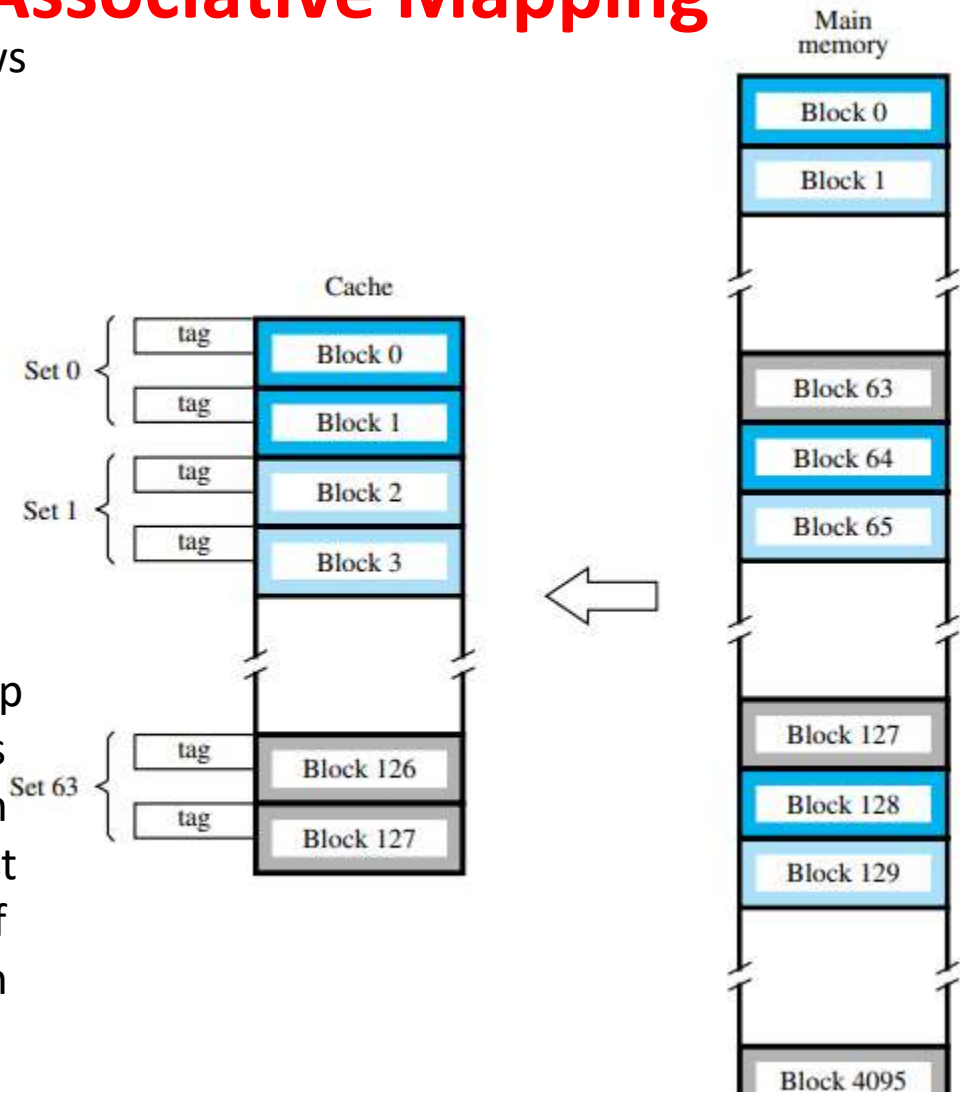
Set Associative Mapping

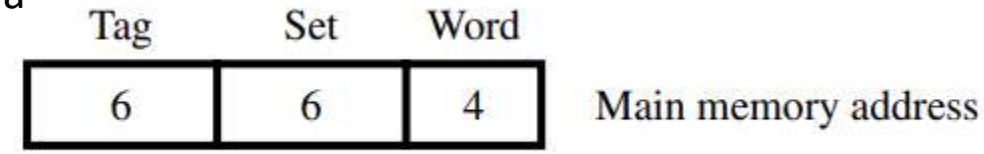
Another approach is to use a combination of the direct- and associative-mapping techniques. The blocks of the cache are grouped into sets, and the mapping allows a block of the main memory to reside in any block of a specific set. Hence, the contention problem of the direct method is eased by having a few choices for block placement. At the same time, the hardware cost is reduced by decreasing the size of the associative search.

Processor generates memory address of 16 bits, which has 3 fields (in set associative mapping technique),

1. 4bits, used to select a **WORD** in the cache block (2power4= 16words)
2. 6bits, used to select a **SET** in cache memory (2power6 = 64 sets)
3. 6bits, **TAG field** - used to identify the block in the selected set.

An example of this set-associative-mapping technique is shown in Figure for a cache with two blocks per set. In this case, memory blocks 0, 64, 128,..., 4032 map into cache set 0, and they can occupy either of the two block positions within this set. Having 64 sets means that the 6-bit set field of the address determines which set of the cache might contain the desired block. The tag field of the address must then be associatively compared to the tags of the two blocks of the set to check if the desired block is present. The number of blocks per set is a parameter that can be selected to suit the requirements of a particular computer. For the main memory and cache sizes in Figure, four blocks per set can be accommodated by a 5-bit set field, eight blocks per set by a 4-bit set field, and so on. The extreme condition of 128 blocks per set requires no set bits and corresponds to the fully-associative technique, with 12 tag bits. The other extreme of one block per set is the direct-mapping method. A cache that has k blocks per set is referred to as a k-way set-associative cache.





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.

Assume a computer has 32 bit addresses. Each block stores 16 words. A direct mapped cache has 256 blocks. In which block (line) of the cache would we look for each of the following addresses. Addresses are given in hexadecimal.

1. 1C2B C052
2. AFFB 00EF

ANSWER: first four bits (first digit/nibble) used to identify a word, 2nd and 3rd digits/nibbles (=8bits) are used to identify a cache block (because, 2power8=256 blocks)

1. 1C2B C 05 2 , 5TH CACHE BLOCK
2. AFFB 0 0E F, 15TH CACHE BLOCK (0E –>14th block, among 0-255 blocks)

A computer has an 8 Gbyte memory with 64 bit word sizes. Each block of memory stores 16 words. The computer has a direct mapped cache of 128 blocks. The computer uses word level addressing. What is the address format. If we change the cache to 4 way set associative cache, what is the new address format.

Direct Mapping

No.Of.Words in a block = 16 = 2power4, hence 4 bits are required to select a word in a block

Main mem in words = 8G bytes/no.of.bytesPerWord = 8 G bytes/8 (because 64bit word = 8 bytes) = 1G words

Hence, 1G words =2power30, hence memory adds is 30 bits

No. of blocks in main memory = 1 G words/ no.of.words.perblock = 1G/16 = 2power30 / 2power4 = 2power26

No. of Cache blocks = 128 = 2power7, hence 7 bits required to identify cache block

No. of bits for TAG field = 30bits – (7+4) = 19 bits

Address format: 30bits -> 19 (tag bits) +7(block) +4(word)

4 way set associative cache

No. of. Words.in a block = 2power4, 4 bits for word

No. of. Sets = 128blocks/no.of.blocks perSet = 128/4 = 32 = 2power5, 5bits for SET

TAG bits = 30 – (5+4) = 21 bits

Address format: 30bits -> 21(tag bits) + 5(set) + 4 (word)

(Note: With 8 GB and a 64 bit word size, there are 8 GB / (8 bytes / word) = 1 GW of memory. This requires 30 bits for an address. Of the 30 bits, we need 4 bits for the word on the line and 7 bits for the block number, leaving 30 – (7 + 4) = 19 bits for the tag. So the address format is 19 – 7 – 4. If we have a 4-way set associative cache instead, then there will be 4 sets with 128 / 4 = 32 blocks per set. So we would only need 5 bits for the block number, leaving 30 – (5 + 4) = 21 bits for the tag. So the address format is 21 – 5 – 4. 4.)

A computer memory system uses a primary memory with 100 nanosecond access time, fronted by a cache memory with 8 nanosecond access time. What is the effective access time if,

a) The hit ratio is 0.7?
b) The hit ratio is 0.9?
c) The hit ratio is 0.95?
d) The hit ratio is 0.99?

h – hit rate , T_P as the access time of the cache, T_S as the access time of the primary memory.

Effective Memory Access time = $T_E = h \cdot T_P + (1 - h) \cdot T_S$

(Effective Access time = Hit rate x Access time of Cache + (1-Hit rate) x Access time of Primary memory)

applying The equation, $T_E = h \cdot 8 + (1 - h) \cdot 100$. ($h = 0.8$, $T_P = 8ns$ $T_S = 100ns$)

a) h = 0.7 $T_E = 0.7 \cdot 8 + (1 - 0.7) \cdot 100 = 0.7 \cdot 8 + 0.3 \cdot 100 = 5.6 + 30.0$ = **35.6ns**

b) h = 0.9 $T_E = 0.9 \cdot 8 + (1 - 0.9) \cdot 100 = 0.9 \cdot 8 + 0.1 \cdot 100 = 7.2 + 10.0$ = **17.2ns**

c) h = 0.95 $T_E = 0.95 \cdot 8 + (1 - 0.95) \cdot 100 = 0.95 \cdot 8 + 0.05 \cdot 100 = 7.6 + .5.0$ = **12.6ns**

d) h = 0.99 $TE = 0.99 \cdot 8 + (1 - 0.99) \cdot 100 = 0.99 \cdot 8 + 0.01 \cdot 100 = 7.92 + 1.0$ = **8.92 ns**

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

Answers:

1. Effective access time = 0.98 * 2 ns + (1-0.98) * 40 ns = 0.98 * 2ns + 0.02*40ns = 1.96 + 0.8 = 2.76ns

2. With the on-chip cache, = 0.94 * .5 ns + .06 * (0.98 * 2 ns + .02 * 40 ns) = 0.94*.5 ns + 0.06 * 2.76 = .636ns.

The speedup is 2.76 / .636 = 4.33

You are asked to implement a 128M by 32 memory (1M = 2^{20}), using only 16M by 8 memory chips.

a) What is the minimum size of the MAR?

b) What is the size of the MBR?

c) How many 16M by 8 chips are required for this design?

Answer: a) 128M = $2^7 \cdot 2^{20} = 2^{27}$, so the minimum MAR size is **27 bits**.

b) The MBR size is **32 bits**.

c) 128M·32 / 16M·8 = 8·4 = **32 chips**.

You are asked to implement a 128M by 32 memory (1M = 2^{20}),
using only 16M by 8 memory chips.

a) What is the minimum size of the MAR?

b) What is the size of the MBR?

c) How many 16M by 8 chips are required for this design?

ANSWER: Remember that 128M = $2^7 \bullet 2^{20} = 2^{27}$ and that 16M = 2^{24}.

a) To address 128M = 2^{27} entries, we need a **27-bit MAR**.

b) Since each entry is 32-bits, we have a **32-bit MBR**.

Design a 16K-by-8 memory using 2K-by-4 memory chips. Arrange the chips in an 8 by 2 array (8 rows of 2 columns) and design the decoding circuitry.

number of chips needed as (16K • 8) / (2K • 4) =
128K / 8K = 16

We are given a collection of 2K-by-4 memory chips. As $2K = 2^{11}$, each of these chips has eleven address lines, denoted $A_{10}A_9A_8A_7A_6A_5A_4A_3A_2A_1A_0$, four data lines, $D_3D_2D_1D_0$, and the required control and select lines. We are asked to design a 16K-by-8 memory. As $16K = 2^{14}$, this memory has 14 address lines, denoted A_{13} through A_0. The memory to be designed has eight data lines, denoted D_7 through D_0, and necessary control lines.