Course – Computer Organization and Architecture

Course Instructor

Dr. Umadevi V

Department of CSE, BMSCE

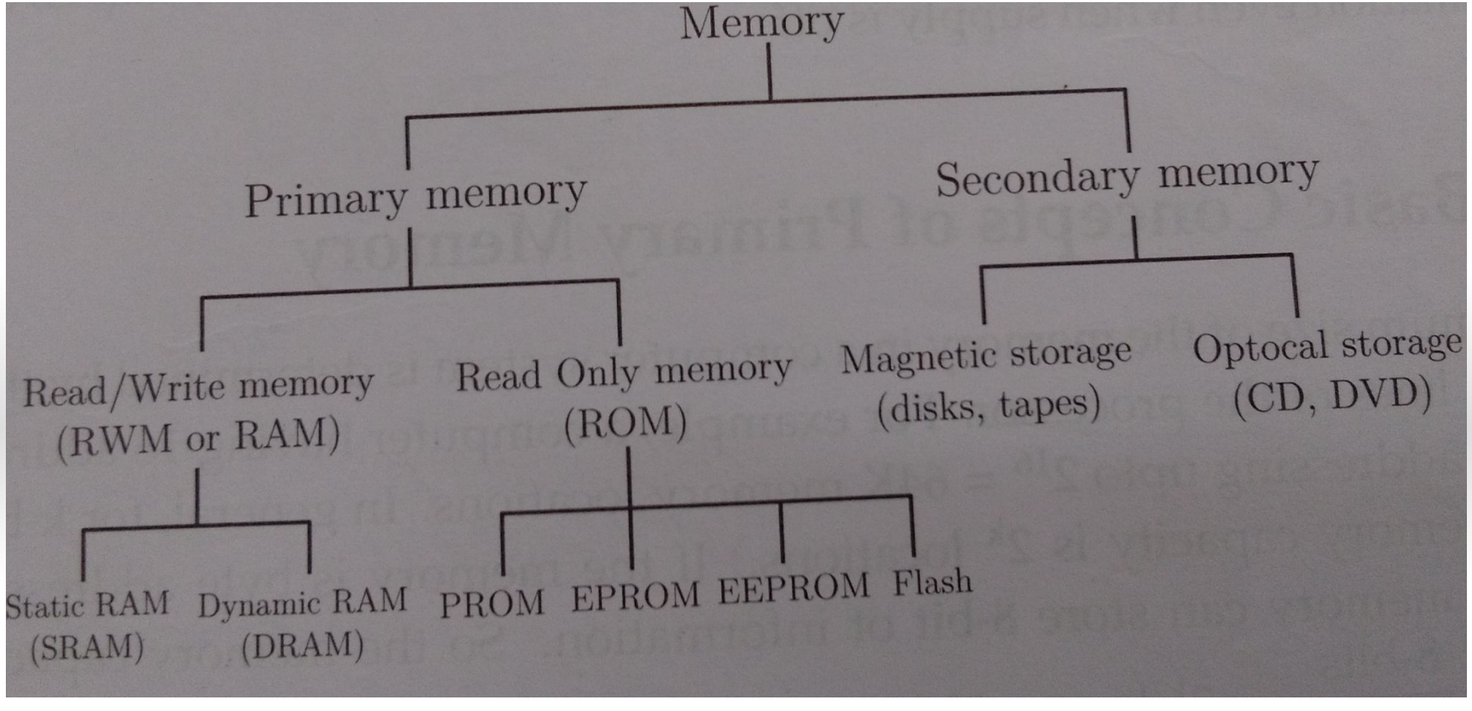
Unit-3

Memory System: Basic Concepts, Semiconductor RAM Memories, Read-only Memories, Direct Memory Access, Memory Hierarchy

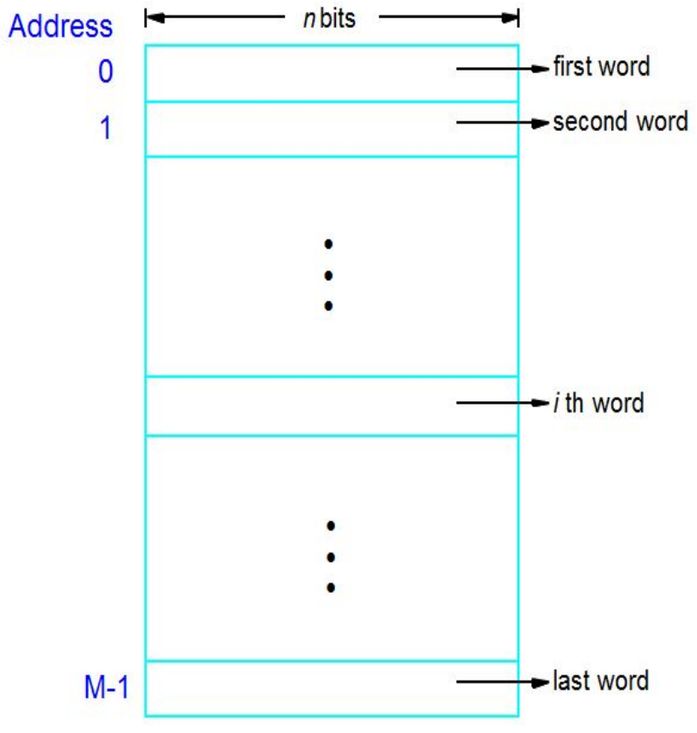
Cache Memories: Mapping Functions, Virtual Memory

Memory Chip

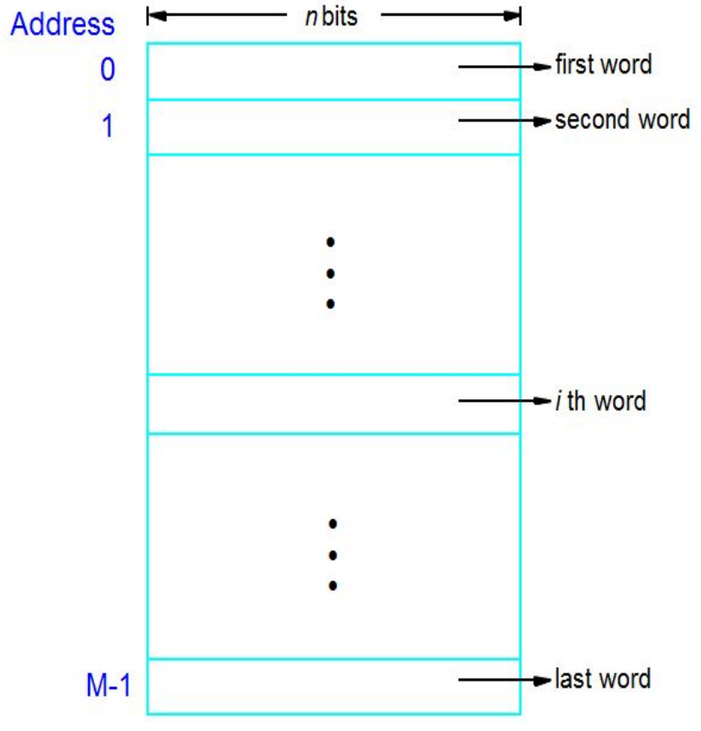
Classification of Memory System



Memory Locations and Addresses



Memory Locations and Addresses



Address Length **K-bits**	Addressable Locations 2^k
2	2^2=4 Locations
3	2^3=8 Locations
4	2^4=16 Locations

Memory

Address K=2bits		Word Length n =8bits = 1 Byte	
0	00	0000 0110	1st Byte or word
1	01	0000 0111	2nd Byte or word
2	10	0000 1000	3rd Byte or word
3	11	0000 1010	4th Byte or word

Memory Locations and Addresses

Address Length **K-bits**	Addressable Locations **2^k**
2	2^2=4 Locations
3	2^3=8 Locations
4	2^4=16 Locations

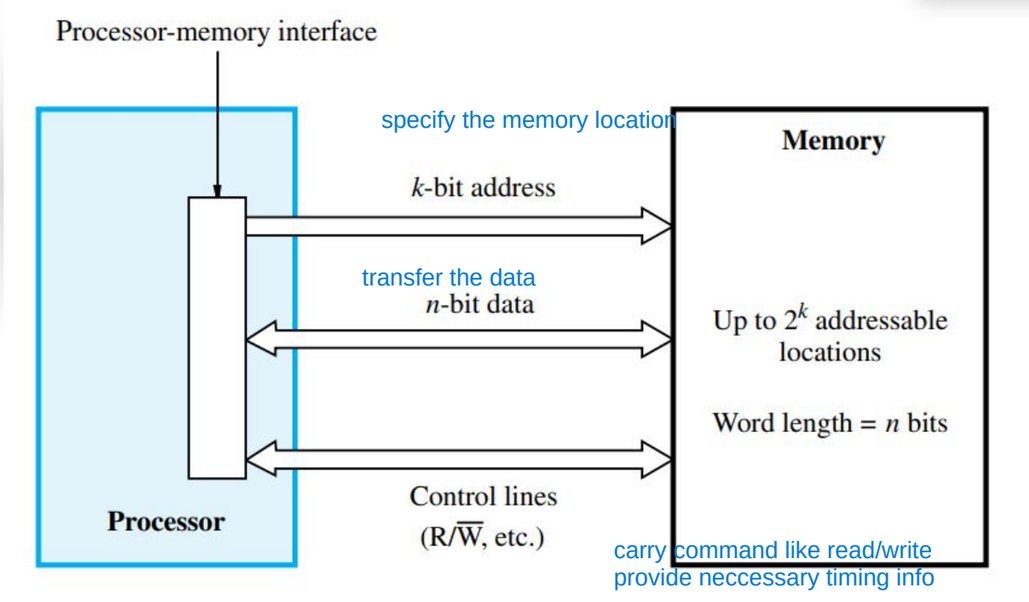
Address	Memory Locations
16 Bit	2^{16} = 64 K
32 Bit	2^{32} = 4G (Giga)
40 Bit	2^{40} = IT (Tera)

Memory

Address K=3bits		Word Length n =8bits = 1 Byte	
0	000	0000 0110	1st Byte or word
1	001	0000 0111	2nd Byte or word
2	010	0000 1000	3rd Byte or word
3	011	0000 1010	4th Byte or word
4	100	0000 1011	5th Byte or word
5	101	0000 1100	6th Byte or word
6	110	0000 1101	7th Byte or word
7	111	0000 1110	8th Byte or word

Connection of the memory to the processor

- The connection between the processor and its memory consists of address, data, and control lines, as shown in Figure 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.



Semiconductor RAM Memories: random access memory/ read/write memory
Internal Organization of Memory Chips

Example : 16x8 Memory Organization

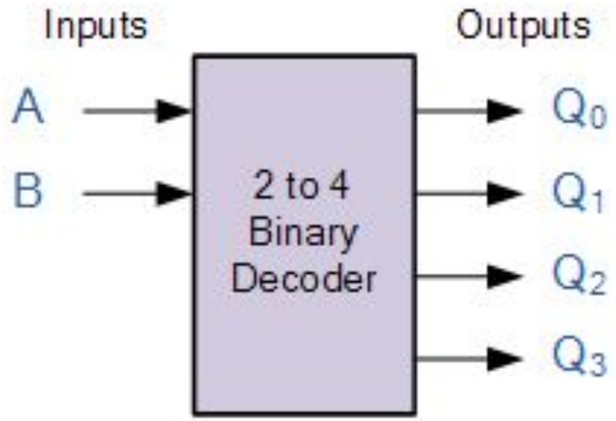
16x8 Memory

16 memory locations
8 bits in each location

Address K=4bits		Word Length n =8bits = 1 Byte	
0	0000	0000 0001	1st Byte or word
1	0001	1000 0011	2nd Byte or word
2	0010	---	
3	0011	---	
.	--	---	
.	--	---	
.	--	---	
15	1111	0000 1110	16th Byte or word

Binary Decoder

- The name "Decoder" means to translate or decode coded information from one format into another, so a binary decoder transforms "n" binary input signals into an equivalent code using 2^n outputs
- An example of a **2-to-4 line decoder** along with its truth table is given as:



Truth Table

A	B	Q_0	Q_1	Q_2	Q_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Example : 16x8 Memory Organization

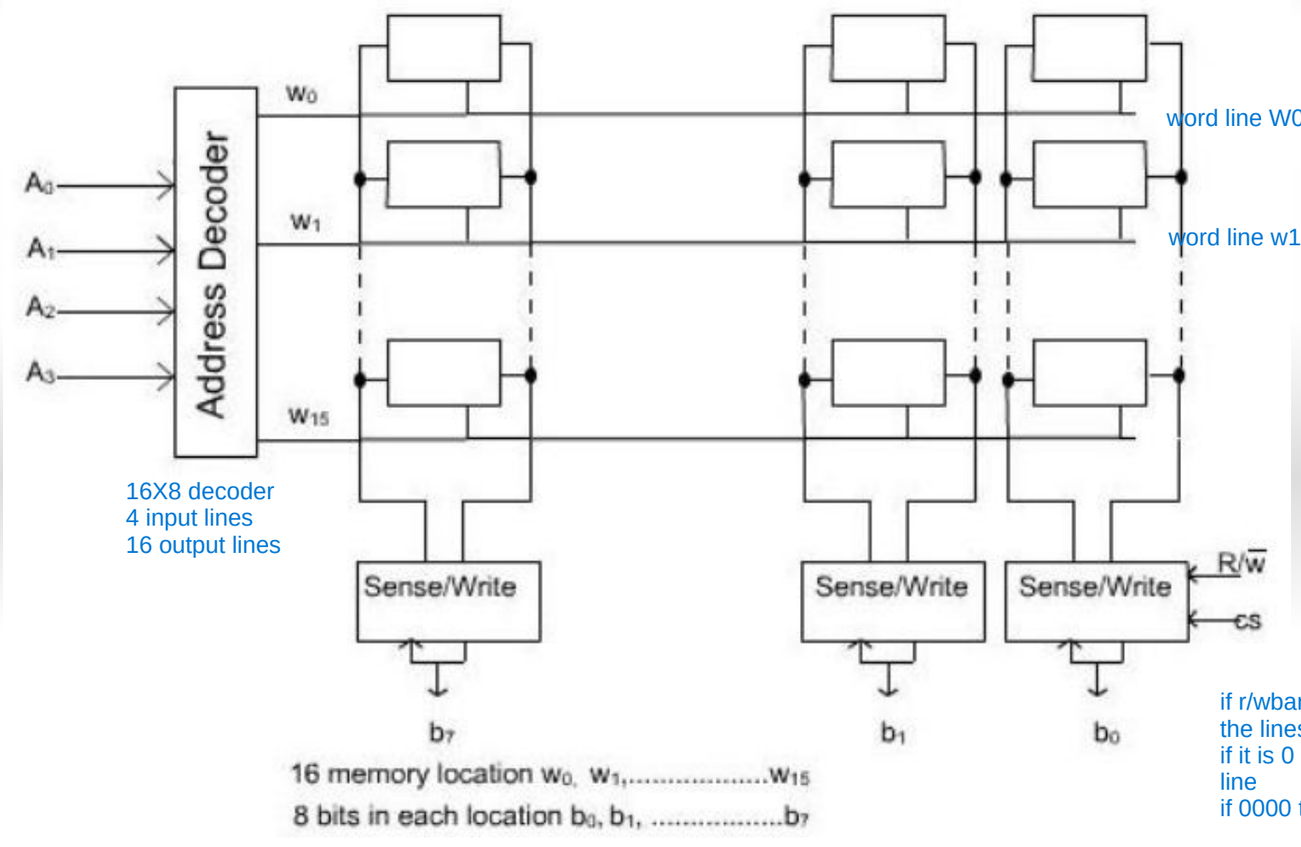
16x8 Memory

Address K=4bits		Word Length n =8bits = 1 Byte
0	0000	0000 0001
1	0001	1000 0011
2	0010	---
3	0011	---
.	--	---
.	--	---
.	--	---
15	1111	0000 1110

Internal Organization 0f 16x8 Memory Chip

8 bits

address lines



word line W0

word line w1

16X8 decoder
4 input lines
16 output lines

chip select

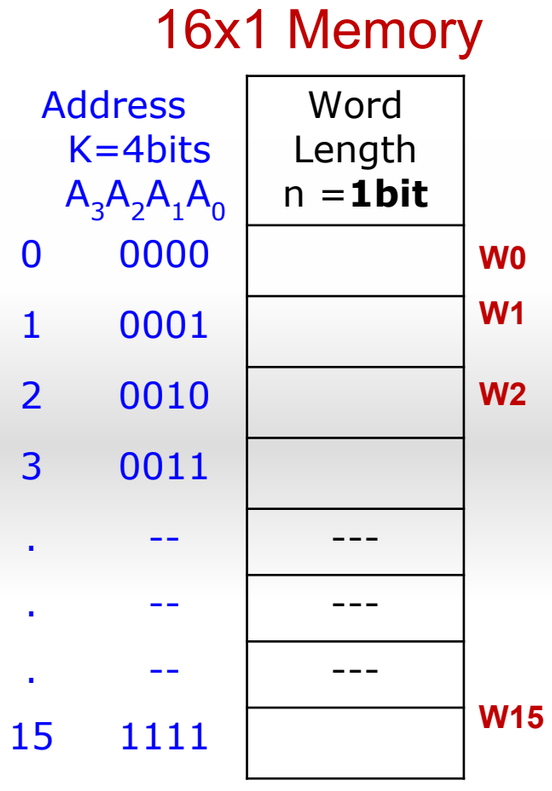
if r/wbar is 1 then reads the content of appropriate word line to the lines b0-b7
if it is 0 then writes the contents from b0-b7 to appropriate word line
if 0000 then W0(word line)

Internal Organization of Memory Chips (Contd…)

- Memory-cells are organized in the form of array
- Each cell is capable of storing 1-bit of information.
- Each row of cells forms a memory-word.
- All cells of a row are connected to a common line called as **Word-Line**.
- The cells in each column are connected to **Sense/Write** circuit by 2-bit-lines.
- The Sense/Write circuits are connected to data-input or output lines of the chip.
- During a write-operation, the sense/write circuit
  - → receive input information &
  - → store input info in the cells of the selected word.

- The data-input and data-output of each Sense/Write circuit are connected to a single bidirectional data-line.
- Data-line can be connected to a data-bus of the computer.
- Following 2 control lines are also used:
  1) **R/W'** → Specifies the required operation.
  2) **CS'** → Chip Select input selects a given chip in the multi-chip memory-system.

Internal Organization of 16x1 Memory Chip



Internal Organization of 16x1 Memory Chip

Method 1:

Using 4-int-16 decoder

Arranging Data cells as single column

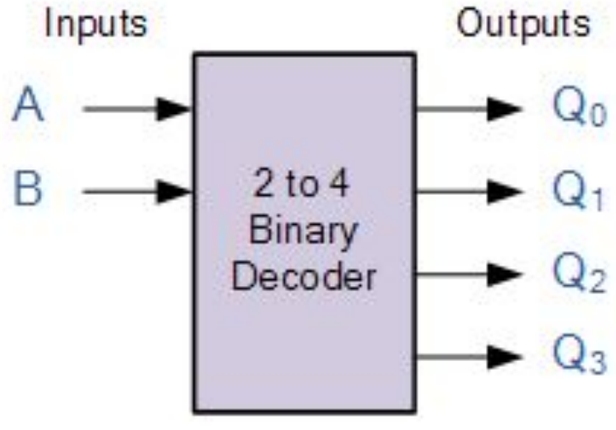
Method 2:

Using 2-into-4 decoder and 4-into-1 Multiplexer

Arranging Data cells as 4 rows and 4 columns

Binary Decoder

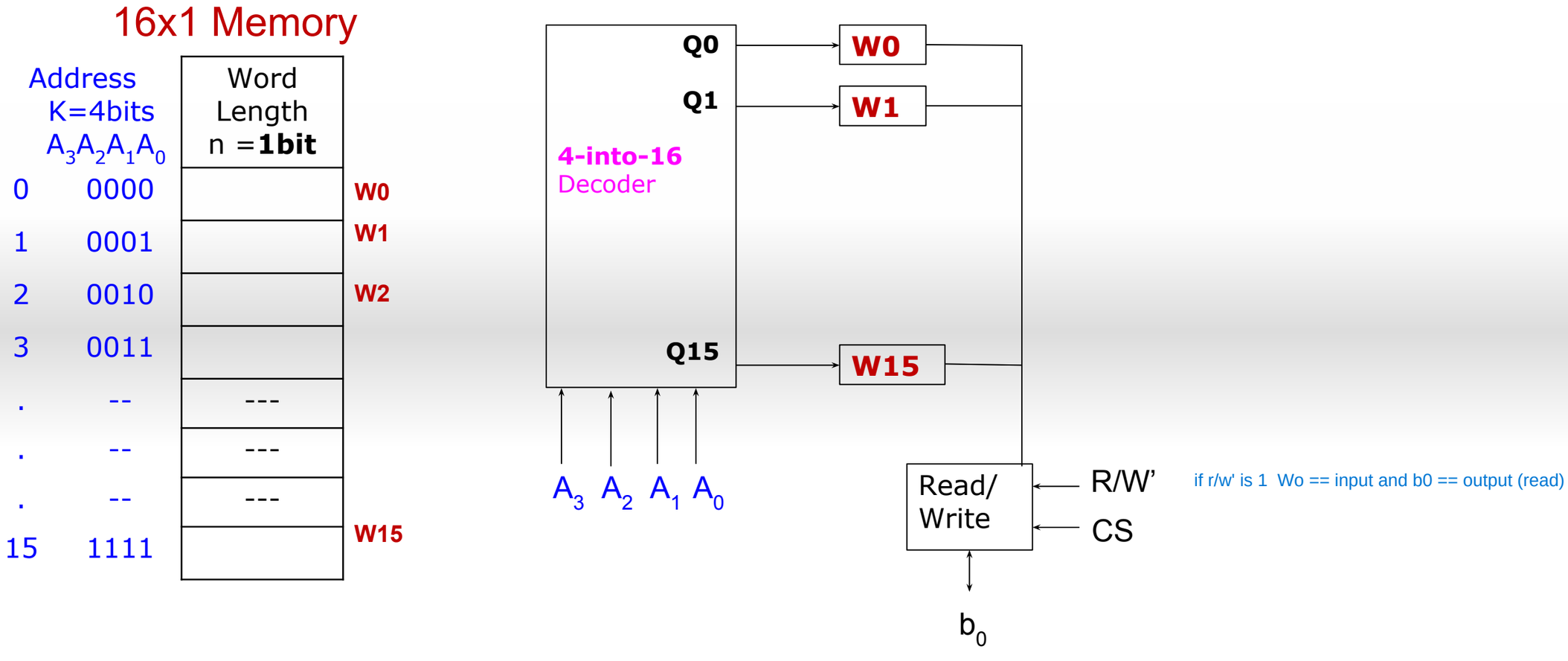
- The name "Decoder" means to translate or decode coded information from one format into another, so a binary decoder transforms "n" binary input signals into an equivalent code using 2^n outputs
- An example of a **2-to-4 line decoder** along with its truth table is given as:



Truth Table

A	B	Q_0	Q_1	Q_2	Q_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Method 1: Internal Organization of 16x1 Memory Chip

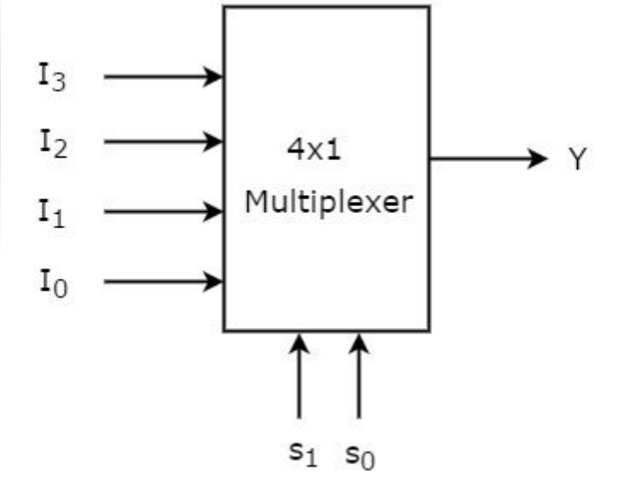


Multiplexer

- **Multiplexer** is a combinational circuit that has maximum of 2^n data inputs, 'n' selection lines and single output line. One of these data inputs will be connected to the output based on the values of selection lines.
- Since there are 'n' selection lines, there will be 2^n possible combinations of zeros and ones. So, each combination will select only one data input. Multiplexer is also called as **Mux**.

Example of4x1 Multiplexer

- 4x1 Multiplexer has four data inputs I_3, I_2, I_1 & I_0, two selection lines s_1 & $s_0$and one output Y. The **block diagram** of 4x1 Multiplexer is shown in the following figure.



Selection Lines		Output
S_1	**S_0**	**Y**
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Method 2: Internal Organization of 16x1 Memory Chip

16x1 Memory

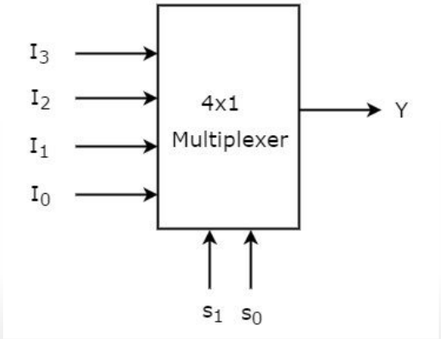
	Address K=4bits $A_3A_2A_1A_0$	Word Length n = **1bit**	
0	0000		W0
1	0001		W1
2	0010		W2
3	0011		W3
.	--	---	
.	--	---	
.	--	---	
15	1111		W15

A_3	A_2	A_1	A_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

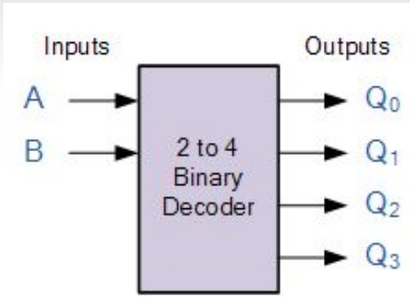
Internal Organization of 16x1 Memory Chip

16x1 Memory

Address K=4bits $A_3A_2A_1A_0$		Word Length n = **1bit**	
0	0000		**W0**
1	0001		**W1**
2	0010		**W2**
3	0011		**W3**
.	--	---	
.	--	---	
.	--	---	
15	1111		**W15**



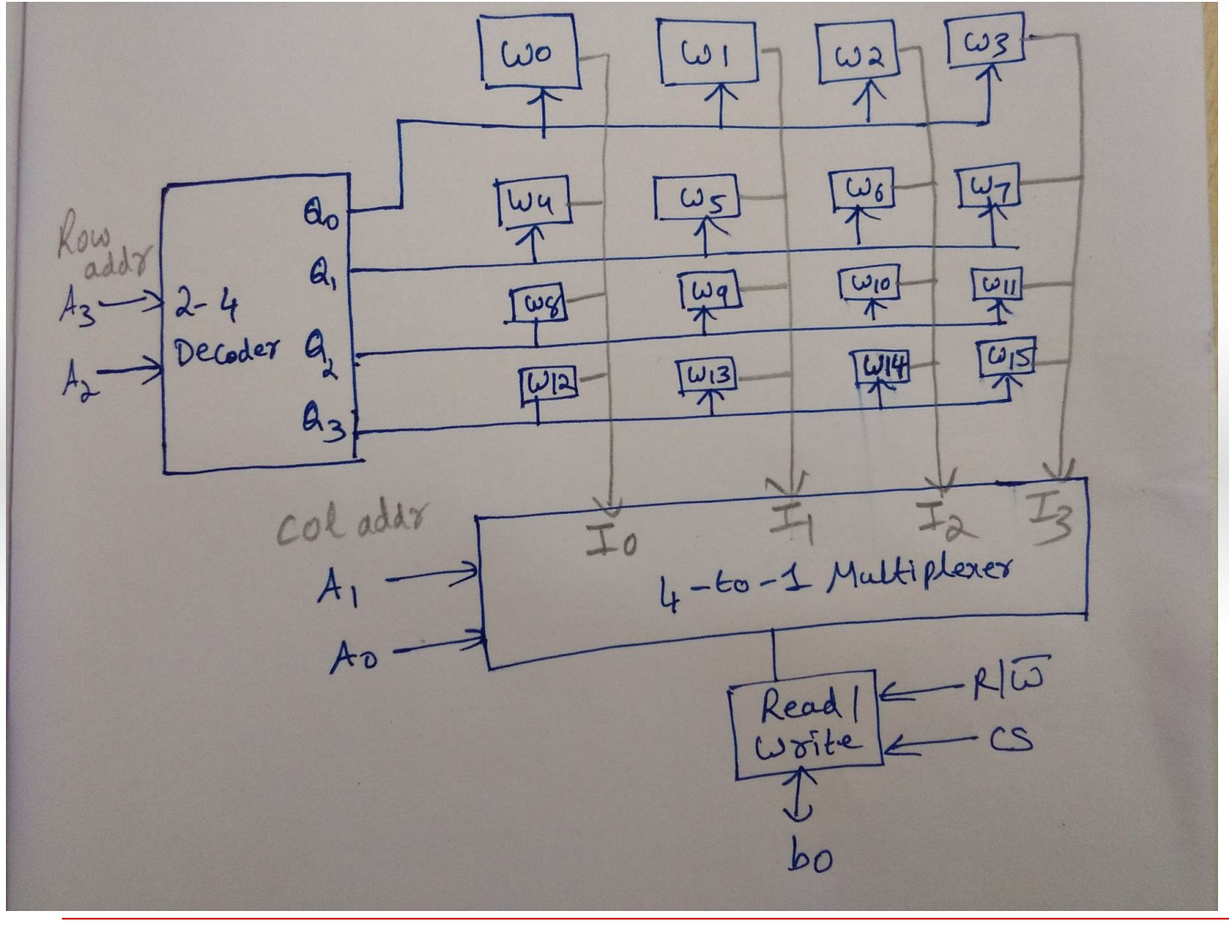
Selection Lines		Output
S_1	**S_0**	**Y**
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3



Truth Table

A	B	Q_0	Q_1	Q_2	Q_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Method 2: Internal Organization of 16x1 Memory Chip



A_3	A_2	A_1	A_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

Internal Organization of Memory Chips

Designing **1Kx1** Memory Organization

Organization of 1Kx1 Memory chip

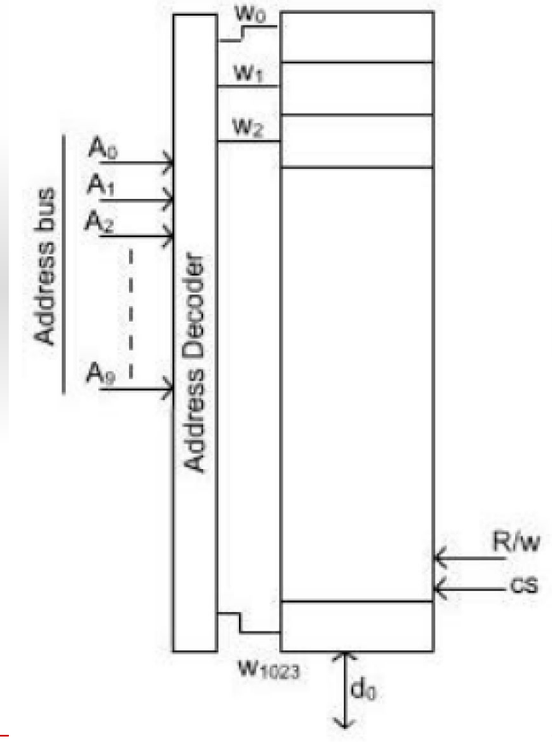
If it is organized as a 1024 x 1 memory chips, then it has got 1024 memory words of size 1 bit only.

Therefore, the size of data bus is 1 bit and the size of address bus is 10 bits (2^10=1024).

A particular memory location is identified by the contents of memory address bus. A decoder is used to decode the memory address. There are two ways of decoding of a memory address depending upon the organization of the memory module.

Method 1:Organization of 1Kx1 Memory chip (Contd…)

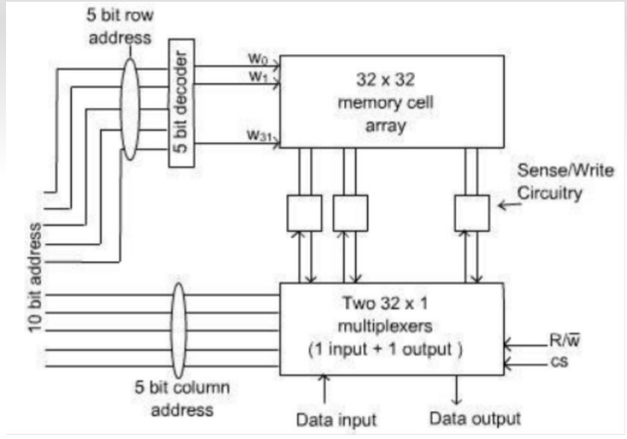
In one case, each memory word is organized in a row. In this case whole memory address bus is used together to decode the address of the specified location. The memory organization of 1024 x 1 memory chip is shown in the figure below



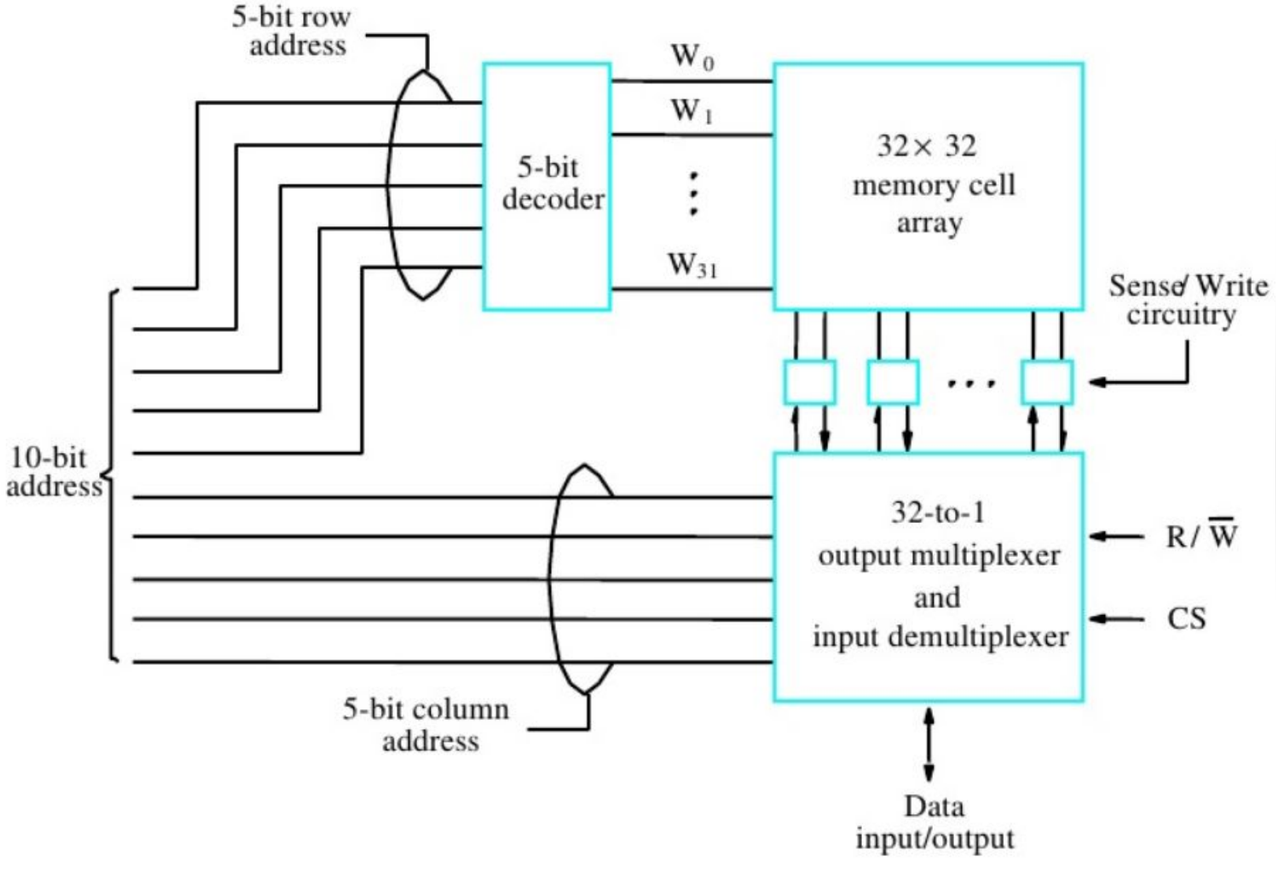
Method 2:Organization of 1Kx1 Memory chip (Contd…)

In second case, several memory words are organized in one row. In this case, address bus is divided into two gropus.

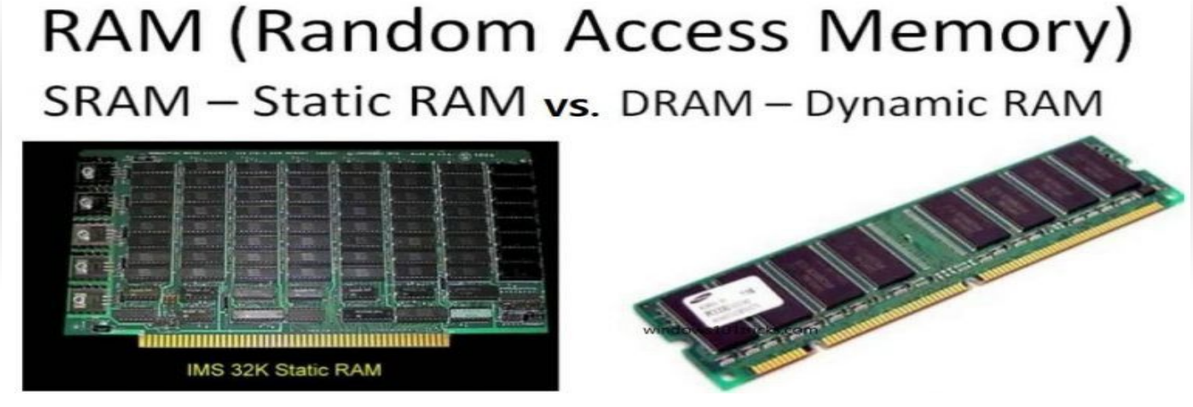
One group is used to form the row address and the second group is used to form the column address. Consider the memory organization of 1024 x 1 memory chip. The required 10-bit address is divided into two groups of 5 bits each to form the row and column address of the cell array. A row address selects a row of 32 cells, all of which are accessed in parallel. However, according to the column address, only one of these cells is connected to the external data line via the input output multiplexers. The arrangement for row address and column address decoders is shown in the figure below



Method 2:Organization of 1Kx1 Memory chip (Contd…)

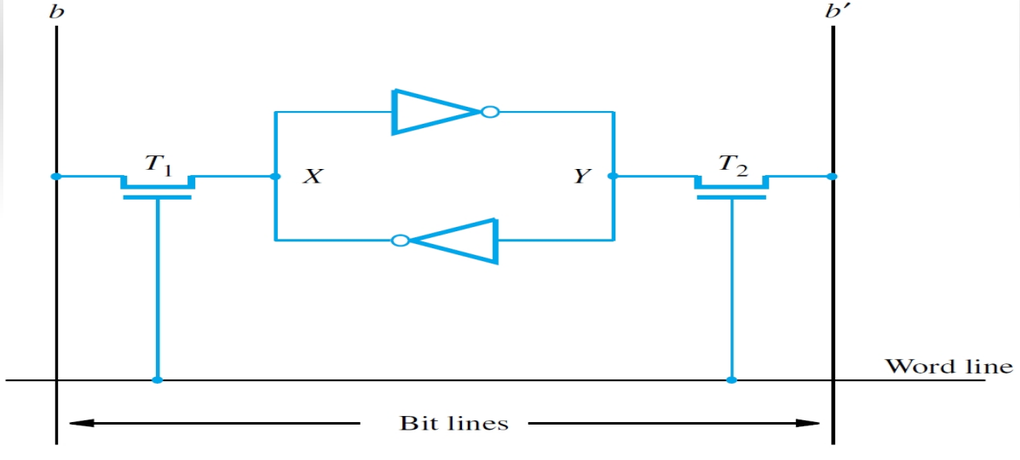


RAM


RAM (Random Access Memory)
SRAM – Static RAM vs. DRAM – Dynamic RAM
IMS 32K Static RAM

Static Memories or Static RAM

- Memories that consist of circuits capable of retaining their state as long as power is applied are known as *static memories*. Figure below illustrates how a *static RAM* (SRAM) cell may be implemented.
- Two transistor inverters are cross connected to implement a basic flip-flop.
- The cell is connected to one word line and two bits lines by transistors T1 and T2 act as switches
- When word line is at ground level, the transistors are turned off and the latch retains its state
- Read operation: In order to read state of SRAM cell, the word line is activated to close switches T1 and T2. Sense/Write circuits at the bottom monitor the state of b and b'



A static RAM Cell

Reading and Writing Operation of SRAM

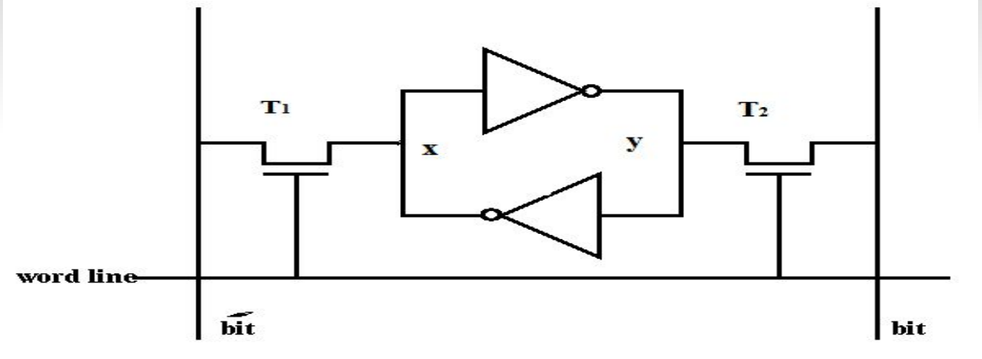
- The diagram below is illustrate Static Random Access Memory (SRAM)Cell for the explanation.

  SRAM cell is make up of 2 inverters are cross-connected to form a latch .The latch is connected to 2 bit line by transistor T1 and T2 . T1 and T2 can switch opened or closed under control of word line.When word line is ground level, the transistors are turned off and the latch retrains its state.

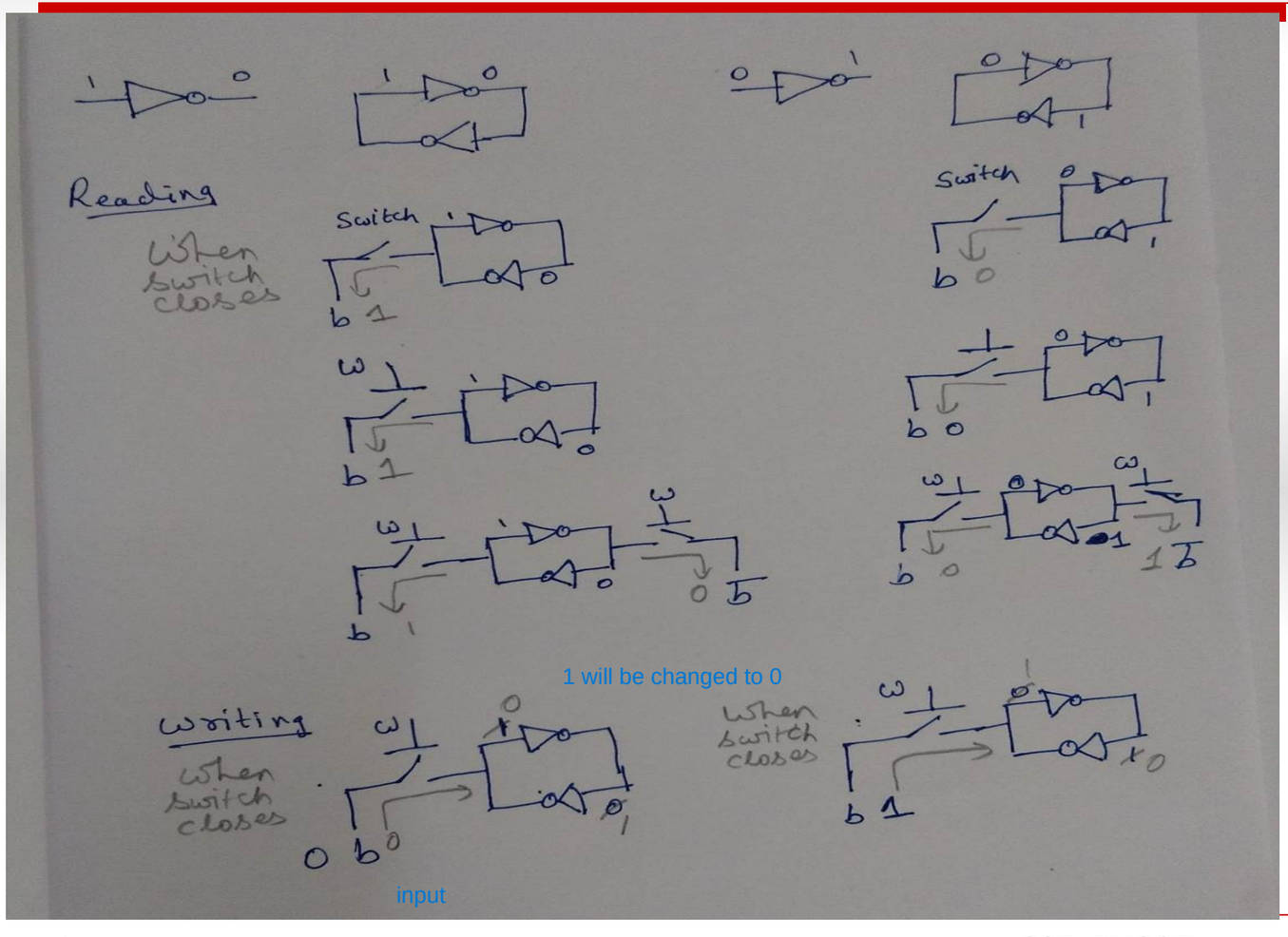
  During the **Read Operation**,Word line is activated to close switches T1 and T2 .
  If the cell is in state 1; the signal on b line is high and the signal on b' line is low.The opposite is true if the cell is in state 0.Thus, b and b' are always complements of each other's.

  The sense /write circuit at the end of the two bit line monitors their state and sets the corresponding output accordingly.

  While in **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 activate the word line.This forces the cell into the corresponding state, which the cell retains when the word line is deactivated.



Reading and Writing Operation of SRAM



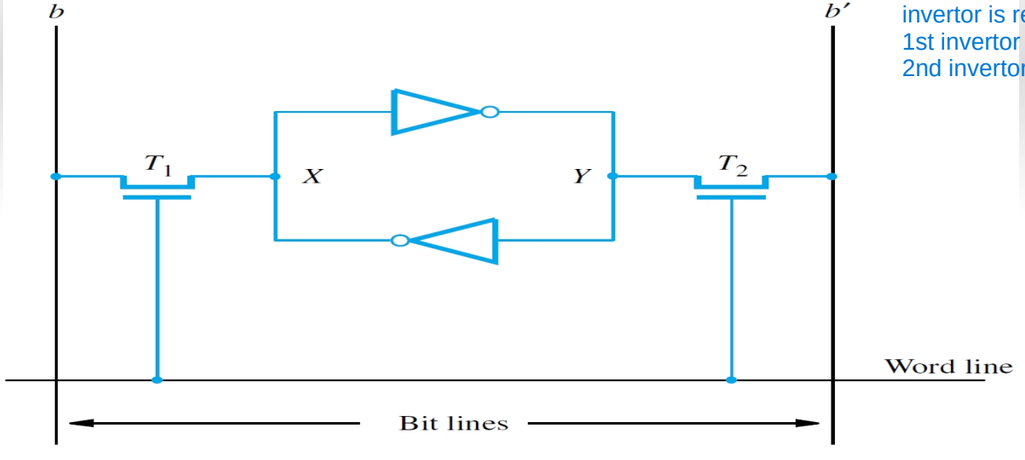
CMOS(Complementary Metal Oxide Semiconductor) Cell

A CMOS realization of the cell in Figure 1 is given in Figure 2. Transistor pairs (*T*3, *T*5) and (*T*4, *T*6) form the inverters in the latch. The state of the cell is read or written. For example, in state 1, the voltage at point *X* is maintained high by having transistors *T*3 and *T*6 on, while *T*4 and *T*5 are off. If *T*1 and *T*2 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, *T*1, *T*2, 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.



invertor is replaced/implemented by 2 transistors
1st invertor ....t3 and t5
2nd invertor ...t4 and t6

Figure 1: A static RAM Cell

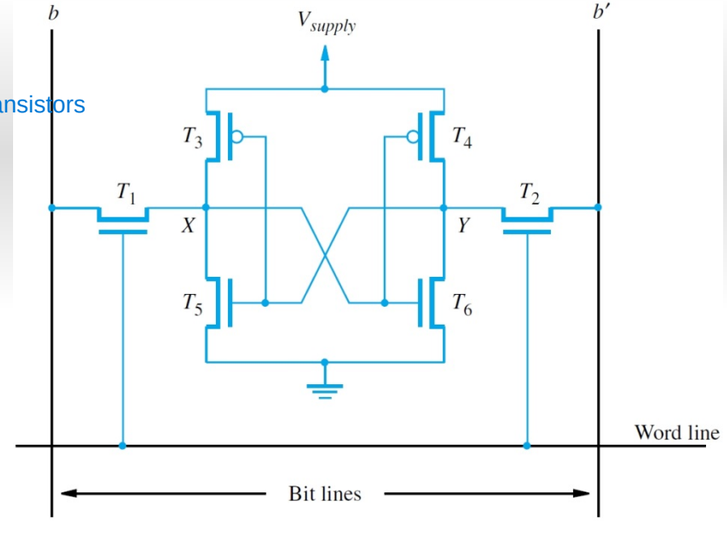
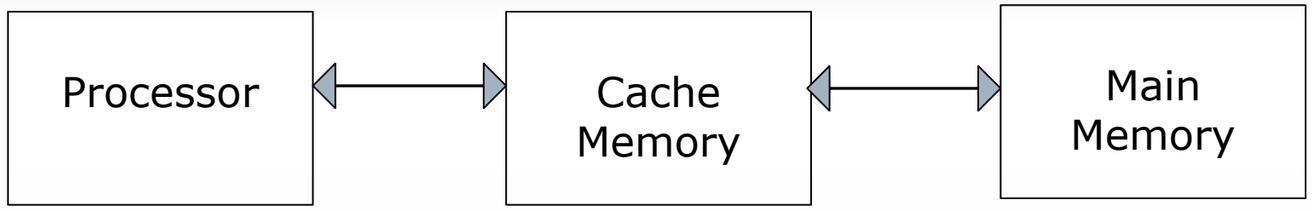


Figure 2: An example of CMOS memory Cell

Types of Random Access Memory (RAM)



RAM

Static RAM (SRAM)
(Cache Memory)

Dynamic RAM (DRAM)
(Main Memory)

Asynchronous DRAM

Synchronous DRAM clock signals will be used

Desktop or Laptops
DDR SDRAM
(Double Data Rate Synchronous Dynamic RAM)
DDR5 (5^{th} Generation)

Mobile Phones
LP DDR SDRAM
(Low Power Double Data Rate Synchronous Dynamic RAM)

Dynamic RAM: ASYNCHRONOUS DRAM

Less expensive RAMs can be implemented if simple cells are used.

- Such cells cannot retain their state indefinitely. Hence they are called Dynamic RAM(DRAM).
- The information stored in a dynamic memory-cell in the form of a charge on a capacitor.
- This charge can be maintained only for tens of milliseconds.
- The contents must be periodically refreshed by restoring this capacitor charge to its full value.

In order to store information in the cell, the transistor T is turned "ON" as shown in figure.

- The appropriate voltage is applied to the bit-line which charges the capacitor.
- After the transistor is turned off, the capacitor begins to discharge.
- Hence, info. stored in cell can be retrieved correctly before threshold value of capacitor drops down.
- During a read-operation,

→transistor is turned "ON"

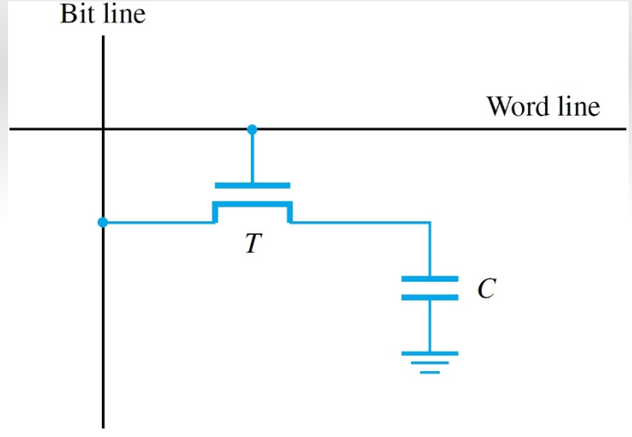
→ a sense amplifier detects whether the charge on the capacitor is above the threshold value.

If(charge on capacitor) > (threshold value)

☐ Bit-line will have logic value "1".

If(charge on capacitor)<(threshold value)

☐ Bit-line will set to logic value "0".

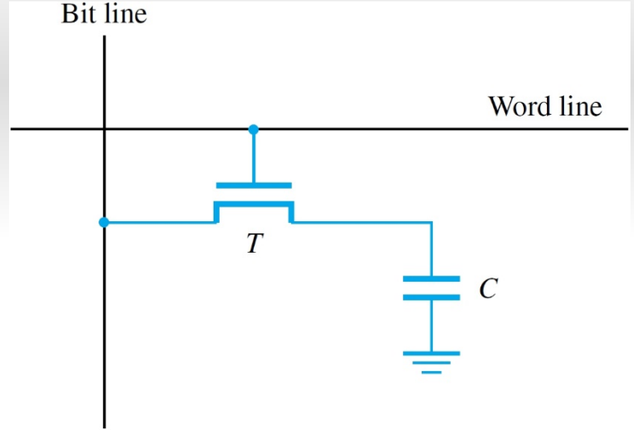


A single-transistor dynamic memory cell

Problem

Consider the dynamic memory cell of Figure . Assume that $C = 50$ femtofarads (10^{-15} F) and that leakage current through the transistor is about 9 picoamperes (10^{-12} A). The voltage across the capacitor when it is fully charged is equal to 4.5 V. The cell must be refreshed before this voltage drops below 3 V. Estimate the minimum refresh rate.

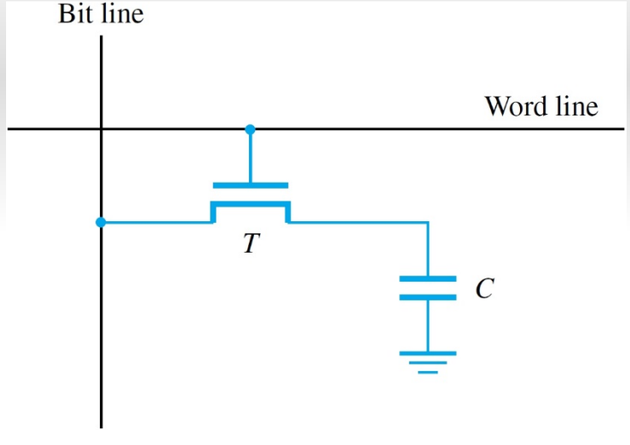
Note:
Femtofardas is coulomb/volt
Coulomb: is the SI unit of electric charge, equal to the quantity of electricity conveyed in one second by a current of one ampere.



Answer

Consider the dynamic memory cell of Figure . Assume that $C = 50$ femtofarads (10^{-15} F) and that leakage current through the transistor is about 9 picoamperes (10^{-12} A). The voltage across the capacitor when it is fully charged is equal to 4.5 V. The cell must be refreshed before this voltage drops below 3 V. Estimate the minimum refresh rate.

Note:
Femtofardas is coulomb/volt
Coulomb: is the SI unit of electric charge, equal to the quantity of electricity conveyed in one second by a current of one ampere.



The minimum refresh rate is given by

$$\frac{50 \times 10^{-15} \times (4.5 - 3)}{9 \times 10^{-12}} = 8.33 \times 10^{-3} \text{ s}$$

Therefore, each row has to be refreshed every 8 ms.

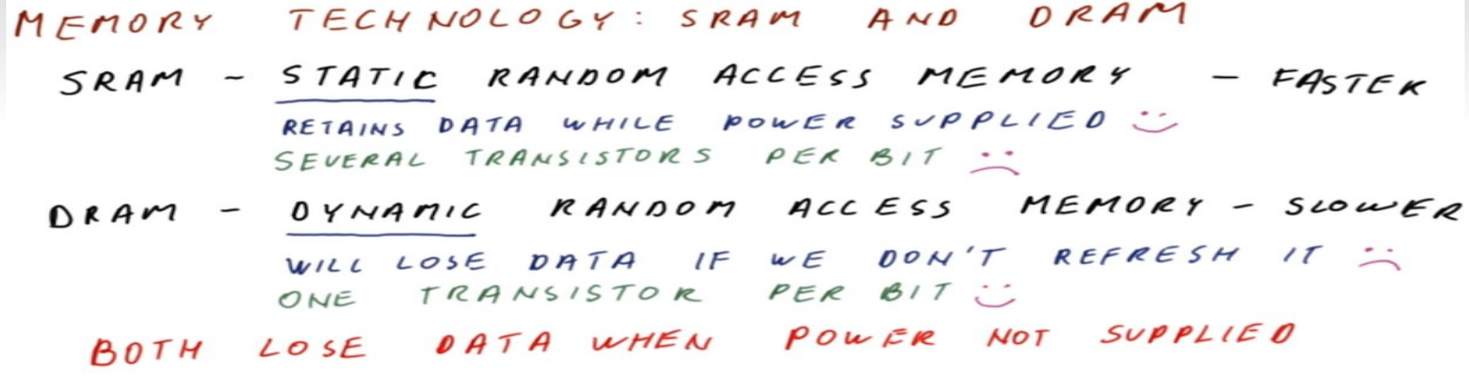
RAM: the stored data is volatile

DRAM:

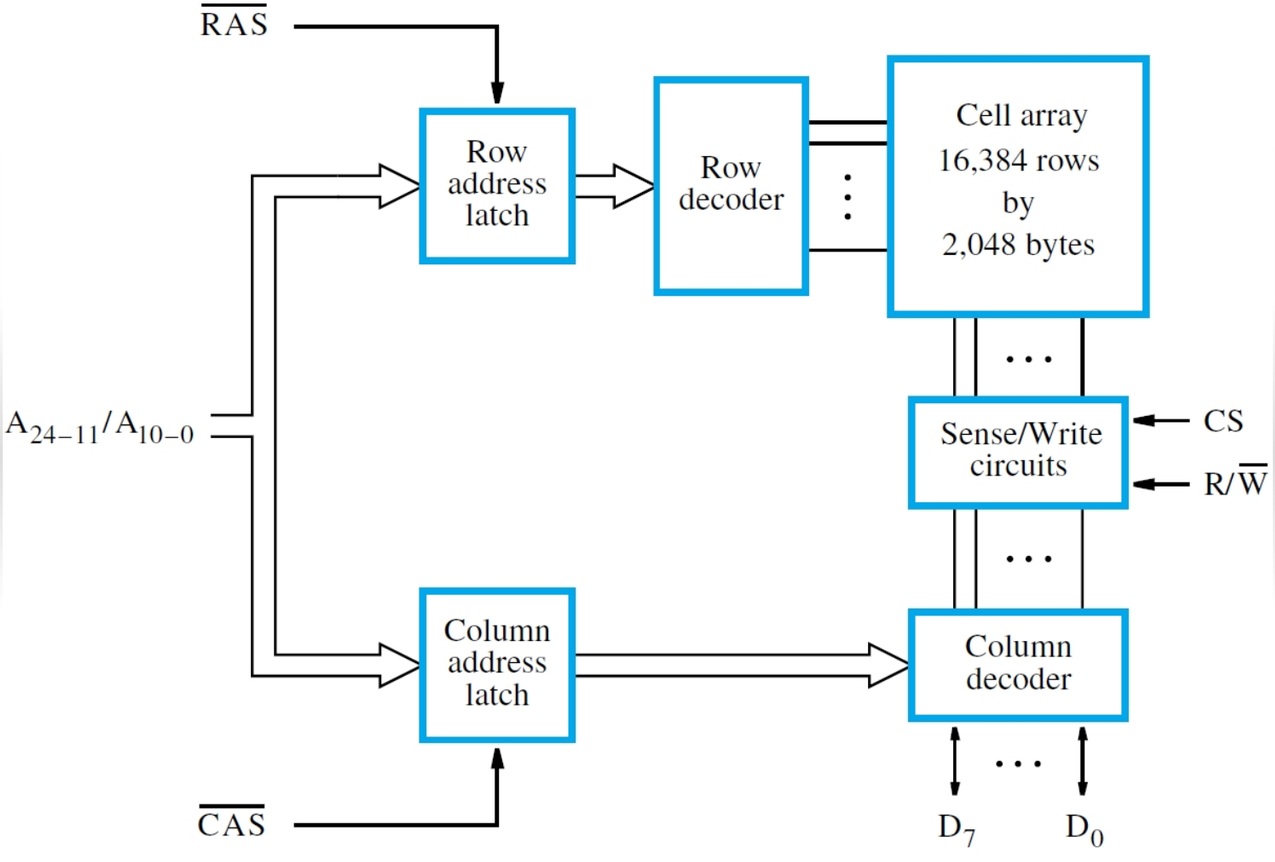
- A capacitor to store data, and a transistor to access the capacitor
- Need refresh operation
- Low cost, and high density
- It is used for main memory

SRAM:

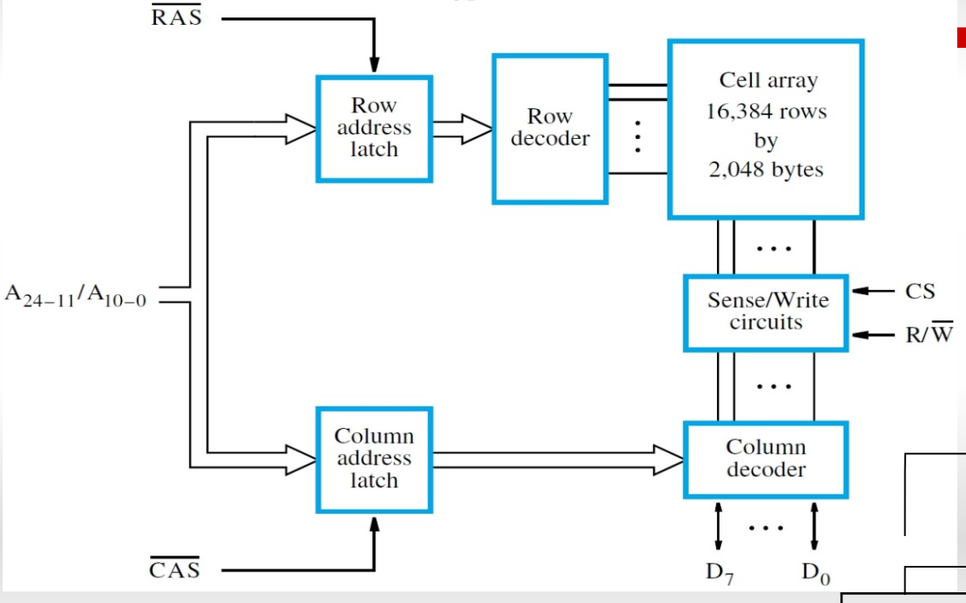
- Consists of a latch
- Don't need the refresh operation
- High speed and low power consumption
- It is mainly used for cache memory and memory in hand-held devices



Internal organization of a 32M × 8 dynamic memory chip



Rough slide to explain:
Internal organization of a 32M × 8 dynamic memory chip



32M x 8

16k x 16K

2048

8 bits = 1 Byte

1	2	3	4	5	6	7	8	----------	16384
2									
3									
--									
16K									

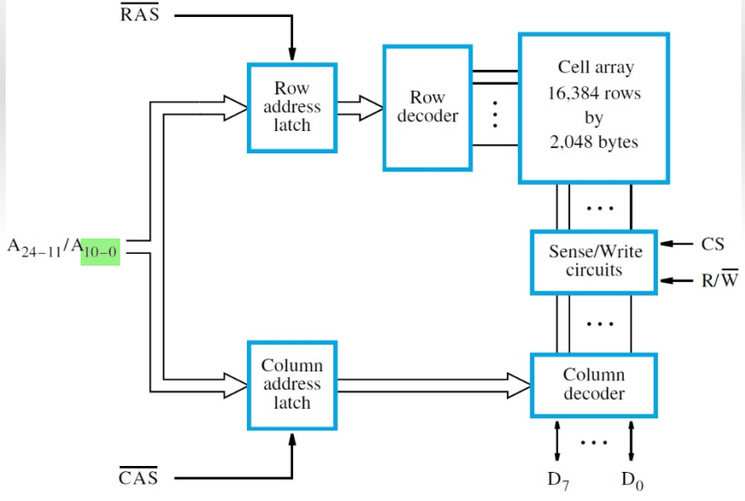
Internal organization of a 32M × 8 dynamic memory chip (Contd…)

A 256-Megabit DRAM chip, configured as 32M × 8, is shown in Figure 8.7. 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.

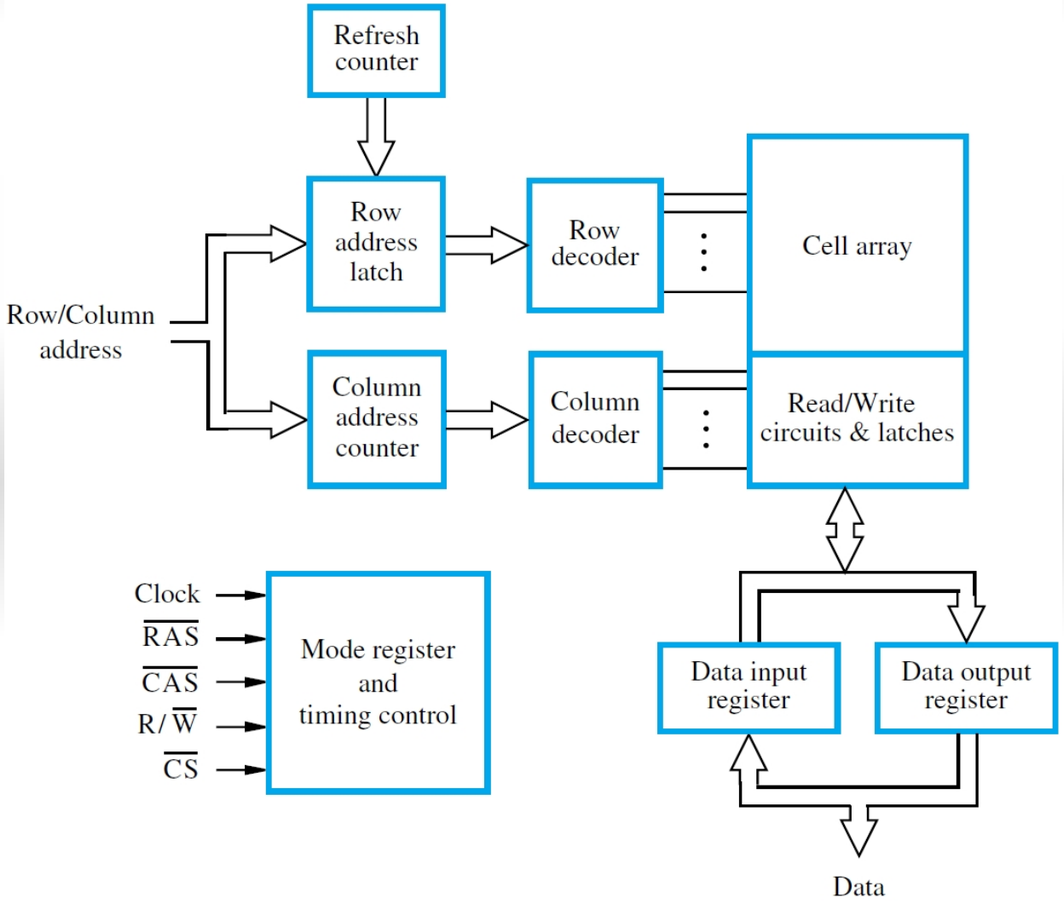
Shortly after the row address is loaded, the column address is applied to the address pins and loaded into the column address latch under control of a second control line called the Column Address Strobe (CAS). The information in this latch is decoded and the appropriate group of 8 Sense/Write circuits 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. We should note that in commercial DRAM chips, the RAS and CAS control signals are active when low. Hence, addresses are latched when these signals change from high to low. The signals are shown in diagrams as RAS and CAS to indicate this fact.

Fast Page Mode

When the DRAM in Figure is accessed, the contents of all 16,384 cells in the selected row are sensed, but only 8 bits are placed on the data lines, D7–0. This byte is selected by the column address, bitsA10–0. A simple addition to the circuit makes it possible to access the other bytes in the same row without having to reselect the row. Each sense amplifier also acts as a latch. When a row address is applied, the contents of all cells in the selected row are loaded into the corresponding latches. Then, it is only necessary to apply different column addresses to place the different bytes on the data lines. This arrangement leads to a very useful feature. All bytes in the selected row can be transferred in sequential order by applying a consecutive sequence of column addresses under the control of successive CAS signals. Thus, a block of data can be transferred at a much faster rate than can be achieved for transfers involving random addresses. **The block transfer capability is referred to as the fast page mode feature**. (A large block of data is often called a page.)



Synchronous DRAM

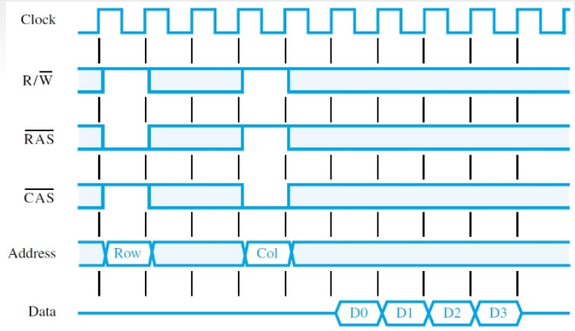


Synchronous DRAM

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). Their structure is shown in Figure in the previous slide. 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

Latency and Bandwidth

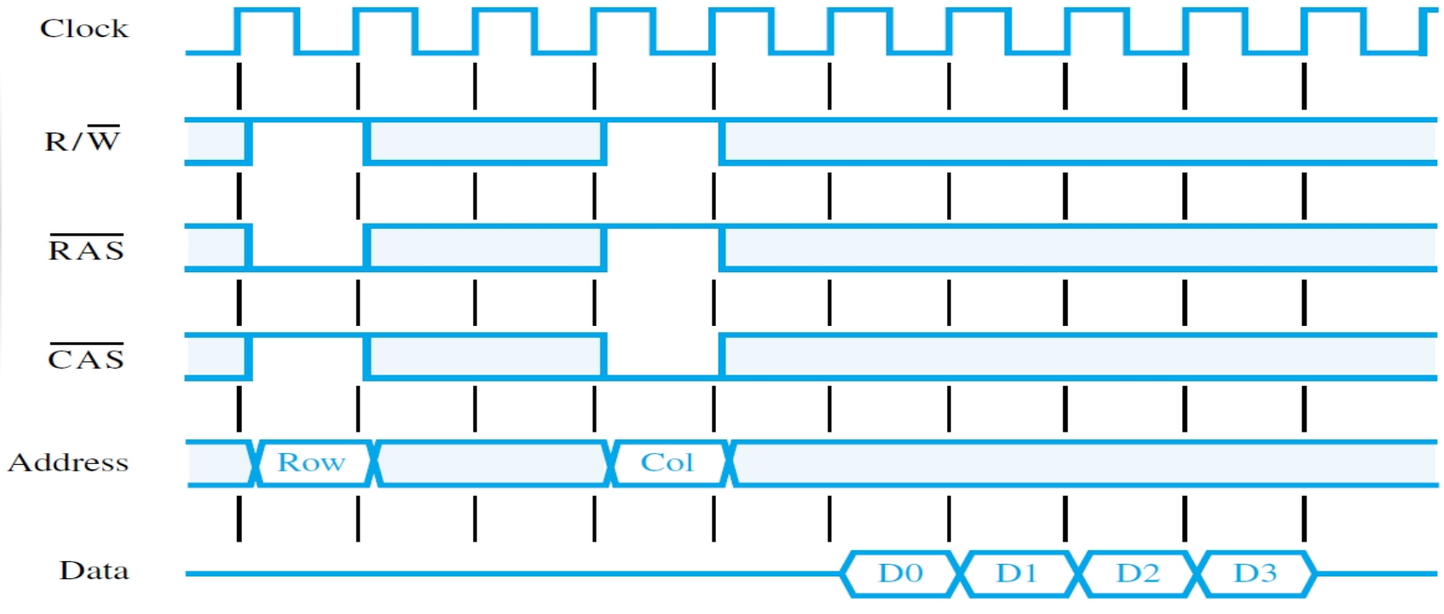
Data transfers to and from the main memory often involve blocks of data. The speed of these transfers has a large impact on the performance of a computer system. The memory access time defined earlier is not sufficient for describing the memory's performance when transferring blocks of data. **During block transfers, memory latency is the amount of time it takes to transfer the first word of a block.** The time required to transfer a complete block depends also on the rate at which successive words can be transferred and on the size of the block. The time between successive words of a block is much shorter than the time needed to transfer the first word. For instance, in the timing diagram in the Figure, the access cycle begins with the assertion of the RAS signal. The first word of data is transferred five clock cycles later. Thus, the latency is five clock cycles. If the clock rate is 500 MHz, then the latency is 10 ns. The remaining three words are transferred in consecutive clock cycles, at the rate of one word every 2 ns. The example above illustrates that we need a parameter other than memory latency to describe the memory's performance during block transfers. **A useful performance measure is the number of bits or bytes that can be transferred in one second. This measure is often referred to as the memory bandwidth.** It depends on the speed of access to the stored data and on the number of bits that can be accessed in parallel. The rate at which data can be transferred to or from the memory depends on the bandwidth of the system interconnections. For this reason, the interconnections used always ensure that the bandwidth available for data transfers between the processor and the memory is very high.

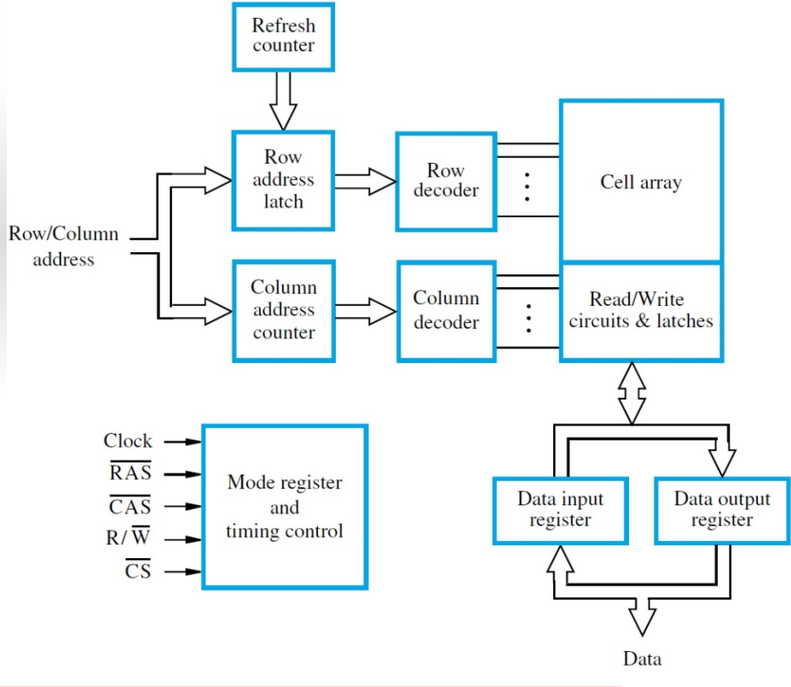


Timing diagram for a typical burst read of length 4 from SDRAM

In the timing diagram in the Figure, the access cycle begins with the assertion of the RAS signal. The first word of data is transferred five clock cycles later. Thus, the latency is five clock cycles. If the clock rate is 500 MHz, then the latency is 10 ns. The remaining three words are transferred in consecutive clock cycles, at the rate of one word every 2 ns.

for one clock cycle == 1/500 X 10*-6 === 2ns
for 5 clock cycles === 10ns (latency)





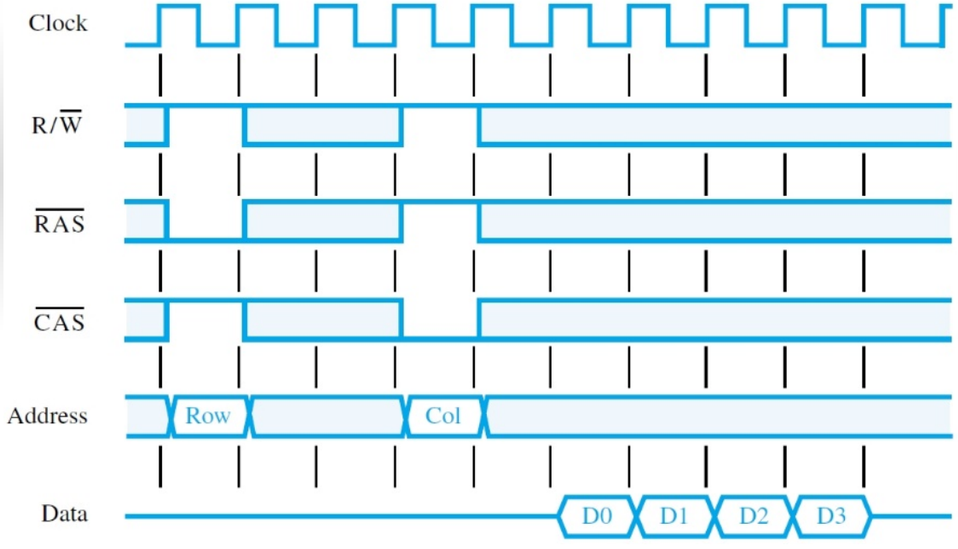
Problem

Consider a main memory constructed with SDRAM chips that have timing requirements depicted in Figure ..., except that the burst length is 8. Assume that 32 bits of data are transferred in parallel. If a 133-MHz clock is used, how much time does it take to transfer:

(*a*) 32 bytes of data

(*b*) 64 bytes of data

What is the latency in each case?



Answer

Consider a main memory constructed with SDRAM chips that have timing requirements depicted in Figure , except that the burst length is 8. Assume that 32 bits of data are transferred in parallel. If a 133-MHz clock is used, how much time does it take to transfer:

(*a*) 32 bytes of data

(*b*) 64 bytes of data

What is the latency in each case?

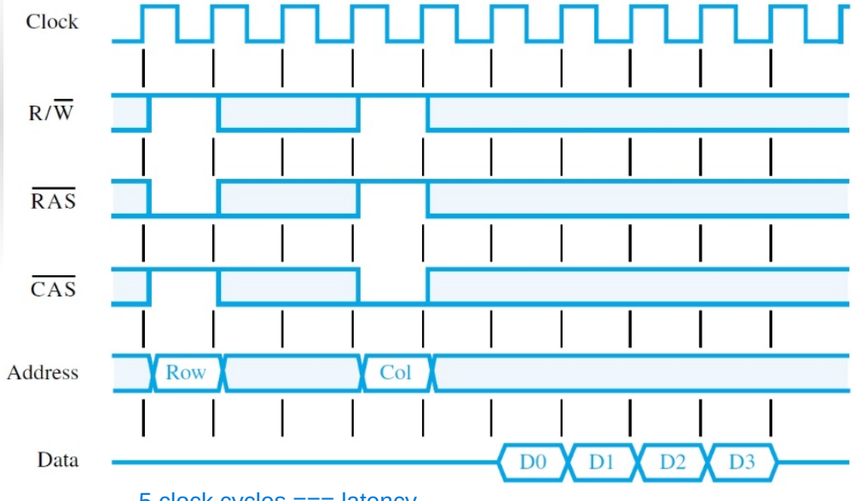
for 32 bytes .....since every clock cycle takes 4 bytes -->8 are required to transfer 32 bytes of memory
so no. of clock cycles will be 8+5
for 64=== 5+8+8
but latency in both the cases will be the same !!

(a) It takes 5 + 8 = 13 clock cycles.

$$\text{Total time} = \frac{13}{(133 \times 10^6)} = 0.098 \times 10^{-6}\ \text{s} = 98\ \text{ns}$$

$$\text{Latency} = \frac{5}{(133 \times 10^6)} = 0.038 \times 10^{-6}\ \text{s} = 38\ \text{ns}$$

(b) It takes twice as long to transfer 64 bytes, because two independent 32-byte transfers have to be made. The latency is the same, i.e. 38 ns.



5 clock cycles === latency

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

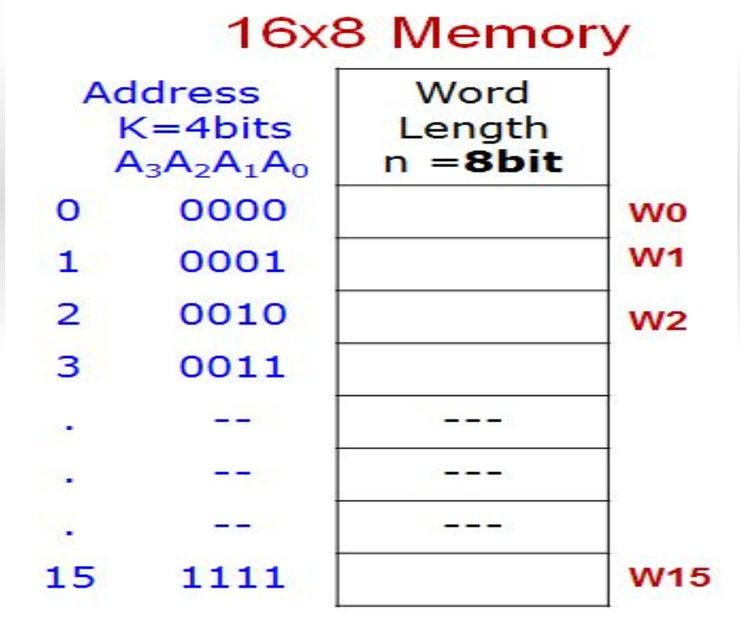
Rambus Memory

The rate of transferring data between the memory and the processor is a function of both the bandwidth of the memory and the bandwidth of its connection to the processor. **Rambus is a memory technology that achieves a high data transfer rate by providing a high-speed interface between the memory and the processor.** One way for increasing the bandwidth of this connection is to use a wider data path. However, this requires more space and more pins, increasing system cost. The alternative is to **use fewer wires with a higher clock speed**. This is the approach taken by Rambus. The key feature of Rambus technology is the use of a differential-signaling technique to transfer data to and from the memory chips. In Rambus technology, signals are transmitted using small voltage swings of 0.1 V above and below a reference value. Several versions of this standard have been developed, with clock speeds of up to 800 MHz and data transfer rates of several gigabytes per second. Rambus technology competes directly with the DDR SDRAM technology. Each has certain advantages and disadvantages. A nontechnical consideration is that the specification of DDR SDRAM is an open standard that can be used free of charge. Rambus, on the other hand, is a proprietary scheme that must be licensed by chip manufacturers.

Rambus technology competes directly with the DDR SDRAM technology. Each has certain advantages and disadvantages. A nontechnical consideration is that the specification of DDR SDRAM is an open standard that can be used free of charge. Rambus, on the other hand, is a proprietary scheme that must be licensed by chip manufacturers.

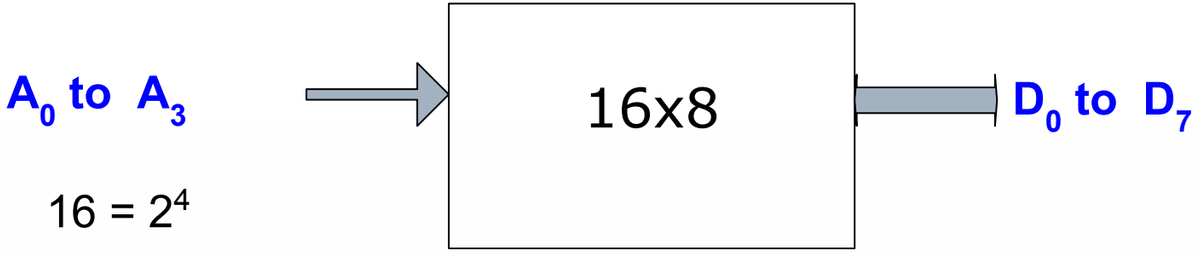
Structure of Large Memories

Design a Memory of Size 16x8 using 4x8 memory chips



Design a Memory of Size 16x8 using 4x8 memory chips

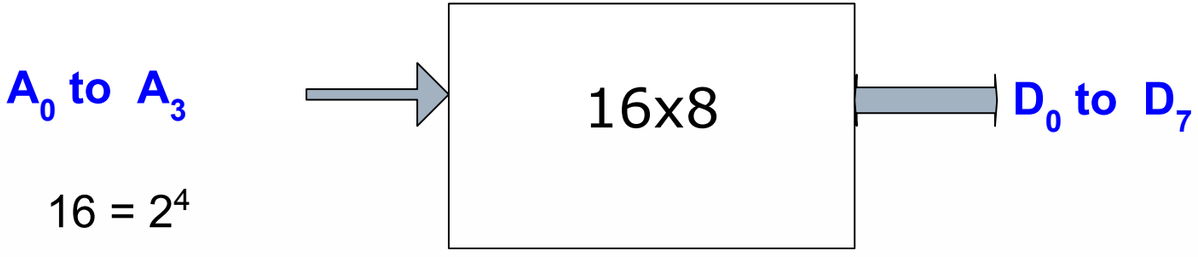
To Design



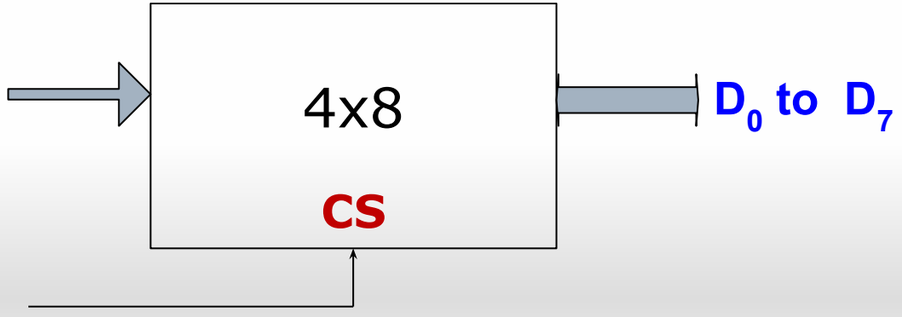
$16 = 2^4$

Design a Memory of Size 16x8 using 4x8 memory chips

To Design



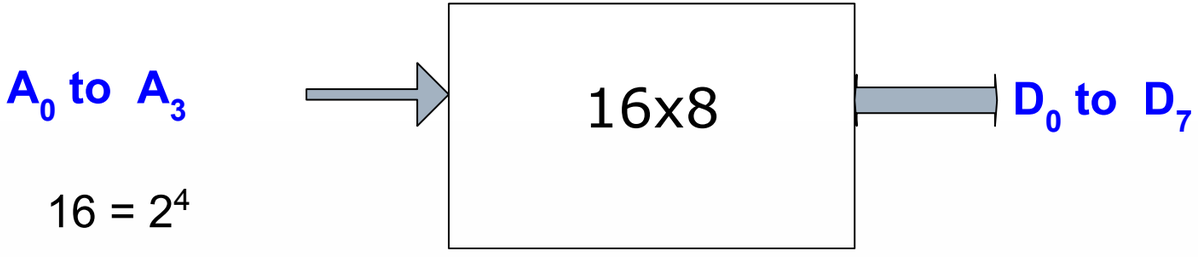
Given



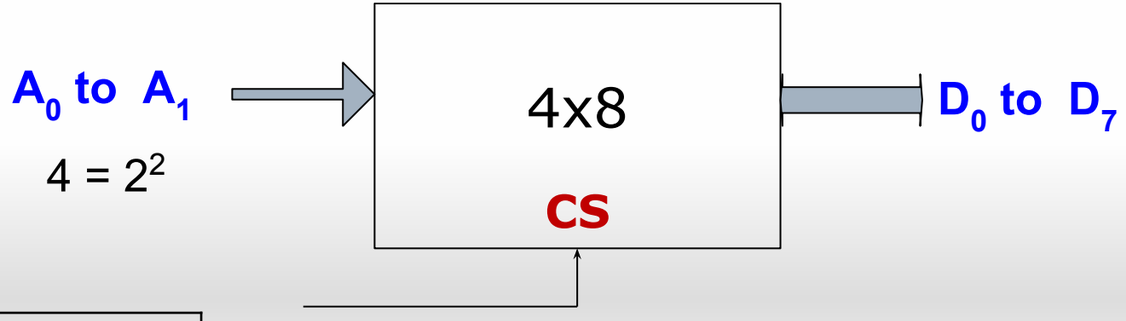
CS: Chip Select

Design a Memory of Size 16x8 using 4x8 memory chips

To Design



Given



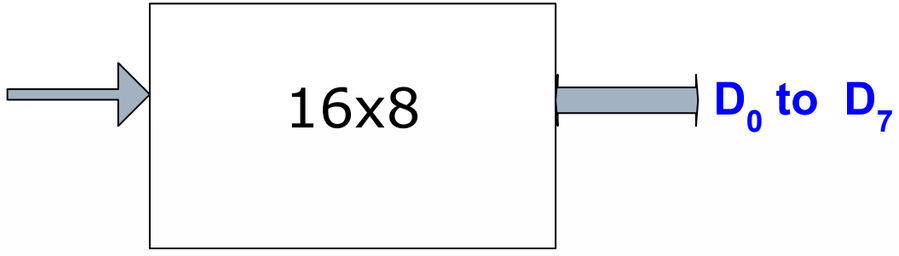
4x 8 Memory

Address K=2bits		Word Length n =8bits
0	00	0x38
1	01	0x27
2	10	0x50
3	11	0x46

CS: Chip Select

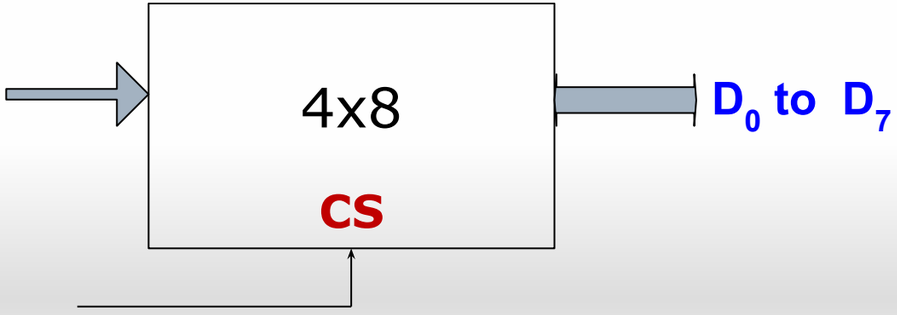
Design a Memory of Size 16x8 using 4x8 memory chips

To Design



$16 = 2^4$

Given



$4 = 2^2$

4x 8 Memory

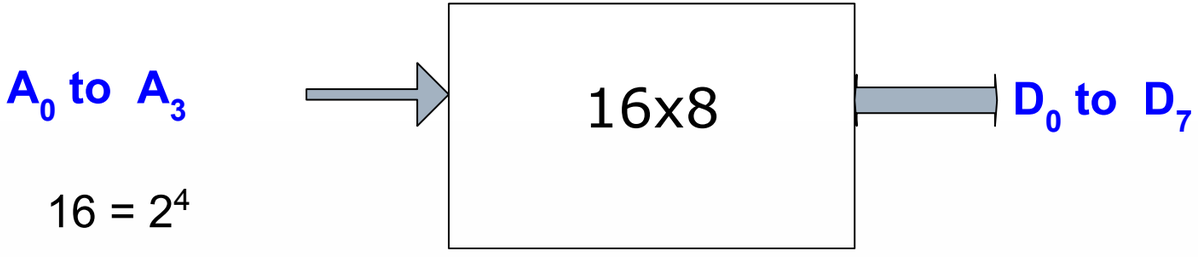
Address K=2bits		Word Length n =8bits
0	00	0x38
1	01	0x27
2	10	0x50
3	11	0x46

A_1	A_0	CS	D_7 to D_0
0	0	1	0x38
0	1	1	0x27
1	0	1	0x50
1	1	1	0x46

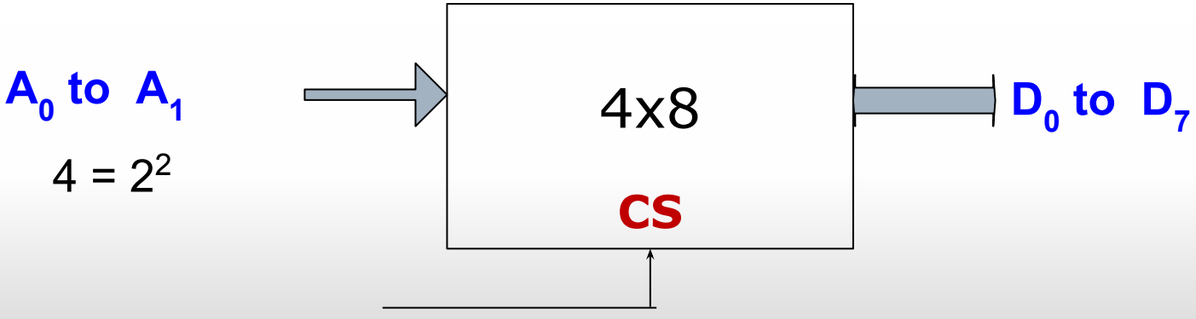
CS: Chip Select

Design a Memory of Size 16x8 using 4x8 memory chips

To Design



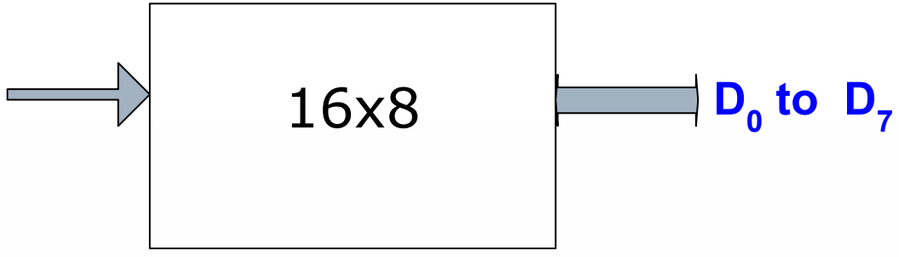
Given



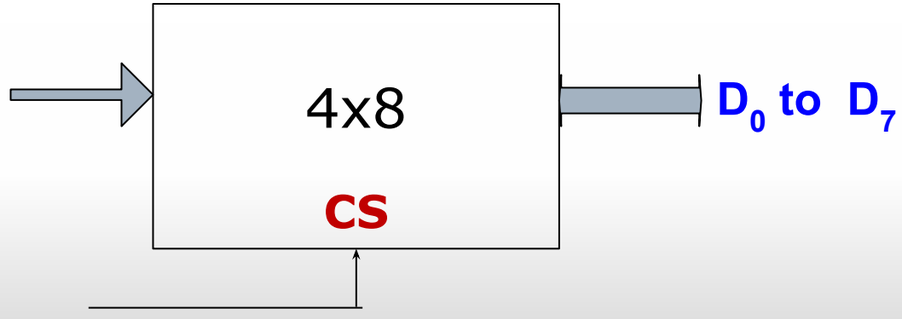
Number of chips required =	**16x8** / **4x8**	= **4** i.e., Four chips of size 4x8 are required to design 16x8

Design a Memory of Size 16x8 using 4x8 memory chips

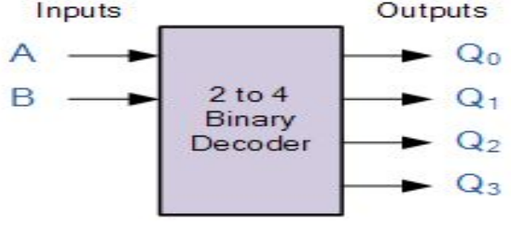
To Design



Given



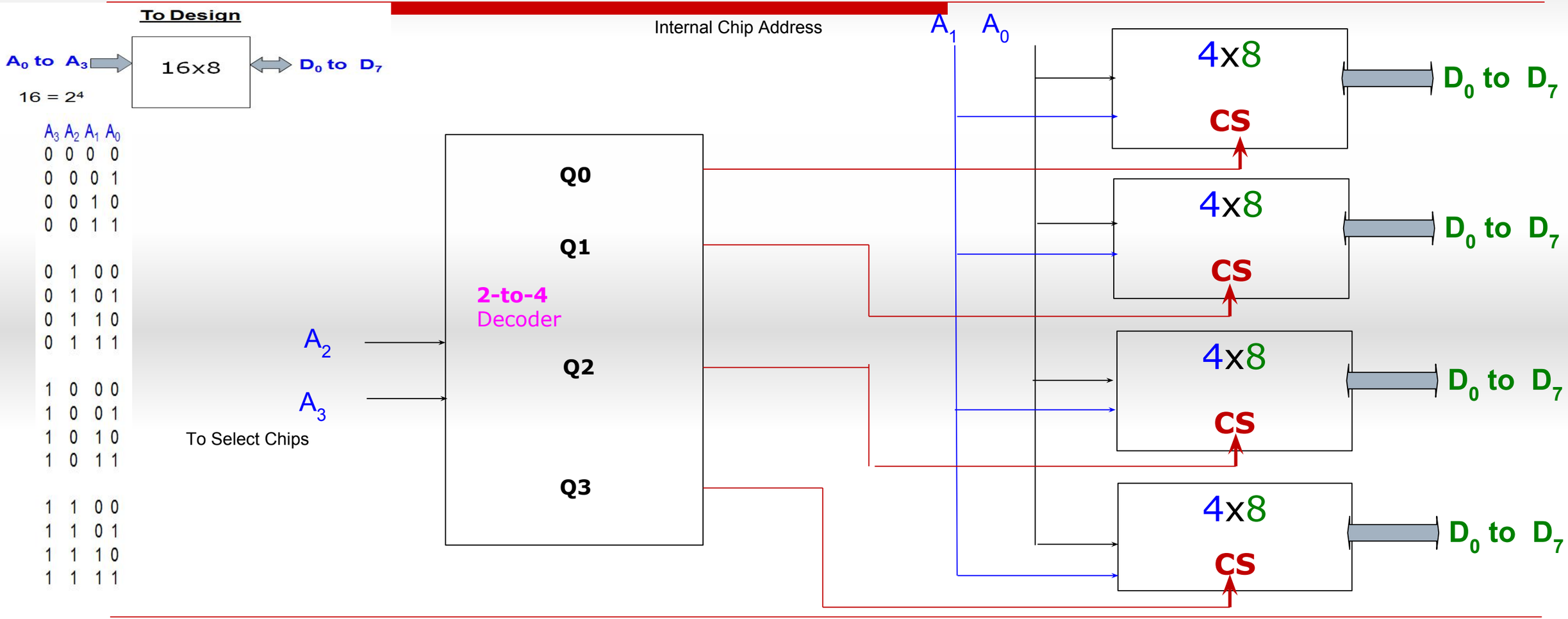
Number of chips required =	**16x8** / **4x8**	= **4** i.e., Four chips of size 4x8 are required to design 16x8



Truth Table

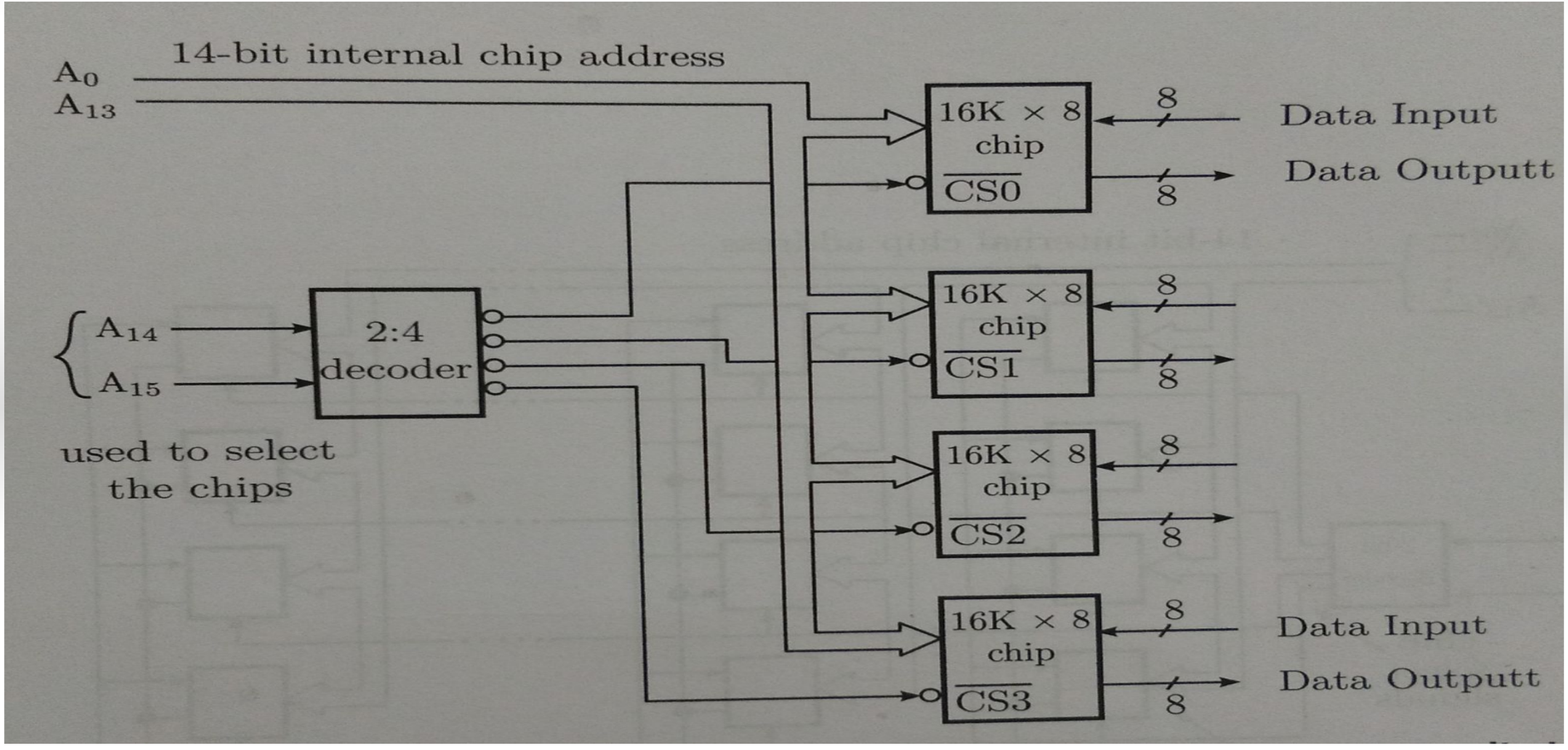
A	B	Q_0	Q_1	Q_2	Q_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Design a Memory of Size 16x8 using 4x8 memory chips



Design a Memory of Size 64Kx8 using 16Kx8 memory chips

Design a Memory of Size 64Kx8 using 16Kx8 memory chips



Question

A memory system of size 26K bytes is required to be designed using memory chips which have 12 address lines and 4 data lines each. The number of such chips required to design the memory system is

(a) 2 (b) 4 (c) 8 (d) 13

Question

A memory system of size 26K bytes is required to be designed using memory chips which have 12 address lines and 4 data lines each. The number of such chips required to design the memory system is

(a) 2 (b) 4 (c) 8 (d) 13

Total Memory System Size = 26K Bytes = 26K address locations × 8 bit data lines — ①

Size of Memory chip = 2^{12} address locations × 4 data lines — ②

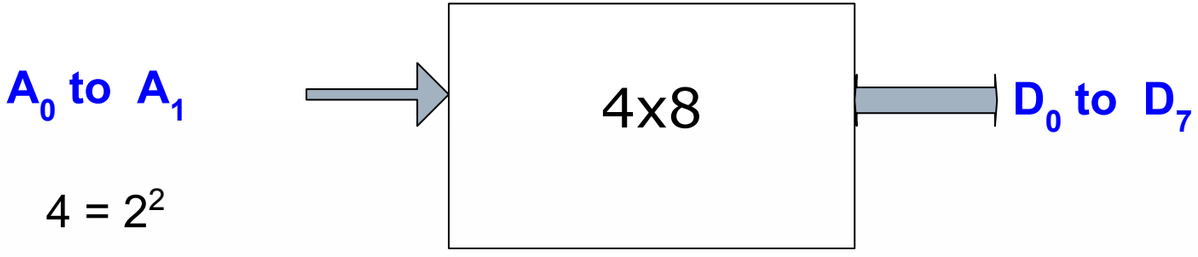
divide ① by ②

$$= \frac{26 \times 2^{10} \times 8}{2^{12} \times 4} = \frac{52 \times 2^2}{2^2} = \frac{52}{4} = 13 \text{ no of Memory Chips}$$

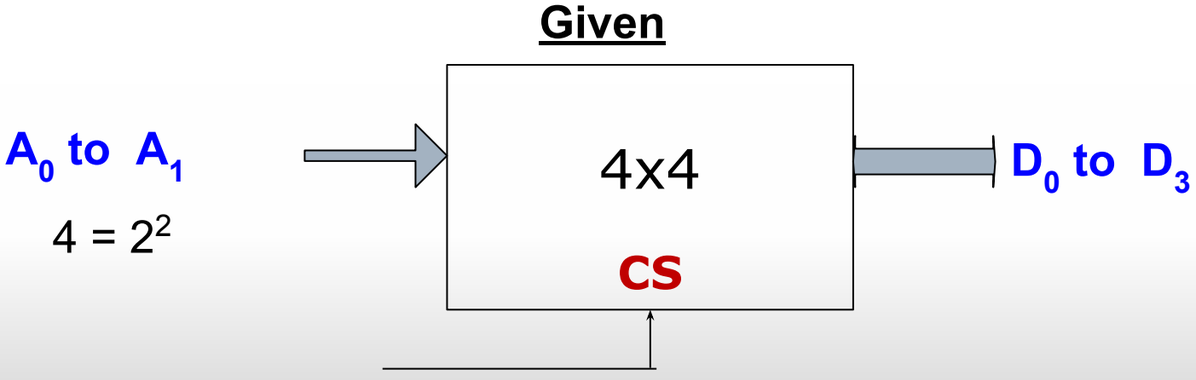
Design a Memory of Size 4x8 using 4x4 memory chips

Design a Memory of Size 4x8 using 4x4 memory chips

To Design



Given



Number of chips required =	**4x8** / **4x4**	= **2** i.e., Two chips of size 4x4 are required to design 4x8

Design a Memory of Size 4x8 using 4x4 memory chips

4x 8 Memory

	Address K=2bits	Word Length n =8bits
0	00	0x38
1	01	0x27
2	10	0x50
3	11	0x46

Design a Memory of Size 4x8 using 4x4 memory chips

4x 8 Memory

	Address K=2bits	Word Length n =8bits
0	00	0x38
1	01	0x27
2	10	0x50
3	11	0x46

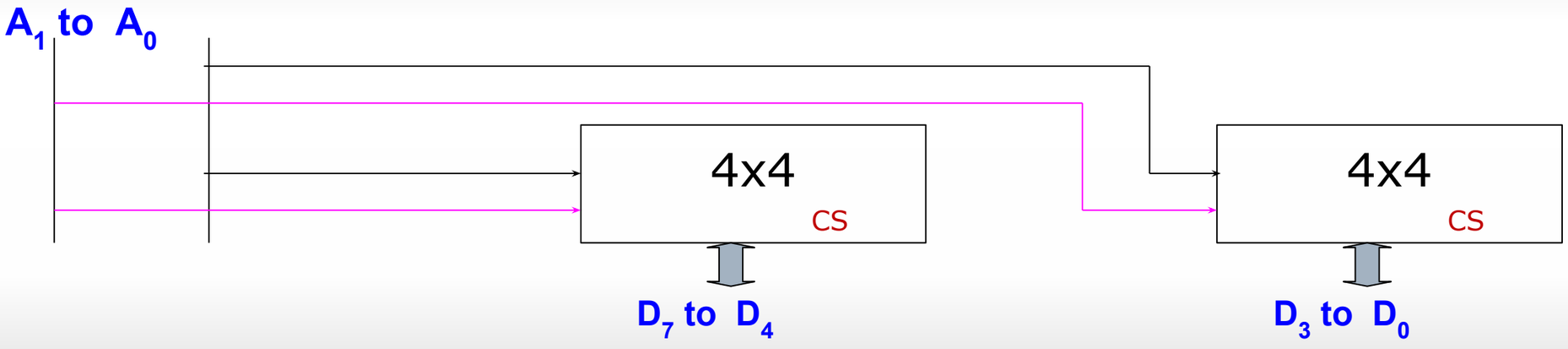
4x 4 Memory

	Address K=2bits	Word Length n =4bits
0	00	0x3
1	01	0x2
2	10	0x5
3	11	0x4

4x 4 Memory

	Address K=2bits	Word Length n =4bits
0	00	0x8
1	01	0x7
2	10	0x0
3	11	0x6

Design a Memory of Size 4x8 using 4x4 memory chips



4x 8 Memory

	Address K=2bits	Word Length n =8bits
0	00	0x38
1	01	0x27
2	10	0x50
3	11	0x46

4x 4 Memory

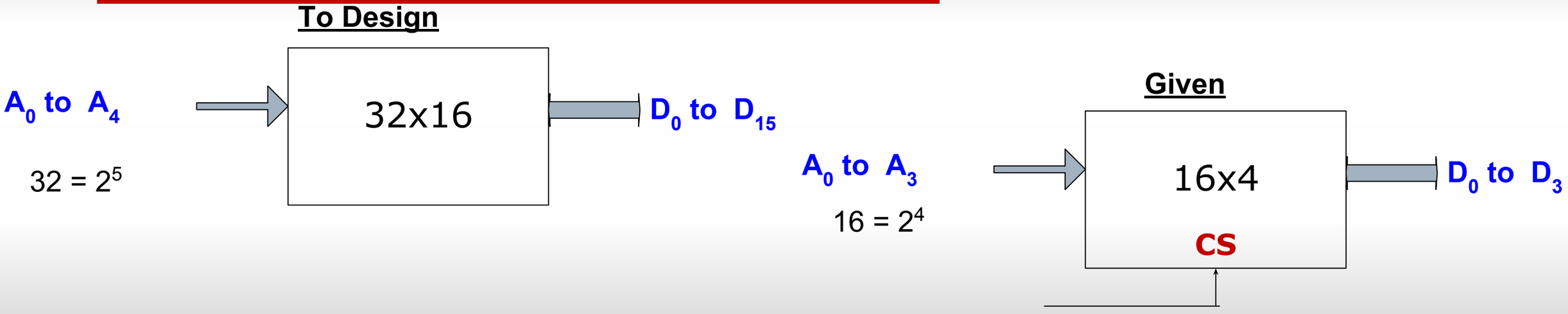
	Address K=2bits	Word Length n =4bits
0	00	0x3
1	01	0x2
2	10	0x5
3	11	0x4

4x 4 Memory

	Address K=2bits	Word Length n =4bits
0	00	0x8
1	01	0x7
2	10	0x0
3	11	0x6

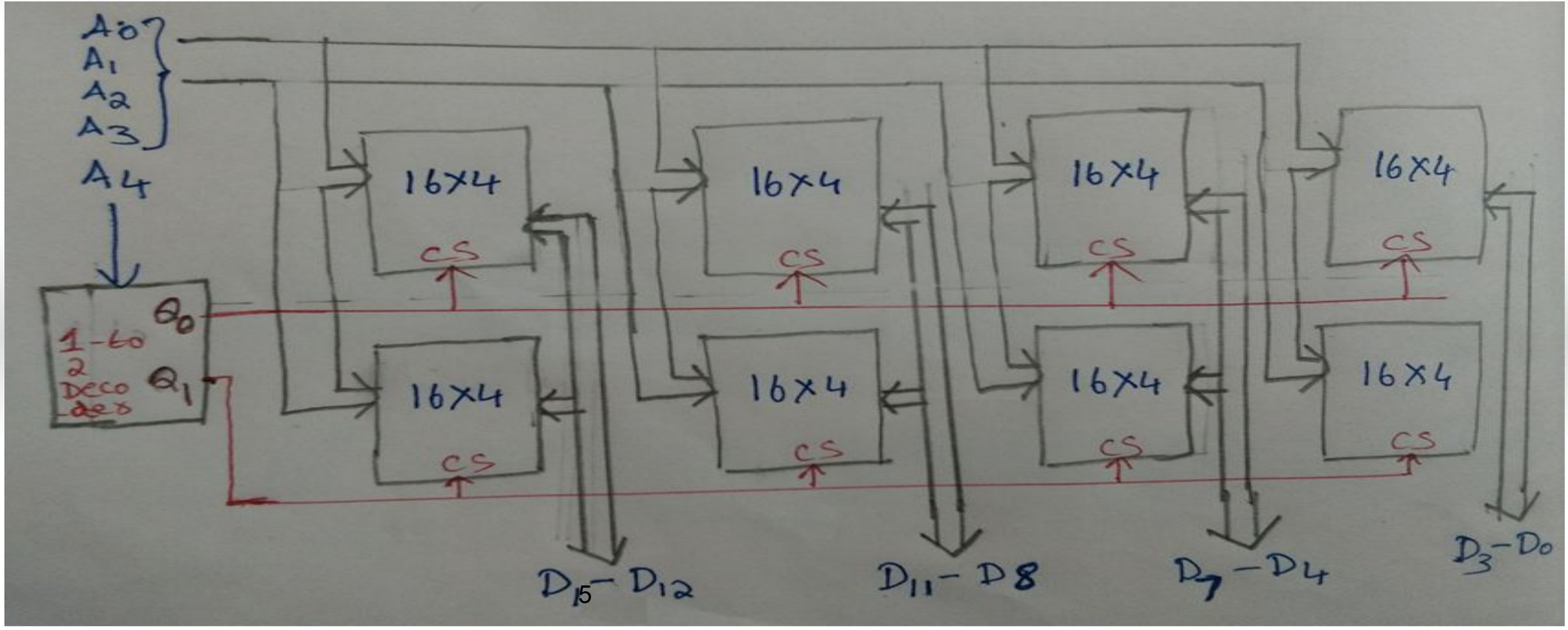
Design a Memory of Size 32x16 using 16x4 memory chips

Design a Memory of Size 32x16 using 16x4 memory chips



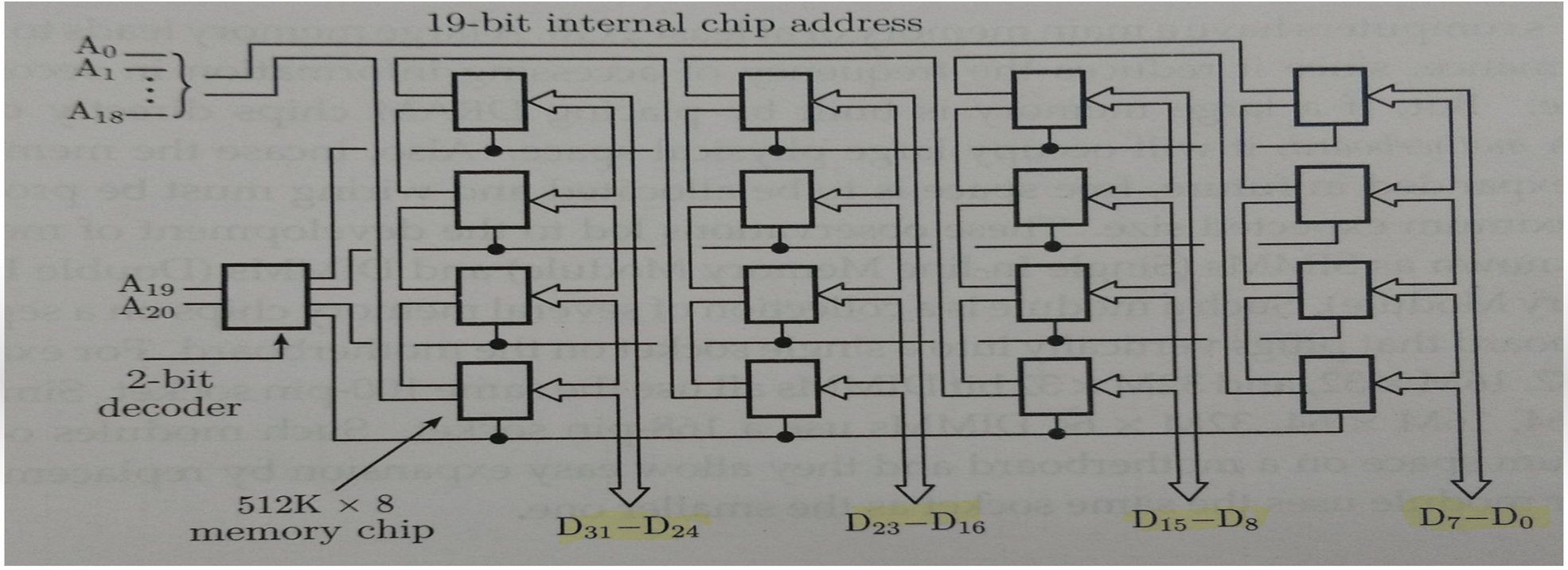
Number of chips required =	**32x16** / **16x4**	= **8** i.e., Eight chips of size 16x4 are required to design 32x16

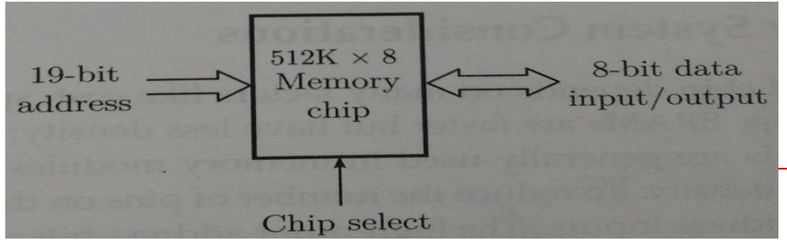
Design a Memory of Size 32x16 using 16x4 memory chips



Design a Memory of Size 2Mx32 using 512Kx8 memory chips

Design a Memory of Size 2Mx32 using 512Kx8 memory chips

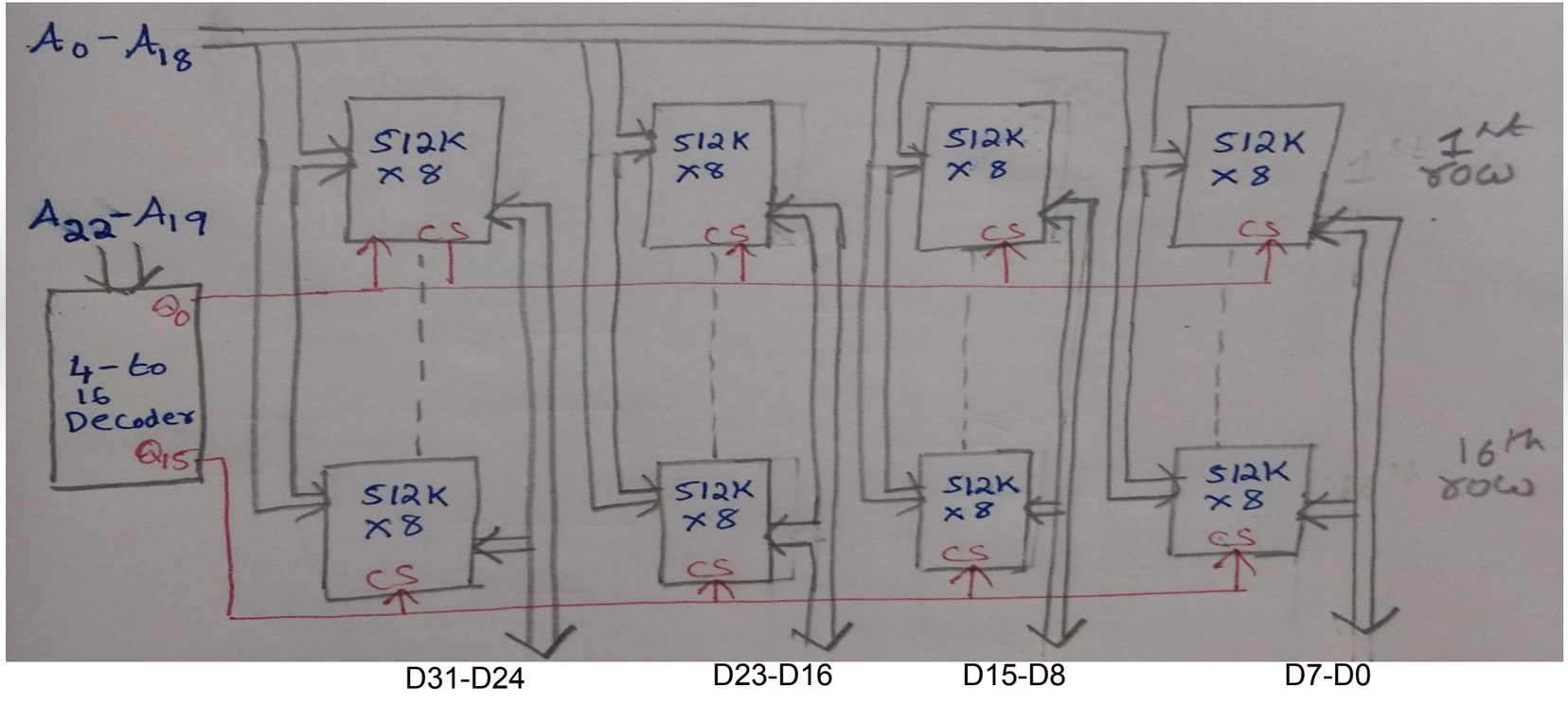




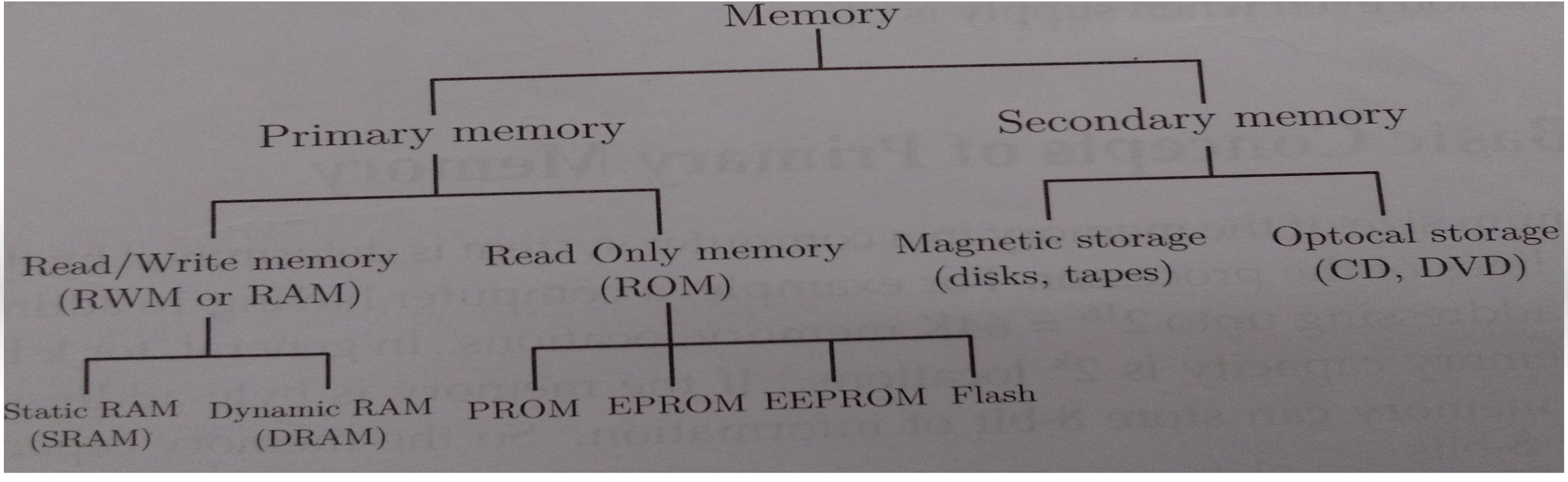
Design a Memory of Size 8Mx32 using 512Kx8 memory chips

Design a Memory of Size 8Mx32 using 512Kx8 memory chips

16 rows(of four 512 x8 chips) are needed. Address lines A18 to A0 are connected to all chips. Address lines A22 to A19 are connected to a 4-bit decoder to select one of the 16 rows.



Classification of Memory System



Difference between RAM and ROM

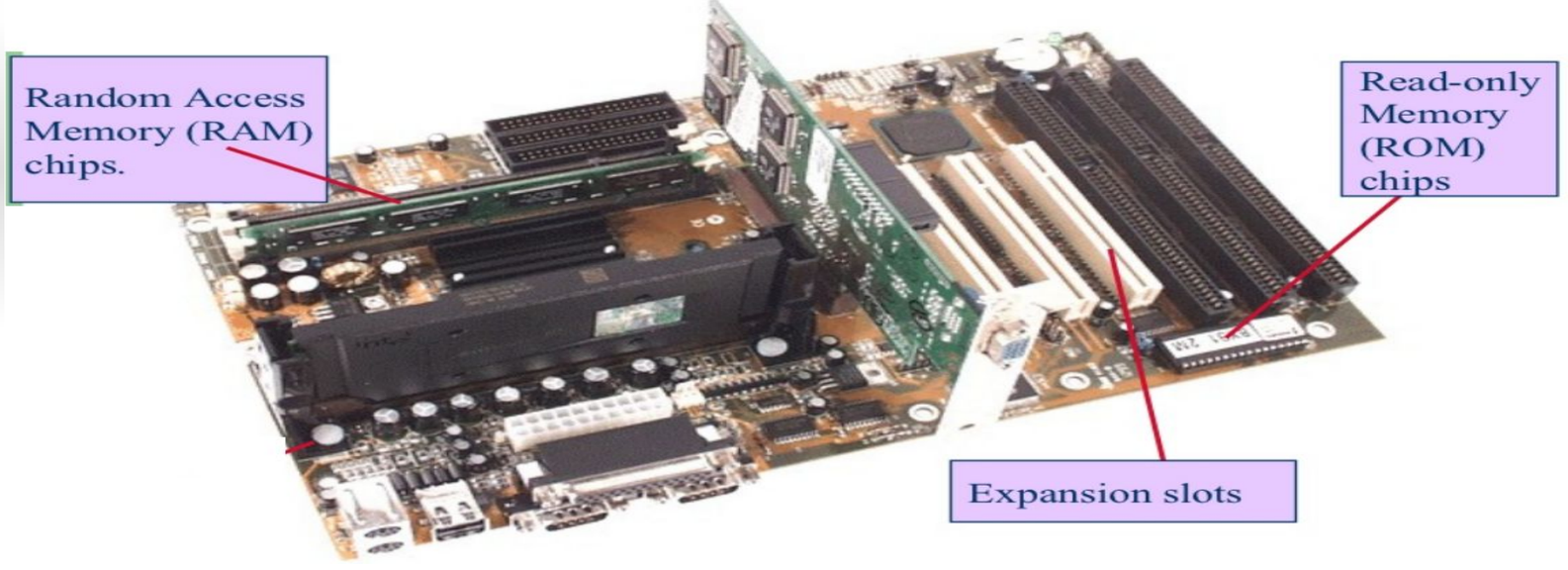
RAM

- RAM stands for **Random Access Memory**
- It is **volatile** in nature. Its contents are erased when power is turned off.

ROM

- ROM stands for **Read only Memory**
- It is **non-volatile** in nature. Its contents are non erased when power is turned off.

Read-Only Memories (ROMs)

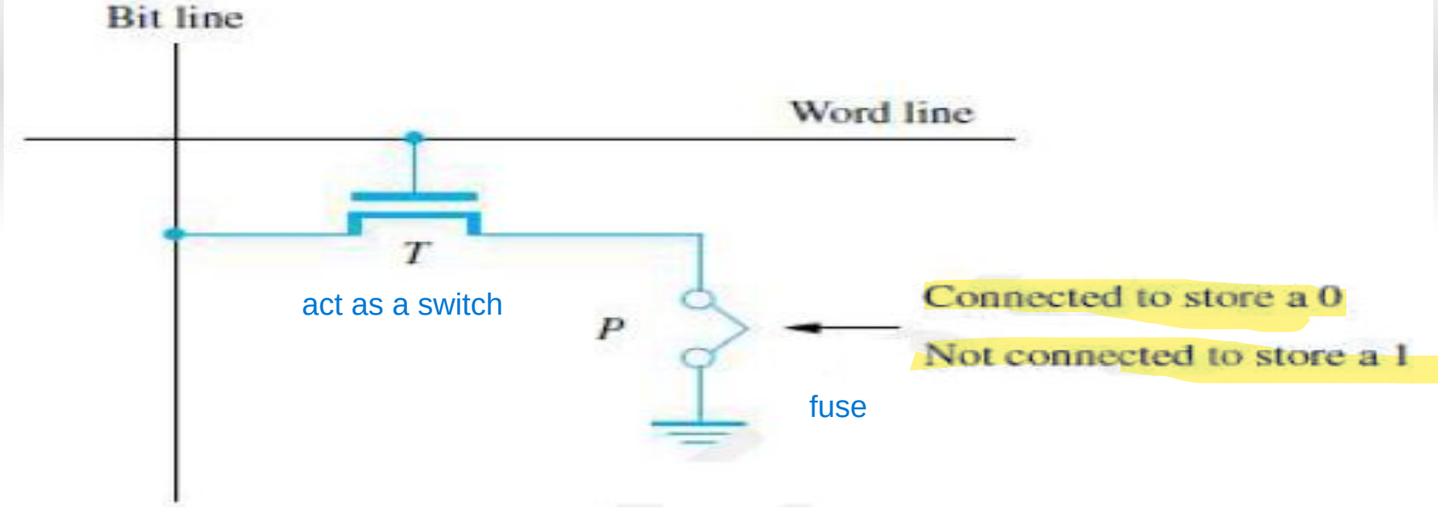


Read-Only Memories (ROMs)

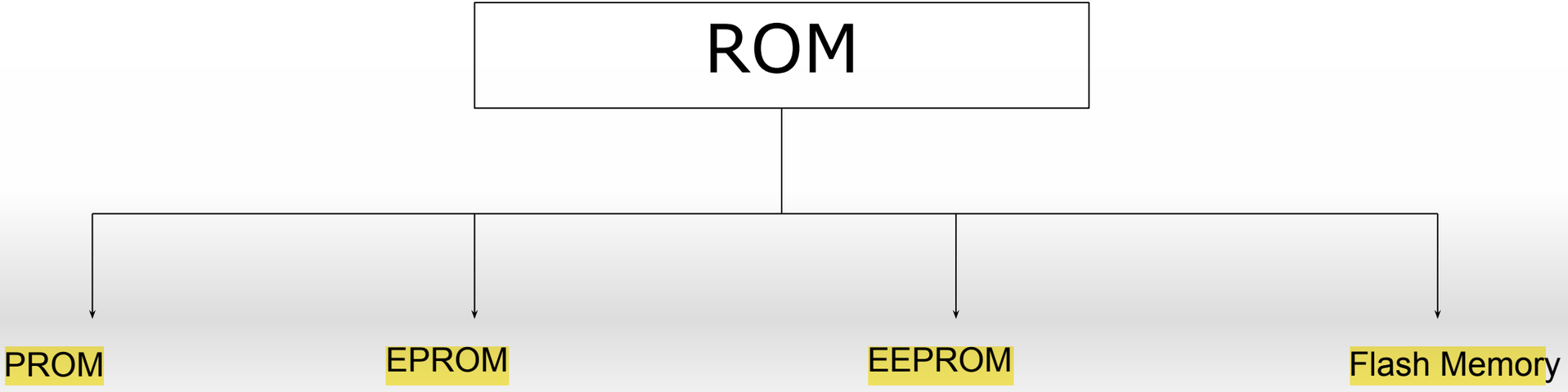
- Many applications need memory devices to retain contents after the power is turned off.
  - For example, computer is turned on, the **operating system** must be loaded from the disk into the memory.
  - Store instructions which would load the OS from the disk.
  - Need to store these instructions so 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).

ROM Cell Structure

- Both SRAM and DRAM chips are volatile, i.e. They lose the stored information if power is turned off.
- Many application requires non-volatile memory which retains the stored information if power is turned off.
- For ex:
  OS software has to be loaded from disk to memory i.e. it requires non-volatile memory.
- Non-volatile memory is used in embedded system.
- Since the normal operation involves only reading of stored data, a memory of this type is called ROM.
  - **At Logic value '0'** → Transistor(T) is connected to the ground point (P).
    Transistor switch is closed & voltage on bit-line nearly drops to zero (Figure).
  - **At Logic value '1'** → Transistor switch is open.
    The bit-line remains at high voltage.



Types of Read-Only Memories



Types of Read-Only Memories

- Read-Only Memory:
  - Data are written into a ROM when it is manufactured.

1. Programmable Read-Only Memory (PROM):
   - Allow the data to be loaded by a user.
   - Process of inserting the data is irreversible. cannot be erased
   - Storing information specific to a user in a ROM is expensive.
   - Providing programming capability to a user may be better.

2. 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 (Ultraviolet) light**. disadvantage

Read-Only Memories (Contd.,)

3. Electrically Erasable Programmable Read-Only Memory (EEPROM):

- To erase the contents of EPROMs, they have to be exposed to ultraviolet light and Physically removed from the circuit.
- But EEPROMs the contents can be **stored and erased electrically**.

4. Flash memory:

- Has similar approach to EEPROM. chip size is usually small
- 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-drive**n.
- Single flash chips are not sufficiently large, so larger memory modules are implemented using flash cards and flash drives.

Flash Card	Flash Drive
Flash cards with a USB interface are widely used and are commonly known as memory keys. They come in a variety of memory sizes. Larger cards may hold as much as 32 Gbytes	Larger flash memory modules have been developed to replace hard disk drives, and hence are called flash drives. Currently, the capacity of flash drives is on the order of 64 to 128 Gbytes

Question

PROM stands for

a) Programmable Read Only Memory.

b) Pre-fed Read Only Memory.

c) Pre-required Read Only Memory.

d) Programmed Read Only Memory.

Question

PROM stands for

a) Programmable Read Only Memory.

b) Pre-fed Read Only Memory.

c) Pre-required Read Only Memory.

d) Programmed Read Only Memory.

Answer:a

Explanation: It allows the user to program the ROM.

Question

EEPROM stands for Electrically Erasable Programmable Read Only Memory.

a) True

b) False

Question

EEPROM stands for Electrically Erasable Programmable Read Only Memory.

a) True

b) False

Answer:a

Question

Which of the following is true with respect to EEPROM?

a. contents can be erased byte wise only

b. contents of full memory can be erased together

c. contents can be erased using ultra violet rays

d. contents can not be erased

Question

Which of the following is true with respect to EEPROM?

a. contents can be erased byte wise only

b. contents of full memory can be erased together

c. contents can be erased using ultra violet rays

d. contents can not be erased

Answer

(c).

contents can be erased using ultra violet rays

Question

The memory devices which are similar to EEPROM but differ in the cost effectiveness is _____.

a) Memory sticks

b) Blue-ray devices

c) Flash memory

d) CMOS

Question

The memory devices which are similar to EEPROM but differ in the cost effectiveness is _____.

a) Memory sticks

b) Blue-ray devices

c) Flash memory

d) CMOS

Answer:c

Explanation: The flash memory functions similar to the EEPROM.

Question

The flash memories find application in _____.

a) Super computers

b) Mainframe systems

c) Distributed systems

d) Portable devices easily handable bcz they require low power requirement

Question

The flash memories find application in _____.

a) Super computers

b) Mainframe systems

c) Distributed systems

d) Portable devices

Answer:d

Explanation: The flash memories low power requirement enables them to be used in a wide range of hand held devices.

Question

The reason for the fast operating speeds of the flash drives is

a) The absence of any movable parts.

b) The integarated electronic hardware.

c) The improved bandwidth connection.

d) All of the above.

Question

The reason for the fast operating speeds of the flash drives is

a) The absence of any movable parts.

b) The integarated electronic hardware.

c) The improved bandwidth connection.

d) All of the above.

Answer:a

Explanation: Since the flash drives have no movable parts their access and seeks times are reasonably reduced.

Direct Memory Access

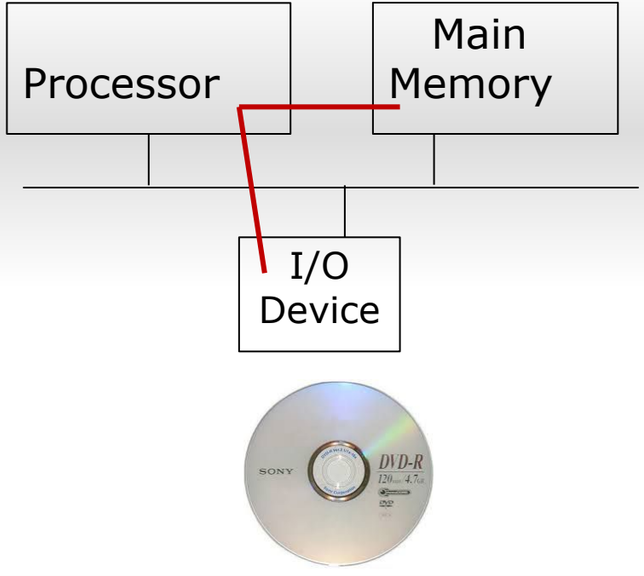
Direct memory Access (DMA)

Direct memory access (DMA) is a method that allows an input/output (I/O) device to send or receive data directly to or from the main memory, without the continuous intervention by the processor.

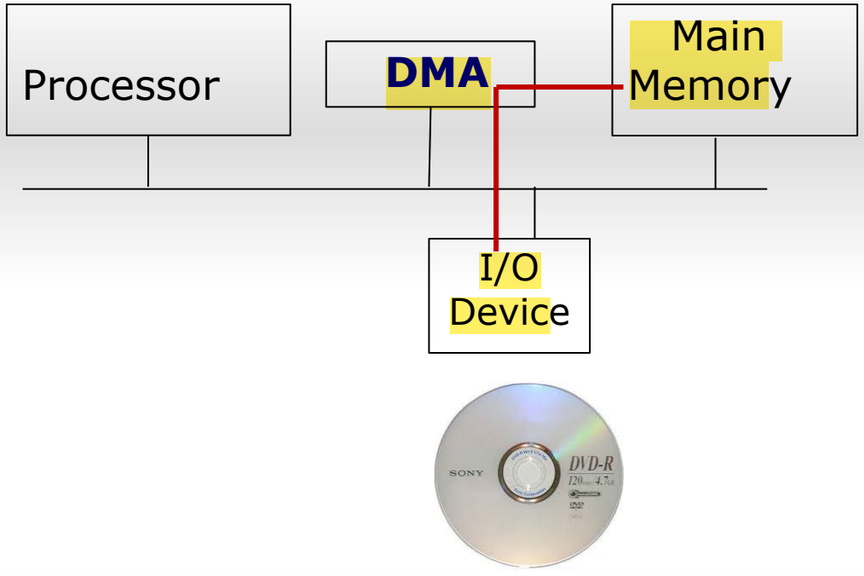
To transfer large blocks of data at high speed, DMA approach will be used.

when large memories have to be transferred DMA is used

Without DMA data transfer



With DMA data transfer



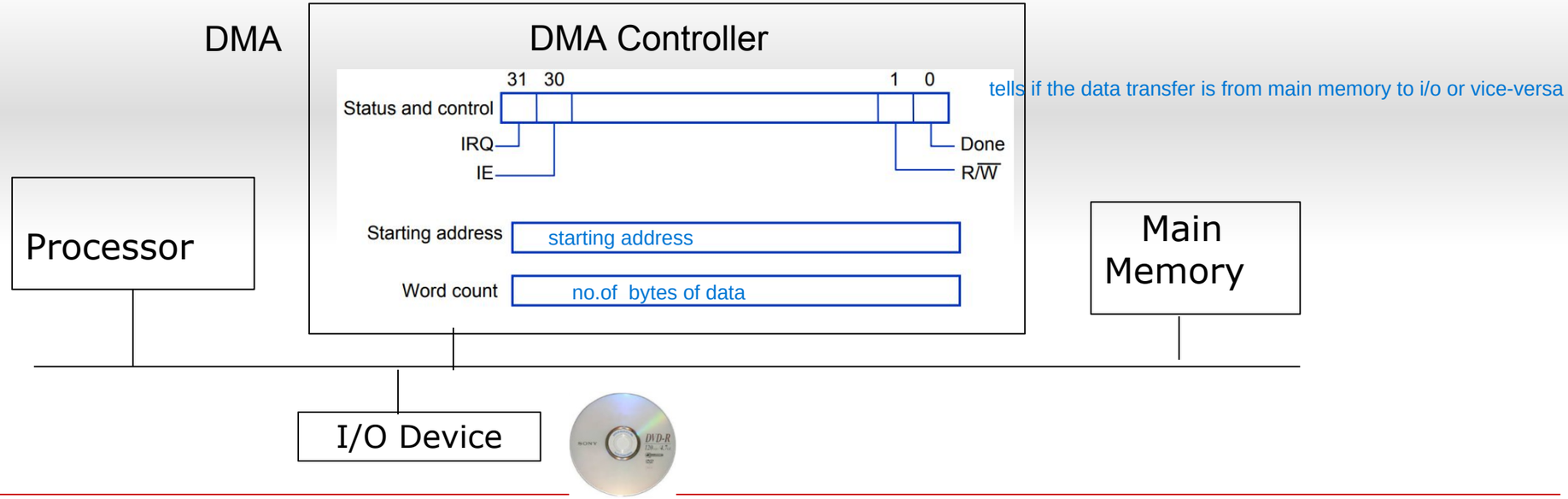
Direct memory Access

Direct Memory Access (DMA): A special control unit may be provided to transfer a block of data directly between an I/O device and the main memory, without continuous intervention by the processor.

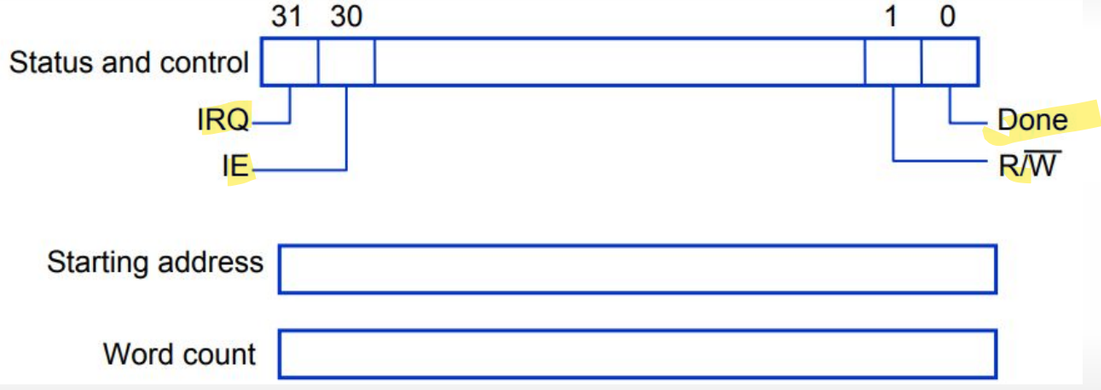
Control unit which performs these transfers is a part of the I/O device's interface circuit. This control unit is called as a DMA controller.

DMA controller performs functions that would be normally carried out by the processor:

- For each word, it provides the memory address and all the control signals.
- To transfer a block of data, it increments the memory addresses and keeps track of the number of transfers.



DMA Operation



Read-- R/W ==1 memory to i/o

- DMA interface has three registers
  1) First register is used for storing starting-address.
  2) Second register is used for storing word-count.
  3) Third register contains status- & control-flags.

- The R/W bit determines direction of transfer.
  If R/W=1, controller performs a read-operation (i.e. it transfers data from memory to I/O),
  Otherwise, controller performs a write-operation (i.e. it transfers data from I/O to memory).
- If Done=1, the controller
  → has completed transferring a block of data and
  → is ready to receive another command. (IE → Interrupt Enable).
- If IE=1, controller raises an interrupt after it has completed transferring a block of data.
- If IRQ=1, controller requests an interrupt.

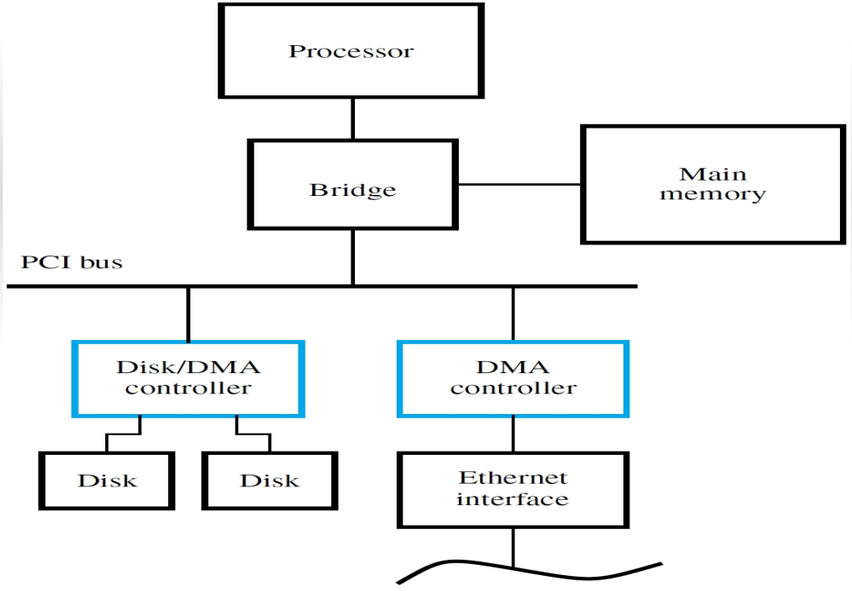
DMA Operation

To start a DMA transfer of a block of data from the main memory to one of the disks, an Operating System routine writes the address and word count information into the registers of the disk controller. The DMA controller proceeds independently to implement the specified operation. When the transfer is completed, this fact is recorded in the status and control register of the DMA channel by setting the Done bit. At the same time, if the IE bit is set, the controller sends an interrupt request to the processor and sets the IRQ bit. The status register may also be used to record other information, such as whether the transfer took place correctly or errors occurred.

Use of DMA controllers in a computer system

Figure shows how DMA controllers may be used in a computer system. One DMA controller connects a high-speed Ethernet to the computer's I/O bus. The disk controller, which controls two disks, also has DMA capability and provides two DMA channels. It can perform two independent DMA operations, as if each disk had its own DMA controller. The registers needed to store the memory address, the word count, and so on, are duplicated, so that one set can be used with each disk.

Note: Peripheral Component Interconnect (**PCI**)



Question

DMA stands for

a. Direct memory access

b. Direct memory allocation

c. Data memory access

d. Data memory allocation

Question

DMA stands for

a. **Direct memory access**

b. Direct memory allocation

c. Data memory access

d. Data memory allocation

Question

The DMA transfers are performed by a control circuit called as

a) Device interface
b) DMA controller
c) Data controller
d) Overlooker

Question

The DMA transfers are performed by a control circuit called as

a) Device interface

b) DMA controller

c) Data controller

d) Overlooker

Answer: (b)

Explanation: The Controller performs the functions that would normally be carried out by the processor.

Question

In a DMA write operation the data is transferred

a. from I/O to memory

b. from memory to I/O

c. from memory to memory

d. from I/O to I/O

Question

In a DMA write operation the data is transferred

a. **from I/O to memory**

b. from memory to I/O

c. from memory to memory

d. from I/O to I/O

Question

When the R/W bit of the status register of the DMA controller is set to 1,

a) Read operation is performed

b) Write operation is performed

Question

When the R/W bit of the status register of the DMA controller is set to 1,

a) **Read operation is performed**

b) Write operation is performed

Question

After the completion of the DMA transfer the processor is notified by
a) Acknowledge signal
b) Interrupt signal
c) WMFC signal
d) None of the above

Question

After the completion of the DMA transfer the processor is notified by
a) Acknowledge signal
b) Interrupt signal
c) WMFC signal
d) None of the above

Answer: (b)
Explanation: The controller raises an interrupt signal to notify the processor that the transfer was complete.

Cache Memory

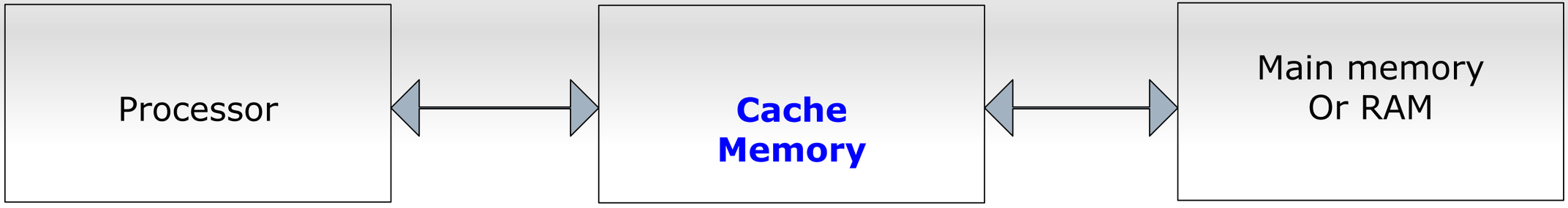
What is the meaning of the word Cache ?
Cache: A **collection of items** of the same type stored in a **hidden** or **inaccessible place**

CPU MEMORY

What is Cache memory ?

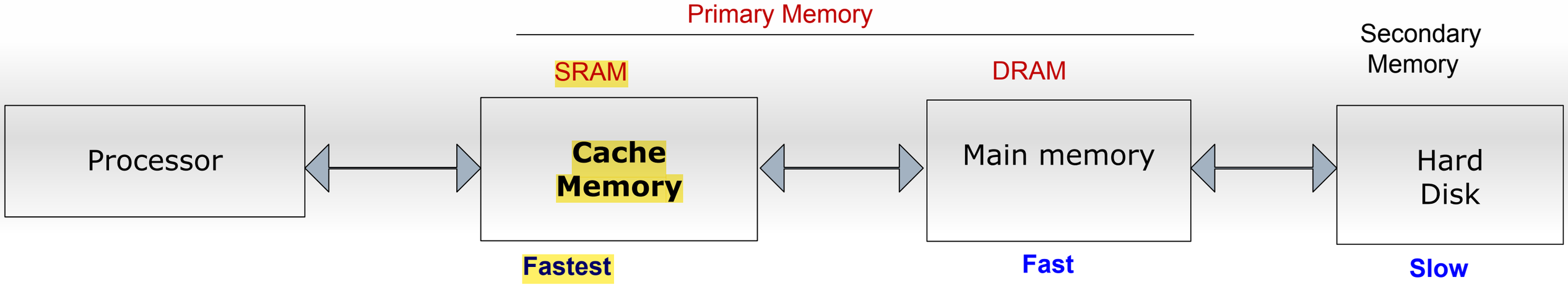
VERY EXPENSIVE

Cache memory is a small-sized volatile computer memory that provides **high-speed data access** to a processor and stores **frequently used computer programs, applications and data.** Cache memory also called CPU memory, which is placed between Random access memory (RAM) and a computer microprocessor.



Why we need Cache Memory ? Or Use of Cache Memory

Cache memory is used to **increase the performance of the Computer.** It holds data and instructions retrieved from RAM to provide faster access to the Processor.

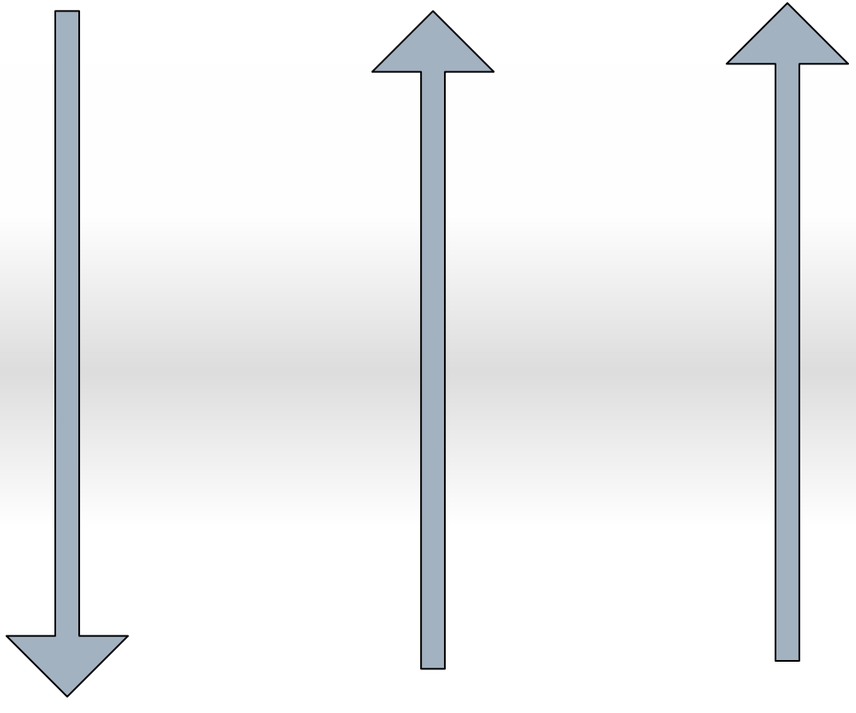


Memory Hierarchy

Increasing **Size**

Increasing **Speed**

Increasing **Cost per bit**



Comparison of Memory Hierarchy

Memory Type	Size/capacity	Access Latency	Cost per GB
Processor registers	100s of Bytes	< 1 ns	Very high
L1 cache	10s of KB	few ns	1000s
L2 cache	Few MB	10-50 ns	100s
Main (primary)	100s of MB/few GB	about 100 ns	10s
Secondary and others	100s of GB/Few TB	Few ms	< 1

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 (*Single In-line *Memory* Module*). Much larger, but much slower than cache memory.*
- *Next level is magnetic disks. Huge amount of in expensive 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.*

Question

The reason for the implementation of the cache memory is _____
a) To increase the internal memory of the system
b) The difference in speeds of operation of the processor and memory
c) To reduce the memory access and cycle time
d) All of the mentioned

Answer: b
Explanation: This difference in the speeds of operation of the system caused it to be inefficient.

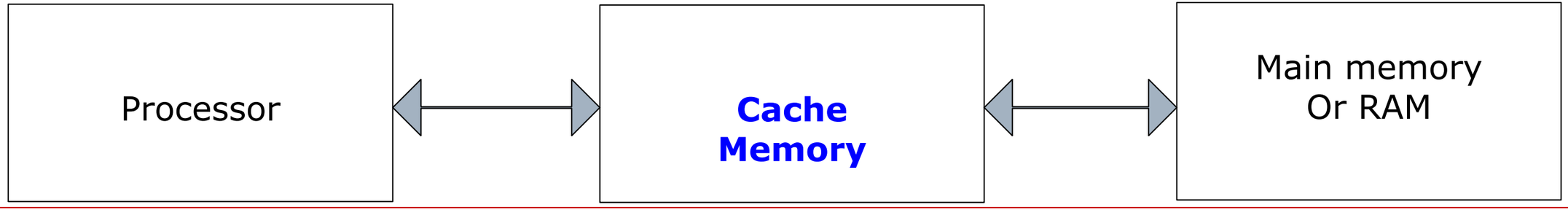
Definition of Cache Hit and Cache Miss

What is Cache Hit?

A **cache hit** is a state in which the data **requested by a processor** is **found** in the cache memory. It is a faster means of delivering data to the processor, as the cache already contains the requested data.

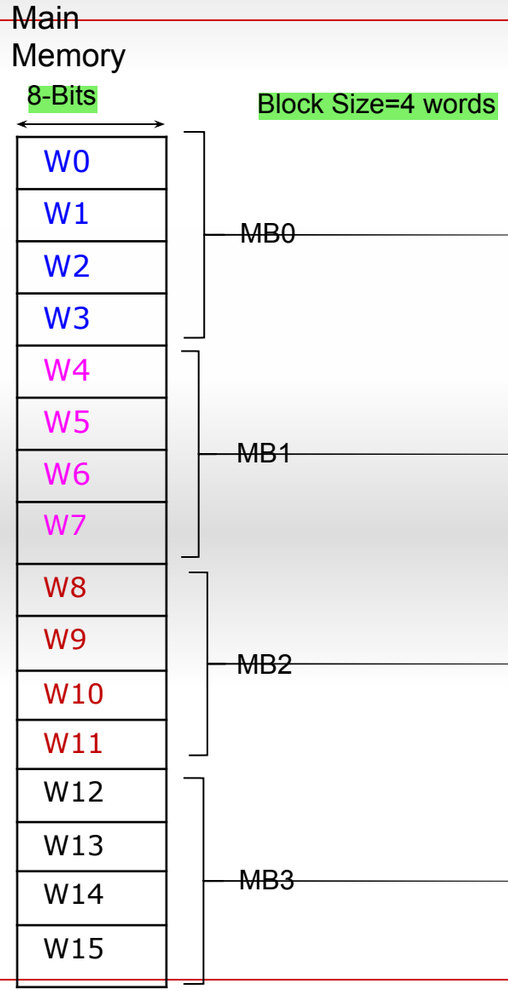
What is Cache Miss ?

Cache miss is a state where the data **requested by a processor** is **not found** in the cache memory. It causes execution delays by requiring the program or application to fetch the data from other cache levels or the main memory.

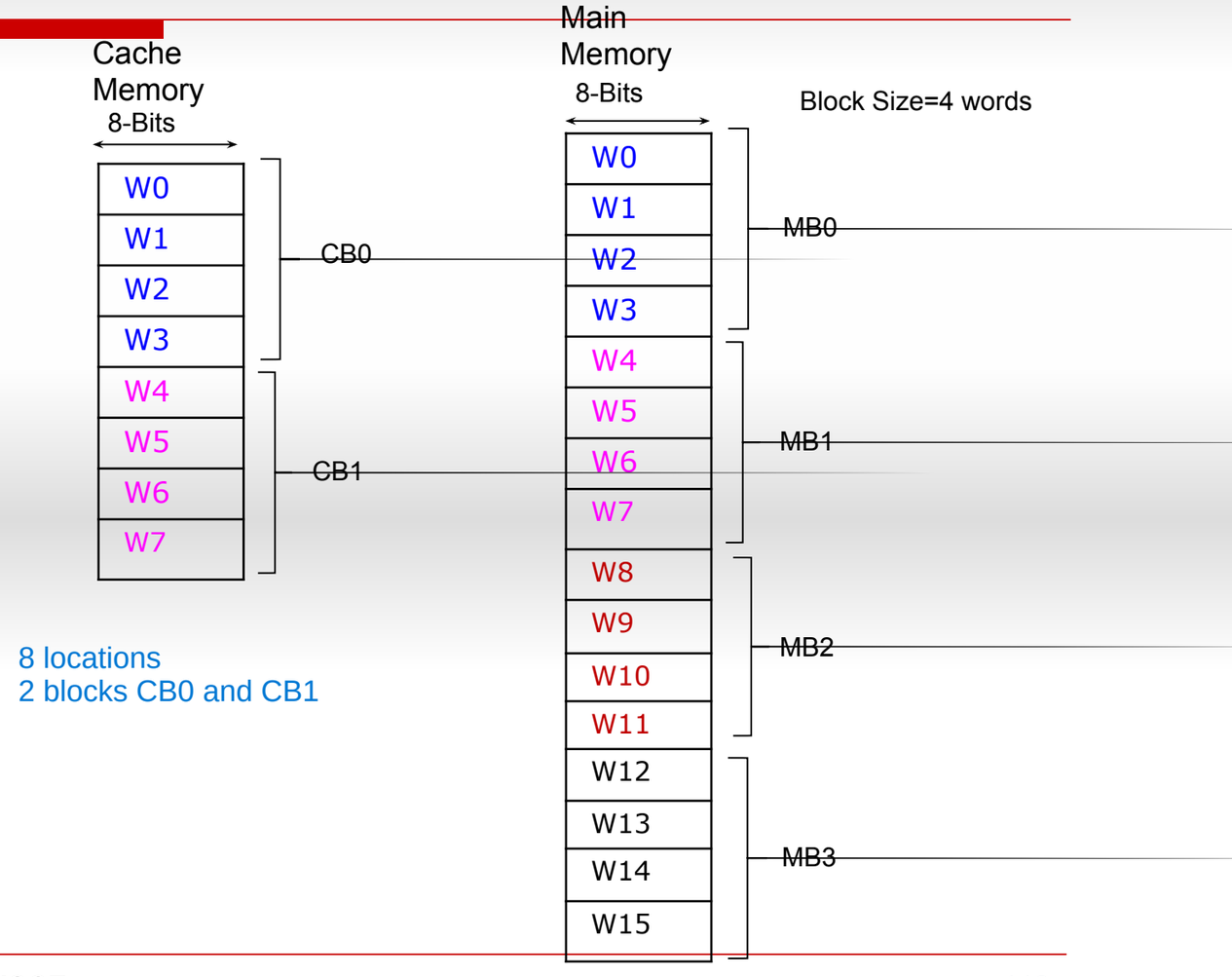


Example

16 locations
divided into blocks
each block == 4 words

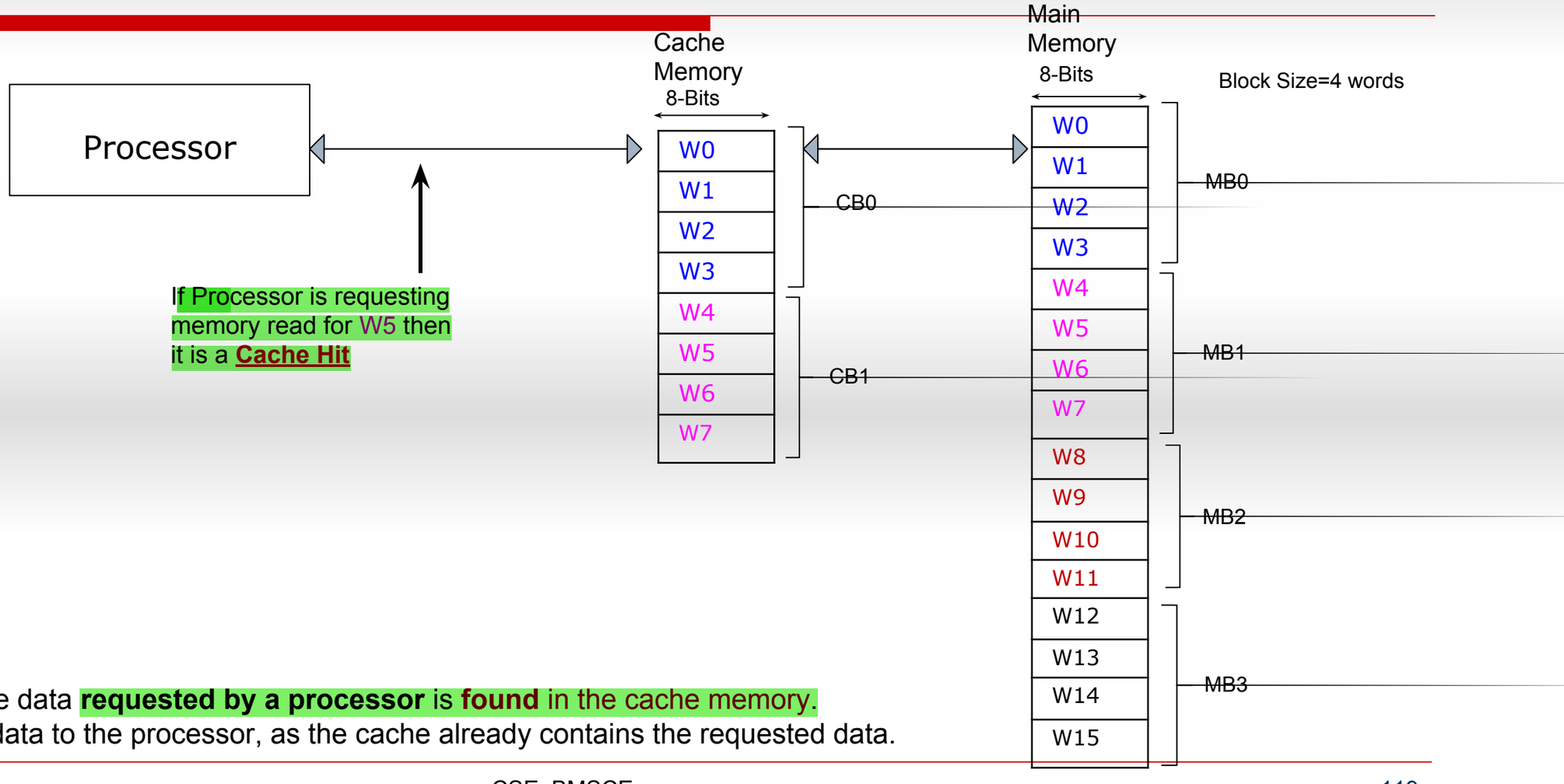


Example



8 locations
2 blocks CB0 and CB1

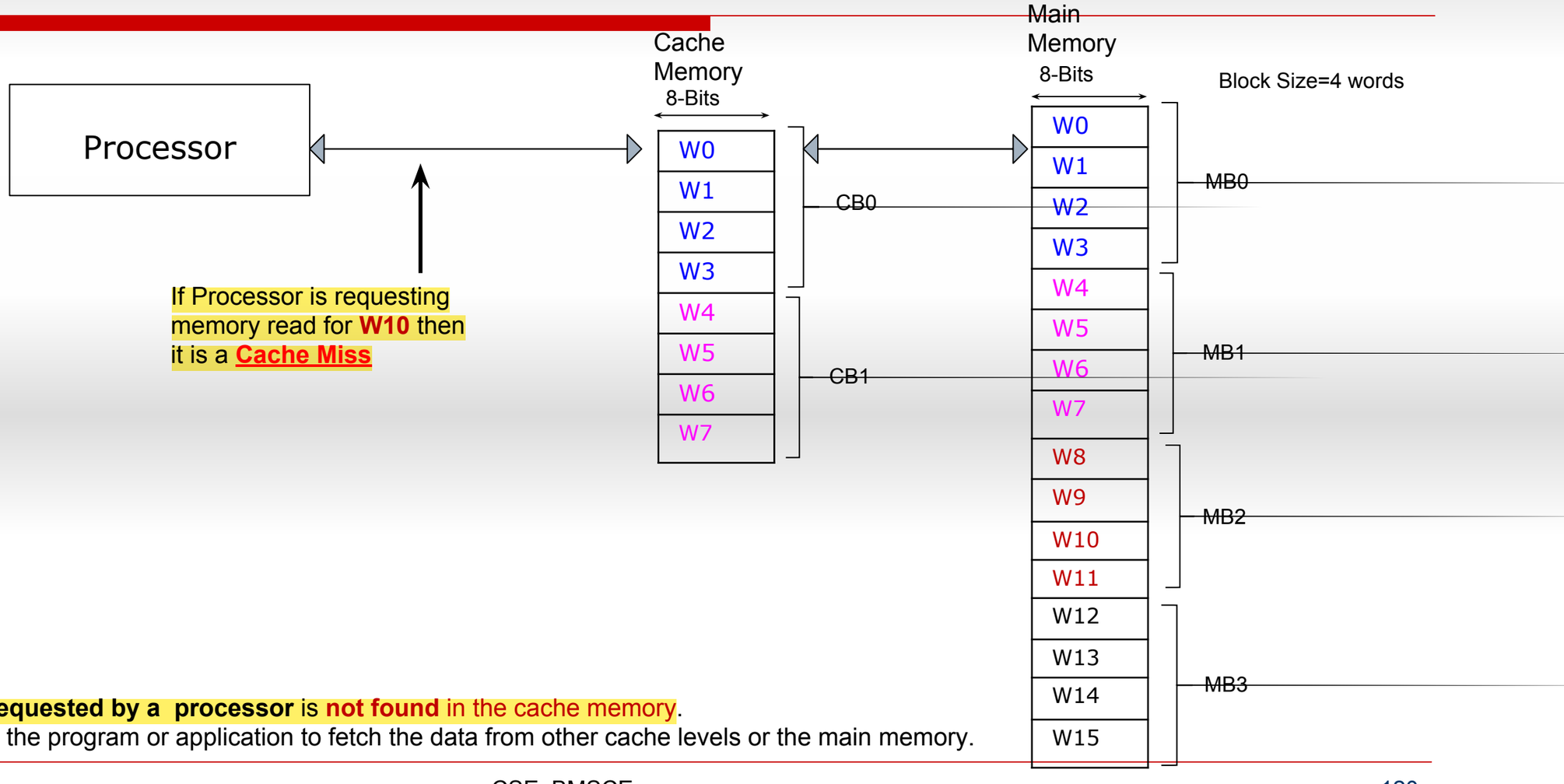
Example



What is Cache Hit?

A **cache hit** is a state in which the data **requested by a processor** is **found** in the cache memory.
It is a faster means of delivering data to the processor, as the cache already contains the requested data.

Example

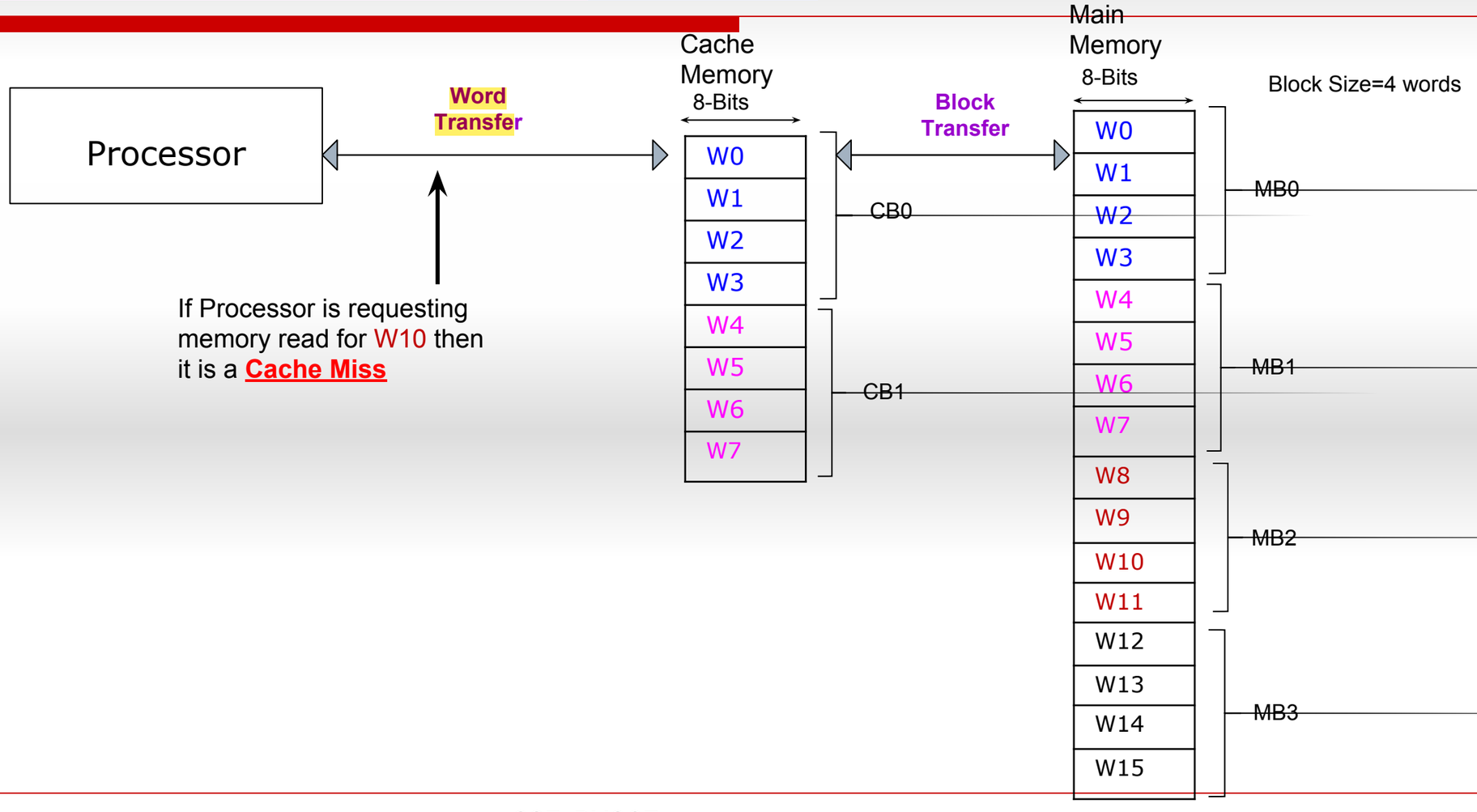


What is Cache Miss ?

Cache miss is a state where the data **requested by a processor** is **not found** in the cache memory.

It causes execution delays by requiring the program or application to fetch the data from other cache levels or the main memory.

How to do Cache replacement happens in case of Cache Miss ?



Cache Replacement or Cache Mapping functions

What happens when we have to replace something from cache to **make room for new data** because **we have a cache miss.**

Question is "Which Blocks do we **Kick Out** in Cache ?"

Cache Replacement or Cache Mapping functions or Cache Mapping Techniques

- Direct mapping
- Associative mapping
- Set-associative mapping

Cache Mapping functions

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

Cache Mapping function: Direct Mapping

The simplest way to determine cache locations in which to store memory blocks is the *direct-mapping* technique.

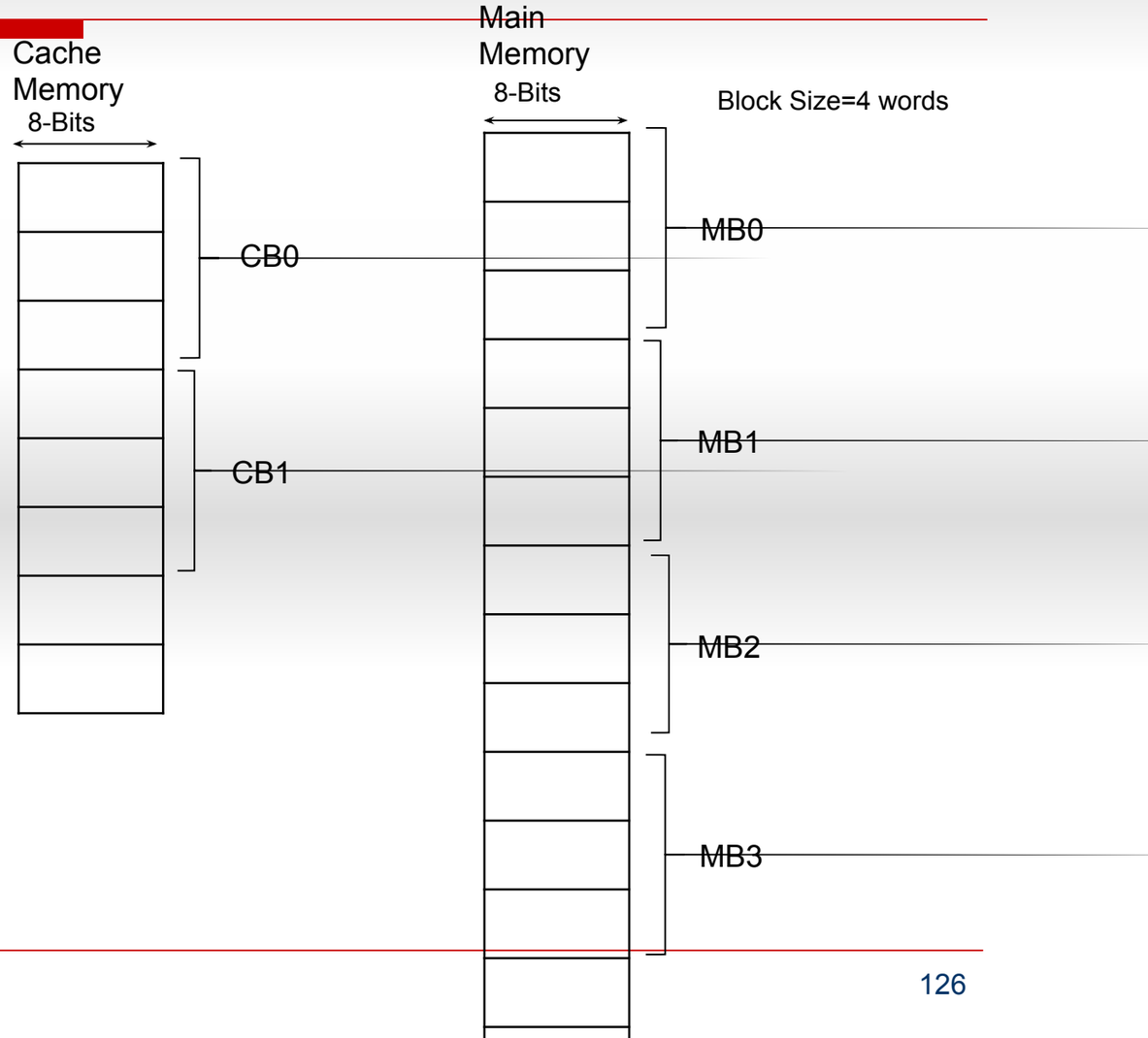
In this technique, block *j* of the main memory maps onto block "*j* modulo (Total number of Blocks in cache)" of the cache.

REMAINDER

Rough slide to explain: Direct Mapping

Direct Mapping Example:

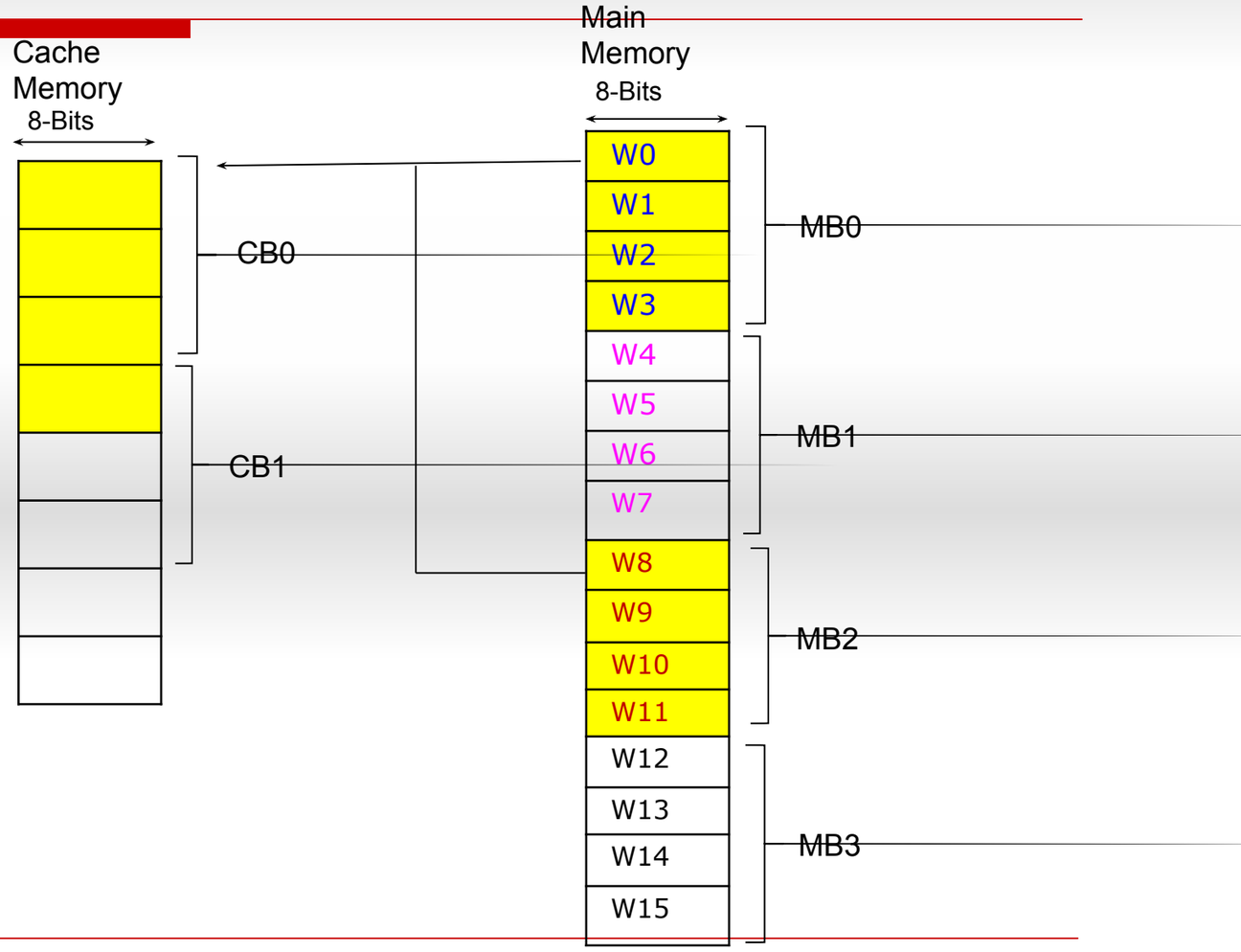
Main Memory Block Number	Direct Mapping	Cache Memory Block Number
j		j % 2



Rough slide to explain: Direct Mapping

Direct Mapping Example:

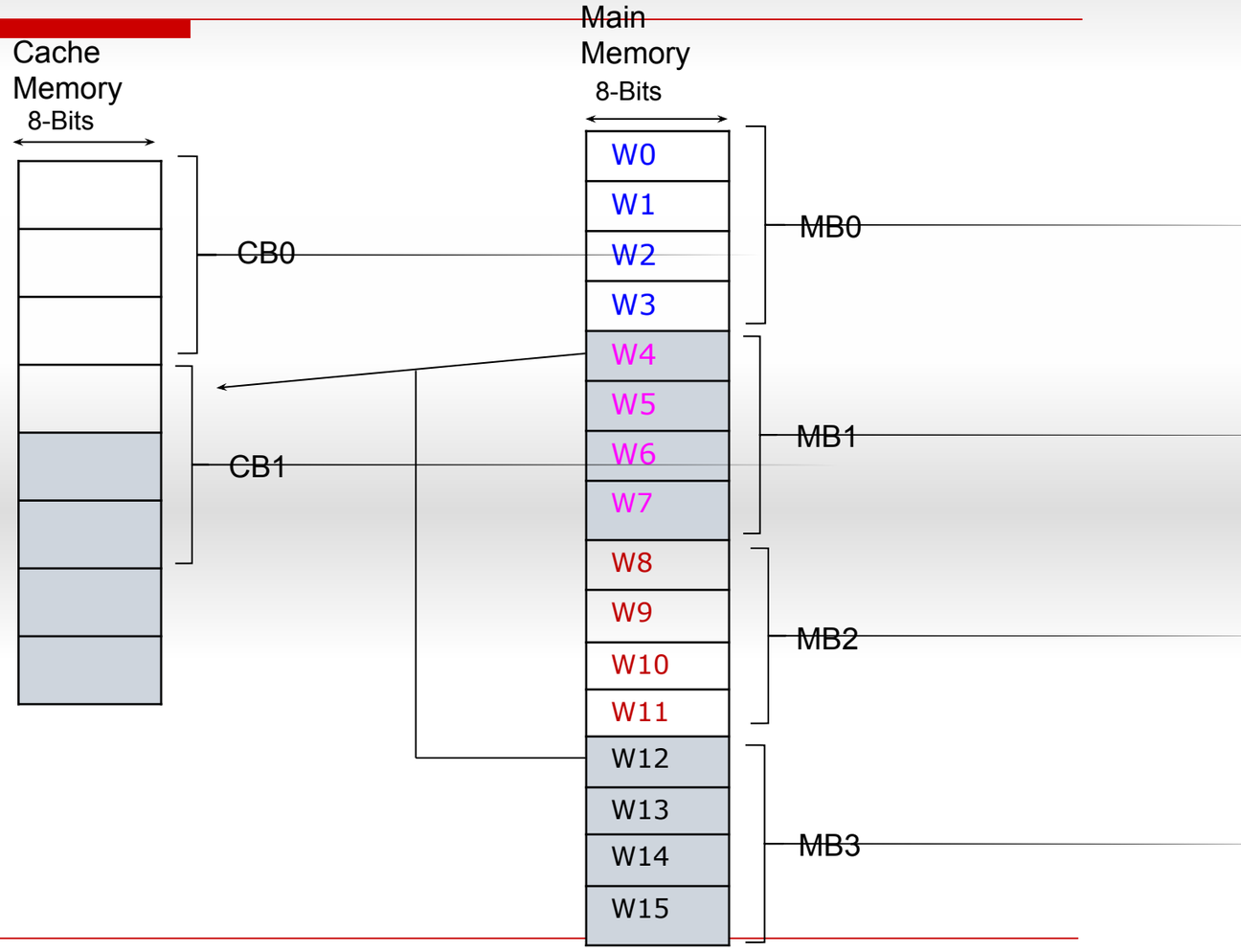
Main Memory Block Number	Direct Mapping	Cache Memory Block Number
j		j % 2



Rough slide to explain: Direct Mapping

Direct Mapping Example:

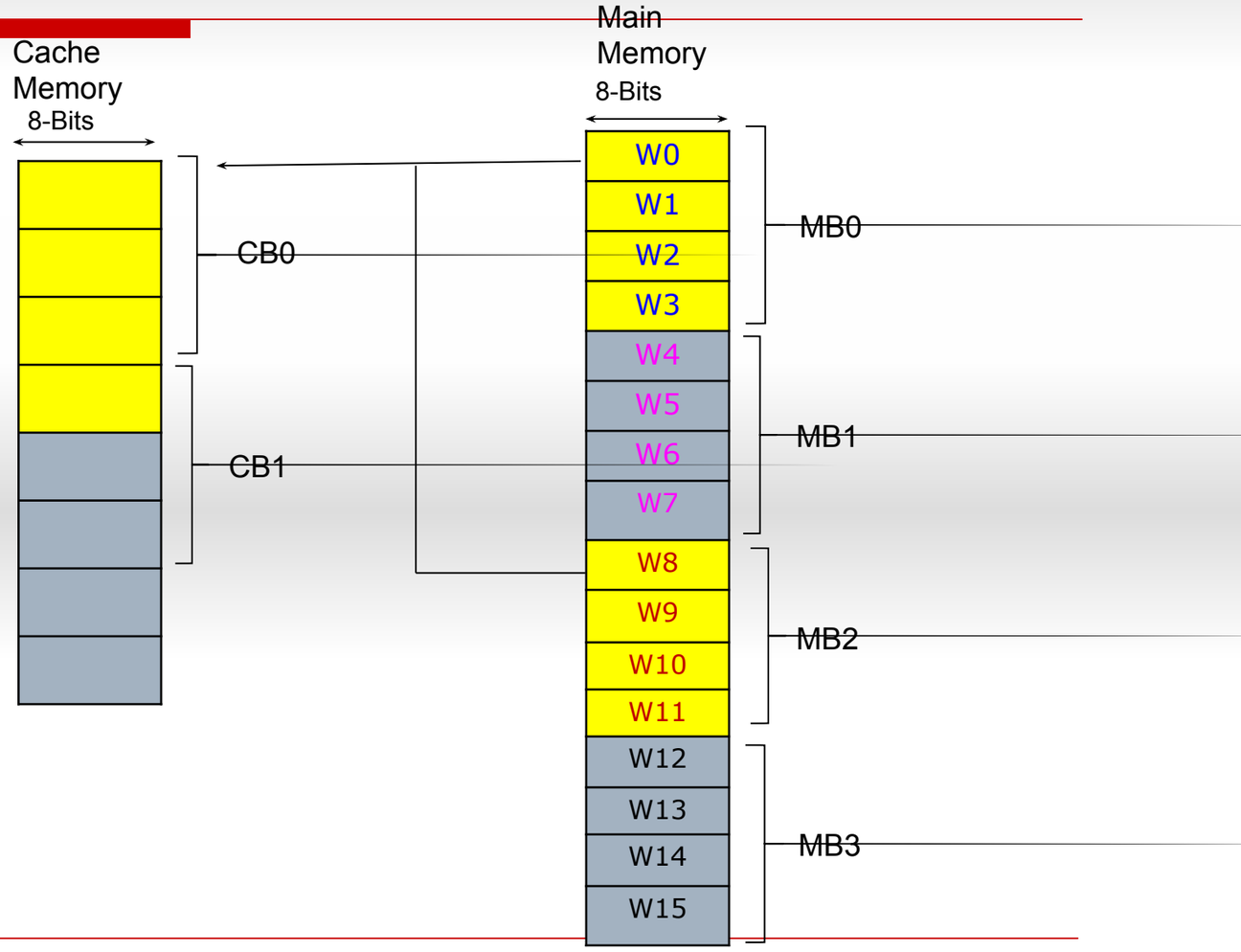
Main Memory Block Number	Direct Mapping	Cache Memory Block Number
j		j % 2



Rough slide to explain: Direct Mapping

Direct Mapping Example:

Main Memory Block Number	Direct Mapping	Cache Memory Block Number
j		j % 2



Rough slide to explain: Direct Mapping

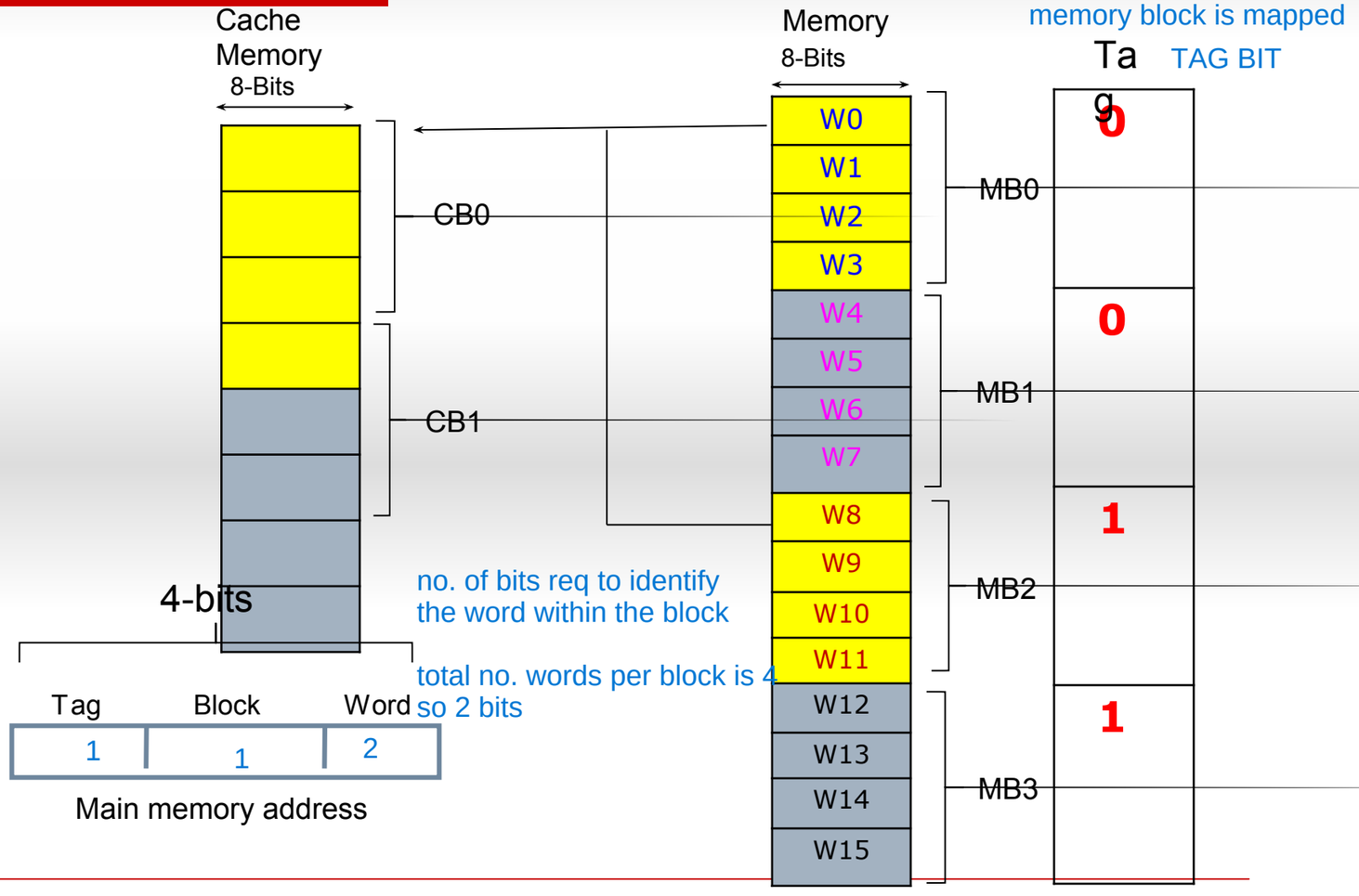
j

j % 2

cache block number

tag bit

0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1



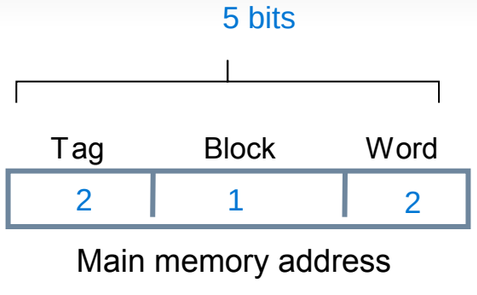
Rough slide to explain: Direct Mapping

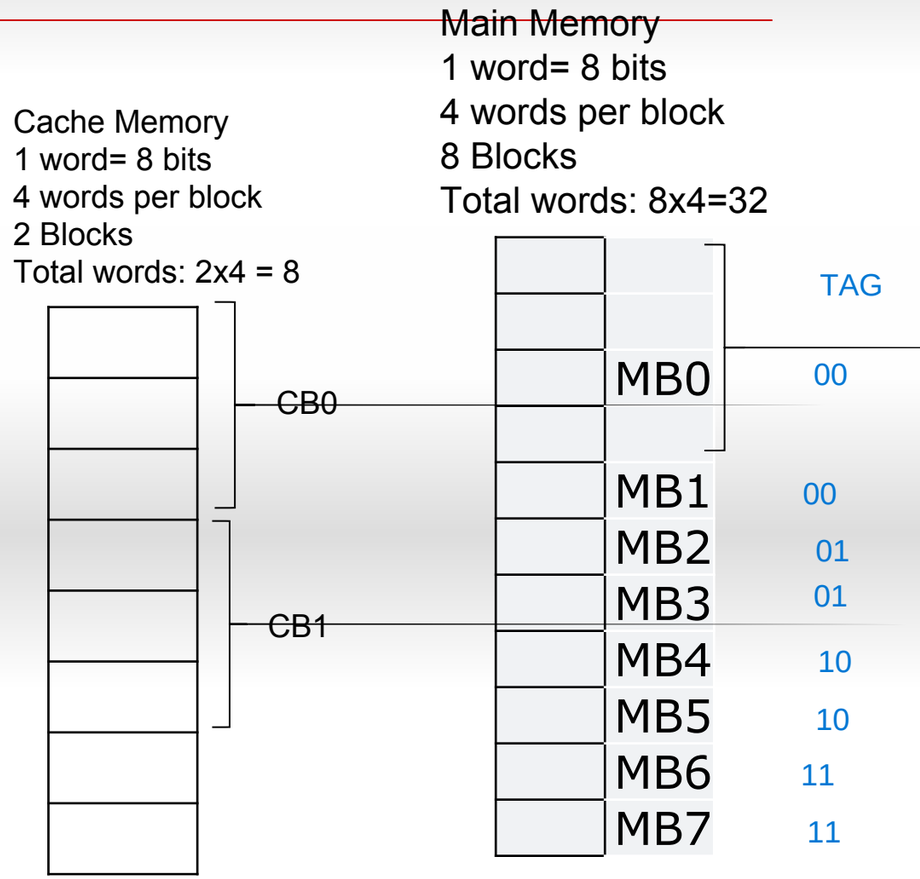
Main Memory Block Number	Direct Mapping	Cache Memory Block Number
j		j % 2

Question,

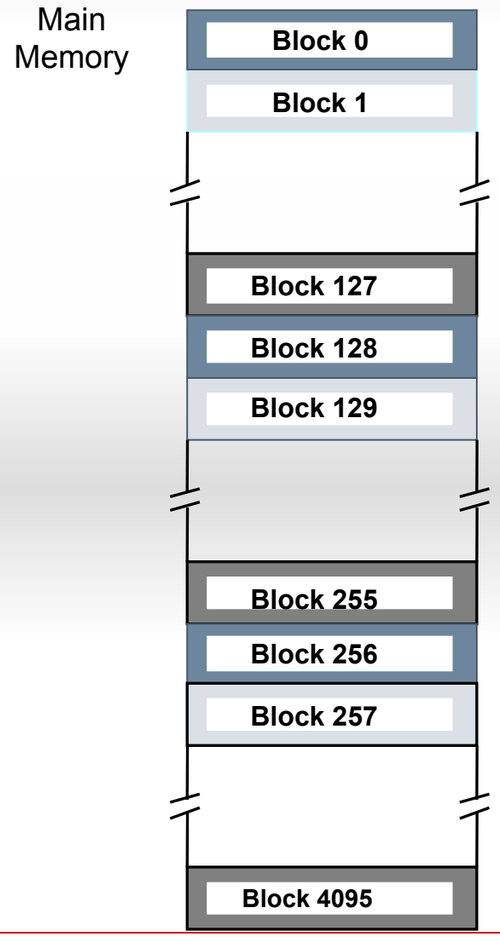
1. Memory block **MB3** maps to which block of Cache block ? 3%2 == 1
2. How many Main memory blocks will contend for each Cache memory block ? 4
3. How bits are required for Tag, Block and Word ?

32 locations == 2 power 5
so total 5 bits are required



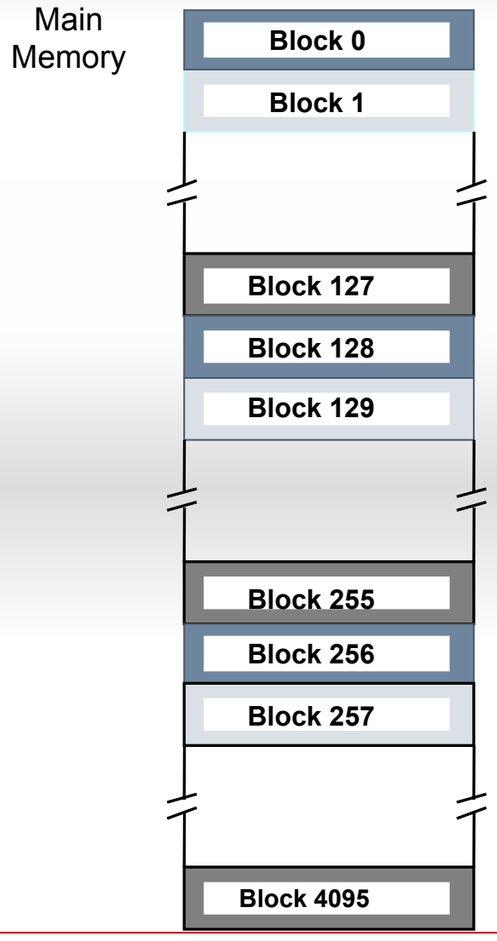


Example to Understand Mapping Functions



Consider, Block Size = 16 Words

Example to Understand Mapping Functions



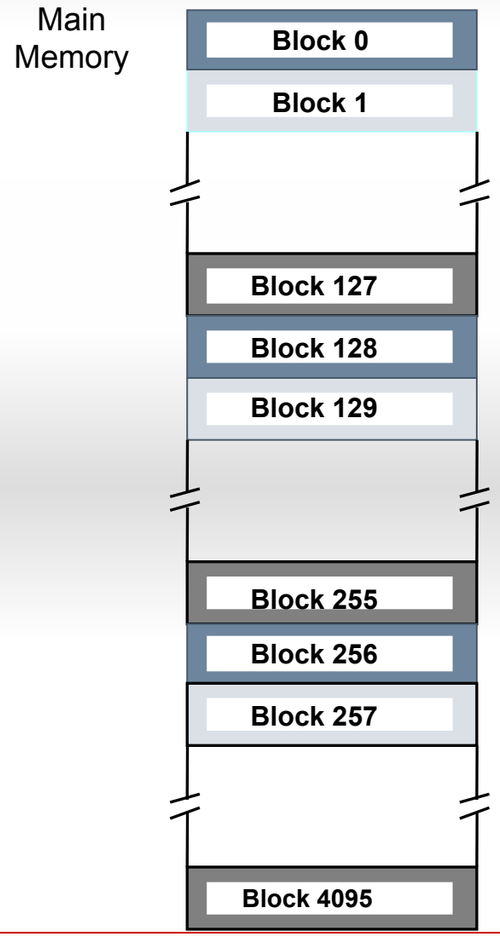
Consider, Block Size = 16 Words

Main Memory

Number of Blocks in Main Memory=4K=**4096**

Total number of words in Main Memory= 4096x16
=65536
=64 x 1024
=64K words

Example to Understand Mapping Functions



Consider, Block Size = 16 Words

Main Memory

Number of Blocks in Main Memory=4K=4096

Hence number of words in Main Memory= 4096x16

=65536

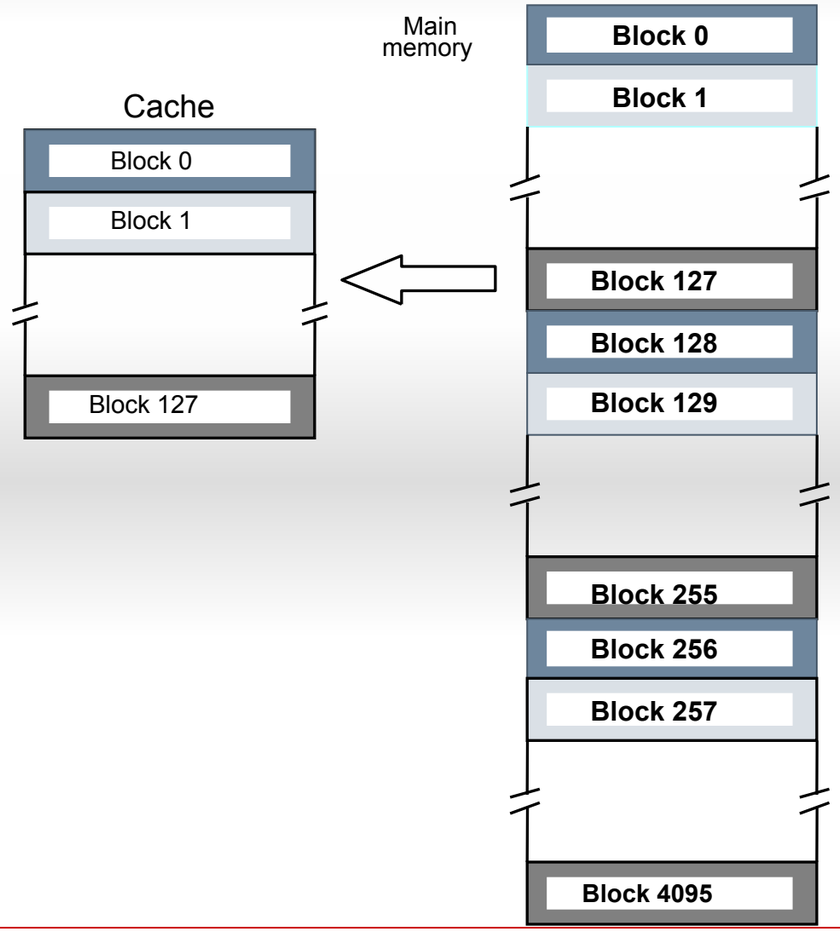
=64x1024

=64K words

Main memory is addressable by a **16-bit** address,

Because 64K = $2^6 x 2^{10} = 2^{16}$

Example to Understand Mapping Functions



Consider, Block Size = 16 Words

Main Memory

Number of Blocks in Main Memory=4K=4096

Hence number of words in Main Memory= 4096x16
=65536
=64x1024
=64K words

Main memory is addressable by a **16-bit** address,

Because 64K = $2^6 x 2^{10} = 2^{16}$

Cache Memory

Cache consisting of **128** blocks of 16 words each.

Total size of cache is =128x16
=2048 (2K) words.

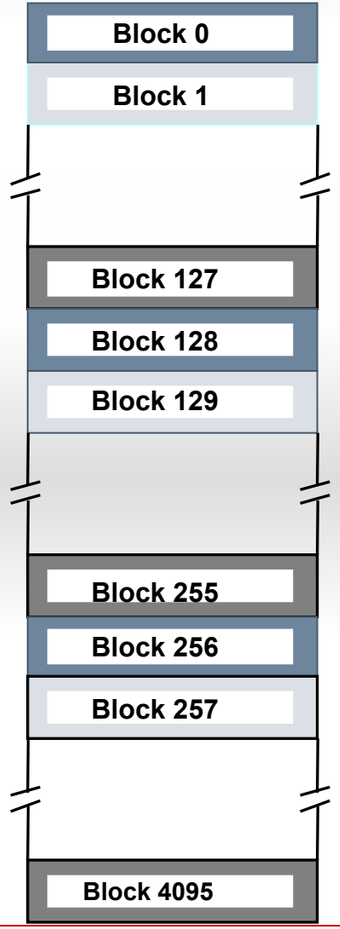
Cache memory is addressable by a **11-bit** address,

Because 2K = $2^1 x 2^{10} = 2^{11}$

Example to Understand Mapping Functions

word -- 4 bits
block--7 bits
tag --5 bits

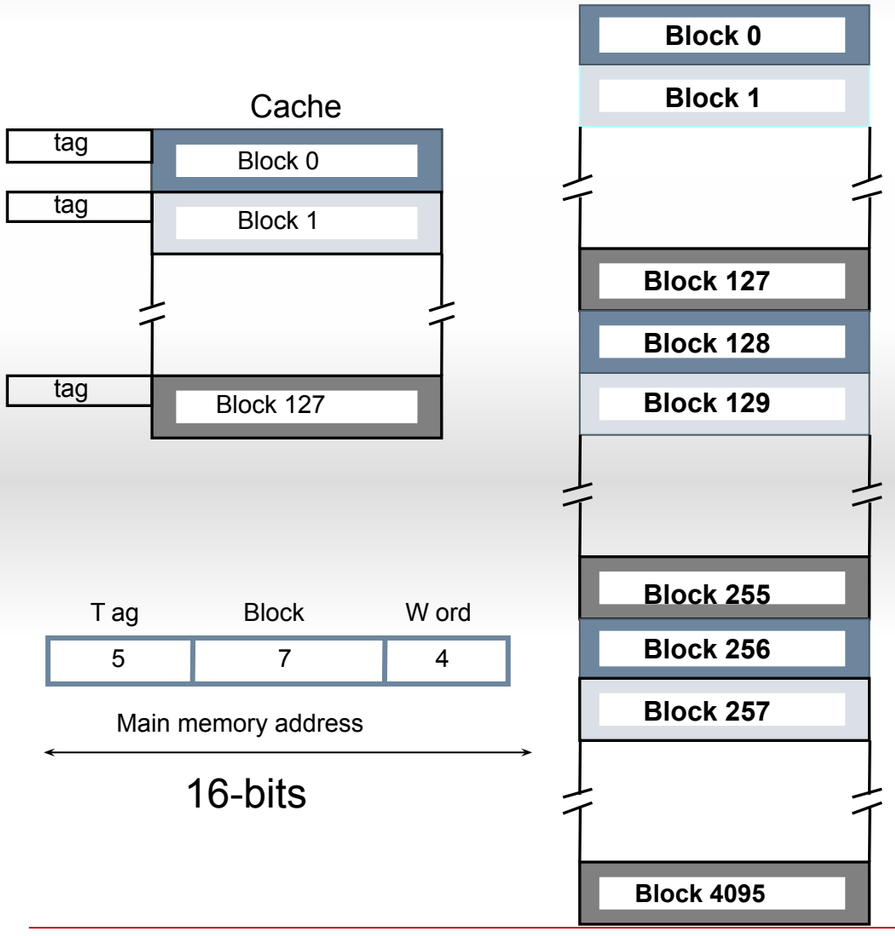
32 blocks of the main memory will contend per cache block



Main Memory

Block 0	Block 128	Block 256	Block 384	…….	…….
Block 1	Block 129	Block 257	Block 385	…….	…….
Block 2	Block 130	Block 258	Block 386	…….	…….
…….	…….	…….	…….	…….	…….
Block 127	Block 255	Block 383	Block 511	…….	Block 4095
0	**1**	**2**	**3**		**31**

Direct Mapping

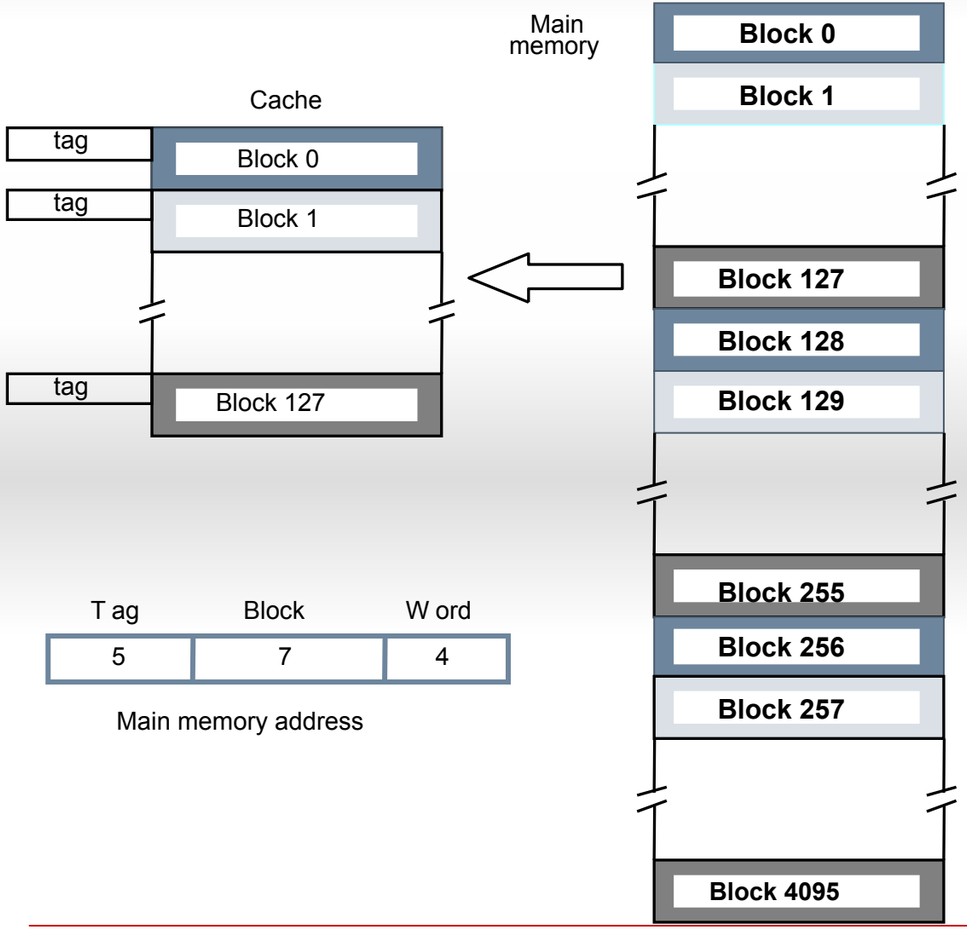


Main Memory (16-Address bits)

Block 0	Block 128	Block 256	Block 384	.....	.....
Block 1	Block 129	Block 257	Block 385	.....	.....
Block 2	Block 130	Block 258	Block 386	.....	.....
.....	.....	.....	.....	.....	.....
Block 127	Block 255	Block 383	Block 511	.....	Block 4095
0	**1**	**2**	**3**		**31**

Main Memory
-16-Address bits
- Block Size = 16 words

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.*

Direct Mapping

*In direct mapping, Block **j** of the main memory maps to **j modulo 128** of the cache.*
0 maps to 0, 129 maps to 1……

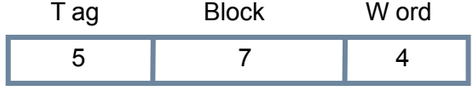
Tag
3
1
0
31

Cache Memory

0	Block 384
1	Block 129
2	Block 2
…	…….
127	Block 4095

Main Memory

Block 0	Block 128	Block 256	Block 384	…….	…….
Block 1	Block 129	Block 257	Block 385	…….	…….
Block 2	Block 130	Block 258	Block 386	…….	…….
…….	…….	…….	…….	…….	…….
Block 127	Block 255	Block 383	Block 511	…….	Block 4095



Main memory address

Main Memory
-16-Address bits
- Block Size = 16 words

Direct Mapping

*In direct mapping, Block **j** of the main memory maps to **j modulo 128** of the cache.*
0 maps to 0, 129 maps to 1……

Tag
3
1
0
…….
31

Cache Memory

0	Block 384
1	Block 129
2	Block 2
…	…….
127	Block 4095

Main Memory

Block 0	Block 128	Block 256	Block 384	…….	…….
Block 1	Block 129	Block 257	Block 385	…….	…….
Block 2	Block 130	Block 258	Block 386	…….	…….
…….	…….	…….	…….	…….	…….
Block 127	Block 255	Block 383	Block 511	…….	Block 4095
0	1	2	3		31

Tag	Block	Word
5	7	4

Main memory address

Main Memory
-16-Address bits
- Block Size = 16 words

Question

A cache is organized in the direct-mapped manner with the following parameters:

Main memory size 64K words
Cache size 1K words
Block size 128 words

i. How many bits are there in a main memory address ?

ii. How many bits are there in each of the TAG, BLOCK and WORD fields ?

Question

A cache is organized in the direct-mapped manner with the following parameters:

Main memory size 64K words
Cache size 1K words
Block size 128 words

i. How many bits are there in a main memory address ?

ii. How many bits are there in each of the TAG, BLOCK and WORD fields ?

Answer

Main memory address length is 16 bits.

TAG field is 6 bits. BLOCK field is 3 bits (8 blocks). WORD field is 7 bits (128 words per block).

Question

A computer memory system has a 64KB byte-addressable main memory with 16-bit addresses. This same system has a one-level cache memory that can hold 16 blocks of data, where each block contains 16 bytes. Assuming this is a direct-mapped cache, to which cache block will main memory address $9A81_{16}$ map?

(A) 0_{16} (B) 1_{16} (C) 8_{16} (D) A_{16}

Question

A computer memory system has a 64KB byte-addressable main memory with 16-bit addresses. This same system has a one-level cache memory that can hold 16 blocks of data, where each block contains 16 bytes. Assuming this is a direct-mapped cache, to which cache block will main memory address $9A81_{16}$ map?

(A) 0_{16} (B) 1_{16} (C) 8_{16} (D) A_{16}

Given, memory address is of 16-bits.

Each cache block contains 16 bytes(i.e. = 2^4 bytes), so the WORD offset field will be of 4 bits (since memory is byte addressable).

Then we have total 16 blocks (i.e = 2^4 blocks) in the cache. So BLOCK field will be of 4 bytes.

Then the remaining 16 - 4 - 4 = 8 bits are for TAG field.

TAG(8 bits)	BLOCK(4 bits)	OFFSET(4 bits)

Memory address given: $(9A81)_{16}$

$1001 - 1010 - 1000 - 0001$

8th block in the cache

So, it maps to block $(8)_{16}$

Question

A computer system uses 4-bit memory addresses. It has a 8-byte cache organized in a direct-mapped manner with 4 bytes per cache block. Assume that the size of each memory word is 1 byte.

(a) Calculate the number of bits in each of the Tag, Block, and Word fields of the memory address.

(b) When a program is executed, the processor reads data sequentially from the following word addresses:

5, 7, 12, 4

All the above addresses are shown in decimal values. Assume that the cache is initially empty. For each of the above addresses, indicate whether the cache access will result in a **hit** or a **miss**.

no. of bits in the main memory is 4.
no. of words per block == 4 == 2 power 2 .....hence the word bit requires 2 bits

no of blocks of cache == 8 byte / 4 byte === 2 one bit for BLOCK

tag == main memory -(block + word) === 4-(2+1) === 1

Note:

Cache hit is a state in which the data **requested by a processor** is **found** in the cache memory.

Cache miss is a state where the data **requested by a processor** is **not found** in the cache memory.

Question

Answer

(a) Block size = 4 bytes = 2^2 bytes = 2^2 words (since 1 word = 1 byte)

Therefore, **Number of bits in the *Word* field = 2**

Cache size = 8-byte = 2^3 bytes

Number of cache blocks = Cache size / Block size = $2^3/2^2 = 2^1$

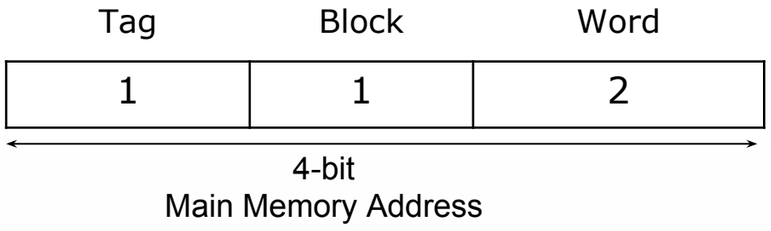
Therefore, **Number of bits in the *Block* field = 1**

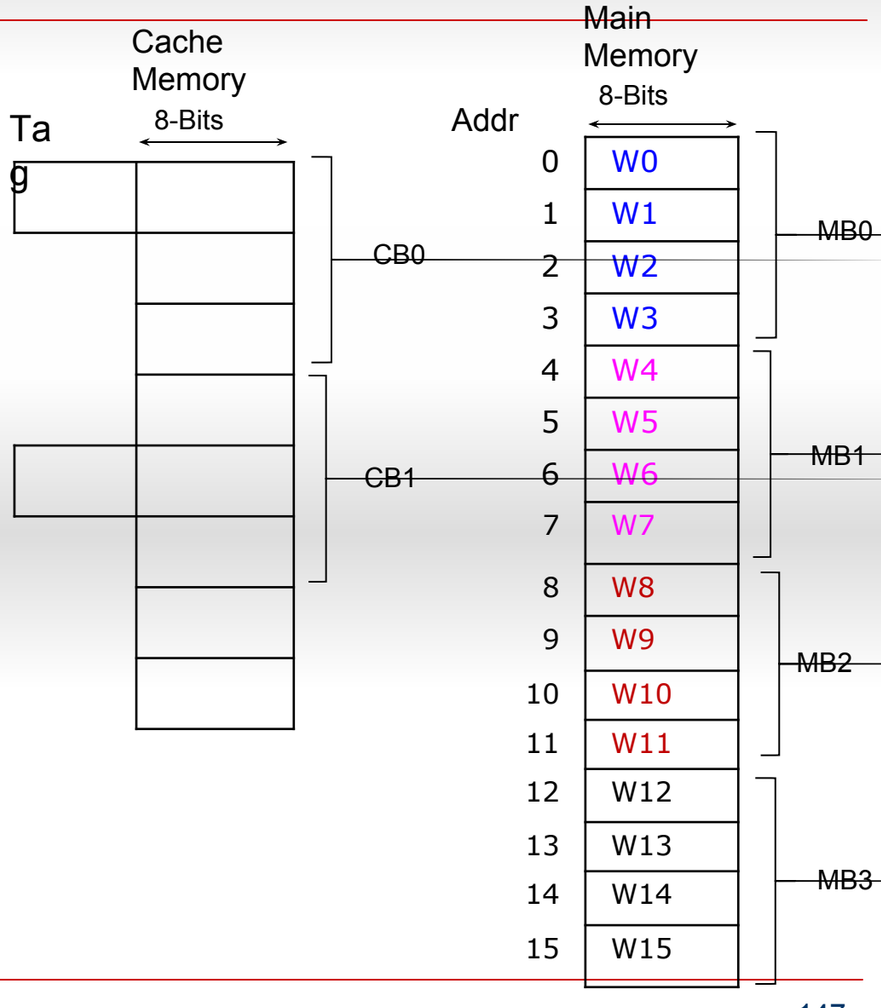
Total number of address bits of main memory = 4

Therefore, **Number of bits in the *Tag* field = 4 - 2 - 1 = 1**

For a given 4-bit address, the 1 most significant bit, represent the *Tag*, the next 1 bit represent the *Block*, and the 2 least significant bits represent the *Word*.

Answer (Contd…)





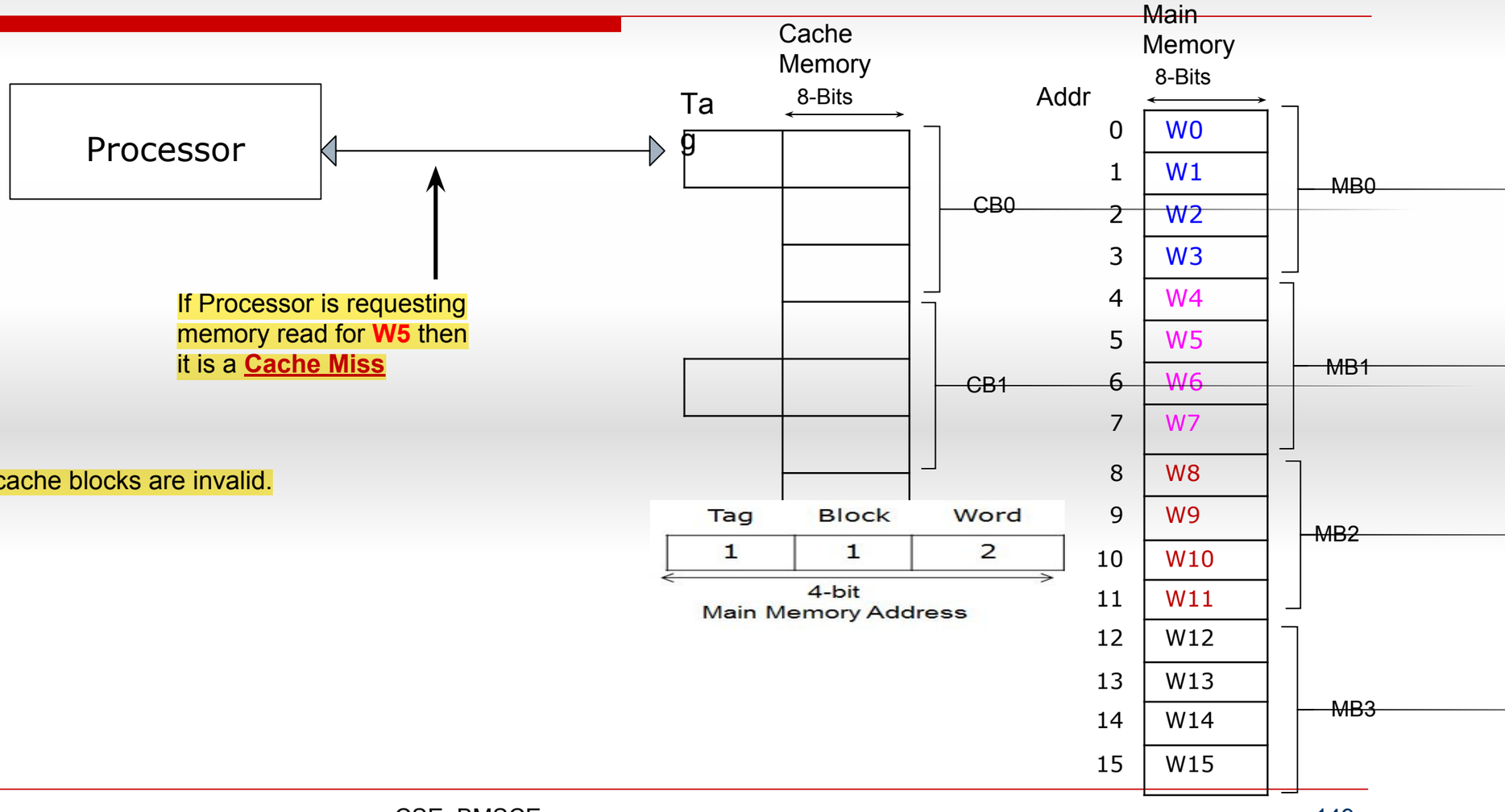
Question

(b) When a program is executed, the processor reads data sequentially from the following word addresses:

5, 7, 12, 4

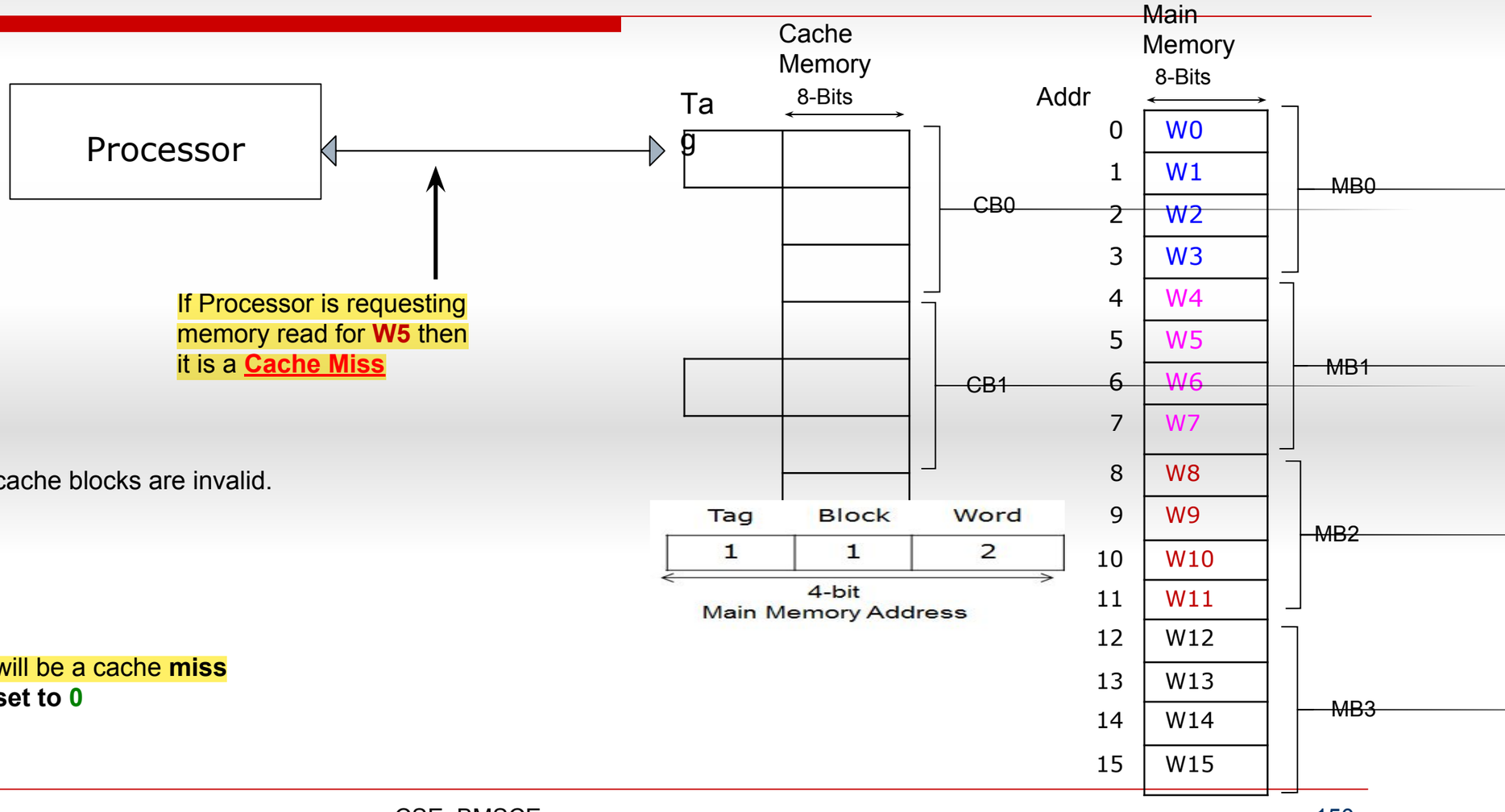
All the above addresses are shown in decimal values. Assume that the cache is initially empty. For each of the above addresses, indicate whether the cache access will result in a **hit** or a **miss**.

Answer (Contd…)



(b) The cache is initially empty. Therefore, all the cache blocks are invalid.

Answer (Contd…)



(b) The cache is initially empty. Therefore, all the cache blocks are invalid.

Access # 1:

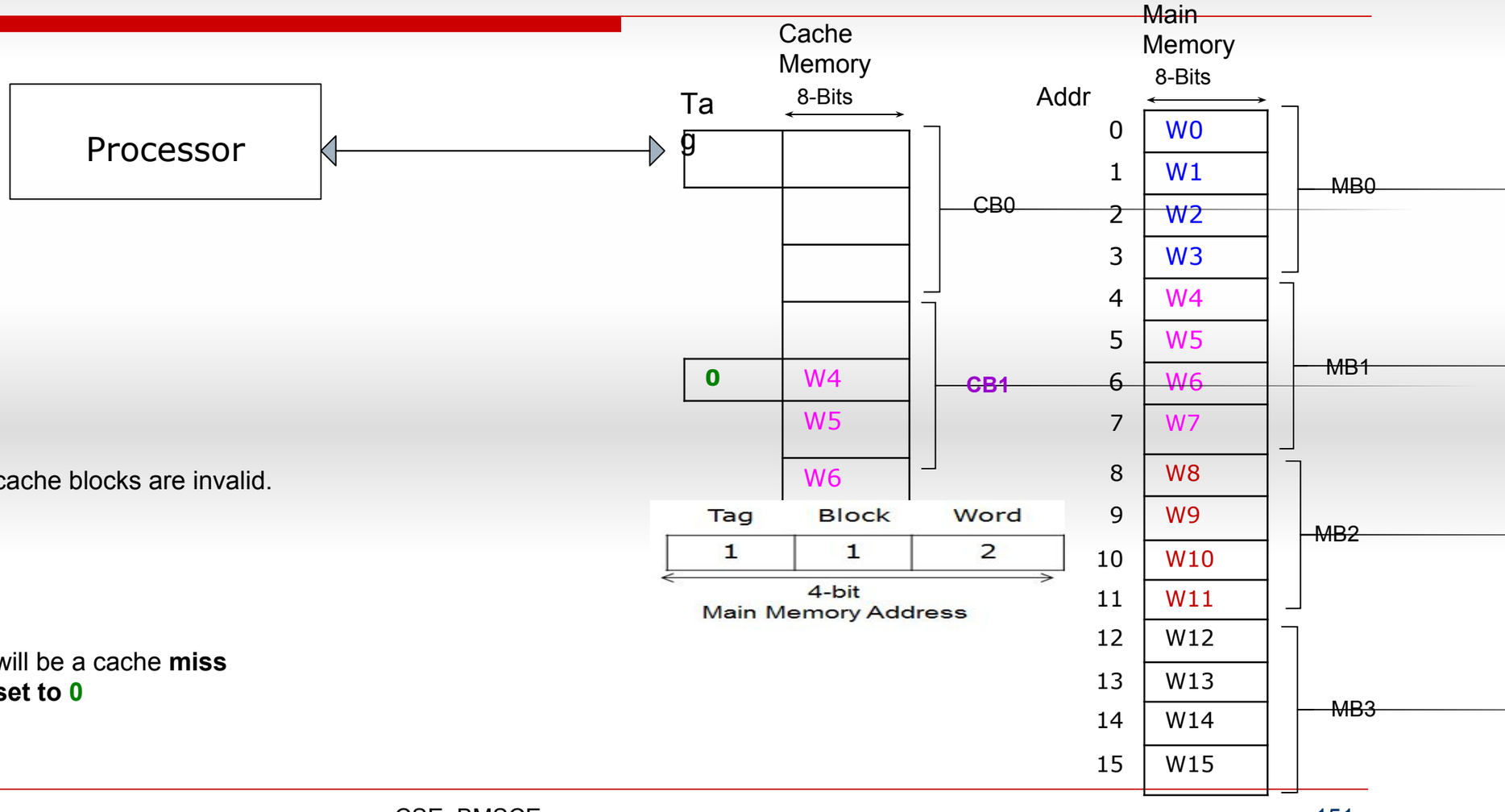
Address = $(5)_{10}$ = $(0101)_2$

For this address, *Tag* = **0**, *Block* = **1**, *Word* = 01

Since the cache is empty before this access, this will be a cache **miss**

After this access, **Tag field for cache block 1 is set to 0**

Answer (Contd…)



(b) The cache is initially empty. Therefore, all the cache blocks are invalid.
Access # 1:
Address = $(5)_{10}$ = $(0101)_2$

For this address, *Tag* = **0**, *Block* = **1**, *Word* = 01
Since the cache is empty before this access, this will be a cache **miss**
After this access, **Tag field for cache block 1 is set to 0**

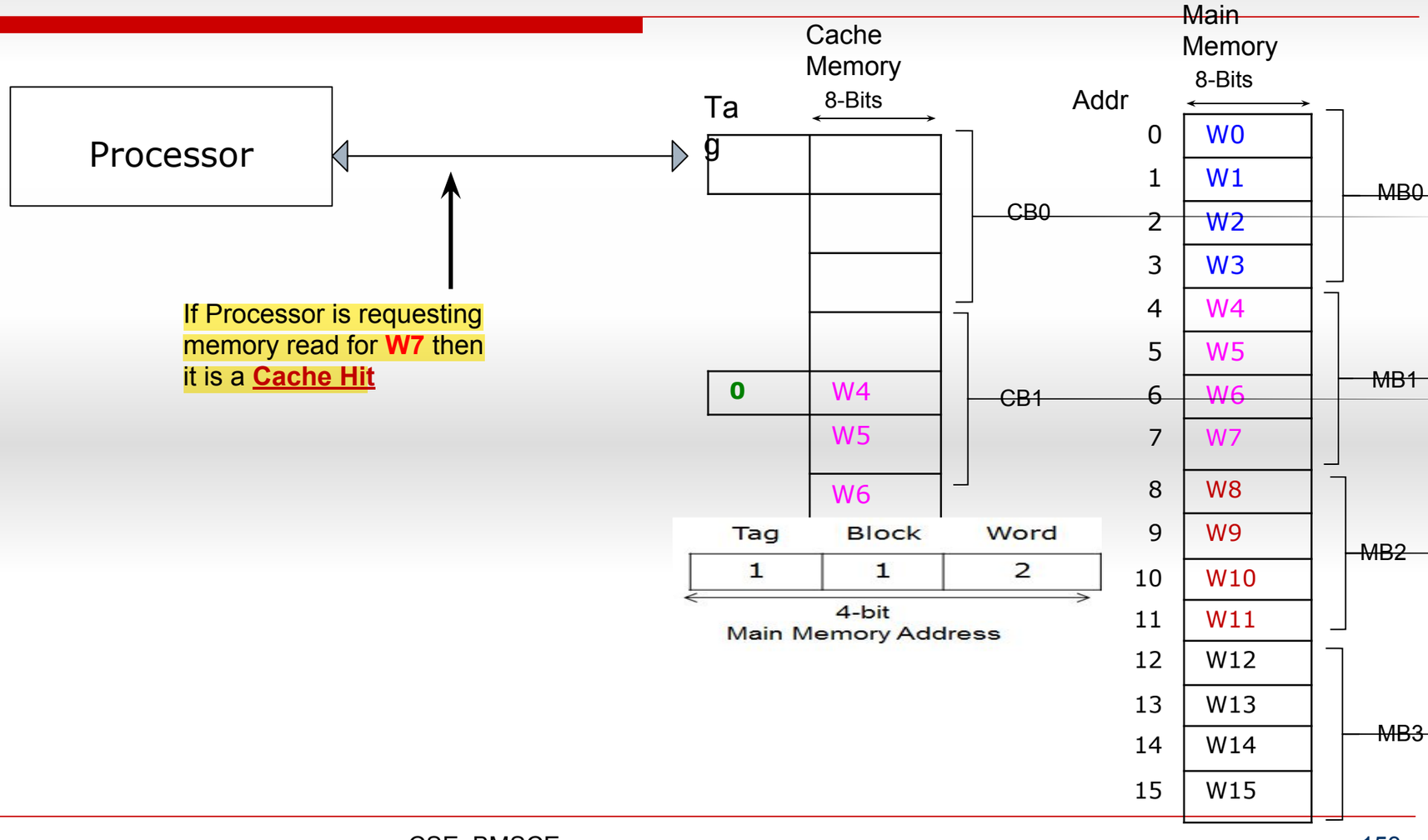
Question

(b) When a program is executed, the processor reads data sequentially from the following word addresses:

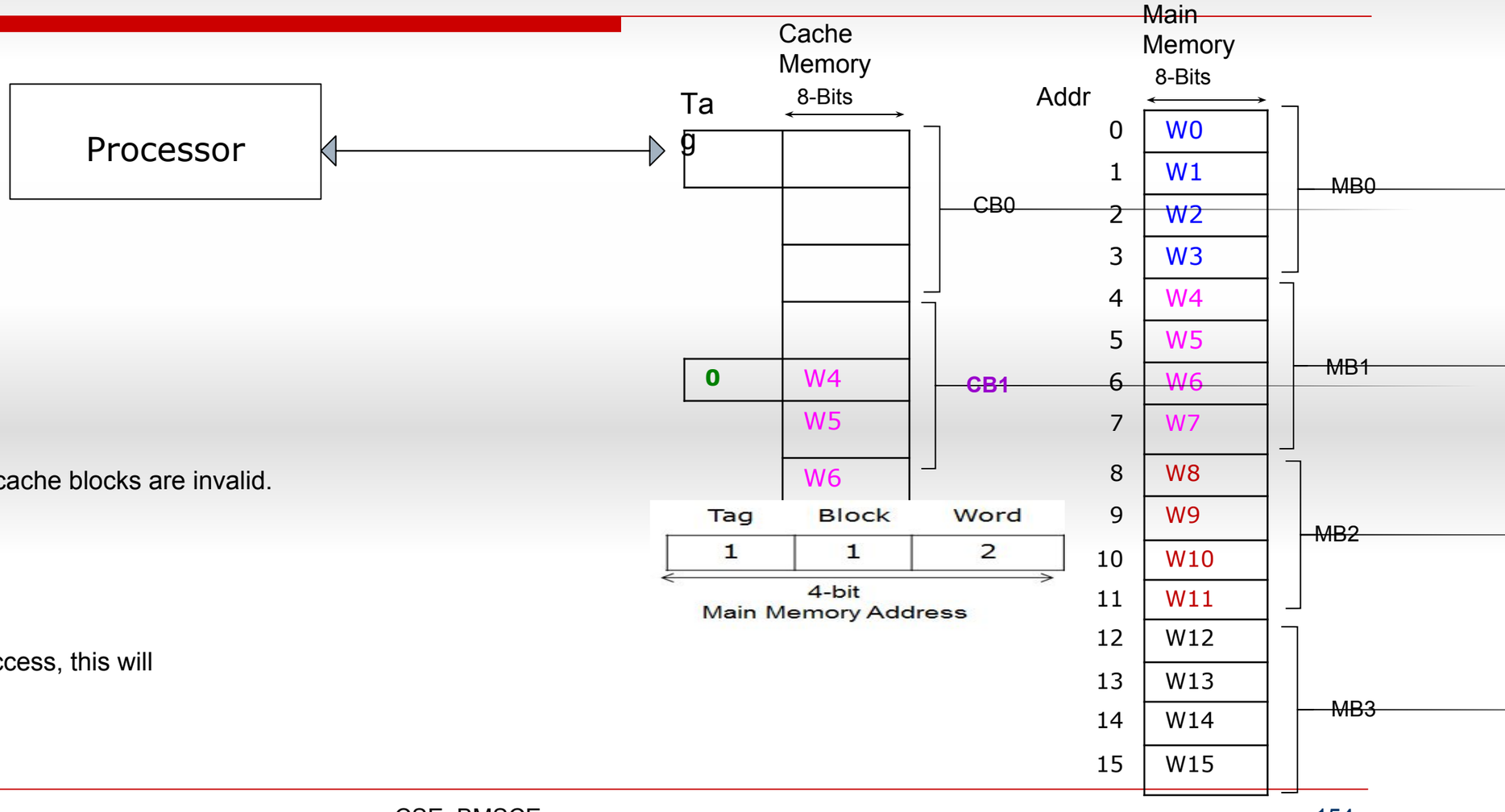
5, **7**, 12, 4

All the above addresses are shown in decimal values. Assume that the cache is initially empty. For each of the above addresses, indicate whether the cache access will result in a **hit** or a **miss**.

Answer (Contd…)



Answer (Contd…)



(b) The cache is initially empty. Therefore, all the cache blocks are invalid.

Access # 2:

Address = $(7)_{10}$ = $(0111)_2$

For this address, *Tag* = **0**, *Block* = **1**, *Word* = 11

Since tag field for cache block **1** is **0** before this access, this will be a cache **hit** (because address tag = block tag)

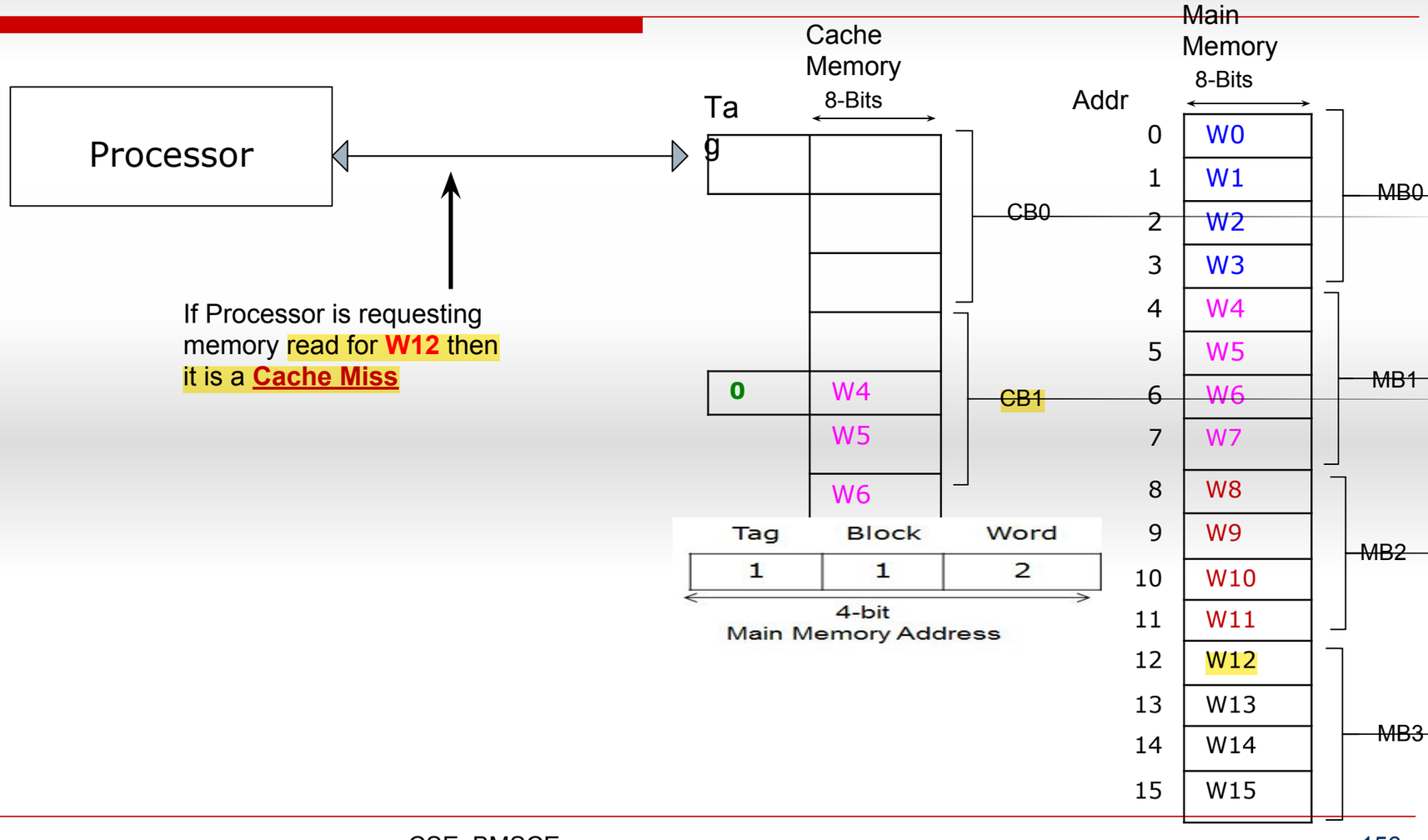
Question

(b) When a program is executed, the processor reads data sequentially from the following word addresses:

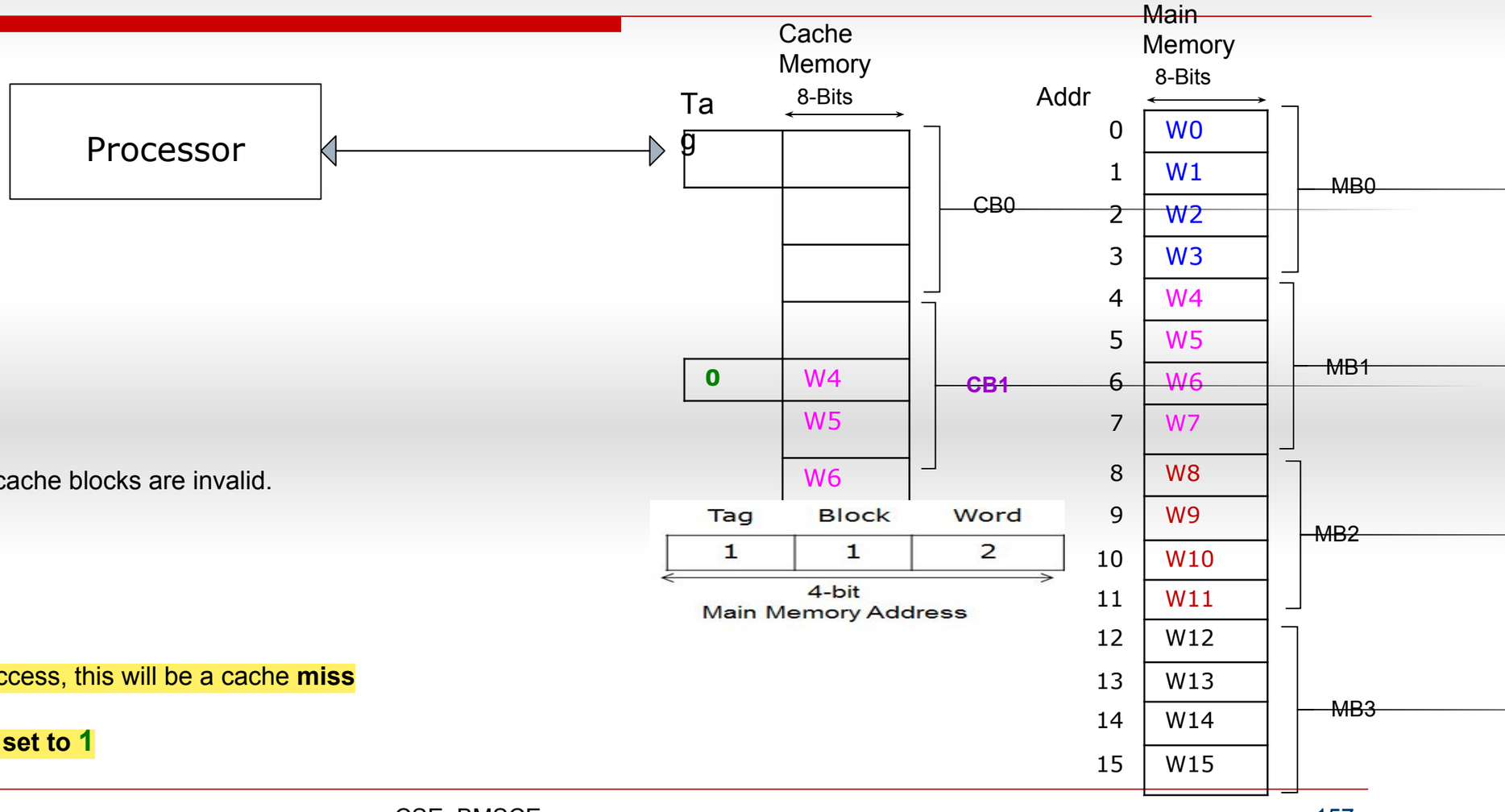
5, 7, **12**, 4

All the above addresses are shown in decimal values. Assume that the cache is initially empty. For each of the above addresses, indicate whether the cache access will result in a **hit** or a **miss**.

Answer (Contd…)



Answer (Contd…)



(b) The cache is initially empty. Therefore, all the cache blocks are invalid.

Access # 3:

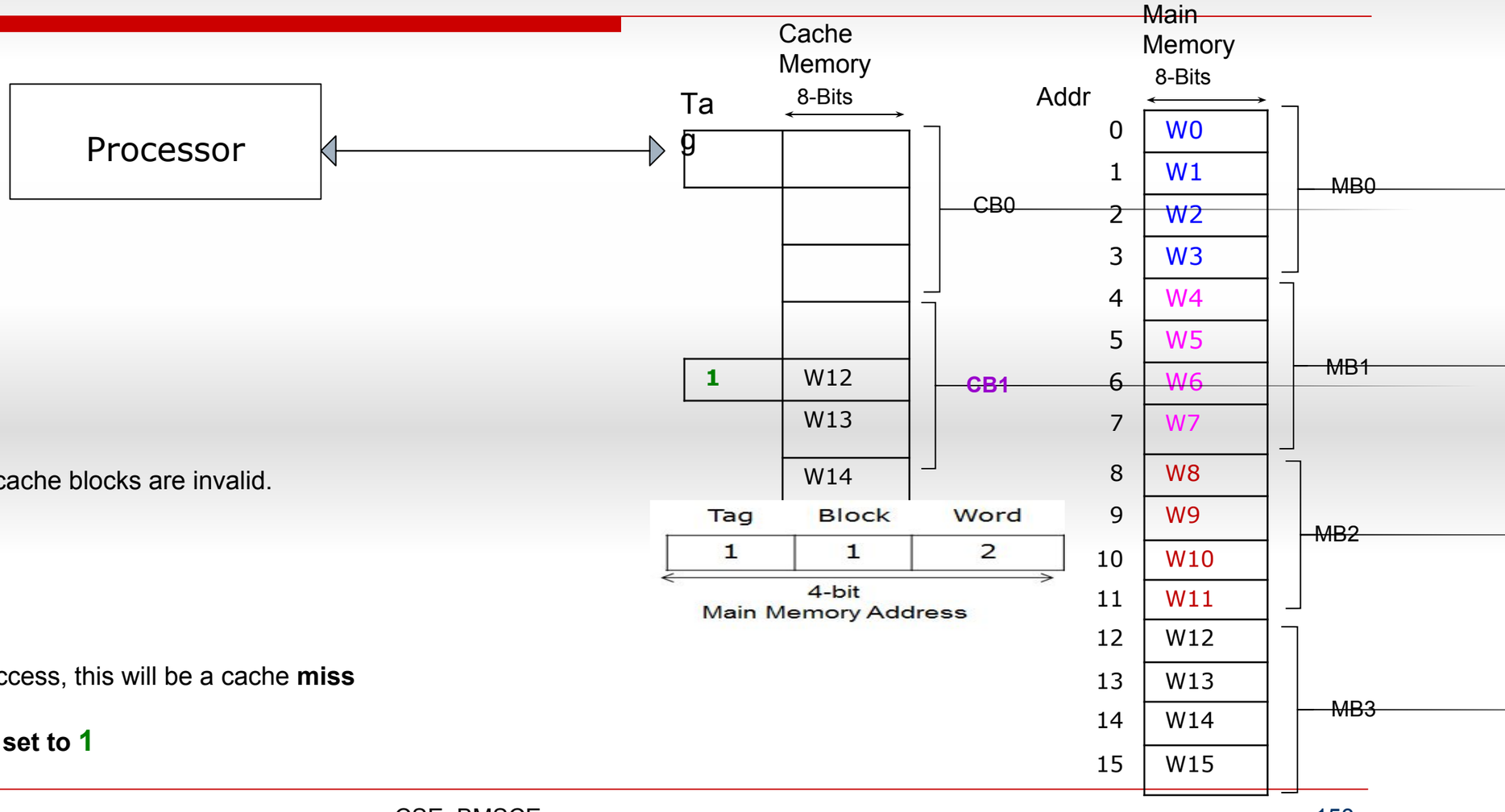
Address = $(12)_{10}$ = $(1100)_2$

For this address, *Tag* = **1**, *Block* = **1**, *Word* = 00

Since tag field for cache block **1** is **1** before this access, this will be a cache **miss** (address tag ≠ block tag)

After this access, **Tag field for cache block 1 is set to 1**

Answer (Contd…)



(b) The cache is initially empty. Therefore, all the cache blocks are invalid.

Access # 1:

Address = $(12)_{10}$ = $(1100)_2$

For this address, *Tag* = **1**, *Block* = **1**, *Word* = 00

Since tag field for cache block **1** is **1** before this access, this will be a cache **miss** (address tag ≠ block tag)

After this access, **Tag field for cache block 1 is set to 1**

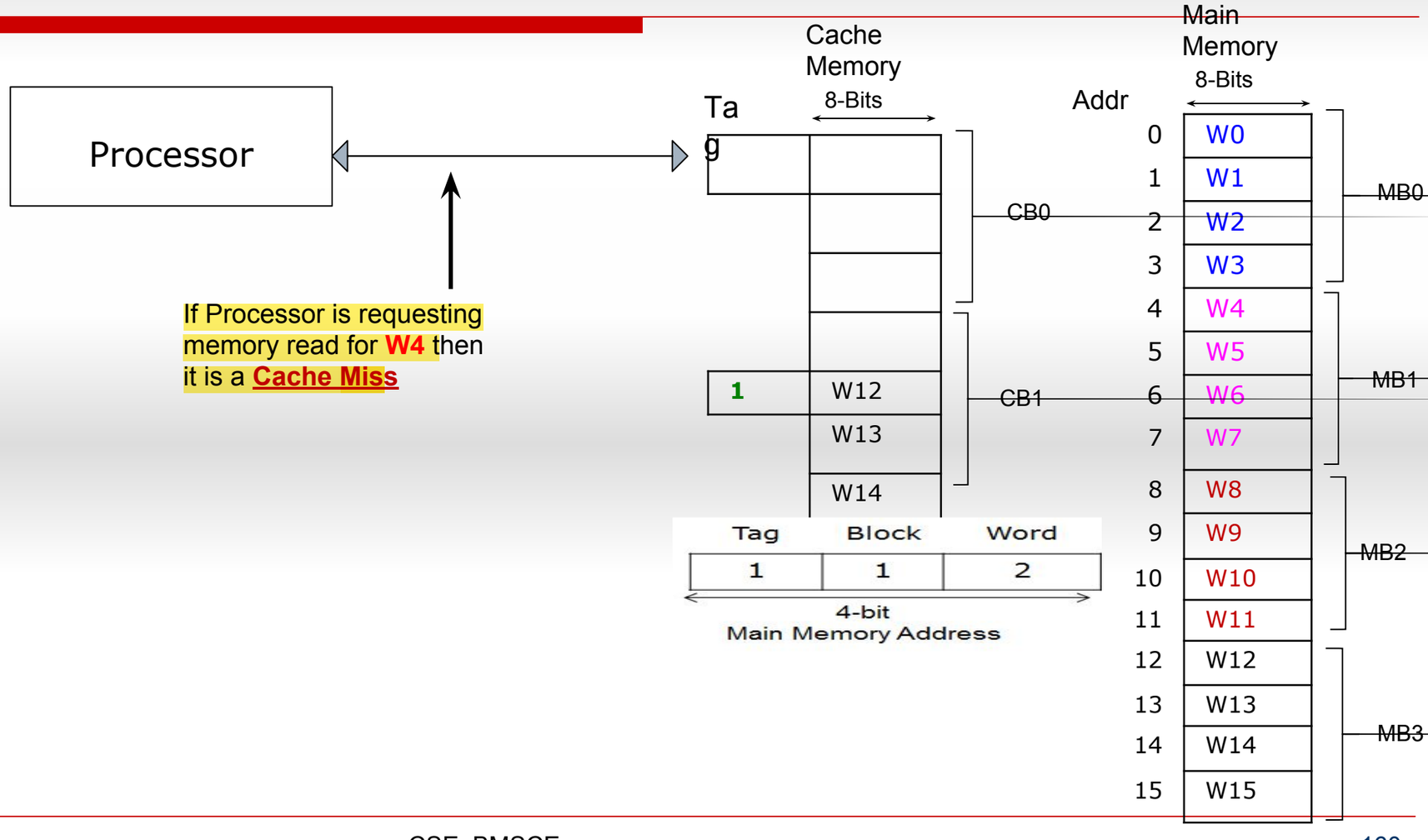
Question

(b) When a program is executed, the processor reads data sequentially from the following word addresses:

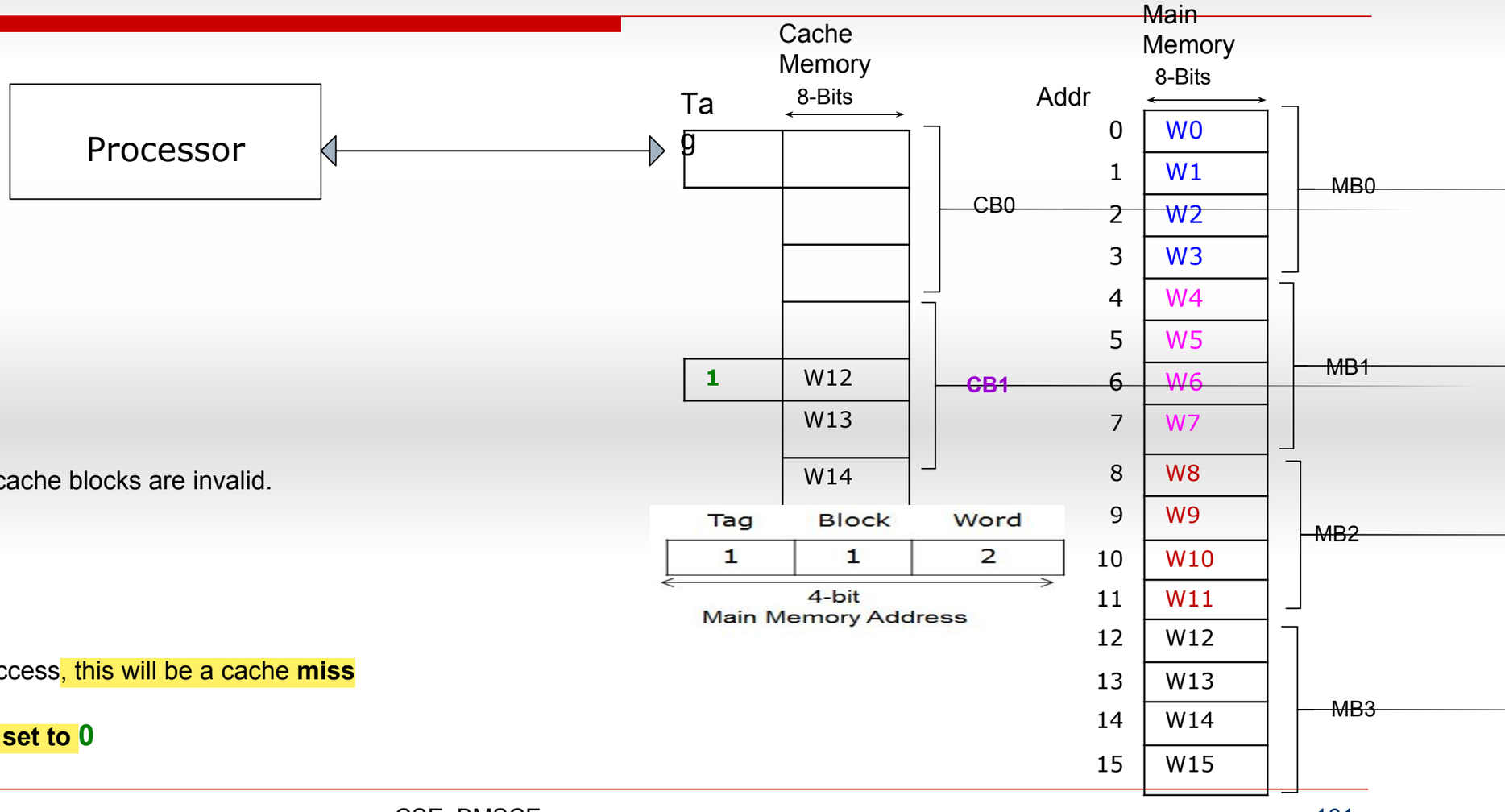
5, 7, 12, **4**

All the above addresses are shown in decimal values. Assume that the cache is initially empty. For each of the above addresses, indicate whether the cache access will result in a **hit** or a **miss**.

Answer (Contd…)



Answer (Contd…)



(b) The cache is initially empty. Therefore, all the cache blocks are invalid.

Access # 4:

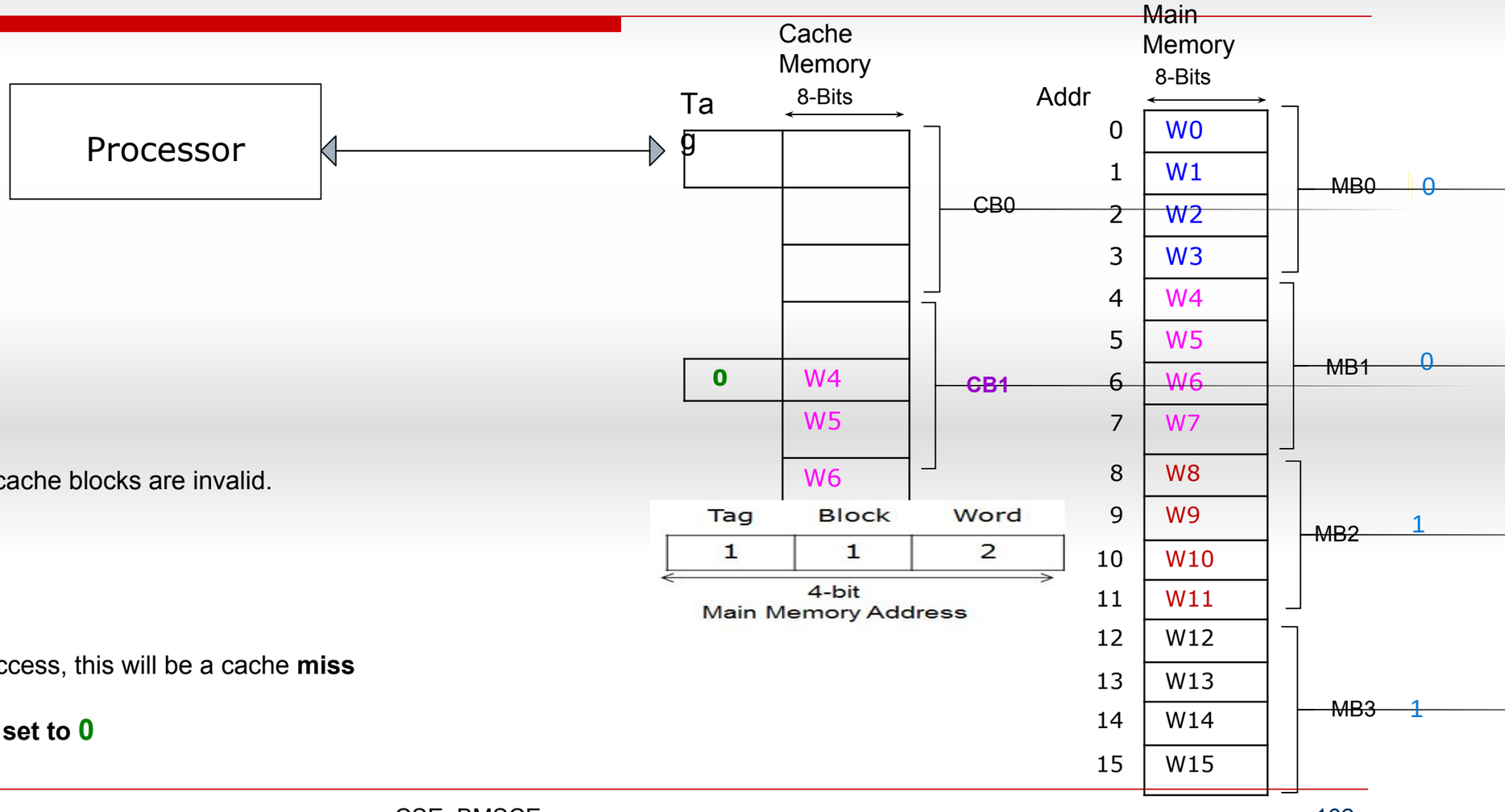
Address = $(4)_{10}$ = $(0100)_2$

For this address, *Tag* = **0**, *Block* = **1**, *Word* = 00

Since tag field for cache block **1** is **1** before this access, this will be a cache **miss** (address tag ≠ block tag)

After this access, **Tag field for cache block 1 is set to 0**

Answer (Contd…)



(b) The cache is initially empty. Therefore, all the cache blocks are invalid.

Access # 1:

Address = $(4)_{10}$ = $(0100)_2$

For this address, *Tag* = **0**, *Block* = **1**, *Word* = 00

Since tag field for cache block **1** is **1** before this access, this will be a cache **miss** (address tag ≠ block tag)

After this access, **Tag field for cache block 1 is set to 0**

Advantages and Disadvantages of Direct mapping

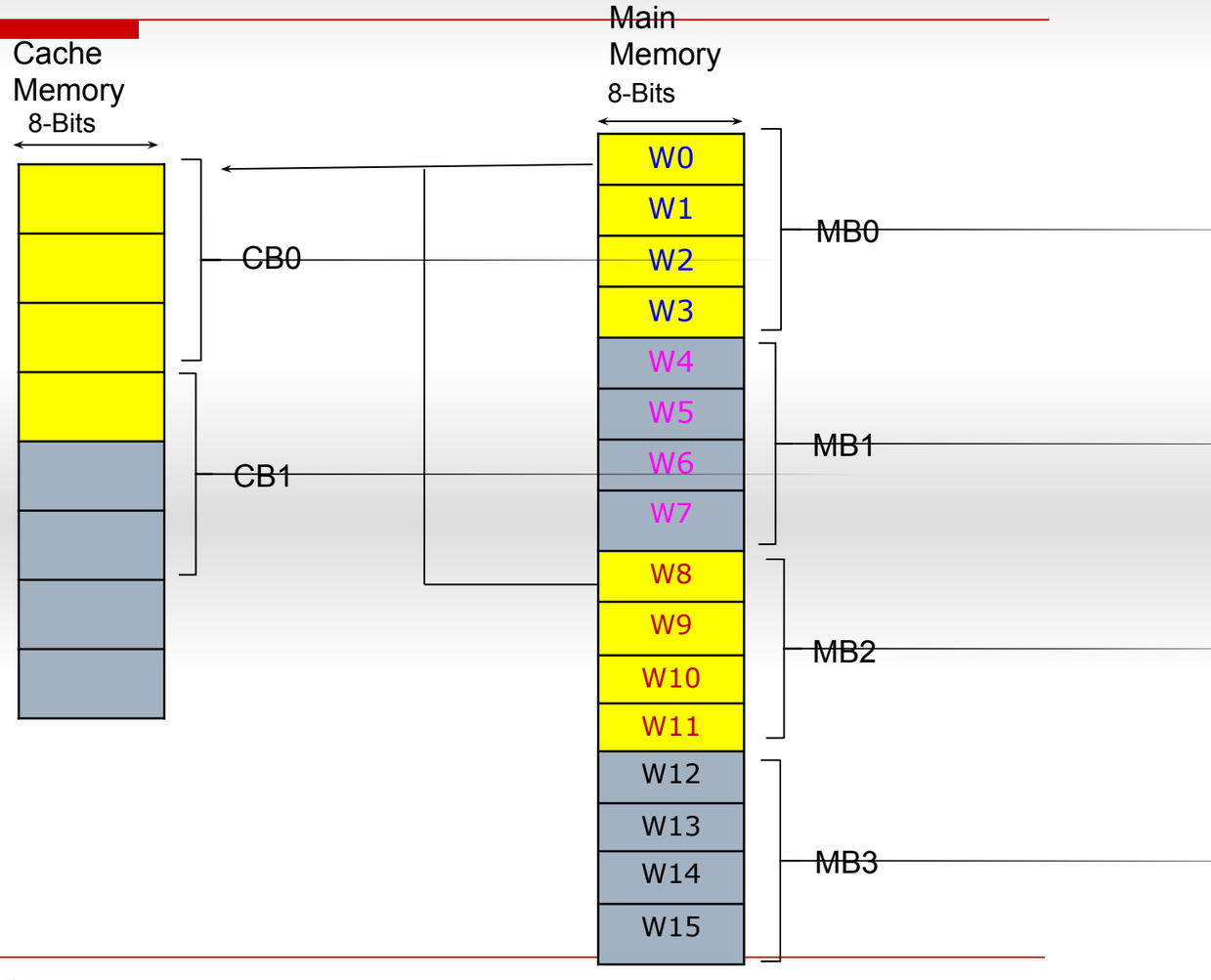
Advantages of direct mapping:

1. The technique is simple
2. The mapping scheme is easy to implement

Disadvantage of direct mapping:

1. Each block of main memory maps to a fixed location in the cache; therefore, if two different blocks map to the same location in cache and they are continually referenced, the two blocks will be continually swapped in and out (known as thrashing or cache contention).

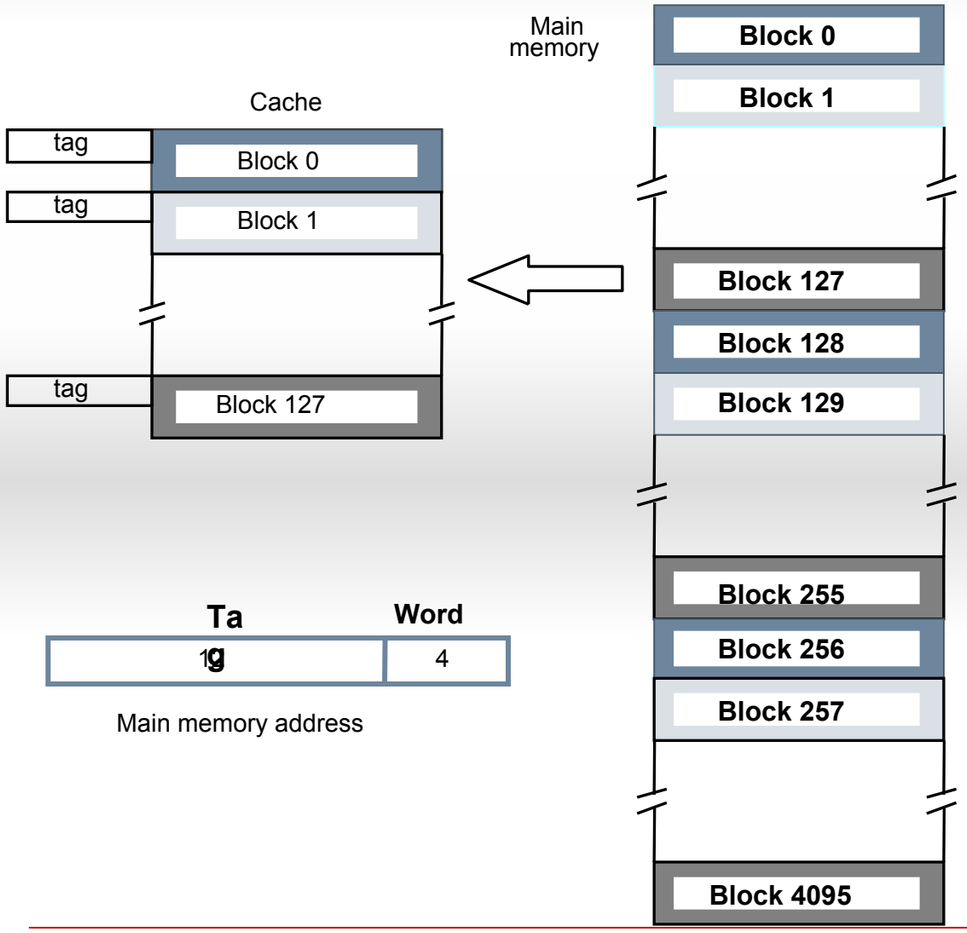
Rough slide to explain: Disadvantage of Direct mapping



Cache Replacement or Cache Mapping functions or Cache Mapping Techniques

- Direct mapping
- Associative mapping
- Set-associative mapping

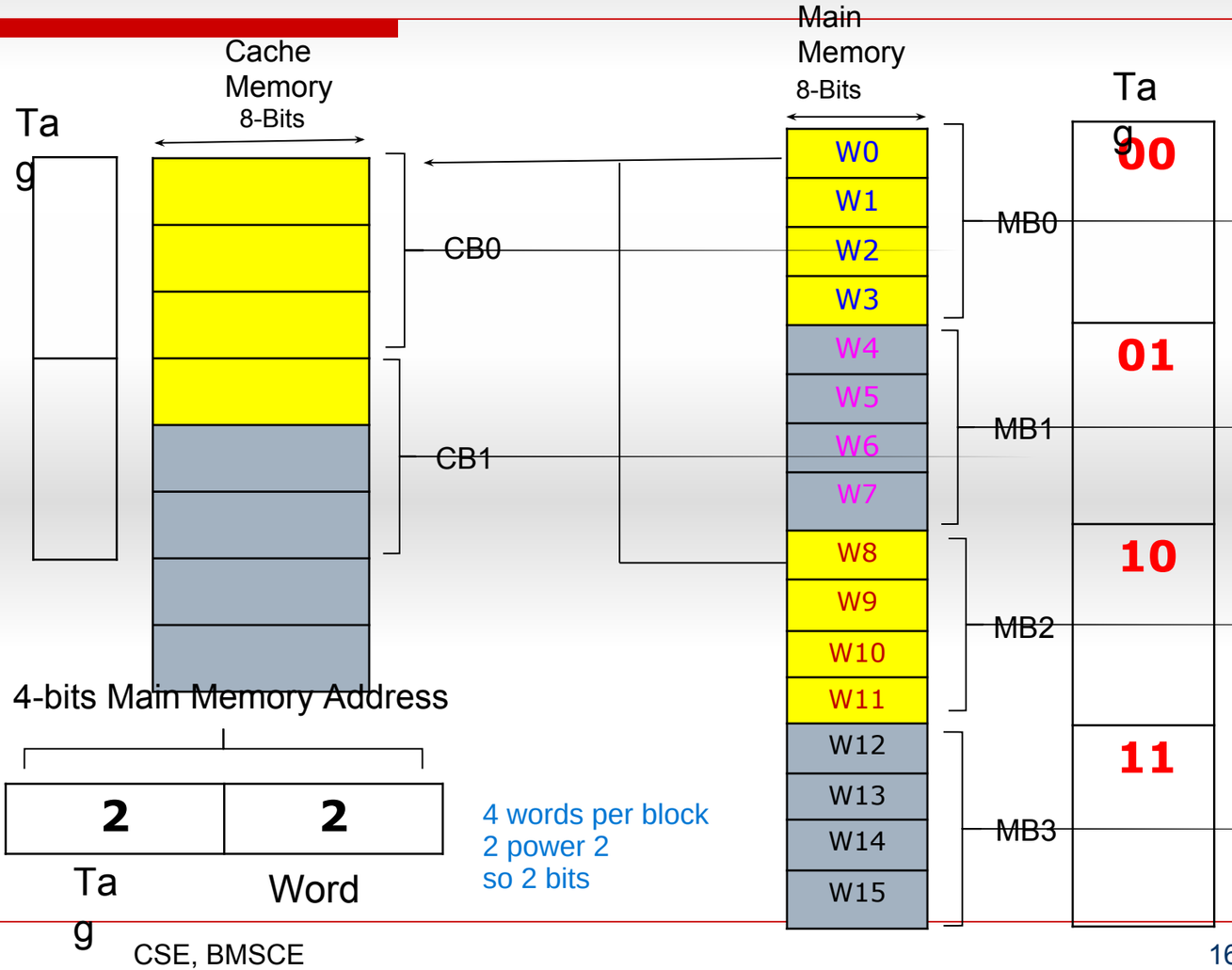
Associative Mapping Example



- *Main memory block can be placed into any cache position.*
- *Memory address is divided into two fields:* WORD AND TAG
  - *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.*

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

Example for Associate Mapping



Advantages and Disadvantages of Associative mapping

Advantage of associative mapping:

1. There is flexibility when mapping a block to any block of the cache

Disadvantages of associative mapping:

1. A replacement algorithm must be used to determine which block of cache to swap out
2. More space is needed for the tag field
3. The most important disadvantage is the complex circuitry needed to examine all of the tags in parallel in the cache

Cache Replacement or Cache Mapping functions or Cache Mapping Techniques

- Direct mapping
- Associative mapping
- Set-associative mapping

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.

Example for Set-Associative mapping

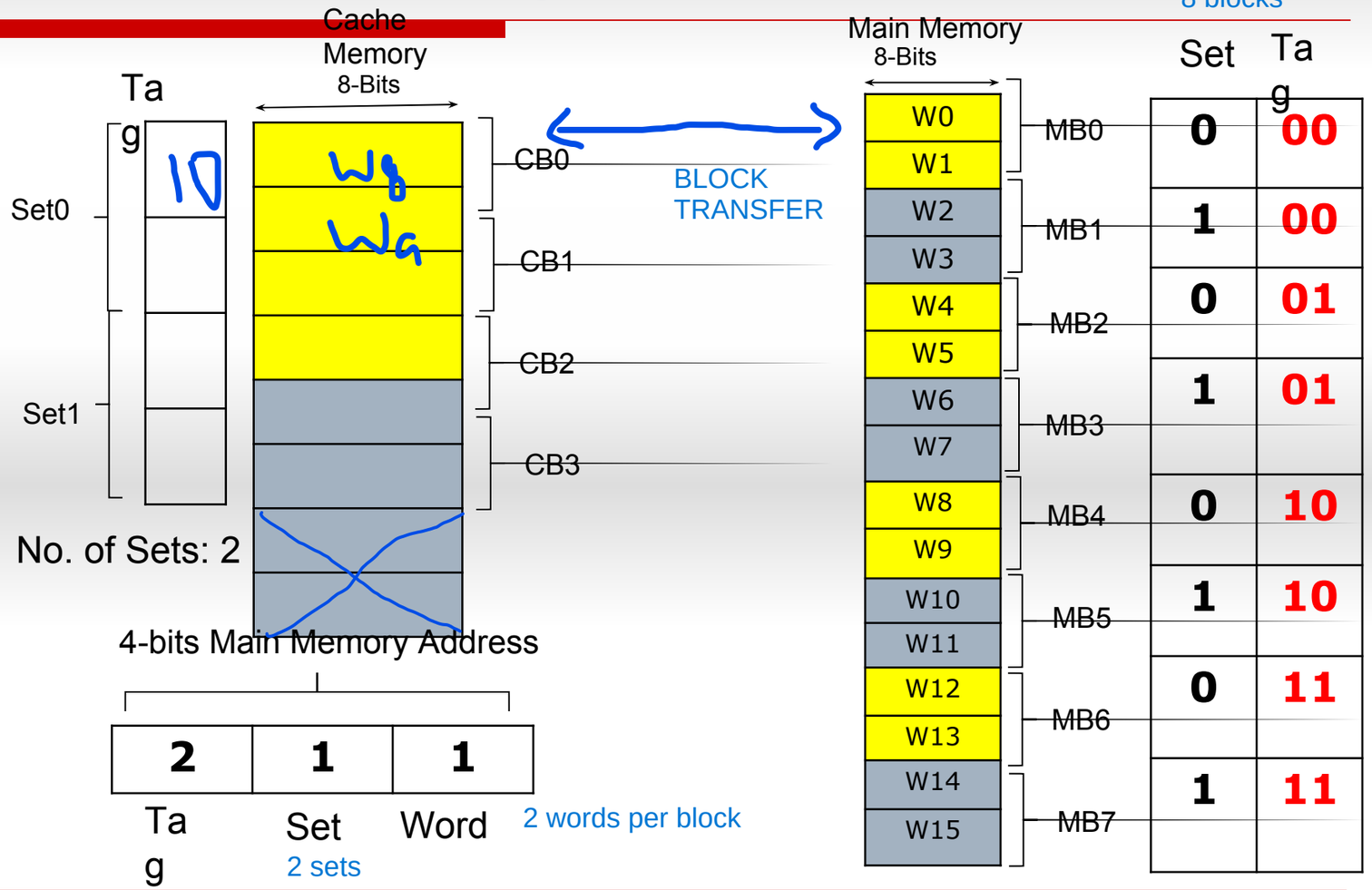
W0 0 0 0 0
0 0 0 1
0 0 1 0
0 0 1 1

0 1 0 0
0 1 0 1
0 1 1 0
0 1 1 1

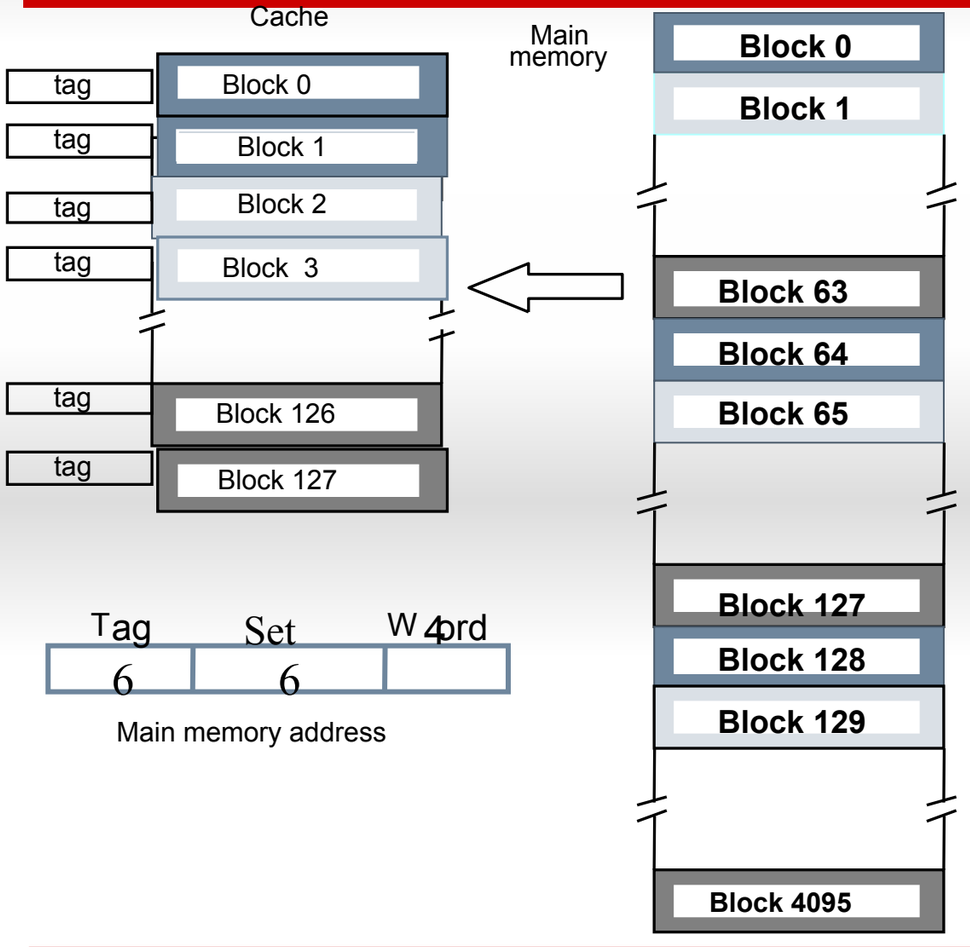
1 0 0 0
1 0 0 1
1 0 1 0
1 0 1 1

1 1 0 0
1 1 0 1
1 1 1 0
W15 1 1 1 1

e.g processor is requesting W9
W9==== 1001
LSB == word
1== second word
second from LSB == set (set 0)



Example for 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.*

Question

A block-set-associate cache consists of a total of 64 blocks divided into 4-block sets *(i.e., four blocks per set)*. The main memory contains 4096 blocks, each consisting of 128 words.

i. How many bits are there in a main memory address ? 12 bits

ii. How many bits are there in each of the TAG, SET and WORD fields ?

Question

A block-set-associate cache consists of a total of 64 blocks divided into 4-block sets (i.e., four blocks per set). The main memory contains 4096 blocks, each consisting of 128 words.

i. How many bits are there in a main memory address ?

ii. How many bits are there in each of the TAG, SET and WORD fields ?

Answer

i. Main Memory size=4096 blocks x128 words

$=4x1024x128$

$=2^2x2^{10}x2^7=2^{19}$

Hence number of bits for address is 19-bits

Question

A block-set-associate cache consists of a total of 64 blocks divided into 4-block sets (i.e., four blocks per set). The main memory contains 4096 blocks, each consisting of 128 words.

i. How many bits are there in a main memory address ?

ii. How many bits are there in each of the TAG, SET and WORD fields ?

Answer

i. Main Memory size=4096x128=4x1024x128=2^2x2^{10}x2^7=2^{19}

Hence number of bits for address is 19-bits

ii.

Number of bits for WORD: BlockSize=128 words = 2^7 ,Hence **7** bits

Question

A block-set-associate cache consists of a total of 64 blocks divided into 4-block sets (i.e., four blocks per set). The main memory contains 4096 blocks, each consisting of 128 words.

i. How many bits are there in a main memory address ?

ii. How many bits are there in each of the TAG, SET and WORD fields ?

Answer

i. Main Memory size=4096x128=4x1024x128=2^2x2^{10}x2^7=2^{19}

Hence number of bits for address is 19-bits

ii.

Number of bits for WORD: BlockSize=128 words = 2^7 ,Hence **7** bits

Number of Sets in Cache =(64/4) = 16 = 2^4 , Hence **4** bits for Set

Question

A block-set-associate cache consists of a total of 64 blocks divided into 4-block sets (i.e., four blocks per set). The main memory contains 4096 blocks, each consisting of 128 words.

i. How many bits are there in a main memory address ?

ii. How many bits are there in each of the TAG, SET and WORD fields ?

Answer

i. Main Memory size=4096x128=4x1024x128=2^2x2^{10}x2^7=2^{19}

Hence number of bits for address is 19-bits

ii.

Number of bits for WORD: BlockSize=128 words = 2^7 ,Hence **7** bits

Number of Sets in Cache =(64/4) = 16 = 2^4 , Hence **4** bits for Set

Therefore, TAG field bits = 19 – (**7** +**4**) = 8 bits.

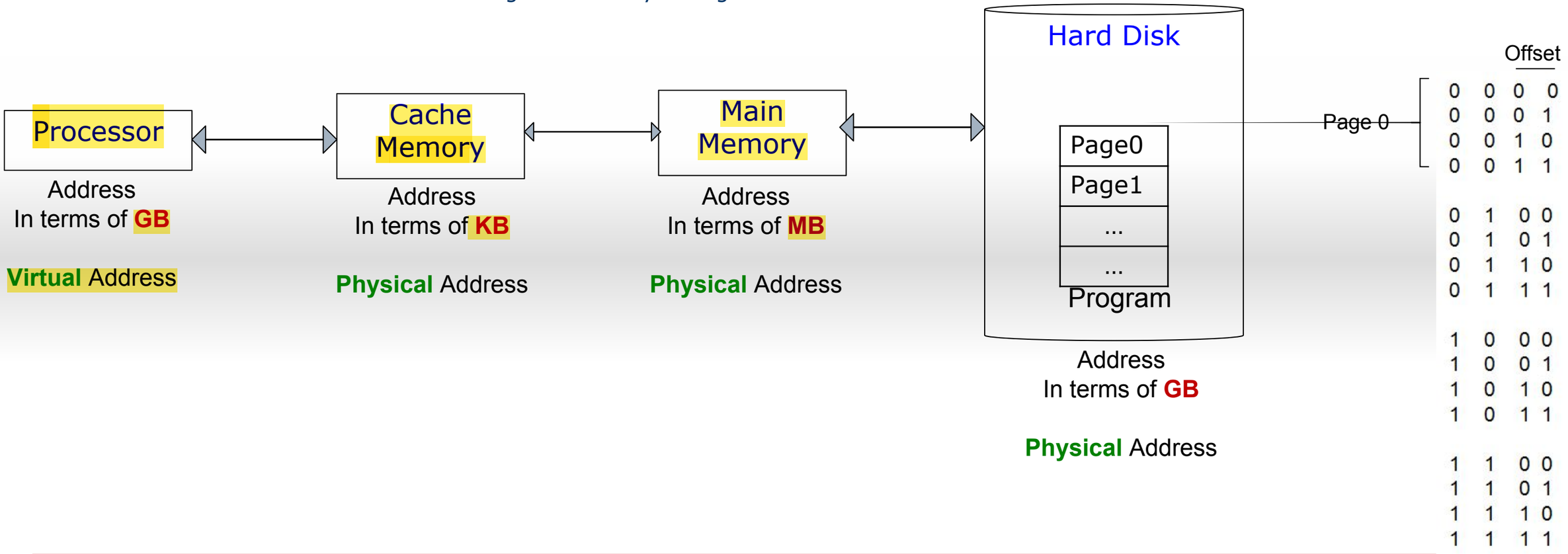
Virtual Memory

Virtual Memory: Creates a illusion to users of very large main memory

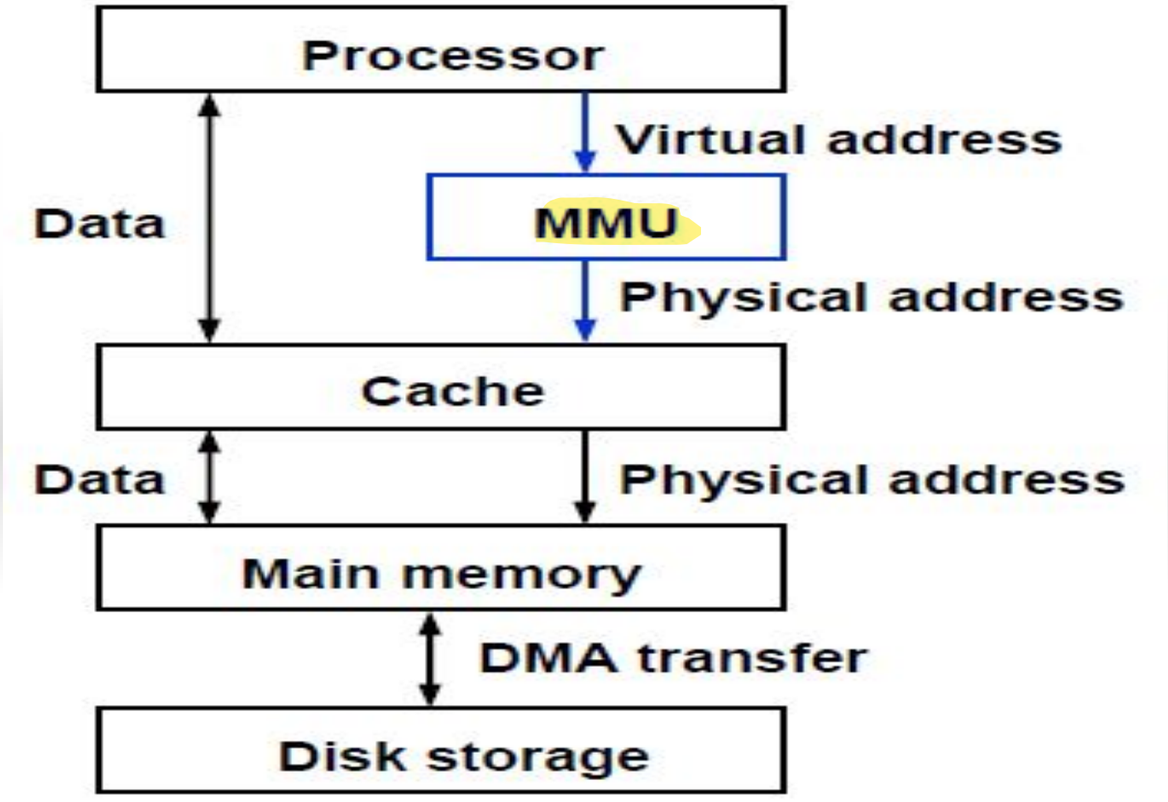
Note: Virtual: "appearing to exist, but not really there"

Rough Slide to explain Virtual Memory

Virtual memory is another memory organization technique in which only the active portions of a program are stored in the main memory, and the remainder is stored on the much larger secondary storage device.



Virtual Memory Organization

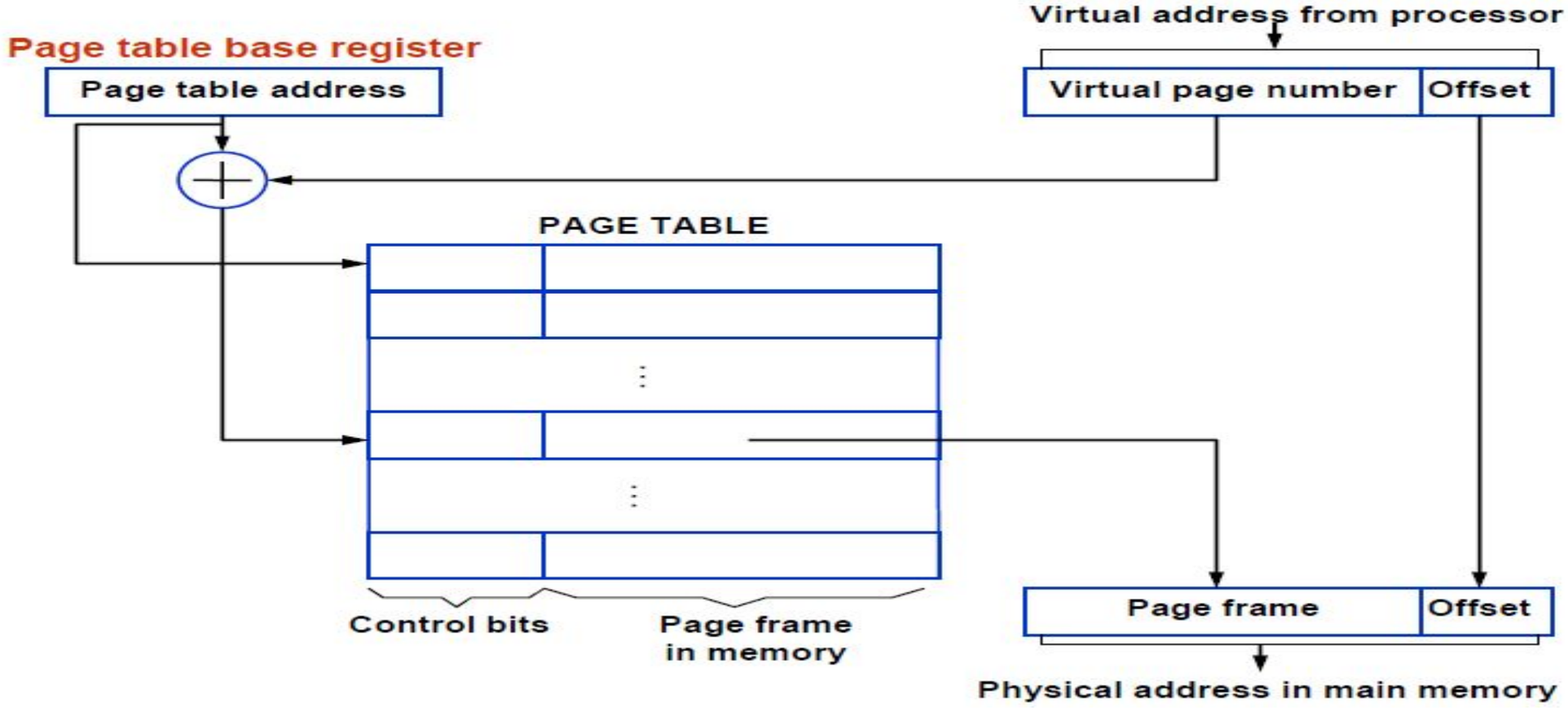


MMU: Memory Management Unit

Virtual Memory Organization (Contd…)

- It refers to a technique that automatically move program/data blocks into the main-memory when they are required for execution
- The address generated by the processor is referred to as a **virtual/logical address**.
- The virtual-address is translated into physical-address by **MMU** (Memory Management Unit).
- During every memory-cycle, MMU determines whether the addressed-word is in the memory.
  - If the word is in memory.
    - Then, the word is accessed and execution proceeds.
    - Otherwise, a page containing desired word is transferred from disk to memory.
- Using DMA scheme, transfer of data between disk and memory is performed.

Virtual Memory Address Translation

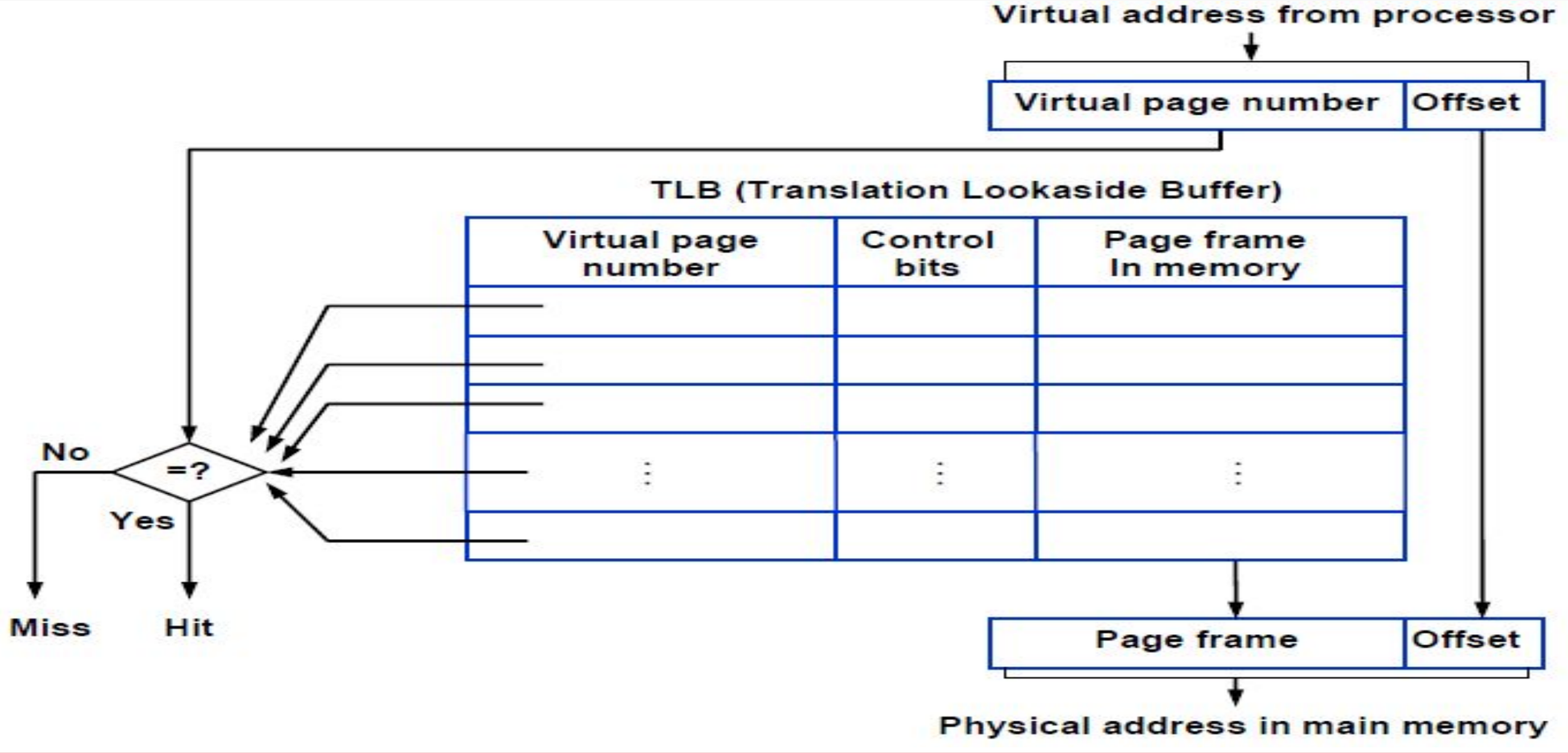


Virtual Address Translation (Contd…)

- All programs and data are composed of fixed length units called **Pages**
  The Page consists of a block-of-words. The words occupy contiguous locations in the memory.
  The pages are commonly range from 2K to 16K bytes in length.
- **Cache Bridge** speed-up the gap between main-memory and secondary-storage.
- Each virtual-address contains
  1) Virtual Page number (Low order bit) and
  2) Offset (High order bit).

  Virtual Page number + Offset → specifies the location of a particular word within a page.
- **Page-table:** It contains the information about
  → memory-address where the page is stored &
  → current status of the page.
- **Page-frame:** An area in the main-memory that holds one page.
- **Page-table Base Register:** It contains the starting address of the page-table
- *Virtual Page Number + Page-table Base register* → Gives the starting address of the page if that page currently resides in memory.
- **Control-bits in Page-table:** The Control-bits is used to
  1) Specify the status of the page while it is in memory.
  2) Indicate the validity of the page.
  3) Indicate whether the page has been modified during its stay in the memory.

Use of an Associative-Mapped TLB (Translation Lookaside-Buffer)



Use of an Associative-Mapped TLB (Contd…)

- The Page-table information is used by MMU for every read/write access .
- The Page-table is placed in the memory but a copy of small portion of the page-table is located within MMU. This small portion is called **TLB** (Translation LookAside Buffer).

  TLB consists of the page-table entries that corresponds to the most recently accessed pages.

  TLB also contains the virtual-address of the entry.

- When OS changes contents of page-table, the control-bit will invalidate corresponding entry in TLB.
- Given a virtual-address, the MMU looks in TLB for the referenced-page.

  If page-table entry for this page is found in TLB, the physical-address is obtained immediately.

  Otherwise, the required entry is obtained from the page-table & TLB is updated.

Page Faults

- Page-fault occurs when a program generates an access request to a page that is not in memory.
- When MMU detects a page-fault, the MMU asks the OS to generate an interrupt.
- The OS
  - → suspends the execution of the task that caused the page-fault and
  - → begins execution of another task whose pages are in memory.
- When the task resumes the interrupted instruction must continue from the point of interruption.
- If a new page is brought from disk when memory is full, disk must replace one of the resident pages.

  In this case, **LRU algorithm** is used to remove the least referenced page from memory.
- A modified page has to be written back to the disk before it is removed from the memory.

  In this case, **Write–Through Protocol** is used.

Question

Virtual memory is _____.

a. An extremely large main memory

b. An extremely large secondary memory

c. An illusion of extremely large main memory

d. A type of memory used in super computers

Question

Virtual memory is _____.

a. An extremely large main memory

b. An extremely large secondary memory

c. **An illusion of extremely large main memory**

d. A type of memory used in super computers

Question

_____ translates the logical address into a physical address.

a) MMU
b) Translator
c) Compiler
d) Linker

Question

_____ translates the logical address into a physical address.

a) MMU

b) Translator

c) Compiler

d) Linker

Answer: a

Explanation: **The MMU (Memory management Unit)** translates the logical address into a physical address by adding an offset.

Question

The associatively mapped virtual memory makes use of _____

a) TLB
b) Page table
c) Frame table
d) None of the mentioned

Question

The associatively mapped virtual memory makes use of _____

a) TLB
b) Page table
c) Frame table
d) None of the mentioned

Answer: a
Explanation: TLB stands for **Translation Look-aside Buffer**.

Question

W.r.t Virtual Memory, What is a page fault ?

- Is an spelling error in a page in memory
- Is a reference to a page which is in another program
- Is an access to a page not currently in main memory
- Always occurs whenever a page is accessed

Question

W.r.t Virtual Memory, What is a page fault ?

- Is an spelling error in a page in memory
- Is a reference to a page which is in another program
- **Is an access to a page not currently in main memory**
- Always occurs whenever a page is accessed

Page fault

When a program generates an access request to a page that is not in the main memory, a **page fault** is said to have occurred. The whole page must be brought from the disk into the memory before access can proceed

When it detects a page fault, the MMU asks the operating system to intervene by raising an exception (interrupt). Processing of the active task is interrupted, and control is transferred to the operating system. The operating system then copies the requested page from the disk into the main memory and returns control to the interrupt task.

Question

1024x1024 array of 32-bit numbers is to be normalized as follows. For each column the largest element is found and all elements of the column are divided by this maximum value. Assume that each page in the virtual memory consists of 4Kbytes and that 1Mbytes of the main memory are allocated for storing data during this computation. Suppose that it takes 40 ms to load a page from the disk to the main memory when a page fault occurs (assume that when we start, the main memory is empty).

(a) How many page faults would occur if the elements of the array are stored in column order in the virtual memory?

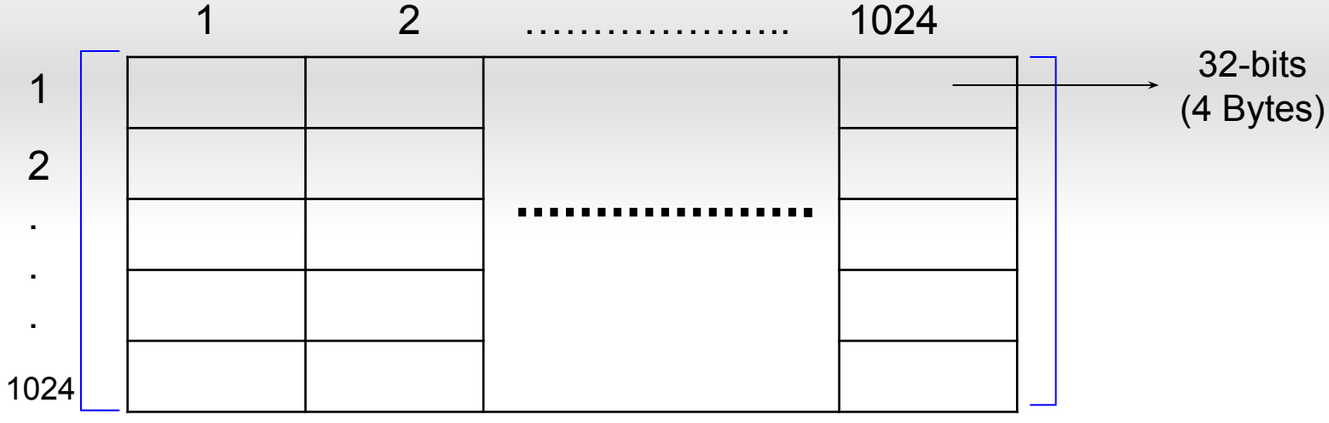
(b) Estimate the total time needed to perform this normalization for the arrangement (a).

Rough slide to explain the Question

1024x1024 array of 32-bit numbers is to be normalized as follows. For each column the largest element is found and all elements of the column are divided by this maximum value. Assume that each page in the virtual memory consists of 4Kbytes and that 1Mbytes of the main memory are allocated for storing data during this computation. Suppose that it takes 40ms to load a page from the disk to the main memory when a page fault occurs (assume that when we start, the main memory is empty).

(a) How many page faults would occur if the elements of the array are stored in column order in the virtual memory?

(b) Estimate the total time needed to perform this normalization for the arrangement (a).



Example of 3x2 matrix

10	20
30	40
50	60

Storing in Column order: 10 , 30, 50, 20, 40, 60

Storing in Row order: 10, 20, 30, 40, 50, 60

Answer

- Each 32-bit number comprises 4 bytes.

Answer

- Each 32-bit number comprises 4 bytes.
- Page Size = 4Kbytes

$= 2^2 x 2^{10}$ $2^{10} = 1024 = 1Kilo$

$= 2^{12}$ bytes

Answer

- Each 32-bit number comprises 4 bytes.
- Page Size = 4Kbytes

$= 2^2 x 2^{10}$ $2^{10}=1024 = 1Kilo$

$= 2^{12}$ bytes

Main Memory Size= 1MBytes

$= 2^{20}$ Bytes

Answer (Contd…)

Each 32-bit number comprises 4 bytes.

Page Size = 4Kbytes

$= 2^2 \times 2^{10}$

$= 2^{12}$ bytes

$2^{10} = 1024 = 1\text{Kilo}$

Main Memory Size= 1MBytes

$= 2^{20}$ Bytes

$$\text{No. Of Pages in Main Memory} = \frac{2^{20}}{2^{12}} = 2^8 = 256 \text{ Pages}$$

Answer (Contd…)

Each 32-bit number comprises 4 bytes.

Page Size = 4Kbytes

$= 2^2 \times 2^{10}$ $\qquad 2^{10} = 1024 = 1\text{Kilo}$

$= 2^{12}$ bytes

Main Memory Size= 1MBytes

$= 2^{20}$ Bytes

No. Of Pages in Main Memory $= \frac{2^{20}}{2^{12}} = 2^8 = 256$ Pages

No. of Pages required to store 1024x1024 matrix

$$= \frac{1024 \; x \; 1024 \; x \; 4}{2^{12}} = \frac{2^{22}}{2^{12}} = 2^{10} = 1024 \text{ Pages}$$

Answer (Contd…)

Each 32-bit number comprises 4 bytes.
Page Size = 4Kbytes
$= 2^2 \times 2^{10}$ $2^{10} = 1024 = 1\text{Kilo}$
$= 2^{12}$ bytes
Main Memory Size= 1MBytes
$= 2^{20}$ Bytes

No. Of Pages in Main Memory $= \frac{2^{20}}{2^{12}} = 2^8 = 256$ Pages

No. of Pages required to store 1024x1024 matrix

$$= \frac{1024 \times 1024 \times 4}{2^{12}} = \frac{2^{22}}{2^{12}} = 2^{10} = 1024 \text{ Pages}$$

a. One column size = 1024x4bytes = $2^{10} \times 2^2 = 2^{12}$, therefore **One page** can hold 1024 numbers.

1024 Columns has to be accessed, therefore 1024 pages has to be accessed . Hence number of Page faults is 1024

Answer (Contd…)

Each 32-bit number comprises 4 bytes.
Page Size = 4Kbytes
$= 2^2 \times 2^{10}$ $2^{10} = 1024 = 1\text{Kilo}$
$= 2^{12}$ bytes
Main Memory Size= 1MBytes
$= 2^{20}$ Bytes

No. Of Pages in Main Memory $= \frac{2^{20}}{2^{12}} = 2^8 = 256$ Pages

No. of Pages required to store 1024x1024 matrix

$$= \frac{1024 \times 1024 \times 4}{2^{12}} = \frac{2^{22}}{2^{12}} = 2^{10} = 1024 \text{ Pages}$$

a. One column size = 1024x4bytes = $2^{10} \times 2^2 = 2^{12}$, therefore **One page** can hold 1024 numbers.
1024 Columns has to be accessed, therefore 1024 pages has to be accessed . Hence number of Page faults is 1024

b. Assuming that the computation time needed to normalize the numbers is negligible compared to the time needed to bring a page from the disk:

Total time for (a) is 1024 x40 ms = **41 s**

Thanks for Listening

CMOS(Complementary Metal Oxide Semiconductor) Cell

A CMOS realization of the cell in Figure 1 is given in Figure 2. Transistor pairs (*T*3, *T*5) and (*T*4, *T*6) form the inverters in the latch. The state of the cell is read or written. For example, in state 1, the voltage at point *X* is maintained high by having transistors *T*3 and *T*6 on, while *T*4 and *T*5 are off. If *T*1 and *T*2 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, *T*1, *T*2, 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.

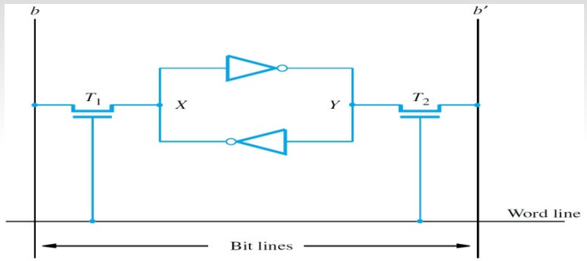


Figure 1: A static RAM Cell

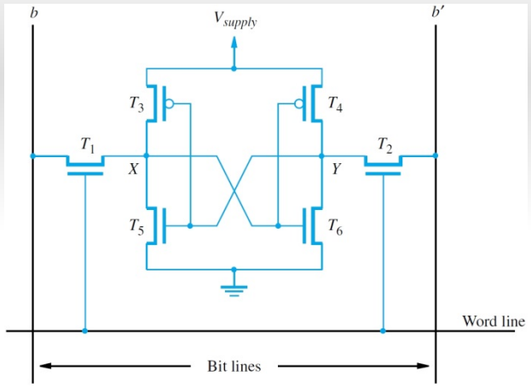


Figure 2: An example of CMOS memory Cell