Reg.No: | | | | | | |

SNS COLLEGE OF TECHNOLOGY

(An Autonomous Institution)
Coimbatore – 641 035.

B.E / B.Tech – Internal Assessment Exam- III
Academic Year 2023-2024 (ODD)
FIFTH SEMESTER (REGULATION R2019)

A

19ITT202 – COMPUTER ORGANIZATION AND ARCHITECTURE

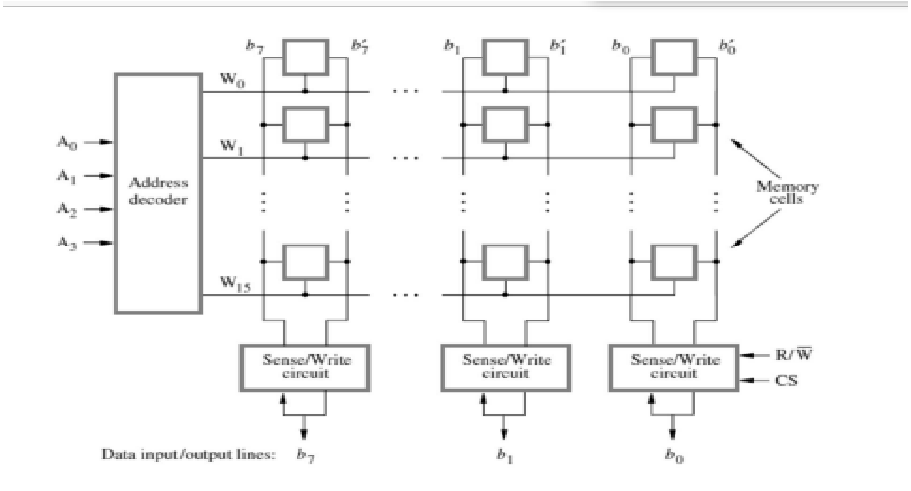
TIME: 1.5 HOURS **MAXIMUM MARKS: 50**

ANSWER ALL QUESTIONS

PART A — (5 x 2 = 10 Marks)			
1.	Differentiate SRAM from DRAM Static RAMs are fast, but their cells require several transistors. Less expensive and higher density RAMs can be implemented with simpler cells. But, these simpler cells do not retain their state for a long period, unless they are accessed frequently for Read or Write operations. Memories that use such cells are called dynamic RAMs (DRAMs). Information is stored in a dynamic memory cell in the form of a charge on a capacitor, but this charge can be maintained for only tens of milliseconds.	CO4	UND
2.	What is memory access time? A useful measure of the speed of memory units is the time that elapses between the initiation of an operation to transfer a word of data and the completion of that operation. This is referred to as the memory access time	CO4	REM
3.	Define read hit and write miss. The cache control circuitry determines whether the requested word currently exists in the cache. If it does, the Read or Write operation is performed on the appropriate cache location. In this case, a read or write hit is said to have occurred Write miss occurs in a computer that uses the write-through protocol, the information is written directly into the main memory	CO4	REM
4.	What is memory mapped I/O? Memory-mapped I/O uses the same address space to address both main memory and I/O devices. The memory and registers of the I/O devices are mapped to (associated with) address values, so a memory address may refer to either a portion of physical RAM or to memory and registers of the I/O device.	CO5	UND

5.	Mention the difference between subroutine and an Interrupt Subroutine. An important departure from this similarity should be noted. A subroutine performs a function required by the program from which it is called. As such, potential changes to status information and contents of registers are anticipated. However, an interrupt-service routine may not have any relation to the portion of the program being executed at the time the interrupt request is received. Therefore, before starting execution of the interrupt service routine, status information and contents of processor registers that may be altered in unanticipated ways during the execution of that routine must be saved. This saved information must be restored before execution of the interrupted program is resumed. In this way, the original program can continue execution without being affected in any way by the interruption, except for the time delay.	CO5	UND

PART- B (2 x 13 = 26 Marks , 1*14=14 Marks)

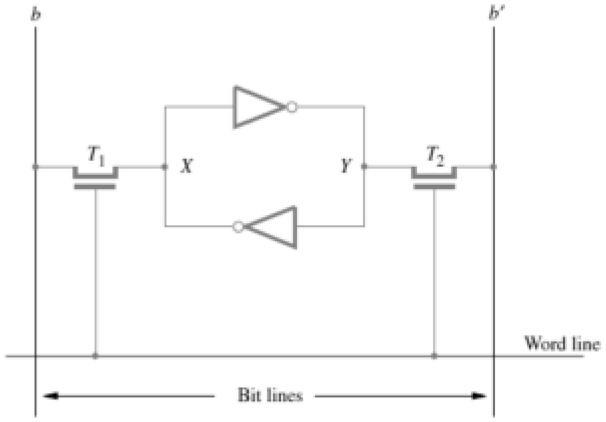
6.	(a)	Describe the basic concepts of Semiconductor RAM memories in detail with diagram. Semiconductor random-access memories (RAMs) are available in a wide range of speeds. Their cycle times range from 100 ns to less than 10 ns. In this section, we discuss the main characteristics of these memories. We start by introducing the way that memory cells are organized inside a chip. **Internal Organization of Memory Chips** Memory cells are usually organized in the form of an array, in which each cell is capable of storing one bit of information. A possible organization is illustrated in Figure. Each row of cells constitutes a memory word, and all cells of a row are connected to a common line referred to as the word line, which is driven by the address decoder on the chip. The cells in each column are connected to a Sense/Write circuit by two bit lines, and the Sense/Write circuits are connected to the data input/output lines of the chip. During a Read operation, these circuits sense, or read, the information stored in the cells selected by a word line and place this information on the output data lines. During a Write operation, the Sense/Write circuits receive input data and store them in the cells of the selected word.  b_7 b'_7 b_1 b'_1 b_0 b'_0 W_0 W_1 W_{15} A_0 A_1 A_2 A_3 Address decoder Memory cells Sense/Write circuit $R/\overline{W}$ CS Data input/output lines: b_7 b_1 b_0 Figure is an example of a very small memory circuit consisting of 16 words of 8 bits each. This is referred to as a 16 × 8 organization. The data input and the data output of each Sense/Write circuit are connected to a single bidirectional data line that can be connected to the data lines	13	CO4	UND

of a computer. Two control lines, R/W and CS, are provided. The R/W (Read/Write) input specifies the required operation, and the CS (Chip Select) input selects a given chip in a multichip memory system. The memory circuit in Figure stores 128 bits and requires 14 external connections for address, data, and control lines. It also needs two lines for power supply and ground connections. Consider now a slightly larger memory circuit, one that has 1K (1024) memory cells. This circuit can be organized as a 128 × 8 memory, requiring a total of 19 external connections. Alternatively, the same number of cells can be organized into a 1K×1 format. In this case, a 10-bit address is needed, but there is only one data line, resulting in 15 external connections.

STATIC MEMORIES

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

hat this state represents the value 1.



Read Operation

In order to read the state of the SRAM cell, the word line is activated to close switches T1 and T2. If the cell is in state 1, the signal on bit line b is high and the signal on bit line b’ is low. The opposite is true if the cell is in state 0. Thus, b and b’ are always complements of each other. The Sense/Write circuit at the end of the two bit lines monitors their state and sets the corresponding output accordingly.

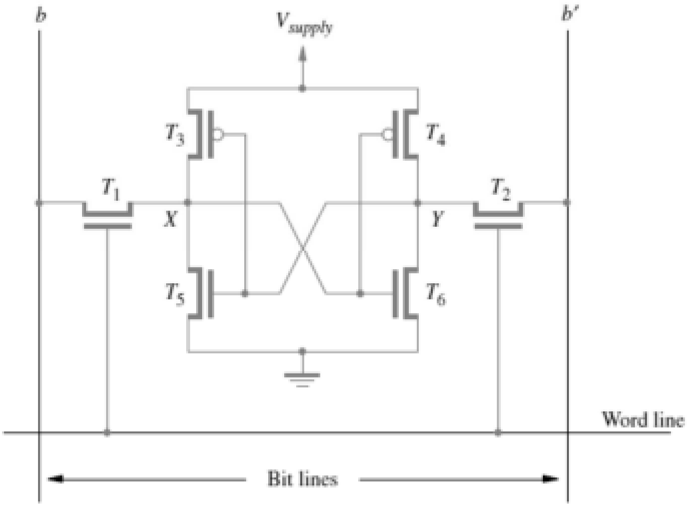
Write Operation

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

CMOS Cell

A CMOS realization of the cell in Figure is given in Figure. Transistor pairs (T3, T5) and (T4, T6) form the inverters in the latch (see Appendix A). The state of the cell is read or written as just explained.

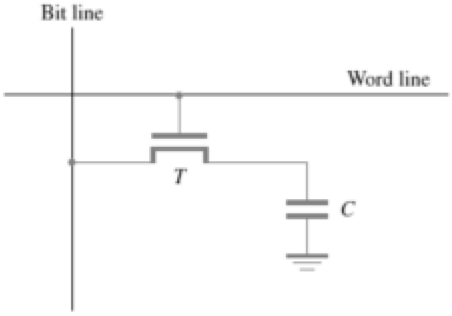
For example, in state 1, the voltage at point X is maintained high by having transistors T3 and T6 on, while T4 and T5 are off. If T1 and T2 are turned on, bit lines b and bwill have high and low signals, respectively.



Continuous power is needed for the cell to retain its state. If power is interrupted, the cell’s contents are lost. When power is restored, the latch settles into a stable state, but not necessarily the same state the cell was in before the interruption. Hence, SRAMs are said to be volatile memories because their contents are lost when power is interrupted. A major advantage of CMOS SRAMs is their very low power consumption, because current flows in the cell only when the cell is being accessed. Otherwise, T1, T2, and one transistor in each inverter are turned off, ensuring that there is no continuous electrical path between Vsupply and ground. Static RAMs can be accessed very quickly. Access times on the order of a few nanoseconds are found in commercially available chips. SRAMs are used in applications where speed is of critical concern.

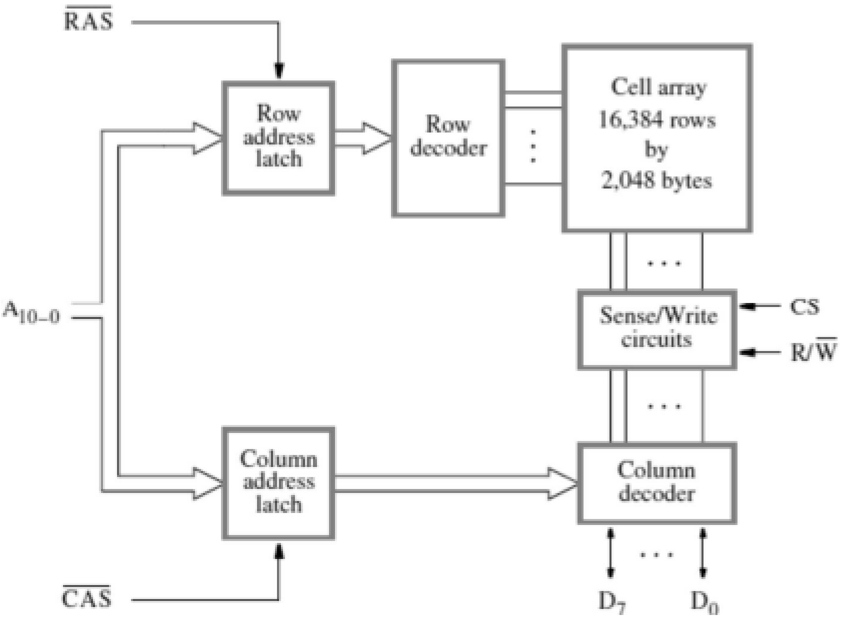
DYNAMIC RAMS

Static RAMs are fast, but their cells require several transistors. Less expensive and higher density RAMs can be implemented with simpler cells. But, these simpler cells do not retain their state for a long period, unless they are accessed frequently for Read or Write operations. Memories that use such cells are called dynamic RAMs (DRAMs). Information is stored in a dynamic memory cell in the form of a charge on a capacitor, but this charge can be maintained for only tens of milliseconds. Since the cell is required to store information for a much longer time, its contents must be periodically refreshed by restoring the capacitor charge to its full value. This occurs when the contents of the cell are read or when new information is written into it. An example of a dynamic memory cell that consists of a capacitor, C, and a transistor, T, is shown in Figure. To store information in this cell, transistor T is turned on and an appropriate voltage is applied to the bit line. This causes a known amount of charge to be stored in the capacitor.



After the transistor is turned off, the charge remains stored in the capacitor, but not for long. The capacitor begins to discharge. This is because the transistor continues to conduct a tiny amount of current, measured in picoamperes, after it is turned off. Hence, the information stored in the cell can be retrieved correctly only if it is read before the charge in the capacitor drops below some threshold value. During a Read operation, the transistor in a selected cell is turned on. A sense amplifier connected to the bit line detects whether the charge stored in the capacitor is above or below the threshold value. If the charge is above the threshold, the sense amplifier drives the bit line to the full voltage representing the logic value 1. As a result, the capacitor is recharged to the full charge corresponding to the logic value 1. If the sense amplifier detects that the charge in the capacitor is below the threshold value, it pulls the bit line to ground level to discharge the capacitor fully. Thus, reading the contents of a cell automatically refreshes its contents. Since the word line is common to all cells in a row, all cells in a selected row are read and refreshed at the same time.

A256-Megabit DRAM chip, configured as 32M × 8, is shown in Figure. 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 loworder 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.

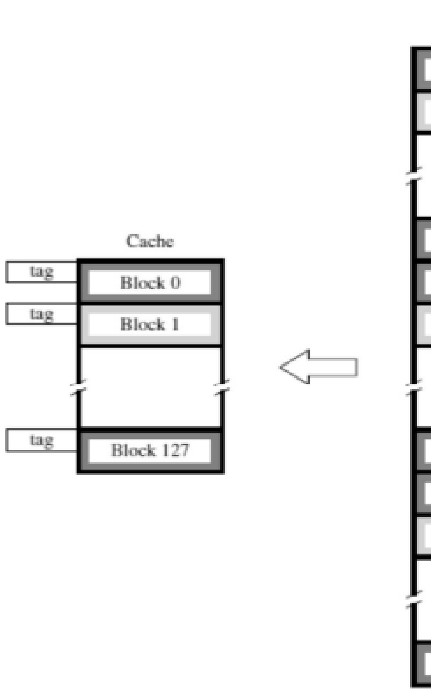
		 $\overline{RAS}$ Row address latch Row decoder Cell array 16,384 rows by 2,048 bytes A_{10-0} Sense/Write circuits CS $R/\overline{W}$ Column address latch Column decoder $\overline{CAS}$ D_7 D_0 Shortly after the row address is loaded, the column address is applied to the address pins and loaded into 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 theD7−0 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. The timing of the operation of the DRAM described above is controlled by the RAS and CAS signals. These signals are generated by a memory controller circuit external to the chip when the processor issues a Read or a Write command. During a Read operation, the output data are transferred to the processor after a delay equivalent to the memory's access time. Such memories are referred to as asynchronous DRAMs.			
(OR)					
	(b)	Discuss the various mapping schemes used in cache design There are several possible methods for determining where memory blocks are placed in the cache. It is instructive to describe these methods using a specific small example. Consider a cache consisting of 128 blocks of 16 words each, for a total of 2048 (2K) words, and assume that the main memory is addressable by a 16-bit address. The main memory has 64K words, which we will view as 4K blocks of 16 words each. For simplicity, we have assumed that consecutive addresses refer to consecutive words. **Direct Mapping**	13	CO4	UND

The simplest way to determine cache locations in which to store memory blocks is the direct-mapping technique. In this technique, block j of the main memory maps onto block j modulo 128 of the cache, as depicted in Figure. Thus, whenever one of the main memory blocks 0, 128, 256, . . . is loaded into the cache, it is stored in cache block 0. Blocks 1, 129, 257, . . . are stored in cache block 1, and so on. Since more than one memory block is mapped onto a given cache block position, contention may arise for that position even when the cache is not full. For example, instructions of a program may start in block 1 and continue in block 129, possibly after a branch. As this program is executed, both of these blocks must be transferred to the block-1 position in the cache. Contention is resolved by allowing the new block to overwrite the currently resident block.

With direct mapping, the replacement algorithm is trivial. Placement of a block in the cache is determined by its memory address. The memory address can be divided into three fields, as shown in Figure. The low-order 4 bits select one of 16 words in a block. When a new block enters the cache, the 7-bit cache block field determines the cache position in which this block must be stored.

The high-order 5 bits of the memory address of the block are stored in 5 tag bits associated with its location in the cache. The tag bits identify which of the 32 main memory blocks mapped into this cache position is currently resident in the cache. As execution proceeds, the 7-bit cache block field of each address generated by the processor points to a particular block location in the cache.

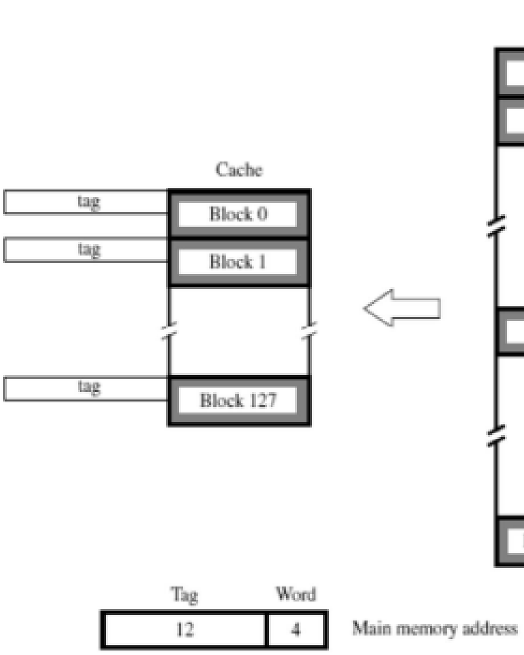
The high-order 5 bits of the address are compared with the tag bits associated with that cache location. If they match, then the desired word is in that block of the cache. If there is no match, then the block containing the required word must first be read from the main memory and loaded into the cache. The direct-mapping technique is easy to implement, but it is not very flexible.



Associative Mapping

Figure shows the most flexible mapping method, in which a main memory block can be placed into any cache block position. In this case, 12 tag bits are required to identify a memory block when it is resident in the cache. The tag bits of an address received from the processor are compared to the tag bits of each block of the cache to see if the desired block is present. This is called the associative-mapping technique. It gives complete freedom in choosing the cache location in which to place the memory block, resulting in a more efficient use of the space in the cache. When a new block is brought into the cache, it replaces (ejects) an existing block only if the cache is full. In this case, we need an algorithm to select the block to be replaced.

Many replacement algorithms are possible. The complexity of an associative cache is higher than that of a direct-mapped cache, because of the need to search all 128 tag patterns to determine whether a given block is in the cache. To avoid a long delay, the tags must be searched in parallel. A search of this kind is called an associative search.

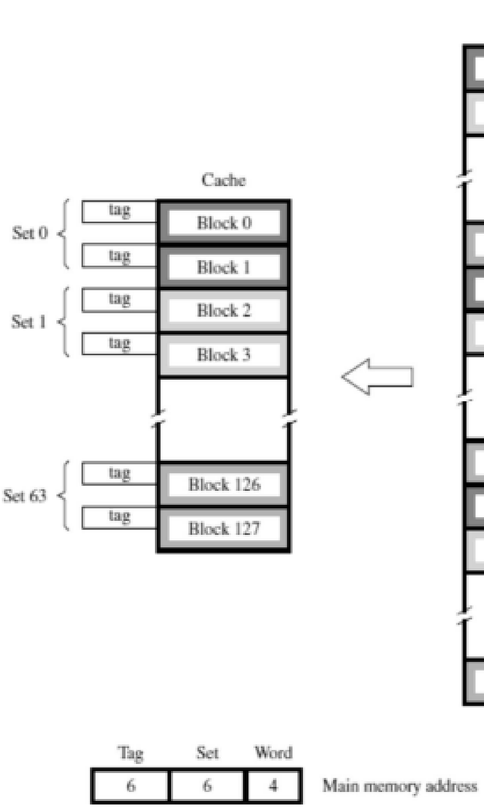


Set-Associative Mapping

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

At the same time, the hardware cost is reduced by decreasing the size of the associative search. An example of this set-associative-mapping technique is shown in Figure 8.18 for a cache with two blocks per set. In this case, memory blocks 0, 64, 128, . . . , 4032 map into cache set 0, and they can occupy either of the two block positions within this set. Having 64 sets means that the 6-bit set field of the address determines which set of the cache might contain the desired block.

The tag field of the address must then be associatively compared to the tags of the two blocks of the set to check if the desired block is present. This two-way associative search is simple to implement. The number

		of blocks per set is a parameter that can be selected to suit the requirements of a particular computer. For the main memory and cache sizes in Figure, four blocks per set can be accommodated by a 5-bit set field, eight blocks per set by a 4-bit set field, and so on. The extreme condition of 128 blocks per set requires no set bits and corresponds to the fully- associative technique, with 12 tag bits. The other extreme of one block per set is the directmapping method. A cache that has k blocks per set is referred to as a k-way set-associative cache.  Main memory Block 0 Block 1 Block 63 Block 64 Block 65 Block 127 Block 128 Block 129 Block 4095 Cache Set 0 Set 1 Set 63 tag Block 0 Block 1 Block 2 Block 3 Block 126 Block 127 Tag Set Word 6 6 4 Main memory address			
7.	(a)	Discuss about Read Only Memories (ROM) Both static and dynamic RAM chips are volatile, which means that they retain information only while power is turned on. There are many applications requiring memory devices that retain the stored information when power is turned off. For example, the need to store a small program in such a memory, to be used to start the bootstrap process of loading the operating system from a hard disk into the main memory. The embedded applications are another important example. Many embedded applications do not use a hard disk and require non-volatile memories to store their software. Different types of non-volatile memories have been developed. Generally, their contents can be read in the same way as for their volatile counterparts discussed above. But, a special writing process is needed to place the information into a non-volatile memory. Since its normal operation involves only reading the stored data, a memory of this type is called a read-only memory (ROM). **ROM**	13	CO4	UND

A memory is called a read-only memory, or ROM, when information can be written into it only once at the time of manufacture. Figure 8.11 shows a possible configuration for a ROM cell. A logic value 0 is stored in the cell if the transistor is connected to ground at point P; otherwise, a 1 is stored. The bit line is connected through a resistor to the power supply. To read the state of the cell, the word line is activated to close the transistor switch.

As a result, the voltage on the bit line drops to near zero if there is a connection between the transistor and ground. If there is no connection to ground, the bit line remains at the high voltage level, indicating a 1. A sense circuit at the end of the bit line generates the proper output value. The state of the connection to ground in each cell is determined when the chip is manufactured, using a mask with a pattern that represents the information to be stored.

PROM

Some ROM designs allow the data to be loaded by the user, thus providing a programmable ROM (PROM). Programmability is achieved by inserting a fuse at point P, in Figure. Before it is programmed, the memory contains all 0s. The user can insert 1s at the required locations by burning out the fuses at these locations using high-current pulses. Of course, this process is irreversible. PROMs provide flexibility and convenience not available with ROMs. The cost of preparing the masks needed for storing a particular information pattern makes ROMs cost effective only in large volumes. The alternative technology of PROMs provides a more convenient and considerably less expensive approach, because memory chips can be programmed directly by the user.

EPROM

Another type of ROM chip provides an even higher level of convenience. It allows the stored data to be erased and new data to be written into it. Such an erasable, reprogrammable ROM is usually called an EPROM. It provides considerable flexibility during the development phase of digital systems. Since EPROMs are capable of retaining stored information for a long time, they can be used in place of ROMs or PROMs while software is being developed. In this way, memory changes and updates can be easily made. An EPROM cell has a structure similar to the ROM cell in Figure. However, the connection to ground at point P is made through a special transistor. The transistor is normally turned off, creating an open switch. It can be turned on by injecting charge into it that becomes trapped inside. Thus, an EPROM cell can be used to construct a memory in the same way as the previously discussed ROM cell. Erasure requires dissipating the charge trapped in the transistors that form the memory cells. This can be done by exposing the chip to ultraviolet light, which erases the entire contents of the chip. To make this possible, EPROM chips are mounted in packages that have transparent windows.

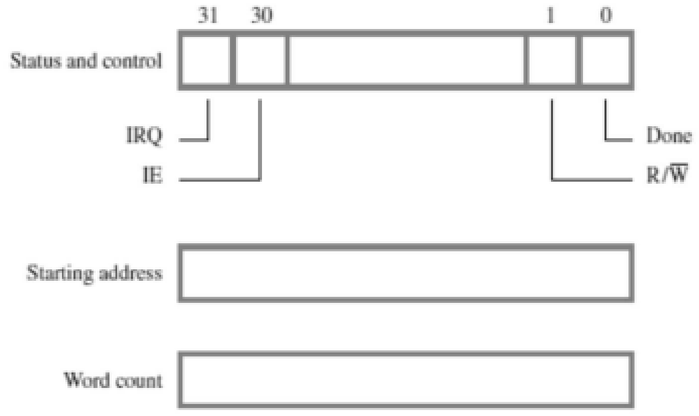
EEPROM

		An EPROM must be physically removed from the circuit for reprogramming. Also, the stored information cannot be erased selectively. The entire contents of the chip are erased when exposed to ultraviolet light. Another type of erasable PROM can be programmed, erased, and reprogrammed electrically. Such a chip is called an electrically erasable PROM, or EEPROM. It does not have to be removed for erasure. Moreover, it is possible to erase the cell contents selectively. One disadvantage of EEPROMs is that different voltages are needed for erasing, writing, and reading the stored data, which increases circuit complexity. However, this disadvantage is outweighed by the many advantages of EEPROMs. They have replaced EPROMs in practice. **Flash Memory** An approach similar to EEPROM technology has given rise to flash memory devices. A flash cell is based on a single transistor controlled by trapped charge, much like an EEPROM cell. Also like an EEPROM, it is possible to read the contents of a single cell. The key difference is that, in a flash device, it is only possible to write an entire block of cells. Prior to writing, the previous contents of the block are erased. Flash devices have greater density, which leads to higher capacity and a lower cost per bit. They require a single power supply voltage, and consume less power in their operation. The low power consumption of flash memories makes them attractive for use in portable, battery-powered equipment. Typical applications include hand-held computers, cell phones, digital cameras, and MP3 music players. In hand-held computers and cell phones, a flash memory holds the software needed to operate the equipment, thus obviating the need for a disk drive. A flash memory is used in digital cameras to store picture data. In MP3 players, flash memories store the data that represent sound. Cell phones, digital cameras, and MP3 players are good examples of embedded systems. Single flash chips may not provide sufficient storage capacity for the applications. Larger memory modules consisting of a number of chips are used where needed. There are two popular choices for the implementation of such modules: flash cards and flash drive			
(OR)					
	(b)	Discuss DMA controller with block diagram. Blocks of data are often transferred between the main memory and I/O devices such as disks. This section discusses a technique for controlling such transfers without frequent, program-controlled intervention by the processor. The discussion concentrates on single-word or single-byte data transfers between the processor and I/O devices. Data are transferred from an I/O device to the memory by first reading them from the I/O device using an instruction such as Load R2, DATAIN which loads the data into a processor register. Then, the data read are stored into a memory location. The reverse process takes place for transferring data from the memory to an I/O device. An instruction to transfer input or output data is executed only after the processor determines that the I/O device is ready, either by polling its status register or by waiting for an interrupt request. In either case, considerable overhead is incurred, because several program	13	CO5	REM

instructions must be executed involving many memory accesses for each data word transferred. When transferring a block of data, instructions are needed to increment the memory address and keep track of the word count. The use of interrupts involves operating system routines which incur additional overhead to save and restore processor registers, the program counter, and other state information.

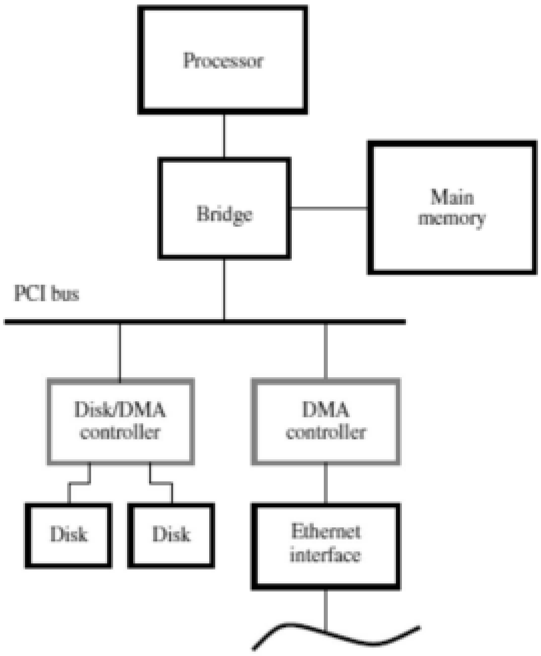
An alternative approach is used to transfer blocks of data directly between the main memory and I/O devices, such as disks. A special control unit is provided to manage the transfer, without continuous intervention by the processor. This approach is called direct memory access, or DMA. The unit that controls DMA transfers is referred to as a DMA controller. It may be part of the I/O device interface, or it may be a separate unit shared by a number of I/O devices. The DMA controller performs the functions that would normally be carried out by the processor when accessing the main memory. For each word transferred, it provides the memory address and generates all the control signals needed. It increments the memory address for successive words and keeps track of the number of transfers.

Although a DMA controller transfers data without intervention by the processor, its operation must be under the control of a program executed by the processor, usually an operating system routine. To initiate the transfer of a block of words, the processor sends to the DMA controller the starting address, the number of words in the block, and the direction of the transfer. The DMA controller then proceeds to perform the requested operation. When the entire block has been transferred, it informs the processor by raising an interrupt. Figure shows an example of the DMA controller registers that are accessed by the processor to initiate data transfer operations. Two registers are used for storing the starting address and the word count. The third register contains status and control flags. The R/W bit determines the direction of the transfer. When this bit is set to 1 by a program instruction, the controller performs a Read operation, that is, it transfers data from the memory to the I/O device. Otherwise, it performs a Write operation. Additional information is also transferred as may be required by the I/O device. For example, in the case of a disk, the processor provides the disk controller with information to identify where the data is located on the disk.



When the controller has completed transferring a block of data and is ready to receive another command, it sets the Done flag to 1. Bit 30 is the Interrupt-enable flag, IE. When this flag is set to 1, it causes the controller to raise an interrupt after it has completed transferring a block of data. Finally, the controller sets the IRQ bit to 1 when it has requested an interrupt. Figure shows how DMA controllers may be used in a computer system such as that in Figure. One DMA controller connects a high-speed Ethernet to the computer’s I/O bus (a PCI bus in the case of Figure). 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.

To start a DMA transfer of a block of data from the main memory to one of the disks, an OS 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.



Bus Arbitration

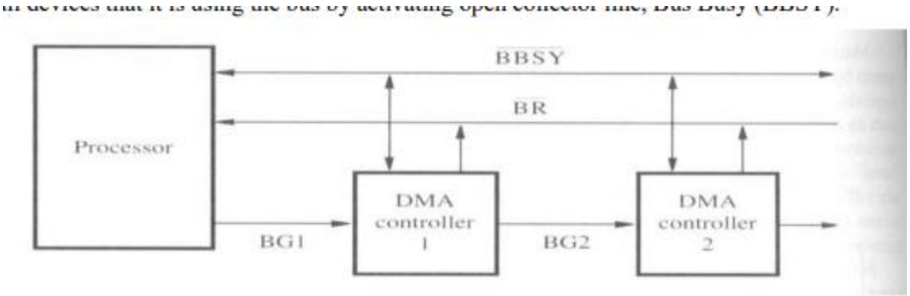
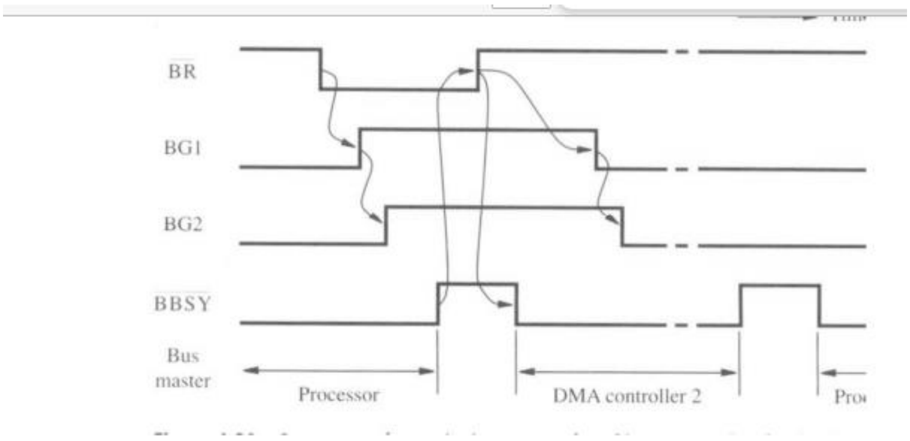
It is the process by which the next device to become the bus master is selected and the bus mastership is transferred to it. Types:

There are 2 approaches to bus arbitration.

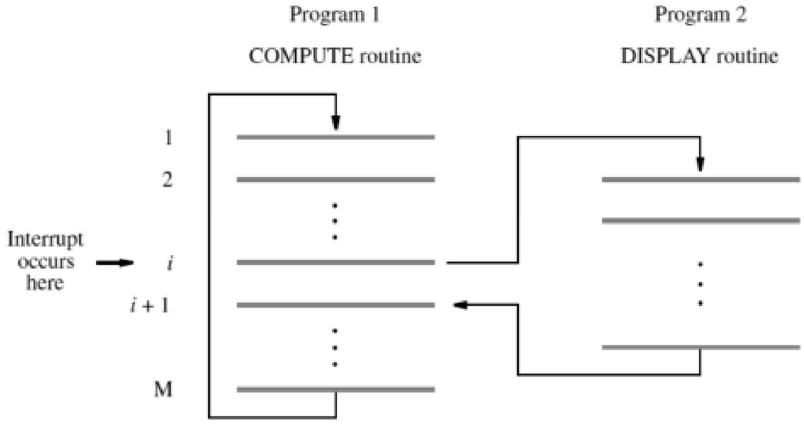
They are, Centralized arbitration (A single bus arbiter performs arbitration) Distributed arbitration (all devices participate in the selection of next bus master).

Centralized Arbitration:

Here the processor is the bus master and it may grants bus mastership to one of its DMA controller. A DMA controller indicates that it needs

		to become the bus master by activating the Bus Request line (BR) which is an open drain line. The signal on BR is the logical OR of the bus request from all devices connected to it. When BR is activated the processor activates the Bus Grant Signal (BGI) and indicated the DMA controller that they may use the bus when it becomes free. This signal is connected to all devices using a daisy chain arrangement. If DMA requests the bus, it blocks the propagation of Grant Signal to other devices and it indicates to all devices that it is using the bus by activating open collector line, Bus Busy (BBSY).  BBSY BR Processor DMA controller 1 DMA controller 2 BG1 BG2 The timing diagram shows the sequence of events for the devices connected to the processor is shown. DMA controller 2 requests and acquires bus mastership and later releases the bus. During its tenure as bus master, it may perform one or more data transfer. After it releases the bus, the processor resources bus mastership  BR BG1 BG2 BBSY Bus master Processor DMA controller 2 Pro			

8.	(a)	Explain about interrupts. In the programmed I/O transfer, the program enters a wait loop in which it repeatedly tests the device status. During this period, the processor is not performing any useful computation. There are many situations where other tasks can be performed while waiting for an I/O device to become ready. To allow this to happen, we can arrange for the I/O device to alert the processor when it becomes ready. It can do so by sending a hardware signal called an interrupt request to the processor. Since the processor is no longer required to continuously poll the status of I/O devices, it can use the waiting period to perform other useful tasks. Indeed, by using interrupts, such waiting periods can ideally be eliminated. The routine executed in response to an interrupt request is called the interrupt-service routine, which is the DISPLAY routine in our example. Interrupts bear considerable resemblance to subroutine calls.	14	CO5	UND



Assume that an interrupt request arrives during execution of instruction i in Figure. The processor first completes execution of instruction i. Then, it loads the program counter with the address of the first instruction of the interrupt-service routine. For the time being, let us assume that this address is hardwired in the processor. After execution of the interrupt-service routine, the processor returns to instruction i + 1. Therefore, when an interrupt occurs, the current contents of the PC, which point to instruction i + 1, must be put in temporary storage in a known location. A Return-from-interrupt instruction at the end of the interrupt-service routine reloads the PC from that temporary storage location, causing execution to resume at instruction i + 1. The return address must be saved either in a designated general-purpose register or on the processor stack.

We should note that as part of handling interrupts, the processor must inform the device that its request has been recognized so that it may remove its interrupt-request signal. This can be accomplished by means of a special control signal, called interrupt acknowledge, which is sent to the device through the interconnection network. An alternative is to have the transfer of data between the processor and the I/O device interface accomplish the same purpose. The execution of an instruction in the interrupt-service routine that accesses the status or data register in the device interface implicitly informs the device that its interrupt request has been recognized.

So far, treatment of an interrupt-service routine is very similar to that of a subroutine. An important departure from this similarity should be noted. A subroutine performs a function required by the program from which it is called. As such, potential changes to status information and contents of registers are anticipated. However, an interrupt-service routine may not have any relation to the portion of the program being executed at the time the interrupt request is received. Therefore, before starting execution of the interrupt service routine, status information and contents of processor registers that may be altered in unanticipated ways during the execution of that routine must be saved. This saved information must be restored before execution of the interrupted program is resumed. In this way, the original program can continue execution without being affected in any way by the interruption, except for the time delay.

The task of saving and restoring information can be done automatically by the processor or by program instructions. Most modern processors save only the minimum amount of information needed to maintain the integrity of program execution. This is because the process of saving and restoring registers involves memory transfers that increase the total execution time, and hence represent execution overhead. Saving registers also increases the delay between the time an interrupt request is received and the start of execution of the interruptservice routine. This delay is called interrupt latency. In some applications, a long interrupt latency is unacceptable. For these reasons, the amount of information saved automatically by the processor when an interrupt request is accepted should be kept to a minimum. Typically, the processor saves only the contents of the program counter and the processor status register. Any additional information that needs to be saved must be saved by explicit instructions at the beginning of the interrupt-service routine and restored at the end of the routine. In some earlier processors, particularly those with a small number of registers, all registers are saved automatically by the processor hardware at the time an interrupt request is accepted. The data saved are restored to their respective registers as part of the execution of the Return-frominterrupt instruction.

Some computers provide two types of interrupts. One saves all register contents, and the other does not. A particular I/O device may use either type, depending upon its response time requirements. Another interesting approach is to provide duplicate sets of processor registers. In this case, a different set of registers can be used by the interrupt-service routine, thus eliminating the need to save and restore registers. The duplicate registers are sometimes called the shadow registers. An interrupt is more than a simple mechanism for coordinating I/O transfers. In a general sense, interrupts enable transfer of control from one program to another to be initiated by an event external to the computer. Execution of the interrupted program resumes after the execution of the interrupt-service routine has been completed. The concept of interrupts is used in operating systems and in many control applications where processing of certain routines must be accurately timed relative to external events. The latter type of application is referred to as real-time processing.

Enabling and Disabling Interrupts

The facilities provided in a computer must give the programmer complete control over the events that take place during program execution. The arrival of an interrupt request from an external device causes the processor to suspend the execution of one program and start the execution of another. Because interrupts can arrive at any time, they may alter the sequence of events from that envisaged by the programmer. Hence, the interruption of program execution must be carefully controlled. A fundamental facility found in all computers is the ability to enable and disable such interruptions as desired.

There are many situations in which the processor should ignore interrupt requests. For instance, the timer circuit should raise interrupt requests only when the COMPUTE routine is being executed. It should be prevented from doing so when some other task is being performed. In another case, it may be necessary to guarantee that a particular sequence of instructions is executed to the end without interruption because the interrupt-service routine may change some of the data used by the instructions in question. For these reasons, some means for enabling and disabling interrupts must be available to the programmer.

It is convenient to be able to enable and disable interrupts at both the processor and I/O device ends. The processor can either accept or ignore interrupt requests. An I/O device can either be allowed to raise interrupt requests or prevented from doing so. A commonly used mechanism to achieve this is to use some control bits in registers that can be accessed by program instructions. The processor has a status register (PS), which contains information about its current state of

		operation. Let one bit, IE, of this register be assigned for enabling/disabling interrupts. Then, the programmer can set or clear IE to cause the desired action. When IE = 1, interrupt requests from I/O devices are accepted and serviced by the processor. When IE = 0, the processor simply ignores all interrupt requests from I/O devices. The interface of an I/O device includes a control register that contains the information that governs the mode of operation of the device. One bit in this register may be dedicated to interrupt control. The I/O device is allowed to raise interrupt requests only when this bit is set to 1.			
	(b)	Summarize in detail about synchronous and asynchronous bus The I/O interface of a device consists of the circuitry needed to connect that device to the bus. On one side of the interface are the bus lines for address, data, and control. On the other side are the connections needed to transfer data between the interface and the I/O device. This side is called a port, and it can be either a parallel or a serial port. A parallel port transfers multiple bits of data simultaneously to or from the device. A serial port sends and receives data one bit at a time. Communication with the processor is the same for both formats; the conversion from a parallel to a serial format and vice versa takes place inside the interface circuit. With double buffering, the transfer of the second character can begin as soon as the first character is loaded from the shift register into the DAT IN register. Thus, provided the processor reads the contents of DATAIN before the serial transfer of the second character is completed, the interface can receive a continuous stream of input data over the serial line. An analogous situation occurs in the output path of the interface. During serial transmission, the receiver needs to know when to shift each bit into its input shift register. Since there is no separate line to carry a clock signal from the transmitter to the receiver, the timing information needed must be embedded into the transmitted data using an encoding scheme. There are two basic approaches. The first is known as asynchronous transmission, because the receiver uses a clock that is not synchronized with the transmitter clock. In the second approach, the receiver is able to generate a clock that is synchronized with the transmitter clock. Hence it is called synchronous transmission. These approaches are described briefly below. **Asynchronous** This approach uses a technique called start-stop transmission. Data are organized in small groups of 6 to 8 bits, with a well-defined beginning and end. In a typical arrangement, alphanumeric characters encoded in 8 bits are transmitted as shown in Figure. The line connecting the transmitter and the receiver is in the 1 state when idle. A character is transmitted as a 0 bit, referred to as the Start bit, followed by 8 data bits and 1 or 2 Stop bits. The Stop bits have a logic value of 1. The 1-to-0 transition at the beginning of the Start bit alerts the receiver that data transmission is about to begin. Using its own clock, the receiver determines the position of the next 8 bits, which it loads into its input register. The Stop bits following the transmitted character, which are equal to 1, ensure that the Start bit of the next character will be recognized. Wher transmission stops, the line remains in the 1 state until another characte is transmitted. To ensure correct reception, the receiver needs to sample	14	CO5	UND

		the incoming data as close to the center of each bit as possible. It does so by using a clock signal whose frequency, fR, is substantially higher than the transmission clock, fT . Typically, fR = 16fT . This means that 16 pulses of the local clock occur during each data bit interval. This clock is used to increment a modulo-16 counter, which is cleared to 0 when the leading edge of a Start bit is detected. The middle of the Start bit is reached at the count of 8. The state of the input line is sampled again at this point to confirm that it is a valid Start bit (a zero), and the counter is cleared to 0. From this point onward, the incoming data signal is sampled whenever the count reaches 16, which should be close to the middle of each incoming bit. Therefore, as long as fR/16 is sufficiently close to fT , the receiver will correctly load the bits of the incoming character. Synchronous Transmission In the start-stop scheme described above, the position of the 1-to-0 transition at the beginning of the start bit in Figure is the key to obtaining correct timing information. This scheme is useful only where the speed of transmission is sufficiently low and the conditions on the transmission link are such that the square waveforms shown in the figure maintain their shape. For higher speed a more reliable method is needed for the receiver to recover the timing information. In synchronous transmission, the receiver generates a clock that is synchronized to that of the transmitter by observing successive 1-to-0 and 0-to-1 transitions in the received signal. It adjusts the position of the active edge of the clock to be in the center of the bit position. A variety of encoding schemes are used to ensure that enough signal transitions occur to enable the receiver to generate a synchronized clock and to maintain synchronization. Once synchronization is achieved, data transmission can continue indefinitely. Encoded data are usually transmitted in large blocks consisting of several hundreds or several thousands of bits. The beginning and end of each block are marked by appropriate codes, and data within a block are organized according to an agreed upon set of rules. Synchronous transmission enables very high data transfer rates .			

Prepared By **Verified By** **HOD**