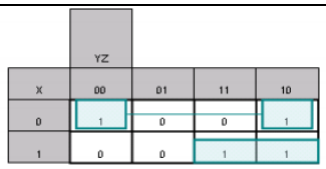
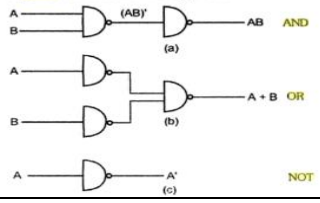
	E.G.S. PILLAY ENGINEERING COLLEGE (An Autonomous Institution, Affiliated to Anna University, Chennai) Nagore Post, Nagapattinam – 611 002, Tamilnadu.	Rev.0 COE/2017/QB

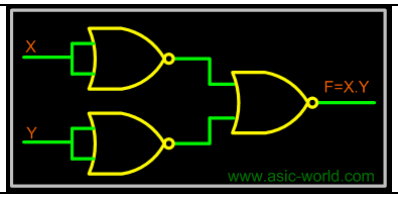
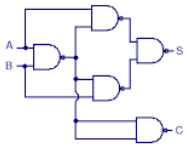
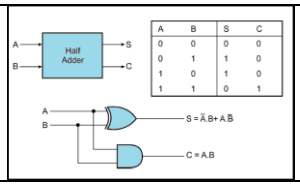
17CA105 COMPUTER ORGANIZATION AND DESIGN				
Academic Year :	2018-19	**Question Bank**	Programme :	P.G – MCA
Year / Semester :	I / I		Course Coordinator :	Mr. S.Selvaganapathy AP/MCA

Course Objectives	**Course Outcomes:**
1. To understand the fundamentals of Boolean logic and functions. 2. To have a thorough understanding of the basic structure and operation of a digital computer. 3. To design and realize digital systems with basic gates and other components using combinational and sequential circuits 4. To discuss in detail about the operation of the arithmetic and logic unit. 5. To study the instruction sets and operation of a processor. 6. To study the different ways of communication with I/O devices and standard I/O Interfaces. 7. To study the hierarchical memory system including cache memories and virtual memory	On the successful completion of the course, students will be able to CO1: Master the binary and hexadecimal number systems including computer arithmetic. CO2: Design and implement digital systems with basic gates and other components using combinational and sequential circuits CO3: Familiarize the Von Neumann architecture. CO4: Familiarize the functional units of the processor and addressing modes, instruction sets. CO5: Familiarize the memories and cache subsystem.

PART – A (2 Mark Questions With Key)				
S.No	**Questions**	**Mark**	**COs**	**BTL**
UNIT I – DIGITAL FUNDAMENTALS				
1	Define Computer Architecture		1	K1
	Computer Architecture Is Defined As The Functional Operation Of The Individual H/W Unit In A Computer System And The Flow Of Information Among The Control Of Those Units	2		
2	List the number systems?	2	1	K1
	i) Decimal Number system ii) Binary Number system iii) Octal Number system iv) Hexadecimal Number system			
3	What is the binary equivalent of the decimal number 368	2	1	K2

	2 | 368 2 | 184 --- 0 2 | 92 --- 0 2 | 46 --- 0 2 | 23 --- 0 2 | 11 --- 1 2 | 5 --- 1 2 | 2 --- 1 2 | 1 --- 0 0 --- 1 Decimal number 368 is 101110000			
4	The simplification of the Boolean expression $\left(\overline{\overline{A}B\overline{C}}\right)+\left(\overline{A\overline{B}C}\right)$ is	2	1	K3
	$\left(\overline{\overline{A}B\overline{C}}\right)+\left(\overline{A\overline{B}C}\right) = \overline{\overline{A}}+\overline{B}+\overline{\overline{C}}+\overline{A}+\overline{\overline{B}}+\overline{C} = A+\overline{B}+C+\overline{A}+B+\overline{C}$ $= (A+\overline{A})(B+\overline{B})(C+\overline{C}) = 1X1X1 = 1$			
5	Convert the octal number 7401 to Binary.		1	K2
	Conversion of Octal number 7401 to Binary: Each octal digit represents 3 binary digits. To convert an octal number to binary number, each octal digit is replaced by its 3 digit binary equivalent shown below	1		
	7 4 0 1 ↓ ↓ ↓ ↓ 111 100 000 001 Thus, $(7401)_8 = (111100000001)_2$	1		
6	Perform 2's complement subtraction of $(7)_{10}-(11)_{10}$.	2	1	K2
	2's Complements Subtraction of $(7)_{10}-(11)_{10}$ First convert the decimal numbers 7 and 11 to its binary equivalents. $(7)_{10} = (0111)_2$ $(11)_{10} = (1011)_2$ in 4-bit system Then find out the 2's complement for 1011 i.e., 1's Complement of 1011 is 0100 2's Complement of 1011 is 0101 So, $(7)_{10}-(11)_{10}$ = 0111 0101 --- 1100 ---			
7	Simplify the following expression Y = (A + B) (A + C') (B' + C')	2	1	K3
	Y = (A + B) (A + C') (B' + C') = (AA' + AC +A'B +BC) (B' + C') [A.A' = 0] = (AC + A'B + BC) (B' + C') = AB'C + ACC' + A'BB' + A'BC' + BB'C + BCC' = AB'C + A'BC'			
8	Prove that ABC + ABC' + AB'C + A'BC = AB + AC + BC	2	1	K3
	ABC + ABC' + AB'C + A'BC=AB(C + C') + AB'C + A'BC =AB + AB'C + A'BC =A(B + B'C) + A'BC =A(B + C) + A'BC =AB + AC + A'BC =B(A + C) + AC =AB + BC + AC =AB + AC +BC ...Proved			

9	State the limitations of karnaugh map.		1	K1
	i) Generally it is limited to six variable map (i.e) more then six variable involving expression are not reduced.	1		
	ii) The map method is restricted in its capability since they are useful for simplifying only Boolean expression represented in standard form	1		
10	Convert (B65F)16 to base 10	2	1	K2
	$= 11 \times 16^3 + 6 \times 16^2 + 5 \times 16^1 + 15 \times 16^0$ $= 11 \times 4096 + 6 \times 256 + 5 \times 16 + 15$ $= 45056 + 1536 + 80 + 15$ $= (46687)_{10}$			
11	Simplify the Boolean function $F(x,y,z) = \Sigma(0,2,6,7)$ using three variable maps.	2	1	K3
	 F= XY + X'Z'			
12	Determine the value of base x if $(193)x = (623)_8$.	2	1	K2
	Solution : i) $(193)_x = (623)_8$ Converting octal into decimal $6\times8^2+2\times8+3 = (403)$ $(623)_8 = (403)_{10}$ $\therefore (193)_x = (403)_{10}$ $1\times x^2+9\times x+3\times x^0 = 403$ $x^2+9x+3 = 403$ $\therefore x = 16$ or $x = -25$			
13	Implement AND, OR and NOT gates using NAND gates	2	1	K3
	The realization of above gates are shown in Fig. 3.85.  			
14	***Why are NAND and NOR gates known as Universal gates ?***	2	1	K2
	Ans. : NAND and NOR are the gates that can be used alone to generate remaining gates such as NOT, AND and OR. Thus, with only any of the two gates, we can implement the logic circuit. Hence, they are called Universal gates.			
15	Determine a AND gate using NOR gates only		1	K2

		2		
UNIT II – COMBINATIONAL AND SEQUENTIAL CIRCUITS				
1	Distinguish between combinational circuit and sequential circuit.	**2**	2	K2
	combinational circuit 1. The circuit whose output at any instant depends only on the input present at that instant only is known as combinationational circuit . 2. This type of circuit has no memory unit . 3. Examples of combinational circuits are half adder, full adder, magnitude comparator, multiplexer, demultiplexe r e.t.c. sequential circuit 1. The circuit whose output at any instant depends not only on the input present but also on the past output a is known as sequental circuit 2. This type of circuit has memory unit for store past output 3. Examples of sequential circuits are Flip flop, register, counter e.t.c.			
2	Design a half adder using NAND gates.	**2**	2	K3
	 Half adder using NAND logic			
3	Construct the truth table of half adder	**2**	2	K2
	 A B S C 0 0 0 0 0 1 1 0 1 0 1 0 1 1 0 1			
4	**What are the different types of flip-flop?**	**2**	2	K2
	There are various types of flip flops. Some of them are mentioned below they are, (1) RS flip-flop (2) SR flip-flop (3) D flip-flop (4) JK flip-flop (5) T flip-flop			
5	**What is the operation of JK flip-flop?**	**2**	2	K2
	When K input is low and J input is high the Q output of flip-flop is set. When K input is high and J input is low the Q output of flip-flop is reset. When both the inputs K and J are low the output does not change When both the inputs K and J are high it is possible to set or reset the Flip-flop (ie) the output toggle on the next positive clock edge.			
6	What do you think about clock generator?	**2**	2	K2
	A **clock generator** is a circuit that produces a timing signal (known as a clock signal and behaves as such) for use in synchronizing a circuit's operation. The signal can range from a simple symmetrical square wave to more complex arrangements. The basic parts that all clock generators share are a resonant circuit and an amplifier.			
7	What is the difference between a truth table and a state table?		2	K2
	A **state table** is essentially a truth **table** in which some of the inputs are the current **state**, and the outputs include the next **state**, along with other outputs. A **state table**is one o f many ways to specify a **state** machine, other	**1**		

	ways being a **state** diagram, and a characteristic equation.			
	A **truth table** is a breakdown of a logic function by listing all possible values the function can attain. Such a **table** typically contains several rows and columns, with the top row representing the logical variables and combinations, in increasing complexity leading up to the final function.	**1**		
8	**Difference between Asynchronous and Synchronous Counter :**	**2**	2	K2
	<table><tr><th>Asynchronous Counter</th><th>Synchronous Counter</th></tr><tr><td>1. Clock input is applied to LSB FF. The output of first FF is connected as clock to next FF.</td><td>1. Clock input is common to all FF.</td></tr><tr><td>2. All Flip-Flops are toggle FF.</td><td>2. Any FF can be used.</td></tr><tr><td>3. Speed depends on no. of FF used for n bit . $f_{max} = \frac{1}{n \times t_p}$</td><td>3. Speed is independent of no. of FF used. $f_{max} = \frac{1}{t_p}$</td></tr><tr><td>4. No extra Logic Gates are required.</td><td>4. Logic Gates are required based on design.</td></tr><tr><td>5. Cost is less.</td><td>5. Cost is more.</td></tr></table>			
9	What is triggering in flip-flop?	**2**	2	K1
	The state of a **flip-flop** is changed by a momentary change in the input signal. This change is called a **trigger** and the transition it causes is said to **trigger** the **flip-flop**. The basic circuits of Figure 2 and Figure 3 require an input **trigger** defined by a change in signal level.			
10	Distinguish between multiplexer and de multiplexer.	**2**	2	K2
	Multiplexer- 1. Many inputs & one output. 2. Data select lines. 3. Parallel to serial conversion. 4. When we design multiplexer, we don't need additional gates. 5. Example- 8:1, 16:1, 32:1 Demultiplexer- 1. One inputs & many output. 2. Data distributer. 3. Serial to parallel conversion. 4. When we design demultiplexer, we need additional gates. 5. Example- 1:8, 1:16, 1:32			
11	Distinguish between encoder and decoder.	**2**	2	K2
	difference between decoder and **encoder**. A **decoder** is a multiply-input, multiply-output combinational logic circuit that converts coded inputs into coded outputs, where the input and output codes are **different**. The most commonly used input code is an n-bit binary code			
12	What are the types of ALU design?	**2**	2	K1
	An **arithmetic logic unit** (**ALU**) is a digital electronic circuit that performs arithmetic and bitwise logical operations on integer binary numbers. ... It is a fundamental building block of many **types** of computing circuits, including the central processing unit (CPU) of computers, FPUs, and graphics processing units.			
13	What is the control unit?	**2**	2	K1
	The control unit (CU) is a component of a computer's central processing unit(CPU) that directs the operation of the processor. It tells the computer's memory, arithmetic/logic unit and input and output devices on how to respond to a program's instructions.			

14	What is the function of a control unit?	**2**	2	K1
	The Control Unit (CU) is digital circuitry contained within the processor that coordinates the sequence of data movements into, out of, and between a processor's many sub-units. The result of these routed data movements through various digital circuits (sub-units) within the processor produces the manipulated data expected by a software instruction (loaded earlier, likely from memory). It controls (conducts) data flow inside the processor and additionally provides several external control signals to the rest of the computer to further direct data and instructions to/from processor external destination's			
15	What do you think about clock generator?	**2**	2	K1
	A **clock generator** is a circuit that produces a timing signal (known as a clock signal and behaves as such) for use in synchronizing a circuit's operation. The signal can range from a simple symmetrical square wave to more complex arrangements. The basic parts that all clock generators share are a resonant circuit and an amplifier			
UNIT III – PROCESSOR FUNDAMENTALS				
1	What is meant by von Neumann architecture?	**2**	3	K1
	The **von Neumann architecture**, which is also known as the **von Neumann** model and Princeton **architecture**, is a computer **architecture** based on that described in 1945 by the mathematician and physicist John **von Neumann** and others in the First Draft of a Report on the EDVAC.			
2	**Define Processor.**	**2**	3	K1
	A processor, or "microprocessor," is a small chip that resides in computers and other electronic devices. Its basic job is to receive input and provide the appropriate output. While this may seem like a simple task, modern processors can handle trillions of calculations per second.			
3	What is pipelining?	**2**	3	K1
	The technique of overlapping the execution of successive instruction for substantial improvement in performance is called pipelining.			
4	Define instruction set processor.	**2**	3	K1
	IR <-[[PC]] The instruction recorder and control logic unit is responsible for implementing the actions specified by the www.vidyarthiplus.com instruction loaded in the IR register. The decoder generates the control signals needed to select the registers involved and direct the transfer of data. The registers, the ALU, and the interconnecting bus are collectively referred to as the data path			
5	**Define** Fetch execute cycle	**2**	3	K1
	An instruction **cycle** (sometimes called a **fetch**–decode–**execute cycle**) is the basic operational process of a computer. It is the process by which a computer retrieves a program instruction from its memory, determines what			

	actions the instruction dictates, and carries out those actions.			
6	What is CPI in computer architecture?	**2**	3	K1
	In **computer architecture**, **cycles per instruction** (aka clock **cycles per instruction**, clocks per instruction, or **CPI**) is one aspect of a **processor's**performance: the average number of clock **cycles per instruction** for a program or program fragment. It is the multiplicative inverse of instructions per cycle.			
7	**Define** Throughput and stage time		3	K1
	Throughput: Number of items (cars, instructions, operations) that exit the pipeline per unit time. Ex: 1 inst / clock cycle, 10 cars/ hour, 10 fp operations /cycle.	**1**		
	Stage time: The pipeline designer's goal is to balance the length of each pipeline stage. Balanced pipeline. In general, stage time = Time per instruction on non-pipelined machine / number of stages. In many instances, stage time = max (times for all stages)	**1**		
8	**Define Data path**	**2**	3	K1
	The data path is the "brawn" of a processor, since it implements the fetch-decode-execute cycle. The general discipline for data path design is to (1) determine the instruction classes and formats in the ISA, (2) design data path components and interconnections for each instruction class or format, and (3) compose the data path segments designed in Step 2) to yield a composite data path.			
9	**List out the data path types**	**2**	3	K1
	R-format Datapath **Load/Store Datapath** **Branch/Jump Datapath**			
10	What is meant by multi core processor?	**2**	3	K1
	By Vangie Beal In consumer technologies, **multi-core** is usually the term used to describe two or more CPUs working together on the same chip. Also called**multicore** technology, it is a type of architecture where a single physical **processor**contains the **core** logic of two or more **processors**.			
11	How does a multi core processor work?	**2**	3	K2
	CPU Basics: Multiple CPUs, Cores, and Hyper-Threading Explained. The central processing unit (**CPU**) in your computer **does** the computational **work** — running programs, basically. But one single-**core CPU** can only perform one task at a time, which is where multiple CPUs, hyper-threading, and **multi-core** CPUs come into play.			
12	What is multi core architecture?	**2**	3	K1
	Multicore refers to an **architecture** in which a single physical processor incorporates the **core** logic of more than one processor. A single integrated circuit is used to package or hold these processors. These single integrated circuits are known as a die.			

13	**Define Hazards**	**2**	3	K1
	Hazard (computer architecture) In the domain of central processing unit (CPU) design, hazards are problems with the **instruction pipeline** in CPU microarchitectures when the next instruction cannot execute in the **following clock cycle**, and can potentially **lead** to **incorrect** computation results.			
14	**What are the types of pipeline hazards?**	**2**	3	K2
	1. Pipeline hazards are situations that prevent the next instruction in the instruction stream from executing during its designated clock cycles. 2. Any condition that causes a stall in the pipeline operations can be called a hazard. 3. There are primarily three types of hazards: i. Data Hazards ii. Control Hazards or instruction Hazards iii. Structural Hazards.			
15	**Define stored program**	**2**	3	K1
	A **stored-program computer** is one that stores **program** instructions in electronic memory. Often the **definition** is extended with the requirement that the treatment of **programs** and data in memory be interchangeable or uniform.			
UNIT IV – MEMORY				
1	**Define Addressing Mode.**	**2**	4	K1
	Addressing modes are an aspect of the instruction set **architecture** in most central processing unit (CPU) designs. The various **addressing modes** that are defined in a given instruction set **architecture** define how machine language instructions in that **architecture** identify the operand(s) of each instruction.			
2	What is Virtual Memory	**2**	4	K1
	Virtual Memory: As we know that a Computer is designed for Performing the Multiple Tasks at a Time and for this Some Memory is also used by the Computer for executing the instructions those are given by the user. But when there is a Situation when the Memory (RAM) which is required by the user is high from the Available Memory. So at that situation we will use the Concept of Virtual Memory.			
3	What is the immediate addressing mode?	**2**	4	K1
	An **immediate** operand has a constant value or an expression. When an instruction with two operands uses **immediate addressing**, the first operand may be a **register**or memory location, and the second operand is an **immediate** constant. The first operand defines the length of the data.			
4	What is register indirect addressing mode?	**2**	4	K1
	Register indirect addressing means that the location of an operand is held in a**register**. It is also called indexed **addressing** or base **addressing**. **Register indirect addressing mode** requires three read operations to access an operand.			
5	List out the Types Of Addressing Modes	**2**	4	K1

	Following are the types of Addressing Modes: Register Addressing Mode Direct Addressing Mode Register Indirect Addressing Mode Immediate Addressing Mode Index Addressing Mode			
6	List any differences between virtual memory and physical memory?	**2**	4	K2
	Swapping – A process must be in main memory to be executed, and since the main memory is shared by many processes, a process can be swapped temporarily out of memory to a backing store (fast disk) to release memory to other process when it's idle, and then bring it back into memory for execution. The total transfer time is directly proportional to the amount of memory swapped, and as such; this process can create a performance problem. **Protection** – Since the main memory is shared by many processes, protection of memory space is required, and as such; the CPU hardware compares every address generated in the user mode with the registers. Any attempt by a program execution in the user mode to access operating-system memory or other user's memory results in a trap to the operating system, which treats the attempt as a fatal error. **Fragmentation** – Different allocation memory strategies such as first-fit and best-fit suffer from external fragmentation. As processes are loaded and removed from memory, the free memory space is broken into little pieces. External fragmentation exists when there is enough total memory space to satisfy a request but the available spaces are not contiguous. **Paging** – Is a memory-management scheme that allows the physical address space for a process to be non-contiguous. Paging avoids fragmentation and the need of compaction. Such scheme solves the problems with memory fitting of varying sizes onto the backing store where many systems were suffering from before the introduction of paging.			
7	**Define address translation**	**2**	4	K1
	Network **address translation** (NAT) is a method of remapping one IP **address** space into another by modifying network **address** information in Internet Protocol (IP) datagram packet headers while they are in transit across a traffic routing device.			
8	**Define Cache Memory**	**2**	4	K1
	The Cache Memory is the Memory which is very nearest to the CPU , all the Recent Instructions are Stored into the Cache Memory. The Cache Memory is attached for storing the input which is given by the user and which is necessary for the CPU to Perform a Task. But the Capacity of the Cache Memory is too low in compare to Memory and Hard Disk.			
9	What is the difference between l1 l2 and l3 cache?	**2**	4	K2

	A CPU **cache** is a smaller faster memory used by the central processing unit (CPU) of a computer to reduce the average time to access memory. **L1** (Level 1), **L2**, **L3 cache** are some specialized memory which work hand in hand to improve computer performance.			
10	What are the different types of mappings used in cache memory.	**2**	4	K2
	The three different types of mapping used for the purpose of cache memory are as follow, Associative mapping, Direct mapping and Set-Associative mapping.			
11	**Define direct Mapping**	**2**	4	K1
	Direct mapping: In direct mapping the RAM is made use of to store data and some is stored in the cache. An address space is split into two parts index field and tag field. The cache is used to store the tag field whereas the rest is stored in the main memory. Direct mapping`s performance is directly proportional to the Hit ratio.			
12	**Define LRU Replacement.**	**2**	4	K1
	When a page fault occurs, throw out the page that has been unused for the longest time. This strategy is called **LRU** (**Least Recently Used**) paging.			
13	What is a page of memory?	**2**	4	K1
	A **page**, **memory page**, or virtual **page** is a fixed-length contiguous block of virtual**memory**, described by a single entry in the **page** table. It is the smallest unit of data for **memory** management in a virtual **memory** operating system.			
14	**Define paging in memory management**	**2**	4	K1
	In computer operating systems, **paging** is a **memory management** scheme by which a computer stores and retrieves data from secondary storage for use in main **memory**. In this scheme, the operating system retrieves data from secondary storage in same-size blocks called pages.			
15	**Compare L1,L2,L3 Cashes**	**2**	4	K2

Cache	Bulldozer	Piledriver	Steamroller
Level 1 code	64 kB, 2-way, 64 B line size, shared between two cores.	64 kB, 2-way, 64 B line size, shared between two cores.	96 kB, 3-way, 64 B line size, shared between two cores.
Level 1 data	16 kB, 4-way, 64 B line size, per core. Latency 3-4 clocks.	16 kB, 4-way, 64 B line size, per core. Latency 3-4 clocks.	16 kB, 4-way, 64 B line size, per core. Latency 3-4 clocks.
Level 2	1 - 2 MB, 16-way, 64 B line size, shared between two cores. Latency 21 clocks. Read throughput 1 per 4 clock. Write throughput 1 per 12 clock.	2 MB, 16-way, 64 B line size, shared between two cores. Latency 20 clocks. Read throughput 1 per 4 clock. Write throughput 1 per 12 clock.	2 MB, 16-way, 64 B line size, shared between two cores. Latency 19 clocks. Read throughput 1 per 4 clock. Write throughput 1 per 6 clock.
Level 3	0 - 8 MB, 64-way, 64 B line size, shared between all cores. Latency 87 clock. Read throughput 1 per 15 clock. Write throughput 1 per 21 clock.	0 - 8 MB, 64-way, 64 B line size, shared between all cores. Latency 87 clock. Read throughput 1 per 15 clock. Write throughput 1 per 21 clock.	None

Table 14.4. Cache sizes on AMD Bulldozer, Piledriver and Steamroller

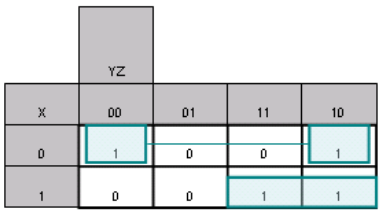
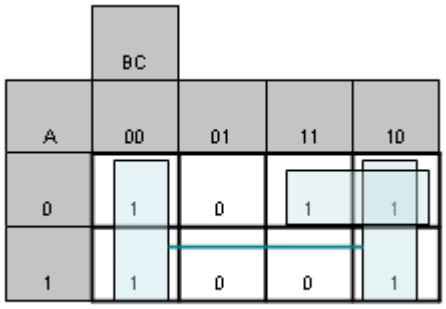
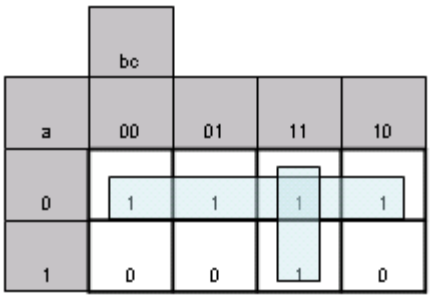
UNIT V – DATA TRANSFER				
1	**What is data transfer?**	**2**	**5**	**K1**
	Data transfer instruction move **data** from one place in the **computer** to another without changing the **data** content. The most common **transfers** are between memory and processes registers, between processes register & input or output, and between processes register themselves. (Typical **data transfer** instruction) Name.			
2	**What is Programmed** I/O?	**2**	**5**	**K1**
	Programmed input/output (PIO) is a method of transferring data between			

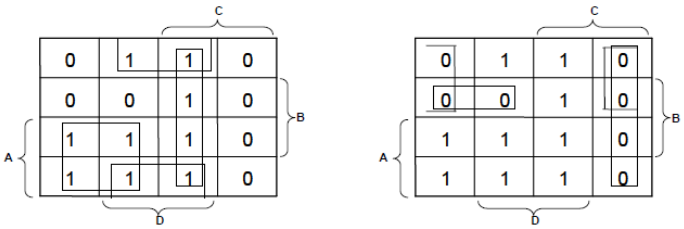
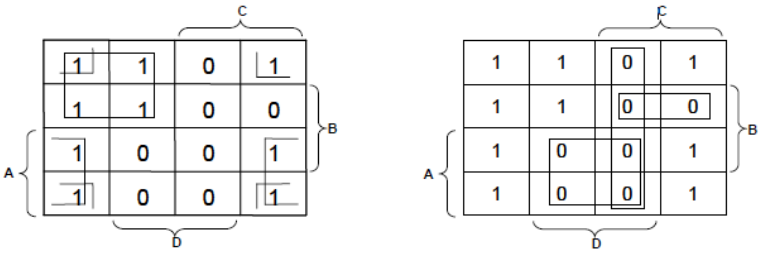
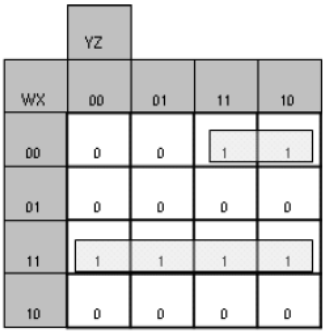
	the CPU and a peripheral, such as a network adapter or an ATA storage device.			
3	What is asynchronous data transfer explain the concept of handshaking?	**2**	**5**	**K2**
	Handshaking usually uses two additional hardware lines, one is called "strobe" and the other is called "acknowledge". The sender provides the signal to the strobe line and the receiver provides the signal to the acknowledge line. **Handshaking** can be used in both parallel **data transfer** and serial **data transfer**.			
4	**Difference Between Serial And Parallel Transfer**	**2**	**5**	**K2**
	In **Parallel transmission**: it happens across a parallel wire. Parallel wires are level & thick, comprise multiple, little cables. Each cable can take a single bit of detail. A parallel cable can take multiple bits at the similar time, one for each cable. An eight cable **parallel wire**, for instance- could take a totally byte of data. This outcome in earlier data transmission per second, entire things being similar. These devices have an **extensive** data bus than serial devices & can hence transfer data in word of one or additional bytes at a time. As **outcome**, there is an expedite in parallel transmission bit rate over serial transmission bit rate. The **serial transmission** happens over a solo cable, one bit at a time. This kind of statement is named 'serial' not purely, because the **data transmit** one bit at a time, but also since these bits should be prepared in an **exacting** way so that transmissions can be prepared & deemed dependable. It is generally **inexpensive** as only a simply channel b/w sender & receiver is necessary, ex: – the sender broadcasts the seven bits creation up an ASCII character serially in series.			
5	What is the difference between full and half duplex?	**2**	**5**	**K2**
	The Difference Between Half and **Full Duplex** Explained. "**Duplex**" simply means you're able to send and receive data (most often the human voice) from the same device whether that be with your phone, 2-way radio, or PC. **Half-duplex** devices let you send and receive, but only one-way at a time.			
6	What is full duplex and half duplex communication?	**2**	**5**	**K1**
	Full-duplex communication between two components means that both can transmit and receive information between each other simultaneously. Telephones are **full-duplex** systems so both parties on the phone can talk and listen at the same time. ... A simple illustration of a **half-duplex communication** system			
7	What is a bus interface?	**2**	**5**	**K1**
	The External **Bus Interface**, usually shortened to EBI, is a computer **bus** for interfacing small peripheral devices like flash memory with the processor. It is used to expand the internal **bus** of the processor to enable connection with external memories or other peripherals.			
8	**Define polling in Data transfer**	**2**	**5**	**K1**
	polling (1) A **communications** technique that determines when a terminal is ready to send data. The computer continually interrogates its connected terminals in a round robin sequence. If a terminal has data to send, it sends back an acknowledgment and the transmission begins.			

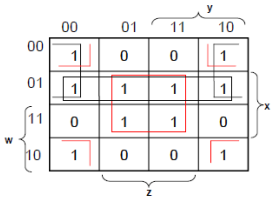
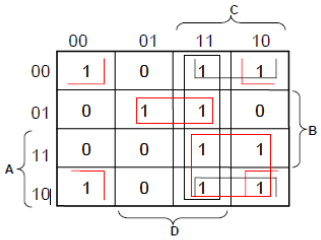
9	**What is interrupt?**	**2**	**5**	**K1**
	interrupt is a signal to the processor emitted by hardware or software indicating an event that needs immediate attention. ... Hardware **interrupts** are used by devices to communicate that they require attention from the operating system.			
10	**What is hardware Interrupt?**	**2**	**5**	**K1**
	an **interrupt** is a signal to the processor emitted by **hardware** or software indicating an event that needs immediate attention. ...**Hardware interrupts** are used by devices to communicate that they require attention from the operating system.			
11	What is the interrupt cycle?	**2**	**5**	**K1**
	An **instruction cycle** (sometimes called fetch-and-execute cycle, fetch-decode-execute cycle, or FDX) is the basic **operation cycle** of a computer. It is the process by which a computer retrieves a **program instruction** from its memory, determines what actions the instruction requires, and carries out those actions.			
12	**define interrupt vector**	**2**	**5**	**K1**
	An "**interrupt vector** table" (IVT) is a data **structure** that associates a list of **interrupt** handlers with a list of**interrupt** requests in a table of **interrupt vectors**.			
13	What is meant by vectored interrupt?	**2**	**5**	**K1**
	a **vectored interrupt** is an I/O **interrupt** that tells the part of the computer that handles I/O **interrupts** at the hardware level that a request for attention from an I/O device has been received and and also identifies the device that sent the request.			
14	What are the different types of interrupts?	**2**	**5**	**K1**
	There are two **types of interrupts**: hardware **interrupts** and software **interrupts**. Hardware **interrupts** are used by devices to communicate that they require attention from the operating system. ... The act of initiating a hardware **interrupt** is referred to as an **interrupt** request (IRQ).			
15	**What is DMA?**	**2**	**5**	**K1**
	Direct Memory Access (**DMA**) is a capability provided by some **computer** bus **architectures** that allows data to be sent directly from an attached device (such as a disk drive) to the memory on the **computer's** motherboard.			
Note : 15 Questions with answer key must be prepared in each unit				

PART – B (12 Mark Questions with Key)				
S.No	**Questions**	**Mark**	**COs**	**BTL**
UNIT I – DIGITAL FUNDAMENTALS				
1	Explain in detail about Number system and its conversion with example.	12	1	K2
	System Radix Allowable Digits Binary 2 0,1	4		

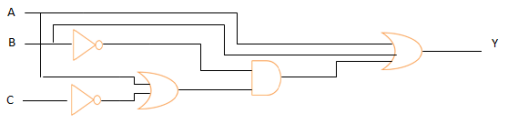
	Octal 8 0,1,2,3,4,5,6,7 Decimal 10 0,1,2,3,4,5,6,7,8,9 Hexadecimal 16 0,1,2,3,4,5,6,7,8,9,A, B, C, D, E, F - $(4021.2)_5$ = $4 \times 5^3 + 0 \times 5^2 + 2 \times 5^1 + 1 \times 5^0 + 2 \times 5^{-1} = (511.4)_{10}$ $4 \times 125 + 0 + 10 + 1 + 2 \times (1/5)$ $500 + 11 + .4$ - $(B65F)_{16}$ = $11 \times 16^3 + 6 \times 16^2 + 5 \times 16^1 + 15 \times 16^0 = (46687)_{10}$ $11 \times 4096 + 6 \times 256 + 5 \times 16 + 15$ $45056 + 1536 + 80 + 15$				
	Octal and Hexadecimal Numbers - The conversion from and to binary, octal and hexadecimal plays an important part in digital computers. Since $2^3 = 8$ and $2^4 = 16$, each octal digit corresponds to three binary digits and each hexadecimal digit corresponds to four binary digits. - Conversion from binary to Octal: $(10\ 110\ 001\ 101\ 011.\ 111\ 100\ 000\ 110)_2 = (26153.7406)_8$ - Conversion from binary to Hexadecimal: $(10\ 1100\ 0110\ 1011.\ 1111\ 0000\ 0110)_2 = (2C6B.F06)_{16}$ - Conversion from Octal to binary: $(673.124)_8 = (110\ 111\ 011.\ 001\ 010\ 100)_2$ - Conversion from Hexadecimal to binary: $(306.D)_{16} = (0011\ 0000\ 0110.\ 1101)_2$ - Conversion from Hexadecimal to Decimal: $(37B)_{16}$ $3 \times 16^2 + 7 \times 16^1 + 11 \times 16^0$ $= 3 \times 256 + 7 \times 16 + 11 \times 1$ $= 768 + 112 + 11$ $= (891)_{10}$	4			
	Number Base Conversions - A binary number can be converted to decimal by forming the sum of powers of 2 of those coefficients whose value is **1**. $(1010.011)_2 = 2^3 + 2^1 + 2^{-2} + 2^{-3} = (10.375)_{10}$ - Similarly, a number expressed in base *r* can be converted to its decimal equivalent by multiplying each coefficient with the corresponding power of *r* and adding. $(630.4)_8 = 6 \times 8^2 + 3 \times 8^1 + 0 \times 8^0 + 4 \times 8^{-1} = (408.5)_{10}$	4			
2	Simplify the following Boolean Functions, using three variable maps: (a) $F(x,y,z)=\Sigma(0,2,6,7)$ (4)	12			

	(b) F(A,B,C)=Σ(0,2,3,4,6) (4) (c) F(A,B,C,D)= Σ(4,6,7,15) (4)		1	K2
	 a. F= XY + X'Z'	4		
	 b. F= A'B + C'	4		
	 a. F=a'+bc	4		
3	Simplify the following expressions in (1) sum of products and (2) products of sums: a. $AC'+B'D+A'CD+ABCD$ (6) b. $(A'+B'+D')(A+B'+C')(A'+B+D')(B+C'+D')$ (6)	12	1	K2

	a.  SOP= $AC' + CD + B'D$ F′=CD′+A′D′+A′BC′ POS= $(C' + D)(A + D)(A + B' + C)$	6		
	b. F= $(A' + B' + D')(A + B' + C')(A' + B + D')(B + C' + D')$ F′=ABD+A′BC+AB′D+B′CD  SOP= $A'C' + B'D' + AD'$ F′= AD+ CD + A′BC POS= $(A' + D')(C' + D')(A + B' + C')$	6		
3.	a. Interpret NAND and NOR are universal gates (8) b. Simplify the following expressions F(W,X,Y,Z)= Σ(2,3,12,13,14,15). (4)	12	1	K2
	a. Any function can be implemented using only NAND or only NOR gates. How can we prove this? (Proof for NAND gates) Any boolean function can be implemented using AND, OR and NOT gates. So if AND, OR and NOT gates can be implemented using NAND gates only, then we prove our point. 1. Implement NOT using NAND 2. Implementation of AND using NAND 3. Implementation of OR using NAND	8		
	b.  F= WX + W′X′Y	4		
4	Find all the prime implicates for the following Boolean functions and determine which are essential: a) $F(w,x,y,z) = \sum(0,2,4,5,6,7,8,10,13,15)$ b) $F(A,B,C,D) = \sum(0,2,3,5,7,8,10,11,14,15)$		1	K2

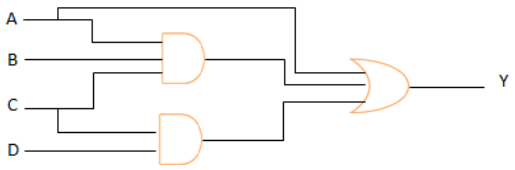
	Solution: Note: Red boxes represent the essential prime implicants. a) $F(w,x,y,z)=\sum(0,2,4,5,6,7,8,10,13,15)$  Essential: xz and x'z'; Non essential: w'x and w'z' F=xz+ x'z'+ w'z' F=xz+ x'z'+ w'x a.	6		
	 Essential prime implicants: AC, B'D' and A'BD Prime implicants: CD and B'C F= AC+B'D'+A'BD+CD F= AC+B'D'+A'BD+B'C b.	6		
5	i.find the complement of F=WX+YZ ; the show that (F.F')=0 and (F+F')=1 (3)	12	1	K2
	ii. Draw logic diagrams to implement the following Boolean expression a) Y=A+B+B'(A+C') (3) b) Y= A+CD+ABC (3) c) Y=A+B'(C+D) (3)			
	i.SOLUTION : F' = (WX+YZ)' (WX)'(YZ)' (W'+X').(Y'+Z') F . F' = (WX+YZ) . (W'+X').(Y'+Z') (WX+YZ).(W'Y'+W'Z'+X'Y'+X'Z') (WX.W'Y' +WX.W'Z'+WX.X'Y'+WX.X'Z'+YZ.W'Y'+YZ.W'Z'+YZ.X'Y'+YZ.X'Z') = 0 F+ F' : Let , (WX = A) and (YZ = B) Then , • F = A+B , • F'=(A+B)' = A' . B' , • F+F' = (A+B) + (A'.B') = (A+A'.B') +B = (A+A')(A+B') +B = (1) (A+B') +B =A+B'+B = A + 1 = 1	3		

a.



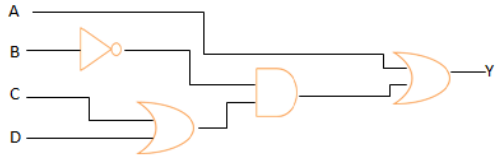
3

b.



3

c.

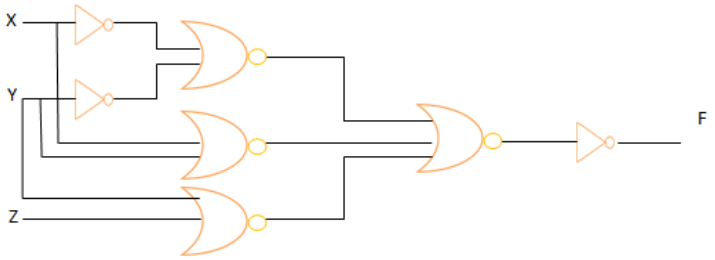


3

6 a)To use NOR & inverter we will use function equivalent to " F " function where , F = (X'+Y')' + (X+Y)' + (Y+Z)' (4) — 12 — 1 — K1

b) **Express the following function as a sum of minterms and as aproduct Of maxterms : F(A,B,C,D) = B'D +A'D +BD (8)**

a.



4

b. **F(A,B,C,D) = B'D +A'D +BD** — 8

A	B	C	D	A'	B'	B'D	A'D
0	0	0	0	1	1	0	0
0	0	0	1	1	1	1	1
0	0	1	0	1	1	0	0
0	0	1	1	1	1	1	1
0	1	0	0	1	0	0	0
0	1	0	1	1	0	0	1
0	1	1	0	1	0	0	0
0	1	1	1	1	0	0	1
1	0	0	0	0	1	0	0
1	0	0	1	0	1	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	0	0
1	1	1	0	0	0	0	0
1	1	1	1	0	0	0	0

	The sum of min terms : F (A,B,C,D) = Σ m(1,3,5,7,9,11,13,15) = A'B'C'D + A'B'CD + A'BC'D + A'BCD + AB'C'D + AB'CD + ABC'D + ABCD T he product of the max terms : F(A,B,C,D) = Π M (0,2,4,6,8,10,12,14) =(A+B+C+D).(A+B+C'+D).(A+B'+C+D).(A+B'+C'+D).(A'+B+C+D+). (A'+B+C'+D).(A'+B'+C+D).(A'+B'+C'+D)			
UNIT II – COMBINATIONAL AND SEQUENTIAL CIRCUITS				
1	(i) Explain about BCD to Seven Segment Decoder			
	1 Introduction 2 Principle of Display Decoder Circuit 3 Theory Behind the Circuit: 4 7 Segment Display Decoder Circuit Design ▪ 4.0.1 K-Map Simplification 5 Display Decoder Circuit Operation 6 Applications of Display Decoder Circuit	12	2	K2
2	(i)Basic operation of master slave D flip-flop with circuit diagram and truth table (ii)Discuss about the working of multiplexer.			
	i. Introduction to the Master Slave Design **ii. TRUTH TABLE** Input: CLK, D; Output: Q, not-Q X, 0: no change, no change X, 1: no change, no change ↑, 0: 0, 1 ↑, 1: 1, 0 iii. Analysis and Verification	6	2	K2
	Working Of Multiplexer. (ii) A **multiplexer** of 2^n inputs has n select lines, which are used to select which input line to send to the output. ... Conversely, a demultiplexer (or demux) is a device taking a single input signal and selecting one of many data-output-lines, which is connected to the single input.	6		
3	**What are combinational circuits? Give suitable block diagram.**	12	2	K2

Combinational circuit is a circuit in which we combine the different gates in the circuit, for example encoder, decoder, multiplexer and demultiplexer. Some of the characteristics of combinational circuits are following −

- The output of combinational circuit at any instant of time, depends only on the levels present at input terminals.
- The combinational circuit do not use any memory. The previous state of input does not have any effect on the present state of the circuit.
- A combinational circuit can have an n number of inputs and m number of outputs.

Block diagram

We're going to elaborate few important combinational circuits as follows.

Half Adder

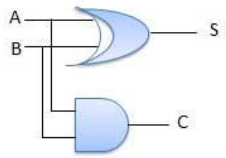
Half adder is a combinational logic circuit with two inputs and two outputs. The half adder circuit is designed to add two single bit binary number A and B. It is the basic building block for addition of two **single** bit numbers. This circuit has two outputs **carry** and **sum**. 1

Block diagram

Truth Table

Inputs		Output	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

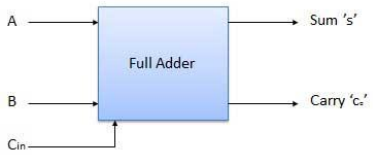
Circuit Diagram



Full Adder

Full adder is developed to overcome the drawback of Half Adder circuit. It can add two one-bit numbers A and B, and carry c. The full adder is a three input and two output combinational circuit.

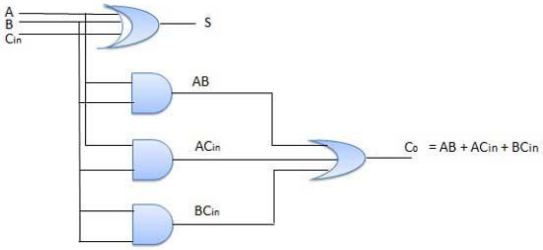
Block diagram



Truth Table

Inputs			Output	
A	B	Cin	S	Co
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Circuit Diagram



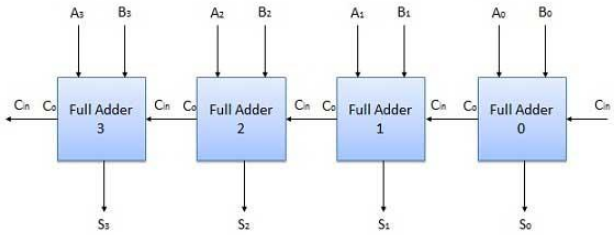
N-Bit Parallel Adder

The Full Adder is capable of adding only two single digit binary number along with a carry input. But in practical we need to add binary numbers which are much longer than just one bit. To add two n-bit binary numbers we need to use the n-bit parallel adder. It uses a number of full adders in cascade. The carry output of the previous full adder is connected to carry input of the next full adder.

4 Bit Parallel Adder

In the block diagram, A_0 and B_0 represent the LSB of the four bit words A and B. Hence Full Adder-0 is the lowest stage. Hence its C_{in} has been permanently made 0. The rest of the connections are exactly same as those of n-bit parallel adder is shown in fig. The four bit parallel adder is a very common logic circuit.

Block diagram



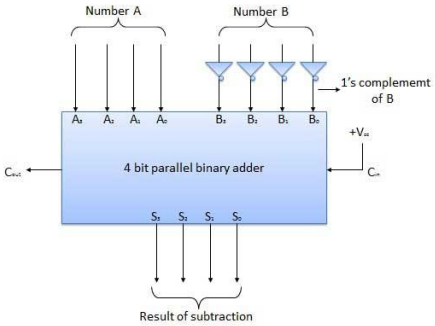
N-Bit Parallel Subtractor

The subtraction can be carried out by taking the 1's or 2's complement of the number to be subtracted. For example we can perform the subtraction (A-B) by adding either 1's or 2's complement of B to A. That means we can use a binary adder to perform the binary subtraction.

4 Bit Parallel Subtractor

The number to be subtracted (B) is first passed through inverters to obtain its 1's complement. The 4-bit adder then adds A and 2's complement of B to produce the subtraction. S_3 S_2 S_1S_0 represents the result of binary subtraction (A-B) and carry output C_{out} represents the polarity of the result. If A > B then Cout = 0 and the result of binary form (A-B) then C_{out} = 1 and the result is in the 2's complement form.

Block diagram



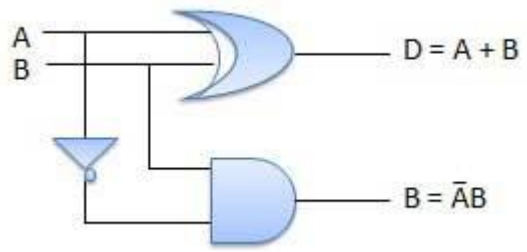
Half Subtractors

Half subtractor is a combination circuit with two inputs and two outputs (difference and borrow). It produces the difference between the two binary bits at the input and also produces an output (Borrow) to indicate if a 1 has been borrowed. In the subtraction (A-B), A is called as Minuend bit and B is called as Subtrahend bit.

Truth Table

Inputs		Output	
A	B	(A – B)	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Circuit Diagram



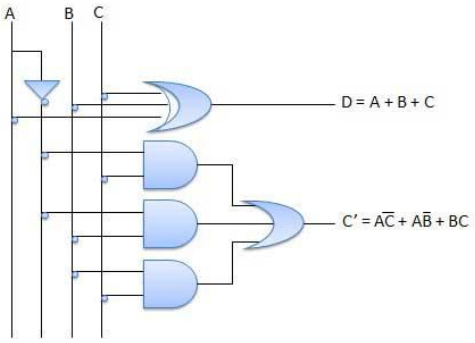
Full Subtractors

The disadvantage of a half subtractor is overcome by full subtractor. The full subtractor is a combinational circuit with three inputs A,B,C and two output D and C'. A is the 'minuend', B is 'subtrahend', C is the 'borrow' produced by the previous stage, D is the difference output and C' is the borrow output.

Truth Table

Inputs			Output	
A	B	C	(A-B-C)	C'
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

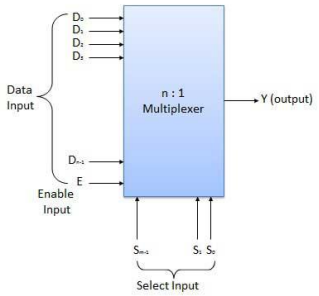
Circuit Diagram



Multiplexers

Multiplexer is a special type of combinational circuit. There are n-data inputs, one output and m select inputs with 2m = n. It is a digital circuit which selects one of the n data inputs and routes it to the output. The selection of one of the n inputs is done by the selected inputs. Depending on the digital code applied at the selected inputs, one out of n data sources is selected and transmitted to the single output Y. E is called the strobe or enable input which is useful for the cascading. It is generally an active low terminal that means it will perform the required operation when it is low.

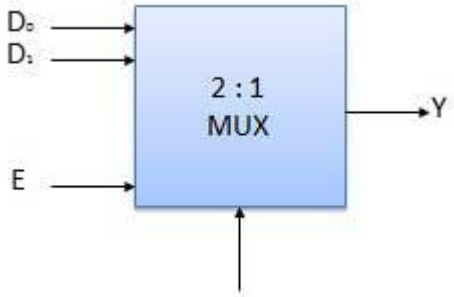
Block diagram



Multiplexers come in multiple variations

- 2 : 1 multiplexer
- 4 : 1 multiplexer
- 16 : 1 multiplexer
- 32 : 1 multiplexer

Block Diagram



Truth Table

Enable	Select	Output
E	S	Y
0	x	0
1	0	D_0
1	1	D_1

x = Don't care

Demultiplexers

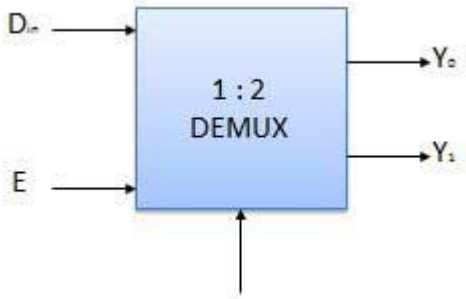
A demultiplexer performs the reverse operation of a multiplexer i.e. it receives one input and distributes it over several outputs. It has only one input, n outputs, m select input. At a time only one output line is selected by the select lines and the input is transmitted to the selected output line. A de-multiplexer is equivalent to a single pole multiple way switch as shown in fig.

Demultiplexers comes in multiple variations.

- 1 : 2 demultiplexer

- 1 : 4 demultiplexer
- 1 : 16 demultiplexer
- 1 : 32 demultiplexer

Block diagram



Truth Table

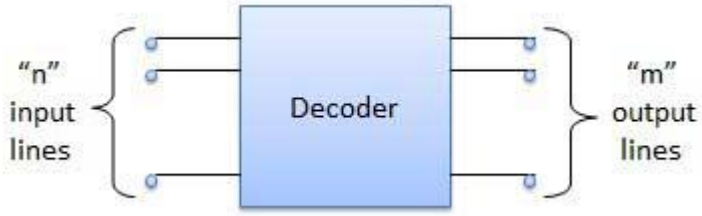
Enable	Select	Output	
E	S	Y0	Y1
0	x	0	0
1	0	0	Din
1	1	Din	0

x = Don't care

Decoder

A decoder is a combinational circuit. It has n input and to a maximum m = 2n outputs. Decoder is identical to a demultiplexer without any data input. It performs operations which are exactly opposite to those of an encoder.

Block diagram



Examples of Decoders are following.

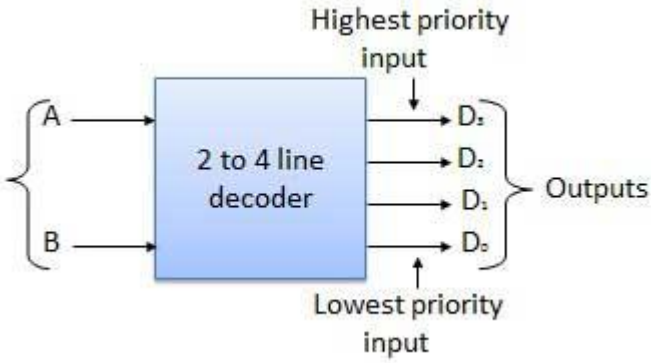
- Code converters
- BCD to seven segment decoders
- Nixie tube decoders
- Relay actuator

2 to 4 Line Decoder

The block diagram of 2 to 4 line decoder is shown in the fig. A and B are the two inputs where D through D are the four outputs. Truth table explains the operations of a decoder. It shows that each output is 1 for

only a specific combination of inputs.

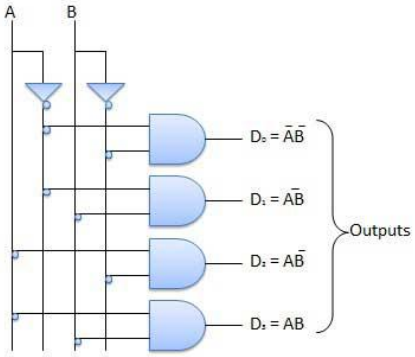
Block diagram



Truth Table

Inputs		Output			
A	B	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
0	1	0	0	1	0
1	1	0	0	0	1

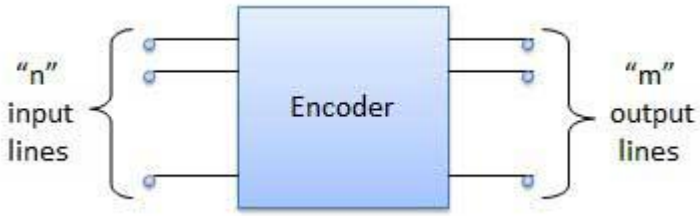
Logic Circuit



Encoder

Encoder is a combinational circuit which is designed to perform the inverse operation of the decoder. An encoder has n number of input lines and m number of output lines. An encoder produces an m bit binary code corresponding to the digital input number. The encoder accepts an n input digital word and converts it into an m bit another digital word.

Block diagram



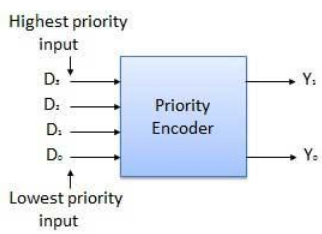
Examples of Encoders are following.

- Priority encoders
- Decimal to BCD encoder
- Octal to binary encoder
- Hexadecimal to binary encoder

Priority Encoder

This is a special type of encoder. Priority is given to the input lines. If two or more input line are 1 at the same time, then the input line with highest priority will be considered. There are four input D_0, D_1, D_2, D_3 and two output Y_0, Y_1. Out of the four input D_3 has the highest priority and D_0 has the lowest priority. That means if $D_3 = 1$ then $Y_1 Y_1 = 11$ irrespective of the other inputs. Similarly if $D_3 = 0$ and $D_2 = 1$ then $Y_1 Y_0 = 10$ irrespective of the other inputs.

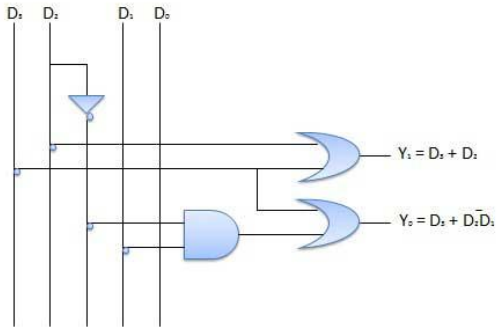
Block diagram



Truth Table

Highest	Inputs		Lowest	Outputs	
D_3	D_2	D_1	D_0	Y_0	Y_1
0	0	0	0	x	x
0	0	0	1	0	0
0	0	1	x	0	1
0	1	x	x	1	0
1	x	x	x	1	1

Logic Circuit

				
4	What is the need for flip flop? Describe the various types of flip flops ?			
	A **flip-flop** or latch is a circuit that has two stable states and can be used to store state information. A **flip-flop** is a bistable multivibrator. The circuit can be made to change state by signals applied to one or more control inputs and will have one or two outputs. 1.SR Flip-flops 2.D flip-flop 3.JK Flip-flop 4.T flip-flops	12	2	K2
5	Determine the procedure for the design of combinational circuits and synchronous sequential circuits.			
	! Sequential Circuits ! Latches ! Flip-Flops ! Analysis of Clocked Sequential Circuits ! State Reduction and Assignment ! Design Procedure	12	2	K2
6	Describe the shift register and design a 4 bit universal shift register	12	2	K2
	Universal Shift Register is a register which can be configured to load and/or retrieve the data in any mode (either serial or parallel) by shifting it either towards right or towards left. In other words, a combined design of unidirectional (either right- or left-shift of data bits as in case of SISO, SIPO, PISO, PIPO) and bidirectional shift register along with parallel load provision is referred to as **universal shift register**. Such a shift register capable of storing n input bits			

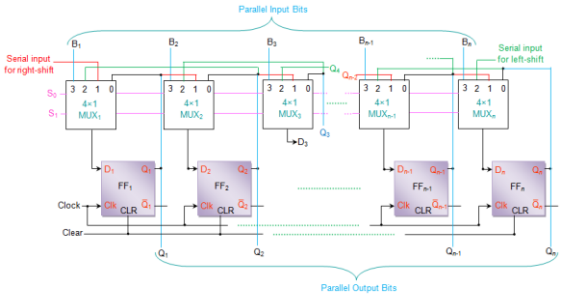


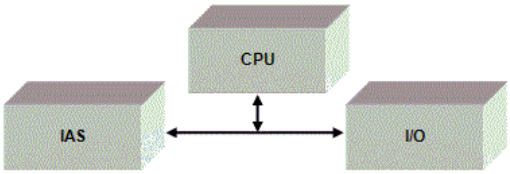
Figure 1 *n*-bit Universal Shift Register

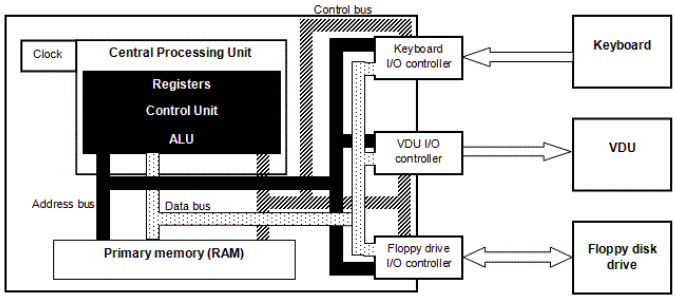
The design uses n 4×1 multiplexers to drive the input pins of n flip-flops in the register which are also connected to clock and clear inputs. All of the multiplexers in the circuit share the same select lines, S_1 and S_0 (pink lines in the figure), in order to select the mode in which the shift registers operates. It is also seen that the MUX driving a particular flip-flop has its

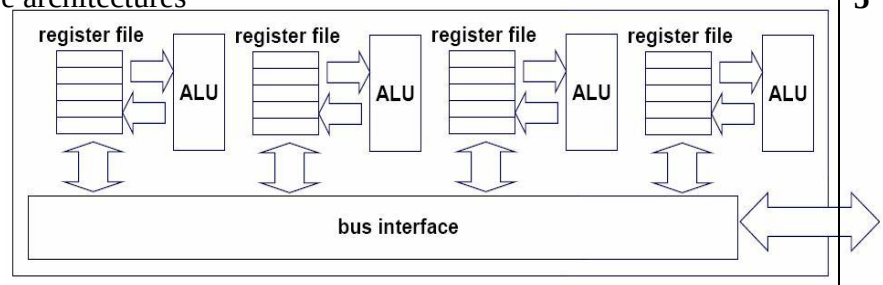
1. First input (Pin Number 0) connected to the output pin of the same flip-flop i.e. zeroth pin of MUX1 is connected to Q1, zeroth pin of MUX2 is connected to Q2, … zeroth pin of MUXn is connected to Qn.
2. Second input (Pin Number 1) connected to the output of the very-previous flip-flop (except the first flip-flop FF1 where it acts like an serial-input to the input data bits which are to be shifted towards right) i.e. first pin of MUX2 is connected to Q1, first pin of MUX3 is connected to Q2, … first pin of MUXn is connected to Qn-1.
3. Third input (Pin Number 2) connected to the output of the very-next flip-flop (except the first flip-flop FFn where it acts like an serial-input to the input data bits which are to be shifted towards left) i.e. second pin of MUX1 is connected to Q2, second pin of MUX2 is connected to Q3,… second pin of MUXn-1 is connected to Qn.
4. Fourth input (Pin Number 3) connected to the individual bits of the input data word which is to be stored into the register, thus providing the facility for parallel loading.

UNIT III - PROCESSOR FUNDAMENTALS

1.	Discuss briefly about **Von Neumann architecture**?			
	Von Neumann architecture In the 1940s, a mathematician called John Von Neumann described the basic arrangement (or architecture) of a computer. Most computers today follow the concept that he described although there are other types of architecture. When we talk about the Von Neumann architecture, we are actually talking about the relationship between the hardware that makes up a Von Neumann-based computer. A Von Neumann-based computer is a computer that: • Uses a single processor. • Uses one memory for both instructions and data. A von Neumann computer cannot distinguish between data and	**5**	**3**	**K2**

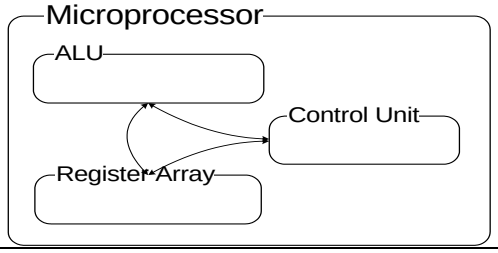
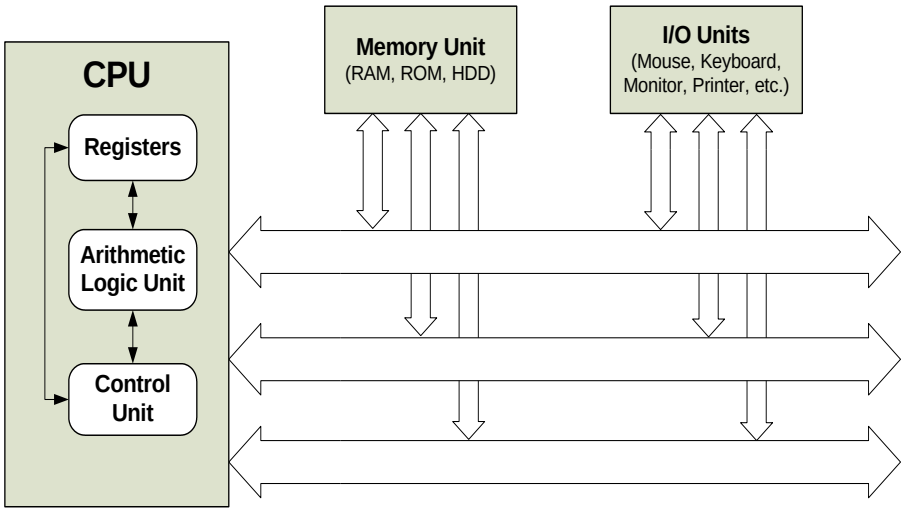
	instructions in a memory location! It 'knows' only because of the *location* of a particular bit pattern in RAM. • Executes programs by doing one instruction after the next in a serial manner using a fetch-decode-execute cycle. In this chapter, we are going to build upon and refine the ideas introduced in an earlier chapter. You should re-read the relevant chapter on CPUs before you start this one. We have already said that the CPU was made up of 4 important components: • The ALU. • The registers. • The control unit. • The IAS (otherwise known as RAM or memory).			
	 A model of a Von Neumann computer system.	**2**		
	We know a few things from before about the Von Neumann CPU. **1) The ALU, or Arithmetic Logic Unit** A Von Neumann CPU has an ALU. This is the part of the CPU that performs arithmetic and logic operations on data and acts as the revolving for the CPU, letting data enter and leave the CPU. We also know that CPUs have a 'word size'. This is the number of bits that can be added, for example, in one go. The bigger a CPU's word size, the more bits it can work on in one clock cycle and the more work you can get done. **2) The Control Unit** A Von Neumann CPU has a control unit. The control unit is in charge of 'fetching' each instruction that needs to be executed in a program by issuing control signals to the hardware. It then decodes the instruction and finally issues more control signals to the hardware to actually execute it. **3) Registers** A Von Neumann CPU has registers. These are very fast memory circuits. They hold information such as the address of the next instruction (Program Counter), the current instruction being executed (Current Instruction Register), the data being worked on and the results of arithmetic and logical operations (Accumulators), information about the last operation (Status Register) and whether an interrupt has happened (Interrupt Register). Registers are covered in	**5**		

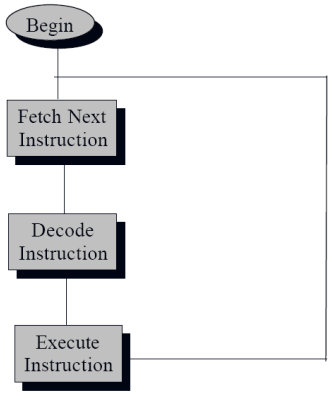
	a lot more detail later in this chapter. **4) The clock** Instructions are carried out to the beat of the clock! Some instructions take one beat and others more than one beat. Very roughly speaking, the faster the clock, the more clock beats you have per second so the more instructions per section you can do and the faster your computer will go.  A computer system showing the I/O controllers.			
2.	Discuss briefly about Processor and their functions?			
	o Functions · Fetch Instruction · Interpret Instruction · Fetch Data minor cycle · Execute Instruction, i.e., process data · Write data results, to memory or I/O module	**4**	**3**	**K2**
	o Major Components of Processor · ALU	**3**		
	o Additional Processor Components Register Organization • User Visible Registers, referenced by machine language General Purpose Registers • Orthogonal – any register can contain the operand for any opcode Dedicated – e.g., stack registers, floating-point registers Data Registers • Used only to hold data, excluding addresses Used to hold data including addresses Address Registers Partially General Purpose, e.g., X register in Pep/8 Dedicated Segment Pointers, i.e., Registers holds the address of the base of the segment Index Registers Indexed addressing, may autoindex Stack Pointer Enables push, pop, etc.	**5**		
3.	**Describe briefly about Multi-core architectures**			
	Replicate multiple processor cores on a single die.	**2**	**3**	**K2**

	Multi-core architectures  	**5**		
	• Interaction with OS • OS perceives each core as a separate processor • OS scheduler maps threads/processes to different cores • Most major OS support multi-core today	**5**		
4	**Discuss briefly about Instruction Format?**			
	Instruction Format All instructions are 32 bits long. There are four types of instruction format. • Arithmetic instruction format • Conditional Branch and Immediate format • Unconditional Jump format • Input and Output instruction format	**2**	**3**	**K2**
	Arithmetic instruction format 2 bits 6 bits 4 bits 4 bits 4 bits 12 bits 00 | OPCODE | S-reg | S-reg | D-reg | 000 The first two bits are always 00, indicating that the instruction is an Arithmetic or Register transfer type of instruction. S-reg is the source register. D-reg is the destination register. The last 12 bits are always 0, as they are not used.	**2**		
	Conditional Branch and Immediate format 2 bits 6 bits 4 bits 4 bits 16 bits 01 | OPCODE | B-reg | D-reg | Address The first two bits are always 01, indicating that the instruction is a Conditional Branch and Immediate type of instruction. B-reg is the base register. D-reg is the destination register. The last 16 bits may be an address or an immediate data. When the last 16 bits contain data, the D-reg is always 0000. The Address may at times be treated as data, which is direct addressing. An indirect Address is calculated as : Effective Address = Content (B-reg) + Address Conditional Branch checks for B and D reg to cause a branch, to a specified Address, or not **Unconditional Jump format** 2 bits 6 bits 24 bits 10 | OPCODE | Address	**2**		

	The first two bits are always 10, indicating that the instruction is an Unconditional Jump type of instruction, with a jump to the specified Address.			
	Input and Output instruction format 2 bits 6 bits 4 bits 4 bit 16 bits 11 | OPCODE | Reg 1 | Reg 2 | Address The first two bits are always 11, indicating that the instruction is an Input and Output type of instruction. The instruction may read the content of Address/Reg 2 into Reg 1. The instruction may write the content of Reg 1 into a specified Address/Reg 2.	**2**		
	Registers There are 16 registers; each of 32 bit long. Reg-0 (0000) being the Accumulator. Reg-1(0001) being the Zero register, which contains the value 0. All other registers are general purpose register.	**2**		
	Buffers Input buffer – containing data read by the program Output buffer – containing data produced by the program Temp buffer – area in memory to store/retrieve the data temporarily.	**2**		
5.	**Discuss briefly about Arithmetic Logic Unit?**			

	Definition
	• Key processing element of a microprocessor that performs arithmetic and logic operations
	Description
	• Directed by Control Unit, ALU performs operations such as ADD, SUB, NOT, OR, AND, XOR
	• Data is inputted from and outputted to the Register Array
	• Control Signals from Control Unit determine what type of operation is performed
	• Input data consists of two operands: operand A and operand B stored in registers and having n bits
	• Output data consists of result S
	• ALU also outputs Status Signals such as:
	  o Zero (when the result of the operation is 0)
	  o Negative (when the operation result is < 0)
	  o Carry (when the operation results in carry)
	  o Overflow (when the result exceeds the number of bits allocated for its storage)
		6		

	○ Etc.			
		**2**		
		**4**		
6.	**Discuss briefly about Instruction Fetch Execute Cycle?**			
	The Instruction Fetch Execute Cycle is one of the most important mental models of computation as aptly put by Prof. Rockford Ross. This embodies the basic principle of how all modern processors work. This functional model has remained more or less the same over the decades no matter how and when the development of processors have taken place ever since the days of Von Newmann architecture to today's Super computers. The principles are fairly simple and can be easily generalized to any processor or Operating System. It further proceeds to explain what happens when a computer is first switched on till the time it is ready to accept instructions from the user. It is very important to appreciate the fact that an Operating System (OS) is just like any other program albeit a little more complex as compared to user written programs. The OS provides the functionality to load and execute other programs and thus possesses certain privileges which user programs do not possess. A user program can and does request for OS intervention through a mechanism called system calls e.g. trap, page fault etc. Once a computer has been powered on it performs a continuous cycle of the following: a. Fetch next instruction from memory b. Decode the instruction	**4**	**3**	**K2**

	c. Execute the instruction			
	 Figure 1: Basic Instruction Fetch Execute Cycle	**3**		
	Instruction Fetch Execute Cycle A more complete form of the Instruction Fetch Execute Cycle can be broken down into the following steps: 1. Fetch Cycle 2. Decode Cycle 3. Execute Cycle 4. Interrupt Cycle **1. Fetch Cycle** The fetch cycle begins with retrieving the address stored in the *Program Counter* (PC). The address stored in the PC is some valid address in the memory holding the instruction to be executed. (In case this address does not exist we would end up causing an interrupt or exception).The Central Processing Unit completes this step by fetching the instruction stored at this address from the memory and transferring this instruction to a special register – *Instruction Register (IR)* to hold the instruction to be executed. The program counter is incremented to point to the next address from which the new instruction is to be fetched. **2. Decode Cycle** The decode cycle is used for interpreting the instruction that was fetched in the Fetch Cycle. The operands are retrieved from the addresses if the need be. **3. Execute Cycle** This cycle as the name suggests , simply executes the instruction that was	**5**		

fetched and decoded.

4. Interrupt Cycle

An interrupt can occur any time during the program execution. Whenever it is caused, a series of events take place so that the instruction fetch execute cycle can again resume after the OS calls the routine to handle the interrupt. Therefore, when an interrupt occurs, the following steps are performed by the OS:

- · Suspend the execution of current instruction
- · Push the address of current instruction on the system stack
- · Loading the PC with the address of the interrupt handler
- · This starts the Instruction Fetch Execute cycle again for the instructions in the Interrupt handler.
- · Set the mode of operation as a privileged one often termed as the *Supervisor mode* so that the OS can execute the handler.
- · Once the OS completes the execution of the interrupt handler, the address of the next instruction to be executed is obtained from popping the value of the address in the stack. The suspended instruction can now continue with its execution.

This cycle of fetching a new instruction, decoding it and finally executing it continues until the computer is turned off. Since we have said that it is mainly the operating system which aids the processor in executing the programs which holds the different instructions, a question which is immediately raised is how does the OS start executing.

Loading of the Operating System

The process by which the OS gets loaded into the memory so that it can start executing is known as the *System Bootup*. During the boot up sequence, a series of instructions need to be executed so that at the end of this sequence the OS is running and can in turn start executing user programs. The boot sequence begins with the following:

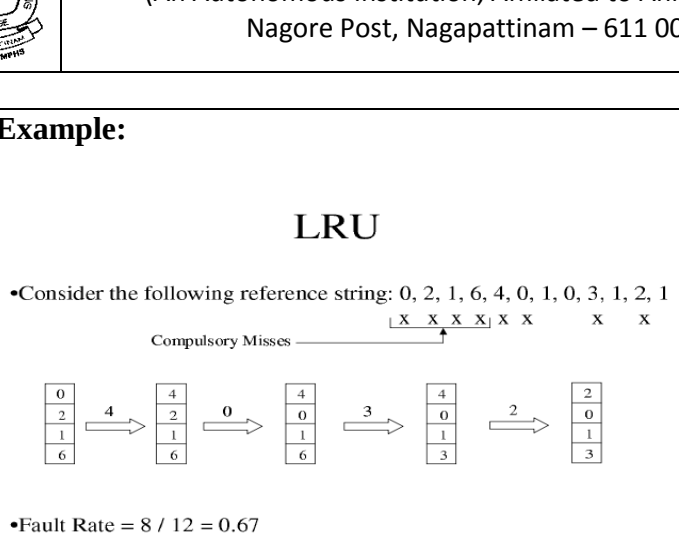
1. The *RESET* pin of the CPU is set to logical high.
2. The code which is found at some specific starting address (0xffffff0 in case of an Intel processor) is executed.
3. 0xffffff0 maps to the persistent memory chip of the computer known as the Read Only Memory (ROM).
4. The series of instructions stored in ROM is called the *Basic Input /Output System (BIOS)*.

Role of BIOS during System Bootup

Although the BIOS performs a number of functions e.g. making sure that all the different chips, hard drives, and the CPU function together as an entity, its most important function is loading the Operating System. When the computer is first powered on, the microprocessor attempts to execute the first instruction. For this purpose as mentioned earlier the

	processor needs to fetch this instruction. The processor cannot fetch it from the Operating System because the OS has not been loaded yet into the memory; it is still residing on the disk. It is the BIOS which provides the processor with the first instructions to be executed in order to load the OS. The other functions performed by the BIOS during the system bootup apart from loading the OS are summarized as: • · A power-on self-test **(POST)** for the different hardware components in the system to make sure everything are functioning properly. • · Activating other BIOS entities e.g. graphics cards. • · Providing a set of low-level routines to enable the OS to interface different hardware devices keyboard, screen, and the ports (serial as well parallel).			
UNIT IV - MEMORY				
1.	**Discuss briefly about Virtual Memory?**			
	Virtual Memory: ∗Virtual memory deals with the main memory size limitations ∗ Provides an illusion of having more memory than the system's RAM	**3**	**4**	**K2**
	Virtual memory concepts ∗ Page replacement policies ∗ Write policy ∗ Page size tradeoff ∗ Page mapping	**6**		
	Page table organization ∗Page table entries	**3**		
2.	**Explain briefly about Paging?**			
	Paging: • Memory is divided into **page frames** all of equal size. • The logical address space divided into **pages** of equal size**.** The memory manager determines 1. The number of pages in the program 2. Locates enough empty page frames to facilitate 3. Loads all of the pages into memory, pages need not be contiguous. A number of tables need to be maintained for this system to operate: 1. **Job Table**- for each job holds the size of the job, the memory location of the Page table. 2. **Page Table** – For each active job the Page , page Frame memory address 3. **Memory Map table** – for each page Frame its location and whether free or busy.	**4**	**4**	**K2**

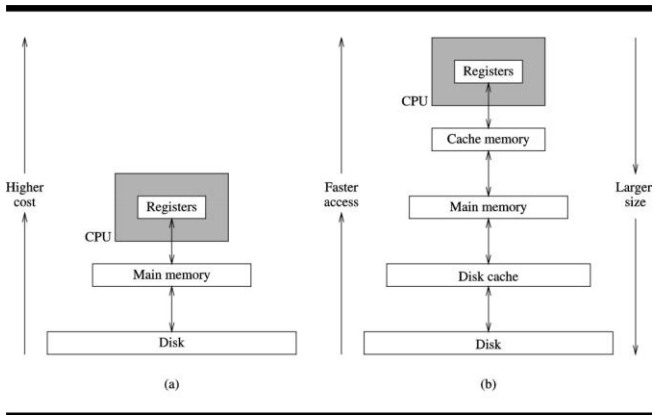
	Page addressing Page | Offset → Frame No | Offset Memory address translation takes place at run time. Reading a word from memory involves translating a virtual or logical address, consisting of a page number and offset, into an actual physical address, consisting of a frame number and offset. This process will make use of the Page Table entries. If the system used 16 bits then it could utilize memory in this fashion: 15 10 0 Page no Displacement with this set up the system would have 32 pages (2^5) each with 2048 bytes (2^{11}) Example If a logical address of 0010100000101010 was encountered this will represent offset 42 on page 5. The Page table would be accessed to see the **Mapping** of page 5. Static paging: • No external fragmentation • Fixed size pages • Internal fragmentation – only on last page • Non- contiguous memory (page table)	**8**		
3.	**Explain briefly about algorithm of LRU Replacement and give example?**			
	LRU(Least Recently Used): In this algorithm, the page that has not been used for longest period of time is selected for replacement. Although LRU is theoretically realizable, it is not cheap. To fully implement LRU, it is necessary to maintain a linked list of all pages in memory, with the most recently used page at the front and the least recently used page at the rear. The difficulty is that the list must be updated on every memory reference. Finding a page in the list, deleting it, and then moving it to the front is a very timeconsuming operation, even in hardware (assuming that such hardware could be built).	**4**	**4**	**K2**

	Example:  	**6**		
	• Two status bit associated with each page. R is set whenever the page is referenced (read or written). M is set when the page is written to (i.e., modified). • When a page fault occurs, the operating system inspects all the pages and divides them into four categories based on the current values of their R and M bits: **Class 0**: not referenced, not modified. **Class 1:** not referenced, modified. **Class 2:** referenced, not modified. **Class 3:** referenced, modified. The NRU (Not Recently Used) algorithm removes a page at random from the lowest numbered nonempty class.	**2**		
4.	Discuss how the cache memory Works, design of cache and their types		**4**	**K2**

cache memory

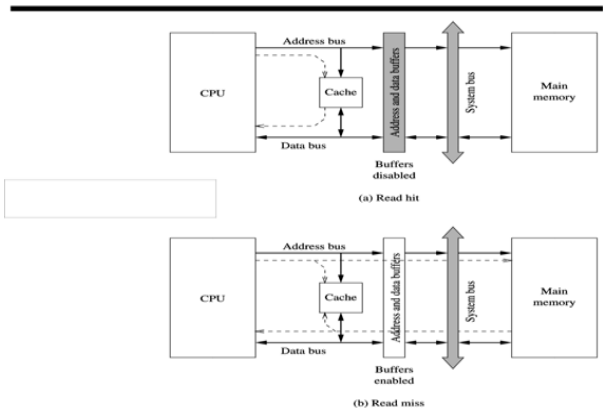
Cache memory is a small amount of fast memory ∗ Placed between two levels of memory hierarchy » To bridge the gap in access times – Between processor and

main memory (our focus) – Between main memory and disk (disk cache) ∗ Expected to behave like a large amount of fast memory



Working of Cache Memory:

Prefetch data into cache before the processor needs it ∗ Need to predict processor future access requirements » Not difficult owing to locality of reference



Cache design basics

- Mapping function

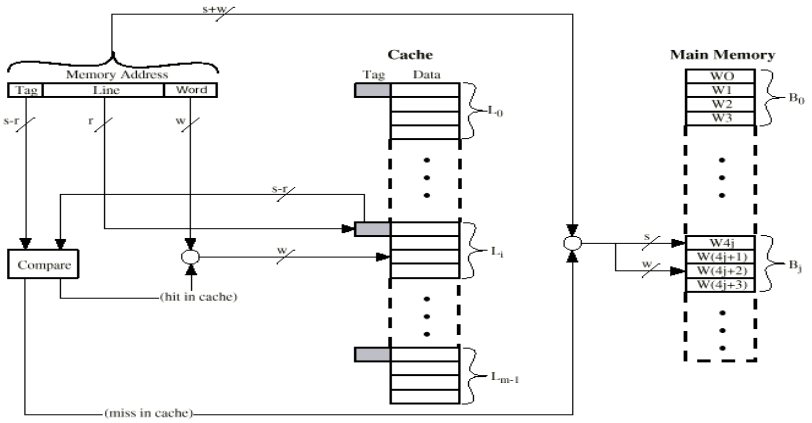
∗Direct mapping

∗ Associative mapping

∗ Set-associative mapping

5.	**Explain about addressing modes and their types?**			
	The simplest addressing mode is to include the operand itself in the instruction, that is, no address information is needed The different ways in which operands can be addressed are called the addressing modes. Addressing modes differ in the way the address information of operands is specified.. This is called immediate addressing. A more involved addressing mode is to compute the address of the operand by adding a constant value to the content of a register. This is called indexed addressing. Between these two addressing modes there exist a number of other addressing modes including absolute addressing, direct addressing, and indirect addressing.	**2**	**4**	**K2**
	Immediate Mode According to this addressing mode, the value of the operand is (immediately) available in the instruction itself.	**2**		
	Direct (Absolute) Mode According to this addressing mode, the address of the memory location that holds the operand is included in the instruction.	**2**		
	Indirect Mode In the indirect mode, what is included in the instruction is not the address of the operand, but rather a name of a register or a memory location that holds the (effective) address of the operand.	**2**		
	Indexed Mode In this addressing mode, the address of the operand is obtained by adding a constant to the content of a register, called the index register.	**1**		
	Relative Mode Recall that in indexed addressing, an index register is used. Relative addressing is the same as indexed addressing except that the program counter (PC) replaces the index register.	**1**		
	Autoincrement Mode This addressing mode is similar to the register indirect addressing mode in the sense that the effective address of the operand is the content of a register, call it the autoincrement register, that is included in the instruction.	**1**		
	Autodecrement Mode Similar to the autoincrement, the autodecrement mode uses a register to hold the address of the operand.	**1**		
6.	**Discuss briefly about types of Cache Mapping?**			
	Direct Mapping Cache Organization	**4**	**4**	**K2**

- Each block of main memory maps to only one cache line
  - i.e. if a block is in cache, it must be in one specific place
  - Address is in two parts
  - Least Significant w bits identify unique word
  - Most Significant s bits specify which one memory block
- All but the LSBs are split into
  - a cache line field r and
  - a tag of s-r (most significant)



Example Direct Mapping Address Structure

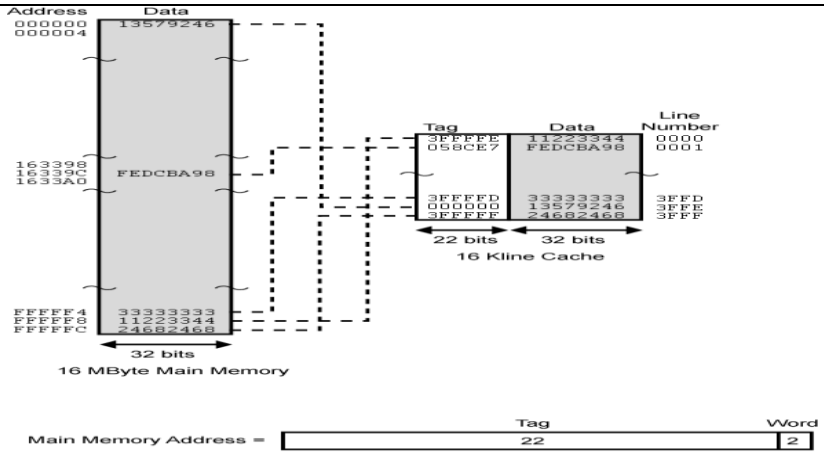
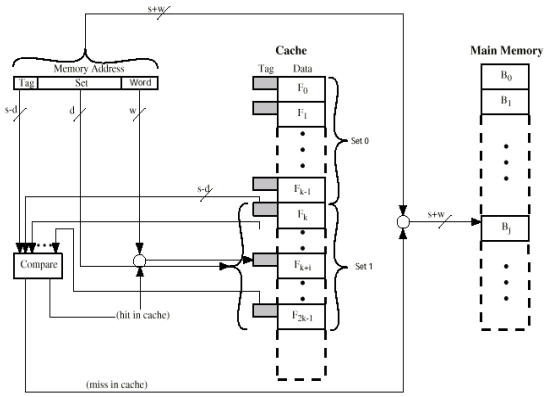
Tag s-r	Line or Slot r	Word w
8	14	2

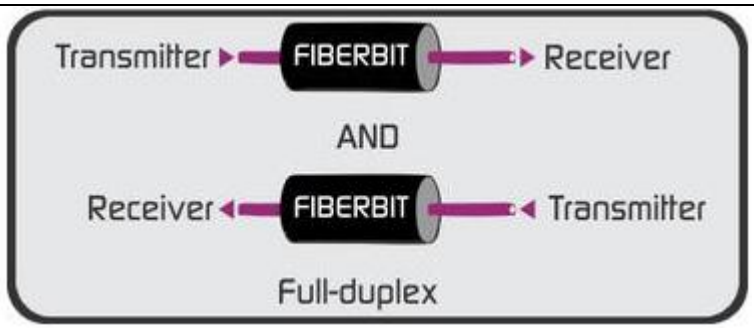
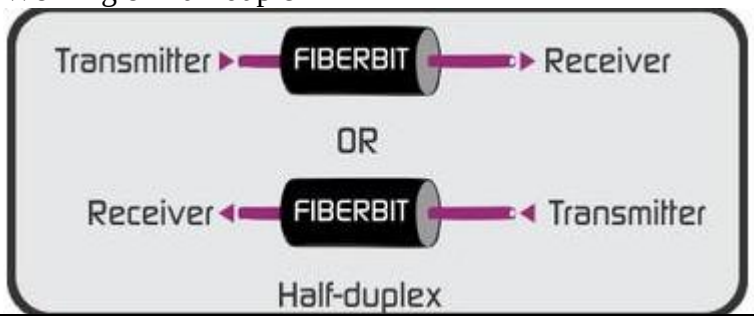
Associative Cache Mapping **4**

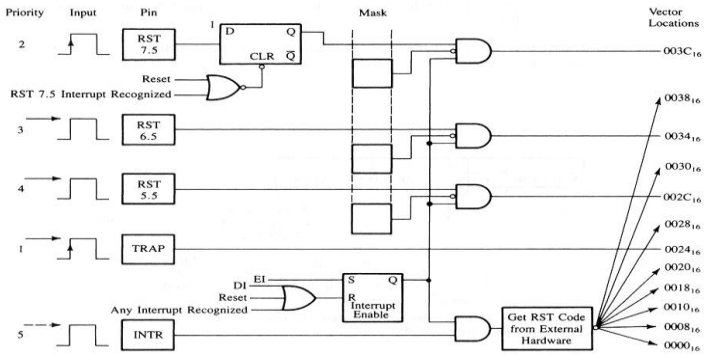
- A main memory block can load into any line of cache
- The Memory Address is interpreted as tag and word
- The Tag uniquely identifies block of memory
- Every line's tag is examined for a match

Cache searching gets expensive/complex or slow

Associative Caching Example

				
	Set Associative Mapping • Cache is divided into a number of sets • Each set contains a number of lines • A given block maps to any line in a given set – e.g. Block B can be in any line of set *i* • e.g. with 2 lines per set – We have 2 way associative mapping – A given block can be in one of 2 lines in only one set	**4**		
	Associative Caching Example 			
UNIT V – DATA TRANSFER				
1	**Explain the characteristics and working of Full duplex?**	12	5	K2
	Full Duplex: Full-Duplex is like the ordinary two-lane highway. In some cases, where traffic is heavy enough, a railroad will decide to lay a double track to allow trains to pass in both directions. In communications, this is most common with networking.	2		
	Characteristics of Full duplex:	5		
	Working of Full duplex:	5		

				
2	**Explain the characteristics and working of Half duplex?**	12	5	K2
	Half Duplex: Half-Duplex is like the dreaded "one lane" road you may have run into at construction sites. Only one direction will be allowed through at a time. Railroads have to deal with this scenario more often since it's cheaper to lay a single track.	2		
	Characteristics of Half duplex	5		
	Working of Half duplex 	5		
3	**Discuss briefly about Interrupt vectors?**			
	Interrupt Vector: • An interrupt vector is a pointer to where the ISR is stored in memory. • All interrupts (vectored or otherwise) are mapped onto a memory area called the Interrupt Vector Table (IVT).	3	5	K2
	Functions of Interrupt vector: • The IVT is usually located in memory page 00 (0000H - 00FFH). There are four interrupt inputs in 8085 that transfer the operation immediately to a specific address: ▪ TRAP : go to 0024 ▪ RST 7.5: go to 003C ▪ RST 6.5 0034 ▪ RST 5.5 002C ▪ RST 7.5, RST 6.5 and RST 5.5 are maskable interrupts, they are acknowledged only if they are not masked ! Finding the address of these vectored interrupts are very easy .Just multiply 8 with the RST value i.e for RST 7.5 the subroutine(ISR) address=8*7.5=60=(3c)H.	6		

	For TRAP ,its RST value is 4.5,then the subroutine address is 8*4.5=36=(24)H. similarly u can calculate for other vector interupt addresses. Memory page for all interrupts are (00).			
		3		
4.	**Explain the working and design issue of Interrupt driven I/O?**		5	K2
	Interrupt driven I/O: The problem with programmed I/O is that the processor has to wait a long time for the I/O module of concern to be ready for either reception or transmission of data. The processor, while waiting, must repeatedly interrogate the status of the I/O module.	02		
	Working e of Interrupt driven I/O: a) From the point of view of the I/O module: • For input, the I/O module services a READ command from the processor. • The I/O module then proceeds to read data from an associated peripheral device. • Once the data are in the modules data register, the module issues an interrupt to the processor over a control line. • The module then waits until its data are requested by the processor. • When the request is made, the module places its data on the data bus and is then ready for another I/O operation. From the processor point of view; the action for an input is as follows: • The processor issues a READ command. • It then does something else(e.g. the processor may be working on several different programs at the same time) • At the end of each instruction cycle, the processor checks for interrupts • When the interrupt from an I/O module occurs, the processor saves the context (e.g. program counter & processor registers) of the current program and processes the interrupt. • In this case, the processor reads the word of data from the I/O module and stores it in memory.	07		

	• If then restores the context of the program it was working on and resumes execution.			
	Design issues for Interrupt: Two design issues arise in implementing interrupt I/O. • There will almost invariably be multiple I/O modules, how does the processor determine which device issued the interrupt • If multiple interrupts have occurred how the processor does decide which one to process	03		
5.	**Explain how to Handling the multiple interrupt?**	12	5	K2
	i. There are several techniques to identify the requesting I/O module. These techniques also provide a way of assigning priorities when more than one device is requesting interrupt service. ii. With multiple lines, the processor just picks the interrupt line with highest priority. During the processor design phase itself priorities may be assigned to each interrupt lines. iii. With software polling, the order in which modules are polled determines their priority. iv. In case of daisy chain configuration, the priority of a module is determined by the position of the module in the daisy chain. The module nearer to the processor in the chain has got higher priorities, because this is the first module to receive the acknowledge signal that is generated by the processor. v. In case of bus arbitration method, more than one module may need control of the bus. Since only one module at a time can successfully transmit over the bus, some method of arbitration is needed. The various methods can be classified into two group – centralized and distributed. vi. In a centralized scheme, a single hardware device, referred to as a bus controller or arbiter is responsible for allocating time on the bus. The device may be a separate module or part of the processor. vii. In distributed scheme, there is no control controller. Rather, each module contains access control logic and the modules act together share bus. viii. It is also possible to combine different device identification techniques to identify the devices and to set the priorities of the devices. As for example multiple			
6.	**i)Explain briefly about serial Data Transfer?**			
	Serial transmission Within a piece of equipment, the distance and hence lengths of wire used to connect each subunit together are short. Thus, it is normal practice to transfer the data between subunits by using a separate piece of wire to carry each bit of the data. This means that there are multiple wires connecting each subunit together and data are said to be exchanged using a **parallel transfer mode**. This mode of operation results in minimal delays in transferring each word.	4	5	K2

When transferring information between two physically separate pieces of equipment, especially if the separation is more than several metres, for reasons of cost and varying transmission delays in the individual wires, it is more usual to use just single pair lines and transmit each octet making up the data a single bit at a time using a fixed time interval for each bit. This mode of operation is known as **bit-serial transmission**.

3

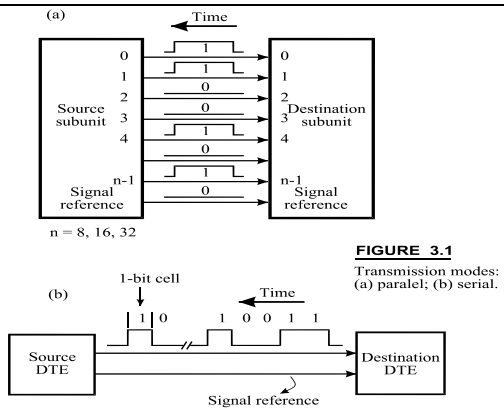


FIGURE 3.1
Transmission modes: (a) paralel; (b) serial.

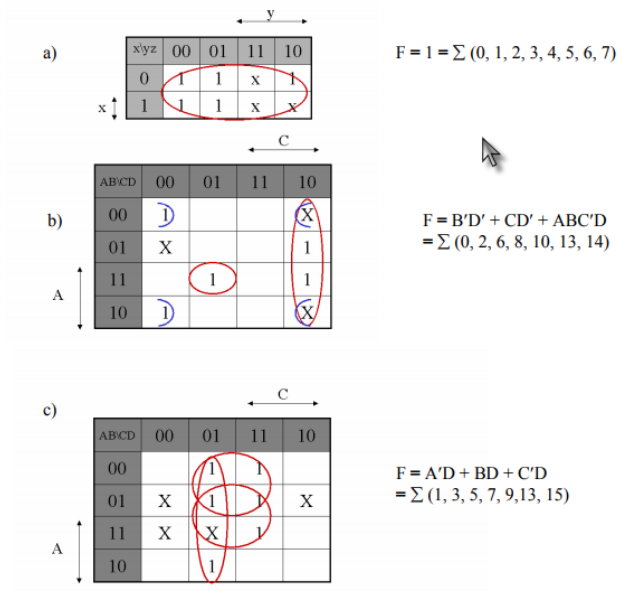
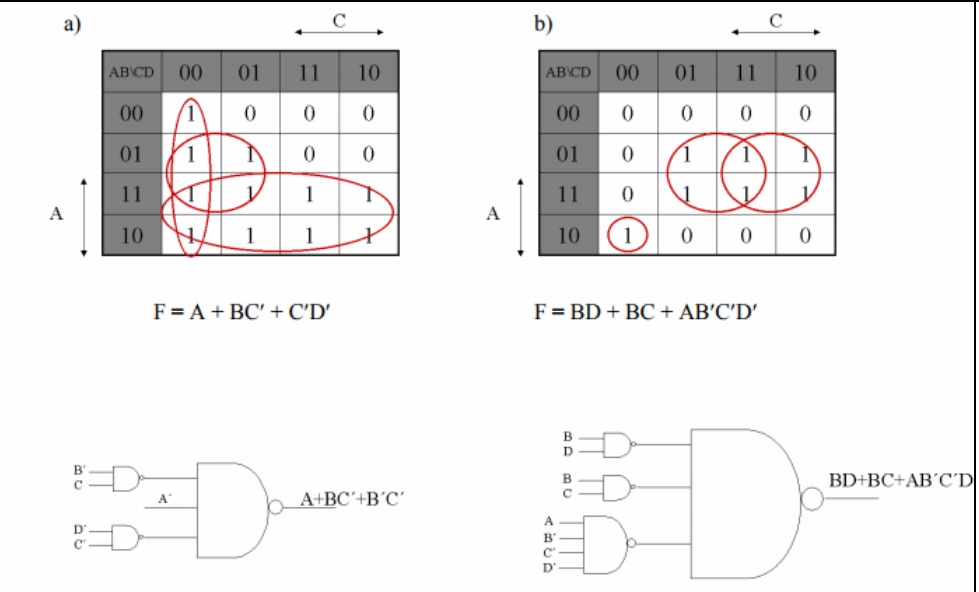
ii)Difference between serial and parallel data Transmission

5

BASIS FOR COMPARISON	**SERIAL TRANSMISSION**	**PARALLEL TRANSMISSION**
Meaning	Data flows in bi-direction, bit by bit	Multiple lines are used to send data i.e. 8 bits or 1 byte at a time
Cost	Economical	Expensive
Bits transferred at 1 clock pulse	1 bit	8 bits or 1 byte
Speed	Slow	Fast
Applications	Used for long distance communication. Eg, Computer to computer	Short distance. Eg, computer to printer

PART – C (20 Mark Questions with Key)

S.No	**Questions**	**Mark**	**COs**	**BTL**
UNIT I – DIGITAL FUNDAMENTALS				
1	Simplify the following Boolean function F , together with the don't-care conditions d, and then express the simplified function in sum of minterms: a) F(x, y, z) = ∑ (0, 1, 2, 4, 5) , d(x, y, z) = ∑(3, 6, 7) (4) b) F(A, B, C, D) = ∑ (0, 6, 8, 13, 14), d(A, B, C, D) = ∑ (2, 4, 10) (8) c) F(A, B, C, D) = ∑ (1,3,5,7,9,15), d(A, B, C, D) = ∑ (4,6,12,13)	20	1	K3

	(8)			
	 a) F = 1 = ∑ (0, 1, 2, 3, 4, 5, 6, 7) b) F = B′D′ + CD′ + ABC′D = ∑ (0, 2, 6, 8, 10, 13, 14) c) F = A′D + BD + C′D = ∑ (1, 3, 5, 7, 9,13, 15)	4 8 8		
2	Simplify the following expressions, and implement them with two-level NAND gate circuits: a) AB′ + ABD + ABD′ + A′C′D′ + A′BC′ b) BD + BCD′ + AB′C′D′	20	1	K3
	 a) F = A + BC′ + C′D′ b) F = BD + BC + AB′C′D′	10 10		
UNIT II – COMBINATIONAL AND SEQUENTIAL CIRCUITS				
1	i.Design a combinational circuit with three inputs x,y,z and three outputs A,B,C. When the binary inputs in 0,1,2 or 3, the binary output is one greater than the input.When the binary input is 4,5,6 or 7,the binary output is one less than the input (15) ii.Explain the operation of JK flip flop (5)	20	2	K3

x	y	z	A	B	C
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	0	1	1
1	0	1	1	0	0
1	1	0	1	0	1
1	1	1	1	1	0

y

x\yz	00	01	11	10
0			1	
1		1	1	1

x

A = xy + xz + yz

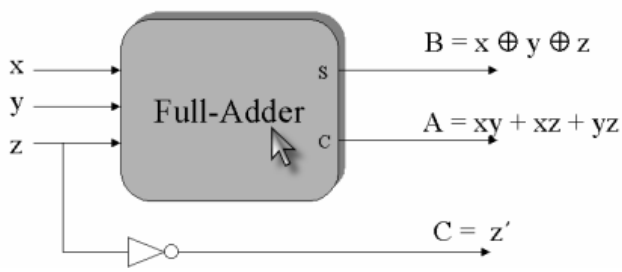
y

x\yz	00	01	11	10
0		1		1
1	1		1	

x

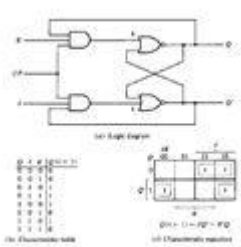
B = x ⊕ y ⊕ z C = z′

i.



iv. JK Flip-Flop

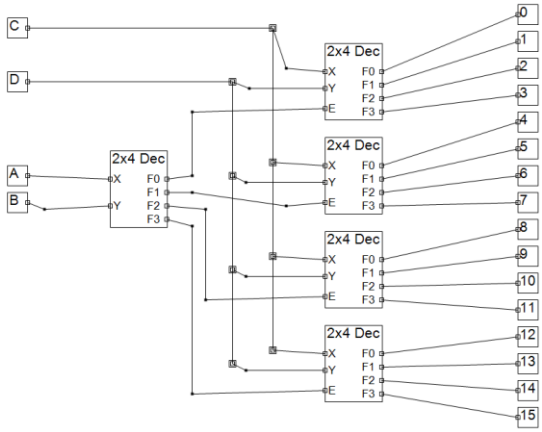
In the previous article we discussed RS and D flip-flops. Now we'll lrean about the other two types of flip-flops, starting with JK flip flop and its diagram.



A JK flip-flop has two inputs similar to that of RS flip-flop. We can say JK flip-flop is a refinement of RS flip-flop. JK means Jack Kilby, a Texas instrument engineer who invented IC. The two inputs of JK Flip-flop is J (set) and K (reset). A JK flip-flop is nothing but a RS flip-flop along with two AND gates which are augmented to it.

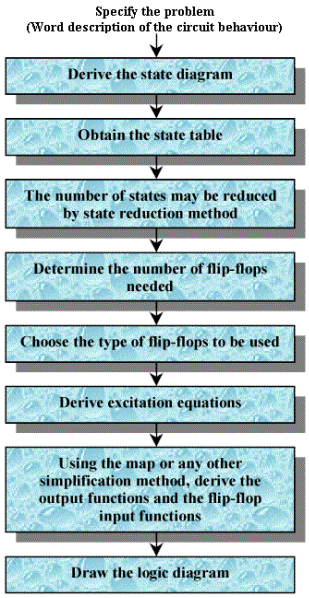
The flip-flop is constructed in such a way that the output Q is ANDed with K and CP. This arrangement is made so that the flip-flop is cleared

	during a clock pulse only if Q was previously 1. Similarly Q' is ANDed with J and CP, so that the flip-flop is cleared during a clock pulse only if Q' was previously 1. When J=K=0 When both J and K are 0, the clock pulse has no effect on the output and the output of the flip-flop is the same as its previous value. This is because when both the J and K are 0, the output of their respective AND gate becomes 0. When J=0, K=1 When J=0, the output of the AND gate corresponding to J becomes 0 (i.e.) S=0 and R=1. Therefore Q' becomes 0. This condition will reset the flip-flop. This represents the RESET state of Flip-flop. When J=1, K=0 In this case, the AND gate corresponding to K becomes 0(i.e.) S=1 and R=0. Therefore Q becomes 0. This condition will set the Flip-flop. This represents the SET state of Flip-flop. When J=K=1 Consider the condition of CP=1 and J=K=1. This will cause the output to complement again and again. This complement operation continues until the Clock pulse goes back to 0. Since this condition is undesirable, we have to find a way to eliminate this condition. This undesirable behavior can be eliminated by Edge triggering of JK flip-flop or by using master slave JK Flip-flops. The characteristic table explains the various inputs and the states of JK flip-flop.			
2	i.Discuss and draw the circuit diagram of a 4 to 16 line decoder with five 2 to 4 line decoders with enable. (12) ii. Steps involved in the design of sequential circuits (8)	20	2	K2
	A 4x16 decoder has 4 inputs and 16 outputs, with the outputs going high for the corresponding 4-bit input. Similar is the case of a 2x4 decoder except for its 2 inputs and 4 outputs. Assuming all the 2x4 decoders have an enable input, which activates the decoder when the input to it is logic high, 5 such decoders would be required as shown below.	12		



Here, D is the LSB, and A is the MSB. As an example, suppose ABCD = 1100, then the first decoder's output F3 would go high and others low, enabling only bottom-most decoder. The inputs to this decoder is CD = 00, thus its output, F0 goes high. In the same manner other inputs can also be analysed.

ii.



8

UNIT III – PROCESSOR FUNDAMENTALS

1	a.Describe about the ALU design. (12) b.Discuss about Stored programs in processor (8)	20	3	K2
	a.An arithmetic logic unit (ALU) represents the fundamental building block of the central processing unit of a computer. An ALU is a digital circuit used to perform arithmetic and logic operations. What Is an ALU?	12		

An **arithmetic logic unit (ALU)** is a digital circuit used to perform arithmetic and logic operations. It represents the fundamental building block of the **central processing unit (CPU)** of a computer. Modern CPUs contain very powerful and complex ALUs. In addition to ALUs, modern CPUs contain a control unit (CU).

Most of the operations of a CPU are performed by one or more ALUs, which load data from input registers. A **register** is a small amount of storage available as part of a CPU. The control unit tells the ALU what operation to perform on that data, and the ALU stores the result in an output register. The control unit moves the data between these registers, the ALU, and memory.

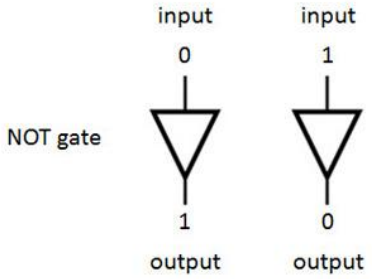
How an ALU Works

An ALU performs basic arithmetic and logic operations. Examples of arithmetic operations are addition, subtraction, multiplication, and division. Examples of logic operations are comparisons of values such as NOT, AND, and OR.

All information in a computer is stored and manipulated in the form of **binary numbers**, i.e. 0 and 1. **Transistor** switches are used to manipulate binary numbers since there are only two possible states of a switch: open or closed. An open transistor, through which there is no current, represents a 0. A closed transistor, through which there is a current, represents a 1.

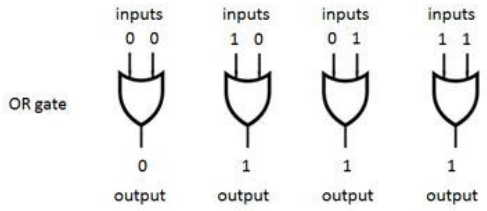
Operations can be accomplished by connecting multiple transistors. One transistor can be used to control a second one - in effect, turning the transistor switch on or off depending on the state of the second transistor. This is referred to as a **gate** because the arrangement can be used to allow or stop a current.

The simplest type of operation is a NOT gate. This uses only a single transistor. It uses a single input and produces a single output, which is always the opposite of the input. This figure shows the logic of the NOT gate:



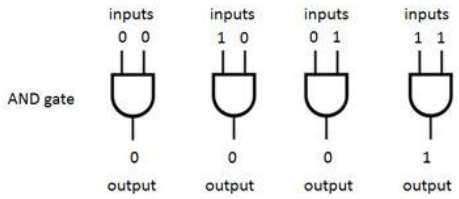
How a NOT gate processes binary data

Other gates consist of multiple transistors and use two inputs. The OR gate results in a 1 if either the first or the second input is a 1. The OR gate only results in a 0 if both inputs are 0. This figure shows the logic of the OR gate:



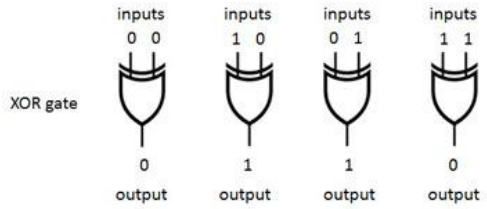
How an OR gate processes binary data

The AND gate results in a 1 only if both the first and second input are 1s. This figure shows the logic of the AND gate:



How an AND gate processes binary data

The XOR gate, also pronounced X-OR gate, results in a 0 if both the inputs are 0 or if both are 1. Otherwise, the result is a 1. This figure shows the logic of the XOR gate:



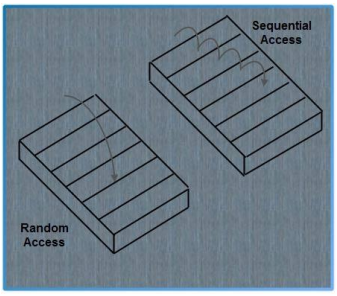
How an XOR gate processes binary data.

The various gates sound a little abstract, but remember that a computer only processes binary data. When you follow the binary logic of these operations, you are starting to think like a computer.

b. A **computer** with a Von Neumann **architecture** stores **program** data and instruction data in the same memory; a **computer** with a Harvard **architecture**has separate memories for **storing program** and data. Both are **stored-program**designs.

8

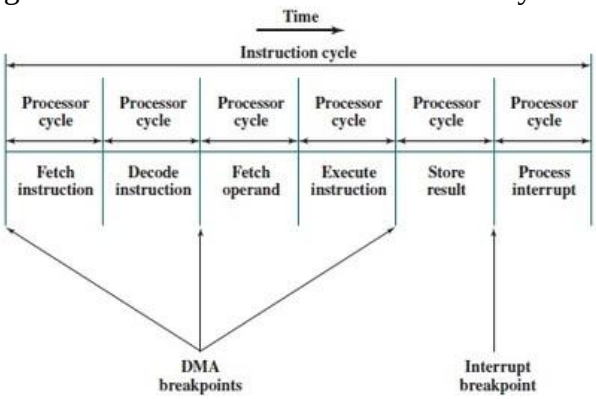
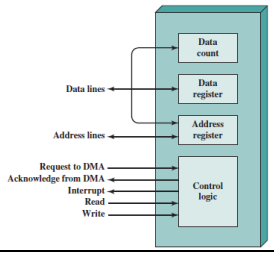
2	**a.Describe the different hazards that cause performance degradation in a pipelined processor. (15)** **b.Define clock rate and instruction count.(5)**	20		
	a. Pipelining is the use of a **pipeline**. Without a **pipeline**, a **computer** processor gets the first instruction from memory, performs the operation it calls for, and then goes to get the next instruction from memory, and so forth. While fetching (getting) the instruction, the arithmetic part of the processor is idle. There are three classes of hazards: • Structural Hazards. They arise from resource conflicts when the hardware cannot support all possible combinations of instructions in simultaneous overlapped execution. • Data Hazards. They arise when an instruction depends on the result of a previous instruction in a way that is exposed by the overlapping of instructions in the pipeline. • Control Hazards.They arise from the pipelining of branches and other instructions that change the PC. b. The **clock rate** typically refers to the frequency at which a chip like a central processing unit (CPU), one core of a multi-core processor, is running and is used as an indicator of the processor's **speed**. It is measured in **clock** cycles per second or its equivalent, the SI unit hertz (Hz). In computer architecture, cycles per **instruction** (aka clock cycles per **instruction**, clocks per **instruction**, or CPI) is one aspect of a processor's performance: the average number of clock cycles per **instruction** for a program or program fragment. It is the multiplicative inverse of **instructions** per cycle.	15 5		
UNIT IV – NEGATIVE FEED BACK AMPLIFIERS				
1	Define addressing mode and Describe the different types of addressing modes	20	4	K2
	Addressing modes are an aspect of the instruction set architecture in most central processing unit (CPU) designs. The various addressing modes that are defined in a given instruction set architecture define how machine language instructions in that architecture identify the operand(s) of each instruction. An addressing mode specifies how to calculate the effective memory address of an operand by using information held in registers and/or constants contained within a machine instruction or elsewhere. In computer programming, addressing modes are primarily of interest			

	to compiler writers and to those who write in assembly languages. **- Immediate Mode:** As the name suggests the instruction in itself contains the operand. **- Register Mode:** In this mode the operands of an instruction are placed in the registers which themselves are placed inside the CPU. **- Direct address mode:** The address part of an instruction in this mode is the effective address. **- Indexed addressing mode:** In this mode in order to obtain the effective address the contents of the index register is added to the instructions address part. **- Relative address mode:** In this mode in order to find out the effective address the contents of the program counter are added to the address part of the instruction.			
2.	**Describe the different methods to access the associative memories.**			
	In computing, an access method is a program or a hardware mechanism that moves data between the computer and an outlying device such as a hard disk (or other form of storage) or a display terminal. The term is sometimes used to refer to the mechanics of placing or locating specific data at a particular place on a storage medium and then writing the data or reading it. It is also used to describe the way that data is located within a larger unit of data such as a data set or file. There are two type of access method random access and sequential access.  The terms random access and sequential access are often used to describe data files. A random-access data file enables you to reador write information anywhere in the file. In a sequential-access file, you can only read and write information sequentially, starting from the beginning of the file. Both types of files have advantages and disadvantages. If you are always accessing information in the same order, a sequential-access file is faster. If you tend to access information randomly, random access is better	20	4	K2

UNIT V – POSITIVE FEED BACK AMPLIFIERS				
1	**Serial and Parallel Transmission in detail**	20		
	Serial and Parallel Transmission Digital data transmission can occur in two basic modes: serial or parallel. Data within a computer system is transmitted via parallel mode on **buses** with the width of the parallel bus matched to the word size of the computer system. Data between computer systems is usually transmitted in **bit serial mode** . Consequently, it is necessary to make a parallel-to-serial conversion at a computer **interface** when sending data from a computer system into a network and a serial-to-parallel conversion at a computer interface when receiving information from a network. The type of transmission mode used may also depend upon distance and required data rate. *Parallel Transmission* In parallel transmission, multiple **bits** (usually 8 bits or a byte/character) are sent simultaneously on different channels (wires, frequency channels) within the same cable, or radio path, and **synchronized** to a clock. Parallel devices have a wider data bus than serial devices and can therefore transfer data in words of one or more bytes at a time. As a result, there is a speedup in parallel transmission bit rate over serial transmission bit rate. However, this speedup is a tradeoff versus cost since multiple wires cost more than a single wire, and as a parallel cable gets longer, the synchronization timing between multiple channels becomes more sensitive to distance. The timing for parallel transmission is provided by a constant clocking signal sent over a separate wire within the parallel cable; thus parallel transmission is considered **synchronous** . Walgreens Photo Prints - Up To 40% Off Photo Prints Get Up To 40% Off All Prints. Use Code GOBIG to Redeem! photo.walgreens.com/Photo/Prints | Sponsored▼ *Serial Transmission* In serial transmission, bits are sent **sequentially** on the same channel (wire) which reduces costs for wire but also slows the speed of transmission. Also, for serial transmission, some overhead time is needed since bits must be assembled and sent as a unit and then disassembled at the receiver. Serial transmission can be either synchronous or **asynchronous** . In synchronous transmission, groups of bits are combined into frames and frames are sent continuously with or without data to be transmitted. In asynchronous transmission, groups of bits are sent as independent units with start/stop flags and no data link synchronization, to allow for arbitrary size gaps between frames. However, start/stop bits maintain physical bit level synchronization once detected. Sato M8480, 8485S/Se Printhead - Is your Printhead China Made?	20	5	K2

Made in USA Printheads. Twice the Life of a Genuine at a Fraction of the Price.
ganson-store.com/USA_Quality/M8485S M8485Se | Sponsored▼

Applications
Serial transmission is between two computers or from a computer to an external device located some distance away. Parallel transmission either takes place within a computer system (on a computer bus) or to an external device located a close distance away.
A special computer chip known as a universal asynchronous receiver transmitter (UART) acts as the interface between the parallel transmission of the computer bus and the serial transmission of the serial port. UARTs differ in performance capabilities based on the amount of on-chip memory they possess.
Examples
Examples of parallel mode transmission include connections between a computer and a printer (parallel printer port and cable). Most printers are within 6 meters or 20 feet of the transmitting computer and the slight cost for extra wires is offset by the added speed gained through parallel transmission of data.
Examples of serial mode transmission include connections between a computer and a modem using the RS-232 **protocol** . Although an RS-232 cable can theoretically accommodate 25 wires, all but two of these wires are for overhead control signaling and not data transmission; the two data wires perform simple serial transmission in either direction. In this case, a computer may not be close to a modem, making the cost of parallel transmission prohibitive—thus speed of transmission may be considered less important than the economical advantage of serial transmission.
Tradeoffs
Serial transmission via RS-232 is officially limited to 20 **Kbps** for a distance of 15 meters or 50 feet. Depending on the type of media used and the amount of external interference present, RS-232 can be transmitted at higher speeds, or over greater distances, or both. Parallel transmission has similar distance-versus-speed tradeoffs, as well as a clocking threshold distance. Techniques to increase the performance of serial and parallel transmission (longer distance for same speed or higher speed for same distance) include using better transmission media, such as **fiber optics** or conditioned cables, implementing repeaters, or using shielded/multiple wires for noise immunity.
Technology
To resolve the speed and distance limitations of serial transmission via RS-232, several other serial transmission standards have been developed including RS-449, V.35, Universal Serial Bus (USB), and IEEE-1394 (Firewire). Each of these standards has different electrical, mechanical, functional, and procedural characteristics. The electrical characteristics define voltage levels and timing of voltage level changes. Mechanical characteristics define the actual connector shape and number of wires.

	Common mechanical interface standards associated with parallel transmission are the DB-25 and Centronics connectors. The Centronics connector is a 36-pin parallel interface that also defines electrical signaling. Functional characteristics specify the operations performed by each pin in a connector; these can be classified into the broad categories of data, control, timing, and electrical ground. The procedural characteristics or protocol define the sequence of operations performed by pins in the connector			
2	**Explain the DMA based data transfer techniques for I/O devices?**	20	5	K2
	Differ from Programmed I/O and Interrupt-Driven I/O, Direct Memory Access is a technique for transferring data within main memory and external device without passing it through the CPU. DMA is a way to improve processor activity and I/O transfer rate by taking-over the job of transferring data from processor, and letting the processor to do other tasks. This technique overcomes the drawbacks of other two I/O techniques which are the time consuming process when issuing command for data transfer and tie-up the processor in data transfer while the data processing is neglected. It is more efficient to use DMA method when large volume of data has to be transferred. For DMA to be implemented, processor has to share its' system bus with the DMA module. Therefore, the DMA module must use the bus only when the processor does not need it, or it must force the processor to suspend operation temporarily. The latter technique is more common to be used and it is referred to as cycle stealing. Figure 5 shows an add-on DMA module cycles in an Instruction Cycle.  Figure 5: DMA and Interrupt Breakpoints during an Instruction Cycle ***Basic Operation of DMA*** When the processor wishes read or send a block of data, it issues a command to the DMA module by sending some information to DMA module. The information includes:  			

Typical DMA Block Diagram

- read or write command, sending through read and write control lines.
- number of words to be read or written, communicated on the data lines and stored in the data count register.
- starting location in memory to read from or write to, communicated on data lines and stored in the address register.
- address of the I/O device involved, communicated on the data lines.

After the information are sent, the processor continues with other work. The DMA module then transfers the entire block of data directly to or from memory without going through the processor. When the transfer is complete, the DMA module sends an interrupt signal to the processor to inform that it has finish using the system bus.

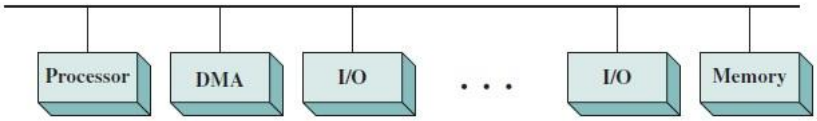
Configurations of DMA

DMA mechanism can be configured in a variety of ways, which are:

- Single-bus, detached DMA
- Single-bus, integrated DMA-I/O
- I/O bus

Single-bus, detached DMA

All modules share the same system bus. The DMA module is acting as a surrogate processor, which uses programmed I/O to exchange data between memory and an I/O module through the DMA module. This configuration is inexpensive, but is inefficient. This is because each transfer of a word consumes two bus cycles.

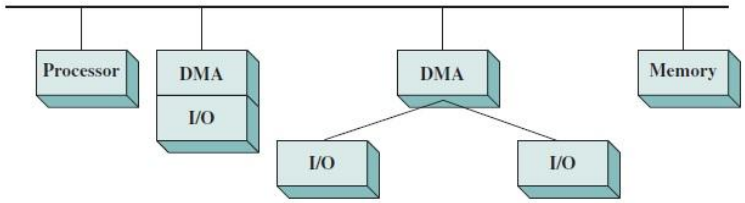


Single-bus, detached DMA

Single-bus, integrated DMA

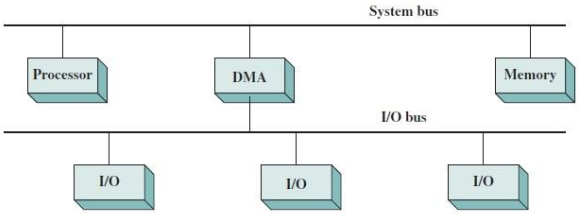
In this configuration, there is a path between the DMA module and one or more I/O module that does not include the system bus. The DMA logic can be a part of an I/O module, or a separate module that controls one or more I/O modules. Therefore, the number of required bus cycles can be cut substantially. The system bus that the DMA module shares with the

processor and memory is used by the DMA module only to exchange data with memory. The exchange of data between the DMA and I/O modules takes place off the system bus.



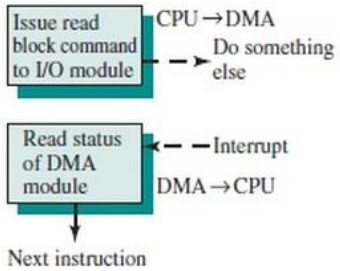
Single-bus, integrated DMA
I/O bus

In this configuration, the concept is further improves from the previous configuration, which is single-bus, integrated DMA. I/O modules are connected to the DMA module using an I/O bus. This can reduce the number of I/O interfaces in the DMA module to one and provides for an easily expandable configuration. The system bus that the DMA module shares with the processor and memory is used by the DMA module only to exchange data with memory. The exchange of data between the DMA and I/O modules takes place off the system bus.



I/O bus

Summary Flows of Using DMA by Processor



1. Issue command to module, by sending nec information to DMA mod
2. Processor does other work
3. DMA acquire control c system, and transfers d and from within memor external device.
4. DMA sends a sign processor when the tran complete, system cont return to processor.

Advantages & Disadvantages of DMA

Advantages	- allows a peripheral device to read from/write to memory without going through the CPU
	- allows for faster processing since the processor can be working on something else while the peripheral can be populating

		memory			
	Disadvantages	- requires a DMA controller to carry out the operation, which increases the cost of the system			
		- cache coherence problems			

Note : 2 Questions with answer key must be prepared in each unit and maximum two sub divisions are allowed.