USN

Internal Assessment Test III – AUG 2022

Sub:	MICROCONTROLLER AND EMBEDDED SYSTEM Sub Code:			18CS44	Branch :	CSE		
Date:	29/08/22	Duration :	90 mins Max Marks: 50 Sem / Sec:	IV Sem A/B/C			OBE	
			Answer any FIVE FULL Questions			ARKS	CO	RBT
1	**Discuss the operational and non-operational quality attributes of an Embedded systems.** Solution: Quality attributes are the non-functional requirements that need to be documented properly in any system design. The Quality attributes of any embedded system are classified into two, namely 1. Operational Quality Attributes 2. Non-Operational Quality Attributes 1. Operational Quality Attributes The operational quality attributes represent the relevant quality attributes related to the embedded system when it is in the operational mode or online mode. The important quality attributes are a. Response b. Throughput c. Reliability d. Maintainability e. Security f. Safety The important Non Operational Attributes quality attributes are a. Testability & Debug-ability b. Evaluability c. Portability d. Time to prototype and market e. Per unit and total cost.					[10] [5] [5]	CO 1	L1
2	a) Explain the fundamental issues in hardware-software co-design. **The following issues are some of the fundamental issues in hardware software co-design.** **1. Selecting the model -** System models are used for capturing and describing the system characteristics. A model is a formal system consisting of objects and composition rules.					[10] [6] [4]	CO 1	L2,L3 Internal Assessment Test III – AUG

It is hard to make a decision on which model should be followed in a particular system design. Most often designers switch between a varieties of models from the requirements specification to the implementation of the system design.

2. Selecting the Architecture - A model only captures the system characteristics and does not provide information on how the system can be manufactured?

The architecture specifies how a system is going to implement in terms of the number and different types of components and the interconnection among them.

Some of the commonly used architectures in system design are

i. The controller architecture - implements the finite state machine model using a state register and two combinational circuits. The state register holds the present state and the combinational circuit's implement the logic for next state and output.

ii. The datapath architecture - is best suited for implementing the data flow graph model where the output is generated as a result of a set of predefined computations on the input data.

A datapath represents a channel between the input and output and in datapath architecture the datapath may contain registers, counters, register files, memories and ports along with high speed arithmetic units.

iii. The Finite State Machine Datapath (FSMD) – this architecture combines the controller architecture with datapath architecture. It implements a controller with datapath.

The controller generatesthe control input whereasthe datapath processesthe data. The datapath contains two types of I/O ports, out of which one acts as the control port for receiving/sending the control signals from/to the controller unit and the second I/O port interfaces the datapath with external world for data input and data output.

iv. The Complex Instruction Set Computing (CISC) - architecture uses an instruction set representing complex operations. It is possible for a CISC instruction set to perform a large complex operation with a single instruction.

The use of a single complex instruction in place of multiple simple instructions greatly reduces the program memory access and program memory size requirement.

v. The Very Long Instruction Word (VLIW) - architecture implements multiple functional units (ALUs, multipliers, etc.) in the datapath. The VLIW instruction packages one standard instruction per functional unit of the datapath.

2022

Sub:

MICROCONTROLLER AND EMBEDDED SYSTEM

Sub Code:

18CS44

Branch:

CSE

Date:

29/08/22

Duration:

90 mins

Max Marks:

50

Sem/Sec:

IV Sem A/B/C

OBE

vi. Parallel processing architecture - implements multiple concurrent Processing Elements (PEs) and each processing element may associate a datapath containing register and local memory.

Single Instruction Multiple Data (SIMD) and **Multiple Instruction Multiple Data (MIMD)** architectures are examples for parallel processing architecture.

In SIMD architecture, a single instruction is executed in parallel with the help ofthe Processing Elements. On the other hand, the processing elements of the MIMD architecture execute different instructions at a given point of time.

3. Selecting the language - A programming language captures a Computational Model and maps it into architecture. Any programing language can be used.

A model can be captured using multiple programming languages like C, C++, C#, Java, etc. for software implementations and for hardware implementations languages like VHDL, System C, Verilog, etc.

On the other hand, a single language can be used for capturing a variety of models. Certain languages are good in capturing certain computational model. For example, C++ is a good candidate for capturing an object oriented model.

4. Partitioning System Requirements into hardware and software - From an implementation perspective, it may be possible to implement the system requirements in either hardware or software (firmware). It is a tough decision making task to figure out which one to opt. various hardware software trade-offs are used for making a decision on the hardware software partitioning.

b) Explain different communication buses used in automotive applications

Automotive applications make use of serial buses for communication, which greatly reduces the amount of wiring required inside a vehicle.

Different types ofserial interface buses deployed in automotive embedded applications are

1. Controller Area Network (CAN) - The CAN bus was originally proposed by **Robert Bosch**. It supports medium speed with data rates up to 125 Kbps and high speed with data rates up to 1Mbps data transfer.

CAN is an event-driven protocol interface with support for error handling in data transmission. It is generally employed in safety system like airbag control; power train systems like engine control and Antilock Brake System (ABS); and navigation systems like GPS etc.

2. Local Interconnect Network (LIN) - LIN bus is a single master multiple slave (up to 16 independent slave nodes) communication interface. LIN is a

Answer any FIVE FULL Questions

MARKS

CO

RBT

1

Discuss the operational and non-operational quality attributes of an Embedded systems.

Solution:

Quality attribute

	low speed, single wire communication interface with support for data rates up to 20 Kbps and is used for sensor/actuator interfacing. LIN bus is employed in applications like mirror controls, fan controls, seat positioning controls, window. Controls, and position controls where response time is not a critical issue. **3. Media-Oriented System Transport (MOST) Bus -** The Media-oriented system transport (MOST) is targeted for automotive audio/video equipment interfacing, used primarily in European cars. A MOST bus is a multimedia fibre-optic point-to-point network implemented in a star, ring' or daisy chained topology over optical fibre cables. The MOST bus-specifications define the physical layer as well as the application layer, network layer, and media access control. MOST bus is an optical fibre cable connected between the Electrical Optical Converter (EOC) and Optical Electrical Converter (OEC), which would translate into the optical cable MOST bus.			s are the non-functional requirements that need to be documented properly in any system design. The Quality attributes of any embedded system are classified into two, namely 1. Operational Quality Attributes

				2. Non-Operational Quality Attributes 1. Operational Quality Attributes The operational quality attributes represent the relevant quality attributes related to the embedded system when it is in the oper

				ational mode or online mode. The important quality attributes are a. Response b. Throughput c. Reliability d. Maintainability e. Security f. Safety The important Non Operational Attributes

				quality attributes are a. Testability & Debug-ability b. Evaluability c. Portability d. Time to prototype and market e. Per unit and total cost. [10] [5] [5] CO1 L1

				2 Explain the fundamental issues in hardware-software co-design. The following issues are some of the fundamental issues in hardware software co-design. 1. Selecting the model - System models are used for capt

				uring and describing the system characteristics. A model is a formal system consisting of objects and composition rules. It is hard to make a decision on which model should be followed in a particular system design. Most

				often designers switch between a varieties of models from the requirements specification to the implementation of the system design. 2. Selecting the Architecture - A model only captures the system characteristics and

				does not provide information on how the system can be manufactured? The architecture specifies how a system is going to implement in terms of the number and different types of components and the interconnection amo

				ng the m. Some of the commonly used architectures in system design are i. The controller architecture - implements the finite state machine model using a state register and two combinational circuits. The state register

				holds the present state and the combinational circuit's implement the logic for next state and output. ii. The data path architecture - is best suited for implementing the data flow graph model where the output is generated as a resul

				t of a set of predefined computations on the input data. A data path represents a channel between the input and output and in data path architecture the data path may contain registers, counters, register files, memories and port

				s alon g with high spee d arith meti c units . iii. The Finit e State Mac hine Data path (FSM D) – this archi tectu re com bine s the cont rolle r archi tectu re with data path archi tectu re. It impl eme nts a cont rolle r with data path. The

				controller generates the control input whereas the data path processes the data. The data path contains two types of I/O ports, out of which one acts as the control port for receiving/sending the control signals from/to the

				controller unit and the second I/O port interfaces the data path with external world for data input and data output. iv. The Complex Instruction Set Computing (CISC) - architecture uses an instruction set representing complex operation

				s. It is possible for a CISC instruction set to perform a large complex operation with a single instruction. The use of a single complex instruction in place of multiple simple instructions greatly reduces the program memory

				access and program memory size requirement. v. The Very Long Instruction Word (VLIW) - architecture implements multiple functional units (ALUs, multipliers, etc.) in the data path. The VLIW instruction packages one stan

				dard instruction per functional unit of the data path. vi. Parallel processing architecture - implements multiple concurrent Processing Elements (PEs) and each processing element may associate a data path containing register

				and local memory. Single Instruction Multiple Data (SIMD) and Multiple Instruction Multiple Data (MIMD) architectures are examples for parallel processing architecture. In SIMD architecture, a single instruction is exec

				uted in parallel with the help ofthe Processing Elements. On the other hand, the processing elements of the MIMD architecture execute different instructions at a given point of time. 3. Selecting the language - A prog

				ramming language captures a Computational Model and maps it into architecture. Any programing language can be used. A model can be captured using multiple programming languages like C, C++, C#, Java, etc. for

				software implementations and for hardware implementations languages like VHDL, System C, Verilog, etc. On the other hand, a single language can be used for capturing a variety of models. Certain languages are good in capt

				uring certain computational model. For example, C++ is a good candidate for capturing an object oriented model. 4. Partitioning System Requirements into hardware and software - From an implementation perspective, it may

				be possible to implement the system requirements in either hardware or software (firmware). It is a tough decision making task to figure out which one to opt. various hardware software trade-offs are used for making a decis

				ion on the hard ware soft ware parti tioni ng. Expl ain diffe rent com muni catio n buse s used in auto moti ve appli catio ns Auto moti ve appli catio ns mak e use of seria l buse s for com muni catio n, whic h great ly redu ces

				the amount of wiring required inside a vehicle. Different types ofserial interface buses deployed in automotive embedded applications are 1. Controller Area Network (CAN) - The CAN bus was originally proposed

				by Robert Bosch. It supports medium speed with data rates up to 125 Kbps and high speed with data rates up to 1Mbps data transfer. CAN is an event-driven protocol interface with support for error handling in data transmission.

				It is generally employed in safety system like airbag control; power train systems like engine control and Antilock Brake System (ABS); and navigation systems like GPS etc. 2. Local Interconnect Network (LIN) - LIN bus

				is a single master multiple slave (up to 16 independent slave nodes) communication interface. LIN is a low speed, single wire communication interface with support for data rates up to 20 Kbps and is used for sensor/actuator inter

				facing. LIN bus is employed in applications like mirror controls, fan controls, seat positioning controls, window. Controls, and position controls where response time is not a critical issue. 3. Media-Oriented System

				Transport (MOST) Bus - The Media-oriented system transport (MOST) is targeted for automotive audio/video equipment interfacing, used primarily in European cars. A MOST bus is a multimedia fibre-optic point-to-poin

				t netw ork impl eme nted in a star, ring’ or daisy chai ned topo logy over optic al fibre cabl es. The MOS T bus-speci ficati ons defin e the physi cal layer as well as the appli catio n layer , netw ork layer , and medi a acce ss cont rol.

				MOST bus is an optical fibre cable connected between the Electrical Optical Converter (EOC) and Optical Electrical Converter (OEC), which would translate into the optical cable MOST bus. [10] [6]

				[4] CO1 L2,L3 3 With FSM model, discuss the design and operation of automatic tea/coffee vending machine. Solution: OBJ The FSM representation for the above requirement is shown

				in the below figure. It contains four states namely i. Wait for coin ii. Wait for User Input iii. Dispense Tea iv. Dispense Coffee. The event ‘Insert Coin’ (5 rupee coin insertion), transitions the state

				to ‘Wait for User Input’. The system stays in this state until a user input is received from the buttons ‘Cancel’, ‘Tea’ or ‘Coffee’. If the event triggered in ‘Wait State’ is ‘Cancel’ button press, the coin is pushed out

				and the state “transitions to ‘Wait for Coin’. If the event received in the ‘Wait State’ is either “Tea’ button press, or ‘Coffee’ button press, the state changes to ‘Dispense Tea’ and ‘Dispense Coffee’ respectively. Once

				the coffee/tea vending is over, the respective states transition back to the ‘Wait for Coin’ state. A few modifications like adding a time out for the ‘Wait State’ (Currently the ‘Wait State’ is infinite; it can be re-designed to

				a time out based ‘Wait State’. If no user input is received within the time out period, the coin is returned back and the state automatically transitions to ‘Wait for Coin’ on the time out event) and capturing other events like, ‘Wat

				er not available’, ‘Tea/ Coffee Mix not available’ and changing the state to an ‘Error State’ can be added to enhance this design. [10] [4] [6] CO1, CO2 L2,L3 4 Define Process. Explain in detai

				l the structure, memory organization and state transition of the process. Solution: Process A ‘Process’ is a program, or part of it, in execution. Process also known as an instance of a program in execution. Mult

				iple instances of the same program can execute simultaneously. A process requires various system resources like CPU for executing the process; memory for storing the code corresponding to the process and associate

				d variables, I/O devices for information exchange, etc. The Structure of a Process The concept of 'Process' leads to concurrent execution (pseudo parallelism) of tasks and thereby the efficient utilisation of the

				CPU and other system resources. Concurrent execution is achieved through the sharing of CPU among the processes. A process mimics (imitate) as a processor in properties and holds a set of registers, proc

				ess statu s, a Prog ram Coun ter (PC) to poin t to the next exec utabl e instr uctio n of the proc ess, a stack for holdi ng the local varia bles asso ciate d with the proc ess and the code corr espo ndin g to the proc ess. This can be

				visualised in below figure A process which inherits all-the properties of the CPU can be considered as a virtual processor, awaiting its turn to have its properties switched into the physical processor.

				When the process gets its turn, its registers and the program counter register mapped to the physical registers of the CPU. A process which inherits all-the properties of the CPU can be considered as a virtual

				processor, awaiting its turn to have its properties switched into the physical processor. When the process gets its turn, its registers and the program counter register mapped to the physical registers of the

				CPU. rom a mem ory pers pecti ve, the mem ory occu pied by the proc ess is segr egat ed into thre e regio ns, nam ely, Stac k mem ory, Data mem ory and Code mem orya s sho wn in abov e figur e. The ‘Stac k’

				memory holds all temporary data such as variables local to the process. Data memory holds all global data for the process. Process States and State Transition The creation of a process to its termination is not a single

				step operation. The process traverses through a series of states during its transition from the newly created state to the terminated state. The cycle through which a process changes its state from “newly created’

				to ‘execution completed’ is known as ‘Process Life Cycle’. The various states through which a process traverses during a Process Life Cycle indicates the current status of the process with respect to time and

				also provides information on what it is allowed to do next. - Below figure represents the various states associated with a process. he state at which a process is being created is referred as ‘Created

				State’. The Operating System recognises a process in the ‘Created State’ but no resources are allocated to the process. The state, where a process is incepted into the memory and awaiting the processor time

				for execution, is known as 'Ready State'. At this stage, the process is placed in the 'Ready list' queue maintained by the OS. The state where the source code instructions corresponding to the process is being

				executed is called ‘Running State’. At which the process execution happens. Blocked State/Wait State refers to a state where a running process is temporarily suspended from execution and does not have immediate

				acce ss to reso urce s. The bloc ked state migh t be invo ked by vario us cond ition s like: the proc ess ente rs a wait state for an even t to occu r (e.g. Wait ing for user inpu ts such as keyb oard inpu t) or waiti ng for getti

				ng acce ss to a shar ed reso urce A state wher e the proc ess com plete s its exec utio n is kno wn as ‘Com plete d State ’. The trans ition of a proc ess from one state to anot her is kno wn as ‘Stat e trans ition’ . Whe n a

				process changes its state from Ready to running or from running to blocked or terminated or from blocked to running, the CPU allocation for the process may also change. [10] [4] [6]

				CO2 L4 5 With a neat diagram, explain operating system architecture. Solution: The Operating System (OS) acts as a bridge between the user applications/ tasks and the underlying system resources through

				a set of system functionalities and services. » The primary functions of an operating systems are – » Make the system convenient to use » Organize and manage the system resources efficiently and correctly.

				The following Figure gives an insight into the basic components of an operating system and their interfaces with rest of the world. The Kernel: » The kernel is the core

				of the operating system. It is responsible for managing the system resources and the communication among the hardware and other system services. » Kernel acts as the abstraction layer between system reso

				urces and user applications. » Kernel contains a set of system libraries and services. » For a general purpose OS, the kernel contains different services like memory management, process management, time

				management, file system management, I/O system management. Process Management: deals with managing the process/ tasks. » Process management includes – » setting up a memory for

				the process » loading process code into memory » allocating system resources » scheduling and managing the execution of the process » setting up and managing Process Control

				Block (PCB) » interprocess communication and synchronization » process termination/ deletion, etc. Primary Memory Management: refers to a volatile memory (RAM), where processe

				s are loaded and variables and shared data are stored. » The Memory Management Unit (MMU) of the kernel is responsible for – » Keeping a track of which part of the memory area is currently used by

				which process » Allocating and De-allocating memory space on a need basis. File System Management: File is a collection of related information. A file could be a program (source code or executable), text files,

				image files, word documents, audio/ video files, etc. » A file system management service of kernel is responsible for – » The creation, deletion and alteration of files » Creation, deletion, and alteration of

				directories » Saving of files in the secondary storage memory » Providing automatic allocation of file space based on the amount of free running space available » Providing flexible naming

				conversion for the files. I/O System (Device) Management: Kernel is responsible for routing the I/O requests coming from different user applications to the appropriate I/O devices of the system. » In a well structured

				OS, direct access to I/O devices is not allowed; access to them is established through Application Programming Interface (API). » The kernel maintains list of all the I/O devices of the system. » „Device

				Manager " of the kernel is responsible for handling all I/O related operations. » The Device Manager is responsible for – » Loading and unloading of device drivers » Exchanging information

				and the system specific control signals to and from the device. Protection Systems: Modern operating systems are designed in such way to support multiple users with different levels of access permissions. » The prot

				ectio n deal s with impl eme nting the secu rity polic ies to restr ict the acce ss of syste m reso urce s and parti cular user by diffe rent appli catio n or proc esse s and diffe rent user. [10] [4] [6] CO2 L1,L3

				6 Illustrate the concept of “deadlock” with a neat diagram. Mention the different conditions favor a deadlock situation. Solution: Deadlock is the condition in which a process is waiting for a resource held by

				another process which is waiting for a resource held by the first process; hence, none of the processes are able to make any progress in their execution. Process A holds a resource „x" and it wants a resource „y" held by

				Process B. Process B is currently holding resource „y" and it wants the resource „x" which is currently held by Process A. Both hold the respective resources and they compete each other to get the resource held by the respectiv

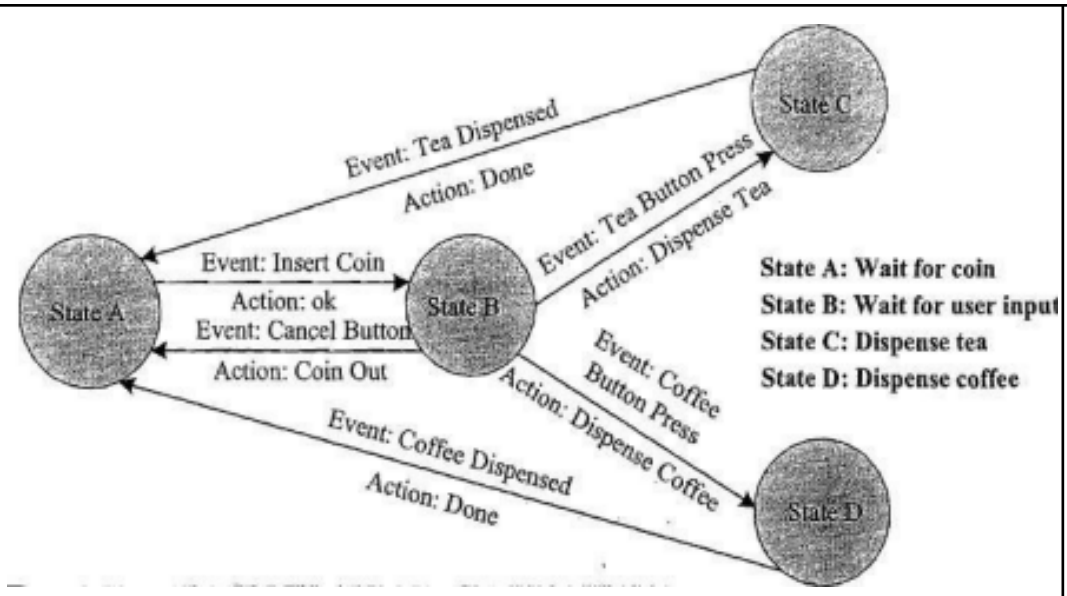
				e processes. Conditions Favoring Deadlock: Mutual Exclusion: The criteria that only one process can hold a resource at a time. Meaning processes should access shared resources with mutual exclusion.

				Hold & Wait : The condition in which a process holds a shared resource by acquiring the lock controlling the shared access and waiting for additional resources held by other processes. No Resource Preemption: The crite

				ria that Operating System cannot take back a resource from a process which is currently holding it and the resource can only be released voluntarily by the process holding it. Circular Wait : A process is waiting for a resource whic

				h is currently held by another process which in turn is waiting for a resource held by the first process. In general there exists a set of waiting process P0, P1 …. Pn with P0 is waiting for a resource held by P1 and P1 is waiti

				ng for a reso urce held by P0, ……,Pn is waiti ng for a reso urce held by P0 and P0 is waiti ng for a reso urce held by Pn and [10] [3] [3] [4] CO2 L4
3	With FSM model, discuss the design and operation of automatic tea/coffee vending machine. Solution:	[10] [4] [6]	CO 1, CO 2	L2,L3



The FSM representation for the above requirement is shown in

the below figure. It contains four states namely

i. Wait for coin
ii. Wait for User Input
iii. Dispense Tea
iv. Dispense Coffee.

The event 'Insert Coin' (5 rupee coin insertion), transitions the state to 'Wait for User Input'. The system stays in this state until a user input is received from the buttons 'Cancel', 'Tea' or 'Coffee'.
If the event triggered in 'Wait State' is 'Cancel' button press, the coin is pushed out and the state "transitions to 'Wait for Coin'.

If the event received in the 'Wait State' is either " Tea' button press, or 'Coffee' button press, the state changes to 'Dispense Tea' and 'Dispense Coffee' respectively.

Once the coffee/tea vending is over, the respective states transition back to the 'Wait for Coin' state.

A few **modifications** like adding a timeout for the 'Wait State' (Currently the 'Wait State' is infinite; it can be re-designed to a timeout based 'Wait State'. If no user input is received within the timeout period, the coin is returned back and the state automatically transitions to 'Wait for Coin' on the timeout event) and capturing other events like, 'Water not available', 'Tea/Coffee Mix not available' and changing the state to an 'Error State' can be added to enhance this design.

4	**Define Process. Explain in detail the structure, memory organization and state transition of the process.** Solution: Process A 'Process' is a program, or part of it, in execution. Process also known as an instance of a program in execution. Multiple instances of the same program can execute simultaneously. A process requires various system resources like CPU for executing the process; memory for storing the code corresponding to the process and associated variables, I/O devices for information exchange, etc. The Structure of a Process The concept of 'Process' leads to concurrent execution (pseudo parallelism) of tasks and thereby the efficient utilisation of the CPU and other system resources. Concurrent execution is achieved through the sharing of CPU among the processes. A process mimics (imitate) as a processor in properties and holds a set of registers, process status, a Program Counter (PC) to point to the next executable instruction of the process, a stack for holding the local variables associated with the process and the code corresponding to the process. This can be visualised in below figure A process which inherits all-the properties of the CPU can be considered as a virtual processor, awaiting its turn to have its properties switched into the physical processor. When the process gets its turn, its registers and the program counter register mapped to the physical registers of the CPU. A process which inherits all-the properties of the CPU can be considered as a virtual processor, awaiting its turn to have its properties switched into the physical processor. When the process gets its turn, its registers and the program counter register mapped to the physical registers of the CPU. rom a memory perspective, the memory occupied by the process is segregated into three regions, namely, Stack memory, Data memory and Code memoryas shown in above figure. The 'Stack' memory holds all temporary data such as variables local to the process. Data memory holds all global data for the process. Process States and State Transition The creation of a process to its termination is not a single step operation. The process traverses through a series of states during its transition from the newly created state to the terminated state. The cycle through which a process changes its state from "newly created' to 'execution completed' is known as 'Process Life Cycle'. The various states through which a process traverses during a Process Life Cycle indicates the current status of the process with respect to time and also provides information on what it is allowed to do next. - Below figure represents the various states associated with a process. he state at which a process is being created is referred as 'Created State'. The Operating System recognises a process in the 'Created State' but no resources are allocated to the process. The state, where a process is incepted into the memory and awaiting the processor time for execution, is known as 'Ready State'. At this stage, the process is placed in the 'Ready list' queue maintained by the OS. The state where the source code instructions corresponding to the process is being executed is called 'Running State'. At which the process execution happens. Blocked State/Wait State refers to a state where a running process is temporarily suspended from execution and does not have immediate access to resources. The blocked state might be invoked by various conditions like: the process enters a	[10] [4] [6]	CO 2	L4

	wait state for an event to occur (e.g. Waiting for user inputs such as keyboard input) or waiting for getting access to a shared resource A state where the process completes its execution is known as 'Completed State'. The transition of a process from one state to another is known as 'State transition'. When a process changes its state from Ready to running or from running to blocked or terminated or from blocked to running, the CPU allocation for the process may also change.			
5	**With a neat diagram, explain operating system architecture.** **Solution:** The ***Operating System (OS)*** acts as a bridge between the user applications/ tasks and the underlying system resources through a set of system functionalities and services. » The primary functions of an operating systems are – » Make the system convenient to use » Organize and manage the system resources efficiently and correctly. The following Figure gives an insight into the basic components of an operating system and their interfaces with rest of the world. ***<u>The Kernel:</u>*** » The *kernel* is the core of the operating system. It is responsible for managing the system resources and the communication among the hardware and other system services. » Kernel acts as the abstraction layer between system resources and user applications. » Kernel contains a set of system libraries and services. » For a general purpose OS, the kernel contains different services like memory management, process management, time management, file system management, I/O system management. ***Process Management:*** deals with managing the process/ tasks. » Process management includes – » setting up a memory for the process » loading process code into memory » allocating system resources » scheduling and managing the execution of the process » setting up and managing Process Control Block (PCB) » inter process communication and synchronization » process termination/ deletion, etc. ***Primary Memory Management:*** refers to a volatile memory (RAM), where processes are loaded and variables and shared data are stored. » The Memory Management Unit (MMU) of the kernel is responsible for – » Keeping a track of which part of the memory area is currently used by which process » Allocating and De-allocating memory space on a need basis. ***File System Management:*** *File* is a collection of related information. A file could be a program (source code or executable), text files, image files, word documents, audio/ video files, etc. » A file system management service of kernel is responsible for – » The creation, deletion and alteration of files » Creation, deletion, and alteration of directories » Saving of files in the secondary storage memory » Providing automatic allocation of file space based on the amount of free running space available » Providing flexible naming conversion for the files. ***I/O System (Device) Management:*** Kernel is responsible for routing the I/O requests coming from different user applications to the appropriate I/O devices of the system.	[10] [4] [6]	CO 2	L1,L3

	» In a well structured OS, direct access to I/O devices is not allowed; access to them is establish through Application Programming Interface (API). » The kernel maintains list of all the I/O devices of the system. » „*Device Manager*“ of the kernel is responsible for handling all I/O related operations. » The Device Manager is responsible for – » Loading and unloading of device drivers » Exchanging information and the system specific control signals to and from the device. ***Protection Systems:*** Modern operating systems are designed in such way to support multiple users with different levels of access permissions. » The *protection* deals with implementing the security policies to restrict the access of system resources and particular user by different application or processes and different user.			
6	Illustrate the concept of “deadlock” with a neat diagram. Mention the different conditions favor a deadlock situation. Solution: *Deadlock* is the condition in which a process is waiting for a resource held by another process which is waiting for a resource held by the first process; hence, none of the processes are able to make any progress in their execution. Process A holds a resource „*x*“ and it wants a resource „*y*“ held by Process B. Process B is currently holding resource „*y*“ and it wants the resource „*x*“ which is currently held by Process A. Both hold the respective resources and they compete each other to get the resource held by the respective processes. *Conditions Favoring Deadlock:* ▫ *Mutual Exclusion:* The criteria that only one process can hold a resource at a time. Meaning processes should access shared resources with mutual exclusion. ▫ *Hold & Wait:* The condition in which a process holds a shared resource by acquiring the lock controlling the shared access and waiting for additional resources held by other processes. ▫ *No Resource Preemption:* The criteria that Operating System cannot take back a resource from a process which is currently holding it and the resource can only be released voluntarily by the process holding it. ▫ *Circular Wait:* A process is waiting for a resource which is currently held by another process which in turn is waiting for a resource held by the first process. In general there exists a set of waiting process P0, P1 …. Pn with P0 is waiting for a resource held by P1 and P1 is waiting for a resource held by P0, ……,Pn is - waiting for a resource held by P0 and P0 is waiting for a resource held by Pn and	[10] [3] [3] [4]	CO 2	L4