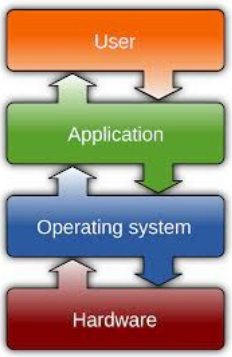
MODULE IV
Operating Systems

CO4	Familiarize RTOS		
M4.01	Describe Kernel,Operating System Architecture and types	2	Understanding
M4.02	ExplainRTOS Kernel functions	3	Understanding
M4.03	ListSelection criteria for RTOS	2	Remembering
M4.04	Outline Micro C/OS-II and its services	2	Understanding
M4.05	List popular Real Time Operating Systems.	1	Remembering

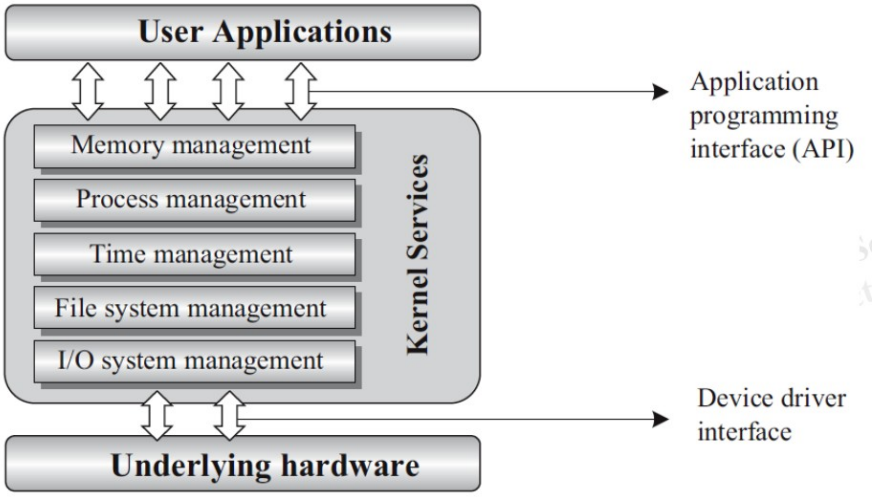
Syllabus Content:
Kernel, types of operating systems- GPOS, RTOS.
Real time operating systems:- Tasks, process, threads, multiprocessing and multi-tasking, task scheduling, types, threads and process scheduling, task communication, task synchronization, device drivers.
List Selection criteria for RTOS.
Overview of Micro C/OS-II and its services.
List popular Real Time Operating Systems (any 10).

Operating System(OS): An operating system is a program that controls the execution of application programs and acts as an interface between applications and the computer hardware.
The primary functions of an operating systems is:

- Makes the system convenient to use
- Organize and manage the system resources efficiently and correctly



General Operating System Architecture:



Kernel: A kernel is a central component of an operating system that acts as an interface between the user applications and the hardware. It contains a set of system libraries & services. For a general purpose OS, the kernel performs following functions:

1) Process Management:

- A multitasking OS allows the computer to perform many different tasks (processes) at the same time. The CPU can only focus on one process at a time. The OS tells the CPU to give each process a time slot.
- Process management deals with managing the processes/tasks. Process management includes:
  - Setting up the memory space for the process
  - Loading the process's code into the memory space
  - Scheduling and managing the execution of the process
  - Synchronization, process termination/deletion, etc.

2) I/O System (Device) Management:

- To perform various actions, processes require access to peripheral devices such as a mouse, keyboard, etc., that are connected to the computer.
- A kernel must maintain a list of available devices. This list may be known in advance configured by the user or detected by the operating system at run time (normally called plug and play).
- In a plug and play system, a device manager first performs a scan on different hardware buses, such as Peripheral Component Interconnect (PCI) or Universal Serial Bus (USB), to detect installed devices, then searches for the appropriate drivers.

3) File System Management:

- Files is a collection of co-related information. It is recorded in some format such as text, pdf, docs, etc & stored on various storage mediums such as flash drives, hard disk drives (HDD), magnetic tapes, optical disks, and tapes, etc.
- The file management in operating system is nothing but software that handles or manages the files (binary, text, pdf, docs, audio, video, etc.) present in computer software.

- The file system in the operating system is capable of managing individual as well as groups of files present in the computer system.
- The file management in the operating system manages all the files present in the computer system with various extensions (such as .exe, .pdf, .txt, .docx, etc.).
- The file system in the operating system help to get details of any file(s) present on our system. The details can be location & Creator of the file, time of file creation and modification, file format etc.

4) Memory Management:

- A computer has different memory types it could use; Registers, Cache, ROM & RAM, Disk storage. The OS is responsible for checking:
  - Which memory if free.
  - Which memory is to be allocated and de-allocated.
  - How to swap between the main memory and secondary memory.
  - Keeps tracks of primary memory.

5) Time Management: Time management provide precise time synchronization for all the applications. These services include:

- Measuring the time used by various activities and processes
- Starting and stopping activities at specific times

Q) Write in details the general architecture of embedded operating system. (7 marks)
Q) What is kernel of an embedded operating system. (1 marks)
Q) Outline kernel in an operating system. (7 marks)

Kernel Space: The memory location at which the kernel code is located is known as kernel space.

User space: It is the memory location where the user applications are loaded & executed.

Types of Kernel:

1) Monolithic Kernel: In Monolithic kernel, the entire operating system runs as a single program in kernel mode. The user services and kernel services uses the same memory space. OS is easy to design and implement. Failure of one component in a monolithic kernel leads to the failure of the entire system. Eg LINUX, MS-DOS, WINDOWS.

2) Micro Kernel: In microkernel the user services and kernel services are implemented in different memory space. OS is complex to design. Failure of one component does not effect the working of entire system. Eg Mach.

Types of Operating system:

General Purpose Operating System (GPOS)	**Real-Time operating system(RTOS)**
• GPOS are operating systems deployed in general computing systems (Desktops, notebooks, smartphones).	• RTOS are operating system employed in embedded system.
• Time response of GPOS is not deterministic.	• Time response of RTOS is deterministic (highly predictable)i.e RTOS decides

	which applications should run in which order and how much time needs to be allocated for each application.
• The GPOS has no task deadline.	• The RTOS has task deadline.
• In the case of a GPOS, the scheduling of task is not based on "priority" always.	• Whereas in RTOS – scheduling is always priority based.
• GPOS kernel is more complex.	• RTOS kernel is comparatively less complex.
• It have large memory.	• It doesn't have large memory.
• It does not optimize the memory resources.	• It optimizes memory resources.
• GPOS is designed for a m ulti-user environment.	• RTOS is designed & developed for a single-user environment.
• Eg: Linux, Windows etc.	• Eg: Windows Embedded Compact, QNX etc.

The Real-Time Kernel: The kernel of RTOS is refered as real-time kernel. The basic functions of a Real-Time kernel are listed below:

- **Task/Process management:** Deals with setting up the memory space for the task, loading task code to memory, allocating system resources, setting up Task Control Block(TCB), termination of a task. TCB is used for holding the information corresponding to a task.

- **Task/Process scheduling:** Deals with sharing the CPU among the various task. A kernel application called 'Scheduler' handles the task scheduling. Scheduler is actually an algorithm implementation which perform the efficient & optimal scheduling of tasks.

- **Task/Process synchronisation:** It enables the tasks to mutually share the resources like buffers, I/O devices etc.

- **Error/Exception handling:** Deals with handling errors/ exceptions raised during the execution of tasks like insufficient memory, timeouts, bus error, divide by zero etc..

- **Memory management:** RTOS uses a block based memory allocation technique, instead of a dynamic memory allocation in GPOS. RTOS kernel uses memory blocks of fixed size & is allocated for a task on a need basis. These blocks are stored in ' Free Buffer Queue'.

- **Interrupt handling:** Deals with handling various types of interrupts. It can be:
    - **Synchronous interrupt:** They occur in sync with currently executing task. Usually software interrupt fall into this category.
    - **Asynchronous interrupt:** They occur not in sync with currently executing task & are usually generated by external devices.

- **Time management:** Deals with accurate time management of RTOS. It is provided with a high resolution Real-Time Clock(RTC) hardware chip & is programmed to interrupt the processor at a fixed rate. This timer interrupt is called 'Timer Tick'.

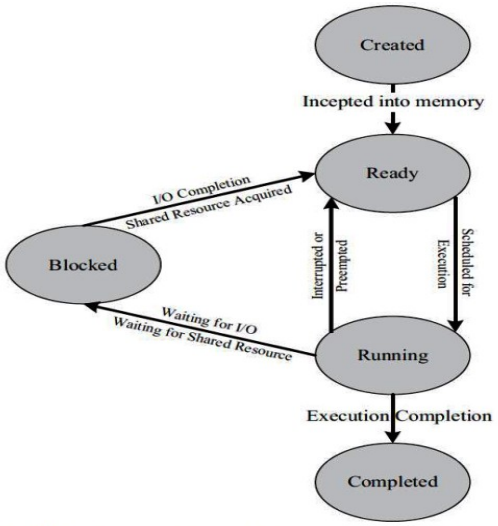
Q) Explain any three functions of RTOS Kernel. (3marks)

Task: A task is defined as the program in execution and the related information maintained by the operating system for the program.

Process: It is a program or part of it, in execution. It is also known as an instance of a program in execution. A process is sequential in execution.

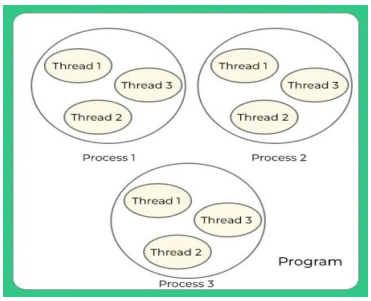
NOTE: In Real Time Applications the terms **'Process', 'Task', 'Job'** refers to same entity in operating system context & most often they are used interchangeably.

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'**.
- **Created State:** The state at which a process is being created is referred as 'Created State'. The Operating System recognizes a process in the 'Created State' but no resources are allocated to the process.
- **Ready State**: The state, where a process is launched into the memory and awaiting the processor time for execution, is known as 'Ready State'.
- **Running State:** The state where in the source code instructions corresponding to the process is being executed is called 'Running State'. Running state is the state at which the process execution happens.
- **Blocked State/Wait State:** It refers to a state where a running process is temporarily suspended from execution and does not have immediate access to resources.
- **Completed State:** A state where the process completes its execution is known as 'Completed State'.
- **State transition:** The transition of a process from one state to another is known as 'State transition'.

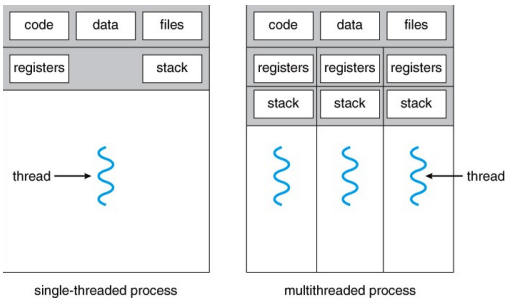
Thread: A thread is a single sequential flow of control within a process. Thread is a segment of a process that is managed by the scheduler independently. It is the smallest executable unit of a process. It is also known as **'lightweight process'**. A process can have many threads of execution. All the threads within one process are interrelated to each other & shared same memory space.



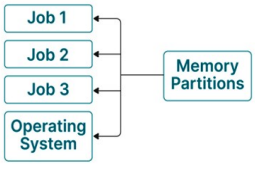
Thread	Process
• Thread is a single unit of execution & is a part of process. It is the smallest executable unit of a process.	• Process is a program or a part of it in execution & contains one or more threads
• A thread does not have data, code & stack memory. It shares the memory within other threads of same process.	• Process has its own data, code & stack memory.
• It takes less time for creation & termination of a thread.	• It takes more time for creation & termination of a process.
• If any user-level thread gets blocked, all of its peer threads also get blocked because OS takes all of them as a single process.	• If one process gets blocked by the operating system, then the other process can continue the execution.
• Inter thread communication is faster.	• Inter process communication is slow.

Single threaded & Multi threaded process:

- A **single-threaded process** is a process with a single thread.
- A **multi-threaded process** is a process with multiple threads.

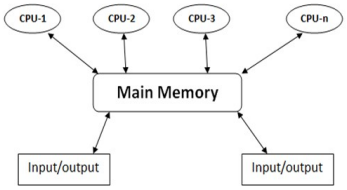


Multiprogramming: It is the ability of an operating system to hold multiple programs in the main memory at the same time ready for execution.



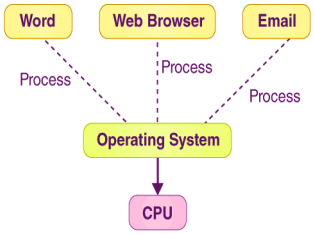
Multiprocessing: It is the ability of an operating system to execute multiple processes/tasks simultaneously. Systems which are capable of performing multiprocessing are known as **multiprocessor systems**. Multiprocessor systems possess **multiple CPUs** and can execute multiple

processes simultaneously.



Q) ……. in an operating systems allows a user to perform more than one task at a time. (1 mark)

Multitasking: The ability of an operating system to hold multiple processes/tasks in memory and switch the processor (CPU) from executing one process to another process is known as **multitasking**. Multitasking involves the switching of CPU from executing one task to another.



Three process involved in multitasking(CPU switching):
1) Context switching: The process of switching CPU among the processes
2) Context saving: The process of saving the details(Register details, memory details, system resource usage details, execution details etc) of a currently running process at the time of CPU switching.
3) Context retrieval: The process of retrieving details of a process, which is going to be executed due to CPU switching.

Types of Multitasking: Depending on how switching act is implemented, multitasking can be classified as follows:
1) Cooperative Multitasking(Non-Preemptive Multitasking): It is a type of multitasking in which the operating system never initiates a switching from a running process to another process, instead a process must voluntarily yield control of the processor from a currently executing process.
2) Preemptive Multitasking: In this type of multitasking, the operating system allow each process to get a regular amount of CPU time.

Task scheduling: Multitasking involves the execution switching among different tasks. Determining which task/process is to be executed at a given point of time is known as task scheduling. The kernel service which implements the scheduling algorithm is known as **'Scheduler'**.
The process of scheduling decision may takes place when a process switches its state to:
1) 'Ready' state from 'Running' state
2) 'Blocked/Wait' state from 'Running' state
3) 'Ready' state from 'Blocked/Waiting' state
4) 'Completed' state

Factors to be considered while selecting scheduling algorithm/criterion:

The selection of a scheduling criterion/algorithm should consider the following factors:

- **CPU Utilization:** It is a direct measure of how much percentage of the CPU is being utilized. Scheduling algorithms should always make the CPU utilisation high
- **Throughput:** It gives an indication of the number of processes executed per unit of time. Throughput of a good scheduler should always be high.
- **Turnaround Time:** It is the amount of time taken by a process for completing its execution. It should be minimal for a good scheduling algorithm**.**
- **Waiting Time:** It is the amount of time spent by a process in the 'Ready' queue waiting to get the CPU time for execution. The waiting time should be minimal for a good scheduling algorithm.
- **Response time:** Time elapsed between the submission of a process and the first response. It should be as least as possible for a good scheduling algorithm.

Various queues associated with CPU task scheduling:

- **Job Queue:** Contains all the processes in the system
- **Ready queue:** Contains all the processes which are ready for execution and waiting for CPU to get their turn for execution. Ready queue will be empty when there is no process ready for running.
- **Device Queue**: Contains set of processes which are waiting for an I/O device.

Types of task scheduling: Based on scheduling algorithm, the scheduling can be classified as:

1) Non preemptive scheduling:

- Employed in systems which implement non-preemptive multitasking.
- The currently executing task/process is allowed to run until it terminates or enters the 'Wait' state waiting for an I/O or system source.

Various types of non-preemptive scheduling:

- **First come First Served(FCFS)/FIFO scheduling:** Allocates CPU time to processes based on the order in which they enter the 'Ready' queue. First entered process is served first.
- **Last Come First Served(LCFS)/LIFO Scheduling:** Allocates CPU time to the processes based on the order in which they are entered in the 'Ready' queue. Last entered process is served first.
- **Shortest Job First(SJF)scheduling:** Sorts the 'Ready' queue each time to pick the process with shortest estimated completion/run time. In SJF, the process with shortest estimated run time is scheduled first, followed by the next shortest process and so on.
- **Priority based scheduling:** A process with high priority is serviced first in the 'Ready' queue. One way of priority assigning is giving priority to the task/process at the time of creation of the task/process. SJF(shortest job first) algorithm can be viewed as a priority scheduling where each task is prioritised in the order of the time required to complete the task.

2) Preemptive Scheduling:

- Employed in systems which implement preemptive multitasking.
- In this scheduling every task in the 'Ready' queue gets a chance to execute.
- When and how often each process gets a chance to execute depend on the type of preemptive scheduling algorithm called 'Scheduler' used for scheduling processes.
- Scheduler can preempt(stop temporarily) the currently executing task and select another task from the 'Ready' queue for execution.

- A task which is preempted by the scheduler is moved to the 'Ready' queue. The act of moving a 'Running' process/task into the 'Ready' queue by the scheduler without the processes requesting for it is known as 'Preemption'.

Types of Preemptive Scheduling:

- **Preemptive SJF Scheduling/Shortest Remaining Time (SRT):** The Preemptive SJF scheduling algorithm sorts the 'Ready' queue when a new process enters the 'Ready' queue and checks whether the execution time of the new process is shorter than the remaining of the total estimated time for the currently executing process. If the execution time of the new process is less, currently executing process is pre-empted and the new process is scheduled for execution.
  The Non-Preemptive SJF scheduling algorithm sorts the 'Ready' queue only after completing the execution of the current process or when the process enters 'Wait' state.
- **Round Robin** (**RR**) **Scheduling**: Means 'Equal chance to all'. Each process in the 'Ready' queue is executed for a predefined time slot. Execution starts with picking up the first process in 'Ready' queue. Its executed for a predefined time and when the predefined time elapses or the process complete(before the predefined time slice), the next process in the 'Ready' queue is selected for execution. This is repeated for all the process in the 'Ready' queue.
- **Priority based Scheduling:** Priority based Preemptive Scheduling is same as Priority based Non-premptive Scheduling except for the switching of execution between tasks. In Preemptive scheduling, any high priority process entering the 'Ready' queue is immediately scheduled for execution whereas in non preemptive any high priority process entering the 'Ready' queue is scheduled only after the currently executing process completes its execution or only when it voluntarily relinquishes the CPU.

Task Communication: The mechanism through which processes/task communicate each other is known as **Inter Process/Task Communication(IPC)**. It is essential for process coordination.
Various IPC mechanisms adopted by Kernel

- **Shared Memory:** Processes share some area of memory to communicate among them. Information to be communicated by the process is written to this shared memory area. Other process which require this information can read the same from the shared memory area. Implementation of shared memory can be done through **Pipes & Memory mapped Objects**.
- **Message Passing:** Its a synchronous information exchange mechanism used for the Inter Process/thread communication. Major difference from shared memory is that through shared memory, lots of data can be shared whereas only limited amount of info/data is passed through message passing. Message passing is relatively fast and free from synchronisation overheads compared to shared memory. Message passing methods includes **Message Queue, Mail box, Signalling**
- **Remote Procedure Call(RPC) and sockets:** Inter Process Communication mechanism used by a process to call a procedure of another process running on the same CPU or on a different CPU which is interconnected in a network. Used for distributed applications like client-server applications. With RPC it is possible to communicate over a heterogeneous network(Network where client and server running on different OS). The CPU/process containing the procedure which needs to be invoked remotely is known as **server**. The CPU/process which initiates an RPC request is known as **client**.

Sockets are used for RPC communication. Its a logical end point in a two way communication link between two applications running on a network. A **port numbe**r is associated with a socket so channel can deliver data to the designated application.

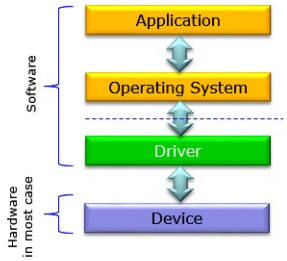
Q) Explain task scheduling in embedded systems. (3 marks)

Task Synchronisation: In a multitasking environment, multiple process run simultaneously and share the system resources. The act of making process aware of the access of shared resources by each process to avoid conflicts is known as 'Task/Process Synchronisation'.
Process/task synchronisation is essential for:

- Avoiding conflicts in resource access(racing, deadlock, starvation etc.)in a multitasking environment.
- Ensure proper sequence of operation across processes.
- Communicating between processes

Q) Explain task communication and task synchronization in embedded system. (7 marks)

Device Drivers: It is a piece of software that acts as a bridge between the operating system and the hardware. Device drivers are responsible for initiating and managing the communication with hardware peripherals. They are responsible for establishing the connectivity, initialising the hardware(setting up various registers of the hardware device) and transferring data. An embedded product may contain different types of hardware components like Wi-Fi modules, I/O devices, storage device etc. The initialisation of these devices and the protocols required for communicating with these devices may be different. All these requirements are implemented in drivers. Hence each hardware requires unique driver component.



Selection criteria for RTOS: The factors include the functional & non-functional requirements for selection an RTOS.

Functional Requirements: These are requirements that specify the desired functionality or operations of a RTOS. It includes:

- **Processor Support**: It is essential to ensure that the processor should support the OS under consideration.
- **Memory Requirement:** Since embedded system is memory constrained, it is essential to have a minimal ROM & RAM requirements for an OS under consideration.
- **Real time capabilities:** It is essential to analyse the real time capabilities of an OS under consideration, as task/process scheduling policies play an important role in real time behaviour.

- **Kernel & Interrupt Latency:** The kernel of the OS may disable interrupts while executing certain services & it may lead to interrupt latency. This latency should be minimal for a high response system.
- **Inter Process Communication & Task Synchronization:** This is a kernel dependent parameter & certain kernel implement policies for avoiding issues in resources sharing.
- **Modularization Support:** It allows the developers to choose essential modules & re-compile OS image for functioning.
- **Support for Networking & Communication**: This include driver support for networking & communication interfaces.
- **Development Language Support:** Certain OS requires run timr libraries for application written in languages like Java, C#, .Net etc.

Non Functional Requirements: They are not related to the software's functional aspect. It concentrates on the expectation and experience of the user.

- **Custom developed or off the shelf:** Custom-developed OS are specifically designed and developed to meet an embedded system's specific needs, while off-the-shelf OS are pre-built OS that can be purchased and used immediately.
- **Cost:** It include developing cost, buying cost, maintenance cost etc.
- **Development and debugging tools availability:** It is a critical factor for selecting an OS, as some OS may be superior in performance, but availability of development & debugging tools are limited.
- **Ease of use:** It specify how easy to use a commercial OS.
- **After sales:** It includes bug or error fixing, patch updation, online support are important factor for selecting an OS.

Q) List the selection criteria for embedded OS. (7 marks)

Micro C/OS-II or µC/OS-II(Micro Controller Operating System Version 2):

- Micro C/OS-II is a simple, easy to use real time kernel written in C language of embedded application development.
- It is a commercial real-time kernel from Micrium Inc.
- It is available for different family of processors/controllers ranging from 8 bit to 64 bit like ARM family of processors, Intel 8085/x86/8051, Freescale 64K series etc.
- It features:
  - **Multitasking:** It is capable of executing multiple application simultaneously without slowing down the system.
  - **Priority based pre-emptive task scheduling:** It chooses the process with the highest priority and runs it until it completes or is stopped by a process with a higher priority.

µC/OS-II Applications: It is used in many embedded systems, including:

- Avionics
- Medical equipment and devices
- Data communications equipment
- Mobile phones, Personal Digital Assistants(PDAs)
- Industrial controls
- Consumer electronics
- Automotive

µC/OS-II services: It includes the following:

1) Task Creation and Management: MicroC/OS-II is a multi-tasking(manage up to 64 tasks) operating system. Each task is an infinite loop and can be in any one of the following 5 states:

- **Dormant:** State where task is created but no resources are allocates.
- **Ready:** State where task is launched into the memory & awaiting the processor time for execution.
- **Running:** State at which the process execution happens in CPU.
- **Waiting/Pending:** State where a running process is temporarily suspended from execution and does not have immediate access to resources.
- **Interrupted:** State at which an interrupt is occurred & CPU execute ISR.

2) Kernel Functions and Initialization: The kernel needs to be initialized & started before executing a task. MicroC/OS-II kernel initialization & OS kernel start is a written as a part of the start up code which is executed before the execution of user tasks.

3) Task Scheduling: MicroC/OS-II support priority based scheduling. Each task has a unique priority ranging from 0 to 63, with 0 being highest priority. Task rescheduling occur whenever a higher priority task becomes 'Ready' to run or when a task enters 'Wait' state or an 'Interrupt' occurs.

4) Inter - task communication: MicroC/OS-II kernel supports inter - task communication for data sharing & interfacing through 'Mailbox' for handling single message & 'Message Queue' for handling multiple messages.

5) Mutual Exclusion and Task synchronization: In multi-tasking environment multiple tasks are executed simultaneously & shared the system resources & data. The access to shared resources should be made mutually exclusive to prevent data corruption. Synchronisation techniques are used to synchronise the their execution.

6) Timing & Reference: The 'Clock Tick' act as the time source for providing timing referencer for time delays & timeouts. It generates periodic interrupts.

7) Memory Management: MicroC/OS-II uses runtime memory allocation & deallocation on a needed basis. The memory is divided into multiple sectors(partitions). Each sector is further divided into blocks of fixed memory size. Each memory sector associates a 'Memory Control Block' with it.

8) Interrupt handling: Under MicroC/OS-II each interrupt is assigned with an 'Interrupt Request Number(IRQ)'. The action required for an interrupt is implemented as a function called 'Interrupt Service Routine(ISR)'. The IRQ links to the corresponding ISR.

Q) Outline the Micro C/OS-II with its services. (7 marks)
Q) Describe the features of Micro C/OS-II.(7 marks)

List popular Real Time Operating Systems(RTOS):

1. **FreeRTOS** (Amazon)
2. **Zephyr** (Linux Foundation)

3. **MQX** (Philips NXP / Freescale)
4. **Keil RTX** (ARM)
5. **µC/OS** (Micrium)
6. **LynxOS** (Lynx Software Technologies)
7. **Integrity** (Green Hills Software)
8. **embOS** (SEGGER)
9. **ThreadX** (Microsoft Express Logic)
10. **Neutrino** (BlackBerry)

Q) List any six popular RTOS. (3 marks)