UNIT-IV

REAL-TIME OPERATING SYSTEMS BASED EMBEDDED SYSTEM DESIGN

veera

CONTENTS

4.1. Operating systems basics

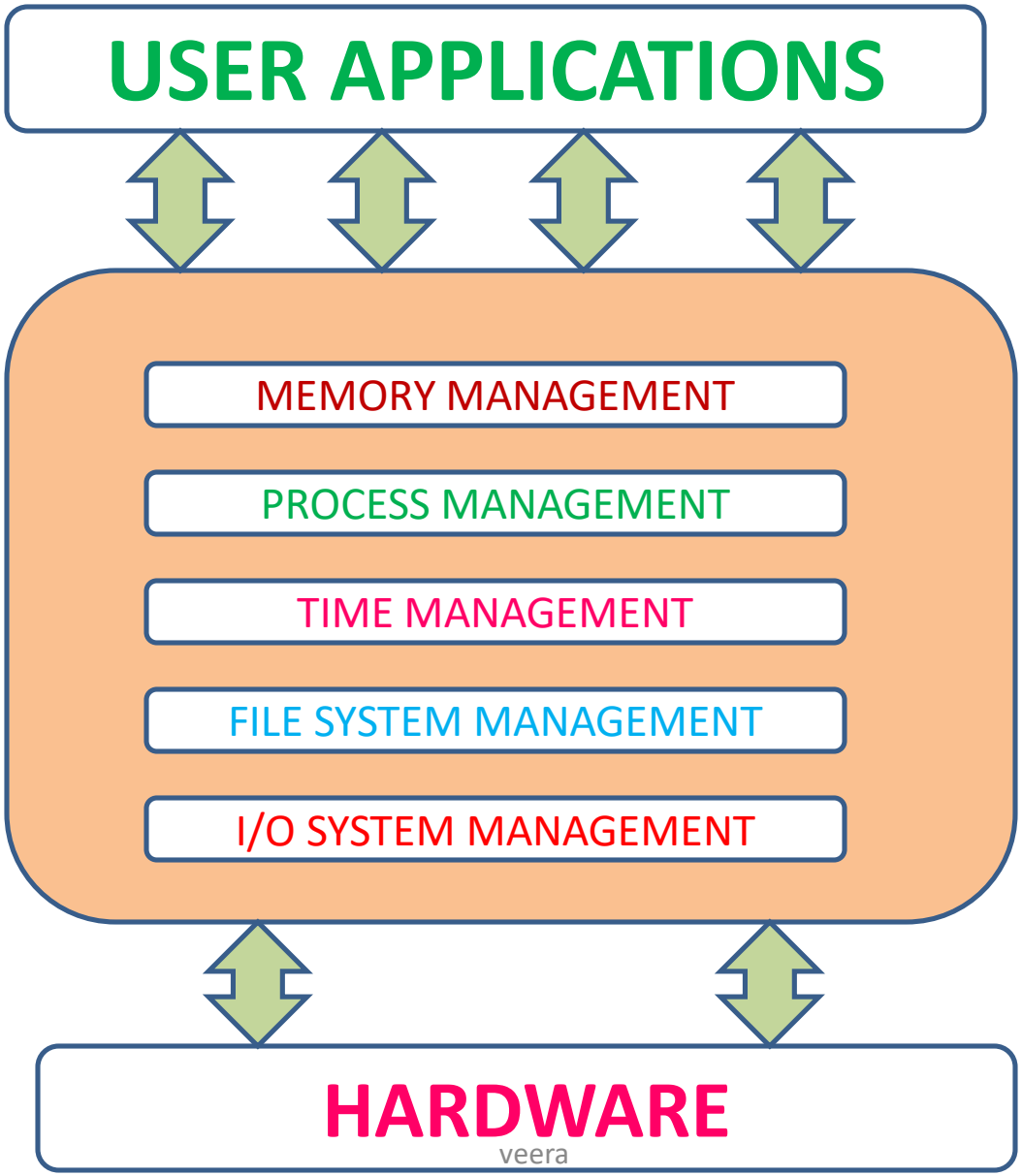
- ✓ The operating system acts as a bridge between the user applications/tasks and underlying the system resources through a set of system functionalities and services.
- ✓ The OS manages the system resources and makes them available to the user applications/tasks on a need basis.
- ✓ A normal computing system is a collection of different I/O subsystems, working and storage memory.

The primary functions of an operating system is

a) Make the system convenient to use

b) Organise and manage the system resources efficiently and correctly.

OS components or Architecture



KERNEL

- Kernel is the core of the operating system and is responsible for managing the system resources and the communication among the hardware and other system services
- Kernel acts as a abstraction layer between system resources and user applications
- Kernel contains a set of system libraries and services.

For a GPOS, the kernel contain different services for handling the following

- Process management
- Memory (primary and secondary)management
- File system management
- I/O system(Device) management
- Protection systems
- Interrupt handler

Process management(PM):

- ✓ It deals with managing the process/tasks
- ✓ PM includes setting up the memory space for the process, loading the process's code into the memory, allocating system resources, scheduling and the managing the execution of the process, setting up and managing the process control block(PCB), IPC and synchronisation, Process termination/ detection etc.

Primary memory management:

- The term primary refers to the volatile memory (RAM) where processes are loaded and variables and shared data associated with each process are stored.
- The memory management unit(MMU) of the kernel is responsible for

a) Keeping track of which part of the memory area is currently used by which process

b) Allocating and de-allocating memory space on a need base(Dynamic memory Allocation)

File system management:

- ✓ file is a collection of related information
- ✓ A file could be a program, text files, image files, word documents,audio/video fiels etc.
- ✓ The file operation is a useful service provided by the OS

The 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 space available.
- Providing a flexible naming convention for the files

I/O system management:

- Kernel is responsible for routing the I/O request coming from different user applications to the appropriate I/O devices of the system.
- In a well-structured OS, the direct accessing of I/O devices are not allowed and the access to them are provided through a set of Application Programming Interfaces(API) exposed by the kernel

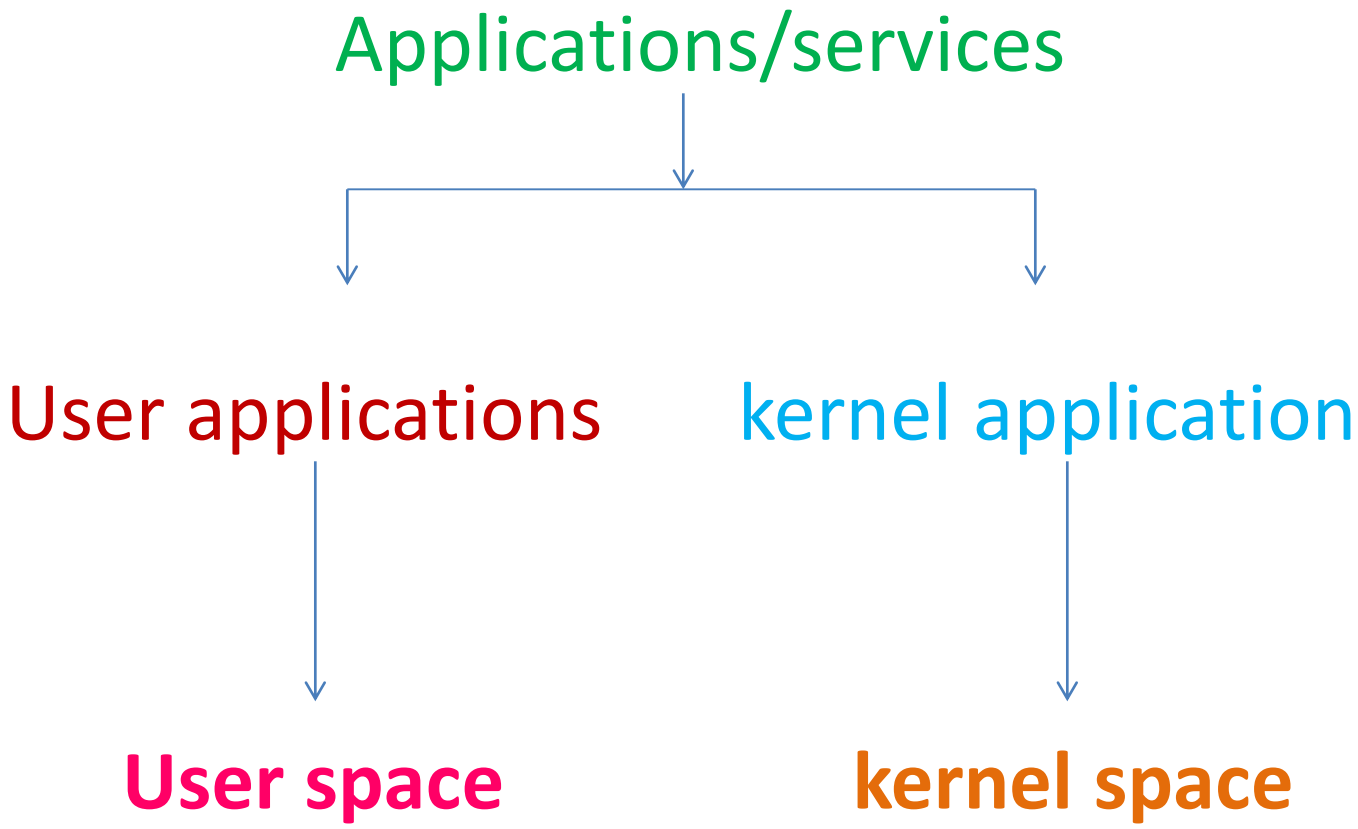
- ✓ The kernel maintains a list of all the I/O devices of the system.
- ✓ The service ‘device manager’ of the kernel is responsible for handling all I/O device related operations
- ✓ The kernel talks to the I/O device through a set of low-level system calls, which are implemented in a service, called device drivers

- **Secondary storage management:**
- It deals with managing the secondary storage memory devices, if any, connected to the system.
- Secondary memory is used as backup medium for programs and data since main memory is volatile.
- In most of the systems, the secondary storage is kept in disks(hard disk).

The secondary storage management service of kernel deals with

- ✓ Disk storage allocation
- ✓ Disk scheduling
- ✓ Free disk space management

Kernel space and user space



Kernel types

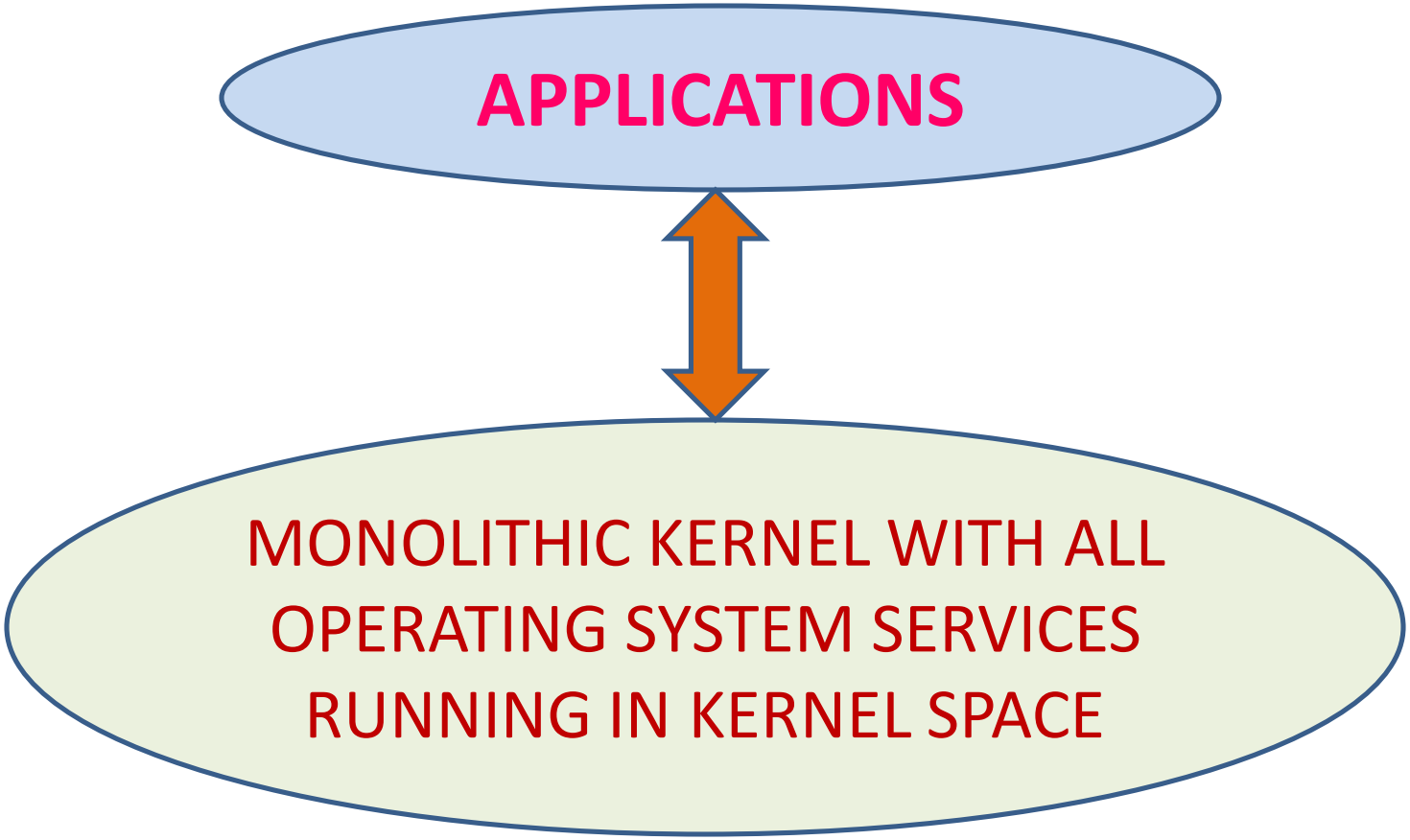
Based on kernel design, kernels can be classified into

A) monolithic kernel

B) microkernel

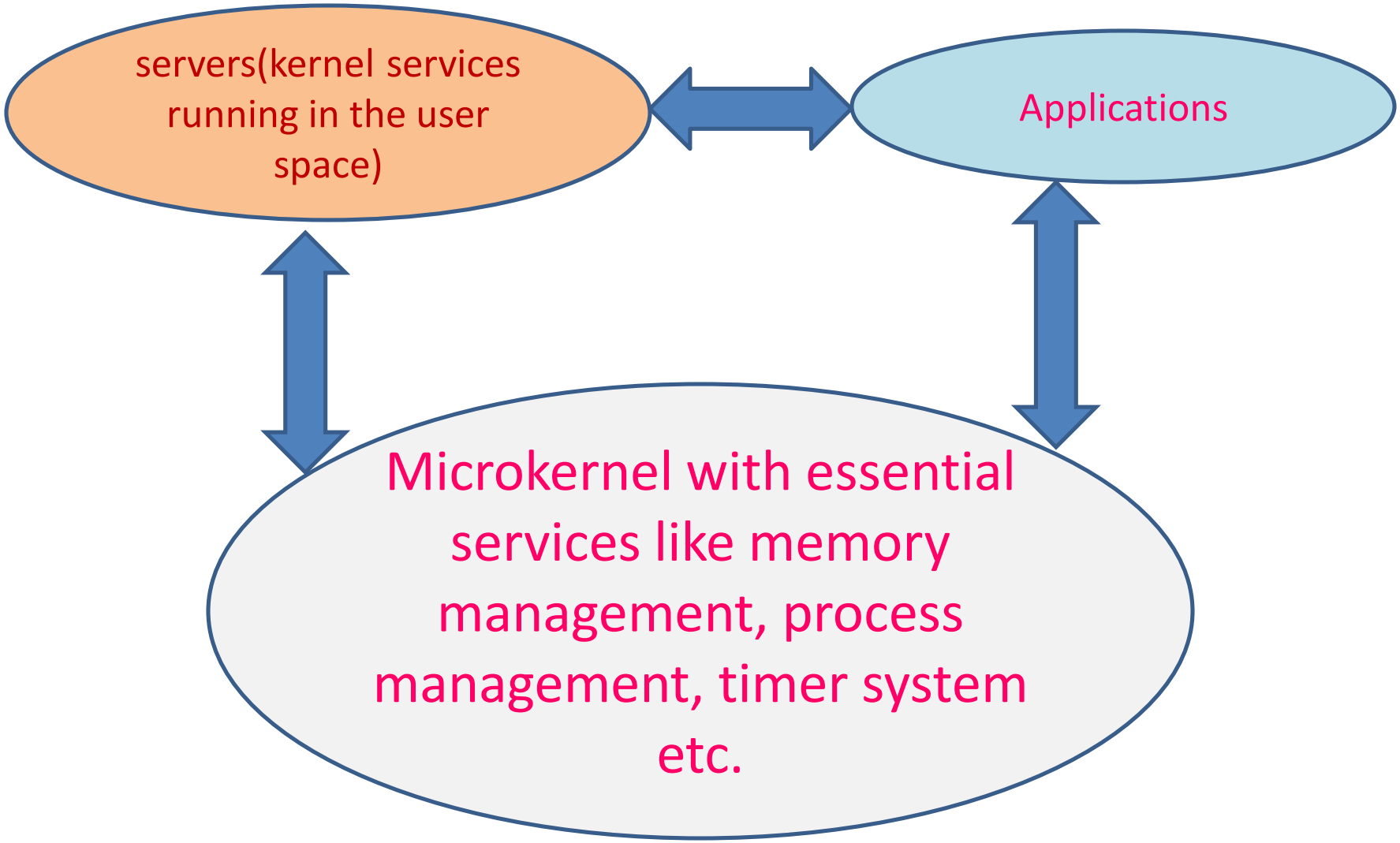
Monolithic Kernel

- ✓ In Monolithic kernel architecture, all kernel services run in the kernel space.
- ✓ Here all kernel modules run within the same memory space under a single kernel thread
- ✓ the major drawback of monolithic kernel is that any error or failure in any one of the kernel modules leads to the crashing of the entire kernel application.
- ✓ Ex: LINUX, SOLARIS,MS-DOS


APPLICATIONS
MONOLITHIC KERNEL WITH ALL OPERATING SYSTEM SERVICES RUNNING IN KERNEL SPACE

MICROKERNEL

- ✓ the microkernel design incorporates only the essential set of operating system services into the kernel.
- ✓ The rest of the operating system services are implemented in programs known as ‘servers’ which runs in user space.
- ✓ Memory management, process management, timer systems and interrupt handlers are the essential services. Which forms the part of microkernel.


servers(kernel services running in the user space)
Applications
Microkernel with essential services like memory management, process management, timer system etc.

4.2. Types operating systems

Operating systems are classified into different types

i) General purpose operating systems

ii) Real-time operating systems

4.2(i) GPOS

- The operating systems, which are deployed in general computing systems, are referred as GPOS
- The kernel of such an OS is more generalised and it contains all kinds of services required for executing generic applications
- GPOS are often quite non-deterministic in behaviour

- Their services can inject random delays into application software and may cause slow responsiveness of an application at unexpected times

- Personal computers/desktop system is a typical example for a systems where GPOS are deployed.

- Windows XP/MS-DOS etc are examples for GPOS.

4.2(ii) RTOS

- Real-time implies deterministic timing behaviour.
- Deterministic timing behaviour in RTOS context means the OS services consumes only known and expected amounts of time regardless the number of services
- The RTOS decides which applications should run in which order and how much time needs to be allocated for each application.

- Predictable performance is the hallmark of well-designed RTOS.

- WindowsCE, QNX, VxWorks, MicroC/OS-II, etc are examples of RTOS.

a) Real-time kernel
b) Hard real-time
c) Soft real-time

a) Real-time kernel

- The kernel of a RTOS is referred as Real-time kernel.
- it is highly specialised and it contains only the minimal set of services required for running the user applications/tasks

- The basic functions of RTK are

i) Task/Process management
ii) Task/Process scheduling
iii) Task/Process synchronisation
iv) Error/Exception handling
v) Memory management
vi) Interrupt handling
vii) Time management

Task/Process management

- Deals with setting up the memory space for the tasks, loading the task's code into the memory space, allocating system resources, setting up a task control block(TCB) for the task and task/process termination/deletion.

- A TCB is used for holding the information corresponding to a task.

- TCB usually contains the following set of information

✓ TASK ID: task identifier number

✓ TASK STATE: the current state of the task

✓ TASK TYPE: task type. Indicates what is the type for this task. the task can be a hard real time or soft real time or background task

✓ TASK PRIORITY: task priority(EX: Task priority=1 for task with priority=1)

✓ TASK CONTEXT POINTER: context pointer. Pointer of context saving.

- TASK MEMORY POINTER: pointers to the code memory, data memory and stack memory for the task.

- TASK SYSTEM RESOURCE POINTER: pointers to systems resources(semaphores,mutex etc) used by task.

- TASK POINTERS: pointers to other TCBs

- Task management service utilises the TCB of a task in the following way

- Creates a TCB for a task on creating task
- Delete/Remove the TCB of a task when the task is terminated or deleted.
- Reads the TCB to get the state of task
- Update the TCB with updated parameters on need basis
- Modify the TCB to change the priority of the task dynamically

Task/Process Scheduling

- Deals with sharing the CPU among various tasks/processes
- A kernel application called ‘Scheduler’ handles the task scheduling
- Scheduler is nothing but an algorithm implementation, which performs the efficient and optimal scheduling of tasks to provide deterministic behaviour.

Task/Process synchronisation

- Deals with synchronising the concurrent access of resources, which is shared across multiple tasks and the communication between various tasks

Error/Exception handling

- Deals with registering and handling the errors occurred/exceptions raised during execution of tasks

- Insufficient memory, timeouts, deadlocks, deadline missing, bus error/ divide by zero, unknown instruction execution, etc. are the examples of errors/exceptions

Memory management

- Compared to GPOS, the memory management function of a n RTOS kernel is slightly different.
- RTOS makes use of “BLOCK” based memory allocation technique, instead of the usual dynamic memory allocation techniques used by the GPOS.
- RTOS kernel uses blocks of fixed size of dynamic memory and the block is allocated for a task on a need basis.

- Errors/Exceptions can happen at the kernel level services or at a task level.

- Deadlock is an example for kernel level exception, whereas timeout is an example for a task level exception

- The OD kernel gives the information about the error in the form of a system call(API)

  GetLastError() API provided by windowsCE RTOS is an example for a such system call.

Interrupt handling

- Deals with the handling of various types of interrupts
- Interrupts provide time behaviour to systems
- Interrupts informs the processor that an external device or an associated tasks requires immediate attention of the CPU
- Interrupts can be either Synchronous or Asynchronous

- Interrupts which occurs in sync with currently executing task is known as synchronous interrupts

  EX: software interrupts,divide by zero, memory segmentation error, etc. are the examples of synchronous interrupts
- Asynchronus interrupts are interrupts, which occurs at any point of execution of any task, and are not in sync with the currently executing task

  EX: the interrupts generated by external devices connected to the processor/controller, timer overflow interrupts, serial data reception/transmission interrupts

Time management

- Accurate time management is essential for providing precise time reference for all applications
- The time reference to kernel is provided by a high-resolution Real-Time Clock(RTC) hardware chip(hardware timer).
- The hardware timer is programmed to interrupt the processor/controller at a fixed rate
- The timer interrupt is referred as ’timer tick’

4.3. TASK, PROCESS, THREADS

TASKS

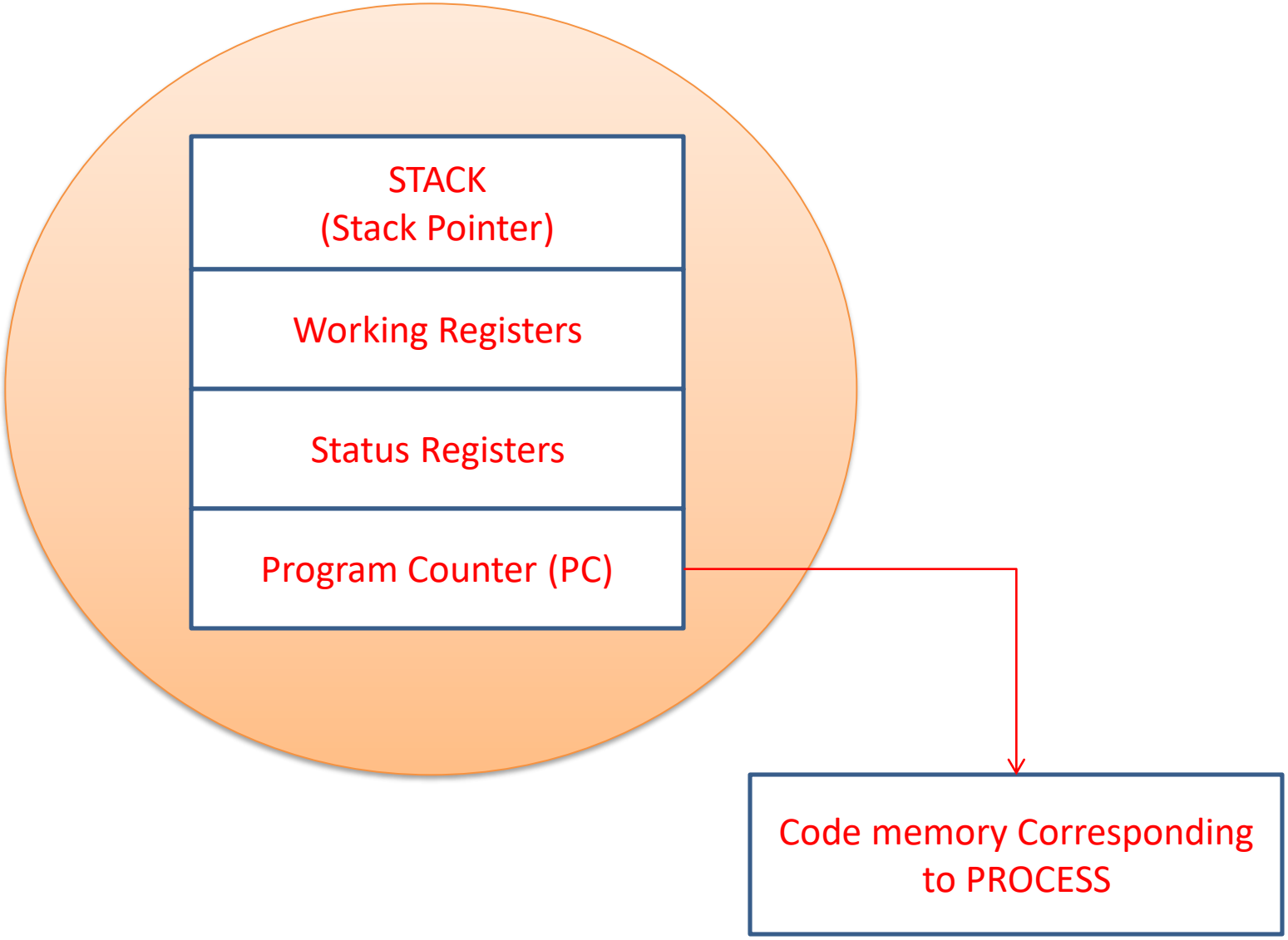
- The term 'task' refers to something to be done
- In the operating systems, a task is defined as the program in execution and the related information maintained by the OS for the program.
- Task is also known as 'job' in OS context.
- A program or part of it in execution is also called a 'process'
- The terms 'task', 'job' and 'process' refer to the same entity in the OS

PROCESS

- ✓ A ‘process’ is a program, or part of it, in execution.
- ✓ Process is also known as instance of a program in execution
- ✓ multiple instances of 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.
- ✓ A process is sequential in execution.

structure of process

- The concept of process leads to concurrent execution of tasks and thereby efficient utilisation of the CPU and other system resources.
- Concurrent execution is achieved through the sharing of CPU among the processes
- A process mimics a processor in properties and holds a set of registers, process status, a program counter(PC) , a stack for holding the local variables associated with the task and the code corresponding to the process.


STACK
(Stack Pointer)
Working Registers
Status Registers
Program Counter (PC)
Code memory Corresponding to PROCESS

- The process has three memory regions. Namely, stack memory, data memory and code memory

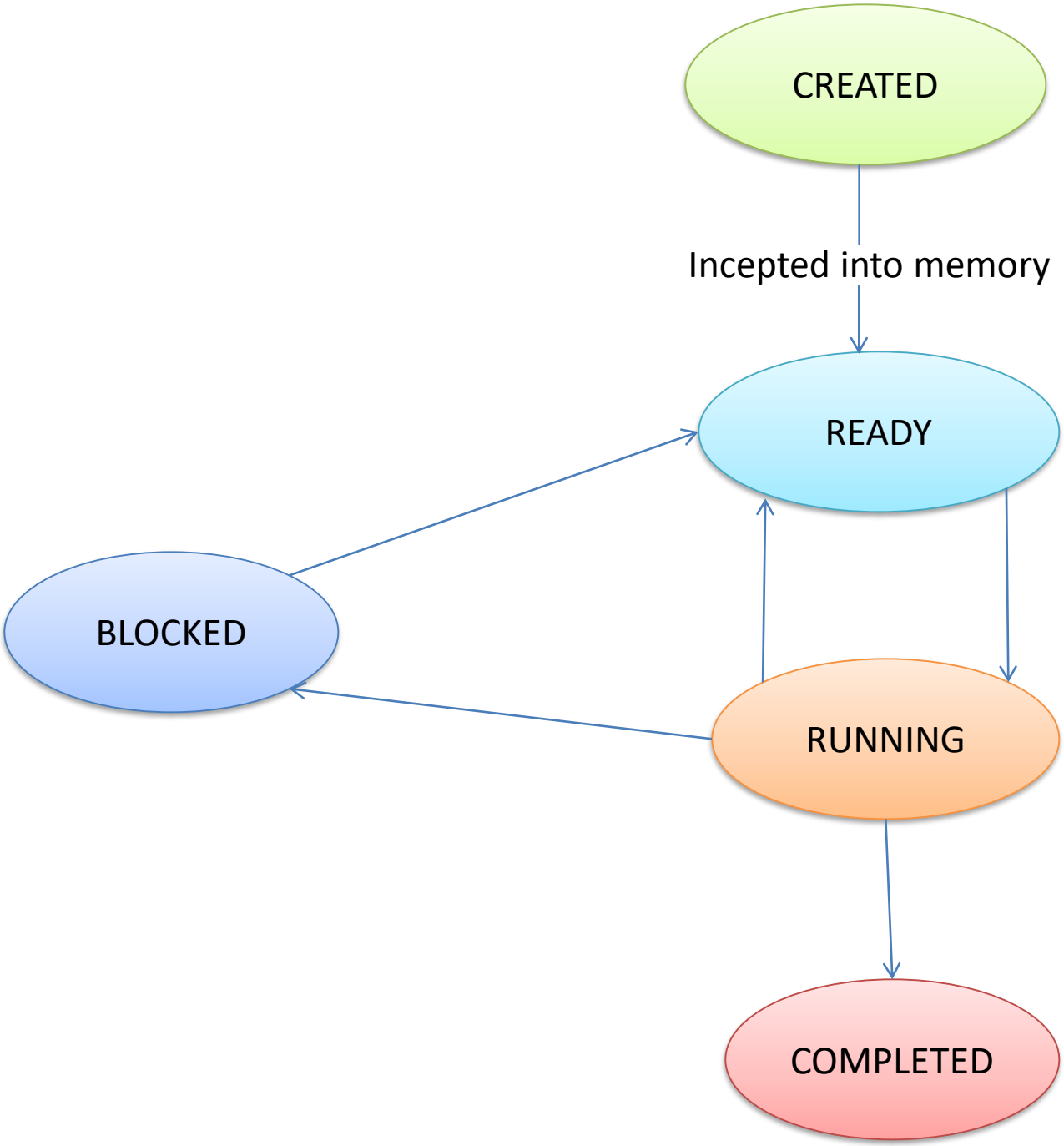
- The stack memory holds all temporary data such as variables local to the process.

- Data memory holds all global data for the process

- The code memory contains the program code corresponding to the process.

Process states and state transition

- 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
Incepted into memory
READY
BLOCKED
RUNNING
COMPLETED

Process management

- Process management deals with the creation of a process, setting up the memory space for the process, loading the process's code into the memory space, allocating system resources, setting up a Process control block for process and process termination.

THREADS

- A thread is a primitive that can execute code.
- A thread is a single sequential flow of control within a process
- A thread is also known as light weight process.
- A process can have many threads of execution.