Embedded System Architecture - CSEN 701

Module 6: *Multi-tasking and Real-Time Systems*

Lecture 13: *Introduction to Multitasking*

Dr. Eng. Catherine M. Elias

catherine.elias@guc.edu.eg

Lecturer, Computer Science and Engineering,
Faculty of Media Engineering and Technology, German University in Cairo

Outline

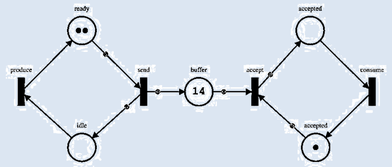
- Concurrency
- Dining Philosopher's Problem
- Operating System
  - Definition
  - Responsibilities
  - Process Managment
  - Resource Management
  - Real-time Operating System
  - Multitasking and Scheduling

The big Picture

The Real-time Embedded System

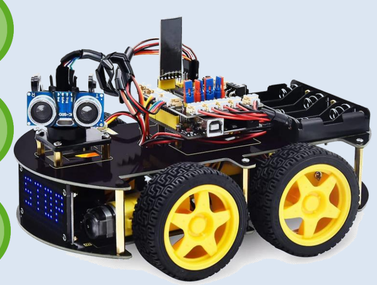
ES Modeling & Design (*2 Modules*)

- System Modeling ✓
- Design Considerations
- Power management and optimization
- Reliability and fault tolerance



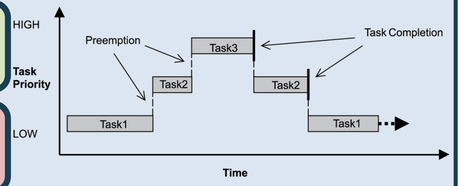
ES Hardware Components (*4 Modules*)

- Microcontroller Fundamentals ✓
- Embedded programming languages ✓
- Embedded Hardware ✓
- Communication and Networking ✓



ES Software Components (*1 Module*)

- Real-Time Systems — Let's go
- Multi-tasking
- Scheduling
- Resource Management



Embedded System Tools & Software Development (*2 Modules*)

- Debugging techniques
- Interrupts and exception handling — Let's go
- Memory management

Parallelism vs Concurrency

- Concurrency is central to embedded systems.
- So far, we only employed the concept in the **modelling of discrete-event systems** using **State charts**.

- A computer program is said to be **concurrent** if different parts of the program **conceptually execute simultaneously** on the **single** hardware.
- A program is said to be **parallel** if different parts of the program **physically execute simultaneously** on **distinct** hardware
  - Multicore processor
  - Multiple servers at the same time
  - Distinctive processors within a SoC (e.g. GPU)

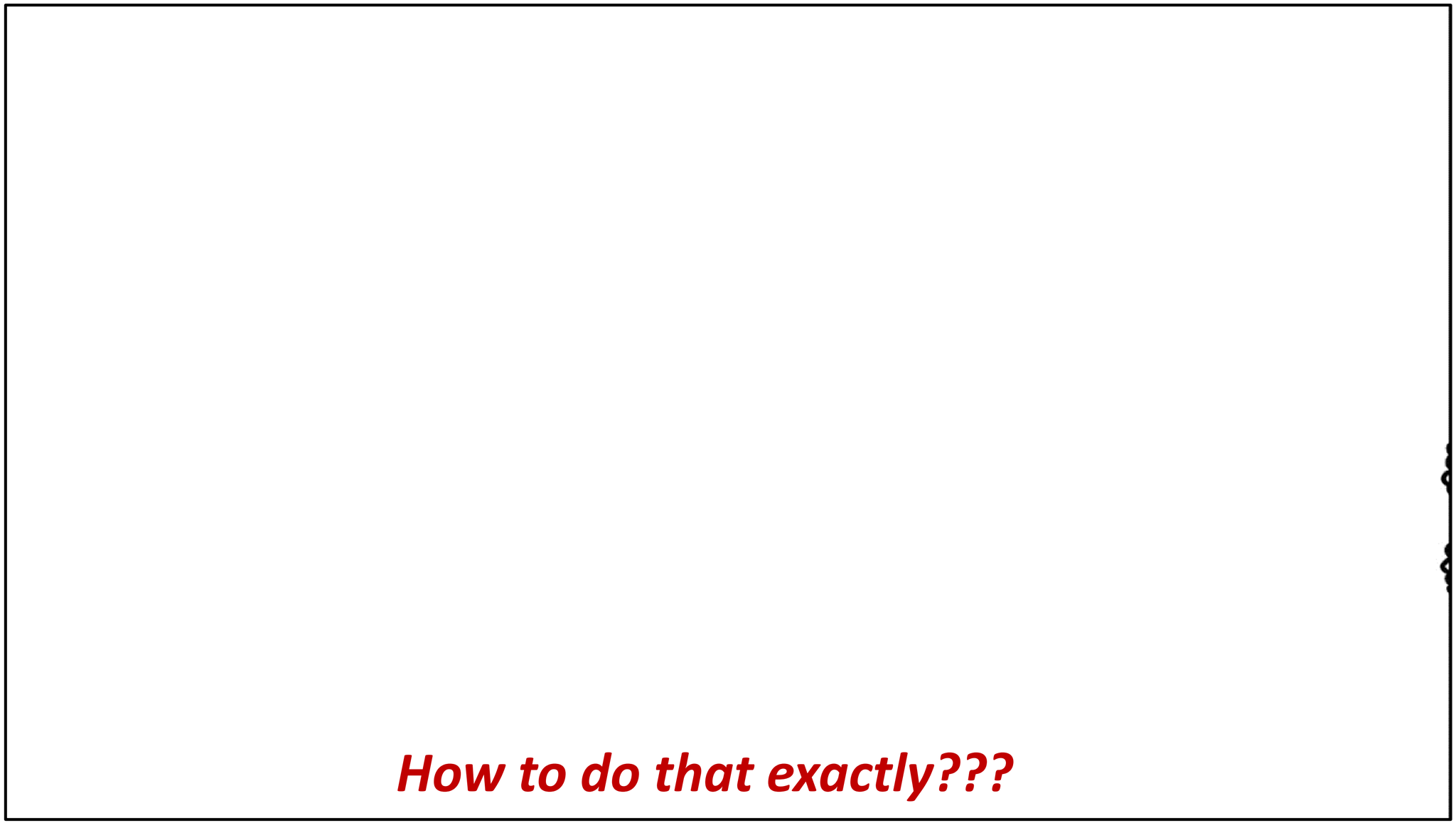
Imperative Programming language

- Non-concurrent programs specify a sequence of instructions to execute.
- ***Imperative Language***
  - ➢It is the class of programming languages that express the computation as a sequence of instructions
    - C
    - Java

- How to write concurrent programs in C?
  - ➢Using thread libraries provided by the OS

What does this mean??

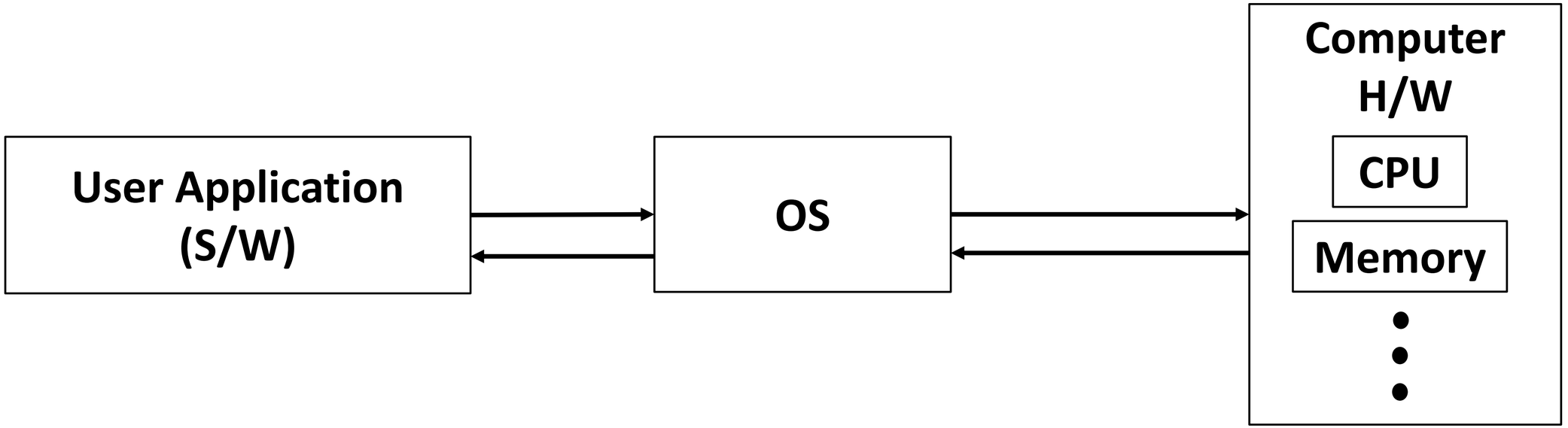
Dining Philosopher's Problem



How to do that exactly???

Definition

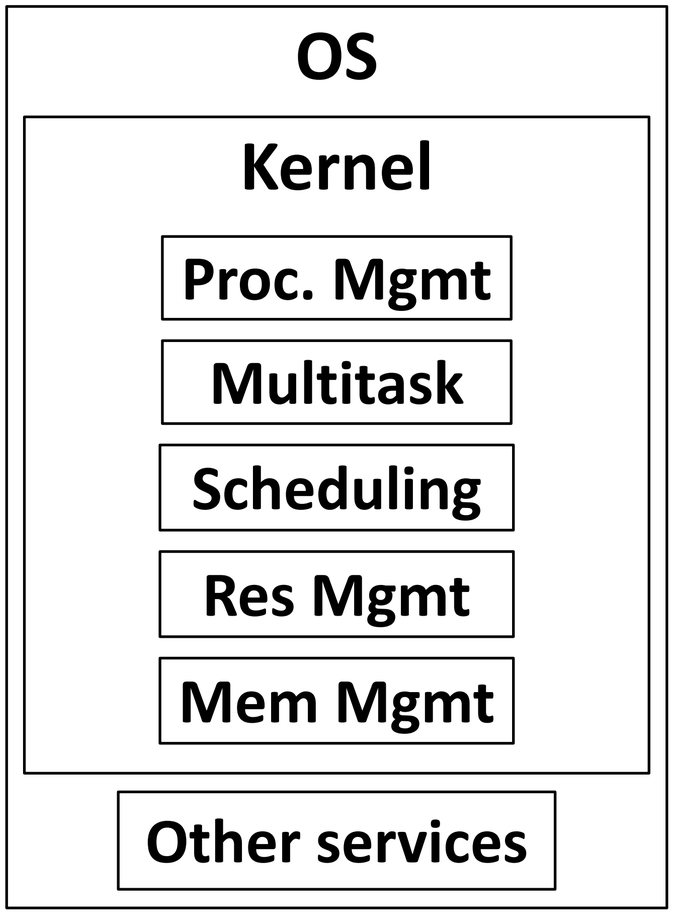
- An operating system is a **software** responsible for **connecting** a computer's **hardware** (H/W) resources (CPU, Memory, IO periferal ,…) with the **software** (S/W) **applications** executed by the user (C program, C++ program, …)



Responsibilities

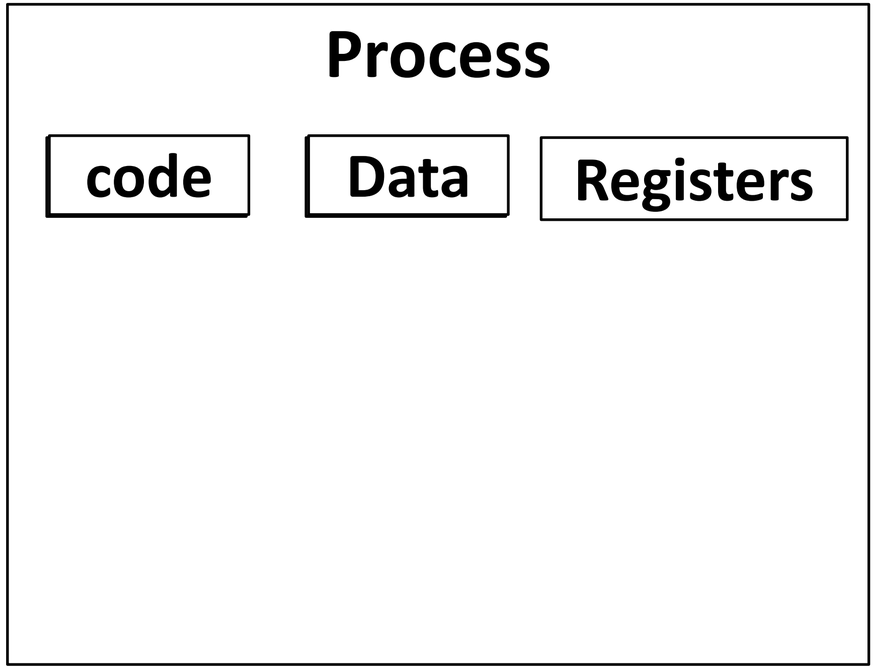
- The OS is composed of different software components, the core component is known as the ***kernel*** which is responsible for:
  - Process/task Management
  - Multitasking and Scheduling
  - Resource Management
  - Memory Management

- Moreover, the OS also handles:
  - I/O interfaces → ***Interrupt Handling***
  - Files Management
  - Other services



Process Managment: *Processes and Threads*

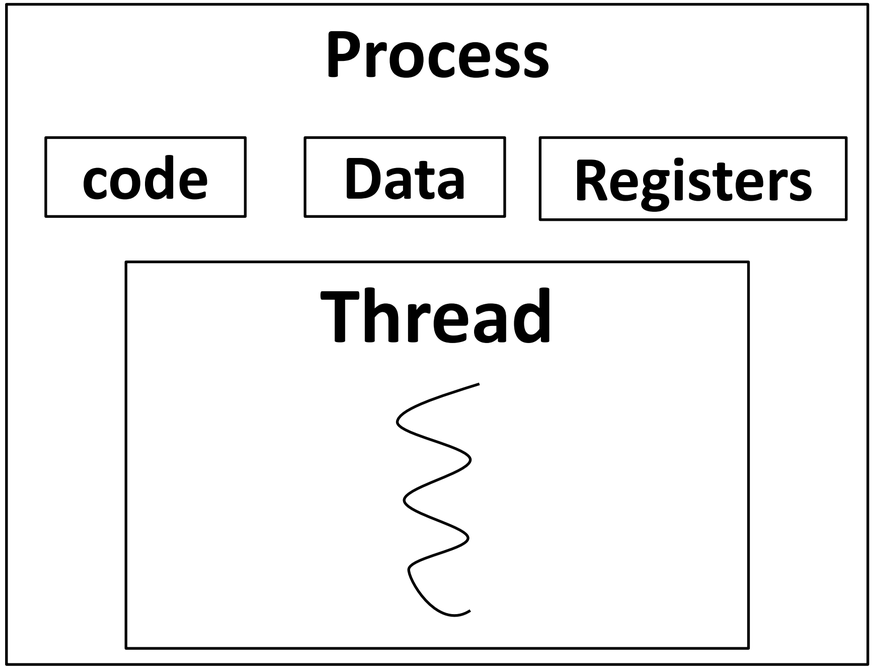
- When the user executes an designed application (C program). The kernel defines a new **process**.
  Note: A Process is also called a task (older term).
- A **process** is defined by its **code**, **registers values**, **Data variables**, .. Etc.
- These values are saved in a **memory space** assigned by the **kernel**.



Process Managment: *Processes and Threads*

- Once a process is defined, the user code is executed by creating a **thread**.
- A ***thread*** is the **basic unit** of a process defining an **execution path of a task** to be passed to the CPU to execute its instruction. **(When a C-function is called, a thread is created to execute its code.)**

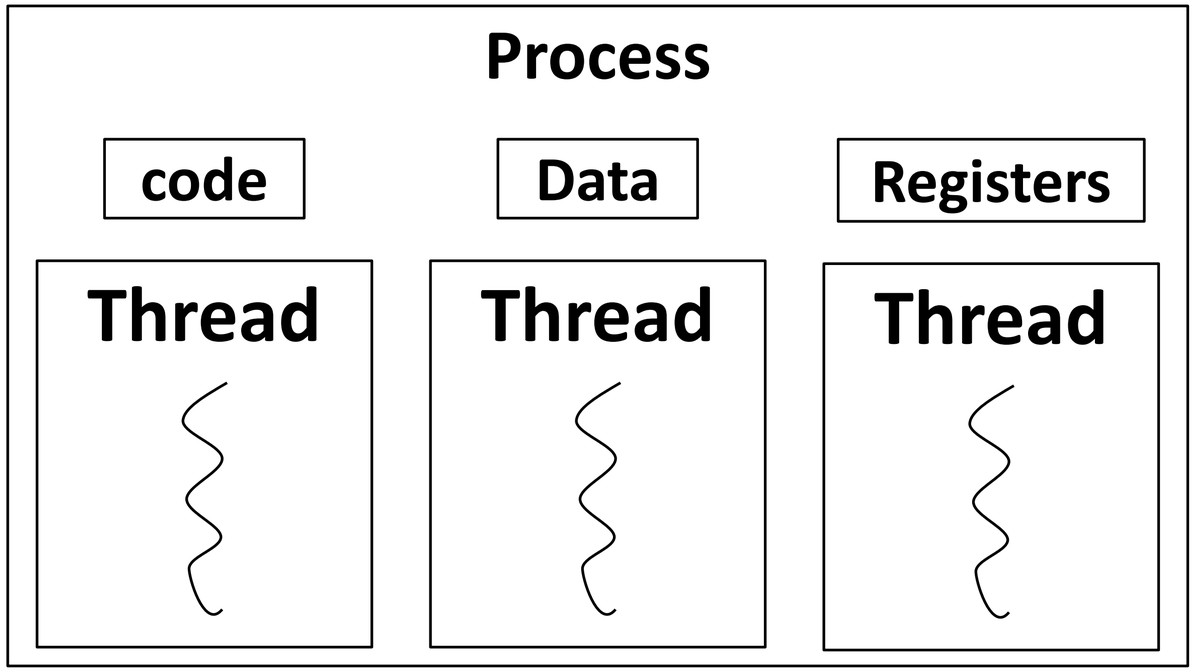
- Once a thread is created and executed, the processor starts executing its instructions using the process resources saved in the memory.



Process Managment: *Processes and Threads*

- The Operating system can create (Multi threading) concurrently, enabling the execution of a number of tasks concurrently **multiple threads** on a single processor. **(Running different instances of C-functions concurrently)**.

- The created threads all share the process resource (variables, register values,..)

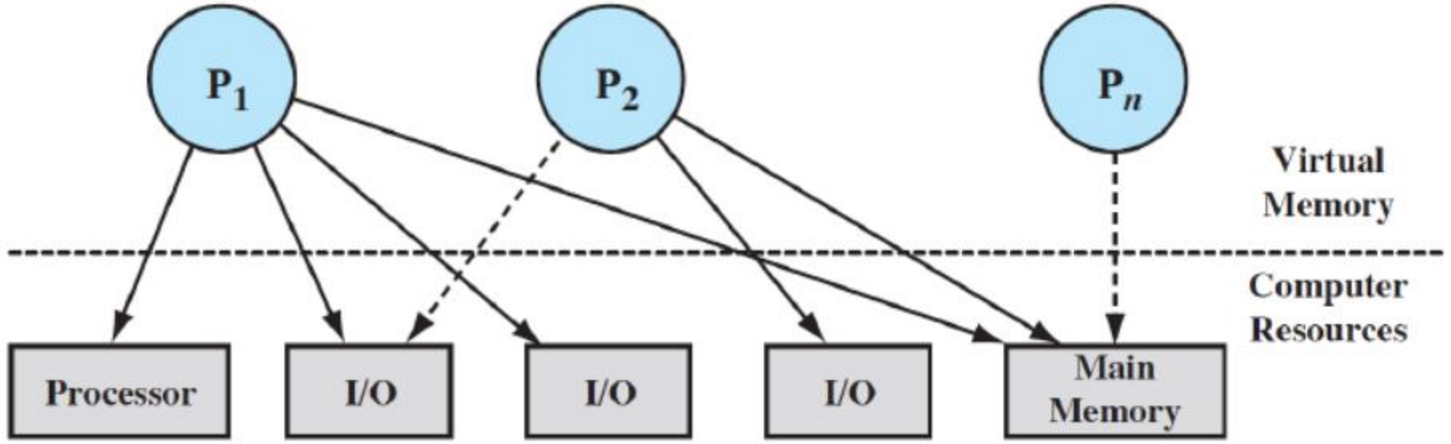


Process Managment: *Processes and Threads*

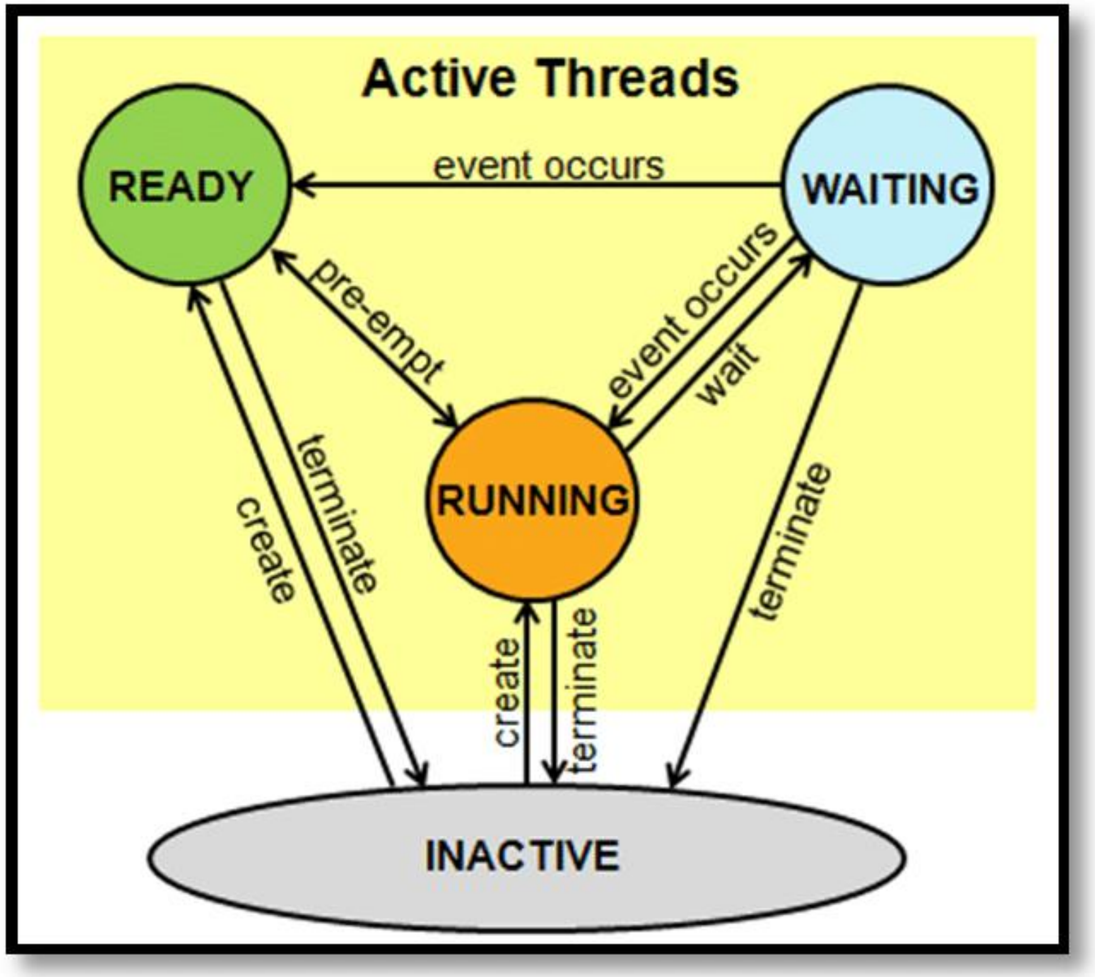
- A processor can **only execute one thread at an instance**.
- Thus, to run concurrent tasks on a processor, the OS schedules which thread should be executed on the processor at each instance → **(Task Scheduling)**
- Moreover, since the threads shares the process resources, the OS controls the accessing of the shared resources between the running tasks. → **(Shared resources Management)**

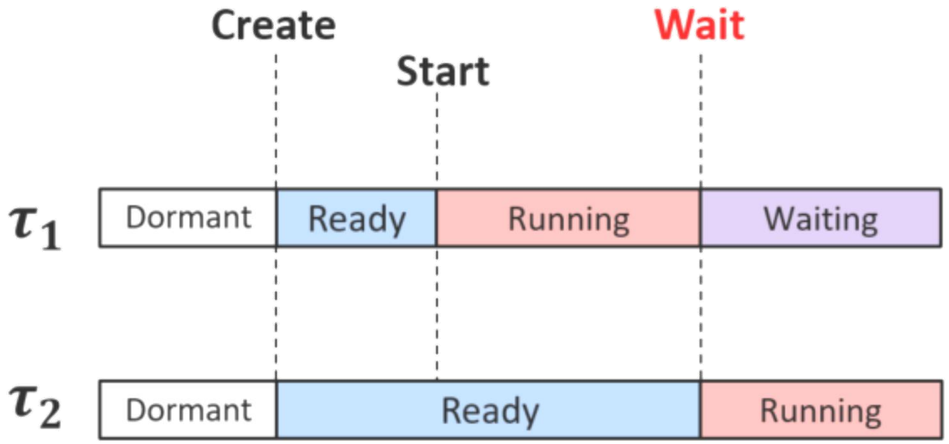
Resource Management: *Allocation and Deadlocks*

- Problems arise in all systems with limited resources and multiple users of resources
  - Computer operation systems
  - Manufacturing processes
  - Traffic processes ...etc.
- **Constraints:** Avoid deadlocks, ensure fairness, and maximize throughput



Scheduling


Active Threads
READY
WAITING
RUNNING
INACTIVE
event occurs
pre-empt
event occurs
wait
create
terminate
create
terminate
terminate

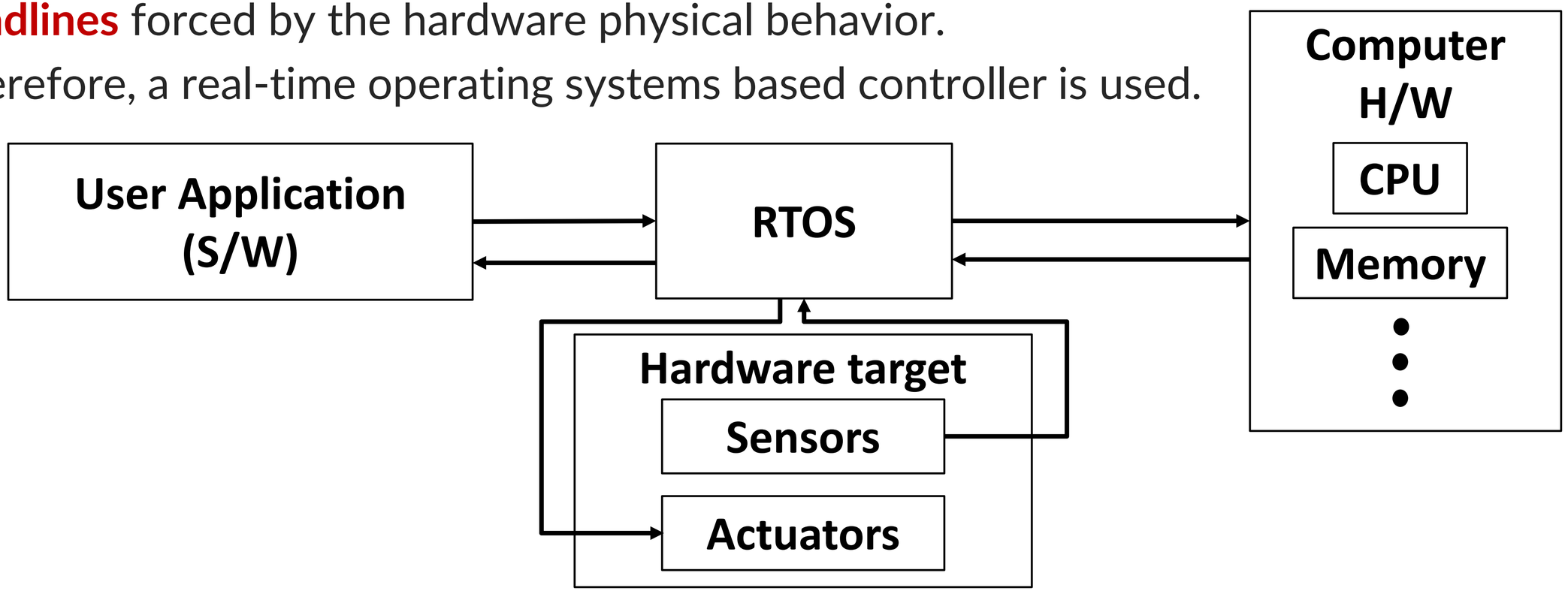

Create
Start
Wait
τ_1
Dormant
Ready
Running
Waiting
τ_2
Dormant
Ready
Running

Scheduling

- Predicting the thread schedule is an iffy (full of uncertainty /doubtful) proposition.
  - **Without an OS**, multithreading is achieved with interrupts. Timing is determined by external events.
  - **Generic OS**, (Linux, Windows, OSX,) provide thread libraries (like"PThreads") and provide no fixed guarantees about when threads will execute.
  - **Real-time OS (RTOSs),** like FreeRTOS, QNX, VxWorks, RTLinux, support a variety of ways of controlling when threads execute (priorities, preemption policies, deadlines, . . .).

Real-time Operating System

- For real-time embedded systems. The **interaction with the environment** introduces **time constraints** to the system.
- Thus the controller must be able to **compute multiple tasks**, while considering **timing deadlines** forced by the hardware physical behavior.
- Therefore, a real-time operating systems based controller is used.

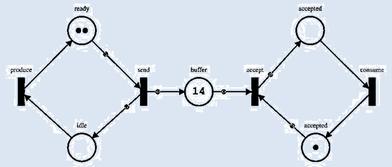


The big Picture

The Real-time Embedded System

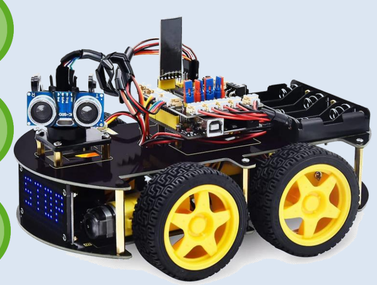
ES Modeling & Design (*2 Modules*)

- System Modeling
- Design Considerations
- Power management and optimization
- Reliability and fault tolerance



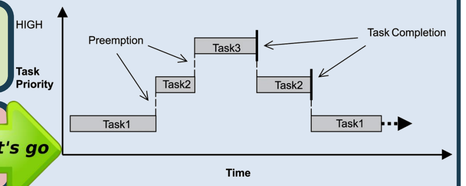
ES Hardware Components (*4 Modules*)

- Microcontroller Fundamentals
- Embedded programming languages
- Embedded Hardware
- Communication and Networking



ES Software Components (*1 Module*)

- Real-Time Systems
- Multi-tasking
- Scheduling
- Resource Management



Let's go

Embedded System Tools & Software Development (*2 Modules*)

- Debugging techniques
- Interrupts and exception handling
- Memory management

Let's go

For Further Inquiries, Please send an email

Catherine.elias@guc.edu.eg,
Catherine.elias@ieee.org

Thank you for your attention!

See you next time ☺