Sharif University of Technology
Department of Computer Science and Engineering

Lec. 1:
Introduction

Real-Time Computing

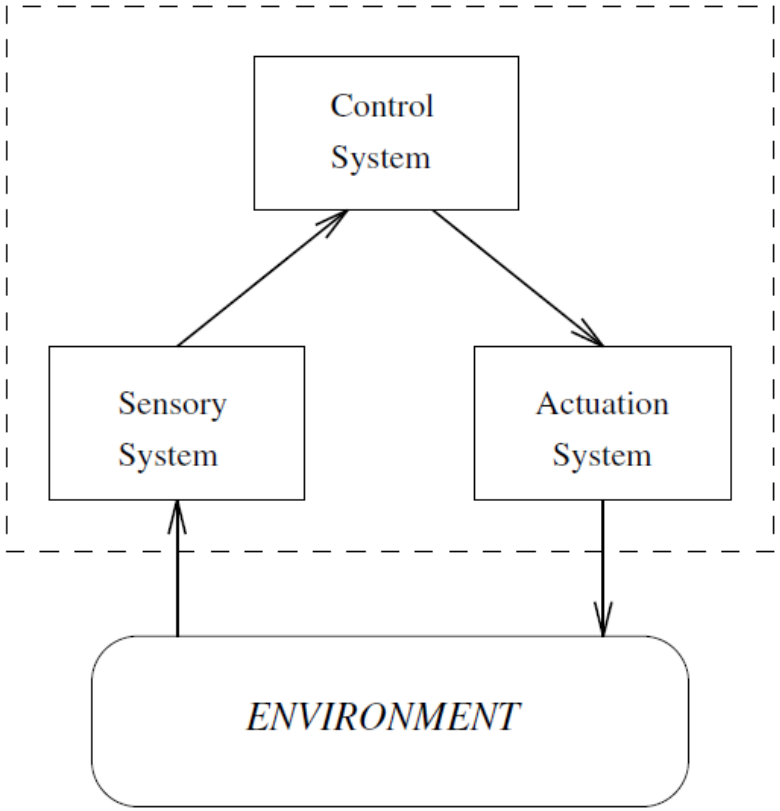
S. Safari
Fall 2023

What Does Real Time Mean?

- The main characteristic that distinguishes real-time computing from other types of computation is **time**.
- The word ***time*** means:
  - The correctness of the system depends not only on the logical result of the computation but also on the time at which the results are produced.
- The word ***real*** indicates:
  - The reaction of the systems to external events must occur during their evolution. As a consequence, the system time (internal time) must be measured using the same time scale used for measuring the time in the controlled environment (external time).

Typical Real-time Architecture

- A block diagram of a typical real-time architecture for controlling a physical system:



A Misconception

- Some people erroneously believe that it is not worth investing in real-time research.
  - Because advances in computer hardware will take care of any real-time requirements.
- Although advances in computer hardware technology will improve system throughput and will increase the computational speed in terms of millions of instructions per second (MIPS), this does not mean that the timing constraints of an application will be met automatically.
- In fact, whereas the objective of fast computing is to minimize the average response time of a given set of tasks, the objective of real-time computing is to meet the individual timing requirement of each task.

Categories of Real-Time Systems

- The main difference between a real-time and a non-real-time task is that a **real-time task** is characterized by a **deadline**, which is the maximum time within which it must complete its execution.
- **Hard**: If producing the results after its deadline may cause catastrophic consequences on the system under control.
- **Firm**: If producing the results after its deadline is useless for the system, but does not cause any damage.
- **Soft**: If producing the results after its deadline has still some utility for the system, although causing a performance degradation.

Hard and Hybrid Real-Time Systems

- A **real-time operating system** that is able to handle hard real-time tasks is called a hard real-time system.
- In general, when an application consists of a **hybrid task set**:
  - All hard tasks should be guaranteed offline
  - Firm tasks should be guaranteed online, aborting or cancelling them if their deadline cannot be met
  - Soft tasks should be handled to minimize their average response time (or maximize Quality of Service (QoS))

Examples of Hard Tasks

- Examples of hard tasks can be found in safety-critical systems.
  - Sensing, actuation, and control activities:
    - Sensory data acquisition
    - Data filtering and prediction
    - Detection of critical conditions
    - Data fusion and image processing
    - Actuator serving
    - Low-level control of critical system components
    - Action planning for systems that tightly interact with the environment

Examples of Firm Tasks

- Examples of firm activities can be found in networked applications and multimedia systems, where skipping a packet or a video frame is less critical than processing it with a long delay.
  - Video playing
  - Audio/video encoding and decoding
  - On-line image processing
  - Sensory data transmission in distributed systems

Examples of Soft Tasks

- Soft tasks are typically related to system-user interactions.
  - The command interpreter of the user interface
  - Handling input data from the keyboard
  - Displaying messages on the screen
  - Representation of system state variables
  - Graphical activities
  - Saving report data

Limits of Current Real-Time Systems

- The main characteristics of kernel-based real-time systems (used to support control applications) include the following:
  - Multitasking
  - Priority-based scheduling
  - Ability to quickly respond to external interrupts
  - Basic mechanisms for process communication and synchronization
  - Small kernel and fast context switch
  - Support of a real-time clock as an internal time reference

Desirable Features of Real-Time Systems

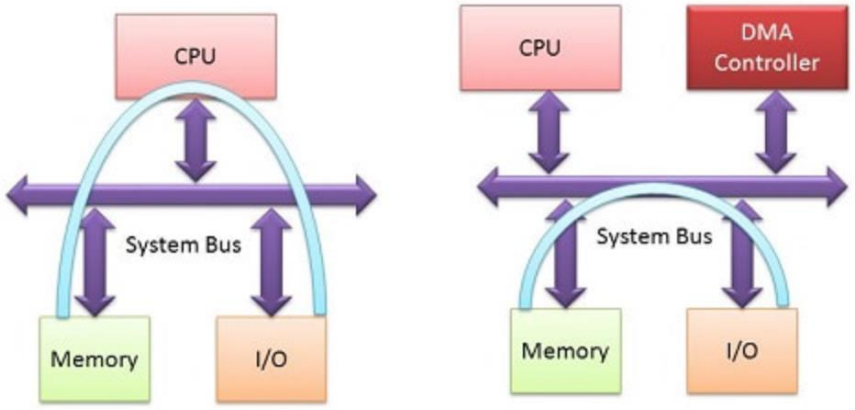
- There are some very important basic properties that real-time systems must have to support critical applications:
  - Timeliness
  - Predictability
  - Efficiency
  - Robustness
  - Fault tolerance
  - Maintainability

The Sources of Nondeterminism of Predictability

- One of the most important properties that a hard real-time system should have is **predictability**.
- The first component that affects the predictability of the scheduling is the **processor itself**.
  - Instruction prefetch, pipelining, **cache** memory, and **direct memory access** (DMA) mechanisms are the cause of nondeterminism
    - Prevent a precise estimation of the worst-case execution times (WCETs)
- Other important components that influence the execution of the task set are **the internal characteristics of the real-time kernel**.
  - The scheduling algorithm, the synchronization mechanism, the types of semaphores, the memory management policy, and the interrupt handling mechanism.

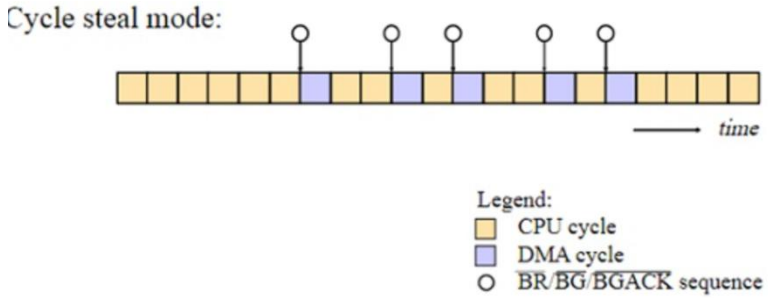
Direct Memory Access (DMA)

- A technique used by many peripheral devices to transfer data between the device and the main memory.
- The purpose of DMA is to relieve the central processing unit (CPU) of the task of controlling the input/output (I/O) transfer.



Direct Memory Access (DMA)

- One of the most common methods is called **cycle stealing**
  - The DMA device steals a CPU memory cycle in order to execute a data transfer.
  - If the CPU and the DMA device require a memory cycle at the same time, the bus is assigned to the DMA device and the CPU waits until the DMA cycle is completed. → there is no way of predicting



- A possible solution: using the memory time-slice method.
  - Each memory cycle is split into two adjacent time slots: one reserved for the CPU and the other for the DMA device.

Cache

- A buffer between the CPU and the random access memory (RAM) to speed up processes' execution.
- It is physically located after the memory management unit (MMU) and is not visible at the software programming level.
- Hit ratio and miss ratio
- The main motivation for having a cache is not only to speed up process execution but also to reduce conflicts with other devices.
- In real-time systems, the cache introduces some degree of nondeterminism.
- Write Issue: any modification made on the cache must be copied to the memory in order to maintain data consistency.
- In preemptive systems, the cache behavior is also affected by the number of preemptions.

Interrupts

- Interrupts generated by I/O peripheral devices represent a big problem for the predictability of a real-time system.
  - Because, if not properly handled, they can introduce **unbounded delays** during process execution.
- The arrival of an interrupt signal causes the execution of a service routine (driver), dedicated to the management of its associated device.

<enable device interrupts>
<wait for interrupt>
<get the result>

- In many operating systems, interrupts are served using a **fixed priority** scheme.
  - Each driver is scheduled based on a static priority, higher than process priorities.

System Calls

- In order to precisely evaluate the worst-case execution time of each task, all kernel calls should be characterized by a bounded execution time.
  - Using the guarantee mechanism
  - Performing the schedulability analysis of the application
    - It is desirable that each kernel primitive be preemptable.
- In fact, any non-preemptable section could possibly delay the activation or the execution of critical activities, causing a timing fault to hard deadlines.

Semaphores

- The typical semaphore mechanism used in traditional operating systems **is not suited** for implementing real-time applications.
  - Because it is subject to the priority inversion phenomenon, which occurs when a high-priority task is blocked by a low-priority task for an unbounded interval of time.
    - Introducing **nondeterministic** delays on the execution of critical tasks.

Memory Management

- Demand paging schemes are not suitable to real-time applications subject to rigid time constraints.
  - Because of the large and unpredictable delays caused by page faults and page replacements.
- **Static allocation** schemes for resources and memory management increase the predictability of the system but reduce its flexibility in dynamic environments.
- The system designer has to make the most suitable choices for balancing predictability against flexibility.

Programming Language

- Current programming languages **are not**:
  - Expressive enough to prescribe certain timing behavior
  - Suited for realizing predictable real-time applications
- Some high-level languages have been proposed to support the development of hard real-time applications.
  - Real-Time Euclid [KS86] is specifically designed to address reliability and guaranteed schedulability issues in real-time systems.
    - Forcing the programmer to specify time bounds and timeout exceptions in all loops, waits, and device accessing statements.
    - It imposes several programming restrictions, such as:
      - » **Absence of dynamic data structures**
      - » **Absence of recursion**
      - » **Time-bounded loops**

Summary

- Typical Real-time Architecture
- Categories of Real-Time Systems
- Limits of Current Real-Time Systems
- Desirable Features of Real-Time Systems
- The Sources of Nondeterminism of Predictability