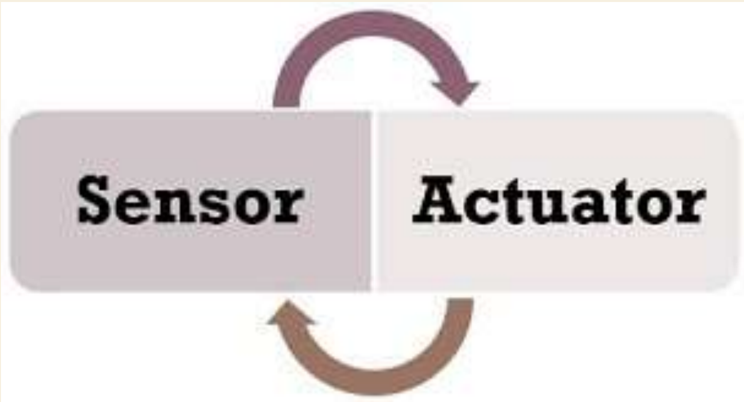
REAL-TIME SOFTWARE DESIGN

REAL-TIME SOFTWARE DESIGN

- Designing embedded software systems whose behaviour is subject to timing constraints
- A system is said to be *real-time* if the total correctness of an operation depends not only upon its logical correctness, but also upon the **time** in which it is performed.

REAL-TIME SYSTEMS

- Systems which monitor and control their environment
- Time is critical. Real-time systems MUST respond within specified times
- Inevitably associated with hardware devices
  - –Sensors: Collect data from the system environment
  - –Actuators: Change (in some way) the system's environment

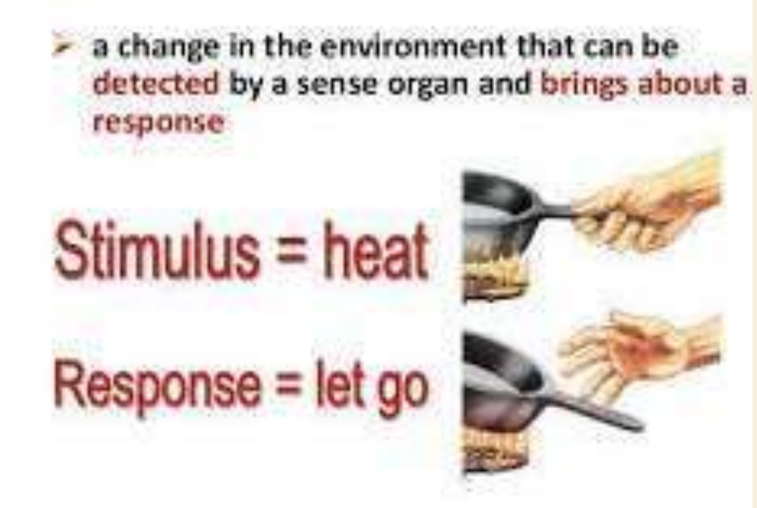


DEFINITION

- A real-time system is a software system where the correct functioning of the system depends on the results produced by the system and the time at which these results are produced
- 'soft' real-time system - system whose operation is degraded if results are not produced according to the specified timing requirements
- 'hard' real-time system - system whose operation is incorrect if results are not produced according to the timing specification

STIMULUS/RESPONSE SYSTEMS

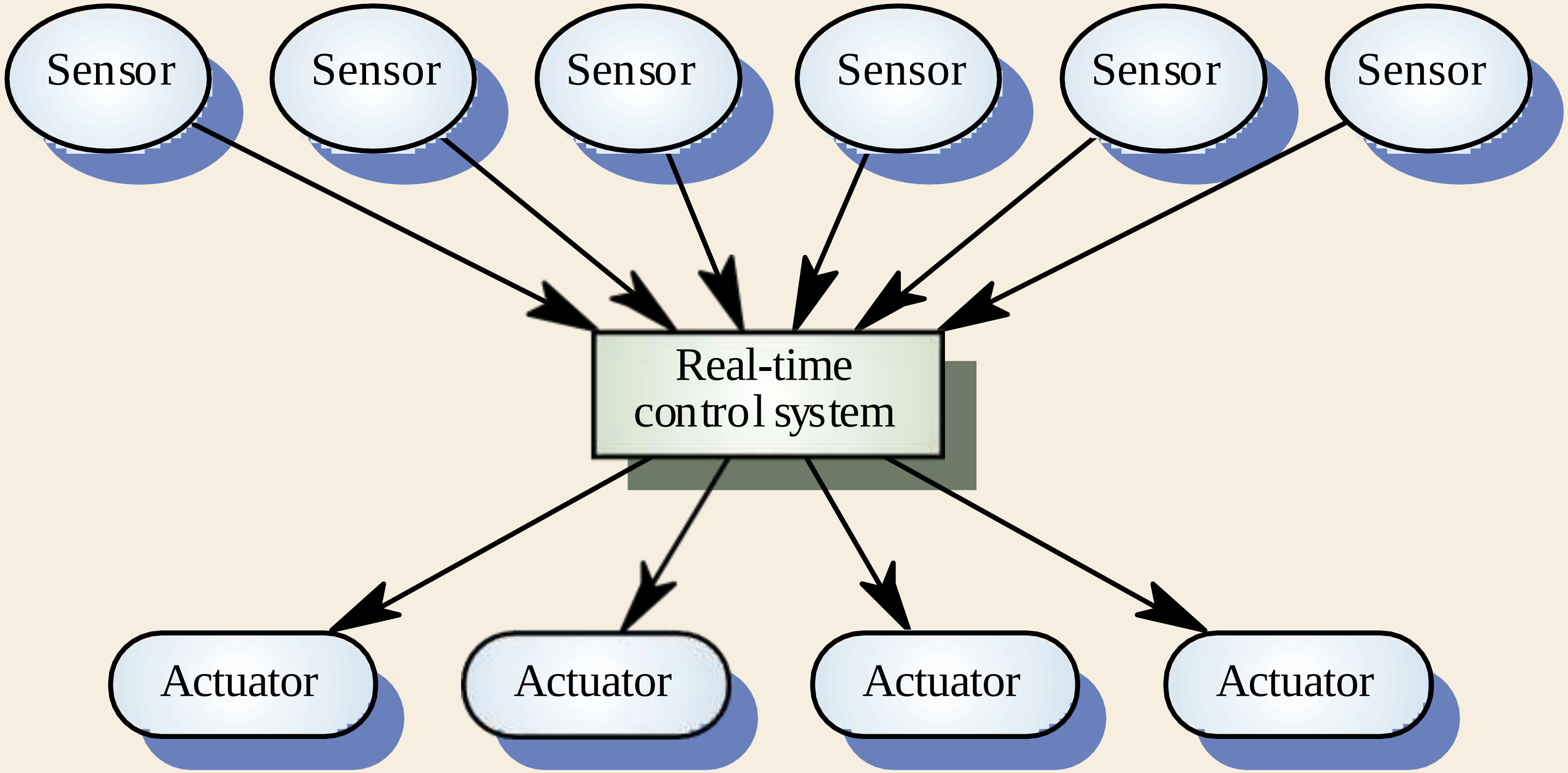
- Given a stimulus, the system must produce a response within a specified time
- Periodic stimuli : Stimuli which occur at predictable time intervals
  - For example, a temperature sensor may be polled 10 times per second
- Aperiodic stimuli : Stimuli which occur at unpredictable times
  - For example, a system power failure may trigger an interrupt which must be processed by the system



ARCHITECTURAL CONSIDERATIONS

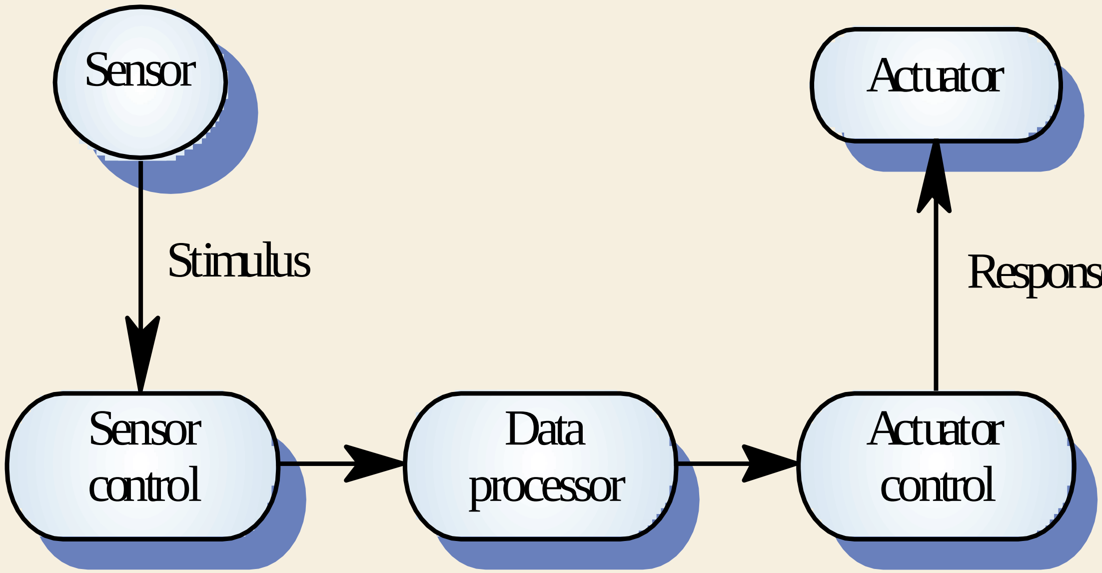
- Because of the need to respond to timing demands made by different stimuli/responses, the system architecture must allow for fast switching between stimulus handlers
- Timing demands of different stimuli are different so a simple sequential loop is not usually adequate
- Real-time systems are usually designed as cooperating processes with a real-time executive controlling these processes

A REAL-TIME SYSTEM MODEL



Source: *Software Engineering, Ian Sommerville*

SENSOR/ACTUATOR PROCESSES



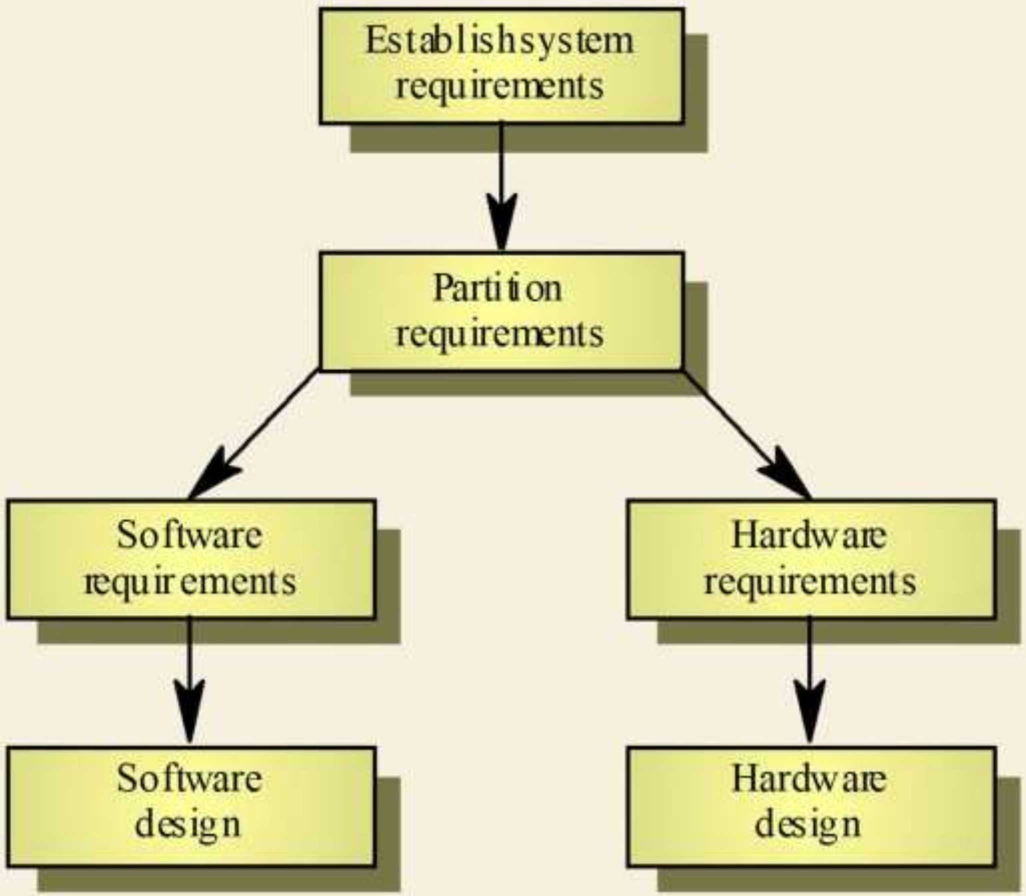
Source: *Software Engineering, Ian Sommerville*

- Sensors control processes
  - Collect information from sensors.
  - May buffer information collected in response to a sensor stimulus
- Data processor
  - Carries out processing of collected information and computes the system response
- Actuator control
  - Generates control signals for the actuator

SYSTEM DESIGN

- Design both the hardware and the software associated with system. Partition functions to either hardware or software
- Design decisions should be made on the basis on non-functional system requirements
- Hardware delivers better performance but potentially longer development and less scope for change

HARDWARE AND SOFTWARE DESIGN



R-T SYSTEMS DESIGN PROCESS

1. Identify the stimuli to be processed and the required responses to these stimuli

2. For each stimulus and response, identify the timing constraints

3. Aggregate the stimulus and response processing into concurrent processes. A process may be associated with each class of stimulus and response

4. Design algorithms to process each class of stimulus and response.

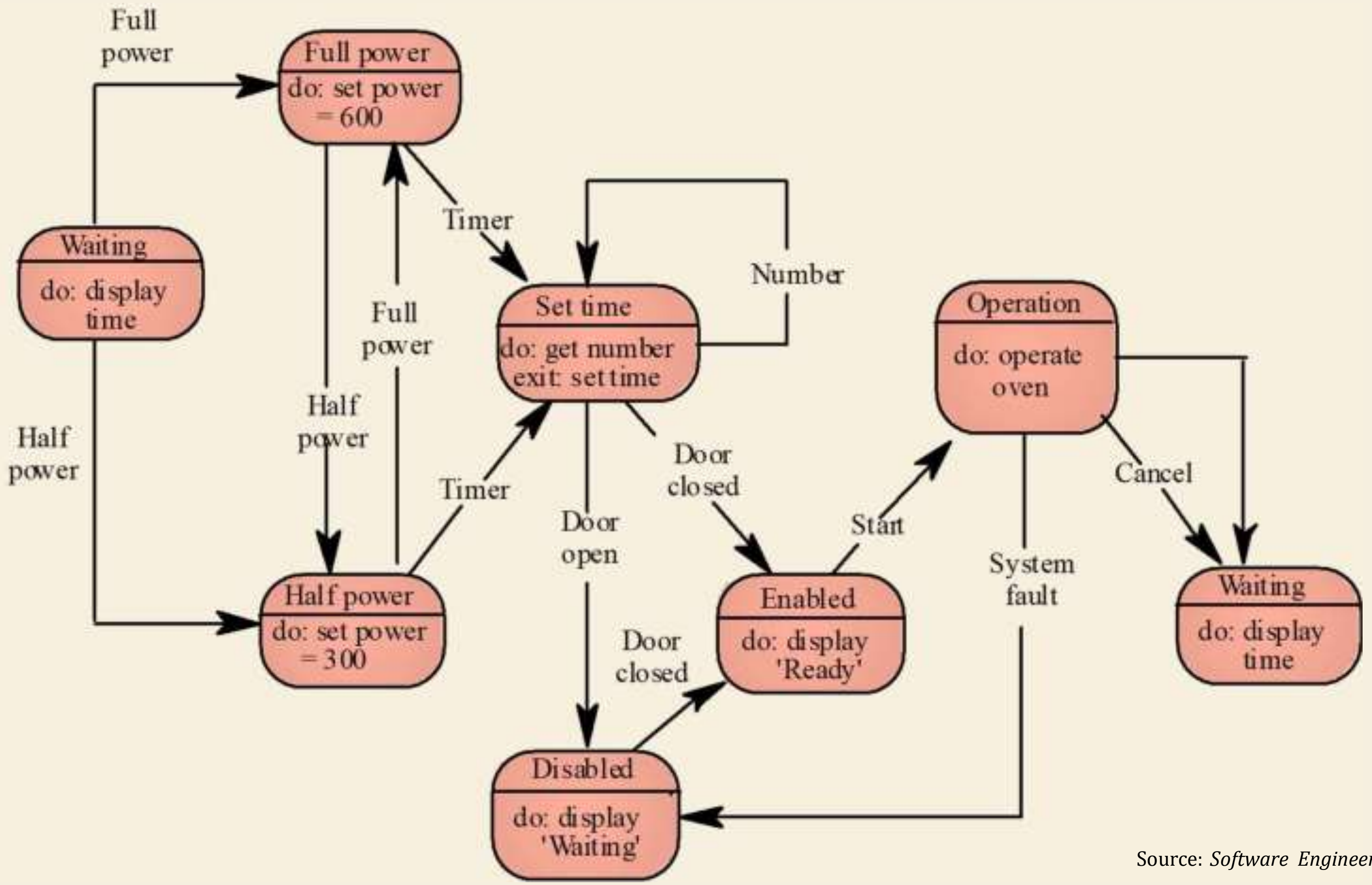
5. Design a scheduling system which will ensure that processes are started in time to meet their deadlines

6. Integrate using a real-time executive or operating system

STATE MACHINE MODELLING

- The effect of a stimulus in a real-time system may trigger a transition from one state to another.
- Finite state machines can be used for modelling real-time systems.
- However, FSM models lack structure. Even simple systems can have a complex model.
- The UML includes notations for defining state machine models

MICROWAVE OVEN STATE MACHINE



Source: *Software Engineering, Ian Sommerville*

ASSESSMENT

1. Match

a. Sensor — Response
b. Actuator — Time Sensor
c. Periodic Stimuli — Power Failure
d. Aperiodic Stimuli — Stimuli

2. Requirements are categorized as________________and ________________ requirements.
3. FSM Stands for ______________________________
4. UML ________________________________

REAL-TIME PROGRAMMING

- Hard-real time systems may have to programmed in assembly language to ensure that deadlines are met
- Languages such as C allow efficient programs to be written but do not have constructs to support concurrency or shared resource management
- Ada as a language designed to support real-time systems design so includes a general purpose concurrency mechanism

JAVA AS A REAL-TIME LANGUAGE

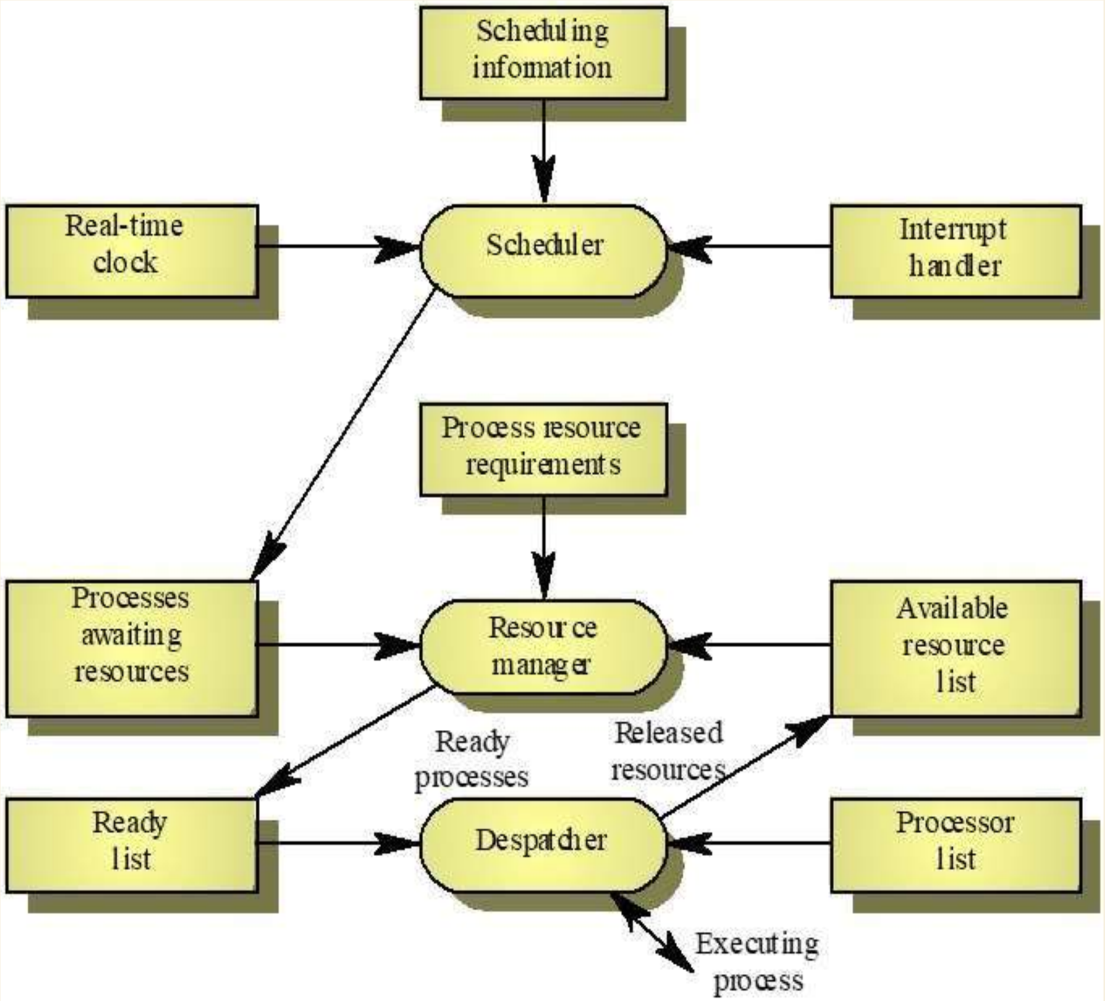
- Java supports lightweight concurrency (threads and synchonized methods) and can be used for some soft real-time systems
- Java 2.0 is not suitable for hard RT programming or programming where precise control of timing is required
  - Not possible to specify thread execution time
  - Uncontrollable garbage collection
  - Not possible to discover queue sizes for shared resources
  - Variable virtual machine implementation
  - Not possible to do space or timing analysis

REAL-TIME EXECUTIVES

- Real-time executives are specialised operating systems which manage the processes in the RTS
- Responsible for process management and resource (processor and memory) allocation
- May be based on a standard RTE kernel which is used unchanged or modified for a particular application
- Does not include facilities such as file management

RT EXECUTIVE COMPONENTS

- Real-time clock
  - Provides information for process scheduling.
- Interrupt handler
  - Manages aperiodic requests for service.
- Scheduler
  - Chooses the next process to be run.
- Resource manager
  - Allocates memory and processor resources.
- Despatcher
  - Starts process execution.



Source: *Software Engineering, Ian Sommerville*

NON-STOP SYSTEM COMPONENTS

- Configuration manager
  - Responsible for the dynamic reconfiguration of the system software and hardware.
  - Hardware modules may be replaced and software upgraded without stopping the systems
- Fault manager
  - Responsible for detecting software and hardware faults and taking appropriate actions (e.g. switching to backup disks) to ensure that the system continues in operation

PROCESS PRIORITY

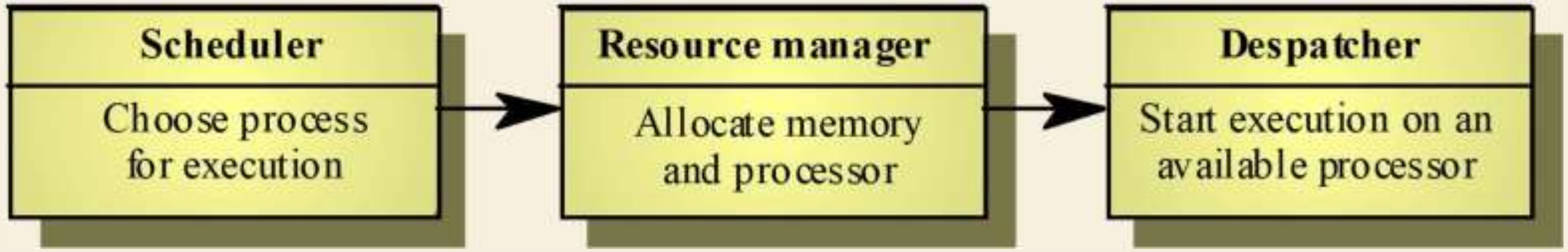
- The processing of some types of stimuli must sometimes take priority
- Interrupt level priority. Highest priority which is allocated to processes requiring a very fast response
- Clock level priority. Allocated to periodic processes
- Within these, further levels of priority may be assigned

INTERRUPT SERVICING

- Control is transferred automatically to a pre-determined memory location
- This location contains an instruction to jump to an interrupt service routine
- Further interrupts are disabled, the **interrupt serviced** and control returned to the interrupted process
- Interrupt service routines MUST be short, simple and fast

PROCESS MANAGEMENT

- Concerned with managing the set of concurrent processes
- Periodic processes are executed at pre-specified time intervals
- The executive uses the real-time clock to determine when to execute a process
- Process period - time between executions
- Process deadline - the time by which processing must be complete



PROCESS SWITCHING

- The scheduler chooses the next process to be executed by the processor.
- This depends on a scheduling strategy which may take the process priority into account
- The resource manager allocates memory and a processor for the process to be executed
- The despatcher takes the process from ready list, loads it onto a processor and starts execution

SCHEDULING STRATEGIES

- Non pre-emptive scheduling
  - Once a process has been scheduled for execution, it runs to completion or until it is blocked for some reason (e.g. waiting for I/O)
- Pre-emptive scheduling
  - The execution of an executing processes may be stopped if a higher priority process requires service
- Scheduling algorithms
  - Round-robin
  - Rate monotonic
  - Shortest deadline first

Reference

Software Engineering 6th Edition Ian Sommerville

Thank You