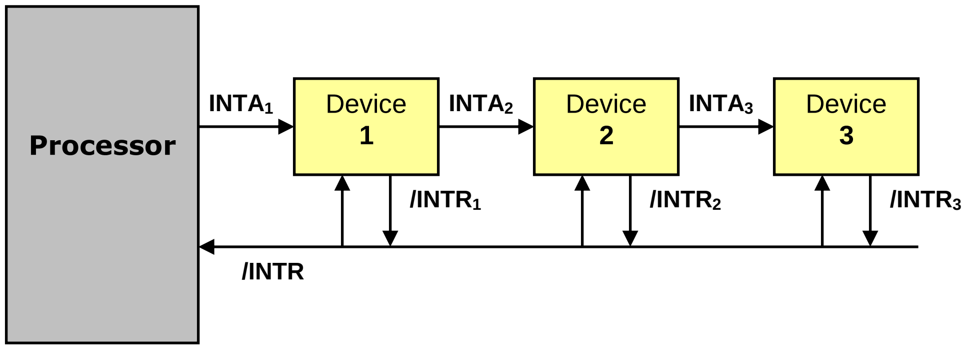
Solved Exercises 2

1. Consider the daisy-chain interrupt arrangement as shown below. Assume that: (1) the Processor asserts the interrupt-acknowledge once it receives a **1-0** transition on its interrupt-request input and enters an ISR, (2) an I/O device deasserts the interrupt-request once it receives a **0-1** transition on its interrupt-acknowledge input, and (3) the Processor deasserts the interrupt-acknowledge once it receives a **0-1** transition on its interrupt-request input.

Each device obeys the following rule: the interrupt-request may NOT be asserted when **/INTR = 0**. Is this rule necessary? Is this rule sufficient? Explain why.



The rule is necessary; otherwise, the system may deadlock. For example, assume that Device 3 asserted **/INTR$_3$**. The Processor received a **1-0** transition on **/INTR** and asserted **INTA$_1$**. Device 3 received a **0-1** transition on **INTA$_3$** and deasserted **/INTR$_3$**. Now let Device 1 violate the rule and assert **/INTR$_1$** before Device 3 deasserted **/INTR$_3$** (i.e., **/INTR** is still **0**). The Processor will never see a **0-1** transition on **/INTR** and will never deassert **INTA$_1$** (**INTA$_1$** becomes stuck at **1**). As Device 1 keeps waiting on a **0-1** transition on **INTA$_1$**, it keeps asserting **/INTR$_1$**. Thus, **/INTR** becomes stuck at **0**. The system hangs…

The rule is also sufficient. The only other input to each device is **INTA**, and any additional rules involving **INTA** are not necessary to ensure the correct operation of the system.

2. Consider the system from **Question 1**. Assume that the Processor automatically disables its interrupt-enable bit before entering an ISR. Also assume that each device has its own interrupt-enable bit: if cleared, the device cannot assert the interrupt-request. We want to allow for nested interrupts, i.e., Device 2 should be able to interrupt the ISR of Device 3, while Device 1 should be able to interrupt the ISR of either Device 2 or Device 3.

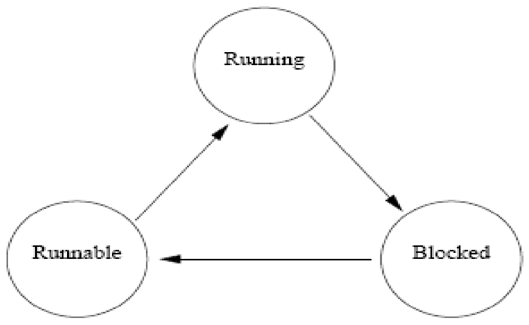
(a) How should each device's ISR manipulate interrupt-enable (IE) bits?
(b) Give an example illustrating some nested interrupt scenario.

(a)
ISR of **Device 1**: Set processor's IE at the end of the ISR.
ISR of **Device 2**: Clear Device 2's IE, clear Device 3's IE, set Device 1's IE, set processor's IE at the beginning of the ISR; set Device 2's IE, set Device 3's IE at the end of the ISR.
ISR of **Device 3**: Clear Device 3's IE, set Device 1's IE, set Device 2's IE, set processor's IE at the beginning of the ISR; set Device 3's IE at the end of the ISR.

(b) Nested interrupts may happen when a lower-priority ISR takes a sufficiently long time to finish. For example, let Device 3 request an interrupt. The processor enters the ISR for Device 3 and sends an interrupt-acknowledge. Device 3 receives the interrupt-acknowledge and deasserts **/BR$_3$**, thus making **/INTR = 1**. However, the ISR of Device 3 may still be in progress at this point. Once **/INTR** becomes deasserted, Device 1 can request an interrupt and get its ISR executed.

3. Solve Problem **4.10** from the textbook.

The transitions given in the question are shown below.

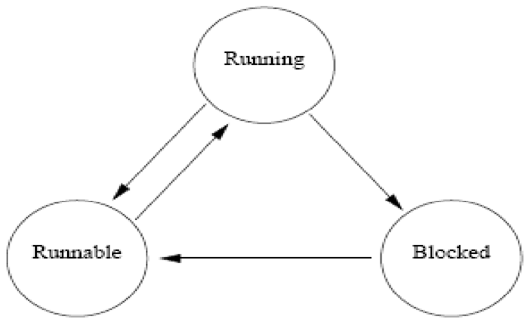


A possible additional transition from Running to Runnable occurs as a result of a timer interrupt at the end of a time slice. This causes the current process to be suspended so that another process can run.

A transition from Runnable to Blocked cannot occur directly. An I/O request must be made while in the Running state to cause a transition to the Blocked state.

A transition from Blocked to Running also cannot occur directly. An I/O request made by a process that is currently blocked eventually completes while that process is still blocked and while some other process is in the Running state. The process whose I/O request has been completed then moves from the Blocked state to the Runnable until it is granted an opportunity to resume execution as a consequence of time slicing.

The one additional transition discussed above is shown in the complete diagram below.



With three states, there can be only six possible transitions. Three valid transitions are described in the question. One valid transition and two invalid transitions are described above. All possible transitions have therefore been considered.

4. The example shown on **Slide 64** of the **"I/O"** lecture notes assumes that the tasks are assigned *Rate Monotonic* (**RM**) priorities. Show the task schedule using *Earliest Deadline First* (**EDF**) priority assignment.

Within the reference timeframe of 120, **T1** is activated 4 times with the deadlines of 30 (t=0, k=0), 60 (t=30, k=1), 90 (t=60, k=2), 120 (t=90, k=3). Therefore, **T1**'s priorities are τ_{10} = 1/30, τ_{11} = 1/60, τ_{12} = 1/90, τ_{13} = 1/120. Task **T2** is activated 3 times with the deadlines of 30 (t=0, k=0), 70 (t=40, k=1), 110 (t=80, k=2). Therefore, **T2**'s priorities are τ_{20} = 1/30, τ_{21} = 1/70, τ_{22} = 1/110. **T3** is activated once with the deadline of 120 (t=0, k=0); therefore, **T3**'s priority is τ_{30} = 1/120.

t=0: **T1** [τ_{10}], **T2** [τ_{20}], **T3** [τ_{30}] ready. Both **T1** and **T2** have the highest priority of 1/30. Dispatch **T1** (e.g., because its period is shorter than **T2**'s).

t=10: **T2** [τ_{20}], **T3** [τ_{30}] ready. **T2** has the highest priority of 1/30. Dispatch **T2**.

t=27: **T3** [τ_{30}] ready. Dispatch **T3**.

t=30: **T1** [τ_{11}], **T3** [τ_{30}] (WCET of 7 remains) ready. **T1** has the highest priority of 1/60. Dispatch **T1** (**T3** is suspended).

t=40: **T2** [τ_{21}], **T3** [τ_{30}] (WCET of 7 remains) ready. **T2** has the highest priority of 1/70. Dispatch **T2** (**T3** is suspended).

t=57: **T3** [τ_{30}] (WCET of 7 remains) ready. Dispatch **T3**.

t=60: **T1** [τ_{12}], **T3** [τ_{30}] (WCET of 4 remains) ready. **T1** has the highest priority of 1/90. Dispatch **T1** (**T3** is suspended).

t=70: **T3** [τ_{30}] (WCET of 4 remains) ready. Dispatch **T3**.

t=74: Idle – no tasks to execute.

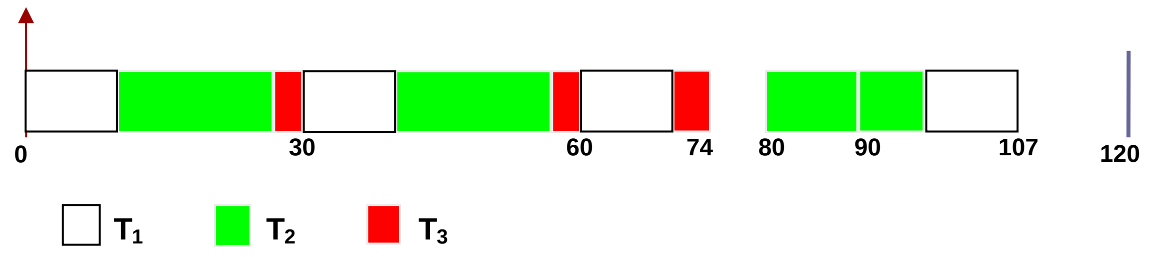
t=80: **T2** [τ_{22}] ready. Dispatch **T2**.

t=90: **T1** [τ_{13}], **T2** [τ_{22}] (WCET of 7 remains) ready. **T2** has the highest priority of 1/110. Dispatch **T2**.

t=97: **T1** [τ_{13}] ready. Dispatch **T1**.

t=107: Idle – no tasks to execute.

t=120: End.


0
30
60
74
80
90
107
120
T1
T2
T3