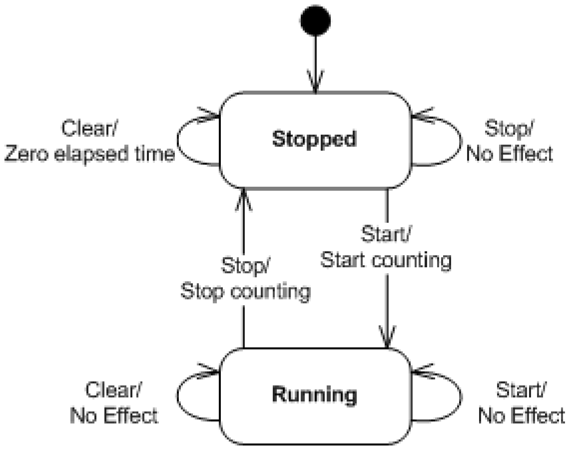
CG2271 Real-Time Operating Systems

Tutorial 1 (Solutions)

For the first three questions, consider a stopwatch as described here. The state machine below presents the desired behavior.



It has the following hardware.

Buttons for start, stop and clear functions.

- Pressing Start starts the stopwatch running. If pressed multiple times, stopwatch continues running without resetting elapsed time.
- Pressing Stop stops the stopwatch from counting.
- Pressing Clear zeroes out the elapsed time if the stopwatch is not running. If it is running, the clear button is ignored.

A timer which triggers an interrupt every 1 ms. The timer drives a counter which counts milliseconds since system start-up, and can be read as elapsed_time_counter.

A display to show elapsed time with 1 ms resolution. The display must be updated 10 times per second.

1. Design pseudocode for the software using event-triggered scheduling with interrupts. Assume that each button can generate an interrupt.

- Use a variable called state to indicate whether the stopwatch is stopped or running
- Use a variable called elapsed_time to track how much time has elapsed since the start button was pressed.
- Use a variable called display_delay to track how many milliseconds remain until the display needs to be updated again.

Answer:

```
Main thread:
state = stopped
display_delay = 100
      elapsed_time = 0


Start ISR:
state = running


Timer ISR:
if state == running
      elapsed_time += 1 ms
      display_delay -= 1
      if display_delay == 0 {
            display_delay = 100
            display elapsed_time
}


Stop ISR:
      state = stopped


Clear ISR
      if state == stopped
            elapsed_time = 0
```

2. Now design pseudocode for the software using a **static scheduler without using any interrupts**. Assume that the timer updates a hardware register called elapsed_time_register every millisecond.

- Use a variable called state to indicate whether the stopwatch is stopped or running
- Use a variable called start_time to record when the start button was pressed.
- Use a variable called stop_time to record when the stop button was pressed.
- Use a variable called next_display_update to indicate when the display needs to be updated next.

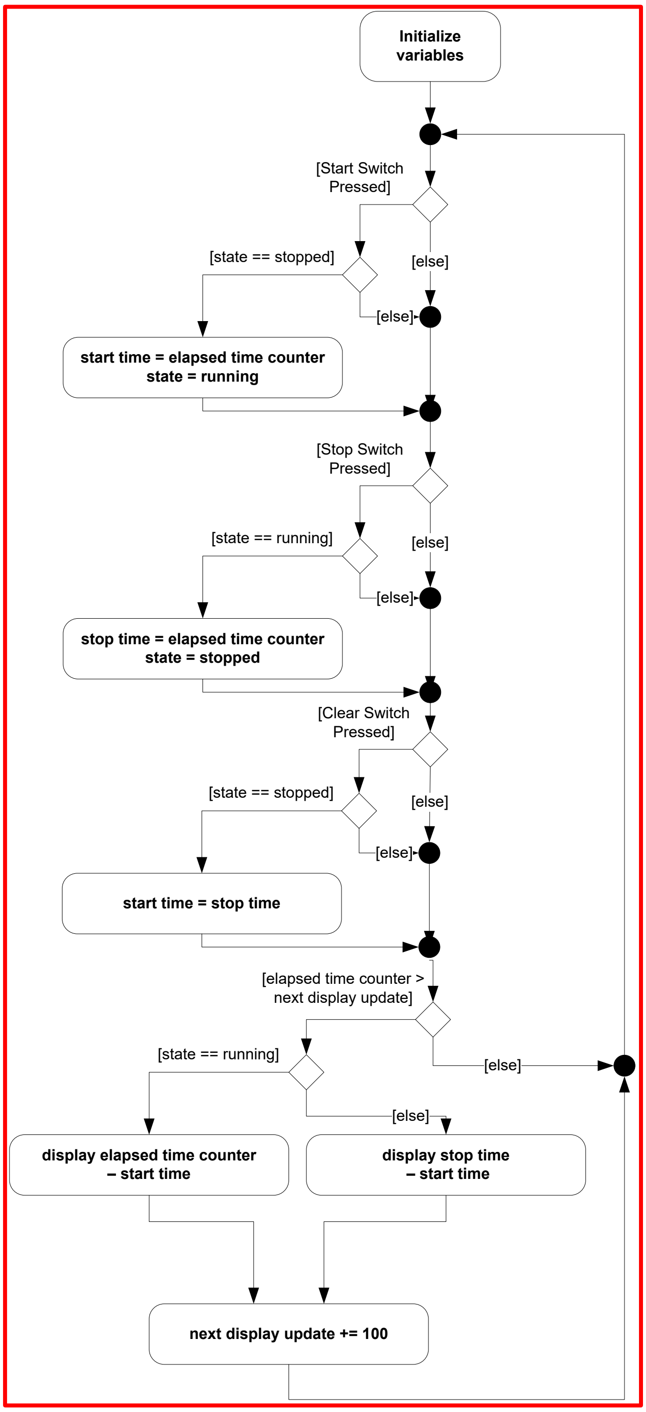
Answers:

```
state = stopped
display elapsed_time_counter
next_display_update = elapsed_time_counter + 100

while (1) {
      if start switch pressed {
            if state == stopped {
                  start_time = elapsed_time_counter
                  state = running
            }
      }
```

```
      if stop switch pressed {
            if state == running {
                  stop_time = elapsed_time_counter
                  state = stopped
            }
      }

      if clear switch pressed {
          if state == stopped {
            start_time = stop_time
          }
      }

      if elapsed_time_counter > next_display_update {
      if (state == running)
            display elapsed_time_counter - start_time
      else
            display stop_time – start_time

      next_display_update = next_display_update + 100
      }
}
```

3. Create a flowchart to represent your solution to the previous question.

Answer:



4. Consider a system with the following tasks. We wish to **minimize** the response time for task C. For each type of scheduler, describe the sequence of processing activities which will lead to the minimum and the maximum response times for task C. Assume that each task is ready to run and there are no further task releases.

Task	Duration
A	3
B	1
C	2

a. Static, non-preemptive scheduler

Answer:

Best Case: Task C starts immediately (at time 0). Tr = 0 + 2 = 2
Worst Case: Task A and Task B run first. Tr = 0 + 3 + 1 + 2 = 6

b. Dynamic, non-preemptive scheduler

Answer:

Best Case: Task C starts immediately (at time 0).
Worst Case: Longest task (A) just started running ε time units ago, so C won't run until it finishes. Tr = 0 + 3 – ε + 2 = 5 – ε

c. Dynamic, preemptive scheduler

Answer:

Best Case: Task C starts immediately (at time 0).
Worst Case: Longest task (A) just started running ε time units ago,
but it is preempted by C.
Tr = 0 + 2 = 2