ISLAMIC UNIVERSITY OF TECHNOLOGY (IUT)

ORGANISATION OF ISLAMIC COOPERATION (OIC)

Department of Computer Science and Engineering (CSE)

SEMESTER FINAL EXAMINATION WINTER SEMESTER, 2022-2023

DURATION: 3 HOURS FULL MARKS: 150

SWE 4739: Embedded Software Development

Programmable calculators are not allowed. Do not write anything on the question paper.

Answer **all 6 (six)** questions. Figures in the right margin indicate full marks of questions whereas corresponding CO and PO are written within parentheses.

---

1. a) Briefly explain 5 key challenges of developing an embedded system. 5 × 5 (CO1) (PO1)

   b) Describe how Harvard Architecture differs from Von Neumann Architecture.

   c) What is System-on-a-Chip (SoC)? How does an embedded system benefit from it?

   d) Briefly explain the purpose behind a hardware-software co-design flow.

   e) Describe any 5 types of Transparency goals in a Distributed System with examples.

2. An Answering Machine takes the following steps to service an incoming call:

   I. Initially, the machine stays in a "line-scanning" mode, where it senses the telephone line for any incoming call.

   II. When a call arrives, the machine rings for a total of 20 seconds and waits to be answered.

   III. If someone picks up the call within 20 seconds, then the machine goes into "communication" mode and stays there until hung up. After that, it goes back to the "line-scanning" mode.

   IV. However, if the call is not answered within 20 seconds, it plays a recorded message to the *caller* asking if they want to leave a message for the *callee*. It then waits for 3 seconds and plays a tone to let the caller know that it is ready to record the message.

   V. At this point, if the *caller* chooses not to leave any message and hangs up the call, the machine goes to the "hung-up" state and, in turn, again back to the "line-scanning" mode.

   VI. On the other hand, if the *caller* starts talking (to leave a message) after the tone, then the machine records for 30 seconds, plays another tone (to notify the end of the recording), and automatically goes to the "hung-up" state and to the "line-scanning" mode sequentially.

   a) i. Design a *Timed Automata* showing all the steps of the answering machine mentioned above. 2×10 (CO3) (PO3)

   ii. Convert the *Timed Automata* designed in 2.a)i. into an *SDL (Specification and Description Language)*.

   b) Design a NAND gate using Petri Nets. 5 (CO3) (PO3)

3. Assume, in a system, there are 3 jobs - $J1$, $J2$, and $J3$ with priorities $P1$ (highest), $P2$, and $P3$ (lowest), respectively. During the operation of the whole system, the jobs $J1$, $J2$, and $J3$ ask for CPU execution at times $t = 15$, $t = 17$, and $t = 0$, respectively. In addition, the system also has 2 shared resources - a and b, that can be used by the jobs.

   There are 3 corresponding programs for the 3 jobs given in Code Snippets 1, 2, and 3, respectively. Pay attention to the comments in the snippets for necessary explanations.

```
void J1(){
    initialize();       // duration to complete = 5 seconds

    /* start of critical section */
    P(b);
    work();             // duration to complete = 5 seconds
    V(b);
    /* end of critical section */

    wrap();             // duration to complete = 5 seconds
}
```

Code Snippet 1: Program for job J1 for Question 3.a)

```
void J2(){
    init();             // duration to complete = 5 seconds

    /* start of critical section */
    P(a);
    perform();          // duration to complete = 5 seconds
    V(a);
    /* end of critical section */

    close();            // duration to complete = 5 seconds
}
```

Code Snippet 2: Program for job J2 for Question 3.a)

```
void J3(){
    preproc();          // duration to complete = 5 seconds

    /* start of critical section */
    P(a);
    proc_1();           // duration to complete = 5 seconds
    P(b);
    proc_2();           // duration to complete = 10 seconds
    V(b);
    proc_3();           // duration to complete = 5 seconds
    V(a);
    /* end of critical section */

    postproc();         // duration to complete = 5 seconds
}
```

Code Snippet 3: Program for job J3 for Question 3.a)

a) i. Given the scenario above, draw the timing diagram showing the sequence of execution for every job and determine whether any Priority Inversion took place at any point in time. Assume no protocol is applied here. 2×10 (CO2) (PO2)

ii. Apply the Priority Inheritance Protocol and draw the updated timing diagram. Additionally, show how the priority of $J3$ changes over time in this case.

b) "Priority Ceiling manages deadlock better than Priority Inheritance" - justify with example. 5 (CO1) (PO1)

4. Consider a hypothetical algorithm called *'Fast Signal Processing'*, given in Algorithm 1, containing the simplest instruction set. Note that the algorithm has a total of 13 instructions and is shown in 3 columns along with a corresponding line number and a colon.

Algorithm 1 Algorithm for Fast Signal Processing for Question 4.

1: $i := 5 \times 9$
2: $j := 5 \div 10$
3: $x := i \times 9$
4: $i := i + 1$
5: $m := i + j$
6: $k := i \times j$
7: $i := i + 1$
8: $y := x + k$
9: $k := k + 1$
10: $a := i - k$
11: $b := m + k$
12: $c := k - a$
13: $z := y + k$

a) i. Design an *ASAP (As Soon As Possible)* and an *ALAP (As Late As Possible)* scheduling for Algorithm1. Assume that you have 3 Functional Units that can execute any operation. 2×10 (CO3) (PO3)

ii. Design Resource-constrained scheduling from Question 4.a)i. if you are only allowed a maximum of two Functional Units. Show ready queue for every step.

b) Analyzing the *ASAP* scheduling in Question 4.a), determine whether it is possible to complete the algorithm within 4 time-steps, even with unlimited Functional Units. You do NOT need to design the Time-constrained scheduling. 5 (CO2) (PO2)

5. a) i. Compare between Merging of tasks and Splitting of tasks. 2 × 5 (CO1) (PO1)

ii. How does Array Folding optimize memory size? Explain with diagrams.

b) Code Snippets 4 and 5 show C programs that perform the same task - accessing multi-dimensional arrays. Considering run-time optimization, which one would you prefer and why? 5 (CO2) (PO2)

```
1 for (int i = 0; i < 1000; i++){
2     for (int j = 0; j < 1000; j++){
3         for (int k = 0; k < 1000; k++ ){
4             array[i][j][k]; }
5     }
6 }
```

Code Snippet 4: A C Program for Accessing Multi-dimensional Array for Question 5.b)

```
1 for (int y = 0; y < 1000; y++){
2     for (int z = 0; z < 1000; z++ ){
3         for (int x = 0; x < 1000; x++){
4             array[z][y][x]; }
5     }
6 }
```

Code Snippet 5: Another C Program for Accessing Multi-dimensional Array for Question 5.b)

c) Suppose, a program consists of 6 basic blocks: *a, b, c, d, e,* and *f*. The sequence of access (SoA) for these blocks is the following: 10 (CO3) (PO3)

a, b, c, d, e, f, a, d, a, f, a, d, c, d, a, c, d, f, a, b, c, d, a, d.

Develop an optimized layout for the given blocks by applying Liao's Algorithm. Show step-by-step process of your layout with proper reasoning.

6. a) Determine whether the two Finite State Machines (FSMs) given in Figure 1 are equivalent. 10 (CO2) (PO2)

(a) Finite State Machine 1 **(b)** Finite State Machine 2

Figure 1: Two Finite State Machines for Question 6.a)

b) Determine, with reason, whether the model in Figure 2 satisfies the following properties: 10 (CO2) (PO2)

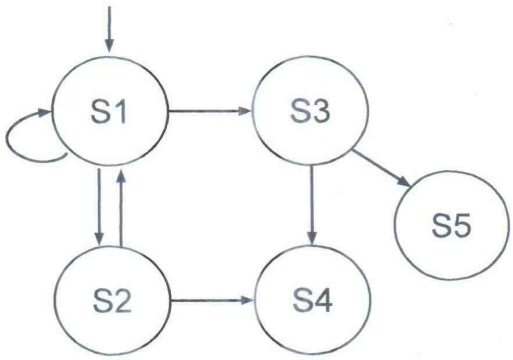

Figure 2: Finite State Machine for Question 6.b)

i. We will always find S1 in our path.
ii. S2 eventually leads to S4
iii. For some path, S4 does NOT occur.
iv. At least for some paths, both S4 and S5 will occur.
v. S2 will never come after S3.

c) How do embedded systems impact NEGATIVELY on the environment? What measures can be taken in this regard? 5 (CO1) (PO1)