Department of
Computer Science and Engineering

University of Dhaka

Assignment 1: ARM-Based Healthcare Monitoring & Billing System

CSE 2106: Microprocessor and Assembly Language Lab

Batch: 30/2nd Year 1st Semester 2025

SmartCare-32: ARM-Based Healthcare Monitoring & Billing System

The year is 2025, and DU Medical Center is planning to modernize its internal patient monitoring and billing system. The current system is slow, fragmented, and heavily dependent on manual entry, causing delays in patient care and inconsistencies in billing.

The hospital purchases a set of ARM Cortex-M microcontroller units, intending to build a dedicated embedded system named SmartCare-32. This system must:

- Continuously monitor patient vitals
- Generate emergency alerts
- Track medication schedules
- Compute treatment and room charges
- Aggregate full billing details
- Store logs securely

Finally, present the summary through serial output. The hospital assigns your development team to build the entire firmware for SmartCare-32 from scratch in ARM Assembly using Keil uVision.

Module 1 — Patient Record Initialization

Story Context: Whenever a patient is admitted, their personal and medical data must be stored in the SmartCare-32 memory.

Technical Requirements:

- Allocate memory for a patient structure containing:
  - Patient ID (32-bit)
  - Name pointer (32-bit)
  - Age (8-bit)
  - Ward number (16-bit)
  - Treatment code (8-bit)
  - Room Daily Rate (32-bit)
  - Pointer to Medicine List (32-bit)
- Initialize each field with test values defined by the student.
- Store the structure at a predefined RAM block (e.g., `0x20000000`).
- Ensure correct alignment (word/halfword/byte).

Module 2 — Vital Sign Data Acquisition

Story Context: The patient is connected to vital sensors (simulated using memory).

Technical Requirements:

- Read simulated vital values (HR, BP, O_2) from fixed memory addresses.
- Maintain a rolling buffer of 10 entries per vital.
- Implement buffer rotation using modulo arithmetic.
- Use LD/ST instructions and pointer increment logic.

Module 3 — Vital Threshold Alert Module

Story Context: The system must warn the nurse if a patient's vitals exceed dangerous limits.
Technical Requirements:

- Compare new readings with thresholds:
  - HR > 120
  - $O_2 < 92$
  - SBP > 160 or < 90
- On threshold violation:
  - Set 1-byte `ALERT` flag.
  - Create a 16-byte alert record containing:
    - Vital type
    - Actual reading
    - Timestamp (counter)
  - Insert the record into an alert buffer in RAM.

Module 4 — Medicine Administration Scheduler

Story Context: Medicines must be administered at fixed intervals.
Technical Requirements:

- For each medicine store:
  - Dosage interval (hours)
  - Last administered timestamp
- Compute:

```
next_due_time = last + interval
```

- Use an internal clock counter.
- Set "`DOSAGE DUE`" flag when current time $\geq$ next_due_time.

Module 5 — Treatment Cost Computation

Story Context: Each treatment code corresponds to a fixed cost.
Technical Requirements:

- Create a lookup table mapping treatment codes to cost values.
- Fetch treatment code from patient structure.
- Store computed treatment cost in billing structure.

Module 6 — Daily Room Rent Calculation

Story Context: Room charges depend on the length of stay and ward category.
Technical Requirements:

- Read daily room rate from patient structure.
- Read number of stay-days from input.
- Compute:

$$room\_cost = rate \times days$$

- If days > 10, apply 5% discount using fixed-point math.

Module 7 — Medicine Billing Module

Story Context: Each medicine has cost parameters.
Technical Requirements:

- Load medicine list pointer.
- For each entry, compute:

$$med\_cost = unit\_price \times quantity \times days$$

- Accumulate cost into billing structure.

Module 8 — Patient Bill Aggregator
Story Context: The system must compute the final payable bill.
Technical Requirements:

- Compute:

```
total_bill = treatment + room + medicine + lab_tests
```

- Perform overflow checking.
- Set an error flag on overflow.
- Store final bill in billing memory block.

Module 9 — Sorting Patients by Criticality

Story Context: ICU triage prioritizes patients with more alerts.
Technical Requirements:

- Sort multiple patient structures based on alert count.
- Use bubble sort or selection sort.
- Swap entire structures in RAM.

Module 10 — UART Summary Report Generator

Story Context: Doctors must receive a quick summary through a serial terminal.
Technical Requirements:

- Generate a formatted summary including:
  - Patient ID
  - Age
  - Ward
  - Latest vitals
  - Total alerts
  - Billing summary
- Steps:
  - Initialize UART.
  - Convert integers to ASCII.
  - Transmit byte-by-byte using UART DR register.

Module 11 — Anomaly Detection & Safety Check

Story Context: System must detect abnormal conditions or failures.
Technical Requirements:

- Check for:
  - Sensor malfunction (same value repeated > 10 times)
  - Invalid medicine dosage (zero values)
  - Memory overflow (address above boundary)
- If detected:
  - Set `ERROR FLAG`
  - Store error record in Flash

Expected Outcome

A fully integrated ARM Assembly program that:

- Monitors vitals
- Generates alerts
- Manages medicine schedules
- Computes complete billing
- Sorts patients for triage priority
- Logs all system actions securely
- Outputs a final summary via UART

Students must demonstrate:

- Modular programming in ARM Assembly
- UART communication
- Algorithmic implementation (sorting, scheduling, billing)

Policy

Copying from the Internet, classmates, seniors, or from any other source is strongly prohibited. 100% marks will be *deducted* if any such copying is detected.