**Project 1: Traffic Signal Controller in MIPS Assembly**

**Objective: Design and implement a traffic signal controller simulation using MIPS assembly language. This project aims to emulate the logic behind traffic light control at an intersection, cycling through red, green, and yellow lights in a sequence that ensures efficient and safe traffic flow.**

- **Develop a program that cycles through the sequence of traffic lights (red, green, yellow) for each direction (e.g., North-South, East-West). Implement appropriate timing for each light, ensuring realistic traffic flow.**
- **Allow the user to input via the console to start the simulation, and if desired, to change the duration of each light during the simulation.**

**Basic Logic Implementation:**

**Implement a safety feature where the yellow light duration is adjusted based on the speed limit of the road, ensuring vehicles have sufficient time to stop.**

**Introduce pedestrian crossing signals that synchronize with the traffic lights, including a button press simulation to request crossing.**

- **Utilize the MIPS assembly language and the MARS simulator for development and testing.**
- **Implement a modular design, with separate procedures for each major functionality (e.g., initializing the lights, changing lights, timing control).**
- **Use system calls for input/output operations.**
- **Source code in MIPS assembly language, with comments explaining the functionality of code blocks and instructions.**

**A detailed in-class presentation (15 minutes) that includes: Project overview and objectives.**

**Design and implementation strategy, including how the timing and sequences are managed.**

**Challenges encountered and solutions applied.**

**Suggestions for further improvement or additional features.**

**Project 2: Elevator Control System in MIPS Assembly**

**Create an elevator control system simulation using MIPS assembly language. This project aims to simulate the basic operations of an elevator, handling floor requests and managing elevator movements efficiently.**

**Objective:**

**Description:**

- **Implement a system that simulates an elevator capable of serving a fixed number of floors (e.g., 5 floors). The program should be able to take floor requests and move the elevator accordingly, displaying the current floor and direction of movement.**
- **Develop a request handling system that efficiently manages multiple floor requests, prioritizing them based on the current direction of the elevator and its position to minimize wait times.**

**Request Queue:**

**Basic Elevator Logic:**

**Incorporate emergency functionalities, such as an emergency stop that halts the elevator until reset, and an alarm system that can be activated by the user.**

**Emergency Features:**

**Technical Requirements:**

- **Use MIPS assembly language programming and test with the MARS simulator.**
- **Structure the program with clear, modular code sections for initialization, request handling, movement control, and user interface.**
- **Employ system calls for basic input/output operations.**

**Deliverables:**

- **MIPS assembly source code with comprehensive comments to explain the purpose and functionality of the code segments.**

**A detailed in-class presentation (15 minutes) that includes: The project scope and objectives.**

**The architecture of the elevator control system, including the logic behind request handling and movement control.**

**The implementation process, highlighting any obstacles faced and the resolutions found.**

**A user manual describing how to interact with the simulation, including floor request submission and emergency operations.**

**Future enhancements or additional features that could be added to improve the simulation**

**Marking Scheme: Project 1 — Traffic Signal Controller in MIPS Assembly**

| Component | Description | Marks |
|---|---|---|
| **1. Report / Documentation** | | **25** |
| • Objective & Problem Description | Clear explanation of what is being simulated and why | 5 |
| • Code Structure & Modularity | How the project is broken into functions/procedures | 5 |
| • Traffic Signal Logic Explanation | Description of the red/green/yellow cycle logic per direction | 5 |
| • Pedestrian & Safety Features | Description of speed-based yellow duration and pedestrian request logic | 5 |
| • Challenges & Future Enhancements | Summary of issues faced, and suggestions for further features | 5 |
| **2. Code Implementation** | | **40** |
| • Functional Signal Simulation | Correct red-green-yellow sequencing for two directions | 10 |
| • Timing Logic | Light durations implemented with appropriate loops/delays | 5 |
| • User Input Handling | Modify durations during runtime (via console) | 5 |
| • Pedestrian Button Logic | Simulated pedestrian crossing integrated into signal sequence | 5 |

| Component | Description | Marks |
|---|---|---|
| • Safety Feature (yellow duration) | Adjusts based on given speed input | 5 |
| • Modular Code and Comments | Use of procedures; code is clean, well-commented | 5 |
| • Use of Syscalls | Input/output handled via proper MIPS system calls | 5 |
| **3. In-Class Presentation (15 min)** | | **25** |
| • Project Overview & Motivation | Clear explanation of purpose and importance of simulation | 5 |
| • Design & Logic Explanation | Explaining timing, light switching, and any user input modules | 10 |
| • Demo & Simulation Walkthrough | A live or recorded demo of a functioning system | 5 |
| • Delivery & Response to Questions | Communication clarity, confidence, and ability to answer queries | 5 |
| **4. Code Testing / Edge Cases** | Demonstrate correctness through multiple test cases | **10** |
| • Normal Case | Complete light sequence with default timings | 5 |
| • Edge Cases | Changes in durations, rapid pedestrian requests, etc. | 5 |

💯 **Total Marks: 100**

---

✅ **Marking Scheme: Project 2 — Elevator Control System in MIPS Assembly**

| Component | Description | Marks |
|---|---|---|
| **1. Report / Documentation** | | **25** |

| Component | Description | Marks |
|---|---|---|
| • Objective & Problem Statement | Explaining what is simulated and its real-world relevance | 5 |
| • Elevator Architecture Description | Logic behind floors, direction, and movement | 5 |
| • Request Queue & Prioritization | Explanation of how floor requests are handled | 5 |
| • Emergency & Alarm Features | How stop/reset/alarm works | 5 |
| • Challenges Faced & Enhancements | Thoughtful discussion of what could be added/improved | 5 |
| **2. Code Implementation** | | **40** |
| • Movement Between Floors | Elevator accurately moves between floors with updated current state | 10 |
| • Request Queue Handling | Prioritizing requests based on current direction and floor | 5 |
| • Emergency Stop/Reset | Working emergency stop feature, resuming from same state | 5 |
| • Alarm Functionality | Triggered and displayed via user input | 5 |
| • Modular Code with Comments | Clear procedures for movement, I/O, request handling | 5 |
| • Console Interaction with Syscalls | Input/output via syscall for floor request, emergency button | 5 |
| • Error Handling | Rejection of invalid floors or repeated requests | 5 |
| **3. In-Class Presentation (15 min)** | | **25** |

| Component | Description | Marks |
|---|---|---|
| • System Overview & Objective | Explained the system's intent and elevator design | 5 |
| • Logic & Modules Breakdown | Verbal explanation of movement logic and how it adapts to requests | 10 |
| • Simulation Demo | Demonstrated working code for floor requests, movement, and emergency | 5 |
| • Communication & Engagement | Clarity in explanation and response to audience questions | 5 |
| **4. Simulation Testing** | | **10** |
| • Normal Request Handling | Test 3–4 random requests and show handling order | 5 |
| • Emergency Case Simulation | Trigger stop or alarm and show proper resume/reset | 5 |