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.

Description:

- **Basic Logic Implementation:** 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.
- **User Interaction:** 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.
- **Safety Features:** 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.
- **Advanced Features (Optional):** Introduce pedestrian crossing signals that synchronize with the traffic lights, including a button press simulation to request crossing.

Technical Requirements:

- 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.

Deliverables:

- Source code in MIPS assembly language, with comments explaining the functionality of code blocks and instructions.
- A detailed report that includes:
    - Project overview and objectives.
    - Design and implementation strategy, including how the timing and sequences are managed.
    - Challenges encountered and solutions applied.
    - Instructions on how to run the simulation in the MARS simulator.

- (Optional) Suggestions for further improvement or additional features.

Project 2: Elevator Control System in MIPS Assembly

Objective: 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.

Description:

- **Basic Elevator Logic:** 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.
- **Request Queue:** 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.
- **Emergency Features:** 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.

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 comprehensive report detailing:
    - 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.