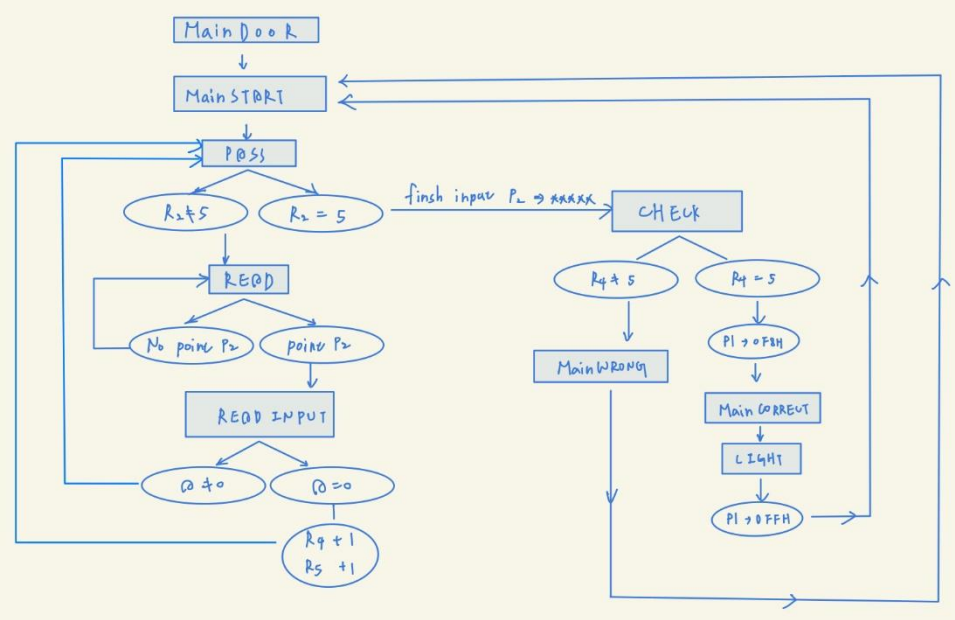
Microprocessor Project Group 3.
Name/Student ID/E-mail: Gina/B1027145/ B1027145@cgu.edu.tw
Riva/B1027148/ B1027148@cgu.edu.tw

- Part1

Flowchart



The program begins by initializing the serial port, timer, and baud rate parameters. It then prints a prompt asking the user to input their password. A variable R2 keeps track of the number of password characters entered. The program execution jumps to a READ loop where it reads input from serial port 2. For each character entered, the program checks if it matches the correct password stored in memory. A counter variable R4 increments by one for every correct character entered.

Once the full password has been input, execution jumps to a CHECK routine that validates if the entered password fully matches the correct one. If R4 equals 5, indicating a full 5-character password has been correctly entered, the program prints "PIN CORRECT!" and sets a control pin low, simulating the unlocking of a door. It then prints "DOOR OPENED!" to confirm. After a brief delay, the program sets the control pin high again, simulating the closing of the door.

If at any point an incorrect password character is entered, the program will print "PIN WRONG!" and loop back to the beginning to await the next password entry attempt. This way the system provides user feedback but also secures the door access until the full, correct password is provided.

The structure allows for secure password validation and appropriate system control reactions, while providing transparency to the user regarding state via printed messages at each step.

- Part2

Flowchart

The provided flowchart illustrates the sequential execution of an assembly code program designed for microcontroller or embedded systems. This program orchestrates a series of actions, starting with an initialization phase configuring timers, setting registers, and initializing variables. The flow then moves into a standby state, where specific ports are configured and a loop is established until a predefined condition is met. Subsequently, the program enters a cyclic activation phase for four terminals, each with its own set of instructions. A counter is decremented during each iteration, and when it reaches zero, the program returns to standby.

1: initialization

The program starts with initialization, configuring Timer 0 in mode 1, and clearing the Timer 0 control register. It also initializes a counter (R3) and sets a value in the external data memory at address 33H.

2: standby state

Moving on to the standby state, various ports are set to specific values. Notably, Port 1 pin 3 is set to 1, and the program enters a loop labeled 'standby' until a condition (P1.0 being set) is met.

3: four terminal activation sections

The program then progresses to four consecutive terminal activation sections (labeled `activate_terminal_1` through `activate_terminal_4`). In each, Timer 0 is configured, Port 0 is set to specific values, and Timer 0 is reset. Additionally, there is a decrement operation on the counter (R3) and a conditional jump (`CJNE R3, #0, activate_terminal_1`) to repeat the activation process until the counter reaches zero. The code concludes with a check for the counter being zero (`AJMP standby`), and if so, it jumps back to the standby state. The 'END' directive marks the end of the program.

4: Conclusion

In summary, this assembly code initializes a system, enters a standby state until a certain condition is met, and then cyclically activates terminals, decrementing a counter until it reaches zero, at which point it returns to standby. This structure suggests a system managing a sequence of terminal activations with a predefined delay between them.