Microprocessors and
Microcontrollers (CCE403)

DOOR-LOCK SYSTEM REPORT

USING 8051 MICROCONTROLLER

UNDER THE SUPERVISION OF:
Eng. Anas Elsayed

Presented By:

Name:	ID:
Nourhan Farag Mohamed	231903707
Farida Waheed Abdel Bary	221903168
Malak Mounier Abdellatif	231903643
Razan Ahmed Fawzy	221903165
Nour Hesham Elsayed	221902960
Lujain Ahmed Yousef	231903614
Raneem Ahmed Refaat	221903114

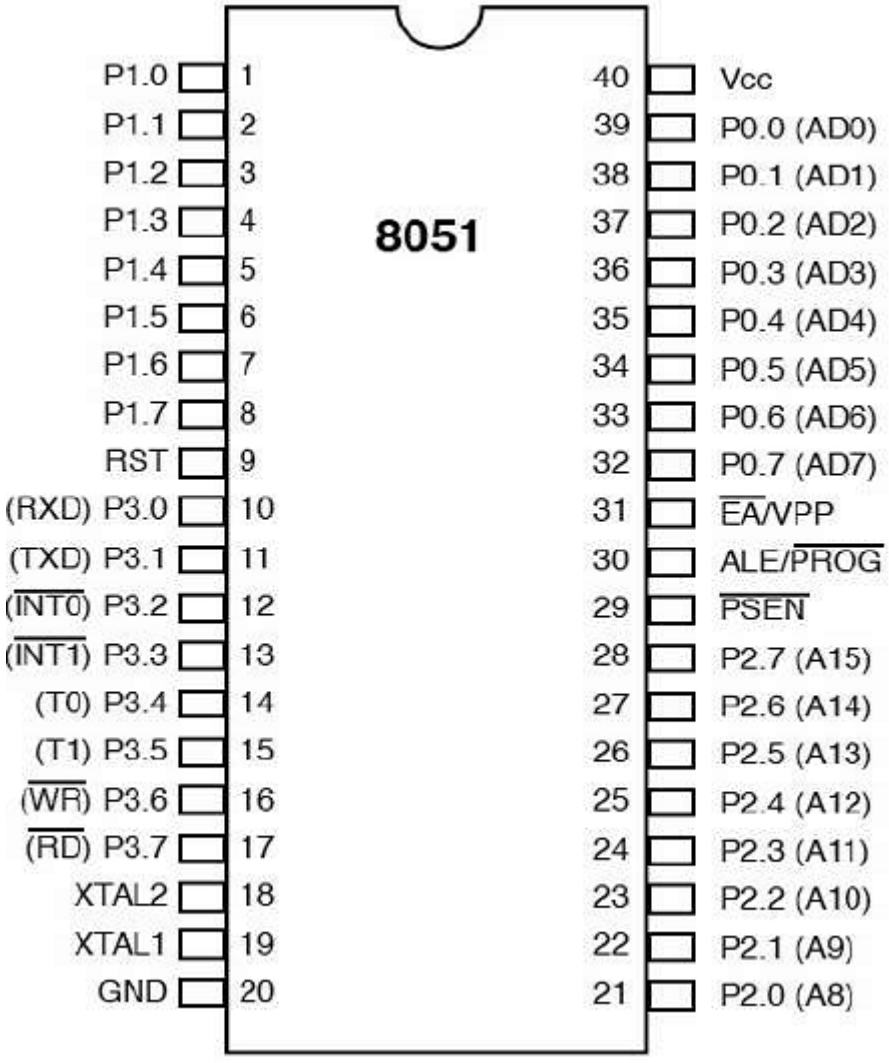
1. Project Overview:

- **Purpose:**

This project focuses on building a **Password-Based Door Lock System.** The purpose of this project is to design and implement a secure door lock system using an 8051 microcontroller. The system integrates an LCD, keypad, LEDs and buzzer to provide an interactive and secure locking mechanism.

- **Scope:**

The system allows access only when the correct password is entered. Upon successful entry, the door unlocks for a limited duration of 5 seconds before automatically locking again. The system monitors 3 incorrect password attempts, providing feedback through the LCD and triggering an alarm with each time the password entered wrong and after three consecutive failures to alert against potential intruders.



2. System Components:

8051 Microcontroller serves as the brain of the system, coordinating all inputs and outputs.

- **Pin Description**

PIN/PORT	PIN NUMBERS	FUNCTION	DESCRIPTION
PORT 0	32-39	8-bit, Bidirectional I/O	Used for both general-purpose I/O and as a multiplexed address/data bus for external memory. It needs external pull-up resistors because it does not have internal pull-ups.
PORT 1	1-8	8-bit, Bidirectional I/O	It has internal pull-ups and is used primarily for general-purpose I/O.
PORT 2	21-28	8-bit, Bidirectional I/O	It has internal pull-ups and can also function as the address bus for external memory.
PORT 3	10-17	8-bit, Bidirectional I/O	It has internal pull-ups and in addition to general-purpose I/O, Port 3 has several alternate functions: • **P3.0 (RXD)**: Serial input pin for UART. • **P3.1 (TXD)**: Serial output pin for UART. • **P3.2 (INT0)** : External interrupt 0. • **P3.3 (INT1)** : External interrupt 1. • **P3.4 (T0)** : Timer 0 external input. • **P3.5 (T1)** : Timer 1 external input. • **P3.6 (WR)**: External memory write strobe. • **P3.7 (RD)**: External memory read strobe.
VCC	40	Power Supply	Provides +5V power to the microcontroller.
GND	20	Ground	Ground connection for the microcontroller.
XTAL1	19	Clock Input	Connects to an external crystal oscillator to generate the clock signal.
XTAL2	18	Clock Output	Connects to an external crystal oscillator to generate the clock signal.
RST	9	Reset Input	Resets the microcontroller when a high logic level is applied for at least two machine cycles.
ALE	30	Address Latch Enable	Demultiplexes the address-data bus.
PSEN	29	Program Store Enable	Used to read external program memory.
EA	31	External Access Enable	Executes code from internal memory when high or from external memory when low.

- **Working of Specific Pins**

PIN	PIN NUMBERS	FUNCTION	DESCRIPTION
XTAL1, XTAL2	18, 19	Clock Input/Output	Connected to an external crystal oscillator to provide the necessary clock signal for the microcontroller's operation. The clock signal determines the speed at which the microcontroller executes instructions. Typically, a 11.0592 MHz crystal is used.
RST	9	Reset Pin	The reset pin is used to initialize the microcontroller. When a high signal is applied to this pin for at least two machine cycles, the microcontroller resets, and the program execution starts from the beginning.

Oscillator Circuit and Clock Signal

- **XTAL1 and XTAL2:** Connected to an external crystal oscillator to generate a precise clock signal, with frequency determined by the crystal's characteristics.
- **Capacitors (C1, C2):** Used to stabilize the oscillations, ensuring a consistent clock frequency.
- **Clock Signal Usage:** Synchronizes the microcontroller's internal operations, timing each instruction execution.
- **Internal Operations:** Drives timers, counters, and other timing-related functions within the microcontroller.

The clock signal is essential for the microcontroller's performance, determining instruction speed, ensuring precise timing for control and communication, and synchronizing internal processes for smooth operation.

LCD is connected to 8051as follows:

- **RS (Register Select):** Connected to Port 2 Pin 0.
- **RW (Read/Write):** Connected to Port 2 Pin 1.
- **E (Enable):** Connected to Port 2 Pin 2.
- **Data Pins (D0-D7):** Connected to Port 3 Pins 0-7.

By connecting these pins to the microcontroller, the system can control the LCD to display messages, prompt for password entries, and provide feedback to the user.

Keypad is connected to 8051 as follows:

- **Rows:** Connected to Port 1 Pins 0-3.
- **Columns**: Connected to Port 1 Pins 4-7.

This configuration allows the microcontroller to scan the keypad 4x3 matrix and detect which key is pressed by the user.

LEDs are connected to 8051 as follows:

- **Red LED (D1):** Connected to Port 2 Pin 3.
- **Green LED (D2):** Connected to Port 2 Pin 4.

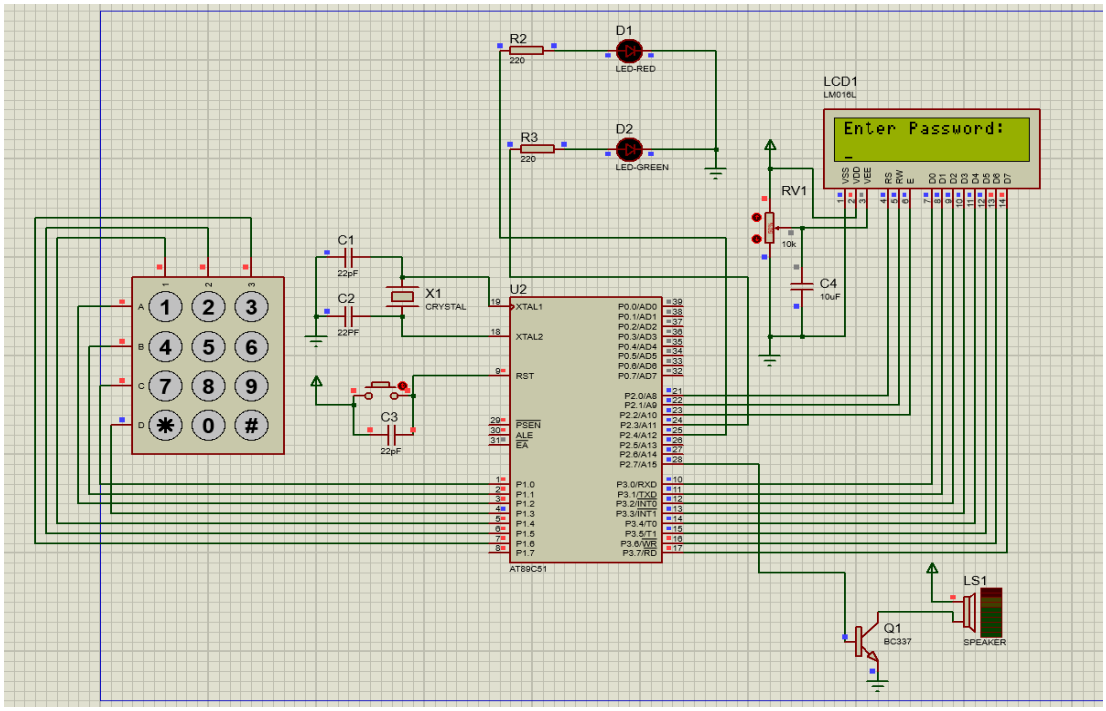
By connecting these pins to the microcontroller, the system can control the LEDs to indicate password status. The green LED turns on when the password is correct, while the red LED turns on when the password is incorrect, providing visual feedback to the user.

Buzzer is connected to the microcontroller via a transistor to Port 0 Pin 0. The transistor acts as a current amplifier, ensuring the buzzer receives enough current and voltage to operate properly. A buzzer is used to provide audio feedback for incorrect password entries and to alert against potential intruders after three incorrect attempts.

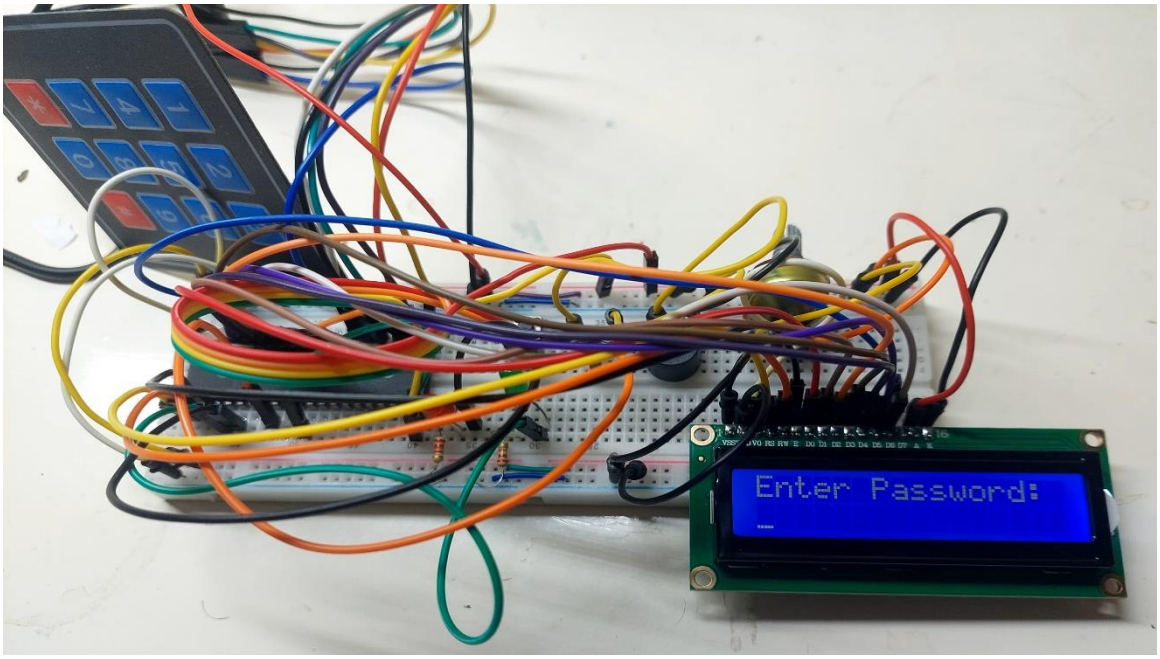
Additional Components

- **Resistors** and **capacitors** for circuit stability.
- **Power supply** to provide the necessary voltage and current for the components.

3. Simulation Circuit Diagram:



4. Circuit on real life:



5. Algorithm:

1. Initialize the system and display a welcome message on the LCD.
2. Wait for user input from the keypad.
3. Read and store the entered password.
4. Compare the entered password with the stored password.
   - If the password is correct:
     - Unlock the door.
     - Display "Access Granted", then “Opening Door” on the LCD.
     - Wait for 5 seconds.
     - Lock the door.
     - Display "Closing Door" on the LCD.
   - If the password is incorrect:
     - Increment the attempt counter.
     - Display "Wrong Pass" on the LCD.
     - If three consecutive incorrect attempts are detected:
       - Trigger the buzzer.
       - Display "Intruder Alert" on the LCD.

6. Code Implementation:

- **Program Initialization:** The code starts with setting the number of attempts and initializing the LCD. Set the origin to 0000H, the starting address. Initialize the number of attempts. Clear the accumulator. Clear Port 2.

```
1  ORG 0000H
2  MOV R5,#3D ;number of attempts
3  MOV A,#0
4  MOV p2, #00h
```

- **Display Start-Up Message:** The start-up message is displayed on the LCD using LCD_INIT and SEND_DAT subroutines.

```
5  ;---------------------------------
6  Start_up_MSG:
7  ACALL LCD_INIT ; initialize LCD
8  MOV DPTR,#INITIAL_MSG_1 ;DPTR point to initial text
9  ACALL SEND_DAT ;DISPLAY DPTR content on LCD
10 ACALL DELAY2 ;GIVE DELAY
11 ACALL CLRSCR ; clear our screen
12 ;---------------------------------
```

- **Main Loop:** The main loop initializes the LCD, displays a message, reads a keypad input, and checks the password.

```
12 ;---------------------------------
13 MAIN:
14 CLR P2.7 ; port 2.7 will be used for buzzer
15 CLR P2.4 ; led lockdown
16 CLR P2.3 ; led Pass
17 ACALL LCD_INIT ; initialize LCD
18 MOV DPTR,#INITIAL_MSG_2 ;DPTR point to initial text
19 ACALL SEND_DAT ;DISPLAY DPTR content on LCD
20 ACALL DELAY ;GIVE DELAY
21 ACALL LINE2 ;MOVE TO LINE 2
```

```
ACALL READ_KEYPRESS ;take input from keypad
ACALL DELAY ;give some delay
ACALL CLRSCR ; clear our screen
MOV DPTR, #CHECK_PASS_MSG ;send checking code.. msg to lcd
ACALL SEND_DAT
ACALL DELAY2
ACALL CHECK_PASSWORD  ;CHECK for correct password
SJMP MAIN ;short jump to main
;--------------------------------
```

- **LCD Initialization (LCD_INIT):** Initializes the LCD by sending a series of commands stored in MYDATA:

```
;--------------------------------
LCD_INIT:MOV DPTR,#MYDATA
C1:CLR A
MOVC A,@A+DPTR
JZ DAT ; jump if A  = 0
ACALL COMNWRT
ACALL  DELAY
INC DPTR
SJMP C1
DAT:RET
;--------------------------------
```

- **Sending Data to LCD (SEND_DAT):** Sends data from the memory location pointed to by DPTR to the LCD:
  1-Define string using DB command at some address
  2-Add 0 at the end
  3-Move data pointer to that address
  4-Take first byte of String and move to accumulator A,
  5-Display this data of A to LCD
  6-Increment DPTR up to we get 0

```
;--------------------------------
SEND_DAT:
CLR A
MOVC A,@A+DPTR
JZ AGAIN ; jump if A = 0
ACALL DATAWRT
ACALL DELAY
INC DPTR
SJMP SEND_DAT
AGAIN: RET
;--------------------------------
```

- **Reading Keypress (READ_KEYPRESS):** Reads 5 keypresses from the keypad and stores them in memory starting at address 160:
  1-Make all the rows and columns high/VCC (1)
  2-Make the first starting row = ground/low (0)
  3-If the user presses any key from starting row, then that column will become grounded indicates that key is pressed
  4-Check for all the columns if it is grounded then load the ASCII value of the key to accumulator and stop the process

5-If none of the column is grounded, then make this row 1/High voltage and repeat the same process for the other rows.
6-For 5 input we have to check continuously until we get 5 inputs

```
;--------------------------------
READ_KEYPRESS:
MOV R0, #5       ; R0 = 5 (Number of keypresses to read)
MOV R1, #160     ; R1 = 160 (Address to store keypresses in memory)
ROTATE:ACALL KEY_SCAN  ; Take the input key
MOV R7, A        ; Store key in R7 for processing
MOV A, R7        ; Move key to A for comparison
CJNE A, #23H, STORE_KEY ; Compare with '#' (23H) and branch if not equal
SJMP SKIP_DISPLAY ; If the key is '#', skip displaying it
STORE_KEY:
MOV A, R7        ; Load the key again from R7
MOV @R1, A       ; Store key at the address in R1
ACALL DATAWRT    ; Display the key on the LCD
ACALL DELAY2     ; Delay
ACALL DELAY2     ; Another delay to ensure smooth input
SKIP_DISPLAY:
MOV A, R7        ; Load the key again from R7
MOV @R1, A       ; Store key at the address in R1
INC R1           ; Move to the next memory location
DJNZ R0, ROTATE ; Repeat for 5 keypresses
RET
;--------------------------------
```

- **Checking Password (CHECK_PASSWORD):** Compares the entered password with the stored password:

```
;-----------------------------------
CHECK_PASSWORD:MOV R0,#5D  ;R0 = 5
MOV R1,#160D ; R1= 160
MOV DPTR,#PASSWORD ;DPTR Point to actual PASSWORD
RPT:CLR A ; A = 0
MOVC A,@A+DPTR ; A = FIRST NUMBER OF THE ACTUAL PASSWORD
XRL A,@R1 ; XOR with the actual password
;if both the numbers are equal then A = 0;
JNZ FAIL ; jump if a not = 0
INC R1
INC DPTR
DJNZ R0,RPT ;repeat this process for 5 times
ACALL SUCCESS
RET
;-----------------------------------
```

- **Success Routine (SUCCESS):** Handles the case when the entered password is correct:

```
;-----------------------------------
SUCCESS: ACALL CLRSCR
SETB p2.3
ACALL DELAY2
MOV DPTR,#TEXT_S1
ACALL SEND_DAT ;display correct password
ACALL DELAY2
ACALL CLRSCR ; clear our screen
MOV DPTR,#TEXT_S2
ACALL SEND_DAT ;display opening door
```

```
 95  ACALL DELAY2
 96  CLR P2.3
 97  ACALL DELAY3 ; GIVE SECOND DELAY
 98  ACALL CLRSCR
 99  MOV DPTR, #TEXT_S3
100  ACALL SEND_DAT
101  ACALL DELAY2
102  ACALL DELAY3; GIVE SECOND DELAY
103  ACALL Start_up_MSG
104  MOV R5,#3D ;reset attempts value
105  RET
106  ;----------------------------
```

- **Failure Routine (FAIL):** Handles the case when the entered password is incorrect:

```
106  ;----------------------------
107  FAIL:ACALL CLRSCR
108  SETB p2.4
109  SETB p2.7
110  ACALL DELAY2
111  MOV DPTR,#TEXT_F1
112  ACALL SEND_DAT ;display incorrect text
113  ACALL DELAY2
114  ACALL LINE2
115  MOV DPTR, #TEXT_F2
116  ACALL SEND_DAT ;display access denied text
117  ACALL DELAY2
118  CLR p2.4
119  CLR p2.7
120  DJNZ R5,LOOP
121  ACALL ALERT
122  LOOP: ACALL ATTEMPT
123  LJMP MAIN ;go to main funtion
124  ;-----------------------------
```

- **Attempt Counter (ATTEMPT):** Displays the number of attempts left:

```
124  ;-----------------------------
125  ATTEMPT: ACALL CLRSCR
126  MOV DPTR,#ATTEMPT_TEXT ;number of attempts left
127  ACALL SEND_DAT
128  ACALL DELAY2
129  MOV A,#48D ; 48 = 0
130  ADD A,R5
131  DA A
132  ACALL DATAWRT
133  ACALL DELAY
134  ACALL DELAY2
135  ACALL DELAY2
136  RET
137  ;-----------------------------
```

- **Alert Routine (ALERT):** Displays an alert when attempts run out:

```
;-------------------------------
ALERT:
ACALL CLRSCR
SETB P2.7
MOV DPTR, #ALERT_TEXT ;display alert text
ACALL SEND_DAT
ACALL DELAY3
CLR P2.7
MOV R5,#3D
ACALL Start_up_MSG
LJMP MAIN
;-------------------------------------------------
```

- **Key Scan (KEY_SCAN):** Scans the keypad for keypresses:

```
;-------------------------------------------------
KEY_SCAN:MOV P1,#11111111B  ;TAKE INPUT FROM PORT 1
;CHECKING FOR ROW 1 COLUMN 1
CLR P1.0  ;first row checking #11111110
JB P1.4, NEXT1 ;when 1 column is 1 then no button is pressed , check for next column
MOV A,#55D ; if above fails then 7 is pressed , A =7
RET
NEXT1:JB P1.5,NEXT2 ; ROW 1 COULMN 2
MOV A,#56D ; A = 8
RET
NEXT2: JB P1.6,NEXT3 ; ROW 1 COLUMN 3
MOV A,#57D ; A=9
RET
NEXT3:SETB P1.0 ; ROW 1 IS RESET
CLR P1.1 ;CHECK FOR ROW 2
JB P1.4, NEXT4 ; ROW 2 COLUMN 1
MOV A,#52D ; A = 4
RET
NEXT4:JB P1.5,NEXT5 ; ROW 2 COLUMN 2
MOV A,#53D  ;A = 5
RET
NEXT5: JB P1.6,NEXT6 ; ROW 2 COLUMN 3
MOV A,#54D ;A = 6
RET
NEXT6:SETB P1.1 ;ROW IS RESET
CLR P1.2 ; CHECK FOR ROW 3
JB P1.4, NEXT7 ; ROW 3 COLUMN 1
MOV A,#49D  ;A = 1
RET
NEXT7:JB P1.5,NEXT8 ; ROW 3 COLUMN 2
MOV A,#50D ;A =2
RET
NEXT8: JB P1.6,NEXT9 ; ROW 3 COLUMN 3
MOV A,#51D ;A = 3
RET
NEXT9:SETB P1.2 ; ROW 3 IS RESET
CLR P1.3 ; CHECK FOR ROW 4
JB P1.4, NEXT10 ; ROW 4 COLUMN 1
MOV A,#42D ; A = *
RET
NEXT10:JB P1.5,NEXT11; ROW 4 COLUMN 2
MOV A,#48D ; A = 0
RET
```

```
NEXT11: JB P1.6,NEXT12 ; ROW 4 COLUMN 3
MOV A,#35D   ; A = #
RET
NEXT12:LJMP KEY_SCAN ; again check for keys
;-----------------------------------------------
```

- **Command Write to LCD (COMNWRT):** Sends a command to the LCD:
  1-Copy the command value to port 3
  2-Make register select = 0
  3-Select write mode by making R/W = 0
  4-Give a high to low Pulse to enable (port value high, delay then low)

```
;-----------------------------------------------
COMNWRT:MOV P3,A  ;to send command
CLR P2.0 ; R/s = 0
CLR P2.1 ;R/w =0
SETB P2.2 ;high
ACALL DELAY ; delay
CLR P2.2 ;low
RET
```

- **Data Write to LCD (DATAWRT):** Sends data to the LCD:
  1-Copy the ASCII value of data to port 3
  2-Make register select = 1
  3-Select write mode by making R/W = 0
  4-Give a high to low Pulse to enable (port value high, delay then low)

```
DATAWRT: MOV P3,A  ;to send data
SETB P2.0
CLR P2.1
SETB P2.2
ACALL DELAY
CLR P2.2
RET
;-----------------------------------------------
```

- **Move to Line 2 of LCD (LINE2):** Moves the cursor to the second line of the LCD.

```
;-----------------------------------------------
LINE2: MOV A,#0C0H ;move to line 2 of LCD
ACALL COMNWRT
RET
;---------------------------------
```

- **Delay Routines:**

1. **DELAY**: Generates a delay of 0.036 seconds.

```
DELAY: MOV R3,#65
HERE2: MOV R4,#255
HERE: DJNZ R4,HERE
DJNZ R3,HERE2
RET
;time delay = 0.036 S
;----------------------------------------
```

2. **DELAY2**: Generates a delay of 0.271 seconds.

```
;-----------------------------------------
DELAY2: MOV R3,#250D ; R3  = 250
        MOV TMOD,#01 ; timer 0 mode 1
BACK2:  MOV TH0,#0FCH
        MOV TL0,#018H  ;initial count value = FC18 is loaded into timer
        SETB TR0 ;starting timer
HERE5:  JNB TF0,HERE5 ;monitor Timer flag if it is 1
        CLR TR0 ; stop the timer
        CLR TF0 ; reset the timer flag
        DJNZ R3,BACK2 ; repeat this process 250 times
        RET
;time delay = 0.271 S
;-----------------------------------------
```

3. **DELAY3**: Generates a delay of 5 seconds.

```
;-----------------------------------------
DELAY3:MOV TMOD,#10H ;Timer 1, mod 1
MOV R3,#70 ; for multiple delay
AGAIN1: MOV TL1,#00H ;TL1=08,low byte of timer
MOV TH1,#00H ;TH1=01,high byte , TIMER = 0000
SETB TR1 ;Start timer 1
BACK: JNB TF1,BACK ;until timer rolls over
CLR TR1 ;Stop the timer 1
CLR TF1 ;clear Timer 1 flag
DJNZ R3,AGAIN1 ;if R3 not zero then
RET
;time delay = 4977.45mS = 5S
```

- **Data and Message Storage:** Sets the origin to 500H for data storage. The DB directives store messages and the password in memory. **DB**: Defines data bytes for various messages and initialization data for the LCD.

```
;-----------------------------------------
CLRSCR: MOV A,#01H
ACALL COMNWRT
RET
;-----------------------------------------
ORG 500H
DB 10000000B,01000000B,11000000B,00100000B,10100000B,01100000B,11100000B,00010000B,00110000B
MYDATA: DB 38H,0EH,01,06,80H,0;

INITIAL_MSG_1:    DB "Welcome Home -_-",0
INITIAL_MSG_2:    DB "Enter Password: ",0
CHECK_PASS_MSG:   DB "CHECKING PASS...",0
PASSWORD:DB 49D,50D,51D,52D,35D,0   ;PASSWORD = 1 2 3 4 #
TEXT_F1: DB "WRONG PASS",0
TEXT_F2: DB "ACCESS DENIED",0
TEXT_S1: DB "ACCESS GRANTED",0
TEXT_S2: DB "OPENING DOOR",0
TEXT_S3: DB "CLOSING DOOR", 0
ALERT_TEXT: DB "INTRUDER ALERT !",0
ATTEMPT_TEXT: DB "ATTEMPTS LEFT:",0
LOCKDOWN_TEXT: DB "LOCKDOWN STARTED",0
END
```

7. Testing and Results:

- **Testing Procedure**
  - Enter the correct password and verify the door unlocks for 5 seconds.
  - Enter an incorrect password and verify the LCD displays "Incorrect Password".
  - Enter three consecutive incorrect passwords and verify the buzzer sounds and the LCD displays "Alarm Triggered".
- **Results**

The system successfully locks and unlocks the door based on the entered password. The LCD provides appropriate feedback, and the buzzer sounds after three incorrect attempts, indicating potential intrusion.