

Nome:	Paulo Vinicius Araujo Feitosa	R.A.:	24.122.042-5
Nome:	Guilherme Marcato Mendes Justiça	R.A.:	24.122.045-8

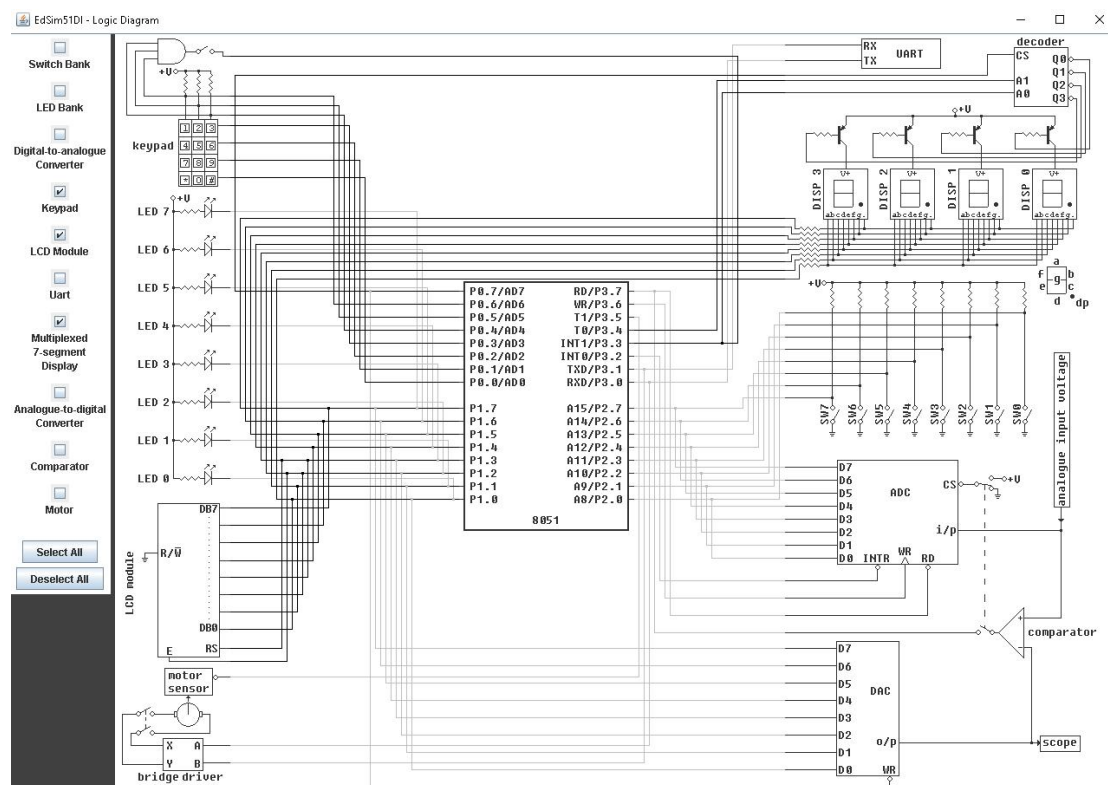
Projeto de Arquitetura de Computadores

• Descrição do Projeto

O projeto "Segurança do Brasil" visa criar um sistema de controle de acesso baseado no micro-controladores utilizando o ambiente de simulação EdSim51. O objetivo principal do projeto é desenvolver um painel de senha que permite aos usuários inserirem uma senha por um Keypad 4x3 e, caso a senha seja correta, o usuário pode fechar a porta ou mudar a senha.

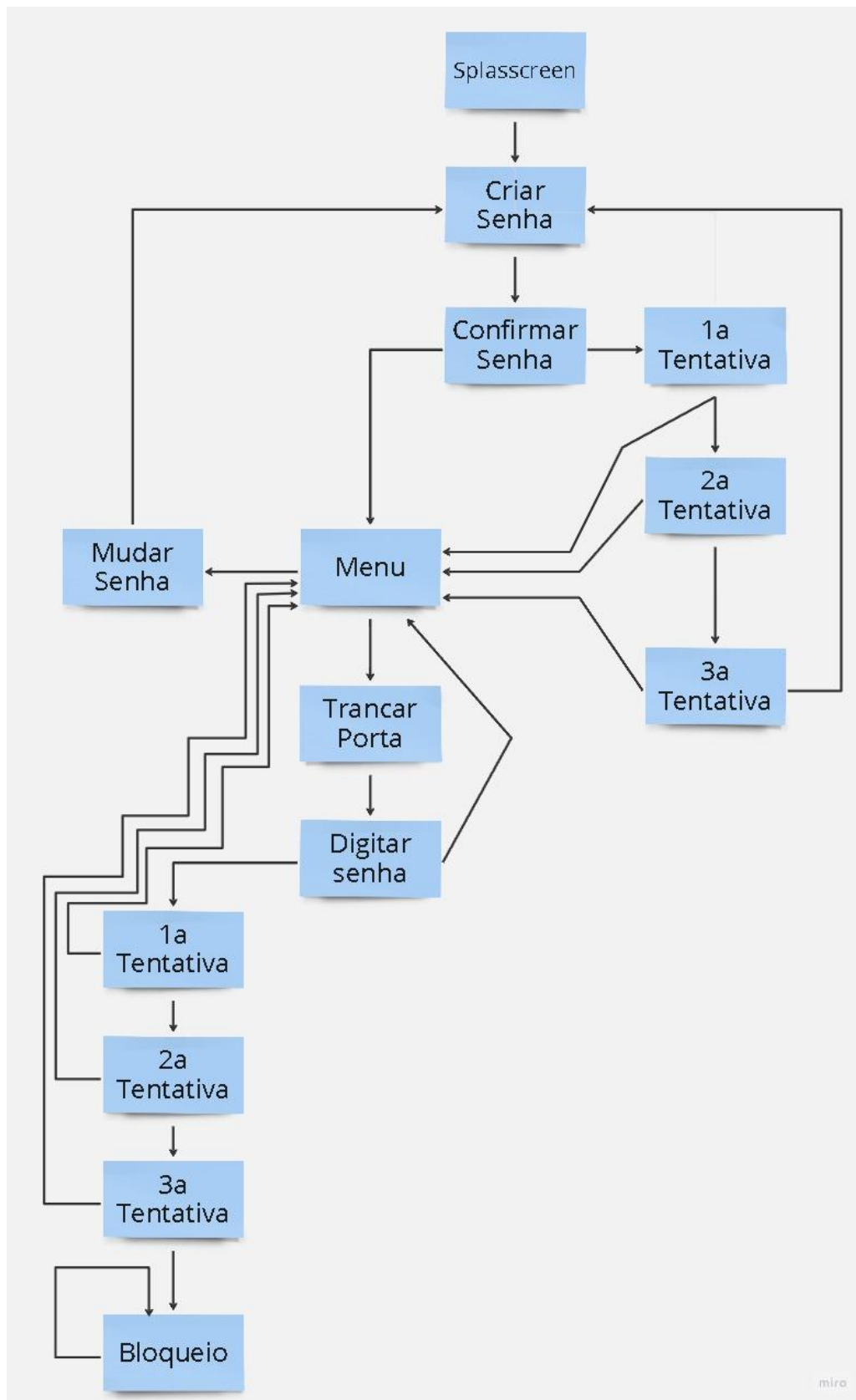
• Desenhos esquemáticos

O desenho esquemático do edsim51, mostrando as partes que estão sendo utilizadas.



Está selecionado o LCD Module, o Keypad e Multiplexed 7-segment Display .

•Fluxograma ou Diagrama



- Imagens da simulação realizada na IDE

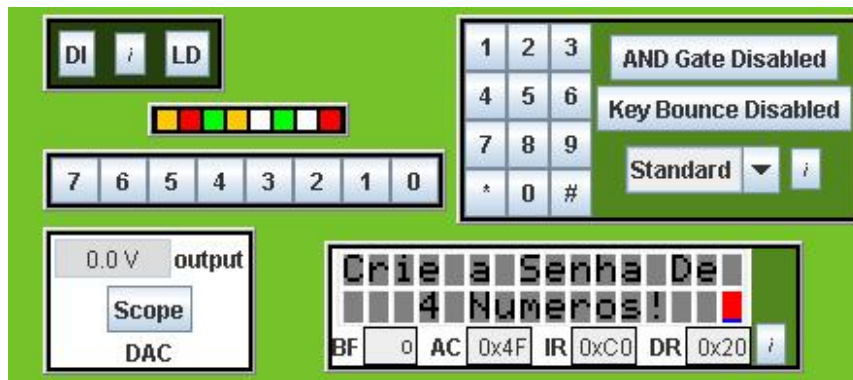


Figura 1 - Display mostrando ao usuario para criar a senha

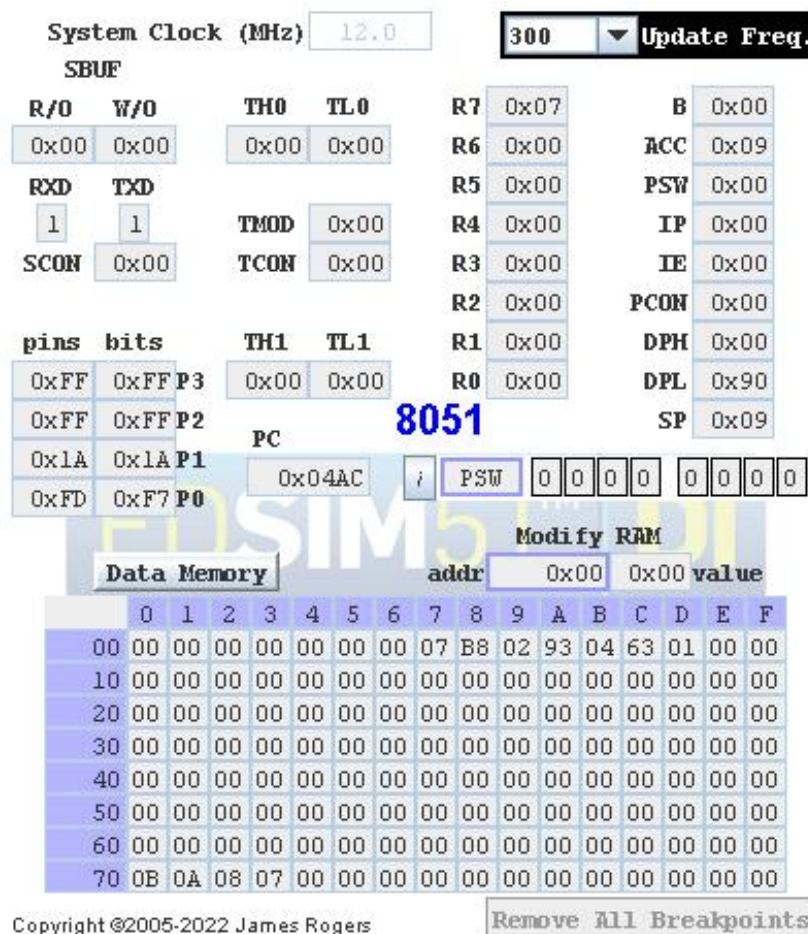


Figura 2 - Memória Interna com a senha salva

- **Imagens da simulação realizada na IDE**

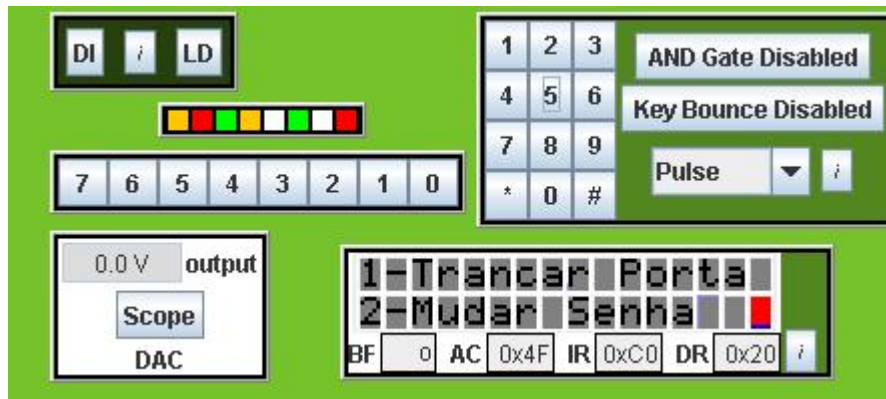


Figura 3 - Menu do programa

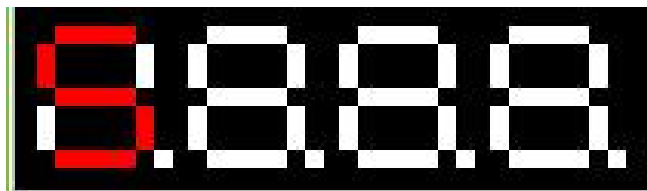


Figura 4 - Display 7seg com 5 da senha selecionado

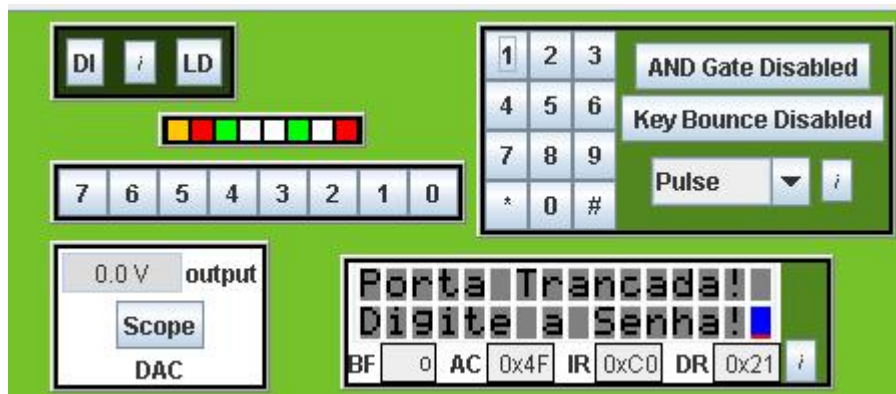


Figura 5 - Display mostrando que a porta está trancada

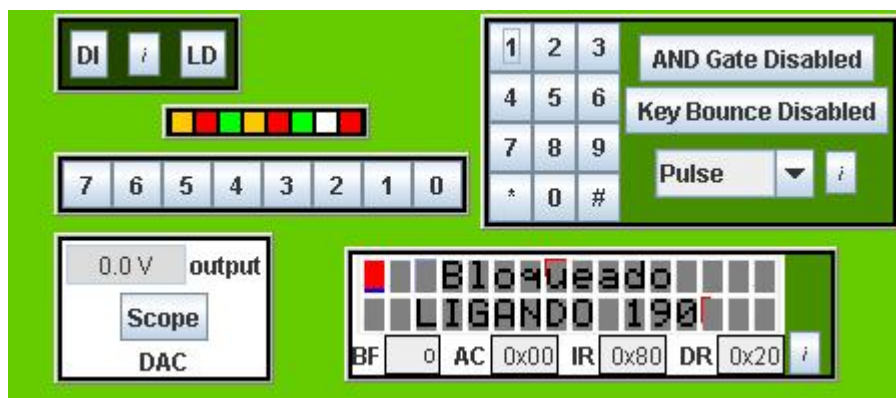


Figura 6 - Porta bloqueada

• **Discussões e conclusões**

Podemos deixar o microprocessador 8051 para os nossos ancestrais. O processo de desenvolvimento deste projeto foi bem desafiador, tivemos muita dificuldade com o estilo da IDE do edSim51 e com a própria linguagem do assembly. Por ter estas dificuldades no projeto, conseguimos aprender algumas funções com mais profundidade, além das pesquisas externas pela internet.

Sempre é bom aprender a usar as linguagens mais antigas ou de mais baixo nível, porém o uso destas linguagem para criar programa simples, como um sistema de criptografia, pode deixar o programador frustrado pela quantidade de linhas e comandos. A linguagem do assembly é uma linguagem em baixo nível, se a compararmos com Python ou Java, e o uso do assembly no simulador do 8051 (que é um microprocessador bem simples que deixou de ser usado no século passado) deixou ainda mais difícil pelas suas limitações de comandos. Se fizemos uma comparação com a linguagem C, podemos ver uma grande diferença de linhas e comandos que o programador precisa escrever para ter o mesmo sistema rodando em sua máquina.

O projeto foi bem concluído, o processo inicial foi bem estressante e arduo, porém após se acostumar com o edSim51 e com os comandos do assembly o projeto foi se desenvolvendo numa progressão mais rápida do que nós esperávamos.

•Código-fonte

```
Org      00H
;=====MAPEAMENTO DE
PINOS=====
    LCD      EQU P1
    E        EQU P1.2
    RS       EQU P1.3
;=====
=====
;=====CHAMA AS INSTRUÇÕES DE ACORDO COM O DATA
SHEET=====
    LCALL    FUNCTIONSET
    LCALL    DISPLAYon_off
    LCALL    ENTRYMODE
;=====
=====
;=====FUNÇÃO
PRINCIPAL=====

MAIN:
    MOV A,    #00h
    LCALL    posicionaCursor
    MOV DPTR, #INICIO
    LCALL    LCDW
```

```
    MOV A,    #40h
    LCALL    posicionaCursor
    MOV DPTR, #INICIO2
    LCALL    LCDW
    LJMP     CRIARSENHA
```

```
CRIARSENHA:
```

```

LCALL    CLEARDISPLAY
MOV  A,    #00h
LCALL    posicionaCursor
MOV  DPTR, #TEMSENHA
LCALL    LCDW

```

```

MOV  A,    #40h
LCALL    posicionaCursor
MOV  DPTR, #TEMSENHA2
LCALL    LCDW

```

```

LCALL    leituraTeclado
MOV  70H, R7
LCALL    leituraTeclado
MOV  71H, R7
LCALL    leituraTeclado
MOV  72H, R7
LCALL    leituraTeclado
MOV  73H, R7

```

```

LJMP     LOOP1TENTATIVA

```

```

SAVIO:
    LJMP SAVIO

```

```

;=====DATA=====
=====

```

```

INICIO:
    DB      'P','a','u','l','o','/','G','u','i','l','h','e','r','m',
'e',0
INICIO2:
    DB      'S','e','g','u','r','a','n','c','a',' ','D','o',' ',
',','B','R',0
TEMSENHA:
    DB      'C','r','i','e',' ','a',' ','S','e','n','h','a',' ',
',','D','e',0
TEMSENHA2:
    DB      ' ',' ',' ','4',' ','N','u','m','e','r','o','s','!',' ',
',',' ',0
CONFIRMARSENHA:
    DB      ' ','C','o','n','f','i','r','m','e',' ',
',','S','e','n','h','a',0
DIGITARSENHA2:

```

```

    DB      'D','i','g','i','t','e',' ','a','
','S','e','n','h','a','!',0
SENHAINCORRETA:
    DB      ' ','S','e','n','h','a',' ','E','r','r','a','d','a','
','!',0
SEGUNDATENTATIVA:
    DB      ' ','2','a',' ','T','e','n','t','a','t','i','v','a','
','!',0
TERCEIRATENTATIVA:
    DB      ' ','3','a',' ','T','e','n','t','a','t','i','v','a','
','!',0
PORTALIBERADA:
    DB      'P','o','r','t','a','
','L','i','b','e','r','a','d','a','!',0
TRANCARPORTA:
    DB      '1','-','T','r','a','n','c','a','r','
','P','o','r','t','a',0
MUDARSENHAESCRITO:
    DB      '2','-','M','u','d','a','r',' ','S','e','n','h','a','
',' ',0
MUDARSENHAESCRITO2:
    DB      ' ',' ','M','u','d','a','r',' ','S','e','n','h','a','
',' ',0
PORTATRANCADA:
    DB      'P','o','r','t','a','
','T','r','a','n','c','a','d','a','!',0
BLOQUEIOESCRITO:
    DB      ' ',' ',' ','B','l','o','q','u','e','a','d','o',' ','
',' ',0
POLICIA:
    DB      ' ',' ','L','I','G','A','N','D','O',' ','1','9','0','
',' ',0
CLEAN:
    DB      ' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ','
',' ',0

```

```

;=====SUBROTINAS=====
=====

```

```

;=====SUBROTINA QUE MANDA DO DPTR PARA O
LCD=====
LCDW:
    CLR     A
    MOVC    A,@A+DPTR

```



```

    INC     DPTR                ;A RECEBE O VALOR NO PRÓXIMO ENDEREÇO.
    JZ      finish             ;VERIFICA SE O VETOR ESTÁ NO FINAL.
;=====
=====
    MOV     LCD,A              ;MANDA SUPERIOR NIBBLE
    SETB    RS                 ;GARANTE QUE A MENSAGEM SEJA ENVIADA
    LCALL   NE                 ;BORDA DE DESCIDA NO ENABLE(NEGATIVE EDGE)
;=====
=====
    RR      A
    RR      A
    RR      A
    RR      A
    MOV     LCD,A              ;MANDA NIBBLE INFERIOR
;=====
=====
    SETB    RS                 ;GARANTE QUE A MENSAGEM SEJA ENVIADA
    LCALL   NE                 ;BORDA DE DESCIDA NO ENABLE(NEGATIVE EDGE)
    LCALL   delay
    LJMP    LCDW
    RET

```

```

finish:
    RET
;=====SUBROTINA DA FUNCTION SET DE ACORDO COM O DATA SHEET=====
FUNCTIONSET:
    CLR     RS                 ;MANDA INSTRUÇÃO(GARANTE QUE A
INFORMAÇÃO SEJA MANDADA, CASO PRECISE SER CHAMADA DEPOIS DE "SEND
DATA")
    MOV     LCD, #00100000b     ;db7 db6 db5 db4 E RS x x (PAGINA 42)
    LCALL   NE                 ;BORDA DE DESCIDA NO ENABLE(NEGATIVE
EDGE)
    LCALL   delay
    LCALL   NE
    MOV     LCD, #10000000b     ;N = DB7 = 1 -> DUAS LINHAS
                                ;F = DB6 -> FONTE
    LCALL   NE
    LCALL   delay
    RET
;=====ENTRYMODE=====
=====
ENTRYMODE:
    CLR     RS                 ;MANDA INSTRUÇÃO

```

```

MOV        LCD,#00000000B
LCALL     NE          ;BORDA DE DESCIDA NO ENABLE(NEGATIVE EDGE)
MOV        LCD,#01110000B
LCALL     NE          ;BORDA DE DESCIDA NO ENABLE(NEGATIVE EDGE)
LCALL     delay       ;ESPERA O BF ZERAR
RET

;=====DISPLAY ON/OFF
CONTROL=====
DISPLAYon_off:
    CLR        RS          ;MANDA INSTRUÇÃO
    MOV        LCD,#00000000B
    LCALL     NE
    MOV        LCD,#11110000B
    LCALL     NE
    LCALL     delay       ;ESPERA O BF ZERAR
    RET

;=====SEGUNDA LINHA=====
=====
SEGUNDA_LINHA:
    CLR        RS
    MOV        LCD,#11000000B
    LCALL     NE
    MOV        LCD,#00000000B
    LCALL     NE
    LCALL     DELAY
    RET

;=====CLEAR DISPLAY=====
=====
CLEAR DISPLAY:
    ;CLR        RS
    ;MOV        LCD,#00000000B
    ;LCALL     NE
    ;MOV        LCD,#00010000B
    ;LCALL     NE

```

```

MOV A,      #00h
LCALL      posicionaCursor

```

```

LCALL     DELAY
MOV DPTR, #CLEAN
LCALL     LCDW

```

```

MOV A,      #40h
LCALL      posicionaCursor

```

```

LCALL    DELAY
MOV DPTR, #CLEAN
LCALL    LCDW

```

```

RET
;=====SUBROTINA DE DELAY
50US=====
delay:
    MOV R0,    #50
    DJNZ R0,    $
    RET
;=====NEGATIVE EDGE NO
ENABLE=====
NE:
    SETB      E
    CLR       E
    RET

```

```

;----- Pulse -----
-----
Pulse:    SetB E      ; /*P1.2 is connected to 'E' pin of LCD module*
          Clr  E      ; / negative edge on E
          Ret
;----- SendChar -----
-----
SendChar: Mov C, ACC.7      ; /
          Mov P1.7, C      ; /
          Mov C, ACC.6      ; /
          Mov P1.6, C      ; /
          Mov C, ACC.5      ; /
          Mov P1.5, C      ; /
          Mov C, ACC.4      ; /
          Mov P1.4, C      ; / high nibble set
          ;LJMP $
          LCALL Pulse

```

```

          Mov C, ACC.3      ; /
          Mov P1.7, C      ; /
          Mov C, ACC.2      ; /
          Mov P1.6, C      ; /
          Mov C, ACC.1      ; /
          Mov P1.5, C      ; /
          Mov C, ACC.0      ; /

```

```
Mov P1.4, C      ; / low nibble set
```

```
LCALL Pulse
```

```
LCALL Delay      ; wait for BF to clear
```

```
Mov R1,#55h
```

```
Ret
```

```
;----- Scan Row -----  
-----
```

```
ScanKeyPad: CLR P0.3      ;Clear Row3  
            LCALL IDCode0 ;LCALL scan column subroutine  
            SetB P0.3     ;Set Row 3  
            JB F0,Done    ;If F0 is set, end scan
```

```
            ;Scan Row2  
            CLR P0.2      ;Clear Row2  
            LCALL IDCode1 ;LCALL scan column subroutine  
            SetB P0.2     ;Set Row 2  
            JB F0,Done    ;If F0 is set, end scan
```

```
            ;Scan Row1  
            CLR P0.1      ;Clear Row1  
            LCALL IDCode2 ;LCALL scan column subroutine  
            SetB P0.1     ;Set Row 1  
            JB F0,Done    ;If F0 is set, end scan
```

```
            ;Scan Row0  
            CLR P0.0      ;Clear Row0  
            LCALL IDCode3 ;LCALL scan column subroutine  
            SetB P0.0     ;Set Row 0  
            JB F0,Done    ;If F0 is set, end scan
```

```
LJMP ScanKeyPad ;Go back to scan Row3
```

```
Done:      Clr F0      ;Clear F0 flag before exit  
            Ret
```

```
;-----  
-----
```

```
;----- Scan column subroutine -----  
-----
```

```
IDCode0:   JNB P0.4, KeyCode03 ;If Col0 Row3 is cleared - key found  
            JNB P0.5, KeyCode13 ;If Col1 Row3 is cleared - key found  
            JNB P0.6, KeyCode23 ;If Col2 Row3 is cleared - key found
```

```
RET
```

```
KeyCode03: SETB F0          ;Key found - set F0  
           Mov R7,#'3'      ;Code for '3'  
           RET
```

```
KeyCode13: SETB F0          ;Key found - set F0  
           Mov R7,#'2'      ;Code for '2'  
           RET
```

```
KeyCode23: SETB F0          ;Key found - set F0  
           Mov R7,#'1'      ;Code for '1'  
           RET
```

```
IDCode1:   JNB P0.4, KeyCode02 ;If Col0 Row2 is cleared - key found  
           JNB P0.5, KeyCode12 ;If Col1 Row2 is cleared - key found  
           JNB P0.6, KeyCode22 ;If Col2 Row2 is cleared - key found  
           RET
```

```
KeyCode02: SETB F0          ;Key found - set F0  
           Mov R7,#'6'      ;Code for '6'
```

```
RET
```

```
KeyCode12: SETB F0          ;Key found - set F0  
           Mov R7,#'5'      ;Code for '5'  
           RET
```

```
KeyCode22: SETB F0          ;Key found - set F0  
           Mov R7,#'4'      ;Code for '4'  
           RET
```

```
IDCode2:   JNB P0.4, KeyCode01 ;If Col0 Row1 is cleared - key found  
           JNB P0.5, KeyCode11 ;If Col1 Row1 is cleared - key found  
           JNB P0.6, KeyCode21 ;If Col2 Row1 is cleared - key found  
           RET
```

```
KeyCode01: SETB F0          ;Key found - set F0  
           Mov R7,#'9'      ;Code for '9'  
           RET
```

```
KeyCode11: SETB F0          ;Key found - set F0  
           Mov R7,#'8'      ;Code for '8'  
           RET
```

```

KeyCode21:  SETB F0          ;Key found - set F0
            Mov R7,#'7'      ;Code for '7'
            RET

```

```

IDCode3:    JNB P0.4, KeyCode00 ;If Col0 Row0 is cleared - key found
            JNB P0.5, KeyCode10 ;If Col1 Row0 is cleared - key found
            JNB P0.6, KeyCode20 ;If Col2 Row0 is cleared - key found
            RET

```

```

KeyCode00:  SETB F0          ;Key found - set F0
            Mov R7,#'#'      ;Code for '#'
            RET

```

```

KeyCode10:  SETB F0          ;Key found - set F0
            Mov R7,#'0'      ;Code for '0'
            RET

```

```

KeyCode20:  SETB F0          ;Key found - set F0
            Mov R7,#'*'      ;Code for '*'
            RET

```

;=====POSICIONA 0

CURSOR=====

posicionaCursor :

CLR RS ; clear RS - indicates that instruction is being sent to module

```

SETB P1.7 ; /
MOV C, ACC.6 ; /
MOV P1.6, C ; /
MOV C, ACC.5 ; /
MOV P1.5, C ; /
MOV C, ACC.4 ; /
MOV P1.4, C ; / high nibble set
SETB E ; /
CLR E ; / negative edge on E
MOV C, ACC.3 ; /
MOV P1.7, C ; /
MOV C, ACC.2 ; /
MOV P1.6, C ; /
MOV C, ACC.1 ; /
MOV P1.5, C ; /
MOV C, ACC.0 ; /
MOV P1.4, C ; / low nibble set
SETB E ; /

```

```
CLR E ; | negative edge on E
LCALL delay ; wait for BF to clear
RET
```

```
;===== 3 TENTATIVAS DE ESCREVER A SENHA
INICIAL =====
```

LOOP1TENTATIVA:

```
LCALL    CLEARDISPLAY
MOV A,    #00h
LCALL    posicionaCursor
MOV DPTR, #CONFIRMARSENHA
LCALL    LCDW
```

```
LCALL    leituraTeclado
MOV A, R7
MOV R3,A
LCALL    leituraTeclado
MOV A, R7
MOV R4,A
LCALL    leituraTeclado
MOV A, R7
MOV R5,A
LCALL    leituraTeclado
MOV A, R7
MOV R6,A
```

```
MOV A,R3
CJNE A,70H,LOOP2TENTATIVA
MOV A,R4
CJNE A,71H,LOOP2TENTATIVA
MOV A,R5
CJNE A,72H,LOOP2TENTATIVA
MOV A,R6
CJNE A,73H,LOOP2TENTATIVA
LJMP LIBEROU
```

LOOP2TENTATIVA:

```
LCALL    CLEARDISPLAY
MOV A,    #00h
LCALL    posicionaCursor
```

```
MOV DPTR, #SENHAINCORRETA
LCALL LCDW
```

```
MOV A, #40h
LCALL posicionaCursor
MOV DPTR, #SEGUNDATENTATIVA
LCALL LCDW
```

```
LCALL leituraTeclado
MOV A, R7
MOV R3,A
LCALL leituraTeclado
MOV A, R7
MOV R4,A
LCALL leituraTeclado
MOV A, R7
MOV R5,A
LCALL leituraTeclado
MOV A, R7
MOV R6,A
```

```
MOV A,R3
CJNE A,70H,LOOP3TENTATIVA
MOV A,R4
CJNE A,71H,LOOP3TENTATIVA
MOV A,R5
CJNE A,72H,LOOP3TENTATIVA
MOV A,R6
CJNE A,73H,LOOP3TENTATIVA
LJMP LIBEROU
```

LOOP3TENTATIVA:

```
LCALL CLEARDISPLAY
MOV A, #00h
LCALL posicionaCursor
MOV DPTR, #SENHAINCORRETA
LCALL LCDW
```

```
MOV A, #40h
LCALL posicionaCursor
MOV DPTR, #TERCEIRATENTATIVA
LCALL LCDW
```



```

LCALL    leituraTeclado
MOV A, R7
MOV R3,A
LCALL    leituraTeclado
MOV A, R7
MOV R4,A
LCALL    leituraTeclado
MOV A, R7
MOV R5,A
LCALL    leituraTeclado
MOV A, R7
MOV R6,A

```

```

MOV A,R3
CJNE A,70H,REFACA
MOV A,R4
CJNE A,71H,REFACA
MOV A,R5
CJNE A,72H,REFACA
MOV A,R6
CJNE A,73H,REFACA
LJMP LIBEROU

;===== Refazer a senha inicial
=====
REFACA:
    LCALL    CLEARDISPLAY
    MOV A,    #00h
    LCALL    posicionaCursor
    MOV DPTR, #TEMSENHA
    LCALL    LCDW

```

```

MOV A,    #40h
LCALL    posicionaCursor
MOV DPTR, #TEMSENHA2
LCALL    LCDW

```

```

LCALL    leituraTeclado
MOV 70H, R7
LCALL    leituraTeclado
MOV 71H, R7
LCALL    leituraTeclado
MOV 72H, R7
LCALL    leituraTeclado
MOV 73H, R7

```

```

LCALL    CLEARDISPLAY
MOV  A,   #00h
LCALL    posicionaCursor
MOV  DPTR, #CONFIRMARSENHA
LCALL    LCDW

```

```

    LJMP  LOOP1TENTATIVA
;=====IR MENU CASO SENHA ESTEJA
CERTA=====
LIBEROU:

```

```

LCALL    CLEARDISPLAY
MOV  A,   #00h
LCALL    posicionaCursor
MOV  DPTR, #PORTALIBERADA
LCALL    LCDW

```

```

LJMP     escolhaMenu

```

```

escolhaMenu:

```

```

LCALL    CLEARDISPLAY
MOV  A,   #00h
LCALL    posicionaCursor
MOV  DPTR, #TRANCARPORTA
LCALL    LCDW
MOV  A,   #40h
LCALL    posicionaCursor
MOV  DPTR, #MUDARSENHAESCRITO
LCALL    LCDW

```

```

LCALL leituraTeclado
CJNE R7, #0BH, escolhaMenu2
LJMP  TRANCOU

```

```

escolhaMenu2:

```

```

    CJNE R7, #0AH, escolhaMenu
    LJMP  mudarSenha

```

```

;=====TRANCAR A
PORTA=====
TRANCOU:

```

```

LCALL    CLEARDISPLAY
MOV  A,   #00h

```

```

    LCALL    posicionaCursor
    MOV DPTR, #PORTATRANCADA
    LCALL    LCDW
    MOV A,    #40h
    LCALL    posicionaCursor
    MOV DPTR, #DIGITARSENHA2
    LCALL    LCDW
    LJMP     DESTRANCARPORTA

;=====MUDAR A
SENHA=====
mudarSenha:
    LCALL    CLEARDISPLAY
    MOV A,    #00h
    LCALL    posicionaCursor
    MOV DPTR, #MUDARSENHAESCRITO2
    LCALL    LCDW
    LJMP     CRIARSENHA
    LJMP     LIBEROU

```

```

;=====DESTRANCAR A PORTA COM 3 TENTATIVAS JA
COM A SENHA DEFINIDA=====

```

```

DESTRANCARPORTA:
    LCALL    leituraTeclado
    MOV A, R7
    MOV R3,A
    LCALL    leituraTeclado
    MOV A, R7
    MOV R4,A
    LCALL    leituraTeclado
    MOV A, R7
    MOV R5,A
    LCALL    leituraTeclado
    MOV A, R7
    MOV R6,A

```

```

    MOV A,R3
    CJNE A,70H,DESTRANCARPORTA2
    MOV A,R4
    CJNE A,71H,DESTRANCARPORTA2
    MOV A,R5
    CJNE A,72H,DESTRANCARPORTA2
    MOV A,R6
    CJNE A,73H,DESTRANCARPORTA2

```

```
LJMP LIBEROU
```

DESTRANCARPORTA2:

```
LCALL    CLEARDISPLAY
MOV A,    #00h
LCALL    posicionaCursor
MOV DPTR, #SENHAINCORRETA
LCALL    LCDW
```

```
MOV A,    #40h
LCALL    posicionaCursor
MOV DPTR, #SEGUNDATENTATIVA
LCALL    LCDW
```

```
LCALL    leituraTeclado
MOV A, R7
MOV R3,A
LCALL    leituraTeclado
MOV A, R7
MOV R4,A
LCALL    leituraTeclado
MOV A, R7
MOV R5,A
LCALL    leituraTeclado
MOV A, R7
MOV R6,A
```

```
MOV A,R3
CJNE A,70H,DESTRANCARPORTA3
MOV A,R4
CJNE A,71H,DESTRANCARPORTA3
MOV A,R5
CJNE A,72H,DESTRANCARPORTA3
MOV A,R6
CJNE A,73H,DESTRANCARPORTA3
LJMP LIBEROU
```

DESTRANCARPORTA3:

```
LCALL    CLEARDISPLAY
MOV A,    #00h
LCALL    posicionaCursor
MOV DPTR, #SENHAINCORRETA
LCALL    LCDW
```

```

MOV A,      #40h
LCALL      posicionaCursor
MOV DPTR,   #TERCEIRATENTATIVA
LCALL      LCDW

```

```

LCALL      leituraTeclado
MOV A, R7
MOV R3,A
LCALL      leituraTeclado
MOV A, R7
MOV R4,A
LCALL      leituraTeclado
MOV A, R7
MOV R5,A
LCALL      leituraTeclado
MOV A, R7
MOV R6,A

```

```

MOV A,R3
CJNE A,70H,BLOQUEIO
MOV A,R4
CJNE A,71H,BLOQUEIO
MOV A,R5
CJNE A,72H,BLOQUEIO
MOV A,R6
CJNE A,73H,BLOQUEIO
LJMP LIBEROU

```

```

;=====BLOQUEAR A PORTA E CHAMAR A
POLICIA=====

```

BLOQUEIO:

```

LCALL      CLEARDISPLAY
MOV A,      #00h
LCALL      posicionaCursor
MOV DPTR,   #BLOQUEIOESCRITO
LCALL      LCDW
MOV A,      #40h
LCALL      posicionaCursor
MOV DPTR,   #POLICIA
LCALL      LCDW
LJMP BLOQUEIO

```

```
;=====LER O INPUT DO  
USUARIO=====
```

```
leituraTeclado:
```

```
CLR F0  
MOV A,#0  
MOV P0,#11111110b  
LCALL COLUMN_VERIFY  
MOV P0,#11111101b  
LCALL COLUMN_VERIFY  
MOV P0,#11111011b  
LCALL COLUMN_VERIFY  
MOV P0,#11110111b  
LCALL COLUMN_VERIFY
```

```
JNB F0,leituraTeclado  
RET
```

```
REST_LOOP:
```

```
MOV DPTR,#VETOR_DISPLAY  
MOV R7, A  
MOVC A,@A+DPTR  
MOV P1,A
```

```
JNB P0.4, $  
JNB P0.5, $  
JNB P0.6, $
```

```
SETB F0
```

```
COLUMN_VERIFY:
```

```
JNB P0.4, REST_LOOP  
INC A  
JNB P0.5, REST_LOOP  
INC A  
JNB P0.6, REST_LOOP  
INC A  
RET
```

```
VETOR_DISPLAY:
```

```
DB 11111111B ;|PosiÃ§Ã£o 0: Nada |
```

```
DB 11000000B ;/Posição 1: Zero |
DB 11111111B ;/Posição 2: Nada |
DB 10010000B ;/Posição 3: Nove |
DB 10000000B ;/Posição 4: Oito |
DB 11111000B ;/Posição 5: Sete |
DB 10000010B ;/Posição 6: Seis |
DB 10010010B ;/Posição 7: Cinco |
DB 10011001B ;/Posição 8: Quatro|
DB 10110000B ;/Posição 9: Três |
DB 10100100B ;/Posição 10: Dois |
DB 11111001B ;/Posição 11: Um |
RET
```

```
gotKey:
    SETB F0
    RET
END
```

