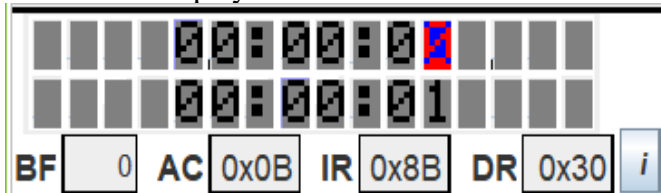


|                                   |                    |
|-----------------------------------|--------------------|
| Nome: Ana Jéssica Soares da Silva | R.A.: 22.121.100-6 |
| Nome: João Victor Passos de Moura | R.A.: 22.121.117-0 |

# Projeto de Arquitetura de Computadores

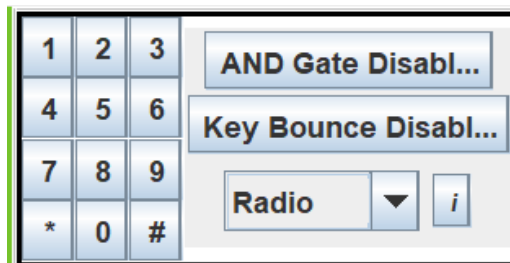
## 1. Descrição do Projeto

O projeto tem como objetivo realizar uma simulação de um Relógio digital com despertador, onde é utilizado o EdSim51 e o Display LCD:



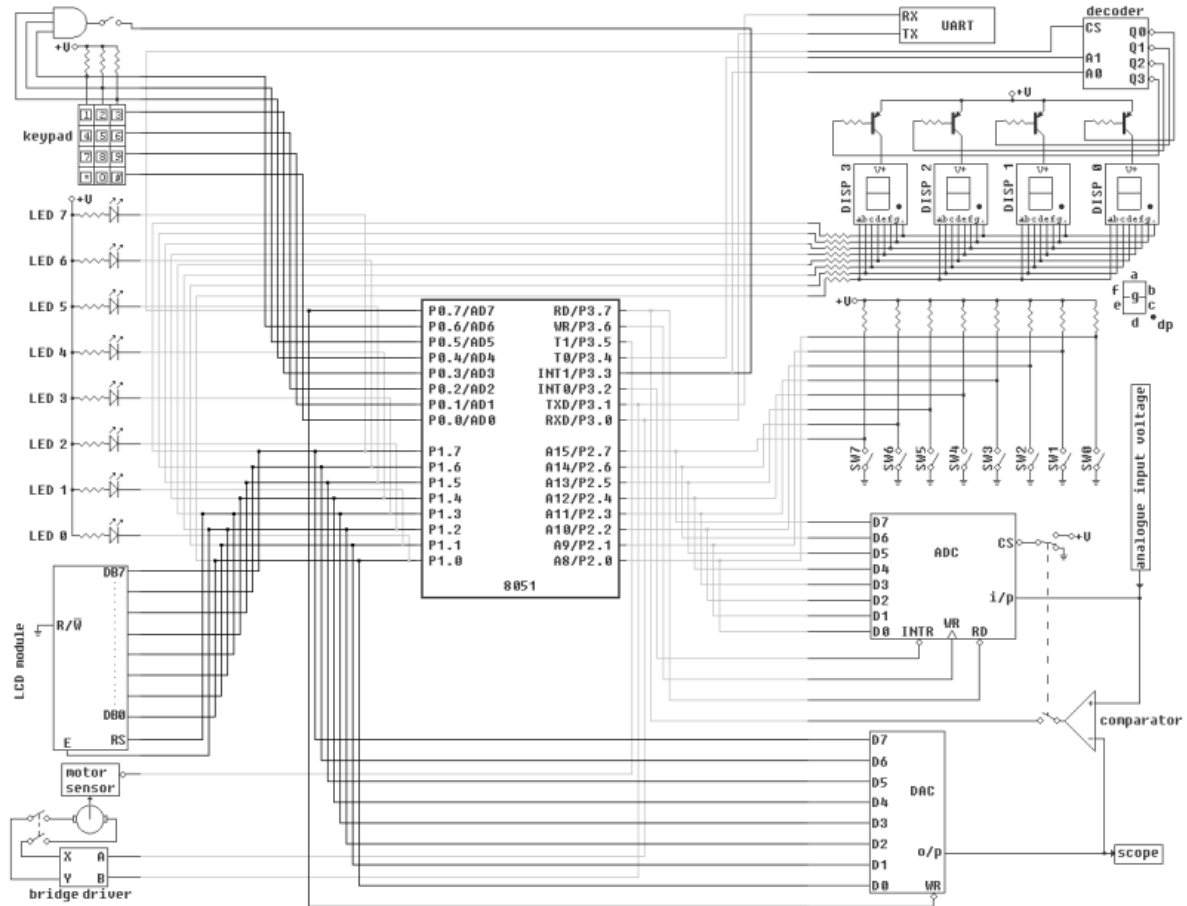
No modo RUN, o relógio roda normalmente, e a parte de cima, fica zerado até o usuário incrementar e selecionar o horário para acionar o despertador, que quando acionado, mostrará uma mensagem de alerta “ACORDA” e após a mensagem, ele espera o usuário selecionar outro horário para despertar.

Como funciona o despertador:



Quando o usuário clicar nos números do teclado, a contagem de valores ira iniciar, sendo ela feita até o numero 6 (00:00:00), podendo inserir qualquer valor, e quando o relógio chegar no valor inserido, ele passa a mensagem, e depois espera outro comando.

## 2. Desenhos esquemáticos



### 3. Fluxograma ou Diagrama

System Clock (MHz)

**SBUF**

|      |      |      |      |    |      |      |      |
|------|------|------|------|----|------|------|------|
| R/O  | W/O  | TH0  | TL0  | R7 | 0x16 | B    | 0x00 |
| 0x00 | 0x00 | 0x00 | 0x00 | R6 | 0x00 | ACC  | 0x37 |
| RXD  | TXD  | TMOD | 0x00 | R5 | 0x00 | PSW  | 0x81 |
| 1    | 1    | TCON | 0x00 | R4 | 0x00 | IP   | 0x00 |
| SCON | 0x00 |      |      | R3 | 0x00 | IE   | 0x00 |
|      |      |      |      | R2 | 0x00 | PCON | 0x00 |
| pins | bits | TH1  | TL1  | R1 | 0x00 | DPH  | 0x00 |
| 0xFF | 0xFF | P3   | 0x00 | R0 | 0x4C | DPL  | 0x00 |
| 0xFF | 0xFF | P2   |      |    |      | SP   | 0x0D |
| 0x7B | 0x7B | P1   |      |    |      |      |      |
| 0xF7 | 0xF7 | P0   |      |    |      |      |      |

**PC**  **8051**

**PSW**

**Modify RAM**

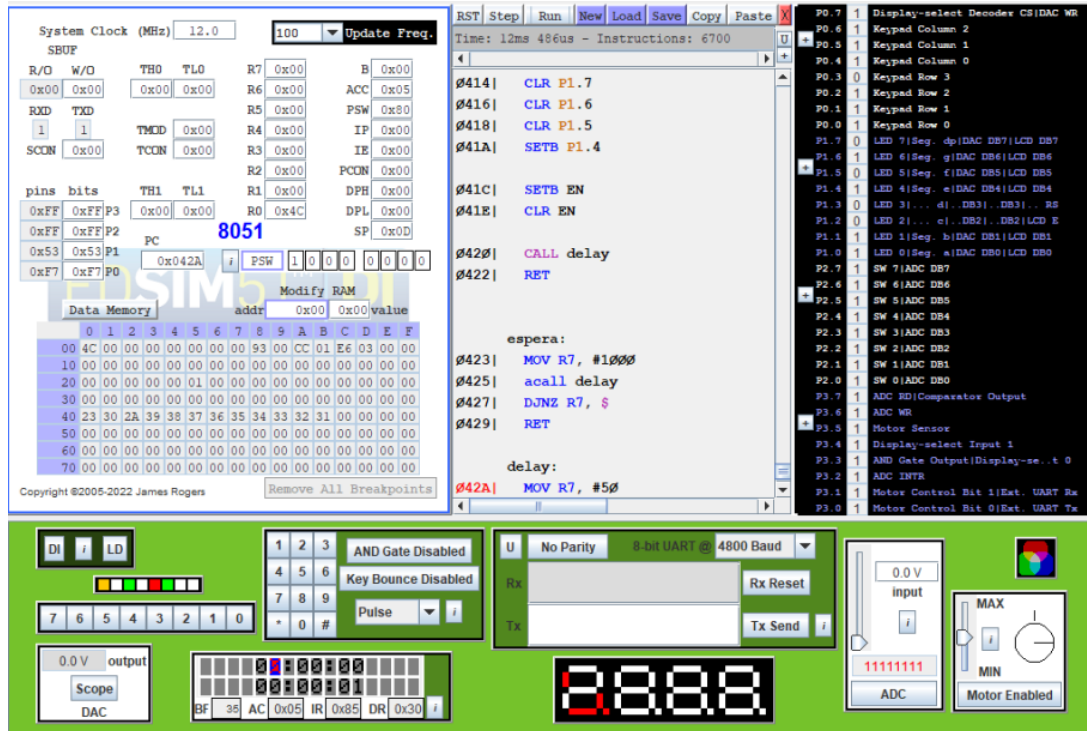
**Data Memory**

|    | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | A  | B  | C  | D  | E  | F  |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 00 | 4C | 00 | 00 | 00 | 00 | 00 | 00 | 16 | 00 | BE | 01 | B9 | 00 | 00 | 00 | 00 |
| 10 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 20 | 00 | 00 | 00 | 00 | 00 | 01 | 07 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 30 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 40 | 23 | 30 | 2A | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 | 31 | 00 | 00 | 00 | 00 |
| 50 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 60 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 70 | 03 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |

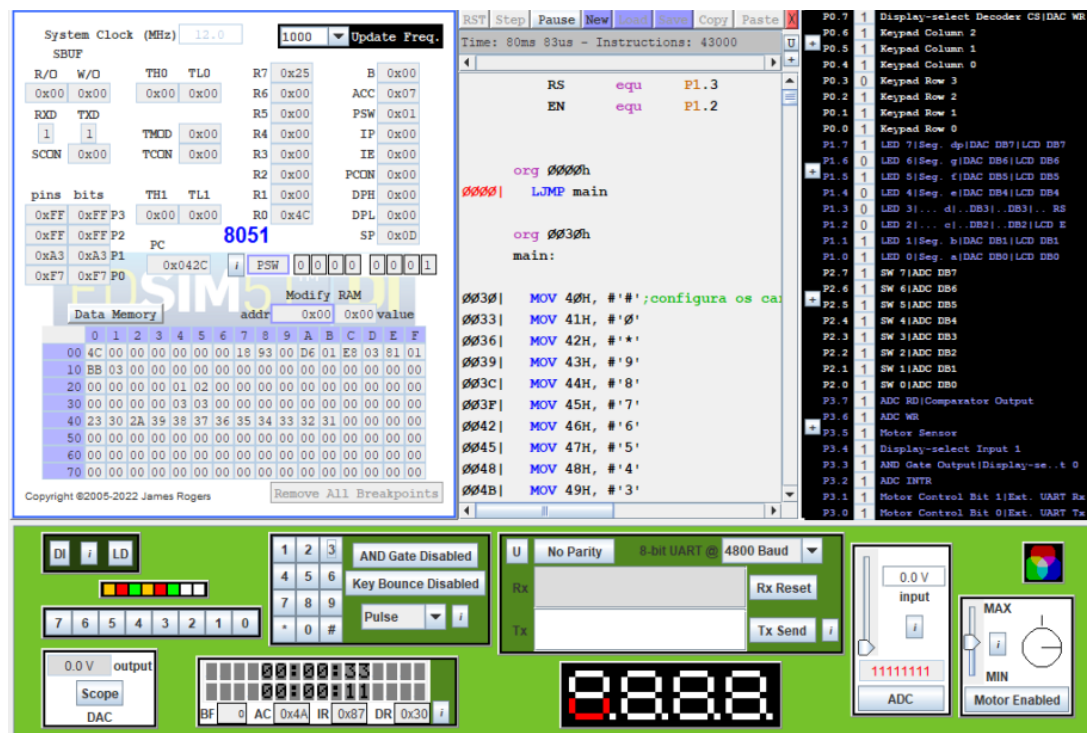
Copyright ©2005-2022 James Rogers

[ ] - Números do meu relógio  
[ ] - Temporizador  
[ ] - Contagem de números inseridos no meu despertador  
[ ] - Números do meu teclado  
[ ] - Armazenamento usado por funções mais simples do sistema

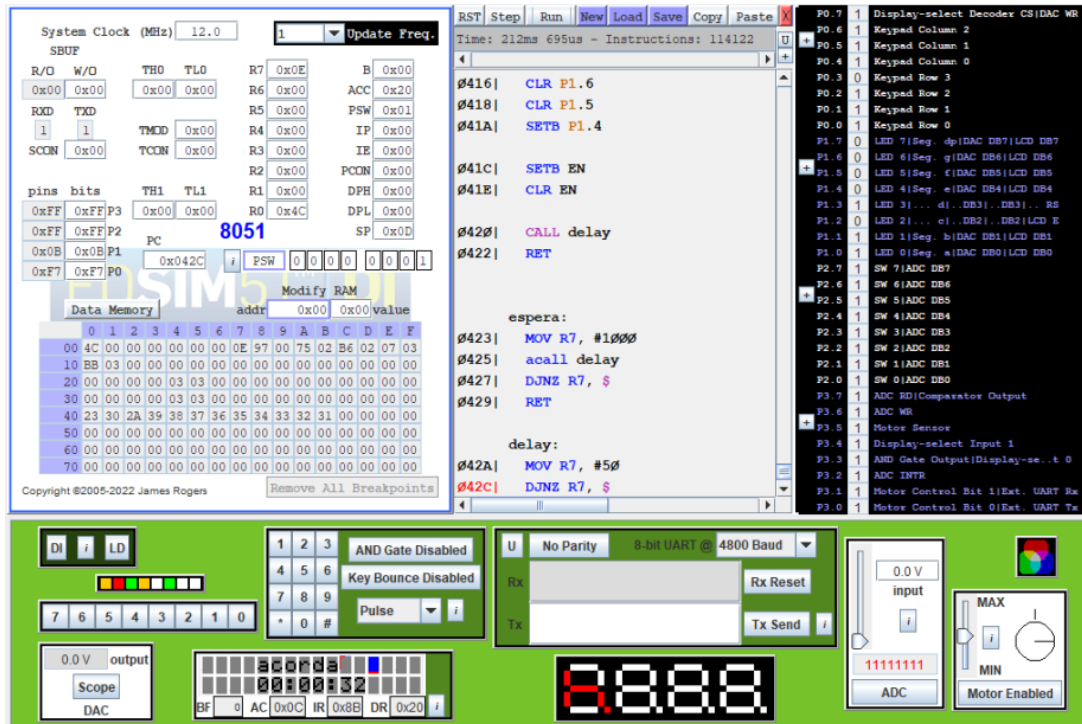
## 4. Imagens da simulação realizada na IDE



Exibição do relógio digital e do despertador, que será incrementado pelo usuário



Exibindo o horário proposto pelo usuário para acionar o despertador



Exibição da mensagem “ACORDA”, como método de aviso

## 5. Discussões e conclusões

Descreva sobre o processo de desenvolvimento do projeto, tais como:

- Descreva os desafios encontrados no projeto, as dificuldades.**

Durante o desenvolvimento do projeto, foi surgindo vários desafios e obstáculos, mas um que possa ser destacado como o que mais teve nossa atenção foi a incrementação dos valores utilizando a tabela ASCII, pois a logica desenvolvida foi utilizar inteiros e passar esses valores para hexadecimal, para fazer o relógio rodar.
- O que você aprendeu de novo com o projeto (extraclasse, que não foi ensinado em aula).**

Durante o projeto, pode se dizer que estudamos várias e várias formas de realizá-lo e entre essas logicas pensadas, a parte da criação da incrementação foi algo pesquisado por fora.
- Qual a sua visão em relação a programação em linguagem assembly (houve a necessidade de conhecer bem o hardware?) em comparação com outras linguagens.**

A linguagem assembly é uma versão legível da linguagem de máquina, tendo uma passagem quase sempre direta, não envolvendo muito processamento, acabando sendo mais simples, mas também tem benefícios, sendo eles você ter uma visão mais ampla de como as tarefas são executadas para as quais são programadas, entretanto existem limitações por se tratar de uma comunicação simples, como por exemplo, a utilização de um IF, pois na linguagem assembly essa utilização consome mais linhas de código e mais processos, do que utiliza-lo em uma linguagem mais avançada (Alto nível).

## 6. Código-fonte

```
RS      equ      P1.3
EN      equ      P1.2

org 0000h
    LJMP menu

org 0030h
menu:
;configura os caracteres em char para o teclado
    MOV 40H, #'#';23
    MOV 41H, #'0';30
    MOV 42H, #'*';2A
    MOV 43H, #'9';39
    MOV 44H, #'8';38
    MOV 45H, #'7';37
    MOV 46H, #'6';36
    MOV 47H, #'5';35
    MOV 48H, #'4';34
    MOV 49H, #'3';33
    MOV 4AH, #'2';32
    MOV 4BH, #'1';31

    mov 20h,#0H ;hora2
    mov 21h,#0H ;hora1
    mov 22h,#0H ;minuto2
    mov 23h,#0H ;minuto1
    mov 24h,#0H ;segundo2
    mov 25h,#0H ;segundo1


    mov 30h,#0H ;hora2 alarme
    mov 31h,#0H ;hora1 alarme
    mov 32h,#0H ;minuto2 alarme
    mov 33h,#0H ;minuto1 alarme
    mov 34h,#0H ;segundo2 alarme
    mov 35h,#0H ;segundo1 alarme

    mov 70h,#0h;contador de digitos do alarme

    ACALL lcd_init;incia o lcd
    ACALL estrutura;imrpime o separador de hora, minuto e segundo no LCD
ROTINA:
    ACALL leituraTeclado;le o teclado
```

```
MOV A, #01h  
ACALL posicionaC
```

```
MOV A, #40h  
ADD A, R0  
MOV R0, A  
MOV A, @R0  
mov b,a
```

```
acall digito;realiza a contagem de digitos  
acall posiciona; posisiona os digitos na memoria  
mov a,r0  
acall impresao;imprime todos os valores  
acall atualiza;atualiza o relógio  
acall alarme;verifica se  nessesario dispara o alarme
```

```
JMP ROTINA;refaz o loop
```

posiciona;; verifica se existe digitos para serem possionados na memoria

```
mov a,r0  
subb a,#4ch  
jnz lugar  
ret
```

lugar;; posisiona os digitos na memoria

```
mov a,70h  
mov psw,#00h  
subb a,#1h  
jz local0
```

```
mov a,70h  
mov psw,#00h  
subb a,#2h  
jz local1
```

```
mov a,70h  
mov psw,#00h  
subb a,#3h  
jz local2
```

```
mov a,70h  
mov psw,#00h  
subb a,#4h
```

```
jz local3
```

```
mov a,70h  
mov psw,#00h  
subb a,#5h  
jz local4
```

```
mov a,70h  
mov psw,#00h  
subb a,#6h  
jz local5
```

```
local0:;Posição 1  
    acall ajuste  
    mov 30h,r0  
    ret
```

```
local1:;Posição 2  
    acall ajuste  
    mov 31h,r0  
    ret
```

```
local2:;Posição 3  
    acall ajuste  
    mov 32h,r0  
    ret
```

```
local3:;Posição 4  
    acall ajuste  
    mov 33h,r0  
    ret
```

```
local4:;Posição 5
```



```
acall ajuste
mov 34h,r0
ret
```

local5;;Posição 6

```
acall ajuste
mov 35h,r0
mov 70h,#00h
ret
```

atualiza:

```
acall seg1
ret
```

seg1;;atualiza o 1 digito do segundo

```
inc 25h
mov a,25h
mov psw,#00h
subb a,#0ah
jz seg2
ret
```

seg2;;atualiza o 2 digito do segundo

```
mov 25h,#00h
inc 24h
mov a,24h
mov psw,#00h
subb a,#06h
jz min1
ret
```

min1;;atualiza o 1 digito do minuto

```
mov 24h,#00h
inc 23h
mov a,23h
mov psw,#00h
subb a,#0ah
jz min2
ret
```

min2;;atualiza o 2 digito do minuto

```
mov 23h,#00h
```

```
    inc 22h  
    mov a,22h  
    mov psw,#00h  
    subb a,#6h  
    jz hora1  
    ret
```

hora1;;atualiza o 1 digito da hora

```
    mov 22h,#00h  
    inc 21h  
    mov a,21h  
    mov psw,#00h  
    subb a,#0ah  
    jz hora2  
    mov a,21h  
    mov psw,#00h  
    subb a,#4h  
    jz dia1  
    ret
```

hora2;;atualiza o 2 digito da hora

```
    mov 21h,#00h  
    inc 20h  
    ret
```

dia1;;verifica se ja passou 1 dia

```
    mov a,20h  
    mov psw,#00h  
    subb a,#2h  
    jz dia2  
    ret
```

dia2;;reseta o relógio quando tiver decorrido 1 dia

```
    mov 20h,#00h  
    mov 21h,#00h  
    ret
```

estrutura;;imprime o separador de hora, minuto e segundo no LCD

```
    MOV A, #46h
```

```
ACALL posicionaC
```

```
MOV A, #3ah
```

```
ACALL enviaLetra
```

```
MOV A, #49h
```

```
ACALL posicionaC
```

```
MOV A, #3ah
```

```
ACALL enviaLetra
```

```
MOV A, #6h
```

```
ACALL posicionaC
```

```
MOV A, #3ah
```

```
ACALL enviaLetra
```

```
MOV A, #9h
```

```
ACALL posicionaC
```

```
MOV A, #3ah
```

```
ACALL enviaLetra
```

```
ret
```

```
impresao;;imprime os valores da memoria
```

```
MOV A, #44h
```

```
ACALL posicionaC
```

```
MOV A, 20H
```

```
ADD A,#30H
```

```
ACALL enviaLetra
```

```
MOV A, #45h
```

```
ACALL posicionaC
```

```
MOV A, 21H
```

```
ADD A,#30H
```

```
ACALL enviaLetra
```

```
MOV A, #47h  
ACALL posicionaC
```

```
MOV A, 22H  
ADD A,#30H  
ACALL enviaLetra
```

```
MOV A, #48h  
ACALL posicionaC
```

```
MOV A, 23H  
ADD A,#30H  
ACALL enviaLetra
```

```
MOV A, #4Ah  
ACALL posicionaC
```

```
MOV A, 24H  
ADD A,#30H  
ACALL enviaLetra
```

```
MOV A, #4Bh  
ACALL posicionaC
```

```
MOV A, 25H  
ADD A,#30H  
ACALL enviaLetra
```

```
MOV A, #4h  
ACALL posicionaC
```

```
MOV A, 30H  
ADD A,#30H  
ACALL enviaLetra
```

```
MOV A, #5h  
ACALL posicionaC
```

```
MOV A, 31H  
ADD A,#30H  
ACALL enviaLetra
```

```
MOV A, #7h  
ACALL posicionaC
```

```
MOV A, 32H  
ADD A,#30H  
ACALL enviaLetra
```

```
MOV A, #8h  
ACALL posicionaC
```

```
MOV A, 33H  
ADD A,#30H  
ACALL enviaLetra
```

```
MOV A, #0Ah  
ACALL posicionaC
```

```
MOV A, 34H  
ADD A,#30H  
ACALL enviaLetra
```

```
MOV A, #0Bh  
ACALL posicionaC
```

```
MOV A, 35H  
ADD A,#30H  
ACALL enviaLetra
```

```
RET
```

```
digito:;incrementa o digito  
mov a,R0
```

```
subb a,#4CH  
jnz incrementa  
ret
```

```
incrementa;;incrementa o digito  
inc 70h  
ret
```

```
ajuste;;converte os valores do teclado para hexadecimal
```

```
mov a,r0  
mov psw,#00h  
subb a,#4bH  
jz caso1
```

```
mov a,r0  
mov psw,#00h  
subb a,#4aH  
jz caso2
```

```
mov a,r0  
mov psw,#00h  
subb a,#49H  
jz caso3
```

```
mov a,r0  
mov psw,#00h  
subb a,#48H  
jz caso4
```

```
mov a,r0  
mov psw,#00h  
subb a,#47H  
jz caso5
```

```
mov a,r0  
mov psw,#00h  
subb a,#46H  
jz caso6
```

```
mov a,r0  
mov psw,#00h  
subb a,#45H  
jz caso7
```

```
mov a,r0
mov psw,#00h
subb a,#44H
jz caso8
```

```
mov a,r0
mov psw,#00h
subb a,#43H
jz caso9
```

```
mov a,r0
mov psw,#00h
subb a,#41H
jz caso0
```

```
ret
```

```
caso1;;caso 1 ajuste do teclado
mov r0,#1h
ret
```

```
caso2;;caso 2 ajuste do teclado
mov r0,#2h
ret
```

```
caso3;;caso 3 ajuste do teclado
mov r0,#3h
ret
```

```
caso4;;caso 4 ajuste do teclado

mov r0,#4h
ret
```

```
caso5;;caso 5 ajuste do teclado

mov r0,#5h
ret
```

```
caso6;;caso 6 ajuste do teclado
```

```
    mov r0,#6h
    ret
caso7::caso 7 ajuste do teclado

    mov r0,#7h
    ret
caso8::caso 8 ajuste do teclado

    mov r0,#8h
    ret
caso9::caso 9 ajuste do teclado

    mov r0,#9h
    ret
caso0::caso 0 ajuste do teclado
    mov r0,#0h
    ret

alarme::chama o sistema de alarme
    acall check1
    ret

check1::compara o digito para ver a nessecidade de soar o alarme
    mov a,20h
    mov psw,#00h
    subb a, 30h
    jz check2
    ret

check2::compara o digito para ver a nessecidade de soar o alarme
    mov a,21h
    mov psw,#00h
    subb a, 31h
    jz check3
    ret

check3::compara o digito para ver a nessecidade de soar o alarme
    mov a,22h
    mov psw,#00h
    subb a, 32h
    jz check4
    ret

check4::compara o digito para ver a nessecidade de soar o alarme
    mov a,23h
```



```
mov psw,#00h
subb a, 33h
jz check5
ret
```

check5:;compara o digito para ver a nessecidade de soar o alarme

```
mov a,24h
mov psw,#00h
subb a, 34h
jz check6
ret
```

check6:;compara o digito para ver a nessecidade de soar o alarme

```
mov a,25h
mov psw,#00h
subb a, 35h
jz disparador
ret
```

disparador:;realiza os procedimentos do alarme

```
acall atividade
acall delay
acall delay
acall delay
acall delay
acall delay
acall delay
acall delay
acall delay
acall delay
acall estrutura
ret
```

atividade:;printa o alarme

```
MOV A, #4h
ACALL posicionaC
```

```
MOV A, #61H
ACALL enviaLetra
```

```
MOV A, #5h
ACALL posicionaC
```

```
MOV A, #63H
ACALL enviaLetra
```

```
MOV A, #6h  
ACALL posicionaC
```

```
MOV A, #6fH  
ACALL enviaLetra
```

```
MOV A, #7h  
ACALL posicionaC
```

```
MOV A, #72H  
ACALL enviaLetra
```

```
MOV A, #8h  
ACALL posicionaC
```

```
MOV A, #64H  
ACALL enviaLetra
```

```
MOV A, #9h  
ACALL posicionaC
```

```
MOV A, #61H  
ACALL enviaLetra
```

```
MOV A, #0ah  
ACALL posicionaC
```

```
MOV A, #20H  
ACALL enviaLetra
```

```
MOV A, #0bh  
ACALL posicionaC
```

```
MOV A, #20H  
ACALL enviaLetra
```

ret

leituraTeclado:

MOV R0, #0

MOV P0, #0FFh

CLR P0.0

CALL colScan

JB F0, finish

SETB P0.0

CLR P0.1

CALL colScan

JB F0, finish

SETB P0.1

CLR P0.2

CALL colScan

JB F0, finish

SETB P0.2

CLR P0.3

CALL colScan

JB F0, finish

finish:

RET

colScan:

JNB P0.4, gotKey

INC R0

JNB P0.5, gotKey

INC R0

JNB P0.6, gotKey

INC R0

```
    RET
gotKey:
    SETB F0
    RET

lcd_init:

    CLR RS

    CLR P1.7
    CLR P1.6
    SETB P1.5
    CLR P1.4

    SETB EN
    CLR EN

    CALL delay

    SETB EN
    CLR EN

    SETB P1.7

    SETB EN
    CLR EN

    CALL delay

    CLR P1.7
    CLR P1.6
    CLR P1.5
    CLR P1.4

    SETB EN
    CLR EN
```

SETB P1.6

SETB P1.5

SETB EN

CLR EN

CALL delay

CLR P1.7

CLR P1.6

CLR P1.5

CLR P1.4

SETB EN

CLR EN

SETB P1.7

SETB P1.6

SETB P1.5

SETB P1.4

SETB EN

CLR EN

CALL delay

RET

enviaLetra:

SETB RS

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

SETB EN

CLR EN

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
```

```
SETB EN
CLR EN
```

```
CALL delay
CALL delay
RET
```

posicionaC:

```
CLR RS
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
```

```
SETB EN
CLR EN
```

```
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
```

```
SETB EN
CLR EN
```

```
CALL delay
CALL delay
RET
```

retornaCursor:

```
CLR RS
```

```
CLR P1.7  
CLR P1.6  
CLR P1.5  
CLR P1.4
```

```
SETB EN  
CLR EN
```

```
CLR P1.7  
CLR P1.6  
SETB P1.5  
SETB P1.4
```

```
SETB EN  
CLR EN
```

```
CALL delay  
RET
```

clearDisplay:

```
CLR RS  
CLR P1.7  
CLR P1.6  
CLR P1.5  
CLR P1.4
```

```
SETB EN  
CLR EN
```

```
CLR P1.7  
CLR P1.6  
CLR P1.5  
SETB P1.4
```

```
SETB EN  
CLR EN
```

```
CALL delay  
RET
```

espera:

```
MOV R7, #1000  
acall delay  
DJNZ R7, $
```

RET

delay:

MOV R7, #50

DJNZ R7, \$

RET