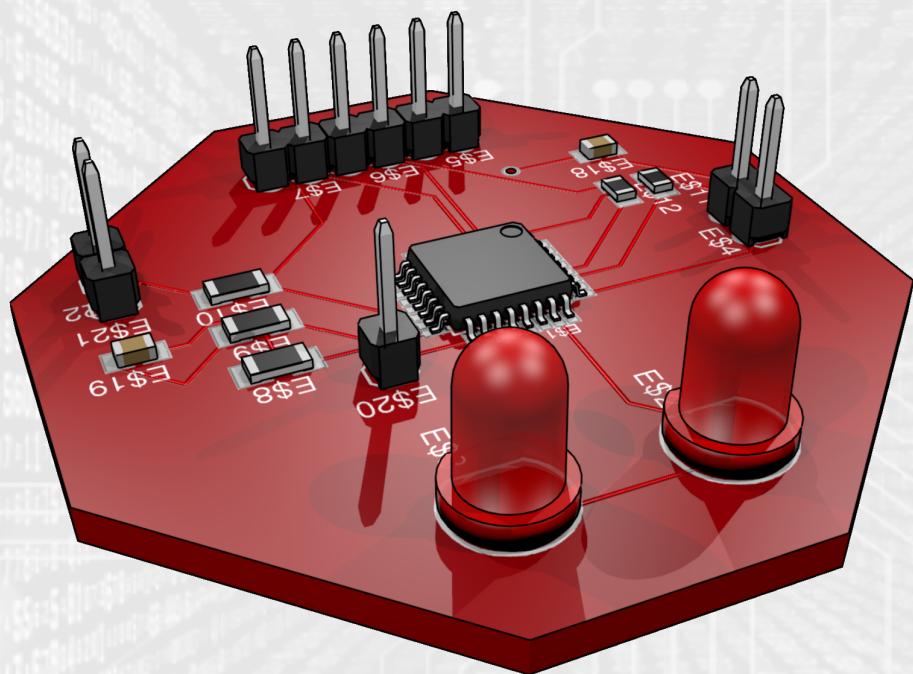


# Projeto de Dispositivos Decodificados Utilizando o Microcontrolador 80c51

Aluno: Fabiano Aparecido Marino  
NºUsp:7143980

8 de maio de 2016





Uma das características do microcontrolador 80c51 que o faz diferenciado é a possibilidade de se construir um hardware com dispositivos decodificados e fazer o envio de um dado para um devido dispositivo de forma que o endereço em que o mesmo está decodificado possibilite o envio do dado ao dispositivo. Isto é possível devido ao compartilhamento da porta *P0* e *P2* sendo que a *P0* envia os dados e junção das portas possibilita o endereçamento do ponteiro DPTR.

O objetivo do seguinte projeto é construir um hardware com o 80c51 de forma que contenha os seguintes periféricos:

- Conversor ADC *Analog to Digital Converter*.
- Conversor DAC *Digital to Analog Converter*.
- RS232 Terminal.
- Memória Ram de 8k.
- Memória Rom de 8k.
- Entrada Para Contador ou interrupção, podendo ser programada para ler um optoacoplador ou dispositivos dessa natureza.
- Teclado de Dezesseis Botões.

Abaixo encontrase a descrição, breve, de cada componente e o endereço que o mesmo ocupa.

- *Memória Ram 6264 0000h–1ffh* : Trata-se de uma memória volátil de capacidade de armazenamento de 8kbytes. O datasheet da mesma encontra-se no seguinte endereço: <http://users.ece.utexas.edu/~valvano/Datasheets/6264.pdf>.
- *Memória Rom 2764 2000h–3fffh* : Trata-se de uma memória não volátil de 8kbytes de armazenamento que pode ser utilizada nos projetos caso se necessite de mais memória do que os 4kbytes que o 80c51 oferece. O datasheet do componente pode ser encontrado com no seguinte endereço : [http://www.futurlec.com/Memory/2764\\_Datasheet.shtml](http://www.futurlec.com/Memory/2764_Datasheet.shtml)
- *Conversor ADC0808 8000h–9fffh* : Trata-se do componente responsável por possibilitar a conversão de analógica para digital possibilitando assim o processamento dos dados provindos de meios externos a placa na forma analógica. O datasheet do componente encontra-se no seguinte endereço : <http://pdf.datasheetcatalog.com/datasheet/nationalsemiconductor/DS005687.PDF>
- *Conversor DAC0808 6000h–7fffh* : Trata-se do componente responsável por fazer a conversão dos dados processados para sua forma analógica para que possam ser utilizados em atuadores o que inclui uma diversidade de possibilidades. O datasheet do componente encontra-se no seguinte endereço : <http://www.learn-c.com/adc0809.pdf>
- *Latch 74hc373* : Trata-se do dispositivo responsável por fazer a demultiplexação entre duto e dados e endereço das portas p2 e p0 do micro 80c51. [http://www.nxp.com/documents/data\\_sheet/74HC\\_HCT373.pdf](http://www.nxp.com/documents/data_sheet/74HC_HCT373.pdf)
- *74ls138* : Trata-se do dispositivo responsável por decodificar cada elemento periférico que recebera ou enviara ao micro informações referente ao duto de dados. O datasheet do componente encontra-se no seguinte endereço : [http://pdf.datasheetcatalog.com/datasheets/90/232315\\_DS.pdf](http://pdf.datasheetcatalog.com/datasheets/90/232315_DS.pdf)
- *Display LM016* : Dispositivo que possibilita a comunicação com o usuário visualmente. Trata-se de um display de duas linhas e dezesseis colunas.



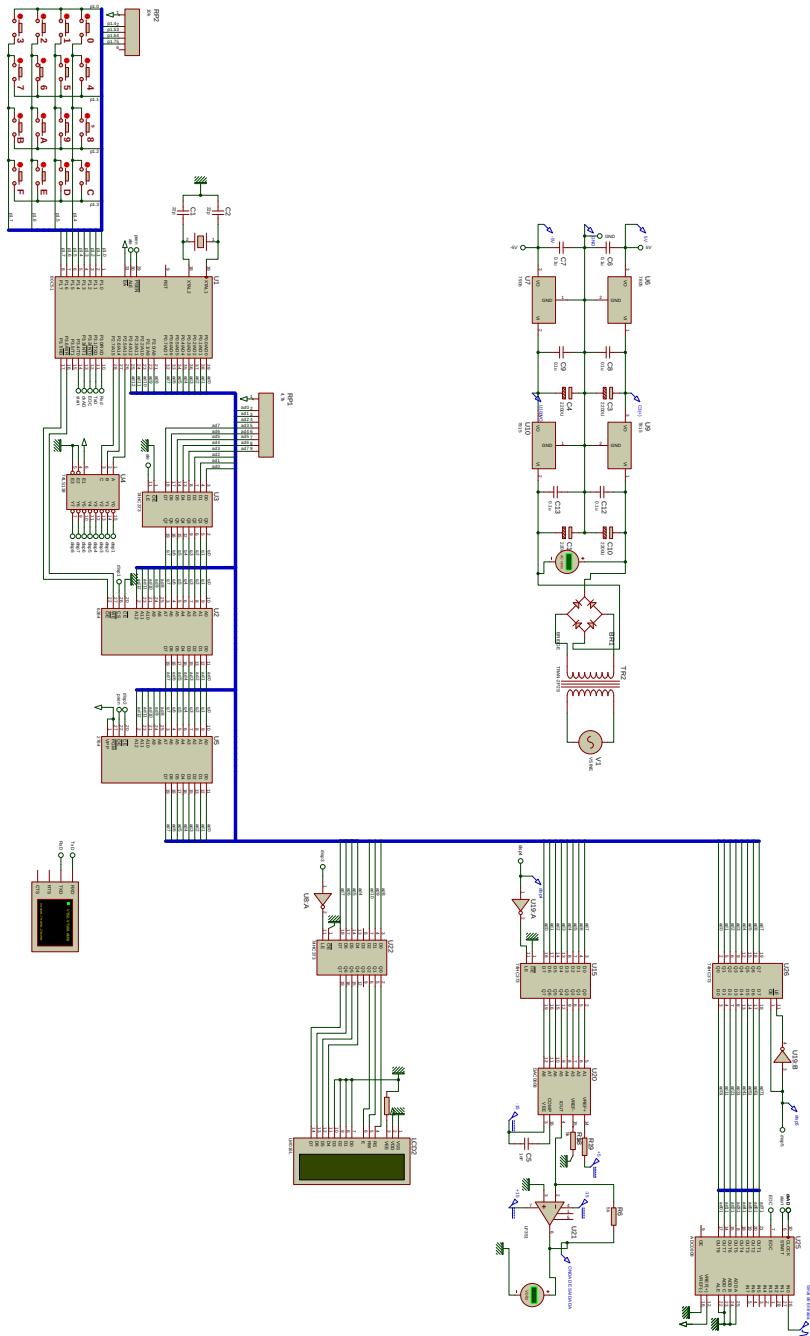
- *Teclado de Cristal Líquido 4000h–5ffh* Possibilita ao usuário que entre com valores que posteriormente serão processados pelo micro.
- *Comunicação serial padrão RS232* Possibilita a comunicação com o usuário via interface serial.
- *Fonte de Alimentação com tensões de fornecimento de +5, -5, +15, -15V*
- *Driver de motor de passo* A placa acompanha um driver para controle de um motor de passo, sendo o mesmo constituido de um controlador l297 e um dispositivo de potência, o l298, pronto para ser usado na placa microprocessada.

Na Próxima pagina encontra-se o esquemático que corresponde ao projeto em questão.O mesmo foi criado e simulado no software proteus ISIS e sua confecção se deu na Plataforma Ares.



## 1 Simulações de todas as possibilidades de caminho dos dados no barramentos e periféricos

As imagens abaixo mostram as simulações da memória ram, conversor AD e DA e display de sete segmentos. O programa para simulação dos periféricos iterativos, como teclado matricial e terminal serial ficam disponíveis no apêndice.



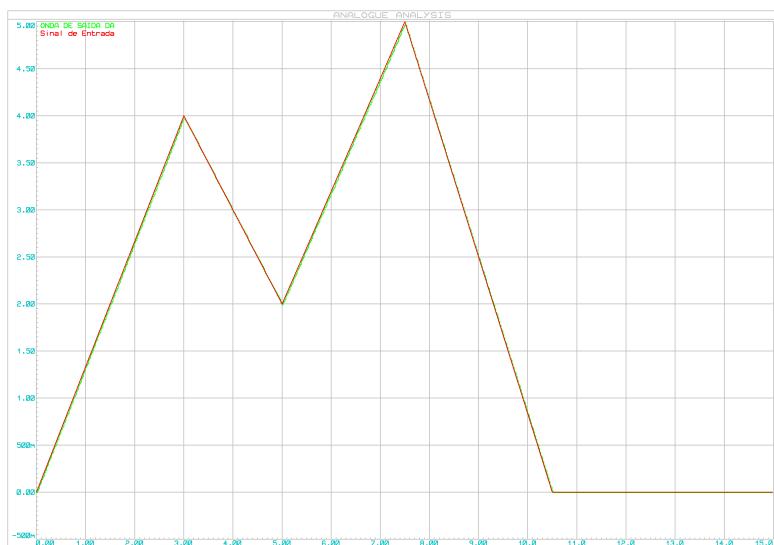


Figura 1: Simulação Conversores AD DA e 80c51

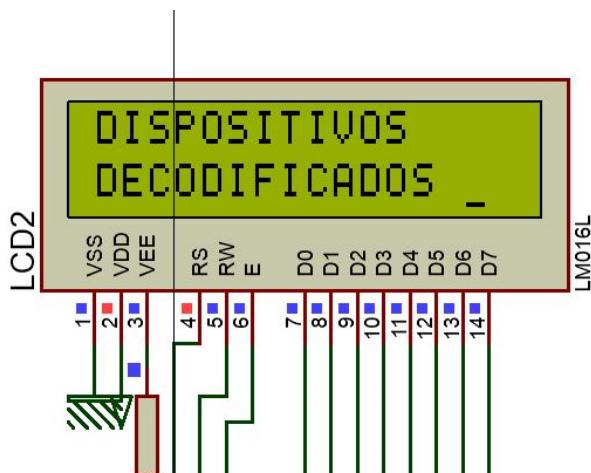


Figura 2: Display LCD Funcionando No Projeto de Dispositivos Decodificados

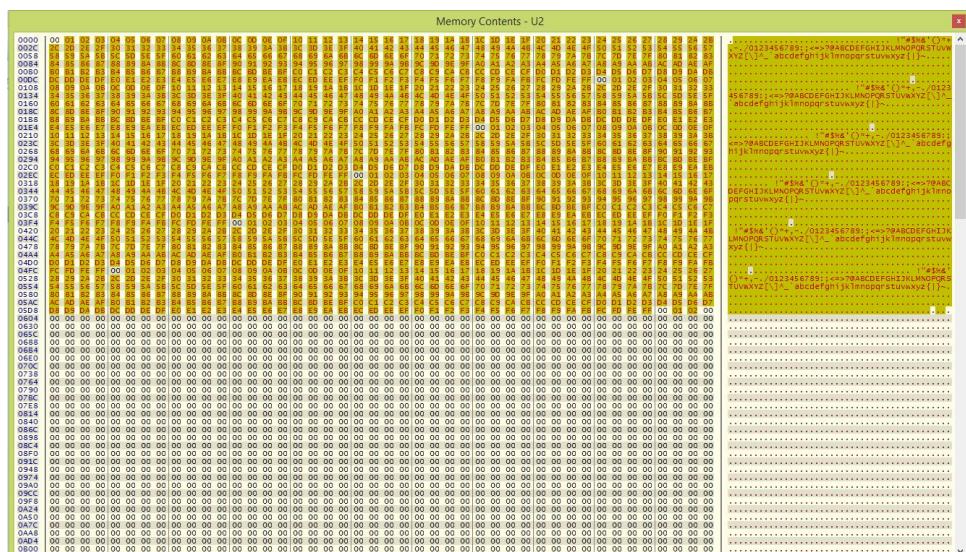


Figura 3: Conteúdo da Memória Ram Após sua Utilização

Abaixo encontra-se a placa resultante do projeto em questão.

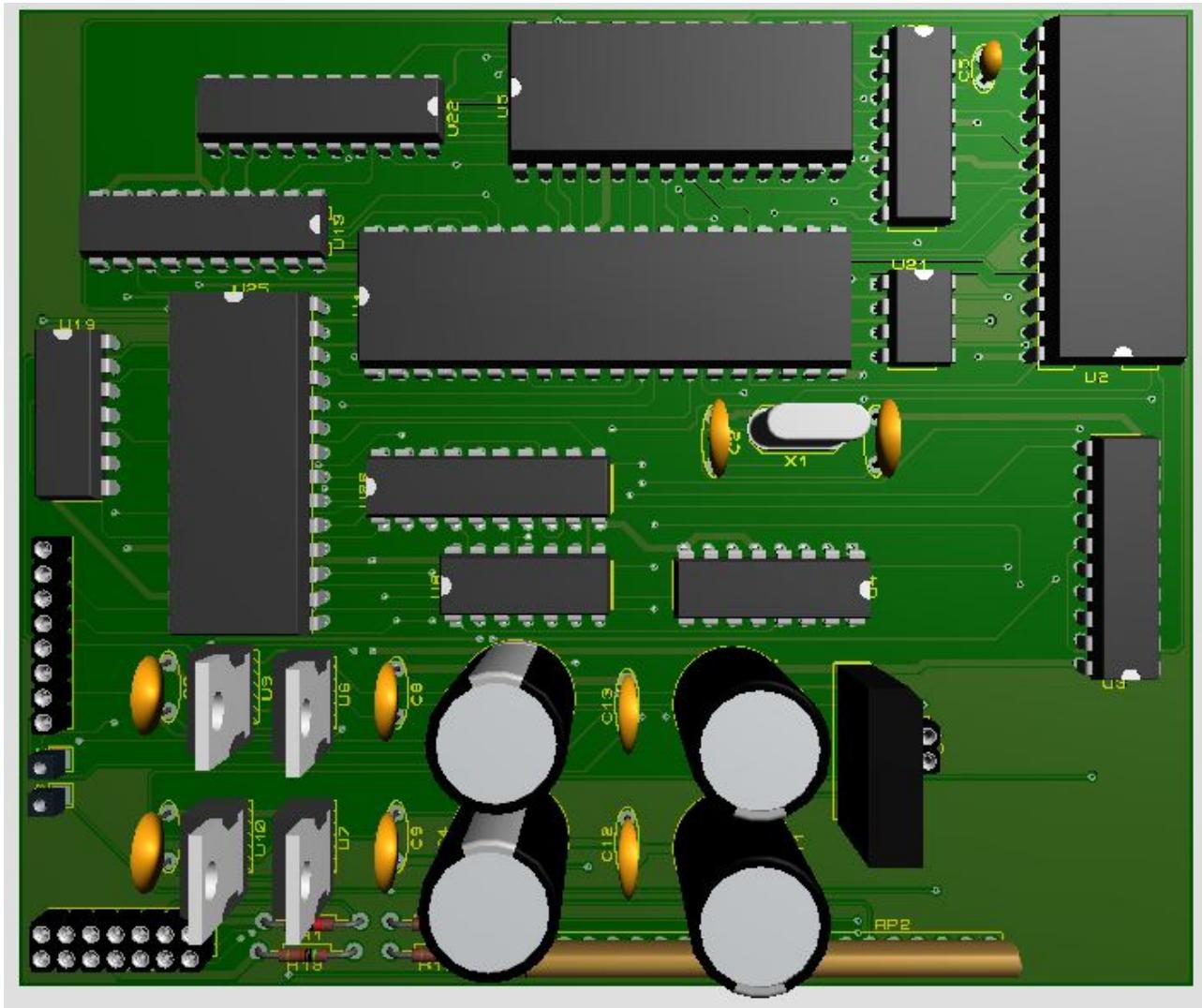


Figura 4: Placa Renderizada do projeto em questão



## 2 Driver Do Motor de Passo

A imagem com o esquemático do driver encontra-se abaixo e sua renderização em placa.

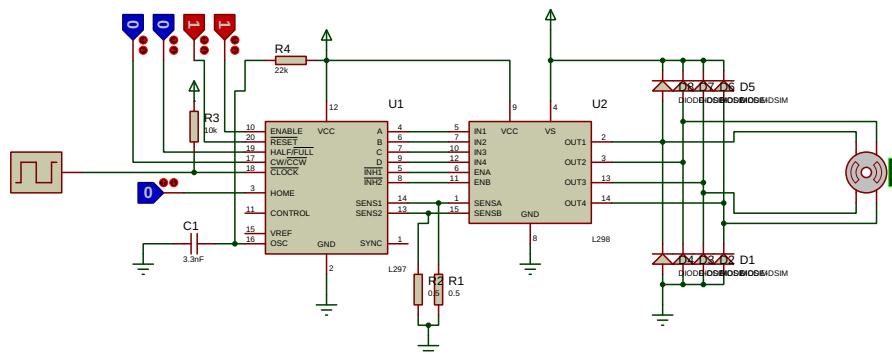


Figura 5: Conteúdo da Memória Ram Após sua Utilização

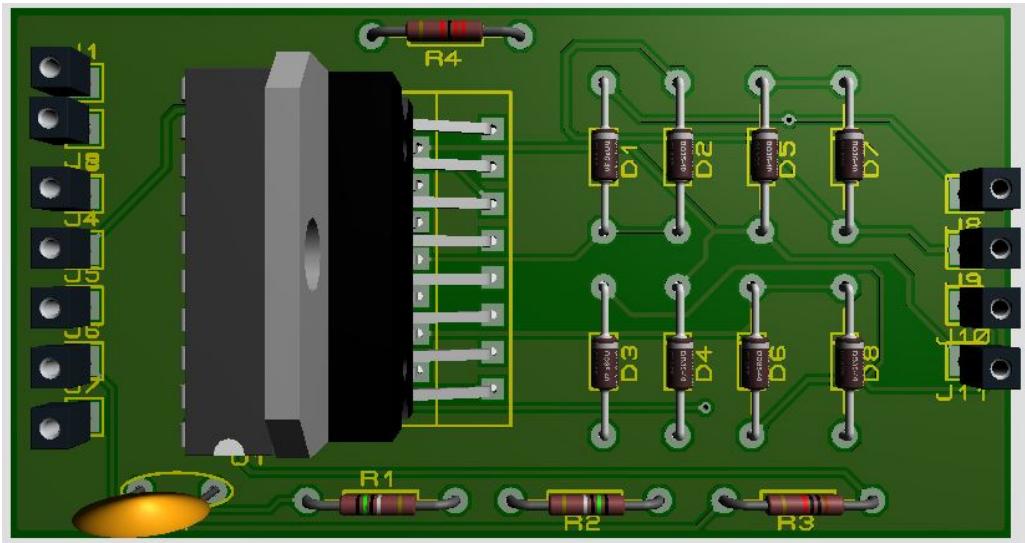


Figura 6: Driver do motor de passo rendeirizado



### 3 Programas

#### 3.1 Programa Conversor AD DA

```
1  CLOCK  EQU P3.3 ;
2  START  EQU P3.4 ;
3  EOC    EQU P3.2 ;

5  ORG 0000H;
6  SJMP INICIO;

7
8  ORG 0003H
9  MOV A,P0;
10 RETI;

11 ORG 000BH;      INTERRUP O DO TIMER0 PARA O START DO ADC
12 CPL CLOCK;
13 LCALL INICIA_TIMER0;
14 RETI;

17 INICIO: MOV P0,#0FFH;
18   SETB P2.7;
19   CLR P2.6;
20   CLR P2.5;
21   CLR CLOCK;
22   CLR START;
23   LCALL INTERRUPTION_CONFIG;
24   LCALL INICIA_TIMER0;
25   ;LCALL DELAY10US;
26   LCALL PULSE;

27 LOOP:
28   CLR P2.7;
29   CLR P2.6;
30   CLR P2.5;
31   MOV P0,#0FFH;
32   JNB EOC, $;
33   SETB P2.7;
34   CLR P2.6;
35   CLR P2.5;

37
38   MOV A,P0;
39   MOV P0,A;
40   CLR P2.7;
41   SETB P2.6;
42   SETB P2.5;

43   LCALL DELAY10US;
44   LCALL PULSE;

47 SJMP LOOP;

49 INTERRUPTION_CONFIG:
50   SETB EA;
51   ;SETB EX0;
52   SETB ET0;
53   SETB PT0;
54   MOV TCON,#11H;
55   RET;

57 INICIA_TIMER0:
58   MOV TH0,#0FFH;
59   MOV TL0,#0FEH;
```



```
61     SETB TR0;  
62     RET;  
  
63 DELAY10US:  
64     MOV R0, #003h  
65     NOP  
66     DJNZ R0, $  
67     NOP  
68     NOP  
69     RET;  
  
70 PULSE:  
71     SETB START;  
72     MOV R2, #002h  
73     MOV R1, #087h  
74     MOV R0, #00Fh  
75     NOP  
76     DJNZ R0, $  
77     DJNZ R1, $-5  
78     DJNZ R2, $-9  
79     MOV R0, #00Eh  
80     DJNZ R0, $  
81     CLR START;  
82     RET;  
  
84 END;
```

### 3.2 Programa Display LCD

```
1 RS EQU P2.0 ;  
2 ;RS=0 ENVIA INSTRUES PARA O DISPLAY  
3 ;RS=1 ENVIA DADOS A SEREM MOSTRADOS NO DISPLAY.  
4 RW EQU P2.1 ;  
5 ;RW=0 PARA A ESCITA OU ENVIO DE INSTRUES NO DISPLAY  
6 ;RW=1 PARA A LEITURA DE CARACTERES DO LCD  
7 E EQU P2.2 ;  
8 ; AP S O DADO SER COLOCADO NO DUTO, TANTO PARA INSTRUES COMO ESCRITA DE CARACTERES,  
9 DEVE SER DADO UM PULSO DE ENABLE NO DISPLAY.  
10 DADOS_INSTRUCOES EQU P0 ;  
11 ;FAZ UMA DIRETIVA NOMIAL PARA A PORTA AONDE ESTE LIGADO OS DADOS DO DISPLAY.  
12  
13 ORG 0000H;  
14 CLR P2.5 ;  
15 SETB P2.6 ;  
16 CLR P2.7 ;  
17 LCALL INIT_LCD ;  
18 MOV A,#00H; ENDEREO DA PRIMEIRA POSIO DA PRIMEIRA LINHA.  
19 LCALL POS_LCD ;  
20 MOV DPTR,#TAB1 ;  
21 LCALL ENVIA_FRASE ;  
22 MOV A,#40H; ENDEREO DA PRIMEIRA POSIO DA SEGUNDA LINHA.  
23 LCALL POS_LCD ;  
24 MOV DPTR,#TAB2 ;  
25 LCALL ENVIA_FRASE ;  
26 MOV P1,#00H ;  
27 SJMP $ ;  
  
28 INIT_LCD :  
29     MOV A,#32H; N O SEI OQUE SIGNIFICA.(PESQUISAR)  
30     LCALL WRITEINSTRUCOES ;
```



```
31 MOV A,#28H;    MODO 2 LINHAS DE 4 BITS COM MASTRIZ 5/7 PONTOS.  
32 LCALL WRITEINSTRUCOES;  
33 MOV A,#0EH;    SEM CURSOR.  
34 LCALL WRITEINSTRUCOES;  
35 MOV A,#06H;    INCREMENTA O CURSOR A CADA ESCRITA DE DADOS.  
36 LCALL WRITEINSTRUCOES;  
37 MOV A ,#01H;    LIMPA O DISPLAYE RETORNA O CURSOR PARA O INICIO.  
38 LCALL WRITEINSTRUCOES;  
39 RET;  
  
41 WRITEINSTRUCOES:  
42     SETB E;  
43     CLR RS;      MODO DE INTRUS  ES ;  
44     CLR RW;      MODO DE ESCRITA DO LCD;  
45     MOV R7,A;  
46     ANL A,#0F0H;  
47     MOV DADOS_INSTRUCOES,A;  
48     CLR E;  
49     LCALL WAITLCD;  
50     SETB E;  
51     MOV A,R7;  
52     ANL A,#0FH;  
53     SWAP A;  
54     MOV DADOS_INSTRUCOES,A;  
55     CLR E;  
56     LCALL WAITLCD;  
57     SETB E;  
58     RET;  
  
59 WRITEDADOS:  
60     SETB E;  
61     SETB RS;      MODO DE DADOS/CARACTERES;  
62     CLR RW;      MODO DE ESCRITA;  
63     MOV R7,A;  
64     ANL A,#0F0H;  
65     MOV DADOS_INSTRUCOES,A;  
66     CLR E;  
67     LCALL WAITLCD;  
68     SETB E;  
69     MOV A,R7;  
70     ANL A,#0FH;  
71     SWAP A;  
72     MOV DADOS_INSTRUCOES,A;  
73     CLR E;  
74     LCALL WAITLCD;  
75     RET;  
  
76 WAITLCD:  
77     MOV R1, #006h  
78     MOV R0, #0E4h  
79     NOP  
80     DJNZ R0, $  
81     DJNZ R1, $-5  
82     NOP  
83     NOP  
84     NOP  
85     NOP  
86     RET;  
  
87 POS_LCD:  
88     ADD A,#80H;  
89     LCALL WRITEINSTRUCOES;  
90     RET;
```



```
95 ENVIA_FRASE:  
    CLR A;  
    MOVC A,@A+DPTR;  
CONTINUA:  
99    LCALL WRITEDADOS;  
    INC DPTR;  
101   CLR A;  
    MOVC A,@A+DPTR;  
103   CJNE A,#'$' ,CONTINUA;  
    RET;  
105  
TAB1:   DB 'ASSEMBLY 4 BITS '$'  
TAB2:   DB 'COM DUAS LINHAS '$'  
        END;
```

```
ORG 0000H;  
2  
    LCALL CONFIG_SERIAL;  
4  
COLUNA1:  
6    MOV P1,#00H;  
    CLR P1.3;  
8    SETB P1.0 ;  
ZERO:  
10   JNB P1.4 ,UM;  
12  
    MOV DPTR,#TAB1;  
    LCALL ENVIA_FRASE;  
14    MOV A,#00H;  
    LCALL ENVIA_CARACTER;  
16    LCALL DELAY;  
UM:  
18   JNB P1.5 ,DOIS;  
20  
    MOV DPTR,#TAB1;  
    LCALL ENVIA_FRASE;  
22    MOV A,#01H;  
    LCALL ENVIA_CARACTER;  
24    LCALL DELAY;  
DOIS:  
26   JNB P1.6 ,TRES;  
28  
    MOV DPTR,#TAB1;  
    LCALL ENVIA_FRASE;  
30    MOV A,#02H;  
    LCALL ENVIA_CARACTER;  
32    LCALL DELAY;  
TRES: JNB P1.7 ,COLUNA2;  
34  
    MOV DPTR,#TAB1;  
    LCALL ENVIA_FRASE;  
38    MOV A,#03H;  
    LCALL ENVIA_CARACTER;  
40    LCALL DELAY;  
COLUNA2:  
42    CLR P1.0 ;  
44    SETB P1.1 ;  
QUATRO:  
46
```



```
        JNB P1.4 ,CINCO ;  
48      MOV DPTR,#TAB1 ;  
50      LCALL ENVIA_FRASE ;  
51      MOV A,#04H ;  
52      LCALL ENVIA_CARACTER ;  
53      LCALL DELAY ;  
54  
CINCO:  
55      JNB P1.4 ,SEIS ;  
56  
57      MOV DPTR,#TAB1 ;  
58      LCALL ENVIA_FRASE ;  
59      MOV A,#05H ;  
60      LCALL ENVIA_CARACTER ;  
61      LCALL DELAY ;  
62  
SEIS:  
63      JNB P1.6 ,SETE ;  
64  
65      MOV DPTR,#TAB1 ;  
66      LCALL ENVIA_FRASE ;  
67      MOV A,#06H ;  
68      LCALL ENVIA_CARACTER ;  
69      LCALL DELAY ;  
70  
71 SETE:  
72      JNB P1.7 ,COLUNA3 ;  
73  
74      MOV DPTR,#TAB1 ;  
75      LCALL ENVIA_FRASE ;  
76      MOV A,#07H ;  
77      LCALL ENVIA_CARACTER ;  
78      LCALL DELAY ;  
79  
80 COLUNA3:  
81      CLR P1.1 ;  
82      SETB P1.2 ;  
83 OITO:  
84      JNB P1.4 ,NOVE ;  
85  
86      MOV DPTR,#TAB1 ;  
87      LCALL ENVIA_FRASE ;  
88      MOV A,#08H ;  
89      LCALL ENVIA_CARACTER ;  
90      LCALL DELAY ;  
91  
92 NOVE:  
93      JNB P1.5 ,LETRA_A ;  
94  
95      MOV DPTR,#TAB1 ;  
96      LCALL ENVIA_FRASE ;  
97      MOV A,#09H ;  
98      LCALL ENVIA_CARACTER ;  
99      LCALL DELAY ;  
100  
101 LETRA_A:  
102      JNB P1.6 ,LETRA_B ;  
103  
104      MOV DPTR,#TAB1 ;  
105      LCALL ENVIA_FRASE ;  
106      MOV A,#0AH ;  
107
```



```
110 LCALL ENVIA_CARACTER;
111 LCALL DELAY;

112 LETRA_B:
113 JNB P1.7 ,COLUNA4;

116 MOV D PTR,#TAB1;
117 LCALL ENVIA_FRASE;
118 MOV A,#0BH;
119 LCALL ENVIA_CARACTER;
120 LCALL DELAY;

122 COLUNA4:
123 CLR P1.2 ;
124 SETB P1.3 ;

125 LETRA_C:
126 JNB P1.4 ,LETRA_D;

128 MOV D PTR,#TAB1;
129 LCALL ENVIA_FRASE;
130 MOV A,#0CH;
131 LCALL ENVIA_CARACTER;
132 LCALL DELAY;

134 LETRA_D:
135 JNB P1.5 ,LETRA_E;

136 MOV D PTR,#TAB1;
137 LCALL ENVIA_FRASE;
138 MOV A,#0DH;
139 LCALL ENVIA_CARACTER;
140 LCALL DELAY;

142 LETRA_E:
143 JNB P1.6 ,LETRA_F;

146 MOV D PTR,#TAB1;
147 LCALL ENVIA_FRASE;
148 MOV A,#0EH;
149 LCALL ENVIA_CARACTER;
150 LCALL DELAY;

152 LETRA_F:
153 JNB P1.7 ,FIM;

154 MOV D PTR,#TAB1;
155 LCALL ENVIA_FRASE;
156 MOV A,#0FH;
157 LCALL ENVIA_CARACTER;
158 LCALL DELAY;

160 FIM:
161 LJMP COLUNA1;

164 CONFIG_SERIAL:
165 MOV TMOD,#20H      ;TIMER 1 COMO TEMPORIZADOR DE 8 BITS AUTO-RECARREGVEL.
166 MOV TH1,#253D      ;TH1 E TL1 CARREGADOS PARA BAUD RATE = 9600 .
167 MOV TL1,#253D
168 MOV SCON,#01000000B ;SERIAL MODO 1 (TAXA VARIVEL (10 BITS)) E RECEP O
169                 ;DESABILITADA;
170 SETB TR1           ;LIGA TIMER 1
171 RET;
```



```
172 CONVERTE_HEX_ASC:  
173     CJNE A,#0AH,TESTEA;  
174 TESTEA:  
175     JC NUM;  
176 LETRA:  
177     CLR C;  
178     ADD A,#37H;  
179     RET;  
180 NUM:  
181     CLR C;  
182     ADD A,#30H;  
183     RET;  
184  
185 ENVIA_FRASE:  
186     CLR A;  
187     MOVC A,@A+DPTR;  
188     MOV SBUF,A;  
189     JNB TI,$;  
190     CLR TI;  
191     INC DPTR;  
192     CLR A;  
193     MOVC A,@A+DPTR;  
194     CJNE A,#'$',ENVIA_FRASE;  
195     RET;  
196  
197 ENVIA_CARACTER:  
198     MOV R1,A;  
199     ANL A,#0F0H;  
200     SWAP A;  
201     LCALL CONVERTE_HEX_ASC;  
202     MOV SBUF,A;  
203     JNB TI,$;  
204     CLR TI;  
205     LCALL SEGUNDO_CARACTER;  
206     RET;  
207  
208 SEGUNDO_CARACTER:  
209     MOV A,R1;  
210     ANL A,#0FH;  
211     LCALL CONVERTE_HEX_ASC;  
212     MOV SBUF,A;  
213     JNB TI,$;  
214     CLR TI;  
215     RET;  
216  
217 DELAY:  
218     MOV R2, #01Ah  
219     MOV R1, #0B1h  
220     MOV R0, #017h  
221     NOP  
222     DJNZ R0, $  
223     DJNZ R1, $-5  
224     DJNZ R2, $-9  
225     MOV R0, #06Eh  
226     DJNZ R0, $  
227     NOP  
228     RET;  
229  
230 TAB1:    DB 0AH,0DH,'BOTAO PRESSIONADO = ', '$';  
231     END;  
232
```



### 3.3 Programa do preenchimento da memória ram

```
1      ORG 0000H;  
2  
3      MOV DPTR,#0000H;  
4      MOV A,#00H;  
5  
6  
7 RAM:  
8     MOVX @DPTR,A;  
9     INC DPTR;  
10    INC A;  
11    SJMP RAM;  
12    END;  
13
```