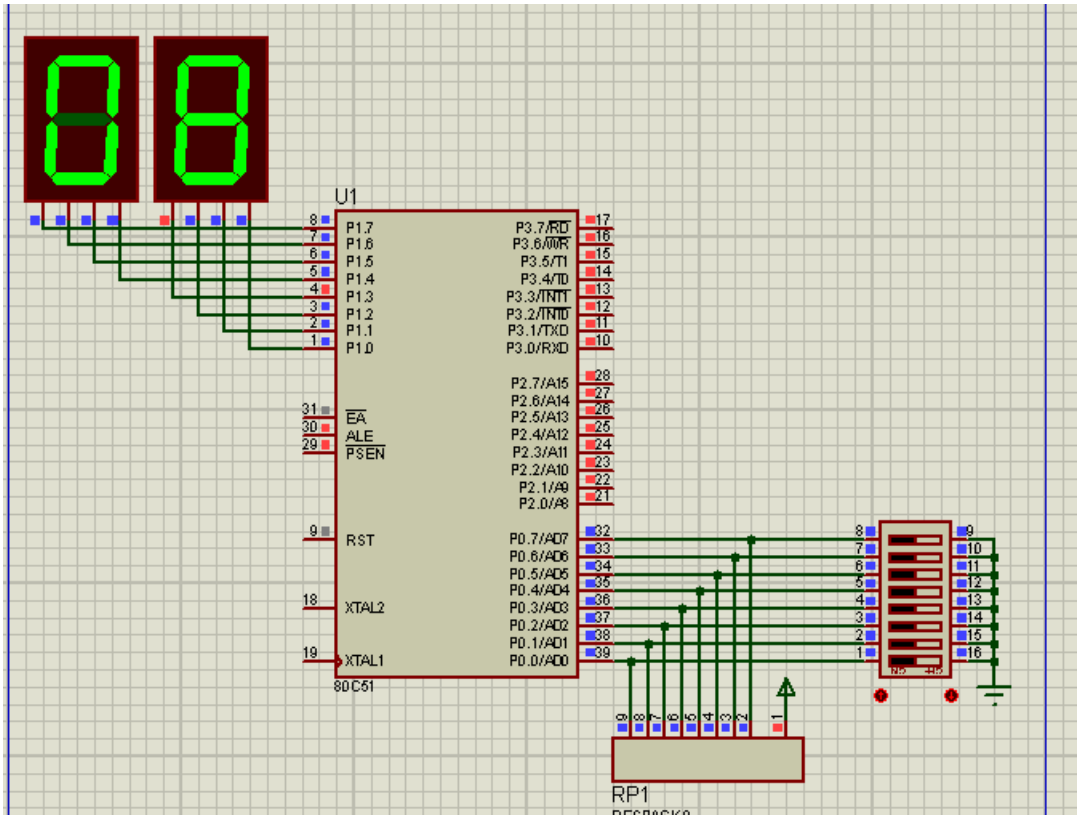
Problem 25



This problem will find a match of a word within a sentence.

We select the sentence by dip-switch

7-segment will display the first match position if found else it will disply EE → not found; or 00 if we select non-existence sentence

Variables

```
1  WORD_POINTER EQU 30H
2  LINE_POINTER EQU 32H
3  INDEXW EQU 34H
4  INDEXL EQU 35H
5
6
7  LSIZE EQU 36H
8  WSIZE EQU 37H
9  SEARCH_SIZE EQU 38H
10 INDEXL2 EQU 39H
```

1→ pointer to word address

2→ pointer to line address

3,4→ indexes used as offsets to word/line pointer

7,8→ line/word size

9→ search size = line size – word size

10→ indexL2 → another index for the line, used as a temporary for indexL

Main code

```
14  START:
15     CLR A
16     MOV INDEXW,A
17     MOV INDEXL,A
18
19     MOV DPTR,#WORD
20     MOV WORD_POINTER,DPL
21     MOV WORD_POINTER+1,DPH
22     CALL GET_STRING_SIZE
23     MOV WSIZE,A
24
25     MOV A,P0
26     CJNE A,#0,TEST2
27     MOV DPTR,#LINE1
28     JMP START1
29  TEST2:
30     CJNE A,#1,TEST3
31     MOV DPTR,#LINE2
32     JMP START1
33  TEST3:
34     CJNE A,#2,TEST4
35     MOV DPTR,#LINE3
36     JMP START1
37  TEST4:
38     MOV P1,#0
39     JMP START
```

15-17 → reset word/line index

19-21 → load word_pointer with the 16-bits address of the 'WORD" to be searched for

22-23→ calculate word size

25-38 → read P0 and determine which line will be used to search within (we have 3 lines in this example)

```
41  START1:
42     MOV LINE_POINTER,DPL
43     MOV LINE_POINTER+1,DPH
44     CALL GET_STRING_SIZE
45     MOV LSIZE,A
46
47     MOV A,LSIZE
48     CLR C
49     SUBB A,WSIZE
50     MOV SEARCH_SIZE,A
51
52  REPEAT:
53     CALL GET_FIRST_BYTE_MATCH
54     JNC NO_MATCH
55
56     CALL MATCH_WHOLE_WORD
57     JC MATCH_FOUND
58     INC INDEXL
59     JMP REPEAT
60  MATCH_FOUND:
61     MOV P1,INDEXL
62     JMP START
63
64  NO_MATCH:
65     MOV P1,#0EEH
66     JMP START
```

42-43→Load line_pointer with 16-bits line address

44-45→ get line size

47→ calculate search_size = Lsize-Wsize → this is the last index to search within a line where after that the remaining characters in the line is less than word size.

53→ call a function that will find the first match with the first character of the word. If no carry → no match found → 65 will display EE

If match found, we call a function to match whole word (56); if carry is '1' → match found→ display match position/index (61);else (58-59) increment line index to point to the next character to the previous match index, and repeat

```
69  GET_STRING_SIZE:
70     MOV B,#0
71  NEXT_CHAR:
72     CLR A
73     MOVC A,@A+DPTR
74     INC DPTR
75     CJNE A,#0,CHECK_CR
76     MOV A,B
77     RET
78  CHECK_CR:
79     CJNE A,#13,CHECK_LF
80     MOV A,B
81     RET
82  CHECK_LF:
83     CJNE A,#10,COUNT_CHAR
84     MOV A,B
85     RET
86  COUNT_CHAR:
87     INC B
88     JMP NEXT_CHAR
89  RET
```

This function will read the string pointed to by DPTR till It reaches its end defined by CR or LF or null. During this process it counts the number of characters and store it in ACC.

70→First it clears B (counter)

72-74→ reads a character

75,79,83→ check if it is null or CR or LF to end the function with ACC has the number of characters (76,80,84); else

87-88→ increment the counter and repeate

```
91  GET_FIRST_BYTE_MATCH:
92     MOV DPL,WORD_POINTER
93     MOV DPH,WORD_POINTER+1
94     CLR A
95     MOVC A,@A+DPTR
96     MOV B,A
97  CONT1:
98     MOV DPL,LINE_POINTER
99     MOV DPH,LINE_POINTER+1
100    MOV A,INDEXL
101    MOVC A,@A+DPTR
102    CJNE A,B,NEXT1
103    SETB C
104    RET
105 NEXT1:
106    INC INDEXL
107    MOV A,INDEXL
108    CLR C
109    SUBB A,SEARCH_SIZE
110    JC CONT1
111    CLR C
112 RET
```

This function will search for the first character in the word

92-96 read first character of the word into B

98-101 read the first character in the line

102 → if equal first word character; set carry (103) and return with indexL point to the first match;else

106→ increment indexL to point to next character

107-110 → check if indexL didn't reach search_size → return with carry cleared (111-112); else search again.

```
114 MATCH_WHOLE_WORD:
115    MOV INDEXL2,INDEXL
116    MOV INDEXW,#0
117    MOV R7,WSIZE
118    DEC R7
119 ALL_BYTES:
120    INC INDEXL2
121    INC INDEXW
122
123    MOV DPL,#WORD
124    MOV A,INDEXW
125    MOVC A,@A+DPTR
126    MOV B,A
127
128    MOV DPL,LINE_POINTER
129    MOV DPH,LINE_POINTER+1
130    MOV A,INDEXL2
131    MOVC A,@A+DPTR
132
133    CJNE A,B,NOT_MATCH2
134
135    DJNZ R7,ALL_BYTES
136    SETB C
137    RET
138 NOT_MATCH2:
139    CLR C
140    RET
```

This function will match the rest of the word

115→ use another line index "INDEXL2" to preserve last line index "INDEXL"

116→ start with word index = 0

117→ R7 is the loop counter (loaded by the word size-1)

120-121 → for loop start by incrementing both indexL2/IndexW

123-126 → read word(indexw) into B

128-131 → read line(indexL2) into A

133 → if A not equal B → no match → clear carry (139) and return

135→ repeat for all word characters.

136→ we reach here if always A = B → whole word match → set carry (136) and return

How word/ lines are stored

```
156 WORD: DB "FINAL",0
157 LINE1: DB "THIF IS FINAL EXAM",13,10
158 LINE2: DB "THIF IS MID EXAM",13,10
159 ;LINE3: DB "FILE FOLDER FINISH",13,10
160 LINE3: DB "FILE FOLDER FFINAL",13,10
```

We use either CR or LF or null as an indicator for line/word end