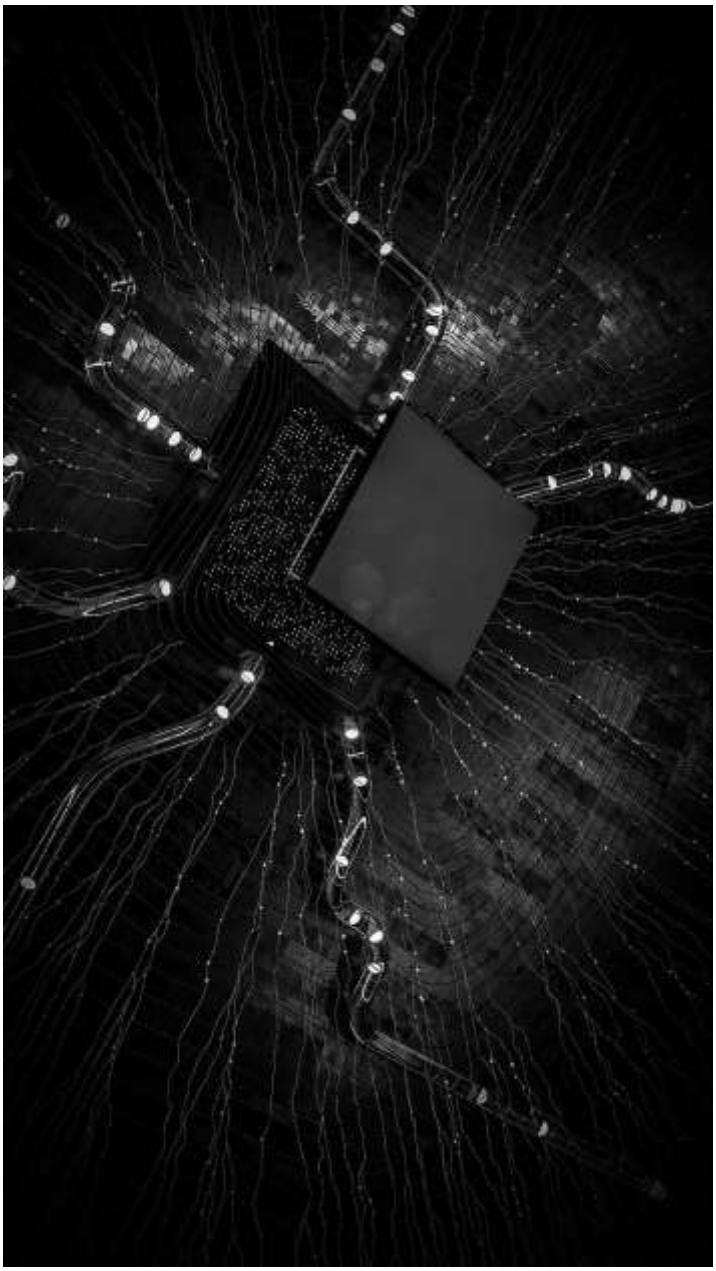


آزمایشگاه ریزپردازنده

تمرین چهار

سینا عربی – سالار جهانشیری – امیرعلی وکیلی

سوالات تحلیلی:

- **کاربرد تراشه 8255 ، سیگنالهای کنترلی و نحوه کار با آن را شرح دهید.**

تراشه 8255

این تراشه یک مدار مجتمع چند منظوره است که در سیستم های کامپیوتری اولیه برای برای ارائه پورت های ورودی/خروجی قابل برنامه ریزی استفاده میشد.

کاربرد های تراشه 8255

- جمع آوری داده های حسگر
- کنترل موتور ها و LED ها
- رابط با دستگاه های جانبی

سیگنال های کنترلی تراشه 8255

- Reset: این سیگنال برای ریست کردن 8255 به حالت پیش فرض استفاده میشود.
- Ao,A1:این سیگنال ها برای انتخاب بین سه پورت استفاده میشود.
- WR: سیگنال برای نوشتن
- RD : سیگنال برای خواندن
- CS:سیگنال برای انتخاب ۸۲۵۵

نحوه کار:

ابتدا میبایست سیگنال های کنترلی مناسب را برای انتخاب پورت I/O را تنظیم کنیم.

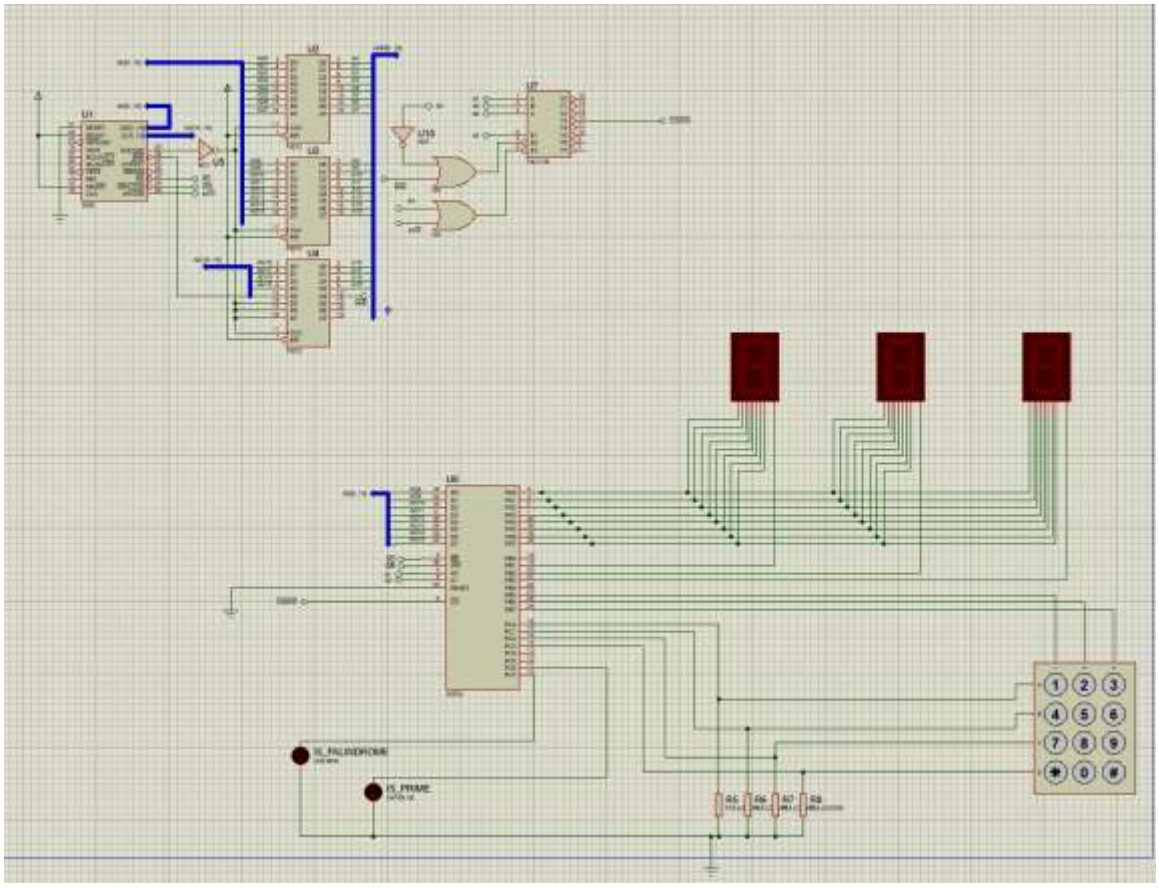
سپس CS را صفر میکنیم تا تراشه انتخاب شود.

سپس میتوان داده ها را با توجه به پورت انتخاب شده با توجه به سیکنال های WR,RD خواند یا نوشت.

- **نحوه کار با صفحه کلید ماتریسی و خواندن کلیدهای آن را شرح دهید.**

در این صفحه کلید ماتریسی سه ستون و چهار سطر داریم که اینها به پورت های IO وصل میشوند و وقتی یک کلید را میزنیم، فاصله خازن کم میشود و جریان برقرار میشود و یک اختلاف ولتاژی بدست میاید که یک طرف به زمین وصل است و طرف دیگر به باتری و یا power وصل می‌باشد،هرطلاقی در این مدار یک نقطه و یا یک کلید در صفحه را نشان میدهد که با فشار کلید و برقراری جریان و اختلاف ولتاژ ستون و سطر مورد نظر آن کلید فعال میشود و اگر این خروجی ها را به پورت های IO وصل کنیم میتوانیم با حالت بندی کلید ها تشخیص دهیم که چه کلیدی فشار داده شده است.

دستور کار:



شماتیک کلی مدار(در هر بخش به صورت جزئی توضیح داده شده)

برای نمایش اعداد ورودی توسط کاربر از یک قطعه 7-segment با دیکودینگ مود آنود انتخاب کردیم، برای نمایش اعداد روی این نمایشگر عدد باینری معادل هر عدد که هر تکه از 7-seg را روشن می‌کند، بدست آورده و در آرایه Digits ذخیره می‌کنیم.

آدرس مودینگ قطعه 8255a برای نوشتن در پورت A و B، 4 بیت کم‌ارزش تر C برای ورودی و 4 بیت پر ارزش تر برای نوشتن C استفاده می‌کنیم.

```
;You may write one time jobs here
      MOV DX, 0AFh
      MOV AL, 081h          ; IO(1),Mode 0(00), Port A output(0),
                            ;Port C upper output, Port b Mod 0 output,
                            ;PortC lower input
      OUT DX, AL

     ; BH FIRST DIGIT, DL SECOND, DH last Digit
     MOV AH, DIGITS[0]
     MOV BL, AH
     MOV BH, AH
     MOV DL, AH
     MOV DH, AH
     ; At first assign 0 to all
```

ابتدا تمامی رجیسترها را با عدد دیکود شده 0 برای 7-seg مقدار دهی اولیه می‌کنیم.

در ادامه در یک لوپ بی‌نهایت عملیات ورودی گرفتن از کاربر را تعریف می‌کنیم، به این صورت که با هر بار فشردن دکمه از صفحه کلید ماتریسی از طریق پورت‌ها بررسی می‌کنیم که در کدام سطر و ستون دکمه ای فشرده شده و سپس با توجه به آن عدد، حالت دیکود شده آن برای 7-seg را در رجیستر BL نگه می‌داریم.

```
ACTIVE_FIRST_COL:
      MOV AL, 00100000b                ; Activate Column 1
      OUT 0ABh, AL                     ; Send to PORT B

      IN AL, 0ADh                      ; Read from PORT C
      AND AL, 0Fh                      ; See the Lower 4 bits of PORT C
      JNZ CHECK_FIRST_COL

ACTIVE_SECOND_COL:
      MOV AL, 01000000b                ; Activate Column 2
      OUT 0ABh, AL

      IN AL, 0ADh
      AND AL, 0Fh
      JNZ CHECK_SECOND_COL

ACTIVE_THIRD_COL:
      MOV AL, 10000000b                ; Activate Column 3
      OUT 0ABh, AL

      IN AL, 0ADh
      AND AL, 0Fh
      JNZ CHECK_THIRD_COL

CHECK_FIRST_COL:
  CHECK_ONE:
      TEST AL, 01h                     ; was it one ?
      JZ CHECK_FOUR
      CALL SAVE_DELETED_DIGIT          ; Secure the last digit that is currently inside of BL
      MOV BL, DIGITS[1]
      CALL SCROLL_TO_LEFT              ; Scroll the numbers to the left
      CALL SHOW                        ; SHOW 1
      JMP ENDLESS

  CHECK_FOUR:
      TEST AL, 02h                     ; was it four ?
      JZ CHECK_SEVEN
      CALL SAVE_DELETED_DIGIT          ; Secure the last digit that is currently inside of BL
      MOV BL, DIGITS[4]
      CALL SCROLL_TO_LEFT
      CALL SHOW
      JMP ENDLESS
```

در ادامه با فشرده شدن عدد بعدی هر عدد را به رجیستر بعدی منتقل می‌کنیم(طبق قرار رقم اول در BH، رقم دوم در DL و رقم سوم را در DH نگه می‌داریم)

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
SCROLL_TO_LEFT PROC NEAR                    ; BL-1, BH-2, DL-3,DH-4 AH TEMP
   MOV AH, BL
   MOV BL, BH
   MOV BH, DL
   MOV DL, DH
   MOV DH, AH
RET
SCROLL_TO_LEFT ENDP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

در مرحله بعد برای نمایش اعداد روی 7-seg رجیسترهای شامل اعداد دیکود شده را در پورت متصل به نمایشگر می‌نویسیم. بین هر نوشتن یک wait پیاده‌سازی شده تا اعداد به درستی از روی پورت در نمایشگر نشان داده شوند.

```
        MOV AL, 2h
        OUT 0ABh, AL

        MOV AL, BH
        OUT 0A9h, AL

        CALL WAIT_A_LITTLE

        MOV AL, 4h
        OUT 0ABh, AL

        MOV AL, DL
        OUT 0A9h, AL

        CALL WAIT_A_LITTLE

        MOV AL, 8h
        OUT 0ABh, AL

        MOV AL, DH
        OUT 0A9h, AL

        CALL SAVE_INPUT_NUMBER

        CALL WAIT_A_LITTLE

        ;JMP ENDLESS
RET
SHOW_NUMBERS ENDP
```

سپس برای بررسی اول یا پالیندرومی بودن عدد به سراغ متد Save_Input_Number می‌رویم، برای پیاده سازی این بخش اعدادی که برای 7-seg دیکود شده بود را دوباره به عدد دسیمال انکود کرده و عدد سه رقمی را تولید می‌کنیم.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;
FUNC_DECODE_TO_DECIMAL PROC NEAR

                CMP ENCODED_VAL, 0C0h
                JNE NEXT1
                MOV THE_DECODED_VAL,0
                RET
NEXT1:          CMP ENCODED_VAL, 0F9h
                JNE NEXT2
                MOV THE_DECODED_VAL,1
                RET
NEXT2:          CMP ENCODED_VAL, 0A4h
                JNE NEXT3
                MOV THE_DECODED_VAL,2
                RET
NEXT3:          CMP ENCODED_VAL, 0B0h
                JNE NEXT4
                MOV THE_DECODED_VAL,3
                RET
NEXT4:          CMP ENCODED_VAL, 99h
                JNE NEXT5
                MOV THE_DECODED_VAL,4
                RET
NEXT5:          CMP ENCODED_VAL, 92h
                JNE NEXT6
                MOV THE_DECODED_VAL,5
                RET
NEXT6:          CMP ENCODED_VAL,82h
                JNE NEXT7
                MOV THE_DECODED_VAL,6
                RET
NEXT7:          CMP ENCODED_VAL,0F8h
                JNE NEXT8
                MOV THE_DECODED_VAL,7
                RET
NEXT8:          CMP ENCODED_VAL,80h
                JNE NEXT9
                MOV THE_DECODED_VAL,8
                RET
NEXT9:          MOV THE_DECODED_VAL,9
                RET

RET
FUNC_DECODE_TO_DECIMAL ENDP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
SAVE_INPUT_NUMBER PROC NEAR
   SUB AX, AX

   MOV ENCODED_VAL, BH
   CALL FUNC_DECODE_TO_DECIMAL
   ADD Al,THE_DECODED_VAL

   MUL NUMBER_10
   MOV ENCODED_VAL, DL
   CALL FUNC_DECODE_TO_DECIMAL
   ADD Al,THE_DECODED_VAL

   MUL NUMBER_10
   MOV ENCODED_VAL, DH
   CALL FUNC_DECODE_TO_DECIMAL
   ADD Al,THE_DECODED_VAL

   MOV NUMB, Al
   MOV NUMB_WORD, AX


   CALL IS_PRIME

   CALL IS_PALINDROME
   CALL CREATE_LED_PORT_NUMBER

RET
SAVE_INPUT_NUMBER ENDP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

در ادامه متدهای IS_PRIME و IS_PALINDROME برای بررسی اول بودن و پالیندرومی بودن عدد صدا زده می‌شوند و درصورتی که تایید شود نتیجه در فلگ‌های FLAG_IS_PRIME و FLAG_IS_PAL بصورت اعداد باینری بیتی که خروجی را روی پورت C به LED های مربوطه وصل شده می‌نویسیم.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
IS_PRIME PROC NEAR
     PUSH CX
     PUSH DX

     MOV AX, NUMB_WORD
     CMP AX, 1D
     JE IS_NOT_PRIME

     MOV CX, 02H

LOOP_CHECK:
     MOV AX, NUMB_WORD
     CMP CX, AX
     JE RETURN_TRUE


     SUB DX, DX
     DIV CX
     INC CX
     CMP DX, 0H
     JNE LOOP_CHECK

     IS_NOT_PRIME:
     MOV FLAG_IS_PRIME, 0H

     JMP END_IS_PALINDORME



     RETURN_TRUE:
      MOV FLAG_IS_PRIME, 01000000B

     END_IS_PALINDORME:
      POP DX
      POP CX

     RET

IS_PRIME ENDP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

کد بررسی اول بودن عدد N با بررسی بخش‌پذیری بر اعداد کوچکتر از N.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
IS_PALINDROME PROC
     PUSH BX

     CALL REVERSE
     SUB AX, AX
     MOV AX, NUMB_WORD
     MOV BX, REVERSED_NUMB
     CMP AX, BX
     JNE IS_PALINDROME_FALSE
     MOV FLAG_IS_PAL, 10000000B

     POP BX
     RET

     IS_PALINDROME_FALSE:
         MOV FLAG_IS_PAL, 0H
         POP BX
     RET

IS_PALINDROME ENDP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

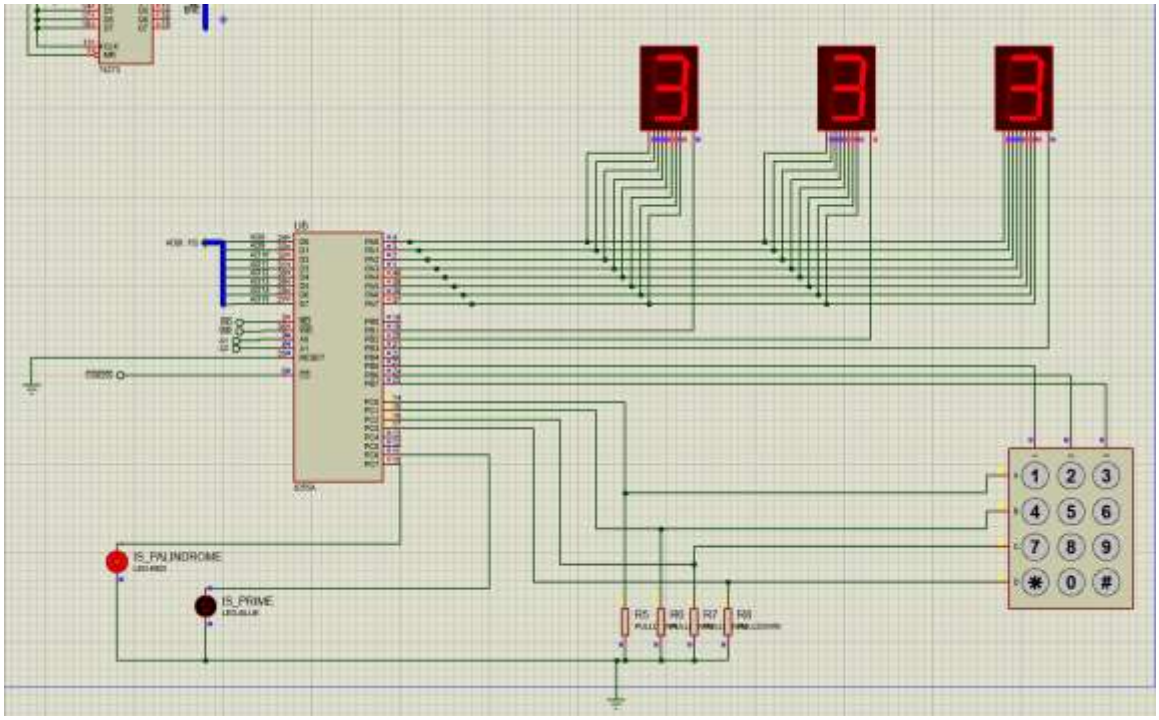
```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
REVERSE PROC
    PUSH BX
    PUSH DX

    MOV AX, NUMB_WORD
    SUB BX, BX

LOOP_REVERSE:
    SUB DX, DX
    DIV NUMBER_10_WORD
    MOV SI, OFFSET REVERSED_NUMB
    MOV [SI], AX
    MOV AX, BX
    MOV DI, OFFSET REM
    MOV [DI], DX
    MUL NUMBER_10_WORD
    MOV BX, AX
    MOV DX, [DI]
    ADD BX, DX
    MOV AX, [SI]
    CMP AX, 0
    JNE LOOP_REVERSE

    MOV REVERSED_NUMB, BX
    POP DX
    POP BX
    RET


REVERSE ENDP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

بررسی پالیندرومی بودن عدد abc با بررسی برابری آن با عدد cba.

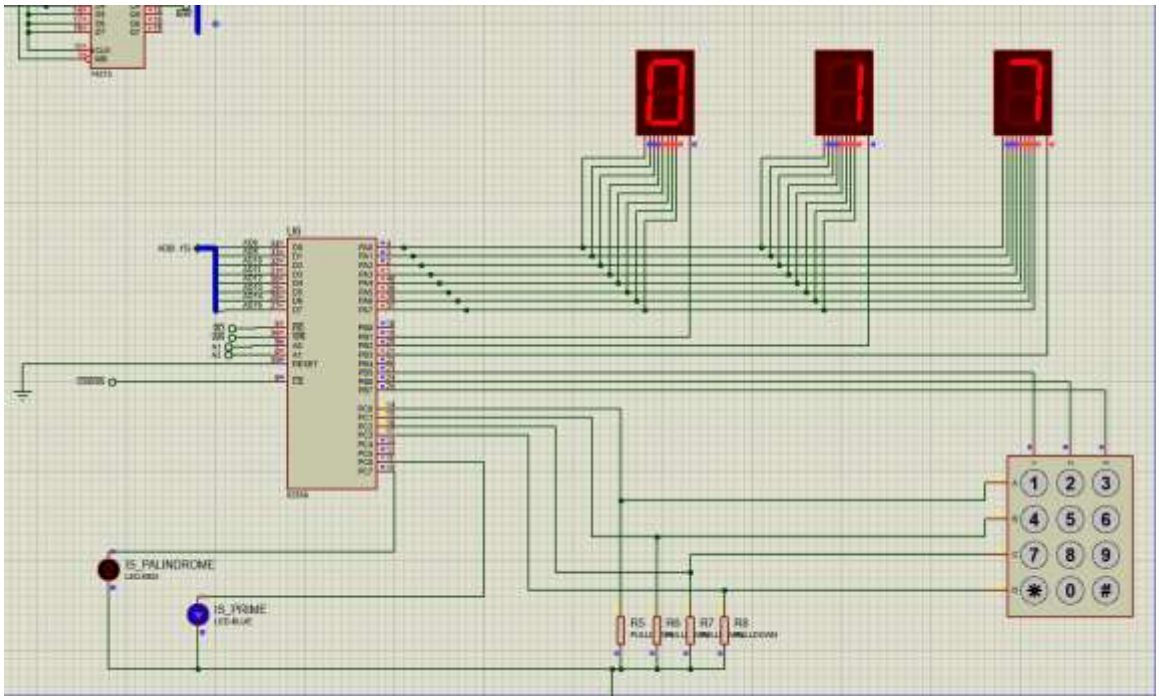
در نهایت با استفاده از فلگ‌های ذخیره شده خروجی روی پورت C را تعیین می‌کنیم تا LED های مربوط به هر بخش روشن یا خاموش شوند.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
CREATE_LED_PORT_NUMBER PROC
      MOV AL, FLAG_IS_PRIME
      OR AL, FLAG_IS_PAL

      OUT 0ADh, AL      ;WRITE TO PORT C UPPER BITS LED OFF/ON
      RET

CREATE_LED_PORT_NUMBER ENDP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

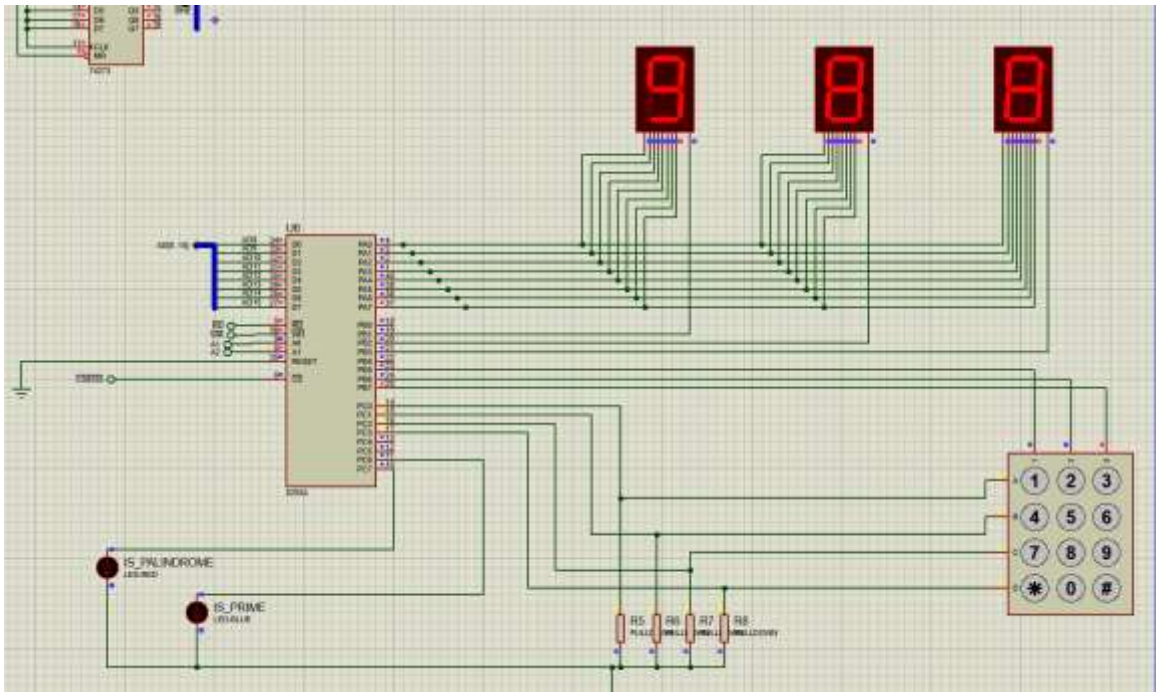
نمونه‌های ورودی و خروجی:



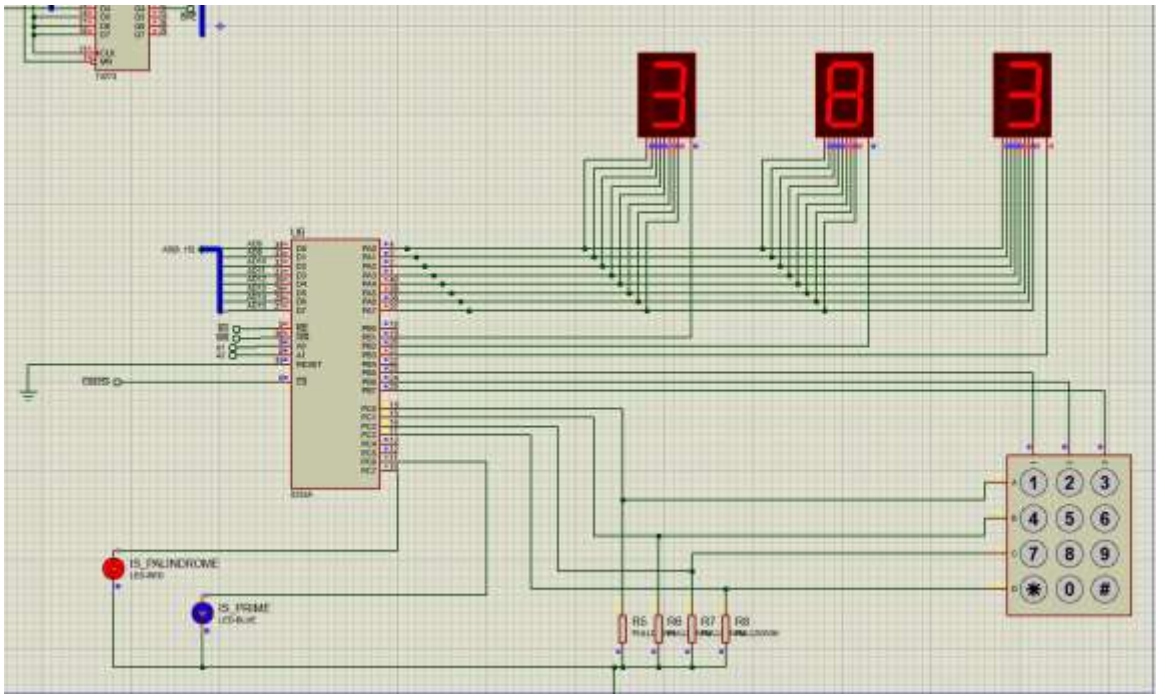
- 333 یک عدد پالیندرومی و غیر اول می‌باشد.



- 17 یک عدد اول و غیر پالیندرومی می‌باشد.



- 988 یک عدد غیر اول و غیر پالیندرومی می‌باشد.



- 383 یک عدد اول و پالیندرومی می‌باشد.