פתרון מבחן במיקרו בקרים ושפה עילית 2014 תשע"ד

שאלה 5

להלן תת שגרה בשפת הסף של המיקרו בקר 8051 .

```
1.      PROC: MOV R0,#20H
2.            MOV R3,#7
3.            MOV R4,#0
4.            MOV R5,#0
5.      NEXT: MOVA,@R0
6.            ANL A,#01
7.            JNZ NB
8.            MOV A,@R0
9.            ADD A,R4
10.           MOV R4,A
11.           SJMP CONT
12.     NB:   MOV  A,@R0
13.           ADD A,R5
14.           MOV  R5,A
15.     CONT: INC R0
16.           DJNZ R3,NEXT
17.           RET
```

א. הסבר את הפקודות שבשורות 5 , 6 , 9 , 16 ו 17 .
ב. הסבר מה מבצעת תת השגרה.
ג. בטבלה שלהלן נתונים התכנים של תאי הזיכרון שכתובתם 20H ÷ 26H ב RAM הפנימי של המיקרו בקר, לפני ביצוע תת השגרה.

כתובת התא	20H	21H	22H	23H	24H	25H	26H
תוכן התא	13H	25H	16H	19H	1AH	08H	0BH

מה יהיו התכנים של R4 ו R5 בסיום ביצוע תת השגרה.

תשובה 5

א.

שורה 5

Next: MOV A,@R0

העבר אל האקומולאטור נתון מזיכרון ה RAM הפנימי מהכתובת עליה מצביע הנתון שב R0 . NEXT היא תווית – LABEL – כתובת.

שורה 6

ANL A,#01

בצע פעולת AND לוגי בין הנתון שבאקומולאטור ובין הנתון 1 (00000001). התוצאה תהיה באקומולאטור. בעזרת פקודה זו מאפסים את 7 הביטים הגבוהים ובודקים מה יש בביט הנמוך – LSB. אם באקומולאטור יש 0 (בביט הנמוך יש 0) מסקנה – המספר זוגי. אם באקומולאטור אין 0 מסקנה – המספר הוא אי זוגי.

שורה 9

ADD A,R4

חבר את הנתון שבאקומולאטור עם הנתון שברגיסטר R4 (בבנק הנוכחי). התוצאה תהיה באקומולאטור והדגלים מושפעים.

שורה 16

DJNZ R3, NEXT

חסר 1 מהנתון שברגיסטר R3 ואם אחרי החיסור הנתון ב R3 לא אפס קפוץ לתווית – כתובת – NEXT. אם הנתון הוא אפס עבור לפקודה הבאה.

שורה 17

RET

RET הוא קיצור של RETurn – חזור מפרוצדורה (או בשמות אחרים - שגרה, רוטינה, סאב רוטינה).
"כתובת החזרה" "נשלפת" מהמחסנית וחוזרים אל הכתובת בתוכנית ממנה יצאנו לפרוצדורה.

ב.

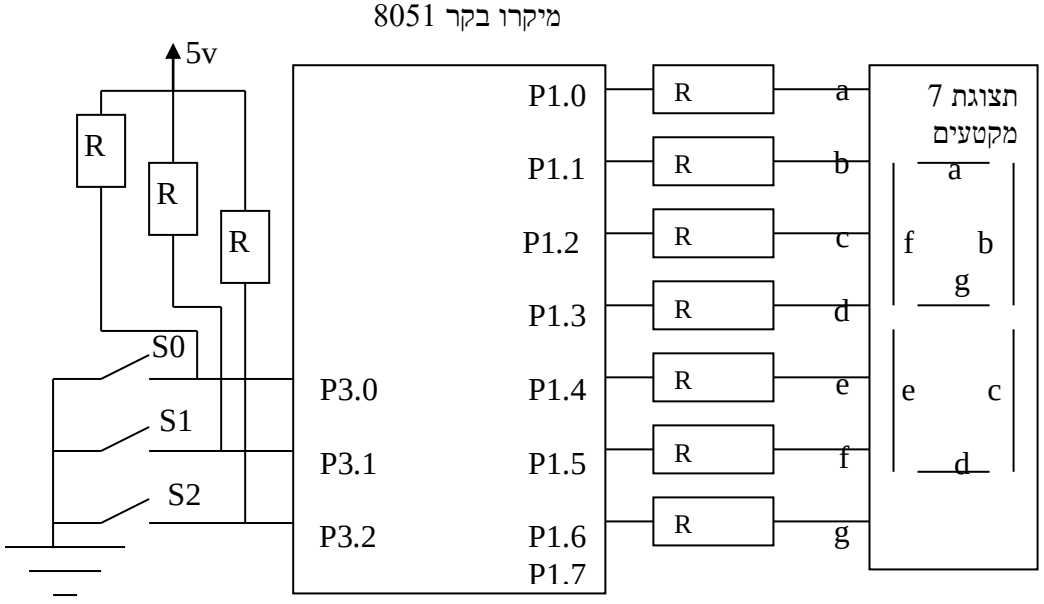
התוכנית פועלת על בלוק כתובות בזיכרון הנתונים הפנימי של המיקרו בקר, החל מכתובת 20H עד כתובת 26H (כולל) , סה"כ 7 כתובות. היא מביאה מכל כתובת את הנתון ובודקת האם הוא זוגי או אי זוגי ובהתאמה היא מחברת את כל הנתונים הזוגיים ואת סכומם היא שומרת ב R4 ואת הנתונים האי זוגיים היא מסכמת ושומרת ב R5 .
רגיסטר R0 הוא רגיסטר המצביע על הכתובות בזיכרון הנתונים הפנימי.
רגיסטר R3 הוא "מונה לולאה" והוא מראה על כמה כתובות בזיכרון יש לעבור בתוכנית.
R4 רגיסטר המכיל את סכום המספרים הזוגיים .
R5 רגיסטר המכיל את סכום המספרים האי זוגיים.

ג.

R4 = 16H + 1AH + 8 = 38H
R5 = 13H + 25H + 19H + 0BH = 5CH

<u>שאלה 6</u>

באיור לשאלה נתונה תצוגת 7 מקטעים מסוג CC (קתודה משותפת) המחוברת להדקים P1.0 ÷ P1.6 . להדקים P3.0 ÷ P3.2 מחוברים המפסקים S0 ÷ S2 .



כתוב תת שגרה בשפת הסף של המיקרו בקר 8051 או תכנית בשפת C שלו , שתבצע את הפעולות :
1. תבדוק כמה מהמפסקים סגורים 2. תציג את מספר המפסקים הסגורים בתצוגת 7 המקטעים.

<u>תשובה 6</u>

נרשום את התוכנית גם בשפת אסמבלי ולאחריה בשפת C51 . נזכור שמפסק פתוח מכניס '1' ומפסק סגור מכניס '0' .
נרשום טבלה שבה נבדוק מה צריך להוציא לפורט 1 כדי להדליק את הספרות 0 , 1 , 2 , ו 3 בתצוגת 7 המקטעים (אין צורך לבדוק את כל הספרות כי מקסימום 3 מפסקים יכולים להיות סגורים).

הספרה	P1.6(g)	P1.5(f)	P1.4(e)	P1.3(d)	P1.2(c)	P1.1(b)	P1.0 (a)	המספר בהקסה
0	0	1	1	1	1	1	1	3F
1	0	0	0	0	1	1	0	6
2	1	0	1	1	0	1	1	5B
3	1	0	0	1	1	1	1	4F

בתוכניות שנרשום נבצע גם את סעיף 1 וגם את סעיף 2 .

התוכנית בשפת C51 . נרשום פונקציה

```
include <8051.h>
code unsigned char seven_segment[] ={ 0x3f,6,0x5b,0x4f}; // הגדרת מערך עבור התצוגה
void counts_closed_sw ( )      //  פונקציה הסופרת את כמות המפסקים הסגורים
{
 unsigned char  counter = 0;
 if(P3_0 = = 0)          //  ( ? המפסק סגור) 0 יש  0 האם בפורט 3 נקודה
   counter++;
if(P3_1 = = 0)          //  ( ? המפסק סגור) 0 יש  1 האם בפורט 3 נקודה
  counter++;
if(P3_2==0)             //  ( ? המפסק סגור) 0 יש  2 האם בפורט 3 נקודה
  counter++;
P1=seven_segment[counter];
// counter במיקום  seven_segment  הוצא לפורט 1 את הנתון שבמערך
}
```

תוכנית נוספת בשפת c51 . היא יותר ארוכה (ופחות יעילה) אבל אולי מובנת יותר

```
include <8051.h>
code unsigned char seven_segment[] ={ 0x3f,6,0x5b,0x4f}; // הגדרת מערך עבור התצוגה
void counts_closed_sw ( )      //  פונקציה הסופרת את כמות המפסקים הסגורים
{
 unsigned char  counter = 0;
 if ( (P3_0 = = 0 ) & & (P3_1 = = 0) && P3_2 = = 0))  //   0  0  0
   counter=3;
else if ( (P3_0 = = 0 ) & & (P3_1 = = 0) &&  P3_2 ))  //    0  0  1
  counter=2;
else if ( (P3_0 = = 0 ) & &  P3_1  &&  (P3_2 = = 0))  //     0  1  0
  counter=2;
else if ( (P3_0 = = 0) & &  P3_1  &&  P3_2 )          //     0  1  1
```

```
  counter = 1;
else if ( P3_0 & &  ( P3_1 = = 0 ) &&  (P3_2 = = 0 ))  //    1  0  0
  counter=2;
else if ( P3_0  & &  ( P3_1 = = 0 )  &&  P3_2  )        //    1  0  1
  counter=1;
else if ( P3_0  & &   P3_1   && P3_2  = = 0 )       //    1  1  0
  counter=1;
else if ( P3_0  & &   P3_1  && P3_2  )       //    1  1  1
  counter = 0 ;
if (counter = = 0 )
        P1 = 0x3f;      // להדלקת הספרה  0  בתצוגת 7 המקטעים
else if ( counter = = 1)
        P1 = 6;         // הדלקת הספרה 1 בתצוגת 7 המקטעים
else if ( counter = = 2)
        P1 =  5 b ;     // הדלקת הספרה 2 בתצוגת 7 המקטעים
else if ( counter = = 1)
        P1 = 4f;        // הדלקת הספרה 3 בתצוגת 7 המקטעים
}
```

התוכנית באסמבלי : נרשום כפרוצדורה (תת שגרה).

```
COUNTS_CLOSED_SW:
        MOV A,#0            ;  אתחול מונה כמות מפסקים סגורים ל 0
        MOV C,P1.0          ;  העבר לדגל הנשא את מצב  p1.0
        JC CONT1            ; cont1 אם בדגל יש  1 – המפסק פתוח - קפוץ ל
        INC A               ;  אם המפסק סגור הגדל את המונה ב 1
CONT1:  MOV C,P1.1
        JC CONT2
        INC A
CONT2:  MOV C,P2.2
        JC CONT3
        INC A
CONT3: CJNE A,#0,CONT4
        MOV P1,#3FH
        SJMP SOF
CONT4: CJNE A,#1,CONT5
        MOV P1,# 6
        SJMP SOF
CONT5: CJNE A,#2,CONT6
        MOV P1,#5BH
        SJMP SOF
CONT6:  CJNE A,#3,SOF
        MOV P1,#4FH
SOF:   RET
```

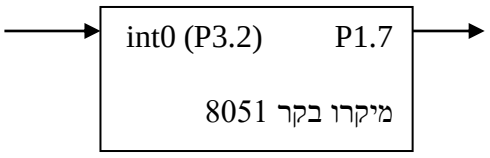
הערה : הנגדים המחוברים לתצוגת 7 המקטעים ואלו המחוברים למפסקים נקראים בסרטוט R . ערכם איננו שווה. ערך הנגדים המחוברים לתצוגת 7 המקטעים הוא כ 220 אוהם וערך הנגדים שמחוברים למפסקים ונקראים Pull Up Resistors הם בסדר גודל של עשרות קילו אוהם.

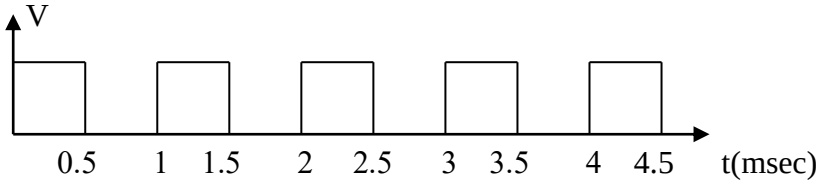
<u>שאלה 7</u>

להלן תוכנית בשפת C של המיקרו בקר 8051

```
1.      #include <8051.h>
2.      unsigned char C;
3.      sbit P1_7 = 0x97 ;
4.      void int0 ( ) interrupt 0
5.      {
6.          C++;
7.   if ( C = = 0xfe )   C=0;
8.   }
9.   void main  ( )
10.  {
11.         C = 0  ;
12.         P1_7 = 1 ;
13.         EX0 = 1 ;
14.         EA = 1 ;
15.         TCON = 0x01 ;
16.          while ( 1 )
17.         {
18.             if ( ( C % 2 = = 0 )    P1_7 = 1 ;
19.             else  P1_7 = 0 ;
20.         }
21.  }
```

באיור הבא מתואר האות שנכנס להדק $\overline{\text{int0}}$ (P3.2) של המיקרו בקר.





א. הסבר את ההוראות שבשורות 2 , 3 , 4 , 16 .

ב. הסבר את משמעות השורות 13 , 14 ו 15 . היעזר בדף הנוסחאות של המיקרו בקר לניתוח מילת הבקרה.

ג. העתק למחברתך את האות בהדק P3.2 וסרטט מתחתיו , בהתאמה , את צורת האות בהדק P1_7 במהלך ביצוע התכנית.

ד. משנים את ההוראה שבשורה 18 להוראה הבאה : if ((C % 3 = 0)) P1_7 = 1 ; .
כיצד ישפיע שינוי זה על אות המוצא בהדק P1_7 ? נמק .

<u>תשובה 7</u>

א.
שורה 2

unsigned char C;

הגדרת משתנה בשם C מטיפוס תווי לא מסומן.
שורה 3

sbit P1_7 = 0x97;

הגדרה של ביט באזור ה SFR (הרגיסטרים המיוחדים) שהכתובת שלו היא 97H והוא נקרא P1_7 .
למעשה ההגדרה הזו מיותרת כי P1_7 מוצהר בקובץ הכותר 8051.h .

שורה 4

void int0 ( ) interrupt 0

הגדרה של פונקציה בשם int0 . הפונקציה איננה מחזירה ערך (לפי המילה void) והיא גם לא מקבלת ערכים (בין הסוגריים הקטנים לא רשום כלום) . המילה interrupt 1 היא מילה שמורה האומרת שהפונקציה היא פסיקה מספר 0 - כלומר פסיקה חיצונית 0 - ויש לרשום את הפונקציה החל מכתובת 3 בזיכרון התוכנית).

שורה 16

while (1)

לולאת while שבה בודקים האם מה שבסוגריים הוא TRUE . היות ובסוגריים רשום 1 וזה תמיד TRUE אז השורות 17 עד 20 מתבצעות בלולאה אין סופית.

ב.
ניעזר ברגיסטר IE הנראה כך :

EA	X	X	ES	ET1	EX1	ET0	EX0

בשורה 14 שמים בביט EA '1' . אם בביט הזה שמים 0 זה חוסם את כל הפסיקות ללא תלות במה שנרשם בהמשך הרגיסטר. היות ושמנו בו '1' אז מאפשרים את כל הפסיקות הרשומות בהמשך הרגיסטר וששמנו בהן '1' .
שורה 13 EX0=1; אומרת לשים 1 בביט EX0 ברגיסטר IE . ביט זה מאפשר פסיקה חיצונית 0 שנכנסת בהדק P3.2 של המיקרו בקר. אם היינו שמים בביט הזה 0 לא היינו מאפשרים את הפסיקה הזו.

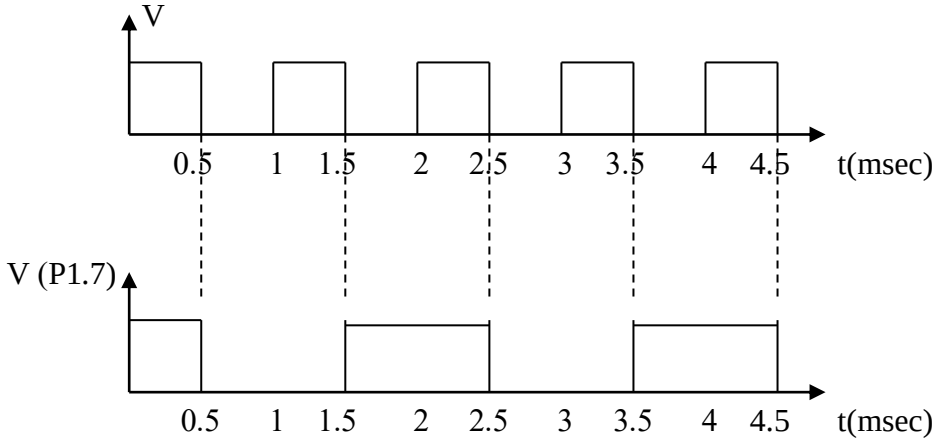
בשורה 15 רשום TCON=0x01 . נראה את רגיסטר ה TCON :

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
0	0	0	0	0	0	0	1

ביט IT0 '1' . ביט זה מופעל על ידי המשתמש בתוכנה ובעזרתו המשתמש קובע האם לעבוד עם פסיקה חיצונית 0 על ירידות – EDGE (רק כאשר יש ירידה מ 1 ל 0) ולא על רמה – LEVEL .
IE0=0 – איפוס ביט שמראה שהייתה בקשת פסיקה חיצונית 0 .
IT0 ו IE0 - ההסבר כמו בשתי הסיביות הקודמות אבל לגבי פסיקה חיצונית 1 .
TR0=0 – לא נותנים לטיימר 0 לרוץ, כלומר לספור.
TF0=0 - מאפסים את הביט שמראה שהייתה גלישה של טיימר 0 (ובקשת פסיקה שלו).
TR1 ו TF1 - הסבר דומה לשתי הסיביות הקודמות אבל לגבי טיימר 1 .

ג.

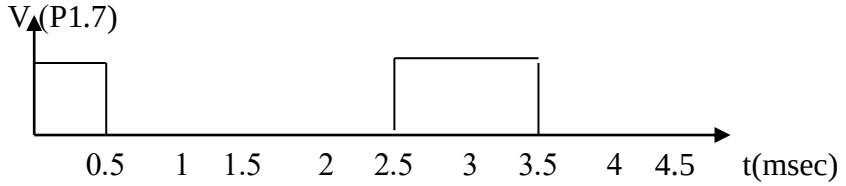
השורות 18 ו 19 בודקות האם המספר במשתנה C הוא זוגי או אי זוגי ובהתאמה מוציאות ל p1.7 1 כאשר המספר זוגי או 0 כאשר המספר איננו זוגי.
כל ירידה של האות בכניסת הפסיקה החיצונית 0 גורם לעבור לכתובת הפסיקה שם מעלים את C ב 1 .
בלולאת ה while בודקים האם c%2 = = 0 , כלומר האם המספר במשתנה C זוגי ואם כן מוציאים מוציאים לפורט 1 נקודה 7 '1' . אם המספר אי זוגי מוציאים להדק הזה '0' . בתחילת התוכנית ב C יש 0 ואז יוצא ב P1.7 = 1 . עם הירידה הראשונה יעלה C ל 1 ואז הוא יהיה אי זוגי ולכן p1.7=0 .
בירידה הבאה C=2 ואז P1.7=1 וכך הלאה.



ד.

הפקודה c%3 = = 0 בודקת האם המספר במשתנה C מתחלק ב 3 ללא שארית . במקרה שכן יצא ל p1.7 '1' .
במקרה שלנו 2 ירידות מ 1 ל 0 של האות בהדק הפסיקה יגרמו ל P1.7 = 0 וכל ירידה שלישית יהיה P1_7=1; . צורת הגל שנקבל :

1. בהתחלת התוכנית C=0 ו C % 3 = 0 ולכן P1_7=1; .
2. ב t=0.5msec יש ירידה בהדק הפסיקה ומקדמים את C ב 1 ויהיה בו 1 . מכאן C % 3 = 1 ולא 0 ולכן P1_7 = 0 .
3. ב t=1.5 יש ירידה נוספת בהדק הפסיקה שוב נגדיל את C ואז C = 2 . C%2=2 ולא 0 ולכן שוב P1_7=0 .
4. בירידה הבאה בהדק הפסיקה t=2.5msec C יגדל ויהיה C=3 ואז C%3=0 ומכאן P1_7=1
5. בירידה הבאה ב t=3.5msec מגדילים את C ב 1 ומקבלים C=4 וחוזרים לסעיף 2 והתהליך חוזר על עצמו.



שאלה 8

נתון בלוק נתונים המתחיל בכתובת 50H בזיכרון הפנימי של המיקרו בקר 8051 וגודלו עשרים בתים.
בלוק הנתונים מכיל ערכים שלמים ממינוס 9 ועד פלוס 9 .
כתוב תת שגרה בשפת הסף של המיקרו בקר 8051 או תכנית בשפת C שלו שתבצע את הפעולות :
1. תאפס את תאי הזיכרון הפנימי שתוכנם הוא מספר שלילי. 2. תסכם את הנתונים החיוביים בבלוק ותציב את הסכום בתא שכתובתו 4fH בזיכרון הפנימי.

פתרון שאלה 8

נפתור גם עם שפת C51 ולאחריה שפת אסמבלי ונכניס לפתרון גם את סעיף 1 וגם את סעיף 2

```
#include <8051.h>
void zero_negatives_sum_positives ( )          // שם הפונקציה
{
data at 0x50 char block[20]  ;
// הגדרה של בלוק בן 20 כתובות מטיפוס תווי (שים לב שהגדרנו את המשתנה כתווי ולא כתווי לא
// מסומן) כדי שיוכל להכיל גם מספרים שליליים.
data at 0x4f unsigned char sum=0;    // 4fH התוצאה תהיה בתא
unsigned char x;                     // אינדקס במערך
for (x=0 ; x<20 ; x++}
 {
  if( block[x] < 0 )
        block[x]=0;
 else
        sum=sum+block[x];
 }
}
```

נרשום גם פונקציה באסמבלי :

```
ZERO_NEGATIVES_SUM_POSITIVES :
        MOV R0,#50H         ; מצביע על הכתובות בזיכרון
        MOV R7,#0           ; מכיל את סכום המספרים החיוביים
        MOV R6,#20          ; מונה לולאה. מראה כמה בתים יש לבדוק
AGAIN:  MOV A,@r0
          JNB ACC.7,NEXT ; NEXT קפוץ ל  אם ביט הסימן שווה 0 כלומר המספר חיובי
          MOV @R0,#0      ; איפוס הנתון בכתובת כי המספר היה שלילי
          SJMP CONT
NEXT:   ADD A,R5   ;חיבור הנתון החיובי שבאקומולאטור עם הרגיסטר שמסכם את כל החיוביים
         MOV R5,A
         INC  R5
         DJNZ R6,AGAIN
         RET
```

הערה : במקום השורה מתחת לתווית ה again הבודקת האם המספר חיובי ואז קופצת ל NEXT או ממשיכה לשורה הבאה כי המספר שלילי ניתן היה גם לרשום בדרך ארוכה יותר :

```
        RLC A
        JNC NEXT
```

ואז בתווית NEXT היינו צריכים להכניס את הפקודה MOV A,@R0 כדי להחזיר את הנתון לאקומולאטור כי הנתון השתנה בפקודת הסיבוב ורק אז לבצע את הפקודה ADD A,R5 .

הערה : נראה לי שהנתון שבשאלה לגבי גודל הערכים החיוביים והשליליים מ 9 עד מינוס 9 מיותר .