

## אלקטרוניקה ומחשבים ג'

שתי יחידות לימוד (השלמה לחמש יחידות לימוד)  
(כיתה י"א)

### הוראות לנבחן

א. משך הבחינה: שלוש שעות.

ב. מבנה השאלון ומפתח ההערכה: בשאלון זה תשע שאלות בשני פרקים. יש לענות על חמש שאלות, שאלה אחת לפחות מכל פרק. לכל שאלה – 20 נקודות. סך הכול – 100 נקודות.

ג. חומר עזר מותר לשימוש: מחשבון.

ד. הוראות מיוחדות:

- ענה על מספר השאלות הנדרש בשאלון. המעריך יקרא ויעריך את מספר התשובות הנדרש בלבד, לפי סדר כתיבתן במחברתך, ולא יתייחס לתשובות נוספות.
- התחל כל תשובה לשאלה בעמוד חדש.
- רשום את כל תשובותיך אך ורק בעט.
- הקפד לנסח את תשובותיך כהלכה, ולסרטט את תרשימיך בהירות.
- כתוב את תשובותיך בכתב-יד ברור, כדי לאפשר הערכה נאותה שלהן.
- אם לדעתך חסרים נתונים הדרושים לפתרון שאלה, אתה רשאי להוסיף אותם, אך עליך להסביר מדוע הוספת אותם.
- בכתיבת פתרונות חישוביים, קבלת מֶרֶב הנקודות מותנית בהשלמת כל המהלכים שלהלן, בסדר שהם רשומים בו:
  - \* רישום הנוסחה המתאימה.
  - \* הצבה של כל הערכים ביחידות המתאימות.
  - \* חישוב (אפשר באמצעות מחשבון).
  - \* רישום התוצאה המתקבלת, ולצדה יחידות המידה המתאימות.
  - \* ליווי הפתרון החישובי בהסבר קצר.

בשאלון זה 12 עמודים ו-32 עמודי נוסחאון.

ההנחיות בשאלון זה מנוסחות בלשון זכר,  
אך מכוונות הן לנבחנות והן לנבחנים.

## השאלות

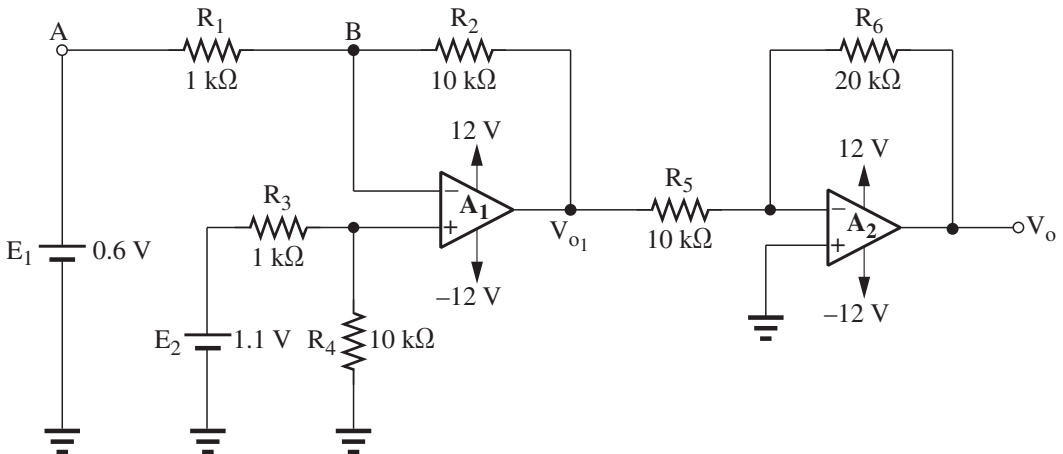
בשאלון זה שני פרקים ובהם תשע שאלות. יש לענות על חמש שאלות בלבד, שאלה אחת לפחות מכל פרק.

### פרק ראשון: מבוא להנדסת אלקטרוניקה

ענה על שאלה אחת לפחות מבין השאלות 1-5 (לכל שאלה – 20 נקודות).

#### שאלה 1

באיור לשאלה 1 מתואר מעגל חשמלי, הכולל מגברי שרת אידאליים.



#### איור לשאלה 1

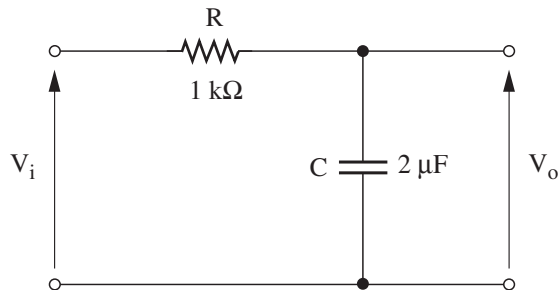
א. חשב את הזרם בנגד  $R_1$  וקבע את כיוונו (מ- $A^-$  ל- $B^-$  או מ- $B^-$  ל- $A^-$ ).

ב. חשב את המתח במוצא מגבר השרת הראשון,  $V_{o1}$ .

ג. חשב את מתח המוצא  $V_o$ .

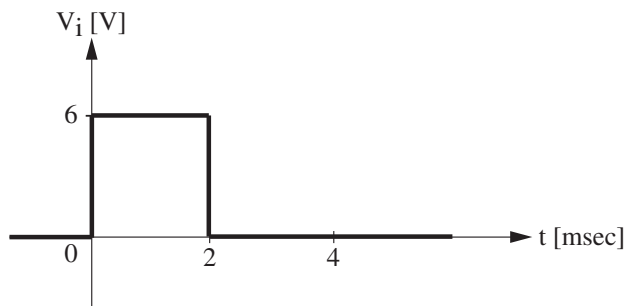
**שאלה 2**

באיור א' לשאלה 2 מתוארת רשת חשמלית.



**איור א' לשאלה 2**

למבוא הרשת מספקים את הדופק המתואר באיור ב' לשאלה.



**איור ב' לשאלה 2**

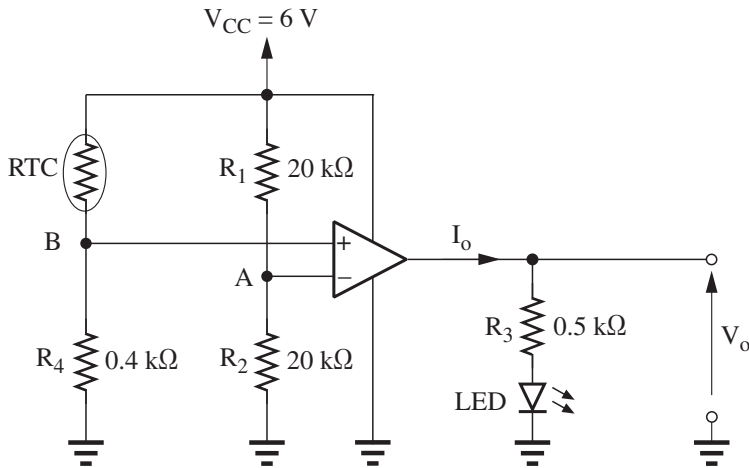
**א.** העתק למחברתך את מתח המבוא ( $V_i$ ), וסרטט מתחתיו, בהתאמה, את מתח המוצא ( $V_o$ ), כפונקציה של הזמן.

**ב.** חשב את מתח המוצא:

1. בזמן  $t = 1 \text{ msec}$  .2. בזמן  $t = 4 \text{ msec}$

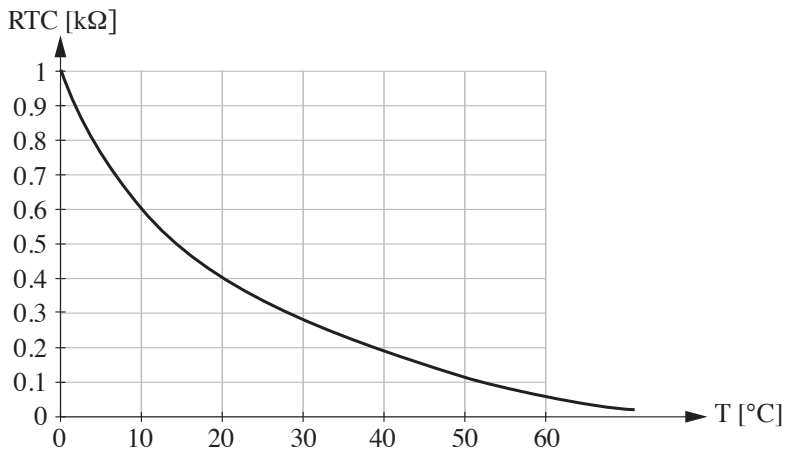
**שאלה 3**

באיור א' לשאלה 3 נתון מעגל חשמלי. מגבר השרת שבמעגל - אידיאלי. חיישן הטמפרטורה (RTC) שבמעגל הוא נגד שערכו תלוי בטמפרטורת הסביבה. נתון:  $V_{LED} = 1.5 \text{ V}$ .



**איור א' לשאלה 3**

- א. חשב את המתח בנקודה A, ורשום את תחום ערכי המתח בנקודה B שעבורו נורית ה-LED דולקת.
- ב. באיור ב' לשאלה נתון האופייין של חיישן הטמפרטורה (RTC), המתאר את התנגדותו של החיישן כפונקציה של טמפרטורת הסביבה.



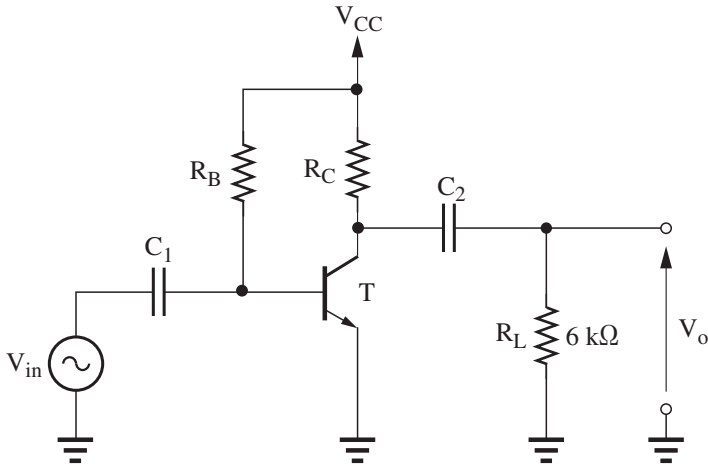
### איור ב' לשאלה 3

מגדילים את טמפרטורת הסביבה של חיישן הטמפרטורה (RTC) מ- $0^{\circ}C$ .  
החל מאיזו טמפרטורה ה-LED במוצא המעגל ידלק?

ג. חשב את הזרם במוצא מגבר השרת ( $I_0$ ) כאשר נורית ה-LED דולקת.

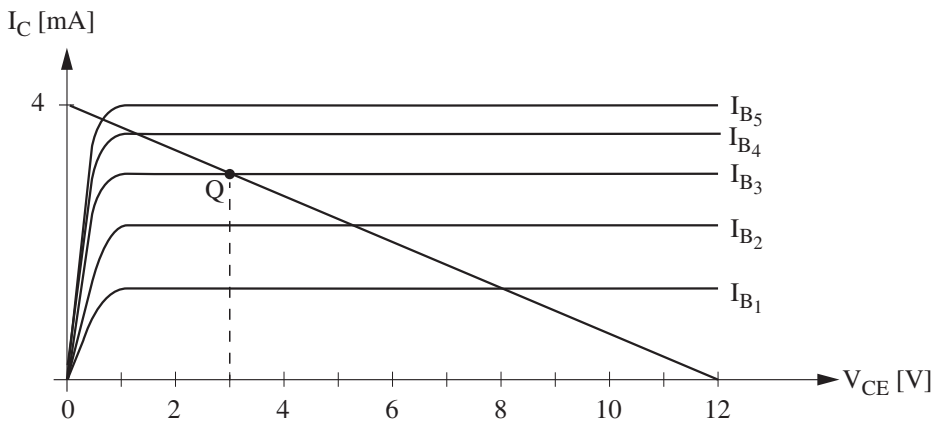
**שאלה 4**

באיור א' לשאלה 4 נתון המעגל החשמלי של מגבר טרנזיסטורי. היגבי הקבלים במעגל - זניחים. נתוני הטרנזיסטור T הם:  $\beta = h_{fe} = 100$ ,  $h_{ie} = 1 \text{ k}\Omega$ ,  $V_{BE} = 0.7 \text{ V}$ .



**איור א' לשאלה 4**

באיור ב' לשאלה נתון גרף המתאר את אופייני המוצא של הטרנזיסטור. על הגרף מסורטט קו העבודה ועליו מסומנת נקודת העבודה, Q, של הטרנזיסטור.

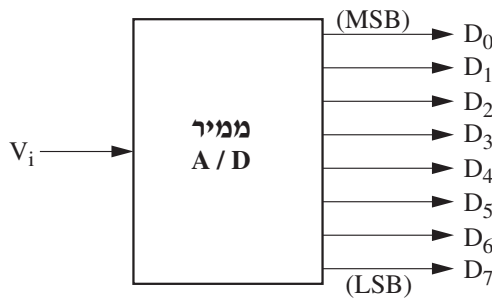


**איור ב' לשאלה 4**

- א. היעזר בגרף, מצא את ערך המתח  $V_{CC}$ , וחשב את התנגדות הנגד  $R_C$ .
- ב. חשב את הזרמים  $I_C$  ו- $I_B$  בנקודת העבודה Q.
- ג. חשב את התנגדות הנגד  $R_B$ .
- ד. חשב את הגבר המתח של המעגל,  $A_V = \frac{V_o}{V_{in}}$ .

## שאלה 5

באיור לשאלה 5 נתון תרשים של ממיר אות אנלוגי לאות ספרתי (A/D) בעל שמונה סיביות.



### איור לשאלה 5

כל הסיביות נמצאות במצב '0' כאשר מתח־המבוא הוא  $V_i = 0\text{ V}$ . כאשר מתח־המבוא הוא  $V_i = 2\text{ V}$ , מתקבלת במוצא הממיר המילה הבינארית  $11001000_{(2)}$ .

- א. מהו כושר ההבחנה (הרזולוציה) של הממיר ב- $\text{mV}$ ?
- ב. מצא את המילה הבינארית המתקבלת במוצא הממיר עבור מתח־המבוא  $V_i = 0.95\text{ V}$ .
- ג. במוצא הממיר מתקבלת המילה הבינארית  $10000001_{(2)}$ . מהו מתח־המבוא המתאים למילה הבינארית הזו?

## פרק שני: מבוא להנדסת מחשבים

ענה על שאלה אחת לפחות מבין השאלות 6-9 (לכל שאלה – 20 נקודות).

### שאלה 6

להלן תת־שגרה הכתובה בשפת־הסף של המיקרו־מעבד 8086/88.

```
1.   MYSUB:   MOV SI, 200H
2.           MOV CX, 5H
3.   NEXT:    MOV AL, [SI]
4.           ROR AL, 1
5.           MOV [SI+5], AL
6.           INC SI
7.           DEC CX
8.           JNZ NEXT
9.           RET
```

בטבלה שלהלן נתונים תוכני התאים  $209H \div 200H$  לפני ביצוע התת־שגרה.

209H	208H	207H	206H	205H	204H	203H	202H	201H	200H	כתובת התא
00H	80H	40H	20H	10H	08H	04H	02H	01H	00H	תוכן התא

א. הסבר את ההוראות שבשורות 3, 4, 5 ו־8.

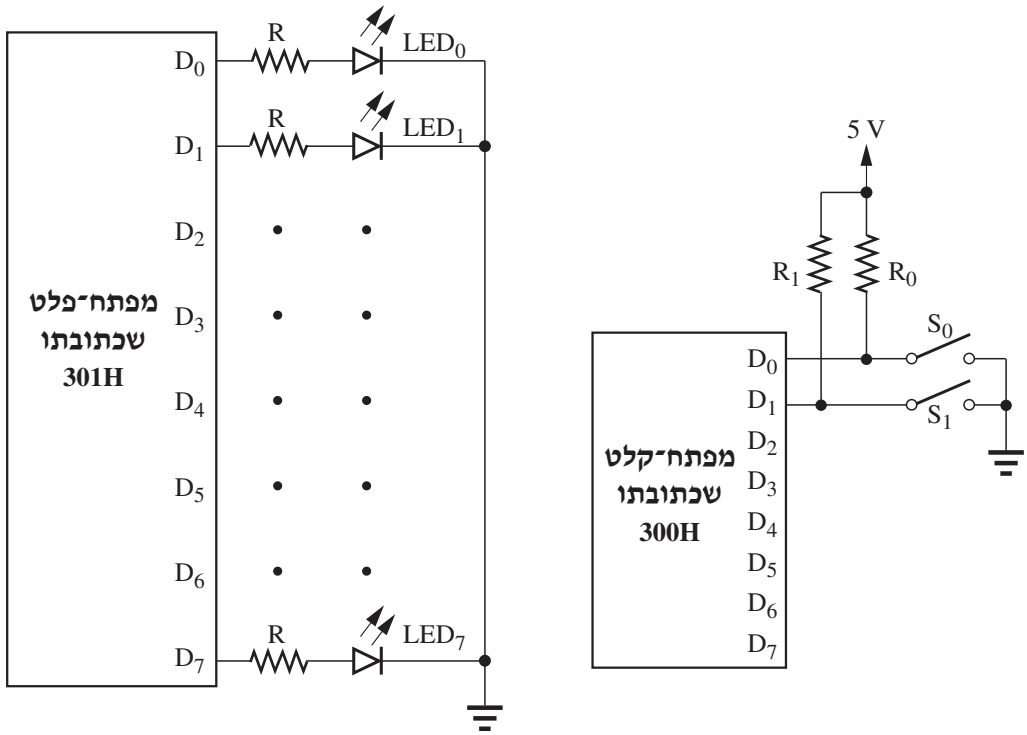
ב. רשום את תוכני התאים  $209H \div 200H$  לאחר ביצוע התת־שגרה. נמק את תשובתך או היעזר בטבלת־מעקב אחר האוגרים SI, CX ו־AL.

ג. קבע ערכים לתוכני התאים שכתובותיהם  $204H \div 200H$ , כך שהתוכן של כל אחד מהתאים שכתובותיהם  $208H \div 205H$  לאחר ביצוע תת־השגרה יהיה 01H, והתוכן של התא שכתובתו 209H יהיה 55H.



שאלה 7

באיור לשאלה 7 נתון תרשים של מפתח-פלט שכתובתו 301H ומפתח-קלט שכתובתו 300H .



איור לשאלה 7

כתוב תכנית בשפת-הסף של המיקרו-מעבד 8086/88, שתבצע את ההוראות הבאות בלולאה אינסופית:

1. תקלוט את מצב שני המפסקים המחוברים למפתח-הקלט.
2. אם המפסק  $S_0$  סגור והמפסק  $S_1$  פתוח - כל הנוריות  $LED_0 \div LED_7$  ידלקו.
3. אם המפסק  $S_0$  פתוח והמפסק  $S_1$  סגור - כל הנוריות  $LED_0 \div LED_7$  תהיינה כבויות.
4. במצבים האחרים של המתגים - לא יהיה שינוי במצב הנוריות.

## שאלה 8

להלן שני קטעי תכנות: קטע מתכנית בשפת Visual Basic וקטע מתכנית בשפת C.

### קטע מתכנית בשפת Visual Basic :

```
1. Dim arr(20) As Integer =  
   {6,-2,5,0,-1,0,0,-3,-4,7,8,9,4,6,3,-8,2,0,-1,-7}  
2. Dim i As Integer, n As Integer, p As Integer, z As Integer  
3. n = 0  
4. p = 0  
5. z = 0  
6. For i = 0 To 19  
7.     Console.WriteLine("{0}", arr(i))  
8. Next  
9. For i = 0 To 19  
10.    If arr(i) > 0 Then  
11.        p = p + 1  
12.    Else if arr(i) = 0 Then  
13.        z = z + 1  
14.    Else if arr(i) < 0 Then  
15.        n = n + 1  
16.    End If  
17. Next  
18. Console.WriteLine("p={0} n={1} z={2}", p, n, z)
```

קטע מתכנית בשפת C :

```
1. #include <stdio.h>
2. void main(void)
3. {
4.     int arr[20]=
       {6,-2,5,0,-1,0,0,-3,-4,7,8,9,4,6,3,-8,2,0,-1,-7};
5.     int i, n = 0, p = 0, z = 0 ;
6.     for (i = 0 ; i < 20 ; i++)
7.         printf("%d,", arr[i]);
8.     for (i = 0 ; i < 20 ; i++)
9.         {
10.            if (arr[i]>0) p++ ;
11.            else if (arr[i]==0) z++ ;
12.            else if (arr[i]<0) n++ ;
13.        }
14. printf("\n p = %d  n = %d  z = %d", p, n, z);
15. }
```

בחר בקטע התכנית בשפת C או בקטע התכנית בשפת Visual Basic . ציין את בחירתך בתחילת תשובתך, וענה על הסעיפים שלהלן:

**א.** הסבר את ההוראות בהתאם לקטע התכנית שבחרת:

- בקטע התכנית בשפת C : 4, 6, 7 ו-10 .

- בקטע התכנית בשפת Visual Basic : 1, 6, 7 ו-10 .

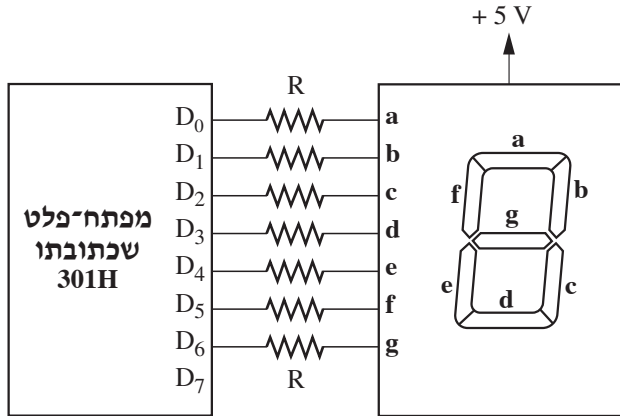
**ב.** מה יופיע על צג המחשב לאחר ביצוע השורות 1 ÷ 7 בקטע התכנית בשפת C או השורות 1 ÷ 8 בקטע התכנית בשפת Visual Basic ?

**ג.** מה יופיע על צג המחשב לאחר ביצוע קטע התכנית כולו?

**ד.** הוסף קטע-קוד שישכם את חמשת האיברים האחרונים במערך, ויצגי את סכומם על צג המחשב.

## שאלה 9

באיור לשאלה 9 נתון תרשים של מפתח־פלט שכתובתו 301H, המחובר לתצוגת שבעה מקטעים (7-seg) מסוג אנודה משותפת (CA).



תצוגת שבעה מקטעים  
מסוג CA

### איור לשאלה 9

כתוב תכנית בשפת C או בשפת Visual Basic שתבצע את הפעולות שלהלן:

1. תגדיר את המערך שלהלן, הכולל עשרה איברים מטיפוס שלם:

3 25 14 7 46 78 15 34 26 54

2. תחלק כל אחד מהאיברים ב־2, ותבדוק את השארית: אם השארית היא 0 - תוצג הספרה 0

בתצוגת ה־7-seg. אם השארית היא 1 - תוצג הספרה 1 בתצוגת שבעת המקטעים.

התכנית תבצע השהיה של שנייה אחת בין כל שתי בדיקות סמוכות.

### בהצלחה!

זכות היוצרים שמורה למדינת ישראל.  
אין להעתיק או לפרסם אלא ברשות משרד החינוך.

אין להעביר את הנוסחאון  
לנבחן אחר

## נוסחאון באלקטרוניקה ומחשבים

(32 עמודים)

### 1. נוסחאות באלקטרוניקה תקבילית

#### חישובי הגבר

הגבר מתח -  $A_V$   
מתח מוצא -  $V_o$  [V]  
מתח מבוא -  $V_i$  [V]

$$A_V = \frac{V_o}{V_i}$$

הגבר מתח בדציבלים -  $A_V$  [dB]

$$A_V = 20 \log \frac{V_o}{V_i}$$

הגבר זרם -  $A_I$   
זרם מוצא -  $I_o$  [A]  
זרם מבוא -  $I_i$  [A]

$$A_I = \frac{I_o}{I_i}$$

הגבר זרם בדציבלים -  $A_I$  [dB]

$$A_I = 20 \log \frac{I_o}{I_i}$$

הגבר הספק -  $A_P$   
הספק מוצא -  $P_o$  [W]  
הספק מבוא -  $P_i$  [W]  
התנגדות נגד העומס -  $R_L$  [ $\Omega$ ]  
התנגדות מבוא -  $R_i$  [ $\Omega$ ]

$$A_P = \frac{P_o}{P_i} = A_V \cdot A_I = A_V^2 \cdot \frac{R_L}{R_i} = A_V^2 \cdot \frac{R_i}{R_L}$$

הגבר הספק בדציבלים -  $A_P$  [dB]

$$A_P = 10 \log \frac{P_o}{P_i}$$

הגבר כולל של N דרגות  
המחוברות בשרשרת (קסקדה) -  $A_{VT}$

$$A_{VT} = A_{V1} \cdot A_{V2} \cdot A_{V3} \cdot \dots \cdot A_{VN}$$

$$A_{VT} [\text{dB}] = A_{V1} [\text{dB}] + A_{V2} [\text{dB}] + A_{V3} [\text{dB}] + \dots + A_{VN} [\text{dB}]$$

הגבר כולל בדציבלים של  
N דרגות המחוברות בשרשרת  
(קסקדה) -  $A_{VT} [\text{dB}]$

### מאזן הספקים

הספק מבוא -  $P_I$  [W]

הספק נצרך מהספקים -  $P_{CC}$  [W]

הספק העומס -  $P_L$  [W]

הספק מבוזבז -  $P_{diss}$  [W]

$$P_I + P_{CC} = P_L + P_{diss}$$

### משוב שלילי

הגבר עם משוב (בחוג סגור) -  $A_f$  [A]

הגבר ללא משוב -  $A$   
(הגבר חוג פתוח)

מקדם משוב -  $\beta$

$$A_f = \frac{A}{1 + \beta A}$$

### משוב מתח טורי

התנגדות מבוא עם משוב -  $R_{if}$  [ $\Omega$ ]

התנגדות מבוא ללא משוב -  $R_i$  [ $\Omega$ ]

התנגדות מוצא עם משוב -  $R_{of}$  [ $\Omega$ ]

התנגדות מוצא ללא משוב -  $R_o$  [ $\Omega$ ]

$$R_{if} = R_i (1 + \beta A)$$

$$R_{of} = \frac{R_o}{1 + \beta A}$$

טרנזיסטור דו-נושאי (בתחום הפעיל)

בהזנחת זרם הזליגה  $I_{CO}$  :

$$I_C = \beta I_B, \quad I_E = (\beta + 1) I_B, \quad I_E = I_C + I_B$$

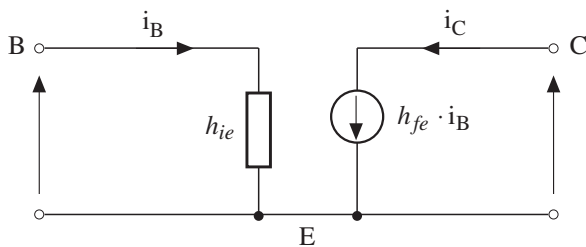
זרם הקולט -  $I_C$  [A]

זרם הפולט -  $I_E$  [A]

זרם הבסיס -  $I_B$  [A]

$$\alpha = \frac{I_C}{I_E} = \frac{\beta}{\beta + 1}, \quad \beta = \frac{\alpha}{1 - \alpha}$$

תרשים תמורה מקורב מסוג h של טרנזיסטור דו-נושאי



טרנזיסטור בחיבור פולט (אמיטר) משותף

	ללא נגד $R_E$	עם נגד $R_E$
$A_I$	$h_{fe}$	$h_{fe}$
$R_i$	$h_{ie}$	$h_{ie} + (1 + h_{fe}) \cdot R_E$
$A_V$	$-\frac{h_{fe} \cdot R_L}{h_{ie}}$	$-\frac{h_{fe} \cdot R_L}{R_i}$
$R_o$	$\infty$	$\infty$

מגברי שרת

מגבר מהפך

$$A_V = -\frac{R_f}{R_1}$$

- נגד המשוב -  $R_f$  [ $\Omega$ ]  
הנגד המחובר לכניסה המהפכת -  $R_1$  [ $\Omega$ ]

מגבר עוקב

$$A_V = 1 + \frac{R_f}{R_1}$$

- נגד המשוב -  $R_f$  [ $\Omega$ ]  
הנגד היוצא מהכניסה המהפכת לאדמה -  $R_1$  [ $\Omega$ ]

2. נוסחאות באלקטרוניקה ספרתית

- הערך הרגעי של המתח -  $v(t)$  [V]  $v(t) = V_\infty - (V_\infty - V_{0+})e^{-\frac{t}{\tau}}$   
הערך שאליו המתח שואף להגיע כאשר  $t \rightarrow \infty$  -  $V_\infty$  [V]  
הערך ההתחלתי של המתח -  $V_{0+}$  [V]  $t = -\tau \cdot \ln\left(\frac{V_\infty - v(t)}{V_\infty - V_{0+}}\right)$

- קבוע זמן -  $\tau$  [sec]  $\tau = RC$   
התנגדות -  $R$  [ $\Omega$ ]  
קיבול -  $C$  [F]

- תדר חצי הספק עליון של רשת מעבירת נמוכים -  $f_H$  [Hz]  $f_H = \frac{1}{2\pi\tau}$

- תדר חצי הספק תחתון של רשת מעבירת גבוהים -  $f_L$  [Hz]  $f_L = \frac{1}{2\pi\tau}$

- זמן עלייה של רשת מעבירת נמוכים -  $t_r$  [sec]  $t_r = 2.2\tau$



**3. אוסף פקודות למיקרו-מעבדים 8086/88**

**מקרא לאוגר הדגלים:**

- X – מושפע מהפעולה (Modified)
- U – לא מוגדר אחרי הפעולה (Undefined)
- R – מוחזר מהמחסנית (Returned)
- 0 – מתאפס
- 1 – מקבל '1'

ADC	ADC destination, source Add with carry			Flags	O D I T S Z A P C
				X	X X X X X
Operands	Clocks	Transfers*	Bytes	Coding Example	
register, register	3(3)	—	2	ADC AX, SI	
register, memory	9(10)+EA	1	2-4	ADC CX, BETA [SI]	
memory, register	16(10)+EA	2	2-4	ADC ALPHA [BX] [SI], DI	
register, immediate	4(4)	—	3-4	ADC BX, 256	
memory, immediate	17(16)+EA	2	3-6	ADC GAMMA, 30H	
accumulator, immediate	4(3-4)	—	2-3	ADC AL, 5	

ADD	ADD destination, source Addition			Flags	O D I T S Z A P C
				X	X X X X X
Operands	Clocks	Transfers*	Bytes	Coding Example	
register, register	3(3)	—	2	ADD CX, DX	
register, memory	9(10)+EA	1	2-4	ADD DI, [BX].ALPHA	
memory, register	16(10)+EA	2	2-4	ADD TEMP, CL	
register, immediate	4(4)	—	3-4	ADD CL, 2	
memory, immediate	17(16)+EA	2	3-6	ADD ALPHA, 2	
accumulator, immediate	4(3-4)	—	2-3	ADD AX, 200	

AND	AND destination, source Logical and			Flags	O D I T S Z A P C
				X	X X U X X
Operands	Clocks	Transfers*	Bytes	Coding Example	
register, register	3(3)	—	2	AND AL, BL	
register, memory	9(10)+EA	1	2-4	AND CX, FLAG_WORD	
memory, register	16(10)+EA	2	2-4	AND ASCII [DI], AL	
register, immediate	4(4)	—	3-4	AND CX, 0F0H	
memory, immediate	17(16)+EA	2	3-6	AND BETA, 01H	
accumulator, immediate	4(3-4)	—	2-3	AND AX, 01010000B	

<b>CALL</b>	CALL target Call a procedure			Flags O D I T S Z A P C
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
near-proc	19(14)	1	3	CALL NEAR__PROC
far-proc	28(23)	2	5	CALL FAR__PROC
memptr 16	21(19)+EA	2	2-4	CALL PROC_TABLE [SI]
regptr 16	16(13)	1	2	CALL AX
memptr 32	37(38)+EA	4	2-4	CALL [BX], TASK [SI]
<b>CLC</b>	CLC (no operands) Clear carry flag			Flags O D I T S Z A P C 0
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
(no operands)	2(2)	—	1	CLC
<b>CLI</b>	CLI (no operands) Clear interrupt flag			Flags O D I T S Z A P C 0
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
(no operands)	2(2)	—	1	CLI
<b>CMP</b>	CMP destination, source Compare destination to source			Flags O D I T S Z A P C X X X X X X X
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
register, register	3(3)	—	2	CMP BX, CX
register, memory	9(10)+EA	1	2-4	CMP DH, [ALPHA]
memory, register	9(10)+EA	1	2-4	CMP [BP+2], SI
register, immediate	4(3)+EA	—	3-4	CMP BL, 02H
memory, immediate	10(10)+EA	1	3-6	CMP RADAR [BX], [DI],3420H
accumulator, immediate	4(3-4)	—	2-3	CMP AL, 00010000B

<b>CMPS</b>	CMPS des-string, source-string Compare string			Flags O D I T S Z A P C X           X X X X X
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
dest-string,	22(22)	2	1	CMPS BUFF1, BUFF2
source-string (repeat)				
dest-string, source-string	9+22/rep (5+22/rep)	2/rep	1	REPE CMPS ID, KEY

<b>DAA</b>	DAA (no operands) Decimal adjust for addition			Flags O D I T S Z A P C U           X X X X X
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
(no operands)	4(4)	—	1	DAA

<b>DAS</b>	DAS (no operands) Decimal adjust for subtraction			Flags O D I T S Z A P C U           X X X X X
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
(no operands)	4(4)	—	1	DAS

<b>DEC</b>	DEC destination Decrement by 1			Flags O D I T S Z A P C X           X X X X
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
reg 16	3(3)	—	1	DEC AX
reg 8	3(3)	—	2	DEC AL
memory	15(15)+EA	2	2-4	DEC ARRAY [SI]

<b>DIV</b>	DIV source Division, unsigned			Flags O D I T S Z A P C U U U U U
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
reg 8	80-90(29)	—	2	DIV CL
reg 16	144-162(38)	—	2	DIV BX
mem 8	86-96+EA (35)	1	2-4	DIV [ALPHA]
mem 16	150-168+ EA(94)	1	2-4	DIV TABLE [SI] / DIV [TABLE + SI]

<b>IN</b>	IN accumulator, port Input byte or word			Flags O D I T S Z A P C
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
accumulator, immed 8	10(10)	1	2	IN AL, 0FEH / IN AX, 0FEH
accumulator, DX	8(8)	1	1	IN AL, DX / IN AX, DX

<b>INC</b>	INC destination Increment by 1			Flags O D I T S Z A P C X X X X
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
reg 16	3(3)	—	1	INC CX
reg 8	3(3)	—	2	INC BL
memory 8	15(15)+EA	2	2-4	INC ALPHA [DI] [BX]

<b>INT</b>	INT interrupt-type Interrupt			Flags O D I T S Z A P C X X
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
immed 8 (type=3)	52(45)	5	1	INT 3
immed 8 (type ≠ 3)	52(47)	5	2	INT 67

<b>IRET</b>	IRET (no operands) Interrupt Return			Flags O D I T S Z A P C R R R R R R R R R
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
(no operands)	32(28)	3	1	IRET

<b>JC</b>	JC short-label Jump if carry			Flags O D I T S Z A P C
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
short-label	16 or 4 (13 or 4)	—	2	JC CARRY_SET

<b>JE/JZ</b>	JE/JZ short-label Jump if equal/Jump if zero			Flags O D I T S Z A P C
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
short-label	16 or 4 (13 or 4)	—	2	JZ ZERO

<b>JMP</b>	JMP target Jump			Flags O D I T S Z A P C
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
short-label	15(13)	—	2	JMP SHORT
near-label	15(13)	—	3	JMP WITHIN_SEGMENT
far-label	15(13)	—	5	JMP FAR_LABEL
memptr 16	18(17)+EA	1	2-4	JMP [BX] TARGET
regptr 16	11(11)	—	2	JMP CX
memptr 32	24(26)+EA	2	2-4	JMP OTHER_SEG

<b>JNC</b>	JNC short-label Jump if not carry			Flags O D I T S Z A P C
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
short-label	16 or 4 (13 or 4)	—	2	JNC NOT_CARRY

<b>JNE/JNZ</b>	JNE/JNZ short-label Jump if not equal/Jump if not zero			Flags O D I T S Z A P C
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
short-label	16 or 4 (13 or 4)	—	2	JNE NOT_EQUAL

<b>LEA</b>	LEA destination, source Load effective address			Flags O D I T S Z A P C
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
reg 16, mem 16	2(6)+EA	—	2-4	LEA BX, [BP] [DI]

<b>LOOP</b>	LOOP short – label Decrement cx and jump if not zero			Flags O D I T S Z A P C
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
short label	17/5(15/5)	—	2	LOOP AGAIN

<b>MOV</b>	MOV destination, source Move			Flags O D I T S Z A P C
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
memory, accumulator	10(9)	1	3	MOV ARRAY [SI], AL
accumulator, memory	10(8)	1	3	MOV AX, TEMP_RESULT
register, register	2(2)	—	2	MOV AX, CX
register, memory	8(12)+EA	1	2-4	MOV BP, STACK_TOP
memory, register	9(9)+EA	1	2-4	MOV COUNT [DI], CX
register, immediate	4(3-4)	—	2-3	MOV CL, 2
memory, immediate	10(12-13)	1	3-6	MOV MASK [BX] [SI], 2CH
	+EA			
seg-reg, reg 16	2(2)	—	2	MOV ES, CX
seg-reg, mem 16	8(9)+EA	1	2-4	MOV DS, SEGMENT_BASE
reg 16, seg-reg	2(2)	—	2	MOV BP, SS
memory, seg-reg	9(11)+EA	1	2-4	MOV [BX] SEG_SAVE, CS

<b>MOVS/MOVSW</b>	MOVS/MOVSW (no operands) Move string (byte/word)			Flags O D I T S Z A P C
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
(no operands)	18(9)	2	1	MOVS
(repeat) (no operands)	9+17/rep (8+8/rep)	2/rep	1	REP MOVSW

<b>MUL</b>	MUL source Multiplication, unsigned			Flags O D I T S Z A P C X U U U U X
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
reg 8	70-77 (26-28)	—	2	MUL BL
reg 16	118-133 (35-37)	—	2	MUL CX
mem 8	76-83+	1	2-4	MUL MONTH [SI]
mem 16	EA(32-34) 124-139+ EA(41-43)	1	2-4	MUL [BAUD_RATE]

<b>NOP</b>	NOP (no operands) No Operation			Flags O D I T S Z A P C
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
(no operands)	3(3)	—	1	NOP

<b>NOT</b>	NOT destination Logical not			Flags O D I T S Z A P C
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
register	3(3)	—	2	NOT AX
memory	16(3)+EA	2	2-4	NOT [CHARACTER]

<b>OR</b>	OR destination, source Logical inclusive or			Flags O D I T S Z A P C X X X U X X
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
register, register	3(3)	—	2	OR AL, BL
register, memory	9(10)+EA	1	2-4	OR DX, PORT_ID [DI]
memory, register	16(10)+EA	2	2-4	OR FLAG_BYTE, CL
accumulator, immediate	4(3-4)	—	2-3	OR AL, 01101100B
register, immediate	4(4)	—	3-4	OR CX, 01H
memory, immediate	17(16)+EA	2	3-6	OR [BX], CMD_WORD

<b>OUT</b>	OUT port, accumulator Output byte or word			Flags O D I T S Z A P C
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
immed 8, accumulator	10(9)	1	2	OUT 44, AX
DX, accumulator	8(7)	1	1	OUT DX, AL

<b>POP</b>	POP destination Pop word off stack			Flags O D I T S Z A P C
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
register	8(10)	1	1	POP DX
seg-reg (CS illegal)	8(8)	1	1	POP DS
memory	17(20)+EA	2	2-4	POP [PARAMETER]

<b>PUSH</b>	PUSH source Push word onto stack			Flags O D I T S Z A P C
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
register	11(10)	1	1	PUSH SI
seg-reg (CS legal)	10(9)	1	1	PUSH ES
memory	16(16)+EA	2	2-4	PUSH RETURN_CODE [SI]

<b>RCL</b>	RCL destination, count Rotate left through carry			Flags O D I T S Z A P C X X
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
register, 1	2(2)	—	2	RCL CX, 1
register, CL	8+4/bit (5+1/bit)	—	2	RCL AL, CL
memory, 1	15(15)+EA	2	2-4	RCL ALPHA, 1
memory, CL	20+4/bit (17+1/bit) +EA	2	2-4	RCL [BP].PARAM, CL



<b>RCR</b>	RCR destination, count Rotate right through carry			Flags O D I T S Z A P C X X
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
register, 1	2(2)	—	2	RCR BX, 1
register, CL	8+4/bit (5+1/bit)	—	2	RCR BL, CL
memory, 1	15(15)+EA	2	2-4	RCR [BX].STATUS, 1
memory, CL	20+4/bit (17+1/bit) +EA	2	2-4	RCR ARRAY [DI], CL

<b>REP</b>	REP (no operands) Repeat string operation			Flags O D I T S Z A P C
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
(no operands)	2(2)	—	1	REP MOVS DEST, SRCE

<b>RET</b>	RET optional-pop-value Return from procedure			Flags O D I T S Z A P C
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
(intra-segment, no pop)	16(16)	1	1	RET
(intra-segment, pop)	20(18)	1	3	RET 4
(inter-segment, no pop)	26(22)	2	1	RET
(inter-segment, pop)	25(25)	2	3	RET 2

<b>ROL</b>	ROL destination, count Rotate left			Flags
				O D I T S Z A P C X X X X X X X
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
register, 1	2(2)	—	2	ROL BX, 1
register, CL	8+4/bit (5+1/bit)	—	2	ROL DI, CL
memory, 1	15(15)+EA	2	2-4	ROL FLAG_BYTE [DI], 1
memory, CL	20+4/bit (17+1/bit) +EA	2	2-4	ROL ALPHA, CL

<b>ROR</b>	ROR destination, count Rotate right			Flags
				O D I T S Z A P C X X X X X X X
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
register, 1	2(2)	—	2	ROR BX, 1
register, CL	8+4/bit (5+1/bit)	—	2	ROR BX, CL
memory, 1	15(15)+EA	2	2-4	ROR PORT_STATUS, 1
memory, CL	20+4/bit (17+1/bit) +EA	2	2-4	ROR CMD_WORD, CL

<b>SBB</b>	SBB destination, source Subtract with borrow			Flags
				O D I T S Z A P C X X X X X X X
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
register, register	3(3)	—	2	SBB BX, CX
register, memory	9(10)+EA	1	2-4	SBB DI, [BX].PAYMENT
memory, register	16(10)+EA	2	2-4	SBB BALANCE, AX
accumulator, immediate	4(3-4)	—	2-3	SBB AX, 2
register, immediate	4(4)	—	3-4	SBB CL, 1
memory, immediate	17(16)+EA	2	3-6	SBB COUNT [SI], 10

<b>STC</b>	STC (no operands) Set carry flag			Flags O D I T S Z A P C 1
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
(no operands)	2(2)	—	1	STC

<b>STI</b>	STI (no operands) Set interrupt enable flag			Flags O D I T S Z A P C 1
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
(no operands)	2(2)	—	1	STI

<b>SUB</b>	SUB destination, source Subtraction			Flags O D I T S Z A P C X X X X X
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
register, register	3(3)	—	2	SUB CX, BX
register, memory	9(10)+EA	1	2-4	SUB DX, MATH_TOTAL [SI]
memory, register	16(10)+EA	2	2-4	SUB [BP+2], CL
accumulator, immediate	4(3-4)	—	2-3	SUB AL, 10
register, immediate	4(4)	—	3-4	SUB SI, 5280

<b>TEST</b>	TEST destination, source Non-destructive logical and			Flags O D I T S Z A P C X X X U X X
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
register, register	3(3)	—	2	TEST SI, DI
register, memory	9(10)+EA	1	2-4	TEST SI, END_COUNT
accumulator, immediate	4(3-4)	—	2-3	TEST AL, 00100000B
register, immediate	5(4)	—	3-4	TEST BX, 0CC4H
memory, immediate	11(10)+EA	—	3-6	TEST [RETURN_COUNT], 01H

<b>XCHG</b>	XCHG destination, source Exchange registers			Flags O D I T S Z A P C
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
accumulator, reg 16	3(3)	—	1	XCHG AX, BX
memory, register	17(17)+EA	2	2-4	XCHG SEMAPHORE, AX
register, register	4(4)	—	2	XCHG AL, BL

<b>XLAT</b>	XLAT source-table Translate			Flags O D I T S Z A P C
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
source-table	11(11)	1	1	XLAT ASCII_TAB

<b>XOR</b>	XOR destination, source Logical exclusive or			Flags O D I T S Z A P C 0 X X U X X
<b>Operands</b>	<b>Clocks</b>	<b>Transfers*</b>	<b>Bytes</b>	<b>Coding Example</b>
register, register	3(3)	—	2	XOR CX, BX
register, memory	9(10)+EA	1	2-4	XOR CL, MASK_BYTE
memory, register	16(10)+EA	2	2-4	XOR ALPHA [SI], DX
accumulator, immediate	4(3-4)	—	2-3	XOR AL, 01000010B
register, immediate	4(4)	—	3-4	XOR SI, 00C2H
memory, immediate	17(16)+EA	2	3-6	XOR RETURN_CODE, 0D2H

#### 4. נוסחאון בשפת C

נוסחאון זה מתאים למהדר Microsoft Visual C++ 2010 Express Edition.  
חלקים ממנו מתאימים גם למהדרים אחרים.

Data Types (טיפוסי נתונים)

Name	Description	תאור	Size*	Range*
char	Character or small integer	תו בודד	1 byte	-128 to 127
unsigned char	Unsigned small integer	תו בודד ללא סימן	1 byte	0 to 255
short	Short Integer	מספר שלם קטן	2 bytes	-32768 to 32767
unsigned short	Unsigned short integer	מספר שלם קטן ללא סימן	2 bytes	0 to 65535
int	Integer	מספר שלם	4 bytes	-2147483648 to 2147483647
unsigned int	Unsigned integer	מספר שלם ללא סימן	4 bytes	0 to 4294967295
float	Floating point number	מספר ממשי	4 bytes	+/- 3.4e +/- 38 (~7 digits)
double	Double floating point number	מספר ממשי ארוך	8 bytes	+/- 1.7e +/- 308 (~15 digits)

\*הערכים של עמודות אלו תלויים במבנה המחשב שבו נעשה הידור התוכנית.

דוגמאות:

```
char a;
float number;
int b, c;
unsigned short NewNumber;
```

### Preprocessor directives (הנחיות לקדם-מהדר)

Description	Syntax	Example
macro definitions	#define identifier replacement	#define ArrSize 100

identifier – מזהה ; replacement – תחליף

### Operators (אופרטורים)

Description	תאור	Operator
Assignment	השמה	=

### Initialization of variables (אתחול משתנים)

```
int d = 0;  
d=75; // decimal number  
d=0x4b; // hexadecimal number
```

### Arithmetic operators (אופרטורים חשבוניים)

Description	תאור	Operator
Addition	חיבור	+
subtraction	חיסור	-
multiplication	כפל	*
division	חילוק	/
modulo	שארית	%

**Relational and equality operators (אופרטורים להשוואה ויחסים)**

Description	תאור	Operator
Equal to	שווה	==
Not equal to	שונה	!=
Greater than	גדול מ.	>
Less than	קטן מ.	<
Greater than or equal to	גדול שווה מ.	>=
Less than or equal to	קטן שווה מ.	<=

**Logical operators (אופרטורים לוגיים בין ביטויים)**

Description	תאור	Operator
NOT	היפוך	!
AND	וגם	&&
OR	או	

**Bitwise Operators (אופרטורים על סיביות)**

Description	תאור	ASM equivalent	Operator
AND	וגם	AND	&
Inclusive OR	או כולל	OR	
Exclusive OR	או מוציא	XOR	^
Bit inversion	היפוך	NOT	~
Shift Left	הזזה שמאלה	SHL	<<
Shift Right	הזזה ימינה	SHR	>>

**Basic Input/Output (קלט/פלט בסיסי)**

Description	Syntax	Example
Standard Output	<code>int putchar ( int character );</code>	<code>int a='G' ; putchar(a) ;</code>
Standard Input	<code>int getchar ( void );</code>	<code>int c; c=getchar() ;</code>

**Formatted Input/Output (פלט לפי תבנית)**

Description	Syntax	Example
Formatted output	<code>printf(format[,arg1,arg2,...]);</code>	<code>int num=10; printf("num=%d\n", num) ;</code>
Formatted Input	<code>scanf( format [,arg1,arg2,...]);</code>	<code>int num; scanf ("%d" , &amp;num) ;</code>

Specifier	Operator	פלט	Example
<code>%c</code>	Character	תו בודד	a
<code>%d</code>	Signed decimal integer	עשרוני שלם	133
<code>%e</code>	Scientific notation	עשרוני כולל נקודה וחזקה של 10	3.012e+4
<code>%f</code>	Decimal floating point	עשרוני כולל נקודה עשרונית	123.45
<code>%s</code>	String of characters	מחרוזת תווים	Hello
<code>%x</code>	Unsigned hexadecimal integer	הקסדצימלי ללא סימן	3fe



**Conditional Structures (מבני בקרה – משפטי תנאי)**

Description	Syntax	Example
if	<pre>if (condition) {     statements ; }</pre>	<pre>if (d == 100) {     printf("d is 100"); }</pre>
if .. else	<pre>scanf (condition)     statement1; else     statement2 ;</pre>	<pre>if (d == 100)     printf("d is 100"); else     printf("d is not 100");</pre>
if .. else if .. else	<pre>if (condition)     statement1 ; else if (condition)     statement2 ; else     statement3 ;</pre>	<pre>if (d &gt; 0)     printf("d is positive"); else if (d &lt; 0)     printf("d is negative"); else     printf("d is 0");</pre>

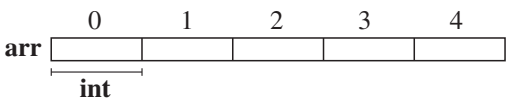
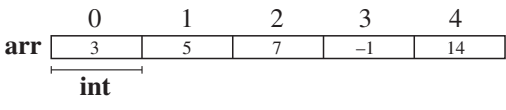
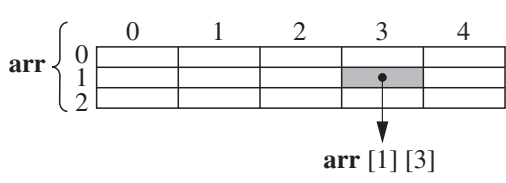
תנאי - condition ; הצהרה - statement

**Iteration Structures (מבני בקרה - לולאות)**

Description	Syntax	Example
while loop	<pre>while (expression) {     statements ; }</pre>	<pre>while (n&gt;0) {     printf(" %d \n",n);     n--;</pre>
do-while loop	<pre>do {     statements ; } while (condition);</pre>	<pre>do {     printf("Enter 0 to end: ");     scanf("%d",&amp;n); }while (n != 0);</pre>
for loop	<pre>for (initialization; condition; increase) {     statements ; }</pre>	<pre>for (i=0; i&lt;10; i++) {     printf(" %d \n",i); }</pre>

condition – תנאי ; הצהרה – statement

Arrays (מערכים)

Description	Syntax	Example
<p>הגדרת מערך חד מימדי</p> 	<p>type name [elements];</p>	<p>int arr[5];</p>
<p>אתחול והצבת ערכים במערך</p> 	<p>type name [elements] = {value1,..valueN};</p>	<p>int arr[5] = {3,5,7,-1,14};</p>
<p>הגדרת מערך דו מימדי</p> 	<p>type name [elements, elements];</p>	<p>int arr[3][5];</p>

elements - פרטים ; value - ערך

Structure of a program (מבנה כללי של תוכנית)

```
#include <stdio.h>

void main(void)
{
}

```

**Hardware Input/Output (קלט/פלט בסיסי מחומרה)**

Description	Syntax	Example
Hardware Output	Out32(hardware address, value);	Out32 (0x378, 0xAA) ;
Hardware Input	Inp32(hardware address);	int dataIN;  dataIN=Inp32 (0x379) ;

hardware address – כתובת חומרה ; value – ערך

```
#include <stdio.h>

short _stdcall Inp32(short PortAddress);

void _stdcall Out32(short PortAddress, short data);

void main(void)
{
    int dataIN;

    Out32 (0x378, 0xAA) ;

    dataIN=Inp32 (0x379) ;
}
```

**Sleep Function (פונקציית השהיה)**

Description	Syntax	Example
Suspends the execution of the current thread until the time-out interval elapses	void Sleep ( dword dwMilliseconds );	Sleep(2000);

\*For windows 32-bit registry a DWORD is a 4-bytes unsigned int.

```
#include <windows.h>

void main(void)
{
    Sleep(2000);
}
```

**5. נוסחאון בשפת VB**

נוסחאון זה מתאים למהדר Microsoft Visual Basic 2010 Express Edition.  
 חלקים ממנו מתאימים גם למהדרים אחרים.

**Data Types (טיפוסי נתונים)**

Data Type	Size in Bytes	Description	תאור	Type
Byte	1	8-bit unsigned integer	8 ביט ללא סימן	System.Byte
Char	2	16-bit Unicode characters	16 ביט ללא סימן	System.Char
Integer	4	32-bit signed integer	מספר שלם 32 ביט	System.Int32
Double	8	64-bit floating point variable	מספר ממשי 64 ביט	System.Double
Long	8	64-bit signed integer	מספר שלם 64 ביט	System.Int64
Short	2	16-bit signed integer	מספר שלם 16 ביט	System.Int16
Single	4	32-bit floating point variable	מספר ממשי 32 ביט	System.Single
String	Varies	Non-Numeric Type	מחרוזת תווים	System.String
Date	8	Date	תאריך	System.Date
Boolean	2	Non-Numeric Type	ערך בוליאני אמת או שקר	System.Boolean
Decimal	16	128-bit floating point variable	מספר ממשי ארוך 128 ביט	System.Decimal

דוגמאות:

```
Dim a, b As Byte
Dim x As Integer = -20
Dim s As String = "Hello"
```

### Operators (אופרטורים)

Description	תאור	Operator
Assignment	השמה	=

#### Initialization of variables (אתחול משתנים)

Dim d As Integer

d = 75                    \ decimal

d = &H4B                \ hexadecimal

#### Arithmetic operators (אופרטורים חשבוניים)

Description	תאור	Operator
Addition	חיבור	+
Subtraction	חיסור	-
Multiplication	כפל	*
Division	חילוק	/
Modulo	שארית	Mod
Integer Division	חילוק שלמים	\
Exponential	חזקה	^

**Relational and equality operators (אופרטורים להשוואה ויחסים)**

Description	תאור	Operator
Equal to	שווה	=
Not equal to	שונה	<>
Greater than	גדול מ-	>
Less than	קטן מ-	<
Greater than or equal to	גדול שווה מ-	>=
Less than or equal to	קטן שווה מ-	<=

**Logical operators (אופרטורים לוגיים בין ביטויים)**

Description	תאור	Operator
NOT	היפוך	Not
AND	וגם	And
OR	או	Or

**Bitwise Operators (אופרטורים על סיביות)**

Description	תאור	ASM equivalent	Operator
AND	וגם	AND	And
Inclusive OR	או כולל	OR	Or
Exclusive OR	או מוציא	XOR	Xor
Bit inversion	היפוך	NOT	Not
Shift Left	הזזה שמאלה	SHL	<<
Shift Right	הזזה ימינה	SHR	>>

**(קלט/פלט בסיסי) Basic Input/Output in Console Application**

Description	Syntax	Example
Standard Output	<code>Console.WriteLine(String)</code>	<pre>Dim value As String = "Hello"  Console. WriteLine(value)</pre>
Standard Input	<code>String = Console.ReadLine()</code>	<pre>Dim returnValue As String  returnValue = Console.ReadLine()</pre>

**(פלט לפי תבנית) Formatted Output In Console Application**

Description	Syntax	Example
Formatted output	<code>Console.WriteLine _</code> <code>("{ N [: formatCharacter ]}", arg0, ... argN)</code>	<pre>Dim i As Integer = 123456  Console.WriteLine ("{0:D}", i)</pre>

Format Character	Output	פלט	Example
D or d	Signed decimal integer	עשרוני שלם	133
E or e	Scientific notation	עשרוני, כולל נקודה וחזקה של 10	3.012e+4
F or f	Decimal floating point	עשרוני, כולל נקודה עשרונית	123.45
X or x	Unsigned hexadecimal integer	הקסדצימלי ללא סימן	3fe



**Basic Input/Output in Windows Application (קלט/פלט בסיסי)**

Description	Syntax	Example
Standard Output	MsgBox(String) or MessageBox.Show(String)	<pre>Dim str As String = "Hello" MsgBox(str) MessageBox.Show(str)</pre>
Standard Input	String = InputBox(String)	<pre>Dim s As String Dim num As Integer s = InputBox("Enter some text") num = InputBox("Enter Number") *</pre>

\* Automatic casting – המרת טיפוסים אוטומטית

**Iteration Structures (מבני בקרה - לולאות)**

Description	Syntax	Example
while loop	While expression statement End While	<pre>While n &gt; 0     MsgBox(n)     n = n - 1 End While</pre>
do-while loop	Do statement Loop While condition	<pre>Do     n = InputBox("Enter 0 to end: ") Loop While n &lt;&gt; 0</pre>
for loop	for initialization To condition Step increase statement Next	<pre>For i = 0 To 6 Step 2     MsgBox("i=" &amp; i) Next</pre>

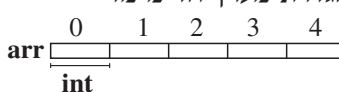
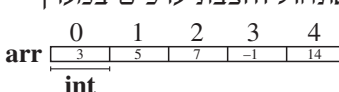
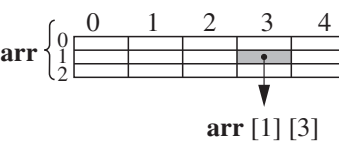
condition – תנאי ; statement – הצהרה

**Conditional Structures (מבני בקרה - משפטי תנאי)**

Description	Syntax	Example
if	If condition Then [statements] End If	If d = 100 Then MsgBox("d is 100") End If
if .. else	If condition Then [statements] Else [statements] End If	If d = 100 Then MsgBox("d is 100") Else MsgBox("d is not 100") End If
if .. else if .. else	If condition Then [statements] Else If condition Then [statements] ... Else [statements] End If	If d > 0 Then MsgBox("d is positive") ElseIf d < 0 Then MsgBox("d is negative") Else MsgBox("d is 0") End If

תנאי - condition ; הצהרה - statement

**Arrays (מערכים)**

Description	Syntax	Example
<p>הגדרת מערך חד מימדי</p>  <p>arr [0] [1] [2] [3] [4] int</p>	Dim name (elements) As type	Dim arr(4) As Integer
<p>אתחול והצבת ערכים במערך</p>  <p>arr [0] [1] [2] [3] [4] [3] [5] [7] [-1] [14] int</p>	Dim name (elements) As type = {value1,..valueN}	Dim arr1() As Integer = {3, 5, 7, -1, 14}
<p>הגדרת מערך דו מימדי</p>  <p>arr {0 [ ] [ ] [ ] [ ] 1 [ ] [ ] [ ] [ ] 2 [ ] [ ] [ ] [ ] arr [1] [3]</p>	Dim name (elements, elements) type	Dim arr2(2, 4) As Integer

elements - פרטים ; value - ערך

**Structure of a program in Console Application (מבנה כללי של תוכנית):**

```
Module Module1
    Sub Main()
    End Sub
End Module
```

**Structure of a program in Windows Application (מבנה כללי של תוכנית):**

```
Public Class Form1
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
    End Sub
End Class
```

**(קלט/פלט בסיסי מחומרה) Hardware Input/Output**

Description	Syntax	Example
Hardware Output	Out32(hardware address, value);	Out32 (&H378, &HAA) ;
Hardware Input	Inp32(hardware address);	int dataIN;  dataIN=Inp32 (&H379) ;

hardware address – כתובת חומרה ; value – ערך

Public Class Form1

```
Private Declare Function Inp32 Lib "inout32.dll" Alias
    "Inp32" (ByVal PortAddress As Integer) As Integer

Private Declare Sub Out32 Lib "inout32.dll" Alias "Out32"
    (ByVal PortAddress As Integer, ByVal Value As Integer)

Private Sub Form1_Load(ByVal sender As System.Object, ByVal
    e As System.EventArgs) Handles MyBase.Load

    Dim dataIN As Integer

    Out32 (&H378, &HAA)

    dataIN = Inp32 (&H379)

End Sub
```

End Class

**(פונקציית השהיה) Sleep Function**

Description	Syntax	Example
Suspends the execution of the current thread until the time-out interval elapses.	Public Shared Sub Sleep ( _ Milliseconds Timeout As Integer _)	System.Threading.Thread. Sleep(2000)

**בהצלחה!**