

## 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:
    1. ענה על מספר השאלות הנדרש בשאלון. המעריך יקרא ויעריך את מספר השאלות הנדרש בלבד, לפי סדר כתיבתן במחברתך, ולא יתייחס לתשובות נוספות.
    2. התחל כל תשובה לשאלה בעמוד חדש.
    3. רשום את כל תשובותיך אך ורק בעט.
    4. הקפד לנסח את תשובותיך כהלכה, ולסרטט את תרשימיך בבהירות.
    5. כתוב את תשובותיך בכתב-יד ברור, כדי לאפשר הערכה נאותה שלהן.
    6. אם לדעתך חסרים נתונים הדרושים לפתרון שאלה, אתה רשאי להוסיף אותם, אך עליך להסביר מדוע הוספת אותם.
    7. 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 לנבחנות ולנבחנים כאחד.

**בהצלחה!**

## השאלות

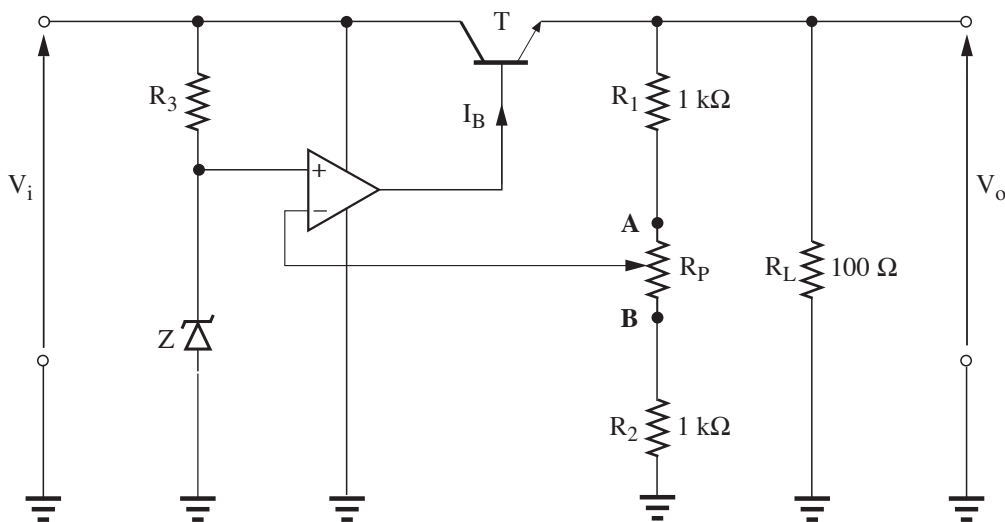
בשאלון זה תשע שאלות בשני פרקים. יש לענות על חמש שאלות בלבד,  
שאלה אחת לפחות מכל פרק.

### 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 של מייצב מתח. מגבר השרת במעגל – אידיאלי.  
נתוני המעגל הם:  $R_P = 1 \text{ k}\Omega$  ;  $\beta = 100$  ;  $V_Z = 2 \text{ V}$  ;  $V_i = 10 \text{ V}$  .

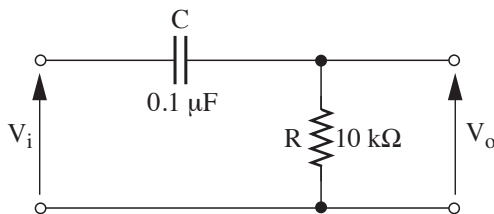


איור לשאלה 1

- א. חשב את מתח המוצא  $V_o$  , כשהזחלן של הנגד המשתנה  $R_P$  נמצא בנקודה A .
- ב. חשב את מתח המוצא  $V_o$  , כשהזחלן של הנגד המשתנה  $R_P$  נמצא בנקודה B .
- ג. חשב את זרם הבסיס ( $I_B$ ) של הטרנזיסטור T , כשהזחלן של הנגד המשתנה  $R_P$  נמצא בנקודה B .

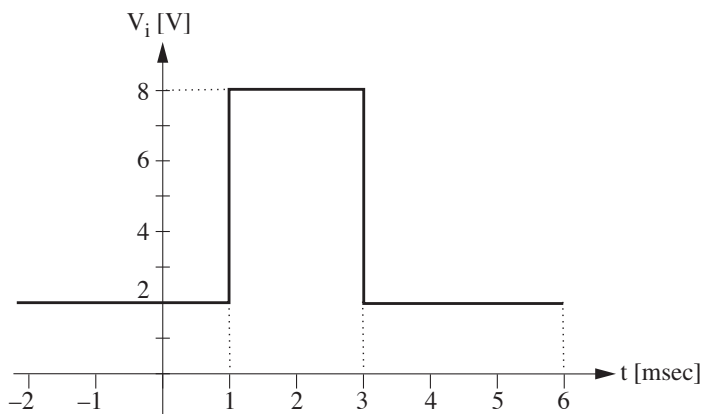
## שאלה 2

באיור א' לשאלה 2 מתוארת רשת חשמלית.



## איור א' לשאלה 2

למבוא הרשת מספקים את הדופק המתואר באיור ב' לשאלה.



## איור ב' לשאלה 2

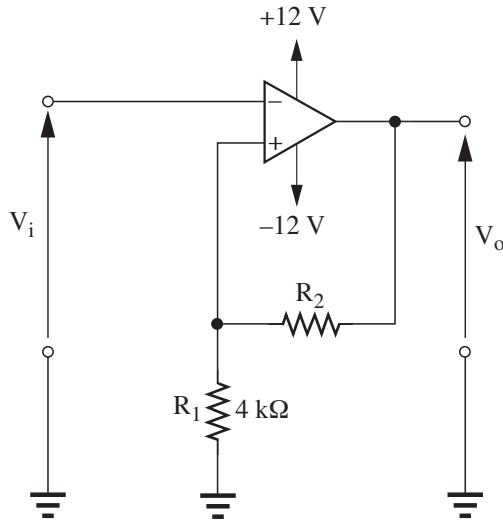
א. העתק למחברתך את מתח המבוא ( $V_i$ ), וסרטט מתחתיו, בהתאמה, את מתח המוצא ( $V_o$ ), כפונקציה של הזמן.

ב. חשב את מתח המוצא בזמן  $t = 2 \text{ msec}$ .

ג. חשב את מתח המוצא בזמן  $t = 4 \text{ msec}$ .

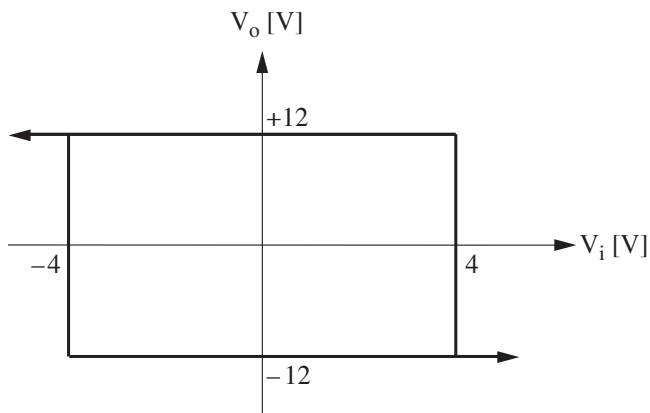
### שאלה 3

באיור א' לשאלה 3 מתואר המעגל החשמלי של משווא מסוג שמיט טריגר. מגבר השרת שבמעגל – אידיאלי.



#### איור א' לשאלה 3

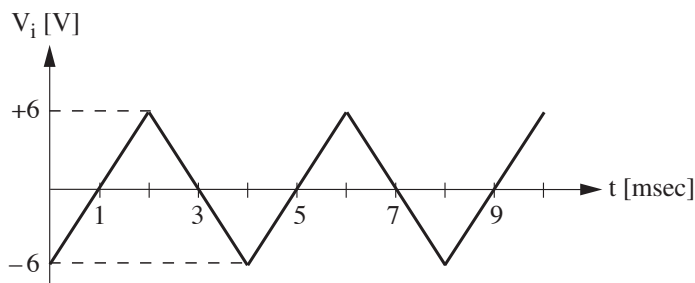
באיור ב' לשאלה מתואר אופיין המעבר של המשוואה.



#### איור ב' לשאלה 3

א. חשב את ההתנגדות של הנגד  $R_2$ .

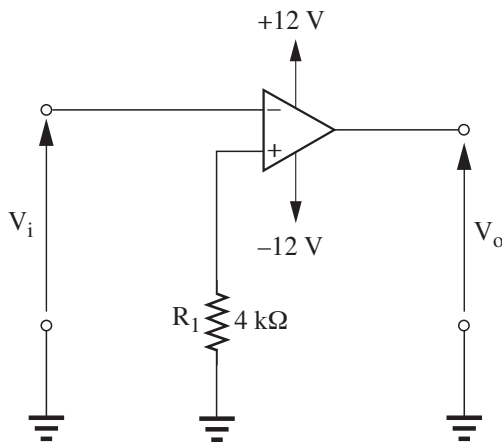
ב. מספקים למבוא המעגל גל משולש, כמתואר באיור ג' לשאלה.



איור ג' לשאלה 3

העתק למחברתך את מתח המבוא ( $V_i$ ), וסרטט מתחתיו, בהתאמה, את מתח המוצא ( $V_o$ ), כפונקציה של הזמן. ציין את הערך המרבי ואת הערך המזערי של המתחים על-גבי הגרפים.

ג. מסירים את הנגד  $R_2$  מהמעגל, כמתואר באיור ד' לשאלה.



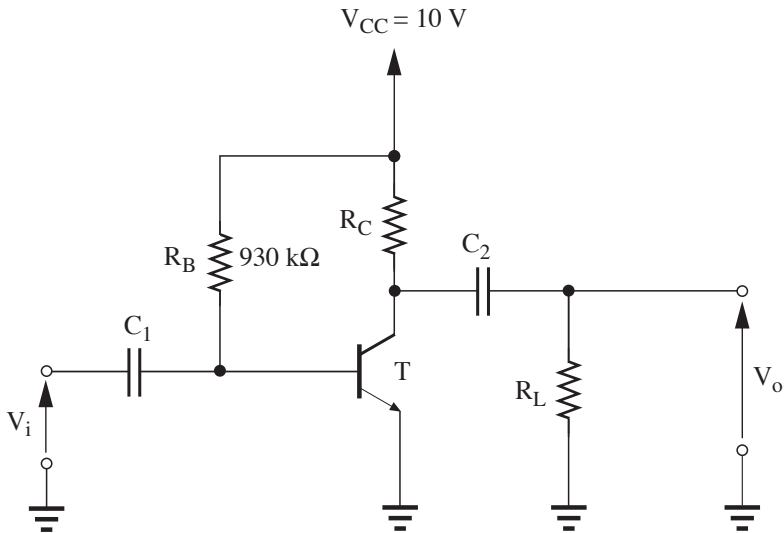
איור ד' לשאלה 3

סרטט את אופיין המעבר, המתאר את מתח המוצא של המעגל כפונקציה של מתח המבוא,  $V_o = f(V_i)$ .

#### שאלה 4

באיור לשאלה 4 מתואר המעגל החשמלי של מגבר טרנזיסטורי. הָגְגִי הקבלים במעגל – זניחים.

נתוני הטרנזיסטור T הם:  $\beta = h_{fe} = 100$  ;  $h_{ie} = 2 \text{ k}\Omega$  ;  $V_{BE} = 0.7 \text{ V}$  .



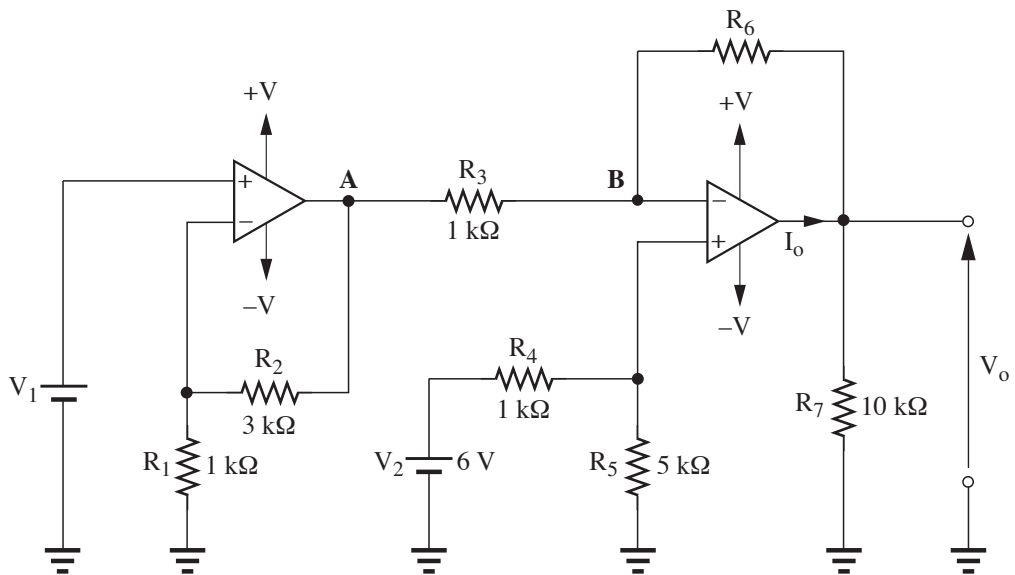
#### איור לשאלה 4

- א. חשב את זרם הבסיס ( $I_B$ ) של הטרנזיסטור, כאשר לא מספקים מתח למבוא־המעגל.
- ב. חשב את ערכו של הנגד  $R_C$ , הנדרש כדי לקבל מתח  $V_{CE} = 4 \text{ V}$ , כאשר לא מספקים מתח למבוא־המעגל.
- ג. מספקים למבוא המעגל מתח  $V_i = 20 \text{ mV}$ , ומתקבל מתח מוצא  $V_o = 2 \text{ V}$ .  
חשב את ההתנגדות של הנגד  $R_L$ .

## שאלה 5

באיור לשאלה 5 נתון מעגל חשמלי הכולל מגברי־שרת אידיאליים.

נתוני המעגל הם:  $V_A = 4\text{ V}$  (המתח בנקודה A),  $I_0 = 2\text{ mA}$ .



## איור לשאלה 5

- א. חשב את המתח  $V_1$ .
- ב. חשב את המתח בנקודה B.
- ג. חשב את מתח המוצא  $V_0$ .
- ד. חשב את ההתנגדות של הנגד  $R_6$ .

**פרק שני: מבוא להנדסת מחשבים**

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

**שאלה 6**

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

**קטע מתכנית בשפת C**

```
1.   int arr[3][4];
2.   int h,w,c;
3.   for(w=0;w<4;w++)
4.   {
5.       for(h=0;h<3;h++)
6.       {
7.           printf("Enter a Number:");
8.           scanf("%d",&arr[h][w]);
9.       }
10.  }
11.  c=0;
12.  for(w=0;w<4;w++)
13.  {
14.      for(h=0;h<3;h++)
15.      {
16.          if(arr[h][w]>=5) c++;
17.      }
18.  }
19.  printf("%d",c);
```

המשך בעמוד 9

**קטע מתכנית בשפת Visual Basic**

```

1. Dim arr(2, 3) As Integer
2. Dim h As Integer, w As Integer, c As Integer
3. For w = 0 To 3
4.     For h = 0 To 2
5.         arr(h, w) = InputBox("Enter a Number:")
6.     Next
7. Next
8. c = 0
9. For w = 0 To 3
10.    For h = 0 To 2
11.        If arr(h, w) >= 5 Then c = c + 1
12.    Next
13. Next
14. MsgBox(c)

```

בחר בקטע התכנית בשפת C או בקטע התכנית בשפת Visual Basic, וענה על סעיפים א'-ג' שלהלן:

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

\* בקטע התכנית בשפת C – 1, 8, 12, 16.

\* בקטע התכנית בשפת Visual Basic – 1, 5, 9, 11.

**ב.** הסבר מה מבצע קטע התכנית.

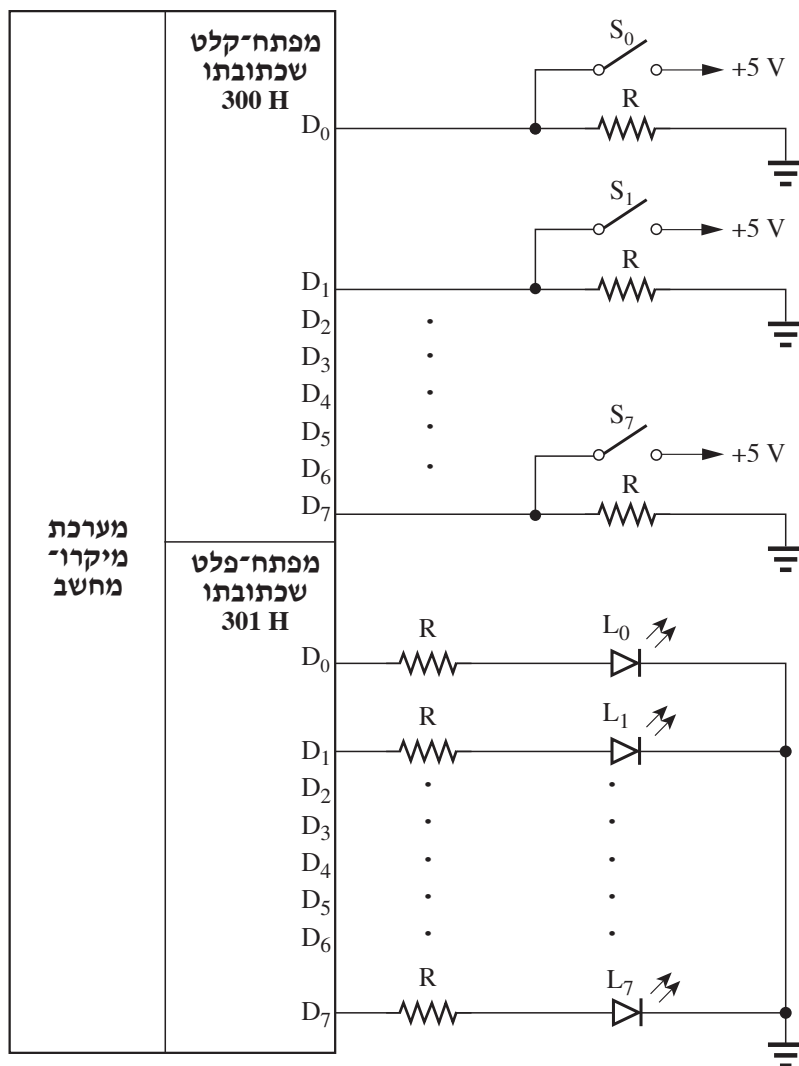
**ג.** 1. הצג את המערך arr כטבלה דו־ממדית, ורשום בו את תכני התאים לאחר הרצת קטע התכנית, אם המשתמש הקליד את המספרים הבאים (משמאל לימין):

3    4    7    2    -1    2    -4    7    4    7    0    -9

2. מה יהיה ערכו של המשתנה c לאחר ביצוע קטע התכנית?

## שאלה 7

באיור לשאלה 7 מתוארת מערכת מיקרו-מחשב המבוססת על המיקרו-מעבד 8086/88. המערכת מכילה שמונה מפסקים המחוברים למפתח-קלט שכתובתו 300 H, ושמונה נוריות LED המחוברות למפתח-פלט שכתובתו 301 H.



איור לשאלה 7

כאשר מפסק סגור – ערך הסיבית המתאימה לו הוא '1'.

כאשר מפסק פתוח – ערך הסיבית המתאימה לו הוא '0'.

להלן תת־שגרה הכתובה בשפת־הסף של המיקרו־מעבד 8086/88, הפועלת על מערכת המיקרו־מחשב המתוארת באיור.

```
1.  MAIN:      MOV    DX, 300 H
2.                  IN     AL, DX
3.                  MOV    CL, 0
4.                  MOV    BX, 8
5.  NEXT:      RCL     AL, 1
6.                  JNC    NOTC
7.                  INC    CL
8.  NOTC:      DEC     BX
9.                  JNZ    NEXT
10.                 STC
11.                 MOV    AL, 0
12.                 RCL     AL, CL
13.                 MOV    DX, 301 H
14.                 OUT     DX, AL
15.                 RET
```

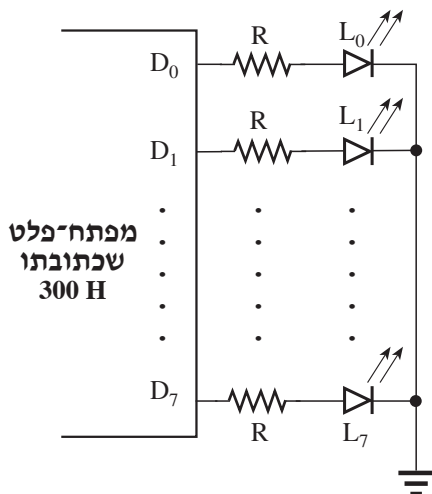
**א.** הסבר את ההוראות 2, 5, 10 ו־14 בתת־השגרה.

**ב.** מה יהיה תוכנו של האוגר AL לאחר ביצוע הוראה 2, כאשר המפסקים  $S_0$  ו־ $S_7$  סגורים וכל המפסקים האחרים פתוחים?

**ג.** מה יהיה תוכנו של האוגר AL בסיום תת־השגרה, כאשר המפסקים  $S_0, S_1, S_4$  ו־ $S_5$  סגורים והמפסקים  $S_2, S_3, S_6$  ו־ $S_7$  פתוחים?

## שאלה 8

באיור לשאלה 8 נתון תרשים של מפתח-פלט שכתובתו 300 H .



איור לשאלה 8

כתוב תכנית בשפת C או בשפת Visual Basic , הקולטת מהמשתמש מספר שלם.

אם המספר שנקלט חיובי – יידלקו נוריות ה-LED  $L_0 \div L_3$  .

אם המספר שנקלט הוא 0 או שלילי – יידלקו נוריות ה-LED  $L_4 \div L_7$  .

## שאלה 9

כתוב תת-שגרה בשפת-הסף של המיקרו-מעבד 8086/88 , הבודקת את תוכן האוגר AL :

אם הערך של ארבע הסיביות  $D_4 D_5 D_6 D_7$  באוגר גדול מהערך של ארבע הסיביות  $D_0 D_1 D_2 D_3$

בו, האוגר BL יקבל את הערך FF H . אם לא – האוגר BL יקבל את הערך 00 H .

**לדוגמה:** אם ערכו של האוגר AL הוא B2 H – האוגר BL יקבל את הערך FF H .

## 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 -  $R_i$  [ $\Omega$ ]

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

הגבר הספק בדציבלים -  $A_P$  [dB]

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

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

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

$$A_{VT} \text{ (dB)} = A_{V1} \text{ (dB)} + A_{V2} \text{ (dB)} + A_{V3} \text{ (dB)} + \dots + A_{VN} \text{ (dB)}$$

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

### מאזן הספקים

הספק מבוא -  $P_I \text{ [W]}$   
הספק נצרך מהספקים -  $P_{CC} \text{ [W]}$   
הספק העומס -  $P_L \text{ [W]}$   
הספק מבוזבז -  $P_{diss} \text{ [W]}$

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

### משוב שלילי

הגבר עם משוב (בחוג סגור) -  $A_f \text{ [A]}$   
הגבר ללא משוב -  $A$   
(הגבר חוג פתוח)  
מקדם משוב -  $\beta$

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

### משוב מתח טורי

התנגדות מבוא עם משוב -  $R_{if} \text{ [\Omega]}$   
התנגדות מבוא ללא משוב -  $R_i \text{ [\Omega]}$   
התנגדות מוצא עם משוב -  $R_{of} \text{ [\Omega]}$   
התנגדות מוצא ללא משוב -  $R_o \text{ [\Omega]}$

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

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

### טרנזיסטור דו-נושאי (בתחום הפעיל)

בהזנחת זרם הזליגה :  $I_{CBO}$

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

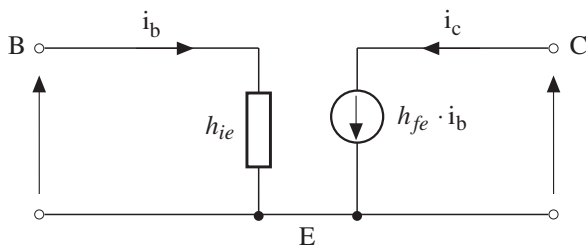
זרם הקולט -  $I_C$  [A]

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

### מגברי שרת

#### מגבר מהפך

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

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

#### מגבר עוקב

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

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

## 2. נוסחאות באלקטרוניקה ספרתית

- הערך הרגעי של המתח  $- v(t)$   $[V]$   
הערך שאליו המתח שואף  $- V_\infty$   $[V]$   
להגיע כאשר  $t \rightarrow \infty$   
הערך ההתחלתי של המתח  $- V_{0+}$   $[V]$

$$v(t) = V_\infty - (V_\infty - V_{0+}) e^{-\frac{t}{\tau}}$$

$$t = -\tau \cdot \ln \left( \frac{V_\infty - v(t)}{V_\infty - V_{0+}} \right)$$

- קבוע זמן  $- \tau$   $[\text{sec}]$

$$\tau = RC$$

- 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  $- t_r$   $[\text{sec}]$

$$t_r = 2.2\tau$$

### 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<br>Add with carry |            |       | Flags                            |
|------------------------|-------------------------------------------|------------|-------|----------------------------------|
|                        |                                           |            |       | O D I T S Z A P C<br>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<br>Addition |            |       | Flags                            |
|------------------------|-------------------------------------|------------|-------|----------------------------------|
|                        |                                     |            |       | O D I T S Z A P C<br>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<br>Logical and |            |       | Flags                            |
|------------------------|----------------------------------------|------------|-------|----------------------------------|
|                        |                                        |            |       | O D I T S Z A P C<br>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                |

| CALL      | CALL target<br>Call a procedure |            |       | Flags O D I T S Z A P C |
|-----------|---------------------------------|------------|-------|-------------------------|
| Operands  | Clocks                          | Transfers* | Bytes | Coding Example          |
| 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]    |

| CLC           | CLC (no operands)<br>Clear carry flag |            |       | Flags O D I T S Z A P C<br>0 |
|---------------|---------------------------------------|------------|-------|------------------------------|
| Operands      | Clocks                                | Transfers* | Bytes | Coding Example               |
| (no operands) | 2(2)                                  | —          | 1     | CLC                          |

| CLI           | CLI (no operands)<br>Clear interrupt flag |            |       | Flags O D I T S Z A P C<br>0 |
|---------------|-------------------------------------------|------------|-------|------------------------------|
| Operands      | Clocks                                    | Transfers* | Bytes | Coding Example               |
| (no operands) | 2(2)                                      | —          | 1     | CLI                          |

| CMP                    | CMP destination, source<br>Compare destination to source |            |       | Flags O D I T S Z A P C<br>X X X X X X X |
|------------------------|----------------------------------------------------------|------------|-------|------------------------------------------|
| Operands               | Clocks                                                   | Transfers* | Bytes | Coding Example                           |
| 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<br>Compare string |                   |              | Flags<br>O D I T S Z A P C<br>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<br>(5+22/rep)                           | 2/rep             | 1            | REPE CMPS ID, KEY                                   |

|                 |                                                  |                   |              |                                                     |
|-----------------|--------------------------------------------------|-------------------|--------------|-----------------------------------------------------|
| <b>DAA</b>      | DAA (no operands)<br>Decimal adjust for addition |                   |              | Flags<br>O D I T S Z A P C<br>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)<br>Decimal adjust for subtraction |                   |              | Flags<br>O D I T S Z A P C<br>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<br>Decrement by 1 |                   |              | Flags<br>O D I T S Z A P C<br>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]                                    |

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

| IN                   | IN accumulator, port<br>Input byte or word |            |       | Flags<br>O D I T S Z A P C |
|----------------------|--------------------------------------------|------------|-------|----------------------------|
| Operands             | Clocks                                     | Transfers* | Bytes | Coding Example             |
| 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      |

| INC      | INC destination<br>Increment by 1 |            |       | Flags<br>O D I T S Z A P C<br>X X X X |
|----------|-----------------------------------|------------|-------|---------------------------------------|
| Operands | Clocks                            | Transfers* | Bytes | Coding Example                        |
| 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]                   |

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

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

|                 |                                 |                   |              |                         |
|-----------------|---------------------------------|-------------------|--------------|-------------------------|
| <b>JC</b>       | JC short-label<br>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<br>(13 or 4)            | —                 | 2            | JC CARRY_SET            |

|                 |                                                 |                   |              |                         |
|-----------------|-------------------------------------------------|-------------------|--------------|-------------------------|
| <b>JE/JZ</b>    | JE/JZ short-label<br>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<br>(13 or 4)                            | —                 | 2            | JZ ZERO                 |

|                 |                    |                   |              |                         |
|-----------------|--------------------|-------------------|--------------|-------------------------|
| <b>JMP</b>      | JMP target<br>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<br>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<br>(13 or 4)                 | —                 | 2            | JNC NOT_CARRY           |

|                 |                                                           |                   |              |                         |
|-----------------|-----------------------------------------------------------|-------------------|--------------|-------------------------|
| <b>JNE/JNZ</b>  | JNE/JNZ short-label<br>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<br>(13 or 4)                                      | —                 | 2            | JNE NOT_EQUAL           |

|                 |                                                   |                   |              |                         |
|-----------------|---------------------------------------------------|-------------------|--------------|-------------------------|
| <b>LEA</b>      | LEA destination, source<br>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<br>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<br>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)<br>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<br>(8+8/rep)                               | 2/rep             | 1            | REP MOVSW               |

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

|                 |                                   |                   |              |                         |
|-----------------|-----------------------------------|-------------------|--------------|-------------------------|
| <b>NOP</b>      | NOP (no operands)<br>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<br>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<br>Logical inclusive or |                   |              | Flags O D I T S Z A P C<br>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<br>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<br>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<br>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<br>Rotate left through carry |                   |              | Flags O D I T S Z A P C<br>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<br>(5+1/bit)                                | —                 | 2            | RCL AL, CL                     |
| memory, 1       | 15(15)+EA                                           | 2                 | 2-4          | RCL ALPHA, 1                   |
| memory, CL      | 20+4/bit<br>(17+1/bit)<br>+EA                       | 2                 | 2-4          | RCL [BP].PARAM, CL             |

|                 |                                                      |                   |              |                                |
|-----------------|------------------------------------------------------|-------------------|--------------|--------------------------------|
| <b>RCR</b>      | RCR destination, count<br>Rotate right through carry |                   |              | Flags O D I T S Z A P C<br>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<br>(5+1/bit)                                 | —                 | 2            | RCR BL, CL                     |
| memory, 1       | 15(15)+EA                                            | 2                 | 2-4          | RCR [BX].STATUS, 1             |
| memory, CL      | 20+4/bit<br>(17+1/bit)<br>+EA                        | 2                 | 2-4          | RCR ARRAY [DI], CL             |

|                 |                                              |                   |              |                         |
|-----------------|----------------------------------------------|-------------------|--------------|-------------------------|
| <b>REP</b>      | REP (no operands)<br>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 MOVSB DEST, SRCE    |

|                         |                                                 |                   |              |                         |
|-------------------------|-------------------------------------------------|-------------------|--------------|-------------------------|
| <b>RET</b>              | RET optional-pop-value<br>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                   |

| ROL          | ROL destination, count<br>Rotate left |            |       | Flags O D I T S Z A P C<br>X X |
|--------------|---------------------------------------|------------|-------|--------------------------------|
| Operands     | Clocks                                | Transfers* | Bytes | Coding Example                 |
| register, 1  | 2(2)                                  | —          | 2     | ROL BX, 1                      |
| register, CL | 8+4/bit<br>(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<br>(17+1/bit)<br>+EA         | 2          | 2-4   | ROL ALPHA, CL                  |

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

| SBB                    | SBB destination, source<br>Subtract with borrow |            |       | Flags O D I T S Z A P C<br>X X X X X |
|------------------------|-------------------------------------------------|------------|-------|--------------------------------------|
| Operands               | Clocks                                          | Transfers* | Bytes | Coding Example                       |
| 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)<br>Set carry flag |                   |              | Flags O D I T S Z A P C<br>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)<br>Set interrupt enable flag |                   |              | Flags O D I T S Z A P C<br>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<br>Subtraction |                   |              | Flags O D I T S Z A P C<br>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<br>Non-destructive logical and |                   |              | Flags O D I T S Z A P C<br>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<br>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<br>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<br>Logical exclusive or |                   |              | Flags O D I T S Z A P C<br>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 | int putchar ( int character ); | int a='G' ;<br><br>putchar(a) ; |
| Standard Input  | int getchar ( void );          | int c;<br><br>c=getchar() ;     |

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

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

| Specifier | Operator                     | פלט                           | Example  |
|-----------|------------------------------|-------------------------------|----------|
| %c        | Character                    | תו בודד                       | a        |
| %d        | Signed decimal integer       | עשרוני שלם                    | 133      |
| %e        | Scientific notation          | עשרוני כולל נקודה וחזקה של 10 | 3.012e+4 |
| %f        | Decimal floating point       | עשרוני כולל נקודה עשרונית     | 123.45   |
| %s        | String of characters         | מחרוזת תווים                  | Hello    |
| %x        | 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> |

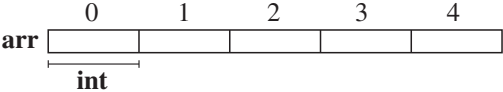
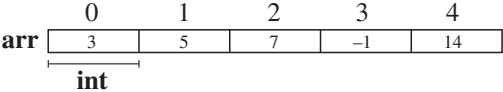
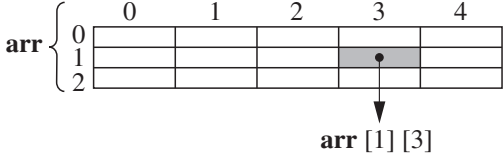
condition – תנאי ; הצהרה – statement

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

elements – פרטים ; ערך – value

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;<br><br>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            | >>       |

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

| Description     | Syntax                      | Example                                                                 |
|-----------------|-----------------------------|-------------------------------------------------------------------------|
| Standard Output | Console.WriteLine(String)   | Dim value As String<br>= "Hello"<br><br>Console.<br>WriteLine(value)    |
| Standard Input  | String = Console.ReadLine() | Dim returnValue As<br>String<br><br>returnValue =<br>Console.ReadLine() |

Formatted Output In Console Application (פלט לפי תבנית)

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

| 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)            | Dim str As String = "Hello"                                                                                  |
|                 | or                        | MsgBox(str)                                                                                                  |
|                 | MessageBox.Show(String)   | MessageBox.Show(str)                                                                                         |
| Standard Input  | String = InputBox(String) | Dim s As String<br>Dim num As Integer<br>s = InputBox("Enter some text")<br>num = InputBox("Enter Number") * |

\* Automatic casting – המרת טיפוסים אוטומטית

### Iteration Structures (מבני בקרה - לולאות)

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

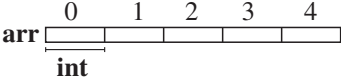
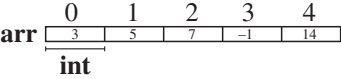
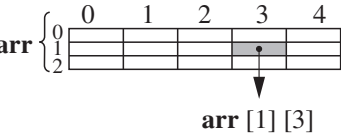
הצהרה – statement ; תנאי – condition

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

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

תנאי - condition ; הצהרה - statement

Arrays (מערכים)

| Description                                                                                                  | Syntax                                          | Example                                   |
|--------------------------------------------------------------------------------------------------------------|-------------------------------------------------|-------------------------------------------|
| הגדרת מערך חד מימדי<br>     | Dim name (elements) As type                     | Dim arr(5) As Integer                     |
| אתחול והצבת ערכים במערך<br> | Dim name (elements) As type = {value1,..valueN} | Dim arr1() As Integer = {3, 5, 7, -1, 14} |
| הגדרת מערך דו מימדי<br>     | Dim name (elements, elements) type              | Dim arr2(3, 5) 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
```

Hardware Input/Output (קלט/פלט בסיסי מחומרה)

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

hardware address – כתובת חומרה – ; value – ערך

Public Class Form1

```
Private Declare Function Inp32 Lib "inpout32.dll" Alias
    "Inp32" (ByVal PortAddress As Integer) As Integer

Private Declare Sub Out32 Lib "inpout32.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
```

Sleep Function (פונקציית השהיה)

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