**8086**

8086 הוא רכיב אלקטרוני שכולל בתוכו 40 הדקים(רגליים). המיוחד שבו זה המבנה הכולל בתוכו אוגרים(רכיבים אלקטרוניים) שמאפשרים לבצע עיבוד של הנתונים באופן מהיר ואחסון של הנתונים. כאשר בונים משחב אנו צריכים בדר"כ 4 יחידות בסיסיות.

מעבד(עושה את העבודה השחורה), רכיב זיכרון, התקן קלט(מקלדת, עכבר) והתקן פלט(מסך, מנורה). ברכיב הזיכרון יש בדר"כ שני סוגים:

1. RAM – מאפשר קריאה וכתיבה של תוכניות
2. ROM – מאפשר קריאה בלבד. מיועד בעיקר למפתח תוכניות כך שלאחר שנגמר שלב הפיתוח הוא צורב את התוכניות ברכיב מיוחד ולא מאפשר שינוי בתוכנית.

כדי להעביר מידע בין האוגרים השונים לבין רכיבי הזיכרון יש יחדה מיוחדת שנקראת יחידת הבקרה שתפקידה למנוע התנגשויות בין הנתונים לקווי הכתובת ז"א יש אפשרות להעביר את המידע דרך אותם קווים של קווי הכתובת כדי למנוע התנגשות יחידת הבקרה תאפשר תזמון מדויק בניהם ל – 8086 יש 40 הדקים ומתוכם 20 מיועדים לקווי כתובת. את קווי הכתובת מסמנים בהדקים מ – A0 – A19 (A– address) ז"א שמספר הכתובות שניתן הוא 220

**הוראות בשפת סף:**

MOV operand1, operand2 🡪 משמעות פקודה זו – העתק את תוכן אופרנד 2 אל תוך אופרנד 1

ADD operand1, operand2 🡪 משמעות פקודה זו – הוספה של אופרנד 2 אל אופרנד 1

SUB operand1, operand2 🡪 משמעות פקודה זו – לחסר מאופרנד 1 את אופרנד 2

XCHG operand1, operand2 🡪 משמעות פקודה זו – החלף בין הערכים של האופרנדים

SHR operand1, operand2 🡪 הזזה של כל הבתים באופרנד היעד ימינה כמה פעמים לפי ערך אופרנד המקור. בבתים החסרים משמאל יכנסו 0.

SHL operand1, operand2 🡪 הזזה של כל הבתים באופרנד היעד שמאלה כמה פעמים לפי ערך אופרנד המקור. בבתים החסרים מימין יכנסו 0..

ROR operand1, operand2 🡪 הזזה של כל הבתים באופרנד היעד ימינה כמה פעמים לפי ערך אופרנד המקור. הבתים שנמחקו יכנסו לשמאל.

ROL operand1, operand2 🡪 הזזה של כל הבתים באופרנד היעד שמאלה כמה פעמים לפי ערך אופרנד המקור. הבתים שנמחקו יכנסו לימין.

\*בשפה זו חשוב להדגיש שההוראות מתורגמות לשפת מכונה על ידי המהדר(אסמבלר). האסמבלר היא תכנית הקוראת את הוראות התוכנית כמחרוזת, ומפרקת אותה לאופרטור ואופרנדים ובהתאם לכך מתרגמת את ההוראה לקוד מסוים בשפת מכונה.

הוראה בשפת סף: MOV AX, a מתורגמת להוראה בשפת מכונה בה קוד הפעולה הוא: A1h

* הנתונים יכולים מוצגים בשיטה בינארית, עשרונית או הקסאדצימאלית.
* נתונים אלו יתורגמו על ידי האסמבלר לנתונים בינאריים
* כדי להבדיל בין הייצוגים השונים, אנו מוסיפים סימול בסוף המספר:
  1. הסימול d בסיום מספר מציין שיטת ספירה עשרונית, ובהתאם המספר 23d יתורגם ל: 000101112.
  2. הסימול h מציין מספר בשיטה הקסאדצימאלית

**0**AFh, 10h **(אפס מוביל שמספר מתחיל באות)**

* + הסימול b לציון מספר בינארי, לדוגמה: 100101011b.
  + ברירת מחדל מספר ללא סימול הוא מספר עשרוני

**להוראות בשפת סף יש מבנה אחיד 4 שדות:**

[Label:] Mnemonic [Operands] [;Comment]

* סימול [ ] משמעותו אופציונלי, שדה חובה יחידי הוא שם מנמוני של האופרטור

**תפקיד השדות:**

1. השדה הראשון הוא תווית Label אותה ניתן להוסיף לחלק משורות התוכנית.

התווית מציינת את כתובת ההוראה הביצועית אליה מוצמדת התווית. לאחר התווית ובצמוד אליה, יש להוסיף את התו ' : '.

שימוש נפוץ בתווית, בהוראות קפיצה, ומימוש הוראות תנאי ולולאות. דוגמה:

start:

בזמן טעינת התכנית לזיכרון תווית מוחלפת בכתובת.

1. השדה השני הוא חובה והוא מציין את קוד הפעולה(ADD,MOV…)
2. השדה השלישי הוא רשימת אופרנדים (operand) המכילים את הארגומנטים עליהם מתבצעת ההוראה

אופרנד יכול להיות(תלוי בהוראה): נתון, אוגר או משתנה בזיכרון .

מספר אופרנדים במעבד 8086 הוא בין 0 לשניים.

1. משמש להערות. נוכל לכתוב הערה לאחר ' ; '. כל מה שבאותה שורה לאחר הנקודה פסיק המכונה תתעלם מזה.

**הזיכרון**

* יחידת הזיכרון של המחשב משמשת לאחסון נתונים והוראות של תוכניות.
  1. חלק מן הנתונים מאוחסנים בה לפני ביצוע התוכנית, חלקם במהלכה וחלקם נשמרים בזיכרון גם לאחר שביצוע התוכנית הסתיים
* ניתן לחלק את הזיכרון של המחשב לשני סוגים:
  1. זיכרון ראשי בו מאוחסנים הנתונים והתוכניות שנמחקים עם כיבוי המחשב (ניתוק המתח).
  2. זיכרון המשני, שמשמש כיחידת זיכרון קבועה, נשמרים נתונים ותוכניות לאחר כיבוי המחשב. המידע בזיכרון משני נשמר במדיה מגנטית או אופטית כגון בדיסק קשיח (Hard Disk), בתקליטורים (CD – Compact Disk) ובתקליטונים (דיסקטים).

ליחידות זיכרון יש מאפיינים רבים אולם בשלב נציין רק את המאפיינים החשובים לכתיבת תכנית בשפת סף, והם:

* **גודל של תא זיכרון** – קובע את מספר הסיביות ואת הערך הגדול ביותר שאפשר לאחסן בזיכרון.
* במעבד 8086 גודל של תא זיכרון הוא byte 1 ובו ניתן לאחסן 256 ערכים שונים.
* במידה וצריך לאחסן ערכים גדולים יותר ניתן להשתמש במספר תאי זיכרון על-פי הצורך.
  1. למשל, כדי לאחסן את הערך 1000 נשתמש בשני תאי זיכרון.
* **כתובת של תא בזיכרון** אותה המעבד רושם בפס הכתובות כאשר הוא פונה לקרוא מהזיכרון או לכתוב בזיכרון.
* לכל תא יש כתובת ייחודית הנקראת **כתובת מוחלטת**
* בפס הכתובות של מעבד 8086 יש 20 קווי כתובת ולכן מספר הכתובות השונות שניתן ליצור

220 = 1,048,576

* מפת הזיכרון מתחילה בכתובת 00000h והכתובת האחרונה היא FFFFFh.
* מענים 896K-1M רכיב ROM המכיל את ה- BIOS , תכנה צרובה שמספקת שירותי בדיקה של תקינות החומרה עם העלאת המחשב וממשקים ליחידות חמרה ושירותים לפסיקות תכנה
* מענים 640K-896K רכיב ROM לקריאה בלבד מענים אלו שייכים לרכיבי קלט/פלט שונים כמו מסך
* מענים מ- 0 עד 640K זיכרון לקריאה /כתיבה לשימוש המתכנת, לכאן נטענת התוכנית שאנו מריצים(RAM).

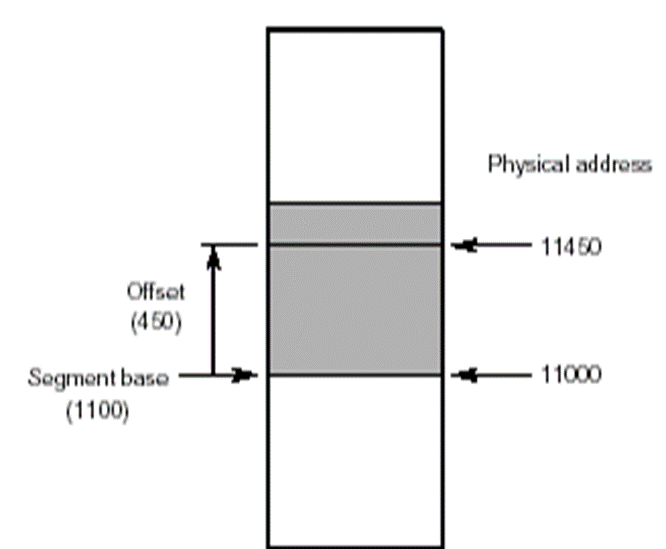
זיכרון RAM

BIOS

זיכרון

ROM

* מעבד 8086 מחלק את מרחב הזיכרון שהוא לינארי לארבעה מקטעים (או סגמנטים segments) שגודל מקסימאלי של כל אחד מהם יכול להיות **64KB** ולכל אחד יש שם וייעוד שונה (ראה אוגרי סגמנט מה תפקידם).



* כתובת של תא בזיכרון מורכבת משני חלקים:

היסט + כתובת תחילת הסגמנט

כתובת תחילת סגמנט שמורה באוגר סגמנט

(CS, DS,..)

כתובת ההיסט באחד מאוגרי ההצבעה ואוגרי האינדקס וגם אוגר נתונים BX

(SI, DI, BX…).

* כתובת של תא בתוך סגמנט נקרא **היסט** (offset) או **כתובת לוגית** ולעיתים גם **כתובת אפקטיבית**, והיא מציינת את מיקומו היחסי של תא מתחילת הסגמנט.

בתוכנית אופרנד שמכיל כתובת מספיק שיציין את ההיסט. כתובת הסגמנט היא בדרך כלל קבועה ונקבעת כאשר תכנית נטענת לזיכרון

לדוגמה, ההיסט (הכתובת הלוגית ) של הוראה היא 0450h

זה התא שמרוחק 450 בתים מתחילת מקטע הקוד.

**קביעת כתובת בסיס של מקטע**

* אוגר מקטע קוד יכול להכיל כתובת שהיא כפולה של 16
* כפולה זו נקראת **פסקה (**paragraph)
* באמצעות אוגר CS ניתן לכסות את כל מרחב הכתובות בזיכרון 1MB בכפולות של 16
  + לדוגמה, כאשר CS=0, הכתובת שהוא מצביע עליה היא 0\*16
  + כאשר CS=1, הכתובת שהוא מצביע עליה היא 1\*16
  + כאשר CS=k, הכתובת שהוא מצביע עליה היא k\*16

**יתרונות שימוש במקטעים**

* תאימות למעבד קודם 8085 שבו מרחב הזיכרון הוא רק 64KB (מקטע אחד מכסה את כל התחום)
* בתוכנית אנו מציינים את ההיסט של תא (רק 16 סיביות) . המעבד מחשב כתובת מוחלטת בעזרת כתובת אוגר המקטע המתאים
* הפרדה בין נתונים והוראות
* אפשרות להריץ אותה התוכנית ובכל פעם לשנות את רק את הנתונים.
* מנגנון זה מאפשר להריץ מספר תוכניות, כאשר לכל תכנית מוקצים סגמנטים נפרדים ,

מנגנון זה מאפשר שיתוף נתונים בין מספר תוכניות.

**גודל מקטע הנתונים**

גודל מקטע הנתונים נקבע על פי מספר המשתנים וגודלם

לדוגמה: הגדרנו 3 בתים,

גודל המקטע יהיה הכפולה הקטנה ביותר של 16 וגדולה מ- 3 ולכן

הגודל של מקטע נתונים יהיה 16 בתים (פסקה אחת) אבל רק שלושה בתים ראשונים יכילו את המשתנים שהגדרנו . יתר 13 הבתים הם "זיכרון מבוזבז".

**תיאור הנחיות אסמבלר**

* הנחיה מס. 1..MODEL SMALL

הנחיה MODEL מתייחסת להגדרת הסגמנטים בזיכרון, והערך SMALL מגדיר תכנית בה יש מקטע קוד אחד ומקטע נתונים אחד.

* שלוש ההנחיות הבאות מגדירות את המקטעים:

הנחיה מס. 2 קובעת את תחילת מקטע המחסנית ואת גודל המחסנית.

. STACK 100h

בתוכנית זו אנו לא משתמשים במחסנית אולם אנו נכלול הנחיה זו בתוכניות שנכתוב, ובהמשך נרחיב על תפקיד המחסנית אופן השימוש וכיצד קובעים את גודלה.

הנחיה מס. 3 מגדירה את תחילת מקטע נתונים

.DATA

הנחיה מס. 7 מציינת את תחילת מקטע קוד הוראות ולמעשה כך נקבע גם סיום מקטע נתונים.

. CODE

**נוכל לראות זאת בדוגמת קוד כאן:**

.**MODEL SMALL**

**.STACK 100h**

**.DATA**

**הגדרת משתנים**

**.CODE**

start:

; אתחול מקטע נתונים

MOV    AX, @DATA

MOV DS, AX

**גוף התוכנית**

;סיום התוכנית והחזרת בקרה למערכת ההפעלה

MOV   AX, 4C00h

INT   21h

**END** start

**הגדרת משתנים במקטע הנתונים:**

* הנחיות מס. 4 עד מס. 6 מגדירות שלושה נתונים שיאוחסנו במקטע הנתונים:
  + משתנה בשם a מטיפוס מילה מוגדר על ידי DW ומאותחל לערך 2.

a DW 2

* + משתנה בשם b מטיפוס מילה מוגדר על ידי DW ומאותחל לערך 3.

b DW 5

* + משתנה בשם sum מטיפוס מילה מוגדר על ידי DW ללא ערך תחילי מצוין ע"י התו '?'.

**sum** DW ?

שורה מס. 8 מתחילה בתווית המסתיימת בנקודתיים

start:

תווית היא שם לוגי לכתובת בזיכרון, והתווית start מציינת את תחילת קטע הוראות התוכנית.

שתי הוראות הביצוע הראשונות, רשומות בשורות 10 ו- 11 ותפקידן לאתחל את אוגר הסגמנט בכתובת הבסיס של מקטע הנתונים:

10. mov ax, @data ; AX= כתובת בסיס של מקטע הנתונים

1. mov ds, ax ; DS = AX

הוראה מס. 10 מעבירה לאוגר AX את מען תחילת סגמנט הנתונים. הסימול @DATA מסמן למהדר (אסמבלר) את מען מקטע נתונים כפי שנקבע בזמן הרצת התוכנית. בזמן הידור ערכו נקבע לאפס ורק בזמן הקישור הוא מקבל את כתובת של סגמנט הנתונים כפי שנקבע בטעינה של תכנית לזיכרון.

הוראת מס. 11 היא הוראת השמה המעתיקה כתובת זו מאוגר AX אל אוגר סגמנט נתונים DS.

**הגדרת גוף תכנית במקטע הקוד**

שורות מס. 13 עד 15, הוראות בשפת סף המגדירות פעולת חיבור a + b והשמת .sumהתוצאה בזיכרון בתא שכתובתו הוא

1. mov ax, a         ; ax = a
2. add ax, b         ; ax = a + b
3. mov    sum, AX       ; sum = ax

סיום התוכנית והחזרת בקרה למערכת ההפעלה

שורות מס. 17 ומס. 18, אותן נרשום בכל תכנית, מאפשרות יציאה מסודרת מהרצה של תכנית והחזרת הבקרה למערכת הפעלה.

17. mov ax, 4C00h

18. int 21h

שורה מס. 19 היא הנחיה האחרונה בתוכנית והיא מציינת לאסמבלר את סוף התוכנית והמהדר לא יתייחס לכל ההוראות שיופיעו לאחר הנחיה .

19. END start

**הגדרת משתנים בשפת סף**

מבנה הגדרה של משתנה הכולל שלושה ארגומנטים:

var-name Data-type init-value

ארגומנט ראשון הוא שם משתנה באורך עד 31 תווים, המתחיל באות ויכול להכיל אותיות, מספרים ותווים מיוחדים '$', '-', '@', '?', '.'.

ארגומנט שני מציין את טיפוס נתונים והוא מתחיל תמיד באות D, שמציינת קיצור של המילה Define , ולאחריו אות נוספת המציינת טיפוס מתוך חמישה טיפוסי נתונים אפשריים.

טיפוסי משתנים במעבד 8086:

DB Byteמקצה בית אחד-

DW Word - מקצה 2 בתים

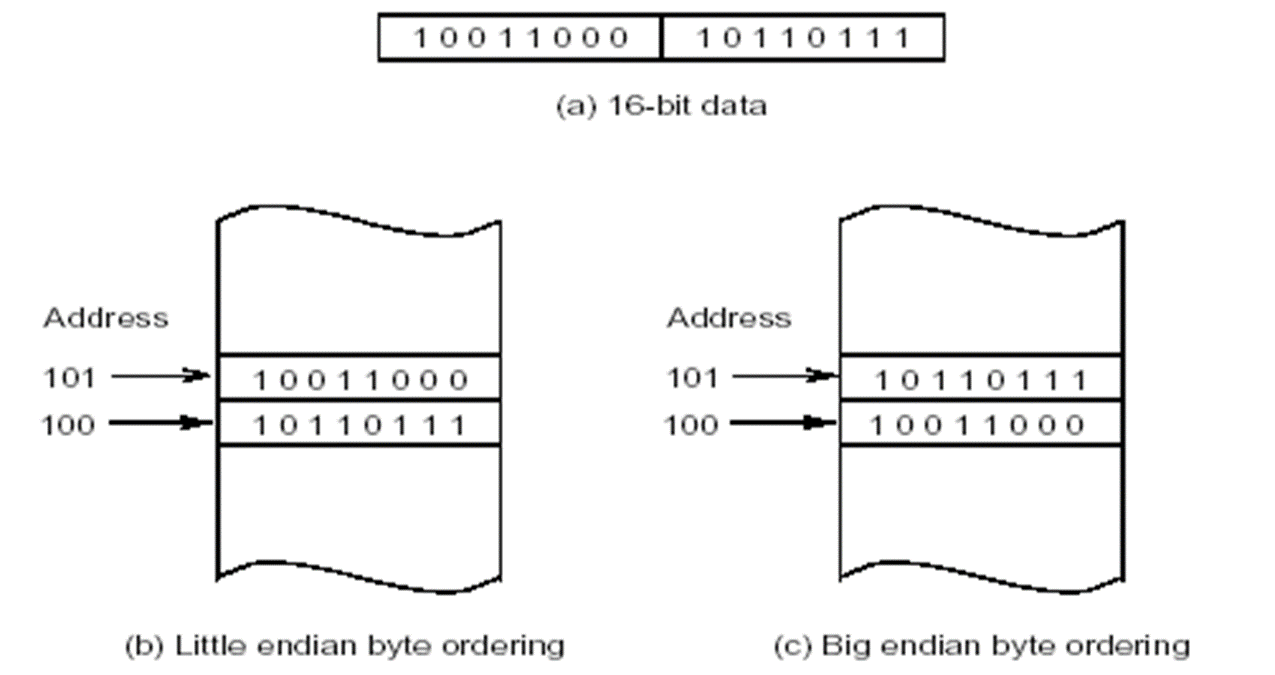
DD Double word - מקצה 4 בתים

DQ Quadra word - מקצה 8 בתים

DT Tera Word – מקצה 10 בתים

**סדר טעינת נתונים מטיפוס מילה לתאי זיכרון מטיפוס בית:**

בית תחתון של המילה לכתובת נמוכה ובית עליון של המילה לכתובת גבוהה



**שיטות מיעון אוגרים:**

**מיידי לאוגר** : העברת מספר לתוך אוגר נקרא מיון מיידי.

MOV AX, 0FFH

**אוגר לאוגר:** העברת המידע שיש באוגר לאוגר אחר

MOV AX,BX

**אוגר לזיכרון:** העברת המידע מאוגר לכתובת

MOV [DI], BX

**זיכרון לאוגר:** העברת המידע מהכתובת של אוגר אל תוך אוגר

MOV BX,[DI]

**סוגי הוראות:**

* הוראות אריתמטיות
* הוראות לוגיות
* הוראות לולאה
* הוראות קפיצה
* הוראות פסיקה
* הוראות הזזה וסיבוב