הגדרות משתנים ופקודת mov

1MB = 1024 \* 1024 ⇨ 1KB \* 1KB

210 \* 210 = 1,048,576 byte

## זיכרון המחשב:

וגודל הזיכרון מחשב שלנו הוא שלנו ◄ 1Mb 220

למחשב שלנו כ - 1,048,576 תאי זיכרון.

נחליט שאנו רוצים להשתמש רק ב 1048 - תאים מתוכם לשמירת נתונים.

איך נזכור את כולן? איך נשתמש בהן בלי להתבלבל?

{220 = 1,048,576 ⇨ זיכרון המחשב}

{210 = 1,048 ⇨ סגמנט הנתונים }

משתנה הוא תא זיכרון אחד או יותר אשר ניתן לשמור בו נתונים.

כמו [קופסה] לשמירת מידע.

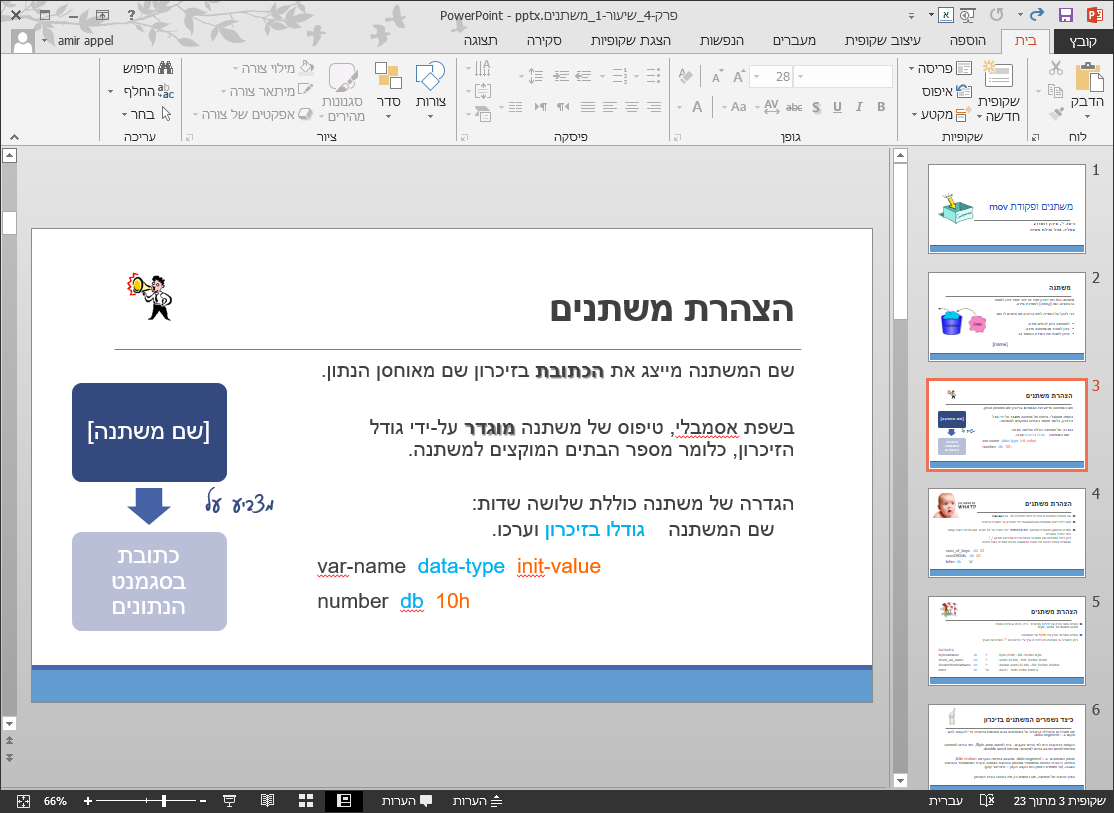
בתחילת התכנית יש להגדיר מראש בתוכנה את המשתנים בהם נשתמש. כדי שהמעבד יכין להם מקום בסגמנט הנתינים. הגדרה זו נקראת הצהרת נתונים.

כדי להקל על הפנייה לתא הזיכרון אנו נותנים לו שם.

* ניתן להשים במשתנה מידע.
* ניתן לאחזר ממנו מידע.
* וניתן לשנות את המידע השמור בו.

## הצהרת משתנים

שם המשתנה מייצג את **הכתובת** בזיכרון שם מאוחסן הנתון.

בשפת אסמבלי, טיפוס של משתנה **מוגדר** על-ידי גודל הזיכרון, כלומר מספר הבתים

המוקצים למשתנה.

הגדרה של משתנה כוללת שלושה שדות:

שם המשתנה גודלו בזיכרון וערכו.

var-name data-type init-value

number db 10h

* את הצהרת המשתנים כותבים תחת הכותרת של **DATASEG**
* נהוג לתת לשם המשתנה שם משמעותי כדי שיצביע על תפקידו בתכנית.
* השדה הראשון בהצהרת השתנה, **שם המשתנה**, יכול להכיל עד 31 תווים,   
  הוא מתחיל באות קטנה ויכול להכיל מספרים.  
  ניתן לתת למשתנה שם המורכב מכמה מילים שבינים הסימן "\_"   
  אפשרות אחרת לכתוב את האות הראשונה במילה השנייה בשם המשתנה באות גדולה.
* num\_of\_boys db 32
* numOfGirls db 42
* letter db 'a'
* השדה השני בהצהרה על משתנים מציין את **טיפוס** הנתונים: בית, מילה או מילה כפולה.  
  Byte, word or double word
* השדה השלישי מציין את **הערך** של המשתנה.   
  ניתן להצהיר על משתנה ולא לתת לו ערך ע"י כתיבה של ? בשדה של הערך.

DATASEG

ByteVarName db ? ; byte (8 bit) - DB: Define Byte

WordVarName dw ? ; word (16 bit) - DW: Define Word

DoubleWordVarName dd ? ; double word (32 bit) - DD: Define Double

letter db ‘A’ ; ASCII code of the letter A

## כיצד נשמרים המשתנים בזיכרון?

אנו מצהירים בתחילת התכנית על המשתנים בהם נשתמש בתכנית כדי להקצות להם מקום ב – data segment.

הקצאת הכתובות היא לפי בתים עוקבים.

בית למשנה מסוג Byte,

שני בתים למשתנה מטיפוס word

וארבע בתים למשתנה מטיפוס double word.

במעבדים של אינטל, מוקצים כמה בתים עוקבים למשתנים מטיפוס בית, שני בתים עוקבים למשתנה מטיפוס מילה, ולטיפוס מילה כפולה מוקצים ארבעה בתים עוקבים.

אחסון המשתנים ב – data segment מתבצע בשיטה הנקראת little endian,   
בשיטה זו הבית הפחות משמעותי מאוחסן בכתובת הנמוכה והבית המשמעותי בכתובת הגבוה. (מי שמופיע ראשון הוא הקצה הקטן – אינדיאני קטן).

אחסון המשתנים ב – data segment מתבצע בשיטה הנקראת little-endian . לפי שיטה זו, הבית הפחות משמעותי מאוחסן בכתובת הנמוכה, והבית המשמעותי של המילה מאוחסן בכתובת העוקבת.

השם little-endian , דהיינו: מי שמופיע ראשון הוא "הקצה הקטן"  
השם נבחר בהומור, שכן הוא מזכיר "אינדיאני קטן" – little Indian.

בציון כתובת של משתנה, אנו רושמים רק את כתובת הבית התחתון

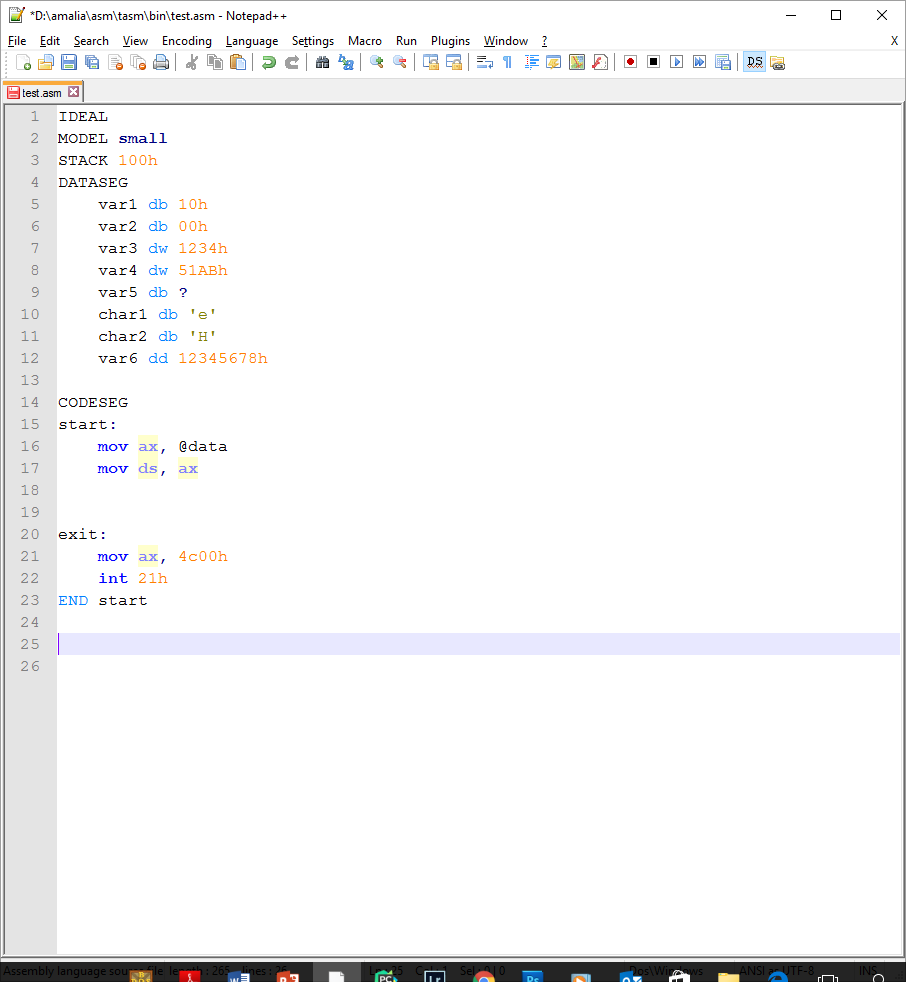
כיצד נשמרים המשתנים בזיכרון?

זיכרון המחשב הוא מערך של תאי זיכרון ממוספרים.

mov ax, @data

mov ds, ax

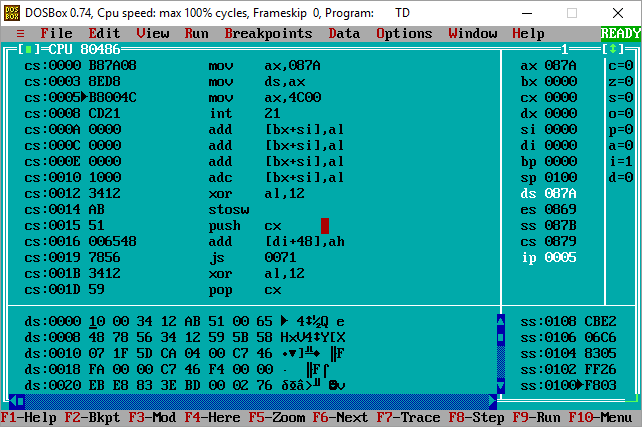
בתחילת התכנית אנו כותבים תמיד את שתי שורות הקוד:

אשר מציינות למחשב היכן מתחיל סגמנט הנתונים.

בתכנית הזו יש רק הצהרה על משתנים:

כיצד נשמרים המשתנים בזיכרון המחשב?

|  |  |
| --- | --- |
| מיקום בזיכרון | הצהרת משתנה |
| ds: 00 | var1 db 10h |
| ds: 01 | var2 db 00h |
| ds: 02 | var3 dw 1234h |
| ds: 04 | var4 dw 51ABh |
| ds: 06 | var5 db ? |
| ds: 07 | char1 db 'e' |
| ds: 08 | char2 db 'H' |
| ds: 09 | var6 dd 12345678h |



## גישה לזיכרון המחשב

DATASEG

num1 db 11h

num2 db 22h

CODESEG

start:

mov ax, @data

mov ds, ax

mov al, [ds:00]

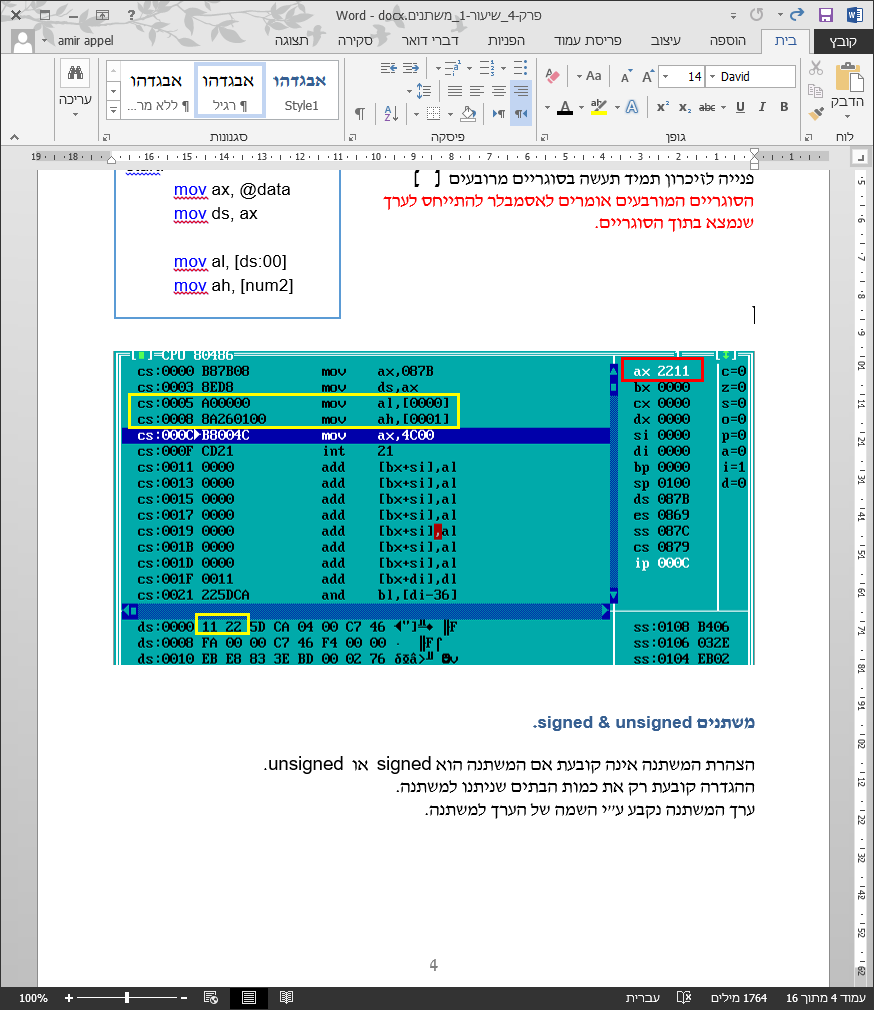
mov ah, [num2]

ניתן לגשת לכל מקום בזיכרון המחשב ע"י פניה לכתובת שלו בסגמנט הנתונים

או פניה לשם המשתנה.

פנייה לזיכרון תמיד תעשה בסוגריים מרובעים **[ ]**

הסוגריים המורבעים אומרים לאסמבלר להתייחס לערך שנמצא בתוך הסוגריים.



## לדלג

## משתנים signed & unsigned.

הצהרת המשתנה אינה קובעת אם המשתנה הוא signed או unsigned.

ההגדרה קובעת רק את כמות הבתים שניתנו למשתנה.

ערך המשתנה נקבע ע"י השמה של הערך למשתנה.

DATASEG

136 = 128+8

136d = 1000 1000b ⇨ 88h

120 = 64+32+16+8

120d = 0111 1000b

-120d = 1000 0111+1 = 1000 1000b ⇨ 88h

var1 db ?

var2 db ?

CODESEG

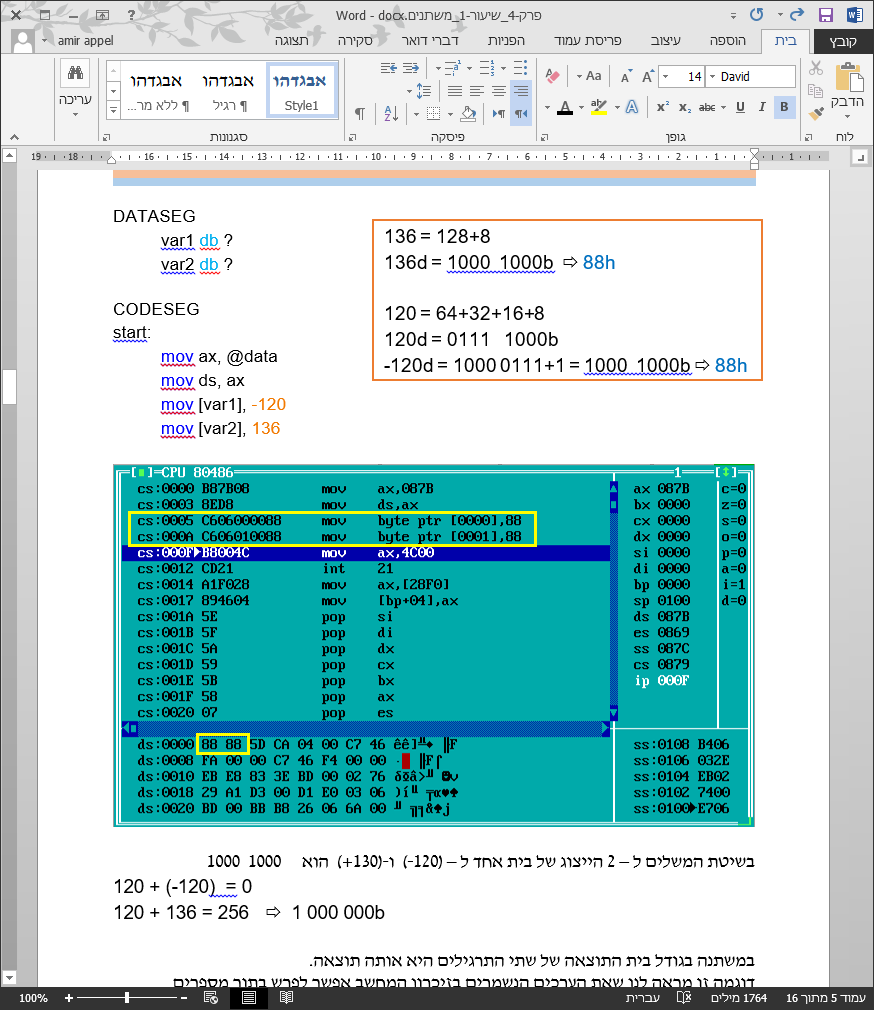
start:

mov ax, @data

mov ds, ax

mov [var1], -120

mov [var2], 136



בשיטת המשלים ל – 2 הייצוג של בית אחד ל – (120-) ו-(130+) הוא 1000 1000

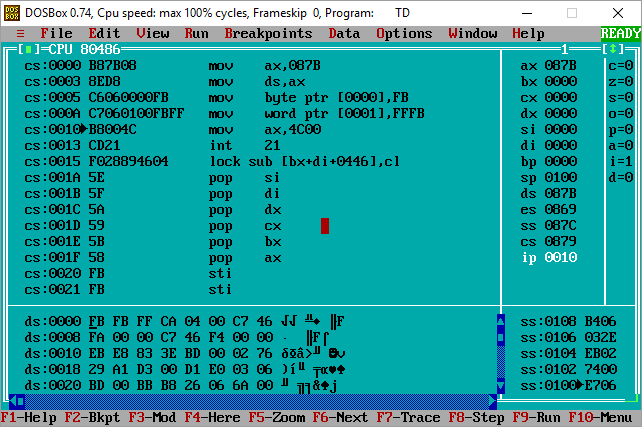
120 + (-120) = 0

120 + 136 = 256 ⇨ 1 000 000b

במשתנה בגודל בית התוצאה של שני המשתנים היא אותה תוצאה.

דוגמה זו מראה לנו שאת הערכים הנשמרים בזיכרון המחשב אפשר לפרש בתור מספרים מסומנים או לא מסומנים. האחריות לפרשנות היא של המשתמש.

ומה קורה בהגדרה של מילה?

DATASEG

var1 db ?

var2 dw ?

CODESEG

start:

mov ax, @data

mov ds, ax

mov [var1], -5

mov [var2], -5

תא זיכרון 00: בגודל בית מכיל את הערך FB

תא זיכרון01: בגודל מילה, מיכל את הערך FF FB

מחרוזת

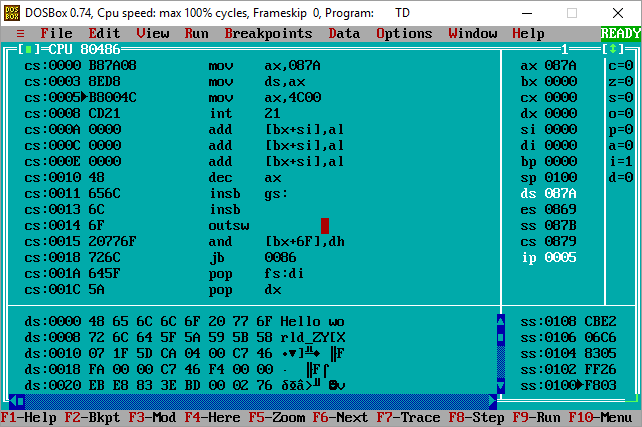
ניתן להגדיר משתנה בגודל בית ולשים בו אוסף של תווים.

DATASEG

string db 'Hello world'

האסמבלר יודע להתייחס להגדרה זו כאל מחרוזת של 11 תווי ascii שכל אחד מהם הוא בגודל בית.

למעשה האסמבלר מתייחס להגדרה זו כאילו הגדרנו 11 תווים שמרנו אותם בזיכרון בזה אח זה.



## פקודת mov

mov - קיצור של move

הפקודה מקבלת שני אופרנדים (operands)- מקור ויעד ומעתיקה את המקור ליעד.

יוצרת העתק של source (מקור) בתוך destination (יעד)

איננה משנה את source

mov destination**,** source

mov מקור **,**יעד

mov register, register

mov register, constant

mov register, memory

mov memory, register

mov memory, constant

לא ניתן להעביר נתונים מזיכרון לזיכרון.

גודל הרגיסטרים והמשתנים חייב להיות דומה זהה. אחרת נקבל הודעת שגיאה.

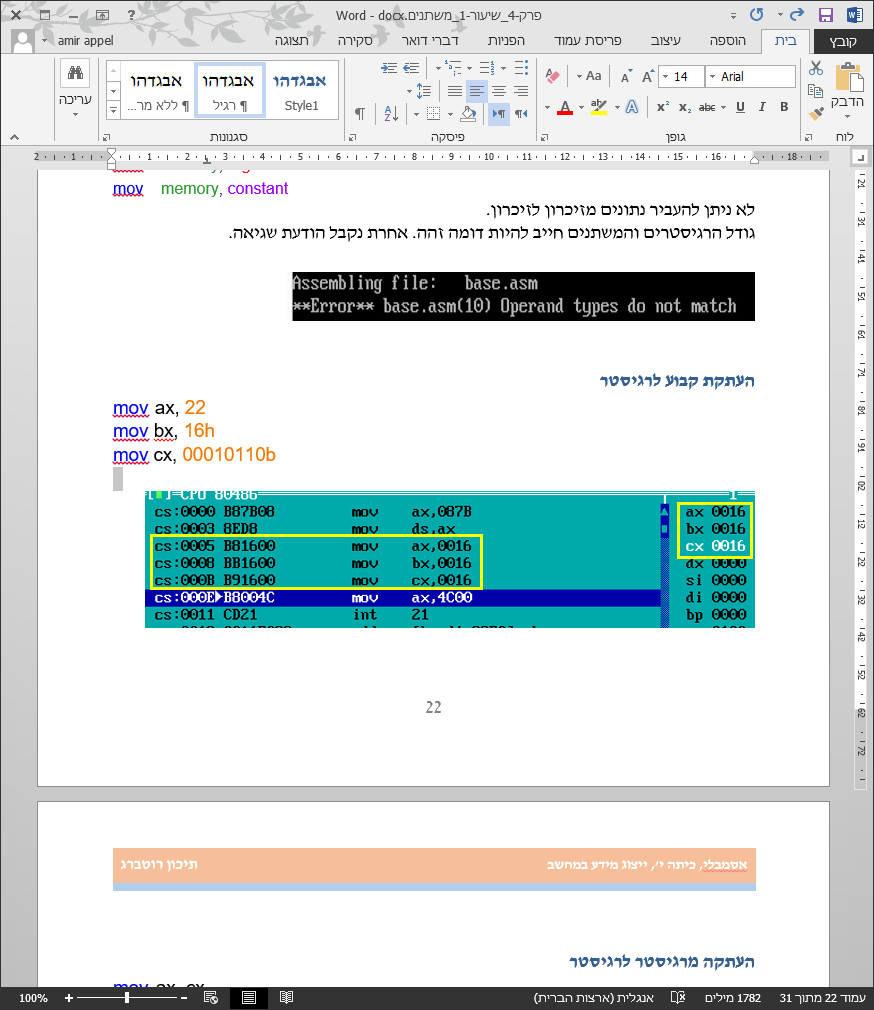


העתקת קבוע לרגיסטר

mov ax, 22

mov bx, 16h

mov cx, 00010110b



## העתקה מרגיסטר לרגיסטר

mov ax, cx

mov ax, bx

mov cx, bx

mov cl, ah

העתקה של תא בזיכרון ל רגיסטר

mov register, [memory]

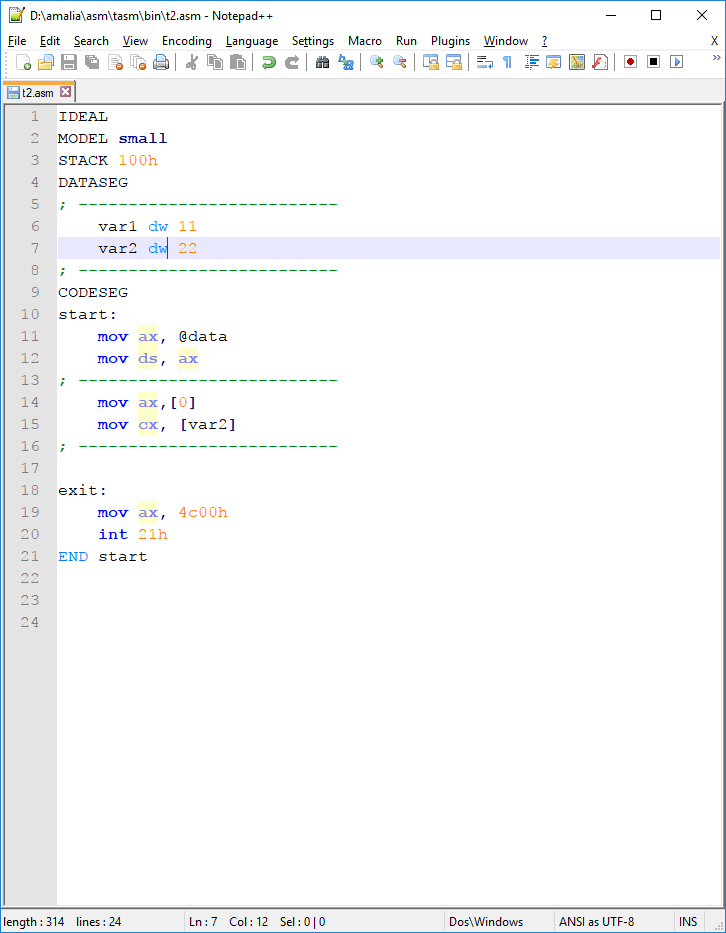
mov ax, [0]

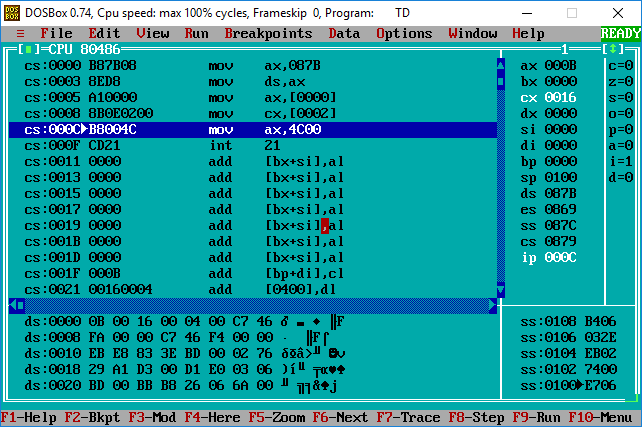
[0] כתובת ישירה בסגמנט הנתונים

העתקת הערך של הערך בכתובת 00 בסגמנט הנתונים ds (var1) לרגיסטר ax.

mov cx, [Var]

העתקת הערך של המשתנה בשם var לרגיסטר cx.





העתקה של רגיסטר לתא בזיכרון.

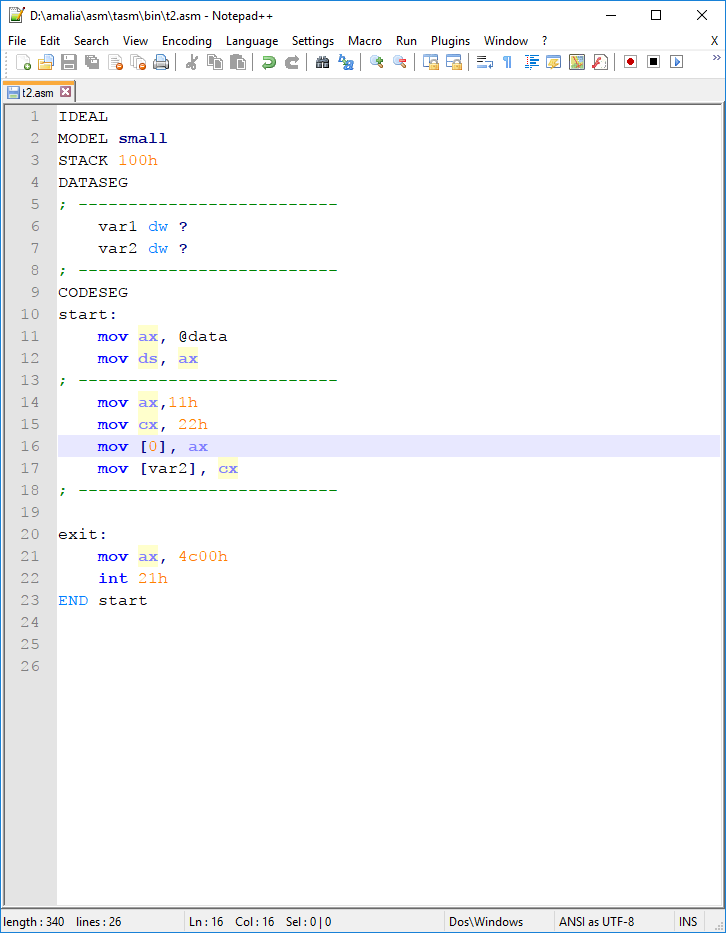
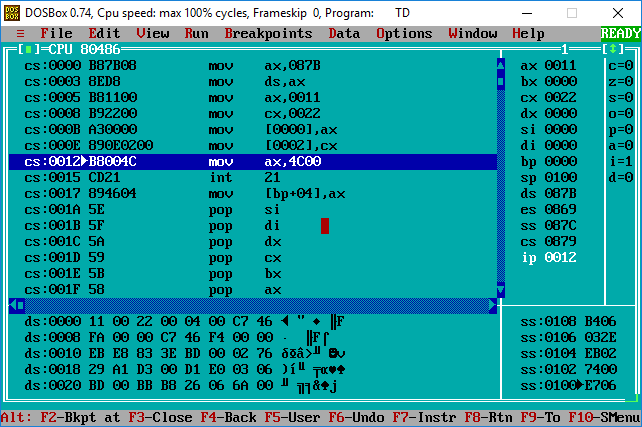
mov [memory], register

mov [0], ax

ax יועתק לתא הזיכרון בכתובת 0

mov [var2], cx,

ax יועתק למשתנה בשם var



mov [bx], ax

mov [bx+1], cx

ax יועתק לכתבות בזיכרון שרגיסטר bx מצביע עליו.

ax יועתק לכתובת בזיכרון שרגיסטר bx מצביע עליו +1.

DATASEG

var dw ?

CODESEG

המשתנה var נמצא בכתובת 00.

ax, מקבל את הערך 10h ⇦ 16d

cx, מקבל את הערך 100h ⇦ 160d

bx, מקבל את העדך 0

[תא זיכרון] בכתובת 00 מקבל את הערך 100h

[תא זיכרון] בכתובת 02 מקבל את הערך 10h

start:

mov ax, @data

mov ds, ax

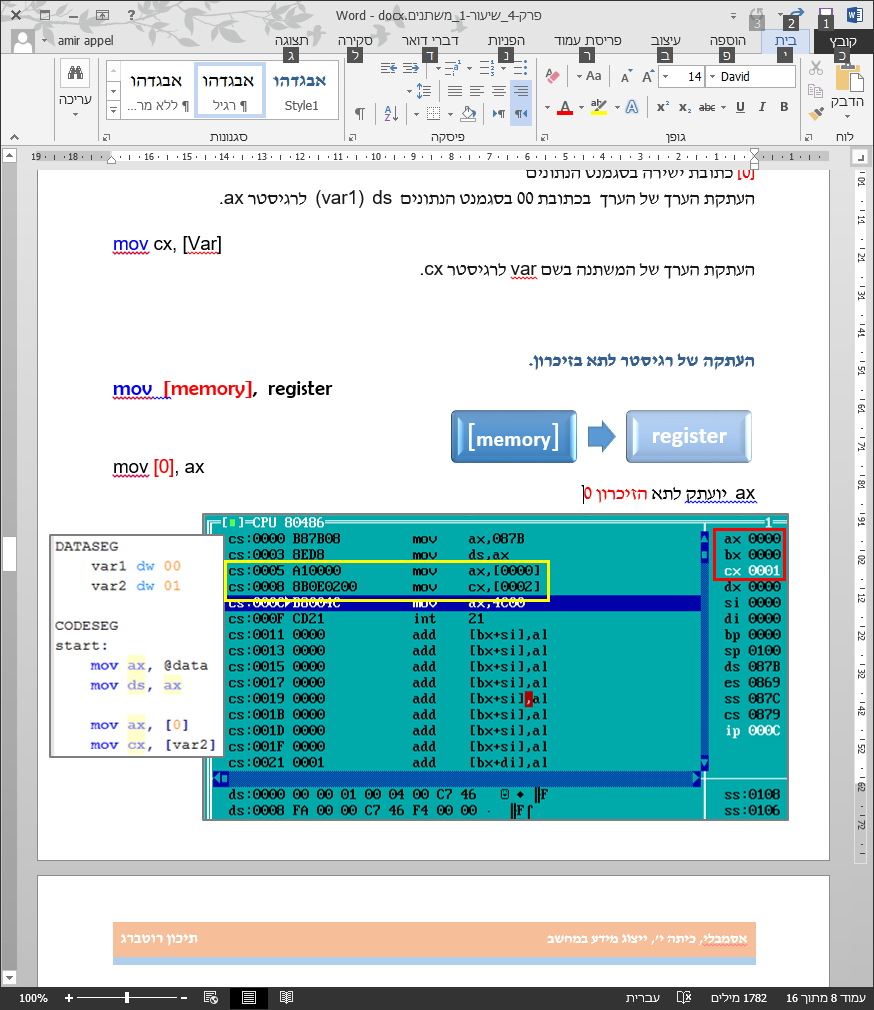
mov ax, 10h

mov cx, 100h

mov bx, 0

mov [bx], cx

mov [bx+2], ax



תרגול

הזזת השחקן על פני לוח המשחק.

## bg3

שמרו את התכנית שיצרתם בשיעור הקודם בשם חדש והמשיכו לעבוד בקובץ החדש.

זכרו כדי שנוכל להריץ את הקובץ ב – DosBox עליכם לשמור את הקובץ בדיסק c

**C:\Heights\PortableApps\TASM\BIN**

בשיעור הקודם למדנו למקם את הסמן על פני המסך על ציר x וציר ה – y.

רגיסטר dl קובע את המיקום של הסמן על ציר ה – x

רגיסטר dh קובע את המיקום של הסמן על ציר ה – y

; print chararcter

; set cursore location

mov dh,6 ; row

mov dl, 7 ; column

mov bh, 0 ; page number

mov ah, 2

int 10h

; draw smiley - ascii 2 at cursor position

mov ah, 9

mov al, 2 ; aL = character to display

mov bx, 00Eh ; bh = Background bl = Foreground

mov cx, 1 ; cx = number of times to write character

int 10h

כדי שבהמשך נוכל להזיז את הדמות נרצה לשנות על הקבועים המשפעים על המיקום של הסמן.

לכן נקבע 3 משתנים:

מיקום על ציר x , מיקום על ציר y וצבע השחקן.

**במקטע הנתונים -** DATASEG נוסיף הצהרה על המשתנים:

; character cordinates

x\_cord db 7 ; column

שימו לב 🎔 , גדלי המשתנים, הם בהתאם לרגיסטרים של הפסיקה

y\_cord db 6 ; row

color dw 0Eh ; color

**נשנה את הפסיקות למיקום הסמן ולציור השחקן כך שנעשה שימוש במשתנים.**

; print chararcter

; set cursore location

mov dh, [y\_cord] ; row

mov dl, [x\_cord] ; column

mov bh, 0 ; page number

mov ah, 2

int 10h

; draw smiley - ascii 2 at cursor position

mov ah, 9

mov al, 2 ; aL = character to display

mov bx, [color] ; bh = Background bl = Foreground

mov cx, 1 ; cx = number of times to write character

int 10h

הריצו את התכנית ובדקו שאין לכם טעויות.

כדי להזיז את הדמות נמחק אותה ע"י ציור הדמות בצבע שחור color = 0 ונצייר אותה במיקום החדש בצבע.

קטע הקוד הבא, מוחק את הדמות ומצייר אותה במיקום חדש.

ושוב כדי להשאיר את המסך פתוח הוספנו פסיקה להמתנה למקש.

מהו מיקום הדמות החדשה על ציר x?

ומהו מיקום הדמות החדשה על ציר y?

;============ On key press, move character ===========

mov [color], 0 ; black color

"מחיקת הדמות" ציור הדמות בשחור במיקום הסמן .

; draw smiley - ascii 2 at cursor position

mov ah, 9

mov al, 2 ; aL = character to display

mov bx, [color] ; bh = Background bl = Foreground

mov cx, 1 ; cx = number of times to write character

int 10h

mov [y\_oord], 19

השמת ערכים חדשים לציור הדמות החדשה במיקום ובצבע חדש.

mov [x\_cord] , 5

mov [color], 4 ; red color

; print chararcter

; set cursore location

mov dh, [y\_cord] ; row

הזזת הסמן למיקום החדש וציור הדמות במיקום החדש

mov dl, [x\_cord] ; column

mov bh, 0 ; page number

mov ah, 2

int 10h

; draw smiley - ascii 2 at cursor position

mov ah, 9

mov al, 2 ; aL = character to display

mov bx, [color] ; bh = Background bl = Foreground

mov cx, 1 ; cx = number of times to write character

int 10h

; waits for character

mov ah, 0h

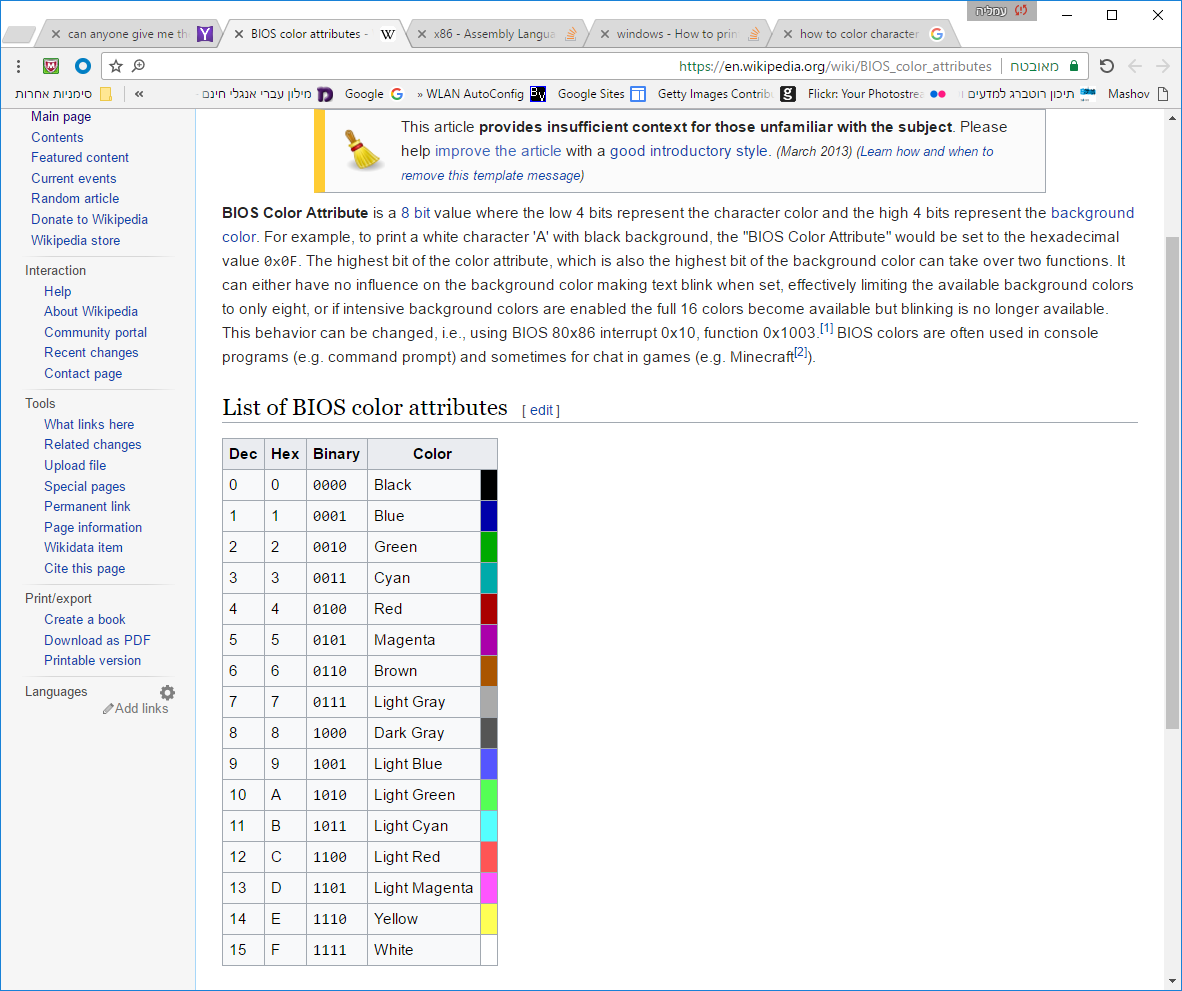
המתנה למקש

int 16h

הריצו את התכנית ובדקו את התוצאה.

האם בהקשת מקש הופיעה דמות צהובה על המסך, ובהקשה על המקש הבא הדמות נמחקה והופיעה במקום אחר בצבע אדום?

**ציור השחקן במיקומים שונים על המסך.**

1. הוסיפו את הקוד לתכנית שלכם.

לאחר ציור הדמות במיקום הראשון, ולפני שינוי התצוגה לתצוגת טקסט ויציאה מהתכנית הגרפית.

1. מחקו את הדמות וציירו אותה בצד שמאל למטה של מסך המשחק, בצבע כחול בהיר

[x\_cord] = 7, [y\_cord] = 18 cvhr, [color] = 09

1. מחקו את הדמות וציירו אותה בצד ימין למטה של מסך המשחק, בצבע ירוק בהיר.

[x\_cord] = 25, [y\_cord] = 17, [color] = 10

1. הוסיפו לתכנית שלכם הוראות שבהקשה הבאה על מקש, נוסיף למסך דמות חדשה בצורה ובצבע שונה, ללא מחיקת הדמות הקודמת.

### תווי אסקי:

