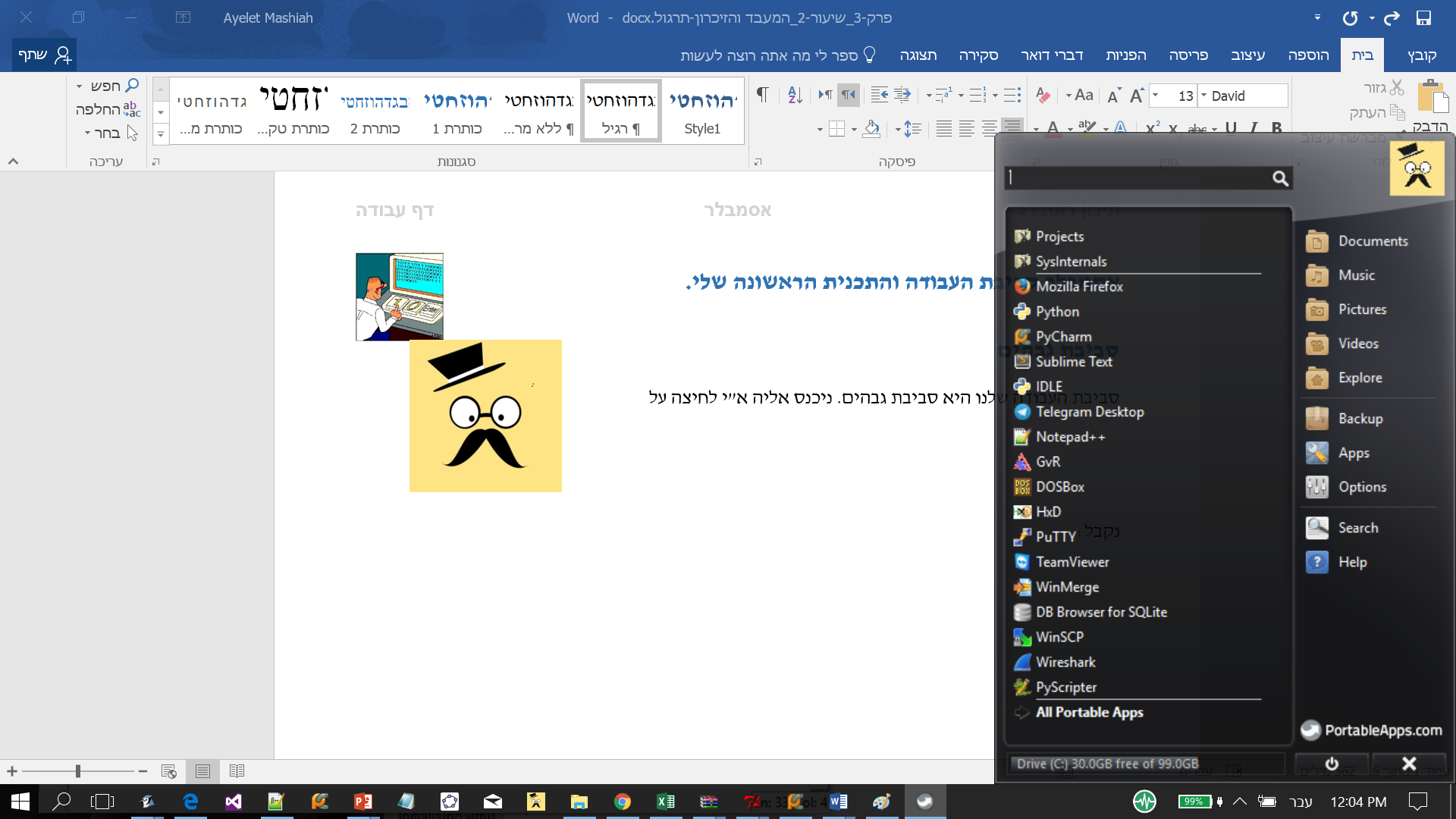
# הרגיסטרים והזיכרון

## אסמבלר סביבת העבודה - תזכורת.

סביבת גבהים

סביבת העבודה שלנו היא סביבת גבהים. ניכנס אליה ע"י לחיצה על

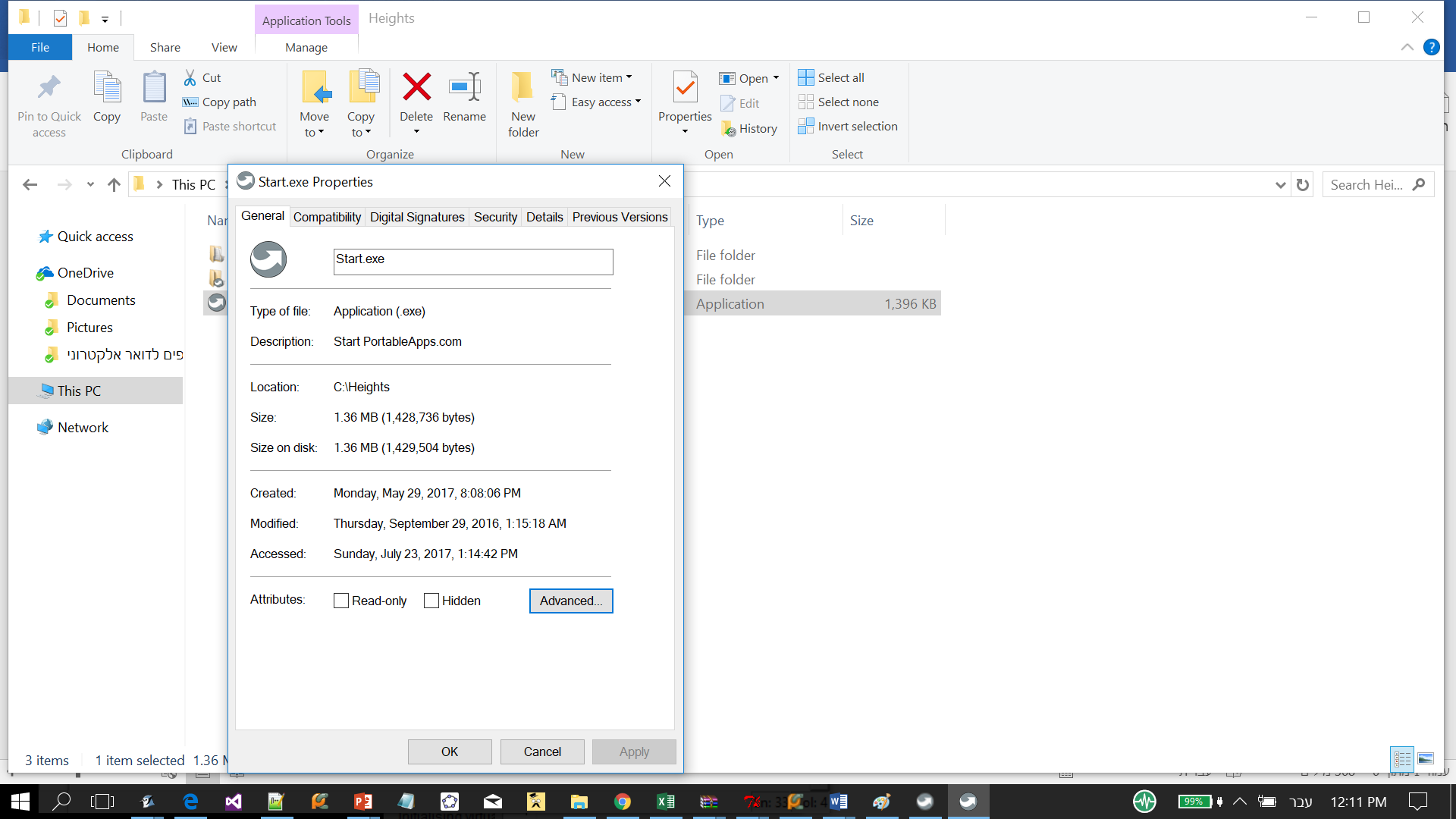
נקבל על המסך את החלון:

בחלון זה נמצאים הכלים בהם

נשתמש:

* Noteped ++
* DosBox

לאחר שהעלנו את החלון בפעם הראשונה, נוכל לגשת אליו באמצעות החץ העגול הנמצא בתחתית המסך



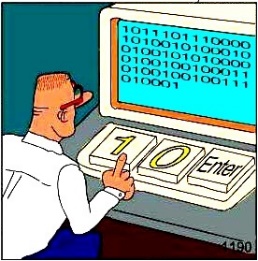
פקודות mov

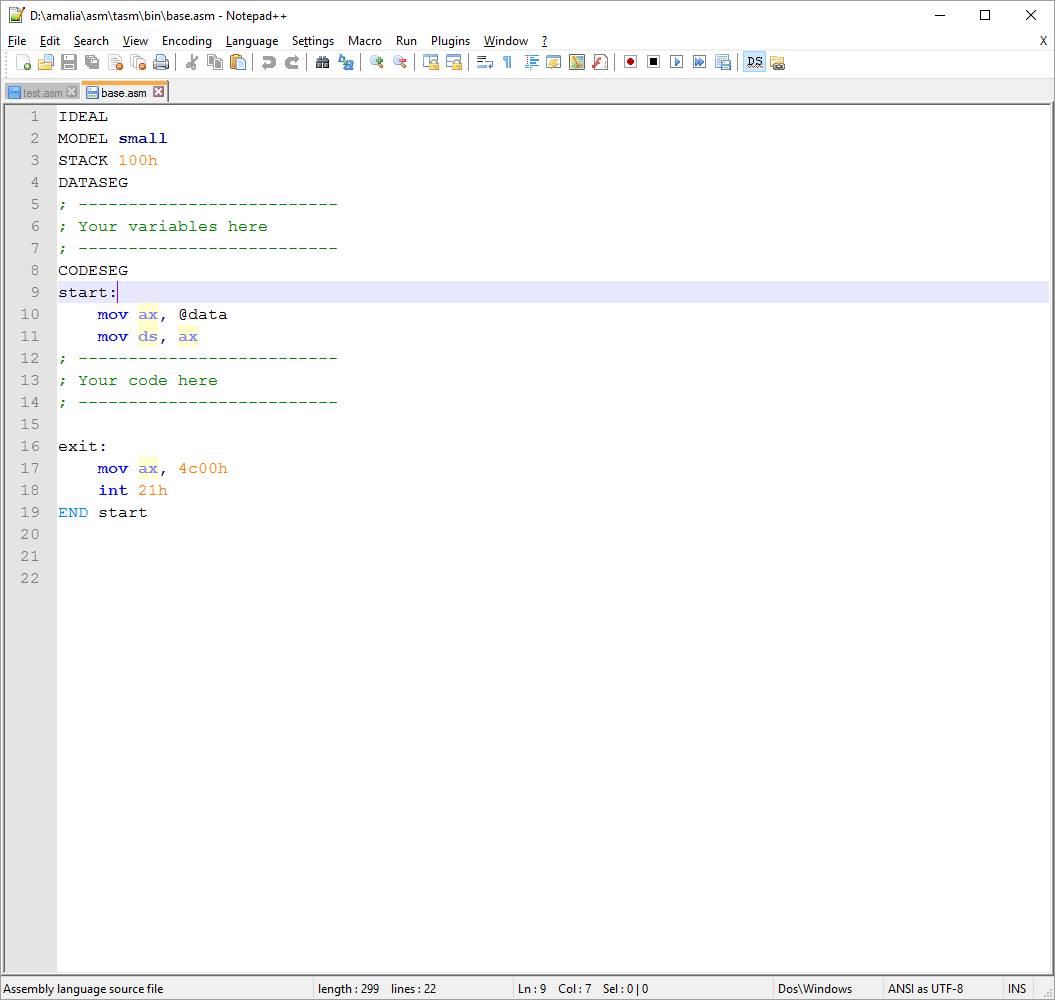
פקודת **mov**, הינה פקודה המשימה ערכים לרגיסטרים או לתאי זיכרון.

אנו זוכרים שתאי הזיכרון במחשב מחולקים לתאים עבור הקוד שאנו כותבים, סגמנט הפקודות – codesegment, ולתאים עבור המשתנים, סגמנט הנתונים – datasegment.

גם בדיבאגר כאשר אנו פותחים את תצוגת ה – cpu אנו רואים את המידע בחלוקה לאזור הקוד – code segment , אזור הרגיסטרים

ואזור המשתנים Data segment.



פתחו את Noatped++

התחל

gvahim

Noteped++

העלו את הקובץ ⇦ base.asm

ב Notepad++ בחרו: פתח ⇦ קובץ ⇦ ונווטו לתיקייה:

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

שמרו את הקובץ בשם **mov2**

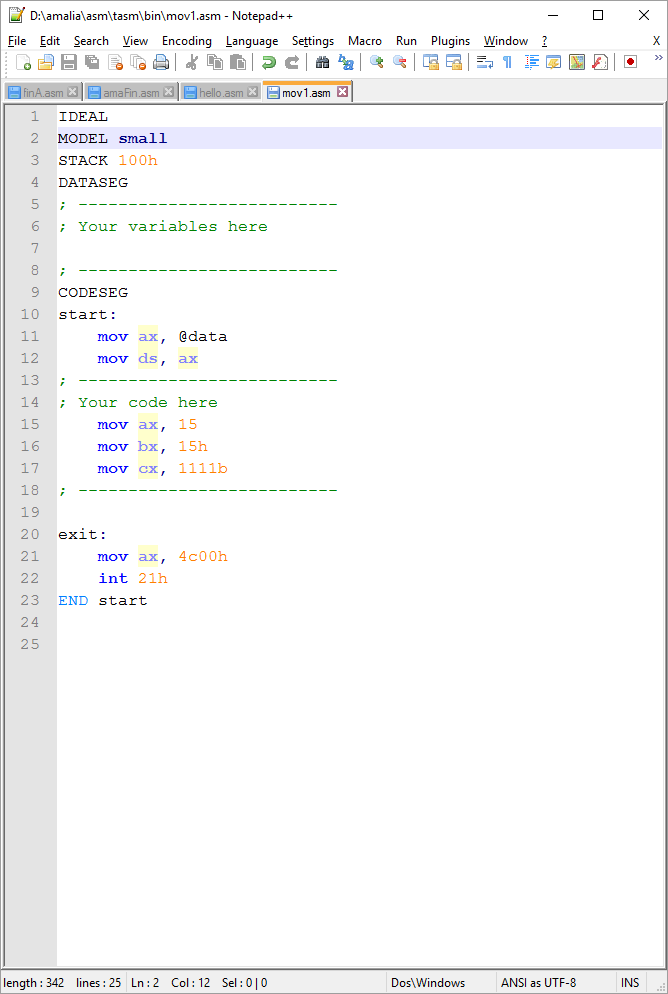
שם חוקי להרצה הוא שם בעל לא יותר מ - 8 אותות וסימנים לועזיים המתחיל באות.

**בחרו בסיומת - .asm**

הקלידו את התוכנית הבאה בסגמנט הנתונים

נשים ערכים לרגיסטרים השונים ונבדוק את התוצאה בתוך ה – TD.

כתבו:



mov ax, 15

mov bx, 15h

mov cx, 1111b

**שמרו את התכנית.**

כדי שתוכלו להריץ את התוכנית עליכם לשמור אותה גם ב-

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

**הריצו את התוכנית:**

נעלה מתוך גבהים את DOSBox

זהו אמולטור של מערכת הפעלה Disk Operating System של חברת Microsoft.

**הריצו את הקומפיילר:**

**tasm /zi mov2**

אם קיבלת הודעת שגיאה בדקו באיזו שורה השגיאה ותקנו. אם קיבלתם הודעה שהתכנית תקינה, **הריצו את הלינקר.**

**tlink /v mov2**

**הריצו את הטורבו דיבאגר:**

**td mov2**

Alt+enter – עובר למצב של תצוגת ה – TD למסך מלא וחזרה.

להריץ פקודה, פקודה את ההוראות ⇦ יש להקיש F8.

בסוף ההרצה בדקו את מצב הרגיסטרים:

מעבר לתצוגת cpu ⇦ Alt + v

בתפריט שעולה נבחר ב – Cpu

תשובה

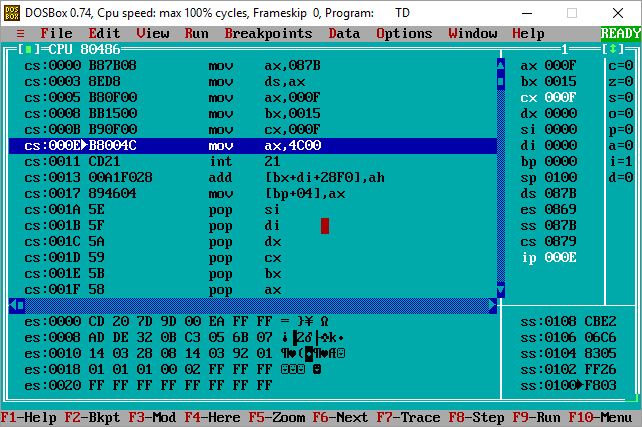
1510 ⇨ F16

15h ⇨ 1516

1111b ⇨ F16 (1510)

נבדוק מהם הערכים ברגיסטרים?

* מדע רגיסטר ax קיבל את הערך 0F?
* מדוע רגיסטר bx, קיבל את הערך 15?
* ומדוע רגיסטר cx קיבל את הערך 0f?



סגמנט הקוד

סגמנט הזיכרון

רגיסטריםם

**המסך בנוי כך:**

חלון ימני עליו: סגמנט הקוד – code segment.

**cs:** 0000 (סגמנט הקוד ומספר הפקודה)

חלון אמצעי: רגיסטרים

חלון ימני תחתון: סגמנט הנתונים – data segment

פעולת ההשמה של הערכים לרגיסטרים השונים ניתנה בבסיסי ספירה שונים:

* ax קיבל את הערך 1510, והציג את הערך בבסיס 16.
* bx קיבל את הערך 1516, (h מייצג מספר הקסדצימאלי), והציג את הערך בבסיס 16.
* bx קיבל את הערך 1510, (b מייצג מספר בינארי), והציג את הערך בבסיס 16.

ליציאה מה - turbo debugger הקישו **Alt+x**.

חזרו לתכנית שכתבתם ב – Noteped++

שנו את פעולות ההשמה של הקבועים לרגיסטרים:

mov ax, 1234h

mov bx, 4321h

mov cx, 0abcdh

mov dx, 0dcbah

שימו לב 🎔, מספר בבסיס 16 שמתחיל באות יקבל 0 לפני האות הראשונה.

**dx:**

**cx:**

**bx:**

**ax:**

tasm /zi mov2

tlink /v mov2

td mov2

שמרו את התכנית והריצו אותה שוב.

איזה ערכים קיבלו הרגיסטרים?

להריץ פקודה, פקודה את ההוראות ⇦ יש להקיש F8.

בסוף ההרצה בדקו את מצב הרגיסטרים:

עברו לתצוגת המעבד של המחשב ⇦ **Alt + v**

בתפריט שעולה בחרו ב – **Cpu**

נעביר את המידע שברגיסטרים לתאי זיכרון שבסגמנט הנתונים.

תא זיכרון מציינים במיקום התא בסגמנט הנתונים בתוך סוגריים מרובעים.

למשל [00] יפנה לתא הזיכרון שמיקומו 0.

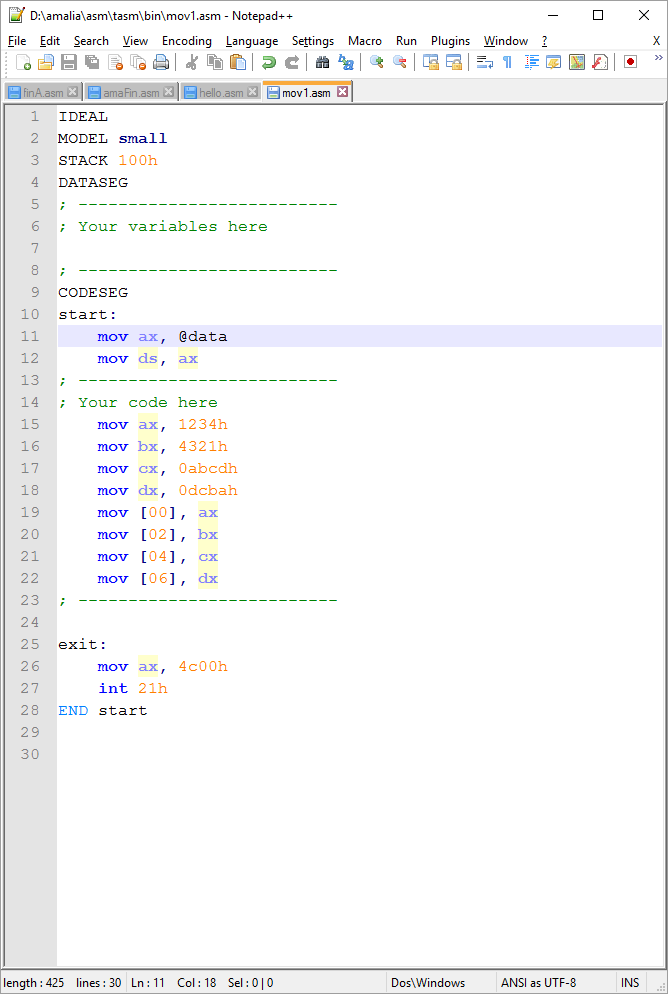
התא הזיכרון הבא יהיה [01] .

מאחר והרגיסטרים שלנו הם רגיסטרים בני 16 ביטים נשמור את ax לתאים [00] [01], אבל נפנה רק לתא הזיכרון הראשון.

נשמור את bx ל- [02] [03], אבל נפנה רק לתא הזיכרון [02].

נשמור את cx ל- [04] [05], אבל נפנה רק לתא הזיכרון [04].

נשמור את dx ל- [06] [07], אבל נפנה רק לתא הזיכרון [06].

חזרו ל – Notped++ **והוסיפו** את הפקודות הבאות:

mov [00], ax

mov [02], bx

mov [04], cx

mov [06], dx

**הריצו שוב את התכנית.**

וב –td הריצו שוב פקודה, פקודה

בעזרת מקש **F8**.

**בדקו את הרגיסטרים ואת סגמנט הנתונים:**

כדי לראות את המידע המאוחסן בזיכרון המחשב עלינו לעבור ב – td למקטע הנתונים הנמצא בחלקו התחתון של המסך בעזרת מקש – Tab. (היקוש Tab 4 פעמים)

צירוף המקשים **Ctrl + g** יפתח חלון עזר בו נבקש מה – td להציג את המידע במקטע הנתונים Data Segment מתא הזיכרון הראשון.

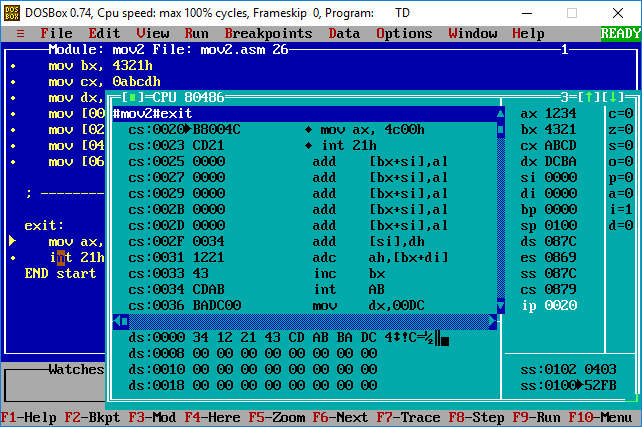
בחלון נקיש **ds:0** ו - **Enter**

את התצוגה של מקטע הנתונים בצעו לאחר שתי הפקודות הראשונות של התכנית

בדקו את המידע ברגיסטר ax ובתאים [00] [01].

ברגיסטר ax שמרנו את המספר 1234h בסגמנט הנתונים המספר שמור 34 – בתא [00]

ו – 12 בתא [01]. המספר בזיכרון שמור "כאילו הפוך" מהמספר ברגיסטרים.



**הסיבה לכך היא שהמידע ברגיסטרים שמור בבתים. כל רגיסטר בנוי משני בתים.**



רגיסטר ax בנוי משני בתים ויכול לשמש גם כשני רגיסטרים נפרדים al ו – ah.

הערכים הנמוכים של הרגיסטר (8 בתים ימניים) נקראים al, a low.

הערכים הגבוהים של הרגיסטר (8 בתים שמאלים) נקראים ah, high a.

## חלוקה לרגיסטרים בני 8 ביט

|  |  |  |
| --- | --- | --- |
| **16 bit** | **8 bit** | **8 bit** |
| AX | AH | AL |
| BX | BH | BL |
| CX | CH | CL |
| DX | DH | DL |

לעיתים נרצה להשתמש ברגיסטר של 8 ביט בלבד.

הרגיסטרים AX, BX, CX, DX הם בגודל 16 ביט כדי לאפשר גמישות לטיפול בבתים בודדים, יש חלוקה משנית לרגיסטרים בגודל 8 ביט

רגיסטר al עבר לתא [00] ורגיסטר ah עבר לתא [01].

לכן המידע ברגיסטר 1234 ⇦ נשמר בזיכרון כ - 3412

**byte Low order** ייכתב בכתובת הנמוכה יותר

**High order byte**  ייכתב בכתובת **הגבוהה** יותר.

אם נסתכל בזיכרון משמאל לימין בעצם המספר יראה הפוך

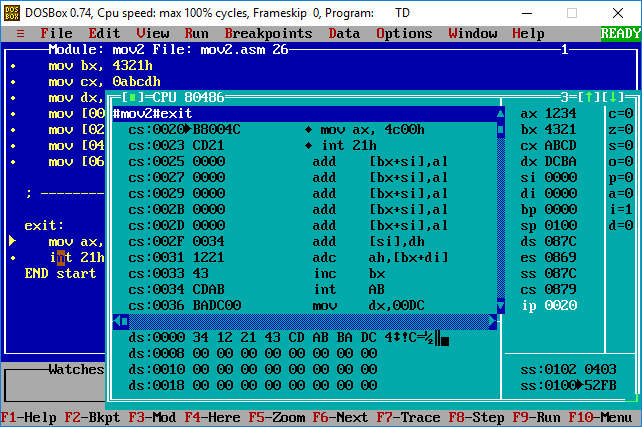
לדוגמא:  **1234** ⇦ [34] [12]



השיטה של שמירת הנתונים בזיכרון נקראת **Little Ending.**   
המידע הפחות משמעותי בכתובת הנמוכה, המידע היותר משמעותי בכתובת הגבוהה.

מתחילים בכתובת הנמוכה.

**מספרים בזיכרון תמיד יראו "כאילו הפוכים"**



בדקו את העתקת המידע מרגיסטרים bx, cx, ו – dx לתאי הזיכרון.

האם אתם מבינים את הצורה בה נשמר המידע?

איזה מידע שמור בתא זיכרון [02], מאיזה רגיסטר המידע הועבר לזיכרון?

איזה מידע שמור בתא זיכרון [03], מאיזה רגיסטר המידע העובר לזיכרון?

איזה רגיסטר שמור בתא זיכרון [04]? מהו המידע השמור בו?

איזה מידע שמור בתא זיכרון [06]? מאיזה רגיסטר המידע העובר לזיכרון?

איזה רגיסטר שמור בתא זיכרון [05]? מהו המידע השמור בו?

|  |  |  |
| --- | --- | --- |
| תא זיכרון | ערך | מאיזה רגיסטר הועבר המידע |
| [2] |  |  |
| [3] |  |  |
| [4] |  |  |
| [5] |  |  |
| [6] |  |  |

כאשר מעתיקים מידע מרגיסטר לרגיסטר חייבים ששני הרגיסטרים יהיו בעלי אותו גודל.

לדוגמה:

ההוראה mov al, bh תקינה,

וההוראה mov bl, ax שגוייה.

**הוסיפו** את הפקודות הבאות לתכנית:

* העבירו את המידע מתא הזיכרון [03] לרגיסטר dl.
* העבירו אתה מידע מתא זיכרון [04] לרגיסטר dh.

מה מראה רגיסטר dx? הריצו ובדקו.

**dx:**

* העבירו את המידע מתא הזיכרון [00] לרגיסטר ah.
* העבירו אתה מידע מתא זיכרון [01] לרגיסטר al.

מה מראה רגיסטר ax? הריצו ובדקו.

**ax:**

* העבירו את המידע מרגיסטר cx לרגיסטר dx.
* העבירו את המידע מרגיסטר dx לתא זיכרון [08].  
  איזה ערך קיבלת תא זיכרון [09]?

**[ ] [ ]**