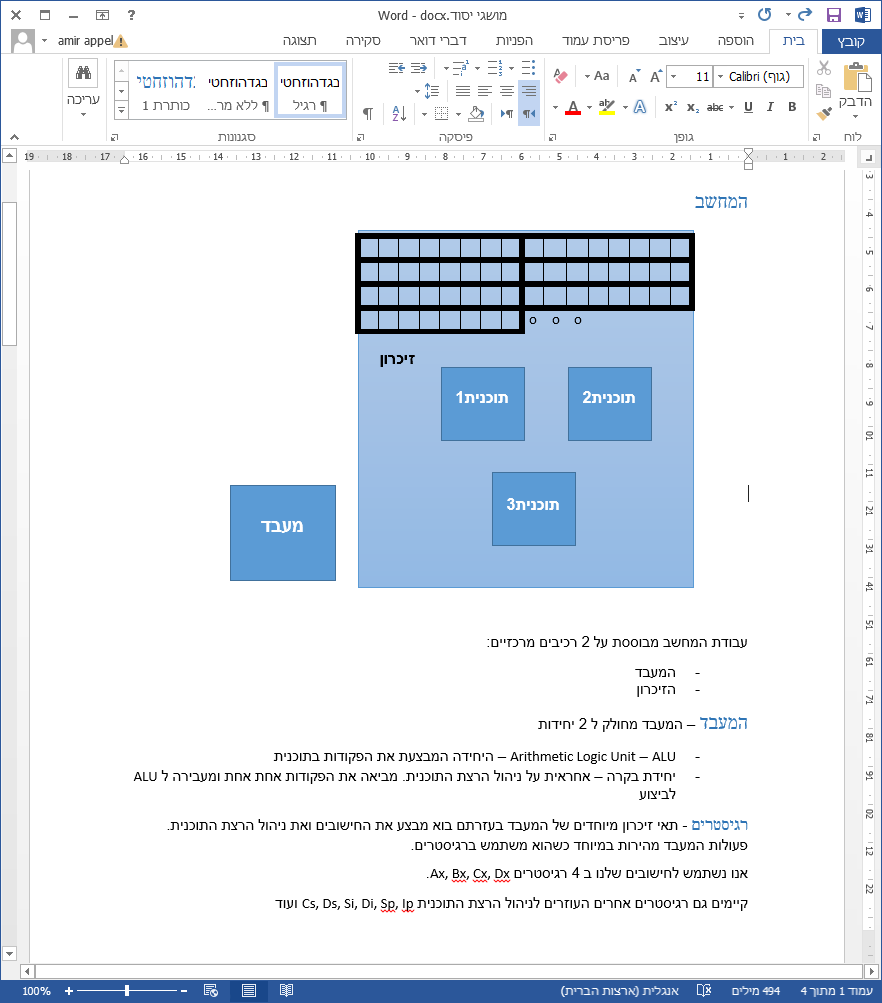
המחשב



# המחשב

עבודת המחשב מבוססת על 2 רכיבים מרכזיים:

* המעבד
* הזיכרון

**המעבד מחולק ל 2 יחידות:**

**ALU –** **Arithmetic Logic Unit,** היחידה המבצעת את הפקודות בתוכנית

**יחידת בקרה** – אחראית על ניהול הרצת התוכנית. מביאה את הפקודות אחת אחת ומעבירה ל ALU לביצוע

# המעבד

בהתאם לארכיטקטורת פון נוימן הפקודות אותן מקבל המעבד מאפשרות לו:

* **קריאת מידע** מהזיכרון או מהתקנים שונים,
* **ביצוע פעולות** חשבוניות ולוגיות על מידע זה
* **וכתיבת תוצאות** החישוב בחזרה לזיכרון או לחלופין שליחתו להתקנים חיצוניים.

יחידת העיבוד המרכזית של המעבד מתקשרת עם הזיכרון ועם יחידות הקלט / פלט באמצעות קווי תקשורת שנקראים Bus, או בעברית פסים.

ישנם שלושה פסים עיקריים ולכל אחד תפקיד ייחודי, פס הנתונים (Data Bus), פס המענים (Address Bus), ופס הבקרה (control bus).

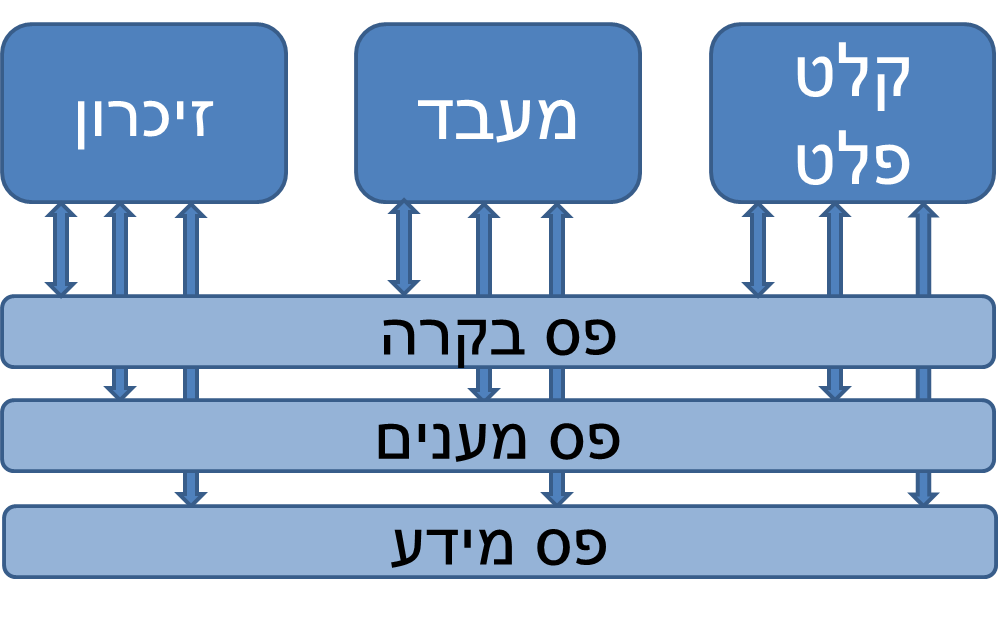
**פס הנתונים** משמש להעברת נתונים בין המרכיבים השונים של המחשב.  
מעבד שיש לו פס נתונים רחב יותר יוכל לכתוב ולקרוא מהזיכרון בקצב מהיר יותר ולכן יעבוד מהר יותר ממעבד זהה שיש לו פס נתונים עם פחות קווים.

**פס המענים** מעביר מידע בין אזור מוגדר בזיכרון (או ביחידת I/O) לבין המעבד.  
כאשר התוכנה רוצה לפנות למקום כלשהו בזיכרון, או ליחידת input / output כלשהי, היא מכניסה את כתובת הזיכרון המבוקש לתוך פס המענים. רכיבים אלקטרוניים ששולטים על הזיכרון מזהים את הכתובת שבפס המענים ודואגים לשלוח למעבד את המידע מהכתובת המבוקשת, או לכתוב לזיכרון בכתובת הנ"ל את המידע ששלח המעבד.

**פס הבקרה** תפקידו של פס הבקרה לעשות סדר בדרך שבה המעבד מתקשר עם יתר הרכיבים, האם לכתוב מידע לזיכרון או לקרא ממנו?

פס הבקרה מכיל שני קווים, קו קריאה (read) וקו כתיבה (write), והם קובעים את כיוון העברת המידע, כאשר לשניהם ערך 1, המעבד והזיכרון לא מתקשרים זה עם זה.

אם קו ה־ read מכיל אפס, המעבד קורא מהזיכרון. אם קו ה־ write מכיל אפס, המעבד כותב לזיכרון.



* הוראה
* כתובת
* מידע

# זיכרון המחשב

במחשב **יחידת הזיכרון** הבנויה כמערך.

בכל תא ניתן לאחסן מספר.

מספר זה יכול להיות **נתון**, **הוראה** או **כתובת של תא בזיכרון**.

המחשב פונה אל לתא מסוים כדי לבצע אחת משתי הפעולות:

1. קריאה של נתון או הוראה
2. כתיבה של נתון.

לשם כך יש לכל תא בזיכרון כתובת.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **FF** | **FE** | **FD** | **FC** |  |  |  |  |  |  |  |  |  | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

אפשר לחשוב על הזיכרון בתור מערך של בתים – טבלה של בתים.

כתובתו של הבית הראשון היא 0

כתובתו של הבית האחרון היא 2n – 1, כאשר n הוא רוחב פס המענים

לכן, עבור מעבד בעל 20 ביטים בפס המענים,

הזיכרון הוא מערך בגודל 1,048,575 בתים ⇦ (220 - 1)

# המעבד - Central Processing Unit

לצורף ביצוע פקודה המעבד מבצע את הפעולות הבאות:

**חישוב הכתובת בה נמצאת הפקודה הבאה** ⇦ code segment

**קריאת הפקודה הבאה לביצוע מהזיכרון**

**פיענוח הפקודה:** בשלב זה המעבד מפענח מהי הפקודה שצריכה להתבצע, איזו יחידת ביצוע להפעיל ואילו משתנים מעורבים.

**חישוב**: בשלב זה קוראים ערכים מהזיכרון במידה וצריך ⇦ רגיסטרים או data segment  
ובהתאם לסוג הפעולה מופעלת יחידת החישוב המתאימה של המעבד לביצוע הפעולות האריתמטיות והלוגיות הרלוונטיות.

**כתיבה חזרה לזיכרון:** הערך המחושב נכתב למקום אחסון התוצאה data segment.

לשם כך המעבד בנוי מהיחידות הבאות:

* **אוגרים** (Registers או "רגיסטרים"): כל אוגר הוא יחידה אחת של זיכרון פנימי מהיר ביותר הנמצא לרוב בתוך יחידת העיבוד המרכזית.
* **יחידת בקרה:** יחידה האחראית על שלבי הקריאה והפיענוח של פקודות.
* **יחידה אריתמטית-לוגית** (ALU): יחידה זו מבצעת פעולות במספרים שלמים (חיבור, חיסור, השוואה) ופעולות לוגיות שונות. היחידה נחשבת ללב המעבד.

רגיסטרים - תאי זיכרון מיוחדים הנמצאים במעבד, בעזרתם הוא מבצע את החישובים ואת ניהול הרצת התוכנית. פעולות המעבד מהירות במיוחד כשהוא משתמש ברגיסטרים במקום בתאי הזיכרון של המחשב.

* הם **מהירים** מאוד לגישה.
* יש כמות מצומצמת של רגיסטרים ולחלקם תפקידים מיוחדים.
* רגיסטרים – אוגרים, הם מקום מצוין לשמור יידע באופן זמני.
* במעבד 80x86 של אינטל יש 8 רגיסטרים של 16bit.

## רגיסטרים כללים

תפקיד: אוגרי הנתונים משתתפים בביצוע הוראות מסוגים שונים (הוראות אריתמטיות, לוגיות, הזזה,...)

תכולה: ניתן לאחסן בהם נתונים או כתובות

מספר האוגרים: ישנם ארבעה סוגים של אוגרי נתונים:

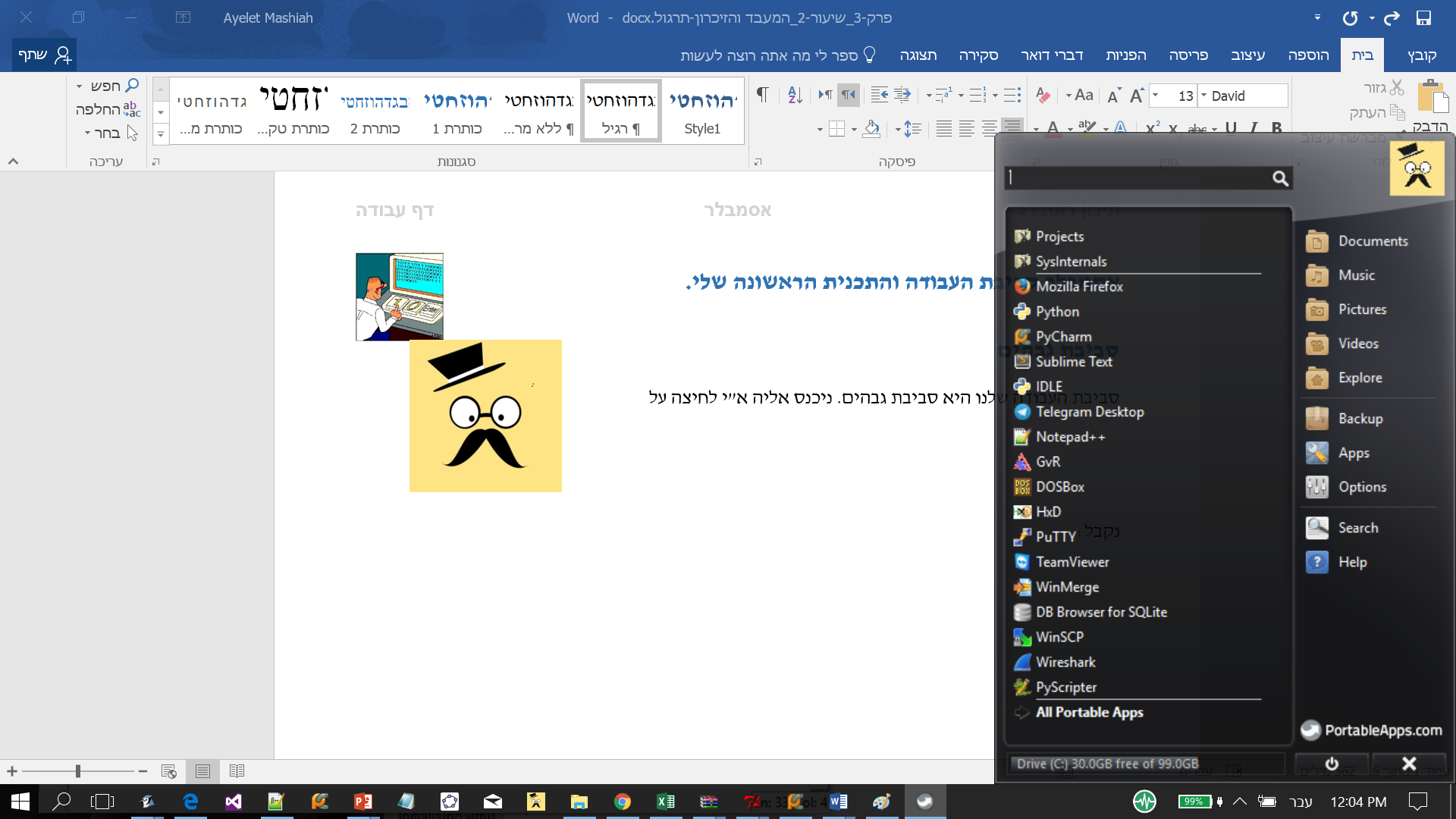
AX, BX, CX ו-DX.

|  |  |  |  |
| --- | --- | --- | --- |
| **הרגיסטר** | **שם לועזי** | **שם עברי** | **תיאור ושימוש עיקרי** |
| **AX** | Accumulator register | צובר | משמש לרוב הפעולות האריתמטיות והלוגיות. למרות שניתן לבצע פעולות חישוב גם בעזרת רגיסטרים אחרים, השימוש ב-AX הוא בדרך כלל יעיל יותר. |
| **BX** | Base address register | בסיס | בעל חשיבות מיוחדת בגישה לזיכרון. בדרך כלל משמש לשמירת כתובות בזיכרון. |
| **CX** | Count register | מונה | מונה דברים. בדרך כלל נשתמש בו לספירת כמות הפעמים שהרצנו לולאה, לכמות התווים בקובץ או במחרוזת. |
| **DX** | Data register | מידע | משמש לשתי פעולות מיוחדות: ראשית, ישנן פעולות אריתמטיות שדורשות מיקום נוסף לשמירת התוצאה. שנית, כשפונים להתקני I/O, רגיסטר DX שומר את הכתובת אליה צריך לפנות. |

# אסמבלר סביבת העבודה והתכנית הראשונה שלי.

סביבת גבהים

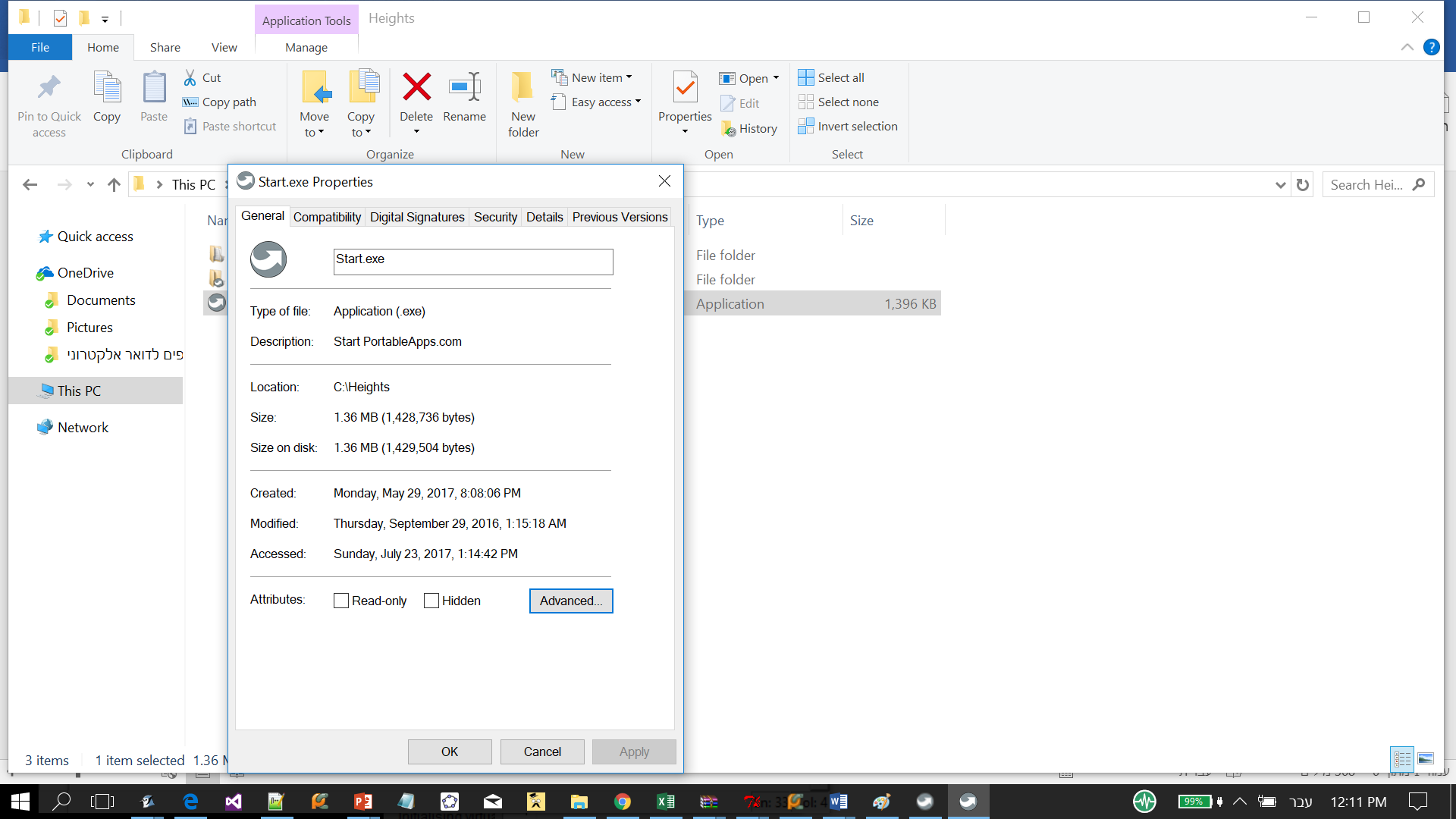
סביבת העבודה שלנו היא סביבת גבהים. ניכנס אליה ע"י לחיצה על

נקבל על המסך את החלון:

בחלון זה נמצאים הכלים בהם

נשתמש:

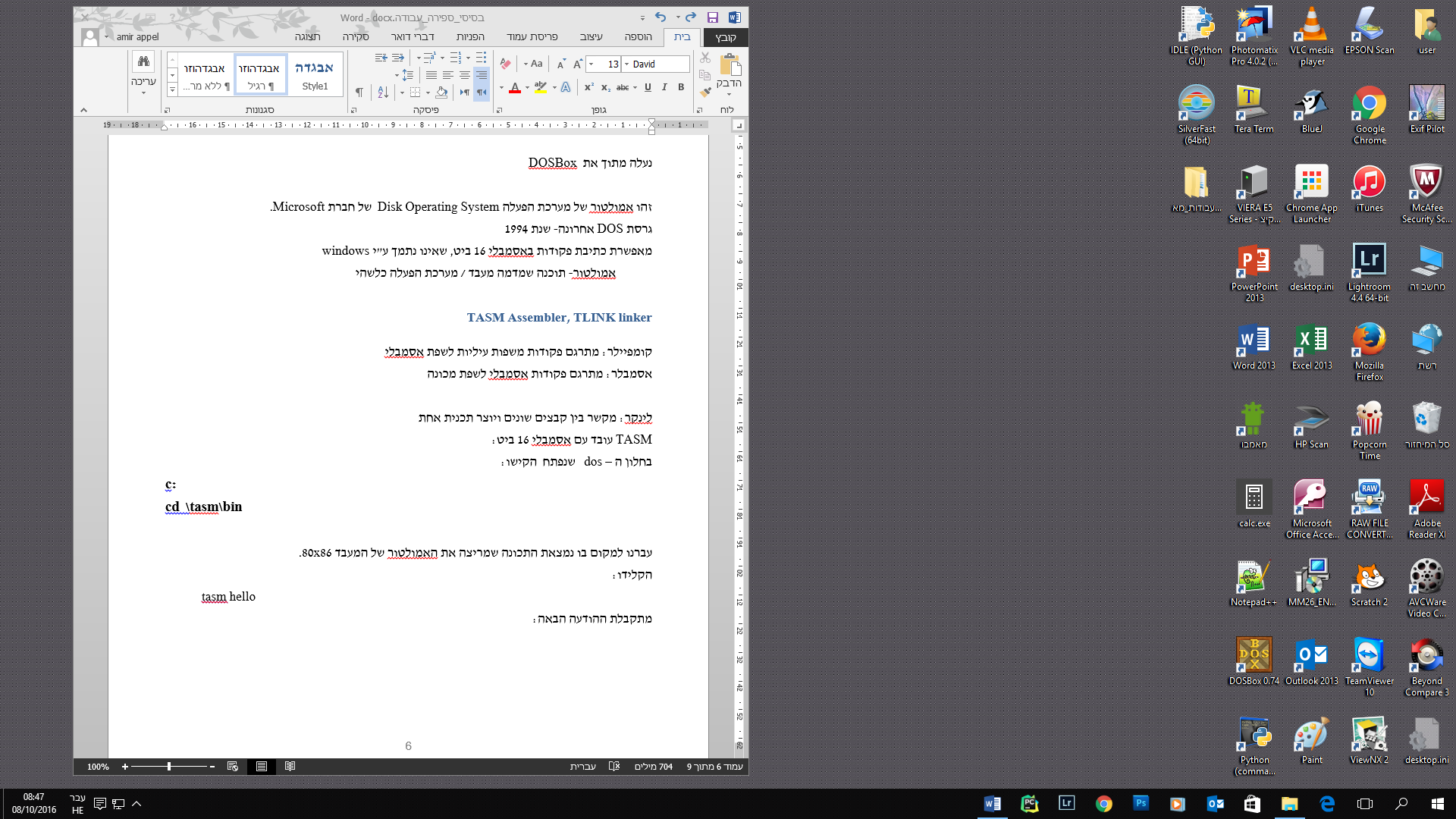
* Noteped ++
* DosBox



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

## תרגיל 1:

נכתוב ונריץ את התכנה הראשונה שלנו באסמבלי:

לכתיבה תכנית נשתמש בתכנת Noatped++

התחל

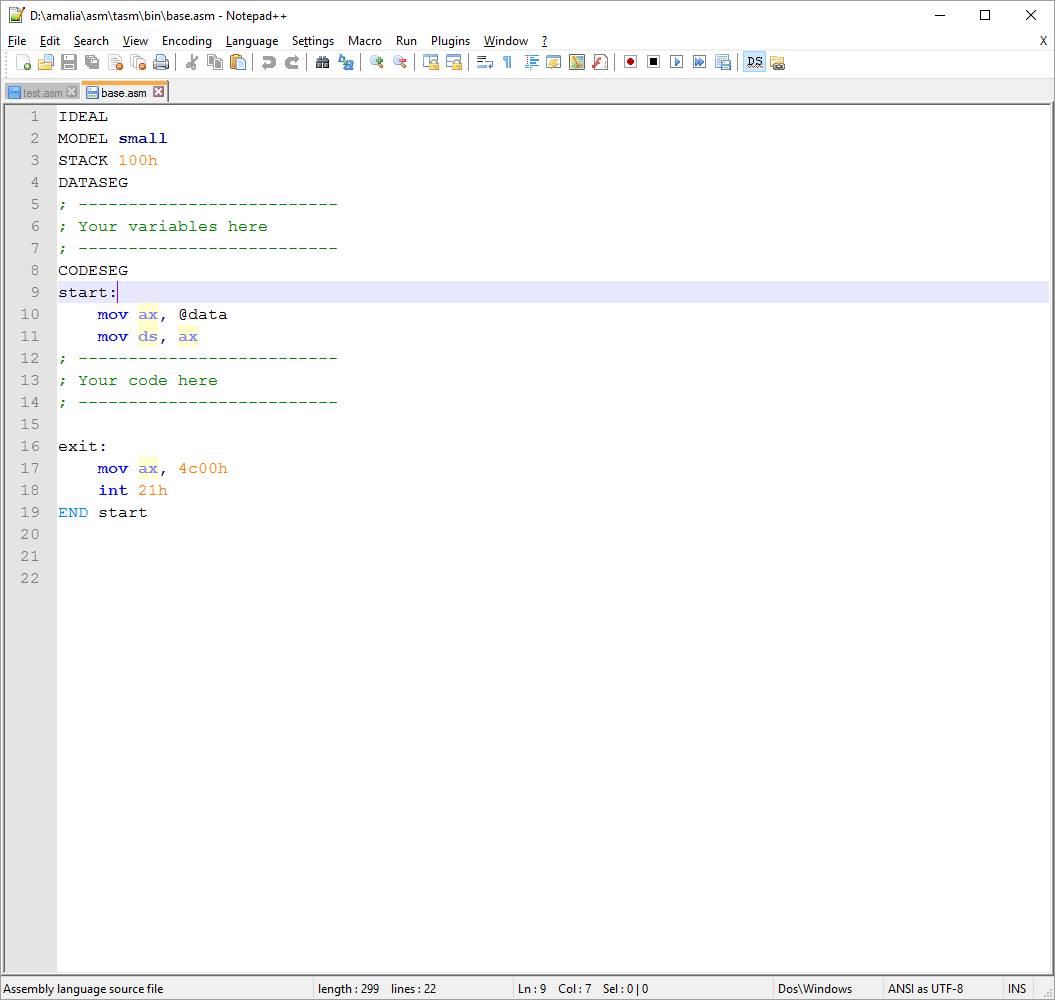
gvahim

Notped++

העלו את הקובץ ⇦ base.asm

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

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

בחרו בקובץ **base.asm** ולחצו פתח

**תוכן הקובץ**

**IDEAL** - הנחיה לאסמבלר, סגנון כתיבה. כתיבה פשוטה שמתאימה למתכנתים מתחילים.

**model**- הנחיה לאסמבלר, ארגון הסגמנטים (קובעת לאסמבלי שהתוכנית מכילה שלושה סגמנטים (איזור זיכרון: code פקודות, data נתונים ו – stack מחסנית לשמירת משתנים).

**start** – תווית שמסמנת למעבד מאיפה להתחיל את ריצת התוכנית.

**END** - הנחיה לאסמבלר, יש לבצע המרה לקוד מכונה עד לשורה זו ולהתחיל את התכנית מהתווית start

**התכנית הראשונה אותה נכתוב תדפיס למסך את המילה "Hello!"**

* ניצור משתנה בשם msg מסוג מחרוזת  
  אוסף של תווים 'במרכאות' שבסופם התו $ המסמן לאסמבלי שזהו סוף המחרוזת
* ונכתוב הוראה לאסמבלי להדפיס את המחרוזת.

תחת **DATASEG** נגדיר משתנה שהוא מחרוזת (אוסף של תווים)

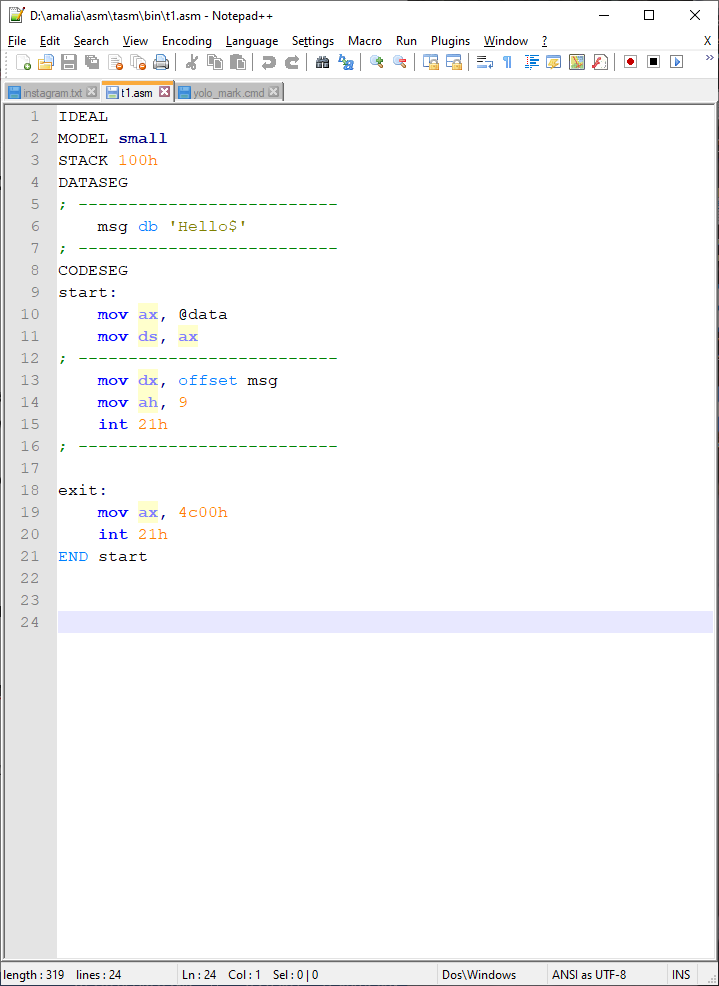
msg db 'Hello$' ; סימן ה - $ מסמל סוף מחרוזת

תחת ה – **CODESEG** נקליד את ההוראות להדפסת המחזורת.

mov dx, offset msg ; the address of or message in register dx

mov ah, 9 ; ah=9 - "print string" sub-function

int 21h ; call dos services



כדי שתוכלו להריץ את התוכנית עליכם לשמור אותה במקום בו שמור המהדר (הקומפיילר) בכונן **C**:.

C:\Heights\PortableApps\TASM\BIN

**שם חוקי להרצה הוא שם בעל 8 אותות וסימנים לועזיים המתחיל באות.**

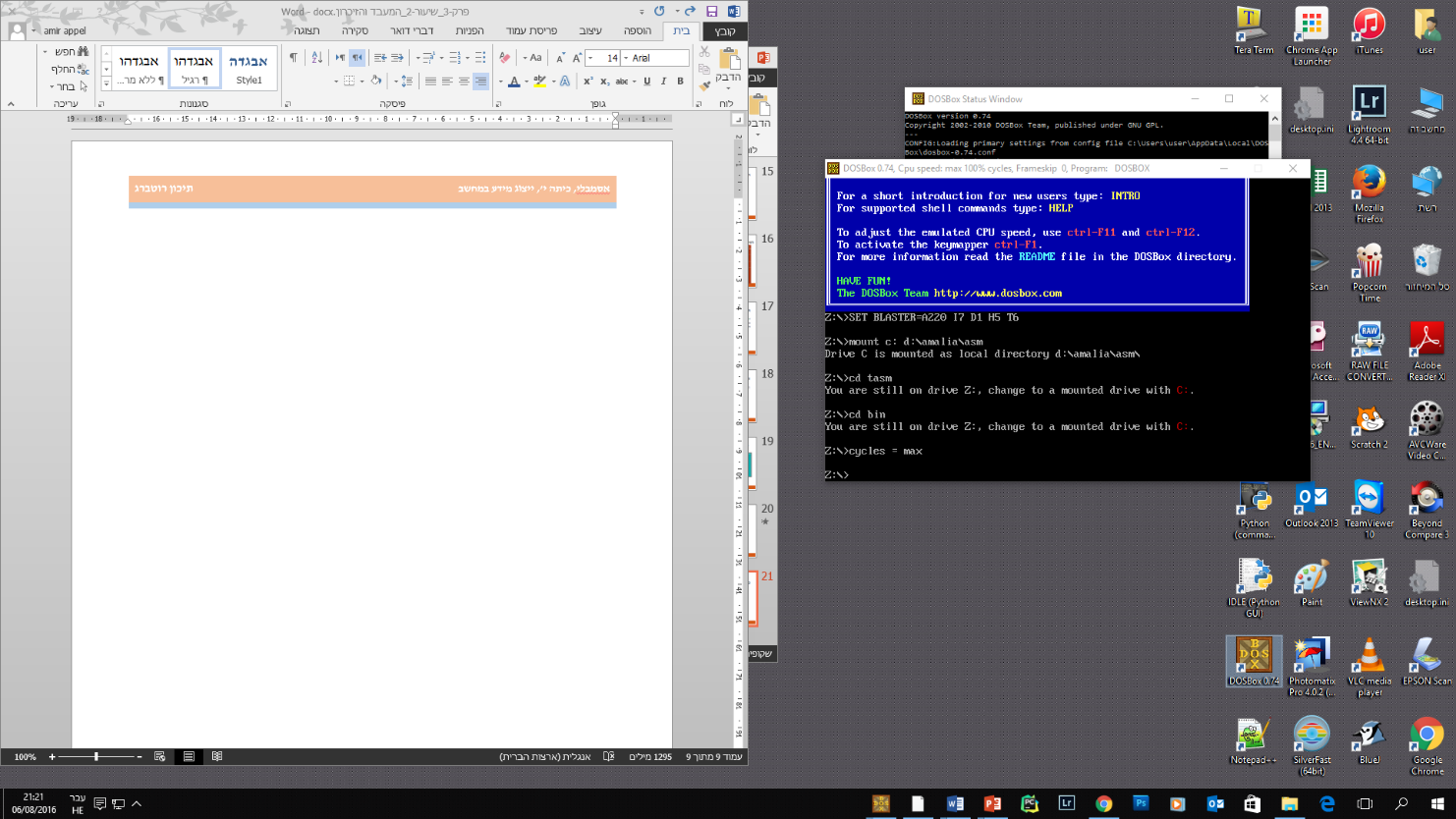
**שמרו את הקובץ תחת השם:**

**hello**

**בחרו בסיומת - .asm**

**נריץ את התכנית:**

נעלה מתוך גבהים את **DOSBox**



DosBox זהו אמולטור - תוכנה המדמה את מערכת הפעלה Disk Operating System של חברת Microsoft, גרסת DOS אחרונה- שנת 1994 המאפשרת כתיבת פקודות באסמבלי 16 ביט, שאינו נתמך ע"י windows.

**אמולטור**: תוכנה שמדמה מעבד / מערכת הפעלה כלשהי

**קומפיילר**: מתרגם פקודות משפות עיליות לשפת אסמבלי

**אסמבלר**: מתרגם פקודות אסמבלי לשפת מכונה

**לינקר**: מקשר בין קבצים שונים ויוצר תכנית אחת

TASM עובד עם אסמבלי 16 ביט:

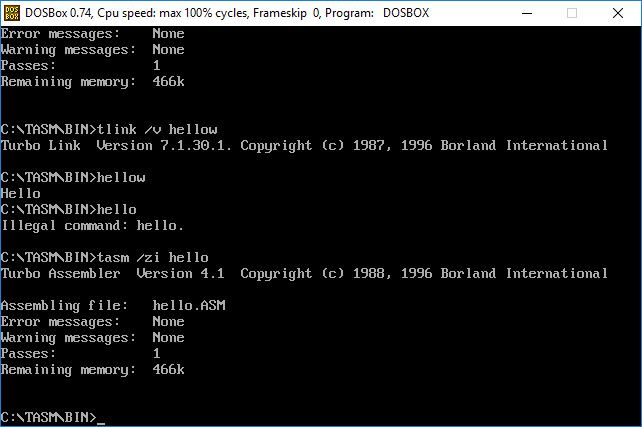
בחלון ה – dos שנפתח בדקו שאתם במקום הנכון :

C:\ TASM\BIN

הקלידו:

**tasm /zi hello**

מתקבלת ההודעה הבאה:

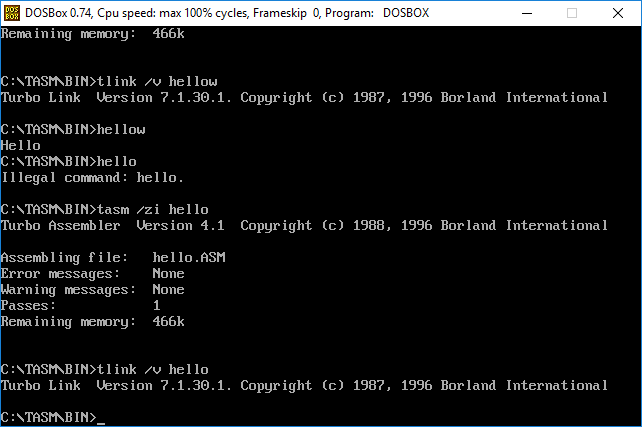


תוצאה: נוצר קובץ **hello.obj** ⇦ הקובץ הומר לשפת מכונה.

הקישו:

**tlink /v hello**

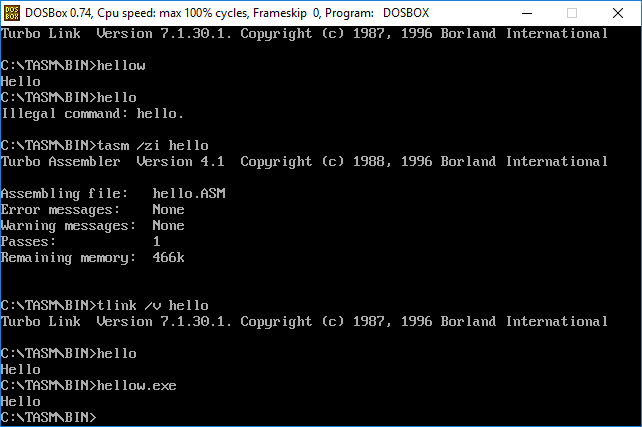
מתקבלת ההודעה הבאה:



תוצאה: נוצר קובץ **hello.exe** ⇦ קובץ הניתן להרצה ע"י המחשב.

הריצו את התכנית   
הקישו : **hello.exe**

קיבלתם על המסך הדפסה של המילה hello



עכשיו נריץ את התוכנית דרך Turbo Debugger – TD

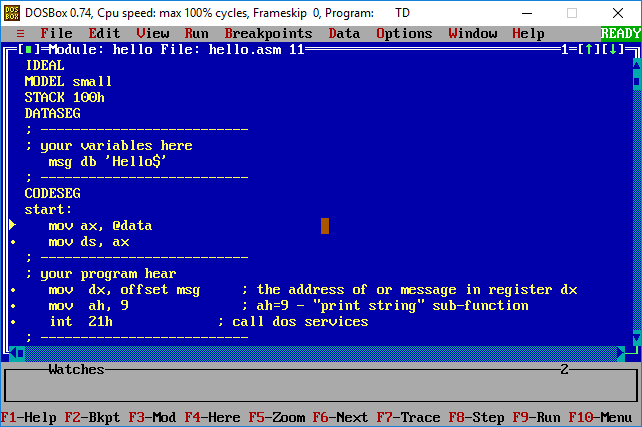
הקלידו:

**td hello**

## מסך ה – Turbo Debugger

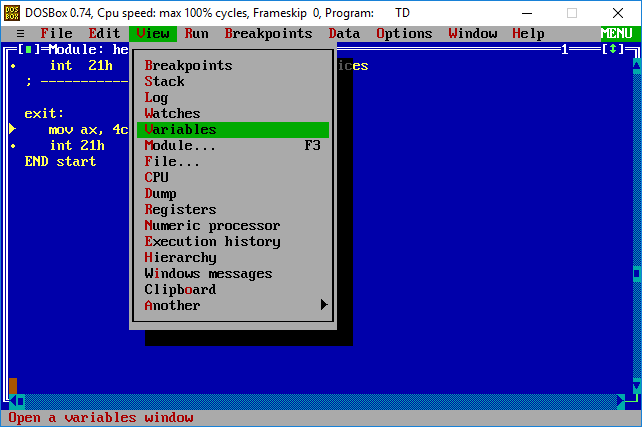
במסך המרכזי הפקודות להרצה:

בצד שמאל של הפקודות סימון משולש קטן ⮚ המצביע על הפקודה הבאה להרצה

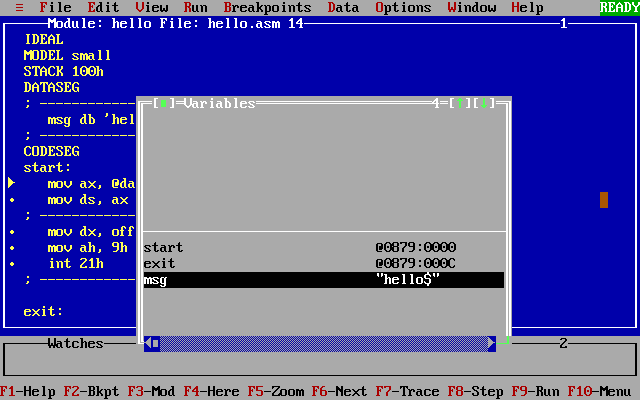


כדי שנוכל לראות את הנתונים שלנו בנוסף לפקודת ⇦ בחרו בשורת התפריט באפשרות של תצוגה **View** או **Alt + v**

בתפריט זה בחרו באפשרות של **Variables**



נקבל את פירוט המשתנים בחלון.

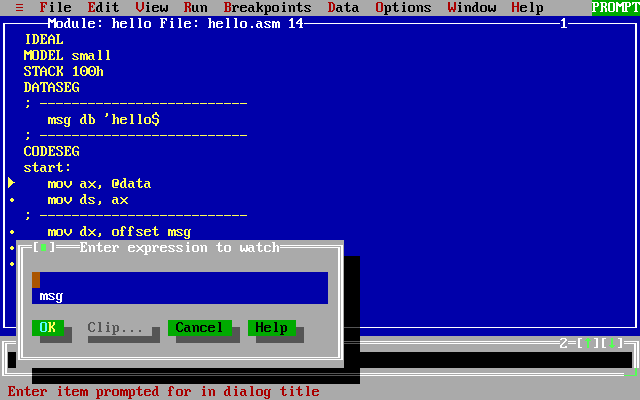


בתכנית שלנו משתנה אחד msg לגבי start ו exit נדון בהמשך

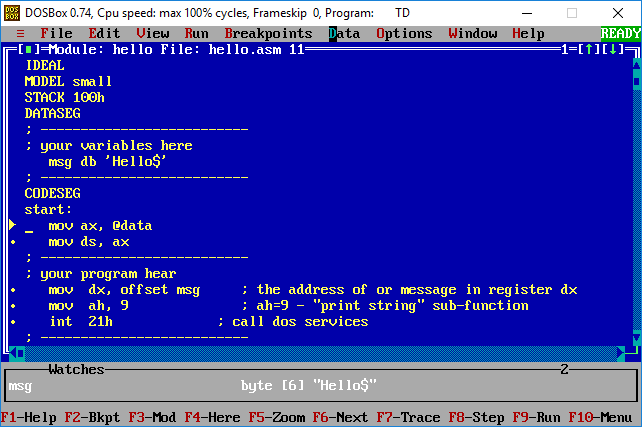
אפשרות נוספת לצפייה במשתנים שלנו היא בחלון ה Watches.

עימדו עליו ולחצו enter

בחלון שנפתח כתבואת שם המשתנה שלנו: msg



נקבל בשורה למטה את המשתנה msg וערכו “hello”



הקשה על F8 מריצה את פקודות התכנית.  
הקישו פעמיים על F8 ועמדו עם הסמן על הפקודה

**mov dx, offset msg**

כדי לראות איך המידע נראה בזיכרון המחשב

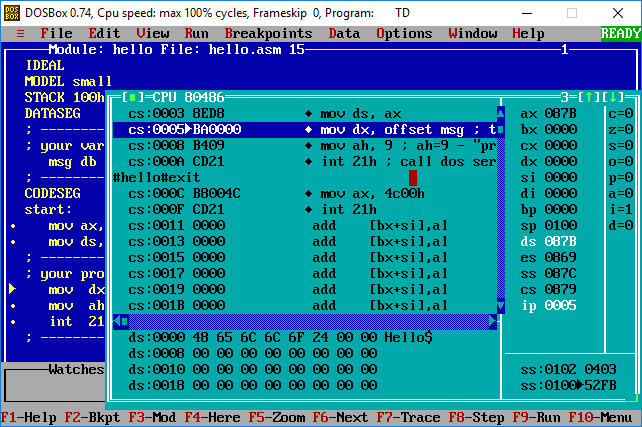
נבחר עם העכבר במילה **View** שבשורת התפריט העליונה או בצמד המקשים **Alt + v**

עולה תפריט שבו נבחר באפשרות של **CPU** או באות **C**

במסך שעולה נוכל לראות את מבנה המעבד:

הרגיסטרים

הפקודות להרצה



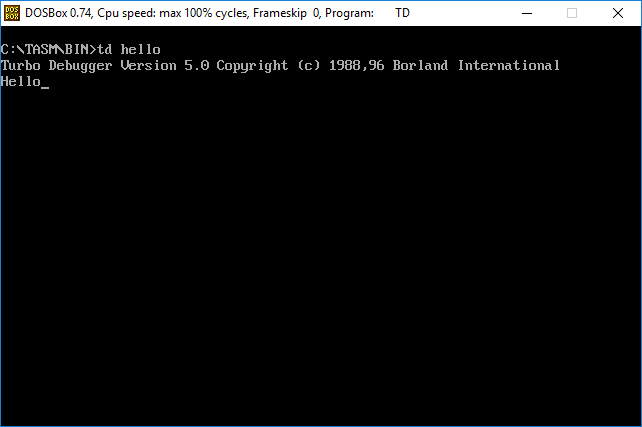
סגמנט הזיכרון

שמו לב 🎔 ניתן לראות את הנתון ששמור בסגמנט הנתונים כמחרוזת של אותיות. האותיות המרכיבות את המילה Hello הייצוג שלהם בקוד ASCII והייצוג שלהם כאותיות. האות הראשונה של המחרוזת H שמור בתא זיכרון בכתובת [0], האות הבאה בכתובת [1] וכך הלאה.   
נגיעה **בריבוע** **הירוק** העליון של תצוגת ה – cpu תסגור אותה.

נמשיך להריץ את התכנית נקיש על F8. עד לסופה ⇦ אחרי תווית ה – **exit**.

כדי לראות את תוצאת הריצה נקיש על **Alt + F5**או **Alt + w** אשר יפתח לנו את תפריט התצוגה, נבחר באפשרות של **User screen.**

נראה את תוצאת הריצה



מקש **Esc** יחזיר אותנו למסך של ה – TD (Turbo Debugger)

כדי לצאת מה – TD הקישו על **Alt+x**.

חזרו לתכנית שכתבתם ב – Noteped++

tasm / zi hello

tlink /v hello

td hello

שנו את תוכן המחרוזת (כתבו את שמכם)

* הריצו שוב את התכנית. תחילה המירו את הקובץ לשפת מכונה
* הפכו את התכנית לתוכנית להרצה במחשב
* הריצו את התכנית ב – turbo debugger

בדקו את המחרוזת שלכם ⇦ בחרו בשורת התפריט באפשרות של תצוגה **View** או **Alt + v**

בתפריט זה בחרו באפשרות של **Variables**

הריצו את התכנית בעזרת מקש F8, בסוף ההרצה בדקו את מסך המשתמש ע"י המקשים

**Alt + F5.**

**שמרו את הקובץ hello.asm ל – disk on key האישי שלכם.**

## תרגיל 2

חזרו לעבוד בתיקייה

C:\Heights\PortableApps\TASM\BIN

|  |  |  |
| --- | --- | --- |
| פקודה המדפיסה למסך **תו אחד בלבד**, את הסימן 'x'  mov dl, 'x'  mov ah, 2h  int 21h | פקודה למעבר שורה:  mov dl, 0ah  mov ah, 2h  int 21h | פקודה המדפיסה למסך מחרוזת.  לצורך כך יש ליצור ראשית משתנה של מחרוזת  mov dx, offset msg  mov ah, 9  int 21h |

נסו לצייר למסך משלוש ישר זווית של x  
הפקודה המדפיסה תו אחד למסך, מדפיסה תו אחד בלבד.

להדפסת מספר תווים עליכם להשתמש בפקודה המדפיסה **מחרוזת**.

x

xx

xxx

נסו לצייר ריבוע 4 x 4 של o

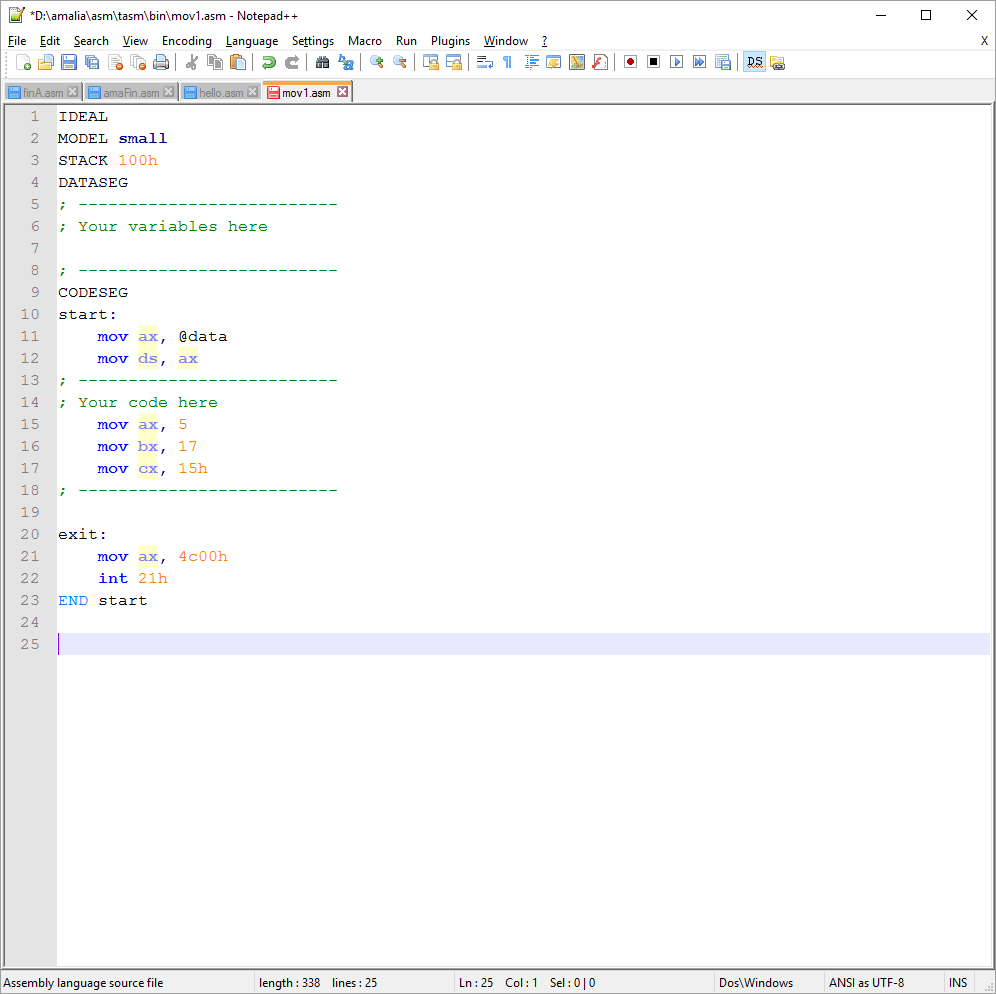
oooo

oooo

oooo

oooo

**שמרו את הקובץ hello.asm ל – disk on key האישי שלכם.**



## תרגיל 3

פקודת **mov**, היא פקודה אשר משימה נתונים לרגיסטר או למשתנה.

פתחו תכנית חדשה,

העתיקו לתוכנה את ההוראות הדרושות להרצת תכנית באסמבלי. מתוך קובץ base.asm

שמרו את התכנית תחת השם mov1 , ובסיומת asm.

כתבו ב – CODESEG

את הפקודות הבאות.

mov ax, 5

mov bx, 17

mov cx, 15h

שמרו את הקובץ, ל- C:\Heights\PortableApps\TASM\BIN

הריצו את הקובץ

tasm / zi mov1

tlink / v mov1

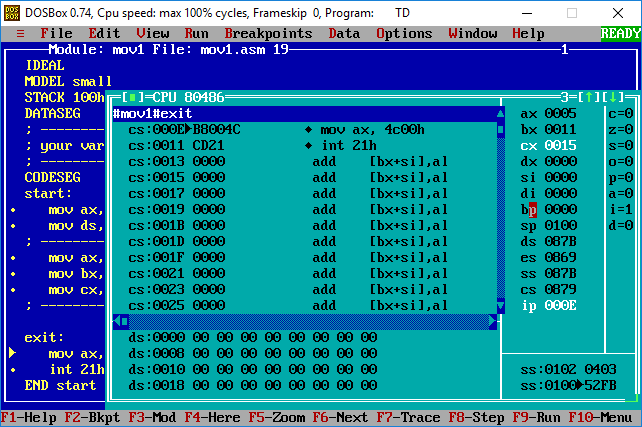
td mov1

בסוף הרצת התכנית (F8 – מעבר פקודה פקודה) בדקו את מצב הרגיסטרים.

נבחר עם העכבר במילה **View** שבשורת התפריט העליונה או בצמד המקשים **Alt + v**

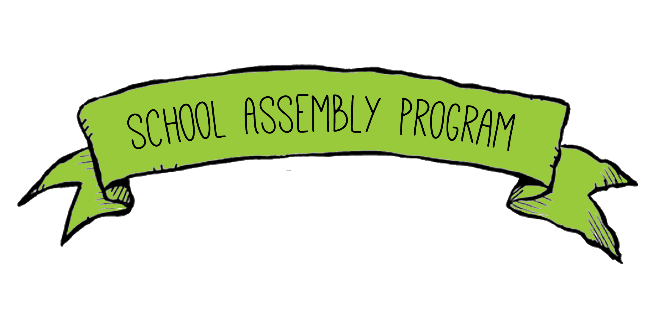
עולה תפריט שבו נבחר באפשרות של **CPU** או באות **C**

שימו לב 🎔, כיצד נראים הערכים בתוכנית וכיצד ברגיסטרים. מדוע?



כיצד "רואה" המחשב את הערכים?

15h מסמן למחשב שהערך הוא בבסיס 16 – הקסדצימאלי.

10b מסמן למחשב שהערך הוא בבסיס 2 – בינארי.

מהו הערך של 15h בבסיס 10?