גישה לזיכרון (DATASEG)

שימוש ברגיסרים עבור גישה עקיפה לזיכרון יכול להיעשות בעזרת bx, si ו-di
לא ניתן להשתמש ב- ax, cx או dx (תיווצר שגיאת קומפילציה)

בד"כ נשתמש ב- bx כדי לשמור כתובת של משהו בזיכרון (משתנה, מערך, מחרוזת וכו') ובמידה ונרצה היסט קבוע (offset) נוסיף אותו ל- bx.

לדוגמה:

```
mov bx,3
mov ax,[bx+2] ;  ax  העתק 2 בתים מכתובת 5 לאוגר
```

נסו למשל להריץ את התוכנית הבאה בדיבאגר ועיקבו אחרי מה ניכנס ל- ax:
(קודי ה- ASCII של האותיות הקטנות באנגלית מתחילים מ 61 עבור a)

```
IDEAL
MODEL small
STACK 100h
DATASEG
abc   db 'abcdefghij'
CODESEG
start:
        mov ax, @data
        mov ds, ax
        mov bx,3
        mov ax,[bx+2]
exit:
        mov ax, 4c00h
        int 21h
END start
```

בשיעור האחרון ראינו שאם הגדרנו מערך של בתים בזיכרון וניסינו לקרוא בייט לאוגר ax קיבלו הודעת שגיאה כך:

```
DATASEG
 Array  db 0AAh, 0BBh, 0CCh, 0DDh, 0EEh, 0FFh
CODESEG
 mov   ax,[Array+2]
```

התיקון יכול להיעשות בשתי דרכים, או לציין אוגר של 8 ביטים (כמו al למשל) , או לכתוב:

```
mov    ax,[word ptr Array+2]
```

האפשרות השנייה הייתה גורמת ל-2 בתים החל מכתובת Array+2 להיכנס ל- ax , כלומר ax היה מכיל DDCC
(בגלל שיטת האיחסון של little Endian הם הפוכים)

בדף הבא יש תיאור של התרגיל בית (תרגיל 6.8 - כדאי לקרוא גם את ההקדמה)

(רמז לפיתרון - אם תריצו בדיבאגר פקודה כמו: mov ax,3 תוכלו לראות לאיזה אופקוד זה מיתרגם. אם תצליחו 'לשתול' אופקוד זה בתוכנית, במיקום רצוי - יש מצב שתוכלו לפתור את הבעיה :)

שינוי קוד התוכנית בזמן ריצה (הרחבה)

מה דעתכם ליצור תוכנית שהקוד שלה משתנה תוך כדי ריצה?

אחד היתרונות המעניינים של אסמבלי על פני שפות עיליות, הוא שיש לנו גישה מלאה לזיכרון – כולל זיכרון הקוד – ואנחנו יכולים לעשות בו כרצוננו. בואו נראה איך עושים את זה. נתחיל בהסבר תיאורטי קצר:

נניח שאנחנו פונים למקום בזיכרון [1], לדוגמה על ידי הפקודה

```
mov [1], al
```

איך האסמבלר יודע לתרגם את [1] לכתובת מלאה בגודל 20 ביטים?

היזכרו בהסבר על שיטת הסגמנט והאופסט. הפעולה הראשונה שהאסמבלר עושה היא להניח שאתם מתכננים לפנות למקום שנמצא בסגמנט הנתונים. כיוון ש-ds מכיל את כתובת סגמנט הנתונים, הפעולה שהאסמבלר יבצע היא לקחת את ds, להכפיל ב-16 (היזכרו מדוע) ולהוסיף לו 1 (האופסט).

למעשה הפקודה האחרונה זהה לפקודה הבאה:

```
mov [ds:1], al
```

כל ההבדל הוא שהפעם ציינו במפורש מה הסגמנט שאליו אנחנו כותבים, בעוד שלפני כן נתנו לאסמבלר להניח באיזה סגמנט מדובר.

כעת ננסה משהו קצת משוגע:

תרגיל 6.8: תרגיל אתגר- שינוי תוכנית תוך כדי ריצה

לפניכם קטע קוד. העתיקו אותו לתוך התוכנית שלכם בשלמותו:

```
xor ax, ax

xor bx, bx

add ax, 2

add ax, 2
```

א. כפי שניתן לראות, בסיום קטע הקוד ערכו של ax הינו 4. המשימה שלכם היא לגרום לכך שבסיום הריצה של קטע הקוד, ערכו של ax יהיה 3. אך ישנן מספר מגבלות:

- יש להעתיק את קטע הקוד בשלמותו ובלי להוסיף שורות קוד באמצע (מותר לפני ואחרי)
- אין להשתמש בפקודות קפיצה או בתוויות
- יש להשתמש אך ורק בפקודות mov

ב. כעת גירמו לכך שבסיום הריצה של קטע הקוד, ערכו של ax יהיה 3 וערכו של bx יהיה גם הוא 3. כל הכללים מסעיף א' תקפים גם כעת.