**C Home\_assignment**

*יהונתן ארמא 207938903*

**Part B:**

1. ההבדל בין משתנים גלובליים ללוקאליים:

בשפת c, משתנים ניתן להגדיר בתוך פונקציות ומחוץ לפונקציות.

משתנים המוגדרים מחוץ לפונקציות נקראים 'משתנים גלובליים' ומשתנים המוגדרים בתוך פונקציות נקראים 'משתנים לוקאליים'.

ה-“scope” של המשתנים הגלובליים הוא כל התוכנית, כלומר כל פונקציה בתוכנית (יותר נכון ב-file כל עוד לא הגדרנו את המשתנה ב-extern) יכולה 'לראות' את המשתנה ולהשתמש בו.

ה-“scope” של המשתנים הלוקאליים הוא הפונקציה או "הסוגריים המסלוסלים" בהם הם הוגדרו, כלומר אם משתנה הוגדר בפונקציה מסוימת אז לא ניתן 'לראות' אותו בפונקציה אחרת, ואם משתנה הוגדר בתוך בלוק מסוים (לדוגמה “if”) אז לא ניתן 'לראות' אותו מחוץ לבלוק.

עוד הבדל בין השניים – משתנים לוקאליים שמורים במחסים (Stack frame) בעוד שמשתנים גלובליים שמורים מחוץ ל-Stack. (BSS או Data)

בתוכנית שהוצגה לנו:

דוגמה למשתנה גלובלי:

maxTrace (שורה 34) – חוץ מזה שסומן לנו בגדול שזה החלק של המשתנים הגלובליים, ניתן להבין שזה משתנה גלובלי מכיוון שהוא לא בתוך אף פונקציה (הוא לא חלק משום “scope”) ולכן ה-“scope” שלו הוא כל התוכנית.

דוגמה למשתנה לוקאלי:

Trace (שורה 101) – משתנה זה הוגדר לנו בתוך פונקציה (ComputeTrace) ולכן ה-“scope” שלו הוא רק בתוך הפונקציה הזאת.

1. ניתן לראות שכאשר עושים Debug לתוכנית מקבלים עליה נתונים בזמן אמת.

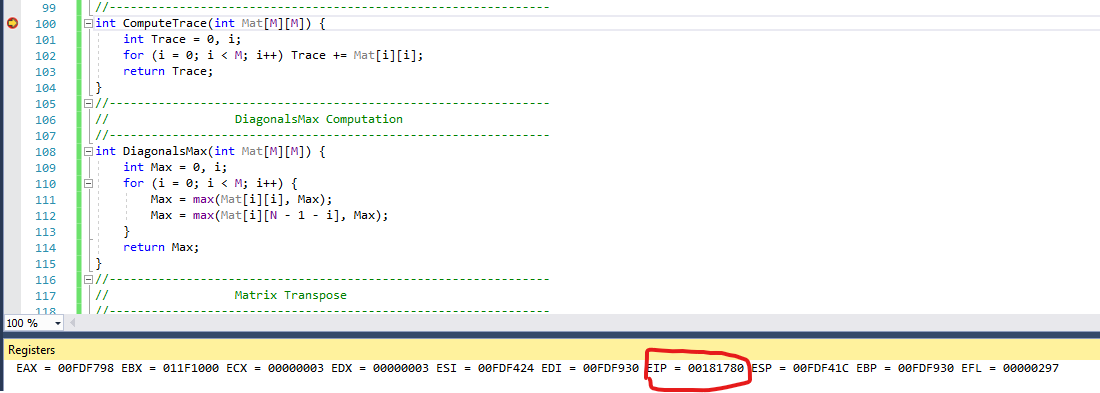
A screenshot of a computer

Description automatically generatedלדוגמה, בריצה זו קיבלנו שהערך של המשתנה Mat הוא "0x00d8fac8" שזה הכתובת של האיבר הראשון במערך.

בנוסף, הכתובת האחרונה של המערך תהיה הכתובת שקבלנו ועוד 4 (שזה גודל int) כפול M (אורך המטריצה) כפול M (רוחב המטריצה) פחות 1 (כי התחלנו לספור מ-0). ומכיוון שבמקרה שלנו M=10 אז נקבל שהאיבר האחרון במערך שמור ב-offset של 396 מההתחלה של המערך כלומר במרחק הקסה של 18C. ולכן יהיה שמור בכתובת "0x00d8fc54".

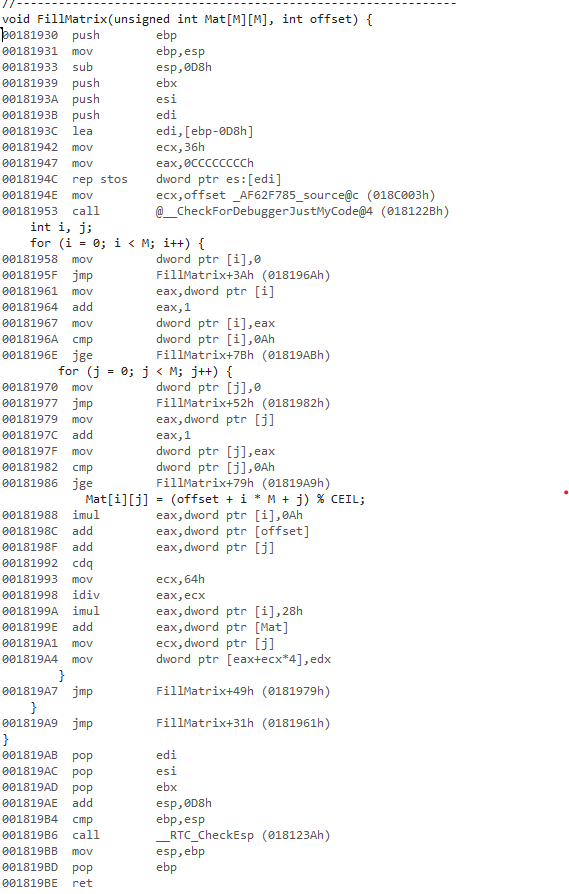
נבחין שכתובות אלו ישתנו עם כל הרצה של התוכנית ולא ישארו קבועות.

1. הפעם פתחנו את החלון של הרגיסטרים ב-Debug Mode, ושמנו BreakPoint בדיוק בהגעה לפונקציה ComputeTrace. קיבלנו את הערך הבא:



ניתן לראות שערך הרגיסטר ה- EIP (כלומר PC) הוא 00181780.

1. נבחין שזוהי הפונקציה FillMatrix ב- DisAssembly:



ניתן להבחין כי:

הכתובת הראשונה היא 00181930

וזוהי כתובת הפונציקה.

שם הפונקציה FillMatrix מרמז על כך שזאת הפונקציה שממלאת לנו את המטריצה ואכן הפונקציה הזאת מקבלת מטריצה (תאכלס עושה שימוש בכתובת של המטריצה וממלאת שם ערכים) וממלאת את הכניסות במטריצה בערכים לפי נוסחה.

גודל הפונקציה:

ניתן לראות שהפונקציה מתחילה בכתובת 00181930 ונגמרת בכתובת 001819BE.

סה"כ לאחר חיסור בהקסא:

8E כלומר 142 בייטים.

הפונקציה נשמרת ב-RAM באזור שנקרא

Code segment

או בשם אחר Text Segment.

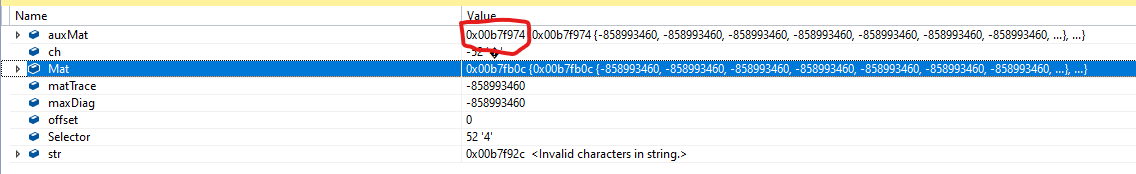
שזהו מיקום ReadOnly ב-Ram שנמצא בתחילתו.

1. נבחין שמשתנה auxMat הוא משתנה לוקאלי שמוגדר בתוך פונקציית ה-main.

כלומר ניתן לגשת אליו בכל פונקציית ה-main ורק בתוכה.

יש לשים לב שהוא נשלח לפונקציה אחרת מה-main, וזה אולי יכול לבלבל, אבל בעצם מה שנשלח זה הכתובת של המערך (כי ערך של משתנה מסוג מערך הוא בעצם הכתובת בה הוא שמור) ובגלל שב-C השימוש הוא ב- Call By Value אז הכתובת מגיעה לפונקציה הנקראת והיא יכול לערוך את המערך. אבל, לא בגלל שניתן לגשת למשתנה דרכה.

כתובתו בבפונקציית ה-main:



והוא שמור על ה-stack.

1. לפי Disassembly:

A computer screen shot of a white screen

Description automatically generated