מטלת הגשה בשפת C במבנה מחשבים ספרתיים

שם המגיש: איילון קפל

תעודת זהות: 207807025

**B. שאלות חלק תיאורטי:**

1. ראשית, ישנו מונח הנקרא scope המתאר את מרחב המשתנים אשר מוקצה בכניסה לפונקציה, לולאה, תנאי וכו', המרחב הזה יהיה נגיש רק עבור המקום שהוא הוגדר בו, לכן לדוגמה אם קיימות שתי לולאות (שאינן אחת בתוך השנייה) ובכל אחת מוגדרים משתנים בעלי אותו שם, כאשר התוכנה רצה המשתנים האלו יהיו שמורים תחת כתובות שונות בזיכרון ובכך למעשה שונים זה מזה, כלומר פונקצייה אחת לא נגישה למרחב המשתנים שפונקצייה אחרת מגדירה, המשתנים שתוארו באופן זה הינם **משתנים לוקאליים**.

זאת בשונה **ממשתנים גלובליים** שהם משתנים שאנו מגדירים אותם להיות נגישים לכל הקוד ובכך תתאפשר לנו גישה אליהם בכל פונקצייה\לולאה\תנאי וכו'.

בקוד הניתן בexample2 ניתן לראות כי המשתנה menu **הינו משתנה גלובלי** מפני שהוא מוגדר בהיררכיה הגבוהה ביותר (ולא בתוך פונקצייה לדוגמה) ולכן מתאפשרת אליו הגישה לכל אורך הקוד כמו לדוגמה בפונקציה main, לסיכום הסקופ של משתנה זה הוא כל התוכנה.

ולעומת זאת נראה כי לדוגמה המשתנה i **הינו משתנה לוקאלי** המוגדר במספר פונקציות אשר כל אחת שומרת אותו בכתובת נפרדת בזיכרון, הסקופ של משתנה זה הינו הפונקצייה שבה הוא מוגדר.

1. Calendar

   Description automatically generatedלאחר הרצה של התוכנית עד להגעה לbreakpoint בנקודה בה יצרנו את המטריצה Mat ומילאנו אותה בערכים, חלון disassembly מציג עבורה את הערכים הבאים:

ניתן לראות כי המטריצה בנויה מ10 מערכים אשר כל אחד מהם מכיל 10 משתנים מסוג int.

הגודל 10 נקבע על ידי הקבוע M.

הכתובת ההתחלתית של המטריצה הינה 0x007af7a8 והאיבר האחרון נגמר בכתובת

0x007af910+28=0x007af938, כלומר, המטריצה הינה בגודל

0x007af938-0x007af7a8=0x938-0x7a8=0x190=400(decimal), כלומר, 400 ביטים.

הסיבה לכך היא שמשתנה מסוג int נשמר על ידי 32 ביטים בזיכרון, כלומר 4 בתים (כאשר כל כתובת בזיכרון מייצגת בית, 8 ביטים), מכיוון שיש 100 משתנים כאלו במטריצה (מטריצה בגודל 10X10) המקום שהיא תופסת בזיכרון הינו אכן 400 ביטים.

1. כפי שנכתב בפורום במודל, יש להתעלם מהסעיף הזה בגלל בעיה בניסוח
2. על מנת לראות את ערך הרגיסטרים או בפרט את רגיסטר PC בהגעה לפקודה הראשונה של הפונקציה compute trace נשים breakpoint בנקודה הזו כפי שניתן לראות בתמונה:

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generatedלאחר הגעה לנקודה זו נראה כי:

הפקודה הראשונה בהגעה לפונקציה בשפת assembly נמצאת בכתובת 0x009E17A8, ערך זה נשמר ברגיסטר EIP-Extended Instruction Pointer השקול לרגיסטר PC-Program Counter והוא מחזיק את הכתובת של הפקודה שאותה יש לבצע כעת.

1. לאחר השמה של breakpoint בהגעה לפונקצייה fillmatrix והרצה של התוכנית באופן כזה אשר קורא לפונקציה אנו רואים כי הפונקציה נמצאת בכתובת הבאה: 0x00F61930   
   A picture containing text

   Description automatically generated  
     
    כמו כן, הכתובת בה הפונקצייה מסתיימת הינה: 0x00F619AD כאשר הפקודה return גוררת הפעלה של מספר פקודות pop אשר ביציאה מהפונקצייה מחזירות את המידע אשר נדחף לרגיסטרים בעת הכניסה לפונקצייה.  
   Diagram, text

   Description automatically generated

מכאן אנו מבינים כי גודל הפונקציה הינו 125 בתים, סוג הזיכרון שבו התוכנית שמורה במהלך הריצה הינו RAM מפני שכאשר אנו נכנסים לפונקצייה, נוצר scope חדש שבו מוגדרים משתנים חדשים ונעשים חישובים אשר הינם זמניים, לאחר ביצוע הפונקצייה התוצאות של הפעולות שהיא יכלה לבצע הן עדכון של ערכים בזיכרון (לדוגמה כתיבה של ערכים שונים למטריצה אשר שמורה במקום נתון בזיכרון והפונקצייה קיבלה את המקום הזה כפויינטר) או החזרת ערך כלשהו, לאחר מכן כל המידע לגבי הפונקצייה הכלול במשתנים שהיא הגדירה וכו' אינו רלוונטי יותר ולכן אנו רוצים שהוא יימחק, מסיבה זו הזיכרון של הפונקצייה הינו RAM שכן זהו זיכרון נדיף ולכן השימושים בו הם לחישובים זמניים, בניגוד לתוכנית עצמה אשר שמורה במקום קבוע בזיכרון (ואינה תימחק) מכיוון שהיא מכילה את המידע על הפקודות שיש לבצע ומידע זה חייב להיות זמין לנו תמיד ולכן לא יכול להיות זמני.

כמו כן, הקשר בין הפעולות שמבצעת הפונקצייה לבין השם שלה הן שהפונקצייה מקבלת פויינטר למטריצה אשר קיימת בזיכרון ובנוסף לכך מקבלת ערך offset ובאמצעות כך מחשבת ערכים בעלי חוקיות כלשהי התלויה בoffset ומזינה אותם באיברי המטריצה, מכאן השם של הפונקצייה fillmatrix שכן הפונקצייה "ממלאת" את המטריצה בערכים ובכך נוכל לבצע פעולות על המטריצה כאשר יש בה ערכים כלשהם (לעומת לפני הקריאה לפונקצייה בפעם הראשונה בה במטריצה לא הוגדרו שום ערכים).

1. המשתנה auxmat מוגדר בתחילת הפונקצייה main ולכן למעשה זהו הscope שלו, כלומר, כל לולאה, תנאי, פונקצייה וכו' אשר מבוצעות במהלך הפונקצייה main (וגורמות לפתיחת scope נוסף) נגישות עדיין למשתנה.

Graphical user interface, application

Description automatically generatedנראה כי הכתובת בה הוא הוגדר בזיכרון בעת ריצת התוכנית הינה : 0x00cff828

1. A screenshot of a computer

   Description automatically generated with medium confidenceלאחר הוספת קטע הקוד לתוכנית שאנו מריצים נוכל להסתכל בתרגום של הקומפיילר עבורה לשפת assembly ונראה כי התוצאה הינה:

ניתן לראות כי שורת הקוד מתורגמת למספר פעולות בסיסיות יותר בשפת assembly כמתואר בתמונה, בין היתר הפעולות הינן הזזות של ערכים לרגיסטרים על מנת לעשות פעולות, קפיצה למקומות בזיכרון כתלות בתוצאות חישוב מכיוון שקיים תנאי שלפיו נעשית בדיקה האם a>b וכתוצאה של הבדיקה הזו אנו נבצע אחת מבין שתי פעולות שונות ועוד פעולות שבוצעו על מנת לממש את שורת הקוד.