Preface Report- Lab 1

Tomer Sharon 314969601
Alon Gorka 206924698

שאלות תיאורטיות:

1. משתנה לוקאלי:

- משתנה לוקאלי נוצר בתוך פונקציה או בלוק בתוך הקוד.
- אותן משתנים ניתנים לשימוש רק באותן הפונקציות/בלוקים בהן הן נוצרו.
- הסקופ של משתנה לוקאלי מוגבל רק לפונקציה הנוכחית, כשאר יוצאים מהפונקציה כך המקום בזיכרון שהמשתנה שמור משתחרר בהתאמה.

דוגמה למשתנה לוקאלי הוא Trace שהסקופ שלו הוא הפונקציה ComputeTrace .

```
//-----------------------------------------------------------------
//                    Trace Computation
//-----------------------------------------------------------------
int ComputeTrace(int Mat[M][M]){
        int Trace=0,i;
        for(i=0 ; i<M ; i++) Trace += Mat[i][i];
        return Trace;
}
```

משתנה גלובאלי:

- משתנה גלובאלי נוצר מחוץ לפונקציה מסוימת או בלוק מסוים.
- ניתנים לשימוש בכל הקובץ בו הם נוצרו.
- הסקופ של המשתנה הגלובלי, הוא מרגע יצירתו ולאורך כל הקובץ.

```
//-------------- Global variables ------------
  int maxTrace,maxDiag;
```

2. לאחר הרצת ה-setup code, מיקום המטריצה Mat2 הוא:
:IAR

Expression	Value	Location	Type
Mat2	<array>	Memory:0x1F6A	int[10][10]
[0]	<array>	Memory:0x1F6A	int[10]
[1]	<array>	Memory:0x1F7E	int[10]
[2]	<array>	Memory:0x1F92	int[10]
[3]	<array>	Memory:0x1FA6	int[10]
[4]	<array>	Memory:0x1FBA	int[10]
[5]	<array>	Memory:0x1FCE	int[10]
[6]	<array>	Memory:0x1FE2	int[10]
[7]	<array>	Memory:0x1FF6	int[10]
[8]	<array>	Memory:0x200A	int[10]
[9]	<array>	Memory:0x201E	int[10]

Go to Mat2 Memory

```
00001f60  86 61 01 00 80 6e 52 21 00 00 6e 62 5e 30 40 b7
00001f70  65 ca 95 38 32 0f fa 42 cf 2b 40 35 05 e5 86 08
00001f80  99 14 75 7b 58 74 ca 49 91 e5 a9 ed bf d6 75 04
00001f90  98 5a 91 be 24 b6 52 75 ac 67 7f 5e 97 da 2d 62
00001fa0  81 f0 4f 25 32 f9 4b 31 ac c3 ea 5e 8b e6 f2 5f
00001fb0  5f 6d 50 1c b9 07 00 b3 2f 56 3a d8 e2 9b 89 52
00001fc0  74 b7 43 ec e2 db 1a 94 b2 56 30 58 f9 d7 75 38
00001fd0  91 18 b2 ce 13 ba f2 d8 e9 e1 52 20 13 48 5a c1
00001fe0  5d c9 54 6f bd bd e3 06 86 12 3c 32 a6 74 bc ec
00001ff0  ac 84 db 0a 35 61 99 31 12 88 6d 66 b5 d4 df 0b
00002000  4b 98 47 f6 f9 99 dc 0e 31 00 98 f5 13 db 8b d4
00002010  d1 ef 33 37 0b 5f 6a 7f 81 5c f3 8e 3c 0b 61 7c
00002020  f1 ab 27 87 b4 bf bb a8 dd bb 0d 62 a3 01 2a b8
00002030  45 2b 00 00 01 00 02 00 03 00 04 00 05 00 06 00
00002040  07 00 08 00 09 00 0a 00 0b 00 0c 00 0d 00 0e 00
```

הטווח הכתובות של המטריצה הוא בין 0x1F6A לבין 0x2031. משתנים שנוצרים בעזרת פונקציות, נשמרים ב-RAM, ולכן המטריצה שנוצרה באמצעות הפונקציה FillMatrix נשמרת שם.

:CCS

Mat2	int[10][10]	[[0,0,0,0,0...],[0,0,0,0,0...],[0,0,0,...	0x04C6

3. IAR: המהדר קובע את תחילת המחסנית ב- 0x00000190:

```
void main(){
main:
?cstart_end:
 002118    120A          push.w  R10
 00211A    8031 0190     sub.w   #0x190,SP
```

CCS:

בואפן דומה, גם ב-CCS המחסנית נקבעת למיקום 0x190:

```
 16    void main(){
       main():
c082:    120A          PUSH    R10
c084:    1209          PUSH    R9
c086:    8031 0190     SUB.W   #0x0190,SP
```

4. IAR: תוכן הרגיסטר SP הוא 0x1F62.
CCS: תוכן הרגיסטר SP יהיה 0x0268.

Core Registers		Core Registers
PC	0xC2D8	Core
SP	0x0268	Core

5. **IAR:** כתובת הפונקציה FillMatrix בזיכרון היא 0x00002324. מצאנו אותו ע"י ריצה בדיבאגר עד לפקודה הראשונה, ומשם שימוש ב-disassembly על מנת למצוא את כתובת הפקודה האחרונה.
הפקודה הראשונה:

```
FillMatrix:
 002324    153B                pushm.w #4,R11
for(i=0 ; i<M ; i++){
```

הפקודה האחרונה:

```
 002350    531E                inc.w   R14
for(i=0 ; i<M ; i++){
 002352    903E 000A           cmp.w   #0xA,R14
 002356    3402                jge     0x235C
for(j=0 ; j<M ; j++){
 002358    430D                clr.w   R13
 00235A    3FF7                jmp     0x234A
}
 00235C    1738                popm.w  #4,R11
 00235E    0110                reta
```

ולכן גודל הפונקציה הוא:

$$02360 - 02324 = (3C)_{hex} = (60)_{10}$$

לאחר בדיקה בחלון ה-Memory בסביבת העבודה ניתן לראות שהפונקציה שוכנת בזיכרון הפלאש.

CCS: הפונקציה מתחילה ב-C288, ומסתיימת בc242 ולכן גודלה 4A.

```
        FillMatrix():
c288:     120A                PUSH    R10
c28a:     1209                PUSH    R9
c28c:     1208                PUSH    R8
c28e:     4C08                MOV.W   R12,R8
```

$$0C2D2 - 0C288 = (4A)_{hex} = (74)_{10}$$

6. IAR: נסתכל ב-CycleCounter, ונחסר את הזמן המחזור שבו התחלנו את הפונקציה מזמן המחזור בו סימנו לבצע אותה. נקבל:

```
FillMatrix(Mat1);
002126    410C              mov.w   SP,R12
002128    503C 00C8         add.w   #0xC8,R12
```

CYCLECOUNTER	=	62
CCTIMER1	=	62

```
}
00235C    1738              popm.w  #4,R11
00235E    0110              reta
int ComputeTrace(int Mat[M][M]){
ComputeTrace:
002360    151B              pushm.w #2,R11
```

R13	=	0x0000A
R14	=	0x0000A
R15	=	0x00012
CYCLECOUNTER	=	3408
CCTIMER1	=	3408

$$3408 - 62 = 3346_{10}$$

CCS: בהרצה על הבקר נקבל: באמצעות שימוש בכלי ה-Profile Clock, בתחילת הפונקציה אנחנו ב 5084 מחזורי שעון CPU.

7. IAR: הסקופ של המשתנה mat2Trace הוא ה-main, כי שם המשתנה מוגדר ושם ניתן

```
//-------------------------------------------
void main(){
  WDTCTL = WDTPW | WDTHOLD;      // Stop WDT

  int Mat1[M][M],Mat2[M][M];
  int mat1Trace,mat2Trace,max1Diag,max2Diag;
  int Selector=0;

  FillMatrix(Mat1);
```

להשתמש בו.

מיקומו בזמן הסקופ הוא: רגיסטר R12L

CCS: באופן דומה.

mat2Trace	int	816	Register R12

8. קוד האסמבלי המקביל לשורה (גם בIAR וגם ב-CCS) הוא:

```
maxTrace = mat1Trace > mat2Trace ? mat1Trace : mat2Trace;
002120    9F0E              cmp.w   R15,R14
002122    3403              jge     0x212A
002124    4F82 1100         mov.w   R15,&maxTrace
002128    0110              reta
00212A    4E82 1100         mov.w   R14,&maxTrace
```