Preface LAB1

יהונתן ארמא 207938903

יובל סעיד 206921892

חלק E

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 14) – 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 57) – משתנה זה הוגדר לנו בתוך פונקציה (ComputeTrace) ולכן ה-"scope" שלו הוא רק בתוך הפונקציה הזאת.

2) IAR:

Mat2	<array>	Memory:0x26C	int[10][10]
[0]	<array>	Memory:0x26C	int[10]
[1]	<array>	Memory:0x280	int[10]
[2]	<array>	Memory:0x294	int[10]
[3]	<array>	Memory:0x2A8	int[10]
[4]	<array>	Memory:0x2BC	int[10]
[5]	<array>	Memory:0x2D0	int[10]
[6]	<array>	Memory:0x2E4	int[10]
[7]	<array>	Memory:0x2F8	int[10]
[8]	<array>	Memory:0x30C	int[10]
[9]	<array>	Memory:0x320	int[10]
[0]	90	Memory:0x320	int
[1]	91	Memory:0x322	int
[2]	92	Memory:0x324	int
[3]	93	Memory:0x326	int
[4]	94	Memory:0x328	int
[5]	95	Memory:0x32A	int
[6]	96	Memory:0x32C	int
[7]	97	Memory:0x32E	int
[8]	98	Memory:0x330	int
[9]	99	Memory:0x332	int

הכתובת היא – 0x26C
אפשר לראות שהמטריצה מכסה מכתובת 0x26C עד לכתובת 0x332 כולל.
סה"כ C6 כתובות בהקסא שזה 198 . ונוסיף 2 בייטים עבור המילה האחרונה במערך.
סה"כ המטריצה לוקחת 200 בייטים של כתובות. נבין שזה 100 תאים במערך וכל אחד דורש שני בייטים כי במעבד זה int הוא 2 בייטים.
כתובת זו היא של ה- Stack frame כי מטריצה זו מוגדרת בפונקציה של ה-main.

:CCS

Mat2	int[10][10]	[[0,0,0,0,0...],[0,0,0,0,0...],[0,0,0,0,0...],[0,0,0,0,0.....	0x04C6
[0]	int[10]	[0,0,0,0,0...]	0x04C6
[1]	int[10]	[0,0,0,0,0...]	0x04DA
[2]	int[10]	[0,0,0,0,0...]	0x04EE
[3]	int[10]	[0,0,0,0,0...]	0x0502
[4]	int[10]	[0,0,0,0,0...]	0x0516
[5]	int[10]	[0,0,0,0,0...]	0x052A
[6]	int[10]	[0,0,0,0,0...]	0x053E
[7]	int[10]	[0,0,0,0,0...]	0x0552
[8]	int[10]	[0,0,0,0,0...]	0x0566
[9]	int[10]	[0,0,0,0,0...]	0x057A
[0]	int	0	0x057A
[1]	int	0	0x057C
[2]	int	0	0x057E
[3]	int	0	0x0580
[4]	int	0	0x0582
[5]	int	0	0x0584
[6]	int	0	0x0586
[7]	int	0	0x0588
[8]	int	0	0x058A
[9]	int	0	0x058C

גם פה נשים לב כי טווח הכתובות הוא בין 0x4C6 ל-0x58C שזה C6 וכאמור זה 198.

3) :IAR

```
SP          = 0x03FE
```

:CCS

SP	0x03FE	Core

4) :CCS

נשים לב ראשית ששמנו ברייק פוינט בנקודה המתאימה :

```
55 //-----------------------------------------------------------------
56 int ComputeTrace(int Mat[M][M]){
57     int Trace=0,i;
58     for(i=0 ; i<M ; i++) Trace += Mat[i][i];
59     return Trace;
60 }
```

וערך ה-SP הינו:

SP	0x026A	Core
SR	0x0003	Core

0x026A

:IAR

שוב שמנו ברייק פוינט במקום המתאים:

```
55 //-----------------------------------------------------------------
56 int ComputeTrace(int Mat[M][M]){
57     int Trace=0,i;
58     for(i=0 ; i<M ; i++) Trace += Mat[i][i];
59     return Trace;
60 }
61 //-----------------------------------------------------------------
62 //                    DiagonalsMax Computation
```

```
00C2AA    4030 C312        br      #0x
int ComputeTrace(int Mat[M][M]){
ComputeTrace:
00C2AE    120A             push.w  R10
00C2B0    120B             push.w  R11
int Trace=0,i;
00C2B2    430E             clr.w   R14
for(i=0 ; i<M ; i++) Trace += Mat[i][i];
```

והערך הינו:

```
SP          = 0x026A
```

5) CCS:

נשים לב ראשית שעשינו Disassembly בשביל לראות את הכתובות:

```
 98     void FillMatrix(int Mat[M][M]){
        FillMatrix():
c288:     120A                PUSH    R10
c28a:     1209                PUSH    R9
c28c:     1208                PUSH    R8
c28e:     4C08                MOV.W   R12,R8
100         for(i=0 ; i<M ; i++){
c290:     430A                CLR.W   R10
c292:     903A 000A           CMP.W   #0x000a,R10
c296:     341E                JGE     ($C$L23)
101             for(j=0 ; j<M ; j++){
        $C$L20:
c298:     430B                CLR.W   R11
c29a:     903B 000A           CMP.W   #0x000a,R11
c29e:     3416                JGE     ($C$L22)
102                 Mat[i][j] = i*M+j;
        $C$L21:
c2a0:     4A0C                MOV.W   R10,R12
c2a2:     403D 000A           MOV.W   #0x000a,R13
c2a6:     12B0 C330           CALL    #__mspabi_mpyi
c2aa:     4C09                MOV.W   R12,R9
c2ac:     5B09                ADD.W   R11,R9
c2ae:     4A0C                MOV.W   R10,R12
c2b0:     403D 0014           MOV.W   #0x0014,R13
c2b4:     12B0 C330           CALL    #__mspabi_mpyi
c2b8:     4B0F                MOV.W   R11,R15
c2ba:     5F0F                RLA.W   R15
c2bc:     5F0C                ADD.W   R15,R12
c2be:     580C                ADD.W   R8,R12
c2c0:     498C 0000           MOV.W   R9,0x0000(R12)
101             for(j=0 ; j<M ; j++){
c2c4:     531B                INC.W   R11
c2c6:     903B 000A           CMP.W   #0x000a,R11
c2ca:     3BEA                JL      ($C$L21)
100         for(i=0 ; i<M ; i++){
        $C$L22:
c2cc:     531A                INC.W   R10
c2ce:     903A 000A           CMP.W   #0x000a,R10
c2d2:     3BE2                JL      ($C$L20)
105     }
        $C$L23:
c2d4:     4030 C35E           BR      #__mspabi_func_epilog_3
 56     int ComputeTrace(int Mat[M][M]){
```

נבחין שהפונקציה כתובה מהשורה – 0xC288 עד 0xC2D4 וזה 0x4C וזה 76 בייטים.

זיכרון זה הוא זיכרון Flash או מה שנקרא Code segment. האזור בזיכרון בו שמורות הפקודות של התוכנית.

IAR:

תחילת הפונקציה:

```
FillMatrix:
 00C262    120A          push.w  R10
 00C264    120B          push.w  R11
 00C266    1208          push.w  R8
for(i=0 ; i<M ; i++){
```

סוף הפונקציה:

```
for(i=0 ; i<M ; i++){
 00C2A0    903E 000A          cmp.w    #0xA,R14
 00C2A4    3402               jge      0xC2AA
for(j=0 ; j<M ; j++){
 00C2A6    430D               clr.w    R13
 00C2A8    3FF7               jmp      0xC298
}
 00C2AA    4030 C312          br       #0xC312
int ComputeTrace(int Mat[M][M]){
```

וטווח הכתובות הוא מ-0xC262 ועד 0xC2AA וסה"כ 0x48 ובדצימלי 72 בייט.

6) ב- IAR ביצענו עצירה של התוכנית בעזרת BreakPoint בפקודה הראשונה של הפונקציה ועוד BreakPoint בפקודה האחרונה של הפונקציה. זמן הריצה הוא ההפרש בין השתיים. כלומר:
END-START = 3013-77 = 2936

7) נבחין כי משתנה זה מוגדר בפונקציה של ה-main. ולכן ה-scope של המשתנה הוא בפונקציית ה-main בלבד. כלומר הוא משתנה לוקאלי.
מיקומו בזמן ה- scope הוא על המחסנית (כיאה למשתנה לוקאלי).

8) <u>CCS:</u>
נבחין שעצרנו ב- Disassembly וקיבלנו את הקוד באסמבלי:

```
 35          maxTrace = mat1Trace > mat2Trace ? mat1Trace : mat2Trace;
c07a:   9A0C                CMP.W   R10,R12
c07c:   3401                JGE     ($C$L7)
c07e:   4A0C                MOV.W   R10,R12
      $C$L7:
c080:   4C82 0202           MOV.W   R12,&maxTrace
 36          Selector = 0;
```

<u>IAR:</u>

```
        maxTrace = mat1Trace > mat2Trace ? mat1Trace : mat2T...
00C05A    9A0C                cmp.w    R10,R12
00C05C    3403                jge      0xC064
00C05E    4A82 0200           mov.w    R10,&maxTrace
00C062    3C02                jmp      0xC068
00C064    4C82 0200           mov.w    R12,&maxTrace
```

תודה!