## חלק תיאורטי – עבור IAR.

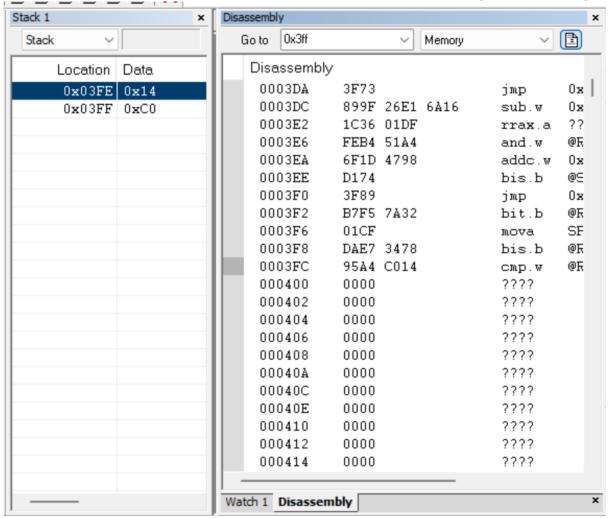
1. משתנה הינו מקום בזכרות שיכול להכיל מידע ממגוון סוגים, אשר נעשה בו שימוש בקטע מהקוד, או בקוד כולו. החלק בקוד בו נעשה שימוש במשתנה נקרא הסקופ (scope) של המשתנה.

משתנה גלובלי הינו משתנה שמוכר לכל חלק בקוד. משתנה כזה יוגדר בצורה שכל הפונקציות יוכלו לגשת אליו ולשנות אותו, הסקופ של משתנה כזה הוא כל הקוד. משתנה לוקאלי או מקומי הוא משתנה שיוגדר בפונקציה ספציפית ורק בתוכה נוכל לגשת אליו ולשנות אותו. יתרון אחד של משתנה כזה הוא שניתן "למחזר" אותו, כגון המשתנה i שבא לידי שימוש בלולאות שונות לכל אורך הקוד, מבלי לדרוס אחד את השני. דוגמה למשתנה גלובלי בקוד לדוגמה הוא maxTrace הסקופ שלו הוא כל הקוד. דוגמה למשתנה לוקלי הוא selector, הסקופ שלו הוא פונקציית ה-main.

20. כתובת המערך Mat2 בזיכרון היא 0x26C וטווח הכתובות הוא 200 כתובות או 0x40. סוג זיכרון זה הוא Stack מכיוון שהמשתנה הוא לוקלי בתוך פונקציית

車	Mat2	<array></array>	Memory: 0x26C
	[0]	<array></array>	Memory:0x26C
	[1]	<array></array>	Memory:0x280
	[2]	<array></array>	Memory:0x294
±	[3]	<array></array>	Memory: 0x2A8
±	[4]	<array></array>	Memory: 0x2BC
±	[5]	<array></array>	Memory:0x2D0
±	[6]	<array></array>	Memory: 0x2E4
±	[7]	<array></array>	Memory:0x2F8
±	[8]	<array></array>	Memory:0x30C
····	[9]	<array></array>	Memory:0x320
	[0]	15171	Memory:0x320
	[1]	31089	Memory:0x322
	[2]	-23927	Memory:0x324
	[3]	-15876	Memory:0x326
	[4]	-23963	Memory:0x328
	[5]	28105	Memory: 0x32A
	[6]	-30253	Memory:0x32C
	[7]	-23219	Memory:0x32E
	[8]	-4619	Memory:0x330
İ	[9]	8493	Memory:0x332

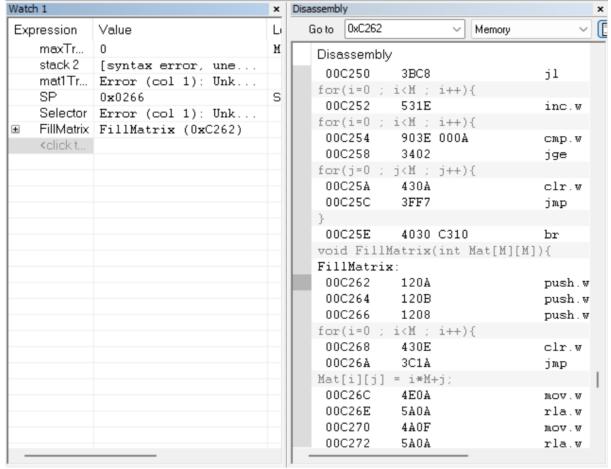
3. מיקום המחסנית הנקבע ע"י המהדר הוא 0x3FF.



4. תוכן הPC כאשר הPC מצביע על השורה הראשונה בפונקציית PC מצביע על השורה הראשונה בפונקציית PC מצביע על השוראת לפונקציה לראשונה. זאת מכיוון שכאשר הקוד ירוץ על כל computeTrace, כתובת הפקודה שקוראת לפונקציה לראשונה. זאת מכיוון שכאשר הקוד ירוץ על כל הפונקציה, הוא יטען את הכתובת לPC וכך התוכנה תחזור לעבוד מאיפה שהפונקציה נקראה.

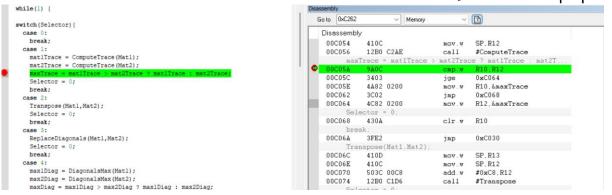
Register ×						
CPU Registers		~				
PC	=	0xC2B2				
SP	=	0x0266				
⊞SR	=	0x0000				
R4	=	0x5DC1				
R5	=	0x2810				
R6	=	0x3A30				
R7	=	0x7D95				
R8	=	0x1F47				
R9	=	0x2AFB				
R10	=	0x0001				
R11	=	0x3774				
R12	=	0x0334				
R13	=	0x000A				
R14	=	0x000A				
R15	=	0x0000				
CYCLECOUNTER	=	3046				
CCTIMER1	=	3046				
CCTIMER2	=	3046				
CCSTEP	=	14				

watch היא 262 כמו שניתן לראות בחלון הא FillMatrix היא 6xC262. הכתובת של הפונקציה :disassembly



טווח הכתובות של הפונקציה היא 0xC262-0xC2AA אורך הפונקציה היא 0x48 כתובות. הפונקציה מצויה בזיכרוו ה-Flash.

- בכדי להבין מהו זמן ריצת הפונקציה נציב breakpoint בפקודה הראשונה של הפונקציה.
   בנוסף לכך, נציב breakpoint נוסף בפקודה האחרונה של ה פונקציה ונאמר כי ההפרש בניהם הוא זמן ריצת הפונקציה.
- ${\tt CYCLECOUNTER@End-CYCLECOUNTER@Start=3013-86=2927} {\tt MCLK}$
- הוא משתנה לוקאלי שממוקם בתוך פונקציית הmain ולכן הסקופ של המשתנה .7 main הוא השתנה לוקאלי שממוקם בתוך פונקציית הmain בזמן הscope. מיקומו של המשתנה הוא במחסנית (המכילה משתנים לוקאלים) בהתאם לתצורת הזיכרון.
- disassembly על השורה הזאת בקוד, נריץ ב-debugger על השורה הזאת בקוד, נריץ ב-Breakpoint מהן פקודות האסמבלי עבור שורה זאת:



## חלק תיאורטי – עבור CCS.

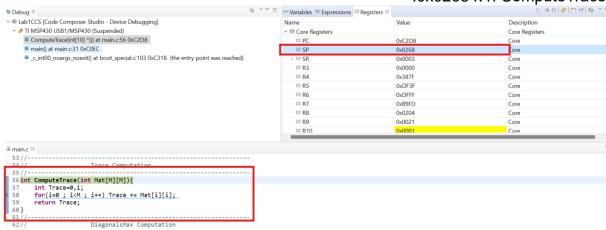
2. כתובת מערך Mat2 הינה 0x04C6 ואורך המערך הוא 200 כתובות, 2 בתים לכל ערך במטריצה. סוג זיכרון זה הוא זיכרון דינאמי.

	-	· · · · · · · · · · · · · · · · · · ·	•
Name	Туре	Value	Location
∨ 🥏 Mat2	int[10][10]	[[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
v 🥏 [9]	int[10]	[0,0,0,0,0]	0x057A
(×)= [O]	int	0	0x057A
(×)= [1]	int	0	0x057C
(×)= [2]	int	0	0x057E
(x)= [3]	int	0	0x0580
(×)= [4]	int	0	0x0582
(x)= [5]	int	0	0x0584
(x)= [6]	int	0	0x0586
(×)= [7]	int	0	0x0588
(x)= [8]	int	0	0x058A
(×)= [9]	int	0	0x058C

3. את תחילית המחסנית נמצא באמצעות הביטוי "stack\_end", המראה 3



4. תוכן רגיסטר SP כאשר רגיסטר PC מצביע על הפקודה הראשונה לביצוע בפונקציית ComputeTrace.

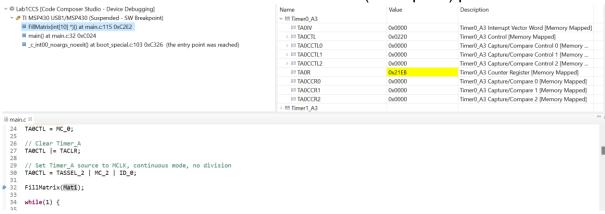


5. באמצעות חלון הdisassembly ניתן לראות שכתובת הפונקציה disassembly בזיכרון היא א 2 באמצעות חלון הOxC2D4 בזיכרון היא נמשכת עד הכתובת 0xC2D4, כלומר 0x4C, או

שמורה ב-Flash.

```
98
      void FillMatrix(int Mat[M][M]){
      FillMatrix():
                              PUSH
                                       R10
c288:
        120A
        1209
                                       R9
c28a:
                              PUSH
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
        341E
                                       ($C$L23)
c296:
                              JGE
101
               for(j=0; j<M; j++){
      $C$L20:
```

6. את זמן הריצה של הפונקציה נמצא באמצעות שתי break points,בתחילת ובסוף הפונקציה וע"י שימוש בטיימר TAO ואיפוסו לפני הקריאה לפונקציה. ראינו ברגיסטר המנייה את מספר מחזורי השעון (מסומן בצהוב).



- 7. mat2Trace הוא משתנה לוקאלי שממוקם בתוך פונקציית הmain ולכן הסקופ של scope. בזמן הפספית (המכילה scope. בזמן המשתנה הוא במחסנית (המכילה משתנים לוקאלים) בהתאם לתצורת הזיכרון.
  - 8. נשים breakpoint על שורת הקוד הנ"ל וניפתח את חלון הdisassembly:

```
42
             maxTrace = mat1Trace > mat2Trace ? mat1Trace : mat2Trace;
c084:
         990C
                               CMP.W
                                        R9, R12
c086:
         3401
                               JGE
                                        ($C$L7)
         490C
c088:
                               MOV.W
                                        R9,R12
      $C$L7:
c08a:
         4C82 0202
                               MOV.W
                                        R12,&maxTrace
 44
             break;
c08e:
         3FE7
                               JMP
                                        ($C$L5)
```