בס"ד

אוניברסיטת בן-גוריון בנגב
Ben-Gurion University of the Negev

# קובץ הכנה ניסוי מעבדה מס' 1 – חלק 2

## Tutorial 1.2 – IAR IDE, Code Running, Debug

מעבדת מיקרומחשבים – המחלקה להנדסת חשמל ומחשבים

מס' קורס - 361.1.3353

כתיבה ועריכה: חנן ריבוא

מהדורה 1 – שנה"ל תשע"ו

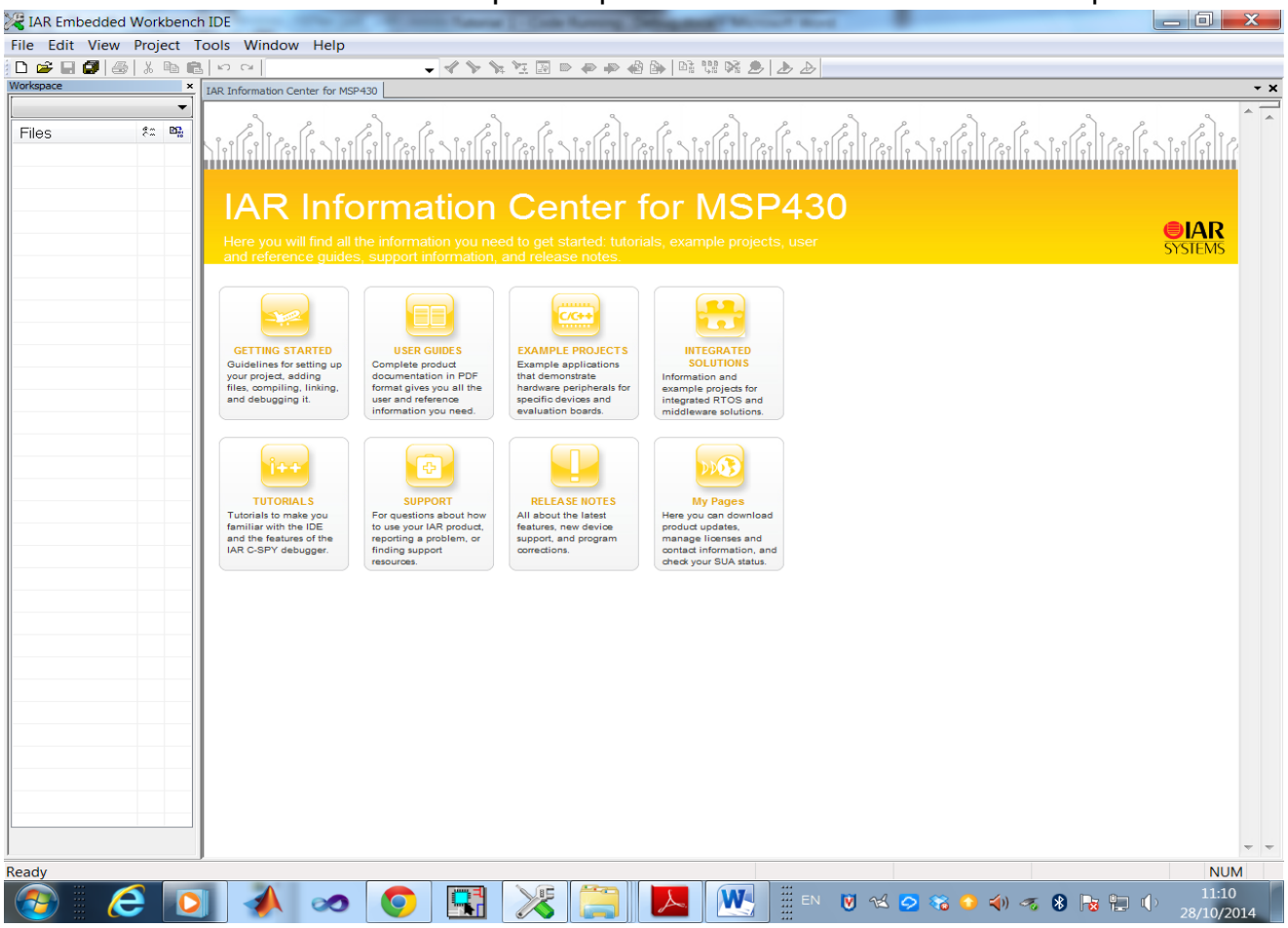
## A. הקדמה:

בתרגול זה נתמקד בהרצת קוד אסמבלי בסביבת הפיתוח ( IDE= Integrated Development Environment) הנקראת IAR . באופן כללי סביבת הפיתוח משמשת לפיתוח קוד מכונה (בינארי) מתוך טקסט (קוד) הנרשם ב-Editor של סביבת הפיתוח שאותו יש לצרוב לתוך זיכרון הבקר לצורך שליטה עליו.
נבצע בהמשך סימולציית הרצת קוד שכתבנו בפתרון תרגול כיתה 1.
בפיתוח קוד בסביבת הפיתוח ישנם 3 שלבים:

1. **סימולציה** – סביבת הפיתוח משמשת סימולטור לבקר שלנו. את הקוד שכתבנו נפעיל ב**מצב סימולטור** והוא ירוץ בפועל ב-PC (מחשב אישי) בלבד, לצורך דימוי הבקר.
2. **Debug** – הקוד שכתבנו ייצרב לבקר (מה-PC) ובהפעלתו הוא ירוץ בבקר ולא ב-PC, אולם ישנה תקשורת בין הבקר ל-PC לצורך תמיכה ב-DEBUG (נקודות עצירה, ריצה בצעדים, בדיקת ערכי רגיסטרים, ערכים בזיכרון וכו').
3. **Active Application** – הקוד שכתבנו ייצרב לבקר (מה-PC) ובהפעלתו הוא ירוץ בבקר בלבד ללא קשר עם ה-PC (בשונה ממצב DEBU). מצב זה מונה גם Stand Alone, מאחר ובמצב זה הבקר בפני עצמו ללא קשר ל-PC.

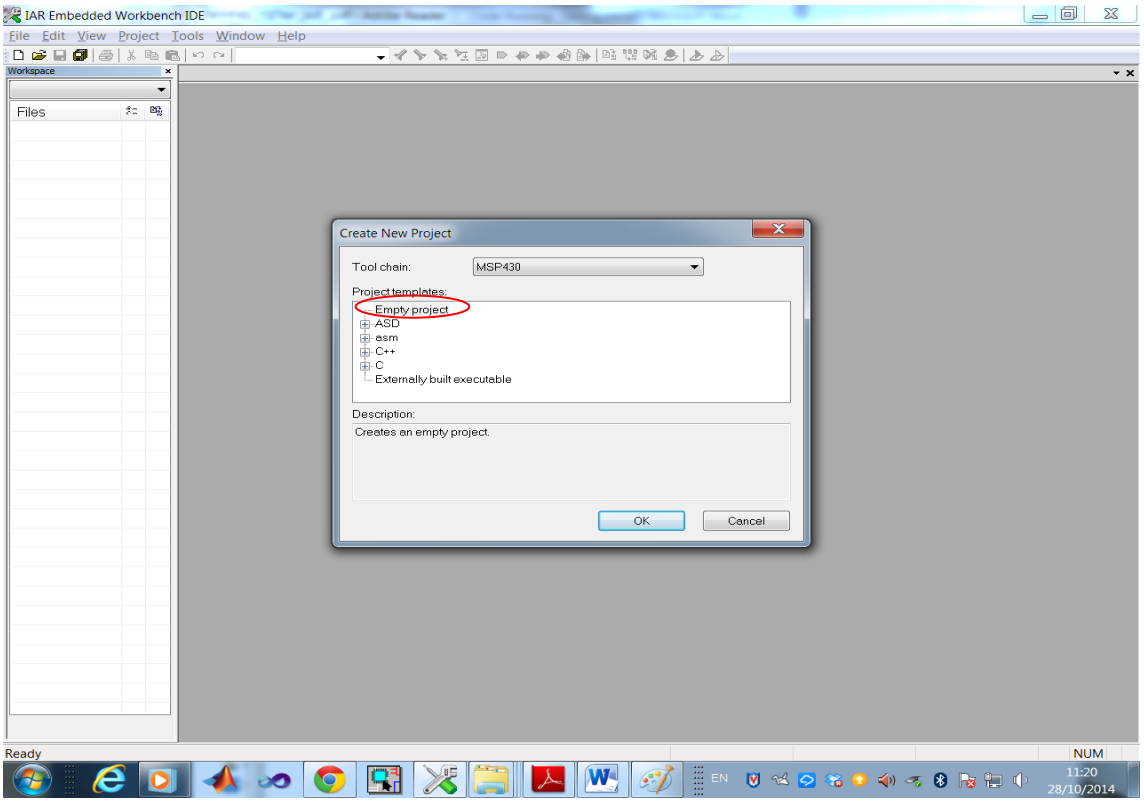
## B. תחילת עבודה בסביבת הפיתוח דורשת **פתיחת פרויקט**. להלן שלבי פתיחת פרויקט של סביבת IAR IDE.

1. עבודה בסביבת הפיתוח במצב סימולטור אפשרית גם בביתכם. אפשר להוריד את קובץ ההפעלה המצורף.
   IAR - IDE setup.zip
2. לאחר ההתקנה נפתח את תוכנת סביבת הפיתוח ומתקבל החלון הבא:

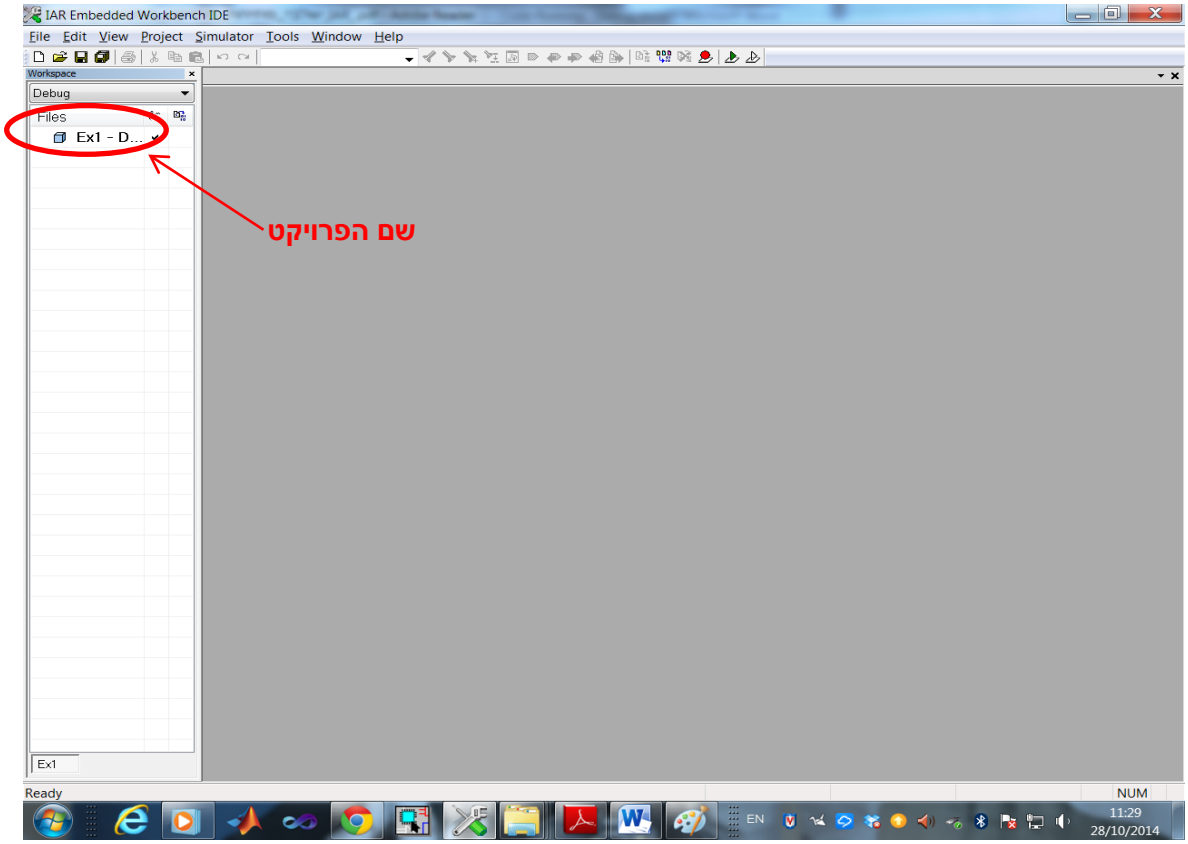



3. **לאחר מכן, רקע החלון נעשה אפור** → Workspace → New → File

4. לאחר מכן, ייפתח החלון הבא → Create New Project → Project




לחצו OK ושמרו את מיקום הפרויקט , במקום בו תבחרו במחשב. לאחר המירה ייפתח החלון הבא (שם הפרויקט שבחרתם יופיע צד שמאל למעלה).

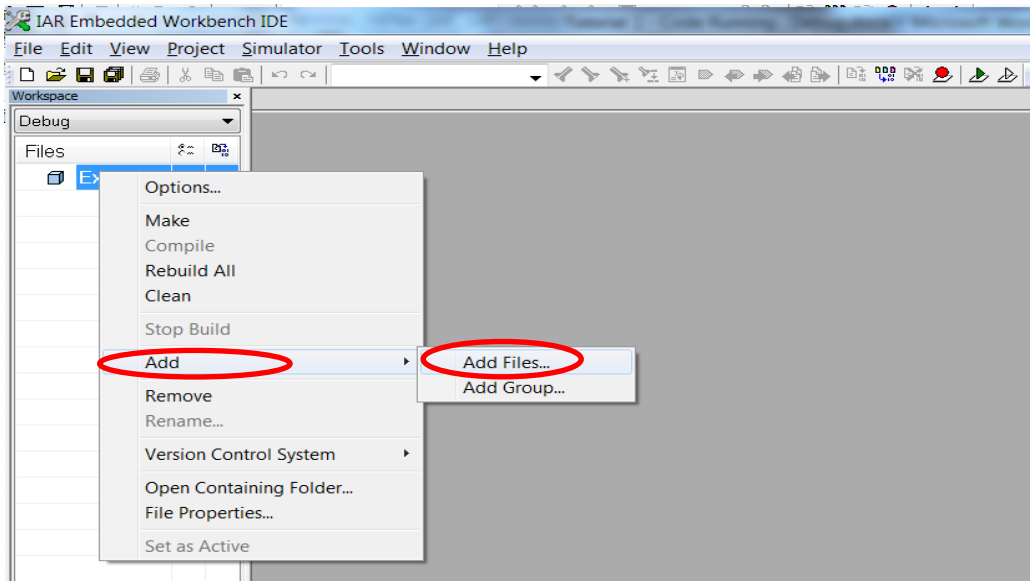


5. כדי לצרף **קובץ מקור** (קובץ קוד) ישנן 2 אפשרויות:

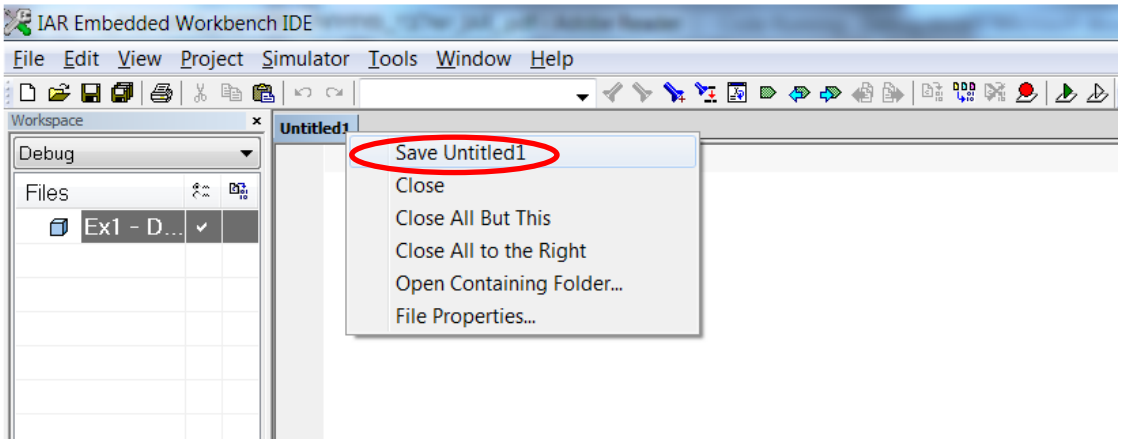
★ **אפשרות 1:** נכין קובץ txt ולתת סיומת s43. (שם הקובץ לפי בחירתנו) לצרף באופן עצמאי לתיקיית הפרויקט ששמרנו בשלב קודם. קובץ עם סיומת s43. מהווה קובץ קוד בשפת אסמבלי, קובץ עם סיומת c. מהווה קובץ קוד בשפת C. בקורס זה נכתוב תוכניות בשפת אסמבלי בלבד.

צירוף קובץ קוד לפרויקט - סדר פעולות:

שמירת קובץ קוד בתיקיית הפרויקט לא מצרפת אותו באופן אוטומטי לפרויקט, לכן נבצע את הפעולה הבאה. לחיצת ימנית בעכבר על שם הפרויקט -> Add -> Add Files -> נפתח חלון ובו יש לבחור את קבצי הקוד s43. אותם נרצה לצרף לפרויקט, בסיום שמות קבצים אלו יופיעו תחת שם הפרויקט.



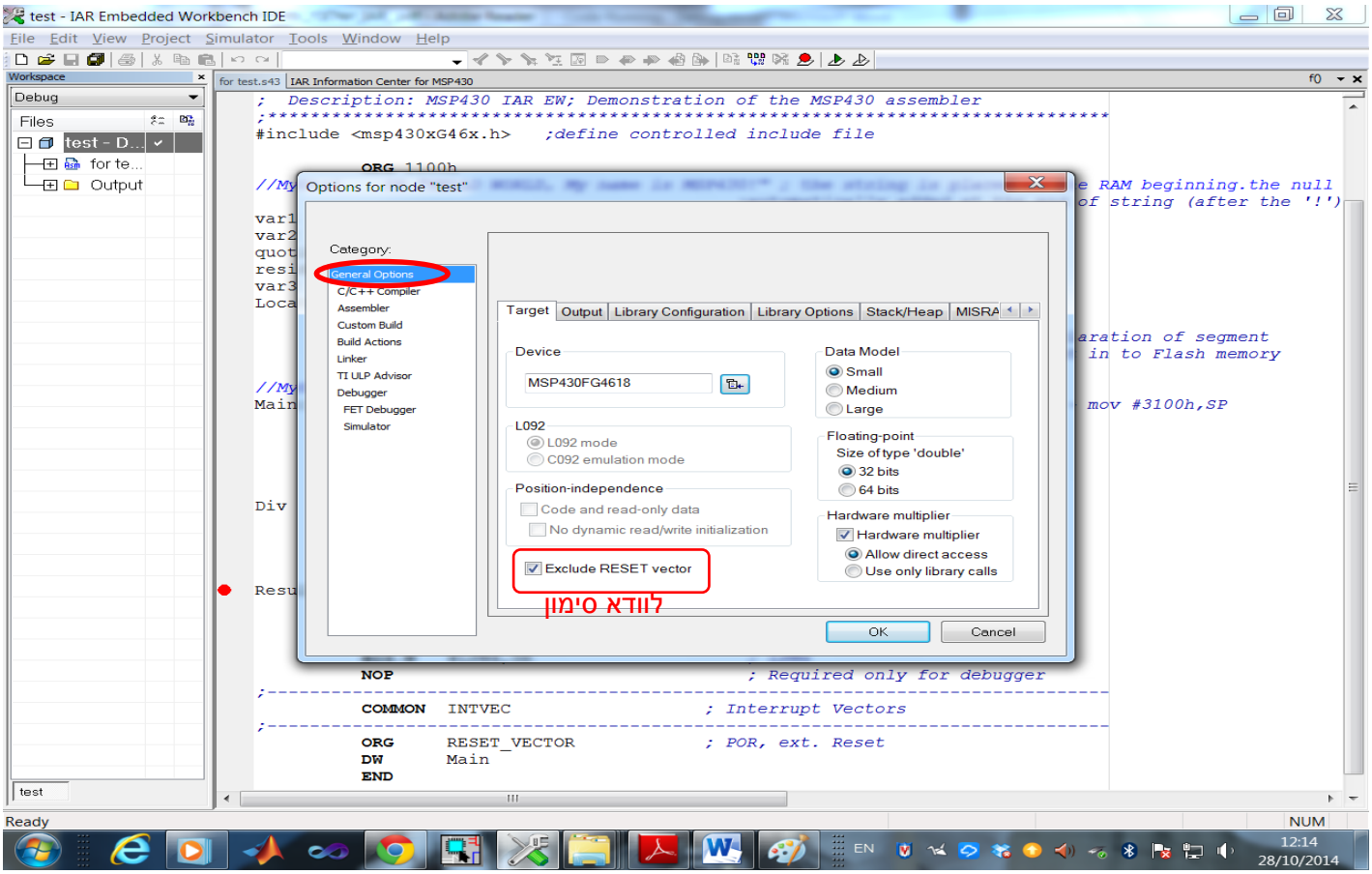

★ **אפשרות 2:** בלחיצה על הלשונית בצורת דף חלק לבן (צד שמאל למעלה) , ייפתח דף חלק לבן ובו נכתוב את קוד התכנית שלנו. לחיצת ימנית בעכבר על שם הקובץ , יש אופציית שמירה.



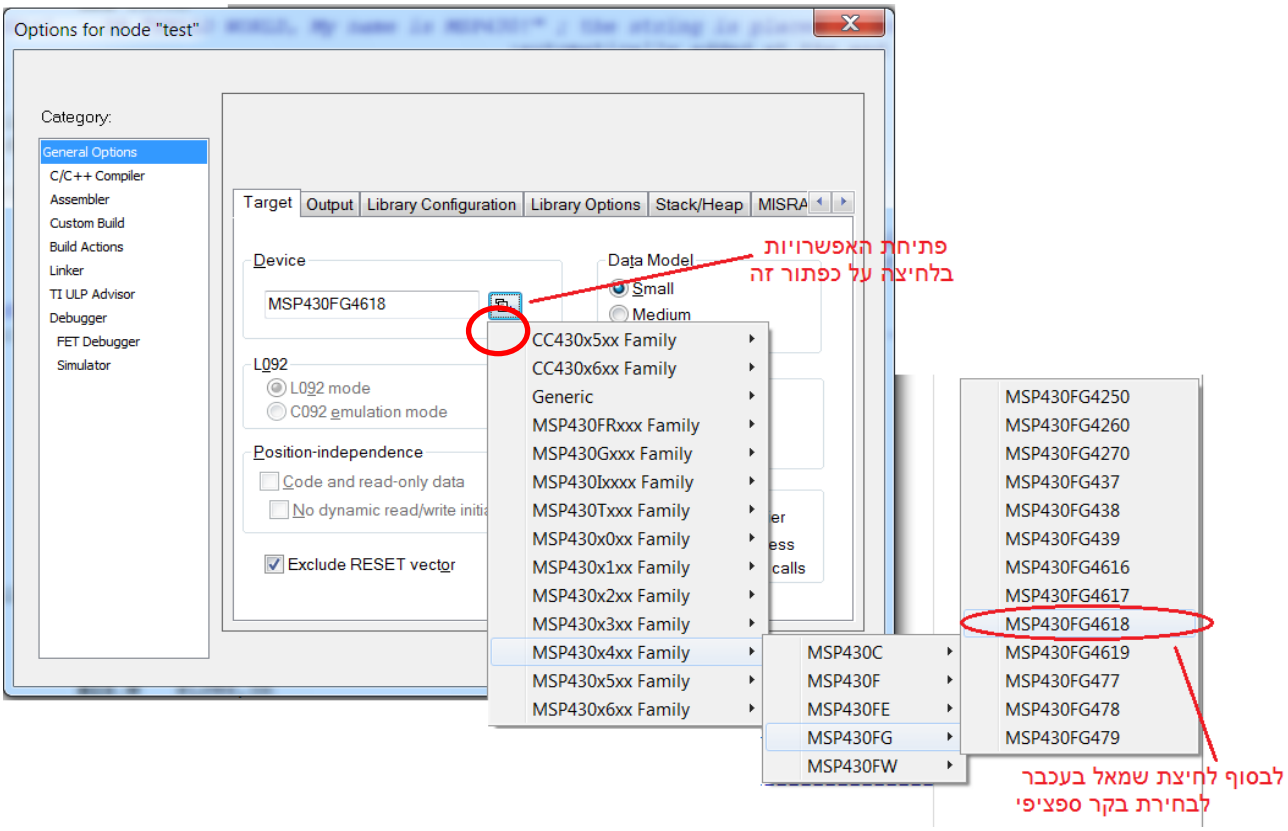

נשמור את הקובץ בתיקיית הפרויקט וניתן לו שם עם סיומת s43. את ההמשך נבצע כמתואר באפשרות1.

6. <u>בחירת פלטפורמה חומרתית של הבקר בסביבת הפיתוח</u>:

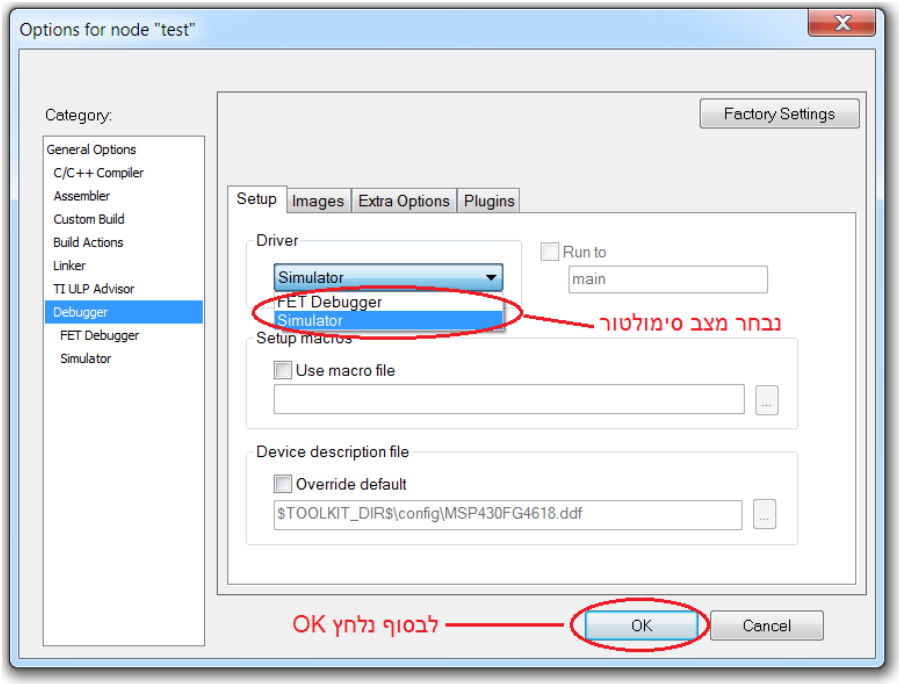
לחיצת ימנית בעכבר על שם הפרויקט -> Options -> נפתח חלון ובו נסמן את הנדרש.




נעדכן בחלון הנ"ל את שדה Device המתאים לבקר שלנו, בלחיצה על הכפתור מימין לשדה זה.



7. **מצב סימולטור** – בחירת סביבת הפיתוח לעבודה במצב סימולטור יש לפעול כך:




בשלב זה שאר ההגדרות יישארו לפי הגדרות ברירת המחדל של היצרן.

8. שמירת ה- Workspace :

**File→ Save Workspace→נפתח חלון לשמירה, יש לשמור בתיקיית הפרויקט**

9. הידור הפרויקט:

**Project → Rebuild All**

(אם שכחתם לבצע שלב קודם, תתבקשו תחילה לשמור את ה-Workspace ורק אח"כ הוא יבצע את בניית הפרויקט).

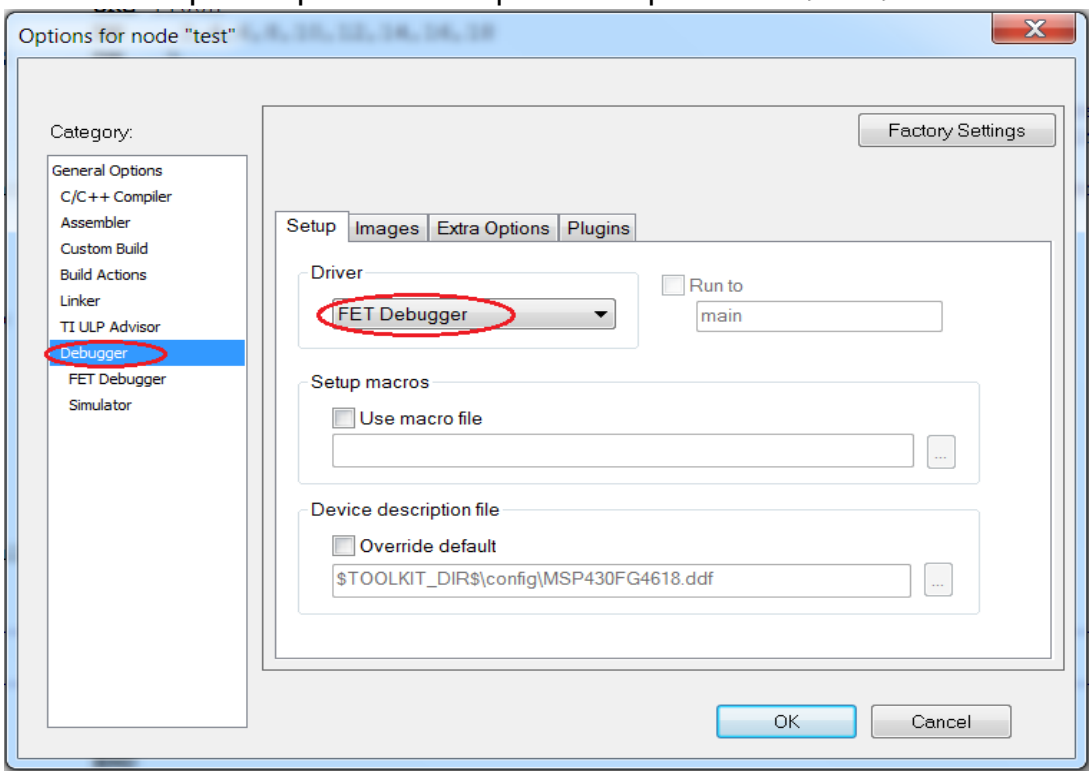
10. לתחילת ביצוע הסימולציה.

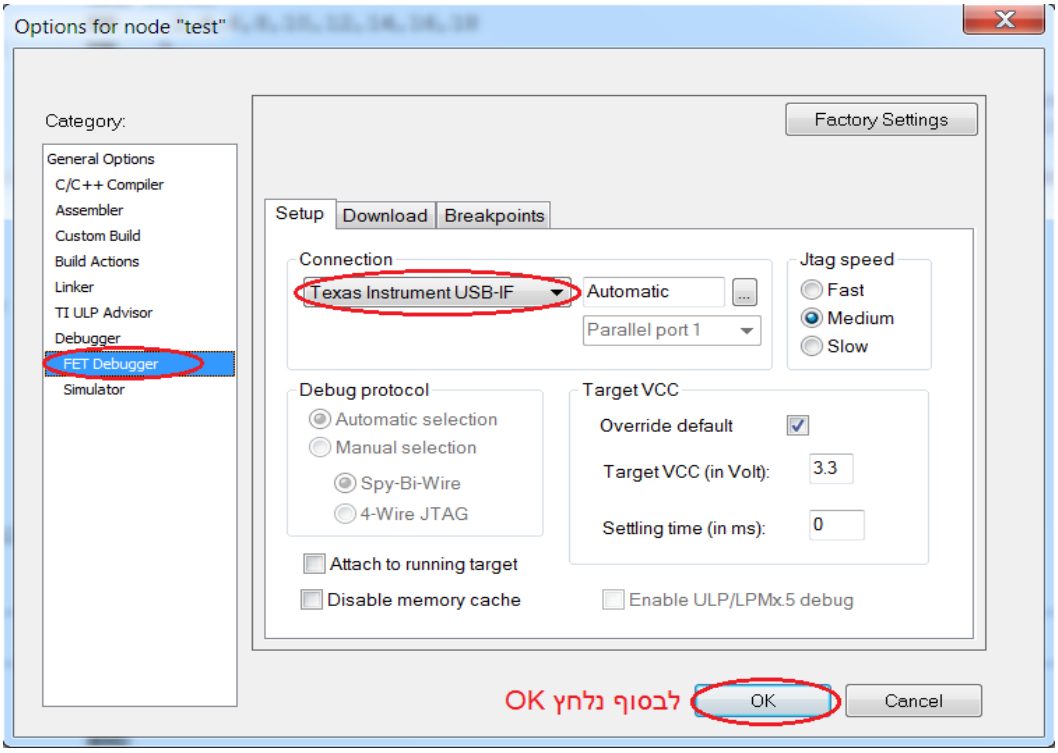
**Project → Download and Debug →** (כמו שלמדתם בקורס JAVA) **נכנסתם למצב סימולציה**

**תוכלו לדלג בשלב זה על סעיפים 11,12 ולעבור לכותרת סימולציית תרגילים – תרגול כיתה מס' 1**

11. **מצב DEBUG** - בחירת סביבת הפיתוח לעבודה במצב DEBUG יש לפעול כך:

לחיצת ימנית בעכבר על שם הפרויקט -> Options -> נפתח חלון ובו נסמן את הנדרש.






נבנה את הפרויקט: Project → Rebuild All

נכנס למצב DEBUG (צריבת קוד לזיכרון הבקר והרצתו בבקר בלבד עם יכולות DEBUG דרך ה-PC):

**נכנסתם למצב DEBUG** (כמו שלמדתם בקורס JAVA) → **Project → Download and Debug**

אפשר להריץ את הקוד כמו במצב סימולציה כמפורט לעיל עם הבדל אחד, מדובר במצב אמת (קוד רץ בבקר ולא ב-PC).

12. מצב **Active Application –**

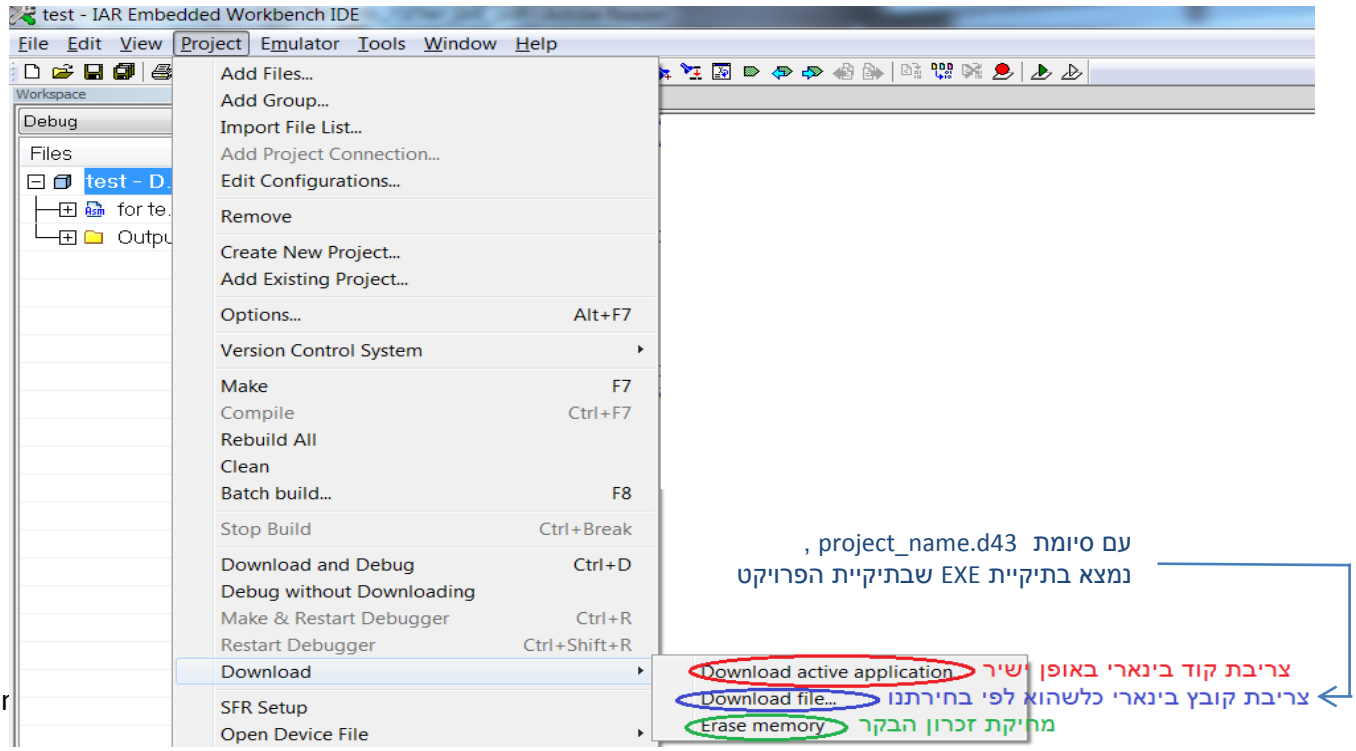
במצב זה נרצה לצרוב את הקובץ הבינארי של התכנית לזיכרון הבקר ומאז הבקר עובד במנותק מה-PC.

מצב זה הוא התוצר הסופי של האפליקציה אותה בצענו. כאשר נרצה לבצע פעולת RESETלבקר, נלחץ על לחצן RESET הנמצא על כרטיס ערכת הפיתוח.

נבנה את הפרויקט: Project → Rebuild All

נבצע צריבת קובץ בינארי לזיכרון הבקר:

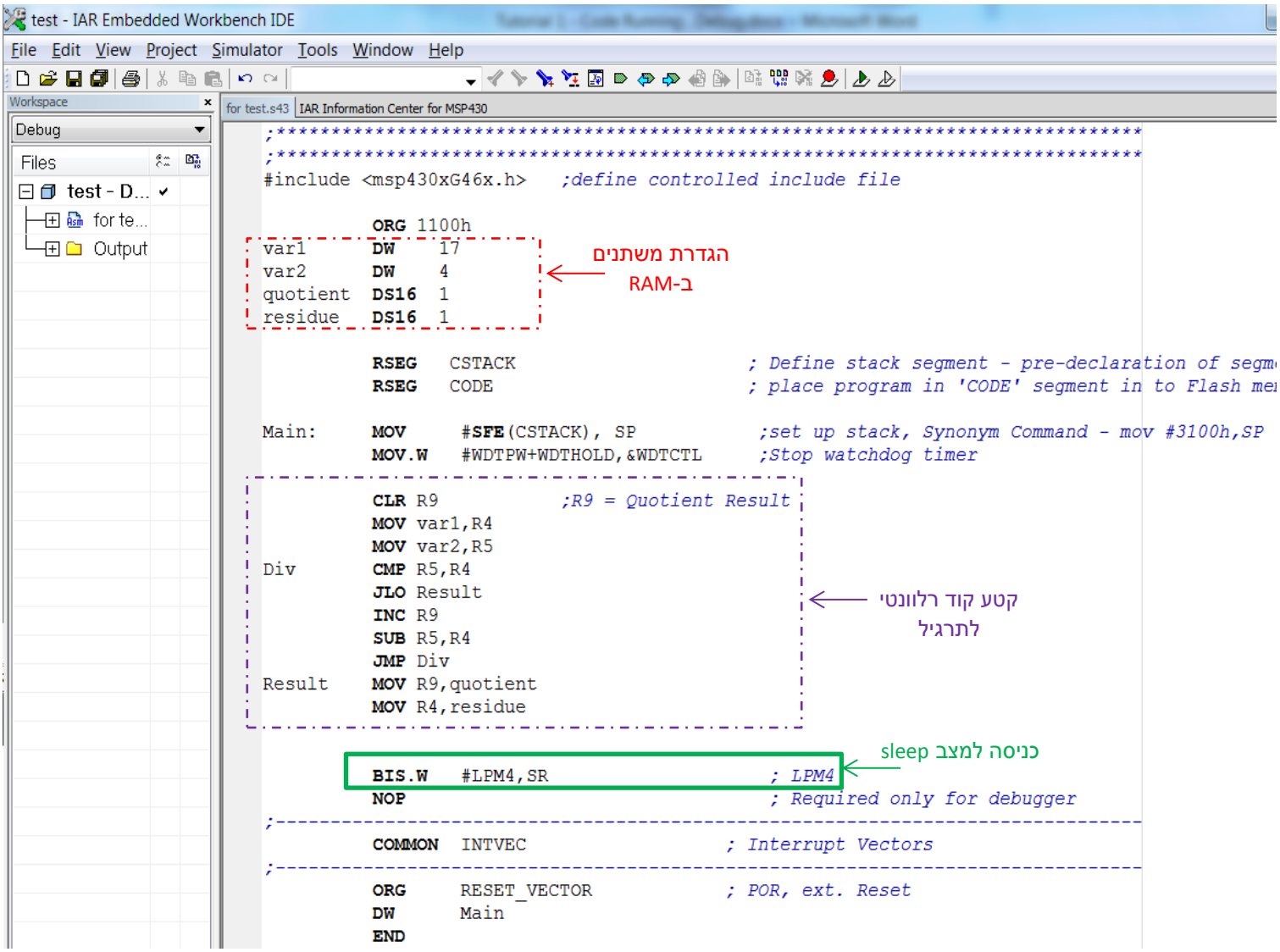
שלוש אפשרויות → Project → Download



# סימולציית תרגילים – תרגול כיתה מס' 1

**1.** חילוק בין מספרים שלמים:

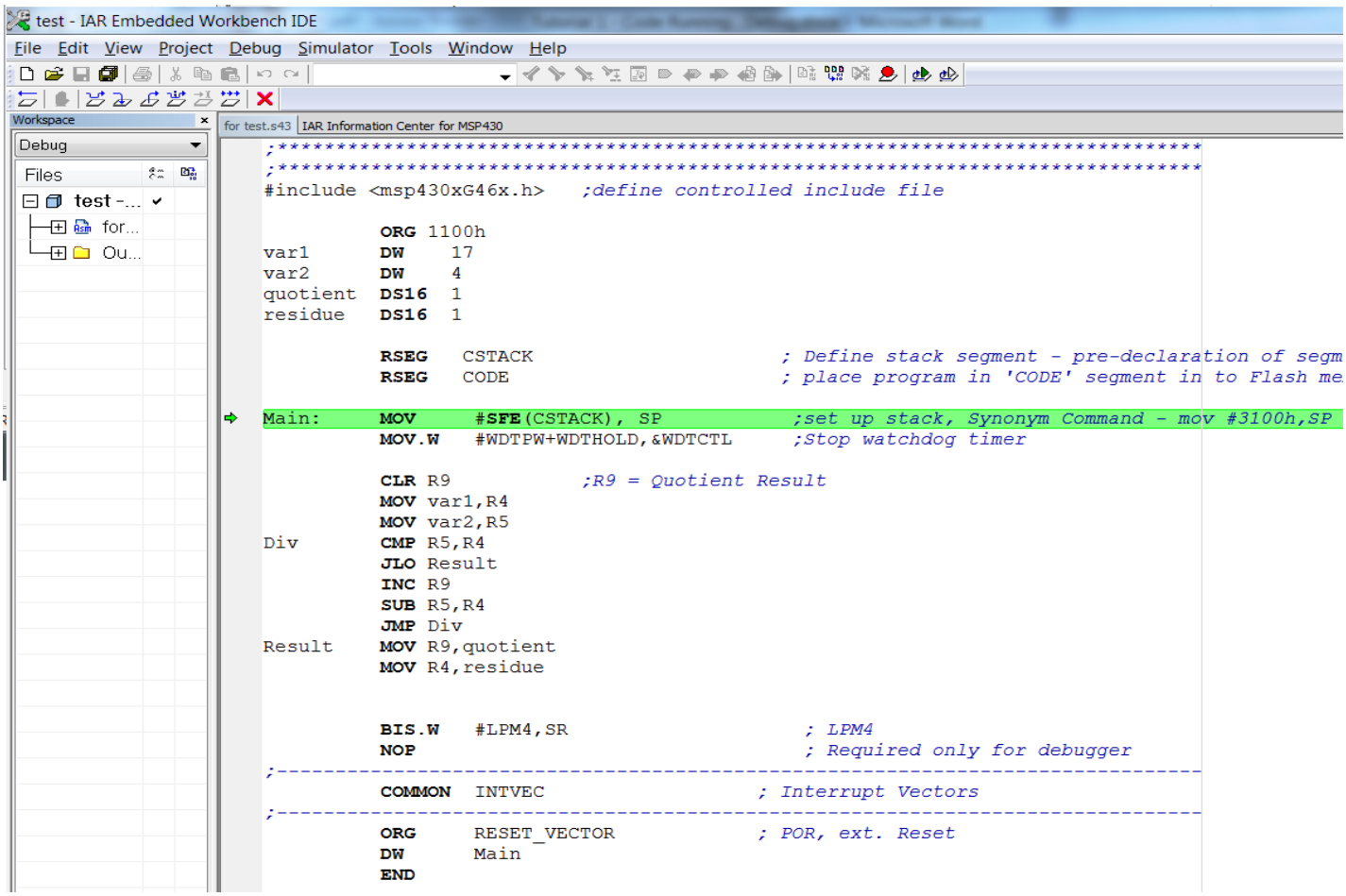
$var1 \div var2 \rightarrow quotient =$ מנה $, residue =$ שארית (ללא שימוש במחלק חומרתי)




באופן כללי מבלי להיכנס לפרטים (תלמדו בהרצאה) הבקר מריץ פקודה אחר פקודה באופן טורי (כתובת אחר כתובת החל מנקודת התחלה מסוימת) מתוך הזיכרון. כאשר הגענו לסיום התכנית נרצה למנוע מהבקר להגיע לאזור בזיכרון שאינו מכיל פקודות ולהימנע מפעולות שאינן רצויות. יש 2 אפשרויות:

- ★ נדרש לוודא שהתוכנית כוללת לולאה אינסופית.
- ★ שימוש בפקודת LPM (תלמדו עליה בהמשך) אשר משמשת לרמות sleep שונות של הבקר.

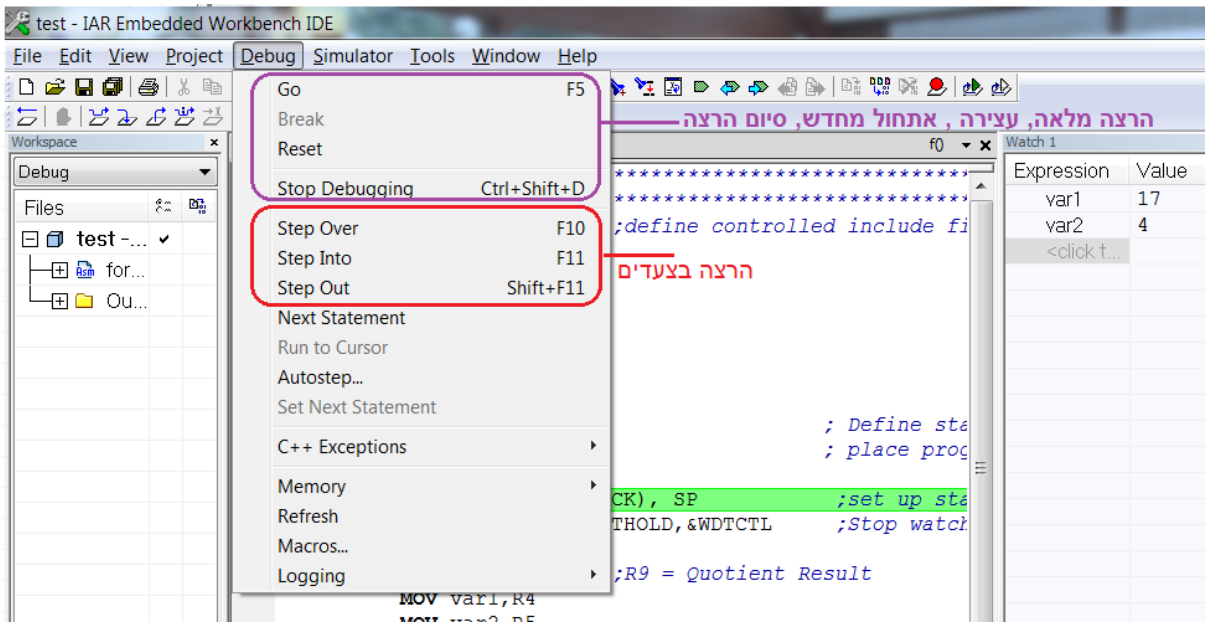
נבצע הרצה לקוד ונסתכל על ערכי הרגיסטרים, משתנים והזיכרון. בכניסה למצב סימולציה המסך נראה כך:שורת הפקודה הצבועה בירוק היא נקודת ההתחלה אליה נגיע במצב של RESET.




נפתח את החלונות הבאים (רגיסטרים, משתנים וזיכרון) כדי לראות את ערכי התוכנית בכל שלב שנרצה.

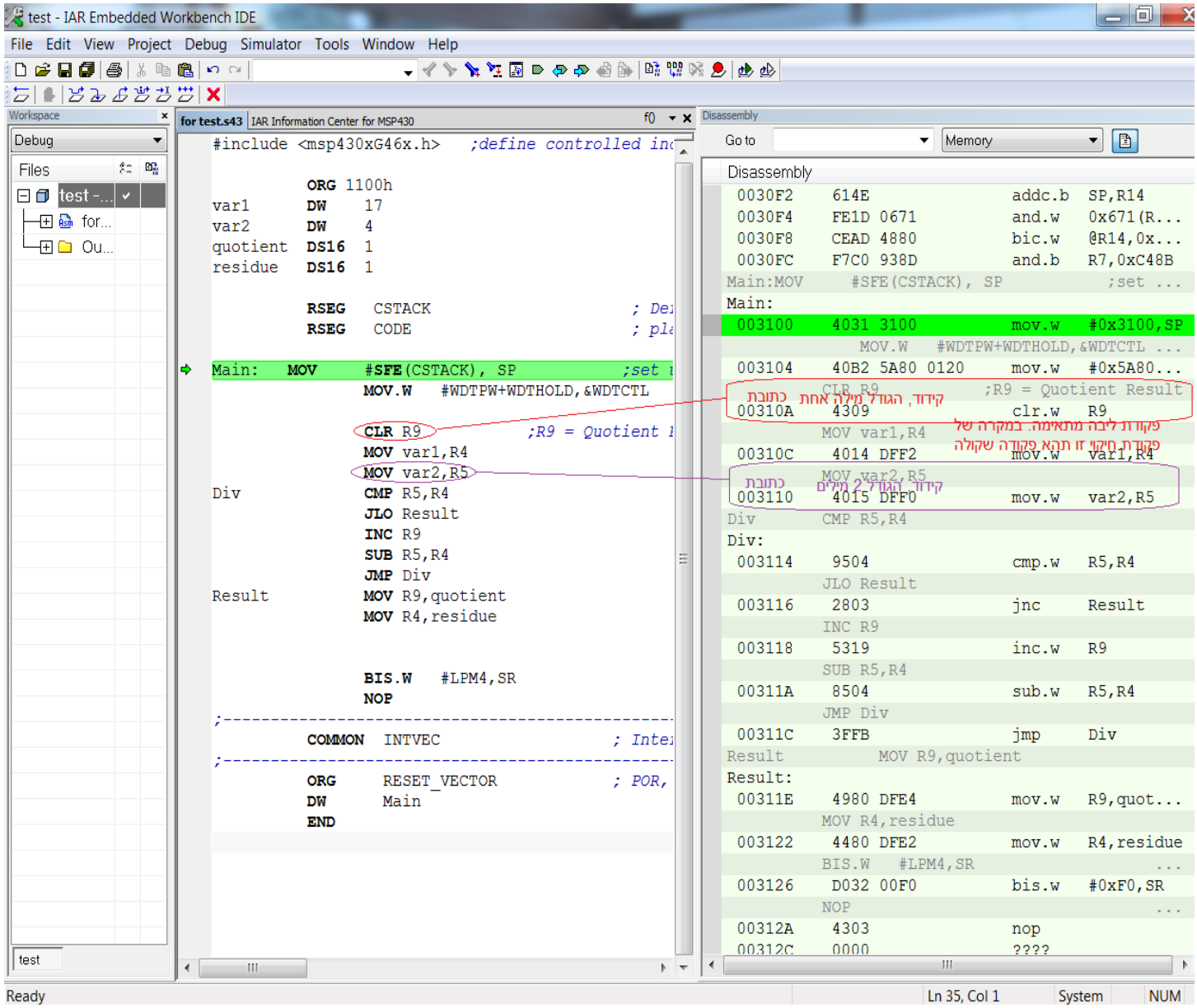
- <u>לפתיחת חלון Disassembly</u>: בחלון זה נראה את מבנה הפקודות לפי **כתובת כל פקודה בזיכרון , גודלה והקידוד שלה לקוד מכונה**. **View → Disassembly**
- <u>לפתיחת חלון רגיסטרים</u>: **View → Register**
- <u>לפתיחת חלון משתנים</u>: **View → Watch → Watch1 ÷ Watch4**
- <u>לפתיחת חלון הזיכרון</u>: **נבחר בלשונית לראות את זיכרון ה- RAM→ View → Memory**

אנו מעוניינים כעת רק בחלונות **רגיסטרים, משתנים וזיכרון**. תוכלו בזמנכם החופשי לצעוד צעד אחר צעד.

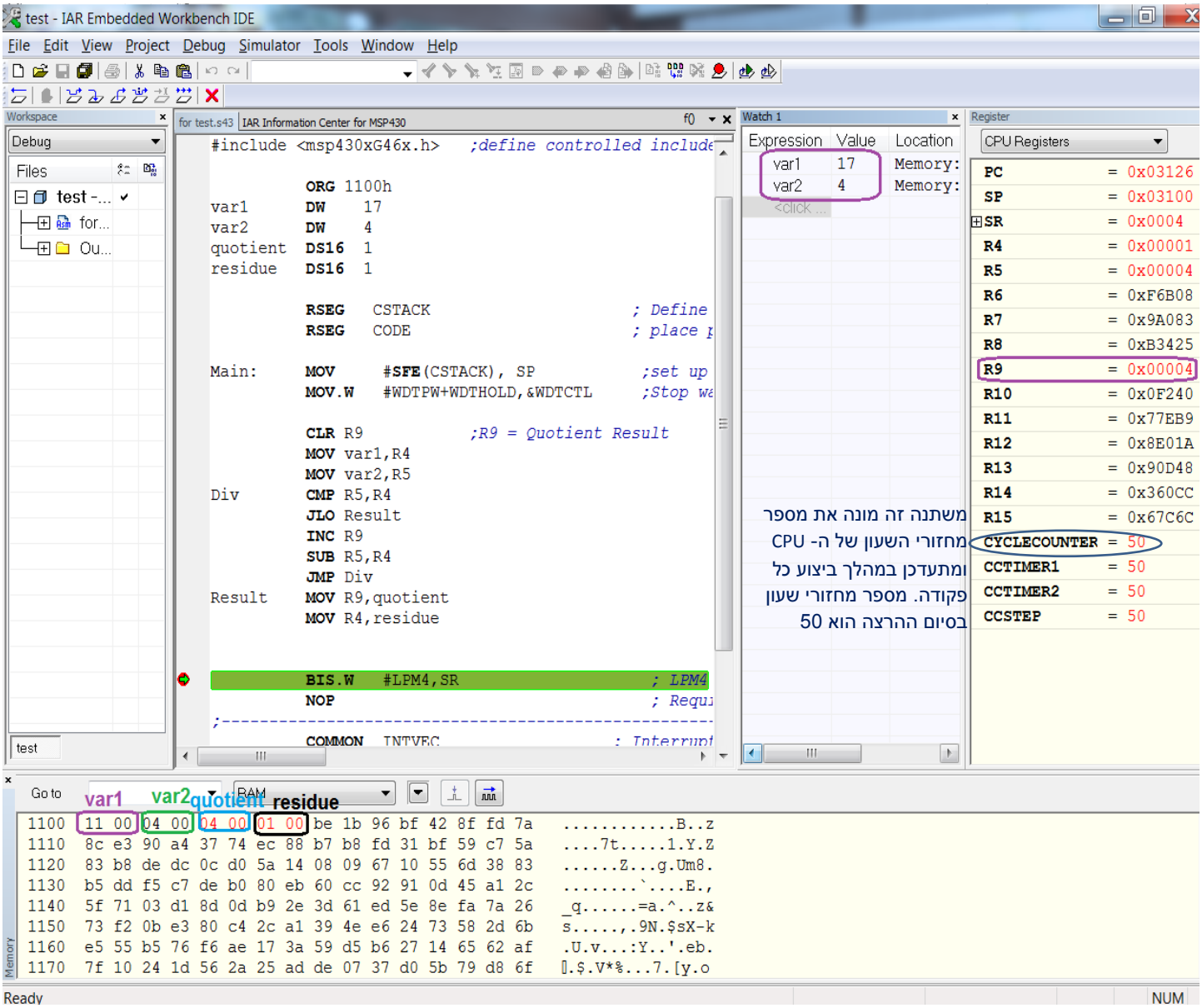


## חלון Disassembly:

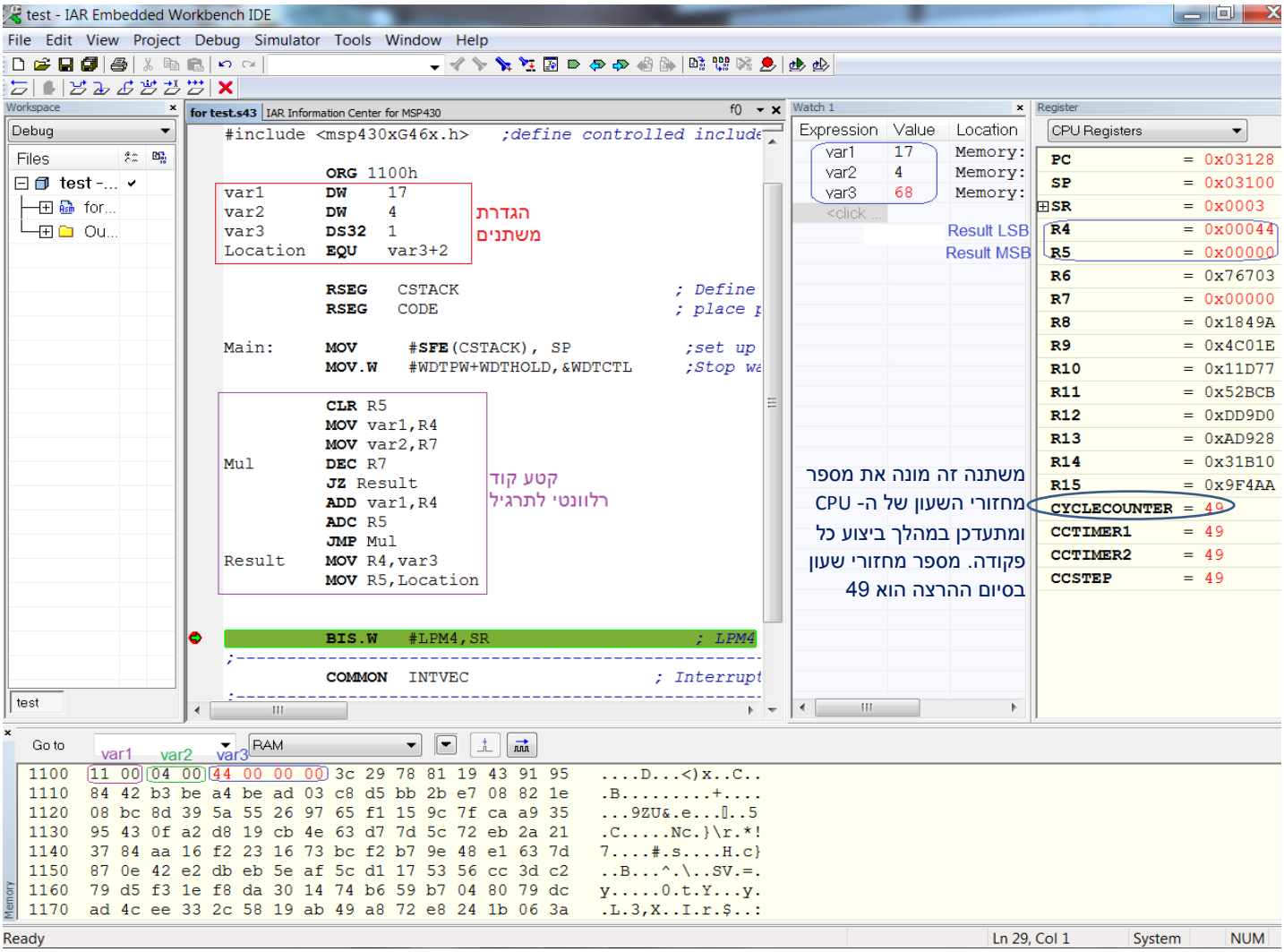
בחלון זה נוכל לראות עבור כל פקודה את כתובתה בזיכרון, גודלה ב-word וקידודה (קוד מכונה).



לבסוף, נשים נקודת עצירה בסוף התוכנית (אחרי הפקודה האחרונה כדי שגם היא תתבצע) ונבצע ריצה מלאה (F5) כדי לראות הערכים בסיומה.



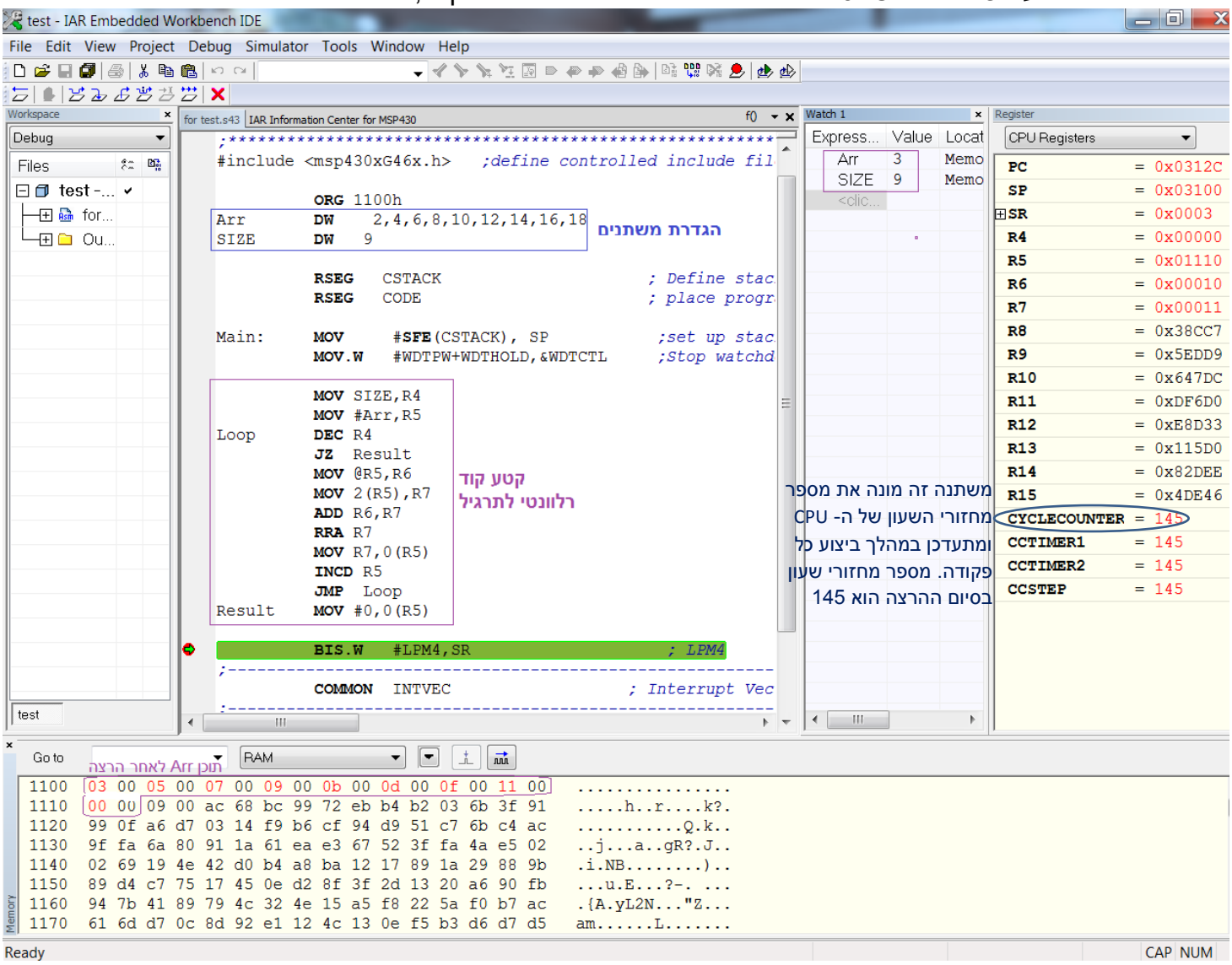
**2.** <u>כפל בין מספרים שלמים</u>: $var1 \cdot var2 \rightarrow var3$ (ללא מכפל חומרתי)
לאחר ביצוע השלבים המפורטים בתרגיל 1. לאחר הרצה מתקבל,




משתנה התוצאה var3 , גודלו 32bit מאחר ותוצאת כפל בין 2 מספרים בגודל 16bit יכולים לגלוש לתחום 32bit. רגיסטר R4 מהווה חלק תחתון של תוצאת המכפלה (Result LSB) ורגיסטר R5 מהווה חלק עליון של תוצאת המכפלה (Result MSB). הפלטפורמה שלנו הינה Little Endian ולכן תוכן R4 מועבר לזיכרון בכתובת var3 ורגיסטר R5 מועבר לזיכרון בכתובת var3+2.

3. חישוב ממוצע בין 2 אברים סמוכים במערך בכתובת **Arr** בגודל **SIZE** (לדרוס את מערך המקור ולאפס איבר אחרון).

לאחר ביצוע השלבים המפורטים בתרגיל 1 ולאחר מלאה הרצה מתקבל,




4. עליך לכתוב תוכנית **לחישוב העקבה (TRACE) של מטריצה** סימטרית.

העקבה של מטריצה סימטרית בגודל nxn מוגדרת ע"י סכום איברי האלכסון של המטריצה, כדלקמן:

עבור מטריצה סימטרית $A_{nxn}$ העקבה נתונה ע"י הנוסחה: $\mathbf{Trace(A)=a_{11}+a_{22}+\ldots\ldots+a_{nn}}$

**גודל כל איבר במטריצה הוא 32 סיביות** הערך המתקבל עבור עקבת המטריצה המחושבת יש לאחסן ברגיסטרים **R12 עד R15** . כאשר החלק הנמוך LSB של העקבה יאוחסן ברגיסטר R12,R13 והחלק הגבוה MSB של העקבה יאוחסן ברגיסטר R14,R15 . כאשר **Matrix** הינה מטריצה ותווית זו היא כתובת האיבר הראשון של המטריצה ו- **Lines** מסמן את מספר השורות של המטריצה.

את תוצאת העקבה יש לאחסן במשתנה בשם Trace.

מטריצה מאוחסנת בזיכרון ע"י מערך של שורות (שורה אחר שורה החל משורה ראשונה).

- Part of Diagonal index = MatrixPTR+4*i+Lines*4j
- 0<= i <=Lines-1 and 0<= j <=Lines-1 for all Parts of Diagonal i=j So, Part of Diagonal index = MatrixPTR+4i(Lines+1)
- Trace=sum of M[MatrixPTR+4i(Lines+1)] when 0<= i <=Lines-1

```
        #include <msp430xG46x.h>   ;define controlled include file

        ORG 1100h
Matrix  DC32  1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16
Lines   DW    4
Trace   DS32  2

        RSEG   CSTACK              ; Define stack segment - pre-declaration of segment
        RSEG   CODE                ; place program in 'CODE' segment in to Flash memory

Main: MOV     #SFE(CSTACK), SP         ;set up stack, Synonym Command - mov #3100h,SP
      MOV.W   #WDTPW+WDTHOLD,&WDTCTL     ;Stop watchdog timer

      mov    #Matrix,R5
      clr    R12
      clr    R13
      clr    R14
      clr    R15
      mov    Lines,R6
      mov    Lines,R7
      inc    R6
      rla    R6
      rla    R6
L1    add    @R5,R12
      adc    R13
      adc    R14
      adc    R15
      add    R6,R5
      dec    R7
      jnz    L1
      mov    #Trace,R5
      mov    R12,0(R5)
      mov    R13,2(R5)
      mov    R14,4(R5)
      mov    R15,6(R5)

      BIS.W   #LPM4,SR              ; LPM4
;-------------------------------------------------------------------------------
        COMMON  INTVEC              ; Interrupt Vectors
;-------------------------------------------------------------------------------
        ORG     RESET_VECTOR        ; POR, ext. Reset
        DW      Main
        END
```

לאחר ביצוע השלבים המפורטים בתרגיל 1 ולאחר מלאה הרצה מתקבל,

```
#include <msp430xG46x.h>    ;define controlled include

        ORG 1100h
Matrix  DC32  1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16
Lines   DW    4
Trace   DS32  2

        RSEG   CSTACK                    ; Define s
        RSEG   CODE                      ; place pr

Main:   MOV     #SFE(CSTACK), SP         ;set up sta
        MOV.W   #WDTPW+WDTHOLD,&WDTCTL   ;Stop watch

        mov     #Matrix,R5
        clr     R12
        clr     R13
        clr     R14
        clr     R15
        mov     Lines,R6
        mov     Lines,R7
        inc     R6
        rla     R6
        rla     R6
L1      add     @R5,R12
        adc     R13
        adc     R14
        adc     R15
        add     R6,R5
        dec     R7
        jnz     L1
```

משתנה זה מונה את מספר מחזורי השעון של ה- CPU ומתעדכן במהלך ביצוע כל פקודה. מספר מחזורי שעון בסיום ההרצה הוא 71.

```
Matrix  1   Memory:0x1
Lines   4   Memory:0x1
Trace   34  Memory:0x1

PC            = 0x03146
SP            = 0x03100
SR            = 0x0003
R4            = 0x0F726
R5            = 0x01142
R6            = 0x00014
R7            = 0x00000
R8            = 0x94B89
R9            = 0x0F426
R10           = 0x82D72
R11           = 0x5A60F
R12           = 0x00022
R13           = 0x00000
R14           = 0x00000
R15           = 0x00000
CYCLECOUNTER  = 71
CCTIMER1      = 71
CCTIMER2      = 71
CCSTEP        = 71

1100  01 00 00 00 02 00 00 00 03 00 00 00 04 00 00 00  ................
1110  05 00 00 00 06 00 00 00 07 00 00 00 08 00 00 00  ................
1120  09 00 00 00 0a 00 00 00 0b 00 00 00 0c 00 00 00  ................
1130  0d 00 00 00 0e 00 00 00 0f 00 00 00 10 00 00 00  ................
1140  04 00 22 00 00 00 00 00 00 00 04 5c a3 88 ef 69  ..".......\...i
1150  db 5c 3b 94 a5 45 a7 62 59 d0 0a 2d 16 58 81 5a  .\;..E.bY..-.X.Z
1160  39 da 7b 06 e6 3f 92 52 ad b5 dc bd 97 57 32 9f  9.{..?.R.....W2.
```

Matrix

Trace

Lines

Trace