Table of Contents

דו"ח מכין, מעבדה מס' 2 – Stack, Routine, Macro

A. חומר עזר:

- ספר מעבדה MSP430x4xx user guide עמודים: 45, 407-414 (ללא עמודים 411,412)
- Tutorial 2.1, Tutorial 2.2 (חומר כתוב + וידאו).

B. חלק תיאורטי:

1. הסבר מהי מחסנית, את הצורך בה ואופן שימושה.
2. הסבר מהי רוטינה את הצורך בה ואופן שימושה וכיצד היא משפיעה על המחסנית.
3. הסבר מהי פונקציית MACRO את הצורך בה ואופן שימושה, רשום טבלת יתרונות וחסרונות בין פונקציית MACRO לבין רוטינה.
4. הגדירו מהו SCOPE של משתנה וכיצד הוא נקבע ? נדרש פירוט והסבר
5. הגדירו את ההבדל בין סוגי האובייקטים הבאים ותנו דוגמה לכל אחד מהם:
   משתנה גלובאלי, משתנה לוקאלי, קבוע בזיכרון, קבוע מסוג text, רגיסטר

C. חלק מעשי – כתיבת קוד באסמבלי בנוסף לכתיבת קוד שקול ב Python:

- קוד באסמבלי נדרש לכתוב בקובץ מקור חדש בשם **pre2.s43**
- קוד שקול ב Python בקובץ מקור **pre2.py** (השתמשו ב Colab ראו רפרנס בתוכן תרגול 2).

כאשר $Id1$, $Id2$ הם שני מערכים בגודל 8 המכילים את מספרי ת"ז (8 ספרות נמוכות), של חברי הקבוצה.

נדרש לכתוב קוד מבוסס פונקציה למימוש הביטוי הנתון בטבלה הבאה המכילה עשר גרסאות שונות בהתאם למספר ת"ז :

נדרש לכתוב פונקציה בשפת אסמבלי (**בשימוש מחסנית בלבד – ראו הסבר מפורט בסעיף E**).

הגרסה לביצוע הינה לפי ספרת האחדות של מספר הזהות הנמוך $Id_i < Id_j$ מבין שני בני הזוג.

נדרש להגדיר את מערכי ת"ז Id1, Id2 מטיפוס int (16 bit) המהווים מערכי מקור, משתנה SIZE מטיפוס int המגדיר גודל מערך ואת משתנה המטרה num מטיפוס int בצורה הבאה:

דוגמה: עבור זוג סטודנטים עם מספרי ת"ז הבאים **315212875** ,ID2=2**0447105**0 ,ID1

מספר הגרסה לביצוע הוא 0.

```
Id1     DW    04471050      // int Id1[8]={0,4,4,7,1,0,5,0};
Id2     DW    15212875      // int Id2[8]={ 1,5,2,1,2,8,7,5};
SIZE    DW    8             // int SIZE=8;
num     DS16  1             // int num;
```

הערה: כיתוב הערות בצבע ירוק נועד לצורך העשרה והבנת הקשר בין שפות עיליות C/C++ המיועדות לתכנות מערכות RT EMBEDDED ולבין שפת אסמבלי.

Version	Function	Note
0	$num = \sum_{i=0}^{SIZE-1} (ID1[i] \cdot ID2[i])$	
1	$num = \sum_{i=0}^{SIZE-1} (ID1[i] + ID2[i])$	
2	$num = \sum_{i=0}^{SIZE-1} (ID1[i]\ \boldsymbol{or}\ ID2[i])$	
3	$num = \sum_{i=0}^{SIZE-1} (ID1[i]\ \boldsymbol{xor}\ ID2[i])$	
4	$num = \sum_{i=0}^{SIZE-1} (ID1[i] - ID2[i])$	
5	$num = \sum_{i=0}^{SIZE-1} (ID1[i]\ \boldsymbol{and}\ ID2[i])$	
6	$num = \sum_{i=0}^{SIZE-1} \boldsymbol{max}(ID1[i], ID2[i])$	
7	$num = \sum_{i=0}^{SIZE-1} \boldsymbol{min}(ID1[i], ID2[i])$	
8	$num = \sum_{i=0}^{SIZE-1} \boldsymbol{min\_odd}(ID1[i], ID2[i])$	If there is no odd ID digit, the result is 0
9	$num = \sum_{i=0}^{SIZE-1} \boldsymbol{max\_even}(ID1[i], ID2[i])$	If there is no even ID digit, the result is 0

D. עקרונות לכתיבת פונקציה בשפת אסמבלי בשימוש מחסנית בלבד:

1. הקדמה:

פונקציה היא למעשה רוטינה המכילה מעטפות קוד התומכות בלפחות אחת מהשניים:

a. תמיכה בשליחת ארגומנטים דרך המחסנית מה caller (קטע קוד ממנו מתבצעת קריאה לפונקציה) ל callee (רוטינה = קטע קוד המהווה את גוף הביצוע של הפונקציה).

b. תמיכה בשליחת ערך החזרה דרך המחסנית מהרוטינה (מה callee) חזרה ל caller.

2. שלבי שליחת ארגומנטים דרך המחסנית מה caller ל callee:

a. בקטע קוד ממנו מתבצעת קריאה לפונקציה נטען את כל הארגומנטים של הפונקציה למחסנית בעזרת פקודת PUSH.W לפי סדר מוגדר מראש ולאחר מכן נקרא לגוף הפונקציה (לרוטינה) בעזרת פקודת CALL.

b. בתחילת קטע קוד של גוף הפונקציה (רוטינה) נבצע שליפה של הארגומנטים לתוך רגיסטרים בעזרת פקודת POP.W בצורת LIFO ביחס לשלב a.

שימו לב:

- פקודת POP.W הראשונה שולפת מהמחסנית לרגיסטר את הכתובת לחזרה של ה caller.
- מספר פקודות PUSH.W + פקודת CALL בשלב a = מספר פקודות POP.W בשלב b.

משמעות השוויון, לאחר שליפת הארגומנטים מהמחסנית חובה שערך רגיסטר SP יהיה שווה לערכו בתחילת ה caller.

c. מימוש גוף הפונקציה נעשה בעזרת הרגיסטרים לתוכם שלפנו את הארגומנטים מהמחסנית בשלב b.

3. שלבי שליחת ערך החזרה דרך המחסנית מה callee חזרה ל caller:

a. בסיום קטע קוד גוף הפונקציה (רוטינה) במידה והפונקציה מחזירה ערך, נטען את ערך ההחזרה למחסנית ולסיום לפני ביצוע פקודת ret נטען למחסנית את הכתובת לחזרה של ה caller (השמור ברגיסטר).

b. בהגעה חזרה ל caller (עקב ביצוע פקודת ret) נשלוף מהמחסנית את ערך ההחזרה בעזרת פקודת POP.W ובזה הפונקציה הגיעה לסיומה (כעת ערך רגיסטר SP שווה לערך לפני הקריאה לפונקציה).

4. דוגמא מסכמת:

כתיבת קוד למיון מערך בסדר עולה והחזרת הערך ה min של אברי מערך. מיון המערך מבוסס אלגוריתם Selection Sort .

להלן קישור לקוד בשפת אסמבלי וקוד שקול בשפת Python מבוסס פונקציה הכוללת ארגומנטים וערכי החזרה.

E. שימוש בסימולטור:

- לבדיקת התוכנית יש להריצה בסימולטור.
- רשום את גודל התוכנית (לפי כתובת ראשונה ואחרונה של התוכנית בשימוש Disassembly)
- רשום את זמן הריצה שלה (ראה משתנה CYCLECOUNTER בחלון הרגיסטרים, המונה את מספר מחזורי השעון כמתואר ב- Tutorial 1.2). ערך תדר ברירת המחדל של שעון MCLK הוא:

$$f_{MCLK} = 32 \cdot 32768 = 2^{20} = 1,048,576\ Hz \rightarrow T_{MCLK} = \frac{1}{2^{20}} \approx 0.954\ \mu sec$$

F. צורת הגשה דוח מכין:

- הגשת מטלת דוח מכין תיעשה ע"י העלאה למודל של תיקיית zip מהצורה **id1_id2.zip** (כאשר id1 < id2), **רק הסטודנט עם הת"ז id1 מעלה את הקבצים למודל.**
- התיקייה תכיל את שני הפרטים הבאים בלבד:
  - ✓ **קובץ pre_lab$_x$.pdf** – מכיל תשובות לחלק תיאורטי דו"ח מכין
  - ✓ תיקייה בשם **IAR** - מכילה את קובצי המקור (קבצים עם סיומת **.s43***) של מטלה מעשית דוח מכין.
  - ✓ תיקייה בשם **Python** - מכילה את קובצי המקור (קבצים עם סיומת **.py***) של מטלה מעשית דוח מכין.

G. צורת הגשה דוח מסכם:

- הגשת מטלת דוח מכין תיעשה ע"י העלאה למודל של תיקיית zip מהצורה **id1_id2.zip** (כאשר id1 < id2), **רק הסטודנט עם הת"ז id1 מעלה את הקבצים למודל.**
- התיקייה תכיל את שני הפרטים הבאים בלבד:
  - ✓ **קובץ final_lab$_x$.pdf** – מכיל תיאור והסבר לדרך הפתרון של מטלת זמן אמת.
  - ✓ תיקייה בשם **IAR** - מכילה את קובצי המקור (קבצים עם סיומת **.s43***) של מטלה מעשית דוח מכין.
  - ✓ תיקייה בשם **Python** - מכילה את קובצי המקור (קבצים עם סיומת **.py***) של מטלה מעשית דוח מכין.

בהצלחה.