<u>הסבר עבור קוד האסמבלי:</u>

ב – Main הגדרנו את ה – SP צמוד לתחילת הפלאש (בפועל – בדחיפה הראשונה, ה – SP יצביע 2 כתובות כלפי מטה, וכך גם עבור כל דחיפה, בהתאם למספר האיברים הנמצאים במחסנית). דחפנו מצביעים למערכי המקור, ובנוסף, את גודל המערכים. ביצענו קריאה לפונקציה.

בפונקציה: ביצענו MOV, בהתאם לסדר הדחיפה למחסנית, של גודל המערכים לרגיסטר 6 (רגיסטר זה ישמש כמונה איטרציות), של מצביע למערך 2 לרגיסטר 7, ושל מצביע למערך 1 לרגיסטר 8. ניקינו (איפסנו) את רגיסטר 9 כדי שישמש כמספר הכתובות שיש להסיט כאשר מבצעים אחסון למערך התוצאה.

בתוך הלולאה שבפונקציה: שלחנו את המספרים מהמצביעים לרגיסטרים – ממערך 2 לרגיסטר 10 וממערך 1 לרגיסטר 11. לאחר מכן, קידמנו את המצביעים כך שיצביעו למספרים הבאים במערכים. ביצענו RLA פעמיים למספר שהגיע ממערך 2. ביצענו את פעולת ה – XOR, ואחסנו את הערך שהתקבל במקום המתאים במערך התוצאה על ידי שימוש ברגיסטר 9 כקובע ההיסט הנדרש. קידמנו את רגיסטר ההיסט כך שבלולאה הבאה יצביע למקום השמור הבא. החסרנו 1 ממונה האיטרציות (רגיסטר 6), וביצענו את הלולאה פעם נוספת כל עוד מונה האיטרציות לא התאפס.

כאשר מונה האיטרציות התאפס ויצאנו מהפונקציה, הגדלנו את ה – SP בהתאם למס' האיברים שנשארו במחסנית לצורך ריקונה.

הסבר לקוד פייתון

כדי לפתור את המטלה הגדרנו שני מערכי מקור המכילים את המערכים הנתונים ומשתנה שמכיל את גודלם:

```
#==================== Source objects ================
arr1 = [21,80,3,49,18,81,77,13]
arr2 = [31,13,6,65,25,10,52,40]
SIZE = len(arr1)
```

הגדרנו שני מערכים ריקים שיכילו את הספרות בצורתם הבינארית ואת המערך הסופי שיכיל את התנאי המבוקש בשאלה. הגדרנו בנוסף שני מערכים להצגת הפלט בספרות בינאריות והקסדצימליות.

```
#=================== Target objects =================
arr_mix = []
bin_mix_arr = []
hex_mix_arr = []
# Initialize the arrays in binary
binary_arr1 = []
binary_arr2 = []
```

המרנו את המערכים לצורה בינארית וחתכנו את ההתחלה שמכילה אות באנגלית 0ו, בנוסף ביצענו הזזה כמבוקש במטלה לarr2:

```
#================= Algorithm code ==================
for i in range(SIZE):
    binary_arr1.append(bin(arr1[i])[2:])
    binary_arr2.append((bin(arr2[i]))[2:] + "00")
```

הוספנו אפסים למחרוזות שיצרנו כדי שתהיה באורך 16 תווים :

```
# Extend every binary number to contain 16 bits
for i in range(SIZE):
  while len(binary_arr1[i]) < 16:
    binary_arr1[i] = '0' + binary_arr1[i]

  while len(binary_arr2[i]) < 16:
    binary_arr2[i] = '0' + binary_arr2[i]
```

עשינו (xor) כמבוקש בשאלה, ומילאנו את המערכים להצגה בבינארי והקסדצימלי:

```
for i in range(SIZE):

    arr_mix.append(int(binary_arr1[i], 2) ^ int(binary_arr2[i], 2))
    bin_mix_arr.append(bin(arr_mix[i]))
    hex_mix_arr.append(hex(arr_mix[i]))
```

נדפיס את התוצאות בפומט המתאים:

```
#=====================================================
#                  Output Printing
#=====================================================
# Output integer format Printing
print("Integer format Printing")
print(arr_mix)

# Output Bin format Printing
print("\nBin format Printing")
print("Num is:",(bin_mix_arr))

# Output Hex format Printing
print("\nHex format Printing")
print("Num is:",(hex_mix_arr))
```