שאלה 1 – מעקב אחר פקודות:

לפניכם קטע קוד. נתון כי **הכתובת של תחילת מקטע הנתונים היא 0xDEADBEEF**. עליכם לעקוב אחר הפקודות ולרשום תוכן של נתון מבוקש במקומות שמבקשים מכם (בערכי **הקסדצימלי**).

במידה ומתבצעת פקודה לא חוקית בשלב מסוים, יש לרשום X במקום שצריך להשלים, ולהתייחס כאילו הפקודה מעולם לא נרשמה:

```
.global _start

.data
arr: .short 1, 0xEA, 0x2, 0x3, 0b1010
b: .long 0x19283746
c: .quad 0x0404202102052021

.bss
.lcomm jk, 8
.lcomm g_byte, 4

.text
_start:
xor %rcx, %rcx
movl $0x2345, %ebx
movb $0, %bl
```

ערך rbx: **0x0000000000002300**.

```
xor %rax, %rax
xor %rsi, %rsi
add b, %rax, %rbx
```

ערך rbx: **X**.

```
lea 2(arr), %rbx
```

ערך rbx: **X**.

```
lea (arr), %rbx
movb 3(%rbx), %al
```

ערך rax: **0x0000000000000000**.

```
mov %bh, %al
xor %al, %sil
shr $5, %rsi
movw -8(%rbx, %rsi, 2), %dx
```

ערך dx: **0x00EA**.

```
shl $1, %rsi
movb $0x68, g_byte
```

```
addb (%rbx, %rsi, 2), g_byte
```

ערך הבית שב- g_byte: **X**.

```
shr $6, %rax
inc %ax
```

ערך rax: **0x0000000000000003**.

```
mov $jk, %rcx
lea c, %rbx
movw arr+3, %ax
ror $2, %ax
```

ערך rax: **0x0000000000000080**.

```
xor %ax, %ax
incb %ax
```

ערך rax: **X.**

```
movq (%rbx), %rbx
mov $0x40, %si
dec %rcx
movl %ebx, 2(%rcx)
```

התוכן שבבית בכתובת 3+jk: **0x05**.

```
movb $78, b
```

ערך הבית b (הבית שb מהווה פניה אליו): **0x4e**.

```
movq $arr, b
```

ערך הבית b (הבית שb מהווה פניה אליו): **0xef**.

```
movswq (b), %rdx
```

ערך rdx: **0xFFFFFFFFFFFFBEEF**.

```
mov $0x8529, %ax
cwd
```

ערך rdx: **0xFFFFFFFFFFFFFFFF**.

```
movw $-0x33, jk
idivw jk
```

ערך eax: **0x00000268**. ערך edx: **0xFFFFFFE1**.

```
movq $0x100, (b)
imul $2, b, %rdx
```

ערך rax: **0x0000000000000268** . ערך rdx: **0x0000000000000200**

```
xor %rax, %rax
mov $0xff, %ax
mov $4, %bl
mov $30, %rdx
imulb %bl
```

ערך al: **0xFC.** ערך dl: **0x1E .**

```
mov $256, %ax
mov $20, %bx
mul %bx
```

ערך ax: **0x1400.** ערך dx: **0x0000.**

שאלה 2 – תרגום מC לאסמבלי:

לפניכם קטעי קוד בשפת c עליכם לתרגם כל קטע בשפת c לאסמבלי על ידי השלמת המקומות שמסומנים בקו. במידה וכל השורה מסומנת בקו עליכם להשלים את השורה איך שאתם רוצים אך עליכם להשתמש בפוקדה אחת בלבד! נתון ש-a ו-b הוגדרו כ int.

מומלץ לעבור על "אופטימיזציה אריתמטית" מתרגול 2, ולראות דוגמאות לפני המעבר על השאלה.

<u>הערה 1:</u> בשורה הרביעית הרווח אחרי lea(אינו טעות. אין להשלים שם ערך. זהו רמז (וחלק מהסינטקס).

<u>הערה 2:</u> נזכיר כי '|' בשפת C היא הפעולה or.

על מנת למנוע בלבול מסופקת לכם דוגמה בשורה הראשונה:

קוד בשפת **c**	קוד אסמבלי
`a += b;`	`movl ___b__, %eax` `addl ____%eax__, __a____`
`a = a / 8;`	`sarl $3, a`

`a = 9*a;`	`movl a, %eax`
	`lea (%eax, %eax, 8), %eax`
	`mov %eax, a`

`b = b*8;`	`movl b, %ebx`
	`lea ( , %ebx , 8), %ebx`
	`mov %ebx, b`

`a = b*2 - 11 + a;`	`movl a, %eax`
	`movl b, %ebx`
	`lea -11(%eax, %ebx, 2), %eax`
	`mov %eax, a`

`a++`	`incl a`
`a = 5*a;`	`imul $5 , a , %eax` `mov %eax, a`

`a = a*a*a*a;`	`movl a, %eax`
	`imul %eax, %eax`
	`imul %eax, %eax`
	`mov %eax, a`

`if (a >= 0)`
`  b = 0;`
`else`
`  b = -1;`
	`movl a, %eax`
	`cdq`
	`movl %edx, b`

שאלה 3 – לולאות ומספרים:

בשאלה זו נשתמש במספרים חסרי סימן (unsigned).
בנוסף, נניח כי הוגדר משתנה n>0 שגודלו 4 בייטים ושכל ה-General Purpose Registers מכילים 0 בתחילת התוכנית (הכוונה היא לרגיסטרים שמשתמשים בהם לחישובים ולא לריגסטרים מיוחדים כמו rip או rflags)
שירה המנהלת הבכירה בסופר כתבה קטע קוד. לפניכם הקוד ששירה כתבה:

```
_start:
  xor %ax, %ax
  mov $1, %bx
  mov (n), %cx

.L1:
  mov %bx, %r9w
  imul %bx, %r9w
  add %r9w, %ax
  inc %bx
  dec %cx
  test %cx, %cx
  jne .L1
END:
```

1. נתון שבתחילת התוכנית $n = 10$ (בעשרוני).
מה יהיה ערך רגיסטר ax בסיום קטע התוכנית (בעת ההגעה לתווית END)? כתבו את התשובה גם בבסיס דצימלי וגם בהקסדצימלי (וכתבו את כל הבתים שלו ב-hexa)?

דצימלי: $\sum_{n=1}^{10} n^2 = 385$

הקסדצימלי : 0x0181

2. הסבירו במשפט מה עושה קטע הקוד

התכנית סוכמת את ריבועי המספרים מ1 ועד n (במקרה שצויין, n=10 ולכן הסכימה היא 1+4+9+16+25+….+100). הקוד שומר את התוצאה ברגיסטר %ax.

3. רמזי הסטודנט החרוץ הבחין שעבור $n = 70$ מוחזרת תשובה לא נכונה. מה הסיבה לכך?
מהו המספר הגדול ביותר שניתן לשים ב-n בתחילת הריצה, ועדיין לקבל תשובה נכונה?
רמז: desmos

כשסוכמים את ריבועי המספרים מ- 1 ועד 58 (ומעלה), הסכום הוא 66,729 (בהקסה: 0x104A9), והתוצאה גולשת מ-2 בתים. לכן, ברגיסטר %ax נקבל 0x04A9 – שזוהי תוצאה שגויה. לכן, נסיק כי ה- n הגדול ביותר הוא 57 (ונקבל 0xF875 = %ax).

<u>השאלה ממשיכה בעמוד הבא</u>

4. רמזי, שרוצה לקבל את פרס העובד המצטיין , שינה את הקוד:

```
_start:
  xor %ax, %ax
  mov $1, %bx
  mov (n), %cx

.L1:
  mov %bx, %r9w
  imul %bx, %r9w
  add %r9d, %eax
  inc %bx
  dec %cx
  test %cx, %cx
  jne .L1
END:
```

אך הקוד עדיין מחזיר תשובות שגויות עבור מספרים גדולים מהמספר שמצאתם בסעיף הקודם. מדוע?

במידה והקוד של רמזי עדיין צריך לקיים את אותה הפונקציונליות של הקוד של שירה (בו התוצאה נשמרת ברגיסטר %ax), אז כשנפנה לערך ב- %ax יתבצע slicing למספר, ועל אף ש- %eax מחזיק את הערך הנכון - 0x104A9, ה- slicing שיתבצע יוביל להחזרת התוצאה השגויה 0x04A9.

במידה והכוונה היא לשנות את הפונקציונליות של הקוד של שירה, ולהחזיר את התשובה ברגיסטר %eax במקום %ax, אז באופן דומה לסעיף הקודם, עבור n גדול מידי נקבל שהתשובה תחרוג מטווח הייצוג של %eax (ארבעה בתים), ושוב נקבל תוצאות שגויות.

5. השלימו את השורות הבאות, כך שיתקבל קוד **<u>חסר לולאות</u>** שיחזיר את אותה תוצאה ב-ax עבור כל n. ניתן להשאיר שורות ריקות.

```
_start:
  mov (n), %ax

  mov %ax, %bx

  add $1 , %bx

  mov %ax, %cx

  imul $2, %cx

  add $1, %cx

  mul %bx
```

```
    mul %cx
    mov $6, %r9
    divw %r9w
END:
```