שאלה 1 – מעקב אחר פקודות:

לפניכם קטע קוד. נתון כי הכתובת של תחילת **data section** היא **0xDEADBEEF**. עליכם לעקוב אחר הפקודות ולרשום תוכן של נתון מבוקש במקומות שמבקשים מכם (בערכי הקסדצימלי).
אם הפקודה לא חוקית בשלב מסוים, יש לרשום **X** במקום שצריך להשלים, ולהתייחס כאילו הפקודה מעולם לא נרשמה.
בנוסף, נמקו מה הבעיה בפקודה.

```
.global _start

.section .data
arr: .short 6, 0xEA, 0x22, 0x4B1D, 0b1010
buff: .fill 10, 2, 0x42
id: .long 0x19283746
key: .quad 0x0406282309052021

.section .bss
.lcomm a, 8
.lcomm b, 4

.section .text
_start:
  xor %rcx, %rcx
  movl $0x5432, %ebx
  movb $4, %bl
```

ערך rbx: 0x5404

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

ערך rbx: X יותר מידי אופרנדים

```
  lea 4(arr), %rbx
```

ערך rbx: X שיטת מיעון עקיפה מכילה רגיסטר וקבוע ולא label/כתובת אבסולוטית.

```
  lea (buff), %rbx
  movb 4(%rbx), %al
```

ערך rax: 0x42

```
  movb 7(%rbx), %al
```

ערך rax: 0x0

```
  lea (arr), %rbx
  mov %bh, %al
  xor %al, %sil
  shr $5, %rsi
```

ערך rsi: 0x5

```
  movw -4(%rbx, %rsi, 2), %dx
```

ערך **dx**: 0x4B1D

```
  shl $1, %rsi
  movb $0x68, b
  addb (%rbx, %rsi, 2), b
```

ערך הבית b (הבית שb מהווה פניה אליו): X פקודת זיכרון זיכרון

השאלה ממשיכה בעמוד הבא

```
mov $0xFFFF00, %rax
shr $8, %rax
inc %ax
```

ערך rax: 0x0

```
movw arr+3, %ax
ror $2, %ax
```

ערך rax: 0x880

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

ערך rax: X סופית לא מתאימה

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

ערך הבית 4+a (הבית ש- 4+a מהווה פניה אליו): 0x09

```
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: 0XFFFF FFFF FFFF BEEF

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

ערך rdx: 0XFFFF FFFF FFFF FFFF

טעות נפוצה: שימו לב שמינוס 1 זה לא פתרון נכון! (ביקשנו תשובה בהקסא. אף אחד לא הבטיח חשיבות לסימן).

```
movw $-0x9F, a
idivw a
```

ערך **eax**: 0x89

ערך **edx**: 0xFFFF FFC1

idiv משפיע רק על 16 הביטים התחתונים, אבל השאר לא מתאפסים.

```
movq $0x123, (b)
imul $3, b, %rdx
```

ערך rax: 0x89

ערך rdx: 0x369

```
xor %rax, %rax
mov $0xfc, %ax
mov $4, %bl
mov $015, %rdx
imulb %bl
```

ערך **al**: 0xf0

ערך **dl**: 0xd

```
leaq $0x40FE67, %rdx
```

ערך rdx: X אין לקבוע כתובת אפקטיבית

שאלה 2 – תרגום מC לאסמבלי:

לפניכם קטעי קוד בשפת c עליכם לתרגם כל קטע בשפת c לאסמבלי על ידי השלמת המקומות שמסומנים בקו. אם כל השורה מסומנת בקו עליכם להשלים את השורה בכל דרך שתרצו, אך עם פקודה אחת בלבד!

נתון ש-a ו-b הוגדרו כ int. מותר לכם להשתמש בכל רגיסטר עזר שתרצו.

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

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

הערה 2: נזכיר כי '~' בשפת C היא הפעולה not.

על מנת למנוע בלבול מסופקת לכם דוגמה בשורה הראשונה:

קוד בשפת **c**	קוד אסמבלי
`a += b;`	`movl b, %eax` `addl %eax, a`
`a = a / 16;`	`sarl $4, a`

`a = 3*a;`	`movl a, %eax`
	`lea (%eax, %eax, 2), %eax`
	`mov %eax, a`

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

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

`a = b*2 - 24 + a;`	`movl a, %eax`
	`movl b, %ebx`
	`lea -24(%eax,%ebx,2), %eax`
	`mov %eax, a`

`a--`	`decl a` (ה-suffix של ה-l הכרחי!)

`a = ~(1<<16)`	`mov $0x10000 %eax`
	`not %eax`
	`mov %eax, a`
	(אפשר גם מראש להעביר 0xFFFEFFFF ובפקודה השניה לשים nop)

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

שאלה 3 – לולאות ומספרים:

בשאלה זו נשתמש במספרים חסרי סימן (unsigned).

בנוסף, נניח כי הוגדר משתנה $n > 0$ שגודלו 16 ביט ושכל ה-General Purpose Registers מכילים 0 בתחילת התוכנית (הכוונה היא לרגיסטרים שמשתמשים בהם לחישובים ולא לרגיסטרים מיוחדים כמו rip או rflags)

קורנליוס האיום כתב את קטע קוד הבא:

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

.L1:
  mov %bx, %r9w
  imul %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 (כתבו את כל הבתים שלו ב-hexa)?
0xBD1 = 3025

2. איזו נוסחה/ביטוי מתמטי מחשב קטע הקוד הנ"ל?

$$\sum_{i=0}^{n} i^3 = \left(\frac{n(n+1)}{2}\right)^2$$

3. יהודית שבאה לבקר את קורנליוס שמה לב שעבור $n = 55$ מוחזרת תשובה לא נכונה. מה הסיבה לכך? מהו המספר הגדול ביותר שניתן לשים ב-n בתחילת הריצה, ועדיין לקבל תשובה נכונה?

הסיבה היא חריגת זיכרון ברגיסטרים. הרגיסטר ax יכול לשמור מספר בגודל של 2^{16} לכל היותר, אבל באיטרציה ה-23 הסכום החלקי שמצטבר גדול מכך. הרגיסטר שבו נשתמש אינו רחב מספיק כדי להכיל מספר זה ולכן לא נוכל להשתמש בו בשביל להחזיק את הסכום. לכן $n = 22$ היא התשובה.

השאלה ממשיכה בעמוד הבא

4. סיוון, האויבת של יהודית, רצתה להראות שהיא הכי טובה. לכן הציגה את הקוד שלה לפתרון הנוסחה:

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

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

ענו על סעיף 3 שוב, הפעם בהתייחס לקוד של סיוון.

שוב הסיבה היא חריגה של רגיסטרים (overflow). הפעם אמנם משתמשים ב-eax ולכן לא שם הבעיה, אבל בשלב כלשהו n^3 יהיה יותר מדי גדול, כך שיהיה overflow ברגיסטר הזמני ((r9w.
עבור $n < 41$ עדיין נקבל תשובה נכונה, אבל עבור $n \geq 41$ יהיו חריגות ב-r9w. לכן $n = 40$ היא התשובה.
(בפועל גם ב-eax יהיו בעיות החל מ-n מסוים, כי תוצאת הנוסחה יכולה לחרוג גם מ-32 ביט, אבל השגיאות בסעיף זה יתחילו ב-n קטן יותר ולכן להתייחס כאן ל-eax זו טעות)

5. השלימו את השורות הבאות, כך שיתקבל קוד חסר לולאות שיחזיר את ב-**rax** את התוצאה של הנוסחה מסעיף 2 בצורה נכונה לכל n חסר סימן בגודל 16 ביט.

```
_start:
  movzwq  n(%rip), %rdi
  leaq    1(%rdi), %rax
  imulq   %rdi, %rax
  shrq    %rax
  imulq   %rax, %rax
END:
```

(יש כמובן עוד פתרונות)

(צריך להשתמש ב- movzwq כי n הוא רק 16 ביטים סתם mov לרגיסטר יביא עוד מידע שאנחנו לא יודעים מהו מהזיכרון)

(צריך לעשות shift רק של ביט אחד לפני שמעלים בריבוע, או לחלופים לעשות shift של 2 ביטים לאחר שמעלים)