ארגון ותכנות המחשב

תרגיל 1 - חלק יבש

המתרגל האחראי על התרגיל: תומר כץ.

שאלות על התרגיל – ב- **Piazza** בלבד.

הוראות הגשה:

- ההגשה בזוגות.
- על כל יום איחור או חלק ממנו, שאינו באישור מראש, יורדו 5 נקודות.
  - ניתן לאחר ב-3 ימים לכל היותר.
- הגשות באיחור יתבצעו דרך אתר הקורס.
- לכל שאלה יש לרשום את התשובה במקום המיועד לכך.
- יש לענות על גבי טופס התרגיל ולהגיש אותו באתר הקורס כקובץ **PDF**.
  - ניתן להקליד את התשובות במסמך ה-WORD, או לכתוב אותן על גבי גרסת ה-PDF בעזרת הטאבלט החביב עליכן. העיקר להגיש בסופו של דבר קובץ PDF לבדיקה, בכתב ברור וקריא.

שאלה 1 – מעקב אחר פקודות:

לפניכם קטע קוד. נתון כי הכתובת של תחילת **data section** היא **xDEADBEEF0**. 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: 0x0000000000005404

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

ערך rbx: x לא קיימת פקודה ADD עם שלושה פרמטרים

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

ערך rbx: 0xDEADBEF3

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

ערך rax: 0x0000000000000042

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

ערך rax: 0x0000000000000000

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

ערך rsi: 0x0000000000000005

```
  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): 0xaa

השאלה ממשיכה בעמוד הבא

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

ערך rax: 0x0000000000000000

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

ערך rax: 0x0000000000000880

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

ערך rax: X
קלט לא תקין של 16 ביט במקום 8

```
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: 0xFFFFFFFFFFFFBEEF

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

ערך rdx: 0xFFFFFFFFFFFFFFFF

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

ערך **eax**: 0x00000089

ערך **edx**: 0x0000ffc1=-0x3f

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

ערך rax: 0x0000000000000089

ערך rdx: 0x0000000000000369

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

ערך **al**: 0xf0

ערך **dl**: 0x0d

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

ערך rdx: X
לקבוע אין 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`

`a = ~(1<<16)`	`movl $0x00010000, %eax`
	`xor $0xFFFFFFFF, %eax`
	`mov %eax, a`

~~a=a*a*a*a;~~
a=a*a*a*a
	`movl a, %eax`
	`imul %eax, %eax`
	`imul %eax, %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)?

3025 – דצימלי

0x0bd1 – Hex

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

$$\sum_{i=1}^{n} i^3 = \frac{i^2 (i+1)^2}{4}$$

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?

בפקודה של הכפל, כשאנחנו מחשבים את המספר בחזקת 3, אנו עושים את זה לתוך 16 ביטים, המספר הגדול ביותר שאפשר להחזיק במקרה הזה הוא 2 בחזקת 16 פחות 1, 55 בחזקת 3 יוצא מספר גדול מכך ולכן נחשב רק חלק ממנו בסכימה. לכן המספר הגדול ביותר הוא 40. עם זאת נשים לב כי רגיסטר אי אייקס גם מוגבל לאותו מספר, לכן גם הסכום אמור להיות מוגבל, כפי שנראה בחישוב, נקבל כי המספר המקסימלי הפעם הוא $\boxed{22}$.

ex

$$i^3 \le 2^{16} - 1$$

$$i \le \sqrt[3]{2^{16} - 1} = 40.31$$

$$i \le 40$$

$$\frac{i^2 (i+1)^2}{4} \le 2^{16}$$

<u>השאלה ממשיכה בעמוד הבא</u>

$$\frac{i(i+1)}{2} \le 2^8$$

$$i(i+1) \le 2^9 \Rightarrow \boxed{i \le 22}$$

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 %r9w, %eax  →  אאאב?  פקודה לא חוקית?
  inc %bx
  dec %cx
  test %cx, %cx
  jne .L1
END:
```

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

כעת רגיסטר אי איי איקס יכול להכיל 2 בחזקת 32, לכן המגבלה על הסכום נהיית חלשה, מאותו חישוב נמצא שוב פעם את ה n המקסימלי ונקבל: eax

כי לגביי הסכום אפשר עד 361. עם זאת הייתה לנו את המגבלה של 40, ולכן $\boxed{40}$ הוא המספר המקסימלי

$\frac{i^2(i+1)^2}{4} \le 2^{32}$

$\frac{i(i+1)}{2} \le 2^{16}$

$i(i+1) \le 2^{17} \Rightarrow \boxed{i \le 361}$

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. כמובן הניחו כי n מוגדר לכם מראש ב-section אחר ואין צורך להגדירו. ניתן להוסיף שורות, אך קוד עם יותר מ-5 פקודות יקבל ניקוד חלקי בלבד.

```
_start:
```

1) mov n, %ax
2) imul %rax, %rax
3) incw n
4) imul n, %rax
5) imul n, %rax
6) shr $2, %rax

END: