

## 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 **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, במקומו 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
```

0x00000000000005404

ערך rbx:

```
    xor %rax, %rax
```

```
    xor %rsi, %rsi
```

```
    add b, %rax, %rbx
```

X  
לא קיימת פקודה  
ADD  
עם שלושה פרמטרים

ערך rbx:

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

0xDEADBEF3

ערך rbx:

```
    lea (buff), %rbx
```

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

0x0000000000000042

ערך rax:

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

0x0000000000000000

ערך rax:

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

```
    mov %bh, %al
```

```
    xor %al, %sil
```

```
    shr $5, %rsi
```

0x0000000000000005

ערך rsi:

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

0x4b1d

ערך dx:

```
    shl $1, %rsi
```

```
    movb $0x68, b
```

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

0xaa

ערך הבית ב (הבית שש מהווה פניה אליו):

השאלה ממשיכה בעמוד הבא

```

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

```

ערך rax: 0x0000000000000000

```

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

```

ערך rax: 0x00000000000000880

```

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 ב (הבית ש- 8 מהווה פניה אליו): 0x4e

```

movq $arr, b

```

ערך הבית ב (הבית ש- 8 מהווה פניה אליו): 0xEF

```

movsq (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: 0x00000000000000089

ערך rdx: 0x00000000000000369

```

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 מ C 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 שתמצאו, אך עם פקודה אחת בלבד! נתון ש-a ו-b הוגדרו כ int. מותר לכם להשתמש בכל רגיסטר עזר שתמצאו. 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;	<pre>movl <u>b</u>, %eax addl <u>%eax</u>, <u>a</u></pre>
a = a / 16;	<pre>sarl <u>\$4</u>, <u>a</u></pre>
a = 3*a;	<pre>movl a, %eax lea (<u>%eax</u>, <u>%eax</u>, <u>2</u>), <u>%eax</u> mov %eax, a</pre>
b = b*8;	<pre>movl b, %ebx lea ( , <u>%ebx</u>, <u>8</u> ), %ebx mov %ebx, b</pre>
<pre>if (a &gt;= 0)     b = 0; else     b = -1;</pre>	<pre>movl a, %eax <u>CDQ</u> <hr/> movl %edx, b</pre>
a = b*2 - 24 + a;	<pre>movl a, %eax movl b, %ebx <u>lea -24(%eax, %ebx, 2), %eax</u> <hr/> mov %eax, a</pre>
a--	<pre><u>decl a</u> <hr/></pre>
a = ~(1<<16)	<pre><u>movl \$0x00010000, %eax</u> <hr/> <u>xor \$0xFFFFFFFF, %eax</u> <hr/> mov %eax, a</pre>
<pre><del>xxxxxxxxxxxxxxxx</del> a=a*a*a</pre>	<pre>movl a, %eax <u>imul %eax, %eax</u> <hr/> <u>imul %eax, %eax</u> <hr/> mov %eax, a</pre>

## 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:
```

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

3025 - 0x0bd1  
Hex

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

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

- 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 איי אקס גם מוגבל לאותו מספר, לכן גם הסכום אמור להיות מוגבל, כפי שנראה בחישוב, נקבל כי המספר המקסימלי הפעם הוא 22.

$$\begin{aligned} i^3 &\leq 2^{16} - 1 \\ i &\leq \sqrt[3]{2^{16} - 1} = 40.31 \\ i &\leq 40 \end{aligned} \quad \left| \quad \begin{aligned} \frac{i^2(i+1)^2}{4} &\leq 2^{16} \\ \frac{i(i+1)}{2} &\leq 2^8 \\ i(i+1) &\leq 2^9 \Rightarrow i \leq 22 \end{aligned} \right.$$

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.

*eax*

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

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

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

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

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

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, %rax

$$rax = n$$

2) inc %rax

$$rax = n + 1$$

3) imul n, %rax

$$rax = (n+1)n$$

4) imul %rax, %rax

$$rax = n^2(n+1)^2$$

5) shr \$2, %rax

$$rax = \frac{n^2(n+1)^2}{4}$$

END: