

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.
- תיקונים בקובץ ממורקרים.

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 הקסדימלי).
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

.lcomm c, 10

.section .text

_start:

xor %rcx, %rcx

movl \$0x5432, %ebx

movb \$4, %bl

ערך rbx: 0x00000000000005404

xor %rax, %rax

xor %rsi, %rsi

add b, %rax, %rbx

ערך rbx: X – פעולות add הן מקבלות 2 ארגומנטים ולא 3 ולכן rbx לא משתנה

0x00000000000005404

lea 4(arr), %rbx

ערך rbx: X פעולה זו צריכה להכיל רגיסטר וקבוע ולא label או כתובת אבסולוטית

0x00000000000005404

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 ב (הבית שש מהווה פניה אליו): X אי אפשר את הפקודה עם 2 גישות לזיכרון

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

ערך rax: 0x0000000000000000

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

ערך rax: 0x0000000000000080

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

ערך rax: X incb מוסיף 1 למילה בגודל 8 אבל ax בגודל 16 ולכן לא הגיוני ועל כן ערך

Rax הוא: 0x0000000000000080

```
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): 0x9

```
movb $78, b
```

ערך הבית ב (הבית שש מהווה פניה אליו): 0x4E

```
movq $arr, b
```

ערך הבית ב (הבית שש 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 (137)

ערך edx: 0xFFFFFFFFC1 (-63)

```
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 (-16)

ערך dl: 0x0D

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

ערך rdx: X לא עובד משום שבlea אנו מצפים לארגומנט ראשון שהוא זכרון ולא כתובת אבסולוטית

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! בתאים עם כמה שורות קוד חייבים למלא את כולן.

נתון ש-a ו-b הוגדרו כ int וכל הרגיסטרים מאותחלים ל-0. 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;	movl b, %eax addl %eax, a
a = a / 16;	sarl \$4, a
a = 3*a;	movl a, %eax leal (%eax, %eax, 2), %eax mov %eax, a
b = b*8;	movl b, %ebx leal (, %ebx, 8), %ebx mov %ebx, b
if (a >= 0) b = 0; else b = -1;	movl a, %eax imul \$1 movl %edx, b
a = b*2 - 24 + a;	movl a, %eax movl b, %ebx leal -24(%eax,%ebx,2) %eax mov %eax, a
a--	dec a
a = ~(1<<16)	mov %eax, \$-1 not %eax mov %eax, a
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)?

נשים לב כי a הוא הסכום של i בחזקת 3 לאורך n לכן התשובה
 $ax = (3025)_{10} = 0x0BD1$

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

$$ax = \sum_i^n i^3$$

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?

נשים לב כי ax בגודל 4 בתים לכן הערך הכי גדול שנוכל לשמור בחלק זה הוא יהיה בהקסימלי 0xFFFF שהוא בערך דצימלי הוא 65535. ולכן הח הגדול ביותר אשר נוכל לבחור הוא $n=22$ אשר יתן לנו משום שכאשר נגדיל את n אפילו ב1 נקבל ערך גדול מהרצוי שהוא יהיה $ax=76176$.

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.

כעת משום שאם n באורך של 8 בתים לכן הערך הגדול ביותר שניתן לשמור הוא 4294967295 בדצימלי ולכן כאשר $n=55$ התוצאה תיהי נכונה ונוכל להגדיל את n עד 361.

5. השלימו את השורות הבאות, כך שיתקבל קוד חסר לולאות שיחזיר את 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:

    mov n %rax

    lea 1(%rax), %rbx

    shr %rax, 1

    mul %rbx

    mul %rax
```

END: