טכניון – מכון טכנולוגי לישראל

# ארגון ותכנות המחשב

# תרגיל 1 - חלק יבש

המתרגל האחראי על התרגיל: נדב פליקוביץ.

**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 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: 0x2300

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: 0x0

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: 0x3

mov $jk, %rcx

lea c, %rbx

movw arr+3, %ax

ror $2, %ax

ערך rax: 0x80

xor %ax, %ax

incb %ax

ערך rax: X

movq (%rbx), %rbx

mov $0x40, %si

dec %rcx

movl %ebx, 2(%rcx)

התוכן שבבית בכתובת jk+3: 0x5

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: 0x268 ערך edx: 0xffffffel

movq $0x100, (b)

imul $2, b, %rdx

ערך rax: 0x268 ערך rdx: 0x200

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: 00x140 ערך dx: 0x0

# 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.

**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 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 (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)?

= 385 = 0x181 ax = (10\*11\*21)/6 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. הסבירו במשפט מה עושה קטע הקוד

הקוד עובר על המספרים מ 1 ועד n וסוכם אותם בריבוע ומכניס התוצאה לתוך ax. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. רמזי הסטודנט החרוץ הבחין שעבור מוחזרת תשובה לא נכונה. מה הסיבה לכך?

מהו המספר הגדול ביותר שניתן לשים ב-n בתחילת הריצה, ועדיין לקבל תשובה נכונה?

רמז: desmos

\_\_\_\_\_\_\_ כאשר 0xffff הוא המספר המקסימלי שניתן לייצג ב16 ביטים. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**השאלה ממשיכה בעמוד הבא**

1. 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?

השגיאה עדיין מתקבלת עבור מספרים גדולים מהמספר שמצאנו כי גם לרגסטר eax יש טווח ייצוג וכל מספר שחורג מטווח זה גורם לשגיאה בקוד הנ"ל. בפרט נקבל כי המספר המקסימלי שניתן לייצג ברגסטר eax הינו 0xffffffff והמספר המקסימלי שיעבוד בקוד שנכתב בשאלה הינו 1290 ועבור מספרים שגדולים ממנו נקבל overflow ומספרים אלה הם בפרט גדולים מהמספר שמצאנו בסעיף הקודם. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. השלימו את השורות הבאות, כך שיתקבל קוד **חסר לולאות** שיחזיר את אותה תוצאה ב-ax עבור כל . ניתן להשאיר שורות ריקות.

\_start:

mov (n), %ax

mov $1, %bx\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

mov (n), %cx\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

mov $6, %dx\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

add (n), %bx\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

add (n), %cx\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

add $1, %cx\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

imul %bx, %ax\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

imul %cx, %ax\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

idiv %dx\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

END: