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 #rcx = 0

movl $0x2345, %ebx # ebx = 0x2345

movb $0, %bl # bl = 0

ערך rbx: 0x2300.

xor %rax, %rax #rax = 0

xor %rsi, %rsi # rsi = 0

add b, %rax, %rbx #illegal

ערך rbx: X.

lea 2(arr), %rbx # illegal

ערך rbx: X.

lea (arr), %rbx #rbx = &arr

movb 3(%rbx), %al #al = \*(arr+3) = 0

ערך rax: 0x0.

mov %bh, %al #al = 0x20, %rbx=0xDEADBEEF, second byte from right

xor %al, %sil #sil = %al = 0x20

shr $5, %rsi # sil = 0b100000>>5 = 0b1

movw -8(%rbx, %rsi, 2), %dx # dx = \*(arr + 5\*2 – 8) = \*(arr+2) = 0xEA

ערך dx: 0xEA.

shl $1, %rsi #rsi = 0b1<<1 = 0b2

movb $0x68, g\_byte #g\_byte = 0x68

addb (%rbx, %rsi, 2), g\_byte #g\_byte = g\_byte + \*(arr + 2\*2) = g\_byte + 2 = 0x6a

ערך הבית שב- g\_byte: x

shr $6, %rax # rax = 0xBE>>6=0x2

inc %ax # rax++ 🡪 rax = 3 = 0x3

ערך rax: 0x3.

mov $jk, %rcx # rcx = address of jk

lea c, %rbx # rbx = address of c

movw arr+3, %ax

ror $2, %ax # ax was (0000 0000 0000 0001) 🡪 ax = (0100 0000 0000 0000), carry = 0

ערך rax: 0x80.

xor %ax, %ax # ax = 0

incb %ax # error - %ax is 2 bytes, incb fits 1 byte

ערך rax: 0x0.

movq (%rbx), %rbx # rbx = c

mov $0x40, %si

dec %rcx

movl %ebx, 2(%rcx)

התוכן שבבית בכתובת jk+3: 0x80

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: 0xfffffffffff

movw $-0x33, jk

idivw jk

ערך eax: 0x268. ערך edx: 0xffe1.

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: 0x1400. 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++ | Add $1, a |
| a = 5\*a; | imul $5, (a), %eax  mov %eax, a |
| a = a\*a\*a\*a; | movl a, %eax  mul %eax, %eax  mul %eax, %eax  mov %eax, a |
| if (a >= 0)  b = 0;  else  b = -1; | movl a, %eax  cwd  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 0x183

1. הסבירו במשפט מה עושה קטע הקוד

סכום ריבועים של 1 עד n.

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

הערך המקסימלי שניתן לשמור ברגיסטר התוצאה הוא 216 ועבור n=70 הסכום חוצה את זה ולכן התוצאה לא נכונה. עבור n=57 התוצאה תהיה נכונה כי לא תהיה חריגה ועובר 58 כבר תהיה חרגיה

**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?

כעת יש יותר ביטים בהם ניתן לשמור את התוצאה הנוכחית (64 ) אך עדיין ניתן להגיע למצב שהסכום יוצא גדול יותר מגבולות הרגיסטר, עם זאת רק עבור n גדול יותר מקודם. נגיע לחריגה מאותה סיבה שהגענו אליה בשאלה הקודמת כאשר הסכום גדול מ264

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 %ax, %bx

add $1, %bx

mul %bx, %ax

add %ax, %bx

mul %bx, %ax

div $6, %ax

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

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

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

END: