טכניון – מכון טכנולוגי לישראל

# 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, **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: 0x5404

xor %rax, %rax

xor %rsi, %rsi

add b, %rax, %rbx

ערך rbx: X – הפעולה add מקבלת רק שני ארגומנטים.

lea 4(arr), %rbx

ערך rbx: X – אסור לכתוב 4(arr) כי שיטת המיעון צריכה להיות או קבועה או  
עקיפה ופה זה מעורבב.

lea (buff), %rbx

movb 4(%rbx), %al

ערך rax: 0x42

movb 7(%rbx), %al

ערך rax: 0x0

lea (arr), %rbx

mov %bh, %al

xor %al, %sil

shr $5, %rsi

ערך rsi: 0x5

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): X – לא אפשרי ששני האופרנדים יגשו לזיכרון

mov $0xFFFF00, %rax

shr $8, %rax

inc %ax

ערך rax: 0x0

movw arr+3, %ax

ror $2, %ax

ערך rax: 0x880

xor %ax, %ax

incb %ax

ערך rax: X – ax בגודל שני בתים לכן incb שפועלת על בית אחד לא תעבוד.

mov $a, %rcx

lea key, %rbx

movq (%rbx), %rbx

mov $0x40, %si

dec %rcx

movl %ebx, 2(%rcx)

ערך הבית a+4 (הבית ש- a+4 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:** 0x89

ערך **edx:** 0xFFFFFFC1

movq $0x123, (b)

imul $3, b, %rdx

ערך rax: 0x89

ערך rdx: 0x369

xor %rax, %rax

mov $0xfc, %ax

mov $4, %bl

mov $015, %rdx

imulb %bl

ערך **al:** 0xF0

ערך **dl:** 0xD

leaq $0x40FE67, %rdx

ערך rdx: X – האופרנד הראשון חייב להתייחס למקום בזכרון.

# 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  leal -24(%eax, %ebx, 2), %eax  mov %eax, a |
| a-- | decl a |
| a = ~(1<<16) | movl $0x10000, %eax  notl %eax  mov %eax, a |
| a = a\*a\*a\*a\*a\*a\*a\*a; | movl a, %eax  imull %eax, %eax  imull %eax, %eax  imull %eax, %eax  mov %eax, a |

# שאלה 3 – לולאות ומספרים:

בשאלה זו נשתמש במספרים חסרי סימן (unsigned).

בנוסף, נניח כי הוגדר משתנה 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 //ax = 0x0

mov $1, %bx //bx = 0x1

mov (n), %cx //cx = n

.L1:

mov %bx, %r9w //r9w = bx

imul %bx, %r9w // r9w = bx^2

imul %bx, %r9w // r9w = bx^3

add %r9w, %ax //ax += bx^3

inc %bx //bx++

dec %cx //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** (כתבו את כל הבתים שלו ב-hexa)?

ax=3025=0x0BD1

1. איזו נוסחה/ביטוי מתמטי מחשב קטע הקוד הנ"ל?
2. יהודית שבאה לבקר את קורנליוס שמה לב שעבור מוחזרת תשובה לא נכונה. מה הסיבה לכך?

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

נרצה שיתקיים , כי אחרת נגרם overflow והתשובה לא נכונה. נקבל כי התשובה היא כי:

1. 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

inc %bx

dec %cx

test %cx, %cx

jne .L1

END:

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

הקוד לא תקין כי בשורה add %r9w, %eax הרגיסטר eax מגודל 4 בתים והרגיסטר r9w מגודל 2 בתים.

1. השלימו את השורות הבאות, כך שיתקבל קוד **חסר לולאות** שיחזיר את **ב-rax** את התוצאה של הנוסחה מסעיף 2 בצורה נכונה לכל חסר סימן בגודל 16 ביט. כמובן הניחו כי מוגדר לכם מראש ב-section אחר ואין צורך להגדירו. ניתן להוסיף שורות, אך קוד עם יותר מ-5 פקודות יקבל ניקוד חלקי בלבד.

\_start:

movzwq (n), %rcx

leaq 1(%rcx), %rax

mulq %rcx

shrq $1, %rax

mulq %rax

END: