### הטכניון – מכון טכנולוגי לישראל

**ארגון ותכנות המחשב**

תרגיל 1 - חלק יבש

המתרגל האחראי על התרגיל: תום הרמן.

שאלותיכם במייל בעניינים מנהלתיים בלבד, יופנו רק אליו.

**כתבו בתיבת subject: יבש 1 את"ם.**

שאלות בעל-פה ייענו על ידי כל מתרגל.

תאריך הגשה: 02.05.21

הוראות הגשה:

* לכל שאלה יש לרשום את התשובה במקום המיועד לכך.
* יש לענות על גבי טופס התרגיל ולהגיש אותו באתר הקורס כקובץ PDF.
* על כל יום איחור או חלק ממנו, שאינו בתיאום עם המתרגל האחראי על התרגיל, יורדו 5 נקודות.
* הגשות באיחור יש לשלוח למייל של אחראי התרגיל בצירוף פרטים מלאים של המגישים (שם+ת.ז).
* שאלות הנוגעות לתרגיל יש לשאול דרך הפיאצה בלבד.
* ההגשה בזוגות.

# שאלה 1 – מעקב אחר פקודות:

לפניכם קטע קוד. נתון כי הכתובת של תחילת מקטע הנתונים היא 0x492054. 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

.section .data

arr: .short 0, 1, 0x42, 0x67, 0x1234

b: .int 0x19283746

c: .quad 0x0404202102052021

.section .bss

.lcomm garbage, 8

.lcomm g\_byte, 1

.section .text

\_start:

xor %rcx, %rcx

movq $0x1234, %rax

movb $0, %al

ערך rax: \_\_\_\_\_0x1200\_\_\_.

xor %rax, %rax

xor %rsi, %rsi

lea b+1, %rbx

ערך rbx: \_\_\_\_0x49205F\_.

movb 3(%rbx), %al

ערך al: \_\_\_\_\_0x21\_\_\_\_\_.

mov %bh, %al

xor %al, %sil

shr $3 %rsi

movw -11(%rbx, %rsi, 2), %dx

ערך dx: \_\_\_0x1234\_\_.

shr $2, %rsi

movb $0x66, g\_byte

addb (%rbx, %rsi, 4), g\_byte

ערך הבית שב- g\_byte: \_\_\_\_\_X\_\_\_\_\_\_.

xor %rax, %rax

dec %ax

ערך rax: \_\_\_\_\_\_\_0xFFFF\_\_\_\_\_.

mov $garbage, %rcx

lea c, %rbx

movw arr+4, %ax

ror $4, %ax

ערך rax: \_\_\_\_\_\_\_0x2004\_\_\_\_\_.

xor %ax, %ax

incb %ax

ערך rax: \_\_\_\_\_\_\_\_X\_\_\_\_\_\_.

movq (%rbx), %rbx

mov $0x40, %si

dec %rcx

movl %ebx, 2(%rcx)

התוכן שבבית בכתובת garbage+3 \_\_\_\_\_0x05\_\_\_\_\_\_\_.

mov $78, b

ערך הבית b (הבית שb מהווה פניה אליו): \_\_\_\_\_\_\_\_\_X\_\_\_\_\_\_.

movq $arr, b

ערך הבית c (הבית שc מהווה פניה אליו): \_\_\_\_\_\_\_\_\_\_0x0\_\_\_\_.

movswq (b), %rdx

ערך rdx: \_\_\_\_0x2054\_\_.

mov $0x9937, %ax

cwd

ערך rdx: \_\_\_\_0xFFFF\_\_\_\_\_\_.

movw $-0x33, garbage

idivw garbage

ערך eax: \_\_\_\_0x203\_\_\_\_. ערך edx: \_\_\_\_\_\_0xFFD0\_\_\_\_\_.

movq $0x11, (b)

imul $0x8, b, %rdx

ערך rdx: \_\_\_\_\_\_\_0x88\_\_.

xor %rax, %rax

mov $0xff, %ax

mov $3, %bl

mov $33, %rdx

imulb %bl

ערך ax: \_\_\_0xFFFD\_\_\_\_\_\_. ערך dx: \_\_\_\_0x21\_\_\_\_\_\_\_\_\_.

mov $380, %ax

mov $760, %bx

mul %bx

ערך ax: \_\_\_\_0x6820\_\_\_\_\_. ערך dx: \_\_\_\_0x4\_\_\_\_\_\_.

shl $19, %edx

lea 24(%edx, %eax, 8), %eax

ערך ax: \_\_\_\_\_0x4118\_\_\_\_\_\_. ערך dx: \_\_\_\_\_\_\_0\_\_\_\_.

# 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 / 16; | sarl \_$4\_, \_a\_\_ |
| a = a\*3; | movl a, %eax  lea (%eax, %eax , 2), %eax  mov %eax, a |
| a = a\*8; | movl a, %eax  lea ( , %eax, 8), %eax  mov %eax, a |
| a = 8b - 13 + a; | movl a, %eax  movl b, %ebx  lea -13(%eax, %ebx, 8), %eax  mov %eax, a |
| a |= (1<<16); | Or 0x10000, a |
| a = 4\*a; | imul 4, a, %eax  mov %eax, a |
| a = a\*a\*a\*a\*a\*a\*a\*a; | movl a, %eax  imul %eax  imul %eax  imul %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 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)

רוס המרצה בקורס כתב קטע קוד. לפניכם הקוד שרוס כתב:

A picture containing table

Description automatically generated

1. נתון שבתחילת התוכנית n=10 (בעשרוני).

מה יהיה ערך רגיסטר eax בסיום קטע התוכנית? כתבו את התשובה גם בבסיס דצימלי וגם בהקסדצימלי (וכתבו את כל הבתים שלו ב-hexa)?

**פתרון:**

דצימלי – 89

הקסדצימלי – 0x0059

1. הסבירו במשפט אחד מה עושה התוכנית.

**פתרון:**

מחשבת את האיבר הn+1 בסדרת פיבונצ'י

1. מוניקה הסטודנטית החרוצה שמה לב שעבור n=50 מוחזרת תשובה לא נכונה. מה הסיבה לכך?

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

**פתרון:**

הסיבה לכך היא שעבור n=50 האיבר הn+1 שמתקבל בסדרת פיבונצ'י גדול מדי להציגו ב32 ביט ואינו נכנס למשתנה בגודל 4 בתים (eax) ולכן במקרה הזה המספר שיתקבל שגוי.

המספר הגדול ביותר שניתן לשים ב-n ועדיין לקבל תשובה נכונה הוא 46

1. מוניקה, שרוצה להרשים את רוס, שינתה את הקוד:

A picture containing table

Description automatically generated

לצערה, הקוד עדיין לא תקין. מדוע?

**פתרון:**

בשורה: mov %eax, %edx, אנו מעבירים את הערך של רגיסטר eax ל-32 הביטים התחתונים של rdx, אך בהמשך התוכנית, אנו משתמשים בכל רגיסטר rdx ואיננו יכולים לדעת מה נמצא ב4 הבתים העליונים שלו – ייתכן שיש שם ערך זבל, מה שיפגע בחישוב.

1. לאחר שמוניקה תיקנה את התכנית שלה, רוס ראה שהיא עדיין לא מסופקת, והציע לה דרך לתמוך במספרים גדולים עוד יותר.

רוס מסביר למוניקה (שכבר יודעת!) על הפקודה adc. הסבירו למוניקה איך ניתן לחבר מספרים בגודל 16 בתים בעזרת פקודה זו.

בפרט, חברו את המספר rbx:rax (8 הבייטים התחתונים שמורים בrax, ו8 העליונים שמורים בrbx) אל המספר rdx:rcx.

**פתרון:**

נרצה להשתמש ב-adc. נערוך את הקוד כך שהקוד החדש הינו:

# initialize registers

mov $1, %rax

xorq %rbx, %rbx

mov %rax, %rcx

xorq %rdx, %rdx

mov n, %r10 # new counter

# loop and calculate

loop:

dec %r10

je end

mov %rax, %r8

mov %rbx, %r9

mov %rcx, %rax

mov %rdx, %rbx

add %r8, %rcx

adc %r9, %rdx

jmp loop

end:

ret

1. מוניקה שמה לב שאחת מהפקודות בקוד של רוס, ואף בקוד שלה, מיותרת. על איזו פקודה מדובר, ולמה?

**פתרון:**

הפקודה המיותרת היא: test %ecx, %ecx.

פקודת ה-test הנ"ל מעדכנת את ZF להיות 1 אם %ecx == 0, אבל ZF גם מתעדכן ל-1

אם התוצאה האחרונה היא 0. במקרה שלנו, התוצאה האחרונה תהיה 0 כאשר נכנס ללולאה בפעם

ה- n ונעשה dec %rcx. כלומר רצינו לקפוץ ל-end ולשם כך בקוד השגוי זה נבדק פעמיים במקום פעם אחת.