הטכניון – מכון טכנולוגי לישראל

ארגון ותכנות המחשב

תרגיל 2 - חלק יבש - פתרון

המתרגל האחראי על התרגיל: תומר כץ.

שאלותיכם במייל בעניינים מנהלתיים בלבד, יופנו רק אליו.

כתבו בתיבת **subject**: יבש **2** את"ם.

שאלות בעל-פה ייענו על ידי כל מתרגל.

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.

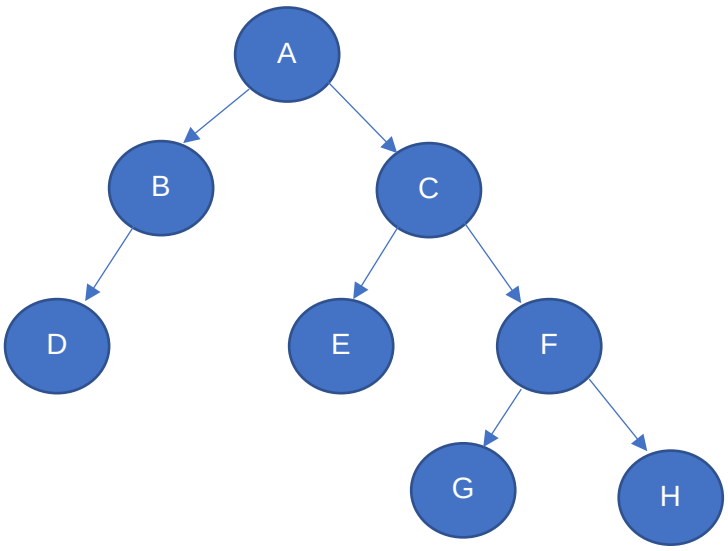
שאלה 1 (45 נק') – שגרות:

ג'וני סטודנט אחראי כל בוקר בשש ו30 כותב קוד אסמבלי. לפניכם מקטע הנתונים שג'וני כתב:

```
.section .data
A:  .long 3
    .quad B
    .quad C
B:  .long 4
    .quad D
    .quad 0
C:  .int 5
    .quad E
    .quad F
D:  .int 7
    .quad 0
    .quad 0
E:  .int 8
    .quad 0
    .quad 0
F:  .int 9
    .quad G
    .quad H
G:  .int 10
    .quad 0
    .quad 0
H:  .int 11
    .quad 0
    .quad 0
```

א. ציירו את הגרף המתקבל מפירוש מקטע הנתונים (מומלץ להסתכל בתרגול 3 תרגיל 1 ולהיזכר שם על אופן פירוש הזיכרון כרשימה מקושרת). בכל צומת בגרף ציינו את התווית המתאימה לו בלבד (אין צורך לציין ערכים נוספים) (3 נקודות)

פתרון:



ג'וני לא מפחד משגרה שוחקת ולכן כותב את השגרה func וקוד שמתשמש בה:

```
.section .text
.global _start
_start:
    mov $8, %esi
    mov $A, %rdi
    call func
    movq $60, %rax
    movq $0, %rdi
    syscall

func:
    pushq %rbp
    movq %rsp, %rbp
    cmp (%rdi), %esi
    jne continue
    mov $1, %eax
    jmp finish

continue:
    cmpq $0, 4(%rdi)
    je next
    pushq %rdi
    mov 4(%rdi), %rdi
    call func
    pop %rdi
    cmp $1, %eax
    je finish
next:
    cmpq $0, 12(%rdi)
    je fail
    pushq %rdi
    mov 12(%rdi), %rdi
    call func
    pop %rdi
    cmp $1, %eax
    je finish

fail:
    mov $0, %eax
finish:
    leave
    ret
```

ב. נתון שבתחילת התוכנית ערך של rsp הוא x.כאשר x הוא מספר בקסדצימלי. מה הוא הערך המקסימלי ומה הערך המינימלי שrsp יכיל לאורך ריצת התוכנית? תנו נוסחא שהמספרים בה הם בבסיס הקסדצימלי (בטאו את התשובה בהאמצעות x). (5 נקודות)

הערך המקסימלי הוא x כי המחסנית רק קטנה לאורך התוכנית.
הערך המינימלי בתוכנית הוא x-40 נבחין שהוא מתקבל כאשר rdi מצביע על D. נבחין שכל קריאה לפונקציה דוחפת למעשה שני דברים (בני 8 בתים) למחסנית. הכוונה היא לכותבת החזרה מהשגרה ולערך rbp שנדחף בכניסה לפוקנציה. בנוסף כל שמתבצעת קריאה ריקורסיבית גם rdi נכנס למחסנית ולכן כאשר A מבצע קריאה ריקורסיבת של הפונקציה על rdi B גם נכנס למחסנית ובאופן דומה גם כשB עושה קריאה על D. מכאן יש עוד שתי דחיפות של 8 בתים למחסנית. נבחין שבעת הקריאה של D לא דוחפים את rdi כי אין קריאה ריקורסיבית.

ג. רשמו מה יהיה פלט הפונקציה עבור קטע הקוד הנוכחי (7 נקודות)

הפלט הוא 1

ד. המירו את הפונקצייה לשפת c על ידי כך שתשלימו את המקומות החסרים בקוד. העיזרו בהגדרת structה שנתונה לכם (10 נקודות):

structה הנתון:

```
typedef struct _Node {
    int data;
    struct _Node *left;
    struct _Node *right;
} Node;
```

הערה1: שני הפרמטרים צריכים להיות תואמים לשני הפרמטרים של פונקצית האסמבלי גם מבחינת תפקיד וגם מבחינת סדר. כלומר, root צריך להתאים בתפקידו לפרמטר הראשון שמועבר לפוקנציה בשפת אסמבלי גם מבחינת הקונבנצייה שלמדנו.

הערה2: אורך הקו לא מלמד על אורך האיבר שצריך להשלים. **מותר להשלים יותר ממילה אחת בכל קו אך לא יותר מפקודה אחת!**

```
_int__ func ( Node *root, _int__ x){
        If (root->data == __x_ )
                _return 1_;
        if (root->left != null)
          if (func(root->left, x))
                return _1_;
        if (root->right != null)
           return _func(root->right, x);
        return 0;
}
```

התקבלו גם תשובות לגבי ערך החזרה שהוא bool

הערה: בסעיפים הבאים יש כל מיני שינויים בקוד. כל שינוי מתקיים רק בסעיף בו מופיע. זאת אומרת הסעיפים לא תלויים אחד בשני.

ה. מוני חבר של ג'וני הוא לא כמו ג'וני. הוא אוהב לעשות שינויים רבים בקוד. הוא מחליט לקחת את המקטע הנתונים של ג'וני ולשנות בכל struct את הint בquad. כלומר מקטע הנתונים ישתנה כך:

```
.data
A:  .quad 3
    .quad B
    .quad C
B:  .quad 4
    .quad D
    .quad 0
```

ובאופן דומה כל שאר האותיות יחליפו את הנתון הראשון במקום בint בquad.
רשמו את השינויים שצריכים להיות בקוד על מנת שיעבוד בצורה תקינה עם מקטע הנתונים החדש (5 נקודות)

נצטרך לבצע את השינויים הבאים:
כיוון וגודל הנתון בצומת הוא עכשיו 8 בתים כל גישה לבן הראשון תצטרך להיות בהזחה של 8 בתים מrdi ולא כ 4 ומכאן כל גישה מהצורה 4(%rdi) תשתנה ל8(%rdi) (שורות 46 ו49). ובאופן דומה כל גישה לבן השני של צומת תוחלף מ12(%rdi) ל 16(%rdi) (שורות 58 ו61).
בנוסף יש צורך לשנות את השימוש בesi לrsi <u>אך לא הורדו נקודות לאלו שלא ציינו שינוי זה</u>

ו. ג'וני מתחיל להתעייף מהשגרה ומחליט לקום ולשנות את מבנה הנתונים באופן הבא:

```
11  D: .int 7
12     .quad A
```

מה יהיה פלט התוכנית? יש לסמן תשובה מבין התשובות הבאות ולנמק במשפט **אחד**: (5 נקודות)

a. התוכנית תסתיים ופלט הפונקציה יהיה 1
b. התוכנית תסתיים ופלט הפונקציה יהיה b
c. התוכנית תכנס ללולאה אנסופית
d. התוכנית תקרוס במהלך ריצה
e. התוכנית כלל לא תבנה

שימו לב לכך שאמנם לוגית יש כאן לולאה אינסופית אבל בכל קריאה ריקורסיבית דוחפים איברים למחסנית.

ז. פתאום ג'וני כמו מוני! מחליט לבצע שינויים נוספים ולא שגרתיים בקוד מול כל שינוי שג'וני מציע עליכם לכתוב האם נכונות השגרה תיפגע (האם יש קלט עבורו השגרה לאחר השינוי שונה מהשגרה לפני השינוי). הסיברו **בקצרה** את תשובתכם!: (10 נקודות)

a. מחיקת הפקודת push וpop שבשורות 60 ו 57..

אין שינוי כי אין צורך לשחזר את ערך rdi ביציאה מהקריאה הריקורסיבית

b. מחיקת הפקודה pop בשורה 60

אין שינוי כי פקודת leave מחזירה את rsp למצב של להצביע על כתובת החזרה ובכך בעצם מוציאה מהמחסנית את כל האיברים.

c. מחיקת push וpop שבשורות 48 51

עלול להיות שינוי בעיות כי במהלך הקריאות הריקורסיביות ערך rdi משתנה ויש צורך לשחזר את ערכו בשביל לסרוק את תת העץ של הבן השני.

d. הוספת פקודה push %rdi אחרי continue בשורה 45

אין שינוי מאותה סיבה שב-b אין שינוי

e. הוספת הפוקדה push %rdi אחרי continue בשורה 45, שינוי פוקדת הpop שבשורה 51 לפקודה: mov (%rsp), %rdi ומחיקת הפקודת push וpop בשורות 60 57.

אין שינוי פקודת הmov שהוספנו במקום פקודת הpop משחזרת את rdi דבר שמאפשר מעבר על תת העץ השני של הצומת. בנוסף זה שאין pop שמוציא את rdi לא פוגע בגלל הפקודה leave.

שאלה 2 (30 נק') – קריאות מערכת:

ג'ואי מרגיש מתוסכל מכך שחבריו חושבים שהוא פחות חכם מהם. לכן, הוא מחליט להרשים אותם בעזרת כתיבת קוד אסמבלי.

א. לפניכם מקטע הנתונים שג'ואי כתב מבלי ערכי הנתונים עצמם:

```
.section .data
msg1: .ascii ???????
msg2: .ascii ???????
msg1_len: .quad ____
msg2_len: .quad ____
all_msg_len: .quad ____
```

ג'ואי לא יודע עדיין אילו מחרוזות הוא יכתוב. עליכם להשלים את המקומות הריקים שקשורים לאורכי המחרוזות כך שמשתנה msg1_len יהיה האורך של msg1, בmsg2_len יהיה האורך של msg2 ובמשתנה all_msg_len יהיה שווה לסכום אורכי המחרוזות msg1 וmsg2. שימו לב עליכם לעשות זאת בצורה כזו שהאורכים יהיו נכונים בעת ריצת התוכנית ללא קשר לאיזה מחרוזות ג'ואי ישים בmsg1 ובmsg2. (3 נקודות)

פתרון:

```
.section .data
msg1: .ascii ???????
msg2: .ascii ???????
msg1_len: .quad msg2 - msg1
msg2_len: .quad msg1_len - msg2
all_msg_len: .quad msg1_len - msg1
```

ב. כעת נתון מקטע הנתונים שכולל את המחרוזות:

```
.section .data
msg1: .ascii "HOW YOOOU DOOIN?"
msg2: .ascii "JOEY DOESN'T SHARE FOOD!"
msg1_len: .quad ____
msg2_len: .quad ____
all_msg_len: .quad ____
```

לפניכם נתונה התוכנית שג'ואי כתב ומוצגת כאן גם הפונקציה שכתב:

```
Joey_func:
        cmp %rbx, %r9
        je end
        # addb $0x20, (%rsi)
        test $1, %rbx
        jnz skip
        syscall
skip:   inc %rsi
        inc %rbx
        call Joey_func
end:    ret
```

```
.section .text
.global _start
_start:
    mov $msg1, %rsi
    mov $1, %rdi
    mov $1, %rdx
    mov $1, %rax
    xor %rbx, %rbx

    movq msg1_len, %r9
    call Joey_func
```

מה יודפס בסיום ריצת הקוד? (שימו לב השורה השלישית בפונקציה נמצאת בהערה ולא רלוונטית לסעיף). (5 נקודות)

HWYOUDON

ג. כעת מורידים את הסולמית שנמצאת בפונקציה (וכעת הפקודה חלק מהקוד) בנוסף מחליפים את השורה movq msg1_len, %r9 בשורה: movq all_msg_len, %r9.

הערה: שינויים אלו ילוו אותנו גם בסעיפים הבאים (בסעיפים ד - ו השינויים בסעיף ג עדיין תקפים).

מה יודפס כעת בסיום ריצת הקוד? (5 נקודות)

hwyoudonje@osG@hr@od

ד. בזמן שג'ואי אכל בסלון סנדוויץ, חיית המחמד שלו (אפרוח) טיילה על המקלדת והוסיפה את הפקודה:

inc %r9. הפקודה נוספה שורה לפני הקריאה לפונקציה של ג'ואי בתוכנית הראשית.

מה יהיה פלט התוכנית כעת? (2 נקודות)

hwyoudonje@osG@hr@od0

ה. חברה טובה של ג'ואי פיבי אמרה לו ששימוש ברגיסטר r9 מביא מזל רע. ג'ואי נלחץ נורא והחליט שיש לבצע שינוי בקוד מבלי לשנות את תוצאות הפעולה של הפונקציה (כלומר הפלט צריך להיות זהה). כיוון ולא ידע איך לשנות את הקוד הוא החליט לבקש את עזרת חבריו.

בסעיף הזה יופיעו העצות של כל החברים. עליכם לרשום ליד כל עצה האם היא לדעתכם תעזור לג'ואי. נמקו **בקצרה(!)** (10 נקודות)

צ'נדלר מציע להחליף את השימוש בr9 בשימוש בrcx.

לא יעזור, לאחר הקיראת מערכת rcx יכיל את הכתובת של הפקודה הבאה לביצוע אחרי הקריאת מערכת ולכן התוכן שלו ידרס לאחר הקריאה הראשונה. מכאן באיטרציה הבאה ערכו כבר יהיה שונה מערכו המקורי.

מוניקה מציעה להחליף את השימוש בr9 בשימוש בr11.

לא יעזור. לאחר הקריאת מערכת ערך r11 יהיה שווה לערך של אוגר הדגלים לפני קריאת המערכת. לכן, התוכן המקורי של אוגר זה ידרס אחרי הקיראת מערכת הראשונה. מכאן באיטרציה הבאה ערכו כבר יהיה שונה מערכו המקורי.

פיבי מציעה להחליף את השימוש בr9 בשימוש בrdi.

לא יעזור, rdi קובע לאן תודפס המחרוזת. לכן אסור לדרוס אותו.

רייצ'ל מציעה להחליף את השימוש בr9 בשימוש בr12.

השינוי יעזור. אין מניעה להשתמש ברגיסטר זה.

רוס מציע להחליף את השימוש בr9 בשימוש בrbp.

יעזור, אין שימוש באוגר הזה ולמעשה הוא לא חייב לבוא בהקשר של מחסנית כלל.

ו. חבריו של ג'ואי מסבירים לו שהשימוש שלו ברקורסיה מיותר ובזבזני והוא יכול את אותו קוד בדיוק לכתוב בלולאות. ג'ואי מחליט לבצע את השינויים הבאים:
בתוכנית הראשית בשורה שלפני ביצוע הפקודה call ג'ואי מוסיף את הפקודה:
mov $Joey_func, %rcx

ובתוך הפונקציה ג'ואי מוחק את השורה בה יש שימוש בפקודה call והחליף אותה בפקודה:
jmp *%rcx
שימו לב שהתווית end נמצאת אחרי פקודה זו.
לצורך הבהרה הפונקציה נראת כך כעת:

```
Joey_func:
        cmp %rbx, %r9
        je end
        addb $0x20, (%rsi)
        test $1, %rbx
        jnz skip
        syscall
skip:   inc %rsi
        inc %rbx
        jmp *%rcx
end:    ret
```

כיצד שינוי זה ישפיע על אופן ריצת הפונקציה. מה יודפס אם נריץ את הפונקציה? (5 נקודות)
האיטרציה הראשונה תתבצע ותודפס האות h אך לאחר הקריאת מערכת הראשונה הערך של rcx ישתנה להיות הכתובת של השורה הבאה אחרי syscall. לכן ביצוע הפקודה: jmp *%rcx שקולה למעשה לקפיצה לתווית skip. ומכאן קיבלנו לולאה אינסופית.

שאלה 3 (25 נק') – רמות הרשאה ואוגר הדגלים:

א. הפקודה pushfq דוחפת את הערך של אוגר הדגלים למחסנית. והפקודה popfq מוציאה את אוגר הדגלים מהמחסנית. הסבירו כיצד באמצעות שילוב של שתי פקודות אלו ניתן להדליק את הדגלים OF וCF. שימו לב במידה ואחד הדגלים כבר דלוק יש להשאירו דלוק כלומר, בסיום התהליך על שני הדגלים להיות דולקים. אין לשנות את שאר הביטים בריגסטר הדגלים. בנוסף, אין לשנות אף רגיסטר שהוא לא rflags, rip, rsp (גם לא באופן זמני). (7 נקודות)
הערה: במידה ובדקתם את עצמכם באמצעות דיבגר וראיתם שנדלק גם דגל TF זה בסדר תלמדו בהמשך מדוע הוא נדלק תוך כדי דיבוג.
הכניסו את אוגר הדגלים למחסנית על יד pushfq. כעת נבחין שדגל OF הוא ביט מספר 11 וCF הוא ביט מספר 0. נבצע פקודת or בין ראש המחסנית לערך 801 בהקסדצימלי. נבחין שערך זה הוא בבינארי הכל 0 מלבד ביטים מספר 0 ו11.. כתוצאה מהor ביטים 0 ו11 בוודאות יהיו דולקים בראש המחסנית ושאר הביטים לא ישתנו. לסיום נבצע popfq ובכך נעדכן את אוגר הדגלים.

ב. הולי התחמנית רוצה לאפשר לעצמה גישה ישירה אל התקני הקלט פלט ללא צורך בקריאות מערכת. איזה שינוי **באוגר הדגלים** יכול לעזור להולי במטרתה? (4 נקודות)
הערה: לא צריך לציין פקודה ספציפית, רק להגיד מה צריך לעשות ברמה התיאורטית

ביטים 12 ו13 באוגר הדגלים מייצגים את הIOPL. IOPL הוא התנאי על רמת ההרשאה שמשתמש צריך כדי לגשת להתקני קלט פלט. אם מתקיים $CPL \leq IOPL$ אז המשתמש יוכל לגשת להתקני הקלט פלט. אם הולי תשנה את IOPL להיות הערך המקסימלי (11) כלומר לרמת ההרשאה הנמוכה ביותר, אז התנאי הזה יתקיים גם ברמת ההרשאה הנמוכה ביותר. וכך היא תוכל לגשת ישרות להתקני הקלט פלט.

ג. הולי מחליטה לנסות את התעלול מסעיף א' רק שבמקום לשנות את CF וOF היא רוצה לשנות את IOPL. להפתעתה, היא לא מצליחה לשנות את הביטים הללו. הסבירו מה ההגיון בכך שהיא לא מצליחה לשנות את IOPL? התייחסו לצורך בקריאות מערכת (4 נקודת)
במידה והיה ניתן לשנות את הIOPL היה אפשר לגשת ישירות להתקני הקלט פלט מכל רמת הרשאה ולמעשה לא היה צורך בקריאות מערכת. גישה ישירה להתקני קלט פלט הייתה פותחת שער למשתמשים זדוניים לפגוע בכלל התוכניות במחשב ולא רק בתוכנית שהם עצמם מריצים.

ד. וולי החבר המבולבל של הולי מתלבט כיצד ניתן לחסום פסיקות תוכנה לכן הוא שואל את הולי. אילו מבין התשובות הבאות על הולי לענות לו? יש לסמן את האפשרות הנכונה. (5 נקודות)
1. כיבוי דגל IF באוגר הדגלים
2. הדלקת דגל IF באוגר הדגלים
3. שינוי CPL ל00
4. לא ניתן לחסום פסיקות תוכנה.

לא ניתן לחסום פסיקות תוכנה כיוון והן מגיעות מבפנים בצורה יזומה של המעבד. כלומר המעבד הוא זה שמבקש את הפסיקה ולא ממשיך בלעדיה ומכאן אין טעם לחסום את הפסיקה. תשובה נכונה היא 4.

ה. כעת נתון שוולי הצליח להגיע למצב שבו CPL שווה ל0. וולי מעוניין לחסום פסיקות חומרה שאינן מועברות דרך כניסת NMI. כיצד הוא יכול לעשות זאת? (5 נקודות)

1. כיבוי דגל IF באוגר הדגלים
2. הדלקת דגל IF באוגר הדגלים
3. עליו לחבר את הפסיקות לכניסת NMI ואז לכבות את דגל IF
4. לא ניתן לחסום פסיקות חומרה ולכן לא יצליח.

הפסיקות שאינן מועברות דרך כניסת NMI כן ניתנות לחסימה. אלו שמחוברות לכנסית NMI הן אלו שלא ניתנות לחסימה (הפסיקות שמדווחות על בעיות חומרה קריטיות).הדרך לחסום היא באמצעות כיבוי דגל IF.