מגישים: אנטוני סלבין, נועם גולדנשטיין

שאלה 1 – קידוד פקודות

לאחר מתקפת הסייבר הכבדה על הטכניון הבינו מומחי המחשבים בבניין טאוב שנוצרה בעיה.

1. בגלל המתקפה האחרונה, כל האסמבלרים בפקולטה הפסיקו לתרגם פקודות לשפה מכונה. עזרו לסגל קומפילציה לתקן את הנזק ע"י תרגום הפקודות הבאות בצורה תקינה מאסמלי (AT&T syntax) לשפת מכונה.
הערה: יש למלא את הערכים בhexadecimal.

```
<start>:
400000: 4D 31 E4                    xor %r12, %r12

400003: 49 C1 E8 02                 shr $2, %r8

400007: 83 E9 05                    sub $5, %ecx

40000A: 4C 8D 05 0B 00 00 00        lea 11(%rip) , %r8

400011: FF 25 34 12 00 00           jmp *0x1234(%rip)
```

2. מה יהיה ערכו של רגיסטר 8r בעת ההגעת הקוד לכתובת 0x400011? : 0x40001C
3. סגל קומפי הצליח לתקן את כל האסמבלרים בעזרתכם אבל כעת התגלתה בעיה אחרת. המעבדים בפקולטה לא מצליחים לעשות decode לפקודות. תרגמו את הרצף הבינארי הבא מפקודות מכונה לפקודות אסמבלי.

```
55 48 89 e5 48 83 ec 08
```

הרצף הנ"ל נתון בהקסא, משמאל לימין (הבית הראשון ברצף הוא 0x55). את רצף הפקודות שמקודד עליכם לכתוב בשורות הבאות:

push %rbp

mov %rsp, %rbp

sub $8, %rsp

<u>הערות</u>: כל פקודה חייבת להופיע בשורה נפרדת. ניתן להשאיר שורות ריקות.

מגישים: אנטוני סלבין, נועם גולדנשטיין

שאלה 2 – קבצי ELF וקישור סטטי

לרגל המונדיאליטו חברכם גיא החילט לכתוב תוכנית באסמבלי המתפרשת על שני קבצים.

U20worldCup1.asm

```
.global _start
.extern s, len, overtime

.section .text
_start:
    movq $1, %rax
    movq $1, %rdi
    movq $s, %rsi
    movq len , %rdx
    syscall
    movl $end, overtime(%rip)
    jmpq *overtime
    movq $60, %rax
    syscall
end:
    imulq %rax, %rdx
    movq %rdx , %rdi
    movq $60, %rax
    syscall
```

U20worldCup2.asm

```
.global s,len, overtime

.extern _start

.section .data
s: .ascii "El EL Israel!\n"
len: .quad len-s
overtime: .quad _start
```

להלן תוכן הקבצים:

גיא התלהב מהקוד שכתב והריץ בטרמינל את הפקודות הבאות:

```
as U20worldCup1.asm -o U20worldCup1.o
as U20worldCup2.asm -o U20worldCup2.o
ld U20worldCup1.o U20worldCup2.o -o U20worldCup.out
./U20worldCup.out
```

גיא טס לצפות במשחקים בארגנטינה ושם הוא דיבר עם אוהדים מכל העולם. התברר לגיא שאף אחד מהם לא יודע איך טבלאות הסמלים של שני הקבצים יראו.

1) עזרו לאוהדי העולם ומלאו את טבלאות הסמלים של `U20worldCup1.o` ושל `U20worldCup2.o`.

הערות:

1. ניתן להשאיר שורות ריקות
2. בעמודה Nxt עליכם לכתוב את שם ה section או UND (ולא מספר).

U20worldCup1.o symbol table:

(section) Nxt	Bind(נראות)	name
.text	global	_start
Und	global	s
Und	global	len
Und	global	overtime
.text	local	end

השאלה ממשיכה בעמוד הבא

U20worldCup2.o symbol table:

(section) Nxt	Bind(נראות)	name
.data	global	s
.data	global	len
.data	global	overtime
Und	global	_start

גיא החליט להתחפש כדי שאף אחד לא יזהה אותו ולכן גם חבריו של גיא לא מזהים אותו. בשביל לדעת באמת מי זה גיא אותם חברים הראו לו את טבלת הsection header של הקובץ U20worldCup1.o שנוצרה ע"י הרצת הפקודה : readelf -S U20worldCup1.o. ואת התוכן של הקובץ U20worldCup1.o ע"י הפקודה hexdump. להלן התוצאות:

Readelf -S U20worldCup1.o:

```
Section Headers:
  [Nr] Name              Type             Address           Offset
       Size              EntSize          Flags  Link  Info  Align
  [ 0]                   NULL             0000000000000000  00000000
       0000000000000000  0000000000000000           0     0     0
  [ 1] .text             PROGBITS         0000000000000000  00000040
       0000000000000049  0000000000000000  AX       0     0     1
  [ 2] .rela.text        RELA             0000000000000000  00000188
       0000000000000078  0000000000000018   I       5     1     8
  [ 3] .data             PROGBITS         0000000000000000  00000089
       0000000000000000  0000000000000000  WA       0     0     1
  [ 4] .bss              NOBITS           0000000000000000  00000089
       0000000000000000  0000000000000000  WA       0     0     1
  [ 5] .symtab           SYMTAB           0000000000000000  00000090
       00000000000000d8  0000000000000018           6     5     8
  [ 6] .strtab           STRTAB           0000000000000000  00000168
       000000000000001b  0000000000000000           0     0     1
  [ 7] .shstrtab         STRTAB           0000000000000000  00000200
       0000000000000031  0000000000000000           0     0     1
```

2) אותם חברים רצו שגיא יסמן בHexdump את מקטע הtext בשביל להוכיח שהוא הגיא האמיתי. עזרו לגיא וסמנו את מקטע הtext בhexdump הבא:

Hexdump U20worldCup1.o:

```
0000000 457f 464c 0102 0001 0000 0000 0000 0000
0000010 0001 003e 0001 0000 0000 0000 0000 0000
0000020 0000 0000 0000 0000 0238 0000 0000 0000
0000030 0000 0000 0040 0000 0000 0040 0008 0007
0000040 c748 01c0 0000 4800 c7c7 0001 0000 c748
0000050 00c6 0000 4800 148b 0025 0000 0f00 c705
0000060 0005 0000 0000 0000 ff00 2524 0000 0000
0000070 c748 3cc0 0000 0f00 4805 af0f 48d0 d789
0000080 c748 3cc0 0000 0f00 0005 0000 0000 0000
0000090 0000 0000 0000 0000 0000 0000 0000 0000
00000a0 0000 0000 0000 0000 0000 0000 0003 0001
00000b0 0000 0000 0000 0000 0000 0000 0000 0000
00000c0 0000 0000 0003 0003 0000 0000 0000 0000
00000d0 0000 0000 0000 0000 0000 0000 0003 0004
00000e0 0000 0000 0000 0000 0000 0000 0000 0000
```

השאלה ממשיכה בעמוד הבא

מגישים: אנטוני סלבין, נועם גולדנשטיין

לצורך הסעיף הבא נתון פלט הobjdump של U20worldCup1.o:

```
0000000000000000 <_start>:
   0:  48 c7 c0 01 00 00 00       mov    $0x1,%rax
   7:  48 c7 c7 01 00 00 00       mov    $0x1,%rdi
   e:  48 c7 c6 00 00 00 00       mov    $0x0,%rsi
  15:  48 8b 14 25 00 00 00       mov    0x0,%rdx
  1c:  00
  1d:  0f 05                      syscall
  1f:  c7 05 00 00 00 00 00       movl   $0x0,0x0(%rip)        # 29 <_start+0x29>
  26:  00 00 00
  29:  ff 24 25 00 00 00 00       jmpq   *0x0
  30:  48 c7 c0 3c 00 00 00       mov    $0x3c,%rax
  37:  0f 05                      syscall

0000000000000039 <end>:
  39:  48 0f af d0                imul   %rax,%rdx
  3d:  48 89 d7                   mov    %rdx,%rdi
  40:  48 c7 c0 3c 00 00 00       mov    $0x3c,%rax
  47:  0f 05                      syscall
```

3) מלאו את הטבלה הבאה של הrelocation של הtext section:

offset	type	Symbol name	addend
0x11	קבוע	s	0
0x19	קבוע	len	0
0x21	יחסי	overtime	-8
0x25	קבוע	.text	0x39
0x2c	קבוע	overtime	0

<u>הערה</u>: ב"Type "ניתן להשלים רק "יחסי" או "קבוע" ואין צורך להשתמש בשמות המלאים.

4) האם בניית התוכנית תצליח? (יווצר קובץ הרצה תקין?) הקיפו את התשובה הנכונה. **כן / לא**

5) בהמשך לסעיף הקודם, אם עניתם לא הסבירו מדוע. אם כן רשמו מה יהיה פלט התוכנית ומה ערך היציאה שלה.

פלט: "El EL Israel!\n" ערך יציאה: 196

מגישים: אנטוני סלבין, נועם גולדנשטיין

שאלה 3 – קישור דינמי

1) לפניכם קוד של ספריה דינאמית שקומפלה:

```
extern int value;

void change_value(int a, int b){
        value++;
        value = a + 2*value * b;
        value = value -2;
}
```

כמה תיקונים יצטרך לעשות הקשר הדינאמי עבור הסמל value? הסבירו את איפה יתבצעו התיקונים.

ידרש תיקון אחד והוא יהיה בכניסה המתאימה ב GOT

2) נתון לכם PLT של תוכנה מסוימת.

```
Disassembly of section .plt:

0000000000001020 <.plt>:
    1020:       ff 35 e2 2f 00 00       pushq  0x2fe2(%rip)        # 4008 <_GLOBAL_OFFSET_TABLE_+0x8>
    1026:       ff 25 e4 2f 00 00       jmpq   *0x2fe4(%rip)        # 4010 <_GLOBAL_OFFSET_TABLE_+0x10>
    102c:       0f 1f 40 00             nopl   0x0(%rax)

0000000000001030 <printf@plt>:
    1030:       ff 25 e2 2f 00 00       jmpq   *0x2fe2(%rip)        # 4018 <printf@GLIBC_2.2.5>
    1036:       68 00 00 00 00          pushq  $0x0
    103b:       e9 e0 ff ff ff          jmpq   1020 <.plt>
```

נתמקד בפקודה בכתובת 0x1030.

i) מה סוג הקפיצה שבו משתמשים?

אבסולוטית עקיפה

ii) מהו סוג האופרנד (אם מדובר בכתובת, ציינו שיטת מיעון)?

סוג האופרנד הוא הכתובת בזיכרון של הכניסה ב-GOT שבה שמורה הכתובת של printf

ושיטת המיעון היא יחסית (rip + disp)

iii) האם ידוע לאיזה כתובת נקפוץ בעת ביצוע הפקודה? אם כן מהי הכתובת ואם לא מדוע לא ניתן לדעת ומה כן ניתן לדעת על אותה כתובת. (תשובה בעמוד הבא)

אי אפשר לדעת לאיזה כתובת נקפוץ כי לא ידוע אם התבצע lazy binding וגם במידה וכן לא ידוע אם זו הפעם הראשונה בה קראנו לפונקציה printf. מה שאנו יודעים על הכתובת הזו הוא שהיא תיטען לכתובת 0x4018, ולכן אנחנו יודעים את מה שכתוב בתשובה לשאלה 3 (הבאה).

3) הסבירו מה תכיל הכתובת 0x4018 בתחילת ריצת התוכנית. התייחסו למקרה שבו התוכנית קומפלה עם lazy binding ולמקרה שבו היא לא.

אם התוכנית קומפלה עם lazy binding הכתובת הנ"ל תכיל את הכתובת 0x1036 ב-plt, אם התוכנית קומפלה בלי lazy binding הכתובת הנ"ל תכיל את הכתובת שבה printf שמורה (ממומשת) בזיכרון

4) הסבירו מתי נרצה לקמפל עם lazy binding ומתי לא נרצה.

נרצה לקמפל עם lazy binding אם טוענים הרבה ספריות עם הרבה פונקציות. אנחנו לא נרצה לקמפל עם lazy binding אם התוכנית שלנו רגישה לחוסר וודאות ועלולה לקבל עצירה נקודתית בשביל התיקונים הדרושים בזמן הריצה ולכן תעדיף לעשות את כל התיקונים לפני הריצה בנוסף אם הלינקר הדינאמי זורק שגיאה נדע זאת לפני ריצת התוכנית ולא במקום אקראי במהלך הריצה.