ארגון ותכנות המחשב

תרגיל 4 - חלק יבש

המתרגל האחראי על התרגיל: אדיר רחמים.

שאלותיכם במייל בעניינים מנהלתיים בלבד, יופנו רק אליו.

כתבו בתיבת **subject**: יבש **4** אתם.

שאלות בעל-פה ייענו על ידי כל מתרגל.

הוראות הגשה (לקרוא!!!):

- ההגשה בזוגות.
- שאלות הנוגעות לתרגיל יש לשאול דרך הפיאצה בלבד.
- על כל יום איחור או חלק ממנו, שאינו בתיאום עם המתרגל האחראי על התרגיל, יורדו 5 נקודות.
  - ניתן להגיש לכל היותר באיחור של 3 ימים (כאשר שישי ושבת נחשבים יחד כיום אחד בספירה).
  - הגשות באיחור יש לשלוח למייל של אחראי התרגיל בצירוף פרטים מלאים של המגישים (שם+ת.ז).

נושא התרגיל: קישור סטטי, קישור דינמי וניהול זיכרון

חומר דרוש: לשאלה 1 – עד תרגול 8, לשאלה 2 – עד תרגול 12, לשאלה 3 – עד תרגול 9

שאלה 1

נתונים ארבעת קבצי הקוד (a.asm, b.asm, c.c, d.c) הבאים:

a.asm:

```
.global a_1, _start, msg, msg_len
.extern b_turn

.data
a_1: .int 0x4
msg: .ascii “I’m excited to see what will be printed!\n”
msg_len: .quad msg_len – msg

.text
_start:
    movq $msg, %rdi
    movq (msg_len), %rsi
    call my_print
    call b_turn
    mov %rax, %rdi
    mov $60, %rax
    syscall
my_print:
    mov $1, %eax
    mov %rsi, %rdx
    mov %rdi, %rsi
    mov $1, %esi
    syscall
    ret
```

המשך השאלה בעמוד הבא

b.asm:

```
.global b_turn, everybody_print
.extern d_turn, len
.data
msg: .ascii “I “
msg_len: .int 0x1

.text
b_turn:
    movl $6, len(%rip)
    movq $msg, %rdi
    movq $msg_len-msg, %rsi
    call everybody_print
    call d_turn
    ret

everybody_print:
    mov $1, %eax
    mov %rsi, %rdx
    mov %rdi, %rsi
    mov $1, %esi
    syscall
    ret
```

המשך השאלה בעמוד הבא

c.c:

```
extern int everybody_print(char* msg, int msg_len);
static char* msg = “ATAM!\n“;
int len = 6;
static int var;
int c_turn(){
everybody_print(msg, len);
return 0;
}
```

d.c:

```
extern int var;
extern int everybody_print(char* msg, int msg_len);
extern int c_turn();
static char* msg = “love “;
int msg_len = 5;

int d_turn(){
var = 3;
everybody_print(msg, msg_len);
return c_turn();
}
```

התוכנית הודרה וקושרה באופן הבא:

```
gcc -nostdlib -fno-pic -c -o c.o c.c

gcc -nostdlib -c -o d.o d.c
as -o a.o a.asm
as -o b.o b.asm
ld -o abc.out a.o b.o c.o d.o
```

המשך השאלה בעמוד הבא

כמו כן להלן פלט של שתי פקודות נוספות שהורצו:

objdump -d c.o

```
c.o:     file format elf64-x86-64


Disassembly of section .text:

0000000000000000 <c_turn>:
   0:   55                      push   %rbp
   1:   48 89 e5                mov    %rsp,%rbp
   4:   8b 15 00 00 00 00       mov    0x0(%rip),%edx        # a <c_turn+0xa>
   a:   48 8b 05 00 00 00 00    mov    0x0(%rip),%rax        # 11 <c_turn+0x11>
  11:   89 d6                   mov    %edx,%esi
  13:   48 89 c7                mov    %rax,%rdi
  16:   e8 00 00 00 00          callq  1b <c_turn+0x1b>
  1b:   b8 00 00 00 00          mov    $0x0,%eax
  20:   5d                      pop    %rbp
  21:   c3                      retq
```

objdump -d d.o

```
d.o:     file format elf64-x86-64


Disassembly of section .text:

0000000000000000 <d_turn>:
   0:   55                      push   %rbp
   1:   48 89 e5                mov    %rsp,%rbp
   4:   c7 05 00 00 00 00 03    movl   $0x3,0x0(%rip)        # e <d_turn+0xe>
   b:   00 00 00
   e:   8b 15 00 00 00 00       mov    0x0(%rip),%edx        # 14 <d_turn+0x14>
  14:   48 8b 05 00 00 00 00    mov    0x0(%rip),%rax        # 1b <d_turn+0x1b>
  1b:   89 d6                   mov    %edx,%esi
  1d:   48 89 c7                mov    %rax,%rdi
  20:   e8 00 00 00 00          callq  25 <d_turn+0x25>
  25:   b8 00 00 00 00          mov    $0x0,%eax
  2a:   e8 00 00 00 00          callq  2f <d_turn+0x2f>
  2f:   5d                      pop    %rbp
  30:   c3                      retq
```

המשך השאלה בעמוד הבא

ענו על הסעיפים הבאים:

1. רשמו את טבלת הסמלים של כל קובץ *.o לפי הפורמט הבא(כפי שראינו בתרגול):
ניתן להשאיר שורות ריקות.

a.o

שם	נראות	Section

b.o

שם	נראות	Section

המשך השאלה בעמוד הבא

c.o

שם	נראות	Section

d.o

שם	נראות	Section

המשך השאלה בעמוד הבא

2. מלאו את טבלאת ה-relocation של כל אחד מהקבצים לפי הפורמט הבא (אין צורך לציין כתובת מדויקת לתיקון, גם אם אתם יודעים אותה):

a.o

שורה(רשמו את הפקודה במלואה)	סוג(יחסי/אבסולוטי)	Symbol + addend

b.o

שורה(רשמו את הפקודה במלואה)	סוג(יחסי/אבסולוטי)	Symbol + addend

המשך השאלה בעמוד הבא

c.o

שורה(רשמו את הפקודה במלואה)	סוג(יחסי/אבסולוטי)	Symbol + addend

d.o

שורה(רשמו את הפקודה במלואה)	סוג(יחסי/אבסולוטי)	Symbol + addend

3. האם התוכנית תהודר ותקושר כשורה? ענו כן/לא.
אם התשובה היא כן, מה התוכנית תדפיס בריצתה? האם היא תסיים את ריצתה בהצלחה?
אם לא, ציינו את כל הסיבות לכך שלא נוצר קובץ ריצה.

תשובות בלי הסבר לא יתקבלו.

שאלה 2

נתון רצף פונקציות בקוד מסוים ביחד עם כמות הזיכרון הדרושה בעבורם.

הניחו שהפונקציות האלו רצות אחת אחרי השנייה (J1 ראשונה, J2 אחריה, ואז J3 ולבסוף J4) וצריכת הזיכרון של הקוד בין כל אחת מארבע הפונקציות הללו היא זניחה.

להלן פירוט צריכת הזיכרון של ארבעתן:

Function number	Memory required
J1	25 K
J2	200K
J3	500K
J4	60K

וכן זוהי תמונת הזיכרון (האזורים שהם Reserved שייכים לחלקים קודמים באותה תוכנית ולכן כבר מוקצים ולא ניתן להקצות בהם):

B1-20K FREE	10K Reserved	**B2**-60K FREE	15K Reserved	**B3**-800K FREE	400K Reserved	**B4**-500K FREE	100K Reserved	**B5**-200K Free

כלומר, זוהי טבלת הבלוקים הפנויים בזיכרון:

Memory block	Memory block size
B1	20K
B2	60K
B3	800K
B4	500K
B5	200K

1. מהו ה-External Fragmentation של הזיכרון רגע לפני תחילת ריצת J1?
2. הקוד הנ"ל רץ עם מנהל זיכרון, שאחראי לבצע את פעולות הקצאות ושחרור הזיכרון לפי בקשות המשתמש (אין GC). <u>רשמו את תמונת הזיכרון</u> בסיום הקצאת הזיכרון של המנהל, לפי שיטות הקצאת זיכרון הבאות:
   I. First fit
   II. Best fit
   III. Worst fit

<u>וכן מהו ה-Fragmentation שיהיה בכל שלב</u>. (דיוק של 2 ספרות אחרי הנקודה מספיק)

שימו לב- ניתן להקצות חלקי בלוק, למשל, אם מתוך בלוק בגודל 500 יוקצו 100, 400 יישארו פנויים להקצאות הבאות.

אם ההקצאה נכשלת, ציינו זאת וכן באיזה שלב(בהקצאה של איזה פונקציה) ההקצאה תיכשל.

שאלה 3

נתונים הקבצים הבאים:

```
#include <stdio.h>
int print_dynamic(int a, int b);

int count = 0;

int main() {
printf("Will it run?\n");
print_dynamic(1,2);
return 0;
}
```

הרצנו את הפקודות הבאות:

gcc -shared -fPIC -o dynamic.so dynamic.c

sudo mv dynamic.so /usr/lib

gcc -no-pie -o

```
#include <
extern in

int print_dynamic(int a,
c
printf("a+b equals: %d\n
ret
```

המשך השאלה בעמוד הבא

וחלק מהפלט שקיבלנו עבור הפקודה האחרונה הינו:

```
Disassembly of section .plt:

0000000000400510 <.plt>:
  400510:       ff 35 f2 0a 20 00       pushq  0x200af2(%rip)        # 601008 <_GLOBAL_OFFSET_TABLE_+0x8>
  400516:       ff 25 f4 0a 20 00       jmpq   *0x200af4(%rip)        # 601010 <_GLOBAL_OFFSET_TABLE_+0x10>
  40051c:       0f 1f 40 00             nopl   0x0(%rax)

0000000000400520 <print_dynamic@plt>:
  400520:       ff 25 f2 0a 20 00       jmpq   *0x200af2(%rip)        # 601018 <print_dynamic>
  400526:       [redacted]              1
  40052b:       e9 e0 ff ff ff          jmpq   400510 <.plt>

0000000000400530 <printf@plt>:
  400530:       ff 25 ea 0a 20 00       jmpq   *0x200aea(%rip)        # 601020 <printf@GLIBC_2.2.5>
  400536:       [redacted]              2
  40053b:       e9 d0 ff ff ff          jmpq   400510 <.plt>
```

```
0000000000400627 <main>:
  400627:       55                      push   %rbp
  400628:       48 89 e5                mov    %rsp,%rbp
  40062b:       48 8d 3d b2 00 00 00    lea    0xb2(%rip),%rdi        # 4006e4 <_IO_stdin_used+0x4>
  400632:       b8 00 00 00 00          mov    $0x0,%eax
  400637:       e8 3                    callq  4 <printf@plt>
  40063c:       be 02 00 00 00          mov    $0x2,%esi
  400641:       bf 01 00 00 00          mov    $0x1,%edi
  400646:       e8 5                    callq  6 <print_dynamic@plt>
  40064b:       b8 00 00 00 00          mov    $0x0,%eax
  400650:       5d                      pop    %rbp
  400651:       c3                      retq
  400652:       66 2e 0f 1f 84 00 00    nopw   %cs:0x0(%rax,%rax,1)
  400659:       00 00 00
  40065c:       0f 1f 40 00             nopl   0x0(%rax)
```

וכן הפלט עבור הפקודה: readelf -r main.ot:

```
Relocation section '.rela.dyn' at offset 0x498 contains 2 entries:
  Offset          Info           Type           Sym. Value    Sym. Name + Addend
000000600ff0  000300000006 R_X86_64_GLOB_DAT 0000000000000000 __libc_start_main@GLIBC_2.2.5 + 0
000000600ff8  000400000006 R_X86_64_GLOB_DAT 0000000000000000 __gmon_start__ + 0

Relocation section '.rela.plt' at offset 0x4c8 contains 2 entries:
  Offset          Info           Type           Sym. Value    Sym. Name + Addend
7             000100000007 R_X86_64_JUMP_SLO 0000000000000000 print_dynamic + 0
8             000200000007 R_X86_64_JUMP_SLO 0000000000000000 printf@GLIBC_2.2.5 + 0
```

המשך השאלה בעמוד הבא

ענו על הסעיפים הבאים:

1. כתבו את החסר בכל אחד מהריבועים מעלה:
   – 1
   – 2
   – 3
   – 4
   – 5
   – 6
   – 7
   – 8

2. מה תהיה הנראות(ה-Binding) של התווית print_dynamic בטבלת הסמלים הדינאמית של dynamic.so(LOCAL/GLOBAL)?
3. אופק, מהנדסת מצטיינת ב-ATAM Corporation, החליפה בטעות את הקובץ dynamic.so ב-usr/lib// בקובץ ELF ללא קוד, וניסתה להריץ את הקובץ main.out:
   a. מה התוכנית תדפיס? הסבירו!
   b. האם היא תסיים את ריצתה? הסבירו!