גרסא ג ענו על כל השאלות, **במקום המוקצב לכך בלבד**. חומר עזר: מותר.

בסעיף "אמריקאי" (Multiple choice) סמנו **בצורה ברורה** עיגול סביב האות הנכונה, 4 נקודות לתשובה נכונה, ועל תשובה לא נכונה 1- נקודות. על תשובה חסרה תקבלו 0 נקודות. בכל מקרה הניקוד של שאלה שלמה לא יהיה קטן מ-0.

שאלה 1: תכנות באסמבלר (16%)
סעיף 1

במעבד ממשפחת 68000, רגיסטר A1 מצביע לתחילת רשימה משורשרת (ראו ציור). יש לספור ברגיסטר D0 את אברי הרשימה שערכם שונה מ-0. משמעות הפקודות נתונה בהערות. הקוד שיבצע זאת נכון הוא:

a)
```
      XOR.L     D0, D0     ;XOR D0 with  D0 (.L means 32 bit operand)
 S:   CMPQ.L    #0, A1     ; Compare A1 to 0
      BRE       L          ; Branch (jump) to L if equal
      CMPQ.L    #0, 4(A1)  ; Compare operand at address A1+4 to 0
      BRE       L1         ; Branch (jump) to L1 if equal
      ADDQ.L    #1, D0     ; Add value 1 to D0 (32 bit operand)
L1:   MOVE.L    (A1), A1   ; Move 32 bits from memory at address A1 to reg. A1
      BRA       S          ; Unconditionally branch (jump) to S
L:
```

b)
```
      MOVEQ.L   #0, D0     ;Move value 0 to D0 (.L means32 bit operand)
 S:   CMPQ.L    #0, A1     ; Compare A1 to 0
      BRE       L          ; Branch (jump) to L if equal
      CMPQ.L    D0, 4(A1)  ; Compare operand at address A1+4 to D0
      BRE       L1         ; Branch (jump) to L1 if equal
      ADDQ.L    #1, D0     ; Add value 1 to D0
L1:   MOVE.L    (A1), A1   ; Move from memory at address A1 to reg. A1
      BRA       S          ; Unconditionally branch (jump) to S
L:
```

c)
```
      MOVEQ.L   #0, D0     ;Move value 0 to D0 (.L means32 bit operand)
 S:   CMPQ.L    #0, A1     ; Compare A1 to 0
      BRE       L          ; Branch (jump) to L if equal
      ADDQ.L    #1, D0     ; Add value 1 to D0
L1:   MOVE.L    (A1), A1   ; Move from memory at address A1 to reg. A1
      BRA       S          ; Unconditionally branch (jump) to S
L:
```

d)
```
 S:   CMPQ.L    #0, 4(A1)  ; Compare operand at address A1+4 to 0
      BRE       L1         ; Branch (jump) to L1 if equal
      ADDQ.L    #1, D0     ; Add value 1 to D0
L1:   MOVE.L    (A1), A1   ; Move from memory at address A1 to reg. A1
      BRA       S          ; Unconditionally branch (jump) to S
```

סעיף 2:

דרוש לבצע השמה של מספר של 32 bits ממשתנה y למשתנה x על מעבד 80486 (רוחב BUS של 32 BIT). סמנו את כל קטעי הקוד הנכונים. (כאשר תאור הפעולה בשפת C הוא: x=y;)

1)
```
mov     esi, y
mov     edi, x
movsd
```

2)
```
mov     ax, [y+2]
mov     [x+2], ax
mov     ax, [y]
mov     [x], ax
```

3)
```
mov     dword [x], [y]
```

4)
```
mov     ax, [y]
mov     [x], ax
mov     ax, [y+2]
mov     [x+2], ax
```

סעיף 3

נתון קטע הקוד הבא:

```
PUSH DWORD  p
PUSH DWORD  [c]
CALL  blah
MOV   EBX, [p]
MOV  EBX, [EBX]
MOV   [EBX], EAX
ADD   ESP, 8
```

מימוש של איזו משורות C הבאות מתאים קטע זה להיות?

1) p = blah(*c, &p);
2) **p = blah(c, &p);
3) p = blah(c, &p);
4) *p = blah(p, &c);
5) **p = blah(*c, p);

סעיף 4:

ב-EAX נמצא הערך 9. יש לרשום 4 פקודות שונות בלבד שכל אחת מהן יגרמו לכך שברגיסטר EAX יהיה הערך 0xFFFFFFF6, ללא שימוש ב-MOV.

שאלה 2: מהירות ביצועים וקודים (16%)

סעיף 1:
נתון צופן תיקון השגיאות הבא:

DATA WORD	CODE WORD
00	00000
01	00111
10	11001
11	11110

בעזרת קוד זה ניתן תמיד:
א) לגלות שתי שגיאות או לתקן שגיאה אחת.
ב) לתקן שתי שגיאות.
ג) לגלות שגיאה אחת, אבל לא מאפשר תיקון שגיאות.
ד) לא מאפשר תיקון שגיאות או מחיקות וגם לא מבטיח כלל גלוי שגיאה.

סעיף 2:
סטודנט טען שהוא יכול, תוך שמירה על כך שיהיו 4 מילות קוד בינאריות באורך 5, לשנות את הקוד של השאלה הקודמת כך שיגלה תמיד שלוש שגיאות.
א) הסטודנט טעה, כי לא ניתן למצוא קוד כזה בתנאים הנתונים בכלל.
ב) ניתן לביצוע בדיוק כפי שטען הסטודנט.
ג) ניתן לבצוע אם נוסיף סיבית שישית שהיא PARITY של שתי הסיביות של ה-DATA WORD.
ד) בכלל לא צריך לתקן את הקוד, כי הוא כבר מבטיח גילוי של שלוש שגיאות.

סעיף 3:
במחשב 80486, מספר הגישות לזיכרון הנדרש לביצוע הפקודה הבאה הוא (קידוד ה-OPCODE וה-ADDRESSING MODE דורש 2 בתים):

```
xor     dword  [0x3001a80], 0x12345
```

במקרה הגרוע ביותר (כולל FETCH) הוא:
א) 8 ב) 6 ג) 5 ד) 4

סעיף 4:
תוכנית P רצה ב-9 שניות על מעבד שתדירותו 2GHZ. התוכנית דורשת בצוע 4 מיליארד פקודות, חצי מהן פקודות של נקודה צפה. בהנחה שעשו אופטימיזציה לפקודות של נקודה צפה והקטינו את ה CPI מ-5 ל-3, ואת שאר הפקודות הצליחו לשפר מ- CPI של 4 ל-2 בלבד, בכמה שניות תרוץ התוכנית לאחר השיפור?

א) 8 ב) 7 ג) 6 ד) 5

שאלה 3: נושאי SYSTEM, קלט-פלט, אסמבלר, ותהליך יצירת קובץ ההרצה. (20%)
נתון קטע הקוד הבא, בצורה של LISTING של NASM. הסעיפים הבאים מתייחסים לקוד זה.

```
 1                                  global _start
 2                                  section .data
 3 00000000   03000000                x: dd 3
 4
 5 00000004   8B0D[00000000]    _start: mov   ecx, [x]
 6 0000000A   000D[16000000]         r: add   byte [l+6], cl
 7 00000010   C605[00000000]30       l: mov   byte [x],  48
 8 00000017   51                        push  ecx
 9 00000018   B804000000                mov   eax, 4      ;  For "write" system call
10 0000001D   BB01000000                mov   ebx, 1      ;  to standard output
11 00000022   B9[00000000]              mov   ecx, x      ;  "buffer"
12 00000027   BA01000000                mov   edx, 1      ;  byte count
13 0000002C   CD80                      int   0x80
14 0000002E   59                        pop   ecx
15 0000002F   E209                      loop  r, ecx      ; Decrement ecx,
                                                            jump relative if not zero
16 00000031   BB00000000                mov   ebx, 0
17 00000036   B801000000                mov   eax, 1      ; For "exit" system call
18 0000003B   CD80                      int   0x80
```

סעיף 1: מה תעשה התוכנית הזאת?
א) תדפיס את מחרוזת התווים 484848 ל-STDOUT
ב) תדפיס את מחרוזת התווים 356 ל-STDOUT
ג) תדפיס את מחרוזת התווים 000 ל-STDOUT
ד) תדפיס את מחרוזת התווים 333 ל-STDOUT
ה) לא תדפיס כלום ל- STDOUT

סעיף 2: משמעות הקוד המתורגם בשורה 5 הוא:
א) 8B0D מקודד את הפעולה MOV ואת שיטות המען, וה-00000000 את הערך שיועבר ל-ECX.
ב) 8B0D מקודד את הפעולה MOV ואת שיטות המען, וה-00000000 את הכתובת בזכרון של הערך שיועבר ל-ECX.
ג) 8B0D מקודד את הפעולה MOV ואת שיטות המען, וה-00000000 את הכתובת בזכרון של הערך שיועבר ל-ECX, אבל הסוגריים המרובעים סביב 00000000 מראים שכתובת זאת לא סופית ותתוקן בשלב ה-RELOCATION של ה-LINKER.
ד) 8B0D מקודד את הפעולה MOV ואת שיטות המען, וה-00000000 את הכתובת בזכרון של הערך שיועבר ל-ECX, והסוגרים המרובעים סביב 00000000 מראים ששיטת המען כאן היא ABSOLUTE ולא IMMEDIATE.

סעיף 3: הגרסה של NASM שהפיקה את ה-LISTING נכתבה כפרויקט גמר. אמנם היא מייצרת קובץ OBJECT נכון, אך יתכן שיש בה עדיין בגים בהפקת ה-LISTING. במקרה דלעיל:

א) תרגום הפקודה בשורה 9 שגוי, כי ביקשנו להעביר את המספר 4 ובתרגום כתוב 04000000

ב) שורה 5 שגויה, כי לא השאירו מקום בקוד לסימבול start_

ג) התרגום בשורה 15 שגוי, ה- BYTE השני של הוראת ה-LOOP שבשורה זו צריך להיות D9 ולא 09

ד) אף אחד המקרים המצוינים לעיל אינו שגיאה ב-LISTING.

סעיף 4: מתוכנית readelf שהופעלה על קובץ ה- EXECUTABLE של התוכנית, קיבלנו:

Entry point address: 0x8049058

כתובתו של משתנה x בזמן ריצת תוכנית זו הוא:

א) 0x08049058

ב) 0

ג) 0x08049054

ד) 0x03000000

סעיף 5: האם הקוד שב- LISTING הוא STRICTLY POSITION INDEPENDENT?

א) כן, לחלוטין.

ב) לא, בגלל הפקודה LOOP בשורה 15.

ג) לא, בגלל התייחסות ל"משתנה" x.

ד) לא, בגלל השימוש ב-LABEL של start_

ה) לא, מכל הסיבות ב', ג', ד'.

שאלה 4: נושאי SYSTEM, קלט-פלט, ותהליך יצירת קובץ ההרצה. (16%)

סעיף 1: במערכת ההפעלה LINUX, מועברת הבקרה למערכת ההפעלה:

א) בעקבות פסיקה 0x03 (DEBUG)

ב) בעקבות פסיקת segmentation fault

ג) בעקבות פסיקה חיצונית (למשל מ-HARD DISK)

ד) בכל הפסיקות המצוינות למעלה.

ה) בכל הפסיקות שלמעלה חוץ מפסיקת DEBUG: במקרה זה מועברת הבקרה חזרה ישירות ל-DEBUGGER ללא מעבר בקוד של מערכת ההפעלה.

סעיף2: ידוע כי cat משרשר מספר קבצים ומדפיס אותם ל stdout. כמו כן, /bin/csh הוא SHELL.
מה יודפס אחרי ביצוע השורות הבאות?

```
cat /bin/csh   /bin/ls > aaa
chmod u+x aaa
./aaa
```

א) segmentation fault

ב) רשימת הקבצים ב-directory הנוכחי.

ג) רשימת הקבצים ב-directory של /bin/ls.

ד) יודפס PROMPT והתוכנית aaa תחכה לקלט. היא פועלת כ-COMMAND INTERPRETER, אחרי יציאה מתוכנית זאת, לא תודפס רשימת קבצים.

סעיף 3:

במחשב מבוסס על 80X86 ישנו UART שבו רגיסטר STATUS שבו הסיבית MSB היא 1 אם הגיע נתון בתקשורת הטורית. הכתובת של ה-DEVICE הוא PORT מספר 0x0090, כאשר ה- RECEIVE BUFFER וגם ה-TRANSMIT BUFFER בכתובת 2 של ה-DEVICE ורגיסטר ה-STATUS בכתובת 3 של ה-DEVICE. דרושה פונקציה echo שבודקת אם הגיע נתון בתקשורת הטורית, ואם כן היא פולטת אותו אל ה-TRANSMIT BUFFER, ואחרת חוזרת בלי לעשות כלום. כל הרגיסטרים של ה- UART הם של 8 סיביות. סמנו את כל המקרים בהם הקוד באסמבלר שיבצע זאת נכון.

```
1) get: in      al, 0x0093
        and     al, 0x80
        jz      done
        in      al, 0x0092
        out     0x0092, al
done:   ret

2) get:  in     al, 0x03
        or      al, 0x80
        jz      done
        in      al, 0x02
        out     0x02, al
done:   ret

3) get:  in     al, 0x0090
        cmp     al, 2
        jz      done
        in      al, 0x0092
        out     0x0092, al
done:   ret

4) get:  in     al, 0x0093
        xor     al, 0x80
        jns     done
        in      al, 0x0092
        out     0x0092, al
done:   ret
```

סעיף 4: נתון הקובץ hello.c הבא:

```
#include<stdio.h>
int main( ) {     printf("Hello world\n"); }
```

עתה מבצעים את סדרת שורות הפקודה:

```
gcc –o blah hello.c
./blah > 1
chmod  733 1
./blah > 1
```

התוצאה של שורת הפקודה האחרונה היא:

א) יודפס Hello world למסך.

ב) יודפס Hello world לקובץ בשם 1.

ג) תתקבל שגיאה: permission denied על קובץ 1.

ד) תתקבל שגיאה: segmentation fault

שאלה 5: תכנות ב-C וקריאות מערכת (16%)

סעיף 1: מה יודפס בעת ביצוע הקוד הבא?

```
char * str="0124";
printf("%x", str[str[3]-str[2]+str[4]] );
```

א) 1
ב) 2
ג) 0x31
ד) 32
ה) Segmentation fault או ערך "זבל" אקראי.

סעיף 2: במעבדה 9 השתמשנו בפונקציה mmap למפות את הזיכרון של תוכנית נטענת לזיכרון של התהליך הנוכחי וזאת כדי להעביר את השליטה לתוכנית הנטענת ולאפשר לה לרוץ. באילו דגלים עלינו להשתמש ע"מ למפות את קטעי התוכנית הנטענת? יש לבחור את הצירוף המתאים ביותר להשלים את שורת הקוד הבאה:

```
void *map = mmap(vaddr, length, flags, _____, fd, offset);
```

הערך המתאים הוא:

א) MAP_SHARED | MAP_EXECUTABLE
ב) MAP_PRIVATE | MAP_FIXED
ג) MAP_ PRIVATE | MAP_ SHARED
ד) 0

הערה: ניתן להניח כי שאר המשתנים מאותחלים בצורה נכונה:
vaddr – היא הכתובת הנכונה למיפוי הסגמנט. length – הוא אורך הסגמנט כולל הריפוד הנדרש.
flags – הם הדגלים המתאימים ע"פ הגדרות הסגמנט. fd – הוא מספר הקובץ הנטען בטבלת הקבצים הפתוחים. offset – המיקום בקובץ בו שוכן הסגמנט אותו יש למפות.

סעיף 3: נתונה התוכנית הבאה:

```
main( ) {  if(fork( ) +  fork( ) ) printf("Bingo!\n");   }
```

הפלט שיופיע על המסך יהיה:

א) Bingo!
ב) Bingo!
Bingo!
ג) 3 פעמים Bingo!
ד) 4 פעמים Bingo!

סעיף 4: מה תדפיס התוכנית הבאה, אם ניתן לה את הקלט 01234 ?

```
main( ) {
        char *buf;
        gets(buf);       printf("Got  %x\n", *(unsigned int *)buf);
}
```

1. Got 01234
2. Got 33323130
3. תמיד Segmentation fault
4. לפעמים Got 33323130, ולפעמים Segmentation fault
5. אחר (נא לכתוב בשורה זו):

שאלה 6: מבנה קבצי ELF (16%)

בכל סעיפי שאלה זו יש להתיחס לקובץ ELF לפי HEXEDIT בדף המצורף. טבלת הסימבולים מתחילה ב-OFFSET של 0x01E0, וטבלת שמות ה-sections (ה-shstrtab.) מתחילה ב-0x0094 בקובץ, ה- ENTRY POINT הוא: 0x8048067, וה- SECTION HEADER TABLE מתחיל ב-OFFSET של 200.

סעיף 1: איזו מהתשובות הבאות מכילות רק שמות סימבולים שאינם SECTIONS מקובץ זה?

א) Assad Kaput Iran
ב) Mursi Erdogan! Hizballa
ג) Qusayr Assad Erdogan!
ד) ELF Qusayr

-

סעיף 2: באיזה section נמצאת מחרוזת Qusayr?

א) .text
ב) Mursi
ג) Erdogan!
ד) Kaput

סעיף 3: מה ערכו ההתחלתי של המשתנה Kaput? (לא ערך הסימבול!)

א) 0
ב) 0x8048094
ג) 0x1f0
ד) 0xffffffff, כלומר 1-

סעיף 4: מה צריך להיות כתוב במקום ה- XX XX XX XX בקובץ?

א) 08 04 80 67
ב) 00 02 00 00
ג) 08 04 80 94
ד) 00 00 00 94

בהצלחה!