

13.8.98

2/3

ארגון המחשב וספרות סף
מכחן סיום - מועד ב'
הצהה: ד"ר איתן רון
בדרגת: מאירה שמעון

הנחיות

- 1) מותר כל חומר עזר מניר (בכתביד ומודפס). אין להשתמש במחשבים ניידים (Notebooks).
- 2) זמן הבחינה - 3 שעות.
- 3) כל הרוטיניות שעלייך לכתוב חייבות להיות באסמבלאר.
- 4) מותר לצרף דפים לטופס הבחינה אם יש צורך בכך.
- 5) שים לב שהניקוד על השאלות 4 - 1 הוא שווה.
- 6) למרות שהניקוד של השאלות עולה על 100, הצירנו המרבי שנייתן לקבל בבחינה הוא 100.

שאלות	נקודות
/ 25	1
/ 25	2
/ 25	3
/ 25	4
/ 7	5
סה"כ /107	



כתב קוד הממש "מונח שכנות" עבור תוכנית אפליקציה.

כלומר, עלייך לכתוב קוד היוצר מנגנון אשר, בביי גשר למה שמצבע המשך התוכנית, כל הזמן יופיע בaczha השמאלי העליון זמן שחלק מתחלת הריצה של התוכנית בשניות.

אסור המנגנון שלא ישב או ישנה משאו לתוכנית הראשית (למעט ניצול קטן מהמשך הדפסת המונח עצמו).

הנח שהקוד הבא קובע את מיקום הסמן (Cursor) לaczha השמאלי של המסך:

```
; Set Cursor pos to (0,0)
MOV AH,2
MOV DX,0
INT 10h
```

הנח שהקוד הבא מוחזיר את מיקום הנוכחי של הסמן (Cursor) בתוך DX:

```
; Get Cursor Position into DX
MOV AH,3
INT 10h
```

לדוגמא, פلت אפשרי של התוכנית הבאה:

```
;
Main:
    MOV AX,@DATA           ; Standard Program prefix
    MOV DS,AX               ;
    .....
    MOV DX,OFFSET String
    MOV AH,9
    INT 21h

    ; Wait for key
    MOV AH,0
    INT 16h
    ; Print the key
    MOV DL,AL
    MOV AH,2
    INT 21h
;
; Return to DOS
Done:
    .....
    MOV AH,4Ch
    INT 21h
    END Main
```

יהיה:

Clock: 0028 secs



שאלה מספר 2 (25 נקודות)

2/5

בשפת C גיימת אפשרות להעביר לרוטינה כפרמטר כתובת `צרכו` של רוטינה אחרת. עזרת הכתיבה זו, הרוטינה שמקבלת את הפרמטר יכולה לגרoa לרוטינה שהכתובה שלה הוא הפרמטר. האמצעי זה קרא "פונקציה" ומאפשרת לכתוב רוטינה שגוראות לרוטינות שונות בקריאות שונות בהתאם לערכים שונים של הפרמטר.

לדוגמא, להלן הגדרות של רוטינת C בשם `:vectorize`

```
int vectorize(int (*op)(), int avec[], int range)
{
    int i, tempr;

    tempr = avec[0];
    for (i = 1; i < range; ++i)
        tempr = (*op)(tempr, avec[i]);

    return tempr;
}
```

הרוטינה `vectorize` "מכילה" פונקציה C הפעלת על שני אופרנדים לפעולה על מערך (וקטור) של מספרים.

למשל אם נתונים לנו שתי פונקציות C:

```
int sum(int x1, int x2)
{
    return(x1 + x2);
}

int max(int x1, int x2)
{
    return( (x1 > x2) ? x1 : x2 );
}
```

או

```
int vector[8];
.....
result = vectorize(sum, vector, 8);
result = vectorize(max, vector, 8);
```

يحسب את הסכום של שמונה איברי `vector` לתוך `result`
 יحسب את המქסימום של שמונה איברי `vector` לתוך `result`
 א. כתוב באSEMBLER את הקוד של `vectorize`. הנה שהגדרה של `vectorize`.

```
extern int vectorize(int (*op)(int,int),
                     int avec[], int range);
```

ב. כתוב מחדש את **vectorize** כאשר עכשו (בכל שלושת הרווחיות) מדבר במספרים **int long int** במקום **int**. ההגדרה של **vectorize** עכשו הוא

```
extern long int vectorize(long int (*op)(long, long),
                           long int avec[], int range);
```

ג. כתוב מחדש את **vectorize** כאשר עכשו (בכל שלושת הרווחיות) מדבר במספרים **double** במקום **int**. ההגדרה של **vectorize** עכשו הוא

```
extern double vectorize(double (*op)(double, double),
                        double avec[], int range);
```



שאלה מספר 3 (25) נגידות

א. עליך לכתוב רוטינית אסמלר הנקרהית מ-c

```
extern unsigned long int  
    int mul(unsigned int x, unsigned int y);
```

24

הממש כפל מספרים שלמים אי שימוש בפקודה `TUL` - למחשה לדוגמה:

גלאייד להשתמש בגיisha שלמדת בבית הספר הייסודי;

110010	= 50	דצ'מל
x	x	דצ'מל
1011	= 11	דצ'מל
<hr/>		
110010		
+ 110010		
+ 000000		
+ 110010		
<hr/>		
1000100110	= 550	

כלומר שניתו להסתכל על כפל של שני מספרים שלמים באורך 16 ביט בתור סכום של 16 מספרים מודulosים.

ב. כיצד ישנה תשובה ב-א. אם מדובר בהדמיה של פגודת המכונה **MUL** ולא **?MUL**

כתב רוטינה

```
extern long int int imul(int x, int y);
```

לדוגמא, פلت אפשרי של תוכנית הבאה:

```
void main(void)
{
    int u,v;
    long int res;

    puts("\nEnter two unsigned integers:");
    scanf("%d %d", &u, &v);
    res = int_imul( u, v);
    printf("%d * %d = %ld\n", u,v, res);

} /* main */
```

ר' הירח:

```
E:\>test_imul  
Enter two unsigned integers:
```

נקודות הסתעפות מותנית הם למעשה תנאי מורכב על הדגמים.

למשל `label JG` הוא למעשה "הסתעף ל-label בתנאי $S=0$ וגם $OF = SF$ ".

א. עליך לכתוב מקרו בשם `J_CFI_SF_OF` שמשה הסטעפות מותנית שאין לו פקודת מכונה:

לכנתיבת השורה `J_CFI_SF_OF label` יהיה המשמעות:
"הסתעף ל-label בתנאי $S=1$ וגם $CF = OF$."

יותר לכך להשתמש באוגרים כלשהם לכל מטרה. עליך לדאוג לכך שאת המקרו יאפשר לקרו אמצע לא מוגבל של פעמים.

ב. כתוב גטע קוד הבזק את המקרו ב-א. עבור המקרה $SF=OF=1, CF=0$.

CF הוא בית 0, SF הוא בית 7, ו- OF הוא בית 11 באוגר הדגמים.



שאלה מספר 5 (7 נקודות)

219

תחות Protected Mode, לא ניתן ע"י הפקודה CALL על כל גירסאות להסתעף
מבודד מרמת פריווילגיה אחת לרמת פריווילגיה אחרת.

הראה שאילו ניתן היה לעשות דבר זהה, אזי בעזרת אחת הגירסאות של פקודת
CALL, תוכנית ברמת פריווילגיה הנמוכה ביותר (3) יכולה לגרום לרוטינה
שהה לרווח ברמת פריווילגיה גבוהה (0). באופן ספציפי יותר, הנח שלתוכנית
גירימת רוטינה Anything

Anything PROC

....

Anything ENDP

הראה כיצד התוכנית יכולה לגרום Anything לרווח Anything ברמת פריווילגיה
עליה.



ארגון המחשב ושפט ספ (203.1130)
סמסטר ב' תשנ"ט
בחינה סופית - מועד א'

הוראות לנבחן:

- משך הבחינה שלוש שעות.
- מותר להשתמש בכל חומר עוזר, למעט מחשבים מכל סוג.
- יש להסביר על כל השאלות.
- יש לרשום את התשובות בגוף השאלון במקומות המיועדים לכך.
- נא לכתוב בכתב יד ברור ונקי. מומלץ להשתמש בעפרון ומחק.
- בשאלון זה 12 דפים, כולל דף זה. ודא כי כל הדפים נמצאים.

ב הצלחה!

ציוון	ניקוד	
19	25	שאלה 1
25	25	שאלה 2
24	25	שאלה 3
23	25	שאלה 4
91	100	סה"כ



19
23

3

10

שאלה מס' 1 (המשך)

- ג. נ. תרגם את המספר העשווני 76- לבסיס בינרי בشرط המשלweis ל-2, ברוחב של 8 ביטים.

	64	32	16	1	4	2	1
1	0	1	1	0	1	0	0

- חשב את סכוםם של שני המספרים שהלו בחלוקת המשלים ל-2. המספרים הינם בסיס הקשה-דצימלי וברוחב של 16 ביטים.

		1	
6	E	3	4
B	0	D	1
1	F	0	5

האם יש לנו מון הביט השמאלי ביותר בפועל חיבור זו?

האם פועלת חיבור זו גורמת לגלישיה? מ-
אליהו - עזרא - יונה נסויים

ד. להלן מספר פקודות אשר גורמות לשגיאה בזמן אSEMBLY.
עבור כל פקודה, הסבר בקצרה מהי השגיאה.

int 256 .i

If a segment register is one operand to MOV,
the other has to be a memory location or
constant as []^{ds} or 1000, ds .
~~or~~ mov 1000, ds .ii
or general register.

bx [ax] add bx, [ax] .iii

• جنگل های ایرانی

(,) 60) Words 6 1,178 My Way push ah .iv

- נבור כל אחד משלשת קטני הקוד שלהלו, רשום את תוכנו של האוגר α ביצוע הקטנו.

```
1          mov dx,0 .ii)
2          mov ax,-1 ;
3          c: inc dx .
4          shr ax,2 .
5          jb c    if cf=1
```

```
        mov      dx,0   0  .ii  
        mov      cx,9  
b: inc      dx      1  
        cmp      dx,cx  1,9  
        loopnz b
```

```
mov dx,0          .i  
mov ax,7ffffh    (32768)  
a: inc dx        j y + 1      (-32768)  
add ax,2        ay = 32769 =  
jg a            f = 1
```

$$dx = \frac{5}{111111111}$$

$$dx = \frac{5}{\underline{\hspace{2cm}}}$$

$$dx = \underline{16313} \times$$

שאלה מס' 2 (קודות)

השיגרה `_findMax` שמוגדרת להלן מקבלת שני פרמטרים במחסנית: כתוות של מערך של מילים, ומספר המילים במערך. השיגרה מוחפשת את המילה הגדולה ביותר במערך, כאשר ייצוג המספרים הוא בשיטת המשלים ל-2, ומוחזירה את ערכיה של מילה זו באונר `ax`.

223

```

1  _findMax proc    near
2      push    bp      → start of
3      mov     bp, sp   → bp - points at old bp, bp+2 at old ip
4      push    bx      → bp-2 } local variables
5      push    cx      → bp-4 }
6      mov     bx, [bp+4] → bp+4 - points at start of array
7      mov     ax, [bx]  → the "max" word (initialized with the first word)
8      mov     cx, [bp+6] → bp+6 - points at the number of words
9      dec    cx      → decrement words counter
10     jcxz   endFind → if there was only one word - exit - it has the "max"
11     select: add    bx, 2   → get + the next word
12     cmp    ax, [bx]  → compare the "max" with: "MAX"
13     jge    next   → if the "max" is greater or equal loop back 2 cmd
14     mov    ax, [bx]  → set the "max" word to be the "next" word (the one we just read)
15     next:  loop   select → loop back 4 cmds.
16     endFind: pop   cx      } restore all our friends
17           pop    bx
18           pop    bp
19           ret
20     endp

```

→ asta la rista

א. להלן הגדרת המודול וSEGMENT הנתונים של התוכנית.

```

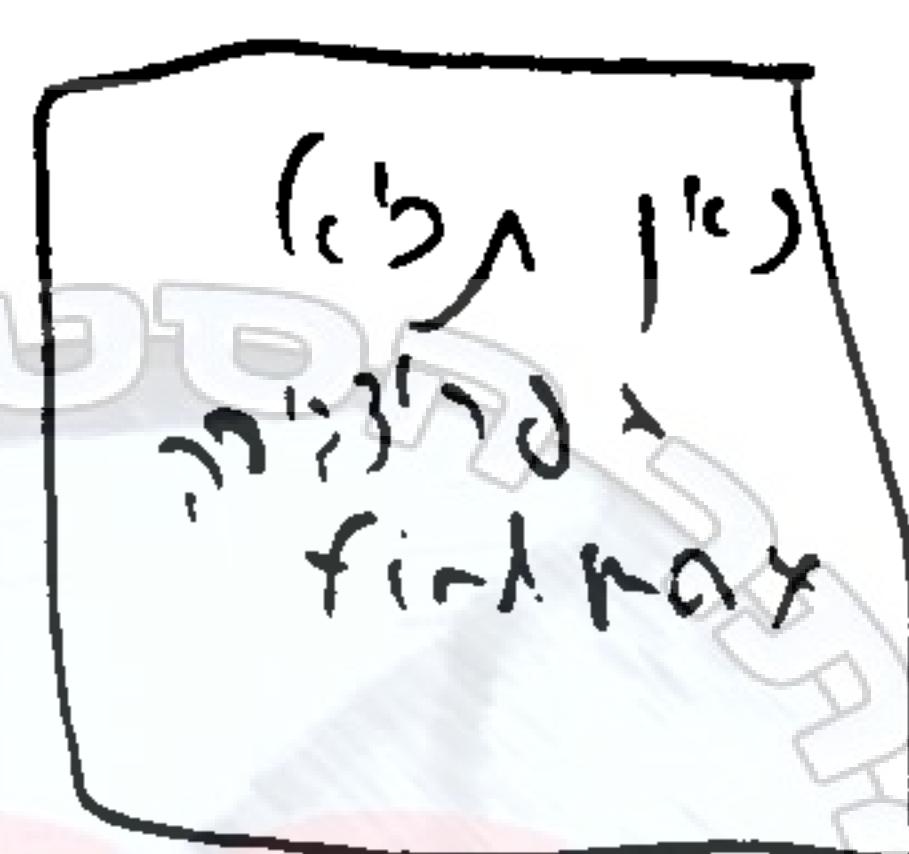
.model use16 small
.data
numbers dw 4,-3,-5,4,9,-8,2,6
len    dw 8
result dw ?

```

כתב קטע קוד בשפת אסמבלי שקורא לשיגרה `_findMax` נס המערך `numbers` והאורך `len`, ומציב את הערך המוחזר מן השיגרה בתוך המשתנה `result`.
הקפץ על כללי תיכנות נכוון של קרייה לשיגרה.

מהו תוכנו של המשתנה `result` בסיום קטע הקוד?

.code



Main:

```

        mov ax, @data    ; sets ds, to start of data
        mov ds, ax
        push len          ; free stack space from parameters
        push offset numbers
        call _findMax    ; free stack space from parameters
        add sp, 4          ; save result
        mov result, ax    ; free stack space from parameters
        mov ax, 4[esp]    ; free stack space from parameters
        int 21h

```

המשר שאלת מס' 2 בדף הבא

2.00N

שאלה מס' 2 (המשר)

ב. רוצים לשנות את השיגורה `findMax` כך שתתקבל מערך של בתיים במקום מילימ', ותחזיר את התוצאה באורך לו. רשום את מספרי השורות בשיגורה שיש לשנות, ואת קוד האסמליה החדש של שורות אלה. יש לבצע אך וرك שינויים הכרחיים.

224

<u>מס' שורה</u> 7 8/ 9/ 10/ 11/ 12/ 13/ 14/ 15/	<u>קוד חדש</u> $xor ax, ax$ $mov al, byte ptr [bx]$ $inc bx$ $cmp al, byte ptr [bx]$ $mov al, byte ptr [bx]$ ret
--	--

ג. רוצים שהשיגורה `findMax` תהיה מוגדרת far במקום near. רשום את מספרי השורות בשיגורה שיש לשנות, ואת קוד האסמליה החדש של שורות אלה. יש לבצע אך וرك שינויים הכרחיים.

<u>מס' שורה</u> 1 6 9	<u>קוד חדש</u> $-findmax proc far$ $mov bx, [bp+6]$ $mov cx, [bp+8]$
--------------------------------	---

ד. רוצים לשנות את השיגורה `findMax` כך שתחפש את המילה הגדולה ביותר במערך, כאשר יוצג המספרים הוא ללא סימן. רשום את מספרי השורות בשיגורה שיש לשנות, ואת קוד האסמליה החדש של שורות אלה. יש לבצע אך וرك שינויים הכרחיים.

<u>מס' שורה</u> 13	<u>קוד חדש</u> $jal next$
-----------------------	------------------------------

ה. מה יש להוסיף לקובץ המקור של השיגורה `findMax` כדי שאפשר יהיה לקרוא לשיגורה מתוך תוכנית בשפת C ?
 רשום את הכתובת (header) של השיגורה בשפת C, כפי שתופיע בתוכנית הקוראת.
 כ-2.1.4.4.4:

✓

```

Code
public - findMax
{
    int max;
}
  
```

#include <iostream.h>

extern int FindMax (int +, int +);

: C ->



שאלת מס' 2 (המשר)

השлага `findBest` (ראה להלן) הינה הכללה של השлага `findMax`. בשירה `findBest`, המאפיין של הערך שמחפשים בערך קבוע, אלא מונדר כIFORMTER נוסף במחסנית. פרמטר זה הוא מצבייע לשлага, אשר תפקידה לבחור במילה ה"טובה יותר" מבין זוג מילים, לפי מאפיין כלשהו (למשל, המילה הגדולה יותר מבין הפתאים, הקטנה יותר, וכו'). "שיגרת הבחירה" מחליפה את הבחירה הקבועה (של המילה הגדולה יותר) שמתבצעת ב-`findMax` (שורות 12-14).

שיגרת הבחירה מקבלת כIFORMTER זוג מילים: באוגר `ax` ובאוגר `dx`, ומחזירה באוגר `ax` את המילה ה"טובה יותר" מבין הפתאים. האוגר `dx` נשאר ללא שינוי.

דוגמה, השירה `better1` (ראה להלן) בוחרת במילה הגדולה יותר, כאשר ייצוג המספרים הוא בשיטת המשלים ל-2. השירה `findBest` עם הפרמטר `better1` מימוש לפיך מקרה פרטי זהה לשירה `findMax`.

להלן הגדרת השירה `findBest`. השורות המוטנות בכוכבית הן אלה שנוספו או השתנו ביחס לשירה `findBest`. כמו כן, מוגדרות שתי שגרות בחירה: `better1`-`better2`, ודוגמה לקריאה לשירה `findMax`.

<pre> findBest proc push bp mov bp,sp push dx ;* push bx push cx mov bx,[bp+4] mov ax,[bx] mov cx,[bp+6] dec cx jcxz endFind add bx,2 mov dx,[bx] ;* call [bp+8] ;* next: loop find endFind: pop cx pop bx pop dx ;* pop bp ret endp </pre>	<pre> better1 proc cmp ax,dx jge end1 mov ax,dx end1: ret endp better2 proc cmp ax,dx jnb end2 mov ax,dx end2: ret endp main: proc mov ax,@data mov ds,ax push offset better1 push len push offset numbers call findBest add sp,6 mov result,ax end main </pre>
---	--

1. מהו החיפוש שמבצע השירה `findBest` עם הפרמטר `better2` ? בדיקה בתייאור המאפיין של שיגרת הבחירה.

לעומת `jnb` (`jne`) או `jne` (`jnb`) מושג unsigned (`unsigned`) או `signed` (`signed`) מושג `signed` (`unsigned`)



2. נתון סמנט הנתונים מסעיף א'. מהו הערך שתחזיר השירה `findBest` בקריאה של להלן?

```

push offset better2
push len
push offset numbers
call findBest
add sp,6

```



$$AX = -3 \equiv 65533$$

שאלה מס' 2 (המשר)

ג. כתוב שירה בסיס better3, שמקבלת כפרמטרים זוג מילים: באורך ax ובאורך dx, ומחזירה באורך ax את המילה שערכה המוחלט קטן יותר, כאשר ייצוג המספרים בשיטת המשלים ל-2. הקפץ על כללי תיכנות נוכניות של שירה. רמז: להיפוך הסימן ניתן להשתמש בפקודה neg.

226

proc better3

? loc1 push ax
push dx

test ax, 10000000-0000 0000 b

jz checkdx :

neg ax ; ax was relative

checkdx : test dx, 1000 0000 0000 0000 b

jz compare :

neg dx ; dx was negative

compare : cmp ax, dx ; compare

jbe dontchange :

pop ax ; ax was smaller so ax=dx

pop dx ; get rid of original from stack

jmp short end3

dontchange : pop dx ; restore ax & dx as they were

end3 : ret ; frags all, folks

endp

~~push dx~~

~~push ax~~

~~pop dx~~

~~pop ax~~

~~push dx~~

~~push ax~~

~~pop dx~~

~~pop ax~~

ד. נתון סרメント הנתונים מסעיף א'. מהו הערך שתחזיר השירה findBest בקריאה שלהלו?

```

push offset better3
push len
push offset numbers
call findBest
add sp, 6

```

$$ax = 2$$



שאלה מס' 3 (25 נקודות)

א. ביצוע הפקודה `mov` אינה אפשרית העתקה בין שני אופרנדים בזיכרון. הגדר מacro בשם `move` מרחיב את הפקודה `mov`, ומאפשר העתקה גם בין שני אופרנדים בזיכרון.

לדוגמא, נניח כי `var1`-`var2` הינם מילים בסגמנט הנתוניים. אז הקריאה `move var2,var1` מעתיקה את `var1` `var2` אל .

רמז: השתמש במחסנית.

```
move macro x,y
    push y
    pop x
endm
```

ב. להלן הגדרת המacro `comp`.

```
1 comp macro x,y
2     ifidni <&y>,<ax>
3         cmp x,y
4         exitm
5     endif
6     push ax
7     mov ax,x
8     cmp ax,y
9     pop ax
10    endm
```

לפניך שלוש קראיות למacro `comp`. רשות את קוד האסטטלי שנפרש על ידי כל קראיה. הנח כי `var1`-`var2` הינם מילים בסגמנט הנתוניים.

`comp var1,ax` .iii

`Cmp Var1,ax`

`comp es,bx` .ii

`push ax`
`mov ax,es`
`cmp ax,bx`
`pop ax`

`comp var1,var2` .i

`push ax`
`mov ax,var1`
`cmp ax,Var2`
`pop ax`

ג. 1. הקריאה `comp bx,es` גורמת לשגיאת אסטטלי. הסבר מהי השגיאה.

כיוון שהוא יכול לטעות בזיהוי אסטטלי, אין סיטה לאgregates_segment registers. לעומת זאת, הקריאה `comp es,bx` (ראה סעיף ב' או) אינה גורמת לשגיאת. הסבר מדוונ קיים הבדל בין שתי הקראיות.

ii. הסבר מהי השגיאה שגורמת לקריאה `comp bh,ch`.

opWord sizes do not match

אנו רצוי דגש על פ' וט' וט' וט'

שאלת מס' 3 (המשך)

ד. הסבר בקצרה מה עושה המאקרו **.comp**. מהו מנגנון השימוש במאקרו? (בוחן: ראה סעיף ג').

ה. המקרה `movc` שהגדרתנו נתונה להלו, מעתיק את האופרנד y אל האופרנד x אם וורק אס מתקיים התנאי (y cond x). האופרנד `cond` הוא סימת של פקודת הסתעפות מותנית כלשהי (כגון `l1`, `g`, ו`cld`).

לדוגמה, הקריאה `movc ax,bx,ae` מעתיקת את האונגר `ax` אל האונגר `bx`, כאשר היצוג המספרי הוא ללא סימן (בגלל השימוש בתנאי `(ae)`).

השלם את הגדרת המacro `movc` במקומות המסומנים בקן.

movc	macro	x,y,cond
	local	<u>Movit, Movend</u>
	cmp	x,y
	j & cond	<u>Movit</u>
	jmp	<u>Movend</u>
movit:	mov	x,y
movend:		
		endm

המקורו \max , שהגדרתנו נתונה להלן, מציב באופרנד x את הגדל מבין שני האופרנדים x ו- y , כאשר הייצוג המספרי הוא בשיטת המשלים ל-2. האופרנד y נשאר ללא שינוי. השלם את הגדרת המקורו \max במקום המסומן בקוו.

max macro x,y ←
 movc x,y
 endm

הגדיר את המאקרו $\max3$ (ראה שילץ להלן), שמחזיב באופרנד x את הגדול מבין שלושת האופרנדים x, y, z .

עליך להשתמש בקריאה למקנון max. אין להשתמש ישירות בפקודות `mov` או `cmp`.

max3 macro x,y,z

```
max x,y
      x,z
endm
```

על ידי הכנסת שני שינרים פשוטים בהגדרת המאקרו move נאפשר למאקרו לקבל אופרנדים נוספים. הסבר בקצרה למה הכוונה. נא לא לכתוב קוד.

רְדָבֶר בְּקַצְרָה לִמְהָה הַכּוֹנוֹנָה. נָא לֹא לְכַתּוֹב קוֹד.

שאלה מס' 4 (25 נקודות)

בידוע, חוץ המקלט יכול להכיל 15 תווים בלבד. כל עוד הוכנית שמתבצעת אינה משתמשת בקלט מחוץ למקלט, לא תתאפשר הקלה של יותר מאשר 15 תווים. זה מוגבל, שכן לעיתים נוח להקליד קלט לתכנית זו עסוקה בדברים אחרים. הוכנית שללון מדגימה כיצד אפשר להתגבר על מוגבלת המקום בחוץ למקלט.

נקזה שטח, גדול כרצוננו, באחד הסגמנטים של הוכנית המשתמש, ונכתבת תוספת לשיגרת הטפוף בפסקת המקלט (9 אמ'), אשר תעביר אל השטח שהוקצה כל תו שմגיע מן המקלט. רק כאשר שטח גדול זה מתמלא לגמרי, לא תתאפשר הקלטה נוספת. למנגנון זה יתרון נוסף, והוא שתכנית המשתמש יכולה לגשת ישירות אל התווים שהוקלו, ללא צורך בשירותי מערכת הפעלה.

להדגמת המנגנון, הוכנית הראשית שללון ממתינה עד שהשטח שהוקצה מתמלא לגמרי, ואז מדפיסה בבת אחת את כל הקלט על המסך ומרוקנת את השטח. שיגרת המקלט והתוכנית הראשית מסמנות זו לחן כאשר השטח מתמלא או מתרוקן לגמרי.

<pre> 1 .model use16 small 2 .stack 100h 3 .data 4 db 10,13 5 keybrdBuf db 1000 dup (?) 6 trailer db 10,13,'\$' 7 repeat db 3 8 .code 9 oldInt9 dd ? 10 bufStart dd far ptr keybrdBuf 11 bufLen dw trailer-keybrdBuf 12 bufIndex dw 0 13 isBuffFull db 0 14 newInt9: pushf 15 call oldInt9 16 cmp isBuffFull,1 17 je ret9 18 push es 19 push ax 20 push bx 21 push di 22 mov ah,1 23 int 16h 24 jz exit9 25 mov ah,1 26 int 21h 27 les bx,bufStart 28 mov di,bufIndex 29 mov es:[bx+di],al 30 inc di 31 mov bufIndex,di 32 cmp di,bufLen 33 jl exit9 34 mov bufIndex,0 35 mov isBuffFull,1 36 exit9: pop di 37 pop bx 38 pop ax 39 pop es 40 ret9: iret </pre>	<pre> 41 main: mov di,@data 42 mov ds,di 43 mov ax,3509h 44 int 21h 45 mov word ptr oldInt9,bx 46 mov word ptr oldInt9+2,es 47 push ds 48 mov dx,cs 49 mov ds,dx 50 mov dx,offset newInt9 51 mov ax,2509h 52 int 21h 53 pop ds 54 idle: cmp isBuffFull,0 55 je idle 56 mov dx,offset keybrdBuf-2 57 mov ah,9 58 int 21h 59 dec repeat 60 jz exit 61 mov isBuffFull,0 62 jmp idle 63 exit: lds dx,oldInt9 64 mov ax,2509h 65 int 21h 66 mov ah,4ch 67 int 21h 68 end main </pre>
--	--

המשר שאלה מס' 4 בדף הבא

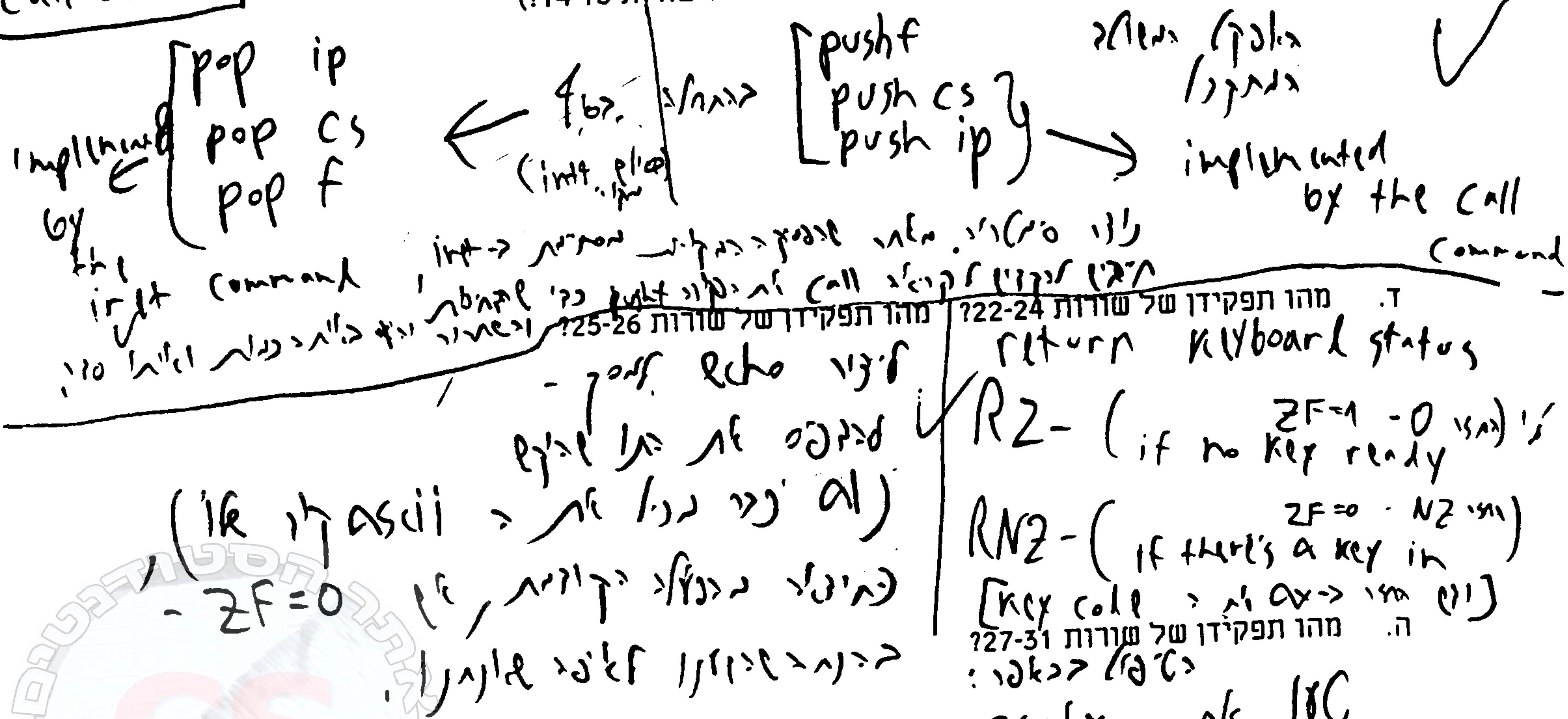
שאליה מס' 4 (המשך)

עליך להסביר את אופו פונולית של התכנית שבדף הקודם, לפי השאלות של halo.
בזה: השאלות נועדו להנחות אחרות בהבנת התמונה הכללית. מומלץ לעבור על כל השאלות מראש.

- ב.** שיגרת השירות החדש של פסיקת המקלט (`newInt9`) מפעילה קודם כל את שיגרת השירות הישנה (הוצבנית על ידי `9Impo`), וורק לאחר מכן מבצעת משימות נוספות. הסבר מדווקע.
(בש: מה עשו שיגרת הפסיקה הישנה?)

interrupt 9: hardware interrupt request 1

- כיצד מתבצע המעבר אל שיגורת הפסיקה היינונה? מה מבטיח שנדיזלן אל שיגורת הפסיקה החדשה? (במ: מהו האפקט המשולב של שורדות 14-15?)



המשר שאלת מס' 4 בדף הבא

Li λ Sir, a yale all, like first fit

<http://cs.haifa.ac.il/students/>

שאלה מס' 4 (המשך)

1. אילו מרכיבים מקבל המשתנה `isBuffFull` במשר מהלן התבנית? איזה חלק של התבנית (SIGRHT המקלדת, או התבנית הראשית) מציב כל אחד מרכיבים אלה במשתנה?

הסביר כיצד הדבר נעשה. (ראו: ראיה סעיף ו').

הוּא נאך - ייְהוָה - 61- זענאי אומרג' גזען
ונאך הגדת נאך ('Na ts' v'Na) אריסטון (Ariston).

מִהוּ תְּפִקִּידָן שֶׁל שׂוֹרְדוֹת 16-17 ? | 55-54 |

butler-> הַבְּתָלֵר (בְּתָלֵר) בְּתָלֵר בְּתָלֵר
buff-> בְּפָטִיר בְּפָטִיר בְּפָטִיר

butter ->  

الله - يسوع - ماريا

וְזַהֲרָה אֶלְעָם
יְהִי כְּבָשָׂר וְלֵב

(סידנא עלי) ר' ירמיה

ט. מה יקרה אם הקלט מון המקלדת מכיל את התו “\$“?

לעתה נזקק לנקוט בפתרון. נזכיר שבנוסף לטבלה שהשיטה שהוקצתה נזקק את תשובתך.

repeat = 3 -
repeat = 2 -
repeat = 1 -
repeat = 0 -

א. **(שאלת רשות)** מזכה עד ל-5 נקודות **בונוס**). מה לדעתך אם נמחק את שורות 22-24 מהתכנית? תן הסבר מילולי קצר.

מהתכוית? תנו הסבר מילולי קצר.

$$\int u' du = 1 \cdot u$$

ארגון המחשב ושפט סב (203.1130)

סמסטר ב', תשנ"ט
בחינה סופית - מועד ב'

הוראות לנבחן

- מישר הבחינה שלוש פעומות.
- מושך להשתמש בכל חומר, למעט מחשבים מכל סוג.
- יש להסביר על כל השאלות.
- יש לרешום את התשובות בגוף השאלון במקומות הסויידים לכך.
- נא לכתוב בכתב יד ברוד ונקה. מומלץ להשתשפּ בעורdon ומקה.
- בשאלון זה 21 דפים, כולל דף זה. הוא כי בכל הדפים נמצאים.

ב הכל!

מספר	שם	ציון
		שאלה 1
25		
		שאלה 2
25		
		שאלה 3
25		
		שאלה 4
25		
		סה"כ
		100



שאלה מס' 4 (המשך)

עליך להסביר את אונז פועלתה של הוכנית שבודק הקוד, לפי השאלות בסעיפים א'-ג', שלහן. במ"ט: השאלות נועדו להנחות אותן בהבנת הטעינה הכללית. מומלץ לעבור על כל השאלות וראש.

א. מהו תפקידו של שורות 61-55?

ב. מה גורם להפעלה קטע הקוד שבשורות 15-8?
מה גודת להפעלה קטע הקוד שבשורות 16-35?

ג. מהו תפקידו של שורות 18-16?

ד. מהו תפקידו של המשטנה *zomer*?
מהי המשמעות כאשר ערכו חיובי? כאשר שרכו 0 או שלילי?

ה. מהו תפקידו של שורות 14-10?

שאלה מס' 4 (המשך)

1. מה שופת הלולאה בשורות 27-25? מתי יצא מLOOP זה?

מה שופת שורה 28 ובאיזה מקרה היא מופעלת?
מה עופת שורה 33 ובאיזה מקרה היא מופעלת?

2. מה יקרה אם נמתק את שורה 25? האם לדעך יש לבן חשיבות? נמק את תשובתך.

ט. (פאלת ושות, מכה עד ל- 5 נקודות בונוס)
כיצד ישתנה השירות אם נחליף את שורה 27 בקטע הקוד שלללו?

```

27.1      jg   checkTime
27.2      mov  ah,1
27.3      int  16h
27.4      jnz  getChar

```

שאלה מס' 4 (המשך)

כתב תוכנית בשפת אסמבלי, אשר קולצת תווים ממסך ומדפיסה אותם על המסך (תוכנית echo). בכל פעם שנעברת דקה ללא הקלדה תרץ וופטן צפוף. התוכנית משתמשת באסף מזקלד התו ESC. השימוש בשלך התוכנית שלילן.

הנחיות:

- חובה להשתמש הגדש של המקלדת (פסיקה ah=0fh, 16h).
- הנח כי השרות החדש כבר טוען בזיכרון.
- אין להגדיר פסיקות חדשות נוספות.
- להדפסת צו על המסך, השימוש בפסיקה ah=2, 21h.
- קוד ה- ascii של ESC הוא 1bh.
- להשמנת צפוף, יש להדפיס על המסך את קוד ה- ascii .7

```
.model small  
bell equ 7  
escape equ 1bh  
.stack 100h
```

```
.code
```

```
exit:    mov ah, 4ch  
         int 21h  
         end
```



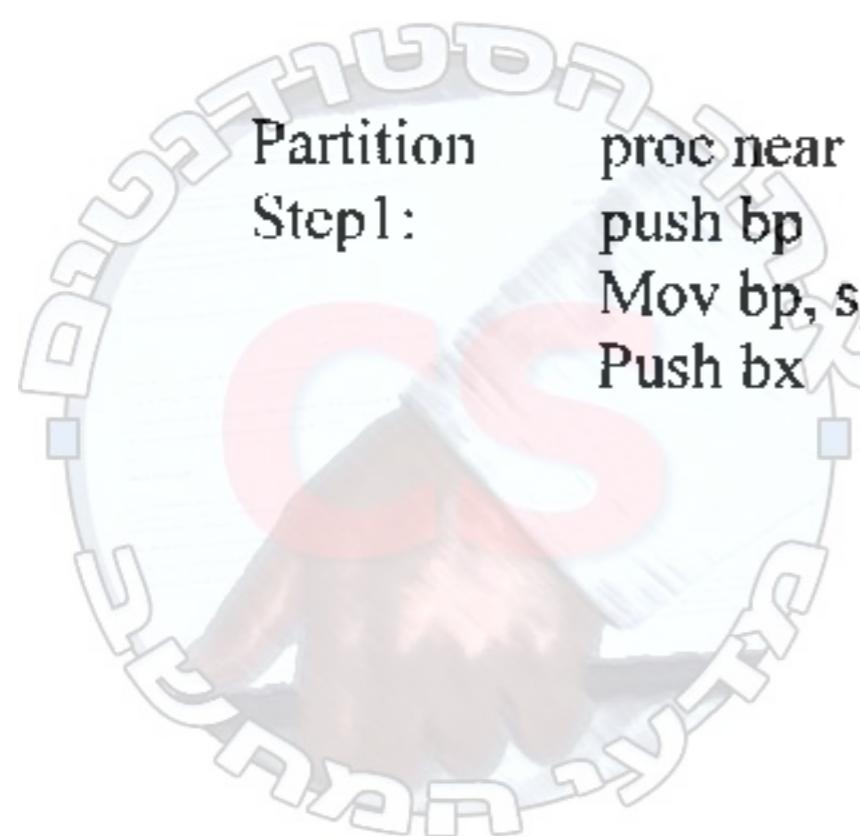
- (1) במהלך התוכנית, `isBuffFull` יהיה אפס עד שיקלטו כל 1000 התווים. בשלב זה, הפסיקה תציב בו אחד. אז התוכנית תדפיס את colum, ואז את `isBuffFull`, ותקלוט 1000 תווים נוספים עד שה`isBuffFull` יהיה אחד שוב. ככלומר הפסיקה מאותה לתוכנית שהబאפר מלא, והתוכנית מאותה שהבאפר התרוקן שנית.
- (2) הפסיקה בודקת אם האינדקס הנוכחי (המקום הריק הבא) שווה לגודל הבאפר, ואם כן מציבה 1 כ-`isBuffFull`. התוכנית מדפיסה ("מראקנת" כביכול) את התווים ואז מציבה 0.
- (3) 16-17 – אם הבאפר מלא, לא יקלטתו לבאפר שלנו, ונחזיר לתוכנית הראשית. 54-55 – לולאה "כמעט אינסופית", עד שיקלטו 1000 תווים מהמקלדת.
- (4) ב-`echo` הכל יודפס, אך בהדפסה שנייה (מהתוכנית הראשית) יודפס עד ה-\$ הראשון בגל השימוש בשירות 9 של פסיקה `h.21`.
- (5) 3 פעמים – עד שהמשנה `repeat` הגיע לאפס.

תשנ"ט מועד ב'

- (1) (א) העברת פרמטר `by Val` – זה פרמטר בגודל ידוע מראש שMOVEDר כמו שהוא במחסנית. משנהו `by Address` – ניתנת לנו כתובתו ולכון גם אופציה לשנותו בסביבה הקוראת. גודלו לא חייב להיות ידוע מראש.
- (ii) `Call far` מתייחס ללייבל עם סגמנט ווגם אופסט, בעוד `near call` מתייחס לסגמנט הנוכחי, ו-`x` הוא אופסט בלבד.
- (iii) מלבדות (פסיקת תוכנה) נקרה ע"י המתכנת בפקודה `int` כאשר `k` מייצג את מספר הפסיקה. חריגה היא פסיקה שמתאפשרת במקרים מיוחדים (חלוקת ב-0, `segmentation fault` וכו').
- (iv) `Sub` משפיעה על הדגלים, `mov` לא.
- (v) בעlion (EQU) `X` קבוע, בעוד `שהתחthon` (`1 = X`) אפשר שינוי במהלך הפרה-פרוססינג.
- (6) (ב) `ex` `-l .Sub cx, 1 .Dec ex` תחילית "rep" מוריידה את ערכו של `cx`.
- (ג) (i) `-86`.
(ii) `BD63`. נשא – כן. גליישה – כן.
(iii) לא, מכיוון שהחישור מס' חיובי יכול להניב תוצאה חיובית או שלילית, ולכון אין אף פעם גליישה (overflow).
- (7) (i) אי אפשר להגדיר בית בגודל 256 – הטווח הוא 0-255.
(ii) לא ניתן לעשות `cs pop` כי לא ניתן לשנות את CS דרך פקודות העברת. הוא מתעדכן בפקודות קפיצה, כגון `mov` ו-`call`.
(iii) אי אפשר להעביר מזוכרון לזכרון.
(iv) Loop לא מקבל לייבלים גדולים מ-128 בכל כיוון, וגם `loop` מקבל לייבלים ולא כמותות.
(h) (i) `dx = 7` (ii) `dx = 12` (iii) `dx` יהיה (בבנייה) 1 ואחריו הכל אפסים. (-32,768)

- (2) (א) כל סעיף מוגדר אחרי הליבל המתאים.

Partition Step1:
 proc near
 push bp
 Mov bp, sp
 Push bx



	Push di	
	Push si	
Step2:	mov bx, [bp+4]	
	Mov di, [bp+6]	
	Mov si, 0	
	Shl di, 1	מערך של מילימ, צריך להכפיל את מספר האיברים ב-2 ;
	Sub di, 2	
Step3:	test [bx+si], 1	
	Jnz, step4	
	Add si, 2	
	Jmp step3	
Step4:	test [bx+di], 1	
	Jz step5	
	Sub di, 2	לקפוץ ב-2 כי אלו מילימ ;
	Jmp step4	
Step5:	cmp si, di	
	Jnb step6	
	Xchg ax, [bx+si]	מחליף בין שני משתני הזכרון. ניתן להשתמש ב - ;
	Xchg ax, [bx+di]	; push, pop
	Xchg ax, [bx+si]	אך ערכו של האוגר ממילא נדרש בהחזרה ;
	Jmp step3	
Step6:	shr si, 1	
	Mov ax, si	
	Pop si	
	Pop di	
	Pop bx	
	Pop bp	
	Ret	
Partition	endp	

Push len
 Push offset list
 Call partition
 Add sp, 4
 Mov count, ax

(ב)

- ג) תוצאה: -5, -1, -4, 0, 2, -7, 9, .count = 6, .list = 6, -4, 0, 2, -7, 9.
 - ד) מניחים כי יש מספר אחד לפחות אי-זוגי.
 - ה) צריך לציין `_partition`, יש לשנות את שם השגרה ל- `_partition`. גשנה את step2 ע"ז הגדלה הקידיות במתנית ב-2 כל אחת: 6, 8 במקומ 4, 6. כותרת השגרה ב-C תהיה:
- ```
extern int partition (int [], int);
```
- ו) טענה זו נכונה – השגרה יכולה תבוצע ב- (n)O כי לכל איבר יבוצע מקסימום 2 בדיקות, אם הוא נמצא במקומות "לא טוב" – כלומר, אם הוא אי-זוגי שיבדק ע"ז SI, או זוגי שיבדק ע"ז DI.
  - ז) תוצאה: -5, -4, -7, 0, 2, 6, 9. .count = 5. List = 1, 6, 9, 0, 2, -7, -4.
  - ח) הפוקודה `Ror` תפעיל כך: אם המספר חיובי הוא מהפוך אותו לו זוגי. אם המספר שלילי הוא מהפוך לא-זוגי. כלומר השגרה תפרק בין החיובים (במקור) לשילולים. `Ror` חזרה את המספרים לקדמותם ולכן בסוף הקטע יהיה המערך מופרד לחובי/שלילי.



(3)

חלק I:

א) הקוד שנפרש מהקראות למקרו:

| isZero1 cax | isZero2 ax                        | isZero3 al                                                                  |
|-------------|-----------------------------------|-----------------------------------------------------------------------------|
| Mov eax, 0  | Cmp ax, 0<br>Je set2<br>Mov ax, 0 | Cmp al, 0<br>Je set3<br>Mov al, 0<br>Jmp exit3<br>Set3: mov al, 1<br>Exit3: |

ב) הבדיקה `ifidni` אינה מתאימה כי המתכנת בודק אם שם הרגיסטר הוא 0 ולא אם תוכנו שווה 0.  
הבדיקה מתבצעת בזמן הפה-פרוסטיג ולא בזמן ריצחה.

ג) ב-`isZero2` יש פקודה לפה-פרוסטור, `exitm`, שמספקת את פרישת קוד המקרו, ככלומר החלק  
השני של המקרו עם הליבל `set2` לא ייפרש לעולם. בכך `je set2` מיצור שגיאת קומpileציה.

ד) הליבלים אינם לוקאליים וכן בכל תוכנית שתקרה למקרו יותר מפעם אחת, נמצא מספר ליבלים  
בעל אותו שם. התוכניות הללו לא יתקמפלו. אם המקרו יקרא רק פעם אחת, התוכנית תתקמפל.

(ה)

```
isZero4 macro reg
 local set, exit
 cmp reg, 0
 je set
 mov reg, 0
 jmp exit
set: mov reg, 1
exit: endm
```

חלק II:

ו) מקבל הודעת שגיאה בזמן אסמבלי על שורה מס' 5, כי פקודת `mov` לא מקבלת שני פרמטרים של  
זכרון. מצד אחד זהו בעיה של המשתמש במקרו כיון שהוא הוגדר לרגיסטרים בלבד, אך מצד שני  
המקרו אינו מטפל בכל סוג הפרמטרים האפשריים וזה בעיה במימושו.

ז) הבעיה שהתגללה בזמן ריצחה: נזחוף משתנה בגודל מילה למורთ שהקצנו רק בית בודד ל-`reg` (ראה  
איור), וכך ידרס בית אחד מהנתונים האחרים, או מ-BP(?)

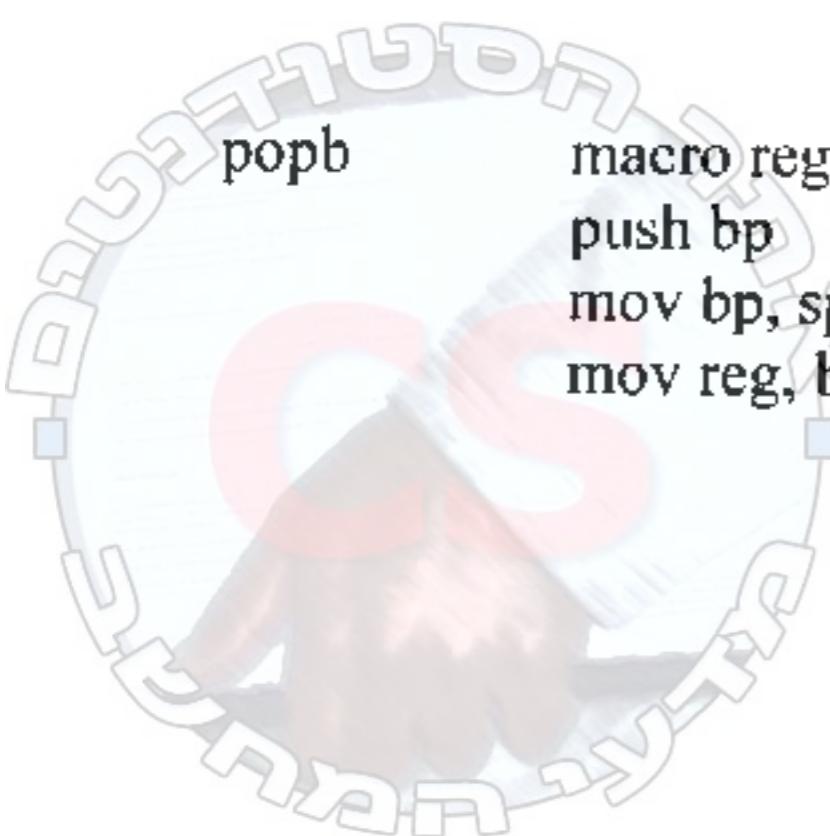
| ערך SP לאחר BP | (ריך) |
|----------------|-------|
| 77             |       |
| 78             |       |
| 79             |       |
| 80             |       |
| 81             |       |
| 82             |       |

ח) שינוי זה הופך את הבעיה בסעיף ז' לטעות בזמן אסמבלי, ככלומר המתכנת יכול להזות אותה ולטפל  
בזה – המקרו אינו אמר לטפל ברגיסטר בגודל מילה. ככלומר הבעיה נפתרת.

ט) עכשו ניתן להעביר `immediate` בגודל בית.

(ט)

```
popb macro reg
 push bp
 mov bp, sp
 mov reg, byte ptr [bp+2]
```



```
pop bp
inc sp
endm
```

- (4)
- א) 55-61 – השמה של שתי הפסיקות החדשות ב-TSR, אחרי הישוב מס' ה-paragraphs הנחוץ.
- ב) 8-15 – כל פעם שמבצעת פסיקה השעון. 15-35 – כל פעם שהמשתמש קורא ל-16h.int.
- ג) מטרת שורות 18-16 היא שהשורות 35-19 יבצעו רק אם נקרא שירות 0ffh של פסיקה 16; כל שורת אחר יעבור לשירות היישן ע"י שרשור, ויתבצע כרגע.
- ד) שימושות המשתנה timer: כמה שניות נותרו להמתנה למו שיכניס המשתמש. כאשר timer חיובי, סימן שאנו משתמשים ומהכים לקליטתתו וערך יורדת כל שניה ב-1. כאשר ערכו מגיע מחיובי לאפס, נגמר זמן ההמתנה שלנו ונמצאת פסיקה 16. אם המשתמשណה נotonin מראש (ב-X) ערך 0 או שלילי, נבדוק רק פעם אחת אם התקבלתו ואם לא, נמצאת הפסיקה ללא המתנה. (אם כן, נחזיר אותו כרגע).
- ה) 10-14 – מוניות את הזמן שעבר מזמן הקראיה האחרון לפסיקת השעון, בודקות אם עברה שנייה, ואם כן, נחסיר 1 מה-timer.
- ו) הלולאה ב-37-23 בודקת האם התקבלתו בזמן המתנה המורשת. יוצאים מהלולה בעבר מס' השניות שהוגדר ב-axp, או אם התקבלה אותה.
- ז) שורה 28 מאותחת למשתמש ע"י הדלקת ZF שלא נקלטתו במספר השניות הנתון. שורה 33 מאותחת שנקלטתו בזמן הנתון ע"י כיבוי ZF. השני בדגלים יורגש רק ביציאה מהפסיקה למשתמש.
- ח) לדעתו שורה 31, שמצויה 0 במשתנה timer, אינה משפיעה כיון שאחריה אנו יוצאים מהפסיקה (אם נkówרים ל-16h.int אך לשירות אחר) ובפעם הבאה שניכנס, נציב ערך אחר ב-timer.
- ט) (בונוס) במקרה זה הפסיקה לא תסתיים ברגע שתקלותתו, אלא תבצע כך: אם היהתו בכניסה ל-Int היא תצא מייד, אחרת תחכה בדיקת מספר השניות שהוקצתה ו ורק אז תבדוק אם נקלטתו.

Minute equ 60

(5)

.code

```
wait: mov dx, minute
 mov ah, 0ffh
 int 16h
 jnz print
 mov ah, 2
 mov dl, bell
 int 21h
 jmp wait
```

```
print: cmp al, escape
 je exit
 mov ah, 2
 mov dl, al
 int 21h
 jmp wait
```

```
exit: mov al, 1
 int 21h
 end
```

שאלה מס' 1 (25 נקודות)

א. להלן מספר זוגות של מושגים או פקודות בשפת אסטREL. עבור כל זוג, הסבר בקצרה את ההבדל המשמעותי בין שני המושגים או הפקודות.

i. השברת פרמטר by value

ii. השברת פרמטר by address

call far ptr x .ii

call near ptr x

iii. מלכזהן (trap, software interrupt) .iii

(exception)

sub ax,ax .iv

mov ax,0

x eq 1 .v

x = 1

ב. רשום ארבע פקודות שונות, כל אחת בעלת שם (mnemonic) אחסן, שגורמת להקטנה ב-1 של תוכן האוגר ax. אין להניח דבר על תוכן האוגרים או הזיכרון בתחילה מחזור הפקודה.

המשך שאלה מס' 1 בדף הבא

## שאלה מס' 1 (המשך)

ג. נ. הרגס-לבסים עשווני את המספר הבינרי 010101010 שヒנו בשיטת המפלים ל-2 וברוחב של 8 ביטים.

ה. טני המספרים  $x \cdot y$  שלහן הינם בבסיס הקפה-דצימלי וברוחב של 16 ביטים. חשב את ההפרש  $y - x$  בשיטת המפלים ל-2.

|         |   |   |   |   |
|---------|---|---|---|---|
| x       | 6 | E | 3 | 4 |
| y       | B | 0 | D | 1 |
| $x - y$ |   |   |   |   |

האם יש Möglichkeit אל הביט השמאלי ביזהר בעולות חישוב זו?

האם פועלות חישוב זו גורמת לגלישה?

iii. האם פועלות חישוב בין שני מספרים חיוביים כלשנים בשיטת המפלים ל-2 יכולה לגורום לגלישה? נמק את תשובה.

ד. להלן מספר פקודות אשר גודמות לשגיאה בזמן אספבל. שבור כל פקודה, הסבר בקצרה מהי השגיאה.

i. db 256

pop cs .ii

mov [di], [si] .iii

x: loop x-1000 .iv

ה. שבור כל אחד משלשה קטעי הקוד שלහן, דשומ את תוכנו של האורגן ap ברגמר ביצוע הקטע.

```
mov dx,0 .iii
mov ax,-1
mov bx,0
c: inc dx
dec ax
inc bx
cmp ax,bx
jle c
```

dx=\_\_\_\_\_

```
mov dx,0 .ii
mov cx,7
b: inc dx
mov ax,1
sal ax,cl
and ax,0Fh
loopz b
```

dx=\_\_\_\_\_

```
mov dx,0 .i
mov bl,16
var = 1
a: add dx,var
var = var+1
add dx,var
rol bl,1
jnc a
```

dx=\_\_\_\_\_

## שאלה מס' 2 (25 נקודות)

השיטה `partition` מקבלת שני פרמטרים במחסנית: כתובות של משרך של טילים, ומספר האיברים במשרך. פרמטר הכתובת הוא זה הקרוב יותר לדגם הפסמנית. המשרך נמצא בסגנון הנתונים. כל איבר במשרך הוא מספר. השיטה מסדרת מחרוזת את איברי המשרך כך שבל המספרים הזוגיים יופיעו לפני כל המספרים האי-זוגיים. השיטה מודידה באורך `ax` את מספר האיברים הזוגיים במשרך. מניחים כי במשרך יש לפחות מספר אחד זוגי ולפחות מספר אחד אי-זוגי.

- א. להלן אלגוריתם לשיטה `partition`, המורכב מטישה צפדים. עלין למסח את האלגוריתם בשפה אסטנלי בולוקי.ubo כל צעד, כתוב קטע קוד שבחצץ את הפעולות הנדרשות.

1. שטוח את תוכן האוגדים בהם משתמש השיטה.

```
partition proc near
step1: push bp
```

2. הצב באורך `ax` את כתובת המשרך. הצב באוגדים `is` ו- `ip` נוראים כך ש- `(is+ax)` יצביע על האיבר הראשון במשרך. step2:

3. בדוק האם האיבר ש- `(is+ax)` מצביע עליו הוא ספק זוגי; אם כן, הגדל את האורך `is` כך ש- `(is+ax)` מצביע על האיבר הבא במשרך, וחווד שוב על צעד 3. אחרת,ubo כל צעד הבא. step3:

4. בדוק האם האיבר ש- `(ip+ax)` מצביע עליו הוא מספר אי-זוגי; אם כן, הקטן את האורך `ip` כך ש- `(ip+ax)` מצביע על האיבר הקודם במשרך, וחווד שוב על צעד 4. אחרת,ubo כל צעד הבא. step4:

5. אם `ip < is`, המהלך בין שני איברי המשרך המזובעים על ידי `(is+ax)` ו- `(ip+ax)`, וחזור לצעד 3. אחרת,ubo כל צעד הבא. step5:

6. חלק את תוכן האורך `is` ב- 2 והעתק את התוצאה אל האורך `ax`, שוחר את האוגדים שנשמרו, וחזור מן השיטה. step6:

```
pop bp
ret
partition endp
```

## שאלה מס' 2 (המשך)

ב. נתון כרגע ה- `data` מילולו.

```
.data
list dw 1,-4,9,-7,2,0,6,-5
len dw (len-list)/2
count dw ?
```

כתבו קוד בשפת אסמבלי שקורא לשינגדה `partition` עם המערך `list` והאורך `len`, ומציין את הערך החוזר מן השינגדה בתוכה `count`. הקוד על בללי תיכנות נכון של קריאה לשינגדה.

ג. דשומ את תוכן המערך `list` והמספר `count` בرمד ביצוע קטע הקוד מסעיף ב'.

`list = _____`      `count = _____`

ד. בצד 3 באלגוריתם של `partition`, האוגר אס מקדים כל עוד `[si+bx]` מצביע על מספר זוגי. מה מונע מהמצביע `[si+bx]` לצא מנובלות המערך?

ה. מה יש לשנות או להוסיף בקובץ האסמבלי של השינגדה `partition.C`, כדי שניתן יהיה לך לא לשינגדה שתוור תוכנית בשפה C?

רשום את הכתובת (header) של השינגדה, כפי שתופיע בתחום הקוראה בשפה C.

ו. הוציאים לבדוק את יעילות הביצוע של השינגדה `partition`. ממציא האלגוריתם טוען כדלקמן: "מספר הפעולות לאיבר שמבצע השינגדה חסום על ידי קבוע. החסם אינו תלוי במספר האיברים במערך, לא במספר האיברים הזוגיים או האי-זוגיים, ולא בסדר שביניהם". האם טענה זו נכונה? נמק את תשובתך.

שאלות מס 2 (המשך)

להלכטן קוד שמשתמש בשירה partition. נתון כרגע ה-`data` מטעיב ב-`list` ומשתנה count בוגר ביצוע קטע הקוד.

```
1 mov cx,len
2 xor bx,bx
3 rotl: rol list[bx],1
4 add bx,2
5 loop rotl
6 push len
7 push offset list
8 call partition
9 mov count,ax
10 add sp,4
11 mov cx,len
12 xor bx,bx
13 rotr: ror list[bx],1
14 add bx,2
15 loop rotr
```

ח. תאר מה עשו קטע הקוד שבסעיף ז'.  
יש להזכיר הפעם איבוחי, ולא דקלוט של הפקודות המהויבות.

## שאלה מס' 3 (25 נקודות)

### חלק I

מתקנת חסר ניסיון קיבל משימה לכתחוך מאקדו. המאקרו מקבל כפרמטר אונר רב-תכליתי. אם תובן האונר הוא 0, המאקרו מציב באוגר את הערך 1, ואחרות, את הערך 0. המתכנתה היכן שלוש גירסאות שונות של המאקרו, אך בבדיקות שנערכו, שלושתן נתגלו כטעויות.

להלן שלוש הגירסאות של המאקרו.

```
isZero1 macro reg
 ifidni <®>,<0>
 mov reg,1
 else
 mov reg,0
 endif
endm
```

```
isZero2 macro reg
 cmp reg,0
 je set2
 mov reg,0
 exitm
set2: mov reg,1
endm
```

```
isZero3 macro reg
 cmp reg,0
 je set3
 mov reg,0
 jmp exit3
set3: mov reg,1
exit3: endm
```

א. דשום את קוד האסמבלי שנפרש על ידי כל אחת בשלוש הקוריאות למאקרו שללן.

isZero1 eax

isZero2 ax

isZero3 al

ב. הבדיקה `<ifidni <&reg>,<0>` שמבצע המאקרו `isZero1` אינה מתאימה. הסבר מדוע.

ג. הסבר מהי הבעה במאקרו `isZero2`.

ד. הסבר מהי הבעה במאקרו `isZero3`.  
האם בעיה זו היוצץ בכל תכנית אשר משתמש במאקרו? נמק את תשובתך.

ה. כהוב נידשה מתקנת של המאקרו, בשם `isZero4`, אשר עונה על כל הדרישות וושובחת נכון.

isZero4 macro reg

## שאלה מס' 3 (המשך)

### חלק II

כידון, הפקודות `push` ו-`pop` אינן מאפשרות לדוחף ולשלוף מן המחסנית בית בודד. להלן המاكדו `push`, אשר מקבל כפרמטר אונג בנויל בית (`ah`, `ah`, ובגד') דוחף את תוכנו למחסנית.

```
1 pushb macro reg
2 dec sp
3 push bp
4 mov bp,sp
5 mov [bp+2],reg
6 pop bp
7 endm
```

1. איזו הדרישה תתקבל בזמן אסמבלי עבור הקודיאה `[ax] push ?`  
האם הדרישה זו מוצביעה על בעיה במימוש המاكדו? נמק את תשובה.
2. איזו בעיה תונוצר בקודיאה `ax push` ומהי היא התגללה (בזמן אסמבלי או בזמן ריצה)?

3. נכניס שיפור קל במאקרו `push`, על ידי החלפת השורה 5 בשורה שללה.

```
5 mov byte ptr [bp+2],reg
```

האם שינוי זה פותר את הבעיה אונגה זיהית בסעיף 2? נמק את תשובה.

ט. איזה סוג נוספים של אופרנד ניתן להנביר למאקרו המשויף מסעיף ח?

ח. כתוב מאקרו בשם `popd`, אשר מקבל כאופרנד אונג בנויל בית, ושולף מראש המחסנית בית בודד אל תוך האונג.

```
popb macro reg
```



## שאלה מס' 4 (25 נקודות)

בזען, פסיקת השירותים המקלטת ah=16 מספקת שני שירותים עיקריים, כלהלן:  
באשר ah=ah, מותבצעת הוצאה של התו הבא מחוץ המקלט והשברתו למשתמש. במידה וחוץ  
כך, מותבצעת המטנה נס להקלטת זו. קוד ה- scan של התו מוחזר באונד ah, וקוד ה- ah באונד ah.

כאשר ah=ah, מותבצעת בדיקה האם יש או אין תווים בחוץ המקלט, זאת בלי להמתין להקלטת,  
ובלי להוציאו ממנה החוץ. אם ההזרה מן הפסיקה, הדגל ZF מצבין האם יש (ZF=0) או אין (ZF=1)  
תווים בחוץ. אם יש, קוד ה- scan של התו הבא בחוץ מוחזר באונד ah, וקוד ה- ah באונד ah.

בשאלה זו נגידיך שורות חדשות עבור פסיקה ah=0ffh, אשר מופעל כאשר ah=0ffh. השורות החדשות מהוות  
שילוב של שני השירותים שתוארו לעיל, והוא מבצע כדלקמן. אם יש תווים בחוץ המקלט, התו  
הבא מוצא מן החוץ ומועבר למשתמש. אחרת, מותבצעת המטנה להקלטת זו במשך זמן טונבל.  
במידה ובמשך זמן זה מוקלץ תו, הוא מוצא מן החוץ ומועבר למשתמש. במידה ולא, בתום זמן  
המטנה הפסיקה תוחמת למשתמש ללא השברת תו. ככל מקרה, עם ההזרה מן הפסיקה, הדגל ZF  
מצין האם מועבר (0 ZF=0) או לא מועבר (1 ZF=1) תו למשתמש. אם מועבר תו, קוד ה- scan של ah מוחזר  
באונד ah, וקוד ה- ah באונד ah.

משך הזמן שהשירותים ממתחים להקלטת זו נקבע על ידי המשתמש, ומועבר כפרמטר לפסיקה באונד dx.  
ערכו של האונד dx הוא מספר השניות שיש להמתין.

להלן הוכנית אשל מגידיה את שירותים המקלט החדש וטוענת אותו ליזכרון.

```

1 .model small
2 .code
3 pspPara dw ?
4 oldInt1c dd ?
5 oldInt16 dd ?
6 timer dw 0
7 msecs dw 0
8 newInt1c: cmp timer,0
9 jle exit1c
10 add msecs,55
11 cmp msecs,1000
12 jl exit1c
13 sub msecs,1000
14 dec timer
15 exit1c: jmp oldInt1c
16 newInt16: cmp ah,0ffh
17 je doNewFF
18 jmp oldInt16
19 doNewFF: push bp
20 mov bp,sp
21 mov msecs,0
22 mov timer,dx
23 getStat: mov ah,1
24 int 16h
25 jnz getChar
26 checkTime: cmp timer,0
27 jg getStat
28 noChar: or byte ptr [bp+6],01000000b
29 jmp exit16
30 getChar: mov ah,0
31 mov timer,0
32 int 16h
33 and byte ptr [bp+6],10111111b
34 pop bp
35 iret
36 install: mov pspPara,es
37 mov ax,351ch
38 inc 21h
39 mov word ptr oldInt1c,bx
40 mov word ptr oldInt1c+2,es
41 mov ax,3516h
42 int 21h
43 mov word ptr oldInt16,bx
44 mov word ptr oldInt16+2,es
45 mov ax,seg newInt1c
46 mov ds,ax
47 mov dx,offset newInt1c
48 mov ax,251ch
49 int 21h
50 mov ax,seg newInt16
51 mov ds,ax
52 mov dx,offset newInt16
53 mov ax,2516h
54 int 21h
55 mov dx,offset install
56 add dx,15
57 shr dx,4
58 add dx,seg install
59 sub dx,pspPara
60 mov ah,31h
61 int 21h
62 .stack 100h
63 end install

```

ארגו המחשב ושפט טפ (203.1130)  
קמפוס ב' תש"ס  
בחינה סופית - מועד א'

הוראות לנבחן

- משך הבחינה שלוש שעות.
- מותר להשתמש בכל החומר שנד, למינס מחשבים ומחשבונים מכל סוג.
- יש להסביר על כל השאלות.
- יש לרשותם את התשובות בגין השאלון במקומות המיועדים לכך.
- נא לנחות בכתב יד ברור ובקין, מומלץ להשתמש בטפרון ומתק.
- בשאלון זה 12 דפים, כולל דף זה. הדא כי כל הדפים נמצאים.

ב ה צ ל ת ה !

| מספר | ציוון | נקודות |
|------|-------|--------|
|      | 25    | שאלה 1 |
|      | 25    | שאלה 2 |
|      | — 25  | שאלה 3 |
|      | 25    | שאלה 4 |
|      | 100   | סה"כ   |



שאלה מס' 4 נקודת (ל)

— עבור הדור העתידי של מעבדי X86 מתכוננים פקודה חדשה בשם `exec`. פקודה זו מקבלת אופרנד אחד, המונדר כמו האופרנד של פקודה `cpn`. הפקודה `exec dest` גורמת לביצוע הפקודה שנמצאת בכתובת שמקבלת מן האופרנד `dest`, מיד לאחר מכן הביצוע ממשיך בפקודה שנמצאת אחריו `exec`.

בשורה 4, נופיעו הפקודות `exec` ו-`chmod 777`. מיד לאחר מכן, הביצוע בדגם של הלן, הפקודה `exec` בשורה 5 גורמת לביצוע הפקודה בשורה 1. תחשיב בשורה 4, ולפיכך, בשורה 5 גורמת להזoffset כוכבית.

```
1 dest: mov dl,'*'
2 mov dl,0
3 exec dest
4 mov ah,2
5 int 21h
```

מכיוון שהפקודה `exec` אינה קיימת במשבדים מודרני, נעשה לה סימולציה בעדרת מלכודות (פסיקת תוכנה). נבחר במלכודות מס' 550, שאינה בשימוש צורך אחר, ולבן נוכל לתוכנתה כרצונו.

בשגרות הראות בדוגמה, כדי לנשוח סימולציה לפקודה `dest exec` בקטע הקוד לעיל, נחליף את שורה 3

```
3.1 mov bx,seg dest
3.2 mov ex,bx
3.3 mov bx, offset dest
3.4 int 0f6h
```

שגרת השירות של מלכודת `new` נזונה להלן. ביצוע השגירה מתחילה בכתובות `new`.

הנח כי וקטור הפסיקה כבר מעודכן לכתובות זו.

```
21 pop saveRegs
22 push es
23 push bx

24 push bp
25 mov bp,sp
26 or word ptr
27 pop bp

28 iret
```

השורה וויאגרה תח'ו, ו בדמ הרא

#### שאלה מס' 4 (המשך)

- א. עליך להסביר כיצד נעשית הסימולציה לפקודה `exec` על ידי שגרת הסדרות של מלכודת #90. שבדף הקודם, השאלות להלן ינתו אותך בהבנת הקוד. טומלץ לעזרך על כל השאלות מראש.
- ב. מהים הפרמטרים שמקבלת שגרת השירותים של מלכודת #90 והיכן הם מועברים?

ii. מה עשוות שורות 17 – 29?

iii. מה עשוות שורות 23 – 21? מה נמצא במשתנה `saveRetAddr` אחרי ביצוע שורה אל? מה עשוה שורה 26?

iv. איזו פקודה מתבצעת מיד אחרי שורה 28? מה מיוחד במחב הדגמים בעת הביצוע?

v. כיצד מניעים לביצוע שורה 29?

vi. מה עשוות שורות 37 – 36? מה עשוה שורה 40?

vii. איזו פקודה מתבצעת מיד אחרי שורה 42?

### שאלה מס' 4 (המשך)

ב. כתוב קטע קוד שיחליף את שורה 7 בדוגמא שלහן, לצורך ביצוע סימולציה של הפקודה exec. תיכון: האופרנד של exec מתחילה בדיקת המודול cmdp.

```

1 .data
2 cmdp dd ?
3 .code
4 mov ax,@data
5 mov ds,ax
6 ...
7 exec cmdp

```

ג. ברגם הסימולציה של פקודת exec כלשהו, מצב הדגלים (למעט הדגל AF) הוא זה שקיים ברגע ביצוע הפקודה עלייה מצביית האופרנד של exec. הסבר כיצד מתקבל אפקט זה.

ד. נניח כי האופרנד של exec מצביע על הפקודה target push. האם הסימולציה תגרום להסתעפות אל הפקודה שב-target? נסken.

ה. נניח כי האופרנד של exec מצביע על פקודת push. ברגם הסימולציה, היכן נמצא האיבר שנדרש על ידי פקודת push זו? הסבר.

ו. נניח כי האופרנד של exec מצביע על הפקודה push. איזו בעיה נוצרת בסימולציה? בpush מה קורה לדגל AF במבנה וביציאה משגרת המפעלת מפקודת pop זו?

ז. האם האפשריות למנע את הבעה מסעיף ו' היא שהסימולציה לא תבצע את הפקודה עליה מצביע האופרנד של exec, אם זו פקודת pop. כיצד ניתן למשתמש באפקט זה בשגרות השדרות של מלכודות אטומי? תנו הסבר מילולי קצר, לא כתיבת קוד.

```
exit: mov ah, 4eh
 int 21h
 end
```

## תש"ס מועד א'

(1)

(א)

(i) בשמאל ה-CF ידלק ובימני לא.

(ii) הקריאה לפונקציה (משמאלי) תבוצע ע"י call ותכניס אוטומטית את ה-IP וה-CS כיוון שהוא far-pointer. בשביל להגעה לLABEL k (מימין) יבוצע jmp, ונאלץ לדחוף זדנית את ה-IP (לא CS).

(iii) השמאלי יקופז ל-x שהוא פוינטר מטיפוס far בעוד שבימני x הוא פוינטר near. (שנייהם ישנו את הרגלים אותו ערך).

(ב) Call far func .push EAX .add sp, -4 .Sub sp, 4 .(рак ב-386).

(ג) (i) dx = 131 (ii) dx = 16 (iii) dx = 18 (הסבר: צריך שיתאפסו ZF,SF,OF בזמן נתנו כחוצה פעולה dec, מה שקרה לראשונה כ-10 יורד מ-127 ל-126, לאחר שעבר על כל השליליות, אז SF=1, עבר לחובבים כאשר OF=1).

(ד) .ICCD2 = 1C530 h + 7A2

(ה)

|          | בבסיס 2 | בבסיס 16 | בבסיס 10 |
|----------|---------|----------|----------|
| 11001100 | CC      | -52      |          |
| 01110101 | 75      | +117     |          |
| 10101010 | AA      | -86      |          |

(ו) התשובות מימין לשמאלי.

| OF |
|----|
| 1  |
| 0  |
| 1  |
| 0  |

(ז)

| decimal | sign | Exponent | Mantissa    |
|---------|------|----------|-------------|
| -37.5   | 1    | +5       | 1.001011000 |
| 0.3125  | 0    | -2       | 1.01        |

(ח) .double1 = -1.25 .float2 = 54.25

(2)

(א)

push offset buf  
 push offset str  
 call compress  
 add sp, 4  
 mov length, ax

(ב) 0, -127, 3, -127, -3, 0.



- (ג) עד 128 – בהגדירה, כי אנו משתמשים בשיטת המשלים ל-2. אך במימוש ניתן להגיע בראצ' אחד רק למספרים בתחום 127 עד 127-.  
 (ד) השגרה מפסקה את תפקודה ומוחזירה 1-ב-א.  
 (ה) בכל רצף אוטוית מלבד הראשון, מספר התווים יהיה קטן באחד; אם יהיה רצף של TWO יחיד, יידלגו עליו להחלוטין.  
 (ו) לא – יש לשנות את שמה ל compress\_, בכל המקומות ששם מופיע, וגם יש לכתוב בראש התוכנית public\_compress כדי לציין שנייה לקרוא לה מבחוץ.  
 (ז) ניתן להעביר את שני התווים האלו כפרמטרים במחסנית, ולשמור אותם כמשתנים של הפונקציה, כך: ?1 db ?2 byte, ובתחילת הפונקציה לטעון מהמחסנית אל המשתנים.  
 (ח) כן, מחרוזות עם המספר 128-.

## Public\_uncompress

```

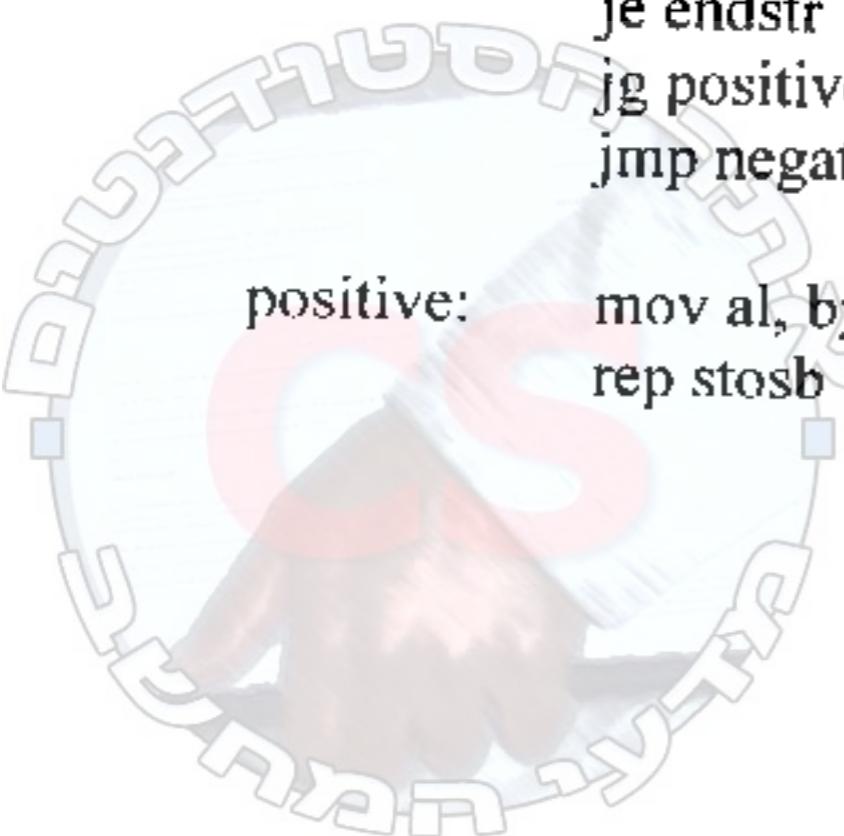
Proc _uncompress near
Push bp
Mov bp, sp
Push bx לדוחוף את הרגיסטרים המתאים ;
Push di
Push es
Push cx

Push ds ; cs ; לטען את cs
Pop es
Cld
Mov bx, [bp+4]
Mov di, [bp+6]
Mov cx, [bx]
Cmp cx, 0
Jc endstr
Jg positive

```

```
negative: Neg cx
 mov al, byte2
 rep stosb
 inc bx
 mov cx, [bx]
 cmp cx, -128
 je error
 cmp cx, 0
 je endstr
 jg positive
 jmp negative

positive: mov al, byte1
 rep stosb
```



```

inc bx
mov cx, [bx]
cmp cx, 0
je endstr
jl negative
jmp positive

endstr: mov [di], 0
 mov ax, di
 sub ax, [bp+6]
 jmp exit

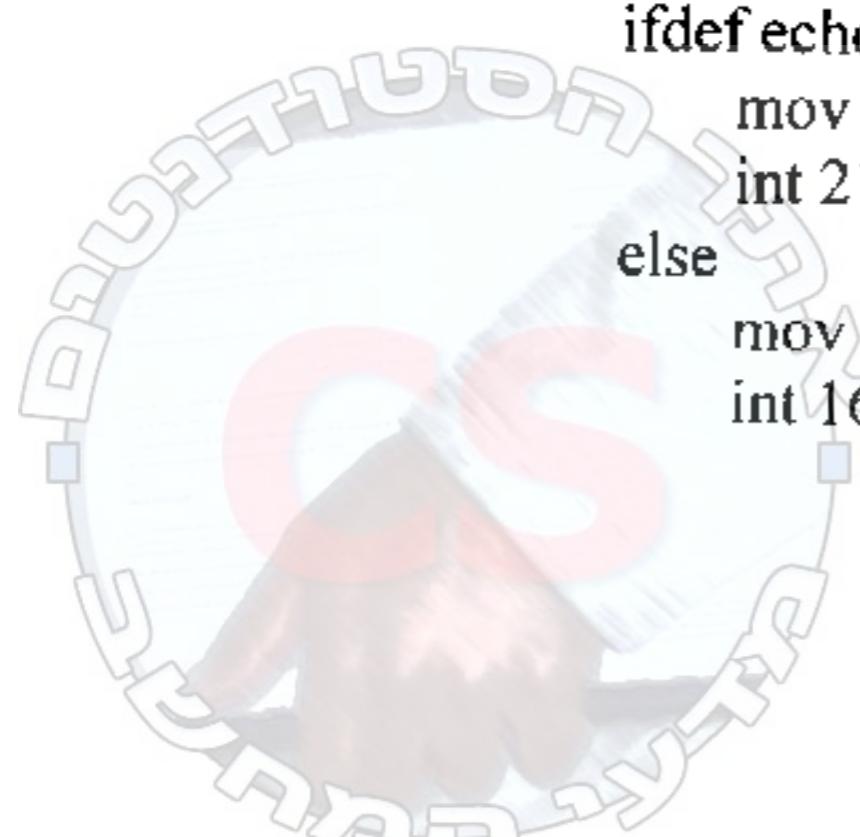
error: mov ax, -1

exit: pop cx
 pop es
 pop di
 pop bx
 pop bp
 ret

endp

```

|       |                                                                                                   |     |
|-------|---------------------------------------------------------------------------------------------------|-----|
|       |                                                                                                   | (3) |
| xtos  | macro<br>push bp<br>mov bp, sp<br>xchg ax, [bp+2]<br>xchg ax, [bp+4]<br>xchg ax, [bp+2]<br>pop bp | (8) |
| endm  |                                                                                                   |     |
| jcxnz | macro dest<br>local yes<br>jcxz yes<br>jmp dest                                                   | (2) |
| yes:  | endm                                                                                              |     |
| getc  | macro<br>ifdef echo<br>mov ah, 1<br>int 21h<br>else<br>mov ah, 0<br>int 16h                       | (2) |



```
endif
endm
```

(7)

```
pute macro n
 push dx
 push ax
 ifnb <&n>
 mov dl, n
 endif
 mov ah, 2
 int 21h
 pop ax
 pop dx
endm
```

(8)

הקוד שיפרש:

(i)

| Mystery 11 | Mystery -9 | Mystery 0 |
|------------|------------|-----------|
| Db '1'     | Db ' '     | Db '0'    |
| Db '0'     | Db '1'     |           |
| Db '1'     | Db '0'     |           |
| Db '1'     | Db '0'     |           |
|            | Db '1'     |           |

- (ii) פורש רצף של תווים, המיצגים בビינארי את המספר שנשלח לה כפרמטר.  
 (iii) תוכנו של AX לא מוגדר בזמן אסמליל ולא תהיה דרך לבדוק את ערכו מבחינת פרה-פרוססינג.

(9)

```
fact macro dest, n
 if n lt 0
 mov dest, n
 else
 result = 1
 num = n
 rept n
 result = result * num
 num = num - 1
 endm
 mov dest, result
 endif
endm
```

(4)

(8)

- (i) שגרת ה-int מקבלת כפרמטרים את כתובות ה-dest ב-es ו-bx. כלומר, ב- bx:es נמצאת הכתובת של ה"פקודה מהרץ למסלול" שיש לבצע.  
 (ii) 9-17 – שומרות את int – פסיקת ה-single step המקורית, ומזכירתה במקומה פסיקה זמנית חדשה, .afterExec

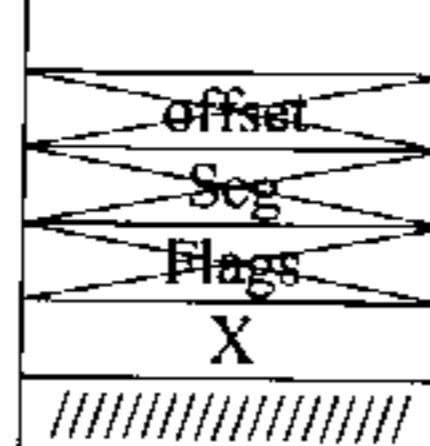


- (iii) לאחר ביצוע שורה 23-21, ב-`saveRetAddr` תימצא הכתובת המלאה (סגןט + אופסט) של הפקודה בה קראנו ל-`exec` (או, יותר נכון, לשגרה הממלצת של `exec`). במקומה תוכנס על המחסנית הכתובת של הפקודה מוחוץ למסלול. שורה 26 מדילקה את דגל ה-`TF` (`single step`) בדגלים המוחזרים מהפסקה, שייטענו כאשר נבצע `iret`.
- (iv) מיד אחריו שורה 28 תבצע השורה `dest` המיווחלת. מכיוון ה-`TF` ידליך, מיד אחריה תבוצע פסקה מס' 1.
- (v) מכיוון שורה 29 הוכנסה לפסקה מס' 1, הן יתבצעו מיד אחריה השורה `dest`.
- (vi) 36-37 – גורמות לכנתה החזרה להיות השורה שבה קראנו לפסקה `0f6h`. 40 – מכבה את דגל ה-`TF` עבור דגלי החזרה.
- (vii) אחרי פקודה 42 תבוצע הפקודה הרצiosa במסלול – כלומר פקודה 4 בקוד המקורי.

(ב)

```
Mov bx, word ptr [cmdp]
Mov es, bx
Mov bx, word ptr [cmdp+2]
Int 0f6h
```

- (g) אפקט זה מתאפשר מכיוון שבכניסה לפסקה `single step`, כמו כל פסקה אחרת, נשמרים הדגלים. הפסקה רק מכבה את `TF` ע"י שורה 40 – פעולה `AND` עם הכל 1 חוץ מה-`TF`, כלומר שאר הדגלים נותרים כמו שהם.
- (d) לא – מכיוון שברגע שהפקודה תסתיים, לפני ההסתעפות לכנתה `target`, יופעל ה-`single-step` ו ישזר את כתובות החזרה המקוריות של `exec`. כלומר – במקום Kapoor ל-`target` נקבע מקום המקורי.
- (h) מצב המחסנית:

**Push x**

|        |                            |
|--------|----------------------------|
| Int 1h | טעינת כתובות מלאה ודגלים ; |
| .      | .                          |
| .      | משחק עם כתובות החזרה ;     |
| iret   | הוצאת כתובות ודגלים ;      |

ביציאה מהפסקה `step single` ולכון גם מהטימולציה, האיבר שנדחף יהיה בראש המחסנית.

- (1) הבעה נוצרת מכיוון שככל `int` יגרור כיבוי של ה-`TF` שיושוחר רק ביציאה מה-`int` זהה. כלומר רק לאחר הפקודה הבאה נקרא ל-`int`, ואז תתרחש פקודה לא מתוכננת.
- (2) ניתן להסתכל על הכתובת הנשלחת ב-`es:bx` ל-`0f6h-int`. תוכנה יהיה ה-`opcode` של הפקודה `int k`, וכן יכול להשווות את השדות הרלוונטיים ויצאת מ-`0f6h-int` בלי לבצע כלום.

**תשס"ב מועד א'**(1)  
(2)

שאלה מס' 1 (25 נקודות)

- א. להלן מספר זוגות של קטעי קוד בטפחת אסמבלי. עבור כל זוג, הסבר בקצרה את ההבדל  
המזהה בין שני הקטעים.

```
mov bx,2
add bx,-1
```

```
mov bx,2 .i
sub bx,1
```

```
p proc far
 mov ax,0
 ret
endp
```

```
p: mov ax,0 .ii
 ret
```

```
push 100h
push seg x
push offset x
izet
```

```
push 100h .iii
popf
jmp x
```

- ב. רשום أدכט פקודות שונות, כל אחת בעלת שם (mnemonic) אחלה, אשר תגרום להקטנתה ב-4.  
 של האונר cs. אין להניח דבר לגבי תוכן האונרים או היזכרון בתחילת מחרוז הפקודה.

- ג. עבור כל אחד משלשות קטעי הקוד של להלן, רשום את תוכנו של האונר ax בגרף ביצוע הקטע  
 המשתמש בבסיס עשרוני.

```
xor dx,dx .iii
mov cx,1
c: inc dx
dec cl
pushf
pop ax
and ax,08c0h
jnz c

dx= _____
```

```
xor dx,dx .ii
mov cx,1
b: inc dx
rcl cx,1
inc cx
jb b

dx= _____
```

```
xor dx,dx .i
mov cx,1
a: inc dx
shr cx,1
loop a

dx= _____
```

### שאלה מס' 1 (המשך)

- ד. תרגם את הכתובת הלוגית (segment:offset) שלහלו לכתובת פיזית, כשההמעבד ב מצב real.  
רשמי את התוצאה בבסיס 16.

1C53h:7A2h



- ה. השלים את הפעולה שלහלו על ידי הרגום המספרים בסיסים לבסיסי. בסיס 2 ו- 16 המשמש ביצוע בשיטת המשלים 2-2 ברוחב 8 ביטים. בסיס 10, המשמש ביצוג הרגיל (עם סימן).

| בסיס 2   | בסיס 16 | בסיס 10 |
|----------|---------|---------|
|          | -52     |         |
|          | 75      |         |
| 10101010 |         |         |

- ו. בצע את פעולות החיבור והחיסור שלහלו בשיטת המשלים 2-2 ברוחב של 16 ביטים. כל המספרים נתונים בסיס 16. דשומ גם את התוצאות בסיס 16.  
ציין את ערכי הדגליים CF ו- OF במעבר כל פעולה, כדי שהיו נקבעים על ידי ביצוע במעבד 8x8.

|                          |                          |                          |                          |
|--------------------------|--------------------------|--------------------------|--------------------------|
| 0709<br>+<br><u>7548</u> | 8052<br>-<br><u>0f6e</u> | 7f45<br>+<br><u>c1e2</u> | 7f45<br>-<br><u>c1e2</u> |
| CF=                      | CF=                      | CF=                      | CF=                      |
| OF=                      | OF=                      | OF=                      | OF=                      |

- ז. תרגם את המספרים שבבלה מבסיס 10 לייצוג סטנדרטי בשיטת הנקודה הצפה בסיס 2. רשמי את הנקודה לא bias (ראה דוגמא).

| Decimal | sign | Exponent | Mantissa |
|---------|------|----------|----------|
| 9.0     | 0    | +3       | 1.001    |
| -37.5   |      |          | 1.       |
| 0.3125  |      |          | 1.       |

- ח. להלן מספר משתנים של תכנית אסמבלי, המכילו עדכיהם בשיטת הנקודה הצפה. תרגם&ען הערךים לבסיס 10 ביצוג ללא חזקה (ראה דוגמא).  
חכורת: בשיטת הנקודה הצפה, שדה החזקה הוא מספר ללא סימן הכלול bias.

float1 dd C1180000h dd -9.5

float2 dd 42590000h dd 0.3125

double1 dq C008000000000000h

## שאלה מס' 2 (25 נקודות)

הגדה: לזרבי שאלה זו, מחרחת היא רצף של בתים המסתווים בבית שמכיל את הערך 0. אורך המחרחת הוא מספר הבתים, לא כולל הבית המסייע. זהה מחרחת כמו בשפה C.

נתונה הפונקציה שבורתה (בשפת C): (չטף C):

הפרמטר `inStr` הוא מצביע למחרחת שמכילה שני טוני תווים בלבד: התו 'x' והתו 'z'.

הפרמטר `outStr` הוא מצביע לחיצן שנודלו בלחץ מוגבל (בלוטה, מניחים כי יש מספיק מקום בתחום).

הפונקציה `compress` דוחשת את המחרחת `inStr`. ומצהה מחרחות חדשות (מחרחת דחוסה), שהיא

קצתה יותר מ- `inStr`, ובניהם באופן שמאפשר לשחזר פוננה את המחרחות המקוריות.

המחרחות הדחוסה נכתבת אל החיצן `outStr`. אורך המחרחות הדחוסה מוחדר כערך הפונקציה.

הגדה: הדחיסה נעשית בשיטה הבאה. רצף של סיב בתים שכולם מכילים את התו 'x' מוחלפים בבית

בוזד שמכיל את המספר `m`. רצף של סיב בתים שכולם מכילים 'z' מוחלפים בבית בוזד שמכיל את

המספר `n`. המספרים `m` ו- `n` מוצגים בשיטת המשלים ל-2.

לדוגמא, אם נדחס את המחרות `string1` שלמטה לפיה זו, נקבל מחרחת הינה `string2`.

```
string1 db 'xxxxxxxxxxxx',0
string2 db 3,-1,1,-5,4,0
```

להלן מימוש הפונקציה `compress`. הפרמטרים מוחבדים לפי המוסכמות של שפת C.

```

1 .model uses16 small
2 .code
3 byte1 equ 'x'
4 byte2 equ 'y'
5 compress proc near
6 push bp
7 mov bp,sp
8 push bx
9 push cx
10 push di
11 push es
12
13 mov ax,ds
14 mov es,ax
15 mov di,[bp+4]
16 mov bx,[bp+6]
17 cld
18
19 nextRun: mov al,[di]
20 cmp al,0
21 je endString
22
23 cmp al,byte1
24 je scan
25 cmp al,byte2
26 jne badString
27
28 scan: mov cx,128
29 repe scash
30 dec di
31
32 encode: sub cx,127
33 cmp al,byte2
34 je insert
35 neg cx
36
37 badString: mov ax,-1
38 jmp exit
39
40 endString: mov byte ptr [bx],0
41 mov ax,bx
42 sub ax,[bp+6]
43
44 exit: pop es
45 pop di
46 pop cx
47 pop bx
48 pop bp
49 ret
50
51 compress endp

```

## שאלה מס' 2 (המשך)

- א. כתוב קטע קוד שקורא לשגרה `compress` כדי לדחוס את המחרוזת `str` שלগלו לתוך התוצאת `buf`, ומציב את אורך המחרצת הדחוסה במשתנה `length`.

```
.data
str db 'xxyy',130 dup('x'),130 dup(0,'y') dup (?)
buf db 100 dup (?)
length dw ?
.code
mov ax,@data
mov ds,ax
```

- ב. רשות את המחרחות הדומות שנוצרת על ידי השגרה `compress` בקריאה שבוטע?

- ג. מהם הערבים המksamלים האפשריים שבור `#` וubah `#` לשי הגדלת שיטה הדומות שבז' הקודם? נמק.  
מהם הערבים המksamלים בהם משתמש במיומש השיטה בשגרה `compress`?

- ד. מה שורה השגרה `compress` כאשר המחרצת המושבדת על ידי הפורט `inst` אינה חוקית?

- ה. מה יקרה אם נמחק את שורה 26?

- ו. האס ניתן לקרוא לשגרה `compress`, כפי שהוא ממומשת בשפת אסמבלי בדף הקודם, גם מתוך תוכניות בשפת C? נמק.

- ז. הוציאם לשככל את הפונקציה `compress`, כך שתבצע דחיסה של מחרחות המכילה שני סימני תווים בלשף, לאו דווקא התווים 'x' ו-'y'. בבל קריאה לפונקציה, התווים עשוויים להיות מסוגיים אחדים. הצע כיצד למסב שיכלול זה. תן הסבר מילולי קצר, בלא כתיבת קוד.

## שאלה מס' 2 (המשך)

i. נתונה הפונקציה שבסותרתה:

```
int uncompress(char *inStr, char *outStr);
```

הפרמטר `inStr` הוא מצביע למחזורת דחועה, והפרמטר `outStr` הוא מצביע להוצאת גורעל בלתי מוגבל. הפונקציה מבצעת טרנספורמציה הופוכה לו של הפונקציה `compress`.  
כלומר, הפונקציה `uncompress` בונה מן המחרוזת הדחוסה `inStr` את המחרוזת המלאה (שהבילה רק תווים מהסוגים 'x' ו-'y'). התוצאה נכתבת אל החזוץ `outStr`.  
אורך המחרוזת המלאה מוחזר כערך הפונקציה.

ii. האם קיימת מחרוזת שהיא בלתי חוקית ביחס לדימטר `inStr` לפונקציית `uncompress`?  
נקן את תשובהך. תן דאה סעיף ג'.

iii. ממש את הפונקציה `compress` בשפת אסמבלי.  
יש לאפשר קריאות לפונקציה זו מתוכניות בשפת אסמבלי וזה מתוכניות בשפת C.  
חובה להשתמש בפקודה `stos` לבניית המחרוזת המלאה.



שאלה מס' 3 (25 נקודות)

א. כתוב מאקרו בשם `soz` אשר מחליף בין שתי ה=טילים שבראש המהיטנית.

ב. הפקודה `dest zxcz` מבצעת הסתעפות אל `dest` אם תוכן האונגר `cx` הוא 0. הפקודה אינה משנה את הדגמים. לא קיימת פקודת דומה שמבצעת הסתעפות כאשר האונגר `cx` שונה מ-0.

כ. כתוב מאקרו בשם `zmxz` שמקבל אופרנד היחיד `dest` המוגדר כמו האופרנד של הפקודה `zxcz`. המאקרו מבצע הסתעפות אל `dest` אם האונגר `cx` שונה מ-0. המאקרו אינו משנה את הדגמים.

ג. כתוב מאקרו בשם `getc` שקוראתו מן המקלצת. קוד ה-`i` של `ascii` של התו מוחזר באונגר `ta`. התו יודפס על המסך רק אם האסמבול הופעל עם האופציה `Decho`. יש לבנות את המאקרו כך שבל קריאה תפרוש פקודת `ta` אחת בלבד. מותר להדوس את תוכן האונגר `ta`.

ד. כתוב מאקרו בשם `putc` שמקבל אופרנד אחד בגודל בית בכל שיטת מיעון. המאקרו מודפס על המסך את התו אשר קוד ה-`i` של `ascii` שלו ניתן על ידי האופרנד. אם הקריאה למאקרו היא ללא אופרנד, יודפס התו שנמצא באונגר `tp`. אפשר להרווש את תוכן האונגרים.

לדוגמא, הקריאה: 'x' `putc` תדפיס את התו 'x'. הקריאה: `ta putc` תדפיס את התו שנמצא באונגר `ta`. ואילו הקריאה: `putc ta` תדפיס את התו שנמצא באונגר `tp`.

### שאלה מס' 3 (המשן)

ה. להלן הגדרת המacro mystery.

```

mystery macro n
 if n lt 0
 db '-'
 mystery -n
 exitm
 elseif n gt 1
 mystery n/2
 endif
 if (n/2)*2 eq n
 db '0'
 else
 db '1'
 endif
endm

```

רשותם את קטע הקוד בשפת אסטREL שນפרש על ידי כל אחת משלוש הקוריות שלහלו לmacro mystery. יש לדפוס בק שורות שייצרו קוד מכונה. אין צורך לדחוס שורות אחרות (כגון הנחיות לאסטREL מותנה, וכך).

mystery 11

mystery -9

mystery 0

ii. תאר מה טושה המacro mystery.

iii. איזו בעיה תיווצר בקריאה ? mystery ax



המשן שאלה מס' 3 בדף הבא

### سؤالה מס' 3 (המשך)

1. כתוב מacro בשם `fact` שמקבל שני אופרנדים. האופרנד הראשון `dest` מוגדר כמו האופרנד הראשוני של הפקודה `mov`. האופרנד השני `t` הוא מספר מיידי (קבוע). המacro מציב ב- `dest` את הערך `t`. עבור `0<t`, נגיד `t=1`.

`fact t fact eax` תציב באgreg `eax` את הערך 020. הקראיה 3-,-

על המacro לבצע את כל החישוב בזמן אסטטלי, ולפרוש פקודת מכונה אתה ויחידה, טהרא פקודת `mov`. חובה להשתמש באלגוריתם איטרטיבי (לא רקורסיבי).

הערה: אין צורך לבודד בדיקה האם `t` גולש מקיבולת האופרנד `dest`. במקרה זה האסטטלי יודיע ממילא על שגיאה בפקודה `mov` שנפרשה.

`fact macro dest,t`



ארגון המחשב ושפת ספ (203.1130)

סמסטר ב', תשס"ב  
בחינה סופית - מועד א'

הוראות לנבחן:

- משך הבחינה שלוש שעות.
- מותר להשתמש בכל חומר נahr, למעט מחשבים ומחשבונים מכל סוג.
- יש להסביר על כל השאלות.
- יש לרשום את התשובות בגוף השאלון במקומות המיועדים לכך.
- נא לבתוב בכתב יד ברזר ונקוי. מומלץ להשתמש בעפרון ומחק.
- בשאלון זה 13 דפים, כולל דף זה. ודא כי כל הדפים נמצאים.

ב ה צ ל ח ה !

| ניקוד | ציון   |
|-------|--------|
| 25    | שאלה 1 |
| 25    | שאלה 2 |
| 25    | שאלה 3 |
| 25    | שאלה 4 |
| 100   | סה"כ   |



## שאלה מס' 4 (25 נקודות)

מתוכננת גרסה חדשה של מערכת הפעלה DOS. בגרסה זו נוספו 256 שירותים חדשים, המופעלים בעחרת מנגנון הפקיות, בדומה ל- 256 שירותים המופיעים.

לצורך גישה לשירותים החדשניים, הוסיפו למערכת הפעלה טבלה חדשה של וקטורי פסיקה, שנמצאת בזיכרון החל מבתוות פיזית 400 (כזכור, הטבלה הקיימת נמצאת בכתובת פיזית 0). לטבלה החדשה מבנה זהה לטבלה הקיימת.

שירותי המערכת החדשים מופעלים מהתוכנית בעזרת החונית אסמבילד חדשה: הן trap 255 הוא מספר השירות החדש המבוקש (לדוגמא: trap 45). הנהניה זו אומرت כאשר הן trap 255 הוא מכוון לשירות קוד אשר יגרמו להפעלת שגרת פסיקה דרך הכניסה ה- החדשה של וקטורי הפסיקה. (הערה: פקודת המבונה int פונעת כרגע ללא שום שינוי).

בידוע, מעבדי 80x תומכים רק ב- 256 פסיקות, וכטובן שלא קיימת פקודת מכונה להפעלת הפקיות החדשניות. כדי להתגבר על מגבלות חומרה אלה, החליטו יצרני מערכת הפעלה להשתמש בפסיקה 255 int מקבוצת הפסיקות הקיימות, במנגנון אשר דרכו ניתן יהיה להציג לפסיקות החדשניות. (הנימ שהפסיקה 255 int לא מנוצלת בגרסאות קודמות של מערכת הפעלה).

האסמבילד עבר שינוי כדי לתרום בהנחיה החדש trap. כאשר האסמבילד נתקל בתוכנית בשורה הן trap הוא פורט בתוכנית את קטע הקוד הבא:

```
int 255
db
```

שגרת הסדרות של הפסיקה 255 int שונתה גם היא בהתאם, והיא נתונה להלן.

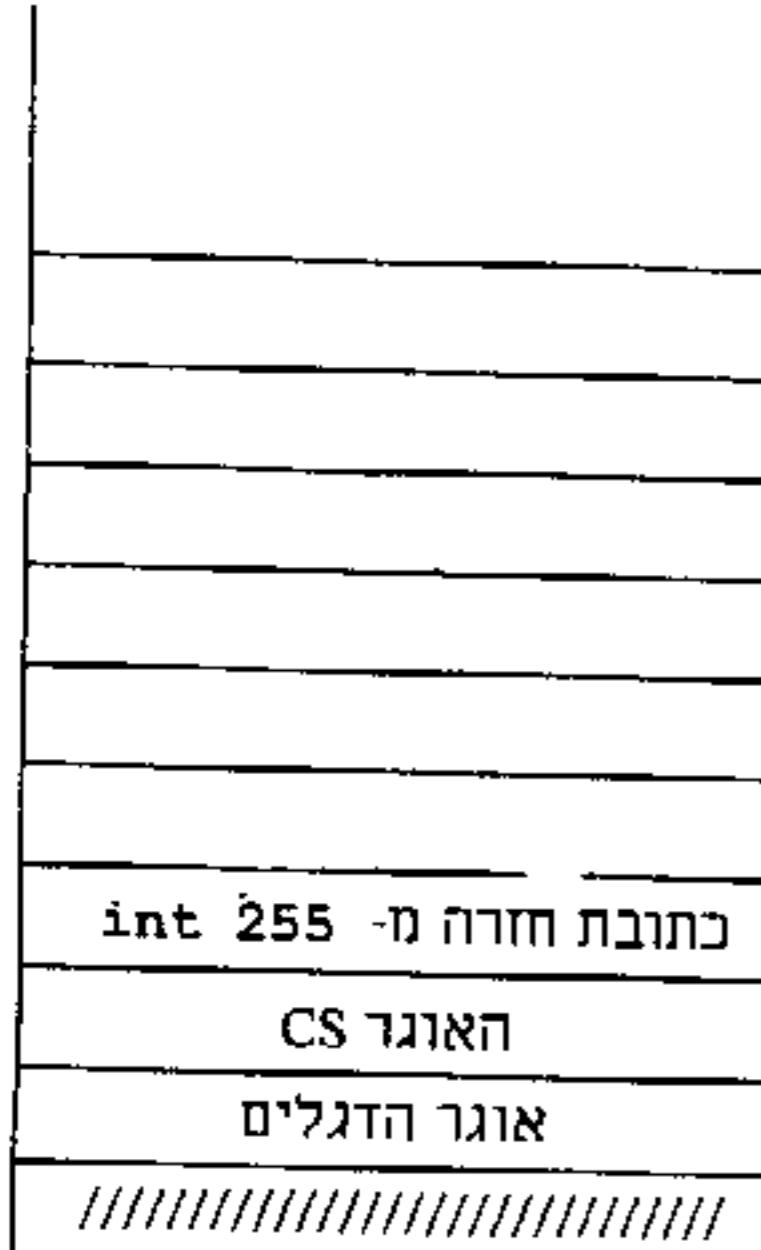
```

1 int255 proc far
2 pushf
3
4 sub sp,4
5
6 push bp
7 mov bp,sp
8 push es
9 push bx
10
11 les bx,[bp+8]
12 mov bl,es:[bx]
13 mov bh,0
14
15 inc word ptr [bp+8]
16
17 shl bx,2
18
19 push 80h
20 pop es
21
22 push dword ptr es:[bx]
23 pop dword ptr [bp+2]
24
25 pop bx
26 pop es
27 pop bp
28
29 iret
30 endp

```

### שאלה מס' 4 (המשך)

א. נניח כי מתבצעת הפקודה `int t = 255` הנמצאת בקטע הקוד שנפרש על ידי הנקה **9 trap**. עליך להסביר כיצד פועלת שגרת השזהות של הפקודה `int t = 255` שבזף הקוד. **מומלץ** לנבוות על כל השאלות מדרаш.



ב. פרט את תוכן המחסנית אחורי ביצוע שורה 7.  
 רשום כל טיליה במחסנית בנפרד בשרטוט ממשמאלי.  
 כמו כן, שרטט להיכן במחסנית מצביעים האוגרים `ka` ו- `ds`.

ג. מה עשוות שורות 10-8?  
 מה מכיל האונד `ax` אחורי ביצוע שודות אלה?

ד. מה עשוות שורה 11? מדוע היא נחוצה?

ה. מה עשוות שורה 12? מה מכיל האונד `ax` אחורי ביצוע שודת זו?

ו. מה עשוות שודות 14-13?

ז. מה תפקידה של שורה 3? מה דוחפות הפקודה `push` בשורה 15 למחסנית?  
 להיכן מועבר ערך זה על ידי הפקודה `pop` בשורה 16?

ח. להיכן עובר הביצוע אחדי שורה 20? אילו דגלים נמצאים באונד הדגלים אחורי שורה 20?

ט. מה נשאר במחסנית בגמר הפקודה בשורה 20? لماذا משמש תוכנו זה?



## שאלה מס' 4 (המשך)

ב. האם אפשר לשנות את השורות 9-8 בשגרת השירות של פסיקה 255 `int` כדי נמק?

```
8 mov bx,[bp+8]
9 mov ah,[bx:cs]
```

ג. הוחלט להכין שינוי במנגנון הפעולה של שירות המערכת החדש.  
בשורה 9 נתקל בתכנית בשורה `n trap` הוא פורס את קטע הקוד הבא.

```
push n
int 255
add sp,2
```

עליך לנתח את שגרת השירות של הפסיקה 255 `int` בהתאם לשינוי זה בהנחה `trap`.  
רשום את מספרי השירותים שיש לשנות, ואת קוד האסטבלי החדש של שירותים אלה. אם יש צורך  
למחוק שורה, ציין זאת ליד מספר השורה. הכנס שינויים הכרחיים בלבד.

טט' שורה

קוד אסטבלי חדש

ד. מתוכנת חסר אחריות כתוב תכנית ובה קטע הקוד שלහלו.

```
push 5000
int 255
add sp,2
```

בהתבונת השינויים שהכנתם בסעיף ג', מה יקרה בשגרת השירות של 255 `int` כשיתבצע קטע קוד זה?

ה. להלן צמה נוספת לשינוי במנגנון הפעולה של שירותים.  
בשורה 9 נתקל בתכנית בשורה `n trap` והוא פורס את קטע הקוד הבא.

```
mov ah,n
int 255
```

הנץ כי שגרת השירות של הפסיקה 255 `int` מעודכנת בהתאם לשינוי זה בהנחה `trap`.  
ציין בקצרה יתרונות וחסרונות של הצעה זו, בהשוואה למנגנון המקורי בדף 10.



## שאלה מס' 4 (המשך)

1. עליך להגדיל את כמות השירותים החדשניים מ- 256 ל-512, וזאת על ידי הכנסת שינויים קלים בלבד במנגנון שבדף 10. הנח כי טבלת וקטורי הפסיקות החדשנית הורחבה ל- 512 כניסה.

2. רשם את קטע הקוד שיפורו האסמבלי עבור הhnihie **trap** כאשר  $111 \leq n \leq 0$ .

ii. רשם את מספרי השורות שיש לשנות בשגרת השירות של הפסיקה 255 **int** ואת קוד האסמבלי החדש של שירותים אלה. אם יש צורך למחוק שורה, ציין זאת ליד מספר השורה. הכנס שינויים הכרחיים בלבד.

מס' שורה

קוד אסמבלי חדש

3. בהינתן השינויים שהכנסה בטעמי ו', האם אפשר להגדיל את כמות השירותים החדשניים ל- 65536 (כלומר ל-  $2^{16}$ ) ללא שינויים נוספים בשגרת השירות של 255 **int**? נמק!

4. כתוב קטע קוד אשר יבצע החלפה של וקטור הפסיקה של **trap 9** בטבלת וקטורי הפסיקות החדשנית. כתובות שגרת הפסיקה הקודמת תישמר במשתנה **saveTrap9**, והכתובת הנסצת תהיה של התוית **myTrap9**. המשמש בשולץ התוכנית הראשית שלהן. הנח כי התוכנית מתמילה לרוץ בתוית **main**. רמז: האם אפשר להעזר בכך בשידות הקיום **int 21h ?**

```
.model use16 small
.stack 100h
.code
saveTrap9 dd ?
myTrap9: . .
 iret
main:
; replace interrupt vector for new trap 9
```



```
jz end
popf
pop ax
jmp dest
end: popf
 pop ax
endm
```

(5)

```
jnd macro dest
 local no
 jd no
 ifnb <dest>
 jmp dest
 endif
no: endm
```

(6)

שימוש לב שהפרמטרים נדחפים (עפ"י הגדרת המקרו) מהסוף להתחלה, כלומר מימין לשמאל.

(i)

| Callp f1, x, ax, [bx], 100 | Callp f2, y, z | Callp f3 |
|----------------------------|----------------|----------|
| Push 100                   | Push z         |          |
| Push [bx]                  | Push y         |          |
| Push ax                    | Call f2        |          |
| Push x                     | Add sp, 4      |          |
| Call f1                    |                |          |
| Add sp, 8                  |                |          |

(ii) Nump סופר את מס' הפרמטרים שנדחפו, כדי לנוקות את המחסנית בסיום הפעולה.

(h) שימוש לב לסדר דחינת הפרמטרים.

```
calld macro fname, p1, p2, p3, p4
ifdef revp
 callp fname, p4, p3, p2, p1
else
 callp fname, p1, p2, p3, p4
endif
endm
```

(7)

```
dbw macro name, n
if n gt 127 or n lt -128 ; צור מילה ;
 ifnb <&name>
 &name& dw &n
 else
 dw &n
 endif
else
 ifnb <&name>
 &name& db &n
 else
 db &n
```



```

 endif
 endif
endm

```

(j)

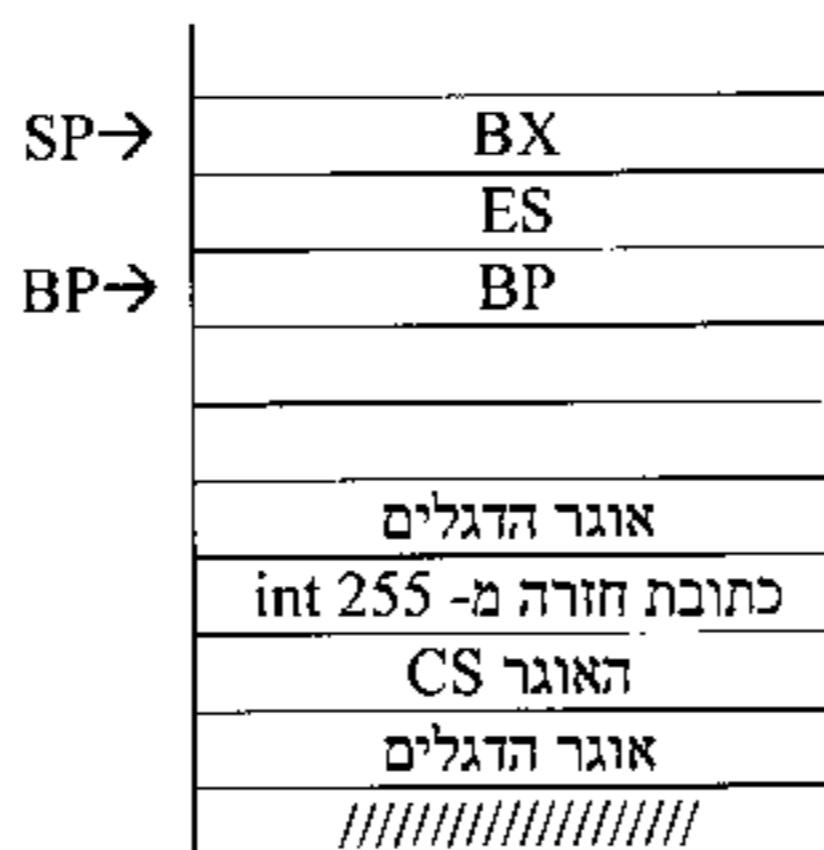
(i) ייפרש: dw var4. בזמן ריצה תיווצר בעיה כיון שבדי ליצג את המספר 35,000, בשיטת המשלים ל-2 דרישים 17 בתיים, קלומר pp ולא dw.

(ii) ייפרש: dw. אין בעיה בקריאה. אם מתכנת ריצה לגשת למילה זו הוא יאלץ לחשב את האופסט שלה.

(4)

(x)

(i)



(ii) קוראות את המקום הבא בסגמנט הקוד, מה שمبיא את הערך של ch-l-ax, כך שב-ax יהיה ch.

(iii) IP מקבל את הפקודה הבאה לביצוע אחרי הבית שבו מופיע ch, כדי שנחזור למקום הנכוון בסוף ביצוע int 255.

(iv) שורה 12 מבצעת הכפלת b-4, ולכן BX יכיל את הערך ch. זהו האופסט של ה-trap הרצוי מכיוון שהוא כל פסיקה הוא 4 בתיים.

(v) 13-14 – מכניות 80h ל-ES נך שביחסוב הכתובת הפיזית, ES יצביע על 800h, תחילת וקטור הפסיקות החדשנות.

(vi) 3 – שמורת מקום לכתובת הקפיצה הנכונה של ה-trap, שוחשב במהלך פסיקה 15. – הפקודה Push דוחفت את הכתובת הלוגית המלאה של ה-trap, ושורה 16 מעבירה ערך זה למקום הריך שנשמר בשורה 3.

(vii) לאחר שורה 20 נקבע ל-ch trap, והדגלים הם אלו שנשמרו בשורה 2 – שהיו בקריאה ל- trap .

(viii) אחרי שורה 20 נשארת על המהסנית כתובות החזרה לפקודה שאחורי ch db, והדגלים שהיו בכניסה ל-255 הונצחים. קלומר כשהפסיקה החדשנה (n) (trap) תבצע Iret, הפקודה תשוחזר וכן הדגלים, כמו בפסקה רגילה.

(b) לא, מכיוון שפסיקות הן כמו פונקציות מטיפוס far במובן שהן עשוות להחליף את סגמנט הקוד. לא ניתן להסתמך על כך שSEGMENT הקוד הנוכחי הוא זה שקיים במחסנית.

(g)

| מספר שורה  | קוד חדש         |
|------------|-----------------|
| במקום 8-10 | Mov bx, [bp+14] |
| 11         | למחוק           |

(d) אין פסיקה 5000 (יש רק 255-0), ולכן ניכנס (לאחר שורה 20) לכתובת שהיא מחוץ לווקטור הפסיקות שלנו. אין לדעת מה יקרה.



ה) היתרון של שימוש ב-AH הוא ביצועים מהירים יותר ופחות פקודות מאשר מציאת המשנה שבקובד. חסרונו: זה פחות בטוח להשתמש ב-AH לפני ביצוע פקודות int כי יש במקרה המשנות אותו. זה לא עובד עם שפת C מבחינה קריאה לפונקציה. יותר בטוח ומehler משתי השיטות זה לשמר את הערך על המחסנית, גם עבור שימושים עתידיים וגם כדי לחסוך ברגיסטר.

(i)

(ii)

Int 255

Dw n

| מספר שורה  | קוד חדש                |
|------------|------------------------|
| 9-10 במקום | Mov bx, cs:[bx]        |
| 11         | Add word ptr [bp+8], 2 |

ז) לא ניתן להגדיל את כמהות השירותים ל- $2^{12}$ , כי שורה 12 תזורך את שתי הספרות ה-MSB ולכן ניתן להוסיף עד  $2^{14}$  פסיקות, שיתפסו בדוק גודל סגמנט (כל פסיקה – 4 בתים).

ח) הערכה: לא ניתן להשתמש בשירות הקיים של int 21h כי הוא מעתיק לווקטור הפסיקות המקורי.  
Mov ax, 80h

Mov es, ax

Mov ax, es:[9\*4]

Mov word ptr saveTrap9, ax

Pushf

Cli

Mov ax, offset myTrap9

Mov es:[9\*4], ax

Mov ax, seg myTrap9

Mov es:[9\*4+2], ax

Popf



שאלה מס' 1 (25 נקודות)

- א. להלן מספר זוגות של קטעי קוד בשפת אסמבלי. עבור כל זוג, חשב בקצורה מהו הבדל המהוות בין שני הקטעים.

|                      |                              |
|----------------------|------------------------------|
| jmp t                | .i.<br>cmp ax, 0<br>jae t    |
| mov ah, 0<br>int 16h | .ii.<br>mov ah, 7<br>int 21h |
| rol ax, 8            | .iii.<br>ror ax, 8           |
| .model use16 small   | .iv.<br>.model use32 small   |

- ב. עבור כל אחד משלש קטעי הקוד שלහלן, רשם את תוכנו של הארג' **ax** בגרף ביצוע הקטע.  
רשום את התשובה בבסיס 10.

|                                                                                                                                                                                                      |                                                                                                                                                   |                      |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|----------------------|
| <pre>push 0 popf mov cx, 0 mov es, cx mov word ptr es:4, offset x mov word ptr es:6, seg x mov bx, 100h a: inc cx inc bx push bx popf cmp cx, 8 jg y sub sp, 6 x: add sp, 6 jmp a y: cx= _____</pre> | <pre>b: xor cx, cx     mov ax, 0aaaaah     inc cx     rol ax, 1     sbb ax, 0     jnz b c: mov cx, 256    inc cl    jns c    jg c cx= _____</pre> | <pre>cx= _____</pre> |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|----------------------|

המשר שאלת מס' 1 בדף הבא

### שאלה מס' 1 (המשן)

- תרגם את הכתובת הלוגית (segment:offset) שלහלן לכתובת פיזית, כשהמונד בזיכרון real.  
רשום את התוצאה בסיס 16.

3b28h:0b52h



- בצע את פעולות החיבור והחיסור שלහלן בשיטת זמשלים ל-2 ברוחב של 16 ביטים.  
כל המספרים נתונים בסיס 16. רשום גם את התוצאה בסיס 16.  
ציין את ערכי הדגליים CF ו-OF בגמר כל פעולה, כפי שהיו נקבעים על ידי ביצוע במעבד 8086X.

|                                                              |                                                              |                                                              |                                                              |
|--------------------------------------------------------------|--------------------------------------------------------------|--------------------------------------------------------------|--------------------------------------------------------------|
| $\begin{array}{r} 5f45 \\ + \\ \underline{c2f3} \end{array}$ | $\begin{array}{r} 5d43 \\ - \\ \underline{b2f3} \end{array}$ | $\begin{array}{r} 1727 \\ + \\ \underline{3a80} \end{array}$ | $\begin{array}{r} 8163 \\ - \\ \underline{1f8d} \end{array}$ |
| CF = _____                                                   | CF = _____                                                   | CF = _____                                                   | CF = _____                                                   |
| OF = _____                                                   | OF = _____                                                   | OF = _____                                                   | OF = _____                                                   |

- תרגם את המספרים שבבלה מבסיס 10 לייצוג סטנדרטי בשיטת הנקודה הצפה בסיס 2.  
רשום את החזקה לא bias (ראה דוגמא).

| Decimal | sign | Exponent | Mantissa |
|---------|------|----------|----------|
| 9.0     | 0    | +3       | 1.001    |
| -44.125 |      |          | 1.       |
| 0.28125 |      |          | 1.       |

- להלן משתנים של תכנית אסטטלי, המכילים ערכים בשיטת הנקודה הצפה.  
תרגם את הערכים לבסיס 10 לייצוג לא חזקה (ראה דוגמא).  
מקורות: בשיטת הנקודה הצפה, שדה החזקה הוא מספר לא סימן הכל bias.

float1 dd c1180000h                                  -9.5

float2 dd bf000000h                                  \_\_\_\_\_

double1 dq 3ff8000000000000h                          \_\_\_\_\_



## שאלה מס' 2 (25 נקודות)

להלן השגורה `compute_list`, הכתובת בשפת C. השגורה מקבלת שני פרמטרים: כתובת של מערך, ומספר האיברים במערך. השגורה מחזירה ערך כלשהו המחשב תוך כדי מעבר על האיברים במערך.

```
extern int compute(char *list, int index, int value);

int compute_list(char *list, int length)
{
 int result = 0;
 int i;
 for (i=0; i<length; i++)
 result = compute(list, i, result);

 return result;
}
```

השגורה `compute_list` ננדחת בקריאה לשגורה `compute`. מהות הערך שמחושב על ידי השגורה השגורה `compute` נקבע על ידי השגורה `compute`. גודאות שונות של `compute` יגדירו חישוב שונה.

להלן גרסה של השגורה `compute`, אשר בעורמתה השגורה `compute_list` מחשבת את האינדקס של האיבר הגדל ביותר במערך. שים לב: אברי המערך הם מספרים בשיטת המפלים ל-2.

הגדעה: האינדקס מוגדר כמו בŞפוח C. במערך המכיל **n** איברים, האינדקס **i** הוא בטווח  $1 \leq i \leq n$ .

```
1 public _compute
2
3 : int compute(char *list, int index, int value)
4 _compute prod near
5 push bp
6 mov bp,sp
7 push bx
8
9 mov bx,[bp+6]
10 add bx,[bp+4]
11 mov al,[bx]
12
13 mov bx,[bp+8]
14 add bx,[bp+4]
15 mov bl,[bx]
16
17 cmp al,bl
18 mov ax,[bp+8]
19 jle done
20 mov ax,[bp+6]
21
22 done: pop bx
23 pop bp
24 ret
25 endp
```

א. מדוע שם השגורה בקטע קוד האסמבלי לעיל הוא `_compute` ולא `compute`?  
למה נחוצה שורה מס' 1 בקוד זה?



## שאלה מס' 2 (המשך)

ב. להלן שלד של תכנית ראשית בשפת אסמבלי.

```
.model small
.stack 100h
.data
mylist db 40, -95, 255, 0, 0FEh, 127, 40h
listlen dw listlen-mylist
returned dw ?
.code
public

extrn
_main: mov ax,@data
 mov ds,ax

 mov ah,4ch
 int 21h
 end _main
```

- i. כתוב בתוך השלד לעיל קטע קוד אשר מבצע קריאה לשגרה **compute\_list** שבדף הקודם, עם המערך **mylist** שמספר האיברים בו הוא **listlen**. את הערך המוחזר מן השגרה יש להציג לתוכה המשנה **returned**. הקפד על כללי תכנות נכון של קריאה לשגרה.
- ii. השלם את החסר במקומות המסתומים בקו בהנחיות **public** ו- **extrn**, כך שנitin יהיה לביצוע אסמבלי של התכנית הראשית, ולאחר כך קישור עם השגרה **compute\_list**.
- iii. מה תוכנו של המשנה **returned** אחרי ביצוע קטע הקוד בסעיף ב לעיל?

ג. עליך לשנות את מימוש השגרה **compute** שבדף הקודם, כך שהשגרה **Sh** תחזיר את האינדקס של האיבר הקטן ביזוֹן במערך. רשום את מספרי השורות בשגרה **Sh** לשנות, ואת קוד האסמבלי החדש של שורות אלה. בצע שינויים הכרחיים בלבד.

קוד אסמבלי חדש      מע' שורה

ד. עליך לשנות את מימוש השגרה **compute** שבדף הקודם, כך שתעבוד על מערך של איבדים מטיפיים **char** במקום **char**. (הערה: השגרה **compute\_list** תעבור לפחות כל שינוי על מערך כזה – למה?) רשום את מספרי השורות בשגרה **Sh** לשנות, ואת קוד האסמבלי החדש של שורות אלה. אם יש צו"ז להוסיף שורות, רשום את מטפוז בצדקה 1, 7.1, 7.2, 1CD. בצע שינויים הכרחיים בלבד.

קוד אסמבלי חדש      מע' שורה



## שאלה מס 2 (המשן)

- ה. בהינתן מימוש השגרה `compute` שבדף 4, כתוב בשפת אסמבלי קטע קוד אשר ממיין את אברי המערך `mylist` בסדר עולה. סגמנט הנתונים מוגדר בדף הקודם.  
חובב להשתמש בולאה ובתוכה קוריאה לשגרה `compute_list`.  
רמן: בכל איטרציה, הקטן את מספר האיברים במערך המועבר לשגרה `compute_list`.

כתוב בשפת אסמבלי גרסה חדשה של השגרה `compute`, כך שהשגרה `compute_list`

תחזיר את סכום האיברים של המערך המועבר אליו כפרמטר.  
שים לב: אברי המערך הם מטיפוס `char`, בעוד הערך המוחזר על ידי `compute` הוא מטיפוס `int`

```
public _compute
; int compute(char *list, int index, int value)
_compute proc near
```



### שאלה מס' 3 (25 נקודות)

הגדר מacro בשם **add** הפעיל בדומה לפקודת **המכונה asp0t**, אך במקום להעתיק את תוכן האיבר במחזורות אל האוגר **ax**, המacro **מחבר** את תוכן האיבר לאוגר (תוצאת החיבור תהיה ב- **ax**) בגמר ביצוע המacro, ערכיו דגלי התנאים **טייביט** להתחאים לתוצאות החיבור (כמו אחרי פקודת **add**).  
**אסוך** לmacro לשנות את האוגרים ( מלבד האוגרים הרלוונטיים לפעולה שמתבצעת).  
**רמז:** העזר בפקודה **lods0t** ובמחסנית.

הגדר מacro בשם **pz**, שמקבל פרמטר בודד **dest**, זהה לאופרנד של פקודת הסתעפות מותנית.  
הmacro **pz** מבצע הסתעפות אל **dest** אם ורק אם הדגל **zf** באוגר הדגלים **לינק** (כלומר ערכו 1).  
**אסוך** לmacro לגירוט **להתופעות** לוואי, כגון שינוי במבנה של הדגלים או של האוגרים.

**macro dest pz**



הגדר מacro בשם **pnj**, שמקבל פרמטר בודד **dest**, זהה לפרמטר של המacro **pz** משיעף ב-.  
הmacro **pnj** מבצע הסתעפות אל **dest** אם ורק אם הדגל **zf** באוגר הדגלים **ביבוי** (כלומר 0).  
**חוובן** להשתמש בקריאה לmacro **pz**. **אסוך** לmacro **pnj** לגירוט **להתופעות** לוואי.

**macro dest pnj**



### שאלה מס' 3 (המשך)

ד. להלן מacro בשם **callp** המבצע קריאה לשגרה. הפרמטר הראשון של המacro הוא כתובת השגרה. ארבעת הפרמטרים הנוספים של המacro הם פרמטרים עברו השגרה, אשר יועברו אליו במחסנית. המacro מאפשר לקרו לשגרה שמקבלת בין 0 ל-4 פרמטרים במחסנית. אחרי החזרה מהשגרה, המacro דואג לנוקות את המחסנית. הנוכח כי כל הפרמטרים לשגרה הם בגודל מילה.

```
callp macro fname,p1,p2,p3,p4
 nump=0
 itp p,<&p4,&p3,&p2,&p1>
 ifnb <p>
 push p
 nump=nump+1
 endif
 endm
 call fname
 if nump ne 0
 add sp,nump*2
 endif
endm
```

i. להלן שלוש דוגמאות של קריאות למacro **callp**. לכל דוגמא, רשום את הקוד בשפת אסטראלי שמתקבל לאחר פריסת הקריאה למacro. יש לרשום רק שורות שיוצרות קוד מכוונה. אין לרשום הנחיות לאסטראלי מותנה, וכו'.

| callp f1,x,ax,[bx],100 | callp f2,y,,z | callp f3 |
|------------------------|---------------|----------|
|                        |               | )        |

ii. למה משמש משתנה האסטראלי **nump**?

הגדיר מקרו בשם **callp**, המקבל פרמטרים זהים לאלו של המacro **call** מסעיף ד', ופועל באופן דומה, פרט להבדל הבא: אם הוגדר קבוע אסטראלי בשם **ревז** (למשל על ידי האופציה **Drevz**/של **tasm**), המacro **callp** דוחף את הפרמטרים של השגרה למחסנית משמאלי ימינו (כלומר, **ревз** נדחף ראשוון); אחות, הפרמטרים נדחפים מימין לשמאלי. חובב להשתמש בקריאות למacro **callp**.

```
callp macro fname,p1,p2,p3,p4
```



### שאלה מס' 3 (המשך)

הגדיר מקרו בשם **dbw**, שמקבל שני פרמטרים: הראשון הוא סמל **name**, והשני הוא קבוע **n**.  
הmacro מגדיר בזכרון משתנה בשם **name** שמאוחתול לערך **n**. גודל הזכרון שתווסף המשטנה הוא  
**הקטן ביותר** (בית או מילה) שיכולים להכיל את הערך **n** **בשיטת המשלים** ל-2 (כלומר מבלי שיתהפר  
הstein המקורי של **n** כתוצאה מהacuson בזכרון). **בונח כי n לא גולש מגודל מילה.**

דוגמאות:

|                     |                                                                       |
|---------------------|-----------------------------------------------------------------------|
| <b>var1 db -15</b>  | תפרק את הקוד <b>dbw var1,-15</b>                                      |
| <b>(var2 db 200</b> | תפרק את הקוד <b>dbw var2,200</b> ( <u>שים לב</u> : לא 200 <b>db</b> ) |
| <b>var3 dw 80h</b>  | תפרק את הקוד <b>dbw var3,80h</b>                                      |

**dbw macro name,n**

בהתנזה וזוודה שلن למקרו **dbw** מסעיף 1/, רשום את קוד האסטטלי שנפוץ בכל אחת מהקריאהות הבאות לmacro. לכל קריאה, ציין האם תיווצר בעיה, ואם כן הסבר מה הבעיה.

i. **dbw var4,35000**

ii. **dbw ,500**



ארגון המחשב ושפט ספ (203.1130)

סמסטר ב', תשס"ג  
בחינה סופית - מועד א'

הוראות לנבחן:

- משך הבחינה שלוש שעות.
- מותר להשתמש בכל חומר עוזר, למעט מחשבים ומחשובונים מכל סוג.
- יש להסבירNeil כל השאלות.
- יש לרשום את התשובות בגוף השאלון במקומות המזועדים לכך.
- נא לכתוב בכתב יד ברור ונקי. מומלץ להשתמש בעפרון ומתק.
- בשאלון זה 15 דפים, כולל דף זה. וזה כי כל הדפים נמצאים.

ב ה צ ל ח ה !

| ניקוד | צiron |        |
|-------|-------|--------|
| 25    | 13    | שאלת 1 |
| 25    | 20    | שאלת 2 |
| 25    | 12    | שאלת 3 |
| 25    | 19    | שאלת 4 |
| 100   | 64    | סה"כ   |



### שאלה מס' 3 (המשך)

ה. נגיד זוג מacro-ים: whil ו- endwhil המשמשים לכתיבה לולאה בשפת אסטבלי שדומה למבנה הבא בשפת C.

```
while (x cond y) { . . . };
 {
 whil macro id,x,cond,y
 nextw&id: cmp x,y
 j&cond dow&id
 jmp endw&id
 dow&id:
 endm
```

להלן הגדרות המacro-ים.

```
endwhil macro id
 jmp nextw&id
 endw&id:
endm
```

להלן שלוש דוגמאות של לולאות המשמשות במacro-ים לעיל. לכל דוגמא, רשום את הקוד בשפת אסטבלי שמתפרק לאחר פרישת כל הקיימות למacro. אין לרשום את כוורות המacro-ים.

|                                                                                                                                                                                                |                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                        |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>xor si,si whil 1,si,nge,len3     mov str3[si],'     inc si endwhil 1</pre>                                                                                                                | <pre>xor ax,ax xor bx,bx whil 2,ax,mp,100     add ax,bx     inc bx endwhil 2</pre>                                                                      | <pre>xor si,si whil 3,si,b,len1     mov match[si],0     mov al,str1[si]     xor di,di     whil 4,di,b,len2         cmp al,str2[di]         jne skip         inc match[si]         skip: inc di     endwhil 4     inc si endwhil 3</pre>                                                                                                                                |
| <p style="text-align: center;">xor si,si</p> <pre>nextw1: cmp si,1en3         jnge down1         jmp nextw1 down1: mov str3[si],`down;add ax,bx         inc bx         jmp nextw2 endw1:</pre> | <p style="text-align: center;">xor ax,ax<br/>xor bx,bx</p> <pre>nextw2: cmp ax,100         jump down2         jmp endw2 down2: jump nextw3 endw2:</pre> | <p style="text-align: center;">xor si,si</p> <pre>next3:         cmp si,1en1         jb down3         jmp endw3 down3: mov match[si],0         mov al,str1[si]         xor di,di nextw4: cmp di,1en2         jlo down4         jmp endw4 down4: jump al,str2[di]         jne skip         inc match[si] skip: inc di         jmp nextw4 endw4: inc si jmp nextw3</pre> |

המשך שאלה מס' 3 בדף הבא

### שאלות מס' 3 (המשך)

1. מהו תפקידו של הפורטර `pi` במקרו-ים `whil` ו- `endwhil` מסעיף ה'?

✓ *labels* כנקודות  $\rightarrow$  נספוחות

2. איך בעה קיימת בקוד שנפרש בדוגמה האמצעית בסעיף ה'?

*add x,y*

↙  
↙ *לינק יוניק לינק כבוי, מוקף*

3. בעדרת המקרו-ים מסעיף ה', נגיד זוג מקרו-ים: `do` ו- `enddo`, המשמשים לכתיבה לולאה בשפת אסמבלי שדומה למבנה הבא בשפת C.

`do { . . . } while (x cond y);`

להלן הגדרת המקרו `enddo`.

`enddo macro id  
    endwhil id  
endm`

השלט להלן את הגדרת המקרו `do` במקומות המוטוונים בקוו. אין להוסיף פרמטרים או שורות.

✓  
`do macro id,x,cond,y  
    jmp loop_label  
    whil id,x,cond,y  
endm`



שאלה מס' 4 (25 נקודות)

ולכן

בגלל טעות ביצור, יצא לשוק סדרה פגומה של מעבדי 86X, בהן לא מוגדרת הפקודה pushf (אשר קוד המכוונה שלה הוא הבית הבוזץ 9ch).

כיזען, כאשר המעבד מנסה לבצע פקודת מכונה שאינה מוגדרת, נוצרת חריגה "illegal opcode", שמשמעותה היא 6. הכתובת שנכנסה למחסנית בעת המעבר לשגרת השירות של חריגה 6 היא זו של הפקודה שגרמה לחריגת.

כדי להתגבר על הפגם במעבד, הוחלט לשככל את שגרת השירות של חריגה 6 כדלקמן. אם הפקודה שגרמה לחריגת pushf, שגרת השירות תבצע סימולציה של פקודת זו. אין שינוי בטיפול בשאר המקרים של חריגת זו.

להלן שגרת השירות החדשה של פסיקה 6, המממשת שכלו זה. השגרה מתחליה בתוית oldInt06. הנה כי וקטור הפסיקה של שגרת השירות המקורי נשמד במשתנה oldInt06 שמודגד בסגמנט הקוד. כמו כן, הנה כי רוחב אוגר הדגלים הוא תמיד 16 ביטים.

```
1 oldInt06 dd ?

2 newInt06: sub sp,2
3 push bp
4 mov bp,sp
5 push bx
6 push es

7 les bx,[bp+4]
8 cmp byte ptr es:[bx],9ch
9 je pushFlags

10 pop es
11 pop bx
12 pop bp
13 add sp,2

14 jmp oldInt06

15 pushFlags: mov bx,[bp+8]
16 xchg bx,[bp+6]
17 xchg bx,[bp+4]
18 mov [bp+2],bx

19 inc word ptr [bp+2]

20 pop es
21 pop bx
22 pop bp
23 iret
```

אליה מס' 4 (המשנה)

עליך להסביר כיצד פונלאת שגרת השירות החדש של חירגה 6, שבז'ף הקודם, השאלות שלהן ינחו אוטר בהבנת הקוד. מומלץ לעבור על כל השאלות מראש. שים לב: על ההסבירים לתייחס למחאות הפעולות המתבצעות, ולא רק לצטט במיללים את שמות הפקדות והօפרודיות.

- i. מה שושות שורות 9-7? באילו מקרים מתרכנת ההסתעפות בשורה 9?

השורה הינה כזו שמשתמשה בטבלת הסדר וטבלת הסדר בטבלת הסדר.

ii. מה עשו שורה 14? כיצד חוזרים משגרת החירגה כאשר מתרכנת שורה זו? הנו מודעים לכך ש

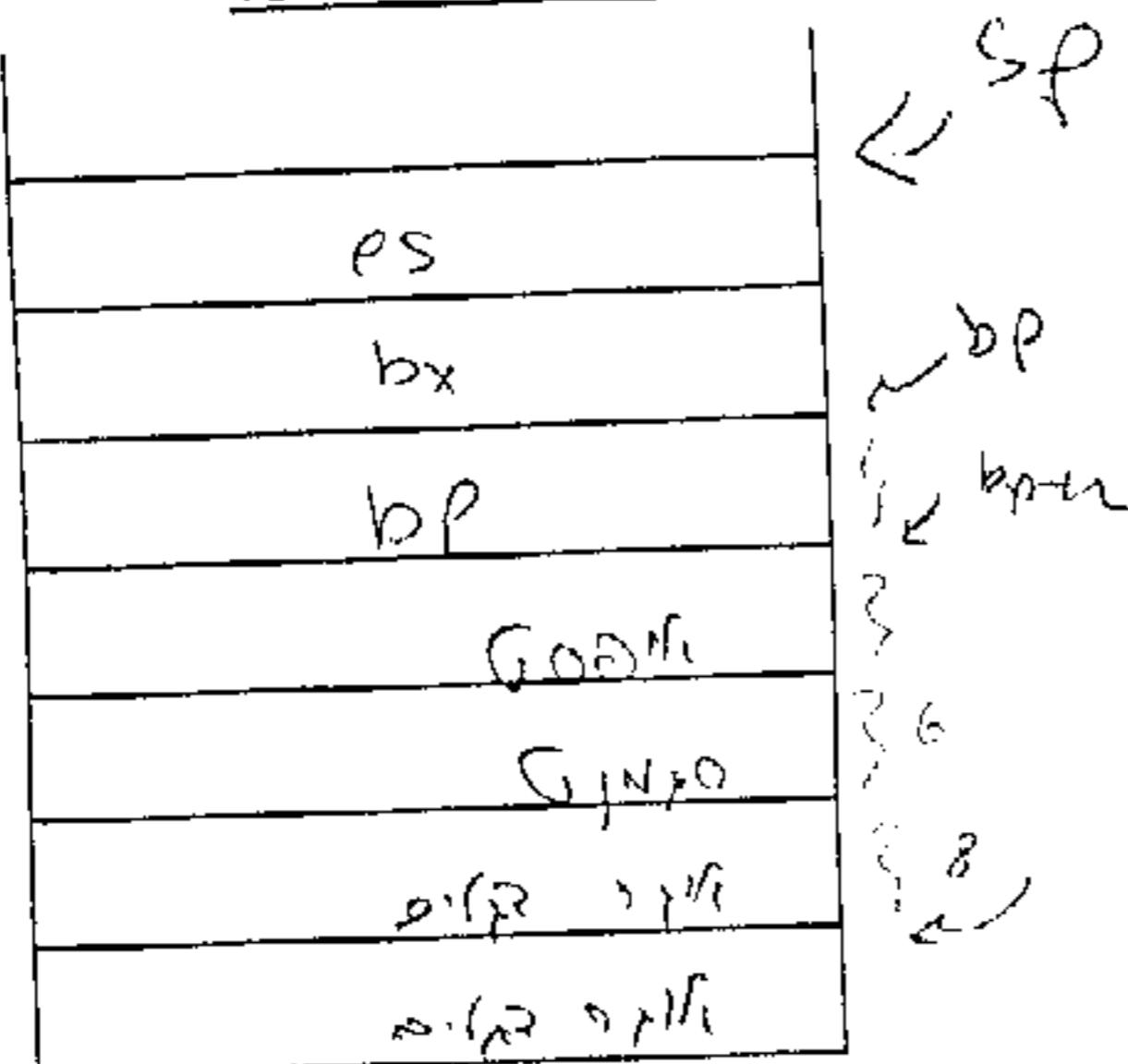
|           |           |           |
|-----------|-----------|-----------|
| טבלת הסדר | טבלת הסדר | טבלת הסדר |
| טבלת הסדר | טבלת הסדר | טבלת הסדר |
| טבלת הסדר | טבלת הסדר | טבלת הסדר |

 דוחה את השורה.

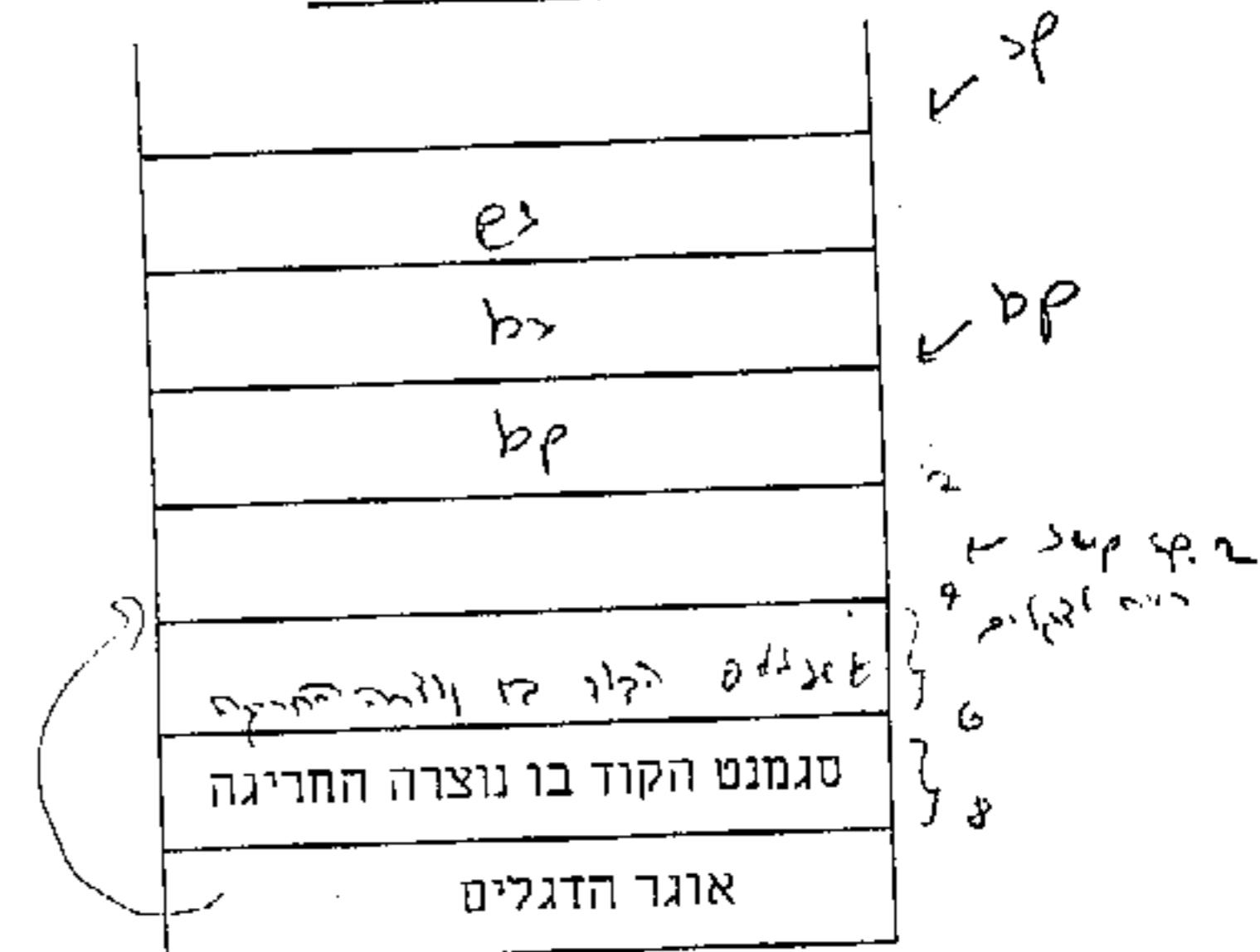
iii. דוחה השורה ומי יתאפשר?

iv. במחסנית הוא בגודל מילה. ציריך גם לאיזה איבר במחסנית מצבייעים האוגרים sp-1-dp.

בגרות ביצוע שורה 18



בתחילה ביצוע שורה 15



- iv. מה תפקידה של פורה ? מה עשוות שורות 15-18 ?

וְיַעֲשֵׂה יְהוָה כָּל־אָמִרָתֶךָ וְיַעֲשֵׂה יְהוָה כָּל־אָמִרָתֶךָ

15-18 mm  
tail 10-12 mm

7. מה נושא שורה 19?

offset > IP > AF ARP

<sup>vi</sup> מה נמצא בראש המחסנית לאחר ביצוע שורה 23?

$$= \sqrt{2 - \gamma^2}$$

### שאליה מס' 4 (המשך)

בשורה 8 בקוד שגרת השרות החדשה, האם השימוש ב-casting בעבור האופרנד הראשון הכרחי? הסבר.

לכט ב- 16. 10. 1945  
ב- 16. 10. 1945  
ב- 16. 10. 1945

מודיע מוגדר המשתנה `old_ip` דזוקא בסגמנט הקוד? כיצד, אם בכלל, ניתן להגדיר משתנה זה בסגמנט הנחותיים? הסבר.

دیگر اینها را نمی‌توانند میلیون‌ها کسری داشتند و این از این‌جا شروع شد.

מסתבר כי במעבדים מהסדרה הפקודה `pop` (אשר קוד המכוונה שלה הוא `hbuzzd 4p9`). עליך להוסיף לשגרה החדשת של חריגה 6 קוד אשר יבצע סימולציה של הפקודה `pop`, בדומה לSIMOLציה של הפקודה `pushf`.

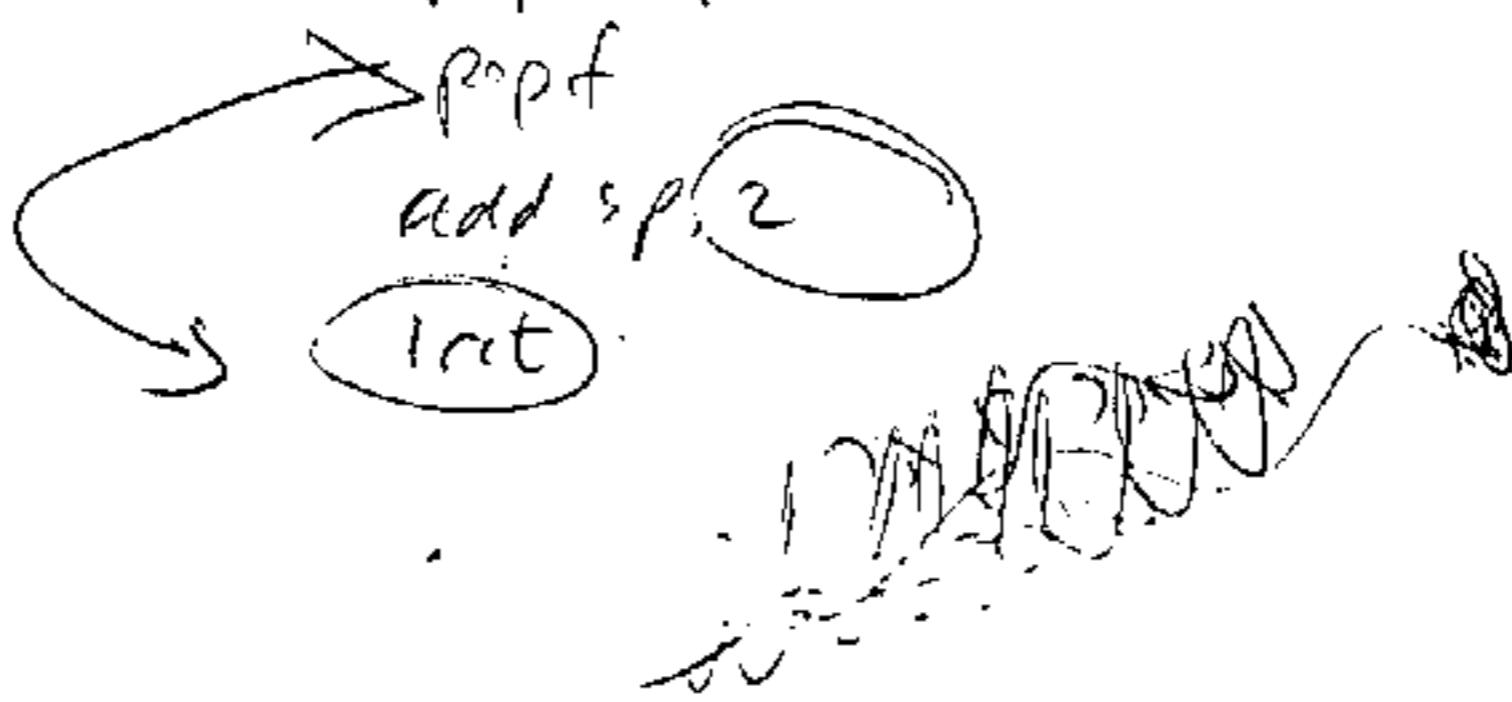
רשום להלן את התוספות לקוד האסטבלי של שגרת החירגה. עליך למספר את השורות הנוספות בהתאם למספר השורות הקיימות. לדוגמא: שורות שייתווסףו אחרי שורה 9 ימוספרו 9.1, 9.2 וכו''). אם יש צורך לשנות או למחוק שורה קיימת, רשום את מספר השורה ואת קוד האסטבלי החדש שלה.

9.1 cmp byte ptr es:[bx], al  
9.2 je RopFlags

24 RopFlags:

```
mov bx, [bp+8] } ; bx = flags
;
xchq bx, [bp+23] } ; bx = space
;
xchq bx, [bp+67]
xchq bx, [bp+4]
inc word ptr [bp+12]
pop es
pop bx
pop bp
```

bx → C  
bx → N  
bx → P



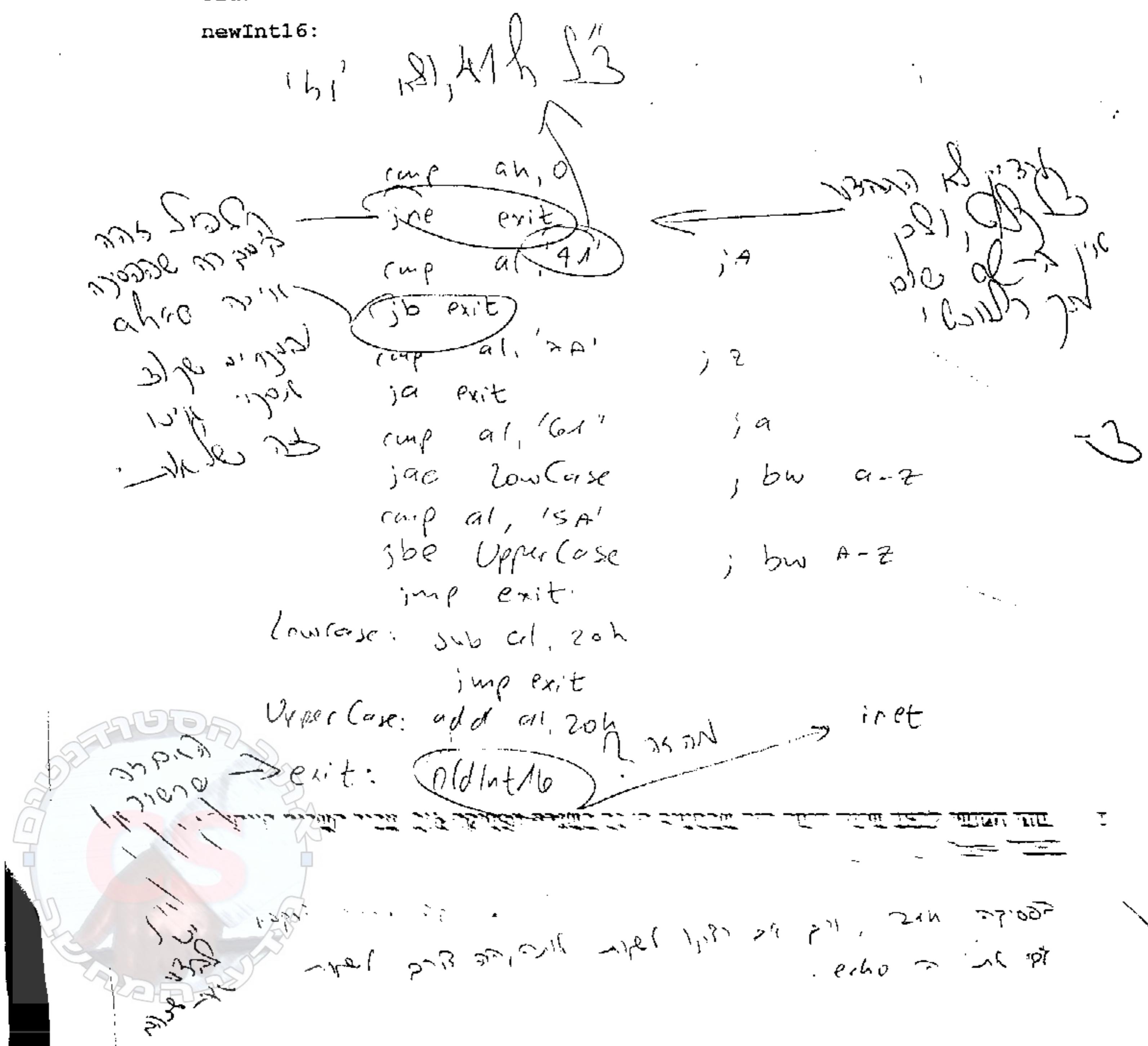
כידוע, פסיקת שירותים המקלדת ah 16h מספקת, בין השאר, את השירות הבא: כאשר ah=0, מתחבצתה הוצאה של התו הבא מחוץ למקלדת; קוד ה-ascii של התו מוחדר באוגר ah, ואילו קוד ה-scan מוחדר לאונגר ah.

עליך לכתוב שגרת שרות חדשה לפסיקה 16h, שבה השדות עבור ah ישנה כדלקמן. אם התו שהוקלד ונכנס לחוצץ המקלדת הוא אות אנגלית קטנה (lowercase), יוחזר באורך al קוד ה-ascii של האות הגדולה (uppercase) המקבילה. ולהפוך, אם התו בחוצץ המקלדת הוא אות גדולה, יוחזר באורך al קוד ה-ascii של האות הקטנה המקבילה. עבור כל TWO אחד, יוחזר באורך al קוד ה-ascii המקורי שבחוצץ המקלדת. קוד ה-scan יוחזר באורך ah, ללא שינוי מהשורות המקורי. אין שינוי בשאר השירוטים של פסיקה 16.

הנ"ח כי טבלת קודי ה-ascii שבתוכף היא זו שבחוברת הקודס (עמ' 337-340 במחודשה הרביעית).  
שגרת השרות החדשה מתחילה בתוויות newInt16. הנו כ"י וקטוור הפסיקה של שגרת השירות  
המקורית נשמר במשתנה oldInt16 שמוגדר בסעמנט הצעדי.

oldInt16 dd ?

newInt16::



### שאלה מס' 1 (25 נקודות)

נתון שדגל הסימן דלוק (1=ז), והאוגר ax מכיל ערך שלילי. לבדוק כל אחת מהפקודות שלהן, ציין מתי היא גורמת לכיבוי דגל הסימן: תמיד, לפעמים בז' ולפעמים לא, או אף פעם. הקפ ברגול את התשובות הנכונות.

תמיד / לפעמים / אף פעם

sar ax,1  
sal ax,1  
shr ax,1  
add ax,ax  
neg ax  
or ax,bx  
mov ax,0  
sbb ax,ax

$\frac{-1}{2}$

ב. לגבי האסטבלר, הקשר (linker) והטען (loader), אילו מהطنנות שלהן נכוןות? תracן יותר מתשובה אחת נכונה. הקפ ברגול את כל התשובות הנכונות.

- i. הקשר מכין מידע שיישמש את הטען.
- ii. האסטבלר מכין מידע שיישמש את הקשר אך לא את הטען.
- iii. האסטבלר מכין מידע שיישמש גם את הקשר וגם את הטען.
- iv. הטען אינו מסוגל לתפקידו אלא מידע שהוכן על ידי האסטבלר.

$\frac{1}{2}$

ג. הכרנו את כל התוכנות הבא ונבוד שפת אסטבלר: כאשר קודאים לשגרה עם פרמטרים במחסנית, המיקום הקורא הוא האחראי להזאת הפרמטרים מן המחסנית לאחר החזרה מן השגרה. אילו מהطنנות שלהן נכוןות לגבי כל זה? תracן יותר מתשובה אחת נכונה. הקפ ברגול את כל התשובות הנכונות.

- i. הכלל הכרחי תמיד, כי השגרה אינה מסוגלת להוציא את הפרמטרים מן המחסנית.
- ii. הכלל משמש לצורך קישור עם תכניות הבתובות בשפה עילית.
- iii. הכלל מאפשר לתחת לתוכנית מבנה מסוית ומודולרי.
- iv. הכלל הכרחי כאשר משתמש בשגרה שמקבלת מספר משתנה של פרמטרים במחסנית.
- v. הכלל סיותר כאשר גם השגרה וגם המיקום הקורא בתוכים בשפת אסטבלר.

$\frac{1}{2}$

ד. עבור כל אחד משלש קטעי הקוד שלהן, דשומ את תוכנו של האוגר ax בגמר ביצוע הקטע. רשום את התשובה בסיסים 10.

```
.data
o list db 5,4,3,2,1,0
6 count dw count-list-1
```

```
.code
x: xor dx,dx
 xor bx,bx
 mov cx,word ptr list[bx]
a: inc dx
 loop a
 inc bx
 cmp bx,count
 js x
```

dx = 5 X  
-3

```
b:
 xor ah,ah
 inc dx
 inc ah
 sahf
 jns b
 jnz b
 jnc b
```

start ah into flgs

dx = 2

```
c:
 inc dx
 adc ax,1
 sar ax,1
 jb c
```

dx = 5 X  
-2

dx = 0  
0000/0000/0000/0000

המבחן שאלה מס' 1 בדף הבא

אתר הסטודנטים – החוג למדעי המחשב, אוניברסיטת חיפה

### עליה מס' 1 (המשך)

תרגם את הכתובות הלוגיות (segment:offset) שלහן לכתובות פיזית, כשהמניבד במצב real.  
רשום את התוצאה בבסיס 16.

$$\begin{array}{r} 11 \\ 314e0 \\ - cdef \\ \hline 1cf \end{array}$$

314eh: 0cdefh

|   |   |   |   |   |
|---|---|---|---|---|
| 3 | e | 1 | c | f |
|---|---|---|---|---|



10111010

11100

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

1101

110

## שאלות מס' 2 (25 נקודות)

א. הפונקציה הרקורסיבית ( $q(bx, cx)$ ) מוגדרת להלן על ידי אלגוריתם בשפה C.

```

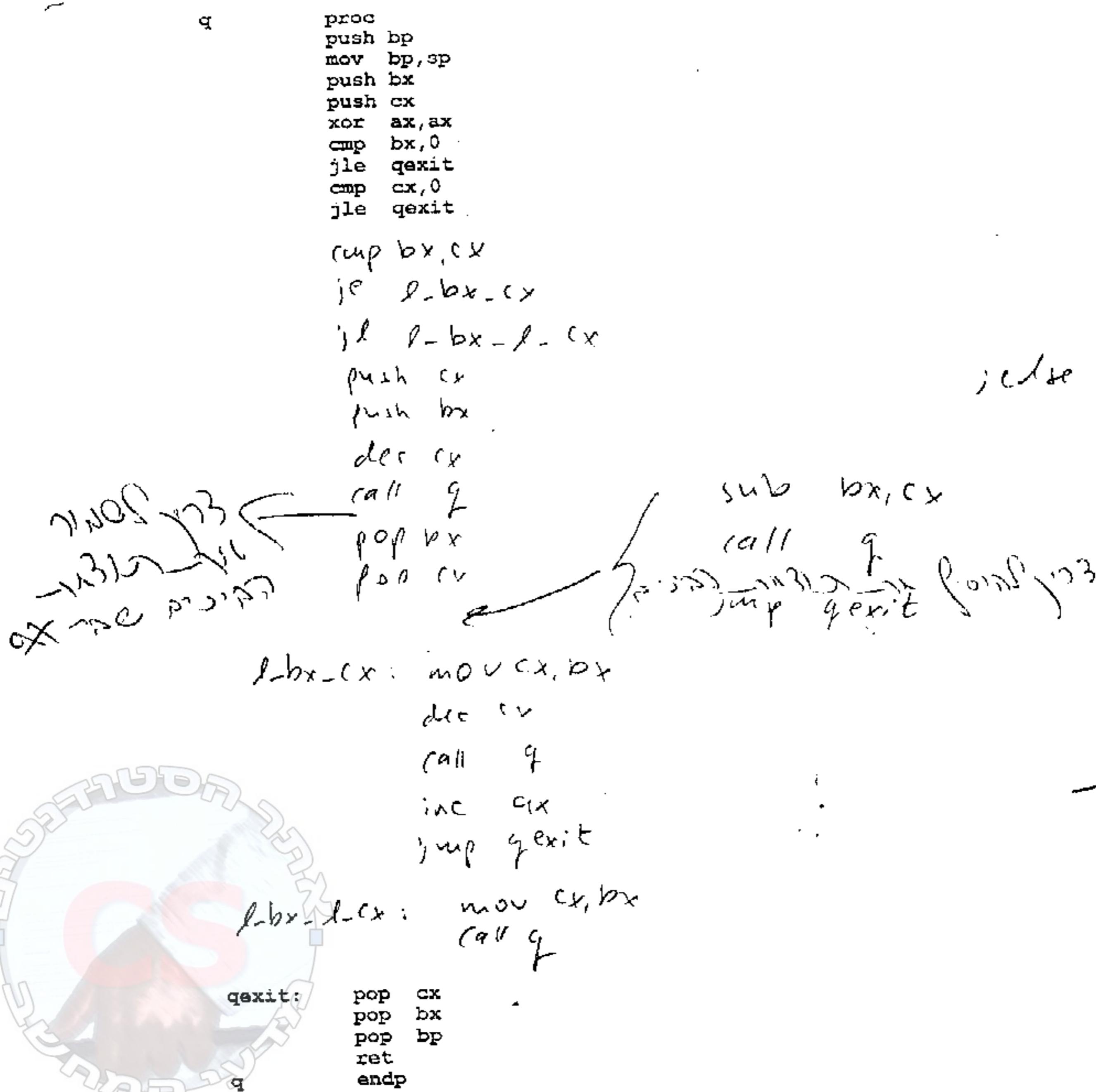
int q(int bx, int cx)
{
 if ((bx<=0) || (cx<=0)) return 0;
 else if (bx==cx) return q(bx,bx-1)+1;
 else if (bx<cx) return q(bx,bx);
 else return q(bx,cx-1)+q(bx-cx,cx);
}

```

עליך למסח את הפונקציה  $q$  בשפה אסמבילר, בהתאם לדרישות הבאות:

- המימוש בנוורת שגרה אחת בלבד, בשים  $q$ . אין להשתמש בשגרות פנימיות ונספות.
- הפרמטרים  $bx$  ו-  $cx$  מועברים באוגרים  $ax$  ו-  $cx$  בהתאם. התוצאה מוחזרת באוגר  $ax$ .
- מותר להשתמש באוגרים  $ax$ ,  $bx$ ,  $cx$ ,  $bp$  ו-  $sp$  בלבד. אין להשתמש באוגרים אחרים.
- אסור להשתמש במשתנים בזיכרון.
- מותר להשתמש במחסנית אנדרט לצורך שימושה ושבור של אוגרים.

גנום כי לא תהיה גלישה מהאגרים במשך החישוב.  
השלט את קוד השגרה  $q$  בתוך השלד שהלן.



המשר שאלת מס' 2 בדף הבא

### שאלות מס' 2 (המשר)

ב. כתוב שגורה בשם `cube`, שמקבלת במחסנית מספר  $x$  מטיפוס ממשי, המיציג בשיטת הנקודה הצפה ברוחב 64 ביטים. השגורה מחזירה באורך(0) ST את הערך  $x^3$  (החזקת השלישי של  $x$ ).

```

cube proc
 push bp
 mov bp, sp
 fld dword ptr [bp+4]
 fld dword ptr [bp+4]
 fmul st(1), st(0)
 fmulp st(1), st(0)
 pop bp
 ret
cube endp

```

ג. בסגמנט הנתונים מוגדרים שני משתנים, `num`-`res`, שנייהם מטיפוס ממשי בנקודה צפה ברוחב 64 ביטים. כתוב קוד הקודא לשגורה `cube` עם הפרמטר `num`, ומציב את הערך המוחזר טן השגורה לתוך המשתנה `res`. הקפד על כללי התוכנות הנכון של קדיאת השגורה.

```

.data
num dq 123.456
res dq ?
.code
 push num
 call cube
 add esp, 8
 fstp res
}

```



המשר שאלות מס' 2 בדף הבא

### שאלה מס 2 (המשג')

**הגדלה:** במחוזות חווים, "המקום הנבחר" הוא הבית הראשון (מצד שמאל) במחוזות שבכיל את הטע רוח (בלומר את הערך 40 בקוד ascii).

להלן נחתה השגרה `findSpace`. השגרה מקבלת כפרמטר באוגרים `di:es` כתובות לוגית מלאה (סמנט ויחסט) של מחרוזת תווים. אורך המחרוזת מועבר כפרמטר באוגר `cx`, ביצוג ללא סימן.

השגרה מוחפשת את המיקום הנבחר בנסיבות התווים שהועברה כפרמטר. אם המיקום הנבחר נמצא, השגרה מוחזירה באוגרים `ip:es` את הכתובת הלוגית המלאה שלו, ובאוגר `ex` את האורך הנותר של המחרוזת (כלומר, החל מון המיקום הנבחר ועד סוף המחרוזת המקורית, כולל שני הקצוות). אם לא נמצא במחרוזת המיקום הנבחר, השגרה מוחזירה באוגר `ex` את השunk והורסת את תוכו האוגר `ip`.

```
1 findSpace: push ax
2 xor al,al
3 add al,20h
4 cld
5 repne scasb
6 jne endSpace
7 inc cx
8 dec di
9 endSpace: pop ax
10 ret
```

- השגרה `findSpace` מחזירה באורך  $x$  את הערך 0 אם ורק אם המקום הנבחר אינו נמצא במחוזות. הסבר כיצד השגרה מקיימת טענה זו.

מה קורה כאשר המקום הנבחר הוא הבית האחרון במחוזות? מה קורה כאשר השגרה מקבלת מחוזות באורך 0?

(DRAFT OF ANSWER) (x) is 0 and is enough for both S and T to hold

לפניהם, מילויים נוראים (במידה ונתנו איזה איז)  $\mathcal{Z}_F = 0$  !  
ולפניהם  $0 = \mathcal{C}_X$ , כלומר (ויש)  $\mathcal{Z}_F = 0$  מה שטף. וזה מוכיח  
ההיפזה ( $X$  מילוי איזוטרוני) (במילים אחרות) שקיים מילוי איזוטרוני ב- $X$ .

כחות בשפת אסמבלי שגורה בשם `findNotSpace`; הזזה בהגדרתה לשגרה `findSpace`, מלבד הבדל אחד: "המקום הנבחר" אותו מ Chapman השגורה מוגדר כבית הראשון במחזורת שאין מכיל את התו רווח.

רמז: נדרשים שינויים קלים בלבד בשגרה `findSpace`.

findspace : push ax  
21061 523  
2431 44  
31203 714  
14211 522  
333  
mid nor al,al  
add al,20  
ld  
→ repe scasb  
je. jne endspace  
inc cx  
dec di  
outspace: pop ax  
ret

המבחן שאלת מס' 2 בדף הבא

## אלגוריתם מס' 2 (המשר)

הגדרה: במחזורת תווים, "אסימון" (token) הוא תותח מחרצת מקטימאלית שאיננה מכילה אף תו רווח.

לדוגמה: במחזרות "Hello World" שני אסימונים: "Hello" ו- "World". השגרה `numTokens` מקבלת במחסנית שני פרמטרים: כתוות לוגית מלאה (סגמנט והיסט) של מחזרות תווים, ואורך המחרצת (בגודל מילה בייצוג ללא סימן). השגרה מחזירה באורך ax את מספר האסימונים במחזרות.

כתבו להלן את השגרה `numTokens` בשפת אסמבלי. חובה לשימוש בקライאות לשגרות `findSpace` ו- `findNotSpace`. הקפז על כללי התכונות הנכון של שגרה.

```

numTokens proc
 push bp
 mov bp, sp
 push di
 push es
 xor ax, ax
 mov ax, [bp+6]
 mov es, ax
 mov di, [bp+4]
 mov cx, [bp+8]

 l1: push di
 call findSpace
 pop si
 cmp di, si
 je equal
 inc cx

equal: call findNotSpace
 pop bp
 ret
endp.

```

כתב בשפת אסמבלי שגרה בשם `argc`, ללא פרמטרים, שמחזירה באורך ax את מספר האסימונים שבשורת הפקודה של DOS שהפעילה את התכנית, לא כולל שם הפקודה עצמה.

חובה לשימוש בקライאה לשגרה `numTokens` מסעיף ד'. חובה לשימוש בקライאה לשגרה `stam`.

לדוגמה: שורת הפקודה שלהן מפעילה את התכנית `stam`, שמבצעת קריאה לשגרה `argc`. הערך שמחזר על ידי `argc` בדוגמה זו הוא 6.

C:\>stam yom shel hol im boker kahol

זכורת: המחרצת של שורת הפקודה (לא כולל שם הפקודה) נמצאת בסגמנט ה-PSP, החל מהיישט 48. אורך המחרצות נמצא במבנה שבהיישט 40h. הנץ כי כתוות התחילת סגמנט ה-PSP נמצאת באורך es.

```

argc proc
 xor ax, ax
 mov al, 40h ; המץ אלט
 xor ah, ah
 cmp ax, 0 ; המץ אונט
 je exit
 push ax ; המץ אונט
 push es ; המץ אונט
 call numTokens ; המץ אונט
 add sp, 4 ; המץ אונט
 pop es ; המץ אונט
 pop ax ; המץ אונט
 je exit
 dec ax ; המץ אונט
 ret ; המץ אונט
endp.

```

כתב קטע קצר בשפת אסמבלי, שקורא לשגרה `argc` מסעיף ה', ולאחר מכן מדפיס בצורה נאה את התוצאה המוחזרת מהשגרה, וזאת בעזרת קריאה לשגרה `printf` של שפת C. הגדר בסגמנט הנתוניים את מחרוזת ההדפסה המועברת ל-`printf`.

```

printf str 0x100, " %d\n"
call argc
add sp, 4
push ax
mov offset print-str, ax

```

### שאלה מס' 3 (25 נקודות)

א. בידוע, פקודת המבונת `pop` שולפת איבר מראש מהנסנית, ומעבירה אותו לאופרנד של הפוקודה. אין אפשרויות, באמצעות הפוקודה `pop`, לשולף איבר מראש מהנסנית ופושט לזרוק אותו (דוחינו לא להעביר אותו לאף מקום).

עלין להגדיר מאקרו בשם `popp`, המקבל פרמטר בוודד אופציונאלי זהה לאופרנד של פקודת המבונת `pop`. כאשר מתחבצנת קדיאה למאקרו הכללת פרמטר, המאקרו פועל באופן זהה לפוקודה `pop`. ואילו כאשר הקריאה היא ללא פרמטר, המאקרו שולף מראש מהנסנית מילה אחת וזרק אותה. המאקרו `popp` אינו משפייע על הדגלים, וזאת בדיק בפי שמתנהגת פקודת המבונת `pop`. יש להימנע מתחזות לוויאי, כגון שינוי האוגרים (פרט לאוגר `sp`).

(-3)

```

popp ax ax
 if b < x >
 pop x
 exitm
 end if
 mov [ip] ax
 pop ax 1186
 endm

```

ב. הגדר מאקרו בשם `puship`, ללא פרמטרים, הדוחף למשונית את תוכן האונגר `ip` (instruction pointer) כפי שהוא בתחילת ביצוע המאקרו (כלומר, היחסט אל הבית הראשון בקוד המבונת הנפרש על ידי המאקרו). על המאקרו לפרק פקודת מבונת אחד בלבד. המאקרו אינו משפייע על הדגלים.  
הערה: ראה גם סעיף ג' להלן.

(-2)

```

puship mov ip
 mov [ip] 1186
 push ip
 endm

```

ג. הגדר מאקרו בשם `akipuship`, ללא פרמטרים, הדוחף למשונית את תוכן האונגר `ip` כפי שהוא בגמר ביצוע המאקרו (כלומר היחסט אל הבית הראשון שאחרי קוד המאקרו). על המאקרו לפרק פקודת מבונת אחד בלבד, השונה מפקודת המבונת שפורש המאקרו `puship` מסעיף ב'. המאקרו אינו משפייע על הדגלים.

(-2)

```

akipuship mov ip
 mov ip, [sp+12]
 push ip
 endm

```

### שאלה מס' 3 (המשר)

ד. הגדר מacro בשם pivot, שכותרתו מוגנה להלן.

pivot macro p,n1,n2,n3,n4,n5,n6,n7,n8

כל הפרמטרים לmacro הם קבועים מספריים שלמים בגודל מילה. יתכן וחלק מהפרמטרים n8-n1  
לא מוגבר בהדריה כלשהי לmacro.

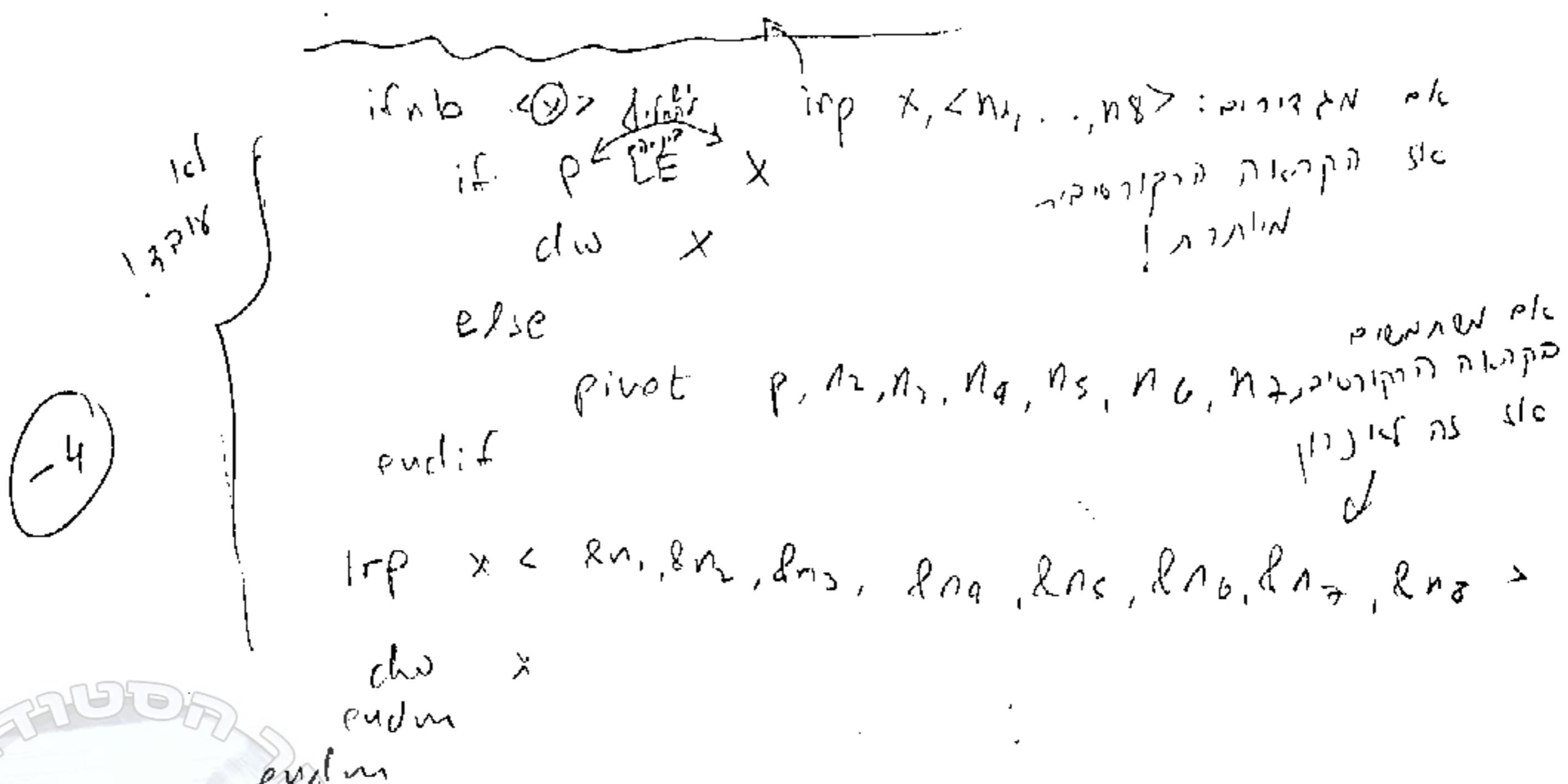
macro מקצה בזיכרון רצף של מילים, כמספר הפרמטרים 8-n1 שהונברדו בפועל בקריאה.  
המילים הראשונות ברצף מכילות את אלה מהפרמטרים 8-n1 שנדבכם קשוו או שווה לערך ק  
(לפי סדר ההופעה של ערכיהם בקריאה לmacro). המילים הבאות ברצף מכילות את כל  
הפרמטרים הנוגדים מ בין 8-n1 (גם כאן לפי סדר הערכים בקריאה לmacro).

לדוגמא, הקריאה: 50,100,100h,100,50,-100,231,pivot תפרק את הקוד שללן.

```
dw 50
dw -100
dw 100
dw 50
dw 231
dw 100h
```

יש לבנות את המacro באופן יעיל, תוך שימוש במבני אסםבי מוגנה מתאימים.

pivot macro p,n1,n2,n3,n4,n5,n6,n7,n8



ארגון המחשב ושפה ספ (203.1130)  
סמסטר ב', תשס"ג  
בחינה סופית - מועד ב'

הוראות לנבחן:

- משך הבחינה שלוש שעות.
- מותר להשתמש בכל חומר עוזר,elman מחשבים ומחשבוניים מכל סוג.
- יש להסביר על כל האשאלות.
- יש לרשום את התשובות בגין השאלון במקומות המיועדים לכך.
- נא לכתוב בכתב יד ברור ונקי. מומלץ להשתמש בעפרון וטחק.
- בשאלון זה 15 דפים, כולל דף זה. וזה כי כל הדפים נמצאים.

בצלחה!

| מספר | שם | ניקוד | ציוון  |
|------|----|-------|--------|
| 17   |    | 25    | שאלת 1 |
| 16   |    | 25    | שאלת 2 |
| 22   |    | 25    | שאלת 3 |
| 18   |    | 25    | שאלת 4 |
| 73   |    | 100   | סה"כ   |



### שאלה מס' 3 (המשך)

ד. ביצוען הקישור לשגרה (פקודות `call -1` ו-`ret`) אינו לשמור את ערכי הדגלים, חוץ לנוכח מנגנון הקישור לפסיקה (פקודות `int -1` ו-`iret`). ברצונו לשביל את מנגנון הקישור לשגרה, כך שהדגלים יישמרו בעת הקראיה לשגרה, וישוחזרו עם החזרה מהשגרה.

לשם כך נגיד זוג מאקרוים בשם `callf` ו-`retf`. המאקרו `callf` מקבל פרמטר יחיד, זהה לפרמטר של פקודת המכונה `call`. המאקרו `retf` הוא ללא פרמטרים. זוג המאקרוים מוגדר כך שהשים וושם בהם ימדי, בהתאם לזוג פקודות המכונה `call -1` ו-`ret`, יגרום לאפקט המבוקש של שימור הדגלים.

בטבלה שלහן ארכנו גרסאות שונות של הגוזרת המאקרו `callf`. עבור כל גרסה, עלין להגדיר גרסה מתאימה של המאקרו `retf`. רשות כל הגוזרת של `retf` מתחת לבן הזוג `callf` המתאים.

ראה דוגמא בטבלה.

אין לשנות את הגרסאות הנთונות של המאקרו `callf`.

ונח כי הקראיה לשגרה והשגרה עצמה נמצאים באותו טגמנט הקוד.

|                                                       |                                                                                                                                         |                                                                                |                                                                                                                          |
|-------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------|
| <pre>callf macro dest pushf call dest popf endm</pre> | <pre>callf macro dest pushf call far ptr dest endm</pre>                                                                                | <pre>callf macro dest local lbl push offset lbl pushf jmp dest lbl: endm</pre> | <pre>callf macro dest pushf call dest endm</pre>                                                                         |
| <pre>retf macro ret endm</pre>                        | <pre>retf macro push bp mov bp,sp push ax xchg ax,[bp+4] xchg ax,[bp+6] xchg ax,[bp+2] xchg ax,[bp+3] pop ax pop bp popf ret endm</pre> | <pre>retf macro popf pop bp jmp [bx]</pre>                                     | <pre>retf macro push bp mov bp,sp push ax xchg ax,[bp+2] xchg ax,[bp+4] xchg ax,[bp+3] pop ax pop bp popf ret endm</pre> |

$$ax = bp \quad [bp+4] = ax$$

$$ax = flags \quad [bp+4] = flags$$

$$ax = ax \quad [bp+4] = flags$$



### שאלה מס' 3 (המשן)

ה. להלן הגדת המאקרו mystery. הפרמטרים x ו-y הם מספרים שלמים.

```

1 mystery macro x,y
2 if x lt 0
3 mystery -x/2,y
4 exitm
5 endif

6 n = y

7 rept x
8 if (n gt 16000) or (n lt -16000)
9 exitm
10 elseif (n le 127) and (n ge -128)
11 db n
12 else
13 dw n
14 endif

15 n = -n*2
16 endm

17 endm

```

ו. להלן ארבע דוגמאות של קריאות למאקרו mystery. לכל דוגמא, רשום את הקוד בשפת אסמבלי שמתබל לאחר פריסת הקריאה למאקרו. יש לרשום רק שורות שיוצרות קוד מכונה (אין לרשום הנחיות לאסמבלי מותנה, הצבות במשתני אסמבילר וכו').

| mystery 6,10                                            | mystery -9,-2                   | mystery 1000,-6000    | mystery -1,3 |
|---------------------------------------------------------|---------------------------------|-----------------------|--------------|
| db 10<br>db -20<br>db 40<br>db -80<br>dw 160<br>dw -320 | db -2<br>db 4<br>db -8<br>db 16 | dw -6000<br>dw 12,000 | dw 0<br>dw 0 |

ו. רשום קריאה למאקרו mystery אשר גורמת לפritis 100 בתים, שיכולים מאותחלים לנרדף.

mystery 100,0



## שאלה מס' 4 (25 נקודות)

לצורך בדיקות של תוכנה, נהוג לפעמים לבצע סימולציה של קלט מהמקלדת על ידי קריית תווים מהזיכרון, במקום לעבוז עם המקלדת עצמה.

למיושן מנגנון הסימולציה, נקזה בזיכרון חוותץ אשר בו ימצאו תוכי הקלט, ונשנה את שגרת השירות של פסיקת שירותים המקלדת #16, כך שתשתמש בתווים מהוירז זה במקום מהמקלד.

כדי לאפשר יחד גמישות, חוותץ הסימולציה אינו קבוע, אלא ניתן לשינוי על ידי המשמש. הגישה להו הבא בחווץ היא דרך מצביע מלא (סגןנט והיסט), הרשות במקומות קבוע ומוסכם, בכתובות הפיזיות #822. מספר התווים שנוחתו בחווץ רשות גם הוא במקומות קבוע ומוסכם, בכתובות הפיזיות #820, בייצוג ללא סימן ובסוד מילה. בעת אתחול מנגנון הסימולציה, המילה בכתובות הפיזיות #820 מאותחלת ל-0.

הסימולציה מתבצעת על ידי שירות מס' 0 של פסיקה #16 (כלומר בשניבנים לפסיקה עם ah=0). השירות מחזיר בכל פנס את התו הבא בחווץ הסימולציה, ומעדכו את המצביע להו הבא ואת מספר התווים שנוחתו. התו מוחזר באונגר ah. מכיוון שקוד ה-scan של המקש במקלדת אינו נגיש, מוחזר באונגר ah הערך 0.

במקרה שהוחץ הסימולציה התרזוקן, השירות עובר לבצע קלט רגיל מן המקלדת האמיתית. השירות יחוור לבצע סימולציה אם וכאשר המשמש יקזה חוותץ חדש (על ידי עדכון המצביע ומספר התווים).

להלן שגרת השירות החדשה של פסיקה #16, הממשת את הסימולציה. השגרה מתחילה בכתובת newInt16.  
הנח כי וקשור הפסיקה של שגרת השירות המקורי נשמר בכתובת oldInt06 שמוגדר בסגןנט הקוד.

```

1 inLen equ 820h
2 inPtr equ 822h
3 oldInt16 dd ?
4 newInt16: push ds
5 push es
6 push bx
7 xor bx,bx
8 mov es,bx
9 cmp ah,0
10 je new0
11 chain16: pop bx
12 pop es
13 pop ds
14 jmp oldInt16
15 new0: cmp word ptr es:inLen,0
16 je chain16
17
18
19
20
21
22 exit16: pop bx
23 pop es
24 pop ds
25 iret

```

שאלות מט' 4 (המשך)

עליך להסביר כיצד פועלת שגרת השירות החדש של פסיקה 46 שבדף הקחתם. השאלות שלhalb  
יונחו אותך בהבנת הקוד. מומלץ לעבור על כל השאלות מראש. שים לב: על ההסברים להתייחס  
למהות הפעולות המתבצעות, ולא רק לצטט במילים את שמות הפקודות והאופרנציות.

i. מה נעשות שוודות 10-9? באילו מקרים מתבצעת ההסתעפות בשורה 10?

ii. מה עשו שורה 114? כיצד חוזרים משגרת הפסיכה כאשר טחבעו שורה זו?

intlk. for single LSR and offset  
source intlk. for single offset source is given by  
 $\delta \approx 1.5^\circ$  when  $\rightarrow 2.0^\circ - 3.5^\circ$   
old intlk. = 1.34

iii. להיכן מציביע האופרנד es:inLen בטעורה ?15

٢٣٦٨٠٣ فی اسٹرالیا و مکانیم ۱۹۸۰ء ۸۲۰ کیجے ۳۵۰

מה קורה לגבי הקלט כאשר סחובות הנסיבות זו? מה עשוות שורות 15-16? באיזה מקרה מתחבצת הטענות בשורה 16?

0-8 → ASCII char. 0-8 OR 0-100% & length byte of file → 15-16 →  
length int 16h & key code 15h → 16 → 0-100% → scan code, 11 → 0-16 16 →  
ah → scancode → ah → 0-16 → 0-16 → 0-16 → 0-16 → 0-16 → 0-16 → 0-16 →  
V → 0-16 → 0-16 → 0-16 → 0-16 → 0-16 → 0-16 → 0-16 → 0-16 →

וְהַ עוֹשָׂת שׂוֹרֶת 17-18 ? מָה תּוֹכֵן האונגר al אֲחָדִי בְּצֹעַן שׂוֹרֶת ?

351.00 instant & 600-00 more & 1 ds -5 went it off  
2) " " " offset -2 " " dx -5

(829h - 28 مارس ۱۹۳۴) نویسندگان  
۲۱h ۱۸ ۰۷h ۱۴h

מִה עוֹשָׂת שׂוֹרֶת ?20-21 .vi

0.351 N 0.7 → 5' 10 4/2

שאליה מס' 4 (המשך)

- ב. כתוב קטע קוד בתכנית המשתמש, שמאתחל את חיצן הטימולציה למחדרת `inputBuf`, המוגדרת בסגמנט הנתונים להלן.

```
.data
inputBuf db "this is just some arbitrary input"
bufLen dw bufLen-inputBuf

.code
xor bx, bx
mov es, bx
mov ax, bufLen
mov es:bufLen, ax
mov ax, seg inputBuf
mov [es:inPtr+2], ax
mov ax, offset inputBuf
mov es:inPtr, ax
```

5.  $\text{buff}_N \cdot \text{buff}_{N-1} \dots \text{buff}_1$   $\xrightarrow{\text{buff}_N}$   
 $\xrightarrow{\text{buff}_{N-1}}$   $\dots$   $\xrightarrow{\text{buff}_1}$

- אם יוכל המשמש להפסיק בכל עת את הסימולציה, ולעבור לקלט רגיל מן המקלצת? הסבר.

(80h ~~لـ~~) 3F1400 → sets up of the picture counter → at offset 18h, />  
push ax  
mov ax, 0  
mov [esinLen], ax  
pop ax  
→ 101, 2 = 10.0-5

- ברצוננו להוכיח את הסימולציה גם לשירות מס' 1 של פסיקה 46. כאמור, שירות זה בודק האם קייםתו בקלט, אך איןנו מוציא את התו מהקלט. גם בשירות זה, כמו בשירות מס' 5, נשתמש בחוצץ הסימולציה. במקרה שחוצץ הסימולציה התרוקן, השירות עובר לבדיקה הקלט מהמקלדת האמיתית. לפיכך, השירות יציין שאין קלט רק כאשר גם חוצץ הסימולציה וגם חוצץ המקלדת ריקים.

עליך למסח את שורות מס' 1 החדש של פסיקה 16h. רשום להלן את התוספות לקוד האסטבלי של שגרת הפסיקה. עליך למספר את השורות הנוספות בהתאם למספריו השוררות הקיימות.  
לדוגמא: שורות שיתווסףו אחרי שורה 10 ימוספרו; 10.1, 10.2 וכד'. אם יש צורך לשנות או למחוק שורה קיימת, רשום את מספר השורה ואת קוד האסטבלי החדש שלה.

10.1 cmp ah,1  
10.2 je new1

```
14.1 reInt: cmp word ptr es:inLen,0
14.2 je chain/6

14.3 lds bx,es:inPtr
14.4 mov al,ds:[bx]
14.5 mov ah,0

14.6 jmp exit/6
```

-2)

$\text{Z}_F = \emptyset$   $\text{S}_{133}^{\text{opt}}$

### שאלה מס' 4 (המשך)

ה. קצב הקלט מוחוץ הסימולציה מהיר בהרבה מקצב הקלט מהמקלדת האמיתית. כדי לדמות את המצב האמתי יותר טוב, נוסף לсимולציה קוצב זמן. הקוצב יגרום לכך שיחלפו בקצבן 200 מיל-שניות מרגע הוצאתתו מוחוץ הסימולציה ונד שניתן יהיה להוציא את התו הבא.

למיושם קוצב הזמן, נגדיר דגל בגודל בית בכתובת הפיזית `h826`. פסיקת שירות השעון `h1ch` תרים את הדגל (תציב בו 1) כל 200 מיל-שניות. במקביל, שירות מס' 0 של פסיקה `h16` ימתין בלולאה כל עוד הדגל איטו טורם, ולפנוי החזרה מן הפסיקה יוריד את הדגל ל-0.

להלן חוספת לקוד השגורה החדש של פסיקה `h16`, למיושם קוצב הזמן. מספרי השורות הנוספות הם בהתאם למספרי השורות הקיימות.

```
16.1 inReady equ 826
16.2 sti
16.3 delay: cmp byte ptr es:inReady,1
16.4 jne delay
16.5 mov byte ptr es:inReady,0
```

עליך לממש את שגרת השורות החדש של פסיקת שירות השעון `h1ch`, לפי התיאור לעיל. השגורה מתחילה בכתובת `cnewInt1c`. הנץ כי וקטור הפסיקה של שגרת השירות המקורי נשמר במשתנה `oldInt1c` שמוגדר בסגמנט הקוד.

```
.code
oldInt1c dd ?
newInt1c:
```

- 5 -



### שאלה מס' 1 (25 נקודות)

- א. נתון שדגל הגילשה כבוי ( $0 = \text{off}$ ), והאוגר  $al$  מכיל את הערך 128. עברור כל אחת מהפקודות שלහן, ציין מתי היא גורמת להדלקת דגל הגילשה ( $1 = \text{on}$ ): תמיד, לפחות פעם אחת, או אף פהם. הקפ בעיגול את התשובה הנכונה.

$-128 + -128 = 5$

**תמצית / לפחות פעם אחת**  
**תמצית / לפחות פעם אחת**

**neg al**  
**add al,al**  
**sub al,al**  
**add al,bl**  
**rcr al,1**  
**sar al,1**  
**shl al,1**  
**sahf**

- ב. באסמבילד של שני מעברים, אילו מהטישיות שלහן מבוצעות במעבר הראשון?  
תתכן יותר מתשובה אחת וכונה. הקפ בעיגול את כל התשובות הנכונות.

- i. פריסת הקיימות למאגר. (שיי → )
- ii. הבנת קובץ קוד המכוונה (object module).
- iii. ניפוי שגיאות.
- iv. פתרון התייחסויות לסמלים חיצוניים.
- v. בניית טבלת הסמלים.

- ג. לאילו מטרות משמשת הvariable `model`. של האסמבילר?  
תתכן יותר מתשובה אחת וכונה. הקפ בעיגול את כל התשובות הנכונות.

- i. לציין את דגם המעבד שעליו אמורה התכנית לרוץ.
- ii. לאפשר כתיבת תכניות גדולות התופסות יותר מסegment קוד אחד.
- iii. לקבוע את הגודל המקורי המקורי של הסגמנטים.
- iv. להרשות שימוש באוגרים הרוב תכלייטים המורחבים (ברוחב 32 ביטים).

- ד. עברור כל אחד משלשת קטעי הקוד שלහן, ורשום את תוכנו של האוגר `ax` בגמר ביצוע הקטע.  
רשום את התשובה בבטיס 10.

|                                                                                                                                                                                    |                                                                                                                                                          |                                                                                            |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------|
| <pre> .data x db '111122233334400',0 .code xor dx,dx mov di,@data mov es,di lea di,x cld mov cx,-1 a: inc dx mov ax,es:[di] repe scasw sub di,2 cmp byte ptr es:[di],0 ja a </pre> | <pre> xor dx,dx xor ax,ax mov bx,sp mov cx,100h mov es,ax mov word ptr es:4,offset y mov es:6,cs b: inc dx mov sp,bx push cx popf y: inc al jns b </pre> | <pre> xor dx,dx xor cl,cl mov ax,5555h mov cl,33 c: inc dx sub cl,2 xor ax,cl jno c </pre> |
| $dx =$ <input type="text"/>                                                                                                                                                        | $dx =$ <input type="text"/>                                                                                                                              | $dx =$ <input type="text"/>                                                                |

המשר שאלה מס' 1 בדף הבא

### שאלות מס' 1 (המשך)

ל. תרגם את הכתובת הלוגית (segment:offset) של הלו לכתובת פיזית, כשההמעבד פועל במחצבי real.  
רשוט את התוצאה בסיס 16.

4d750  
+ 0b10c  
5885C

|   |   |   |   |   |
|---|---|---|---|---|
| 5 | 8 | 8 | 5 | C |
|---|---|---|---|---|

צין את ערבי הדגלים CF ו- OF בגמר כל פנולה, כפי שהיו נקבעים על ידי ביצוע בטען 86<sup>x</sup>.  
כל המספרים נתוניים בבסיס 16. רשום גם את התוצאות בבסיס 16.  
בצע את פועלות החיבור והחיסוך של halo בשיטת המשליש ל- 2 ברוחב של 16 ביטים.

|                                                              |                                                              |                                                              |                                                              |
|--------------------------------------------------------------|--------------------------------------------------------------|--------------------------------------------------------------|--------------------------------------------------------------|
| $\begin{array}{r} 6f8f \\ - 7274 \\ \hline F01b \end{array}$ | $\begin{array}{r} 6f8f \\ + 7274 \\ \hline E103 \end{array}$ | $\begin{array}{r} df67 \\ - 9505 \\ \hline 4A62 \end{array}$ | $\begin{array}{r} df67 \\ + 9505 \\ \hline 446C \end{array}$ |
| $CF = \underline{\underline{1}}$                             | $CF = \underline{\underline{0}}$                             | $CF = \underline{\underline{0}}$                             | $CF = \underline{\underline{1}}$                             |
| $OF = \underline{\underline{0}}$                             | $OF = \underline{\underline{1}}$                             | $OF = \underline{\underline{0}}$                             | $OF = \underline{\underline{1}}$                             |

תרגם את המספרים שבבלה מבחן 10 לייצוג סטנדרטי בשיטת הקודה הצפה בבחן 2.  
רשום את הזרקה ללא bias (ראח דונחא)

| Decimal | Sign | Exponent | Mantissa  |
|---------|------|----------|-----------|
| 9.0     | 0    | +3       | 1.001     |
| 24.125  | 0    | -4       | 1.1000001 |
| -0.25   | 1    | -2       | 1.0       |

**תזכורת:** בשיטת הנקודה הצפה, שדה החזקה הוא מספר לא סימן הכלל bias.

```
float1 dd 0c118000h
float2 dd 41C1000h
long1 dq 0000000000000000h
```

$$\begin{array}{r}
 -9.5 \\
 + 24.125 \\
 \hline
 -0.95 \\
 \hline
 \end{array}$$
  

$$\begin{array}{r}
 \overset{e}{1} \overset{f}{0} \overset{g}{1} \overset{h}{1} \overset{i}{1} \overset{j}{1} \overset{k}{1} \overset{l}{1} \overset{m}{0} \overset{n}{0} \overset{o}{0} \overset{p}{0} \overset{q}{0} \overset{r}{0} \overset{s}{0} \overset{t}{0} \\
 \hline
 \end{array}$$
  

$$\begin{array}{r}
 1023 \\
 - 2 \\
 \hline
 1021 \\
 - 1023 \\
 \hline
 - 1023
 \end{array}
 \quad 
 \begin{array}{r}
 1.00 \times 2^{-2} = 0.01
 \end{array}$$

## שאלה מס' 2 (25 נקודות)

להלן מספר הגדירות:

- "ריצה" היא רצף של בתים שköלט מביליט את אותו ערך. אורן הריצה הוא מספר הבתים בሪיצה.
  - התחלת הריצה היא הבית שבכתחובת הנמצאה ביותר בሪיצה.
  - לצורך שאלה זו, מחרוזת היא רצף של בתים מסוימים בבית שמקביל את השם 5. אורן המחרוזת הוא מספר הבתים, לא כולל הבית מסוים. דהיינו, זו הערך מחרוזת כמו בשפה C לדוגמה, נתונה המחרוזת הבאה:
- ```
string1 db 'xxxxxxxxxxxxxx1111xx',0
```
- בדוגמה זו, הרץ '1111' הוא ריצה באורך 4, הרץ 'xxx' הוא ריצה באורך 3, והו 'x' הוא ריצה באורך 1. לעומת זאת, הרץ 'xx' אינו מהו ריצה.
- + הרץ 'xxxxxxxxxxxxxx' הוא ריצה באורך 15, וזהי הריצה הארוכה ביותר בכתובת.
- א. כתוב בשפה אסטטלי שגרה בסיס `getrun proc`, מקבלת כפרמטר באוגרי `es` ו-`di` כתובות לוגית טלאה (סמנט והיסט). השגרה מחזירה באוגר `al` את האורך של הריצה הארוכה ביותר בכתובת שמהיה.
- בכתובת [`ip:es`]. האורך מוחזר ביצוג ללא טימן. אם אורך הריצה גדול מ-255, מוחזר הערך 255.

הקפד על כללי התכונות הנכון של שגרה.

השגרה: תרגום מ-assembly ל-assembly ↓
הארוכה ביותר

```

getrun proc
    push bp
    push bx
    mov bp, sp
    push cx
    push si
    push dx
    mov cx, 255
    xor bx, bx
    xor ax, ax
    xor dx, dx
    mov si, di
    inc di
    cmp byte PTR[es:di], 0
    je f1
    scasb
    jne f2
    inc dx
    jmp f1
    inc si
    mov di, si
    cmp byte PTR[es:di], 0
    je exit
    loop f1
exit:
    mov al, bl
    pop dx
    pop si
    pop cx
    pop bp
    ret
endp

```



שאלה מס' 2 (המשך)

ב. נתונה השגרה compress שבודדה (בשפת C):

```
unsigned int compress(char *inStr, char *outStr)
```

הפרמטר `inStr` הוא מצביע למחזורת בסגמנט הנתוני. הפרמטר `outStr` הוא מצביע לחיצז בסגמנט הנתוניים, שגודלו בלתי מוגבל (קלוטר, מניחס כי יש תמיד מספיק מקום לחיצז).

השגרה `compress` דוחשת את המחרוזת `inStr`. נוצרת מחזורת חדשה (מחזורת דחוסה), שהיא קצרה יותר מ-`inStr`, ובנוסףobarן שמאפשר לשזר ממנה את המחרוזת המקורית. השגרה כותבת את המחרוזת הדחוסה לתוך החיצז `outStr`, ומוחזירה את אורך המחרוזת הדחוסה.

הדחוסה נשית בשיטה הבאה. ריצה באורך `n` בתים (255≤n) מוחלפת בשני בתים כלהלן:

הבית הראשון מכיל את התו שמרכיב את הריצה, והבית השני מכיל את המספר `n` (ביצוג ללא סימן).

לדוגמא, אם נדחס את המחרוזת `string1` שלhalbו לפי שיטה זו, נקבל מחרוזת זהה `-2.string2`.

```
string1 db 'xxxxxxxxxxxxxxxy1111xx',0
string2 db 'x',15,'y',1,'1',4,'x',2,0
```

בדוגמה זו, אורך המחרוזת המקורית `string1` הוא 22, ואילו אורך המחרוזת הדחוסה `string2` הוא 8.

להלן מימוש השגרה `compress` בשפת אסטREL. הפרמטרים מועברים לפי המוסכמות של שפת C. השגרה משתמשת בקריאה לשגרה `getrun` מטעניף א'.

<pre style="font-family: monospace;">1 compress proc 2 push bp 3 mov bp,sp 4 push bx 5 push di 6 push es 7 8 push ds 9 pop es es = ds 10 mov bx,[bp+6] bx = offset outStr 11 mov di,[bp+4] di = offset inStr 12 13 nextrun: cmp byte ptr [di],0 14 je exitcomp 15 16 mov al,[di] ; מרים באל 17 mov [bx],al ; מרים בbx 18 call getrun 19 mov [bx+1],al ; מרים בbx+1 20 add bx,2 ; מרים בbx+2 21 xor ah,ah ; מרים בbx+3 22 add di,ax ; מרים בbx+4 23 jmp nextrun ; מרים בbx+5 24 25 exitcomp: mov byte ptr [bx],0 26 mov ax,bx 27 sub ax,[bp+6] ; מרים בbx+6 28 29 endp</pre>	
---	--

המשר שגילם בערך 2 בדף הבא

שאלות מס' 2 (המשך)

כתוב בשפת אסמבלי קטע קוד שקורא לשגרה compress כדי לדחוס את המחרחת string שלහלו לתוכה החיצן buffer. לאחר מכן, קטע הקוד מופיע בצורה נאה את אורך המחרחת הדחוסה, בunedt קרייה לשגרה printf של שפת C. הגדר בסגנון הנתוניים את מהירות הפורט המועברת לprintf.

ii. דשום את תוכן המחרוזת הדחוסה שנוצרת על ידי השגורה compress בקריאה שבסעיף 2. השתמש במבנה הדומה למחרוזת הדחוסה string2 מהדוגמה בדף הקודם.

'a', 6, 'b', 3, 'c', 1, 'd', 1, 'a', 2, 'e', 255, 'e', 245, 0 ✓

iii. מה עשו השלגורה compress כאשר המחרחת המועברת בפורמט `tar` היא באורן?

($\sigma = \text{wind } y^{de}$) \circ $ax \rightarrow \text{signall outStr} \rightarrow \sigma$ \circ ax

✓

iv. במקרים מסוימים, אורך המחרוזת הדחוסה שנבנית על ידי השגרה compress גדול מأורך המחרוזת המקורית. הסביר מה מאפיין את תוכן המחרוזת המקורית במקרים אלו.

outStr -> null & eものが標準でGが付属するので

1

יינו אילו שנויים יש לבצע בקוד השרשרת compress כדי שniton יהיה לקרוא לה מתחם תכנית אכתובה בשפה C.

public _compress
_compress proc

16

אכט טו

שאלה מס' 2 (המשך)

2. כתונה הפעלה `uncompress` שכותרתה (בשפת C):

```
unsigned int uncompress(char *inStr, char *outStr)
```

הפרמטר `inStr` הוא מצביע למחוזות בסגמנט הנתונים, והפרמטר `outStr` הוא מצביע לחיצן בגודל בלתי מוגבל בסגמנט הנתונים. השגורה מבצעת פרנטפורמציה ההפוכה לזו של השגורה `compress` מסעיף ב'. ככלומר, השגורה `uncompress` מקבלת מחוזות דחוסה `inStr`, משחזרת ממנה את המחרחות המלאה המקורי, וכותבת את המחרחות המלאה אל החיצן `outStr`. השגורה מוחזיקה את אורך המחרחות המלאן.

i. לא כל מחרוזת נתונה יכולה לשמש כפורמטור חוקי `inStr` לשגרה `uncompress`. הסבר טענה זו.

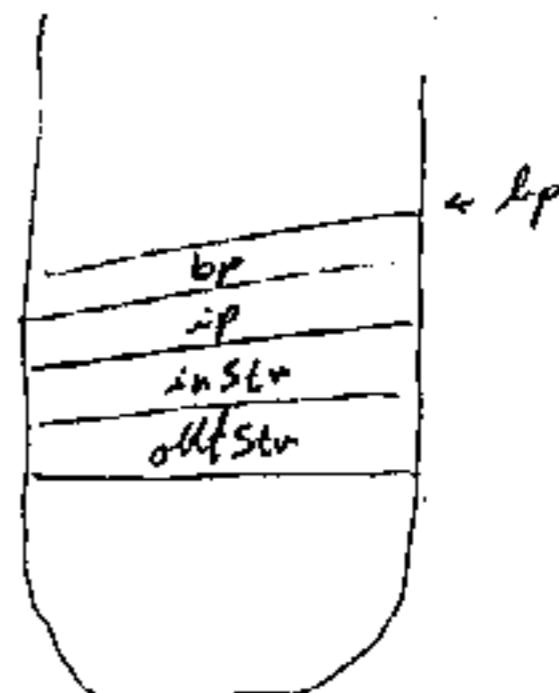
compress only if the ex-is not the word - in this case the verb is
also not the word for the verb is not the word for the verb

כטוב את השגרה `uncompress` בשפת אסמבלי. העבר את הפורמטדים לפי המוסכמות של שפת C. חוובה לשתמש בפקודה `rep stos` לבניית המחרחת המלאה.
אפשר להניח שהמחרחת `Stm` היא חוקית. הקפד על כללי התכונות הנכונות של שגרה.

public _uncompress
_uncompress proc
push bp
mov bp, sp
push cx
push bx
push di

push es

xor ax, ax
mov bx, [bp+4]
mov di, [bp+6]
f1: mov ~~bx~~, [bx+1]
xor ch, ch adj cl
mov al, [bx]
mov dx, ds
mov es, ds
rep stosb
add bx, 2
cmp byte ptr [bx3, 0
je exit
jmp f1



```
cmp byte ptr [a], 0  
je exit
```

1

```
exit:    mov byte ptr [di], 0
        dec di
        push ax,[di]
        sub ax,[bx+6]
        pop es
        pop di
        pop bx
```

```
pop cx  
pop bp  
ret  
endif
```

שאליה מס' 2 (המשך)

ד. כתוב שגורה ב-SIMPLIFIED-CFPTRIM במחסנית שני מספרים y ו- x מטיפוס ממשי, המוצגים בשיטת הנקודה הצפה ברוחב 32 ביטים. השגורה מודירה באוגר(0) את המוצען של שני המספרים, כלומר את הערך $2/(y+x)$. הקפד על כללי התכונות הנכון של שגורה.

average Proc

fadd

fld num1

fch st(1)

fdiv

ret

endp

st 0



ו. בסגנון הנתונים מוגדרים שלשה משתנים, $1-\text{num}$, $2-\text{num}$, $3-\text{res}$, כולם מטיפוס ממשי בנקודה, צפה, ברוחב 32 ביטים. כתוב קטע קוד הקורא לשגורה average עם הפרמטרים $1-\text{num}$, $2-\text{num}$, ומציב את השך המוחזר מן השגורה לתוך המשתנה $.avg$. הקפד על כללי התכונות הנכון של קריאה לשגורה.

.data
num1 dd 12.34
num2 dd 56.78
avg dd ?

.code

finit

fld num1

fld num2

call average

fst avg

mov ah,4ch

int 21h

end



שאלה מס' 3 (25 נקודות)

- א. כידון, במניבד 8086, פקודת המכונה push אינה יכולה לקבל אופרנד טיבידי (קבוע). הגדר מאקרו בשם `pushfp` שמקבל פרמטר קבוע בגודל מילה. המاكרו מכניס את הקבוע לראש המחסנית. על המاكרו לעבוד נכון נכוון במניבד 8086. בדומה לפקודה `push`, אוסף למاكרו לשנות את הדגמים.

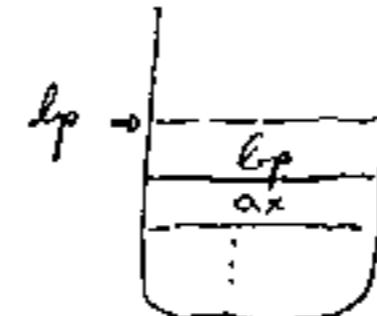
`pushfp macro x`

```

    push ax
    mov ax, x
    push bp
    mov bp, sp
    sub ax, [bp+2]
    pop bp

```

`endm`



- ב. בתכנון הדור הבא של מעבדי x86 הוחלט לוותר על מרבית הסתעפות המותנית, זאת כדי לפנות קודי פונולח (opcode) עבור פקודות חדשות מתקדמות. נותרו רק הסתעפות המותנית שפונולות לפי דגל בודן, כלומר, צלקמן: `jz, jnz, js, jns, jc, jnc, jo, jno`.

הגדר מאקרו בשם `jz` שמקבל פרמטר ייחד זהה לאופרנד של פקודה המכונה `je`. המاكרו מבצע הסתעפות מותנית לפי הגדידה של הפקודה `je`. נל המاكרו לעבוד נכון נכוון במעבדים מהדור הבא. בדומה לפקודה המכונה `je`, אוסף למاكרו לשנות את הדגמים.

`jz macro dest`

`local end, ck1, ck2, doj.`

`push ax`

`pushf`

`pushf`

`pop ax SF=0`

`jnz doj SF=1`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`

`and ax, 0000 1000 1000, 00000b`

`test ax, 0`