שאלה מספר 1 (25 נקודות)

נניח שלתוכנית אסמבלר יש את המשתנים הבאים:

```
    .DATA
StopFlag  DW  0
Buffer DB 6 DUP ('#'),13,10,'$'
Prompt_Str DB 'Input up to now is:',10,13,'$'
Index  DW  0
```

הראה כיצד ניתן להשתמש בפסיקת ה-Timer (פסיקה מספר 8) המרחשת 18.27 פעמים בשניה בכדי:

במידה והמשתמש לחץ על מקש, על כל לחיצת מקש (למעט ESC) יש להעביר את קוד ה-ASCII למקום הנוכחי ב-Buffer.

השימוש ב- Buffer הינו ציקלי, אחרי שנכתב תוכן לתו השישי חוזרים להתחלה.

במידה והמשתמש לא לחץ על מקש, אין לבצע המתנה.

במידה והמשתמש לוחץ ESC, להעביר 1 ל- StopFlag.

הנך רשאי להניח שהמשתמש לוחץ רק על המקשים הדפיסים, או ESC.

לדוגמא, פלט אפשרי של התוכנית הבאה:

```
CODE
     .386
   ; Cause time delay
Delay  MACRO
       LOCAL Aloop
       STI
       MOV  ECX,30000000
  Aloop:
       STI
       DEC  ECX
       CMP  ECX,0
       JNE  Aloop
       ENDM  ; Delay
;
 ..............................
```

```
Forever1:
     CMP StopFlag,1
     JE  Done
;
     LEA  DX,Prompt_Str
     MOV  AH,9
     INT  21h
     ;
     LEA DX,Buffer
     MOV  AH,9
     INT  21h
     DELAY
     JMP  Forever1
;  Return to DOS
Done:
```

יהיה:

```
D:> intr.exe
Input up to now is:
######
Input up to now is:        <--- '2'  -ו  '1'  נלחצו
123###
Input up to now is:        <---   כאן לא נלחצו מקשים
123###
Input up to now is:        <--- '6' -ו '5', '4'  נלחצו
123456
Input up to now is:        <--- ESC -ו  '8' ,  '7'  נלחצו
783456
D:>
```

תרגיל בית 4

שאלה מספר 2 (25 נקודות)

כתוב רוטינת אסמבלר חניתנת לקריאה מתוך תוכנית C המשמשת את ה-prototype הבא:

```
extern  void complex_add(long int *real1, long int *imag1,
                         long int real2, long int imag2 );
```

הרוטינה complex_add מקבלת שני פוינטרים לשלמים (long int) ושני מספרים שלמים (long int).

complex_add מפרשת את ארבע המספרים הללו כתאור של 2 מספרים מרוכבים, ומחשבת את הסכום של שני המספרים הללו בחזרה לתוך הראשון.

לדוגמא, פלט אפשרי של התוכנית הבאה:

```
void main(void)
{
 char info[80];
 long int r1, i1, r2, i2;

 puts("Enter four long integers (r1, i1, r2, i2):");
 gets(info);
 sscanf(info,"%ld %ld %ld %ld", &r1, &i1, &r2, &i2);
 printf("(%ld + %ldi) + (%ld + %ldi) = ", r1, i1, r2, i2 );

 complex_add(&r1, &i1, r2, i2 );
 printf(" (%ld + %ldi) \n", r1, i1 );

} /* main */
```

יהיה:

```
Enter four long integers (r1, i1, r2, i2):
10   20   300   400
(10 + 20i) + (300 + 400i) =  (310 + 420i)
```

```
omplex_add PROC
;-----------------------------------------------------------------------
USH BP
OV BP,SP
USHF
USH SI
USH DI
                    שמירת אוגדים

OV AX,[BP+4]        AX <-- &r1
OV AX,[AX]          AX <-- r1
OV BX,[BP+10]       BX <-- r2
DD AX,BX            AX = AX + BX

OV AX,[BP+6]        AX <-- &i1
OV AX,[AX]          AX <-- i1
OV BX,[BP+14]       BX <-- i2
DD AX,BX            AX = AX + BX

OP DI
OP SI               שחרור אוגדים
OPF
OP BP
;-----------------------------------------------------------------------
ET
omplex_add ENDP
```

DI	
SI	
FLAGS	
BP	
RET_ADD	(IP,CS)
&i2	[BP+4]
&i1	[BP+6]
r2	[BP+10]
i2	[BP+14]

<u>שאלה מספר 3</u> (25 נקודות)

באופן כללי, כאשר מעוניינים לממש באסמבלר את החישוב:

קבוע + אוגר2 + אוגר1 = אוגר3

יש צורך לממש את החישוב בשלוש פקודות מכונה.

בתנאים מסוימים מאד ניתן לממש את החישב בפקודה אחת ע"י שימוש ב-LEA.

לדוגמא: LEA AX,[BX+SI+7]

כתוב <u>מקרו</u>

```
Add_Reg16  MACRO Target_Reg, Reg1, Reg2, Const
```

הפורש את הקוד הטוב לחישוב Target_Reg = Reg1 + Reg2 + Const.

במידה והדבר אפשרי, החישוב יתבצע ע"י LEA. אחרת ימומש ע"י שלוש פקודות מכונה.

לדוגמא,

```
Add_Reg16 AX,BX,DI,7
```

הוא דוגמא לקריאה שיפרוש פקודת LEA.

```
Add_Reg16 Trarget_Reg, Reg1, Reg2, Const

 Check_BX :
;--------------------;
 IFDIF <Reg1>,<BX>
 JMP Check_BP
 ELSE
 JMP Check_SI
 ENDIF

 Check_BP :
;--------------------;
 IFDIF <Reg1>,<BP>
 JMP Prisa_1
 ELSE
 JMP Check_SI
 ENDIF

 Check_SI :
;--------------------;
 IFDIF <Reg2>,<SI>
 JMP Check_DI
 ELSE
 JMP Prisa_2
 ENFIF

 Check_DI :
;--------------------;
 IFDIF <Reg2>,<DI>
 JMP Prisa_1
 ELSE
 JMP Prisa_2
 ENDIF

 Prisa_1 :
;--------------------;
 ADD Reg1,Reg2
 ADD Reg1,Const
 LEA Trarget_Reg,Reg1
 JMP Sof

 Prisa_2 :
;--------------------;
 LEA Trarget_Reg,[Reg1+Reg2+Const]

 Sof :
;--------------------;
 ENDM
```

חישוב ה - Offset

Base		Index		
BX		SI		
או	+	או	+	Displacement
BP		DI		

שאלה מספר 4 (25 נקודות)

א. כתוב רוטינת אסמבלר בשם ParMaxCharString המקבלת מערך תוים ומחשבת לתוכו את סידרת המקסימומים החלקיים:
כל איבר במערך מכיל את המקסימום של כל התוים עד אליו (כולל).

הקלט של הרוטינה תועבר ע"י:

CX - אורך מערך התוים,
SI - כתובת ההתחלה יחסית ל-DS.

לדוגמא, אם

```
.DATA
String  DB '0321548679'
.CODE
MOV  CX,10
LEA  SI,String
CALL ParMaxCharString
```

יגרום ל-String להכיל את הערך '0333558889'.

ב.
כתוב רוטינה המקבלת פרמטר תוכנית ראשית ומדפיסה את סידרת המקסימומים החלקיים שלו, מבלי להשתמש בשטח זכרון נוסף.

הינך רשאי להשתמש ברוטינה ParMaxCharString, גם אם לא ממשת אותו.

לדוגמא, ריצה אפשרית של התוכנית היא כלהלן:

```
D:\> ParMaxParm.exe 12321454
12333455
```

סעיף א' :

```
ParMaxCharString PROC
;-----------------------------------------------------------------;
L1 :

MOV AX,[SI]
CMP AX,[SI+1]          בודקים את המערך בלולאה
JLE Cont               בכל שלב משווים איבר במערך
MOV [SI+1],AX          עם האיבר הבא אחריו
Cont :                 ומבצעים עדכון לאיבר מסויים
 INC SI                רק אם האיבר שבא לפניו
 LOOP L1               גדול ממנו
;-----------------------------------------------------------------;
RET
ParMaxCharString ENDP
```

סעיף ב' :

```
PSP PROC
;-----------------------------------------------------------------;
ASSUME NOTHING
ASSUME CS:_TEXT            שינוי ES
  MOV AX,@DATA             להצבעה על
  MOV ES,AX              DATA_SEGMENT
ASSUME ES: DATA

MOV CX,DS:[80h]        CX <-- גודל המערך בתווים
MOV SI,DS:[81h]        SI <--   תוכן המערך עצמו

CALL ParMaxCharString

MOV AH,40h
MOV BX,1               הדפסת המערך
MOV DX,SI
INT 21h
;-----------------------------------------------------------------;
RET
PSP ENDP
```

אוניברסיטת חיפה

החוג למתימטיקה ומדעי המחשב

23.7.97

מבנה מחשבים ושפות סף

מבחן סיום - מועד א'

מרצה: ד"ר איתן רון

מתרגל: עדכן אגבריה

הערות

1) מותר כל חומר עזר מודפס. אין להשתמש במחשבים ניידים (Notebooks).
2) זמן הבחינה - 3 שעות.
3) כל הרושמים שעליך לכתוב חייבות להיות באסמבלר.
4) מותר לצרף דפים לטופס הבחינה אם יש צורך בכך.
5) למרות שהניקוד של השאלות עולה על 100, הציון המירבי שניתן לקבל בבחינה הוא 100.

שאלה	נקודות
1	/ 30
2	/ 30
3	/ 20
4	/ 20
5	/ 8
סה"כ	/108

שאלה מספר 1 (30 נקודות)

בשפת C קיימת רוטינה void sleep(int sec) שקריאה עליה גורמת לעיכוב של sec שניות.

עליך לממש גירסה משלך של void sleep(int sec) הגורמת לעיכוב של sec שניות או מעט יותר (עד 6% יותר). כלומר, הרוטינה שלך תשמש לגרימת המתנה בזמן של מספר שניות רצוי, כאשר רמת דיוק הנדרשת ממנה היא עד 6% כלפי מעלה.

הרוטינה sleep אינה מסתמכת של שום עזרה מן התוכנית הראשית, ואינה מניחה שהתוכנית הראשית מביאה בחשבון את שיטת פעולתה של sleep.

כתוב רוטינת אסמבלר שממשת את הרוטינה sleep. שים לב שהמימוש של ההמתנה חייב להיות בילתי תלויה במהירות של המחשב שבו מדובר (286, 386, 486, 586).

ההנחה היא כמובן שמדובר באפליקצית DOS.

לדוגמא, פלט אפשרי של התוכנית הבאה:

```
/* testslp1.c - test sleep */
#include <stdio.h>

extern void sleep(int secs);

void main()
{
 int n;

 puts("Enter secs");
 scanf("%d", &n);
 puts("Before sleep");
 sleep(n);
 puts("After sleep");

}
```

יהיה:

```
Enter secs
30
Before sleep
After sleep
```

כאשר ההדפסה של After sleep יהיה 31.8 - 30 שניות אחרי ההדפסה של Before sleep.

שאלה מספר 2 (30 נקודות)

בשפת C קיימות רוטינות setjmp ו- longjmp המאפשרות הסתעפיות "לא-לוקליות" - הסתעפות לבקודה רצויה בפונקציה קודמת, גם אם היא מספר רמות למעלה.

דוגמא לצורך של הרוטינות הללו הינו במקרה של "תפריט ראשית":

```
printf("Press 1 for ...\n, 2 for ..., ... q, Q for quit\n");
c = getch(stdin);

switch(c) {
        case '1': option1();
        case '2': option2();
             ....
        case 'q': case 'Q': return;
        }
```

מה יקרה אם הרוטינות option1() בעצמם יקראו לרוטינות נוספות שבהם יתעורר הצורך של "חזרה לתפריט ראשית"? לא ניתן לפתור את הבעיה בעזרת goto (או jmp) למשל משום שה-return שינבא מלחיצת q יבצע חזרה לא למערכת ההפעלה, אלא לרוטינה פנימית.

הדרך לפתור את הבעיה הזו להגדיר רוטינה setjmp "לסמן" נקודה בתוכנית ורוטינה longjmp המדמה (ואו בעצם משחזר) ביצוע חזרה ל-setjmp.
setjmp יצטרך לשמר איפורמציה שחייב להיות מועבר ל-longjmp בכדי לאפשר ל-longjmp לבצע את תפקידו.

לפיכך יהיה הגדרה של שטח לשימור מידע jmp_buf,

אילה יהיו ההגדרות של setjmp ו-longjmp:

```
extern int   setjmp(jmp_buf *jmpb);

extern void   longjmp(jmp_buf *jmpb, int retval);
```

ותוכנית "התפריט הראשית" תיראה כך:

```
jmp_buff buff; /* Global variable */

main()
{
.......

printf("Enter 1 for ...\n, 2 for ..., ... q, Q for quit\n");
c = getch(stdin);

setjmp(&buff); /* Mark point in program and store info */

switch(c) {
        case '1': option1();
        case '2': option2();
            ....
        case 'q': case 'Q': return;
        }
```

וביצוע longjmp(&buff, ret_val) בתוכנית פנימית יבצע חזרה ל-switch כאילו חזר מ-setjmp.

setjmp מחזיר כתוצאת פונקציה 0 כאשר נקרא ישירות ("הקריאה הראשונית") וקריאה ל-longjmp גורם לחזרה כביכול מ-setjmp עם תוצאת פונקציה שמועבר ל-longjmp בפרמטר השלם ret_val. בצורה כזו אפשר להבדיל בין החזרות השונות מ-setjmp כאשר הדבר נחוץ.

עליך לממש את הרוטינות setjmp ו-longjmp באסבלר עבור TURBO C במודל SMALL.

השתמש בהגדרה הבאה של jmp_buf:

```
typedef struct __jmp_buf {
    unsigned    save_sp;        [BX]
    unsigned    save_ip;        [BX+2]
    unsigned    save_bp;        [BX+4]
    unsigned    save_old_bp;    [BX+6]
    unsigned    save_flags;     [BX+8]
    unsigned    save_di;        [BX+70]
    unsigned    save_si;        [BX+72]
}   jmp_buf;
```

דוגמא מלאה של הרוטינות longjmp ו-setjmp:

הפלט של התוכנית הבאה:

```
void subroutine(jmp_buf);

int main(void)
{
```

```
    int value;
    jmp_buf jumper;

    value = setjmp(jumper);
    if (value != 0)
    {
       printf("longjmp with value %d\n", value);
       exit(value);
    }
    printf("About to call subroutine ... \n");
    subroutine(jumper);

    return 0;
}

void subroutine(jmp_buf jumper)
{
   longjmp(jumper,4);
}
```

יהיה:

```
About to call subroutine ...
longjmp with value 4
```

שאלה מספר 3 (20 נקודות)

כתוב מקרו בשם Gen_Fibo_Table MACRO Alabel,N, המקבלת פרמטר שלם N ושם משתנה Alabel ויוצרת מערך של מספרי WORD בשם Alabel שהיא למעשה טבלה של N מספרי פיבונצי הראשונים.

שים לב שאינך יודע מהו ערכו של N וגודל המערך צריך להיות בדיוק N מספרים בגודל WORD.

מספרי פיבונצ'י הם כמובן הסידרה של מספרים המתחילה ב- 1,1 וכל מספר הבא אחריהם הינו סכום שני קודמיו, כלומר 1,1,2,3,5,8,13,21,34,...

לדוגמא, אם בתוכנית מופיעה הקוד

```
Gen_Fibo_Table Fibo_Table,20
```

אזי Fibo_Table יהיה מערך של 20 מספרי WORD שיכיל את 20 מספרי פיבונצ'י הראשונים וקטע הקוד

```
MOV  BX,7*2
MOV  AX,Fibo_Table[BX]
```

יקרא את מספר פיבונצ'י השביעי (13) לתוך AX.

שאלה מספר 4 (20 נקודות)

כתוב פרוצדורה בשם Test_Mem_Byte המקבלת כתובת אבסולוטית (מספר יחיד) של byte בזכרון דרך אוגר EAX ובודקת אם הוא (הזכרון בחומרה) תקין. הבדיקה של תא זכרון יעשה ע"י כתיבה וקריאה של שני ערכים שונים לתוך התא ובדיקה אם הערך שנקרא הוא גם הערך שנכתב. במידה והזכרון נמצא תקין Test_Mem_Byte מחזיר 0, אחרת יחזיר 1.

הינך רשאי להניח שהכתובת האבסולוטית הוא בתוך 1M = 1048576 התאים הראשונים של הזכרון.

לדוגמא, התוכנית הראשית הבאה קוראת ל-Test_Mem_Byte על מנת לבדוק את הכתובת האבסולוטית 512473.

```
Main:
     MOV AX,@DATA
     MOV DS,AX
     MOV AH,9          ; Set print option for int 21h
     MOV DX,OFFSET TestMsg  ; Set DS:DX to point to TestMsg
     INT 21h                ;  Print TestMsg

     MOV  EAX,512473
     CALL  Test_Mem_Byte
;
     CMP  AX,0 ; Memory OK?
     JNE Not_Ok  ; No, Print Error
     MOV DX,OFFSET Ok_Msg  ; Yes, Print OK
     JMP Ok
Not_Ok:
     MOV DX,OFFSET Deffect_Msg  ; Print NOT OK
Ok:
     MOV AH,9          ; Set print option for int 21h
     INT 21h                ;  Print appropriate message
;
     MOV AH,4Ch          ; Set terminate option for int 21h
     INT 21h          ; Return to DOS (terminate program)
     END  Main
```

שאלה מספר 5 (8 נקודות)

ב-Protected Mode יש למעשה 3 רמות הגנה של מערכת ההפעלה בפני תוכניות האפליקציה:

1. מנגנון הפסיקות,
2. הזכרון הוירטואלי,
3. רמת הפריוילגיה.

הסבר

א. כיצד הזכרון הוירטואלי חיוני להגן על השליטה של מערכת ההפעלה על מנגנון הפסיקות,

ב. כיצד רמת הפריוילגיה חיוני להגן על השליטה של מערכת ההפעלה על מנגנון הפסיקות,

ג. כיצד רמת הפריוילגיה חיוני להגן על השליטה של מערכת ההפעלה על הזכרון הוירטואלי.