תוכניות דוגמא dem_key1.c, dem_key5.c

התוכניות הללו תפקידן להמחיש שימוש ב-INT 16h. dem_key1.c ממחישה שימוש באופציה AH = 0 קריאת מקש מהמקלדת עם המתנה אם יש צורך בכך. dem_key5.c ממחישה שימוש באופציה AH = 2 בדיקת סטטוס המקלדת.

התוכנית dem_key1.c

קטע ה-inline assembler

```
MOV AH,0
INT 16h
MOV scan_temp,AH
MOV inp_char,AL
```

הינו קריאת מקש (עם המתנה) והצבת ה-Scan code לתוך משתנה ה-C scan_temp והצבת ה-Ascii code לתוך משתנה ה-C inp_char. משם תוכנית ה-C מדפיסה אותם בצורה נוחה למשתמש.

התוכנית הראשית ממשת לולאה החוזרת על עצמה עד לרגע שהמשתמש לוחץ על המקש Esc שהתוכנית מזהה לפי Scan code = 1. בדוגמת הריצה הספציפית הזו נלחצו

```
 '9',
'Shift-9',
'r',
'Shift-r',
F1,
F10,
Shift-F1,
Shift-F10
```

שימו לב שללחיצות עצמם אין השתקפות בפלט (echo) - זה אף פעם לא אוטומטי ומיושם בתוכנה ברובד זה או אחר.
מהפלט אפשר לראות שללחיצות '9' ו-'Shift-9' אותו Scan code (10) וכנ"ל ללחיצות 'r' ו-'Shift-r' (19) אך הם נבדלים ב-Ascii code ('9') לעומת ')','r' לעומת 'R'). מי שיריץ את התוכנית יראה שמספרי ה-Scan code ניתנו למקשים בעיקרו של דבר לפי מיקומם במקלדת (ביחוד לאותם מקלדות של ה-PC מראשית שנות השמונים).

למקשים שאין להם Ascii code בדרך כלל מוחזר ב-AL אפס, ונסיון להדפיס אפס מתבטא ברווח. ללחיצת Esc יש משום מה Ascii code = 27 שמשום מה מוחק תו מהפלט, בגלל זה נמחק גרש בשורה האחרונה בפלט.

למקש F1 יש את Scan code = 59 ול-F10 יש את Scan code = 68 למעשה כל המקשים F1 - F10 יש Scan code = 59 - 68 בהתאמה. לחיצת המקשים הללו ביחד עם Shift גורמת לתוספת של 25 ל-Scan code אולי בגלל שאין למקשים הללו Ascii code ולא ניתן להבדיל ביניהם באמצעותו.

התוכנית dem_key5.c

התוכנית dem_key5.c משתמשת ברוטינת ה-BIOS INT 16h, AH = 2 לבדיקת סטטוס המקלדת. הרוטינה הזו מחזירה את בית הדגלים של המקלדת KB_FLAG ב-AL. התוכנית גם קוראת 2 הבתים בכתובות אבסולוטיות 417h (1041) ו-418h (1042) כלומר קצת מעבר ל-IV. כפי שאנחנו רואים בית 417h הינו בעצם KB_FLAG ואילו בית 418h הוא בעצם KB_FLAG_1. התוכנית מדפיסה את הסיביות באמצעות טכניקה בשפת C המאפשר גישה לסיביות של בתי זכרון הנקרא bit fields. שימו לב שתוכנית יכולה באמצעות KB_FLAG ו-KB_FLAG_1 לדעת אם המקשים Shift, Ctrl, Alt לחוצים (אפילו להבדיל בין 2 ה-Shift-ים), לדעת אם הנוריות Caps lock, Num lock, Scroll Lock דלוקות או לא ואפילו לדעת אם המקשים הללו לחוצים כרגע להבדיל ממצב הנוריות (הנוריות יכולות להיות כבויות גם אם המקש לחוץ). ניתן גם להבחין אם המקלדת נעולה ע"י מפתח (Hold) דבר שנתמך בחלק מה-PC-ים.

```
/* dem_key1.c - demostrate use of int 16h, AH = 0  */

#include <stdio.h>

void main(void)
 {

   unsigned int scan_code;
   char scan_temp, inp_char;

   printf("\nPress ESC to exit program\n");

   do {
        printf("Press any key (almost)\n:");

        asm  {
               MOV AH,0            ; /* BIOS read char from buffer option */
               INT 16h             ; /* BIOS read char from buffer        */
               MOV scan_temp,AH    ; /* Transfer scan code to program     */
               MOV inp_char,AL     ; /* Transfer char to program          */
             }

        scan_code = (unsigned int) scan_temp;

        if (scan_code == 1)
            printf("You pressed ESC, \n");

        printf("You pressed key assigned"
              " scan code = %d, char_value= '%c'\n",
                                    scan_code,  inp_char);


    } while(!(scan_code == 1));


 } /* main */
```

```
E:\>dem_key1

Press ESC to exit program
Press any key (almost)
:You pressed key assigned scan code = 10, char_value= '9'
Press any key (almost)
:You pressed key assigned scan code = 10, char_value= '('
Press any key (almost)
:You pressed key assigned scan code = 19, char_value= 'r'
Press any key (almost)
:You pressed key assigned scan code = 19, char_value= 'R'
Press any key (almost)
:You pressed key assigned scan code = 59, char_value= ' '
Press any key (almost)
:You pressed key assigned scan code = 84, char_value= ' '
Press any key (almost)
:You pressed key assigned scan code = 68, char_value= ' '
Press any key (almost)
:You pressed key assigned scan code = 93, char_value= ' '
Press any key (almost)
:You pressed ESC,
You pressed key assigned scan code = 1, char_value= '

E:\>
```

```
/* dem_key5.c - demostrate use of int 16h, AH = 2, including aux byte. */

#include <stdio.h>

typedef struct status_byte
{
  unsigned int right_shift_key : 1;
  unsigned int left_shift_key : 1;
  unsigned int ctrl_key : 1;
  unsigned int alt_key : 1;
  unsigned int scroll_lock_on : 1;
  unsigned int num_lock_on : 1;
  unsigned int caps_lock_on : 1;
  unsigned int insert_mode_on : 1;
}  STATUS_BYTE;

typedef struct aux_byte
{
  unsigned int unused : 3;
  unsigned int hold_on : 1; /* ctrl-num lock */
  unsigned int scroll_lock_depressed : 1;
  unsigned int num_lock_depressed : 1;
  unsigned int caps_lock_depressed : 1;
  unsigned int insert_key_depressed : 1;
}  AUX_BYTE;


void main(void)
 {

   STATUS_BYTE sbyte, sbyte1;
   char *p, *p1;
   AUX_BYTE  abyte;

   asm  {
          MOV AH,2            ; /* BIOS read keyboard status option */
          INT 16h             ; /* BIOS read keyboard status        */
          MOV BYTE PTR sbyte,AL    ; /* Transfer char to program   */
          PUSH ES
          MOV  AX,0
          MOV  ES,AX            /* ES = 0  */
          MOV  AL,ES:[418h]     /* read phisical locations 417h, 418h */
          MOV  BYTE PTR abyte,AL
          MOV  AL,ES:[417h]
          MOV  BYTE PTR sbyte1,AL
          POP ES
        }

   p = (char *) &sbyte;
   p1 = (char *) &sbyte1;

   printf("\n sbyte = %d, sbyte1 = %d\n", (int) *p, (int) *p1);
```

```
    printf("\nKeyboard status:\n\n");
    printf("Right shift: %d,\nLeft shift: %d,\nCtrl key: %d,\nAlt key: %d,\n",
             (unsigned int) sbyte.right_shift_key,
               (unsigned int) sbyte.left_shift_key,
                 (unsigned int) sbyte.ctrl_key,
                   (unsigned int) sbyte.alt_key);

    printf("Scroll lock on: %d,\nNum lock on: %d,"
       "\nCaps Lock on: %d,\nInsert mode: %d",
             (unsigned int) sbyte.scroll_lock_on,
               (unsigned int) sbyte.num_lock_on,
                 (unsigned int) sbyte.caps_lock_on,
                   (unsigned int) sbyte.insert_mode_on);
    putchar('\n');

    printf("\nAuxilary byte:\n\n");
    printf("Hold on: %d,\nScroll lock dep: %d,\nNum lock dep: %d,"
                "\nCaps Lock dep: %d,\nInsert key dep: %d",
             (unsigned int) abyte.hold_on,
             (unsigned int) abyte.scroll_lock_depressed,
               (unsigned int) abyte.num_lock_depressed,
                 (unsigned int) abyte.caps_lock_depressed,
                   (unsigned int) abyte.insert_key_depressed);
    putchar('\n');

  } /* main */
```

```
E:\>DEM_KEY5

 sbyte = 97, sbyte1 = 97

Keyboard status:

Right shift: 1,
Left shift: 0,
Ctrl key: 0,
Alt key: 0,
Scroll lock on: 0,
Num lock on: 1,
Caps Lock on: 1,
Insert mode: 0

Auxilary byte:

Hold on: 0,
Scroll lock dep: 1,
Num lock dep: 1,
Caps Lock dep: 0,
Insert key dep: 0

E:\>
```

תוכניות דוגמא testslp1.c, mysleep1.asm

התוכנית המשולבת הזו ממחישה איך ב-DOS תוכנית אפליקציה יכולה להתשמש במנגנון הפסיקות לצרכיה ובאופן עקרוני איך מערכות הפעלה עושות זאת היום. העיקר בתוכניות הדוגמא הללו הוא מימוש mysleep, מעין מימוש פרטי של הרוטינה sleep של שפת C. כמו ב-sleep הסטנדרטי mysleep מקבלת פרמטר שלם אותו היא מפרשת כזמן הרדמה בשניות ועוצרת את התוכנית הקוראת לה לזמן הזה. זהו שירות שימושי למדי המאפשרת לתוכנית לממש אלמנטים זמניים (למשל תצוגה זמנית על המסך).

המימוש של השרות הזה כאן הוא באמצעות השתלטות על פסיקה 8 (פסיקת הזמן) ומתוך הנחה שהפסיקה מתרחשת 18.2065 פעמים בשניה.

מה שהמימוש של mysleep עושה למעשה הוא קביעת רוטינה פנימית שלו Timer בתור הרוטינה מטפלת בפסיקה 8 (תוך שימור כתובת רוטינת הטיפול המקורית), התמרת זמן ההרדמה משניות לפסיקות שעון ע"י הכפלה ב-182065 וחילוק ללא שארית ב-10000. את התוצאה mysleep שומר ב-AX. זהו דיוק מספיק בכדי לממש עיכוב בדיוק גבוה למספר רב למדי של שניות.

mysleep מאתחל מונה בשם counter באפס. הרוטינה Timer מקדם את המונה הזה פעם אחת לכל פסיקת שעון, ו-mysleep ממש לולאה המשווה את זמן העיכוב בפסיקות שעון לערך של המונה בלולאה שהוא יוצא ממנה רק כאשר המונה עובר את זמן ההרדמה. למעשה העיכוב בזמן מתבצע בפועל כאן.

פקודות הלולאה הם

```
MOV Counter,0
Delay:
CMP AX,Counter
JA Delay
```

מיד אחרי יציאה מהלולאה הזו mysleep משחזר את כניסה 8 ב-IV וחוזר לקוד הקורא.

נקודה טכנית היא העובדה שפסיקה 8 היא מאותם פסיקות שיש להודיע לחומרה שהיא טופלה ואחת הדרכים לדאוג לכך היא לקרוא רוטינת הטיפול בפסיקה 8 המקורית. לפיכך הקוד של Timer הוא

```
Timer PROC FAR
;
  PUSHF
  CALL DWORD PTR Timer_Offs
  ;
  INC Counter
  ;
  IRET
;
Timer ENDP
```

כאן אני מנצל את העובדה ש-Timer_Seg נמצא מיד אחרי Timer_Offs.
השימוש בסגמנט הקוד לאכסון מידע הוא נוח במימוש פסיקות, כי על הערך של CS תמיד אפשר לסמוך.

שימו לב שכתובת אבסולוטית 32 (= 4*8) עד 35 היא הכתובת כל כניסה מספר 8 ב-IV. הפקודות

```
MOV AX,ES:[32]
MOV Timer_Offs
MOV AX,ES:[34]
MOV Timer_Seg,AX
```

הם פקודות השימור של הכניסה המקורית ואילו הפקודות

```
CLI
MOV WORD PTR ES:[32],OFFSET Timer
MOV ES:[34],CS
STI
```

הם קביעת הרוטינה Timer כרוטינת הטיפול בפסיקה. כאן אני מנצל את העובדה שאני יודע ש-Timer ו-mysleep נמצאים באותו תאור סגמנט, דבר שלא היה בהכרח נכון אם הם היו בקבצים נפרדים.

```
/* testslp1.c - test mysleep */

#include <stdio.h>

extern void mysleep( int secs );

void main()
{
  int n=20;

  printf("Before mysleep(%d):\n", n);
  mysleep(n);
  printf("After mysleep(%d):\n", n);

}  /* main */
```

```
C:\temp>tcc -ml -r- testslp1.c mysleep1.asm
Turbo C++ Version 3.00 Copyright (c) 1992 Borland International
testslp1.c:
mysleep1.asm:
Turbo Assembler  Version 3.1  Copyright (c) 1988, 1992 Borland
International

Assembling file:   mysleep1.ASM
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  390k

Turbo Link  Version 5.0 Copyright (c) 1992 Borland International

Available memory 4107872

C:\temp>testslp1
Before mysleep(20):
After mysleep(20):

C:\temp>
```

```
; mysleep1.asm - demonstrate Interrupt service routine
;
     .MODEL LARGE
     .STACK  100h
     .DATA
C182       DD 182065
C100       DD 10000
      ;
     .CODE
     .386
Timer_Offs DW 0
Timer_Seg DW  0
Counter   DW  0
;
; My Ctrl-Break ISR
;
Timer PROC FAR
;
     PUSHF
     CALL DWORD PTR Timer_Offs
     ;
     INC  Counter
Return:
     IRET    ;
;
Timer ENDP
```

```
;
;
_mysleep  PROC  FAR
      PUBLIC _mysleep
      PUSH BP
      MOV  BP,SP
      PUSH ES
;
      MOV  AX,0
      MOV  ES,AX
         ;
         ;
;
      MOV  AX,ES:[32]          ; Preserve original ISR pointers
      MOV  Timer_Offs,AX
      MOV  AX,ES:[34]
      MOV  Timer_Seg,AX
      ;
      ; Change Timer pointers
      CLI
      MOV  WORD PTR ES:[32],OFFSET Timer
      MOV  ES:[34],CS
      STI
      MOV  AX,DS
      MOV  ES,AX
                  ;
;
;
      XOR  EAX,EAX
      MOV  AX,[BP+6]    ; Retrieve secs parameter
          ; Compute AX = secs * 18.2065 to convert to ticks
      MUL  C182     ;
      DIV  C100

      MOV  CS:[Counter],0            ; Init counter
Delay1:
      ;
      CMP  AX,CS:[Counter]   ; wait n secs
      JG  Delay1
;
;  Return to caller
      ; Restore old Timer
      ;
                  ;
      MOV  AX,0
      MOV  ES,AX
      MOV  AX,Timer_Offs    ;
      CLI
      MOV  ES:[32],AX
      MOV  AX,Timer_Seg    ;
      MOV  ES:[34],AX
      STI
                  ;
      POP ES
      POP BP
      RET     ; return to caller
      ENDP _mysleep
      END
```

תוכניות דוגמא msleep1.c, msleep2.c

התוכנית msleep1.c היא תוכנית מימוש של הרדמה הכתוב כמעט כולו ב-C.

כמו במימוש הקודם של ההרדמה הוא מקבל הרדמה בשניות, התוכנית מחשבת קירוב לזמן הזה ביחידות של פסיקות שעון (מכפילה ב-182 ומחלקת ללא שארית ב-10), מאתחלת מונה גלובלי בערך הזה ובהתרחש כל פסיקה שעון מפחיתה את המונה באחד עד שהוא מתאפס.

מה שמיוחד במימוש הזה הוא העובדה שרוטינת ההרדמה מבצעת בעצמה את ההודאה לחומרה שהפסיקה טופלה, ולא מסתמכת על שימוש עקיף של קוד BIOS מקורי. פקודות האסמבלי:

```
MOV AL,20h
OUT 20h,AL
```

הינם הפקודות הנחוצות להודעה לחומרה שפסיקת הזמן טופלה ברוב המחשבים האישיים. בכדי להמנע מהצורך לממש קוד כזה, שעשוי להיות נכון רק עבור פלטפורמות מסוימות, המשמיה של העברת הודעה לחומרה נעשבה בדרך כלל ע"י קריאה פנימית לרוטינת ה-BIOS המקורית. זה בדיוק מה שהתוכנית msleep2.c עושה. הפקודות שבהם אני קורא לרוטינת ה-BIOS המקורית בכדי שתודיע לחומרה שהפסיקה טופלה (ואגב כך עשויה לעשות דברים נוספים שלא ידועים לי) הם:

```
PUSHF
CALL DWORD PTR Int8Save
```

```
/* msleep1.c - Implement sleep myself, Pure C version  */

#include <stdio.h>
#include <dos.h>

volatile int My_Sleep_Counter;

void interrupt (*Int8Save) (void);  /* Pointer to function */

void interrupt My_Int8_Handler(void)
   {
     asm {              /* Notify hardware:
                           Interrupt has been serviced */
          PUSH AX
          MOV AL,20h
          OUT 20h,AL
          POP AX
         }

     My_Sleep_Counter--;
   } /* My_Int8_Handler */

void my_sleep(int secs)
{
Int8Save = getvect(8);               /* Preserve old pointer */
setvect(8, My_Int8_Handler);           /* Set entry to new handler */

My_Sleep_Counter = secs*182/10;

while(My_Sleep_Counter > 0)
    ;
setvect(8,Int8Save);               /* Restore old pointer */

}  /* my_sleep */

void main(void)
 {
  int secs;

   printf("Enter sleep time in seconds:\n");
   scanf("%d", &secs);

   system("time");
   printf("sleeping ---\n");

   my_sleep(secs);

   system("time");

   printf("Terminating ...\n");

 } /* main */
```

```
E:\USERS\EYTAN\ASM>tcc -ml -r- msleep1.c
Turbo C++ Version 3.00 Copyright (c) 1992 Borland International
msleep1.c:
Turbo Link  Version 5.0 Copyright (c) 1992 Borland International

        Available memory 4121432

E:\USERS\EYTAN\ASM>msleep1.exe
Enter sleep time in seconds:
60
Current time is  16:14:47.13
Enter new time:
sleeping ---
Current time is  16:14:48.18
Enter new time:
Terminating ...

E:\USERS\EYTAN\ASM>
```

```
/* msleep2.c - Implement sleep myself, Pure C version  */

#include <stdio.h>
#include <dos.h>

volatile int My_Sleep_Counter;

void interrupt (*Int8Save) (void);  /* Pointer to function */

void interrupt My_Int8_Handler(void)
   {
     asm {              /* Notify hardware:
                           Interrupt has been serviced */
          PUSHF
          CALL DWORD PTR Int8Save
         }

     My_Sleep_Counter--;
   } /* My_Int8_Handler */

void my_sleep(int secs)
{
Int8Save = getvect(8);                /* Preserve old pointer */
setvect(8, My_Int8_Handler);            /* Set entry to new handler */

My_Sleep_Counter = secs*182/10;

while(My_Sleep_Counter > 0)
    ;
setvect(8,Int8Save);                /* Restore old pointer */

}  /* my_sleep */

void main(void)
 {
  int secs;

   printf("Enter sleep time in seconds:\n");
   scanf("%d", &secs);

   system("time");
   printf("sleeping ---\n");

   my_sleep(secs);

   system("time");

   printf("Terminating ...\n");

 } /* main */
```

```
E:\USERS\EYTAN\ASM>tcc -ml -r- msleep2.c
Turbo C++ Version 3.00 Copyright (c) 1992 Borland International
msleep2.c:
Turbo Link  Version 5.0 Copyright (c) 1992 Borland International

        Available memory 4121432

E:\USERS\EYTAN\ASM>msleep2
Enter sleep time in seconds:
60
Current time is  16:17:01.20
Enter new time:
sleeping ---
Current time is  16:18:02.72
Enter new time:
Terminating ...

E:\USERS\EYTAN\ASM>
```