בפסיקה מספר 16h הינה פסיקת התוכנה של המקלדת. תוכניות שמעוניינות באינפורמציה מהמקלדת פונות אליה, בדרך כלל בעקיפין דרך מערכת ההפעלה והיא כאילו התוכנית הסטנדרטית לקרוא מידע מהמקלדת. זוהי פסיקת תוכנה של ה-BIOS ובעיקרו של דבר היא מהווה מעין קוד המכיר את מבנה הנתונים של ה-BIOS ומתפקדת התאם.

בשטח זיכרון מיוחד בהתחלה של הזיכרון של המחשב (מיד אחרי השטח של IV, כלומר מכתובת 1024 ואילך) ישנו שטח זיכרון של ה-BIOS שמכיל בין השאר שטח המשקף את מצב המקלדת, איזה מקשים נלחצו וטרם נקראו ע"י שום תוכנית, מצב הנורות (Caps Lock, Insert, Scroll, Num), מצב מקשי הסטטוס (ה-Shift, Alt, Ctrl) וכו'. למשל בבתים בכתובות מוחלטות 417h (1041) ו- 418h (1042) ישנם בתים המהווים משתני דגלים של מצב הנורות ומקשי הסטטוס. נוסף לכך יש שם, חוצץ לשמירת מידע על עד ל-20 לחיצות מקשים שלא נקראו עדיין (מעבר לכך התגובה ללחיצות נוספות יהיה צפצוף). השטח הזה מתוחזק ומעודכן בעיקר ע"י ה-ISR של פסיקה מספר 9, פסיקת החומרה של המקלדת. המידע שנשמר שם הוא על לחיצות על (כמעט) כל המקשים של המקלדת, כולל המקשים F1 - F12, Page Down, Esc, Ins, Del ... וכו'.

כאשר תוכנית מבקשת מ-INT 16h אינפורמציה על לחיצת מקש, האיפורמציה מגיעה בשתי צורות: קוד Ascii וקוד Scan. קוד ה-Ascii הוא בדרך כלל מה שאנחנו צריכים, למשל אם נלחץ על המקש המסומן A נקבל או את הקוד Ascii 'a' או 'A' בהתאם למצב נורת ה-Caps Lock ומקשי ה-Shift וכו'. קוד ה-Scan הוא מספר יחודי לכל מקש, בתחום 1 - 127. צריך לזכור שיש מקשים שאין להם קוד Ascii (כמו F1, Esc, Page Down) ויש קודי Ascii שיש להם יותר ממקש אחד (למשל הספרות העשרוניות, הסימנים +, -, *, /).

ל- INT 16h אופציות רבות, הנבחרות לפי הערך של האוגר AH ברגע הקריאה, השימושיות ביותר מבחינתינו הם:

INT 16h, AH = 0, קריאת מקש:

קריאה ל-INT 16h כאשר AH = 0 פירושו בקשה לקבלת מידע על לחיצת מקש של המשתמש. אם אין מידע מסוג זה בהמתנה, החזרה מהקריאה תתעקב עד שחיצה כזו תתבצע (כלומר התוכנית <u>תעצר</u> לזמן בלתי מוגבל). עם החזרה (בין עם אחרי המתנה ובין שלא) יהיו באוגר AX האינפורמציה המבוקשת: ב-AL יהיה קוד ה-Ascii של המקש שנלחץ ואילו ב-AH יהיה קוד Scan של המקש. אותה לחיצת מקש שהמידע עליה מוחזרת לקורא ל-INT 16h נחשבת ל"נקראה" קרי "מבוטלת" כלומר שקריאה נוספת ל-INT 16h אופציה AH = 0 לא תתיחס אליה ותקרא את האינפורמציה

על הלחיצה הבאה. זה נראה מובן מאליו, אבל זה לא נכון לאופציה הבאה.

,INT 16h
AH = 1, עיון במקש:

האופציה הזו דומה לאופציה AH = 0, אבל במספר הבדלים מהותיים. ראשית, הקריאה תמיד חוזרת מיד לקורא ל-INT 16h, גם אם אין אינפורמציה על לחיצת מקש שטרם נקראה. התוכנית הקוראת יכולה להבחין אם היתה לחיצת מקש או לא לפי הערך של הדגל ZF, ZF = 1 אם לא היתה אינפורמציה של לחיצת מקש בהמתנה לקריאה, אם היתה ZF = 0. במידה והיתה אינפורמציה ללחיצת מקש ממתינה, היא תוחזר ב-AX ממש כמו ב-INT 16h אופציה AH = 0, אך בניגוד למקרה הקודם הקריאה ל-INT 16h אופציה AH = 1 אינה "מבטלת" את הלחיצה. הקריאה הבאה ל-INT 16h באופציה AH = 0 או AH = 1 יקראו את אותו מקש עצמו. כלומר קריאה ל-INT 16h אופציה AH = 1 היא קריאה "לא מחייבת" של המקש, רק מעין הצצה לבדוק מה יש שם, אם בכלל. למשל, אם נרצה לבדוק אם יש אינפורמציה על לחיצת מקש ממתינה ובמידה ויש לקרוא אותה אבל מבלי להמתין ללחיצה כזו במידה ואין, אנחנו נקרא קודם ל-INT 16h אופציה AH = 1, נבדוק ש-ZF = 0 ובמידה וזה המצב, נקרא ל-INT 16h אופציה AH = 0.

הקוד יכול להראות כך:

```
MOV AH,1
INT 16h
JZ No_Info1
 MOV AH,0
 INT 16h
 JMP With_Info1
No_Info1:

  ....

With_Info1:
```

,INT 16h
AH = 2, סטטוס המקלדת. מחזיר ב-AL אחד משני בתי הדגלים של המקשים המיוחדים (נורות ה-Caps Lock, Num, Scroll, מצב מקשי ה-Shift, Alt, Ctrl). מידע נוסף ניתן לקרוא מכתובת מוחלטת 418h או ע"י קריאה ל-INT 16h אופציה AH = 12h.

,INT 16h

5 = AH, הדמית מקש. אפשר בתוכנית ליצור מצב שבו "כאילו" נלחץ מקש כלשהוא, כאשר האינפורמציה של לחיצת המקש אותו רוצים לדמות מועברת דרך CX: CL יהיה קוד ה-Ascii ו-CH קוד ה-Scan.

Type 16 Interrupt Operations

The Type 16 (Keyboard I/O) interrupt calls KEYBOARD_IO, which starts at location F000:E82E. This routine lets you select from three different operations, based on a value in AH:

- If AH=0, KEYBOARD_IO reads the scan code of the next key in the buffer into AH and its character code into AL, then advances the buffer pointer. If the buffer is empty, KEYBOARD_IO waits for a key to be pressed before proceeding.
- If AH=1, KEYBOARD_IO returns the status of the keyboard buffer in the Zero Flag (ZF). If the buffer is empty, ZF is 1. If any key codes are available for reading, ZF is 0. If ZF is 0, the next available character is in AX, and the entry remains in the buffer.
- If AH=2, KEYBOARD_IO returns a keyboard status byte in AL. A description of this byte follows.

The KEYBOARD_IO routine affects only AX and the flags.

The upper half of Figure 6-6 shows the arrangement of the status byte returned by the AH=2 option. In this byte (KB_FLAG in the BIOS listing) the upper four bits tell you whether various keyboard modes are on (1) or off (0), and the lower four bits tell you whether the Alt, Ctrl, or shift key is being pressed.

The lower half of Figure 6-6 shows a companion byte to KB_FLAG that gives additional keyboard status information. The KEYBOARD_IO routine uses this second byte (KB_FLAG_1) internally but provides no way to read it into a register. However, KB_FLAG_1 follows KB_FLAG in the BIOS, so to find out how KB_FLAG_1 is configured, you read the contents of location 41811 (KB_FLAG is at 41711).

The first KEYBOARD_IO option, AH=0, is convenient for setting up *interactive* operations with the computer. In such operations you essentially tell the computer to wait for an operator to type something at the keyboard before proceeding. Let's take a look at some possible applications.

A Single-Key Read Operation

Suppose your program has displayed a message that instructs the operator to press either "Y" or "N" to signify Yes or No. A Y response makes the program jump to a set of instructions labeled YES, and an N makes it jump to instructions labeled NO. Any other key makes the program wait until it receives either Y or N. This sequence should do the job:

```
        STI                     ;Enable interrupts
GET_KEY: MOV    AH,0            ;Read a key
        INT     16H
```

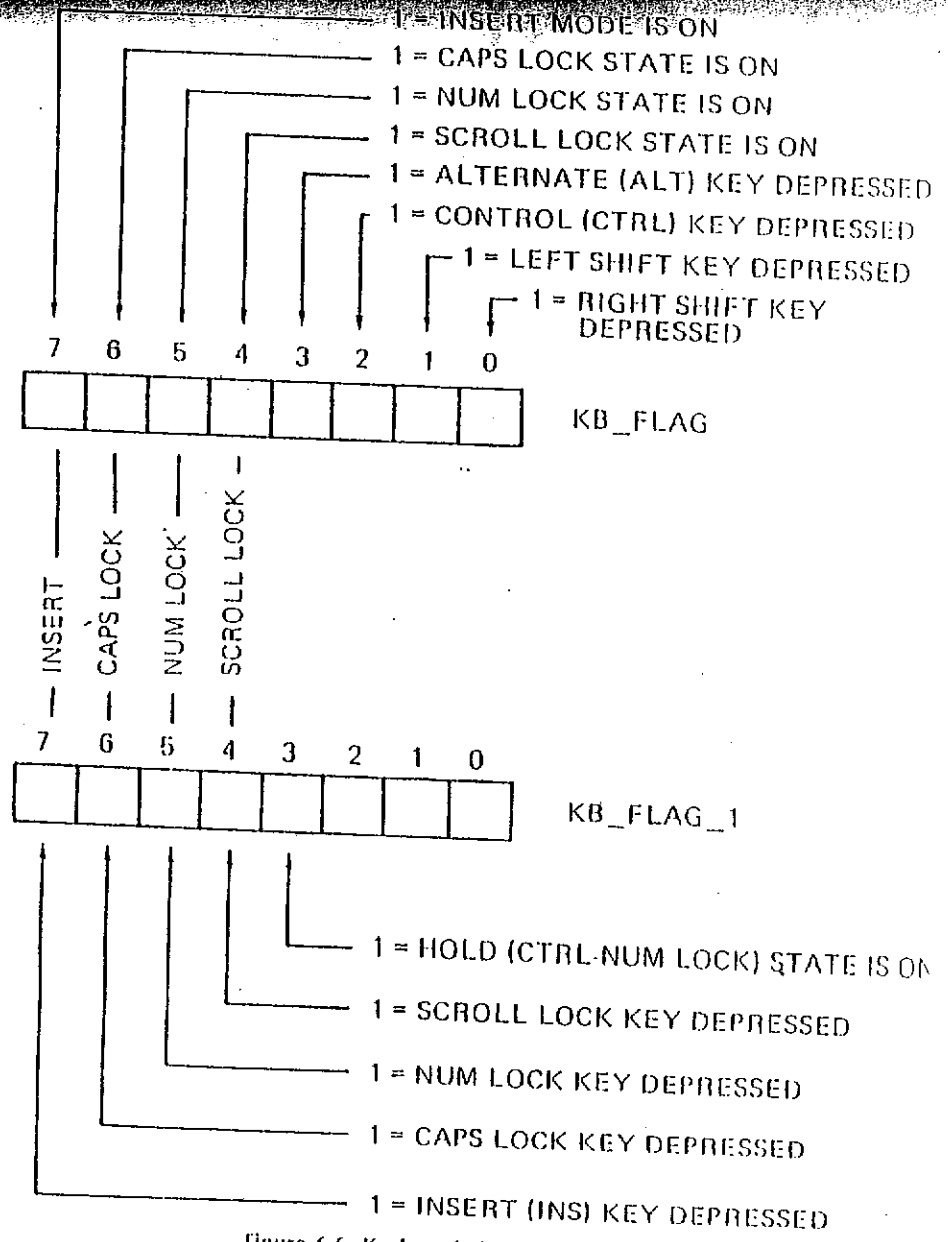

Figure 6-6. Keyboard I/O shift status bytes.

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 מממשת 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.

```
/* 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:\>
```

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_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:\>
```