Marian WRZESIEŃ
Przemysłowy Instytut
Automatyki i Pomiarów PIAP
W a r s z a w a

PRZERWANIA PROGRAMOWE W PC

Przedstawiono istotę przerwań sprzętowych i programowych dla komputera PC. Omówiono przerwania programowe dla programów napisanych w języku C i skompilowanych przy użyciu pakietów MSC i QUICK firmy Microsoft Corporation oraz Turbo C i Turbo C++ firmy Borland International. Aplikacje przerwań zademonstrowano na przykładzie dziesięciu funkcji przerwania 0x33 (mysz), których interpretacja została umieszczona w przedstawionym programie testowym.

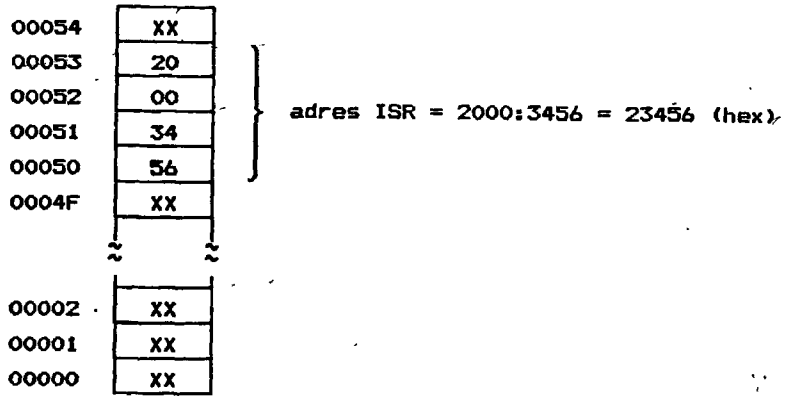
1. WSTĘP

W komputerze PC występują trzy typy przerwań: przerwania sprzętowe, przerwania programowe oraz stany wyjątkowe procesora. Jakkolwiek artykuł skupia się na przerwaniach programowych, to omówiono także pozostałe typy przerwań. Każdy z procesorów Intel: 8088, 8086, 80286, 80386 i 80486 ma m.in. dwa wyprowadzenia: INTR dla przerwania maskowalnego oraz NMI dla przerwania niemaskowalnego.

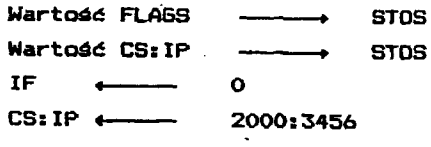
2. MECHANIZM PRZERWANIA MASKOWALNEGO

Przerwanie maskowalne jest to takie przerwanie, którego obsługa może być zezwolona (enable) bądź zabroniona (disable), w zależności od stanu znacznika IF (interrupt enable flag) zawartego w rejestrze procesora FLAGS i sterowanego programowo. W przypadku, w którym na skutek określonego zdarzenia do programowalnego sterownika przerwań (PIC) zostanie podany sygnał żądania przerwania (IRQ), sterownik PIC wystawia procesorowi sygnał przerwania (INTR). Pojawienie się sygnału INTR (przy zezwoleniu

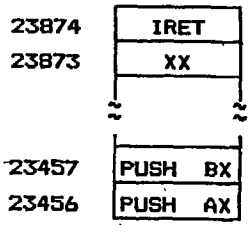
1) wektor przerwań jest równy 14h,
2) offset dla tablicy wektorów przerwań jest równy 50h,
3) odczyt adresu ISR z tablicy przerwań



4) przypisanie wartości podczas przerwania



5) wykonanie procedury przerwania



6) podstawienie pierwotnych wartości

Rys. 1. Kolejne czynności przy obsłudze przerwania

(IF = 1)) powoduje, że po zakończeniu bieżącej instrukcji procesora następuje przejście do obsługi przerwania. Obsługa przerwania rozpoczyna się wysłaniem przez procesor sygnału potwierdzenia INTA do sterownika PIC. Podczas drugiego cyklu INTA sterownik PIC wystawia na szynę danych 8-bitowy wektor przerwań (IV), który zostaje wykorzystany przez procesor do odczytania adresu procedury przerwania (ISR). Adres ten jest przechowywany w tablicy wektorów przerwań. Tablica wektorów przerwań rozpoczyna się na początku obszaru pamięci (SEG = Oh) i zawiera 256 4-bajtowych adresów ISR. Poprzez dwukrotne przesunięcie bitowe wartości IV procesor generuje offset dla tej tablicy, wskazując tym samym miejsce odczytu adresu ISR. Adres ISR, uformowany jako wskazanie (pointer) w standardzie 80x86 tj. segment: offset, umożliwia skok procesora do miejsca przechowywania procedury obsługi przerwania oraz jej wykonanie. Czynności te poprzedzone są zabezpieczeniem zawartości zarówno wewnętrznych rejestrów, jak i adresu kodu wykonywanego w momencie pojawienia się przerwania. Rysunek 1 przedstawia kolejne czynności przy obsłudze przykładowego przerwania 14 h. Zwrócono tu uwagę na odłożenie wartości rejestru FLAGS na stos i wyzerowanie bitu IF w rejestrze FLAGS. Prawidłowa realizacja przerwania wymaga, by po zakończeniu jego obsługi program powrócił do miejsca, w którym przerwano jego działanie (CS:IP) oraz odnowił wartość rejestru FLAGS poprzez pobranie jej ze stosu. Jest to realizowane instrukcją IRET umieszczoną na końcu procedury przerwania. Chociaż w maszynach z procesorami 286, 386, 486 możliwych jest 15 wejść IRQi, to sterownik PIC obsługujący te wejścia, po uwzględnieniu priorytetu przerwań, wystawia zawsze tylko jeden wektor przerwań.

3. MECHANIZM PRZERWANIA NIEMASKOWALNEGO

Przerwanie niemaskowalne jest obsługiwane po pojawieniu się sygnału na wejściu procesora NMI. W takim przypadku procesor nie wysyła sygnału potwierdzania INTA, lecz od razu przechodzi do obsługi przerwania. Ponieważ przerwanie NMI ma wyższy priorytet niż dowolne przerwanie INTR, więc jest ono wykorzystywane w przypadkach krytycznych, takich jak na przykład błąd parzystości pamięci.

4. STANY WYJĄTKOWE PROCESORA

Tablica 1. **Stany wyjątkowe procesora**

Stany wyjątkowe	wektor	88/86	286	386/486
błąd dzielenia	00h	+	+	+
pojedynczy krok	01h	+	+	+
sprawdzenie obszaru tablicy	05h	—	+	+
błędny kod operacji	06h	—	+	+
urządzenie niedostępne	07h	—	+	+
podwójny stan wyjątkowy	08h	—	+	+
przepełniony segment koprocesora	09h	—	+	+
nieważny segment stanu zadania	0Ah	—	+	+
brak segmentu	0Bh	—	+	+
przepełniony segment stosu	0Ch	—	+	+
naruszenie głównego zabezpieczenia	0Dh	—	+	+
błąd stronicowania	0Eh	—	—	+
błąd koprocesora	10h	—	+	+

Oznaczenia: + występuje, — nie występuje

5. PRZERWANIA SPRZĘTOWE

Przerwania sprzętowe są realizowane przez sterownik PIC 8259A. Śledzi on stan realizacji przerwań, kontroluje priorytety, zabezpiecza przed przyjęciem przerwania w trakcie wykonywania tego samego przerwania oraz wystawia wektory przerwań. W tablicy 2 przedstawiono użycie przerwań sprzętowych w PC. Priorytet tych przerwań jest tym wyższy, im niższa jest wartość indeksu i w IRQ_i. Mechanizm obsługi przerwań sprzętowych jest zgodny z opisem podanym w p. 2.

Tablica 2. **Użycie przerwań sprzętowych w systemie PC**

IRQ	Przypisanie stałe	Normalne użycie
NMI	błąd parzystości pamięci	
0	zegar systemowy	
1	klawiatura	
2 (9)		sieć lub karta VGA
3		COM2 lub COM4
4		COM1 lub COM3
5		dysk twardy XT lub LPT2
6		sterownik dysku miękkiego
7		LPT1
8	zegar czasu rzeczywistego AT	
10		ogólne
11		ogólne
12		ogólne
13	koprocesor matematyczny AT	
14		sterownik dysku twardego AT
15		ogólne

Przerwanie IRQ9 jest stosowane zamiast IRQ2 w komputerach, w których pierwotna ośmiowejściowa wersja sterownika PIC została rozszerzona o sterownik slave, dołączany do wejścia IRQ2 sterownika master. Prawa kolumna tablicy wskazuje na przerwania sprzętowe ogólnego zastosowania, ze wskazaniem na ich najczęstsze przypisanie.

6. PRZERWANIE PROGRAMOWE

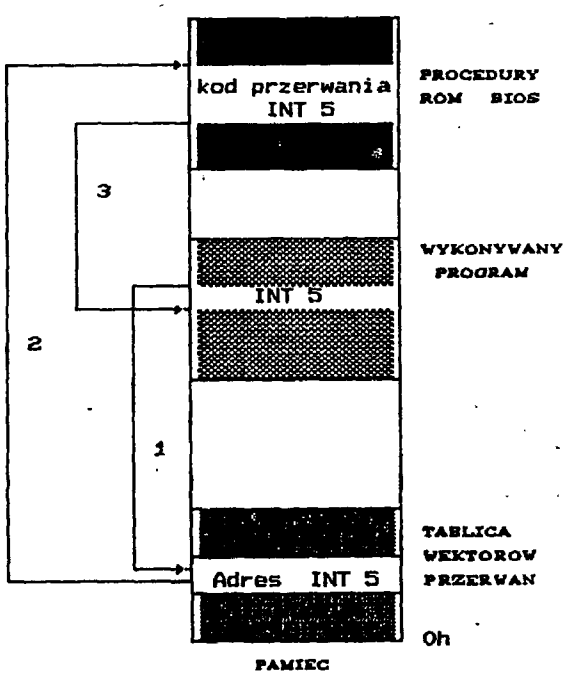
Przerwanie programowe jest realizowane na skutek wystąpienia w programie instrukcji INT <int_number>, gdzie int_number jest numerem przerwania. To przerwanie jest traktowane przez procesor tak jak INTR, lecz różni się tym, że wektor przerwania jest określony programowo. Stąd też, przy tego typu przerwaniu, nie jest

wymagane wystawianie przez procesor sygnału potwierdzenia INTA dla pobrania wektora przerwania. Przerwanie to ma priorytet wyższy od przerwania INTR i NMI; obejmuje ono dostęp do funkcji BIOS i DOS komputera PC. Schemat realizacji tego przerwania przedstawiono na rys. 2.

1. pobranie adresu przetwarzania z tablicy wektorów przerwań

2. skok do podprogramu przerwania

3. powrót do programu głównego w wyniku wykonania instrukcji IRET w podprogramie



Rys. 2. Schemat realizacji przerwania software'owego

7. OPROGRAMOWANIE PRZERWAŃ

Oprogramowanie przerwań zostanie omówione dla kompilatorów języka C: Turbo C (wersja 1, 1.5, 2), Turbo C++ (wersja 1, 2, 3) firmy Borland International oraz MSC (wersja 3, 4, 5, 6, 7) i Quick C (wersja 1, 2, 2.5) firmy Microsoft Corporation. Wymienione pakiety oprogramowania charakteryzują się identycznym podejściem przy uwzględnianiu przerwań i pomimo tego, że podejście to nie jest kompatybilne z normą ANSI oraz ISO/IEC 9899:1990(E), to pakiety te są wystarczająco reprezentatywne, by je

przyjąć jako wzorcowe. W celu omówienia realizacji przerwań wybrano niektóre funkcje przerwania 33h obejmującego obsługę myszy.

7.1. Realizacja przerwania programowego

W języku C wymagane argumenty są zazwyczaj dostarczane do wywoływanych funkcji przez stos, który jest również wykorzystywany do przekazywania programowi wartości zwracanych przez wywoływane funkcje. Przerwanie programowe ma charakter funkcji z tą różnicą, że argumenty, jak i wartość zwracana funkcji przerwaniowej, są przekazywane poprzez rejestry procesora. W języku C generowanie przerwania programowego umożliwia procedura (funkcja):

```
int int86(int_number, union REGS *inr, union REGS *outr);
```

której argumentami są numer przerwania i unie typu REGS, o postaci:

```
union REGS{
      struct WORDREGS x;
      struct BYTEREGS h;
```

przy czym struktury są określone następująco:

```
struct WORDREGS{
       int ax, bx, cx, dx, si, di, cflag, flags;
};
struct BYTEREGS{
       int ah, al, bh, bl, ch, cl, dh, dl;
};
```

Unia REGS inr umożliwia przekazywanie 16-bitowych słów rejestrowych (argumentów przerwania) AX, BX, CX, DX poprzez strukturę WORDREGS, oraz 8-bitowych połówek tych słów AH, AL, BH, BL, CH, CL, DH, DL poprzez strukturę BYTEREGS, w zależności od wymaganego konkretnego przerwania, a unia REGS outr zapewnia zwracanie odpowiednich wartości rejestrowych będących wynikiem przerwania poprzez analogiczne struktury stanowiące elementy tej unii.

W przypadkach, w których przerwanie wymaga przekazania wartości segmentowych

Tablica 3. Funkcje przerwania myszy

00h	reset/kontrola istnienia sterownika myszy
01h	wyświetlanie kursora myszy
02h	ukrycie kursora myszy
03h	odczyt pozycji i statusu przycisków myszy
04h	przesunięcie kursora myszy do wskazanej pozycji
05h	odczyt liczby wciśnięć przycisków myszy
06h	odczyt liczby zwolnień przycisków myszy
07h	zdefiniowanie zakresu ruchu poziomego kursora myszy
08h	zdefiniowanie zakresu ruchu pionowego kursora myszy
09h	zdefiniowanie kształtu kursora myszy w trybie graficznym
0ah	zdefiniowanie maski kursora myszy w trybie tekstowym
0bh	odczyt ostatniego zakresu ruchu kursora myszy
0ch	zdefiniowanie programu obsługi (handlera) zdarzenia
0dh	zezwolenie na emulację pióra świetlnego
0eh	zabronienie emulacji pióra świetlnego
0fh	zdefiniowanie szybkości ruchu kursora myszy
10h	zdefiniowanie obszaru eliminującego kursor myszy
13h	zdefiniowanie przyspieszenia ruchu kursora myszy
14h	wymiana programów obsługi (handlerów) zdarzeń
15h	odczyt statusu bufora
16h	zapamiętanie statusu myszy
17h	odtworzenie statusu myszy
18h	zainstalowanie programu obsługi (handlera) zdarzeń myszy
19h	odczyt adresu programu obsługi (handlera) zdarzeń myszy
1ah	zdefiniowanie czułości odczytu położenia kursora myszy
1bh	odczyt czułości odczytu położenia kursora myszy
1ch	zdefiniowanie współczynnika wartości zmiany położenia
1dh	zdefiniowanie numeru strony wyświetlania kursora myszy
1eh	odczyt numeru strony wyświetlania kursora myszy
1fh	zdeaktywowanie sterownika (drivera) myszy
20h	zaktywowanie sterownika (drivera) myszy
21h	zresetowanie sterownika (drivera) myszy
24h	kontrola typu myszy i numeru IRQi

ES, CS, SS i DS lub, gdy jawnie specyfikuje się rejestry segmentu miejsca dostępu w pamięci, język C dostarcza procedury (funkcji):

```
int int86x(int_number, union REGS *inr, union REGS *outr,
           struct SREGS *segr);
```

z dodatkowym, w stosunku do int86(...), argumentem typu struct SREGS o postaci:

```
struct SREGS{
        int es, cs, ss, ds;
}
```

Ewentualny błąd wykonania przerwania wskazywany jest przez umieszczanie niezerowej wartości w komórce outr.x.flags oraz przypisanie zmiennej globalnej **doserrno** wartości kodu błędu. Wartości stanowiące efekt przerwania przekazywane są przez rejestry unii REGS outr.

Zwykle jeden numer przerwania int_number umożliwia dostęp do całej kategorii usług szczególnych (funkcji przerwania), wyselekcjonowanych odpowiednimi wartościami przekazanymi przez rejestry. W tablicy 3 przedstawiono zestaw funkcji przerwania myszy o numerze przerwania int_number = 33h. W przypadku tego przerwania, o wyborze funkcji przerwania decyduje wartość wpisana do rejestru AX (inr.x.ax). Dla ilustracji, niektóre z tych funkcji przyjęto jako przykłady realizacji przerwania programowego w języku C.

8. WYBRANE FUNKCJE PRZERWANIOWE

W celu zaprezentowania realizacji przerwań zostaną podane przykładowe funkcje zawierające pierwsze dziesięć pozycji tablicy 3. Definicje tych funkcji wykorzystują wartości stałe, deklaracje i prototypy zestawione w poniższym pliku nagłówkowym "mouse.h". Plik "mouse.h" nie jest stałym elementem języka C.

```
/*                          mouse.h                          */
#define MOUSE                                   0x33
#define MOUSE_RESET                             0x0
#define MOUSE_SHOW_CURSOR                       0X1
#define MOUSE_HIDE_CURSOR                       0X2
#define MOUSE_GET_POS_BUTTON                    0x3
#define MOUSE_PUT_POS_CURSOR                    0x4
```

```
#define MOUSE_GET_DATA_BUT_PRESSED                    0x5
#define MOUSE_GET_DATA_BUT_RELEASED                   0x6
#define MOUSE_SET_HCR                                 0x7
#define MOUSE_SET_VCR                                 0x8
#define MOUSE_SET_GRAPH_CURSOR                        0x9
struct  MOUSEE {
        int row;
        int column;
        int status;
        int nr_pressed;
        int nr_released;
        int button;
};
void mouse_reset( struct MOUSEE *my);
void mouse_show_cursor(void);
void mouse_hide_cursor(void);
void mouse_get_but_pos(struct MOUSEE *my);
void mouse_put_pos_cursor(int column, int row);
void mouse_get_data_but_pressed(int button, struct MOUSEE *my);
void mouse_get_data_but_released(int button, struct MOUSEE *my);
void mouse_set_hcr(int min_column, int max_column);
void mouse_set_vcr(int min_row, int max_row);
void mouse_set_graph_cursor(const int cursor[32], int  hot_column,
                            int hot_row);
```

8.1. Funkcja zerująca

Ta funkcja ma za zadanie sprawdzić, czy mysz została zainstalowana i wskazać typ myszy (liczba klawiszy), oraz sprowadzić do stanu początkowego (zresetować) sprzęt i oprogramowanie związane z obsługą myszy. Funkcja ta:
— ustawia kursor w centrum ekranu (przerw. 04h),
— chowa kursor (przerw. 02h),
— kasuje obszar eliminujący kursor (przerw. 10h),
— ustawia atrybut maski kursora tekstowego na inwersję (przerw. 0ah),
— ustawia zakres ruchu kursora na cały ekran (przerw. 07h i 08h),
— ustawia stronę rysowania na zerową (przerw. 1dh),
— zezwala emulację pióra świetlnego (przerw. 0dh), oraz
— ustawia szybkość ruchu kursora na 8:8 poziomo i 8:16 pionowo (przerw. 0fh).

```
#include <dos.h>
#include "mouse.h"
void mouse_reset(struct MOUSEE *my){
     union REGS inr, outr;
     inr.x.ax = MOUSE_RESET;
     int86(MOUSE, &inr, &outr);
     my->status = outr.x.ax;
     my->button = outr.x.bx;
}
```

Wynik procedury sprawdzania zainstalowania myszy zawiera komórka my.status struktury MOUSEE my.

8.2. Funkcja uwidaczniająca kursor myszy

Ta funkcja powoduje zapoczątkowanie wyświetlania kursora myszy po uprzednim wywołaniu funkcji mouse_reset() zakończonym sukcesem:

```
#include <dos.h>
#include "mouse.h"
void mouse_show_cursor(void){
     union REGS inr;
     inr.x.ax = MOUSE_SHOW_CURSOR;
     int86(MOUSE, &inr, &inr);
}
```

Ze względu na to, że powyższe przerwanie nie zwraca wartości, nie jest wymagane deklarowanie unii REGS outr. Uwaga ta dotyczy także pozostałych przerwań mających tę samą cechę.

8.3. Funkcja ukrywająca kursor

Ta funkcja powinna być wywoływana po uprzednim wywołaniu funkcji "show_cursor()":

```
#include <dos.h>
#include "mouse.h"
void mouse_hide_cursor(void){
     union REGS inr;
     inr.x.ax = MOUSE_HIDE_CURSOR;
     int86(MOUSE, &inr, &inr);
}
```

8.4. Funkcja odczytująca pozycję i numer klawisza myszy

Ta funkcja zwraca trzy wartości, które są przekazywane programowi poprzez strukturę MOUSEE my:

```
#include <dos.h>
#include "mouse.h"
void mouse_get_but_pos(struct MOUSEE *my){
     union REGS inr, outr;
     inr.x.ax = MOUSE_GET_POS_BUTTON;
     int86(MOUSE, &inr, &outr);
     my->button = outr.x.bx;
     my->column = outr.x.cx;
     my->row    = outr.x.dx;
}
```

8.5. Funkcja pozycjonująca kursor myszy

Przy wywołaniu tej funkcji należy uwzględniać aktualny tryb pracy monitora. W trybie graficznym wartości column i row należy mnożyć przez 8:

```
#include <dos.h>
#include "mouse.h"
void mouse_put_pos_cursor(int column, int row){
     union REGS inr;
```

```
    inr.x.ax = MOUSE_PUT_POS_CURSOR;
    inr.x.cx = column;
    inr.x.dx = row;
    int86(MOUSE, &inr, &inr);
}
```

8.6. Funkcja odczytująca liczbę wciśnięć klawisza i pozycję kursora, od ostatniego wywołania tej funkcji

Argument funkcji — button — wskazuje klawisz myszy poddany kontroli zliczeń. Wybór ten nie blokuje odczytu wartości my.button wskazującej dowolny inny klawisz wciśnięty w momencie wywołania funkcji, ale zliczenia zachodzą tylko dla klawisza zadeklarowanego argumentem funkcji:

```
#include <dos.h>
#include "mouse.h"
void mouse_get_data_but_pressed(int button, struct MOUSEE *my){
    union REGS inr, outr;
    inr.x.ax = MOUSE_GET_DATA_BUT_PRESSED;
    inr.x.bx = button;
    int86(MOUSE, &inr, &outr);
    my->button       = outr.x.ax;
    my->nr_pressed   = outr.x.bx;
    my->column       = outr.x.cx;
    my->row          = outr.x.dx;
}
```

8.7. Funkcja odczytująca liczbę zwolnień klawisza i pozycję kursora, od ostatniego wywołania tej funkcji

Działanie tej funkcji jest analogiczne do omówionej w p. 8.6.

```
#include <dos.h>
#include "mouse.h"
void mouse_get_data_but_released(int button, struct MOUSEE *my){
    union REGS inr, outr;
    inr.x.ax = MOUSE_GET_DATA_BUT_RELEASED;
    inr.x.bx = button;
    int86(MOUSE, &inr, &outr);
```

```
        my->button        = outr.x.ax;
        my->nr_released   = outr.x.bx;
        my->column        = outr.x.cx;
        my->row           = outr.x.dx;
    }
```

8.8. Funkcja definiująca poziomy zakres ruchu kursora myszy

W modzie tekstowym argumenty min_column oraz max_column należy dzielić przez 8.

```
#include <dos.h>
#include "mouse.h"
void mouse_set_hcr(int min_column, int max_column){
     union REGS inr;
     inr.x.ax = MOUSE_SET_HCR;
     inr.x.cx = min_column;
     inr.x.dx = max_column;
     int86(MOUSE, &inr, &inr);
}
```

8.9. Funkcja definiująca pionowy zakres ruchu kursora myszy

W modzie tekstowym argumenty min_row oraz max_row należy dzielić przez 8.

```
#include <dos.h>
#include "mouse.h"
void mouse_set_vcr(int min_row, int max_row){
     union REGS inr;
     inr.x.ax = MOUSE_SET_VCR;
     inr.x.cx = min_row;
     inr.x.dx = max_row;
     int86(MOUSE, &inr, &inr);
  }
```

8.10. Funkcja definiująca kształt kursora graficznego myszy

Ta funkcja jest przykładem zastosowania procedury int86x(), która wykorzystuje zarówno unię REGS, jak i strukturę SREGS. Argumenty hot_column i hot_row wskazują punkt odniesienia, w stosunku do aktualnych wartości współrzędnych położenia kursora. Natomiast 64-bajtowa tablica "cursor[32]" definiuje kształt kursora oraz sposób jego wizualizacji na tle już istniejącego obrazu. Adres tej tablicy jest przekazywany w formacie intelowskim, tj. ES:DS (segment i offset).

```
#include <dos.h>
#include "mouse.h"
void mouse_set_graph_cursor(const int cursor[32], int hot_column,
                            int hot_row){
     union REGS inr;
     struct SREGS sinr;
     void far *ps;
     ps = cursor;
     inr.x.ax = MOUSE_SET_GRAPH_CURSOR;
     inr.x.bx = hot_column;
     inr.x.cx = hot_row;
     inr.x.dx = FP_OFF(ps);
     sinr.es  = FP_SEG(ps);
     int86x(MOUSE, &inr, &inr, &sinr);
}
```

9. TESTOWANIE FUNKCJI

Na koniec, w celu przetestowania zdefiniowanych wyżej funkcji można wykonać program, którego postać źródłowa, napisana z wykorzystaniem kompilatora TURBO C, jest następująca:

```
#include "mouse.h"
#include <stdio.h>
#include <graphics.h>
#include <conio.h>
#include <dos.h>
static  int cursor[32] = {
          0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
          0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
          0xf00f,0xf81f,0xfc3f,0xfe7f,0xffff,0xffff,0xf7ef,0xf3cf,
          0xf18f,0xf00f,0xf00f,0xf00f,0xf00f,0xf00f,0xf00f,0xf00f
```

```
};
int main(void){
     struct MOUSEE my;
     int i=0;
     int graphdriver, graphmode;
     int old_button=0, old_row=0, old_column=0, old_nr_pressed=0;

     clrscr();

     printf("URUCHOMIENIE KURSORA");
     mouse_reset(&my);
     while(i++<5){
       delay(300);
       mouse_show_cursor();
       delay(300);
       mouse_hide_cursor();
     }
     mouse_show_cursor();
     clrscr();
     printf("POBRANIE NUMERU PRZYCISKU MYSZY I POZYCJI WSKAZNIKA"
            "MYSZY");
     printf("Esc - nr3nr przycisku     kolumna       wiersz");
     do{
        mouse_get_but_pos(&my);
        if(my.button!=old_button||my.column!=old_column||
           my.row!=old_row){
          gotoxy(5,5);
          printf("%3d", my.button);
          gotoxy(20,5);
          printf("%3d", my.column);
          gotoxy(35,5);
          printf("%3d", my.row);
          gotoxy(5,5);
          old_button = my.button;
          old_column = my.column;
          old_row    = my.row;
        }
```

```
}while(my.button!=3);
clrscr();

printf("POBRANIE LICZBY WCISNIEC PRZYCISKU MYSZY OD"
       "OSTATNIEGO WYWOLANIA\n");
printf("Esc - nr3, przykladowy przycisk nr 2");
printf("nr przycisku   # przycisniec    kolumna     wiersz");
do{
   gotoxy(52,6);
   printf("%c",0xfe);
   gotoxy(1,6);
   for(i=0;i<51;i++){
     putch('.');
     delay(100);
   }
   gotoxy(1,6);
   clreol();
   mouse_get_data_but_pressed(1, & my);
   if(my.button!=old_button||old_nr_pressed!=my.nr_pressed||
      my.column!=old_column||my.row!=old_row){
      gotoxy(5,5);
      printf("%3d", my.button);
      gotoxy(20,5);
      printf("%3d", my.nr_pressed);
      gotoxy(35,5);
      printf("%3d", my.column);
      gotoxy(50,5);
      printf("%3d", my.row);
      gotoxy(5,5);
      old_button = my.button;
      old_nr_pressed = my.nr_pressed;
      old_column = my.column;
      old_row = my.row;
   }
}while(my.button!=3);

detectgraph(&graphdriver, &graphmode);
initgraph(&graphdriver, &graphmode,"");
```

```
mouse_reset(&my);
mouse_show_cursor();
outtextxy(150,0,"POZYCJONOWANIE KURSORA");
for(i=0;i<300;i+=50){
  mouse_put_pos_cursor(i,i);
  delay(300);
}
cleardevice();
getch();
outtextxy(150,0,"OGRANICZENIE ZAKRESU RUCHU KURSORA");
mouse_set_hcr(200,300);
mouse_set_vcr(100,200);
getch();
cleardevice();
outtextxy(150,0,"DEFINICJA KURSORA GRAFICZNEGO");
mouse_set_graph_cursor(&cursor[0], 0,0);
getch();
closegraph();
return (0);
```

W programie tym postać funkcji dotyczących działania myszy jest zgodna z przedstawionymi w p. 8. Pozostałe funkcje myszy można zrealizować w sposób analogiczny do zaprezentowanego powyżej.

10. UWAGI KOŃCOWE

Funkcje bazujące na przerwaniach, opisane w p. 8.1—8.10, zostały opracowane przy założeniu, że wartości zwracane przez przerwania są dostarczane programowi poprzez strukturę MOUSEE my. Zostało to przyjęte dla wygody i przejrzystości, co jest istotne zwłaszcza przy dużych programach. Alternatywą jest każdorazowe podawanie adresów wymaganych argumentów, co znacznie jednak utrudnia poruszanie się po bardziej skomplikowanym programie.