

WT/NP/17.05

Jakub Pomykała 209897

Ocena:

Oddano:

## Proste jądro systemu operacyjnego

ARCHITEKTURA KOMPUTERÓW 2 – PROJEKT  
INF 2014/15

PROWADZĄCY:

DR INŻ. TADEUSZ TOMCZAK

# Spis treści

<b>1</b>	<b>Wprowadzenie</b>	<b>3</b>
1.1	Plan projektu i osiągnięcia . . . . .	3
1.2	Podstawowe pojęcia . . . . .	3
1.3	Środowisko pracy i narzędzia . . . . .	5
<b>2</b>	<b>Praca jądra systemu w trybie chronionym</b>	<b>5</b>
2.1	Przełączanie procesora w tryb chroniony . . . . .	5
2.2	Obsługa przerwania i wyjątków za pomocą kontrolera przerwania . . . . .	5
2.3	Oprogramowanie kontrolera przerwania . . . . .	6
2.4	Obsługa przerwania pochodzącego z czasomierza systemowego . . . . .	6
2.5	Przełączenie zadań z wykorzystaniem przerwania czasomierza systemowego . . . . .	6
<b>3</b>	<b>Zakończenie</b>	<b>6</b>
3.1	Wnioski i możliwości dalszego rozwoju jądra . . . . .	6
<b>4</b>	<b>Listing kodów źródłowych</b>	<b>7</b>
4.1	<i>DEFSTR.TXT</i> - struktura deskryptorów . . . . .	7
4.2	<i>MAIN.ASM</i> - kod główny . . . . .	7
4.3	<i>PODST.TXT</i> - podstawowe funkcje jądra . . . . .	11
4.4	<i>OBSLPUL.TXT</i> - kod obsługi pułapek . . . . .	19
4.5	<i>TXTPUL.TXT</i> - tekst i atrybuty użyte w kodzie obsługi pułapek . . . . .	23
4.6	<i>RODZPUL.TXT</i> - lista pułapek . . . . .	23
<b>5</b>	<b>Bibliografia</b>	<b>24</b>

# 1 Wprowadzenie

## 1.1 Plan projektu i osiągnięcia

Projekt polegał na napisaniu prostego jądra systemu operacyjnego, przejścia w tryb chroniony i przełączaniu zadań za pomocą przerwań wywoływanych poprzez zegar systemowy. Kod źródłowy jądra został napisany w Turbo Assemblerze i uruchamiany jest w DOSBoxie 0.74. Początkowy plan zakładał napisanie jądra, bootloadera i uruchamianie jądra na komputerze PC z procesorem Intel Pentium z dyskietki. Niestety nie udało mi się skończyć pisać bootloadera, dlatego jądro uruchamiane jest w emulatorze DOSBox. Plan prac wyglądał następująco:

- przygotowanie środowiska pracy oraz narzędzi
- przełączenie procesora w tryb chroniony
- obsługa pamięci rozszerzonej
- obsługa przerwań i wyjątków
- przełączanie zadań przez przerwanie czasomierza systemowego

## 1.2 Podstawowe pojęcia

1. **tryb rzeczywisty** - jest to tryb pracy mikroprocesorów z rodziny procesorów x86, w którym procesor pracuje jak Intel 8086. Tryb ten nie zapewnia ochrony pamięci przed użyciem jej przez inny proces oraz obsługi wielozadaniowości. Dostępna jest jedynie 1-megabajtowa przestrzeń adresowa
2. **tryb chroniony procesora** - tryb pracy procesora, który umożliwia adresowanie pamięci przekraczającej 1-megabajt pamięci, sprzętowa ochrona pamięci, wsparcie w przełączeniu kontekstu procesora, stronicowanie pamięci (32 bitowe procesory)
3. **deskryptor** - 64-bitowa struktura danych w której przechowywane są informacje na temat miejsca w pamięci danego segmentu, typu, rozmiaru, zasady dostępu do segmentu oraz pozostałe informacje przydatne przy dostępie do segmentu w trybie chronionym procesora.
4. **tablice deskryptorów** - w trybie chronionym posługujemy się tablicami deskryptorów, wyróżniamy trzy podstawowe struktury:
  - Global Descriptor Table (GDT) - globalna tablica, zawiera deskryptory, które mogą być wykorzystane przez dowolne zadanie w systemie. Przechowują pamięć ekranu oraz ogólnie dostępne segmenty kodu i danych
  - Local Descriptor Table (LDT) - lokalna tablica, zawiera deskryptory dostępne tylko dla konkretnego zadania
  - Interrupt Descriptor Table (IDT) - tablica deskryptorów przerwań, używana do poprawnego reagowania na przerwanie oraz wyjątki
5. **rejestry segmentowe** - zawierają adresy bazowe tablic systemowych, służą do organizacji segmentacji w trybie chronionym

- Global Descriptor Table Registers (GDTR) - liniowy adres bazowy i rozmiar globalnej tablicy deskryptorów
- Interrupt Descriptor Table Registers (IDTR) - liniowy adres bazowy i rozmiar tablicy deskryptorów przerwań
- Local Descriptor Table Registers (LDTR) - selektor segmentu tablicy deskryptorów lokalnych
- Task Registers (TR) - rejestr stanu zadania, selektor stanu zadania

6. **selektor** - w trybie chronionym procesora selektory są umieszczone w rejestrach segmentowych. Format selektora prezentuje się następująco:

- INDEX - indeksu w tablicy deskryptorów, bity numer 15 - 3
- TI - wyróżnika tablicy, czy tablica jest globalna (0) czy lokalna (1), bit numer 2
- RPL - poziomu uprzywilejowania, bity numer 1 - 0

7. **segmentacja pamięci w trybie chronionym** - każdy segment danych bądź stosu jest opisany parametrami:

- lokalizacja w przestrzeni adresowej pamięci
- zasady dostępu
- 8 bajtowa struktura danych nazywana deskryptorem

Tablice mogą zawierać od 8 bajtów do 64kB (8192 deskryptory)

Odwołanie do odpowiedniego deskryptora wykonuje się za pomocą selektora zapisanego w jednym z 16 bitowych rejestrów segmentowych:

- rejestr DS, ES, FS, GS - segment musi mieć zezwolenie tylko do odczytu
- rejestr SS - musi mieć ustawione prawa zapisu oraz odczytu
- rejestr CS - wymaga prawa kodu wykonywalnego

FS oraz GS są dostępne tylko w trybie chronionym. W przypadku wpisania błędnego selektora do rejestru segmentowego otrzymamy błąd "Ogólnego naruszenia ochrony".

8. **przerwanie** - jest to sygnał, który powoduje zmianę przepływu sterowania, niezależnie od aktualnie wykonywanego programu. W przypadku pojawienia się przerwania wstrzymywany jest aktualnie wykonywane zadanie i następuje skok do innego miejsca w kodzie, np. procedury. Procedura ta wykonuje czynności związane z obsługą przerwania i na końcu wydaje instrukcję powrotu z przerwania, która powoduje powrót do programu realizowanego przed przerwaniem. Rozróżniamy kilka typów przerwania:

- programowe - wywoływane przez programistę, instrukcją INT + kod przerwania, lub w przypadku operacji niedozwolonych, np. dzielenie przez zero
- sprzętowe - generowane przez urządzenia zewnętrzne, np. obsługa klawiatury, czyli wciśnięcie jakiegoś klawisza, może to też być drukarka, myszka, dysk twardy itp.
- wyjątki - generowane przez zewnętrzne układy procesora

9. **kontroler przerw** - układ obsługi przerw w komputerach PC jest zbudowany z dwóch połączonych kaskadowo układów 8259A, dzięki temu możliwa jest obsługa 15 przerw sprzętowych - wejście IRQ2 układu master jest połączone z wyjściem układu slave. Kontroler klawiatury znajduje się na linii IRQ1, a czasomierz systemowy na linii IRQ0
10. **czasomierz systemowy (lub zegar systemowy)** - jest to fizyczne urządzenie znajdujące się na płycie głównej komputera, odpowiedzialne za dostarczanie aktualnego czasu i daty do komputera. Odpowiada również za dostarczanie sygnałów synchronizujących działanie podzespołów komputera z dokładnością do tysięcznych części sekundy.
11. **czasomierz systemowy (lub zegar systemowy)** - jest to fizyczne urządzenie znajdujące się na płycie głównej komputera, odpowiedzialne za dostarczanie aktualnego czasu i daty do komputera. Odpowiada również za dostarczanie sygnałów synchronizujących działanie podzespołów komputera z dokładnością do tysięcznych części sekundy.
12. **zadanie (ang. task)** - rozumiemy jako wykonywany program lub niezależny jego fragment
13. **Task State Segment (TSS)** - segment stanu zadania jest rekordem wchodzącym w skład segmentu danych lub oddzielnym segmentem o niewielkim rozmiarze. Każde zadanie ma swój segment stanu zadania. Segmentowi TSS odpowiada systemowy deskryptor tego segmentu, przechowywany w globalnej tablicy deskryptorów. Struktura jest analogiczna do deskryptora pamięci, jedyna różnica polega na różnych kodach typów segmentów.

## 1.3 Środowisko pracy i narzędzia

Jądro systemu było testowane za pomocą programu DOSBox 0.74 na komputerze z systemem Windows 8.1 x64. Program DOSBox 0.74 jest pełnym emulatorem procesora Intel 80386 udostępnianym na licencji GNU GPL. Kod jądra był asemblerowany za pomocą TASM.exe (Turbo Assembler) oraz linkowany za pomocą TLINK.exe (Turbo Linker).

## 2 Praca jądra systemu w trybie chronionym

### 2.1 Przełączanie procesora w tryb chroniony

Procesor na początku swojego działania znajduje się w trybie rzeczywistym, żeby przełączyć go w tryb chroniony musimy zdefiniować strukturę globalnej tablicy deskryptorów (GDT), następnie w rejestrze CR0 ustawić pierwszy bit (tzw. bit PE - Protection Enable) na 1. Można to zrobić za pomocą instrukcji SMSW lub MOV. Od tej pory nasz procesor pracuje w trybie chronionym. Żeby powrócić do trybu rzeczywistego wystarczy, że wyzerujemy bit PE w rejestrze CR0.

### 2.2 Obsługa przerw i wyjątków za pomocą kontrolera przerw

W wprowadzenia obsługi przerw musimy:

- utworzyć tablicę deskryptorów przerw IDTR
- umieścić w niej adresy procedur obsługi wyjątków, które będą w programie

- załadować adres tablicy IDT do rejestru IDTR za pomocą instrukcji LIDTR
- odpowiednie do potrzeb skonfigurowanie kontrolera przerwań

## 2.3 Oprogramowanie kontrolera przerwań

Zaprogramowanie pracy kontrolera przerwań polega na zamaskowaniu nieobsługiwanych programowo przerwań sprzętowych (np. myszka czy dysk twardy), zależy nam jedynie na obsłudze zegara systemowego. W pliku PODST.TXT makropolecenie które przyjmuje jako parametr maskę przerwań układu. Wartość 1 na danej pozycji oznacza zablokowanie przerwań na tej linii. Czasomierz systemowy, który posłuży nam do wywoływania przerwań systemowych znajduje się na linii IRQ0. W takim razie użyjemy maski FEh, która binarnie wynosi 1111 1110. Co oznacza że jedynymi przerwaniami sprzętowymi jakie będziemy otrzymywać będą przerwania z czasomierza systemowego.

## 2.4 Obsługa przerwania pochodzącego z czasomierza systemowego

W momencie poprawnej konfiguracji kontrolera przerwań ostatnim krokiem do obsługi przerwań jest odblokowanie ich otrzymywania za pomocą instrukcji STI. W tym momencie z każdym przerwaniami czasomierza program będzie przenosić się do linii 55 w pliku MAIN.ASM, gdzie następuje obsługa przerwania. Obsługa przerwania to przełączenie zadania na jedno z dwóch za pomocą instrukcji JMP, czyli skoku odległego. W momencie skoku do odpowiedniego zadania, wywoływane jest przerwanie programowe, (instrukcja INT) którego obsługa polega na wyświetleniu informacji o aktywnym zadaniu. Następnie wykonywane jest makro OPOZNIENIE przy pomocy dwóch zagnieżdżonych pętli i wykorzystaniu makra z pliku PODST.TXT (linie 312 - 326). Sygnał zakończenia przerwania, czyli informacja dla kontrolera przerwań o zakończeniu obsługi przerwania poprzez zapis wartości 20H na port 20H. Skok na początek aktualnie wykonywanego zadania. W momencie przyjscia kolejnego przerwania jądro znów znajdzie się na linii 56 i całą procedurę rozpocznie się od nowa.

## 2.5 Przełączenie zadań z wykorzystaniem przerwań czasomierza systemowego

Podczas przełączania zadania procesor zapamiętuje kontekst bieżącego zadania w jego segmencie stanu zadania, a następnie odczytuje z TSS kontekst nowego zadania, zawierający selektor segmentu i offset, od którego należy rozpocząć jego realizację. Następnie wykonywany jest rozkaz skoku odległego

# 3 Zakończenie

## 3.1 Wnioski i możliwości dalszego rozwoju jądra

Realizacja projektu pozwoliła mi na dokładniejsze poznanie procesorów jakie zachodzą we współczesnych systemach operacyjnych. Dzięki praktyce lepiej poznałem teorię architektury komputerów, mogłem dowiedzieć się jak działa jeden z najważniejszych elementów komputera, czyli procesor. Dzięki podziałowi projektu na kilka plików tekstowych z kodem źródłowym, projekt jest bardzo skalowalny. Z łatwością można dodać do niego obsługę klawiatury, czy innych urządzeń zewnętrznych.

## 4 Listing kodów źródłowych

### 4.1 *DEFSTR.TXT* - struktura deskryptorów

```
1 ;struktura opisujaca deskryptor segmentu
2 DESKR STRUC
3   LIMIT DW 0 ;rozmiar
4   BASE_1 DW 0 ;
5   BASE_M DB 0
6   ATTR_1 DB 0 ;atrybuty
7   ATTR_2 DB 0 ;atrybuty
8   BASE_H DB 0
9 DESKR ENDS
10
11 ;Struktura opisujaca furtke pulapki:
12 TRAP STRUC
13   OFFS_1 DW 0 ;Offset procedury obslugi (bity 0...15).
14   SEL DW 16 ;Selektor segmentu programu.
15   RSRV DB 0 ;Bajt zarezerwowany.
16   ATTR DB 8FH ;Obecnosc + furtka pulapki.
17   OFFS_H DW 0 ;Offset procedury obslugi (bity 16-31).
18 TRAP ENDS
19
20 ;Struktura opisujaca furtke przerwania:
21 INTR STRUC
22   IOFFS_1 DW 0
23   ISEL DW 16
24   IRSRV DB 0
25   IATTR DB 8EH
26   IOFFS_H DW 0
27 INTR ENDS
```

### 4.2 *MAIN.ASM* - kod główny

```
1 .386P
2 DANE SEGMENT USE16
3   GDT_NULL DESKR <0,0,0,0,0,0> ;segment 0
4   GDT_DANE DESKR <DANE_SIZE-1,0,0,92H,0,0> ;segment 8
5   GDT_PROGRAM DESKR <PROGRAM_SIZE-1,0,0,98H,0,0> ;segment 16
6   GDT_STOS DESKR <513,0,0,92H,0,0> ;segment 24
7   GDT_EKRAN DESKR <4095,8000H,0BH,92H,0,0> ;segment 32
8   GDT_TSS_0 DESKR <103,0,0,89H,0,0> ;segment 40
9   GDT_TSS_1 DESKR <103,0,0,89H,0,0> ;segment 48
10  GDT_TSS_2 DESKR <103,0,0,89H,0,0> ;segment 56
11  GDT_SIZE = $ - GDT_NULL
12
13 ;Tablica deskryptorow przerwan IDT
14 IDT LABEL WORD
```

```

15      INCLUDE          RODZPUL.TXT
16      IDT_0            INTR <PROC_0>
17      IDT_SIZE = $ - IDT
18      PDESKR           DQ      0
19      ORG_IDT           DQ      0
20
21      WELCOME           DB      'Architektura_komputerow_-_Jakub_Pomykala_2098
22  INFO                 DB      'POWROT_Z_TRYBU_CHRONIONEGO_$'
23
24      INCLUDE          TXTPUL.TXT
25
26      T0_ADDR           DW 0,40   ;adresy zadan wg segmentow powyzej
27      T1_ADDR           DW 0,48
28      T2_ADDR           DW 0,56
29
30      TSS_0             DB 104 DUP (0)
31      TSS_1             DB 104 DUP (0)
32      TSS_2             DB 104 DUP (0)
33
34      ZADANIE_1         DB      '1'
35      ZADANIE_2         DB      '2'
36      PUSTE             DB      '_'
37      AKTYWNE_ZADANIE   DW 0
38      CZAS              DW 0
39
40      POZYCJA_1         DW 320
41      POZYCJA_2         DW 2560
42      POZYCJA           DW 0
43
44  DANE_SIZE= $ - GDT_NULL
45  DANE ENDS
46
47  PROGRAM SEGMENT 'CODE' USE16
48      ASSUME CS:PROGRAM, DS:DANE, SS:STK ;informacja dla TASMa jakie segm
49  POCZ LABEL WORD
50
51  INCLUDE OBSLPUL.TXT
52  INCLUDE PODST.TXT
53  PROC_0 PROC
54
55      PUSH  AX
56      PUSH  DX
57
58      CMP    AKTYWNE_ZADANIE,1      ;czy AKTYWNE_ZADANIE == 1?
59      JE     ETYKIETA_ZADANIE_1    ;jesli tak to skaczemy do ETYKIETA_ZADA
60
61      CMP    AKTYWNE_ZADANIE,0      ;czy AKTYWNE_ZADANIE == 0?
62      JE     ETYKIETA_ZADANIE_2    ;jesli tak to skaczemy do ETYKIETA_ZADA

```



```

63         JMP     DALEJ
64
65     ETYKIETA_ZADANIE_1:
66     MOV AKTYWNE_ZADANIE, 0
67     JMP DWORD PTR T0_ADDR           ;przelaczenie zadania na zadanie nr 1
68         JMP DALEJ
69
70     ETYKIETA_ZADANIE_2:
71     MOV AKTYWNE_ZADANIE, 1
72     JMP DWORD PTR T2_ADDR           ;przelaczenie zadania na zadanie nr 2
73
74     DALEJ:
75     POP     DX
76         POP     AX
77         IRETD
78
79 PROC_0   ENDP
80
81 START:
82         WPISZ_DESKRYPTORY
83
84     PM_TASKS TSS_0,TSS_1,GDT_TSS_0,GDT_TSS_1
85         XOR     EAX, EAX
86         MOV     AX, OFFSET TSS_2
87         ADD     EAX, EBP
88         MOV     BX, OFFSET GDT_TSS_2
89         MOV     [BX].BASE_1, AX
90         ROL     EAX, 16
91         MOV     [BX].BASE_M, AL
92
93         ;zadanie 1 ze stosem 256
94         MOV WORD PTR TSS_1+4CH, 16           ;CS (SEGMENT PROGRAMU)
95         MOV WORD PTR TSS_1+20H, OFFSET ZADANIE1 ;IP (SEGMENT adresu powrotu)
96         MOV WORD PTR TSS_1+50H, 24           ;SS (SEGMENT STOSU)
97         MOV WORD PTR TSS_1+38H, 256         ;SP (SEGMENT wielkosc stosu)
98         MOV WORD PTR TSS_1+54H, 8           ;DS (SEGMENT DANYCH)
99         MOV WORD PTR TSS_1+48H, 32          ;ES (SEGMENT EKRANU)
100
101         STI                                     ;ustawienie znacznika zestawienia
102         PUSHFD                                ;przeslanie znacznikow na szczyt stosu
103         POP EAX
104
105         MOV DWORD PTR TSS_1+24H, EAX          ;zapisujemy ee flags
106
107         ;zadanie 2 ze stosem 256
108         MOV WORD PTR TSS_2+4CH, 16           ;CS (SEGMENT PROGRAMU)
109         MOV WORD PTR TSS_2+20H, OFFSET ZADANIE2 ;IP (SEGMENT adresu powrotu)
110         MOV WORD PTR TSS_2+50H, 24           ;SS (SEGMENT STOSU)

```

```

111      MOV WORD PTR TSS_2+38H, 256           ;SP (SEGMENT wielkosc stosu
112      MOV WORD PTR TSS_2+54H, 8             ;DS (SEGMENT DANYCH)
113      MOV WORD PTR TSS_2+48H, 32            ;ES (SEGMENT EKRANU)
114
115      MOV DWORD PTR TSS_2+24H, EAX
116
117      CLI                                     ;blokujemy przerwania
118      WPISZ_IDTR                             ;zapisujemy tablice deskryp
119      KONTROLER_PRZERWAN 0FEH                ;konfigurujemy kontroler pr
120      TRYB_CHRONIONY                          ;przechodzimy w tryb chroni
121
122      MOV AX, 32
123      MOV ES, AX
124      MOV GS, AX
125      MOV FS, AX
126      MOV AX, 40                             ;Zaladowanie rejestru zadania (TR)
127      LTR AX                                 ;deskryptorem segmentu stanu
128
129      CZYSC_EKRAN
130      OPOZNIENIE 100
131      WYPISZ WELCOME,47,30,ATRYB
132
133      STI                                     ;zezwalamy na przerwania
134
135      ;zadanie ktore wypisuje jedyнки na ekranie
136      ZADANIE1 PROC
137      ZADANIE_1_PETLA:
138          MOV AL, ZADANIE_1
139          MOV BX, POZYCJA_1
140          MOV AH, 02h
141          MOV ES:[BX], AX
142
143          INT 2                               ;wywołanie przerwania z informacja o ak
144          OPOZNIENIE 200
145
146          ADD POZYCJA_1, 2
147
148          MOV AL, 20H                         ;sygnal konca obsługi przerwania
149          OUT 20H, AL
150
151          JMP ZADANIE_1_PETLA
152      ZADANIE1 ENDP
153
154      ;zadanie ktore wypisuje dwójki na ekranie
155      ZADANIE2 PROC
156      ZADANIE_2_PETLA:
157          MOV AL, ZADANIE_2
158          MOV BX, POZYCJA_2

```

```

159      MOV AH, 02h
160      MOV ES:[BX], AX
161
162      INT 3                                ;wywołanie przerwania z informacja o ak
163      OPOZNIENIE 300
164
165      ADD POZYCJA_2, 2
166
167      MOV AL, 20H                          ;sygnał końca obsługi przerwania
168      OUT 20H, AL
169
170      JMP ZADANIE_2_PETLA
171 ZADANIE2 ENDP
172
173 PROGRAM_SIZE= $ - POCZ
174 PROGRAM ENDS
175 STK      SEGMENT STACK 'STACK'
176          DB 256*3 DUP(0)
177 STK      ENDS
178 END START

```

### 4.3 *PODST.TXT* - podstawowe funkcje jądra

```

1  CZY_DOSTEPNY_FAST_A20 MACRO
2  LOCAL BRAK, KONIEC_A20
3      MOV AX, DANE
4      MOV DS, AX
5      MOV AX, 2403H
6      INT 15H
7      JC BRAK_A20
8      CMP AH, 0
9      JNE BRAK_A20
10     TEST BX, 2
11     JZ BRAK_A20
12     MOV FAST_A20, 1
13     JMP KONIEC_A20
14
15     BRAK_A20:
16     MOV FAST_A20, 0
17     KONIEC_A20:
18 ENDM
19
20 CZY_A20 MACRO
21     PUSH AX
22     PUSH BX
23     MOV AL, [0:0]
24     MOV BL, AL
25     NOT BL

```

```

26  XCHG BL, [0FFFFH:10H]
27  CMP AL, [0:0]
28  MOV [0FFFFH:10H], BL
29  POP BX
30  POP AX
31  ENDM
32
33  A20_ON MACRO
34  LOCAL KONIEC_A20,BRAK_FAST_A20, PETLA1_A20, PETLA2_A20, PETLA3_A20, PETLA4_A20
35  ; Czy A20 juz aktywne:
36  MOV A20,0
37  CZY_A20
38  JE KONIEC_A20
39  CMP FAST_A20, 1
40  JNE BRAK_FAST_A20
41  ; Fast a20:
42  MOV A20, 1
43  IN AL, 92H
44  OR AL, 2
45  AND AL, 0FEH
46  OUT 92H, AL
47  CZY_A20
48  JE KONIEC_A20
49
50  BRAK_FAST_A20:
51  ; Uaktywnienie A20 poprzez sterownik klawiatury:
52  ; Oczekiwanie na pusty bufor wejscowy:
53  XOR AX, AX
54  PETLA1_A20:
55  IN AL, 64H
56  BTR AX, 1
57  JC PETLA1_A20
58
59  ; Wyslanie komendy odczytu portu wyjsciowego:
60  MOV AL, 0D0H ; Rozkaz odczytu portu wyjsciowego.
61  OUT 64H, AL
62  XOR AX, AX
63  PETLA2_A20:
64  IN AL, 64H
65  BTR AX, 0 ; Stan bufora wyjsciowego
66  ; (0 pusty, 1 dane sa jeszcze w buforze).
67  JNC PETLA2_A20
68
69  ; Odczyt stanu portu wyjsciowego:
70  XOR AX, AX
71  IN AL, 60H
72  PUSH AX
73  PETLA3_A20:

```

```

74  IN AL, 64H
75  BTR AX, 1                ;Oczekiwanie na pusty bufor wejsciowy.
76  JC PETLA3_A20
77
78  ;Komenda zapisu do portu wyjsciowego:
79  MOV AL, 0D1H            ;Rozkaz zapisu portu wyjsciowego.
80  OUT 64H, AL
81  PETLA4_A20:
82  XOR AX, AX
83  IN AL, 64H
84  BTR AX, 1                ;Oczekiwanie na pusty bufor wejsciowy.
85  JC PETLA4_A20
86
87  ;Zapis portu wyjsciowego:
88  POP AX
89  OR AL, 10B
90  OUT 60H, AL
91  MOV A20, 2
92  CZY_A20
93  JE KONIEC_A20
94
95  ; nieskonczona petla
96  JMP $
97  KONIEC_A20:
98  ENDM
99
100 A20_OFF MACRO
101 LOCAL KONIEC_A20, PETLA1_A20, PETLA2_A20, PETLA3_A20, PETLA4_A20, WYLACZ_FA
102  CMP A20, 0
103  JE KONIEC_A20
104  CMP A20, 1
105  JE WYLACZ_FAST
106
107  ;Dezaktywacja A20 poprzez sterownik klawiatury:
108  ;Oczekiwanie na pusty bufor wejsciowy:
109  XOR AX, AX
110  PETLA1_A20:
111  IN AL, 64H
112  BTR AX, 1
113  JC PETLA1_A20
114
115  ;Wyslanie komendy odczytu statusu
116  MOV AL, 0D0H            ;Rozkaz odczytu portu wyjsciowego.
117  OUT 64H, AL
118  XOR AX, AX
119  PETLA2_A20:
120  IN AL, 64H
121  BTR AX, 0                ;Stan bufora wyjsciowego

```

```

122                                     ;(0 pusty, 1 dane sa jeszcze w buforze).
123     JNC PETLA2_A20
124
125 ; Odczyt stanu portu wyjsciowego:
126     XOR AX, AX
127     IN AL, 60H
128     PUSH AX
129     PETLA3_A20:
130     IN AL, 64H
131     BTR AX, 1                                     ;Oczekiwanie na pusty bufor wejsciowy.
132     JC PETLA3_A20
133
134 ;Komenda zapisu do portu wyjsciowego:
135     MOV AL, 0D1H                                     ;Rozkaz zapisu portu wyjsciowego.
136     OUT 64H, AL
137     PETLA4_A20:
138     XOR AX, AX
139     IN AL, 64H
140     BTR AX, 1                                     ;Oczekiwanie na pusty bufor wejsciowy.
141     JC PETLA4_A20
142
143 ;Zapis portu wyjsciowego:
144     POP AX
145     AND AL, 11111101B
146     OUT 60H, AL
147     MOV A20, 2
148     CZY_A20
149     JE KONIEC_A20
150 ; Wylaczenie A20 metoda fast A20:
151     WYLACZ_FAST:
152     IN AL, 92H
153     AND AL, 0FCH
154     OUT 92H, AL
155     KONIEC_A20:
156 ENDM
157
158 WPISZ_DESKRYPTORY MACRO
159     MOV AX,DANE
160     MOV DS,AX
161     MOV DL,0                                     ;20-bitowy adres bazowy segmentu danych.
162     SHLD DX,AX,4
163     SHL AX,4
164     MOV BX,OFFSET GDT_DANE                                     ;Wpisanie adresu bazowego
165                                     ;segmentu danych do odpowiednich
166     MOV [BX].BASE_1,AX                                     ;pol deskryptora GDT_DANE.
167     MOV [BX].BASE_M,DL
168     MOV AX,CS
169     MOV DL,0

```

```

170  SHLD DX,AX,4
171  SHL AX,4
172  MOV BX,OFFSET GDT_PROGRAM
173  MOV [BX].BASE_1,AX
174  MOV [BX].BASE_M,DL
175  MOV AX,SS
176  MOV DL,0
177  SHLD DX,AX,4
178  SHL AX,4
179  MOV BX,OFFSET GDT_STOS
180  MOV [BX].BASE_1,AX
181  MOV [BX].BASE_M,DL
182  MOV BX,OFFSET GDT_DANE
183      ;Przepisanie adresu bazowego oraz granicznego segmentu danych do
184      ;pseudoskryptora opisujacego globalna tablice deskryptorow.
185  MOV AX,[BX].BASE_1
186  MOV WORD PTR PDESKR+2,AX
187  MOV DL,[BX].BASE_M
188  MOV BYTE PTR PDESKR+4,DL
189  MOV WORD PTR PDESKR,GDT_SIZE-1
190  LGDT PDESKR                      ;Zaladowanie rejestru GDTR.
191  ENDM
192
193  TRYB_CHRONIONY MACRO
194      SMSW AX                      ;Przelaczenie procesora w tryb
195      OR AX,1                      ;pracy chronionej.
196      LMSW AX
197      DB 0EAH                      ;Skok odlegly do etykiety
198      DW OFFSET CONTINUE ;continue oraz segmentu
199      DW 10H                      ;okreslonego selektorem
200                                ;10h (segment programu).
201  CONTINUE:
202      MOV AX,08                      ;Zaladowanie selektora
203      MOV DS,AX                      ;segmentu danych.
204      MOV AX,18H                      ;Zaladowanie selektora
205      MOV SS,AX                      ;segmentu stosu.
206  ENDM
207
208  TRYB_RZECZYWISTY MACRO WYLACZYC_A20,PRZYWROCIC_IDTR
209      RETURN:
210      MOV AX,DANE      ;Procesor pracuje w trybie Real.
211      MOV DS,AX        ;Inicjalizacja rejestrow segmentowych.
212      MOV AX,STK
213      MOV SS,AX
214      IF PRZYWROCIC_IDTR EQ 1
215          LIDT ORG_IDT
216      ENDIF
217      IF WYLACZYC_A20 EQ 1

```

```

218     A20_OFF
219     ENDIF
220     STI                                ; Odblokowanie przerwan.
221     MOV AH,9                          ; Wydruk tekstu zapisane w zmiennej INFO.
222     MOV DX,OFFSET INFO
223     INT 21H
224     MOV AX,4C00H                      ; Koniec pracy programu.
225     INT 21H
226 ENDM
227
228 KONTROLER_PRZERWAN MACRO MASKA
229 ;PROGRAMOWANIE KONTROLERA PRZERWAN
230     MOV DX,20H                        ; Inicjacja pracy ukkladu
231     MOV AL,11H                        ; icw1=11h
232     OUT DX,AL
233     INC DX
234     MOV AL,20H                        ; icw2=20h (offset wektora przerwan)
235     OUT DX,AL
236     MOV AL,4                          ; icw3=04h (uklad master)
237     OUT DX,AL
238     MOV AL,1                          ; icw4=01h (tryb 8086/88)
239     OUT DX,AL
240     MOV AL, MASKA                    ; ocw1=0fdh (maska przerwan - master)
241     OUT DX,AL
242     MOV DX,0A1H                      ; Maska przerwan slave
243     MOV AL,0FFH
244     OUT DX,AL
245 ENDM
246
247 WPISZ_IDTR MACRO
248     MOV WORD PTR PDESKR,IDT_SIZE-1
249     XOR EAX,EAX
250     MOV AX,OFFSET IDT
251     ADD EAX,EBP
252     MOV DWORD PTR PDESKR+2,EAX
253     SIDT ORG_IDT
254     LIDT PDESKR
255 ENDM
256 ; Odpowiednik instrukcji PUSHA w chwili
257 ; gdy korzystanie ze stosu jest niewygodne:
258 PUSH_REG MACRO
259     MOV [SCHOWEK_REJESTROW] , EAX
260     MOV [SCHOWEK_REJESTROW+4] , EBX
261     MOV [SCHOWEK_REJESTROW+8] , ECX
262     MOV [SCHOWEK_REJESTROW+12] , EDX
263     MOV [SCHOWEK_REJESTROW+16] , ESI
264     MOV [SCHOWEK_REJESTROW+20] , EDI
265 ENDM

```



```

266 ;Przywraca rejestry zachowane makrem PUSH_REG:
267 POP_REG MACRO
268     MOV EAX, [SCHOWEK_REJESTROW]
269     MOV EBX, [SCHOWEK_REJESTROW+4]
270     MOV ECX, [SCHOWEK_REJESTROW+8]
271     MOV EDX, [SCHOWEK_REJESTROW+12]
272     MOV ESI, [SCHOWEK_REJESTROW+16]
273     MOV EDI, [SCHOWEK_REJESTROW+20]
274 ENDM
275
276 ;wypisanie na ekranie tekstu
277 WYPISZ MACRO NAZWA_ZM, WIELKOSC_ZM, OFFSET_EKRANU, ATRYBUT_ZN
278 LOCAL PETLA
279     MOV BX, OFFSET NAZWA_ZM
280     MOV CX, WIELKOSC_ZM
281     MOV AL, [BX]
282     MOV SI, 0
283     PETLA:
284         MOV ES:[SI+OFFSET_EKRANU], AL
285         MOV AL, ATRYBUT_ZN
286         MOV ES:[SI+OFFSET_EKRANU+1], AL
287         INC BX
288         INC SI
289         INC SI
290         MOV AL, [BX]
291     LOOP PETLA
292 ENDM
293
294
295 PM_TASKS MACRO S_TSS0, S_TSS1, G_TSS0, G_TSS1
296     MOV AX, SEG DANE
297     SHL EAX, 4
298     MOV EBP, EAX
299     XOR EAX, EAX
300     MOV AX, OFFSET S_TSS0
301     ADD EAX, EBP
302     MOV BX, OFFSET G_TSS0
303     MOV [BX].base_1, AX
304     ROL EAX, 16
305     MOV [BX].base_m, AL
306 ; TSS_1:
307     XOR EAX, EAX
308     MOV AX, OFFSET S_TSS1
309     ADD EAX, EBP
310     MOV BX, OFFSET G_TSS1
311     MOV [BX].base_1, AX
312     ROL EAX, 16
313     MOV [BX].base_m, AL

```

```

314 ENDM
315
316 ;dwie zagniezdzone petle w celu opoznienia dalszego wykonywania kodu
317 OPOZNIENIE MACRO ILE
318 LOCAL PRZEBIEG1
319 LOCAL PRZEBIEG2
320     MOV BX, ILE
321     PRZEBIEG2:
322         MOV AX, 0FFFFH
323     PRZEBIEG1:
324         SUB AX, 1
325         CMP AX, 0
326         JNZ PRZEBIEG1
327
328         SUB BX, 1
329         CMP BX, 0
330         JNZ PRZEBIEG2
331 ENDM
332
333 ;wyczyszczenie calego ekranu po przez wypisanie
334 ;na calym ekranie spacji/znakow bialych
335 CZYSC_EKRAN MACRO
336 LOCAL EKRN
337     EKRN:
338         MOV AL, PUSTE
339         MOV BX, POZYCJA
340         MOV ES: [BX], AX
341
342         ADD POZYCJA, 2
343
344         CMP POZYCJA, 6000
345         JNE EKRN
346 ENDM
347
348 ;zmiana koloru calego ekranu na podany w parametrze
349 KOLORUJ_EKRAN MACRO KOLEK
350 LOCAL KLR
351     KLR:
352         MOV AL, PUSTE
353         MOV BX, POZYCJA
354         MOV AH, 06h
355         MOV ES: [BX], AX
356
357         ADD POZYCJA, 1
358
359         CMP POZYCJA, 6000
360         JNE KLR
361 ENDM

```

#### 4.4 *OBSLPUL.TXT* - kod obsługi pułapek

```
1 exc_0    PROC ; Wyjatek nr 0
2    MOV AX,32
3    MOV ES,AX
4    MOV BX,OFFSET tekst_0
5    MOV CX,21
6    MOV AL,[BX]
7        MOV SI,0
8 petla0 :
9    MOV ES:[SI+160],AL
10        MOV AL,atryb_0
11        MOV ES:[SI+161],AL
12    ADD BX,1
13        ADD SI,2
14    MOV AL,[BX]
15    LOOP petla0
16        IRETD
17 exc_0    ENDP
18
19 exc_1    PROC ; Wyjatek nr 1
20        MOV AX,32
21    MOV ES,AX
22    MOV BX,OFFSET tekst_1 ; tekst 'obsługa przerwania 1'
23    MOV CX,26
24    MOV AL,[BX]
25        MOV SI,0
26 petla1 :
27    MOV ES:[SI+160],AL
28        MOV AL,atryb_1
29        MOV ES:[SI+161],AL
30    ADD BX,1
31        ADD SI,2
32    MOV AL,[BX]
33    LOOP petla1
34        IRETD
35 exc_1    ENDP
36
37 exc_2    PROC
38
39 MOV AX,32
40    MOV ES,AX
41    MOV BX,OFFSET tekst_2 ; aktywne zadanie numer 1 (23)
42    MOV CX,23
43    MOV AL,[BX]
44        MOV SI,0
45 petla2 :
46    MOV ES:[SI+160],AL
```

```

47         MOV AL, atryb_2
48         MOV ES:[SI+161],AL
49     ADD BX,1
50         ADD SI,2
51     MOV AL,[BX]
52     LOOP petla2
53         IRETD
54
55 exc_2    ENDP
56
57 exc_3    PROC
58 MOV AX,32
59     MOV ES,AX
60     MOV BX,OFFSET tekst_3 ;aktywne zadanie numer 2 (23)
61     MOV CX,23
62     MOV AL,[BX]
63         MOV SI,0
64 petla3:
65     MOV ES:[SI+160],AL
66         MOV AL, atryb_3
67         MOV ES:[SI+161],AL
68     ADD BX,1
69         ADD SI,2
70     MOV AL,[BX]
71     LOOP petla3
72         IRETD
73 exc_3    ENDP
74
75 exc_4    PROC
76 exc_4    ENDP
77
78 exc_5    PROC
79 exc_5    ENDP
80
81 exc_6    PROC
82 exc_6    ENDP
83
84 exc_7    PROC
85 exc_7    ENDP
86
87 exc_8    PROC
88 exc_8    ENDP
89
90 exc_9    PROC
91 exc_9    ENDP
92
93 exc_10   PROC ;Wyjatek nr 10
94     MOV AX,32

```

```

95     MOV ES,AX
96     MOV BX,OFFSET tekst_10
97     MOV CX,18
98     MOV AL,[BX]
99         MOV SI,0
100 petla10:
101     MOV ES:[SI+160],AL
102         MOV AL,atryb_10
103         MOV ES:[SI+161],AL
104     ADD BX,1
105         ADD SI,2
106     MOV AL,[BX]
107     LOOP petla10
108     IRETD
109 exc_10     ENDP
110
111 exc_11     proc
112         MOV AX,32
113     MOV ES,AX
114     MOV BX,OFFSET tekst_x
115     MOV CX,15
116     MOV AL,[BX]
117     MOV SI,0
118 petla11:
119     MOV ES:[SI+160],AL
120         MOV AL,atryb
121         MOV ES:[SI+161],AL
122     ADD BX,1
123         ADD SI,2
124     MOV AL,[BX]
125     LOOP petla11
126     IRETD
127 exc_11     endp
128
129 exc_12     proc
130         MOV AX,32
131     MOV ES,AX
132     MOV BX,OFFSET tekst_x
133     MOV CX,15
134     MOV AL,[BX]
135     MOV SI,0
136 petla12:
137     MOV ES:[SI+160],AL
138         MOV AL,atryb
139         MOV ES:[SI+161],AL
140     ADD BX,1
141         ADD SI,2
142     MOV AL,[BX]

```

```

143     LOOP petla12
144         IRETD
145 exc_12     endp
146
147 exc_13     PROC ; Wyjatek nr 13
148             MOV AX,32
149             MOV ES,AX
150             MOV BX,OFFSET tekst13
151             MOV CX,36
152             MOV AL,[BX]
153             MOV SI,0
154 petla13 :
155             MOV ES:[SI+160],AL
156             MOV AL,atryb13
157             MOV ES:[SI+161],AL
158             ADD BX,1
159             ADD SI,2
160             MOV AL,[BX]
161             LOOP petla13
162             IRETD
163 exc_13     ENDP
164
165 exc_14     proc
166             MOV AX,32
167             MOV ES,AX
168             MOV BX,OFFSET tekst_x
169             MOV CX,15
170             MOV AL,[BX]
171             MOV SI,0
172 petla14 :
173             MOV ES:[SI+160],AL
174             MOV AL,atryb
175             MOV ES:[SI+161],AL
176             ADD BX,1
177             ADD SI,2
178             MOV AL,[BX]
179             LOOP petla14
180             IRETD
181 exc_14     endp
182
183 exc_       PROC ; Procedura obsługi wyjątku nr 0
184             MOV AX,32 ; (dzielenie przez 0)
185             MOV ES,AX ; Wyświetlenie na ekranie tekstu TEKST_0
186             MOV BX,OFFSET tekst_x
187             MOV CX,15
188             MOV AL,[BX]
189             MOV SI,0
190 petla_ :

```

```

191  MOV ES:[SI+160],AL
192      MOV AL, atryb
193      MOV ES:[SI+161],AL
194      ADD BX,1
195      ADD SI,2
196  MOV AL,[BX]
197  LOOP petla_
198      IRETD
199 exc_ ENDP

```

#### 4.5 *TXTPUL.TXT* - tekst i atrybuty użyte w kodzie obsługi pułapek

```

1  tekst_x      DB 'Obsługa_wyjatku'
2  atryb        DB 02h
3
4  tekst_0      DB 'Dzielenie_przez_zero!'
5  atryb_0      DB 02h
6
7  tekst_1      DB 'Przykładowe_przerwanie'
8  atryb_1      DB 02h
9
10 tekst_2      DB 'Aktywne_zadanie_numer_1'
11 atryb_2      DB 02h
12
13 tekst_3      DB 'Aktywne_zadanie_numer_2'
14 atryb_3      DB 02h
15
16 tekst_10     DB 'Bledny_segment_TSS'
17 atryb_10     DB 02h
18
19 tekst13      DB 'Ogólne_naruszenie_mechanizmu_ochrony'
20 atryb13      DB 02h

```

#### 4.6 *RODZPUL.TXT* - lista pułapek

```

1  exc0      trap <exc_0>
2  exc1      trap <exc_1>
3  exc2      trap <exc_2>
4  exc3      trap <exc_3>
5  exc4      trap <exc_4>
6  exc5      trap <exc_5>
7  exc6      trap <exc_6>
8  exc7      trap <exc_7>
9  exc8      trap <exc_8>
10 exc9      trap <exc_9>
11 exc10     trap <exc_10>
12 exc11     trap <exc_11>
13 exc12     trap <exc_12>
14 exc13     trap <exc_13>

```

```
15 exc14      trap <exc_>
16 trap 17 DUP (<exc_>)
```

## 5 Bibliografia

### Literatura

- [1] W. Stanisławski, D.Raczyński *Programowanie systemowe mikroprocesorów rodziny x86*, PWN, Warszawa 2010. ISBN 978-83-01-16383-9.
- [2] J. Biernat, *Architektura komputerów*, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2005. ISBN 83-7085-878-3.
- [3] G.Syck, *Turbo Assembler - Biblia użytkownika*, LT&P, Warszawa 1994. ISBN 83-901237-2-X.
- [4] J. Bielecki, *Turbo Assembler*, PLJ, Warszawa 1991. ISBN 83-85190-10-4.