Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №4 по дисциплине

Архитектура вычислительных систем

Студент Глубоков Г.В.

Группа АИ-20

Руководитель Болдырихин О. В.

Липецк 2022 г.

Цель работы

Изучение сегментирования и обработки прерываний в защищенном режиме процессоров IA-32

Задание кафедры

Написать на языке ассемблера программу, выполняющую ввод, обработку и вывод в защищенном режиме.

Как и в предыдущей работе программа должна включать подпрограммы. Хотя бы одна из подпрограмм должна быть дальней.

Главная программа выполняет необходимую инициализацию, в том числе готовит необходимые структуры данных: глобальную таблицу дескрипторов и таблицу дескрипторов прерываний – и переключает процессор в защищенный режим. Далее в защищенном режиме вызывается подпрограмма ввода с клавиатуры, которая последовательно заполняет массив исходных данных в памяти. Главная программа последовательно передает исходные данные подпрограмме преобразования через стек. Подпрограмма проверяет исходные данные на допустимость или корректность, при положительном исходе проверки осуществляет преобразование и возвращает результат главной программе. Главная программа размещает результаты в другой области памяти также в виде массива. Результаты выводятся на дисплей с помощью соответствующей подпрограммы.

Исследовать работу программы в обычном режиме, ввести исходные данные, получить результаты, сделать скриншот.

Исследовать работу программы в отладчике до загрузки регистра таблицы дескрипторов прерываний. По результатам исследования определить поля базы дескрипторов сегментов глобальной таблицы дескрипторов, значения регистров GDTR и IDTR.

По листингу определить моменты переключения процессора в защищенный и реальный режим, перехода в код защищенного режима и возврата в код реального режима.

Выполнить необходимые сравнения, сделать выводы.

№	Задача, выполняемая программой	Расположение GDT в программе	Расположение обработчика прерываний клавиатуры
8	Преобразование из кода с контролем по четности	В отдельном сегменте	В сегменте кода защищенного режима

Листинг программы

```
1                                       ;tasm /m PM4IO.asm
2                                       ;tlink      /3 PM4IO.obj
3                                       .386p
4                                       00000000                    SSeg  segment stack
5                                       00000000  0100*(??)        Sbegin        db 100h
                                        dup(?)
6                                       =0100                       Ssize = $ - Sbegin
7                                       00000100                    SSeg  ends
      8
9       00000000                  DSeg32        segment use32
10      00000000  02*(????????)                X dd 2       dup(?)
11      00000008  02*(????????)                Y dd 2       dup(?)
12      00000010                  DSeg32        ends
     13
14              00000000                GDT_Seg segment use32
15              00000000                        GDT  label byte
16              00000000  08*(00)                     db 8 dup(0)
17              00000008  FF FF  00 00 00 9A 00+            CS16Dsc db
0FFh,0FFh,0,0,0,10011010b,0,0
18              00
19              00000010  FF FF  00 00 00 9A CF+            CS32Dsc db
0FFh,0FFh,0,0,0,10011010b,11001111b,0
20              00
21              00000018  FF FF  00 00 00 92 CF+            DS32Dsc db
0FFh,0FFh,0,0,0,10010010b,11001111b,0
22              00
23              00000020  FF FF  00 00 00 92 CF+            SS32Dsc db
0FFh,0FFh,0,0,0,10010010b,11001111b,0
24              00
25              00000028  FF FF  00 80 0B 92 CF+            VSegDsc db
0FFh,0FFh,0,80h,0Bh,10010010b,11001111b,0
26              00
27              00000030  FF FF  00 00 00 9A CF+            GDT_Dsc   db
0FFh,0FFh,0,0,0,10011010b,11001111b,0
28              00
29              00000038  FF FF  00 00 00 9A CF+            CSPrDsc   db
0FFh,0FFh,0,0,0,10011010b,11001111b,0
30              00
31              =0040                                   GDT_l = $-GDT
     32
     33     00000040  003F                              gdtr  dw GDT_l-1
34    00000042  ????????                dd ?
35              =0008                           CS16Sel equ 0000000000001000b
36              =0010                           CS32Sel equ 0000000000010000b
37              =0018                           DS32Sel equ 0000000000011000b
38              =0020                           SS32Sel equ 0000000000100000b
39              =0028                           VSegSel equ 0000000000101000b
40              =0030                           GDT_Sel   equ 0000000000110000b
41              =0038                           CSPrSel   equ 0000000000111000b
     42
    43      00000046                  GDT_Seg ends
     44
45                                        0000                      CSeg16        segment
                                          use16
46                                        assume      CS:CSeg16, DS:CSeg32,
SS:SSeg,      ES:GDT_Seg
     47
48          0000  68 0000s        start:        push CSeg32
```

```
49          0003  1F                      pop ds
50          0004  68 0000s                push GDT_Seg
51          0007  07                      pop es
    52
53          0008  E4 92                         in al,       92h
54          000A  0C 02                         or al,       2
55          000C  E6 92           out 92h, al
    56
57          000E  66| 33 C0               xor eax, eax
58          0011  8C C8                         mov ax, cs
59          0013  66| C1 E0 04                  shl eax, 4
60          0017  26: 67| A3 0000000Ar    mov word ptr CS16Dsc+2, ax
61          001E  66| C1 E8 10                  shr eax, 16
62          0022  26: 67| A2 0000000Cr    mov byte ptr CS16Dsc+4, al
63          0029  B8 0000s                mov ax, CSeg32
64          002C  66| C1 E0 04                  shl eax, 4
65          0030  66| 50                 push eax
66          0032  26: 67| A3 00000012r    mov word ptr CS32Dsc+2, ax
67          0039  66| C1 E8 10                  shr eax, 16
68          003D  26: 67| A2 00000014r    mov byte ptr CS32Dsc+4, al
69          0044  B8 0000s                mov ax, DSeg32
70          0047  66| C1 E0 04                  shl eax, 4
71          004B  26: 67| A3 0000001Ar    mov word ptr DS32Dsc+2, ax
72          0052  66| C1 E8 10                  shr eax, 16
73          0056  26: 67| A2 0000001Cr    mov byte ptr DS32Dsc+4, al
74          005D  B8 0000s                mov ax, SSeg
75          0060  66| C1 E0 04                  shl eax, 4
76          0064  26: 67| A3 00000022r    mov word ptr SS32Dsc+2, ax
77          006B  66| C1 E8 10                  shr eax, 16
78          006F  26: 67| A2 00000024r    mov byte ptr SS32Dsc+4, al
79          0076  B8 0000s                mov ax, GDT_Seg
80          0079  66| C1 E0 04                  shl eax, 4
81          007D  66| 50                 push eax
82          007F  26: 67| A3 00000032r    mov word ptr GDT_Dsc+2, ax
83          0086  66| C1 E8 10                  shr eax, 16
84          008A  26: 67| A2 00000034r    mov byte ptr GDT_Dsc+4, al
85          0091  B8 0000s                mov ax, CSegPr
86          0094  66| C1 E0 04                  shl eax, 4
87          0098  26: 67| A3 0000003Ar    mov word ptr CSPrDsc+2, ax
88          009F  66| C1 E8 10                  shr eax, 16
89          00A3  26: 67| A2 0000003Cr    mov byte ptr CSPrDsc+4, al
    90
    91
    92
93              00AA  66| 58                       pop eax
94              00AC  66| 05 00000000r             add eax, offset GDT
95              00B2  66| 26: 67| A3      +        mov dword ptr gdtr+2, eax
96              00000042r
    97
98              00BA  26: 67| 0F 01    15    +                 lgdt fword ptr     gdtr
99              00000040r
   100
101          00C3  68 0000s                push CSeg32
102          00C6  07                      pop es
   103
   104
105          00C7  66| 58                 pop eax
106          00C9  66| 05 00000000r       add eax, offset IDT
107          00CF  66| 67| A3 00000802r           mov dword ptr idtr+2, eax
   108
```

```
109             00D6  67| 0F 01 1D          +              lidt fword ptr     idtr
110             00000800r
    111
    112         00DE  68 0000s          push GDT_Seg
    113         00E1  1F                       pop ds
    114
    115         00E2  FA                       cli
    116
    117
    118         00E3  0F 20     C0                  mov eax, cr0
    119         00E6  0C 01           or al,       1
    120         00E8  0F 22     C0                  mov cr0, eax
    121
122         00EB  66                   db 66h
123         00EC  EA                   db 0EAh
124         00ED  0000089Dr            dd offset PMentry
125         00F1  0010                       dw CS32Sel
    126
127         00F3  0F 20   C0           RMret:       mov eax, cr0
128         00F6  24 FE                      and al, 0FEh
129         00F8  0F 22   C0                 mov cr0, eax
130         00FB  EA                   db 0EAh
131         00FC  0100r                      dw $+4
132         00FE  0000s                      dw CSeg16
    133
134         0100  BA 0000s             mov dx, SSeg
135         0103  8E D2                      mov ss, dx
136         0105  BC 0100              mov sp, SSize
    137
138             0108  B8 0000s               mov ax, CSeg32
139             010B  8E D8                      mov ds, ax
140             010D  67| 0F 01 1D          +              lidt fword ptr
                  idtr_real
141             00000806r
    142
   143          0115  FB                       sti
    144
145         0116  B4 00                      mov ah, 0
146         0118  CD 16                      int 16h
    147
148         011A  B4 4C                      mov ah, 4Ch
149         011C  CD 21                      int 21h
150         011E                CSeg16       ends
    151
152                                 00000000               CSegPr       segment
                                       use32
153                                 assume    cs:CSegPr, ds:DSeg32
154                                 00000000               outnum       proc
155                                 00000000  50                   push eax
156                                 00000001  53                   push ebx
157                                 00000002  AD                   lodsd
158                                 00000003  BB 80000000              mov ebx,
                                    80000000h
159                                 00000008  85 C3               olloop: test eax,
                                    ebx
160                                 0000000A  75 08                       jnz bit1
161                                 0000000C  66| 26: C7 07       0730    mov word ptr
                                    es:[edi],        0730h
162                                 00000012  EB 06                     jmp NxtBit
```

```
163                                 00000014  66| 26: C7 07       0731  bit1:  mov word
                                    ptr es:[edi],    0731h
164                                 0000001A  47            NxtBit: inc edi
165                                 0000001B  47                    inc edi
166                                 0000001C  D1 EB                         shr ebx, 1
167                                 0000001E  73 E8                         jnc olloop
168                                 00000020  66| 26: C7 07       0720    mov word ptr
                                    es:[edi],        0720h
169                                 00000026  47                    inc edi
170                                 00000027  47                    inc edi
171                                 00000028  5B                    pop ebx
172                                 00000029  58                    pop eax
173                                 0000002A  CB                    db 0CBh
174                                 0000002B                outnum      endp 175
                                       0000002B                     CSegPr      ends
    176
    177                                         00000000                    CSeg32
                                                segment use32
    178                                         assume        cs:CSeg32, ds:DSeg32
    179
    180     00000000                      IDT  label byte
    181
182                                 ;INT 00 - 07
183                                 00000000  08*(080Cr 0010 8E00 +       dw 8 dup(small
                                       offset int_handler,CS32Sel,8E00h,0)
184                                 0000)
185                                 ;INT 08 (irq0)
186                                 00000040  080Dr  0010 8E00 0000              dw
                                    small offset irq0_7_handler,CS32Sel,8E00h,0
187                                 ;INT 09 (irq1)
188                                 00000048  0814r  0010 8E00 0000              dw
                                    small offset
irq1_handler,CS32Sel,8E00h,0
    189                                 ;INT 0Ah - 0Fh      (IRQ2 -       IRQ8)
    190     00000050  06*(080Dr 0010 8E00 +           dw 6 dup(small      offset
irq0_7_handler,CS32Sel,8E00h,0)
191                                 0000)
192                                 ;INT 10h - 6Fh
193                                 00000080  61*(080Cr 0010 8E00 +       dw 97
                                    dup(small offset       int_handler,CS32Sel,8E00h,0)
194                                 0000)
    195                                 ;INT 70h - 78h      (IRQ8 -       IRQ15)
    196     00000388  08*(0845r 0010 8E00 +           dw 8 dup(small      offset
irq8_15_handler,CS32Sel,8E00h,0)
197                                 0000)
198                                 ;INT 79h - FFh
199                                 000003C8  87*(080Cr 0010 8E00 +       dw 135
                                       dup(small offset int_handler,CS32Sel,8E00h,0)
200                                 0000)
201                                 =0800                    idt_size = $-IDT
    202
    203     00000800  07FF                     idtr  dw idt_size-1
204  00000802  ????????          dd ?
    205
    206     00000806  03FF 0000 0000  idtr_real dw 3FFh,0,0
207
208     0000080C                int_handler:
209     0000080C  CF                    iretd
    210
211     0000080D                irq0_7_handler:
```

```
212     0000080D  50                    push eax
213     0000080E  B0 20                        mov al, 20h
214     00000810  E6 20                        out 20h, al
215     00000812  58                    pop eax
216     00000813  CF                    iretd
    217
   218      00000814                    irq1_handler:
    219
220     00000814  50                    push eax
221     00000815  06                    push es
222     00000816  E4 60                        in al,       60h
223     00000818  3C 02                        cmp al, 02h
224     0000081A  74 0E                        je num1
225     0000081C  3C 0B                        cmp al, 0Bh
226     0000081E  75 18                        jne skip
227     00000820  D1 E2                        shl edx, 1
228     00000822  66| 26: C7 07  0730   mov word ptr es:[edi],    0730h
229     00000828  EB 0B                        jmp NxtNum
230     0000082A  D1 E2                        num1: shl edx,     1
231     0000082C  83 CA   01                   or edx, 1
232     0000082F  66| 26: C7 07  0731   mov word ptr es:[edi],    0731h
233     00000835  47                    NxtNum: inc edi
234     00000836  47                    inc edi
235     00000837  49                    dec ecx 236  00000838                   skip:
    237     00000838  E4 61                         in al,       61h
    238     0000083A  0C 80                         or al,       80h
    239     0000083C  E6 61            out 61h, al
    240
241     0000083E  B0 20                        mov al, 20h
242     00000840  E6 20                        out 20h, al
243     00000842  07                    pop es
244     00000843  58                    pop eax
245     00000844  CF                    iretd
    246
247     00000845                irq8_15_handler:
248     00000845  50                    push eax
249     00000846  B0 20                        mov al, 20h
250     00000848  E6 A1                        out 0A1h, al
251     0000084A  58                    pop eax
252     0000084B  CF                    iretd
    253
254     0000084C                inpp  proc
255     0000084C  52                    push edx
256     0000084D  51                    push ecx
257     0000084E  50                    push eax
258     0000084F  33 D2                        xor edx, edx
259     00000851  33 C9                        xor ecx, ecx
260     00000853  B9 00000020                  mov ecx, 20h
261     00000858  83 F9   00            inn: cmp ecx, 0
262     0000085B  75 FB                        jne inn
263     0000085D  89 16                        mov ds:[esi], edx
264     0000085F  83 C6   04                   add esi, 4
265     00000862  83 C7   02                   add edi, 2
266     00000865  58                    pop eax
267     00000866  59                    pop ecx
268     00000867  5A                    pop edx
269     00000868  C3                    ret
270     00000869                inpp  endp
    271
272     00000869                mainproc     proc
```

```
273     00000869  8B EC                       mov ebp, esp
274     0000086B  51                      push ecx
275     0000086C  8B 45   04                mov eax, [ebp + 4]
276     0000086F  33 DB                     xor ebx, ebx
277     00000871  85 C0                 c1: test eax, eax
278     00000873  74 07                     je end_c1
279     00000875  D1 E8                     shr eax, 1
280     00000877  83 D3   00                adc ebx, 0     281 0000087A  EB F5
                          jmp c1     282     0000087C                    end_c1:
283     0000087C  8B C3                     mov eax, ebx
284     0000087E  A9 00000001               test eax, 1b
285     00000883  74 02                     jz tr
286     00000885  75 0C                     jnz fal
287     00000887  8B 45   04            tr: mov eax, [ebp + 4]
288     0000088A  25 7FFFFFFF               and eax, 7FFFFFFFh
289     0000088F  8B D8                     mov ebx, eax
290     00000891  EB 05                     jmp ext
291     00000893  BB 00000000           fal: mov ebx, 0
292     00000898                   ext:
    293
  294     00000898  89 5D       04                mov [ebp+4], ebx
  295     0000089B  59                    pop ecx
  296     0000089C  C3                    ret
  297     0000089D                        mainproc endp
    298
    299     0000089D                      PMentry:
    300
  301     0000089D  66| BA 0018               mov dx, DS32Sel
  302     000008A1  8E DA                           mov ds, dx
  303     000008A3  66| BA 0020         mov dx, SS32Sel
  304     000008A7  8E D2                           mov ss, dx
  305     000008A9  BC 00000100               mov esp, Ssize
  306     000008AE  66| BA 0028               mov dx, VSegSel
  307     000008B2  8E C2                           mov es, dx
  308     000008B4  33 FF                           xor edi, edi
    309
310     000008B6  B8 07200720               mov eax, 07200720h
311     000008BB  B9 000003E8               mov ecx, 80*25*2/4
312     000008C0  F3> AB                rep stosd
313     000008C2  33 FF                     xor edi, edi
    314
   315      000008C4  FB                     sti
    316
317     000008C5  BE 00000000r              lea esi, X
318     000008CA  B9 00000002               mov ecx, 2
319     000008CF  33 D2                     xor edx, edx
320     000008D1  E8 FFFFFF76           ixloop: call inpp
321     000008D6  E2 F9                     loop ixloop
    322
323     000008D8  66| BA 0018               mov dx, DS32Sel
324     000008DC  8E C2                     mov es, dx
    325
326     000008DE  BE 00000000r              lea esi, x
327     000008E3  BF 00000008r              lea edi, y     328 000008E8
        B9 00000002                     mov ecx, 2     329 000008ED
                  mloop:
330     000008ED  AD                    lodsd
331     000008EE  50                    push eax
332     000008EF  E8 FFFFFF75               call mainproc
333     000008F4  58                    pop eax
```

```
334     000008F5  AB                    stosd
335     000008F6  E2 F5                 loop  mloop
    336
337     000008F8  66| BA 0028           mov dx,VSegSel
338     000008FC  8E C2                 mov es,dx
339     000008FE  33 FF                 xor edi, edi
    340
341     00000900  BE 00000008r          lea esi, y
342     00000905  33 FF                 xor edi, edi
343     00000907  BF 00000140           mov edi, 80*4
344     0000090C  BB 80000000           mov ebx, 80000000h
345     00000911  B9 00000002           mov ecx, 2
346     00000916                obloop: ;call outnum
347     00000916  9A                    db 9Ah
348     00000917  00000000r             dd offset outnum
349     0000091B  0038                  dw CSPrSel
350     0000091D  E2 F7                 oloop:loop obloop
    351
352                                       0000091F  EA                db 0EAh
353                                       00000920  000000F3r           dd offset
                                          RMret
354                                       00000924  0008                      dw
                                          CS16Sel
355                                       00000926              CSeg32        ends
356                                       end start
```

Исследование работы программы

Процесс выполнения кода реального режима приведен в таблице 1.

Таблица 1 - Состояния системы после выполнения каждой команды

№	Адрес команды	Команда на машинном языке	Команда на языке ассемблера	IP	IR	Содержимое изменившихся регистров, ячеек памяти и портов ввода-вывода
1	0000	68A408	push 08A4	0003	68	SP=00FE SS[00FE]=08A4
2	0003	1F	pop ds	0004	1F	DS=08A7 SP=0100
3	0004	688A08	push 088A	0007	68	SP=00FE SS[00FE]=088A

4	0007	07	pop es	0008	07	DS=088A SP=0100
5	0008	E492	in al, 92	000A	E4	AX=0002
6	000A	0C02	or al, 02	000C	0C	C=0 Z=0 S=0 O=0 P=0 A=0
7	000C	E692	out 92, al	000E	E6	Порт 92h=02
8	000E	6633C0	xor eax, eax	0011	6633C0	EAX=00000000 C=0 Z=1 S=0 O=0 P=1 A=0
9	0011	8CC8	mov ax, cs	0013	8CC8	AX=088F
10	0013	66C1E004	shl eax, 04	0017	66C1E 0	EAX=000088F0 C=0 Z=0 S=0 O=0 P=1 A=1
11	0017	67A30A 000000	mov [0000 000A], ax	001E	67A3	DS[0000 000A]=88F0

12	001E	66C1E810	shr eax, 10	0022	66C1E 8	EAX=0 C=1 Z=1 S=0 O=0 P=1 A=1
13	0022	67A20C 000000	mov [0000 0000C], al	0029	67A2	DS[0000000C]= 0
14	0029	B8A408	mov ax, 08A4	002C	B8	AX=08A4

15	002C	66C1E004	shl eax, 04	0030	66C1E0	EAX=00008A40 C=0 Z=0 S=0 O=0 P=1 A=1
16	0030	6650	push eax	0032	6650	SP=00FC SS[00FE]=0000 SS[00FC]=8A40
17	0032	67A312 000000	mov [0000 0012], ax	0039	67A3	DS[0000 0012]=8A40
18	0039	66C1E810	shr eax, 10	003D	66C1E8	EAX=0 C=1 Z=1 S=0 O=0 P=1 A=1
19	003D	67A214 000000	mov [0000 0014], al	0044	67A2	DS[00000014]=0
20	0044	B88908	mov ax, 0889	0047	B8	AX=0889
21	0047	66C1E004	shl eax, 04	004B	66C1E0	EAX=00008890 C=0 Z=0 S=0 O=0 P=1 A=1
22	004B	67A31A 000000	mov [0000 001A], ax	0052	67A3	DS[0000 001A]=8890
23	0052	66C1E810	shr eax, 10	0056	66C1E8	EAX=0

						C=1 Z=1 S=0 O=0 P=1 A=1

24	0056	67A21C 000000	mov [0000 001C], al	005D	67A2	DS[0000001C]= 0
25	005D	B87908	mov ax, 0879	0060	B8	AX=0879
26	0060	66C1E004	shl eax, 04	0064	66C1E 0	EAX=00008790 C=0 Z=0 S=0 O=0 P=1 A=1
27	0064	67A322 000000	mov [0000 0022], ax	006B	67A3	DS[0000 0022]=8790
28	006B	66C1E810	shr eax, 10	006F	66C1E 8	EAX=0 C=1 Z=1 S=0 O=0 P=1 A=1
29	006F	67A224 000000	mov [0000 0024], al	0076	67A2	DS[00000024]=0
30	0076	B88A08	mov ax, 088A	0079	B8	AX=088A
31	0079	66C1E004	shl eax, 04	007D	66C1E 0	EAX=000088A0 C=0 Z=0 S=0 O=0 P=0 A=1
32	007D	6650	push eax	007F	6650	SP=00F8 SS[00F8]=88A0 SS[00FA]=0000
33	007F	2667A332 000000	mov es: [0000 0032], ax	0086	2667A3	ES[0000 0032]=88A0

34	0086	66C1E810	shr eax, 10	008A	66C1E8	EAX=0

						C=1 Z=1 S=0 O=0 P=1 A=1
35	008A	2667A234 000000	mov es: [0000 0034], al	0091	2667A2	ES[00000034]=0
36	0091	B8A108	mov ax, 08A1	0094	B8	AX=08A1
37	0094	66C1E004	shl eax, 04	0098	66C1E0	EAX=00008BA1 C=0 Z=0 S=0 O=0 P=0 A=1
38	0098	2667A33A 000000	mov es: [0000 003A], ax	009F	2667A3	ES[0000 003A]=8BA1
39	009F	66C1E810	shr eax, 10	00A3	66C1E8	EAX=0 C=1 Z=1 S=0 O=0 P=1 A=1
40	00A3	2667A23C 000000	mov [0000 003C], al	00AA	2667A2	DS[0000003C]=0
41	00AA	6658	pop eax	00AC	6658	EAX=000088A0 SP=00FC
42	00AC	660500 000000	add eax, 00000000	00B2	6605	EAX=000088A0 C=0 Z=0 S=0 O=0 P=1 A=0

43	00B2	662667 A342000000	mov [0000 0042], eax	00BA	6626 67A3	DS[00000042]= 000088A0
44	00BA	26670F01154 0 000000	lgdt [0000 0040]	00C3	26670F 0115	GDTR=0000 88A0003F
45	00C3	68A408	push 08A4	00C6	68	SP=00FA SS[00FA]=08A4

46	00C6	07	pop es	00C7	07	ES=08A4 SP=00FC
47	00C7	6658	pop eax	00C9	6658	SP=0100 EAX=00008A40
48	00C9	660500 000000	add eax, 00000046	00CF	6605	EAX=00008A40 C=0 Z=0 S=0 O=0 P=0 A=0
49	00CF	6667A302 080000	mov [0000 0802], eax	00D6	6667A3	DS[00000802]= 00008A40
50	00D6	670F011D 00080000	lidt [0000 0800]	00DE	670F01 1D	IDTR=00008A 4007FF
51	00DE	688A08	push 088A	00E1	688D08	SP=00FE SS[00FE]=088A
52	00E1	1F	pop ds	00E2	1F	DS=088A SP=100
53	00E2	FA	cli	00E3	FA	I=0
54	00E3	0F20C0	mov eax, cr0	00E6	0F20	

55	00E6	0C01	or al, 01	00EB	0C	
56	00E8	0F22C0	mov cr0, eax	00F3	0F22C0	Бит PE=1
57	00EB	66EA9D08 00001000	jmp 0010: 0000089D	0010: 0000 0893	66EA	
Работа программы в защищенном режиме						
58	00F3	0F20C0	mov eax, cr0	00F6	0F20C0	
59	00F6	24FE	and al, FE	00F8	24	
60	00F8	0F22C0	mov cr0, eax	00FB	0F22C0	Бит PE=0
61	00FB	EA00019208	jmp 088F:0100	0100	EA	
62	0100	BA7908	mov dx, 0879	0103	BA	DX=0879
63	0103	8ED2	mov ss, dx	0105	8ED2	SS=0879
64	0105	BC0001	mov sp, 0100	0108	BC	SP=0100
65	0108	B8A408	mov ax, 08A4	010B	B8	AX=08A7
66	010B	8ED8	mov ds, ax	010D	8ED8	DS=08A7

67	010D	670F011D 06080000	lidt [00000806]	0115	670F01 1D	IDTR=00000000 03FF
68	0115	FB	sti	0116	FB	I=1
69	0116	B400	mov ah, 00	0118	B4	AH=00
70	0118	CD16	int 16	011A	CD	
71	011C	B44C	mov ah, 4C	011E	B4	AH=4C
72	011E	CD21	int 21		CD	

Пример работы программы приведен на рисунке 1.

Рисунок 1 – Пример работы программы

Значения полей базы дескрипторов сегментов и регистров GDTR и IDTR:

- CSeg16Dsc: 000088F0
- CSeg32Dsc: 00008A40
- DSeg32Dsc: 00008890
- SSegDsc: 00008790
- GDT_Dsc: 000088A0
- CSegPrDsc: 00008BA1 GDTR: 000088A0003F
- IDTR: 00008A4007FF

Анализ результатов исследования

Переключение процессора в защищенный режим осуществляется установкой бита PE:

```
mov eax, cr0
or al, 1 mov
cr0, eax
```

Переключение процессора обратно в реальный режим осуществляется сбросом бита PE: mov eax, cr0 and al, 0FEh mov cr0, eax

Переход в код защищенного режима осуществляется командами:

```
db 66h db 0EAh                ;код команды JMP FAR
dd offset PMentry  ;смещение внутри сегмента dw CS32Sel
                ;адрес сегмента кода защищенного режима
```

Возврат в код реального режима осуществляется командами:

```
db 0EAh                ;код команды JMP FAR
dd offset RMret     ;смещение внутри сегмента dw CS16Sel
;адрес сегмента кода реального режима
```

Схема перехода процессора к обработчику прерывания клавиатуры от нажатия клавиши до выполнения первой команды обработчика выглядит следующим образом:

Работой клавиатуры управляет специальная электронная схема – контроллер клавиатуры. В его функции входит распознавание нажатой клавиши и помещение закрепленного за ней кода в свой выходной регистр (порт) с номером 60h. Код клавиши, поступающий в порт, называется сканкодом и является, по существу, порядковым номером клавиши.

Как нажатие, так и отпускание любой клавиши вызывает сигнал аппаратного прерывания, заставляющий процессор прервать выполняемую программу и перейти на программу системного обработчика прерываний от клавиатуры.

Вначале процессор получает номер прерывания, который представляет собой индекс соответствующего дескриптора в IDT. Затем процессор читает из шлюза прерывания селектор сегмента кода, в котором находится обработчик прерывания и смещение обработчика прерывания от начала этого сегмента. По селектору он находит дескриптор, из которого извлекает базу сегмента и, таким образом, определяет полный логический адрес обработчика, он же является адресом первой команды обработчика.

Сравнения:

1. GDT (Global Descriptor Table) и LDT (Local Descriptor Table) - таблицы глобальных и локальных дескрипторов. Это таблицы 8-байтных структур, называемых дескрипторами сегментов, где находится начальный адрес сегмента вместе с другой необходимой информацией. При адресации в защищенном режиме в сегментных регистрах находятся специальные 16битные структуры, называемые селекторами. Бит 2 селектора является индикатором использования одной из таблиц дескрипторов. Если данный бит равен 0, то используется GDT, а если данный бит равен 1, то используется LDT.

Операционная система собирает все таблицы дескрипторов, чтобы процессор знал, где искать дескрипторы, и при необходимости загружает их при помощи привилегированных команд процессора. При этом GDT может быть только одна, а LDT – на каждую задачу.

Внешние прерывания, программные прерывания, исключения обрабатываются с использованием таблицы дескрипторов прерываний

(Interrupt Descriptor Table, IDT). Исключение – это событие, которое происходит, если команда вызывает ошибку. Например, попытка деления на ноль генерирует исключение. Однако есть исключения, например, контрольные точки, которые происходят при других условиях. IDT содержит множество дескрипторов различных шлюзов: прерываний, ловушек и задач – которые предоставляют доступ к обработчикам прерываний и исключений. Так же как и GDT, IDT не является сегментом. Линейный базовый адрес и лимит IDT содержатся в регистре таблицы дескрипторов прерываний (Interrupt Descriptor Table Register).

2. Дескрипторы могут быть несистемными (дескриптор сегмента кода или сегмента данных) и системными, среди которых можно выделить специальные (дескрипторы шлюзов): шлюз вызова, ловушки, прерывания или задачи.

Если в дескрипторе бит четвертого байта доступа равен 0, дескриптор называется системным. В этом случае биты от нулевого до третьего байта доступа определяют один из 16 возможных типов дескриптора.

Шлюзы прерываний и ловушек используются для вызова обработчиков соответственно прерываний и исключений типа ловушки. Они указывают точку входа обработчика, его разрядность и уровень привилегий. При передаче управления обработчику процессор помещает в стек флаги и адрес возврата так же, как и в реальном режиме, но после этого для некоторых исключений в стек помещается дополнительный код ошибки, откуда следует, что не все обработчики можно завершать простой командой IRETD (IRET). Единственное различие между шлюзом прерывания и ловушки состоит в том, что при передаче управления через шлюз прерывания автоматически запрещаются дальнейшие прерывания, пока обработчик не выполнит IRETD (IRET).

3. При внутрисегментном (ближнем) вызове подпрограммы при переходе в сегмент с теми же привилегиями в реальном режиме, режиме x86 или в защищенном режиме в стек помещается только смещение команды, следующей за командой CALL, от начала сегмента кода, то есть текущее значение регистра IP, а при дальнем вызове — полный логический адрес (пара CS:IP или CS:EIP).

Вывод

При выполнении лабораторной работы я написал на языке ассемблера программу, выполняющую преобразование числа из кода с контролем по четности в защищенном режиме. Исходные данные вводятся в память с клавиатуры, результаты выводятся на дисплей также в защищенном режиме. Вывод результатов на экран выполняет дальняя подпрограмма.

В результате выполнения лабораторной работы я узнала следующее:

Надежность системы, функционирующей в защищенном режиме, обусловлена следующим:

1) возможность задания необходимого размера сегмента и контролем адресации памяти вне пределов сегментов.

2) байт доступа дескриптора сегмента кода содержит бит разрешения чтения сегмента (бит 1). Если этот бит установлен в 1, программа может считывать содержимое сегмента кода. В противном случае процессор может только выполнять этот код, т. е. программа не может модифицировать сегмент кода. Это означает невозможность создания самомодифицирующихся программ для защищенного режима. Впрочем, возможность модификации кода остается. Для сегмента кода можно создать еще один, алиасный дескриптор, в котором этот сегмент отмечен как сегмент данных. Для него можно разрешить запись, установив тот же самый бит 1, и модифицировать код программы во время ее выполнения.