



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ЛИПЕЦКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Факультет автоматизации и информатики
Кафедра автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №4

по дисциплине “Архитектура вычислительных систем”

Студент АС-21-1 _____ Станиславчук С. М.
(подпись, дата)

Руководитель
Ст. преподаватель _____ Болдырихин О. В.
(подпись, дата)

Липецк 2023

Цель работы

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

Ход выполнения лабораторной работы

1. 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 04*(????????) X dd 4 dup(?)
11 00000010 04*(????????) Y dd 4 dup(?)
12 00000020 00C0 bit_mask dw 1100000000000000b
13 00000022 0000 errors_count dw 0b
14 00000024 DSeg32 ends
15
16 00000000 IDT_Seg segment use32
17 assume CS:IDT_Seg
18
19 00000000 irq1_handler:
20
21 00000000 50 push eax
22 00000001 06 push es
23
24 00000002 E4 60 in al, 60h
25 00000004 3C 0B cmp al, 0Bh
26 00000006 75 0E jne Not0
27 00000008 C1 E2 04 shl edx, 4
28 0000000B 66| 26: C7 07 0730 mov word ptr es:[edi], 0730h
29 00000011 E9 00000126 jmp NxtNum
30 00000016 3C 02 Not0: cmp al, 02h
31 00000018 75 11 jne Not1
32 0000001A C1 E2 04 shl edx, 4
33 0000001D 83 CA 01 or edx, 1h
```

34	00000020	66 26: C7 07 0731		mov word ptr es:[edi], 0731h
35	00000026	E9 00000111		jmp NxtNum
36	0000002B	3C 03	Not1:	cmp al, 03h
37	0000002D	75 11		jne Not2
38	0000002F	C1 E2 04		shl edx, 4
39	00000032	83 CA 02		or edx, 2h
40	00000035	66 26: C7 07 0732		mov word ptr es:[edi], 0732h
41	0000003B	E9 000000FC		jmp NxtNum
42	00000040	3C 04	Not2:	cmp al, 04h
43	00000042	75 11		jne Not3
44	00000044	C1 E2 04		shl edx, 4
45	00000047	83 CA 03		or edx, 3h
46	0000004A	66 26: C7 07 0733		mov word ptr es:[edi], 0733h
47	00000050	E9 000000E7		jmp NxtNum
48	00000055	3C 05	Not3:	cmp al, 05h
49	00000057	75 11		jne Not4
50	00000059	C1 E2 04		shl edx, 4
51	0000005C	83 CA 04		or edx, 4h
52	0000005F	66 26: C7 07 0734		mov word ptr es:[edi], 0734h
53	00000065	E9 000000D2		jmp NxtNum
54	0000006A	3C 06	Not4:	cmp al, 06h
55	0000006C	75 11		jne Not5
56	0000006E	C1 E2 04		shl edx, 4
57	00000071	83 CA 05		or edx, 5h
58	00000074	66 26: C7 07 0735		mov word ptr es:[edi], 0735h
59	0000007A	E9 000000BD		jmp NxtNum
60	0000007F	3C 07	Not5:	cmp al, 07h
61	00000081	75 11		jne Not6
62	00000083	C1 E2 04		shl edx, 4
63	00000086	83 CA 06		or edx, 6h
64	00000089	66 26: C7 07 0736		mov word ptr es:[edi], 0736h
65	0000008F	E9 000000A8		jmp NxtNum
66	00000094	3C 08	Not6:	cmp al, 08h
67	00000096	75 11		jne Not7
68	00000098	C1 E2 04		shl edx, 4
69	0000009B	83 CA 07		or edx, 7h

70	0000009E	66 26: C7 07 0737		mov word ptr es:[edi], 0737h
71	000000A4	E9 00000093		jmp NxtNum
72	000000A9	3C 09	Not7:	cmp al, 09h
73	000000AB	75 11		jne Not8
74	000000AD	C1 E2 04		shl edx, 4
75	000000B0	83 CA 08		or edx, 8h
76	000000B3	66 26: C7 07 0738		mov word ptr es:[edi], 0738h
77	000000B9	E9 0000007E		jmp NxtNum
78	000000BE	3C 0A	Not8:	cmp al, 0Ah
79	000000C0	75 0E		jne Not9
80	000000C2	C1 E2 04		shl edx, 4
81	000000C5	83 CA 09		or edx, 9h
82	000000C8	66 26: C7 07 0739		mov word ptr es:[edi], 0739h
83	000000CE	EB 6C		jmp NxtNum
84	000000D0	3C 1E	Not9:	cmp al, 1Eh
85	000000D2	75 0E		jne NotA
86	000000D4	C1 E2 04		shl edx, 4
87	000000D7	83 CA 0A		or edx, 0Ah
88	000000DA	66 26: C7 07 0741		mov word ptr es:[edi], 0741h
89	000000E0	EB 5A		jmp NxtNum
90	000000E2	3C 30	NotA:	cmp al, 30h
91	000000E4	75 0E		jne NotB
92	000000E6	C1 E2 04		shl edx, 4
93	000000E9	83 CA 0B		or edx, 0Bh
94	000000EC	66 26: C7 07 0742		mov word ptr es:[edi], 0742h
95	000000F2	EB 48		jmp NxtNum
96	000000F4	3C 2E	NotB:	cmp al, 2Eh
97	000000F6	75 0E		jne NotC
98	000000F8	C1 E2 04		shl edx, 4
99	000000FB	83 CA 0C		or edx, 0Ch
100	000000FE	66 26: C7 07 0743		mov word ptr es:[edi], 0743h
101	00000104	EB 36		jmp NxtNum
102	00000106	3C 20	NotC:	cmp al, 20h
103	00000108	75 0E		jne NotD
104	0000010A	C1 E2 04		shl edx, 4
105	0000010D	83 CA 0D		or edx, 0Dh

106	00000110	66 26: C7 07 0744	mov word ptr es:[edi], 0744h
107	00000116	EB 24	jmp NxtNum
108	00000118	3C 12	NotD: cmp al, 12h
109	0000011A	75 0E	jne NotE
110	0000011C	C1 E2 04	shl edx, 4
111	0000011F	83 CA 0E	or edx, 0Eh
112	00000122	66 26: C7 07 0745	mov word ptr es:[edi], 0745h
113	00000128	EB 12	jmp NxtNum
114	0000012A	3C 21	NotE: cmp al, 21h
115	0000012C	75 11	jne NotF
116	0000012E	C1 E2 04	shl edx, 4
117	00000131	83 CA 0F	or edx, 0Fh
118	00000134	66 26: C7 07 0746	mov word ptr es:[edi], 0746h
119	0000013A	EB 00	jmp NxtNum
120	0000013C	47	NxtNum: inc edi
121	0000013D	47	inc edi
122	0000013E	49	dec ecx
123			
124	0000013F	E4 61	NotF: in al, 61h
125	00000141	0C 80	or al, 80h
126	00000143	E6 61	out 61h, al
127			
128	00000145	B0 20	mov al, 20h
129	00000147	E6 20	out 20h, al
130	00000149	07	pop es
131	0000014A	58	pop eax
132	0000014B	CF	iretd
133	0000014C		IDT_Seg ends
134			
135	0000		CSeg16 segment use16
136			assume CS:CSeg16, DS:CSeg32, S
S:SSeg			
137			
138	0000	68 0000s	start: push CSeg32
139	0003	1F	pop ds
140			

141	0004	E4 92	in al, 92h
142	0006	0C 02	or al, 2
143	0008	E6 92	out 92h, al
144			
145	000A	66 33 C0	xor eax, eax
146	000D	8C C8	mov ax, cs
147	000F	66 C1 E0 04	shl eax, 4
148	0013	67 A3 0000000Ar	mov word ptr CS16Dsc+2, ax
149	0019	66 C1 E8 10	shr eax, 16
150	001D	67 A2 0000000Cr	mov byte ptr CS16Dsc+4, al
151	0023	B8 0000s	mov ax, CSeg32
152	0026	66 C1 E0 04	shl eax, 4
153	002A	66 50	push eax
154	002C	67 A3 00000012r	mov word ptr CS32Dsc+2, ax
155	0032	66 C1 E8 10	shr eax, 16
156	0036	67 A2 00000014r	mov byte ptr CS32Dsc+4, al
157	003C	B8 0000s	mov ax, DSeg32
158	003F	66 C1 E0 04	shl eax, 4
159	0043	67 A3 0000001Ar	mov word ptr DS32Dsc+2, ax
160	0049	66 C1 E8 10	shr eax, 16
161	004D	67 A2 0000001Cr	mov byte ptr DS32Dsc+4, al
162	0053	B8 0000s	mov ax, SSeg
163	0056	66 C1 E0 04	shl eax, 4
164	005A	67 A3 00000022r	mov word ptr SS32Dsc+2, ax
165	0060	66 C1 E8 10	shr eax, 16
166	0064	67 A2 00000024r	mov byte ptr SS32Dsc+4, al
167	006A	B8 0000s	mov ax, IDT_Seg
168	006D	66 C1 E0 04	shl eax, 4
169	0071	67 A3 00000032r	mov word ptr IDT_Dsc+2, ax
170	0077	66 C1 E8 10	shr eax, 16
171	007B	67 A2 00000034r	mov byte ptr IDT_Dsc+4, al
172	0081	B8 0000s	mov ax, CSegPr
173	0084	66 C1 E0 04	shl eax, 4
174	0088	67 A3 0000003Ar	mov word ptr CSPrDsc+2, ax
175	008E	66 C1 E8 10	shr eax, 16
176	0092	67 A2 0000003Cr	mov byte ptr CSPrDsc+4, al

177				
178				
179				
180	0098	66 58		pop eax
181	009A	66 50		push eax
182	009C	66 05 00000000r		add eax, offset GDT
183	00A2	66 67 A3 00000042r		mov dword ptr gdtr+2, eax
184				
185	00A9	67 0F 01 15	+	lgdt fword ptr gdtr
186		00000040r		
187				
188				
189	00B1	66 58		pop eax
190	00B3	66 05 00000046r		add eax, offset IDT
191	00B9	66 67 A3 00000848r		mov dword ptr idtr+2, eax
192				
193	00C0	67 0F 01 1D	+	lidt fword ptr idtr
194		00000846r		
195				
196	00C8	FA		cli
197				
198				
199	00C9	0F 20 C0		mov eax, cr0
200	00CC	0C 01		or al, 1
201	00CE	0F 22 C0		mov cr0, eax
202				
203	00D1	66		db 66h
204	00D2	EA		db 0EAh
205	00D3	000008EDr		dd offset PMentry
206	00D7	0010		dw CS32Sel
207				
208	00D9	0F 20 C0	RMret:	mov eax, cr0
209	00DC	24 FE		and al, 0FEh
210	00DE	0F 22 C0		mov cr0, eax
211	00E1	EA		db 0EAh
212	00E2	00E6r		dw \$+4


```

213      00E4  0000s                                dw CSeg16
214
215      00E6  BA 0000s                                mov dx, SSeg
216      00E9  8E D2                                mov ss, dx
217      00EB  BC 0100                                mov sp, SSize
218
219      00EE  B8 0000s                                mov ax, CSeg32
220      00F1  8E D8                                mov ds, ax
221      00F3  67| 0F 01 1D      +                    lidt fword ptr idtr_real
222              0000084Cr
223
224      00FB  FB                                    sti
225
226      00FC  B4 00                                mov ah, 0
227      00FE  CD 16                                int 16h
228
229      0100  B4 4C                                mov ah, 4Ch
230      0102  CD 21                                int 21h
231      0104                                CSeg16 ends
232
233 00000000                                CSegPr segment use32
234                                assume cs:CSegPr, ds:DSeg32
235 00000000                                outnum proc
236 00000000 50                                push eax
237 00000001 53                                push ebx
238 00000002 51                                push ecx
239 00000003 AD                                lodsd
240 00000004 B9 00000008                        mov ecx, 8
241 00000009 8B D0                                outloop: mov edx, eax
242 0000000B 81 E2 F0000000                        and edx, 0F000000h
243 00000011 C1 EA 14                                shr edx, 20
244 00000014 80 FE 0A                                cmp dh, 0Ah
245 00000017 72 03                                jc next
246 00000019 80 C6 07                                add dh, 07h
247 0000001C 80 C6 30                                next:add dh, 30h
248 0000001F 26: 88 37                                mov es:[edi], dh

```

```

249 00000022 83 C7 02          add edi, 2
250 00000025 C1 E0 04          shl eax, 4
251 00000028 E2 DF          loop outloop
252 0000002A 66| 83 C7 02      add di, 2
253 0000002E 59          pop ecx
254 0000002F 5B          pop ebx
255 00000030 58          pop eax
256 00000031 CB          db 0CBh
257 00000032          outnum endp
258 00000032      CSegPr ends
259
260 00000000          CSeg32 segment use32
261          assume cs:CSeg32, ds:DSeg32
262
263 00000000          GDT label byte
264 00000000 08*(00)          db 8 dup(0)
265 00000008 FF FF 00 00 00 9A 00+ CS16Dsc db 0FFh,0FFh,0,0,0,10011010b,0,
0
266          00
267 00000010 FF FF 00 00 00 9A CF+ CS32Dsc db 0FFh,0FFh,0,0,0,10011010b,11
001111b,0
268          00
269 00000018 FF FF 00 00 00 92 CF+ DS32Dsc db 0FFh,0FFh,0,0,0,10010010b,11
001111b,0
270          00
271 00000020 FF FF 00 00 00 92 CF+ SS32Dsc db 0FFh,0FFh,0,0,0,10010010b,11
001111b,0
272          00
273 00000028 FF FF 00 80 0B 92 CF+ VSegDsc db 0FFh,0FFh,0,80h,0Bh,10010010
b,11001111b,0
274          00
275 00000030 FF FF 00 00 00 9A CF+ IDT_Dsc db 0FFh,0FFh,0,0,0,10011010b,11
001111b,0
276          00
277 00000038 FF FF 00 00 00 9A CF+ CSPrDsc db 0FFh,0FFh,0,0,0,10011010b,11
001111b,0

```

```

278          00
279          =0040          GDT_1 = $-GDT
280
281 00000040 003F          gdtr    dw GDT_1-1
282 00000042 ????????          dd ?
283          =0008          CS16Sel equ 0000000000001000b
284          =0010          CS32Sel equ 00000000000010000b
285          =0018          DS32Sel equ 00000000000011000b
286          =0020          SS32Sel equ 00000000000010000b
287          =0028          VSegSel equ 000000000000101000b
288          =0030          IDT_Sel equ 000000000000110000b
289          =0038          CSPrSel equ 000000000000111000b
290
291 00000046          IDT      label byte
292
293          ;INT 00 - 07
294 00000046 08*(0852r 0010 8E00 +          dw 8 dup(small offset int_handl
er,CS32Sel,8E00h,0)
295          0000)
296          ;INT 08 (irq0)
297 00000086 0853r 0010 8E00 0000          dw small offset irq0_7_handler,
CS32Sel,8E00h,0
298          ;INT 09 (irq1)
299 0000008E 0000r 0030 8E00 0000          dw small offset irq1_handler,ID
T_Sel,8E00h,0
300          ;INT 0Ah - 0Fh (IRQ2 - IRQ8)
301 00000096 06*(0853r 0010 8E00 +          dw 6 dup(small offset irq0_7_ha
ndler,CS32Sel,8E00h,0)
302          0000)
303          ;INT 10h - 6Fh
304 000000C6 61*(0852r 0010 8E00 +          dw 97 dup(small offset int_hand
ler,CS32Sel,8E00h,0)
305          0000)
306          ;INT 70h - 78h (IRQ8 - IRQ15)
307 000003CE 08*(085Ar 0010 8E00 +          dw 8 dup(small offset irq8_15_h
andler,CS32Sel,8E00h,0)

```

```

308          0000)
309                                     ;INT 79h - FFh
310 0000040E 87*(0852r 0010 8E00 +      dw 135 dup(small offset int_han
dler,CS32Sel,8E00h,0)
311          0000)
312          =0800                      idt_size = $-IDT
313
314 00000846 07FF                      idtr    dw idt_size-1
315 00000848 ????????                  dd ?
316
317 0000084C 03FF 0000 0000          idtr_real dw 3FFh,0,0
318
319 00000852                          int_handler:
320 00000852 CF                          iretd
321
322 00000853                          irq0_7_handler:
323 00000853 50                          push eax
324 00000854 B0 20                      mov al, 20h
325 00000856 E6 20                      out 20h, al
326 00000858 58                          pop eax
327 00000859 CF                          iretd
328
329 0000085A                          irq8_15_handler:
330 0000085A 50                          push eax
331 0000085B B0 20                      mov al, 20h
332 0000085D E6 A1                      out 0A1h, al
333 0000085F 58                          pop eax
334 00000860 CF                          iretd
335
336 00000861                          inpp    proc
337 00000861 52                          push edx
338 00000862 51                          push ecx
339 00000863 50                          push eax
340 00000864 33 D2                      xor edx, edx
341 00000866 33 C9                      xor ecx, ecx
342 00000868 B9 00000004                mov ecx, 4h

```

343	0000086D	83 F9 00	inn: cmp ecx, 0
344	00000870	75 FB	jne inn
345	00000872	89 16	mov ds:[esi], edx
346	00000874	83 C6 04	add esi, 4
347	00000877	83 C7 04	add edi, 4
348	0000087A	58	pop eax
349	0000087B	59	pop ecx
350	0000087C	5A	pop edx
351	0000087D	C3	ret
352	0000087E		inpp endp
353			
354	0000087E		mainproc proc
355	0000087E	8B EC	mov ebp, esp
356	00000880	33 D2	xor edx, edx
357	00000882	8B 55 04	mov edx, [ebp+4]
358	00000885	66 8B C2	mov ax, dx
359	00000888	66 BB 0000	mov bx, 0b
360	0000088C	66 C7 05 00000020r +	mov bit_mask, 110000000000000b
361		00C0	
362	00000895	66 C7 05 00000022r +	mov errors_count, 0b
363		0000	
364	0000089E	66 8B 45 04	c1: mov ax, [ebp+4]
365	000008A2	66 23 05 00000020r	and ax, bit_mask
366	000008A9	66 3B 05 00000020r	cmp ax, bit_mask
367	000008B0	74 08	je add_1
368	000008B2	66 3D 0000	cmp ax, 0
369	000008B6	74 0B	je add_0
370	000008B8	EB 0E	jmp add_e
371	000008BA	66 D1 E3	add_1: shl bx, 1
372	000008BD	66 83 C3 01	add bx, 1b
373	000008C1	EB 0F	jmp end_if
374	000008C3	66 D1 E3	add_0: shl bx, 1
375	000008C6	EB 0A	jmp end_if
376	000008C8	66 FF 05 00000022r	add_e: inc errors_count
377	000008CF	66 D1 E3	shl bx, 1
378	000008D2	66 C1 2D 00000020r +	end_if: shr bit_mask, 2

```

379          02
380 000008DA 66| 83 3D 00000020r +      cmp bit_mask, 0
381          00
382 000008E2 75 BA                      jne c1
383 000008E4 66| 8B C3                    mov ax, bx
384 000008E7 89 45 04                    mov [ebp+4], eax
385 000008EA 8B E5                      mov esp, ebp
386 000008EC C3                      ret
387 000008ED                      mainproc endp
388
389 000008ED                      PMentry:
390
391 000008ED 66| BA 0018                    mov dx, DS32Sel
392 000008F1 8E DA                      mov ds, dx
393 000008F3 66| BA 0020                    mov dx, SS32Sel
394 000008F7 8E D2                      mov ss, dx
395 000008F9 BC 00000100                mov esp, Ssize
396 000008FE 66| BA 0028                    mov dx, VSegSel
397 00000902 8E C2                      mov es, dx
398 00000904 33 FF                      xor edi, edi
399
400 00000906 B8 07200720                mov eax, 07200720h
401 0000090B B9 000003E8                mov ecx, 80*25*2/4
402 00000910 F3> AB                      rep stosd
403 00000912 33 FF                      xor edi, edi
404
405 00000914 FB                      sti
406
407 00000915 BE 00000000r          lea esi, X
408 0000091A B9 00000004                mov ecx, 4
409 0000091F 33 D2                      xor edx, edx
410 00000921 E8 FFFFFFF3B          ixloop: call inpp
411 00000926 E2 F9                      loop ixloop
412
413 00000928 66| BA 0018                    mov dx, DS32Sel
414 0000092C 8E C2                      mov es, dx

```

```

415
416 0000092E BE 00000000r      lea esi, x
417 00000933 BF 00000010r      lea edi, y
418 00000938 B9 00000004      mov ecx, 4
419 0000093D AD      mloop: lodsd
420 0000093E 50      push eax
421 0000093F E8 FFFFFFF3A      call mainproc
422 00000944 58      pop eax
423 00000945 AB      stosd
424 00000946 E2 F5      loop    mloop
425
426 00000948 66 BA 0028      mov dx,VSegSel
427 0000094C 8E C2      mov es,dx
428 0000094E 33 FF      xor edi, edi
429
430 00000950 BE 00000010r      lea esi, y
431 00000955 33 FF      xor edi, edi
432 00000957 BF 00000140      mov edi, 80*4
433 0000095C BB 80000000      mov ebx, 80000000h
434 00000961 B9 00000004      mov ecx, 4
435 00000966      obloop: ;call outnum
436 00000966 9A      db 9Ah
437 00000967 00000000r      dd offset outnum
438 0000096B 0038      dw CSPrSel
439 0000096D E2 F7      oloop:loop obloop
440
441 0000096F EA      db 0EAh
442 00000970 000000D9r      dd offset RMret
443 00000974 0008      dw CS16Sel
444 00000976      CSeg32 ends
445      end start

```

2. Результаты исследования

Процесс выполнения кода реального режима приведён в таблице 1.

Таблица 1 - Выполнение команд кода реального режима

Номер команды	Адрес команд ы	Команда на машинном языке	Команда на языке ассемблер	Содержание изменившихся регистров, ячеек памяти и портов ввода/вывода	Значения флагов процессора
1	0000	685731	push 3157	ip = 0003 ir = 68 sp = 0100 ss[0100] = 3157	
2	0003	1F	pop ds	ip = 0004 ir = 1F ds = 3157 sp = 0102	
3	0004	E492	in al, 92	ip = 0006 ir = E4 ax = 0002	
4	0006	0C02	or al, 02	ip = 0008 ir = 0C	cf = 0 zf = 0 sf = 0 of = 0 pf = 0 af = 0
5	0008	E692	out 92, al	ip = 000A ir = E6 I/O Port 92h = 02	
6	000A	6633C0	xor eax, eax	ip = 000D ir = 6633C0 eax = 00000000	cf = 0 zf = 1 sf = 0 of = 0

					pf = 1 af = 0
7	000D	8CC8	mov ax, cs	ip = 000F ir = 8CC8 ax = 3142	
8	000F	66C1E004	shl eax, 04	ip = 0013 ir = 66C1E0 eax = 00031420	cf = 0 zf = 0 sf = 0 of = 0 pf = 0 af = 0
9	0013	67A30A000000	mov [0000000A], ax	ip = 0019 ir = 67A3 ds [0000000A] = 00031420	
10	0019	66C1E810	shr eax, 10	ip = 001D ir = 66C1E8 eax = 00000003	cf = 0 zf = 0 sf = 0 of = 0 pf = 1 af = 0
11	001D	67A20C000000	mov [0000000C], al	ip = 0023 ir = 67A2 ds [0000000C] = 03	
12	0023	B85731	mov ax, 3157	ip = 0026 ir = B8 ax = 3157	
13	0026	66C1E004	shl eax, 04	ip = 002A ir = 66C1E0 eax = 00031570	cf = 0 zf = 0 sf = 0 of = 0 pf = 0 af = 0

14	002A	6650	push eax	ip = 002C ir = 6650 sp = 00FE ss [0100] = 0003 ss [00FE] = 1570	
15	002C	67A312000000	mov [00000012], ax	ip = 0032 ir = 67A3 ds [00000012] = 00031570	
16	0032	66C1E810	shr eax, 10	ip = 0036 ir = 66C1E8 eax = 00000003	cf = 0 zf = 0 sf = 0 of = 0 pf = 1 af = 0
17	0036	67A214000000	mov [00000014], al	ip = 003F ir = 67A2 ds [00000014] = 03	
18	003C	B82A31	mov ax, 312A	ip = 003F ir = B8 ax = 312A	
19	003F	66C1E004	shl eax, 04	ip = 0043 ir = 66C1E0 eax = 000312A0	cf = 0 zf = 0 sf = 0 of = 0 pf = 1 af = 0
20	0043	67A31A000000	mov [0000001A], ax	ip = 0049 ir = 6731 ds [0000001A] = 000312A0	
21	0049	66C1E810	shl eax, 10	ip = 004D ir = 66C1E8 eax = 00000003	cf = 0 zf = 0 sf = 0

					of = 0 pf = 1 af = 0
22	004D	67A21C000000	mov [0000001C], al	ip = 0053 ir = 67A2 ds [0000001C] = 03	
23	0053	B81A31	mov ax, 311A	ip = 0056 ir = B8 ax = 311A	
24	0056	66C1E004	shl eax, 04	ip = 005A ir = 66C1E0 eax = 000311A0	cf = 0 zf = 0 sf = 0 of = 0 pf = 1 af = 0
25	005A	67A322000000	mov [00000022], ax	ip = 0060 ir = 67A3 ds [00000022] = 000311A0	
26	0060	66C1E810	shr eax, 10	ip = 0064 ir = 66C1E8 eax = 00000003	cf = 0 zf = 0 sf = 0 of = 0 pf = 1 af = 0
27	0064	67A224000000	mov [00000024], al	ip = 006A ir = 67A2 ds [00000024] = 03	
28	006A	B82D31	mov ax, 312D	ip = 006D ir = B8 ax = 312D	
29	006D	66C1E004	shl eax, 04	ip = 0071 ir = 66C1E0	cf = 0 zf = 0

				eax = 000312D0	sf = 0 of = 0 pf = 1 af = 0
30	0071	67A332000000	mov [00000032], ax	ip = 0077 ir = 67A3 ds [00000032] = 000312D0	
31	0077	66C1E810	shr eax, 10	ip = 007B ir = 66C1E8 eax = 00000003	cf = 0 zf = 0 sf = 0 of = 0 pf = 1 af = 0
32	007B	67A234000000	mov [00000034], al	ip = 0081 ir = 67A2 ds [00000034] = 03	
33	0081	B85331	mov ax, 3153	ip = 0084 ir = B8 ax = 3153	
34	0084	66C1E004	shl eax, 04	ip = 0088 ir = 66C1E0 eax = 00031530	cf = 0 zf = 0 sf = 0 of = 0 pf = 1 af = 0
35	0088	67A33A000000	mov [0000003A], ax	ip = 008E ir = 67A3 ds [0000003A] = 00031530	
36	008E	66C1E810	shr eax, 10	ip = 0092 ir = 66C1E8 eax = 00000003	cf = 0 zf = 0 sf = 0 of = 0

					pf = 1 af = 0
37	0092	67A23C000000	mov [0000003C], al	ip = 0098 ir = 67A2 ds [0000003C] = 03	
38	0098	6658	pop eax	ip = 009A ir = 6658 sp = 0102	
39	009A	6650	push eax	ip = 009C ir = 6650 sp = 00FE ss [0100] = 0003 ss [00FE] = 1570	
40	009C	660500000000	add eax, 00000000	ip = 00A2 ir = 6605 eax = 00031570	cf = 0 zf = 0 sf = 0 of = 0 pf = 0 af = 0
41	00A2	6667A342000000	mov [00000042], eax	ip = 00A9 ir = 6667A3 ds [00000042] = 00031570	
42	00A9	670F011540000000	lgdt [00000040]	ip = 00B1 ir = 670F0115 GDTR = 00031570003F	
43	00B1	6658	pop eax	ip = 00B3 ir = 6658 sp = 0102	
44	00B3	660564000000	add eax, 00000046	ip = 00B9 ir = 6605 eax = 000315B6	

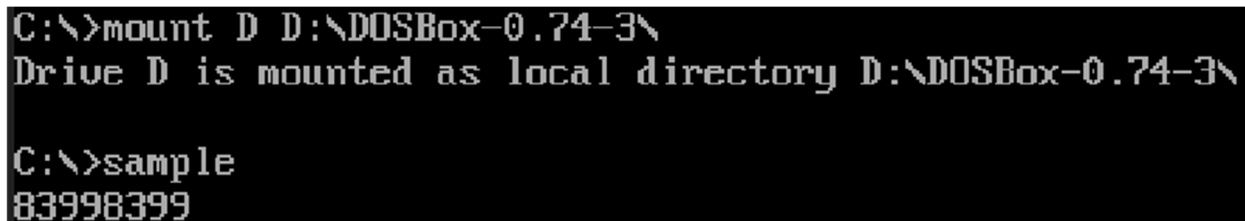
45	00B9	6667A348080000	mov [00000848], eax	ip = 00C0 ir = 6667A3 ds [00000848] = 000315B6	
46	00C0	670F011D46080000	lidt [00000846]	ip = 00C8 ir = 670F011D IDTR = 000315B607FF	
47	00C8	FA	cli	ip = 00C9 ir = FA i = 0	
48	00C9	0F20C0	mov eax, cr0	ip = 00CC ir = 0F20C0	
49	00CC	0C01	or al, 01	ip = 00CE ir = 0C	
50	00CE	0F22C0	mov cr0, eax	ip = 00D1 ir = 0F22C0 Bit PE = 1	
51	00D1	66EAB70800001000	jmp 0010: 000008B7	ip = 00D9 ir = 66EA	

Работа программы в защищённом режиме

52	00D9	0F20C0	mov eax, cr0	ip = 00DC ir = 0F20C0	
53	00DC	24FE	and al, FE	ip = 00DE ir = 24	
54	00DE	0F22C0	mov cr0, eax	ip = 00E1 ir = 0F22C0 Bit PE = 0	
55	00E1	EAE6004231	jmp 3142:00E6	ip = 00E6 ir = EA	
56	00E6	BA1A31	mov dx, 311A	ip = 00E9 ir = BA dx = 311A	
57	00E9	8EDA	mov ss, dx	ip = 00EB	

				ir = 8EDA ss = 311A	
58	00EB	BC0001	mov sp, 0100	ip = 00EE ir = BC sp = 0100	
59	00EE	B85731	mov ax, 3157	ip = 00F1 ir = B8 ax = 3157	
60	00F1	8ED8	mov ds, ax	ip = 00F3 ir = 8ED8 ds = 3157	
61	00F3	670F011D4C080000	lidt [0000084C]	ip = 00FB ir = 670F11 IDTR = 0000315703FF	
62	00FB	FB	sti	ip = 00FC ir = FB i = 1	
63	00FC	B400	mov ah, 00	ip = 00FE ir = B4 ax = 0000	
64	00FE	CD16	int 16	ip = 0100 ir = CD	
65	0100	B44C	mov ah, 4C	ip = 0102 ir = B4 ax = 4C00	
66	0102	CD21	int 21	ip = 0104 ir = CD	
67	0104				

Пример работы программы приведён на рисунке 1.



```
C:\>mount D D:\DOSBox-0.74-3\  
Drive D is mounted as local directory D:\DOSBox-0.74-3\  
  
C:\>sample  
83998399
```

Рисунок 1 – Ввод входных значений “8399” с клавиатуры и вывод результата “8399” на дисплей в консоли Turbo Debugger (sample – название .asm файла)

Значения полей базы дескрипторов:

1. CSeg16Dsc: 00031420
2. CSeg32Dsc: 00031570
3. DSeg32Dsc: 000312A0
4. SSegDsc: 000311A0
5. IDT_Dsc: 000312D0
6. CSegPrDsc: 00031530

Значения регистров GDTR и IDTR:

- GDTR: 00031570003F
- IDTR: 000315B607FF

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:

1. Контроллер клавиатуры распознает нажатие клавиши и помещение кода в порт 60h.
2. Нажатие клавиши вызывает сигнал аппаратного прерывания.

3. Процессор получает номер прерывания (индекс соответствующего дескриптора в IDT).
4. Процессор читает из шлюза прерывания селектор сегмента кода, в котором находится обработчик прерывания, и смещение по которому находится обработчик.
5. Извлечение базы сегмента по селектору, получен полный логический адрес обработчика.
6. Выполнение первой команды, адрес которой соответствует адресу обработчика.

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.

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 16-битные 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

В ходе выполнения данной лабораторной работы мной была реализована программа, выполняющая преобразование числа в упакованный двоично-десятичный код. Исходные данные вводятся в память с клавиатуры, результаты выводятся на дисплей в защищенном режиме. Вывод результатов на экран выполняет дальняя подпрограмма.