Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчёт

по лабораторной работе № 2

Название: Защищенный режим

Дисциплина: Операционные системы

Студент ИУ7-55Б _____ Д.В. Сусликов

(Группа) (Подпись, дата) (И.О. Фамилия)

Преподаватель _____ Н.Ю. Рязанова

(Подпись, дата) (И.О. Фамилия)

Москва, 2020

Содержание

1 Листинг программы

Ниже показан Листинг программы.

```
.586p

seg_descr struc
lim   dw 0
base_l  dw 0
base_m  db 0
attr_1  db 0
attr_2  db 0
base_h  db 0
seg_descr ends


int_descr struc
offs_l  dw 0
sel   dw 0
cntr    db 0
attr  db 0
offs_h  dw 0
int_descr ends


seg_stack segment  para stack 'STACK'
stack_start db  256 dup(?)
stack_size = $-stack_start
seg_stack   ENDS

seg_data segment para 'DATA'
gdt_null  seg_descr <>

gdt_flatDS  seg_descr <0FFFFh, 0, 0, 10010010b, 10001111b, 0>

gdt_16bitCS seg_descr <seg_rm_size-1, 0, 0, 10011000b, 00000000b
```

```
    , 0>

gdt_32bitCS seg_descr <seg_pm_size-1, 0, 0, 10011000b, 01000000b
    , 0>

gdt_32bitDS seg_descr <data_size-1, 0, 0, 10010010b, 01000000b,
   0>

gdt_32bitSS seg_descr <stack_size-1, 0, 0, 10010110b, 01000000b,
    0>

gdt_PM_videobuffer_32bit seg_descr <4095, 8000h, 0Bh, 10010010b,
    01000000b, 0>

gdt_size = $-gdt_null

gdtr  df 0

sel_flatDS      equ    8
sel_16bitCS     equ   16
sel_32bitCS     equ   24
sel_32bitDS     equ   32
sel_32bitSS     equ   40
sel_VideoBuf    equ   48

IDT label byte

int_descr 13 dup (<0, sel_32bitCS, 0, 10001111b, 0>)
trap_13 int_descr <0, sel_32bitCS, 0, 10001111b, 0>
int_descr 18 dup (<0, sel_32bitCS, 0, 10001111b, 0>)

int08 int_descr <0, sel_32bitCS, 0, 10001110b, 0>

int09 int_descr <0, sel_32bitCS, 0, 10001110b, 0>

```

```
63      idt_size = $-IDT
64
65      idtr    df 0
66
67      idtr_real dw  3FFh, 0, 0
68
69      master  db 0
70      slave db 0
71
72      mark_08 dw 0
73      time_08 dd 0
74      screen_pos  dd 2 * 80
75
76      ascii db 0, 27
77      db '1234567890-+', 8
78      db 9, 'QWERTYUIOP[]', 0
79      db 0,   'ASDFGHJKL', 59, 39, 96, 0
80      db      '\ZXCVBNM,./', 0
81      db 0, 0, 0, 0, ' ', 0, 0
82
83      msg_rm    db 'Real Mode', 10, '$'
84      msg_pm db 'Protected Mode', 10, '$'
85      msg_skip db 10,10,10, '$'
86
87      data_size = $-gdt_null
88      seg_data ends
89
90      seg_pm segment para public 'CODE' use32
91      assume cs:seg_pm, ds:seg_data, ss:seg_stack
92      pm_start:
93      mov ax, sel_32bitDS
94      mov ds, ax
95      mov ax, sel_VideoBuf
96      mov es, ax
97      mov ax, sel_32bitSS
```

```
98      mov ss, ax
99      mov eax, stack_size
100     mov esp, eax
101
102     sti
103     call calc_memory_amount
104
105     start_08:
106     test mark_08, 1
107     jz start_08
108
109
110     cli
111
112     db  0EAh
113     dd  offset real_mode_return
114     dw  sel_16bitCS
115
116     new_int08 proc uses eax
117     mov  eax, time_08
118
119     call timing
120
121     inc eax
122     mov time_08, eax
123
124     mov al, 20h
125     out 20h, al
126
127     iretd
128     new_int08 endp
129
130     new_int09 proc uses eax ebx edx
131     in  al, 60h
132
```

```
133     cmp al, 1Ch
134     jne print_key
135     or mark_08, 1
136     jmp leave_stop
137
138     print_key:
139     cmp al, 80h
140     ja leave_stop
141
142     xor ah, ah
143     xor ebx, ebx
144     mov bx, ax
145
146     mov dl, ascii[ebx]
147     mov ebx, screen_pos
148     mov es:[ebx], dl
149
150     add ebx, 2
151     mov screen_pos, ebx
152
153     leave_stop:
154     in  al, 61h
155     or  al, 80h
156     out 61h, al
157     and al, 7Fh
158     out 61h, al
159
160     mov al, 20h
161     out 20h, al
162
163     iretd
164     new_int09 endp
165
166     plug_trap13:
167     pop eax
```

```
168     iretd
169
170
171     to_ascii proc
172     cmp dl, 10
173     jl number
174     add dl, 'A' - '0' - 10
175     number:
176     add dl, '0'
177     ret
178     to_ascii endp
179
180     print_by_eax proc uses ecx ebx edx
181     add ebx, 10h
182     mov ecx, 8
183     print_eax:
184     mov dl, al
185     and dl, 0Fh
186     call to_ascii
187     mov es:[ebx], dl
188     ror eax, 4
189     sub ebx, 2
190     loop print_eax
191     ret
192     print_by_eax endp
193
194     calc_memory_amount proc uses ds eax ebx
195     mov ax, sel_flatDS
196     mov ds, ax
197     mov ebx, 100001h
198     mov dl, 12
199     mov ecx, 0FFEFFFFEh
200
201     amount:
202     mov dh, ds:[ebx]
```

```
203     mov ds:[ebx], dl
204     cmp ds:[ebx], dl
205     jnz mem_end
206
207     mov ds:[ebx], dh
208     inc ebx
209     loop amount
210
211     mem_end:
212     mov eax, ebx
213     xor edx, edx
214
215     mov ebx, 100000h
216     div ebx
217
218     mov ebx, 0
219     call print_by_eax
220
221     mov ebx, 20
222     mov al, 'M'
223     mov es:[ebx], al
224
225     mov ebx, 22
226     mov al, 'B'
227     mov es:[ebx], al
228     ret
229     calc_memory_amount endp
230
231     print_time proc uses eax ebx ecx edx
232     mov al, dl
233     xor ecx, ecx
234
235     add ebx, 4
236     mov cx, 2
237
```

```
238     mov dl, 10
239     print_numeral:
240     xor ah, ah
241     div dl
242     add ah, '0'
243     mov es:[ebx], ah
244     sub ebx, 2
245     loop print_numeral
246     ret
247     print_time endp
248
249     timing proc uses eax ebx ecx edx
250     mov ecx, 65536
251     xor edx, edx
252     mul ecx
253     mov ecx, 1193180
254     div ecx
255
256     xor edx, edx
257     mov ecx, 60
258     div ecx
259
260     mov ebx, 80
261     call print_time
262
263     mov dh, ':'
264     mov es:[ebx], dh
265     sub ebx, 6
266
267     xor edx, edx
268     div ecx
269     call print_time
270
271     mov dh, ':'
272     mov es:[ebx], dh
```

```
273     sub ebx, 6
274     xor dh, dh
275
276     mov dl, al
277     call print_time
278
279     ret
280     timing endp
281
282     seg_pm_size = $-pm_start
283     seg_pm ends
284
285     seg_rm segment para public 'CODE' use16
286     assume cs:seg_rm, ds:seg_data, ss: seg_stack
287
288     start:
289     mov ax, seg_data
290     mov ds, ax
291
292     mov ax, seg_pm
293     mov es, ax
294
295     mov ah, 09h
296     lea dx, msg_rm
297     int 21h
298
299     mov ah, 09h
300     lea dx, msg_pm
301     int 21h
302
303     push eax
304     mov ah, 10h
305     int 16h
306     pop eax
307
```

```
308     xor eax, eax
309     mov ax, seg_rm
310     shl eax, 4
311     mov word ptr gdt_16bitCS.base_l, ax
312     shr eax, 16
313     mov byte ptr gdt_16bitCS.base_m, al
314     mov byte ptr gdt_16bitCS.base_h, ah
315
316
317     mov ax, seg_pm
318     shl eax, 4
319     mov word ptr gdt_32bitCS.base_l, ax
320     shr eax, 16
321     mov byte ptr gdt_32bitCS.base_m, al
322     mov byte ptr gdt_32bitCS.base_h, ah
323
324
325     mov ax, seg_data
326     shl eax, 4
327     mov word ptr gdt_32bitDS.base_l, ax
328     shr eax, 16
329     mov byte ptr gdt_32bitDS.base_m, al
330     mov byte ptr gdt_32bitDS.base_h, ah
331
332
333     mov ax, seg_stack
334     shl eax, 4
335     mov word ptr gdt_32bitSS.base_l, ax
336     shr eax, 16
337     mov byte ptr gdt_32bitSS.base_m, al
338     mov byte ptr gdt_32bitSS.base_h, ah
339
340
341     mov ax, seg_data
342     shl eax, 4
```

```
343     add eax, offset gdt_null
344
345     mov dword ptr gdtr + 2, eax
346     mov word ptr  gdtr, gdt_size-1
347     lgdt fword ptr gdtr
348
349
350     lea eax, es:plug_trap13
351     mov trap_13.offs_l, ax
352     shr eax, 16
353     mov trap_13.offs_h, ax
354
355     lea eax, es:new_int08
356     mov int08.offs_l, ax
357     shr eax, 16
358     mov int08.offs_h, ax
359
360     lea eax, es:new_int09
361     mov int09.offs_l, ax
362     shr eax, 16
363     mov int09.offs_h, ax
364
365     mov ax, seg_data
366     shl eax, 4
367     add eax, offset IDT
368
369
370     mov  dword ptr idtr + 2, eax
371     mov  word ptr  idtr, idt_size-1
372
373     in  al, 21h
374     mov master, al
375     in  al, 0A1h
376     mov slave, al
377     mov al, 11h
```

```
378     out 20h, al
379
380     mov al, 32
381     out 21h, al
382     mov al, 4
383     out 21h, al
384     mov al, 1
385     out 21h, al
386
387     mov al, 0FCh
388     out 21h, al
389
390     mov al, 0FFh
391     out 0A1h, al
392
393     cli
394     in  al, 70h
395     or  al, 80h
396     out 70h, al
397
398     lidt fword ptr idtr
399
400     in  al, 92h
401     or  al, 2
402     out 92h, al
403
404     push ds
405     mov ax, 40h
406     mov ds, ax
407     mov eax, dword ptr ds:[6Ch]
408     pop ds
409     mov dword ptr time_08, eax
410
411
412     mov eax, cr0
```

```
413     or eax, 1
414     mov cr0, eax
415
416     db  66h
417     db  0EAh
418     dd  offset pm_start
419     dw  sel_32bitCS
420
421     real_mode_return:
422     mov eax, cr0
423     and al, 0FEh
424     mov cr0, eax
425
426     db  0EAh
427     dw  $+4
428     dw  seg_rm
429
430     mov eax, seg_data
431     mov ds, ax
432     mov eax, seg_pm
433     mov es, ax
434     mov ax, seg_stack
435     mov ss, ax
436     mov ax, stack_size
437     mov sp, ax
438
439     mov al, 11h
440     out 20h, al
441     mov al, 8
442     out 21h, al
443     mov al, 4
444     out 21h, al
445     mov al, 1
446     out 21h, al
447
```

```
448     mov al, master
449     out 21h, al
450     mov al, slave
451     out 0A1h, al
452
453     lidt  fword ptr idtr_real
454
455     in  al, 70h
456     and al, 7FH
457     out 70h, al
458     sti
459
460     mov ah, 09h
461     lea dx, msg_rm
462     int 21h
463
464     mov ax, 4C00h
465     int 21h
466
467     seg_rm_size = $-start
468     seg_rm  ends
469     end start
```

2 Пример работы

Рисунок 1 – Пример работы

Рисунок 2 – Пример работы

Рисунок 3 – Пример работы

Рисунок 4 – Пример работы