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»

О Т Ч Е Т

по лабораторной работе № 0 2

Название ***Защищенный режим.***

Дисциплина: ***Операционные системы***

Студент	***ИУ7И-56Б***		**Нгуен Ф. С.**
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель			**Рязанова Н. Ю.**
		(Подпись, дата)	(И.О. Фамилия)

Москва, 2020

❖ Задание:

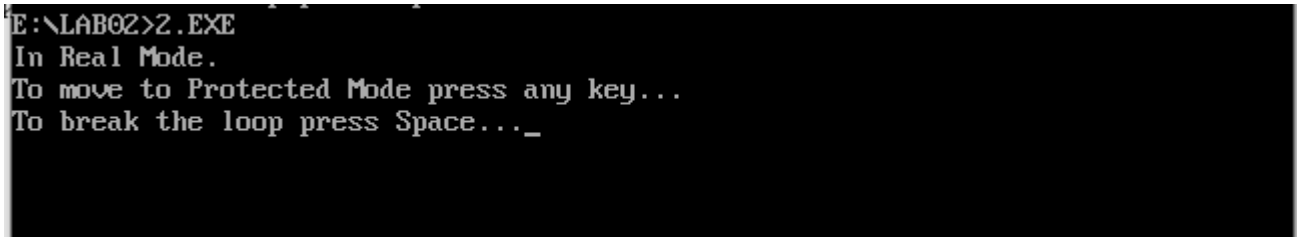
Написать программу, переводящую компьютер в защищенный режим (32-разраядный режим работы компьютеров на базе процессоров Intel).

Программа начинает работать в реальном режиме. Для перевода в защищенный режим выполняются необходимые действия. В защищенном режиме программа работает на нулевом уровне привилегий.

В защищенном режиме программа должна:

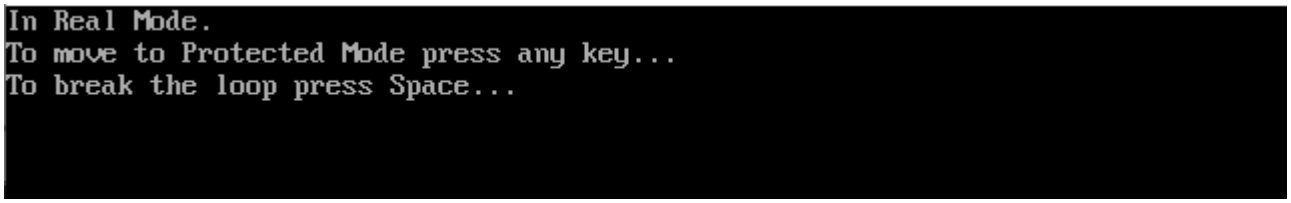
- определить объем доступной физической памяти;
- осуществить ввод с клавиатуры строки с выводом введенной строки на экран;
- получить информацию на экране от системного таймера или в виде мигающего курсора, или в виде количества тиков с момента запуска программы на выполнение, или в виде значения реального времени.

Затем программа корректно возвращается в реальный режим с соответствующими сообщениями.

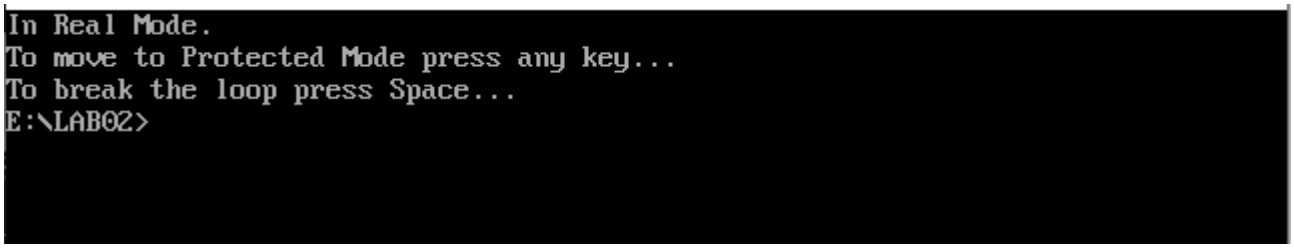


Press Key

Press Enter



Press Space



```
1.  .386p
2.
3.  descr struc
4.      lim         dw 0
5.      base_l  dw 0
6.      base_m  db 0
7.      attr_1  db 0
8.      attr_2  db 0
9.      base_h  db 0
10. descr ENDS
11.
12. int_descr struc
13.     offs_l  dw 0
14.     sel         dw 0
15.     counter db 0
16.     attr    db 0
17.     offs_h  dw 0
18. int_descr ENDS
19.
20. ; Protected mode
21. PM_seg  SEGMENT PARA PUBLIC 'DATA' USE32
22.                     ASSUME  CS:PM_seg
23.
24.     ; Таблица дескрипторов сегметов GDT
25.     GDT     label   byte
26.
27.     ; нулевой дескриптор
28.     gdt_null    descr <>
29.
30.     ; 32-битный 4-гигабайтный сегмент с базой = 0
31.     gdt_flatDS  descr <0FFFFh,0,0,92h,11001111b,0>    ; 92h = 10010010b
32.
33.     ; 16-битный 64-килобайтный сегмент кода с базой RM_seg
34.     gdt_16bitCS descr <RM_seg_size-1,0,0,98h,0,0> ; 98h = 10011010b
35.
36.     ; 32-битный 4-гигабайтный сегмент кода с базой PM_seg
37.     gdt_32bitCS descr <PM_seg_size-1,0,0,98h,01000000b,0>
38.
39.     ; 32-битный 4-гигабайтный сегмент данных с базой PM_seg
40.     gdt_32bitDS descr <PM_seg_size-1,0,0,92h,01000000b,0>
41.
42.     ; 32-битный 4-гигабайтный сегмент данных с базой stack_seg
43.     gdt_32bitSS descr <stack_l-1,0,0, 92h, 01000000b,0>
44.
45.     gdt_size = $-GDT ; размер нашей таблицы GDT+1байт (на саму метку)
46.
47.     gdtr    df 0
48.
49.     ; имена для селекторов
50.     SEL_flatDS      equ   8
51.     SEL_16bitCS     equ   16
52.     SEL_32bitCS     equ   24
53.     SEL_32bitDS     equ   32
54.     SEL_32bitSS     equ   40
55.
56.     ; Таблица дескрипторов прерываний IDT
57.     IDT label   byte
58.
59.     ; первые 32 элемента таблицы
60.     trap_f int_descr 12 dup (<0, SEL_32bitCS, 0, 8Eh, 0>) ; 12 первые элемента
61.     trap_13 int_descr <0, SEL_32bitCS, 0, 8Eh, 0>  ; 13ое исключение
62.     trap_s int_descr 19 dup (<0, SEL_32bitCS, 0, 8Eh, 0>)  ; 19 остальные элемента
63.
64.     ; дескриптор прерывания от таймера
65.     int08 int_descr <0, SEL_32bitCS,0, 8Eh, 0>
66.
67.     ; дескриптор прерывания от клавиатуры
68.     int09 int_descr <0, SEL_32bitCS,0, 8Eh, 0>
69.
70.
71.     idt_size = $-IDT ; размер нашей таблицы IDT+1байт (на саму метку)
72.
```

```
73.     idtr    df 0 ; переменная размера 6 байт как Регистр таблицы дескрипторов прерываний ID
    TR
74.
75.     idtr_real dw    3FFh,0,0 ; содержимое регистра IDTR в реальном режиме
76.
77.     master      db 0            ; маска прерываний ведущего контроллера
78.     slave       db 0            ; ведомого
79.
80.     EScape      db 0            ; флаг - в реальный режим, если ==1
81.     time_08     dd 0            ; счетчик прошедших тиков таймера
82.
83.         msg1 db 'IN ReAL Mode.$'
84.         msg2 db 13, 10, 'To Move to Protected Mode press any key...'
85.              db 13, 10, 'To break the loop press Space...$'
86.         msg3 db 13, 10, 'IN Protected Mode'
87.              db 13, 10, 'To break the loop press Enter...$'
88.
89.         ; Таблица символов ASCII для перевода из скан кода в код ASCII.
90.         ; Номер скан кода = номеру соответствующего элемента в таблице:
91.     ASCII_table db 0, 0, 49, 50, 51, 52, 53, 54, 55, 56, 57, 48, 45, 61, 0, 0
92.             db 81, 87, 69, 82, 84, 89, 85, 73, 79, 80, 91, 93, 0, 0, 65, 83
93.             db 68, 70, 71, 72, 74, 75, 76, 59, 39, 96, 0, 92, 90, 88, 67
94.             db 86, 66, 78, 77, 44, 46, 47
95.     OUT_position    dd 1E0h ; Позиция печати вводимого текста
96.
97.
98.
99.
100.        print_str macro str
101.                MOV AH,9
102.                MOV DX, str
103.                INT  21h
104.        endm
105.
106.        create_number macro
107.                local number1
108.                    CMP  DL, 10
109.                    JL number1
110.                    ADD DL, 'A' - '0' - 10
111.                number1:
112.                    ADD DL, '0'
113.        endm
114.
115.        print_EAX macro
116.                local prcyc1
117.                    PUSH ECX
118.                    PUSH DX
119.
120.                    MOV ECX,8
121.                    ADD EBP,0B8010h
122.
123.
124.                prcyc1:
125.                    MOV DL, AL
126.                    AND DL, 0Fh
127.                    create_number 0
128.                    MOV ES:[EBP],DL
129.                    ROR EAX,4
130.
131.                    SUB EBP,2
132.                    LOOP prcyc1
133.
134.                    SUB EBP,0B8010h
135.                    POP DX
136.                    POP ECX
137.        endm
138.
139.
140.            ; точка входа в 32-битный защищенный режим
141.        PM_entry:
142.                MOV AX,SEL_32bitDS
143.                MOV DS,AX
144.                MOV AX,SEL_flatDS
```

```
145.                MOV ES,AX
146.                MOV AX,SEL_32bitSS
147.                MOV EBX,stack_l
148.                MOV SS,AX
149.                MOV ESp,EBX
150.
151.                ; разрешить прерывания, запрещенные ранее ещё в реальном режиме
152.                STI
153.
154.                CALL    compute_memory
155.
156.        work:
157.                TEST    EScape, 1
158.                JZ  work
159.
160.        goback:
161.                CLI
162.
163.                db  0EAh
164.                dd  offset RM_return
165.                dw  SEL_16bitCS
166.
167.        new_INT08:
168.                PUSH EAX
169.                PUSH EBP
170.                PUSH ECX
171.                PUSH DX
172.                MOV  EAX,time_08
173.
174.                PUSH EBP
175.                MOV EBP, 0
176.                print_EAX 0
177.                POP EBP
178.
179.                INC EAX
180.                MOV time_08,EAX
181.
182.                POP DX
183.                POP ECX
184.                POP EBP
185.
186.                MOV AL,20h
187.                OUT 20h,AL
188.                POP EAX
189.
190.                IRETD
191.
192.        new_INT09:
193.                PUSH EAX
194.                PUSH EBX
195.                PUSH EBP
196.                PUSH EDX
197.
198.                IN  AL,60h       ; Получаем скан-код нажатой клавиши из порта клавиатуры
199.
200.                CMP AL,1Ch       ; Сравниваем с кодом Enter
201.                JNE not_leave
202.                MOV EScape,1
203.                JMP leav
204.        not_leave:
205.                CMP  AL,80h
206.                JA leav
207.                XOR AH,ah
208.                MOV BP,AX
209.                MOV DL, ASCII_table[EBP]
210.                MOV EBP,0B8000h
211.                MOV EBX,OUT_position
212.                MOV ES:[EBP+EBX],DL
213.
214.                ADD EBX,2
215.                MOV OUT_position,EBX
216.        leav:
217.
```

```
218.              IN  AL,61h
219.              OR  AL,80h
220.              OUT 61h,AL
221.
222.              ; Посылаем сигнал EOI:
223.              MOV AL,20h
224.              OUT 20h,AL
225.
226.              POP EDX
227.              POP EBP
228.              POP EBX
229.              POP EAX
230.
231.              ; Выходим из прерывания:
232.              IRET
233.
234.      compute_memory  proc
235.
236.              push    DS
237.              MOV AX, SEL_flatDS
238.              MOV DS, AX
239.              MOV EBX, 100001h
240.              MOV DL,     10101010b
241.
242.              MOV ECX, 0FFEFFFFEh
243.      check:
244.              MOV DH, DS:[EBX]
245.              MOV DS:[EBX], DL
246.              CMP DS:[EBX], DL
247.              jnz end_of_memory
248.              MOV DS:[EBX], DH
249.              INC EBX
250.
251.              LOOP    check
252.      end_of_memory:
253.              POP DS
254.              xor EDX, EDX
255.              MOV EAX, EBX
256.              MOV EBX, 100000h
257.              div EBX
258.
259.              PUSH EBP
260.              MOV EBP,20
261.              print_EAX 0
262.              POP EBP
263.              RET
264.      compute_memory  ENDP
265.
266.      except_1 proc
267.                  iret
268.      except_1 endp
269.
270.      except_13 proc
271.          pop eax
272.          iret
273.      except_13 endp
274.
275.
276.          PM_seg_size = $-GDT
277.      PM_seg  ENDS
278.
279.      stack_seg   SEGMENT  PARA STACK 'STACK' use32
280.          stack_start db  100h dup(?)
281.          stack_l = $-stack_start
282.      stack_seg   ENDS
283.
284.      RM_seg  SEGMENT PARA PUBLIC 'CODE' USE16
285.          ASSUME CS:RM_seg, DS:PM_seg, SS:stack_seg
286.
287.      start:
288.
289.              MOV   AX,PM_seg
```

```
290.                MOV   DS,AX
291.
292.                MOV AH, 09h
293.                MOV EDX, offset msg1
294.                INT  21h
295.
296.                MOV AH, 09h
297.                MOV EDX, offset msg2
298.                INT  21h
299.
300.                ;ожидаем ввода клавиатуры
301.                PUSH EAX
302.                MOV AH,10h
303.                INT  16h
304.
305.                CMP  AL, 20h
306.                JE exit
307.                POP EAX
308.
309.                ; очистить экран
310.                MOV AX,3
311.                INT     10h
312.
313.                PUSH PM_seg
314.                POP DS
315.
316.                xor EAX,EAX
317.                MOV AX,RM_seg
318.                shl EAX,4
319.                MOV word ptr gdt_16bitCS.base_l,AX
320.                shr EAX,16
321.                MOV byte ptr gdt_16bitCS.base_m,AL
322.                MOV AX,PM_seg
323.                shl EAX,4
324.                PUSH EAX        ; для вычисления адреса idt
325.                PUSH EAX        ; для вычисления адреса gdt
326.                MOV word ptr GDT_32bitCS.base_l,AX
327.                MOV word ptr GDT_32bitSS.base_l,AX
328.                MOV word ptr GDT_32bitDS.base_l,AX
329.                shr EAX,16
330.                MOV byte ptr GDT_32bitCS.base_m,AL
331.                MOV byte ptr GDT_32bitSS.base_m,AL
332.                MOV byte ptr GDT_32bitDS.base_m,AL
333.
334.                ; вычислим линейный адрес GDT
335.                POP EAX
336.                ADD EAX,offset GDT
337.                MOV dword ptr gdtr+2,EAX
338.                MOV word ptr gdtr, gdt_size-1
339.                LGDT fword ptr gdtr
340.
341.                ; вычислим линейный адрес IDT
342.                POP EAX
343.                ADD EAX,offset IDT
344.                MOV dword ptr idtr+2,EAX
345.                MOV word ptr idtr, idt_size-1
346.
347.                ; Заполним смещение в дескрипторах прерываний
348.                MOV EAX, offset new_INT08
349.                MOV int08.offs_l, AX
350.                SHR EAX, 16
351.                MOV int08.offs_h, AX
352.                MOV EAX, offset new_INT09
353.                MOV int09.offs_l, AX
354.                SHR EAX, 16
355.                MOV int09.offs_h, AX
356.
357.                MOV EAX, offset except_1
358.                 MOV trap_f.offs_l, AX
359.                 SHR EAX, 16
360.                 MOV trap_f.offs_h, AX
361.
```

```
362.                MOV EAX, offset except_1
363.                MOV trap_s.offs_l, AX
364.                SHR EAX, 16
365.                MOV trap_s.offs_h, AX
366.
367.                MOV EAX, offset except_13
368.                MOV trap_13.offs_l, AX
369.                SHR EAX, 16
370.                MOV trap_13.offs_h, AX
371.
372.               ; сохраним маски прерываний контроллеров
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.               MOV AL, 20h
380.               OUT 21h, AL
381.               MOV AL, 4
382.
383.               OUT 21h, AL
384.               MOV AL, 1
385.               OUT 21h, AL
386.
387.               MOV AH, 09h
388.               MOV EDX, offset msg3
389.               INT  21h
390.
391.               ; Запретим все прерывания в ведущем контроллере, кроме IRQ0 и IRQ1
392.               MOV AL, 0FCh
393.               OUT 21h, AL
394.
395.               MOV AL, 0FFh
396.               OUT 0A1h, AL
397.
398.               ; загрузим IDT
399.               LIDT    fword ptr idtr
400.
401.
402.               ; если мы собираемся работать с 32-битной памятью, стоит открыть A20
403.               ; A20 - линия ("шина"), через которую осуществляется доступ ко всей
404.               IN  AL,92h
405.               OR  AL,2
406.               OUT 92h,AL
407.
408.               CLI
409.
410.               IN  AL,70h
411.               OR  AL,80h
412.               OUT 70h,AL
413.
414.               MOV EAX,cr0
415.               OR  AL,1
416.               MOV cr0,EAX
417.
418.               db  66h
419.               db  0EAh
420.               dd  offset PM_entry
421.               dw  SEL_32bitCS
422.
423.       RM_return:
424.               MOV EAX,cr0
425.               AND AL,0FEh
426.               MOV cr0,EAX
427.
428.               db  0EAh
429.               dw  $+4
430.               dw  RM_seg
431.
432.               MOV AX,PM_seg
433.               MOV DS,AX
```

```
434.                MOV ES,AX
435.                MOV AX,stack_seg
436.                MOV BX,stack_l
437.                MOV SS,AX
438.                MOV sp,BX
439.
440.                ;перепрограммируем ведущий контроллер обратно на вектор 8
441.                MOV AL, 11h
442.                OUT 20h, AL
443.                MOV AL, 8
444.                OUT 21h, AL
445.                MOV AL, 4
446.                OUT 21h, AL
447.                MOV AL, 1
448.                OUT 21h, AL
449.
450.                MOV AL, master
451.                OUT 21h, AL
452.                MOV AL, slave
453.                OUT 0A1h, AL
454.
455.                LIDT fword ptr idtr_real
456.
457.                IN  AL,70h
458.                AND AL,07FH
459.                OUT 70h,AL
460.
461.                STI
462.
463.                MOV AX,3
464.                INT 10h
465.
466.                JMP start
467.
468.                MOV AH, 09h
469.                MOV EDX, offset msg1
470.                INT  21h
471.        exit:
472.                MOV AH,4Ch
473.                INT 21h
474.
475.        RM_seg_size = $-start
476.        RM_seg  ENDS
477.        END start
```