



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»

Отчет по лабораторной работе №2 по курсу "Операционные системы"

Тема Защищённый режим

Студент Фролов Е. А.

Группа ИУ7-55Б

Оценка (баллы) _____

Преподаватели Рязанова Н.А.

Код программы

```
1 .386p
2
3 descr struct
4     limit    dw 0
5     base_l   dw 0
6     base_m   db 0
7     attr_1   db 0
8     arrt_2   db 0
9     base_h   db 0
10 descr ENDS
11
12 intr struct
13     offs_l   dw 0
14     sel       dw 0
15     rsrv     db 0
16     attr     db 0
17     offs_h   dw 0
18 intr ENDS
19
20 pm_seg      SEGMENT PARA PUBLIC 'CODE' USE32
21     assume cs:pm_seg
22
23     gdt      label byte
24     gdt_null descr <>
25     gdt_data descr <0FFFFh, 0, 0, 92h, 0CFh, 0>
26     gdt_code16 descr <rm_seg_size-1, 0, 0, 98h, 0, 0>
27     gdt_code32 descr <pm_seg_size-1, 0, 0, 98h, 0CFh, 0>
28     gdt_data32 descr <pm_seg_size-1, 0, 0, 92h, 0CFh, 0>
29     gdt_stack32 descr <stack_size-1, 0, 0, 92h, 0CFh, 0>
30     gdt_size = $ - gdt
31
32     gdtr dw gdt_size - 1
33         dd ?
34
35     ; Селекторы сегментов
36     sel_data      equ 8
37     sel_code16    equ 16
38     sel_code32    equ 24
39     sel_data32    equ 32
40     sel_stack32   equ 40
41
42     ; таблица дескрипторов прерываний IDT
43     idt           label byte
44     trap1         intr 13 dup (<0, sel_code32, 0, 8Fh, 0>)
45     trap13        intr <0, sel_code32, 0, 8Fh, 0>
```

```

46 trap2      intr 18 dup (<0, sel_code32, 0, 8Fh, 0>)
47 int_time   intr <0, sel_code32, 0, 8Eh, 0>
48 int_keyboard intr <0, sel_code32, 0, 8Eh, 0>
49 idt_size=$-idt
50 idtr       dw idt_size-1
51           dd ?
52
53 rm_idtr    dw 3FFh, 0, 0
54
55 hex        db 'h'
56 hex_len=$-hex
57 mb         db 'MB'
58 mb_len=$-mb
59
60 realm_msg  db 'Now_DOS_in_real_mode.'
61 to_pm_msg  db 'DOS_switched_to_protected_mode.'
62 to_pm_msg_len=$-to_pm_msg
63 timer_msg  db 'Timer_ticks:   '
64 timer_msg_len=$-timer_msg
65 memory_msg db 'Available_memory:'
66 memory_msg_len=$-memory_msg
67 esc_from_pr db 'To_change_to_real_mode_press_ESC'
68 esc_from_pr_len=$-esc_from_pr
69 ret_to_rm_msg db 'DOS_switched_to_real_mode.'
70
71 to_ascii   db 0, 1Bh, '1', '2', '3', '4', '5', '6', '7', '8', '9', '0',
              '_ ', '=', 8
72           db ' ', 'q', 'w', 'e', 'r', 't', 'y', 'u', 'i', 'o', 'p', '[',
              ', ]', '$'
73           db ' ', 'a', 's', 'd', 'f', 'g', 'h', 'j', 'k', 'l', ';', ' ',
              '"', 0
74           db '\ ', 'z', 'x', 'c', 'v', 'b', 'n', 'm', ' ', '.', '/', 0,
              0, 0, ' ', 0, 0
75           db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
76
77 attr1      db 1Fh
78 attr2      db 2Fh
79 screen_addr dd 640
80 timer      dd 0
81
82 master     db 0
83 slave      db 0
84
85 ; Макрос вывода строки в видеобуфер
86 print_str macro msg, len, offset
87 local print

```

```

88     push ebp
89     mov ecx, len
90     mov ebp, 0B8000h
91     add ebp, offset
92     xor esi, esi
93     mov ah, attr2
94 print:
95     mov al, byte ptr msg[esi]
96     mov es:[ebp], ax
97     add ebp, 2
98     inc esi
99     loop print
100    pop ebp
101 endm
102
103 ; Макрос отправки сигнал EOI контроллеру прерываний
104 send_eoi macro
105     mov al, 20h
106     out 20h, al
107 endm
108
109 pm_start:
110     ; Установить сегменты защищенного режима
111     mov ax, sel_data
112     mov ds, ax
113     mov es, ax
114     mov ax, sel_stack32
115     mov ebx, stack_size
116     mov ss, ax
117     mov esp, ebx
118
119     ; Разрешить маскируемые прерывания
120     sti
121
122     ; Вывод информации в видеобуфер
123     print_str to_pm_msg, to_pm_msg_len, 380
124     print_str timer_msg, timer_msg_len, 540
125     print_str memory_msg, memory_msg_len, 5 * 160 + 60
126     print_str esc_from_pr, esc_from_pr_len, 6 * 160 + 60
127
128     call available_memory
129     jmp $
130
131     ; Обработчик исключения общей защиты
132     exc13 proc
133     pop eax

```

```

134     iret
135     exc13     endp
136
137     ; Обработчик остальных исключений
138     dummy_exc proc
139     iret
140     dummy_exc endp
141
142     ; Обработчик прерывания от системного таймера
143     int_time_handler:
144     push eax
145     push ebp
146     push ecx
147     push dx
148
149     mov eax, timer
150
151     ; Вывести счетчик в видеобуфер
152     mov ebp, 0B8000h
153     mov ecx, 8
154     add ebp, 550 + 2 * (timer_msg_len)
155     mov dh, attr2
156 main_loop:
157     mov dl, al
158     and dl, 0Fh
159     cmp dl, 10
160     jl less_than_10
161     sub dl, 10
162     add dl, 'A'
163     jmp print
164
165 less_than_10:
166     add dl, '0'
167
168 print:
169     mov es:[ebp], dx
170     ror eax, 4
171     sub ebp, 2
172     loop main_loop
173
174     ; Инкрементировать и сохранить счетчик
175     inc eax
176     mov timer, eax
177
178     send_eoi
179     pop dx

```

```

180     pop ecx
181     pop ebp
182     pop eax
183
184     iretd
185
186 ; Обработчик прерывания от клавиатуры
187 int_keyboard_handler:
188     push    eax
189     push    ebx
190     push    es
191     push    ds
192
193     ; Получить из порта клавиатуры скан-код нажатой клавиши
194     in      al, 60h
195
196     ; Нажата клавиша ESC
197     cmp al, 01h
198     je      esc_pressed
199
200     ; Нажата необслуживаемая клавиша
201     cmp al, 39h
202     ja      skip_translate
203
204     ; Преобразовать скан-код в ASCII
205     mov bx, sel_data32
206     mov ds, bx
207     mov ebx, offset to_ascii
208     xlatb
209     mov bx, sel_data
210     mov es, bx
211     mov ebx, screen_addr
212
213     ; Нажата клавиша Backspace
214     cmp al, 8
215     je      backspace_pressed
216
217     ; Вывести символ на экран
218     mov es:[ebx+0B8000h], al
219     add dword ptr screen_addr, 2
220     jmp skip_translate
221
222 backspace_pressed:
223     ; Удалить символ
224     mov al, '␣'
225     sub ebx, 2 ; смещаемся на один символ влево (предыдущая цифра в EAX)

```

```

226     mov es:[ebx+0B8000h], al
227     mov screen_addr, ebx
228
229 skip_translate:
230     ; Разрешить работу клавиатуры
231     in al, 61h
232     or al, 80h
233     out 61h, al
234
235     send_eoi
236     pop ds
237     pop es
238     pop ebx
239     pop eax
240
241     iretd
242
243 esc_pressed:
244     ; Разрешить работу клавиатуры
245     in al, 61h
246     or al, 80h
247     out 61h, al
248
249     send_eoi
250     pop ds
251     pop es
252     pop ebx
253     pop eax
254
255     ; Запретить маскируемые прерывания
256     cli
257
258     ; Вернуться в реальный режим
259     db 0EAh
260     dd offset rm_return
261     dw sel_code16
262
263 ; Процедура определения доступного объема оперативной памяти
264 available_memory proc
265     push    ds
266
267     mov ax, sel_data
268     mov ds, ax
269     ; Пропустить первый мегабайт памяти
270     mov ebx, 100001h
271     ; Установить проверочный байт

```

```

272     mov dl, 0FFh
273     mov ecx, 0FFEFFFFFFh ;в есх кладём количество оставшейся памяти
274
275 check:
276     ; Проверка сигнатуры
277     mov dh, ds:[ebx]
278     mov ds:[ebx], dl
279     cmp ds:[ebx], dl
280     jnz end_of_memory
281     mov ds:[ebx], dh
282     inc ebx
283     loop check
284
285 end_of_memory:
286     pop ds
287     xor edx, edx
288     mov eax, ebx
289     ; Разделить на мегабайт
290     mov ebx, 100000h ; делим на 1 Мб, чтобы получить результат в мегабайтах
291     div ebx
292
293     push ecx
294     push dx
295     push ebp
296
297     ; Вывести объем памяти на экран
298     mov ebp, 0B8000h
299     mov ecx, 8
300     add ebp, 5 * 160 + 2 * (memory_msg_len + 7) + 60
301     mov dh, attr2
302 cycle:
303     mov dl, al
304     and dl, 0Fh
305     cmp dl, 10
306     jl number
307     sub dl, 10
308     add dl, 'A'
309     jmp print_m
310 number:
311     add dl, '0'
312 print_m:
313     mov es:[ebp], dx
314     ror eax, 4
315
316     sub ebp, 2
317     loop cycle

```



```

318     sub ebp, 0B8000h
319
320     pop ebp ;восстанавливаем потраченное смещение ЕВР
321     pop dx
322     pop ecx
323     ret
324 available_memory endp
325
326     pm_seg_size=$-gdt
327 pm_seg ENDS
328
329
330 rm_seg SEGMENT PARA PUBLIC 'CODE' USE16 ; USE16 - используем нижние части ре
    гистров
331     assume cs:rm_seg, ds:pm_seg, ss:s_seg
332
333 ; Макрос очистки экрана
334 cls macro
335     mov ax, 3
336     int 10h
337 endm
338
339 ; Макрос печати строки
340 print_str macro msg
341     mov ah, 9
342     mov edx, offset msg
343     int 21h
344 endm
345
346 rm_start:
347     mov ax, pm_seg
348     mov ds, ax
349
350     cls
351
352     mov AX, 0B800h
353     mov ES, AX
354     mov DI, 220
355     mov cx, 21
356     mov ebx, offset realm_msg
357     mov ah, attr1
358     mov al, byte ptr[ebx]
359 screen0:
360     stosw
361     inc bx
362     mov al, byte ptr[ebx]

```

```

363     loop screen0
364
365     ; Вычислить базы для всех используемых дескрипторов сегментов
366     xor eax, eax
367     mov ax, rm_seg
368     shl eax, 4
369     mov word ptr gdt_code16 + 2, ax
370     shr eax, 16
371     mov byte ptr gdt_code16 + 4, al
372     mov ax, pm_seg
373     shl eax, 4
374     push eax
375     push eax
376     mov word ptr gdt_code32 + 2, ax
377     mov word ptr gdt_stack32 + 2, ax
378     mov word ptr gdt_data32 + 2, ax
379     shr eax, 16
380     mov byte ptr gdt_code32 + 4, al
381     mov byte ptr gdt_stack32 + 4, al
382     mov byte ptr gdt_data32 + 4, al
383
384     ; вычислим линейный адрес GDT
385     pop eax
386     add eax, offset GDT ; в eax будет полный линейный адрес GDT
387     mov dword ptr gdtr + 2, eax
388     mov word ptr gdtr, gdt_size - 1
389
390     ; Установить регистр GDTR
391     lgdt fword ptr gdtr
392
393     ; Вычислить линейный адрес IDT
394     pop eax
395     add eax, offset idt
396     mov dword ptr idtr + 2, eax
397     mov word ptr idtr, idt_size - 1
398
399     ; Заполнить смещения в дескрипторах прерываний
400     mov eax, offset dummy_exc
401     mov trap1.off_1, ax
402     shr eax, 16
403     mov trap1.off_h, ax
404     mov eax, offset exc13
405     mov trap13.off_1, ax
406     shr eax, 16
407     mov trap13.off_h, ax
408     mov eax, offset dummy_exc

```

```

409     mov trap2.off_1, ax
410     shr eax, 16
411     mov trap2.off_h, ax
412     mov eax, offset int_time_handler
413     mov int_time.off_1, ax
414     shr eax, 16
415     mov int_time.off_h, ax
416     mov eax, offset int_keyboard_handler
417     mov int_keyboard.off_1, ax
418     shr eax, 16
419     mov int_keyboard.off_h, ax
420
421     ;сохраним маски прерываний контроллеров
422     in al, 21h
423     mov master, al
424     in al, 0A1h
425     mov slave, al
426
427     ; Перепрограммировать ведущий контроллер прерываний
428     mov dx, 20h
429     mov al, 11h
430     out dx, al
431     inc dx
432     mov al, 20h
433     out dx, al
434     mov al, 4
435     out dx, al
436     mov al, 1
437     out dx, al
438
439     ; Запретить все прерывания в ведущем контроллере, кроме IRQ0 и IRQ1
440     mov al, 11111100b
441     out dx, al
442
443     ; Запретить все прерывания в ведомом контроллере
444     mov dx, 0A1h
445     mov al, 0FFh
446     out dx, al
447
448     ; Загрузить регистр IDTR
449     lidt fword ptr idtr
450
451     ; Открыть линию A20
452     mov al, 0D1h
453     out 64h, al
454     mov al, 0DFh

```

```

455     out 60h, al
456
457     ; Отключить маскируемые и немаскируемые прерывания
458     cli
459     in al, 70h
460     or al, 80h
461     out 70h, al
462
463     mov eax, cr0
464     or al, 1 ; сбрасываем флаг защищенного режима
465     mov cr0, eax
466
467     ; Перейти в сегмент кода защищенного режима
468     db 66h
469     db 0EAh
470     dd offset pm_start
471     dw sel_code32
472
473 rm_return:
474     ; Перейти в реальный режим сбросом соответствующего бита регистра CR0
475     mov eax, cr0
476     and al, 0FEh
477     mov cr0, eax
478
479     ; Сбросить очередь и загрузить CS
480     db 0EAh
481     dw $+4
482     dw rm_seg
483
484     ; Восстановить регистры для работы в реальном режиме
485     mov ax, pm_seg
486     mov ds, ax
487     mov es, ax
488     mov ax, s_seg
489     mov ss, ax
490     mov ax, stack_size
491     mov sp, ax
492
493     mov al, 11h
494     out 20h, al
495     mov al, 8 ;отправка смещения
496     out 21h, al
497     mov al, 4
498     out 21h, al
499     mov al, 1
500     out 21h, al

```

```

501
502     ; Восстановить маски контроллеров прерываний
503     mov al, master
504     out 21h, al
505     mov al, slave
506     out 0A1h, al
507
508     ; Загрузить таблицу дескрипторов прерываний реального режима
509     lidt fword ptr rm_idtr
510
511     ; Закрыть линию A20
512     mov al, 0D1h
513     out 64h, al
514     mov al, 0DDh
515     out 60h, al
516
517     ; Разрешить немаскируемые и маскируемые прерывания
518     in al, 70h
519     and al, 07FH
520     out 70h, al
521     sti
522
523     mov AX, 0B800h
524     mov ES, AX
525     mov DI, 7*160+60
526     mov cx, 26
527     mov ebx, offset ret_to_rm_msg
528     mov ah, attr1
529     mov al, byte ptr [ebx]
530 screen01:
531     stosw
532     inc bx
533     mov al, byte ptr [ebx]
534     loop screen01
535
536     mov ah, 4Ch
537     int 21h
538     rm_seg_size = $-rm_start
539 rm_seg ENDS
540
541 s_seg SEGMENT PARA STACK 'STACK'
542     stack_start db 100h dup(?)
543     stack_size=$-stack_start
544 s_seg ENDS
545     end rm_start

```

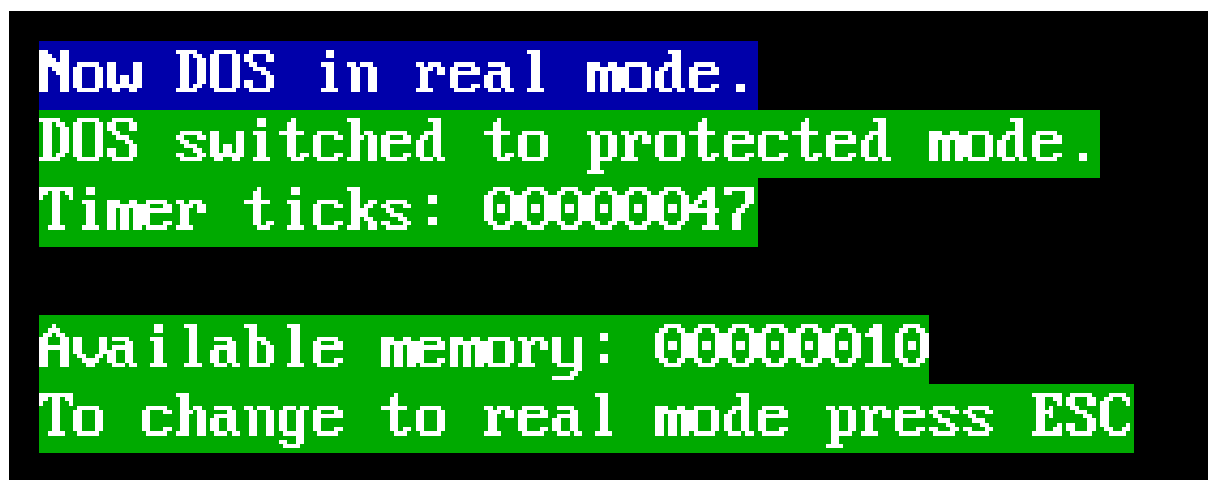
Демонстрация работы программы

В обработчике прерывания отдельно обрабатывается BackSpace и Enter.

В случае BackSpace последний перед курсором символ стирается.

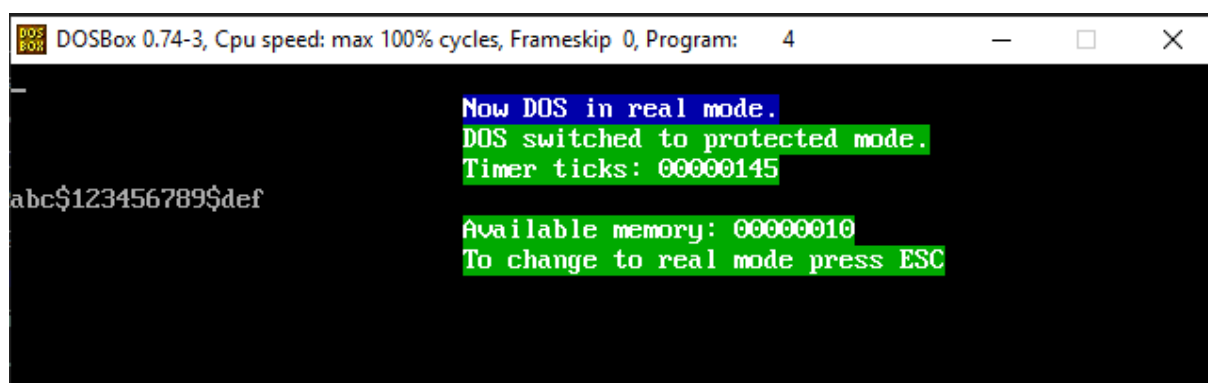
В случае Enter происходит смена регистра и печатается символ доллара.

В начале программа стартует в реальном режиме и переходит в защищённый режим:



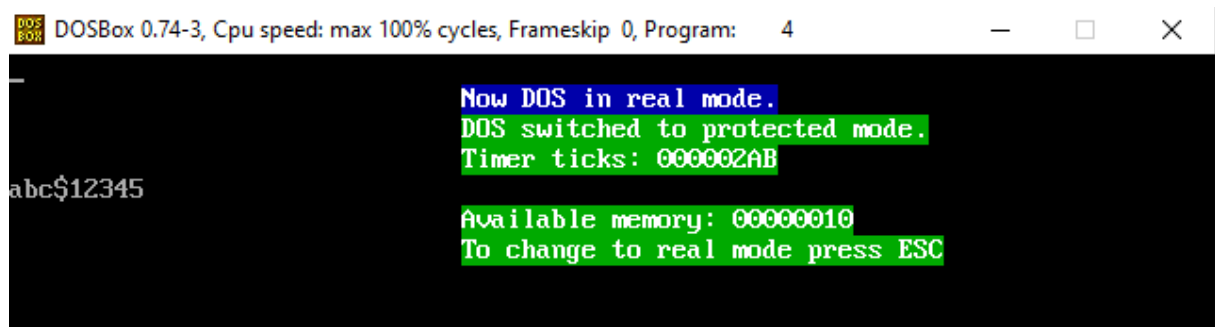
```
Now DOS in real mode.  
DOS switched to protected mode.  
Timer ticks: 00000047  
  
Available memory: 00000010  
To change to real mode press ESC
```

После перехода в защищённый режим программа подсчитывает количество доступной памяти. Далее, пользователь может вводить доступные ему символы, а с помощью Enter выводить знак доллара.



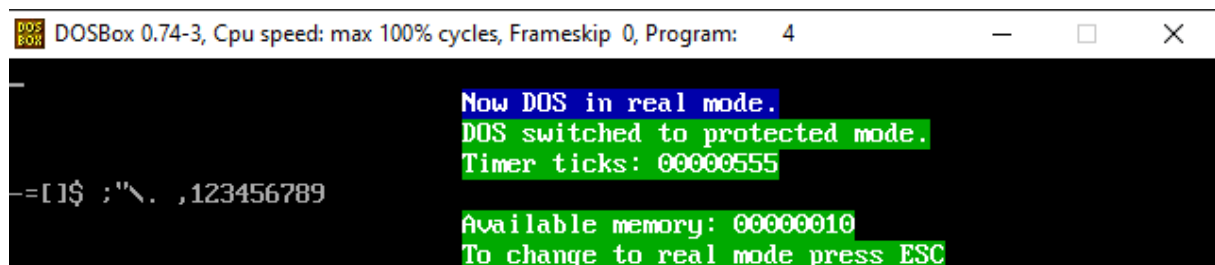
```
DOSBox 0.74-3, Cpu speed: max 100% cycles, Frameskip 0, Program: 4  
Now DOS in real mode.  
DOS switched to protected mode.  
Timer ticks: 00000145  
  
Available memory: 00000010  
To change to real mode press ESC
```

Работа клавиши BackSpace:



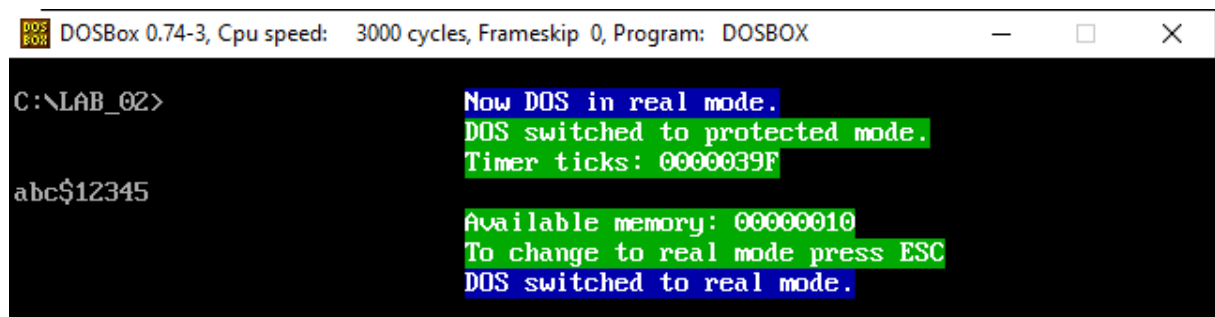
```
DOSBox 0.74-3, Cpu speed: max 100% cycles, Frameskip 0, Program: 4
Now DOS in real mode.
DOS switched to protected mode.
Timer ticks: 000002AB
Available memory: 00000010
To change to real mode press ESC
abc$12345
```

Полный набор доступных символов:



```
DOSBox 0.74-3, Cpu speed: max 100% cycles, Frameskip 0, Program: 4
Now DOS in real mode.
DOS switched to protected mode.
Timer ticks: 00000555
Available memory: 00000010
To change to real mode press ESC
-=[!$ ;"\\. ,123456789
```

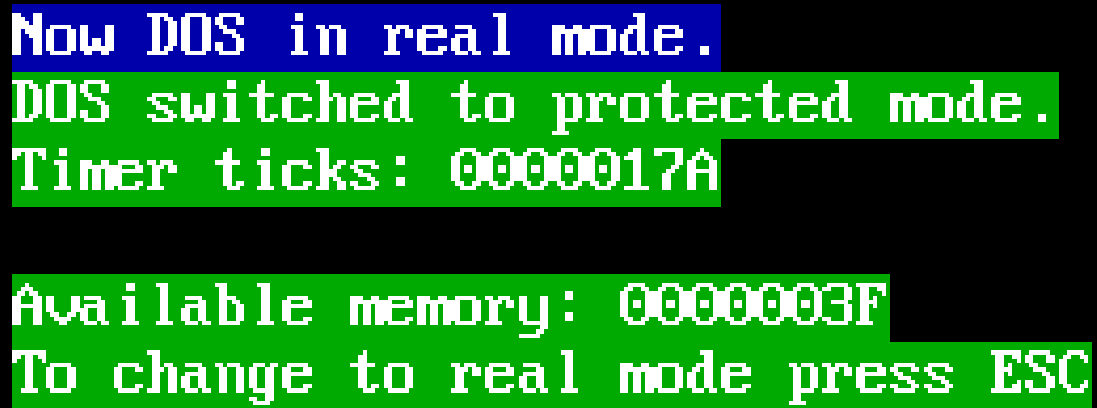
В конце программа по нажатию Esc возвращается в реальный режим и завершается:



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
C:\\LAB_02>
Now DOS in real mode.
DOS switched to protected mode.
Timer ticks: 0000039F
Available memory: 00000010
To change to real mode press ESC
DOS switched to real mode.
abc$12345
```

Максимальная память, доступная в DOSBOX

Также я заменил в файле dosbox-0.74-3.conf поле memsize на 64 и убедился, что максимальное число памяти не превысило 63.



The screenshot shows the DOSBOX startup screen with a black background. The text is displayed in a monospaced font. The first line, "Now DOS in real mode.", is highlighted in blue. The subsequent lines, "DOS switched to protected mode.", "Timer ticks: 0000017A", "Available memory: 0000003F", and "To change to real mode press ESC", are highlighted in green.

```
Now DOS in real mode.  
DOS switched to protected mode.  
Timer ticks: 0000017A  
  
Available memory: 0000003F  
To change to real mode press ESC
```