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»
по теме «Защищенный режим»

Группа ИУ7-54Б

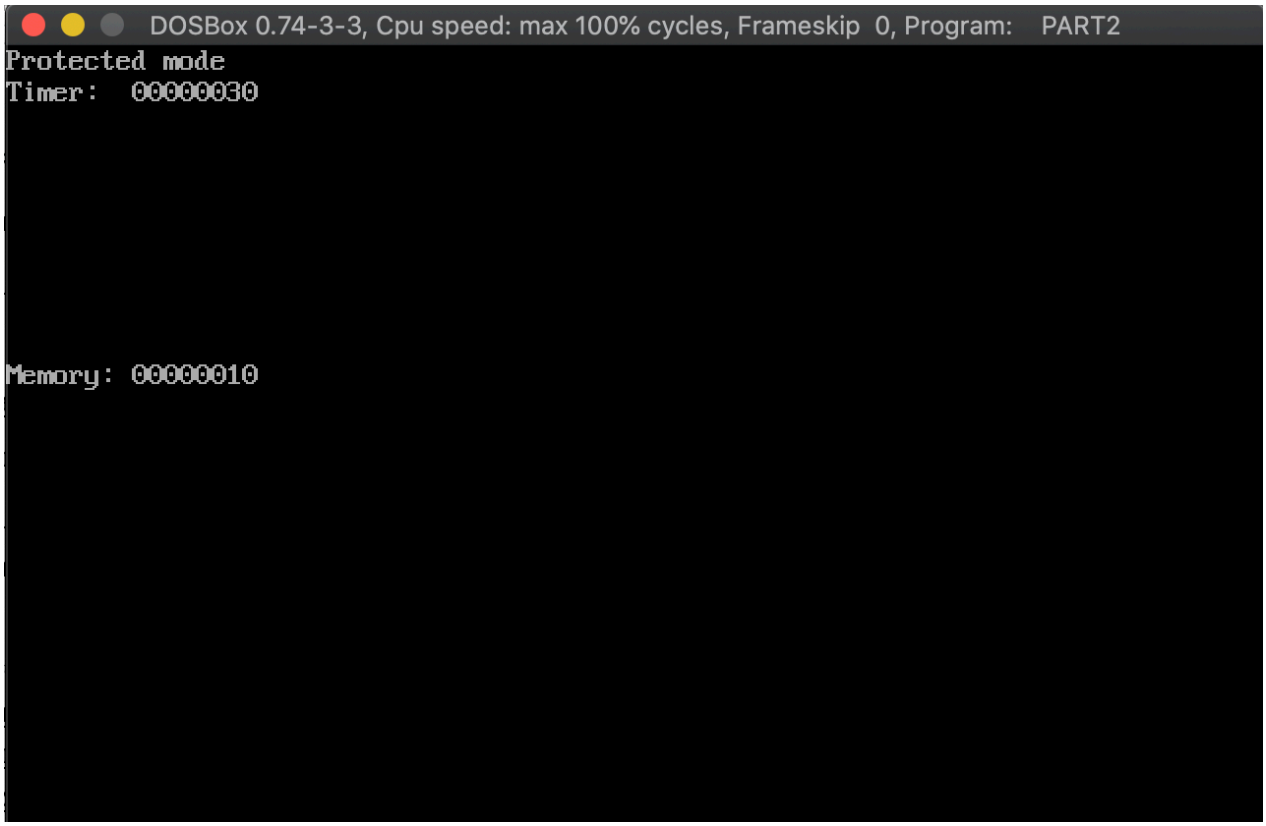
Студент Садулаева Т.Р.

Преподаватель Рязанова Н. Ю.

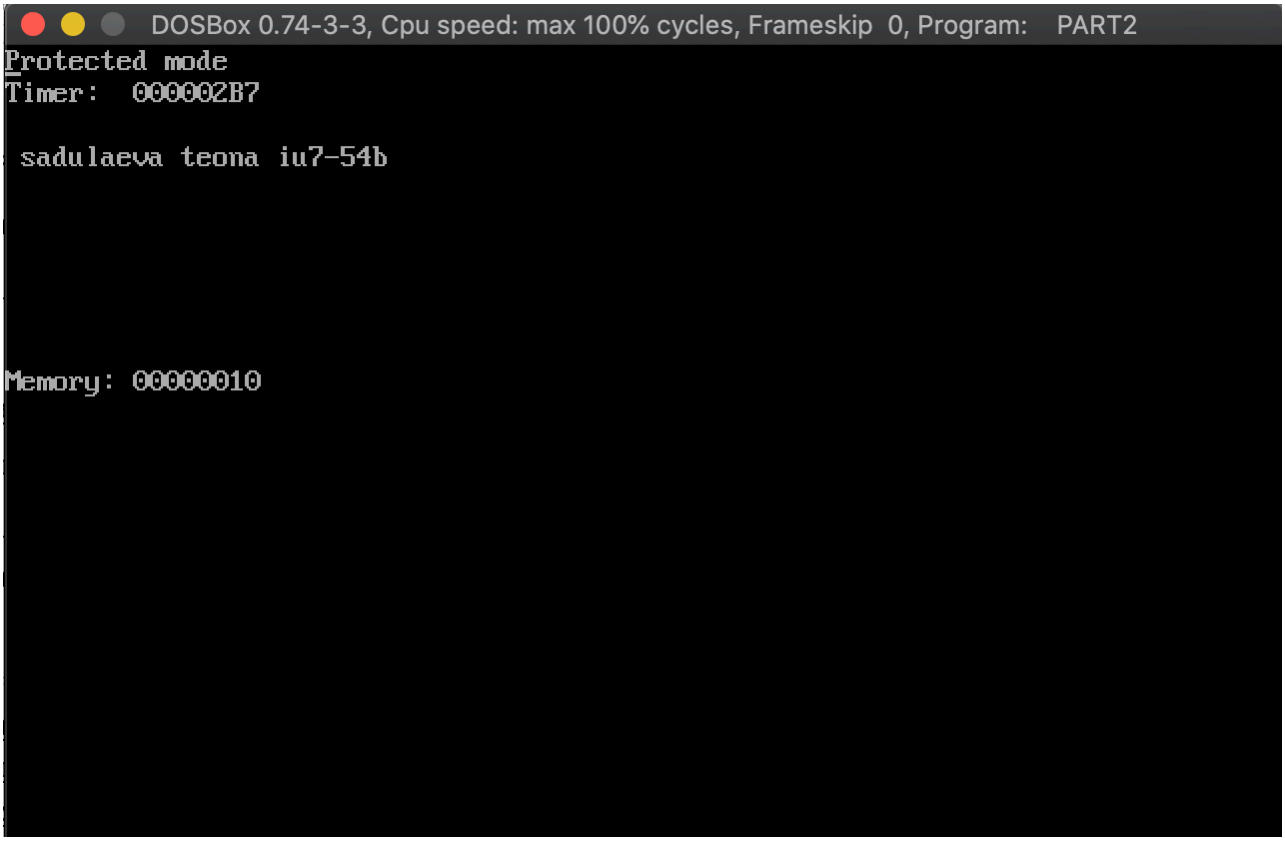
Москва, 2020 г.

Демонстрация работоспособности программы

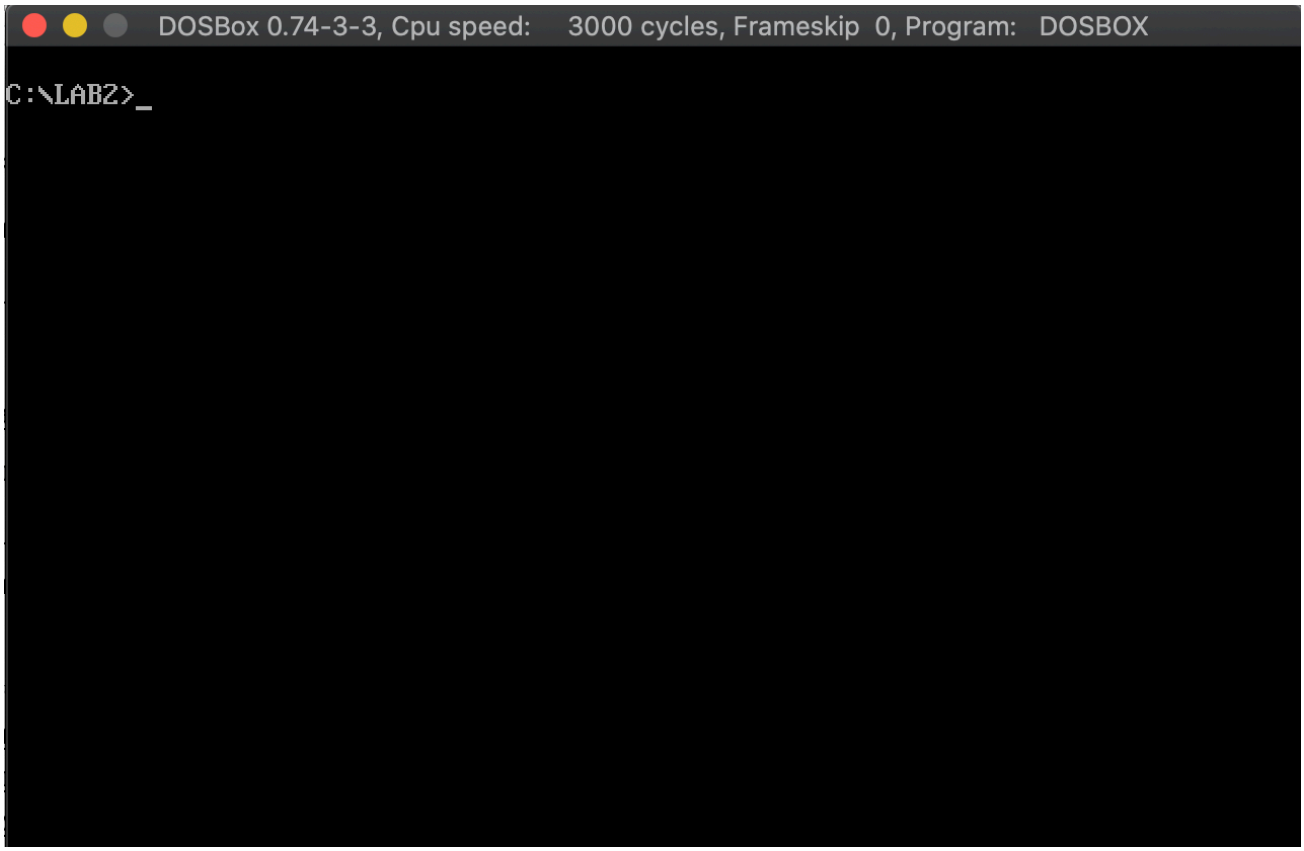
Запуск



Демонстрация работы клавиатуры



Демонстрация выхода из защищенного режима



Код программы

```
.386p
num_output macro  v,r,c
   local  cycle,num,print

   push   EAX
   push   ECX
   push   EDX
   push   EBP

   mov       AX, v
   mov       EBP, r * 160 + c * 2
   mov       ECX, 4
   add       EBP, 0B8000h
cycle:
   mov       DL, AL
   and       DL, 0Fh
   cmp       DL, 10
   jl    num
   add       DL, 'a' - 10
   jmp       print
num:
   add       DL, '0'
print:
   mov       ES:[EBP], DL
   ror       EAX, 4
   sub       EBP, 2
   loop   cycle

   pop       EBP
   pop       EDX
   pop       ECX
   pop       EAX
endm

msg_output macro  msg, size, r, c
   local  show

   push   EBP
   push   EAX
   push   ECX
   push   ESI

   xor       EAX, EAX
   mov       EBP, r * 160 + c * 2
   add       EBP, 0B8000h
   mov       ECX, size
   xor       ESI, ESI
show:
   mov       AL, byte ptr msg[ESI]
```

```
   mov       ES:[EBP], AL
   add       EBP, 2
   inc       ESI
   loop   show

   pop       ESI
   pop       ECX
   pop       EAX
   pop       EBP
endm

segdesc    struc
   limit  dw 0
   base_l dw 0
   base_m db 0
   attr_1 db 0
   arrt_2 db 0
   base_h db 0
segdesc    ends

intdesc    struc
   offs_l dw 0
   sel    dw 0
   rsrv   db 0
   attr   db 0
   offs_h dw 0
intdesc    ends

data16 segment    'data' use16

   gdt    label  byte
      gdt_null             segdesc<>
      gdt_4gb      segdesc <0FFFFh, 0, 0, 10010010b, 10001111b, 0>
      gdt_code16   segdesc<code16_size-1,0,0,10011000b,00000000b,0>
      gdt_code32   segdesc<code32_size-1,0,0,10011000b,01000000b,0>
      gdt_data16   segdesc<data16_size-1,0,0,10011000b,00000000b,0>
      gdt_stack32 segdesc<stack32_size-1,0,0,10011000b,01000000b,0>
   gdt_size = $-gdt

;Псевдодескриптор GDTR
   gdtr   dw gdt_size-1  ;Limit GDT
          dd ?          ;Линейный 32-битный адрес GDT

;Селекторы сегментных дескрипторов
   sel_4gb                 equ 8
   sel_code16              equ 16
   sel_code32              equ 24
   sel_data16              equ 32
   sel_stack32             equ 40
```

```
;Таблица дескрипторов прерываний
   idt    label  byte
      trap1           intdesc    13 dup (<,sel_code32,,8fh>)
      trap13          intdesc    <0, sel_code32,,8fh>
;Исключение общей защиты
      trap2           intdesc    18 dup (<,sel_code32,,8fh>)
      timer           intdesc    <,sel_code32,,8eh>
;Дескриптор прерывания от таймера
      keyboard        intdesc    <,sel_code32,,8eh>
;Дескриптор прерывания от клавиатуры
   idt_size = $-idt

;Псевдодескриптор IDTR
   idtr   dw idt_size-1
         dd ?

   idtr_r dw 3ffh,0,0

   msg_proctected_mode    db 'Protected mode'
   msg_proctected_mode_size = $-msg_proctected_mode

   msg_real_mode  db 'Real mode', '$'
   msg_real_mode_size = $-msg_real_mode-1

   msg_time   db 'TIME: '
   msg_time_size = $-msg_time

   msg_memory db 'MEMORY: '
   msg_memory_size = $-msg_memory

   msg_trap13 db 'Trap 13 :  EIP ='
   msg_trap13_size = $-msg_trap13


   elem_position  dd 20*160
   cnt_timer  dw 0

;Маски прерываний ведущего и ведомого контроллера
   master db 0
   slave  db 0

   ascii_table db 0,1bh,'1','2','3','4','5','6','7','8','9','0','-
','=',8
            db '
','q','w','e','r','t','y','u','i','o','p','[',']','$'
            db ' ','a','s','d','f','g','h','j','k','l',';','"',0
            db '\','z','x','c','v','b','n','m',',','.','/',0,0,0,'
',0,0
            db 0,0,0,0,0,0,0,0,0,0,0
```

```
data16_size = $-gdt
data16 ends

code32 segment    'code' use32
   assume cs:code32, ds:data16, ss:stack32

main32:
;Установка селекторов в сегментные регистры
   mov       AX, sel_4gb
   mov       ES, AX

   mov    AX, sel_data16
   mov       DS, AX

   mov       AX, sel_stack32
   mov       SS, AX

   mov       EBX, stack32_size
   mov       ESP, EBX

;Разрешение маскируемых и немаскируемых прерываний
   in    AL, 70h
   and       AL, 7Fh
   out       70h, AL
   sti

   msg_output msg_proctected_mode, msg_proctected_mode_size, 1, 0
   msg_output msg_time, msg_time_size, 1, 65
   msg_output msg_memory, msg_memory_size, 1, 45

   call   find_empty_mem

   jmp $

;Запрещение маскируемых и немаскируемых прерываний
   cli
   in    AL, 70h
   or    AL, 80h
   out       70h, AL

find_empty_mem proc
   push   EAX
   push   EBX
   push   EDX

   mov       EBX, 100001h

   mov       DL, 11101011b
   mov       ECX, 0FFEFFFFFh
```

```
check:
   mov       DH, ES:[EBX]
   mov       ES:[EBX], DL
   cmp       ES:[EBX], DL
   jnz       end_of_memory
   mov       ES:[EBX], DH
   inc       EBX
   loop   check

end_of_memory:

   xor       EDX, EDX
   mov       EAX, EBX
   mov       EBX, 100000h
   div       EBX
   pop       EDX
   pop       EBX

   num_output AX, 1, 55

   pop       EAX
   ret
find_empty_mem endp

;Заглушка для исключений 1-12 и 14-32
dummy_exc proc
   iretd
dummy_exc endp

;Заглушка для 13 исключения

exc13  proc
   pop       EAX
   pop       EAX
   msg_output msg_trap13, msg_trap13_size, 5, 0
   num_output AX, 5, 30
   shr    EAX, 16
   num_output AX, 5, 28
   iretd
exc13  endp

;Обработчик прерывания INT 8H
timer_handler:
   push   EAX

   mov       AX, cnt_timer
   inc       AX
   mov       cnt_timer, AX

   num_output cnt_timer, 1, 75
```

```
;Посылаем сигнал EOI контроллеру прерываний
   mov       AL, 20h
   out       20h, AL

   pop       EAX
   iretd

;Обработчик прерывания INT 9H
keyboard_handler:
   push   EAX
   push   EBX
   push   ES

;Чтение скан-код нажатой клавиши из порта клавиатуры
   in    AL, 60h

;Сравнение с кодом ESC.
   cmp       AL, 01h
   je    esc_pressed

;Определение скан-кода
   cmp       AL, 39h
   ja    skip_translate

   mov       EBX, offset ascii_table
   xlatb

   mov       EBX, elem_position

;Проверка на нажатие Backspace
   cmp       AL, 8
   je    bs_pressed

   mov       ES:[EBX+0B8000h], AL
   add       dword ptr elem_position, 2
   jmp       short skip_translate

bs_pressed:
   mov       al, ' '
   sub       ebx, 2
   mov       ES:[EBX+0b8000h], AL
   mov       elem_position, EBX
skip_translate:
;Разрешить работу клавиатуры
   in    AL, 61h
   or    AL, 80h
   out       61h, AL

;Посылаем сигнал EOI контроллеру прерываний
```

```
   mov       AL, 20h
   out       20h, AL

   pop       ES
   pop       EBX
   pop       EAX
   iretd

esc_pressed:
;Разрешить работу клавиатуры, послать EOI и восстановить регистры.
   in    AL, 61h
   or    AL, 80h
   out       61h, AL
   mov       AL, 20h
   out       20h, AL
   pop       ES
   pop       EBX
   pop       EAX

;Запрещаем маскируемые и немаскируемые прерывания
   cli
   in    AL, 70h
   or    AL, 80h
   out       70h, AL

;Возврат в реальный режим
   db    0EAh
   dd    return16
   dw    sel_code16

   code32_size = $-main32
code32 ends


code16 segment para public 'CODE' use16
   assume cs:code16, ds:data16

main16:
;Очистка консоли
   mov       AX, 3
   int       10h

;Подготовка сегментных регистров
   mov       AX, data16
   mov       DS, AX

;Вывод сообщения msg_real_mode
   mov    AH, 09h
   mov    EDX, offset msg_real_mode
   int    21h
```

```
;Вычисление базы для всех используемых дескрипторов сегментов
   xor       EAX, EAX
   mov       AX, code16
   shl       EAX, 4
   mov       gdt_code16.base_l, AX
   shr       EAX, 16
   mov       gdt_code16.base_m, AL

   mov       AX, code32
   shl       EAX, 4
   mov       gdt_code32.base_l, AX
   shr       EAX, 16
   mov       gdt_code32.base_m, AL

   mov       AX, stack32
   shl       EAX, 4
   mov       gdt_stack32.base_l, AX
   shr       EAX, 16
   mov       gdt_stack32.base_m, AL

   mov       AX, data16
   shl       EAX, 4
   mov    EBP, EAX
   mov       gdt_data16.base_l, AX
   shr       EAX, 16
   mov       gdt_data16.base_m, AL

;Вычислим линейный адрес GDT
   mov       EAX, EBP
   add       EAX, offset gdt
   mov       dword ptr gdtr+2, EAX
   mov       word ptr gdtr, gdt_size-1

;Загрузим в GDTR псевдодескриптор gdtr
   lgdt   fword ptr gdtr

;Аналогично вычислим линейный адрес IDT
   mov       EAX, EBP
   add       EAX, offset idt
   mov       dword ptr idtr+2, EAX
   mov       word ptr idtr, idt_size-1

;Заполним смещение в дескрипторах прерываний
   mov       EAX, offset dummy_exc
   mov       trap1.offs_l, AX
   mov       trap2.offs_l, AX
   shr       EAX, 16
   mov       trap1.offs_h, AX
   mov       trap2.offs_h, AX
```

```
   mov       EAX, offset exc13
   mov       trap13.offs_l, AX
   shr       EAX, 16
   mov       trap13.offs_h, AX

   mov       EAX, offset timer_handler
   mov       timer.offs_l, AX
   shr       EAX, 16
   mov       timer.offs_h, AX

   mov       EAX, offset keyboard_handler
   mov       keyboard.offs_l, AX
   shr       EAX, 16
   mov       keyboard.offs_h, AX


;Сохраним маски ведомого и ведущего контроллеров прерываний
   in    AL, 21h
   mov       master, AL

   in    AL, 0A1h
   mov       slave, AL

;Перепрограммируем ведущий контроллер прерываний
   mov       AL, 11h
   out       20h, AL

   mov       AL, 20h
   out       21h, AL

   mov       Al, 4
   out       21h, AL

   mov       AL, 1
   out       21h, AL

;Запретим все маскируемые прерывания в ведущем контроллере, кроме IRQ0
(таймер) и IRQ1(клавиатура)
   mov       AL, 0FCh   ;Маска прерываний 11111100
   out       DX, AL

; Запретим все маскируемые прерывания в ведомом контроллере
   mov       DX, 0A1h
   mov       AL, 0FFh
   out       DX, AL

;Загрузим в IDTR псевдодескриптор idtr
   lidt   fword ptr idtr
```

```
;Область верхней памяти
   mov       AL, 0D1h
   out       64h, AL
   mov       AL, 0dfh
   out       60h, AL

;Запрещаем маскируемые и немаскируемые прерывания
   cli
   in    AL, 70h
   or    AL, 80h
   out       70h, AL

;Перейти в непосредственно защищенный режим установкой соответствующего
бита PE регистра CR0
   mov       EAX, CR0
   or    AL, 1
   mov       CR0, EAX

; Загрузить sel_code32 в регистр CS
; far jump в p_entry
   db 66h
   db 0eah
   dd offset main32
   dw sel_code32

;-Начиная с этой строчки будет выполняться код защищенного режима

return16:
; Переход в реальный режим

;Закрываем линию A20
   mov       AL, 0D1h
   out       64h, AL
   mov       Al, 0DDh
   out       60h,al

;Сбрасываем флаг PE системного регистра CR0
   mov       EAX, CR0
   and       AL, 0FEh
   mov       CR0, EAX

;Сбросить очередь и загрузить CS реальным числом
   db    0EAh
   dw    $+4
   dw    code16

;Восстановить регистры для работы в реальном режиме
   mov       AX, code32
   mov       ES, AX
```

```
   mov    AX, data16
   mov       DS, AX

   mov       AX, stack32
   mov       SS, AX

   mov       BX, stack32_size-1
   mov       SP, BX

;Реинициализация контроллера прерываний
   mov       AL, 11h
   out       20h, AL

   mov       AL, 8
   out       21h, AL

   mov       AL, 4
   out       21h, AL

   mov       AL, 1
   out       21h, AL

;Восстанавливаем маски контроллеров прерываний
   mov       AL, master
   out       21h, AL
   mov       AL, slave
   out       0A1h, AL

;Загружаем таблицу дескрипторов прерываний реального режима
   lidt   fword ptr idtr_r

;Разрешаем маскируемые и немаскируемые прерывания
   in    AL, 70h
   and       AL, 7Fh
   out       70h, AL
   sti

;Установка курсора
   mov       AH, 2
   xor       BX, BX
   mov       DX, 200h
   int       10h

;Вывод сообщения
   mov    AH, 9
   mov    EDX, offset msg_real_mode
   int    21h


   mov       AH, 4Ch
```

```
   int       21h

code16_size = $-main16
code16 ends

stack32    segment para stack 'stack'
   stack_start    db 100h dup(?)
stack32_size = $-stack_start
stack32    ends

end main16
```