```
.386P     ;разрешение инструкций 386
.MODEL large

;структура для описания дескрипторов сегментов
descr   STRUC
limit   dw  0
base_1  dw  0
base_2  db  0
attr    db  0
lim_atr db  0
base_3  db  0
ENDS

;макрос инициализации дескрипторов
load_descr  MACRO  des, seg_addr, seg_size
       mov  des.limit, seg_size
       xor  eax, eax
       mov  ax, seg_addr
       shl  eax, 4
       mov  des.base_1, ax
       rol  eax, 16
       mov  des.base_2, al
       ENDM

atr    MACRO  descr, bit1, bit2, bit3
;использование условных директив для проверки наличия параметров
       atr_&descr=constp or bit1 or bit2 or bit3
       ENDM

;структура для описания псевдодескриптора gdtr
point  STRUC
lim    dw  0
adr    dd  0
       ENDS

;атрибуты для описания дескрипторов сегментов
;постоянная часть байта AR для всех сегментов - биты 0, 4, 5, 6, 7
constp  equ     10010000b
;бит 1
code_r_n   equ  00010000b     ;кодовый сегмент: чтение запрещено;
code_r_y   equ  00000010b     ;кодовый сегмент: чтение разрешено
data_wm_n  equ  00000000b     ;сегмент данных: модификация запрещена
data_wm_y  equ  00000010b     ;сегмент данных: модификация разрешена;
;бит 2
code_n     equ  00000000b     ;обычный сегмент кода
code_p     equ  00000100b     ;подчиненный сегмент кода
data_      equ  00000000b     ;для сегмента данных
stack_     equ  00000000b     ;для сегмента стека
;бит 3
code_      equ  00001000b     ;сегмент кода
data_stk   equ  00000000b     ;сегмент данных или стека

stk  segment stack 'stack' use16
     db  256 dup (0)
stk  ends

;таблица глобальных дескрипторов
```

```
gdt_seg  segment para public 'data' use16
gdt_0          descr  <0, 0, 0, 0, 0, 0>    ;никогда не используется
               atr    gdt_gdt_8, data_wm_y, data_, data_stk
;ниже описываем саму gdt
gdt_gdt_8      descr  <0, 0, 0, atr_gdt_gdt_8, 0, 0>
;не используем
gdt_ldt_10     descr  <0, 0, 0, 0, 0, 0>
               atr    gdt_ds_18, data_wm_y, data_, data_stk

gdt_ds_18      descr  <0, 0, 0, atr_gdt_ds_18, 0, 0>  ;сегмент
                                                      ;данных
               atr    gdt_vbf_20, data_wm_y, data_, data_stk
gdt_es_vbf_20  descr  <0, 0, 0, atr_gdt_vbf_20, 0, 0> ;видеобуфер
               atr    gdt_ss_28, data_wm_y, stack_, data_stk
gdt_ss_28      descr  <0, 0, 0, atr_gdt_ss_28, 0, 0>  ;сегмент стека
               atr    gdt_cs_30, code_r_y, code_n, code_
gdt_cs_30      descr  <0, 0, 0, atr_gdt_cs_30, 0, 0>  ;сегмент кода
gdt_size=$-gdt_0-1  ;размер GDT минус 1
GDT_SEG ENDS

data  segment para public 'data' use16
point_gdt  point  <gdt_size,0>
hello  db  "Protected mode"
hel_size=$-hello
data_size=$-point_gdt-1
data  ends

code  segment byte public 'code' use16
;сегмент кода с 16-разрядным режимом адресации
assume  cs:code,ss:stk
main  proc
      mov  ax, stk
      mov  ss, ax
;заполняем таблицу глобальных дескрипторов
assume  ds:gdt_seg
      mov  ax, gdt_seg
      mov  ds, ax
      load_descr  gdt_gdt_8, GDT_SEG, gdt_size
      load_descr  gdt_ds_18, DATA, data_size
      load_descr  gdt_es_vbf_20, 0b800h, 3999
      load_descr  gdt_ss_28, STK, 255
      load_descr  gdt_cs_30, CODE, 0ffffh   ;code_size
assume  ds:data
      mov  ax, data
      mov  ds, ax
;загружаем gdtr
      xor  eax, eax
      mov  ax, gdt_seg
      shl  eax, 4
      mov  point_gdt.adr, eax
      lgdt point_gdt
      cli
      mov  al, 80h
      out  70h, al
;переключаемся в защищенный режим
      mov  eax, cr0
      or   al, 1
```

```
      mov  cr0, eax
;настраиваем регистры
      db  0eah ;машинный код команды jmp
      dw  offset protect ;смещение метки перехода в сегменте команд
      dw  30h  ;селектор сегмента кода в таблице GDT
protect:
;загрузить селекторы для остальных дескрипторов
      mov  ax, 18h
      mov  ds, ax
      mov  ax, 20h
      mov  es, ax
      mov  ax, 28h
      mov  ss, ax
;в защищенном режиме: выводим строку в видеобуфер
      mov  cx, hel_size  ;длина сообщения
      mov  si, offset hello   ;адрес строки сообщения
      mov  di, 1640 ;начальный адрес видеобуфера для вывода
      mov  ah, 07h  ;атрибут выводимых символов
outstr1:
      mov  al, [si]
      mov  es:[di], ax
      inc  si
      inc  di
      inc  di
      loop outstr1

;формирование дескрипторов для реального режима
assume  ds:gdt_seg
      mov  ax, 8h
      mov  ds, ax
      mov  gdt_ds_18.limit, 0ffffh
      mov  gdt_es_vbf_20.limit, 0ffffh
      mov  gdt_ss_28.limit, 0ffffh
      mov  gdt_cs_30.limit, 0ffffh
assume  ds:data
;загрузка теневых дескрипторов
      mov  ax, 18h
      mov  ds, ax
      mov  ax, 20h
      mov  es, ax
      mov  ax, 28h
      mov  ss, ax
      db   0eah
      dw   offset jump
      dw   30h
jump:
      mov  eax, cr0
      and  al, 0feh
      mov  cr0, eax
      db   0eah
      dw   offset r_mode
      dw   code
r_mode:
      mov  ax, data
      mov  ds, ax
      mov  ax, stk
      mov  ss, ax
```

```
        sti
        xor  al, al
        out  70h, al
        mov  ax, 4c00h
        int  21h
main    endp
code    ends
        end  main
```