33.3.2 Охарактеризувати послідовність дій при переключенні в захищений режим

Перевірка поточного режиму процесора

Перевірка виконується шляхом аналізу стану біта 0 регістра управління cr0.

```
mov    eax,cr0
test   al,1
jnz    MOD_error ; процесор уже в захищеному режимі
```

Крім того, необхідно розблокувати 20-й розряд адрес ПЕОМ

```
in     al,92h
or     al,2
out    92h,al
```

1.1. **Підготовка системних структур даних для роботи в захищеному режимі**

1.1.1. Формування глобальної таблиці дескрипторів (GDT).

Визначимо структуру мови Асемблера для опису дескриптора сегменту, наприклад:

```
;              ОПИС ДЕСКРИПТОРА СЕГМЕНТА
descr      struc
limit_1    dw     0  ; молодші 2 байта розміру сегменту (розряди 0-15)
base_1     dw     0  ; молодші 2 байта фізичної 32-розрядної адреси сегменту
base_2     db     0  ; 3-тій ба    йт фізичної 32-розрядної адреси сегменту
attrib     db     0  ; байт атрибутів сегменту
bt6        db     0  ; старші 4 розряди байту – атрибути сегменту,
                     ; молодші 4 розряди байту - розряди 16-19 розміру сегмету
base_3     db     0  ; 4-тий байт фізичної 32-розрядної адреси сегменту
descr      ends
```

Визначимо необхідні значення байтів атрибутів дескрипторів сегментів, формуючи біт присутності рівним 1 (**p** = 1, сегмент присутній в пам'яті), значення поля привілеїв рівним 0 (**dpl** = 00, системний рівень), наприклад:

; БАЙТИ АТРИБУТІВ

code_seg equ 10011010b; непідпорядкований сегмент кодів, дозволено читати

data_seg equ 10010010b; сегмент даних, дозволено записувати

stack_seg equ 10010110b; сегмент стека

ЗАУВАЖЕННЯ. Як сегмент стека може використовуватись сегмент даних з дозволом на записування. Різниця лише в трактовці процесором поля limit.

Визначимо глобальну таблицю дескрипторів, розміщуючи в ній дескриптори у такій рекомендованій послідовності:

1) 0 - вий дескриптор
2) дескриптор самої GDT
3) дескриптори сегментів даних (ds,es)
4) дескриптор сегмента стека (ss)
5) дескриптор сегмента основної програми (cs)

Рекомендується розміщувати GDT в окремому логічному сегменті, наприклад:

```
GDT         segment   para public 'data' use16
; Визначення таблиці глобальних дескрипторів
selector0               descr <>
deskr_gdt               descr <>
deskr_ds                descr <>
deskr_es                descr <>; video
deskr_ss                descr <>
deskr_cs                descr <>
gdt_size                equ   $-selector0-1
GDT         ends
```

Оскільки в даній лабораторній роботі прийнято системний (найвищий) рівень привілеїв і не використовується локальна таблиця дескрипторів, то зміщення дескрипторів в таблиці дескрипторів співпадають з селекторами дескрипторів, тому відпадає необхідність в спеціальному формуванні селекторів.

Сформуємо, далі, дескриптори у глобальній таблиці дескрипторів:

```
;  загальна частина для всіх дескрипторів
```

```
mov                     ax,GDT

mov                     ds,ax

assume                  ds:GDT


;Формування дескриптора сегмента кодів

mov                     deskr_cs.limit_1,code_size  ; розмір
сегмента кодів

; формування фізичної адреси сегмента кодів

xor                     eax,eax

mov                     ax,_CODE

shl                     eax,4 ; відлуння реального режиму від
TASM

                              ; TASM транслює _CODE для
реального режиму

mov                     dword ptr deskr_cs.base_1,  eax

; атрибути сегмента кодів

mov                     deskr_cs.attrib,code_seg

;-------------------------------------------------------
```

Зауважимо, що поле bt6 сформованого дескриптора залишається з 0 - вим значенням. Тобто розмір сегментів ми будемо задавати в байтах (біт гранулярності **G**=0), значення розмірів сегментів не буде більше за 64Кбайт (молодші 4 розряди bt6 дорівнюють 0), а розрядність адрес і даних по умовчанню буде дорівнювати 16 (біт розрядності **D**=0 або **B**=0), що відповідає параметру Use16 в директивах Segment.

Необхідно звернуть увагу на те, що в наведеному прикладі запис в дескриптор фізичної адреси сегменту спрощений, оскільки старший байт фізичної адреси сегменту записується не в поле base_3 дескриптора, а в поле attrib. Тому команду запису в поле attrib необхідно розміщувати **після** команди запису в поле base_1. Що ж стосується поля base_3, то його значення залишається нульовим, оскільки в даній роботі всі сегменти розміщуються в межах першого мегабайта пам'яті.

Приклад формування дескриптора для GDT :

```
; дескриптор GDT формується для доступу до GDT системними

; програмами в захищеному режимі

;-------------------------------------------------------

mov     deskr_gdt.limit_1,gdt_size
```

```
xor       eax,eax
mov      ax,GDT
shl       eax,4
mov       dword ptr deskr_gdt.base_1,eax
mov deskr_gdt.attrib,  data_seg
;-------------------------------------------------------
```

Аналогічно формуються дескриптори сегментів даних.

Доцільно для захищеного режиму сформувати окремий сегмент стека. Такий сегмент необхідно визначить без атрибута STACK директиви SEGMENT, оскільки TASM та TLINK назначать цей же сегмент як сегмент стека і в режимі реальних адрес.

```
ST_p                    segment       use16
Db                      1000 dup (0)
Top_stp          equ    $; Top_stp=1000
ST_p                    ends
```

Формування дескриптора сегмента стека має відмінність щодо поля limit - воно вказує не на максимально можливе, а на мінімально можливе зміщення в сегменті. Встановлення в якості мінімального зміщення нульового зміщення деякими процесорами сприймається не адекватно. Тому, враховуючи проведення послідуючих експериментів, рекомендується встановлювати поле limit в значення, наприклад, 10:

```
;-------------------------------------------------------
mov       deskr_ss.limit_1,  10
xor       eax,eax
mov      ax, ST
shl       eax,4
mov       dword ptr deskr_ss.base_1,eax
mov deskr_ss.attrib,  stack_seg
;-------------------------------------------------------
```

Деяку відмінність має формування дескриптора сегмента відео буфера, оскільки розмір та фізична адреса цього сегменту визначаються архітектурою ПЕОМ і фіксуються на апаратному рівні.

```
mov       deskr_es.limit_1,0ffffh;
mov      dword ptr deskr_es.base_1, 0b8000h
```

```
mov deskr_    es.attrib,  data_seg
```

1.1.2. Формування та запис вказівника глобальної таблиці дескрипторів.

До програмно доступних регістрів процесора відноситься спеціальний регістр GDTR, який повинен містить фізичну адресу глобальної таблиці дескрипторів та її розмір. Завантаження цього регістру може виконуватись в реальному режимі за допомогою команди LGDT процесора, яка повинна адресувати 6-байтний вказівник

```
;ОПИС ВКАЗІВНИКА ТАБЛИЦІ ДЕСКРИПТОРІВ

point_dt      struc

lim           dw       0

adr           dd       0

pointdt       ends
```

Визначимо вказівник для GDT (в логічному сегменті даних _DATA):

```
pgdt        pointdt   <>
```

Формування вказівника і завантаження GDT (вважаємо, що регістр DS містить адресу сегменту _DATA, а Асемблер за допомогою директиви assume про це проінформований):

```
; формування  GDT

xor         eax,eax

mov         ax,GDT

shl         eax,4

mov         pgdt.adr,eax

mov         ax,gdt_size

mov         pgdt.lim,ax

; завантаження  GDT

lgdt        pgdt
```

1.2. Переключення процесора в захищений режим

Перед переключенням у захищений режим потрібно заборонити переривання, а також зберегти логічну адресу вершини стека реального режиму, наприклад, в сегменті даних:

```
Top_real_mode       dd      ?
```

Переключення у захищений режим виконується установкою в 1 молодшого біта регістра управління CR0:

```
Cli
Mov                 word ptr Top_real_mode,sp
Mov                 word ptr Top_real_mode+2,ss
mov     eax,cr0
or      al,1
mov     cr0,eax
```

Після переходу в захищений режим необхідно в сегментні регістри завантажити селектори. Завантаження селектора в регістр CS може відбутися тільки з допомогою між сегментної команди переходу. В захищеному режимі сегментна складова логічної адреси є селектором сегмента і транслятором TASM не генерується. Тому команда між сегментної передачі управління формується "вручну" по-байтно:

```
db      0eah            ;машинний код команди jmp
dw      offset protect      ;зміщення мітки переходу в сегменті кодів
dw      offset deskr_cs     ;селектор сегмента кодів


protect:
;завантажити селектори для інших дескрипторів
; припускаємо, що RPL=0
mov     ax,offset deskr_ss
mov     ss,ax
mov     sp,Top_stp
mov     ax,offset deskr_ds
mov     ds,ax
mov     ax,offset deskr_es
mov     es,ax
```

1.3. **Виведення повідомлення на екран**

В захищеному режимі шляхом прямого запису в сторінку відео буфера вивести коротке повідомлення. Приклад виведення повідомлення mess довжиною 20 байт:

```
mov     cx,20
```

```
mov     si,offset mess
mov     di,1640     ;початкова адреса виведення
mov     ah,07h      ;атрибут символів
outmes:
mov     al,[si]
mov     es:[di],ax
inc     si
add     di,2
loop    outmes
```