

Министерство образования Республики Беларусь  
Учреждение образования «Белорусский государственный университет  
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей  
Кафедра информатики  
Дисциплина «Архитектура вычислительных систем»

**ОТЧЕТ**  
к лабораторной работе №3  
на тему:  
**«МУЛЬТИЗАДАЧНОСТЬ В ЗАЩИЩЕННОМ РЕЖИМЕ»**  
БГУИР 6-05-0612-02 67

Выполнил студент группы 353503  
КОХАН Артём Игоревич

---

(дата, подпись студента)

Проверил ассистент каф. информатики  
КАЛИНОВСКАЯ Анастасия Александровна

---

(дата, подпись преподавателя)

Минск 2025

## **1 ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ**

Изучить принципы и средства реализации мультизадачности в защищенном режиме процессора. Получить практические навыки по программированию и использованию этих средств. Необходимо написать программу, реализующую мультизадачность в защищенном режиме. Программа должна переключить процессор в защищенный режим, а затем запустить на выполнение 2-3 задачи.

## 2 ВЫПОЛНЕНИЕ РАБОТЫ

Реализация программы, выполняющей переключение задач. Запускает две задачи, Передающие управление друг другу 80 раз, задачи выводят, на экран символы ASCII с небольшой задержкой.

```
.386p
RM_seg segment para public "CODE" use16
    assume cs:RM_seg,ds:PM_seg,ss:stack_seg
start:
    push    PM_seg
    pop     ds
    mov     eax,cr0
    test    al,1
    jz      no_V86
    mov     dx,offset v86_msg
err_exit:
    push    cs
    pop     ds
    mov     ah,9
    int     21h
    mov     ah,4Ch
    int     21h

no_V86:
    mov     ax,1600h
    int     2Fh
    test    al,al
    jz      no_windows
    mov     dx,offset win_msg
    jmp     short err_exit

v86_msg db "Процессор в режиме V86 - нельзя переключиться в PM$"
win_msg db "Программа запущена под Windows - нельзя перейти в кольцо
0$"

no_windows:
    mov     ax,3
    int     10h
    mov     ax,RM_seg
    shl     eax,4
    mov     word ptr GDT_16bitCS+2,ax
    shr     eax,16
    mov     byte ptr GDT_16bitCS+4,al
    mov     ax,PM_seg
    shl     eax,4
    mov     word ptr GDT_32bitCS+2,ax
    mov     word ptr GDT_32bitSS+2,ax
    shr     eax,16
    mov     byte ptr GDT_32bitCS+4,al
    mov     byte ptr GDT_32bitSS+4,al
    xor     eax,eax
```

```

mov     ax,PM_seg
shl     eax,4
push    eax
add     eax,offset GDT
mov     dword ptr gdtr+2,eax
lgdt    fword ptr gdtr
pop     eax
push    eax
add     eax,offset TSS_0
mov     word ptr GDT_TSS0+2,ax
shr     eax,16
mov     byte ptr GDT_TSS0+4,al
pop     eax
add     eax,offset TSS_1
mov     word ptr GDT_TSS1+2,ax
shr     eax,16
mov     byte ptr GDT_TSS1+4,al
mov     al,2
out     92h,al
cli
in      al,70h
or      al,80h
out     70h,al
mov     eax,cr0
or      al,1
mov     cr0,eax
db      66h
db      0EAh
dd      offset PM_entry
dw      SEL_32bitCS

RM_return:
mov     eax,cr0
and     al,0FEh
mov     cr0,eax
db      0EAh
dw      $+4
dw      RM_seg
mov     ax,PM_seg
mov     ds,ax
mov     es,ax
mov     ax,stack_seg
mov     bx,stack_1
mov     ss,ax
mov     sp,bx
in      al,70h
and     al,07FH
out     70h,al
sti
mov     ah,4Ch
int     21h
RM_seg  ends

```

```
PM_seg segment para public "CODE" use32
```

```

        assume cs:PM_seg

GDT      label      byte
          db      8 dup(0)
GDT_flatDS      db      0FFh,0FFh,0,0,0,10010010b,11001111b,0
GDT_16bitCS     db      0FFh,0FFh,0,0,0,10011010b,0,0
GDT_32bitCS     db      0FFh,0FFh,0,0,0,10011010b,11001111b,0
GDT_32bitSS     db      0FFh,0FFh,0,0,0,10010010b,11001111b,0
GDT_TSS0        db      067h,0,0,0,0,10001001b,01000000b,0
GDT_TSS1        db      067h,0,0,0,0,10001001b,01000000b,0
gdt_size = $ - GDT
gdtr        dw      gdt_size-1
          dd      ?

SEL_flatDS     equ    001000b
SEL_16bitCS    equ    010000b
SEL_32bitCS    equ    011000b
SEL_32bitSS    equ    100000b
SEL_TSS0       equ    101000b
SEL_TSS1       equ    110000b

TSS_0          db      68h dup(0)
TSS_1          dd      0,0,0,0,0,0,0,0
          dd      offset task_1
          dd      0,0,0,0,0,stack_12,0,0,0B8140h
          dd

SEL_flatDS,SEL_32bitCS,SEL_32bitSS,SEL_flatDS,0,0
          dd      0
          dd      0

PM_entry:
        mov      ax,SEL_flatDS
        mov      ds,ax
        mov      es,ax
        mov      ax,SEL_32bitSS
        mov      ebx,stack_1
        mov      ss,ax
        mov      esp,ebx
        mov      ax,SEL_TSS0
        ltr      ax
        mov      edi,0B8000h
        mov      al,'A'

task_0:
        mov      byte ptr ds:[edi],al
        db      0EAh
        dd      0
        dw      SEL_TSS1
        add      edi,2
        inc      al
        cmp      al,'k'
        jb      task_0
        db      0EAh
        dd      offset RM_return
        dw      SEL_16bitCS

```

```

task_1:
    mov     byte ptr ds:[edi],al
    inc     al
    add     edi,2
    db      0EAh
    dd      0
    dw      SEL_TSS0
    mov     ecx,100000000h
    loop    $
    jmp     task_1

PM_seg  ends

stack_seg segment para stack "STACK"
stack_start      db      100h dup(?)
stack_1 = $ - stack_start
stack_task2      db      100h dup(?)
stack_l2 = $ - stack_start
stack_seg ends

end      start

```

Результат работы программы изображен на рисунке 1.

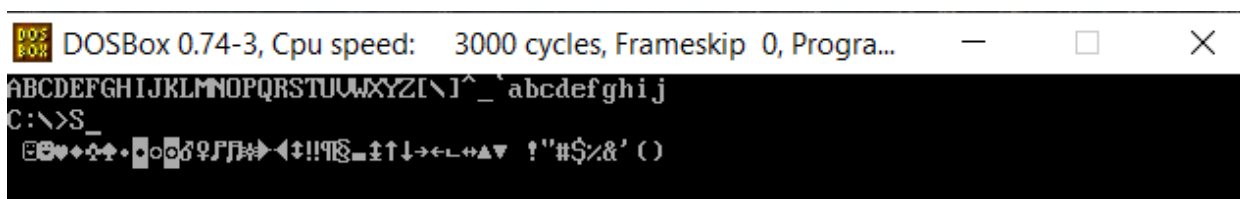


Рисунок 1 – Результат выполнения программы

## ВЫВОД

В ходе лабораторной работы были изучены принципы и средства реализации мультизадачности в защищенном режиме процессора. Были рассмотрены механизмы переключения задач, основанные на использовании дескрипторов TSS (Task State Segment) и шлюзов задач, а также команды, обеспечивающие управление задачами: FAR CALL, FAR JMP, INT и IRET.

Был изучен механизм сохранения контекста задач, который позволяет продолжить выполнение задач с того места, где они были прерваны, без потери данных. Это достигается за счет сохранения состояния регистров и указателей стека в дескрипторе TSS, который также обеспечивает изоляцию задач на уровне сегментов и страниц памяти.

Благодаря изучению битовой карты ввода/вывода было получено представление о том, как ограничить доступ к ресурсам ввода/вывода для различных задач. Это позволяет повысить уровень безопасности системы за счет предотвращения несанкционированного доступа к ресурсам.

В результате выполнения лабораторной работы были получены практические навыки программирования и управления мультизадачностью в защищенном режиме, что позволяет более эффективно использовать ресурсы процессора, обеспечивая параллельное выполнение задач.