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 практической работе №2

на тему:

**«ЗАЩИЩЕННЫЙ РЕЖИМ 32-РАЗРЯДНЫХ ПРОЦЕССОРОВ»**

БГУИР 6-05-0612-02 005

|  |
| --- |
| Выполнила студентка группы 353504  АНТОНОВА Лидия Сергеевна |
|  |
| (дата, подпись студента) |
| Проверил ассистент каф. Информатики  КАЛИНОВСКАЯ Анастасия Александровна |
|  |
| (дата, подпись преподавателя) |

Минск 2025

# 1 ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

Написать программу, переключающую процессор в защищенный режим, выводящую на экране сообщение и затем возвращающую процессор в реальный режим.

# 2 ВЫПОЛНЕНИЕ РАБОТЫ

В рамках задания была разработана загрузочная программа на ассемблере, реализующая переключение процессора из реального режима в защищённый и обратно. Для этого были настроены таблицы GDT и IDT, выполнена инициализация сегментных регистров, а также произведена установка соответствующих битов в регистре CR0. В защищённом режиме программа выводит сообщение на экран с использованием прямой записи в видеопамять, после чего по нажатию клавиши возвращает процессор в реальный режим. Работа демонстрирует базовые принципы перехода между режимами работы x86-процессора и взаимодействия с BIOS и видеопамятью. Программа протестирована в эмуляторе QEMU.

[BITS 16]

[ORG 0x7C00]

jmp start

gdt\_start:

gdt\_null: dq 0

gdt\_code: dw 0xFFFF, 0x0000, 0x9A00, 0x00CF

gdt\_data: dw 0xFFFF, 0x0000, 0x9200, 0x00CF

gdt\_end:

gdt\_descriptor:

dw gdt\_end - gdt\_start - 1

dd gdt\_start

idt\_start:

dq 0

idt\_end:

idt\_descriptor:

dw idt\_end - idt\_start - 1

dd idt\_start

start:

mov si, menu\_rm

call print\_bios

.wait\_rm:

mov ah, 0

int 16h

cmp al, 'P'

je enter\_pm

jmp .wait\_rm

enter\_pm:

cli

lgdt [gdt\_descriptor]

lidt [idt\_descriptor]

mov eax, cr0

or eax, 1

mov cr0, eax

jmp 0x08:pm\_entry

[BITS 32]

pm\_entry:

mov ax, 0x10

mov ds, ax

mov es, ax

mov ss, ax

mov esp, 0x90000

mov esi, msg\_pm

mov edi, 0xB8000

call print\_pm

mov esi, menu\_pm

call print\_pm

.wait\_pm:

call read\_key\_pm

cmp al, 0x13 ; R

je return\_rm

jmp .wait\_pm

read\_key\_pm:

.wait\_press:

in al, 0x64

test al, 1

jz .wait\_press

in al, 0x60

cmp al, 0xF0

je .wait\_press

ret

return\_rm:

mov eax, cr0

and eax, 0xFFFFFFFE

mov cr0, eax

mov ax, 0

mov ds, ax

mov es, ax

mov ss, ax

db 0xEA

dw rm\_entry

dw 0x0000

[BITS 16]

rm\_entry:

mov ax, 0

mov ds, ax

mov es, ax

mov ss, ax

mov si, msg\_back

call print\_bios

jmp start

print\_bios:

mov ah, 0x0E

.next:

lodsb

cmp al, 0

je .done

int 10h

jmp .next

.done:

ret

print\_pm:

.next:

lodsb

cmp al, 0

je .done

mov ah, 0x07

stosw

jmp .next

.done:

ret

menu\_rm db "Real Mode Menu:", 13, 10, "P - Protected Mode", 13, 10, 0

menu\_pm db 13, 10, "Protected Mode Menu:", 13, 10, "R - Return to Real Mode", 13, 10, 0

msg\_pm db "You are in Protected Mode", 13, 10, 0

msg\_back db "Returned to Real Mode", 13, 10, 0

times 510-($-$$) db 0

dw 0xAA55

# 4 АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ

В ходе выполнения программы удалось успешно реализовать переход процессора в защищённый режим и обратно. На этапе входа в защищённый режим на экране отображается сообщение *«You are in Protected Mode»*, что подтверждает корректную инициализацию сегментных регистров и работу с видеопамятью в 32-битном режиме (см. рисунок 4.1). После возврата в реальный режим на экране снова появляется BIOS-заголовок и меню *«Real Mode Menu»*, подтверждая восстановление окружения и возврат к 16-битной модели исполнения (см. рисунок 4.2). Это демонстрирует стабильность переходов и правильность настройки сегментных регистров, GDT, IDT и флагов CR0.

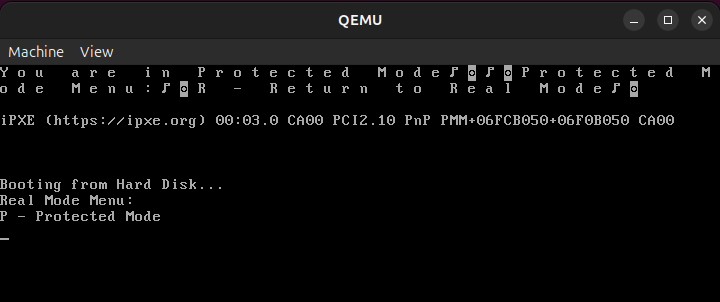


Рисунок 4.1 – Процессор в защищенном режиме

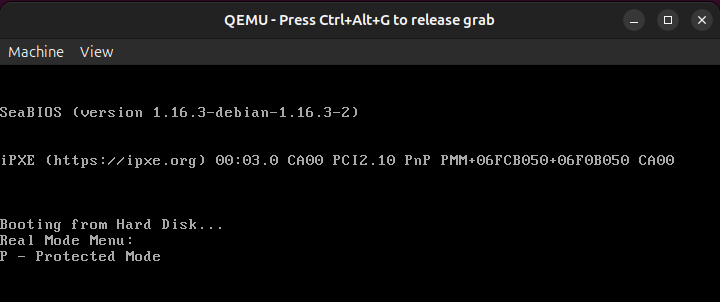


Рисунок 4.2 – Процессор в реальном режиме

# ВЫВОД

Работа с защищённым режимом процессора продемонстрировала ключевые аспекты низкоуровневого программирования на архитектуре x86. Удалось реализовать корректный переход из реального режима в защищённый, настроить таблицы дескрипторов (GDT и IDT), инициализировать сегментные регистры и управлять флагами управления в CR0. Вывод сообщений в видеопамять подтвердил успешную работу в 32-битной среде без обращения к BIOS. Возврат в реальный режим также прошёл стабильно, что свидетельствует о правильной деинициализации и восстановлении окружения. Полученные результаты подтверждают понимание механизма переключения режимов и практическое применение архитектурных особенностей процессора.