Министерство науки и высшего образования РФ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Курский государственный университет»

Кафедра программного обеспечения и администрирования информационных систем

Направление подготовки математическое обеспечение и администрирование информационных систем

Форма обучения очная

**Отчет**

**по лабораторной работе №2**

«Адресация межсегментных

переходов»

Выполнил:

студент группы 213 Файтельсон А.А.

Проверил:

доцент кафедры ПОиАИС Кривонос А.В.

Курск, 2024

**Цели работы:** изучение принципов функционирования памяти и микропроцессора компьютера при последовательном исполнении

команд программы. Приобретение навыков использования арифметических команд при написании ассемблерных программ. Приобретение навыков использования поразрядных логических команд при написании ассемблерных программ. Получение представления об особенностях обработки данных разных размерностей и режимах доступа к данным при выполнении арифметических операций.

**Листинг программы**

data1 segment ; Начало сегмента данных data1

binary\_val1 db 01011101b; Определение переменной binary\_val1 с двоичным значением 01011101

data1 ends ; Конец сегмента данных data1

data2 segment ; Начало сегмента данных data2

jmp\_data dd 0 ; Переменная jmp\_data, резервируем 4 байта (32 бита), инициализируем 0.

; Используется для хранения полного сегментно-смещенного адреса

data2 ends ; Конец сегмента данных data2

data3 segment ; Начало сегмента данных data3

binary\_val3 db 00011101b; Определение переменной binary\_val3 с двоичным значением 00011101

data3 ends ; Конец сегмента данных data3

data4 segment ; Начало сегмента данных data4

jmp\_data1 dd 0 ; Переменная jmp\_data1, резервируем 4 байта, инициализируем 0

data4 ends ; Конец сегмента данных data4

code1 segment ; Начало сегмента кода code1

nop ; Нет операции (задержка или выравнивание)

nop ; Нет операции

nop ; Нет операции

metka1: ; Метка metka1, указывает на адрес в сегменте code1

mov ax, data1 ; Загружаем адрес сегмента data1 в регистр AX

mov ds, ax ; Устанавливаем сегмент данных (DS) на data1

mov ax, 0 ; Очищаем регистр AX

mov bl, binary\_val1 ; Загружаем значение binary\_val1 в регистр BL

ror bl, 2 ; Циклический сдвиг BL вправо на 2 бита

and bl, 00000001b ; Оставляем только младший бит BL (маскирование)

mov ax, code2 ; Загружаем адрес сегмента code2 в регистр AX

push ax ; Сохраняем адрес сегмента code2 в стек

mov ax, 0 ; Очищаем регистр AX

mov ax, data4 ; Загружаем адрес сегмента data4 в AX

mov ds, ax ; Устанавливаем сегмент данных (DS) на data4

mov ax, 0 ; Очищаем регистр AX

mov [jmp\_data1+2], word ptr [code2]; Сохраняем сегментный адрес code2 в старшие 2 байта jmp\_data1

;mov [jmp\_data1], word ptr [metka2]; Сохраняем смещение метки metka2 в младшие 2 байта jmp\_data1

jmp dword ptr [jmp\_data1]; Переход по 32-битному адресу из jmp\_data1

code1 ends ; Конец сегмента кода code1

code2 segment ; Начало сегмента кода code2

;metka2: ; Метка metka2, указывает на адрес в сегменте code2

mov ax, data3 ; Загружаем адрес сегмента data3 в AX

mov ds, ax ; Устанавливаем сегмент данных (DS) на data3

mov ax, 0 ; Очищаем регистр AX

mov bh, binary\_val3 ; Загружаем значение binary\_val3 в регистр BH

ror bh, 4 ; Циклический сдвиг BH вправо на 4 бита

and bh, 00000001b ; Оставляем только младший бит BH (маскирование)

mov ax, data4 ; Загружаем адрес сегмента data4 в AX

mov ds, ax ; Устанавливаем сегмент данных (DS) на data4

mov ax, 0 ; Очищаем регистр AX

mov [jmp\_data1+2], word ptr [code3]; Сохраняем сегментный адрес code3 в старшие 2 байта jmp\_data1

mov [jmp\_data1], word ptr [correct\_work]; Сохраняем смещение метки correct\_work в младшие 2 байта jmp\_data1

jmp dword ptr [jmp\_data1]; Переход по 32-битному адресу из jmp\_data1

code2 ends ; Конец сегмента кода code2

code3 segment ; Начало сегмента кода code3

start: ; Точка входа в программу

mov ax, data1 ; Загружаем адрес сегмента data1 в AX

mov ds, ax ; Устанавливаем сегмент данных (DS) на data1

mov ax, 0 ; Очищаем регистр AX

mov al, binary\_val1 ; Загружаем значение binary\_val1 в регистр AL

shl al, 1 ; Линейный сдвиг AL влево на 1 бит

mov ax, data3 ; Загружаем адрес сегмента data3 в AX

mov ds, ax ; Устанавливаем сегмент данных (DS) на data3

mov ax, 0 ; Очищаем регистр AX

mov al, binary\_val3 ; Загружаем значение binary\_val3 в регистр AL

shr al, 2 ; Линейный сдвиг AL вправо на 2 бита

mov ax, data2 ; Загружаем адрес сегмента data2 в AX

mov ds, ax ; Устанавливаем сегмент данных (DS) на data2

mov ax, 0 ; Очищаем регистр AX

mov [jmp\_data+2], word ptr [code1]; Сохраняем сегментный адрес code1 в старшие 2 байта jmp\_data

mov [jmp\_data], word ptr [metka1]; Сохраняем смещение метки metka1 в младшие 2 байта jmp\_data

jmp dword ptr [jmp\_data]; Переход по 32-битному адресу из jmp\_data

correct\_work: ; Метка для выхода из программы

mov ax, 4c00h ; Код выхода из программы (DOS)

int 21h ; Прерывание для завершения программы

code3 ends ; Конец сегмента кода code3

end start ; Завершение программы, указатель на точку входа (start)

**Вывод:** изучили принципы функционирования памяти и микропроцессора компьютера при выполнении межсегментных переходов;