Министерство науки и высшего образования Российской Федерации

ФГАОУ ВО «Уральский федеральный университет   
имени первого Президента России Б.Н. Ельцина»

Институт новых материалов и технологий

Кафедра «Теплофизика и информатика в металлургии»

**ОТЧЕТ**

**по практической работе №1**

**по дисциплине «Операционные системы»**

Направление 09.03.02 «Информационные системы и технологии» (уровень бакалавриата)

Образовательная программа  
09.03.02/33.02 «Информационные системы и технологии» (СУОС)

Студент

группы НМТ-323901 Е.А.Кириллова

Преподаватель:

Доцент, к.т.н. А.С.Истомин

Екатеринбург

2024

**Ход работы**

1. Был подготовлен тестовый стенд на базе виртуальной машины, которая эмулирует работу реального оборудования. Виртуальная машина была настроена с использованием программного обеспечения VirtualBox (Рисунок 1),

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 1

1. Для тестирования создается загрузочный образ в формате .img, содержащий двоичный код программы. Образ подключен к виртуальной машине через контроллер floppy.
2. Мы создаем программу, которая запускается сразу после включения компьютера. Эту программу нужно написать на языке ассемблера, чтобы она могла общаться с компьютером на самом низком уровне.
3. Формат загрузочного сектора (MBR) (см. Рисунок 2)

Адрес загрузки: Загрузчик загружается BIOS в память по адресу 0x7C00.

Подпись загрузочного сектора: Последние 2 байта загрузочного сектора (512-й байт) содержат значение 0xAA55, которое BIOS использует для проверки валидности сектора.

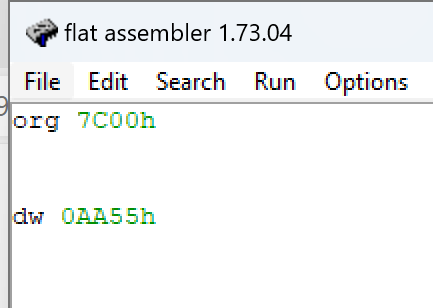


Рисунок 2

1. Определение данных name и group(см. Рисунок 3): Здесь мы создаем текстовые строки.

* db (define byte): Определяет массив байтов в памяти. Каждый символ строки сохраняется как один байт.
* Строки оканчиваются символом 0 (нулевой байт), который означает конец строки.

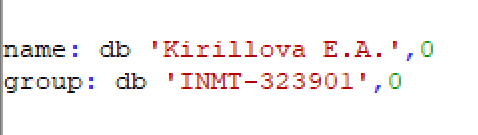


Рисунок 3

1. Текстовый режим (по умолчанию). Когда компьютер работает в стандартном текстовом режиме, каждый символ на экране отображается в виде определенного символа (например, буквы или цифры). Листинг представлен на рисунке 4.

* mov ah, 00h: Устанавливаем функцию BIOS для изменения видеорежима (int 10h).
* mov al, 13h: Указываем режим 0x13 — это графический режим с разрешением 320x200 пикселей и палитрой на 256 цветов.
* В этом режиме работаем с пикселями на экране, а не с текстом.
* int 10h: Вызываем прерывание BIOS, которое выполняет функцию изменения видеорежима.

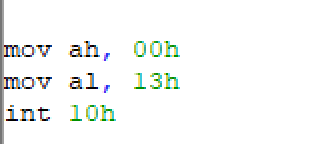


Рисунок 4

1. Вывод имени

7.1 Установка расположения (см. Рисунок 5)

* mov si, name: Загружаем в регистр si адрес строки name. Регистр si указывает, откуда брать символы для вывода.
* mov dl, 14: Устанавливаем координату X для вывода текста (положение на экране по горизонтали).
* mov dh, 10: Устанавливаем координату Y для вывода текста (положение на экране по вертикали).

Горизонтальная позиция 14 — это 14-й символ по горизонтали. В текстовом режиме, каждый символ имеет фиксированное место, обычно это 8x8 пикселей. Вертикальная позиция 10 — это 10-й ряд по вертикали, где также используется стандартная высота строки для текстового режима. В текстовом режиме, каждый символ занимают 8x8 пикселей. Поэтому координаты 14 и 10 означают:

Символ будет выведен в 14-м столбце и в 10-й строке.

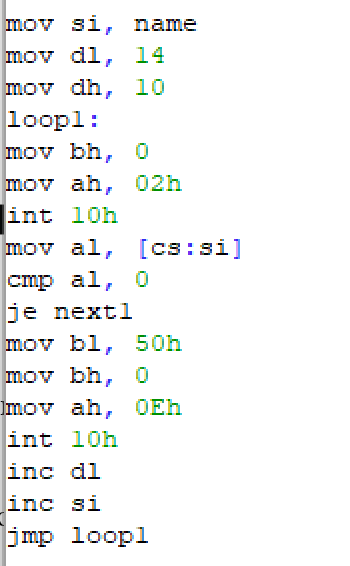


Рисунок 5

* 1. Основной цикл вывода текста. Метка loop1(см. Рисунок 6):

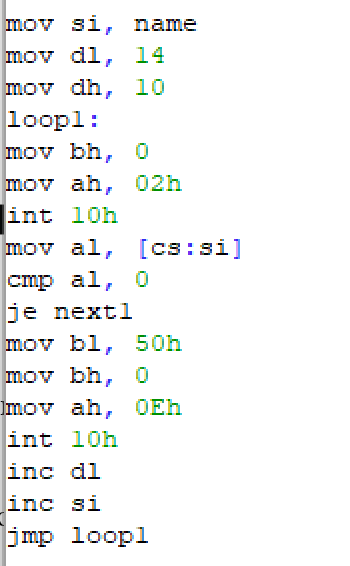


Рисунок 6

Метка loop1: Метка (или метка перехода) — это ярлык, который указывает на определенное место в коде. Когда программа встречает команду типа jmp, je, jne и другие переходы, она использует метку для того, чтобы понять, куда "прыгнуть" (перейти).

* mov bh, 0: Устанавливаем номер страницы дисплея в 0 (обычно используется только одна страница).
* mov ah, 02h: Устанавливаем функцию BIOS для позиционирования курсора.
* int 10h: Выполняем позиционирование курсора на заданные координаты (dl, dh).
* mov al, [cs:si]: Берем следующий символ из строки name (адрес строки хранится в si). Символ загружается в регистр al.
* cmp al, 0: Сравниваем символ с нулем (0). Ноль в конце строки говорит о завершении строки.
* je next1: Если символ равен 0, переход к метке next1 (значит, строка закончилась).
* mov bl, 50h: Устанавливаем цвет текста. Задает цвет текста (фиолетовый) и фоновый цвет (черный).
* mov bh, 0: Задаем номер страницы дисплея (всегда 0).
* mov ah, 0Eh: Устанавливаем функцию BIOS для вывода символа в текстовом режиме.
* int 10h: Выводим символ, который находится в регистре al.
* inc dl: Увеличиваем горизонтальную координату X, чтобы следующий символ выводился чуть правее.
* inc si: Увеличиваем указатель строки, чтобы перейти к следующему символу.
* jmp loop1: Возвращаемся к началу цикла, чтобы обработать следующий символ.

1. После завершения вывода имени (name), программа начинает выводить номер группы (group) по аналогии с именем (листинг представлен на рисунке 7).

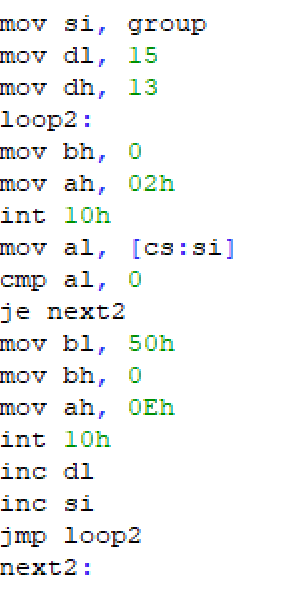


Рисунок 7

1. Рисование в графическом режиме

9.1 Подготовка (см. Рисунок 8)

* mov si, coordinates — Устанавливаем регистр si на адрес данных координат (где хранится coordinates).
* mov cx, [cs:si] — Загружаем x-координату.
* cmp cx, 0 — Если x-координата равна 0, это значит, что мы дошли до конца данных, так как последний элемент в списке — это dw 0. Это сигнализирует, что рисование завершено.
* add cx, 160 — Мы добавляем 160 к x-координате, что фактически сдвигает начало координат. Почему так? В графическом режиме 0x13 экран имеет 320 пикселей по горизонтали. Это сдвиг позволяет "центрировать" ваши координаты по отношению к центру экрана.
* add si, 2 — Переходим ко второй части пары координат (y).
* mov dx, [cs:si] — Загружаем y-координату.
* add dx, 160 — Аналогично x, добавляем 160 к y-координате, чтобы сдвигать вертикальное положение, приближая координаты к центру экрана.
* mov al, 3 — Устанавливаем цвет пикселя (в данном случае, цвет 3 — это зеленый).
* int 10h — Вызываем BIOS, чтобы нарисовать пиксель с заданными координатами.

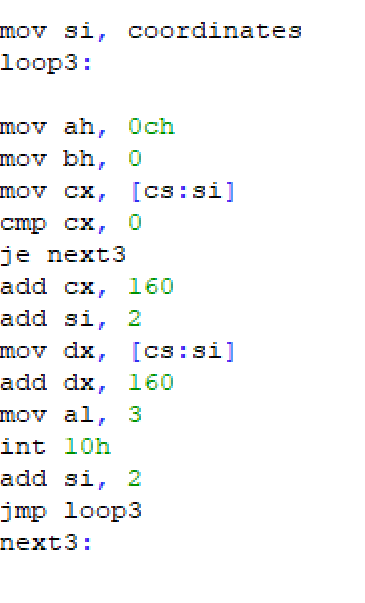


Рисунок 8

* 1. Попиксельное рисование (код представлен на рисунке 9)
* dw (define word): Каждый элемент задается двумя байтами (по 2 байта на каждую координату). Каждый dw — это пара чисел: первое — это x-координата, второе — y-координата.
* Строки, как dw 6, 1, обозначают точку с координатами (6, 1), то есть x=6, y=1.
* Последний dw 0 — это индикатор конца данных, после которого программа перестает рисовать.

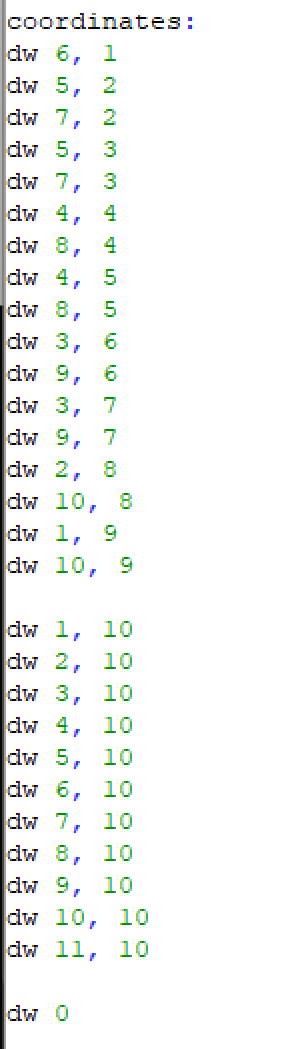


Рисунок 9

1. Готовая программа представлена на рисунке 10.



Рисунок 10

ЗАКЛЮЧЕНИЕ

В ходе выполнения практической работы мы изучили основы программирования на языке ассемблера, освоили работу с регистрами процессора, памятью, прерываниями BIOS, а также научились переключать видеорежим, выводить текст и рисовать графику на экране. Работа в реальном режиме процессора позволила понять, как происходит взаимодействие с аппаратным обеспечением на низком уровне.