Министерство образования Республики Беларусь

Учреждение Образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Лабораторная работа №4

по дисциплине «Программирование на языке ассемблера»

на тему «Создание видеоигры»

вариант №4

|  |  |  |
| --- | --- | --- |
| Выполнил:  студент гр. 250504  Солодков М. Д |  | Проверил:  Туровец Н.О. |

Минск 2023

**Цель работы:** Ознакомиться в рамках создания видеоигры с обработкой нажа-тий кнопок клавиатуры, рассмотреть прямой доступ к видеопамяти с целью формирования игрового поля и информации для пользователя.

**Теоретические сведения** Для выполнения работы требуется рассмотреть следующие элементы языка ассемблера и операционной системы:

1. *Прямой доступ к видеопамяти.*

Кроме использования прерываний DOS, описанных в лабораторной рабо-те №2, программа может выводить текст на экран с помощью пересылки данных в специальную область памяти, связанную с видеоадаптером – видеопамять. Этот вариант вывода более быстр, чем при выводе символов через прерывания, а также позволяет формировать в консоли определенные эффекты, часто не используемые в режиме вывода в позицию курсора.

В большинстве текстовых видеорежимов под видеопамять отводится спе-циальная область памяти, начинающаяся с абсолютного адреса B800h:0000h и заканчивающаяся адресом B800h:FFFFh. Все, что программа запишет в эту область памяти, будет пересылаться в память видеоадаптера и отображаться на экране.

В текстовых режимах для хранения каждого изображенного символа ис-пользуются два байта:

-- байт с ASCII-кодом символа;

-- байт атрибута символа (указывает цвет символа и фона, мигание).

Байт атрибута символа имеет следующий формат (биты):

-- 7 – символ мигает (по умолчанию) или фон яркого цвета (если его действие было переопределено прерыванием 10h).

-- 6 – 4 – цвет фона.

-- 3 – символ яркого цвета (по умолчанию) или фон мигает

Для установки требуемого программе видеорежима используется прерывание 10h (видеосервис) BIOS. Видеорежимы отличаются друг от друга разрешением (для графических) и количеством строк и столбцов (для текстовых), а также количеством возможных цветов. В данной лабораторной работе использование графических режимов видеоадаптера не требуется, поэтому в описании прерываний эта информация будет опущена.

2. *Обработка нажатия кнопок клавиатуры.*

Обработка нажатий на клавиатуру может производиться различными спо-собами:

-- с помощью прерываний ввода символов DOS;

-- с помощью прерываний ввода символов BIOS;

-- с помощью прямого доступа к буферу клавиатуры

-- с помощью доступа к портам ввода-вывода клавиатуры.

Ввод символов с помощью функций прерывания DOS 21h рассмотрен ранее в лабораторной работе №2. По сравнению с функциями DOS, прерывание BIOS 16h предоставляет больше возможностей для считывания данных и управления клавиатурой и такой доступ практически эквивалентен по производительности прямому доступу к буферу клавиатуры.

Каждой клавише на клавиатуре соответствует уникальный код, называемый скан-код. Этот код посылается клавиатурой при каждом нажатии и отпускании клавиши и обрабатывается BIOS – записывается в кольцевой буфер клавиатуры.

К буферу клавиатуры также можно обратиться напрямую – буфер находится по адресу 0000h:041Eh и занимает 16 слов, по 0000h:043Dh включи-тельно. Каждый символ хранится в буфере в виде слова, в таком же виде, как возвращает функция 01h прерывания INT 16h.

По адресу 0000h:041Ah находится адрес (ближний) по которому будет расположен следующий введенный символ (указатель на начало буфера), а по адресу 0000h:041Ch лежит адрес конца буфера. Т.к. буфер клавиатуры – за-кольцован, то если эти адреса начала и конца буфера равны, то буфер пуст.

Иногда буфер клавиатуры размещается в другой области памяти, тогда адрес его начала хранится в области данных BIOS по адресу 0480h, а конца – по адресу 0482h.

3. *Доступ к системным часам.*

Персональный компьютер содержит два устройства для управления про-

цессами:

-- часы реального времени (RTC) – имеют автономное питание, используют-ся для чтения/установки текущих даты и времени, установки будильника и для вызова прерывания IRQ8 (INT 4Ah) каждую миллисекунду;

-- системный таймер – используется одновременно для управления контрол-лером прямого доступа к памяти, для управления динамиком и как генератор импульсов, вызывающий прерывание IRQ0 (INT 8h) *18,2* раза в секунду.

Для видеоигры, создаваемой в данной лабораторной работе, указанные выше устройства лучше всего использовать на уровне функций DOS или BIOS как средство для определения текущего времени, организации задержек и фор-мирования случайных чисел.

BIOS отслеживает каждый отсчет системного таймера с помощью своего обработчика прерывания IRQ0 (INT 8h) и увеличивает на 1 значение 32-битного счетчика, который располагается в памяти по адресу 0000h:046Ch, причем при переполнении этого счетчика байт по адресу 0000h:0470h уве-личивается на 1. Программа может считывать значение этого счетчика в цикле (например, просто командой MOV) и таким образом организовывать задержки,

(например, пока ждать пока счетчик не увеличится на 1 (минимальная задержка будет равна приблизительно 55 микросекундам)). Для работы со счетчиком времени в BIOS есть функции

**Код программы (.exe)**

        JUMPS

.model small

.stack 100h

.data

offsetX dw  0

width   EQU 120

level   db  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

        db  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

        db  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

        db  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

        db  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

        db  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

        db  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

        db  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

        db  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

        db  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0

        db  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0

        db  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0

        db  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0

        db  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0

        db  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0

        db  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 0, 0, 2, 2, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0

        db  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0

        db  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0

        db  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0

        db  0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 2, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0

        db  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1

        db  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1

        db  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1

        db  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1

        db  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1

chars   db  ' ',        ' ',        ' ',        'o',        179,        48

colors  db  00110000b,  00100000b,  01100000b,  00111110b,  00111111b,  00111111b

;           0 sky,      1 grass,    2 wall,     3 coin,     4 flagpole, 5 flag

entityX     dw  20, 65

entityY     db  19, 19

entityDir   db  0,  0

entityCount db  2

curPage     db  0

row         db  0

column      db  0

playerX     dw  10

playerY     db  18

coinsCollected      db  0

maxCoins            db  5

fallState           db 0

coinsStr            db "Coins:$"

levelCompletedStr   db "Level completed!$"

levelFailedStr      db "Level failed$"

.code

start:

    mov     ah, 0h

    mov     al, 03h

    int     10h    ;установка видеорежима

    mov     ax, @data

    mov     ds, ax

    push    ds

    pop     es

lp:

    call    processKeys

    call    render

    call    processWorld

    mov ah, 86h

    mov cx, 0000h

    mov dx, 208Dh

    int 15h

    ;работа с системным таймером

    jmp     lp; бесконечный цикл

    mov ah,08h

    int 21h

    mov ax,0003h

    int 10h

    mov ah, 4Ch

    int 21h

    ;завершение программы

render proc

    mov ah, 0

    mov al, curPage

    inc ax

    mov bh, 4

    div bh

    mov curPage, ah

    ;переключение видеостраниц

    mov ah, 01h

    mov ch, 00010000b

    int 10h

    ;установка видеорежима

    mov cx, 25

    mov row, 0

rowLoop:

    push    cx

    mov     cx, 80

    mov     column, 0

colLoop:

    mov     al, ' ';установка в каждой ячейке пробела для очистки

    mov     dh, row

    mov     dl, column

    call    drawChar

    inc     column

    loop    colLoop

    pop     cx

    inc     row

    loop    rowLoop

    ;циклы которые проходятся по 25 строкам и 80 столбцам экрана.

    mov     al, 1

    mov     bl, 00000000b

    mov     dx, playerX

    mov     bh, playerY

    call    drawOnMap

    ;отображение персонажа на карте

    mov     al, 186

    inc     bh

    call    drawOnMap

    ;добавляют элемент на карту

    mov     ch, 0

    mov     cl, entityCount

    mov     al, 15

    mov     si, 0

    ;отражение игровых существ

;отображение сцены, очистка, отображение персонажей и обьектов

drawEntity:

    mov     bx, si

    mov     dx, entityX[si + bx]

    mov     bh, entityY[si]

    ;координаты х и у, массивы содержащие координаты обьектов

    mov     bl, 00000100b

    call    drawOnMap

    inc     si

    loop    drawEntity

    lea     si, coinsStr

    mov     dh, 0

    mov     dl, 0

    mov     bl, 00001110b

    call    drawString

    inc     dl

    mov     al, 'o'

    mov     bl, 00001110b

    mov     ch, 0

    mov     cl, coinsCollected

    jcxz    endDrawCollectedCoins

;отображает обьекты, текстовую строку с кол-вом монет

drawCollectedCoins:

    call    drawChar

    inc     dl

    loop    drawCollectedCoins

endDrawCollectedCoins:

    mov     cl, maxCoins

    sub     cl, coinsCollected

    mov     bl, 00000000b

    jcxz    endDrawRemainingCoins

drawRemainingCoins:

    call    drawChar

    inc     dl

    loop    drawRemainingCoins

endDrawRemainingCoins:

    mov     al, curPage

    mov     ah, 5h

    int     10h

    ret

render  endp

;вызываем разные процедуры для отображения собранных монет на экране

drawOnMap proc

    push    dx

    sub dx, offsetX

    cmp     dx, 0

    jl endDrawOnMap

    cmp dx, 79

    jg endDrawOnMap

    mov dh, bh

    call drawChar

endDrawOnMap:

    pop dx

    ret

drawOnMap endp

;проверка положения символа относительно границ экрана

drawChar    proc

    push    dx

    push    cx

    push    bx

    push    ax

    push    bx

    push    ax

    mov     bh, curPage

    mov     ah, 02h

    int     10h

    mov     bh, dh

    mov     dh, 0

    add     dx, offsetX

    call    getBlock

    pop     ax

    pop     cx

    and     cl, 00001111b

    mov     bh, 0

    cmp     chars[bx], ' '; проверка на символ фона(пробел)

    je      noBackgroundChar

    mov     al, chars[bx]

    mov     cl, 0

noBackgroundChar:

    mov     bl, colors[bx]

    or      bl, cl; обьединение цветового атрибута символа и бита цвета

    mov     bh, curPage

    mov     ah, 09h

    mov     cx, 1

    int     10h

    pop     ax

    pop     bx

    pop     cx

    pop     dx

    ret

drawChar  endp

;отображение символа с цчетом цвета на экране, установка курсора на экране

drawString  proc

    push    ax

drawStringLoop:

    cmp     byte ptr [si], '$'

    je      endDrawString

    mov     al, [si]

    call    drawChar

    inc     dl

    inc     si

    jmp     drawStringLoop

endDrawString:

    pop     ax

    ret

drawString endp

;процедура позволяет отобразить строку символов на экране, последовательно вызывая DrawChar //курсор двигается вправо

setBlock proc; установка типа блока

    push    bx

    push    ax

    mov     al, bl

    call    getBlockOffset

    mov     level[bx], al

    pop     ax

    pop     bx

    ret

setBlock endp

getBlock proc; получение типа блока

    push    ax

    push    bx

    call    getBlockOffset

    mov     bl, level[bx]

    pop     ax

    mov     bh, ah

    pop     ax

    ret

getBlock endp

getBlockOffset   proc; вычисление смещения блока

    push    ax

    push    dx

    mov     ah, 0

    mov     al, bh

    mov     bx, width

    mul     bx

    pop     dx

    mov     bx, dx

    add     bx, ax

    pop     ax

    ret

getBlockOffset endp;смещение блока в игровом поле, нужно для доступа к определенным блокам

isBlockEmpty proc

    call    getBlock

    cmp     bl, 0

    je      blockIsEmpty

    cmp     bl, 3

    je      blockIsEmpty

    cmp     bl, 4

    je      blockIsEmpty

    cmp     bl, 5

    je      blockIsEmpty

    ; если блок равен 0, 3, 4, 5 блок является пустым(блок не взаимодействует с персонажем)

    ret

blockIsEmpty:

    ret

isBlockEmpty endp

;получение значения для блоков и узнаем какие блокки явл пустыми или имеют особое значение

;обработка нажатия клавиш

processKeys      proc

    mov     ah, 01h     ;

    int     16h         ;получение нажатой клавиши

    jnz     processKey;если клавиша была нажата

    ret

processKey:

    mov     ah, 00h

    int     16h

    cmp     al, 'd'

    je      goRight

    cmp     al, 'a'

    je      goLeft

    cmp     al, ' '

    je      jump

    cmp     al, 27 ; авершение игры

    je      quit

    ret

goLeft:;проверка для клавиши влево

    mov     ax, offsetX

    cmp     playerX, ax

    jle     pkRet

    mov     bh, playerY

    mov     dx, playerX

    dec     dx

    call    isBlockEmpty

    jne     pkRet

    inc     bh

    call    isBlockEmpty

    jne     pkRet

    mov     ax, offsetX

    add     ax, 40

    cmp     playerX, ax

    jg      movePlayerLeft

    cmp     offsetX, 0

    jle     movePlayerLeft

    dec     offsetX

    dec     playerX

    ret

movePlayerLeft:

    dec     playerX

    ret

goRight:;проверка для клавиши вправо

    mov     ax, offsetX

    add     ax, 79

    cmp     playerX, ax

    jge     pkRet

    mov     bh, playerY

    mov     dx, playerX

    inc     dx

    call    isBlockEmpty

    jne     pkRet

    inc     bh

    call    isBlockEmpty

    jne     pkRet

    mov     ax, offsetX

    add     ax, 40

    cmp     playerX, ax

    jl      movePlayerRight

    mov     ax, offsetX

    add     ax, 80

    cmp     ax, width

    jge     movePlayerRight

    inc     offsetX

    inc     playerX

    ret

movePlayerRight:

    inc     playerX

    ret

jump:;проверка для клавиши прыжок

    mov     bh, playerY

    add     bh, 2

    mov     dx, playerX

    call    isBlockEmpty

    je      pkRet

    mov     cx, 4

    mov     bh, playerY

jumpLoop:

    dec     bh

    call    isBlockEmpty

    jne     pkRet

    dec     playerY

    loop    jumpLoop

    ret

quit:

    mov ah,08h

    int 21h

    mov ax,0003h

    int     10h

    mov     ah, 4Ch

    int     21h

pkRet:

    ret

processKeys endp

collectCoin proc;процедура для сбора монет в игре

    push    bx

    call    getBlockOffset

    cmp     level[bx], 3

    jne     notCoin

    mov     level[bx], 0;обнуления значения

    inc     coinsCollected

notCoin:

    pop     bx

    ret

collectCoin  endp      ;была ли собрана монета и счетчик монет

checkLevelCompleted      proc; процедура для проверки на завершения уровня

    push    bx

    call    getBlock ;текущее положение

    cmp     bl, 4

    je      levelCompleted

    cmp     bl, 5

    je      levelCompleted

    jmp     levelNotCompleted

levelCompleted:

    lea     si, levelCompletedStr[0] ;успешное завершение уровня

    mov     dh, 5

    mov     dl, 30

    mov     bl, 00000000b

    call    drawString

    mov ah,08h

    int 21h

    mov ax,0003h

    int     10h

    mov     ah, 4Ch

    int     21h

levelNotCompleted:

    pop     bx

    ret

checkLevelCompleted endp ;роверка завершен ли уровень упешно и выводит сообщения

checkLevelFailed    proc ;тоже самое только для проверки на проигранный уровень

    push    bx

    push    cx

    push    si

    cmp     bh, 24

    jge     levelFailed

    mov     ch, 0

    mov     cl, entityCount

    jcxz    levelNotFailed

    mov     si, 0

checkEntityCollision:

    push    bx

    mov     bx, si

    cmp     dx, entityX[bx + si]

    pop     bx

    jne     endCheckEntityCollision

    cmp     bh, entityY[si]

    je      levelFailed

endCheckEntityCollision:

    inc     si

    loop    checkEntityCollision

    jmp     levelNotFailed

levelFailed:

    lea     si, levelFailedStr[0]

    mov     dh, 5

    mov     dl, 30

    mov     bl, 00000000b

    call    drawString

    mov ah,08h

    int 21h

    mov ax,0003h

    int     10h

    mov     ah, 4Ch

    int     21h

    pop     si

    pop     cx

    pop     bx

    ret

levelNotFailed:

    pop     si

    pop     cx

    pop     bx

    ret

checkLevelFailed         endp

moveEntity      proc    ;процедура для обновления позиции игрока, позиции направления и движение обьектов

    push    bx

    push    dx

    mov     bx, si

    mov     dx, entityX[bx + si]

    mov     bh, entityY[si]

    inc     bh

    call    isBlockEmpty

    jne     endEntityFall

    inc     entityY[si]

endEntityFall:

    cmp     entityDir[si], 0

    je      tryMoveEntityRight

tryMoveEntityLeft:

    dec     bh

    dec     dx

    cmp     dx, 0

    jle     swapDirection

    call    isBlockEmpty

    jne     swapDirection

    mov     bx, si

    dec     entityX[bx + si]

    jmp     endMoveEntity

tryMoveEntityRight:

    dec     bh

    inc     dx

    cmp     dx, width

    jge     swapDirection

    call    isBlockEmpty

    jne     swapDirection

    mov     bx, si

    inc     entityX[bx + si]

    jmp     endMoveEntity

swapDirection:

    xor     entityDir[si], 1

endMoveEntity:

    pop     dx

    pop     bx

    ret

moveEntity  endp ; проверки и попытки на передвижения, сравнение позиций и проверки позиций на пустой блок или нет

processWorld proc ;основная процедура для игрового поля

    mov     ch, 0

    mov     cl, entityCount

    mov     si, 0

moveEntityLoop:

    call    moveEntity

    inc     si

    loop    moveEntityLoop

    mov     dx, playerX

    mov     bh, playerY

    call    collectCoin

    call    checkLevelCompleted

    call    checkLevelFailed

    inc     bh

    call    collectCoin

    call    checkLevelCompleted

    call    checkLevelFailed

    mov     dx, playerX

    mov     bh, playerY

    add     bh, 2

    call    isBlockEmpty

    jne     onGround

    cmp     fallState, 2

    jne     nextFallState

    inc     playerY

    mov     fallState, 0

    ret

nextFallState:

    inc     fallState

    ret

onGround:

    ret

processWorld endp ;управление движения обьектов, сбора монет, завершение уровня и взаимодейсвие с игровым полем

end start

**Вывод программы**

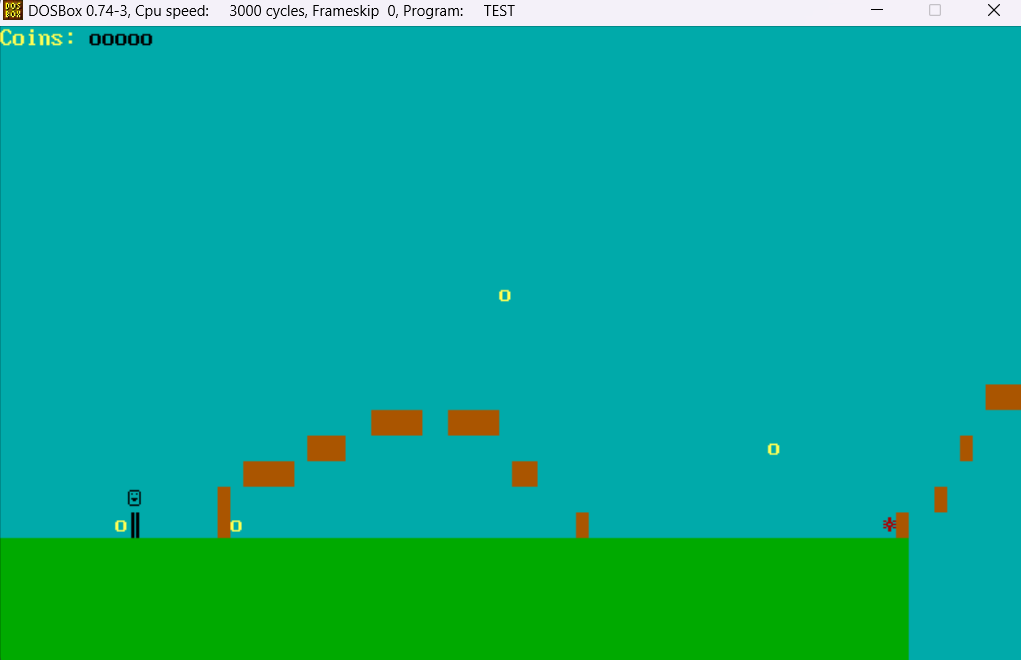


Рисунок 1 – Результат работы программы