**4 ЛАБОРАТОРНАЯ РАБОТА № 4**

**ПРОГРАММИРОВАНИЕ ВНЕШНИХ УСТРОЙСТВ**

**4.1 Цель работы**

Изучение способов организации взаимодействия процессора и внешних устройств(ВУ) в составе ЭВМ.

**4.2 Краткие теоретические сведения**

Модели внешних устройств (ВУ), используемые в описываемой системе, реализованы по единому принципу. С точки зрения процессора они представляют собой ряд программно-доступных регистров, лежащих в адресном пространстве ввода/вывода. Размер регистров ВУ совпадает с размером ячеек памяти v регистров данных процессора - шесть десятичных разрядов. Доступ к регистрам ВУ осуществляется по командам in аа, out аа, где аа - двухразрядный десятичный адрес регистра ВУ. Таким образом, общий объем адресного пространства ввода/вывода составляет 100 адресов. Следует помнить, что адресные пространства памяти и ввода/вывода в этой модели разделены. Разные ВУ содержат различное число программно-доступных регистров каждому, из которых соответствует свой адрес, причем нумерация адресов всех ВУ начинается с 0. При создании ВУ ему ставится в соответствие базовый адрес в пространстве ввода/вывода, и все адреса его регистров становятся смещениями относительно этого базового адреса. Если в системе создаются несколько ВУ, то их базовые адреса следует выбирать с учетом величины адресного пространства, занимаемого этими устройствами, исключая наложение адресов. Если ВУ способно формировать запрос на прерывание, то при создании ему ставится в соответствие вектор прерывания - десятичное число. Разным ВУ должны назначаться различные векторы прерываний. Программная модель учебной ЭВМ комплектуется набором внешних устройств, включающим: 1. контроллер клавиатуры; 2. дисплей; 3. блок таймеров; 4. тоногенератор. Связь процессора и ВУ может осуществляться в синхронном или асинхронном режиме. Синхронный режим используется для ВУ, всегда готовых к обмену. В нашей модели такими ВУ являются дисплей и тоногенератор - процессор может обращаться к этим ВУ, не анализируя их состояние (правда дисплей блокирует прием данных после ввода 128 символов, формируя флаг ошибки). Асинхронный обмен предполагает анализ процессором состояния ВУ, которое определяет готовность ВУ выдать или принять данные, или факт осуществления некоторого события, контролируемого системой. К таким устройствам в нашей модели можно отнести клавиатуру и блок таймеров. Анализ состояния ВУ может осуществляться процессором двумя способами: – в программно-управляемом режиме; – в режиме прерывания. В первом случае предполагается программное обращение процессора к регистру состояния ВУ с последующим анализом значения соответствующего разряда слова состояния. Такое обращение следует предусмотреть в программе с некоторой периодичностью, независимо от фактического наступления контролируемого события (например, нажатие клавиши). Во втором случае при возникновении контролируемого события ВУ формирует процессору запрос на прерывание программы, по которому процессор и осуществляет связь с ВУ.

**4.3 Выполнение лабораторной работы**

Задание согласно варианту 2. Программа ввода символов с клавиатуры с выводом на дисплей. (Клавиатура, дисплей, таймер). Очистка буфера клавиатуры после ввода 50 символов или каждые 10 с. В таблице 4.1 приведен код с описанием выполнение задания без прерывания.

Таблица 4.1 – выполнение задания без прерывания.

|  |  |
| --- | --- |
| Команда | Описание |
| RD #101 | Инициализация клавиатуры |
| OUT 1 |  |
| RD #010 |  |
| OUT 1 |  |
| RD #103 |  |
| OUT 1 |  |
| RD #101 | Инициализация дисплея |
| OUT 11 |  |
| RD #11 |  |
| OUT 11 |  |
| RDI #01011 | Запуск таймера |
| OUT 22 |  |
| INPUT: IN 21 | Считываем значение мс из таймера Т1 |
| SBI #10000 | Проверка, не прошло ли 10 секунд |
| JNS 22 |  |
| RD R1 | Заносим номер текущего символа в регистр R1 |
| ADD R1 |  |
| WR R1 |  |
| SUB #50 | Проверка, не вышли ли за границы буфера |
| JS 30 |  |
| RD #101 | Очистка таймера и буфера |
| OUT 1 |  |
| RDI #01000 |  |
| OUT 20 |  |
| RDI #01011 |  |
| OUT 22 |  |
| IN 0 | Считываем с клавиатуры |
| OUT 10 | Выводим на дисплей |
| JMP INPUT | Читаем следующий символ |

В таблице 4.2 приведен код с прерыванием.

Таблица 4.2 – выполнение задания с прерываниями

|  |  |
| --- | --- |
| Код | Описание |
| rd #29 | Заносим адрес обработчика |
| wr 100 | в таблицу прерываний |
| rd #000101 | Инициализируем |
| out 1 | Клавиатуру |
| rd #000011 |  |
| out 1 |  |
| rd #000103 |  |
| OUT 1 | Дисплей |
| RD #000101 |  |
| OUT 11 |  |
| RD #000011 |  |
| OUT 11 |  |
| RDI #01011 | Таймер |
| OUT 22 |  |
| EI | Разрешаем прерывания |
| MAIN: |  |
| IN 21 | Проверка таймера |
| SBI #10000 |  |
| JS MAIN |  |
| RD #101 |  |
| OUT 1 |  |
| RDI #01000 |  |
| OUT 20 |  |
| RDI #01011 |  |
| OUT 22 |  |
| JMP MAIN |  |
| INT\_\_PROC: | Обработчик прерывания |
| NOP |  |
| RD #0 | Инициализируем счетчик |
| WR R1 |  |
| INPUT: IN 0 | Считываем символ с клавиатуры |
| OUT 10 | Выводим на экран |
| RD R1 |  |
| ADD #1 |  |
| SUB #50 | Проверка на размер буфера |
| WR R1 |  |
| JNZ INPUT | Если не 50, то выводим следующий |
| RD #101 | Очищаем буфер и клавиатуру |
| OUT 1 |  |
| RDI #01000 |  |
| OUT 20 |  |
| IRET | Возврат с прерывания |

**4.4 Выводы**

В данной лабораторной работе был рассмотрен механизм прерывания. Каждое прерывание имеет свой вектор, который, как правило, располагается в памяти в виде таблицы прерываний, где располагается адрес обработчика прерывания. Сама процедура вызова обработчика похожа на обычный вызов под программы с некоторыми изменениями. Отличия полагают в занесении по мимо адреса возврата, заносятся и состояние вектора флагов.