МИНИСТЕРСТВО ПО РАЗВИТИЮ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И КОММУНИКАЦИЙ РЕСПУБЛИКИ УЗБЕКИСТАН
ТАШКЕНТСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ ИМЕНИ МУХАММАДА АЛ-ХОРАЗМИЙ

ФАКУЛЬТЕТ КОМПЬЮТЕРНОГО ИНЖИНИРИНГА
Кафедра «Компьютерных систем»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

для выполнения практических работ по дисциплине

ОРГАНИЗАЦИЯ КОМПЬЮТЕРА

ТАШКЕНТ-2022

УДК 378.147.88(073)

Авторы: Е.В. Ким, Н.С.Атаджанова, «Компьютерные системы» / ТУИТ, 94 с. Ташкент, 2022 г.

В методическом указании представлена последовательность практических работ по предмету «Организация компьютера», направленных на закрепление теоретических знаний полученных во время лекционных занятий и развитии практических навыков связанных с: изучением состава персонального компьютера на основе существующей конфигурации путём её разборки и дальнейшей сборки на тренировочном стенде, изучение логических элементов компьютера и их таблиц истинности, а также построение триггеров в программе Logisim, выполнением программы записи данных в память из регистра, изучения команды чтения/записи в память с косвенной адресацией, получением представления и навыков о соединении устройств с шиной, изучением команд загрузки регистров – в микропроцессоре ***Intel 8080***. Каждая практическая работа выполняется студентом самостоятельно.

Методическое указание предназначено для использования в учебном процессе студентами 2 курса, обучающимся по всем направлениям Ташкентского университета информационных технологий имени Мухаммада ал-Хоразмий.

Рассмотрен и одобрен на заседании Совета Ташкентского университета информационных технологий имени Мухаммада ал-Хоразмий

2022 год “_____”_____, протокол № ____

Практическая работа №1
Состав персонального компьютера

Целью работы является изучение состава персонального компьютера (ПК) на основе существующей конфигурации путем ее разборки и дальнейшей сборки на тренировочном стенде.

Теоретическая часть

Компьютер (от англ. computer – вычислитель) представляет собой программируемое электронное устройство, способное обрабатывать данные и производить вычисления, а также выполнять другие задачи манипулирования символами.

Современный компьютер – это программно-аппаратный комплекс, который состоит из аппаратной (hardware) и программной (software) частей.

Аппаратная часть построена, в основном, с использованием электронных и электромеханических элементов и устройств.

Программная часть необходима для выполнения программ, т.е. заранее заданных, четко определенных последовательностей арифметических, логических и других операций.

Архитектурой компьютера называется его описание на некотором общем уровне, включающее описание пользовательских возможностей программирования, системы команд, системы адресации, организации памяти и т.п.

Архитектура компьютера обычно определяется совокупностью ее свойств, существенных для пользователя. Основное внимание при этом уделяется структуре и функциональным возможностям компьютера, которые можно разделить на основные и дополнительные.

Основные функции определяют его назначение: обработка и хранение информации, обмен информацией с внешними объектами.

Дополнительные функции повышают эффективность выполнения основных функций: обеспечивают эффективные режимы работы, диалог с пользователем, высокую надежность и др.

Структура компьютера – это некоторая модель, устанавливающая состав, порядок и принципы взаимодействия входящих в нее компонентов.

Структура основана на общих логических принципах, позволяющих выделить в любом компьютере следующие главные устройства:

1. Память (запоминающее устройство), состоящую из перенумерованных ячеек;

2. Процессор, включающий в себя устройство управления (УУ) и арифметико-логическое устройство (АЛУ);

3. Устройство ввода;

4. Устройство вывода.

Эти устройства соединены каналами связи, по которым передается информация.

Персональным компьютером (ПК) называют сравнительно недорогой универсальный микрокомпьютер, рассчитанный на одного пользователя (рис. 1.1).

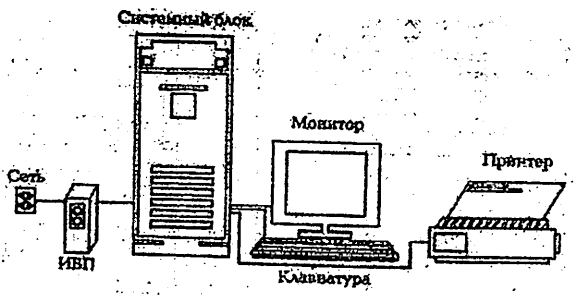


Рис. 1.1. Состав основных блоков ПК

Основные блоки ПК:

- Сеть – электрическая сеть, обычно 220 вольт.
- Системный блок – содержит основное аппаратное обеспечение (элементы) компьютера.
- Монитор – средство для визуального отображения информации.
- Клавиатура (мышь) – устройство ввода данных.
- Принтер – устройство для вывода текстовой или графической информации на твердый физический носитель.

Основные составляющие системного блока ПК

Процессор (CPU)

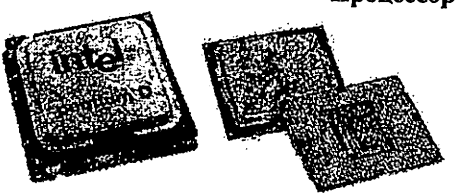


Рис. 1.2. Примеры процессоров

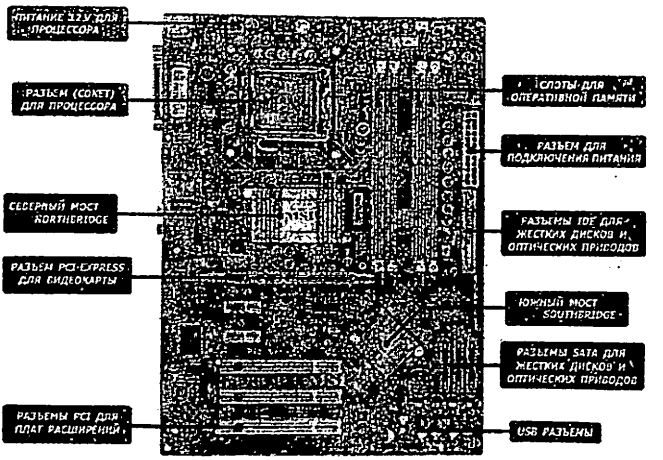
Микролроцессор – устройство, отвечающее за выполнение арифметических, логических и операций управления, записанных в машинном коде, реализованный в виде одной микросхемы. В настоящее время наиболее широкое распространение получили платы компаний Intel (рис.1.2) и AMD, которые различаются по стоимости, производительности, и количеству ядер (1, 2, 4, 6, 8).

Системная плата

Системная (материнская) плата (англ. Motherboard или mainboard – главная плата) – это сложная многослойная печатная плата, на которой устанавливаются основные компоненты персонального компьютера (рис.1.3). Как правило, материнская плата содержит разъёмы для подключения дополнительных контроллеров.

В типичный состав системной платы обычно входят:

1. Разъемы для подключения питания.
2. Разъем (socket) для подключения процессора.
3. Слоты для установки оперативной памяти.
4. Разъемы для подключения жестких дисков и оптических приводов (IDE, SATA).
5. USB – разъемы.
6. Разъемы PCI (Peripheral component interconnect – дословно, взаимосвязь периферийных устройств).
7. Разъем для подключения видеокарты (PCI – EXPRESS).
8. Северный мост (Northbridge) - обеспечивает бесперебойную передачу данных в связке «Центральный процессор - Оперативная память – Графический адаптер».



9. Южный мост (в платах Intel называется ICH - I/O Controller Hub) обеспечивает передачу данных между портами USB, оптическими приводами и жесткими дисками, также отвечает за устройства ввода: клавиатуру, мышь.

Рис. 1.3. Системная плата

Оперативная память

Оперативная память (оперативное запоминающее устройство – ОЗУ) предназначена для временного хранения данных и команд, необходимых процессору для выполнения им операций.

Виды оперативной памяти:

1.DDR SDRAM (Double Data Rate Synchronous Dynamic Random Access Memory) – синхронная динамическая память с произвольным доступом и удвоенной скоростью передачи данных, а также является самым старым видом оперативной памяти, которую можно еще сегодня купить (рис.1.4).

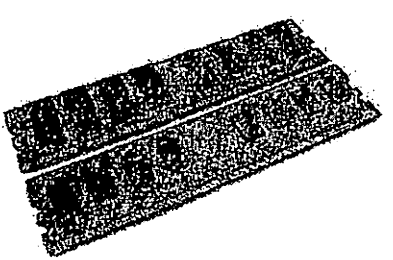
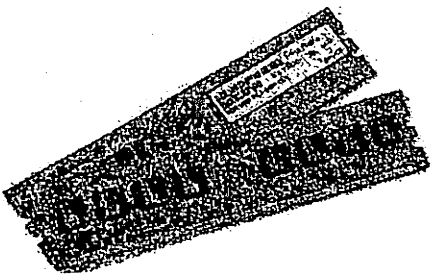


Рис. 1.4. DDR

2.DDR2 SDRAM (double-data-rate two synchronous dynamic random access memory) – синхронная динамическая память с произвольным доступом и удвоенной скоростью передачи данных, второе поколение (рис. 1.5). Тоже



устаревший вид оперативной памяти.

Рис. 1.5. DDR2

3.DDR3 SDRAM (double-data-rate three synchronous dynamic random access memory) – синхронная динамическая память с произвольным доступом и удвоенной скоростью передачи данных, третье поколение (рис. 1.6). Широко используется в современных компьютерах.

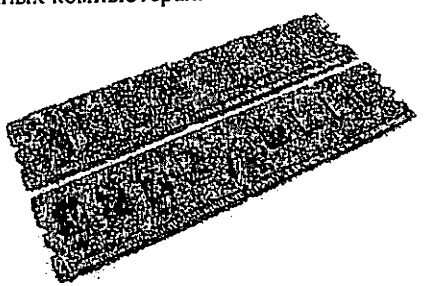


Рис. 1.6. DDR3

4.DDR4 SDRAM (double-data-rate four synchronous dynamic ran- dom access memory) – новый тип оперативной памяти, являющийся эволюционным развитием предыдущих поколений DDR.

Рис. 1.7. DDR4

Все виды памяти не совместимы друг с другом.

Видеокарта

Видеокарта (от англ. Videocard – графическая плата, графический адаптер, видеоадаптер) – устройство, преобразующее изображение, находящееся в памяти компьютера, в видеосигнал для монитора (рис. 1.8). Дополнительная видеокарта необходима лишь при условии, что ПК используется для обработки сложной графики.

Жесткий диск

Накопитель на жёстких магнитных дисках (НЖМД), жёсткий диск, винт, хард, харддиск, HDD (Hard Disk Drive) – энергонезависимое, перезаписываемое компьютерное запоминающее устройство. Является основным накопителем данных практически во всех современных компьютерах. При выборе жесткого диска необходимо учитывать его интерфейс. В настоящее время наиболее распространен интерфейс SATA (Serial ATA) (рис. 1.8), но для модернизации старых ПК необходимо использовать жесткий диск с интерфейсом IDE (Integrated Drive Electronics), либо позднее он стал называться ATA (Advanced Technology Attachment) (рис. 1.9).

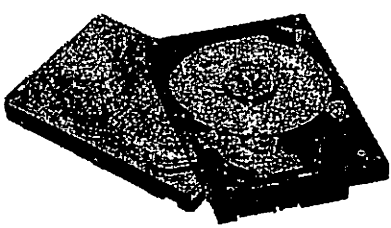


Рис. 1.8. HDD с интерфейсом SATA

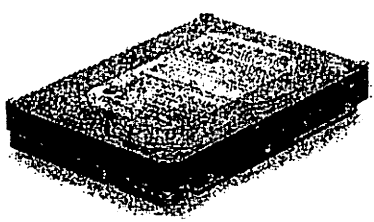
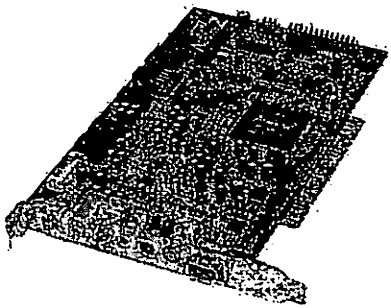


Рис.1.9. HDD с интерфейсом IDE

Звуковая карта

Звуковая плата (англ. sound card – звуковая карта или музыкальная плата) – это плата, которая позволяет работать со звуком на ком пьютере. В настоящее время звуковые карты бывают как встроенными в материнскую плату, так и



отдельными платами расширения или как внешними устройствами.

Рис. 1.10. Звуковая карта (плата расширения)

Оптический привод

Оптический привод – устройство чтения, записи дисков (рис.1.12). Можно выделить несколько основных типов данных устройств:

1. CD-ROM – данный привод способен читать только CD.

2. CD-RW – позволяет не только считывать информацию с обычных компакт-дисков, но и записывать еѐ на CD-R и CD-RW.

3. DVD-ROM - устройство, способное читать компакт-диски DVD.

4. DVD-CD-RW Combo - так называемый Combo-драйв, который сочетает в себе функции таких устройств, как DVD-ROM и CD- RW и, соответственно, может записывать диски CD-R и CD-RW, считывать как обычные CD, так и DVD.

5. DVD-RW –позволяет не только читать диски CD/DVD, но и записывать как обычные CD-R/CD-RW-носители, так и куда более ѐмкие DVD-R/DVD-RW/DVD+R/DVD+RW.

6. Blu-Ray, BD (blue ray — синий луч, и Disk - диск) — формат оптического носителя, используемый для записи и хранения цифровых данных, включая видео высокой чѐткости с повышенной плотностью.

Блок питания

Блок питания предназначен для снабжения элементов системного блока

компьютера электрической энергией (рис. 1.12). Мощность блока питания – это одно из важнейших свойств его параметров. Чем мощнее система, тем больше ее электропитание.

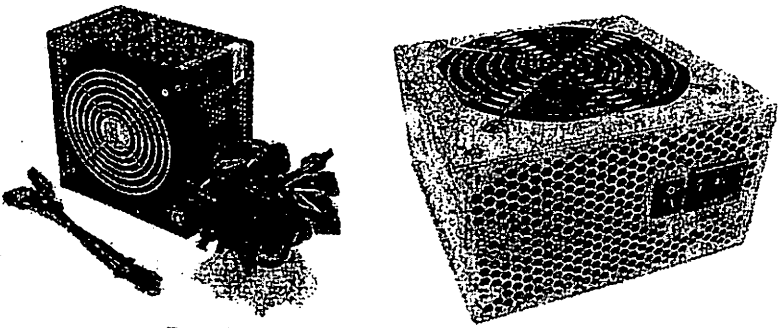


Рис. 1.12. Компьютерные блоки питания

Накопители и носители информации

Диске та – портативный магнитный носитель информации, используемый для многократной записи и хранения данных сравнительно небольшого объема

Диски оптические (CD – Compact Disk, DVD - Digital Versatile Disc). В качестве носителей оптический привод использует плоские многослойные диски диаметром 8 или 12 см.

USB Flash Drive (флешка) – носитель информации, используемый для хранения, переноса и обмена данными, резервного копирования и др. Подключается к компьютеру или иному считывающему устройству через стандартный разъм USB.

Примеры накопителей показаны на рисунке 1.13.

а) б) в)

Рис. 13. Примеры накопителей а) дискета б)CD-R в) Flash Drive

Задание

Таблица 1

Элементы системного блока

Название элемента	Фирма производитель и модель	Основные характеристики	Разъемы
Процессор	Intel/AMD и т.д.	Тактовая частота, разрядность, размер кэш-памяти, рабочее напряжение, тип памяти.	Сокет
Блок питания	Espada/FSD и т.д.	Форм-фактор, мощность, система охлаждения, габариты.	Основной разъем для питания, разъем для питания процессора (видеокарты), IDE, SATA, Molex, Floppy
Материнская плата	ASUS/Microstar/MSI и т.д.	Форм-фактор, Процессор (сокет), BIOS/UEFI, форм-фактор поддерживаемой памяти, мин и макс частота памяти, слоты расширения, аудио(чипсет звукового адаптера).	Основной разъем для питания, Разъем для питания процессора, разъем питания процессорного кулера, типы портов накопителей (IDE, SATA)

Жесткий диск	Samsung/Maxtor и т.д.	Объем, Объем кэш-памяти, Максимальная скорость передачи данных.	Интерфейс подключения.
ОЗУ	Samsung/NoName и т.д.	Форм-фактор памяти, Объем, Тактовая частота, пропускная способность.	Тип памяти
Привод оптических дисков*	Samsung и т.д.	Вид привода, скорость чтения CD/DVD/BD.	Интерфейс подключения.
Дисковод для гибких дисков*	Panasonic и т.д.	Время доступа, носитель-дискеты, скорость передачи данных, скорость вращения шпинделя, форм-фактор.	Интерфейс подключения.
Сетевая карта*	Compex, D-Link и т.д.	Скорость пере дачи данных.	Интерфейс подключения, разъемы для подключения.
Видеокарта*	ASUS/MSI и т.д.	Объем ви-деопамяти,тип памяти.	Видео разъемы, интерфейс подключения.

* при наличии в комплектации

Методика выполнения задания

1. Изучить каждый элемент ПК, записать в таблицу 1 название, фирму производителя, основные характеристики элементов и разъем для подключения.
2. Зарисовать схему соединения элементов ПК.

Требования к содержанию и оформлению отчета

Отчет по лабораторной работе должен содержать: а) титульный лист;

б) описание хода выполнения работы;

в) таблицу с описание всех элементов ПК; г) схему соединения элементов ПК;

д) заключение по выполненной работе; е) ответы на контрольные вопросы.

Контрольные вопросы

1. Что понимается под персональным компьютером?
2. Какие основные блоки содержит ПК?
3. Что называется архитектурой компьютера?

Практическая работа №2

Работа с арифметическо-логическими операциями

Цель работы: изучить позиционную систему счисления, изучение логических элементов компьютера и их таблиц истинности, а также построение триггеров в программе Logisim.

Теоретическая часть

Здесь значение каждого числа зависит от его положения в номере (позиции).

Например, 23=2*10+3

32=3*10+2

Бинарная система включена в позиционные системы, аналогичные десятичной системе (арабская система).

VII=5+1+1=7

VI=5+1=6

IV=5-1=4

Значение (число) различных чисел, используемых для представления числа в позиционной системе, является основой системы счисления R. Значения чисел от 0 до (R-1).

Как правило, требуемое число N может быть выражено в следующем порядке в системе счисления на основе «R»:

$N=a_{m-}; * P^{m-1}+a_{m-2}*P^{m-2}+\ldots+a_k*P^k+\ldots+a_1*P^1+a_0*P^0+\ldots+a_{-1}*P^{-1}+a_{-2}*P^{-2}+\ldots+a_{-s}*P^{-s}$ (1)

Следующие индексы указывают местоположение числового числа: положительные значения индексов указывают всю часть числа (m-цифры), а отрицательные - дробную часть (S-цифры). Основой двоичной системы счисления является R = 2, и для представления данных используются только два числа: 0 и 1. Существуют правила для преобразования из одной системы счисления в другую, включая (1).

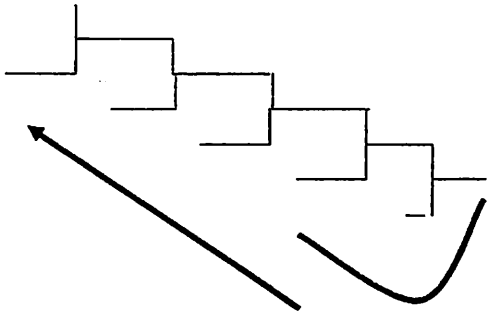
Например, $101110{,}101_2 = 1*2^5+0*2^4+1*2^3+1*2^2+1*2^1+0*2^0+1*2^{-1}+0*2^{-2}*+1*2^{-3}=46{,}625_{10}$

Таким образом, формула (1) может быть использована для преобразования числа из любой позиционной системы в десятичную систему. Сложно использовать формулу (1) для преобразования десятичной системы в любую основанную систему счисления. Для упрощения целесообразно преобразовать целую часть десятичного числа в отдельную двоичную систему, а дробную часть - в отдельную двоичную систему.

1. Целая часть, а затем разделенная часть также делятся на базу системы серийных номеров «R». Если результат последовательных делений становится равным «0», процесс останавливается. Например, 46.62510 → (2) мы конвертируем

из десятичного в двоичное.

Мы переведем целое число 46 отдельно.



Мы пишем остатки справа налево: $101110=46_{10}$

Давайте преобразуем дробную часть в двоичную систему: числа удобны для сокращения двоичных чисел.

16-разрядная система счисления часто используется в программировании.В этой системе буквы используются для обозначения числа больше 9: A=10, B=11, C=12, D=13, E=14, F=15.

Например, число 16 - F17B, в двоичной форме - 1111000101111011, в десятичной форме – 61819

Чтобы преобразовать двоичное число в 16, вам нужно разбить двоичное число с маленьких бит на тетрады.

Например, в 111010 (2) 16 появление этого числа A = 1010, 3 = 0011

Следовательно, 111010 (2) = 3A (16) будет равным.

Таблица 2.1

10-ая	16-ая	Двоичная			
		8	4	2	1
0	0	0	0	0	0
1	1	0	0	0	1
2	2	0	0	1	0
3	3	0	0	1	1
4	4	0	1	0	0
5	5	0	1	0	1
6	6	0	1	1	0
7	7	0	1	1	1
8	8	1	0	0	0
9	9	1	0	0	1
10	A	1	0	1	0

11	B	1	0	1	1
12	C	1	1	0	0
13	D	1	1	0	1
14	E	1	1	1	0
15	F	1	1	1	1

Двоичная арифметика

В микропроцессорах (МП) сложение, вычитание и умножение выполняются как простые арифметичесике действия. Во многих МП существуют команды сложения и вычитания, однако они не имеют команд умножения и деления (например, Intel 8086, 8088).

Далее приведены примеры на сложение, вычитание и умножение двоичных чисел.

Сложение

00111011 + 00101010 = 01100101 (2)

59 + 42 = 101 (10)

Вычитание

01010101 − 00111001 = 00011100 (2)

85 − 57 = 28 (10)

Умножение

1101 * 101:

1101
+ 0000
1101

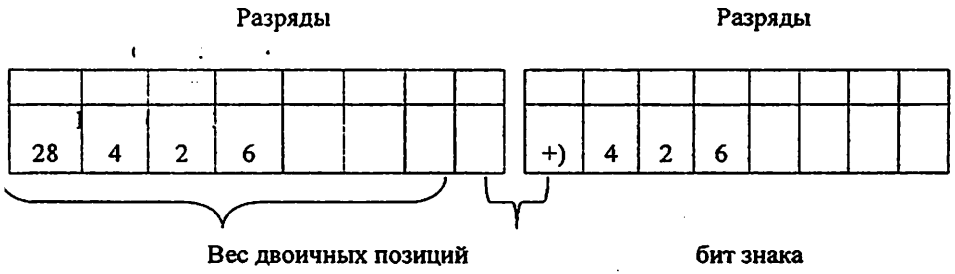
1000001 (2)

13 * 5 = 65 (10)

Дополнительный код

Обычно компьютер обрабатывает информацию в двоичном коде. Если нужно обработать знаковое число, то используется дополнительный код. Для того, чтобы

объяснить дополнительный код, изобразим регистры МП или ячейки памяти:



а) расположение двоичных позиций; б) расположение положительных чисел

в) расположение отрицательных чисел

Рис.2.1. Изображение регистра МП.

На рис.1. приведена структура образца 8-разрядного регистра. Обычно седьмой бит считается знаковым. Если число положительное пишется -"0", если отрицательное – "1".

Таблица 2.2.

Десятичные	Знаковые числа	Приложение
+127	0111 1111	Положительные числа записаны в прямом двоичном виде
...	...	
+8	0000 1000	
+7	00000111	
+6	00000110	
+5	00000101	

+4	00000100	
+3	00000011	
+2	00000010	
+1	00000001	
0	00000000	
-1	11111111	
-2	11111110	Отрицательные числа записываются в дополниельном коде (т.е. все разряды переводятся на обратный код (инверсия) и к маленькому (последнему) разряду прибавляется 1).
-3	11111101	
-4	11111100	
-5	11111011	
-6	11111010	
-7	11111001	
-8	11111000	
...	...	
-128	10000000	

Арифметика в дополнительном коде

Причина выполнения операций МП в дополнительном коде – это существование возможности выполнения операций инверсия (получение обратного кода) и инекрементация (добавление "1" к младшему разряду). Не умеет выполнять операции с прямым кодом. В его структуре имеются только сумматоры, поэтому МП для выполнения операций вычитания пользуется дополнительнм кодом.

Сложим числа "5" и "3" (в дополнительном коде).

Десятичный вид: Двоичный вид:

(+5) 00000101

\+ +00000011

(+3)

00001000

(+8) $_{(2)}=8_{(10)}$

Дополнительный код положитльныхх чисел равняется их прямому коду.

Сложим числа "+7" и "-3". В дополнительном коде данные числа будут выглядеть так: $+7_{(10)}=0000\ 0111_{(2)}$ и $-3_{(10)}=1111\ 1101_{(2)}$. Выполним сложение:

1 число	(+7)	0000 0111
	+	+
2 число	(-3)	1111 1101
	(+4)	100000100

переполнение

В связи с переполнением 8-разрядного регистра, "1" выбрасывается и получаем результат: 0000 $0100_{(2)}$ т.е, "$+4_{(10)}$"

Теперь сложим числа "+3" и "-8". В дополнительном коде данные числа будут выглядеть так: $+3_{(10)}$=0000 $0111_{(2)}$ и $-8_{(10)}$=1111 $1101_{(2)}$. Выполним сложение:

1 число	(+3)	0000 0011
	+	+
2 число	(-8)	1111 1000
	(- 5)	1111 1011

Расположение чисел на компьютере

Числа можно расположить на компьютере двумя способами:

1. Естественный вид, т.е. с постоянной точкой;
2. Нормальный вид, т.е. с плавающей запятой.

Числа в форме с **постоянной точкой** точка, которая отделяет целую часть от дробной всегда стоит на месте. Например, в десятичной с.с. если целой части отделено 5 разрядов, дробной части отделено 5 разрядов, то числа в данном разряде записываются так:

+00721,35500
+00000,00325
-10211,20260

Эта форма записи чисел очень простая и естественная, но диапозон чисел маленький. Поэтому данная форма не подходит для выполнения вычислений. Например, при умножении чисел, в целой части может произойти переполнени, далее продолжить умножение смысла не будет. Данная форма записи чисел в современных компьютерах используется для обработки целых чисел в качестве

дополнительной формы. Числа с такой формой записи в памяти компьютера хранится тремя способами:

1. Половина слова – обычно 16 бит (2 байта);
2. Целое слово – 32 бит (4 байта);
3. Двойное слово – 64 бит (8 байт).

Если число с постоянной точкой отрицательное, то на разрядную сетку записывается в виде дополнительного кода.

Числа с **плавающей запятой** записываются с помощью мантиссы и порядка. В данном случае абсолютное значение мантиссы должно быть меньше 1, а порядок – целое число. Общий вид числа:

$$N=\pm M*P^{\pm r},$$

где M – мантисса числа ($|M|<1$);

r- порядок числа (целое число);

p- основа системы счисления.

Например, выщеуказанные числа в нормализованном виде записываются так:

$$+00721355*10^{3}$$

$$+325*10^{-3}$$

$$-102112026*10^{5}$$

Нормальная форма используется для обеспечения большого диапазона чисел и в современных компьютерах считается основной формой записи чисел. Например, если p=2, m=22 и r=10, диапазон числа будет от 10^{-300} до 10^{300}.

Нужно сказать, что все числа с плавающей запятой на памяти компьютера хранятся в нормализованном виде. Значит, для двоичных чисел имеет место выражение $0,5< |M|<1$.

Разрядная сетка компьютера для такого вида записи имеет следующую структуру:

- Нулевой разряд – знак числа (0-положительное, 1-отрицательное);
- От 1 до 7-разряда- порядок числа записывается в прямом коде, пустые разряды заполняются нулями. В первом разряде принадлежащей порядку, записывается знак порядка числа;
- От 8 до 31 (или 63) слева направо записывается мантисса, пустые разряды заполняются нулями.

Задания:

1. 873,9X (10) → (2)
2. 11011011X (2) → (10)
3. 0.101100X (2) → (10) и (16)
4. 110111,01X (2) → (10) и (16)

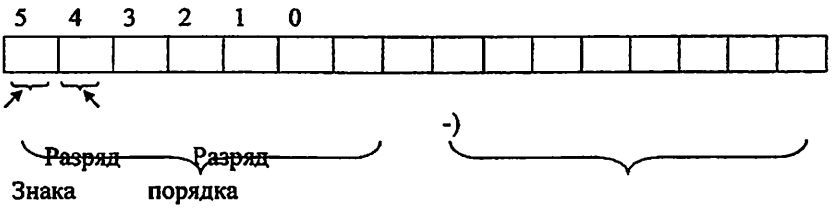
5. Найдите эквиваленты в десятичном виде следующих чисел.

а) 11111011; б)00001111; в) 10001111; г)01110111.

6. N=101X, N=-111X – приведите обратный нормализованный код указанных чисел.

7. Приведите дополнительный код вышеуказанных чисел, а также запишите их в 16-разрядном формате на памяти компьютера в форме плавающей запятой (Здесь, мантиссе-8 разряд, порядку-8 разряд распределена).

Разряды



Объяснение: Вместо буквы X вставьте свой номер из списка. В нужных местах переведите данное число на указанную систему счисления.

Цифровая логика компьютера.

Любая информация в компьютере или в другом устройстве вычислительной техники представляется в виде двоичного цифрового сигнала. Все виды информации (текстовая, графическая, звуковая, видео-) кодируются в последовательности нулей и единиц.

Алгебра логики

Алгебра логики появилась в середине XIX в. в трудах английского математика Джорджа Буля.

Логическое высказывание – это любое повествовательное предложение, в отношении которого можно однозначно сказать, истинного оно или ложно. При этом не всякое предложение является логическим высказыванием. Например, «Хороший студент» не является логическим высказыванием, так как невозможно судить об его истинности или ложности, а высказывание «Иванов – хороший студент» яв ляется логическим высказыванием.

Употребляемые слова и словосочетания «*не*», «*и*», «*или*», «*если, то*», «*тогда и только тогда*» позволяют из уже заданных высказы ваний строить новые. Такие

слова и словосочетания называются *ло гическими связками*. При этом высказывания, образованные из других высказываний с помощью логических связок, называются *составны-ми*. Высказывание, не являющееся составным, *называется элементарным*.

Для того чтобы обращаться к логическим высказываниям им назначаются имена. Например, через А обозначим высказывание

«Петя был во Франции», а через В высказывание «Петя был в Ита- лии». Тогда составное высказывание примет вид - «Петя был и во Франции, и в Италии», далее можно записать кратко: «А И В». Здесь И – логическая связка, А,В – логические переменные, которые могут принимать только два значения: истинна и ложь, обозначаемые 1 и 0.

В алгебре логики высказывания могут принимать лишь два значения: истинна – 1 и ложь – 0.

С логическими высказываниями можно производить следующиеоперации:

1. Операция отрицания (НЕ).
2. Операция конъюнкции (И).
3. Операция дизъюнкции (ИЛИ).
4. Операция импликации (ЕСЛИ...ТО).
5. Операция эквиваленции (ТОГДА, И ТОЛЬКО ТОГДА).

Порядок выполнения логических операций задается круглыми скобками, но для уменьшения числа скобок договорились считать,что сначала выполняется операция отрицание (НЕ), затем конъюнкции (И), затем дизъюнкции (ИЛИ) и в последнюю очередь импликации. Это называется приоритетом операций.

Логические основы устройства компьютера

Логический элемент компьютера – это часть электронной логической схемы, которая реализует элементарную логическую функцию.

Логическими элементами компьютеров являются электронные схемы И, ИЛИ, НЕ, И-НЕ, ИЛИ-НЕ и др. (называемые также вентилями), а также триггер.

Триггер - это устройство позволяющее запоминать, хранить и считывать информацию (каждый триггер может хранить 1 бит информации).

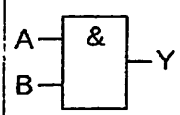
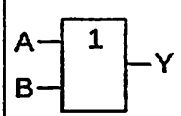
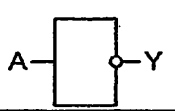
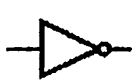
С помощью этих схем можно реализовать любую логическую функцию, описывающую работу устройств компьютера. Обычно у элементов бывает от 2 до 8 входов и один или два выхода. Чтобы представить два логических состояния 1 и 0, соответствующие им

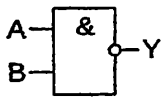
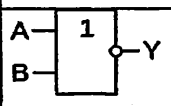
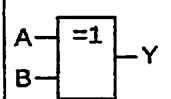
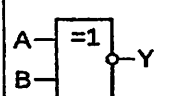
входные и выходные сигналы имеют один из двух установленныхуровней напряжения, например 5 и 0 В. Высокий уровень обычно соответствует значению «истинна» (1), а низкий – значению «ложь» (0).Каждый логический элемент имеет свое условное обозначение, которое выражает его логическую функцию, но не указывает на то,какая электронная схема в нем реализована. Это упрощает запись и понимание сложных схем.

Работу логических элементов описывают с помощью таблиц истинности. Основные структурные схемы логических элементов компьютера и их таблицы истинности, представлены в таблице 3.2.1.

Таблица 3.2.1.

Структурные схемы логических элементов компьютера

Условное обозначение	Структурная схема(отечественное обозначение)	Структурная схема(зарубежное обозначение)	Таблица истинности
И			A B Y (A&B) 0 0 0 0 1 0 1 0 0 1 1 1
ИЛИ			A B Y (A∨B) 0 0 0 0 1 1 1 0 1 1 1 1
НЕ			A Y (Ā) 0 1 1 0

<table>
<tr><td>И-НЕ</td><td> 
</td><td>  </td><td>A B Y
0 0 1
0 1 1
1 0 1
1 1 0</td></tr>
<tr><td>ИЛИ-НЕ</td><td> 
</td><td>  </td><td>A B Y
0 0 1
0 1 0
1 0 0
1 1 0</td></tr>
<tr><td>Исключающее ИЛИ</td><td> 
</td><td>  </td><td>A B Y
0 0 0
0 1 1
1 0 1
1 1 0</td></tr>
<tr><td>Исключающее ИЛИ-НЕ</td><td> 
</td><td>  </td><td>A B Y
0 0 1
0 1 0
1 0 0
1 1 1</td></tr>
</table>

Триггер

Важнейшей структурной единицей оперативной памяти компьютера, а также внутренних регистров процессора, является триггер.

Триггер можно построить из двух логических элементов ИЛИ и двух элементов НЕ, схема триггера показана на рисунке 3.3.1. Им соответствует таблица истинности 3.3.1. Также схему триггера можно реализовать на двух элементах ИЛИ-НЕ, либо И-НЕ, заменив элементы ИЛИ и НЕ на соответствующие.

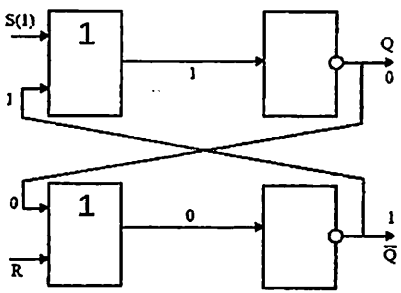


Рис. 3.3.1. Схема RS-триггера

Таблица 3.3.1.
Таблица истинности

S	R	Q	Q
0	0	Хранение бита	
0	1	1	0
1	0	0	1

Тип триггер на рисунке 3.3.1. соответствует – RS-триггеру (от англ. reset – сброс, set – установка). Он имеет два симметричных входа R и S и два симметричных выхода Q и Q (от англ. Quit = выход). В обычном состоянии на входы R и S триггера подан сигнал 0 и триггер хранит 0. Для записи 1 (Q=1) на вход S подается сигнал 1. После того как сигнал на входе S исчезнет, состоянии сохранится, т.е. будет состояние хранение бита информации.

Для того чтобы сбросить информацию подается сигнал 1 на вход

R, после чего триггер возвращается к исходному (нулевому) состоянию (Q=0,Q=1). Если на входы S и R подан сигнал 0, то состояние не меняется. Подача сигнала 1 на оба входа S и R приводит к неординарному результату, поэтому эта комбинация входных сигналов запрещена.

Также существуют другие типы триггеров, такие как T-

триггеры, D-триггеры, JK-триггеры. Также принято рассматривать асинхронные и синхронные триггеры. На рисунке 3.3.1. представлен асинхронный RS-триггер. Синхронный триггер характеризуется дополнительным синхросигналом на входе.

Составление логических схем в программе Logisim

Logisim – это инструмент, позволяющий разрабатывать и моделировать электрические (логические) схемы, используя графический интерфейс пользователя. На рисунке 3.4.1. представлен интерфейсLogisim.

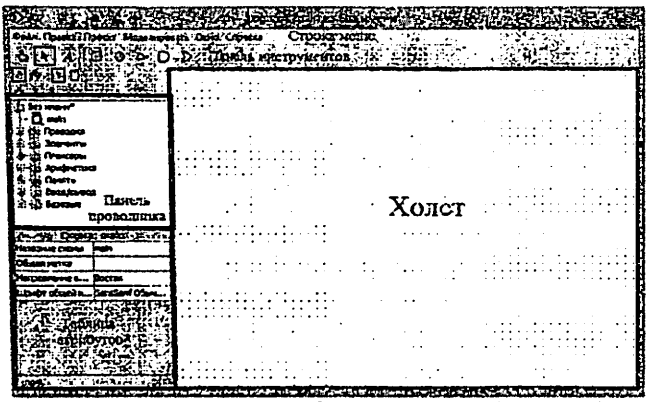


Рис. 3.4.1. Графический интерфейс Logisim

Рабочая область Logisim разделена на три части: холст, панель проводника и таблица атрибутов. Холст – это место рисования схем. Панель инструментов содержит инструменты (элементы) для достижения поставленной цели. Таблица атрибутов содержит свойства выбранного элемента.

Для реализации логических схем в Logisim существуют элементы И, ИЛИ, НЕ, И-НЕ, ИЛИ-НЕ, исключающее ИЛИ, исключающее ИЛИ-НЕ, доступ к которым можно получить в панели проводника в разделе «Элементы» (рис. 3.4.2).

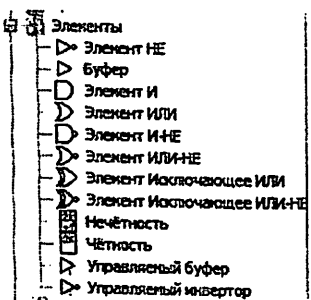


Рис. 3.4.2. Раздел панели проводника «Элементы»

Помимо логических элементов необходимы генераторы сигналов, которые представлены в Logisim в виде тактов (

) вразделе «Проводка».

Контакт используется как для подачи сигналов (0 или 1), так и для отображения значений выходов элементов. В таблице атрибутов контакта настраиваются его свойства (рис. 3.4.3).

Выделение: Контакт	
Направление	Восток
Выход?	Нет
Биты данных	1
Три состояния?	Нет
Обращение с плав...	Не менять
Метка	
Направление метки	Запад
Шрифт метки	SansSerif Обычный...

Рис. 3.4.3. Таблица атрибутов контакта

Основные свойства контакта определяются при его помещении на холст, где *направление* – это сторона выхода или входа контакта, *выход* – значение является ли данный контакт выходом или входом, *биты данных* – определяет количество значений, которое может принимать контакт (1 бит – это 2 значения: 0 и 1), *три состояния* – указывает, что у контакта может быть 3 состояния (0, 1 и не работает).

Логические элементы также имеют свойства, которые можнонастроить в зависимости от решаемой задачи. Существуют такие свойства как направление, биты данных, количество входов, размер элемента и возможность инвертировать значения входов (рис. 3.4.4.).

Выделение: Элемент ИЛИ-НЕ	
Направление	Восток
Биты данных	1
Размер элемента	Средний
Выходное значение	0/1
Метка	
Шрифт метки	SansSerif Обычный 12
Инвертировать 1 (Сверху)	Нет
Инвертировать 2 (Снизу)	Нет

Рис. 3.4.4. Таблица атрибутов элемента ИЛИ-НЕ

Пример построения асинхронного RS-триггера на 2 элементах ИЛИ-НЕ показан на рисунке 3.4.5.

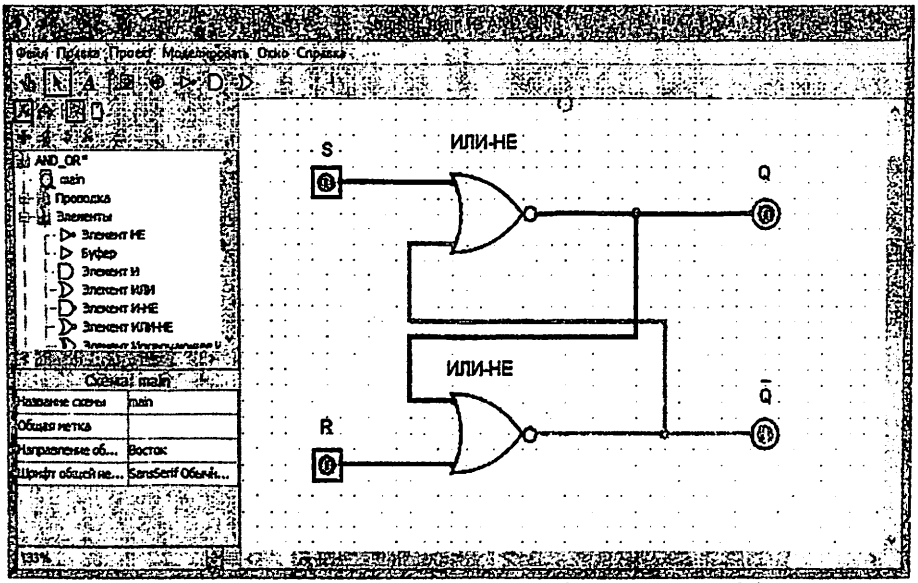


Рис. 3.4.5. Асинхронный RS-триггер в Logisim

Здесь для реализации асинхронного RS-триггера используется 2 элемента ИЛИ-НЕ, и 4 контакта, причем 2 контакта на входы S и R и2 контакта на выходы Q и $\bar{Q}$.

4. Задание на лабораторную работу

Составить логическую схему для высказывания и RS-триггера. Вариант задания выдается преподавателем.

5. Методика выполнения задания

1. Изучить теоретическую часть.
2. Составить логическую схему для высказываний, указанных в таблице 1 в программе Logisim.

Таблица 1

Варианты заданий

N	*Высказывание*
1	□A & B□ □ C & □A □ C□
2	□A & B□ □ C & □A&C□
3	□A & B□&C □ □A □ C□
4	C □ □A □ B□ & □A □ C□
5	□A & *C*□ □ C & □*B* □ C□
6	□A & B□ □ C & □A □ *B*□
7	(□A □ B□&C) & □A □ C□
8	□A & B□ □ C & □A □ C□

1. F=A∨B& $\bar{C}$, если A=1, B=1, C=0.
2. F = ¬(A∨B&C), если A=1, B=1, C=0.
3. F =¬A∨B&C, если A=1, B=1, C=0.
4. F =(A∨B)&(C∨B), если A=1, B=1, C=0.
5. F = ¬(A&B&C), если A=1, B=1, C=0.
6. F=B& $\bar{A}$∨ $\bar{B}$&A, если A=0, B=1.
7. F=¬(A&B&C)∨(B&C∨ $\bar{A}$), если A=1, B=1, C=0.
8. F = A ∨ B&$\bar{C}$, если A=1, B=1, C=0
9. F = $\bar{A}$ ∨ B&C, если A=1, B=1, C=0
10. F = A&B ∨ $\bar{C}$, если A=1, B=1, C=0
11. F = A&B ∨ $\bar{C}$, если A=1, B=1, C=0
12. F = A$\bar{\&}$B ∨ C, если A=1, B=1, C=0
13. F = ($\bar{A}$&$\bar{B}$) ∨ ($\bar{B}$&$\bar{C}$), если A=1, B=1, C=0
14. F = ($\bar{A}$∨$\bar{B}$)&($\bar{A}$&$\bar{C}$), если A=1, B=1, C=0
15. F = ($\bar{A}$&$\bar{B}$)&($\bar{A}$∨$\bar{C}$), если A=1, B=1, C=0
16. F = ($\bar{A}$&$\bar{B}$) ∨ ($\bar{C}$&$\bar{A}$), если A=1, B=1, C=0
17. F = ($\bar{A}$∨$\bar{C}$)&($\bar{B}$&$\bar{C}$), если A=1, B=1, C=0
18. F = $\bar{A}$∨B&A, если A=1, B=0
19. F = A&B∨($\bar{C}$∨A), если A=1, B=1, C=0
20. F = $\bar{A}$ & (B ⇒ C)если A=1, B=1, C=0
21. F = $\bar{A}$ &$\bar{B}$∨ A &B, если A=1, B=1, C=0
22. F = (A ⇒ B) &C, если A=1, B=1, C=0

23. $F = (A \Rightarrow \bar{B}) \& C$, если A=1, B=1, C=0
24. $F = (A \Rightarrow \bar{B}) \& \bar{C}$, если A=1, B=1, C=0
25. $F = (\bar{A} \& \bar{B}) \vee (A \& B)$, если A=1, B=1, C=0
26. $F = (A \& B) \vee (\bar{A} \& C)$, если A=1, B=1, C=0
27. $F = (\bar{A} \vee \bar{B}) \& (\bar{B} \& \bar{C})$, если A=1, B=1, C=0
28. $F = (\bar{A} \& \bar{C}) \& (B \vee \bar{C})$, если A=1, B=1, C=0
29. $F = (\bar{A} \vee \bar{C}) \vee \neg (\bar{B} \vee \bar{C})$, если A=1, B=1, C=0
30. $F = \neg (\bar{A} \& B) \vee (\bar{B} \vee \bar{C})$, если A=1, B=1, C=0
31. $F = \neg (A \vee C) \& \neg (\bar{B} \& \bar{C})$, если A=1, B=1, C=0
32. $F = (A \vee B) \& (B \vee \bar{C})$, если $\bar{A}$=1, B=1, C=0
33. $F = (\bar{A} \vee \bar{C}) \& \bar{C} \& \bar{B}$, если A=1, B=1, C=0
34. $F = \bar{A} \vee (\bar{B} \& \bar{C})$, если A=1, B=1, C=0
35. $F = \bar{A} \& (\bar{B} \& \bar{C})$, если $\overline{A}$=1, B=1, C=0

3. Заполнить таблицу 2, указывая значения A, B, C, и также получаемое значение Y на выходе, используя программу Logisim.

Таблица 2

Таблица истинности

A	B	C	Y
0	0	0	y_1
0	0	1	y_2
0	1	0	y_3
...	...	...	...

4. Составить схему асинхронного RS-триггера на основе элементов указанных в таблице 3. Удостовериться в возможности хранения бита информации. Составить таблицу истинности.

Таблица 3

Варианты элементов

№	Элементы
1	2шт «ИЛИ»
2	2шт «НЕ»
3	
4	2шт «И»*
5	2шт «НЕ»
6	

7	2шт «И-НЕ»*
8	

*при составлении схемы с элементом «И» входы R и S должны быть инвертированы.

6. Требования к содержанию и оформлению отчета

Отчет по лабораторной работе должен содержать:а) титульный лист;

б) описание хода выполнения работы;в) логическая схема высказывания;

г) схема асинхронного RSтриггера;д) таблицы истинности;

е) заключение по выполненной работе;ж) ответы на контрольные вопросы.

Контрольные вопросы

1. Что такое логическое высказывание?
2. Какие операции можно производить с высказываниями?
3. Что такое триггер?
4. Какие типы триггеров существуют?
5. Какие структурные схемы у элементов И, НЕ, ИЛИ? Привести таблицы истинности.

Практическая работа №3

Изучение выполнение команд и управления операциями

Цель работы: Изучить возможности командной оболочки и способы применения основных команд и утилит ОС Windows XP при работе с файлами и дисками.

Краткие теоретические сведения

Командная оболочка — это отдельный программный продукт, который обеспечивает прямую связь между пользователем и операционной системой (ОС). Текстовый пользовательский интерфейс в виде командной строки предоставляет среду, в которой выполняются команды, программы и служебные утилиты с текстовым интерфейсом. В командной оболочке и результат выполнения утилит и программ отображается на экране в виде, сходном с командным интерпретатором

Command.com MS-DOS. Командная оболочка ОС Windows XP использует интерпретатор команд **Cmd.exe**, который осуществляет перевод введенной команды в понятный ОС вид, загружает приложения (утилиты) и управляет потоками данных между ними.

Имеется возможность использовать командную оболочку для создания и редактирования пакетных файлов (также называемых сценариями), что позволяет автоматизировать выполнение обычных задач. Например, можно использовать сценарии для автоматизации управления учетными записями пользователей и ежедневной архивацией в нерабочие часы. Также можно использовать сервер сценариев ОС Windows XP, **Cscript.exe**, для выполнения сложных сценариев посредством командной оболочки.

Выполнение операций с помощью пакетных файлов является более эффективным, чем с помощью текстового интерфейса пользователя. Командные или пакетные файлы принимают все команды, доступные из командной строки. Возможность, ориентированная непосредственно на пользователя, позволяет настроить окно командной строки для облегчения визуализации и просмотра, а также для усиления контроля текущего выполнения приложений. Чтобы реализовать эту возможность, необходимо для примера выполнить следующие действия:

1. Загрузите командную оболочку:

- нажмите **Пуск | Выполнить**,
- наберите в появившемся окне **Cmd.exe** (или просто **cmd**),
- нажмите **Enter** для ввода.

2. Кликните правой кнопкой «Мыши» в верхней части появившегося командного окна и выберите команду **Свойства** из контекстного меню командной оболочки.

3. В диалоговом окне **Свойства** выберите вкладку **Общие.**

4. В области **Запоминание команд** вкладки **Общие** выберите или введите значение

999 в поле **Размер буфера**, а затем выберите или введите значение **5** в поле **Количество буферов.**

5. В области **Редактирование** установите флажки **Выделение мышью** и **Быстрая вставка.**

6. В диалоговом окне **Свойства** выберите вкладку **Расположение.**

7. В области **Размер буфера экрана** вкладки **Расположение** введите или выберите значение **2500** в поле **Высота.**

8. На вкладке **Расположение** выполните следующие действия:

- в области **Размер буфера экрана** увеличьте значение параметра **Ширина**,

- в области **Размер окна** увеличьте значение параметра **Высота**,

- в области **Размер окна** увеличьте значение параметра **Ширина**,

- снимите флажок **Автоматический выбор**, а затем в области **Положение окна** измените значения полей **Левый** и **Верхний край**,

9. В диалоговом окне **Свойства** выберите вкладку **Шрифт.**

10. На вкладке **Шрифт** выполните следующие действия:

- в области **Шрифт** выберите необходимый шрифт,

- в области **Размер** выберите необходимый размер шрифта.

11. В диалоговом окне **Свойства** выберите вкладку **Цвета.**

12. На вкладке **Цвета** выполните следующие действия:

- установите флажок **Текст на экране** и выберите цвет текста, кликнув манипулятором по соответствующему полю,

-установите флажок **Фон текста** и выберите цвет фона, кликнув манипулятором по соответствующему полю,

13. Обратите внимание на то, как влияют параметры пунктов 8-12 на внешний вид командной оболочки.

14. Кликните **ОК** для ввода.

15. В диалоговом окне **Изменение свойств** выберите пункт **«Сохранить свойства для других окон с тем же именем»** или альтернативный вариант **«Изменить ярлык для запуска этого окна»** и подтвердите ввод.

При изучении возможностей командной оболочки очень важным

является изучение синтаксической структуры ввода команд. Необходимо помнить, что синтаксическая структура отображается в том порядке, в котором следует вводить соответствующую команду и следующие за ней параметры, если таковые имеются.

Следующий пример команды **Xcopy** иллюстрирует разнообразие синтаксических форматов текста, а в табл. 3.1 приведены интерпретации этих форматов.

Xcopy *источник* [*результат*] **[/w] [/p] [/c] [/v] [/q] [/f] [/l] [/g] [/d[:***мм-дд-гггг***]] [/u] [/i] [/s [/e]] [/t] [/k] [/r] [/h] [{/a|/m}] [/n] [/o] [/x] [/exclude:***файл1***[+[***файл2***]][+[***файл3***]]**
[{/y|/-y}] [/z].

Таблица 3.1. Интерпретация текстовых форматов при вводе команд

Формат	**Значение**
Курсив	Данные, которые должен ввести пользователь
Полужирный шрифт	Элементы, которые следует
	вводить точно, как показано
Пропуск (...)	Параметры могут повторяться несколько раз в командной строке
В квадратных скобках ([ ])	Необязательные элементы
В фигурных скобках ({ }); варианты разделены вертикальной чертой (|). Пример: {четные | нечетные}	Набор значений, из которого можно выбрать только одно значение
Шрифт Courier	Текст кода или выхода программы

Кроме того, имеется возможность вкладывать командные оболочки в **Cmd.exe**, открывая новый экземпляр **Cmd.exe** из командной строки. По умолчанию каждый экземпляр **Cmd.exe** наследует среду своего родительского приложения **Cmd.exe**. Вложение экземпляров **Cmd.exe** позволяет вносить в локальную среду изменения, которые не повлияют на родительское приложение **Cmd.exe**. Это позволяет сохранять исходную среду **Cmd.exe** и

возвращаться к ней после удаления вложенной командной оболочки. Изменения вложенной командной оболочки не сохраняются.

При работе с командной строкой команды являются зарезервированными словами, что означает, что нельзя объявлять переменные, имена которых совпадают с именами этих команд. Большинство команд ОС Windows XP было заимствовано разработчиками из дисковой ОС MS-DOS, которая изначально являлась операционной системой с интерфейсом командной строки и использовалась ранее на персональных компьютерах. Как и в других ОС, например в OS/2, MS-DOS позволяла преобразовывать ввод с клавиатуры в команды, организовывать такие действия, как запись и чтение с дисков, вывод на экран, управление с помощью клавиатуры и множество других внутренних операций, обеспечивающих выполнение программ и организацию файлов.

В 32-битной ОС Windows XP в виде командной оболочки методом эмуляции реализован режим MS-DOS, позволяющий выполнять все указанные выше действия по работе с файлами и дисками. Кроме того, ОС Windows XP поддерживает и расширяет практически все функциональные возможности системы MS-DOS, о которых достаточно полно описано в разделе **«Новые способы выполнения типичных задач»** справки операционной системы.

Дополнительную информацию по возможностям командной оболочки, а также все множество команд доступных при работе с ней наряду с параметрами и примерами применения можно получить в справке ОС Windows XP (**Пуск | Справка и поддержка**) в разделах **«Общие сведения о командной оболочке», «Справочник по параметрам командной строки»** и **«Новые средства командной строки».**

3.2. Подготовка к выполнению лабораторной работы

К числу основных команд и служебных утилит, используемых при работе с файлами, дисками и томами в ОС Windows XP посредством командной оболочки, относятся: **Assoc, Attrib, Cacls, Cd, Chdir, Chkdsk, Chkntfs, Comp, Compact, Convert, Copy, Date, Del, Dir, Diskcomp, Diskcopy, Erase, Fc, Find, Findstr, Format,**

Label, Md, Mkdir, Move, Print, Rd, Recover, Ren, Rename, Replace, Rmdir, Subst, Tree, Type, Vol, Xcopy и другие. Дополнительная информация по этим командам, а также примеры их использования доступны в справке ОС Windows XP (**Пуск | Справка и поддержка**) в соответствующих разделах. Справку также можно получить, набрав в окне командной оболочки строку **Help** и нажав **Enter** для ввода. Полный список команд ОС Windows XP, в том числе официально не декларированных в справке ОС (например, команда **Shutdown**), может быть найден на официальном сайте по адресу http://www.microsoft.com.

В настоящей лабораторной работе предполагается ознакомление с основным набором команд и служебных утилит для работы с файлами и выполнение нескольких учебных заданий с применением командной оболочки.

Перед началом выполнения лабораторной работы в среде ОС Windows XP необходимо выполнить следующее:

1) загрузить ОС Windows XP и активировать справочное меню (**Пуск | Справка и поддержка**);

2) ознакомиться с описанием и синтаксисом ввода командного интерпретатора **Cmd.exe**;

3) ознакомиться с описанием и синтаксисом ввода приведенных команд и служебных утилит.

3.3. Порядок выполнения лабораторной работы

Лабораторная работа выполняется последовательно в соответствии с определенным порядком и включает в себя два учебных задания.

3.3.1. Учебное задание №1. Изучение основных команд и служебных утилит при работе с файлами в ОС Windows XP.

Порядок выполнения:

I. Загрузить командную оболочку:

- нажмите **Пуск | Выполнить**,
- наберите в появившемся окне **Cmd.exe** (или просто **cmd**),
- нажмите **Enter** для ввода.

II. Одной из самых важных команд, доступной в среде командной оболочки и предназначенной для копирования одного или нескольких файлов из точки расположения, заданной одним маршрутом, в место назначения, определяемое другим маршрутом, является команда **Copy**. Копирование можно производить в файлы с теми же именами (если они располагаются в разных каталогах) или с другими, изменяя их в процессе копирования. Наряду с этим, при формировании команды в командной строке можно употреблять символы звездочка (*) и вопрос (?), что обеспечивает копирование не одного файла, а целой группы.

Синтаксис команды **Copy**:

Copy [/d] [/v] [/n] [{/y | /-y}] [/z] [{/a | /b}] *источник* [{/a | /b}] [+ *источник* [{/a | /b}] [+ ...]] [*назначение* [{/a | /b}]],

где параметр:

/d — указывает на возможность создания зашифрованного файла.

/v — проверяет правильность копирования путем сравнения копий файлов.

/n — использует короткое имя копируемого файла, если таковое имя имеется и при этом не удовлетворяет стандарту 8.3.

/y — отменяет вывод запроса на подтверждение перезаписи существующего конечного файла.

/-y — инициирует вывод запроса на подтверждение перезаписи существующего конечного файла.

/z — копирует файлы по сети в режиме перезапуска. Если во время фазы копирования теряется сетевое подключение (например, если сервер переходит в автономный режим, разрывая подключение), команда позволяет продолжить копирование после восстановления подключения. Кроме того, этот параметр позволяет отобразить сведения о завершении операции копирования в процентах для каждого файла группы.

/a — указывает на текстовый файл в формате ASCII.

/b — указывает на то, что файл является бинарным. Этот параметр задается по умолчанию и обеспечивает считывание командным интерпретатором количества байт, равного размеру файла в каталоге.

источник — обязательный параметр, задающий расположение файла или набора файлов, которые требуется скопировать. Этот параметр может быть задан полным именем файла, включающим имя диска с двоеточием (:), имя папки, собственно имя файла. Символ плюс (+) осуществляет объединение *источников*.

назначение — обязательный параметр, задающий место расположения, в которое требуется скопировать файл или набор файлов. Этот параметр может быть задан полным именем файла, включающим имя диска с двоеточием (:), имя папки, собственно имя файла. Если конечный файл не задан, по умолчанию файлы будут скопированы с тем же именем, датой и временем создания в текущий каталог на текущем диске. Если при этом исходный файл находится в текущем каталоге на текущем диске, выполнение команды завершается и выводится следующее сообщение об ошибке: «*Невозможно скопировать файл в себя. Скопировано файлов: 0*».

Необходимо отметить, что команда **Copy** не осуществляет копирование файлов, имеющих длину, равную 0 байт. Для выполнения этой операции служит команда **Xcopy**.

Если требуется установить текущую дату и время в качестве даты модификации файла без изменения его содержимого, необходимо воспользоваться следующим синтаксисом: **Copy /b** *источник*+,, Запятые заменяют параметр *назначение*.

Дополнительная информация по данной команде, а также примеры ее использования доступны в справке ОС Windows XP (**Пуск | Справка и поддержка**) в соответствующем разделе. Справку также можно получить, набрав в окне командной оболочки строку **Copy /?** и нажав **Enter** для ввода.

Задание №3.1а.

Исследовать основные способы применения команды копирования **Copy** на конкретных примерах.

1. Скопируйте все файлы с определенным расширением, расположенные в месте, путь к которому задайте самостоятельно, в точку назначения, заданную путем d:\Temp\.
2. Скопируйте файл, расположенный в месте, путь к которому задайте самостоятельно, в точку назначения, заданную другим путем. Инициируйте запрос на подтверждение перезаписи конечного файла в случае, если он существует.
3. Продублируйте файл с определенным именем, путь к которому задайте самостоятельно, в точку назначения, заданную тем же путем, добавив к началу имени файла строку «copy-».
4. Объедините два текстовых (.txt) файла, пути к которым задайте самостоятельно, в один файл с полным именем d:\Temp\test.txt.
5. Введите фрагмент текста с клавиатуры, используя ее *источник* **Con,** в текстовый файл, путь к которому задайте самостоятельно. Признаком конца ввода строки является **Enter**. Признаком конца ввода текста в файл являются нажатые клавиши

   Ctrl+Z и Enter.
6. Добавьте несколько строк с клавиатуры в конец существующего текстового файла, полученного в предыдущем пункте текущего задания.

При выполнении задания используйте следующие инструкции:

- по каждому из пунктов задания в окне командной оболочки наберите соответствующую команду с необходимыми ключами,
- нажмите **Enter** для ввода,
- изучите полученный результат и сделайте вывод о проделанной работе,
- запишите полученную информацию в отчет, заполнив табл. 1.2.

Таблица 3.2. Результаты выполнения команды Copy

№ п/п.	Команда с ключами	Результат и вывод по способу применения команды
1.		
2.		
3.		
4.		
5.		
6.		

III. Другая команда, дополняющая, расширяющая функционал предыдущей команды и осуществляющая копирование не только файлов, групп файлов, но и каталогов, включая подкаталоги, называется **Xcopy.**

Синтаксис команды **Xcopy**:

Xcopy *источник* [*результат*] [/w] [/p] [/c] [/v] [/q] [/f] [/l] [/g] [/d[:*мм-дд-гггг*]]

[/u] [/i] [/s [/e]] [/t] [/k] [/r] [/h] [{/a | /m}] [/n] [/o] [/x]

[/**exclude**:*файл1*[+[*файл2*]][+[*файл3*]] [{/y | /-y}] [/z],

где параметр:

источник — обязательный параметр, задающий местонахождение и имена

файлов для копирования. Этот параметр должен задавать или диск, или путь.

результат — обязательный параметр, задающий место расположения, в которое требуется скопировать файл или набор файлов. Этот параметр может быть задан полным именем файла, включающим имя диска с двоеточием (:), имя папки, собственно имя файла. Если параметр *результат* не задан, копирование будет производиться в текущий каталог.

/w — выводит следующее сообщение с ожиданием подтверждения начала копирования: «*Нажмите любую клавишу, чтобы начать копирование файлов*»

/p — запрашивает подтверждение при создании файла-*результата*.

/c — игнорирует ошибки в процессе копирования.

/v — в процессе копирования проверяет каждый скопированный файл на соответствие его оригиналу.

/q — отменяет вывод на экран сообщений команды и имен файлов в процессе копирования.

/f — выводит имена исходных файлов и файлов-результатов в процессе копирования.

/l — отображает список копируемых файлов.

/g — разрешает копирование шифрованных файлов в конечную папку, не поддерживающую шифрование.

/u — копирует (обновляет) только те файлы-*источники*, которые уже существуют в каталоге *результата.*

/d[:*мм-дд-гггг***]** — копирует только файлы, измененные не ранее заданной даты. Если не включить значение *мм-дд-гггг*, копируются все файлы-*источники*, которые новее существующих файлов-*результатов*. Эта возможность позволяет обновлять только измененные файлы.

/i — если *источником* является каталог или источник содержит подстановочные знаки (например, звездочка *) и *результат* не существует, считается, что *результат* — это имя каталога, и при этом создается новый каталог. Затем команда копирует все указанные файлы в этот новый каталог. По умолчанию команда запрашивает подтверждение, является ли параметр *результат* каталогом или файлом.

/s — копирует каталоги и подкаталоги в случае, если они не пусты.

/e — копирует все подкаталоги, включая пустые.

/t — копирует структуру подкаталога (дерево) без файлов.

/k — копирует файлы с атрибутом «только для чтения» с сохранением этого атрибута у скопированных файлов.

/r — заменяет файлы, доступные «только для чтения».

/h — копирует системные и скрытые файлы.

/a — копирует только те файлы, которые имеют атрибут «архивный».

/m — копирует только те файлы, которые имеют атрибут «архивный». В отличие от параметра /a, параметр **/m** очищает атрибут «архивный» у скопированных файлов.

/n — копирует файлы с использованием коротких имен ФС NTFS. Этот параметр требуется при копировании из ФС NTFS в ФС FAT или когда на диске*результате* требуется использование стандарта 8.3. как в ФС FAT.

/o — копирует сведения о принадлежности файлов и избирательной таблице управления доступом (DACL).

/x — копирует сведения о параметрах аудита файла и системной таблице управления доступом (SACL) (подразумевается наличие параметра /p).

/exclude:*файл1*[+[*файл2*]][+[*файл3*]] — определяет список файлов, содержащих строки. Каждая строка должна находиться в отдельной линии в файле. Если одна из строк совпадает с любой частью абсолютного пути копируемого файла, то такой файл копироваться не будет.

/y — отменяет вывод запроса на подтверждение перезаписи существующего конечного файла.

/-y — инициирует вывод запроса на подтверждение перезаписи существующего конечного файла.

/z — копирует файлы по сети в режиме перезапуска. Если во время фазы копирования теряется сетевое подключение (например, если сервер переходит в автономный режим, разрывая подключение), команда позволяет продолжить копирование после восстановления подключения.

Дополнительная информация по данной команде, а также примеры ее использования доступны в справке ОС Windows XP **(Пуск | Справка и поддержка)** в соответствующем разделе. Справку также можно получить, набрав в окне командной оболочки строку **Xcopy /?** и нажав **Enter** для ввода.

Задание №3.1б.

Исследовать основные способы применения команды копирования **Xcopy** на конкретных примерах.

1. Скопируйте все файлы и подкаталоги, включая пустые и скрытые, расположенные в месте, путь к которому задайте самостоятельно, в точку назначения на другом локальном диске. При этом инициируйте запрос на подтверждение перезаписи.
2. Скопируйте дерево каталогов, включая пустые, расположенные в месте, путь к которому задайте самостоятельно, в точку назначения на другом локальном диске.
3. Скопируйте все файлы с атрибутами «архивный» и «только для чтения» с сохранением этого атрибута для файлов-*результатов*, расположенные в месте, путь к которому задайте самостоятельно, в точку назначения, заданную путем d:\Temp\.
4. Скопируйте все файлы и подкаталоги с датой не позднее определенной. Путь к *источнику* и точке *назначения* задайте самостоятельно. Отобразите список файлов в процессе копирования.

При выполнении задания используйте следующие инструкции:

- по каждому из пунктов задания в окне командной оболочки наберите соответствующую команду с необходимыми ключами,
- нажмите **Enter** для ввода,

- изучите полученный результат и сделайте вывод о проделанной работе, запишите полученную информацию в отчет, заполнив табл.3.3.

Таблица 3.3. Результаты выполнения команды Xcopy

№ п/п.	Команда с ключами	Результат и вывод по способу применения команды
1.		
2.		
3.		
4.		

IV. Команда **Move** служит для перемещения одного или нескольких файлов из одного каталога в другой.

Синтаксис команды **Move**:

Move [{/y|/-y}] [*источник*] [*результат*],

где параметр:

источник — полное имя одного или нескольких файлов, предназначенных для

перемещения.

результат — полное имя места назначения, куда требуется переместить выбранные файлы.

/y — отменяет вывод запроса на подтверждение перезаписи существующего файла-*результата*.

/-y — инициирует вывод запроса на подтверждение перезаписи существующего файла-*результата*.

Дополнительная информация по данной команде, а также примеры ее использования доступны в справке ОС Windows XP (**Пуск | Справка и поддержка**) в соответствующем разделе. Справку также можно получить, набрав в окне командной оболочки строку **Move /?** и нажав **Enter** для ввода.

Задание №3.1в.

Исследовать основные способы применения команды перемещения **Move** на конкретных примерах.

1. Скопируйте пять любых файлов с определенным расширением, расположенные в месте *источника*, путь к которому выберите самостоятельно, в точку назначения, заданную путем d:\Temporary\. При копировании воспользуйтесь любым методом, изученным ранее.

2. Воспользовавшись командой единожды, переместите все только что скопированные файлы, заданные путем d:\Temporary\, обратно в место *источника*. При этом инициируйте вывод запроса на подтверждение перезаписи.

При выполнении задания используйте следующие инструкции:

- по каждому из пунктов задания в окне командной оболочки наберите соответствующую команду с необходимыми ключами,
- нажмите **Enter** для ввода,
- изучите полученный результат и сделайте вывод о проделанной работе,
- запишите полученную информацию в отчет, заполнив табл. 3.4.

Таблица 3.4. Результаты выполнения команды **Move**

№ п/п.	Команда с ключами	Результат и вывод по способу применения команды
1.		
2.		

V. Команда **Replace** предназначена для замены файлов в каталоге - *назначения* файлами с теми же именами из другого каталога - *источника*. Команда может быть применена для добавления файлов, которых еще не имеется в каталоге.

Синтаксис команды **Replace**:

Replace [*диск1*:][*путь1*] *имя_файла* [*диск2*:][*путь2*] **[/a] [/p] [/r] [/w]**
Replace [*диск1*:][*путь1*] *имя_файла* [*диск2*:][*путь2*] **[/p] [/r] [/s] [/w] [/u]**,

где параметр:

[*диск1*:][*путь1*] *имя_файла* — задает местонахождение и имя файла или набора исходных файлов.

[*диск2*:][*путь2*] — задает местонахождение файла-*результата*. Если

параметр не задан, используется текущий диск и каталог.

/a — добавляет, а не перезаписывает файлы в каталог-*результат*. Нельзя использовать данный параметр совместно с параметрами /s или /u.

/p — добавляет или перезаписывает файлы с подтверждением.

/r — замещает файлы, предназначенные только для чтения, так же, как и обычные файлы. Если этот параметр не задан, а программа пытается заменить файл, предназначенный только для чтения, на экран будет выведено сообщение об ошибке и операция замены будет остановлена.

/w — перед началом поиска исходных файлов система находится в состоянии ожидания, пока пользователь вставит диск в дисковод. Если ключ не задан, замена (добавление) файлов начинается сразу после нажатия клавиши **Enter**.

/s — ищет по всем подкаталогам каталога-*назначения* и заменяет файлы с подходящими именами. Нельзя использовать данный параметр совместно с параметром **/a**.

/u — заменяет (обновляет) только те файлы, которые имеют более раннюю дату модификации, чем файлы в исходном каталоге. Нельзя использовать данный параметр совместно с параметром /a.

Ограничение команды **Replace** заключается в том, что она не может быть использована для обновления скрытых или системных файлов.

Дополнительная информация по данной команде, а также примеры ее использования доступны в справке ОС Windows XP (**Пуск | Справка и поддержка**) в соответствующем разделе. Справку также можно получить, набрав в окне командной оболочки строку **Replace /?** и нажав **Enter** для ввода.

Задание №3.1г.

Исследовать основные способы применения команды замены **Replace** на конкретных примерах.

1. Скопируйте три любых файла, расположенные в месте каталога - *источника*, путь к которому выберите самостоятельно, в каждый из двух каталогов - *назначения*, заданных следующими путями d:\Temp\Begin\ и d:\Temp\-End\. При копировании воспользуйтесь любым методом, изученным ранее.

2. Замените первый по порядку файл в каталоге - *назначения* d:\Temp\End\ файлом, расположенным в каталоге - *источнике* d:\Temp\Begin\, осуществив подтверждение замены.

3. Замените второй по порядку файл с более ранней датой модификации и путем - *назначения* d:\Temp\End\ файлом, расположенным в

каталоге - *источнике* d:\Temp\Begin\, предварительно каким-либо образом его модифицировав.

4. Активируйте атрибут «только для чтения» у третьего по порядку файла в каталогах d:\Temp\Begin\ и d:\Temp\End\. Замените третий по порядку файл в каталоге

- *назначения* d:\Temp\End\ файлом, расположенным в каталоге - *источнике* d:\Temp\Begin\.

При выполнении задания используйте следующие инструкции:

- по каждому из пунктов задания в окне командной оболочки наберите соответствующую команду с необходимыми ключами,

- нажмите **Enter** для ввода,

-изучите полученный результат и сделайте вывод о проделанной работе,

- запишите полученную информацию в отчет, заполнив табл. 3.5.

Таблица 3.5. Результаты выполнения команды **Replace**

№ п/п.	Команда с ключами	Результат и вывод по способу применения команды
1.		
2.		
3.		
4.		

VI. Команда **Ren (Rename)** предназначена для переименования файла.

Синтаксис команды **Ren:**

Ren [*диск*:][*путь*] *имя_файла1 имя_файла2*,

где параметр:

[*диск:*][*путь*] *имя_файла1* — имя и место расположения файла, который требуется переименовать. *имя_файла2* — новое имя файла; при переименовании не могут быть заданы новый

диск или каталог.

Дополнительная информация по данной команде, а также примеры ее использования доступны в справке ОС Windows XP (**Пуск | Справка** и

поддержка) в соответствующем разделе. Справку также можно получить, набрав в окне командной оболочки строку **Ren /?** и нажав **Enter** для ввода.

Задание №3.1д.

Исследовать основные способы применения команды переименования **Ren (Rename)** на конкретных примерах.

1. Скопируйте пять любых файлов с определенными разрешениями, расположенные в месте, путь к которому выберите самостоятельно, в точку назначения, заданную путем d:\Temp\. При копировании воспользуйтесь любым методом, изученным ранее.
2. Измените типы всех скопированных файлов, заданных путем d:\Temp\, на другой, выбранный самостоятельно тип.
3. Переименуйте все файлы, заданные путем d:\Temp\, в файлы с именами

Renamed1.Ren, Renamed2.Ren, … , Renamed5.Ren.

При выполнении задания используйте следующие инструкции:

- по каждому из пунктов задания в окне командной оболочки наберите соответствующую команду с необходимыми ключами,
- нажмите **Enter** для ввода,
- изучите полученный результат и сделайте вывод о проделанной работе,
- запишите полученную информацию в отчет, заполнив табл. 3.6.

Таблица 3.6. Результаты выполнения команды **Rename**

№ п/п.	**Команда с ключами**	**Результат и вывод по способу применения команды**
1.		
2.		
3.		

VII. Утилита и одноименная команда **Fc** обеспечивает сравнение двух файлов и вывод различий между ними.

Синтаксис команды **Fc**:

Fc [/a] [/b] [/c] [/l] [/lb*n*] [/n] [/t] [/u] [/w] [/*nnnn*] [*диск1*:][*путь1*]*имяфайла1*

[*диск2*:][*путь2*]*имяфайла2*,

где параметр:

/a — задает сокращенный вывод сравнения в текстовом режиме ASCII. Вместо вывода всех различающихся строк, выводятся только начальная и конечная строки отличающихся участков.

/b — сравнивает файлы в бинарном режиме. При этом два файла сравниваются байт за байтом без сопоставления их после найденного отличия. Этот режим используется по умолчанию для сравнения бинарных файлов с расширениями: .exe,

.com, .sys, .obj, .lib или .bin.

/c — сравнивает без учета заглавных и строчных букв.

/l — сравнивает файлы в текстовом режиме ASCII. При этом два файла сравниваются строка за строкой с их сопоставлением, после того как найдено отличие. Этот режим используется по умолчанию для сравнения файлов с любыми расширениями, исключая бинарные: .exe, .com, .sys, .obj, .lib или .bin.

/lb*n* — задает количество строк *n* для внутреннего буфера. Если количество отличающихся строк в сравниваемых файлах превышает заданное по умолчанию число для длины буфера в 100 строк, сравнение прекращается и выводится сообщение об ошибке: «*Не удается выполнить синхронизацию строк. Слишком много различий между файлами*».

/n — задает вывод номеров строк при сравнении в текстовом режиме.

/t — предотвращает команду от преобразования меток табуляции в пробелы. По умолчанию табуляторы заменяются пробелами с остановкой в каждой восьмой позиции.

/u — задает сравнение файлов в текстовом формате Unicode.

/w — задает сжатие пробелов и табуляций при сравнении. Если в строке содержится несколько пробелов или табуляций подряд, при использовании ключа /w они будут рассматриваться как один. При этом игнорируются и не сравниваются пробелы и табуляции в начале и в конце строки.

/nnnn — задает количество совпадающих строк при сопоставлении файлов. Если количество совпадающих строк в файле меньше *nnnn*, выводятся совпадающие строки как отличающиеся.

[*диск1*:][*путь1*] *имя файла 1* — обязательный параметр, задающий местоположение и имя первого файла для сравнения.

[*диск2*:][*путь2*] *имя файла 2* — обязательный параметр, задающий местоположение и имя второго файла для сравнения.

При сравнении файлов в текстовом режиме ASCII, отображаются различия между ними в следующем порядке:

- *имя файла 1*,
- строки из параметра *имя файла 1*, отличающиеся в файлах,
- первая строка, совпадающая в обоих файлах,
- *имя файла 2*,
- строки из параметра *имя файла 2*, отличающиеся в файлах, - первая строка, совпадающая в обоих файлах.

При сравнении файлов в бинарном режиме, отображаются найденные несоответствия в виде **xxxxxxxx: yy zz**. Величина **xxxxxxxx** задает относительный шестнадцатеричный адрес пары различающихся байтов, отсчитываемый от начала файла. Шестнадцатеричные величины **yy** и **zz** представляют различающиеся байты из файлов с именами 1 и 2 соответственно.

Дополнительная информация по данной команде, а также примеры ее использования доступны в справке ОС Windows XP (**Пуск | Справка и поддержка**) в соответствующем разделе. Справку также можно получить, набрав в окне командной оболочки строку **Fc /?** и нажав **Enter** для ввода.

Задание №3.1е.

Исследовать основные способы применения команды сравнения **Fc** на конкретных примерах.

1. Сравните два текстовых файла, пути к которым задайте самостоятельно. Результат сравнения выведите в файл Result.txt (**Приложение 1**).
2. Сравните два бинарных файла, пути к которым задайте самостоятельно. Результат сравнения добавьте в файл Result.txt (**Приложение 1**).

При выполнении задания используйте следующие инструкции:

- по каждому из пунктов задания в окне командной оболочки наберите соответствующую команду с необходимыми ключами,
- нажмите **Enter** для ввода,
- изучите полученный результат и сделайте вывод о проделанной работе,
- запишите полученную информацию в отчет, заполнив табл. 3.7.

Таблица 3.7. Результаты выполнения команды **Fc**

№	Команда с ключами	Результат и вывод

п/п.		по способу применения команды
1.		
2.		

VIII. Команды **Del (Delete)** и **Erase** удаляют файлы или группы файлов из текущего каталога.

Синтаксис команд **Del** и **Erase**:

Del [*диск*:][*путь*] *имя_файла* [...] [/p] [/f] [/s] [/q] [/a[:*атрибуты*]]

Erase [*диск*:][*путь*] *имя_файла* [...] [/p] [/f] [/s] [/q] [/a[:*атрибуты*]]

где параметр:

[*диск*:][*путь*] *имя_файла* — обязательный параметр, задающий расположение и имя файла для удаления. В случае, если предполагается удалить несколько файлов, их имена перечисляются запятой (,) или точкой с запятой (;).

/p — удаляет каждый файл с подтверждением.

/f — удаляет файлы с атрибутом «Только для чтения».

/s — удаляет заданные файлы в каталоге и всех его подкаталогах.

/q — не выводит подтверждение на удаление.

/a — удаляет файлы с заданными атрибутами. Список атрибутов следующий: «r» — только для чтения, «a» — архивный, «s» — системный, «h» — скрытый, «-» — префикс «нет».

Дополнительная информация по данной команде, а также примеры ее использования доступны в справке ОС Windows XP (**Пуск | Справка и поддержка**) в соответствующем разделе. Справку также можно получить, набрав в окне командной оболочки строку **Del (Erase) /?** и нажав **Enter** для ввода.

Задание №3.1ж.

Исследовать основные способы применения команд удаления **Del** и **Erase** на конкретных примерах.

1. Скопируйте все файлы, расположенные в месте, путь к которому выберите самостоятельно, в точку назначения, заданную путем d:\Temp\. При копировании воспользуйтесь любым методом, изученным ранее.

2. Удалите выбранный самостоятельно файл, заданный путем d:\Temp\, запросив подтверждение на удаление.

3. Удалите все файлы с атрибутом «Системный», расположенные в месте, заданном путем d:\Temp\. Подтверждение на удаление не выводить.

4. Удалите все файлы с определенным расширением, расположенные в месте, заданном путем d:\Temp\, запросив подтверждение на удаление.

5. Удалите все оставшиеся файлы, включая каталоги, расположенные в месте, заданном путем d:\Temp\. Подтверждение на удаление не выводить.

При выполнении задания используйте следующие инструкции:

- по каждому из пунктов задания в окне командной оболочки наберите соответствующую команду с необходимыми ключами,
- нажмите **Enter** для ввода,
- изучите полученный результат и сделайте вывод о проделанной работе,
- запишите полученную информацию в отчет, заполнив табл. 3.8.

Таблица 3.8. Результаты выполнения команды **Delete**

№ п/п.	Команда с ключами	Результат и вывод по способу применения команды
1.		
2.		
3.		
4.		
5.		

Следующая группа команд предназначена работы непосредственно с каталогами и файлами, заданными полными именами. К их числу относятся: **Dir, Cd (ChDir), Md (MkDir), Rd (RmDir)** и другие. Рассмотрим более подробно основные из них.

IX. Команда **Dir** выводит список полных имен файлов и подкаталогов каталога, размер в байтах каждого из них, время, дату создания и последнего изменения. Команда **Dir** также выводит общее число перечисленных файлов и каталогов, их общий размер и свободное пространство на локальном диске.

Синтаксис команды **Dir**:

Dir [*диск*:][*путь*][*имя_файла*] [...] [/p] [/q] [/w] [/d]
[/a[[:]*атрибуты*]][/o[[:]*порядок_сортировки*]] [/t[[:]*поле_сортировки*]] [/s] [/b] [/l] [/n]
[/x] [/c] [/4]

где параметр:

[*диск*:][*путь*] — задает путь, список файлов по которому будет выведен.

имя_файла — задает имя файла или группы файлов, сведения о которых требуется вывести.

/a [[:] *атрибуты*] — выводит имена только тех файлов и каталогов, которые имеют указанные атрибуты. Если параметр /a не указан, выводятся имена всех файлов, за исключением системных и скрытых. Если параметр /a указан без *атрибутов*, выводятся имена всех файлов, включая скрытые и системные. Ниже приведен список значений, которые могут быть использованы при задании параметра *атрибуты* (табл. 3.9).

/p — выводит сведения постранично.

/q — выводит сведения о владельце файла.

/w — выводит сведения о файлах несколькими колонками.

- Таблица 3.9. Значения параметра *атрибуты*

№ п/п.	Значение	Описание
1.	**H**	Скрытые файлы
2.	**S**	Системные файлы
3.	**D**	Каталоги
4.	**A**	Файлы, готовые к архивированию
5.	**R**	Файлы, доступные только для чтения
6.	**-h**	Файлы, не являющиеся скрытыми
7.	**-s**	Файлы, не являющиеся системными

8.	-d	Только файлы (не каталоги)
9.	-a	Файлы, не изменявшиеся после последнего архивирования
10.	-r	Файлы, не имеющие атрибута «Только чтение»

/d — соответствует **/w**, но с сортировкой сведений о файлах по столбцам.

/o [[**:**]*порядок_сортировки*] — управляет порядком сортировки и вывода имен файлов и каталогов. Если параметр **/o** не задан, выводятся имена в том порядке, в котором они записаны в каталоге. Если параметр **/o** использован без параметра *порядок_сортировки*, выводятся имена каталогов в алфавитном порядке, затем выводятся имена файлов в алфавитном порядке. Ниже приведен список значений, которые могут быть использованы при задании параметра *порядок_сортировки* (табл. 3.10).

Таблица 3.10. Значения параметра *порядок_сортировки*

№ п/п.	Значение	Описание
1.	N	Сортировка по именам в алфавитном порядке
2.	E	Сортировка по расширениям в алфавитном порядке
3.	D	Сортировка по дате и времени от ранних к поздним
4.	S	Сортировка по размеру от меньших к большим
5.	G	Сортировка с группированием каталогов перед файлами

6.	-n	Сортировка по именам в обратном алфавитном порядке (от Z к A)
7.	-e	Сортировка по расширению в обратном алфавитном порядке (от .ZZZ к .AAA)
8.	-d	Сортировка по дате и времени от поздних к ранним
9.	-s	Сортировка по размеру от больших к меньшим
10.	-g	Сортировка с группировкой каталогов после файлов

/t [[:]*поле_времени*] — задает поля времени для вывода и сортировки. Ниже приведен список значений, которые могут быть использованы при задании параметра *поле_времени* (табл. 3.11).

Таблица 3.11. Значения параметра *поле_времени*

№ п/п.	Значение	Описание
1.	C	Создание
2.	A	Последнее обращение
3.	W	Последняя запись

/s — перечисляет все случаи обнаружения определенного имени файла в указанном каталоге и всех его подкаталогах.

/b — перечисляет каждое имя файла (включая расширение) или каталога

на отдельной строке.

/l — выводит несортированный список имен файлов и каталогов строчными буквами.

/n — выводит список в расширенном формате с именами файлов в правой части экрана.

/x — выводит сокращенные имена файлов ФС NTFS и FAT.

/c — выводит разделитель десятичных разрядов в размере файлов.

/4 — отображает год в четырехзначном формате.

Дополнительная информация по данной команде, а также примеры ее использования доступны в справке ОС Windows XP (**Пуск | Справка и поддержка**) в соответствующем разделе. Справку также можно получить, набрав в окне командной оболочки строку **Dir /?** и нажав **Enter** для ввода.

Задание №3.1з.

Исследовать основные способы применения команды **Dir** на конкретных примерах. 1. Выведите постранично содержимое каталога C:\Windows\, включая вложенные подкаталоги и файлы.

2. Выведите постранично все каталоги и файлы на локальном диске D: в алфавитном порядке с сортировкой по столбцам и паузой после заполнения каждого экрана.

3. Выведите все файлы с расширением .doc на локальном диске D: в алфавитном порядке с сортировкой по колонкам. Вывод осуществите в файл Doc-Files.txt (**Приложение 1**).

4. Выведите все каталоги на локальном диске C: в алфавитном порядке. Результат добавьте в файл DocFiles.txt (**Приложение 1**).

5. Добавьте сведения о владельцах файлов системного каталога C:\Windows\ в файл DocFiles.txt (**Приложение 1**).

При выполнении задания используйте следующие инструкции:

- по каждому из пунктов задания в окне командной оболочки наберите соответствующую команду с необходимыми ключами,
- нажмите **Enter** для ввода,
- изучите полученный результат и сделайте вывод о проделанной работе,
- запишите полученную информацию в отчет, заполнив табл. 3.12.

Таблица 3.12. Результаты выполнения команды **Dir**

№ п/п.	Команда с ключами	Результат и вывод по способу применения команды
1.		
2.		
3.		
4.		
5.		

X. Следующая команда **Cd (ChDir)** выводит имя текущего каталога или осуществляет переход в другую папку.

Синтаксис команды **Cd (ChDir)**:

Cd [**[/d]** [*диск:*][*путь*] [..]] [**[/d]** [*диск:*][*путь*] [..]], **Chdir** [**[/d]** [*диск:*][*путь*] [..]] [**[/d]** [*диск:*][*путь*] [..]],

где параметр:

/d — осуществляет смену текущего диска или каталога на диске.

[*диск:*][*путь*] — задает имя диска и каталога, в который требуется перейти.

[..] — переходит в родительскую папку или на уровень выше.

Дополнительная информация по данной команде, а также примеры ее использования доступны в справке ОС Windows XP (**Пуск | Справка и поддержка**) в соответствующем разделе. Справку также можно получить, набрав в окне командной оболочки строку **Cd (ChDir) /?** и нажав **Enter** для ввода.

Задание №3.1и.

Исследовать основные способы применения команды перехода в другой каталог **Cd**

(ChDir) на конкретных примерах.

1. Смените текущий каталог на каталог, полный путь к которому задан следующим образом C:\WINDOWS\Help\Tours\WindowsMediaPlayer\Video\.
2. Перейдите из подкаталога ..\Video на уровень выше.

3. Смените текущий каталог на каталог, полный путь к которому задан следующим образом C:\WINDOWS\Help\Tours\WindowsMediaPlayer\Audio\.

4. Перейдите из подкаталога ...\Audio на два уровня выше.

5. Смените текущий локальный диск на диск D:

При выполнении задания используйте следующие инструкции:

- по каждому из пунктов задания в окне командной оболочки наберите соответствующую команду с необходимыми ключами,
- нажмите **Enter** для ввода,
- изучите полученный результат и сделайте вывод о проделанной работе,
- запишите полученную информацию в отчет, заполнив табл.3.13.

Таблица 3.13. Результаты выполнения команды **Cd**

№ п/п.	Команда с ключами	Результат и вывод по способу применения команды
1.		
2.		
3.		
4.		
5.		

XI. Команда **Md (MkDir)** создает каталог или подкаталог.

Синтаксис команды **Md (MkDir)**:

Mkdir [*диск*:]*путь*, **Md** [*диск*:]*путь*,

где параметр:

[*диск*:] — задает диск, на котором будет создана новая папка.

путь — задает имя и размещение новой папки.

Дополнительная информация по данной команде, а также примеры ее использования доступны в справке ОС Windows XP (**Пуск | Справка и поддержка**) в соответствующем разделе. Справку также можно получить,

набрав в окне командной оболочки строку **Md (MkDir) /?** и нажав **Enter** для ввода.

Задание №3.1к.

Исследовать основные способы применения команды создания каталога **Md (MkDir)** на конкретных примерах.

1. Создайте каталог, путь к которому выберите самостоятельно.
2. Единожды воспользовавшись командой, создайте каталог, полный путь к которому задан следующим образом d:\Temp\VMGroup\MyPath\.

При выполнении задания используйте следующие инструкции:

- по каждому из пунктов задания в окне командной оболочки наберите соответствующую команду с необходимыми ключами,
- нажмите **Enter** для ввода,
- изучите полученный результат и сделайте вывод о проделанной работе,
- запишите полученную информацию в отчет, заполнив табл. 3.14.

Таблица 3.14. Результаты выполнения команд **Md**

№ п/п	Команда с ключами	Результат и вывод по способу применения команды
1.		
2.		

XII. Команда **Rd (RmDir)** удаляет каталоги и подкаталоги.

Синтаксис команды **Rd (RmDir)**:

Rmdir [*диск*:]*путь* [/s] [/q], **Rd** [*диск*:]*путь* [/s] [/q],

где параметр:

[*диск*:]*путь* — задает диск и местоположение каталога для удаления.

/s — удаляет дерево каталогов вместе с файлами.

/q — удаляет каталоги без запроса подтверждения.

Дополнительная информация по данной команде, а также примеры ее использования доступны в справке ОС Windows XP (**Пуск | Справка и поддержка**) в соответствующем разделе. Справку также можно получить,

набрав в окне командной оболочки строку **Rd (RmDir) /?** и нажав **Enter** для ввода.

Задание №3.1л.

Исследовать основные способы применения команды удаления каталога **Rd (RmDir)** на конкретных примерах.

1. Удалите подкаталог третьего уровня MyPath, созданный в предыдущем задании **№1к.**
2. Скопируйте несколько файлов, расположенных в месте, путь к которому выберите самостоятельно, в точку назначения, заданную путем d:\Temp\VMGroup\. При копировании воспользуйтесь любым методом, изученным ранее.
3. Единожды воспользовавшись командой, без запроса подтверждения удалите дерево каталогов d:\Temp\VMGroup\, включая подкаталог второго уровня VMGroup с содержащимися внутри файлами.

При выполнении задания используйте следующие инструкции:

- по каждому из пунктов задания в окне командной оболочки наберите соответствующую команду с необходимыми ключами,
- нажмите **Enter** для ввода,
- изучите полученный результат и сделайте вывод о проделанной работе,
- запишите полученную информацию в отчет, заполнив табл. 3.15.

Таблица 3.15. Результаты выполнения команды **Rd**

№ п/п	Команда с ключами	Результат и вывод по способу применения команды
1.		
2.		
3.		

- Отчет по практической работе оформляется в соответствии с требованиями. В группе файл «Пример оформления отчета»;
- краткое описание служебных команд и утилит, предназначенных для работы с файлами и дисками в среде командной оболочки (подраздел 3.1);
- результаты исследований работы служебных команд и утилит в соответствии учебными заданиями практической работы;
- заполненные таблицы учебных заданий практической работы;
- выводы о проделанной работе.

Практическая работа №4.
Ознакомление с работой учебного микропроцессорногокомплекса.

Цель работы: ознакомиться с учебным микропроцессорным комплексом на базе восьмиразрядного CISC микропроцессора i8080 (отечественный аналог КР580ВМ80) с архитектурой фон Неймановского типа.

Теоретическая часть

Учебный микропроцессорный комплекс (УМК) представляет собой совокупность аппаратных и программных средств, которые позволяют изучать работу микропроцессора и других программируемых интегральных схем. Основа аппаратных и программных средств УМК -центральный блок, представляющий собой по структуре внутрисхемный эмулятор с программным монитором - вариант отладочного комплекса, используемого при создании программных средств для микропроцессорных контроллеров. Аппаратные средства центрального блока УМК представлены на рис. 4.1. Программные средства (системный монитор) содержатся в постоянном запоминающем устройстве (ПЗУ1) центрального блока и занимают объем 1 килобайт с адреса 0 по адрес 03ffh. Системный монитор содержит программные средства, обеспечивающие:

- начальный запуск микропроцессора,
- работу в непрерывном или шаговом режиме,
- фиксацию точек останова с сохранением состояния процессора в стеке,
- работу клавиатуры и индикатора, выполнение некоторых встроенных процедур.

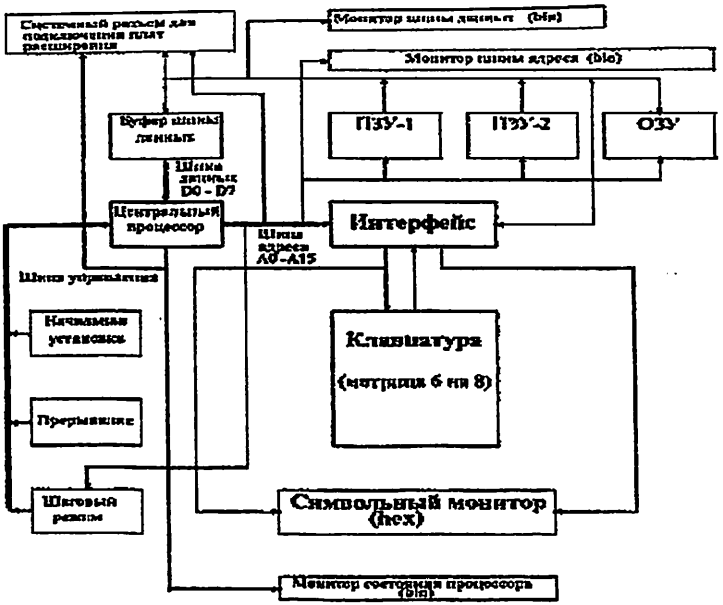


Рис. 4.1 Структурная схема центрального блока УМК

Кроме того, предусмотрена прямая двоичная индикация состояния микропроцессора (PSW)и его шины данных и адреса.

ПЗУ2 объёмом 1К занимает адреса с 03ffh до 07ffh и зарезервировано для расширения системных возможностей УМК. Оперативное запоминающее устройство (ОЗУ или в англоязычной аббревиатуре RAM) статического типа (SRAM) объёмом 1К предназначено для хранения программ пользователя (прикладных программ), организации стека монитора и стека пользователя, а также для поддержки работы системного монитора. Начальный адрес ОЗУ - 0800h.

В УМК предусмотрено расширение аппаратных возможностей за счет подключения к системному разъему центрального блока (см. рис. 4.1) плат расширения, каждая из которых содержит комплект дополнительных аппаратных средств.

Кроме того, большинство плат расширения дает возможность создания произвольных аппаратных структур на макетном поле.

Перечень плат расширения:

- М1- параллельный интерфейс;
- ПГМ - программатор РПЗУ и последовательный интерфейс;

- ПС - светодиодная матрица и дополнительная клавиатура;
- АЦА - аналоговый интерфейс ввода-вывода;
- М2 - расширение объема ЗУ и параллельный интерфейс;
- КОП – магистральный приборный интерфейс;
- ППИ - таймер с аппаратным и акустическим выходом.

Включение УМК

После включения кнопкой "ВКЛ" не должны светиться индикаторы перегрузки блокапитания. В противном случае необходимо выключить УМК и после выдержки около 5 секунд повторить включение.

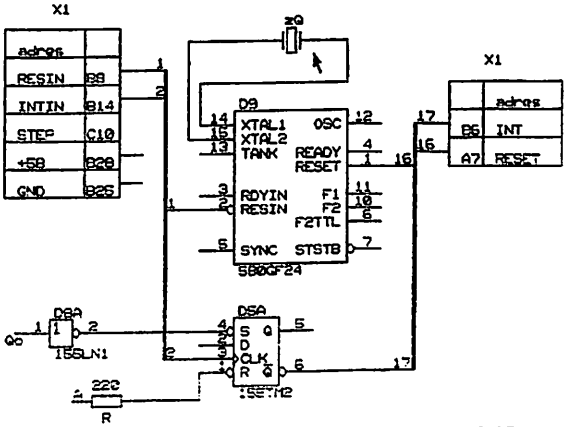


Рис. 4.2 Схема формирования сигнала "RESET

После включения питания нажать и отпустить кнопку "СБРОС". В момент ее отпускания на входе "RESET" микропроцессора формируется сигнал высокого логического уровня для его начальной установки. На Рис. 4.2 представлены фрагменты электрической принципиальной схемы для цепи формирования сигнала " RESET". На рисунках сохранены все обозначения и нумерация элементов, принятые на электрических принципиальных схемах технического описания УМК завода изготовителя, поэтому порядок индексов отличается от того, который предписан ЕСКД.

"Окончательное формирование системного сигнала "RESET" происходит в системном генераторе КР580ГФ24 (элемент D9 на Рис. 15). Для согласования контактной пары клавиши "СБРОС" с входом системного генератора используется стандартный набор аппаратных средств, основное назначение которых – исключение дребезга контактов при их размыкании и замыкании. Дребезг контактов – это переходный процесс, заключающийся в многократных коммутациях контактов при их соединении или разъединении. Длительность процесса составляет около 5 mc и зависит от упругости

контактов и их поверхностных свойств.

Многократные коммутации приводят к ошибкам при вводе информации с любых контактных датчиков, в т.ч. клавиатуры. Для исключения ошибок используются программные и схемные методы. В данном случае рассматривается наиболее распространенный схемный метод с использованием RS-триггера.

Схема, представленная на Рис. 15 содержит несколько элементов, расположенных в различных блоках УМК. Для их электрической связи используются разъёмы и межблочные соединения (межблочный монтаж). На электрических принципиальных схемах межблочные соединения, как правило, не приводятся, и наличие электрической связи подразумевается между одноименными контактами одноименных разъемов. Для упрощения графики сложных электрических принципиальных схем используются условные групповые линии, каждая из которых содержит необходимое число реальных соединительных линий. Каждая реальная линия при входе в канал нумеруется цифрами или буквенно-цифровым символом. Кроме номеров линий на принципиальной схеме приводятся номера выводов корпуса интегральной схемы и ее позиционный номер. Межблочные соединения приводятся на отдельной схеме соединений.

Встроенные процедуры

После прохождения сигнала “RESET”, сформированного аппаратным путем, происходит обнуление внутренних регистров МП. Устанавливается машинный цикл М1 – чтение кода первой операции из ячейки ЗУ с нулевым адресом. С нулевого адреса микропроцессор начинает выполнение программ системного монитора и продолжает их выполнение в циклическом режиме, поддерживая работу клавиатуры, индикатора и других элементов структуры. В этом режиме УМКожидает нажатия оператором одной из функциональных клавиш для выполнения стандартной процедуры, входящей в состав программ системного монитора. В состоянии ожидания выбора процедуры в старшем разряде индикатора формируется знак "-".

Выполнение (“ВП”)

Процедура “ВП” (аналог “Enter” в компьютере) подтверждает принятые ранее установки оператора.

Пробел (“_”)

Нажатие клавиши “_” разделяет элементы вводимой с клавиатуры информации, например, адрес и данные или два разных адреса. При записи в память или чтении из памяти процедура, вызываемая клавишей “_”, выполняет операцию “инкремент” для кода адреса.

Обращение к внутренним регистрам (“РГ")

После нажатия клавиши "РГ" микропроцессор выполняет процедуру обращения крегистрам и находится в режиме ожидания имени регистра. Для ввода имени регистра используется основная шестнадцатеричная клавиатура УМК. Список регистров и их обозначения на клавиатуре:

- "А"- регистр аккумулятор, поддерживающий работу АЛУ;
- " B,C,D,E,H,L"- регистры общего назначения;
- " SL,SH"- регистры младшего и старшего байта счетчика стека;
- "PL,PH"- регистры младшего и старшего байта счетчика команд.

После ввода имени регистра, в двух младших разрядах индикатора УМК появляется содержимое данного регистра, представленное в шестнадцатеричной системе счисления. В старших разрядах - имя регистра.

Очередность действий при модификации содержимого регистров:

- нажатие "РГ" для входа в процедуру обращения к регистрам ввод имени регистра чтение содержимого регистра на светодиодном индикаторе;
- набор нового байта данных на шестнадцатеричной клавиатуре;
- нажатие клавиши "-" для записи нового байта данных в регистр или подтверждения существующего байта данных и перехода к режиму ожидания имени нового регистра;
- нажатие клавиши "ВП" для подтверждения сделанных изменений и выходаиз процедуры.

Содержимое регистров сохраняется до нажатия клавиши “СБРОС”. Обращение к ячейкам памяти (“П”)

- Очередность действий при модификации содержимого ОЗУ:
- нажатие "П" для входа в процедуру обращения к памяти набор начального

адреса на шестнадцатеричной клавиатуре и ввод адреса нажатием клавиши "-";

- чтение содержимого данной ячейки ЗУ на светодиодном индикаторе;
- набор нового байта данных на шестнадцатеричной клавиатуре;

нажатие клавиши "-" для записи нового байта данных или подтверждения существующего и перехода к следующему адресу массива (операция инкремент для кода адреса);

нажатие клавиши "ВП" для подтверждения сделанных изменений в содержимом ячеек памяти и выход из процедуры.

Начать выполнение программы ("СТ")

Процедура "СТ" предназначена для запуска любой программы, расположенной в массиве ОЗУ или ПЗУ.

Очередность действий при запуске программы:

- нажатие "СТ" для входа в процедуру;

• набор начального адреса программы на шестнадцатеричной клавиатуре иввод адреса нажатием клавиши "-";

• набор конечного адреса программы на шестнадцатеричной клавиатуре;

• ввод адреса и запуск программы пользователя нажатием клавиши "ВП".

В момент нажатия клавиши "ВП" микропроцессор выходит из программ системного монитора и выполняет программу пользователя. После окончания выполнения программы пользователя микропроцессор возвращается в системный монитор и в старших разрядах индикатора появляется адрес ячейки ЗУ, где расположен последний код выполненной программы. До нажатия клавиши "СБРОС" состояние всех регистров микропроцессора соответствует их состоянию на момент окончания программы пользователя. Состояние регистровсохраняется в стеке и восстанавливается при обращении к ним.

Подсчет контрольной суммы ("КС")

Очередность действий при запуске процедуры определения контрольной суммы:

• нажатие "КС" для входа в процедуру;

• набор начального адреса массива ЗУ на шестнадцатеричной клавиатуре иввод адреса нажатием клавиши "-";

• набор конечного адреса массива ЗУ на шестнадцатеричной клавиатуре;

• ввод адреса и подсчет контрольной суммы при нажатии "ВП".

После выполнения директивы в младших разрядах индикатора выводится значение контрольной суммы. Директива используется для проверки правильности загрузки программ пользователя.

Запись константы ("ЗК")

Процедура предназначена для заполнения константой заданного массива ЗУ.Очередность действий при запуске процедуры для записи константы:

• нажатие "ЗК" для входа в процедуру;

• набор начального адреса массива ЗУ на шестнадцатеричной клавиатуре иввод адреса нажатием клавиши "-";

• набор конечного адреса массива ЗУ на шестнадцатеричной клавиатуре и вводнажатием клавиши "-";

• набор однобайтовой константы на шестнадцатеричной клавиатуре;

• ввод и запись в массив нажатием клавиши "ВП".Перемещение массива данных ("ПМ")

Очередность действий при запуске процедуры перемещения:

• нажатие "ПМ" для входа в процедуру;

• набор начального адреса перемещаемого массива ЗУ на шестнадцатеричнойклавиатуре и ввод адреса нажатием клавиши "-";

• набор конечного адреса перемещаемого массива ЗУ на шестнадцатеричной клавиатуре и ввод адреса нажатием клавиши "-";

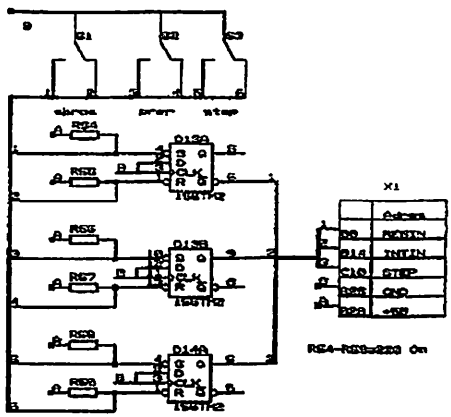
• набор начального адреса размещения на шестнадцатеричной клавиатуре;

• ввод адреса и выполнение перемещения нажатием клавиши "ВП".

При размещении программ в новом массиве ЗУ, например, при копировании в пространство ОЗУ системных программ, необходимо учитывать, что адреса ветвления и адреса, формируемые на основе заданных базовых адресов не будут модифицированы под новое адресное пространство, что приведет к ошибкам в работе программы.

Средства отладки программ, аппаратная и программная реализация шагового режима

Учебный микропроцессорный комплекс обладает набором средств для отладки и редактирования программ, представленных в машинных кодах. Для



облегчения этого процесса при ограниченных возможностях УМК рекомендуется использовать модульный принцип при подготовке программ в ассемблере. Используемый в лабораторном курсе ассемблер не имеет средств для компоновки программы из отдельных модулей. При модульном принципе создания сложных программ каждый модуль должен быть скомпилирован в отдельную программу и отлажен средствами УМК. Для отладки программных модулей в УМК предусмотрен режим пошагового выполнения программы по машинным циклам и по командам. В шаговом режиме к шине данных и шине адреса микропроцессора подключаются двоичные индикаторы (см. Рис. 1). Двоичные индикаторы шины адреса и шины данных разбиты на тетрады для удобства восприятия двоичных чисел в шестнадцатеричной форме. Кроме того, предусмотрен двоичный индикатор кода состояния микропроцессора.

Рис. 4.3. Схема формирования сигнала "STEP" и устранения дребезга

контактов аппаратными средствами

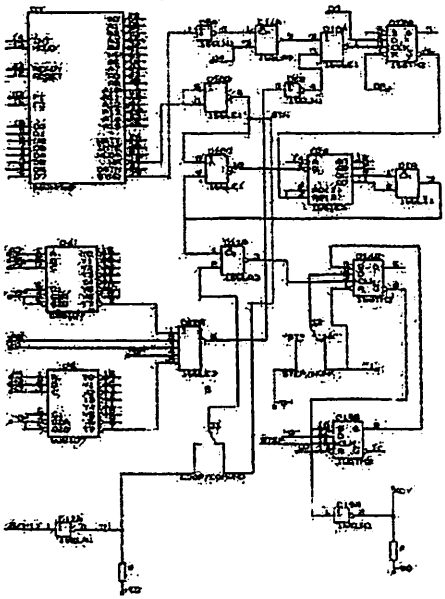


Рис. 4.4 Схема порта шагового режима

Шаговый режим выполнения программы устанавливается аппаратным путем при фиксации клавиши "РБ/ШГ" (Рис. 4) и может быть сброшен или восстановлен программным путем. Шаговый режим реализуется в микропроцессорных устройствах за счет использования внутреннего цикла ожидания микропроцессора. В цикл ожидания микропроцессор входит по сигналу на входе "RDY" ("ГОТОВНОСТЬ"). Окончательное формирование сигнала " RDY" происходит в системном генераторе КР580ГФ24. Формирование входного сигнала "RD" Y для системного генератора выполнено программно-аппаратными средствами. На Рис. 17 приведен фрагмент схемы формирования сигнала "RDY" и порта шагового режима, доступного для программного управления.

В шаговом режиме переключатель S2 коммутирует низкий логический уровень на вход D- триггера D14-1. Низкий логический уровень на выходе инвертора D-19 появится при наличии стробирующего фронта по входу С триггера D14-1 (переход из 1 в 0) с выхода элемента D11. Стробирующие импульсы для записи информации в триггер выбираются переключателем S1 и соответствуют машинным или командным циклам микропроцессора. Условием прохождения импульсов является наличие высокого уровня на

верхнем входе элемента D11. Высокий уровень на верхнем входе элемента D11 образуется после прохождения через счетчик D13 определенного числа импульсов машинных циклов МП (импульсов с выхода SYN МП). Задержка включения шагового режима необходима для выполнения в непрерывном режиме процедуры "СТ" системного монитора при запуске программы пользователя в шаговом режиме.

Запрет счета импульсов машинных циклов, т.е. фактически сброс шагового режима, происходит при наличии высокого уровня на инверсном выходе триггера D12. Элементы D22,D23,D20, D12 образуют порт шагового режима. Выполнение машинного цикла или команды в шаговом режиме происходит при нажатии клавиши "ШАГ", инициирующей управляющий сигнал " STEP". Процесс формирования сигнала "STEP" можно проследить по схемам на рис.4.2 и рис.4.3. Сигнал "STEP" выводит микропроцессор из режима ожидания. Возврат в режим ожидания происходит в момент прихода очередного импульса начала машинного цикла или начала команды.

Процесс отладки программ значительно упрощается при использовании подпрограмм или процедур, в т.ч. имеющихся в составе системного монитора. Основой для использования подпрограмм является условие сохранения (восстановления) состояния микропроцессора при выполнении подпрограммы или целенаправленное изменение этого состояния по результату работы подпрограммы. В микропроцессоре ВМ80 автоматически (на микропрограммном уровне) сохраняется только содержимое программного счетчика (" PC"), т.е. адрес возврата из подпрограммы. Для этого используется стековая память, организованная в массиве ОЗУ с помощью регистра стека "SP".

Для сохранения содержимого других регистров МП используются программные средства обращения к стеку. При начальной установке микропроцессора в регистр стека загружается адрес, являющийся нижней границей ОЗУ. Каждое обращение к стеку для записи информации уменьшает содержимое регистра "SP" на единицу. Каждое обращение для чтения информации из стека увеличивает содержимое регистра "SP" на единицу. При выполнении операции " RET" микропроцессор считывает в качестве адреса возврата текущие значения двух байтов из вершиныстека.

Практическая часть

НАЗВАНИЕ	МНЕМОНИКА	ФОРМАТ	ФУНКЦИЯ	РЕГИСТРОВАЯ ПАРА
Загрузка A из памяти	LDA adr (адрес ячейки – вариант студента)	3	A – (adr)	BC, DE
Сохранение A в память	STA adr (адрес ячейки – вариант студента)	3	(adr) – A	BC, DE
Межрегистровая пересылка	MOV r1, r2	1	r1 – r2	
Загрузка непосредственного 16-битного операнда	MVI r, D16	2	r – D8	
Загрузка непосредственного 16-битного операнда	LXI rp, D16	3	rp – D16	BC, DE, HL, SP

Контрольные вопросы.

1. Что составляет основу аппаратных и программных средств УМК?
2. Какой объем имеет ПЗУ и ОЗУ?
3. За счет чего производится расширение аппаратных возможностей?
4. Как формируется сигнал RESET?

Практическая работа №5
Изучение характеристик типов памяти и процессоров обмена информацией

Цель работы: выполнить программы записи данных в память из регистра, изучить команды чтения/записи в память с косвенной адресацией.

Теоретические сведения

Память представляется как последовательность ячеек размером в один байт. Каждая ячейка имеет свой адрес в диапазоне от 0 до 65535. Для удобства обычно используется шестнадцатеричное значение адреса, тогда диапазон адресации составляет ***0000h – FFFFh***.

В микропроцессорной системе адресации адрес ячейки памяти указывается в самой команде во втором и третьем байтах команды (прямая адресация). В общем виде это выглядит следующим образом

КОП ad16, где КОП – код операции (чтение или запись); ***ad16*** – адрес ячейки памяти. В памяти такая команда будет размещена следующим образом

КОП ad16 (младший байт) ad16 (старший байт), т.е. после байта кода операции располагается сначала младший байт адреса, а затем – старший. Косвенная адресация предполагает, что адрес ячейки памяти будет располагаться в регистровых парах ***HL, DE, BC***. Для каждой конкретной команды работы с памятью закреплена своя регистровая пара. Таким образом, прежде чем выполнить такую команду необходимо сначала задать адрес в соответствующей регистровой паре.

Например,

LXI H, 0800h

MOV M, A; запись в память содержимое регистра ***A*** по адресу, находящемуся в регистровой паре ***HL*** или LXI D, 0900H ***STAX D***; запись в память содержимое регистра ***A*** по адресу, находящемуся в регистровой паре ***DE***

Программа работы

1. Команды записи в память с прямой адресацией.

Существуют две команды прямой адресации записи в память:

STA ad16 запись в память по прямому адресу ***ad16*** содержимого регистра ***A***;

SHLD ad16 запись в память содержимого регистровой пары ***HL***. Причем по адресу ***ad16*** будет записано содержимое регистра ***L***, а по адресу ***ad16+1*** будет записано содержимое регистра ***H***.

Пример: Запишите в память, начиная с адреса ***0000h***, коды следующих команд, используя прямую адресацию (табл. 5.1).

Таблица 5.1

Пример кода программы

Адрес	Команда	Машинный код	Комментарий
0000	***MVI A,FFh***	3E FF	Запись в аккумулятор значения ***FFh***
0002	***STA 0110h***	32 10 01	Запись в память содержимого ре-гистра ***A*** по адресу ***0110h***
0005	***LXI H,3536h***	21 36 35	Загрузка регистровой пары ***HL*** числом ***3536h***. Младший байт данных загружается в регистр ***L***,а старший байт – в регистр ***H***.
0008	***SHLD 0150h***	22 50 01	Запись в память содержимого регистра ***L*** по адресу ***0150h***, содержимого регистра ***H*** по адресу ***0151h***

Выполните эту последовательность команд в пошаговом режиме и пронаблюдайте, как изменяется содержимое регистров ***A, H, L*** и содержимое ячеек памяти ***0110h, 0150h, 0151h***. Значения регистров и ячеек памяти должны быть следующими

A = FFh; H = 35h; L = 36h; (0110h) = FFh; (0150h) = 36h; (0151h) = 35h

2. Команды чтение памяти с прямой адресацией.

Аналогично командам записи с прямой адресацией существуют две команды чтения памяти с конкретным адресом

LDA ad16 загрузка регистра ***A*** из ячейки памяти с адресом ***ad16***;

LHLD ad16 чтение памяти по прямому адресу ***ad16*** в регистровую пару ***HL***. При этом в регистр ***H*** будет записано содержимое ячейки с адресом ***ad16+1***, а в регистр ***L*** содержимое ячейки памяти с адресом ***ad16***.

Пример: Запишите в память по адресу ***0000h*** коды следующих команд (табл. 3.16).

Таблица 5.2

Пример кода программы

Адрес	Команда	Машинный код	Комментарий

0000	***LDA 0190h***	3A 90 01	Чтение в регистр *A* содержимогоячейки с адресом ***0190h***
0003	***LHLD 0190h***	2A 90 01	Чтение в регистр *L* содержимого ячейки с адресом ***0190h***, а в регистр *H* содержимого ячейки с адресом ***0191h***

Вручную внесите в ячейки памяти следующие значения

(0190h) = ABh; (0191h) = CDh.

Выполните эту последовательность команд в пошаговом режиме и пронаблюдайте, как изменяется содержимое регистров *A, H, L*. Значения регистров должны быть следующими:

A = ABh; H = CDh; L = ABh.

Команды чтения/записи в память с косвенной адресацией.

Общий вид команды

MOV M, R запись в память содержимого регистра;

MOV R, M загрузка регистра из ячейки памяти, адрес, которыйнаходится в регистровой паре *HL*. *R* – регистр общего назначения *A, B, C, D, E, H, L*.

Пример: Запишите в память, начиная с адреса ***0000h***, коды следующей программы (табл. 3.17).

Таблица 5.3

Пример кода программы

Адрес	Команда	Машинный код	Комментарий
0000	***MVI A, 0AAh***	3E AA	загрузка регистров
0002	***MVI B, 0BBh***	06 BB	
0004	***MVI C, 0CCh***	0E CC	
0006	***MVI D, 0DDh***	16 DD	
0008	***MVI E, 0EEh***	1E EE	
000A	***LXI H, 0100h***	21 00 01	загрузка ***HL=0100h***, адрес ***M***
000D	***MOV M, A***	77	запись в ***M=A***, по адресу ***HL***
000E	***LXI H, 0101h***	21 01 01	
0011	***MOV M, C***	71	
0012	***LXI H, 0102h***	21 02 01	
0015	***MOV M, B***	70	

0016	***LXI H, 0103h***	21 03 01	
0019	***MOV M, E***	73	
001A	***LXI H, 0104h***	21 04 01	
001D	***MOV M, D***	72	
001E	***LXI H, 0105h***	21 05 01	
0021	***MOV M, H***	74	
0022	***LXI H, 0106h***	21 06 01	
0025	***MOV M, L***	75	

Выполните эту последовательность команд. Значения ячеек памяти должны быть следующими

0100h = AAh; 0101h = CCh; 0102h = BBh; 0103h = EEh; 0104h =DDh; 0105h =01h;0106h = 06h

Пример: Запишите в память, начиная с адреса ***0000h***, коды следующей программы (табл. 5.4).

Таблица 5.4

Пример кода программы

Адрес	Команда	Машинный код	Комментарий
0000	***LXI H, 0100h***	21 00 01	загрузка ***HL=0100h***, адрес ***M***
0003	***MOV E, M***	5E	чтение ***E=M***, по адресу ***HL***
0004	***LXI H, 0101h***	21 01 01	и т.д.
0007	***MOV D, M***	56	
0008	***LXI H, 0102h***	21 02 01	
000B	***MOV C, M***	4E	
000C	***LXI H, 0103h***	21 03 01	
000F	***MOV B, M***	46	
0010	***LXI H, 0104h***	21 04 01	
0013	***MOV A, M***	7E	
0014	***LXI H, 0105h***	21 05 01	
0017	***MOV H, M***	66	
0018	***LXI H, 0106h***	21 06 01	

001B	*MOV L, M*	6E	

Заполните вручную соответствующие ячейки памяти (*0100h = AAh,0101h = CCh, 0102h = BBh, 0103h = EEh, 0104h = DDh, 0105h = 01h, 0106h = 06h*). Выполните эту последовательность команд. Значения регистров должны быть следующими

A = DDh; B = EEh; C = BBh; D = CCh; E = Aah; H = 01h; L = 06h

1. Команды чтения/записи при адресации через регистровые пары *BC, DE.*

STAX B запись содержимого регистра *A* в память, адрес в регистровой паре *BC*;

STAX D запись содержимого регистра *A* в память, адрес в регистровой паре *DE*

LDAX B чтение содержимого памяти в регистр *A*, адрес в регистровой паре *BC*;

LDAX D чтение содержимого памяти в регистр *A*, адрес в регистровой паре *DE*.

Пример: Запишите в память, начиная с адреса 0000h, коды программы (табл. 3.19).

Таблица 5.5 *Пример кода программы*

Адрес	Команда	Машинный код	Комментарий
0000	***LXI B, 0100h***	01 00 01	Загрузка *BC←0100h*
0003	***MVI A, 0Fh***	3E 0F	Загрузка *A←0Fh*
0005	***STAX B***	02	Запись в *M←A* по адресу *BC*
0006	***LXI D, 0110h***	11 10 01	Загрузка в *DE←0110h*
0009	***MVI A, 0F0h***	3E F0	Загрузка в *A←F0h*
000B	***STAX D***	12	Запись в *M←A* по адресу *DE*

Выполните эту последовательность команд. Значения ячеек памяти должны быть следующими

0100h = 0Fh, 0110h = F0h.

Пример: Запишите в память, начиная с адреса 0000h, коды программы (табл. 3.20).

Таблица 5.6. *Пример кода программы*

Адрес	Команда	Машинный код	Комментарий
0000	***LXI D, 0100h***	11 00 09	загрузка в *DE←0100h*

0003	***LDAX D***	1A	чтение в ***A←M*** по адресу ***DE***
0004	***MOV L, A***	6F	пересылка ***L←A***
0005	***LXI B, 0110h***	01 10 09	загрузка в ***BC←0110h***
0008	***LDAX B***	0A	чтение в ***A←M*** по адресу ***BC***
0009	***MOV H, A***	67	;пересылка ***H←A***

Заполните вручную соответствующие ячейки памяти (***0100h←0Fh, 0110h←F0h***). Выполните эту последовательность команд. Значения регистров должны быть следующими

H←F0h, L←0Fh.

Практическая часть

1. Напишите и выполните программу записи данных в память из регистра ***A***, в соответствии с табл. 3.21. Для этого используйте команду загрузки регистра ***A*** и команды записи в память регистра ***A*** по прямому адресу.

Таблица 5.7

Варианты заданий записи данных в память

Вариант 1	Адрес	*0100*	*0105*	*0107*	*010C*	*0120*	*0126*
	Данные	*00h*	*01h*	*05h*	*0Ah*	*BBh*	*12h*
Вариант 2	Адрес	*0200*	*0202*	*0205*	*020C*	*0215*	*0220*
	Данные	*25h*	*12h*	*50h*	*A0h*	*BCh*	*1Dh*
Вариант 3	Адрес	*0101*	*0110*	*0120*	*0130*	*0135*	*0140*
	Данные	*15h*	*A1h*	*5Ah*	*A6h*	*00h*	*21h*
Вариант 4	Адрес	*0100*	*0105*	*0107*	*010B*	*0123*	*0126*
	Данные	*22h*	*33h*	*44h*	*55h*	*AAh*	*CCh*
Вариант 5	Адрес	*0200*	*0210*	*0220*	*0226*	*0228*	*0240*
	Данные	*FFh*	*10h*	*01h*	*03h*	*0Bh*	*12h*
Вариант 6	Адрес	*0100*	*0105*	*010C*	*0115*	*0120*	*0130*
	Данные	*66h*	*12h*	*05h*	*0Ah*	*D0h*	*D3h*
Вариант 7	Адрес	*0100*	*0101*	*0111*	*0112*	*0120*	*0121*
	Данные	*00h*	*07h*	*09h*	*0Bh*	*B0h*	*12h*
Вариант 8	Адрес	*0200*	*0201*	*0205*	*0206*	*0210*	*0211*
	Данные	*25h*	*12h*	*50h*	*0Ah*	*0Ch*	*D1h*
Вариант 9	Адрес	*0101*	*0102*	*0120*	*0121*	*0135*	*0136*
	Данные	*15h*	*1Ah*	*A5h*	*6Ah*	*00h*	*21h*
Вариант 10	Адрес	*0100*	*0101*	*0107*	*0108*	*0123*	*0124*
	Данные	*22h*	*33h*	*44h*	*55h*	*AAh*	*CCh*
Вариант 11	Адрес	*0210*	*0211*	*0220*	*0221*	*0228*	*0229*
	Данные	*FFh*	*1Ah*	*A1h*	*03h*	*B0h*	*12h*

Вариант 12	Адрес	0150	0151	0161	0162	0200	0201
	Данные	66h	B2h	05h	A0h	0Dh	3Dh
Вариант 13	Адрес	0100	0101	0111	0115	0120	0121
	Данные	22h	17h	09h	0Ah	80h	90h
Вариант 14	Адрес	0150	0201	0202	0210	0212	0213
	Данные	35h	42h	1Ah	0Ah	C0h	D3h
Вариант 15	Адрес	0101	0102	0120	0125	0135	0136
	Данные	16h	1Ah	50h	60h	0Ah	2Dh
Вариант 16	Адрес	0100	0105	0107	0108	0110	0124
	Данные	25h	13h	43h	56h	0Ah	C0h
Вариант 17	Адрес	0210	0211	0220	0221	0228	0240
	Данные	0Fh	10h	12h	D3h	B1h	1Ch
Вариант 18	Адрес	0150	0151	0160	0162	0200	0201
	Данные	65h	B4h	05h	A1h	0Ch	35h
Вариант 19	Адрес	0100h	0101h	0108h	0109h	0121h	0122h
	Данные	01h	05h	25h	10h	1Bh	2Ah
	Регистр	B	C	D	E	L	H
Вариант 20	Адрес	0200h	0201h	0215h	0216h	0220h	0221h
	Данные	35h	26h	51h	0A1h	0B0h	11h
	Регистр	B	C	D	E	L	H
Вариант 21	Адрес	0101h	0102h	0125h	0126h	0135h	0136h
	Данные	16h	29h	60h	0A7h	3Bh	A1h
	Регистр	B	C	D	E	L	H
Вариант 22	Адрес	0110h	0111h	0150h	0151h	0212h	0213h
	Данные	99h	35h	88h	56h	5Ah	0Ch
	Регистр	B	C	D	E	L	H
Вариант 23	Адрес	0200h	0201	0221	0222h	0230h	0231
	Данные	0F5h	0EEh	1Eh	1Ah	0BBh	33h
	Регистр	B	C	D	E	L	H
Вариант 24	Адрес	0105h	0106h	010Ch	010Dh	0183h	0184h
	Данные	19h	38h	96h	0A1h	16h	13h
	Регистр	B	C	D	E	L	H
Вариант 25	Адрес 1	0100h	0101h	0108h	0109h	0121h	0125h
	Адрес 2	0200h	0201h	0208h	0209h	0235h	0240h
	Данные	11h	51h	25h	1Ah	10h	2Bh
Вариант 26	Адрес 1	0200h	0201h	0215h	0216h	0220h	0229h
	Адрес 2	0250h	0251h	0208h	0209h	0135h	0140h
	Данные	35h	26h	51h	A1h	B0h	11h

Вариант 27	Адрес 1	*0101h*	*0102h*	*0120h*	*0126h*	*0135h*	*0136h*
	Адрес 2	*0300h*	*0301h*	*0308h*	*0310h*	*0335h*	*0336h*
	Данные	*26h*	*29h*	*61h*	*A8h*	*4Bh*	*11h*
Вариант 28	Адрес 1	*0110h*	*0111h*	*0150h*	*0180h*	*0212h*	*0213h*
	Адрес 2	*0200h*	*0201h*	*0205h*	*0207h*	*0250h*	*0251h*
	Данные	*9Ah*	*35h*	*90h*	*58h*	*6Ah*	*1Ch*
Вариант 29	Адрес 1	*0200*	*0201*	*0221*	*0222h*	*0230h*	*0240h*
	Адрес 2	*0100*	*0101*	*0121*	*0122h*	*0140h*	*0150h*
	Данные	*F7h*	*E0h*	*10h*	*A6h*	*0Bh*	*33h*
Вариант 30	Адрес 1	*0105h*	*0106h*	*010Ch*	*010Dh*	*0180h*	*0190h*
	Адрес 2	*0210h*	*0211h*	*0220h*	*0221h*	*0290h*	*0300h*
	Данные	*1Ah*	*3Dh*	*96h*	*A1h*	*1Eh*	*1Dh*

Практическое занятие №6.

Организация соединения устройств с шиной

Цель работы: Получить представление и навыки о соединения устройств с шиной.

Теоретическая часть

Виды шин

•□Шины данных - все шины, которые используются для передачи данных между процессором компьютера и периферией. Для передачи могут использоваться как последовательный, так и параллельный методы, можно передавать от одного до восьми бит за один раз. По размеру данных, которые можно передать за один раз такие шины делятся на 8, 16, 32 и даже 64 битные;

•□Адресные шины - связаны с определенными участками процессора и позволяют записывать и читать данные из оперативной памяти;

•□Шины питания - эти шины питают электричеством различные, подключенные к ним устройства;

•□Шина таймера - эта шина передает системный тактовый сигнал для синхронизации периферийных устройств, подключенных к компьютеру;

•□Шина расширений - позволяет подключать дополнительные компоненты, такие как звуковые или ТВ карты;

Наиболее распространенные типы шин в компьютере для расширений:

•□ISA - Industry Standard Architecture;

•□EISA - Extended Industry Standard Architecture;

•□MCA - Micro Channel Architecture;

•□VESA - Video Electronics Standards Association;

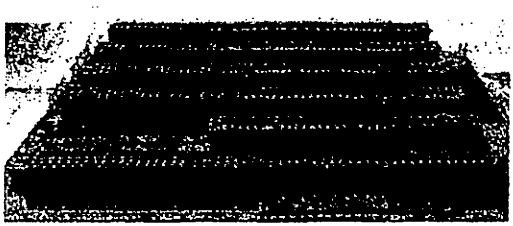
•□PCI - Peripheral Component Interconnect;

•□PCI-E - Peripheral Component Interconnect Express;

•□PCMCIA - Personal Computer Memory Card Industry Association (также известна как PC bus);

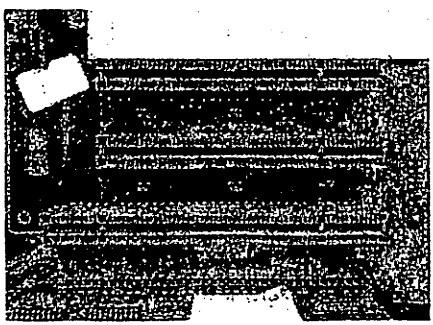
•□AGP - Accelerated Graphics Port;

•□SCSI - Small Computer Systems Interface.



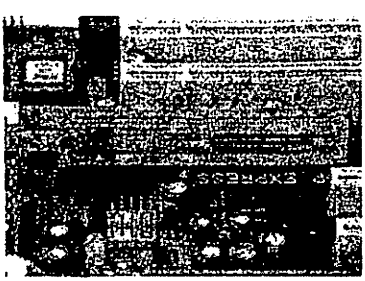
ШИНА ISA

Раньше это был наиболее распространенный тип шины расширения. Он был разработан компанией IBM для использования в компьютере IBM PC-XT. Эта шина имела разрядность 8 бит. Это значит, что можно было передавать 8 бит или один байт за один раз. Шина работала с тактовой частотой 4,77 МГц. Для процессора 80286 на базе IBM PC-AT была сделана модификация конструкции шины, и теперь она могла передавать 16 бит данных за раз. Иногда 16 битную версию шины ISA называют AT.



ШИНА MCA

Компания IBM разработала эту шину в качестве замены для ISA, для компьютера PS/2. Шина получила еще больше усовершенствований по сравнению с ISA. Например, была увеличена частота до 10 МГц, а это привело к увеличению скорости, а также шина могла передавать 16 или 32 бит данных за раз. Также была добавлена технология Bus Mastering. На плате каждого расширения помещался мини-процессор, эти процессоры контролировали большую часть процессов передачи данных освобождая ресурсы основного процессора.



ШИНА AGP

Необходимость передачи видео высокого качества с большой скоростью привела к разработке AGP. Accelerated Graphics Port (AGP) подключается к процессору и работает со скоростью шины процессора. Это значит, что видеосигналы будут намного быстрее передаваться на видеокарту для обработки. AGP использует оперативную память компьютера для хранения 3D изображений. По сути, это дает видеокарте неограниченный объем видеопамяти. Чтобы ускорить передачу данных Intel разработала AGP как прямой путь передачи данных в память. Диапазон скоростей передачи – 264 Мбит до 1,5 Гбит.

Порт

•Много разрядный вход и/или выход в компьютер, в т.ч. – стандартное устройство сопряжения ввода-вывода ;

•Разъем - место подсоединения внешнего (периферийного) устройства;

•Точка взаимодействия двух процессов;

•Конец логического канала.

В ПЭВМ различают порты следующих типов:

•параллельный порт - Разновидность портов, обеспечивающая относительно большую скорость ввода и вывода данных за счет того,

что биты передаются через них одновременно (параллельно);

•используются в частности для подключения принтеров;

имеет обозначение LTP с порядковым номером - LTP1, LTP2 и LTP3;

•последовательный порт

•Разновидность портов, обеспечивающая асинхронное подсоединение устройств.

Например - мыши, модема и др.; Имеет обозначение COM с порядковым номером - COM1, COM2, COM3, COM4.

Некоторые периферийные устройства могут подключаться как к параллельным, так и последовательным портам; игровой порт - Порт, снабженный специализированным аналогово-цифровыми преобразователем и предназначенный для подключения одного или двух джойстиков; аналоговый порт - Стандартный разъем RJ-11, к которому подключаются аналоговые устройства, например, телефон, факс, модем и т.д.

AGP (Accelerated Graphics Port) - " Ускоренный графический порт", также известный под именем FireWare - " Огненный провод". Соответствует спецификации IEEE-1394, ориентированной на подключение к ПЭВМ широкополосных и быстродействующих периферийных устройств (например, видеокамер, наборов для редактирования видео в домашних условиях и т.п.) и более реалистичного отображения трехмерной графики.

Ускорение работы программ, интенсивно использующих графику (например, для трехмерного моделирования), достигается за счет организации прямого доступа видеоадаптера к основной памяти компьютера. Предполагается, в частности, должны появиться видео платы расширения FireWare, способные передавать данные со скоростью 200-400 Мбит/с. К этому времени прогнозируется также начало выпуска ПЭВМ с портом AGP. Для работы с AGP потребуются процессоры типа Pentium Pro и ОЗУ объемом от 24 до 32 Мбайт

ECP (Enhanced Capabilities Port) - Одна из двух (см. ниже " EPP ") модификаций параллельного порта, ориентированная на работу с высокоскоростными периферийными устройствами (например, лазерными принтерами). Обеспечивает скорость передачи данных 2.5 Мбайт/с

EPP (Enhanced Parallel Port) - Одна из двух (см. выше "ECP") модификаций параллельного порта, ориентированная на работу с высокоскоростными периферийными устройствами (например, лазерными принтерами). Обеспечивает скорость передачи данных 2.5 Мбайт/с

Задания

1. Идентифицируйте внутренние интерфейсы системной платы.
2. Дайте сравнительную характеристику внутренних интерфейсов целевой системной платы.
3. Программными средствами идентифицируйте внутренние интерфейсы целевого компьютера. Дайте их сравнительную характеристику. Выявите достоинства и недостатки.
4. Оформить отчет по проделанной работе.

Контрольные вопросы.

1. Какие типы внутренних интерфейсов вам известны?
2. Дайте сравнительную характеристику шины ISA
3. Дайте сравнительную характеристику шины PCI
4. Дайте сравнительную характеристику шины AGP

Практическая работа №7

Команды загрузки регистров. Команды пересылки

Цель работы: написать и выполнить программу загрузки регистров

Теоретические сведения

В микропроцессоре ***Intel 8080*** для программирования доступны 16-разрядный счетчик команд (***PC***) содержит адрес команды, которая подлежит выполнению;

- 16-разрядный регистр указатель стека (***SP***), определяет адрес специализированной области ОЗУ-стека;
- 8-разрядный регистр-накопитель (***A***), используется для хранениях накопления результата в арифметических, логических операциях, а также в операциях ввода-вывода и сдвига. Кроме того, он может быть использован в качестве регистра общего назначения для хранения данных;
- шесть 8 –разрядных регистров общего назначения (РОН):

B,C,D,E,H,L;

- 8-разрядный регистр признаков ***PSW*** (***F***) содержит биты условий: ***C*** – перенос, ***AC*** – вспомогательный перенос, ***S*** – знак, ***Z*** – ноль, ***P*** – четность. Данные биты устанавливаются в зависимости от результата операции при выполнении арифметических, логических команд, команд сдвига и сравнения. Распределение условий в байте признаков показано в табл. 3.3.

Таблица 7.3. *Распределение битов в байте состояния*

7	6	5	4	3	2	1	0
S	***Z***	***0***	***AC***	***0***	***P***	***1***	***C***

Регистры общего назначения могут использоваться для манипуляции 16-разрядными данными. Для этого регистры объединяются в пары следующим образом: ***BC, DE, HL***, где соответственно, первый регистр используется для хранения старшего байта (например, ***B***), а второй –для хранения младшего байта (***C***), а также ***PSW*** (***A*** + ***C***) слово состояния программы.

Программа работы

1. Команды загрузки регистров общего назначения

Общий вид команды: ***MVI R, d8,*** где ***R*** – идентификатор регистра: ***A, B, C, D, E, H, L***, а ***d8*** – непо-средственный операнд (байтовое число).

Пример: Запишите в память, начиная с адреса ***0000h***, последовательность команд, представленных в табл. 7.4.

Таблица 7.4. *Программа загрузки внутренних регистров*

Адрес	Команда	Машинный код	Комментарий
0000	***MVI A,00***	3E 00	загрузка регистра ***A←00h***
0002	***MVI B,01***	06 01	загрузка регистра ***B←01h***
0004	***MVI C,02***	0E 02	загрузка регистра ***C←02h***
0006	***MVI D,03***	16 03	загрузка регистра ***D←03h***
0008	***MVI E,04***	1E 04	загрузка регистра ***E←04h***
000A	***MVI H,05***	26 05	загрузка регистра ***H←05h***
000C	***MVI L,06***	2E 06	загрузка регистра ***L←06h***

Выполнить эту последовательность команд в пошаговом режиме и пронаблюдать, как изменяется содержимое регистров общего назначе- ния ***A, B, C, D, E, H, L***. Значения регистров должны быть следующими:

A = 00h; B = 01h; C = 02h; D = 03h; E = 04h; H = 05h; L = 06h.

1. Команды загрузки регистров 16-разрядными данными

Общий вид команды: ***LXI R, <dh dl>***

где ***R*** – идентификатор пары регистров: ***B, D, H;dh*** – старший байт 16-разрядного операнда;

dl – младший байт 16-разрядного операнда.

Пример: Записать в память, начиная с адреса ***0000h***, последовательность команд, представленных в табл. 7.5.

Таблица 7.5. *Программа загрузки регистровых пар*

Адрес	Команда	Машинный код	Комментарий
0000	***LXI B, 3132h***	01 32 31	Загрузка регистровой пары ***BC*** чис- лом ***3132h***. Младший байт данныхзагружается в регистр ***C***, а старший байт – в регистр ***B***.

00 03	*LXI D,* *3334h*	11 34 33	Загрузка регистровой пары ***DE*** чис- лом ***3334h***. Младший байт данныхзагружается в регистр ***E***, а старший байт – в регистр ***D***.
00 06	*LXI H,* *3536h*	21 36 35	Загрузка регистровой пары ***HL*** чис- лом ***3536h***. Младший байт данныхзагружается в регистр ***L***, а старший байт – в регистр ***H***.

Примечание: в памяти располагается сначала младший байт операнда, затем – старший.

Выполнить выше описанную последовательность команд в пошаговом режиме и пронаблюдать, как изменяется содержимое регистров***B, C, D, E, H, L***. Значения регистров должны быть следующими:

B = 31h; C = 32h; D = 33h; E = 34h; H = 35h; L = 36h.

2. Команды загрузки регистра указателя стека.

Команда непосредственной загрузки регистра указатель стека имеет вид: ***LXI SP, <dh dl>,***

где *dh* – старший байт 16-разрядного операнда;

dl – младший байт 16-разрядного операнда.

Пример: Записать в память, начиная с адреса ***0000h***, последовательность команд, представленных в табл. 7.6.

Таблица 7.6. *Программа загрузки указателя стека*

Адрес	Команда	Машинный код	Комментарий
0000	***LXI SP, 0B01h***	31 01 0B	загрузка указателя стека: ***SP←0B01h***
0003	***LXI SP, 0100h***	31 00 01	загрузка указателя стека: ***SP←0100h***

Выполнить эту последовательность команд в пошаговом режиме и просмотрите содержание регистра указателя стека после каждого шага выполнения программы.

После первого шага, т.е. после выполнения микропроцессором первой команды, содержимое регистра указателя стека должно бытьравным ***SP = 0B01h***.

После второго шага, т.е. после выполнения микропроцессором второй команды, содержимое регистра указателя стека должно быть равным ***SP =***

0100h.

Команда косвенной загрузки регистра указатель стека имеет вид:

SPHL

По этой команде в указатель стека загружается содержимое регистровой пары *HL*. Поэтому, чтобы в указатель стека загрузить, например, число *0210h*, его предварительно надо загрузить в регистровую пару *HL*.

Пример: Записать в память, начиная с адреса *0000h*, последовательность команд, представленных в табл. 7.7.

Таблица 7.7

Программа косвенной загрузки указателя стека

Адрес	Команда	Машинный код	Комментарий
0000	***LXI H, 0210h***	21 10 02	загрузка *HL←0210h*
0003	***SPHL***	F9	загрузка *SP←HL*

Выполнить эту последовательность команд в пошаговом режиме и просмотрите содержание регистра указателя стека после выполнения программы. После выполнения содержимое регистра указателя стекадолжно быть равным *SP = 0210h*.

3. Команды пересылки.

Общий вид команды: ***MOV R1, R2,***

где *R1* – идентификатор регистра получателя: *A, B, C, D, E, H, L*;*R2* – идентификатор регистра источника: *A, B, C, D, E, H, L.*

Пример: Записать в память, начиная с адреса *0000h*, последовательность команд, представленных в табл. 7.8.

Таблица 7.8

Программа пересылки данных между регистрами

Адрес	Команда	Машинный	Комментарий
0000	***MVI A, FFh***	3E FF	загрузка регистра *A←FFh*
0002	***MOV B, A***	47	пересылка B←*A*
0003	***MOV C, B***	48	пересылка *C←B*
0004	***MOV D, C***	51	пересылка *D←C*
0005	***MOV E, D***	5A	пересылка *E←D*
0006	***MOV H, E***	63	пересылка *H←E*
0007	***MOV L, H***	6C	пересылка *L←H*

Выполнить эту последовательность команд в пошаговом режиме и пронаблюдать, как изменяется содержимое регистров *A, B, C, D, E, H, L*. Значения регистров должны быть следующими:

A = FFh; B = FFh; C = FFh; D = FFh; E = FFh; H = FFh; L =FFh.

4. Команда загрузки счетчика команд.

Общий вид команды: ***PCHL***

По этой команде в счетчик команд записывается содержимое пары регистров ***HL***. Т.о., для того, чтобы загрузить в счетчик команд адрес ***0100h***, необходимо сначала это число загрузить в регистровую пару ***HL***.

Пример: Записать в память по адресу ***0000h***, последовательность команд, представленных в табл. 7.9.

Таблица 7.9

Программа косвенной загрузки программного счетчика

Адрес	Команда	Машинный код	Комментарий
0000	***LXI H, 0100h***	21 00 01	***HL←0100h***
0003	***PCHL***	E9	***PC ← HL***

Выполнить эту последовательность команд в пошаговом режиме и пронаблюдайте, как изменяется содержимое регистров ***H, L*** и программного счетчика ***PC***.

Контрольные задания

1. Написать и выполнить программу загрузки регистров в соответствии с табл. 7.10.

Таблица 7.10

Варианты заданий загрузки регистров

Вариант	1	2	3	4	5	6
	B:←F0h	*B:←01h*	*B:←AEh*	*B:←21h*	*B:←23h*	*B:←BBh*
	C:←33h	*C:←35h*	*C:←FB h*	*C:←16h*	*C:←45h*	*C:←CC h*
	D:←EE h	*D:←EA h*	*D:←35h*	*D:←E1 h*	*D:←10h*	*D:←D1 h*
	E:←AAh	*E:←A1h*	*E:←26h*	*E:←D5 h*	*E:←62h*	*E:←EEh*
	H:←00h	*H:←A1h*	*H:←16h*	*H:←01h*	*H:←A5h*	*H:←AA h*
	L:←19h	*L:←18h*	*L:←AAh*	*L:←20h*	*L:←97h*	*L:←FFh*
	A:←FFh	*A:←F5h*	*A:←FEh*	*A:←25h*	*A:←F1h*	*A:←43h*
Вариант	7	8	9	10	11	12

	B:←23h	B:←23h	B:←0Eh	B:←11h	B:←23h	B:←8Bh
	C:←45h	C:←45h	C:←0Bh	C:←66h	C:←33h	C:←1Ch
Вариант	**13**		**14**		**15**	
	BC←FFFFh		BC←00FFh		BC←0000h	
	DE←0123h		DE←0124h		DE←0F0Fh	
	HL←55AAh		HL←5555h		HL←1579h	
Вариант	**16**		**17**		**18**	
	BC←0E0Eh		BC←1100h		BC←DDDDh	
	DE←E0E0h		DE←4545h		DE←ABCDh	
	HL←000Eh		HL←536Ah		HL←DCBAh	
Вариант	**19**		**20**		**21**	
	BC←13EFh		BC←1234h		BC←FEDCh	
	DE←A734h		DE←5678h		DE←BA98h	
	HL←1FA9h		HL←9ABCh		HL←7654h	
Вариант	**22**		**23**		**24**	
	BC←3210h		BC←2468h		BC←0000h	
	DE←35DFh		DE←ACE2h		DE←1111h	
	HL←5555h		HL←468Ah		HL←2222h	
Вариант	**25**		**26**		**27**	
	SP←0100h		SP←0101h		SP←0802h	
	SP←0200h		SP←0202h		SP←0812h	
	SP←0300h		SP←0203h		SP←0822h	
Вариант	**28**		**29**		**30**	
	SP←0200h		SP←0800h		SP←0702h	

Проверить правильность выполнения программы.

Практическая работа №8
Организация обмена информацией с внешними устройствами

Целью работы: исследование методов подключения и организации обмена информацией с простейшими устройствами ввода-вывода. Изучение программных способов маскирования данных и организации условных переходов в микроЭВМ.

Теоретическая часть

К командам ввода-вывода МП i8080A относятся команды IN <A1> и OUT <A1>. При выполнении команды IN <A1> микроЭВМ считывает число из входного устройства с адресом <A1> и записывает его в аккумулятор. При выполнении команды OUT <A1> МП записывает число из аккумулятора в выходное устройство с адресом <A1>. Так как адрес устройства указывается в одном байте, то с помощью этих команд микроЭВМ может обмениваться информацией не более чем с 256 внешними устройствами.

Обмен данными между микроЭВМ и внешними устройствами может вызываться как в определенных местах в программе, так и по сигналам прерывания. В последнем случае подпрограмма обмена данными с внешним устройством будет вызываться за счет перевода микроЭВМ в режим обслуживания прерывания.

Переключатели стенда используются для имитации передачи данных от внешнего устройства. К входному устройству (с адресом 05h или 20h) подключены светодиоды (I0 - I7) для индикации чисел, поступающих в него. Светодиоды O0 - O7 указывают число, записанное в выходном устройстве.

Простейшая программа (программа 6) перезаписи числа с входного устройства (с адресом 05h) в выходное устройство (с адресом 05h или 30h) имеет вид:

Программа 1

Адрес	Код	Метка	Мнемокод	Комментарий
0800	DB 05	CNT:	IN 05h	; Записать число из входного устройства с ; адресом 05h в аккумулятор
0802	D3 05		OUT 05h	; Записать число из аккумулятора в ; выходное устройство с адресом 05h
0804	C3 00 08		JMP CNT	; Идти на CNT

Организация условных переходов в микроЭВМ осуществляется с помощью регистра признаков МП (см. выше).

Регистр признаков имеет пять разрядов, каждый из которых устанавливается по определенному правилу в соответствии с выполнением МП последней команды (см. приложение).

Во многих случаях при выполнении программ необходимо проверять или изменять (маскировать) состояние одного или нескольких разрядов числа в аккумуляторе. Это можно осуществить с помощью следующих команд логических операций:

- ✓ логического умножения числа в аккумуляторе и маски, которое очищает разряд числа, если в соответствующем разряде маски будет записан "0", и не изменяет его, если в разряде маски записана "1";
- ✓ логического сложения числа в аккумуляторе и маски, которое устанавливает разряд числа в "1", если в таком же разряде маски будет записана " 1", и не изменяет его. если в этом разряде записан "0";
- ✓ логического исключающего ИЛИ" числа в аккумуляторе и маски, которое инвертирует содержание разряда числа, если в соответствующем разряде маски записана "1", и не изменяет его, если, в этом разряде записан "0".

Примеры использования этих команд приведены в табл. 8.1.

Таблица 8.1

Мнемокод	Машинный код	Число в аккумуляторе	Маска	Комментарий	Результат в аккумуляторе
		00111010	10101100		00101000
		11111111	00100010	Логическое	00100010
		00000000	00100010	умножение	00000000
ANI <D1>	E6 <D1>	10101010	00100010	содержимого	00100010
		11110000	11111111	аккумулятора с	11110000
		00001111	11111111	байтом D1	00001111
		00100010	00000000		00000000
				Логическое	
		00111010	10101100	сложение	10111110
ORI <D1>	F6 <D1>	00001111	00001111	содержимого	00001111
		11110000	00001111	аккумулятора с	11111111
				байтом D1	
				Логическое	
		00111010	10101100	"исключающее	10010110
XRI <D1>	EE <D1>	00001111	00001111	ИЛИ"	00000000
		11110000	00001111	содержимого	11111111
				аккумулятора с	
				байтом D1	

Проведение логических операций возможно также с содержимым аккумулятора и внутренними регистрами МП. В этом случае команды - однобайтные. При выполнении всех логических команд задействуются разряды Z, S, P, AC регистра признаков (в разряд CY записывается 0). Это позволяет проверять состояние любого разряда числа и выполнять условные переходы в программах. Программа маскирования отдельных разрядов числа (программа 7), записанного во входном устройстве, приведена ниже. Программа помещает результат маскирования в выходное устройство.

Программа 2

Адрес	Код	Метка	Мнемокод	Комментарий
0800	**DB 05**	**CNT:**	**IN 05h**	**; Получить число из входного устройства**
0802	**E6 20**		**ANI 20h**	**; Выполнить логическую операцию**
0804	**D3 05**		**OUT 05h**	**; Записать результат в выходное устройство**
0806	**C3 00 08**		**JMP CNT**	**; Продолжать**

словные переходы организуются в программах с помощью команд условных переходов. При выполнении этих команд МП проверяет состояние соответствующего разряда регистра признаков. Если при проверке состояния разряда регистра признаков условие не подтверждается. то выполняется следующая по порядку команда программы. Все команды условных переходов - трёхбайтные: первый байт содержит код команды, второй и третий байты - адрес передами управления. Таким образом, команды условных переходов позволяют строить ветвящиеся алгоритмы и в зависимости от текущего значения результата выполнения программы переходить на различные участки программы.

Ниже приведена программа (программа 8) для определения "1" в пятом разряде числа, записанного во входном устройстве. Программа использует маскирование числа и условный переход.

Программа 3

Адрес	Код	Метка	Мнемокод	Комментарий
0800	**DB 05**	**WAIT:**	**IN 05h**	**; Получить число из входного устройства**
0802	**E6 20**		**ANI 20h**	**; Проверить состояние пятого разряда числа**
0804	**CA 00 08**		**JZ WAIT**	**; Идти на WAIT, если в пятом разряде был 0 ; (Z=1)**
0807	**CF**		**RST 1**	**; Окончить выполнение программы**

В представленных ранее программах имел место лишь один цикл, в котором работала микроЭВМ. Программа ожидания появления "1" во втором и пятом разрядах числа, записанного во входном устройстве (программа 9), содержит два цикла.

Программа 4

Адрес	Код	Метка	Мнемокод	Комментарий
0800	DB 05	WAIT1:	IN 05h	; Получить число из входного ; устройства
0802	E6 04		ANI 00000100b	; Выключен ли второй переключатель?
0804	CA 00 08		JZ WAIT1	; Если нет, продолжить WAIT1
0807	3E FF		MVI A, FFh	; если да, зажечь светодиоды
0809	D3 05		OUT 05h	; выходного регистра
080B	DB 05	WAIT2:	IN 05h	; Получить число из входного ; устройства
080D	E6 20		ANI 00100000b	; выключен ли пятый переключатель?
080F	CA 0B 08		JZ WAIT2	; Если нет, продолжать WAIT2
0812	3E 00		MVI A, 00h	; Если да, погасить светодиоды
0814	D3 05		OUT 05h	; выходного регистра
0816	C3 00 08		JMP WAIT1	; Повторить программу

Задания

1. Ознакомьтесь с командами ввода-вывода МП i8080A.

2. Изучите группу логических команд и команд условной передачи управления.

3. Ознакомьтесь с разрядами регистра признаков МП и правилами записи в них "1".

4. Ознакомьтесь с программами 1,2,3,4.

5. Самостоятельно разработайте программы: а) включения светодиодов выходного устройства, если число, записанное во входном устройстве, больше 3; б) включения светодиодов выходного устройства, если число, записанное во входном устройстве, больше 3, но меньше 8.

6. Видоизмените программу 8 так, чтобы микроЭВМ реагировала на 0 в пятом разряде при записанных "1" во всех остальных разрядах.

7. Исследуйте программу 6.

7.1. Введите в микроЭВМ программу 1. Осуществить пуск программы.

7.2. Убедитесь что при выполнении программы микроЭВМ постоянно переписывает данные с входного устройства в выходное. Для этого с помощью переключателей входного устройства измените числа, записанные в нем. Информация о числах в устройствах ввода-вывода отображается светодиодами.

8. Исследуйте программу 2.

8.1. Введите в микроЭВМ программу 2. Осуществите пуск программы и исследуйте результат выполнения по числу, записанному в выходное устройство.

8.2. Заменяя в программе 2 двухбайтную команду ANI <D> на однобайтные ANA A, XRA A, ORA A, исследуйте результат их выполнения по числу, записанному в выходном устройстве.

9. Исследуйте программу 3.

9.1. Введите в микроЭВМ программу 3. Осуществить пуск программы и убедитесь, что при ее выполнении микроЭВМ реагирует лишь на те числа во входном устройстве, которые содержат "1" в пятом разряде. После окончания выполнения программы (выполнения в программе команды RST 1) в разряде 3 регистра состояния записана "1".

9.2. Исследуйте видоизмененную программу 3, позволяющую микроЭВМ реагировать на "0" в пятом разряде при записанных "1" во всех остальных разрядах.

10. Исследуйте программу 4.

10.1. Введите в микроЭВМ программу 4. Осуществить пуск программы и убедитесь, что при наличии "1" лишь во втором разряде числа входного регистра светодиоды выходного регистра включены и микроЭВМ работает в цикле WAIT2 ожидания появления единицы в пятом разряде числа.

10.2. Запишите "0" во второй разряд входного устройства. Запишите "0" в пятый разряд входного устройства и убедитесь, что светодиоды выходного устройства выключаются и микроЭВМ находится при выполнении цикла WAIT1 программы.

10.3. Установите "1" одновременно во втором и пятом разрядах числа во входном устройстве и проверьте, что микроЭВМ последовательно выполняет оба цикла (WAIT1, WAIT2) программы.

11. Исследуйте программы, самостоятельно разработанные в п. 5 задания.

Содержание отчета

Отчет должен содержать:

1. Цель работы.

2. Результаты исследования выполнения программ 1-4.

3. Самостоятельно разработанные и исследованные в процессе выполнения лабораторной работы программы, указанные в п. 5 задания.

4. Выводы.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

Основная литература

1. David Patterson John Hennessy. Computer Organization and Design. 5th Edition. 2013.
2. Шамаева О.Ю. Архитектура компютера. Конспект лексии. МЕИ. Москва, 2015.
3. С.А.Орлов, Б.Я.Силкер. Организатсия ЭВМ и систем: Учебник для вузов. 2-е изд. — СПб.: Питер, 2011. — 688 с
4. А.В.Павлов, Архитектура вычислителных систем - СПб: Университет ИТМО, 2016. – 86 с.
5. Z.Z.Miryusupov, J.X.Djumanov. «Kompyuter arxitekturasi». /TATU. 144 bet. Toshkent, 2017
6. Musaev M.M. "Kompyuter tizimlari va tarmoqlari". Toshkent.: "Aloqachi" nashriyoti, 2013 yil. 8 bob. 394 bet. – Oliy o'quv yurtlari uchun qo'llanma.
7. Баденко В.Л. Высокопроизводителные вычисления. Учебное пособие. СПб. Изд. Политехнического университета. 2010. -180 с.
8. Таненбаум Э., Остин Т. Архитектура компютера // 6-е издание. СПб.: Питер, 2013. — 811 с

Дополнительная литература

1. Mirziyoev Sh.M. Buyuk kelajagimizni mard va olijanob xalqimiz bilan birga quramiz. Toshkent. «O'zbekiston», NMIU, 2017. – 488 b.
2. Mirziyoev Sh.M. Qonun ustuvorligi va inson manfaatlarini ta'minlash – yurt taraqqiyoti va xalq farovonligining garovi. Toshkent. «O'zbekiston», NMIU, 2017. – 48 b.
3. Mirziyoev Sh.M. Erkin va farovon, demokratik O'zbekiston davlatini birgalikda barpo etamiz. Toshkent. «O'zbekiston», NMIU, 2016. – 56 b.
4. Бройдо В.Л. Вычислителные системы, сети и телекоммуникатсии – СПб.: Питер. 2003.
5. Довгий П. С., Поляков В. И. Прикладная архитектура базовой модели протсессора Интел. Учебное пособие по диссиплине «Организатсия ЭВМ и систем». – СПб.: НИУ. ИТМО, 2012. – 115 с.
6. Юров В.И. Ассемблер. Учебник для вузов. 2-е изд. -СПб.: Питер, 2010. -637с
7. Хорошевский В.Г. Архитектура вычислителных систем. Учебное пособие. М.: Изд. МГТУ им. Н.Э.Баумана. 2008. - 534
8. Столингс У. Структурная организатсия и архитектура компютерных систем. М.: Вилямс, 2002.- 896 с.
9. Соломенчук В.Г., Соломенчук П.В. Железо персоналных компютеров 2010. СПб.: БХВ Петербург, 2010. – 448 с.

Интернет сайты

1. www.intuit.ru
2. http://tuitfiles
3. http://www.kgtu.runnet.ru
4. http://www.piter.com

ОГЛАВЛЕНИЕ

Формат 60x84 1/16. Печ.лист 6.
Заказ № 16. Тираж 20.
Отпечатано в «Редакционно издательском»
отделе при ТУИТ.
Ташкент ул. Амир Темур, 108.