Содержание

1. Введение. Обзор существующих решений.

Микропроцессоры (МП) представляют собой автономные функционально законченные устройства, состоящие из одной или нескольких программно-управляемых интегральных микросхем высокой степени интеграции, включающие все средства, необходимые для обработки информации и управления данными и рассчитанные на совместную работу с устройствами памяти и ввода-вывода информации. Создание микропроцессоров привело к широкому внедрению универсальных вычислительных средств в разные отрасли техники.

По своей структурной и функциональной организации микропроцессоры аналогичны процессорам цифровых ЭВМ. Главными же отличительными признаками микропроцессоров является небольшая длина операндов, относительно небольшая емкость внутренней оперативной памяти и хранение программ и микропрограмм в постоянной памяти. Также преимуществами являются мультиплексный режим передачи информации по внутренним и внешним каналам, простая система команд.

Применяют микропроцессоры совместно с микропроцессорными наборами, которые представляют собой совокупности совместимых интегральных микросхем, разработанных для построения различных средств обработки информации. Обычно в микропроцессорные наборы входят: микропроцессор, ОЗУ, ПЗУ, а также интегральные микросхемы микропрограммного управления, ввода-вывода информации или интерфейса внешних устройств. Необходимо отметить, что сами по себе микропроцессоры еще неспособны решать какие-либо задачи, связанные с обработкой информации. Для этого из интегральных микросхем, входящих в микропроцессорный набор, необходимо организовать микроЭВМ, представляющие собой конструктивно завершенные вычислительные устройства, оформленные в виде устройства со своим источником тактового питания, интерфейсом ввода-вывода и комплексом программного обеспечения.

Структура микропроцессора должна удовлетворять трем основным требованиям: быть функционально гибкой, обеспечить достаточно высокое быстродействие и допускать недорогую технологическую реализацию. Поскольку семейство восьмиразрядных однокристальных микроконтроллеров Intel MCS 51 идеально подходит под эти требования, оно и будет использовано в данной работе. Кроме того, использование микроЭВМ этого семейства по сравнению с MCS 48 обеспечивает увеличение объема памяти команд и памяти данных. Новые возможности ввода-вывода и периферийных устройств расширяют диапазон применения и снижают общие затраты системы. В зависимости от условий использования, быстродействие системы увеличивается минимум в два с половиной раза и максимум в десять раз.

2. Архитектура МК-51.

Intel 8051 — это однокристальный микроконтроллер гарвардской архитектуры, который был впервые произведен в 1980 году для использования во встраиваемых системах. Был чрезвычайно популярен в течение 1980-ых, однако в настоящее время устарел и вытеснен более современными устройствами. Существует также советский клон данной микросхемы, КР1816ВЕ51 (далее МК-51) с характеристиками:

- ✓ состоит из процессорного ядра (CPU), ОЗУ, ПЗУ, портов, логики управления прерываниями, двух 16-битных таймеров и т. д;
- ✓ шина данных — 8-ми битная;
- ✓ шина адреса — 16 битная;
- ✓ встроенное ОЗУ — 128 байт памяти данных;
- ✓ встроенное ПЗУ — 4 Кб памяти программ;
- ✓ 4 порта ввода/вывода: двунаправленный и три квазидвунаправленных;
- ✓ два уровня приоритета прерываний;
- ✓ энергосберегающий режим.

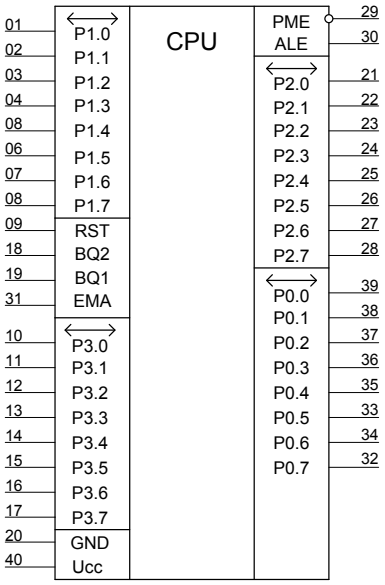

Рисунок 2.1. – Условное графическое обозначение МК-51

Таблица 2.1. – Назначение выводов МК-51

№ вывода	**Обозначение**	**Назначение**	**Тип**
1-8	P1.0-P1.7	Двунаправленный порт P1.	Вход/Выход
9	RST	Сигнал общего сброса.	Вход
10-17	P3.0-P3.7	Двунаправленный порт с дополнительными функциями.	Вход/Выход
	P3.6	Сигнал разрешения записи во внешнюю память данных – WR.	Выход
	P3.7	Сигнал разрешения чтения из внешней памяти данных – RD.	
18	BQ2	Выводы для подключения кварцевого резонатора.	Выход
19	BQ1		Вход
20	GND	Общий вывод.	
21-28	P2.0-P2.7	Двунаправленный порт P2.	
29	PME	Активное значение сигнала разрешает чтение из внешней памяти программ.	Выход
30	ALE	Выходной сигнал разрешения	

		фиксации адреса.	
31	EMA	Активное значение сигнала означает чтение кода из внутренней памяти.	Выход
40	Ucc	Напряжение питания (+5В).	Вход
32-39	P0.0-P0.7	Двунаправленный программируемый порт ввода/вывода.	

РОН и определяемые пользователем программно-управляемые флаги расположены в адресном пространстве внутреннего ОЗУ данных.

Таблица 2.2. – Назначение регистров

Обозначение	**Наименование**	**Адрес**
ACC*	Аккумулятор	0E0h
B*	Регистр B	0F0h
PSW*	Регистр состояния программы	0D0h
SP	Указатель стека	081h
DPTR	Указатель данных на 2 байта.	083h - 082h
P0*	Порт 0	080h
P1*	Порт 1	090h
P2*	Порт 2	0A0h
P3*	Порт 3	0B0h
IP*	Регистр приоритетов прерывания	0B8h
IE*	Регистр разрешения прерывания	0A8h

* – регистры, допускающие побитовую адресацию

Ниже описываются функции регистров, приведенных в таблице 2.2.

Аккумулятор. Команды, предназначенные для работы с аккумулятором, используют мнемонику “A”, например, MOV A, P2. Мнемоника “ACC” используется, к примеру, при побитовой адресации аккумулятора. Так, символическое имя пятого бита аккумулятора при использовании ассемблера будет следующим: ACC.5.

Регистр B. Используется во время операций умножения и деления. Для других инструкций рассматривается как дополнительный и сверхоперативный.

Регистр состояния программы. Информация приведена в таблице 2.3.

Таблица 2.3. – Назначение разрядов регистра PSW

Биты	**Наименование**		**Назначение битов**	**Доступ к биту**
7	CY		Флаг переноса.	Аппаратно или программно.
6	AC		Флаг дополнительного переноса.	Аппаратно или программно.
5	F0		Флаг, определяемый пользователем.	Программно.
4	RS1		Указатели банка рабочих регистров.	Программно.
3	RS0			Программно.
	RS1	RS0		
	0	0	Банк 0 с адресами (00h-07h).	
	0	1	Банк 1 с адресами (08h-0Fh).	
	1	0	Банк 2 с адресами (010h-017h).	
	1	1	Банк 3 с адресами (018h-01Fh).	
2	OV		Флаг переполнения.	Аппаратно или программно.
1	-		Резервный.	Программно.
0	P		Бит четности. Сбрасывается/ устанавливается в каждом цикле инструкций для указания четного/нечетного количества разрядов аккумулятора, находящихся в состоянии “1”.	Аппаратно или программно.

Указатель стека SP. 8-битовый регистр, содержимое которого инкрементируется перед записью данных в стек при выполнении команд PUSH

и CALL. При начальном сбросе указатель стека устанавливается в 07h, ведь область стека в ОЗУ данных начинается с адреса 08h.

Указатель данных. Указатель данных содержит 16-битовый адрес при обращении к внешней памяти. Может использоваться как два независимых восьмибитовых регистра.

Порт0-Порт3. Регистрами специальных функций P0, P1, P2, P3 являются регистры-“защелки” портов P0, P1, P2, P3.

Регистры управления. Регистры специальных функций IP, IE, TMOD, TCON, SCON и PCON содержат биты управления и биты состояния системы прерываний, таймеров/счетчиков и последовательного порта.

Также предусмотрена возможность задания частоты внутреннего генератора с помощью кварца, LC-цепочки или внешнего генератора.

Несмотря на то, что архитектура семейства МК-51 основана на архитектуре семейства МК-48, она все же не является полностью с ней совместимой. В новом семействе имеется ряд новых режимов адресации, дополнительные инструкции, расширенное адресное пространство и ряд других аппаратных отличий. Расширенная система команд обеспечивает побайтовую и побитовую адресацию, двоичную и двоично-десятичную арифметику, индикацию переполнения и определения четности/нечетности, возможность реализации логического процессора.

Важнейшей и отличительной чертой архитектуры семейства МК-51 является то, что АЛУ может наряду с выполнением операций над 8-разрядными типами данных манипулировать одноразрядными данными. Отдельные программно-доступные биты могут быть установлены, сброшены или заменены их дополнением, могут пересылаться, проверяться и использоваться в логических вычислениях. Поддержка простых типов данных (при существующей тенденции к увеличению длины слова) может с первого взгляда показаться шагом назад, но благодаря такому мощному АЛУ, набор инструкций микроЭВМ семейства МК-51 одинаково хорошо подходит как для

применений управления в реальном масштабе времени, так и для алгоритмов с большим объемом данных.

3. Организация памяти в МК-51

В архитектуре МК-51 память данных и память программ разделены. Каждая из них может иметь размер до 64 Кб, выбор одной из двух матриц памяти осуществляется сигналами PME, WR, RD. Организация памяти в микроконтроллерах семейства МК-51 иллюстрируется на рисунке 3.1.

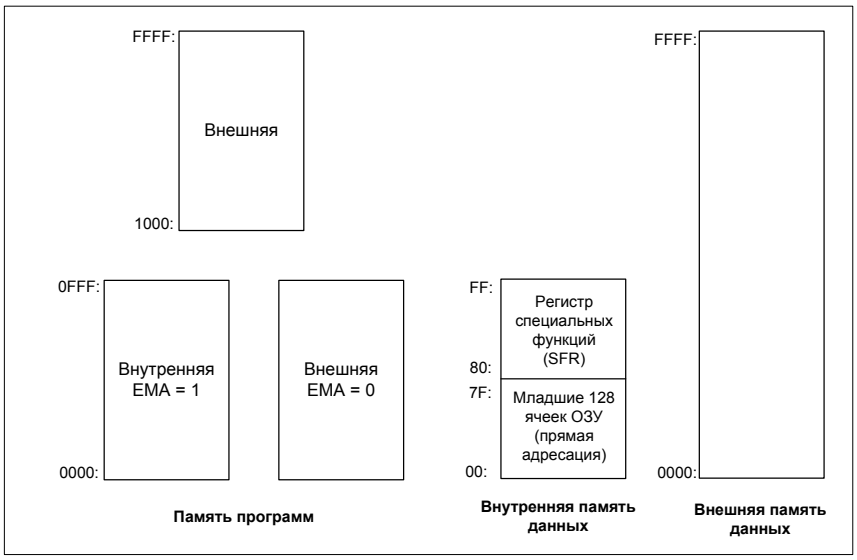

Рисунок 3.1. – Организация памяти в архитектуре МК-51

Память программ.

Память программ разделяется на резидентную (находящуюся внутри микросхемы) и внешнюю, для реализации которой требуются дополнительные микросхемы. РПП представляет собой ПЗУ, емкостью 4 Кб (адреса от 0 до 0FFFFh). РПП может быть отключена путем подачи низкого уровня на вход EMA. Адреса 0h, 03h, 0Bh, 013h, 01Bh и 023h имеют специальное назначение:

- ✓ 00h – начальный адрес пуска;
- ✓ 03h – вектор внешнего прерывания INT0;
- ✓ 0Bh – вектор прерывания от таймера T/C0;
- ✓ 013h – вектор внешнего прерывания INT1;
- ✓ 01Bh – вектор прерывания от таймера T/C1;
- ✓ 023h – вектор прерывания от последовательного интерфейса.

Подключение внешней памяти программ показано на рисунке 3.2.

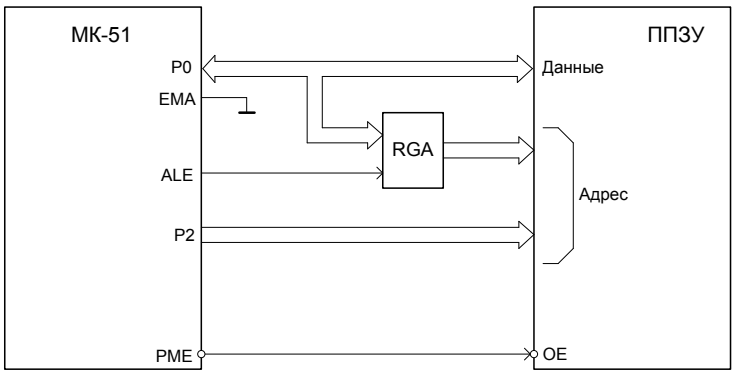

Рисунок 3.2. – Схема подключения к МК-51 внешнего ППЗУ программ

При обращениях к внешней памяти программ всегда формируется 16-разрядный адрес, младший байт которого выдается через порт P0, а старший – через порт P2. При этом байт адреса, выдаваемый через порт P0, должен быть зафиксирован во внешнем регистре по спаду сигнала ALE. Порт P0 работает как мультиплексированная шина адрес/данные: выдает младший байт счетчика команд, а затем переходит в высокоимпедансное состояние и ожидает прихода байта из ППЗУ программ. Когда младший байт адреса находится на выходах порта P0, сигнал ALE защелкивает его в адресном регистре RG. Старший байт адреса находится на выходах порта P2 в течение всего времени обращения к ППЗУ. Сигнал PME разрешает выборку байта из ППЗУ, после чего выбранный байт поступает на порт P0 МК-51 и вводится в микроЭВМ.

Адресация в памяти программ – непосредственная или косвенная базовая индексная. В первом случае из памяти программ выбирается константа, явно заданная в команде. Например, при выполнении инструкции MOV R2, #15 в регистр пересылается константа 15. Во втором случае в качестве индексного регистра используется аккумулятор, а в качестве базового – регистр-указатель данных DPTR или счетчик команд PC. Чтение операндов выполняется командами MOVC.

Доступ к внешней памяти программ осуществляется в любом случае, если программный счетчик содержит число большее, чем максимальная ячейка внутренней памяти программ.

Память данных.

Память данных предназначена для приема, хранения и выдачи информации, используемой в процессе выполнения программы. Память данных, расположенная на кристалле микроЭВМ, состоит из регистра адреса ОЗУ, дешифратора, ОЗУ и указателя стека.

Регистр адреса ОЗУ предназначен для приема и хранения адреса выбираемой с помощью дешифратора ячейки памяти, которая может содержать как бит, так и байт информации.

ОЗУ представляет собой 128 восьмиразрядных регистров, предназначенных для приема, хранения и выдачи различной информации.

Указатель стека представляет собой восьмиразрядный регистр, предназначенный для приема и хранения адреса ячейки стека, к которой было последнее обращение. При выполнении команд LCALL, ACALL содержимое указателя стека увеличивается на 2, а при выполнении RET, RETI – уменьшается на 2. При выполнении команды PUSH содержимое указателя стека увеличивается на 1, а при выполнении POP – уменьшается на 1. После сброса в указателе стека устанавливается адрес 07h, что соответствует началу стека с адресом 08h.

Внутренняя память данных состоит из двух областей: оперативной памяти (ОЗУ) с адресами 0h-07Fh и области регистров специальных функций, занимающей адреса 080h-0FFh. Распределение пространства внутренней памяти данных показано на рисунке 3.3.

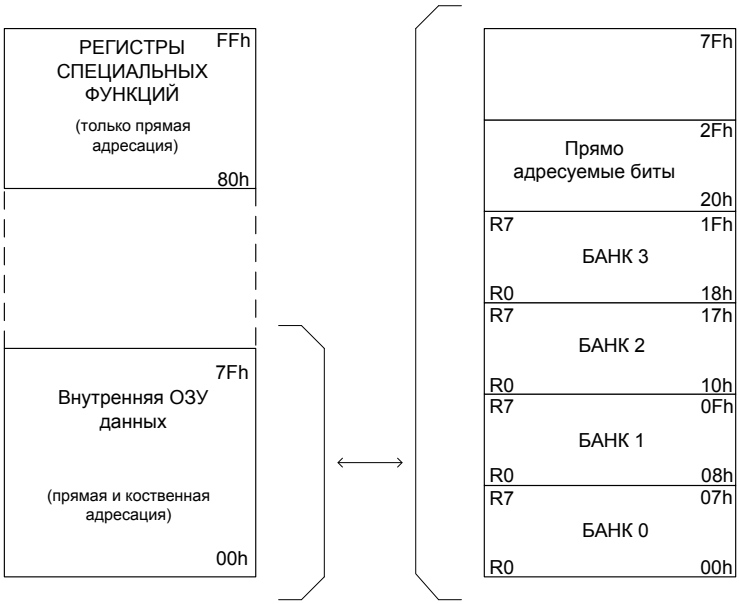

Рисунок 3.3. – Адресное пространство внутренней памяти данных

Физически внутреннее ОЗУ данных и область регистров специальных функций являются отдельными устройствами. Все ячейки внутреннего ОЗУ данных могут адресоваться с использованием прямой и косвенной адресации. Кроме того, внутреннее ОЗУ данных имеет следующие особенности. Младшие 32 байта внутреннего ОЗУ данных сгруппированы в 4 банка по 8 регистров в каждом (БАНК0-БАНКЗ на рисунке 3.3). Команды программы могут обращаться к регистрам, используя их имена R0-R7. Два бита PSW (указатели банка рабочих регистров RS0 и RS1) определяют, с регистрами какого банка производятся манипуляции. Наличие такого механизма работы с ячейками ОЗУ позволяет экономить память программ, т. к. команды, работающие с регистрами R0-R7, короче команд, использующих прямую адресацию. Следующие после банков регистров 16 байт (адрес 020h-02Fh) образуют область ячеек, к которым возможна побитовая адресация. Набор команд микроЭВМ семейства МК-51 содержит значительное количество инструкций, позволяющих работать с отдельными битами, используя при этом прямую адресацию. 128 бит, составляющих рассматриваемую область внутреннего ОЗУ данных, имеют адреса 0h-07Fh и предназначены для работы с такими инструкциями. Обращение к внутреннему ОЗУ данных всегда осуществляется с использованием 8-разрядного адреса.

При включении питания содержимое ОЗУ будет иметь случайное значение.

Пространства внутренней и внешней памяти данных не пересекаются, т. к. доступ к ним осуществляется с помощью разных команд. Для работы с внешней памятью данных существуют специальные команды MOVX, которые не влияют на внутреннюю память данных. Обращение к ячейкам внешней памяти данных осуществляется только с использованием косвенной адресации по регистрам R0 и R1 активного банка регистров внутреннего ОЗУ (команды типа MOV @Ri) или по регистру специальных функций DPTR (команды типа MOV @DPTR). В первом случае будет формироваться 8-разрядный, а во втором – 16-разрядный адрес.

При обращениях к внешней памяти данных, адрес выводится через порт P0 (младший байт) и порт P2 (старший байт). Точно так, как и в случае внешней памяти программ, объем внешней памяти данных может быть увеличен за счет использования портов до 1 Гб.

Обмен байтом данных (запись и чтение) производится через порт P0, т. е. порт используется как шина адреса/данных в режиме мультиплексирования.

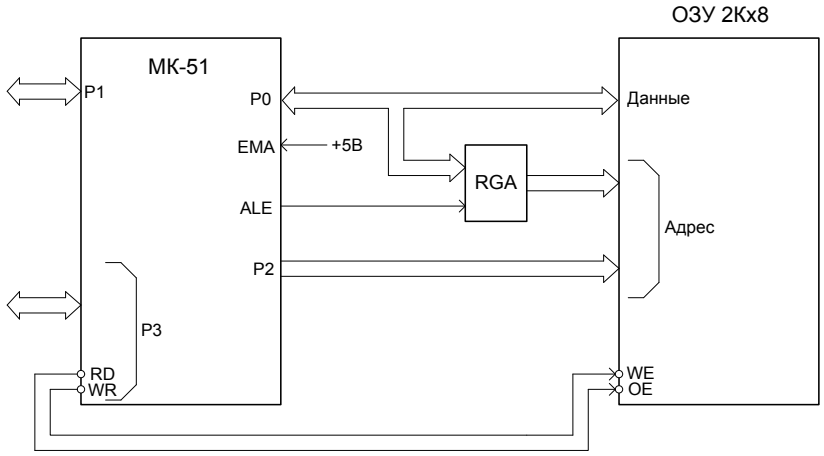

Рисунок 3.4. – Страничная организация внешней памяти данных

На рисунке 3.4 показана страничная организация внешней памяти данных. Приведенная схема позволяет работать с памятью данных емкостью 2 Кб, используя команды типа MOVX @Ri. Порт P0 при этом работает как мультиплексированная шина адрес/данные, а три линии порта P2 адресуют

страницы внешнего ОЗУ. Остальные 5 линий порта P2 могут использоваться в качестве линий ввода/ вывода.

4. Система команд

Система команд МК-51 предоставляет большие возможности обработки данных, обеспечивает реализацию логических, арифметических операций, а также управление в режиме реального времени. Реализована побитовая, потетрадная (4 бита), побайтовая (8 бит) и 16-разрядная обработка данных.

БИС семейства МК-51 – 8-разрядная микроЭВМ: ПЗУ, ОЗУ, регистры специального назначения, АЛУ и внешние шины имеют байтовую организацию. Двухбайтовые данные используются только регистром-указателем (DPTR) и счетчиком команд (PC). Следует отметить, что регистр-указатель данных может быть использован как двухбайтовый регистр DPTR или как два однобайтовых регистра специального назначения DPH и DPL. Счетчик команд всегда используется как двухбайтовый регистр.

Набор команд имеет 42 мнемонических обозначения (аббревиатур) команд для конкретизации 33 функций этой системы.

Синтаксис большинства команд ассемблерного языка состоит из мнемонического обозначения функции, вслед за которым идут операнды, указывающие методы адресации и типы данных. Различные типы данных или режимы адресации определяются установленными операндами, а не изменениями мнемонических обозначений. Например, аббревиатура “MOV” используется восемнадцатью различными командами для обработки трех типов данных (битов, байтов, адресов) в различных адресных пространствах.

Мнемонические обозначения функций однозначно связаны с конкретными комбинациями способов адресации и типами данных. Всего в системе команд возможно 111 таких сочетаний.

В машинном коде команда занимает один, два или три байта.

При частоте тактового генератора, равной 12 МГц, 64 одноцикловые команды выполняются за 1 мкс (12 тактов), 45 двухцикловых – за 2 мкс (24 такта) и 2 (MUL, DIV) четырехцикловых выполняются за 4 мкс (48 тактов).

В системе команд семейства МК-51 отсутствуют специальные команды ввода-вывода, управление таймерами/счетчиками и др. (как было в МК-48).

Систему команд микроЭВМ условно можно разбить на пять групп:

- ✓ арифметические команды;
- ✓ логические команды с байтовыми переменными;
- ✓ команды передачи данных;
- ✓ команды битового процессора;
- ✓ команды ветвления программ и передачи управления.

В наборе команд микроконтроллера имеются следующие арифметические операции:

- ✓ сложение ADD;
- ✓ сложение с учетом флага переноса ADDC;
- ✓ вычитание с займом SUBB;
- ✓ инкрементирование (увеличение на 1) INC;
- ✓ декрементирование (уменьшение на 1) DEC;
- ✓ десятичная коррекция DA;
- ✓ умножение MUL;
- ✓ деление DIV.

Действия производятся над целыми числами без знака. При операции умножения содержимое аккумулятора А умножается на содержимое регистра В, и результат размещается следующим образом: младший байт в регистре В, старший – в регистре А. В случае выполнения операции деления целое от деления помещается в аккумулятор А, остаток – в регистр В.

Система команд рассматриваемого микроконтроллера позволяет реализовать логические операции:

- ✓ И (ANL);

- ✓ ИЛИ (ORL);
- ✓ ИСКЛЮЧАЮЩЕЕ ИЛИ (XRL).

Логические операции выполняются над аккумулятором или непосредственно над портами ввода/вывода.

Существуют логические операции, которые выполняются только на аккумуляторе:

- ✓ сброс всех восьми разрядов A (CLR A);
- ✓ инвертирование всех восьми разрядов A (CPL A);
- ✓ циклический сдвиг влево и вправо без учета флага переноса (RR A; RL A);
- ✓ циклический сдвиг влево и вправо с учетом флага переноса (RRC A; RLC A);
- ✓ обмен местами старшей и младшей тетрад внутри аккумулятора (SWAP A).

Команды ветвления позволяют реализовывать условные операторы и операторы циклов. Доступны следующие команды:

- ✓ безусловный переход: LJMP, AJMP, SJMP;
- ✓ вызов и возврат из подпрограммы: LCALL, ACALL, RET, RETI;
- ✓ проверка содержимого аккумулятора: JZ, JNZ, CJNE, JMP;
- ✓ проверка флага переноса C: JC, JNC;
- ✓ проверка содержимого любого бита в битовом пространстве: JB, JNB, JBC.

Все команды условных переходов осуществляются относительно содержимого счетчика команд с адресом перехода, вычисляемым CPU во время выполнения команды.

Трехбайтовые команды перехода и вызова LCALL, LJMP (с 16-разрядным адресом) позволяют осуществлять переход и обращение по любому адресу адресного пространства памяти программ емкостью 64 Кбайт. Если необходим переход в пределах области памяти программ 2 Кб, то можно использовать

команды перехода и вызова с 11-разрядным адресом (ACALL, AJMP). Переход внутри участка памяти, определяемый 8-разрядной величиной смещения, осуществляется по команде SJMP.

Команды проверки содержимого аккумулятора и флага переноса C могут быть использованы для реализации проверки различных условий. При этом содержимое не изменяется.

Косвенный переход JMP @A+DPTR обеспечивает ветвление программы по содержимому аккумулятора, что позволяет реализовывать операцию перехода по заданному коду.

Пример микропрограммы, подающей на порт P1 сигналы разной длины:

```
; x2 - P1(7), x1 - P1(6), y5 - P1(5),
; y4 - P1(4), y3 - P1(3), y2 - P1(2),
; y1 - P1(1)
MOV A, #11000000b
JB7 x2
; Block 1
; y1 = 18 (17)
MOV R7, #02h
ORL P1, #00000110b
label1:
      DJNZ R7, label1
NOP
ANL P1, #11000100b
; y2 = 280 (279.5)
MOV R7, #032h
label2:
      DJNZ R7, label2
NOP
ANL P1, #11000000b
JMP block_2
; End block 1
x2:
      JB6 x1
      JMP block_2
      x1:
            ; Block 4
            ; y3 = 80 (79.5)
            ORL P1, #00001000b
            MOV R7, #0Eh
            label3:
                  DJNZ R7, label3
            ANL P1, #11000000b
            JMP program_end
            ; End block 4
block_2:
      ; Block 2
      ; y1 = 18 (17)
      MOV R7, #02h
      ORL P1, #00001010b
      label4:
            DJNZ R7, label4
      NOP
      ANL P1, #11001000b
      ; y3 = 80 (82)
      MOV R7, #0Bh
      label5:
            DJNZ R7, label5
      ANL P1, #11000000b
      ; End block 2
      ; Block 3
      ; y5 = 60 (59.5)
      ORL P1, #00110000b
      MOV R7, #0Ah
      label6:
            DJNZ R7, label6
      ANL P1, #11010000b
      ; y4 = 720 (719.5)
      MOV R7, #82h
      label7:
            DJNZ R7, label7
      ANL P1, #11000000b
      ; End block 3
program_end:
                  END
```

5. КПП и КПДП. Реализация прерываний.

5.1 Программируемые периферийные адаптеры.

Рассмотрим программируемые периферийные адаптеры (ППА). Основное их назначение – разработка программируемых устройств ввода/вывода для МПС. Структурная схема ППА К580ВВ55 приведена на рисунке 5.1.

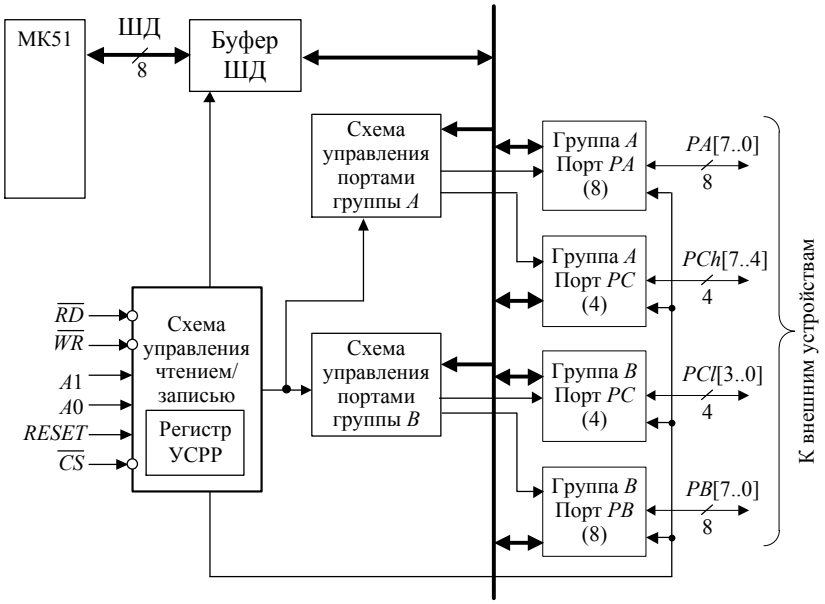

Рисунок 5.1. – Структурная схема ППА К580ВВ55

Адаптер 580ВВ55 обеспечивает ввод/вывод по трем дополнительным восьмиразрядным портам РА, РВ, РС. Причем порт РС может быть использован в качестве двух четырехразрядных портом PCh – старшая тетрада порта РС, PCl – младшая тетрада порта РС.

В состав ППА входят следующие функциональные блоки:

- ✓ буфер шины данных D7-D0;
- ✓ схема управления чтением/записью данных в регистр ППА;
- ✓ группа А порта РА – порт ввода/вывода РА группы А;
- ✓ группа В порт РВ – порт ввода/вывода РВ группы В;
- ✓ группа С порт РВ – порт ввода/вывода РВ группы В;
- ✓ группа С порт РС – порт ввода/вывода PCh группы А;
- ✓ группа В порт РС – порт ввода/вывода PCl группы В.

Схемы управления портами группы А и В содержат регистр управления, который задает режимы работы портов.

Все порты оснащены буферными регистрами, через которые производится связь между ППА и внешними шинами.

Для увеличения количества линий связи МК-51 с объектом управления можно подключить дополнительные 4-разрядные порты Р4-Р7. Наиболее просто это достигается при использовании специальной ИС КР580ВР43. В этом случае обеспечивается выполнение всех четырех команд с дополнительными портами, причем каждый вывод порта может быть настроен на ввод или вывод информации. Команды выполняются за 2 цикла. В первом цикле на выводы Р4-Р7 выдается управляющее слово, а во втором – через указанные выводы осуществляется обмен информацией.

5.2 Прерывания.

Под прерыванием понимают временную приостановку выполнения программы и переход на подпрограмму с возможностью возврата на прерванную. Прерывания можно классифицировать следующим образом: внутренние и внешние. Внутренние делятся на аппаратные и программные.

Микроконтроллеры семейства МК-51 обеспечивают поддержку пяти источников прерываний: двух внешних прерываний, поступающих по входам INT0 и INT1, двух прерываний от таймеров/счетчиков, прерывание от последовательного порта.

Запросы на прерывание фиксируются в регистрах специальных функций микроконтроллера: флаги IE0, IE1, TF0, TF1 запросов на прерывание от INT0, INT1, T/C0 и T/C1 содержатся в регистре управления TCON, а флаги RI и TI запросов на прерывание от последовательного порта – в регистре SCON управления последовательным портом.

Флаги TF0 и TF1 устанавливаются аппаратно при переполнении соответствующего таймера/счетчика и сбрасываются аппаратно при передаче управления программе обработки соответствующего прерывания.

Флаги TI и RI устанавливаются аппаратно схемой последовательного интерфейса соответственно после окончания передачи или приема байта и сбрасываются только программным путем.

Все указанные флаги запросов на прерывания программно доступны для установки и сброса. Программная установка флага запроса на прерывание приводит к такой же реакции микроконтроллера, что и аппаратная установка того же самого флага.

Сброс флагов IE0 и IE1 выполняется аппаратно при обслуживании прерывания только в том случае, если прерывание было настроено на восприятие спада сигнала INTx. Если прерывание было настроено на восприятие уровня сигнала запроса, то сброс флага IEx должна выполнять программа обслуживания прерывания, воздействуя на источник прерывания для снятия им запроса.

Каждый вид прерывания индивидуально разрешается или запрещается установкой или сбросом соответствующих бит регистра разрешения прерывания IE. Этот регистр содержит также и бит общего запрещения всех прерываний.

При одновременном поступлении запросов прерывания от источников, имеющих различные приоритеты, сначала обрабатывается запрос от более приоритетного источника.

В случае одновременного поступления нескольких запросов на прерывания с одинаковым приоритетом порядок их обработки определяется аппаратными средствами микроконтроллера и не может быть изменен программно.

При переходе на подпрограмму обработки прерывания, запрещаются все другие, имеющие уровень приоритета, равный уровню обслуживаемого прерывания.

Возврат из обработчика прерываний осуществляется с помощью команды RETI, которая восстанавливает из стека значение PC и логику приоритетов прерываний.

5.3 Режим прямого доступа в память.

Одним из способов обмена данными с ВУ является обмен в режиме прямого доступа к памяти (ПДП). В этом режиме обмен данными между ВУ и основной памятью микроЭВМ происходит без участия процессора. Обменом управляет электронные схемы, внешние по отношению к процессору. Обычно схемы, управляющие обменом, размещаются в специальном контроллере, который называется контроллером прямого доступа к памяти.

Обмен данными в режиме ПДП позволяет использовать в микроЭВМ быстродействующие внешние запоминающие устройства, такие, например, как накопители на жестких магнитных дисках, поскольку ПДП может обеспечить время обмена одним байтом данных между памятью и ВЗУ, равное циклу обращения к памяти. Для реализации этого режима необходимо обеспечить непосредственную связь контроллера ПДП и памяти микроЭВМ. В целях сокращения количества линий в шинах микроЭВМ контроллер ПДП подключается к памяти посредством шин адреса и данных системного интерфейса. При этом возникает проблема совместного использования шин системного интерфейса микропроцессором и контроллером ПДП. Можно выделить два основных способа ее решения: реализация обмена с “захватом цикла” и с блокировкой микроконтроллера.

Существуют две разновидности прямого доступа к памяти с “захватом цикла”. Наиболее простой состоит в том, что для обмена используются те машинные циклы процессора, в которых он не обменивается данными с памятью. В такие циклы контроллер ПДП может обмениваться данными с памятью, не мешая работе процессора. Однако возникает необходимость выделения таких циклов, например в некоторых процессорах формируется специальный управляющий сигнал. Более распространенным является ПДП с

“захватом цикла” и принудительным отключением процессора от шин системного интерфейса. Для реализации такого режима, системный интерфейс микроЭВМ дополняется двумя линиями для передачи управляющих сигналов “Требование прямого доступа к памяти” (ТПДП) и “Предоставление прямого доступа к памяти” (ППДП).

Управляющий сигнал ТПДП формируется контроллером прямого доступа к памяти. Микропроцессор, получив этот сигнал, приостанавливает выполнение очередной команды, не дожидаясь ее завершения, выдает на системный интерфейс управляющий сигнал ППДП и отключается от шин системного интерфейса. С этого момента все шины системного интерфейса управляются контроллером ПДП. Используя шины системного интерфейса, он осуществляет обмен одним байтом или словом данных с памятью микроЭВМ и затем, сняв сигнал ТПДП, возвращает управление системным интерфейсом микроконтроллеру. Как только контроллер ПДП будет готов к обмену следующим байтом, он вновь “захватывает” цикл микропроцессора и т.д. В промежутках между сигналами ТПДП, микропроцессор продолжает выполнять команды программы. Тем самым выполнение программы замедляется, но в меньшей степени, чем при обмене в режиме прерываний.

Применение в микроЭВМ обмена данными с ВУ в режиме ПДП всегда требует предварительной подготовки, а именно: для каждого ВУ необходимо выделить область памяти, используемую при обмене, указать ее размер, т. е. количество записываемых в память или читаемых из памяти байт (слов) информации. Следовательно, контроллер ПДП должен обязательно иметь в своем составе регистр адреса и счетчик байт (слов). Перед началом обмена с ВУ в режиме ПДП микропроцессор должен выполнить программу загрузки. Эта программа обеспечивает запись в указанные регистры контроллера ПДП начального адреса выделенной ВУ памяти и ее размера в байтах или словах в зависимости от того, какими порциями информации ведется обмен. Сказанное не относится к начальной загрузке программ в память в режиме ПДП. В этом

случае содержимое регистра адреса и счетчика байт слов устанавливается переключателями или перемычками непосредственно на плате контроллера.

Использование БИС ПДП позволяет существенно сократить аппаратные затраты при реализации прямого доступа к памяти.

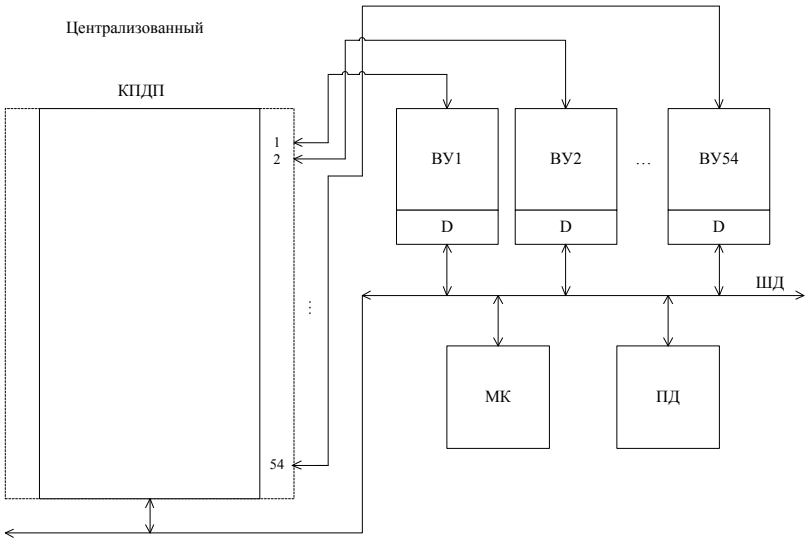

Рисунок 5.2. – Реализация централизованного КПДП

6. Выполнение индивидуального задания.

Задание: Разработать программу, которая вычисляет функцию X = X3*X4 + ($X9^2$ + $X10^2$)

6.1 Текст программы.

```
X3high          EQU 0h
X3mean          EQU 0FFh
X3low           EQU 0FFh
X4high          EQU 0h
X4mean          EQU 0FFh
X4low           EQU 0FFh
X9high          EQU 0h
X9mean          EQU 0h
X9low           EQU 0FFh
X10high         EQU 0h
X10mean         EQU 0h
X10low          EQU 0FFh
Adr_X3high      EQU 020h
Adr_X3mean      EQU 021h
Adr_X3low       EQU 022h
Adr_X4high      EQU 023h
Adr_X4mean      EQU 024h
Adr_X4low       EQU 025h
Adr_Zhigh_high  EQU 028h
Adr_Zmean_high  EQU 029h
Adr_Zlow_high   EQU 02Ah
Adr_Zhigh_low   EQU 02Bh
Adr_Zmean_low   EQU 02Ch
Adr_Zlow_low    EQU 02Dh
Adr_X9high      EQU 030h
Adr_X9mean      EQU 031h
Adr_X9low       EQU 032h
Adr_X91high     EQU 033h
Adr_X91mean     EQU 034h
Adr_X91low      EQU 035h
Adr_Yhigh_high  EQU 038h
Adr_Ymean_high  EQU 039h
```

```
      Adr_Ylow_high     EQU 03Ah
      Adr_Yhigh_low     EQU 03Bh
      Adr_Ymean_low     EQU 03Ch
      Adr_Ylow_low      EQU 03Dh
      Adr_X10high       EQU 040h
      Adr_X10mean       EQU 041h
      Adr_X10low        EQU 042h
      Adr_X101high      EQU 043h
      Adr_X101mean      EQU 044h
      Adr_X101low       EQU 045h
      Adr_Whigh_high    EQU 048h
      Adr_Wmean_high    EQU 049h
      Adr_Wlow_high     EQU 04Ah
      Adr_Whigh_low     EQU 04Bh
      Adr_Wmean_low     EQU 04Ch
      Adr_Wlow_low      EQU 04Dh
      Size              EQU 24
; load operands
      MOV Adr_X3high, #X3high
      MOV Adr_X3mean, #X3mean
      MOV Adr_X3low, #X3low
      MOV Adr_X4high, #X4high
      MOV Adr_X4mean, #X4mean
      MOV Adr_X4low, #X4low
; analyse sign
      MOV A, Adr_X3high
      XRL A, Adr_X4high
      JNB ACC.7, nosign1
; save sign
      SETB PSW.1
nosign1:
      CLR 07h
      CLR 01Fh
      MOV R0, #Size
multiplying1:
; shift X3 to the right
      CLR C
      MOV A, Adr_X3high
      RRC A
      MOV Adr_X3high, A
      MOV A, Adr_X3mean
      RRC A
      MOV Adr_X3mean, A
      MOV A, Adr_X3low
      RRC A
      MOV Adr_X3low, A
      JNC C01
; add X4 and Z
      CLR C
      MOV A, Adr_X4low
      ADDC A, Adr_Zlow_high
      MOV Adr_Zlow_high, A
      MOV A, Adr_X4mean
      ADDC A, Adr_Zmean_high
      MOV Adr_Zmean_high, A
      MOV A, Adr_X4high
      ADDC A, Adr_Zhigh_high
      MOV Adr_Zhigh_high, A
C01:
; shift Z to the right
      CLR C
      MOV A, Adr_Zhigh_high
      RRC A
      MOV Adr_Zhigh_high, A
      MOV A, Adr_Zmean_high
      RRC A
      MOV Adr_Zmean_high, A
      MOV A, Adr_Zlow_high
      RRC A
      MOV Adr_Zlow_high, A
      MOV A, Adr_Zhigh_low
      RRC A
      MOV Adr_Zhigh_low, A
      MOV A, Adr_Zmean_low
      RRC A
      MOV Adr_Zmean_low, A
      MOV A, Adr_Zlow_low
      RRC A
      MOV Adr_Zlow_low, A
      DJNZ R0, multiplying1
      JNB PSW.1, nosign_z
      SETB 047h
nosign_z:
; load operands
```

```
      MOV Adr_X9high, #X9high
      MOV Adr_X9mean, #X9mean
      MOV Adr_X9low, #X9low
      MOV Adr_X91high, #X9high
      MOV Adr_X91mean, #X9mean
      MOV Adr_X91low, #X9low
; analyse sign
      MOV A, Adr_X9high
      XRL A, Adr_X91high
      JNB ACC.7, nosign2
; save sign
      SETB PSW.1
nosign2:
      CLR 07h
      CLR 01Fh
      MOV R0, #Size
multiplying2:
; shift X9 to the right
      CLR C
      MOV A, Adr_X9high
      RRC A
      MOV Adr_X9high, A
      MOV A, Adr_X9mean
      RRC A
      MOV Adr_X9mean, A
      MOV A, Adr_X9low
      RRC A
      MOV Adr_X9low, A
      JNC C02
; add X91 and Y
      CLR C
      MOV A, Adr_X91low
      ADDC A, Adr_Ylow_high
      MOV Adr_Ylow_high, A
      MOV A, Adr_X91mean
      ADDC A, Adr_Ymean_high
      MOV Adr_Ymean_high, A
      MOV A, Adr_X91high
      ADDC A, Adr_Yhigh_high
      MOV Adr_Yhigh_high, A
C02:
; shift Y to the right
      CLR C
      MOV A, Adr_Yhigh_high
      RRC A
      MOV Adr_Yhigh_high, A
      MOV A, Adr_Ymean_high
      RRC A
      MOV Adr_Ymean_high, A
      MOV A, Adr_Ylow_high
      RRC A
      MOV Adr_Ylow_high, A
      MOV A, Adr_Yhigh_low
      RRC A
      MOV Adr_Yhigh_low, A
      MOV A, Adr_Ymean_low
      RRC A
      MOV Adr_Ymean_low, A
      MOV A, Adr_Ylow_low
      RRC A
      MOV Adr_Ylow_low, A
      DJNZ R0, multiplying2
      JNB PSW.1, nosign_
      SETB 047h
nosign_y:
; load operands
      MOV Adr_X10high, #X10high
      MOV Adr_X10mean, #X10mean
      MOV Adr_X10low, #X10low
      MOV Adr_X101high, #X10high
      MOV Adr_X101mean, #X10mean
      MOV Adr_X101low, #X10low
; analyse sign
      MOV A, Adr_X10high
      XRL A, Adr_X101high
      JNB ACC.7, nosign3
; save sign
      SETB PSW.1
nosign3:
      CLR 07h
      CLR 01Fh
      MOV R0, #Size
multiplying3:
; shift X10 to the right
```

```
      CLR C
      MOV A, Adr_X10high
      RRC A
      MOV Adr_X10high, A
      MOV A, Adr_X10mean
      RRC A
      MOV Adr_X10mean, A
      MOV A, Adr_X10low
      RRC A
      MOV Adr_X10low, A
      JNC C03
; add X101 and W
      CLR C
      MOV A, Adr_X101low
      ADDC A, Adr_Wlow_high
      MOV Adr_Wlow_high, A
      MOV A, Adr_X101mean
      ADDC A, Adr_Wmean_high
      MOV Adr_Wmean_high, A
      MOV A, Adr_X101high
      ADDC A, Adr_Whigh_high
      MOV Adr_Whigh_high, A
C03:
; shift W to the right
      CLR C
      MOV A, Adr_Whigh_high
      RRC A
      MOV Adr_Whigh_high, A
      MOV A, Adr_Wmean_high
      RRC A
      MOV Adr_Wmean_high, A
      MOV A, Adr_Wlow_high
      RRC A
      MOV Adr_Wlow_high, A
      MOV A, Adr_Whigh_low
      RRC A
      MOV Adr_Whigh_low, A
      MOV A, Adr_Wmean_low
      RRC A
      MOV Adr_Wmean_low, A
      MOV A, Adr_Wlow_low
      RRC A
      MOV Adr_Wlow_low, A
      DJNZ R0, multiplying3
      JNB PSW.1, nosign_w
      SETB 047h
nosign_w:
; analyse sign
      MOV A, Adr_Zhigh_high
      XRL A, Adr_Yhigh_high
      JNB ACC.7, nosign4
; save sign
      SETB PSW.1
nosign4:
      CLR 07h
      CLR 01Fh
; add Z and Y
      CLR C
      MOV A, Adr_Zlow_low
      ADDC A, Adr_Ylow_low
      MOV Adr_Ylow_low, A
      MOV A, Adr_Zmean_low
      ADDC A, Adr_Ymean_low
      MOV Adr_Ymean_low, A
      MOV A, Adr_Zhigh_low
      ADDC A, Adr_Yhigh_low
      MOV Adr_Yhigh_low, A
      MOV A, Adr_Zlow_high
      ADDC A, Adr_Ylow_high
      MOV Adr_Ylow_high, A
      MOV A, Adr_Zmean_high
      ADDC A, Adr_Ymean_high
      MOV Adr_Ymean_high, A
      MOV A, Adr_Zhigh_high
      ADDC A, Adr_Yhigh_high
      MOV Adr_Yhigh_high, A
      JNB PSW.1, nosign_t
      SETB 047h
nosign_t:
; analyse sign
      MOV A, Adr_Whigh_high
      XRL A, Adr_Yhigh_high
      JNB ACC.7, nosign5
; save sign
```

```
        SETB PSW.1
nosign5:
        CLR 07h
        CLR 01Fh
; add W and Y
        CLR C
        MOV A, Adr_Wlow_low
        ADDC A, Adr_Ylow_low
        MOV Adr_Ylow_low, A
        MOV A, Adr_Wmean_low
        ADDC A, Adr_Ymean_low
        MOV Adr_Ymean_low, A
        MOV A, Adr_Whigh_low
        ADDC A, Adr_Yhigh_low
        MOV Adr_Yhigh_low, A
        MOV A, Adr_Wlow_high
        ADDC A, Adr_Ylow_high
        MOV Adr_Ylow_high, A
        MOV A, Adr_Wmean_high
        ADDC A, Adr_Ymean_high
        MOV Adr_Ymean_high, A
        MOV A, Adr_Whigh_high
        ADDC A, Adr_Yhigh_high
        MOV Adr_Yhigh_high, A
        JNB PSW.1, nosign_r
        SETB 047h
nosign_r:
        END
```

6.2 Блок-схема алгоритма.

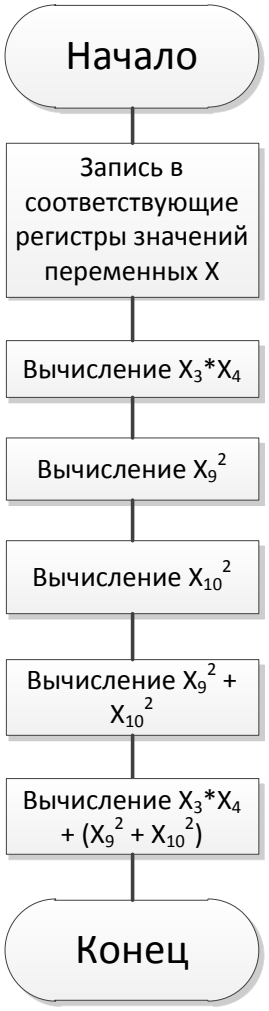

Рисунок 6.2.1 – Блок-схема алгоритма вычисления значения функции

6.3 Структурная схема системы.

Структурная схема МПС приведена на чертеже ИАЛЦ 460104 004. Е1. Ее ядром является микроконтроллер КР1816ВЕ51.

В состав МПС входят такие основные функциональные части:

1. Микроконтроллер МК-51.
2. Внешняя память данных – 5 страниц по 64 Кб, внешняя память программ – 10 страниц по 32 Кб.
3. Централизованный контроллер прямого доступа к памяти – КПДП.
4. Внешние устройства – 54 единиц.
5. Централизованный контроллер приоритетного прерывания – КПП.
6. Дополнительные порты.
7. Периферийный адаптер ВВ55 для подключения портов Р4, Р7.

Таблица распределения памяти адресов для внешних устройств в 10 странице внешней памяти данных приведена на рисунке 6.3.1.

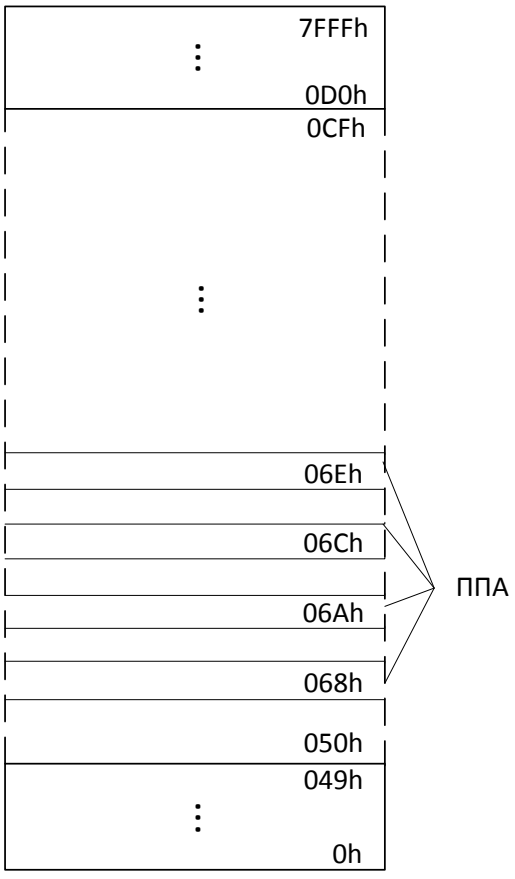

Рисунок 6.3.1 – Таблица распределения памяти внешних устройств

7. Выводы.

В работе была разработана микропроцессорная система на основе МК-51 с подключением внешней памяти данных, внешней памяти программ, периферийного адаптера, внешних устройств. МК-51 – это усовершенствованный МК-48, ряд нововведений в этом микроконтроллере позволяет облегчить написание программ под него, а также работу с портами. По сравнению с предшественником, он имеет встроенную память программ, расширенный набор команд. При разработке системы, имеющей определенное назначение, необходимо ответственно отнестись к выбору микроконтроллера со всеми необходимыми возможностями и параметрами.

8. Список литературы.

1. Бояринов А.Е., Дьяков И.А. – Архитектура микроконтроллеров MCS-51 – Тамбов: “Издательство ТГТУ”, 2005.
2. Конспект лекций по курсу “Архитектура ЭВМ”.
3. Жабин В.И., Ткаченко В.В. – Однокристальные и микропрограммируемые ЭВМ. – Киев, “Диалектика”, 1995.
4. Жабин В.И., Ткаченко В.В., Макаров В.В., Зайцев А.А. – Архитектура однокристальных ЭВМ. – Киев, “Век”, 1997.
5. http://library.tuit.uz/skanir_knigi/book/osnovi_mikroprosessor/osnovi_mikropros_1.htm
6. http://ru.wikipedia.org/wiki/Intel_8051