Национальный Технический Университет Украины

“Киевский Политехнический Институт”

Факультет Информатики и Вычислительной Техники

Кафедра вычислительной техники

Расчетно-графическая работа

по дисциплине “Архитектура компьютера”

Тема «Обработка информации в ЭВМ

на программном, микропрограммном и аппаратном уровнях».

**Выполнил:**

студент ІІІ-го курса

группы ИВ-73

Ашаев Ю.Н.

№ зач. кн. 7301

1 1100 1000 0101 2

Руководитель:

Ткаченко В.В.

Киев 2009

**Содержание**

[Введение 3](#_Toc249779970)

[1. Обзор МПС 4](#_Toc249779971)

[2. Архитектура МПС 5](#_Toc249779972)

[2.1 Разработка структурной схемы 5](#_Toc249779973)

[2.1.1 Память программ 5](#_Toc249779974)

[2.1.2 Память данных 6](#_Toc249779975)

[2.1.3 Внешние устройства 7](#_Toc249779976)

[2.1.4 Режим прерываний 8](#_Toc249779977)

[2.1.5 Прямой доступ к памяти (КПДП) 11](#_Toc249779978)

[3. Система команд 13](#_Toc249779979)

[3.1 Основные команды 13](#_Toc249779980)

[3.1.1 Команды арифметические и логические 13](#_Toc249779981)

[3.1.2 Команды передачи управления. 15](#_Toc249779982)

[3.1.2 Команды управления режимом работы МК 16](#_Toc249779983)

[4. Программная часть 18](#_Toc249779984)

[4.1 Карта распределения ОП 18](#_Toc249779985)

[4.2 Алгоритм программы 18](#_Toc249779986)

[4.3 Алгоритм умножения двух чисел 19](#_Toc249779987)

[4.4 Алгоритм деления двух чисел 19](#_Toc249779988)

[4.5 Код микропрограммы 20](#_Toc249779989)

# Введение

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.

Для сокращения количества типов интегральных микросхем, уменьшения их стоимости и сокращения времени разработки в настоящее время широко используются многофункциональные и универсальные интегральные микросхемы с программным управлением, ставшие основой построения микропроцессоров. Из-за своей универсальности микропроцессоры могут производиться большими сериями, что определяет их низкую стоимость.

Создание микропроцессоров привело к широкому внедрению универсальных вычислительных средств в те отрасли техники, где любые другие вычислительные средства, созданные ранее, не могли быть использованы. По своей структурной и функциональной организации микропроцессоры аналогичны процессорам цифровых ЭВМ. Главными же отличительными признаками микропроцессоров является выполнение всех их структурных единиц в виде высокоинтегрированных микросхем, небольшая длина операндов, относительно небольшая емкость внутренней оперативной памяти, хранение программ и микропрограмм в постоянной памяти, мультиплексный режим передачи информации по внутренним и внешним каналам, простая система команд. В отличие от интегральных микросхем с узкой функциональной ориентацией, которые выпускались небольшими партиями или вообще были заказными изделиями, микропроцессоры по существу стали первыми крупносерийными интегральными микросхемами общего назначения. Это объясняется следующими их важнейшими качествами:

При проектировании и производстве цифровых систем на основе микропроцессоров оказывается возможным значительное сокращение материальных, трудовых и временных затрат, так как микропроцессоры представляют собой стандартное средство обработки информации.

Применение микропроцессоров расширяет возможности вычислительной техники, при этом цифровые системы могут приобретать новые свойства без изменения состава технических средств.

Применяют микропроцессоры совместно с микропроцессорным наборами, которые представляют собой совокупности совместимых интегральных микросхем, разработанных для построения различных средств обработки информации. Обычно в микропроцессорные наборы входят: микропроцессор, ОЗУ, ПЗУ, ППЗУ, а также интегральные микросхемы микропрограммного 3управления, ввода-вывода информации или интерфейса внешних устройств. Необходимо отметить, что сами по себе микропроцессоры еще не способны 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 автономного 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. Высокая функциональная гибкость микропроцессора, необходимая для создания эффективного программного обеспечения, достигается микропрограммным управлением, за счет побайтовой обработки и адресации данных, развитой системы прерываний и большего числа внутренних регистров.

# 1. Обзор МПС

Микропроцессорная система (МПС) представляет собой функционально законченное изделие, состоящее из одного или нескольких устройств, главным образом микропроцессорных: микропроцессора и/или микроконтроллера.

В микроЭВМ шины адреса и данных объединены в ШАД. Разрядность ШАД - 32 бита, что обеспечивает адресацию 8МБ оперативной памяти и работу с 32-битными данными. Для работы с ШАД имеются интерфейсные регистры РА и РД. Регистр РА может выдавать на ШАД свою младшую, либо старшую часть.

Оперативная память с изменяемыми разделами (MVT). Каждой задаче выделяется объём памяти, необходимый для её решения (т.е. динамическое разделение). Выделяется квант памяти – это параграф (2i); чаще бывает, что квант времени – это 16 ячеек памяти. Компиляция программ производится в условных адресах. При загрузке программ в память к условному адресу прибавляется базовый адрес. Существуют такие дисциплины выделения раздела: 1. первый, который подходит по размеру; 2. наименьший, подходящий по размеру. При таких выделениях разделов возникают не очень хорошие последствия - фрагментация памяти. Дефрагментация производится с помощью специальных Utility.

Обмен данных между внешними устройствами и оперативной памятью осуществляется через КПДП. Обработка векторных прерываний от ВУ выполняется КПП. КПДП и КПП совмещены и распределены по интерфейсам внешних устройств.

Арифметическая и логическая обработка данных осуществляется в БОД. В нем имеется набор рабочих регистров, которые могут адресоваться либо регистрами RA, RB, либо адресными полями AdA, AdB регистра микрокоманд РМК.

ПМК - память микрокоманд - служит для хранения микропрограмм.

ЭВМ обладает блоком совмещения и выборки команд. БМУ управляет выборкой, распаковкой и выполнением микрокоманд. Через мультиплексор внешних условий он взаимодействует с ВУ, БУ SVS, БОД, РАПП и РАПДП. Буферы М, Р и V обеспечивают интерфейс между локальной шиной, шиной адреса ветвления и БМУ.

Структуры ОЭВМ серии 1816 и их команд таковы, что в случае необходимости функционально-логические воз­можности могут быть расширены. С использование внеш­них дополнительных БИС постоянной и оперативной па­мяти адресное пространство может быть расширено, а путем подключения различных интерфейсных БИС число линий связи ОЭВМ с объектом управления мо­жет быть увеличено практически без ограничений.

ОЭВМ серии 1816 требуют одного источника электро­питания напряжением +5В ± 10%, рассеивают мощность около 1,5 Вт и работают в диапазоне температур от 0 до 700С. по входам и выходам серии 1816 электрически совместимы с интегральными схемами ТТЛ.

ОЭВМ МК 48 может работать в диапазоне частот син­хронизации от 1 до 6 МГц, а минимальное время выпол­нения команды составляет 2,5 мкс.

Микроконтроллер состоит из следующих узлов:

— однокристальной ЭВМ со схемой внешнего тактового генератора и схемой формирования сигнала «сброс»;

— регистра-защелки младшего байта адреса внешнего запоминающего устройства;

— памяти программ, объемом 4 Кбайта;

— памяти данных, объемом 1 Кбайт со страничной адресацией 256 байт на страницу и схемой выбора ОЗУ;

— схемы управления записью-чтением внешних устройств;

— адаптера параллельного интерфейса со схемами приемника и передатчика по стандарту ИРПС;

— трехканального таймера;

— контроллера клавиатуры и индикации;

— схемы прерываний.



Рисунок 1.1 - Структурная схема МК48

# 2. Архитектура МПС

Разрабатывается микропроцессорная система (МПС), ядром которой является процесор 1816ВЕ48.

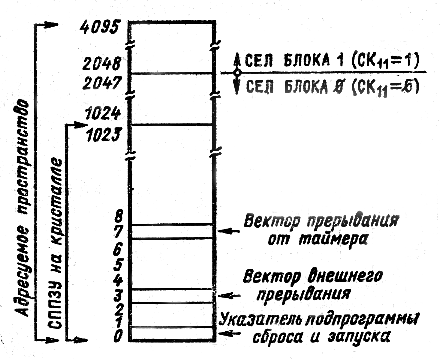
В состав разрабатываемой МПС должны входить процессор (П), основная память (ОП), содержащая ОЗУ и ПЗУ, а также внешние устройства (ВУ), контроллеры прерываний и прямого доступа к памяти.

Микропроцессор имеет 16-разрядную операционную часть. Объем внешней памяти программ 3К, внешней памяти данных – 8К. Шина адреса и данных объединённая, КПП и КПДП - централизованные. Каждое ВУ имеет фиксированный адрес в адресном пространстве периферийных устройств. Всего 116 внешних устройств.

## 2.1 Разработка структурной схемы

### 2.1.1 Память программ

Память программ (или - память команд, ПК) реализована в резидентном СППЗУ емкостью 1 Кбайт. Максимальное адресное пространство, отводимое для программ, составляет 4 Кбайт. Счетчик команд (СК, или программный счетчик - PC) содержит 12 разрядов, но инкрементируются в процессе счета только младшие 11 разрядов. Поэтому счетчик команд из предельного состояния 7FFH (если только по этому адресу не расположена команда передачи управления) перейдет в состояние 000H. Состояние старшего разряда счетчика команд может быть изменено специальными командами ("SEL МВ0" и "SEL MB1"). Подобный режим работы СК позволяет создать два блока памяти емкостью по 2 Кбайт каждый. Карта адресов памяти программ показана на рисунке



Кроме разделения на банки, память программ делится на страницы по 256 байт. Это связано с тем, что команды условных переходов модифицируют только 8 младших разрядов адреса, т.е. обеспечивают переход внутри страницы. При переходе к подпрограммам обслуживания прерываний автоматически обнуляется разряд РС[11] счётчика команд. В связи с этим подпрограммы обслуживания прерываний должны размещаться в нулевом банке памяти. Способы адресации операндов в памяти программ:

- непосредственная;

- косвенная с использованием аккумулятора.

В первом случае операнд содержится в байте, следующем за кодом команды. При косвенной адресации в качестве указателя адреса операнда в текущей странице (или в странице 3) применяется аккумулятор А.

На структурной схеме, которая дана в приложении 1, элементы представляют страницы памяти программ ПП по 1К. Выбор страницы осуществляется при помощи DC, на который поступает адрес с порта Р2. На дешифратор подаётся 2 бита. Всего адресуется 3 страницы памяти программ по 1К.

В резидентной памяти программ имеются три специализированных адреса:

адрес 0, к которому передается управление сразу после окончания сигнала системного сброса СБРОС; по этому адресу должна находиться команда безусловного перехода к началу программы (точка входа после начальной установки);

адрес 3, по которому расположен вектор прерывания от внешнего источника (точка входа при обработке внешних прерываний);

адрес 7, по которому расположен вектор прерывания от таймера или начальная команда подпрограммы обслуживания прерывания по признаку переполнения таймера/счетчика событий (точка входа при обработке прерываний по переполнению таймер-счетчика).

ПК служит для записи как команд, так и таблиц констант. Для отладки программ в микро-ЭВМ существует возможность отключения внутренней ПК подачей сигнала высокого уровня на вывод РВП (ЕА) микро-ЭВМ.

### 2.1.2 Память данных

На структурной схеме, которая дана в приложении 1, элементы представляют страницы памяти данных ПД 256\*8 бит. Выбор страницы осуществляется при помощи DC, на который поступает адрес с порта Р1. На дешифратор подаётся 6 бит (биты 0..5). Всего 32 страницы памяти данных, то есть в сумме 8К (ПД1-ПД32).

Память данных, как и память программ, разделяется на внутреннюю и внешнюю. Внутренняя память данных представляет собой ОЗУ ёмкостью 64 байта. Память содержит два банка регистров общего назначения (рабочих регистров). Банк 0 включает регистры R0-R7 с адресами 0-7, а банк 1 – регистры R0-R7, которые имеют адреса 24-31. Выбор регистрового банка осуществляется командами SEL RB0 и SEL RB1, которые устанавливают признак RB, находящийся в 4-м разряде PSW. Специальная команда для проверки RB отсутствует, но признак можно проанализировать, переслав содержимое PSW в А и выполнив переход по признаку B4, который проверяется командой JB4. Ячейки с адресами 8-23 могут использоваться как 8-уровневый стек 16-разрядных слов или как ячейки ОЗУ данных с произвольным доступом.

Способы адресации для доступа к внутренней памяти данных:

- прямая регистровая;

- косвенная регистровая.

Прямая регистровая адресация используется для обращения к регистрам общего назначения (R0-R7 при RB=0 или R0’-R7’ при RB=1). В этом случае 3-разрядный адрес регистра присутствует в коде команды.

С помощью косвенной регистровой адресации можно обратиться к любому байту внутренней памяти данных (в том числе, к регистровым банкам и стеку). В качестве указателя адреса операции в этом случае используются регистры R0 и R1, если выбран нулевой регистровый банк или R0 и R1, если выбран 1 регистровый банк. Косвенная регистровая адресация применяется и при выполнении команд обращения к внешней памяти данных.

Так как в качестве указателя адреса используютя 8-разрядные регистры R0, R1, R0’ и R1’, то максимальный объём внешней памяти данных может составлять 256 байт. Увеличение объёма внешней памяти данных может быть достигнуто программно с применением страничной адресации.



Рис. 2.2. Карта распределения адреса внутренней памяти данных

### **2.1.3 Внешние устройства**

Всего внешних устройств 116, их адреса с 0h до 0E6h (116 ВУ по 2 байта на регистр слова состояния и регистр данных, которые отличаются младшим битом).

Также подключён ППА с адресами 98h, 99h, 9Ah, 9Bh, которые относятся к 32 странице ПД. Составлен селектор адреса для подключения ППА или ПД32, если выбрана 32 страница. Если старшие 6 бит совпадают с адресами для ППА и выбрана 32 страница – подключается ППА, если адрес не совпадает и выбрана 32 страница, то подключается ПД32.

В данном режиме инициатором обмена является процессор. Для синхронизации используется бит готовности в порте ВУ. Этот бит устанавливается контроллером ВУ, когда оно готово к обмену, и сбрасывается при обращении к порту данных.

При наличии нескольких ВУ для их обслуживания используется разные методы, среди них полинг, т.е. опрос ВУ в соответствии с их приоритетом. Режим опроса готовности имеет преимущества и недостатки по сравнению с другими режимами.

Преимущества: простота реализации интерфейса ВУ, в процессе функционирования программы можно менять приоритеты ВУ. Недостатки: снижение производительности за счет непроизводительного расхода команд процессора на опрос ВУ, трудно предусмотреть аварийное или экстренное обслуживание некоторого ВУ.

В моей расчетно-графической работе полинг не используется, вместо этого используется централизированный контроллер приоритетного доступа, который устанавливает приоритеты обслуживания для ВУ и подает сигнал готовности к прерыванию на процессор.

### 2.1.4 Режим прерываний

Под прерыванием понимают временную приостановку выполнения программы и переход на другую подпрограмму с возможностью возврата на прерванную.

Прерывания можно классифицировать следующим образом: внутренние и внешние.

Внутренние делятся на аппаратные и программные.

Внешние делятся на безвекторные и векторные.

Аппаратные прерывания: требование прерывания формируется определенными схемами процессора при наступлении определенных событий (деление на ноль, зависание при обращении к памяти или ВУ и т.д.)

Программные прерывания: вызываются при выполнении команд прерывания программы. Эти команды могут вводиться программистом в исходную программу или вставляться компилятором в процессе компилирования.

Эти прерывания удобны в процессе отладки системы (они имитируют внешние прерывания), а также являются универсальным средством для вызова стандартных подпрограмм ОС.

Внутренние прерывания имеют фиксированные начальные адреса подпрограмм для их обслуживания.

Безвекторные прерывания: Процессор имеет специальные входы для поступления запросов на прерывания программы. Для некоторых входов существуют стандартные подпрограммы обслуживания с фиксированным начальным адресом (сбой по питагию постоянного тока, сбой по питанию переменного тока, сигнал от внешнего таймера).

Векторные: любому ВУ можно разрешить прерывание программы. Для подачи сигнала такого прерывания используется один вход процессора. Идентификация устройства процессором осуществляется с помощью чтения на шине данных вектора (номера ВУ). Специальная процедура на аппаратном или програмном уровне ставит в соответствие вектору начальный адрес подпрограммы обслуживания.

В процессе инициализации системы процессор записывает в регистр состояния ВУ единицу в бит разрешения прерывания, если это устройство будет работать в режиме прерывания.

Кроме этого процессор может записывать вектор в регистр вектора. Регистр вектора может быть тумблерным.

Когда ВУ готово к обмену, устанавливается бит готовности в регистре состояния своим контроллером. По совпадении сигналов готовности и разрешения прерывания формируется низким уровнем сигнал требование прерывания на общей однопроводной шине.

Процессор проверяет сигнал после выполнения команды и формирует последовательно два сигнала по шине управления: подготовка и подтверждение прерывания.

По сигналу подготовка во всех ВУ запрещается изменение всех триггеров. В каждом интерфейсе коммутируется путь прохождения сигнала подтверждение прерывания. Если ВУ выставляло требование прерывания, то цепь прохождения дальше сигнала ПП разрывается, а в этом ВУ по данному сигналу выдается вектор на шине данных, который принимается процессором.

**** Рисунок 2.1.4.1 Структурная схема централизованного КПП.

В целом аппаратно прерывания делятся на:

1) Централизированные (с централизированным КПП)

2) Децентрализированные (распределенным КПП)

Мы используем централизированный контролер ПП.

Один из вариантов реализации централизованного КПП показан на рисунке 2.1.4.1. Рассмотрим его детальнее.

Дешифратор МК преобразует МК прерывания в необходимые управляющие сигналы для блоков микросхемы. Рг Пр служит для записи входных сигналов прерывания, то есть IRQi (ВУ) и представляет собой 8-разрядный регистр. Фиксаторы функционируют как регистраторы отрицательных импульсов на входах Рг Пр. Рг Ь – 8-миразрядный регистр. Содержимое в этот регистр может быть выведено на шину М или загружено из него. Схема обнаружения прерывания СОПр выявляет наличие сигнала на любом из немаскированных выходов прерывания. 8-разрядный шифратор приоритетов (ПШ) определяет: какой из маскированных сигналов прерываний имеет наивысший приоритет и функционирует двоичный вектор прерывания. По МК чтения вектора READ\_VR этот двоичный вектор прерывания записывается в РгV. Впоследствии, сохраненное значение вектора может быть использовано дл сброса запросов на прерывание. В этой же МК устройство приращения прибавляет V=V+1 и новое значение будет зафиксировано в РгС (регистр слова состояния). Т.о. РгС указывает уровень на 1 больший, чем уровень, соответствующий последнему программируемому вектору. На шине V присутствует вектор прерывания. По этой же команде его можно считать с шины данных. Схема сравнения значений векторов прерываний указывает, когда вектор V>=S. С помощью схемы формирователя УС формируется запрос на прерывание, параллельный и последовательный запрет, и выходы межгрупповой связи. Сигнал SV (переполнение регистра состояния) служит для запрещения всех прерывании. Он указывает на то, что был прочитан вектор прерывания, имеющий наибольший приоритет и что регистр состояния заполнен.

Существует другой вариант реализации централизованного КПП на основе микросхем приоритетного прерывания КМ1804ВН1, которая позволяет обрабатывать запросы от 8 устройств. Одна микросхема КМ1804ВР3 осуществляет прием и кодирование от 8 микросхем КМ1804ВН1. Таким образом для реализации схемы на 116 внешних устройств потребуется 16 микросхем КМ1804ВН1 и 2 микросхемы КМ1804ВР3. Структурно-функциональная схема такого устройства прилагается ниже на рис. 2.1.4.2



Рисунок 2.1.4.1 Структурно функциональная схема централизованного КПП.

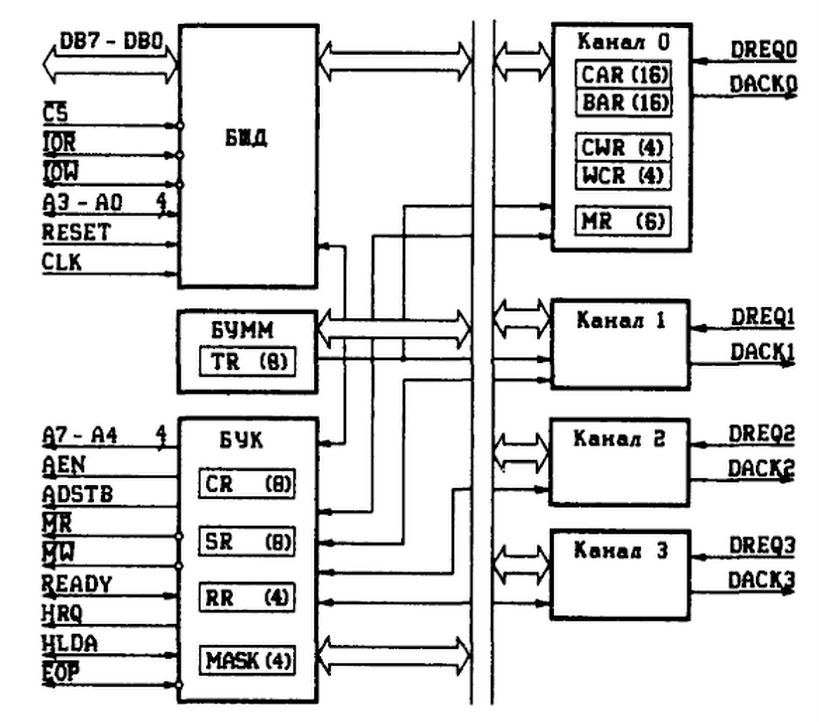
### 2.1.5 Прямой доступ к памяти (КПДП)

Данный режим используется для разгрузки процессора при обмене массивами данных между ОП и ВУ.

Инициатором обмена является процессор, который выполняет инициализацию контроллера прямого доступа к памяти и запускает его.

В дальнейшем два активных устройства (процессор и КПДП) поочередно захватывают шину, за счет чего осуществляется параллельная работа этих устройств.

КПДП имеет ряд адресов в адресном пространстве ВУ. На рис.1 КПДП представлен элементом КПДП.



**Рис. 2.1.5** Структурная схема КПДП



Рис. 2.1.5.1 Аппаратная реализация режима ПДП

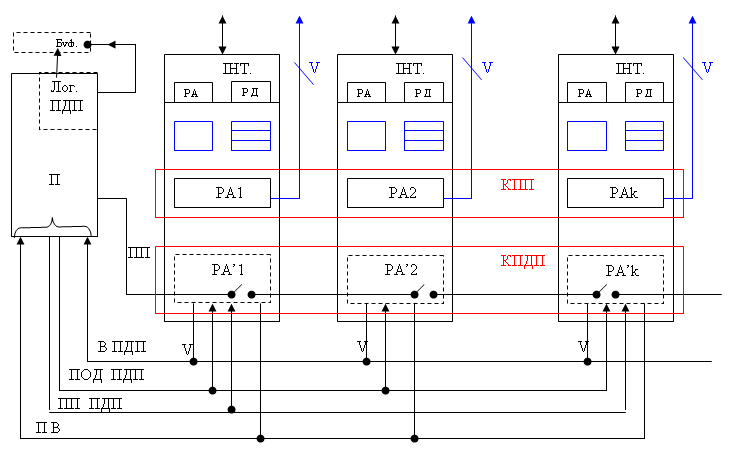


Рис. 2.1.5.2 Реализация децентрализированного КПП и КПДП

РАi – распределенный арбитр КПП

РА’i - распределенный арбитр КПДП

ТПДП – требование ПДП

ПОД ПДП – подготовка ПДП

ПП ПДП – подтверждение ПДП

ПВ – подтверждение выборки

В каждом ВУi присутствует схема РАi КПДП для подключения выборочного (например, i-го устройства) к ОП; Происходит «захват шины» ВУi, процессор отключается (сигналом ОЕ ПДП, который вырабатывается сигналом П).

# 3. Система команд

## 3.1 Основные команды

Основные команды МК48 включают в себя 96 основных команд и ориентированы на реализацию процедур управления. Все команды имеют формат один или два байта (70% команд однобайтные). Время выполнения команд составляет 2.5 или 5.0 мкс (один или дна машинных цикла соответственно) при тактовой частоте 6.0 МГц. Большинство команд выполняется за один машинный цикл. За два машинных цикла выполняются команды с непосредственным операндом, ввода/вывода и передачи управления.

Основная группа команд пересылки данных. Данная группа состоит из 24 команд. Все команды (кроме MOV PSW, А) не оказывают воздействия на флаги. Команды пересылки данных внутри МК выполняются за один машинный цикл, обмен с внешней памятью и портами требует двух машинных циклов. Пример некоторых операций показан в таблице 3.1:

**Таблица 3.1**

|  |  |
| --- | --- |
| Название команды | Мнемокод |
| Пересылка регистра в аккумулятор | MOV А,Rn |
| Пересылка байта из РПД в аккумулятор | MOV A,@Ri |
| Пересылка непосредственного операнда в аккумулятор | MOV A,#d |
| Пересылка аккумулятора в регистр | MOV Rn,A |
| Пересылка непосредственного операнда в регистр | MOV Rn,#d |
| Пересылка аккумулятора в РПД | MOV @Ri,A |
| Пересылка непосредственного операнда в РПД | MOV @Ri,#d |

### 3.1.1 Команды арифметические и логические

1. Группа команд арифметических операции. Данная группа состоит из 12 команд и позволяет выполнять следующие операции над 8-битными целыми двоичными числами без знака: двоичное сложение (АDD), двоичное сложение с учетом переноса (АDDС) , десятичная коррекция (DA) , инкремент (INС) и декремент (DЕС). Пример некоторых операций показан в таблице 3.1.1.1:

**Таблица** 3.1.1.1

|  |  |
| --- | --- |
| Название команды | Мнемокод |
| Сложение регистра с аккумулятором | ADD А,Rn |
| Сложение константы с аккумулятором | ADD A,#d |
| Сложение регистра с аккумулятором и переносом | ADDC A,Rn |
| Инкремент аккумулятора | INC А |
| Инкремент регистра | INC Rn |
| Декремент аккумулятора | DEC A |
| Декремент регистра | DEC Rn |

1. Группа команд логических операций. Данная группа состоит из 28 команд и позволяет выполнять следующие операции над байтами: дизъюнкцию, конъюнкцию, исключающее ИЛИ, инверсию, сброс и сдвиг. Две команды (сброс и инверсия) позволяют выполнять операции над битами. Пример некоторых операций показан в таблице 3.1.1.2:

**Таблица 3.1.1.2**

|  |  |
| --- | --- |
| Название команды | Мнемокод |
| Логическое И регистра и аккумулятора | ANL A,Rn |
| Логическое И константы и аккумулятора | ANL А,#d |
| Логическое ИЛИ регистра и  аккумулятора | ORL A,Rn |
| Логическое ИЛИ константы и  аккумулятора | ORL A,#d |
| Исключающее ИЛИ регистра и аккумулятора | XRL A,Rn |
| Исключающее ИЛИ константы и аккумулятора | XRL A,#d |
| Сброс аккумулятора | CLR A |
| Инверсия аккумулятора | CPL A |
| Циклический сдвиг влево аккумулятора | RL А |
| Циклический сдвиг вправо аккумулятора | RR А |
| Сброс переноса | CLR С |
| Сброс флага F1 | CLR F1 |
| Инверсия переноса | CPL С |
| Инверсия флага F1 | CPL F1 |

**Пример программы с приведённым алгоритмом:**

F = ((X1+X2)&(X3-X4-1))\2 – (X5 V X6) \*4

**Алгоритм:**

**Код программы:**

Начало

Конец

(X1+X2) -> R0

(X3-X4-1) -> A

(A&R0) -> A

Rrc (A) -> R0

Rlc (X5 V X6) -> A

Rlc (A)

(-A + 1) ->A

(A+R0) -> P1

;F = ((X1+X2)&(X3-X4-1))\2 – (X5 V X6) \*4

;Reading Xx

In A, P1

Mov R1, A

In A, P1

Mov R2, A

In A, P1

Mov R3, A

In A, P1

Mov R4, A

In A, P1

Mov R5, A

In A, P1

Mov R6, A

;X1+X2

Mov A, R2

Add A, R1

Clr C

Mov R0, A

;X3-X4-1

Mov A, R4

Cpl A

Add A, R3

Clr C

Anl A, R0 ;(X1+X2)&(X3-X4-1)

Clr C

Rrc A ;(X1+X2)&(X3-X4-1))\2

Clr C

Mov R0, A

Mov A, R5

Orl A, R6 ;(X5 V X6)

Clr C

Rlc A

Clr C

Rlc A ;(X5 V X6) \*4

Cpl A

Inc A ; -(X5 V X6) \*4 ->Dk

Add A, R0 ;((X1+X2)&(X3-X4-1))\2 – (X5 V X6) \*4

Clr C

Mov R0, A

Outl P1, A

### 3.1.2 Команды передачи управления.

Группа команд передачи управления. Данную группу образуют 19 команд передачи управления, из них две команды безусловного перехода, 14 команд условного перехода, команда вызова подпрограмм и две команды возврата из подпрограмм. Пример некоторых операций показан в таблице 3.1.2.1:

Таблица 3.1.2.1

|  |  |
| --- | --- |
| Название команды | Мнемокод |
| Безусловный переход | JMP ad11 |
| Декремент регистра и переход, если не нуль | DJNZ Rn,ad |
| Переход, если перенос | JC ad |
| Переход ,если нет переноса | JNC ad |
| Переход, если аккумулятор содержит нуль | JZ ad |
| Переход, если аккумулятор содержит не нуль | JNZ ad |
| Переход, если флаг F1 установлен | JF1 ad |
| Возврат из подпрограммы | RET |

### 3.1.2 Команды управления режимом работы МК

Группа команд управления режимом работы МК. В эту группу входят команды управления таймером/счетчиком, прерываниями и флагами переключения банков регистров и банков ПП. Пример некоторых операций показан в таблице 3.5:

Таблица 3.5

|  |  |
| --- | --- |
| Название команды | Мнемокод |
| Запуск таймера | STRT T |
| Запуск счетчика | STRT CNT |
| Останов таймера/счетчика | STOP TCNT |
| Выбор нулевого банка регистров | SEL RB0 |
| Выбор первого банка регистров | SEL RB1 |
| Выбор нулевого банка ПП | SEL МВ0 |
| Выбор первого банка ПП | SEL МВ1 |
| Холостая команда | NOP |

**Пример программы с приведённым алгоритмом:**

**

; установка порта Р1

ANL P1, #11000000B; условия X2 X1

;----------------- проверка X1

IN A, P1

JB6 LB1

; ----------------- вершина 1

MOV A,#11111001b ; [-7] дк

MOV T,A

ORL P1, #11001110b ; первая вершина - выдача y1y2y3

STRT T ; заддержка 7\*80=560

T11: JTF T12

JMP T11

T12: ANL P1,#0C0H;очистка выходных сигналов

; ----------------- вершина 2

ORL P1, #11100000b; вторая вершина - выдача y5

MOV R5, #2; задержка 3\*5=15

T21: DJNZ R5, T21

ANL P1,#0C0H;очистка выходных сигналов

; ----------------- вершина 3

LB3:

MOV A,#11111000b ; [-8] дк

MOV T,A

ORL P1, #11010010b ; третья вершина - выдача y4y1

STRT T ; заддержка 8\*80=640

T311: JTF T312

JMP T311

T312: MOV R3, #11; задержка 5+5\*11=60

T32: DJNZ R3, T32

ANL P1,#0C0H;очистка выходных сигналов

; ----------------- выход

JMP LB\_END

; ----------------- вершина 2

LB1:

ORL P1, #11100000b; вторая вершина - выдача y5

MOV R5, #2; задержка 3\*5=15

T22: DJNZ R5, T22

ANL P1,#0C0H;очистка выходных сигналов

;----------------- проверка X2

IN A, P1

JB7 LB2

JMP LB3

; ----------------- вершина 4

LB2:

MOV R4, #66; задержка 66\*5=330

ORL P1, #11010000b; четвертая вершина - выдача y4

T4: DJNZ R4, T4

ANL P1,#0C0H;очистка выходных сигналов

LB\_END:

ANL P1,#0C0H;очистка выходных сигналов

END

# 4. Программная часть

Х1\*Х2 Считывание с порта Р1

R1 - старшие разряды A

R2 - младшие разряды A

R3 - старшие разряды B

R4 - младшие разряды B

R6=0 R5 = 0

R7 = 10H - счётчик

R0= 0AH - адрес начала

CALL MUL

Результат: в памяти по адресам

0Dh 0Ch 0Bh 0Ah

## 4.1 Карта распределения ОП

|  |  |
| --- | --- |
| … | … |
| 0Dh | Старшие разряды A\*B |
| 0Ch | Старшие разряды A\*B |
| 0Bh | Младшие разряды A\*B |
| 0Ah | Младшие разряды A\*B |
| 0Eh | Частное от деления C/D |
| 10h | Старшие разряды Z |
| 0Fh | Младшие разряды Z |
|  |  |

Х3/Х4 Считывание с порта Р1

R2 - разряды C

R4 - разряды D

CALL DIVI

результат получаем в **R5**

и записываем его в память

**R5 0Eh**

Х5 Считывание с порта Р1

16h - старшие разряды Z

15h - младшие разряды Z

## 4.2 Алгоритм программы

Задание по варианту:

X = (A\*B) + (C/D) +Z

Суммирование результатов

Которые находятся по адресам:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A\*B | 0DH | 0CH | 0BH | 0AH |
| C/D | 0EH |  |  |  |
| Z | 10H | 0FH |  |  |

Итоговый результат будет находится по адресам

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| X | 0DH | 0CH | 0BH | 0AH |

Операнды 16-разрядные, данные вводятся с порта Р1

Блок-схема алгоритма вычисления выражения

## 4.3 Алгоритм умножения двух чисел

P3:=L1(P3)

P1:=R1(P1)

C:= C - 1

C = 0

Да

Нет

Х1\*=1

P2 := P2 + P1

Нет

Да

C := 16

Р1 – множитель;

С – счетчик;

Р3 – множитель;

Х1\* - старший разряд Р3;

Р2 – результат;

Р3 – 16 разрядный;

Р1, Р2 – 32 разрядный;

P3 🡨

R1 R2

R3 R4 R5 R6

2n-1

CM

2n-1

2n-1

@R0+3 @R0+2 @R0+1 @R0

P2

P1 🡪

## 4.4 Алгоритм деления двух чисел



Р4.R3 – делитель;

R2.R1 –делимое;

R5 – результат;

R5 🡨

R2.R1

2n

2n

CM

2n

R4.R3 🡪

## 4.5 Код микропрограммы

;A\*B + C/D + Z

In A, P1

Mov R1, A ; старшие разряды A

In A, P1

Mov R2, A ; младшие разряды A

In A, P1

Mov R3, A ; старшие разряды B

In A, P1

Mov R4, A ; младшие разряды B

Mov R6, #0

Mov R5, #0

Mov R7, #10H ; счётчик в умножении

Mov R0, #AH ; адрес начала младших разрядов результата

call Mull

;result D C B A

In A, P1

Mov R2, A ; C

In A, P1

Mov R4, A ; D

CALL Div

Mov R0, #0EH

Mov A, R5

Mov @R0, A ; запись частного в память

;result E

IN A,P1

MOV R0,#10h ; занесение адреса памяти

MOV @R0,A ; старшие разряды Z

IN A,P1

MOV R0,#0Fh ; занесение адреса памяти

MOV @R0,A ; младшие разряды Z

;result 10 F

Clr C

Mov R0, #CH

Mov A, @R0 ; считывания разрядов результата умножения A\*B

Mov R2, A

Mov R1, #0FH

Mov A, @R1 ; считывания разрядов Z

Addc A, R2 ; суммирование результататов и фиксация признака переноса

Mov @R0, A ; запись результата в память в ячейку Сh

Mov R0, #DH

Mov A, @R0 ; считывания разрядов результата умножения A\*B

Mov R2, A

Mov R1, #10H

Mov A, @R1 ; считывания разрядов результата умножения Z\*Z

Addc A, R2 ; суммирование результатов и фиксация признака переноса

Mov @R0, A ; запись результата в память в ячейку Dh

;--------------------

Clr C

Mov R0, #0EH

Mov A, @R0 ; считывания разрядов результата деления C/D

Mov R2, A

Mov R1, #DH

Mov A, @R1 ; считывания разрядов результата

Add A, R2 ; суммирование результататов и фиксация признака переноса

Mov @R1, A ; запись результата в память в ячейку 0Dh

;result in D C B A

jmp endofall

;-------------

Div:

;======== ПОДПРОГРАММА ДЕЛЕНИЯ ДВУХ ВОСЬМИРАЗРЯДНЫХ ЧИСЕЛ Х/У =======

; R1 - вспом. Х

; R2 - Х

; R3 - вспом. У

; R4 - У

; R5 - результат

; R0 - вспом. знак

; Инициализация

MOV R3, 0

MOV R1, 0

MOV R5, 0FFh

; Запоминание знака в R0

MOV R0, 80H

MOV A, R4

XRL A, R2

JB7 CLR\_SIGN

MOV R0, 0H

; Очистка знаков

CLR\_SIGN:

MOV A, R4

ANL A, 7FH

MOV R4, A

MOV A, R2

ANL A, 7FH

MOV R2, A

; Проверка знака остатка R2(1)

LOOP:

MOV A, R2

JB7 PREPARE\_ADD

; Перевод в ДК в случае вычитания R7.R6 = -R2.R1дк

SUBL:

MOV A, R3

CPL A

ADD A, 1

MOV R6, A

MOV A, R4

CPL A

ADDC A, 0

MOV R7, A

JMP ADDL

; Подготовка в случае сложения R7.R6 = +R2.R1

PREPARE\_ADD:

MOV A, R3

MOV R6, A

MOV A, R4

MOV R7, A

; Сложение R2.R1 = R2.R1 + R7.R6

ADDL:

CLR C

MOV A, R6

ADD A, R1

MOV R1, A

MOV A, R7

ADDC A, R2

MOV R2, A

; Получение цифры R2.R1 = R2.R1 + R7.R6

NUM:

MOV A, R5

RLC A

MOV R5, A

; Сдвиг l[R4.R3].C

SHR:

CLR C

MOV A, R4

RRC A

MOV R4, A

MOV A, R3

RRC A

MOV R3, A

; Условие выхода R3(1)

MOV A, R5

JB7 LOOP

; Восстановление знака

MOV A, R0

ORL A, R5

; Запись результата

MOV R5, A

Ret

;======== ПОДПРОГРАММА УМНОЖЕНИЯ ДВУХ 16-разрядных ЧИСЕЛ Х\*У =======

; R1 - старшие разряды Х

; R2 - младшие разряды Х

; R3 - старшие разряды У

; R4 - младшие разряды У

; R5 - вспомогательные разряды У

; R6 - вспомогательные разряды У

; R0 - адрес начала результата

Mull:

Mov A, R3 ; Старшие разряды в А

Clr C

Rrc A ; Сдвиг А вправо А[7] = 0 фиксируем признак с

Mov R3, A

Mov A, R4 ; разряды в А

Rrc A ; Сдвиг А вправо А[7] = с фиксируем признак с

Mov R4, A

Mov A, R5 ; разряды в А

Rrc A ; Сдвиг А вправо А[7] = с фиксируем признак с

Mov R5, A

Mov A, R6 ; Младшие разряды в А

Rrc A ; Сдвиг А вправо А[7] = с фиксируем признак с

Mov R6, A

Clr C ; очистка признак с

Ll0:

Mov A, R1

Jb7 Ll1 ; Проверка Х1\* (старшего разряда)

Ll3:; X1\*=0;

; P1 сдвигаем вправо -------------------------------------------------------------------

Mov A, R3 ; Старшиер разряды в А

Clr C

Rrc A ; Сдвиг А вправо А[7] = 0 фиксируем признак с

Mov R3, A

Mov A, R4 ; разряды в А

Rrc A ; Сдвиг А вправо А[7] = с фиксируем признак с

Mov R4, A

Mov A, R5 ; разряды в А

Rrc A ; Сдвиг А вправо А[7] = с фиксируем признак с

Mov R5, A

Mov A, R6 ; Младшие разряды в А

Rrc A ; Сдвиг А вправо А[7] = с фиксируем признак с

Mov R6, A

Clr C ; очистка признак с

; P3 сдвигаем влево --------------------------------------------------------------------

Mov A, R2

Rlc A ; Сдвиг А влево А[0] = 0 фиксируем признак с

Mov R2, A

Mov A, R1

Rlc A ; Сдвиг А влево А[0] = с фиксируем признак с

Mov R1, A

Jmp Mn ; Прыжок на метку

Ll1:

; Х1\* = 1 ------------------------------------------------------------------------------------

; Р2 = Р2 + Р1

; Считываем данные и сумируем. Результат записываем в память

Mov A, @R0 ; считали данные

Add A, R6 ; просумировали

Mov @R0, A ; результат обратно в память

Inc R0 ; R0 = R0 + 1 - для получения адреса следующих разрядов

Mov A, @R0 ; считали данные

Add A, R5 ; просумировали с учётом переноса

Mov @R0, A ; результат обратно в память

Inc R0 ; R0 = R0 + 1 - для получения адреса следующих разрядов

Mov A, @R0 ; считали данные

Add A, R4 ; просуммировали с учётом переноса

Mov @R0, A ; результат обратно в память

Inc R0 ; R0 = R0 + 1 - для получения адреса следующих разрядов

Mov A, @R0 ; считали данные

Add A, R3 ; просумировали с учётом переноса

Mov @R0, A ; результат обратно в память

Dec R0

Dec R0

Dec R0

Jmp Ll3

Mn:

Djnz R7, Ll0 ; прыжок на метку LLO если R7 не равно нулю и декримент R7

Ret

endofall:

end