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-01

Ромас А.С.

№ зач. кн. 0123

1111011

Руководитель:

Ткаченко В.В.

Киев 2012

**Содержание**

Введение

1.Обзор МПС

2.Архитектура МПС

2.1 Разработка структурной схемы

2.1.1 Память программ (ПП)

2.1.2 Память данных (ПД)

2.1.3 Внешние устройства (ВУ)

2.1.4 Режим прерываний (КПП)

2.1.5 Прямой доступ к памяти (КПДП)

3. Система команд

3.1 Основные команды

3.1.1 Команды арифметические и логические

3.1.2 Команды передачи управления.

3.1.3 Команды управления режимом работы МК

4. Программная часть

4.1 Алгоритм умноження 2-х чисел

5. Приложение

Код программи

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.

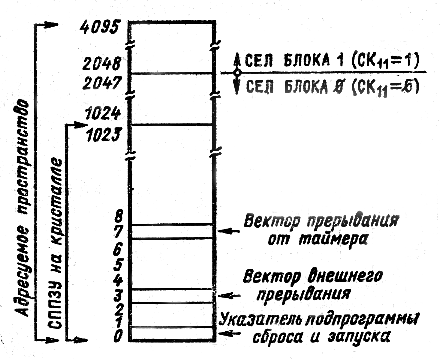
В состав разрабатываемой МПС должны входить процессор (П), основная память (ОП), содержащая ОЗУ и ПЗУ, а также внешние устройства (ВУ), контроллеры прерываний и прямого доступа к памяти.

Микропроцессор имеет 16-разрядную операционную часть. Объем внешней памяти программ 4К, внешней памяти данных – 64К. Шина адреса и данных объединённая, КПП и КПДП - централизованные. Каждое ВУ имеет фиксированный адрес в адресном пространстве периферийных 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



В резидентной памяти программ имеются три специализированных адреса:

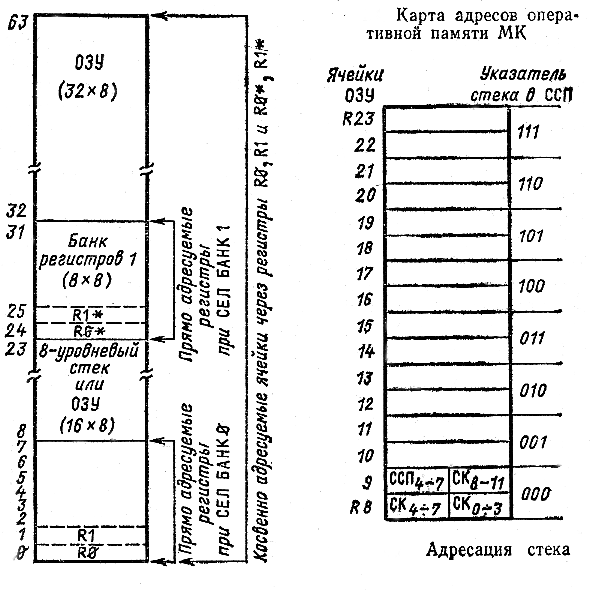
адрес 0, к которому передается управление сразу после окончания сигнала системного сброса СБРОС; по этому адресу должна находиться команда безусловного перехода к началу программы (точка входа после начальной установки);

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

Память данных(или - оперативная память, ОЗУ) - резидентная память данных емкостью 64 байта. Имеет в своем составе два банка рабочих регистров 0-7 и 24-31 по восемь регистров в каждом. Структура регистров (карта адресов) памяти данных МК представлена на следующем рисунке.   
  
  
  
Выбор одного из двух банков выполняется по команде "SEL RB". Рабочие регистры доступны командам с прямой адресацией, а все ячейки ОЗУ доступны по командам с косвенной адресацией. В качестве регистров косвенного адреса используются регистры R0, R1, R0\*, R1\*

Ячейки ОЗУ с адресами 8 - 23 адресуются указателем стека SP из ССП (PSW) [(см. рисунок регистра флагов в описании АЛУ)](http://www.kosmos.mk.ua/usmtu/mpk/k1816/k1816org2.html%22). Они могут быть использованы в качестве 8-уровневого стека (см. рис. карты адресов памяти МК выше). Если уровень вложенности подпрограмм меньше 8, то незадействованные в стеке регистры можно использовать как ячейки ОЗУ. Используемая ячейка определяется числом в 3-разрядном указателе стека SP, который является частью слова состояния программы. Порядок записи информации в стек приведен на рисунке. Загрузка в стек приводит к увеличению SP, чтение из стека - к уменьшению. При переполнении стека содержимое перезаписывается в регистры R8 и R9, т. е. SP изменяется со 111 на 000. При выполнении операций чтения из стека может происходить изменение SP с 000 на 111. Слово состояния программы РSW микро-ЭВМ может читаться и загружаться с помощью аккумулятора. Возможность загружать РSW необходима для перезапуска микро-ЭВМ после сбоя питания. Четыре старших бита PSW заносятся в стек и могут быть восстановлены при возврате с использованием специальной команды возврата.

МК 1816 не имеет команд загрузки байта в стек или его извлечения из стека, и в нем (в стеке) фиксируются только содержимое СК и старшая тетрада ССП (признаки). В связи с этим, не допускается во вложенных подпрограммах использование одноименных ячеек ОЗУ в качестве рабочих регистров. Если же такая необходимость возникает, то загрузку в стек и извлечение из него можно выполнять программно, путем передачи PSW в аккумулятор, выделения по маске указателя стека, передачи его в регистр косвенного адреса R0 или R1 с последующим обращением к ОЗУ по сформированому таким образом адресу вершины стека.

Практически все команды с обращением к ОЗУ оперируют с одним байтом. Однако по командам вызова и возврата осуществляется доступ к двухбайтным словам. В памяти данных слова хранятся так, что старший байт слова располагается в ячейке с большим адресом. Необходимо отметить, что в памяти программ порядок расположения байт по старшинству при хранении двухбайтных слов обратный.

В расширенных МК-системах, где используется внешнее ОЗУ, через регистры косвенного адреса R0 и R1 возможен доступ к внешней памяти объемом 256 байт. В случае необходимости в МК-системах можно применять внешнее ОЗУ требуемого объема, если, например, использовать 4 бита порта ввода - вывода информации для выбора одной из 16 страниц внешней памяти, каждая из которых имеет объем 256 байт. При этом 4-битный указатель страниц становится дополнением ССП и должен сохраняться в стеке при обработке прерываний.

2.1.3 Внешнее устройство

ВУ имеет фиксированный адрес в адресном пространстве периферийных устройств. Помимо регистра данных и регистра состояния каждое ВУ содержит регистр режима, регистр начальнога адреса ОП, регистр-счетчик. Эти регистры обеспечивают прямой доступ ВУ к памяти. Доступ к этим регистрам пользователь может получить через команды IN и OUT микропроцессора в адресном пространстве ВУ. Организация памяти MFT (с фиксированными разделами) с защитой по маскам. Структурная схема интерфейса ВУ представлена на рисунке 2.1.3.1

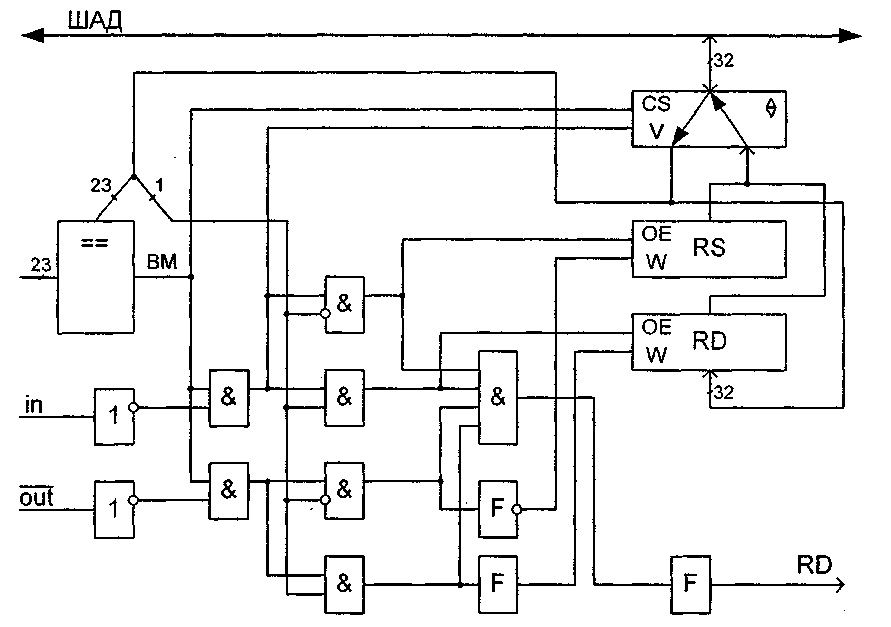


Рисунок 2.1.3.1 - Структурная схема интерфейса ВУ.

2.1.4 Режим прерываний (КПП)

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 деляться 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 програмистом 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. Регисьтр 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 одеопроводной 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 комутируется 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.

Мы используем централизованный контролер ПП.

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.

**** Рисунок 2.1.4.1 Структурная схема централизованного КПП.

**2.1.5 Прямой доступ к памяти (КПДП)**

Режим ПДП является самым скоро­стным способом обмена, который реализуется с помощью специальных аппа­ратных средств — контроллеров ПДП без использования программного обеспе­чения. Для осуществления режима ПДП контроллер должен выполнить ряд последовательных операций (рис. 6.1):

1) принять запрос DREQ на ПДП от ВУ;

2) сформировать запрос HRQ на захват шин для ЦП;

3) принять сигнал HLDA, подтверждающий этот факт после того, как ЦП войдет в состояние захвата (ШД, ША, ШУ в z-состоянии);

4) сформировать сигнал DACK, сообщающий ВУ о начале выполнения циклов ПДП;

5) сформировать на ША адрес ячейки памяти, предназначенный для обмена;

6) выработать сигналы MR, IOW и MW, IOR, обеспечивающие управление обменом;

7) по окончании ПДП либо повторить цикл ПДП, изменив адрес, либо прекратить ПДП, сняв запросы на ПДП.

Циклы ПДП выполняются с последовательно расположенными ячейками памяти, поэтому контроллер ПДП должен иметь счетчик адреса ОЗУ. Число циклов ПДП определяется специальным счетчиком. Управление обменом осу­ществляется специальной логической схемой, формирующей в зависимости от типа обмена пары управляющих сигналов: MR, IOW (циклы чтения), MW, IOR (циклы записи).

Из изложенного следует, что контроллер ПДП по запросу должен взять на себя управление системными шинами и выполнять совмещенные циклы чте­ния/вывода или записи/ввода до тех пор, пока содержимое счетчика циклов ПДП не будет равно нулю. На рис. 6.1 показана структурная схема МПС с контроллером ПДП.

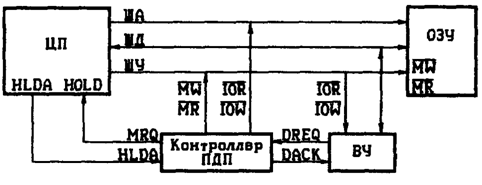


Рисунок 2.1.5.1 Структурная схема централизованного КПДП.

**Назначение выводов КПДП.**

**CLK** — вход для подключения тактового генератора FCLK —3 МГц.

**CS** — выбор кристалла. CS—0 разрешает работу КПДП.

**RESET** — сброс. Сигнал высокого уровня переводит КПДП в исходное состояние, устанавливая в нуль регистры команд, условий, временного хране­ния, а также устанавливая в единицу все разряды маски.

**READY** — готовность. Входной сигнал, используемый для синхронизации работы КПДП с медленнодействующими устройствами.

**HLDA** — подтверждение захвата. Входной сигнал, используемый ЦП для сообщения КПДП о возможности выполнения циклов ПДП.

**DREQ3—DREQ0** — входы запросов на ПДП от внешних устройств. Полярность запросов задается программно. Сигналы на этих входах должны удерживаться до прихода сигнала DACK. В исходном состоянии приоритет запросов естественный, DREQ0 имеет наивысший приоритет.

**DB7—DB0** — двунаправленная шина данных с буфером, имеющим z-co-стояние. В циклах ПДП на эти линии выдается восемь старших разрядов ад­ресного кода, которые необходимо «защелкнуть» на внешнем регистре сигна­лом ADSTB. В режиме работы с ЦП по этим линиям осуществляется прием/пе­редача данных.

**IOR** — чтение; как вход используется ЦП для чтения содержимого внутренних регистров КПДП; как выход в режиме ПДП разрешает выдачу данных из внешних устройств.

**IOW** — запись; как вход используется ЦП для загрузки данных в ре­гистры КПДП; как выход в режиме ПДП разрешает запись данных в регистры внешних устройств.

**ЕОР** — окончание процесса. Вход/выход, используемый для указания окончания процесса передачи данных в режиме ПДП. Подавая на этот вход сигнал низкого уровня, можно прекратить передачу данных. После завершения передачи данных по одному из каналов на выходе устанавливается сигнал

**ЕОР=0**. По этому сигналу (внешнему или внутреннему) снимается запрос и обслуживание прекращается. Если установлен режим автоинициализации, то происходит загрузка рабочих регистров данного канала содержимым базовых регистров, а разряды регистра маски не меняются. В режимах без автоинициа­лизации разряды маски и разряд ТС в слове-состоянии устанавливаются в со­ответствии с состоянием обслуженного канала. При передаче память — память вывод ЕОР ориентирован на выход, и по окончании счета на этом выходе фор­мируется сигнал. Если вывод ЕОР не используется, то он должен быть под­ключен через резистор к шине питания ( + 5 В) для предотвращения формиро­вания ложных сигналов окончания процесса.

**A3—АО** — адресные входы/выходы. Используются как входные в режиме работы с ЦП и для адресации к каналам и регистрам каналов КПДП. В режи­ме ПДП являются выходами, по которым передаются четыре младших разряда адреса ОЗУ.

**А7—А4** — адресные выходы, на которые в режиме ПДП передаются со­ответствующие разряды адреса ОЗУ. В режиме работы с ЦП переходят в z-co-стояние.

**HRQ** — выход запроса захвата на управление системной шиной. Запрос на ПДП ЦП.

**DACK3—DACKO** — подтверждение ПДП. Выходные линии, на которые выдаются сообщения для ВУ о возможности выполнения циклов ПДП. По­лярность сигнала задается программно. После сигнала RESET на выходах DACK устанавливается нуль.

**AEN**—разрешение адреса. AEN = 1 устанавливается на время выдачи восьми старших разрядов адреса ОЗУ на линии DB7—DB0.

**ADSTB** — строб адреса. Выход, на котором формируется импульс (строб), осуществляющий запись старших разрядов (AI5—А8) адреса ОЗУ с шин DB7—DB0 во внешний буферный регистр.

**MEMR** — чтение из памяти. Выход, используемый в режиме ПДП для управления операцией чтения из памяти.

**MEMW** — запись в память. Выход, используемый в режиме ПДП для управления операцией записи в память.

**Ucc** — шина питания ( + 5 В).

**GND** — общий.

**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 = 4(x1&x2)-(x3-x4+1)-(x5vx6-1)/2

**Алгоритм:**

**Код программы:**

Начало

Конец

x1&x2

4(x1&x2)

x3-x4+1

4(x1&x2)-(x3-x4+1)

(x5vx6-1)

(x5vx6-1)/2

4(x1&x2)-(x3-x4+1)-(x5vx6-1)/2

Sel Rb0

; r0 - l bits of x1

; r1 - h bits of x1

; r2 - l bits of x2

; r3 - h bits of x2

Ins A, Bus

Mov R0, A

Ins A, Bus

Mov R1, A

Ins A, Bus

Mov R2, A

Ins A, Bus

Mov R3, A

; (x1&x2)

; put result into x1

; for l bits

Mov A, R0

Anl A, R2

Mov R0, A

; for h bits

Mov A, R1

Anl A, R3

Mov R1, A

; 4\*(x1&x2)

; put result into x1

Mov A, R0

Rlc A

Mov R0, A

Mov A, R1

Rlc A

Mov R1, A

Mov A, R0

Rlc A

Mov R0, A

Mov A, R1

Rlc A

Mov R1, A

; r2 - l bits of x5

; r3 - h bits of x5

; r4 - l bits of x6

; r5 - h bits of x7

Ins A, Bus

Mov R2, A

Ins A, Bus

Mov R3, A

Ins A, Bus

Mov, R4, A

Ins A, Bus

Mov R5, A

; (x5vx6)

; put result into x5

Mov A, R2

Orl A, R4

Mov R2, A

Mov A, R3

Orl A, R5

Mov R3, A

; (x5vx6)-1

; put result into x5

Clr C

Mov A, #FFH

Add A, R2

Mov R2, A

Mov A, R3

Addc A, #FFH

Mov R3, A

; (x5vx6)/2

; put result into x5

Clr C

Mov A, R3

Rrc A

Mov R3, A

Mov A, R2

Rrc A

Mov R2, A

; r4 - l bits of x4

; r5 - h bits of x4

; r6 - l bils of x3

; r7 - h bits of x3

Ins A, Bus

Mov R4, A

Ins A, Bus

Mov R5, A

Ins A, Bus

Mov R6, A

Ins A, Bus

Mov R7, A

; (x3-x4)

; put result into x3

; make x4 negative

Clr C

Mov A, R4

Cpl A

Add A, #1

Mov R4, A

Mov A, R5

Cpl A

Addc A, #0

Mov R5, A

; subtraction

Clr C

Mov A, R6

Add A, R4

Mov R6, A

Mov A, R7

Addc A, R5

Mov R7, A

; (x3-x4)+1

; put result into x3

Clr C

Mov A, R6

Add A, #1

Mov R6, A

Mov A, R7

Addc A, #0

Mov R7, A

; 4(x1&x2)-(x3-x4+1)

; put result into x1

; make x3 negative

Clr C

Mov A, R6

Cpl A

Add A, #1

Mov R6, A

Mov A, R7

Cpl A

Addc A, #0

Mov R7, A

; subtraction

Clr C

Mov A, R0

Add A, R6

Mov R0, A

Mov A, R1

Addc A, R7

Mov R1, A

; 4(x1&x2)-(x3-x4+1) - (x5vx6-1)/2

; put result into x1

; make negative

Clr C

Mov A, R2

Cpl A

Add A, #1

Mov R2, A

Mov A, R3

Cpl A

Addc A, #0

Mov R3, A

; subtraction

Clr C

Mov A, R0

Add A, R2

Mov R0, A

Mov A, R1

Addc A, R3

Mov R3, 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

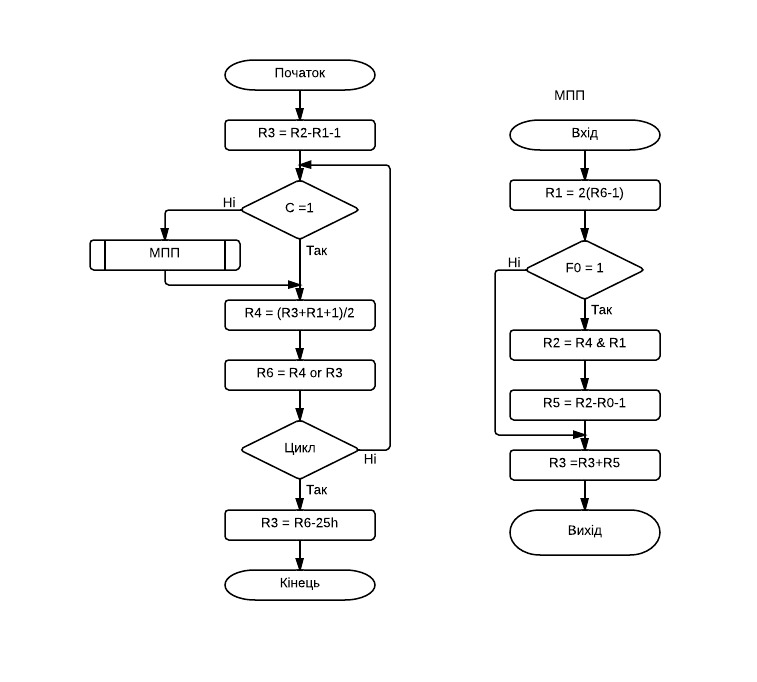
|  |  |
| --- | --- |
| Название команды | Мнемокод |
| Безусловный переход | JMP ad11 |
| Декремент регистра и переход, если не нуль | DJNZ Rn,ad |
| Переход, если перенос | JC ad |
| Переход ,если нет переноса | JNC ad |
| Переход, если аккумулятор содержит нуль | JZ ad |
| Переход, если аккумулятор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

**Пример программы с приведённым алгоритмом:**

**Алгоритм:**

Пример программы с приведённым алгоритмом:

Алгоритм:



Листинг программы:

Sel Rb0

Movd A, P5

Swap A

Mov R0, A

Movd A, P4

Add A, R0

Mov R0, A

Movd A, P5

Swap A

Mov R1, A

Movd A, P4

Add A, R1

Mov R1, A

Movd A, P5

Swap A

Mov R2, A

Movd A, P4

Add A, R2

Mov R2, A

Movd A, P5

Swap A

Mov R3, A

Movd A, P4

Add A, R3

Mov R3, A

Movd A, P5

Swap A

Mov R4, A

Movd A, P4

Add A, R4

Mov R4, A

Movd A, P5

Swap A

Mov R5, A

Movd A, P4

Add A, R5

Mov R5, A

Movd A, P5

Swap A

Mov R6, A

Movd A, P4

Add A, R6

Mov R6, A

Movd A, P5

Swap A

Mov R7, A

Movd A, P4

Add A, R7

Mov R7, A

; r3 = r2 - r1 - 1

Mov A, R1

Cpl A

Add A, #1

Add A, R2

Add A, #FFH

Mov R3, A

Loop:

; if c = 1

Jc Nompp

; mpp

; r1 = 2(r6 - 1)

Mov A, R6

Dec A

Clr C

Rlc A

Mov R1, A

Sel Rb1

Mov A, R1

Addc A, #0

Mov R1, A

Sel Rb0

Cpl F0

Jf0 Nof

; r2 = r4 & r1

Mov A, R4

Anl A, R1

Mov R2, A

; r5 = r2 - r0 - 1

Mov A, R0

Cpl A

Add A, #1

Add A, R2

Dec A

Mov R5, A

Nof:

Cpl F0

; r3 = r3 + r5

Mov A, R3

Add A, R5

Mov R5, A

Sel Rb1

Mov A, R3

Addc A, #0

Mov R3, A

Sel Rb0

Nompp:

; r4 = (r3+r1+1)/2

Clr C

Mov A, R3

Add A, R1

Inc A

Rrc A

Mov R4, A

; (r6 = r4 v r3)

Mov A, R4

Orl A, R3

Mov R6, A

Djnz R7, Loop

; r3 = r6 + 25h

Mov A, R6

Add A, #25H

Mov R3, A

Sel Rb1

Mov A, R3

Addc A, #0

Mov R3, A

Sel Rb0

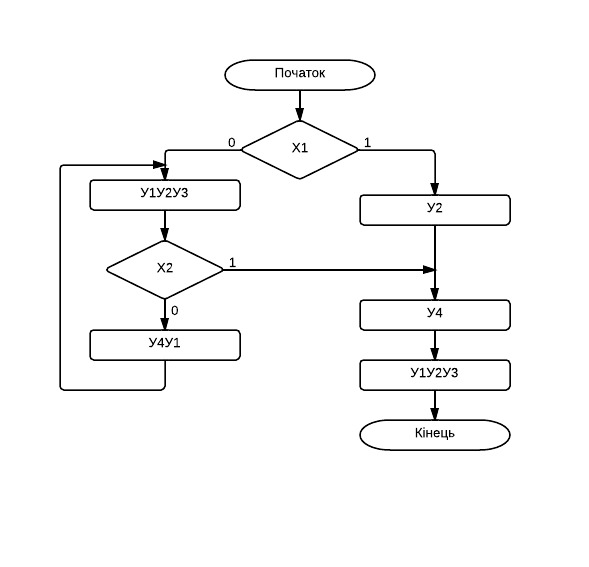
**3.1.2 Команды управления режимом работы МК**

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:**

****

Sel Rb0

Mov A, #0

Anld P5, A

Movd A, P4

Jb1 Top2

; top1

Top1:

Mov A, #7 ; 5

Orld P5, A ; 5

Nop

Nop

Mov A, #5

Anld P5, A

; y2 delay finished

; delay for y1 222

Mov A, #FEH ; 5

Mov T, A ; 2.5

Strt T ; 2.5

; begin 160

Lbl1:

Jtf End1

Jmp Lbl1

End1:

; end 160

Nop

; begin 30

Mov R7, #3 ; 5

Lbl2: Djnz R7, Lbl2

;end 30

Mov A, #4

Anld P5, A

; delay for y1 ended

Mov A, #FBH

Mov T, A

Strt T

Lbl4:

Jtf End2

Jmp Lbl4

End2:

Mov R7, #13

Lbl22: Djnz R7, Lbl22

Nop

Mov A, #0

Anld P5, A

; delay for y3 ended

Movd A, P4

Jb2 Top4

; top3

Top3:

Mov A, #9

Orld P5, A

Mov A, #FEH

Mov T, A

Strt T

Somelbl0: Jtf Somelbl3

Jmp Somelbl0

Somelbl3:

Mov R7, #6

Somelbl1: Djnz R7, Somelbl1

;somelbl2:

Nop

Mov A, #8

Anld P5, A

; delay for y1 ended

Mov A, #FDH

Mov T, A

Strt T

Somelbl2: Jtf Somelbl4

Jmp Somelbl2

Somelbl4:

Nop

Nop

Nop

Nop

Nop

Mov A, #0

Anld P5, A

; delay for y4 ended

Jmp Top1

; top2

Top2:

Mov A, #2

Orld P5, A

Nop

Nop

Mov A, #0

Anld P5, A

; delay for y2 ended

; top4

Top4:

Mov A, #4

Orld P5, A

Mov A, #FBH

Mov T, A

Strt T

Somelbl5: Jtf Somelbl6

Jmp Somelbl5

Somelbl6:

Mov R7, #14

Somelbl7: Djnz R7, Somelbl7

; delay for y4 ended

Mov A, #0

Anld P5, A

; top11

Top11:

Mov A, #7 ; 5

Orld P5, A ; 5

Nop

Nop

Mov A, #5

Anld P5, A

; y2 delay finished

; delay for y1 222

Mov A, #FEH ; 5

Mov T, A ; 2.5

Strt T ; 2.5

; begin 160

Ll1:

Jtf En1

Jmp Ll1

En1:

; end 160

Nop

; begin 30

Mov R7, #3 ; 5

Ll2: Djnz R7, Ll2

;end 30

Mov A, #4

Anld P5, A

; delay for y1 ended

Mov A, #FBH

Mov T, A

Strt T

Ll4:

Jtf En2

Jmp Ll4

En2:

Mov R7, #13

Ll22: Djnz R7, Ll22

Nop

Mov A, #0

Anld P5, A

; delay for y3 ended

End

**4. Программная часть**

**4.1 Алгоритм умножения 2-х чисел**

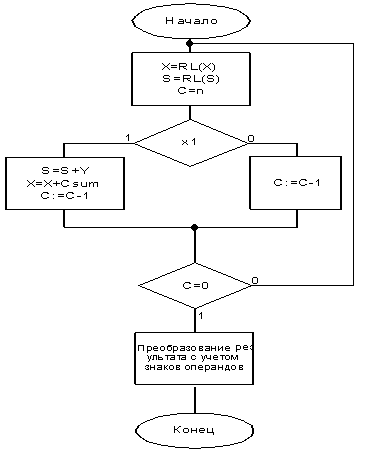
Блок-схема алгоритма вычисления выражения

Блок-схема алгоритма умножения 3-им способом:

1-й множитель Х

2-й множитель У

S – сумма накопления, операнды 16-разрядные



1

**X**🡨 n+1

2 **S**🡨 n+1

1 **СМ** n

1 **Y** n

X1

+1

MYMUL:

CLR F0; очистка признака F0

CLR F1; очистка признака F1

MOV A, R1; в аккумулятор старшие разряды 1го множителя для проверки знака

ANL A, #10000000B; если Х отрицательное,

JZ Point10; то инвертируем знаковый разряд

MOV A, R1; и инвертируем признак F0

ANL A, #01111111B; выделяем значимую часть множителя (старшие разряды)

MOV R1, A; в R1 положительное число

CPL F0; инвертируем признак F0 (признак 1-го множителя)

Point10:

MOV A, R3; загоняем в аккумулятор старшие разряды 2-го множителя для проверки знака

ANL A, #10000000B ; если Y отрицательное,

JZ Point4; то инвертируем знаковый разряд

MOV A, R3; и инвертирум признак F1

ANL A, #01111111B

MOV R3, A; в старших разрядах 2-го множителя положительное число

CPL F1; инвертируем признак F1

MOV A, R1

ANL A, #10000000B

JZ Point4

MOV A, R1

ANL A, #01111111B

MOV R1, A

CPL F0

JMP Point4

Point4:

MOV R7, #16; в счётчик записываем разрядность операндов - 16

Point3:

MOV A, R6; сдвиг младших разрядов S влево

RLC A

MOV R6, A

MOV A, R5; сдвиг влево накопителя суммы (старших разрядов)

RLC A

MOV R5, A

MOV A, R2; сдвиг влево младших разрядов 1-го множителя

RLC A

MOV R2, A

MOV A, R1; Сдвиг влево Х (старшие разряды)

RLC A

MOV R1, A

JC Point1

DEC R7; инкремент счётчика

MOV A, R7; проверяем следует ли заканчивать цикл

JNZ Point3; зацикливаем

JMP Next2

Point1:

MOV A, R6; если получили перенос = 1

ADD A, R4; к накопителю суммы прибавляем 2-ой множитель

MOV R6, A; назад в S

JC Point2

JMP Next1

Point2:

INC R5; при возникновении переноса, следует прибавить 1 к старшим разрядам S

Next1:

MOV A, R5; суммируем 2-ой множитель и S (старшие разряды)

ADD A, R3

MOV R5, A

JC Point8

JMP Next3

Point8:

INC R2; инкремент младших рязрядов 1-го множителя

JC Point9; при возникновении переноса

JMP Next3

Point9:

INC R1; инкремент старших разрядов 1-го множителя

Next3:

CLR C; очистка переноса

DEC R7; декремент счётчика

MOV A, R7; проверка счётчика на 0

JNZ Point3

Next2:

JF0 Point5; определение знака результата

JMP Point6

Point5:

JF1 Point7

JMP Point11

Point7:

MOV A, R1; формирование положительного результата

ANL A, #01111111B

MOV R1, A; если были одинаковые знаки множителей

JMP Progend

Point6:

JF1 Point11

JMP Progend

Point11:

MOV A, R1; если знаки множителей различны, то результат отрицательный

ORL A, #10000000B

MOV R1, A

Progend:

RET; возврат в программу

**Приложение**

Код микропрограммы

; X = (X1-X2)+(X3\*X4)+(X7+X8)^2

; sposib - 3; rozrjadnist - 16

; pk

;

; result 20h.21h.22h.23h

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

X1\_high EQU #0FFh

X1\_low EQU #0FFh

X2\_high EQU #07Fh

X2\_low EQU #0FFh

X3\_high EQU #0FFh

X3\_low EQU #0FFh

X4\_high EQU #0FFh

X4\_low EQU #0FFh

X7\_high EQU #0FFh

X7\_low EQU #0FFh

X8\_high EQU #0FFh

X8\_low EQU #0FFh

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

; X1-X2

; put result into 20h.21h.22h.23h

SEL RB1

MOV R0, #22h

MOV R4, X1\_high

MOV A, R4

MOV @R0, A

INC R0

MOV R4, X1\_low

MOV A, R4

MOV @R0, A

INC R0

MOV R4, X2\_high

MOV A, R4

CPL A

ADD A, #1

MOV @R0, A

INC R0

MOV R4, X2\_low

MOV A, R4

CPL A

ADDC A, #0

MOV @R0, A

CALL adding

; X3 \* X4

SEL RB1

MOV R1, X3\_high

MOV R2, X4\_high

SEL RB0

MOV R1, X3\_low

MOV R2, X4\_low

CALL multiply

; put result into 24h.25h.26h.27h

SEL RB1

MOV R0, #24h

MOV A, R4

MOV @R0, A

INC R0

SEL RB0

MOV A, R4

SEL RB1

MOV @R0, A

SEL RB1

INC R0

MOV A, R5

MOV @R0, A

INC R0

SEL RB0

MOV A, R5

SEL RB1

MOV @R0, A

; (X1-X2) + (X3 \* X4)

CALL adding

; X7+X8

; put result into 20h.21h.22h.23h

SEL RB1

MOV R0, #22h

MOV R4, X1\_high

MOV A, R4

MOV @R0, A

INC R0

MOV R4, X1\_low

MOV A, R4

MOV @R0, A

INC R0

MOV R4, X2\_high

MOV A, R4

ADD A, #1

MOV @R0, A

INC R0

MOV R4, X2\_low

MOV A, R4

MOV @R0, A

CALL adding

; (X7+X8)^2

CALL multiply

; put result into 24h.25h.26h.27h

SEL RB1

MOV R0, #24h

MOV A, R4

MOV @R0, A

INC R0

SEL RB0

MOV A, R4

SEL RB1

MOV @R0, A

SEL RB1

INC R0

MOV A, R5

MOV @R0, A

INC R0

SEL RB0

MOV A, R5

SEL RB1

MOV @R0, A

; (X1-X2)+(X3\*X4)+(X7+X8)^2

CALL adding

; multiply

; h1 R1 RB1

; l1 R1 RB0

; h2 - R2 RB1

; l2 - R2 RB0

; result

; h R4\_RB1.R4\_RB0

; l R5\_RB1.R5\_RB0

multiply:

; clear registers

SEL RB0

MOV A, R3

XRL A, R3

MOV R3, A

MOV A, R4

XRL A, R4

MOV R4, A

MOV A, R5

XRL A, R5

MOV R5, A

SEL RB1

MOV A, R3

XRL A, R3

MOV R3, A

MOV A, R4

XRL A, R4

MOV R4, A

MOV A, R5

XRL A, R5

MOV R5, A

; save sign

SEL RB1

CLR F1

MOV A, R1

XRL A, R2

JB7 sign\_minus ; decide for sign

JMP sign\_plus

sign\_minus:

CPL F1

sign\_plus:

MOV A, R1

ANL A, #7Fh

MOV R1, A

MOV A, R2

ANL A, #7Fh

MOV R2, A

; right shift

CLR C

SEL RB1

MOV A, R2

RRC A

MOV R2, A

SEL RB0

MOV A, R2

RRC A

MOV R2, A

SEL RB1

MOV A, R3

RRC A

MOV R3, A

loop:

; check h bit

SEL RB1

MOV A, R1

CPL A

JB7 zero

; Z: R4\_RB1.R4\_RB0.R5\_RB1.R5\_RB0

; Y: R2\_RB1.R2\_RB0.R3\_RB1.R3\_RB0

; sum

SEL RB0

CLR C

MOV A, R5

ADDC A, R3

MOV R5, A

SEL RB1

MOV A, R5

ADDC A, R3

MOV R5, A

SEL RB0

MOV A, R4

ADDC A, R2

MOV R4, A

SEL RB1

MOV A, R4

ADDC A, R2

MOV R4, A

zero:

; left shift

CLR C

SEL RB0

MOV A, R1

RLC A

MOV R1, A

SEL RB1

MOV A, R1

RLC A

MOV R1, A

; right shift

CLR C

SEL RB1

MOV A, R2

RRC A

MOV R2, A

SEL RB0

MOV A, R2

RRC A

MOV R2, A

SEL RB1

MOV A, R3

RRC A

MOV R3, A

SEL RB0

MOV A, R3

RRC A

MOV R3, A

; check equals 0

SEL RB1

MOV A, R1

JNZ loop

; result sign

JF1 rez\_minus

JMP rez\_plus

rez\_minus:

SEL RB1

MOV A, R4

ORL A, #80h

MOV R4, A

rez\_plus:

RET

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

;sum program

; h1 - 20h.21h

; l1 - 22h.23h

; h2 - 24h.25h

; l2 - 26h.27h

; put result into

; 20h.21h.22h.23h

adding:

SEL RB0

MOV R0, #20h

MOV A, @R0

JB7 sign\_add\_first\_minus

JMP sign\_add\_first\_plus

sign\_add\_first\_minus:

ANL A, #7Fh

MOV @R0, A

MOV R0, #23h

MOV A, @R0

CPL A

MOV @R0, A

DEC R0

MOV A, @R0

CPL A

MOV @R0, A

DEC R0

MOV A, @R0

CPL A

MOV @R0, A

DEC R0

MOV A, @R0

CPL A

MOV @R0, A

CLR C

MOV R0, #23h

MOV A, @R0

ADDC A, #1

MOV @R0, A

DEC R0

MOV A, @R0

ADDC A, #0

MOV @R0, A

DEC R0

MOV A, @R0

ADDC A, #0

MOV @R0, A

DEC R0

MOV A, @R0

ADDC A, #0

MOV @R0, A

sign\_add\_first\_plus:

MOV R0, #24h

MOV A, @R0

JB7 sign\_add\_second\_minus

JMP sign\_add\_second\_plus

sign\_add\_second\_minus:

ANL A, #7Fh

MOV @R0, A

MOV R0, #27h

MOV A, @R0

CPL A

MOV @R0, A

DEC R0

MOV A, @R0

CPL A

MOV @R0, A

DEC R0

MOV A, @R0

CPL A

MOV @R0, A

DEC R0

MOV A, @R0

CPL A

MOV @R0, A

CLR C

MOV R0, #27h

MOV A, @R0

ADDC A, #1

MOV @R0, A

DEC R0

MOV A, @R0

ADDC A, #0

MOV @R0, A

DEC R0

MOV A, @R0

ADDC A, #0

MOV @R0, A

DEC R0

MOV A, @R0

ADDC A, #0

MOV @R0, A

sign\_add\_second\_plus:

CLR C

SEL RB1

MOV R0, #23h

MOV R1, #27h

MOV R2, #4h

add\_loop:

MOV A, @R0

ADDC A, @R1

MOV @R0, A

DEC R0

DEC R1

DJNZ R2, add\_loop

RET

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

ending:

END