

Введение

Использование вычислительной техники предполагает наличие глубоких знаний, объединяющих умение программно управлять процессами вычислений, а также умение оценивать сущность процессов, происходящих в ЭВМ, на аппаратном уровне. Как правило, в производственном процессе применяются микропроцессорные системы. Относительно низкая стоимость, малые габариты и потребляемая мощность, высокая надежность и исключительная гибкость в применении ставят микропроцессорные приборы вне конкуренции по сравнению с любой другой элементной базой цифровых устройств.

Основной технической базой автоматизации управления технологическими процессами являются специализированные микропроцессорные устройства (МПУ). Они являются предметом изучения на этапе базовой подготовки, предшествующей рассмотрению в специальных дисциплинах различных применений микропроцессорных устройств.

При изучении специализированных микропроцессорных устройств рассматриваются приемы проектирования как аппаратных, так и программных средств МПУ. Проектирование аппаратных средств требует знания особенностей микропроцессорных комплектов микросхем различных серий и функциональных возможностей микросхем, входящих в состав используемого комплекта, умения правильно выбирать серию. Проектирование программных средств требует знаний, необходимых для выбора метода и алгоритма решения задач, входящих в функции микропроцессорного устройства, для составления программ (часто с использованием языков низкого уровня – языка кодовых комбинаций, языка Ассемблера), а также умения использовать средства отладки программ.

Основными задачами цикла лабораторных работ по дисциплине Микропроцессорные системы в автоматизации и управлении являются: изучение структуры, режимов функционирования и состава команд микропроцессорной системы, а также формирование у студентов начальных навыков программирования на языке ассемблер и машинных кодов. Лабораторные работы выполняются на учебных микропроцессорных комплексах (УМК), построенных на базе микропроцессора KP580BM80A.

Учебный микропроцессорный комплекс позволяет практически овладеть методикой разработки и отладки программ. На основе знаний, полученных в процессе работы с УМК, значительно легче понять принцип функционирования микропроцессорных систем на основе микропроцессоров других серий.

Лабораторная работа №1

Цель работы: изучение структуры микропроцессорной системы на основе УМК, системы команд микропроцессора КР580ВМ80А (МП КР580) и принципов построения программ в машинных кодах для микропроцессорных систем управления.

Основные положения:

УМК имеет структуру, представленную на рисунке 1.

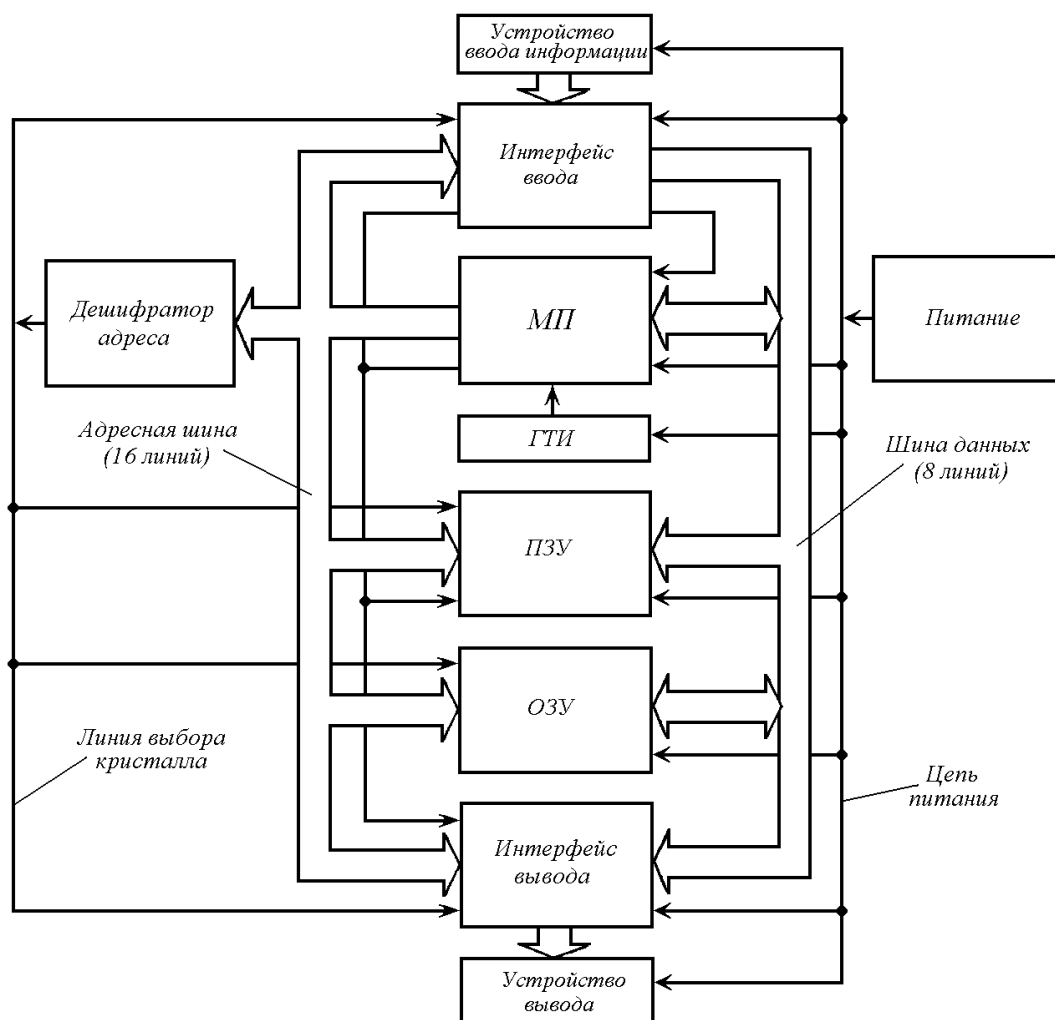


Рисунок 1. Структурная схема микропроцессорной системы

В качестве основных элементов микропроцессорной системы (МП) можно определить оперативное запоминающее устройство (ОЗУ), постоянное запоминающее устройство (ПЗУ), центральный процессор (ЦП) и интерфейсы ввода-вывода информации.

МП (ЦП) является центром всех операций. Ему необходимы питание и тактовые импульсы. Генератор тактовых импульсов (ГТИ) может быть отдельным устройством или входить в состав кристалла МП. ГТИ предназначен для синхронизации функционирования всех устройств входящих в состав МП-

системы. Дешифратор адреса обеспечивает подключение одного из устройств МП-системы к шине адреса. На вход дешифратора адреса подключены 4 линии адресной шины.

Типовым примером микропроцессорной системы является учебный микропроцессорный комплекс, построенный на базе микропроцессора КР580ВМ80А. УМК содержит шестнадцать адресных линий, которые составляют однонаправленную шину адресов, а также восемь линий, которые составляют двунаправленную шину данных. Ввод данных осуществляется с помощью цифровой клавиатуры, содержащей основные буквенные символы шестнадцатеричной системы счисления.

МП КР580 имеет ПЗУ и ОЗУ, объемом 2 Кбайта и 1 Кбайт соответственно. Схематическое представление полей памяти УМК представлено на рисунке 3. Первый Кбайт памяти ПЗУ отводится под программу «Монитор», которая загружается в верхнюю область ОЗУ при включении МП-комплекса. Удаление или изменение области ОЗУ, предназначенной для работы программы «Монитор», приводит к неработоспособности УМК (требуется перезагрузка). Второй Кбайт ПЗУ используется для хранения описаний команд, используемых пользователем во время выполнения программы.

Программы пользователя могут помещаться только в «ОЗУ пользователя». При работе программы пользователь имеет право использовать адресное пространство соответствующее «ОЗУ пользователя». Не соблюдение данных правил приводит к неработоспособности УМК, либо к потере данных.

Как было ранее сказано, в МП КР580 существует семь регистров, которые пользователь может использовать для хранения и изменения данных. Также в МП КР580 имеется несколько внутренних регистров, которые пользователь может просматривать в процессе выполнения программы, а также содержимое этих регистров может влиять на выполнение программы пользователя. Некоторые регистры применяются в качестве специализированных, и используются непосредственно МП.

Блок регистров. МП КР580ИК80А содержит следующие программно-доступные 8-разрядные регистры:

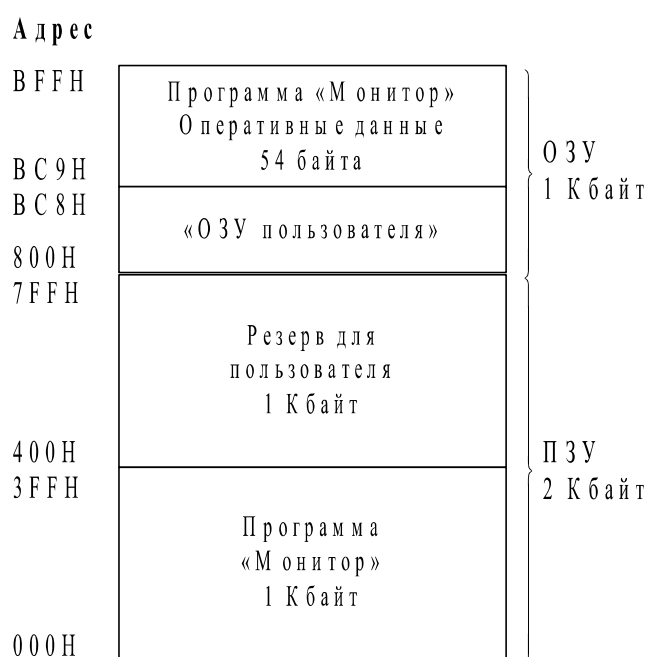


Рисунок 3. Поля памяти учебного микропроцессорного комплекса

1. Аккумулятор (или регистр *A*) является ядром всех операций МП, к которым относятся арифметические, логические, загрузки или размещения данных и ВВ. Это 8-разрядный регистр.

2. Регистры общего назначения *BC*, *DE* и *HL* могут быть использованы как шесть 8-разрядных или три 16-разрядные пары регистров в зависимости от текущей выполняемой команды. Как и в типовом МП, пара *HL* (фирмой Intel названа указателем данных) может быть использована для указания адреса. Несколько команд используют пары *BC* и *DE* в качестве указателя адреса, но обычно они являются регистрами хранения данных.

3. Счетчик команд *PC* всегда указывает на ячейку памяти следующей для выполнения команды.

4. Указатель стека *SP* является специальным регистром—указателем адреса (или данных), который всегда указывает на вершину стека в ОЗУ. Это 16-разрядный регистр.

5. Регистр признаков (или индикаторов) содержит пять одноразрядных индикаторов, в которых содержится информация, относящаяся к состоянию МП. Эти указатели используются условными ветвлениями программы, вызовами подпрограмм и возвратами из подпрограмм.

S	Z	0	AC	0	P	1	CY
---	---	---	----	---	---	---	----

S - Признак «знака»; принимает значение старшего разряда результата; если результат отрицательный S=1;

Z - Признак нуля; если результат равен нулю, то Z=1, иначе Z=0

AC - Признак вспомогательного переноса; если есть перенос между тетрадами байта, то AC=1, иначе AC=0

P - Признак четности; если число единиц в байте результата четно, то P=1, иначе P=0

CY - Признак переноса (заема); если при выполнении команды возник перенос из старшего разряда или заем в старший разряд, то CY=1, иначе CY=0.

Модификация регистра признаков происходит после любой арифметической, либо логической операций. Команды, использующие регистр признаков, анализируют состояние одного из битов и производят передачу управления программой по адресу, указанному в командной строке.

Общие регистры (*B*, *C*, *D*, *E*, *H*, *L*) используются для хранения операндов, промежуточных и конечных результатов, а также адресов и индексов при косвенной и индексной адресациях.

Аккумулятор (регистр *A*) используется в качестве источника одного из операндов и места, где фиксируется результат операции. В команде аккумулятор в явном виде не адресуется. На использование аккумулятора в операции указывает код операции команды. Иначе говоря, в отношении аккумулятора применяется подразумеваемая адресация, что позволяет применять одноадресные команды, имеющие сравнительно короткий формат. Использование аккумулятора и общих регистров позволяет при выполнении

команд уменьшить количество обращений к памяти и тем самым повысить быстродействие МП.

Наличие в блоке регистров специализированного регистра косвенного адреса HL позволяет иметь команды с подразумеваемой косвенной адресацией, т. е. без указания в команде номера регистра, хранящего исполнительный адрес.

Особенностью блока регистров МП является наличие в его составе схемы инкрементатора/декрементатора, которая производит над содержимым регистров (без привлечения АЛУ) операцию прибавления/вычитания единицы. Схема инкрементатора/декрементатора позволяет реализовать процедуры автоматического задания приращений при операциях с адресами не только в регистре-указателе стека, но и в счетчике команд.

При выполнении операций в МП возникает потребность в кратковременном хранении некоторых операндов и результатов выполнения операций. Для этой цели служат регистры временного хранения данных W и Z, которые являются программно-недоступными для пользователя. Использование регистров временного хранения позволяет МП за один цикл выполнения команды реализовать, например, такую операцию, как обмен содержимым двух регистров.

Для повышения эффективности операций со словами двойной длины и операций формирования и пересылок двухбайтных адресов имеется возможность оперировать с содержимым пар регистров B и C, D и E, H и L как с составными словами двойной длины, т.е. в МП автоматически выполняется операция конкатенации над содержимым пары регистров. При этом реализуются так называемые тандемные пересылки, состоящие в передаче в цикле выполнения команды последовательно друг за другом 2 байт информации.

В состав блока регистров входит регистр-защелка адреса памяти PC. Собственно регистр адреса недоступен программисту. Однако любая пара регистров (BC, DE, HL) может быть использована для задания адресов команд и данных в программе. Этот адрес под воздействием соответствующих команд не только может быть загружен в регистр-защелку адреса, но и модифицирован (посредством схемы инкрементатор/декрементатор) в процессе загрузки. Регистр-защелка адреса передает адрес в буферную схему и далее в шину адреса.

Имея последовательность действий выполняемых программой можно составить программу на языке ассемблер. Для этого надо подобрать мнемокод (мнемокод или мнемоника – условное обозначение команды языка ассемблер) для каждого действия (столбец 3 таблицы 1). Мнемокоды действий выполняемых МП КР580 приведены в приложении 1.

Форматы команд

Основная функция любого процессора – это выполнение команд. Типовая команда, в общем случае, должна указывать:

- подлежащую выполнению операцию;
- адреса исходных данных (операндов), над которыми выполняется операция;

- адрес, по которому должен быть помещен результат операции.

В соответствии с этим команда состоит из двух частей: операционной и адресной (рисунок 1).

операционная	адресная
--------------	----------

Рисунок 1 - Структура команды

Формат команды определяет ее структуру, то есть количество двоичных разрядов, отводимых под всю команду, а также количество и расположение отдельных полей команды.

Поле называется совокупность двоичных разрядов, кодирующих составную часть команды. При создании ЭВМ выбор формата команды влияет на многие характеристики будущей машины. Оценивая возможные форматы, нужно учитывать следующие факторы: общее число различных команд; общую длину команды; тип полей команды (фиксированной или переменной длины) и их длина; простоту декодирования; адресуемость и способы адресации; стоимость оборудования для декодирования и исполнения команд.

Длина команды. Это важнейшее обстоятельство, влияющее на организацию и емкость памяти, структуру шин, сложность и быстродействие ЦП. С одной стороны, удобно иметь в распоряжении мощный набор команд, то есть как можно больше кодов операций, операндов, способов адресации, и максимальное адресное пространство. Однако все это требует выделения большего количества разрядов под каждое поле команды, что приводит к увеличению ее длины. Вместе с тем, для ускорения выборки из памяти желательно, чтобы команда была как можно короче, а ее длина была равна или кратна ширине шины данных. Для упрощения аппаратуры и повышения быстродействия ЭВМ длину команды обычно выбирают кратной байту, поскольку в большинстве ЭВМ основная память организована в виде 8-битовых ячеек. В рамках системы команд одной ВМ могут использоваться разные форматы команд. Обычно это связано с применением различных способов адресации. В таком случае в состав кода команды вводится поле для задания способа адресации (СА), и обобщенный формат команды приобретает вид, показанный на рисунке 2. В большинстве ВМ одновременно уживаются несколько различных форматов команд.

Коп	СА	Адресная часть
-----	----	----------------

Рисунок 2 - Обобщенный формат команды

Разрядность поля кода операции. Количество двоичных разрядов, отводимых под код операции, выбирается так, чтобы можно было представить любую из операций. При заданной длине кода команды приходится искать компромисс между разрядностью поля кода операции и адресного поля.

Большее количество возможных операций предполагает длинное поле кода операции, что ведет к сокращению адресного поля, то есть к сужению адресного пространства. Для устранения этого противоречия иногда длину поля кода операции варьируют. Изначально под код операции отводится некое фиксированное число разрядов, однако для отдельных команд это поле расширяется за счет нескольких битов, отнимаемых у адресного поля.

Разрядность адресной части. В адресной части команды содержится информация о местонахождении исходных данных и месте сохранения результата операции. Обычно местонахождение каждого из операндов и результата задается в команде путем указания адреса со соответствующей ячейки основной памяти или номера регистра процессора. Принципы использования информации из адресной части команды определяет **систем адресации**. Система адресации задает **число адресов в команде** команды и приняты **способы адресации**.

Количество адресов в команде. Для определения количества адресов, включаемых в адресную часть, используют термин **адресность**. В «максимальном» варианте необходимо указать три компонента: адрес первого операнда, адрес второго операнда и адрес ячейки, куда заносится результат операции. В принципе может быть добавлен еще один адрес, указывающий место хранения следующей инструкции. В итоге имеет место **четыреадресный формат команды** (рисунок 3).

Операция	Адреса			
Код операции	1-й операнд	2-й операнд	Результат	Следующая команда

Рисунок 3 - Четыреадресный формат команды

В фон-Неймановских ЭВМ необходимость в четвертом адресе отпадает, поскольку команды располагаются в памяти в порядке их выполнения, и адрес очередной команды может быть получен за счет простого увеличения адреса текущей команды в счетчике команд. Это позволяет перейти к **трехадресному формату команд** (Рисунок 4). Требуется только добавить в систему команд ЭВМ команды, способные изменять порядок вычислений.

Операция	Адреса		
Код операции	1-й операнд	2-й операнд	Результат

Рисунок 4 - Трехадресный формат команды

К сожалению, и в трехадресном формате длина команды может оказаться весьма большой. Так, если адрес ячейки основной памяти имеет длину 32 бита, а длина кода операции - 8 бит, то длина команды составит 104 бита (13 байт).

Если по умолчанию взять в качестве адреса результата адрес одного из операндов (обычно второго), то можно обойтись без третьего адреса, и в итоге получаем **двухадресный формат команды** (Рисунок 5). Естественно, что в этом случае соответствующий операнд после выполнения операции теряется.

Операция	Адресат	
Коп	1 операнд	2 операнд / результат

Рисунок 5 - Двухадресный формат команды

Команду можно еще более сократить, перейдя к **одноадресному формату** (Рисунок 6), что возможно при выделении определенного стандартного места для хранения первого операнда и результата. Обычно для этой цели используется специальный регистр центрального процессора (ЦП), известный под названием **аккумулятора**, поскольку здесь аккумулируется результат.

Операция	Адрес
Код операции	1-й или 2-й операнд

Рисунок 6 - Одноадресный формат команды

Применение единственного регистра для хранения одного из операндов и результата является ограничивающим фактором, поэтому помимо аккумулятора часто используют и другие регистры ЦП. Так как число регистров в ЦП невелико, для указания одного из них в команде достаточно иметь сравнительно короткое адресное поле. Соответствующий формат носит название **полутораадресного** или **регистрового формата** (Рисунок 7).

Операция	Адреса	
Код операции	Регистр	2-й операнд

Рисунок 7- Полутораадресный формат команды

Наконец, если для обоих операндов указать четко заданное местоположение, а также в случае команд, не требующих операнда, можно получить **нультадресный формат команды** (Рисунок 8).

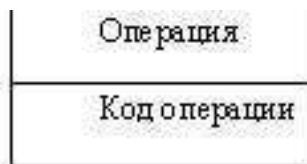


Рисунок 8 - Нульадресный формат команды

В таком варианте адресная часть команды вообще отсутствует или не задействуется.

Выбор адресности команд. При выборе количества адресов в адресной части команды обычно руководствуются следующими критериями:

- емкостью запоминающего устройства, требуемой для хранения программы;
- временем выполнения программы;
- эффективностью использования ячеек памяти при хранении программы.

Адресность и емкость запоминающего устройства

Емкость запоминающего устройства для хранения программы E_A можно оценить из соотношения:

$$E_A = N_A \cdot R_K,$$

где N_A - количество программ в программе;

R_K - разрядность команды,

A - индекс, указывающий адресность команд программы.

При выборе количества адресов по критерию «емкость ЗУ» предпочтение следует отдавать одноадресным командам.

Адресность и время выполнения программы. Время выполнения одной команды складывается из времени выполнения операции и времени обращения к памяти.

Для трехадресной команды последнее суммируется из четырех составляющих времени:

- выборки команды;
- выборки первого операнда;
- выборки второго операнда;
- записи в память результата.

Одноадресная команда требует двух обращений к памяти:

- выборки команды;
- выборки операнда.

Как видно, на выполнение одноадресной команды затрачивается меньше времени, чем на обработку трехадресной команды, однако для реализации одной трехадресной команды, как правило, нужно три одноадресных. Этих соображений тем не менее не достаточно, чтобы однозначно отдать предпочтение тому или иному варианту адресности. Определяющим при выборе является тип алгоритмов, на преимущественную реализацию которых ориентирована конкретная ЭВМ.

Возможные типы алгоритмов условно разделим на три группы:

- последовательные;
- параллельные;
- комбинированные.

Двухадресные команды в плане времени реализации алгоритмов занимают промежуточное положение между одноадресными и трехадресными. Несколько лучшие показатели дают полутадресные команды, в которых, с одной стороны, сохраняются преимущества одноадресных команд для последовательных алгоритмов, а с другой - повышается эффективность реализации параллельных и комбинированных алгоритмов.

Любая микроЭВМ может делать только то, что ей предписывает человек. Программа для микроЭВМ - это последовательность команд, которые микроЭВМ распознает и в соответствии с этим выполняет определенные действия. Каждая команда инициирует выполнение определенного действия.

Каждая микроЭВМ способна воспринимать и выполнять точно определенный для нее набор команд. Количество и тип команд изменяются в зависимости от возможностей и назначения микроЭВМ.

Различают три основных уровня программирования:

- 1) программирование в машинных кодах;
- 2) программирование в кодах языка Ассемблера;
- 3) программирование на языках высокого уровня.

Программирование в машинных (двоичных) кодах - трудоемкая и сложная для человека задача. Будучи языком цифр, он неудобен для описания вычислительных процессов, требует от программиста больших усилий при написании и отладке программ.

К достоинствам данного вида программирования можно отнести следующее: для написания программ требуется лишь знания системы команд конкретной микроЭВМ, а для выполнения написанных таким образом программ микроЭВМ не нуждается ни в каком дополнительном программном обеспечении. Кроме того, при использовании машинного языка можно достигнуть максимальной гибкости в реализации технических возможностей микроЭВМ.

Для упрощения процесса написания, отладки и чтения программы используют мнемонический (символический) код. Каждую машинную команду представляют простым трех- или четырехбуквенным мнемоническим символом. Мнемонические символы значительно легче связать с машинными операциями, потому что их можно выбрать таким образом, чтобы они напоминали названия команд.

Программу, написанную в мнемонических кодах, необходимо транслировать в машинные коды. Сделать это можно двумя способами: вручную с помощью таблицы соответствия системы команд для конкретной микроЭВМ (прил. 3) или специальной программой, называемой Ассемблером. Ассемблер сравнивает каждую мнемоническую команду со списком команд и заменяет ее двоичным эквивалентом. Такой процесс получил название программной трансляции.

Язык Ассемблера, получивший свое название от имени программы, преобразующей программу на таком языке, в машинные коды, имеет ряд преимуществ. Он позволяет, наряду с программированием в машинных кодах, гибко и полно реализовать технические возможности микроЭВМ. В то же время, использование языка Ассемблера избавляет программиста от трудоемкой работы с машинными двоичными кодами.

Язык Ассемблера является машинно-зависимым (машинно-ориентированным) и, следовательно, отражает аппаратные особенности той микроЭВМ, для которой создан. Поэтому программы, составленные с использованием языка Ассемблера (или машинного языка), обладают наименьшей мобильностью по отношению к различным микроЭВМ.

Большой мобильностью обладают программы, составленные на алгоритмических языках высокого уровня (например Фортран, Бейсик, ПЛ/М, Паскаль, Ада, СИ и др.), занимающих верхнее положение в иерархии языков программирования. Будучи приближенными к привычной математической нотации и, в ряде случаев обеспечивая естественную форму описания вычислительных процессов, они достаточно просты и удобны в программировании, но не всегда позволяют в полной мере реализовать технические возможности ЭВМ. Кроме того, результирующие машинные программы, получаемые после трансляции программ с алгоритмических языков, обычно менее эффективны с точки зрения объема и быстродействия, по сравнению с программами, написанными в кодах языка Ассемблера и в машинных кодах. Применение языков высокого уровня предполагает обязательное наличие транслятора, представляющего собой сложный программный комплекс.

Выполнение команд микропроцессором

Микропроцессор КР580ИК80А имеет фиксированный набор команд. Работа микропроцессора по реализации каждой команды программы основана на принципе микропрограммного управления. То есть, каждая команда реализуется как некоторая последовательность микрокоманд (или микроопераций), приводящая к требуемому результату.

Считываемая из памяти микропроцессором команда, точнее ее восьмиразрядный двоичный код (код операции), поступает в регистр команд, где и хранится в течение времени ее выполнения. По результату дешифрирования кода операции происходит формирование последовательности микроопераций, процесс выполнения которой и определяет все последующие операции по выполнению считанной команды.

При этом выполнение отдельных микроопераций синхронизируется сигналами тактового генератора.

В микропроцессоре выполнение каждой команды можно разбить на ряд основных операций. Время, отведенное на выполнение операции обращения к памяти или устройству ввода-вывода, составляет машинный цикл. Следовательно, процесс выполнения команды состоит из стольких машинных

циклов, сколько обращений к памяти или к устройствам ввода-вывода требуется для ее выполнения. Каждая команда в зависимости от ее вида может занимать от одного до пяти машинных циклов.

В свою очередь, каждый машинный цикл может состоять из 3 - 5 машинных тактов. Под машинным тактом подразумевается интервал времени, соответствующий одному периоду тактовых импульсов.

На рисунке 9 приведена временная диаграмма команды LDA (Загрузить данные в аккумулятор из ячейки памяти, адрес которой находится во втором и третьем байте команды). Данная команда состоит из четырех машинных циклов: C1 - выборка команды (код команды помещается в регистр команд); C2 - обращение к памяти (восемь младших разрядов адреса выбираются из памяти и помещаются в регистр Z); C3 - обращение к памяти (восемь старших разрядов адреса выбираются из памяти и помещаются в регистр W); C4 - аккумулятор загружается данными из памяти по адресу, содержащемуся в регистровой паре WZ. В свою очередь первый машинный цикл C1 команды LDA состоит из четырех машинных тактов T1, T2, T3, T4, а машинные циклы C2, C3, C4 - из трех.

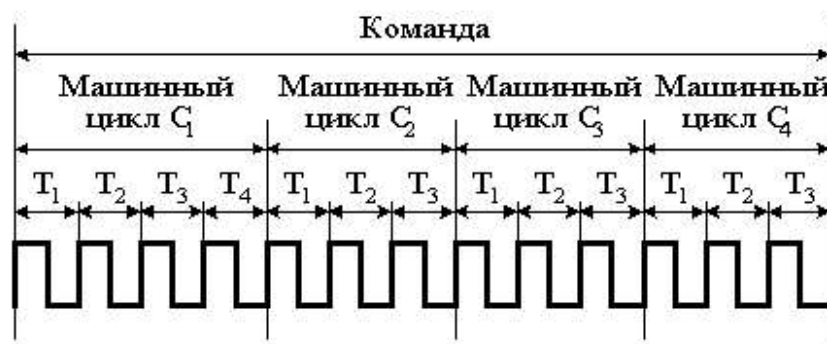


Рисунок 9 - Временная диаграмма команды LDA

Система команд микропроцессора KP580BM80A представлена 244 кодами операций.

Все команды можно классифицировать по трем основным признакам: по длине команды; по виду адресации к данным; по функциональному признаку.

По длине команды делятся на однобайтовые, двухбайтовые и трехбайтовые.

Двухбайтовые и трехбайтовые команды в памяти занимают соответственно две или три последовательно расположенные ячейки. При этом первый байт всегда отводится под код операции, а второй или второй и третий байты содержат либо данные, либо адрес, по которому они находятся в памяти.

По функциональному признаку все команды могут быть разделены на пять групп:

- 1) команды пересылки данных;
- 2) арифметические команды;
- 3) логические команды;
- 4) команды переходов;
- 5) команды управления и работы со стеком.

К группе команд пересылки данных относятся команды, пересылающие содержащиеся в них данные в регистры или в память, команды пересылки данных между регистрами микропроцессора и между регистрами и памятью.

В мнемонике команд регистр-приемник указывается вслед за символическим кодом команды, а через запятую - регистр-источник (прил. 2). Например: команда MOV M, A - пересылает данные из регистра A (аккумулятора) в ячейку памяти по адресу указанному в регистровой паре HL.

Арифметические команды предназначены для выполнения арифметических операций над данными, хранящимися в регистрах и ячейках памяти. Все команды этой группы оказывают влияние на значения разрядов флагового регистра, так как при их выполнении меняются знаки используемых чисел, возникают сигналы переноса, появляются нулевые результаты и т.п.

Логические команды служат для выполнения логических (или булевых) операций над данными, содержащимися в регистрах, ячейках памяти, а так же над флагами регистров. К ним относятся следующие операции: инверсия (НЕ), логическое суммирование (ИЛИ), логическое умножение (И), суммирование по модулю 2, сравнение, сдвиг, дополнение до 1 и до 2. Как и команды предыдущей группы, все логические команды оказывают влияние на флаги состояния.

Система команд микропроцессора KP580BM80A позволяет использовать следующие виды адресации:

- регистровая;
- косвенная регистровая;
- прямая;
- непосредственная;
- неявная.

Регистровая адресация.

В случае регистровой адресации операнд отыскивается во внутреннем регистре микропроцессора. Регистровые адресации всегда являются одно байтовыми, потому что они не требуют адресов и данных вне микропроцессора. Примером таких команд являются команды: MOV A,D (переслать содержимое регистра D в регистр A); MOV H,L (переслать содержимое регистра L в регистр H).

Косвенная регистровая адресация.

При этом способе адресации пара регистров HL указывает на адрес операнда в памяти. Примером являются такие команды, как MOV A,M (переслать содержимое ячейки памяти, адрес которой находится в паре HL, в аккумулятор); MOV M,D и т.п. При этом способе адресации могут также использоваться регистровые пары BC и DE. В этом случае будут

использоваться команды STAX YZ (сохранение содержимого аккумулятора в памяти, адрес которой находится в паре YZ) и LDAX YZ (загрузка в аккумулятор содержимым ячейки памяти, адрес которой находится в паре YZ). При использовании этих команд вместо YZ подставляется конкретная регистровая пара (если адрес в паре BC, то записывается B; если DE, то записывается D). На пример LDAX B, STAX D.

Прямая адресация.

В случае прямой адресации 2-ой и 3-ий байт команды прямо указывает на адрес операнда в памяти (т.о. адрес данных следует за кодом операции). Этим адресом может быть адрес памяти (LDA 22FFH, STA 50DDH и т.п) , а также номер порта ввода/вывода (IN E4H, OUT 7FH).

Непосредственная адресация.

В случае непосредственных команд операнд следует сразу же за кодом операции. Примером являются: MVI A,13H; LXI H,80AH; LXI D,7F00H и т.п.

Неявная адресация.

Этот метод используется, если не нужно искать данные или адреса в других регистрах микропроцессора, памяти или устройствах ввода/вывода. Все события происходят внутри микропроцессора.

Задача № 1

Записать в регистр A однобайтовое число 47H и проинвертировать его.

Программа оформляется в виде:

адреса	код	команды	Примечание
0800H	3E	MVI A,47H	Занесение байта данных в накопитель
0801H	47		
0802H	2F	CMA	Инвертируем содержимое накопителя
0803H	76	HLT	Конец программы

Перед вводом программы в память МП требуется перевести программу с языка ассемблер в машинные коды и назначить каждому коду, либо операнду (операндами называются данные используемые в командах программы) адрес ячейки памяти в ОЗУ МП-системы. Выбранные адреса должны соответствовать ОЗУ МП. В противном случае при загрузке программы в память МП можно потерять часть или все данные.

Для перевода программы с языка ассемблер в машинные коды требуется выполнить следующие действия:

1. для каждой команды, представленной в строке 3 таблицы 1, требуется подобрать машинный код, представляющий собой код операции , соответствующей выбранной команде (приложение 1);
2. определить операнды (данные, следующие за мнемокодом операции);
3. определить соответствие последовательных адресов памяти для каждой команды и операнда.

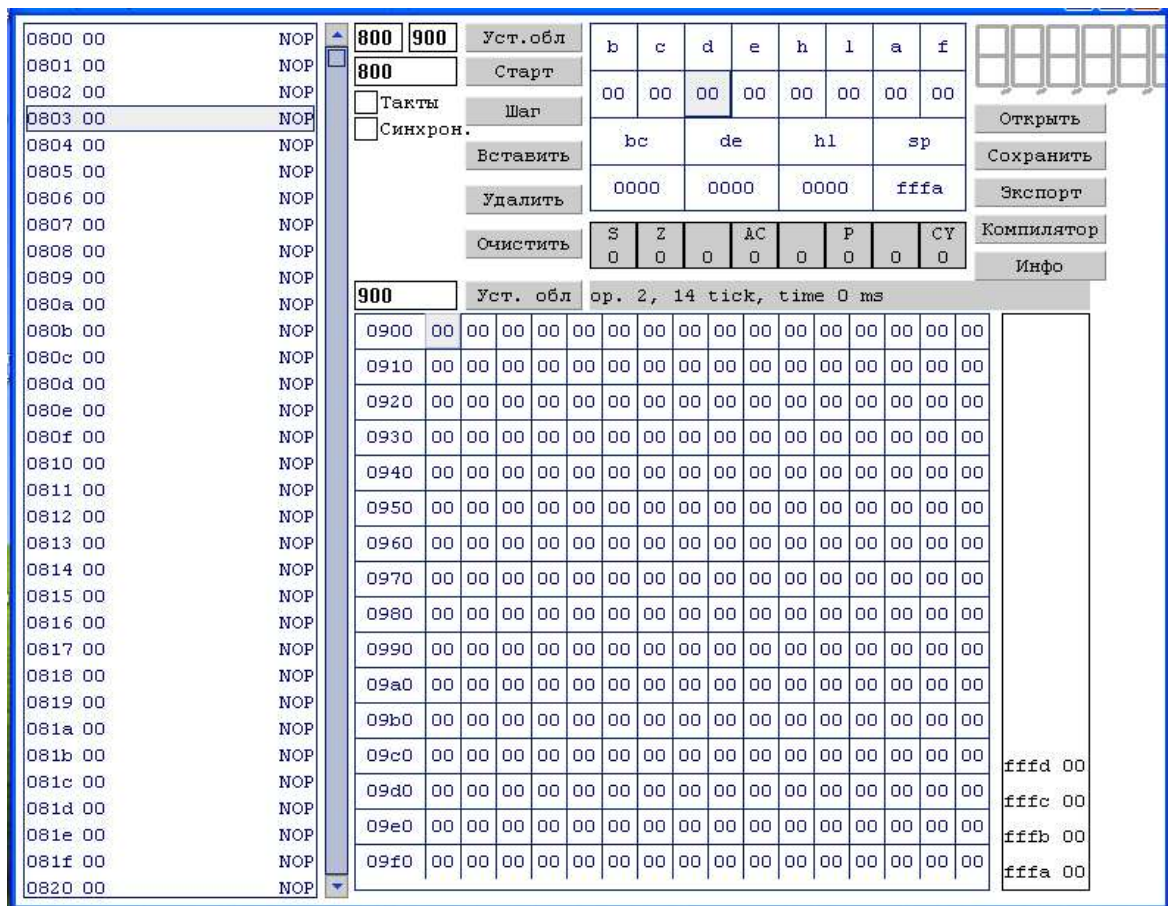
Каждая команда, используемая в МП КР580, имеет определенный формат. В соответствии с этим операции делятся на однобайтные, двухбайтные и трехбайтные. Однобайтные операции не содержат дополнительных операндов. Двухбайтные содержат один байт операнд, трехбайтные имеют двухбайтные операнды. Следует учесть, что *при записи трехбайтной команды в ОЗУ МП-системы, сначала следует код, затем младший бай и только потом старший бай данных.*

Для подготовки УМК к работе требуется выполнить следующие действия:

1. На рабочем столе выбрать иконку



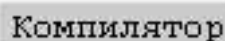
и открыть её.



2. Выбираем установочную область памяти

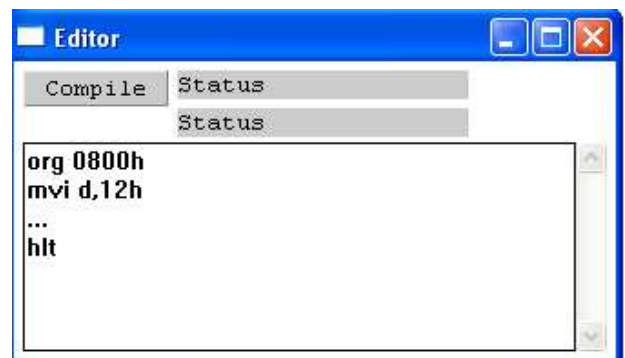


3. Включаем компилятор

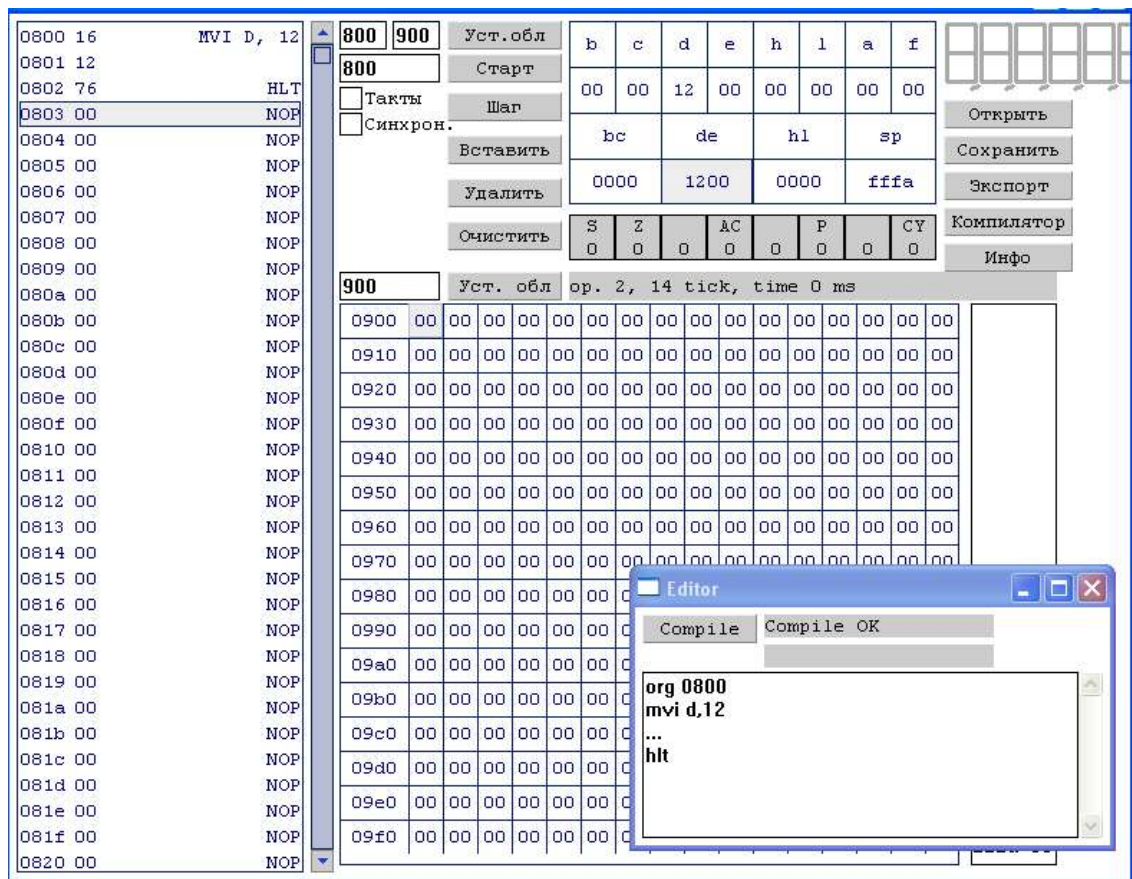


Компилятор — транслятор, который осуществляет перевод всей исходной программы в эквивалентную ей результирующую программу на языке машинных команд (микропроцессора или виртуальной машины).

4. Осуществляем набор команд в компиляторе (всегда начинаем работу с **org 0800**, заканчиваем **hlt**). После ввода программы нажимаем на иконку **Compile**, если все верно, то высвечивается иконка **Compile Ok**.



5. Осуществить **Старт** с заявленного адреса (т.е. начало программы).
6. Просмотр выполнения программы (Регистры, адрес).



Задания к 1 лабораторной работе

Результат выполнения программ представить в виде таблицы 5.1.

Число по адресу 0900H	Число по адресу 0901H	Число по адресу 0902H	Содержимое А	Состояние битов регистра F S= Z= AC= P= C=

1. Разработать программу которая вычитает однобайтовые числа которые расположены в регистрах В и А, результат помещает в регистр С. (Числа заносятся программно).
2. Поменять содержимое регистровых пар DE и HL местами. Числа в регистры заносим программно.

3. Разработать программу которая записывает в регистр С 15Н, в регистр В 23 Н и производит сложение, результат помещает в регистр В.
4. Сложить содержимое ячейки памяти, с адресом 0900Н и содержимое ячейки памяти, с адресом 0901Н. Результат поместить в регистр В.
5. Уменьшить содержимое регистровой пары DE на содержимое регистровой пары HL. Числа в регистровые пары заносим программно. результат поместить в регистровую пару DE.
6. Поменять регистровые пары HL и BC местами. Числа в регистровые пары заносим программно.
7. Написать программу, увеличивающую содержимое ячейки памяти по адресу 0900Н на 5Н и размещающую результат в ячейку памяти по адресу 0901.
8. Написать программу сложения двух двухбайтовых чисел, одно из которых расположено в памяти, начиная с адреса 0900Н, другое с адреса 0902Н. Результат разместить в памяти с адреса 0904Н. Перед выполнением программы записать по исходным адресам двухбайтовые числа, указанные преподавателем.
9. Заменить в прог.4. команду сложения содержимого аккумулятора с регистром В (ADD B) командой сравнения CMP B. Ввести программу в память УМК. Записать в ячейку памяти по адресу 0900Н число большее, чем по адресу 0901Н. Запустить программу на выполнение. После выполнения проанализировать результат выполнения .
10. Записать по адресу 0900Н число меньшее, чем по адресу 0901Н. (п.4)
11. 10. Записать по адресам 0900Н и 0901Н два одинаковые числа. Запустить программу на выполнение. Исследовать результат выполнения (п.4).
12. Заменить в программе 4.3. команду ADD B на команды INR A, DCR A, ADD A, ANA A, ORA A, XRA A. Исследовать результат выполнения, содержимое аккумулятора и флагового регистра F. Результат представить в виде таблицы 5.2.

Таблица 5.2.

Команда	Содержимое аккумулятора	Содержимое аккумулятора после операции	Содержание битов флагового регистра F

Вопросы для самопроверки

1. Из чего состоит память микропроцессора?
2. Нарисуйте структуру учебной микро-ЭВМ.
3. Что происходит при попытке записи данных в ПЗУ?
4. Шина (адреса, данных, управления) является однонаправленной.
5. Посредством 16 линий адресной шины можно получить доступ к (кол-во) ячейкам памяти и (кол-во) устройствам ввода-вывода.
6. Нарисуйте внутреннюю структуру микропроцессора и укажите основное назначение его компонентов.
7. Какие биты входят в состав регистра признаков микропроцессора?
8. Какие команды арифметических и логических операций выполняет микропроцессор КР580ВМ80А?
9. Какие методы адресации используются в микропроцессоре?
10. Как влияют арифметические и логические операции на биты флагового регистра?

Список литературы

1. Юров В.Б. Assembler [Текст] : учебник студентам вузов, изучающим архитектуру микропроцессоров Intel в рамках соответствующих дисциплин / - СПб.: М.: Харьков :Минск : Питер, 2001. - 624 с.
2. Складов В.А. Программирование на языке Ассемблера [Текст] : учебное пособие / Складов В.А. - М. : Высшая школа, 1999. - 151,[1]с.
3. Пузанкова Д.В. Микропроцессорные системы [Текст] : учебное пособие для студентов вузов, обучающихся по направлению подготовки бакалавров и магистров "Информатика и вычислительная техника"/-СПб.: Политехника, 2002. - 936 с. : ил. - (Учебное пособие для вузов).
4. Пузанкова Д.В. Сквозные образовательные программы высшего и среднего профессионального технического образования по направлениям: "Радиотехника", "Автоматизация и управление" [Текст] : методический материал / - М. : Новый учебник, 2004. - 73 с.

Приложение 1

Команды ассемблера микропроцессора КР580ИК80А

Команды пересылки

Команда	Описание	Код	Длина
Mov r1,r2	Пересылка данных из регистра r2 в регистр r1		1
Mov M, r	Пересылка данных из регистра r в память		1
Mov r, M	Пересылка данных из памяти в регистр r		1
XCHG	Обмен данными между парами регистров HL и DE	EB	1
MVI r ()	Занесение байта данных в регистр r		2
MVI M ()	Занесение байта данных в память	36	2
LDA (адрес)	Загрузка содержимого ячейки с указанным адресом в накопитель	3A	3
LHLD (адрес)	Загрузка в регистры H, L содержимого ячеек с указанным адресом и адресом на единицу большим	2A	3
STAX rp	Занесение содержимого накопителя в ячейку, косвенно адресуемую парой регистров rp (B,D)		1
STA (адрес)	Занесение содержимого накопителя в ячейку с указанным адресом	32	3
SHLD (адрес)	Занесение содержимого регистра HL в память с указанным адресом и адресом на единицу большим	22	3
LXI rp ()	Занесение двух байтов данных в пару регистров (B,D,H,SP)		3
LDAX rp	Загрузка в накопитель содержимого ячейки, косвенно адресуемую парой регистров rp (B,D)		1

Вычислительные команды

ADD	Сложение содержимого регистра r и накопителя		1
ADD M	Сложение содержимого ячейки памяти и накопителя	86	1
ADC r	Сложение содержимого регистра r и накопителя с учётом переноса C		1
ADC M	Сложение содержимого ячейки памяти и накопителя с учётом переноса C	8E	1
ADI ()	Сложение байта с содержимым накопителя	C6	2
ACI ()	Сложение байта с содержимым накопителя с учетом переноса	CE	2
SUI ()	Вычитание байта из содержимого накопителя	D6	2
SUB r	Вычитание содержимого регистра r из содержимого накопителя		1
SBI ()	Вычитание байта из содержимого накопителя с учетом заёма	DE	2
SUB M	Вычитание содержимого памяти из содержимого накопителя	96	1
SBB r	Вычитание содержимого регистра r из содержимого накопителя с заёмом		1
SBB M	Вычитание содержимого памяти из содержимого накопителя с заёмом	9E	1

Логические команды

ANA r	Подразрядное И над содержимым регистра r и накопителя		1
ANA M	Подразрядное И над содержимым памяти и накопителя	A6	1
XRA r	Подразрядное ИСКЛЮЧАЮЩЕЕ ИЛИ над содержимым регистра r и накопителя		1
XRA M	Подразрядное ИСКЛЮЧАЮЩЕЕ ИЛИ над содержимым памяти и накопителя	AE	1
ORA r	Подразрядное ИЛИ над содержимым регистра r и накопителя		1
ORA M	Подразрядное ИЛИ над содержимым памяти и накопителя	B6	1
CMP r	Сравнение содержимых регистра r и накопителя		1
CMP M	Сравнение содержимых памяти и накопителя	BE	1
ANI ()	Подразрядное И над содержимым накопителя и байтом	E6	2
XRI ()	Подразрядное ИСКЛЮЧАЮЩЕЕ ИЛИ над содержимым накопителя и байтом	EE	2
ORI ()	Подразрядное ИЛИ над содержимым накопителя и байтом	F6	2
CPI ()	Сравнение байта с содержимым накопителя	FE	2
DAD rp	Сложение содержимого пары регистров rp (B,D,H,SP) с содержимым пары регистров H,L		1
INR r	Увеличение содержимого регистра r на единицу		1
DCR r	Уменьшение содержимого регистра r на единицу		1
DCR M	Уменьшение содержимого памяти на единицу	35	1
INR M	Увеличение содержимого памяти на единицу		1
INX rp	Увеличение содержимого пары регистров rp (B,D,H,SP) на единицу		1
DCX rp	Уменьшение содержимого пары регистров rp (B,D,H,SP) на единицу		1

Команды сдвига

RLC	Циклический сдвиг содержимого накопителя влево	7	1
RRC	Циклический сдвиг содержимого накопителя вправо	0F	1
RAL	Циклический сдвиг содержимого накопителя влево через перенос	17	1
RAR	Циклический сдвиг содержимого накопителя вправо через перенос	1F	1
DAA	Преобразование содержимого накопителя в двоично-десятичный код	27	1
CMA	Поразрядное инвертирование накопителя	2F	1

STC	Установка признака переноса в единицу	37	1
CMC	Инвертирование признака переноса	3F	1
PCHL	Занесение содержимого регистров H,L в счетчик команд	E9	1

Условный и безусловный переход

JMP (адрес)	Безусловный переход по указанному адресу	C3	3
JC (адрес)	Переход при наличии переноса	DA	3
JNC (адрес)	Переход при отсутствии переноса	D2	3
JZ (адрес)	Переход при нуле	CA	3
JNZ (адрес)	Переход при отсутствии нуля	C2	3
JP (адрес)	Переход при плюсе	F2	3
JM (адрес)	Переход при минусе	FA	3
JPE (адрес)	Переход при чётности	EA	3
JPO (адрес)	Переход при нечётности	E2	3
CALL (адрес)	Вызов подпрограммы	CD	3
CC (адрес)	Вызов подпрограммы при переносе	DC	3
CNC (адрес)	Вызов подпрограммы при отсутствии переноса	D4	3
CZ (адрес)	Вызов подпрограммы при нуле	CC	3
CNZ (адрес)	Вызов подпрограммы при отсутствии нуля	C4	3
CP (адрес)	Вызов подпрограммы при плюсе	F4	3
CM (адрес)	Вызов подпрограммы при минусе	FC	3
CPE (адрес)	Вызов подпрограммы при чётности	EC	3
CPO (адрес)	Вызов подпрограммы при нечётности	E4	3
RET	Возврат	C9	1
RC	Возврат при переносе	D8	1
RNC	Возврат при отсутствии переноса	D0	1
RZ	Возврат при нуле	C8	1
RNZ	Возврат при отсутствии нуля	C0	1
RP	Возврат при плюсе	F0	1
RM	Возврат при минусе	F8	1
RPE	Возврат при чётности	E8	1
RPO	Возврат при нечётности	E0	1
RST (номер)	Повторный запуск с адреса	CF	1

Команды стёка, ввода, вывода и управления

IN(канал)	Ввод данных из накопителя в указанный канал	D8	2
OUT(канал)	Вывод данных из накопителя в указанный канал	D3	2
PUSH гр	Занесение содержимого пары регистров гр (B,D,H,PSW) в стёк		1
POP гр	Выдача данных из стёка в пару регистров гр (B,D,H,PSW) в стёк		1
XTHL	Обмен данными между вершиной стёка и парой регистров H,L	E3	1
SPHL	Занести в указатель стёка содержимое регистров H,L	F9	1
DI	Запретить прерывание	F3	1
EI	Разрешить прерывание	FB	1
NOP	Отсутствие операции	0	1
HLT	Остановка	76	1

Приложение 2

КОМАНДЫ АССЕМБЛЕРА МИКРОПРОЦЕССОРА КЗ580ИК80А

Мнемокод <i>Ком/пересылка</i>	Код	Мнемокод	Код	Мнемокод	Код	Мнемокод	Код	Мнемокод	Код
MOV A,A	7F	MOV H,A	67	LXI D,&	11	DAD B	09	INR D	14
MOV A,B	78	MOV H,B	60	LXI H,&	21	DAD D	19	INR E	1C
MOV A,C	79	MOV H,C	61	LXI SP,&	31	DAD H	29	INR H	24
MOV A,D	7A	MOV H,D	62	LDAX B	0A	DAD SP	39	INR L	2C
MOV A,E	7B	MOV H,E	63	LDAX D	1A	<i>Логические операции</i>			3D
MOV A,H	7C	MOV H,H	64	STAX B	0A	ANA A	A7	DCR B	05
MOV A,L	7D	MOV H,L	65	STAX D	12	ANA B	A0	DCR C	0D
MOV B,A	47	MOV L,A	6F	<i>Арифм/операции</i>			A1	DCR D	15
MOV B,B	40	MOV L,B	68	ADD A	87	ANA D	A2	DCR E	1D
MOV B,C	41	MOV L,C	69	ADD B	80	ANA E	A3	DCR H	25
MOV B,D	42	MOV L,D	6A	ADD C	81	ANA H	A4	DCR L	2D
MOV B,E	43	MOV L,E	6B	ADD D	82	ANA L	A5	INX B	03
MOV B,H	44	MOV L,H	6C	ADDE	83	XRA A	AF	INX D	13
MOV B,L	45	MOV L,L	6D	ADD H	84	XRA B	A8	INX H	23
MOV C,A	4F	MOV M,A	77	ADD L	85	XRA C	A9	INX SP	33
MOV C,B	48	MOV M,B	70	ADC A	8F	XRA D	AA	DCX B	0B
MOV C,C	49	MOV M,C	71	ADC B	88	XRA E	AB	DCX D	1B
MOV C,D	4A	MOV M,D	72	ADC C	89	XRA H	AC	DCX H	2B
MOV C,E	4B	MOV M,E	73	ADC D	8A	XRA L	AD	DCX SP	3B
MOV C,H	4C	MOV M,H	74	ADC E	8B	ORA A	B7	PUSH B	C5
MOV C,L	4D	MOV M,L	75	ADC H	8C	ORA B	B0	PUSH D	D5
MOV D,A	57	MOV B,M	46	ADC L	8D	ORA C	B1	PUSH H	E5
MOV D,B	50	MOV C,M	4E	SUB A	97	ORA D	B2	PUSH PSW	F5
MOV D,C	51	MOV D,M	56	SUB B	90	ORA E	B3	POP B	C1
MOV D,D	52	MOV E,M	5E	SUB C	91	ORA H	B4	POP D	D1
MOV D,E	53	MOV H,M	66	SUB D	92	ORA L	B5	POP H	E1
MOV D,H	54	MOV L,M	6E	SUB E	93	CMP A	BF	POP PSW	F1
MOV D,L	55	MVI A,#	3E	SUB H	94	CMP B	B8		
MOV E,A	5F	MVI B,#	06	SUB L	95	CMP C	B9		
MOV E,B	58	MVI C,#	0E	SBB A	9F	CMP D	BA		
MOV E,C	59	MVI D,#	16	SBB B	98	CMP E	BB		
MOV E,D	5A	MVI E,#	1E	SBB C	99	CMP H	BC		
MOV E,E	5B	MVI H,#	26	SBB D	9A	CMP L	BD		
MOV E,H	5C	MVI L,#	2E	SBB E	9B	INR A	3C		
MOV E,L	5D	LXI B,&	10	SBB H	9C	INR B	04		
				SBB L	9D	INR C	0C		

Арифметические команды микропроцессоров Intel 8080/8085.

Арифметические команды предназначены для выполнения операций сложения, сложения с переносом, вычитания, вычитания с заемом, инкрементирования, декрементирования, десятичной коррекции аккумулятора. Результат всех арифметических операций остается в аккумуляторе.

ADD r (Add register). Сложение содержимого регистра $(A) \leftarrow (A) + (r)$. Содержимое регистра r складывается с содержимым аккумулятора. Результат помещается в аккумулятор.

Циклов – 1; периодов $T - 4$; адресация – регистровая; индикаторы – Z, S, P, CY, AC .

ADD M (Add memory). Сложение данных памяти $(A) \leftarrow (A) + M(HL)$. Содержимое ячейки памяти, адрес которой содержится в регистрах H и L , складывается с содержимым аккумулятора. Результат помещается в аккумулятор.

Циклов – 2; периодов $T - 7$; адресация – косвенная регистровая; индикаторы – Z, S, P, CY, AC .

ADI data 8 (Add immediate). Непосредственное сложение $(A) \leftarrow (A) + \text{data}$. Содержимое байта 2 команды складывается с содержимым аккумулятора. Результат помещается в аккумулятор.

Циклов - 2; периодов $T - 7$; адресация - непосредственная.

ADC r (Add register with carry). Прибавление содержимого регистра и переноса $(A) \leftarrow (A) + (r) + (CY)$. Содержимое регистра r и индикатора переноса (бит переполнения) складывается с содержимым аккумулятора. Результат помещается в аккумулятор.

Циклов – 1; периодов $T - 4$; адресация – регистровая; индикаторы - Z, S, P, CY, AC .

ADC M (Add memory with carry). Прибавление содержимого памяти и переноса $(A) \leftarrow (A) + M(HL) + (CY)$. Содержимое ячейки памяти, адресом которой является содержимое пары регистров HL , и индикатора переноса складывается с содержимым аккумулятора. Результат помещается в аккумулятор.

Циклов – 2; периодов $T - 7$; адресация – косвенная регистровая; индикаторы - Z, S, P, CY, AC .

ACI data 8 (Add immediate with carry). Непосредственное сложение с учетом переноса $(A) \leftarrow (A) + \text{data 8} + (CY)$. Содержимое байта 2 команды и индикатора переноса складывается с содержимым аккумулятора. Результат помещается в аккумулятор.

Циклов – 2; периодов $T - 7$; адресация – непосредственная; индикаторы - Z, S, P, CY, AC .

SUB r (Subtract register). Вычитание содержимого регистра $(A) \leftarrow (A) - (r)$. Содержимое регистра r вычитается из содержимого аккумулятора. Результат помещается в аккумулятор.

Циклов – 1; периодов $T - 5$; адресация – регистровая; индикаторы - Z, S, P, CY, AC .

SUB M (Subtract memory). Вычитание содержимого памяти $(A) \leftarrow (A) - M(HL)$. Содержимое ячейки памяти, адрес которой является содержимым пары HL , вычитается из содержимого аккумулятора. Результат помещается в аккумулятор.

Циклов – 2; периодов $T - 7$; адресация – косвенная регистровая; индикаторы - Z, S, P, CY, AC .

SUI data 8 (Subtract immediate). Непосредственное вычитание данных $(A) \leftarrow (A) - \text{data 8}$. Содержимое байта 2 команды вычитается из содержимого аккумулятора. Результат помещается в аккумулятор.

Циклов – 2; периодов $T - 7$; адресация – непосредственная; индикаторы - Z, S, P, CY, AC .

SBB r (Subtract register with borrow). Вычитание содержимого регистра и переноса $(A) \leftarrow (A) - (r) - (CY)$. Содержимое регистра r и индикатора переноса CY вычитается из содержимого аккумулятора. Результат помещается в аккумулятор.

Циклов – 1; периодов $T - 5$; адресация – регистровая; индикаторы - Z, S, P, CY, AC .

SBB M (Subtract memory with borrow). Вычитание содержимого памяти и переноса $(A) \leftarrow (A) - M(HL) - (CY)$. Содержимое ячейки памяти, адрес которой является содержимым пары HL , и индикатора переноса CY вычитается из содержимого аккумулятора. Результат помещается в аккумулятор.

Циклов – 2; периодов $T - 7$; адресация – косвенная регистровая; индикаторы - Z, S, P, CY, AC .

SBI data 8 (Subtract immediate with borrow). Непосредственное вычитание данных и переноса $(A) \leftarrow (A) - \text{data 8} - (CY)$. Содержимое байта 2 команды и индикатора переноса CY вычитается из содержимого аккумулятора. Результат помещается в аккумулятор.

Циклов – 2; периодов $T - 7$; адресация – непосредственная; индикаторы - Z, S, P, CY, AC .

INR r (Increment register). Инкремент содержимого регистра $(r) \leftarrow (r) + 1$. Содержимое регистра r увеличивается на 1. Устанавливаются все индикаторы состояния, за исключением CY .

Циклов – 1; периодов $T - 5$; адресация – регистровая; индикаторы - Z, S, P, AC .

INR M (Increment memory). Инкремент содержимого памяти $M(HL) \leftarrow M(HL) + 1$. Содержимое ячейки памяти, адрес которой содержится в паре HL , увеличивается на 1. Устанавливаются все индикаторы состояния, за исключением CY .

Циклов – 3; периодов $T = 10$; адресация – косвенная регистровая; индикаторы - Z, S, P, AC.

DCR r (Decrement register). Декремент содержимого регистра (r) $\leftarrow (r) - 1$. Содержимое регистра r уменьшается на 1. Устанавливаются все индикаторы состояния, за исключением CY.

Циклов – 1; периодов $T = 5$; адресация – регистровая; индикаторы - Z, S, P, AC.

DCR M (Decrement memory). Декремент содержимого памяти $M(HL) \leftarrow M(HL) - 1$. Содержимое ячейки памяти, адрес которой содержится в паре HL, уменьшается на 1. Устанавливаются все индикаторы состояния, за исключением CY.

Циклов – 3; периодов $T = 10$; адресация – косвенная регистровая; индикаторы - Z, S, P, AC.

INX rp (Increment register pair). Инкремент содержимого пары регистров (rp) $\leftarrow (rp) + 1$. Содержимое пары регистров rp увеличивается на 1. Не устанавливаются никакие индикаторы состояния.

Циклов – 1; периодов $T = 5$; адресация – регистровая; индикаторы не изменяются.

DCX rp (Decrement register pair). Декремент содержимого пары регистров (rp) $\leftarrow (rp) - 1$. Содержимое пары регистров rp уменьшается на 1. Не устанавливаются никакие индикаторы состояния.

Циклов – 1; периодов $T = 5$; адресация – регистровая; индикаторы не изменяются.

DAD rp (Add register pair to hand L). Сложить содержимое пары регистров с содержимым пары HL $(HL) \leftarrow (HL) + (rp)$. Содержимое пары регистров rp складывается с содержимым пары HL. Устанавливается только индикатор CY. Он устанавливается в 1, если есть перенос при сложении с удвоенной точностью, если нет – сбрасывается в 0.

Циклов – 3; периодов $T = 10$; адресация – регистровая; индикаторы - CY.

DAA (Decimal adjust accumulator). Десятичный корень аккумулятора. 8-ми разрядное число в аккумуляторе разбивается на два 4-х разрядных двоично-десятичных. Далее выполняются следующие действия:

- 1) если значение младшей тетрады больше 9 или устанавливается индикатор AC, то к содержимому аккумулятора добавляется 6;
- 2) если значение старшей тетрады аккумулятора больше 9 или устанавливается индикатор переноса CY, то 6 добавляется к значению старшей тетрады аккумулятора.

Циклов – 1; периодов $T = 5$; индикаторы - Z, S, P, CY, AC.