Санкт-Петербургский политехнический университет Петра Великого

Институт Компьютерных Наук и Технологий

Кафедра «Распределенные вычисления и компьютерные сети»

**Отчет по лабораторной работе**

Исследование однокристального микропроцессора с фиксированным списком команд

Дисциплина «Архитектура вычислительных систем»

Выполнил

студент группы № 43507/1 ФИО Борисов В.Б.

Преподаватель ФИО Кузьмин А.А .

Санкт-Петербург

2016

**1. Цель исследования:**

Цель данного цикла работ - ознакомление с организацией МП-систем на основе однокристальных МП 8080 (КР580ВМ80А) и приобретение навыков программирования микропроцессоров указанного типа.

**2. Структурные схемы и краткое описание работы МП и исследуемой микропроцессорной системы.**

КР580ВМ80А – восьмиразрядный однокристальный микропроцессор с фиксированной системой команд без возможности аппаратного наращивания разрядности. Структурная схема МП содержит следующие блоки: блок АЛУ, блок регистров общего назначения (РОН) со схемой выбора регистра и выходным мультиплексором, блок синхронизации и управления (БСУ), буферы адресов (БА) и данных (БД).

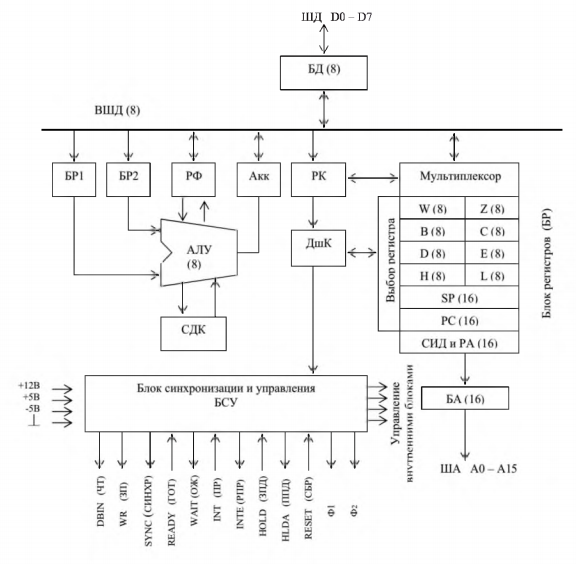


Рис. 1: Структурная схема МП КР580ВМ80А

Блок АЛУ обеспечивает выполнение арифметических и поразрядных логических операций, а также операций циклического сдвига. В состав этого блока входят 8-разядное АЛУ, регистр результата — аккумулятор (А), 8-разрядные буферные регистры (БР1 и БР2), регистр признаков (флагов) РФ и схема двоично-десятичной коррекции (СДК). В выполнении арифметических и поразрядных логических операций совместно с АЛУ принимают участие аккумулятор и буферные регистры. Признаки результата указанных операций фиксируются в регистре признаков, содержащем флаги нуля Z, знака S, переноса, паритета Р и вспомогательного (межтетрадного) переноса АС. Наряду с операциями над 8-разрядными двоичными числами, МП допускает выполнение арифметической операции сложения над операндами в формате двоично-десятичных чисел.

Блок регистров предназначен для приема, хранения и выдачи различной информации, используемой в выполнении команд. В состав этого блока входят шесть 16-битовых регистра: три пары 8-разрядных регистров общего назначения, счетчик команд PC, указатель стека SP и регистр временного хранения W-Z. Для выполнения операций инкремента/декремента содержимого регистров блок регистров дополнен схемой инкремента/декремента СИД. Шесть регистров общего назначения В, С, D, Е, 27 H, L, наряду с их непосредственным использованием в 8-разрядных операциях, могут объединяться в регистровые пары ВС, DE, HL. В командах регистровые пары обозначаются по имени старшего регистра в паре В, D, Н. Они могут хранить 16-битные операнды или использоваться в качестве указателей памяти. 16-разрядный регистр РА предназначен для запоминания адреса операнда при обращении к памяти на время машинного цикла. Выход регистр РА соединен с буферным регистром адреса БА. Счетчик команд PC хранит адрес текущей ячейки программной памяти. После выбора очередного байта любой команды содержимое PC увеличивается на единицу. Указатель стека SP адресует вершину стека. В микропроцессорных системах с МП КР580ВМ80А стек моделируется в оперативной памяти. В этих МП содержимое SP увеличивается при выборке данных из стека и уменьшается при загрузке данных в стек, при этом обмен данными между МП и стеком осуществляется 16-разрядными словами путем последовательной передачи по шине старшего и младшего байтов слова. Программно недоступный регистр W-Z используется для временного хранения второго и третьего байтов многобайтных команд.

Блок управления МП содержит регистр команд (РК), дешифратор команд (ДшК) и схемы синхронизации и управления (БСУ). С помощью этого блока обеспечивается формирование сигналов, настраивающих операционный блок на выполнение операций, определяемых кодом команды, и сигналов, осуществляющих внешние обмены между МП и внешними устройствами.

Знание особенностей структуры микропроцессора, назначения его выводов, электрических и конструктивных параметров необходимо при разработке аппаратной части микропроцессорной системы. В разработанной и изготовленной системе для практического использования МП (составления прикладных программ) достаточно знать его программную модель и систему команд.

В состав программной модели системы на базе МП КР580ВМ80А входят: 8- битные регистры блока РОН (В, С, D, Е, Н, L, А), 16-битовые регистровые пары (В, D, Н), указатель стека SP, 256 портов ввода и 256 портов вывода, ячейки памяти общим числом до 64 К. При обращении к памяти используется прямая и косвенная адресации. Основным указателем памяти при косвенной адресации является регистровая пара HL. Ячейка памяти, адрес которой определяется содержимым пары HL, обозначается М (от memory — память). Указателями памяти также могут выступать регистровые пары ВС, DE и указатель стека SP.

Система команд МП состоит из 78 базовых команд и 111 кодов операций. одно-, двух- и трехбайтовым. Код операции любой команды размещается в первом байте, а второй и третий байты, если они являются частью команды, могут содержать адрес операнда в памяти или сам операнд. В качестве операндов могут использоваться байты и 16-битные слова, при этом большинство операндов команд являются байтами. Для адресации используют следующие типы адресации: прямую, регистровую, непосредственную, неявную и косвенную.

Прямая адресация, при которой в коде команды указывается 16-битный адрес, используется в командах пересылок для задания адреса операнда в памяти. Для обозначения прямого адреса операнда в памяти в мнемониках команд применяется аббревиатура addr.

Регистровая адресация применяется для адресации регистров блока РОН (8-разрядных регистров В, С, D, Е, Н, L, А и 16-разрядных регистровых пар В, D, Н).

Операндом команд с непосредственной адресацией может быть только источник непосредственных данных. Непосредственные данные - это 8- или 16-битные константы или прямые адреса, для представления которых используется второй или второй и третий байт команды. МП КР580ВМ80А выполнен по схеме одноадресного вычислителя. В нем один из операндов размещается в аккумуляторе, и результат помещается в аккумулятор. При выполнении команд арифметических и поразрядных логических операций аккумулятор адресуется неявно. Неявная адресация используется и в некоторых других командах, например, в командах работы со стеком и в командах загрузки/запоминания содержимого аккумулятора из памяти/в память.

Косвенную адресацию применяют для обращения к операндам в памяти. Выше отмечено, что указателями адреса при косвенной адресации могут выступать регистровые пары ВС, DE, HL и указатель стека SP.

Систему команд МП КР580ВМ80А по функциональному признаку удобно подразделить на нескольких групп:

• команды пересылок;

• команды арифметических и логических операций;

• команды передачи управления.

В группу команд пересылок включены команды с мнемониками *MOV* (собственно пересылки), *PUSH, POP* (загрузки в стек и извлечения из стека), команды ввода-вывода *IN, OUT* и некоторые другие, в том числе команды обмена, загрузки и запоминания содержимого регистровых пар. Команды пересылок не модифицируют флаги результата.

МП КР580ВМ80А, как отмечено выше, выполнен по схеме одноадресного вычислителя. При выполнении арифметических и поразрядных логических операций один из операндов команд этой группы всегда размещается в аккумуляторе, и результат операции помещается в аккумулятор. В качестве второго операнда может использоваться содержимое любого регистра блока РОН или ячейки памяти М, адресуемой косвенно по адресу в регистровой паре HL. По результату операции модифицируются флаги.

Команды передачи управления предназначены для изменения естественного порядка выполнения команд программы при реализации разветвляющихся и циклических алгоритмов, вызовов подпрограмм и возврата из них. В системе команд МП КР580ВМ80А содержится сравнительно большое число команд передачи управления, которые подразделяются на безусловные и условные переходы. Команды безусловного перехода (*JMP*), вызова подпрограмм *(CALL)* и возврата из них *(RET)* передают управление по адресу, указываемому в команде *(JMP, CALL)* или по адресу, выбираемому из стека *(RET).*

Лабораторные работы выполняются на лабораторной установке DiLaB 8080, реализованной на базе универсального стенда DiLaB с встроенной моделью ядра микропроцессорной системы с МП 8080. Внутренняя организация лабораторного стенда соответствует типовой структуре микропроцессорной системы, выполненной на СБИС микропроцессорного комплекта К580. Моделируемое ядро системы объединяет: модель блока центрального процессора, модель генератора двухфазных тактовых последовательностей импульсов Ф1 и Ф2, модель системного контроллера сигналов управления обменом информации, модель блока шинных формирователей адреса и данных, модель порта ввода-вывода параллельной информации и блок оперативного запоминающего устройства.

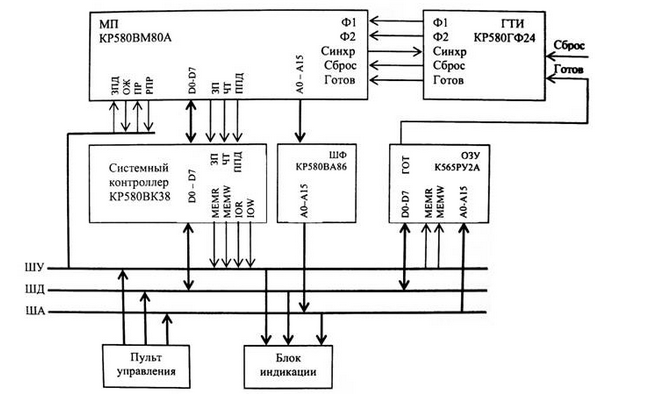


Рис. 2: Функциональная схема лабораторного стенда DiLaB 8080-2 c МП КР580ВМ80А

Основным подрежимом выполнения программы является режим пошагового выполнения команд по машинным циклам (с остановкой после каждого машинного цикла) - **режим WORK** S **М** (S - step, М - машинный цикл). В этом режиме запуск на выполнение машинного цикла осуществляется при нажатии клавиши RIGHT (ПУСК) блока PUSH BUTTONS.

**Режим WORK S С** (S - step, С - command) является режимом пошагового выполнения команд по командным циклам (с остановкой после каждого командного цикла). Перейти в этот режим можно только из режима WORK S М при нажатии кнопки ≪1≫ клавиатуры KEYPAD. В режиме WORKS С при нажатии клавиши RIGHT (ПУСК) блока PUSH BUTTONS выполняется очередная команда программы.

**Режим WORK С** (С - continues) - режим автоматического (непрерывного) выполнения команд программы. Режим WORK С устанавливается при нажатии кнопки ≪С≫ клавиатуры KEYPAD. В этом режиме программа выполняется полностью, (если это не циклическая программа).

Специальным подрежимом работы стенда DILAB 8080 является режим вывода параллельной информации OUT (ВЫВОД). В режиме OUT для индикации результатов выполнения программы используется блок семисегментных индикаторов LED MONITOR, который может быть реализован в виде блоков статической или динамической индикации.

Программирование пользовательских задач на учебных стендах осуществляется методом «ручного» программирования – программирования в машинных кодах. При программировании в машинных кодах пользователь максимально приближен к аппаратным средствам. Он должен хорошо знать структуру процессора, его систему команд и другие особенности архитектуры. «Ручное» программирование в наибольшей степени подходит для целей обучения основам организации ВМ, принципам построения и функционирования.

**3. Выполнение тестовых программ**

***Программа 1*** позволяет загрузить требуемые данные в два заданных регистра РОН.

|  |  |  |  |
| --- | --- | --- | --- |
| Адрес | Код команды | Мнемоника команды | Комментарий |
| 000  001  002 | 041  200  000 | LXI H  B2  B3 | Загрузка нач адреса  Мл.байт адреса  Ст. байт адреса |
| 003  004  005 | 001  001  010 | LXI B  B2  B3 | Запись исходных данных в В и С  В2-> (С)  B3 -> (B) |
| 006 | 160 | MOV M, B | Запоминание В |
| 007 | 043 | INX H | Формирование след адреса памяти |
| 010 | 161 | MOV M, C | Запоминание С |
| 011 | 166 | HLT | Останов |

Данная программа первоначально устанавливает адрес H=000 L=200 в регистровую пару HL. Затем в регистры C и B регистровой пары BC записываются, соответственно, 001 и 010. По команде MOV M, B происходит запоминание содержимого регистра B в памяти по адресу 000 200. Команда INX H инкрементирует адрес, расположенный в HL, и следующая команда записывает в память по этому адресу содержимое регистра C, после чего по команде HLT работа программы завершается.

Содержимое памяти после исполнения программы:

Адрес Данные

000 200 010

000 201 001

Вывод – запись данных в регистры произошла успешно.

***Программа 2*** иллюстрирует работу процессора при выполнении команд обращения к стеку.

Перед обращением к стеку необходимо инициализировать указатель стека, т.к. изначально мы не может предполагать, что в нем хранится. Это может привести к ошибке в работе программы.

При выполнении команды *PUSH RP* в ячейку памяти с адресом (SP – 1) записывается содержимое старшего регистра регистровой пары, а в ячейку с адресом (SP – 2) – содержимое младшего регистра этой пары. Стек растет в область младших адресов, поэтому содержимое указателя стека уменьшается на 2.

При выполнении команды *POP RP* данные из вершины стека, адресуемой SP, передаются в младший регистр регистровой пары, а в старший регистр загружается содержимое ячейки (SP + 1). После этого содержимое SP увеличивается на 2.

Предварительно в стек (в ячейки памяти с адресами L = 200 и L = 201) записаны числа 222 и 333 соответственно. Необходимо разработать программу, позволяющую извлечь данные, предварительно записанные в стек, изменить их на единицу и занести скорректированные данные снова в стек.

|  |  |  |  |
| --- | --- | --- | --- |
| Адрес | Код команды | Мнемоника команды | Комментарий |
| 000  001  002 | 061  200  000 | LXI SP  B2  B3 | Инициируем указатель стека  Мл байт адреса  Ст байт адреса |
| 003 | 341 | POP H | Загружаем данные из стека |
| 004 | 043 | INX H | Увеличим значение H на 1 |
| 005 | 055 | DCR L | Уменьшим значение L на 1 |
| 006 | 345 | PUSH H | Загрузим модифицированные данные в стек |
| 007 | 166 | HLT | Останов |

В результате в памяти по адресу H = 000, L = 200 записано 221, а по адресу L = 201 – 334.

***Программа 3*** реализует суммирование содержимого аккумулятора с содержимым ячейки памяти.

|  |  |  |  |
| --- | --- | --- | --- |
| Адрес | Код команды | Мнемоника команды | Комментарий |
| 000  001  002 | 041  200  000 | LXI H  B2  B3 | Загрузка начального адреса памяти  Мл байт адреса  Ста байт адреса |
| 003  004 | 076  010 | MVI A  B2 | Загрузка в (A) слагаемого |
| 005 | 206 | ADD M | Сложение значения (А) и (М) |
| 006 | 167 | MOV M, A | Запись в память результата |
| 007 | 166 | HLT | Останов |

Исходные данные:

Адрес Данные

000 200 107

Результат выполнения программы:

Адрес Данные

000 200 117

После каждого выполнения программы содержимое ячейки памяти по адресу L=200 увеличивается на 010.

***Программа 3.1.*** Разработайте программу вычитания двух чисел, одно из которых расположено в регистре блока РОН. Результат нужно занести в память.

|  |  |  |  |
| --- | --- | --- | --- |
| Адрес  H = 000, L = | Код  команды | Мнемоника команд | Комментарии |
| 000  001  002 | 041  200  000 | LXI H  B2  B3 | Загрузка начального адреса памяти  Мл. байт  Ст. байт |
| 003  004 | 076  047 | MVI A  B2 | Загрузка в аккумулятор числа 047 |
| 005  006 | 006  021 | MVI B  B2 | Загрузка в регистр B числа 021 |
| 007 | 220 | SUB B | Вычитание (А)-(В) |
| 010 | 167 | MOV M,A | (А) → (M) |
| 011 | 166 | HLT | Останов |

В результате выполнения по адресу H = 000, L = 200 хранится результат вычитания

047 – 021 = 026.

***Программа 3.2*** реализует сложение двоично-десятичных чисел, хранящихся в ячейках по адресам L=200 и L=201, и записывает результат в ячейку по адресу L=202.

|  |  |  |  |
| --- | --- | --- | --- |
| Адрес | Код команды | Мнемоника команд | Комментарии |
| L=000  001  002 | 041  200  000 | LXI H  B2  B3 | Загрузка начального адреса памяти  Мл. байт адреса  Ст. байт адреса |
| 003 | 176 | MOV A,M | (M) → (A) |
| 004 | 043 | INX H | Инкремент адреса памяти |
| 005 | 206 | ADD M | (A) + (M) → (A) |
| 006 | 047 | DAA | Десятичная коррекция результата в А |
| 007 | 043 | INX H | Инкремент адреса памяти |
| 010 | 167 | MOV M,A | (A) → (M) |
| 011 | 166 | HLT | Останов |

Перед выполнением программы запишем 044(двоично-десятичное 24) по адресу L=200 и 111(двоично-десятичное 49) по адресу L=201. После выполнения в ячейке по адресу L=202 будет записано число 163(двоично-десятичное число 73).

Результат выполнения программы правильный.

***Программа 3.3*** реализует вычитание двоично -десятичных чисел. Для вычитания двоично-десятичных чисел необходимо производить преобразование вычитаемого в дополнительный код, потому что в противном случае команда DAA, применяемая после вычитания чисел, приводит к некорректным результатам.

|  |  |  |  |
| --- | --- | --- | --- |
| Адрес | Код команды | Мнемоника команды | Комментарий |
| 000  001  002 | 041  200  000 | LXI H  B2  B3 | Загрузка начального адреса памяти  Мл байт адреса  Ст байт адреса |
| 003  004 | 076  231 | MVI A  B2 | Загружаем в (A) число 99  992/10 = 2318 |
| 005  006 | 226  074 | SUB M  INR A | (A)-M -> (A)  Инкрементирование содержимого аккумулятора |
| 007 | 043 | INX H | Формирование следующего адреса памяти |
| 010 | 206 | ADD M | (A)+(M) -> (A) |
| 011 | 047 | DDA | Двоично-десятичная коррекция |
| 012  013 | 043  167 | INX H  MOV M,A | Формирование следующего адреса  Перемещение из A в память |
| 014 | 166 | HLT | Останов |

Исходные данные:

Адрес Содержимое

000 200 024(двоично-десятичное 14)

000 201 142(двоично-десятичное 62)

Результат выполнения программы:

Адрес Содержимое

000 202 110(двоично-десятичное число 48)

***Программа 3.4*** реализует сложение двух 16-разрядных двоичных чисел.

|  |  |  |  |
| --- | --- | --- | --- |
| Адрес | Код команды | Мнемоника команд | Комментарии |
| L=000  001  002 | 041  200  000 | LXI H  B2  B3 | Загрузка начального адреса памяти  Мл. байт адреса  Ст. байт адреса |
| 003  004  005 | 001  371  174 | LXI B  B2  B3 | Загрузка пары регистров  B2 → (C)  B3 → (В) |
| 006  007  010 | 021  073  202 | LXI D  B2  B3 | Загрузка пары регистров  B2 → (E)  B3 → (D) |
| 011 | 171 | MOV A,C | (C) → (A) |
| 012 | 203 | ADD E | (A) + (E) → (A) |
| 013 | 167 | MOV M,A | (A) → (M) |
| 014 | 043 | INX H | Инкремент адреса памяти |
| 015 | 170 | MOV A,B | (B) → (A) |
| 016 | 212 | ADC D | (A) + (D) + C → (A) |
| 017 | 167 | MOV M,A | (A) → (M) |
| 020 | 166 | HLT | Останов |

16-разрядные двоичные числа записываются самой программой в регистровые пары B и D. Результат записывается в ячейки памяти по адресам L=200 и L=201.

В данном случае реализуется сложение чисел 174 371 и 202 073. После выполнения программы в ячейке по адресу L=200 записано число 064 (младший байт результата), а по адресу L=201 – число 377 (старший байт результата). Команда ADC RS складывает содержимое аккумулятора с содержимым RS с учётом флага переноса.

***Программа 3.5*.** Выполните вычитание двух 16-разрядных чисел.

|  |  |  |  |
| --- | --- | --- | --- |
| Адрес | Код команды | Мнемоника команды | Комментарий |
| 000  001  002 | 001  043  126 | LXI B  B2  B3 | Загружаем первое число.  Мл байт  Ст байт |
| 003  004  005 | 021  044  125 | LXI D  B2  B3 | Загружаем второе число  Мл байт  Ст байт |
| 006 | 173 | MOV A, E | Переместим данные из регистра Е в аккумулятор |
| 007 | 221 | SUB C | Вычтем значение из регистра С из аккумулятора |
| 010  011  012 | 062  200  000 | STA Addr  B2  B3 | Запомним мл байт в ячейку с адресом 200  Мл байт адреса  Ст байт адреса |
| 013 | 172 | MOV A, D | Переместим в аккумулятор данные из регистра D |
| 014 | 230 | SBB B | Вычтем с учетом переноса значение регистра В из аккумулятора |
| 015  016  017 | 062  201  000 | STA Addr  B2  B3 | Перенесем значение из аккумулятора в ячейку 201  Мл байт адреса  Ст байт адреса |
| 020 | 166 | HLT | Останов |

В результате выполнения программы получим, что в ячейке 200 будет значение 003, а в ячейке 201 – 001, что с учетом переноса соответствует действительности. Данная программа в результате её выполнения записала в регистры Н и L числа 043 и 126 соответственно, а в регистры D и E - 044 и 125. Далее выполняется побайтное вычитание младших и старших байтов регистровых пар. Второе вычитание выполняется не командой SUB, а командой SBB, которая учитывает бит заёма, сохранившийся после первого вычитания. Результат записывается в ячейки памяти с адресами 200 и 201.

***Программа 4***реализует «обнуления» заданной области памяти.



Перед выполнением программы запишем в некоторые ячейки памяти с адресами от L=011 до L=040 ненулевые значения. После выполнения программы в автоматическом режиме содержимое всех ячеек по адресам от L=011 до L=040 обнулилось. При повторном выполнении программы в пошаговом режиме оказалось, что часть программы, записанная в ячейках от L=000 до L=004, стёрта. Это произошло, потому что в программе не предусмотрены средства останова, и программа стирала все ячейки, начиная с L=011, до тех пор, пока не перебрала все адреса памяти до конца, а затем продолжила стирать, начиная с самой первой ячейки. Когда была стерта ячейка с адресом L=004, в которой был записан код команды, осуществляющей обнуление, в программе не осталось команд, изменяющих содержимое памяти. Поэтому данные в ячейках по адресам от L=005 до L=010 сохранились.

Для корректного завершения программы её необходимо изменить:

|  |  |  |  |
| --- | --- | --- | --- |
| Адрес | Код команды | Мнемоника команд | Комментарии |
| L=000  001  002 | 041  015  000 | LXI H  B2  B3 | Загрузка начального адреса памяти  Мл. байт адреса  Ст. байт адреса |
| 003 | 227 | SUB A | Обнуление A |
| 004 | 167 | MOV M,A | (A) → (M) |
| 005 | 043 | INX H | Инкремент адреса памяти |
| 006 | 175 | MOV A,L | (L) → (A) |
| 007  010 | 326  041 | SUI  B2 | (A) – B2 → (A) |
| 011  012  013 | 302  003  000 | JNZ  B2  B3 |  |
| 014 | 166 | HLT |  |

Данная программа обнуляет содержимое ячеек по адресам от L=015 до L=040.

***Программа 5*** иллюстрирует формирование Слова Состояния Программы PSW при выполнении различных групп команд микропроцессора.

|  |  |  |  |
| --- | --- | --- | --- |
| Адрес | Код команды | Мнемоника команды | Комментарий |
| 000  001 | 006  200 | MVI B  B2 | Помещаем число 200 в регистр В |
| 002  003 | 076  001 | MVI A  B2 | Помещаем число 001 в аккумулятор |
| 004 | 250 | XRA B | Выполняем исключающее ИЛИ |
| 005 | 027 | RAL | Циклический сдвиг влево через перенос |
| 006  007  010 | 061  202  000 | LXI SP  B2  B3 | Загружаем указатель стека |
| 011 | 365 | PUSH PSW | Загружаем данных состояния флагов |
| 012 | 166 | HLT | Останов |

Слово Состояния Программы PSW состоит из содержимого аккумулятора и регистра флагов РФ. Содержимое регистра флагов РФ:

http://www.studfiles.ru/html/2706/242/html_b4zhgH0gr5.Hm7K/htmlconvd-KWkkpY_html_7dd68385.png

S – это знак (старший бит аккумулятора), Z – флаг нуля, AC – флаг вспомогательного (межтетрадного) переноса, P – флаг чётности и C – флаг переноса.

Проанализируем значение Слова Состояния Программы PSW, формируемое при выполнении команд программы:

|  |  |  |  |
| --- | --- | --- | --- |
| Мнемоника команды | Значение PSW | | Комментарии |
| (A) | РФ |
| MVI B data | xxxxxxxx | xx0x0x1x | при сбросе состояние PSW не определено |
| MVI A data | 00000001 | xx0x0x1x | Запись 001 в (А) |
| XRA B | 10000001 | 10000110 | Сложение по модулю 2 чисел 001 и 200, устанавливает биты S, P и сбрасывает биты Z, C |
| RAL | 00000010 | 10000111 | Циклический сдвиг влево через перенос, устанавливает бит C |
| LXI SP data | 00000010 | 10000111 | Не меняет значение PSW |
| PUSH PSW | 00000010 | 10000111 |
| HLT | 00000010 | 10000111 |

После выполнения программы 5 проверим содержимое ячеек памяти. По адресу L=200: 10 000 111 (содержимое регистра флагов). По адресу L=201: 00 000 010 (содержимое аккумулятора).

***Программа 6*.** Программа позволяет проверить действие команд условного перехода.

|  |  |  |  |
| --- | --- | --- | --- |
| Адрес | Код команды | Мнемоника команды | Комментарий |
| 000  001  002 | 061  200  000 | LXI SP  B2  B3 | Инициализация указателя стека |
| 003 | 361 | POP PSW | Извлечение слова состояния из стека |
| 004  005  006 | \*\*\*  000  000 | \*  B2  B3 | JNZ/JZ/JNC/JC/JPO/JPE/JP/JM  Мл адрес байта  Ст адрес байта |
| 007 | 166 | HLT | Останов |
| … | … | … | … |
| 200 | \*\*\* | - | Слово состояния программы PSW |
| 201 | 000 | - |

Далее произведем изменение программы в соответствии с таблицей и пронаблюдаем изменения в её поведении.

|  |  |
| --- | --- |
| Данные по адресу 004 | Данные по адресу 200 |
| 322/332 | 002/003 |
| 342/352 | 002/006 |
| 362/372 | 002/202 |

При каждом из таких изменений программа будет либо возвращаться в начало, либо выполнять команду HLT. Переход происходит по одному из флагов.

|  |  |  |  |
| --- | --- | --- | --- |
| Команда | Код команды | Регистр признаков | Переход |
| JC | 332 | 002 | нет |
| JNC | 322 | 002 | есть |
| JC | 332 | 003 | есть |
| JNC | 322 | 003 | нет |
| JPO | 342 | 002 | есть |
| JPE | 352 | 002 | нет |
| JPO | 342 | 006 | нет |
| JPC | 352 | 006 | есть |
| JP | 362 | 002 | есть |
| JM | 372 | 002 | нет |
| JP | 362 | 202 | нет |
| JM | 372 | 202 | есть |

***Программа 7*** демонстрирует действие команд вызова подпрограммы и возврата в основную программу.

|  |  |  |  |
| --- | --- | --- | --- |
| Адрес | Код команды | Мнемоника команд | Комментарии |
| L=000  001  002 | 061  067  000 | LXI SP  B2  B3 | Инициализация указателя стека  Мл. байт адреса  Ст. байт адреса |
| 003  004  005 | 315  015  000 | CALL addr  B2  B3 | Безусловный вызов подпрограммы |
| 006 | 166 | HLT | Останов |
| 015 | 311 | RET | Безусловный возврат из подпрограммы |

Команды CALL addr и JMP addr отличаются тем, что JMP addr просто изменяет содержимое счётчика команд, а CALL addr перед тем, как его изменить, сохраняет его в стеке для возможности возврата.

***Программа 8*** поясняет действие команд RST N и RET при работе с прерывающими программами.



Описание работы программы по машинным циклам:

|  |  |  |
| --- | --- | --- |
| Адрес  (состояние ША) | Содержимое памяти  (состояние ШД) | Комментарий |
| 000 | 061 | Инициализация указателя стека |
| 001 | 067 |
| 002 | 000 |
| 003 | 317 | Вызов подпрограммы по адресу L=010 |
| 066 | 000 | Загрузка в стек содержимого счётчика команд |
| 065 | 004 |
| 010 | 311 | Возврат из подпрограммы |
| 065 | 004 | Загрузка адреса из стека в счётчик команд |
| 066 | 000 |
| 004 | 166 | Останов |

Команды CALL addr и RST N отличаются тем, что CALL addr может вызвать подпрограмму по любому адресу, а RST N только по одному из восьми адресов (0N0, где N=0..7). Но RST N – это однобайтная команда, а CALL addr – трёхбайтная.

***Программа 9*** демонстрирует работу нескольких прерывающих друг друга вложенных программ.



Алгоритм работы программы:

Вызов подпрограммы 1 → вызов подпрограммы 2 → вызов подпрограммы 3 → вызов подпрограммы 4 → вызов подпрограммы 5 → возврат из подпрограммы 5 → возврат из подпрограммы 4 → возврат из подпрограммы 3 → возврат из подпрограммы 2 → возврат из подпрограммы 1 → останов.

**2. Индивидуальное задание: Программа циклического сдвига 16-разрадного числа вправо. Циклический вывод результата на динамическую индикацию.**

Изучение программируемого параллельного интерфейса КР580ВВ55 выполняется на стенде с помощью блоков статической и динамической индикации, реализованных на основе 4 семисегментных индикаторов LED MONITOR и СБИС КР580ВВ55.

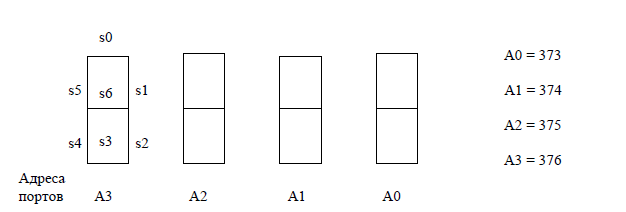
****

рис.3. Кодирование отдельных сегментов семисегментных индикаторов и

адреса портов четырехразрядного дисплея

Программирование ППИ КР580ВВ55 осуществляется командой ОUT port. В ее адресной части указывается двоичный код адреса регистра управления хххххх11. Если в исследуемой системе используется несколько БИС параллельного интерфейса, то код хххххх является кодом номера соответствующей БИС КР580ВВ55.

Обмен данными при использовании ППИ выполняется командами ввода IN port и вывода ОUT port, в которых значение port является адресом соответствующего регистра ППИ.

Изучение программируемого параллельного интерфейса КР580ВВ55 выполняется на стенде с помощью дополнительных внешних устройств, в качестве которых используются блоки статической и динамической индикации и модель дорожного светофора.

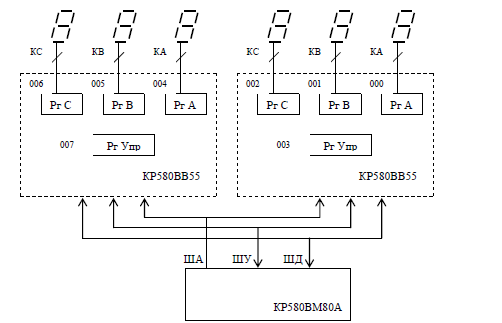


Рис.4 Структурная схема статической индикации.

Таблица. Основная часть программы вывода чисел на статистическую индикацию

|  |  |  |  |
| --- | --- | --- | --- |
| Адрес | Код команды | Мнемоника команды | Комментарии |
| 000 | 006 | MVI B,B2 | загрузка B2 (101) в регистр B |
| 001 | 101 | B2 |  |
| 002 | 016 | MVI C,B2 | загрузка B2 (202) в регистр C |
| 003 | 202 | B2 |  |
| 004 | 076 | MVI A | Помещаем число 200 в аккумулятор |
| 005 | 200 | B2 |  |
| 006 | 323 | OUT PORT | вывод |
| 007 | 003 | PORT |  |
| 010 | 323 | OUT PORT | вывод |
| 011 | 007 | PORT |  |
| 012 | 315 | CALL | вызов вывода по адресу B3 B2 (000 037) |
| 013 | 037 | B2 |  |
| 014 | 000 | B3 |  |
| 015 | 067 | STC | установка флага переноса в 1 |
| 016 | 077 | CMC | инверсия флага переноса |
| 017 | 171 | MOV A,C | запись в аккумулятор значение регистра C |
| 020 | 037 | RAR | циклический сдвиг вправо значения в аккумуляторе через перенос |
| 021 | 117 | MOV A,C | запись значения из аккумулятора в регистр С |
| 022 | 170 | MOV A,B | запись в аккумулятор значение регистра B |
| 023 | 037 | RAR | циклический сдвиг вправо значения в аккумуляторе через перенос |
| 024 | 107 | MOV A,B | запись значения из аккумулятора в регистр B |
| 025 | 322 | JNC | переход по нулевому значению флага переноса (переноса нет) в B3 B2 (000 031) |
| 026 | 031 | B2 |  |
| 027 | 000 | B3 |  |
| 030 | 014 | INR C | инкремент значения в регистре C |
| 031 | 315 | CALL | вызов вывода по адресу B3 B2 (000 037) |
| 032 | 037 | B2 |  |
| 033 | 000 | B3 |  |
| 034 | 303 | JMP | безусловный переход по адресу B3 B2 (000 135) |
| 035 | 135 | B2 |  |
| 036 | 000 | B3 |  |
| 037 | 170 | MOV A,B | запись в аккумулятор значение регистра B |
| 040 | 017 | RRC | циклический сдвиг вправо значения в аккумуляторе |
| 041 | 017 | RRC | циклический сдвиг вправо значения в аккумуляторе |
| 042 | 017 | RRC | циклический сдвиг вправо значения в аккумуляторе |
| 043 | 017 | RRC | циклический сдвиг вправо значения в аккумуляторе |
| 044 | 346 | ANI B2 | (A) = (A) ^ B2 |
| 045 | 017 | B2 |  |
| 046 | 041 | LXI H | загрузка начального адреса памяти B3 B2 (002 000) |
| 047 | 000 | B2 |  |
| 050 | 002 | B3 |  |
| 051 | 205 | ADD L | (A) = (A) + (L) |
| 052 | 157 | MOV L,A | запись значения из аккумулятора в регистр L |
| 053 | 176 | MOV A,M | запись значения из памяти в аккумулятор |
| 054 | 323 | OUT PORT | вывод |
| 055 | 004 | PORT |  |
| 056 | 170 | MOV A,B | запись в аккумулятор значение регистра B |
| 057 | 346 | ANI B2 | (A) = (A) ^ B2 |
| 060 | 017 | B2 |  |
| 061 | 041 | LXI H | загрузка начального адреса памяти B3 B2 (002 000) |
| 062 | 000 | B2 |  |
| 063 | 002 | B3 |  |
| 064 | 205 | ADD L | (A) = (A) + (L) |
| 065 | 157 | MOV L,A | запись значения из аккумулятора в регистр L |
| 066 | 176 | MOV A,M | запись значения из памяти в аккумулятор |
| 067 | 323 | OUT PORT | вывод |
| 070 | 002 | PORT |  |
| 071 | 171 | MOV A,C | запись в аккумулятор значение регистра C |
| 072 | 017 | RRC | циклический сдвиг вправо значения в аккумуляторе |
| 073 | 017 | RRC | циклический сдвиг вправо значения в аккумуляторе |
| 074 | 017 | RRC | циклический сдвиг вправо значения в аккумуляторе |
| 075 | 017 | RRC | циклический сдвиг вправо значения в аккумуляторе |
| 076 | 346 | ANI B2 | (A) = (A) ^ B2 |
| 077 | 017 | B2 |  |
| 100 | 041 | LXI H | загрузка начального адреса памяти B3 B2 (002 000) |
| 101 | 000 | B2 |  |
| 102 | 002 | B3 |  |
| 103 | 205 | ADD L | (A) = (A) + (L) |
| 104 | 157 | MOV L,A | запись значения из аккумулятора в регистр L |
| 105 | 176 | MOA A,M | запись значения из памяти в аккумулятор |
| 106 | 323 | OUT PORT | вывод |
| 107 | 001 | PORT |  |
| 110 | 171 | MOV A,C | запись в аккумулятор значение регистра C |
| 111 | 346 | ANI B2 | (A) = (A) ^ B2 |
| 112 | 017 | B2 |  |
| 113 | 041 | LXI H | загрузка начального адреса памяти B3 B2 (002 000) |
| 114 | 000 | B2 |  |
| 115 | 002 | B3 |  |
| 116 | 205 | ADD L | (A) = (A) + (L) |
| 117 | 157 | MOV L,A | запись значения из аккумулятора в регистр L |
| 120 | 176 | MOV A,M | запись значения из памяти в аккумулятор |
| 121 | 323 | OUT PORT | вывод |
| 122 | 000 | PORT |  |
| 123 | 021 | LXI D | загрузка B3 B2 (377 377) в регистровую пару D E |
| 124 | 377 | B2 |  |
| 125 | 377 | B3 |  |
| 126 | 033 | DCX D | декремент значения в регистровой паре D E |
| 127 | 024 | INR D | инкремент значения в регистре D |
| 130 | 025 | DCR D | декремент значения в регистре D |
| 131 | 302 | JNZ | переход по нулевому значению флага нулевого значения (ненулевой результат) в B3 |
| 132 | 126 | B2 |  |
| 133 | 000 | B3 |  |
| 134 | 311 | RET | выход из вывода |
| 135 | 303 | JMP | безусловный переход по адресу B3 B2 (000 015) |
| 136 | 015 | B2 |  |
| 137 | 000 | B3 |  |

Таблица символов

|  |  |  |
| --- | --- | --- |
| Адрес | Содержимое памяти | Символ |
| 002 000 | 077 | 0 |
| 002 001 | 006 | 1 |
| 002 002 | 133 | 2 |
| 002 003 | 117 | 3 |
| 002 004 | 146 | 4 |
| 002 005 | 155 | 5 |
| 002 006 | 175 | 6 |
| 002 007 | 007 | 7 |
| 002 010 | 177 | 8 |
| 002 011 | 157 | 9 |
| 002 012 | 167 | A |
| 002 013 | 174 | B |
| 002 014 | 071 | C |
| 002 015 | 136 | D |
| 002 016 | 171 | E |
| 002 017 | 161 | F |

После запуска программы, на стенде загораются сегменты на блоке статистической индикации и поочередно в цикле выводятся числа, которые являются результатом сдвига вправо предыдущего числа.

**5. Вывод.**

В ходе выполнения работы были изучены принципы организации и особенности функционирования микропроцессора серии К580, принципы проектирования и отладки микроконтроллеров на основе МП, также были получены навыки программирования микропроцессоров данного типа. Программирование подобного микропроцессора важна на данном этапе обучения, т.к. позволяет хорошо разобраться в управлении процессора и его командах.

Стоит отметить, что основной, как по мне, недостаток – это то, что если у нас есть большая программа, большая часть которой это просто вывод на сегменты, то подобное программирование превращается в тяжелую рутину и осложняет выполнение работы. Также стоит сказать, что с данным стендом стоит быть осторожнее в том плане, что были замечены случаи, когда малейшее возмущение будто случайный сдвиг стенда или задевание провода приводило к выключению данного стенда, а это означает, что записанная программа стирается и приходится начинать сначала, собственно это можно списать на возраст стенда. Еще один недостаток можно подчеркнуть из предыдущего предложения – это отсутствие сохранения программы.