Санкт-Петербургский поли­технический университет Петра Великого

Инс­титут компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

Отчет по лабораторной работе

по дисциплине «Микропроцессорные системы»

«Изучение вычислительных возможностей МК SAB 80C515»

|  |
| --- |
| \_\_\_\_\_\_\_\_\_\_\_  *подпись*  \_\_\_\_\_\_\_\_\_\_\_  *подпись* |
| \_\_\_\_\_\_\_\_\_\_\_  *подпись* |

Работу выполнили студенты группы № 43501/3 Бояркин Н.С.

Кан В.С.

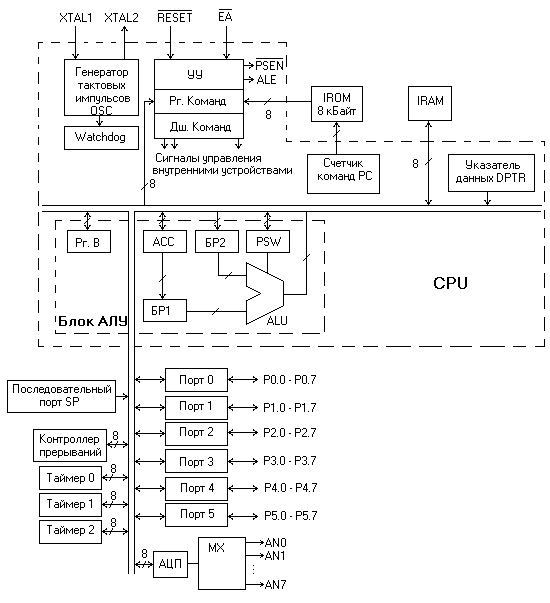
Работу принял преподаватель Кузьмин А.А.

Санкт-Петербург

2016

1. **Цель работы**
2. Знакомство с программно-аппаратным комплексом поддержки проектирования микроконтроллерных систем на базе МКSAB80C515;
3. Изучение системы команд МК семействаMCS51 на примере выполнения простейших программ.
4. **Программа работы**
5. На примере тестовой программы, осуществляющей обнуление ячеек заданной области внутренней памяти данных МК, ознакомиться с полным циклом создания прикладного программного обеспечения.
6. Разработать и выполнить программу, которая заполняет ячейки заданной области памяти линейно возрастающими значениями.
7. Разработать и выполнить программу вычисления арифметического выражения заданного вида.
8. Разработать и выполнить программу вычисления логического выражения заданного вида с использование команд битового процессора.
9. Разработать и выполнить программу, которая осуществляет заполнение последовательных ячеек внешней памяти значениями, линейно изменяющимися в заданных диапазонах.
10. Разработать и выполнить программу функциональной обработки данных.
11. **Теория**

В состав 8-битного МК SAB 80C515 входят:

1. 8-разрядное АЛУ и схемы аппаратной реализации умножения и деления;
2. Внутреннее ПЗУ (IROM) программ и констант объемом 8 Кбайт;
3. Внутреннее ОЗУ (RRAM) данных объемом 256 байт;
4. Шесть программируемых портов ввода-вывода (P1 - P5);
5. Порт ввода аналоговых сигналов (P6);
6. Полный дуплексный последовательный порт SP с фиксированной и переменной скоростью обмена;
7. Три программируемых 16-битных таймера/счетчика;
8. 4-канальный блок быстрого ввода-вывода внешних событий, обладающий дополнительными возможностями формирования ШИМ-сигналов;
9. 8-канальный аналого-цифровой преобразователь со встроенным блоком программируемых эталонных напряжений;
10. Сторожевой таймер (WDT);
11. 4-уровневая система прерываний от 12 источников прерываний;
12. Внутренний стек глубиной 256 байт;

**Программно-аппаратный комплекс SHELL51**

Проектирование микроконтроллерных систем проводится с использованием инструментальных программных и программно-аппаратных средств. Используемая программная среда SHELL51 (рис. 1) ориентирована на применение в составе комплекса, включающего ЭВМ и микроконтроллер, подключенный к ЭВМ.

Среда Shell51 позволяет работать с исходным представлением программы в виде текста на языке ассемблера. С помощью кнопки “ЗАПУСК” активизируется выполнение процесса перевода исходного текста программы в представление, пригодное для загрузки в микроконтроллер. В случае появления ошибок и предупреждений со стороны транслятора детальная информация о результатах трансляции и компоновки будет расположена на специальной вкладке “ЛИСТИНГИ”.

Программный комплекс SHELL51 содержит симулятор, который позволяет загрузить оттранслированную программу в память и отследить ее выполнение по шагам, до установленного пользователем адреса или целиком.

Вкладка “Окна памяти” позволяет загрузить программу в микроконтроллер, запустить её и передать содержимое памяти микроконтроллера обратно на компьютер для анализа полученных

результатов.

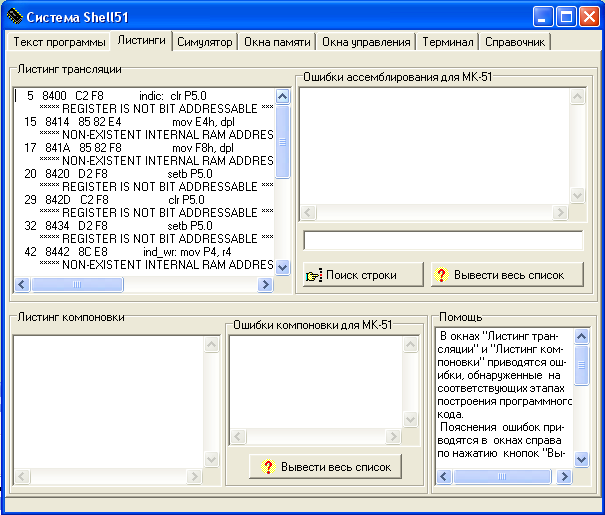


Рис. 1. Окно SHELL51.

**Лабораторный стенд СТК-1**

Лабораторный стенд (рис. 2) предназначен для изучения особенностей построения и функционирования встраиваемых микроконтроллерных систем. В состав стенда входят одноплатный контроллер, совокупность устройств дискретного и аналогового ввода/вывода, два генератора тестовых воздействий и коммутационное поле. На плате контроллера размещены МК SAB 80C535, микросхемы внешней памяти программ и данных (ОЗУ объемом 32 Кбайт), микросхемы внешней памяти программ «резидентного» монитора, схемы физического последовательного канала с оптоэлектронной развязкой и набор вспомогательных микросхем сопряжения МК с внешними устройствами.

Совокупность устройств дискретного ввода-вывода в стенде представлена клавиатурой 4х4 (устройство ввода) и блоком жидкокристаллических индикаторов ЖКИ (устройство вывода). Поскольку и клавиатура, и ЖКИ требуют программного управления, то по отношению к МК они являются объектами управления, что и отражено на схеме в их обозначении. Совокупность устройств аналогового ввода-вывода представлена регулируемым источником постоянного напряжения (потенциометром) и интегрирующим RC-звеном.

В качестве генераторов тестовых воздействий использованы источник гармонического сигнала «C» и источник импульсных сигналов «П». При формировании гармонических сигналов обеспечена возможность регулирования частоты и амплитуды. Регулируемыми параметрами генератора импульсных сигналов является частота и скважность;

Коммутационное поле позволяет задавать структуру аппаратных связей между элементами стенда и портами ввода-вывода МК. Внешние соединения реализуются с помощью набора проводников (специальных «шин»), прилагаемого к аппаратуре стенда.

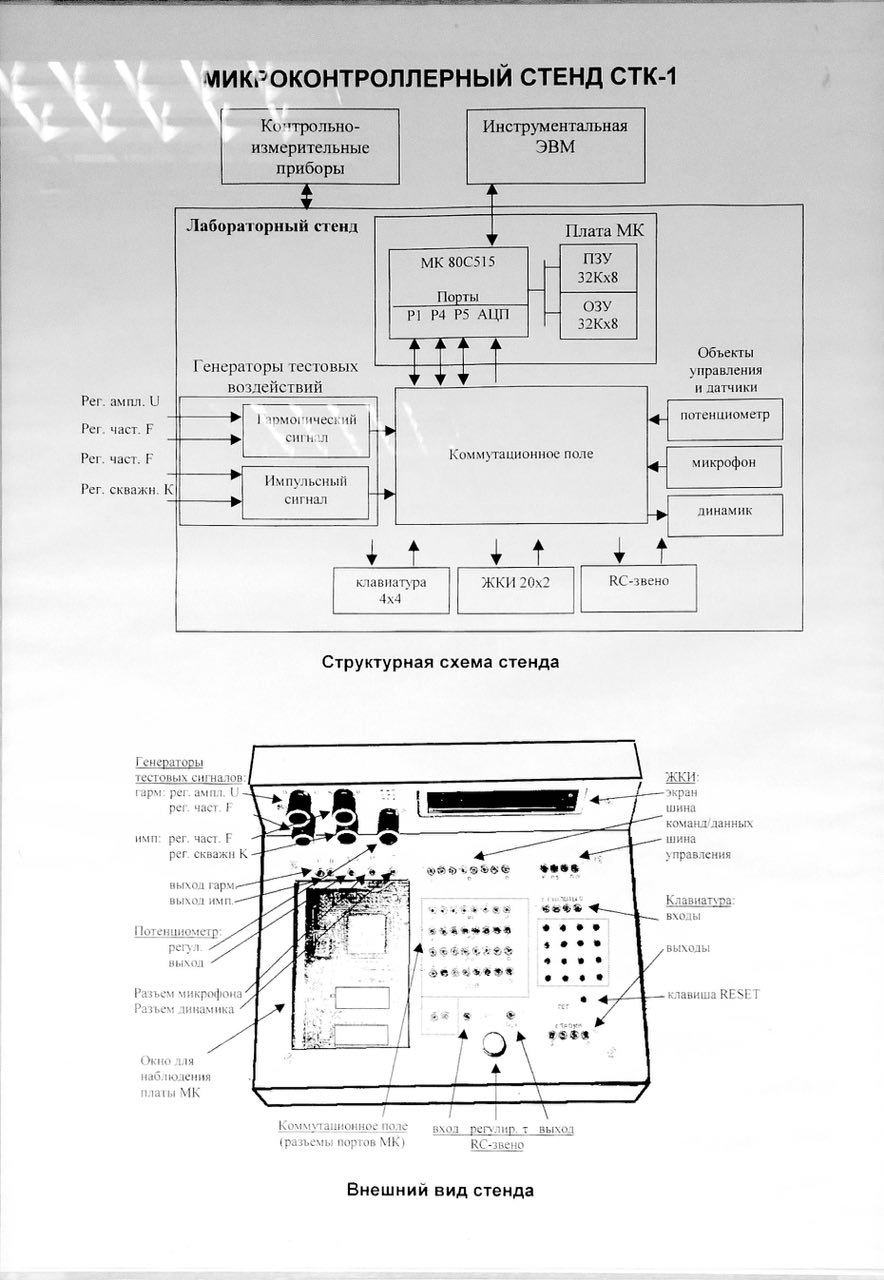
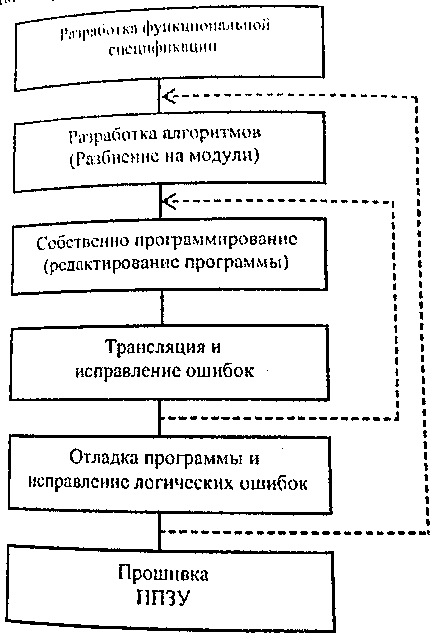


Рис. 2. Структурная схема лабораторного стенда.

1. **Выполнение первого цикла работ**

**Цикл разработки ПО на примере тестовой программы (proba.asm)**

Был рассмотрен каждый этап цикла создания прикладного программного обеспечения для МК на примере тестовой программы proba.asm:

1. Разработка функциональной спецификации (программа proba.asm должна осуществлять обнуление ячеек заданной области внутренней памяти)
2. Разработка алгоритмов (программа уже разработана)
3. Программирование (ввод программы в окно ввода SHELL51)
4. Трансляция и исправление ошибок (программа была оттранслирована и ошибок обнаружено не было, однако для примера была специально добавлена ошибка; после повторной трансляции программы появилось сообщение об ошибке во вкладке «Листинги»)
5. Отладка программы и исправление логических ошибок (отладка программы была осуществлена при помощи симулятора)
6. Прошивка ППЗУ (убедившись, что программа работает верно и логические ошибки отсутствуют, она была загружена микроконтроллер)

Тестовая программа proba.asm:

org 8400h ;размещение программы

;в памяти микроконтроллера

;начиная с адреса 0х8400

mov a,#0h ;запись в аккумулятор числа 0

mov r0,#50h ;запись в регистр

;первого адреса для обнуления

m1: mov @r0,a ;запись в ячейку по адресу,

;указанному в регистре

;числа из аккумулятора

inc r0 ;инкремент аккумулятора

;для получения следующего адреса

cjne r0,#60h,m1 ;условный переход

;с инкрементом счетчика

ret ;возврат из подпрограммы

**Ответы на вопросы**

Способы адресации в prog.asm:

* Непосредственная и регистровая адресация для mov a,#0h.
* Непосредственная и регистровая адресация для mov r0,#50h.
* Косвенная и регистровая адресация для mov @r0,a.
* Регистровая для inc r0.
* Регистровая и непосредственная для cjne r0,#60h,m1.

Директива org 8400h размещает программу в памяти микроконтроллера, начиная с адреса 0x8400.

Цикл реализовывается с помощью команды cjne. Она увеличивает счетчик команд на 3, а затем сравнивает число в регистре R0 с числом 0х60. Если равенство не выполняется, то в счетчик команд загружается адрес начала цикла, то есть метки m1*.*

**Модификация программы proba.asm (1.3.asm)**

Программа proba.asm была модифицирована для линейного изменения заданной области памяти.

Измененная тестовая программа 1.3.asm:

org 8400h ;размещение программы в памяти микроконтроллера,

;начиная с адреса 0х8400

mov a,#0A0h ; записали в аккумулятор число A0h

mov r0,#50h ; записали в регистр число 50h

m1: mov @r0, #a ; записываем число, находящиеся в аккумуляторе по адресу из

; регистра r0

dec a ; уменьшаем число в аккумуляторе

inc r0 ; инкрементируем адрес, нах-ся в регистре r0

cjne r0,#60h,m1 ; выполняем до тех пор, пока r0!=60h

ret ; возврат из подпрограммы

Изменение заключается в декременте аккумуляторе с каждой итерацией цикла. Таким образом ячейки с адресами 0x50 – 0x60 содержат линейно уменьшающиеся значения 0xA0 – 0x90.

**Разработка программы вычисления арифметического выражения (1.4.asm)**

Была разработана программа для вычисления арифметического выражения вида:

F = A \* [(B + C)/2 – 2\*B / C], где B ≤127; (B + C)/2 > 2\*B / C; C ≠ 0;

Алгоритм программы:

1. Запись данных в начальные регистры (R0(A), R1(B), R2(C))
2. Сохранение результата сложения В и С в буферном регистре (R3=R1(B)+R2(C))
3. Деление суммы В и С на 2 (R3=R3/2)
4. Умножение B на 2 и сохранение в аккумуляторе результата (ACC=R1(B)\*2)
5. Деление значения аккумулятора (R1(B)\*2) на число С и сохранение это уже в ненужном регистре R1(B) (R1=ACC/R2(C))
6. Сохранение разницы результатов п.3 и п.5 в аккумуляторе (ACC=R3-R1)
7. Умножение аккумулятора на 2 и вывод результата операции в регистры R1, R2.

Программа 1.4.asm:

org 8400h

mov r0, #10h ;A

mov r1, #22h ;B

mov r2, #40h ;C

mov a, r1

add a, r2

mov r3, a

mov a, r3

mov b, #2h

div ab

mov r3, a

mov a, r1

mov b, #2h

mul ab

mov b, r2

div ab

mov r1, a

mov a, r3

subb a, r1

mov b, r0

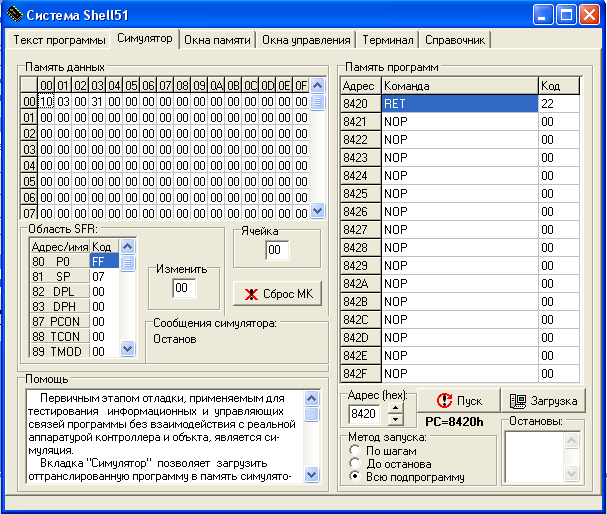
mul ab

mov r1,b

mov r2,a

ret

Результат выполнения программы в SHELL51:



Результатом выполнения программы является число 0300h, лежащее в регистрах R1, R2. И действительно, если считать только целые части 10h\*((22h+40h)/2h-(2h\*22h)/40h)=10h\*(31h-44h/40h)=10h\*(31h-1h)=0300h.

Протестируем программу на разных исходных данных:

|  |  |
| --- | --- |
| Исходные данные | Результат |
| А = 10h, В = 22h, С = 40h | 0300h |
| А = 15h, В = 16h, С = 30h | 02CAh |
| А = 30h, В = 32h, С = 60h | 0CF0h |

**Разработка программы вычисления логического выражения (1.5.asm)**

Была разработана программа для вычисления логического выражения вида:

Y = (a AND (NOT b)) XOR (c OR d)

Алгоритм программы:

1. Задаем адрес битовых переменных и сами переменные. Соответствие задаваемых битов переменным: 3-d, 2-c, 1-b, 0-a (например 0Bh=00001011b > d=1, c=0, b=1, a=1).
2. f1 = a and not b
3. f2 = c or b
4. f3 = f1 xor f2 = (f1 and not f2) or (not f1 and f2)
5. Запись результата в 28й бит от адреса 20h.

Программа 1.5.asm:

org 8400h

mov r0,#20h ; operands addrs

mov @r0,#0Bh ; value, 4 little bits 3-d 2-c 1-b 0-a

; f1(08h) = A(0h) and not B(1h)

mov C, 00h

anl C, /01h

mov 08h,C

; f2(09h) = C or D

mov C, 02h

orl C, 03h

mov 09h,c

; f1(08h) xor f2(09h)

mov C,08h

anl C,/09h

mov 0ah,C

mov C,09h

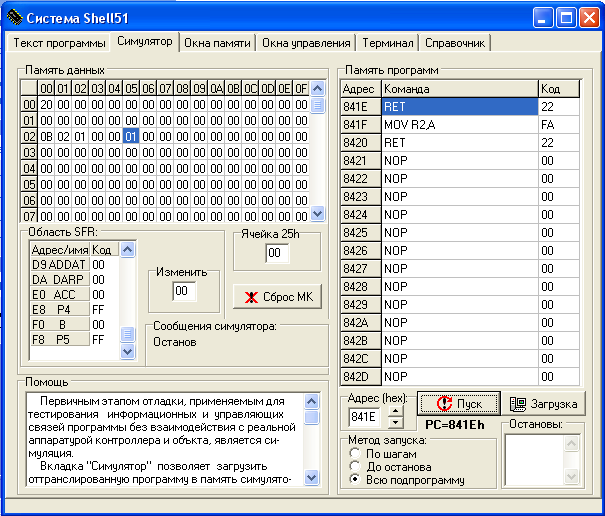
anl C,/08h

orl C,0ah

mov 28h,C

ret

Результат выполнения программы в SHELL51:



Результатом выполнения программы является 1, лежащая в нулевом бите регистра по адресу 25h. И действительно, Y = (1 and (not 1)) xor (0 or 1) = 0 xor 1 = 1.

Протестируем программу на разных исходных данных:

|  |  |
| --- | --- |
| Исходные данные | Результат |
| a=1, b=1, c = 0, d = 1 | 1 |
| a=1, b=0, c = 0, d = 0 | 1 |
| a=1, b=1, c = 0, d = 0 | 0 |
| a=1, b=0, c = 0, d = 1 | 0 |

**Разработка программы заполнение последовательных ячеек внешней памяти (1.6.asm)**

Была разработана программа для заполнения ячеек памяти в некотором диапазоне (например 8400h – 8420h) заданными линейно изменяющимися значениями: 0 – 10h – 0.

Алгоритм программы:

1. Инициализируем адрес внешней памяти dptr и переменные значения и счетчика.
2. Выполняем первый цикл 10h раз, увеличивая dptr и увеличивая переменную загружаемого значения.
3. Выполняем второй цикл 10h раз, увеличивая dptr и уменьшая переменную загружаемого значения.

Программа 1.6.asm:

оrg 8500h

mov dptr, #8400h

mov r0, #0h

mov r1, #10h

m1: mov a, r0

movx @dptr, a

inc r0

inc dptr

dec r1

mov a, r1

jnz m1

mov r1, #11h

m2: mov a, r0

movx @dptr, a

dec r0

inc dptr

dec r1

mov a, r1

jnz m2

ret

Итоговый результат:

|  |  |
| --- | --- |
| Адрес | Код |
| 8400h | 00h |
| 8401h | 01h |
| 8402h | 02h |
| … | … |
| 8410h | 10h |
| 8411h | 0Fh |
| 8412h | 0Eh |
| … | … |
| 841Fh | 01h |
| 8420h | 00h |

**Разработка программы функциональной обработки данных (1.7.asm)**

Была разработана программа для поиска минимального, максимального и среднего арифметического в массиве чисел, заданных во внешней памяти.

Алгоритм программы:

1. Инициализируем массив во внешней памяти с помощью dptr.
2. Инициализируем переменные поиска минимума, максимума, суммы и счетчика.
3. Запускаем цикл со счетчиком по массиву.
   1. Если текущее число больше текущего максимального, то оно становится максимальным (if(@dptr > max) max = @dptr).
   2. Если текущее число меньше текущего минимального, то оно становится минимальным (if(@dptr < min) min = @dptr).
   3. Фиксируем сумму элементов.
4. Делим сумму на количество элементов и получаем среднее арифметическое.
5. Результат: максимальное в R2, минимальное в R1, среднее в R3.

Программа 1.7.asm:

org 8400h

mov dptr,#8500h

mov a, #10h

movx @dptr, a

inc dptr

mov a, #1h

movx @dptr, a

inc dptr

mov a, #fh

movx @dptr, a

inc dptr

mov a, #3h

movx @dptr, a

inc dptr

mov a, #11h

movx @dptr, a

mov dptr, #8500h

mov r1, #ffh ;min

mov r2, #0h ;max

mov r3, #0h ;sum

mov r4, #0h ;cntr

m0: movx a, @dptr

mov b, r1

cjne a, f0h , m1

jmp minres

min: mov r1, a

jmp minres

m1: jc min

minres: mov b, r2

cjne a, f0h , m2

jmp maxres

max: mov r2, a

jmp maxres

m2: jnc max

maxres: add a, r3

mov r3, a

inc r4

inc dptr

cjne r4, #5h,m0

mov a, r3

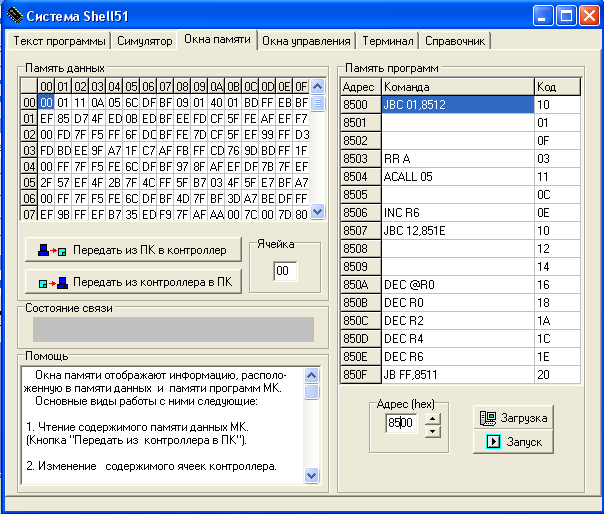
mov b, #5h

div ab

mov r3, a

ret

Результат выполнения программы в SHELL51:



Результатом выполнения программы является min=01h, max=11h, ave=0Ah. И действительно, для массива 10h, 01h, 0Fh, 03h, 11h это правильные значения.

|  |  |
| --- | --- |
| Исходные данные | Результат |
| (10h, 01h, 0Fh, 03h, 11h) | min=01h, max=11h, ave=0Ah |
| (20h, 0Ah, 05h, 1Fh, 03h) | min=03h, max=20h, ave=10h |
| (2Ah, 01h, 04h, 13h, 41h) | min=01h, max=41h, ave=1Ah |

1. **Вывод**

В данной лабораторной работе были проделаны первые шаги в понимании работы программно-аппаратного комплекса поддержки проектирования микроконтроллерных систем на базе микроконтроллера МКSAB80C515, а так же изучены системы команд МК семейства MCS51 на примере выполнения простейших программ, таких как:

1. Обнуление заданной области внутренней памяти.
2. Вычисление арифметических выражений и логических выражений с помощью.
3. Команд битового процессора.
4. Заполнение области памяти линейно меняющими значениями.
5. Функциональная обработка данных.

Также в программах была продемонстрирована работа с внешней памятью. Из внешней памяти также легко читать и записывать значения, как и из внутренней памяти. Также с помощью внешней памяти можно напрямую изменять код самой программы.