ЛАБОРАТОРНАЯ РАБОТА №3
ИСПОЛЬЗОВАНИЕ ПОДПРОГРАММ

ЦЕЛЬ РАБОТЫ: выработать навыки использования подпрограмм при разработке программы работы микроконтроллера.

ОБЩИЕ СВЕДЕНИЯ

При разработке программы работы микроконтроллера широко используется механизм работы с подпрограммами – автономными программными модулями, которые решают некоторую частную задачу и могут вызываться из различных участков основной программы.

Для вызова подпрограммы используется команда `call address` (call – звать), которая вызывает подпрограмму, размещенную по указанному адресу или метке. Вызванная подпрограмма выполняется однократно, и после ее завершения контроллер продолжает выполнение основной программы. Поэтому при вызове подпрограммы контроллер запоминает адрес точки возврата, т.е. номер команды, которая будет выполняться после завершения подпрограммы.

Вызов подпрограммы предполагает, что в памяти данных выделена специальная область, называемая стеком. Для работы со стеком используется особый регистр специальных функций **SP** (сокращение слов Stack Pointer – указатель стека). Этот регистр содержит адрес свободной ячейки стека. При записи в стек сначала байт записывается в указанную ячейку, а затем содержимое **SP** увеличивается на 1. При чтении сначала содержимое **SP** уменьшается на 1, а затем читается байт из указанной ячейки. Таким образом, при работе со стеком используется правило «последним вошел – первым вышел».

При вызове подпрограммы устройство управления микроконтроллера автоматически помещает в стек двухбайтовый адрес точки возврата. При этом содержимое **SP** автоматически увеличивается на 2. Микроконтроллер переходит к выполнению команд подпрограммы. Подпрограмма заканчивается командой `ret` (сокращение слова Return – возвращение). Исполняя эту команду, устройство управления микроконтроллера читает из стека адрес точки возврата и записывает его в счетчик команд **PC**. Микроконтроллер возвращается к выполнению основной программы, при этом содержимое **SP** уменьшается на 2.

При вызове подпрограммы и ее завершении работа со стеком выполняется автоматически без участия программиста. Однако имеется возможность прямой работы со стеком для записи и чтения данных. Для этих целей служат команды `push address` (push – толкать), которая записывает в стек содержимое указанной ячейки памяти с автоматическим увеличением указателя стека, и команда `pop address` (pop – закладывать), по которой происходит чтение байта из стека и запись по указанному адресу с автоматическим уменьшением указателя стека.

Работа подпрограммы обычно сопровождается модификацией данных в

используемых регистрах. При этом важно не «испортить» данные, которые, возможно, используют другие участки программы. Обычно это содержимое аккумулятора, **PSW** и регистров общего назначения. Для этого часто используют закрепление различных банков РОН за основной программой и подпрограммами, сохранение значений аккумулятора и **PSW** в стеке при вызове подпрограммы.

ЗАДАНИЕ И ВАРИАНТЫ ВЫПОЛНЕНИЯ РАБОТЫ

Написать программу, реализующую циклическое изменение числа Z, хранящегося в ОЗУ, в соответствии с заданной формулой. Однобайтовые входные данные X для вычислений должны последовательно считываться из массива данных, хранящихся в памяти программ.

Считываемые из памяти программ данные используются для вычисления результата, если выполнено условие, указанное в задании. Критерий прекращения вычислений также задается заданием.

Вычисление по заданной формуле должно выполняться в подпрограмме. Подпрограмма должна вызываться в основном цикле, в котором выполняются считывание данных из памяти программ и проверка соответствия данных заданным условиям.

Рекомендуемые значения для массива данных: 209, 78, 203, 251, 146, 225, 170, 91, 15,92, 58, 55, 217, 39, 162, 23, 112, 8, 227, 200, 17,116, 200, 64, 105.

Варианты выполнения лабораторной работы представлены в таблице 5.

Таблица 5 – Варианты выполнения лабораторной работы

№	Банк РОН	Формула для вычисления	Условие использования байта данных X	Критерий прекращения вычислений
1	2	Z = 2X + Z/2	X>20	Цикл выполнен 10 раз
2	1	Z = X – 8Z	X>40	Цикл выполнен 12 раз
3	2	Z = 3X + 100+Z	X>50	Z >1024
4	3	Z = -X – 2Z	X<0	Цикл выполнен 7 раз
5	1	Z = 2X + Z	X<150	Z >500
6	1	Z = 4X + Z	X - четное число	Цикл выполнен 5 раз
7	2	Z = X – 2Z	X - четное число	Цикл выполнен 9 раз
8	3	Z = 4X – 2Z	X - четное число	Цикл выполнен 7 раз
9	3	Z = X/2 + Z	X - четное число	Z >800
10	1	Z = X/4 – Z	X - четное число	Цикл выполнен 10 раз
11	2	Z = X/2 – 2Z	X - нечетное число	Z <-1024
12	3	Z = X – Z/4	X - нечетное число	Цикл выполнен 11 раз

Окончание таблицы 5

№	Банк РОН	Формула для вычисления	Условие использования байта данных X	Критерий прекращения вычислений
13	1	Z = X – 4Z	X - нечетное число	Z <-512
14	1	Z = X or Z	X - нечетное число	Цикл выполнен 9 раз
15	2	Z = X and Z	X - нечетное число	Z=1
16	3	Z = X or (Z/2)	X - нечетное число	Цикл выполнен 6 раз
17	2	Z = X and (-Z)	X>45	Цикл выполнен 10 раз
18	1	Z = (-X) or Z	X<0	Z =127
19	2	Z = 4(X – Z)	X>80	Цикл выполнен 7 раз
20	3	Z = (X + Z)/4	X<200	Z >1200

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Порядок выполнения работы и содержание отчета изложены в описании лабораторной работы №2.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Чем отличаются команды `acall Subr` и `lcall Subr`?
2. Сколько байт нужно предусмотреть в стеке для сохранения данных регистра **PC**?
3. Почему в команде `push ACC` следует писать **ACC**, а не **A**?
4. Сколько байт займут в стеке данные регистра **PSW**?
5. Какие действия выполняет микроконтроллер по команде `ret`?