Государственное образовательное учреждение
Высшего профессионального образования
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

# Методическое пособие по курсу «Микропроцессоры в измерительных устройствах»

# 3

**В.Н. Бориков**

## ПРОГРАММИРОВАНИЕ ВЫЧИСЛИТЕЛЬНЫХ ФУНКЦИЙ AVR - МИКРОКОНТРОЛЛЕРОВ

**УЧЕБНОЕ ПОСОБИЕ ДЛЯ ПРОВЕДЕНИЯ ПРАКТИЧЕСКИХ ЗАНЯТИЙ**

Томск 2010

УДК 621.38

Бориков В. Н., Методическое пособие по микропроцессорам: Программирование вычислительных функций AVR - микроконтроллеров. Учебное пособие для проведения практических занятий - Томск: Изд. ТПУ, 2010. – 48с.

Методические указания рассмотрены и рекомендованы к изданию методическим семинаром кафедры компьютерных измерительных систем и метрологии 31.08.2010 г.

## 1 Системы счисления

При программировании микроконтроллеров AVR пользуются различными системами счисления: двоичной, десятичной и шестнадцатеричной. В таблице 1 приведен пример счета в различных системах счисления.

Таблица 1.1 - Пример счета в различных системах счисления

| Двоичная (8 разрядов) | Десятичная (3 разряда) | Шестнадцатеричная (2 разряда) |
|---|---|---|
| 00000000 | 000 | 00 |
| 00000001 | 001 | 01 |
| 00000010 | 002 | 02 |
| 00000011 | 003 | 03 |
| 00000100 | 004 | 04 |
| 00000101 | 005 | 05 |
| 00000110 | 006 | 06 |
| 00000111 | 007 | 07 |
| 00001000 | 008 | 08 |
| 00001001 | 009 | 09 |
| 00001010 | 010 | 0A |
| 00001011 | 011 | 0B |
| 00001100 | 012 | 0C |
| 00001101 | 013 | 0D |
| 00001110 | 014 | 0E |
| 00001111 | 015 | 0F |
| 00010000 | 016 | 10 |
| 00010001 | 017 | 11 |
| и т.д. | | |

Двоичный разряд (или *бит)*, расположенный справа, называется младшим значащим разрядом (МЗР) или младшим значащим битом (Least Significant Bit – *LSB*), а также битом 0. Нулевой бит показывает количество единиц в числе. Единица равна $2^0$. Бит, расположенный левее (бит 1), показывает количество «двоек», следующий бит (бит 2) показывает количество «четверок» и т.д. Отметим, что $2 = 2^1$, а $4 = 2^2$, т.е. номер бита соответствует степени двойки, представляемой этим битом. **Нумерация битов ведется справа налево**. Совокупность 8 битов называется байтом. Самый старший бит двоичного слова (например, 7-й бит байта) называется

старшим значащим разрядом (СЗР) или старшим значащим битом (Most Significant Bit - MSB).

**Десятичная система счисления**

Десятичные числа являются числами с *основанием 10*, с десятью различными цифрами (от 0 до 9).

**Двоичная система счисления**

Двоичные числа являются числами с *основанием 2* (т.е. каждая цифра может принимать только два значения: 0 или 1).

Чтобы преобразовать десятичное число в двоичное, необходимо найти наибольшую степень двойки, которая будет меньше этого числа, вычесть и многократно повторить указанные вычисления.

**Пример 1.1** Найдем двоичный эквивалент десятичного числа 83.

Наибольшая степень двойки, меньше 83, равна $64 = 2^6$. Бит $6 = 1$.

Разность $83 - 64 = 19$, $32 > 19$, поэтому бит $5 = 0$;

$16 < 19$, поэтому бит $4 = 1$.

Разность $19 - 16 = 3$, $8 > 3$, поэтому бит $3 = 0$;

$4 > 3$, поэтому бит $2 = 0$;

$2 < 3$, поэтому бит $1 = 1$.

Разность $3 - 2 = 1$, $1 = 1$, поэтому бит $0 = 1$.

Таким образом, двоичный эквивалент десятичного числа 83 равен **1010011**, или $83_{(10)} = 1010011_{(2)}$.

В то же время существует другой метод, который, вероятно, покажется более легким. Возьмите число, которое вы хотите преобразовать, и разделите его на два. Если остаток равен единице (т.е. число было нечетным), запишите единицу. Затем снова разделите результат на два и так далее, записывая остаток *слева* от предыдущего значения, до тех пор, пока делимое (и остаток) не станет равным единице.

**Пример 1.2** Найдем двоичный эквивалент десятичного числа 83.

Делим 83 на 2. Частное 41, остаток 1.

Делим 41 на 2. Частное 0, остаток 1.

Делим 20 на 2 Частное 10, остаток 0.

Делим 10 на 2. Частное 5, остаток 0.

Делим 5 на 2. Частное 2, остаток 1.

Делим 2 на 2. Частное 1, остаток 0.

Делим 1 на 2. Частное 0, остаток 1.

Таким образом, двоичный эквивалент десятичного числа 83 равен 1010011, или $83_{(10)} = 1010011_{(2)}$.

**Задание 1.1** Найдите двоичный эквивалент десятичного числа 199.

**Задание 1.2** Найдите двоичный эквивалент десятичного числа 170.

**Шестнадцатеричная система счисления**

Числа в шестнадцатеричной системе имеют *основание 16* и представлены 16 различными цифрами (0, 1, 2, 3,4, 5, 6, 7, 8, 9, A, B, C, D, E и F).

Аналогично двоичным числам разряд 0 шестнадцатеричного числа показывает количество единиц ($16^0 = 1$), разряд 1 - количество чисел 16 ($16^1 = 16$) и т.д. Чтобы преобразовать десятичное число в шестнадцатеричное (вместо этого слова часто используют сокращение «hex»), следует определить, сколько в числе содержится единиц и сколько чисел 16.

**Пример 1.3** Преобразуем десятичное число 59 в шестнадцатеричное. В числе 59 содержится три числа 16, поэтому 1-й разряд равен 3. Разность $59 - 48 = 11$; число 11 соответствует шестнадцатеричному B, поэтому 0-й разряд равен B. Следовательно, искомое число равно 3B.

**Задание 1.3** Найдите шестнадцатеричный эквивалент десятичного числа 199.

**Задание 1.4** Найдите шестнадцатеричный эквивалент десятичного числа 170.

Одной из полезных особенностей шестнадцатеричной системы, является очень простое преобразование двоичных чисел в шестнадцатеричные. Если разбить двоичное число на 4-битные группы (называемые *полубайтами*, или *тетрадами*), то каждая такая группа будет соответствовать одному шестнадцатеричному разряду.

**Пример 1.4** Преобразуем число 01101001 в шестнадцатеричную систему счисления. Делим число на полубайты: 0110 и 1001. Нетрудно заметить, что 0110 соответствует $4+2=6$, а 1001 соответствует $8+1=9$. Таким образом, указанное 8-битное число равно 69 в шестнадцатеричной системе. Очевидно, что это преобразование выполнить гораздо проще, чем в случае десятичных чисел, поэтому при программировании шестнадцатеричные числа используются намного чаще.

**Задание 1.5.** Преобразуйте 11100111 в шестнадцатеричное число.

## 2 Основные булевы операции

Существуют четыре основные булевые операции: ИЛИ, И, ИСКЛЮЧАЮЩЕЕ ИЛИ и НЕ.

Обычно в микропроцессорах предусматриваются команды для выполнения этих операций. Все указанные операции, за исключением НЕ, выполняются при двух переменных на входе, и в каждом случае получается только один выход. Операция НЕ характеризуется одним входом и одним выходом.

### Операция ИЛИ

Если обе входные переменные A и B вентиля «ИЛИ» имеют значение 0, то на выходе его появится 0. Во всех остальных случаях выход принимает значение 1. Обычно для обозначения этой операции используются символы «+» или «U» (объединение).

Пример операции ИЛИ приведен в таблице 2.1.

Таблица 2.1 – Операция ИЛИ

| A | B | AUB |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

**Пример 2.1** Проведем логическую операцию ИЛИ с двумя 8-разрядными числами: 10111100 U 11110000.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Первое число | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| Второе число | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| Результат | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |

**Задание 2.1** Проведите логическую операцию ИЛИ с двумя 8-разрядными числами: 10101010 U 10110000.

**Операция И**

Если обе входные переменные A и B вентиля «И» принимают значение 1, то на выходе появляется 1. Во всех остальных случаях выход принимает значение 0. Операция обозначается символами «•» (умножение) или «∩» (пересечение).

Пример операции И приведен в таблице 2.2.

Таблица 2.2 – Операция И

| A | B | A∩B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

**Пример 2.2** Проведем логическую операцию И с двумя 8-разрядными числами: 10111100 ∩ 11110000.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Первое число | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| Второе число | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| Результат | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |

**Задание 2.2** Проведите логическую операцию И с двумя 8-разрядными числами: 10101010 U 10110000.

**Операция ИСКЛЮЧАЮЩЕЕ ИЛИ**

Если обе входные переменные A и B вентиля «ИСКЛЮЧАЮЩЕЕ ИЛИ» имеют одинаковое значение, то выход принимает значение 0. В остальных случаях выход равен 1. Операция обозначается символами ⊕ или ∀.

Пример операции ИСКЛЮЧАЮЩЕЕ ИЛИ приведен в таблице 2.3.

Таблица 2.3 – Операция ИСКЛЮЧАЮЩЕЕ ИЛИ

| A | B | A ⊕ B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

**Пример 2.3** Проведем логическую операцию ИСКЛЮЧАЮЩЕЕ ИЛИ с двумя 8-разрядными числами: 10111100 ⊕ 11110000.

Первое число 1 0 1 1 1 1 0 0
Второе число 1 1 1 1 0 0 0 0
Результат 0 1 0 0 1 1 0 0

**Задание 2.3** Проведите логическую операцию ИСКЛЮЧАЮЩЕЕ ИЛИ с двумя 8-разрядными числами: 10101010 ⊕ 10110000.

**Операция НЕ**

НЕ инвертирует любую двоичную цифру или группу цифр, т.е.

НЕ 1 = 0

НЕ 0 = 1

**Пример 2.4** Проведем логическую операцию НЕ с 8-разрядным числом 10111100.

Число 1 0 1 1 1 1 0 0
Результат 0 1 0 0 0 0 1 1

**Задание 2.4** Проведите логическую операцию НЕ с 8-разрядным числом 10101010.

Остальные логические функции реализуются с использованием четырех основных логических операций. Например, комбинация И и НЕ дает функцию И-НЕ, а комбинация ИЛИ и НЕ - функцию ИЛИ-НЕ.

## 3 Арифметические операции

### Двоичное сложение

Сложение двоичных чисел выполняется абсолютно по тем же правилам, что и десятичных. Посмотрим различные комбинации битов.

$0 + 0 = 0$ нет переноса

$1 + 0 = 1$ нет переноса

$1 + 1 = 0$ перенос 1

$1 + 0 + 0 = 1$ нет переноса

$1 + 1 + 0 = 0$ перенос 1

$1 + 1 + 1 = 1$ перенос 1

**Пример 3.1** Сложим два 4-разрядных числа: 0100 + 0111

| | |
|---|---|
| Промежуточный перенос | 1 |
| Первое слагаемое | 0 1 0 0 ($4_{10}$) |
| Второе слагаемое | 0 1 1 1 ($7_{10}$) |
| Результат | 1 0 1 1 ($11_{10}$) |

Таким образом, результатом сложения является 4-разрядное число 1011.

**Пример 3.2** Сложим два 8-разрядных числа: 01010011 + 00111000

| | |
|---|---|
| Промежуточный перенос | 1 1 |
| Первое слагаемое | 0 1 0 1 0 0 1 1 ($83_{10}$) |
| Второе слагаемое | 0 0 1 1 1 0 0 0 ($56_{10}$) |
| Результат | 1 0 0 0 1 0 1 1 ($139_{10}$) |

Таким образом, результатом сложения является 8-разрядное число 100010011.

**Задание 3.1** Вычислите 01011010 + 00001111, используя двоичное сложение.

**Двоичное вычитание**

Двоичное вычитание можно выполнить, применяя арифметику двоичного дополнения.

Двоичное дополнение (дополнительный код) двоичного числа можно получить, заменив в числе нули единицами, а единицы - нулями, прибавив затем единицу. На первом шаге этого преобразования формируется обратный код числа.

Двоичное вычитание может быть выполнено прибавлением дополнительного кода вычитаемого к уменьшаемому. В качестве примера рассмотрим следующее двоичное вычитание:

| | | |
|---|---|---|
| Уменьшаемое | 0 0 1 0 0 1 1 1 | ($39_{10}$) |
| Вычитаемое | 0 0 0 1 1 0 1 0 | -($26_{10}$) |
| | 0 0 0 0 1 1 0 1 | ($13_{10}$) |

С помощью дополнительного кода вычитание можно выполнить следующим образом:

| | |
|---|---|
| | 0 0 0 1 1 0 1 0 |
| Инвертирование | 1 1 1 0 0 1 0 1 |
| + | 1 |
| Дополнительный код вычитаемого | 1 1 1 0 0 1 1 0 |

| | | |
|---|---|---|
| Уменьшаемое | 0 0 1 0 0 1 1 1 | (3910) |
| Вычитаемое в дополнительном коде | 1 1 1 0 0 1 1 0 | |
| 1- перенос | 0 0 0 0 1 1 0 1 | (1310) |

Перенос означает, что результат положительный.

Отсутствие цифры переноса означает, что результат имеет отрицательный знак, и окончательный ответ представляет собой дополнительный код результата сложения уменьшаемого с дополнительным кодом вычитаемого.

**Двоичное умножение**

Для проведения операции умножения двоичных чисел, необходимо использовать следующее правило:

При умножении двоичного числа на 2 мы получаем число, которое образуется в результате сдвига каждого разряда исходного двоичного числа влево на 1 разряд. Аналогично при умножении на 4 происходит сдвиг каждого разряда влево на 2 разряда, при умножении на 8 – на 3 разряда и т.д.

**Пример 3.3** Умножим 8-разрядное число 00101011 на 2.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Исходное число | | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| Сдвиг влево | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | |
| Результат | | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |

Таким образом, результат умножения: 00101011*2=01010110.

**Задание 3.2** Умножьте 8-разрядное число 00010011 на 4.

Произведем умножение двоичного числа на 3. Данную операцию можно заменить сложением исходного числа, умноженного на 2 и просто исходного числа.

**Пример 3.4** Умножим 8-разрядное число 00101011 на 3.

Т.е. 00101011*3=00101011*2 + 00101011

1) Умножаем 00101011 на 2, (один сдвиг влево):

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Исходное число | | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| Сдвиг влево | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | |
| Результат | | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |

2) Суммируем число, полученное при умножении исходного на 2 с исходным числом:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| «Сдвинутое» число | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| Исходное число | | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| Результат | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

**Задание 3.3** Умножьте 8-разрядное число 00010011 на 5.

Следовательно, рассмотрев приведенные выше примеры, мы выяснили, что умножение двоичных чисел осуществляется с помощью операций сложения и сдвига всех разрядов исходного числа влево.

**Двоичное деление**

Деление двоичных чисел проводится подобно двоичному умножению, только в этом случае при делении числа на 2, 4, 8 и т.д. каждый разряд исходного числа сдвигается вправо на 1, 2, 3 и.т.д. разряды соответственно.

**Отрицательные числа**

Рассмотрим задачу представления положительных и отрицательных чисел 8-разрядными словами. Поскольку не существует ни одного способа представить восемью битами более чем 256 различных чисел, их диапазон ограничивается величинами ± 127. Первые 128 чисел, от 0 до 127 (7F в шестнадцатеричной системе), определяются как положительные числа. Отрицательные числа получаются обратным отсчетом от нуля. Так же и в аппаратном реверсивном счетчике - если содержимое регистра равно 00000000 и происходит уменьшение этого числа на единицу, то следующим показанием счетчика будет 1111 1111 (FF в шестнадцатеричной системе). Следовательно, FF является шестнадцатеричным представлением числа — 1. Такое представление носит название дополнения до двух (или дополнительного кода числа). В табл. 2.5 приведены дополнительные коды чисел от — 8 до +7.

Заметим, что самый старший разряд показывает знак числа. Если этот разряд равен нулю, то число положительное, если единице - отрицательное.

Процедура вычисления дополнительного кода проста. Для положительных чисел дополнительные коды совпадают с их представлением в двоичной системе (как показывают первые восемь строк табл. 2.5). В случае отрицательных чисел для определения дополнительного кода необходимо произвести следующие действия:

1. Записать двоичное представление абсолютной величины числа (например, 00000101 для числа -5).

2. Произвести инвертирование (получить обратный код) двоичного числа (например, для числа 00000101 это будет 11111010).

3. Прибавить единицу для получения дополнительного кода числа (например,

11111010+1=11111011= = дополнительный код числа — 5).

Такую же процедуру нужно проделать и для получения абсолютного значения отрицательного числа по его дополнительному коду: следует образовать дополнение числа, выражающего дополнительный код, и затем прибавить единицу, Рассмотрим, например, дополни

тельный код числа 11111011:

11111011=00000100 00000100+1= 00000101= 5

Следовательно, 11111011 есть дополнительный код отрицательного числа - 5.

Заметьте, что число 11111011 можно было бы также интерпретировать как десятичное число 251, если бы его рассматривали как правильное двоичное число, а не как дополнительный код. Поэтому необходимо уметь отличать числа, являющиеся дополнительными кодами, и помнить, что с ними нужно обращаться соответствующим образом.

Дополнительные коды очень удобны для арифметических действий. Выражаю их числа при сложении, вычитании, умножении или делении также дают числа, представляющие собой дополнительные коды. Обычно они используются в микропроцесеорных системах, которые должны оперировать как с положительными, так с отрицательными числами.

Таблица 2 - Дополнительные коды чисел от —8 до +7

| Десятичное число | Двоичный дополнительный код |
|---|---|
| 7 | 0000 0111 |
| 6 | 0000 0110 |
| 5 | 0000 0101 |
| 4 | 0000 0100 |

Окончание таблицы 2

| Десятичное число | Двоичный дополнительный код |
|---|---|
| 3 | 0000 0011 |
| 2 | 0000 0010 |
| 1 | 0000 0001 |
| 0 | 0000 0000 |
| -1 | 1111 1111 |
| -2 | 1111 1110 |
| -3 | 1111 1101 |
| -4 | 1111 1100 |
| -5 | 1111 1011 |
| -6 | 1111 1010 |
| -7 | 1111 1001 |
| -8 | 1111 1000 |

**Большие и малые числа**

Несмотря на то что дополнительный код обеспечивает возможность представления отрицательных чисел, их диапазон пока еще ограничивается целыми числами, абсолютная величина которых менее 129. Диапазон этот можно расширить несколькими способами в зависимости от требуемой его ширины и необходимой степени точности.

Самый простой способ расширения диапазона чисел, который можно предложить - увеличить количество двоичных разрядов в представлении каждого числа. Это часто достигается использованием двойного слова для выражения одного числа (рисунок 1). В микропроцессоре та же цель может быть достигнута использованием двойных регистровых команд, которые обрабатывают 16 бит одновременно. Использование двойной длины слов для представления одного числа называется представлением с двойной точностью. В 8-разрядных микропроцессорах этот способ позволяет представлять числа в диапазоне от 0 до 65 535, или от — 32 767 до +32767.

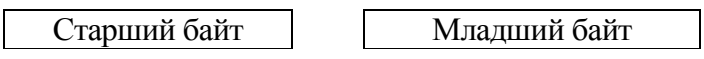


Рисунок 1

Ясно, что двойная точность расширяет диапазон целых чисел, но как быть с числами, меньшими единицы или лежащими, скажем, в интервале между 3 и 4? На рисунке 2 показан способ представления таких значений в

форме с фиксированной запятой. В этом примере для запоминания чисел используются два байта. Первый байт - это целая часть числа, т. е. та часть, которая стоит слева от десятичной (в действительности от двоичной) запятой. Второй байт - это дробная часть числа, т.е. часть, стоящая справа от двоичной запятой. Указанный способ позволяет, например, представлять столь малые числа, как 2-8 = 1/256, а также дробные числа, например 3,17. Однако разрешающая способность для дробных чисел при этом способе представления ограничивается величиной 1/256 (примерно 0,004), а представляемые числа лежат в диапазоне ±127.

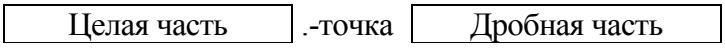


Рисунок 2


Рисунок 3

Область rгрименения указанного способа представления чисел можно расширить, если использовать для записи каждой части числа не один, а несколько байт. Однако до тех пор, пока это будет требовать относительно большого объема памяти для хранения каждого числа, представление в форме с фиксированной запятой таких чисел, как 360 000 000 000 или 0,000000297, будет оставаться неприемлемым. Заметим, что эти числа содержат много нулей, которые просто «держат место» в соответствующих позициях. Такие числа могут быть без труда представлены с использованием экспоненциальноrо представления, при котором отдельно представляются мантисса и порядок числа. Мантисса - это собственное значение числа, лежащее в диапазоне от 0 до 1. Например, число 360 000 000 000 можно записать как 0,36х 1012 (0,36-мантисса, а 12-порядок), а число 0,000000297 - как 0,297 х 10 -6 (0,297-мантисса, а -6-порядок).

Предположим далее, что для представления каждого числа используются 2 байта, как показано на рисунке 3. Один байт отводится для мантиссы, а другой-для порядка числа. Диапазон представляемых таким способом чисел при условии, что и мантисса, и порядок хранятся в виде дополнительных кодов, составляет ±10±127. Это очень широкая область

значений, так как 10127-действительно очень большое число, а 10-127-очень малое.

Такой способ, называемый представлением в форме с плавающей запятой, употребляется обычно при работе с числами, значения которых изменяются в широком диапазоне. В целях повышения разрешающей способности этого представления часто для записи числа используется более 2 байт (при этом увеличивается количество разрядов мантиссы).

Заметим, что используемый способ представления чисел должен указываться в явном виде, с тем чтобы обеспечивалось правильное декодирование каждого числа. Одни и те же два информационных байта могут декодироваться совершенно по-разному, если их запись интерпретировать как пару чисел в дополнительном коде, как одно число с фиксированной запятой или как одно число с плавающей запятой. Программные средства обработки числовых данных должны «знать», какой именно способ представления чисел используется в каждом конкретном случае.

**Представление десятичных чисел**

Большинство микропроцессорных систем оснащается десятичными устройствами ввода-вывода, каковыми являются, например, клавиатура и дисплей. А поскольку десятичная форма чисел является естественной для большинства людей, то и болыпая часть микропроцессорных систем должла предусматривать именно эту форму.

Трудность заключается здесь в отыскании удобного метода записи десятичных чисел с использованием двоичной процессорной логики. Предположим, например, что с клавиатуры считывается десятичное число 28. Оно должно быть преобразовано в двоичный эквивалент 0001 1100 (1С-в шестнадцатеричной форме). Однако, если далее это число должно быть выведено на экран десятичного дисплея, оно долж-но подвергнуться обратному преобразованию в две десятичные цифры «2» и «8».

Альтернативный способ заключается в том, чтобы взять каждую из десятичных цифр 2 и 8 и преобразовать их независимо в два 4-разрядных двоичных числа, которые затем упаковываются в один байт; в результате число 28 будет закодировано как 00101000. Такое преобразование называется двоично-десятичным кодированием десятичных чисел. Заметим, что в этом случае двоичные значения от 1010 до 1111 никогда не употребляются.

Обыкновенно двоично-кодированное представление десятичных чисел используется в системах, имеющих десятичные устройства ввода-вывода, чтобы избежать процесса десятично-двоичного преобразования. Единственный недостаток этого способа заключается в том, что он не эффективен в смысле использования памяти. Самое большое десятичное число, которое можно запомнить в одном байте, используя этот способ, равно 99, тогда как в чисто двоичном представлении это 255. Арифметические действия с двоично-десятичными числами также не удобны, поскольку эта система счисления не является «естественной». Однако большинство микропроцессоров снабжено специальными командами для обработки двоично-десятичных чисел ( команды десятичной коррекции DAA).

**Представление буквенно-цифровых данных**

Многие микропроцессорные системы должны производить действия не только над цифрами, но и над буквами. Например, в терминале вычислительной машины должно осуществляться считывание символов с клавиатуры и пересылка их в память ЭВМ. Поэтому буквы тоже должны быть представлены тем или иным способом в виде двоичных чисел.

Для этого существует широко распространенный код ASCII, называемый стандартным американским кодом для обмена информацией. В нем каждому символу присваивается определенное значение (двоичное). Заметим, что так же, как и при любых других способах представления данных, важен контекст, в котором они находятся. Например, 0101 0100

может быть как двоичным представлением десятичного числа 84, так и двоично-десятичным представлением десятичного числа 54 (символ Т в коде ASCII). Некоторые коды таблицы являются управляющими и имеют специальное назначение. Например, код ОА используется для вывода одной строки данных на печатающее устройство или на экран дисплея.

Представление буквенно-цифровых данных в системе кодирования ASCII, приведено в таблице 3.

Таблица 3 - Представление буквенно-цифровых данных в системе кодирования ASCII

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 00 | NUL | 20 | Пробел | 40 | @ | 60 | . |
| 01 | SOH | 21 | ! | 41 | A | 61 | a |
| | | 30 | 0 | | | | |
| | | 31 | 1 | | | | |
| | | 32 | 2 | | | | |
| | | 33 | 3 | | | | |
| | | 34 | 4 | | | | |
| | | 35 | 5 | | | | |
| | | 36 | 6 | | | | |
| | | 37 | 7 | | | | |
| | | 38 | 8 | | | | |
| | | 39 | 9 | | | | |
| | | 3A | : | | | | |

Кодирование символов - операция в достаточной мере произвольная, и поэтому существует множество самых различных кодов. В настоящее время наиболее распространен код ASCII, а в прошлом очень популярным был другой код, так называемый код Бодо. В машинах фирмы IBM используется расширенный двоично-десятичный код для обмена информацией EBCDIC (Extended Binary Coded Decimal Interchange Code).

## 4 Вычислительные функции AVR-микроконтроллеров

К вычислительным функциям AVR-микроконтроллеров относятся арифметические и логические команды микроконтроллеров, приведенные в таблице 4.

Таблица 4 - Арифметические и логические команды AVR-микроконтроллеров

| Мнемони-ческое обозначение | Операнды | Описание | Операция | Флаги | Кол-во тактов |
|---|---|---|---|---|---|
| ADD | Rd, Rr | Сложить два регистра | Rd ← Rd + Rr | Z,C,N, V,H | 1 |
| ADC | Rd, Rr | Сложить с переносом | Rd ← Rd + Rr + C | Z,C,N, V,H | 1 |
| ADIW | Rdl, K | Сложить непосредственное значение со словом | Rdh:Rdl←Rdh:Rdl+ K | Z,C,N, V,S | 2 |
| SUB | Rd, Rr | Вычесть два регистра | Rd ← Rd - Rr | Z,C,N, V,H | 1 |
| SUBI | Rd*, K | Вычесть константу | Rd ← Rd - K | Z,C,N, V,H | 1 |
| SBC | Rd, Rr | Вычесть с переносом | Rd ← Rd - Rr - C | Z,C,N, V,H | 1 |
| SBCI | Rd*, K | Вычесть с переносом | Rd ← Rd - K - C | Z,C,N, V,H | 1 |
| SBIW | Rdl, K | Вычесть непосредственное значение из слова | Rdh:Rdl ←Rdh:Rdl - K | Z,C,N, V,S | 2 |
| AND | Rd, Rr | Логическое И | Rd ← Rd AND Rr | Z,N,V | 1 |
| ANDI | Rd*, K | Логическое И | Rd ← Rd AND K | Z,N,V | 1 |
| OR | Rd, Rr | Логическое ИЛИ | Rd ← Rd OR Rr | Z,N,V | 1 |
| ORI | Rd*, K | Логическое ИЛИ | Rd ← Rd OR K | Z,N,V | 1 |
| EOR | Rd, Rr | Исключающее ИЛИ | Rd ← Rd XOR Rr | Z,N,V | 1 |
| COM | Rd | Дополнение до 1 | Rd ← $FF - Rd | Z,C,N, V | 1 |
| NEG | Rd | Дополнение до 2 | Rd ← $00 - Rd | Z,C,N, V,H | 1 |
| SBR | Rd*, K | Установка бит (-ов) в регистре | Rd ← Rd OR K | Z,N,V | 1 |
| CBR | Rd*, K | Сброс бит (-ов) в регистре | Rd ← Rd AND (FFh - K) | Z,N,V | 1 |
| INC | Rd | Увеличить на 1 | Rd ← Rd + 1 | Z,N,V | 1 |
| DEC | Rd | Уменьшить на 1 | Rd ← Rd - 1 | Z,N,V | 1 |
| TST | Rd | Проверить на 0 или минус | Rd ← Rd AND Rd | Z,N,V | 1 |
| CLR | Rd | Очистить регистр | Rd ← Rd XOR Rd | Z,N,V | 1 |

Окончание таблицы 4

| Мнемони-ческое обозначение | Операнды | Описание | Операция | Флаги | Кол-во тактов |
|---|---|---|---|---|---|
| SER | Rd | Установить регистр | Rd ← $FF | None | 1 |
| CP | Rd, Rr | Сравнить | Rd = Rr | Z,N,V,C,H | 1 |
| CPC | Rd, Rr | Сравнить с учетом переноса | Rd = Rr = C | Z,N,V,C,H | 1 |
| CPI | Rd*, K | Compare Register with Immediate | Rd = K | Z,N,V,C,H | 1 |

Обозначения, используемые в таблице 4:

Rd - регистр-приемник результата, $0 \le d \le 31$;

Rd* - регистр-приемник результата, $16 \le d \le 31$;

Rr - регистр-источник;

K - символьная или численная константа (8 бит);

C - флаг переноса.

**Описание арифметических и логических команд AVR-микроконтроллеров.**

**4.1 Команда ADD** - Сложить без переноса

Описание:

Сложение двух регистров без добавления содержимого флага переноса (C), размещение результата в регистре назначения Rd.

Операция:

| Rd ← Rd + Rr | | |
|---|---|---|
| Синтаксис: | Операнды: | Счетчик программ: |
| ADD Rd, Rr | 0 < d < 31, 0 < r < 31 | PC ← PC + 1 |

16-разрядный код операции:

| 0000 | 11rd | dddd | rrrr |
|---|---|---|---|

Булевы выражения регистра статуса (SREG):

| I | T | H | S | V | N | Z | C |
|---|---|---|---|---|---|---|---|
| - | - | <=> | <=> | <=> | <=> | <=> | <=> |

| | |
|---|---|
| H: | Rd3*Rr3+Rr3+R3+R3*Rd3<br>Устанавливается если есть перенос из бита 3, в ином случае очищается |
| S: | NEV, Для проверок со знаком |
| V: | Rd7*Rr7*R7+Rd7*Rr7*R7<br>Устанавливается если в результате операции образуется переполнение дополнения до двух, в ином случае очищается |
| N: | R7<br>Устанавливается если в результате установлен MSB, в ином случае очищается |
| Z: | R7*R6*R5*R4*R3*R2*R1*R0<br>Устанавливается если результат $00, в ином случае очищается |
| C: | Rd7*Rr7+Rr7*R7+R7*Rd7<br>Устанавливается если есть перенос из MSB результата, в ином случае очищается |
| R: | (Результат) соответствует Rd после выполнения команды |

**Пример:**

```
add r1,r2   ; Сложить r2 с r1 (r1=r1+r2)
adc r28,r28 ; Сложить r28 с самим собой
                (r28 = r28+r28)
```

**4.2 Команда ADC** - Сложить с переносом

Описание:

Сложение двух регистров и содержимого флага переноса (C), размещение результата в регистре назначения Rd.

Операция:

| Rd ← Rd + Rr + C | | |
|---|---|---|
| Синтаксис: | Операнды: | Счетчик программ: |
| ADC Rd, Rr | 0 < d < 31, 0 < r < 31 | PC ← PC + 1 |

16-разрядный код операции:

| | | | |
|---|---|---|---|
| 0001 | 11rd | dddd | rrrr |

Булевы выражения регистра статуса (SREG):

| I | T | H | S | V | N | Z | C |
|---|---|---|---|---|---|---|---|
| - | - | <=> | <=> | <=> | <=> | <=> | <=> |

| | |
|---|---|
| H: | Rd3*Rr3+Rr3+R3+R3*Rd3<br>Устанавливается если есть перенос из бита 3, в ином случае очищается |
| S: | NEV, Для проверок со знаком |
| V: | Rd7*Rr7*R7+Rd7*Rr7*R7<br>Устанавливается если в результате операции образуется переполнение дополнения до двух, в ином случае очищается |
| N: | R7<br>Устанавливается если в результате установлен MSB, в ином случае очищается |
| Z: | R7*R6*R5*R4*R3*R2*R1*R0<br>Устанавливается если результат $00, в ином случае очищается |
| C: | Rd7*Rr7+Rr7*R7+R7*Rd7<br>Устанавливается если есть перенос из MSB результата, в ином случае очищается |
| R: | (Результат) соответствует Rd после выполнения команды |

**Пример:**

```
; СложитьR1 : R0 с R3 : R2
add r2, r0 ; Сложить младший байт
adc r3, r1 ; Сложить старший байт с переносом
```

**4.3 Команда SUB** - Вычесть без переноса

Описание:

Вычитание содержимого регистра-источника Rr из содержимого регистра Rd, размещение результата в регистре назначения Rd.

Операция:

| Rd ← Rd - Rr | | |
|---|---|---|
| Синтаксис: | Операнды: | Счетчик программ: |
| SUB Rd, Rr | 0 < d < 31, 0 < r < 31 | PC ← PC + 1 |

16-разрядный код операции:

| | | | |
|---|---|---|---|
| 0001 | 10rd | dddd | rrrr |

Булевы выражения регистра статуса (SREG):

| I | T | H | S | V | N | Z | C |
|---|---|---|---|---|---|---|---|
| - | - | <=> | <=> | <=> | <=> | <=> | <=> |

| | |
|---|---|
| H: | Rd3*Rr3+Rr3+R3+R3*Rd3<br>Устанавливается если есть перенос из бита 3, в ином случае очищается |
| S: | NEV, Для проверок со знаком |
| V: | Rd7*Rr7*R7+Rd7*Rr7*R7<br>Устанавливается если в результате операции образуется переполнение дополнения до двух, в ином случае очищается |
| N: | R7<br>Устанавливается если в результате установлен MSB, в ином случае очищается |
| Z: | R7*R6*R5*R4*R3*R2*R1*R0<br>Устанавливается если результат $00, в ином случае очищается |
| C: | Rd7*Rr7+Rr7*R7+*R7*Rd7<br>Устанавливается если абсолютное значение содержимого Rr больше, чем абсолютное значение Rd, в ином случае очищается |
| R: | (Результат) соответствует Rd после выполнения команды |

**Пример:**

```
sub r13, r12 ; Вычесть r12 из r13
brne noteq   ; Перейти если r12 <> r13
noteq:  nop   ; Перейти по назначению (пустая операция)
```

**4.4 Команда SUBI** - Вычесть непосредственное значение

Описание:

Вычитание константы из содержимого регистра, размещение результата в регистре назначения Rd.

Операция:

| Rd ← Rd - K | | |
|---|---|---|
| Синтаксис: | Операнды: | Счетчик программ: |
| SUBI Rd,K | 0 < d < 31, 0 < K < 255 | PC ← PC + 1 |

16-разрядный код операции:

| | | | |
|---|---|---|---|
| 0101 | KKKK | dddd | KKKK |

Булевы выражения регистра статуса (SREG):

| I | T | H | S | V | N | Z | C |
|---|---|---|---|---|---|---|---|
| - | - | <=> | <=> | <=> | <=> | <=> | <=> |

| | |
|---|---|
| H: | Rd3*Rr3+Rr3+R3+R3*Rd3<br>Устанавливается если есть перенос из бита 3, в ином случае очищается |
| S: | NEV, Для проверок со знаком |
| V: | Rd7*Rr7*R7+Rd7*Rr7*R7<br>Устанавливается если в результате операции образуется переполнение дополнения до двух, в ином случае очищается |
| N: | R7<br>Устанавливается если в результате установлен MSB, в ином случае очищается |
| Z: | R7*R6*R5*R4*R3*R2*R1*R0<br>Устанавливается если результат $00, в ином случае очищается |
| C: | Rd7*Rr7+Rr7*R7+*R7*Rd7<br>Устанавливается если абсолютное значение содержимого Rr больше, чем абсолютное значение Rd, в ином случае очищается |
| R: | (Результат) соответствует Rd после выполнения команды |

**Пример:**

```
    subi r22, $11 ; Вычесть $11 из r22
    brne noteq    ; Перейти если r22 < > $11
    . . .
noteq:  nop       ; Перейти по назначению (пустая операция)
```

**4.5 Команда SBC** - Вычесть с переносом

Описание:

Вычитание содержимого регистра-источника и содержимого флага переноса (C) из регистра Rd, размещение результата в регистре назначения Rd.

Операция:

| Rd ← Rd - Rr - C | | |
|---|---|---|
| Синтаксис: | Операнды: | Счетчик программ: |
| SBC Rd, Rr | 0 < d < 31, 0 < r < 31 | PC ← PC + 1 |

16-разрядный код операции:

| | | | |
|---|---|---|---|
| 0000 | 10rd | dddd | rrrr |

Булевы выражения регистра статуса (SREG):

| I | T | H | S | V | N | Z | C |
|---|---|---|---|---|---|---|---|
| - | - | <=> | <=> | <=> | <=> | <=> | <=> |

| | |
|---|---|
| H: | Rd3*Rr3+Rr3+R3+R3*Rd3<br>Устанавливается если есть перенос из бита 3, в ином случае очищается |
| S: | NEV, Для проверок со знаком |
| V: | Rd7*Rr7*R7+Rd7*Rr7*R7<br>Устанавливается если в результате операции образуется переполнение дополнения до двух, в ином случае очищается |
| N: | R7<br>Устанавливается если в результате установлен MSB, в ином случае очищается |
| Z: | R7*R6*R5*R4*R3*R2*R1*R0*Z<br>Предшествовавшее значение остается неизменным если результат равен нулю, в ином случае очищается |
| C: | Rd7*Rr7+Rr7*R7+ R7*Rd7<br>Устанавливается если абсолютное значение содержимого Rr плюс предшествовавший перенос больше, чем абсолютное значение Rd, в ином случае очищается |
| R: | (Результат) соответствует Rd после выполнения команды |

**Пример:**

```
        ; Вычесть r1 : r0 из r3 : r2
  sub   r2, r0 ; Вычесть младший байт
  sbc   r3, r1 ; Вычесть старший байт с переносом
```

**4.6 Команда SBCI** - Вычесть непосредственное значение с переносом

Описание:

Вычитание константы и содержимого флага переноса (C) из содержимого регистра, размещение результата в регистре назначения Rd.

Операция:

| Rd ← Rd - K - C | | |
|---|---|---|
| Синтаксис: | Операнды: | Счетчик программ: |
| SBCI Rd,K | 0 < d < 31, 0 < K < 255 | PC ← PC + 1 |

16-разрядный код операции:

| 0100 | KKKK | dddd | KKKK |
|---|---|---|---|

Булевы выражения регистра статуса (SREG):

| I | T | H | S | V | N | Z | C |
|---|---|---|---|---|---|---|---|
| - | - | <=> | <=> | <=> | <=> | <=> | <=> |

| | |
|---|---|
| H: | Rd3*K3+K3*R3+R3*Rd3<br>Устанавливается если есть заем из бита 3, в ином случае очищается |
| S: | NEV, Для проверок со знаком |
| V: | Rd7*Rr7*R7+Rd7*Rr7*R7<br>Устанавливается если в результате операции образуется переполнение дополнения до двух, в ином случае очищается |
| N: | R7<br>Устанавливается если в результате установлен MSB, в ином случае очищается |
| Z: | R7*R6*R5*R4*R3*R2*R1*R0*Z<br>Предшествовавшее значение остается неизменным, если результат равен нулю, в ином случае очищается |
| C: | Rd7*K7+K7*R7+ R7*Rd7<br>Устанавливается если абсолютное значение константы плюс предшествовавший перенос больше, чем абсолютное значение Rd, в ином случае очищается |
| R: | (Результат) соответствует Rd после выполнения команды |

**Пример:**

```
        ; Вычесть $4F23 из r17 : r16
subi  r16, r23  ; Вычесть младший байт
sbci  r17, $4F  ; Вычесть старший байт с переносом
```

**4.7 Команда AND** - Выполнить логическое AND

Описание:

Выполнение логического AND между содержимым регистров Rd и Rr и помещение результата в регистр назначения Rd.

Операция:

| Rd ← Rd AND Rr | | |
|---|---|---|
| Синтаксис: | Операнды: | Счетчик программ: |
| AND Rd, Rr | 0 < d < 31, 0 < r < 31 | PC ← PC + 1 |

16-разрядный код операции:

| 0010 | 00rd | dddd | rrrr |
|---|---|---|---|

Булевы выражения регистра статуса (SREG):

| I | T | H | S | V | N | Z | C |
|---|---|---|---|---|---|---|---|
| - | - | - | <=> | 0 | <=> | <=> | - |

| | |
|---|---|
| S: | NEV, Для проверок со знаком |
| V: | 0<br>Очищен |
| N: | R7<br>Устанавливается если в результате установлен MSB, в ином случае очищается |
| Z: | R7*R6*R5*R4*R3*R2*R1*R0<br>Устанавливается если результат $00, в ином случае очищается |
| R: | (Результат) соответствует Rd после выполнения команды |

**Пример:**

```
and r2, r3  ; Поразрядное and r2 и r3, результат поместить в r2
ldi r16, 1  ; Установить маску 0000 0001 в r16
and r2, r16 ; Выделить бит 0 в r2
```

**4.8 Команда ANDI** - Выполнить логическое AND с непосредственным значением.

Описание:

Выполнение логического AND между содержимым регистра Rd и константой и помещение результата в регистр назначения Rd.

Операция:

| Rd ← Rd AND K | | |
|---|---|---|
| Синтаксис: | Операнды: | Счетчик программ: |
| ANDI Rd, K | 0 < d < 31, 0 < K <255 | PC ← PC + 1 |

16-разрядный код операции:

| 0111 | KKKK | dddd | KKKK |
|---|---|---|---|

Булевы выражения регистра статуса (SREG):

| I | T | H | S | V | N | Z | C |
|---|---|---|---|---|---|---|---|
| - | - | - | <=> | 0 | <=> | <=> | - |

| | |
|---|---|
| S: | NEV, Для проверок со знаком |
| V: | 0<br>Очищен |
| N: | R7<br>Устанавливается если в результате установлен MSB, в ином случае очищается |
| Z: | R7*R6*R5*R4*R3*R2*R1*R0<br>Устанавливается если результат $00, в ином случае очищается |
| R: | (Результат) соответствует Rd после выполнения команды |

**Пример:**

andi r17, $0F ; Очистить старший ниббл r17

andi r18, $10 ; Выделить бит 4 в r18

andi r19, $AA ; Очистить нечетные биты r19

**4.9 Команда OR** - Выполнить логическое OR

Описание:

Команда выполняет логическое OR содержимого регистров Rd и Rr и размещает результат в регистре назначения Rd.

Операция:

| Rd ← Rd OR Rr | | |
|---|---|---|
| Синтаксис: | Операнды: | Счетчик программ: |
| OR Rd, Rr | 0 < d < 31,0 < r <31 | PC ← PC + 1 |

16-разрядный код операции:

| 0010 | 10rd | dddd | rrrr |
|---|---|---|---|

Булевы выражения регистра статуса (SREG):

| I | T | H | S | V | N | Z | C |
|---|---|---|---|---|---|---|---|
| - | - | - | <=> | 0 | <=> | <=> | - |

| | |
|---|---|
| S: | NEV, Для проверок со знаком |
| V: | 0<br>Очищен |
| N: | R7<br>Устанавливается если в результате установлен MSB, в ином случае очищается |
| Z: | Rd7*R6*R5*R4*R3*R2*R1*R0<br>Устанавливается если результат $00, в ином случае очищается |
| R: | (Результат) соответствует Rd после выполнения команды |

**Пример:**

```
or r15, r16  ; Выполнить поразрядное or между регистрами
bst r15, 6   ; Сохранить бит 6 регистра 15 во флаге T
brst ok      ; Перейти если флаг T установлен
 . . .
ok:  nop     ; Перейти по назначению (пустая операция)
```

**4.10 Команда ORI** - Выполнить логическое OR с непосредственным значением

Описание:

Команда выполняет логическое OR между содержимым регистра Rd и константой и размещает результат в регистре назначения Rd.

Операция:

| Rd ← Rd OR K | | |
|---|---|---|
| Синтаксис: | Операнды: | Счетчик программ: |
| ORI Rd, K | 16 < d < 31,0 < K <255 | PC ← PC + 1 |

16-разрядный код операции:

| | | | |
|---|---|---|---|
| 0110 | KKKK | dddd | KKKK |

Булевы выражения регистра статуса (SREG):

| I | T | H | S | V | N | Z | C |
|---|---|---|---|---|---|---|---|
| - | - | - | <=> | 0 | <=> | <=> | - |

| | |
|---|---|
| S: | NEV, Для проверок со знаком |
| V: | 0<br>Очищен |
| N: | R7<br>Устанавливается если в результате установлен MSB, в ином случае очищается |
| Z: | Rd7*R6*R5*R4*R3*R2*R1*R0<br>Устанавливается если результат $00, в ином случае очищается |
| R: | (Результат) соответствует Rd после выполнения команды |

**Пример:**

```
ori  r16, $F0 ; Установить старший ниббл r16
ori  r17, 1   ; Установить бит 0 регистра r17
```

**4.11 Команда EOR -** Выполнить исключающее OR

Описание:

Выполнение логического исключающего OR между содержимым регистра Rd и регистром Rr и помещение результата в регистр назначения Rd.

Операция:

| Rd← Rd XOR Rr | | |
|---|---|---|
| Синтаксис: | Операнды: | Счетчик программ: |
| EOR Rd, Rr | 0 < d < 31,0 < r < 31 | PC ← PC + 1 |

16-разрядный код операции:

| | | | |
|---|---|---|---|
| 0010 | 01rd | dddd | rrrr |

Булевы выражения регистра статуса (SREG):

| I | T | H | S | V | N | Z | C |
|---|---|---|---|---|---|---|---|
| - | - | - | <=> | 0 | <=> | <=> | - |

| | |
|---|---|
| S: | NEV, Для проверок со знаком |
| V: | 0<br>Очищен |
| N: | R7<br>Устанавливается если в результате установлен MSB, в ином случае очищается |

| | |
|---|---|
| Z: | R7*R6*R5*R4*R3*R2*R1*R0<br>Устанавливается если результат $00, в ином случае очищается |
| R: | (Результат) соответствует Rd после выполнения команды |

**Пример:**

```
eor r4, r4  ; Очистить r4
eor r0, r22 ; Поразрядно выполнить исключающее or между r0 и r22
```

**4.12 Команда COM** - Выполнить дополнение до единицы

Описание:

Команда выполняет дополнение до единицы (реализует обратный код) содержимого регистра Rd.

Операция:

| Rd ← $FF - Rd | | |
|---|---|---|
| Синтаксис: | Операнды: | Счетчик программ: |
| COM Rd | 0 < d < 31 | PC ← PC + 1 |

16-разрядный код операции:

| | | | |
|---|---|---|---|
| 1001 | 010d | dddd | 0000 |

Булевы выражения регистра статуса (SREG):

| I | T | H | S | V | N | Z | C |
|---|---|---|---|---|---|---|---|
| - | - | - | <=> | 0 | <=> | <=> | 1 |

| | |
|---|---|
| S: | NEV, Для проверок со знаком |
| V: | 0<br>Очищен |
| N: | R7<br>Устанавливается если в результате установлен MSB, в ином случае очищается |
| Z: | R7*R6*R5*R4*R3*R2*R1*R0<br>Устанавливается если результат $00, в ином случае очищается |
| C: | 1<br>Установлен |
| R: | (Результат) соответствует Rd после выполнения команды |

**Пример:**

```
        com r4    ; Выполнить дополнение до единицы r4
        breq zero ; Перейти если ноль
                . . .
zero:   nop       ; Перейти по назначению (пустая операция)
```

**4.13 Команда NEG -** Выполнить дополнение до двух

Описание:

Заменяет содержимое регистра Rd его дополнением до двух. Значение $80 остается неизменным.

Операция:

| Rd ← $00 - Rd | | |
|---|---|---|
| Синтаксис: | Операнды: | Счетчик программ: |
| NEG Rd | 0 < d < 31 | PC ← PC + 1 |

16-разрядный код операции:

| 1001 | 010d | dddd | 0001 |
|---|---|---|---|

Булевы выражения регистра статуса (SREG):

| I | T | H | S | V | N | Z | C |
|---|---|---|---|---|---|---|---|
| - | - | <=> | <=> | <=> | <=> | <=> | <=> |

| | |
|---|---|
| H: | R3*Rd3<br>Устанавливается если есть заем из бита 3, в ином случае очищается |
| S: | NEV, Для проверок со знаком |
| V: | R7*R6*R5*R4*R3*R2*R1*R0<br>Устанавливается при переполнении дополнения до двух от подразумеваемого вычитания из нуля, в ином случае очищается. Переполнение дополнения до двух произойдет если и только если содержимое регистра после операции (результат) будет $80. |
| N: | R7<br>Устанавливается если в результате установлен MSB, в ином случае очищается |
| Z: | Rd7*R6*R5*R4*R3*R2*R1*R0<br>Устанавливается если результат $00, в ином случае очищается |
| C: | R7+R6+R5+R4+R3+R2+R1+R0<br>Устанавливается если есть заем в подразумеваемом вычитании из нуля, |

| | |
|---|---|
| | в ином случае очищается. Флаг C будет устанавливаться во всех случаях, за исключением случая, когда содержимое регистра после выполнения операции будет $80. |
| R: | (Результат) соответствует Rd после выполнения команды |

**Пример:**

```
    sub  r11, r0  ; Вычесть r0 из r11
    brpl positive ; Перейти если результат положительный
    neg  r11      ; Выполнить дополнение до двух r11
positive:  nop    ; Перейти по назначению (пустая операция)
```

**4.14 Команда SBR -** Установить биты в регистре

Описание:

Команда выполняет установку определенных битов в регистре Rd. Команда выполняет логическое ORI между содержимым регистра Rd и маской-константой K и размещает результат в регистре назначения Rd.

Операция:

| Rd ← Rd OR K | | |
|---|---|---|
| Синтаксис: | Операнды: | Счетчик программ: |
| SBR Rd,K | 16 < d < 31, 0 < K < 255 | PC ← PC + 1 |

16-разрядный код операции:

| | | | |
|---|---|---|---|
| 0110 | KKKK | dddd | KKKK |

Булевы выражения регистра статуса (SREG):

| I | T | H | S | V | N | Z | C |
|---|---|---|---|---|---|---|---|
| - | - | - | <=> | 0 | <=> | <=> | - |

| | |
|---|---|
| S: | NEV, Для проверок со знаком |
| V: | 0<br>Очищен |
| N: | R7<br>Устанавливается если в результате установлен MSB, в ином случае очищается |
| Z: | R7*R6*R5*R4*R3*R2*R1*R0<br>Устанавливается если результат $00, в ином случае очищается |

| | |
|---|---|
| R: | (Результат) соответствует Rd после выполнения команды |

**Пример:**

```
sbr r16, 3F0 ; Установить биты 0 и 1 в r16
sbr r17, $F0 ; Установить старшие 4 бита в r17
```

**4.15 Команда CBR** - Очистить биты в регистре

Описание:

Очистка определенных битов регистра Rd. Выполняется логическое AND между содержимым регистра Rd и комплементом постоянной K

Операция:

| Rd ← Rd * ($FF - K) | | |
|---|---|---|
| Синтаксис: | Операнды: | Счетчик программ: |
| CBR Rd | 16 < d < 31, 0 < K <255 | PC ← PC + 1 |

16-разрядный код операции - смотри команду ANDI с комплементом K.

Булевы выражения регистра статуса (SREG):

| I | T | H | S | V | N | Z | C |
|---|---|---|---|---|---|---|---|
| - | - | - | <=> | 0 | <=> | <=> | - |

| | |
|---|---|
| S: | NEV, Для проверок со знаком |
| V: | 0<br>Очищен |
| N: | R7<br>Устанавливается если в результате установлен MSB, в ином случае очищается |
| Z: | R7*R6*R5*R4*R3*R2*R1*R0<br>Устанавливается если результат $00, в ином случае очищается |
| R: | (Результат) соответствует Rd после выполнения команды |

**Пример:**

```
cbr r16, $F0 ; Очистить старший ниббл регистра r16
cbr r18, 1   ; Очистить бит в r18
```

**4.16 Команда INC** - Инкрементировать

Описание:

Добавление единицы - 1 - к содержимому регистра Rd и размещение результата в регистре назначения Rd. Флаг переноса регистра статуса данной командой не активируется, что позволяет использовать команду DEC использовать при реализации счетчика циклов для вычислений с повышенной точностью. При обработке чисел без знаков за командой могут выполняться переходы BREQ и BRNE. При обработке значений в форме дополнения до двух допустимы все учитывающие знак переходы.

Операция:

| Rd ← Rd + 1 | | |
|---|---|---|
| Синтаксис: | Операнды: | Счетчик программ: |
| INC Rd | 0 < d < 31 | PC ← PC + 1 |

16-разрядный код операции:

| | | | |
|---|---|---|---|
| 1001 | 010d | dddd | 0011 |

Булевы выражения регистра статуса (SREG):

| I | T | H | S | V | N | Z | C |
|---|---|---|---|---|---|---|---|
| - | - | - | <=> | <=> | <=> | <=> | - |

| | |
|---|---|
| S: | NEV, Для проверок со знаком |
| V: | R7*R6*R5*R4*R3*R2*R1*R0<br>Устанавливается если в результате получено переполнение дополнения до двух, в ином случае очищается. Переполнение дополнения до двух будет если и только если перед операцией содержимое Rd было $7F. |
| N: | R7<br>Устанавливается если в результате установлен MSB, в ином случае очищается |
| Z: | R7*R6*R5*R4*R3*R2*R1*R0<br>Устанавливается если результат $00, в ином случае очищается |
| R: | (Результат) соответствует Rd после выполнения команды |

**Пример:**

```
        clr r22      ; Очистить r22
loop:   inc r22      ; Увеличить на 1 r22
                . . .
        cpi r22, $4F ; Сравнить r22 с $4F
        brne loop    ; Перейти если не равно
        nop          ; Продолжать (пустая операция)
```

**4.17 Команда DEC** - Декрементировать

Описание:

Вычитание единицы - 1 - из содержимого регистра Rd и размещение результата в регистре назначения Rd. Флаг переноса регистра статуса данной командой не активируется, что позволяет использовать команду DEC использовать при реализации счетчика циклов для вычислений с повышенной точностью. При обработке чисел без знаков за командой могут выполняться переходы BREQ и BRNE. При обработке значений в форме дополнения до двух допустимы все учитывающие знак переходы.

Операция:

| Rd ← Rd - 1 | | |
|---|---|---|
| Синтаксис: | Операнды: | Счетчик программ: |
| DEC Rd | 0 < d < 31 | PC ← PC + 1 |

16-разрядный код операции:

| | | | |
|---|---|---|---|
| 1001 | 010d | dddd | 1010 |

Булевы выражения регистра статуса (SREG):

| I | T | H | S | V | N | Z | C |
|---|---|---|---|---|---|---|---|
| - | - | - | <=> | <=> | <=> | <=> | - |

| | |
|---|---|
| S: | NEV, Для проверок со знаком |
| V: | R7*R6*R5*R4*R3*R2*R1*R0<br>Устанавливается если в результате получено переполнение дополнения до двух, в ином случае очищается. Переполнение дополнения до двух будет если и только если перед операцией содержимое Rd было $80. |

| | |
|---|---|
| N: | R7<br>Устанавливается если в результате установлен MSB, в ином случае очищается |
| Z: | R7*R6*R5*R4*R3*R2*R1*R0<br>Устанавливается если результат $00, в ином случае очищается |
| R: | (Результат) соответствует Rd после выполнения команды |

**Пример:**

```
        ldi r17, $10 ; Загрузить константу в r17
loop:   add r1, r2   ; Сложить r2 с r1
        dec r17      ; Уменьшить на 1 r17
        brne loop    ; Перейти если r17 <> 0
        nop          ; Продолжать (пустая операция)
```

**4.18 Команда TST** - Проверить на ноль или минус

Описание:

Регистр проверяется на нулевое или отрицательное состояние. Выполняется логическое AND содержимого регистра с самим собой. Содержимое регистра остается неизменным.

Операция:

| Rd ← Rd AND Rd | | |
|---|---|---|
| Синтаксис: | Операнды: | Счетчик программ: |
| TST Rd | 0 < d < 31 | PC ← PC + 1 |

16-разрядный код операции:

| | | | |
|---|---|---|---|
| 0010 | 00dd | dddd | dddd |

Булевы выражения регистра статуса (SREG):

| I | T | H | S | V | N | Z | C |
|---|---|---|---|---|---|---|---|
| - | - | - | <=> | 0 | <=> | <=> | - |

| | |
|---|---|
| S: | NEV, Для проверок со знаком |
| V: | 0<br>Очищен |
| N: | R7 |

| | |
|---|---|
| | Устанавливается если в результате установлен MSB, в ином случае очищается |
| Z: | R7*R6*R5*R4*R3*R2*R1*R0<br>Устанавливается если результат $00, в ином случае очищается |
| R: | (Результат) соответствует Rd |

**Пример:**

```
    tst  r0   ; Проверить r0
    breq zero ; Перейти если r0 = 0
    . . .
zero:  nop    ; Перейти по назначению (пустая операция)
```

**4.19 Команда CLR** - Очистить регистр

Описание:

Очистка регистра. Команда выполняет Exclusive OR содержимого регистра с самим собой. Это приводит к очистке всех битов регистра.

Операция:

| Rd ← Rd XOR Rd | | |
|---|---|---|
| Синтаксис: | Операнды: | Счетчик программ: |
| CLR Rd | 0 < d < 31 | PC ← PC + 1 |

16-разрядный код операции:

| | | | |
|---|---|---|---|
| 0010 | 01dd | dddd | dddd |

Булевы выражения регистра статуса (SREG):

| I | T | H | S | V | N | Z | C |
|---|---|---|---|---|---|---|---|
| - | - | - | 0 | 0 | 0 | 1 | - |

| | |
|---|---|
| S: | 0<br>Очищен |
| V: | 0<br>Очищен |
| N: | 0<br>Очищен |
| Z: | 1<br>Устанавливается |

| R: | (Результат) соответствует Rd после выполнения команды |
|---|---|

**Пример:**

```
      clr r18      ; Очистить r18
loop:  inc r18     ; Увеличить на 1 r18
                . . .
      cpi r18, $50 ; Сравнить r18 с $50
      brne loop
```

**4.20 Команда SER** - Установить все биты регистра

Описание:

Значение $FF заносится непосредственно в регистр назначения Rd.

Операция:

| Rd ← $FF | | |
|---|---|---|
| Синтаксис: | Операнды: | Счетчик программ: |
| SER Rd | 0 < d < 31 | PC ← PC + 1 |

16-разрядный код операции:

| 1110 | 1111 | dddd | 1111 |
|---|---|---|---|

Булевы выражения регистра статуса (SREG):

| I | T | H | S | V | N | Z | C |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |

**Пример:**

```
  clr  r16     ; Очистить r16
  ser  r17     ; Установить r17
  out  #18, r16 ; Записать нули в Порт B
  nop          ; Задержка (пустая операция)
  out  #18, r17 ; Записать единицы в Порт B
```

**4.21 Команда ADIW** - Сложить непосредственное значение со словом

Описание:

Сложение непосредственного значения (0-63) с парой регистров и размещение результата в паре регистров. Команда работает с четырьмя верхними парами регистров, удобна для работы с регистрами указателями.

Операция:

| Rdh:Rdl ← Rdh:Rdl + K | | |
|---|---|---|
| Синтаксис: | Операнды: | Счетчик программ: |
| ADIW Rdl, K | dl □ {24,26,28,30}, 0 < K < 63 | PC ← PC + 1 |

16-разрядный код операции:

| 1001 | 0110 | KKdd | KKKK |
|---|---|---|---|

Булевы выражения регистра статуса (SREG):

| I | T | H | S | V | N | Z | C |
|---|---|---|---|---|---|---|---|
| - | - | - | <=> | <=> | <=> | <=> | <=> |

| | |
|---|---|
| S: | NEV, Для проверок со знаком |
| V: | Rdh7 R15<br>Устанавливается если в результате операции образуется переполнение дополнения до двух, в ином случае очищается |
| N: | R15<br>Устанавливается если в результате установлен MSB, в ином случае очищается |
| Z: | R15*R14*R13*R12*R11*R10*R9*R8*R7*R6*R5*R4*R3*R2<br>Устанавливается если результат $0000, в ином случае очищается |
| C: | R15*Rdh7<br>Устанавливается если есть перенос из MSB результата, в ином случае очищается |
| R: | (Результат) соответствует Rdh:Rdl после выполнения команды (Rdh7-Rdh0 = R15-R8, Rdl7-Rdl0 = R7-R0) |

**Пример:**

```
adiw r24, 1  ; Сложить 1 с r25:r24
adiw r30, 63 ; Сложить 63 с Z указателем (r31 : r30)
```

**4.22 Команда SBIW** - Вычесть непосредственное значение из слова

Описание:

Вычитание непосредственного значения (0-63) из пары регистров и размещение результата в паре регистров. Команда работает с четырьмя верхними парами регистров, удобна для работы с регистрами указателями.

Операция:

| Rdh:Rdl ← Rdh:Rdl - K | | |
|---|---|---|
| Синтаксис: | Операнды: | Счетчик программ: |
| SBIW Rdl, K | dl □ {24,26,28,30}, 0 < K < 63 | PC ← PC + 1 |

16-разрядный код операции:

| 1001 | 0111 | KKdd | KKKK |
|---|---|---|---|

Булевы выражения регистра статуса (SREG):

| I | T | H | S | V | N | Z | C |
|---|---|---|---|---|---|---|---|
| - | - | - | <=> | <=> | <=> | <=> | <=> |

| | |
|---|---|
| S: | NEV, Для проверок со знаком |
| V: | Rdh7 R15<br>Устанавливается если в результате операции образуется переполнение дополнения до двух, в ином случае очищается |
| N: | R15<br>Устанавливается если в результате установлен MSB, в ином случае очищается |
| Z: | R15*R14*R13*R12*R11*R10*R9*R8*R7*R6*R5*R4*R3*R2*R1*R0<br>Устанавливается если результат $0000, в ином случае очищается |
| C: | R15*Rdh7<br>Устанавливается если абсолютное значение константы K больше абсолютного значения содержимого регистра Rd, в ином случае очищается |
| R: | (Результат) соответствует Rdh:Rdl после выполнения команды (Rdh7-Rdh0 = R15-R8, Rdl7-Rdl0 = R7-R0) |

**Пример:**

```
sbiw  r24, 1  ; Вычесть 1 из r25:r24
sbiw  r28, 63 ; Вычесть 63 из Y указателя (r29 : r28)
```

**4.23 Команда CP** - Сравнить

Описание:

Команда выполняет сравнение содержимого двух регистров Rd и Rr. Содержимое регистров не изменяется. После этой команды можно выполнять любые условные переходы.

Операция:

| Rd = Rr | | |
|---|---|---|
| Синтаксис: | Операнды: | Счетчик программ: |
| Cp Rd, Rr | 0 < d < 31,0 < r < 31 | PC ← PC + 1 |

16-разрядный код операции:

| 0001 | 01rd | dddd | rrrr |
|---|---|---|---|

Булевы выражения регистра статуса (SREG):

| I | T | H | S | V | N | Z | C |
|---|---|---|---|---|---|---|---|
| - | - | <=> | <=> | <=> | <=> | <=> | <=> |

| | |
|---|---|
| H: | Rd3*Rr3+Rr3*R3+R3*Rd3<br>Устанавливается если есть заем из бита 3, в ином случае очищается |
| S: | NEV, Для проверок со знаком |
| V: | Rd7*Rd7*R7+Rd7*Rr7*R7<br>Устанавливается если в результате операции образуется переполнение дополнения до двух, в ином случае очищается |
| N: | R7<br>Устанавливается если в результате установлен MSB, в ином случае очищается |
| Z: | Rd7*Rr7+Rr7*R7+R7*Rd7<br>Устанавливается если результат $00, в ином случае очищается |
| C: | Rd7*Rr7+Rr7*R7+R7*Rd7<br>Устанавливается если абсолютное значение Rr больше абсолютного значения Rd, в ином случае очищается |
| R: | (Результат) после выполнения команды |

**Пример:**

```
        cp r4, r19 ; Сравнить r4 с r19
        brne noteq ; Перейти если r4 <> r19
                . . .
noteq:  nop        ; Перейти по назначению (пустая операция)
```

**4.24 Команда CPC** - Сравнить с учетом переноса

Описание:

Команда выполняет сравнение содержимого двух регистров Rd и Rr и учитывает также предшествовавший перенос. Содержимое регистров не изменяется. После этой команды можно выполнять любые условные переходы.

Операция:

| Rd = Rr = C | | |
|---|---|---|
| Синтаксис: | Операнды: | Счетчик программ: |
| CPC Rd, Rr | 0 < d < 31,0 < r < 31 | PC ← PC + 1 |

16-разрядный код операции:

| 0000 | 01rd | dddd | rrrr |
|---|---|---|---|

Булевы выражения регистра статуса (SREG):

| I | T | H | S | V | N | Z | C |
|---|---|---|---|---|---|---|---|
| - | - | <=> | <=> | <=> | <=> | <=> | <=> |

| | |
|---|---|
| H: | Rd3*Rr3+Rr3*R3+R3*Rd3<br>Устанавливается если есть заем из бита 3, в ином случае очищается |
| S: | NEV, Для проверок со знаком |
| V: | Rd7*Rd7*R7+Rd7*Rr7*R7<br>Устанавливается если в результате операции образуется переполнение дополнения до двух, в ином случае очищается |
| N: | R7<br>Устанавливается если в результате установлен MSB, в ином случае очищается |
| Z: | R7*R6*R5*R4*R3*R2*R1*R0*Z<br>Предшествующее значение остается неизменным если результатом является ноль, в ином случае очищается |
| C: | Rd7*Rr7+Rr7*R7+R7*Rd7<br>Устанавливается если абсолютное значение Rr плюс предшествовавший перенос больше абсолютного значения Rd, в ином случае очищается |
| R: | (Результат) после выполнения команды |

**Пример:**

```
                ; Сравнить r3 : r2 с r1 : r0
        cp r2, r0  ; Сравнить старший байт
        cpc r3, r1 ; Сравнить младший байт
        brne noteq ; Перейти если не равно
                . . .
noteq:  nop        ; Перейти по назначению (пустая операция
```

**4.25 Команда CPI** - Сравнить с константой

Описание:

Команда выполняет сравнение содержимого регистра Rd с константой. Содержимое регистра не изменяется. После этой команды можно выполнять любые условные переходы.

Операция:

| Rd = K | | |
|---|---|---|
| Синтаксис: | Операнды: | Счетчик программ: |
| CPI Rd, K | 0 < d < 31, 0 < K <255 | PC ← PC + 1 |

16-разрядный код операции:

| 0011 | KKKK | dddd | KKKK |
|---|---|---|---|

Булевы выражения регистра статуса (SREG):

| I | T | H | S | V | N | Z | C |
|---|---|---|---|---|---|---|---|
| - | - | <=> | <=> | <=> | <=> | <=> | <=> |

| | |
|---|---|
| H: | Rd3*Rr3+Rr3*R3+R3*Rd3<br>Устанавливается если есть заем из бита 3, в ином случае очищается |
| S: | NEV, Для проверок со знаком |
| V: | Rd7*K7*R7+ Rd7*K7*R7<br>Устанавливается если в результате операции образуется переполнение дополнения до двух, в ином случае очищается |
| N: | R7<br>Устанавливается если в результате установлен MSB, в ином случае очищается |
| Z: | R7*R6*R5*R4*R3*R2*R1*R0<br>Устанавливается если результат $00, в ином случае очищается |
| C: | Rd7*K7+K7*R7+R7*Rd7<br>Устанавливается если абсолютное значение K больше абсолютного значения Rd, в |

| | |
|---|---|
| | ином случае очищается |
| R: | (Результат) после выполнения команды |

**Пример:**

```
        cpi r19, 3 ; Сравнить r19 с 3
        brne error ; Перейти если r4 <> 3
                . . .
error:  nop        ; Перейти по назначению (пустая операция)
```

При описании арифметических и логических команд AVR – микроконтроллеров использовались следующие обозначения:

**Регистр статуса (SREG):**

| | |
|---|---|
| **SREG**: | Регистр статуса |
| **C**: | Флаг переноса |
| **Z**: | Флаг нулевого значения |
| **N**: | Флаг отрицательного значения |
| **V**: | Флаг-указатель переполнения дополнения до двух |
| **S**: | NV, Для проверок со знаком |
| **H**: | Флаг полупереноса |
| **T**: | Флаг пересылки, используемый командами BLD и BST |
| **I**: | Флаг разрешения/запрещения глобального прерывания |

**Регистры и операнды:**

| | |
|---|---|
| **Rd**: | Регистр назначения (и источник) в регистровом файле |
| **Rr**: | Регистр источник в регистровом файле |
| **R**: | Результат выполнения команды |
| **K**: | Литерал или байт данных (8 бит) |
| **k**: | Данные адреса константы для счетчика программ |
| **b**: | Бит в регистровом файле или I/O регистр (3 бита) |
| **s**: | Бит в регистре статуса (3 бита) |
| **X, Y, Z**: | Регистр косвенной адресации (X=R27:R26, Y=R29:R28, Z=R31:R30) |
| **P**: | Адрес I/O порта |
| **q**: | Смещение при прямой адресации (6 бит) |

**I/O регистры:**

| | |
|---|---|
| **RAMPX, RAMPY, RAMPZ**: | Регистры связанные с X, Y и Z регистрами, обеспечивающие косвенную адресацию всей области СОЗУ микроконтроллера с объемом СОЗУ более 64 Кбайт |

**Стек:**

| | |
|---|---|
| **STACK**: | Стек для адреса возврата и опущенных в стек регистров |
| **SP**: | Указатель стека |

**Флаги:**

| | |
|---|---|
| <=> | Флаг, на который воздействует команда |
| **0**: | Очищенный командой Флаг |
| **1**: | Установленный командой флаг |
| **-**: | Флаг, на который не воздействует команда |

# МЕТОДИЧЕСКОЕ ПОСОБИЕ ПО МИКРОПРОЦЕССОРАМ

**Кн. 3.**

**БОРИКОВ ВАЛЕРИЙ НИКОЛАЕВИЧ**

# ПРОГРАММИРОВАНИЕ ВЫЧИСЛИТЕЛЬНЫХ ФУНКЦИЙ AVR - МИКРОКОНТРОЛЛЕРОВ

**УЧЕБНОЕ ПОСОБИЕ ДЛЯ ПРОВЕДЕНИЯ ПРАКТИЧЕСКИХ ЗАНЯТИЙ**