### **Лабораторная работа №3. Арифметическая обработка данных**

*Цель работы:*

− изучение способов представления числовых данных в микроконтроллерах,

− изучение двоичных арифметических операций,

− программирование арифметических процедур.

**Представление двоичных чисел в микроконтроллерах**

При обработке числовой информации в микроконтроллерах обычно полагают, что целые числа имеют формат с фиксированной точкой справа − *D = dn-*1*dn-*2*…d*1*d*0*,* дробные числа имеют формат с точкой слева − *D = d-*1*d-*2*…d-(n-*1*)d-n,* где *n* – число разрядов. Обрабатываемые числа могут быть числами со знаком и без знака.

При целочисленном представлении разряд *d*0 – младший разряд целого числа с весом 20, старший разряд *dn*-1 используется для представления знака числа (0 – положительный, 1 – отрицательный). Старший цифровой разряд – разряд  *dn*-2 с весом 2*n*-2. При обработке чисел без знака разряд *dn-*1 является цифровым с весом 2*n*-1 .

Пример 1:

а) прямой двоичный код положительного числа А= +15: [+15]пр = 0(знак)0001111;

б) прямой двоичный код отрицательного числа А= –112: [–112]пр = 1(знак)1110000.

Целые отрицательные числа обычно представляют в виде дополнений до 2*n*. Диапазон представления целых двоичных чисел со знаком в дополнительном коде в n-разрядном формате составляет от – 2(*n*-1) до + (2(*n-*1) –1).

Пример 2:

а) отрицательное число –112 в дополнительном коде: [–112] доп = 28 – ǀAǀ = 10010000;

б) отрицательное число –128 в дополнительном коде: [–128] доп = 10000000.

В общем случае дополнение любого целого *n*-разрядного числа *D* с основанием *b* системы счисления можно получить путем вычитания *D* из *bn*. Если *D* находится в пределах от 1 до *bn* –1, то при вычитании получается другое число в тех же пределах. Если *D*=0, то результат вычитания равен *bn* и имеет вид 100…0 при общем числе разрядов, равном (*n*+1). Отбросив цифру старшего разряда, получим 0. Следовательно, в системе представления чисел дополнением до *bn* существует только одно представление 0. В системе, где целые отрицательные числа представлены в дополнительном коде, число является положительным, если значение старшего разряда *dn*-1 = 0, и отрицательным, если *dn*-1 = 1. Десятичный эквивалент двоичного числа, представленного дополнительным кодом, вычисляется так же, как и для числа без знака, за исключением того, что вес старшего разряда принимается равным –2(*n*–1), а не +2(*n*–1).

В процессорах с регистр-регистровой архитектурой операнды перед началом операции размещаются в регистрах общего назначения, а результат операции помещается на место одного их операндов. Таким образом, операнды и результат операции имеют одинаковую разрядность, что необходимо учитывать при оценке и интерпретации результатов выполнения операций, привлекая при необходимости сопровождающие операцию признаки результата (флаги операции).

Загрузку операндов в регистры осуществляют либо командой загрузки регистра указанной в команде константой (**LDI Rd, k**), либо путем пересылки (например, ввод с порта **IN Rd, PINx** или загрузка из ячейки памяти). При этом константа **k**, представляющая отрицательное десятичное число, загружается в дополнительном коде.

Пример 3. При выполнении команды загрузки **LDI Rd, -15** в регистре Rd будет получен двоичный код [–15]доп=0b11110001. Тот же результат будет при выполнении команды **LDI Rd, 241**.

Как видно из примера содержимое регистра приемника одинаково представляет число со знаком (–15) и число без знака (255).

Что касается непосредственно самих операций обработки данных, выполняемых процессором, необходимо иметь в виду, что инструкции сложения **ADD** и вычитания **SUB** не делают различий между операндами (знаковые или беззнаковые), обрабатывая их одинаковым образом согласно заданному коду операции. Это означает, что числа со знаком в дополнительном коде складываются и вычитаются процессором так же, как числа без знака той же длины. Что касается оценки (интерпретации) значений исходных операндов и результатов вычислений, эта функция целиком возлагается на пользователя программы.

Для оценки исходных операндов и результатов операции можно воспользоваться следующими правилами:

а) для перевода целого положительного (или беззнакового) двоичного числа в десятичный эквивалент используется формула подсчета суммы:

S = ∑аi\*2i, (i=0,..n-1),

где аi – значение i-го двоичного разряда, \*– знак умножения, n – количество двоичных разрядов.

б) для перевода целого отрицательного двоичного числа из дополнительного кода в десятичный эквивалент можно найти дополнение числа и подсчитать затем сумму со знаком минус:

S = –∑аi\*2i , (i=0,..n-1)

в) представив целое отрицательное двоичное число в виде суммы двух слагаемых 0b10…00 и 0b0a(n-2)…а1a0 можно также воспользоваться формулой:

S = –2 (n-1) + ∑аi\*2i, (i=0,..n-2)

Пример 4: а) десятичный эквивалент положительного двоичного числа 0b01101010 = +106;

б) для оценки отрицательного двоичного числа [А]доп = 0b10000111 сначала находим дополнение 0b01111001, затем подсчитываем сумму со знаком минус –121;

в) двоичное число [А]доп = 0b10000111 можно представить в виде суммы 0b10000111 = 0b10000000 + 0b00000111 = –128 +7 = –121.

### **Сложение и вычитание двоичных чисел**

При сложении А+В оба слагаемых из регистров хранения поступают непосредственно на входы сумматора процессора. На выходах сумматора образуется сумма, сохраняемая в регистре операнда А, а в регистре признаков фиксируется значение переноса на выходе старшего разряда сумматора (флаг ***С=р8***). Как уже было сказано, операнды могут быть представлены числами со знаком или без знака.

Для характеристики суммы беззнаковых чисел используют признаки наличия/отсутствия переноса (С) и признак нулевого результата (Z), для суммы знаковых чисел добавляются признаки знака суммы N (старший знаковый разряд суммы) и признак переполнения V.

При вычитании А–В сигнал операции вычитания с дешифратора кода операции процессора используется для передачи в сумматор обратного кода вычитаемого /B и прибавления единицы на вход переноса младшего разряда сумматора (***p0*=1**) для получения дополнительного кода вычитаемого ([B]доп=[B]обр+1), как того требует алгоритм вычитания с помощью сумматора. Таким образом, вычитание заменяется сложением уменьшаемого с дополнительным кодом вычитаемого (А+[B]доп). На выходах сумматора образуется разность, сохраняемая в регистре уменьшаемого А, а в регистре признаков фиксируется значение заема (***С=****/****р*8**). По аналогии с операцией сложения операнды могут быть представлены числами со знаком или без знака.

Для характеристики разности беззнаковых чисел используют как и при сложении признак наличия/отсутствия заема и признак нуля, для знаковых чисел добавляются признаки знака разности N и переполнения V.

Графически 8-разрядные двоичные (2-разрядные шестнадцатеричные) числа со знаком в дополнительном коде показаны на круговой диаграмме (рис. 1,а) точками (внутри круга указаны их десятичные значения, снаружи – шестнадцатеричные изображения). Сложение с положительным числом N легко интерпретировать, перемещая числовой указатель по ходу часовой стрелки на N позиций; вычитание – против хода часовой стрелки или перемещая по ходу часовой стрелки на (256–N) позиций, что равносильно замене вычитания сложением с дополнением числа до 28 = 256. Если при сложении (или вычитании) происходит переход через условную границу УГ (–128 ÷ +127), фиксируется признак переполнения.

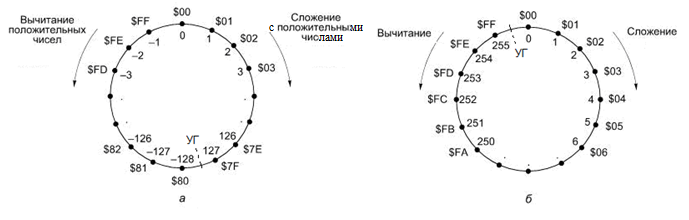


Рис. 1. Круговые диаграммы

а) для чисел со знаком в диапазоне –128 ÷ + 127, б) для чисел без знака в диапазоне 0 ÷ 255

*Правило выявления переполнения*.

При сложении переполнение возникает только в том случае, если слагаемые имеют одинаковые знаки, а знак суммы отличается от знака слагаемых. Правило переполнения можно сформулировать иначе, используя понятие переносов, возникающих при сложении двоичных кодов. Переполнение *V* возникает, если значения переносов в знаковый разряд *p*7 и из знакового разряда *p*8 различны (***V*= *p*8 ⊕ *p*7**). Из анализа рис. 1,а следует, что переполнение при сложении с положительными числами возникает, если указатель перейдет условную границу между позициями +127 и –128 в направлении по часовой стрелке (положительное переполнение), при сложении отрицательных чисел (равносильно вычитанию положительных чисел) при пересечении условной границы в направлении против часовой стрелки (отрицательное переполнение).

При вычитании переполнение происходит, если операнды (уменьшаемое и вычитаемое) имеют разные знаки, а знак разности отличается от знака уменьшаемого. А поскольку операция вычитания в процессоре сводится к операции сложения уменьшаемого с вычитаемым (без учета знаков операндов), представленным дополнением (*А – В = А + [*–*B]*доп), то правило выявления переполнения (***V*= *p*8 ⊕ *p*7**) остаётся таким же, как при операции сложения. В графической интерпретации на рис. 1,а переполнение возникает при вычитании по аналогии со сложением, если указатель при перемещении пересечет условную границу между позициями +127 и –128 в том или ином направлении.

Пример 5. Даны два 8-разрядных двоичных числа со знаком:

[A]доп = 0b1001 0100, В = 0b0111 0001.

Найти сумму (А+В) и разность (А – В). Определить значение флага переполнения V для каждой операции.

a) Cумма А + В: 1001 0100 б) Разность А - В: 1001 0100

+ 0111 0001 + 1000 1111

1← **0000 0101** 1← **0010 0011**

Переносы *p***8**=1, *р***7**=1. Переносы *р***8**=1, *р***7**=0.

Флаги *С=*1*,* *V*=0 (переполнения нет). Флаги *С=*0*,* *V*=1 (переполнение есть),

N (знак)=0 -> (сумма > 0). N (знак)=0 -> (разность > 0).

Проверка: (–108) + (+113) = ( +5) Проверка: (–108) – (+113) ≠ (+35)

Сумма в 8-разрядном формате верна. Значение разности в 8-разрядном формате ошибочно из-за переполнения.

На рис. 1,б представлены 8–разрядные двоичные (2-разрядные шестнадцатеричные) числа без знака и их десятичные представления. Видно, что двоичные кодовые комбинации занимают те же позиции, что и на рис. 1,а, а сложение и вычитание можно осуществить, перемещая указатель на N позиций в том или ином направлении. При сложении чисел без знака результат выходит за пределы диапазона представления при пересечении условной границы между 255 и 0. В этом случае говорят о возникновении переноса из старшего разряда *р***8**, а признак результата операции сложения представляют переносом *С*=*р***8**=1. При вычитании чисел без знака при пересечении границы между 0 и 255 возникает заём, а разность получается в дополнительном коде. Но так как вычитание N можно заменить сложением с дополнительным кодом числа N, равным (256–N), то из диаграммы видно, что заём возникает без образования переноса из старшего разряда (*р***8**=0). Тот же вывод следует при выполнении операции в машинном коде.

Пример 6. Даны два шестнадцатеричных (двоичных) кода, представляющие беззнаковые числа: A = $05 =0b0000 0101, В = $07 =0b0000 0111.

Вычитая из шестнадцатеричного числа А = $05 число В = $07, имеем:

А – В = А + [–B]доп = $05 – $07 = $05 + $F8(обратный код В) +1 = $FE = –2

или в двоичном коде: 0000 0101

+ 1111 1000

0000 0001

1111 1110 = [–2]доп

Машинного переноса (при сложении двоичных кодов) из старшего разряда нет (*p*8 = 0). При этом признак результата операции *С* представляют заёмом (*С* = /*p*8 = 1), определяемый по отсутствию переноса при выполнении операции вычитания. (Прим. Альтернативно значение флага заема можно определить в виде разности ***С=*1*– p*8**).

И наоборот, вычитая из числа В = $07 число А = $05, имеем:

В – А = В + [–А]доп = $07 – $05 = $07 + $FА(обратный код А) +1 = $102 = $100 (перенос) + $02 = 2

или в двоичном коде: 0000 0111

+ 1111 1010

0000 0001

1← 0000 0010= 2

Перенос *p*8 =1, что при вычитании соответствует отсутствию признака заёма(*С* = /*p*8 = 0).

### **Умножение чисел без знака**

В практике программирования микроконтроллеров для выполнения операции умножения *С=А*×*В* часто используют методы умножения путем сложения ряда частичных произведений *C*= ∑ 2*iАbi*, где *bi* - значение разряда множителя (*i* = 0,1,...*n*-1). Один из алгоритмов умножения, начиная с младших разрядов множителя, со сдвигом вправо суммы частичных произведений приведен на рис. 2.

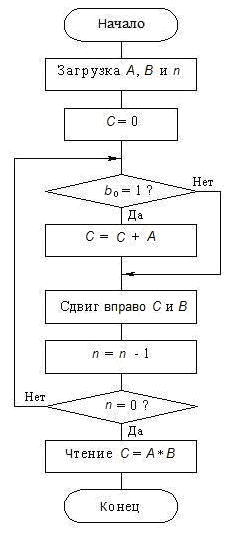


Рис. 2. Схема алгоритма умножения, начиная с младших разрядов множителя

Этот алгоритм может быть использован для получения произведения двух двоичных чисел без знака. Количество итераций умножения *n*определяется числом разрядов множителя. Поскольку в процессе умножения на каждой итерации осуществляется сдвиг множителя *В* на один разряд вправо, на место освобождаемого разряда можно записать выталкиваемый при сдвиге вправо разряд произведения *C*. Таким образом, 2*n*-разрядное произведение можно получить, объединив содержимое *n*-разрядного регистра, в котором формируется старшая часть произведения, и регистра множителя *B*, в котором после выполнения умножения окажется младшая часть произведения.

Микроконтроллеры ATmega8515 выполняют шесть операций умножения: беззнаковых чисел MUL, чисел со знаком MULS, беззнакового числа на число со знаком MULSU, дробных беззнаковых чисел FMUL, дробных со знаком FMULS, дробного беззнакового и дробного со знаком FMULSU.

Приведем ряд примеров на умножение (С = А×В).

MUL – умножение целых беззнаковых чисел

1) А = 240 = 1111 00002,

В = 225 = 1110 00012,

С = 54 000 = 1101 0010 1111 00002.

MULS – умножение целых чисел со знаком

2) А = –10 = 1111 01102,

В = + 10 = 0000 10102,

С = –100 = 1111 1111 1001 11002.

3) А = –10 = 1111 01102,

В = –56 = 1100 10002,

С = + 560 = 0000 0010 0011 00002.

MULSU – умножение числа со знаком (А) на число без знака (В)

4) А = –10 = 1111 01102,

В = 200 = 1100 10002,

С = –2000 = 1111 1000 0011 00002.

5) А = –1 = 1111 11112,

В = 255 = 1111 11112,

С = –255 = 1111 1111 0000 00012.

Для умножения 2-байтовых беззнаковыых сомножителей *A* (*AH*:*AL*) и *B* (*BH*:*BL*) примем во внимание:

A×B= (28*AH*+*AL*) (28*BH*+*BL*) = (*AL*×*BL*) + (28*AH*×*BL*) + (28*AL*×*BH*) + (216*AH*×*BH*).

Из этого следует, что алгоритм умножения можно представить как последовательность из четырех операций умножений (*MUL* *AL*,*BL*; *MUL AH*,*BL*; *MUL AL*,*BH*; *MUL AH*,*BH*) с последующим суммированием взвешенных частичных произведений. (Прим. – Для сомножителей, представленных со знаком, для перемножения потребуются операции: MUL, MULSU, MULS при использовании специальной схемы сложения).

**Деление целых чисел**

Для типичного алгоритма целочисленного деления *C* = *A*/*B* делимым является двойное слово *AH*:*AL* (два байта), а делителем – одинарное *B* (один байт); частное *C* и остаток получают в виде одинарных слов. При выполнении деления необходимо исключить возможность деления на 0. Если для представления частного потребуется более одного слова, то фиксируется переполнение. Перед выполнением деления необходимо проверить условие – делитель должен быть больше старшего слова делимого (*B* > *AH*).

При делении целых чисел можно использовать алгоритм деления без восстановления остатка и алгоритм с восстановлением остатка.

Схема алгоритма с восстановление остатка приведена на рис. 3. Алгоритм деления представляет собой итерационную процедуру. На каждой итерации сначала удваивается делимое (на первой итерации) или остаток (на всех последующих) путем сдвига влево на один разряд, затем вычитается делитель и определяется цифра частного по знаку разности. Если разность положительная, определяемая на данной итерации цифра частного *ci* =1, если разность отрицательная, цифра частного *ci* =0. Восстановление остатка выполняется путем сложения делителя с остатком после вычитания на текущей итерации деления. Деление выполняется до получения всех цифр частного.

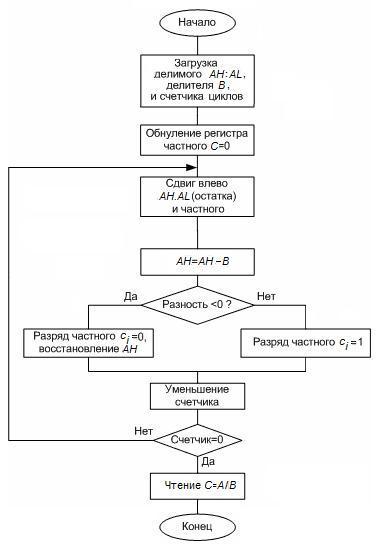


Рис. 3. Схема алгоритма деления с восстановлением остатка

Алгоритм деления без восстановления остатка представляет собой итерационную процедуру, на каждой итерации которой производится либо вычитание делителя *В*, заменяемое сложением с дополнительным кодом [–*B*]доп, либо прибавление *В*, в зависимости от знака остатка, полученного на предыдущей итерации деления. Если полученный остаток больше или равен 0, при очередной итерации деления производится вычитание *В*; если остаток меньше 0 − прибавление *В*. Перед каждым вычитанием (или сложением) остаток удваивается путем сдвига влево. На начальной итерации деления делимое сдвигается на один разряд влево.

Деление чисел со знаком можно выполнить разными способами. Если исходные операнды заданы в прямых кодах, то путем сложения по модулю 2 знаковых разрядов можно определить знак частного. Модули делимого и делителя можно разделить, используя один из вышеописанных алгоритмов. Для выяснения переполнения необходимо выполнить пробное вычитание *A*–2(*n*-1) *B*, резервируя один разряд *n*-разрядного частного для знака.

Ниже приведен пример деления 16-разрядного числа *А* на 8-разрядное число *В* с восстановлением остатка.

*А* = 1024 = 00000100.00000000, *В* = 10 = 00001010,

–*B* = [–10]доп = 11110110,

*С* = *с*7*с*6*с*5*с*4*с*3*с*2*с*1*с*0 – частное, x – бит, свободно определяемый при сдвиге.

### 00000100.00000000 делимое *А* (*AH*.*AL*)

11110110 пробное вычитание *В*

11111010 так как разность меньше 0, переполнения нет

00001000.0000000x сдвиг *А* влево

11110110 вычитание *В*

11111110 1-й остаток меньше 0, разряд частного *с*7 = 0

00010000.000000xx сдвиг влево восстановленного *AH*

11110110 вычитание *В*

00000110 2-й остаток больше 0, разряд частного *с*6 =1

00001100.00000xxx сдвиг остатка

11110110 вычитание *В*

00000010 3-й остаток, *с*5 = 1

00000100.0000xxxx сдвиг остатка

11110110 вычитание *В*

11111010 4-й остаток, *с*4 = 0

00001000.000xxxxx сдвиг восстановленного *АH*

11110110 вычитание *В*

11111110 5-й остаток, *с*3 = 0

00010000.00xxxxxx сдвиг восстановленного *АH*

11110110 вычитание *В*

00000110 6-й остаток, *с*2 = 1

00001100.0xxxxxxx сдвиг остатка

11110110 вычитание В

00000010 7-й остаток, с1 = 1

00000100.xxxxxxxx сдвиг остатка

11110110 вычитание *В*

11111010 8-й остаток, *с*0 = 0

11111010

00001010 прибавление *В*

00000100 восстановлен остаток *АH* = 4

*С* = 0b01100110 = 102

***Практическая часть***

### **Программирование арифметических операций**

**Задание 1.** Изучить программу для исследования арифметических операций в стартовом наборе STK500, приведенную ниже.

Программой предусмотрен ввод кода операции, операндов, выполнение заданной операции и показ результатов.

В стартовом наборе STK500 всего восемь кнопок общего назначения (SW7…SW0). При тестировании арифметических операций эти кнопки используются следующим образом: кнопки SW0…SW2 – для ввода младшего (AL) и старшего байта (AH) первого операнда и одного байта второго операнда (BL), SW3…SW6 – для выполнения операций сложения, вычитания, умножения и деления, SW7 – для просмотра.

**Программа 3.1**

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;Программа тестирования в STK500 двоичных арифметических операций

; сложения, вычитания, умножения, деления

;Порт PD - порт управления для выбора операндов и операций

;Порт PB - порт индикации исходных операндов и результатов операции

;Соединения шлейфами: порт PB-LED, порт PD-SW

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.include "m8515def.inc" ;файл определений для ATmega8515

;назначение входов порта PD

.equ SW\_op\_AL = 0 ;кнопка выбора операнда op\_AL

.equ SW\_op\_AH = 1 ;кнопка выбора операнда op\_AH

.equ SW\_op\_BL = 2 ;кнопка выбора операнда op\_ВL

.equ SW\_ADD = 3 ;кнопка операции сложения res=op\_AL+op\_ВL

.equ SW\_SUB = 4 ;кнопка операции вычитания res=op\_AL-op\_ВL

.equ SW\_MUL = 5 ;кнопка операции умножения shov.res=op\_AL x op\_ВL

.equ SW\_DIV = 6 ;кнопка операции деления res=op\_AH.op\_AL/op\_ВL

.equ SW\_SHOW = 7 ;кнопка для просмотра признаков сложения-вычитания,

;старшего байта произведения или остатка при делении

.def op\_AL = r16 ;1-й операнд АL

.def op\_AH = r17 ;старший байт делимого AH

.def op\_BL = r18 ;2-й операнд ВL

.def res = r1 ;результат операции (сумма, разность,

; младший байт произведения или частное)

.def show = r31 ;регистр признаков сложения-вычитания,

; старшего байта произведения или остатка при делении

.def mul\_l = r21 ;младший байт произведения

.def mul\_h = r22 ;старший байт произведения

.def copy\_AH = r23 ;копия старшего байта делимого

.def copy\_AL = r24 ;копия младшего байта делимого

.def copy\_BL = r25 ;копия множителя

.def temp = r26 ;временный регистр

.def sw\_reg = r27 ;регистр состояния кнопок

.def count = r28 ;число операндов в таблице операндов

.def c\_bit = r29 ;счетчик циклов умножения (деления)

.macro vvod ;ввод операнда

lpm ;считывание байта из flash-памяти в r0

mov @0,r0 ; и пересылка в регистр операнда

mov res, r0

adiw zl, 1 ;увеличение указателя адреса на 1

dec count

brne exit

ldi ZL,low(tabl\_op\*2) ;перезагрузка начала таблицы операндов

ldi ZH,high(tabl\_op\*2) ; в регистр Z

ldi count, 10 ;число заданных операндов в таблице 10

exit: nop

.endmacro

.org $000

;Инициализация стека, портов, адреcного регистра Z

ldi temp,low(RAMEND) ;установка

out SPL,temp ; указателя стека

ldi temp,high(RAMEND) ; на последнюю

out SPH,temp ; ячейку ОЗУ

ser temp ;настройка

out DDRB,temp ; порта PB

out PORTB,temp ; на вывод

clr temp ;настройка

out DDRD,temp ; порта PD

ser temp ; на

out PORTD,temp ; ввод

ldi ZL,low(tabl\_op\*2) ;загрузка адреса таблицы операндов

ldi ZH,high(tabl\_op\*2) ; в регистр Z

ldi count,10 ;число операндов 10

;Опрос кнопок и выполнение заданных действий

LOOP: in sw\_reg,PIND

sbrs sw\_reg,0

rjmp f\_op\_AL

sbrs sw\_reg,1

rjmp f\_op\_AH

sbrs sw\_reg,2

rjmp f\_op\_BL

sbrs sw\_reg,3

rjmp add\_bin

sbrs sw\_reg,4

rjmp sub\_bin

sbrs sw\_reg,5

rjmp mul\_bin

sbrs sw\_reg,6

rjmp div\_bin

sbrc sw\_reg,7

rjmp loop

mov res,show

rjmp outled

;Выборка 1-го операнда из таблицы операндов

f\_op\_AL: vvod op\_AL

rjmp outled

;Выборка старшего байта 1-го операнда (при делении)

f\_op\_AH: vvod op\_AH

rjmp outled

;Выборка 2-го операнда

f\_op\_BL: vvod op\_BL

rjmp outled

;Сложение 8-разрядных операндов

add\_bin: mov res,op\_AL

add res,op\_BL

in show,SREG ;выборка из регистра SREG

rjmp outled

;Вычитание 8-разрядных операндов

sub\_bin: mov res,op\_AL

sub res,op\_BL

in show,SREG ;выборка из регистра SREG

rjmp outled

;Умножение 8-разрядных операндов

mul­\_­bin: mul op\_AL,op\_BL

mov show,r1 ;копируем старший и

mov res,r0 ; младший байт произведения

rjmp outled

;Деление 16-разрядного числа на 8-разрядное

div\_bin: sbrc op\_AH,7 ;ошибки исходных данных

rjmp error

sbrc op\_BL,7

rjmp error

tst op\_BL ;ошибка при делении на 0

breq error

cp op\_AH,op\_BL ;ошибка при переполнении

brge error

clr res ;обнуляем частное

ldi c\_bit,8 ; число итераций

mov copy\_AH,op\_AH

mov copy\_AL,op\_AL

L4: clc

rol copy\_AL ;сдвиг

rol copy\_AH ; делимого

lsl res ;сдвиг частного влево

sub copy\_AH,op\_BL ;вычитание делителя

brcs recov ;если остаток < 0,переход

inc res ; иначе добавить 1 в частное

rjmp L5

recov: add copy\_AH,op\_BL ;восстановление остатка

L5: dec c\_bit

brne L4

mov show,copy\_AH ;пересылка остатка

rjmp outled

error: clr temp ;сигнал об ошибке деления

out PORTB, temp

rcall delay

ser temp

out PORTB, temp

rjmp wait

outled: com res

out portb,res

rcall delay

wait: in sw\_reg,PIND ;ждать, пока кнопка не отпущена

com sw\_reg

brne wait

rjmp loop

; Задержка

DELAY: ldi r19,10

m1: ldi r20,250

m3: ldi r21,250

m2: dec r21

brne m2

dec r20

brne m3

dec r19

brne m1

ret

; Таблица операндов в шестнадцатеричном представлении

tabl\_op: .db 0xE5,0x10,0x1E,0xAA,0x6C,0xC7,0x1D,0xE2,0xD7,0x9B

Ввести в таблицу операндов в конце программы вместо приведенных в тексте программы 10 операндов из таблицы вариантов (табл.1).

Таблица 1. Таблица вариантов программируемых операндов в шестнадцатеричном коде

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№ вар.** | **Беззнаковые** | | **Числовые операнды со знаком в дополнительном коде** | | | | | | | |
| **AL0 BL0** | | **AL1 BL1** | | **AL2 BL2** | | **AL3 BL3** | | **AL4 BL4** | |
| 1 | 0x9C | 0xF0 | 0xF6 | 0x2A | 0x 6F | 0x5F | 0x8A | 0x5F | 0xF6 | 0xB1 |
| 2 | 0xA6 | 0xE4 | 0xE8 | 0x3B | 0x7E | 0x6D | 0x4B | 0x6B | 0x56 | 0xA6 |
| 3 | 0xB5 | 0xD7 | 0xD4 | 0xC4 | 0x8B | 0x4B | 0x6D | 0x4E | 0xA8 | 0x94 |
| 4 | 0xC7 | 0xB6 | 0xC7 | 0x5D | 0x9D | 0x8A | 0x0C | 0x8A | 0x69 | 0x7B |
| 5 | 0xD8 | 0xA5 | 0xB3 | 0x6E | 0x4C | 0x3C | 0x9F | 0x7D | 0xB0 | 0xB8 |
| 6 | 0xE4 | 0x96 | 0xA6 | 0x7F | 0x5A | 0x9E | 0x2E | 0x9C | 0xDC | 0xE9 |
| 7 | 0xF3 | 0x8A | 0x82 | 0x8D | 0x3F | 0x18 | 0x55 | 0xA8 | 0xCB | 0xFD |
| 8 | 0x80 | 0x7B | 0x70 | 0x9F | 0x2E | 0x8F | 0x8F | 0xC9 | 0x84 | 0x6E |
| 9 | 0x2F | 0x5F | 0x61 | 0x1C | 0x1D | 0xFB | 0xB0 | 0xD4 | 0x9D | 0x56 |
| 10 | 0x7D | 0x64 | 0x59 | 0xB9 | 0x9C | 0xA5 | 0xAC | 0xF6 | 0xAA | 0x9B |
| 11 | 0x6C | 0xFE | 0x7A | 0xAA | 0xFA | 0xD6 | 0xE7 | 0xE4 | 0xC7 | 0xCF |
| 12 | 0x5B | 0xD4 | 0x9B | 0xCE | 0xAC | 0xC8 | 0xF5 | 0x8B | 0x5E | 0xB4 |
| 13 | 0x4A | 0xC8 | 0x8C | 0xDB | 0xB0 | 0xE9 | 0xC3 | 0x92 | 0x90 | 0xF8 |
| 14 | 0x3E | 0xA5 | 0x6D | 0xFA | 0xC9 | 0xB1 | 0xE2 | 0x59 | 0x4B | 0xE0 |

После загрузки программы в микроконтроллер проверить работу программы на плате, перебирая операнды таблицы с помощью кнопок SW0, SW2 и наблюдая выбираемые операнды в двоичном коде на светодиодной линейке.

**Сложение/вычитание двоичных чисел**

**Задание 2.** Выполнить ряд примеров на сложение и вычитание, выбирая операнды слагаемых АL и ВL нажатием кнопок SW0 и SW2. Объяснить результаты операций при нажатиях кнопки SW3 (сложение) и SW4 (вычитание), рассматривая операнды как беззнаковые числа, затем как числа со знаком. В последнем случае загружаемые из таблицы операндов программы отрицательные числа, содержащие единицу в старшем разряде, следует рассматривать в дополнительном коде. Нажатие кнопки SW7 показывает признаки результата операции, формируемые в регистре SREG (табл.2): C – перенос при сложении (заем при вычитании), Z – признак нулевого результата, N – знак результата при операциях с числами со знаком, V – переполнение разрядной сетки, S=N⊕V – знак результата вне зависимости от переполнения, H – межтетрадный перенос (заем).

Таблица 2. **Байт признаков результата**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № разряда | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Флаг | − | − | H | S | V | N | Z | C |

Результаты наблюдений (исходные операнды, результаты операций и признаки) привести в табл. 3 в двоичном (A2, B2) и десятичном (А10,В10) виде. При оценке результатов соблюдать типы обрабатываемых переменных (беззнаковые целые или целые со знаком).

Таблица 3. **Результаты**

|  |  |  |  |
| --- | --- | --- | --- |
| Число А2/A10 | Число B2/B10 | А + В /  А − В | Признаки:  HSVNZC |
| 11000001/193  Беззнаковое | 01111111/127  Беззнаковое | 01000000 (64) /  01000010 (66) | 1 − − − 0 1  1 − − − 0 0 |
| 11000001/-63  Со знаком | 01111111/+127  Со знаком | 01000000 (+64) /  01000010 (+66) | 1 0 0 0 0 1  1 1 1 0 0 0 |

**Умножение и деление целых чисел**

**Задание 3.** Выполнить ряд примеров умножения 8-разрядных двоичных чисел с разными операциями умножения, выбирая соответствующие им операнды из таблицы операндов и проверяя результаты с помощью калькулятора. Нажатие кнопки SW5 показывает младший байт произведения, SW7 – старший байт.

**Задание 4.**Выполнить деление беззнаковых чисел, 16-разрядного делимого на 8-разрядный делитель, с восстановлением остатка при условиях, что делитель не равен 0 и его значение не вызовет переполнения, а также делимое и делитель заданы с нулевыми значениями старших разрядов. Если деление невозможно, выводится предупреждение путем зажигания и гашения всех светодиодов. Нажатие кнопки SW6 показывает частное, SW7 – остаток.

Выполнить 2-3 примера на деление двоичных чисел, самостоятельно подобрав делимое и делитель. Подобрать пример с максимальными значениями делимого *AH.AL* и делителя *B*, при которых частное *С* будет равно 0b11111111, изменив в случае необходимости программную таблицу операндов.

Запротоколировать деление двух операндов по шагампо образцу примера из описания алгоритма, указывая промежуточные значения в регистрах делимого (остатка) *AH.AL* и частного *С*.

**Задание 5\*.** Заменить в программе операции двоичного сложения и вычитания процедурами 2-10 сложения (вычитания), воспользовавшись рекомендованной литературой. Проверить работу процедур на примерах.

**Оформление отчета**

Отчет должен содержать:

а) базовую программу и алгоритм работы программы, схемы используемых алгоритмов умножения и деления;

б) протоколы выполнения всех операций сложения, вычитания, умножения и деления с расшифровкой двоичных результатов и признаками операций по образцу табл.3.

При защите уметь выполнять операции в машинном коде над заданными операндами и определять флаги (признаки) операций.

**Контрольные вопросы**

1. Какие числа со знаком представлены кодами 0b0111 1111, 0b1000 0000, 0b1000 0001?

2. Какие беззнаковые числа представлены кодами 0b1000 0000, 0b1000 0001, 0b0111 1111?

3. Написать ряд примеров на сложение/вычитание двоичных чисел со знаком, представив отрицательные числа в дополнительном коде. Результаты перевести в десятичный вид. Определить признаки операций. Написать программы для проверки в AVR Studio 4.

4. Используя рекомендуемую литературу, написать программные процедуры для 2-10 сложения/вычитания и протестировать их.

**Рекомендуемая литература**

В.Я. Хартов Микроконтроллеры AVR. Практикум для начинающих. Издательство МГТУ им. Н.Э. Баумана. М., 2012 г.