ЛАБОРАТОРНАЯ РАБОТА 3

“ПРОГРАММИРОВАНИЕ ВЕТВЯЩИХСЯ ПРОГРАММ В МП INTEL X86”

Рассмотрим решение следующей задачи.

1. Задание. Дано выражение:

$$X = \begin{cases} A^2 / B - 1, & \text{если } a > b \\ -A, & \text{если } a = b \\ (70 - B)/(A + 1), & \text{если } a < b \end{cases}$$

где исходные значения необходимо разделить на заданную константу, т.е. A= A/(var+2), B= B/(var+3);

var – номер варианта, равный 35.

Написать ветвящийся алгоритм вычисления значения арифметического выражения и реализовать его в виде программы для МП Intel x86.

Данное задание надо понимать так: в зависимости от соотношения между операндами A и B (равны ли они, больше операнд A или B) должно быть вычислено *только одно* значение выражения. Например, если значения операндов A и B совпадают (A=B=`5` или A=B=`0`), то результат X должен быть равен значению –A. Если операнд A больше операнда B (A=−20, B=−`50` или A= 20 B=`0`), то должно быть вычислено значение выражения A^2/B-1 и присвоено результирующей переменной X. Если же операнд A меньше операнда B (A= -10, B=`10` или A= 20, B=`30`), то будет вычислено значение выражения (70-B)/(A-1) и результат сохранен в переменной X.

Определим исходные данные, необходимые для решения поставленной задачи. Для этого рассмотрим правую часть формулы. Идентификаторы и числа, указанные в правой части выражения, являются исходными данными алгоритма и программы, реализующей данный алгоритм. В правой части выражения определены такие идентификаторы: A и B, а также константа 70.

Константой также является номер варианта, равный 35.

Таким образом, можно объявить исходные данные для алгоритма.

2. Исходные данные.

A, B – переменные, длинные целые знаковые числа.

70, 35 – константы, длинное целое знаковое число.

Определим имя переменной, в которую будет записан полученный результат, являющийся решением поставленной задачи. Для этого рассмотрим левую часть формулы. Идентификатор, указанный в левой части выражения, и будет именем переменной, предназначенной для хранения

результата работы программы, реализующей данный алгоритм. В левой части выражения определен единственный идентификатор X. Таким образом, можно объявить имя выходной переменной.

3. Требуемый результат.

X – переменная, длинное целое знаковое число.

4. Разработка алгоритма.

Алгоритм решения задачи включает такие шаги:

```
    Ввод исходных переменных и констант;
    Вычислить условное арифметическое выражение на C;
    Разделить исходные переменные A и B на константы на ассембле-
ре;
    Если (A==B)
      Вычислить выражение X=-A; //(1)
    иначе
      Если (A>B)
        Вычислить выражение X= A^2/B-1;//(2)
      иначе
        Вычислить выражение X=(70-B)/(A-1); //(3)
    Вывод полученного результата
```

В приведенном выше описании алгоритма знаками `//` начинаются комментарии, которые следуют до конца строки. Если внимательно анализировать алгоритм, можно заметить, что выражение (2) содержит дробь, знаменатель которой может обратиться в ноль. Это приведет к аварийному завершению программы. Чтобы не допустить этого программа должна обнаруживать такую ситуацию и блокировать ее. Для этого в ветвь алгоритма, вычисляющую значение выражения (2) необходимо ввести дополнительную проверку на равенство нулю знаменателя. Это означает, что в случае, если A>B, перед тем как вычислять значение выражения нужно проверять, не равна ли нулю переменная B. Если переменная B действительно равна нулю, дальнейшие вычисления надо прекратить и сообщить об этом пользователю. В этом случае мы устанавливаем в единицу переменную `err` и завершаем выполнение программы.

Аналогичная ситуация наблюдается и в выражении (3), где знаменатель также может обратиться в ноль. В этом случае поступаем так же: устанавливаем в единице признак ошибки `err` и завершаем выполнение программы.

Окончательная версия алгоритма приводится ниже.

5. Описание алгоритма на псевдокоде.

```
Ввод исходных переменных и констант;
Вычислить условное арифметическое выражение на C;
// Разделить исходные переменные A и B на константу 37;
A1 = A/var+2;
B1 = B/var+3;
// Сбросить признак ошибки err;
err = 0;
Если (A==B)
  Вычислить выражение X=-A; //(1)
```

```
иначе
  Если (A>B)
  {
   // Проверить знаменатель в выражении (2)
    Если (B==0)
      err=1;
    иначе
      Вычислить выражение X= A^2/B-1;//(2)
  }
  иначе
  {
  // Вычислить знаменатель в выражении (3)
    tmp = A-1;
   // Проверить знаменатель в выражении (3)
    Если (tmp==0)
      err=1;
    иначе
      Вычислить выражение X=(70-B)/(A-1); //(3)
  }
Вывод полученного результата
```

Детальный алгоритм решения задачи представлен ниже (рис.1). В одном блоке указываются те операции, которые можно выполнять последовательно слева направо. Для выполнения промежуточных вычислений используется отдельная вершина в схеме алгоритма. Чтобы ссылаться на отдельные блоки в схеме алгоритма удобно их пронумеровать, как показано на рисунке.

6. Описание схемы алгоритма.

Блок 1 – "Приглашение к вводу данных". Данный блок выдает на дисплей приглашение к вводу данных с клавиатуры.

Блок 2 – "Ввод переменных". Данный блок выполняет ввод чисел с клавиатуры и размещение их в памяти, выделенной для переменных. В данном случае вводятся переменные A и B. Пользователь может вводить различные числа.

Блок 3 – "Вычисление выражения на C". Данный блок делит исходные числа на константы 37 и 38, а также вычисляет условное арифметическое выражение на языке C.

Блок 4 – "Инициализация данных". Данный блок делит исходные числа A и B на константы 37 и 38 записывает полученные значения в регистры. В дальнейшем эти значения будут использоваться в выражении вместо исходных значений переменных A и B. Кроме того, в этом блоке сбрасывается в ноль переменная `err`, являющаяся флажком ошибок. В случае возникновения ошибки в ходе выполнения программы эта переменная может быть только установлена единицу.

Блок 5 - "Перейти, если "равно"". Данный блок сравнивает переменные A и B, определяет отношение между ними и выполняет переход на блок 10, если отношение "равно" истинно. Если отношение "равно" ложно переходим на следующий 5-й блок.

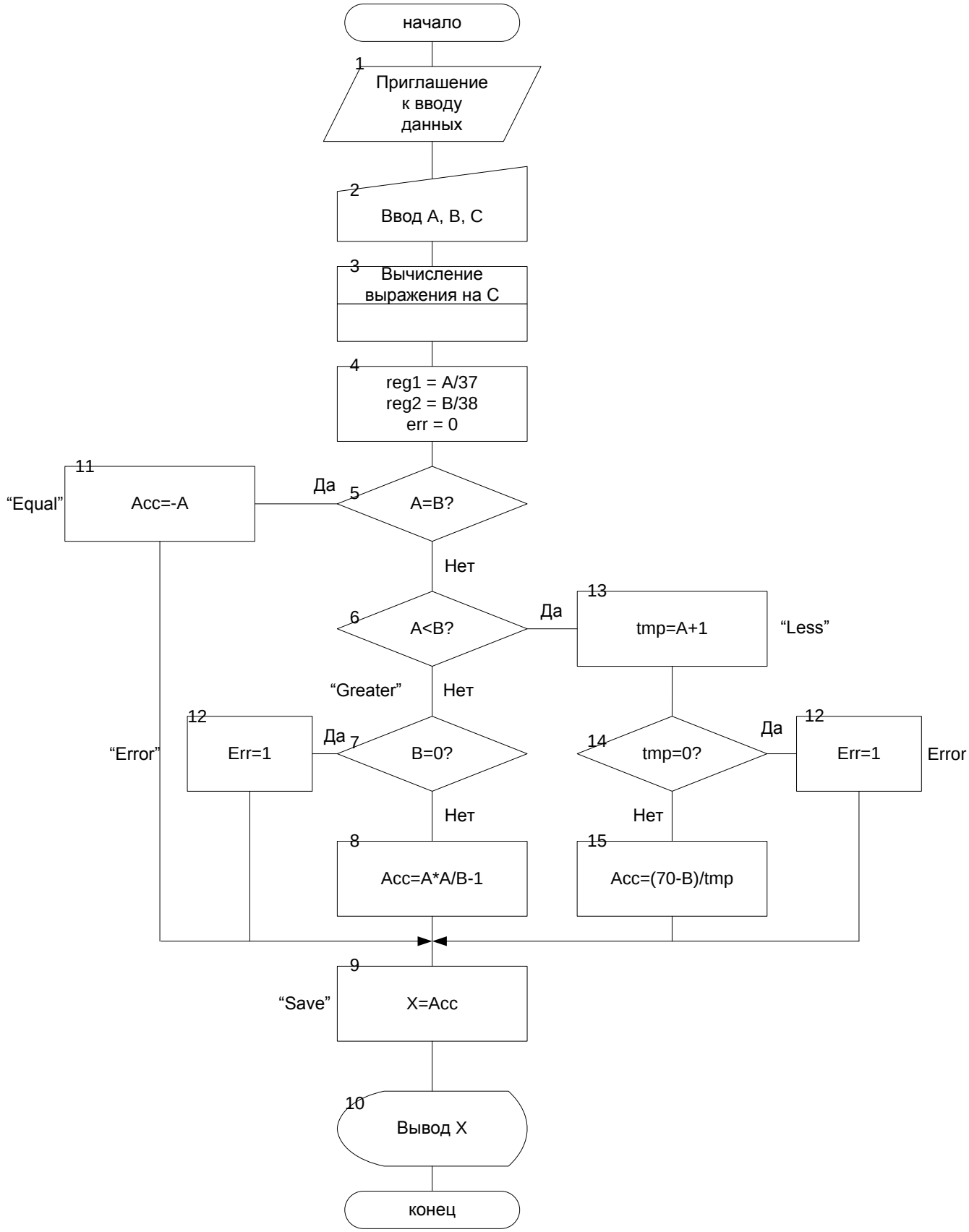

Рис.1. Схема алгоритма вычисления значения выражения

$$X = \begin{cases} A^2 / B - 1, & если\ \ a > b \\ -A, & если\ \ a = b \\ (70 - B)/(A+1), & если\ \ a < b \end{cases}$$

Блок 6 – "Перейти, если "меньше". В этом блоке сравниваются переменные A и B, определяется отношение между ними и выполняет переход на блок 12, если отношение "меньше" истинно. Если отношение "меньше" ложно переходим на следующий 6-й блок.

Блок 7 – "Проверить переменную B". Если мы попали на этот блок, то отношение между операндами A и B "больше" и мы будем вычислять выражение (A^2/B-1). Это выражение в знаменателе содержит переменную B, которая может обратиться в нуль. Данный блок проверяет переменную B, и, если она равна нулю, имеет место ошибка, управление передается на блок 11. В противном случае переходим на следующий блок 7.

Блок 8 – "Вычислить выражение для отношения "больше"". В этом блоке соблюдены следующие условия: A > B и B не равно нулю. Данный блок вычисляет значение выражения (A^2/B-1), не опасаясь возможного деления на ноль. Результат остается в аккумуляторе.

Блок 9 – "Сохранить результат". В данный блок попадаем после вычислений всех трех ветвей программы. После вычислений в каждой ветви результат остается в аккумуляторе. В данном блоке содержимое аккумулятора сохраняется в ячейке памяти с именем X.

Блок 10 – "Вывод результатов на дисплей". Данный блок выводит на дисплей значение переменной X, хранящей результат вычислений.

Блок 11 – "Вычислить выражение для отношения "равно"". Данный блок изменяет знак переменной A, сохраняет это значение в аккумуляторе и передает управление на блок 8.

Блок 12 – "Установить признак ошибок". В этом блоке устанавливается флажок ошибок - переменная err и выполнение программы завершается. После этого блока управление передается на блок 8.

Блоки 13-15 вычисляют значение выражения, если отношения между операндами A и B меньше, т.е. (70-B)/(A+1).

Блок 13 – "Вычислить знаменатель". Данный блок вычисляет выражение (A+1), являющееся знаменателем искомого выражения.

Блок 14 – "Знаменатель равен нулю?". Данный блок проверяет значение знаменателя, вычисленного в блоке 12, и, если он равен нулю, имеет место ошибка, и управление передается на блок 11. В противном случае переходим на следующий блок 14.

Блок 15 – "Вычислить выражение для отношения "меньше"". В этом блоке соблюдены следующие условия: A < B и выражение (A+1) не равно нулю. Данный блок вычисляет значение выражения 70-B/(A+1), не опасаясь возможного деления на ноль. Результат остается в аккумуляторе.

7. Кодирование алгоритма.

Выполним кодирование алгоритма решения задачи в виде программы на языке ассемблер. Программа представляет собой бесконечный цикл, который начинается в строках 25, а заканчивается – в строке 129. В этом цикле выполняется вывод приглашения к вводу данных, вычисление выражения на C, вычисление выражения на ассемблере и вывод полученных результатов.

Блок 1 – "Приглашение к вводу данных" - реализуют команды, расположенные в строках 27-32.

Блок 2 – "Ввод переменных" - реализует функция ввода `scanf()`, расположенная в строке 33.

Блок 3 – "Вычисление выражения на С" – реализуют команды, расположенные в строках 36-61.

Блок 4 – "Инициализация данных" – реализуют команды, расположенные в строках 43-53. В строке 64 с помощью директивы ассемблера присваивания "=" символическому имени DENOM_A присваивается значение константы 35. В строке 66 обнуляется переменная-флажок ошибок `err_a`. Строки 70-74 делят исходную переменную B на константу (DENOM_A+3) и сохраняют результат в регистре `edi`. В дальнейшем, всякий раз, когда потребуется переменная B, мы будем использовать содержимое регистра `edi`. Строки 76-79 делят исходную переменную A на константу (DENOM_A+2) и оставляют результат в аккумуляторе.

Блоки 5 и 6 - "Перейти, если "равно"" и "Перейти, если "меньше"" реализуют команды, расположенные в строках 84-86. Команда сравнения в строке 84 сравнивает два операнда A и B. Команда “условного перехода, если равно” (строка 85) выполняет переход на фрагмент, вычисляющий арифметическое выражение в блоке 10. Данный блок располагается, начиная с метки Equal. Команда “условного перехода, если меньше” (строка 86) выполняет переход на фрагмент, вычисляющий арифметическое выражение в блоке 13. Данный блок располагается, начиная с метки Less. Если соотношение между операндами таково, что ни один из переходов не произошел, то между операндами имеет место отношение "больше" и мы попадаем на фрагмент, вычисляющий арифметическое выражение в блоке 7, который начинается с метки Greater.

Блоки 7-8 реализуют команды из фрагмента Greater, расположенного в строках 89-96. Блок 7 реализуют команды в строках 89-90. В начале операнд B, хранящийся в регистре `edi`, сравнивается с нулем (строка 89), и с помощью команды "условного перехода, если ноль" (строка 90) выполняется переход на метку Error. Начиная с этой метки, располагается фрагмент обработки ошибок, реализующий блок 12.

Блок 8 – "Вычислить выражение для отношения "больше"" – реализуют команды, расположенные в строках 92-95, которые вычисляют значение выражения (A^2/B-1). Результат остается в аккумуляторе. Последняя команда этого блока – команда безусловного перехода `jmp`, выполняющая переход на фрагмент Save, сохраняющий в ячейке памяти результат, полученный в аккумуляторе. Фрагмент Save реализует блок 9 схемы алгоритма.

Блок 11 – "Вычислить выражение для отношения "равно"" – вычисляют команды, расположенные строках 99-100. Команда `neg` изменяет знак операнда A, расположенного в аккумуляторе

(строка 99) и при помощи команды `jmp` передает управление на фрагмент Save. Результат остается в аккумуляторе.

Блоки 13-15 реализуют фрагмент программы Less, который вычисляет значение выражения (70-B)/(A+1). Этот фрагмент располагается в строках 104-111. Вначале в строке 104 вычисляется знаменатель, сравнивается с нулем. Если он равен нулю, с помощью команды условного перехода, выполняется переход на фрагмент обработки ошибок. Если же переход не произошел, то корректное выражение вычисляют команды, расположенные в строках 107-111. Результат остается в аккумуляторе. Сразу же за этим фрагментом в строках 113-114 располагается фрагмент Save, поэтому передавать управление на него с помощью команды `jmp` не нужно.

Блок 9 – "Сохранить результат" – реализует команда в строке 113. Команда безусловного перехода (строка 114) передает управление на блок вывода результатов, пропуская блок обработки ошибок в строке 117. В данный блок попадаем после вычислений всех трех ветвей программы. После вычислений в каждой ветви результат остается в аккумуляторе. В данном блоке содержимое аккумулятора сохраняется в ячейке памяти с именем X_A.

Блок 12 – "Установить признак ошибок" – реализует единственная команда inc, которая увеличивает на единицу признак ошибок err (строка 117). Напомню, что в начале программы этот признак всегда сбрасывается в ноль.

Блок 10 – "Вывод результатов на дисплей" – реализуют функции вывода, расположенные в строках 121-128. Данный блок выводит на дисплей значение переменной X, хранящей результат вычислений.

8. Текст программы с комментариями.

```
 1: //+================================================================
 2:  // File lab_2.cpp
 3:  // Ветвящаяся программа
 4:  // Вариант 35
 5:  // Эта программа вычисляет условное арифметическое выражение
 6:  //     { a^2/b-1,      a>b }
 7:  // X = < -a,           a=b >
 8:  //     { (70-b)/(a+1), a<b }
 9:  // где a=a/(35+2), b=b/(35+3)
10:  //
11:  // Выполняется проверка на переполнение
12:  //
13:  // (C) Дужий В.И., 2012
14:  //-================================================================
15:  #include <stdio.h>
16:
17:  #define   VAR  35
18:  long int  a, a1, b, b1, x, x_a, tmp;
19:  int err;  // Ошибка в выражении на C
20:  int err_a;// Ошибка в выражении на ассемблере
21:
22:  int main()
23:  {
24:    printf("\n\t\t(C) Дужий В.И., 2012");
```

```
25:    for (;;)
26:    {
27:      printf("\n\tВычислить выражение:");
28:      printf("\n\t    { a^2/b-1,      a>b }");
29:      printf("\n\tX = < -a,           a=b >");
30:      printf("\n\t    { (70-b)/(a+1), a<b }");
31:      printf("\n\tгде a=a/37, b=b/38");
32:      printf("\nПожалуйста, введите целые числа A и B : ");
33:      scanf("%li%li",&a,&b);
34:  //============================== C ================================
35:  // Разделить исходные переменные
36:      a1 = a/(VAR+2);
37:      b1 = b/(VAR+3);
38:  // Сбросить признак ошибки на C
39:      err = 0;
40:  // Вычислить выражение
41:  // (a==b)?
42:      if (a1==b1)
43:        x = -a1;
44:      else
45:  // (a>b)?
46:        if (a1>b1)
47:        {
48:          if (b1==0)
49:            err = 1;
50:          else
51:            x = a1*a1/b1-1;
52:        }
53:  // (a<b)?
54:        else
55:        {
56:          tmp = a1+1;
57:          if (tmp==0)
58:            err = 1;
59:          else
60:            x = (70-b1)/tmp;
61:        };
62:  //============================ Assembler ============================
63:
64:
65:  // err_a=0; Нет ошибок
66:     movl    err_a,0
67:  // Разделить исходные переменные на знаменатель VAR
68:  __asm{
69:  // b1=b/DENOM_A+3
70:          mov    ebx,VAR+3
71:          mov    eax,b
72:          cdq
73:          idiv   ebx
74:          mov    edi,eax       // b1 -> edi
75:  // a1=a/DENOM_A+2
76:          mov    ebx,VAR+2
77:          mov    eax,a
78:          cdq
79:          idiv   ebx            // a1 -> esi
80:  }
81:  __asm{
82:  // Вычислить выражение
83:  // if (a1==b1)
```

```
 84:         cmp    eax,edi
 85:         je     Equal
 86:         jl     Less
 87:  Greater:
 88:  // if (b1==0) err_a = 1
 89:         test   edi,edi
 90:         je     Error
 91:  // else x = a1*a1/b1-1;
 92:         imul   eax,eax
 93:         cdq
 94:         idiv   edi
 95:         dec    eax
 96:         jmp    Save
 97:  Equal:
 98:  //   x = -a1;
 99:         neg    eax
100:         jmp    Save
101:  // else x = (70-b1)/tmp
102:  Less:
103:  // tmp = a1+1 -> eax
104:         inc    eax
105:  // if ((a1+1)==0) err_a = 1
106:         je     Error
107:         sub    edi,70         // b1-70 -> edi
108:         neg    edi             // 70-b1 -> edi
109:         xchg   edi,eax        // tmp <-> (70-b1)
110:         cdq
111:         idiv   edi             // (70-b1)/tmp -> eax
112:  Save:
113:         mov    x_a,eax         // eax -> x
114:         jmp    End
115:  // err_a = 1
116:  Error:                          // err = 1
117:         inc    err_a
118:  End:
119:  }
120:  // Вывод результатов
121:     if (err)
122:       printf("***( C )*** Ошибка: попытка деления на 0\n");
123:     else
124:       printf("Результат ( C ): %li\n",x);
125:     if (err_a)
126:       printf("***(Asm)*** Ошибка: попытка деления на 0\n");
127:     else
128:       printf("Результат (Asm): %li\n",x_a);
129:  }
130:     return 0;
131:  }
```

9. Тестирование программы.

Составим тестовые примеры, которые позволят найти ошибки в программе. Для правильного выбора тестовых примеров следует проанализировать поставленную задачу и выбрать значения исходных данных, используя следующие соображения:

– исходные данные должны содержать все возможные комбинации чисел с *различными комбинациями знаков* (все числа положительные, все числа отрицательные, некоторые числа положи-

тельные, в то время как другие - отрицательные). Полученный результат должен лежать *в пределах* допустимого диапазона представления, т.е. от –2 147 000 000 до +2 147 000 000. В приведенной ниже таблице это тесты 1-4.

– выбрать такие значения исходных чисел, чтобы полученный результат выходил *за пределы верхней границы* диапазона, т.е. был бы больше +2 147 000 000. В приведенной ниже таблице это тест 5.

– выбрать такие значения исходных чисел, чтобы полученный результат выходил *за пределы нижней границы диапазона*, т.е. был бы меньше -2 147 000 000. При решении поставленной задачи такая ситуация невозможна, и поэтому такой тест в приведенной ниже таблице отсутствует.

– выбрать такие значения исходных чисел, чтобы в процессе вычислений *знаменатель был бы равен 0*, что приведет к делению на 0 в процессе вычислений. В приведенной ниже таблице это тесты 6 и 7.

10. Тестовые примеры.

Номер	Исходные данные		Ожидаемый результат	Полученный результат	Цель теста
	A	B			
1	300	200	11		(a>b) & (b<>0)
2	-300	-500	-5		(a>b) & (b<>0)
3	800	30	???		(a>b)&(b=0), деление на 0
4	600	-30	???		(a>b)&(b=0), деление на 0
5	500	500	-500		a=b
6	-700	-700	700		a=b
7	-50	400	???		a<b)&(a=0), деление на 0

11. Протокол тестирования программы приводится ниже.

```
                (C) Дужий В.И., 2012
        Вычислить выражение:
            { a^2/b-1,      a>b }
        X = < -a,           a=b >
            { (70-b)/(a+1), a<b }
        где a=a/37, b=b/38
Пожалуйста, введите целые числа A и B : 300 200
Результат ( C ): 11
Результат (Asm): 11

        Вычислить выражение:
            { a^2/b-1,      a>b }
        X = < -a,           a=b >
            { (70-b)/(a+1), a<b }
        где a=a/37, b=b/38
Пожалуйста, введите целые числа A и B : 300 30
***( C )*** Ошибка: попытка деления на 0
***(Asm)*** Ошибка: попытка деления на 0
```

```
        Вычислить выражение:
            { a^2/b-1,      a>b }
        X = < -a,           a=b >
            { (70-b)/(a+1), a<b }
        где a=a/37, b=b/38
Пожалуйста, введите целые числа A и B : 500 500
Результат ( C ): -13
Результат (Asm): -13

        Вычислить выражение:
            { a^2/b-1,      a>b }
        X = < -a,           a=b >
            { (70-b)/(a+1), a<b }
        где a=a/37, b=b/38
Пожалуйста, введите целые числа A и B : -50 400
***( C )*** Ошибка: попытка деления на 0
***(Asm)*** Ошибка: попытка деления на 0

        Вычислить выражение:
            { a^2/b-1,      a>b }
        X = < -a,           a=b >
            { (70-b)/(a+1), a<b }
        где a=a/37, b=b/38
Пожалуйста, введите целые числа A и B : 100 500
Результат ( C ): 19
Результат (Asm): 19

        Вычислить выражение:
            { a^2/b-1,      a>b }
        X = < -a,           a=b >
            { (70-b)/(a+1), a<b }
        где a=a/37, b=b/38
Пожалуйста, введите целые числа A и B :^C
```