ЛАБОРАТОРНА РОБОТА 1

ЛІНІЙНІ ПРОГРАМИ. АРИФМЕТИЧНІ КОМАНДИ МП INTEL x86

Рассмотрим решение следующей задачи.

1. **Задание**. Дано выражение:

$$(X^2-9)/(X-3)=X+3$$

где исходное значение необходимо разделить на заданную константу, т.е. X= X/(var+2);

var – номер варианта, равный 35.

Написать линейный алгоритм вычисления значения левой и правой частей арифметического выражения и реализовать его в виде программы на С и на ассемблере. В процессе вычислений программа должна контролировать возможное переполнение.

2. Определим исходные данные, необходимые для решения поставленной задачи. Для этого рассмотрим идентификаторы и числа, используемые в тождестве. Идентификаторы и числа, указанные в выражении, являются исходными данными алгоритма и программы, реализующей данный алгоритм. В левой части выражения определен идентификатор X и число 9, а в правой – число 3. Таким образом, можно объявить исходные данные для алгоритма.

Исходные данные.

X – переменная, длинное целое знаковое число.

9, 3, 35 – константы, длинные целые знаковые числа.

3. Определим имена переменных, в которые будут записаны полученные результаты, являющиеся решением поставленной задачи. Поскольку данное тождество нужно вычислить на С и на ассемблере, то необходимо определить два комплекта переменных, в которые будет записан полученный результат: отдельно для С и отдельно для ассемблера. После вычисления левой части тождества на С результат будет записан в переменную `Left`, а значение правой части - в переменную `Right`. Результаты вычисления левой и правой частей тождества на ассемблере будут храниться в переменных `Left_a` и `Right_a` соответственно. Разумеется, программист может по своему усмотрению объявлять другие имена переменных. Таким образом, можно объявить имена выходной переменной.

Требуемый результат.

Left, Right – переменные, длинные целые знаковые числа для фрагмента на С.

Left_a, Right_a – переменные, длинные целые знаковые числа для фрагмента на ассемблере.

4. Разработка алгоритма.

Алгоритм решения задачи включает такие шаги:

```
Ввести исходные переменные;
Вычислить левую часть тождества на C;
Вычислить правую часть тождества на C;
Вычислить левую часть тождества на ассемблере;
Вычислить правую часть тождества на ассемблере;
Вывести значение левой и правой частей тождества на C;
Вывести значение левой и правой частей тождества на ассемблере.
```

По условию задачи исходную переменную X необходимо разделить на константу.

Кроме того, необходимо контролировать переполнение, которое может возникнуть в процессе вычисления выражения. В процессе вычисления используются операции сложения, вычитания и умножения. В общем случае переполнение может возникнуть после выполнения каждой арифметической команды, и, следовательно, переполнение также нужно определять после каждой арифметической команды. Однако такие проверки сильно замедлят выполнение программы и увеличат размер кода. Чтобы избежать этих неприятностей, определим команду или команды, выполнение которых с б*о*льшей вероятностью может привести к появлению ошибок. Такой командой является команда умножения, которая в общем случае приводит к получению результата двойной длины. Например, после умножения двух двухразрядных десятичных чисел результатом будет четырехразрядное число. Например, 90*90=8100. Поэтому при перемножении сравнительно небольших чисел может возникнуть переполнение разрядной сетки. Например,

50 000*50 000 = 25 0000 0000 = 2 500 000 000 – переполнение.

В процессе умножения возникло переполнение, т.к. максимальным знаковым 32-битным числом является значение около 2 100 000 000.

Для получения переполнения после операции сложения и вычитания нужно обрабатывать числа, близкие к максимальному. Например,

0x7FFF FFFF + 1 = 0x 8000 0000 – переполнение.

Если в выражении встречается операция деления, то следует проверять возможность обращения знаменателя в ноль, поскольку при делении на ноль также возникает переполнение. Такую проверку при выполнении деления необходимо выполнять в случае, если знаменатель является вычисляемым значением, если же знаменатель является ненулевой константой то такая проверка не нужна.

Таким образом, можно рекомендовать выполнять проверку на переполнение *после* каждой команды умножения, а также проверять на ноль значение вычисляемого знаменателя *перед* делением.

После внимательного анализа исходного тождества, можно заметить, что операция ум-

ножения встречается только в одном фрагменте левой части тождества. Поэтому проверку переполнения нужно выполнять только во время вычисления левой части тождества после вычисления значения X^2 и после вычисления значения знаменателя X-3. Уточненный алгоритм решения задачи будет иметь вид:

```
Ввести исходные переменные;
Вычислить левую часть тождества на C;
Вычислить правую часть тождества на C;
Разделить исходную переменную X на константу 37 на ассемблере;
Вычислить левую часть тождества на ассемблере;
Определить, имеется ли переполнение;
Вычислить правую часть тождества на ассемблере;
Вывести значение левой и правой частей тождества на C;
```

Если нет переполнения, вывести значение левой и правой частей тождества на ассемблере.

Алгоритм решения задачи на псевдокоде.

```
Ввести исходные переменные;
Вычислить левую часть тождества на C;
Вычислить правую часть тождества на C;
// Сбросить признак ошибки err;
err = 0;
// Разделить исходную переменную X на константу 37;
X1 = X/37;
// Вычислить тождество на ассемблере;
// Вычислить левую часть тождества;
tmp = X1*X1; // Вычислить произведение
// Определить, имеется ли переполнение;
Если переполнение, err = 1;
denom = X1-3; // Вычислить знаменатель
// Определить, имеется ли переполнение;
Если переполнение, err = 1;
// Продолжить вычисление левой части тождества;
Left = (tmp-9)/denom;
// Вычислить правую часть тождества на ассемблере;
Right = X1+3;
// Вывести значение левой и правой частей тождества на C;
// Если нет переполнения, вывести значение левой и правой
// частей тождества на ассемблере;
```

Если (err==0) Вывести сообщение об ошибке;

иначе Вывести значение левой и правой частей тождества.

Детальный алгоритм решения задачи представлен ниже на рисунке 1.

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 вводится переменная X. Пользователь может вводить различные числа.

Блок 3 – "Вычисление выражения на C". Данный блок делит исходное число на константу 37, а также вычисляет левую и правую часть выражения на языке C.

Блоки 4-10 вычисляют левую и правую часть тождества на ассемблере.

Блок 4 – "Инициализация данных". Данный блок делит исходное число X на константу 37 и записывает полученное значение в регистр. В дальнейшем именно это значение будет использоваться в выражении вместо значения переменной X. Кроме того, в этом блоке сбрасывается в ноль переменная `err`, являющаяся флажком ошибок. В дальнейшем в случае возникновения ошибки эта переменная может быть только установлена единицу.

Блок 5 – "Вычислить числитель в левой части выражения". Данный блок вычисляет значение выражения (X^2-9). Результат следует сохранить в некотором регистре.

Блок 6 – "Проверить переполнение". Поскольку предыдущий блок содержит операцию умножения необходимо выполнить проверку на переполнение. Если *после* выполнения команды умножения имело место знаковое переполнение (признак OF = 1), то флажок ошибок `err` нужно установить в единицу.

Блок 7 – "Вычислить знаменатель в левой части выражения". Данный блок вычисляет значение выражения (X-3). Результат следует сохранить в некотором регистре.

Блок 8 – "Проверить переполнение". Поскольку следующий блок содержит операцию деления необходимо выполнить проверку на переполнение. Если вычисленный в предыдущем блоке знаменатель равен нулю, то после выполнения команды деления будет переполнение, поэтому флажок ошибок `err` нужно установить в единицу.

Блок 9 – "Вычислить окончательное значение левой части тождества". Данный блок выполняет деление значения `tmp`, полученного в блоке 5, на значение `denom`, полученного в блоке 7. Результат сохраняется в переменной `left`.

Блок 10 – "Вычислить значение правой части тождества". Данный блок вычисляет значение выражения (X+3). Результат сохраняется в переменной `right`.

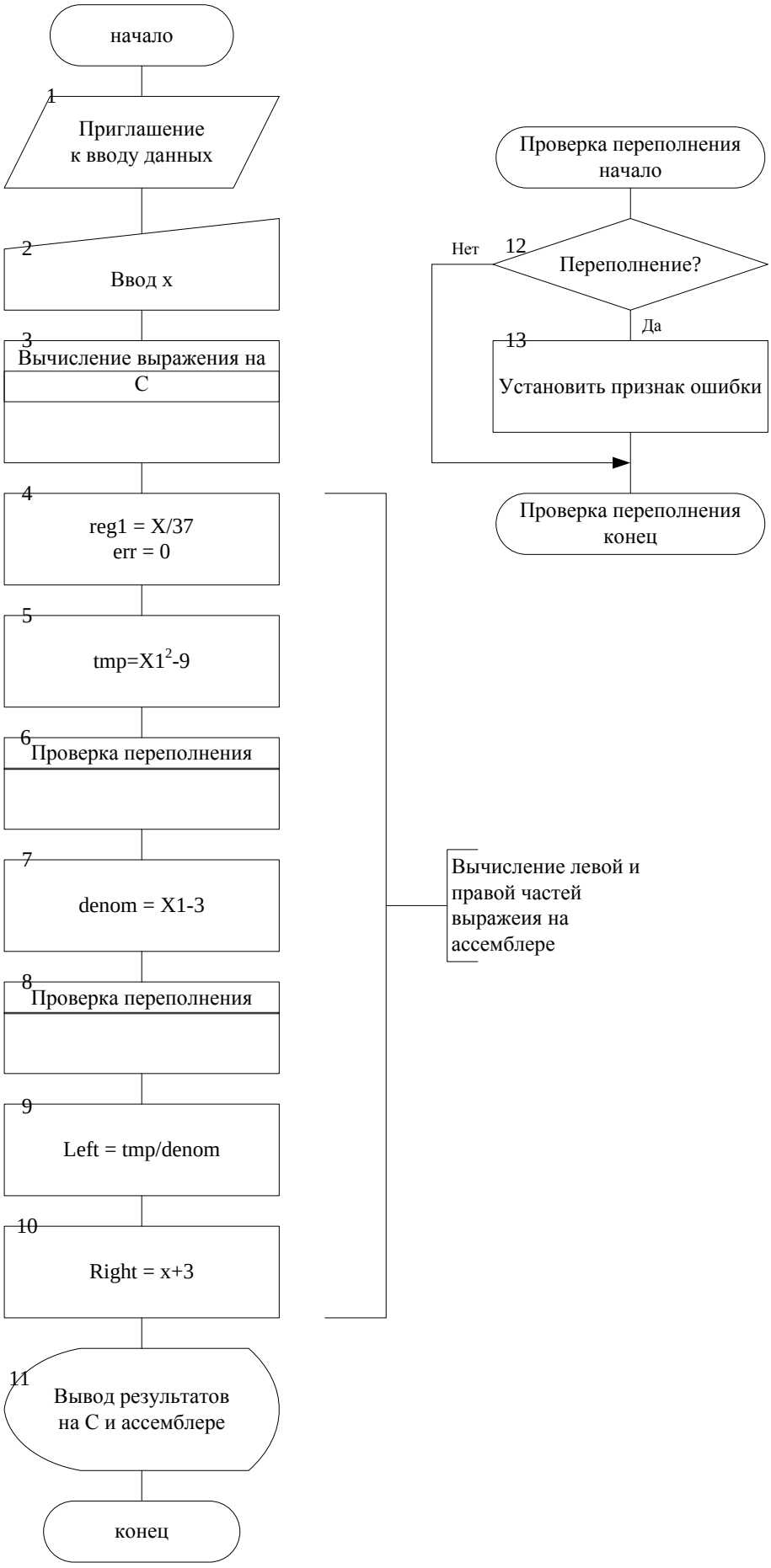

Рисунок 1 - Схема алгоритма вычисления тождества
$(X^2-9)/(X-3)=X+3$

Блок 11 – "Вывод результатов на дисплей". Данный блок выводит на дисплей значение переменных `left` и `right`.

5. Кодирование алгоритма.

Теперь можно выполнить кодирование алгоритма решения задачи в виде программы на языке ассемблер. Программа представляет собой бесконечный цикл, который начинается в строках 24-25, а заканчивается – в строке 91. В этом цикле выполняется вывод приглашения к вводу данных, вычисление выражения на С, вычисление выражения на ассемблере и вывод полученных результатов.

Блок 1 – "Приглашение к вводу данных" - реализуют команды, расположенные в строках 26-28.

Блок 2 – "Ввод переменных" - реализует функция ввода `scanf()`, расположенная в строке 29.

Блок 3 – "Вычисление выражения на С" – реализуют команды, расположенные в строках 32-40.

Блок 4 – "Инициализация данных" – реализуют команды, расположенные в строках 43-53. В строке 43 с помощью директивы ассемблера присваивания "=" символическому имени DENOM_A присваивается значение константы 37. В строке 45 обнуляется переменная-флажок ошибок `err_a`. Строки 49-53 делят исходную переменную X на константу DENOM_A и сохраняют результат в регистре `esi`. Вначале константа помещается в регистр `ebx` (строка 49), поскольку команда деления `idiv` не может делить на константу, далее содержимое ячейки памяти с адресом X помещается в аккумулятор (строка 50) и знак числа в аккумуляторе размножается в регистре `edx` (строка 51). Следующая команда `idiv` делит содержимое регистровой пары `edx:eax` на константу в регистре `ebx` (строка 52) и полученное в аккумуляторе частное сохраняется в индексном регистре `esi` (строка 53). В дальнейшем, всякий раз, когда потребуется переменная X мы будем использовать содержимое регистра `esi`.

Блок 5 – "Вычислить числитель в левой части выражения" – реализуют команды, расположенные в строках 58-59.

Блок 6 – "Проверить переполнение" – выполняют команды, расположенные в строках 60-62. Если в результате выполнения команды умножения `imul` *не был* установлен флажок переполнения OF, то с помощью команды "условного перехода, если нет переполнения" (строка 60) выполнение программы продолжается далее, начиная с метки L1 (строке 63). Если же флажок переполнения OF все-таки был установлен, т.е. имело место переполнение, то в строке 61 устанавливается признак ошибки и выполнение программы прекращается, для чего в строке 62 выполняется безусловный переход на метку End, расположенную в строке 78. Команда вычитания в строке 64 завершает вычисление числителя.

Блок 7 – "Вычислить знаменатель в левой части выражения" - реализуют команды в строках 67-68.

Блок 8 – "Проверить переполнение" – реализуют команды в строках 69-71. Если после вычитания, результат не равен нулю, то с помощью команды "условного перехода, если не ноль" выполнение программы будет продолжено, начиная с метки L2. Если результат оказался равен нулю, то выполняются строки 70-71, в которых устанавливается признак ошибки (строка 70) и выполняется переход на конец программы (строка 71).

Блок 9 – "Вычислить окончательное значение левой части тождества" – реализуют команды, расположенные в строках 72-75, которые делят числитель выражения, расположенный в аккумуляторе на знаменатель, расположенный в регистре `ebx`. Полученный результат сохраняется в памяти (строка 75) в ячейке с адресом `left_a`.

Блок 10 – "Вычислить значение правой части тождества" – выполняют команды в строках 79-80. Полученный результат сохраняется в ячейке памяти с адресом `right_a`.

Блок 11 – "Вывод результатов на дисплей" – выполняют функции вывода языка С, расположенные в строках 82-91.

Текст программы с комментариями.

```
 1:  //+==================================================================
 2:  // File lab_1.c
 3:  // Линейная арифметическая программа
 4:  // Вариант 37
 5:  // Эта программа проверяет арифметическое тождество
 6:  // (x^2-9)/(x-3) = x+3,
 7:  // где x=x/37
 8:  //
 9:  // Для этого вычисляют значение левой и правой частей этого тождества
10:  // на С и на ассемблере
11:  //
12:  // (C) Дужий В.И., 2012
13:  //-==================================================================
14:  #include <stdio.h>
15:
16:  #define   DENOM  37
17:  long int  x, x1, left, right, left_a, right_a;
18:  int err;   // Ошибка в левой части выражения на С
19:  int err_la;// Ошибка в левой части выражения в ассемблере
20:
21:  int main()
22:  {
23:    printf("\n\t\t(C) Дужий В.И., 2012");
24:    for (;;)
25:    {
26:      printf("\n\tПроверить тождество (x^2-9)/(x-3) = x+3,");
27:      printf("\n\tгде x=x/37");
28:      printf("\nПожалуйста, введите целое число X : ");
29:      scanf("%li",&x);
30:  //============================== C ================================
31:  // Разделить исходные переменные
32:      x1 = x/DENOM;
```

```
33:  // Сбросить признак ошибки на С
34:      err = 0;
35:  // Вычислить выражение
36:      if ((x1-3)==0)
37:        err = 1;
38:      else
39:        left = (x1*x1-9)/(x1-3);
40:      right= x1+3;
41:  //============================ Assembler ============================
42:  // Объявить константу для ассемблера
43:  __asm{
44:  // err_la=0; Нет ошибок
45:           mov    err_la,0
46:  // Разделить исходные переменные на знаменатель DENOM
47:  //
48:  // x1=x/DENOM
49:          mov    ebx,DENOM
50:          mov    eax,x
51:          cdq
52:          idiv   ebx
53:          mov    esi,eax
54:  //
55:  // Вычислить левую часть тождества
56:  //(x1*x1-9)/(x1-3) --> left
57:  // Вычислить числитель
58:          mov    eax,esi
59:          imul   eax,eax
60:          jno    L1
61:          mov    err_la,1
62:          jmp    End
63:  L1:
64:          sub    eax,9
65:  // Вычислить знаменатель
66:  //
67:          mov    ebx,esi
68:          sub    ebx,3
69:          jnz    L2
70:          mov    err_la,1
71:          jmp    End
72:  L2:
73:          cdq
74:          idiv   ebx
75:          mov    left_a,eax
76:  // Вычислить правую часть тождества
77:  // x1-3 --> right_a
78:  End:
79:          mov    right_a,esi
80:          add    right_a,3    }
81:  // Вывод результатов
82:     if (err){
83:       printf("\n***( C )*** Ошибка в левой части выражения");}
84:     else {
85:       printf("Левая часть тождества ( C ): %li",left);}
86:     printf("\nПравая часть тождества( C ): %li",right);
87:     if (err_la){
88:       printf("\n***(Asm)*** Ошибка в левой части выражения");}
89:     else {
90:       printf("\nЛевая часть тождества (Asm): %li",left_a);}
91:     printf("\nПравая часть тождества(Asm): %li\n",right_a);
```

```
 92:  }
 93:     return 0;
 94:  }
```

6. 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 положительные, 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.

Тестовые примеры.

Номер	Исходные данные, x	Ожидаемый результат	Полученный результат	Цель теста
1	3700	103		Положительное число
2	-3700	-97		Отрицательное число
3	10000	273		Положительное число
4	-10000	267		Отрицательное число
5	**1850000**	???		Переполнение (>+2147000000)
6	**111**	???		Деление на 0
7	**145**	???		Деление на 0

Протокол тестирования программы приводится ниже.

```
        (C) Дужий В.И., 2012
        Проверить тождество (x^2-9)/(x-3) = x+3,
        где x=x/37
Пожалуйста, введите целое число X : 3700
Левая часть тождества ( C ): 103
Правая часть тождества( C ): 103
Левая часть тождества (Asm): 103
Правая часть тождества(Asm): 103
```

```
        Проверить тождество (x^2-9)/(x-3) = x+3,
        где x=x/37
Пожалуйста, введите целое число X : -3700
Левая часть тождества ( C ): -97
Правая часть тождества( C ): -97
Левая часть тождества (Asm): -97
Правая часть тождества(Asm): -97

        Проверить тождество (x^2-9)/(x-3) = x+3,
        где x=x/37
Пожалуйста, введите целое число X : 10000
Левая часть тождества ( C ): 273
Правая часть тождества( C ): 273
Левая часть тождества (Asm): 273
Правая часть тождества(Asm): 273

        Проверить тождество (x^2-9)/(x-3) = x+3,
        где x=x/37
Пожалуйста, введите целое число X : -10000
Левая часть тождества ( C ): -267
Правая часть тождества( C ): -267
Левая часть тождества (Asm): -267
Правая часть тождества(Asm): -267

        Проверить тождество (x^2-9)/(x-3) = x+3,
        где x=x/37
Пожалуйста, введите целое число X : 111

***( C )*** Ошибка в левой части выражения
Правая часть тождества( C ): 6
***(Asm)*** Ошибка в левой части выражения
Правая часть тождества(Asm): 6

        Проверить тождество (x^2-9)/(x-3) = x+3,
        где x=x/37
Пожалуйста, введите целое число X : 145

***( C )*** Ошибка в левой части выражения
Правая часть тождества( C ): 6
***(Asm)*** Ошибка в левой части выражения
Правая часть тождества(Asm): 6

        Проверить тождество (x^2-9)/(x-3) = x+3,
        где x=x/37
Пожалуйста, введите целое число X : 150
Левая часть тождества ( C ): 7
Правая часть тождества( C ): 7
Левая часть тождества (Asm): 7
Правая часть тождества(Asm): 7

        Проверить тождество (x^2-9)/(x-3) = x+3,
        где x=x/37
Пожалуйста, введите целое число X : 1850000
Левая часть тождества ( C ): -35901
Правая часть тождества( C ): 50003
***(Asm)*** Ошибка в левой части выражения
Правая часть тождества(Asm): 50003
```

```
        Проверить тождество (x^2-9)/(x-3) = x+3,
        где x=x/37
Пожалуйста, введите целое число X : -1850000
Левая часть тождества ( C ): 35897
Правая часть тождества( C ): -49997
***(Asm)*** Ошибка в левой части выражения
Правая часть тождества(Asm): -49997
```