Министерство образования Республики Беларусь  
Учреждение образования   
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

КАФЕДРА ИНФОРМАТИКИ

Отчёт по лабораторной работе №1

По дисциплине «Архитектура вычислительных систем»  
По теме «ОПЕРАЦИИ СЛОЖЕНИЯ И ВЫЧИТАНИЯ С ФИКСИРОВАННОЙ ТОЧКОЙ»

Выполнил:  
студент гр. 153503  
Киселёва Елизавета Андреевна

Проверил:

Преподаватель

Калиновская Анастасия Александровна

Минск 2023

# 1 Цель работы

Рассмотреть представление чисел в прямом, обратном и дополнительном кодах. Изучить процессы выполнения сложения и вычитания над целыми числами с фиксированной точкой. Составить программу реализации алгоритмов сложения, вычитания.

**2 Постановка задачи**

***Задание к лабораторной работе 1***

Написать программу эмулятора АЛУ, реализующего операции сложения и вычитания с фиксированной точкой над двумя введенными числами, с возможностью пошагового выполнения алгоритмов.

# 3 Теоретические сведения

**3.1 Прямой код**

Существует несколько соглашений о едином формате представления как положительных, так и отрицательных чисел. Всех их объединяет то, что старший бит слова (с точки зрения европейца — самый левый, или бит, которому при представлении числа без знака должен быть приписан самый большой вес) является битом хранения знака или знаковым разрядом. Все последующие биты слова представляют значащие разряды числа, которые в каждом формате интерпретируются по-своему. Значение 1 в знаковом разряде интерпретируется как представление всем словом отрицательного числа.

|  |  |
| --- | --- |
| 0001001 = | +18 |
| 10010010 = | -18 |

Формат представления чисел в прямом коде неудобен для использования в вычислениях. Во-первых, сложение и вычитание положительных и отрицательных чисел выполняется по-разному, а потому требуется анализировать знаковые разряды операндов. Во-вторых, в прямом коде числу 0 соответствуют две кодовых комбинации:

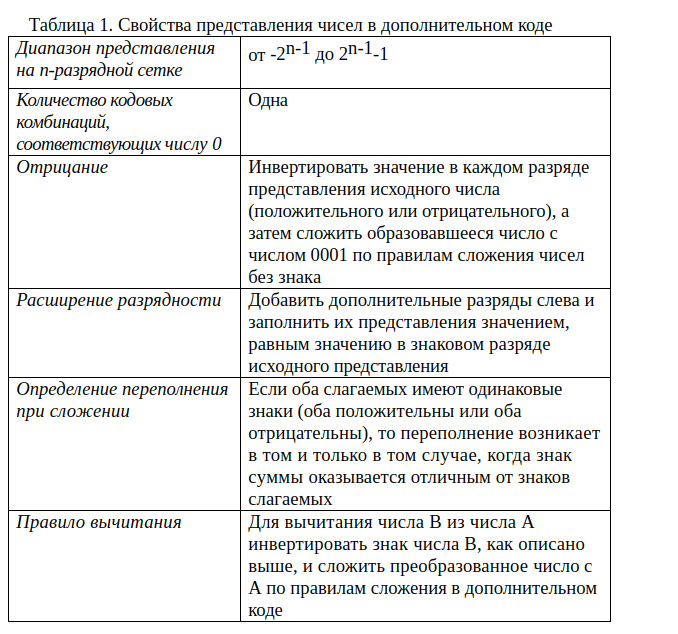
|  |  |
| --- | --- |
| 0000000= | +010 |
| 10000000= | -010 |
|  |  |

Это также неудобно, поскольку усложняется анализ результата на равенство нулю, а такая операция в программах встречается очень часто.

Из-за этих недостатков прямой код практически не применяется при peaлизации в АЛУ арифметических операций над целыми числами. Вместо этого более широкое применение находит другой формат, получивший наименование дополнительного кода.

## **3.2 Дополнительный код**

Как и в прямом, в дополнительном коде старший разряд в разрядной сетке отводится для представления знака числа. Остальные разряды интерпретируются не так, как в прямом коде. В табл. 1 перечислены основные свойства дополнительного кода и правила выполнения арифметических операций в дополнительном коде, которые мы рассмотрим в этом и следующем разделах.



В большинстве описаний дополнительного кода основное внимание уделяется технике формирования представления отрицательного числа по представлению соответствующего положительного, причем не приводится формальное доказательство работоспособности описанной схемы. Мы решили нарушить эту традицию, и в данном разделе, а также в следующем будем основываться на описании, в котором это представление рассматривается в терминах взвешенной суммы значений разрядов. Такой способ мы уже использовали выше при описании представления чисел без знака и целых чисел со знаком в прямом коде. Преимущество такой методики в том, что она не оставляет ни малейших сомнений в справедливости излагаемых правил выполнения арифметических операций в любых частных случаях.

## **3.3 Сложение и вычитание**

Рассмотрим операцию сложения на примере:

*Выполнить сложение чисел A = 53 и B = 14 в двоичной системе счисления.*

Переведем числа в двоичную систему счисления:

5310 **= 110101**2 ,  
1410 **= 1110**2 .

Запишем числа «A» и «B» столбиком, одно под другим, начиная с младших разрядов (нумерация разрядов начинается с нуля).

**Сложим** поразрядно числа «A» и «B» записывая результат в «C» начиная с младших разрядов. Весь процесс сложения наших чисел представлен в таблице 1.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Разряды | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| А | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| B | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| C | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |

Таблица 1. Сложение чисел 53 и 14

Теперь опишем как происходит описанный выше процесс сложения и вычитания чисел в нашем эмуляторе АЛУ:

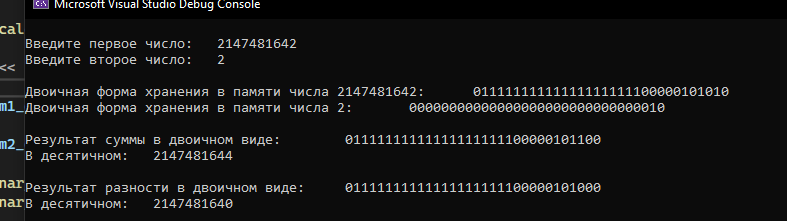
Изучим блоки эмулятора АЛУ. Главный блок представляет собой модуль сложения двух чисел. Процесс сложения происходит по правилам сложения беззнаковых чисел. Вычитание чисел достигается инвертированием второго числа в противоположный знак. Данный блок назовем *сумматором*.

Результат передается либо в один из регистров слагаемых (этот вариант показан на схеме), либо в третий регистр результата. Кроме кода результата сумматор формирует сигнал переполнения, который фиксируется в битовом флаге переполнения. Значение флага интерпретируется следующим образом: 0 — переполнение отсутствует, 1 – присутствует. При выполнении операции вычитания код вычитаемого, хранящийся перед началом операции в регистре В, передается на схему, выполняющую операцию отрицания, а уже с выхода этой схемы код поступает на вход сумматора.

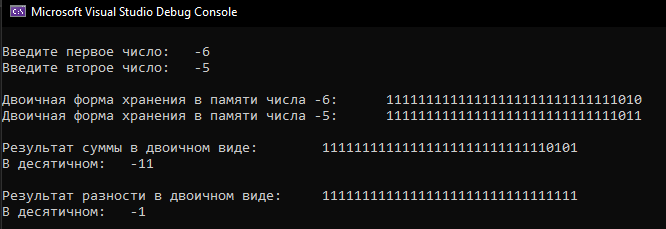
# 4 Пример работы программы

# 

# Тестовый пример 1



# Тестовый пример 2



# Тестовый пример 3

# 5 Выводы

В ходе выполнения лабораторной работы была написана программа, реализующая операции сложения и вычитания над числами с фиксированной точкой в дополнительном коде.

# Приложение 1. Текст программы

#include <iostream>

#include <regex>

#include <functional>

#include <string>

using namespace std;

typedef long long ll;

int result[33];

int number1[33];

int number2[33];

template<typename ReturnValue>

ReturnValue CorrectInput(const string& variable\_name, const regex& pattern,

function<ReturnValue(string& input)> converter) {

string value;

do {

cout << "Введите " << variable\_name << ":\t";

cin >> value;

} while (!regex\_match(value, pattern));

return converter(value);

}

void to\_binary(int mask[], ll n) {

int j = 31;

while (n > 0) {

mask[j--] = n % 2;

n >>= 1;

}

}

void to\_additional(int arr[]) {

for (int i = 0; i < 32; ++i)

arr[i] = (arr[i] + 1) % 2;

int j = 31;

arr[j]++;

while (arr[j] == 2) {

arr[j--] %= 2;

arr[j]++;

}

}

ll to\_dec(int arr[]) {

ll res = -(arr[0] \* pow(2, 31));

for (int i = 1; i < 32; ++i)

res += arr[i] \* pow(2, 31 - i);

return res;

}

void sum(int a[], int b[], bool issum) {

int j = 31;

int ram = 0;

while (j >= 0) {

result[j] = a[j] + b[j] + ram;

ram = result[j] / 2;

result[j] %= 2;

--j;

}

if ((result[0] != a[0] || result[0] != b[0]) && a[0] == b[0]) {

string op = issum ? " СУММЫ" : " РАЗНОСТИ";

cout << "\n\nПЕРЕПОЛНЕНИЕ" << op;

}

}

void substraction(int a[], int b[]) {

to\_additional(b);

sum(a, b, 0);

}

void print\_array(int arr[]) {

for (int i = 0; i < 32; ++i)

cout << arr[i];

}

int main() {

setlocale(LC\_ALL, "rus");

cout << '\n';

ll num1\_dec = CorrectInput<ll>(string("первое число"), regex("-?[0-9]\*"), [](string& s) {return atoi(s.c\_str()); });

ll num2\_dec = CorrectInput<ll>(string("второе число"), regex("-?[0-9]\*"), [](string& s) {return atoi(s.c\_str()); });

to\_binary(number1, abs(num1\_dec));

to\_binary(number2, abs(num2\_dec));

if (num1\_dec < 0)

to\_additional(number1);

if (num2\_dec < 0)

to\_additional(number2);

cout << "\nДвоичная форма хранения в памяти числа " << num1\_dec << ":\t";

print\_array(number1);

cout << "\nДвоичная форма хранения в памяти числа " << num2\_dec << ":\t";

print\_array(number2);

sum(number1, number2, true);

cout << "\n\nРезультат суммы в двоичном виде:\t";

print\_array(result);

cout << "\nВ десятичном:\t" << to\_dec(result);

substraction(number1, number2);

cout << "\n\nРезультат разности в двоичном виде:\t";

print\_array(result);

cout << "\nВ десятичном:\t" << to\_dec(result) << '\n';

return 0;

}