Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра информатики

Отчет по лабораторной работе №2

Эмуляция АЛУ. Операция умножения целых чисел

Студент: гр. 853504

Вечеринский Максим Сергеевич

Руководитель: старший преподаватель

Шиманский В.В.

Минск 2020

СОДЕРЖАНИЕ

1. Введение
2. Постановка задачи
3. Программная реализация
4. Выводы

Литература

Приложение 1 – Текст программы

1.Введение

1.1 Представление целых чисел.

В двоичной системе счисления числа представляются с помощью комбинации единиц и нулей, знака "минус" и знака разделяющей точки между целой и дробной частью числа. Например, десятичное число -1.312510 в двоичном виде будет выглядеть как -1001.01012. Но в компьютере мы не можем хранить и обрабатывать символы знака и разделяющей точки — для "машинного" представления чисел могут использоваться только двоичные цифры (0 и 1). Если операции выполняются только с неотрицательными числами, то формат представления очевиден. В машинном слове из 8 бит можно представить числа в интервале от 0 до 255. Пример:

|  |  |
| --- | --- |
| *00000000  =* | *0* |
| *00000000  =* | *0* |
| *00000001  =* | *1* |
| *00101001  =* | *41* |
| *10000000  =* | *128* |
| *11111111* | *255* |

В общем случае n-битовая последовательность двоичных цифр an-1an-2…a1a0 интерпретирована как целое число А, значение которого равно

1.2. 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.

00010010 = +18

10010010 = -18

Общее правило математически формулируется следующим образом:

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:

00000000 = +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 реализации в АЛУ арифметических операций над целыми числами. Вместо этого более широкое применение находит другой формат, получивший наименование дополнительного кода.

1.3. Двоичное умножение.

Двоичный множитель-это электронная схема, используемая в цифровой электронике, например в компьютере, для умножения двух двоичных чисел. Он построен с использованием двоичных сумматоров.

Для реализации цифрового множителя можно использовать различные компьютерные арифметические методы. Большинство методов включают вычисление набора частичных продуктов, а затем суммирование частичных продуктов вместе. Этот процесс аналогичен методу, который преподается младшим школьникам для проведения длительного умножения на базе целых чисел-10, но был модифицирован здесь для применения к системе счисления с базой-2 (двоичной).

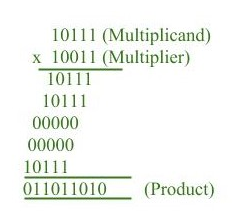


Рис.К.1 Простейшее умножение двоичных чисел

В процессе умножения мы рассматриваем последовательные биты множителя, сначала наименее значимый бит.

Если бит множителя равен 1, то множитель копируется вниз, иначе 0 копируются вниз.

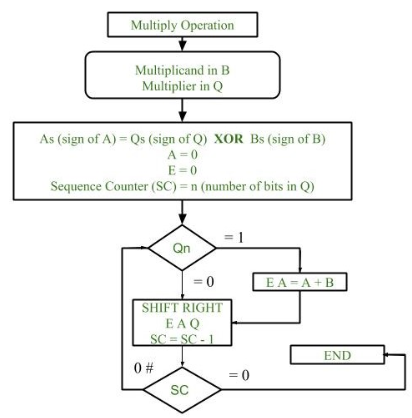


Рис.К2. Блок-схема умножения

1. Первоначально множитель хранится в регистре B, а множитель - в регистре Q.
2. Знак регистров B (Bs) и Q (Qs) сравниваются с использованием функции XOR (т. е. если оба знака одинаковы, то выход операции XOR равен 0, если не 1) и выход хранится в As (знак регистра).

Примечание: первоначально 0 назначается для регистрации триггера A и E. Счетчик последовательностей инициализируется значением n, n - это количество битов в Множителе.

1. Теперь проверяется наименее значимый бит множителя. Если он равен 1, то добавьте содержимое регистра а с Мультипликандом (регистр В) и результат будет назначен в регистр с битом переноса во флип-флопе E. содержимое E A Q смещается вправо на одну позицию, т. е. содержимое E смещается на самый значимый бит (MSB) A, а наименее значимый бит A смещается на самый значимый бит Q.
2. Если Qn = 0, то аналогично выполняется только операция сдвига вправо по содержимому E A Q.
3. Содержание счетчика последовательностей уменьшается на 1.
4. Проверьте содержимое счетчика последовательностей (SC), если оно равно 0, завершите процесс и конечный продукт присутствует в регистре A и Q, в противном случае повторите процесс.

2. Постановка задачи

2.1. Текст задания

Эмуляция АЛУ. Реализовать операцию умножения целых чисел.

2.2. Примечание к заданию

Реализовать ввод двух чисел. Вычислить сумму произведение в двоичной системе счисления и вывести результат на экран.

3. Программная реализация

3.1. С консоли вводятся два числа в десятичной системе счисления. Затем реализуется перевод их в двоичную систему счисления. В прямом коде реализуется произведение. Затем на экран выводится результат произведения в двоичной системе счисления и в десятичной.

3.2. Примеры

3.2.1. Тест для «a = 5, b = 6»

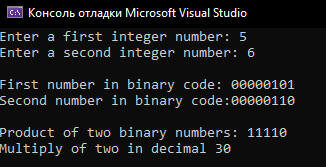


Рисунок 1. Скриншот для «a = 5, b = 6»

Ответ: Произведение = 30.

3.2.2. Тест для «a = 8, b = -3»

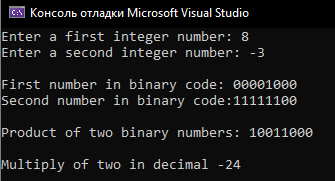


Рисунок 2. Скриншот для «a = 8, b = -3»

Ответ: Произведение = -24.

3.2.3. Тест для «a = 1000, b = 0»

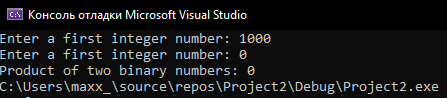


Рисунок 3. Скриншот для «a = 1000, b = 0»

Ответ: Произведение = 0.

3.2.4. Тест для «a = 200, b = 200»

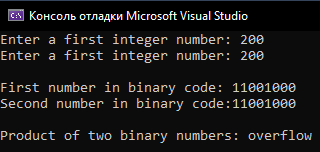


Рисунок 4. Скриншот для «a = 200, b = 200»

Ответ: Произведение = overflow

3.2.5. Тест для «a = -200, b = 250»

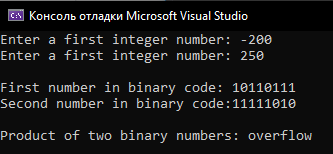


Рисунок 5. Скриншот для «a = -200, b = 250»

Ответ: Произведение = overflow

3.2.7. Резюме.

В своих примерах я старался рассмотреть максимально разнообразный набор случаев. Были произведение двух положительных чисел, двух отрицательных чисел, отрицательного и положительного чисел, нуля и положительного числа.

Также были рассмотрены экстремальный случай поведения программы: неправильный ввод.

.

4. Выводы

В ходе выполнения лабораторной работы ознакомился с понятиями прямого кода, произведения в прямом коде, изучил различные алгоритмы произведения бинарных чисел в прямом коде.

Так же рассмотрел работу арифметико-логического устройства (АЛУ), рассмотрел различные нюансы использования АЛУ. Для реализации использовал язык С++ и интегрированную среду разработки Visual Studio.

Литература

1. Волорова Н. А. Лабораторный практикум по курсу «Архитектура вычислительных систем» для студентов специальности «Информатика» /985-444-487-2 – Мн.: БГУИР, 2003. — 32 с.: ил.

2. Tanenbaum A. S. «Structured computer organization» /Vrije Universiteit Amsterdam, The Netherlands, 2013. — 810 c.:ил.

ПРИЛОЖЕНИЕ 1

Текст программы

#include <iostream>

#include <stdio.h>

#include <math.h>

//

#include <string>

#include <stdio.h>

#include <cstdlib>

#include <math.h>

#include <stdlib.h>

using namespace std;

int bits = 8;

long binary1, binary2, multiply = 0;

long int1, int2;

string sa, sb;

string dvoichnyiCode(long x) {

string value = "";

for (int i = 0; i < bits; i++)value += '0';

x = abs(x);

for (int i = 0; x > 0; i++) {

int j = bits - i - 1;

value[j] = (x % 2) ? '1' : '0';

x /= 2;

}

return value;

}

int binaryproduct(int binary1, int binary2)

{

int i = 0, remainder = 0, sum[20];

int binaryprod = 0;

while (binary1 != 0 || binary2 != 0)

{

sum[i++] = (binary1 % 10 + binary2 % 10 + remainder) % 2;

remainder = (binary1 % 10 + binary2 % 10 + remainder) / 2;

binary1 = binary1 / 10;

binary2 = binary2 / 10;

}

if (remainder != 0)

sum[i++] = remainder;

--i;

while (i >= 0)

binaryprod = binaryprod \* 10 + sum[i--];

return binaryprod;

}

string obratnyiCode(string value) {

value[0] = '1';

for (int i = 1; i < value.length(); i++) {

value[i] = (value[i] == '1') ? '0' : '1';

}

return value;

}

int getValue()

{

while (true)

{

int a;

cin >> a;

if (cin.fail())

{

cin.clear();

cin.ignore(1000, '\n');

cout << "You have entered wrong input. Try again" << endl;

}

else

{

cin.ignore(32676, '\n');

return a;

}

}

}

int main()

{

int digit, factor = 1;

cout << "Enter a first integer number: ";

int int1 = getValue();

cout << "Enter a second integer number: ";

int int2 = getValue();

if (int1 == 0 || int2 == 0) {

printf("Product of two binary numbers: 0");

return 0;

}

cout << "\nFirst number in binary code: ";

if (int1 < 0) {

cout << obratnyiCode(dvoichnyiCode(int1)) << endl;

sa = dvoichnyiCode(-int1);

}

else {

cout << dvoichnyiCode(int1) << endl;

sa = dvoichnyiCode(int1);

}

cout << "Second number in binary code:";

if (int2 < 0) {

cout << obratnyiCode(dvoichnyiCode(int2)) << endl;

sb = dvoichnyiCode(-int2);

}

else {

cout << dvoichnyiCode(int2) << endl;

sb = dvoichnyiCode(int2);

}

binary1 = stoi(sa);

binary2 = stoi(sb);

while (binary2 != 0)

{

digit = binary2 % 10;

if (digit == 1)

{

binary1 = binary1 \* factor;

multiply = binaryproduct(binary1, multiply);

}

else

binary1 = binary1 \* factor;

binary2 = binary2 / 10;

factor = 10;

}

int count = 0;

int numb = multiply;

while (numb) {

numb /= 10;

count++;

}

cout << "\nProduct of two binary numbers: ";

if (int1 \* int2 > 0) cout << multiply;

else

{

cout << '1';

for (int i = 1; i < bits - count; i++)

cout << '0';

cout << multiply << endl;

}

cout << endl;

cout << "Multiply of two in decimal " << int1 \* int2 << endl;

return 0;

}