Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра информатики

Отчет по лабораторной работе №3

Эмуляция АЛУ. Операция деления 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 деление

Если умножение выполняется путем многократных сдвигов и сложений, то деление, будучи операцией обратной умножению,— путем многократных сдвигов и вычитаний.

(ПРАВИЛЬНЫЕ ДРОБИ, БЕЗ ЦЕЛОГО.)

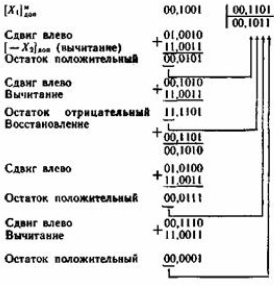
*При представлении чисел с фиксированной запятой деление возможно, если делимое по модулю меньше делителя, в противном случае произойдет переполнение разрядной сетки*.

Так же, как и при «ручном» делении, разряды частного при делении чисел на машине определяются (начиная со старшего) путем последовательного вычитания делителя из остатка, полученного от предыдущего вычитания. Однако здесь операция вычитания заменяется операцией сложения остатка с отрицательным делителем, представленным в обратном или дополнительном коде. Знак частного определяется сложением по модулю два кодов знаков делимого и делителя.

В машинах вместо сдвига делителя вправо осуществляется сдвиг остатка влево, что, по сути, ничего не изменяет.

При делении с восстановлением остатка отрицательный остаток восстанавливается суммированием с положительным делителем. Восстановленный остаток сдвигается влево на один разряд. Из сдвинутого остатка вновь вычитается делитель. По знаку полученного остатка определяется цифра очередного разряда частного. Процесс деления продолжается до получения заданного числа цифр частного, обеспечивающего необходимую точность результата.

Процесс деления начинается со сдвига делимого влево на один разряд, после чего к нему прибавляется делитель, представленный, например, в дополнительном модифицированном коде:



1) Определить знак частного суммированием по модулю два содержимых знаковых разрядов делимого и делителя.

2) Из делимого вычесть делитель. Если остаток отрицательный, перейти к пункту 3. В противном случае вычисление закончить (произошло переполнение).

3) Запомнить знак остатка.

4) Сдвинуть остаток на один разряд влево.

5) Присвоить делителю знак, обратный знаку остатка, запомненному в п. 2.

6) Сложить сдвинутый остаток и делитель (с учетом знака).

7) Присвоить цифре частного значение, противоположное коду знака остатка.

8) Повторять выполнение пунктов 3—7 до тех пор, пока не будет обеспечена требуемая точность вычисления частного.

Деление целых ненулевых n-разрядных (не считая знаковых разрядов) чисел А:В, представленных в прямом (для простоты) коде, приводит к получению целого частного С и целого остатка 0, которому присваивается знак делимого; знак частного вычисляется как сумма по модулю два операндов А и В.

Деление выполняется в следующей последовательности.

1) Делитель В сдвигается влево (нормализуется), так чтобы в старшем информационном разряде оказалась 1;подсчитывается количество сдвигов S; частное от деления может быть не более (S + 1) разрядов, не равных нулю.

2) Выполняется (S+1) цикл деления модулей |А| на IB’l где В' — нормализованное В, в результате находится(S+ 1) разряд частного, начиная со старшего из (S+ 1)младших.

3) Полученный в последнем цикле деления остаток Rs+1, если он положительный, сдвигается вправо на S разрядов; если же Rs+1 < 0 (отрицательный), то остаток восстанавливается: к нему добавляется |В'|, т. е.[Rs+1]вост = Rs+1+|B'|. После этого выполняется сдвиг вправо на S разрядов. В результате получается целый остаток от деления.



2. Постановка задачи

2.1. Текст задания

Эмуляция АЛУ. Реализовать операцию деления целых чисел.

2.2. Примечание к заданию

Реализовать ввод двух чисел. Вычислить частное и остаток 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. Работа выполняется только для 8-ми разрядных значений.

3.2. Примеры

3.2.1. Тест для «делитель = 5, делимое = 135»

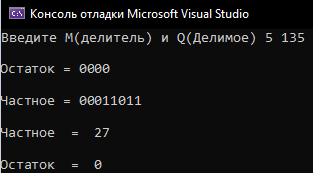


Рисунок 1. Скриншот для «делитель = 5, делимое = 135»

Ответ: Частное = 27, остаток = 0.

3.2.2. Тест для «делитель = 3, делимое = 55»

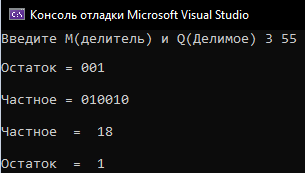


Рисунок 2. Скриншот для «делитель = 3, делимое = 55»

Ответ: Частное = 18, остаток = 1.

3.2.3. Тест для «делитель = -4, делимое = 22»

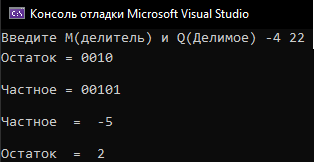


Рисунок 3. Скриншот для «делитель = -4, делимое = 22»

Ответ: Частное = -5, остаток = 2.

3.2.4. Тест для «делитель = 4, делимое = 10» с восстановлением остатка.

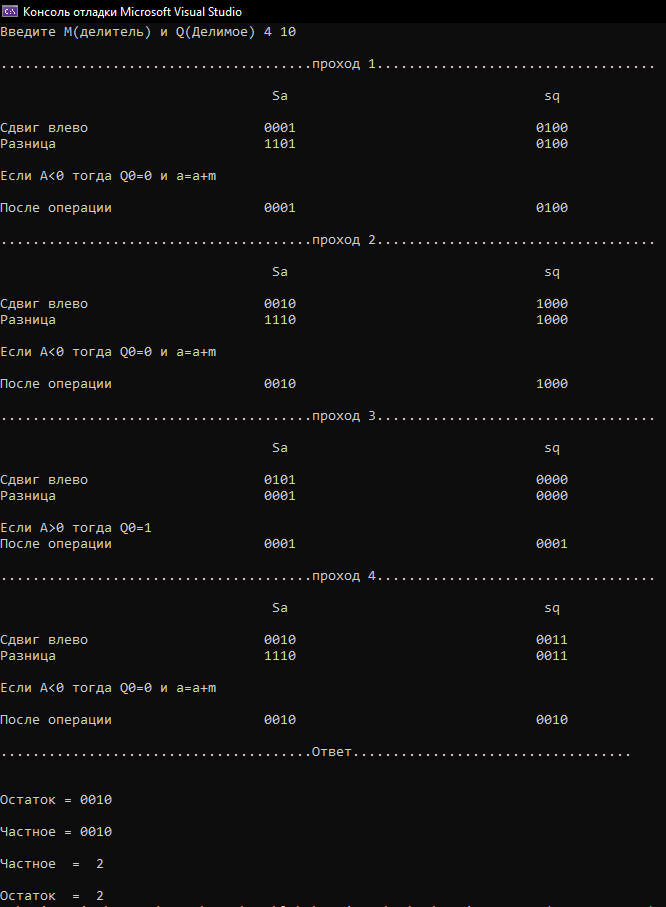


Рисунок 4. Скриншот для «делитель = 4, делимое = 10»

Ответ: Частное = 2, остаток = 2.

3.2.5. Тест для «делитель = 0, делимое = 4»

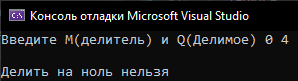


Рисунок 5. Скриншот для «делитель = 0, делимое = 4»

Ответ: на ноль делить нельзя

3.2.6. Тест для «делитель = 6, делимое = 2» с восстановлением остатка

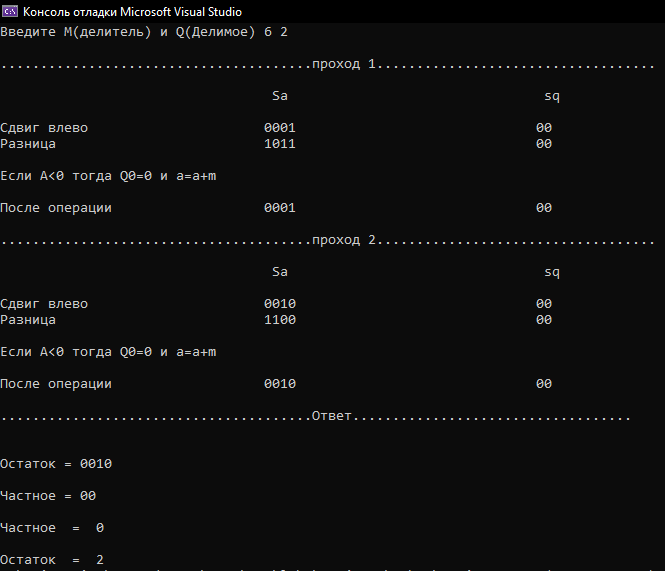


Рисунок 6. Скриншот для «делитель = 6, делимое = 2»

Ответ: Частное = 0, остаток = 2

3.2.7. Тест для «делитель =100000000000000000000000000000000000000000000, делимое = 1000» с восстановлением остатка

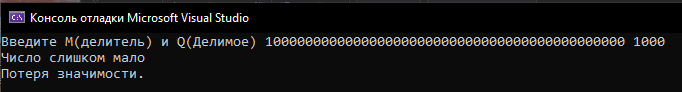


Рисунок 7. Скриншот для «делитель = 100000000000000000000000000000000000000000000, делимое = 1000»

Ответ: Значение слишком мало, потеря значимости.

3.2.8. Резюме.

В своих примерах я старался рассмотреть максимально разнообразный набор случаев. Частное 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, деления в прямом коде, изучил различные алгоритмы деления 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<stdlib.h>

#include<process.h>

#include<stdio.h>

#include<math.h>

#include<conio.h>

#include <clocale>

typedef struct stack

{

int top;

int a[100];

}c;

void lshift(c\* sq, c\* sa);

void comp(c\* sm);

void diff(c\* sa, c\* sm);

void sum(c\* sa, c\* sm);

void push(int b, c\* s);

void bin(int n, c\* s);

int convert(c\* s);

void display(c\* s);

int q1 = 0;

int main()

{

setlocale(LC\_ALL, "Russian");

c sm, sq, sa, temp;

int tempm, tempq;

sm.top = -1;

sq.top = -1;

temp.top = -1;

sa.top = -1;

int i = 0;

for (i = 0; i < 100; i++)

{

sm.a[i] = 0;

sq.a[i] = 0;

sa.a[i] = 0;

}

int q = 0, m = 0, steps;

printf("Введите M(делитель) и Q(Делимое) ");

scanf\_s("%d%d", &m, &q);

if (m == 0)

{

printf("\nДелить на ноль нельзя\n");

return 0;

}

if (q / m < 0.0000000000000000000000000000000001)

{

printf("Число слишком мало");

printf("\nПотеря значимости.\n\n");

return 0;

}

tempm = m;

tempq = q;

if (m < 0)

{

m = -m;

}

if (q < 0)

{

q = -q;

}

bin(m, &sm);

bin(q, &sq);

sm.top += 1;

sa.top = sm.top;

steps = sq.top;

for (i = 0; i <= steps; i++)

{

printf("\n.......................................проход %d................................... \n ", i + 1);

printf("\n");

lshift(&sq, &sa);

printf(" Sa sq");

printf("\n\nСдвиг влево");

printf(" ");

display(&sa);

printf(" ");

display(&sq);

diff(&sa, &sm);

printf("\nРазница ");

printf(" ");

display(&sa);

printf(" ");

display(&sq);

if (sa.a[sa.top] == 1)

{

printf("\n\nЕсли A<0 тогда Q0=0 и a=a+m ");

sq.a[0] = 0;

sum(&sa, &sm);

printf("\n\nПосле операции");

printf(" ");

display(&sa);

printf(" ");

display(&sq);

}

else

{

printf("\n\nЕсли A>0 тогда Q0=1 ");

sq.a[0] = 1;

printf("\nПосле операции");

printf(" ");

display(&sa);

printf(" ");

display(&sq);

}

printf("\n");

}

printf("\n.......................................Ответ................................... \n ");

printf("\n\nОстаток = ");

display(&sa);

printf("\n\nЧастное = ");

display(&sq);

if (tempm < 0 || tempq < 0)

{

printf("\n\nЧастное = -%d ", convert(&sq));

printf("\n\nОстаток = %d ", convert(&sa));

}

else

{

printf("\n\nЧастное = %d ", convert(&sq));

printf("\n\nОстаток = %d ", convert(&sa));

}

return 0;

}

void lshift(c\* sq, c\* sa)

{

int i, temp, ta = sa->a[sa->top], tq = sq->a[sq->top];

for (i = (sq->top); i > 0; i--)

{

sq->a[i] = sq->a[i - 1];

}

for (i = sa->top; i > 0; i--)

{

sa->a[i] = sa->a[i - 1];

}

sa->a[i] = tq;

}

void diff(c\* sa, c\* sm)

{

c temp;

temp.top = -1;

int i;

temp.top = sm->top;

for (i = sm->top; i >= 0; i--)

{

temp.a[i] = sm->a[i];

}

comp(&temp);

int carry = 0;

for (i = 0; i <= sm->top; i++)

{

if ((sa->a[i] == 0) && temp.a[i] == 0 && carry == 0)

{

sa->a[i] = 0;

}

else if (((sa->a[i] == 0) && (temp.a[i] == 0)) && carry != 0)

{

sa->a[i] = 1;

carry = 0;

}

else if (((sa->a[i] == 0) && carry == 0) && (temp.a[i] != 0))

{

sa->a[i] = 1;

}

else if (((carry == 0) && (temp.a[i] == 0)) && sa->a[i] != 0)

{

sa->a[i] = 1;

}

else if ((carry == 0) && ((temp.a[i] != 0) && sa->a[i] != 0))

{

sa->a[i] = 0;

carry = 1;

}

else if ((carry != 0) && ((temp.a[i] == 0) && sa->a[i] != 0))

{

sa->a[i] = 0;

carry = 1;

}

else if ((carry != 0) && ((temp.a[i] != 0) && sa->a[i] == 0))

{

sa->a[i] = 0;

carry = 1;

}

else

{

sa->a[i] = 1;

carry = 1;

}

}

}

void comp(c\* sm)

{

int i;

for (i = 0; i <= sm->top; i++)

{

if ((sm->a[i]) == 0)

sm->a[i] = 1;

else

sm->a[i] = 0;

}

for (i = 0; i <= sm->top; i++)

{

if (sm->a[i] == 0)

{

sm->a[i] = 1;

break;

}

else

{

sm->a[i] = 0;

}

}

}

void sum(c\* sa, c\* sm)

{

int carry = 0, i;

for (i = 0; i <= sm->top; i++)

{

if ((sa->a[i] == 0) && sm->a[i] == 0 && carry == 0)

{

sa->a[i] = 0;

}

else if (((sa->a[i] == 0) && (sm->a[i] == 0)) && carry != 0)

{

sa->a[i] = 1;

carry = 0;

}

else if (((sa->a[i] == 0) && carry == 0) && (sm->a[i] != 0))

{

sa->a[i] = 1;

}

else if (((carry == 0) && (sm->a[i] == 0)) && sa->a[i] != 0)

{

sa->a[i] = 1;

}

else if ((carry == 0) && ((sm->a[i] != 0) && sa->a[i] != 0))

{

sa->a[i] = 0;

carry = 1;

}

else if ((carry != 0) && ((sm->a[i] == 0) && sa->a[i] != 0))

{

sa->a[i] = 0;

carry = 1;

}

else if ((carry != 0) && ((sm->a[i] != 0) && sa->a[i] == 0))

{

sa->a[i] = 0;

carry = 1;

}

else

{

sa->a[i] = 1;

carry = 1;

}

}

}

void push(int b, c\* s)

{

s->a[++(s->top)] = b;

}

void bin(int n, c\* s)

{

int rem;

for (int i = n; i != 0; i = i / 2)

{

rem = i % 2;

push(rem, s);

}

}

int convert(c\* s)

{

int i, sum = 0;

for (i = s->top; i >= 0; i--)

{

sum = sum + (s->a[i]) \* pow(2, i);

}

return sum;

}

void display(c\* s)

{

for (int i = s->top; i >= 0; i--)

{

printf("%d", s->a[i]);

}

}