МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра вычислительной техники

УТВЕРЖДАЮ
Проректор по учебной работе
_____О.Г. Локтионова
«__» _____ 2016г.

МОДЕЛИРОВАНИЕ МИКРОПРОГРАММ ВЫПОЛНЕНИЯ АРИФМЕТИЧЕСКИХ ОПЕРАЦИЙ В ДВОИЧНОЙ СИСТЕМЕ СЧИСЛЕНИЯ

Методические рекомендации к лабораторным работам
для студентов направления 09.03.01

Курск 2016

2

УДК 004.04

Составители: И.Е. Чернецкая, Г.В. Петрухин

Рецензент
Кандидат технических наук, доцент *Ю.А. Халин*

Моделирование микропрограмм выполнения арифметических операций в двоичной системе счисления: методические рекомендации к лабораторным работам/ Юго-Зап. гос. ун-т; сост.; И.Е. Чернецкая, Г.В. Петрухин. – Курск, 2016. - 32 с.: - ил. 5 , табл. 1.– Библиогр.: с. 32

Содержат сведения по вопросам моделирования микропрограмм выполнения арифметических операций. Представлены теоретические материалы выполнения операции сложения и вычитания чисел в двоичной системе счисления, заданных в прямых, дополнительных и обратных кодах; операции умножения и деления. Излагаются методические указания по подготовке и выполнению лабораторных работ.

Методические рекомендации соответствуют рабочей программе дисциплины «Теория вычислительных процессов».

Предназначены для студентов направления 09.03.01 очной и заочной форм обучения.

Текст печатается в авторской редакции

Подписано в печать 30.12.16 Формат 60*84 1/16.
Усл. печ. л. . Уч.-изд. л. Тираж 50 экз. Заказ Бесплатно.
Юго-Западный государственный университет.
305040 Курск, ул. 50 лет Октября, 94.

Оглавление

Цель работы 4
1 Теоретические основы 4
2 Порядок выполнения лабораторной работы 23
3 Содержание отчета 28
4 Контрольные вопросы 29
5 Варианты задания на лабораторную работу 31
Список литературы 32

Цель работы

Целью работы является моделирование микропрограмм выполнения арифметических операций в двоичной системе счисления.

1 Теоретические основы

Интенсивное развитие информатики с ее алгоритмическими и техническими проблемами основывается на первичных построениях, связанных с разработкой алгоритмов и их аппаратной поддержки для выполнения арифметических операций в рамках архитектуры фон Неймана.

Архитектура фон Неймана отражает последовательные способы выполнения операций с параллелизмом работы арифметико-логического устройства в соответствии с заданной разрядной сеткой и формой представления чисел. Еще одной особенностью рассматриваемой архитектуры является поадресное обращение к запоминающему устройству с обменом по длине разрядной сетки. Два основных достоинства машин с указанной архитектурой:

- простота структуры технических средств;
- универсальность.

В данном методическом указании будет рассматриваться алгоритм, который выполняется с использованием единственного параллельного сумматора, регистров общего назначения и регистров с многофункциональной аппаратной поддержкой (сдвиг, кодовые преобразования, счетчик и т.д). В качестве примера: операнды и результаты операций, представлены в прямых кодах, преобразовываться могут оба операнда, в сумматоре используется дополнительный код.).

Прямой код числа A=-0,001101 – машинное изображение $[A]_{пр}=1,001101$ (все цифровые разряды отрицательного числа остаются

неизменными, а в знаковой части записывается единица). Положительное число в прямом коде не меняет своего изображения, т.е правила преобразования чисел в прямой код записываются в следующем виде

$$[A]_{пр} = \begin{cases} A, & если\ A \geq 0, \\ 1 + |A|, & если\ A < 0. \end{cases}$$

Дополнительный код числа является математическим дополнением основанию системы счисления

$$|A| + [A]_{д} = q\ ,$$

где q – основание системы счисления.

Положительные числа не меняют своего изображения в дополнительном коде и правила преобразования в дополнительный код записываются следующим образом:

$$[A]_{д} = \begin{cases} A, & если\ A \geq 0, \\ q - |A|, & если\ A < 0. \end{cases}$$

Дополнительным кодом числа *A=-0,101110* является число $[A]_{д}=1,010010$, т.е. инвертируется каждый значащий разряд, за исключением последнего значащего разряда, для которого инвертирование не производится.

Операция выполняется в три этапа: вначале в нужных случаях преобразуются один или оба операнда из прямого в дополнительный код, производится сложение на сумматоре и затем полученный результат преобразуется из дополнительного в прямой код.

Операнд *A* преобразуется при операциях *A+B, A-B*, если *A* – отрицательно.

Операнд *B* преобразуется при операциях *A+B* если *B*- отрицательно, и при операции *A-B*, если *B* – положительно.

Отрицательные числа поступают на сумматор в дополнительном коде.

Рассмотрим возможные случаи, в которых не возникает переполнения разрядной сетки, что позволяет складывать автоматные изображения чисел по правилам двоичной арифметики, не разделяя знаковую и цифровую части:

1) $A>0,\ B>0,\ A+B<1$.

Так как $[A]_{\partial} = A$, $[B]_{\partial} = B$ то $[A]_{\partial} + [B]_{\partial} = A + B = [A + B]_{\partial}$ - результат положительный.

2) $A<0,\ B>0,|A|>B$.

Здесь $[A]_{\partial} = A + q$, $[B]_{\partial} = B$. Таким образом, в результате арифметического сложения получится, что $[A]_{\partial} + [B]_{\partial} = A + B + q$ - результат отрицательный.

3) $A<0,\ B>0,|A|<B$.

Здесь $[A]_{\partial} = A + q$, $[B]_{\partial} = B + q$. Таким образом, в результате арифметического сложения получится, что значение этой суммы больше q, и появляется единица переноса из знакового разряда, что равносильно изъятию из суммы q единиц и результат равен $[A]_{\partial} + [B]_{\partial} = A + B$.

4) $A<0,\ B<0,|A+B|<1$

Здесь $[A]_{\partial} = A + q$, $[B]_{\partial} = B + q$. В результате арифметического сложения получится:

$[A]_{\partial} + [B]_{\partial} = A + B + q + q = [A + B]_{\partial}$ - результат отрицательный. В данном случае появляется единица переноса из знакового разряда.

Сумматор обратного кода, оперирующий изображениями чисел в обратных кодах, имеет циклический перенос из знакового разряда в младший разряд цифровой части.

Обратный код числа A=0,010101 это машинное изображение числа $[A]_{об}=1,101010$, где инвертируются все цифры как знаковой части, так и цифровой., т.е. обратный код двоичного числа есть инверсное изображение

самого числа, в котором все разряды исходного числа принимают инверсное (обратное) значение, т.е. все нули заменяются на единицы, а все единицы – на нули.

Правила преобразования чисел в обратный код можно сформулировать следующим образом:

$$[A]_{об} = \begin{cases} A, & если\ \ A \geq 0, \\ 1q - q^{-n} + |A|, & если\ \ A < 0, \end{cases}$$

где */A/* - абсолютная величина *A;*

n-количество разрядов после запятой в изображении числа.

При проектировании цифровых автоматов необходимо учитывать неоднозначное изображение нуля в обратном коде: *+0* изображается *0,00…0*, а *–0* изображается *1,11…1*.

Операнд *A* преобразуется при операциях *A+B, A-B*, если *A* – отрицательно.

Операнд *B* преобразуется при операциях *A+B* если *B*- отрицательно, и при операции *A-B*, если *B* – положительно.

Отрицательные числа поступают на сумматор в инверсном коде.

Правила сложения чисел с использованием обратного кода.

Существуют следующие случаи:

5) *A>0, B>0, A+B<1*.

$$[A]_{об} + [B]_{об} = A + B = [A + B]_{об}$$

6) *A<0, B>0,| A|>B*.

Здесь $[A]_{об} = A + q - q^{-n}$, $[B]_{об} = B$. Таким образом, в результате арифметического сложения получится, что $[A]_{об} + [B]_{об} = A + B + q - q^{-n} = [A + B]_{об}$ - результат отрицательный.

7) *A<0, B>0,| A|<B.*

Здесь $[A]_{об} = A + q - q^{-n}$. Таким образом, в результате арифметического сложения получится

$$[A]_{об} + [B]_{об} = q - q^{-n} + A + B,$$

правая часть этого выражения становится больше *q,*. Так как имеется цепь переноса из знакового разряда в младший разряд (величина переноса из знакового разряда равна $q - q^{-n}$), то $[A]_{об} + [B]_{об} = [A + B]_{об}$, результат положительный.

8) *A<0, B<0,| A+B|<1*

Здесь $[A]_{об} = A + q - q^{-n}$, $[B]_{об} = B + q - q^{-n}$. В результате арифметического сложения получится:

$[A]_{об} + [B]_{об} = A + B + q - q^{-n} + q - q^{-n}$ В данном случае появляется единица переноса из знакового разряда, что равносильно изъятию из суммы величины $q - q^{-n}$, т.е. $[A]_{об} + [B]_{об} = [A + B]_{об}$.

9) |A|=B, A<0, B>0

Тогда $[A]_{об} = A + q - q^{-n}$. Следовательно

$[A]_{об} + [B]_{об} = A + B + q - q^{-n} = q - q^{-n}$ - результат указывает на то, что сумма равна нулю (получим одно из изображений нуля в обратном коде).

Пример 1.

Найти сумму чисел *A=-0,0101* и *B=0,0111.*

Решение:

$$\begin{array}{r} [A]_{об} = 1{,}1010 \\ [B]_{об} = 0{,}0111 \\ \hline 0{,}0001 \\ + \qquad\quad 1 \\ \hline [C]_{об} = 0{,}0010 \end{array}$$

Ответ: *C=0,0010.*

Пример 2.

Найти сумму чисел *A=0,0101* и *B=-0,0111.*

Решение:

$$\begin{array}{r} [A]_{об} = 0{,}0101 \\ [B]_{об} = 1{,}1000 \\ \hline [C]_{об} = 1{,}1101 \end{array}.$$

Ответ: *C=-0,0010.*

Переполнение разрядной сетки

При сложении чисел одинакового знака, представленных в форме с фиксированной запятой, может возникнуть переполнение разрядной сетки.

Признак переполнения разрядной сетки сумматора обратного кода – знак результата, противоположный знакам операндов.

A=0,0111 и *B=0,1101.*

$$\begin{array}{r} [A]_{об} = 0{,}0111 \\ [B]_{об} = 0{,}1101 \\ \hline [C]_{об} \neq 1{,}0100 \end{array}.$$

A=-0,0110 и *B=-0, 1101.*

$$\begin{array}{r} [A]_{об} = 1{,}1001 \\ [B]_{об} = 1{,}0010 \\ \hline [C]_{об} \neq 0{,}1011 \end{array}.$$

Рисунок 1 – Изображения чисел

Для обнаружения переполнения разрядной сетки в составе автомата должны быть предусмотрены аппаратные средства, автоматически вырабатывающие признак переполнения – сигнал φ. Чтобы обнаружить переполнение разрядной сетки при использовании инверсных кодов при операции сложения, вводится вспомогательный разряд в знаковую часть изображения числа, который называют разрядом переполнения (см. рисунок 1). Также представлены изображения положительного и отрицательного чисел (см. рисунок 1 а,б).

Такое представление числа называется модифицированным. В случае переполнения сигнал φ=1.

$$\overline{Sg_1} \wedge Sg_2 = \overline{1},\ Sg_1 \wedge Sg_2 = 1.$$

В остальных случаях φ=0.

$[A]_{об}^{м} = 00{,}0111 - \textit{модифицированное изображение } A;$

$[B]_{об}^{м} = 00{,}1101 - \textit{модифицированное изображение } B;$

$[C]_{об} = 01{,}0100 - 01 - \textit{признак переполнения взнаковых разрядах}.$

$[A]_{об}^{м} = 11{,}1001 - \textit{модифицированное изображение } A;$

$[B]_{об}^{м} = 11{,}0010 - \textit{модифицированное изображение } B;$

$\overline{[C]_{об}} = 10{,}1100 - 10 - \textit{признак переполнения взнаковых разрядах}.$

Операция умножения в современных ЭВМ чаще всего выполняется суммированием сдвинутых на один или несколько разрядов частичных произведений, каждое из которых является результатом умножения множимого на соответствующий разряд (разряды) мантиссы.

При точном умножении двух чисел количество значащих цифр произведения может в пределе достичь двойного количества значащих цифр сомножителей. Еще сложнее возникает ситуация при умножении нескольких чисел. Поэтому в произведении только в отдельных случаях используют двойное количество разрядов, обычно же ограничиваются количеством разрядов, которое имели сомножители. Здесь учитывается то обстоятельство, что правила приближенных вычислений рекомендуют оставлять в произведении столько же значащих цифр, сколько их содержится в наименее точном из сомножителей. Младшие цифры результата при этом отбрасываются, а старшие обычно округляются по известным правилам, с тем, чтобы ошибка произведения стала знакопеременной и ее математическое ожидание было равно 0 с учетом равновероятности любых значений отброшенных младших разрядов.

Наиболее просто операция умножения в ЭВМ выполняется в прямом коде. При этом на первом этапе определяется знак произведения путем сложения знаковых цифр сомножителей по модулю 2.

Затем производится перемножение модулей сомножителей по правилам арифметики согласно двоичной таблице умножения. Результату присваивается полученный ранее знак, так как умножение производится в двоичной системе счисления частные произведения либо равны 0, либо самому сомножителю, сдвинутому на соответствующее количество разрядов.

Процессом накопления СЧП можно управлять с помощью цифр множителя в соответствии с выражением:

$$C = A \cdot B = A\sum b_i 2^i = A \cdot b_1 2^{-1} + A \cdot b_2 2^{-2} + ... + A \cdot b_n 2^{-n}$$

Управление процессом умножения может начинаться как с младших разрядов множителя, так и со старших.

При этом полную сумму можно получить двумя путями:

1) Сдвигом множимого на требуемое количество разрядов и добавлением полученного очередного частного произведения к ранее накопленной сумме;

2) Сдвигом суммы ранее полученных частных произведений на каждом шаге на один разряд и последующим добавлением множимого либо 0.

Основываясь на вышеизложенном, можно создать четыре варианта схем машинного умножения.

1) Умножение младшими разрядами множителя со сдвигом суммы частичных произведений вправо.

Полученное выражение можно представить в виде схемы Горнера для вычисления полиномов.

$$C = \left(\left(\left(...\left(\left(0 + A + A \cdot b_n\right)2^{-1} + A \cdot b_{n-1}\right)2^{-1} + ... + A \cdot b_{n-i}\right)2^{-1} + ... + A \cdot b_2\right)2^{-1} + ... + A \cdot b_1\right)2^{-1}$$

Это выражение может быть сведено к n-кратному выполнению цикла:

$Ci+1=(Ci+Abn-i)2^{-1}$ при начальных $i=0$, $Co=0$.

В каждом цикле множимое либо добавляется к СЧП (если $bi=1$), либо нет ($bi=0$), после чего сумма умножается на 2^{-1}, т.е. сдвигается на один разряд вправо. После окончания n-го цикла образуется искомое произведение, т.е. $Cn=C=AB$.

Очередную цифру множителя, управляющую суммированием частичных произведений, удобнее всего снимать с младшего разряда регистра множителя, в котором в каждом цикле производится сдвиг содержимого на один разряд вправо.

Реализация данного способа требует $2n$ разрядного сумматора. Однако, если дополнительные разряды сумматора исключить, то произведение будет вычисляться с тем же количеством разрядом, что и сомножители. Однако для округления результата желательно удлинить сумматор на один дополнительный разряд, чтобы после вычисления произведения прибавить к дополнительному разряду единицу и вывести результат без дополнительного разряда. Количество разрядов сумматора может быть уменьшенного вдвое также при вычислении полноразрядного произведения, если использовать освобождающиеся разряды регистра множителя, соединив при этом младший разряд регистра сумматора со старшим разрядом регистра множителя и объединив их в общий сдвиговый регистр.

Пример.

Найти произведение чисел $A=0,0111$ и $B=0,0011$.

Решение:

0,00111 – *множимое*

0,00011 – *множитель*

0,00111 – *частное произведение после* 1 – *го цикла*

0,00111 – *сдвиг СЧП вправо*

0,10101 – *результат умножения*

Ответ: *C=0,10101*

2) Умножение младшими разрядами множителя со сдвигом множимого влево.

Выражение представим в виде:

$C=2^{-n}(Abn+2Abn-1+...+2^{i}Abn-i+...+2^{n-1}Abi)$.

Вычисление этого выражения сводится к кратному выполнению цикла:

Ci+1=Ci+Aibn-i, где *Ai=2ai-1*, при начальных значениях *i=0, Co=0, Ao=A.*

В каждом цикле умножения множимое сдвигается на один разряд влево и либо прибавляется к СЧП (при *bi=1*), либо нет (*bi=0*).

Пример.

Найти произведение чисел *A=0,0111* и *B=0,0011.*

Решение:

0,00111 – *множимое*

0,00011 – *множитель*

0,00111 – *частное произведение после* 1 – *го цикла*

0,00111 – *сдвиг множимого влево*

0,10101 – *результат умножения*

Ответ: *C=0,10101*

3) Умножение старшими разрядами множителя со сдвигом СЧП влево:

Выражение преобразуем к виду:

$C=2^{-(n+1)}$ ((...(...((0+Abi)2+Ab_2)2+...+Abi)2+...+Abn-$_1$)2+Abn)2

При этом умножение сводится к n-кратному повторению цикла Ci+1=(Ci+Abi$_{+1}$)2 с начальными значениями i=0, Co=0.

Тогда управление умножением будет производиться цифрами множителя, начиная со старших разрядов. СЧП в каждом цикле будет сдвигаться на один разряд влево.

Пример.

Найти произведение чисел *A=0,0111* и *B=-0,0011.*

Решение:

0,00111 – *множимое*

0,00011 – *множитель*

0,00000 – *частное произведение после* 1 – *го цикла*

0,00000 – *сдвиг СЧП влево*

0,00000 – *сдвиг СЧП влево*

0,00111 – *прибавление множимого*

0,00111 – *сдвиг СЧП влево*

0,10101 – *результат умножения*

Ответ: *C=0,10101*

4) Умножение старшими разрядами множителя со сдвигом множимого вправо.

Выражение представим в виде

C=A2^{-1}bi+A^{-2}b2+...+A2^{-i}bi+...+A2^{-n}b и тогда вычисление произведения может быть сведено к n-кратному выполнению цикла:

Ai$_{+1}$=Ai 2^{-1}

Ci$_{+1}$=Ci+Ai$_{+1}$bi$_{+1}$ при начальных i=0 Ao=A Co=0. Т.е. в каждом числе множимое сдвигается на один разряд вправо и в зависимости от значения управляющего разряда множителя либо прибавляется к СЧП, либо нет.

Пример.

Найти произведение чисел *A=0,0111* и *B=0,0011*.

Решение:

$0{,}00111$ – *множимое*

$0{,}00011$ – *множитель*

$0{,}00111$ – *частное произведение после* 4 – *го цикла*

$0{,}00111$ – *сдвиг множимого вправо*

$0{,}10101$ – *результат умножения*

Ответ: *C=0,10101*

Анализ приведенных схем умножения показывает, что

1) Длительность процесса умножения по любой схеме составляет и циклов:

$$Ty = n\tau_ц$$

Однако длительность циклов в разных схемах неодинакова. Так, во 2 и 4 схемах $\tau_ц = \tau_{см}$, и учитывая, что $\tau_{см} > \tau_{сдв}$, эти схемы позволяют ускорить процесс выполнения операции умножения за счет совмещения операции сложения ЧП и сдвигов множимого.

2) По количеству оборудования предпочтение следует отдать первой, а потом 3 схеме. Однако, с учетом, что обычно требуется n-разрядный результат, схема 4 может быть также значительно упрощена.

3) Схемы 3 и 4 легче приспособить для выполнения операции деления.

4) Схемы 2 и 4 позволяют не устанавливать регистр СЧП в 0 перед выполнением следующей операции.

Отсюда можно заключить, что наиболее удобными для применения в ЭВМ являются 1 и 4 схемы умножения.

Основные методы деления

Деление в ЭВМ, так же, как умножение проще всего выполнять в прямом коде. Но в отличие от умножения дробных сомножителей, где не может возникнуть переполнение разрядной сетки, при делении правильных дробей такое переполнение возможно в случае, когда делимое больше делителя. Признаком переполнения является появление единицы в знаковом разряде частного, что грубо искажает результат.

При делении чисел с фиксированной запятой в АЛУ формируется частное *Y,* которое в общем случае является приближенным частным, то есть

$$A/B \approx Y .$$

Это связано, в первую очередь, с тем, что мы берем только *n* старших разрядов точного частного, исходя из конечной длины регистров и того, что операция деления является операцией, порождающей иррациональность.

Знак частного как и при выполнении операции умножения получается с использованием логической функции «сложение по модулю 2» булевых переменных, являющихся знаками делимого и делителя. Частное определяется путем деления модулей исходных чисел.

При этом во избежание переполнения разрядной сетки должно соблюдаться условие $|A|<|B|$.

Деление представляет собой циклический процесс: на каждом *i-м* цикле определяется очередная цифра частного y_i.

Пронумеруем разряды частного:

$$Y = A/B = 0, y_1, y_2, ..., y_{n+2} .$$

Общее количество циклов деления равно *n+2*, потому что для округления требуется *n+1* разряд и один разряд под проверку переполнения.

Первый цикл отличается от других тем, что в нем проверяется неравенство $A \geq B$, на втором цикле определяется y_2, на третьем - y_3 и т.д.

Рассмотрим процесс деления чисел с фиксированной запятой.

Основными двумя методами деления являются: метод деления чисел с восстановлением остатков и метод без восстановления остатка.

Деление чисел с восстановлением остатков

Пусть требуется разделить A на B с точностью до i-го разряда.

Тогда:

$$Y_i = A/B = 0, y_1, y_2, ..., y_i = y_1 \cdot 2^{-1} + y_2 \cdot 2^{-2} + ... + y_i \cdot 2^{-i}$$

При любом значении i должно выполняться неравенство $0 \leq A - BY_i < B2^{-i}$, т.е. остаток от деления $(A-BY_i)$ должен быть меньше делителя, умноженного на 2^{-i}.

Преобразуем это выражение к виду:

$0 \leq (A-BY_i)2^i < B \qquad 0 \leq R_i < B.$

Цифры частного определяются последовательно, начиная со старшего разряда. Допустим, что в результате выполнения i циклов получены старшие i разрядов частного, т.е. приближенное значение частного Y_i. В следующем $(i+1)$-ом цикле будет получено значение $(i+1)$-го разряда частного. Исходными данными для этого цикла являются Y_i и R_i.

Цифра Y_{i+1} может иметь одно из двух значений: 1 или 0. Если $y_{i+1}=0$, то

$Y_{i+1} = Y_i + y_{i+1} \; 2^{-(i+1)} = Y_i$

$R_{i+1} = (A - BY_{i+1})2^{i+1} = 2R_i$

т.е. в частное записывается 0 при условии $0 \leq R_{i+1} = 2R_i < B$

Если $y_{i+1}=1$, то

$Y_{i+1} = Y_i + 2^{-(i+1)}$

$R_{i+1}=(A-BY_i+1)2^{i+1}=(A-BY_i-B\,2^{-(i+1))}\,2^{i+1}=2R_i-B$

т.е. цифра частного равна 1, если выполняется условие:

$0\le R_{i+1}=2R_i-B<B$ *или* $B\le 2R_i\le 2B$.

Так как всегда выполняется одно из приведенных условий, то для определения текущей цифры частного достаточно проверить одно из них. Обычно проверяют первое. Левая часть этого неравенства выполняется заведомо, т.к. очередной остаток перед началом следующего шага всегда является положительным. Для проверки правой части неравенства сравним разность $(2R_i-B)$ с нулем. Если эта разность окажется отрицательной, то в $(i+1)$ разряд частного запишем 0 и для подготовки исходных данных для $(i+2)$ цикла определим R_{i+1} следующим образом:

$R_{i+1}=(2R_i-B)+B=2R_i$.

Если разность $2R_i-B$ окажется положительной, то запишем в $(i+1)$ разряд частного 1, а в качестве исходного значения для $(i+2)$го цикла используем вычисленную разность: $R_{i+1}=2R_i-B$.

Исходными данным для первого цикла являются

$Y_o=0\ R_o=(A-BY_o)2^o=A<B$.

После завершения n-го цикла мы получим n-значное частное Y_n, вычисленное с недостатком $R_n=(A-BY_n)2^n$.

Правило деления с восстановлением остатка формулируется следующим образом. Делитель вычитается из делимого и определяется знак нулевого остатка. Если остаток положительный, то в псевдознаковом разряде частного проставляется 1, при появлении которой формируется признак переполнения разрядной сетки и операция прекращается. Если остаток отрицательный, то в псевдознаковом разряде частного записывается 0, а затем производится восстановление делимого путем добавления к остатку делителя. Далее выполняется сдвиг восстановленного делимого на один разряд влево и повторное вычитание делителя. Знак

получаемого таким образом остатка определяет первую значащую цифру частного: если остаток положителен, но в первом разряде частного записывается 2, если отрицательный - 0.

Далее, если остаток положителен, то он сдвигается влево на 1 разряд и из него вычитается делитель для определения следующей цифры частного. Если остаток отрицателен, то к нему прибавляется делитель для восстановления предыдущего остатка, затем восстановленный остаток сдвигается на 2 разряд влево и от него вычитается делитель для определения следующей цифры частного и т.д. до получения необходимого количества цифр частного с учетом одного разряда для округления (см. рисунок 2).

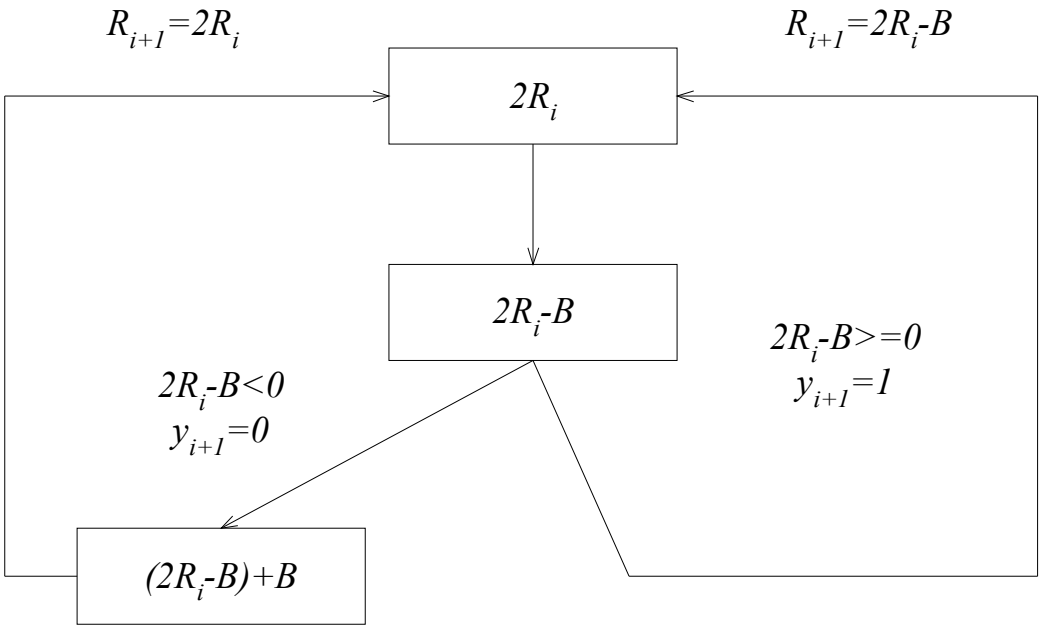

Рисунок 2 – Диаграмма для одного ($i+1$) цикла

Цифры частного получаются как инверсное значение знаковых разрядов текущего остатка, которые принимают значение *00*V*11*. Однако при сдвиге остатка влево в знаковых разрядах может возникнуть ситуация *01*, что является признаком переполнения и вырабатывается команда «аварийный останов» и осуществляется коррекция программы, если форма представления фиксированная.

Деление без восстановления остатка

Рассмотренный выше процесс деления с восстановлением остатка является аритмичным процессом с переменным числом шагов.

Операцию можно упростить и получить каждую цифру за 2 шага.

Рассмотрим случай, когда $R_i<0$. В предыдущем способе в этом случае выполнялись следующие операции:

1) восстановление остатка

$R'_i=R_i+|B|=2R_{i-1}-|B|+|B|=2R_{i-1}$

2) Сдвиг восстановленного остатка влево:

$\overleftarrow{R}'i=2R'i=2R_{i-1}\ 2=4R_{i-1}$

3) вычитание модуля делителя из восстановленного и сдвинутого влево остатка для определения следующего остатка

$R_{i+1}=4R_{i-1}-|B|$.

Если не восстанавливать остаток, а сразу сдвинуть отрицательный R_i на один разряд влево, то получим

$R_{i+1}=2R_i=2(2R_{i-1}-|B|)=4R_{i-1}-2|B|$.

Результат в данном случае отличается от действительного на величину $+|B|$. Поэтому в качестве второго шага необходимо произвести коррекцию результата на эту величину:

$R_{i+1}=4R_{i-1}-2|B|+|B|=4R_{i-1}-|B|$.

В результате получаем требуемую величину последующего остатка за 2 шага.

Диаграмма выполнения алгоритма деления без восстановления остатка представлена на рисунке 3.

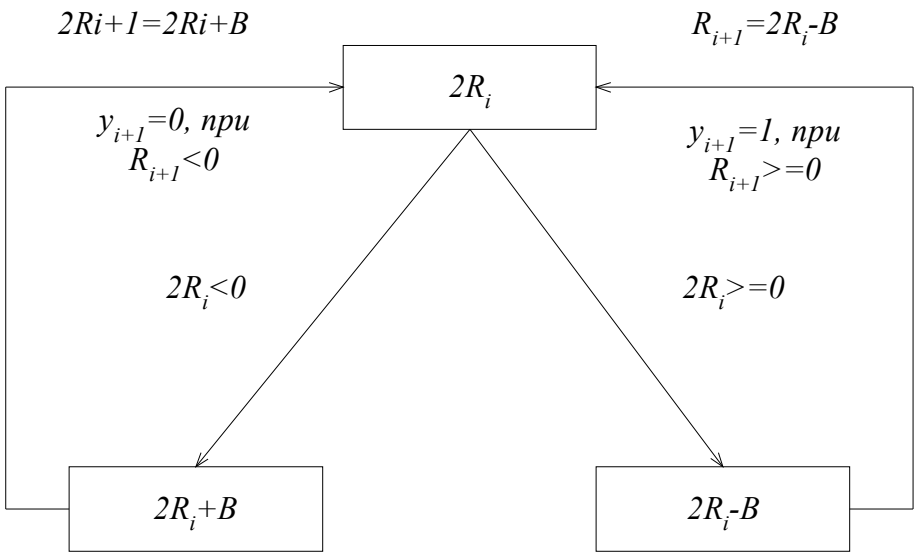

Рисунок 3 – Диаграмма выполнения алгоритма деления без восстановления остатка

Таким образом, чтобы определить следующую цифру частного, необходимо сдвинуть текущий остаток влево на 1 разряд, а затем алгебраически прибавить к нему модуль делителя, которому приписывается знак, противоположный знаку текущего остатка. Знак полученного таким образом остатка и определяет следующую цифру частного: если он положителен, то в частном записывается 1, если отрицателен - 0.

Операция сдвигов и алгебраических сложений повторяется до тех пор, пока в частном не получится требуемое количество цифр.

2 Порядок выполнения лабораторной работы

Выполнение 1 этапа. ***Сложение-вычитание двоичных чисел, представленных в дополнительных кодах***

Выбирается число разрядов исходных операндов и число разрядов сумматора (таблица 1), в зависимости от заданного варианта.

Описывается содержательный алгоритм (совокупная схема) выполнения арифметических операций сложения и вычитания в дополнительных кодах.

Производится кодирование микроопераций и условий.

Проектируется операционный автомат арифметико-логического устройства.

Описываются микрокоманды и условия.

Программная модель строится исходя из следующих ограничений.

– Входные и выходные данные представляют собой двоичные векторы с заданной разрядностью.

– Каждое устройство, включенное в схему операционного автомата, должно быть представлено в виде подпрограммы (функции, модуля).

– Головная часть программы должна включать в себя последовательность микрокоманд и условных операторов и предусматривать возможности просмотра промежуточных результатов.

Рассмотрим более подробно последовательность выполнения лабораторной работы.

Выбирается число разрядов исходных операндов и число разрядов сумматора (таблица 1), в зависимости от заданного варианта, например разрядность операндов равна 8 разрядам и разрядность сумматора равна 4-м.

Описывается содержательный алгоритм (совокупная схема) выполнения арифметических операций сложения и вычитания (см. рисунок 4). В качестве примера представлена содержательная граф-схема для разрядности операндов равной 8 разрядам и разрядности сумматора равной 4-м., учитывая следующие ограничения.

1. Операнд, записанный в регистре *A*, по модулю больше операнда регистра *B*

$$A>B.$$

2. При алгебраическом сложении и вычитании в результате не должно быть переполнения разрядной сетки.
3. Вычитание числа *B* из числа *A* выполняется по следующему правилу

$$A-B=A+(-B).$$

Производится кодирование микроопераций и условий (рисунок 4).

Проектируется операционный автомат арифметико-логического устройства, представленный на рисунке 5. В регистры *RgA* и *RgB* производится запись чисел, заданных в прямых кодах, поступающих с шины *Z*. В *RgC* записывается результат операции сложения и вычитания чисел в двоичной системе счисления с использованием дополнительного кода. На шину *Z* поступает результат арифметической операции, заданный в прямом коде. В триггер переноса *ТгР* производится запись переноса из младшей части числа в старшую. Счетчик *Ст* вырабатывает управляющую информацию для мультиплексоров MUX и демультиплексора DMUX.

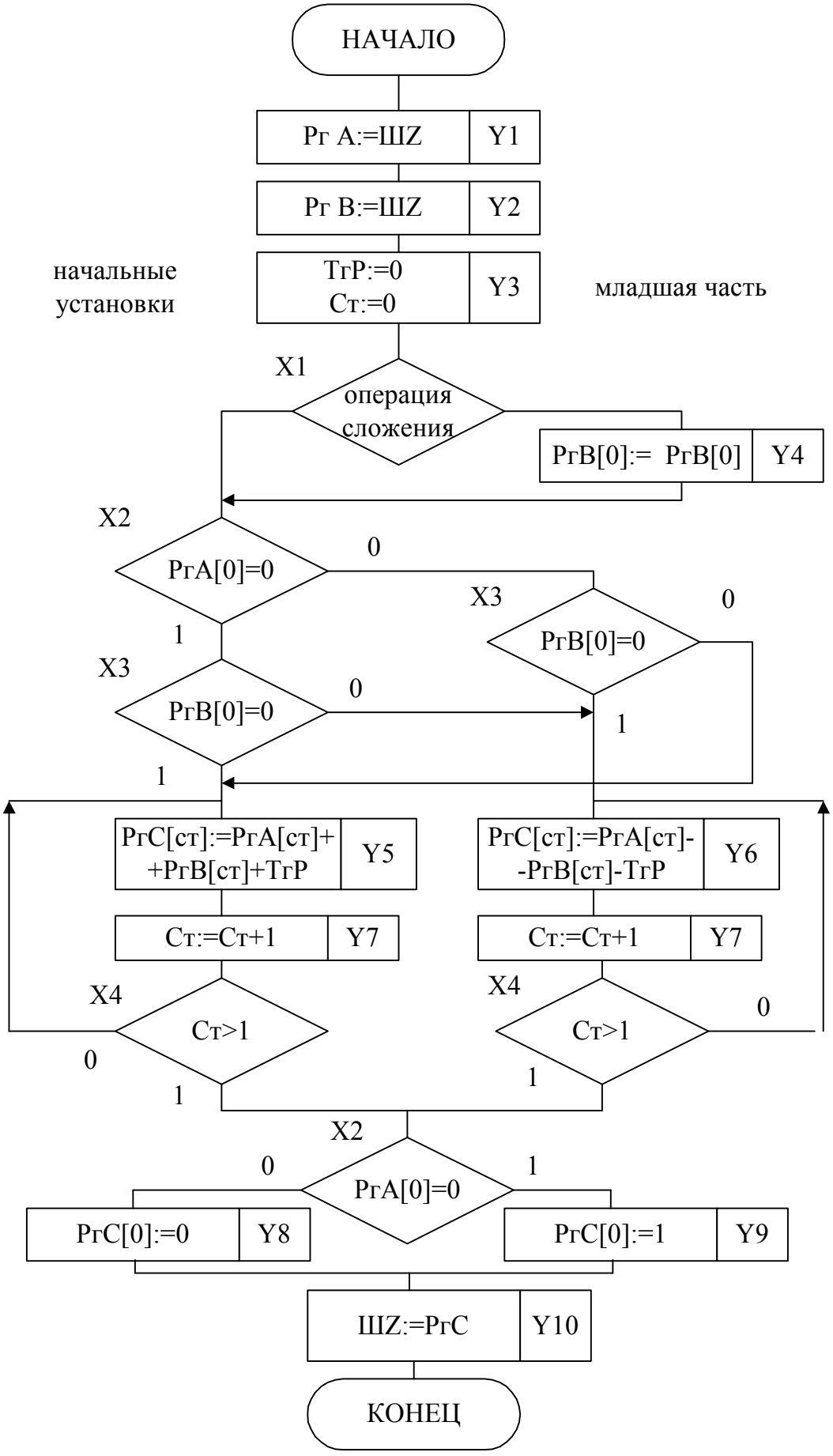

Рисунок 4 – Содержательный алгоритм (совокупная схема)

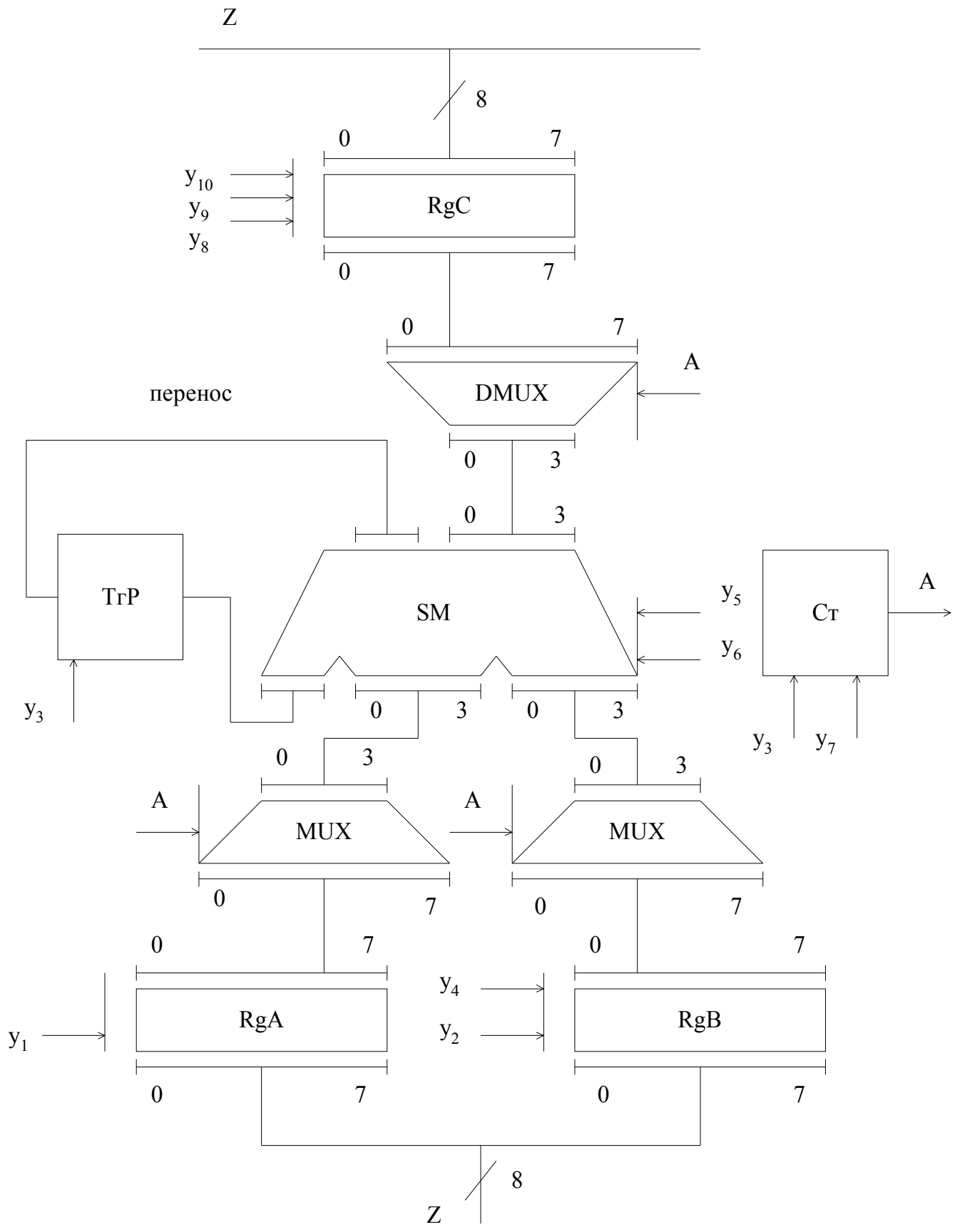

Рисунок 5 – Структурный операционный автомат для выполнения операции сложения-вычитания

Производится описание микрокоманд и условий.

y_1: *RgA:=ШZ;*

y_2: *RgB:= ШZ*

y_3: *TгP:=0; Cm:=0;*

y_4: *RgB[0]:= ⎤RgB[0];*

y_5: *RgC[cm];= RgA[cm] +RgB[cm]+TгP;*

y_6: *RgC[cm];= RgA[cm] -RgB[cm]-TгP;*

y_7:*Cm:=Cm+1;*

y_8: *RgC[0]:=0;*

y_9: *RgC[0]:=1;*

y_{10}: *ШZ:=RgC;*

x_1: *определение операции;*

x_2: *RgA[0];*

x_3: *RgB[0];*

x_{14}:*Cm>1.*

Затем производится программирование математической модели арифметических операций сложения-вычитания чисел, заданных в прямых кодах с использованием дополнительных кодов на языке высокого уровня.

Этап 2. Аналогичным образом производится выполнение моделирования операции сложение-вычитание двоичных чисел на сумматорах обратного кода.

Этап 3. Используя теоретически основы, представленные в методических указаниях аналогично производится выполнение моделирования операции умножения двоичных чисел.

Этап 4. Используя теоретически основы, представленные в методических указаниях аналогично производится выполнение моделирования операции деления двоичных чисел.

3 Содержание отчета

1. Задание на лабораторную работу.
2. Содержательный алгоритм выполнения арифметических операций.
3. Структура операционного автомата выполнения арифметических операций сложения и вычитания чисел, заданных в соответствующих кодах.
4. Составленный список микрокоманд и условий.
5. Разработанная программа на алгоритмическом языке высокого уровня, моделирующая полученную микропрограмму.
6. Составленные тестовые примеры для различных соотношений знаков операндов.
7. Выводы по работе.

Математическая модель строится исходя из следующих ограничений.

1. Входные и выходные данные представляют собой двоичные векторы с заданной разрядностью.
2. Каждое устройство, включенное в схему операционного автомата, должно быть представлено в виде подпрограммы (функции, модуля).
3. Головная часть программы должна включать в себя последовательность микрокоманд и условных операторов и предусматривать возможности просмотра промежуточных результатов.

4 Контрольные вопросы

1. Написать изображения чисел –31,-5,-18, 45 в прямом и дополнительном кодах в двоичной системе счисления.
2. Вычесть из числа –6 число 9, предварительно представив их в двоичной системе счисления.
3. Сложить на сумматоре дополнительного кода числа –28 и 45, предварительно представив их в двоичной системе счисления.
4. Написать изображения чисел –13,-47,-28, 28 в инверсном коде двоичной системы счисления.
5. Вычесть из числа –6 число 9, предварительно представив их в двоичной системе счисления.
6. Сложить на сумматоре инверсного кода числа –28 и 45, предварительно представив их в двоичной системе счисления.
7. Определить признак переполнения разрядной сетки на сумматоре обратного кода при сложении отрицательных чисел и положительных чисел.
8. Перемножить два числа в двоичной системе счисления умножением младшими разрядами множителя со сдвигом суммы частичных произведений вправо (рассмотреть различные комбинации знаковых разрядов). Задание выдает преподаватель.
9. Перемножить два числа в двоичной системе счисления умножением младшими разрядами множителя со сдвигом множимого влево (рассмотреть различные комбинации знаковых разрядов). Задание выдает преподаватель.
10. Перемножить два числа в двоичной системе счисления умножением старшими разрядами множителя со сдвигом СЧП влево (рассмотреть различные комбинации знаковых разрядов). Задание выдает преподаватель.

11. Перемножить два числа в двоичной системе счисления умножением старшими разрядами множителя со сдвигом множимого вправо (рассмотреть различные комбинации знаковых разрядов). Задание выдает преподаватель.
12. Произвести операцию деления двух чисел в двоичной системе счисления методом деления с восстановлением остатков (рассмотреть различные комбинации знаковых разрядов). Задание выдает преподаватель.
13. Произвести операцию деления двух чисел в двоичной системе счисления методом деления без восстановления остатков (рассмотреть различные комбинации знаковых разрядов). Задание выдает преподаватель.
14. Возможно ли переполнение разрядной сетки при делении чисел, представленных в форме с плавающей запятой.
15. Перечислите существующие методы ускоренного деления чисел. Приведите соответствующие примеры.

5 Варианты задания на лабораторные работы

Таблица 1 – Разрядность операндов и сумматора

Номер варианта	Разрядность операндов	Разрядность сумматора
1.	8	2
2.	16	2
3.	32	2
4.	8	4
5.	16	4
6.	32	4
7.	8	8
8.	16	8
9.	32	8
10.	8	2
11.	16	2
12.	32	2
13.	8	4
14.	16	4
15	32	4
16	8	8
17	16	8
18	32	8
19	8	2
20	16	2
21	32	2
22	8	4
23	16	4
24	32	4
25	8	8
26	16	8
27	32	8
28	8	2
29	16	2
30	32	2
31	8	4
32	16	4
33	32	4

Список литературы

1. Информатика. Базовый курс [Текст]: учебное пособие / под ред. С. В. Симоновича. - 3-е изд. - СПб.: Питер, 2012. - 640 с.
2. Громов, Ю.Ю. Архитектура ЭВМ и систем [Электронный ресурс]: учебное пособие / Ю.Ю. Громов, О. Г. Иванова, М. Ю. Серегин. - Тамбов: Издательство ФГБОУ ВПО «ТГТУ», 2012. - 200 с.
3. Процедурно-модульное программирование на Delphi [Текст]: учебное пособие / С. Г. Емельянов [и др.]. - Москва: Аргамак-Медиа, 2014. - 352 с. - ISBN 978-5-00024-0 20-5