Арифметические основы ЭВМ

Перед вами картина времен "безграмотной-необразованной" РИ Николая II. Сельская школа, детям лет 11-12. А теперь образованные взрослые дяди и тети, решите в уме (как сельские дети), написанное на доске.

1895 год.

$$\frac{10^2 + 11^2 + 12^2 + 13^2 + 14^2}{365}$$

Основные понятия.

Определение 1. *Системой счисления* называется совокупность цифровых знаков и правил их записи, применяемая для однозначного изображения чисел.

Определение 2. Под *системой счисления* понимается способ представления любого числа посредством некоторого алфавита символов, называемых «цифрами».

Определение 3. *Системой счисления* называют систему приемов и правил, позволяющих установить взаимно однозначное соответствие между любым числом и его представлением в виде конечного числа символов, называемых «цифрами».

Понятие системы счисления включает в себя:

- Алфавит, используемый для записи чисел (цифры, знаки);
- Способ записи чисел;
- Однозначность представления любого числа.

Системы счисления принято разделять на два класса:

- позиционные;
- непозиционные.

Для ***позиционных*** систем счисления значение каждой цифры однозначно определяется ее положением (позицией) в числе.

Для ***непозиционных*** систем счисления значение цифры не зависит от ее положения в числе.

Пример. Римская системы счисления:

$$LXVI = (66)_{10};$$

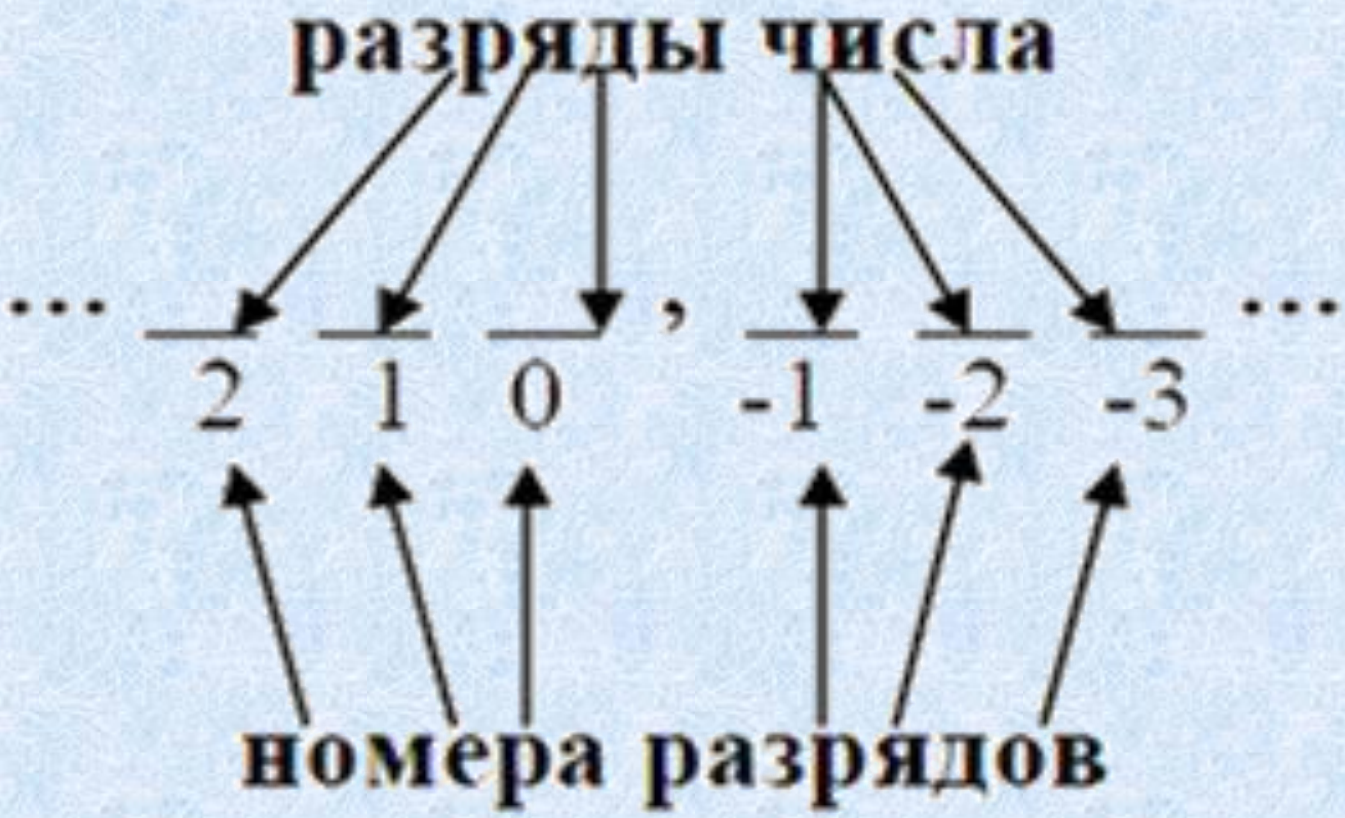
$$XLIV = (44)_{10}.$$

Два числа состоят из одинаковых цифр (L = 50, X = 10, V = 5, I = 1), однако имеют разные значения.

Римскую систему счисления нельзя считать классической непозиционной:

$$VI = 5 + 1 = 6;\ IV = 5 - 1 = 4.$$

Для позиционных систем счисления каждая позиция в числе, на которой может находиться цифра, называется ***разрядом числа***. Нумерацию разрядов принято производить влево и вправо от запятой следующим образом:



Под ***весом разряда*** принято понимать количественное значение цифры данного разряда в числе. Фактически, вес разряда представляет собой множитель, на который умножается цифра этого разряда при получении значения числа.

Для системы счисления с основанием **S** вес i-го разряда определяется в виде:

$$V_i = S^i \ .$$

Основание системы счисления показывает, во сколько раз вес i-ого разряда числа больше веса предыдущего (i-1)- го разряда.

Под основанием системы счисления можно понимать:

- Количество разнообразных цифр, используемых при записи чисел.
- Основание степени для определения веса разряда.

Запись значения основания в любой системе счисления имеет вид: **S = 10.**

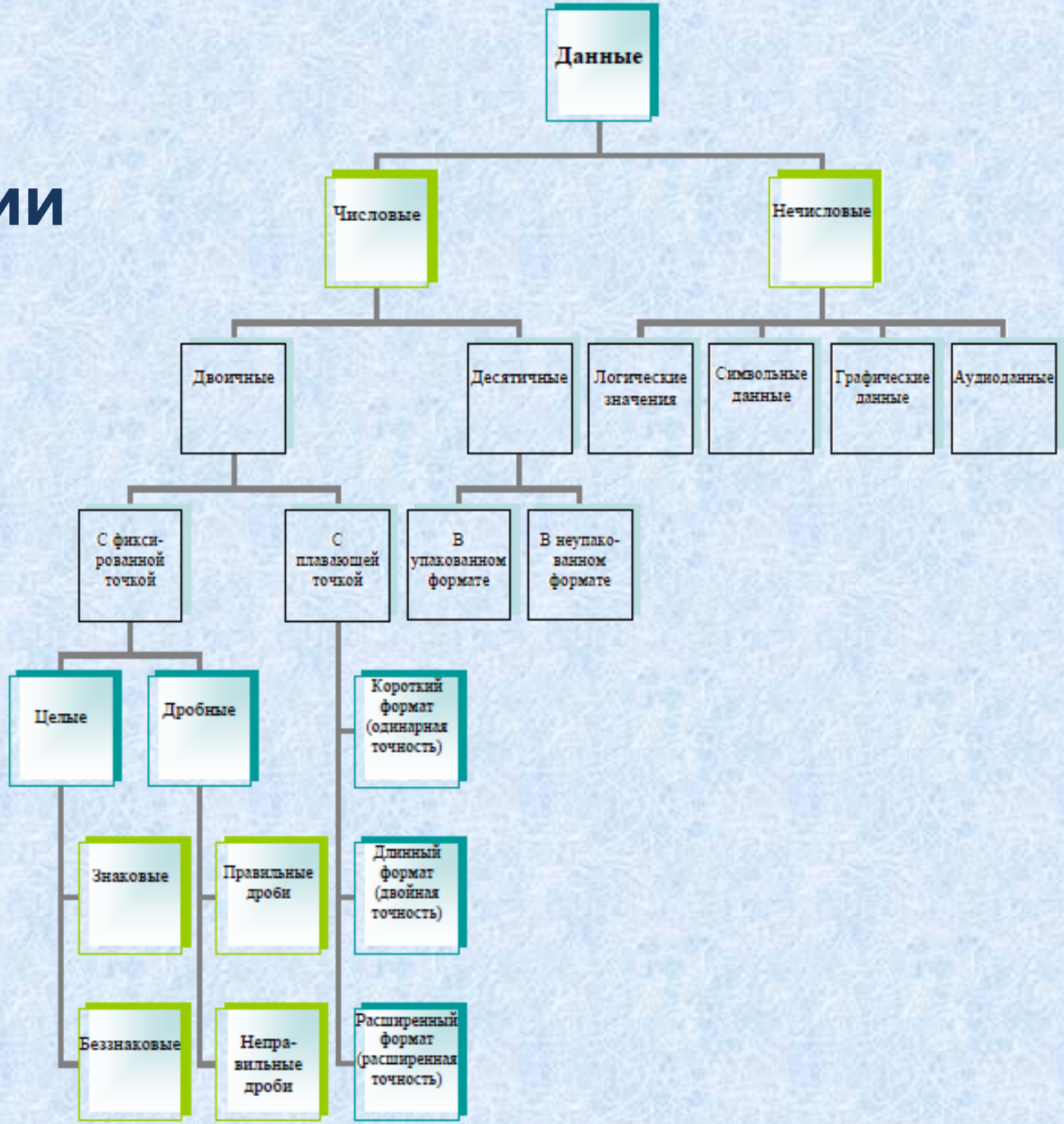
В учебнике «Прикладная теория цифровых автоматов» Савельева А.Я. доказывается, что оптимальной (с точки зрения затрат оборудования на представление и хранение чисел) является система счисления с основанием **e** ≈2,72.

Представление чисел в ЭВМ

Классификация данных, используемых в ЭВМ

В отношении данных достаточно широко используется термин «***аппаратная поддержка***». Принято считать, что данные некоторого типа в определенных форматах являются аппаратно-поддерживаемыми в рамках определенной модели компьютера (процессора), если в системе команд процессора имеются команды для обработки данных этого типа в соответствующих форматах. Например, в базовой ЭВМ аппаратно поддерживаются целые знаковые числа в 16-битном формате.

Схема в виде дерева классификации



К основным типам нечисловых данных, обладающих аппаратной поддержкой, принято относить логические значения и символьные данные.

Для логических значений характерным свойством является их побитовая обработка, то есть каждый бит логического значения обрабатывается независимо от других битов формата. Как правило, отдельные биты логических значений объединяются в стандартные форматы, которые в терминологии фирмы Intel имеют следующие наименования:

Byte (**B**) байт – 8 бит;

Word (**W**) слово – 16 бит;

Double Word (**DW**) двойное слово– 32 бита;

Quadro Word (**QW**) учетверенное слово– 64 бита.

Аппаратная поддержка логических значений реализуется на уровне логических команд, таких как:

AND – поразрядная конъюнкция (логическое И);

OR – поразрядная дизъюнкция (логическое ИЛИ);

XOR – исключительное «или» (для двух операндов операция XOR совпадает со сложением по модулю 2);

NOT – инверсия (логическое НЕ).

Символьные данные используются для представления в ЭВМ разнообразной текстовой информации.

В современных компьютерах для представления символьных данных используется код **ASCII** – American Standard Code for Information Interchange (Американский стандартный код для обмена информацией).

Аппаратная поддержка символьных данных в процессорах Intel 80X86 реализуется на уровне специальных команд обработки строк, основными из которых являются:

MOVS – пересылка строк;
CMPS – сравнение строк;
SCAS – сканирование строки.

Каждая команда обработки строк рассчитана на обработку одного элемента строки длиной в байт, слово или двойное слово, однако использование перед этими командами специального префикса **REP** (повторение) позволяет осуществлять обработку строки произвольной длины с заданным числом элементов. Использование команд обработки строк с префиксом **REP** можно рассматривать как аппаратную поддержку элементарной структуры данных типа «строка».

Числовые данные

Аппаратная поддержка числовых данных реализуется прежде всего на уровне арифметических команд, таких как:

ADD – сложение;

SUB – вычитание;

MUL – умножение;

DIV – деление.

Десятичные числа

Десятичные числа используются в ЭВМ на этапе ввода исходных данных или этапе вывода результатов для поддержки удобного интерфейса с пользователем.

В ЭВМ десятичные числа представляются в двоично-кодированной форме, в связи с чем их часто называют ***двоично-десятичными числами.***

В современных ЭВМ для кодирования десятичных цифр используется код **8421**, который характеризуется естественным представлением десятичных цифр с помощью двоичной тетрады:

0 – 0000	5 – 0101
1 – 0001	6 – 0110
2 – 0010	7 – 0111
3 – 0011	8 – 1000
4 – 0100	9 – 1001

Десятичные числа принято представлять в ЭВМ в одном из двух форматов:

- упакованном (**PACK**);
- неупакованном (**UNPACK**).

В ***упакованном формате*** в каждом байте числа содержатся две десятичные цифры. Обычно упакованный формат называют **BCD-форматом** (Binary Coded Decimal).

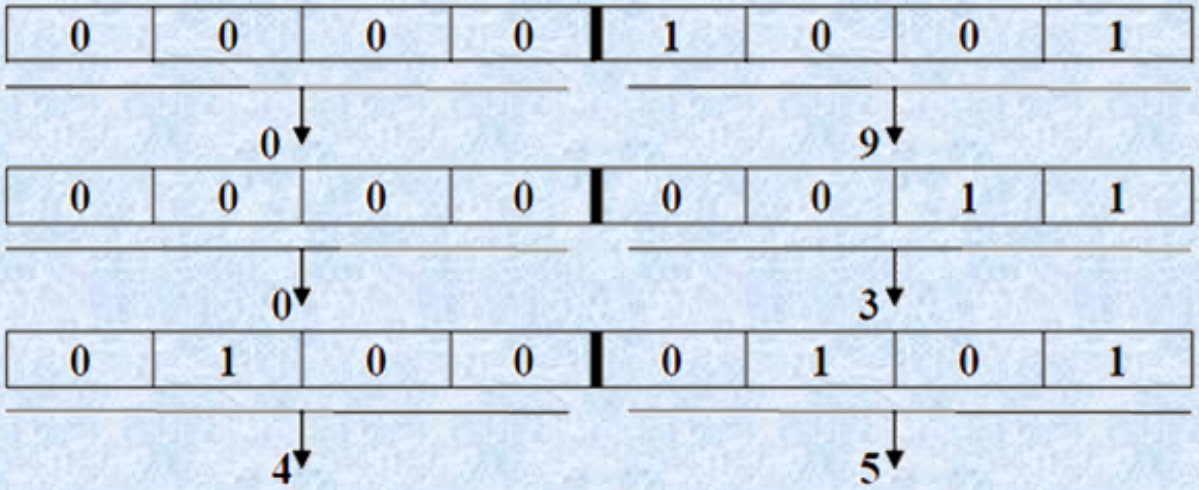
В ***неупакованном (распакованном) формате*** в каждом байте числа представляется только одна десятичная цифра. Типичным примером неупакованного формата является представление десятичных цифр в коде ASCII.

В дальнейшем представление десятичных чисел в неупакованном формате будем называть **ASCII-форматом**. В данном формате для представления десятичной цифры отводится младшая тетрада байта (младший полубайт), старшая тетрада байта (старший полубайт) принимает стандартное значение $(0011)_2 = (3)_{10.}$

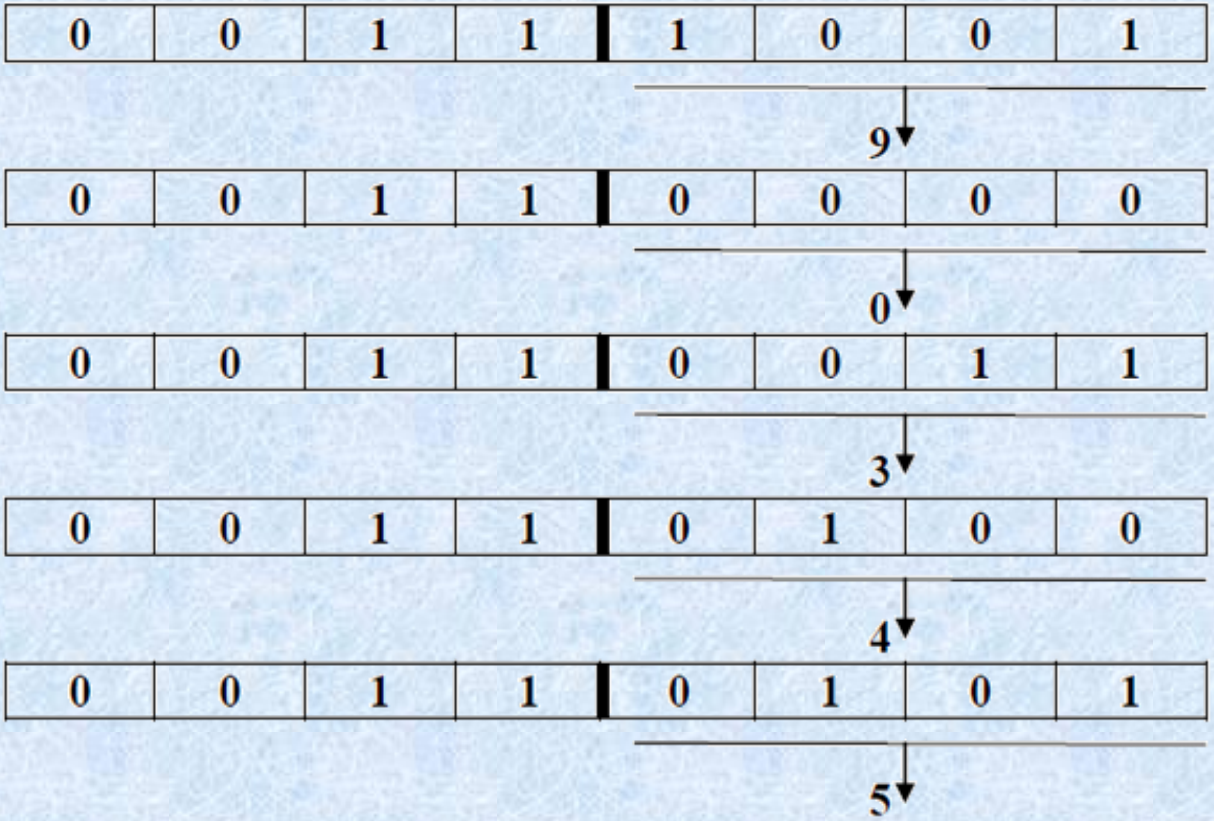
Пример.

Представить число **90345** в BCD и ASCII- форматах.

BCD:


0 0 0 0 1 0 0 1
0
9
0 0 0 0 0 0 1 1
0
3
0 1 0 0 0 1 0 1
4
5

ASCII:


0 0 1 1 1 0 0 1
9
0 0 1 1 0 0 0 0
0
0 0 1 1 0 0 1 1
3
0 0 1 1 0 1 0 0
4
0 0 1 1 0 1 0 1
5

На этапе ввода числовых данных и вывода числовых результатов десятичные числа представляются в ASCII-формате. Их преобразование в BCD-формат может быть реализовано либо на аппаратном, либо на программном уровне.

В системе команд процессоров семейства Intel X86 отсутствуют команды преобразования десятичных чисел из одного формата в другой, значит, это преобразование можно реализовать на программном уровне.

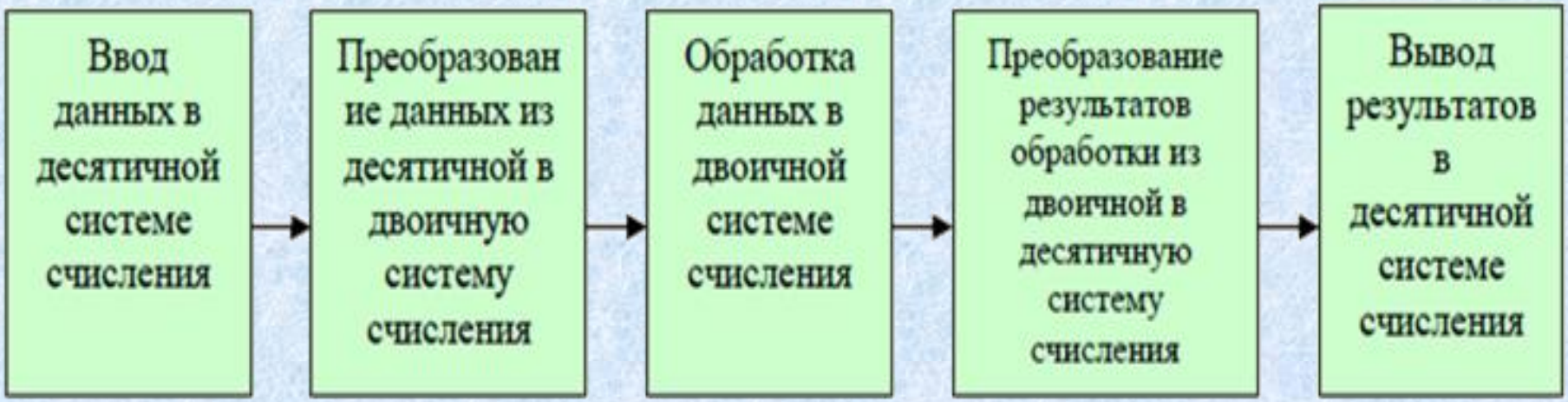
В любой ЭВМ поддерживаются как двоичные, так и десятичные числа, естественно выглядит реализация не только двоичной, но и десятичной арифметики для обработки десятичных чисел.

Из-за разделения двоичных чисел на две формы представления (с фиксированной запятой и с плавающей запятой) практически в любом компьютере реализована аппаратная поддержка как целочисленной арифметики (для чисел с фиксированной запятой), так и арифметики с плавающей запятой. Так, например, в процессорах Intel наряду с целочисленным АЛУ (ALU) имеется также АЛУ для операций над числами с плавающей запятой, которое входит в состав FPU. Для того, чтобы подчеркнуть целочисленность АЛУ, его аббревиатуру достаточно часто дополняют буквой I: IALU.

Двоичная и десятичная арифметики и области их применения

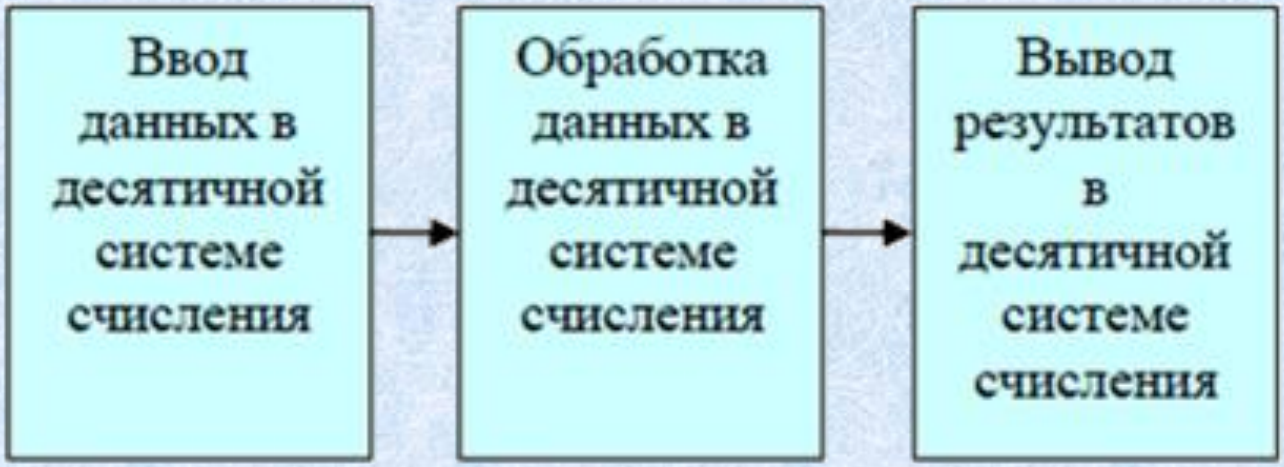
Для обработки данных в ЭВМ возможно применение одной из двух следующих схем:

1)



двоичная арифметика

2)



десятичная арифметика

Первую схему обработки данных целесообразно применять при сравнительно небольших объемах перерабатываемых данных и достаточно большом объеме вычислений (операций), приходящихся на единицу данных. Подобным свойством обладают так называемые ***научно-технические задачи.***

Вторую схему обработки целесообразно применять при больших объемах обрабатываемых данных и небольшом объеме вычислений, приходящихся на каждую единицу данных. Подобным свойством обладают так называемые ***экономические (коммерческие) задачи.***

Аппаратная поддержка второй схемы обработки данных в ЭВМ предполагает наличие в составе процессора помимо традиционного двоичного АЛУ еще и ***десятичного АЛУ.*** Использование подобных АЛУ характеризуются ЭВМ, относящиеся к классам Main Frame. В ПК и рабочих станциях на базе практически всех современных процессоров десятичное АЛУ отсутствует.

Двоичные числа с фиксированной запятой
Знаковые и беззнаковые числа

Основной особенностью представления целых знаковых чисел является использование дополнительного кода. Дополнительный код позволяет значительно упростить основные арифметические операции (сложение, вычитание) по сравнению с прямым кодом.

Дополнительный код числа:

$$[X]_{доп.} = \begin{cases} X, \text{ при } X \geq 0, \\ 2^{n-1} + |X|, \text{ при } X < 0. \end{cases}$$

Пример.

$n = 5 \quad X = -13$

$[X]_{пр} = 1.1101 = 2^4 + 13 = 29$

$[X]_{доп} = 2^5 - |X| = \_1.0000 \quad = 1.0011 = 32 - 13 = 19$

$\underline{1.1101}$

1.0011

Основными форматами, в которых представляются целые числа, являются:

1 байт ;

2 байта – слово;

4 байта – двойное слово;

8 байтов – учетверенное слово.

Диапазон представления знаковых целых чисел

$$-2^{n-1} \le A_{зн} \le 2^{n-1}-1$$

$1.\underbrace{00\ldots00}_{n-1}$ $0.\underbrace{11\ldots11}_{n-1}$

Отрицательное число по модулю на единицу больше положительного.

Для байтного формата: n = 8

$$-128 \le A_{зн} \le 127$$

Диапазон представления беззнаковых целых чисел

$$0 \le A_{бзн} \le 2^n - 1$$

$\underbrace{00\ldots0}_{n}$ $\underbrace{11\ldots1}_{n}$

Для байтного формата: n = 8

$$0 \le A_{зн} \le 255$$

Диапазон представления дробных чисел

Для правильной n-разрядной дроби:

$$\mathbf{2^{-n} \le |A_{др}| \le 1 - 2^{-n}}$$

0.00…01 0.11…1

Неправильная дробь в целой части содержит обязательную и единственную единицу.

$$\mathbf{1 \le |A_{др.непр.}| \le 2 - 2^{-(n+1)}}$$

1.00…0 1.11…1

Числа с плавающей запятой

В формате представления чисел с плавающей запятой имеются три части:

- **знак** (представляется крайним левым битом формата)
- **мантисса** (представляется в виде правильной или неправильной двоичной дроби);
- **порядок** (представляется в общем случае как целое число со знаком).

С учетом этих частей значение числа с плавающей запятой представляется в виде:

$$A_{пз} = (-1)^{signA} \bullet M_A \bullet S^{PA}$$

signA – знак; P_A – порядок числа;

M_A –мантисса числа А; **S** – основание порядка.

Основные особенности представления чисел с плавающей запятой в современных ЭВМ

Классы ЭВМ:

- ***Универсальные ЭВМ – Mainframe*** (MF), такие как: IBM System 370;

  Модели ЕС ЭВМ: ЕС 1010; ЕС 1065; ЕС 1087.

- ***Мини ЭВМ***, такие как:

  DEC: PDP-11; VAX-11;

  Модели СМ ЭВМ: СМ 1- СМ 4; СМ 1420,…

- ***ПК*** (например, на базе процессоров Intel 80X86, Pentium). Стандарт IEEE 754

1. Мантисса представляется в прямом коде независимо от знака числа.

2. Порядок числа представляется ***со смещением*** как целое число без знака. Величина смещения обычно равна весу старшего разряда смещенного порядка или на единицу меньше его.

3. В качестве основания порядка используются:

Классы:
1. Универсальные ЭВМ – **S = 16**;
2. Мини ЭВМ – **S = 2**;
3. ПК – **S = 2**.

4. В целях повышения точности в основном используются ***нормализованные числа*** с плавающей запятой.

Определение. Число с плавающей запятой называется ***нормализованным***, если старшая цифра его мантиссы является значащей (не 0).

5. Мантисса чисел в классах универсальных ЭВМ и мини ЭВМ является правильной дробью, а в стандарте IEEE – неправильной.

6. Использование в качестве основания порядка S=2 и использования в основном нормализованных чисел в мини ЭВМ и ПК требует, чтобы старшая цифра мантиссы была равна 1. Старшая единица мантиссы с целью увеличения точности в формате не представляется, а лишь подразумевается, и называется ***скрытым разрядом*** (скрытой единицей).

7. В целях разумного компромисса между точностью представления и скоростью обработки данных в ЭВМ используется несколько форматов:

- ***короткий формат*** (формат одинарной точности) - 32 разряда;
- ***длинный формат*** (формат двойной точности) – 64 разряда;
- ***расширенный формат*** (формат расширенной точности) :

для MF и Мини ЭВМ – 128 разрядов;

для IEEE – 80 разрядов.

В классах MF и Мини ЭВМ увеличивается разрядность мантиссы, в стандарте IEEE – и мантиссы и порядка.

Диапазон представления чисел с плавающей запятой

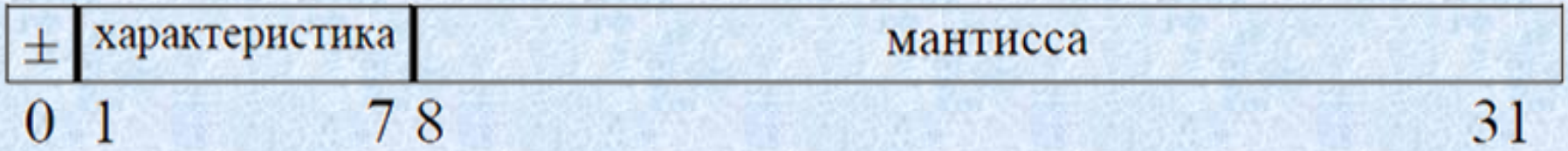
Определяется в отношении модуля нормализованного числа.

$$M_{Amin}\, S^{PAmin} \le | A_{п.т.\ норм} | \le M_{Amax}\, S^{PAmax}$$

Для классов:

MF

S = 16, d = 64 – смещение порядка.



Характеристика: $0 \le X_A \le 127$;

Порядок: $-64 \le P_A \le 63$.

$$1/16 \bullet 16^{-64} \le | A_{п.т.\ норм} | < 1 \bullet 16^{63}$$

Мини ЭВМ

S = 2; d = 128 – смещение порядка.

Характеристика: $0 \leq X_A \leq 225$;

Порядок: $-128 \leq P_A \leq 127$.

$$1/2 \bullet 2^{-128} \leq | A_{\text{п.т. норм}} | < 1 \bullet 2^{127}$$

Стандарт IEEE

S = 2;

d = 127 – для короткого формата;

d = 1023 – для длинного формата;

d = 16383 – для расширенного формата.

Скрытая единица имеется только в коротком или длинном формате, в расширенном формате она представляется в явном виде.

При определении диапазона представления чисел необходимо учитывать особенности стандарта IEEE-754:

Крайние значение характеристики (порядка) во всех форматах зарезервированы и для представления нормализованных чисел не используются.

Максимальное значение характеристики при знаке + и нулевой мантиссе зарезервировано для $+\infty$ и для представления не чисел (NAN).

Максимальное значение характеристики с единицей в старшем разряде мантиссы используется для представления - ∞.

Минимальное значение характеристики используется для представления ненормализованных чисел со знаками +, – и для представления нуля (+0 и -0).

Поэтому диапазон характеристики

для короткого формата: $1 \leq X_A \leq 254$;

диапазон чисел: $\mathbf{2^{-126} \leq |\, A_{п.т.\ норм}\, | < 2^{128}.}$

для длинного формата: $\mathbf{10^{-308} \leq |\, A_{п.т.\ норм}\, | < 10^{308}.}$

для расширенного формата:

$$\mathbf{10^{-4932} \leq |\, A_{п.т.\ норм}\, | < 10^{4932}}$$

Точность представления чисел

Каждая десятичная дробь представляется в виде бесконечной двоичной дроби, что в условиях ограниченного формата приводит к возникновению погрешности.

Максимальная абсолютная погрешность имеет место в том случае, когда все отбрасываемые разряды равны единице.

0 . 10 ……110 111 …….1

n n+1 ∞

$$\Delta Адр.\max = \sum_{i=n+1}^{\infty} 2^{-i} = 2^{-n},$$

где n- разрядность числа.

$$\delta Адр.\min = \left|\frac{\Delta Адр.\max}{Адр.\max}\right| = \left|\frac{2^{-n}}{1-2^{-n}}\right| \approx 2^{-n},$$

$$\delta Адр.\max = \left|\frac{\Delta Адр.\max}{Адр.\min}\right| \approx \left|\frac{2^{-n}}{2^{-n}}\right| = 1.$$

Погрешность представления чисел с плавающей точкой определяется погрешностью мантиссы как дробного числа. $\left|A_{n.m.}\right| = \left|M_A\right| \cdot S^{P_A}$

$$\delta An.m. = \left|\frac{\Delta A\, n.m.}{An.m.}\right| = \left|\frac{\Delta M_A \cdot S^{P_A}}{M_A \cdot S^{P_A}}\right| = \left|\frac{\Delta M_A}{M_A}\right| = \left|\frac{2^{-n}}{1/S}\right| = S \cdot 2^{-n}$$

Формула справедлива для правильных и неправильных дробей.

Точность представления чисел для различных типов машин

ЕС: $\delta A_{п.т.} = 16 \bullet 2^{-24} = 2^{-20} = 10^{-6}$;

СМ: $\delta A_{п.т.} = 2 \bullet 2^{-24} = 2^{-23} = 10^{-7}$;

IEEE: $\delta A_{п.т.} = 10^{-7}$.

Методы округления чисел с плавающей точкой

Округление производится для повышения точности представления чисел.

Методы округления оговариваются стандартом IEEE 754.

- ***Округление усечением*** – не вписывающиеся в сетку разряды отбрасываются;
- ***Округление к ближайшему*** – анализируется старший отбрасываемый разряд, если он равен 1, то к младшему разряду мантиссы добавляется 0;
- ***Округление к ближайшему большему*** (округление к + ∞)- для положительных чисел 1 добавляется в младший разряд мантиссы, для отрицательных чисел мантисса не меняется.
- ***Округление к ближайшему меньшему*** (округление к -∞) – для отрицательных чисел единица добавляется в младший разряд мантиссы.

Все методы, кроме метода усечения, позволяют уменьшить максимальную относительную погрешность. По умолчанию используется метод округления к ближайшему.

Регистр флагов

Арифметические флаги формируются арифметическими командами и являются признаками их результата.

Флаги управления оказывают влияние на процесс выполнения программ.

К арифметическим флагам относятся:

- **(CF) Carry Flag –** флаг переноса, в нем фиксируется перенос из старшего разряда при сложении и заем в старший разряд при вычитании. При умножении CF показывает возможность (=0) и невозможность (=1) представления произведения в том же формате, что и операндов.

- **(PF) Parity Flag** – флаг паритета (четности). Устанавливается в единицу при наличии четного числа единиц в младшем байте результата, в противном случае - сбрасывается.. PF используется в качестве аппаратной поддержки контроля по четности.

- **(AF) Auxiliary Carry Flag** - флаг вспомогательного переноса, в котором фиксируется межтетрадный перенос при сложении и межтетрадный заем при вычитании. Этот флаг используется командами десятичной арифметики.
- **(ZF) Zero Flag –** флаг нуля, устанавливается при нулевом значении результата, в противном случае сбрасывается.
- **(SF) Sign Flag –** флаг знака, в котором копируется старший разряд результата.

- **(OF) Overflow Flag** флаг переполнения.

Устанавливается в командах сложения и вычитания, если результат не помещается в формате, при этом и операнды и результат интерпретируются как знаковые числа. Аппаратно он формируется совпадением переносов из двух старших разрядов при сложении и заемов в два старших разряда при вычитании (если они совпадают, то флаг равен нулю).

Выполнение арифметических операция в ЭВМ

Сложение целых чисел

Операции двоичного сложения реализуются поразрядно, начиная с младшего разряда, с учетом возникающих межразрядных переносов.

a_i	b_i	P_{i-1}	S_i	P_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

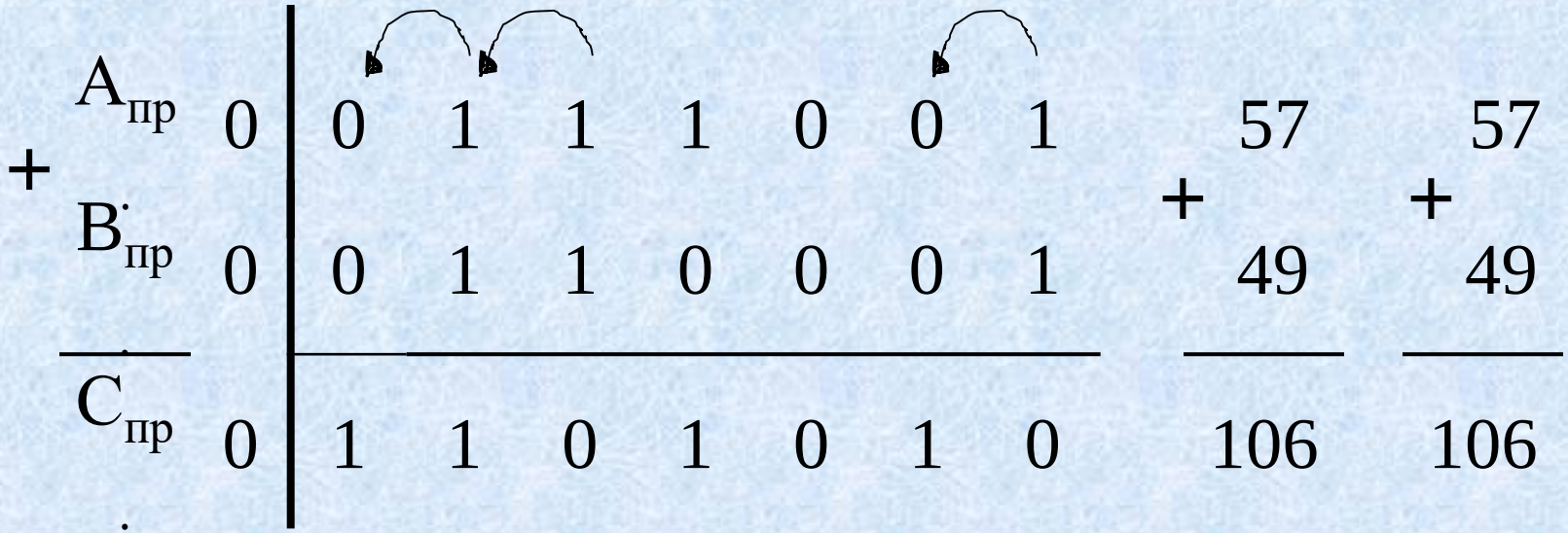
P_{i-1} – перенос из предыдущего разряда;

P_i – перенос из i-го разряда;

S_i – сумма i-го разряда.

Пример. Выполнить операцию сложения A = 57, B = 49.

A>0, B>0.



CF=0 ZF=0

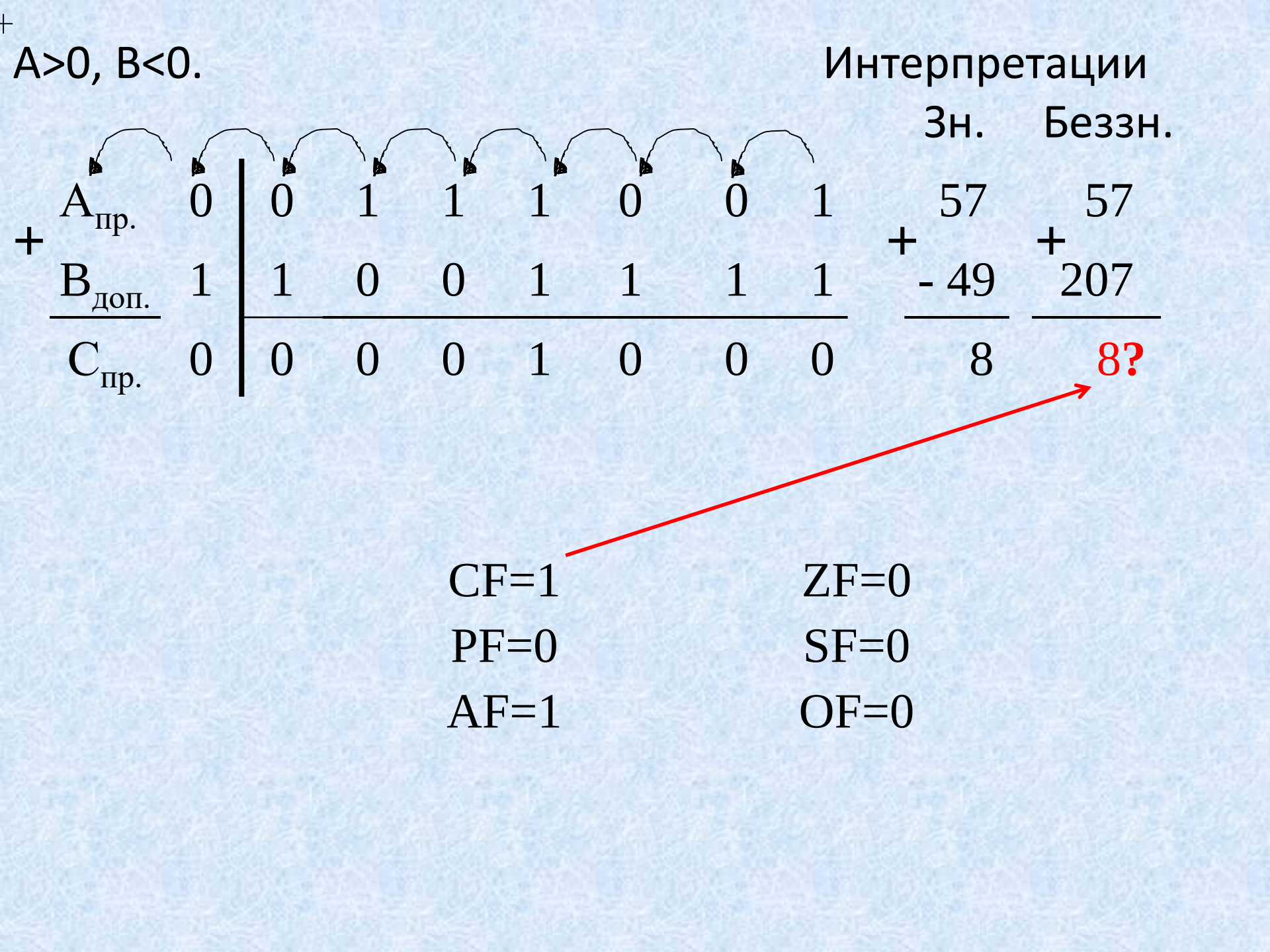
PF=1 SF=0

AF=0 OF=0

A>0, B<0.

Интерпретации

									Зн.	Беззн.
$A_{пр.}$	0	0	1	1	1	0	0	1	57	57
+ $B_{доп.}$	1	1	0	0	1	1	1	1	+ - 49	+ 207
$C_{пр.}$	0	0	0	0	1	0	0	0	8	8?



CF=1 ZF=0

PF=0 SF=0

AF=1 OF=0

A<0, B>0.

Интерпретации

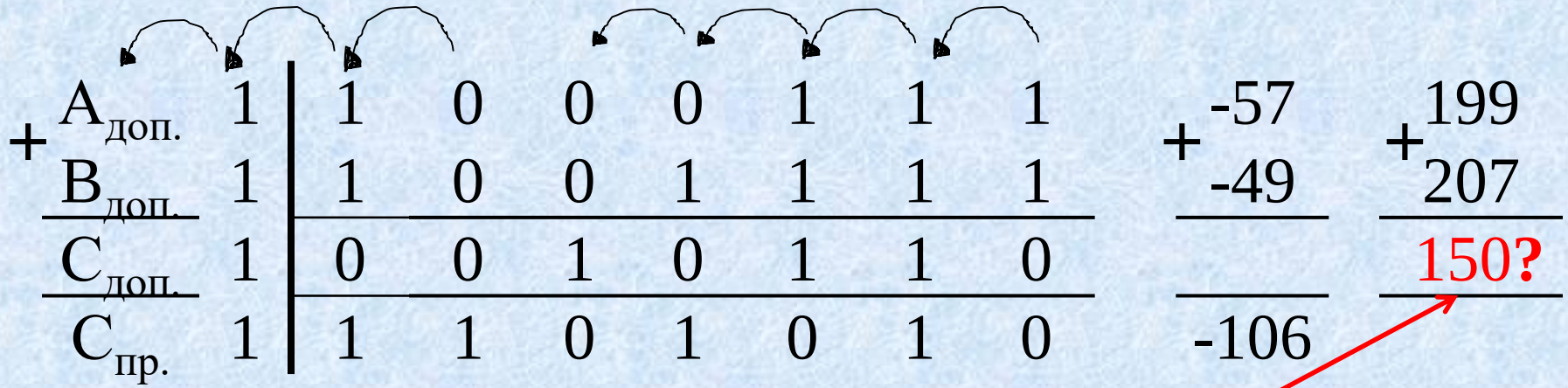
									Зн.	Беззн.
$A_{доп}$	1	1	0	0	0	1	1	1	-57	199
+ $B_{пр.}$	0	0	1	1	0	0	0	1	+ 49	+ 49
$C_{доп}$	1	1	1	1	1	0	0	0		248
$C_{пр.}$	1	0	0	0	1	0	0	0	-8	

CF=0 ZF=0

PF=0 SF=1

AF=0 OF=0

A<0, B<0.


Интерпретации
Зн. Беззн.
$A_{доп.}$ 1 1 0 0 0 1 1 1 -57 199
+ + +
$B_{доп.}$ 1 1 0 0 1 1 1 1 -49 207
$C_{доп.}$ 1 0 0 1 0 1 1 0 150?
$C_{пр.}$ 1 1 1 0 1 0 1 0 -106
CF=1 ZF=0
PF=1 SF=1
AF=1 OF=0

Пример. Выполнить операцию сложения A = 57, B = 96.

A

0	0	1	1	1	0	0	1

B

7							0
0	1	1	0	0	0	0	0

A>0, B>0.

									Интерпретации	
									Зн.	Беззн.
$A_{пр.}$	0	0	1	1	1	0	0	1	57	57
+ $B_{пр.}$	0	1	1	0	0	0	0	0	+ 96	+ 96
$C_{доп}$	1	0	0	1	1	0	0	1		153
$C_{пр.}$	1	1	1	0	0	1	1	1	-103	?

CF=0 ZF=0

PF=1 SF=1

AF=0 OF=1

A<0, B<0.

Интерпретации

									Зн.	Беззн.
$A_{доп.}$	1	1	0	0	0	1	1	1	-57	199
+									+	+
$B_{доп.}$	1	0	1	0	0	0	0	0	-96	160
$C_{пр.}$	0	1	1	0	0	1	1	1	103 ?	103?

CF=1 ZF=0

PF=0 SF=0

AF=0 OF=1

A = 79, B = 49.

	7							0
A	0	1	0	0	1	1	1	1
B	0	0	1	1	0	0	0	1

A>0, B>0.

Интерпретации

									Зн.	Беззн.
		0	1	0	0	1	1	1	1	79	79
+	$A_{пр.}$	0	0	1	1	0	0	0	1	+ 49	+ 49
	$B_{пр.}$	0	0	0	0	0	0	0	0		128
	$C_{доп.}$	1	0	0	0	0	0	0	0		
	$C_{пр.}$	1	0	0	0	0	0	0	0	-128	?

CF=0 ZF=0

PF=0 SF=1

AF=1 OF=1

A<0, B<0.

Интерпретации

									Зн.	Беззн.
$A_{доп.}$	1	1	0	0	1	1	1	1	-79	177
+ $B_{доп.}$	1	0	1	1	0	0	0	1	+ -49	+ 207
$C_{доп}$	1	0	0	0	0	0	0	0		128?
$C_{пр.}$	1	0	0	0	0	0	0	0	-128	

CF=1 ZF=0

PF=0 SF=1

AF=1 OF=0

Переполнение при сложении чисел возникает только в том случае, если операнды имеют одинаковые знаки. Переполнение фиксируется тремя способами:

- сравнение знаков операндов и суммы: если знак суммы отличается от знаков операндов, то фиксируется переполнение;
- сравнение переносов из двух старших разрядов: если они не совпадают, то фиксируется переполнение;
- использование модифицированного знака (под знак отводится два разряда, второй разряд дублирует знак).

Вычитание целых чисел (в формате IEEE)

1. Вычитание можно проводить двумя способами:
Сведение вычитания к сложению, заменяя операнд ***B*** на противоположный.
2. Выполнение прямого (непосредственного) вычитания производится поразрядно, начиная с младших разрядов, с учетом возникающих межразрядных заемов.

Пример.
Для заданных чисел A и B выполнить операцию знакового вычитания со всеми комбинациями знаков операндов.

Вычитание реализуется по следующей таблице:

a_i	b_i	Z_{i-1}	r_i	Z_i
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Z_{i-1}- заем из i-го разряда;

r_i – разность;

Z_i – заем в i-й разряд из (i-1)-го разряда.

1. A = 67, B = 51.

$[A]_{пр.}$=0.1000011

$[B]_{пр.}$=0.0110011

A>0, B>0.

Интерпретации

									Зн.	Беззн.
$A_{пр.}$	0	1	0	0	0	0	1	1	67	67
−									−	−
$B_{пр.}$	0	0	1	1	0	0	1	1	51	51
$C_{пр.}$	0	0	0	1	0	0	0	0	16	16

CF=0 ZF=0

PF=0 SF=0

AF=0 OF=0

A<0, B>0.

$[A]_{пр.}$=1.1000011; $[A]_{доп.}$=1.0111101

$[B]_{пр.}$=0.0110011

									Интерпретации	
									Зн.	Беззн.
$\mathbf{A_{доп.}}$	**1**	**0**	**1**	**1**	**1**	**1**	**0**	**1**	**- 67**	**189**
$\mathbf{B_{пр.}}$	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**1**	**— 51**	**— 51**
$\mathbf{C_{доп.}}$	**1**	**0**	**0**	**0**	**1**	**0**	**1**	**0**		**138**
$C_{пр.}$	1	1	1	1	0	1	1	0	**-118**	

CF=0 ZF=0

PF=0 SF=0

AF=0 OF=0

A<0, B>0.

[A]пр.=1.1000011;

[B]пр.=1.0110011; [B]доп.=1.1001101

Интерпретации

									Зн.	Беззн.
$A_{пр.}$	0	1	0	0	0	0	1	1	-67	67
$B_{доп.}$	1	1	0	0	1	1	0	1	-51	205
$C_{пр.}$	0	0	0	1	0	0	0	0	118	**118?**

CF=1 ZF=0

PF=0 SF=0

AF=1 OF=0

A<0, B<0.

Интерпретации

									Зн	Беззн
$A_{доп.}$	1	0	1	1	1	1	0	1	-67	189
− $B_{доп.}$	1	1	0	0	1	1	0	1	-51	205
$C_{доп.}$	1	1	1	1	0	0	0	0		**240?**
$C_{пр.}$	1	0	0	1	0	0	0	0	-16	

CF=1 ZF=0

PF=1 SF=1

AF=0 OF=0

2. A = 67, B = 64.

A<0, B>0.

									Зн	Беззн
$A_{доп.}$	1	0	1	1	1	1	0	1	-67	189
– $B_{пр.}$	0	1	0	0	0	0	0	0	– 64	– 64
$C_{пр.}$	0	1	1	1	1	1	0	1	125?	125

Интерпретации

CF=0 ZF=0

PF=1 SF=1

AF=0 OF=1

A>0, B<0.

Интерпретации

									Зн	Беззн
$A_{пр.}$	0	1	0	0	0	0	1	1	67	67
– $B_{доп.}$	1	1	0	0	0	0	0	0	-64	192
$C_{доп.}$	1	0	0	0	0	0	1	1		**131?**
$C_{пр.}$	1	1	1	1	1	1	0	1	**-125?**	

CF=1 ZF=0

PF=0 SF=0

AF=0 OF=1

3. A = 77, B = 51.

A>0, B<0.

									Зн	Беззн
$A_{пр.}$	0	1	0	0	1	1	0	1	**77**	77
$B_{доп.}$	1	1	0	0	1	1	0	1	-51	205
$C_{доп.}$	1	0	0	0	0	0	0	0		**128?**
$C_{пр.}$	1	0	0	0	0	0	0	0	**-128?**	

Интерпретации

CF=1
PF=0
AF=0

ZF=0
SF=1
OF=1

A<0, B>0. Интерпретации

									Зн	Беззн
$A_{доп.}$	1	0	1	1	0	0	1	1	-77	179
–									–	–
$B_{пр.}$	0	0	1	1	0	0	1	1	51	51
$C_{доп.}$	1	0	0	0	0	0	0	0		128
$C_{.пр.}$	1	0	0	0	0	0	0	0	-128	

CF=0 ZF=0

PF=0 SF=1

AF=0 OF=0

Для знакового вычитания переполнение возможно только при разных знаках операндов.

О переполнении можно судить по одному из двух признаков:

1. знак результата отличается от знака первого операнда;
2. несовпадение заемов в два старшие разряда (один из них присутствует, а другой нет).

Операция умножения целых чисел и принципы ее реализации в ЭВМ

Умножение двоичных чисел состоит в последовательном умножении множимого на отдельные разряды множителя с суммированием результатов умножения. Результат умножения множимого на один разряд множителя принято называть *частным произведением* (СЧП).

Пример.

A = 13, B = 11.

Первый способ:

$$
\begin{array}{r}
1\,1\,0\,1 \\
\times\ 1\,0\,1\,1 \\
\hline
1\,1\,0\,1 \\
1\,1\,0\,1 \\
1\,1\,0\,1 \\
\hline
1\,0\,0\,0\,1\,1\,1\,1
\end{array}
\qquad (143)_{10}
$$

Второй способ:

$$
\begin{array}{r}
1\,1\,0\,1 \\
\times\ 1\,0\,1\,1 \\
\hline
1\,1\,0\,1 \\
1\,1\,0\,1 \\
1\,1\,0\,1 \\
\hline
1\,0\,0\,0\,1\,1\,1\,1
\end{array}
$$

Особенности операций умножения целых чисел:

- каждое частное произведение либо совпадает с множимым, либо равно нулю;
- формируемые частные произведения должны быть определенным образом сдвинуты друг относительно друга для их последующего суммирования;
- частное произведение можно формировать, начиная как от младших, так и от старших разрядов множителя;
- для результата умножения требуется количество цифр, равное сумме количества цифр операндов. Если операнды имеют одинаковое количество цифр, то для суммы потребуется $2n$ разрядов.

Особенности реализации операций умножения в ЭВМ:

1. В операционном устройстве для умножения двоичных чисел должен использоваться многоразрядный двоичный сумматор, поэтому умножение реализуется в виде последовательного многошагового процесса, на каждом шаге которого проводится умножение на один разряд множителя. Для фиксации суммы частных произведений необходимо использовать 2n-разрядный регистр при n-разрядных операндах. Перед началом операции этот регистр необходимо обнулить;

2. На каждом шаге умножения анализируется определенный разряд множителя. Если он равен 1, то на этом шаге проводится сложение суммы частных произведений с множимым. Если разряд равен 0, то сложение не проводится;

3. Каждый шаг умножения должен сопровождаться сдвигом суммы частных произведений (СЧП) относительно неподвижного множимого, или наоборот: сдвиг множимого относительно неподвижной СЧП;

4. Умножение можно начинать как с младших, так и со старших разрядов множителя;

5. В целях упрощения схемы умножения регистр множителя реализуется как сдвигающий, это дает возможность анализировать только один разряд регистра. При выполнении умножения, начиная от младших разрядов, схема анализа привязывается к младшему разряду регистра множителя, регистр при этом сдвигается вправо;
При реализации умножения, начиная от старших разрядов, схема анализа привязывается к старше-му разряду множителя, реализуется сдвиг вправо.
6. Для фиксации завершения операции в операци-онном устройстве умножения должен быть ис-пользован счетчик (inc или dec), который считает количество разрядов множителя;

7. Так как возможно умножение, начиная от старших и с младших разрядов (то есть сдвигается множимое или СЧП), то можно использовать четыре способа (схемы) умножения.

Способы (схемы) реализации умножения в ЭВМ

- начиная от младших разрядов множителя со сдвигом множимого влево;
- начиная от младших разрядов со сдвигом СЧП вправо;
- начиная от старших разрядов со сдвигом множимого вправо;
- начиная от старших разрядов со сдвигом СЧП влево.

Анализ схем

1. В схемах умножения со сдвигом множимого для его представления требуется два *n*-разрядных регистра.

2. Для схем умножения со сдвигом СЧП для представления множимого требуется *n*-разрядный регистр.

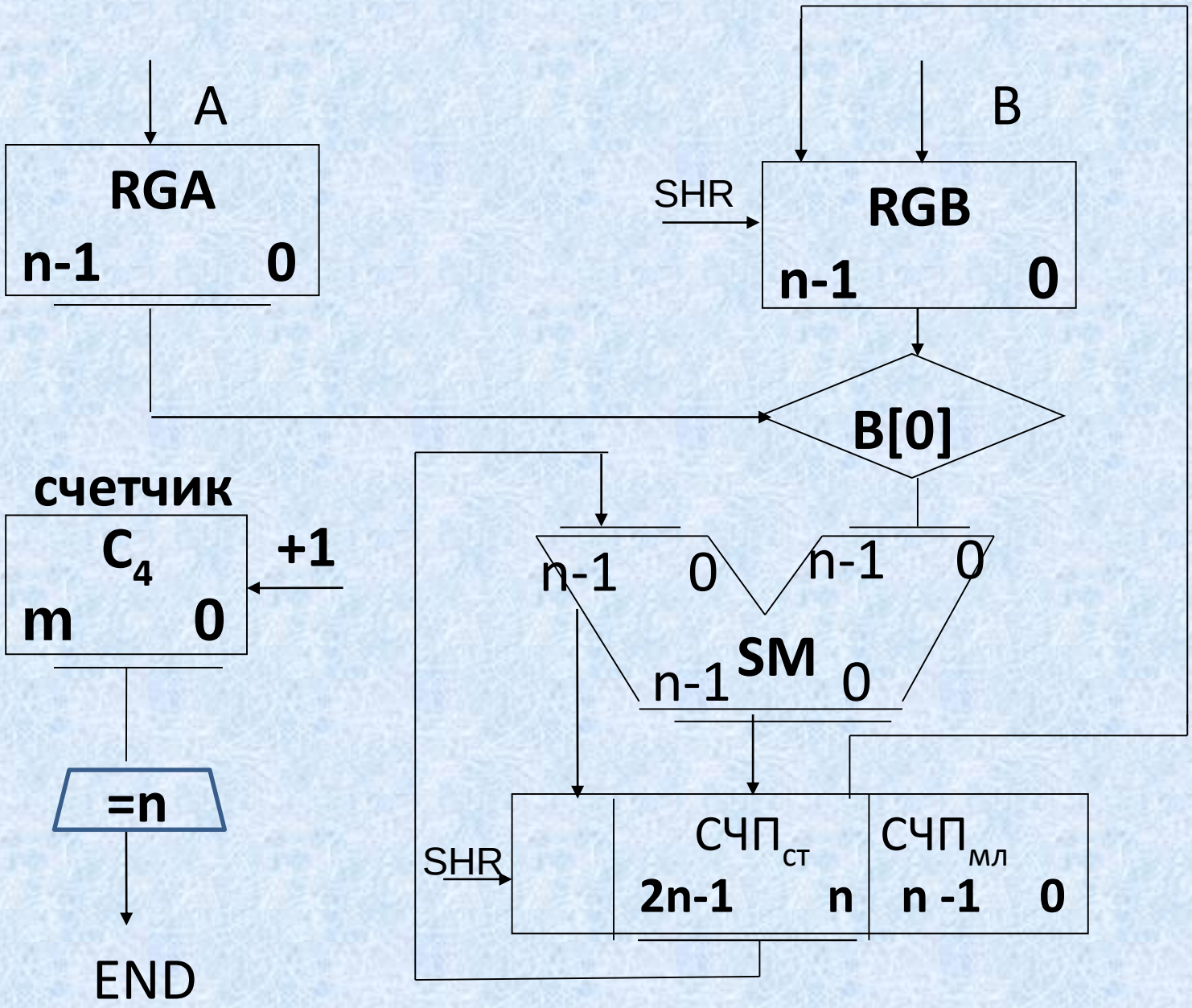
3. В схемах умножения, начиная от старших разрядов со сдвигом множителя вправо, необходимо использовать 2*n*-разрядный регистр.

4. Для схем умножения, начиняя от старших разрядов со сдвигом СЧП влево, требуется 2*n*-разрядный регистр для хранения суммы СЧП.

5. Для схем умножения, начиная от младших разрядов со сдвигом СЧП вправо, требуется n-разрядный регистр.

В целях экономии оборудования практически во всех ЭВМ реализована схема умножения, начиная от младших разрядов множителя со сдвигом СЧП вправо.

Упрощенная схема операционного устройства для реализации умножения по этому способу представлена на рисунке.


A
RGA
n-1
0
B
SHR
RGB
n-1
0
B[0]
счетчик
C4
+1
m
0
n-1
0
n-1
0
SM
n-1
0
=n
SHR
СЧП ст
СЧП мл
2n-1
n
n -1
0
END

Умножение чисел с фиксированной запятой

Основные положения

Использование дополнительных кодов позволяет не переводить отрицательные числа в прямой код и отрицательный результат в дополнительный код.

Знак умножается так же, как разряды, результат получается в нужном коде.

Умножение в дополнительных кодах с применением коррекции.

Правильный результат получается только при положительных операндах.

$$C = [A_{пр.}] \times [B_{пр.}]$$

При A<0, B>0 получаем псевдо-произведение:

$C^* = [A_{доп.}]\cdot[B_{пр.}] = [2^n - |A|]\cdot[B_{пр.}] = \mathbf{2^n \cdot B_{пр.}} - |A| \cdot B_{пр.}$

Должно быть: $C = 2^{2n} - |A| \cdot B_{пр}$

При A>0, B<0: $A_{пр.}\cdot [2^n - |B|] = \mathbf{2^n \cdot A_{пр}} - A_{пр}\cdot|B|$

Должно быть: $C = 2^{2n} - A_{пр}\cdot|B|$

При A<0, B<0: $(2^n - |A|)\cdot(2^n - |B|) =$

$= 2^{2n} \mathbf{- 2^n\cdot|B| - 2^n\cdot|A|} + |A| \cdot |B|$

Должно быть: $C = |A| \cdot |B|$

Два вида коррекции

1. Коррекция окончательного результата состоит в вычитании множимого из старших разрядов СЧП, которое может быть заменено на сложение с дополнением множимого. Эта коррекция проводится **при отрицательном множителе**.

2, При **отрицательном множимом** коррекция проводится во время умножения и сводится к модифицированному сдвигу СЧП (при сдвиге СЧП вправо в свободный разряд заносится единица).

Примечание. В случае отрицательного множимого (A<0,B>0) при умножении на младшие нули проводится обычный (не модифицированный) сдвиг – в свободный разряд заносится 0.

Пример. В разрядной сетке длиной в байт (один разряд знаковый и семь – цифровых) выполнить операцию умножения заданных чисел А и В со всеми комбинациями знаков, используя метод умножения в дополнительных кодах с применением коррекции. При выполнении операции использовать способ умножения с поразрядным анализом множителя, начиная от его младших разрядов со сдвигом СЧП вправо.

A = 15, B = 13, n = 5.

$[A_{пр.}] = 1.1111$ $[B_{пр.}] = 0.1101$

$[A_{доп.}] = 1.0001$

а) A<0, B>0:

№	Дейст-вия	СЧП (старшие разряды)					СЧП (младшие разряды)					Пояснения
0	СЧП	0	0	0	0	0	0	1	1	0	1	Обнуление старших разрядов СЧП.
1	$[A]_{доп.}$	1	0	0	0	1						Сложение СЧП с множимым.
	СЧП	1	0	0	0	1	0	1	1	0	1	
	СЧП→	1	1	0	0	0	1	0	1	1	0	Сдвиг СЧП.
2	СЧП→	1	1	1	0	0	0	1	0	1	1	Сдвиг СЧП.
3	$[A]_{доп.}$	1	0	0	0	1						Сложение СЧП с множимым.
	СЧП	0	1	1	0	1	0	1	0	1	1	
	СЧП→	1	0	1	1	0	1	0	1	0	1	Сдвиг СЧП.

4	$[A]_{доп.}$	1	0	0	0	1						Сложение СЧП с множимым.
	СЧП	0	0	1	1	1	1	0	1	0	1	
	СЧП→	1	0	0	1	1	1	1	0	1	0	Сдвиг СЧП.
5	СЧП	1	1	0	0	1	1	1	1	0	1	

$C_{доп.} = (1.100111101)_2$

$C_{пр.} = (1.011000011)_2 = -195$

b) A>0, B<0:

$[A_{пр.}] = 0.1111$ $[B_{пр.}] = 1.1101$

$[B_{доп.}] = 1.0011$

№	Действия	СЧП (старшие разряды)					СЧП (младшие разряды)					Пояснения
0	СЧП	0	0	0	0	0	1	0	0	1	1	Обнуление старших разрядов СЧП.
1	$[A]_{пр.}$	0	1	1	1	1						Сложение СЧП с множимым.
	СЧП	0	1	1	1	1	1	0	0	1	1	
	СЧП→	0	0	1	1	1	1	1	0	0	1	Сдвиг СЧП.
2	$[A]_{пр.}$	0	1	1	1	1						Сложение СЧП с множимым.
	СЧП	1	0	1	1	0	1	1	0	0	1	
	СЧП→	0	1	0	1	1	0	1	1	0	0	Сдвиг СЧП.
3	СЧП→	0	0	1	0	1	1	0	1	1	0	Сдвиг СЧП.
4	СЧП→	0	0	0	1	0	1	1	0	1	1	Сдвиг СЧП.

5	$[A]_{пр.}$	0	1	1	1	1						Сложение СЧП с множимым.
	СЧП	1	0	0	0	1	1	1	0	1	1	
	СЧП→	0	1	0	0	0	1	1	1	0	1	Сдвиг СЧП.
Ко	$[-A]_{доп.}$	1	0	0	0	1						
рр	СЧП	1	1	0	0	1	1	1	1	0	1	

$C_{пр.} = (1.011000011)_2 = -195$

b) A<0, B<0:

$[A_{пр.}] = 1.1111$ **$[B_{пр.}] = 1.1101$**

$[A_{доп.}] = 1.0001$ **$[B_{доп.}] = 1.0011$**

№	Действия	СЧП (старшие разряды)					СЧП (младшие разряды)					Пояснения
0	СЧП	0	0	0	0	0	1	0	0	1	1	Обнуление старших разрядов СЧП.
1	$[A]_{доп.}$	1	0	0	0	1						Сложение СЧП с множимым.
	СЧП	1	0	0	0	1	1	0	0	1	1	
	СЧП→	1	1	0	0	0	1	1	0	0	1	Сдвиг СЧП.
2	$[A]_{доп.}$	1	0	0	0	1						Сложение СЧП с множимым.
	СЧП	0	1	0	0	1	1	1	0	0	1	
	СЧП→	1	0	1	0	0	1	1	1	0	0	Сдвиг СЧП.
3	СЧП→	1	1	0	1	0	0	1	1	1	0	Сдвиг СЧП.
4	СЧП→	1	1	1	0	1	0	0	1	1	1	Сдвиг СЧП.

5	$[A]_{доп.}$	1	0	0	0	1						Сложение СЧП с множимым.
	СЧП	0	1	1	1	0	0	0	1	1	1	
	СЧП→	1	0	1	1	1	0	0	1	1	1	Сдвиг СЧП.
Ко	$[-A]_{пр.}$	0	1	1	1	1						
рр	СЧП	0	0	1	1	0	0	0	0	1	1	

$C_{пр.} = (0.011000011)_2 = 195$

Умножения в дополнительных кодах без применения коррекции. Метод Бута.

Особенности метода.

Сложение или вычитание множимого на каждом шаге зависит от того, как после сдвига вправо меняется младший разряд множителя.

Если он поменялся с 0 на 1, то происходит вычитание множимого из СЧП. При изменении младшего разряда множителя с 1 на 0 происходит сложение множимого с СЧП.
Если младший разряд множителя не изменился, то производится только сдвиг. При реализации этого метода происходит чередование сложения и вычитания, поэтому старший разряд СЧП в явном виде представляет его знак. При сдвиге знак СЧП сохраняется.

Эффективность метода Бута по сравнению с обычным методом поразрядного умножения проявляется для тех множителей, в которых имеются длинные последовательности единиц.

Примечание.

При умножении на младшую единицу производится вычитание множимого. При умножении на младшие нули осуществляется только сдвиг нулевой СЧП и множителя до появления первой единицы.

Пример. A =11, B = 15, n = 5.

$[A_{пр.}] = 0.1011$ $[B_{пр.}] = 0.1111$
$[A_{доп.}] = 1.0101$ $[B_{доп.}] = 1.0001$

а) A>0, B>0:

№	Действия	СЧП (старшие разряды)					СЧП (младшие разряды)					Пояснения
0	СЧП	0	0	0	0	0	0	1	1	1	1	Обнуление старших разрядов СЧП.
1	$[-A]_{доп.}$	1	0	1	0	1						Младший разряд множителя равен 1: вычитание множимого из СЧП
	СЧП	1	0	1	0	1	0	1	1	1	1	
	СЧП→	1	1	0	1	0	1	0	1	1	1	Сдвиг СЧП.
2	СЧП→	1	1	1	0	1	0	1	0	1	1	Сдвиг СЧП.
3	СЧП→	1	1	1	1	0	1	0	1	0	1	Сдвиг СЧП.
4	СЧП→	1	1	1	1	1	0	1	0	1	0	Сдвиг СЧП.

	$[A]_{пр.}$	0	1	0	1	1						При сдвиге младший разряд множителя изменился с 1 на 0: сложение СЧП с множимым
5	СЧП	0	1	0	1	0	0	1	0	1	0	
	СЧП→	0	0	1	0	1	0	0	1	0	1	Сдвиг СЧП.

$C_{пр.} = (0.010100101)_2 = 165$

а) A<0, B<0:

№	Действия	СЧП (старшие разряды)					СЧП (младшие разряды)					Пояснения
0	СЧП	0	0	0	0	0	1	0	0	0	1	Обнуление старших разрядов СЧП.
1	$[-A]_{пр.}$	0	1	0	1	1						вычитание множимого из СЧП
	СЧП	0	1	0	1	1	1	0	0	0	1	
	СЧП→	0	0	1	0	1	1	1	0	0	0	Сдвиг СЧП.
2	$[A]_{доп.}$	1	0	1	0	1						младший разряд изменился с 1 на 0: сложение СЧП с множимым
	СЧП	1	1	0	1	0	1	1	0	0	0	
	СЧП→	1	1	1	0	1	0	1	1	0	0	

3	СЧП→	1	1	1	1	0	1	0	1	1	0	Сдвиг СЧП.
4	СЧП→	1	1	1	1	1	0	1	0	1	1	Сдвиг СЧП.
5	$[-A]_{пр.}$	0	1	0	1	1						младший разряд множителя изменился с 0 на 1: вычитание множимого из СЧП
	СЧП	0	1	0	1	0	0	1	0	1	1	
	СЧП→	0	0	1	0	1	0	0	1	0	1	Сдвиг СЧП.

$C_{пр.} = (0.010100101)_2 = 165$

Операция целочисленного деления и способы ее реализации в ЭВМ

Особенности двоичного деления

Пример. A = 130, B = 10.

A/B = C(R),

где A – делимое,

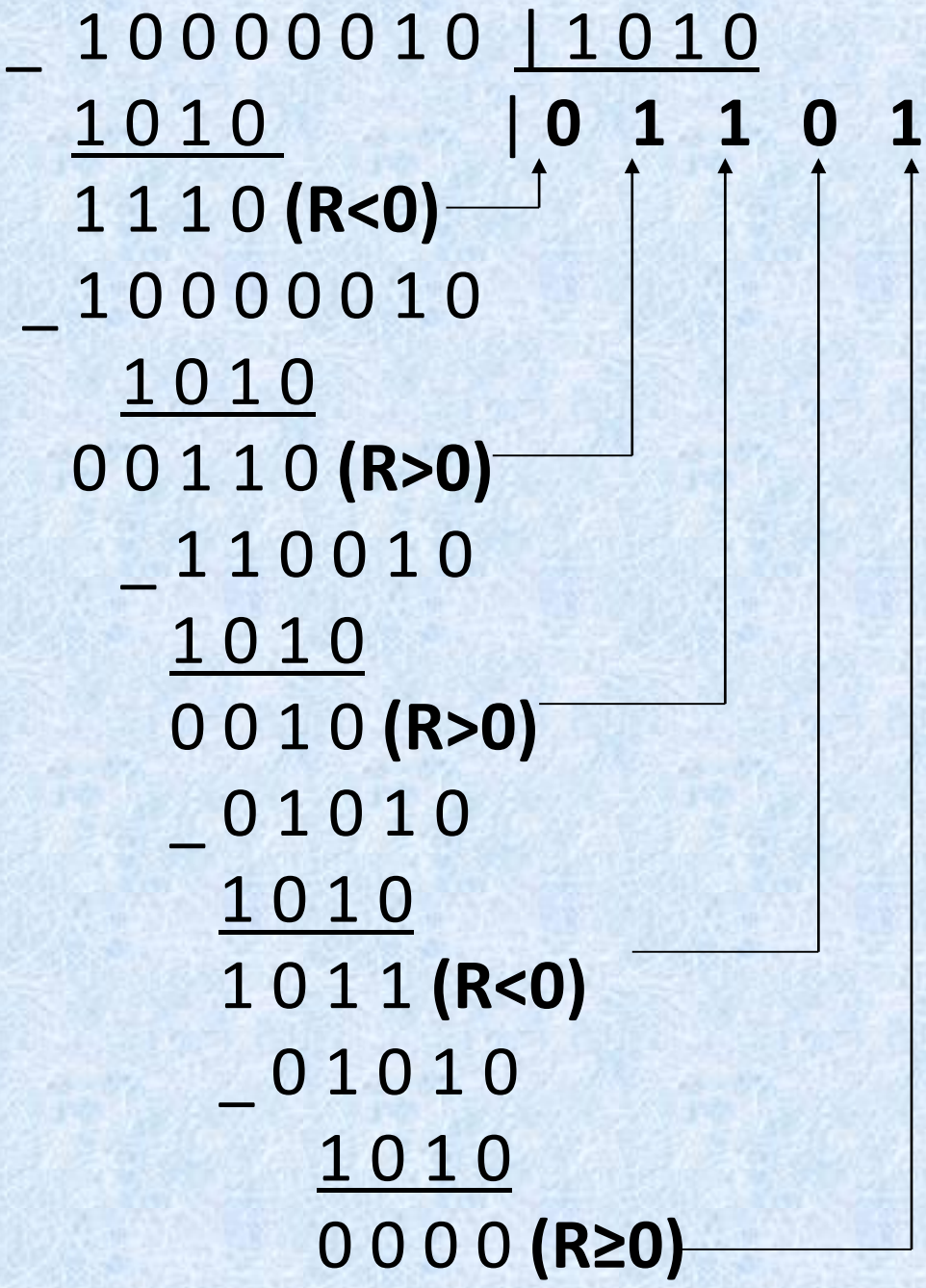
B – делитель,

C – частное,

R – остаток.

$A = (130)_{10} = (10000010)_2$

$B = (10)_{10} = (1010)_2$


1 0 0 0 0 0 1 0 | 1 0 1 0
1 0 1 0 | 0 1 1 0 1
1 1 1 0 (R<0)
1 0 0 0 0 0 1 0
1 0 1 0
0 0 1 1 0 (R>0)
1 1 0 0 1 0
1 0 1 0
0 0 1 0 (R>0)
0 1 0 1 0
1 0 1 0
1 0 1 1 (R<0)
0 1 0 1 0
1 0 1 0
0 0 0 0 (R≥0)

Из проделанного примера отчетливо проявляются следующие особенности двоичного деления:

1) Процесс деления сводится к последовательному вычитанию делителя первоначально из делимого, а далее – из получаемых текущих остатков (под текущим остатком будем понимать промежуточный результат вычитания делителя из делимого на или очередного остатка на последующих шагах).

2) На начальном шаге делитель совмещается со старшими разрядами делимого, а затем на каждом шаге делитель сдвигается на одну позицию (разряд) вправо относительно неподвижного текущего остатка. На последнем шаге делитель совмещается с младшими разрядами текущего остатка.

3) Цифры частного, вырабатываемые на каждом шаге, определяются знаком текущего остатка. Для остатка большего или равного 0 цифра частного равна 1, для остатка, который меньше 0, цифра частного равна 0.

4) О знаке текущего остатка можно судить по наличию или отсутствию заема в старший разряд при вычитании. Если заем есть, то результат вычитания (т.е. текущий остаток) отрицателен. Если заем отсутствует, то результат вычитания и, соответственно, текущий остаток не отрицателен.

5) При получении отрицательного остатка на очередном шаге перед переходом к следующему шагу необходимо выполнить ***восстановление остатка путем сложения отрицательного остатка с делителем.***

Особенности реализации целочисленного деления в ЭВМ

- Т.к. операция деления является обратной по отношению к умножению, а результат сложения n-разрядных сомножителей представляется в 2n-разрядном формате, то делимое по сравнению с делителем так же должно представляться в удвоенном формате.

- В качестве результата деления формируется не только частное, но и остаток. Операция деления с остатком должна удовлетворять следующему соотношению:

$$B \cdot C + R = A$$

Для большинства современных процессоров (в том числе и фирмы Intel) по окончании операции деления частное занимает младшие разряды делимого, а остаток – старшие разряды делимого.

• В целях экономии оборудования на каждом шаге операции деления осуществляется не сдвиг делителя вправо относительно неподвижного текущего остатка, а сдвиг текущего остатка влево относительно неподвижного делителя. При этом делитель должен совмещаться со старшими разрядами сначала делимого, а на последующих шагах – текущего остатка.

• Подробный подход к реализации операции деления называется «метод деления с восстановлением остатка». В целях экономии времени выполнения операции деления в современных процессорах деление реализуется с использованием метода без восстановления остатка. Идея метода сводится к тому, что после получения отрицательного остатка на очередном шаге деления осуществляется его сдвиг влево так же, как и для положительного остатка, однако на следующем шаге производится не вычитание делителя из остатка, а сложение делителя с остатком.

Обоснование метода

Допустим, что на i-ом шаге деления получен остаток $R_i<0$, тогда для получения очередного остатка R_{i+1} на (i+1)-ом шаге над текущим остатком R_i выполняются следующие действия:

а) По методу с восстановлением остатка:

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

$(R_i + B)$ – сложение с делителем;

$(R_i + B)\cdot 2$ – сдвиг влево на 1 разряд;

$(R_i + B)\cdot 2 - B$ – вычитание делителя.

б) По методу без восстановления остатка:

$$R_{i+1} = 2R_i + B$$

$2R_i$ – сдвиг отрицательного остатка влево;

$2R_i + B$ – сложение с делителем.

• При некоторых соотношениях между делимым и делителем может оказаться, что частное не помещается в формат делителя. Подобная ситуация возникает также, если делитель равен 0. Этот особый случай распознается и фиксируется на начальных шагах операции деления. Наличие подобного случая классифицируется как некорректность целочисленного деления и приводит к прерыванию выполняемой программы. Выход на это прерывание может быть зафиксирован при попытке разделить двухбайтное делимое, равное 1000, на однобайтный делитель, равный 1,2 или 3, при выполнении команды DIV (беззнаковое деление), а при выполнении команды IDIV также при B = 4 или 5.

Деление беззнаковых целых чисел

В основном беззнаковое деление реализуется по общим принципам, описанным ранее. Это означает, что цифра частного, вырабатываемая на каждом шаге деления, определяется знаком текущего остатка, получаемого на данном шаге. При отрицательном остатке вырабатывается цифра частного, равная 0; при положительном остатке – равная 1.

Проверка корректности беззнакового деления

Для n-разрядного делителя и, следовательно, n-разрядного частного условие корректности принимает вид:

$$A/B \le 2n - 1 \Rightarrow A/B < 2n$$

$$A < B \cdot 2n$$

$$\mathbf{A - B \cdot 2n < 0}$$

Из полученного условия следует, что для проверки корректности беззнакового деления необходимо произвести вычитание делителя из старших разрядов делимого. Если полученная разность отрицательна, то деление корректно, т.е. частное помещается в формате делителя. Если же результат так называемого «пробного вычитания» положителен или равен 0, то результат деления в виде частного не может быть помещен в n-разрядный формат. При обнаружении такого случая генерируется прерывание выполняемой программы по причине ***некорректности целочисленного деления***. *Для процессоров Intel это прерывание является стандартным, ему присвоен номер, равный нулю.*

Пример беззнакового деления.

A = 141, B = 13, C = 10, R = 11.

Минимальный формат делителя для выполнения операции n=4.

$A = (141)_{10} = (10001101)_2$

$B = (13)_{10} = (1101)_2$

N шага	Операнды	Действия		Комментарии
		Делимое/остаток старш. разр.	Делимое/остаток младш. разр.	
0				
	A	1 0 0 0	1 1 0 1	
	B	1 1 0 1		
	$R_0<0$	1 0 1 1	1 1 0 1	Делимое корректно

1	$\overleftarrow{R_0}$ B $R_1>0$	+0111 1101 0100	1010 1011	
2	$\overleftarrow{R_1}$ B $R_2<0$	1001 1101 1100	0110 0110	
3	$\overleftarrow{R_2}$ B $R_3>0$	+1000 1101 0101	1100 1101	
4	$\overleftarrow{R_3}$ B $R_4<0$	1011 1101 1110	1010 1010	частное = $(10)_{10}$
5	B R	1101 1011		Коррекция остатка. Остаток = $(11)_{10}$

Пояснения к примеру.

1. Формируемые на каждом шаге (кроме первого) цифры частного помещаются в освобождающийся при сдвиге влево младший разряд остатка.
2. Знак остатка определяется наличием или отсутствием переноса из старшего разряда при сложении или заема в старший разряд при вычитании. Наличие переноса при сложении свидетельствует о получении положительного остатка, а его отсутствие – о получении отрицательного остатка. В свою очередь наличие заема при вычитании свидетельствует о получении отрицательного остатка, а его отсутствие – о получении положительного остатка.

3.По результату пробного вычитания на начальном (нулевом) шаге осуществляется проверка коррек-тности деления, поэтому на данном шаге никакой цифры частного не формируется.

Пример фиксации некорректности деления.

A = 141, B = 8.

N шага	Операнды	Действия		Комментарии
		Делимое/остаток старш. разр.	Делимое/остаток младш. разр.	
0	A B $R_0 \geq 0$	1000 1000 0000	1101	Заем отсутствует Некорректность деления

Для рассматриваемых делимого и делителя С=17. Аналогичная ситуация фиксации некорректности деления будет иметь место и при B<8, в том числе и при B=0.

Возможные модернизации метода деления

1. Для явного представления знака остатка можно использовать дополнительный старший бит как в регистре остатка, так и в сумматоре-вычитателе. Следует иметь в виду, что при сдвиге остатка влево может произойти искажение старшего бита остатка, интерпретируемого как знаковый. В связи с этим действия на очередном шаге алгоритма деления следует определять значением знакового бита остатка до сдвига, а не после.

2. Для упрощения метода операцию вычитания делителя можно заменить операцией сложения с его дополнительным кодом.

В процессорах Intel беззнаковое деление реализуется командой DIV, а знаковое - командой IDIV.

Деление знаковых чисел

Аналогично операции умножения знаковое деление может быть реализовано одним из двух методов:

- метод деления в прямых кодах;
- метод деления в дополнительных кодах.

Основные особенности метода деления в прямых кодах

1. Отрицательные операнды предварительно преобразуются из дополнительного кода в прямой.
2. Деление модулей операндов осуществляется аналогично делению для беззнаковых чисел.
3. Знак частного вырабатывается отдельным действием как сумма по модулю 2 знаковых разрядов операндов.
4. Знаку остатка присваивается знак делимого. Исключением из правил 3,4 является получение нулевого частного и(или) нулевого остатка, которым, независимо от знаков операндов, присваивается знак + (положительный ноль).

5. Отрицательные результаты в конце операции преобразуются из прямого кода в дополнительный.

6. Существенным отличием знакового деления в прямых кодах от беззнакового деления является проверка корректности деления, выполняемая на начальных шагах алгоритма.

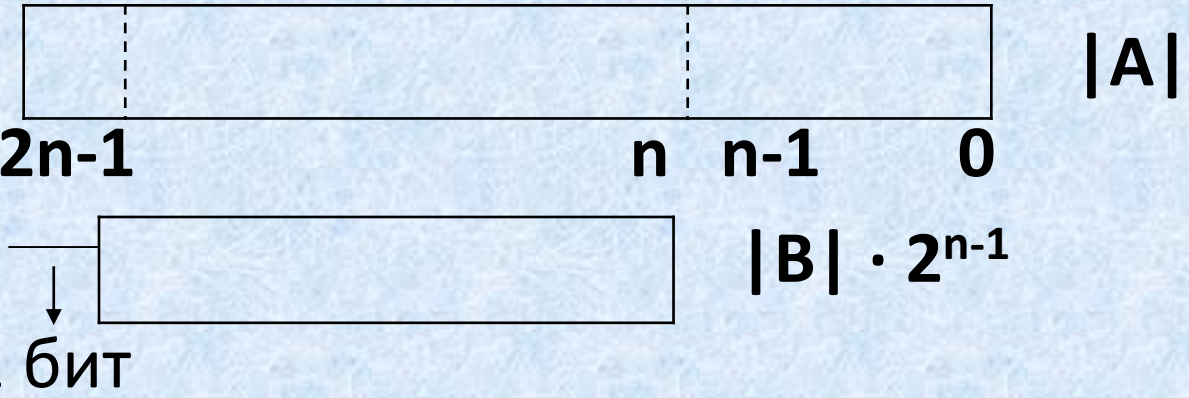
Обоснование метода проверки корректности деления

В связи с тем, что старший бит n-разрядного делителя и, соответственно, частного отводится для представления знака, условие корректности для деления модулей операндов имеет вид:

$|A| / |B| \le 2^{n-1} - 1 \Rightarrow |A| / |B| < 2^{n-1}$

$$|A| / |B| \cdot 2^{n-1} < 0$$

Из полученного соотношения следует, что на этапе пробного вычитания следует делимое и делитель расположить друг под другом следующим образом:



В целях однообразия выполнения пробного вычитания с основным циклом деления, в котором делитель размещается под старшими разрядами остатка, необходимо перед выполнением пробного вычитания сдвинуть делимое на один разряд влево и после этого произвести вычитание делителя из его старших разрядов.

Основные особенности метода деления в дополнительных кодах

1. Цифры частного, формируемые на каждом шаге, определяются не только знаком остатка, но и знаком делителя. При их совпадении цифра частного равна 1, при несовпадении – 0.
2. Действие, выполняемое над текущим остатком на каждом шаге, определяется не только знаком остатка, но и знаком делителя. При их совпадении производится вычитание делителя из старших разрядов остатка, а при несовпадении – сложение делителя со старшими разрядами остатка. Вычитание делителя может заменяться сложением с его дополнительным кодом.

3. Коррекция остатка выполняется в конце операции (после выработки всех цифр частного) в том случае, если знак последнего остатка не совпадает со знаком делимого. Эта коррекция осуществляется действием над остатком (прибавление или вычитание делителя), аналогичным действиям в основном цикле деления (определяется сравнением знаков остатка и делителя).
4. Коррекция частного выполняется только при отрицательном делимом и нулевом остатке и состоит в инкременте (увеличении на единицу) для положительного частного и декременте для отрицательного частного.

5. Проверка корректности деления реализуется аналогично методу деления в прямых кодах только в случае положительных операндов. Для остальных комбинаций знаков имеют место следующие нюансы:

Обоснование методов проверки корректности

1) **A>0, B>0.**

$A/B < 2^{n-1}$;

$A - B \cdot 2^{n-1} < 0$.

2) **A<0, B<0.**

$A/B < 2^{n-1}$;

$A - B \cdot 2^{n-1} > 0$.

Из полученных соотношений следует, что **проверка корректности при одинаковых знаках делимого и делителя** реализуется следующим образом:

а) выполняется предварительный сдвиг делимого на один разряд влево;

б) из старших разрядов сдвинутого делимого вычитается делитель;

в) знак полученного остатка сравнивается со знаком делимого. Если они совпадают, то деление некорректно, и операция завершается выходом на прерывание. Если знаки делимого и первого остатка не совпадают, то формируется старший разряд частного, интерпретируемый как знак, и осуществляется переход к основному циклу деления.

Знак частного формируется по тем же правилам, что и любая его цифра, т.е. как результат сравнения знаков текущего остатка и делителя.

3) A<0, B>0.

$A/B \geq -2^{n-1}$;

$A/B \geq -2^{n-1} - 1$;

$A + B \cdot 2^{n-1} + B > 0$.

4) A>0, B<0.

$A/B > -2^{n-1} - 1$;

$A + B \cdot 2^{n-1} + B < 0$.

Из полученных соотношений следует, что **проверка корректности деления при разных знаках делимого и делителя** выполняется в такой последовательности:

1. сложение делителя с младшими разрядами делимого;
2. сдвиг полученного остатка влево;
3. сложение делителя со старшими разрядами остатка;
4. знак полученного остатка сравнивается со знаком делимого. Если они совпадают, то деление некорректно, и операция завершается прерыванием. Если знаки делимого и первого остатка не совпадают, то формируется старший разряд частного, интерпретируемый как знак, и осуществляется переход к основному циклу деления. Знак частного формируется по тем же правилам, что и любая его цифра.

Пример. Выполнить операцию деления заданных чисел А и В со всеми комбинациями знаков, используя метод деления в дополнительных кодах. Для представле-ния делимого (А) использовать 16 двоичных разря-дов (один-знаковый,15-цифровых), для представ-ления делителя (В) – 8 разрядов (один-знаковый,7-цифровых). Остаток от деления и частное представляются в той же разрядной сетке, что и делитель.

А = 139, В = 13.

$[A]_{пр.} = 0.010001011$, $[A]_{доп} = 1.101110101$;

$[B]_{пр.} = 0.1101$, $[B]_{доп} = 1.0011$.

а) А<0; В<0.

N шага	Операнды	Действия: Делимое/остаток (старшие разряды)	Действия: Делимое/остаток (младшие разряды)	Комментарии
0	$[A]_{доп}$	1 1 0 1 1	1 0 1 0 1	Делимое
1	$\leftarrow[A]_{доп}$ $[-B]_{пр}$ R_1	-1 0 1 1 1 0 1 1 0 1 0 0 1 0 0 $signR_1 \neq signB$	0 1 0 1 | 0 0 1 0 1 | 0	Сдвиг А. Вычитание В. Деление корректно.
2	$\leftarrow R_1$ $[B]_{доп}$ R_2	-0 1 0 0 0 1 0 0 1 1 1 1 0 1 1 $signR_2 = signB$	1 0 1 0 | 0 1 0 1 0 | 1	Сдвиг остатка влево. Сложение с В. Формирование цифры частного.
3	$\leftarrow R_2$ $[-B]_{пр}$ R_3	-1 0 1 1 1 0 1 1 0 1 0 0 1 0 0 $signR_3 \neq signB$	0 1 | 0 1 0 0 1 | 0 1 0	Сдвиг остатка влево. Вычитание делителя. Формирование цифры частного.
4	$\leftarrow R_3$ $[B]_{доп}$ R_4	-0 1 0 0 0 1 0 0 1 1 1 1 0 1 1 $signR_4 = signB$	1 | 0 1 0 0 1 | 0 1 0 1	Сдвиг остатка влево. Сложение с В. Формирование цифры частного.
5	$\leftarrow R_4$ $[-B]_{пр}$ R_5	-1 0 1 1 1 0 1 1 0 1 0 0 1 0 0 $signR_5 \neq signB$	0 1 0 1 0 0 1 0 1 0	Сдвиг остатка влево. Вычитание делителя. Формирование цифры частного.
6	$[B]_{доп}$ R_6	1 0 0 1 1 1 0 1 1 1	0 1 0 1 0	Коррекция остатка: сложение с делителем Результат

В результате выполнения операции получено положительное частное:

$[C]_{пр} = (0.1010)_2 = (+10)_{10}$

и отрицательный остаток

$[B]_{доп} = (1.0111)_2 = (-9)_{10}$,

которые соответствуют значениям:

$10 \cdot (-13) + 9 = -139$.

а) A<0, B>0.

N шага	Операнды	Действия: Делимое/остаток (старшие разряды)	Действия: Делимое/остаток (младшие разряды)	Комментарии
0	$[A]_{доп}$	1 1 0 1 1	1 0 1 0 1	Делимое
1	$[B]_{пр}$ $[-B]_{пр}$ R_1	0 0 0 0 0 1 1 1 0 0 +1 1 0 0 0 0 1 1 0 1 0 0 1 0 1 $signR_1 = signB$	0 1 1 0 1 0 0 0 1 0 0 0 1 0 $\vert$ 0 0 0 1 0 $\vert$ 1	Сложение с делителем, выровненным по младшим разрядам. Сдвиг остатка влево. Сложение с делителем, выровненным по старшим разрядам. Деление корректно.
2	$\leftarrow R_1$ $[-B]_{доп}$ R_2	+0 1 0 1 0 1 0 0 1 1 1 1 1 0 1 $signR_2 \neq signB$	0 1 0 $\vert$ 1 0 0 1 0 $\vert$ 1 0	Сдвиг остатка влево. Вычитание делителя. Формирование цифры частного.
3	$\leftarrow R_2$ $[B]_{пр}$ R_3	+1 1 0 1 0 0 1 1 0 1 0 0 1 1 1 $signR_3 = signB$	1 0 $\vert$ 1 0 0 1 0 $\vert$ 1 0 0	Сдвиг остатка влево. Сложение с B. Формирование цифры частного.
4	$\leftarrow R_3$ $[-B]_{доп}$ R_4	+0 1 1 1 1 1 0 0 1 1 0 0 0 1 0 $signR_4 = signB$	0 $\vert$ 1 0 1 0 0 $\vert$ 1 0 1 1	Сдвиг остатка влево. Вычитание делителя. Формирование цифры частного.
5	$\leftarrow R_4$ $[-B]_{доп}$ R_5	+0 0 1 0 0 1 0 0 1 1 1 0 1 1 1 $signR_5 \neq signB$	1 0 1 1 0 1 0 1 1 0	Сдвиг остатка влево. Вычитание делителя. Формирование цифры частного.

В результате выполнения операции получено отрицательное частное: $[C]_{доп} = (1.0110)_2 = (-10)_{10}$ и отрицательный остаток: $[B]_{доп} = (1.0111)_2 = (-9)_{10}$, соответствующие значениям: $(-10)\cdot 13 + (-9) = -139$.

Особый случай алгоритма

Такой случай может иметь место только при отрицательном делимом, нулевом остатке и четном частном. В этом случае получаемый окончательный остаток оказывается не равным нулю, но не подлежит коррекции по общим правилам. Для устранения этого недостатка необходимо модернизировать предлагаемый алгоритм, расширив его на подобные ситуации.

$[A]_{пр} = (-168)_{10} = (1.0101000)_2$

$[B]_{пр} = (-14)_{10} = (1.1110)_2$

$[A]_{доп} = (1.1011000)_2$ $[B]_{доп} = (1.0010)_2$

N шага	Операнды	Действия		Комментарии
		Делимое/остаток (старшие разряды)	Делимое/остаток (младшие разряды)	
0	$[A]_{доп}$ $\leftarrow[A]_{доп}$ $[-B]_{пр}$ R_0	1 1 0 1 0 +1 0 1 0 1 0 1 1 1 0 0 0 0 1 1 signR₁ =signB	1 1 0 0 0 1 0 0 0¦ 0 1 0 0 0¦ 0	Делимое Сдвиг делимого влево. Вычитание делителя. Деление корректно.
1	$\leftarrow R_0$ $[B]_{доп}$ R_1	+0 0 1 1 1 1 0 0 1 0 1 1 0 0 1 signR₂ =signB	0 0 0¦ 0 0 0 0 0¦ 0 1	
2	$\leftarrow R_1$ $[-B]_{пр}$ R_2	+1 0 0 1 0 0 1 1 1 0 0 0 0 0 0 signR₃ =signB	0 0¦ 0 1 0 0 0¦ 0 1 0	
3	$\leftarrow R_2$ $[B]_{доп}$ R_3	+0 0 0 0 0 1 0 0 1 0 1 0 0 1 0 signR₄ =signB	0¦ 0 1 0 0 0¦ 0 1 0 1	
4	$\leftarrow R_3$ $[-B]_{пр}$ R_4	+0 0 1 0 0 0 1 1 1 0 1 0 0 1 0 signR₄ =signB	¦0 1 0 1 0 ¦0 1 0 1 1	
5	R_4 $[B]_{доп}$ R_5	_1 0 0 1 0 1 0 0 1 0 0 0 0 0 0		Псевдокоррекция остатка
6	R_5 $[B]_{доп}$ R_6		+0 1 0 1 1 0 0 0 0 1 0 1 1 0 0	коррекция частного

Арифметические операции над числами с плавающей запятой

Операции сложения и вычитания

Рассмотрим основные нюансы операции сложения на примере для десятичных чисел.

$$A = 0,945 \cdot 10^{3} \leftarrow P_A$$

мантисса MA

$$B = 0,847 \cdot 10^{2} \leftarrow P_B$$

мантисса MB

Первоначально необходимо привести операнды к одному порядку. Для этой цели мантиссу числа с меньшим порядком сдвигают влево на один разряд (величина разности порядка).

$B' = 0,\underbrace{0847}_{\text{МВ}} \cdot 10^3 \leftarrow P_B = P_A$

$$M_A = 0,945$$
$$\underline{M_B = 0,0847}$$
$$M_C = M_A + M_B = 1,0297$$

$P_{c=3}$

В результате сложения получается ненормализованный результат (сумма мантисс больше 1, при сложении мантисс происходит переполнение формата). Для приведения результата к нормализованному виду необходимо сдвинуть его мантиссу вправо и увеличить порядок на 1. Таки образом результат сложения $C = 0,10297 \cdot 10^4$.