

Арифметические основы ЭВМ

Что написано на доске

H_2SO_4

(серная кислота)

Что видит учитель

$2+2=4$

Что видят ученики

国家统计局的解释，"幸福指数"用来反映普通百姓对自己的生活条件，包括收入、就业、保障的满意度以及对自然环境的感受。

Что запоминают ученики



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 конеч-

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

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



Под **весом разряда** принято понимать количест-венное значение цифры данного разряда в числе. Фактически, вес разряда представляет собой множитель, на который умножается цифра этого разряда при

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. доказывається, что оптимальной (с точки зрения затрат оборудования на представление и хранение чисел) является система счисления с основанием $e \approx 2,72$.

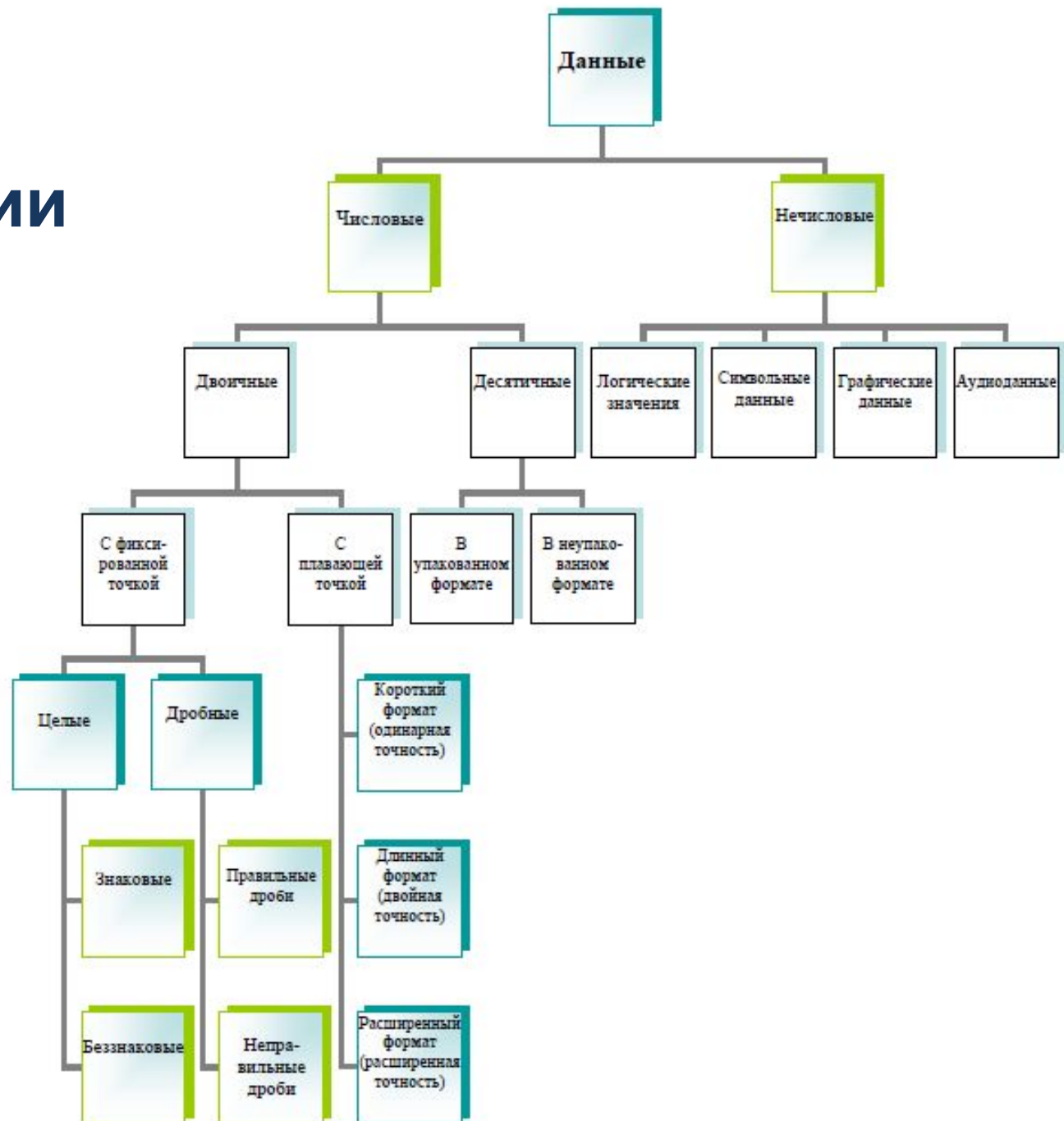
Представление чисел в ЭВМ

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

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;

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 **BСD-форматом** (Binary Coded Decimal).

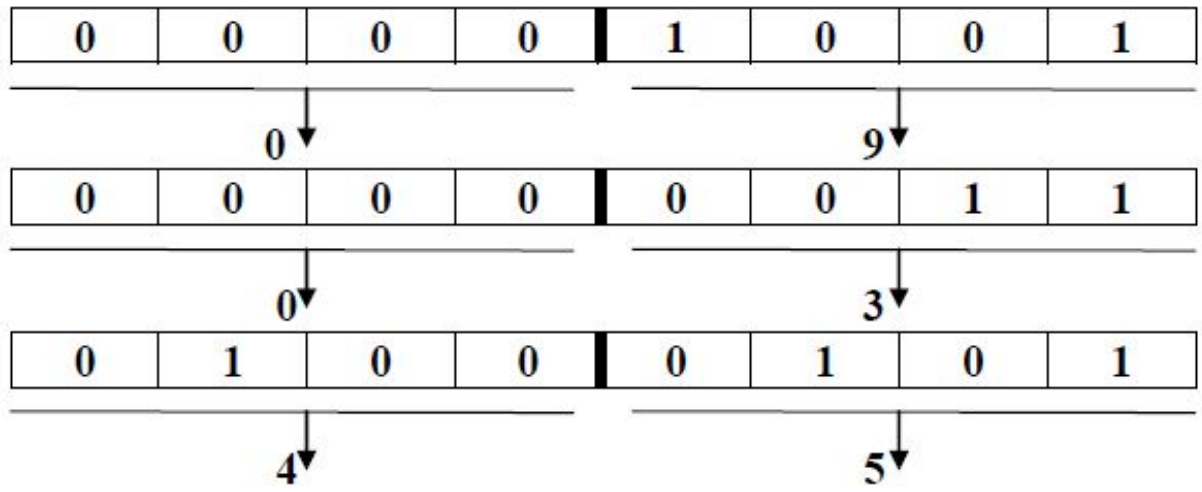
В **неупакованном (распакованном) формате** в каждом байте числа представляется только одна десятичная цифра. Типичным примером неупакованного формата является представление десятичных

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 $(0011)_2 = (3)_{10}$.

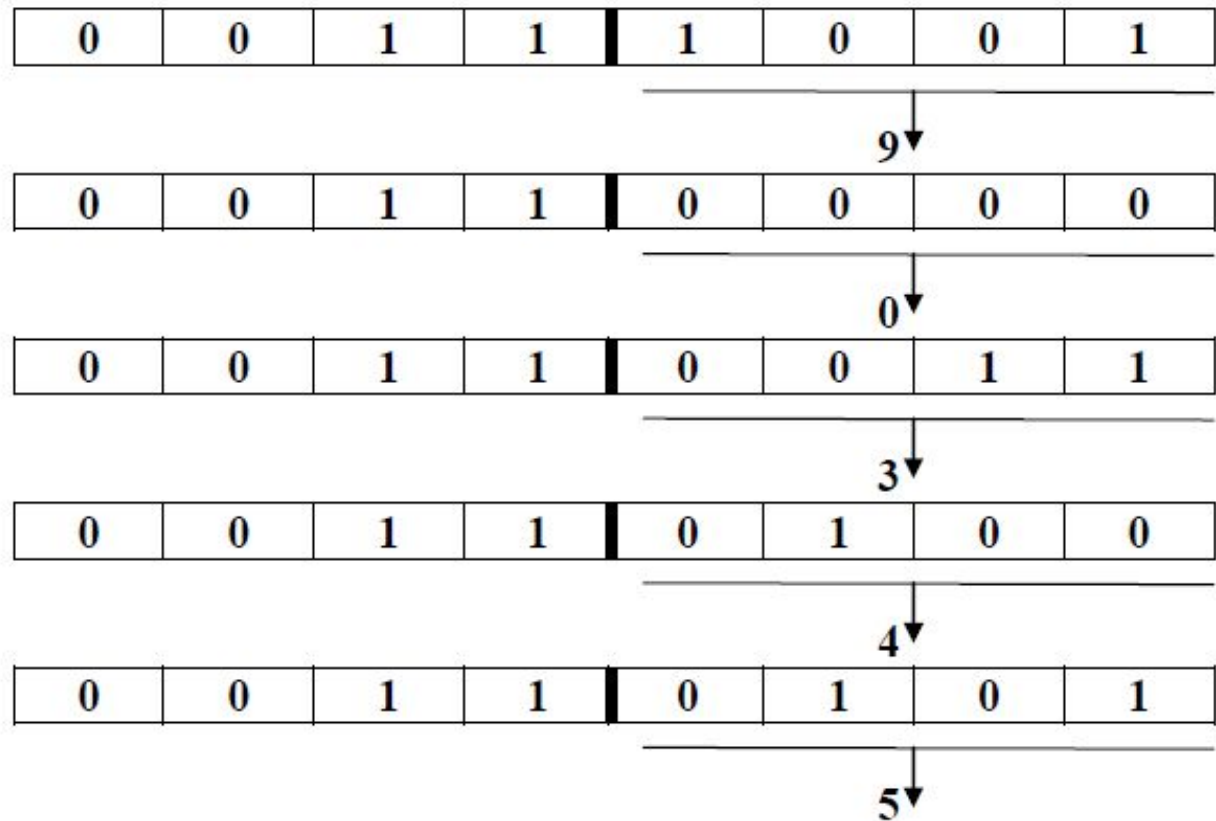
Пример.

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

BCD:



ASCII:



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

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

Двоичная и десятичная арифметики и области их применения

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



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 расчеты

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 аппаратные средства** 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

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:

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

Пример.

$$n = 5 \quad X = -13$$

$$[X]_{\text{пр}} = 1.1101 = 2^4 + 13 = 29$$

$$[X]_{\text{доп}} = 2^5 - |X| = \begin{array}{r} _1.0000 \\ \quad \underline{1.1101} \\ \quad 1.0011 \end{array} = 1.0011 = 32 - 13 = 19$$

Основными форматами, в которых представляются целые числа, являются:

- 1 байт ;
- 2 байта – слово;
- 4 байта – двойное слово;
- 8 байтов – учетверенное слово.

Для байтного формата: $n = 8$

$$0 \leq A_{3n} \leq 255$$

Диапазон представления дробных чисел

Для правильной n -разрядной дроби:

$$2^{-n} \leq |A_{\text{др}}| \leq 1 - 2^{-n}$$

0.00...01 0.11...1

Неправильная дробь в целой части содержит обязательную и единственную единицу.

$$1 \leq |A_{\text{др.нпр.}}| \leq 2 - 2^{-(n+1)}$$

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:

$$A_{пз} = (-1)^{\text{sign}A} \cdot M_A \cdot S^{P_A}$$

signA – знак;

P_A – порядок числа;

M_A – мантисса числа A;

S – основание

Основные особенности представления чисел с плавающей запятой в современных ЭВМ

Классы ЭВМ:

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:

1. Универсальные ЭВМ – $S = 16$;

Классы: 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

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

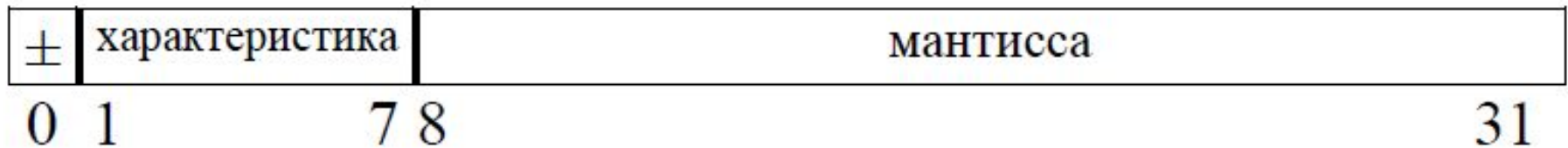
Определяется в отношении модуля
нормализованного числа.

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

Для классов:

MF

S = 16, d = 64 – смещение порядка.



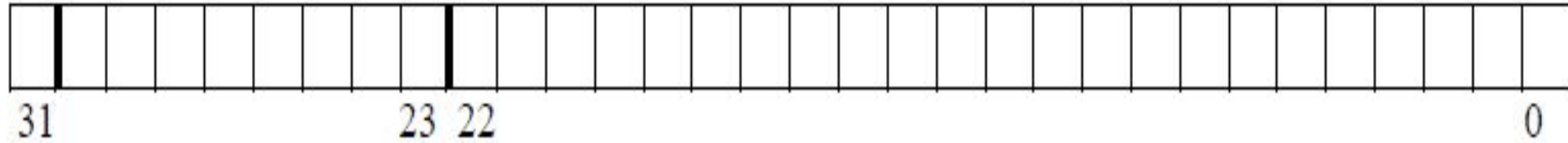
Характеристика: $0 \leq X_A \leq 127$;

Порядок: $-64 \leq P_A \leq 63$.

$$1/16 \cdot 16^{-64} \leq |A_{\text{п.т. норм}}| < 1 \cdot 16^{63}$$

Мини ЭВМ

$S = 2$; $d = 128$ – смещение порядка.



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

Порядок: $-128 \leq P_A \leq 127$.

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

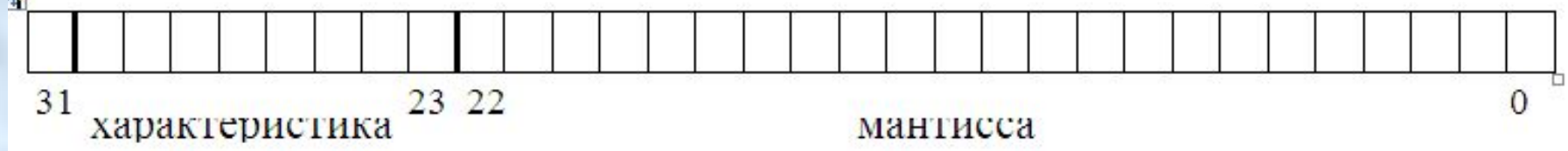
Стандарт IEEE

$S = 2$;

$d = 127$ – 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 значения 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

Максимальное значение характеристики с единицей в старшем разряде мантиссы используется для представления $-\infty$.

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: $2^{-126} \leq |A_{\text{п.т. норм}}| < 2^{128}$.

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

для расширенного формата:

$$10^{-4932} \leq |A| < 10^{4932}$$

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

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

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

$$\delta_{A_{n.m.}} = \left| \frac{\Delta A_{n.m.}}{A_{n.m.}} \right| = \left| \frac{\Delta M_A \cdot S^{P_A}}{M_A \cdot S^{P_A}} \right| = \left| \frac{\Delta M_A}{M_A} \right| = \left| \frac{2^{-n}}{1/S} \right| = S \cdot 2^{-n}$$

Формула справедлива для правильных и неправильных дробей.

Точность представления чисел для различных типов машин

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

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

$$\text{IEEE: } \delta A_{\text{п.т.}} = 10^{-7}.$$

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 1;
- **Округление к ближайшему большему** (округление к $+\infty$)- для положительных чисел 1 добавляется в младший разряд мантиссы, для отрицательных чисел мантисса не меняется.
- **Округление к ближайшему меньшему** (округление к $-\infty$) – 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 арифмети-ческими 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 заемом в два старших разряда при вычитании (если они совпадают, то флаг равен нулю).

Выполнение арифметических операций в ЭВМ

Сложение целых чисел

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

перенос P_{i-1}	сумм. P_i	P_{i-1}	S_i	P_i
0	0	0	0	0
0	0	1	1	0
0	1	0	0	1
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

перенос из предыдущего разряда;

перенос из i -го разряда;

сумма i -го разряда.

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

$A > 0, B > 0.$

Интерпретации

$A_{\text{пр}}$	0	1	1	1	0	0	1	57	57
$B_{\text{пр}}$	0	0	1	1	0	0	0	49	49
$C_{\text{пр}}$	0	1	1	0	1	0	1	106	106

Знаковая



CF=0

ZF=0

PF=1

SF=0

AF=0

OF=0

$A > 0, B < 0.$

Интерпретации
Зн. Беззн.

$A_{\text{пр.}}$	0	0	1	1	1	0	0	1	57	57	
$B_{\text{доп.}}$	1	1	0	0	1	1	1	1	-49	207	
$C_{\text{пр.}}$	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

+	$A_{\text{доп}}$	1	1	0	0	0	1	1	1	3н.	Беззн.
										-57	199
+	$B_{\text{пр.}}$	0	0	1	1	0	0	0	1	+	+
										49	49
	$C_{\text{доп}}$	1	1	1	1	1	0	0	0		248
	$C_{\text{пр.}}$	1	0	0	0	1	0	0	0	-8	



CF=0

ZF=0

PF=0

SF=1

AF=0

OF=0

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

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

$A > 0, B > 0.$

$A_{\text{пр.}}$	0	0	1	1	1	0	0	1
$+ B_{\text{пр.}}$	0	1	1	0	0	0	0	0
$\hline C_{\text{дон}}$	1	0	0	1	1	0	0	1
$C_{\text{пр.}}$	1	1	1	0	0	1	1	1

Интерпретации

$+ 37.$	$+ 37.$
$+ 96$	$+ 96$
\hline	\hline
-103	153
$?$	

CF=0

ZF=0

PF=1

SF=1

AF=0

OF=1



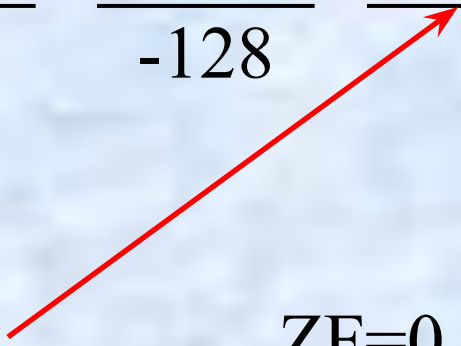
A < 0, B < 0.

A _{доп.}	1	1	0	0	1	1	1	1
+ B _{доп.}	1	0	1	1	0	0	0	1
C_{доп.}	1	0	0	0	0	0	0	0
C _{пр.}	1	0	0	0	0	0	0	0

Интерпретации

	Зн.	Беззн.
+ -79	+ 177	
+ -49	+ 207	
		128?
-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

таблица:

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. $A = 67, B = 51.$

$[A]_{\text{пр.}} = 0.1000011$

$[B]_{\text{пр.}} = 0.0110011$

$A > 0, B > 0.$

Интерпретации

Зн. Беззн.

$A_{\text{пр.}}$	0	1	0	0	0	0	1	1	67	67
$B_{\text{пр.}}$	0	0	1	1	0	0	1	1	51	51
$C_{\text{пр.}}$	0	0	0	1	0	0	0	0	16	16

CF=0

ZF=0

PF=0

SF=0

AF=0

OF=0

$A < 0, B > 0.$

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

$[B]_{\text{пр.}} = 0.0110011$

Интерпретации

$A_{\text{доп.}}$	1	0	1	1	1	1	0	1	367	189
$B_{\text{пр.}}$	0	0	1	1	0	0	1	1	51	51
$C_{\text{доп.}}$	1	0	0	0	1	0	1	0		138
$C_{\text{пр.}}$	1	1	1	1	0	1	1	0	-118	

CF=0 ZF=0

PF=0 SF=0

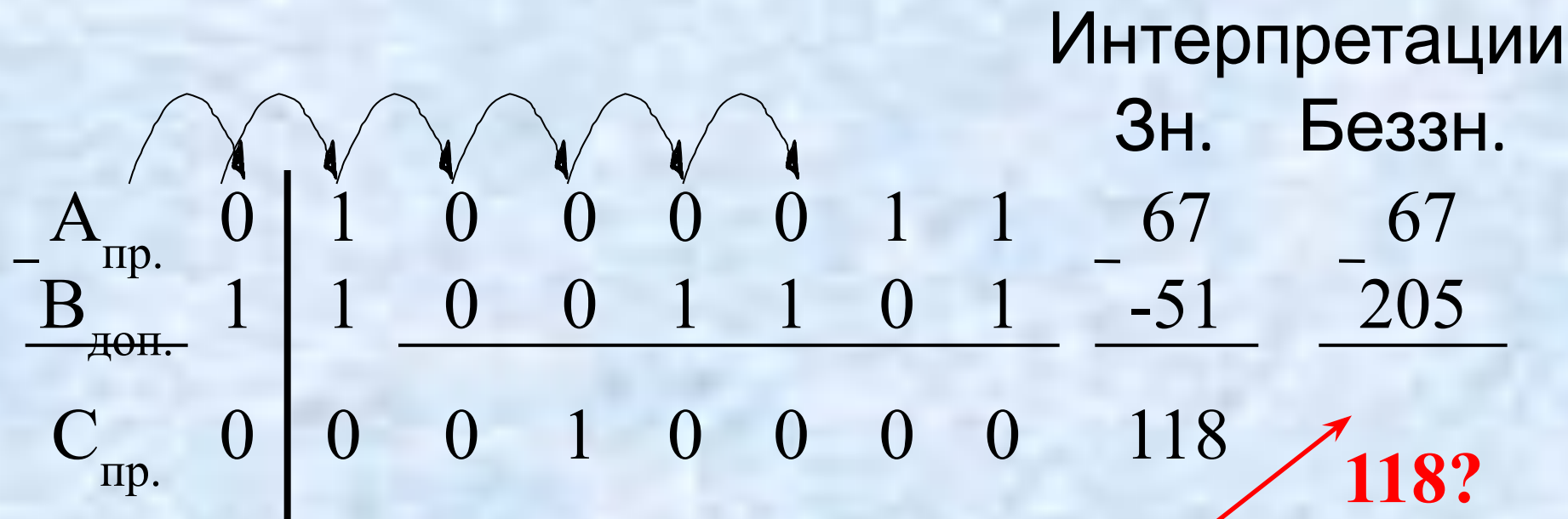
AF=0 OF=0

$A < 0, B > 0.$

$[A]_{\text{пр.}} = 1.1000011;$

$[B]_{\text{пр.}} = 1.0110011; [B]_{\text{доп.}} = 1.1001101$

								Интерпретации		
								Зн.	Беззн.	
$A_{\text{пр.}}$	0	1	0	0	0	0	1	1	67	67
$B_{\text{доп.}}$	1	1	0	0	1	1	0	1	-51	205
$C_{\text{пр.}}$	0	0	0	1	0	0	0	0	118	



CF=1 ZF=0

PF=0 SF=0

AF=1 OF=0

118?

$A < 0, B < 0.$

Интерпретации

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

CF=1 ZF=0
 PF=1 SF=1
 AF=0 OF=0

2. $A = 67, B = 64.$

$A < 0, B > 0.$

Интерпретации

									Зн	Беззн
$A_{\text{доп.}}$	1	0	1	1	1	1	0	1	-67	189
$\overline{B}_{\text{пр.}}$	0	1	0	0	0	0	0	0	$\overline{64}$	$\overline{64}$
$C_{\text{пр.}}$	0	1	1	1	1	1	0	1	125?	125

CF=0

ZF=0

PF=1

SF=1

AF=0

OF=1



$A > 0, B < 0.$

Интерпретации

Зн Беззн

$A_{\text{пр.}}$	0	1	0	0	0	0	1	1	67	67
$B_{\text{доп.}}$	1	1	0	0	0	0	0	0	-64	192
$C_{\text{доп.}}$	1	0	0	0	0	0	1	1		131?
$C_{\text{пр.}}$	1	1	1	1	1	1	0	1	-125?	

CF=1

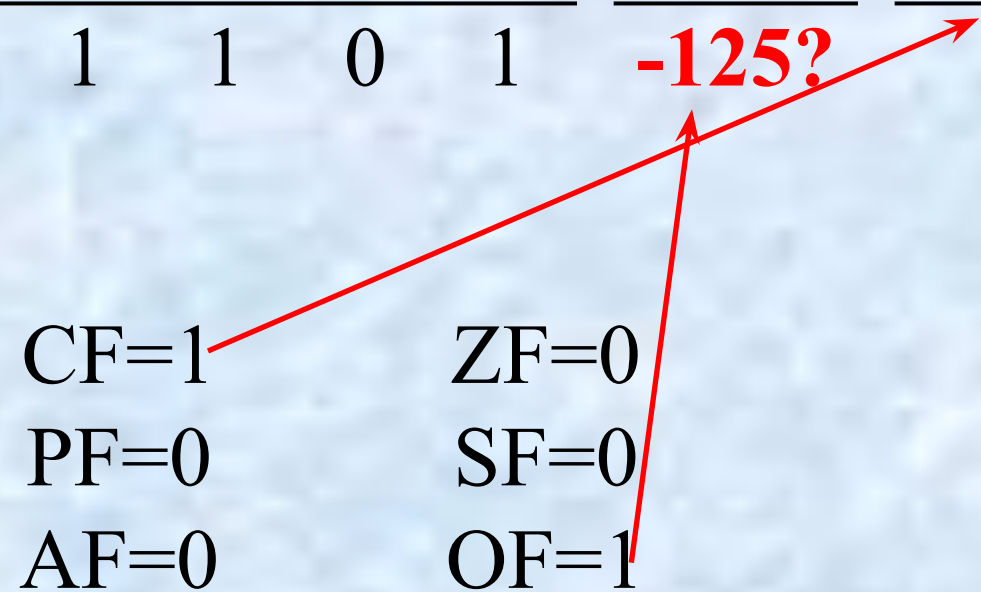
ZF=0

PF=0

SF=0

AF=0

OF=1



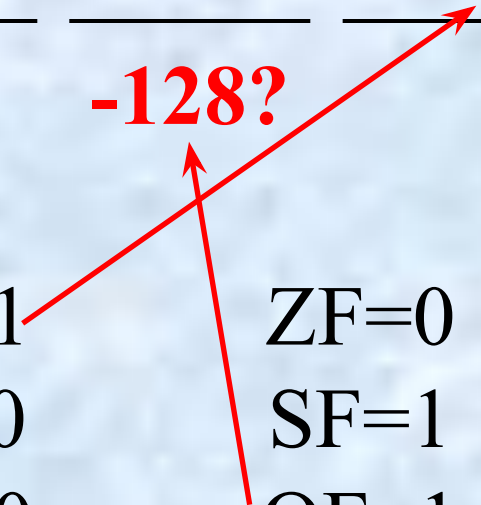
3. $A = 77, B = 51.$

$A > 0, B < 0.$

Интерпретации

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

- CF=1
- PF=0
- AF=0
- ZF=0
- SF=1
- OF=1



$A < 0, B > 0.$

Интерпретации

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

$$\begin{array}{r}
 1101 \\
 \times 1011 \\
 \hline
 1101 \\
 1101 \\
 1101 \\
 \hline
 1101 \\
 \hline
 10001111 \quad (143)_{10}
 \end{array}$$

Второй способ:

$$\begin{array}{r}
 1101 \\
 \times 1011 \\
 \hline
 1101 \\
 1101 \\
 1101 \\
 \hline
 1101 \\
 \hline
 10001111
 \end{array}$$

Особенности операций умножения целых чисел:

- каждое частное произведение либо совпадает с формируемым, либо равно нулю, 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

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

3. Каждый шаг умножения должен проводится,

сопровождать-ся сдвигом суммы частных произведений (СЧП) от-носительно

неподвижного множимого, или наобо-рот: 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 младших разрядов, время анализа операции в устройстве умножения должно привязываться к младшему разряду множителя, привязывается к (числу вправо), который считает количество разрядов

7. Так как возможно умножение, начиная от стар-ших и с младших разрядов (то есть сдвигается мно-жимое 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

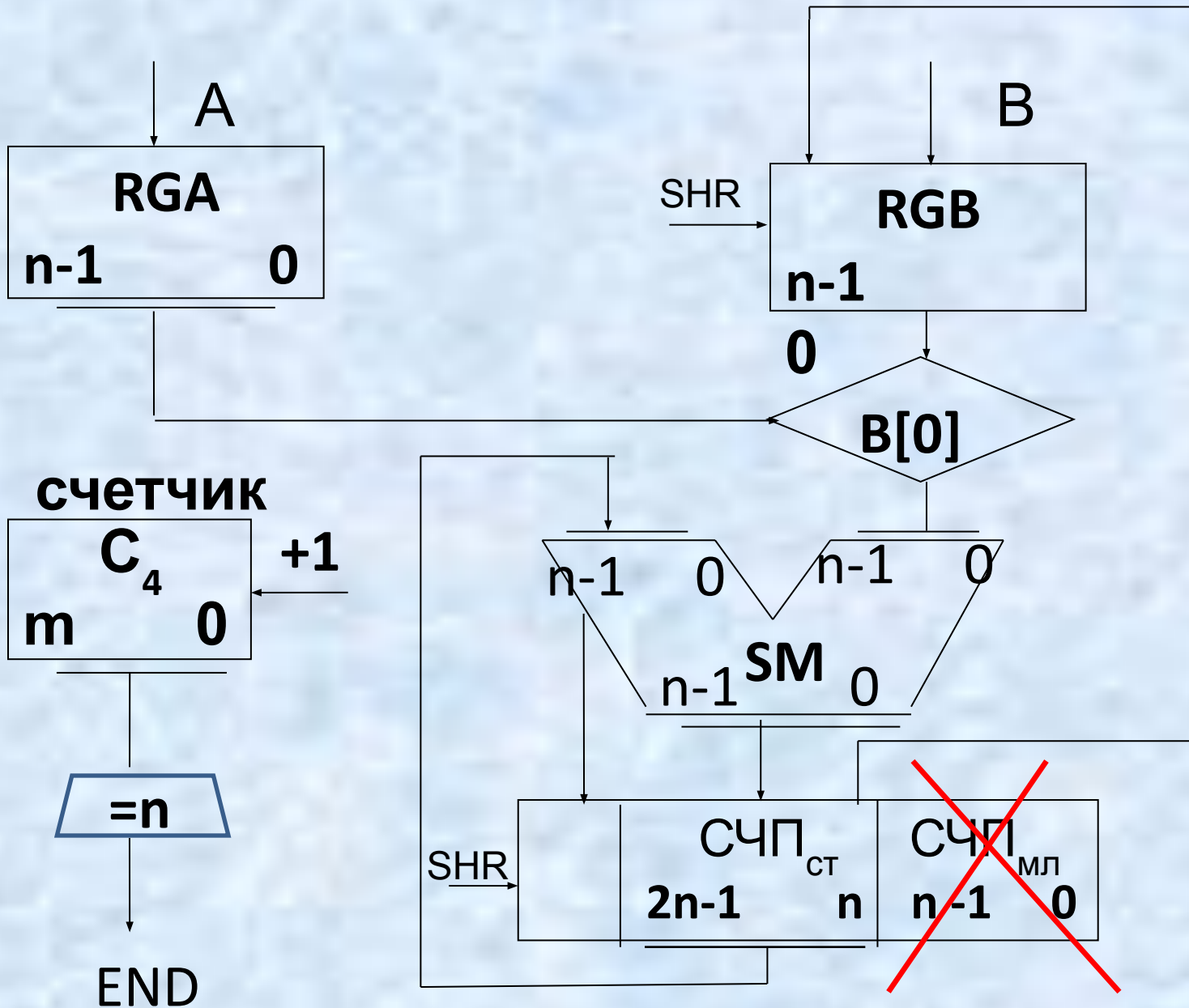
Анализ схем

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 $2n$ -разрядный регистр.
4. Для схем умножения, начиная от старших разрядов со сдвигом СЧП влево, требуется $2n$ -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.



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.

$$C = [A_{\text{пр.}}] \times [B_{\text{пр.}}]$$

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

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

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

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

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

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

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**

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

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

Пример. В разрядной сетке длиной в байт (один разряд знаковый и семь – цифровых) выполнить операцию умножения заданных чисел A и B 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.

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

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

$$[A_{\text{доп.}}] = 1.0001$$

а) $A < 0, B > 0$:

№	Действия	СЧП (старшие разряды)					СЧП (младшие разряды)					Пояснения
0	СЧП	0	0	0	0	0	0	1	1	0	1	Обнуление старших разрядов СЧП.
1	[A] _{дон.}	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] _{дон.}	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	$[A]_{\text{доп.}}$	1	0	0	0	1						Сложение СЧП с множимым.
	СЧП	0	0	1	1	1	1	0	1	0	1	
	СЧП \rightarrow	1	0	0	1	1	1	1	0	1	0	
5	СЧП	1	1	0	0	1	1	1	1	0	1	

$$C_{\text{доп.}} = (1.100111101)_2$$

$$C_{\text{пр.}} = (1.011000011)_2 = -195$$

b) $A > 0, B < 0$:

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

$$[B_{\text{доп.}}] = 1.0011$$

№	Действие	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	$[A]_{\text{пр.}}$	0	1	1	1	1						Сложение СЧП с МНОЖИМЫМ. Сдвиг СЧП.
	СЧП	1	0	0	0	1	1	1	0	1	1	
	СЧП→	0	1	0	0	0	1	1	1	0	1	
Кo pp	$[-A]_{\text{доп.}}$	1	0	0	0	1						
	СЧП	1	1	0	0	1	1	1	1	0	1	

$$C_{\text{пр.}} = (1.011000011)_2 = -195$$

b) $A < 0, B < 0$:

$$[A_{\text{пр.}}] = 1.1111$$

$$[B_{\text{пр.}}] = 1.1101$$

$$[A_{\text{доп.}}] = 1.0001$$

$$[B_{\text{доп.}}] = 1.0011$$

№	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	$[A]_{\text{дон.}}$	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	$[A]_{\text{дон.}}$	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	$[A]_{\text{дон.}}$	1	0	0	0	1						Сложение СЧП с множимым. Сдвиг СЧП.
	СЧП	0	1	1	1	0	0	0	1	1	1	
	СЧП→	1	0	1	1	1	0	0	1	1	1	
<i>Кo</i>	$[-A]_{\text{пр.}}$	0	1	1	1	1						
<i>pp</i>	СЧП	0	0	1	1	0	0	0	0	1	1	

$$C_{\text{пр.}} = (0.011000011)_2 = 195$$

**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

Примечание.

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

Пример. $A = 11, B = 15, n = 5.$

$$\begin{aligned} [A_{\text{пр.}}] &= 0.1011 & [B_{\text{пр.}}] &= 0.1111 \\ [A_{\text{доп.}}] &= 1.0101 & [B_{\text{доп.}}] &= 1.0001 \end{aligned}$$

a) $A > 0, B > 0:$

№	Действие	СЧП (старшие разряды)					СЧП (младшие разряды)					Пояснения
0	СЧП	0	0	0	0	0	0	1	1	1	1	Обнуление старших разрядов СЧП.
1	$[-A]_{\text{дон.}}$	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.

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

$$C_{\text{пр.}} = (0.010100101)_2 = 165$$

a) $A < 0, B < 0$:

№	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	$[-A]_{\text{пр.}}$	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	$[A]_{\text{доп.}}$	1	0	1	0	1						младший разряд изменился с 1 на 0: сложение СЧП с
	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	$[-A]_{\text{пр.}}$	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.

$$C_{\text{пр.}} = (0.010100101)_2 = 165$$

Операция целочисленного деления и способы ее реализации в ЭВМ

Особенности двоичного деления

Пример. $A = 130, B = 10.$

$$A/B = C(R),$$

где A – делимое,

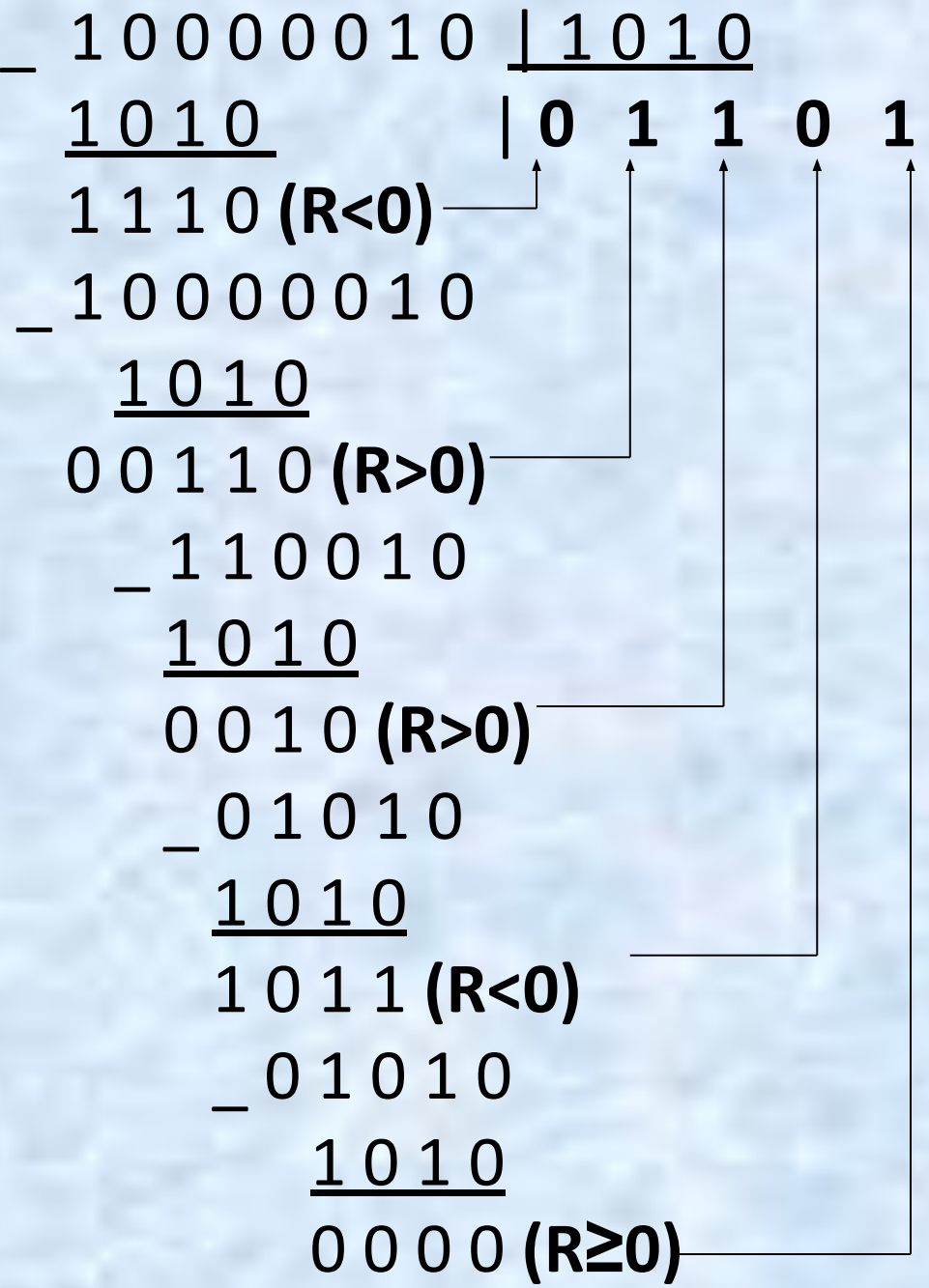
B – делитель,

C – частное,

R – остаток.

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

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



Из проделанного примера отчетливо проявляются следующие особенности

1) Процесс деления сводится к двойному делению: последовательному вычитанию делителя первоначально из делимого, а далее – из получаемых текущих остатков (под текущим остатком будем понимать промежуточный результат вычитания делителя из делимого на i -м очередном шаге деления). Делитель совмещается с 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

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 дели-теля вправо относительно неподвижного текущего остатка, а сдвиг текущего остатка влево относи-тельно неподвижного делителя. При этом дели-тель 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**

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:

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

$(R_i + V)$ – сложение с делителем;

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

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

б) По методу без восстановления остатка:

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

$2R_i$ – сдвиг отрицательного остатка влево;

$2R_i + V$ – 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

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 v , следовательно, n -разрядного частного условие корректности принимает вид:

$$A/v \leq 2^n - 1 \Rightarrow A/v < 2^n$$

$$A < v \cdot 2^n$$

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.**

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 B $R_0 < 0$	$\begin{array}{r} 1000 \\ -1101 \\ \hline 1011 \end{array}$	$\begin{array}{r} 1101 \\ 1101 \end{array}$	Делимое корректно

1	$\overleftarrow{R_0}$ B $R_1 > 0$	$\overset{\curvearrowright}{+} \begin{array}{r} 0111 \\ \underline{1101} \\ 0100 \end{array}$	$\begin{array}{r} 101 0 \\ \vdots \\ 101 1 \end{array}$	
2	$\overleftarrow{R_1}$ B $R_2 < 0$	$\overset{\curvearrowright}{-} \begin{array}{r} 1001 \\ \underline{1101} \\ 1100 \end{array}$	$\begin{array}{r} 01 10 \\ \vdots \\ 01 10 \end{array}$	
3	$\overleftarrow{R_2}$ B $R_3 > 0$	$+ \begin{array}{r} 1000 \\ \underline{1101} \\ 0101 \end{array}$	$\begin{array}{r} 1 100 \\ \vdots \\ 1 101 \end{array}$	
4	$\overleftarrow{R_3}$ B $R_4 < 0$	$- \begin{array}{r} 1011 \\ \underline{1101} \\ 1110 \end{array}$	$\begin{array}{r} 1010 \\ \\ 1010 \end{array}$	частное = $(10)_{10}$
5	B R	$\begin{array}{r} \underline{1101} \\ 1011 \end{array}$		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

3. По результату пробного вычитания на начальном (нулевом) шаге осуществляется проверка корректности 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 $R_0 \geq 0$	$\begin{array}{r} 1000 \\ -1000 \\ \hline 0000 \end{array}$	1101	Заем отсутствует. Некорректность деления.

Для рассматриваемых делимого и делителя $C=17$.

Аналогичная ситуация фиксации некорректности деления будет иметь место и при $B \neq 0$, в том числе и при $B \neq 0$.

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

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 вычисляется 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 +

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.

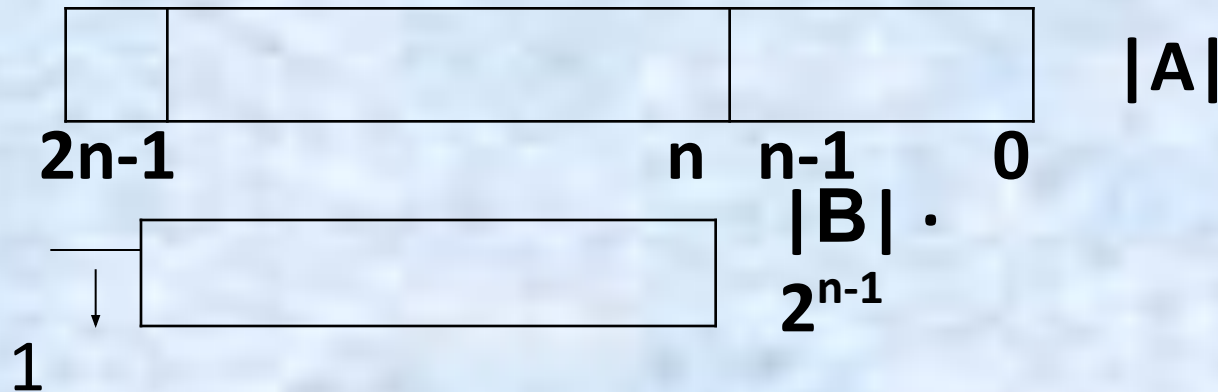
Обоснование метода проверки корректности деления

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

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

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

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

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

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 знака остатка с делителем).

4. Коррекция остатка 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) **$A > 0, B > 0.$**

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

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

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

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

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

Из полученных соотношений следует, что **провер-ка корректности при одинаковых знаках делимо-го и делителя** реализуется следующим образом:

а) выполняется предварительный сдвиг **делимого на один разряд влево**, **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 остат-ка не совпадают, то формируется старший разряд частного, интерпретируемый как знак, и осущест-вляется

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

провер-ка корректности деления при разных знаках де-лимого и делителя

выполняется в такой последо-вательности:

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 зна-ком делимого. Если они совпадают, то деление некорректно, и операция завершается прерыва-нием. Если знаки делимого и первого остатка не совпадают, то формируется старший разряд част-ного, интерпретируемый как знак, и осуществля-ется переход к основному циклу деления. Знак частного формируется по тем же правилам, что и любая

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

$$\mathbf{A = 139, B = 13.}$$

$$[A]_{\text{пр.}} = 0.010001011,$$

$$[B]_{\text{пр.}} = 0.1101,$$

a) $A < 0; B < 0.$

$$[A]_{\text{доп}} = 1.101110101;$$

$$[B]_{\text{доп}} = 1.0011.$$

N шага	Операнды	Действия		Комментарии
		Делимое/остаток (старшие разряды)	Делимое/остаток (младшие разряды)	
0	$[A]_{\text{доп}}$	1 1 0 1 1	1 0 1 0 1	Делимое
1	$\leftarrow [A]_{\text{доп}}$ $[-B]_{\text{пр}}$ R_1	$+1 0 1 1 1$ $\underline{0 1 1 0 1}$ $0 0 1 0 0$	$0 1 0 1 0$ $0 1 0 1 0$	Сдвиг А. Вычитание В. Деление корректно.
2	$\leftarrow R_1$ $[B]_{\text{доп}}$ R_2	$+0 1 0 0 0$ $\underline{1 0 0 1 1}$ $1 1 0 1 1$	$1 0 1 0 0$ $1 0 1 0 1$	Сдвиг остатка влево. Сложение с В. Формирование цифры частного.
3	$\leftarrow R_2$ $[-B]_{\text{пр}}$ R_3	$+1 0 1 1 1$ $\underline{0 1 1 0 1}$ $0 0 1 0 0$	$0 1 0 1 0$ $0 1 0 1 0$	Сдвиг остатка влево. Вычитание делителя. Формирование цифры частного.
4	$\leftarrow R_3$ $[B]_{\text{доп}}$ R_4	$+0 1 0 0 0$ $\underline{1 0 0 1 1}$ $1 1 0 1 1$	$1 0 1 0 0$ $1 0 1 0 1$	Сдвиг остатка влево. Сложение с В. Формирование цифры частного.
5	$\leftarrow R_4$ $[-B]_{\text{пр}}$ R_5	$+1 0 1 1 1$ $\underline{0 1 1 0 1}$ $0 0 1 0 0$	$0 1 0 1 0$ $0 1 0 1 0$	Сдвиг остатка влево. Вычитание делителя. Формирование цифры частного.
6	$[B]_{\text{доп}}$ 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:

$$[C]_{\text{пр}} = (0.1010)_2 = (+10)_{10}$$

и отрицательный остаток

$$[B]_{\text{доп}} = (1.0111)_2 = (-9)_{10},$$

которые соответствуют значениям:

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

a) $A < 0, B > 0.$

N шага	Операнды	Действия		Комментарии
		Делимое/остаток (старшие разряды)	Делимое/остаток (младшие разряды)	
0	$[A]_{\text{доп}}$	1 1 0 1 1	1 0 1 0 1	Делимое
1	$[B]_{\text{пр}}$	0 0 0 0 0	0 1 1 0 1	Сложение с делителем, выровненным по
	$[-B]_{\text{пр}}$ R_1	1 1 1 0 0 + 1 1 0 0 0 <u>0 1 1 0 1</u> 0 0 1 0 1 sign R_1 = sign B	0 0 0 1 0 0 0 1 0 0 0 0 1 0 1	младшим разрядам. Сдвиг остатка влево. Сложение с делителем, выровненным по старшим разрядам. Деление корректно.
2	$\leftarrow R_1$ $[-B]_{\text{доп}}$ R_2	+ 0 1 0 1 0 <u>1 0 0 1 1</u> 1 1 1 0 1 sign R_2 \neq sign B	0 1 0 1 0 0 1 0 1 0	Сдвиг остатка влево. Вычитание делителя. Формирование цифры частного.
3	$\leftarrow R_2$ $[B]_{\text{пр}}$ R_3	+ 1 1 0 1 0 <u>0 1 1 0 1</u> 0 0 1 1 1 sign R_3 = sign B	1 0 1 0 0 1 0 1 0 0	Сдвиг остатка влево. Сложение с B . Формирование цифры частного.
4	$\leftarrow R_3$ $[-B]_{\text{доп}}$ R_4	+ 0 1 1 1 1 <u>1 0 0 1 1</u> 0 0 0 1 0 sign R_4 = sign B	0 1 0 1 0 0 1 0 1 1	Сдвиг остатка влево. Вычитание делителя. Формирование цифры частного.
5	$\leftarrow R_4$ $[-B]_{\text{доп}}$ R_5	+ 0 0 1 0 0 <u>1 0 0 1 1</u> 1 0 1 1 1 sign R_5 \neq sign B	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: $[C]_{\text{доп}} = (1.0110)_2 = (-10)_{10}$ и отрицательный остаток: $[B]_{\text{доп}} = (1.0111)_2 = (-9)_{10}$,

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.

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

$$[B]_{\text{пр}} = (-14)_{10} = (1.1110)_2$$

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

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

$$B' = 0,0847 \cdot 10^3 \leftarrow P_B = P_A$$

MB

$$\begin{aligned} M_A &= 0,945 \\ M_B &= 0,0847 \\ \hline M_C = M_A + M_B &= 1,0297 \end{aligned}$$

$$P_{c=3}$$

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. Таким образом результат сложения $C = 0,10297 \cdot 10^4$.