Перевод чисел в формат с плавающей точкой

Возьмем несколько чисел и представим в формате с плавающей точкой. Несмотря, что для хранения мантиссы данного формата в стандарте IEE754 используется 3 байта, ограничимся одним

A = –13,13; B = 46,46; C = –6,868; D = –0,4444; E = 0,07777; F = 0,1999.

Эти числа можно также представить в формате с плавающей точкой

$A = -1{,}313\times10^{1}$; $B = 4{,}646\times10^{1}$; $C = -6{,}868\times10^{0}$; $D = -4{,}444\times10^{-1}$; $E = 7{,}777\times10^{-2}$; $F = 1{,}999\times10^{-1}$.

Отдельно совершим переход целой части и после запятой.

1. A = –13,13;

$13_{10} = 1101_2$;

Так уже 4 бита есть, то для оставшейся части, с учетом окружения, нужно определить 5 бит.

0,13 ×2 = **0**,26; 0,26 ×2 = **0**,52; 0,52 ×2 = **1**,04; 0,04 ×2 = **0**,08; 0,08 ×2 = **0**,16.

Поскольку при пятом умножении получился 0, то выделенные цифры формируют результат. $0{,}13_{10} = \mathbf{0010}_2$;

Для формирования результата надо перенести запятую на три бита вперед. Тогда степень двойки будет 3. К степени следует добавить 127. и представить в двоичном коде.

$130_{10} = 10000010_2$

Окончательно получим

1 10000010 101**0010**00…

Замечание. При компьютерном преобразовании будут, на самом деле, получены все 24 бита мантиссы.

2. B = 46,46;

$$\begin{array}{r|l} 46 & 8 \\ \cline{2-2} \underline{40} & 5 \\ 6 & \end{array}$$

$46_{10} = 56_8 = 101110_2$.

Еще нужно получить две цифры

0,46 ×2 = **0**,92; 0,92 ×2 = **1**,84; 0,84 ×2 = **1**,68.

С учетом ограничения получим

$0{,}46_{10} = \mathbf{011}_2 = \mathbf{10}_2$.

Для формирования результата надо перенести запятую на пять бит вперед. Тогда степень двойки будет 5. К степени следует добавить 127. и представить в двоичном коде.

$132_{10} = 10000100_2$

Окончательно получим

0 10000100 011101**10**00…

3. C = –6,868;

$6_{10} = 110_2$;

Так уже 3 бита есть, то для оставшейся части, с учетом окружения, нужно определить 6 бит.

0,868 ×2 = **1**,736; 0,736 ×2 = **1**,472; 0,472 ×2 = **0**,944; 0,944 ×2 = **1**,888; 0,888 ×2 = **1**,776; 0,776 ×2 = **1**,552.

Альтернативный вариант

6,944 7,552
Поскольку при пятом умножении получился 0, то выделенные цифры формируют результат. $0{,}868_{10}$ = $\mathbf{110111}_2$ = $\mathbf{11100}_2$.
Альтернативный вариант
0,868 ×8 = **6**,944; 0,944 ×8 = **7**,552.
$0{,}868_{10}$ = $\mathbf{67}_8$ = $\mathbf{110\ 111}_2$ = $\mathbf{11100}_2$
Для формирования результата надо перенести запятую на два бита вперед. Тогда степень двойки будет 2. К степени следует добавить 127. и представить в двоичном коде.
$129_{10} = 1000001_2$
Окончательно получим
1 10000001 10**11100**00…
1 10000010 101**0010**00…
0 10000100 01110**10**00…
1 10000001 10**1110**00…

4. D = –0,4444;
У числа D целой части нет. Поэтому необходимо получить 8 бит из десятичной части числа
0,4444 ×8 = **3**,5552; 0,5552 ×8 = **4**,4416; 0,4416 ×8 = **3**,5328; 0,5328 ×8 = **4**,2624.
Тогда
$0{,}4444_{10}$ = $\mathbf{3434}_8$ = $\mathbf{011\ 100\ 011\ 100}_2$.
Перенесем точку на два знака вправо. Степень получается –2. Добавим к степени 127.
$125_{10} = 01111101_2$.
Тогда окончательно мантисса после округления
$\mathbf{11100011100}_2$ = $\mathbf{11100100}_2$.
Окончательно получим
1 01111101 **1100100**00…

5. E = 0,07777; F = 0,1999.
У числа E целой части нет. Поэтому необходимо получить 8 бит из десятичной части числа
0,07777 ×8 = **0**,62216; 0,62216 ×8 = **4**,97728; 0,97728 ×8 = **7**,81824; 0,81824 ×8 = **6**,54592.
Тогда
$0{,}07777_{10}$ = $\mathbf{0476}_8$ = $\mathbf{000\ 100\ 111\ 110}_2$.
Для нормализации перенесем точку на 4 бита вправо. Степень получается –4. Добавим к степени 127.
$123_{10} = 01111011_2$.
Тогда окончательно мантисса после округления
$\mathbf{100111110}_2$ = $\mathbf{10011111}_2$.
Окончательно получим
0 01111011 **0011111**00…

6. F = 0,1999.
У числа F целой части нет. Поэтому необходимо получить 8 бит из десятичной части числа
0,1999 ×8 = **1**,5992; 0,5992 ×8 = **4**,7936; 0,97728 ×8 = **6**,3488; 0,3488 ×8 = **2**,7904.
Тогда
$0{,}07777_{10}$ = $\mathbf{1462}_8$ = $\mathbf{001\ 100\ 110\ 010}_2$.

Для нормализации перенесем точку на 3 бита вправо. Степень получается –3. Добавим к степени 127.
$124_{10} = 01111100_2$.
Тогда окончательно мантисса после округления
$\mathbf{1100110010}_2 = \mathbf{11001101}_2$.
Окончательно получим
0 01111100 **1001101**00…

Сложение в формате с плавающей точкой

В отличие от сложения чисел с фиксированной точкой, когда может происходить только преобразование числа из отрицательного в положительное или наоборот для выполнения нужной математической последовательности, то для чисел с плавающей точкой необходимо еще предварительно согласовать порядки, а после сложения проводить нормализацию результата.

Пример 1.

Выполнил операцию сложения для чисел A, B и C в разной последовательности. Представим числа в таблице, как это будет в задании. Значение мантиссы сокращены до шести бит с учетом округления.

	Знак	Порядок	Знак	Мантисса
A	0	0011	1	1101**01**
B	0	0101	0	10111**1**
C	0	0010	1	110**111**

Для сложения числа B и C следует уравнять порядки путем сдвига мантиссы на разницу порядков. Сдвигать можно только мантиссу с меньшим порядком. Разрядная сетка должна сохраняться. После сдвига следует правильно округлить результат.
Поэтому C в модифицированном коде
C = $11.110111_{пк}$ (0010 или 2^2) = $11.000110\cancel{111}_{пк}$ (0101 или 2^5) = $11.000111_{пк}$ (0101 или 2^5). К последнему биту был прибавлена 1 как следствие отброшенной единицы, которая была отброшена последней.
Перейдем в дополнительный код для выполнения операции сложения
C = $11.000111_{пк}$ (0101 или 2^5)= $11.111001_{дк}$ (0101 или 2^5). Сейчас уже можно провести сложение

	·	·	·		·	·	·	·	·		порядок
C	+	1	1	.	1	1	1	0	0	1	00.0101
B		0	0	.	1	0	1	1	1	1	00.0101
C+B	~~1~~	0	0	.	1	0	1	0	0	0	00.0101

Поскольку после битов знака положительного числа наблюдается 1, то число нормализовано, и можно переходить к следующей операции.
Число A имеет меньший на 2 порядок, поэтому следует произвести сдвиг его мантиссы на 2.
A = $11.110101_{пк}$ (0011 или 2^3) = $11.001101\cancel{01}_{пк}$ (0101 или 2^5) последним отброшенным является 0, то прибавлять 1 к последнему биту не нужно.
Перейдем в дополнительный код для выполнения операции сложения
A = $11.001101_{пк}$ (0101 или 2^5) = $11.110011_{дк}$ (0101 или 2^5). Сейчас уже можно провести сложение

	·	·	·		·						порядок
C+B	+	0	0	.	1	0	1	0	0	0	00.0101
A		1	1	.	1	1	0	0	1	1	00.0101
$C+B+A_1$	~~1~~	0	0	.	0	1	1	0	1	1	00.0101

Поскольку после битов знака положительного числа наблюдается 0, то число надо нормализовать.

$C+B+A_1 = 00.011011_{\text{пк}}$ (0101 или 2^5) = $00.110110_{\text{дк}}$ (0100 или 2^4).

Для перевода в стандарт IEE754 прибавим к порядку 127

					·	·	·	·	·			
4	+	0	0	.	0	0	0	0	0	1	0	0
127		0	0	.	0	1	1	1	1	1	1	1
		0	0	.	1	0	0	0	0	0	1	1

Число положительное

$C+B+A_1$ = 0 1000011 **101100**00…

Выполним сложение в другой последовательности. Сложим сначала A и C.

Степень числа A больше на один, чем C. Поэтому сдвинем мантиссу числа C на один.

C = $11.110111_{\text{пк}}$ (0010 или 2^2) = $11.011011\cancel{1}_{\text{пк}}$ (0011 или 2^3) = $11.011100_{\text{пк}}$ (0011 или 2^3) К последнему биту был прибавлена 1 как следствие отброшенной единицы, которая была отброшена последней.

C = $11.011100_{\text{пк}}$ (0011 или 2^3) = $11.100100_{\text{дк}}$ (0011 или 2^3).

Число отрицательное. Преобразуем его в дополнительный код, чтобы можно было выполнить сложение

A = $11.110101_{\text{пк}}$ (0011 или 2^3) = $11.001011_{\text{дк}}$ (0011 или 2^3)

	·	·	·								порядок
C	+	1	1	.	1	0	0	1	0	0	00.0101
A		1	1	.	0	0	1	0	1	1	00.0101
C+A	~~1~~	1	0	.	1	0	1	1	1	1	00.0101

Число получилось с переполнением, поскольку наблюдается 10 в модифицированном коде знака. Требуется нормализация. Для нормализации следует сделать сдвиг и, как следствие увеличить степень на 1

C+A = $10.101111_{\text{дк}}$(0011 или 2^3) = $11.010111_{\text{дк}}$(0100 или 2^4) .

Для сложения с числом B необходимо уровнять порядки, и, в частности, порядок промежуточной суммы сдвинуть до степени 5, которая есть у B.

C+A = $11.010111_{\text{дк}}$(0100 или 2^4) = $11.101011_{\text{дк}}$(0101 или 2^5).

	·	·	·		·	·	·	·	·		порядок
C+A	+	1	1	.	1	0	1	0	1	1	00.0101
B		0	0	.	1	0	1	1	1	1	00.0101
$C+B+A_2$	~~1~~	0	0	.	0	1	1	0	1	0	00.0101

Требуется нормализация

$C+B+A_2 = 00.011010_{\text{пк}}$ (0101 или 2^5) = $00.110100_{\text{дк}}$ (0100 или 2^4).

Первая сумма была

$C+B+A_1 = 00.011011_{\text{пк}}$ (0101 или 2^5) = $00.110110_{\text{дк}}$ (0100 или 2^4).

Результат имеет отличие, что присуще операции сложения в формате с плавающей точкой

Пример 2

Выполнил операцию сложения для чисел A, B и C в разной последовательности.

	Знак	Порядок	Знак	Мантисса

D	1	0010	1	**111001**
E	1	0100	0	**101000**
F	1	0011	0	**110011**

Для сложения D и E надо сравнять порядки и сдвинуть мантиссу числа с меньшим порядком, т.е. E.

Число E имеет меньший на 2 порядок, поэтому следует произвести сдвиг его мантиссы на 2.

E = 00.101000 (0101 или 2^{-4}) = 00.001010~~00~~ (0010 или 2^{-2}) последним отброшенным является 0, то прибавлять 1 к последнему биту не нужно.

Для проведения сложения, переведем число D дополнительный код

D = $11.111001_{пк}$ (0010 или 2^{-2}) = $11.000111_{пк}$ (0010 или 2^{-2})

											Порядок
						.	.	.			
E	+	0	0	.	0	0	1	0	1	0	$11.0010_{пк}$
D		1	1	.	0	0	0	1	1	1	$11.0010_{пк}$
D+E		1	1	.	0	1	0	0	0	1	$11.0010_{пк}$

Число нормализовано, так у результата отрицательного числа в дополнительном коде стоит 0 после модифицированного кода знака, поэтому нормализация не требуется.

Число F имеет меньший порядок чем итоговая сумма на один, поэтому сдвинем его мантиссу на 1.

F = 00.110011 (0011 или 2^{-3}) = 00.0011001~~1~~ (0010 или 2^{-2})) = 00.011010 (0010 или 2^{-2}).

Последним отброшенным является 1, то нужно прибавить 1 к последнему биту.

										порядок
				.	.					
D+E	1	1	.	0	1	0	0	0	1	$11.0010_{пк}$
F	0	0	.	0	1	1	0	1	0	$11.0010_{пк}$
$D+E+F_1$	1	1	.	1	0	1	0	1	1	$11.0010_{пк}$

Число не нормализовано, так у результата отрицательного числа в дополнительном коде стоит 1 после модифицированного кода знака, поэтому произведем сдвиг и скорректируем порядок.

$D+E+F_1$ = $11.101011_{дк}$ (0010 или 2^{-2}) = $11.010110_{дк}$ (0011 или 2^{-3})

Перейдем в прямой код

$D+E+F_1$ = $11.010110_{дк}$ (0011 или 2^{-3}) = $11.101010_{пк}$ (0011 или 2^{-3})

Прибавим к порядку 127. для этого перейдем в обратный код (можно и в дополнительном) значение порядка при восьмибитовом представлении значения).

f_m = $11.00000011_{пк}$ = $11.11111100_{ок}$

			.	.		.	.	.	.	.			
–3	+		1	1	.	1	1	1	1	1	1	0	0
127			0	0	.	0	1	1	1	1	1	1	1
		1	0	0	.	0	1	1	1	1	0	1	1

Так как это обратный код, то необходимо прибавить единицу перед модифицированным кодом знака к младшему биту результата.

+	0	0	.	0	1	1	1	1	0	1	1
	0	0	.	0	0	0	0	0	0	0	1
	0	0	.	0	1	1	1	1	1	0	0

$D+E+F_1$ = 1 01111100 **01011**00…

Произведем операцию сложения в другой последовательности.

Разница порядков E и F равна один, при этом меньше у E, поэтому сдвигаем его на один.

E = 00.101000 (0101 или 2^{-4}) = 00.010100~~0~~ (0010 или 2^{-3}) последним отброшенным является 0, то прибавлять 1 к последнему биту не нужно.

			·		·						порядок
E	+	0	0	.	0	1	0	1	0	0	$11.0011_{пк}$
F		0	0	.	1	1	0	0	1	1	$11.0011_{пк}$
F+E		0	1	.	0	0	0	1	0	1	$11.0011_{пк}$

В результате сложения получилось переполнение (биты знака стал 01), поэтому произведем нормализацию.
F+E = 01.000101 (0011 или 2^{-3}) = 00.100010~~1~~ (0010 или 2^{-2}) = 00.100011 (0010 или 2^{-2}).
(Пропуск нормализации считается ошибкой, даже если на последующем этапе потребуется производить сдвиг мантиссы как в нашем случае или в другую сторону)
В результате сравнялись порядки полученной суммы F+E с D поэтому можем переходить сразу к сложению.

								·	·		порядок
F+E	+	0	0	.	1	0	0	0	1	1	$11.0010_{пк}$
D		1	1	.	0	0	0	1	1	1	$11.0010_{пк}$
D+E+F_2		1	1	.	1	0	1	0	1	0	$11.0010_{пк}$

Число не нормализовано, так у результата отрицательного числа в дополнительном коде стоит 1 после модифицированного кода знака, поэтому произведем сдвиг и скорректируем порядок.
D+E+F_2 = $11.101010_{дк}$ (0010 или 2^{-2}) = $11.010100_{дк}$ (0010 или 2^{-3})
Перейдем в прямой код
D+E+F_2 = $11.010100_{дк}$ (0010 или 2^{-3}) = $11.101100_{пк}$ (0010 или 2^{-3})
Результаты не совпадают с первым сложением
D+E+F_1 = $11.101010_{пк}$ (0011 или 2^{-3})

Замечание. 1. Следует отметить, что в данных примерах не отражена операция определения величины и направления. Мы это определяли наглядно, тогда как микропроцессор производит вычитание одного порядка из другого и по знаку и значению определяет мантиссу, которую необходимо сдвинуть и насколько.
2. При выполнении задачи сразу выравнивать порядки у всех трех чисел неправильно, как по сути (в процессе сложения двух порядок может изменится вследствие необходимости проведения нормализации), так как из принципа работы ЭВМ: сложение отдельных чисел производится последовательно.

Оценка погрешности

Погрешность, от перехода в двоичную систему

Рассчитаем значения, которые соответствуют представлению исходных десятичных чисел.
$A_1 = 11.110101_{пк}$ (0011 или 2^3) = (–1) (1 + 0,5 + 0,125 + 0,03125) 2^2 = $–13,25_{10}$.
$B_1 = 00.101111_{пк}$ (0011 или 2^5) = (1 + 0,25 + 0,125 + 0,0625 + 0,03125) 2^5 = 47_{10}.
$C_1 = 11.110111_{пк}$ (0010 или 2^2) = (–1) (1 + 0,5 + 0,125 + 0,0625 + 0,03125) 2^2 = $–6,8750_{10}$.
$D_1 = 11.111001_{пк}$ (0010 или 2^{-2}) = (–1) (1 + 0,5 + 0,25 + 0,03125) 2^{-2} = $–0,4453_{10}$.
$E_1 = 00.101000_{пк}$ (0100 или 2^{-4}) = (1 + 0,25) 2^{-4} = $0,078125_{10}$.
$F_1 = 00.110011_{пк}$ (0011 или 2^{-3}) = (1 + 0,5 + 0,0625 + 0,03125) 2^{-3} = $0,1992_{10}$.

Определим относительную погрешность как модуль разницы между исходным значением и после преобразования отнесенную к исходному значению с умножением на 100%

В таблице приведены значения при 6-bit длине мантиссы, по расчетам как показано выше, так и при 8-bit представлении исходного преобразования.

	Изначальное значение	Для 6-bit мантиссы			Для 8-bit мантиссы		
		Результат обратного перевода	Мантисса	Относ. Погрешность, %	Результат обратного перевода	Мантисса	Относ. Погрешность, %
A	$-1{,}313\times10^{1}$	$-1{,}325\times10^{1}$	110101	0,9139	$-1{,}3125\times10^{1}$	11010010	0,0381
B	$4{,}646\times10^{1}$	$4{,}7\times10^{1}$	101111	1,1623	$4{,}65\times10^{1}$	10111010	0,0861
C	$-6{,}868\times10^{0}$	$-6{,}875\times10^{0}$	110111	0,1019	$-6{,}8750\times10^{0}$	11011100	0,1019
D	$-4{,}444\times10^{-1}$	$-4{,}453125\times10^{-1}$	111001	0,2053	$-4{,}453\times10^{-1}$	11100100	0,2053
E	$7{,}777\times10^{-2}$	$7{,}8125\times10^{-2}$	101000	0,4565	$7{,}76\times10^{-2}$	10011111	0,1714
F	$1{,}999\times10^{-1}$	$1{,}9921875\times10^{-1}$	110011	0,3408	$2{,}002\times10^{-1}$	11001101	0,1477

Произведем сложение исходных значений в десятичной системе

			Порядок	
A1	+	− 1 , 3 2 5	10^{1}	
C1		− 0 , 6 8 7 5	10^{1}	Для сложения сдвинули порядок на 1
	+	− 2 , 0 1 2 5	10^{1}	
B1		4 , 7 0 0 0	10^{1}	
		2 , 6 8 7 5	10^{1}	

$C+B+A_1 = 00.110110_{дк}$ (0100 или 2^4) = $(1 + 0{,}5 + 0{,}125+ 0{,}0625)\ 2^4 = 26_{10}$.

$C+B+A_2 = 00.110100_{дк}$ (0100 или 2^4) = $(1 + 0{,}5 + 0{,}125)\ 2^4 = 27_{10}$.

Погрешность

g_1 = 100%ï (2,6875 – 2,6)/2,6875ï = 3,2558%

g_2 = 100%ï (2,6875 – 2,7)/2,6875ï = 0,4651%

Произведем сложение исходных значений в десятичной системе

			Порядок	
E1	+	0 , 7 8 1 2 5	10^{-1}	Для сложения сдвинули порядок на 1
F1		1 , 9 9 2 1 8 7 5	10^{-1}	
	+	2 , 7 7 3 4 3 7 5	10^{-1}	
D1		− 4 , 4 5 3 1 2 5	10^{-1}	
		− 1 , 6 7 9 6 8 7 5	10^{-1}	

$D+E+F_1 = 11.101010_{пк}$ (0011 или 2^{-3}) = $(-1)(1 + 0{,}25 + 0{,}0625)\ 2^{-3} = -0{,}1640625_{10}$

$D+E+F_2 = 11.101100_{пк}$ (0010 или 2^{-3}) = $(-1)(1 + 0{,}25 + 0{,}125)\ 2^{-3} = -0{,}171875_{10}$

Результаты не совпадают с первым сложением

Погрешность

g_1 = 100%ï (1,6796875 – 1,640625)/1,6796875ï = 2,3256%

g_2 = 100%ï (1,6796875 – 1,71875)/1,6796875ï = 2,3256%

Как видим. операция сложения быстро набирает погрешность. Быстрее чем перевод из десятичной в двоичную при той же разрядной сетке.