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.

Отдельно совершим переход целой части и после запятой.

1.

$13_{10} = 1101_2$;

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

$0{,}13 \times 2 = \mathbf{0}{,}26$; $0{,}26 \times 2 = \mathbf{0}{,}52$; $0{,}52 \times 2 = \mathbf{1}{,}04$; $0{,}04 \times 2 = \mathbf{0}{,}08$; $0{,}08 \times 2 = \mathbf{0}{,}16$.

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 23 бита мантиссы.

2.

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

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

Еще нужно получить две цифры

$0{,}46 \times 2 = \mathbf{0}{,}92$; $0{,}92 \times 2 = \mathbf{1}{,}84$; $0{,}84 \times 2 = \mathbf{1}{,}68$.

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 01110**10**00…

3.

$6_{10} = 110_2$;

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

$0{,}868 \times 2 = \mathbf{1}{,}736$; $0{,}736 \times 2 = \mathbf{1}{,}472$; $0{,}472 \times 2 = \mathbf{0}{,}944$; $0{,}944 \times 2 = \mathbf{1}{,}888$; $0{,}888 \times 2 = \mathbf{1}{,}776$; $0{,}776 \times 2 = \mathbf{1}{,}552$.

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

$0{,}868 \times 8 = \mathbf{6}{,}944$; $0{,}944 \times 8 = \mathbf{7}{,}552$.

$0{,}868_{10} = \mathbf{67}_8 = \mathbf{110\ 111}_2 = \mathbf{11100}_2$

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…
4.
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 один знак вправо. Степень получается –1. Добавим к степени 127.
$126_{10} = 01111110_2$.
Тогда окончательно мантисса после округления
$\mathbf{11100011100}_2 = \mathbf{11100100}_2$.
Окончательно получим
1 01111110 **110010**00…

5.
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
$\mathbf{100111110}_2 = \mathbf{1010000}_2$.
Окончательно получим
0 01111011 **1010000**00…

6.
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 2 бита вправо. Степень получается –2. Добавим к степени 127.
$125_{10} = 01111101_2$.
Тогда окончательно мантисса после округления
$\mathbf{1100110010}_2 = \mathbf{11001101}_2$.
Окончательно получим
0 01111011 **100110**100…

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.
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	110**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	1	0	1	1	1	00.0101
C+B	~~1~~	0	0	.	1	1	0	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	1	0	0	0	0	00.0101
A		1	1	.	1	1	0	0	1	1	00.0101
C+B+A	~~1~~	0	0	.	1	0	0	0	1	1	00.0101

Поскольку после битов знака положительного числа наблюдается 1, то число нормализовано.
Для перевода в стандарт IEE754 прибавим к порядку 127

					·	·	·	·	·	·	·	
5	+	0	0	.	0	0	0	0	0	1	0	1
127		0	0	.	0	1	1	1	1	1	1	1
		0	0	.	1	0	0	0	0	1	0	0

Число положительное
C+B+A = 0 10000100 **000110**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.

$C = 11.110111_{пк}$ (0010 или 2^2) = $11.011011\cancel{1}_{пк}$ (0011 или 2^3) = $11.011100_{пк}$ (0011 или 2^3) К последнему биту был прибавлена 1 как следствие отброшенной единицы, которая была отброшена последней.
$C = 11.011100_{пк}$ (0011 или 2^3) = $11.100100_{дк}$ (0011 или 2^3).
Число отрицательное. Преобразуем его в дополнительный код, чтобы можно было выполнить сложение
$A = 11.110101_{пк}$ (0011 или 2^3) = $11.001011_{дк}$ (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
$C+A = 10.101111_{дк}$(0011 или 2^3) = $11.010111_{дк}$(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.
$C+A = 11.010111_{дк}$(0100 или 2^4) = $11.101011_{дк}$(0101 или 2^5).

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

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

Проведем сложение для другой тройки чисел

	Знак	Порядок	Знак	Мантисса
D	1	0001	1	**111001**
E	1	0100	0	**101000**
F	1	0010	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 3 порядок, поэтому следует произвести сдвиг его мантиссы на 3.
$E = 00.101000$ (0101 или 2^{-4}) = $00.000101\cancel{000}$ (0001 или 2^{-1}) последним отброшенным является 0, то прибавлять 1 к последнему биту не нужно.
Для проведения сложения, переведем число D дополнительный код

$D = 11.111001_{пк}$ (0001 или 2^{-1}) = $11.000111_{пк}$ (0001 или 2^{-1})

								·	·		порядок
E	+	0	0	.	0	0	0	1	0	1	$11.0001_{пк}$
D		1	1	.	0	0	0	1	1	1	$11.0001_{пк}$
D+E		1	1	.	0	0	1	1	0	0	$11.0001_{пк}$

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.
$F = 00.110011$ (0010 или 2^{-2}) = $00.0011001\cancel{1}$ (0001 или 2^{-1})) = 00.011010 (0001 или 2^{-1}) последним отброшенным является 1, то нужно прибавить 1 к последнему биту.

				.	.	.					порядок
D+E	1	1	.	0	0	1	1	0	0		$11.0001_{пк}$
F	0	0	.	0	1	1	0	1	0		$11.0001_{пк}$
D+E+F	1	1	.	1	0	0	1	1	0		$11.0001_{пк}$

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

$F = 11.100110_{дк}$ (0001 или 2^{-1}) = $11.001100_{дк}$ (0010 или 2^{-2})

Перейдем в прямой код

$F = 11.001100_{дк}$ (0010 или 2^{-2}) = $11.110100_{пк}$ (0010 или 2^{-2})

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

$f_m = 11.00000010_{пк} = 11.11111101_{ок}$

					.	.	.	.	.	.	.	
–2	+		1	1	.	1	1	1	1	1	0	1
127			0	0	.	0	1	1	1	1	1	1
		1	0	0	.	0	1	1	1	1	0	0

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

–2	+	0	0	.	0	1	1	1	1	0	0
127		0	0	.	0	0	0	0	0	0	1
		0	0	.	0	1	1	1	1	0	1

D+E+F = 1 01111101 **10100**00…

	Знак	Порядок	Знак	Мантисса
A	0	0011	1	1101**01**
B	0	0101	0	10111**1**
D	1	0001	1	**111001**
E	1	0100	0	**101000**

Произведем умножение числа A на D, представленных в формате с плавающей точкой. Для процесса умножения существую разные подходы. Используем алгоритм со сдвигом промежуточного результата. Знак результата можно определить сразу через операцию XOR знаков двух чисел. Поскольку оба числа отрицательные то произведение их будет положительное. По этой причине можно производить отдельное умножение мантисс без учета знака. Отдельно производится операция сложения порядков. Результат сложения может быть подкорректирован, если по результату перемножения мантисс потребуется нормализация.

	00.110101		Модуль мантиссы числа A
	00.111001		Модуль мантиссы числа D
+	00.000000		Заполнение регистра хранения промежуточного результата
	00.110101		Результат умножения последнего бита числа D на число A
	00.110101		Промежуточная сумма
+	00.011010	1	Результат сдвига влево на один
	00.000000		Результат умножения второго бита с конца числа D на число A
	00.011010		Промежуточная сумма

+	00.001101	0	Результат сдвига влево на один
	00.000000		Результат умножения третьего бита с конца числа D на число A
	00.001101		Промежуточная сумма
+	00.000110	1	Результат сдвига влево на один
	00.110101		Результат умножения четвертого бита с конца числа D на число A
	00.111011		Промежуточная сумма
+	00.011101	1	Результат сдвига влево на один
	00.110101		Результат умножения пятого бита с конца числа D на число A
	01.010010		Промежуточная сумма
	00.101001	0	Результат сдвига влево на один
	00.110101		Результат умножения последнего бита с конца числа D на число A
	01.011110		Окончание процесса умножения (два бита знака указываю на переполнение)
	00.101111	0	Результат нормализации (сдвига влево на один)

В буфере хранения последнего отброшенного бита 0, округление не изменит результат мантиссы. Однако степень следует увеличить на 1 из-за сдвига при нормализации
Произведем сложение порядков a_m и d_m чисел A и D для формирования результата. Порядок числа A три, а D минус два. Представим числа в обратном коде, при восьми битах значения числа
$d_m = 11.00000010_{пк} = 11.11111101_{ок}$

$$
\begin{array}{lcccccccccccc}
 & & \cdot & \cdot & & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \\
d_m & + & 1 & 1 & . & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\
a_m & & 0 & 0 & . & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
\hline
 & 1 & 0 & 0 & . & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
$$

По правилам обратного кода первую единицу, которая вышла за два бита знака следует прибавить к младшему биту

$$
\begin{array}{ccccccccccc}
+ & 0 & 0 & . & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & . & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\hline
 & 0 & 0 & . & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}
$$

К результату следует добавить еще единицу как результат нормализации мантиссы

$$
\begin{array}{cccccccccccc}
 & & & & & & & & & & \cdot & \\
+ & 0 & 0 & . & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
 & 0 & 0 & . & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\hline
 & 0 & 0 & . & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{array}
$$

И для представления в формате IEEE754 прибавим к последней сумме 127.

$$
\begin{array}{lcccccccccccc}
 & & & & & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \\
127 & + & 0 & 0 & . & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
2 & & 0 & 0 & . & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
\hline
 & & 0 & 0 & . & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}
$$

Тогда результат (первый бит указывает на знак, зеленные цифры это порядок + 127, первые 5 черных бит это мантисса произведения без первой единицы)
A×D = 0 10000001 01111000…