1. (6 points) Consider the following snippet of code:

```
1    x = single(-214.6);
2    y = -214.6;
3    format hex
4    x, y
```

Show the output that MATLAB will produce in the Command Window after we execute this code. Note, you can assume that MATLAB rounds up.

Let's begin by finding the exact, infinite repeating binary expansion of $x = -214.6 = \frac{-2146}{10} = \frac{-107}{5}$

Note: we know x requires an infinite repeating binary expansion because the denominator cannot be written as 2^j for any $j \in \mathbb{Z}$, even after transforming this fractional representation into "simplest terms" by canceling out all common factors so that numerator & denominator are relatively prime.

With this in mind, let's transform the leading part and fractional part into binary separately:

$$x = -\underbrace{214}_{\text{leading part}}.\underbrace{6}_{\text{fractional part}}$$

$$= -1 \cdot (214 + 0.6)$$

SAMPLE Exam 2, Problem 1 Solution continued ...)

Note that $(214)_{10} = 128 + 64 + 16 + 4 + 2$

$$= 2^7 + 2^6 + 2^4 + 2^2 + 2^1$$

$$= (11010110)_2$$

To find the binary representation of our fractional part, consider our algorithm described in the extra credit problem of this exam

Step 1: $2 \cdot 0.6 = 1.2 = \boxed{0.2} + 1$

Step 2: $2 \cdot \boxed{0.2} = 0.4 = \boxed{0.4} + 0$

Step 3: $2 \cdot \boxed{0.4} = 0.8 = \boxed{0.8} + 0$

E11, P1 continued)

Step 4: $2 \cdot \boxed{0.8} = 1.6 = 0.6 + 1$

↑ we now can repeat steps 1–4 at infinitum ...

$$\Rightarrow (0.6)_{10} = (0.1001\,1001\,1001\,1001\ldots)_2$$

$$= (0.\overline{1001})_2$$

Matric: We will use the geometric series test

Side Note: Let's confirm this binary representation works

$$0.\overline{1001} = \frac{1}{2^1} + \frac{1}{2^4} + \frac{1}{2^5} + \frac{1}{2^8} + \frac{1}{2^9} + \frac{1}{2^{12}} + \cdots$$

$$= \sum_{k=0}^{\infty} \frac{1}{2^{4k+1}} + \frac{1}{2^{4k+4}}$$

Geometric Series Thm

Recall: if $a, r \in \mathbb{R}$ and $|r| < 1$, then

$$\sum_{k=0}^{\infty} a \cdot r^k = \frac{a}{1-r}$$

$$= \sum_{k=0}^{\infty} \frac{1}{2 \cdot 2^{4k}} + \frac{1}{2^4 \cdot 2^{4k}}$$

E11, P1 continued ...)

$$= \sum_{k=0}^{\infty} \frac{1}{2} \cdot \left[\frac{1}{2^4}\right]^k + \frac{1}{2^4} \cdot \left[\frac{1}{2^4}\right]^k$$

$$= \sum_{k=0}^{\infty} \underbrace{\frac{1}{2}\left[\frac{1}{16}\right]^k}_{a=1/2,\ r=1/16<1} + \underbrace{\sum_{k=0}^{\infty} \frac{1}{16} \cdot \left[\frac{1}{16}\right]^k}_{a=\frac{1}{16},\ r=\frac{1}{16}<1}$$

$$= \frac{1/2}{1 - 1/16} + \frac{1/16}{1 - 1/16}$$

$$= \frac{1}{1 - 1/16} \cdot \left(\frac{1}{2} + \frac{1}{16}\right)$$

$$= \frac{9/16}{15/16}$$

$$= \frac{9}{16} \div \frac{15}{16}$$

$$= \frac{9}{16} \cdot \frac{16}{15} = \frac{9}{15} = \frac{3}{5} = 0.6 \checkmark$$

Ell, Pl continued ...)

Using this work, we conclude that

$$X = -214.6$$

$$= -1 \cdot (214.6)_{10}$$

$$= -1 \cdot (11010110.1001\,1001\,1001\,1001\,1001\ldots)_2$$

$$= -1 \cdot (11010110.\overline{1001})_2$$

Shift 7 places to left

$$= -1 \cdot (11010110.\overline{1001} \times \underbrace{2^{-7} \times 2^{7}}_{\text{equals } 1})$$

normalized hidden bit

$$= -1 \cdot 1.\underbrace{1010110\,\overline{1001}}_{\text{significand}} \times 2^{\boxed{7}}$$

$\boxed{7}$ ← exponent value

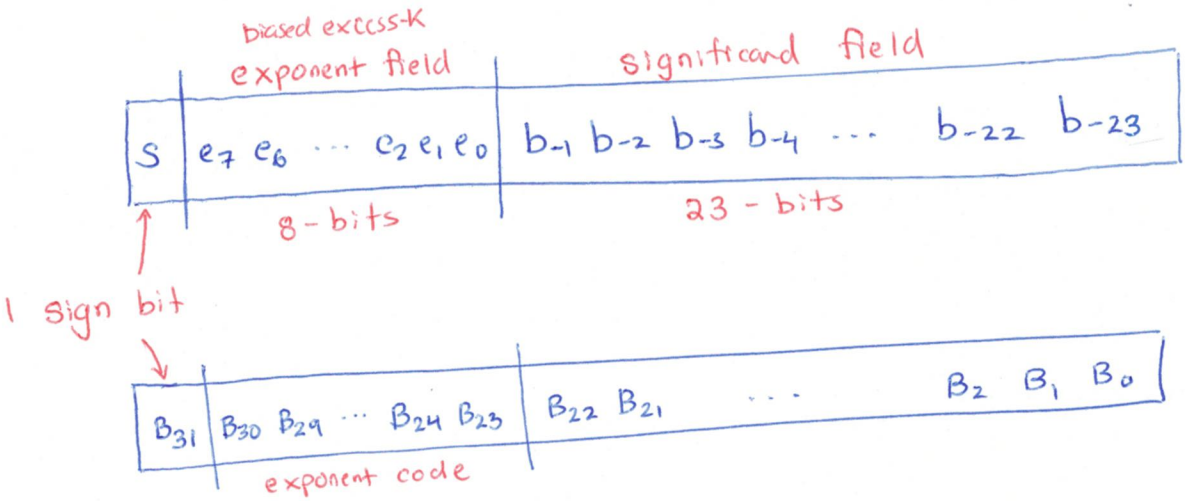
-1 ↑ sign

$\Rightarrow$
- sign bit = $s = 1$ (since $X < 0$ is a negative number)
- interpreted exponent value = $e = 7 = u - K$
- the significand will depend on the data type.

Eli, Pi continued ...)

Now, we recall the single and double precision data formats, as specified in the IEEE 754 format:

binary 32 format has 32 bits partitioned as follows



where the K-bias in this case is given by

$$K = 2^{8-1} - 1$$

$$= 2^{7} - 1$$

$$= 128 - 1 = 127$$

Ell, P1 continued...)

For single precision, we know our exponent code comes from

$$e = 7 = u - K$$

$$= uint8(e_7\, e_6\, e_5\, e_4\, e_3\, e_2\, e_1\, e_0) - 127$$

$$\Rightarrow \quad u = e + K$$

$$\Rightarrow \quad u = 7 + 127$$

$$\Rightarrow \quad u = 134 = 128 + 4 + 2 = 2^7 + 2^2 + 2^1$$

$$\Rightarrow \quad e_7\, e_6\, e_5\, e_4\, e_3\, e_2\, e_1\, e_0 = 1\,0\,0\,0\,0\,1\,1\,0$$

Finally, we want to figure out the significand field. To do so, let's write our normalized number to 29 bits of accuracy:

$$x = -1 \cdot (1.\underset{1}{1}\underset{2}{0}1\,0\,\underset{5}{1}1\,0\,1\,0\,0\,1\,1\,0\,0\,1\,1\,0\,0\,1\,1\,0\,0\,1\,\underset{24}{1}0\,0\,1\,1\,\underset{29}{0}\ldots) \times 2^7$$

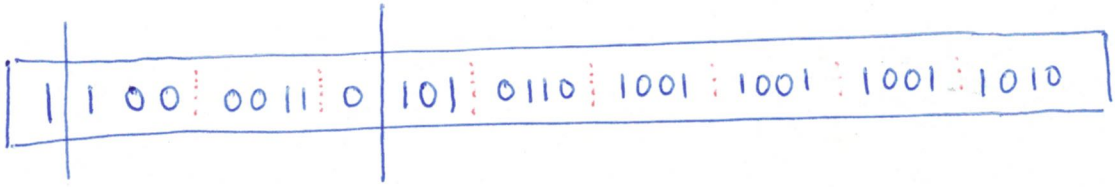
round up

loss of precision happens here and next bits are > 0

$$\Rightarrow x \approx -1 \cdot 1.10110100110011001100110 \boxed{10} \times 2^7$$

E11, P1 continued ...)

⇒ The single precision encoding of $x = -214.6$ is

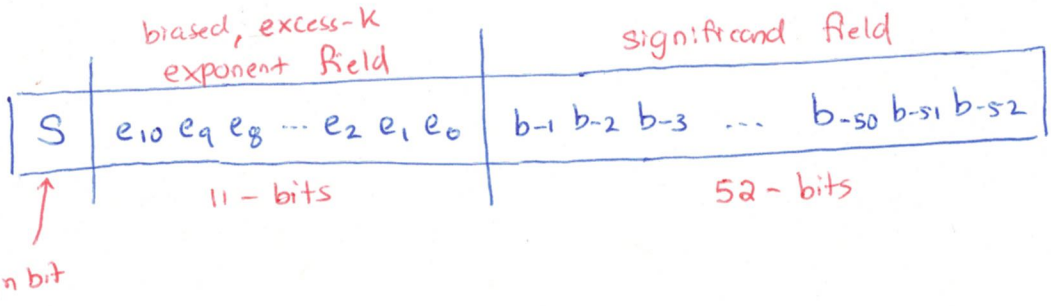


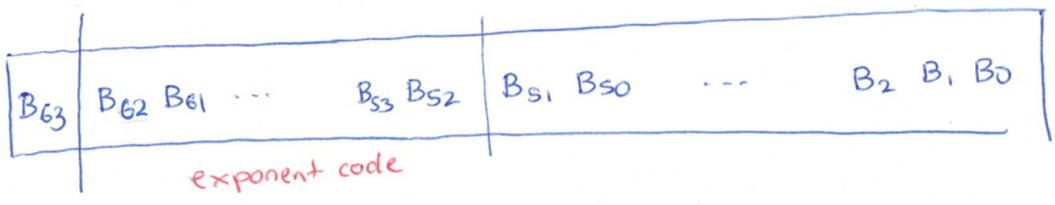
In hexadecimal format, this raw, uninterpreted 32-bit word is

c 3 5 6 9 9 9 a ← this is first output in command window from line 4 of our code snippet.

Let's now move on to analyze MATLAB's native double numerical data encoding that is triggered in line 2 of our code. To this end, recall

binary 64 format has 64-bits partitioned as follows





Ell, Pl continued...)

In binary 64, our K-bias is given by

$$K = 2^{11-1} - 1$$

$$= 2^{10} - 1$$

$$= 1024 - 1 = 1023$$

Then, we can create our exponent code using equation

$$e = 7 = u - K$$

$$\Rightarrow u = e + K$$

$$\Rightarrow u = 7 + 1023$$

$$\Rightarrow u = 1030 = 1024 + 4 + 2 = 2^{10} + 2^2 + 2^1$$

$$\Rightarrow e_{10}\, e_9 \cdots e_1\, e_0 = 1\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0$$

For the last step, let's write our normalized number x to 58 bits of accuracy:

E11, P1 continued...)

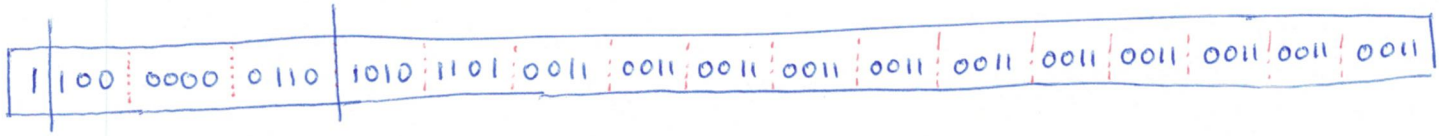
$$x = -1 \cdot (1.1010110100110011001100110011001100110011001100110011001100|1100\ldots) \times 2^{7}$$

loss of precision happens here

⇒ NO need to round up (round down)

$$\Rightarrow x \approx -1 \cdot (1.1010\,1101\,0011\,0011\,0011\,0011\,0011\,0011\,0011\,0011\,0011\,0011\,0011\,\boxed{1}) \times 2^{7}$$

⇒ the double precision encoding of $x = -214.6$ is



In hexadecimal format, this raw, uninterpreted 64-bit word is

c06ad33333333333 ← this is the second output in command window from line 4 of our code snippet

2. (6 points) Using the "chop" rule for rounding (literally chop off the extra bits), set

$$x_1 = \texttt{ufixed8}_{2(5,3)}\left(\frac{181}{16}\right) \qquad \text{and} \qquad x_2 = \texttt{ufixed8}_{2(4,4)}\left(\frac{181}{16}\right)$$

Then, calculate the absolute error $|x_1 - x_2|$ and explain why your answer makes sense?

We begin this problem by focusing on the numerator and denominator. We note that

Numerator: $(181)_{10} = 128 + 32 + 16 + 4 + 1$

$= 2^7 + 2^5 + 2^4 + 2^2 + 2^0$

$= (1011\ 0101)_2$

Denominator: $(16)_{10} = 2^4 = 10000$

$\Rightarrow \quad x = \frac{181}{16} = \frac{1011\ 0101}{10000} = 1011.0101$

$\Rightarrow \quad \text{ufixed8}_{2(4,4)}\left(\frac{181}{16}\right) = x_2 = 1011.0101$

loss of precision happens here

$\text{ufixed8}_{2(5,3)}\left(\frac{181}{16}\right) = x_1 = 01011.010|1$

↖ we chop based on the problem description

E11, E2 P2, continued...)

$\Rightarrow$ The absolute error is given by

$$|x_1 - x_2| = |01011.010 - 1011.0101|$$

$$= |01011.0100 - 01011.0101|$$

$$= |00000.0001|$$

$$= 0.0001$$

$$= \frac{1}{10000}$$

$$= \frac{1}{2^4}$$

$$= \boxed{\frac{1}{16}} = 0.0625$$

This makes sense since we chopped of the least-significant bit: $\frac{181}{16} = \frac{2^7 + 2^5 + 2^4 + 2^2}{2^4} + \boxed{\frac{2^0}{2^4}}$ ← chop happens here.

least-significant bit

3. (10 points) Consider the following raw, uninterpreted 8-bit binary word

$$B = 1110\ 1011$$

What decimal value does the B have if:

A. (2 points) we interpret this number as an unsigned binary integer?

$X = \text{uint8}(B)$

$= \text{uint8}(1110\ 1011)$

$= 2^7 + 2^6 + 2^5 + 2^3 + 2^1 + 2^0$

$= 128 + 64 + 32 + 8 + 2 + 1$

$= +235$

B. (2 points) we interpret this number as a signed integer in signed-magnitude representation?

Recall in signed-magnitude representation, the MSB represents sign

$X = (-1) \cdot (110\ 1011)$

$= (-1) \cdot (2^6 + 2^5 + 2^3 + 2^1 + 2^0)$

$= (-1) \cdot (64 + 32 + 8 + 2 + 1)$

$= -107$

C. (2 points) we interpret this number as a signed integer in twos complemment representation?

Recall in twos-complement, if $B = b_7 b_6 b_5 \cdots b_2 b_1 b_0$, then

$X = \text{int8}(B)$

$= -b_7 \cdot 2^7 + \sum_{K=0}^{6} b_K \cdot 2^K$

$= -2^7 + 2^6 + 2^5 + 2^3 + 2^1 + 2^0$

$= -128 + 64 + 32 + 8 + 2 + 1$

$= -21$

D. (2 points) we interpret this number as a $\texttt{ufixed8}_{2(3,5)}$ representation?

$X = \text{ufixed8}_{2(3,5)}(B)$

$= 111.01011$

$= 2^2 + 2^1 + 2^0 + \frac{1}{2^2} + \frac{1}{2^4} + \frac{1}{2^5}$

$= 4 + 2 + 1 + \frac{1}{4} + \frac{1}{16} + \frac{1}{32}$

$= \frac{128 + 64 + 32 + 8 + 2 + 1}{32} = \frac{235}{32}$

$= 7.34375$

E. (2 points) we interpret this number as a `binary8` representation?

$X = \text{binary8}(B)$

$= (-1)^S \cdot (1.1011) \times 2^e$

$= (-1)^1 \cdot (1.1011) \times 2^3$

$= -(1101.1)$

$= -(2^3 + 2^2 + 2^0 + \frac{1}{2^1})$

$= -(8 + 4 + 1 + 1/2)$

note: $e = U - K = 110 - 011 = 6 - 3 = 3$

$= -\frac{(16+8+2+1)}{2}$

$= -\frac{27}{2}$

$= -13.5$

4. (6 points) In this problem, we study the exact range for the $\texttt{ufixed8}_{2(\ell,f)}$ data format. For each of your answers, please write BOTH the decimal representation and the corresponding raw, uninterpreted binary word that encodes this representation in the stated format.

A. (2 points) What is the smallest positive number that can be encoded in $\texttt{ufixed8}_{2(6,2)}$?

The raw, uninterpreted 8-bit word corresponding to the smallest positive number that can be encoded in $\text{ufixed8}_{2(6,2)}$ is given as

$$B = 0000\ 0001$$

$$\Rightarrow X = \text{ufixed8}_{2(6,2)}(B)$$

$$= 000000.01$$

$$= \frac{1}{2^2} = \boxed{\frac{1}{4}}$$

B. (2 points) What is the largest number that can be encoded in $\texttt{ufixed8}_{2(6,2)}$?

The raw, uninterpreted 8-bit word corresponding to the largest positive number that can be encoded in $\text{ufixed8}_{2(6,2)}$ is given as

$$B = 1111\ 1111$$

$$\Rightarrow X = \text{ufixed8}_{2(6,2)}(B) = 111111.11 = \sum_{K=-2}^{5} 2^K$$

$$= 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 + \frac{1}{2^1} + \frac{1}{2^2} = 32 + 16 + 8 + 4 + 2 + 1 + \frac{1}{2} + \frac{1}{4}$$

$$= \frac{128+64+32+16+8+4+2+1}{4} = \frac{255}{4} = \frac{256-1}{4} = 64 - \frac{1}{4} = \boxed{63.75}$$

C. (2 points) Let $\ell, f \in \mathbb{Z}$ be nonnegative integers with $\ell + f = 8$. What is the range of $x \in \mathbb{Q}_I$ that can be stored exactly in the data format $\texttt{ufixed8}_{2(\ell,f)}$.

We can impose an order on the 8-bit strings:

8-bit string corresponding to min value: $0000\ 0000 = B_m$

8-bit string corresponding to max value: $1111\ 1111 = B_M$

Then all $B \in \mathbb{B}_8$ satisfy condition $B_m \leq B \leq B_M$ (where we introduce notation $\mathbb{B}_8 = \{$ 8-bit, binary words $\}$). Then the set of $x \in \mathbb{Q}_I$ that can be encoded in our $\text{ufixed8}_{2(\ell,f)}(B)$ must lie in range

$\text{ufixed8}_{2(\ell,f)}(B_m) = \boxed{0 \leq x \leq 2^\ell - \frac{1}{2^f}} = \text{ufixed8}_{2(\ell,f)}(B_M)$ and the space between each such x and the next largest value has length $\frac{1}{2^f}$.

5. (6 points) In this problem, we study the exact range for positive, normalized numbers in the `binary8` data format. For each of your answers, please write BOTH the decimal representation and the corresponding raw, uninterpreted binary word that encodes this representation in the stated format.

A. (3 points) What is the smallest, normalized positive number that can be encoded in `binary8`?

The raw, uninterpreted 8-bit word corresponding to the smallest, positive normalized $x \in \mathbb{Q}_I$ is given as

$$B_m = 0\ 001\ 0000$$

Using our binary8 data map, we see this bit string corresponds to value

$$x = \text{binary8}(B_m)$$

$$= \text{binary8}(0\ 001\ 0000)$$

$$= +1 \cdot (\boxed{1}.0000) \times 2^{-2}$$

(hidden bit (normalized))

$$= (0.01)_2 = 2^{-2} = \frac{+1}{4}$$

Side note: binary8 map

Recall in our binary8 map, we ha

	exponent field			significand			
s	e_2	e_1	e_0	b_{-1}	b_{-2}	b_{-3}	b_{-4}
1-bit sign	3-bits			4-bits			

$e = u - K$ with $K = 2^{3-1} - 1 =$

$= 001 - 011$

$= 1 - 3 = -2$

B. (3 points) What is the largest, normalized positive that can be encoded in `binary8`?

The raw, uninterpreted 8-bit word corresponding to the largest, positive normalized $x \in \mathbb{Q}_I$ is given as

$$B_M = 0\ 110\ 1111$$

Using our binary8 data map, we see this bit string corresponds to value

$$x = \text{binary8}(B_M)$$

$$= \text{binary8}(0\ 110\ 1111)$$

$$= +1 \cdot (\boxed{1}.1111) \times 2^{3}$$

(hidden bit (normalized))

$$= +1 \cdot (1111.1)_2$$

$$= 2^3 + 2^2 + 2^1 + 2^0 + 2^{-1} = 8 + 4 + 2 + 1 + \frac{1}{2} = 15.5$$

Side note: binary8 map

In this case, we see

$e = u - K$

$= 110 - 011$

$= 6 - 3$

$= +3$

6. (6 points) Consider the integers $p = 812$ and $q = 2^{24}$. Define the type I numbers

$$x = \frac{p}{q}$$

Assuming $x = \texttt{binar16}(B)$, find the 16–bit raw, uninterpreted binary word that encodes x in the `binary16` data format.

Let's begin this problem by finding the appropriate, exact finite binary expansion of this fraction. To this end, we note

Numerator $p = 812$

$= 512 + 256 + 32 + 8 + 4$

$= 2^9 + 2^8 + 2^5 + 2^3 + 2^2$

$= (1100101100)_2$

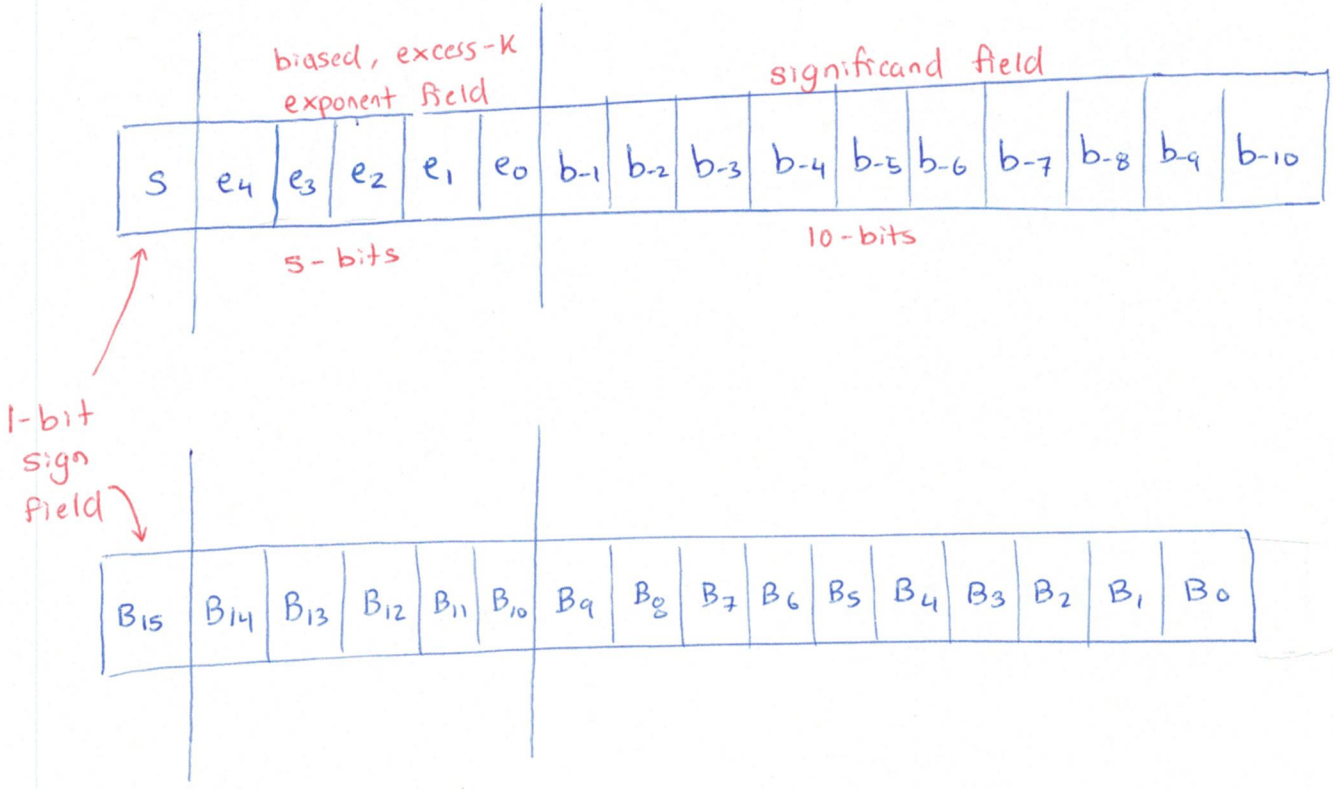
$\Rightarrow x = \frac{p}{q}$

$= \frac{1100101100}{2^{24}}$

$= 0.0000\ 0000\ 0000\ 0011\ 0010\ 1100$

Now, we want to convert this fixed-point representation to a binary16 encoding. To do so, let's recall our knowledge of the binary16 data format.

Recall that binary16 encodes $x \in \mathbb{Q}_I$ using a 16-bit binary word, partitioned into three unique fields



Recall, to interpret the raw 16-bit string as an $x \in \mathbb{Q}_I$, we want to determine if we will use a normalized encoding or not.

This determination is based on the exponent codes. More specifically, we know that each exponent code corresponds to a special interpretation.

Table : Exponent Code Map for Binary16

Note:
$K = 2^{5-1} - 1$
$= 2^4 - 1 = 15$

	Exponent code	Exponent value	
	If raw, uninterpreted 5-bit exponent word $e_4e_3e_2e_1e_0$ is	then the excess-K value of $e = u - K$ in decimal is	and the interpreted numeric value of x = binary16 is
0	0 0000	Special case of subnormals	$x = \pm 0.b_{-1}\cdots b_{-10} \times 2^{-14}$
1	0 0001	e = uint5(00001) - 15 = 1 - 15 = -14	$x = \pm 1.b_{-1}\cdots b_{-10} \times 2^{-14}$
2	0 0010	e = uint5(00010) - 15 = 2 - 15 = -13	$x = \pm 1.b_{-1}\cdots b_{-10} \times 2^{-13}$
⋮	⋮	⋮	⋮
29	1 1101	e = uint5(11101) - 15 = 29 - 15 = 14	$x = \pm 1.b_{-1}\cdots b_{-10} \times 2^{14}$
30	1 1110	e = uint5(11110) - 15 = 30 - 15 = 15	$x = \pm 1.b_{-1}\cdots b_{-10} \times 2^{15}$
31	1 1111	Special case of NaN or $\pm\infty$ (overflow)	$x = \pm\infty$ if $b_{-1} = \cdots = b_{-10} = 0$ OR x = NaN otherwise

In our case, we can write

$$x = \frac{812}{2^{24}} = 1.100101100 \times 2^{-15}$$

as a normalized, positive number.

However, we immediately recognize that the exponent value $e = -15$ is not in the range covered by binary 16 normalized format. But we can write

$$X = 0.1001\,01100 \times 2^{-14}$$

$$= +1 \cdot 0.1001\,01100 \times 2^{-14}$$

This is the special case of **subnormal numbers** and would be stored as follows: