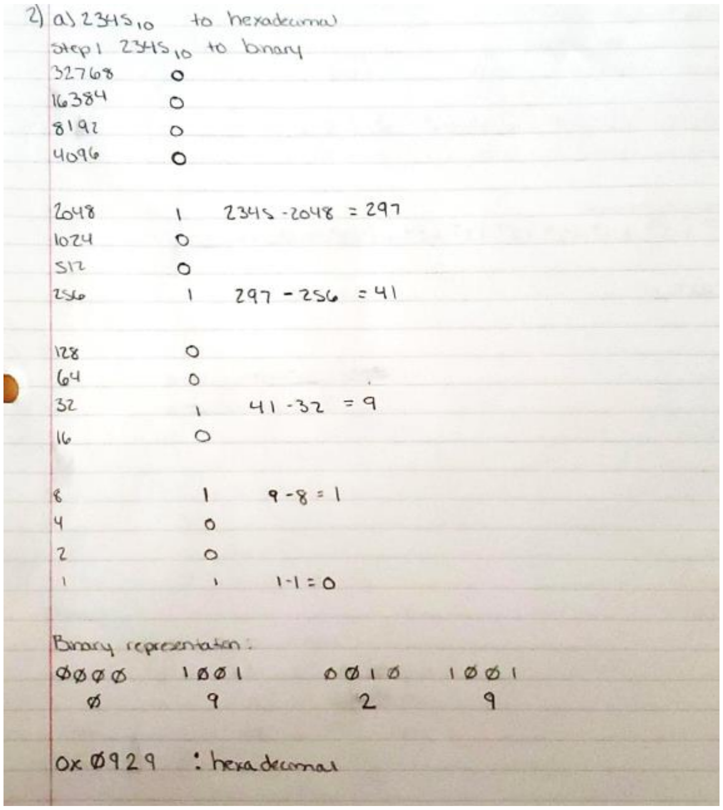
Homework 3

1) The two methods for converting unsigned whole numbers regardless of base are subtraction and the division remainder method.
    a. The subtraction method starts with finding the highest power of the base you are converting to that is less than or equal to the number we are converting. We take note of how many times that power goes into our original value, subtract the amount that base gave us and move on to the next lowest power of the base. If the next base gives us a number too large to be subtracted from the remaining number, then we note a zero for that power and move on to the next lowest power. We continue working down the powers until we subtract out to zero. Any remaining powers after we hit zero will be noted as a zero. At the end add up how many times each power goes into the original number times the base to that power. In other words, if 8^3 goes into your value one time we would have 8^3 *1 + 8^2 * (number of times 8^2 goes in the remainder) + 8^1 * (number of times 8^1 goes in the remainder) + 8^0 * (number of times 8^0 goes in the remainder) to get our new number in base 8.
    b. The division method works by looking at the remainder of dividing the original number by your new base. The first step is to divide your original number by your base. Take note of the remainder. Now take how many times you were able to divide your number with the new base and divide that value by the new base again. In other words, take your quotient from the previous step and make it your dividend and keep your divisor as the new base. Take note of the remainder. Repeat this method until you reach zero as your dividend. Your resulting number in the new base is the remainders read bottom to top.

2)
    a. The answer I got for part a is 0x0929
        i. I got this through the subtraction method. To the left of the ones in the calculations I show the subtraction I used to get this answer. The subtraction method gave me binary which I then converted into hexadecimal by looking at groups of 4bits individually and assigning them a value in hexadecimal.



2) a) 2345_{10} to hexadecimal

step 1 2345_{10} to binary

32768	0	
16384	0	
8192	0	
4096	0	
2048	1	2345 - 2048 = 297
1024	0	
512	0	
256	1	297 - 256 = 41
128	0	
64	0	
32	1	41 - 32 = 9
16	0	
8	1	9 - 8 = 1
4	0	
2	0	
1	1	1 - 1 = 0

Binary representation:

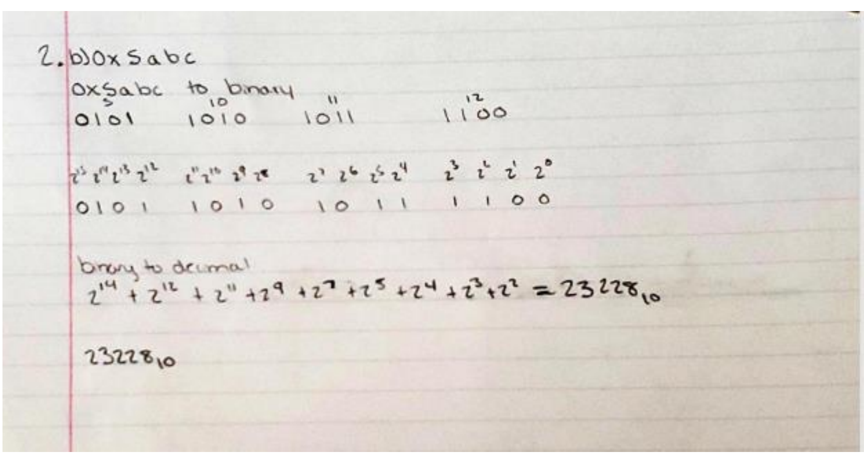
ØØØØ 1ØØ1 ØØ1Ø 1ØØ1

Ø 9 2 9

0xØ929 : hexadecimal

B. For part b I got 23228 in base 10.

I. To get this I split the hexadecimal into the 4 different values and for each of these numbers I changed them to their representation in binary. Each number or letter in hexadecimal can be represented with 4 bits in binary. Once I had those values converted to binary, I added up the powers of two that had a one in their spot. This converted the binary to decimal.



2. b) 0x5abc

0x5abc to binary

5	10	11	12
0101	1010	1011	1100

2^{15} 2^{14} 2^{13} 2^{12}	2^{11} 2^{10} 2^{9} 2^{8}	2^{7} 2^{6} 2^{5} 2^{4}	2^{3} 2^{2} 2^{1} 2^{0}
0 1 0 1	1 0 1 0	1 0 1 1	1 1 0 0

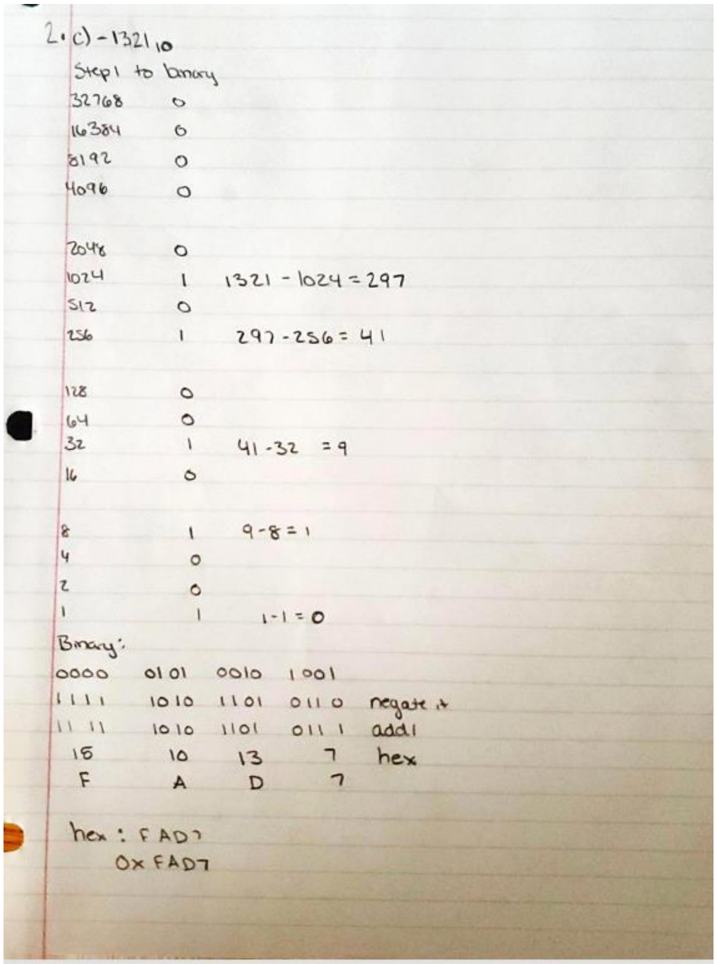
binary to decimal

$2^{14} + 2^{12} + 2^{11} + 2^{9} + 2^{7} + 2^{5} + 2^{4} + 2^{3} + 2^{2} = 23228_{10}$

23228_{10}

C. The answer I got for part c is 0xfad7

I. To get this answer I started by bringing 1321 to binary while taking note that it is a negative value. I used the subtraction method for this conversion. The next thing I did was negate the binary and add one to the result. This was the final string of binary I would convert to hexadecimal. Each group of four bits in binary were then assigned a hexadecimal value and my answer was reached.



2. c) -1321_{10}

Step 1 to binary

32768	0	
16384	0	
8192	0	
4096	0	
2048	0	
1024	1	1321 - 1024 = 297
512	0	
256	1	297 - 256 = 41
128	0	
64	0	
32	1	41 - 32 = 9
16	0	
8	1	9 - 8 = 1
4	0	
2	0	
1	1	1 - 1 = 0

Binary:

0000	0101	0010	1001	
1111	1010	1101	0110	negate &
1111	1010	1101	0111	add 1
15	10	13	7	hex
F	A	D	7	

hex : FAD7
0x FAD7

D. The answer I got for part d was –7628 in base ten.

I. I got this answer by first converting the hexadecimal to binary. I took note that this was a negative value because the most significant digit is a 1. The next step was to subtract 1 from the binary and then negate the binary. This gave me my final binary string that I would convert to decimal. I added up the powers of two which had a one in binary and got 7628 which is negative since the original binary string told us this was a negative value.

2. d) 0x e234

Binary: 14 2 3 4

1110 0010 0011 0100 negative b/c most significant bit 1

1110 0010 0011 0011 Subtract 1

0001 1101 1100 1100 negate

To decimal

$2^{15}\,2^{14}\,2^{13}\,2^{12}$	$2^{11}\,2^{10}\,2^{9}\,2^{8}$	$2^{7}\,2^{6}\,2^{5}\,2^{4}$	$2^{3}\,2^{2}\,2^{1}\,2^{0}$
0 0 0 1	1 1 0 1	1 1 0 0	1 1 0 0

$2^{12} + 2^{11} + 2^{10} + 2^{8} + 2^{7} + 2^{6} + 2^{3} + 2^{2} = -7628_{10}$

3)

a. Answer: 0xff91 flags: z = 0 c = 0 n = 1 v = 1

I. The first step to this problem was to convert the decimal values to binary. To do this I did the subtraction method. To the left of the values, I show the subtraction I did.

II. Since this is subtraction the next step was to negate the subtrahend (222). Once it was negated, I needed to add one to the binary number to get the final binary number we will add.

III. Now we add the binary from 111 to the changed value for 222 normally. This gave me the final binary string to convert to hexadecimal. I split the binary into four-bit sections and changed those sections into a hexadecimal value.

IV. As for the flags, z is zero because not all the bits are zero, c is zero because there was no carry of one on the add at the end, n is one because the most significant bit is 1, and v is one because we were subtracting values with different signs and the subtrahend (222) has the same sign as the difference have the same sign.

3) a) $111_{10} - 222_{10}$

To binary		111_{10}		222_{10}
32768		0		0
16384		0		0
8192		0		0
4096		0		0
2048		0		0
1024		0		0
512		0		0
256		0		0
128		0	222 - 128 = 94	1
64	111 - 64 = 47	1	94 - 64 = 30	1
32	47 - 32 = 15	1		0
16		0	30 - 16 = 14	1
8	15 - 8 = 7	1	14 - 8 = 6	1
4	7 - 4 = 3	1	6 - 4 = 2	1
2	3 - 2 = 1	1	2 - 2 = 0	1
1	1 - 1 = 0	1		0

negate subtrahend

original	0000	0000	1101	1110
negate	1111	1111	0010	0001
add 1	1111	1111	0010	0010

```
  0000   0000   0110   1111
+ 1111   1111   0010   0010
  ------------------------
  1111   1111   1001   0001     or 0xFF91
```

Flags $Z = \emptyset$ $C = \emptyset$ $N = 1$ $V = 1$

b. Answer: 0xb998 flags: z = 0 c = 0 n =1 v = 0

I. Step one was to change the hexadecimal values to binary. Then we add the binary like normal since they are both positive values. Finally, I converted the binary back to hexadecimal.

II. The z flag is 0 because not all the bits are zero. The c flag is 0 because there was no one to carry extra on the calculation. The n flag was set at one because the most significant bit on the result was 1 (negative). The v flag was set at zero because we are adding a positive and a negative value.

3) b) 0xb332 + 0x0666

b	3	3	2
1011	0011	0011	0010
Ø	6	6	6
0000	0110	0110	0110
1011	0011	0011	0010
0000	0110	0110	0110
1011	1001	1001	1000
11	9	9	8
B	9	9	8

0xB998

Z = 0 C = Ø V = Ø N = 1

c. Answer: 0x95b2 flags: z = 0 c = 0 n= 1 v = 1

I. The first step was to convert the hexadecimal value into binary values. Since we are subtracting, we need to take the subtrahends (0xabcd) binary negate it and add one. Once that is done the binary can be added like normal. Then this binary string can be converted back to hexadecimal. II. The z flag I set at zero because not all the bits were zero. I set the c flag at 0 because I did not have to carry one on the last bit. I set the n flag to 1 because the most significant bit is one. I set the v flag to 1 because we have a positive minus a negative and the subtrahend and the difference have the same sign (they are both negative). Another way to look at this is a positive minus a negative is giving us a negative value when it should give us a positive.

3) c. 0x417f - 0xabcd

0x417f to binary

0100 0001 0111 1111

0xabcd to binary

1010 1011 1100 1101 ~> subtrahend

0101 0100 0011 0010 negate subtrahend

0101 0100 0011 0011 add 1

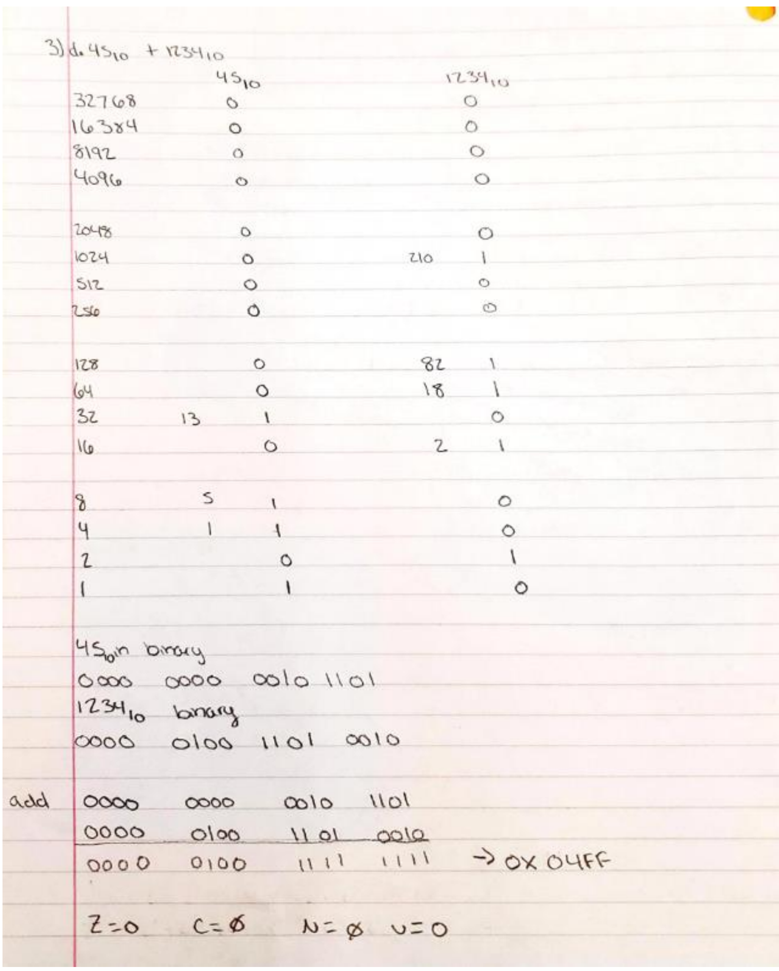
+ 0100	0001	0111	1111	
0101	0100	0011	0011	
1001	0101	1011	0010	0x 95B2

Z = Ø N = 1 C = Ø V = 1

d. Answer: 0x04ff flags: z = 0 c = 0 n = 0 v = 0

I. Step one was to take the decimal values and change them into binary. I used the subtraction method for this. The binary for 45 is on the left and the binary for 1234 is on the right. Since we are adding, I can then take the binary and add it like normal at this point. Once I got that string of binary, I could convert it to the final hexadecimal value.

II. I set the z flag to zero because not all the bits are zero. The c flag is set at zero because I did not have to carry one on the final bit. The n flag is set to zero because the most significant bit is a zero. The v flag is set to zero because we added two positives and we got a positive.



3) d. $45_{10} + 1234_{10}$

		45_{10}		1234_{10}
32768		0		0
16384		0		0
8192		0		0
4096		0		0
2048		0		0
1024		0	210	1
512		0		0
256		0		0
128		0	82	1
64		0	18	1
32	13	1		0
16		0	2	1
8	5	1		0
4	1	1		0
2		0		1
1		1		0

45_{10} in binary
0000 0000 0010 1101
1234_{10} binary
0000 0100 1101 0010

add
0000 0000 0010 1101
0000 0100 1101 0010
0000 0100 1111 1111 → 0X 04FF

Z=0 C=Ø N=Ø V=0

4) One reason 2's compliment is used more often than signed magnitude is because doing math in signed magnitude is very difficult to do. In signed magnitude there is a bit in front that says whether it is positive or negative. While this is an intuitive way to show the number as positive or negative this bit can make it difficult to do math with the binary.

5)

a. sign bit: 1

Exponent: 0111 1011

Mantissa: 11111001101011011

Total: 1 01111011 11111001101011011

I. For this problem we start by converting the integer and decimal parts of our number to binary separately. In this case we had no integer, and I used the subtraction method to do the decimal portion. Then we glue the two binary strings together with a binary point. The next step was to normalize the binary and from that we got the exponent value. When I normalized it, I got 2 ^ -4 so the exponent is 127 + -4 = 123. Then I put the exponent into binary. The signed bit is one since the number is negative. The exponent is 123 in binary. The mantissa version is everything after the decimal point from when we normalized the value up to 23 bits.

5) a. -0.1234567_{10}

Convert Integer portion : 0

Convert fraction portion

.1234567

2^{-1}	0	$.1234567 - 2^{-1} = -\#$
2^{-2}	0	$.1234567 - 2^{-2} = -\#$
2^{-3}	0	$.1234567 - 2^{-3} = -\#$
2^{-4}	1	$.1234567 - 2^{-4} = 0.0609567$
2^{-5}	1	$0.0609567 - 2^{-5} = 0.0297067$
2^{-6}	1	$0.0297067 - 2^{-6} = 0.0140817$
2^{-7}	1	$0.0140817 - 2^{-7} = 0.0062692$
2^{-8}	1	$0.0062692 - 2^{-8} = 0.0023629$
2^{-9}	1	$0.0023629 - 2^{-9} = 4.09 \times 10^{-4}$
2^{-10}	0	$-\#$
2^{-11}	0	$-\#$
2^{-12}	1	$4.09 \times 10^{-4} - 2^{-12} = 1.656 \times 10^{-4}$
2^{-13}	1	$1.656 \times 10^{-4} - 2^{-13} = 4.361 \times 10^{-5}$
2^{-14}	0	$-\#$
2^{-15}	1	$4.361 \times 10^{-5} - 2^{-15} = 1.309 \times 10^{-5}$
2^{-16}	0	$-\#$
2^{-17}	1	$1.309 \times 10^{-5} - 2^{-17} = 5.467 \times 10^{-6}$
2^{-18}	1	$5.467 \times 10^{-6} - 2^{-18} = 1.652 \times 10^{-6}$
2^{-19}	0	$-\#$
2^{-20}	1	$1.652 \times 10^{-6} - 2^{-20} = 6.987 \times 10^{-7}$
2^{-21}	1	$6.987 \times 10^{-7} - 2^{-21} = 2.2188 \times 10^{-7}$

Concatinate

0.000111111001101011011

Normalize

$1.11111001101011011 \times 2^{-4}$

Exponent

$127 + (-4) = 123$

123

128	0	
64	1	$123 - 64 = 59$
32	1	$59 - 32 = 27$
16	1	$27 - 16 = 11$
8	1	$11 - 8 = 3$
4	0	
2	1	$3 - 2 = 1$
1	1	$1 - 1 = 0$

Sign bit 1

Exponent 0111 1011

Mantissa 11111 00 11 01011 011

total

1 0111 1011 11111001101011011

b.

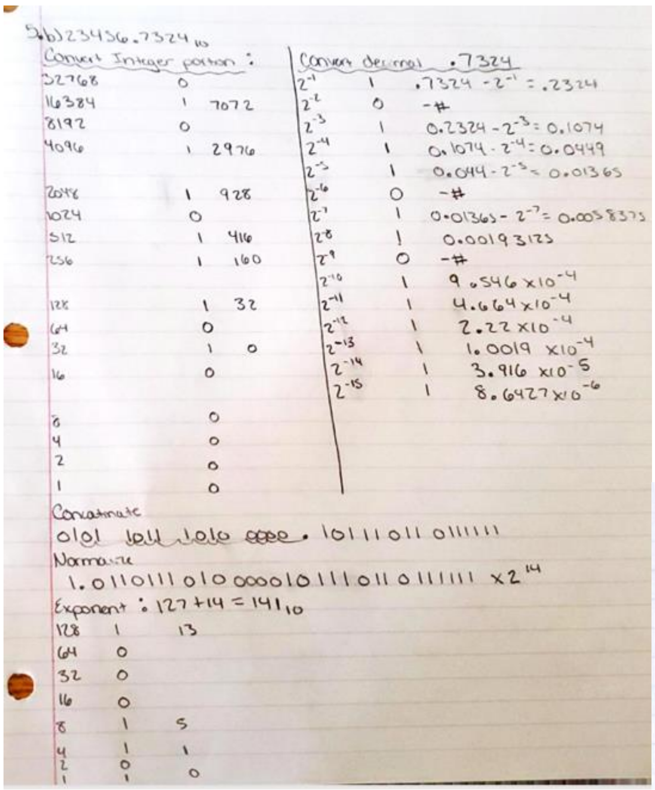
Sign bit: 0

Exponent: 1000 1101

Mantissa: 01101110100000101110110

Total: 0 10001101 01101110100000101110110

I. To start this problem I converted the integer (right) and decimal (right) portions of the number to binary first using the subtraction method. Once I had these binary strings, I concatenated them. Then the concatenated strings got normalized. This gave me an exponent of 141 because 127 plus the power of 14 from the normalized string gives us 141. I then put 141 into binary. The signed bit is zero because it is positive. The exponent portion is 141 in binary. The mantissa comes from everything after the decimal in the normalized string up to 23 bits.



5b) 23456.7324₁₀

Convert Integer portion :

32768	0	
16384	1	7072
8192	0	
4096	1	2976
2048	1	928
1024	0	
512	1	416
256	1	160
128	1	32
64	0	
32	1	0
16	0	
8	0	
4	0	
2	0	
1	0	

Convert decimal .7324

2^{-1}	1	$.7324 - 2^{-1} = .2324$
2^{-2}	0	-#
2^{-3}	1	$0.2324 - 2^{-3} = 0.1074$
2^{-4}	1	$0.1074 - 2^{-4} = 0.0449$
2^{-5}	1	$0.044 - 2^{-5} = 0.01365$
2^{-6}	0	-#
2^{-7}	1	$0.01365 - 2^{-7} = 0.0058375$
2^{-8}	1	0.0019 3125
2^{-9}	0	-#
2^{-10}	1	9.546×10^{-4}
2^{-11}	1	4.664×10^{-4}
2^{-12}	1	2.22×10^{-4}
2^{-13}	1	1.0019×10^{-4}
2^{-14}	1	3.916×10^{-5}
2^{-15}	1	8.6427×10^{-6}

Concatinate

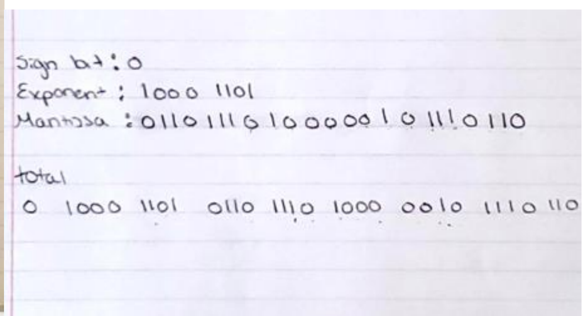
~~0101 1011 1010 0000~~ . 101110110111111

Normalize

$1.0110111010000010111011 0111111 \times 2^{14}$

Exponent : $127 + 14 = 141_{10}$

128	1	13
64	0	
32	0	
16	0	
8	1	5
4	1	1
2	0	
1	1	0



Sign bit : 0

Exponent : 1000 1101

Mantissa : 0110 1110 1000 0010 1110 110

total

0 1000 1101 0110 1110 1000 0010 1110 110

6) The answer I got for problem 6 was 34789.3125 in base 10

I. The sign bit was zero, so we know the value is positive.

II. The next step was to deal with the 8bit exponent. The number I got for the exponent was 142. Then I had to correct the bias on the exponent by subtracting 127 which gave me 15.

III. The next step was to work with the mantissa. I started by normalizing the number. We know at this point that we have ten to the 15th power. The 15 comes from the exponent step above. The next step was to adjust the number for the exponent by moving it right 15.

IV. The next step was to convert the whole number to decimal. This is done by converting all the binary before the binary point. I added up all the powers of two with one in the placement and got 34789.

V. Finally, I had to deal with converting the decimal bit from binary to decimal. This is done using the subtraction method but with two with negative powers. This gave me 0.3125 for the decimal portion

VI. If we add the whole number and decimal portion, we converted together we get 34789.3125 in base ten decimal.

6. a) 0 1000 1110 00001111111001010101

Sign bit Ø so positive value

exponent 1000 1110

128	1
64	0
32	0
16	0
8	1
4	1
2	1
1	0

$128 + 8 + 4 + 2 = 142$

Correct bias ~> $142 - 127 = 15$

Mantissa 000 0111 1110 0101 0101

Normalize $1.00001111110010101 0101 \times 10^{15}$

Adjust for exponent: 1000011111100101.0101

$2^{15}\ 2^{14}\ 2^{13}\ 2^{12}$	$2^{11}\ 2^{10}\ 2^{9}\ 2^{8}$	$2^{7}\ 2^{6}\ 2^{5}\ 2^{4}$	$2^{3}\ 2^{2}\ 2^{1}\ 2^{0}$
1 0 0 0	0 1 1 1	1 1 1 0	0 1 0 1

$2^{15} + 2^{10} + 2^{9} + 2^{8} + 2^{7} + 2^{6} + 2^{5} + 2^{2} + 2^{0} = 34789$

0.5	0
0.25	1
0.125	0
0.0625	1

$0.25 + 0.0625 = 0.3125$

34789.3125_{10}