Name: Sylvia Le

14 + 80.5 = 94.5

Course: COM219

Homework 3

20/20

Question 3

Value in base 10	Binary (IEEE-754 Floating Point)
- First bit = 0 → positive number - The exponent part: 10000011 (excess-127) to decimal 10000011_2 (pure binary) = 131 131 – 127 = 4_{10} → The exponent is 4 - The fraction: 1.101001 Mantissa: $2^{-1} + 2^{-3} + 2^{-6} = 0.640625$ → $1.101001_2 = 1.640625_{10}$ - X = sign x significand x $2^{exponent}$ = $1.640625 \times 2^4 = 26.25$ ✓	0 10000011 10100100000000000000000
- First bit = 1 → negative number - The exponent part: 10000111 (excess-127) to decimal 10000111_2 (pure binary) = 135 135 – 127 = 8_{10} → The exponent is 8 - The fraction: 1.01010100001 Mantissa: $2^{-2} + 2^{-4} + 2^{-6} + 2^{-11} = 0.3286132813$ (or 673/2048, I don't know if my calculator displayed all the digits after decimal point) → $1.01010100001_2 = 1.3286132813_{10}$ - X = sign x significand x $2^{exponent}$ = $-1 \times 1.3286132813 \times 2^8 = -340.125$ ✓	1 10000111 01010100001000000000000

-75.875	- Number is negative → first bit = 1 - 75 to binary is 1001011 (= 64 + 8 + 2 + 1) - The decimal part: + 0.875 x 2 = 1.75 (1) + 0.75 x 2 = 1.5 (1) + 0.5 x 2 = 1.0 (1) → 0.875 to binary is 0.111 → $75.875_{10} = 1001011.111_2$ → <u>Normalized</u>: 1.001011111×2^6 → Significand = 001011111 - The exponent: 6 + 127 = 133 133 to binary is: 10000101 - Combine the sign, the exponent and the significand we have: 1 10000101 00101111100000000000000 ✓
19.71875	- Number is positive → first bit = - 19 to binary is 10011 (= 16 + 2 + 1) - The decimal part: + 0.71875 x 2 = 1.4375 (1) + 0.4375 x 2 = 0.875 (0) + 0.875 x 2 = 1.75 (1) + 0.75 x 2 = 1.5 (1) + 0.5 x 2 = 1.0 (1) → 0.71875 to binary is 0.10111 → $75.875_{10} = 10011.10111_2$ → Normalized: 1.001110111×2^4 → Significand = 001110111 - The exponent: 4 + 127 = 131 131 to binary is: 10000011 - Combine the sign, the exponent and the significand we have: 0 10000011 00111011100000000000000✓

12/12

Question ~~3~~ 4

a) $(-102)_{10} = -2^{15} + 2^{14} + 2^{13} + 2^{12} + 2^{11} + 2^{10} + 2^{9} + 2^{8} + 2^{7} + 2^{4} + 2^{3} + 2$ = 1111 1111 1001 1010 ✓

address p + 1	
address p	1111 1111 1001 1010
address p - 1	

✓

b) 8-bit cell and 16-bit word, little endian

address p + 1	1111 1111
address p	1001 1010
address p - 1	

✓

c)

- For part a: since it's 16-bit cell storing 16-bit words, the byte ordering is the same for big-endian and little-endian → no changes, it's still -102 ✓
- For part b: the binary string if mistakenly read as big endian: 1001 1010 1111 1111

  $-2^{15} + 2^{12} + 2^{11} + 2^{9} + 2^{7} + 2^{6} + 2^{5} + 2^{4} + 2^{3} + 2^{2} + 2 + 1 = -25857_{10}$ ✓

12/15

Question ~~4~~ 5

	Address range of memory chip (hex)	Cell size (bits)	Size of CPU address space (locations)	Number of cells on memory chip	Percentage of chip size in address space	Capacity (bytes)
Ex	00H – 3FH	8 bits	256 locs	64 cells	$2^6/2^8 = 0.25$	64 bytes
A ✓	200H – 3FFH	8 bits	Address is in hex, has three hex digit → 12 binary digits → add. space = 2^{12}	200_{16} = 001000000000_2 $3FF_{16}$ = 001111111111_2 (changes in 9 bits) → 2^9 locations of physical memory	% = #cells/add. space $2^9/2^{12} = 0.125$	Each memory location is a byte → 2^9 x 1 = 512 bytes

				= 512/1 = 512 cells		
B ✓	0000H – ? $2^{\# \text{ bits change}} = 2^{\# \text{memory cells}}$ → #bits change = 11 0000H = 0000000000000000_2 → The other address: 0000011111111111_2 = 07FFH	32 bits	Address is in hex, has four hex digit → 16 binary digits → add. space = 2^{16}	2048 cells (= 2^{11})	% = #cells/add. space $2^{11}/2^{16}$ = 0.03125	Capacity = #cells x cell size = 2048 (2^{11}) x 4 (2^2) = 2^{13} = 2^3 x 2^{10} = 8 KB
C ✓	80000H – ? $2^{\# \text{ bits change}} = 2^{\# \text{memory cells}}$ → #bits change = 18 80000H = 10000000000000000000_2 → The other address: 10111111111111111111_2 = BFFFFH	16 bits	Address is in hex, has five hex digit → 20 binary digits → add. space = 2^{20}	Number of cells = capacity/cell size = $2^{19}/2$ = 2^{18} cells	% = #cells/add. space $2^{18}/2^{20}$ = 0.25	512 KB (kilobytes) (= 2^9 x 2^{10}= 2^{19} bytes)
D ✓	? - ? Address has 4 hex digits Let the beginning address be 0000H $2^{\#\text{bits change}}$ x cell size = capacity → $2^{\#\text{bits change}}$ = $2^{18}/2^2$ = 2^{16} → #bits change = 16 0000H = 0000000000000000_2 → The other address: 1111111111111111_2 = FFFFH	32 bits	% = #cells/add. space → add. Space = $2^{16}/1$ = 2^{16} (→ address has four hex digits)	Number of cells = capacity/cell size = $2^{18}/2^2$ = 2^{16} cells	1	256 KB (= 2^8 x 2^{10} = 2^{18} bytes)
E -3	400000H - 6FFFFFH	64 bits	Address is in hex, has six hex digit → 24 binary digits	400000H = 010000000000000000 0000000_2	% = #cells/add. space ✗ = $2^{22}/2^{24}$ = 0.25	Each memory location is 8 byte → 2^{22} x 8 (2^3) = 2^{25} =

			→ add. space = 2^{24}	6FFFFFH = 011111111111111111111111$_2$ (changes in 22 bits) → 2^{22} locations of physical memory	3x(2^20)/(2^24) = 0.1875	$2^5 \times 2^{20}$ = 32 MB 24 MB 3 Mcells x (64 bits/8 bits/byte) = 3 M x 8 = 24 MB

2^20 x 3 = 3 Mcells

18/18

Question 2 // The small blue number is for carryover / borrow.

(a)
$$\begin{array}{r} 1011 \\ +\ 0011 \\ \hline 1110 \end{array} \quad (= 14_{10})$$

Decimal: $(11)_{10}$ $(8+2+1)$
$+\ (3)_{10}$ $(2+1)$
$(14)_{10}$ $(8+4+2)$

→ Correct result, fit k = 4 for the given rep.

(b)
$$\begin{array}{r} 010100 \\ +\ 011101 \\ \hline 110001 \end{array} = (32+16+1) = 49_{10}$$

Decimal: $16+4 = 20_{(10)}$
$+\ 16+4+8+1 = 29_{(10)}$
$49_{(10)}$

→ Correct result, fit k = 6 for the given rep.

(c)
$$\begin{array}{r} 11110001 \\ -\ 01100011 \\ \hline 10001110 \end{array}$$

(= 128 + 2 + 4 + 8 = 142)

241
99

Decimal: 128 + 64 + 32 + 16 + 1 = 241
64 + 32 + 2 + 1 = 99

$241_{(10)}$
$-\ 99_{(10)}$
$142_{(10)}$

→ Correct result, fit k = 8 for the given rep.

(d) +) 110101' complement = 001010

Add 1 : 001010
$+\ \ \ \ \ \ 1$
001011

(Sign-flipping process finish)

⊕ Check: 110101 (2 complement) = −11
001011 (2 complement) = 11 ✓

+) 100011
−110101

→
$$\begin{array}{r} 100011 \\ +\ 001011 \\ \hline 101110 \end{array}$$

(= −32 + 8 + 4 + 2 = −18)

Decimal: 100011 = -29_{10} (= −32 + 2 + 1)
001011 = 11
−29 + 11 = −18

→ Correct result, fit k = 6 for the given rep.

(e)
$$\begin{array}{r} 100011 \\ +\ 101101 \\ \hline 1010000 \end{array}$$

Decimal: 100011 = −29
101101 = −19
−29 + (−19) = −48 (out of range of 6 bit 2's compleme

The result is correct with 7-bit register (−64 + 16 = −48)

but for 6-bit register, it's incorrect (16)

(d)
$$\begin{array}{r} 00011101 \\ +\ 00101010 \\ \hline 01000111 \end{array}$$

(= 64 + 4 + 2 + 1 = 71)

Decimal: 00011101 = 29 (16 + 8 + 4 + 1)
00101010 = 42 (32 + 8 + 2)
29 + 42 = 71

→ Correct result, fit k = 8 for given rep.

18.5/20

Question 6

(a)

Pairs	Hamming distance
α , β	5
α , u	3
α , σ	6
β , u	4
β , σ	5
u , σ	7

$\Rightarrow$ Distance 3 code ✓

$\rightarrow$ # of error can be detected : $3-1=2$ ✓

# of error can be corrected : $(3-1)/2 = 1$ ✓

(b)

Pairs	Hamming Distance
X , Y	6
X , Z	5
Y , Z	~~4~~ 5 ✗

-1

$\Rightarrow$ Distance ~~4~~ 5 code ✗

$\rightarrow$ # of error can be detected : $4-1=3$ ✗ 5 4

or error can be corrected: $(4-1)/2 = 1$ ✗ 5 2

(round down)

(c) +) Each hex digit = 4 binary digits

$\rightarrow$ data bit = $m = 4 \times 2 = 8$ bits (also enough to cover all possible combinations of 2 digit hex)

+) We have $m + r + 1 \leq 2^r$

$\Leftrightarrow 9 + r \leq 2^r \rightarrow r = 4$

+) B6H = $1011\ 0110_2$. see part c1 table

2FH = $0010\ 1111_2$. see part c2 table

(d) +) Each hex digit = 4 binary digits

$\rightarrow$ data bit = $m = 4 \times 4 = 16$ bits

+) We have $m + r + 1 \leq 2^r$

$\Leftrightarrow 17 + r \leq 2^r \rightarrow r = 5$

+) A3F8H = $1010\ 0011\ 1111\ 1000_2$. see part d1 table

C0AEH = $1100\ 0000\ 1010\ 1110_2$. see part d2 table

(e) See the table.

Partc 1

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
P_1 1		1		0		1		0		1										
	P_2 1	1			1	1			1	1										
			P_4 0	0	1	1					0									
							P_8 0	0	1	1	0									
															P_{16}					
		1		0	1	1		0	1	1	0									

P_1	P_2	m_{15}	P_4	m_{14}	m_{13}	m_{12}	P_8	m_{11}	m_{10}	m_9	m_8	m_7	m_6	m_5	P_{16}	m_4	m_3	m_2	m_1	m_0

3 1's at 3,7,11 $\rightarrow P_1 = 1$

5 1's at 3,6,7,10,11 $\rightarrow P_2 = 1$

2 1's at 6,7 $\rightarrow P_4 = 0$

2 1's at 10,11 $\rightarrow P_8 = 0$

$\rightarrow$ The Hamming code: $\underline{1}\underline{1}1\underline{0}011\underline{0}0110$ ✓
(the underline is for parity bits)

Parte 2

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
P_1 0		0		0		0		1		1										
	P_2 1	0			1	0			1	1										
			P_4 0	0	1	0					1									
							P_8 0	1	1	1	1									
															P_{16}					

		0		0	1	0		1	1	1	1									
P_1	P_2	m_{15}	P_4	m_{14}	m_{13}	m_{12}	P_8	m_{11}	m_{10}	m_9	m_8	m_7	m_6	m_5	P_{16}	m_4	m_3	m_2	m_1	m_0

+) For p_1 : two 1's at 9, 10

$\rightarrow p_1 = 0$

+) For p_2 : three 1's at 6, 10, 11

$\rightarrow p_2 = 1$

+) for p_4 : two 1's at 6, 12

$\rightarrow p_3 = 0$

+) For p_8 : four 1's at 9, 10, 11, 12

$\rightarrow p_8 = 0$

→ The Hamming code : 010001001111 ✓

<u>Part d1</u>

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
P_1 1		1		0		0		0		1		1		1		1		0		0
	P_2 0	1			1	0			0	1			1	1			1	0		
			P_4 1	0	1	0					1	1	1	1					0	0
							P_8 1	0	0	1	1	1	1	1						
															P_{16} 0	1	1	0	0	0

		1		0	1	0		0	0	1	1	1	1	1		1	1	0	0	0
P_1	P_2	m_{15}	P_4	m_{14}	m_{13}	m_{12}	P_8	m_{11}	m_{10}	m_9	m_8	m_7	m_6	m_5	P_{16}	m_4	m_3	m_2	m_1	m_0

→ The Hamming code : 101 1010 100 11111 011000 ✓

Part d2

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
P_1 1		1		1		0		0		0		1		1		0		1		0
	P_2 0	1			0	0			0	0			0	1			1	1		
			P_4 0	1	0	0					0	1	0	1					1	0
							P_8 0	0	0	0	0	1	0	1						
															P_{16} 1	0	1	1	1	0

		1		1	0	0		0	0	0	0	1	0	1		0	1	1	1	0
P_1	P_2	m_{15}	P_4	m_{14}	m_{13}	m_{12}	P_8	m_{11}	m_{10}	m_9	m_8	m_7	m_6	m_5	P_{16}	m_4	m_3	m_2	m_1	m_0

→ The Hamming code : 101010000000101101110 ✓

Parte 1

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
P_1 0		1		1		0		1		0		0		1		0		0		1		0		1
	P_2 0	1			0	0			1	0			0	1			0	0			1	0		
			P_4 0	1	0	0					0	0	0	1					0	1	1	0		
							P_8 0	1	1	0	0	0	0	1									0	1
															P_{16} 1	0	0	0	0	1	1	0	0	1

0	0	1	0	1	0	0	0	1	1	0	0	0	0	1	1	0	0	0	0	1	1	0	0	1

P_1	P_2	m	P_4	m_{14}	m_{13}	m_{12}	P_8	m_{11}	m_{10}	m_9	m_8	m_7	m_6	m_5	P_{16}	m_4	m_3	m_2	m_1	m_0	m	m	m	m

→ All parity bits are correct → no error

→ The databits are : 1100 1100 0010 0001 1001

Using the look up table the hex value is : CC219 H ✓

-0.25

Parte 2

	1	2	3	4	5 ✓	6 ✓	7 ✓	8	9	10	11	12 ✓	13 ✓	14 ✓	15 ✓	16	17 ✓	18 ✓	19 ✓	20 ✓	21 ✓	22 ✓	23 ✓	24 ✓	25 ✓
7 1's ←	P_1 0		1		0		✓ 0		1		0		1		1		1		0		1		1		0
6 1's ←		P_2 1	1			✓ 1	0			0	0			0	1			1	0			1	1		
				P_4 1	0	1	0					0	1	0	1					1	1	1	1		
3' 1's ←								P_8 0	1	0	0	✓ 0	✓ 1	✓ 0	✓ 1									0	0
																P_{16} 0	1	1	0	1	1	1	1	0	0

0	1	1	1	0	1	0	0	1	0	0	0	1	0	1	0	1	1	0	1	1	1	1	0	0
P_1	P_2	m	P_4	m_{14}	m_{13}	m_{12}	P_8	m_{11}	m_{10}	m_9	m_8	m_7	m_6	m_5	P_{16}	m_4	m_3	m_2	m_1	m_0	m	m	m	m

+) Parity bits P_1, P_2, P_8 are incorrect

+) The correct data bit are those checked by P_4 and P_{16}, indicate by the green tick

→ The incorrect bit(s) must be in the list: 3, 4, 9, 10, 11

→ Simplest fix: change bit 11 from 0 → 1

+) The data bit are now: 1010 1010 1011 1011 1100

Using the lookup table, hex value is: AABBCH ✓

-0.25

(f) For part c : length of message is 12 < 16 → condition of no 2 bits closer than 16 bits not satisfied

→ pass ✓

For part d : See the table in the next page

(see table at the back)

Suppose we want to send message FF00 H = 1111 1111 0000 0000_2

Suppose the error is on bit 3 and 20

→ p_1, p_2, p_4, p_{16} incorrect (the value of parity bits here is the value when message is correct)

A way to fix : Change bit 19, from 0 → 1

→ p_1, p_2, p_{16} are correct

Change p_4, from 0 → 1 (parity bits can be an error too, why not ?)

Or change bit 21 and bit 2 ‖

→ The resulted message : 0111 1111 0000 0110

Conclusion : error can be detected, but may not be correctly fixed ✓

For part e : Since the table is just a minor extension of the table in d

Suppose now the message is FF000 H, the error is the same as described above

The same process happens, where it can detect but not correct error

→ Also cannot work properly ✓

A little bit more on part c : So since the condition was not satisfied, in such a scheme like this, only single-bit error can occur

With that in mind, if we use the table in part c, we see that all data bits are checked by 1-2 parity bits. Since only single-bit error, if a bit is wrong, all the parity bit are wrong, you just need to look at the data bit in common that is checked by the incorrect parity, flip that → corrected

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
							✓	✓	✓	✓	✓	✓	✓	✓						
P_1 1		0		1		1		1		1		0		0		0		1		0
	P_2 1	0			1	1			1	1			0	0			0	1		
			P_4 0	1	1	1					1	0	0	0					1	0
							P_8 0	1	1	1	1	0	0	0						
															P_{16} 0	0	0	1	1	0
		1		1	1	1		1	1	1	1	0	0	0		0	0	0	0	0

P_1	P_2	m_{15}	P_4	m_{14}	m_{13}	m_{12}	P_8	m_{11}	m_{10}	m_9	m_8	m_7	m_6	m_5	P_{16}	m_4	m_3	m_2	m_1	m_0