Michael McNichol – Exam 01 – Part 02
247345 (2021SP Microprocessor Assembly Lang (CIS-2420-NET01))

1.) Endian order can be big or little. Big Endian stores the MSB at the lowest memory location of that block where Little Endian is the inverse storing the MSB at the highest memory location of that block.

This is essentially the order of how you read these blocks of memory back whether that is left to right or right to left as you consume each chunk. Personally I find Big Endian more logical to think through but in questioning why we would use little endian ever, I had in previous discussions learned one of the benefits is that if you were to cast a smaller value that was taking up a significant amount of space (say the value 10 decimal in a QWORD), it could be cast down to a WORD without needing to move the bits around at all giving some performance benefits.

I think this highlights the goals of performance in the x86 processing designs as opposed to ARM architectures. x86 uses little endian whereas ARM uses big endian. I think the SPARC chips from Sun Micro had a feature that allowed both big and little endian. It was in direct conflict of….and this is a longer pull in history….the Soul of a New Machine era where flags were used for flipping the bit word length from 8 -> 16 and leveraging of these flags was railed against pretty hard by Data Generals De Castro who was a genius in his own right.

2 – 11: **Attached Photo of Paper**

Note: **On question 7 for the Floating Point I was a little confused because you didn't say the mantissa was 3 bits etc. like you had done in the Assigned work previously. You used the term "Pure Binary" which I'm guessing you just meant place the decimal where it would go and show the bits as the ½, ¼, etc. I get wrapped around the axle fairly easy so may have misread this.*

1) 011111 = 24+7 = 31

2)

A	B	¬B	A ∧ ¬B
T	T	F	F
T	F	T	T
F	T	F	F
F	F	T	F

A	B	¬A	¬A ∨ B	¬(¬A ∨ B)
T	T	F	T	F
T	F	F	F	T
F	T	T	T	F
F	F	T	T	F

De Morgan

3) 10011111
01100000+1
Signed = −97
Unsigned = 159

4) 7 bits for 100_{10}
10 bits for 444_{10}
14 bits for 60000_{10}

5) 4
16
8
128 + 1 = $129_{10} = 81_{16}$

512 + 0 + 10 = $522_{10} = 20A_{16}$

256 + 16 (×7) + 14 = $382_{10} = 17E_{16}$

6) 01010001
10100111
A − F$_{16}$ = -81_{10}

01101001
10010110
9 − 6$_{16}$ = -105_{10}

00010011
11101100
E − D$_{16}$ = -19_{10}

7) $110.111_2 = 6.875_{10}$

$101.1_2 = 5.5_{10}$

8) .7 × 2 = 1 + .4
.4 × 2 = 0 + .8
.8 × 2 = 1 + .6
.6 × 2 = 1 + .2
.2 × 2 = 0 + .4
0 + .8
1 + .6
Infinitely Repeating
.70 cannot be represented properly in decimal

9) 1.10110011

10) Overflow Flag – If the signed value overflows to a value it can't represent, this flag is set
Zero Flag – If an operation produces a 0, this flag is set
Sign Flag – Indicates a negative result if MSB is set

11) UNSIGNED