Name: S M Ragib Rezwan

ID: 103172423

Cos10003, Assesment 01

(Starting point 3 for all)

1)a)1474

Diving 1474 by 16, we get quotient = 92 and remainder= 2

Diving 92 by 16, we get quotient = 5 and remainder= 12 =c

a=10, b=11, c=12, d=13, e=14, f=15

Now divide 5 by 16, we get quotient=0 and remainder=5

Hence, the hex number is, 5C2

So, 1474 (decimal) = 0x 5C2

b) -144

step-1) at first we find the binary form of +144:

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

(144>128, so 1;

new value is 144-128=16; 16<64, so 0; 16<32 so 0; 16=16, so 1; and now we can give rest as zero as cycle has ended)

step-2) then we flip it

hence originally: 1 0 0 1 0 0 0 0

now becomes: 0 1 1 0 1 1 1 1

step-3) then we add +1 to it:

01101111 + 1 = 0111 0000

Therefore final answer = (0b) 0111 0000

c) 79.125

step-1) sign = positive = 0

step-2) finding mantissa number:

79 (decimal) in binary:

	64	32	16	8	4	2	1
(0b)	1	0	0	1	1	1	1

(79>64, so 1; new value => 79-64= 15,

15<32, so 0; 15<16, so 0 again; 15>8, so 1; new value => 15-8=7

7>4, so 1; new value => 7-4=3;   3>2, so 1; new value => 3-2=1

1=1, so 1)

(so “7” place values have been found from 79, so we need 24-7= 17 place values from the 0.125 part, as mantissa has 23 places (in IEEE code) and as the first “1” we got in binary for 79 will be used as part of format for mantissa)

0.125 (decimal) into binary

0.125 * 2 = 0.25   so 0

0.25 * 2 = 0. 5   so 0

0.5 * 2 = 1.0   so 1

0.0 * 2 = 0   so 0, and the rest will be zero as well as it will keep continuing 0*2=0

So 0.125 (decimal) into binary (with 17 places):

(0b)   0010 0000 0000 0000 0

Hence 79.125 (decimal) into binary is: (0b)   1001111.00100000000000000

Or in standard form => 1.00111100100000000000000 * 10^6

So mantissa = 00111100100000000000000

(as we will only take the 23 bits after decimal point)

Step-3) Exponent: 6+127= 133 (decimal)

Hence, 133 in binary,

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

(133>128, so 1; new value => 133-128=5;

5<64, so 0; 5<32, so 0; 5<16, so 0; 5<8, so 0; 5>4, so 1; new value => 5-4 = 1;

1<2, so 0; 1=1, so 1)

Hence exponent: 10000101

Therefore, the entire thing in IEEE = 0 10000101 00111100100000000000000

2) 0010 1110 1001 1010

a) find in hex: 8 4 2 1

(0b) 0010= 2(decimal) = 0x 2 a=10, b=11, c=12, d=13, e=14, f=15

(0b) 1110= 14(decimal) = 0x E

(0b) 1001= 9(decimal) = 0x 9

(0b) 1010= 10(decimal) = 0x A

Hence full number in Hex: 0 x 2 E 9 A

b) converting into two 8 bits format: (0b) 00101110 (0b) 10011010

(0b) 00101110 = (2^1) + (2^2) + (2^3) + (2^5) (in decimal)

=2+4+8+32 = 46 (in decimal)

(0b) 10011010 = (2^1) + (2^3) + (2^4) + (2^7)   (in decimal)

=2+8+16+128 = 154 (in decimal)

Hence, the two unsigned bit integers are = 46,   154

c) 2 “8” bit signed (two’s complement) back into decimal

(0b)   00101110   (0b)   10011010

For   (0b)   00101110,

Step-1) subtract 1 from the number, so:

00101110 - 1 = 00101101

Step-2) un-flip the number, so:

Flipped position (or position now) =   00101101

Hence, previous or un-flipped position =11010010

Step-3) Hence binary number of the positive form of decimal is 11010010

So the positive form of the decimal is = (2^1)+(2^4)+(2^6)+(2^7) = 2+16+64+128= 210

Hence the actual number will be -210

For (0b)   10011010

Step-1) subtract 1 from the number, so:

10011010 – 1 = 10011001

Step-2) un-flip the number, so:

Flipped position (or position now) =   10011001

Hence, previous or un-flipped position =01100110

Step-3) Hence binary number of the positive form of decimal is= (2^1)+(2^2)
+(2^5)+(2^6)=2+4+32+64=102

So the positive form of the decimal is = 102

Hence the actual number will be = -102

So, the numbers will be = -210, -102

d) one half precision floating point as a decimal value:

(according to Wikipedia (https://en.wikipedia.org/wiki/Half-precision_floating-point_format),

the IEEE 754 can take half precision, but with new format:

"1 bit for sign", "5 bit for exponent", "11 bits for significand precision" (I assume it meant mantissa)

It also said "exponent bias is 15". So using that info, I broke the main data into segments)

0 01011 1010011010

Sign = 0 = positive

Exponent = (0b) 01011 – 15 (decimal)

= (2^0) + (2^1) + (2^3) – 15 = 1 + 2 + 8 – 15= -4

So, exponent= -4

Mantissa= 1.1010011010

= (2^0) +(2^(-1)) +(2^(-3)) +(2^(-6)) +(2^(-7)) +(2^(-9)) = 1.650390625

So, mantissa = 1.650390625

So number= ((-1)^sign) * (2^exponent) * (mantissa)

=1 * (2^(-4)) * (1.650390625)

= 0.1031494141

3) c2 b9 f0 9d 9f ba 39 e2 a0 a1

At first we need to go back 1 step, to binary form for each, in the encoded format (i.e. going the reverse way of how we found hex no from Unicode in lecture02)

0x c= 12 (in decimal) =(0b) 1100 8 4 2 1

0x 2= 2 (in decimal) =(0b) 0010 a=10, b=11, c=12, d=13, e=14, f=15

0x b= 11 (in decimal) = (0b) 1011

0x 9= 9 (in decimal) = (0b) 1001

0x f= 15 (in decimal) = (0b) 1111

0x 0= 0 (in decimal) = (0b) 0000

0x 9= 9 (in decimal) = (0b) 1001

0x d= 13 (in decimal) = (0b) 1101

0x 9= 9 (in decimal) = (0b) 1001

0x f= 15 (in decimal) = (0b) 1111

0x b= 11 (in decimal) = (0b) 1011

0x a= 10 (in decimal) = (0b) 1010

0x 3= 3 (in decimal) = (0b) 0011

0x 9= 9 (in decimal) = (0b) 1001

0x e= 14 (in decimal) = (0b) 1110

0x 2= 2 (in decimal) = (0b) 0010

0x a= 10 (in decimal) = (0b) 1010

0x 0= 0 (in decimal) = (0b) 0000

0x a= 10 (in decimal) = (0b) 1010

0x 1= 1 (in decimal) = (0b) 0001

Then I added the consecutive 2 nibbles to make bytes:

11000010 10111001 11110000 10011101 10011111 10111010 00111001 11100010 10100000 10100001

(here, total no of bytes= 10)

(now it's time to separate them into different Unicode with reference to byte table and encoding format)

A) 110***** is the format encoding in 1st byte for 2 bytes given for (U+0080 to U+07FF)

So, 1 Unicode will come from= 11000010 10111001

B) 11110*** is the format encoding in 1st byte for 4 bytes given for (U+10000 to U+10FFFF)

So, 1 Unicode will come from=11110000 10011101 10011111 10111010

C) 0******* is the format in 1st byte for 1 byte given for (U+0000 to U+007F)

So 1 Unicode will come from= 00111001

D) 1110**** is the for 1st byte in 3 bytes given for (U+0800 to U+FFFF)

So 1 Unicode will come from = 11100010 10100000 10100001

So for A) encoded value is 11000010 10111001
So, if we remove the format, we get 00010 111001
Rearranging it with 4 bit sections from the end, 000 1011 1001
Removing the excess 3 "0" from start, 1011 1001
Converting them into hexa, (0b) 1011= 11 (decimal) = 0x B
(0b) 1001= 9 (decimal)= 0x 9
So Unicode= U+00B9

So for B) encoded value is 11110000 10011101 10011111 10111010
So, if we remove the format, we get 000 011101 011111 111010
Rearranging it with 4 bit sections from the end, 0 0001 1101 0111 1111 1010
Removing the excess 1 "0" from start, 0001 1101 0111 1111 1010
Converting them into hexa, (0b) 0001= 1 (decimal) = 0x 1
(0b) 1101= 13 (decimal)= 0x D
(0b) 0111= 7 (decimal)= 0x 7
(0b) 1111= 15 (decimal)= 0x F
(0b) 1010= 10 (decimal)= 0x A

So Unicode= U+1D7FA

So for C) encoded value is 00111001
So, if we remove the format, we get 00111001 (Format matched so didn't remove 0 from here)
Rearranging it with 4 bit sections from the end, 0011 1001
Converting them into hexa, (0b) 0011= 3 (decimal) =0x 3
(0b) 1001= 9 (decimal)=0x 9

So Unicode= U+0039

So for D) encoded value is 11100010 10100000 10100001
So, if we remove the format, we get 0010 100000 100001
Rearranging it with 4 bit sections from the end, 0010 1000 0010 0001 (not removing zeroes as no excess zero here)
Converting them into hexa, (0b) 0010= 2 (decimal) = 0x 2
(0b) 1000= 8 (decimal)= 0x 8

(0b) 0010= 2 (decimal)= 0x 2
(0b) 0001= 1 (decimal)= 0x 1

So Unicode= U+2821

Hence the Unicodes are: U+00B9 U+1D7FA U+0039 U+2821