CS303-HW1 Deniz Cangı

Q1) a)

E3	= 11100011	= 99	= c
73	= 01110011	= 115	= s
B3	= 10110011	= 51	= 3
B0	= 10110000	= 48	= 0
B3	= 10110011	= 51	= 3
20	= 00100000	= 32	= ' '
68	= 01101000	= 104	= h
F7	= 11110111	= 119	= w
31	= 00110001	= 49	= 1
20	= 00100000	= 32	= ' '
F1	= 11110001	= 113	= q
31	= 00110001	= 49	= 1

→ result = cs303 hw1 q1

b) Parity used is odd.

Q2)

1 0110000 = 0
0 0110001 = 1
0 0110010 = 2
1 0110011 = 3
0 0110100 = 4
1 0110101 = 5
1 0110110 = 6
0 0110111 = 7
0 0111000 = 8
1 0111001 = 9

Q3) a) 127's 8-digit 2's complement signed form
0111 1111

1 = 0000 0001
↓ 1's complement
1111 1110
↓ 2's complement
−1 = 1111 1111

```
   1111 111
   0111 1111
 + 1111 1111
 -----------
  X0111 1110
```

result = 01111110 $\xrightarrow{\text{decimal}}$ 126. ✓

127 − 1 = 126 ✓

b) 30 = 00011110

71 = 01000111 $\xrightarrow{\text{2's complement}}$ 10111001

```
   111
  00011110
+ 10111001
----------
  11010111
```

11010111 = −128+64+16+4+2+1 = −41 ✓

30 − 71 = −41 ✓

c) 30 = 00011110

−71 = 10111001 $\xrightarrow{\text{2's comp.}}$ 01000111 = 71

```
   1111
  00011110
+ 01000111
----------
  01100101
```

01100101 = 64+32+5 = 101 ✓

30 − (−71) = 30 + 71 = 101 ✓

d) −60 − (−127) = −60 + 127 = ?

60 = 00111100 $\xrightarrow{\text{2's comp.}}$ −60 = 11000100

127 = 01111111

```
   11111
  11000100
+ 01111111
----------
 X01000011
```

X01000011 = 64+2+1 = 67 ✓

−60 + 127 = 67 ✓

e) 39.5 = 00100111.1

41.75 = 00101001.11

↓ 1's comp.

11010110.00

↓ 2's comp.

−41.75 = 11010110.01

```
   11
  00100111.1
+ 11010110.01
-------------
  11111101.11
```

11111101.11 = −128+64+32+16+8+4+1+ 0.5+0.25 = **−2.25** ✓

39.5 − 41.75 = −2.25 ✓

f) 41.84375 => 41 = 00101001

0.84375	×2 =	1.6875	→	1
0.6875	×2 =	1.375	→	1
0.375	×2 =	0.75	→	0
0.75	×2 =	1.5	→	1
0.5	×2 =	1.0	→	1

41.84375 = 00101001.11011

80.15625 => 80 = 01010000

0.15625	×2 =	0.3125	→	0
0.3125	×2 =	0.625	→	0
0.625	×2 =	1.25	→	1
0.25	×2 =	0.5	→	0
0.5	×2 =	1.0	→	1

80.15625 = 01010000.00101

↓ 2's complement

−80.15625 = 10101111.11011

1 11111 1 11

41.84375 = 00101001.11011

−80.15625 = + 10101111.11011

11011001.10110

=

−128 + 64 + 16 + 8 + 1 + 0.5 + 0.125 + 0.0625

=

−38.3125 ✓

41.84375 − 80.15625 = −38.3125 ✓

Q4) arithmetic modulo 64

64 in binary = 1000000

64 in octal = 100

64 in hexadecimal = 40

128 in binary = 10000000
128 in octal = 200
128 in hexadecimal = 80

192 in binary = 11000000
192 in octal = 300
192 in hexadecimal = C0

For binary: look at rightmost 6 digits to find the modulo 64

For octal: look at rightmost 2 digits to find the modulo 64.

For hexadecimal = need to consider the rightmost 2 digits. if the second lsb is 4, 8 or C it's multiple of 64 but if it's not, it's added to modulo result.

I would use octal number systen because to calculate the modulo, I would need to consider two least significant bits. The digits available in the two rightmost bits are to be the result of the modulo 64.

Q5) $\frac{1}{7}$ → using 8 bits in fraction

$\frac{1}{7} \times 2 = \frac{2}{7} < 1 \rightarrow 0.$

$\frac{2}{7} \times 2 = \frac{4}{7} < 1 \rightarrow 0.$

$\frac{4}{7} \times 2 = \frac{8}{7} > 1 \rightarrow 1$

$\frac{1}{7} \times 2 = \frac{2}{7} < 1 \rightarrow 0$

$\frac{2}{7} \times 2 = \frac{4}{7} < 1 \rightarrow 0$

$\frac{4}{7} \times 2 = \frac{8}{7} > 1 \rightarrow 1$

$\frac{1}{7} \times 2 = \frac{2}{7} < 1 \rightarrow 0.$

$\frac{2}{7} \times 2 = \frac{4}{7} < 1 \rightarrow 0$

using 8 bits in fraction

$\frac{1}{7} = 0.00100100$

to the decimal: $2^{-3} + 2^{-6} = 0.125 + 0.015625$

$= 0.140625$

1/7 in real = 0.142857142857142857...
↳ it goes to infinity.

the precise equivalent in binary form cannot be found using 8-bits. This leads to precision error, which happens with fractions when it cannot convert to binary exactly.