Ahmed Abd-Elsalam Muhammed Afify

HW_1

Changing Formats

In [1]:
```
import struct

def floatToBinary32(value):
    return ''.join(f'{c:0>8b}' for c in struct.pack('!f', value))

def binaryToFloat(value):
    hx = hex(int(value, 2))
    return struct.unpack("f", struct.pack("l", int(hx, 16)))[0]

# float to binary
fl0 = 19.5
binstr = floatToBinary32(fl0)
print(f'Binary equivalent of {fl0}: {binstr}')

# binary to float
fl1 = binaryToFloat(binstr)
print(f'Decimal equivalent of      {binstr}: {fl1}')

print(f'\nSign     ( 1 bit ) = {binstr[0]}\nExponent ( 8 bits) = {binstr[1:9]}\nM

assert fl0 == fl1
```

```
Binary equivalent of 19.5: 01000001100111000000000000000000
Decimal equivalent of      01000001100111000000000000000000: 19.5

Sign     ( 1 bit ) = 0
Exponent ( 8 bits) = 10000011
Mantissa (23 bits) = 00111000000000000000000
```

In [2]:
```
import struct

getBin = lambda x: x > 0 and str(bin(x))[2:] or "-" + str(bin(x))[3:]

def floatToBinary64(value):
    val = struct.unpack('Q', struct.pack('d', value))[0]
    return getBin(val)

def binaryToFloat(value):
    hx = hex(int(value, 2))
    return struct.unpack("d", struct.pack("q", int(hx, 16)))[0]

# floats are represented by IEEE 754 floating-point format which are
# 64 bits long (not 32 bits)

# float to binary
binstr = floatToBinary64(19.5)
print('Binary equivalent of 19.5:')
print(binstr + '\n')

# binary to float
fl = binaryToFloat(binstr)
print('Decimal equivalent of ' + binstr)
print(fl)
```

```
Binary equivalent of 19.5:
1000000001100111000000000000000000000000000000000000000000000000

Decimal equivalent of 1000000001100111000000000000000000000000000000000000000000000000
000000
19.5
```

In [ ]:
```

```