1) Given the binary number 11010111001.01101, convert it to an IEEE754 single-precision floating-point format. Show complete work.

1 sign bit:
0 because positive
scientific notation:
$1.101011100101101 * 2^{10}$
exponent bias:
127 because positive exponent
10 exp bits:
127 + 10 = 137 -> 10001001 (binary)
fraction bits:
101011100101101
floating point representation (sign bit - exp bits - fraction bits):
0100 0100 1101 0111 0010 1101 0000 0000

2) Convert the decimal number -45.75 to its IEEE754 single-precision floating-point representation. Show complete work.

Decimal to Binary:

45	
22	1
11	0
5	1
2	1
1	0
0	1

.75 * 2 = 1.5 -> 1
.5 * 2 = 1.0 - > 1
-45.75 -> -101101.11 (binary)
1 sign bit:
1 because negative
scientific notation:
$-1.0110111 * 2^5$
exponent bias:
127 because positive exponent
5 exp bits:
127 + 5 = 132 -> 10000100 (binary)
fraction bits:

0110111
floating point representation (sign bit - exp bits - fraction bits):
1100 0010 0011 0111 0000 0000 0000 0000

3) Convert the decimal number 5.4 to its IEEE754 single-precision floating-point representation. Show complete work.

5.4	
2	1
1	0
0	1

.4 * 2 = .8 -> 0
.8 * 2 = 1.6 -> 1
.6 * 2 = 1.2 -> 1
.2 * 2 = .4 -> 0
…
5.4 -> 101.0110… (binary)
1 sign bit:
0 because positive
scientific notation:
$1.010110 * 10^2$
exponent bias:
127 because positive exponent
2 exp bits:
127 + 2 = 129 -> 10000001 (binary)
fraction bits:
010110
floating point representation (sign bit - exp bits - fraction bits):
0100 0000 1010 1100 1100 1100 1100 1100

4) Floating point numbers are estimates (not exact). Explain.
Computers have a finite word length which limits the accurate representation of a number. To increase the accuracy of floating point numbers, more digits should be added to the mantissa, with more digits added to the exponent. This limitation in the computer's capability makes floating point numbers not exact estimates.

5)
```
INCLUDE Irvine32.inc

.data
```

```
    grade BYTE 'B', 0

.code
main PROC
    mov al, grade

    cmp al, 'A'
    je case_A

    cmp al, 'B'
    je case_B

    cmp al, 'C'
    je case_C

    jmp default_case

case_A:
    jmp end_switch

case_B:
    jmp end_switch

case_C:
    jmp end_switch

default_case:

end_switch:
    exit
main ENDP

END main
```