Assignment 3-3 Floating Point Representation: Check the DeNormalized Number. Lab 2

1. **Make an assembly program that has the hexadecimal data to represent the smallest normalized positive number and largest denormalized positive number.**

```
## Program to represent 1.0

      .data
val1:   .word  0x00800000
val2:   .word  0x007FFFFF
        .text
      .globl  main

main:
  li $v0, 2       # print floating service code
        lwc1 $f11, val1
        move $a0, $v0
        mfc1 $t1, $f11
        syscall
        lwc1 $f12, val2
        move $a0, $v0
        mfc1 $t2, $f12
        syscall
## End of file
```

2. **Check out the data section**
    a. Compare the two values in the data Section, is there a difference? Why?

```
                User Data Segment

[10010000] 00800000 007fffff 00000000 00000000
```

Yes there is a difference. It's due to 0x00800000 being the smallest normalized positive number while 0x007fffff is the largest denormalized positive number.

   b. **Run this program and compare two values in $f11 and $f12, is there a difference? Why?**

```
FG10     = 0.00000
FG11     = 1.17549e-38
FG12     = 1.17549e-38
FG13     = 0.00000
```

No, there is no difference. Even though they come from different representations, **both numbers are so close to each other that when converting them into a floating point**

number they round to the same value, which is 1.17549e-38. In other words, while 0x00800000 is normalized, 0x007FFFFF is just slightly smaller than the other, but due to precision limits, val2 rounds to the same floating-point value as val1.

c. **Elaborate on the field format of**

i. **Smallest positive number**

0x00800000 = 0000 0000 1000 0000 0000 0000 0000 0000
0 = sign bit = **positive**
0000 0001 = Exponent = **1**
Biased exponent = 1, actual exponent = -126 because **1 - 127 is -126.**
000 0000 0000 0000 0000 0000 = Mantissa = still exactly **1.0**

1.0 x 2^-126 is approximately **1.17549 x 10^38**, **this matches the value in $f11**

ii. **Largest denormalized positive number**

0x007FFFFF=0000 0000 0111 1111 1111 1111 1111 1111
0 = sign bit = **positive**
0000 0000 = **denormalized** according to IEEE 754 standards.
111 1111 1111 1111 1111 1111 = Mantissa = largest possible value for the mantissa in a denormalized number, so it’s 1 - 2^-23, which comes out to approximately **0.9999- and more** which is **very close to 1.0, which also confirms why val1 and val2 shared the same representation in $f11 and $f12.**