Assignment 3-6 Floating Point Representation: Infinity. Lab 5

1. **Make an assembly program that shows Infinity. Divide the greatest number by 0.**

```
## Program: Largest Number / 0 = Infinity
        .data
val1:    .word  0x7F7FFFFF  # Largest Number
           .text
        .globl  main
main:
    li $v0, 2       # print floating service code

    li              $t0, 0
    mtc1            $t0, $f2        # $t0 = 0;
    cvt.s.w         $f4, $f2    # Convert the double precision number to single precision
    l.s     $f6, val1
    div.s           $f9, $f6, $f4   # Largest Number / 0
    li              $v0, 2
    mfc1            $t0, $f9
    mov.s           $f12, $f9
    syscall
## End of file
```

2. **Print the results of division**

Notes: When you divide a positive number (including the largest normalized positive number) by zero, the result is defined as positive infinity.
And if the number is negative divided by 0, we get a negative infinity instead.
While in basic arithmetic the answer is undefined, in IEEE 754 floating-point standard, it provides a way to handle this operation by defining x/0 for positive x as positive infinity. This handling of division by zero is crucial in computer programming and numerical computations, allowing for more robust handling of edge cases in calculations.

   a. **Elaborate the value .word 0x7F7FFFFF**
      i. **With sign / exponent / mantissa**

0x7FFFFFFF = 0111 1111 0111 1111 1111 1111 1111 1111
0 = sign bit = **positive**
1111 1110 = Exponent = 254 - 127 = **127** (largest normalized positive exponent value)
111 1111 1111 1111 1111 1111 = Mantissa = Represents a normalized fraction **approximately equal to 1.99999** and so on.

      ii. What is the meaning of this value?

It's the largest normalized positive value that can be represented in IEEE 754 single-precision floating-point format.

Notes:

Practical Implications*: This number is significant in computing as it defines the upper limit for positive floating-point calculations using single-precision representation. It indicates the maximum range for positive values before transitioning into overflow*

Usage Context*: In scientific computing, graphics, and simulations, being able to represent large values is crucial for accuracy in calculations involving very large quantities, such as distances in astronomy, large-scale physical simulations, and financial models.*

b. Elaborate on the numbers in the registers based on the IEEE 754 format

i. $f9

Showcases **Infinity**, but the value of infinity was previously stored and transferred from $t0 which showcases the hexadecimal representation of 0x7F800000, which is **Positive Infinity**

ii. $t0. Is this infinity? Why? Explain it with IEEE 754 format

Accidentally answered this in the previous question oops,

So **yes this is Infinity** because it stores 0x7F800000.

0x7F800000 = 0111 1111 1000 0000 0000 0000 0000 0000

0 = sign bit = **positive**

1111 1111 = Exponent = When the exponent bits are all 1's, it indicates that the value is **either Infinity or NaN.**

000 0000 0000 0000 0000 0000 = When the mantissa bits are all 0's when the exponent bits are all 1's, this tells us that the value is **Infinity**.

Knowing that the sign bit is positive, we can conclude that this is a **Positive Infinity**