Assignment 3-2 Convert to the Floating Point Representation from Decimal. Lab 1

1. Make a assembly program that has the floating data 1.0

```
## Program to represent 1.0

      .data
val1:    .float  1.0
         .text
      .globl  main

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

Mfc1 stands for move from coprocessor 1. It’s use to transfer data between floating-point coprocessor and the general purpose registers.
$9, Destination register ($t1).
$f12, Source register from which the floating point value will come from.

2. **Then check out the data section**
   a. **What is the value hexadecimal?**

```
3f800000 = 0011 1111 1000 0000 0000 0000 0000 0000
```

First 9 bits are the sign and exponent, so the remaining bits are the mantissa, which holds the actual decimal value **1.0**, so the value the hexadecimal holds is **1.0.**
But if we were to treat it as an **unsigned or signed integer, we would get 1,065,353,216**

   b. **Represent in the IEEE 754 format.**

How it’s sectioned out **0 01111111 00000000000000000000000**
It’s the Binary32 standard, which is represented as 1 sign bit, 8 exponent width bits, and the rest of the 24 (23 explicitly stored) bits as mantissa or the significand precision bits.
0 = sign bit = **positive**
0111 1111 = exponent **decimal value = 127** Bias is 127 for Binary32, so **127 - 127 = Exponent value is 0**
00000000000000000000000 = mantissa. *Note: Mantissa is normalized, meaning that the value is implicitly 1.F, since the mantissa is all zeros it represents* **1.0.**

   c. **Elaborate on the vault $t1 with IEEE 754 format.**

$t1, aka $9, holds 3f800000, which stated above, **holds 1.0 in IEEE 754 format**. If not in IEEE 754, and if it's instead read at it's face value without the IEEE standard, it would hold **1,065,353,216**.