```
*** Given that $t1 = 0xF7AB1B1F, $t2 = 0xA976FBDE what value is stored in $t0 after
    the bit operation is completed (20 pts).

addi $t0, $t1, 0xE3AB

    1111 1111 1111 111        11   111 111
    1111 0111 1010 1011 0001 1011 0001 1111 <-- 0xF7AB1B1F
    1111 1111 1111 1111 1110 0011 1010 1011 <-- 0xE3AB
    ---------------------------------------
    1111 0111 1010 1010 1111 1110 1100 1010 <-- 0xF7AAFECA


or $t0, $t1, $t2

    1111 0111 1010 1011 0001 1011 0001 1111 <-- 0xF7AB1B1F
    1010 1001 0111 0110 1111 1011 1101 1110 <-- 0xA976FBDE
    ---------------------------------------
    1111 1111 1111 1111 1111 1011 1101 1111 <-- 0xFFFFFBDF


xor $t0, $t1, $t2

    1111 0111 1010 1011 0001 1011 0001 1111 <-- 0xF7AB1B1F
    1010 1001 0111 0110 1111 1011 1101 1110 <-- 0xA976FBDE
    ---------------------------------------
    0101 1110 1101 1101 1110 0000 1100 0001 <-- 0x5EDDE0C1


andi $t0, $t2, 0x93CB

    1010 1001 0111 0110 1111 1011 1101 1110 <-- 0xA976FBDE
    0000 0000 0000 0000 1001 0011 1100 1011 <-- 0x93CB
    ---------------------------------------
    0000 0000 0000 0000 0000 0011 1100 1010 <-- 0x3CA

------------------------------------------------------------

*** Do the following operations on the 6 bit two's complement numbers, indicate
    if overflow has occurred or not occurred.

     1111
    101111
+   101101
----------
    011100 (overflow)

------------------------------------------------------------

    101010
-   100101
----------
    011010
+        1
----------
    011011
+   101010
----------
    000101 (no overflow)

------------------------------------------------------------

    111
    010101
+   101110
----------
    000011 (no overflow)


    110101
-   100110
----------
    011001
+        1
----------
    011010
+   110101
----------
    001111 (no overflow)

------------------------------------------------------------

*** Multiply the following 6 bit two's complement numbers showing the result as a 12 bit
    numbers. Convert the number to decimal and show you results for the operation in
    decimal. You CAN NOT change either the order or the sign of the numbers. Show your
    work for the decimal! (10 pts)

(a)
    011101
```

```
x   110001
----------

    000000 011101 -> 29
x   111111 110001 -> -15
-----------------

    333211 1
    000000 011101
    000000 00000
    000000 0000
    000000 000
    000111 01
    001110 1
    011101
    11101
    1101
    101
    01
+   1
-----------------
    777643 221101

%2  111001 001101 -> -435 (expected)

Check magnitude to see if it is truly 435:
   111001 001101
   000110 110010
               1
   -------------
   000110 110011 --> 435

------------------------------------------------------------

(b)
    110111
x   101011
----------

    111111 110111 --> -9
x   111111 101011 --> -21
-----------------

    765432 1111
    111111 110111
    111111 10111
    000000 0000
    111110 111
    000000 00
    111011 1
    110111
    10111
    0111
    111
    11
+   1
-----------------
    GECA96 533321

%2  000010 111101 <-- +189

------------------------------------------------------------

(c)
    010110
x   110001
----------

    000000 010110 <-- +22
x   111111 110001 <-- -15
-----------------

    22211
    000000 010110
    000000 00000
    000000 0000
    000000 000
    000101 10
    001011 0
    010110
    10110
    0110
    110
    10
    0
-----------------
    555432 110110
```

```
%2  111010 110110 <-- -154

------------------------------------------------------------

(d)
    110011
x   100011
----------

    111111 110011 <-- -13
x   111111 100011 <-- -29
-----------------

    543321   11
    111111 110011
    111111 10011
    000000 0000
    000000 000
    000000 00
    111001 1
    110011
    10011
    0011
    011
    11
+   1
-----------------
    CA8765 311221

%2  000101 111001 <-- +377
```