QUESTIION #1

Suppose that you wrote the following statements in C/C++:

```c
int my_int_01;
unsigned int my_int_02;

void main (void)
{
    my_int_02 = 63489;
    my_int_01 = my_int_02;
    printf(“my integer: %d\n”, my_int_01);
}
```

Assuming that the computers that run this program is a 16-bit architecture computer (i.e., the registers in the processors are all 16 bits), what number will you see when you print the value of “my_int_01” in the program? Show all your work to find the number (i.e., please explain how you got the number). Note that “%d” in the `printf` statement means “print it as a decimal number”.

\[
63489 = 1111100000000001 \text{ (as unsigned integer)}
\]

\[
my\_int\_01= 1111100000000001
\]

\[
1111100000000001 \text{ (as a two’s complement integer) } = “-2047”
\]

QUESTIION #2

For 16-bit architecture computers, what is the largest positive integer the processors can handle? What is the smallest negative integer the processors can handle?

QUESTIION #3

Transform the following decimal number to the two’s complement binary number (using the 16-bit format: your processor is a 16-bit architecture processor): \(-63_{(10)}\). Show all your work.
**QUESTION #4**

Transform the following decimal number to the two’s complement binary number (using the 32-bit format: your processor is a 32-bit architecture processor): \(-95_{(10)}\). Show all your work.

**Step 1: bit pattern for 95**

\[ 95 = 0101 \ 1111 \]

**Step 2: invert all bits**

\[ 0101 \ 1111 \]
\[ 1010 \ 0000 \]

**Step 3: add binary one to the LSB**

\[ 1010 \ 0000 \]
\[ 1010 \ 0001 \]

\[ 1010 \ 0001 = -95 \text{ (as a 16-bit two’s complement integer)} \]

**Note** 1010 0001 = 161 (as a 16-bit unsigned integer)

**QUESTION #5**

How the binary bit pattern of \(-63_{(10)}\) for a 16-bit processor can be extended for a 32-bit register?