**ACTIVITIES-08:**

1. Implement the following C++ expression in assembly language, using 32-bit signed operands:

val1 = (val2 / val3) \* (val1 % val2);

var4 = (var1 \* -5) / (-var2 % var3);

1. Write ASM instructions that calculate EAX \* 21 using binary multiplication. Hint: 21 = 24 + 22 + 20.
2. Give an assembly language program to move -128 in ax and expand eax. Using shift and rotate instruction.
3. The greatest common divisor (GCD) of two integers is the largest integer that will evenly divide both integers. The GCD algorithm involves integer division in a loop, described by the following C++ code:

int GCD(int x, int y)

{

x = abs(x); // absolute value

y = abs(y);

do

{

int n = x % y;

x = y; y = n;

} while (y > 0);

return x;

}

Implement this function in assembly language and write a test program that calls the function several times, passing it different values. Display all results on the screen.

1. Create a procedure **Extended\_Sbb** procedure to subtract two 64-bit (8-byte) integers.
2. Create a procedure **Extended\_Add** procedure to add two 64-bit (8-byte) integers.
3. Write a program that performs simple encryption by rotating each plaintext byte a varying number of positions in different directions. For example, in the following array that represents the encryption key, a negative value indicates a rotation to the left and a positive value indicates a rotation to the right. The integer in each position indicates the magnitude of the rotation:

key BYTE -2, 4, 1, 0, -3, 5, 2, -4, -4, 6

Your program should loop through a plaintext message and align the key to the first 10 bytes of the message. Rotate each plaintext byte by the amount indicated by its matching key array value.

Then, align the key to the next 10 bytes of the message and repeat the process.

