ACTIVITY:

Task#1

Write ASM instructions that calculate EAX * 21 using binary multiplication.
Hint: $21 = 2^4 + 2^2 + 2^0$.

Task#2

Give an assembly language program to move -128 in ax and expand eax. Using shift and rotate instruction.

Task#3

The time stamp field of a file directory entry uses bits 0 through 4 for the seconds, bits 5 through 10 for the minutes, and bits 11 through 15 for the hours. Write instructions that extract the minutes and copy the value to a byte variable named **bMinutes**.

Task#4

Write a series of instructions that shift the lowest bit of AX into the highest bit of BX without using the SHRD instruction. Next, perform the same operation using SHRD.

Task#5

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

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

Task#6

Create a procedure **Extended_Add** procedure to add two 64-bit (8-byte) integers.

Task#7 Prime Number Program

Write a procedure named IsPrime that sets the Zero flag if the 32-bit integer passed in the EAX register is prime. Optimize the program's loop to run as efficiently as possible. Write a test pro-
gram that prompts the user for an integer, calls IsPrime, and displays a message indicating whether or not the value is prime. Continue prompting the user for integers and calling IsPrime until the user enters 1.
When calling WriteScaled, pass the number's offset in EDX, the number length in ECX, and the decimaloffset in EBX. Write a test program that displays three numbers of different sizes.

Task#8 Encryption Using Rotate Operations

Write a program that performs simple encryption by rotating each plaintext byte a varying num- ber 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, alignthe key to the next 10 bytes of the message and repeat the process.

Task#9 Bitwise Multiplication

Write a procedure named BitwiseMultiply that multiplies any unsigned 32-bit integer by EAX, using only shifting and addition. Pass the integer to the procedure in the EBX register, and return the product in the EAX register. Write a short test program that calls the procedure and displays theproduct. (We will assume that the product is never larger than 32 bits.) This is a fairly challenging program to write. One possible approach is to use a loop to shift the multiplier to the right, keeping track of the number of shifts that occur before the Carry flag is set. The resulting shift count can then be applied to the SHR instruction, using the multiplicand as the destination operand. Then, the same process must be repeated until you find the next highest bit in the multiplier.

Task#10 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.

Task#11 Display ASCII Decimal

Write a procedure named WriteScaled that outputs a decimal ASCII number with an implied deci- mal point. Suppose the following number were defined as follows, where DECIMAL_OFFSET indicates that the decimal point must be inserted five positions from the right side of the number:

```
DECIMAL_OFFSET = 5
.data
decimal_one BYTE "100123456789765"
```

WriteScaled would display the number like this: 1001234567.89765