Programming HW #6 – 100 Pts

Use 32-bit MASM x86 Assembly Language to design a simple integer calculator.

- **a.** The program should have at least 4 menu options entries of a value up to 32 bits in size: (1) Decimal, (2) Hexadecimal, (3) Binary, (4) Exit. If an invalid menu option is chosen the program should display an error message and offer the option to re-enter.
- **b.** Depending on the choice, the program displays the value entered and then converts the value to the other two bases, with the decimal conversion results including the sign
- **c.** If an invalid entry is provided for the chosen base, the program should display an error message and offer the user the option to re-enter the value.
- **d.** After each conversion the program should offer the menu again, until such a time as the Exit option is chosen.

Example Functionality for a positive Decimal Entry:

```
            (1)  Decimal
            (2)  Hexadecimal
            (3)  Binary
            (4)  Exit

  Please choose one of the menu options:  1

  Please enter a 32-bit Decimal integer:  423

  Conversion Results:

    Decimal Value:  423

    Hexadecimal Value: 0000 01A7

    Binary Value: 0000 0000 0000 0000 0000 0001 1010 0111
```

Example Functionality for a negative Decimal Entry:

```
            (1)  Decimal
            (2)  Hexadecimal
            (3)  Binary
            (4)  Exit

  Please choose one of the menu options:  1

  Please enter a 32-bit Decimal integer:  -3059

  Conversion Results:

    Decimal Value: -3059

    Hexadecimal Value: FFFF F40D

    Binary Value: 1111 1111 1111 1111 1111 0100 0000 1101
```

Programming HW #6 – 100 Pts

Example Functionality for a Hexadecimal entry representing a positive integer value:

```
            (1)  Decimal
            (2)  Hexadecimal
            (3)  Binary
            (4)  Exit
 Please choose one of the menu options:  2
 Please enter a 32-bit Hexadecimal Value: 46A3
 Conversion Results:
   Hexadecimal Value:  0000 46A3
   Decimal Value: 18083
   Binary Value: 0000 0000 0000 0000 0100 0110 1010 0011
```

Example Functionality for a Hexadecimal entry representing a negative integer value:

```
            (1)  Decimal
            (2)  Hexadecimal
            (3)  Binary
            (4)  Exit
 Please choose one of the menu options:  2
 Please enter a 32-bit Hexadecimal Value: 9001 234B
 Conversion Results:
   Hexadecimal Value:  9001 234B
   Decimal Value: -1878973621
   Binary Value: 1001 0000 0000 0001 0010 0011 0100 1011
```

Example Functionality for a Binary entry representing a positive integer value:

```
            (1)  Decimal
            (2)  Hexadecimal
            (3)  Binary
            (4)  Exit
 Please choose one of the menu options:  3
 Please enter a 32-bit Binary Value:
 0001 0010 0011 0100 0101 0110 0111 1000
 Conversion Results:
   Binary Value: 0001 0010 0011 0100 0101 0110 0111 1000
   Hexadecimal Value:  1234 5678
   Decimal Value: 305419896
```

Programming HW #6 – 100 Pts

Example Functionality for a Binary entry representing a negative integer value:

```
        (1)  Decimal
        (2)  Hexadecimal
        (3)  Binary
        (4)  Exit
 Please choose one of the menu options:  3
 Please enter a 32-bit Binary Value:
 1000 0111 0110 0101 0100 0011 0010 0001
 Conversion Results:
   Binary Value: 1000 0111 0110 0101 0100 0011 0010 0001
   Hexadecimal Value:  8765 4321
   Decimal Value: -2023406815
```

Note: Spacing between hexadecimal and binary groups of 4 is optional

Do not use magic numbers. Name and initialize variables in the .data section of your programs.

Submit your **source code (.asm)** files to the assignment drop-box by the due date. **Submit a single .asm file.** Do not submit your .sln project file. Do not submit Word or PDF files containing your code.

To ensure consistent grading, please include all necessary INCLUDE and INCLUDELIB statements directly within your code, exactly as directed in the course Syllabus, Policy Contract, and recent announcement. I copy and paste submissions into a standard Visual Studio project for evaluation. Due to the volume of submissions, I'm unable to modify code to accommodate individual Visual Studio configurations. This requirement also helps maintain academic integrity by ensuring only approved files and libraries are used.

Build and debug your code using Visual Studio 2022.

Comment Requirements:
There must be a preamble block of comments at the beginning of the program that includes at least the following information: Your name, course number/section/title, program title, and date

There must be a block of comments before the main procedure of your program, as well as before each additional procedure (if applicable) that describes the inputs, outputs, memory usage, register usage, and functional description of the procedure.

The program must also contain **non-syntax based individual line comments for EACH line** that make the functional goals and processes of the program self-documenting. These comments should be placed in line with each instruction (not above or below). Use tabs to line things up!

Grading

Functionality: 35 Pts
Output Formatting: 15 Pts
Program Structure and Coding Concepts: 25 Pts
Comments and Internal Documentation: 25 Pts