CS 218 – Assignment #6

Purpose: Become familiar with data conversion, addressing modes, and assembly language macro's.
Due: Wednesday (6/15)
Points: 100

Assignment:

Write a simple assembly language program to calculate the areas of a cube for each cube in a series of cubes. However, the data is provided in ASCII/binary[1] format.

Write two macros; one to convert ASCII/binary to decimal and the other to convert decimal to ASCII/binary.

- Write a macro, **aBin2int**, that will convert an ASCII string (up to 32 characters, NULL terminated) representing an unsigned binary value into a double-word sized integer. The macro arguments are starting address of string and integer (for result). *Note*, in order to convert a series of strings, this macro will be called repeatedly by the main.

- Write a macro, **int2aBin**, that will convert an unsigned double-word sized integer into an ASCII string representing the binary value (32 characters). The macro must store the result into an ASCII string (32 characters, NULL terminated).

The ASCII strings are 32 characters followed by a NULL. All data must be treated as *unsigned* integers (i.e., no negative numbers). As such, the MUL and DIV instructions should be used (not the IDIV, IMUL, or CDQ, etc.).

You will be provided a program template that invokes the macros for each set of data. No changes to the macro parameter lists are allowed. Use the provided data sets. You may declare additional variables as needed. Do **not** change the data types (double-words, words, or bytes) as defined.

The provided main will also invoke a provided print string macro, which will display the strings to the screen. The print macro does not perform any error checking, so the data must be correct in order for the display to work. *Note*, since the program displays the results to the screen, typing the program name directly without the debugger (e.g., **`./ast06`**), which will display the results to the screen once the program is working.

1 For more information, refer to: https://en.wikipedia.org/wiki/Binary_number

Assignment

Write an assembly language program to convert ASCII/binary string to integers and to convert integers to ASCII/binary strings. Using the provided main, the program has four steps are follows:

1. Write the code to convert a string of ASCII digits representing a binary value into an integer (double-word sized). This code should be placed in the provided main at the marked location (step #1) and will convert the string **`aLength`** (binary representation) into an integer stored in the variable **`length`**. This should ***not*** be a macro.

2. Convert the code from step #1 into a macro, ***aBin2int***, which is called multiple times in the next part of the provided template. The empty macro shell is at the top of the provided template at the marked location (step #2).

3. Add the code to compute the cube statistics; sum, average, minimum, and maximum. This will read the **cubeAreas[]** array (when populated). This code is similar to the previous assignment. *Note*, you will not be able to test this code until step #4 is completed.

4. Write the code to convert an integer into a string of ASCII digits representing the binary value (NULL terminated). This code should be placed in the provided main at the marked location (step #2) and will convert the integer stored in the variable **`cubeAreasSum`** into a string **`areasSumString`** (ASCII/binary representation). This should ***not*** be a macro.

5. Convert the code from step #2 into a macro, ***int2aBin***, which is called multiple times in the next part of the provided template. The empty macro shell is at the top of the provided template at the marked location (step #3).

Debugging

The code for a macro will not be displayed in the source window. In order to see the macro code, display the machine code window (**View → Machine Code Window**). In the window, the machine code for the instructions are displayed. However, the step and next instructions will execute the entire macro. In order to execute the macro instructions, the **stepi** and **nexti** commands must be used (inside the macro code). The following base conversion web page may be useful during debugging; (http://www.binaryhexconverter.com/binary-to-decimal-converter). Additionally, the Ubuntu default calculator, **calculator**, includes binary and decimal conversion.

Submission:

- All source files must assemble and execute on Ubuntu with **`yasm`**.
- Submit source files
  - Submit a copy of the program source file via the on-line submission
- Once you submit, the system will score the project and provide feedback.
  - If you do not get full score, you can (and should) correct and resubmit.
  - You can re-submit an unlimited number of times before the due date/time.
- Late submissions will be accepted for a period of 24 hours after the due date/time for any given lab. Late submissions will be subject to a ~2% reduction in points per an hour late. If you submit 1 minute - 1 hour late -2%, 1-2 hours late -4%, … , 23-24 hours late -50%. This means after 24 hours late submissions will receive an automatic 0.

Program Header Block

All source files must include your name, section number, assignment, NSHE number, and program description. The required format is as follows:

```
;  Name: <your name>
;  NSHE ID: <your id>
;  Section: <section>
;  Assignment: <assignment number>
;  Description: <short description of program goes here>
```

Failure to include your name in this format will result in a loss of up to 5%.

Scoring Rubric

Scoring will include functionality, code quality, and documentation. Below is a summary of the scoring rubric for this assignment.

Criteria	Weight	Summary
Assemble	-	Failure to assemble will result in a score of 0.
Program Header	5%	Must include header block in the required format (see above).
General Comments	10%	Must include an appropriate level of program documentation. *Note*, ***must*** include comments for the conversion algorithm being used. Omitting these comments will zero the comments score.
Program Functionality (and on-time)	85%	Program must meet the functional requirements as outlined in the assignment. Must be submitted on time for full score.

Example Output:
The results, as displayed to the screen, would be as follows:

```
ed@ed-vm% ./ast6
---------------------------------------------------------
CS 218 - Assignment #6
Cube Area Information

Cube Sides's:
00000000000000101001110101100110   00000000000000000101010010101010
00000000000010101110010110100110   00000000000000000010100101011000
00000000101100000010000001010110   00001010100000100111101101110110
00000000000011000011001001110110   00000010110100011110000110000000
00000101011011100110000000100110   00001001001100000110110011110110
00010011000001010111111000110110   00000011010011100001100000000000
00000011111100011011110100100110   01001111100101000000011110010110
00000001010001010100111010110110   00101111011001010010000110000000
00010100011000010101101000110110   00001001110101101001111000110110
01010101110000001111111000110110   00001001110001110100011001110110
00001110000011100000111000110110

Cube Area's Sum:   01000011000011111110100100001110
Cube Areas's Ave:  00000011001100011000010100000000
Cube Areas's Min:  00000000000000000010100101011000
Cube Areas's Max:  01010101110000001111111000110110
```

Assignment #6 - Data
Refer to the provided main for the provided data sets.

Debugger Commands:
Below is an example of some of the commands to display some of the variables within DDD for assignment #6. You will need to include the rest.

```
x/21s &cSides
x/21uw &cubeAreas
x/uw &cubeAreasSum
x/uw &cubeAreasAve
x/uw &cubeAreasMin
x/uw &cubeAreasMax
```

Note, in DDD, you can select **View → Execution Window** to display a separate window that shows the output (which would normally be displayed to the screen).