CS 218 - Assignment #6

Purpose: Become familiar with data conversion, addressing modes, and assembly language macro's.
Points: 100

Background:

The tridecimal[1] system (also known as triskadecimal or base-13) is a positional notation numeral system using thirteen as its base. It uses 13 different digits for representing numbers. The digits for base 13 could be 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, and C.

The number thirteen (that is, the number written as "13" in the base ten numerical system) is instead written as "10" in tridecimal (meaning "1 thirteen and 0 units", instead of "1 ten and 0 units"), whereas the digit string "15" means "1 thirteen and 5 units" (i.e. the same number that in decimal is written as "18").

The symbol $\mathbf{A}_{13}$ is used for 10_{10}, $\mathbf{B}_{13}$ is used for 11_{10}, and $\mathbf{C}_{12}$ is used for 12_{10}.

He's good on paper, but doesn't seem like someone we can count on.
RESUME
Jiji MATH
Another victim of triskaidekaphobia.

For example.

base-10	1	2	3	4	5	6	7	8	9	10	11	12
base-13	1	2	3	4	5	6	7	8	9	A	B	C

Assignment

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

1. Write the code to convert a string of ASCII digits representing an trideciaml 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 **`dStr1`** (trideciaml representation) into an integer stored in the variable **`iNum1`**.
2. Convert the code from step #1 into a macro, ***tri2int***, which is called multiple times in the next part of the provided main.

You may assume valid/correct data. As such, no error checking is required. You may add additional variables as needed.

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

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 10%.

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	15%	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)	80%	Program must meet the functional requirements as outlined in the assignment. Must be submitted on time for full score.

Debugging
Since macro's can be difficult to debug, the code for step 1 should be working before attempting step 2.

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. 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 (which are only used for macro's).

To help check results, an on-line conversion is available at the following URL:
http://www.cleavebooks.co.uk/scol/calnumba.htm.

Debugger Commands:
Below is an example of some of the commands to display the variables within DDD.

```
x/dw &iNum1
x/5dw &iLst1
```

Note, in DDD, select **View → Execution Window** to display a window that shows the output.