CS 218 – Assignment #4

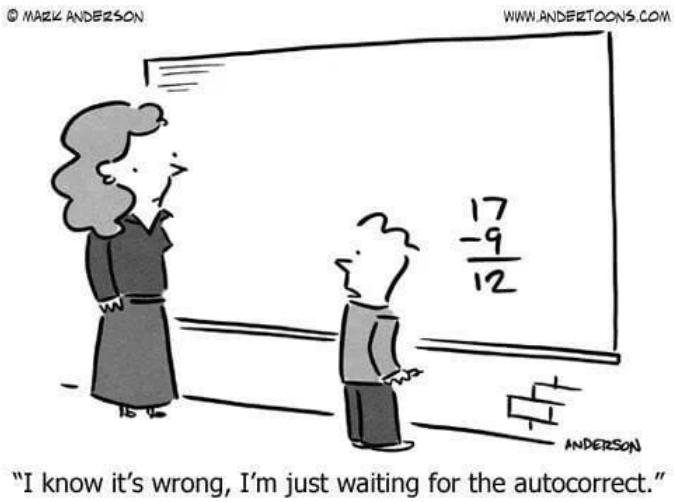
Purpose: Learn to use arithmetic instructions, control instructions, compare instructions, and conditional jump instructions.
Due: Monday (6/13)
Points: 70

Assignment:
Write a simple assembly language program to find the minimum, middle value, maximum, sum, and integer average of a list of numbers. Additionally, the program should also find the sum, count, and integer average for the even numbers. The program should also find the sum, count, and integer average for the numbers that are evenly divisible by 5. Do **not** change the data types (double-words) as defined below. Declare the values:

```
lst         dd     1246,  1116,  1542,  1240,  1677,  1635,  2426,  1820
            dd     1246, -2333,  2317, -1115,  2726,  2140,  2565,  2871
            dd     1614,  2418,  2513,  1422, -2119,  1215, -1525, -1712
            dd     1441, -3622,  -731, -1729,  1615,  1724,  1217, -1224
            dd     1580,  1147,  2324,  1425,  1816,  1262, -2718,  2192
            dd    -1432,  1235,  2764, -1615,  1310,  1765,  1954,  -967
            dd     1515,  3556,  1342,  7321,  1556,  2727,  1227, -1927
            dd     1382,  1465,  3955,  1435, -1225, -2419, -2534, -1345
            dd     2467,  1315,  1961,  1335,  2856,  2553, -1032,  1835
            dd     1464,  1915, -1810,  1465,  1554, -1267,  1615,  1656
            dd     2192, -1825,  1925,  2312,  1725, -2517,  1498, -1677
            dd     1475,  2034,  1223,  1883, -1173,  1350,  1415,   335
            dd     1125,  1118,  1713,  3025
len         dd    100

lstMin      dd    0
lstMid      dd    0
lstMax      dd    0
lstSum      dd    0
lstAve      dd    0

evenCnt     dd    0
evenSum     dd    0
evenAve     dd    0

fiveCnt     dd    0
fiveSum     dd    0
fiveAve     dd    0
```

You may declare additional variables if needed. All data is *signed*. As such, the IDIV/IMUL would be used (not DIV/MUL). The JG/JL/JGE/JLE must be used (as they are for signed data).

Note, for an odd number of items, the middle value is defined as the middle value. For an even number of values, it is the integer average of the two middle values. The 'middle value' does ***not*** require the numbers to be sorted.



"I know it's wrong, I'm just waiting for the autocorrect."

Note, no template is provided. Create the program source file based on the previous assignments.

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.
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.

Debugger Commands:
Due to the looping, when debugging assignment #4, you should learn to set breakpoints within the program.

Create an input file for the debugger. Some useful commands might include:

```
x/100dw &lst
x/dw &len
x/dw &lstMin
x/dw &lstMid
x/dw &lstMax
x/dw &lstSum
x/dw &lstAve
x/dw &evenCnt
x/dw &evenSum
x/dw &evenAve
x/dw &fiveCnt
x/dw &fiveSum
x/dw &fiveAve
```

The commands should be placed in a file (such as 'a4in.txt) so they can be read from within the debugger. The debugger command to read a file is "source <filename>". For example, if the command file is named 'a4in.txt',

```
(gbd) source a4in.txt
```

Based on the above commands, the output will be placed in the file 'a4out.txt'.