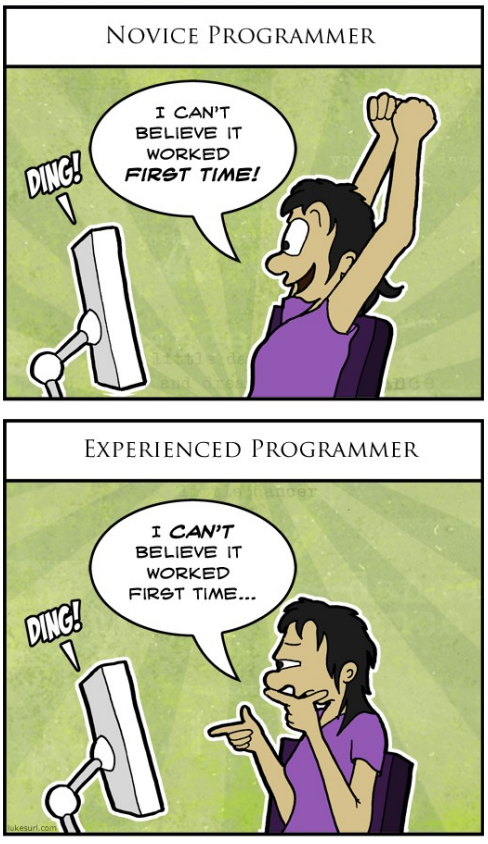
CS 218 – Assignment #7

Purpose: Write a simple assembly language program to sort a list of numbers. Learn to use addressing modes, arithmetic operations, and control instructions.
Points: 120

Assignment:
Write a simple assembly language program to sort a list of integer numbers into ascending (small to large) order. To sort the numbers, use the following Gnome sort[1] algorithm:

```
gnomeSort(a[0..size-1]) {
     i = 1
     j = 2
     while (i < size)
          if (a[i-1] <= a[i])
               i = j
               j = j + 1
          else
               swap a[i-1] and a[i]
               i = i - 1
               if (i = 0)
                    i = 1
}
```

The Gnome Sort is based on the technique used by Dutch gardeners to sort a line of flower pots. Basically, the gardener looks at the flower pot next to him and the previous one; if they are in the right order the gardener steps one pot forward, otherwise he swaps them and steps one pot backwards. Boundary conditions: if there is no previous pot, he steps forward; if there is no pot next to him, he is done.



You ***must*** use the above Gnome sort algorithm (i.e., do **not** use a different sort). *Note*, the algorithm assumes array index's start at 0. If necessary, you can define additional variables. ***Submissions not based on this algorithm will not be scored***.

Additionally, develop three macros as follows:

- ***findSumAve*** - calculate sum and average
- ***findMinMax*** - calculate minimum and maximum
- ***findMid*** - calculate median value

The macros are called in the provided main after the list is sorted (i.e., max=list[len-1] and min=list[0]). *Note,* for an odd number of items, the median value is defined as the middle value. For an even number of values, it is the integer average of the two middle values. The median must be determined *after* the list is sorted.

You are provided a template for this assignment.

1 For more information, refer to: http://en.wikipedia.org/wiki/Gnome_sort

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.
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.

Data Declarations:
Refer to the provide main for the provided data declarations.

```
lst     dd      123,  42, 146,  76, 120,  56, 164,  65, 155,  57
        dd      111, 188,  33,  05,  27, 101, 115, 108,  13, 115
        dd       17,  26, 129, 117, 107, 105, 109,  30, 150,  14
        dd      147, 123,  45,  40,  65,  11,  54,  28,  13,  22
        dd       69,  26,  71, 147,  28,  27,  90, 177,  75,  14
        dd      181,  25,  15,  22,  17,   1,  10, 129,  12, 134
        dd       61,  34, 151,  32,  12,  29, 114,  22,  13, 131
        dd      127,  64,  40, 172,  24, 125,  16,  62,   8,  92
        dd      111, 183, 133,  50,   2,  19,  15,  18, 113,  15
        dd       29, 126,  62,  17, 127,  77,  89,  79,  75,  14
        dd      114,  25,  84,  43,  76, 134,  26, 100,  56,  63
        dd       24,  16,  17, 183,  12,  81, 320,  67,  59, 190
        dd      193, 132, 146, 186, 191, 186, 134, 125,  15,  76
        dd       67, 183,   7, 114,  15,  11,  24, 128, 113, 112
        dd       24,  16,  17, 183,  12, 121, 320,  40,  19,  90
        dd      135, 126, 122, 117, 127,  27,  19, 127, 125, 184
        dd       97,  74, 190,   3,  24, 125, 116, 126,   4,  29
        dd      104, 124, 112, 143, 176,  34, 126, 112, 156, 103
        dd       69,  26,  71, 147,  28,  27,  39, 177,  75,  14
        dd      153, 172, 146, 176, 170, 156, 164, 165, 155, 156
        dd       94,  25,  84,  43,  76,  34,  26,  13,  56,  63
        dd      147, 153, 143, 140, 165, 191, 154, 168, 143, 162
        dd       11,  83, 133,  50,  25,  21,  15,  88,  13,  15
        dd      169, 146, 162, 147, 157, 167, 169, 177, 175, 144
        dd       27,  64,  30, 172,  24,  25,  16,  62,  28,  92
        dd      181, 155, 145, 132, 167, 185, 150, 149, 182,  34
        dd       81,  25,  15,   9,  17,  25,  37, 129,  12, 134
        dd      177, 164, 160, 172, 184, 175, 166,  62, 158,  72
        dd       61,  83, 133, 150, 135,  31, 185, 178, 197, 185
        dd      147, 123,  45,  40,  66,  11,  54,  28,  13,  22
        dd       49,   6, 162, 167, 167, 177, 169, 177, 175, 164
        dd      161, 122, 151,  32,  70,  29,  14,  22,  13, 131
        dd       84, 179, 117, 183, 190, 100, 112, 123, 122, 131
        dd      123,  42, 146,  76,  20,  56,  64,  66, 155,  57
        dd       39, 126,  62,  41, 127,  77, 199,  79, 175,  14
len     dd      350

min     dd      0
med     dd      0
max     dd      0
sum     dd      0
avg     dd      0
```

As necessary, you can define additional variables.

Debugging Tips

- Use comments!!
- Follow the algorithm directly (do not attempt to optimize).
- Comment each part of the algorithm (so you can match the algorithm to the appropriate subset of code).
- Develop a debugger input file first (based on previous ones) carefully verifying the debugger commands based on the specific data types/sizes.
- You can temporarily change the array length to a smaller number (i.e., 5-10) for testing.