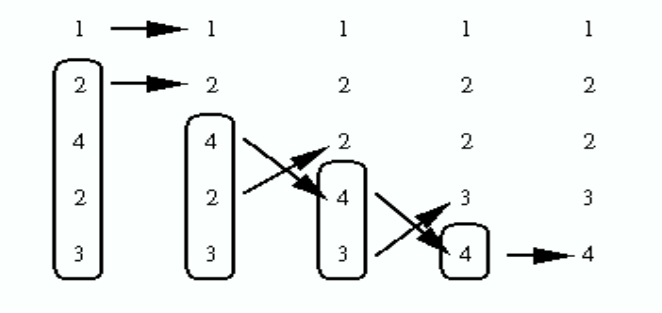
CS 218 – Assignment #7

Purpose:	Write a simple assembly language program to sort a list of numbers. Learn to use addressing modes, arithmetic operations, and control instructions.
Due:	Thursday (6/20)
Points:	100

Assignment:

Write a simple assembly language program to sort a list of signed integer numbers into ascending (small to large) order. Additionally, find the minimum, median, maximum, sum, and average of the list. You can find the minimum and maximum after the list is sorted (i.e., min=list[0] and max=list[len-1]). 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 should write the code for both even and odd length lists as it will be used in the next assignment.



To sort the numbers, use the following bubble sort[1] algorithm:

```
for ( i = (len-1) to 0 ) {
    swapped = false
    for ( j = 0 to i-1 )
        if ( lst(j) > lst(j+1) ) {
            tmp = lst(j)
            lst(j) = lst(j+1)
            lst(j+1) = tmp
            swapped = true
        }
    if ( swapped = false ) exit
}
```

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

All data must be treated as ***signed*** integers (i.e., negative numbers). As such, the IMUL, IDIV, and CDQ instructions should be used (not the DIV and/or MUL). Do not change the provided data types/sizes.

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

Data Declarations:
Refer to the provide main for the provided data declarations.
As necessary, you can define additional variables.

Integer to Binary Macro:
This assignment uses the integer to binary conversion macro from assignment #6. The provided main includes a place to cut-and-paste the code from the assignment #6 macro into the assignment #7 template. The macro is used, along with the provided print string macro, to display output to the screen (as shown below).

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

```
ed@ed-vm% ./ast7
**********************************
CS 218 - Assignment #7

List Statistics:

List Minimum: 11111111111111111101110001011011
List Median:  00000000000000000000010001110110
List Maximum: 00000000000000000010000110101011
List Sum:     00000000000001010100101011011011
List Average: 00000000000000000000011011000110
```

Note, since this program displays output to the screen, it can be executed without the debugger.

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.
- You can temporarily change the array length to a smaller number (i.e., 5-10) for testing.