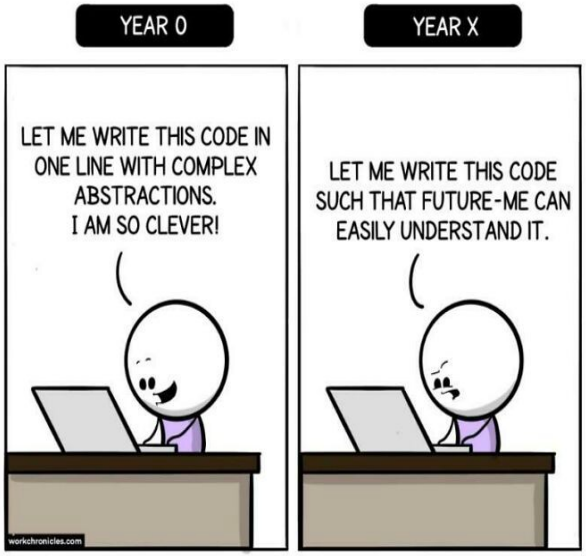
**CS 218 – Assignment #7**

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. Additionally, find the minimum, median, maximum, sum, and average of the list. You should 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.




An Odd/EvenSort[1] is a relatively simple sorting algorithm, developed originally for use on parallel processors. To sort the numbers, use the following Odd/Even Sort algorithm:

```
function oddEvenSort(list) {
     bool sorted = false;
     while (!sorted) {
         sorted = true;
         for (var i=1; i < len-1; i+=2) {
                  if (list[i] > list[i+1]) {
                           swap(list, i, i+1);
                           sorted = false;
                  }
         }
         for (var i=0; i < len-1; i+=2) {
                  if (list[i] > list[i+1]) {
                           swap(list, i, i+1);
                           sorted = false;
                  }
         }
     }
}
```

You ***must*** use the above Odd/Even 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 ***unsigned*** integers (i.e., only positive numbers). As such, the MUL and DIV instructions should be used (not the IDIV and/or IMUL).

1 For more information, refer to: https://en.wikipedia.org/wiki/Odd-even_sort

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

**Integer to Senary Macro**
This assignment uses the integer to quaternary conversion macro, ***int2aSenary***, 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% ./ast07

---------------------------------------------------
CS 218 - Assignment #7
Odd/Even Sort

Minimum:               20
Maximum:           424005
Median:              5133
Sum:             15254352
Average:            20225
```

*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.

**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 assignment. 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: <4-digit-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 3%.

**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. |
| Correct Sort Algorithm | - | Failure to use the provided sort algorithm will result in a score of 0. |
| Program Header | 3% | Must include header block in the required format (see above). |
| General Comments | 7% | Must include an appropriate level of program documentation. |
| Program Functionality (and on-time) | 90% | Program must meet the functional requirements as outlined in the assignment. Must be submitted on time for full score. |