CS 218 – Assignment #7

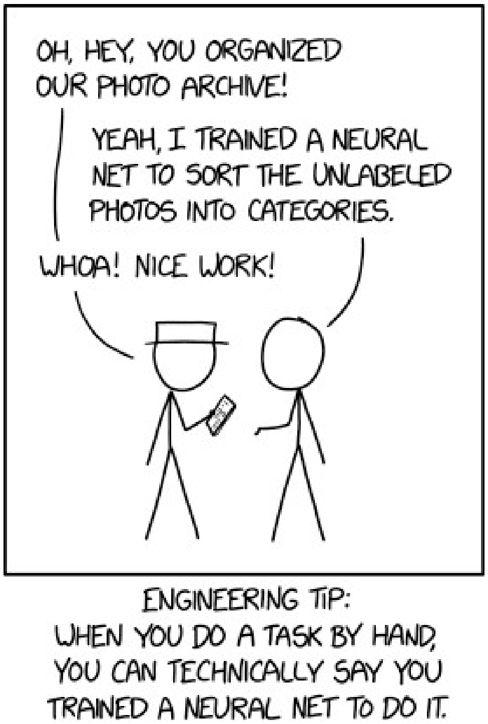
Purpose: Write a simple assembly language program to sort a list of numbers. Learn to use addressing modes, arithmetic operations, and control instructions.
Points: 120

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 Shell sort[1] algorithm:

```
h = 1;
while ( (h*3+1) < length ) {
   h = 3 * h + 1;
}

while ( h>0 ) {
   for (i = h-1; i < length; i++) {
        tmp = lst[i];
        j = i;
        for ( j=i; (j >= h) && (lst[j-h] >
                                 tmp); j = j-h) {
           lst[j] = lst[j-h];
        }
        lst[j] = tmp;
   }
   h = h / 3;
}
```


OH, HEY, YOU ORGANIZED OUR PHOTO ARCHIVE!
YEAH, I TRAINED A NEURAL NET TO SORT THE UNLABELED PHOTOS INTO CATEGORIES.
WHOA! NICE WORK!
ENGINEERING TIP: WHEN YOU DO A TASK BY HAND, YOU CAN TECHNICALLY SAY YOU TRAINED A NEURAL NET TO DO IT.

Source: www.xkcd.com/2173

You ***must*** use the above shell 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.

You are provided a template for this assignment.

Data Declarations:

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

Integer to Septenary Macro:

This assignment uses the integer to septenary conversion macro, ***int2aSept***, 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).

[1] For more information, refer to: http://en.wikipedia.org/wiki/Shell_sort

Example Output:

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

```
ed@ed-vm% ./ast07
------------------------------------------------------
CS 218 - Assignment #7
Shell Sort

Minimum:           +1
Median:         +1634
Maximum:       +41103
Sum:         +3020225
Average:        +5134
```

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 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 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.
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.