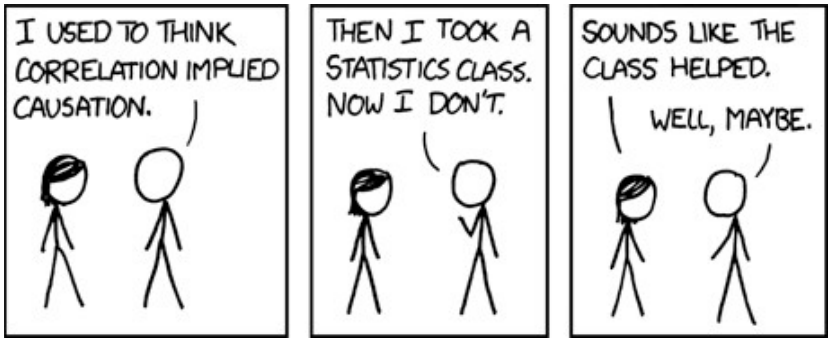
CS 218 – Homework, Asst. #8

Purpose: Learn assembly language functions. Additionally, become more familiar with program control instructions, function handling, and stacks.
Points: 125

Assignment:
Write four simple assembly language functions described below. You will be provided a main function that calls the following functions (for each set of data).



Source: www.xkcd.com/552

- Write a void function, **gnomeSort()**, to sort the numbers into descending order (large to small). You **must** use the gnome Sort algorithm (from assignment 7) and modify the sort order.

- Write a void function, **basicStats()**, to find the minimum, median, maximum, sum, and average for a list of numbers. This function must call the **listMedian()** function tin find the median.

- Write a value returning function, **listMedian()**, to find the median value of a sorted list of numbers. *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. This function should return a result in the **eax** register in accordance with the standard calling convention.

- Write a value returning function, **corrCoefficient()**, to compute the correlation coefficient[1] for the two data sets. The correlation coefficient formula is as follows:

$$r = \frac{\left(\sum_{i=0}^{n-1} x_i y_i\right)}{\sqrt{\left(\sum_{i=0}^{n-1} x_i^2\right)\left(\sum_{i=0}^{n-1} y_i^2\right)}}$$

  Where ∑ is the summation of the values, and ***n*** is the count of points (length). This function uses the provided square root code function (next page). This function should return a floating point result in the **xmm0** register in accordance with the standard calling convention.

All data should be treated as *unsigned* integers (MUL, and DIV). The functions must be in a separate assembly file (**`ast8procs.asm`**). The files will be assembled individually and linked together.

1 For more information, refer to: http://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient

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.
  - Only the functions file (**`ast8procs.asm`**) will be submitted.
- 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.

Floating Point Calculations

The following code will perform the floating point division by converting the double-word integers in **rax** and **r12** into floating point values in **xmm1** and **xmm0**.

```
cvtsi2sd    xmm0, r12              ; x's * y's
cvtsi2sd    xmm1, rax              ; x's^2 * y's^2
sqrtsd      xmm1, xmm1             ; sqrt()
divsd       xmm0, xmm1             ; xmm0 = xmm0 / xmm1
```

The result is left in **xmm0** as required by the standard calling convention.

Compile, Assemble, and Linking Instructions

You will be provided a main function that calls the functions. Your functions should be in a separate file. The files will be assembled individually and linked together.

When compiling, assembling, and linking the files for assignment #8, use the provided **makefile** to assemble, and link. *Note*, **only** the functions file, **`ast8procs.asm`**, will be submitted. The submitted functions file will be assembled and linked with the provided main. As such, do not alter the provided main.

Provided Data Sets:

Do not change the data types of the provided data. You may define additional variables as required.

```
xList1      dd     121,   27,   10,   22,   61
            dd      15,   12,  120,   19,   20
            dd      20,   11,   12
yList1      dd    1230, 1233, 1323, 1241, 1360
            dd    1290, 1118, 1250, 1475, 1423
            dd    1210, 1337, 1226
len1        dd    13
```

The results for data set #1 are shown for reference:

```
xMin1:    0x6010a4:     10
xMed1:    0x6010a8:     20
xMax1:    0x6010ac:     121
xSum1:    0x6010b0:     470
xAve1:    0x6010b4:     36

yMin1:    0x6010b8:     1118
yMed1:    0x6010bc:     1250
yMax1:    0x6010c0:     1475
ySum1:    0x6010c4:     16716
yAve1:    0x6010c8:     1285

r1:       0x6010cc:     0.73181607482401867
```