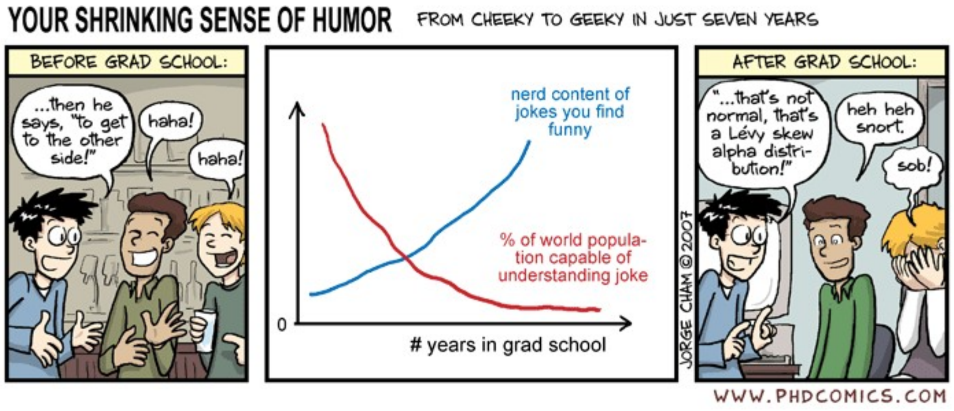
CS 218 – Assignment #9

Purpose: Learn assembly language functions and standard calling convention. Additionally, become more familiar with program control instructions, high-level language function interfacing.

Points: 175

Assignment:

Write the assembly language program to read some data and calculate the kurtosis[1] statistic. The functions are described below. You will be provided a C++ main program that calls the functions.


YOUR SHRINKING SENSE OF HUMOR
FROM CHEEKY TO GEEKY IN JUST SEVEN YEARS
BEFORE GRAD SCHOOL:
...then he says, "to get to the other side!"
haha!
haha!
nerd content of jokes you find funny
% of world population capable of understanding joke
0
# years in grad school
JORGE CHAM © 2007
AFTER GRAD SCHOOL:
"...that's not normal, that's a Lévy skew alpha distribution!"
heh heh snort.
sob!
WWW.PHDCOMICS.COM

- Write a value returning integer function, **lstEstMedian()**, to return the estimated median of a list of numbers (prior to sorting as done in assignments #4 and #5). The estimated the median is computed by summing the first, last, and two middle values and dividing by 4 for even length lists and 3 for odd length lists. The integer function returns the double-word value in ***eax***.
- Write a void function, **combSort()**, to sort the numbers into descending order (large to small). You **must** use the comb sort algorithm (from assignment 7). You will need to again modify the algorithm to change the sort order (from descending to ascending).
- Write a value returning function, **lstSum()**, to find the sum for a list of numbers.
- Write a value returning function, **lstAverage()**, to find the average for a list of numbers. The function must call the **lstSum()** function to find the sum of the passed list.
- Write a value returning function, **lstMedian()**, to find the statistical median for a sorted list of numbers. The function should assume the list is sorted. For an odd number of items, the statistical median value is defined as the middle value. For an even number of values, it is the integer average of the two middle values.
- Write a void function, **lstStats()**, to find the sum, average, minimum, maximum, and median for a list of numbers. The function must call the **lstSum()**, **lstMedian()**, and the **lstAverage()** functions to find the median and average of the passed list.
- Write a value returning function, **lstKurtosis()**, to compute the kurtosis statistic for the data set. *Note*, must perform the summation for the dividend (top) as a quad value. If the divisor (bottom) is 0, do not perform the divide and the final result should be set to 0.
- Write a value returning function, **readNonaryNum()**, that will read a signed ASCII nonary number from the user. The routine should use the system service for reading data from STDIN (into a buffer) and perform the conversion (with the input from the buffer). Leading spaces should be ignored. The sign (+/-) is required. If the conversion is successful and the value is ≥ MINNUM and ≤ MAXNUM, the function should return true and the number (via reference). If the number is not valid or out of range the function should display the appropriate error message (which are passed) and re-prompt for input. If the input line is too long (> 50 characters), the

1 For more information, refer to: https://en.wikipedia.org/wiki/Kurtosis

input should be ignored, and error message displayed, and the function should re-prompt for input. If no input (i.e., user typed enter and nothing else), the function should return false.

All functions must use the stack for the storage of local variables. No static variables should be declared. All data items are ***signed*** integers (IMUL and IDIV instructions). The functions must be in a separate assembly file. The files will be compiled/assembled individually and linked together.

Assemble and Linking Instructions

You will be provided a main function (`main.cpp`) that calls the functions. Your functions should be in a separate file (`ast9procs.asm`). The files will be assembled individually and linked together.

When assembling, and linking the files for assignment #8, use the provided **makefile** to assemble, and link. *Note*, **only** the functions file, `ast9procs.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.

Example Execution

The following is an example execution demonstrating the error handling:

```
ed-vm% ./main
-------------------------------------------------
CS 218 - Assignment #9

Enter Value (nonary): -110
Enter Value (nonary): +212
Enter Value (nonary): -181
Enter Value (nonary): +162152
Error, out of range. Please re-enter.
Enter Value (nonary): +782
Enter Value (nonary):     +000000321
Enter Value (nonary): +901
Error, invalid. Please re-enter.
Enter Value (nonary):          +871
Enter Value (nonary): none
Error, invalid. Please re-enter
Enter Value (nonary):     -zero
Error, invalid. Please re-enter
Enter Value (nonary):
+00000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000
Error, input too long. Please re-enter.
Enter Value (nonary):    +00000000871
Enter Value (nonary): 000+11
Error, invalid. Please re-enter.
Enter Value (nonary):   +658
Enter Value (nonary):

-------------------------------------------------
Program Results

Sorted List:
-154  -90  173  262  539  641  712  712

      Sum =          2795
  Average =           349
  Minimum =          -154
  Maximum =           712
   Median =           400
 Kurtosis =        167366
ed-vm%
```

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 (at a maximum rate of 5 submissions per hour).
- 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: <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.