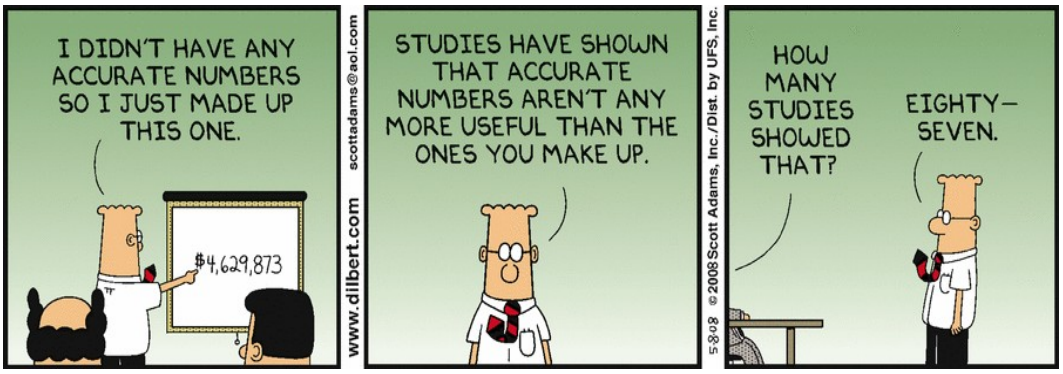
CS 218 – Assignment #9

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 perform a linear regression. The functions are described below. You will be provided a C++ main program that uses the functions.



- Void function, **shellSort()**, to sort the numbers into ascending order (small to large). You **must** use the shell sort algorithm and modify the sort order.
- Value returning function, **lstSum()**, to find the sum for a list of numbers.
- Value returning function, **lstAverage()**, to find the average for a list of numbers. *Note*, this function must call the **lstSum()** function.
- Void function, **basicStats()**, to find the minimum, median, maximum, sum, and average for a list of numbers. The function must call the **lstSum()** and **lstAverage()** functions to find the sum and average of the passed list.
- Value returning function, **linearRegression()**, to compute the linear regression values (b_0 and b_1) for the data sets.
- Void returning function, **readSeptNum()**, that will read an signed ASCII septenary number from the user. The routine must use the system service for reading data from STDIN one characters a a time (into a buffer) and perform the conversion (with the input from the buffer). Leading spaces should be ignored. If the value is less than or greater than the defined minimum and maximum constants, or if the input line is too long (> 50 characters), the input should be ignored and the function should return the appropriate status code. The error strings are provided. The function must return one of the following status codes:
  - SUCCESS (0) → Successful conversion and number within required range.
  - NOSUCCESS (1) → Invalid input entered (i.e., not a valid septenary number).
  - OUTOFRANGEMIN(2) → Valid septenary number, but below minimum value.
  - OUTOFRANGEMAX (3) → Valid septenary number, but above maximum value.
  - INPUTOVERFLOW (4) → User entry character count exceeds maximum length.
  - ENDOFINPUT (5) → Return entered, no characters (for end of the input).

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 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 X Value (+/-septenary): +10
Enter Y Value (+/-septenary): +20
Enter X Value (+/-septenary): 20
Error, invalid number. Please re-enter.
Enter X Value (+/-septenary): +2Q
Error, invalid number. Please re-enter.
Enter X Value (+/-septenary): +30
Enter Y Value (+/-septenary):            +40
Enter X Value (+/-septenary):   +0000000050
Enter Y Value (+/-septenary): +9
Error, invalid number. Please re-enter.
Enter Y Value (+/-septenary): ++10
Error, invalid number. Please re-enter.
Enter Y Value (+/-septenary): +2 00
Error, invalid number. Please re-enter.
Enter Y Value (+/-septenary): +20000
Enter X Value (+/-septenary): +10000000
Error, number above maximum value. Please re-enter.
Enter X Value (+/-septenary): -29
Error, invalid number. Please re-enter.
Enter X Value (+/-septenary): -20
Enter Y Value (+/-septenary): +333333333333333333333333333333333333333333333333333333333333333333333333
Error, user input exceeded length, input ignored. Please re-enter.
Enter Y Value (+/-septenary): +50000
Enter X Value (+/-septenary):
------------------------------------------------

Program Results

Sorted X List:
-14  7  21  35

   X Minimum =           -14
    X Median =            14
   X Maximum =            35
       X Sum =            49
   X Average =            12

Sorted Y List:
14  28  4802  12005

   Y Minimum =            14
    Y Median =          2415
   Y Maximum =         12005
       Y Sum =         16849
   Y Average =          4212

Linear Regression Values
 b0 =  1344
 b1 =  239

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.