CARLETON UNIVERSITY
School of Computer Science

COMP 2003 Computer Organization

Fall 2005

Assignment 4

Due date: November 21, 5 PM (MUST be submitted through the Raven system)
Paper assignments will NOT be accepted.

Note: For your convenience, the marking scheme for this assignment is included.

Write a stand-alone assembly language program that reads a set of positive, 32-bit integers (i.e., value ≥ 0) from the user and reports the following statistics: *minimum*, *maximum*, and *average*. The average statistic is truncated i.e., it ignores the fractional part.

For example, the input values

```
2, 5, 6, 4, 7, 1, 8, 3, 9, 10
```

should produce the following output:

```
Minimum = 1
Maximum = 10
Average = 5
```

You should structure your program as follows:

`main:` The `main` procedure is responsible for the user interface. That is, it prompts the user for the input data and outputs the statistics. After prompting the user for input, it calls the `read_input` procedure to read the input numbers. The main procedure then calls a procedure (say `min_max`) to compute the minimum and maximum statistics. Then the main procedure calls another procedure `average` to compute the truncated average of the input data. Finally, it (`main`) outputs the statistics: *minimum*, *maximum*, and *average*. Write your program such that it can accept at most 20 input values into an array. The user, however, can terminate the input earlier by entering a negative value.

`read_input:` This procedure receives a pointer to an array and its *maximum size*. Note that the maximum size is the amount space in the array in terms of number of integers it can hold. The `read_input` procedure stores the values read from the user and returns the number of values read to `main` via the AX register.

`min_max:` The `min_max` procedure receives a pointer to an array, its size (i.e., number of values in the array), pointers to variables `minimum` and `maximum`. The `min_max` procedure returns the minimum and maximum via `minimum` and `maximum`.

`average:` To compute the average, `main` calls the `average` procedure and passes a pointer to the array and its size. The average value is returned to `main` via the EAX register. Hint: Use the **`div`** instruction to compute the truncated average.

You should not use any global variables. Also, you don't have to worry about error checking in this assignment.

You **must** keep each procedure in a separate file. That is, you will have four .ASM source files.

Note: Your program should **strictly follow** the specifications given here. **Any deviation from the given specifications will incur a 25% penalty.**

Note: Make sure to submit enough evidence that indicates that you have tested your program thoroughly. In particular, identify the border cases and use them to test your program.

You should submit the following three parts:

(1) Your source code

(2) *Test plan and test data output:* For each test case, provide the objective (why you are doing that particular test) and the rationale for selecting the test input. **Zero marks will be given to this part if your assignment provides testing information without this information (i.e., objective and rationale).** In particular, identify the border cases and test them.

(3) Output of your program for the following input values:

Test case 1: `5,2,25,0,25,0,120,30`

Test case 2: `1,2,3,4,5,70,400`

Marking scheme for this assignment (25 marks):

Correctness of the program:	12 marks (if your programs works correctly and your documentation convinces the TA – don't expect marks if you just submit a print out of the source code)
Structure and comments:	4 marks (spacing, indentation, comments, variable names etc.)
Good, clean, and efficient solution:	4 marks (weird or convoluted solutions will be heavily penalized)
Test plans & test data output:	4 marks (zero marks if no objective or rationale as mentioned above)
Output with the test data (given above):	1 mark