CS 271 Computer Architecture and Assembly Language
Programming Assignment #5

Objectives:

1. using register indirect addressing
2. passing parameters
3. generating "random" numbers
4. working with arrays

Description:

Write and test a *MASM* program to perform the following tasks:

1. Introduce the program.
2. Generate *ARRAYSIZE* random integers in the range [*LO* = 10 .. *HI* = 29], storing them in consecutive elements of an *array*. *ARRAYSIZE* should be set to 200.
   HINT: Use Irvine's "RandomRange" and "Randomize" and code from the textbook/slides to generate each random number.
3. Display the list of integers before sorting, 20 numbers per line with two spaces between each value.
4. Sort the list in ascending order (i.e., smallest first). Do not use any textbook-provided sorts.
5. Calculate and display the median value, rounded to the nearest integer.
6. Display the sorted list, 20 numbers per line with two spaces between each value.
7. Generate an array *counts* which holds the number of times each value [10, 29] is seen in *array*.
   For example, *counts*[0] should equal the number instances of the value '10' in *array*. *counts[14]* should equal the number of instances of the value '24' in *array*.
8. Display the array *counts*, 20 numbers per line with two spaces between each value.

Requirements:

1. The title, programmer's name, and brief instructions must be displayed on the screen.
2. *LO, HI,*and *ARRAYSIZE* must be declared and used as global constants.
3. Strings must be passed by reference.
4. The program must be constructed using procedures. At least the following procedures are required:
   A. main
   B. introduction {parameters: *string1* (reference), *string2* (reference), ..)
   C. fillArray {parameters: *array* (reference), *LO* (value), *HI* (value), *ARRAYSIZE* (value)}
   D. sortList {parameters: *array* (reference), *ARRAYSIZE* (value)}
      i. exchangeElements (for most sorting algorithms): {parameters: *array*[i] (reference), *array*[j] (reference), where *i* and *j* are the indexes of elements to be exchanged}
   E. displayMedian {parameters: *array* (reference), *ARRAYSIZE* (value)}
   F. displayList {parameters: *someArray* (reference), *ARRAYSIZE* (value), *someTitle* (reference)}
   G. countList {parameters: *array* (reference), *ARRAYSIZE* (value), *list* (reference), *LO* (value), …}
5. Parameters must be passed on the system stack by value or by reference as noted above.
6. There must be just one procedure to display arrays. This procedure must be called three times: once to display the unsorted *array*, once to display the sorted *array*, and once to display *counts*.
7. All lists must be identified when they are displayed (use the *title* parameter for the *displayList* procedure).
8. Procedures (except main) should not reference .data segment variables by name. *request*, *array*, *counts*, and titles for the sorted/unsorted lists should be declared in the .data segment, but procedures must have them passed by reference. Procedures may use local variables when appropriate. Global constants must only be used in main.
9. The program must use *register indirect addressing* for array elements.
10. The program must be fully documented. This includes a complete header block for the program (in your own words) and for each procedure, and a comment outline to explain each section of code.
11. The code and the output must be well-formatted.
12. Submit your text code file (.asm) to Canvas by the due date.

Extra Credit (Be sure to describe your extras in the program header block)**:**

1. Display the numbers ordered by column instead of by row. (1pt)
2. Derive *counts* before sorting *array*, then use *counts* to sort *array*. For this EC option you will print the unsorted list, then the 'counts' array, then the median of the sorted 'list' and finally the sorted 'list'. (2pts)
3. Generate the numbers into a file; then read the file into the array. (3pts)

To ensure you receive credit for any extra credit options you did, you must add one print statement to your program output PER EXTRA CREDIT which describes the extra credit you chose to work on. You will not receive extra credit points unless you do this. The statement must be formatted as follows...

```
--Program Intro--
**EC: DESCRIPTION

--Program prompts, etc—
```

Notes:

1. The Irvine library provides procedures for generating random numbers. Call *Randomize* once at the beginning of the program (to set up so you don't get the same sequence every time), and call *RandomRange* to get a pseudo-random number. (See the documentation in Lecture slides.)
2. The median is calculated after the array is sorted. It is the "middle" element of the sorted list. If the number of elements is even, the median is the average of the middle two elements (may be rounded).

Example:

```
Sorting and Counting Random integers!                    Programmed by Stephen
This program generates 200 random numbers in the range [10 ... 29], displays the
original list, sorts the list, displays the median value, displays the list sorted in
ascending order, then displays the number of instances of each generated value.

Your unsorted random numbers:
11  24  18  19  26  10  19  15  11  20  16  21  21  19  24  18  12  10  19  20
12  13  13  21  13  21  15  20  28  24  28  14  16  20  28  21  29  24  11  19
24  13  22  19  20  10  12  13  13  15  13  26  27  24  17  17  27  21  21  18
15  15  24  25  13  17  26  27  29  27  27  15  29  18  20  21  23  20  12  13
22  15  19  20  22  18  13  16  20  22  15  23  18  22  12  29  18  18  12  24
20  14  16  14  18  18  14  22  18  11  12  21  15  16  21  21  29  13  26  16
17  15  13  21  10  20  12  24  28  10  18  18  19  28  20  26  23  15  16  16
13  22  28  24  24  12  23  15  24  12  18  24  22  15  19  23  13  27  16  24
29  15  24  28  18  23  28  21  16  23  22  10  24  12  20  16  25  17  25  12
23  22  18  15  23  16  13  24  12  13  24  24  14  15  19  13  25  18  28  13

List Median: 19
Your sorted random numbers:
10  10  10  10  10  10  11  11  11  11  12  12  12  12  12  12  12  12  12  12
12  12  12  13  13  13  13  13  13  13  13  13  13  13  13  13  13  13  13  13
13  14  14  14  14  14  15  15  15  15  15  15  15  15  15  15  15  15  15  15
15  15  16  16  16  16  16  16  16  16  16  16  16  16  16  17  17  17  17  17  18
18  18  18  18  18  18  18  18  18  18  18  18  18  18  18  18  19  19  19  19
19  19  19  19  19  19  20  20  20  20  20  20  20  20  20  20  20  20  20  21
21  21  21  21  21  21  21  21  21  21  21  21  22  22  22  22  22  22  22  22
22  22  23  23  23  23  23  23  23  23  23  24  24  24  24  24  24  24  24  24
24  24  24  24  24  24  24  24  24  24  25  25  25  25  26  26  26  26  26  27
27  27  27  27  27  28  28  28  28  28  28  28  28  28  29  29  29  29  29  29

Your list of instances of each generated number, starting with the number of 10s:
6  4  13  18  5  16  12  5  17  10  13  13  10  9  19  4  5  6  9  6


Goodbye, and thanks for using my program!
```