**Andrew Tran**

**Computer Science 2253 Lab Experiment Ten**

**Objectives:**

1. Utilizing the irvine32 library.

2. Understanding the Loop instruction in Assembler.

3. Utilizing the cmp and conditional jump instructions.

**What you are to do:**

The selection sort coded in C++ is given below:

void selectionSort(int numbers[], int array\_size){

int i, j;

int min, temp;

for (i = 0; i < array\_size; i++){

min = i;

for (j = i+1; j < array\_size; j++)

if (numbers[j] < numbers[min])

min = j;

temp = numbers[i];

numbers[i] = numbers[min];

numbers[min] = temp;

}// end for

}// end sort

Rewrite the above code in assembler utilizing the instructions: loop, cmp, and the conditional jumps. You may not use any **.directives** in the solution of the above problem. You may assume the **maximum** size of the array will be 50 dwords (Note: The actual number of elements in the array can be less than 50.)

To simulate a user entered list you will need to do the following:

1. Generate a random number from 10 to 50. This value will be used to determine the actual number of entries the user would have entered into the list.
2. Fill the array with random numbers from 1 to 1000 according to the number of entries specified by part a above. This should be accomplished in a procedure.

**Note: You will not solicit any sort of input from the user.**

The selection sort should be a separate procedure where the address of the array to be sorted is passed via register and the actual number of elements in the array will also be passed via a register.

**Write a procedure to display the contents of the array**. This procedure should be called prior to calling the selection sort and again after the selection sort has been called. This should be done to verify the selection sort worked.

Copy your assembler code to a word document and capture the output window and paste into a word document directly below your code. You should execute your program 3 times and have 3 output windows pasted into the word document.

**What to hand in:**

1. Print out of the assembler program.
2. Print out of the word document.
3. Place the assembler file into the assignment folder titled Lab Ten.

**Due Date: As indicated on the assignment folder.**

INCLUDE Irvine32.inc

.data

array BYTE 50 DUP(?)

elements dword ? ;number of elements in the array

text1 BYTE "Array: ",0

text2 BYTE "Sorted: "

.code

main PROC

call Randomize ;determine no. of elements in arr

mov eax, 51

call RandomRange

mov DWORD PTR elements, eax

add elements, 10

call fill

mov edx, offset text1 ;unsorted display

call WriteString

mov ecx, offset array

call display

mov edi, offset array

mov ecx, elements

call sort

mov edx, offset text2 ;sorted display

call WriteString

mov ecx, offset array

call display

exit

main endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

fill PROC ;fill the array with random no.

mov eax, 0

mov ecx, elements

mov esi, 0

random:

mov ah, BYTE PTR 1000

call RandomRange

add eax, 1

mov array[esi], ah

inc esi

loop random

ret

fill ENDP

;;;;;;;;;;;;;;;;;;;;;;;;;

display PROC ;display the current array

mov eax, 0

mov ecx, elements

mov esi, 0

displayit:

mov al, array[esi]

call WriteInt

inc esi

loop displayit

call Crlf

ret

display ENDP

;;;;;;;;;;;;;;;;;;;;;;;;

sort PROC ;time to sort em'

dec ecx ;(takes) edi=offset arr, ecx=elements

mov ebx, edi

mov edx, ecx

outer:

mov edi, ebx

mov esi, edi

inc esi

push ecx

mov ecx, edx

inner:

mov al, [esi]

cmp al, [edi]

inc esi

inc edi

jae noSwap

mov al, [esi - 1] ;swap em'

mov ah, [edi - 1]

mov[esi - 1], ah

mov[edi - 1], al

noSwap:

loop inner

pop ecx

loop outer

ret

sort ENDP

END main





