Microsoft (R) Macro Assembler Version 11.00.61030.0 03/19/14 02:25:37

lab7.asm Page 1 - 1

; Juan Pedraza

; CSCI 112 - Lab 7

; Takes 10 numbers from the user

; numbers separated by a space

; puts the numbers into an array and sorts the array using selection sort

; print out the original array and the sorted array in one output box

.586

.MODEL FLAT

INCLUDE io.h

C ; IO.H -- header file for I/O macros (listing suppressed)

C .NOLIST ; turn off listing

C .LIST ; begin listing

C

.STACK 4096

00000000 .DATA

00000000 0000000A [ scoreArray DWORD 10 dup (?) ; unsorted values

00000000

]

00000028 0000000A [ sortedArr DWORD 10 dup(?) ;sorted array. not used. inplace sorting instead w/ scoreArray

00000000

]

00000050 00000000 count1 dword 0

00000054 00000000 indexi DWORD 0 ;index to use in outer loop

00000058 00000000 indexj DWORD 0 ;index to use in inner loop

0000005C 00000000 min DWORD 0 ; current min

00000060 00000000 tmpVal DWORD 0 ;

00000064 45 6E 74 65 72 prompt1 BYTE "Enter ten scores (separated by a space each): ",0

20 74 65 6E 20

73 63 6F 72 65

73 20 28 73 65

70 61 72 61 74

65 64 20 62 79

20 61 20 73 70

61 63 65 20 65

61 63 68 29 3A

20 00

00000093 6F 75 74 70 75 prompt2 BYTE "output is: ",0

74 20 69 73 3A

20 00

0000009F 4F 72 69 67 69 out1 BYTE "Original scores: ",0 ;17 char

6E 61 6C 20 73

63 6F 72 65 73

3A 20 00

000000B1 53 6F 72 74 65 out2 BYTE "Sorted scores: ",0 ;15 char

64 20 73 63 6F

72 65 73 3A 20

00

000000C1 52 65 73 75 6C results BYTE "Results",0

74 73 00

000000C9 000000FA [ finalStr BYTE 250 DUP(?),0 ;final output string

00

] 00

000001C4 0000000B [ tmpStr BYTE 11 DUP(?),0 ;temp string

00

] 00

000001D0 00000028 [ scoreString byte 40 dup(?),0; //for 5 scores - make enough room

00

] 00

000001F9 0000000B [ temp byte 11 dup(?),0

00

] 00

00000205 0000000B [ tempx byte 11 dup(" "),0; //or,..dup(20H); 20H=space

20

] 00

00000211 0000000B [ temp2 byte 11 dup(?),0

00

] 00

00000000 .CODE

00000000 \_MainProc PROC

input prompt1, scoreString, 40; //gets ten scores

0000001E 8D 1D 000001D0 R lea ebx, scoreString

00000024 FF 05 00000050 R outerLoop: inc count1; //outer loop counter++

0000002A 8D 35 00000205 R lea esi, tempx; //flush temp string before using

00000030 8D 3D 000001F9 R lea edi, temp

00000036 FC cld

00000037 B9 0000000B mov ecx, 11

0000003C F3/ A4 rep movsb

0000003E 8D 15 000001F9 R lea edx, temp;

00000044 80 3B 20 innerLoop: cmp byte ptr[ebx], 20h; //if ending mark(space), done

00000047 74 0D je done1;

00000049 80 3B 00 cmp byte ptr[ebx], 00h; //elsif null char, also done

0000004C 74 08 je done1;

0000004E 8A 03 mov AL, byte ptr[ebx]; //otherwise, get 1 byte from input string

00000050 88 02 mov [edx], AL; //and move it to temp

00000052 43 inc ebx; //to next byte in input string

00000053 42 inc edx; //to next byte in temp string

00000054 EB EE jmp innerLoop; //inner loop (temp <- one score)

00000056 done1: ;output prompt2, temp; test display of temp

atod temp; //eax <- temp

00000065 8B 0D 00000050 R mov ecx, count1;

0000006B 49 dec ecx; //counter:1 -> array index:0

0000006C 6B C9 04 imul ecx, 4; //array ele size = 4 bytes

0000006F 89 81 00000000 R mov scoreArray[ecx], eax; //store one score in array

00000075 43 inc ebx; //skip the end mark(space) in the input string

00000076 83 3D 00000050 R cmp count1, 10; //loop 10 times

0A

;cmp count1, 5; //loop 5 times

0000007D 7C A5 jnge outerLoop

;dtoa temp2, scoreArray[0]; or, scorearray+0; //testing display

;output prompt2, temp2

;Juan's code

;make first part of final output string

0000007F 8D 35 0000009F R lea esi, out1

00000085 8D 3D 000000C9 R lea edi, finalStr

0000008B B9 00000011 mov ecx, 17

00000090 FC cld

00000091 F3/ A4 rep movsb

00000093 8D 3D 000000DA R lea edi, finalStr[17]

dtoa tmpStr, scoreArray[0] ;first

000000B5 8D 35 000001C4 R lea esi, tmpStr

000000BB B9 0000000B mov ecx, 11

000000C0 FC cld

000000C1 F3/ A4 rep movsb

000000C3 8D 3D 000000E5 R lea edi, finalStr[17+11]

dtoa tmpStr, scoreArray[4] ;2nd

000000E5 8D 35 000001C4 R lea esi, tmpStr

000000EB B9 0000000B mov ecx, 11

000000F0 FC cld

000000F1 F3/ A4 rep movsb

000000F3 8D 3D 000000F0 R lea edi, finalStr[17+22]

dtoa tmpStr, scoreArray[8] ;3nd

00000115 8D 35 000001C4 R lea esi, tmpStr

0000011B B9 0000000B mov ecx, 11

00000120 FC cld

00000121 F3/ A4 rep movsb

00000123 8D 3D 000000FB R lea edi, finalStr[17+33]

dtoa tmpStr, scoreArray[12] ;4nd

00000145 8D 35 000001C4 R lea esi, tmpStr

0000014B B9 0000000B mov ecx, 11

00000150 FC cld

00000151 F3/ A4 rep movsb

00000153 8D 3D 00000106 R lea edi, finalStr[17+44]

dtoa tmpStr, scoreArray[16] ;5nd

00000175 8D 35 000001C4 R lea esi, tmpStr

0000017B B9 0000000B mov ecx, 11

00000180 FC cld

00000181 F3/ A4 rep movsb

00000183 8D 3D 00000111 R lea edi, finalStr[17+55]

dtoa tmpStr, scoreArray[20] ;6nd

000001A5 8D 35 000001C4 R lea esi, tmpStr

000001AB B9 0000000B mov ecx, 11

000001B0 FC cld

000001B1 F3/ A4 rep movsb

000001B3 8D 3D 0000011C R lea edi, finalStr[17+66]

dtoa tmpStr, scoreArray[24] ;7nd

000001D5 8D 35 000001C4 R lea esi, tmpStr

000001DB B9 0000000B mov ecx, 11

000001E0 FC cld

000001E1 F3/ A4 rep movsb

000001E3 8D 3D 00000127 R lea edi, finalStr[17+77]

dtoa tmpStr, scoreArray[28] ;8nd

00000205 8D 35 000001C4 R lea esi, tmpStr

0000020B B9 0000000B mov ecx, 11

00000210 FC cld

00000211 F3/ A4 rep movsb

00000213 8D 3D 00000132 R lea edi, finalStr[17+88]

dtoa tmpStr, scoreArray[32] ;9nd

00000235 8D 35 000001C4 R lea esi, tmpStr

0000023B B9 0000000B mov ecx, 11

00000240 FC cld

00000241 F3/ A4 rep movsb

00000243 8D 3D 0000013D R lea edi, finalStr[17+99]

dtoa tmpStr, scoreArray[36] ;10nd

00000265 8D 35 000001C4 R lea esi, tmpStr

0000026B B9 0000000B mov ecx, 11

00000270 FC cld

00000271 F3/ A4 rep movsb

00000273 C6 05 00000148 R mov finalStr[17+110], 0dH ;new line

0D

0000027A 8D 3D 00000149 R lea edi, finalStr[17+111]

00000280 8D 35 000000B1 R lea esi, out2

00000286 B9 0000000F mov ecx, 15

0000028B FC cld

0000028C F3/ A4 rep movsb ;finalStr[32+111] next spot

;selection sort

0000028E C7 05 00000054 R mov indexi, 0 ;reset indexes

00000000

00000298 C7 05 00000058 R mov indexj, 0

00000000

000002A2 8D 1D 00000000 R lea ebx, scoreArray ;get address of array

000002A8 83 3D 00000054 R outloop: cmp indexi, 10 ;ten values in array

0A

000002AF 7D 59 jge done2 ;exit if went through whole array

000002B1 8B 03 mov eax, [ebx]

000002B3 A3 0000005C R mov min, eax ;set current index as min

000002B8 8B CB mov ecx, ebx

000002BA 83 C1 04 add ecx, 4 ;second pointer to the next index

000002BD A1 00000054 R mov eax, indexi

000002C2 40 inc eax

000002C3 A3 00000058 R mov indexj, eax

000002C8 83 3D 00000058 R inloop: cmp indexj, 10

0A

000002CF 7D 1C jge inOut

000002D1 8B 01 mov eax, [ecx]

000002D3 39 05 0000005C R cmp min, eax ;check if less than

000002D9 7E 07 jle endIn ;don't need to update min

000002DB A3 0000005C R mov min, eax ;need to update min

000002E0 8B D1 mov edx, ecx ;store address of min

000002E2 FF 05 00000058 R endIn: inc indexj

000002E8 83 C1 04 add ecx, 4 ;move to next index

000002EB EB DB jmp inloop

000002ED inOut: ;swap min to front ebx <- edx

000002ED 8B 03 mov eax, [ebx] ;temp store value that needs to be swapped

000002EF A3 00000060 R mov tmpVal, eax

000002F4 8B 02 mov eax, [edx] ;put min at the front (left)

000002F6 89 03 mov [ebx], eax

000002F8 A1 00000060 R mov eax, tmpVal

000002FD 89 02 mov [edx], eax

000002FF FF 05 00000054 R inc indexi

00000305 83 C3 04 add ebx, 4

00000308 EB 9E jmp outloop

0000030A done2: ;copy sorted array to final string

0000030A 8D 3D 00000158 R lea edi, finalStr[32+111] ;1st

dtoa tmpStr, scoreArray[0]

0000032C 8D 35 000001C4 R lea esi, tmpStr

00000332 B9 0000000B mov ecx, 11

00000337 FC cld

00000338 F3/ A4 rep movsb

0000033A 8D 3D 00000163 R lea edi, finalStr[32+111+11] ;2nd

dtoa tmpStr, scoreArray[4]

0000035C 8D 35 000001C4 R lea esi, tmpStr

00000362 B9 0000000B mov ecx, 11

00000367 FC cld

00000368 F3/ A4 rep movsb

0000036A 8D 3D 0000016E R lea edi, finalStr[32+111+22] ;3nd

dtoa tmpStr, scoreArray[8]

0000038C 8D 35 000001C4 R lea esi, tmpStr

00000392 B9 0000000B mov ecx, 11

00000397 FC cld

00000398 F3/ A4 rep movsb

0000039A 8D 3D 00000179 R lea edi, finalStr[32+111+33] ;4nd

dtoa tmpStr, scoreArray[12]

000003BC 8D 35 000001C4 R lea esi, tmpStr

000003C2 B9 0000000B mov ecx, 11

000003C7 FC cld

000003C8 F3/ A4 rep movsb

000003CA 8D 3D 00000184 R lea edi, finalStr[32+111+44] ;5nd

dtoa tmpStr, scoreArray[16]

000003EC 8D 35 000001C4 R lea esi, tmpStr

000003F2 B9 0000000B mov ecx, 11

000003F7 FC cld

000003F8 F3/ A4 rep movsb

000003FA 8D 3D 0000018F R lea edi, finalStr[32+111+55] ;6nd

dtoa tmpStr, scoreArray[20]

0000041C 8D 35 000001C4 R lea esi, tmpStr

00000422 B9 0000000B mov ecx, 11

00000427 FC cld

00000428 F3/ A4 rep movsb

0000042A 8D 3D 0000019A R lea edi, finalStr[32+111+66] ;7nd

dtoa tmpStr, scoreArray[24]

0000044C 8D 35 000001C4 R lea esi, tmpStr

00000452 B9 0000000B mov ecx, 11

00000457 FC cld

00000458 F3/ A4 rep movsb

0000045A 8D 3D 000001A5 R lea edi, finalStr[32+111+77] ;8nd

dtoa tmpStr, scoreArray[28]

0000047C 8D 35 000001C4 R lea esi, tmpStr

00000482 B9 0000000B mov ecx, 11

00000487 FC cld

00000488 F3/ A4 rep movsb

0000048A 8D 3D 000001B0 R lea edi, finalStr[32+111+88] ;9nd

dtoa tmpStr, scoreArray[32]

000004AC 8D 35 000001C4 R lea esi, tmpStr

000004B2 B9 0000000B mov ecx, 11

000004B7 FC cld

000004B8 F3/ A4 rep movsb

000004BA 8D 3D 000001BB R lea edi, finalStr[32+111+99] ;10nd

dtoa tmpStr, scoreArray[36]

000004DC 8D 35 000001C4 R lea esi, tmpStr

000004E2 B9 0000000B mov ecx, 11

000004E7 FC cld

000004E8 F3/ A4 rep movsb

output results, finalStr

00000503 B8 00000000 mov eax, 0

00000508 C3 ret

00000509 \_MainProc ENDP

END

Microsoft (R) Macro Assembler Version 11.00.61030.0 03/19/14 02:25:37

lab7.asm Symbols 2 - 1

Macros:

N a m e Type

atod . . . . . . . . . . . . . . Proc

atow . . . . . . . . . . . . . . Proc

dtoa . . . . . . . . . . . . . . Proc

input . . . . . . . . . . . . . Proc

output . . . . . . . . . . . . . Proc

wtoa . . . . . . . . . . . . . . Proc

Segments and Groups:

N a m e Size Length Align Combine Class

FLAT . . . . . . . . . . . . . . GROUP

STACK . . . . . . . . . . . . . 32 Bit 00001000 Para Stack 'STACK'

\_DATA . . . . . . . . . . . . . 32 Bit 0000021D Para Public 'DATA'

\_TEXT . . . . . . . . . . . . . 32 Bit 00000509 Para Public 'CODE'

Procedures, parameters, and locals:

N a m e Type Value Attr

\_MainProc . . . . . . . . . . . P Near 00000000 \_TEXT Length= 00000509 Public

outerLoop . . . . . . . . . . L Near 00000024 \_TEXT

innerLoop . . . . . . . . . . L Near 00000044 \_TEXT

done1 . . . . . . . . . . . . L Near 00000056 \_TEXT

outloop . . . . . . . . . . . L Near 000002A8 \_TEXT

inloop . . . . . . . . . . . . L Near 000002C8 \_TEXT

endIn . . . . . . . . . . . . L Near 000002E2 \_TEXT

inOut . . . . . . . . . . . . L Near 000002ED \_TEXT

done2 . . . . . . . . . . . . L Near 0000030A \_TEXT

Symbols:

N a m e Type Value Attr

@CodeSize . . . . . . . . . . . Number 00000000h

@DataSize . . . . . . . . . . . Number 00000000h

@Interface . . . . . . . . . . . Number 00000000h

@Model . . . . . . . . . . . . . Number 00000007h

@code . . . . . . . . . . . . . Text \_TEXT

@data . . . . . . . . . . . . . Text FLAT

@fardata? . . . . . . . . . . . Text FLAT

@fardata . . . . . . . . . . . . Text FLAT

@stack . . . . . . . . . . . . . Text FLAT

\_getInput . . . . . . . . . . . L Near 00000000 FLAT External

\_showOutput . . . . . . . . . . L Near 00000000 FLAT External

atodproc . . . . . . . . . . . . L Near 00000000 FLAT External

atowproc . . . . . . . . . . . . L Near 00000000 FLAT External

count1 . . . . . . . . . . . . . DWord 00000050 \_DATA

dtoaproc . . . . . . . . . . . . L Near 00000000 FLAT External

finalStr . . . . . . . . . . . . Byte 000000C9 \_DATA

indexi . . . . . . . . . . . . . DWord 00000054 \_DATA

indexj . . . . . . . . . . . . . DWord 00000058 \_DATA

min . . . . . . . . . . . . . . DWord 0000005C \_DATA

out1 . . . . . . . . . . . . . . Byte 0000009F \_DATA

out2 . . . . . . . . . . . . . . Byte 000000B1 \_DATA

prompt1 . . . . . . . . . . . . Byte 00000064 \_DATA

prompt2 . . . . . . . . . . . . Byte 00000093 \_DATA

results . . . . . . . . . . . . Byte 000000C1 \_DATA

scoreArray . . . . . . . . . . . DWord 00000000 \_DATA

scoreString . . . . . . . . . . Byte 000001D0 \_DATA

sortedArr . . . . . . . . . . . DWord 00000028 \_DATA

temp2 . . . . . . . . . . . . . Byte 00000211 \_DATA

tempx . . . . . . . . . . . . . Byte 00000205 \_DATA

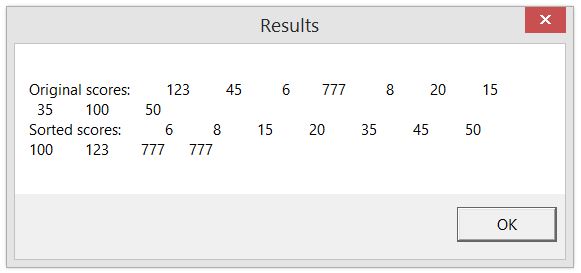
temp . . . . . . . . . . . . . . Byte 000001F9 \_DATA

tmpStr . . . . . . . . . . . . . Byte 000001C4 \_DATA

tmpVal . . . . . . . . . . . . . DWord 00000060 \_DATA

wtoaproc . . . . . . . . . . . . L Near 00000000 FLAT External

0 Warnings

 0 Errors