```
Microsoft (R) Macro Assembler Version 14.00.24210.0         10/30/18 14:02:51
,sortArrays(sortArrays.asm                                   Page 1 - 1
                              TITLE sortArrays(sortArrays.asm)

                              ; Name: Brandon Hough
                              ; CPEN 3710
                              ; Date: October 22, 2018

                              ; Procedure to sort (in its place in memory, without making a copy elsewhere in memory) an array of n unsigned
                              ; doubleword integers starting at address m into ascending numerical order.

                              ; Will print out the two unsorted arrays, the largest un-signed value of the first array, the first and second
                              ; arrays (only the first being sorted), the largest un-signed value of the second array, then finally each the
                              ; two arrays printed out sorted.

                              include Irvine32.inc
                            C ; Include file for Irvine32.lib             (Irvine32.inc)
                            C
                            C ;OPTION CASEMAP:NONE            ; optional: make identifiers case-sensitive
                            C
                            C INCLUDE SmallWin.inc            ; MS-Windows prototypes, structures, and constants
                            C .NOLIST
                            C .LIST
                            C
                            C INCLUDE VirtualKeys.inc
                            C ; VirtualKeys.inc
                            C .NOLIST
                            C .LIST
                            C
                            C
                            C .NOLIST
                            C .LIST
                            C

 00000000                     .data

                              ; initizize array contents for the first array
 00000000 0C0D12AF            firstArray DWORD  0C0D12AFh, 00030256h, 0FFAABBCCh, 0F700F70h,
          00030256
          FFAABBCC
          0F700F70
          00000000
          E222111F
          ABCDEF01
          01234567
                                                 00000000h, 0E222111Fh, 0ABCDEF01h, 01234567h

                              ; initizize array contents for the second array
 00000020 61A80000            secondArray DWORD 61A80000h, 024F4A37h, 0EC010203h, 0FAEEDDCCh,
          024F4A37
          EC010203
```

```
          FAEEDDCC
          2C030175
          84728371
          63AA5678
          CD454443
          22222222
          61B1C2D3
          7A4E96C2
          81002346
          FDB2726E
          65432100
          FFFFFFFF
                                                2C030175h, 84728371h, 63AA5678h, 0CD454443h,
                                                22222222h, 61B1C2D3h, 7A4E96C2h, 81002346h,
                                                0FDB2726Eh, 65432100h, 0FFFFFFFFh

                                ; initilize each prompt text that will be outputed to console
0000005C 41 72 72 61 79         prompt1 BYTE 'Array 1: Unsorted, Array 2: Unsorted',0
          20 31 3A 20 55
          6E 73 6F 72 74
          65 64 2C 20 41
          72 72 61 79 20
          32 3A 20 55 6E
          73 6F 72 74 65
          64 00
00000081 54 68 65 20 6C         prompt2 BYTE 'The largest unsigned value in the first array is: ', 0
          61 72 67 65 73
          74 20 75 6E 73
          69 67 6E 65 64
          20 76 61 6C 75
          65 20 69 6E 20
          74 68 65 20 66
          69 72 73 74 20
          61 72 72 61 79
          20 69 73 3A 20
          00
000000B4 41 72 72 61 79         prompt3 BYTE 'Array 1: Sorted, Array 2: Unsorted',0
          20 31 3A 20 53
          6F 72 74 65 64
          2C 20 41 72 72
          61 79 20 32 3A
          20 55 6E 73 6F
          72 74 65 64 00
000000D7 54 68 65 20 6C         prompt4 BYTE 'The largest unsigned value in the second array is: ', 0
          61 72 67 65 73
          74 20 75 6E 73
          69 67 6E 65 64
          20 76 61 6C 75
          65 20 69 6E 20
          74 68 65 20 73
          65 63 6F 6E 64
```

```
           20 61 72 72 61
           79 20 69 73 3A
           20 00
  0000010B 41 72 72 61 79        prompt5 BYTE 'Array 1: Sorted, Array 2: Sorted',0
           20 31 3A 20 53
           6F 72 74 65 64
           2C 20 41 72 72
           61 79 20 32 3A
           20 53 6F 72 74
           65 64 00

  00000000                       .code
  00000000                       main proc

  00000000  BA 0000005C R        mov edx, OFFSET prompt1           ; store a pointer to the first byte of the prompt1('Array 1: Unsorted,      →
→Array 2: Unsorted')
  00000005  E8 00000000 E        call WriteString                  ; write that previous string of bytes to the command prompt
  0000000A  E8 00000000 E        call Crlf                         ; character return

  0000000F  BE 00000000 R        mov esi, OFFSET firstArray        ; esi points to the memory offset of the first array
  00000014  B9 00000008          mov ecx, LENGTHOF firstArray      ; number of elements in the first array
  00000019  BB 00000004          mov ebx, TYPE firstArray          ; stores the type of bytes in the first array
  0000001E  E8 00000000 E        call DumpMem                      ; dumps memory for first array from Irvine number

  00000023  BE 00000020 R        mov esi, OFFSET secondArray       ; esi points to the memory offset of the second array
  00000028  B9 0000000F          mov ecx, LENGTHOF secondArray     ; number of elements in the second array
  0000002D  BB 00000004          mov ebx, TYPE secondArray         ; stores the type of bytes in the second array
  00000032  E8 00000000 E        call DumpMem                      ; dumps memory for second array from Irvine number
  00000037  E8 00000000 E        call Crlf                         ; character return

  0000003C  BE 00000000 R        mov esi, OFFSET firstArray        ; esi stores the pointer to the beginning of the array
  00000041  B9 00000008          mov ecx, LENGTHOF firstArray      ; ecx stores how many elements are in the array
  00000046  E8 000000C9          call bubbleSort                   ; call sortingProc on first array

  0000004B  BA 00000081 R        mov edx, OFFSET prompt2           ; store a pointer to the first byte of the prompt2('The largest unsigned    →
→value in the first array is: ')
  00000050  E8 00000000 E        call WriteString                  ; write that previous string of bytes to the command prompt
  00000055  8B C3                mov eax, ebx                      ; move largest value that is stored in ebx to eax
  00000057  E8 00000000 E        call WriteHex                     ; print out largest value to screen, WriteHex prints uses eax register

  0000005C  E8 00000000 E        call Crlf                         ; character return
  00000061  E8 00000000 E        call Crlf                         ; character return
  00000066  BA 000000B4 R        mov edx, OFFSET prompt3           ; store a pointer to the first byte of the prompt3('Array 1: Sorted, Array  →
→2: Unsorted')
  0000006B  E8 00000000 E        call WriteString                  ; write that previous string of bytes to the command prompt
  00000070  E8 00000000 E        call Crlf                         ; character return

  00000075  BE 00000000 R        mov esi, OFFSET firstArray        ; esi points to the memory offset of the first array
  0000007A  B9 00000008          mov ecx, LENGTHOF firstArray      ; number of elements in the first array
  0000007F  BB 00000004          mov ebx, TYPE firstArray          ; stores the type of bytes in the first array
  00000084  E8 00000000 E        call DumpMem                      ; dumps memory for first array from Irvine number
```

```
00000089  BE 00000020 R        mov esi, OFFSET secondArray       ; esi points to the memory offset of the second array
0000008E  B9 0000000F          mov ecx, LENGTHOF secondArray     ; number of elements in the second array
00000093  BB 00000004          mov ebx, TYPE secondArray         ; stores the type of bytes in the second array
00000098  E8 00000000 E        call DumpMem                      ; dumps memory for second array from Irvine number
0000009D  E8 00000000 E        call Crlf                         ; character return

000000A2  BE 00000020 R        mov esi, OFFSET secondArray       ; esi stores the pointer to the beginning of the array
000000A7  B9 0000000F          mov ecx, LENGTHOF secondArray     ; ecx stores how many elements are in the array
000000AC  E8 00000063          call bubbleSort                   ; call sortingProc on second array

000000B1  BA 000000D7 R        mov edx, OFFSET prompt4           ; store a pointer to the first byte of the prompt4('The largest unsigned    →
→value in the second array is: ')
000000B6  E8 00000000 E        call WriteString                  ; write that previous string of bytes to the command prompt
000000BB  8B C3                mov eax, ebx                      ; move largest value that is stored in ebx to eax
000000BD  E8 00000000 E        call WriteHex                     ; print out largest value to screen, WriteHex prints uses eax register

000000C2  E8 00000000 E        call Crlf                         ; character return
000000C7  E8 00000000 E        call Crlf                         ; character return
000000CC  BA 0000010B R        mov edx, OFFSET prompt5           ; store a pointer to the first byte of the prompt3('Array 1: Sorted, Array   →
→2: Sorted')
000000D1  E8 00000000 E        call WriteString                  ; write that previous string of bytes to the command prompt
000000D6  E8 00000000 E        call Crlf                         ; character return

000000DB  BE 00000000 R        mov esi, OFFSET firstArray        ; esi points to the memory offset of the first array
000000E0  B9 00000008          mov ecx, LENGTHOF firstArray      ; number of elements in the first array
000000E5  BB 00000004          mov ebx, TYPE firstArray          ; stores the type of bytes in the first array
000000EA  E8 00000000 E        call DumpMem                      ; dumps memory for first array from Irvine number

000000EF  BE 00000020 R        mov esi, OFFSET secondArray       ; esi points to the memory offset of the second array
000000F4  B9 0000000F          mov ecx, LENGTHOF secondArray     ; number of elements in the second array
000000F9  BB 00000004          mov ebx, TYPE secondArray         ; stores the type of bytes in the second array
000000FE  E8 00000000 E        call DumpMem                      ; dumps memory for second array from Irvine number
00000103  E8 00000000 E        call Crlf                         ; character return

00000108  E8 00000000 E        call ReadInt                                  ; keeps up the cmd window to see results of CallMem

                               exit
0000010D  6A 00           *        push   +000000000h
0000010F  E8 00000000 E   *        call   ExitProcess
00000114                       main endp

                               ; ---------------------------------------------------
                               ; This sub-program will sort an array using Bubble Sort
                               ; Recieves: esi = pointer to an array
                               ;           ecx = array size
                               ; Returns: highest value of array in ebx

00000114                       bubbleSort proc

00000114  49                   dec ecx                       ; ecx must be one less than the length of the array
```

```
00000115  8B 1E               mov ebx, [esi]                          ; assume first value is the largest value
00000117  B2 01               mov dl, 1                       ; set dl to 1 to ensure at least one pass through array

00000119                      outerLoop:
00000119  80 FA 00                 cmp dl, 0                  ; compare dl flag to zero
0000011C  74 25                    je endSort                 ; jump to endSort if flag is zero
0000011E  B2 00                    mov dl, 0                  ; else set dl to zero

00000120  51                        push ecx                   ; push ecx count to the stack
00000121  56                        push esi                   ; push esi memory location to the stack (first values location)

00000122                      innerLoop:
00000122  8B 06                     mov eax, [esi]             ; get array value
00000124  3B 46 04                  cmp eax, [esi + 4]         ; compare a pair of values
00000127  76 09                     jbe nextPair               ; if esi <= esi + 4, next pair will be looked at
00000129  77 00                     ja changePositions         ; else esi > esi + 4, pair of values will be switched

0000012B                      changePositions:
0000012B  87 46 04                  xchg eax, [esi + 4]        ; exchange positions of the two values in memory
0000012E  89 06                     mov [esi], eax
00000130  B2 01                     mov dl, 1                  ; set flag that this pass through array had changes occur

00000132                      nextPair:
                                    .if (ebx <= [esi + 4])     ; if ebx is less or equal to then the value at esi + 4
00000132  3B 5E 04            *        cmp    ebx, [esi + 004h]
00000135  77 03               *        ja  @C0001
00000137  8B 5E 04                     mov ebx, [esi + 4]      ; then make ebx that value at esi + 4
                                       .endif
0000013A                    *@C0001:

0000013A  83 C6 04                     add esi, 4                 ; move esi to the next value in memory (4 = DWORD)
0000013D  E2 E3                        loop innerLoop             ; reapeat the inner loop
0000013F  5E                           pop esi                    ; pop esi memory location of the first value from the stack
00000140  59                           pop ecx                    ; pop the ecx count from the stack

00000141  E2 D6                        loop outerLoop             ; reapeat the outer loop

00000143                      endSort:
00000143  C3                           ret                        ; return to main

00000144                      bubbleSort endp
                              end main
Microsoft (R) Macro Assembler Version 14.00.24210.0        10/30/18 14:02:51
```