

```
; Don Spickler
; Simple array program, reads in parameter values and stores in memory
; then prints out array contents.
; Compile with: nasm -f elf array.asm
; Link with (64 bit systems require elf_i386 option): ld -m elf_i386 array.o -o array
; Or run make
; Run with: ./array a1 a2 a3 ... an
```

```
%include 'functions.asm'
```

SECTION .data

```
msg1 db 'Contents: ' ; a message string for output result
msg2 db ' ' ; blank
```

SECTION .bss

```
A resd 100 ; array storage for up to 100 integers.
n resd 1 ; size of the array
```

SECTION .text

```
global _start
```

```
_start:
    pop    ecx            ; first element is the number of arguments
    cmp    ecx, 1         ; if ecx is 1 then halt, no argument or more than one.
    je     finish         ; jump to finish if ecx is 1
    dec    ecx            ; decrement to get size of array
    mov    [n], ecx       ; Store array size

    pop    eax            ; program name.
    mov    edx, A         ; store starting location of memory block for array

readloop:
    cmp    ecx, 0         ; if counter is 0 we stop reading.
    je     writearray     ; when finished, write out array
    pop    eax            ; read next argument from stack (command line)
    call   atoi           ; convert to integer
    mov    [edx], eax     ; store integer in array at current pointer position
    add    edx, 4         ; increment pointer to next location. Storing double words = 4 bytes

    dec    ecx            ; decrement counter
    jmp    readloop       ; loop back

writearray:
    mov    ecx, [n]       ; reset counter to array size
    mov    edx, A         ; store starting location of memory block for array
    mov    eax, msg1      ; print out the output message
    call   sprint

writeloop:
    cmp    ecx, 0         ; test if finished writing.
    je     finish         ; if so go to finish
    mov    eax, [edx]     ; load array element into eax
    call   iprint         ; print integer
    mov    eax, msg2      ; load the space
    call   sprint         ; print space

    add    edx, 4         ; increment pointer to next location. Storing double words = 4 bytes
    dec    ecx            ; decrement counter
    jmp    writeloop      ; loop back

finish:
    mov    eax, msg2      ; load the space
    call   sprintLF       ; print the space with a line feed
    call   quit           ; end program
```