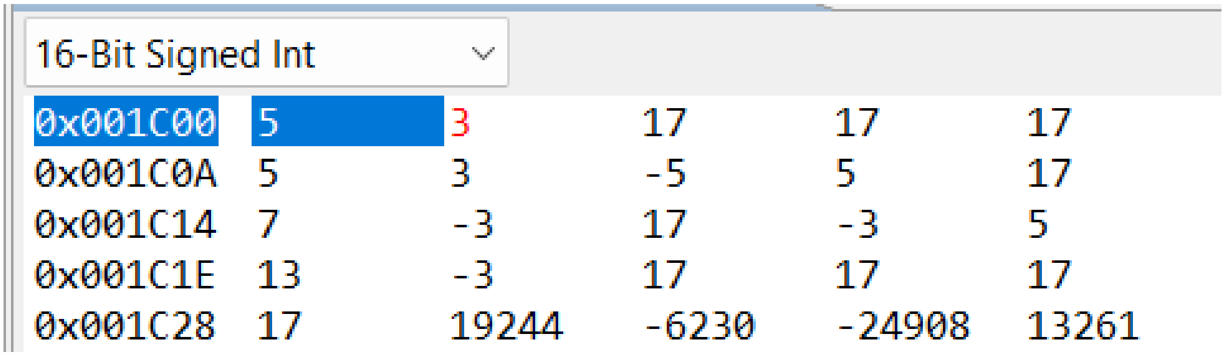
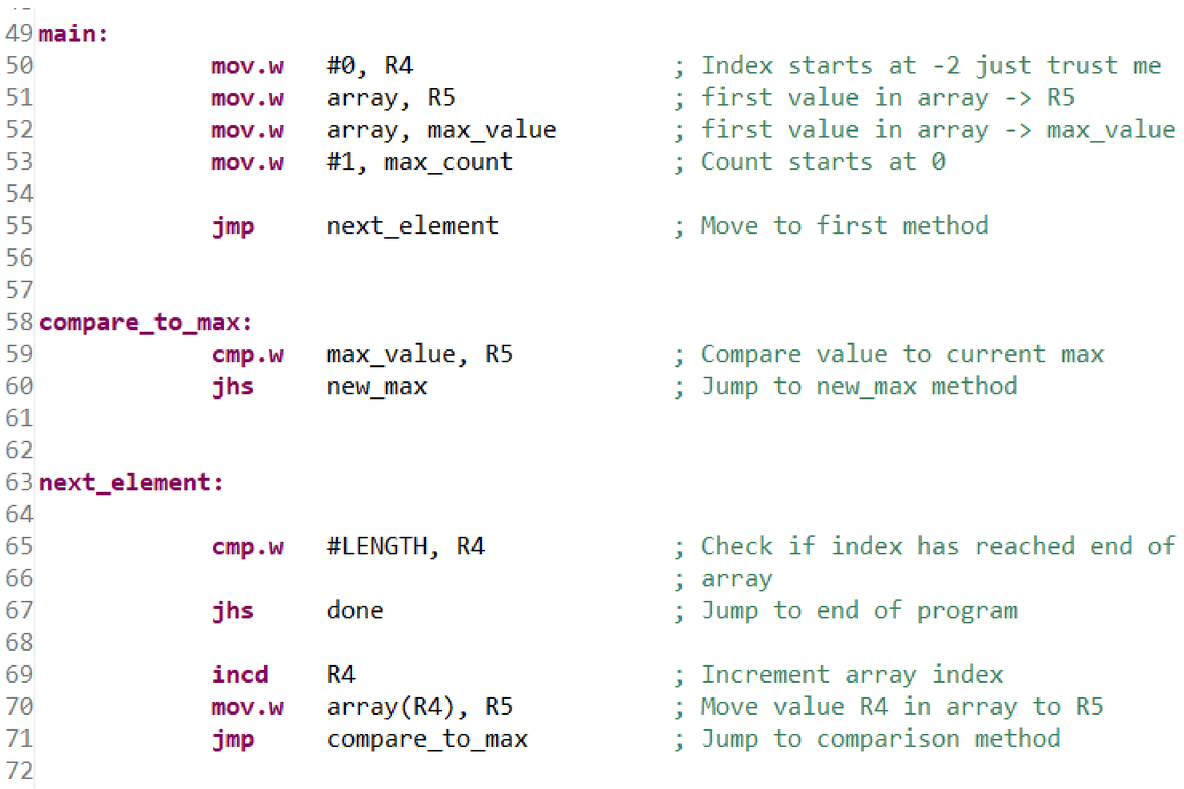
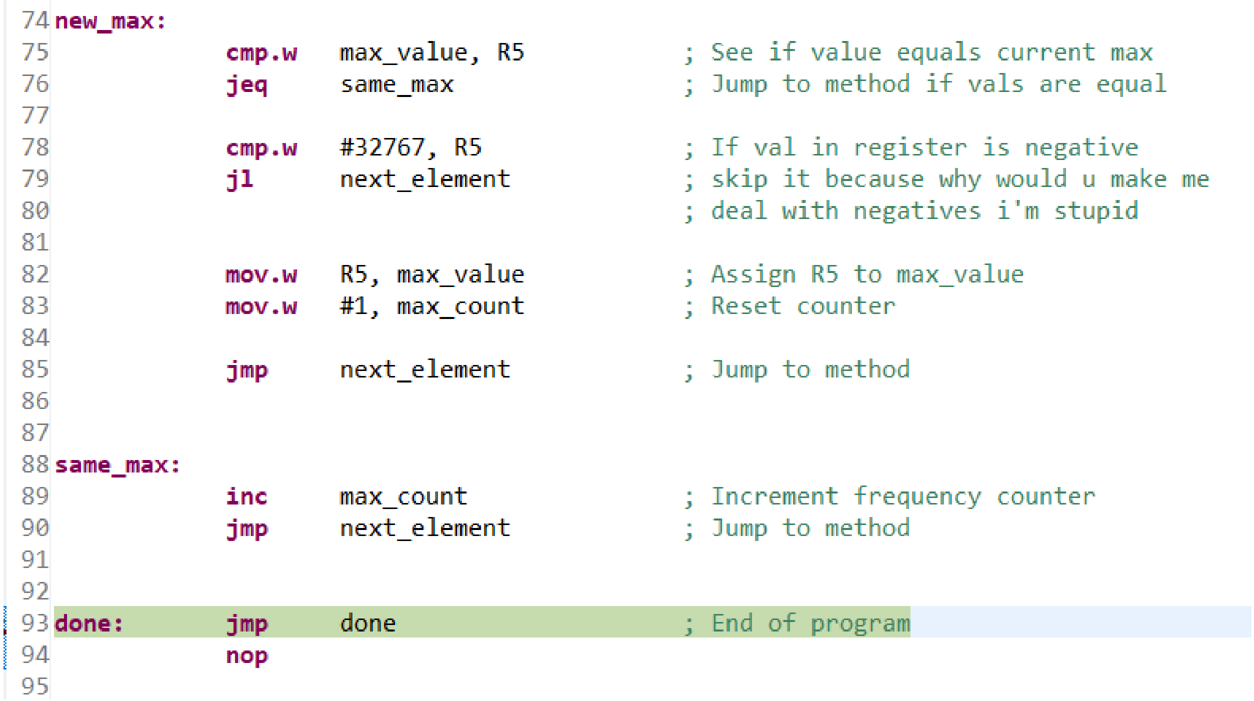


16-Bit Signed Int

```
0x001C00  5        3        17       17       17
0x001C0A  5        3        -5       5        17
0x001C14  7        -3       17       -3       5
0x001C1E  13       -3       17       17       17
0x001C28  17       19244    -6230    -24908   13261
```



```
main:
            mov.w   #0, R4                  ; Index starts at -2 just trust me
            mov.w   array, R5               ; first value in array -> R5
            mov.w   array, max_value        ; first value in array -> max_value
            mov.w   #1, max_count           ; Count starts at 0

            jmp     next_element            ; Move to first method


compare_to_max:
            cmp.w   max_value, R5           ; Compare value to current max
            jhs     new_max                 ; Jump to new_max method


next_element:

            cmp.w   #LENGTH, R4             ; Check if index has reached end of
                                            ; array
            jhs     done                    ; Jump to end of program

            incd    R4                      ; Increment array index
            mov.w   array(R4), R5           ; Move value R4 in array to R5
            jmp     compare_to_max          ; Jump to comparison method

```



```
new_max:
            cmp.w   max_value, R5           ; See if value equals current max
            jeq     same_max                ; Jump to method if vals are equal

            cmp.w   #32767, R5              ; If val in register is negative
            jl      next_element            ; skip it because why would u make me
                                            ; deal with negatives i'm stupid

            mov.w   R5, max_value           ; Assign R5 to max_value
            mov.w   #1, max_count           ; Reset counter

            jmp     next_element            ; Jump to method


same_max:
            inc     max_count               ; Increment frequency counter
            jmp     next_element            ; Jump to method


done:       jmp     done                    ; End of program
            nop

```