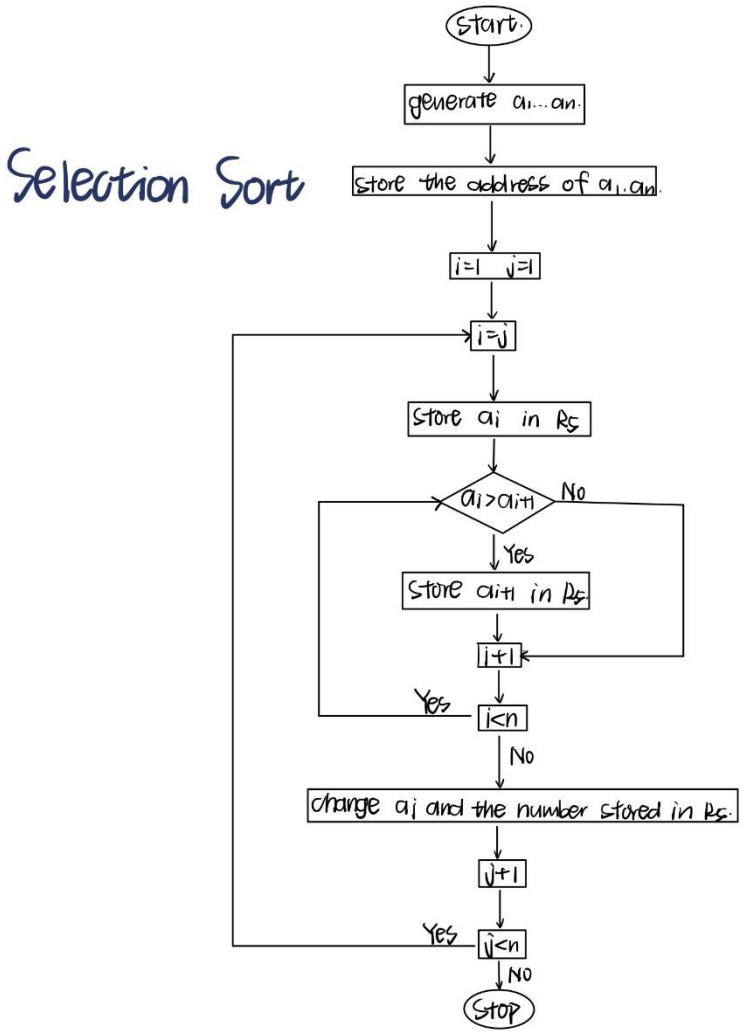
EE 215 Microprocessors LAB #2

Student name：Li Xianzhe
Student ID number：2022214880

Description of Approach

Selection sort is a straightforward sorting algorithm implemented in assembly language. It involves two nested loops. The outer loop selects the minimum element's index, while the inner loop searches for the minimum element from the current index to the end of the array. When the inner loop completes, the smallest element is swapped with the element at the outer loop's current index, placing it in its correct position. This process is repeated until the entire array is sorted, with the outer loop incrementing and the inner loop's search range reducing in each iteration. The assembly code for this may vary depending on the architecture and assembly language used.

Flowchart



Code

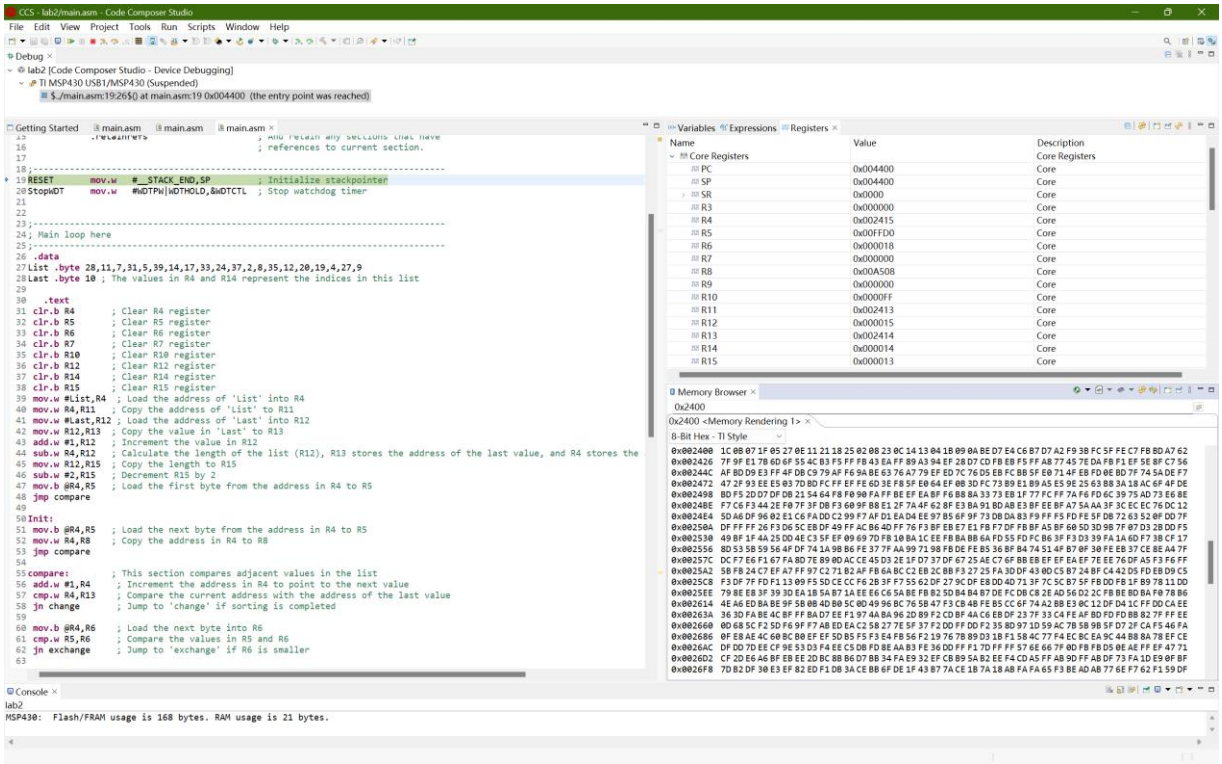
```
.data
List .byte 28,11,7,31,5,39,14,17,33,24,37,2,8,35,12,20,19,4,27,9
Last .byte 10 ; The values in R4 and R14 represent the indices in this list

   .text
 clr.b R4       ; Clear R4 register
 clr.b R5       ; Clear R5 register
 clr.b R6       ; Clear R6 register
 clr.b R7       ; Clear R7 register
 clr.b R10      ; Clear R10 register
 clr.b R12      ; Clear R12 register
 clr.b R14      ; Clear R14 register
 clr.b R15      ; Clear R15 register
 mov.w #List,R4  ; Load the address of 'List' into R4
 mov.w R4,R11   ; Copy the address of 'List' to R11
 mov.w #Last,R12 ; Load the address of 'Last' into R12
 mov.w R12,R13  ; Copy the value in 'Last' to R13
 add.w #1,R12   ; Increment the value in R12
 sub.w R4,R12   ; Calculate the length of the list (R12), R13 stores the
address of the last value, and R4 stores the address of the first value
 mov.w R12,R15  ; Copy the length to R15
 sub.w #2,R15   ; Decrement R15 by 2
 mov.b @R4,R5   ; Load the first byte from the address in R4 to R5
 jmp compare

Init:
 mov.b @R4,R5   ; Load the next byte from the address in R4 to R5
 mov.w R4,R8    ; Copy the address in R4 to R8
 jmp compare

compare:        ; This section compares adjacent values in the list
 add.w #1,R4     ; Increment the address in R4 to point to the next value
 cmp.w R4,R13    ; Compare the current address with the address of the last
value
 jn change       ; Jump to 'change' if sorting is completed

 mov.b @R4,R6    ; Load the next byte into R6
 cmp.w R5,R6     ; Compare the values in R5 and R6
 jn exchange     ; Jump to 'exchange' if R6 is smaller

 jmp compare
```

```
exchange:       ; Swap the smallest byte into R5
 mov.b R6,R5     ; Copy the value in R6 to R5
 mov.w R4,R8     ; Store the address of the smallest value in R8
 jmp compare

change:         ; Move the smallest byte to its correct position
 sub.w R11,R8    ; Calculate the distance from the first number to the
smallest
 jz again        ; If the distance is 0, go to 'again'
 mov.b @R11,R10  ; Load the byte at the address in R11 into R10
 add.w R8,R11    ; Add the distance to R11 to point to the next location
 mov.b @R11,R9   ; Load the byte at the address in R11 into R9
 mov.b R10,0(R11) ; Move the value in R10 to the original address in R11
 sub.w R8,R11    ; Subtract the distance from R11
 mov.b R9,0(R11) ; Move the value in R9 to the original address in R11
 add.w #1,R11    ; Increment R11 to the next location
 jmp again

again:          ; Prepare for the next iteration
 add.w #1,R14    ; Increment R14 to keep track of iterations
 cmp.w R14,R15   ; Compare the current iteration with the length of the
list
 jn exit         ; If sorting is completed, jump to 'exit'

 sub.w R12,R4    ; Reset R4 to the beginning of the list
 add.w R14,R4    ; Move R4 to the next portion of the list
 jmp Init

exit:
 jmp $           ; End of the program




;-------------------------------------------------------------------------
-----
; Stack Pointer definition
;-------------------------------------------------------------------------
-----
           .global __STACK_END
           .sect   .stack

;-------------------------------------------------------------------------
```

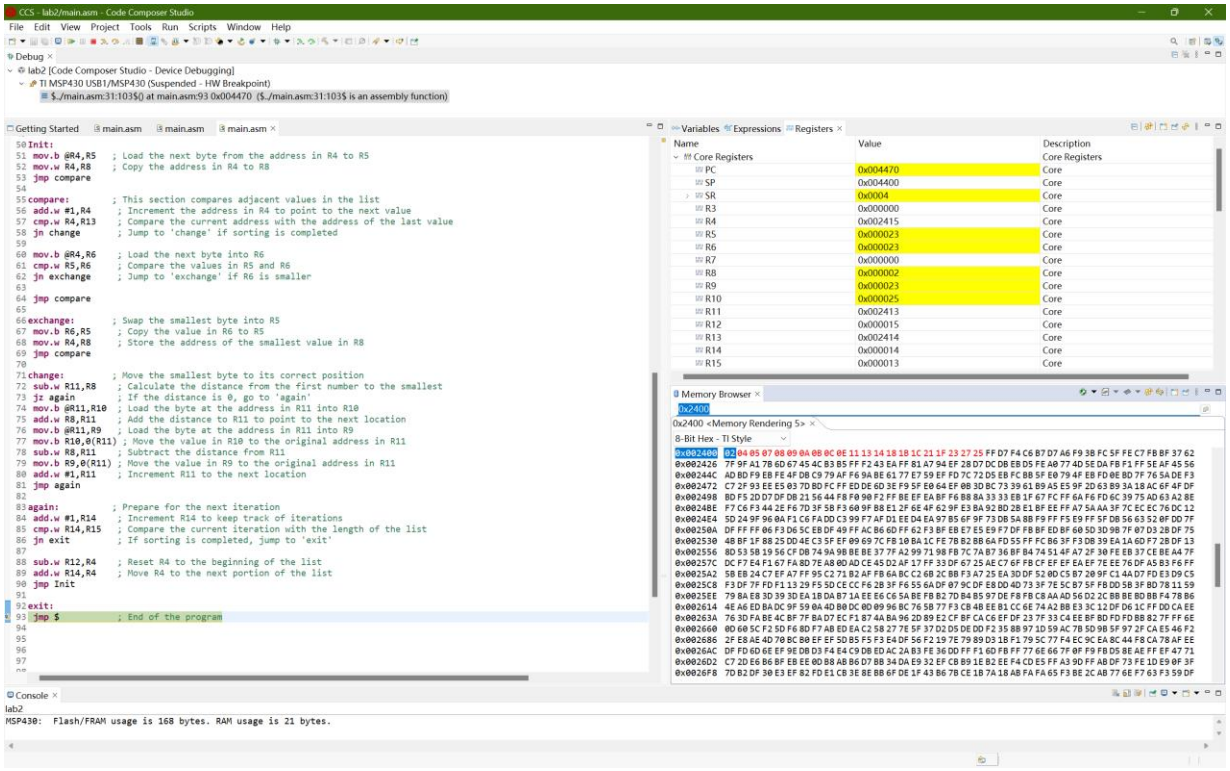
```
-----
; Interrupt Vectors
;-------------------------------------------------------------------------------
-----
            .sect   ".reset"                ; MSP430 RESET Vector
            .short  RESET
```

Screen Snapshots(Results)

before:



after:



Description of Results

Select Sort to browse the list and find the smallest number, and then swap that number with the first number in the list. For the next pass, scan the remaining list (your work list - all items except the first item), find the smallest number and swap it with the second number in the list. Continue until the size of the worklist is less than 2.