**Team Members:**

1. M Praveen kumar CB.EN.U4CSE20449
2. S Jayandhar CB.EN.U4CSE20425
3. M Abishek CB.EN.U4cse20402

**Assembly Code:**

AREA bubble,CODE,READONLY

ENTRY

Main

LDR R0,=0X40000000

MOV R1, #6 ;Index value

STR R1, [R0],#4 ;Stores in memory

MOV R1, #5

STR R1, [R0],#4 ; memory auto increments by 1

MOV R1, #4

STR R1, [R0],#4

MOV R1, #3

STR R1, [R0],#4

MOV R1, #1

STR R1, [R0],#4

MOV R1, #0

STR R1, [R0],#4

MOV R2,#6 ;R2 has the size of the array

MOV R3,#0 ;R3 is the outer loop index

MOV R4,#0 ;R4 is the inner loop index j

Outer

LDR R0,=0X40000000

CMP R3, R2

BGT Exit

MOV R4,#0

SUB R5,R2,R3

SUB R5,R5,#1 ;Inner loop break condition

BL Inner

ADD R3,R3,#1

B Outer

Inner

CMP R4, R5

BXGE LR

LDR R6,[R0],#4

LDR R7,[R0]

CMP R6, R7

MOVGT R8,R6 ;Swap occurs when arr[c]>arr[c+1]

MOVGT R6,R7

MOVGT R7,R8

STR R7,[R0] ;Storing updated register

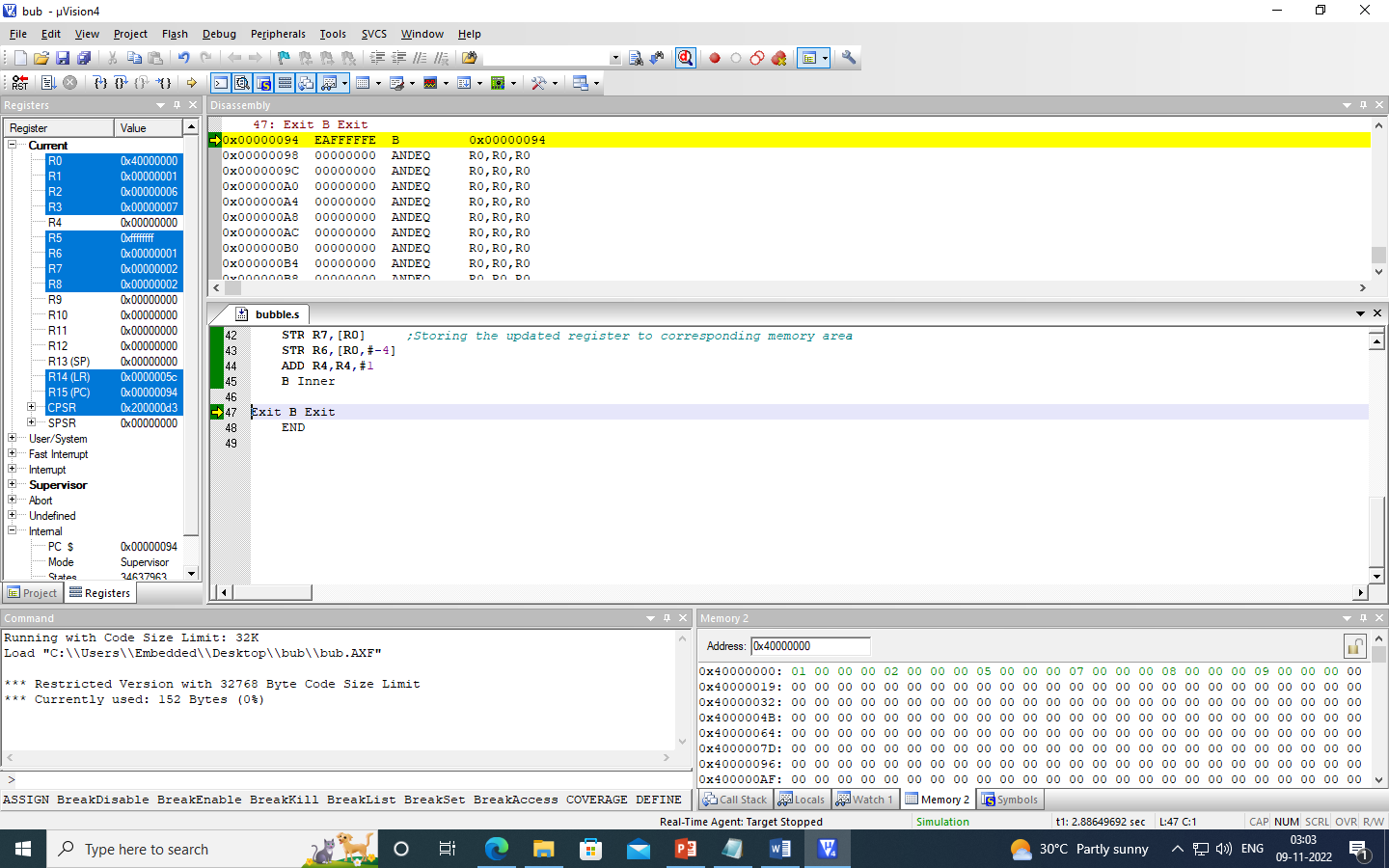
STR R6,[R0,#-4]

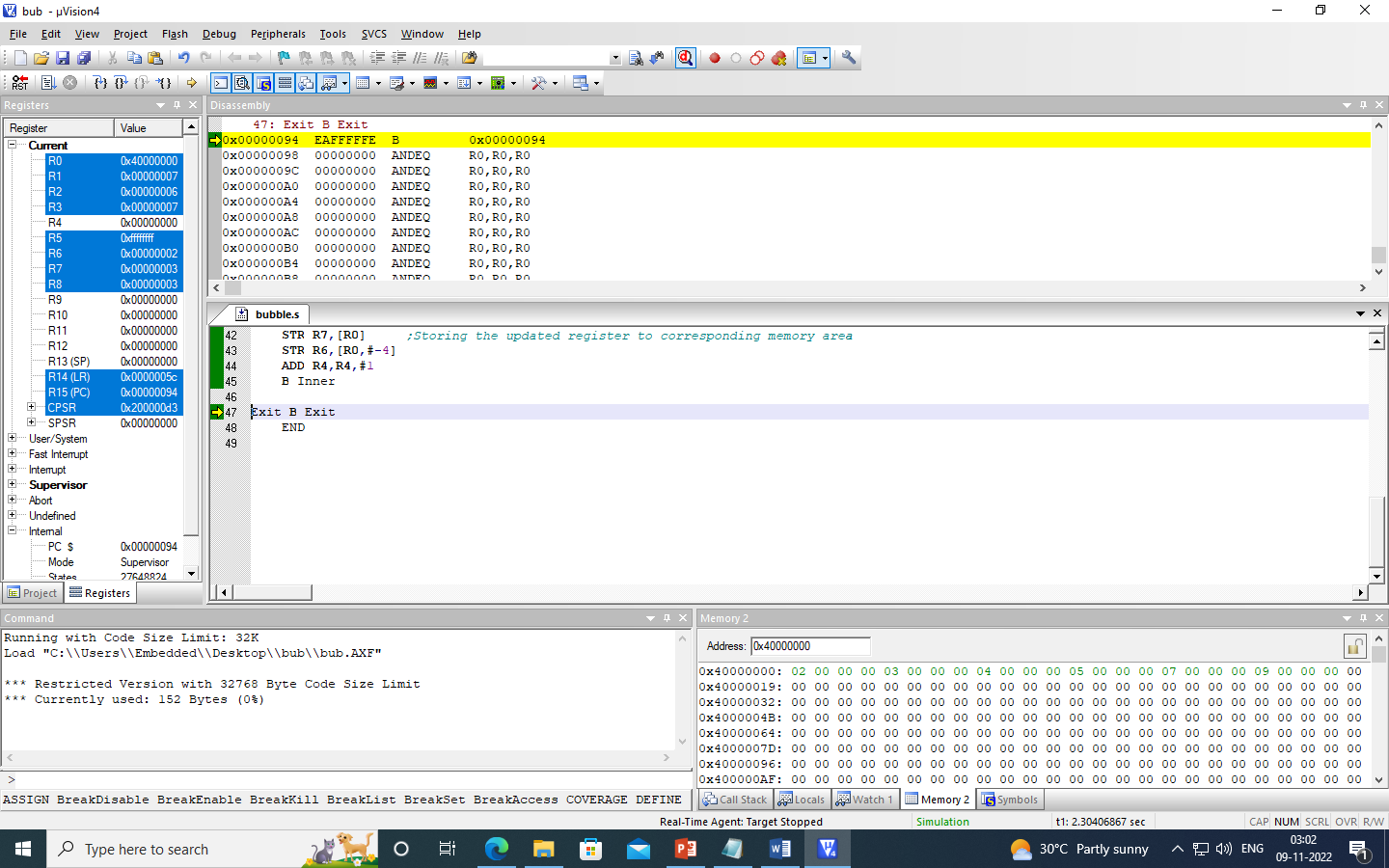
ADD R4,R4,#1

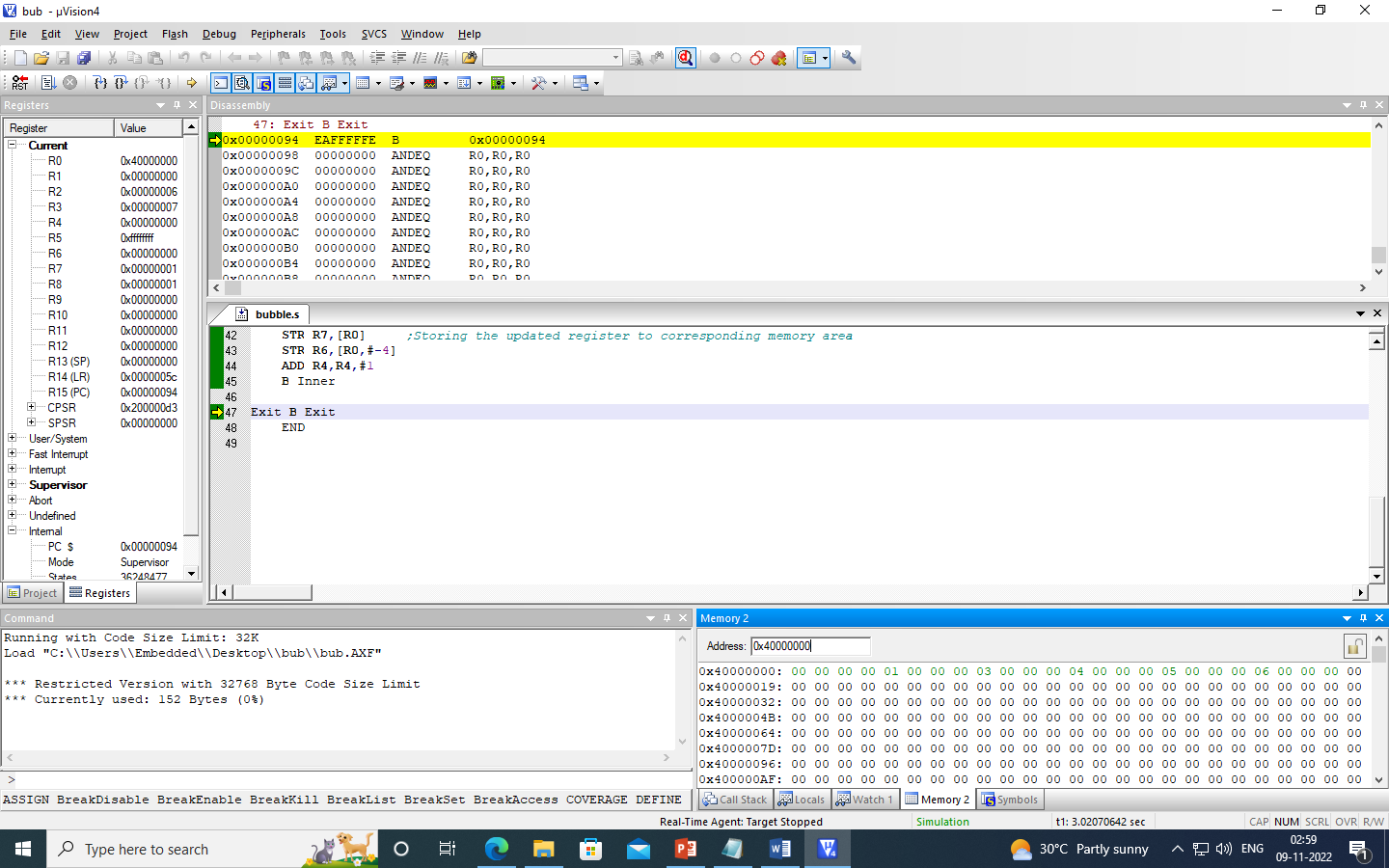
B Inner

Exit B Exit

END







Insertion Code:

AREA DATAARR, DATA, READWRITE

ARRAY DCD 8,2,3,0,1,4,9,5,6,7 ;Array for sorting

AREA MYCODE, CODE, READONLY

ENTRY

EXPORT main

main

MOV R0, #1 ;Start index

LSL R0, R0, #2

MOV R1, #10 ;Size of array

LSL R1, R1, #2

BL INSERTION

LDR R0, =ARRAY

LDR R1, [R0, #0]

LDR R2, [R0, #4]

LDR R3, [R0, #8]

LDR R4, [R0, #12]

LDR R5, [R0, #16]

LDR R6, [R0, #20]

LDR R7, [R0, #24]

LDR R8, [R0, #28]

LDR R9, [R0, #32]

LDR R10, [R0, #36]

B STOP

INSERTION

SUB SP, SP, #12 ;Mov stack for keep current parameter

STR LR, [SP, #8] ;Save return address

STR R0, [SP, #4] ;Save start index argrument

STR R1, [SP, #0] ;Save size of array argrument

CMP R0, R1 ;If start index = size of array

BEQ RET ;Return to main

CMP R0, #0 ;If start index = 0

BEQ RET ;Return to main

;Check if less than after

LDR R12, =ARRAY ;Get address of array

LDR R2, [R12, R0] ;Load array[start] to R2

SUB R4, R0, #4 ;Get value of start-1 in R4

LDR R3, [R12, R4] ;Load array[start-1] to R3

CMP R2, R3 ;if array[start]<array[start-1]

BLT SWAP ;swap it

RECUR

LDR R0, [SP, #4] ;Load start index from this recursion

LDR R1, [SP, #0] ;Load size from this recursion

ADD R0, R0, #4 ;increse start

BL INSERTION ;call recursion for next number

B RET ;return when end function

SWAP

MOV R5, R2 ;temp = array[start]

STR R3, [R12, R0] ;array[start] = array[start-1]

STR R5, [R12, R4] ;arrar[start-1] = temp

MOV R1, R0 ;pass start to size

SUB R0, R0, #4 ;decrese start

BL INSERTION ;call recursion for prev number

B RECUR ;back to current function call

RET

LDR LR, [SP, #8] ;load LR to return caller

ADD SP, SP, #12 ;pop stack

MOV PC, LR ;set LR to PC

STOP

B STOP

END

