**CompE-271**

* I declare that all material in this assignment is my own work except where there is clear reference to the work of others.
* I have read, understood and agree to the SDSU Policy on Plagiarism and Cheating on the university website at <http://go.sdsu.edu/student_affairs/srr/cheating-plagiarism.aspx> , the syllabus and the student-teacher contract for the consequences of plagiarism, including both academic and punitive sanctions.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Remark\*. By submitting this assignment report electronically, you are deemed to have signed the declaration above.*

10/2/2019

[Homework assignment #5]

[Homework 5]

Ckick below to enter/change your Name and RedID

Ethan Nagelvoort, 821234668

**Content**

(\* - Mandatory)

1\*. Description of the problem/method

For the first function, sumofFirstNNumbers, it takes in an int N returns an int that is all the numbers added up till N. To do this I stored variables int sum and i into two registers,r1 and r2. I also set r3 to equal N+1 because when use a loop, I wanted my loop to end and include N. I then used a for loop using cmp to check if I and N+1 are equal. If they are then the code would go to another branch. In the loop i would add I to sum and iterate i. At the end of the loop, the code would branch back to for, or the beginning of the loop. In the return branch I set r0 to equal r1 or sum. I do this because r0 is the return register.

For the second function, sumofEvenNumbers, it takes all the even numbers between the parameters N1 and N2 and returns the sum of them. I then have set r2 to equal 0 and this represents the result. I also store r0/N1 into r3. I then use a while loop to go through the numbers between N1 and N2. I then use bge done so the loop will branch out if n1 >= n2. I then have r4 equal r3 and then lsr and lsl r4 by one. This will help check if r4 is even. Then if r4 is equal to r3, r4 is even. I use a cmp here and create two branches an bne one and a beq one. In the beq one, I add r3 to r2 and iterate r3 by one. In the bne branch, I iterate r3. Also in both branches, I branch back to the while loop. In the done branch, I then use a similar method from before to check if r3 is even cause when r3 is equal to N2 in the while loop, the code does not check if its even. Hence, I check it N2 is even after the while loop and if it is, I add N2 to r2. If not, I branch out to done2. In done2, I set r0 equal to r2.

2. Pseudocode (if required. Mandatory for the Lab assignments, starting from #5 and Projects)

/\*

\* Assembly.S

\*/

.global sumofFirstNNumbers

.data

declare any global variables here

.text

l,

Do not modify and wory abou the next three lines. Include them in every function. This will be clear as we learn more

save stack pointer into register r12

reserve 32 bytes of space for local variables

push link register onto stack -- make sure you pop it out before you return

SUMOFFIRSTNNUMBERS

r3=N,r1=sum, r2=i

set sum = 0

set i = 0

r3 = N+1

for:

r2-r3

if (i==N)

exit loop

sum = sum + i

i = i+1

b for

return:

set N=sum, r0 is return register

Do not modify and worry about the follwoing two lines. Include them in every function. This will be clear as we learn more

pop link register from stack

restore the stack pointer -- Please note stack pointer should be equal to the

value it had when you entered the function .

return from the function by copying link register into program counter

.global sumofEvenNumbers

.data

declare any global variables here

.text

sumofEvenNumbers:

Do not modify and wory abou the next three lines. Include them in every function. This will be clear as we learn more

save stack pointer into register r12

reserve 32 bytes of space for local variables

push link register onto stack -- make sure you pop it out before you return

r2 = result, r0 = n1, r1 = n2

r2 = 0

r3 = r0

while:

if r3 >= r1 then n1>=n2. If this is true, loop ends

bge done

r4 = r3

r3 = r3 >> 1

r3 = r3<<1

cmp r3, r4

beq b1

bne b2

b1:

r2=r2+1

r3=r3+1

b while

b2:

r3=r3+1

b while

while loop does not check if last number is even or not, so following code does

r3= r3 >> 1

r3 = r3 << 1

r2 = r2 + r3

r0=r2

Do not modify and worry about the follwoing two lines. Include them in every function. This will be clear as we learn more

pop link register from stack

restore the stack pointer -- Please note stack pointer should be equal to the

value it had when you entered the function .

return from the function by copying link register into program counter

3\*. C-code

/\*

\* Assembly.S

\*/

.global sumofFirstNNumbers

.data

// declare any global variables here

.text

sumofFirstNNumbers:

// Do not modify and wory abou the next three lines. Include them in every function. This will be clear as we learn more

mov r12,r13 // save stack pointer into register r12

sub sp,#32 // reserve 32 bytes of space for local variables

push {r1} // push link register onto stack -- make sure you pop it out before you return

// SUMOFFIRSTNNUMBERS

//r3=N,r1=sum, r2=i

mov r1, #0 //set sum = 0

mov r2, #0 // set i = 0

ADD r3, r0, #1 // r3 = N+1

for:

cmp r2, r3 //r2-r3

beq return //if (i==N)

//exit loop

ADD r1, r1, r2 //sum = sum + i

ADD r2, r2, #1 //i = i+1

b for

return:

mov r0, r1//set N=sum, r0 is return register

// Do not modify and worry about the follwoing two lines. Include them in every function. This will be clear as we learn more

pop {r1} // pop link register from stack

mov sp,r12 // restore the stack pointer -- Please note stack pointer should be equal to the

// value it had when you entered the function .

bx lr // return from the function by copying link register into program counter

.global sumofEvenNumbers

.data

// declare any global variables here

.text

sumofEvenNumbers:

// Do not modify and wory abou the next three lines. Include them in every function. This will be clear as we learn more

mov r12,r13 // save stack pointer into register r12

sub sp,#32 // reserve 32 bytes of space for local variables

push {r3} // push link register onto stack -- make sure you pop it out before you return

//r2 = result, r0 = n1, r1 = n2

mov r2, #0 // r2 = 0

mov r3, r0 // r3 = r0

while:

cmp r3,r1 //if r3 >= r1 then n1>=n2. If this is true, loop ends

bge done

mov r4, r3 // r4 = r3

lsr r4, r4, #1 // r3 = r3 >> 1

lsl r4, r4, #1 // r3 = r3<<1

cmp r3, r4

beq b1

bne b2

b1:

add r2, r2, r3 // r2=r2+1

add r3, r3, #1

b while

b2:

add r3, r3, #1 // r3=r3+1

b while

done: //while loop does not check if last number is even or not, so following code does

lsr r3, r3, #1 //r3 = r3 >> 1

lsl r3, r3, #1 //r3 = r3 << 1

cmp r3, r1

BNE done2

add r2, r2, r3 //r2 = r2 + r3

done2:

mov r0, r2

// Do not modify and worry about the follwoing two lines. Include them in every function. This will be clear as we learn more

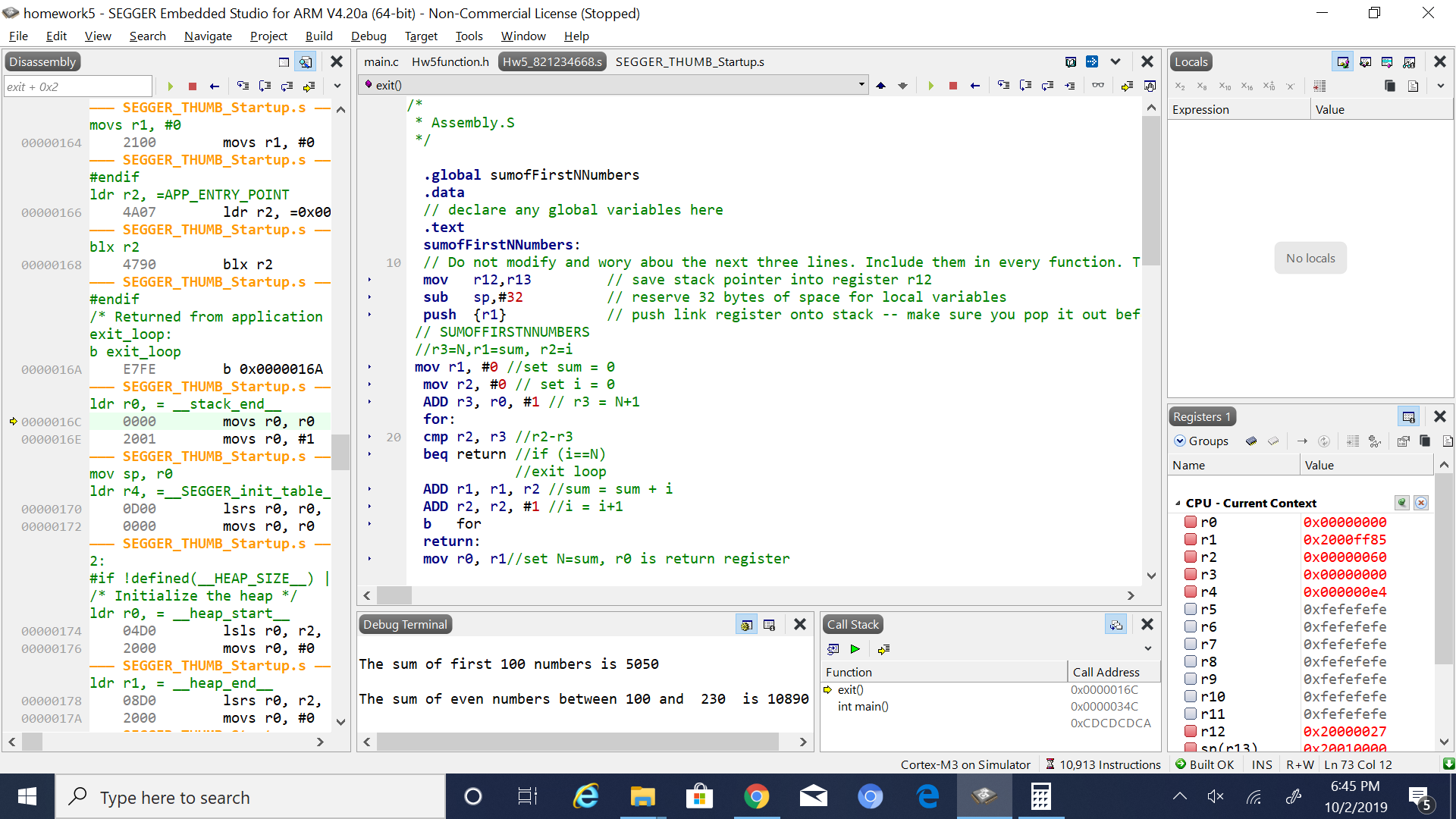
pop {r3} // pop link register from stack

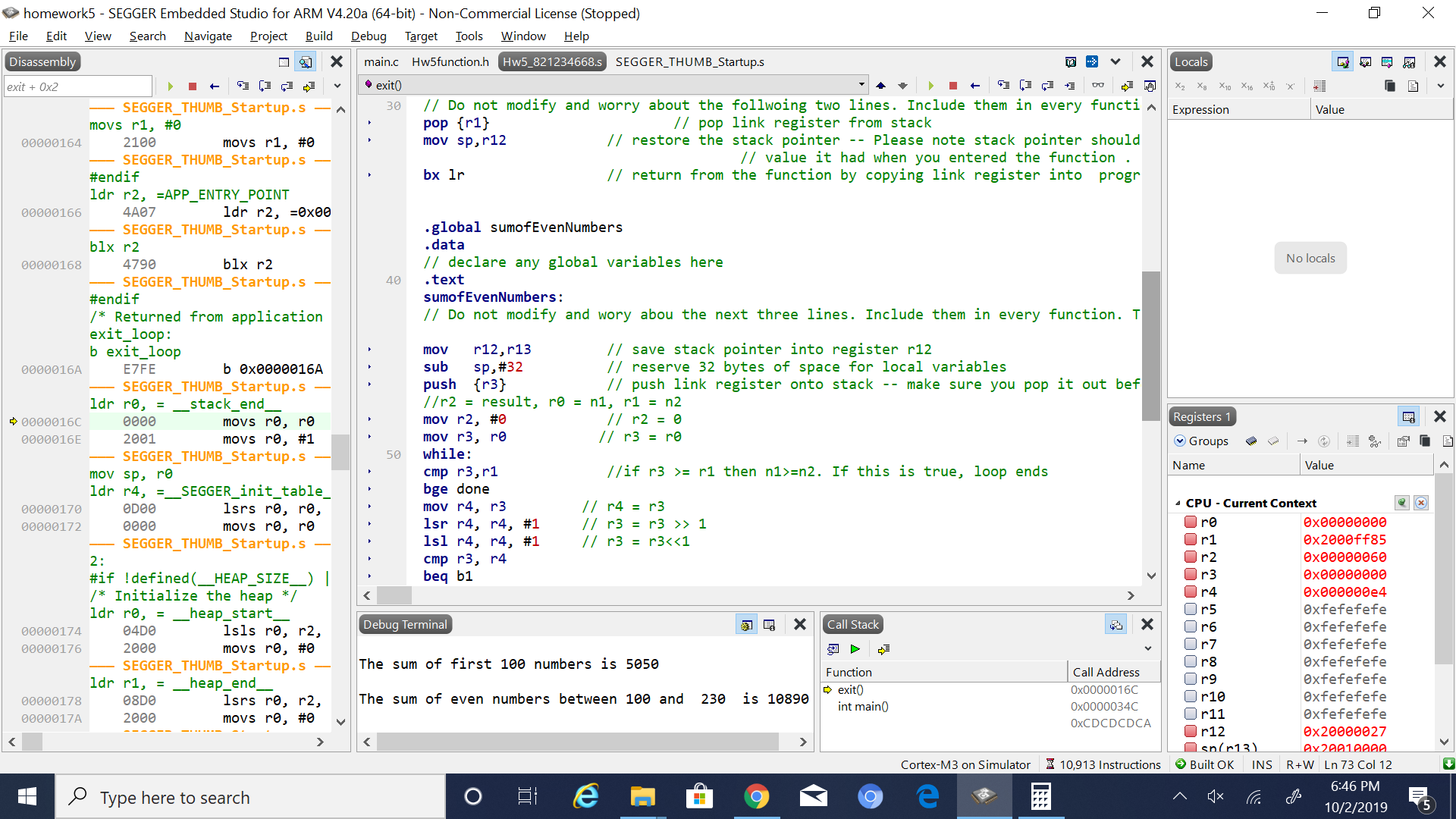
mov sp,r12 // restore the stack pointer -- Please note stack pointer should be equal to the

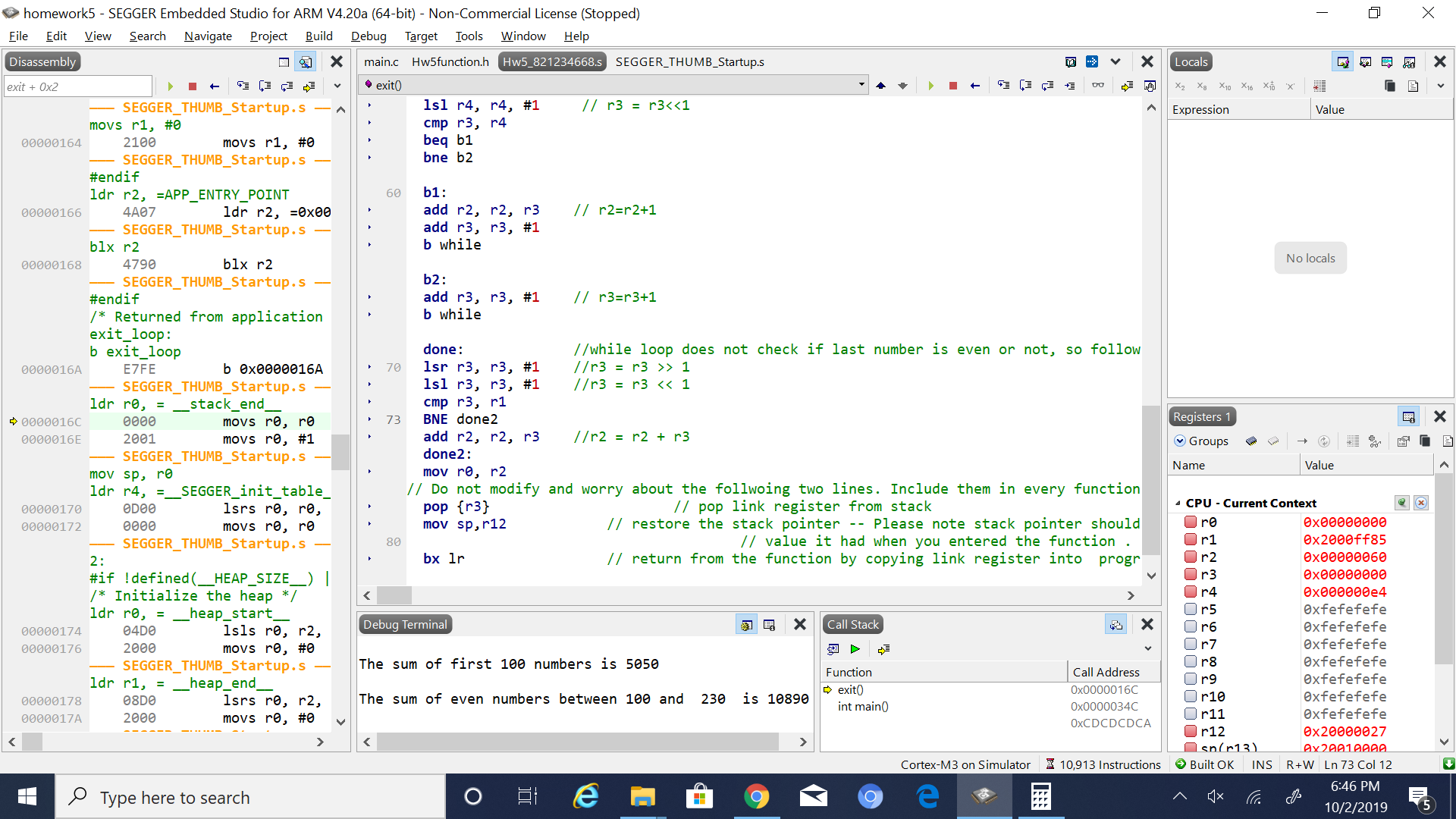
// value it had when you entered the function .

bx lr // return from the function by copying link register into program counter

4\*. Screen capture of the code and the resulting display(s)







5. Conclusion (if applicable)

6\*. References.

Ken arnolds slides on arm coding