**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.*

12/9/2019

[HW#X2-8]

[X2-8]

Ckick below to enter/change your Name and RedID

Ethan Nagelvoort, 821234668

**Content**

(\* - Mandatory)

1\*. Description of the problem/method

I first place n into r1 and I set r2 = 0, which is our x/sum value. I also set r3=0, which is our i. I then add one to r1 because we are going to use a for loop that loops to i<n+1. The program then branches to a for branch which adds r3 to r2 and increments r3 every loop. The program branches out of the loop if r1 = r3. When this happens, r2 will hold the final sum and is placed into r0, making that the return value.

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

sum1ton:

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

r1 = r0, r1 is now n

r2 = 0, r2 is x

r3=0, r3 is i

r1 = r1+1

branch to for

inside for branch

compare r1 and r3

if r1=r3 then branch to end

r2 = r2 + r3

r3 = r3 + 1, increment r3

]branch back to for

inside end branch

r0 = r2

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

//main file

int sum1ton(int n);

int main()

{

int num = sum1ton(6);

int num2 = sum1ton(10);

printf("When 6 is placed into the sum1ton function, %d is returned",num);

printf("\nWhen 10 is placed into the sum1ton function, %d is returned", num2);

return 0;

}

//function file

.global sum1ton

.data

// declare any global variables here

.text

sum1ton:

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

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

push {r2} // push link register onto stack -- make sure you pop it out before you return

mov r1,r0 //r1 = r0, r1 is now n

mov r2,#0 //r2 = 0, r2 is x

mov r3,#0 //r3=0, r3 is i

add r1, r1, #1 //r1 = r1+1

b for //branch to for

for: //inside for branch

cmp r1,r3 //compare r1 and r3

beq end //if r1=r3 then branch to end

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

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

b for //branch back to for

end: //inside end branch

mov r0, r2 // r0 = r2

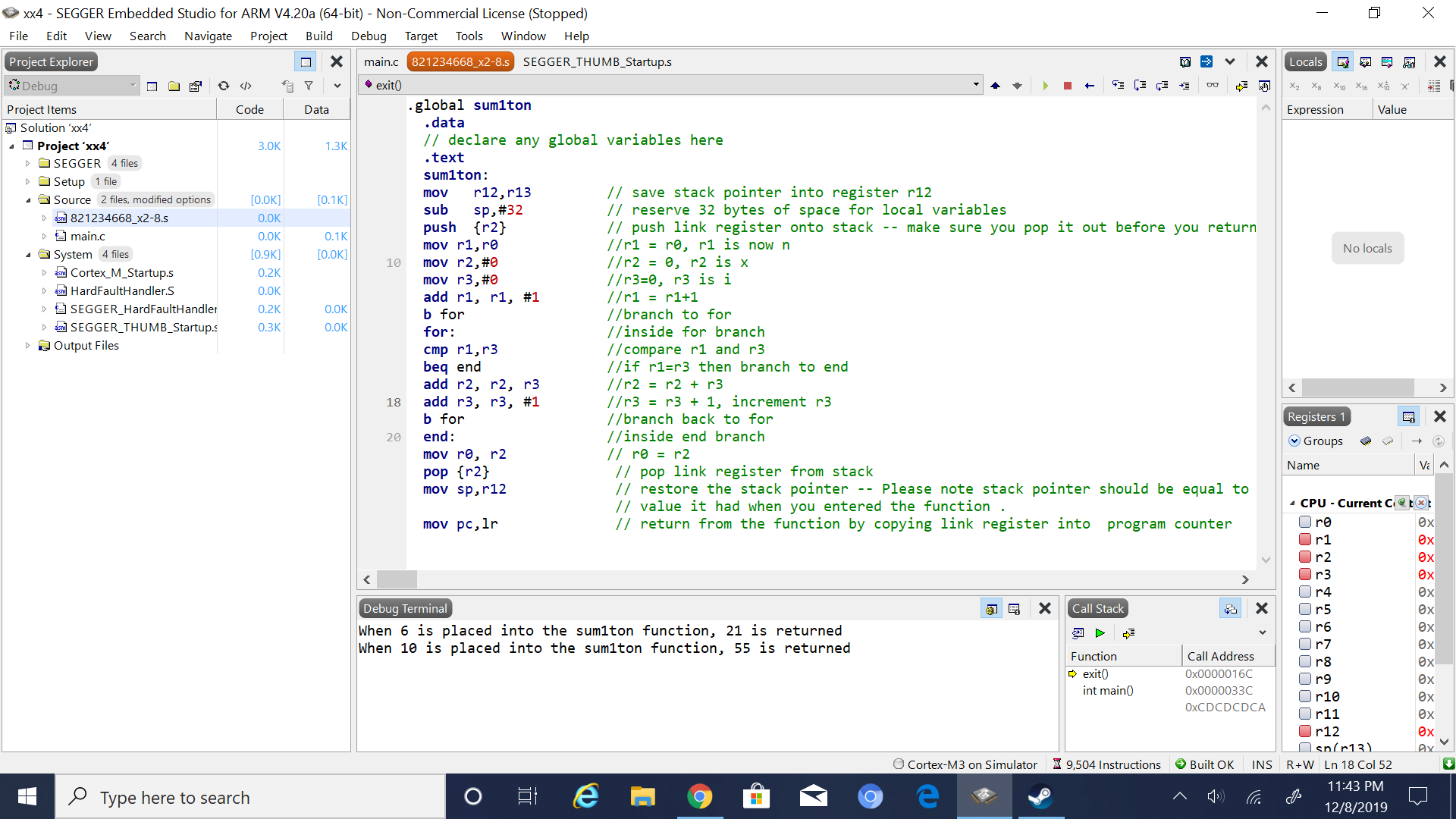
pop {r2} // pop link register from stack

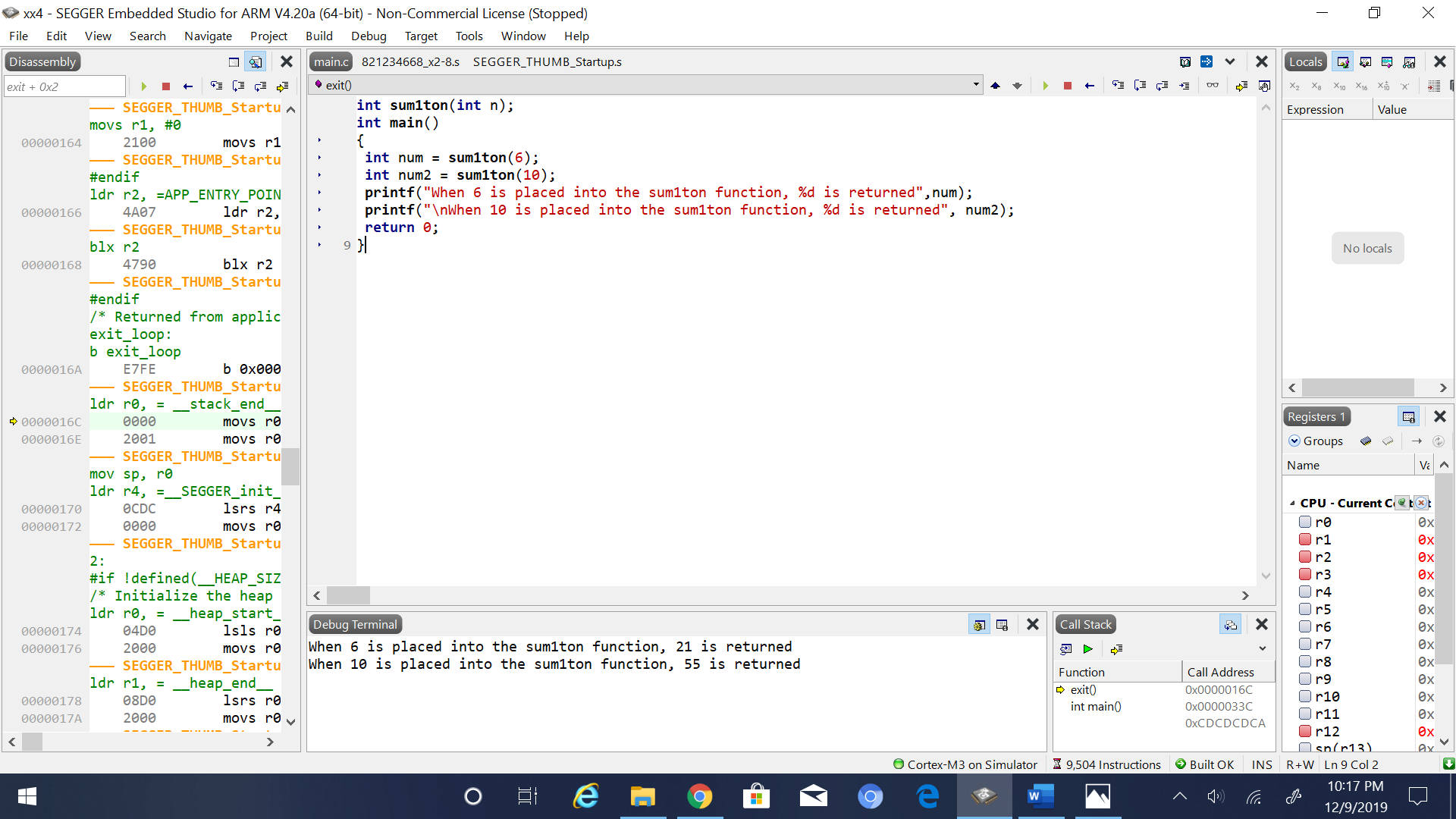
mov sp,r12 // restore the stack pointer -- Please note stack pointer should be equal to the

// value it had when you entered the function .

mov pc,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