**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.*

12/10/2019

[HW#X2-11]

[X2-11]

Ckick below to enter/change your Name and RedID

Ethan Nagelvoort, 821234668

**Content**

(\* - Mandatory)

1\*. Description of the problem/method

I use a while loop to loop through the int array. In the loop, I square each value and add them all together to a register which represents sum. To detect overflow, I compare the previous sum from the previous loop to the new sum gotten through the current loop. If the previous sum is greater than the new/current sum, then overflow is detected and the loop would branch to a different branch where -1 would be returned. If overflow was not detected throughout the course of the while loop, then program will branch out and sum will be returned.

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

sum\_squares:

save stack pointer into register r12

reserve 32 bytes of space for local variables

r6=0, r6 will equal previous value in loop to help detect overflow

r4=0, r4=i

r2=0, r2 = sum

load r0 into r3

inside while branch

compare r4,r1

if r4 = r1, then branch to done

unsigned multiplication: r3=r3\*r3

r2=r2+r3

compare r6 and r2

if r6>r2, then branch to over

r4= r4+1, increment r4

r7=r4<<2, r7=r4+4

load mem(r0+r7) into r3

r6=r2

branch back to while

inside over branch

r0=0

r0=r0-1

branch to end

inside done branch

r0=r2

branch to end

inside end branch

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

#include <stdio.h>

#include <stdlib.h>

#include <stdint.h>

#define SIZE 3

//int sum\_squares(int \*x, SIZE);

int main(void)

{

int arr[] = {1,2,3};

int \*ptr = &arr;

int sum = sum\_squares(ptr,SIZE);

int arr2[] = {2,3,4};

int \*ptr2 = &arr2;

int sum2 = sum\_squares(ptr2,SIZE);

printf("If an array of {1,2,3} is placed into the function and the size is 3, the sum is %d",sum);

printf("\nIf an array of {2,3,4} is placed into the function and the size is 3, the sum is %d",sum2);

printf("\nKeep in mind that if the sum equals -1, then overflow has occured.");

return 0;

}

//function file

.global sum\_squares

.data

// declare any global variables here

.text

sum\_squares:

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 r6, #0 //r6=0, r6 will equal previous value in loop to help detect overflow

mov r4, #0 //r4=0, r4=i

mov r2, #0 //r2=0, r2 = sum

ldr r3, [r0] //load r0 into r3

while: //inside while branch

cmp r4,r1 //compare r4,r1

beq done //if r4 = r1, then branch to done

umull r3,r5,r3,r3 //unsigned multiplication: r3=r3\*r3

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

cmp r6, r2 //compare r6 and r2

bgt over //if r6>r2, then branch to over

add r4, r4, #1 //r4= r4+1, increment r4

lsl r7, r4, #2 //r7=r4<<2, r7=r4+4

ldr r3, [r0, r7] //load mem(r0+r7) into r3

mov r6,r2 //r6=r2

b while //branch back to while

over: //inside over branch

mov r0, #0 //r0=0

sub r0, r0, #1 //r0=r0-1

b end //branch to end

done: //inside done branch

mov r0, r2 //r0=r2

b end //branch to end

end: //inside end branch

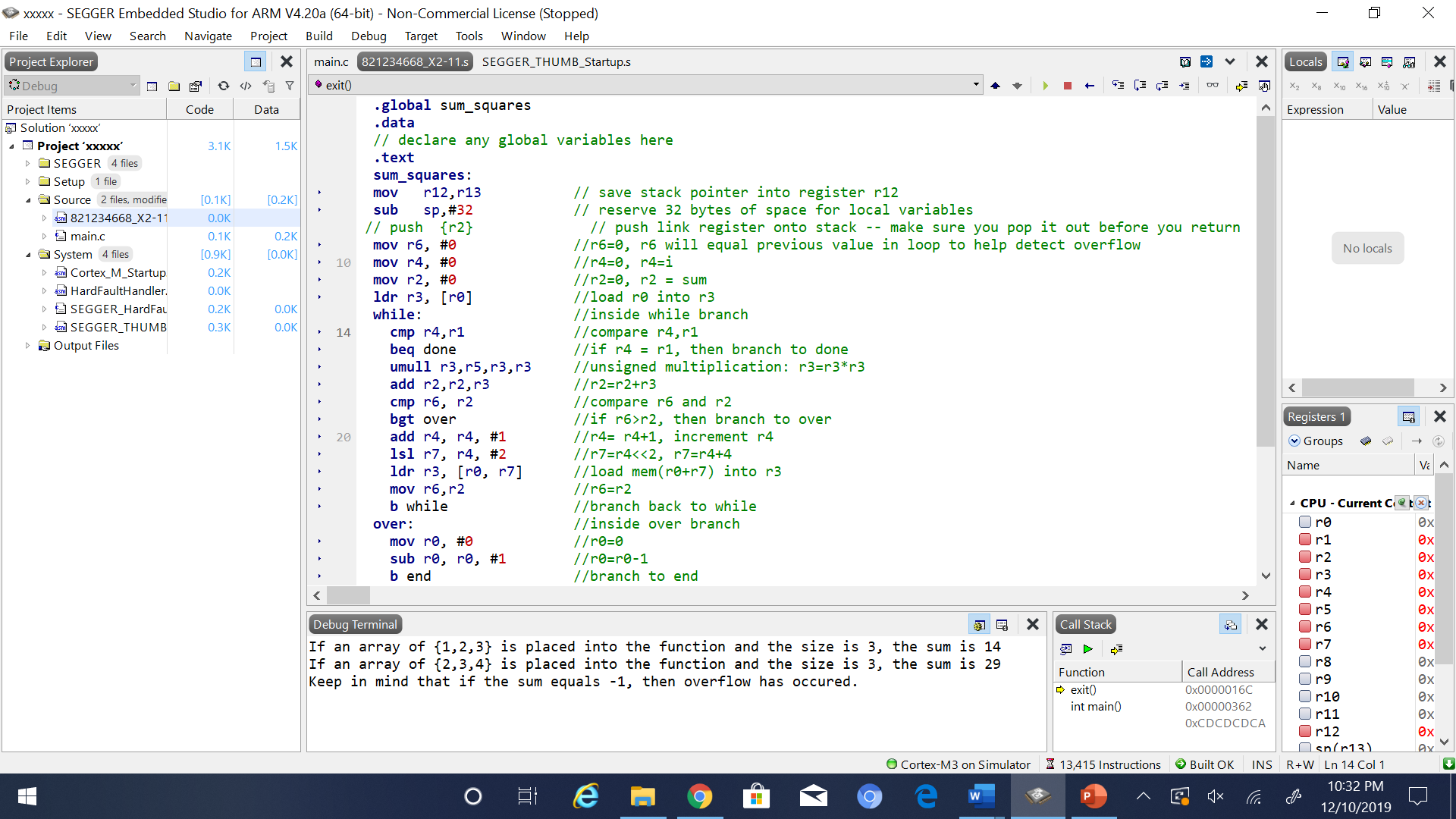
//pop {r2} // pop link register from stack

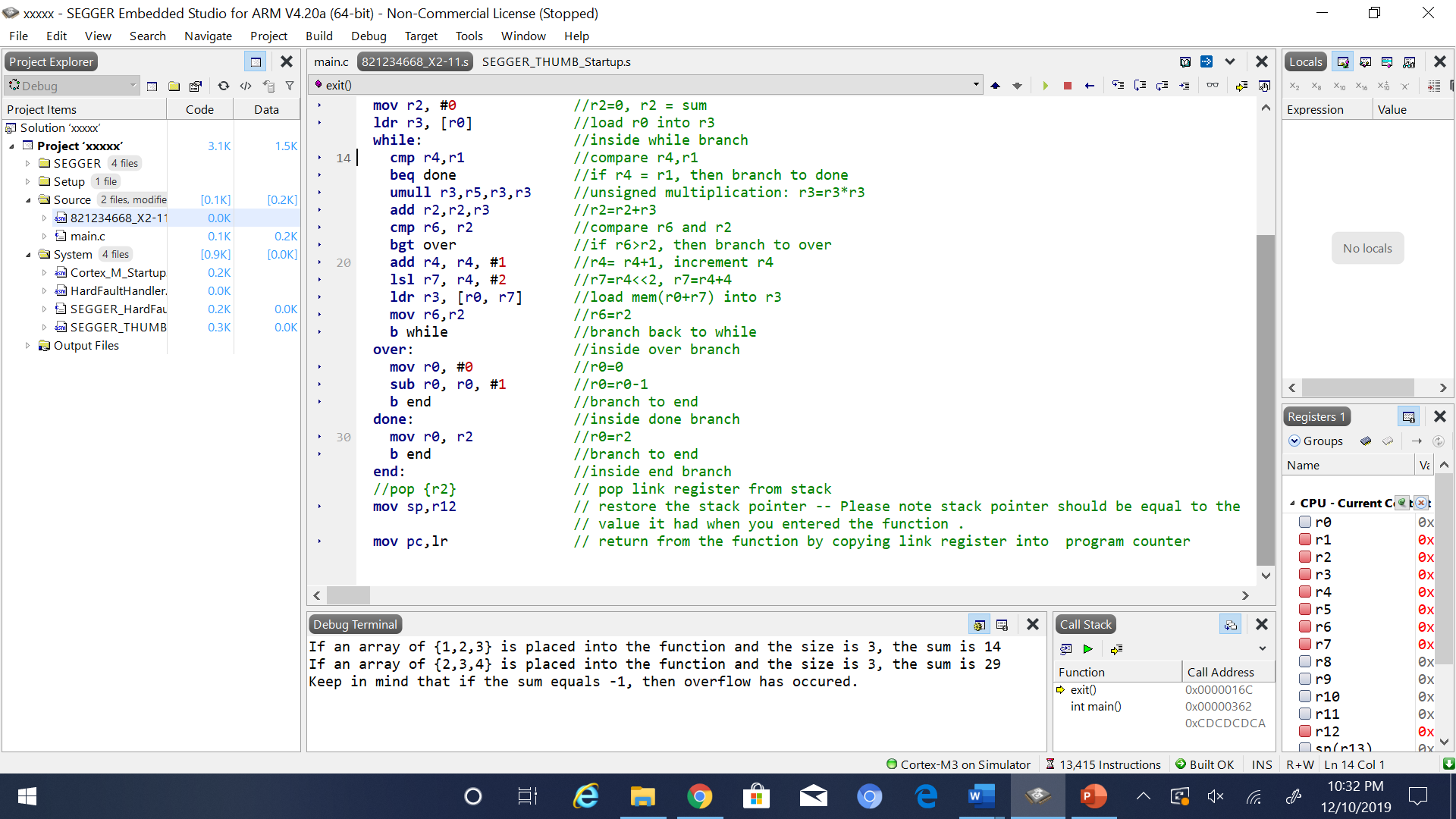
mov sp,r12 // restore the stack pointer -- Please note stack pointer should be equal to the

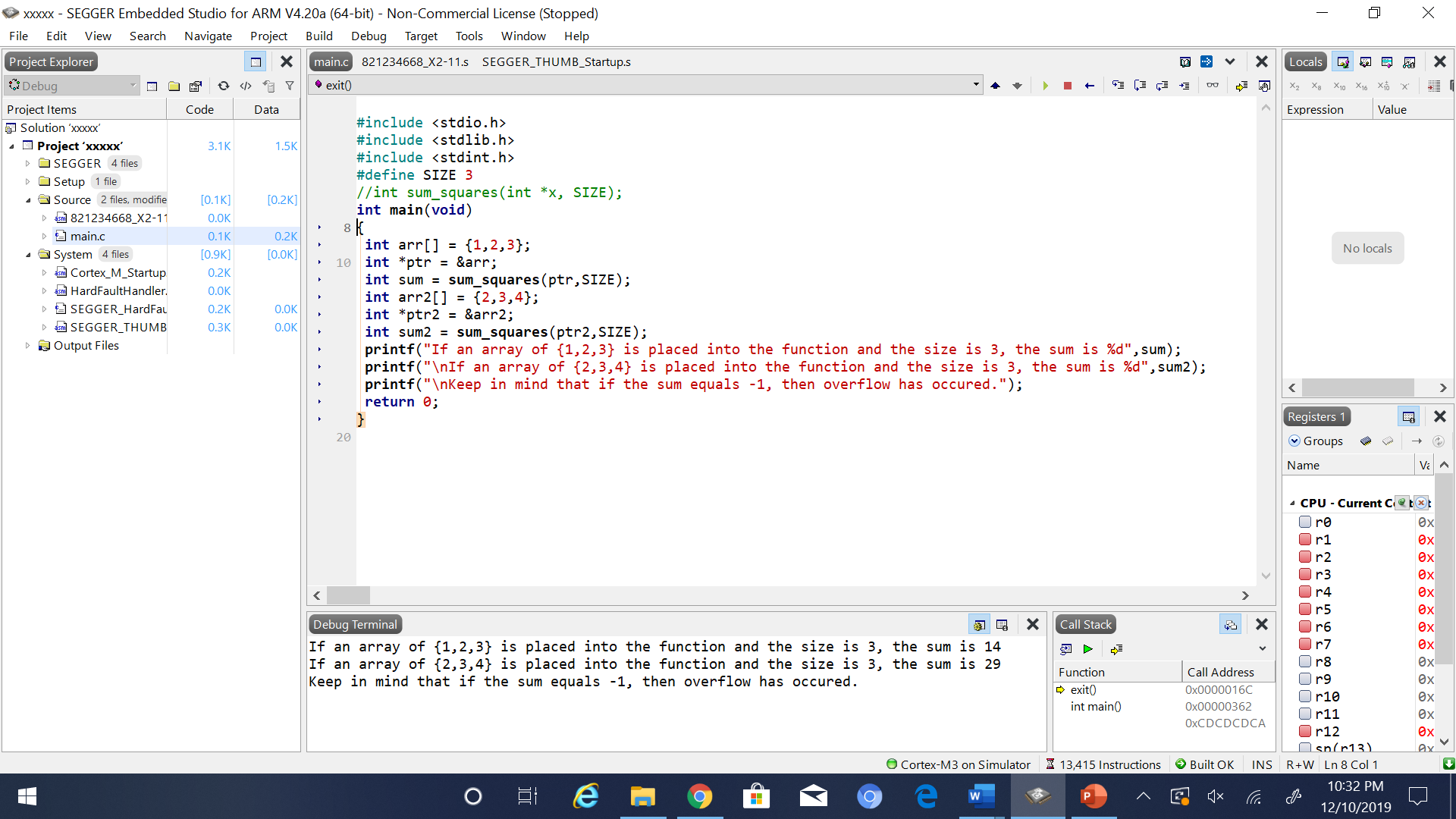
// value it had when you entered the function .

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 Arnold’s slides