**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#X2-9]

[X2-9]

Ckick below to enter/change your Name and RedID

Ethan Nagelvoort, 821234668

**Content**

(\* - Mandatory)

1\*. Description of the problem/method

I set r1 to 0, which will be i. I then load, using ldrb since we are using chars, r0 into r2. The program then enters a while branch, where r1 is incremented and mem(r0,r1) is loaded into r2, so r2 would hold the next char in the string. The program would branch out of the while branch when r2 = 0 because the end of a string is 0. Then r0 would equal r1, which would return the length of the string.

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

strlen:

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=0, r1=i

r2 = mem(r0), use ldrb cause a char is one bit

branch to while

inside while branch

compare r2 and 0

if r2=0, branch to done. 0 means end of string

if r2!=0 branch to cont

inside cont branch:

r1=r1+1, increment r1. r1 is a counter that represents string length

r2=mem(r0+r1)

branch back to while

inside done branch

r0=r1, return the value of r1

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 strlen(char\*string);

int main()

{

char\* word = "thought";

char\* word2 = "Engineering";

int len = strlen(word);

int len2 = strlen(word2);

printf("The length of the word %s is %d",word, len);

printf("\nThe length of the word %s is %d", word2, len2);

}

//function file

.global strlen

.data

// declare any global variables here

.text

strlen:

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

mov r1, #0 //r1=0, r1=i

ldrb r2,[r0] //r2 = mem(r0), use ldrb cause a char is one bit

b while //branch to while

while: //inside while branch

cmp r2,#0 //compare r2 and 0

beq done //if r2=0, branch to done. 0 means end of string

bne cont //if r2!=0 branch to cont

cont: //inside cont branch:

add r1, r1, #1 //r1=r1+1, increment r1. r1 is a counter that represents string length

ldrb r2, [r0,r1] //r2=mem(r0+r1)

b while //branch back to while

done: //inside done branch

mov r0,r1 //r0=r1, return the value of r1

pop {r1} // pop link register from stack

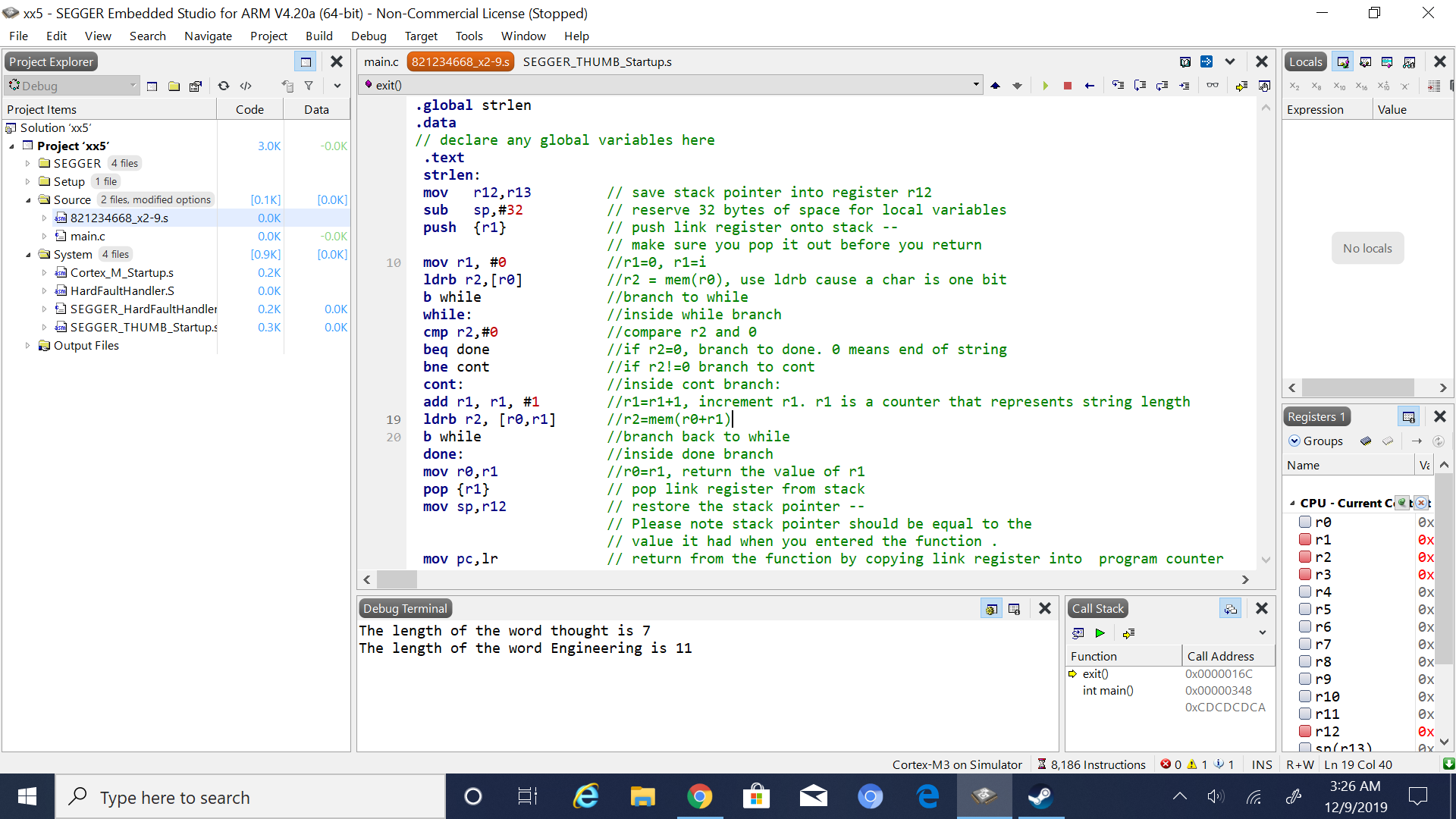
mov sp,r12 // restore the stack pointer --

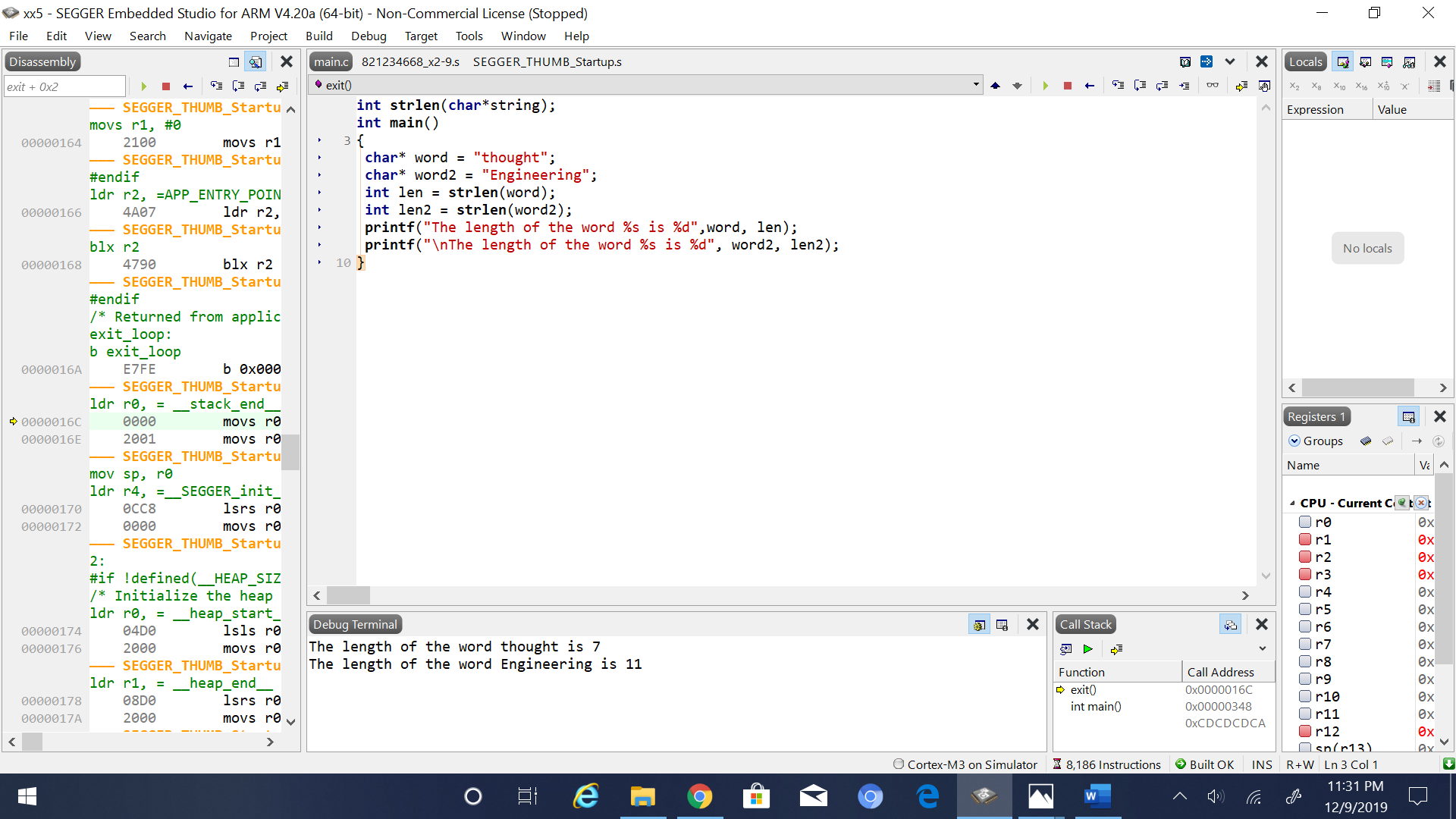
// Please note stack pointer should be equal to the

// value it had when you entered the function .

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