**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.*

11/15/2019

Homework 9

hw9

Ckick below to enter/change your Name and RedID

Ethan Nagelvoort, 821234668

**Content**

(\* - Mandatory)

1\*. Description of the problem/method

I first use cmp to check if seed/r0 is equal to 0 or not. If it isn’t then branch to if, if it is then increment seed and return it. In the if branch, I set the a registers that represents start state to equal seed, lfsr to equal start state, bit, and period to equal 0. Then I branch to a do branch. In it I lsr lfsr by 0, 2, 6, and 7 and exor all those values. I lsr by these values because it relates to the give taps. I place this value into bit. I then (lfsr >> 1) | (bit << 31) because the program is working with bits. I also increment period. I then branch to a while branch. In it is see if lfsr and seed are equal. If so then return period, if not then branch back to do.

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

lfsr32:

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

branch to if branch if r0!=0

r1=r0

increment r1

r0=r1

branch to end

inside if branch:

r1=r0

r2=r1

r3=0

r4=0

branch to do

inside do branch:

r5 = r2 >> 0

r6 = r2 >> 2

r3 = r6

r3 = r5 ^ r6

r5 = r2 >> 6

r6 = r2 >> 7

r3 = r3 ^ r5

r3 = r3 ^ r6

r5 = r2 >> 1

r3 = r2 << 31

r3 = r3 | r5

increment r4

branch to while

inside while branch:

branch to do if r1 != r2

r0=r4

branch to end

inside end branch

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

//Arm version of code:

.global lfsr32

.data

// declare any global variables here

.text

lfsr32:

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

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

push {r4} //push link register onto stack -- make sure you pop it out before you return

cmp r0,#0

bne if //branch to if branch if r0!=0

mov r1,r0 //r1=r0

add r1,r1,#1 //increment r1

mov r0,r1 //r0=r1

b end //branch to end

if: //inside if branch:

mov r1,r0 //r1=r0

mov r2,r1 //r2=r1

mov r3,#0 //r3=0

mov r4,#0 //r4=0

b do //branch to do

do: //inside do branch:

lsr r5, r2, #0 //r5 = r2 >> 0

lsr r6, r2, #2 //r6 = r2 >> 2

mov r3, r6 //r3 = r6

eor r3, r5, r3 //r3 = r5 ^ r6

lsr r5, r2, #6 //r5 = r2 >> 6

lsr r6, r2, #7 //r6 = r2 >> 7

eor r3, r3, r5 //r3 = r3 ^ r5

eor r3, r3, r6 //r3 = r3 ^ r6

lsr r5, r2, #1 //r5 = r2 >> 1

lsl r3, r2, #31 //r3 = r2 << 31

orr r3, r3, r5 //r3 = r3 | r5

add r4, r4, #1 //increment r4

b while //branch to while

while: //inside while branch:

cmp r1, r2

bne do //branch to do if r1 != r2

mov r0, r4 //r0=r4

b end //branch to end

end: //inside end branch

pop {r4} //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

//c version of function

/\*

unsigned long cversionlfsr32(unsigned long seed)

{

//unsigned long is 32 bits

unsigned long answer = 0;

if(seed == 0)

{

seed++;

return seed;

}

unsigned long start\_state = seed;

unsigned long lfsr = start\_state;

unsigned long bit;

unsigned long period = 0;

do

{ /\* taps: 32 30 26 25; feedback polynomial: x^32 + x^30 + x^26 + x^25 + 1 \*/

/\* bit = ((lfsr >> 0) ^ (lfsr >> 2) ^ (lfsr >> 6) ^ (lfsr >> 7)) /\* & 1u \*/;

/\* lfsr = (lfsr >> 1) | (bit << 31);

++period;

}

while (lfsr != start\_state);

return period;

}\*/

//main

#include <stdio.h>

#include <stdlib.h>

void main()

{

unsigned long seed = 0x5AA5FF00;

unsigned long rand = lfsr32(seed);

printf("%lx\n",rand);

for(int i=0; i<20;i++)

{

unsigned long seed1 = rand;

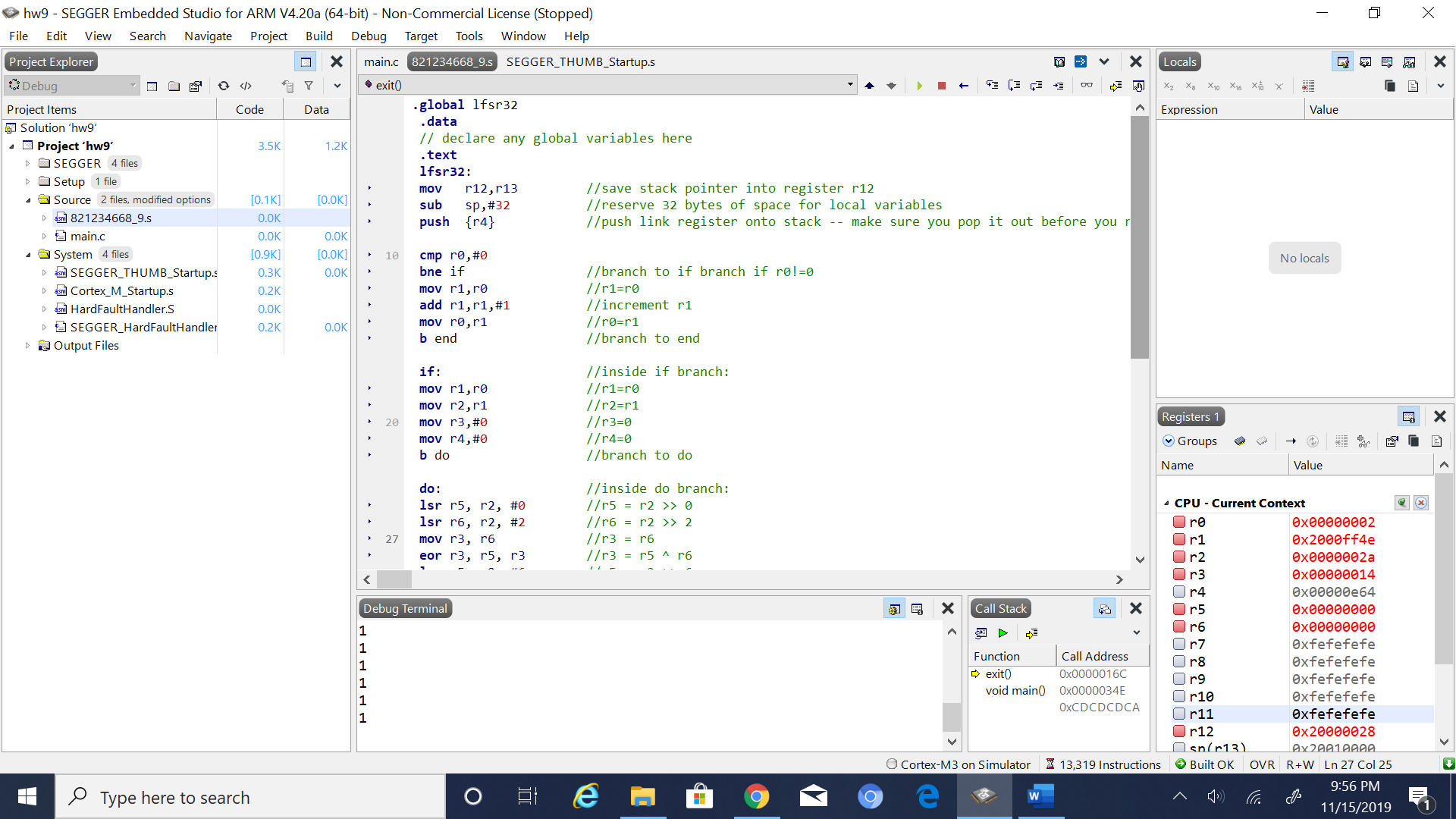
rand = lfsr32(seed1);

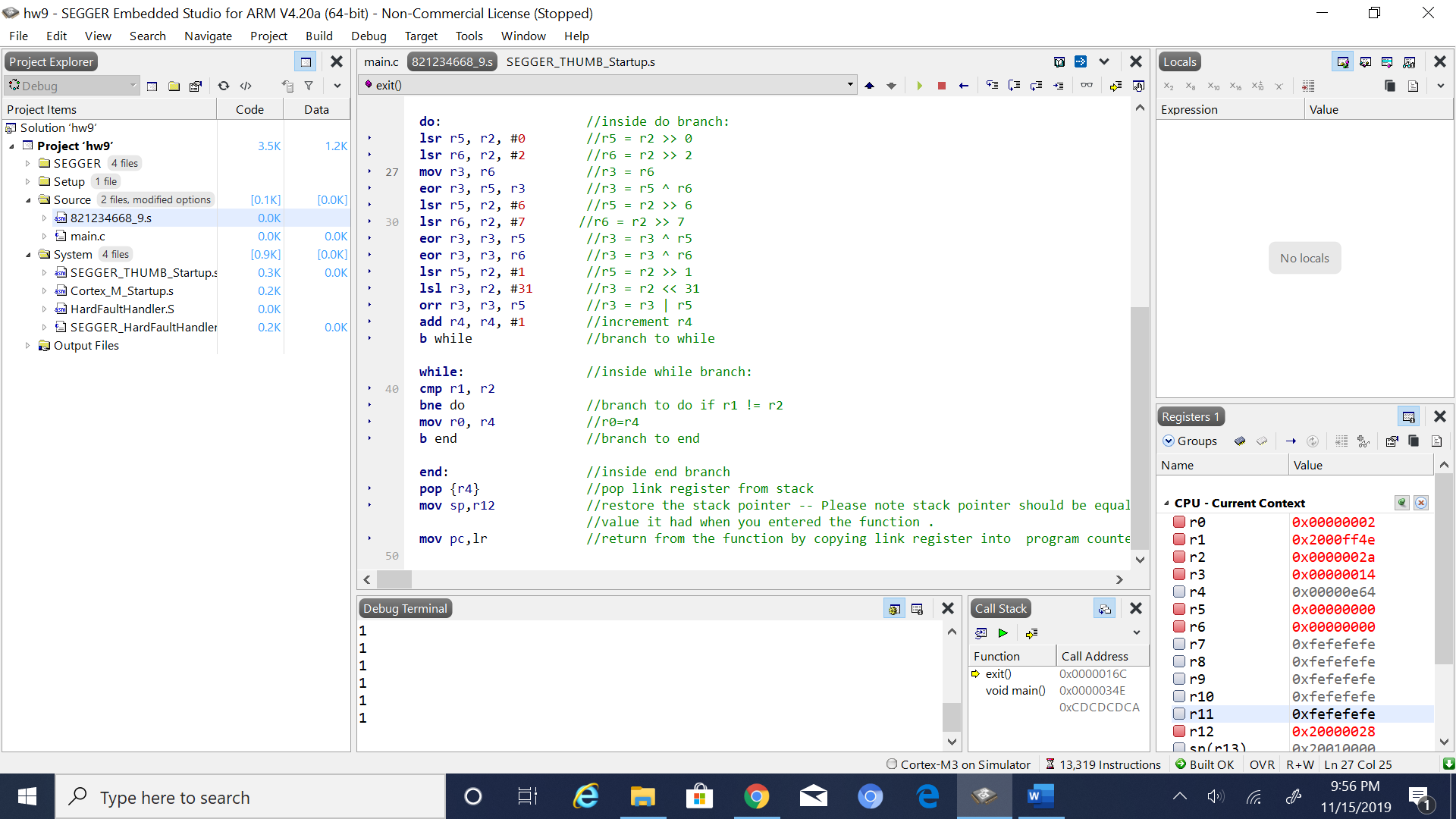
printf("%lx\n",rand);

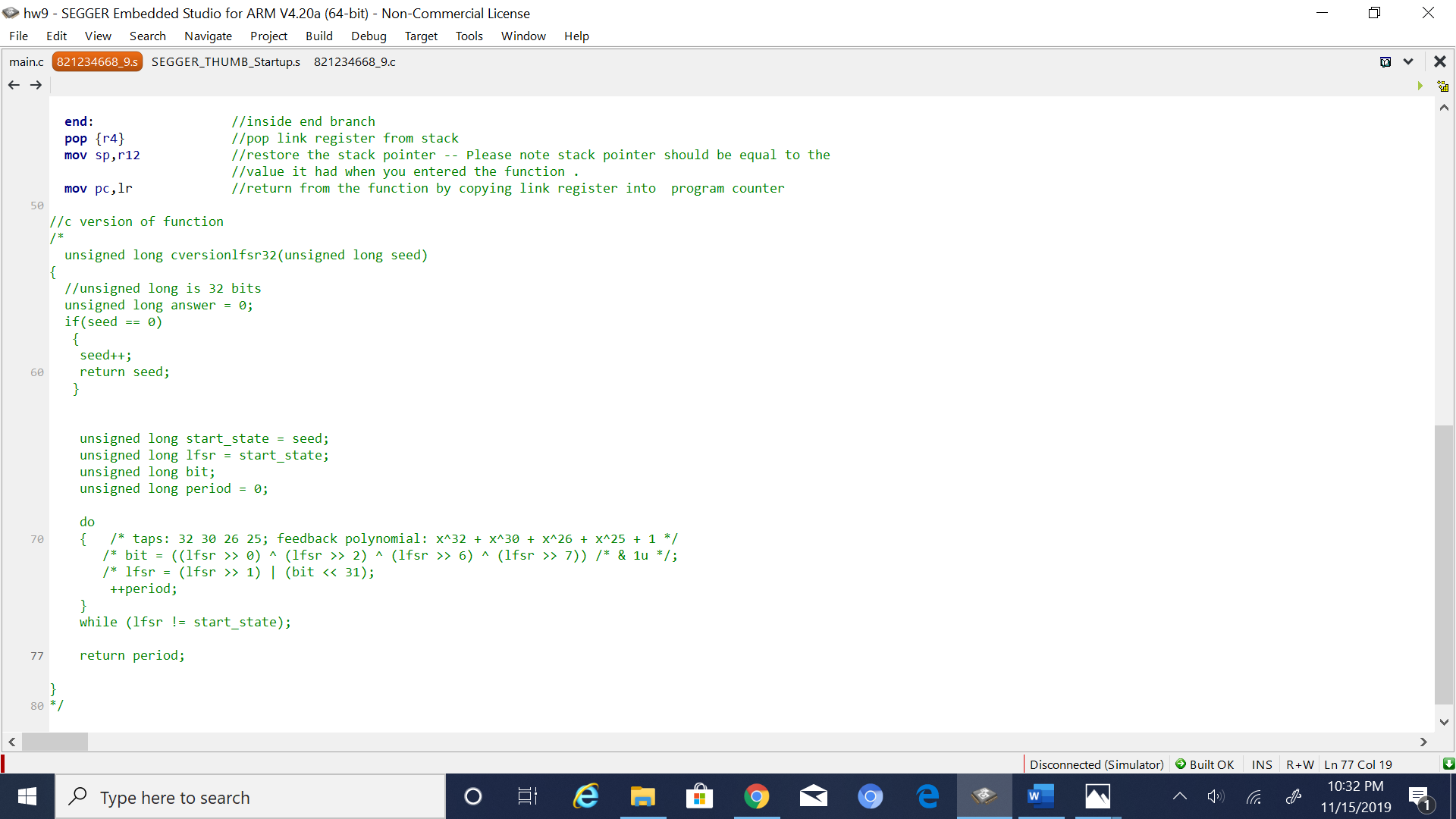
}

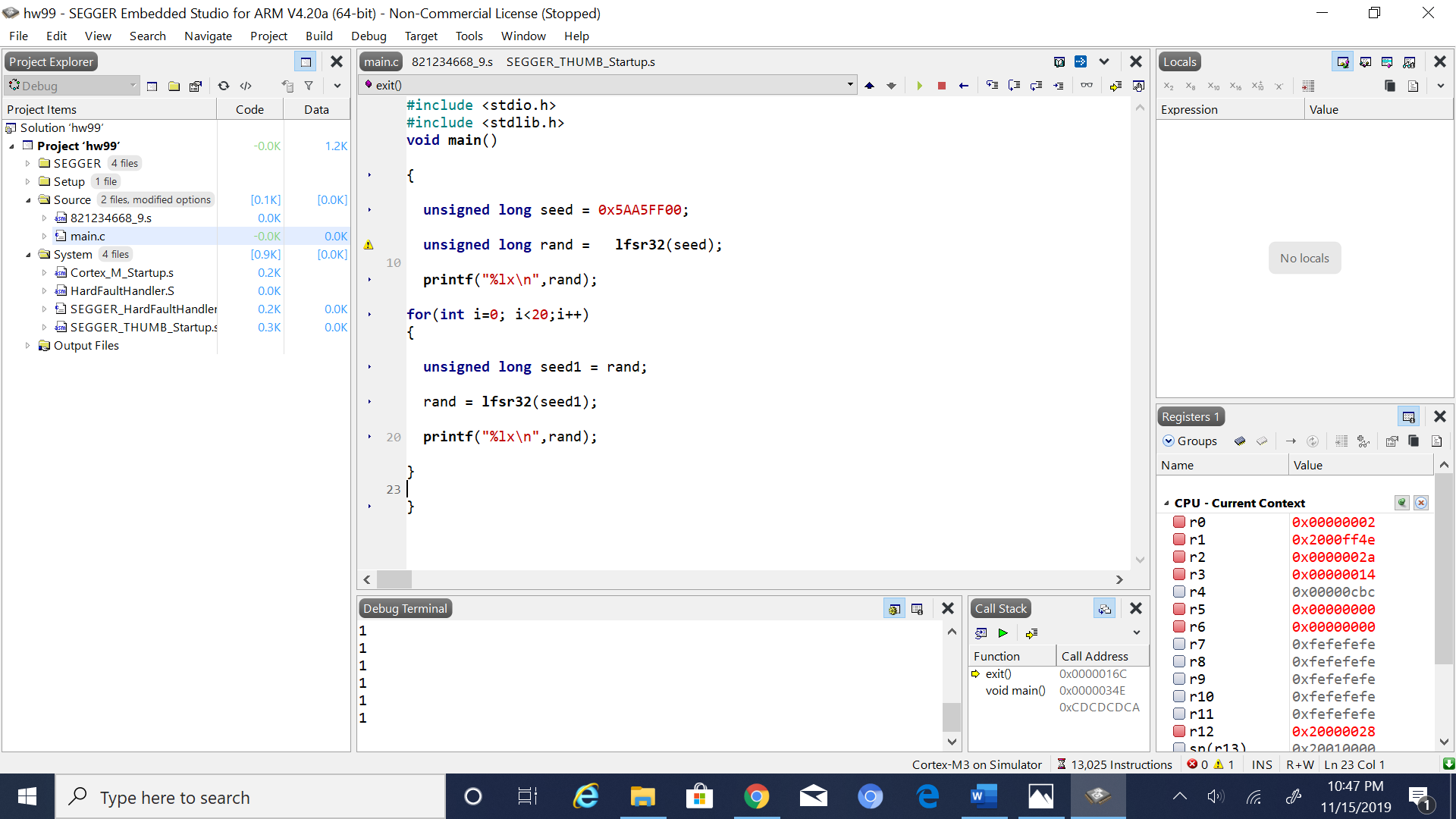
}

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









5. Conclusion (if applicable)

6\*. References.

The Wikipedia link given on blackboard