Assignment 4

Modibonacci

The goal of this program is to create a modified Fibonacci value calculator using the stack and recursive calls. The value of the Nth Modibonacci = (N-3)/2 + (N-2)/2 + (N-1).

The first phase of my Assign4.asm program handles memory location. It then gets and checks the previous value of N. If N matches a base case then it adds that value to eax and pops to the previous ebp. If a base case is not found, then it goes into the primary method L1.

L1:

The first portion of L1 gets the value of (N-1). It subtracts the eax value it receives by 1 and then recursively calls Modibonacci until it calculates the value of (N-1). Once the value is found it saves that value to local storage, and begins to calculate (N-2). Storing it into eax

Once (N-2) is calculated, the value (N-1) is stored in ebx. The value in eax is (N-2), so then we shr eax to get (N-2)/2. Eax and ebx are added together then stored in local storage. It then begins to calculate (N-3).

Once (N-3) is calculated, the value of ( (N-1) + (N-2)/2 ), previous local value, is moved to ebx. Eax now holds the value of (N-3), so we shr on eax to get (N-3)/2. Eax and ebx are added together. This ends the recursive call and the original c++ ebp is restored. Eax is passed to main.cpp

Main.cpp

#include <iostream>

using namespace std;

extern "C" unsigned int Modibonacci(unsigned int n);

void main()

{

unsigned int N;

cout << "Calculate the Nth Modibonacci: ";

cin >> N;

cout << N << " Modibonacci = " << Modibonacci(N);

cout << endl;

}

Assign4.asm

TITLE Calculating Modibonacci (Assign4.asm)

;Chad Bartley

; This program uses recursion to calculate the

; Nth Modibonacci value

.586

.model flat,C

.code

Modibonacci PROC

push ebp ;

mov ebp, esp

sub esp, 4 ;reserve space for local variable

mov eax, [ebp + 8] ; get N

;Base case N = 3

cmp eax, 3

je N3

;Base case N = 2

cmp eax, 2

je N2

;Base case N = 1

cmp eax, 1

je N1

L1:

dec eax ; N - 1

push eax

call Modibonacci ; Alpha recursive call

add esp, 4 ; counter "push eax"

mov [ebp - 4], eax ; save eax to local

mov eax, [ebp + 8] ;Get N

sub eax, 2 ; N - 2

push eax

call Modibonacci ; Beta recursive call

add esp, 4 ; counter "push eax"

mov ebx, [ebp - 4] ; get previous local value (N - 1)

shr eax, 1 ; (N - 2)/2

add eax, ebx ; (N - 1) + (N - 2)/2

mov [ebp - 4], eax ;save (N - 1) + (N - 2)/2 to local

mov eax, [ebp + 8]

sub eax, 3 ;N - 3

push eax

call Modibonacci ; Gamma recursive call

add esp, 4

mov ebx, [ebp - 4] ; put previous value into ebx (N - 1) + (N - 2)/2

shr eax, 1 ; (N - 3)/ 2

add eax, ebx ; (N - 1) + (N - 2)/2 + (N - 3)/ 2

jmp Quit

N1:

mov eax, 1 ; First Modibonacci = 1

jmp Quit

N2:

mov eax, 2 ; Second Modibonacci = 2

jmp Quit

N3:

mov eax, 4 ; Third Modibonacci = 4

jmp Quit

Quit:

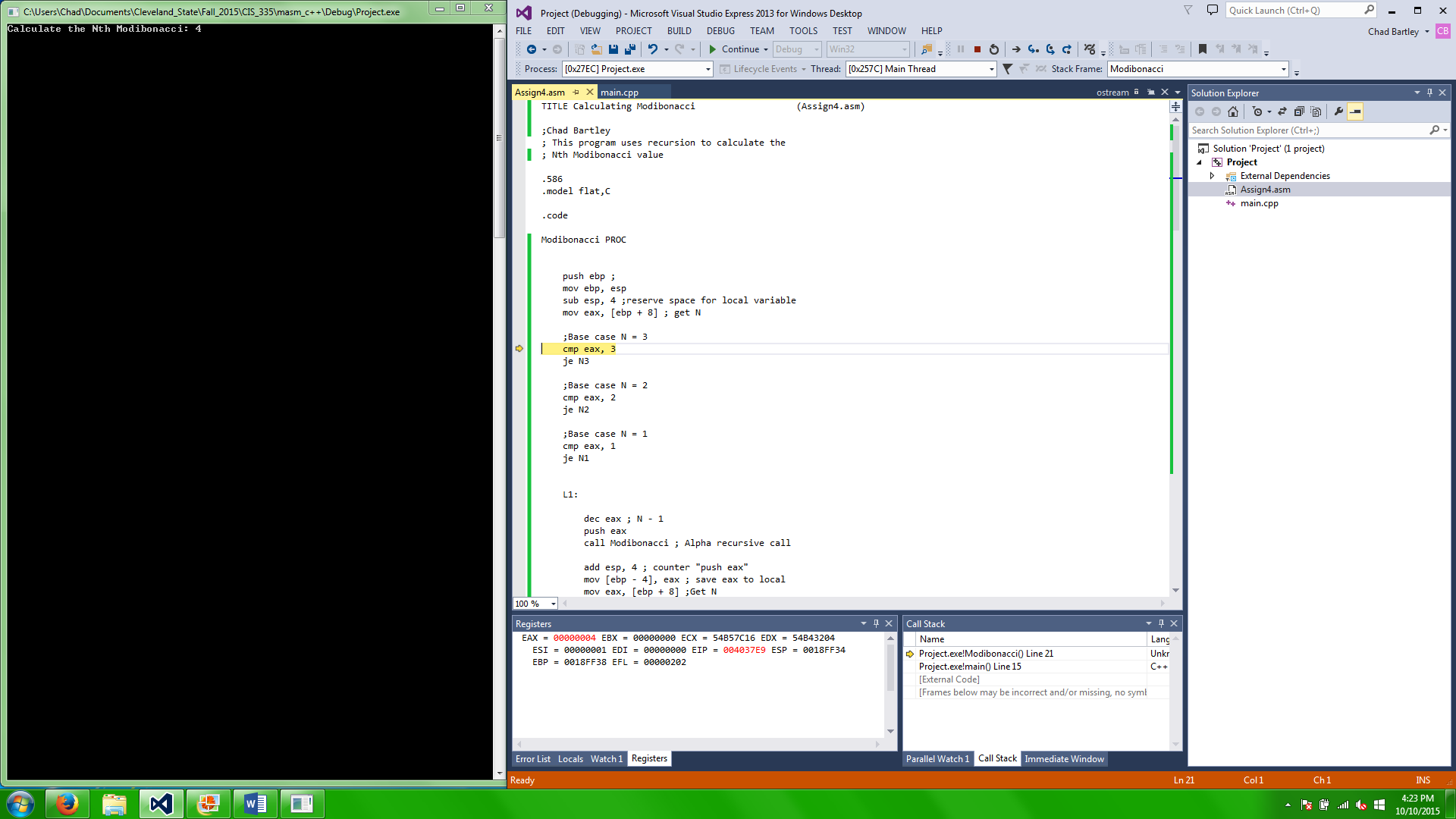
mov esp, ebp

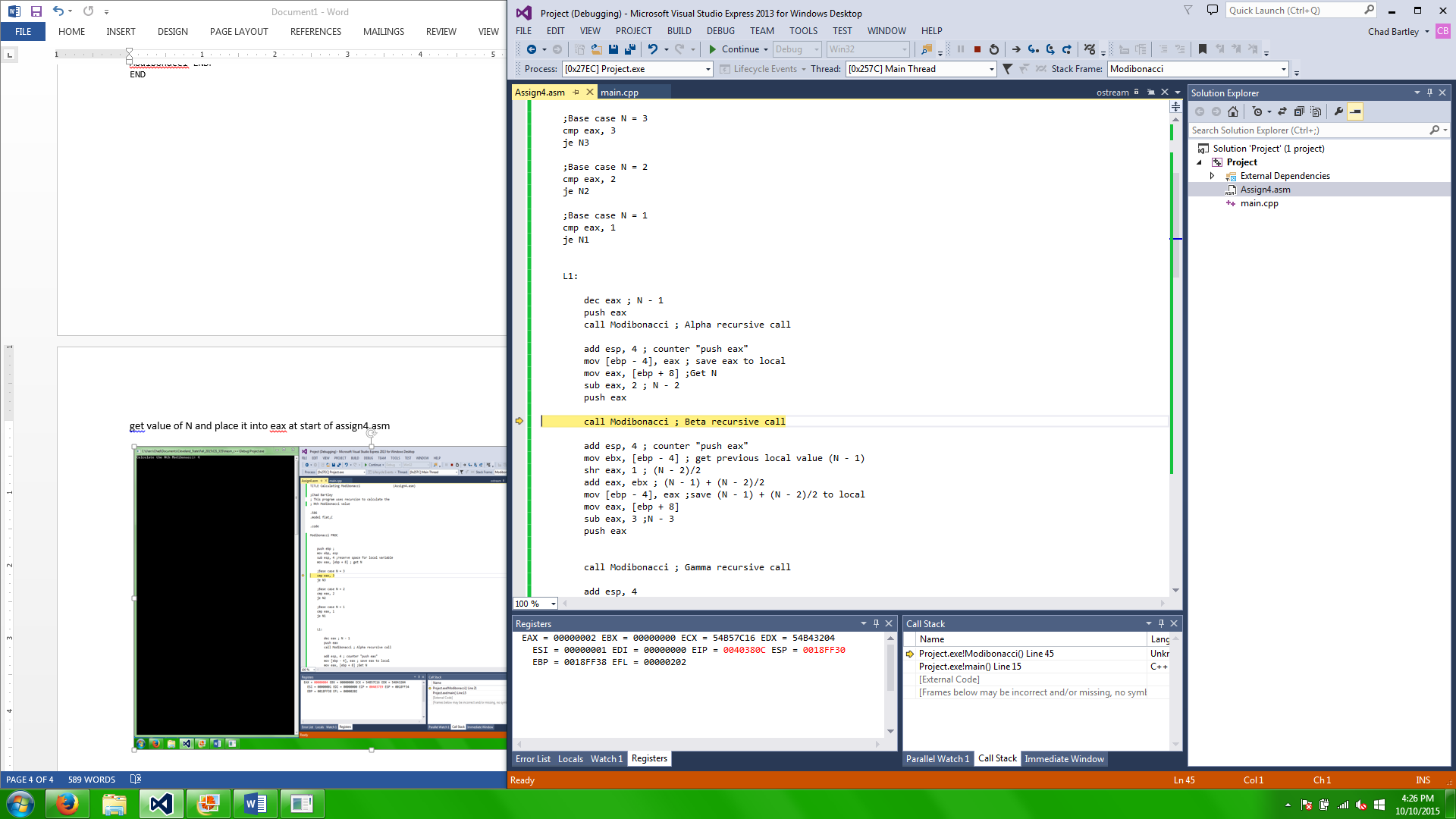
pop ebp

ret

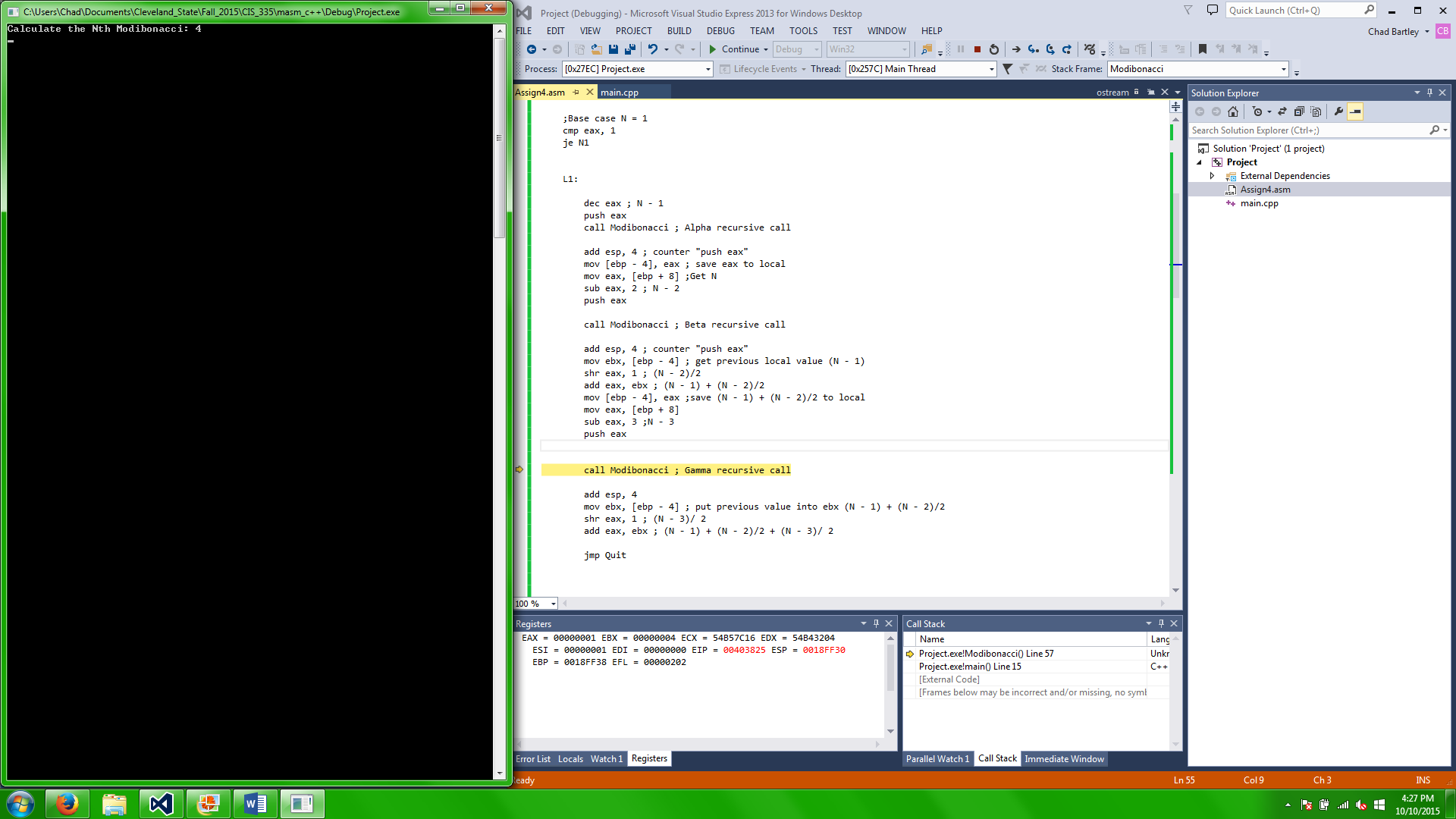
Modibonacci ENDP

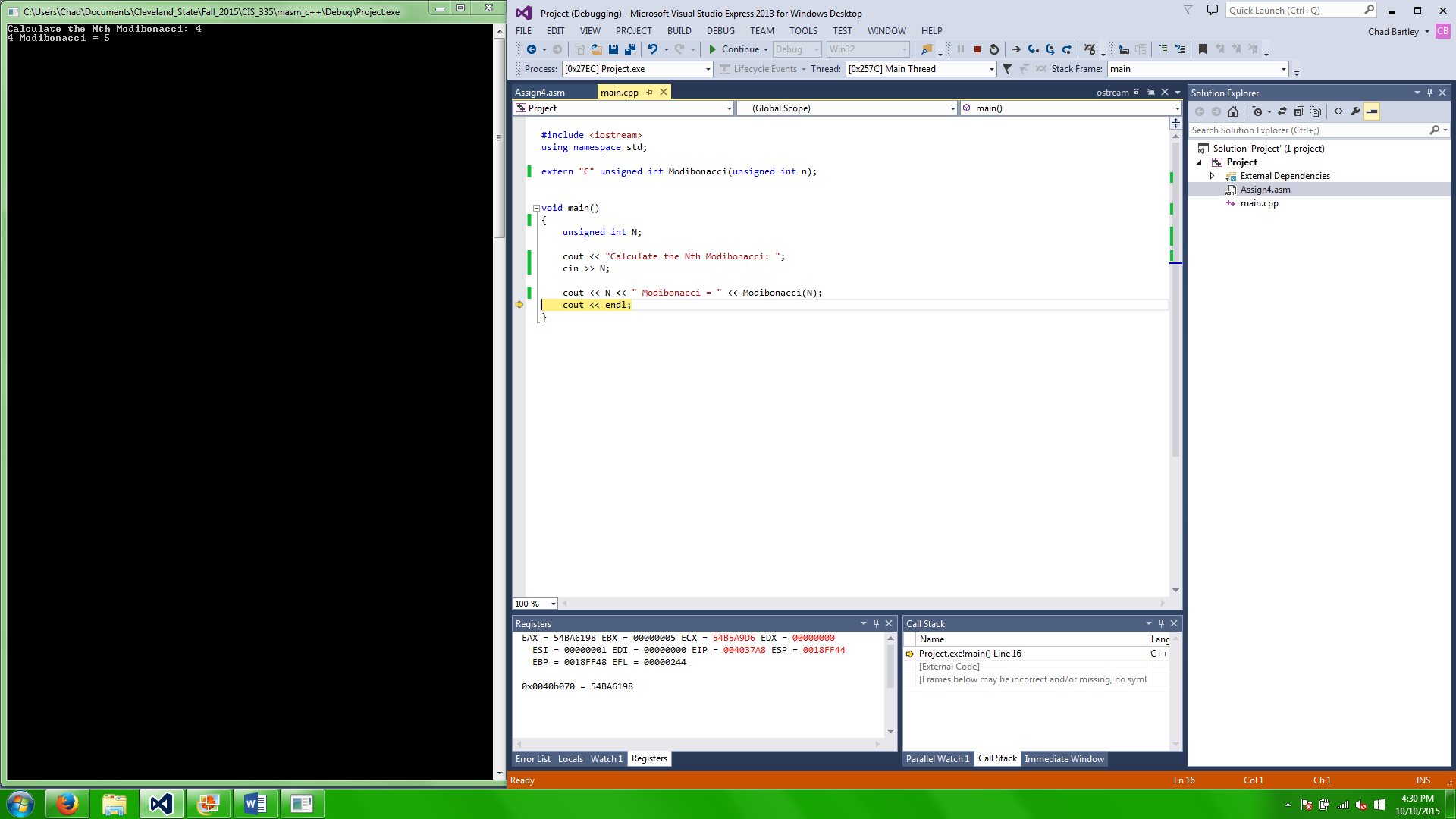
END

get value of N and place it into eax at start of assign4.asm

Finished first phase of L1, now the Beta recursive call is made

Second Phase of L1 finished, now it calls the gamma recursive call



Third phase of L1 has completed and the value for the Nth Modibonacci is placed into eax. Output is correct in the console.

The main difficulty that I had with this assignment was accessing the proper memory location. Both to save and retrieve previous values calculated.

I also had difficulty calculating the correct value. My first several attempts did not include use of the ebx register, but I shortly realized that it was essential. The use of ebx allowed me to store previously calculated aspects of the Modibonacci ( (N-1), ( (N-1)+(N-2)/2), etc ) and add them to the currently calculated value stored in eax. Then by saving eax to local storage I was able to transition to the next phase with all the information needed



Chad Bartley

chbartle

CSU ID: 2589157

Assignment 4