CS 410 C++ to Assembly with Loops Activity

C++ Code Functionality	
C++ Line of Code	**Explanation of Functionality**
#include<iostream>	Includes standard input/output header file, allowing input and output operations.
using namespace std;	Makes symbols from std namespace accessible without needing to prefix them.
int main()	Starting point of program, returns integer when finished.
{	
int num, i;	Declares integers named num and i with no initial values.
int product = 1;	Declares integer named product with a value of 1.
cout << "Enter a number:\n" << endl;	Prints out a statement, ending with 2 newline characters.
cin >> num;	Reads in user input and assigns value to num.
for(i = num ; i > 0; i--)	Using i as the index, loops num number of times.
product = product * i;	Each loop iteration multiply product by i and assign result to product.
cout << "The factorial for " << num << " is: " << product << endl;	Prints out concatenation of statement, num value, and product value.
return 1;	Signals an error during the execution of the program and exits the main function.
}	

C++ to Assembly Alignment	
C++ Line of Code	**Blocks of Assembly Code**
int main()	main: .LFB1493:

int num, i;	movq %rsp, %rbp
	movq %fs:40, %rax
	movq %rax, -8(%rbp)

int product = 1;	movl $1, -12(%rbp)
cout << "Enter a number:\n" << endl;	.LC0: .string "Enter a number:\n" leaq .LC0(%rip), %rsi leaq _ZSt4cout(%rip), %rdi
cin >> num;	movq %rax, %rsi leaq _ZSt3cin(%rip), %rdi

for(i = num ; i > 0; i--)	.L3:
	cmpl $0, -16(%rbp)
	jle .L2
	…
	subl $1, -16(%rbp)
	jmp .L3

product = product * i;	movl -12(%rbp), %eax
	imull -16(%rbp), %eax
	movl %eax, -12(%rbp)

cout << "The factorial for " << num << " is: " << product << endl;	.LC1:
	.string "The factorial for "
	.LC2:
	.string " is: "
	.L2:
	leaq .LC1(%rip), %rsi
	leaq _ZSt4cout(%rip), %rdi
	call _ZStlsISt11char_traitsIcEERSt13basic_ostreamIcT_ES5_PKc@PLT
	movq %rax, %rdx
	movl -20(%rbp), %eax
	movl %eax, %esi
	movq %rdx, %rdi
	call ZNSolsEi@PLT
	leaq .LC2(%rip), %rsi
	movq %rax, %rdi
	call _ZStlsISt11char_traitsIcEERSt13basic_ostreamIcT_ES5_PKc@PLT
	movq %rax, %rdx
	movl -12(%rbp), %eax
	movl %eax, %esi
	movq %rdx, %rdi
	call _ZNSolsEi@PLT
	movq %rax, %rdx
	movq _ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_@GOTPCREL(%rip), %rax
	movq %rax, %rsi
	movq %rdx, %rdi
	call _ZNSolsEPFRSoS_E@PLT
	movl $1, %eax
	movq -8(%rbp), %rcx
	xorq %fs:40, %rcx

return 1;	movl $1, %edi .cfi_def_cfa 7, 8 ret .cfi_endproc

Assembly Functionality	
Blocks of Assembly Code	**Explanation of Functionality**
main: .LFB1493:	Main function declared.

movq %rsp, %rbp
movq %fs:40, %rax
movq %rax, -8(%rbp)
	Reserve space at %rbp for num and -8(%rbp) for i.

movl $1, -12(%rbp)	Assign value of 1 to -12(%rbp).
.LC0: .string "Enter a number:\n"	Define a string value.
leaq .LC0(%rip), %rsi leaq _ZSt4cout(%rip), %rdi	Print out string from .LC0.

movq %rax, %rsi
leaq _ZSt3cin(%rip), %rdi
call _ZNSirsERi@PLT
movl -20(%rbp), %eax
movl %eax, -16(%rbp)
	Read in user input and assign the value to -16(%rbp).

.L3:
cmpl $0, -16(%rbp)
jle .L2
…
subl $1, -16(%rbp)
jmp .L3
	Loop that iterates the number of times defined by the value stored in -16(%rbp).

movl -12(%rbp), %eax
imull -16(%rbp), %eax
movl %eax, -12(%rbp)
	Move -12(%rbp) to %eax, multiply -16(%rbp) by %eax, then assign result to -12(%rbp).

.LC1:
  .string "The factorial for "
.LC2:
  .string " is: "
.L2:
  leaq .LC1(%rip), %rsi
  leaq _ZSt4cout(%rip), %rdi
  call _ZStlsISt11char_traitsIcEERSt13basic_ostreamIcT_ES5_PKc@PLT
  movq %rax, %rdx
  movl -20(%rbp), %eax
  movl %eax, %esi
  movq %rdx, %rdi
  call ZNSolsEi@PLT
  leaq .LC2(%rip), %rsi
  movq %rax, %rdi
  call _ZStlsISt11char_traitsIcEERSt13basic_ostreamIcT_ES5_PKc@PLT
  movq %rax, %rdx
  movl -12(%rbp), %eax
  movl %eax, %esi
  movq %rdx, %rdi
  call _ZNSolsEi@PLT
  movq %rax, %rdx
  movq _ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_@GOTPCREL(%rip), %rax
  movq %rax, %rsi
  movq %rdx, %rdi
  call _ZNSolsEPFRSoS_E@PLT
  movl $1, %eax
  movq -8(%rbp), %rcx
  xorq %fs:40, %rcx
	Print out a concatenated string of .LC1 and .LC2 with the values of -12(%rbp) and -8(%rbp).

movl $1, %edi .cfi_def_cfa 7, 8 ret .cfi_endproc	Assign the value of 1 to %edi and return that value and end the program.