# ACTIVITY 12

Consider the following C++ program and the assembly code that Microsoft Visual C++ generates from it.

1. What calling convention is used for the function *f*?
2. Explain each instruction that is generated for *f* and *main*.
3. The function *f* takes three arguments: a `long` (a 4-byte integer variable), a `short` (2-byte integer), and a `char` (1-byte integer). How are these arguments passed to *f* in the call on line 8– i.e., what does the stack look like? How many bytes of stack space do these arguments occupy?
4. Now, considering your answer to question 3, how are the function's parameters accessed inside *f*?
5. Visual C++ compiles treats both `int` and `long` as 32-bit signed integers (SDWORDs). So, what effect does the cast on line 3 of the C++ code have on the generated code, if any?
6. In the assembly code, under the `; 14` comment, there are two instructions. Why is the value moved into EAX? Why didn't the compiler just generate the following instruction instead?

   `mov DWORD PTR _local2$[ebp], DWORD PTR _local1$[ebp]`

**main.cpp**

```
// Activity 12 -- C++ Code
//
// To generate assembly code from C++ code,
// 1. Change to the Release configuration
//    (to disable /GZ which reserves extra stack space)
// 2. In the project properties, navigate to
//    Configuration Properties > C/C++ > Output Files
//    and set Assembler Output to Assembly With Source Code
// 3. Build, and locate the .asm file in the Release folder

int f(unsigned long a, short b, char c) { // 1
    long local1 = a - b - c;              // 2
    int local2 = (int)local1;             // 3
    return local2;                        // 4
}                                         // 5
                                          // 6
int main() {                              // 7
    return f(10, 20, 30);                 // 8
}                                         // 9
```

**main.asm** (Microsoft Visual C++ non-optimized generated code for main.cpp)

```
; Listing generated by Microsoft (R) Optimizing Compiler Version 16.00.40219.01

	TITLE	C:\Users\jlo0012\Desktop\C++Project\main.c
	.686P
	.XMM
	include listing.inc
	.model flat

INCLUDELIB OLDNAMES

EXTRN	@__security_check_cookie@4:PROC
PUBLIC	_f
```

```
; Function compile flags: /Odtp
; File c:\users\jlo0012\desktop\c++project\main.c
;	COMDAT _f
_TEXT	SEGMENT
_local1$ = -8						; size = 4
_local2$ = -4						; size = 4
_a$ = 8							; size = 4
_b$ = 12						; size = 2
_c$ = 16						; size = 1
_f	PROC						; COMDAT

; 12   : int f(unsigned long a, short b, char c) {

	push	ebp
	mov	ebp, esp
	sub	esp, 8

; 13   :	long local1 = a - b - c;

	movsx	eax, WORD PTR _b$[ebp]
	mov	ecx, DWORD PTR _a$[ebp]
	sub	ecx, eax
	movsx	edx, BYTE PTR _c$[ebp]
	sub	ecx, edx
	mov	DWORD PTR _local1$[ebp], ecx

; 14   :	int local2 = (int)local1;

	mov	eax, DWORD PTR _local1$[ebp]
	mov	DWORD PTR _local2$[ebp], eax

; 15   :	return local2;

	mov	eax, DWORD PTR _local2$[ebp]

; 16   : }

	mov	esp, ebp
	pop	ebp
	ret	0
_f	ENDP
_TEXT	ENDS
PUBLIC	_main
; Function compile flags: /Odtp
;	COMDAT _main
_TEXT	SEGMENT
_main	PROC						; COMDAT

; 18   : int main() {

	push	ebp
	mov	ebp, esp

; 19   :	return f(10, 20, 30);

	push	30					; 0000001eH
	push	20					; 00000014H
	push	10					; 0000000aH
	call	_f
	add	esp, 12					; 0000000cH

; 20   : }

	pop	ebp
	ret	0
_main	ENDP
_TEXT	ENDS
END
```