# **CS 410 C++ to Assembly Activity Template**

Janera Dobson

Southern New Hampshire University

CS-410-T3249

**Step 1:** Explain the functionality of the C++ code.

## C++ Code Functionality

| **C++ Line of Code** | **Explanation of Functionality** |
| --- | --- |
| #include<iostream> | Tells preprocessor to include content of iostream at the very beginning of the program before compiler starts the actual compilation of the code. |
| using namespace std; | Tells the compiler that symbol names defined in the std namespace are brought into the programs scope. Therefore, you can omit the namespace qualifier. |
| int main() | Tells the compiler that our function needs to return some integer at the end of the execution, and we do so by returning 0 at the end of the program. |
| { | Tells the compiler it’s the opening of the scope to the main method. |
| int width=10; | Tells compiler to assign the value 10 to the variable named width which is of type int. |
| int height=5; | Tells the compiler to assign the value 5 to the variable name height of type int. |
| int area; | Tells thee compiler to declare a variable named area of type int. |
| area = width \* height; | Tells the compiler to multiply the values stored in width and height, then assign it to the variable named area. |
| cout<<endl<< area; | Tells the compiler to print out a newline followed by the new values stored in the variable named area. |
| return 0; | The return keyword is used to return some value and here it’s 0 which means success and will terminate the main function with this return statement but returning a non-zero number means failure. |
| } | Tells the compiler this is the end of the main methods scope. |

**Step 2:** Convert the C++ file into assembly code.

.file "assignment1\_1.cpp"

.text

.section .rodata

.type \_ZStL19piecewise\_construct, @object

.size \_ZStL19piecewise\_construct, 1

\_ZStL19piecewise\_construct:

.zero 1

.local \_ZStL8\_\_ioinit

.comm \_ZStL8\_\_ioinit,1,1

.text

.globl main

.type main, @function

main:

.LFB1493:

.cfi\_startproc

pushq %rbp

.cfi\_def\_cfa\_offset 16

.cfi\_offset 6, -16

movq %rsp, %rbp

.cfi\_def\_cfa\_register 6

subq $16, %rsp

movl $10, -12(%rbp)

movl $5, -8(%rbp)

movl -12(%rbp), %eax

imull -8(%rbp), %eax

movl %eax, -4(%rbp)

movq \_ZSt4endlIcSt11char\_traitsIcEERSt13basic\_ostreamIT\_T0\_ES6\_@GOTPCREL(%rip), %rax

movq %rax, %rsi

leaq \_ZSt4cout(%rip), %rdi

call \_ZNSolsEPFRSoS\_E@PLT

movq %rax, %rdx

movl -4(%rbp), %eax

movl %eax, %esi

movq %rdx, %rdi

call \_ZNSolsEi@PLT

movl $0, %eax

leave

.cfi\_def\_cfa 7, 8

ret

.cfi\_endproc

.LFE1493:

.size main, .-main

.type \_Z41\_\_static\_initialization\_and\_destruction\_0ii, @function

\_Z41\_\_static\_initialization\_and\_destruction\_0ii:

.LFB1979:

.cfi\_startproc

pushq %rbp

.cfi\_def\_cfa\_offset 16

.cfi\_offset 6, -16

movq %rsp, %rbp

.cfi\_def\_cfa\_register 6

subq $16, %rsp

movl %edi, -4(%rbp)

movl %esi, -8(%rbp)

cmpl $1, -4(%rbp)

jne .L5

cmpl $65535, -8(%rbp)

jne .L5

leaq \_ZStL8\_\_ioinit(%rip), %rdi

call \_ZNSt8ios\_base4InitC1Ev@PLT

leaq \_\_dso\_handle(%rip), %rdx

leaq \_ZStL8\_\_ioinit(%rip), %rsi

movq \_ZNSt8ios\_base4InitD1Ev@GOTPCREL(%rip), %rax

movq %rax, %rdi

call \_\_cxa\_atexit@PLT

.L5:

nop

leave

.cfi\_def\_cfa 7, 8

ret

.cfi\_endproc

.LFE1979:

.size \_Z41\_\_static\_initialization\_and\_destruction\_0ii, .-\_Z41\_\_static\_initialization\_and\_destruction\_0ii

.type \_GLOBAL\_\_sub\_I\_main, @function

\_GLOBAL\_\_sub\_I\_main:

.LFB1980:

.cfi\_startproc

pushq %rbp

.cfi\_def\_cfa\_offset 16

.cfi\_offset 6, -16

movq %rsp, %rbp

.cfi\_def\_cfa\_register 6

movl $65535, %esi

movl $1, %edi

call \_Z41\_\_static\_initialization\_and\_destruction\_0ii

popq %rbp

.cfi\_def\_cfa 7, 8

ret

.cfi\_endproc

.LFE1980:

.size \_GLOBAL\_\_sub\_I\_main, .-\_GLOBAL\_\_sub\_I\_main

.section .init\_array,"aw"

.align 8

.quad \_GLOBAL\_\_sub\_I\_main

.hidden \_\_dso\_handle

.ident "GCC: (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0"

.section .note.GNU-stack,"",@progbits

**Step 3:** Align each line of C++ code with the corresponding blocks of assembly code.

## C++ to Assembly Alignment

| **C++ Line of Code** | **Blocks of Assembly Code** |
| --- | --- |
| #include<iostream>  using namespace std; | .text  .globl main  .type main, @function |
| int main()  { | main:  .LFB1493:  .cfi\_startproc  pushq %rbp  .cfi\_def\_cfa\_offset 16  .cfi\_offset 6, -16  movq %rsp, %rbp  .cfi\_def\_cfa\_register 6  subq $16, %rsp |
| int width=10; | movl $10, -12(%rbp) |
| int height=5; | movl $5, -8(%rbp) |
| int area;  area = width \* height; | movl -12(%rbp), %eax  imull -8(%rbp), %eax  movl %eax, -4(%rbp) |
| cout<<endl<< area; | call cout  movl $0, %eax  leave |
| return 0; | leave  ret  .cfi\_endproc  .LFE1980:  .size \_GLOBAL\_\_sub\_I\_main, .-\_GLOBAL\_\_sub\_I\_main  .section .init\_array,"aw"  .align 8  .quad \_GLOBAL\_\_sub\_I\_main  .hidden \_\_dso\_handle  .ident "GCC: (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0"  .section .note.GNU-stack,"",@progbits |
| } |  |

**Step 4:** Explain how the blocks of assembly code perform the same tasks as the C++ code.

## Assembly Functionality

| **Blocks of Assembly Code** | **Explanation of Functionality** |
| --- | --- |
| .text  .globl main  .type main, @function | * Starts new section. The “text section is the section in object files that stores executable code. * The .globl main” tells the assembler to add the label that follows it to the list of labels exported by the generated object file. * “.type main,@function” instructs the assembler that the label “main” refers to executable code. |
| main:  .LFB1493:  .cfi\_startproc  pushq %rbp  .cfi\_def\_cfa\_offset 16  .cfi\_offset 6, -16  movq %rsp, %rbp  .cfi\_def\_cfa\_register 6  subq $16, %rsp | * “main:” defines the entry point to “main” function. * “.LFB1493” is a local label that refers to start of the function. * “.cfi\_startproc” is the call frame information directive. Instructs the assembler to emit dwarf format debugging information. * “pushq %rbp” is the standard part of a function prologue in assembly code and it’s saving the current value of the “rbp” register. * “.cfi\_def\_cfa\_offset 16” and “.cfi\_offset 6, -16” is related to the Dwarf debugging informatiionn. * “movq %rsp %rbp” means to set rbp to rsp. * “.cfi\_def\_cfa\_register 6” and “subq $16, %rsp” aligns the stack pointer and then subtracts enough room from it to hold all the locals for the function. |
| movl $10, -12(%rbp) | * Value 10 was moved 12 bytes above our register %rbp |
| movl $5, -8(%rbp) | * Value 5 was moved 8 bytes above our register %rbp |
| movl -12(%rbp), %eax  imull -8(%rbp), %eax  movl %eax, -4(%rbp) | * Moved 12 bytes in the register to eax. * 8 bytes in the register to “imull” instruction performed to multiply rbp and eax then place result inn rbq (-4). |
| call cout  movl $0, %eax  leave | * “call” calls cout * “movl $0,%eax sets the eax to 0 |
| leave  ret | * “leave” and “ret” is equivalent to “return 0.” * .cfi\_endproc * “.LFE1980:   .size \_GLOBAL\_\_sub\_I\_main, .-\_GLOBAL\_\_sub\_I\_main  .section .init\_array,"aw"  .align 8  .quad \_GLOBAL\_\_sub\_I\_main  .hidden \_\_dso\_handle  .ident "GCC: (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0"  .section .note.GNU-stack,"",@progbits” is the actual return instruction, it returns from the function. |