# **CS 410 Binary to C++ With Security Vulnerabilities Activity Template**

**Step 1:** Convert the binary file to assembly code.  
acAssignment6.s (attached)

**Step 2:** Explain the functionality of the blocks of assembly code.

| **Blocks of Assembly Code** | **Explanation of Functionality** |
| --- | --- |
| push %rbp | Push the current base pointer (%rbp) onto the stack, preserving its value. |
| mov %rsp,%rbp | Set up a new stack frame by moving the current stack pointer (%rsp) to the base pointer. |
| lea 0x0(%rip),%rsi # b | Load address specified by RIP relative offset into %rsi. Likely preparing an argument for a function call. |
| lea 0x0(%rip),%rdi # 12 | Load another address specified by RIP relative offset into %rdi. Likely preparing another argument for a function call. |
| callq 17 | Call the function at the offset <\_Z11DisplayMenuv+0x17>. |
| lea 0x0(%rip),%rsi # 1e | Load another address into %rsi. |
| lea 0x0(%rip),%rdi # 25 | Load another address into %rdi. |
| callq 2a | Call the function at the offset <\_Z11DisplayMenuv+0x2a>. |
| lea 0x0(%rip),%rsi # 31 | Load another address into %rsi. |
| lea 0x0(%rip),%rdi # 38 | Load another address into %rdi. |
| callq 3d | Call the function at the offset <\_Z11DisplayMenuv+0x3d>. |
| lea 0x0(%rip),%rsi # 44 | Load another address into %rsi. |
| lea 0x0(%rip),%rdi # 4b | Load another address into %rdi. |
| callq 50 | Call the function at the offset <\_Z11DisplayMenuv+0x50>. |
| lea 0x0(%rip),%rsi # 57 | Load another address into %rsi. |
| lea 0x0(%rip),%rdi # 5e | Load another address into %rdi. |
| callq 63 | Call the function at the offset <\_Z11DisplayMenuv+0x63>. |
| lea 0x0(%rip),%rsi # 6a | Load another address into %rsi. |
| lea 0x0(%rip),%rdi # 71 | Load another address into %rdi. |
| callq 76 | Call the function at the offset <\_Z11DisplayMenuv+0x76>. |
| nop | No operation. Often used for padding or for ensuring alignment. |
| pop %rbp | Pop the previously saved base pointer (%rbp) off the stack, restoring its original value. |
| retq | Return from the function, popping the return address off the stack and jumping to that address. |
| push %rbp | Push the current base pointer (%rbp) onto the stack, preserving its value. |
| mov %rsp,%rbp | Set up a new stack frame by moving the current stack pointer (%rsp) to the base pointer. |
| sub $0x20,%rsp | Allocate 32 bytes on the stack for local variables. |
| mov %fs:0x28,%rax | Load a value from the segment %fs at offset 0x28 into %rax. This is typically used for stack canaries to protect against buffer overflows. |
| mov %rax,-0x8(%rbp) | Store the value from %rax (the stack canary) at offset -0x8 from %rbp. |
| xor %eax,%eax | Zero out the %eax register. |
| movl $0x0,-0x14(%rbp) | Initialize a local variable at offset -0x14 from %rbp to 0. |
| mov -0x14(%rbp),%eax | Move the value of the local variable at offset -0x14 into %eax. |
| cmp $0x5,%eax | Compare the value in %eax to 5. |
| je 308 <main+0x28f> | Jump to the address at offset 308 if the previous comparison is equal. |
| lea 0x0(%rip),%rsi # aa <main+0x31> | Load an address specified by RIP relative offset into %rsi. Likely preparing an argument for a function call. |
| lea 0x0(%rip),%rdi # b1 <main+0x38> | Load another address specified by RIP relative offset into %rdi. Likely preparing another argument for a function call. |
| callq b6 | Call the function at the offset <main+0x3d>. |
| lea 0x0(%rip),%rsi # bd <main+0x44> | Load an address specified by RIP relative offset into %rsi. |
| lea 0x0(%rip),%rdi # c4 <main+0x4b> | Load another address specified by RIP relative offset into %rdi. |
| callq c9 <main+0x50> | Call the function at the offset <main+0x50>. |
| lea 0x0(%rip),%rsi # d0 <main+0x57> | Load an address specified by RIP relative offset into %rsi. |
| lea 0x0(%rip),%rdi # d7 <main+0x5e> | Load another address specified by RIP relative offset into %rdi. |
| callq dc <main+0x63> | Call the function at the offset <main+0x63>. |
| lea 0x0(%rip),%rsi # e3 <main+0x6a> | Load an address specified by RIP relative offset into %rsi. |
| lea 0x0(%rip),%rdi # ea <main+0x71> | Load another address specified by RIP relative offset into %rdi. |
| callq ef <main+0x76> | Call the function at the offset <main+0x76>. |
| lea 0x0(%rip),%rsi # f6 <main+0x7d> | Load an address specified by RIP relative offset into %rsi. |
| lea 0x0(%rip),%rdi # fd <main+0x84> | Load another address specified by RIP relative offset into %rdi. |
| callq 102 <main+0x89> | Call the function at the offset <main+0x89>. |
| lea 0x0(%rip),%rsi # 109 <main+0x90> | Load an address specified by RIP relative offset into %rsi. |
| lea 0x0(%rip),%rdi # 110 <main+0x97> | Load another address specified by RIP relative offset into %rdi. |
| callq 115 <main+0x9c> | Call the function at the offset <main+0x9c>. |
| lea -0x14(%rbp),%rax | Load the address of the local variable at offset -0x14 from %rbp into %rax. |
| mov %rax,%rsi | Move the address in %rax into %rsi. Likely preparing an argument for a function call. |
| lea 0x0(%rip),%rdi # 123 <main+0xaa> | Load an address specified by RIP relative offset into %rdi. |
| callq 128 <main+0xaf> | Call the function at the offset <main+0xaf>. |
| mov -0x14(%rbp),%eax | Move the value of the local variable at offset -0x14 into %eax. |
| cmp $0x1,%eax | Compare the value in %eax to 1. |
| jne 1c9 <main+0x150> | Jump to the address at offset 1c9 if the previous comparison is not equal. |
| lea -0x10(%rbp),%rax | Load the address of the local variable at offset -0x10 from %rbp into %rax. |
| mov %rax,%rsi | Move the address in %rax into %rsi. |
| lea 0x0(%rip),%rdi # 142 <main+0xc9> | Load an address specified by RIP relative offset into %rdi. |
| callq 147 <main+0xce> | Call the function at the offset <main+0xce>. |
| mov %rax,%rdx | Move the value in %rax into %rdx. |
| lea -0xc(%rbp),%rax | Load the address of the local variable at offset -0xc from %rbp into %rax. |
| mov %rax,%rsi | Move the address in %rax into %rsi. |
| mov %rdx,%rdi | Move the value in %rdx into %rdi. |
| callq 159 <main+0xe0> | Call the function at the offset <main+0xe0>. |
| mov -0x10(%rbp),%eax | Move the value of the local variable at offset -0x10 into %eax. |
| mov %eax,%esi | Move the value in %eax into %esi. |
| lea 0x0(%rip),%rdi # 165 <main+0xec> | Load an address specified by RIP relative offset into %rdi. |
| callq 16a <main+0xf1> | Call the function at the offset <main+0xf1>. |
| lea 0x0(%rip),%rsi # 171 <main+0xf8> | Load an address specified by RIP relative offset into %rsi. |
| mov %rax,%rdi | Move the value in %rax into %rdi. |
| callq 179 <main+0x100> | Call the function at the offset <main+0x100>. |
| mov %rax,%rdx | Move the value in %rax into %rdx. |
| mov -0xc(%rbp),%eax | Move the value of the local variable at offset -0xc into %eax. |
| mov %eax,%esi | Move the value in %eax into %esi. |
| mov %rdx,%rdi | Move the value in %rdx into %rdi. |
| callq 189 <main+0x110> | Call the function at the offset <main+0x110>. |
| lea 0x0(%rip),%rsi # 190 <main+0x117> | Load an address specified by RIP relative offset into %rsi. |
| mov %rax,%rdi | Move the value in %rax into %rdi. |
| callq 198 <main+0x11f> | Call the function at the offset <main+0x11f>. |
| mov %rax,%rcx | Move the value in %rax into %rcx. |
| mov -0x10(%rbp),%edx | Move the value of the local variable at offset -0x10 into %edx. |
| mov -0xc(%rbp),%eax | Move the value of the local variable at offset -0xc into %eax. |
| sub %eax,%edx | Subtract the value in %eax from %edx and store the result in %edx. |
| mov %edx,%eax | Move the value in %edx into %eax. |
| mov %eax,%esi | Move the value in %eax into %esi. |
| mov %rcx,%rdi | Move the value in %rcx into %rdi. |
| callq 1af <main+0x136> | Call the function at the offset <main+0x136>. |
| mov %rax,%rdx | Move the value in %rax into %rdx. |
| mov 0x0(%rip),%rax # 1b9 <main+0x140> | Load an address specified by RIP relative offset into %rax. |
| mov %rax,%rsi | Move the address in %rax into %rsi. |
| mov %rdx,%rdi | Move the value in %rdx into %rdi. |
| callq 1c4 <main+0x14b> | Call the function at the offset <main+0x14b>. |
| jmpq 97 <main+0x1e> | Jump to the address at offset 97. |
| mov -0x14(%rbp),%eax | Move the value of the local variable at offset -0x14 into %eax. |
| cmp $0x2,%eax | Compare the value in %eax to 2. |
| jne 268 <main+0x1ef> | Jump to the address at offset 268 if the previous comparison is not equal. |
| mov $0x0,%eax | Set the %eax register to 0. This is often used to specify a return value. |
| mov -0x8(%rbp),%rcx | Move the value from offset -0x8 of %rbp into %rcx. |
| xor %fs:0x28,%rcx | XOR the value from the segment %fs at offset 0x28 with %rcx. This checks if the stack canary has been tampered with. |
| je 321 <main+0x2a8> | If the XOR operation resulted in zero, jump to offset 321. |
| callq 321 <main+0x2a8> | Call the function at the specified offset if the canary was modified (likely an error handling routine). |
| leaveq | Clean up the current stack frame. This is equivalent to mov %rbp, %rsp followed by pop %rbp. |
| retq | Return from the function, popping the return address off the stack and jumping to that address. |
| push %rbp | Push the current base pointer (%rbp) onto the stack, preserving its value. |
| mov %rsp,%rbp | Set up a new stack frame by moving the current stack pointer (%rsp) to the base pointer. |
| sub $0x10,%rsp | Allocate 16 bytes on the stack for local variables. |
| mov %edi,-0x4(%rbp) | Move the value from the %edi register (typically the 1st function argument) to the local variable at offset -0x4 from %rbp. |
| mov %esi,-0x8(%rbp) | Move the value from the %esi register (typically the 2nd function argument) to the local variable at offset -0x8 from %rbp. |
| cmpl $0x1,-0x4(%rbp) | Compare the value of the local variable at offset -0x4 from %rbp to 1. |
| jne 369 <...+0x46> | Jump to the specified offset if the previous comparison is not equal. |
| cmpl $0xffff,-0x8(%rbp) | Compare the value of the local variable at offset -0x8 from %rbp to 0xffff. |
| jne 369 <...+0x46> | Jump to the specified offset if the previous comparison is not equal. |
| lea 0x0(%rip),%rdi # 347 <...+0x24> | Load an address specified by RIP relative offset into %rdi. Likely preparing an argument for a function call. |
| callq 34c <...+0x29> | Call the function at the specified offset. |
| lea 0x0(%rip),%rdx # 353 <...+0x30> | Load an address specified by RIP relative offset into %rdx. Likely preparing an argument for a function call. |
| lea 0x0(%rip),%rsi # 35a <...+0x37> | Load another address specified by RIP relative offset into %rsi. |
| mov 0x0(%rip),%rax # 361 <...+0x3e> | Load an address specified by RIP relative offset into %rax. |
| mov %rax,%rdi | Move the address in %rax into %rdi. |
| callq 369 <...+0x46> | Call the function at the specified offset. |
| nop | No operation. This instruction does nothing and often serves as padding or for alignment purposes. |
| leaveq | Clean up the current stack frame. This is equivalent to mov %rbp, %rsp followed by pop %rbp. |
| retq | Return from the function, popping the return address off the stack and jumping to that address. |

**Step 3:** Convert the assembly code to binary.

assignment6.bin (attached)

**Step 4:** Convert the assembly code to C++ code.

| **Blocks of Assembly Code** | **C++ Code** |
| --- | --- |
| push %rbp  mov %rsp,%rbp  callq 3e1 <main+0x9>  mov $0x0,%eax  pop %rbp  retq | int main()  {  displayMenu();  } |
| push %rbp  mov %rsp,%rbp  sub $0x20,%rsp  mov %fs:0x28,%rax  mov %rax,-0x8(%rbp)  xor %eax,%eax  movl $0x1,-0x10(%rbp)  lea 0x0(%rip),%rsi # 25 <\_Z11displayMenuv+0x25>  lea 0x0(%rip),%rdi # 2c <\_Z11displayMenuv+0x2c>  callq 31 <\_Z11displayMenuv+0x31>  mov %rax,%rdx  mov 0x0(%rip),%rax # 3b <\_Z11displayMenuv+0x3b>  mov %rax,%rsi  mov %rdx,%rdi  callq 46 <\_Z11displayMenuv+0x46>  lea 0x0(%rip),%rsi # 4d <\_Z11displayMenuv+0x4d>  lea 0x0(%rip),%rdi # 54 <\_Z11displayMenuv+0x54>  callq 59 <\_Z11displayMenuv+0x59>  mov %rax,%rdx  mov 0x0(%rip),%rax # 63 <\_Z11displayMenuv+0x63>  mov %rax,%rsi  mov %rdx,%rdi  callq 6e <\_Z11displayMenuv+0x6e>  lea 0x0(%rip),%rsi # 75 <\_Z11displayMenuv+0x75>  lea 0x0(%rip),%rdi # 7c <\_Z11displayMenuv+0x7c>  callq 81 <\_Z11displayMenuv+0x81> mov %rax,%rdx  mov 0x0(%rip),%rax # 8b <\_Z11displayMenuv+0x8b>  mov %rax,%rsi mov %rdx,%rdi  callq 96 <\_Z11displayMenuv+0x96>  lea 0x0(%rip),%rsi # 9d <\_Z11displayMenuv+0x9d>  lea 0x0(%rip),%rdi # a4 <\_Z11displayMenuv+0xa4>  callq a9 <\_Z11displayMenuv+0xa9> mov %rax,%rdx  mov 0x0(%rip),%rax # b3 <\_Z11displayMenuv+0xb3>  mov %rax,%rsi  mov %rdx,%rdi | void displayMenu()  {  int x, y, selection, solution;  int displayLoop = 1;  cout << "----------------" << endl;  cout << "- 1)Add -" << endl;  cout << "- 2)Subtract -" << endl;  cout << "- 3)Multiply -" << endl;  cout << "- 4)Exit -" << endl;  cout << "----------------" << endl;  cin >> selection; |
| test %al,%al je 193 <\_Z11displayMenuv+0x193>  mov $0x0,%esi  lea 0x0(%rip),%rdi # 13d <\_Z11displayMenuv+0x13d>  callq 142 <\_Z11displayMenuv+0x142>  mov $0xa,%edx  mov $0x7fffffff,%esi  lea 0x0(%rip),%rdi # 153 <\_Z11displayMenuv+0x153>  callq 158 <\_Z11displayMenuv+0x158>  lea 0x0(%rip),%rsi # 15f <\_Z11displayMenuv+0x15f>  lea 0x0(%rip),%rdi # 166 <\_Z11displayMenuv+0x166>  callq 16b <\_Z11displayMenuv+0x16b>  lea 0x0(%rip),%rsi # 172 <\_Z11displayMenuv+0x172>  lea 0x0(%rip),%rdi # 179 <\_Z11displayMenuv+0x179>  callq 17e <\_Z11displayMenuv+0x17e>  lea -0x14(%rbp),%rax  mov %rax,%rsi  lea 0x0(%rip),%rdi # 18c <\_Z11displayMenuv+0x18c>  callq 191 <\_Z11displayMenuv+0x191> | while (cin.fail())  {  cin.clear(); // clear input buffer to restore cin to a usable state cin.ignore(INT\_MAX, '\n'); // ignore last input  cout << "You can only enter valid numbers.\n";  cout << "Enter a number.\n"; cin >> selection;  } |
| jmp 121 <\_Z11displayMenuv+0x121>  cmpl $0x0,-0x10(%rbp) je 3c1 <\_Z11displayMenuv+0x3c1>  mov -0x14(%rbp),%eax  cmp $0x1,%eax jne 24e <\_Z11displayMenuv+0x24e>  lea -0x1c(%rbp),%rax  mov %rax,%rsi  lea 0x0(%rip),%rdi # 1b7 <\_Z11displayMenuv+0x1b7>  callq 1bc <\_Z11displayMenuv+0x1bc>  mov %rax,%rdx  lea -0x18(%rbp),%rax  mov %rax,%rsi  mov %rdx,%rdi  callq 1ce <\_Z11displayMenuv+0x1ce>  mov -0x1c(%rbp),%edx  mov -0x18(%rbp),%eax  add %edx,%eax  mov %eax,-0xc(%rbp)  mov -0x1c(%rbp),%eax  mov %eax,%esi  lea 0x0(%rip),%rdi # 1e5 <\_Z11displayMenuv+0x1e5>  callq  1ea <\_Z11displayMenuv+0x1ea>  lea 0x0(%rip),%rsi # 1f1 <\_Z11displayMenuv+0x1f1>  mov %rax,%rdi  callq 1f9 <\_Z11displayMenuv+0x1f9>  mov %rax,%rdx mov -0x18(%rbp),%eax  mov %eax,%esi  mov %rdx,%rdi  callq 209 <\_Z11displayMenuv+0x209>  lea 0x0(%rip),%rsi # 210 <\_Z11displayMenuv+0x210>  mov %rax,%rdi callq 218 <\_Z11displayMenuv+0x218>  mov %rax,%rdx  mov -0xc(%rbp),%eax  mov %eax,%esi  mov %rdx,%rdi  callq 228 <\_Z11displayMenuv+0x228>  mov %rax,%rdx  mov 0x0(%rip),%rax # 232 <\_Z11displayMenuv+0x232>  mov %rax,%rsi  mov %rdx,%rdi  callq 23d <\_Z11displayMenuv+0x23d>  movl $0x0,-0x10(%rbp) callq 249 <\_Z11displayMenuv+0x249> |  |
|  |  |