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

**Step 1:** Convert the binary file to assembly code.

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

**DisplayMenu()**

| **Blocks of Assembly Code** | **Explanation of Functionality** |
| --- | --- |
| push %rbp | Save the base pointer onto the stack |
| mov %rsp,%rbp | Set up new stack frame for the function |
| lea 0xb(%rip),%rsi | Load address of first string (menu line 1) into rsi |
| lea 0x12(%rip),%rdi | Load address of std::cout (or similar) into rdi |
| callq 0x17 | Call print function |
| lea 0x1e(%rip),%rsi | Load string for menu line 2 |
| lea 0x25(%rip),%rdi | Load std::cout again |
| callq 0x2a | Call print function again |
| lea 0x31(%rip),%rsi | Load string for menu line 3 |
| lea 0x38(%rip),%rdi | Load std::cout again |
| callq 0x3d | Call print function again |
| lea 0x44(%rip),%rsi | Load string for menu line 4 |
| lea 0x4b(%rip),%rdi | Load std::cout again |
| callq 0x50 | Call print function again |
| lea 0x57(%rip),%rsi | Load string for menu line 5 |
| lea 0x5e(%rip),%rdi | Load std::cout again |
| callq 0x63 | Call print function again |
| lea 0x6a(%rip),%rsi | Load string for input prompt |
| lea 0x71(%rip),%rdi | Load std::cout again |
| callq 0x76 | Print the final prompt line |
| nop | No operation (used for alignment/padding) |
| pop %rbp | Restore the old base pointer |
| retq | Return from DisplayMenu() |

**Main()**

|  |  |
| --- | --- |
| **Blocks of Assembly Code** | **Explanation of Functionality** |
| push %rbp | Save the base pointer to start the stack frame |
| mov %rsp,%rbp | Set new base pointer for this function |
| sub $0x20,%rsp | Reserve 32 bytes of space on the stack for local variables |
| mov %fs:0x28,%rax | Get the stack canary value for protection against buffer overflows |
| mov %rax,-0x8(%rbp) | Save the canary to stack (used for security check later) |
| xor %eax,%eax | Set eax = 0 |
| movl $0x0,-0x14(%rbp) | Set variable choice = 0 |
| mov -0x14(%rbp),%eax | Load choice into eax |
| cmp $0x5,%eax | Compare choice with 5 |
| je 308 | If choice == 5, jump to end of program and exit |
| lea 0xaa(%rip),%rsi | Load prompt string into rsi |
| lea 0xb1(%rip),%rdi | Load cout into rdi |
| callq b6 | Print the string |
| lea 0xbd(%rip),%rsi | Load next string |
| lea 0xc4(%rip),%rdi | Load cout again |
| callq c9 | Print the string |
| lea 0xd0(%rip),%rsi | Load next string |
| lea 0xd7(%rip),%rdi | Load cout again |
| callq dc | Print the string |
| lea 0xe3(%rip),%rsi | Load next string |
| lea 0xea(%rip),%rdi | Load cout again |
| callq ef | Print the string |
| lea 0xf6(%rip),%rsi | Load next string |
| lea 0xfd(%rip),%rdi | Load cout again |
| callq 102 | Print the string |
| lea 0x109(%rip),%rsi | Load final menu line |
| lea 0x110(%rip),%rdi | Load cout again |
| callq 115 | Print the final prompt |
| lea -0x14(%rbp),%rax | Load address of choice variable |
| mov %rax,%rsi | Move that address into rsi (for cin >> choice) |
| lea 0x123(%rip),%rdi | Load cin into rdi |
| callq 128 | Call function to take user input |
| mov -0x14(%rbp),%eax | Load choice back into eax |
| cmp $0x1,%eax | Check if choice == 1 |
| jne 1c9 | If not equal to 1, jump to next section (likely add or divide) |
| lea -0x10(%rbp),%rax | Load address of first input variable (e.g., a) |
| mov %rax,%rsi | Move address into rsi for input |
| lea 0x142(%rip),%rdi | Load cin stream into rdi |
| callq 147 | Call input function (cin >> a) |
| mov %rax,%rdx | Save result of cin into rdx |
| lea -0xc(%rbp),%rax | Load address of second input variable (b) |
| mov %rax,%rsi | Move address into rsi |
| mov %rdx,%rdi | Move cin stream result into rdi |
| callq 159 | Call input function (cin >> b) |
| mov -0x10(%rbp),%eax | Load value of a into eax |
| mov %eax,%esi | Move a into esi for output |
| lea 0x165(%rip),%rdi | Load cout into rdi |
| callq 16a | Print value of a |
| lea 0x171(%rip),%rsi | Load string " - " into rsi |
| mov %rax,%rdi | Move stream into rdi for next print |
| callq 179 | Print " - " |
| mov %rax,%rdx | Save stream into rdx |
| mov -0xc(%rbp),%eax | Load value of b into eax |
| mov %eax,%esi | Move b into esi |
| mov %rdx,%rdi | Move stream into rdi |
| callq 189 | Print value of b |
| lea 0x190(%rip),%rsi | Load string " = " into rsi |
| mov %rax,%rdi | Move stream into rdi |
| callq 198 | Print " = " |
| mov %rax,%rcx | Save stream into rcx |
| mov -0x10(%rbp),%edx | Load a into edx |
| mov -0xc(%rbp),%eax | Load b into eax |
| sub %eax,%edx | Subtract b from a |
| mov %edx,%eax | Move result into eax |
| mov %eax,%esi | Move result into esi for output |
| mov %rcx,%rdi | Move stream into rdi |
| callq 1af | Print result of subtraction |
| mov %rax,%rdx | Save stream into rdx |
| mov 0x0(%rip),%rax | Load address of newline string or endl |
| mov %rax,%rsi | Move newline string into rsi |
| mov %rdx,%rdi | Move stream into rdi |
| callq 1c4 | Print newline |
| jmpq 97 | Jump back to menu loop start |
| mov -0x14(%rbp),%eax | Load choice into eax |
| cmp $0x2,%eax | Compare choice to 2 |
| jne 268 | If not 2, jump to next condition |
| lea -0x10(%rbp),%rax | Load address of variable a |
| mov %rax,%rsi | Move a’s address into rsi |
| lea 0x1e3(%rip),%rdi | Load cin into rdi |
| callq 1e8 | Take input for a |
| mov %rax,%rdx | Save stream to rdx |
| lea -0xc(%rbp),%rax | Load address of variable b |
| mov %rax,%rsi | Move b’s address into rsi |
| mov %rdx,%rdi | Move stream into rdi |
| callq 1fa | Take input for b |
| mov -0x10(%rbp),%eax | Load value of a |
| mov %eax,%esi | Move a into esi for printing |
| lea 0x206(%rip),%rdi | Load cout into rdi |
| callq 20b | Print value of a |
| lea 0x212(%rip),%rsi | Load string " + " into rsi |
| mov %rax,%rdi | Move stream into rdi |
| callq 21a | Print " + " |
| mov %rax,%rdx | Save stream to rdx |
| mov -0xc(%rbp),%eax | Load value of b |
| mov %eax,%esi | Move b into esi |
| mov %rdx,%rdi | Move stream into rdi |
| callq 22a | Print value of b |
| lea 0x231(%rip),%rsi | Load string " = " into rsi |
| mov %rax,%rdi | Move stream into rdi |
| callq 239 | Print " = " |
| mov %rax,%rcx | Save stream into rcx |
| mov -0x10(%rbp),%edx | Load value of a |
| mov -0xc(%rbp),%eax | Load value of b |
| add %edx,%eax | Add a + b |
| mov %eax,%esi | Move result into esi |
| mov %rcx,%rdi | Move stream into rdi |
| callq 24e | Print result |
| mov %rax,%rdx | Save stream to rdx |
| mov 0x0(%rip),%rax | Load address of newline or endl |
| mov %rax,%rsi | Move newline string into rsi |
| mov %rdx,%rdi | Move stream into rdi |
| callq 263 | Print newline |
| jmpq 97 | Jump back to menu loop start |
| Instruction | Explanation |
| mov -0x14(%rbp),%eax | Load choice |
| cmp $0x3,%eax | Compare to 3 |
| jne 97 | Jump to top of loop if not equal |
| lea -0x10(%rbp),%rax | Load address of variable a |
| mov %rax,%rsi | Move a address into rsi |
| lea 0x282(%rip),%rdi | Load cin into rdi |
| callq 287 | Take input for a |
| mov %rax,%rdx | Save stream to rdx |
| lea -0xc(%rbp),%rax | Load address of variable b |
| mov %rax,%rsi | Move address into rsi |
| mov %rdx,%rdi | Move stream into rdi |
| callq 299 | Take input for b |
| mov -0x10(%rbp),%eax | Load value of a |
| mov %eax,%esi | Move into esi |
| lea 0x2a5(%rip),%rdi | Load cout into rdi |
| callq 2aa | Print value of a |
| lea 0x2b1(%rip),%rsi | Load string " / " |
| mov %rax,%rdi | Move stream into rdi |
| callq 2b9 | Print " / " |
| mov %rax,%rdx | Save stream |
| mov -0xc(%rbp),%eax | Load value of b |
| mov %eax,%esi | Move into esi |
| mov %rdx,%rdi | Move stream into rdi |
| callq 2c9 | Print value of b |
| lea 0x2d0(%rip),%rsi | Load string " = " |
| mov %rax,%rdi | Move stream into rdi |
| callq 2d8 | Print " = " |
| mov %rax,%rcx | Save stream to rcx |
| mov -0x10(%rbp),%eax | Load a |
| mov -0xc(%rbp),%esi | Load b |
| cltd | Sign-extend eax into edx |
| idiv %esi | Divide a / b |
| mov %eax,%esi | Move result into esi |
| mov %rcx,%rdi | Move stream into rdi |
| callq 2ee | Print result |
| mov %rax,%rdx | Save stream |
| mov 0x0(%rip),%rax | Load newline |
| mov %rax,%rsi | Move newline into rsi |
| mov %rdx,%rdi | Move stream |
| callq 303 | Print newline |
| jmpq 97 | Jump back to start of loop |
| mov $0x0,%eax | Set return value to 0 (exit success) | |
| mov -0x8(%rbp),%rcx | Load original stack canary | |
| xor %fs:0x28,%rcx | Compare canary with expected value | |
| je 321 | If equal, jump to function exit | |
| callq 321 | If not equal, call fail routine | |
| leaveq | Clean up stack frame | |
| retq | Return from main() | |

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

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

**DisplayMenu()**

| **Blocks of Assembly Code** | **C++ Code** |
| --- | --- |
| push %rbp mov %rsp, %rbp | void DisplayMenu() { |
| lea 0x463(%rip), %rsi lea 0x200988(%rip), %rdi call puts@plt mov %rax, %rdx mov 0x200a1c(%rip), %rax mov %rax, %rsi mov %rdx, %rdi callq | std::cout << "\n\n--- Calculator Menu ---" << std::endl; |
| lea 0x464(%rip), %rsi lea 0x200960(%rip), %rdi call puts@plt mov %rax, %rdx mov 0x2009f5(%rip), %rax mov %rax, %rsi mov %rdx, %rdi callq | std::cout << "1. Subtract" << std::endl; |
| lea 0x462(%rip), %rsi lea 0x20093c(%rip), %rdi call puts@plt mov %rax, %rdx mov 0x2009d5(%rip), %rax mov %rax, %rsi mov %rdx, %rdi callq | std::cout << "2. Add" << std::endl; |
| lea 0x463(%rip), %rsi lea 0x20091a(%rip), %rdi call puts@plt mov %rax, %rdx mov 0x2009b5(%rip), %rax mov %rax, %rsi mov %rdx, %rdi callq | std::cout << "3. Divide" << std::endl; |
| lea 0x462(%rip), %rsi lea 0x2008fa(%rip), %rdi call puts@plt mov %rax, %rdx mov 0x200995(%rip), %rax mov %rax, %rsi mov %rdx, %rdi callq | std::cout << "4. Multiply" << std::endl; |
| lea 0x45d(%rip), %rsi lea 0x2008da(%rip), %rdi call puts@plt mov %rax, %rdx mov 0x200974(%rip), %rax mov %rax, %rsi mov %rdx, %rdi callq | std::cout << "5. Exit" << std::endl; |
| lea 0x469(%rip), %rsi lea 0x2008bc(%rip), %rdi call puts@plt mov %rax, %rdx mov 0x200954(%rip), %rax mov %rax, %rsi mov %rdx, %rdi call operator<< | std::cout << "-----------------------" << std::endl; |
| lea 0x449(%rip), %rsi lea 0x20089c(%rip), %rdi call puts@plt mov %rax, %rdx mov 0x200934(%rip), %rax mov %rax, %rsi mov %rdx, %rdi call operator<< | std::cout << "Enter your choice: " << std::endl; |
| pop %rbp retq | } |

**Main**

| **Blocks of Assembly Code** | **C++ Code** |
| --- | --- |
| push %rbp mov %rsp, %rbp | int main() { |
| sub $0x10, %rsp | Stack space allocation — implicit in C++ |
| movl $0, -0x4(%rbp) | int choice = 0; |
| movl $0, -0x8(%rbp) movl $0, -0xc(%rbp) | int a = 0, b = 0; |
| jmp 0xbe0 | while (choice != 5) { *(loop jump to condition)* |
| callq 0xb3a <DisplayMenu> | DisplayMenu(); |
| mov -0x4(%rbp), %eax lea -0x4(%rbp), %rsi mov $0x2014d0, %rdi callq 0x00000000000009b0 | cin >> choice; |
| mov -0x4(%rbp), %eax cmp $0x1, %eax jne 0x0000000000000bfd | if (choice == 1) |
| mov $0x20102c, %edi callq 0x0000000000000970 | cout << "Enter two numbers: "; |
| lea -0x8(%rbp), %rsi mov $0x2014d0, %rdi callq 0x00000000000009b0 | cin >> a; |
| lea -0xc(%rbp), %rsi mov $0x2014d0, %rdi callq 0x00000000000009b0 | cin >> b; |
| mov -0x8(%rbp), %edx mov -0xc(%rbp), %eax sub %eax, %edx mov -0x8(%rbp), %esi mov -0xc(%rbp), %eax mov %eax, %ecx mov %edx, %eax mov $0x201038, %edi mov $0x0, %eax callq 0x0000000000000990 | cout << a << " - " << b << " = " << (a - b); |
| jmp 0x0000000000000cbd | continue; (Back to top of loop) |
| cmp $0x2, %eax jne 0x0000000000000c2e | else if (choice == 2) |
| Repeat input logic for two numbers | cin >> a >> b; |
| add %eax, %edx call printf@plt | cout << a << " + " << b << " = " << (a + b); |
| cmp $0x3, %eax jne 0x0000000000000c5f | else if (choice == 3) |
| Repeat input logic | cin >> a >> b; |
| div %eax, %ecx call printf@plt | cout << a << " / " << b << " = " << (a / b); |
| cmp $0x4, %eax jne 0x0000000000000c90 | else if (choice == 4) |
| Repeat input logic | cin >> a >> b; |
| imul %eax, %ecx call printf@plt | cout << a << " \* " << b << " = " << (a \* b); |
| cmp $0x5, %eax jne 0x0000000000000cb0 | else if (choice == 5) |
| mov $0x20106c, %edi call puts@plt | cout << "Exiting program."; |
| jmp 0x0000000000000cbd | continue; (Back to top of loop) |
| mov $0x20108c, %edi call puts@plt | cout << "Invalid choice. Please try again."; |
| jmp 0x0000000000000be0 | } (End of loop block) |
| mov $0x0, %eax | return 0; |
| leave ret | } (End of main) |