# **Anthony W Bradley**

08/10/2025

CS 410 Software Reverse Engineering

Professor Mark Babcock

# **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.

| **Blocks of Assembly Code** | **Explanation of Functionality** |
| --- | --- |
| 0000000000000000 <\_Z11DisplayMenuv> push %rbp mov %rsp,%rbp lea ..., %rsi lea ..., %rdi callq ... (repeated) pop %rbp retq | Displays the calculator menu. Each lea loads a string address, and each callq prints a line of the menu. |
| 0000000000000079 <main> push %rbp mov %rsp,%rbp sub $0x20,%rsp mov %fs:0x28,%rax mov %rax,-0x8(%rbp) xor %eax,%eax movl $0x0,-0x14(%rbp) | Sets up the stack frame and initializes local variables. -0x14(%rbp) stores the user's menu choice. |
| 97: mov -0x14(%rbp),%eax 9a: cmp $0x5,%eax 9d: je 308 | Checks if the user selected option 5 (exit). If so, jumps to the exit block. |
| a3–115: lea + callq sequences | Displays the menu and prompts the user for input. Each lea loads a string address, and callq prints it. |
| 115–128: lea -0x14(%rbp),%rax → callq | Reads the user's menu choice and stores it at -0x14(%rbp). |
| 128–1c4: subtraction block | If user selects 1, this block reads two integers, subtracts them, converts the result to a string, prints it, and loops back to the menu. |
| 1c9–263: addition block | If user selects 2, this block reads two integers, adds them, converts the result to a string, prints it, and loops back to the menu. |
| 268–303: division block | If user selects 3, this block reads two integers, performs signed division (cltd, idiv), converts the result to a string, prints it, and loops back. |
| 308–322: exit block | Cleans up and exits the program. Clears thread-local storage and returns from main. |
| 0000000000000323 <\_Z41\_\_static\_initialization\_and\_destruction\_0ii> push %rbp mov %rsp,%rbp sub $0x10,%rsp cmp $0x1,-0x4(%rbp) cmp $0xffff,-0x8(%rbp) callq ... leaveq retq | Handles static object initialization. Checks parameters and calls constructors if needed. |
| 000000000000036c <\_GLOBAL\_\_sub\_I\_\_Z11DisplayMenuv> mov $0xffff,%esi mov $0x1,%edi callq 323 retq | Invokes the static initialization function before main() runs. Auto-generated by the compiler for global/static objects. |

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

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

| **Blocks of Assembly Code** | **C++ Code** |
| --- | --- |
| nasm section .data menu db "1. Subtract", 10, "2. Add", 10, "3. Divide", 10, "5. Exit", 10 prompt db "Enter choice: ", 10 input\_a db "Enter A: ", 10 input\_b db "Enter B: ", 10 result\_msg db "Result: " divzero db "Error: Division by zero", 10 | cpp const std::string menu = "1. Subtract\n2. Add\n3. Divide\n5. Exit\n"; const std::string prompt = "Enter choice: "; const std::string input\_a = "Enter A: "; const std::string input\_b = "Enter B: "; const std::string result\_msg = "Result: "; const std::string divzero = "Error: Division by zero\n"; |
| nasm section .bss choice resb 2 a resd 1 b resd 1 result\_str resb 20 | cpp int choice = 0; int a = 0, b = 0; std::string result\_str; |
| nasm \_start: menu\_loop: mov rax, 1 mov rdi, 1 mov rsi, menu mov rdx, 41 syscall mov rax, 1 mov rdi, 1 mov rsi, prompt mov rdx, 15 syscall mov rax, 0 mov rdi, 0 mov rsi, choice mov rdx, 2 syscall movzx rax, byte [choice] sub rax, '0' cmp rax, 5 je exit\_program cmp rax, 1 je subtract cmp rax, 2 je add cmp rax, 3 je divide jmp menu\_loop | cpp while (true) { std::cout << menu; std::cout << prompt; std::cin >> choice; if (choice == 5) break; switch (choice) { case 1: subtract(); break; case 2: add(); break; case 3: divide(); break; default: continue; } } |
| nasm read\_inputs: mov rax, 1 mov rdi, 1 mov rsi, input\_a mov rdx, 11 syscall mov rax, 0 mov rdi, 0 mov rsi, result\_str mov rdx, 10 syscall call atoi mov [a], eax mov rax, 1 mov rdi, 1 mov rsi, input\_b mov rdx, 11 syscall mov rax, 0 mov rdi, 0 mov rsi, result\_str mov rdx, 10 syscall call atoi mov [b], eax ret | cpp void read\_inputs() { std::cout << input\_a; std::cin >> a; std::cout << input\_b; std::cin >> b; } |
| nasm subtract: call read\_inputs mov eax, [a] sub eax, [b] call itoa call print\_result jmp menu\_loop | cpp void subtract() { read\_inputs(); int result = a - b; print\_result(result); } |
| nasm add: call read\_inputs mov eax, [a] add eax, [b] call itoa call print\_result jmp menu\_loop | cpp void add() { read\_inputs(); int result = a + b; print\_result(result); } |
| nasm divide: call read\_inputs mov eax, [b] cmp eax, 0 je div\_error mov eax, [a] cdq idiv dword [b] call itoa call print\_result jmp menu\_loop | cpp void divide() { read\_inputs(); if (b == 0) { std::cout << divzero; return; } int result = a / b; print\_result(result); } |
| nasm print\_result: mov rax, 1 mov rdi, 1 mov rsi, result\_msg mov rdx, 8 syscall mov rax, 1 mov rdi, 1 mov rsi, result\_str mov rdx, 10 syscall mov rax, 1 mov rdi, 1 mov rsi, newline mov rdx, 1 syscall ret | cpp void print\_result(int result) { std::cout << result\_msg << result << std::endl; } |