# **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** |
| --- | --- |
| push %rbp  mov %rsp,%rbp  lea 0x0(%rip),%rsi # 0xb <\_Z11DisplayMenuv+11>  lea 0x0(%rip),%rdi # 0x12 <\_Z11DisplayMenuv+18>  callq 0x17 <\_Z11DisplayMenuv+23>  lea 0x0(%rip),%rsi # 0x1e <\_Z11DisplayMenuv+30>  lea 0x0(%rip),%rdi # 0x25 <\_Z11DisplayMenuv+37>  callq 0x2a <\_Z11DisplayMenuv+42>  lea 0x0(%rip),%rsi # 0x31 <\_Z11DisplayMenuv+49>  lea 0x0(%rip),%rdi # 0x38 <\_Z11DisplayMenuv+56>  callq 0x3d <\_Z11DisplayMenuv+61>  lea 0x0(%rip),%rsi # 0x44 <\_Z11DisplayMenuv+68>  lea 0x0(%rip),%rdi # 0x4b <\_Z11DisplayMenuv+75>  callq 0x50 <\_Z11DisplayMenuv+80>  lea 0x0(%rip),%rsi # 0x57 <\_Z11DisplayMenuv+87>  lea 0x0(%rip),%rdi # 0x5e <\_Z11DisplayMenuv+94>  callq 0x63 <\_Z11DisplayMenuv+99>  lea 0x0(%rip),%rsi # 0x6a <\_Z11DisplayMenuv+106>  lea 0x0(%rip),%rdi # 0x71 <\_Z11DisplayMenuv+113>  callq 0x76 <\_Z11DisplayMenuv+118>  pop %rbp  retq | %rbp pushed to current stack  %rbp moved in %rsp  %rsi loaded in 0(%rip)  %rdi loaded in 0(%rip)  %rsi loaded in 0(%rip)  %rdi loaded in 0(%rip)  %rsi loaded in 0(%rip)  %rdi loaded in 0(%rip)  %rsi loaded in 0(%rip)  %rdi loaded in 0(%rip)  %rsi loaded in 0(%rip)  %rdi loaded in 0(%rip)  %rsi loaded in 0(%rip)  %rdi loaded in 0(%rip)  4-byte data is removed in %rbp  Return  Multiple displaymenu are called |
| mov -0x14(%rbp),%eax  cmp $0x5,%eax  je 0x308 <main+655>  lea 0x0(%rip),%rsi # 0xaa <main+49>  lea 0x0(%rip),%rdi # 0xb1 <main+56>  callq 0xb6 <main+61> | %eax moved in -14(%rbp)  %eax is compared to 5  if equal then jump  %rsi loaded in 0(%rip)  %rdi loaded in 0(rip)  Call |
| lea -0x14(%rbp),%rax  mov %rax,%rsi  lea 0x0(%rip),%rdi # 0x123 <main+170>  callq 0x128 <main+175> | %rax loaded in 14(%rbp)  %rsi moved in %rax  %rdi loaded in 0(%rip)  Call |
| lea -0x14(%rbp),%rax  mov %rax,%rsi  lea 0x0(%rip),%rdi # 0x123 <main+170>  callq 0x128 <main+175>  mov -0x14(%rbp),%eax  cmp $0x1,%eax  jne 0x1c9 <main+336>  lea -0x10(%rbp),%rax  mov %rax,%rsi  lea 0x0(%rip),%rdi # 0x142 <main+201>  callq 0x147 <main+206>  mov %rax,%rcx  mov -0x10(%rbp),%edx  mov -0xc(%rbp),%eax  sub %eax,%edx  mov %edx,%eax  mov %eax,%esi  mov %rcx,%rdi  callq 0x1af <main+310>  mov -0x14(%rbp),%eax  cmp $0x2,%eax  jne 0x268 <main+495>  lea -0x10(%rbp),%rax  mov %rax,%rsi  lea 0x0(%rip),%rdi # 0x1e3 <main+362>  callq 0x1e8 <main+367>  mov %rax,%rcx  mov -0x10(%rbp),%edx  mov -0xc(%rbp),%eax  add %edx,%eax  mov %eax,%esi  mov %rcx,%rdi  callq 0x24e <main+469>  mov -0x14(%rbp),%eax  cmp $0x3,%eax  jne 0x97 <main+30>  lea -0x10(%rbp),%rax  mov %rax,%rsi  lea 0x0(%rip),%rdi # 0x282 <main+521>  callq 0x287 <main+526>  mov %rax,%rcx  mov -0x10(%rbp),%eax  mov -0xc(%rbp),%esi | %eax moved in -14(%rbp)  %eax compared to 1  Conditional jump  %rax loaded in -10(%rbp)  %rsi moved in %rax  %rdi loaded in 0(%rip)  Call  %rcx moved in %rax  %edx moved in -10(%rbp)  %eax moved in %rbp  %edx is subtracted in %eax  %eax moved in %edx  %esi moved in %eax  %rdi moved in %rcx  Call  %eax moved in -14(%rbp)  %eax is compared to 2  Conditional jump  %rax loaded in -10(%rbp)  %rsi moved in %rax  %rdi loaded in 0(%rip)  Call  %rcx moved in %rax  %edx moved in -10(%rbp)  %eax moved in %rbp  %edx is added in %eax  %eax moved in %edx  %rdi moved in %rcx  Call  %eax moved in -14(%rbp)  %eax is compared to 3  Conditional jump  %rax loaded in -10(%rbp)  %rsi moved in %rax  %rdi loaded in 0(%rip)  Call  %rcx moved in %rax  %eax moved in -10(%rbp)  %esi moved in %rbp |
| mov %rax,%rsi  mov %rdx,%rdi  callq 0x303 <main+650>  jmpq 0x97 <main+30>  mov $0x0,%eax  mov -0x8(%rbp),%rcx  xor %fs:0x28,%rcx  je 0x321 <main+680>  callq 0x321 <main+680>  leaveq  retq | %rsi moved in %rax  %rdi moved in %rdx  Unconditional jump  %eax moved in 0  %rcx moved in -8(%rbp)  Bitwise operation  if equal then jump  Leave  Return |

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

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

| **Blocks of Assembly Code** | **C++ Code** |
| --- | --- |
| push %rbp  mov %rsp,%rbp  lea 0x0(%rip),%rsi # 0xb <\_Z11DisplayMenuv+11>  lea 0x0(%rip),%rdi # 0x12 <\_Z11DisplayMenuv+18>  callq 0x17 <\_Z11DisplayMenuv+23>  lea 0x0(%rip),%rsi # 0x1e <\_Z11DisplayMenuv+30>  lea 0x0(%rip),%rdi # 0x25 <\_Z11DisplayMenuv+37>  callq 0x2a <\_Z11DisplayMenuv+42>  lea 0x0(%rip),%rsi # 0x31 <\_Z11DisplayMenuv+49>  lea 0x0(%rip),%rdi # 0x38 <\_Z11DisplayMenuv+56>  callq 0x3d <\_Z11DisplayMenuv+61>  lea 0x0(%rip),%rsi # 0x44 <\_Z11DisplayMenuv+68>  lea 0x0(%rip),%rdi # 0x4b <\_Z11DisplayMenuv+75>  callq 0x50 <\_Z11DisplayMenuv+80>  lea 0x0(%rip),%rsi # 0x57 <\_Z11DisplayMenuv+87>  lea 0x0(%rip),%rdi # 0x5e <\_Z11DisplayMenuv+94>  callq 0x63 <\_Z11DisplayMenuv+99>  lea 0x0(%rip),%rsi # 0x6a <\_Z11DisplayMenuv+106>  lea 0x0(%rip),%rdi # 0x71 <\_Z11DisplayMenuv+113>  callq 0x76 <\_Z11DisplayMenuv+118>  pop %rbp  retq | void DisplayMenu() {    cout << "----------------" << endl;    cout << "- 1) Subtract -" << endl;    cout << "- 2) Add -" << endl;    cout << "- 3) Divide -" << endl;    cout << "- 4) Exit -" << endl;    cout << "----------------" << endl;  } |
| mov -0x14(%rbp),%eax  cmp $0x5,%eax  je 0x308 <main+655>  lea 0x0(%rip),%rsi # 0xaa <main+49>  lea 0x0(%rip),%rdi # 0xb1 <main+56>  callq 0xb6 <main+61> | int choice = 0;    DisplayMenu(); |
| mov -0x14(%rbp),%eax  cmp $0x1,%eax  jne 0x1c9 <main+336>  lea -0x10(%rbp),%rax  mov %rax,%rsi  lea 0x0(%rip),%rdi # 0x142 <main+201>  callq 0x147 <main+206>  mov %rax,%rcx  mov -0x10(%rbp),%edx  mov -0xc(%rbp),%eax  sub %eax,%edx  mov %edx,%eax  mov %eax,%esi  mov %rcx,%rdi  callq 0x1af <main+310>  mov -0x14(%rbp),%eax  cmp $0x2,%eax  jne 0x268 <main+495>  lea -0x10(%rbp),%rax  mov %rax,%rsi  lea 0x0(%rip),%rdi # 0x1e3 <main+362>  callq 0x1e8 <main+367>  mov %rax,%rcx  mov -0x10(%rbp),%edx  mov -0xc(%rbp),%eax  add %edx,%eax  mov %eax,%esi  mov %rcx,%rdi  callq 0x24e <main+469>  mov -0x14(%rbp),%eax  cmp $0x3,%eax  jne 0x97 <main+30>  lea -0x10(%rbp),%rax  mov %rax,%rsi  lea 0x0(%rip),%rdi # 0x282 <main+521>  callq 0x287 <main+526>  mov %rax,%rcx  mov -0x10(%rbp),%eax  mov -0xc(%rbp),%esi  cltd  idiv %esi  mov %eax,%esi  mov %rcx,%rdi  callq 0x2ee <main+629> | cin >> choice;    while (choice != 4) {        int num1, num2;          cout << "Please Enter  First Number: " << endl;          cin >> num1;          cout << "Please Enter  Second Number: " << endl;          cin >> num2;      if (choice == 1) {        cout << num1 << " - " << num2 << " = " << num1 - num2 << endl;      } else if (choice == 2) {        cout << num1 << " + " << num2 << " = " << num1 + num2 << endl;      } else if (choice == 3) {        cout << num1 << " / " << num2 << " = " << (double) num1 / (double) num2 << endl;      }      else          {                cout << "ERROR: Please Enter Valid Input" << endl;          } |
| callq 0x321 <main+680>  leaveq  retq | DisplayMenu();          cin >> choice;        }      } |