# **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** |
| --- | --- |
| DisplayMenu()  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>  nop  pop %rbp  retq | Starts a function that displays the menu that is the following:  ----------------   * 1)Add – * 2)Subtract – * 3)Multiple – * 4)Exit –   ---------------- |
| Main()  push %rbp  mov %rsp,%rbp  sub $0x20,%rsp  mov %fs:0x28,%rax  mov %rax,-0x8(%rbp)  xor %eax,%eax  movl $0x0,-0x14(%rbp)  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>  lea 0x0(%rip),%rsi # 0xbd <main+68>  lea 0x0(%rip),%rdi # 0xc4 <main+75>  callq 0xc9 <main+80>  lea 0x0(%rip),%rsi # 0xd0 <main+87>  lea 0x0(%rip),%rdi # 0xd7 <main+94>  callq 0xdc <main+99>  lea 0x0(%rip),%rsi # 0xe3 <main+106>  lea 0x0(%rip),%rdi # 0xea <main+113>  callq 0xef <main+118>  lea 0x0(%rip),%rsi # 0xf6 <main+125>  lea 0x0(%rip),%rdi # 0xfd <main+132>  callq 0x102 <main+137>  lea 0x0(%rip),%rsi # 0x109 <main+144>  lea 0x0(%rip),%rdi # 0x110 <main+151>  callq 0x115 <main+156>    lea -0x14(%rbp),%rax  mov %rax,%rsi  lea 0x0(%rip),%rdi # 0x123 <main+170>  callq 0x128 <main+175> | This starts the main program. It assigns 0 as its initial input. Then a loop is created that the condition is that the loop will continue until the value is equal to 5, which terminates the program. |
| 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,%rdx  lea -0xc(%rbp),%rax  mov %rax,%rsi  mov %rdx,%rdi  callq 0x159 <main+224>  mov -0x10(%rbp),%eax  mov %eax,%esi  lea 0x0(%rip),%rdi # 0x165 <main+236>  callq 0x16a <main+241>  lea 0x0(%rip),%rsi # 0x171 <main+248>  mov %rax,%rdi  callq 0x179 <main+256>  mov %rax,%rdx  mov -0xc(%rbp),%eax  mov %eax,%esi  mov %rdx,%rdi  callq 0x189 <main+272>  lea 0x0(%rip),%rsi # 0x190 <main+279>  mov %rax,%rdi  callq 0x198 <main+287>  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 %rax,%rdx  mov 0x0(%rip),%rax # 0x1b9 <main+320>  mov %rax,%rsi  mov %rdx,%rdi  callq 0x1c4 <main+331>  jmpq 0x97 <main+30> | This compares the user input to 1 and jumps if it isn’t. This then takes two inputs from the user and subtracts them. It out puts the full equation then calls back the display and a new user input. |
| 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,%rdx  lea -0xc(%rbp),%rax  mov %rax,%rsi  mov %rdx,%rdi  callq 0x1fa <main+385>  mov -0x10(%rbp),%eax  mov %eax,%esi  lea 0x0(%rip),%rdi # 0x206 <main+397>  callq 0x20b <main+402>  lea 0x0(%rip),%rsi # 0x212 <main+409>  mov %rax,%rdi  callq 0x21a <main+417>  mov %rax,%rdx  mov -0xc(%rbp),%eax  mov %eax,%esi  mov %rdx,%rdi  callq 0x22a <main+433>  lea 0x0(%rip),%rsi # 0x231 <main+440>  mov %rax,%rdi  callq 0x239 <main+448>  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 %rax,%rdx  mov 0x0(%rip),%rax # 0x258 <main+479>  mov %rax,%rsi  mov %rdx,%rdi  callq 0x263 <main+490>  jmpq 0x97 <main+30> | This compares the user input to 2 and if its not two it jumps. This then takes in 2 values and adds them together. This then outputs the equation of the math. This then calls a new display menu and a new user input call. |
| 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,%rdx  lea -0xc(%rbp),%rax  mov %rax,%rsi  mov %rdx,%rdi  callq 0x299 <main+544>  mov -0x10(%rbp),%eax  mov %eax,%esi  lea 0x0(%rip),%rdi # 0x2a5 <main+556>  callq 0x2aa <main+561>  lea 0x0(%rip),%rsi # 0x2b1 <main+568>  mov %rax,%rdi  callq 0x2b9 <main+576>  mov %rax,%rdx  mov -0xc(%rbp),%eax  mov %eax,%esi  mov %rdx,%rdi  callq 0x2c9 <main+592>  lea 0x0(%rip),%rsi # 0x2d0 <main+599>  mov %rax,%rdi  callq 0x2d8 <main+607>  mov %rax,%rcx  mov -0x10(%rbp),%eax  mov -0xc(%rbp),%esi  cltd  idiv %esi  mov %eax,%esi  mov %rcx,%rdi  callq 0x2ee <main+629>  mov %rax,%rdx  mov 0x0(%rip),%rax # 0x2f8 <main+639>  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> | This compares the user input to the number 3 and if it doesn’t match it jumps. This then takes in 2 values and divides them. After it displays the math equation and then displays a new display menu with a new user input. |
| leaveq  retq | Exit program |

**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>  nop  pop %rbp  retq | void DisplayMenu{  cout<<"----------------"<<endl;  cout<<"- 1)Add -"<<endl;  cout<<"- 2)Subtract -"<<endl;  cout<<"- 3)Multiply -"<<endl;  cout<<"- 4)Exit -"<<endl;  cout<<"----------------"<<endl;  return;  } |
| push %rbp  mov %rsp,%rbp  sub $0x20,%rsp  mov %fs:0x28,%rax  mov %rax,-0x8(%rbp)  xor %eax,%eax  movl $0x0,-0x14(%rbp)  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>  lea 0x0(%rip),%rsi # 0xbd <main+68>  lea 0x0(%rip),%rdi # 0xc4 <main+75>  callq 0xc9 <main+80>  lea 0x0(%rip),%rsi # 0xd0 <main+87>  lea 0x0(%rip),%rdi # 0xd7 <main+94>  callq 0xdc <main+99>  lea 0x0(%rip),%rsi # 0xe3 <main+106>  lea 0x0(%rip),%rdi # 0xea <main+113>  callq 0xef <main+118>  lea 0x0(%rip),%rsi # 0xf6 <main+125>  lea 0x0(%rip),%rdi # 0xfd <main+132>  callq 0x102 <main+137>  lea 0x0(%rip),%rsi # 0x109 <main+144>  lea 0x0(%rip),%rdi # 0x110 <main+151>  callq 0x115 <main+156> | int main(){  int userChoice = 0;  int a;  int b;  while(userChoice != 5){  DisplayMenu();  cin>>input; |
| 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,%rdx  lea -0xc(%rbp),%rax  mov %rax,%rsi  mov %rdx,%rdi  callq 0x159 <main+224>  mov -0x10(%rbp),%eax  mov %eax,%esi  lea 0x0(%rip),%rdi # 0x165 <main+236>  callq 0x16a <main+241>  lea 0x0(%rip),%rsi # 0x171 <main+248>  mov %rax,%rdi  callq 0x179 <main+256>  mov %rax,%rdx  mov -0xc(%rbp),%eax  mov %eax,%esi  mov %rdx,%rdi  callq 0x189 <main+272>  lea 0x0(%rip),%rsi # 0x190 <main+279>  mov %rax,%rdi  callq 0x198 <main+287>  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 %rax,%rdx  mov 0x0(%rip),%rax # 0x1b9 <main+320>  mov %rax,%rsi  mov %rdx,%rdi  callq 0x1c4 <main+331>  jmpq 0x97 <main+30> | if(userChoice == 1){  cin>>a>>b;  cout<<a;  cout<<" - ";  cout<<b;  cout<<" = ";  cout<<a - b<<endl;  } |
| 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,%rdx  lea -0xc(%rbp),%rax  mov %rax,%rsi  mov %rdx,%rdi  callq 0x1fa <main+385>  mov -0x10(%rbp),%eax  mov %eax,%esi  lea 0x0(%rip),%rdi # 0x206 <main+397>  callq 0x20b <main+402>  lea 0x0(%rip),%rsi # 0x212 <main+409>  mov %rax,%rdi  callq 0x21a <main+417>  mov %rax,%rdx  mov -0xc(%rbp),%eax  mov %eax,%esi  mov %rdx,%rdi  callq 0x22a <main+433>  lea 0x0(%rip),%rsi # 0x231 <main+440>  mov %rax,%rdi  callq 0x239 <main+448>  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 %rax,%rdx  mov 0x0(%rip),%rax # 0x258 <main+479>  mov %rax,%rsi  mov %rdx,%rdi  callq 0x263 <main+490>  jmpq 0x97 <main+30> | Else if(userChoice== 2){  cin>>a>>b;  cout<<a;  cout<<" - ";  cout<<b;  cout<<" = ";  cout<<a + b<<endl;  } |
| 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,%rdx  lea -0xc(%rbp),%rax  mov %rax,%rsi  mov %rdx,%rdi  callq 0x299 <main+544>  mov -0x10(%rbp),%eax  mov %eax,%esi  lea 0x0(%rip),%rdi # 0x2a5 <main+556>  callq 0x2aa <main+561>  lea 0x0(%rip),%rsi # 0x2b1 <main+568>  mov %rax,%rdi  callq 0x2b9 <main+576>  mov %rax,%rdx  mov -0xc(%rbp),%eax  mov %eax,%esi  mov %rdx,%rdi  callq 0x2c9 <main+592>  lea 0x0(%rip),%rsi # 0x2d0 <main+599>  mov %rax,%rdi  callq 0x2d8 <main+607>  mov %rax,%rcx  mov -0x10(%rbp),%eax  mov -0xc(%rbp),%esi  cltd  idiv %esi  mov %eax,%esi  mov %rcx,%rdi  callq 0x2ee <main+629>  mov %rax,%rdx  mov 0x0(%rip),%rax # 0x2f8 <main+639>  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> | Else if(userChoise == 3){  cin>>a>>b;  cout<<a;  cout<<" - ";  cout<<b;  cout<<" = ";  cout<<a / b<<endl;  } |
| leaveq  retq | }  Return 0; |