# **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** |
| --- | --- |
| <+0>: push %rbp  <+1>: mov %rsp,%rbp  <+4>: sub $0x20,%rsp  <+8>: mov %fs:0x28,%rax  <+17>: mov %rax,-0x8(%rbp)  <+21>: xor %eax,%eax  <+23>: movl $0x0,-0x14(%rbp)  <+30>: mov -0x14(%rbp),%eax  <+33>: cmp $0x5,%eax  <+36>: je 0x308 <main+655>  <+42>: lea 0x0(%rip),%rsi # 0xaa <main+49>  <+49>: lea 0x0(%rip),%rdi # 0xb1 <main+56>  <+56>: callq 0xb6 <main+61>  <+61>: lea 0x0(%rip),%rsi # 0xbd <main+68>  <+68>: lea 0x0(%rip),%rdi # 0xc4 <main+75>  <+75>: callq 0xc9 <main+80>  <+80>: lea 0x0(%rip),%rsi # 0xd0 <main+87>  <+87>: lea 0x0(%rip),%rdi # 0xd7 <main+94>  <+94>: callq 0xdc <main+99>  <+99>: lea 0x0(%rip),%rsi # 0xe3 <main+106>  <+106>: lea 0x0(%rip),%rdi # 0xea <main+113> | This block of code initializes the first variable and starts a while loop that checks the users input and continues to loop unless the user enters 5. The while loop displays the menu that will prompt the user to enter 1 for add, 2 and subtract, 3 multiply and, 4 exit |
| <+113>: callq 0xef <main+118>  <+118>: lea 0x0(%rip),%rsi # 0xf6 <main+125>  <+125>: lea 0x0(%rip),%rdi # 0xfd <main+132>  <+132>: callq 0x102 <main+137>  <+137>: lea 0x0(%rip),%rsi # 0x109 <main+144>  <+144>: lea 0x0(%rip),%rdi # 0x110 <main+151>  <+151>: callq 0x115 <main+156>  <+156>: lea -0x14(%rbp),%rax  <+160>: mov %rax,%rsi  <+163>: lea 0x0(%rip),%rdi # 0x123 <main+170>  <+170>: callq 0x128 <main+175>  <+175>: mov -0x14(%rbp),%eax  <+178>: cmp $0x1,%eax  <+181>: jne 0x1c9 <main+336>   <+187>: lea -0x10(%rbp),%rax  <+191>: mov %rax,%rsi  <+194>: lea 0x0(%rip),%rdi # 0x142 <main+201>  <+201>: callq 0x147 <main+206>  <+206>: mov %rax,%rdx  <+209>: lea -0xc(%rbp),%rax  <+213>: mov %rax,%rsi  <+216>: mov %rdx,%rdi  <+219>: callq 0x159 <main+224>  <+224>: mov -0x10(%rbp),%eax  <+227>: mov %eax,%esi  <+229>: lea 0x0(%rip),%rdi # 0x165 <main+236>  <+236>: callq 0x16a <main+241>  <+241>: lea 0x0(%rip),%rsi # 0x171 <main+248>  <+248>: mov %rax,%rdi  <+251>: callq 0x179 <main+256> | Calls a string located at 0xef  Loads %rip into %rsi  Loads %rip into %rdi  Calls a string located at 0x102  Loads %rip into %rsi  Loads %rip into %rdi  Calls a string located at 0x115  Loads -0x14(%rbp) into %rax  Moves value held by %rax into %rsi  Loads %rip into %rdi  Calls a string located at 0x128  Moves value -0x14(%rbp) into %eax  Compares 0x1 and %eax  If it is not equal jumps to 0x1c9 or main 336  Loads -0x10(%rbp) into %rax  Moves value in %rax into %rsi  Loads 0x0(%rip) into %rdi  Calls a string at 0x147  Moves value in %rax into %rdx  Loads -0xc(%rbp) into %rax  Moves value in %rax into %rsi  Moves value in %rdx into %rdi  Calls a string at 0x159  Moves the value -0x10(%rbp) into %eax  Moves the value in %eax into %esi  Loads the %rip into %rdi  Calls a string located at 0x16a  Loads %rip into %rsi  Moves the value in %rax into %rdi  Calls a string at 0x179 |
| <+256>: mov %rax,%rdx  <+259>: mov -0xc(%rbp),%eax  <+262>: mov %eax,%esi  <+264>: mov %rdx,%rdi  <+267>: callq 0x189 <main+272>  <+272>: lea 0x0(%rip),%rsi # 0x190 <main+279>  <+279>: mov %rax,%rdi  <+282>: callq 0x198 <main+287>  <+287>: mov %rax,%rcx  <+290>: mov -0x10(%rbp),%edx  <+293>: mov -0xc(%rbp),%eax  <+296>: sub %eax,%edx  <+298>: mov %edx,%eax  <+300>: mov %eax,%esi  <+302>: mov %rcx,%rdi  <+305>: callq 0x1af <main+310>  <+310>: mov %rax,%rdx  <+313>: mov 0x0(%rip),%rax # 0x1b9 <main+320>  <+320>: mov %rax,%rsi  <+323>: mov %rdx,%rdi  <+326>: callq 0x1c4 <main+331>  <+331>: jmpq 0x97 <main+30>  <+336>: mov -0x14(%rbp),%eax  <+339>: cmp $0x2,%eax  <+342>: jne 0x268 <main+495>  <+348>: lea -0x10(%rbp),%rax  <+352>: mov %rax,%rsi  <+355>: lea 0x0(%rip),%rdi # 0x1e3 <main+362>  <+362>: callq 0x1e8 <main+367>  <+367>: mov %rax,%rdx  <+370>: lea -0xc(%rbp),%rax  <+374>: mov %rax,%rsi  <+377>: mov %rdx,%rdi  <+380>: callq 0x1fa <main+385>  <+385>: mov -0x10(%rbp),%eax  <+388>: mov %eax,%esi  <+390>: lea 0x0(%rip),%rdi # 0x206 <main+397>  <+397>: callq 0x20b <main+402>  <+402>: lea 0x0(%rip),%rsi # 0x212 <main+409>  <+409>: mov %rax,%rdi  <+412>: callq 0x21a <main+417>  <+417>: mov %rax,%rdx  <+420>: mov -0xc(%rbp),%eax  <+423>: mov %eax,%esi  <+425>: mov %rdx,%rdi  <+428>: callq 0x22a <main+433>  <+433>: lea 0x0(%rip),%rsi # 0x231 <main+440  <+440>: mov %rax,%rdi  <+443>: callq 0x239 <main+448>  <+448>: mov %rax,%rcx  <+451>: mov -0x10(%rbp),%edx  <+454>: mov -0xc(%rbp),%eax  <+457>: add %edx,%eax  <+459>: mov %eax,%esi  <+461>: mov %rcx,%rdi  <+464>: callq 0x24e <main+469> | Moves the value in %rax into %rdx  Moves the value in -0xc(%rbp) into %eax  Moves the value in %eax into %esi  Moves the value in %rdx into %rdi  Calls a string located at 0x189  Loads the %rip into %rsi  Moves the value In %ax into %rdi  Calls a string in 0x198  Moves the value in %rax into %rcx  Moves the value -0x10(%rbp) into %edx  Moves the value -0xc(%rbp) into %eax  Subtracts the values in %eax and %edx  Moves the value in %edx into %eax  Moves the value in %eax into %esi  Moves the value in %rcx into %rdi  Calls a string at 0x1af  Moves the value in %rax into %rdx  Moves the value in %rip into %rax  Moves the value %rax into %rsi  Moves the value in %rdx into %rdi  Calls a string at 0x1c4  Jumps directly to 0x97 or main 30  Moves -0x14(%rbp) into %eax  Compares 0x2 and %eax  Jumps to 0x268 if not equal  Loads -0x10(%rbp) into %rax  Moves the value in %rax into %rsi  Loads %rip into %rdi  Calls a string 0x1e8  Moves the value in %rax into %rdx  Loads -0xc(%rbp) into %rax  Moves the value in %rax into %rsi  Moves the value in %rdx into %rdi  Calls a string at 0x1fa  Moves the value -0x10(%rbp) into %eax  Moves the value in %eax into %esi  Loas %rip into %rdi  Calls a string at 0x20b  Loads %rip into %rsi  Moves the value in %rax into %rdi  Calls a string at 0x21a  Moves the value in %rax into %rdx  Moves the value in -0xc(%rbp) into %eax  Moves the value in %eax into %esi  Moves the value in %rdx into %rdi  Calls a string at 0x22a  Loads %rip into %rsi  Moves the value in %rax into %rdi  Calls a string at 0x239  Moves the value in %rax into %rcx  Moves the value in -0x10(%rbp) into %edx  Moves the value in -0xc(%rbp) into %eax  Adds the values held by %edx and %eax  Moves the value in %eax into %esi  Moves the value in %rcx into %rdi  Calls a string at 0x24e |
| <+469>: mov %rax,%rdx  <+472>: mov 0x0(%rip),%rax # 0x258 <main+479>  <+479>: mov %rax,%rsi  <+482>: mov %rdx,%rdi  <+485>: callq 0x263 <main+490>  <+490>: jmpq 0x97 <main+30>  <+495>: mov -0x14(%rbp),%eax  <+498>: cmp $0x3,%eax  <+501>: jne 0x97 <main+30>  <+507>: lea -0x10(%rbp),%rax  <+511>: mov %rax,%rsi  <+514>: lea 0x0(%rip),%rdi # 0x282 <main+521>  <+521>: callq 0x287 <main+526>  <+526>: mov %rax,%rdx  <+529>: lea -0xc(%rbp),%rax  <+533>: mov %rax,%rsi  <+536>: mov %rdx,%rdi  <+539>: callq 0x299 <main+544>  <+544>: mov -0x10(%rbp),%eax  <+547>: mov %eax,%esi  <+549>: lea 0x0(%rip),%rdi # 0x2a5 <main+556>  <+556>: callq 0x2aa <main+561>  <+561>: lea 0x0(%rip),%rsi # 0x2b1 <main+568>  <+568>: mov %rax,%rdi  <+571>: callq 0x2b9 <main+576>  <+576>: mov %rax,%rdx  <+579>: mov -0xc(%rbp),%eax  <+582>: mov %eax,%esi  <+584>: mov %rdx,%rdi  <+587>: callq 0x2c9 <main+592>  <+592>: lea 0x0(%rip),%rsi # 0x2d0 <main+599>  <+599>: mov %rax,%rdi  <+602>: callq 0x2d8 <main+607> | Moves the value in %rax into %rdx  Moves the value 0x0(%rip) into %rax  Moves the value in %rax into %rsi  Moves the value in %rdx into %rdi  Calls a string at 0x263  Jumps to 0x97  Moves the value -0x14(%rbp) into %eax  Compares $0x3 and %eax  Jumps to 0x97 if not equal  Loads -0x10(%rbp) into %rax  Moves the value in %rax into %rsi  Loads %rip into %rdi  Calls a string located at 0x287  Moves the value in %rax into %rdx  Loads -0xc(%rbp) into %rax  Moves the value in %rax into %rsi  Moves the value in %rdx into %rdi  Calls a string located at 0x299  Moves the value -0x10(%rbp) into %eax  Moves the value in %eax into %esi  Loads %rip into %rdi  Calls a string at 0x2aa  Loads %rip into %rsi  Moves the value in %rax into %rdi  Calls a string at 0x2b9  Moves the value in %rax into %rdx  Moves the -0xc(%rbp) into %eax  Moves the value in %eax into %esi  Moves the value in %rdx into %rdi  Calls a string at 0x2c9  Loads %rip into %rsi  Moves the value in %rax into %rdi  Calls a string at 0x2d8 |
| <+607>: mov %rax,%rcx  <+610>: mov -0x10(%rbp),%eax  <+613>: mov -0xc(%rbp),%esi  <+616>: cltd  <+617>: idiv %esi  <+619>: mov %eax,%esi  <+621>: mov %rcx,%rdi  <+624>: callq 0x2ee <main+629>  <+629>: mov %rax,%rdx  <+632>: mov 0x0(%rip),%rax # 0x2f8 <main+639>  <+639>: mov %rax,%rsi  <+642>: mov %rdx,%rdi  <+645>: callq 0x303 <main+650>  <+650>: jmpq 0x97 <main+30>  <+655>: mov $0x0,%eax  <+660>: mov -0x8(%rbp),%rcx  <+664>: xor %fs:0x28,%rcx  <+673>: je 0x321 <main+680>  <+675>: callq 0x321 <main+680>  <+680>: leaveq  <+681>: retq | Moves the value in %rax into %rcx  Moves the value -0x10(%rbp) into %eax  Moves the value -0xc(%rbp) into %esi  Clears the %ex register  Divides %esi  Moves the value in %eax into %esi  Moves the value in %rcx into %rdi  Calls a string at 0x2ee  Moves the value in %rax into %rdx  Moves %rip into %rax  Moves %rax into %rsi  Moves %rdx into %rdi  Calls a string at 0x303  Jumps to 0x97  Moves $0x0 into %eax  Moves -0x8(%rbp) into %rcx  Does a bit wise Exclusive OR  Jumps to 0x321 if it is equal  Calls a string at 0x321 |

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

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

| **Blocks of Assembly Code** | **C++ Code** |
| --- | --- |
| <+0>: push %rbp  <+1>: mov %rsp,%rbp  <+4>: sub $0x20,%rsp  <+8>: mov %fs:0x28,%rax  <+17>: mov %rax,-0x8(%rbp)  <+21>: xor %eax,%eax  <+23>: movl $0x0,-0x14(%rbp)  <+30>: mov -0x14(%rbp),%eax  <+33>: cmp $0x5,%eax  <+36>: je 0x308 <main+655>  <+42>: lea 0x0(%rip),%rsi # 0xaa <main+49>  <+49>: lea 0x0(%rip),%rdi # 0xb1 <main+56>  <+56>: callq 0xb6 <main+61>  <+61>: lea 0x0(%rip),%rsi # 0xbd <main+68>  <+68>: lea 0x0(%rip),%rdi # 0xc4 <main+75>  <+75>: callq 0xc9 <main+80>  <+80>: lea 0x0(%rip),%rsi # 0xd0 <main+87>  <+87>: lea 0x0(%rip),%rdi # 0xd7 <main+94>  <+94>: callq 0xdc <main+99>  <+99>: lea 0x0(%rip),%rsi # 0xe3 <main+106>  <+106>: lea 0x0(%rip),%rdi # 0xea <main+113> | #include <iostream>  #include <cstdlib>  int main() {  int choice = 0;  int num1, num2;  while (choice != 5) {  std::cout << "Menu:\n";  std::cout << "1. Subtraction\n";  std::cout << "2. Addition\n";  std::cout << "3. Division\n";  std::cout << "4. Exit\n";  std::cout << "Enter your choice: ";  std::cin >> choice; |
| <+113>: callq 0xef <main+118>  <+118>: lea 0x0(%rip),%rsi # 0xf6 <main+125>  <+125>: lea 0x0(%rip),%rdi # 0xfd <main+132>  <+132>: callq 0x102 <main+137>  <+137>: lea 0x0(%rip),%rsi # 0x109 <main+144>  <+144>: lea 0x0(%rip),%rdi # 0x110 <main+151>  <+151>: callq 0x115 <main+156>  <+156>: lea -0x14(%rbp),%rax  <+160>: mov %rax,%rsi  <+163>: lea 0x0(%rip),%rdi # 0x123 <main+170>  <+170>: callq 0x128 <main+175>  <+175>: mov -0x14(%rbp),%eax  <+178>: cmp $0x1,%eax  <+181>: jne 0x1c9 <main+336>   <+187>: lea -0x10(%rbp),%rax  <+191>: mov %rax,%rsi  <+194>: lea 0x0(%rip),%rdi # 0x142 <main+201>  <+201>: callq 0x147 <main+206>  <+206>: mov %rax,%rdx  <+209>: lea -0xc(%rbp),%rax  <+213>: mov %rax,%rsi  <+216>: mov %rdx,%rdi  <+219>: callq 0x159 <main+224>  <+224>: mov -0x10(%rbp),%eax  <+227>: mov %eax,%esi  <+229>: lea 0x0(%rip),%rdi # 0x165 <main+236>  <+236>: callq 0x16a <main+241>  <+241>: lea 0x0(%rip),%rsi # 0x171 <main+248>  <+248>: mov %rax,%rdi  <+251>: callq 0x179 <main+256> | std::cout << "Enter your choice: ";  std::cin >> choice;  switch (choice) {  case 1:  std::cout << "Enter two numbers: ";  std::cin >> num1 >> num2;  std::cout << num1 << " - " << num2 << " = " << (num1 - num2) << std::endl;  break; |
| <+256>: mov %rax,%rdx  <+259>: mov -0xc(%rbp),%eax  <+262>: mov %eax,%esi  <+264>: mov %rdx,%rdi  <+267>: callq 0x189 <main+272>  <+272>: lea 0x0(%rip),%rsi # 0x190 <main+279>  <+279>: mov %rax,%rdi  <+282>: callq 0x198 <main+287>  <+287>: mov %rax,%rcx  <+290>: mov -0x10(%rbp),%edx  <+293>: mov -0xc(%rbp),%eax  <+296>: sub %eax,%edx  <+298>: mov %edx,%eax  <+300>: mov %eax,%esi  <+302>: mov %rcx,%rdi  <+305>: callq 0x1af <main+310>  <+310>: mov %rax,%rdx  <+313>: mov 0x0(%rip),%rax # 0x1b9 <main+320>  <+320>: mov %rax,%rsi  <+323>: mov %rdx,%rdi  <+326>: callq 0x1c4 <main+331>  <+331>: jmpq 0x97 <main+30>  <+336>: mov -0x14(%rbp),%eax  <+339>: cmp $0x2,%eax  <+342>: jne 0x268 <main+495>  <+348>: lea -0x10(%rbp),%rax  <+352>: mov %rax,%rsi  <+355>: lea 0x0(%rip),%rdi # 0x1e3 <main+362>  <+362>: callq 0x1e8 <main+367>  <+367>: mov %rax,%rdx  <+370>: lea -0xc(%rbp),%rax  <+374>: mov %rax,%rsi  <+377>: mov %rdx,%rdi  <+380>: callq 0x1fa <main+385>  <+385>: mov -0x10(%rbp),%eax  <+388>: mov %eax,%esi  <+390>: lea 0x0(%rip),%rdi # 0x206 <main+397>  <+397>: callq 0x20b <main+402>  <+402>: lea 0x0(%rip),%rsi # 0x212 <main+409>  <+409>: mov %rax,%rdi  <+412>: callq 0x21a <main+417>  <+417>: mov %rax,%rdx  <+420>: mov -0xc(%rbp),%eax  <+423>: mov %eax,%esi  <+425>: mov %rdx,%rdi  <+428>: callq 0x22a <main+433>  <+433>: lea 0x0(%rip),%rsi # 0x231 <main+440  <+440>: mov %rax,%rdi  <+443>: callq 0x239 <main+448>  <+448>: mov %rax,%rcx  <+451>: mov -0x10(%rbp),%edx  <+454>: mov -0xc(%rbp),%eax  <+457>: add %edx,%eax  <+459>: mov %eax,%esi  <+461>: mov %rcx,%rdi  <+464>: callq 0x24e <main+469> | case 2:  std::cout << "Enter two numbers: ";  std::cin >> num1 >> num2;  std::cout << num1 << " + " << num2 << " = " << (num1 + num2) << std::endl;  break; |
| <+469>: mov %rax,%rdx  <+472>: mov 0x0(%rip),%rax # 0x258 <main+479>  <+479>: mov %rax,%rsi  <+482>: mov %rdx,%rdi  <+485>: callq 0x263 <main+490>  <+490>: jmpq 0x97 <main+30>  <+495>: mov -0x14(%rbp),%eax  <+498>: cmp $0x3,%eax  <+501>: jne 0x97 <main+30>  <+507>: lea -0x10(%rbp),%rax  <+511>: mov %rax,%rsi  <+514>: lea 0x0(%rip),%rdi # 0x282 <main+521>  <+521>: callq 0x287 <main+526>  <+526>: mov %rax,%rdx  <+529>: lea -0xc(%rbp),%rax  <+533>: mov %rax,%rsi  <+536>: mov %rdx,%rdi  <+539>: callq 0x299 <main+544>  <+544>: mov -0x10(%rbp),%eax  <+547>: mov %eax,%esi  <+549>: lea 0x0(%rip),%rdi # 0x2a5 <main+556>  <+556>: callq 0x2aa <main+561>  <+561>: lea 0x0(%rip),%rsi # 0x2b1 <main+568>  <+568>: mov %rax,%rdi  <+571>: callq 0x2b9 <main+576>  <+576>: mov %rax,%rdx  <+579>: mov -0xc(%rbp),%eax  <+582>: mov %eax,%esi  <+584>: mov %rdx,%rdi  <+587>: callq 0x2c9 <main+592>  <+592>: lea 0x0(%rip),%rsi # 0x2d0 <main+599>  <+599>: mov %rax,%rdi  <+602>: callq 0x2d8 <main+607> | case 3:  std::cout << "Enter two numbers: ";  std::cin >> num1 >> num2;  if (num2 != 0) {  std::cout << num1 << " / " << num2 << " = " << (num1 / num2) << std::endl;  } else {  std::cout << "Error: Division by zero!" << std::endl;  }  break; |
| <+607>: mov %rax,%rcx  <+610>: mov -0x10(%rbp),%eax  <+613>: mov -0xc(%rbp),%esi  <+616>: cltd  <+617>: idiv %esi  <+619>: mov %eax,%esi  <+621>: mov %rcx,%rdi  <+624>: callq 0x2ee <main+629>  <+629>: mov %rax,%rdx  <+632>: mov 0x0(%rip),%rax # 0x2f8 <main+639>  <+639>: mov %rax,%rsi  <+642>: mov %rdx,%rdi  <+645>: callq 0x303 <main+650>  <+650>: jmpq 0x97 <main+30>  <+655>: mov $0x0,%eax  <+660>: mov -0x8(%rbp),%rcx  <+664>: xor %fs:0x28,%rcx  <+673>: je 0x321 <main+680>  <+675>: callq 0x321 <main+680>  <+680>: leaveq  <+681>: retq | case 4:  std::cout << "Exiting program." << std::endl;  exit(0);  default:  std::cout << "Invalid choice. Please try again." << std::endl;  }  }  return 0;  } |