# **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 0xd02 <main+655> | push rbp to the top of the stack  moves rsp into rbp  subtract 0x20 from rsp  move %fs:0x28 into rax  move rax into -0x8(%rbp)  set eax to 0  move 0x0 into -0x14(%rbp)  move -0x14(%rbp) into eax  compare 0x5 with eax  jump to +655 if equal to 0xd02 |
| <+42>: lea 0x3a5(%rip),%rsi # 0xe49  <+49>: lea 0x201575(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  <+56>: callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  <+61>: lea 0x3a4(%rip),%rsi # 0xe5b  <+68>: lea 0x201562(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  <+75>: callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  <+80>: lea 0x39c(%rip),%rsi # 0xe66  <+87>: lea 0x20154f(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  <+94>: callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  <+99>: lea 0x399(%rip),%rsi # 0xe76  <+106>: lea 0x20153c(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  <+113>: callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  <+118>: lea 0x396(%rip),%rsi # 0xe86  <+125>: lea 0x201529(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  <+132>: callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  <+137>: lea 0x346(%rip),%rsi # 0xe49  <+144>: lea 0x201516(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  <+151>: callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  <+156>: lea -0x14(%rbp),%rax  <+160>: mov %rax,%rsi  <+163>: lea 0x201623(%rip),%rdi # 0x202140 <\_ZSt3cin@@GLIBCXX\_3.4>  <+170>: callq 0x870 <\_ZNSirsERi@plt>  <+175>: mov -0x14(%rbp),%eax | load 0x3a5(%rip) into rsi  load 0x201575(%rip) into rdi (cout)  call function (cout)  load 0x3a4(%rip) into rsi  load 0x201562(%rip) into rdi (cout)  call function (cout)  load 0x39c(%rip) into rsi  load 0x20154f(%rip) into rdi (cout)  call cout function  load 0x399(%rip) into rsi  load 0x20153c(%rip) into rdi (cout)  call cout function  load 0x396(%rip) into rsi  load 0x201529(%rip) into rdi (cout)  call cout function  load 0x346(%rip) into rsi  load 0x201516(%rip) into rdi (cout)  call cout function  load -0x14(%rbp) into rax  move rax into rsi  load 0x201623(%rip) rdi (cin)  call cin function  move -0x14(%rbp) into eax |
| <+178>: cmp $0x1,%eax  <+181>: jne 0xbc3 <main+336>  <+187>: lea -0x10(%rbp),%rax  <+191>: mov %rax,%rsi  <+194>: lea 0x201604(%rip),%rdi # 0x202140 <\_ZSt3cin@@GLIBCXX\_3.4>  <+201>: callq 0x870 <\_ZNSirsERi@plt>  <+206>: mov %rax,%rdx  <+209>: lea -0xc(%rbp),%rax  <+213>: mov %rax,%rsi  <+216>: mov %rdx,%rdi  <+219>: callq 0x870 <\_ZNSirsERi@plt>  <+224>: mov -0x10(%rbp),%eax  <+227>: mov %eax,%esi  <+229>: lea 0x2014c1(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  <+236>: callq 0x8d0 <\_ZNSolsEi@plt>  <+241>: lea 0x327(%rip),%rsi # 0xe92  <+248>: mov %rax,%rdi  <+251>: callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  <+256>: mov %rax,%rdx  <+259>: mov -0xc(%rbp),%eax  <+262>: mov %eax,%esi  <+264>: mov %rdx,%rdi  <+267>: callq 0x8d0 <\_ZNSolsEi@plt>  <+272>: lea 0x30c(%rip),%rsi # 0xe96  <+279>: mov %rax,%rdi  <+282>: callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  <+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 0x8d0 <\_ZNSolsEi@plt>  <+310>: mov %rax,%rdx  <+313>: mov 0x20141d(%rip),%rax # 0x201fd0  <+320>: mov %rax,%rsi  <+323>: mov %rdx,%rdi  <+326>: callq 0x8a0 <\_ZNSolsEPFRSoS\_E@plt>  <+331>: jmpq 0xa91 <main+30>  <+336>: mov -0x14(%rbp),%eax | compare 0x1 with eax  jump to +336 if not equal  load -0x10(%rbp) into rax  move rax into rsi  load 0x201604(%rip) into rdi (cin)  call cin function  move rax into rdx  load -0xc(%rbp) into rdx  move rax into rsi  move rdx into rdi  call function  move -0x10(%rbp) into eax  move eax into esi  load 0x2014c1(%rip) into rdi (cout)  call cout function  load 0x327(%rip) into rsi  move rax into rdi  call function  move rax into rdx  move -0xc(%rbp) into eax  move eax into esi  move rdx into rdi  call function  load 0x30c(%rip) into rsi  move rax into rdi  call function  move rax into rcx  move -0x10(%rbp) into edx  move -0xc(%rbp) into eax  subtract eax from edx  move edx into eax  move eax into esi  move rcx into rdi  call function  move rax into rdx  move 0x20141d(%rip) into rax  move rax into rsi  move rdx into rdi  call function  jump to +30  move -0x14(%rbp) into eax |
| <+339>: cmp $0x2,%eax  <+342>: jne 0xc62 <main+495>  <+348>: lea -0x10(%rbp),%rax  <+352>: mov %rax,%rsi  <+355>: lea 0x201563(%rip),%rdi # 0x202140 <\_ZSt3cin@@GLIBCXX\_3.4>  <+362>: callq 0x870 <\_ZNSirsERi@plt>  <+367>: mov %rax,%rdx  <+370>: lea -0xc(%rbp),%rax  <+374>: mov %rax,%rsi  <+377>: mov %rdx,%rdi  <+380>: callq 0x870 <\_ZNSirsERi@plt>  <+385>: mov -0x10(%rbp),%eax  <+388>: mov %eax,%esi  <+390>: lea 0x201420(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  <+397>: callq 0x8d0 <\_ZNSolsEi@plt> <+402>: lea 0x286(%rip),%rsi # 0xe92  <+409>: mov %rax,%rdi  <+412>: callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  <+417>: mov %rax,%rdx  <+420>: mov -0xc(%rbp),%eax  <+423>: mov %eax,%esi  <+425>: mov %rdx,%rdi  <+428>: callq 0x8d0 <\_ZNSolsEi@plt>  <+433>: lea 0x26b(%rip),%rsi # 0xe96  <+440>: mov %rax,%rdi  <+443>: callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  <+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 0x8d0 <\_ZNSolsEi@plt>  <+469>: mov %rax,%rdx  <+472>: mov 0x20137e(%rip),%rax # 0x201fd0  <+479>: mov %rax,%rsi  <+482>: mov %rdx,%rdi  <+485>: callq 0x8a0 <\_ZNSolsEPFRSoS\_E@plt>  <+490>: jmpq 0xa91 <main+30> | compare 0x2 with eax  jump to 495 if not equal  load 0x10(%rbp) into rax  move rax into rsi  load 0x201563(%rip) into rdi (cin)  call cin function  move rax into rdx  load 0xc(%rbp) into rax  move rax into rsi  move rdx into rdi  call function  move 0x10(%rbp) into eax  move eax into esi  load 0x201420(%rip) into rdi (cout)  call cout function  load 0x286(%rip) into rsi  move rax into rdi  call function  move rax into rdx  move -0xc(%rbp) into eax  move eax into esi  move rdx into rdi  call function  load 0x26b(%rip) into rsi  move rax into rdi  call function  move rax into rcx  move -0x10(%rbp) into edx  move -0xc(%rbp) into eax  add edx to eax  move eax into esi  move rcx into rdi  call function  move rax into rdx  move 0x20137e(%rip) into rax  move rax into rsi  move rdx into rdi  call function  jump to +30 |
| <+495>: mov -0x14(%rbp),%eax  <+498>: cmp $0x3,%eax  <+501>: jne 0xa91 <main+30>  <+507>: lea -0x10(%rbp),%rax  <+511>: mov %rax,%rsi  <+514>: lea 0x2014c4(%rip),%rdi # 0x202140 <\_ZSt3cin@@GLIBCXX\_3.4>  <+521>: callq 0x870 <\_ZNSirsERi@plt>  <+526>: mov %rax,%rdx  <+529>: lea -0xc(%rbp),%rax  <+533>: mov %rax,%rsi  <+536>: mov %rdx,%rdi  <+539>: callq 0x870 <\_ZNSirsERi@plt>  <+544>: mov -0x10(%rbp),%eax  <+547>: mov %eax,%esi  <+549>: lea 0x201381(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  <+556>: callq 0x8d0 <\_ZNSolsEi@plt>  <+561>: lea 0x1e7(%rip),%rsi # 0xe92  <+568>: mov %rax,%rdi  <+571>: callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  <+576>: mov %rax,%rdx  <+579>: mov -0xc(%rbp),%eax  <+582>: mov %eax,%esi  <+584>: mov %rdx,%rdi  <+587>: callq 0x8d0 <\_ZNSolsEi@plt>  <+592>: lea 0x1cc(%rip),%rsi # 0xe96  <+599>: mov %rax,%rdi  <+602>: callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  <+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 0x8d0 <\_ZNSolsEi@plt>  <+629>: mov %rax,%rdx  <+632>: mov 0x2012de(%rip),%rax # 0x201fd0  <+639>: mov %rax,%rsi  <+642>: mov %rdx,%rdi  <+645>: callq 0x8a0 <\_ZNSolsEPFRSoS\_E@plt>  <+650>: jmpq 0xa91 <main+30> | move -0x14(%rbp) into eax  compare 0x3 with eax  jump to +30 if not equal  load -0x10(%rbp) into rax  move rax into rsi  load 0x2014c4(%rip) into rdi (cin)  call cin function  move rax into rdx  load -0xc(%rbp) into rax  move rax into rsi  move rdx into rdi  call function  move -0x10(%rbp) into eax  move eax into esi  load 0x201381(%rip) into rdi (cout)  call cout function  load 0x1e7(%rip) into rsi  move rax into rdi  call function  move rax into rdx  move -0xc(%rbp) into eax  move eax into esi  move rdx into rdi  call function  load 0x1cc(%rip) into rsi  move rax into rdi  call function  move rax into rcx  move -0x10(%rbp) into eax  move -0xc(%rbp) into esi  sign-extend eax into edx::eax  sign divide esi  move eax into esi  move rcx into rdi  call function  move rax into rdx  move 0x2012de(%rip) into rax  move rax into rsi  move rdx into rdi  call function  jump to +30 |
| <+655>: mov $0x0,%eax  <+660>: mov -0x8(%rbp),%rcx  <+664>: xor %fs:0x28,%rcx  <+673>: je 0xd1b <main+680>  <+675>: callq 0x8b0 <\_\_stack\_chk\_fail@plt>  <+680>: leaveq  <+681>: retq | move 0x0 into eax  move -0x8(%rbp) rcx  compare if %fs:0x28 is not equal to rcx  jump to +680 if equal  call function  leave function  return |

**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 0xd02 <main+655> | /\*  \* Binary file created by SNHU  \* C++ from binary file created by Dominic Drury  \*  \* As a note for this recreation. I observed some issues from the start with my file. The first is that there  \* is only a main function and a display function. With this program being one that does 3 distinct functions,  \* I feel that seperating this program into a main, display, add, subtract, and multiply will be best for  \* readabilty, maintainability, and security testing. I also observed that the exit does not function and the  \* arithmatic is not accurate and will need to be redone from scratch. There is also no validation for user input  \* so if a user enters a non comppliant input the program becomes unresponsive. The program also does not give good  \* instructions to the user during computations for what the program is asking the user for that will need to be  \* corrected.  \*/    #include<iostream>    using namespace std;    void displayMenu();  long long add(int num1, int num2);  long long subtract(int num1, int num2);  long long multiply(int num1, int num2);    int main() {  string userInput = ""; // Initialized variable for getting user input, by using string invalid inputs are easier to manage  int num1, num2 = 0; // Initialized varaibles for user inputs  long long result = 0; // Initialized variable for result, set to long long to accomodate range of int max \* int max    while (userInput != "4") { // While the userInput is not 4 (exit from the menu)  displayMenu(); // Display the menu to the user  cin >> userInput; // Gets user menu choice from the provided menu    if (userInput == "1") { // If the user selects add from the menu  // Get first number from the user  cout << "Please enter the first number you wish to add" << endl;  cin >> num1;  if (cin.fail()) { // ensures that an integer was entered  cin.clear(); // clears the cin  cin.ignore(256, '\n'); // ignores cin so that it is prepared for new input  cout << "Invalid input, please enter a number between -2,147,483,646 and +2,147,483,646" << endl;  cin >> num1;  }    // Get second number from the user  cout << "Please enter the second number you wish to add" << endl;  cin >> num2;  if (cin.fail()) { // ensures that an integer was entered  cin.clear(); // clears the cin  cin.ignore(256, '\n'); // ignores cin so that it is prepared for new input  cout << "Invalid input, please enter a number between -2,147,483,646 and +2,147,483,646" << endl;  cin >> num2;  }    result = add(num1, num2);    cout << num1 << " + " << num2 << " = " << result << endl;  }  else if (userInput == "2") { // If the user selects subtract from the menu  // Get first number from the user  cout << "Please enter the first number you wish to subtract" << endl;  cin >> num1;  if (cin.fail()) { // ensures that an integer was entered  cin.clear(); // clears the cin  cin.ignore(256, '\n'); // ignores cin so that it is prepared for new input  cout << "Invalid input, please enter a number between -2,147,483,646 and +2,147,483,646" << endl;  cin >> num1;  }    // Get second number from the user  cout << "Please enter the second number you wish to subtract" << endl;  cin >> num2;  if (cin.fail()) { // ensures that an integer was entered  cin.clear(); // clears the cin  cin.ignore(256, '\n'); // ignores cin so that it is prepared for new input  cout << "Invalid input, please enter a number between -2,147,483,646 and +2,147,483,646" << endl;  cin >> num2;  }    result = subtract(num1, num2);    cout << num1 << " - " << num2 << " = " << result << endl;  }  else if (userInput == "3") { // If the user selects multiply from the menu  // Get first number from the user  cout << "Please enter the first number you wish to multiply" << endl;  cin >> num1;  if (cin.fail()) { // ensures that an integer was entered  cin.clear(); // clears the cin  cin.ignore(256, '\n'); // ignores cin so that it is prepared for new input  cout << "Invalid input, please enter a number between -2,147,483,646 and +2,147,483,646" << endl;  cin >> num1;  }    // Get second number from the user  cout << "Please enter the second number you wish to multiply" << endl;  cin >> num2;  if (cin.fail()) { // ensures that an integer was entered  cin.clear(); // clears the cin  cin.ignore(256, '\n'); // ignores cin so that it is prepared for new input  cout << "Invalid input, please enter a number between -2,147,483,646 and +2,147,483,646" << endl;  cin >> num2;  }    result = multiply(num1, num2);    cout << num1 << " \* " << num2 << " = " << result << endl;  }  else { // catchall for non accepted inputs from the user  cout << "You entered an invalid input, please select an option from the menu" << endl;  }  }  return 0;  } |
| <+42>: lea 0x3a5(%rip),%rsi # 0xe49  <+49>: lea 0x201575(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  <+56>: callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  <+61>: lea 0x3a4(%rip),%rsi # 0xe5b  <+68>: lea 0x201562(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  <+75>: callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  <+80>: lea 0x39c(%rip),%rsi # 0xe66  <+87>: lea 0x20154f(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  <+94>: callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  <+99>: lea 0x399(%rip),%rsi # 0xe76  <+106>: lea 0x20153c(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  <+113>: callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  <+118>: lea 0x396(%rip),%rsi # 0xe86  <+125>: lea 0x201529(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  <+132>: callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  <+137>: lea 0x346(%rip),%rsi # 0xe49  <+144>: lea 0x201516(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  <+151>: callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  <+156>: lea -0x14(%rbp),%rax  <+160>: mov %rax,%rsi  <+163>: lea 0x201623(%rip),%rdi # 0x202140 <\_ZSt3cin@@GLIBCXX\_3.4>  <+170>: callq 0x870 <\_ZNSirsERi@plt>  <+175>: mov -0x14(%rbp),%eax | /\*  \* Displays the menu to the user  \* ----------------  \* - 1)Add -  \* - 2)Subtract -  \* - 3)Multiply -  \* - 4)Exit -  \* ----------------  \*/  void displayMenu() {  cout << "----------------" << endl;  cout << "- 1)Add -" << endl;  cout << "- 2)Subtract -" << endl;  cout << "- 3)Multiply -" << endl;  cout << "- 4)Exit -" << endl;  cout << "----------------" << endl;  } |
| <+178>: cmp $0x1,%eax  <+181>: jne 0xbc3 <main+336>  <+187>: lea -0x10(%rbp),%rax  <+191>: mov %rax,%rsi  <+194>: lea 0x201604(%rip),%rdi # 0x202140 <\_ZSt3cin@@GLIBCXX\_3.4>  <+201>: callq 0x870 <\_ZNSirsERi@plt>  <+206>: mov %rax,%rdx  <+209>: lea -0xc(%rbp),%rax  <+213>: mov %rax,%rsi  <+216>: mov %rdx,%rdi  <+219>: callq 0x870 <\_ZNSirsERi@plt>  <+224>: mov -0x10(%rbp),%eax  <+227>: mov %eax,%esi  <+229>: lea 0x2014c1(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  <+236>: callq 0x8d0 <\_ZNSolsEi@plt>  <+241>: lea 0x327(%rip),%rsi # 0xe92  <+248>: mov %rax,%rdi  <+251>: callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  <+256>: mov %rax,%rdx  <+259>: mov -0xc(%rbp),%eax  <+262>: mov %eax,%esi  <+264>: mov %rdx,%rdi  <+267>: callq 0x8d0 <\_ZNSolsEi@plt>  <+272>: lea 0x30c(%rip),%rsi # 0xe96  <+279>: mov %rax,%rdi  <+282>: callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  <+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 0x8d0 <\_ZNSolsEi@plt>  <+310>: mov %rax,%rdx  <+313>: mov 0x20141d(%rip),%rax # 0x201fd0  <+320>: mov %rax,%rsi  <+323>: mov %rdx,%rdi  <+326>: callq 0x8a0 <\_ZNSolsEPFRSoS\_E@plt>  <+331>: jmpq 0xa91 <main+30>  <+336>: mov -0x14(%rbp),%eax | /\*  \* Takes 2 numbers from the user and returns the sum of those two numbers  \* Ex: if num1 is 1 and num2 is 2 then the function returns 3  \* 1 + 2 = 3  \*/  long long add(int num1, int num2) {  long long result = 0; // Initialized variable for all ranges of the return of int + int  result += num1; // Adds first number to result which ensures no loss of data from conversion  result += num2; // Adds second number to result which ensures no loss of data from conversion  return result; // Return result of arithmatic  } |
| <+339>: cmp $0x2,%eax  <+342>: jne 0xc62 <main+495>  <+348>: lea -0x10(%rbp),%rax  <+352>: mov %rax,%rsi  <+355>: lea 0x201563(%rip),%rdi # 0x202140 <\_ZSt3cin@@GLIBCXX\_3.4>  <+362>: callq 0x870 <\_ZNSirsERi@plt>  <+367>: mov %rax,%rdx  <+370>: lea -0xc(%rbp),%rax  <+374>: mov %rax,%rsi  <+377>: mov %rdx,%rdi  <+380>: callq 0x870 <\_ZNSirsERi@plt>  <+385>: mov -0x10(%rbp),%eax  <+388>: mov %eax,%esi  <+390>: lea 0x201420(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  <+397>: callq 0x8d0 <\_ZNSolsEi@plt> <+402>: lea 0x286(%rip),%rsi # 0xe92  <+409>: mov %rax,%rdi  <+412>: callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  <+417>: mov %rax,%rdx  <+420>: mov -0xc(%rbp),%eax  <+423>: mov %eax,%esi  <+425>: mov %rdx,%rdi  <+428>: callq 0x8d0 <\_ZNSolsEi@plt>  <+433>: lea 0x26b(%rip),%rsi # 0xe96  <+440>: mov %rax,%rdi  <+443>: callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  <+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 0x8d0 <\_ZNSolsEi@plt>  <+469>: mov %rax,%rdx  <+472>: mov 0x20137e(%rip),%rax # 0x201fd0  <+479>: mov %rax,%rsi  <+482>: mov %rdx,%rdi  <+485>: callq 0x8a0 <\_ZNSolsEPFRSoS\_E@plt>  <+490>: jmpq 0xa91 <main+30> | /\*  \* Takes 2 numbers from the user and returns the difference of those two numbers  \* Ex: if num1 is 1 and num2 is 2 then the function returns -1  \* 1 - 2 = -1  \*/  long long subtract(int num1, int num2) {  long long result = 0; // Initialized variable for all ranges of the return of int - int  // Subtracts first number to result which ensures no loss of data from conversion  // (set to num1 - result to avoid -(-) turing into +)  result = num1 - result;  result -= num2; // Subtracts second number to result which ensures no loss of data from conversion  return result; // Return result of arithmatic  } |
| <+495>: mov -0x14(%rbp),%eax  <+498>: cmp $0x3,%eax  <+501>: jne 0xa91 <main+30>  <+507>: lea -0x10(%rbp),%rax  <+511>: mov %rax,%rsi  <+514>: lea 0x2014c4(%rip),%rdi # 0x202140 <\_ZSt3cin@@GLIBCXX\_3.4>  <+521>: callq 0x870 <\_ZNSirsERi@plt>  <+526>: mov %rax,%rdx  <+529>: lea -0xc(%rbp),%rax  <+533>: mov %rax,%rsi  <+536>: mov %rdx,%rdi  <+539>: callq 0x870 <\_ZNSirsERi@plt>  <+544>: mov -0x10(%rbp),%eax  <+547>: mov %eax,%esi  <+549>: lea 0x201381(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  <+556>: callq 0x8d0 <\_ZNSolsEi@plt>  <+561>: lea 0x1e7(%rip),%rsi # 0xe92  <+568>: mov %rax,%rdi  <+571>: callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  <+576>: mov %rax,%rdx  <+579>: mov -0xc(%rbp),%eax  <+582>: mov %eax,%esi  <+584>: mov %rdx,%rdi  <+587>: callq 0x8d0 <\_ZNSolsEi@plt>  <+592>: lea 0x1cc(%rip),%rsi # 0xe96  <+599>: mov %rax,%rdi  <+602>: callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  <+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 0x8d0 <\_ZNSolsEi@plt>  <+629>: mov %rax,%rdx  <+632>: mov 0x2012de(%rip),%rax # 0x201fd0  <+639>: mov %rax,%rsi  <+642>: mov %rdx,%rdi  <+645>: callq 0x8a0 <\_ZNSolsEPFRSoS\_E@plt>  <+650>: jmpq 0xa91 <main+30> | /\*  \* Takes 2 numbers from the user and returns the multiplied result of those two numbers  \* Ex: if num1 is 1 and num2 is 2 then the function returns 2  \* 1 \* 2 = 2  \*/  long long multiply(int num1, int num2) {  long long result = 1; // Initialized variable for all ranges of the return of int \* int (to 1 for multiplication)  result \*= num1; // Multiplies first number to result which ensures no loss of data from conversion  result \*= num2; // Multiplies second number to result which ensures no loss of data from conversion  return result; // Return result of arithmatic  } |
| <+655>: mov $0x0,%eax  <+660>: mov -0x8(%rbp),%rcx  <+664>: xor %fs:0x28,%rcx  <+673>: je 0xd1b <main+680>  <+675>: callq 0x8b0 <\_\_stack\_chk\_fail@plt>  <+680>: leaveq  <+681>: retq |  |