# **CS 410 Binary to Assembly Activity Template**

**Step 1:** List the binary file name.

**Step 2:** Identify the functions in the binary file.

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

**Step 4:** Align the blocks of assembly code with their corresponding function in the binary file.

**Step 5:** Explain the functionality of the blocks of assembly code.

## File One: assignment3\_1.o

| **Functions** | **Blocks of Assembly Code** | **Explanation of Functionality** |
| --- | --- | --- |
| main | 0x000000000040057d <+0>: push %rbp  0x000000000040057e <+1>: mov %rsp,%rbp  0x0000000000400581 <+4>: mov $0x400634,%edi  0x0000000000400586 <+9>: callq 0x400450 <puts@plt>  0x000000000040058b <+14>: mov $0x400648,%edi  0x0000000000400590 <+19>: callq 0x400450 <puts@plt>  0x0000000000400595 <+24>: mov $0x40065c,%edi  0x000000000040059a <+29>: callq 0x400450 <puts@plt>  0x000000000040059f <+34>: mov $0x0,%edi  0x00000000004005a4 <+39>: callq 0x400480 <exit@plt> | This function runs cout 3 times. By running the program we can see what prints out and recreate the program.  Int main() {  cout << “Ship to: John Smith” << endl;  cout << “123 Los Angeles Rd.” << endl;  cout << “Los Angeles, CA 90025” << endl;  return 0;  } |
|  |  |  |
|  |  |  |
|  |  |  |

## File Two: assignment3\_2.o

| **Functions** | **Blocks of Assembly Code** | **Explanation of Functionality** |
| --- | --- | --- |
| main | 0x000000000040062d <+0>: push %rbp  0x000000000040062e <+1>: mov %rsp,%rbp  0x0000000000400631 <+4>: sub $0x20,%rsp  0x0000000000400635 <+8>: mov %fs:0x28,%rax  0x000000000040063e <+17>: mov %rax,-0x8(%rbp)  0x0000000000400642 <+21>: xor %eax,%eax  0x0000000000400644 <+23>: mov $0x400714,%edi  0x0000000000400649 <+28>: callq 0x4004e0 <puts@plt>  0x000000000040064e <+33>: lea -0x20(%rbp),%rax  0x0000000000400652 <+37>: mov %rax,%rsi  0x0000000000400655 <+40>: mov $0x40072b,%edi  0x000000000040065a <+45>: mov $0x0,%eax  0x000000000040065f <+50>: callq 0x400520 <\_\_isoc99\_scanf@plt>  0x0000000000400664 <+55>: lea -0x20(%rbp),%rax  0x0000000000400668 <+59>: mov %rax,%rsi  0x000000000040066b <+62>: mov $0x40072e,%edi  0x0000000000400670 <+67>: mov $0x0,%eax  0x0000000000400675 <+72>: callq 0x4004f0 <printf@plt>  0x000000000040067a <+77>: mov $0x0,%edi  0x000000000040067f <+82>: callq 0x400530 <exit@plt> | This function has a cout, followed by a cin, followed by another cout. Once again by running the program we can recreate the program.  int main() {  string name; // variable name can be anything  cout << "Please enter your name" << endl;  cin >> name;  cout << "Welcome Mr. " << name << endl;  return 0;  } |
|  |  |  |
|  |  |  |
|  |  |  |

## File Three: assignment3\_3.o

| **Functions** | **Blocks of Assembly Code** | **Explanation of Functionality** |
| --- | --- | --- |
| AddNumbers | 0x000000000040062d <+0>: push %rbp  0x000000000040062e <+1>: mov %rsp,%rbp  0x0000000000400631 <+4>: mov %edi,-0x4(%rbp)  0x0000000000400634 <+7>: mov %esi,-0x8(%rbp)  0x0000000000400637 <+10>: mov -0x8(%rbp),%eax  0x000000000040063a <+13>: mov -0x4(%rbp),%edx  0x000000000040063d <+16>: add %edx,%eax  0x000000000040063f <+18>: pop %rbp  0x0000000000400640 <+19>: retq | AddNumbers takes 2 numbers and adds them together, and returns the sum.  int AddNumbers(int a, int b) {  return a + b;  }  Variable a and b could be named anything and represent -4(%rbp) and -8(%rbp) |
| main | 0x0000000000400641 <+0>: push %rbp  0x0000000000400642 <+1>: mov %rsp,%rbp  0x0000000000400645 <+4>: sub $0x10,%rsp  0x0000000000400649 <+8>: mov $0x400734,%edi  0x000000000040064e <+13>: callq 0x4004e0 <puts@plt>  0x0000000000400653 <+18>: lea -0x8(%rbp),%rdx  0x0000000000400657 <+22>: lea -0xc(%rbp),%rax  0x000000000040065b <+26>: mov %rax,%rsi  0x000000000040065e <+29>: mov $0x400747,%edi  0x0000000000400663 <+34>: mov $0x0,%eax  0x0000000000400668 <+39>: callq 0x400520 <\_\_isoc99\_scanf@plt>  0x000000000040066d <+44>: mov -0x8(%rbp),%edx  0x0000000000400670 <+47>: mov -0xc(%rbp),%eax  0x0000000000400673 <+50>: mov %edx,%esi  0x0000000000400675 <+52>: mov %eax,%edi  0x0000000000400677 <+54>: callq 0x40062d <AddNumbers>  0x000000000040067c <+59>: mov %eax,-0x4(%rbp)  0x000000000040067f <+62>: mov -0x8(%rbp),%edx  0x0000000000400682 <+65>: mov -0xc(%rbp),%eax  0x0000000000400685 <+68>: mov -0x4(%rbp),%ecx  0x0000000000400688 <+71>: mov %eax,%esi  0x000000000040068a <+73>: mov $0x40074d,%edi  0x000000000040068f <+78>: mov $0x0,%eax  0x0000000000400694 <+83>: callq 0x4004f0 <printf@plt>  0x0000000000400699 <+88>: mov $0x0,%edi  0x000000000040069e <+93>: callq 0x400530 <exit@plt> | main first calls cout for a sting, and then uses cin to input 2 numbers. Then there is a call to AddNumbers and the result is saved in -4(%rbp). There is then another call the cout. After running the program we can recreate the main function like this:  int main() {  int num1, num2, sum;  cout << "Enter first number: ";  cin >> num1;  cout << "Enter second number: ";  cin >> num2;  // Call AddNumbers function to get the sum  sum = AddNumbers(num1, num2);  std::cout << num1 << " + " << num2 << " = " << sum << std::endl;  return 0;  } |
|  |  |  |
|  |  |  |

## File Four: assignment3\_4.o

| **Functions** | **Blocks of Assembly Code** | **Explanation of Functionality** |
| --- | --- | --- |
| PrintFact | 0x000000000040062d <+0>: push %rbp  0x000000000040062e <+1>: mov %rsp,%rbp  0x0000000000400631 <+4>: sub $0x20,%rsp  0x0000000000400635 <+8>: mov %edi,-0x14(%rbp)  0x0000000000400638 <+11>: movl $0x1,-0x4(%rbp)  0x000000000040063f <+18>: mov -0x14(%rbp),%eax  0x0000000000400642 <+21>: mov %eax,-0x8(%rbp)  0x0000000000400645 <+24>: jmp 0x400669 <PrintFact+60>  0x0000000000400647 <+26>: mov -0x4(%rbp),%eax  0x000000000040064a <+29>: imul -0x8(%rbp),%eax  0x000000000040064e <+33>: mov %eax,-0x4(%rbp)  0x0000000000400651 <+36>: mov -0x8(%rbp),%eax  0x0000000000400654 <+39>: mov %eax,%esi  0x0000000000400656 <+41>: mov $0x400844,%edi  0x000000000040065b <+46>: mov $0x0,%eax | This fuction gets an argument of a number that the user entered in the main function. It sets -4(%rbp) to 1 (we’ll call it result) and sets -8(%rbp) to the value passed to the function (we’ll call this i for iterator). Next it uses a loop to multiply the result by i and save it to result. Then there is a call to cout to output i and then subtract 1 from i. It will continue to do this until i is <= 0. Then it calls cout again and outputs the result. From running the program we can figure out the format of the cout statements. |
| PrintFact (cont.) | 0x0000000000400660 <+51>: callq 0x4004f0 <printf@plt>  0x0000000000400665 <+56>: subl $0x1,-0x8(%rbp)  0x0000000000400669 <+60>: cmpl $0x0,-0x8(%rbp)  0x000000000040066d <+64>: jg 0x400647 <PrintFact+26>  0x000000000040066f <+66>: mov -0x4(%rbp),%eax  0x0000000000400672 <+69>: mov %eax,%esi  0x0000000000400674 <+71>: mov $0x400848,%edi  0x0000000000400679 <+76>: mov $0x0,%eax  0x000000000040067e <+81>: callq 0x4004f0 <printf@plt>  0x0000000000400683 <+86>: mov -0x4(%rbp),%eax  0x0000000000400686 <+89>: leaveq  0x0000000000400687 <+90>: retq | void PrintFact(int number) {  int result = 1;  for (int i = number; i > 0; --i) {  result \*= i;  cout << i << " ";  }  cout << " [" << result << "]" << endl;  } |
| PrintSum | 0x0000000000400688 <+0>: push %rbp  0x0000000000400689 <+1>: mov %rsp,%rbp  0x000000000040068c <+4>: sub $0x20,%rsp  0x0000000000400690 <+8>: mov %edi,-0x14(%rbp)  0x0000000000400693 <+11>: movl $0x0,-0x4(%rbp)  0x000000000040069a <+18>: mov -0x14(%rbp),%eax  0x000000000040069d <+21>: mov %eax,-0x8(%rbp)  0x00000000004006a0 <+24>: jmp 0x4006c0 <PrintSum+56>  0x00000000004006a2 <+26>: mov -0x8(%rbp),%eax  0x00000000004006a5 <+29>: add %eax,-0x4(%rbp)  0x00000000004006a8 <+32>: mov -0x8(%rbp),%eax  0x00000000004006ab <+35>: mov %eax,%esi  0x00000000004006ad <+37>: mov $0x400844,%edi  0x00000000004006b2 <+42>: mov $0x0,%eax  0x00000000004006b7 <+47>: callq 0x4004f0 <printf@plt>  0x00000000004006bc <+52>: subl $0x1,-0x8(%rbp)  0x00000000004006c0 <+56>: cmpl $0x0,-0x8(%rbp)  0x00000000004006c4 <+60>: jg 0x4006a2 <PrintSum+26>  0x00000000004006c6 <+62>: mov -0x4(%rbp),%eax  0x00000000004006c9 <+65>: mov %eax,%esi  0x00000000004006cb <+67>: mov $0x400848,%edi  0x00000000004006d0 <+72>: mov $0x0,%eax  0x00000000004006d5 <+77>: callq 0x4004f0 <printf@plt>  0x00000000004006da <+82>: mov -0x4(%rbp),%eax  0x00000000004006dd <+85>: leaveq  0x00000000004006de <+86>: retq | This function is very similar to the PrintFact function. This fuction gets an argument of a number that the user entered in the main function. It sets -4(%rbp) to 0 (we’ll call it result) and sets -8(%rbp) to the value passed to the function (we’ll call this i for iterator). Next it uses a loop to add I to result and save it to result. Then there is a call to cout to output i and then subtract 1 from i. It will continue to do this until i is <= 0. Then it calls cout again and outputs the result. From running the program we can figure out the format of the cout statements.  void PrintSum(int number) {  int result = 0;  for (int i = number; i > 0; --i) {  result += i;  cout << i << " ";  }  cout << " [" << result << "]" << endl;  } |
| DisplayMenu | 0x00000000004006df <+0>: push %rbp  0x00000000004006e0 <+1>: mov %rsp,%rbp  0x00000000004006e3 <+4>: mov $0x400851,%edi  0x00000000004006e8 <+9>: callq 0x4004e0 <puts@plt>  0x00000000004006ed <+14>: mov $0x400864,%edi  0x00000000004006f2 <+19>: callq 0x4004e0 <puts@plt>  0x00000000004006f7 <+24>: mov $0x400871,%edi  0x00000000004006fc <+29>: callq 0x4004e0 <puts@plt>  0x0000000000400701 <+34>: mov $0x40087e,%edi  0x0000000000400706 <+39>: callq 0x4004e0 <puts@plt>  0x000000000040070b <+44>: mov $0x400851,%edi  0x0000000000400710 <+49>: callq 0x4004e0 <puts@plt>  0x0000000000400715 <+54>: pop %rbp  0x0000000000400716 <+55>: retq | Display menu just has 5 cout calls.  void DisplayMenu() {  cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;  cout << "1. Factorial" << endl;  cout << "2. Summation" << endl;  cout << "3. Quit" << endl;  cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;  } |
| main | 0x0000000000400717 <+0>: push %rbp  0x0000000000400718 <+1>: mov %rsp,%rbp  0x000000000040071b <+4>: sub $0x10,%rsp  0x000000000040071f <+8>: movl $0x0,-0x8(%rbp)  0x0000000000400726 <+15>: jmp 0x4007a0 <main+137>  0x0000000000400728 <+17>: mov $0x0,%eax  0x000000000040072d <+22>: callq 0x4006df <DisplayMenu>  0x0000000000400732 <+27>: mov $0x400886,%edi  0x0000000000400737 <+32>: callq 0x4004e0 <puts@plt>  0x000000000040073c <+37>: lea -0x8(%rbp),%rax  0x0000000000400740 <+41>: mov %rax,%rsi  0x0000000000400743 <+44>: mov $0x400899,%edi  0x0000000000400748 <+49>: mov $0x0,%eax  0x000000000040074d <+54>: callq 0x400520 <\_\_isoc99\_scanf@plt>  0x0000000000400752 <+59>: mov -0x8(%rbp),%eax  0x0000000000400755 <+62>: cmp $0x3,%eax  0x0000000000400758 <+65>: je 0x40077a <main+99>  0x000000000040075a <+67>: mov $0x40089c,%edi  0x000000000040075f <+72>: callq 0x4004e0 <puts@plt>  0x0000000000400764 <+77>: lea -0x4(%rbp),%rax  0x0000000000400768 <+81>: mov %rax,%rsi  0x000000000040076b <+84>: mov $0x400899,%edi  0x0000000000400770 <+89>: mov $0x0,%eax  0x0000000000400775 <+94>: callq 0x400520 <\_\_isoc99\_scanf@plt> | The main function sets choice -8(%rbp) to 0 and then starts a while loop for while choice is not 3. Then it calls DisplayMenu and calls cout to output “Enter your number:”. It then calls cin to get a number and stores that in choice. If it is not equal to 3 there is a call to cout to output “Enter a number:”. Then there is a call to cin and saves that in -4(%rbp) which we’ll call number. If choice is equal to 1 it calls PrintFact and sends number. If choice is equal to 2 is calls PrintSum and sends number. If choice is 3 the loop is exited. C++ code would look something like this:  int main() {  int choice = 0;  int number;  while (choice != 3) {  DisplayMenu();  cout << “Enter your number:” << endl;  cin >> choice;  if (choice != 3) {  cout << “Enter a number:” << endl;  cin >> number;  if (choice == 1) {  PrintFact(number);  }  if (choice == 2) {  PrintSum(number);  }  }  }  } |
| main (cont.) | 0x000000000040077a <+99>: mov -0x8(%rbp),%eax  0x000000000040077d <+102>: cmp $0x1,%eax  0x0000000000400780 <+105>: jne 0x40078e <main+119>  0x0000000000400782 <+107>: mov -0x4(%rbp),%eax  0x0000000000400785 <+110>: mov %eax,%edi  0x0000000000400787 <+112>: callq 0x40062d <PrintFact>  0x000000000040078c <+117>: jmp 0x4007a0 <main+137>  0x000000000040078e <+119>: mov -0x8(%rbp),%eax  0x0000000000400791 <+122>: cmp $0x2,%eax  0x0000000000400794 <+125>: jne 0x4007a0 <main+137>  0x0000000000400796 <+127>: mov -0x4(%rbp),%eax  0x0000000000400799 <+130>: mov %eax,%edi  0x000000000040079b <+132>: callq 0x400688 <PrintSum>  0x00000000004007a0 <+137>: mov -0x8(%rbp),%eax  0x00000000004007a3 <+140>: cmp $0x3,%eax  0x00000000004007a6 <+143>: jne 0x400728 <main+17>  0x00000000004007a8 <+145>: mov $0x0,%edi  0x00000000004007ad <+150>: callq 0x400530 <exit@plt> |  |