# **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** |
| --- | --- | --- |
| int main(){ |  |  |
| cout << "Ship to: John Smith" << endl; | mov $0x400634,%edi  callq 0x400450 <puts@plt> | moves the array of characters into the memory and calls the cout function |
| cout << "123 Los Angeles Rd." << endl; | mov $0x400648,%edi  callq 0x400450 <puts@plt> | moves the array of characters into the memory and calls the cout function |
| cout << "Los Angeles, CA 90025" << endl; | mov $0x40065c,%edi  callq 0x400450 <puts@plt> | moves the array of characters into the memory and calls the cout function |
| return 0; | mov $0x0,%edi  callq 0x400480 <exit@plt> | moves 0 into the memory location and ends the main function |
| } |  |  |

## File Two: assignment3\_2.o

| **Functions** | **Blocks of Assembly Code** | **Explanation of Functionality** |
| --- | --- | --- |
| int main(){ |  |  |
| char userName[50]; | mov %fs:0x28,%rax  mov %rax,-0x8(%rbp) | Initializes variable for userName |
| cout << “Please enter your name” << endl; | mov $0x400714,%edi  callq 0x4004e0 <puts@plt>  lea -0x20(%rbp),%rax | Moves memory location of array of characters and outputs message to user |
| cin >> userName; | mov %rax,%rsi  mov $0x40072b,%edi  mov $0x0,%eax  callq 0x400520 <\_\_isoc99\_scanf@plt>  lea -0x20(%rbp),%rax | Moves memory to allow for the input of userName and inputs the data given by user into that variable |
| cout << Welcome Mr. << userName << endl; | mov %rax,%rsi  mov $0x40072e,%edi  mov $0x0,%eax  callq 0x4004f0 <printf@plt> | Moves memory location of arrach of characters, moves memory location of data given by user, and outputs the welcome message to the user |
| return 0 | mov $0x0,%edi  callq 0x400530 <exit@plt> | moves 0 into the memory location and ends the main function |
| } |  |  |

## File Three: assignment3\_3.o

| **Functions** | **Blocks of Assembly Code** | **Explanation of Functionality** |
| --- | --- | --- |
| int addNumbers(int num1, int num2) {  return num1 + num2;  } |  | Takes 2 numbers as input and returns their sum |
| int main() {  int num1, num2 = 0;  cout << "Enter two numbers:" << endl;  cin >> num1 >> num2;    cout << num1 << " + " << num2 << " = " << addNumbers(num1, num2) << endl;  return 0;  } | push %rbp  mov %rsp,%rbp  sub $0x10,%rsp  mov $0x400734,%edi  callq 0x4004e0 <puts@plt>  lea -0x8(%rbp),%rdx  lea -0xc(%rbp),%rax  mov %rax,%rsi  mov $0x400747,%edi  mov $0x0,%eax  callq 0x400520 <\_\_isoc99\_scanf@plt>  mov -0x8(%rbp),%edx  mov -0xc(%rbp),%eax  mov %edx,%esi  mov %eax,%edi  callq 0x40062d <AddNumbers>  mov %eax,-0x4(%rbp)  mov -0x8(%rbp),%edx  mov -0xc(%rbp),%eax  mov -0x4(%rbp),%ecx  mov %eax,%esi  mov $0x40074d,%edi  mov $0x0,%eax  callq 0x4004f0 <printf@plt>  mov $0x0,%edi  callq 0x400530 <exit@plt> | push rbp to the top of the stack  move rsp into rbp  subtract $0x10 from rsp  move $0x400734 into edi  call puts function (cout)  load rdx into -0x8(%rbp)  load rax into -0xc(%rbp)  move rax into rsi  move $0x400747 into edi  move 0 into eax  call scanf function (cin)  move -0x8(%rbp) into edx  move -0xc(%rbp) into eax  move edx into esi  move eax into edi  call AddNumber function  move eax into 0x4(%rbp)  move -0x8(%rbp) into edx  move -0xc(%rbp) into eax  move -0x4(%rbp) into ecx  move eax into esi  move $0x40074d into edi  move 0 into eax  call puts function  move 0 into edi  call exit function |
|  |  |  |
|  |  |  |

## File Four: assignment3\_4.o

| **Functions** | **Blocks of Assembly Code** | **Explanation of Functionality** |
| --- | --- | --- |
| void printFact(int num) {  int factorial = 1;    for (int i = 0; i < num; ++i) {  cout << num-i << " ";  factorial \*= num - i;  }  cout << " [" << factorial << "]" << endl;  } |  | takes num as input,  creates local variable factorial for storing factorial result and initializes it to 1 for the multiplication  uses for loop that loops num number of times  during each loop factorial is multiplies by the num – the current loop number and prints the num – the loop number so the number counts down from the given number to 1  after the loop the result is printed to the console after 4 spaces (3 by the cout statement and 1 by the loop print) and surrounded by square brackets |
| void printSum(int num) {  int sum = 0;    for (int i = 0; i < num; ++i) {  cout << num - i << " ";  sum += num - i;  }  cout << " [" << sum << "]" << endl;  } |  | takes num as input,  creates local variable sum for storing summation result and initializes it to 0 for the addition  uses for loop that loops num number of times  during each loop num – the current loop number is added to sum and prints the num – the loop number so the number counts down from the given number to 1  after the loop the result is printed to the console after 4 spaces (3 by the cout statement and 1 by the loop print) and surrounded by square brackets |
| void displayMenu(){  cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;  cout << "1. Factorial" << endl;  cout << "2. Summation" << endl;  cout << "3. Quit" << endl;  cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;  cout << "Enter your number:" << endl;  } |  | The display menu prints  "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"  1. Factorial  2. Summation  3. Quit  "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"  to the console whenever it is called |
| int main() {  int menuSelection = 0;  int userNum = 0;    while (menuSelection != 3) {  displayMenu();  cin >> menuSelection;    if (menuSelection == 3) {  return 0;  }    cout << "Enter a number:" << endl;  cin >> userNum;    if (menuSelection == 1) {  printFact(userNum);  }  else if (menuSelection == 2) {  printSum(userNum);  }  }  return 0;  } | movl $0x0,-0x8(%rbp)  jmp 0x4007a0 <main+137>  mov $0x0,%eax  callq 0x4006df <DisplayMenu>  mov $0x400886,%edi  callq 0x4004e0 <puts@plt>  lea -0x8(%rbp),%rax  mov %rax,%rsi  mov $0x400899,%edi  mov $0x0,%eax  callq 0x400520 <\_\_isoc99\_scanf@plt>  mov -0x8(%rbp),%eax  cmp $0x3,%eax  je 0x40077a <main+99>  mov $0x40089c,%edi  callq 0x4004e0 <puts@plt>  lea -0x4(%rbp),%rax  mov %rax,%rsi  mov $0x400899,%edi  mov $0x0,%eax  callq 0x400520 <\_\_isoc99\_scanf@plt>  mov -0x8(%rbp),%eax  cmp $0x1,%eax  jne 0x40078e <main+119>  mov -0x4(%rbp),%eax  mov %eax,%edi  callq 0x40062d <PrintFact>  jmp 0x4007a0 <main+137>  mov -0x8(%rbp),%eax  cmp $0x2,%eax  jne 0x4007a0 <main+137>  mov -0x4(%rbp),%eax  mov %eax,%edi  callq 0x400688 <PrintSum>  mov -0x8(%rbp),%eax  cmp $0x3,%eax  jne 0x400728 <main+17>  mov $0x0,%edi  callq 0x400530 <exit@plt> | moves 0 into rbp  load IP with 0x4007a0  move 0 into eax  call function DisplayMenu  move 0x400886 into edi  call puts function  load rax into -0x8(%rbp)  move rax into rsi  move $0x400899 into edi  mov 0 into eax  call scanf function  move -0x8(%rbp) into eax  compare 3 to eax  jump to 0x40077a if true  move $0x40089c into edi  call puts function  loads rax into -0x4(%rbp)  moves rax into rsi  moves $0x400899 into edi  moves 0 into eax  calls scanf function  moves eax into -0x8(%rbp)  compares eax to 1  jumps to 0x40078e if true  moves eax into -0x4(%rbp)  moves edi to eax  calls PrintFact function  jump to 0x4007a0  move eax into -0x8(%rbp)  compares eax to 2  jumps to 0x4007a0 if true  moves eax into -0x4(%rbp)  moves edi into eax  calls PrintSum function  moves eax into -0x8(%rbp)  compares 3 to eax  jumps to 0x400728 if ture  moves 0 into edi  calls Exit function |