# **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 | push %rbp  mov %rsp,%rbp  mov $0x400634,%edi  callq 0x400450 <puts@plt>  mov $0x400648,%edi  callq 0x400450 <puts@plt>  mov $0x40065c,%edi  callq 0x400450 <puts@plt>  mov $0x0,%edi  callq 0x400480 <exit@plt> | The first two lines set up the stack frame. Next the function stores a value to %edi and then calls the puts function to print it. It does this three times for each string. The three strings print out as,  Ship to: John Smith  123 Los Angeles Rd.  Los Angeles, CA 90025 |

## File Two: assignment3\_2.o

| **Functions** | **Blocks of Assembly Code** | **Explanation of Functionality** |
| --- | --- | --- |
| main | push %rbp  mov %rsp,%rbp  sub $0x20,%rsp  mov %fs:0x28,%rax  mov %rax,-0x8(%rbp)  xor %eax,%eax  mov $0x400714,%edi  callq 0x4004e0 <puts@plt>  lea -0x20(%rbp),%rax  mov %rax,%rsi  mov $0x40072b,%edi  mov $0x0,%eax  callq 0x400520 <\_\_isoc99\_scanf@plt>  lea -0x20(%rbp),%rax  mov %rax,%rsi  mov $0x40072e,%edi  mov $0x0,%eax  callq 0x4004f0 <printf@plt>  mov $0x0,%edi  callq 0x400530 <exit@plt> | The first two lines set up the stack frame. Sub and move are done to shift around values in the registers. Xor clears out eax. Mov moves a string that is outputed using puts function. More movement is done including another string. Scanf is called to take user input. Further movement is done and then printf is performed and edi is cleared out. This results in a programs that asks for you name, takes input, and then prints out “Welcome Mr. “ [name]. |

## File Three: assignment3\_3.o

| **Functions** | **Blocks of Assembly Code** | **Explanation of Functionality** |
| --- | --- | --- |
| main | 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> | The stack is initialized. The string at 0x400734 is outputed. Two values are stored using the function call to scanf. The AddNumbers function is called using these values and adds them together. The values are moved around on the stack and the string at 0x40074d is loaded. Printf is called then to display the two numbers inputed as an addition equation with the result.  Enter two numbers:  1  2  1 + 2 = 3 |
| AddNumbers | push %rbp  mov %rsp,%rbp  mov %edi,-0x4(%rbp)  mov %esi,-0x8(%rbp)  mov -0x8(%rbp),%eax  mov -0x4(%rbp),%edx  add %edx,%eax  pop %rbp  retq | The stack is initilized. The two values passed to the function are stored on the stack and moved to eax and edx. They are then added to each other and the value is returned. |

## File Four: assignment3\_4.o

| **Functions** | **Blocks of Assembly Code** | **Explanation of Functionality** |
| --- | --- | --- |
| PrintFact | push %rbp  mov %rsp,%rbp  sub $0x20,%rsp  mov %edi,-0x14(%rbp)  movl $0x1,-0x4(%rbp)  mov -0x14(%rbp),%eax  mov %eax,-0x8(%rbp)  jmp 0x400669 <PrintFact+60>  mov -0x4(%rbp),%eax  imul -0x8(%rbp),%eax  mov %eax,-0x4(%rbp)  mov -0x8(%rbp),%eax  mov %eax,%esi  mov $0x400844,%edi  mov $0x0,%eax  callq 0x4004f0 <printf@plt>  subl $0x1,-0x8(%rbp)  cmpl $0x0,-0x8(%rbp)  jg 0x400647 <PrintFact+26>  mov -0x4(%rbp),%eax  mov %eax,%esi  mov $0x400848,%edi  mov $0x0,%eax  callq 0x4004f0 <printf@plt>  mov -0x4(%rbp),%eax  leaveq  retq | The stack is initialized. The value passed to the function is stored and moved to a usable register. The number inputed is printed and decremented by 1. This loops until the value equals 1 where it stops. It then prints the facorial total in brackets and returns. |
| PrintSum | push %rbp  mov %rsp,%rbp  sub $0x20,%rsp  mov %edi,-0x14(%rbp)  movl $0x0,-0x4(%rbp)  mov -0x14(%rbp),%eax  mov %eax,-0x8(%rbp)  jmp 0x4006c0 <PrintSum+56>  mov -0x8(%rbp),%eax  add %eax,-0x4(%rbp)  mov -0x8(%rbp),%eax  mov %eax,%esi  mov $0x400844,%edi  mov $0x0,%eax  callq 0x4004f0 <printf@plt>  subl $0x1,-0x8(%rbp)  cmpl $0x0,-0x8(%rbp)  jg 0x4006a2 <PrintSum+26>  mov -0x4(%rbp),%eax  mov %eax,%esi  mov $0x400848,%edi  mov $0x0,%eax  callq 0x4004f0 <printf@plt>  mov -0x4(%rbp),%eax  leaveq  retq | The stack is initialized. Like the print factorial function a number is passed to the function for use. It is stored and moved to a usable location. The number is printed to ther console and decremented by 1. It loops until the value equals 1 where it then prints the sum of all of the numbers in brackets. The function then returns. |
| DisplayMenu | push %rbp  mov %rsp,%rbp  mov $0x400851,%edi  callq 0x4004e0 <puts@plt>  mov $0x400864,%edi  callq 0x4004e0 <puts@plt>  mov $0x400871,%edi  callq 0x4004e0 <puts@plt>  mov $0x40087e,%edi  callq 0x4004e0 <puts@plt>  mov $0x400851,%edi  callq 0x4004e0 <puts@plt>  pop %rbp  retq | The stack is initialized. The strings stored in data are moved to edi and outputed using the put function to display a menu. Below is the displayed menu.  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  1. Factorial  2. Summation  3. Quit  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Enter your number: |
| main | push %rbp  mov %rsp,%rbp  sub $0x10,%rsp  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> | The stack is initialized. The menu function is called to display the menu. Scanf is also called to take user input. If 1 is selected then print factorial function is called, if 2 is selected print sum is called, and if 3 is selected the application terminates. |