# **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 stack frame is set up in the first couple lines. A value is stored to %edi and the puts function is called so that it prints it. This is done for each of the three strings. |

## 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> | Stack frame is set up in the first couple lines. Values are shifted around in the register, and this is from sub and move. So, mov will move a string after it is output, and this is from the puts function. Scanf is for getting input from the user and more moves are done and then there is printf. Edi is cleared. This program ask for a name, takes it and then prints out the string with 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> | Stack is et up and the string at 0x400734 is output. A function call to scanf has stored two values. AddNumbers function is then called and it uses the values which adds them. Values are moved around and string 0x40074d gets loaded. Printf displays the addition equation with the two numbers from earlier. |
| 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 two values are stored on the stack which are moved to eax and edx. The sum of the two numbers are then 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 | Value that is passed is stored then moved to register that can be used. The number that was input is printed and is decreased by 1. So, this will keep looping until it has reached 1 and then it will stop.  It will print and then return. |
| 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 | A number is passed to the function for use. Stored then moved to somewhere it can be used. Number is printed and decreased by 1. This loops until it reaches 1 then prints the sum of all numbers. Function 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 | Strings stored in data are moved. Using the put function the strings are displayed making a menu. |
| 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> | Menu function is called so that the menu can be displayed. So that we can take input from the user we call Scanf. If 1 is selected then what is called is print factorial function. If 2 is selected then print sum is called. 3 will terminate. |