# **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\_1.0

| **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> | Pushes register rbp to the stack  Move register rbp into address of register rsp  Move 0x634 to register edi  Call instruction to call first string “Ship to: John Smith“  Move 0x648 to register edi. This refers to the second string  Call instruction to call second string “123 Los Angeles Rd”  Move 0x65c to register edi. This refers to the third string  Call instruction for the third string “Los Angeles, CA 90025”  Move 0x0 to register edi  Call instruction to exit the program |

## File Two: assignment3\_2.0

| **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> | Push rbp to stack  Move rsp to to register rbp  Subtract 20 from value of register rsp  Move fs:0x28 to register rax  Move rax to rbp at 8 bits  Implement or instruction on eax to set operand from second eax register to the first.  Move 0x714 to register edi. This refers to the first string instruction  Call the first string instruction “Please Enter your Name”  Load effective address of rbp at 20 bits to rax  Move rax to register rsi..  Move 0x072b to register edi. This refers to the input gathered from the user.  Move 0x0 to register eax  Call instruction for the to scan for user input  Load effective address of rbp at 20 bits to rax  Move rax to register rsi  Move 0x072e to register edi. This refers to the second string instruction  Move 0x0 to register eax  Call instruction for second string “Hello, Mr. “ This will then out after the second string the user input that was stored.  Move 0x0 to register edi  Call instruction to exit the program |

## File Three: assignment3\_3.0

| **Functions** | **Blocks of Assembly Code** | **Explanation of Functionality** |
| --- | --- | --- |
| 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 | Register rbp is pushed to the stack  Move value of rsp to register rbp  Move edi to register rbp at 8 bits. This gathers our first variable gathered in the main function.  Move esi to rbp at 8 bits. This gathers our second variable gathered in the main function  Move rbp at 8 bits value to register eax.  Move rbp at 4 bits value to register edx.  Add values of register edx and eax.  Pop the value back to the register and return |
| 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> | Push register rbp to the stack  Move rsp to to register rbp  Subtract 10 from register rsp  Move 0x734 to register edi  Call instruction to output the string “Enter two numbers”  Load effective address of rbp at 8 bits to rdx. This will point to the first variable gathered from user input  Load effective address of rbp at 0xc bits to register rax. This will point to the second variabled gathered from user input  Move rax to register rsi  Move 0x747 to register edi This will be the user input of the first number  Move 0x0 to register eax. This will be the second input number  Call the instruction for the cin function to gatehr the user input  Move rbp at 8 bits to register edx  Move rbp at 0xc bits to register eax  Move edx to register esi. This will be used in the AddNumbers function  Move eax to register edi. This will be used in the AddNumbers function  Call instruction to call the AddNumbers function in the main function  Move eax to rbp at 4 bits  Move rbp at 8 bits to edx  Move rbp at 0xc bits to ecx  Move rbp at 4 bits to register ecx  Move eax to register esi  Move 0x074d to register edi  Move 0x0 to register eax  Call instruction to gather output string to show string and value of the AddNumber function call  Move 0x0 to register edi  Exit the program |

## File Four: assignment3\_4.o

| **Functions** | **Blocks of Assembly Code** | **Explanation of Functionality** |
| --- | --- | --- |
| 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 | Push rbp to the stack  Move rsp to register rbp  Move 0x851 to register edi. This refers to the first string instruction  Call instruction to output “\*\*\*\*\*\*\*\*\*\*\*” string.  Move 0x864 to edi. This refers to the second string instruction  Call instruction to output string “1. Factorial”.  Move 0x871 to register edi. This refers to the third string instruction  Call instruction for string “2. Summation”.  Move 0x87e to register edi. This refers to the fourth string instruction.  Call instruction to output the string “3. Quit”.  Move 0x851 to edi. This refers back to another output of the first string.  Call instruction to ouput string “\*\*\*\*\*\*\*\*\*\*”.  Pop data from register rbp. Return statement of function  This provides a display menu function to be called in the main function |
| 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 | Push rbp to the stack  Move rsp to register rbp  Subtractract 20 from the value of register rsp  Move edi to register rbp at 14 bits  Move 1 to register rbp at 4 bits  Move rbp at 14 bits to register eax  Move eax to rbp at 8 bits  Unconditional jump of the program to PrintFact 60 to progress  Move 4 bits of rbp to register eax  Multiply rbp at 8 bits by eax. This rbp at 8 bits will decrement by 1 less each time throughout the multiplication, Ex: 4\*3\*2\*1  Move eax to rbp at 4 bits  Move rbp at 8 bits to eax  Move eax to esi  Move 0x844 to register edi. This refers to the string instruction  Move 0x0 to register eax  Call instruction to output the string  Subtract 1 from rbp at 8 bits as referred to above to get the next variable value .  Compare 0x0 to rbp at bits to see if the value of the variable is 0.  If the value is greater than 0, jump to PrintFact 26, which will then do the multiplication and subtraction again  Move rbp at 4 bits to register eax  Move eax to register esi  Move 0x848 to register edi. This refers to the string instruction used for the output later  Call instruction to print string “[ %d]. the %d refers to the factorial multiplication applied in the previous steps  Move rbp at 4 bits to register eax  Leave function |
| 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 | Push rbp to the stack  Move rsp to register rbp  Subtract 20 from value of register rsp  Move edi to rbp at 14 bits  Move 0x0 to register rbp at 4 bits  Move rbp at 14 bits to register eax  Move eax to register rbp at 8 bits  Unconditional jump of the program to PrintFact 56 to progress  Move rbp at 8 bits to register eax. This is the number that will be entered into the program  Add eax value to the value of rbp at 4 bits  Move rbp at 8 bits to register eax  Move eax to register esi  Move 0x844 to register edi. This will refer to the strign instruction of the value to output  Move 0x0 to register eax  Call print instruction to output the string %d. This will represent the value input by the user  Subtract 1 from the value of rbp at bits  Compare 0x0 to rbp at 8 bits to to see if the number is gretaer than 0  Jump to PrintSum 26 if the value is greater than 0. This will start the addition process all over again.  Move rbp at 4 bits to register eax  Move eax to register esi  Move 0x848 to register edi. This is referring to the string instruction.  Move 0x0 to register eax  Call instruction to output the string [%d]. The %d will refer to the summation of numbers  Move rbp at 4 bits to eax  Leave function |
| 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> | Push rbp to the stack  Move rsp to to register rbp  Subtract 10 from register rsp  Move 0x0 to register rbp at 8 bits  Unconditional jump to progress the program to main 137  Move 0x0 to register eax  Call instruction to call the DisplayMenu function and output the function  Move 0x886 to register edi. This will refer to the specific string instruction  Call instruction to output the string “Enter your number: “  Load effective address of rbp at 8 bits to register rax. This will be the portion where the user will enter a number to the program  Move rax to register rsi  Move 0x899 to register edi. This will refer to the number that was entered by the user  Move 0x0 to register eax  Call instruction to scan for user input number  Move rbp at 8 bits to register eax  Compare the user input number that is stored in eax to 3.  If value is equal to 3, then jump to main 99, which will exit the program  If not equal, Move 0x89c to register edi  Call instruction to output string “Enter a number: “  Load effective address of rbp at 4 bits to register rax  Move rax to register rsi  Move 0x899 to register edi  Move 0x0 to register eax  Call instruction to scan for user input to the string “Enter your number: “ that will be output  Move rbp at 8 bits to register eax  Compare register eax to 1.  If not equal to 1, jump to main 119  Else move rbp at 4 bits to register eax  Move eax to register edi  Call instruction to use the PrintFact function  Unconditional jump back to the main function  Move rbp at 8 bits to register eax  Compare eax to 2.  If not equal to two jump to main 137  Move rbp at 4 bits t eax  Move eax to register edi  Call instruction to utilize the PrintSum function  Move rbp at 8 bits to eax  Compare register eax to 3  If not equal to 3, jump to main 137 in the main function  Move 0x0 to register edi  Call intruction to exit the program if the jne instruction is not used. |