# **CS 410 Binary to Assembly Activity**

**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** |
| --- | --- | --- |
| Dump of assembler code for function main: | |  |
|  | push %rbp | First two instructions: Function prologues. They push the old base pointer onto the stack, which is saved for a later time.  The function stores the value at %edi.  It is then called with the puts() function and prints.  The output is:  Ship to: John Smith  123 Los Angeles Rd.  Los Angeles, CA 90025 |
| 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> |
| End of assembler dump. | |  |

## File Two: assignment3\_2.o

| **Functions** | **Blocks of Assembly Code** | **Explanation of Functionality** |
| --- | --- | --- |
| Dump of assembler code for function main: | |  |
|  | push %rbp | First two instructions: Function prologues. They push the old base pointer onto the stack, which is saved for a later time.  Sub and mov are used to alter the values within the registers.  Xor is used to quickly clear the register within %eax to zero. The put() function called to request user input.  Output: “Please enter your name”  Scanf() function reads data from user input and is stored. Printf () function is called to display. %edi is then cleared to zero.  The output is:  “Welcome Mr.” then prints user input. |
| 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> |
| End of assembler dump. | |  |

## File Three: assignment3\_3.o

| **Functions** | **Blocks of Assembly Code** | **Explanation of Functionality** |
| --- | --- | --- |
| Dump of assembler code for function main: | |  |
|  | push %rbp | First two instructions: Function prologues. They push the old base pointer onto the stack, which is saved for a later time.  Output: “Enter two numbers” at 0x714 and stored in %edi by using the scanf() function.  The AddNumbers() function used to add the variables and stored at %eax and %edx.  Equation: x + y = z  Printf () function used to display results after the variables are added and the value is returned. |
| 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> |
| End of assembler dump. | |  |

## File Four: assignment3\_4.o

| **Functions** | **Blocks of Assembly Code** | **Explanation of Functionality** |
| --- | --- | --- |
| Dump of assembler code for function main: | |  |
|  | push %rbp | First two instructions: Function prologues. They push the old base pointer onto the stack, which is saved for a later time.  Create the variables, the variables are moved to a register that is operational.  The number that is inputted is printed, subtracted by 1 and verified to see if the value is 1 and then stops. This process is for printing the factorial as well as the summation.  The display menu that is printed:  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   1. Factorial 2. Summation 3. Quit   Enter your number:  The scanf() function is called to ask for the user input. The user will type 1 or 2 and the program will print the selected option, or if the user selects option 3, the program will terminate. |
| 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> |
| End of assembler dump. | |  |