Binary to Assembly Activity

CS-410-R4890 Software Reserve Engineering

Raphael Coloma

March 21, 2024

# **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 | 0x000000000040057d <+0>: push %rbp  0x000000000040057e <+1>: mov %rsp,%rbp | Pushes the values of the registers onto the stack and uses the registers to run the logic. |
| 0x0000000000400581 <+4>: mov $0x400634,%edi  0x0000000000400586 <+9>: callq 0x400450 <puts@plt> | Move string to register %edi. Print “Ship to: John Smith” to screen. End with newline. |
| 0x000000000040058b <+14>: mov $0x400648,%edi  0x0000000000400590 <+19>: callq 0x400450 <puts@plt> | Move string to register %edi. Print “123 Los Angeles Rd.” to screen. End with newline. |
| 0x0000000000400595 <+24>: mov $0x40065c,%edi  0x000000000040059a <+29>: callq 0x400450 <puts@plt> | Move string to register %edi. Print “Los Angeles, CA 90025” to screen. End with newline. |
| 0x000000000040059f <+34>: mov $0x0,%edi  0x00000000004005a4 <+39>: callq 0x400480 <exit@plt> | Move value 0 to register %edi. Return value and terminate function. |

## File Two: assignment3\_2.o

| **Functions** | **Blocks of Assembly Code** | **Explanation of Functionality** |
| --- | --- | --- |
| main | 0x000000000040062d <+0>: push %rbp  0x000000000040062e <+1>: mov %rsp,%rbp  0x0000000000400631 <+4>: sub $0x20,%rsp  0x0000000000400635 <+8>: mov %fs:0x28,%rax | Pushes the values of the registers onto the stack and uses the registers to run the logic. |
| 0x000000000040063e <+17>: mov %rax,-0x8(%rbp) | Reserve 8 bytes above base pointer register %rbp by pushing value of register %rax |
| 0x0000000000400642 <+21>: xor %eax,%eax | Set register %eax to 0 by comparing it to itself. |
| 0x0000000000400644 <+23>: mov $0x400714,%edi  0x0000000000400649 <+28>: callq 0x4004e0 <puts@plt> | Move string to register %edi. Print “Please enter your name” to screen. End with newline. |
| 0x000000000040064e <+33>: lea -0x20(%rbp),%rax  0x0000000000400652 <+37>: mov %rax,%rsi  0x0000000000400655 <+40>: mov $0x40072b,%edi  0x000000000040065a <+45>: mov $0x0,%eax  0x000000000040065f <+50>: callq 0x400520 <\_\_isoc99\_scanf@plt> | Place the address of the variable at 20 bytes above register %rbp into register %rax.  Move value in register %rax to register %rsi.  Move address to register %edi. Call cin function to read value from the user. |
| 0x0000000000400664 <+55>: lea -0x20(%rbp),%rax  0x0000000000400668 <+59>: mov %rax,%rsi  0x000000000040066b <+62>: mov $0x40072e,%edi  0x0000000000400670 <+67>: mov $0x0,%eax  0x0000000000400675 <+72>: callq 0x4004f0 <printf@plt> | Place the address of the variable at 20 bytes above register %rbp into register %rax.  Move value in register %rax to register %rsi.  Move address to register %edi. Print “Welcome Mr. “ to screen appended by the user input value and a newline character. |
| 0x000000000040067a <+77>: mov $0x0,%edi  0x000000000040067f <+82>: callq 0x400530 <exit@plt> | Move value 0 to register %edi. Return value and terminate function. |

## File Three: assignment3\_3.o

| **Functions** | **Blocks of Assembly Code** | **Explanation of Functionality** |
| --- | --- | --- |
| AddNumbers | 0x000000000040062d <+0>: push %rbp  0x000000000040062e <+1>: mov %rsp,%rbp | Pushes the values of the registers onto the stack and uses the registers to run the logic. |
| 0x0000000000400631 <+4>: mov %edi,-0x4(%rbp) | Reserve 4 bytes above base pointer register %rbp by pushing value of register %edi |
| 0x0000000000400634 <+7>: mov %esi,-0x8(%rbp) | Reserve 8 bytes above base pointer register %rbp by pushing value of register %esi |
| 0x0000000000400637 <+10>: mov -0x8(%rbp),%eax | Pop value from stack to register %eax |
| 0x000000000040063a <+13>: mov -0x4(%rbp),%edx | Pop value from stack to register %edx |
| 0x000000000040063d <+16>: add %edx,%eax | Add values from registers %edx and %eax and store result in %eax |
| 0x000000000040063f <+18>: pop %rbp  0x0000000000400640 <+19>: retq | Clear stack. Exit function and return value to main. |
| main | 0x0000000000400641 <+0>: push %rbp  0x0000000000400642 <+1>: mov %rsp,%rbp  0x0000000000400645 <+4>: sub $0x10,%rsp | Pushes the values of the registers onto the stack and uses the registers to run the logic. |
| 0x0000000000400649 <+8>: mov $0x400734,%edi  0x000000000040064e <+13>: callq 0x4004e0 <puts@plt> | Move string to register %edi. Print “Enter two numbers:” to screen. End with newline. |
| 0x0000000000400653 <+18>: lea -0x8(%rbp),%rdx  0x0000000000400657 <+22>: lea -0xc(%rbp),%rax  0x000000000040065b <+26>: mov %rax,%rsi  0x000000000040065e <+29>: mov $0x400747,%edi  0x0000000000400663 <+34>: mov $0x0,%eax  0x0000000000400668 <+39>: callq 0x400520 <\_\_isoc99\_scanf@plt> | Place the address of the variable at 8 bytes above register %rbp into register %rdx and at 12 bytes into register %rax  Move value in register %rax to register %rsi.  Move address to register %edi. Call cin function to read two integer values from the user. |
| 0x000000000040066d <+44>: mov -0x8(%rbp),%edx  0x0000000000400670 <+47>: mov -0xc(%rbp),%eax  0x0000000000400673 <+50>: mov %edx,%esi  0x0000000000400675 <+52>: mov %eax,%edi  0x0000000000400677 <+54>: callq 0x40062d <AddNumbers> | Move user input values from register stack %rbp to registers %edx and %eax. Move values from registers %edx and %eax to registers %esi and %edi to be passed to function AddNumbers. |
| 0x000000000040067c <+59>: mov %eax,-0x4(%rbp)  0x000000000040067f <+62>: mov -0x8(%rbp),%edx  0x0000000000400682 <+65>: mov -0xc(%rbp),%eax  0x0000000000400685 <+68>: mov -0x4(%rbp),%ecx  0x0000000000400688 <+71>: mov %eax,%esi  0x000000000040068a <+73>: mov $0x40074d,%edi  0x000000000040068f <+78>: mov $0x0,%eax  0x0000000000400694 <+83>: callq 0x4004f0 <printf@plt> | Pust returned result on register %eax from AddNumbers function to stack 4 bytes above register %rbp. Pop values from register stack %rbp to registers %edx, %eax, and %ecx. Print to screen the user input values separated by a “ + ”. Followed by “ = “ and the result of adding the two input values. |
| 0x0000000000400699 <+88>: mov $0x0,%edi  0x000000000040069e <+93>: callq 0x400530 <exit@plt> | Move value 0 to register %edi. Return value and terminate function. |

## File Four: assignment3\_4.o

| **Functions** | **Blocks of Assembly Code** | **Explanation of Functionality** |
| --- | --- | --- |
| PrintFact | 0x000000000040062d <+0>: push %rbp  0x000000000040062e <+1>: mov %rsp,%rbp  0x0000000000400631 <+4>: sub $0x20,%rsp | Pushes the values of the registers onto the stack and uses the registers to run the logic. |
| 0x0000000000400635 <+8>: mov %edi,-0x14(%rbp)  0x0000000000400638 <+11>: movl $0x1,-0x4(%rbp)  0x000000000040063f <+18>: mov -0x14(%rbp),%eax  0x0000000000400642 <+21>: mov %eax,-0x8(%rbp)  0x0000000000400645 <+24>: jmp 0x400669 <PrintFact+60> | Push passed variable from register %edi to 14 bytes above %rbp. Push the value 1 to 4 bytes above register %rbp. Move value from 14 bytes above %rbp to register %eax. Push value from register %eax to 8 bytes above register %rbp. Jump PrintFact+60 |
| 0x0000000000400647 <+26>: mov -0x4(%rbp),%eax  0x000000000040064a <+29>: imul -0x8(%rbp),%eax  0x000000000040064e <+33>: mov %eax,-0x4(%rbp)  0x0000000000400651 <+36>: mov -0x8(%rbp),%eax | Pop value from 4 bytes above %rbp to register %eax. Multiply the value 8 bytes above rbp by the value in register %eax and store the result in register %eax. Push the value from register %eax to 4 bytes above %rbp and pop the value from 8 bytes above %rbp to register %eax. |
| 0x0000000000400654 <+39>: mov %eax,%esi  0x0000000000400656 <+41>: mov $0x400844,%edi  0x000000000040065b <+46>: mov $0x0,%eax  0x0000000000400660 <+51>: callq 0x4004f0 <printf@plt> | Move result from register %eax to register %esi. Move string to register %edi. Print to screen the numbers in the factorial. |
| 0x0000000000400665 <+56>: subl $0x1,-0x8(%rbp)  0x0000000000400669 <+60>: cmpl $0x0,-0x8(%rbp)  0x000000000040066d <+64>: jg 0x400647 <PrintFact+26> | Loop conditions  Subtract 1 from value 8 bytes above register %rbp. Store value at 8 bytes above register %rbp. Compare value to 0. If the value is greater than 0, jump to PrintFact+26. |
| 0x000000000040066f <+66>: mov -0x4(%rbp),%eax  0x0000000000400672 <+69>: mov %eax,%esi  0x0000000000400674 <+71>: mov $0x400848,%edi  0x0000000000400679 <+76>: mov $0x0,%eax  0x000000000040067e <+81>: callq 0x4004f0 <printf@plt> | Pop value 4 bytes above %rbp to register %eax. Move value from register %eax to register %esi. Move string to register %edi. Print to screen the factorial result within brackets. |
| 0x0000000000400683 <+86>: mov -0x4(%rbp),%eax  0x0000000000400686 <+89>: leaveq  0x0000000000400687 <+90>: retq | Pop value at 4 bytes above register %rbp to register %eax. Terminate function and return value. |
| Print Sum | 0x0000000000400688 <+0>: push %rbp  0x0000000000400689 <+1>: mov %rsp,%rbp  0x000000000040068c <+4>: sub $0x20,%rsp | Pushes the values of the registers onto the stack and uses the registers to run the logic. |
| 0x0000000000400690 <+8>: mov %edi,-0x14(%rbp)  0x0000000000400693 <+11>: movl $0x0,-0x4(%rbp)  0x000000000040069a <+18>: mov -0x14(%rbp),%eax  0x000000000040069d <+21>: mov %eax,-0x8(%rbp)  0x00000000004006a0 <+24>: jmp 0x4006c0 <PrintSum+56> | Push passed variable from register %edi to 14 bytes above %rbp. Push the value 0 to 4 bytes above register %rbp. Pop value from 14 bytes above %rbp to register %eax. Push value from register %eax to 8 bytes above register %rbp. Jump PrintFact+56 |
| 0x00000000004006a2 <+26>: mov -0x8(%rbp),%eax  0x00000000004006a5 <+29>: add %eax,-0x4(%rbp)  0x00000000004006a8 <+32>: mov -0x8(%rbp),%eax | Pop value from 8 bytes above %rbp to register %eax. Add the value in register %eax to the value 4 bytes above %rbp and store the result 4 bytes above %rbp. Pop the value from 8 bytes above %rbp to register %eax. |
| 0x00000000004006ab <+35>: mov %eax,%esi  0x00000000004006ad <+37>: mov $0x400844,%edi  0x00000000004006b2 <+42>: mov $0x0,%eax  0x00000000004006b7 <+47>: callq 0x4004f0 <printf@plt> | Move result from register %eax to register %esi. Move string to register %edi. Print to screen the numbers in the summation in descending order. |
| 0x00000000004006bc <+52>: subl $0x1,-0x8(%rbp)  0x00000000004006c0 <+56>: cmpl $0x0,-0x8(%rbp)  0x00000000004006c4 <+60>: jg 0x4006a2 <PrintSum+26> | Loop conditions  Subtract 1 from value 8 bytes above register %rbp. Store value at 8 bytes above register %rbp. Compare value to 0. If the value is greater than 0, jump to PrintFact+26. |
| 0x00000000004006c6 <+62>: mov -0x4(%rbp),%eax  0x00000000004006c9 <+65>: mov %eax,%esi  0x00000000004006cb <+67>: mov $0x400848,%edi  0x00000000004006d0 <+72>: mov $0x0,%eax  0x00000000004006d5 <+77>: callq 0x4004f0 <printf@plt> | Pop value 4 bytes above %rbp to register %eax. Move value from register %eax to register %esi. Move string to register %edi. Print to screen the summation result within brackets. |
| 0x00000000004006da <+82>: mov -0x4(%rbp),%eax  0x00000000004006dd <+85>: leaveq  0x00000000004006de <+86>: retq | Pop value at 4 bytes above register %rbp to register %eax. Terminate function and return value. |
| DisplayMenu | 0x00000000004006df <+0>: push %rbp  0x00000000004006e0 <+1>: mov %rsp,%rbp | Pushes the values of the registers onto the stack and uses the registers to run the logic. |
| 0x00000000004006e3 <+4>: mov $0x400851,%edi  0x00000000004006e8 <+9>: callq 0x4004e0 <puts@plt> | Move string to register %edi. Print “\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*” to screen. End with newline. |
| 0x00000000004006ed <+14>: mov $0x400864,%edi  0x00000000004006f2 <+19>: callq 0x4004e0 <puts@plt> | Move string to register %edi. Print “1. Factorial” to screen. End with newline. |
| 0x00000000004006f7 <+24>: mov $0x400871,%edi  0x00000000004006fc <+29>: callq 0x4004e0 <puts@plt> | Move string to register %edi. Print “2. Summation” to screen. End with newline. |
| 0x00000000004006f7 <+24>: mov $0x400871,%edi  0x00000000004006fc <+29>: callq 0x4004e0 <puts@plt> | Move string to register %edi. Print “3. Quit” to screen. End with newline. |
| 0x0000000000400701 <+34>: mov $0x40087e,%edi  0x0000000000400706 <+39>: callq 0x4004e0 <puts@plt> | Move string to register %edi. Print “\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*” to screen. End with newline. |
| 0x000000000040070b <+44>: mov $0x400851,%edi  0x0000000000400710 <+49>: callq 0x4004e0 <puts@plt> | Move string to register %edi. Print “Enter your number:” to screen. End with newline. |
| 0x0000000000400715 <+54>: pop %rbp  0x0000000000400716 <+55>: retq | Clear stack. Exit function and return to main function. |
| main | 0x0000000000400717 <+0>: push %rbp  0x0000000000400718 <+1>: mov %rsp,%rbp  0x000000000040071b <+4>: sub $0x10,%rsp | Pushes the values of the registers onto the stack and uses the registers to run the logic. |
| 0x000000000040071f <+8>: movl $0x0,-0x8(%rbp)  0x0000000000400726 <+15>: jmp 0x4007a0 <main+137> | Reserve space 8 bytes above register %rbp by storing 0 there. Jump to main+137 |
| 0x0000000000400728 <+17>: mov $0x0,%eax  0x000000000040072d <+22>: callq 0x4006df <DisplayMenu> | Store value 0 in register %eax. Call DisplayMenu function. |
| 0x0000000000400732 <+27>: mov $0x400886,%edi  0x0000000000400737 <+32>: callq 0x4004e0 <puts@plt> | Move string to register %edi. Print “ Enter you number:” to screen. End with newline. |
| 0x000000000040073c <+37>: lea -0x8(%rbp),%rax  0x0000000000400740 <+41>: mov %rax,%rsi  0x0000000000400743 <+44>: mov $0x400899,%edi  0x0000000000400748 <+49>: mov $0x0,%eax  0x000000000040074d <+54>: callq 0x400520 <\_\_isoc99\_scanf@plt> | Place the address of the variable at 8 bytes above register %rbp into register %rax.  Move value in register %rax to register %rsi.  Move address to register %edi. Call cin function to read integer value from the user. |
| 0x0000000000400752 <+59>: mov -0x8(%rbp),%eax  0x0000000000400755 <+62>: cmp $0x3,%eax  0x0000000000400758 <+65>: je 0x40077a <main+99> | Switch Statement  Move user input from 8 bytes above register %rbp to register %eax. Compare value to 3. If the values are equal, jump to main+99. |
| 0x000000000040075a <+67>: mov $0x40089c,%edi  0x000000000040075f <+72>: callq 0x4004e0 <puts@plt> | Move string to register %edi. Print “ Enter a number:” to screen. End with newline. |
| 0x0000000000400764 <+77>: lea -0x4(%rbp),%rax  0x0000000000400768 <+81>: mov %rax,%rsi  0x000000000040076b <+84>: mov $0x400899,%edi  0x0000000000400770 <+89>: mov $0x0,%eax  0x0000000000400775 <+94>: callq 0x400520 <\_\_isoc99\_scanf@plt> | Place the address of the variable at 4 bytes above register %rbp into register %rax.  Move value in register %rax to register %rsi.  Move address to register %edi. Call cin function to read integer value from the user. |
| 0x000000000040077a <+99>: mov -0x8(%rbp),%eax  0x000000000040077d <+102>: cmp $0x1,%eax  0x0000000000400780 <+105>: jne 0x40078e <main+119> | Switch Statement  Move user input from 8 bytes above register %rbp to register %eax. Compare value to 1. If the values are not equal, jump to main+119. |
| 0x0000000000400782 <+107>: mov -0x4(%rbp),%eax  0x0000000000400785 <+110>: mov %eax,%edi  0x0000000000400787 <+112>: callq 0x40062d <PrintFact> | Move user input values from 4 bytes above register %rbp to registers %eax. Move values from register %eax to register %edi to be passed to function PrintFact. |
| 0x000000000040078c <+117>: jmp 0x4007a0 <main+137> | Jump to main+137. |
| 0x000000000040078e <+119>: mov -0x8(%rbp),%eax  0x0000000000400791 <+122>: cmp $0x2,%eax  0x0000000000400794 <+125>: jne 0x4007a0 <main+137> | Switch Statement  Move user input from 8 bytes above register %rbp to register %eax. Compare value to 2. If the values are not equal, jump to main+137. |
| 0x0000000000400796 <+127>: mov -0x4(%rbp),%eax  0x0000000000400799 <+130>: mov %eax,%edi  0x000000000040079b <+132>: callq 0x400688 <PrintSum> | Move user input values from 4 bytes above register %rbp to registers %eax. Move values from register %eax to register %edi to be passed to function PrintSum. |
| 0x00000000004007a0 <+137>: mov -0x8(%rbp),%eax  0x00000000004007a3 <+140>: cmp $0x3,%eax  0x00000000004007a6 <+143>: jne 0x400728 <main+17> | Switch Statement  Move user input from 8 bytes above register %rbp to register %eax. Compare value to 3. If the values are not equal, jump to main+17. |
|  | 0x00000000004007a8 <+145>: mov $0x0,%edi  0x00000000004007ad <+150>: callq 0x400530 <exit@plt> | Move value 0 to register %edi. Return value and terminate function. |