# CS 410 Project One Proficiency Test Template

## Explain the functionality of the blocks of assembly code.

### “main” function”

| **Assembly Code Block** | **Explanation of Functionality** |
| --- | --- |
| 0x0000000000000000 <+0>: push %rbp  0x0000000000000001 <+1>: mov %rsp,%rbp  0x0000000000000004 <+4>: lea 0x0(%rip),%rsi # 0xb <main+11>  0x000000000000000b <+11>: lea 0x0(%rip),%rdi # 0x12 <main+18>  0x0000000000000012 <+18>: callq 0x17 <main+23>  0x0000000000000017 <+23>: callq 0x1c <main+28>  0x000000000000001c <+28>: mov %eax,0x0(%rip) # 0x22 <main+34>  0x0000000000000022 <+34>: mov 0x0(%rip),%eax # 0x28 <main+40>  0x0000000000000028 <+40>: cmp $0x1,%eax  0x000000000000002b <+43>: je 0x40 <main+64> | This block of code starts with a push which starts the stack, then has a mov. Which moves the bits into the stack. Then we have a lea which stands for load effective address. After that callq moves 0x17 into the stack. And so on down the list. |
| 0x000000000000004d <+77>: lea 0x0(%rip),%rsi # 0x54 <main+84>  0x0000000000000054 <+84>: lea 0x0(%rip),%rdi # 0x5b <main+91>  0x000000000000005b <+91>: callq 0x60 <main+96>  0x0000000000000060 <+96>: lea 0x0(%rip),%rsi # 0x67 <main+103>  0x0000000000000067 <+103>: lea 0x0(%rip),%rdi # 0x6e <main+110> | Lea moves %rsi into the stack and %rdi into the stack then callq moves 0x60 bits into the stack. |
| 0x00000000000000cd <+205>: callq 0xd2 <main+210>  0x00000000000000d2 <+210>: mov %rax,%rdx  0x00000000000000d5 <+213>: mov 0x0(%rip),%rax # 0xdc <main+220>  0x00000000000000dc <+220>: mov %rax,%rsi  0x00000000000000df <+223>: mov %rdx,%rdi  0x00000000000000e2 <+226>: callq 0xe7 <main+231>  0x00000000000000e7 <+231>: mov 0x0(%rip),%eax # 0xed <main+237> | At the top callq moves 0xd2 into the main stack and so on down the list. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

### ChangeCustomerChoice function

| **Assembly Code Block** | **Explanation of Functionality** |
| --- | --- |
| 0x000000000000042d <+0>: push %rbp  0x000000000000042e <+1>: mov %rsp,%rbp | Push moves %rbp into the stack for this function  Then mov moves %rsp into %rbp |
| 0x0000000000000431 <+4>: lea 0x0(%rip),%rsi # 0x438 <\_Z20ChangeCustomerChoicev+11>  0x0000000000000438 <+11>: lea 0x0(%rip),%rdi # 0x43f <\_Z20ChangeCustomerChoicev+18> | Lea again moves %rip into %rsi and for the sake of this function I believe changes the customer choice. |
| 0x0000000000000488 <+91>: mov 0x0(%rip),%eax # 0x48e <\_Z20ChangeCustomerChoicev+97>  0x000000000000048e <+97>: mov %eax,0x0(%rip) # 0x494 <\_Z20ChangeCustomerChoicev+103> | Again we have some mov commands which just moves the bits from %rip into %eax in the stack |
|  |  |
|  |  |

### CheckUserPermissonAccess Function

| **Assembly Code Block** | **Explanation of Functionality** |
| --- | --- |
| 0x0000000000000120 <+0>: push %rbp  0x0000000000000121 <+1>: mov %rsp,%rbp  0x0000000000000124 <+4>: push %rbx  0x0000000000000125 <+5>: sub $0x48,%rsp  0x0000000000000129 <+9>: mov %fs:0x28,%rax  0x0000000000000132 <+18>: mov %rax,-0x18(%rbp) | From the beginning we have more push and mov commands which again just move bits into the memory  There is also a sub cmd which subtracts those bits from the stack |
| 0x0000000000000136 <+22>: xor %eax,%eax  0x0000000000000138 <+24>: lea -0x45(%rbp),%rax  0x000000000000013c <+28>: mov %rax,%rdi | When there is only 1’s and 0’s there’s only two options one OR the other. Therefor xor means it’s not one but the other. In this case %eax not %eax |
|  |  |
|  |  |

### DisplayInfo Function

| **Assembly Code Block** | **Explanation of Functionality** |
| --- | --- |
| 0x0000000000000241 <+0>: push %rbp  0x0000000000000242 <+1>: mov %rsp,%rbp  0x0000000000000245 <+4>: lea 0x0(%rip),%rsi # 0x24c <\_Z11DisplayInfov+11>  0x000000000000024c <+11>: lea 0x0(%rip),%rdi # 0x253 <\_Z11DisplayInfov+18>  0x0000000000000253 <+18>: callq 0x258 <\_Z11DisplayInfov+23>  0x0000000000000258 <+23>: mov %rax,%rdx  0x000000000000025b <+26>: mov 0x0(%rip),%rax # 0x262 <\_Z11DisplayInfov+33>  0x0000000000000262 <+33>: mov %rax,%rsi  0x0000000000000265 <+36>: mov %rdx,%rdi  0x0000000000000268 <+39>: callq 0x26d <\_Z11DisplayInfov+44>  0x000000000000026d <+44>: lea 0x0(%rip),%rsi # 0x274 <\_Z11DisplayInfov+51>  0x0000000000000274 <+51>: lea 0x0(%rip),%rdi # 0x27b <\_Z11DisplayInfov+58>  0x000000000000027b <+58>: callq 0x280 <\_Z11DisplayInfov+63>  0x0000000000000280 <+63>: lea 0x0(%rip),%rsi # 0x287 <\_Z11DisplayInfov+70>  0x0000000000000287 <+70>: mov %rax,%rdi | Here is another list of cmds including the push mov and lea which moves bits into the stack for the function. |
| 0x000000000000030a <+201>: mov %rax,%rdx  0x000000000000030d <+204>: mov 0x0(%rip),%rax # 0x314 <\_Z11DisplayInfov+211>  0x0000000000000314 <+211>: mov %rax,%rsi  0x0000000000000317 <+214>: mov %rdx,%rdi | Here is another list of mov cmds |
|  |  |
|  |  |
|  |  |