**5-2 Project One**

Dylan Jeffery

SNHU

CS-410

Brian West

2/6/2022

# CS 410 Project One Proficiency Test Template

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

### “main” function”

| **Assembly Code Block** | **Explanation of Functionality** |
| --- | --- |
| push %rbp  mov %rsp,%rbp  lea 0x0(%rip),%rsi # b <main+0xb>  lea 0x0(%rip),%rdi # 12 <main+0x12>  callq 17 <main+0x17>  callq 1c <main+0x1c>  mov %eax,0x0(%rip) # 22 <main+0x22>  mov 0x0(%rip),%eax # 28 <main+0x28> | Mostly initialization |
| cmp $0x1,%eax  je 40 <main+0x40>  lea 0x0(%rip),%rsi # 34 <main+0x34>  lea 0x0(%rip),%rdi # 3b <main+0x3b>  callq 40 <main+0x40>  mov 0x0(%rip),%eax # 46 <main+0x46> | A few comparisons and loading effective addresses, callq is a cout statement. |
| mov 0x0(%rip),%eax # 46 <main+0x46>  cmp $0x1,%eax  je 4d <main+0x4d>  jmp 17 <main+0x17>  lea 0x0(%rip),%rsi # 54 <main+0x54>  lea 0x0(%rip),%rdi # 5b <main+0x5b>  callq 60 <main+0x60>  lea 0x0(%rip),%rsi # 67 <main+0x67>  lea 0x0(%rip),%rdi # 6e <main+0x6e>  callq 73 <main+0x73>  lea 0x0(%rip),%rsi # 7a <main+0x7a>  lea 0x0(%rip),%rdi # 81 <main+0x81>  callq 86 <main+0x86>  lea 0x0(%rip),%rsi # 8d <main+0x8d>  lea 0x0(%rip),%rdi # 94 <main+0x94>  callq 99 <main+0x99>  lea 0x0(%rip),%rsi # a0 <main+0xa0>  lea 0x0(%rip),%rdi # a7 <main+0xa7>  callq ac <main+0xac>  lea 0x0(%rip),%rsi # b3 <main+0xb3>  lea 0x0(%rip),%rdi # ba <main+0xba>  callq bf <main+0xbf>  mov %rax,%rdx  mov 0x0(%rip),%eax # c8 <main+0xc8>  mov %eax,%esi  mov %rdx,%rdi  callq d2 <main+0xd2>  mov %rax,%rdx  mov 0x0(%rip),%rax # dc <main+0xdc>  mov %rax,%rsi  mov %rdx,%rdi  callq e7 <main+0xe7>  mov 0x0(%rip),%eax # ed <main+0xed>  cmp $0x1,%eax  jne f9 <main+0xf9>  callq f7 <main+0xf7>  jmp 109 <main+0x109>  mov 0x0(%rip),%eax # ff <main+0xff>  cmp $0x2,%eax  jne 109 <main+0x109>  callq 109 <main+0x109>  mov 0x0(%rip),%eax # 10f <main+0x10f>  cmp $0x3,%eax  je 119 <main+0x119>  jmpq 4d <main+0x4d>  mov $0x0,%eax  pop %rbp  retq | A few comparisons and more cout statements. Most of this stuff is user prompts and welcoming our user to the program. Several comparison statements at the tail end of this hint to us multiple logic statements and perhaps a while loop is involved. |

### ChangeCustomerChoice function

| **Assembly Code Block** | **Explanation of Functionality** |
| --- | --- |
| push %rbp  mov %rsp,%rbp  lea 0x0(%rip),%rsi # 438 <\_Z20ChangeCustomerChoicev+0xb>  lea 0x0(%rip),%rdi # 43f <\_Z20ChangeCustomerChoicev+0x12>  callq 444 <\_Z20ChangeCustomerChoicev+0x17>  lea 0x0(%rip),%rsi # 44b <\_Z20ChangeCustomerChoicev+0x1e>  lea 0x0(%rip),%rdi # 452 <\_Z20ChangeCustomerChoicev+0x25>  callq 457 <\_Z20ChangeCustomerChoicev+0x2a>  lea 0x0(%rip),%rsi # 45e <\_Z20ChangeCustomerChoicev+0x31>  lea 0x0(%rip),%rdi # 465 <\_Z20ChangeCustomerChoicev+0x38>  callq 46a <\_Z20ChangeCustomerChoicev+0x3d>  lea 0x0(%rip),%rsi # 471 <\_Z20ChangeCustomerChoicev+0x44>  lea 0x0(%rip),%rdi # 478 <\_Z20ChangeCustomerChoicev+0x4b>  callq 47d <\_Z20ChangeCustomerChoicev+0x50>  mov 0x0(%rip),%eax # 483 <\_Z20ChangeCustomerChoicev+0x56>  cmp $0x1,%eax  jne 496 <\_Z20ChangeCustomerChoicev+0x69>  mov 0x0(%rip),%eax # 48e <\_Z20ChangeCustomerChoicev+0x61>  mov %eax,0x0(%rip) # 494 <\_Z20ChangeCustomerChoicev+0x67>  jmp 4f8 <\_Z20ChangeCustomerChoicev+0xcb>  mov 0x0(%rip),%eax # 49c <\_Z20ChangeCustomerChoicev+0x6f>  cmp $0x2,%eax  jne 4af <\_Z20ChangeCustomerChoicev+0x82>  mov 0x0(%rip),%eax # 4a7 <\_Z20ChangeCustomerChoicev+0x7a>  mov %eax,0x0(%rip) # 4ad <\_Z20ChangeCustomerChoicev+0x80>  jmp 4f8 <\_Z20ChangeCustomerChoicev+0xcb> | Prompt the user a few times and take user input. Comparisons indicate if blocks depending on user input. Capable of modifying service decision for clients and then jumps back into menu loop. |
| mov 0x0(%rip),%eax # 4b5 <\_Z20ChangeCustomerChoicev+0x88>  cmp $0x3,%eax  jne 4c8 <\_Z20ChangeCustomerChoicev+0x9b>  mov 0x0(%rip),%eax # 4c0 <\_Z20ChangeCustomerChoicev+0x93>  mov %eax,0x0(%rip) # 4c6 <\_Z20ChangeCustomerChoicev+0x99>  jmp 4f8 <\_Z20ChangeCustomerChoicev+0xcb>  mov 0x0(%rip),%eax # 4ce <\_Z20ChangeCustomerChoicev+0xa1>  cmp $0x4,%eax  jne 4e1 <\_Z20ChangeCustomerChoicev+0xb4>  mov 0x0(%rip),%eax # 4d9 <\_Z20ChangeCustomerChoicev+0xac>  mov %eax,0x0(%rip) # 4df <\_Z20ChangeCustomerChoicev+0xb2>  jmp 4f8 <\_Z20ChangeCustomerChoicev+0xcb>  mov 0x0(%rip),%eax # 4e7 <\_Z20ChangeCustomerChoicev+0xba>  cmp $0x5,%eax  jne 4f8 <\_Z20ChangeCustomerChoicev+0xcb>  mov 0x0(%rip),%eax # 4f2 <\_Z20ChangeCustomerChoicev+0xc5>  mov %eax,0x0(%rip) # 4f8 <\_Z20ChangeCustomerChoicev+0xcb>  nop  pop %rbp  retq | More moves and conditional logic for modifying client service choice. |

### CheckUserPermissonAccess Function

| **Assembly Code Block** | **Explanation of Functionality** |
| --- | --- |
| push %rbp  mov %rsp,%rbp  push %rbx  sub $0x48,%rsp  mov %fs:0x28,%rax  mov %rax,-0x18(%rbp)  xor %eax,%eax  lea -0x45(%rbp),%rax  mov %rax,%rdi | Mostly variable initialization |
| callq 144 <\_Z25CheckUserPermissionAccessv+0x24>  lea -0x45(%rbp),%rdx  lea -0x40(%rbp),%rax  lea 0x0(%rip),%rsi # 153 <\_Z25CheckUserPermissionAccessv+0x33>  mov %rax,%rdi  callq 15b <\_Z25CheckUserPermissionAccessv+0x3b>  lea -0x45(%rbp),%rax  mov %rax,%rdi  callq 167 <\_Z25CheckUserPermissionAccessv+0x47>  movl $0x0,-0x44(%rbp)  lea 0x0(%rip),%rsi # 175 <\_Z25CheckUserPermissionAccessv+0x55>  lea 0x0(%rip),%rdi # 17c <\_Z25CheckUserPermissionAccessv+0x5c>  callq 181 <\_Z25CheckUserPermissionAccessv+0x61>  lea 0x0(%rip),%rsi # 188 <\_Z25CheckUserPermissionAccessv+0x68>  lea 0x0(%rip),%rdi # 18f <\_Z25CheckUserPermissionAccessv+0x6f>  callq 194 <\_Z25CheckUserPermissionAccessv+0x74>  lea 0x0(%rip),%rsi # 19b <\_Z25CheckUserPermissionAccessv+0x7b>  lea 0x0(%rip),%rdi # 1a2 <\_Z25CheckUserPermissionAccessv+0x82>  callq 1a7 <\_Z25CheckUserPermissionAccessv+0x87>  lea -0x40(%rbp),%rax  mov %rax,%rsi  lea 0x0(%rip),%rdi # 1b5 <\_Z25CheckUserPermissionAccessv+0x95>  callq 1ba <\_Z25CheckUserPermissionAccessv+0x9a> | Several cout statements prompting user input and other variables assigned here. Here is where the user is asked for a password. |
| lea -0x40(%rbp),%rax  lea 0x0(%rip),%rsi # 1c5 <\_Z25CheckUserPermissionAccessv+0xa5>  mov %rax,%rdi  callq 1cd <\_Z25CheckUserPermissionAccessv+0xad>  mov %eax,-0x44(%rbp)  cmpl $0x0,-0x44(%rbp)  jne 1dd <\_Z25CheckUserPermissionAccessv+0xbd>  mov $0x1,%ebx  jmp 1e2 <\_Z25CheckUserPermissionAccessv+0xc2>  mov $0x2,%ebx  lea -0x40(%rbp),%rax  mov %rax,%rdi  callq 1ee <\_Z25CheckUserPermissionAccessv+0xce>  mov %ebx,%eax  mov -0x18(%rbp),%rcx  xor %fs:0x28,%rcx  je 23a <\_Z25CheckUserPermissionAccessv+0x11a>  jmp 235 <\_Z25CheckUserPermissionAccessv+0x115> | We see several compare statements as well as a xor je jmp block indicating the presence of a while loop. |
| mov %rax,%rbx  lea -0x45(%rbp),%rax  mov %rax,%rdi  callq 210 <\_Z25CheckUserPermissionAccessv+0xf0>  mov %rbx,%rax  mov %rax,%rdi  callq 21b <\_Z25CheckUserPermissionAccessv+0xfb>  mov %rax,%rbx  lea -0x40(%rbp),%rax  mov %rax,%rdi  callq 22a <\_Z25CheckUserPermissionAccessv+0x10a>  mov %rbx,%rax  mov %rax,%rdi  callq 235 <\_Z25CheckUserPermissionAccessv+0x115>  callq 23a <\_Z25CheckUserPermissionAccessv+0x11a>  add $0x48,%rsp  pop %rbx  pop %rbp  retq | Inside this while loop we have cout statements displaying our user menu. In my C++ code this is where menu choices are handled as well. |

### DisplayInfo Function

| **Assembly Code Block** | **Explanation of Functionality** |
| --- | --- |
| push %rbp  mov %rsp,%rbp  lea 0x0(%rip),%rsi # 24c <\_Z11DisplayInfov+0xb>  lea 0x0(%rip),%rdi # 253 <\_Z11DisplayInfov+0x12>  callq 258 <\_Z11DisplayInfov+0x17>  mov %rax,%rdx  mov 0x0(%rip),%rax # 262 <\_Z11DisplayInfov+0x21>  mov %rax,%rsi  mov %rdx,%rdi  callq 26d <\_Z11DisplayInfov+0x2c> | All of this |
| lea 0x0(%rip),%rsi # 274 <\_Z11DisplayInfov+0x33>  lea 0x0(%rip),%rdi # 27b <\_Z11DisplayInfov+0x3a>  callq 280 <\_Z11DisplayInfov+0x3f>  lea 0x0(%rip),%rsi # 287 <\_Z11DisplayInfov+0x46>  mov %rax,%rdi  callq 28f <\_Z11DisplayInfov+0x4e>  lea 0x0(%rip),%rsi # 296 <\_Z11DisplayInfov+0x55>  mov %rax,%rdi  callq 29e <\_Z11DisplayInfov+0x5d>  mov %rax,%rdx  mov 0x0(%rip),%eax # 2a7 <\_Z11DisplayInfov+0x66>  mov %eax,%esi  mov %rdx,%rdi  callq 2b1 <\_Z11DisplayInfov+0x70>  mov %rax,%rdx  mov 0x0(%rip),%rax # 2bb <\_Z11DisplayInfov+0x7a>  mov %rax,%rsi  mov %rdx,%rdi  callq 2c6 <\_Z11DisplayInfov+0x85> | Is a really long way of showing a lot |
| lea 0x0(%rip),%rsi # 2cd <\_Z11DisplayInfov+0x8c>  lea 0x0(%rip),%rdi # 2d4 <\_Z11DisplayInfov+0x93>  callq 2d9 <\_Z11DisplayInfov+0x98>  lea 0x0(%rip),%rsi # 2e0 <\_Z11DisplayInfov+0x9f>  mov %rax,%rdi  callq 2e8 <\_Z11DisplayInfov+0xa7>  lea 0x0(%rip),%rsi # 2ef <\_Z11DisplayInfov+0xae>  mov %rax,%rdi  callq 2f7 <\_Z11DisplayInfov+0xb6>  mov %rax,%rdx  mov 0x0(%rip),%eax # 300 <\_Z11DisplayInfov+0xbf>  mov %eax,%esi  mov %rdx,%rdi  callq 30a <\_Z11DisplayInfov+0xc9>  mov %rax,%rdx  mov 0x0(%rip),%rax # 314 <\_Z11DisplayInfov+0xd3>  mov %rax,%rsi  mov %rdx,%rdi  callq 31f <\_Z11DisplayInfov+0xde> | Of cout statements |
| lea 0x0(%rip),%rsi # 326 <\_Z11DisplayInfov+0xe5>  lea 0x0(%rip),%rdi # 32d <\_Z11DisplayInfov+0xec>  callq 332 <\_Z11DisplayInfov+0xf1>  lea 0x0(%rip),%rsi # 339 <\_Z11DisplayInfov+0xf8>  mov %rax,%rdi  callq 341 <\_Z11DisplayInfov+0x100>  lea 0x0(%rip),%rsi # 348 <\_Z11DisplayInfov+0x107>  mov %rax,%rdi  callq 350 <\_Z11DisplayInfov+0x10f>  mov %rax,%rdx  mov 0x0(%rip),%eax # 359 <\_Z11DisplayInfov+0x118>  mov %eax,%esi  mov %rdx,%rdi  callq 363 <\_Z11DisplayInfov+0x122>  mov %rax,%rdx  mov 0x0(%rip),%rax # 36d <\_Z11DisplayInfov+0x12c>  mov %rax,%rsi  mov %rdx,%rdi  callq 378 <\_Z11DisplayInfov+0x137>  lea 0x0(%rip),%rsi # 37f <\_Z11DisplayInfov+0x13e>  lea 0x0(%rip),%rdi # 386 <\_Z11DisplayInfov+0x145>  callq 38b <\_Z11DisplayInfov+0x14a>  lea 0x0(%rip),%rsi # 392 <\_Z11DisplayInfov+0x151>  mov %rax,%rdi | Reading variables related to client names and services. |
| callq 39a <\_Z11DisplayInfov+0x159>  lea 0x0(%rip),%rsi # 3a1 <\_Z11DisplayInfov+0x160>  mov %rax,%rdi  callq 3a9 <\_Z11DisplayInfov+0x168>  mov %rax,%rdx  mov 0x0(%rip),%eax # 3b2 <\_Z11DisplayInfov+0x171>  mov %eax,%esi  mov %rdx,%rdi  callq 3bc <\_Z11DisplayInfov+0x17b>  mov %rax,%rdx  mov 0x0(%rip),%rax # 3c6 <\_Z11DisplayInfov+0x185>  mov %rax,%rsi  mov %rdx,%rdi  callq 3d1 <\_Z11DisplayInfov+0x190>  lea 0x0(%rip),%rsi # 3d8 <\_Z11DisplayInfov+0x197>  lea 0x0(%rip),%rdi # 3df <\_Z11DisplayInfov+0x19e>  callq 3e4 <\_Z11DisplayInfov+0x1a3>  lea 0x0(%rip),%rsi # 3eb <\_Z11DisplayInfov+0x1aa>  mov %rax,%rdi  callq 3f3 <\_Z11DisplayInfov+0x1b2>  lea 0x0(%rip),%rsi # 3fa <\_Z11DisplayInfov+0x1b9>  mov %rax,%rdi  callq 402 <\_Z11DisplayInfov+0x1c1>  mov %rax,%rdx  mov 0x0(%rip),%eax # 40b <\_Z11DisplayInfov+0x1ca>  mov %eax,%esi  mov %rdx,%rdi  callq 415 <\_Z11DisplayInfov+0x1d4>  mov %rax,%rdx  mov 0x0(%rip),%rax # 41f <\_Z11DisplayInfov+0x1de>  mov %rax,%rsi  mov %rdx,%rdi  callq 42a <\_Z11DisplayInfov+0x1e9>  nop  pop %rbp  retq | Once all statements are displayed, we return to the menu loop. |