# 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 # 0xb <main+11>  lea 0x0(%rip),%rdi # 0x12 <main+18>  callq 0x17 <main+23>  callq 0x1c <main+28> | %rbp moved in %rsp  %rsi loaded in 0(%rip)  %rdi loaded in 0(%rip)  callq  callq |
| mov %eax,0x0(%rip) # 0x22 <main+34>  mov 0x0(%rip),%eax # 0x28 <main+40>  cmp $0x1,%eax  je 0x40 <main+64>  callq 0x40 <main+64> | 0(%rip) moved in %eax  0(%rip) moved in %eax  Compare %eax  Jump to <main+64> if it is equal  callq |
| mov 0x0(%rip),%eax # 0x46 <main+70>  cmp $0x1,%eax  je 0x4d <main+77>  jmp 0x17 <main+23>  callq 0x60 <main+96> | %eax moved in 0(%rip)  Compare %eax  Jump to <main+77> if it is equal  Jump <main+23>  callq |
| lea 0x0(%rip),%rsi # 0x67 <main+103>  lea 0x0(%rip),%rdi # 0x6e <main+110>  callq 0x73 <main+115>  lea 0x0(%rip),%rsi # 0x7a <main+122>  lea 0x0(%rip),%rdi # 0x81 <main+129>  callq 0x86 <main+134> | %rsi loaded in 0(%rip)  %rdi loaded in 0(%rip)  callq  %rsi loaaded in 0(%rip)  %rdi loaded in 0(%rip)  callq |
| lea 0x0(%rip),%rsi # 0x8d <main+141>  lea 0x0(%rip),%rdi # 0x94 <main+148>  callq 0x99 <main+153>  lea 0x0(%rip),%rsi # 0xa0 <main+160>  lea 0x0(%rip),%rdi # 0xa7 <main+167>  callq 0xac <main+172> | %rsi loaded in 0(%rip)  %rdi loaded in 0(%rip)  callq  %rsi loaded in 0(%rip)  %rdi loaded in 0(%rip) |
| lea 0x0(%rip),%rsi # 0xb3 <main+179>  lea 0x0(%rip),%rdi # 0xba <main+186>  callq 0xbf <main+191>  mov %rax,%rdx  mov 0x0(%rip),%eax # 0xc8 <main+200>  mov %eax,%esi  mov %rdx,%rdi  callq 0xd2 <main+210> | %rsi loaded in 0(%rip)  %rdi loaded in 0(%rip)  callq <main+192>  %rdx moved in %rax  %eax moved in 0(%rip)  %esi moved in %eax  %rdi move in %rdi  callq <main+210> |
| mov %rax,%rdx  mov 0x0(%rip),%rax # 0xdc <main+220>  mov %rax,%rsi  mov %rdx,%rdi  callq 0xe7 <main+231> | %rdx moved in %rax  %rax moved in 0(%rip)  %rsi moved in %rax  %rdi moved in %rdx  callq <main+231> |
| mov 0x0(%rip),%eax # 0xed <main+237>  cmp $0x1,%eax  jne 0xf9 <main+249>  callq 0xf7 <main+247> | %eax moved in 0(%rip)  Compare %eax  Conditional jump  callq <main+247> |
| jmp 0x109 <main+265>  mov 0x0(%rip),%eax # 0xff <main+255>  cmp $0x2,%eax  jne 0x109 <main+265>  callq 0x109 <main+265> | Jump to <main+265>  %eax moved in 0(%rip)  Compare %eax  Conditional jump  callq <main+265> |
| mov 0x0(%rip),%eax # 0x10f <main+271>  cmp $0x3,%eax  je 0x119 <main+281>  jmpq 0x4d <main+77>  pop %rbp  retq | %eax moved in 0(%rip)  Compare %eax  Jump <main+281> if equal to it  Indirect jump to <main+77>  %rbp removed  return |

### ChangeCustomerChoice function

| **Assembly Code Block** | **Explanation of Functionality** |
| --- | --- |
| mov 0x0(%rip),%eax # 0x483 <\_Z20ChangeCustomerChoicev+86>  cmp $0x1,%eax  jne 0x496 <\_Z20ChangeCustomerChoicev+105>  mov 0x0(%rip),%eax # 0x48e <\_Z20ChangeCustomerChoicev+97>  mov %eax,0x0(%rip) # 0x494 <\_Z20ChangeCustomerChoicev+103>  jmp 0x4f8 <\_Z20ChangeCustomerChoicev+203> | %eax moved in 0(%rip)  Compare %eax  Conditional jump  %eax moved in 0(%rip)  0(%rip) moved in %eax  Jump |
| mov 0x0(%rip),%eax # 0x49c <\_Z20ChangeCustomerChoicev+111>  cmp $0x2,%eax  jne 0x4af <\_Z20ChangeCustomerChoicev+130>  mov 0x0(%rip),%eax # 0x4a7 <\_Z20ChangeCustomerChoicev+122>  mov %eax,0x0(%rip) # 0x4ad <\_Z20ChangeCustomerChoicev+128>  jmp 0x4f8 <\_Z20ChangeCustomerChoicev+203> | %eax moved in 0(%rip)  Compare %eax  Conditional jump  %eax moved in 0(%rip)  0(%rip) moved in %eax  Jump |
| mov 0x0(%rip),%eax # 0x4b5 <\_Z20ChangeCustomerChoicev+136>  cmp $0x3,%eax  jne 0x4c8 <\_Z20ChangeCustomerChoicev+155>  mov 0x0(%rip),%eax # 0x4c0 <\_Z20ChangeCustomerChoicev+147>  mov %eax,0x0(%rip) # 0x4c6 <\_Z20ChangeCustomerChoicev+153>  jmp 0x4f8 <\_Z20ChangeCustomerChoicev+203> | %eax moved in 0(%rip)  Compare %eax  Conditional jump  %eax moved in 0(%rip)  0(%rip) moved in %eax  Jump |
| mov 0x0(%rip),%eax # 0x4ce <\_Z20ChangeCustomerChoicev+161>  cmp $0x4,%eax  jne 0x4e1 <\_Z20ChangeCustomerChoicev+180>  mov 0x0(%rip),%eax # 0x4d9 <\_Z20ChangeCustomerChoicev+172>  mov %eax,0x0(%rip) # 0x4df <\_Z20ChangeCustomerChoicev+178>  jmp 0x4f8 <\_Z20ChangeCustomerChoicev+203> | %eax moved in 0(%rip)  Compare %eax  Conditional jump  %eax moved in 0(%rip)  0(%rip) moved in %eax  Jump |
| mov 0x0(%rip),%eax # 0x4e7 <\_Z20ChangeCustomerChoicev+186>  cmp $0x5,%eax  jne 0x4f8 <\_Z20ChangeCustomerChoicev+203>  mov 0x0(%rip),%eax # 0x4f2 <\_Z20ChangeCustomerChoicev+197>  mov %eax,0x0(%rip) # 0x4f8 <\_Z20ChangeCustomerChoicev+203>  nop  pop %rbp  retq | %eax moved in 0(%rip)  Compare %eax  Conditional jump  %eax moved in 0(%rip)  0(%rip) moved in %eax  Jump  %rbp removed  return |

### CheckUserPermissonAccess Function

| **Assembly Code Block** | **Explanation of Functionality** |
| --- | --- |
| lea -0x40(%rbp),%rax  mov %rax,%rdi  callq 0x1cd <\_Z25CheckUserPermissionAccessv+173>  mov %eax,-0x44(%rbp)  cmpl $0x0,-0x44(%rbp)  jne 0x1dd <\_Z25CheckUserPermissionAccessv+189>  mov $0x1,%ebx  jmp 0x1e2 <\_Z25CheckUserPermissionAccessv+194>  mov $0x2,%ebx  lea -0x40(%rbp),%rax  mov %rax,%rdi  callq 0x1ee <\_Z25CheckUserPermissionAccessv+206>  mov %ebx,%eax | %rax loaded in -40(%rax)  %rdi moved in %rax  callq  -44(%rbp) moved in %eax  Logic -44(%rbp) compared by 0  Conditional jump  %ebx is moved by 1  Jump  %ebx is moved by 2  %rax loaded in -40(%rbp)  %rdi moved in %rax  callq  %eax moved in %ebx |
| mov -0x18(%rbp),%rcx  xor %fs:0x28,%rcx  je 0x23a <\_Z25CheckUserPermissionAccessv+282>  jmp 0x235 <\_Z25CheckUserPermissionAccessv+277>  mov %rax,%rbx  lea -0x45(%rbp),%rax  mov %rax,%rdi  callq 0x210 <\_Z25CheckUserPermissionAccessv+240> | %rcx moved in -18(%rbp)  if equal, jump  Jump  %rbx moved in %rax  %rax loaded in -45(%rbp)  %rdi moved in %rax  callq |
| mov %rbx,%rax  mov %rax,%rdi  callq 0x21b <\_Z25CheckUserPermissionAccessv+251>  mov %rax,%rbx  lea -0x40(%rbp),%rax  mov %rax,%rdi  callq 0x22a <\_Z25CheckUserPermissionAccessv+266> | %rax moved in %rbx  %rdi moved in %rax  callq  %rbx moved in %rax  %rax loaded in -40(%rbp)  %rdi moved in %rax  callq |
| mov %rbx,%rax  mov %rax,%rdi  callq 0x235 <\_Z25CheckUserPermissionAccessv+277>  callq 0x23a <\_Z25CheckUserPermissionAccessv+282>  add $0x48,%rsp  pop %rbx  pop %rbp  retq | %rax moved in %rbx  %rdi moved in %rax  callq  callq  %rsp is added in allocated space  %rbx removed  %rbp removed  return |

### DisplayInfo Function

| **Assembly Code Block** | **Explanation of Functionality** |
| --- | --- |
| mov %rax,%rdx  mov 0x0(%rip),%rax # 0x262 <\_Z11DisplayInfov+33>  mov %rax,%rsi  mov %rdx,%rdi  callq 0x26d <\_Z11DisplayInfov+44>  lea 0x0(%rip),%rsi # 0x274 <\_Z11DisplayInfov+51>  lea 0x0(%rip),%rdi # 0x27b <\_Z11DisplayInfov+58>  callq 0x280 <\_Z11DisplayInfov+63> | %rdx moved in %rax  %rax moved in 0(%rip)  %rsi moved in %rax  %rdi moved in %rdx  callq  %rsi loaded in 0(%rip)  %rdi loaded in 0(%rip)  callq |
| lea 0x0(%rip),%rsi # 0x287 <\_Z11DisplayInfov+70>  mov %rax,%rdi  callq 0x28f <\_Z11DisplayInfov+78>  lea 0x0(%rip),%rsi # 0x296 <\_Z11DisplayInfov+85>  mov %rax,%rdi  callq 0x29e <\_Z11DisplayInfov+93>  mov %rax,%rdx  mov 0x0(%rip),%eax # 0x2a7 <\_Z11DisplayInfov+102>  mov %eax,%esi  mov %rdx,%rdi  callq 0x2b1 <\_Z11DisplayInfov+112>  mov %rax,%rdx  mov 0x0(%rip),%rax # 0x2bb <\_Z11DisplayInfov+122>  mov %rax,%rsi  mov %rdx,%rdi  callq 0x2c6 <\_Z11DisplayInfov+133> | %rsi loaded in 0(%rip)  %rdi moved in %rax  callq  %rsi loaded in 0(%rip)  %rdi moved in %rax  callq  %rdx moved in %rax  %eax moved in 0(%rip)  %esi moved in %eax  %rdi moved in %rdx  callq  %rdx moved in %rax  %rax moved in 0(%rip)  %rsi moved in %rax  %rdi moved in %rdx  callq |
| lea 0x0(%rip),%rsi # 0x2cd <\_Z11DisplayInfov+140>  lea 0x0(%rip),%rdi # 0x2d4 <\_Z11DisplayInfov+147>  callq 0x2d9 <\_Z11DisplayInfov+152>  lea 0x0(%rip),%rsi # 0x2e0 <\_Z11DisplayInfov+159>  mov %rax,%rdi  callq 0x2e8 <\_Z11DisplayInfov+167>  lea 0x0(%rip),%rsi # 0x2ef <\_Z11DisplayInfov+174>  mov %rax,%rdi  callq 0x2f7 <\_Z11DisplayInfov+182>  mov %rax,%rdx  mov 0x0(%rip),%eax # 0x300 <\_Z11DisplayInfov+191>  mov %eax,%esi  mov %rdx,%rdi  callq 0x30a <\_Z11DisplayInfov+201>  mov %rax,%rdx  mov 0x0(%rip),%rax # 0x314 <\_Z11DisplayInfov+211>  mov %rax,%rsi  mov %rdx,%rdi  callq 0x31f <\_Z11DisplayInfov+222> | %rsi loaded in 0(%rrip)  %rdi loaded in 0(%rip)  callq  %rsi loaded in 0(%rip)  %rdi moved in %rax  callq  %rsi loaded in 0(%rip)  %rdi moved in %rax  callq  %rdx moved in %rax  %eax moved in 0(%rip)  %esi moved in %eax  %rdi moved in %rdx  callq  %rdx moved in %rax  %rax moved in 0(%rip)  %rsi moved in %rax  %rdi moved in %rdx  callq |
| mov %rax,%rdx  mov 0x0(%rip),%rax # 0x3c6 <\_Z11DisplayInfov+389>  mov %rax,%rsi  mov %rdx,%rdi  callq 0x3d1 <\_Z11DisplayInfov+400>  lea 0x0(%rip),%rsi # 0x3d8 <\_Z11DisplayInfov+407>  lea 0x0(%rip),%rdi # 0x3df <\_Z11DisplayInfov+414>  callq 0x3e4 <\_Z11DisplayInfov+419>  lea 0x0(%rip),%rsi # 0x3eb <\_Z11DisplayInfov+426>  mov %rax,%rdi  callq 0x3f3 <\_Z11DisplayInfov+434>  lea 0x0(%rip),%rsi # 0x3fa <\_Z11DisplayInfov+441>  mov %rax,%rdi  callq 0x402 <\_Z11DisplayInfov+449> | %rdx moved in %rax  %rax moved in 0(%rrip)  %rsi moved in %rax  %rdi moved in %rdx  callq  %rsi loaded in 0(%rip)  %rdi loaded in 0(%rip)  callq  %rsi loaded in 0(%rip)  %rdi moved in %rax  callq  %rsi loaded in 0(%rip)  %rdi moved in %rax  callq |
| mov %rax,%rdx  mov 0x0(%rip),%eax # 0x40b <\_Z11DisplayInfov+458>  mov %eax,%esi  mov %rdx,%rdi  callq 0x415 <\_Z11DisplayInfov+468>  mov %rax,%rdx  mov 0x0(%rip),%rax # 0x41f <\_Z11DisplayInfov+478>  mov %rax,%rsi  mov %rdx,%rdi  callq 0x42a <\_Z11DisplayInfov+489>  nop  pop %rbp  retq | %rdx moved in %rax  %eax moved in 0(%rip)  %rdx moved in %rax  %rax moved in 0(%rip)  %rsi moved in %rax  %rdi moved in %rdx  callq  %rbp removed  return |