# 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 | Reserve Stack Pointer before executing any instructions. |
| lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x17 <main+23> | Get the address of string “Hello! Welcome to our Investment Company” and print it out. |
| call 0x1c <main+28>  mov %eax,0x0(%rip)  mov 0x0(%rip),%eax  cmp $0x1,%eax  je 0x40 <main+64> | Set a variable 0 and Make a loop while the return value from CheckUserPermissionAccess is not 1  Keep calling it the function  If the value is 1 |
| lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x40 <main+64>  mov 0x0(%rip),%eax  cmp $0x1,%eax  je 0x4d <main+77>  jmp 0x17 <main+23> |  |
| lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x60 <main+96>  lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x73 <main+115>  lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x86 <main+134> | Get the address of string “What would you like to do?” and print it  Get the address of string “DISPLAY the client list (enter 1)”  Get the address of string “CHANGE a client's choice (enter 2)” and print it |
| lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x99 <main+153>  lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0xac <main+172>  lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0xbf <main+191> | Get the address of “Exit the program.. (enter 3)” print it  Read a value from the user and store it  Get the address of “%d” to printout the selected value.  Print out the selected value    Get the address of “You chose” and print out the entered value |
| mov %rax,%rdx  mov 0x0(%rip),%eax  mov %eax,%esi  mov %rdx,%rdi  call 0xd2 <main+210> |  |
| mov %rax,%rdx  mov 0x0(%rip),%rax  mov %rax,%rsi  mov %rdx,%rdi  call 0xe7 <main+231>  mov 0x0(%rip),%eax  cmp $0x1,%eax  jne 0xf9 <main+249>  call 0xf7 <main+247> | If the value is 1 then Print the value 1 and print “You chose 1” and call DisplayInfo function  If not jump back and reprint the choices for the user |
| jmp 0x109 <main+265>  mov 0x0(%rip),%eax  cmp $0x2,%eax  jne 0x109 <main+265>  call 0x109 <main+265>  mov 0x0(%rip),%eax | If the value if 2 then print the value 2 and print out “You chose 2“ call ChangeCustomerChoice function  If not jump back and reprint the choices |
| cmp $0x3,%eax  je 0x119 <main+281>  jmp 0x4d <main+77> | If the value is 3 then exit the program |
| mov $0x0,%eax  pop %rbp  ret | Return 0 and exit the main proc |

### ChangeCustomerChoice function

| **Assembly Code Block** | **Explanation of Functionality** |
| --- | --- |
| push %rbp  mov %rsp,%rbp  lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x444 <\_Z20ChangeCustomerChoicev+23>  lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x457 <\_Z20ChangeCustomerChoicev+42>  lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x46a <\_Z20ChangeCustomerChoicev+61>  lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x47d <\_Z20ChangeCustomerChoicev+80> | Get the address of “Enter the number of the client that you wish to change”  Print it out  And get the value from your user and store it  Get the address of “Please enter the client’s new service choice (1= Brokerage, 2 = Retirement)  Print it out  And get the value from the user and store it |
| mov 0x0(%rip),%eax  cmp $0x1,%eax  jne 0x496 <\_Z20ChangeCustomerChoicev+105>  mov 0x0(%rip),%eax  mov %eax,0x0(%rip)  jmp 0x4f8 <\_Z20ChangeCustomerChoicev+203>  mov 0x0(%rip),%eax  cmp $0x2,%eax  jne 0x4af <\_Z20ChangeCustomerChoicev+130>  mov 0x0(%rip),%eax  mov %eax,0x0(%rip)  jmp 0x4f8 <\_Z20ChangeCustomerChoicev+203>  mov 0x0(%rip),%eax  cmp $0x3,%eax  jne 0x4c8 <\_Z20ChangeCustomerChoicev+155> |  |
| mov 0x0(%rip),%eax  mov %eax,0x0(%rip)  jmp 0x4f8 <\_Z20ChangeCustomerChoicev+203>  mov 0x0(%rip),%eax  cmp $0x4,%eax  jne 0x4e1 <\_Z20ChangeCustomerChoicev+180>  mov 0x0(%rip),%eax  mov %eax,0x0(%rip)  jmp 0x4f8 <\_Z20ChangeCustomerChoicev+203>  mov 0x0(%rip),%eax  cmp $0x5,%eax  jne 0x4f8 <\_Z20ChangeCustomerChoicev+203> |  |
| mov 0x0(%rip),%eax  mov %eax,0x0(%rip)  nop  pop %rbp  ret |  |

### 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 | Save current base register value and store current stack pointer in the base pointer and store it in the stack and clear EAX register |
| lea -0x45(%rbp),%rax  mov %rax,%rdi  call 0x144 <\_Z25CheckUserPermissionAccessv+36>  lea -0x45(%rbp),%rdx  lea -0x40(%rbp),%rax  lea 0x0(%rip),%rsi  mov %rax,%rdi  call 0x15b <\_Z25CheckUserPermissionAccessv+59>  lea -0x45(%rbp),%rax  mov %rax,%rdi  call 0x167 <\_Z25CheckUserPermissionAccessv+71>  movl $0x0,-0x44(%rbp)  lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x181 <\_Z25CheckUserPermissionAccessv+97>  lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x194 <\_Z25CheckUserPermissionAccessv+116> | Get the address of string “Enter your username:”  And wait for the user to enter a string value.  Get the address of string “Enter your password:”  And wait for the user to get the value.  Get the address of string of “123”  Check the entered value if it’s equal “123” then call  Display |
| lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x1a7 <\_Z25CheckUserPermissionAccessv+135>  lea -0x40(%rbp),%rax  mov %rax,%rsi  lea 0x0(%rip),%rdi  call 0x1ba <\_Z25CheckUserPermissionAccessv+154>  lea -0x40(%rbp),%rax  lea 0x0(%rip),%rsi  mov %rax,%rdi  call 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  call 0x1ee <\_Z25CheckUserPermissionAccessv+206>  mov %ebx,%eax  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  call 0x210 <\_Z25CheckUserPermissionAccessv+240> |  |
| mov %rbx,%rax  mov %rax,%rdi  call 0x21b <\_Z25CheckUserPermissionAccessv+251>  mov %rax,%rbx  lea -0x40(%rbp),%rax  mov %rax,%rdi  call 0x22a <\_Z25CheckUserPermissionAccessv+266>  mov %rbx,%rax  mov %rax,%rdi  call 0x235 <\_Z25CheckUserPermissionAccessv+277> |  |
| call 0x23a <\_Z25CheckUserPermissionAccessv+282>  add $0x48,%rsp  pop %rbx  pop %rbp  ret |  |

### DisplayInfo Function

| **Assembly Code Block** | **Explanation of Functionality** |
| --- | --- |
| push %rbp  mov %rsp,%rbp  lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x258 <\_Z11DisplayInfov+23>  mov %rax,%rdx  mov 0x0(%rip),%rax  mov %rax,%rsi  mov %rdx,%rdi  call 0x26d <\_Z11DisplayInfov+44>  lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x280 <\_Z11DisplayInfov+63>  lea 0x0(%rip),%rsi  mov %rax,%rdi  call 0x28f <\_Z11DisplayInfov+78> |  |
| lea 0x0(%rip),%rsi  mov %rax,%rdi  call 0x29e <\_Z11DisplayInfov+93>  mov %rax,%rdx  mov 0x0(%rip),%eax  mov %eax,%esi  mov %rdx,%rdi  call 0x2b1 <\_Z11DisplayInfov+112> |  |
| mov %rax,%rdx  mov 0x0(%rip),%rax  mov %rax,%rsi  mov %rdx,%rdi  call 0x2c6 <\_Z11DisplayInfov+133>  lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x2d9 <\_Z11DisplayInfov+152>  lea 0x0(%rip),%rsi  mov %rax,%rdi  call 0x2e8 <\_Z11DisplayInfov+167>  lea 0x0(%rip),%rsi  mov %rax,%rdi  call 0x2f7 <\_Z11DisplayInfov+182>  mov %rax,%rdx  mov 0x0(%rip),%eax  mov %eax,%esi  mov %rdx,%rdi  call 0x30a <\_Z11DisplayInfov+201> |  |
|  |  |
| mov %rax,%rdx  mov 0x0(%rip),%rax  mov %rax,%rsi  mov %rdx,%rdi  call 0x31f <\_Z11DisplayInfov+222>  lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x332 <\_Z11DisplayInfov+241>  lea 0x0(%rip),%rsi  mov %rax,%rdi  call 0x341 <\_Z11DisplayInfov+256>  lea 0x0(%rip),%rsi  mov %rax,%rdi  call 0x350 <\_Z11DisplayInfov+271> |  |
|  |  |
| mov %rax,%rdx  mov 0x0(%rip),%eax  mov %eax,%esi  mov %rdx,%rdi  call 0x363 <\_Z11DisplayInfov+290>  mov %rax,%rdx  mov 0x0(%rip),%rax  mov %rax,%rsi  mov %rdx,%rdi  call 0x378 <\_Z11DisplayInfov+311> |  |
| lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x38b <\_Z11DisplayInfov+330>  lea 0x0(%rip),%rsi  mov %rax,%rdi  call 0x39a <\_Z11DisplayInfov+345>  lea 0x0(%rip),%rsi  mov %rax,%rdi  call 0x3a9 <\_Z11DisplayInfov+360> |  |
| mov %rax,%rdx  mov 0x0(%rip),%eax  mov %eax,%esi  mov %rdx,%rdi  call 0x3bc <\_Z11DisplayInfov+379>  mov %rax,%rdx  mov 0x0(%rip),%rax  mov %rax,%rsi  mov %rdx,%rdi  call 0x3d1 <\_Z11DisplayInfov+400> |  |
| lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x3e4 <\_Z11DisplayInfov+419>  lea 0x0(%rip),%rsi  mov %rax,%rdi  call 0x3f3 <\_Z11DisplayInfov+434>  lea 0x0(%rip),%rsi  mov %rax,%rdi  call 0x402 <\_Z11DisplayInfov+449> |  |
| mov %rax,%rdx  mov 0x0(%rip),%eax  mov %eax,%esi  mov %rdx,%rdi  call 0x415 <\_Z11DisplayInfov+468>  mov %rax,%rdx  mov 0x0(%rip),%rax  mov %rax,%rsi  mov %rdx,%rdi  call 0x42a <\_Z11DisplayInfov+489>  nop  pop %rbp  ret |  |