# 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> | This is the start of the program. It pushs the information from %rbp into the stack, then calls 2 functions. |
| mov %eax,0x0(%rip)  mov 0x0(%rip),%eax  cmp $0x1,%eax  je 0x40  lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  callq 0x40 <main+64> | This is when the user is asked for a username and password. $0x1 is compared to the user input at %eax, if the same then it jumps to location 64. In this case it is the username. |
| mov 0x0(%rip),%eax # 0x46 <main+70>  cmp $0x1,%eax  je 0x4d <main+77>  jmp 0x17 <main+23>  lea 0x0(%rip),%rsi # 0x54 <main+84>  lea 0x0(%rip),%rdi # 0x5b <main+91>  callq 0x60 <main+96> | This is where the password is compared to the user input. If successful it jumps to a 77. Password is 123. It compares $0x1 with $eax. |
| 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>  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>  lea 0x0(%rip),%rsi # 0xb3 <main+179>  lea 0x0(%rip),%rdi # 0xba <main+186>  callq 0xbf <main+191> | The address values of %rip are passed through %rsi and %rdi where callq pushes the return address each time throughout the main. |
| mov %rax,%rdx  mov 0x0(%rip),%eax # 0xc8 <main+200>  mov %eax,%esi  mov %rdx,%rdi  callq 0xd2 <main+210> | %rax is moved to %rdx. %rip is moved to %eax which is then moved to %esi. The value in %rdx is moved to %rdi. Callq pushes the return address at main 210. |
|  |  |
| mov %rax,%rdx  mov 0x0(%rip),%rax # 0xdc <main+220>  mov %rax,%rsi  mov %rdx,%rdi  callq 0xe7 <main+231>  mov 0x0(%rip),%eax # 0xed <main+237>  cmp $0x1,%eax  jne 0xf9 <main+249>  callq 0xf7 <main+247> | %rax is moved to %rdx. %rip is moved to %rax. %rax is then moved to %rsi and %rdx is moved to %rdi which a callq pushes the return address at main 231. %rip is moved to %eax where $0x1 is compared to %eax. After comparing, the conditional jump with a test to follow hits at main 249. Callq pushes the return at main 247. |
| jmp 0x109 <main+265>  mov 0x0(%rip),%eax # 0xff <main+255>  cmp $0x2,%eax  jne 0x109 <main+265>  callq 0x109 <main+265>  mov 0x0(%rip),%eax # 0x10f <main+271>  cmp $0x3,%eax  je 0x119 <main+281>  jmpq 0x4d <main+77>  mov $0x0,%eax | Jumps to another location, then 0x0(%rip) is moved to %eax. Then $0x2 is compared to %eax. This then jumps to another location. This then happens again with $0x3 being compared to a new value of %eax that was taken from location 0x0(%rip). %eax is then zeroed out. |
| pop %rbp  retq | Retrieves last valye that was pushed from the stack then ends the program. |
|  |  |
|  |  |

### Check User Permission

| **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  callq 0x144 <\_Z25CheckUserPermissionAccessv+36> | $0x48 is subtracted by $rsp. $rax is moved to –0x18(%rbp). %eax is set to zero with xor. The address value in –0x45(%rbp) is assigned to %rax which is then moved to %rdi which a callq is called to return at 36. |
| lea -0x45(%rbp),%rdx  lea -0x40(%rbp),%rax  lea 0x0(%rip),%rsi # 0x153 <\_Z25CheckUserPermissionAccessv+51>  mov %rax,%rdi  callq 0x15b <\_Z25CheckUserPermissionAccessv+59>  lea -0x45(%rbp),%rax  mov %rax,%rdi  callq 0x167 <\_Z25CheckUserPermissionAccessv+71> | The value of -0x45(%rbp) is assigned to %rdx, then the value of -0x45(%rbp) is assigned to %rax. %rax is then moved into %rdi. A function call is then done to return the value of 59. %rax is moved to rdi then another function call is done for 71. |
| movl $0x0,-0x44(%rbp)  lea 0x0(%rip),%rsi # 0x175 <\_Z25CheckUserPermissionAccessv+85>  lea 0x0(%rip),%rdi # 0x17c <\_Z25CheckUserPermissionAccessv+92>  callq 0x181 <\_Z25CheckUserPermissionAccessv+97>  lea 0x0(%rip),%rsi # 0x188 <\_Z25CheckUserPermissionAccessv+104>  lea 0x0(%rip),%rdi # 0x18f <\_Z25CheckUserPermissionAccessv+111>  callq 0x194 <\_Z25CheckUserPermissionAccessv+116>  lea 0x0(%rip),%rsi # 0x19b <\_Z25CheckUserPermissionAccessv+123>  lea 0x0(%rip),%rdi # 0x1a2 <\_Z25CheckUserPermissionAccessv+130>  callq 0x1a7 <\_Z25CheckUserPermissionAccessv+135>  lea -0x40(%rbp),%rax | $0x0 is copied into -0x44(%rbp). Then it is copied it into multiple different variables across the program. |
| mov %rax,%rsi  lea 0x0(%rip),%rdi # 0x1b5 <\_Z25CheckUserPermissionAccessv+149>  callq 0x1ba <\_Z25CheckUserPermissionAccessv+154>  lea -0x40(%rbp),%rax  lea 0x0(%rip),%rsi # 0x1c5 <\_Z25CheckUserPermissionAccessv+165>  mov %rax,%rdi  callq 0x1cd <\_Z25CheckUserPermissionAccessv+173> | %rax is then moved into %rsi. Then function is called to produce value 154. -0x40(%rbp) is assigned to %rax and 0x0(%rip) is assigned to %rsi. %rax is then moved to %rdi. Lastly a function is called that produces value 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 | %eax is moved to -0x44(%rbp). $0x0 is compared to 0x44(%rbp) and a conditional jump and test are performed. $0x1 is moved to %ebx and an unconditional jump is performed. $0x2 is moved to %ebx. The address value at –0x40(%rbp) is moved to %rax and then moved to $rdi. A callq is called and %ebx is moved to %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  callq 0x210 <\_Z25CheckUserPermissionAccessv+240> | -0x18(%rbp) is moved to %rxc. A jump is then done to another location. %rax is then moved to %rbx. %rax is then moved to %rdi. A function is then called for value 240. |
| mov %rbx,%rax  mov %rax,%rdi  callq 0x21b <\_Z25CheckUserPermissionAccessv+251>  mov %rax,%rbx  lea -0x40(%rbp),%rax  mov %rax,%rdi  callq 0x22a <\_Z25CheckUserPermissionAccessv+266>  mov %rbx,%rax  mov %rax,%rdi  callq 0x235 <\_Z25CheckUserPermissionAccessv+277>  callq 0x23a <\_Z25CheckUserPermissionAccessv+282>  add $0x48,%rsp | %rbx is moved into %rax while %rax is then moved into %rdi. A call is done for value 251. %rax is then moved to %rbx. A call is then done for value 266. %rbx is then moved to %rax then %rax is then moved to %rdi. 2 calls for values 277 and 282 are done. Value of $0x48 is added with %rsp. |
| pop %rbx  pop %rbp  retq | Values are stored in %rbx and %rbp then the program is exited. |

### Display Info

| **Assembly Code Block** | **Explanation of Functionality** |
| --- | --- |
| push %rbp  mov %rsp,%rbp  lea 0x0(%rip),%rsi # 0x24c <\_Z11DisplayInfov+11>  lea 0x0(%rip),%rdi # 0x253 <\_Z11DisplayInfov+18>  callq 0x258 <\_Z11DisplayInfov+23>  mov %rax,%rdx  mov 0x0(%rip),%rax # 0x262 <\_Z11DisplayInfov+33>  mov %rax,%rsi  mov %rdx,%rdi  callq 0x26d <\_Z11DisplayInfov+44> | %rip is assigned to %rsi and %rdi and a callq is called to return the values at 23. %rax is moved to %rdx and %rip is moved to %rax and then moved to %rsi. %rdx is moved to %rdi. Callq is called to return the value at 44. |
| lea 0x0(%rip),%rsi # 0x274 <\_Z11DisplayInfov+51>  lea 0x0(%rip),%rdi # 0x27b <\_Z11DisplayInfov+58>  callq 0x280 <\_Z11DisplayInfov+63>  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> | 0x0(%rip) is assigned to both %rsi and %rdi. Call is then called for value 63. %rax is moved to %rdi and another call is done for value 78. %rax is then moved to %rdi and then another call is initiated for 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> | %rax is moved to %rdx, then %eax is moved to %esi. %rdx is then movied to %rdi. There is a call for value 112. %rax is then moved to %rdx. Value at 0x0(%rip) is then movied to %rax. %rax is then moved to %rsi while %rdx is movied to %rdi. There is then a call for value 133. |
| 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> | 0x0(%rip) is assigtned to both %rsi and %rdi. A call then is done to return 152. %rax is then moved to %rdi. Another call is done to return 167. %rax is then moved to %rdi. A call is done to return 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> | %rax is moved to %rdx. 0x0(%rip) is then movied into %eax. %eax is moved to %esi while %rdx is movied to %rdi. A call is done to return 201. %rax is then movied to %rdx. The content of 0x0(%rip) is then movied to %rax. Another call is done to return 211. %rax is moved into %rsi and %rdx is movied to %rdi. A call is done to return 222. |
| lea 0x0(%rip),%rsi # 0x326 <\_Z11DisplayInfov+229>  lea 0x0(%rip),%rdi # 0x32d <\_Z11DisplayInfov+236>  callq 0x332 <\_Z11DisplayInfov+241>  lea 0x0(%rip),%rsi # 0x339 <\_Z11DisplayInfov+248>  mov %rax,%rdi  callq 0x341 <\_Z11DisplayInfov+256>  lea 0x0(%rip),%rsi # 0x348 <\_Z11DisplayInfov+263>  mov %rax,%rdi  callq 0x350 <\_Z11DisplayInfov+271> | 0x0(%rip) is assigned to %rsi and %rdi.A call is done to return 241.0x0(%rip) is then assigned to %rsi. %rax is then moved to %rdi. A call is done to return 256. 0x0(%rip) is assigned to %rsi. %rax is then moved to %rdi. A call is done to return 271. |
| mov %rax,%rdx  mov 0x0(%rip),%eax # 0x359 <\_Z11DisplayInfov+280>  mov %eax,%esi  mov %rdx,%rdi  callq 0x363 <\_Z11DisplayInfov+290>  mov %rax,%rdx  mov 0x0(%rip),%rax # 0x36d <\_Z11DisplayInfov+300>  mov %rax,%rsi  mov %rdx,%rdi  callq 0x378 <\_Z11DisplayInfov+311> | %rax is movied to %rdx while 0x0(%rip) is movied to %eax. %eax is then moved to %esi and %rdx is movied to %rdi. A call is done to return the value 290. %rax is then movied to %rdx and 0x0(%rip) is moved to %rax.  %rax is moved to %rsi and %rdx is moved to %rdi. A call is then done to return 311. |
| lea 0x0(%rip),%rsi # 0x37f <\_Z11DisplayInfov+318>  lea 0x0(%rip),%rdi # 0x386 <\_Z11DisplayInfov+325>  callq 0x38b <\_Z11DisplayInfov+330>  lea 0x0(%rip),%rsi # 0x392 <\_Z11DisplayInfov+337>  mov %rax,%rdi  callq 0x39a <\_Z11DisplayInfov+345>  lea 0x0(%rip),%rsi # 0x3a1 <\_Z11DisplayInfov+352>  mov %rax,%rdi  callq 0x3a9 <\_Z11DisplayInfov+360> | 0x0(%rip) is assigned to %rsi and %rdi. A call is done for 330 and 0x0(%rip) is assigned to %rsi again. Another call is done to return 345 and 0x0(%rip) is assigned to %rsi again. Another call is done for 345 and 0x0(%rip) is assigned to %rsi for a third time. %rax is movied to %rdi. A call is done to return 360. |
| mov %rax,%rdx  mov 0x0(%rip),%eax # 0x3b2 <\_Z11DisplayInfov+369>  mov %eax,%esi  mov %rdx,%rdi  callq 0x3bc <\_Z11DisplayInfov+379>  mov %rax,%rdx  mov 0x0(%rip),%rax # 0x3c6 <\_Z11DisplayInfov+389>  mov %rax,%rsi  mov %rdx,%rdi  callq 0x3d1 <\_Z11DisplayInfov+400> | %rax is moved to %rdx and 0x0(%rip) is moved to %eax. %eax is moved to %esi and $rdx is moved to %rdi. A call is done to return 379. %rax is moved to %rdx and 0x0(%rip) is moved to %rax. %rax is then moved to %rsi and %rdx is moved to %rdi. A call is done to return 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> | 0x0(%rip) is assigned to %rsi and %rdi. A call is done to return 419. 0x0(%rip) is then assigned to rsi.  %rax is moved to %rdi. A call is done to return 434. 0x0(%rip) is assigned to %rsi. %rax is moved to %rdi. Another call is done to return 449. |
| 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> | %rax is moved to %rdx. 0x0(%rip) is then moved to %eax. %eax is moved to %esi. %rdx is moved to %rdi. A call is done to return 468. %rax is movied to rdx and 0x0(%rip) is moved to %rax. %rax is moved to %rsi and %rdx is moved to %rdi. A call is done to return 489. |
| nop  pop %rbp  retq | Value is stored at %rbp and program ends. |

### ChangeCustomerChoice

| **Assembly Code Block** | **Explanation of Functionality** |
| --- | --- |
| push %rbp  mov %rsp,%rbp  lea 0x0(%rip),%rsi # 0x438 <\_Z20ChangeCustomerChoicev+11>  lea 0x0(%rip),%rdi # 0x43f <\_Z20ChangeCustomerChoicev+18>  callq 0x444 <\_Z20ChangeCustomerChoicev+23>  lea 0x0(%rip),%rsi # 0x44b <\_Z20ChangeCustomerChoicev+30>  lea 0x0(%rip),%rdi # 0x452 <\_Z20ChangeCustomerChoicev+37>  callq 0x457 <\_Z20ChangeCustomerChoicev+42> | Push %rbp into the stack. Move %rsp to %rbp. 0x0(%rip) is assigned to %rsi and %rdi. A Call is then done to return 23. 0x0(%rip) is assigned to %rsi and %rdi. A call is then done to return 42. |
| lea 0x0(%rip),%rsi # 0x45e <\_Z20ChangeCustomerChoicev+49>  lea 0x0(%rip),%rdi # 0x465 <\_Z20ChangeCustomerChoicev+56>  callq 0x46a <\_Z20ChangeCustomerChoicev+61>  lea 0x0(%rip),%rsi # 0x471 <\_Z20ChangeCustomerChoicev+68>  lea 0x0(%rip),%rdi # 0x478 <\_Z20ChangeCustomerChoicev+75>  callq 0x47d <\_Z20ChangeCustomerChoicev+80>  mov 0x0(%rip),%eax # 0x483 <\_Z20ChangeCustomerChoicev+86> | 0x0(%rip) is assigned to %rsi and %rdi. A call is done to return 61. 0x0(%rip) is assigned to %rsi and %rdi. A call is then done to return 80. 0x0(%rip) is moved to %eax. |
| 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>  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>  mov 0x0(%rip),%eax # 0x4b5 <\_Z20ChangeCustomerChoicev+136>  cmp $0x3,%eax  jne 0x4c8 <\_Z20ChangeCustomerChoicev+155> | $0x1 is compared to %eax. Jumps to 105. Move 0x0(%rip) to %eax and moves %eax into 0x0(%rip). Jumps to 203. Moves 0x0(%rip) to %eax. Compares $0x2 to %eax. Junos to 130. Moves 0x0(%rip) to %eax and moves %eax to 0x0(%rip). Jumps to 203 and moves 0x0(%rip) to %eax. Compares $0x3 to %eax then jumps to 155. |
| mov 0x0(%rip),%eax # 0x4c0 <\_Z20ChangeCustomerChoicev+147>  mov %eax,0x0(%rip) # 0x4c6 <\_Z20ChangeCustomerChoicev+153>  jmp 0x4f8 <\_Z20ChangeCustomerChoicev+203>  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>  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> | Moves 0x0(%rip) to %eax and moves %eax to 0x0(%rip). Jumps to 203 and moves 0x0(%rip) to %eax. Compares $0x4 to %eax. Jumps to 180 and moves 0x0(%rip) to %eax. 0x0(%rip) is then moved into %eax and %eax is moved into 0x0(%rip). Jumps to 203 and 0x0(%rip) is moved to %eax. $0x5 is compared with %eax. Jumps to 203 and 0x0(%rip) is moved to %eax. %eax is moved to 0x0(%rip). |
| nop  pop %rbp  retq | Nop does nothing but the rest Retrieves the last value pushed from the stack and ends the program. |