# CS 410 Project One Proficiency Test Template

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

### **“main” function”**

Assembly Code Block

0000000000000000 <main>:

0: 55 push %rbp

1: 48 89 e5 mov %rsp,%rbp

4: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # b <main+0xb>

b: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 12 <main+0x12>

12: e8 00 00 00 00 call 17 <main+0x17>

17: e8 00 00 00 00 call 1c <main+0x1c>

1c: 89 05 00 00 00 00 mov %eax,0x0(%rip) # 22 <main+0x22>

22: 8b 05 00 00 00 00 mov 0x0(%rip),%eax # 28 <main+0x28>

28: 83 f8 01 cmp $0x1,%eax

2b: 74 13 je 40 <main+0x40>

2d: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 34 <main+0x34>

34: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 3b <main+0x3b>

3b: e8 00 00 00 00 call 40 <main+0x40>

40: 8b 05 00 00 00 00 mov 0x0(%rip),%eax # 46 <main+0x46>

46: 83 f8 01 cmp $0x1,%eax

49: 74 02 je 4d <main+0x4d>

4b: eb ca jmp 17 <main+0x17>

4d: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 54 <main+0x54>

54: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 5b <main+0x5b>

5b: e8 00 00 00 00 call 60 <main+0x60>

60: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 67 <main+0x67>

67: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 6e <main+0x6e>

6e: e8 00 00 00 00 call 73 <main+0x73>

73: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 7a <main+0x7a>

7a: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 81 <main+0x81>

81: e8 00 00 00 00 call 86 <main+0x86>

86: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 8d <main+0x8d>

8d: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 94 <main+0x94>

94: e8 00 00 00 00 call 99 <main+0x99>

99: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # a0 <main+0xa0>

a0: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # a7 <main+0xa7>

a7: e8 00 00 00 00 call ac <main+0xac>

ac: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # b3 <main+0xb3>

b3: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # ba <main+0xba>

ba: e8 00 00 00 00 call bf <main+0xbf>

bf: 48 89 c2 mov %rax,%rdx

c2: 8b 05 00 00 00 00 mov 0x0(%rip),%eax # c8 <main+0xc8>

c8: 89 c6 mov %eax,%esi

ca: 48 89 d7 mov %rdx,%rdi

cd: e8 00 00 00 00 call d2 <main+0xd2>

d2: 48 89 c2 mov %rax,%rdx

d5: 48 8b 05 00 00 00 00 mov 0x0(%rip),%rax # dc <main+0xdc>

dc: 48 89 c6 mov %rax,%rsi

df: 48 89 d7 mov %rdx,%rdi

e2: e8 00 00 00 00 call e7 <main+0xe7>

e7: 8b 05 00 00 00 00 mov 0x0(%rip),%eax # ed <main+0xed>

ed: 83 f8 01 cmp $0x1,%eax

f0: 75 07 jne f9 <main+0xf9>

f2: e8 00 00 00 00 call f7 <main+0xf7>

f7: eb 10 jmp 109 <main+0x109>

f9: 8b 05 00 00 00 00 mov 0x0(%rip),%eax # ff <main+0xff>

ff: 83 f8 02 cmp $0x2,%eax

102: 75 05 jne 109 <main+0x109>

104: e8 00 00 00 00 call 109 <main+0x109>

109: 8b 05 00 00 00 00 mov 0x0(%rip),%eax # 10f <main+0x10f>

10f: 83 f8 03 cmp $0x3,%eax

112: 74 05 je 119 <main+0x119>

114: e9 34 ff ff ff jmp 4d <main+0x4d>

119: b8 00 00 00 00 mov $0x0,%eax

11e: 5d pop %rbp

11f: c3 ret

Explanation of Functionality

makes use of conditional jumps based on values in the %eax register and several function calls. The assembly instructions and general structure suggest that it is a loop that calls functions one after the other in order, making decisions in response to the value of %eax. Depending on whether %eax contains the value 1, 2, or 3, the software does different actions. Here is a high-level understanding of what the function may be doing: First Setup: %rbp is pushed onto the stack and the base pointer (%rbp) is initialized by the function. Addresses are loaded into registers using a number of lea instructions, presumably as function parameters for subsequent calls. Loop with Conditional Jumps and Function Calls: The loop calls different functions according on the value of %eax. At various places in the code, the value of %eax is compared to 1, 2, or 3, and based on the comparison, other functions are called. The loop runs until %eax is equal to 3, at which time the function sets %eax to 0 and returns. This is the exit condition.

### **ChangeCustomerChoice function**

Assembly Code Block

000000000000042d <\_Z20ChangeCustomerChoicev>:

42d: 55 push %rbp

42e: 48 89 e5 mov %rsp,%rbp

431: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 438 <\_Z20ChangeCustomerChoicev+0xb>

438: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 43f <\_Z20ChangeCustomerChoicev+0x12>

43f: e8 00 00 00 00 call 444 <\_Z20ChangeCustomerChoicev+0x17>

444: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 44b <\_Z20ChangeCustomerChoicev+0x1e>

44b: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 452 <\_Z20ChangeCustomerChoicev+0x25>

452: e8 00 00 00 00 call 457 <\_Z20ChangeCustomerChoicev+0x2a>

457: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 45e <\_Z20ChangeCustomerChoicev+0x31>

45e: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 465 <\_Z20ChangeCustomerChoicev+0x38>

465: e8 00 00 00 00 call 46a <\_Z20ChangeCustomerChoicev+0x3d>

46a: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 471 <\_Z20ChangeCustomerChoicev+0x44>

471: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 478 <\_Z20ChangeCustomerChoicev+0x4b>

478: e8 00 00 00 00 call 47d <\_Z20ChangeCustomerChoicev+0x50>

47d: 8b 05 00 00 00 00 mov 0x0(%rip),%eax # 483 <\_Z20ChangeCustomerChoicev+0x56>

483: 83 f8 01 cmp $0x1,%eax

486: 75 0e jne 496 <\_Z20ChangeCustomerChoicev+0x69>

488: 8b 05 00 00 00 00 mov 0x0(%rip),%eax # 48e <\_Z20ChangeCustomerChoicev+0x61>

48e: 89 05 00 00 00 00 mov %eax,0x0(%rip) # 494 <\_Z20ChangeCustomerChoicev+0x67>

494: eb 62 jmp 4f8 <\_Z20ChangeCustomerChoicev+0xcb>

496: 8b 05 00 00 00 00 mov 0x0(%rip),%eax # 49c <\_Z20ChangeCustomerChoicev+0x6f>

49c: 83 f8 02 cmp $0x2,%eax

49f: 75 0e jne 4af <\_Z20ChangeCustomerChoicev+0x82>

4a1: 8b 05 00 00 00 00 mov 0x0(%rip),%eax # 4a7 <\_Z20ChangeCustomerChoicev+0x7a>

4a7: 89 05 00 00 00 00 mov %eax,0x0(%rip) # 4ad <\_Z20ChangeCustomerChoicev+0x80>

4ad: eb 49 jmp 4f8 <\_Z20ChangeCustomerChoicev+0xcb>

4af: 8b 05 00 00 00 00 mov 0x0(%rip),%eax # 4b5 <\_Z20ChangeCustomerChoicev+0x88>

4b5: 83 f8 03 cmp $0x3,%eax

4b8: 75 0e jne 4c8 <\_Z20ChangeCustomerChoicev+0x9b>

4ba: 8b 05 00 00 00 00 mov 0x0(%rip),%eax # 4c0 <\_Z20ChangeCustomerChoicev+0x93>

4c0: 89 05 00 00 00 00 mov %eax,0x0(%rip) # 4c6 <\_Z20ChangeCustomerChoicev+0x99>

4c6: eb 30 jmp 4f8 <\_Z20ChangeCustomerChoicev+0xcb>

4c8: 8b 05 00 00 00 00 mov 0x0(%rip),%eax # 4ce <\_Z20ChangeCustomerChoicev+0xa1>

4ce: 83 f8 04 cmp $0x4,%eax

4d1: 75 0e jne 4e1 <\_Z20ChangeCustomerChoicev+0xb4>

4d3: 8b 05 00 00 00 00 mov 0x0(%rip),%eax # 4d9 <\_Z20ChangeCustomerChoicev+0xac>

4d9: 89 05 00 00 00 00 mov %eax,0x0(%rip) # 4df <\_Z20ChangeCustomerChoicev+0xb2>

4df: eb 17 jmp 4f8 <\_Z20ChangeCustomerChoicev+0xcb>

4e1: 8b 05 00 00 00 00 mov 0x0(%rip),%eax # 4e7 <\_Z20ChangeCustomerChoicev+0xba>

4e7: 83 f8 05 cmp $0x5,%eax

4ea: 75 0c jne 4f8 <\_Z20ChangeCustomerChoicev+0xcb>

4ec: 8b 05 00 00 00 00 mov 0x0(%rip),%eax # 4f2 <\_Z20ChangeCustomerChoicev+0xc5>

4f2: 89 05 00 00 00 00 mov %eax,0x0(%rip) # 4f8 <\_Z20ChangeCustomerChoicev+0xcb>

4f8: 90 nop

4f9: 5d pop %rbp

4fa: c3 ret

Explanation of Functionality

It appears to have several leaps and comparisons depending on the values in the %eax register. It probably alters or adapts a consumer decision by contacting different sections based on the results of many condition checks. Breakdown of Functionality: The function evaluates %eax using values between 1 and 5, doing distinct actions in response to each comparison. A similar operation is taken by the program when %eax matches 1, 2, 3, 4, or 5. It moves the %eax value and may set it or make further modifications before returning. It looks that the function is handling options or choices from the user, verifying the value of %eax, and branching appropriately.

### **CheckUserPermissonAccess Function**

Assembly Code Block

0000000000000120 <\_Z25CheckUserPermissionAccessv>:

120: 55 push %rbp

121: 48 89 e5 mov %rsp,%rbp

124: 53 push %rbx

125: 48 83 ec 48 sub $0x48,%rsp

129: 64 48 8b 04 25 28 00 mov %fs:0x28,%rax

130: 00 00

132: 48 89 45 e8 mov %rax,-0x18(%rbp)

136: 31 c0 xor %eax,%eax

138: 48 8d 45 bb lea -0x45(%rbp),%rax

13c: 48 89 c7 mov %rax,%rdi

13f: e8 00 00 00 00 call 144 <\_Z25CheckUserPermissionAccessv+0x24>

144: 48 8d 55 bb lea -0x45(%rbp),%rdx

148: 48 8d 45 c0 lea -0x40(%rbp),%rax

14c: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 153 <\_Z25CheckUserPermissionAccessv+0x33>

153: 48 89 c7 mov %rax,%rdi

156: e8 00 00 00 00 call 15b <\_Z25CheckUserPermissionAccessv+0x3b>

15b: 48 8d 45 bb lea -0x45(%rbp),%rax

15f: 48 89 c7 mov %rax,%rdi

162: e8 00 00 00 00 call 167 <\_Z25CheckUserPermissionAccessv+0x47>

167: c7 45 bc 00 00 00 00 movl $0x0,-0x44(%rbp)

16e: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 175 <\_Z25CheckUserPermissionAccessv+0x55>

175: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 17c <\_Z25CheckUserPermissionAccessv+0x5c>

17c: e8 00 00 00 00 call 181 <\_Z25CheckUserPermissionAccessv+0x61>

181: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 188 <\_Z25CheckUserPermissionAccessv+0x68>

188: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 18f <\_Z25CheckUserPermissionAccessv+0x6f>

18f: e8 00 00 00 00 call 194 <\_Z25CheckUserPermissionAccessv+0x74>

194: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 19b <\_Z25CheckUserPermissionAccessv+0x7b>

19b: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 1a2 <\_Z25CheckUserPermissionAccessv+0x82>

1a2: e8 00 00 00 00 call 1a7 <\_Z25CheckUserPermissionAccessv+0x87>

1a7: 48 8d 45 c0 lea -0x40(%rbp),%rax

1ab: 48 89 c6 mov %rax,%rsi

1ae: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 1b5 <\_Z25CheckUserPermissionAccessv+0x95>

1b5: e8 00 00 00 00 call 1ba <\_Z25CheckUserPermissionAccessv+0x9a>

1ba: 48 8d 45 c0 lea -0x40(%rbp),%rax

1be: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 1c5 <\_Z25CheckUserPermissionAccessv+0xa5>

1c5: 48 89 c7 mov %rax,%rdi

1c8: e8 00 00 00 00 call 1cd <\_Z25CheckUserPermissionAccessv+0xad>

1cd: 89 45 bc mov %eax,-0x44(%rbp)

1d0: 83 7d bc 00 cmpl $0x0,-0x44(%rbp)

1d4: 75 07 jne 1dd <\_Z25CheckUserPermissionAccessv+0xbd>

1d6: bb 01 00 00 00 mov $0x1,%ebx

1db: eb 05 jmp 1e2 <\_Z25CheckUserPermissionAccessv+0xc2>

1dd: bb 02 00 00 00 mov $0x2,%ebx

1e2: 48 8d 45 c0 lea -0x40(%rbp),%rax

1e6: 48 89 c7 mov %rax,%rdi

1e9: e8 00 00 00 00 call 1ee <\_Z25CheckUserPermissionAccessv+0xce>

1ee: 89 d8 mov %ebx,%eax

1f0: 48 8b 4d e8 mov -0x18(%rbp),%rcx

1f4: 64 48 33 0c 25 28 00 xor %fs:0x28,%rcx

1fb: 00 00

1fd: 74 3b je 23a <\_Z25CheckUserPermissionAccessv+0x11a>

1ff: eb 34 jmp 235 <\_Z25CheckUserPermissionAccessv+0x115>

201: 48 89 c3 mov %rax,%rbx

204: 48 8d 45 bb lea -0x45(%rbp),%rax

208: 48 89 c7 mov %rax,%rdi

20b: e8 00 00 00 00 call 210 <\_Z25CheckUserPermissionAccessv+0xf0>

210: 48 89 d8 mov %rbx,%rax

213: 48 89 c7 mov %rax,%rdi

216: e8 00 00 00 00 call 21b <\_Z25CheckUserPermissionAccessv+0xfb>

21b: 48 89 c3 mov %rax,%rbx

21e: 48 8d 45 c0 lea -0x40(%rbp),%rax

222: 48 89 c7 mov %rax,%rdi

225: e8 00 00 00 00 call 22a <\_Z25CheckUserPermissionAccessv+0x10a>

22a: 48 89 d8 mov %rbx,%rax

22d: 48 89 c7 mov %rax,%rdi

230: e8 00 00 00 00 call 235 <\_Z25CheckUserPermissionAccessv+0x115>

235: e8 00 00 00 00 call 23a <\_Z25CheckUserPermissionAccessv+0x11a>

23a: 48 83 c4 48 add $0x48,%rsp

23e: 5b pop %rbx

23f: 5d pop %rbp

240: c3 ret

Explanation of Functionality

implies that it makes use of function calls and memory access in several phases. Additionally, it processes data using a stack-based structure, which may have to do with verifying user rights and providing various outcomes depending on the permission check. High-Level Capabilities: Setup: The function uses the fs segment register to access thread-local storage and sets up the stack. It also probably retrieves some data for processing. Memory Manipulation and Function Calls: The function appears to handle certain data structures or permissions based on the several lea instructions that follow. Checking Conditions: The function appears to verify a given condition using a comparison (e.g., cmpl $0x0,-0x44(%rbp)) and then sets %ebx based on the result (e.g., mov $0x1,%ebx or mov $0x2,%ebx), indicating that multiple permission levels could be involved. Final Cleanup: After cleaning up a little, the method returns.

### **DisplayInfo Function**

Assembly Code Block

0000000000000241 <\_Z11DisplayInfov>:

241: 55 push %rbp

242: 48 89 e5 mov %rsp,%rbp

245: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 24c <\_Z11DisplayInfov+0xb>

24c: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 253 <\_Z11DisplayInfov+0x12>

253: e8 00 00 00 00 call 258 <\_Z11DisplayInfov+0x17>

258: 48 89 c2 mov %rax,%rdx

25b: 48 8b 05 00 00 00 00 mov 0x0(%rip),%rax # 262 <\_Z11DisplayInfov+0x21>

262: 48 89 c6 mov %rax,%rsi

265: 48 89 d7 mov %rdx,%rdi

268: e8 00 00 00 00 call 26d <\_Z11DisplayInfov+0x2c>

26d: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 274 <\_Z11DisplayInfov+0x33>

274: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 27b <\_Z11DisplayInfov+0x3a>

27b: e8 00 00 00 00 call 280 <\_Z11DisplayInfov+0x3f>

280: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 287 <\_Z11DisplayInfov+0x46>

287: 48 89 c7 mov %rax,%rdi

28a: e8 00 00 00 00 call 28f <\_Z11DisplayInfov+0x4e>

28f: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 296 <\_Z11DisplayInfov+0x55>

296: 48 89 c7 mov %rax,%rdi

299: e8 00 00 00 00 call 29e <\_Z11DisplayInfov+0x5d>

29e: 48 89 c2 mov %rax,%rdx

2a1: 8b 05 00 00 00 00 mov 0x0(%rip),%eax # 2a7 <\_Z11DisplayInfov+0x66>

2a7: 89 c6 mov %eax,%esi

2a9: 48 89 d7 mov %rdx,%rdi

2ac: e8 00 00 00 00 call 2b1 <\_Z11DisplayInfov+0x70>

2b1: 48 89 c2 mov %rax,%rdx

2b4: 48 8b 05 00 00 00 00 mov 0x0(%rip),%rax # 2bb <\_Z11DisplayInfov+0x7a>

2bb: 48 89 c6 mov %rax,%rsi

2be: 48 89 d7 mov %rdx,%rdi

2c1: e8 00 00 00 00 call 2c6 <\_Z11DisplayInfov+0x85>

2c6: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 2cd <\_Z11DisplayInfov+0x8c>

2cd: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 2d4 <\_Z11DisplayInfov+0x93>

2d4: e8 00 00 00 00 call 2d9 <\_Z11DisplayInfov+0x98>

2d9: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 2e0 <\_Z11DisplayInfov+0x9f>

2e0: 48 89 c7 mov %rax,%rdi

2e3: e8 00 00 00 00 call 2e8 <\_Z11DisplayInfov+0xa7>

2e8: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 2ef <\_Z11DisplayInfov+0xae>

2ef: 48 89 c7 mov %rax,%rdi

2f2: e8 00 00 00 00 call 2f7 <\_Z11DisplayInfov+0xb6>

2f7: 48 89 c2 mov %rax,%rdx

2fa: 8b 05 00 00 00 00 mov 0x0(%rip),%eax # 300 <\_Z11DisplayInfov+0xbf>

300: 89 c6 mov %eax,%esi

302: 48 89 d7 mov %rdx,%rdi

305: e8 00 00 00 00 call 30a <\_Z11DisplayInfov+0xc9>

30a: 48 89 c2 mov %rax,%rdx

30d: 48 8b 05 00 00 00 00 mov 0x0(%rip),%rax # 314 <\_Z11DisplayInfov+0xd3>

314: 48 89 c6 mov %rax,%rsi

317: 48 89 d7 mov %rdx,%rdi

31a: e8 00 00 00 00 call 31f <\_Z11DisplayInfov+0xde>

31f: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 326 <\_Z11DisplayInfov+0xe5>

326: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 32d <\_Z11DisplayInfov+0xec>

32d: e8 00 00 00 00 call 332 <\_Z11DisplayInfov+0xf1>

332: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 339 <\_Z11DisplayInfov+0xf8>

339: 48 89 c7 mov %rax,%rdi

33c: e8 00 00 00 00 call 341 <\_Z11DisplayInfov+0x100>

341: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 348 <\_Z11DisplayInfov+0x107>

348: 48 89 c7 mov %rax,%rdi

34b: e8 00 00 00 00 call 350 <\_Z11DisplayInfov+0x10f>

350: 48 89 c2 mov %rax,%rdx

353: 8b 05 00 00 00 00 mov 0x0(%rip),%eax # 359 <\_Z11DisplayInfov+0x118>

359: 89 c6 mov %eax,%esi

35b: 48 89 d7 mov %rdx,%rdi

35e: e8 00 00 00 00 call 363 <\_Z11DisplayInfov+0x122>

363: 48 89 c2 mov %rax,%rdx

366: 48 8b 05 00 00 00 00 mov 0x0(%rip),%rax # 36d <\_Z11DisplayInfov+0x12c>

36d: 48 89 c6 mov %rax,%rsi

370: 48 89 d7 mov %rdx,%rdi

373: e8 00 00 00 00 call 378 <\_Z11DisplayInfov+0x137>

378: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 37f <\_Z11DisplayInfov+0x13e>

37f: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 386 <\_Z11DisplayInfov+0x145>

386: e8 00 00 00 00 call 38b <\_Z11DisplayInfov+0x14a>

38b: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 392 <\_Z11DisplayInfov+0x151>

392: 48 89 c7 mov %rax,%rdi

395: e8 00 00 00 00 call 39a <\_Z11DisplayInfov+0x159>

39a: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 3a1 <\_Z11DisplayInfov+0x160>

3a1: 48 89 c7 mov %rax,%rdi

3a4: e8 00 00 00 00 call 3a9 <\_Z11DisplayInfov+0x168>

3a9: 48 89 c2 mov %rax,%rdx

3ac: 8b 05 00 00 00 00 mov 0x0(%rip),%eax # 3b2 <\_Z11DisplayInfov+0x171>

3b2: 89 c6 mov %eax,%esi

3b4: 48 89 d7 mov %rdx,%rdi

3b7: e8 00 00 00 00 call 3bc <\_Z11DisplayInfov+0x17b>

3bc: 48 89 c2 mov %rax,%rdx

3bf: 48 8b 05 00 00 00 00 mov 0x0(%rip),%rax # 3c6 <\_Z11DisplayInfov+0x185>

3c6: 48 89 c6 mov %rax,%rsi

3c9: 48 89 d7 mov %rdx,%rdi

3cc: e8 00 00 00 00 call 3d1 <\_Z11DisplayInfov+0x190>

3d1: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 3d8 <\_Z11DisplayInfov+0x197>

3d8: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 3df <\_Z11DisplayInfov+0x19e>

3df: e8 00 00 00 00 call 3e4 <\_Z11DisplayInfov+0x1a3>

3e4: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 3eb <\_Z11DisplayInfov+0x1aa>

3eb: 48 89 c7 mov %rax,%rdi

3ee: e8 00 00 00 00 call 3f3 <\_Z11DisplayInfov+0x1b2>

3f3: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 3fa <\_Z11DisplayInfov+0x1b9>

3fa: 48 89 c7 mov %rax,%rdi

3fd: e8 00 00 00 00 call 402 <\_Z11DisplayInfov+0x1c1>

402: 48 89 c2 mov %rax,%rdx

405: 8b 05 00 00 00 00 mov 0x0(%rip),%eax # 40b <\_Z11DisplayInfov+0x1ca>

40b: 89 c6 mov %eax,%esi

40d: 48 89 d7 mov %rdx,%rdi

410: e8 00 00 00 00 call 415 <\_Z11DisplayInfov+0x1d4>

415: 48 89 c2 mov %rax,%rdx

418: 48 8b 05 00 00 00 00 mov 0x0(%rip),%rax # 41f <\_Z11DisplayInfov+0x1de>

41f: 48 89 c6 mov %rax,%rsi

422: 48 89 d7 mov %rdx,%rdi

425: e8 00 00 00 00 call 42a <\_Z11DisplayInfov+0x1e9>

42a: 90 nop

42b: 5d pop %rbp

42c: c3 ret

Explanation of Functionality

seems to be engaged in the gathering, analyzing, and presentation of data. A series of actions involving data being transferred between registers and supplied to functions is shown by the numerous function calls and register manipulations. High-Level Capabilities: Setup: To initialize the stack frame, the method pushes %rbp onto the stack. Memory Operations and Function Calls: The code appears to alter data structures or carry out operations pertaining to information presentation since it has several lea instructions to load addresses into registers and then calls functions. Data Movement: Values that are transferred between %rax, %rdx, and %rsi most likely correspond to data that the called functions are passing and displaying. Loop of Function Calls: A series of related function calls that feed into one another as a consequence show a step-by-step procedure for obtaining and displaying data.