ICS EXE 4

Fall, 2023

**Suppose all the following codes are running on a little-ending x86-64 machine.*

1. Jump Table

Read the C code and the assembly code below, answer the following questions.

```
1.
2.  #define BUF_SIZE 8

3.  void transfer(char *buf) {
4.    int i;
5.    for (i = 0; i < BUF_SIZE; i++) {
6.    switch (buf[i]) {
7.      case 0x1f:
8.      if (i > 0) {
9.        buf[i] += buf[i-1];
10.     }
11.     case 0x20: case 0x21:
12.       buf[i] |= 0xf0;
13.       break;
14.     case 0x22: case 0x25:
15.       *((unsigned *)buf + i/4) = (unsigned)buf[i];
16.       break;
17.     case 0x24:
18.       buf[i] = (char)((*(long *)buf) >> 8);
19.       break;
20.     default:
21.       if (i < BUF_SIZE - 1) {
22.         *(short *)buf[i] = 0x201f;
23.       }
24.       break;
25.     }
26.   }
27. }
```

```
1.  transfer:
2.      pushq     %rbp
3.      movq      %rsp, %rbp
4.      movq      %rdi, -24(%rbp)
5.      movl      $0, -4(%rbp)
6.      jmp  .L2
7.  .L11:
8.      movl      -4(%rbp), %eax
9.      movslq    %eax, %rdx
10.     movq      -24(%rbp), %rax
11.     addq      %rdx, %rax
12.     movzbl    (%rax), %eax
13.     movsbl    %al, %eax
14.     subl      __[1]__, %eax
15.     cmpl      __[2]__, %eax
16.     __[3]__   .L3
17.     movl      %eax, %eax
18.     movq      __[4]__, %rax
19.     jmp  *%rax
20. .L4:
21.     cmpl      $0, -4(%rbp)
22.     jle   .L6
23.     movl      -4(%rbp), %eax
24.     movslq    %eax, %rdx
```

```
54. .L7:
55.     movl      -4(%rbp), %eax
56.     leal      3(%rax), %edx
57.     testl     %eax, %eax
58.     cmovs     %edx, %eax
59.     sarl      $2, %eax
60.     leaq      0(,%rax,4), %rdx
61.     movq      -24(%rbp), %rax
62.     addq      %rax, %rdx
63.     movl      -4(%rbp), %eax
64.     movslq    %eax, %rcx
65.     movq      -24(%rbp), %rax
66.     addq      %rcx, %rax
67.     movzbl    (%rax), %eax
68.     movsbl    %al, %eax
69.     movl      %eax, (%rdx)
70.     jmp  .L9
71. .L8:
72.     movl      -4(%rbp), %eax
73.     movslq    %eax, %rdx
74.     movq      -24(%rbp), %rax
75.     addq      %rax, %rdx
76.     movq      -24(%rbp), %rax
77.     movq      (%rax), %rax
```

```
25.     movq      -24(%rbp), %rax
26.     addq      %rdx, %rax
27.     movl      -4(%rbp), %edx
28.     movslq    %edx, %rcx
29.     movq      -24(%rbp), %rdx
30.     addq      %rcx, %rdx
31.     movzbl    (%rdx), %edx
32.     movl      %edx, %esi
33.     movl      -4(%rbp), %edx
34.     movslq    %edx, %rdx
35.     leaq      -1(%rdx), %rcx
36.     movq      -24(%rbp), %rdx
37.     addq      %rcx, %rdx
38.     movzbl    (%rdx), %edx
39.     addl      %esi, %edx
40.     ________[5]________
41. .L6:
42.     movl      -4(%rbp), %eax
43.     movslq    %eax, %rdx
44.     movq      -24(%rbp), %rax
45.     addq      %rdx, %rax
46.     movl      -4(%rbp), %edx
47.     movslq    %edx, %rcx
48.     movq      -24(%rbp), %rdx
49.     addq      %rcx, %rdx
50.     movzbl    (%rdx), %edx
51.     orl       $-16, %edx
52.     movb      %dl, (%rax)
53.     jmp   .L9
```

```
78.     sarq      $8, %rax
79.     movb      %al, (%rdx)
80.     jmp  .L9
81. .L3:
82.     cmpl      __[6]__, -4(%rbp)
83.     jg   .L9
84.     movl      -4(%rbp), %eax
85.     movslq     %eax, %rdx
86.     movq      -24(%rbp), %rax
87.     addq      %rdx, %rax
88.     ________[7]________
89. .L9:
90.     addl      $1, -4(%rbp)
91. .L2:
92.     cmpl      __[8]__, -4(%rbp)
93.     jle  .L11
94.     popq      %rbp
95.     ret

96.     .section   .rodata
97.     .align 8
98.     .align 4
99. .L5:
100.    /* hidden */
```

1. Jump table under label `.L5` is hidden. Please fill in the hidden part according to the C code given above.
2. Please fill in the blanks in the assembly code. (Hint: [5] and [7] use data movment instructions.)

   [1]_____ [2]_____
   [3]_____ [4]_____
   [5]_____ [6]_____
   [7]_____ [8]_____
3. For the input of function `transfer`, suppose `buf` is a char array with elements given below:

0	1	2	3	4	5	6	7
0x24	0xf0	0x22	0x1f	0x22	0x21	0x1f	0x21

   After the execution of `transfer`, what will `buf` look like? Fill in the table below.

0	1	2	3	4	5	6	7