

Jump Table

1.

```
.quad .L4  
.quad .L6  
.quad .L6  
.quad .L7  
.quad .L3  
.quad .L8  
.quad .L7
```

2.

[1] 31/0x1f	[2] 6
[3] ja	[4] .L5(, %rax, 8)
[5] movb %dl, (%rax)	[6] 6
[7] movw 0x201f, (%rax)	[8] 7

3.

0	1	2	3	4	5	6	7
0xf0	0x1f	0xf0	0xff	0x22	0x1f	0xf0	0x00

1. Jump table under label .L5 is hidden. Please fill in the hidden part according to the C code given above.

```
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. }
```

Base: 0x1f

Length: 7 (0x1f – 0x25)

```

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. }

```

case 0x1f <-> .L4

```

20. .L4:
21.     cmpl    $0, -4(%rbp)
22.     jle     .L6
23.     movl    -4(%rbp), %eax
24.     movslq   %eax, %rdx
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]_____

```

```

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.    }

```

case 0x20: case 0x21: <-> .L6

```

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

```

```

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. }

```

case 0x22: case 0x25: <-> .L7

```

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

```

```

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. }

```

case 0x24: <-> .L8

```

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
78.     sarq     $8, %rax
79.     movb     %al, (%rdx)
80.     jmp     .L9

```

```

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. }

```

```

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]_____

```

default: <-> .L3


```

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) { .L4
9.                     buf[i] += buf[i-1];
10.                }
11.            case 0x20: case 0x21: .L6
12.                buf[i] |= 0xf0;
13.                break;
14.            case 0x22: case 0x25:
15.                *((unsigned *)buf + i/4) .L7 = (unsigned)buf[i];
16.                break;
17.            case 0x24:
18.                buf[i] = (char)((*(long*)buf) >> 8);
19.                break; .L8
20.            default:
21.                if (i < BUF_SIZE - 1) {
22.                    *(short *)buf[i] = 0x201f;
23.                } .L3
24.                break;
25.            }
26.        }
27.    }

```

.L5:

```

.quad .L4
.quad .L6
.quad .L6
.quad .L7
.quad .L3
.quad .L8
.quad .L7

```

8.	movl	-4(%rbp), %eax		
9.	movslq	%eax, %rdx		
10.	movq	-24(%rbp), %rax		
11.	addq	%rdx, %rax		
12.	movzbl	(%rax), %eax	.L5:	
13.	movsbl	%al, %eax	.quad .L4	
14.	subl	Base: 0x1f[1]__, %eax	.quad .L6	
15.	cmpl	length-1: 6[2]__, %eax	.quad .L6	
16.	ja	[3]__ .L3	.quad .L7	
17.	movl	%eax, %eax	.quad .L3	
18.	movq	.L5(,%rax,8)[4]__, %rax	.quad .L8	
19.	jmp	*%rax	.quad .L7	

```

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.	movl	-4(%rbp), %eax	
9.	movslq	%eax, %rdx	
10.	movq	-24(%rbp), %rax	
11.	addq	%rdx, %rax	
12.	movzbl	(%rax), %eax	.L5:
13.	movsbl	%al, %eax	.quad .L4
14.	subl	Base: 0x1f[1]__, %eax	.quad .L6
15.	cmpl	length-1: 6[2]__, %eax	.quad .L6
16.	ja	[3]__ .L3	.quad .L7
17.	movl	%eax, %eax	.quad .L3
18.	movq	.L5(,%rax,8)[4]__, %rax	.quad .L8
19.	jmp	*%rax	.quad .L7

Other answers?

- [2] 7, [3] jae
- [1] 0x1f + X, [2] 6 + X, [4] .L5(-X, %rax, 8)

```

20. .L4:
21.     cml     $0, -4(%rbp)
22.     jle     .L6
23.     movl     -4(%rbp), %eax
24.     movslq    %eax, %rdx
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] movb %dl, (%rax)

```

buf[i] += buf[i-1]

- Type: char
- movb

```

81. .L3:
82.     cmpl    6[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] movw $0x201f, (%rax)

```

```

If (i < BUF_SIZE - 1) {
    *(short *)buf[i] = 0x201f;
}

```

- Type: short
- movw

```
91. .L2:
92.      cmpl    7____[8]____, -4(%rbp)
93.      jle     .L11
94.      popq    %rbp
95.      ret
```

```
for (i = 0; i < BUF_SIZE; i++) {
}
```

```

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. }

```

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

```

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. }

```

i = 0

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


```

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. }

```

i = 1

0	1	2	3	4	5	6	7
0x24	0xf0	0x22	0x1f	0x22	0x21	0x1f	0x21
0xf0	0x1f	0x20	0x1f	0x22	0x21	0x1f	0x21

```

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.    }

```

i = 2

0	1	2	3	4	5	6	7
0x24	0x1f	0x20	0x1f	0x22	0x21	0x1f	0x21
0xf0	0x1f	0xf0	0x1f	0x22	0x21	0x1f	0x21

```

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.    }

```

No break here!!

i = 3

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

```

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. }

```

i = 4

0	1	2	3	4	5	6	7
0x24	0x1f	0xf0	0xff	0x22	0x21	0x1f	0x21
0xf0	0x1f	0xf0	0xff	0x22	0x0	0x0	0x0

```

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. }

```

i = 5

0	1	2	3	4	5	6	7
0x24	0x1f	0xf0	0xff	0x22	0x0	0x0	0x0
0xf0	0x1f	0xf0	0xff	0x22	0x1f	0x20	0x0

```

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. }

```

i = 6

0	1	2	3	4	5	6	7
0x24	0x1f	0xf0	0xff	0x22	0x1f	0x20	0x0
0xf0	0x1f	0xf0	0xff	0x22	0x1f	0xf0	0x0

```

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. }

```

i = 7

0	1	2	3	4	5	6	7
0x24	0x1f	0xf0	0xff	0x22	0x1f	0xf0	0x0
0xf0	0x1f	0xf0	0xff	0x22	0x1f	0xf0	0x0