ICS Quiz 2

Fall, 2019

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

1. Here we go. (26')

The C codes for the function **`foo`** are shown below.

```
1.  void foo(void) {
2.          int a[2][4]={0};
3.          int i;
4.          for (i = 1; i < 8; i++)
5.                  ((int *)a)[i] = i+1;
6.
7.          printf("0x%lx\n", *(((uint64_t *)a[1])-2));
8.  }
```

1. For **line 2-line 5**, the disassembled version of the .o format generated by the assembler is as follow (some machine codes are hidden). Please fill in the blanks of the corresponding assembly codes. (5 * 4')
Hints: (1) The **`cltq`** instruction copies the sign of the doubleword in the **`%eax`** register into the high 32 bits of **`%rax`**) (2) The machine code **`7e`** is instruction **`jle.`**

```
1.   6c1:    48 c7 45 d0 00 00 00      movq   $0x0,-0x30(%rbp)
2.   6c8:    00
3.   6c9:    48 c7 45 d8 00 00 00      movq   $0x0,-0x28(%rbp)
4.   6d0:    00
5.   6d1:    48 c7 45 e0 00 00 00      movq   $0x0,-0x20(%rbp)
6.   6d8:    00
7.   6d9:    48 c7 45 e8 00 00 00      movq   $0x0,-0x18(%rbp)
8.   6e0:    00
9.   6e1:    c7 45 cc 01 00 00 00      movl   $0x1,-0x34(%rbp)
10.  6e8:    eb 20                     jmp    70a <foo+0x60>
11.  6ea:    8b 45 cc                  mov    -0x34(%rbp),%eax
12.  6ed:    48 98                     cltq
13.  6ef:       ***(hidden)***         lea     [1] ,%rdx
14.  6f6:    00
15.  6f7:       ***(hidden)***          [2]   -0x30(%rbp),%rax
16.  6fb:    48 01 d0                  add    %rdx,%rax
17.  6fe:       ***(hidden)***         mov     [3] ,%edx
18.  701:    83 c2 01                  add    $0x1,%edx
19.  704:    89 10                     mov    %edx,(%rax)
20.  706:    83 45 cc 01               addl   $0x1,-0x34(%rbp)
21.  70a:       ***(hidden)***         cmpl    [4] ,-0x34(%rbp)
22.  70e:    7e  [5]                   jle    6ea <foo+0x40>
23.  710:    48 8d 45 d0               lea    -0x30(%rbp),%rax
```

2. What’s the output of the **printf** function in **line 7**? (6’)

2. Let’s jump! (24’)

A C function **`switcher`** and its corresponding assembly code are given below. However, there are some mistakes in the assembly code and the C code is incomplete. Please read the given codes and answer the following questions.

```
1.  switcher:
2.      pushq    %rbp
3.      movq     %rsp, %rbp
4.      movq     %rdi, -24(%rbp) /*i*/
5.      movq     %rsi, -32(%rbp) /*xp*/
6.      movq     %rdx, -40(%rbp) /*yp*/
7.      movq     -40(%rbp), %rax
8.      movq     $0, %rdx
9.      testq    %rax, %rax
10.     cmovne   (%rax), %rdx
11.     movq     %rdx, -8(%rbp) /*ret*/
12.     cmpq     $0, -32(%rbp)
13.     je       .L1
14.     cmpq     $0, -40(%rbp)
15.     jne      .L3
16. .L1:
17.     movl     $0, %eax
18.     jmp      .L2
19. .L3:
20.     movq     -24(%rbp), %rax
21.     subq     $0x20, %rax
22.     cmpq     $0x5, %rax
23.     ja       .L4
24.     movq     .L6(,%rax,8), %rax
25.     jmp      *%rax
26. .L5:
27.     movq     -32(%rbp), %rax
28.     movl     (%rax), %eax
29.     movzlq   %eax, %rdx
30.     movq     -40(%rbp), %rax
31.     movq     %rdx, (%rax)
32.     jmp      .L9
33. .L7:
34.     movq     -32(%rbp), %rax
35.     movzbl   (%rax), %eax
36.     movsbl   %al, %edx
37.     movq     -40(%rbp), %rax
38.     movl     %edx, (%rax)
39.     nop
40. .L9:
41.     addl     $2, -8(%rbp)
42.     jmp      .L10
43. .L8:
44.     movq     -32(%rbp), %rax
45.     movl     (%rax), %eax
46.     movl     %eax, %edx
47.     movq     -40(%rbp), %rax
48.     movb     %dl, (%rax)
49. .L4:
50.     salq     $2, -8(%rbp)
51.     nop
52. .L10:
53.     movq     -8(%rbp), %rax
54. .L2:
55.     popq     %rbp
56.     ret
57.
58. .section     .rodata
59.     .align 8
60. .L6:
61.     .quad    .L5
62.     .quad    .L7
63.     .quad    .L4
64.     .quad    .L8
65.     .quad    .L9
66.     .quad    .L9
```

```
1.  long switcher(long i, unsigned *xp, long *yp) {
2.      long ret = (yp)?*yp:0;
3.      if (!xp || !yp) {
4.          return 0;
5.      }
6.      switch (i) {
7.      [1]______   // case label
8.          *(char *)yp = (char)*xp;
9.      [2]______   // case label
10.         [3]______
11.         break;
12.     [4]______   // case label
13.         [5]______
14.         break;
15.     [6]______    // case label
16.         *(unsigned *)yp = (unsigned)*((char *)xp);
17.         goto Puzzle_label;
18.     [7]______    // case label
19.         *yp = (long)*xp;
20.         goto Puzzle_label;
21.     }
22.     return ret;
23. }
```

1. Fill in the blanks in the C code. The mistakes in the assembly code do **not** prevent you from correctly answer this question (7 * 2').

2. There is a missing label **`Puzzle_lable`** in the C code. **`Puzzle_lable`** should be placed in the front of which line (2')?

3. There are 3 mistakes in the assembly code between **line 7** and **48** and one has been found and is given as an example below. Please find the others (use a line number to represent each mistake) and explain your reasons (2 * 4').

   Example:

   ① Line Number: 41. Reason: **`-8(%rbp)`** saves the local variable **`ret`**. The type of **`ret`** is **`long`** so the assembly instruction should be **`addq`** instead of **`addl`**.

3. It is the time to hack! (30')

Alice uses a simple encryption algorithm to communicate with *Bob*. *Eve* can intercept their encrypted data but he has no idea about the unencrypted message. Somehow, he gets chance to steal the key and the binary encryption program from *Alice's* computer. You are asked to help *Eve* find out the message that *Alice* wants to tell *Bob*.
The template of the encryption program is shown below. The unencrypted message and the key have been stored in **`src`** and **`key`**. Both of them are **5-character** long. The encrypted data is also **5-character** long and will be stored to **`dst`** in this function.

```
void encrypt(char *src, char *key, char *dst);
```

Eve uses a disassembler to disassemble the program. The beautified version of the *encrypt* function from *Alice's* computer is shown below. Answer the following question and help *Eve* to get the message.

```
1.  encrypt:
2.          pushq   %rbp
3.          movq    %rsp, %rbp
4.
5.          movq    %rdi, -24(%rbp)                   /* src pointer */
```

```
        movq    %rsi, -32(%rbp)                     /* key pointer */
        movq    %rdx, -40(%rbp)                     /* dst pointer */

        movl    $0, -4(%rbp)                        /* local variable i */
        movl    $0, -8(%rbp)

loop:
        cmpl    $4, -4(%rbp)
        ja      encrypt_end

        movl    -4(%rbp), %eax
        movslq  %eax, %rdx
        movq    -32(%rbp), %rax
        addq    %rdx, %rax
        movzbl  (%rax), %eax                /* %eax = _[1]_ */
        imull   -4(%rbp), %eax
        movl    %eax, %edx                  /* %edx = _[2]_ */

        movl    -4(%rbp), %eax
        movslq  %eax, %rcx
        movq    -24(%rbp), %rax
        addq    %rcx, %rax
        movzbl  (%rax), %eax                /* %eax = _[3]_ */


        addl    %edx, %eax                  /* %eax = _[4]_ */
        movl    %eax, -8(%rbp)

        movl    -4(%rbp), %eax
        andl    $1, %eax
        testl   %eax, %eax
        je      branch1

        movq    $0, %rax
        movl    -4(%rbp), %eax
        leaq    -1(%rax), %rdx
        movq    -40(%rbp), %rax
        addq    %rdx, %rax
        movzbl  (%rax), %eax                /* %eax = _[5]_ */
        addl    %eax, -8(%rbp)

branch1:
        movl    -4(%rbp), %eax
        movslq  %eax, %rdx
        movq    -40(%rbp), %rax
        addq    %rdx, %rax
        movl    -8(%rbp), %edx
        movb    %dl, (%rax)
        addl    $1, -4(%rbp)
        jmp     loop

encrypt_end:
```

```
58.         popq    %rbp
59.         ret
```

1. *Eve* has marked one of the local variables as `i`, help him fill the blank in the comments use `i, src, dst, key` and instant numbers.

2. What is the condition that the code between **line 34** and **37** checked?

3. The encrypted data and the key that *Eve* intercepted are shown below.

	0	1	2	3	4
Key	0x05	0x01	0x02	0x03	0x04
Dst	0x73	0x75	0x06	0x12	0x14

Try to figure out the unencrypted message that *Alice* wants to tell *Bob* (in hex).

4. What the hack is it? (20')

Bob is doing his algorithm homework. Somehow he falls back to some previous version mistakenly and cannot redo. But fortunately, the binary version remains untouched. The following code is the disassembled from that binary. Help *Bob* finish his homework!

```
1.  f1:
2.      movl    %edx, %eax
3.      movl    %esi, %edx
4.      jmp     .L47
5.  .L49:
6.      addl    $1, %edx
7.  .L54:
8.      cmpl    %eax, %edx
9.      jg      .L50
10.     movslq  %edx, %rcx
11.     movslq  %esi, %r8
12.     movl    (%rdi,%r8,4), %r11d
13.     cmpl    %r11d, (%rdi,%rcx,4)
14.     jle     .L49
15.     jmp     .L50
16. .L52:
17.     subl    $1, %eax
18. .L50:
19.     cmpl    %eax, %edx
20.     jg      .L51
21.     movslq  %eax, %rcx
22.     movslq  %esi, %r8
23.     movl    (%rdi,%r8,4), %r9d
24.     cmpl    %r9d, (%rdi,%rcx,4)
25.     jg      .L52
26. .L51:
27.     cmpl    %eax, %edx
28.     jge     .L47
29.     movslq  %edx, %rcx
30.     leaq    (%rdi,%rcx,4), %r8
31.     movl    (%r8), %r9d
32.     movslq  %eax, %rcx
33.     leaq    (%rdi,%rcx,4), %rcx
34.     movl    (%rcx), %r10d
35.     movl    %r10d, (%r8)
36.     movl    %r9d, (%rcx)
37. .L47:
38.     cmpl    %eax, %edx
39.     jle     .L54
40.     movslq  %esi, %rsi
41.     leaq    (%rdi,%rsi,4), %rcx
42.     movl    (%rcx), %esi
43.     movslq  %eax, %rdx
44.     leaq    (%rdi,%rdx,4), %rdx
45.     movl    (%rdx), %edi
46.     movl    %edi, (%rcx)
47.     movl    %esi, (%rdx)
48.     ret
```

```
int function(int* arr, int l, int h) {
    int p = l;

    while(l <= h) {
        while (___[1]____)
            l++;
        while (___[2]____)
            h--;
        if(l < h) {
              ___[3]___
        }
    }
    ___[4]____
    return h;
}
```

1. Please fill in the blanks of the corresponding C codes.

2. What the function does? Give a proper name to this function.