Lab 3

Practice Problem 3.18

Starting with C code of the form

```
1   int test(int x, int y) {
2       int val = __________;
3       if (__________) {
4           if (__________)
5               val = __________;
6           else
7               val = __________;
8       } else if (__________)
9           val = __________;
10      return val;
11  }
```

GCC generates the following assembly code:

```
    x at %ebp+8, y at %ebp+12
1     movl    8(%ebp), %eax
2     movl    12(%ebp), %edx
3     cmpl    $-3, %eax
4     jge     .L2
5     cmpl    %edx, %eax
6     jle     .L3
7     imull   %edx, %eax
8     jmp     .L4
9   .L3:
10    leal    (%edx,%eax), %eax
11    jmp     .L4
12  .L2:
13    cmpl    $2, %eax
14    jg      .L5
15    xorl    %edx, %eax
16    jmp     .L4
17  .L5:
18    subl    %edx, %eax
19  .L4:
```

Fill in the missing expressions in the C code. To make the code fit into the C code template, you will need to undo some of the reordering of computations done by GCC.

Practice Problem 3.22

A function, `fun_a`, has the following overall structure:

```
int fun_a(unsigned x) {
    int val = 0;
    while ( _________ ) {
        _________;
    }
    return _________;
}
```

The GCC C compiler generates the following assembly code:

```
    x at %ebp+8
 1      movl    8(%ebp), %edx
 2      movl    $0, %eax
 3      testl   %edx, %edx
 4      je      .L7
 5    .L10:
 6      xorl    %edx, %eax
 7      shrl    %edx              Shift right by 1
 8      jne     .L10
 9    .L7:
10      andl    $1, %eax
```

Reverse engineer the operation of this code and then do the following:

A. Use the assembly-code version to fill in the missing parts of the C code.

B. Describe in English what this function computes.

Practice Problem 3.23

A function `fun_b` has the following overall structure:

```
int fun_b(unsigned x) {
    int val = 0;
    int i;
    for ( ________ ; ________ ; ________ ) {
        ________
    }
    return val;
}
```

The GCC C compiler generates the following assembly code:

```
    x at %ebp+8
 1     movl    8(%ebp), %ebx
 2     movl    $0, %eax
 3     movl    $0, %ecx
 4   .L13:
 5     leal    (%eax,%eax), %edx
 6     movl    %ebx, %eax
 7     andl    $1, %eax
 8     orl     %edx, %eax
 9     shrl    %ebx              Shift right by 1
10     addl    $1, %ecx
11     cmpl    $32, %ecx
12     jne     .L13
```

Reverse engineer the operation of this code and then do the following:

A. Use the assembly-code version to fill in the missing parts of the C code.

B. Describe in English what this function computes.

Practice Problem 3.27

Starting with C code of the form

```
1   int test(int x, int y) {
2       int val = __________;
3       if (__________) {
4           if (__________)
5               val = __________;
6           else
7               val = __________;

8       } else if (__________)
9           val = __________;
10      return val;
11  }
```

GCC, with the command-line setting '`-march=i686`', generates the following assembly code:

```
    x at %ebp+8, y at %ebp+12
1     movl    8(%ebp), %ebx
2     movl    12(%ebp), %ecx
3     testl   %ecx, %ecx
4     jle     .L2
5     movl    %ebx, %edx
6     subl    %ecx, %edx
7     movl    %ecx, %eax
8     xorl    %ebx, %eax
9     cmpl    %ecx, %ebx
10    cmovl   %edx, %eax
11    jmp     .L4
12  .L2:
13    leal    0(,%ebx,4), %edx
14    leal    (%ecx,%ebx), %eax
15    cmpl    $-2, %ecx
16    cmovge  %edx, %eax
17  .L4:
```

Fill in the missing expressions in the C code.