Homework 4

Array

Consider the following program:

```
#define LEN 10
int a[LEN][LEN];
void f(void) {
    int i, j;
    for (i = 0; i < LEN; i++)
        for (j = 0; j < LEN; j++) {
            a[i][j] = i * LEN + j;
        }
}
```

Suppose the address of a is 0x10000000. After the function f() finished, fill the following table (if you don't know the value, please write NONE):

%eax	0x10000000
%ecx	22
$0x10000004	
0x10000012	
0xFFFFFF8	
(%eax, %ecx, 8)	

Jump Table

Translate the following switch statements into assembly using jump table.

```
int x = <some value>;
int result = 0;
switch (x) {
    case 24:
        result = x + x;
        break;
    case 27: case 28:
        result = x + 10;
        break;
    case 26:
        result = x * 2;
        // Notice: there is no break here!
    case 29: case 30:
        result = result + 5;
```

```
            break;
        default:
            result = 3;
            break;
}
```

Stack

3. Suppose the initial value of %esp is 0x7FFFFFC4, initial value of %ebp is 0x7FFFFFF4.
The value stored in address 0x7FFFFFC0 is 0x120, value stored in address 0x7FFFFFC4 is 0x200, the value stored in address 0x7FFFFFF4 is 0x2710.
We have following x86 assembly code executed sequentially:
pushl %esp (instruction 1)
movl %esp,%ebp (instruction 2)
popl %ebp (instruction 3)
Question: After each instruction executed, what is the value of %esp and %ebp
(1) Instruction 1:
(2) Instruction 2:
(3) Instruction 3:

Conditional Move

4. After ICS class, Barathrum has written a function like below:

```
int cmov_complex(int x, int y) {
        return x < y? x * y; (x + y) * y;
}
```

(1). Please write down the corresponding assembly code by using conditional move operations.
(2). When Barathrum compiles it with gcc, he finds that there's no cmov at all in the assembly code! Please explain why gcc doesn't use conditional move operations in this case. (**Suggested reading: 3.6.6**)