Homework 8

T1

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):

expression	value
%eax	0x10000000
%ecx	22
$0x10000004	
0x10000012	
0xFFFFFF8	

(%eax, %ecx, 8)

T2

Fill the blanks of the C program:

```
int dw_loop(int x, int y, int n) {
     do{


      }while (        );
      returen x;
}
```

The assembly code is as follows:

x@%ebp+8, y@%ebp+12, n@%ebp+16

```
  movl 8(%ebp), %eax
  movl 12(%ebp), %ecx
  movl 16(%ebp), %edx
.L2:
  addl %edx, %eax
  imull %edx, %ecx
  subl $1, %edx
  testl %edx, %edx
  jle  .L5
  cmpl %edx, %ecx
  jl .L2
.L5:
```

T3

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.

T4

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;
    }
```