Chun-Wei Chen
CSE 351
Homework 2
05/04/13

1. Since this piece of code is essentially the same as sequential instructions to store the address of popl instruction to %eax, the stack is not affected.

2. %edi, %esi, and %ebx are callee-save registers, which means callee must restore the original value before return back to the caller since the caller might need to use the value stored in those register after callee returns. The other three are caller-save registers, which mean the values in those register is saved by the caller prior to calling subroutines; therefore, these registers can be overwrite without restore the original values.

3. Change line 4 to subl $20, %esp

4. My strategy was aligning the elements from the largest size first to the smallest size. I don't think it's going to work all the time since this strategy only works well if structure elements have size of power of 2 and if structure elements are not declare in descending order of size. The general strategy I come up with is aligning the structure elements with size of power of 2 in descending order of size first, then aligning the rest of the elements.

5. 5L

6.

```
int switch_prob(int x, int n) {
        int result = x;
        switch(n) {
                case 32:
                case 34:
                        result <<= 2;
                        break;
                case 35:
                        result >>= 2;
                        break;
                case 36:
                        result *= 3;
                case 37:
                        result = result * result;
                default:
                        result += 10;
        }
        return result;
}
```

7.
A. 8(%ebp): result (structure to be returned), 12(%ebp): s1.a; 16(%ebp): s1.p
B. %esp stores address of s2, 4(%esp) stores argument x, 8(%esp) stores address of argument y, -8(%ebp) holds address of s2, which is also address of s2.sum, and -4(%ebp) holds address of s2.diff
C. Treat each structure's element as individual argument. In this problem, str2 word_sum(str1 s1) is treated as str2 word_sum(int a, int *p).
D. Move the structure to be returned in %eax (or %rax in X86-64) and do something to its elements using address arithmetic such as movl %ecx, 4(%eax).