CS 4400 - Problem Set 4

Rob Johansen

u0531837

---

1. Problem 3.56:

   A. %esi holds x
      %ebx holds n
      %edi holds result
      %edx holds mask

   B. The initial value of result is -1
      The initial value of mask is 1

   C. The test condition for mask is: mask != 0

   D. This is how mask gets updated: mask = mask << n

   E. This is how result gets updated: result ^= mask & x

   F. Here is the entire function, with all code filled in:

```
int loop(int x, int n)
{
    int result = -1;
    int mask;
    for (mask = 1; mask != 0; mask = mask << n) {
        result ^= mask & x;
    }
    return result;
}
```

2. Problem 3.58. Here's the function with the missing code. Note that I removed the default case because it did nothing anyway:

```
int switch3(int *p1, int *p2, mode_t action)
{
      int result = 0;
      switch (action) {
      case MODE_A:
            result = *p1;
            *p1 = *p2;
            break;
```

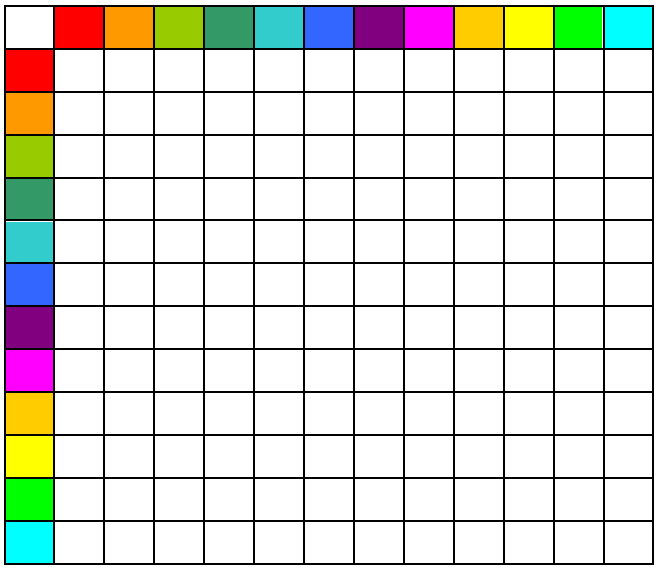
```
        case MODE_B:
            result  = *p2;
            result += *p1;
            *p2 = result;
            break;
        case MODE_C:
            *p2 = 15;
            result = *p1;
            break;
        case MODE_D:
            *p2 = *p1;
        case MODE_E:
            result = 17;
        }
        return result;
}
```

3. Problem 3.62:

   A. The value of M is 13

   B. Value *j* is held in %ecx. Value *i* is unnecessary after the optimizations.

   C. The compiler recognizes that only two elements will be transposed during each iteration of the loop: The topmost "column" element and its corresponding leftmost "row" element. The following diagram illustrates the elements of array *A* that will be transposed (those with matching colors):



Here is a C code version of the transpose() function that makes use of these optimizations:

```c
#define M 13
typedef int Marray_t[M][M];

void transpose(Marray_t A)
{
      int j;
      int *col = &A[0][0];
      int *row = &A[1][0];
      for (j = 1; j < M; j++) {
            int t = *row;
            *row = col[j];
            col[j] = t;
            row += M;
      }
}
```