ECE454 HW2

Yiming Kang 998676730
Zexuan Wang 998851773

In this assignment, we employed various methods to improve the run time (in CPE) of matrix rotation function. A final best speedup of 2.7x is recorded.

Implementation:

1. We used tiling to break the given matrices into smaller chunks to increaes L1 hit rate (spacial locality + fit into L1 cache).
    - Intel i7 4790 has 32kB **deticated** L1 cache per physical CPU core, 8 way set-associative. Each cahce line is 64 bytes long
    - Each pixel has 3 shorts (6 bytes). L1 cache is able to fit 32 * 1024 / 6 = 5461 pixels
    - Since we used src and dst matricies, each matrix has a max size of 5461/2 = 3720 pixels
    - We used a scrpit to find the best dimensions for the matrices (see script_1)
    - The best dimensions are:

Matrix size	Src len (pixel)	Src width (pixel)
< 1024	32	2
> 1024	16	8

1. Loop unrolling

    - We found that this is already done by the compiler in -O2

2. Code motion

    - The preprocessor macro RIDX has a multiplication that can be moved outside of the innermost loop. However doing so had no influence in performance and it's most likely already done by the compiler.

Scripts:

script_1: Look for the ideal size for matrix

```
# for num_y in {8..514..8}
for num_y in 2 4 8 16 32 64 96 128 256 512 1024 2048
do
    for num_x in 2 4 8 16 32 64 96 128 256 512 1024 2048
    # for num_x in {8..514..8}
    do
        sed -i.bak "s/^#define TILE_SIZE_X.*$/#define TILE_SIZE_X ${num_x}/g" kernels.c
        sed -i.bak "s/^#define TILE_SIZE_Y.*$/#define TILE_SIZE_Y ${num_y}/g" kernels.c
        make clean > /dev/null
        make > /dev/null
        OUTPUT="$(./driver_cpe -g)"
        echo "x: ${num_x}; y: ${num_y}"
        echo "${OUTPUT}" | grep "Speedup"
    done
done
```