ECE454 Lab5 Report

Student Name: Gujaing Lin Student Number:1000268239

Student Name: Zhuang Li Student Number: 1000311628

Optimization:

1. We **parallelized** the process of copying and generating result process with **8 threads**
2. We **lock each thread to one CPU** core to avoid content switching, to reduce the cache flush.
3. For the result after 1 iteration, instead using alivep function, we generate a 2^9=512 entries **char array to pre load all possible outcome for 3*3 blocks.** Therefore, we can access the array to get the result instead of calculation.
4. We make 9 variables to store data for each blocks.

   Before optimization, every blocks need read 8 neighbor blocks and itself to calculate the new value.

   Now, we just need read 3 new blocks and used the right 6 blocks from last blocks to calculate result. It reduce memory access from 9 to 6;

5. We **swap row and col**, so we can access the board with less cache miss. To be more specific, the old board access is

   #define BOARD(*__board, __i, __j*) (__board[(__i) + LDA*(__j)])

   And we change it to

   #define BOARD(*__board, __i, __j*) (__board[(__j) + LDA*(__i)])

New Source files

Lifemt.c and lfemt.h: This source file contains our new function with multi threads support.

In lifent.h, we define *struct* __ggWorkerContext contains all the data for the board in order to pass them to functions run by pthread_create.

Function mt_game_of_life, it will take the pointer of input board, output board and all necessary data. This function is just high level wrapper and divide the data into 8 threads.

Function ggWorkerThread, it is optimized version of sequential_game_of_life. Each thread will run this function and calculate data.