Conway’s Game of Life as a multithreaded program  
• Module Code: CMP202 2023/24 Term 2  
• Mini-Project: Mini-Project 1, CPU  
• Student Name: Ari Thomson  
• Student ID: 2204636

# Introduction

For my CPU project I decided to create a limited version of Conway’s game of life, a basic life simulation where cells are either alive or dead based on their surrounding conditions. I aimed to implement parallel techniques by dividing the plane of cells into sections, or “chunks”, to run on threads.

# Application Overview

Conway’s game of life is a relatively simple simulation that came around in 1970, it is a “zero player game” where one sets starting conditions and it plays itself. It follows four basic rules that will be referred to throughout my report.

1. If a live cell has less than two surrounding live cells, it will “die”.
2. If a live cell has four or more surrounding live cells, it will “die”.
3. If a live cell has two or three surrounding live cells, it will “live”.
4. If a dead cell has exactly three surrounding live cells, it will “come to life”.

Normally the simulation is run on an infinite plane however, as mentioned, I instead created a limited reproduction where it instead is run on a twelve-by-twelve grid.

# Parallelisation Strategy and Implementation

To parallelise this problem, I split the grid up into “chunks” which would each be managed by separate threads, however this does not mean that each chunk could act entirely independently as the edge cells of each chunk would need to be read by other edge cells from different chunks.

Each thread would then run through two key phases, the first, it finds what the new value of each cell within its chunk should be based off of the value of surrounding cells, it would write this value to a separate data structure so as to impact the main plane until all cell values were found.

Upon finding all the new values the thread would then wait at a barrier so that all threads could be synchronised.

After all the threads have reached the barrier, each thread would then transfer the values from their temporary data structures into the main array that stores the whole plane.

In the first phase the threads will also find how many cells were alive and dead throughout the simulation by incrementing the corresponding variables, to ensure that these values were always correct I used mutex locks to protect the variables.

# Performance Evaluation

# Conclusion