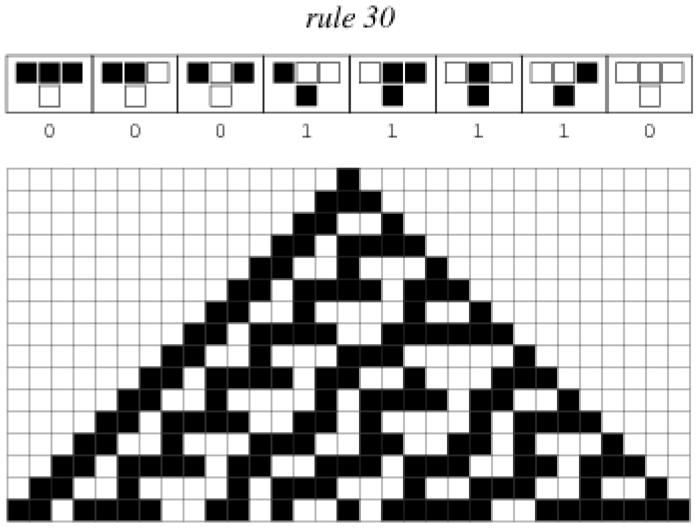
Project Assignment 1: 1D and 2D Cellular Automata

Description of Assignment

A cellular automaton is a collection of "colored" cells on a grid of specified shape that evolves through a number of discrete time steps according to a set of rules based on the states of neighboring cells. The rules are then applied iteratively for as many time steps as desired.

Cellular automata come in a variety of shapes and varieties. One of the most fundamental properties of a cellular automaton is the type of grid on which it is computed. The simplest such "grid" is a one-dimensional line. For a binary automaton, color 0 is commonly called "white," and color 1 is commonly called "black". In addition to the grid on which a cellular automaton lives and the colors its cells may assume, the neighborhood over which cells affect one another must also be specified. The simplest choice is "nearest neighbors," in which only cells directly adjacent to a given cell may be affected at each time step.

The simplest type of cellular automaton is a binary, nearest-neighbor, one-dimensional automaton.
There are 256 such automata, each of which can be indexed by a unique binary number whose decimal representation is known as the "rule" for the particular automaton.
An illustration of rule 30 is shown here together with the evolution it produces after 15 generations starting from a single black cell. For example, a black cell with two black neighbors becomes white in the next generation.



PART 1: In this part, you will be given a text file containing a binary sequence of arbitrary length. Your C code should take the file name from the command line, should read the file, and store the text into the computer memory. Then your code should get the rule number (0 to 255), and the number of generations from the standard input.

You should write an assembly subroutine, to generate the next sequence, from the input binary sequnce and the rule number. The generated sequence should then be illustrated by outputting it to the standard output. For black and white cells, you can choose any appropriate ASCII character. Your final output should show all the generations, one below the other, similar to the upper figure.

It is your responsibility to choose the best parameter passing method between the main function and the subroutine. The function(s) are not required to perform error checking on the parameters, but must follow C calling conventions and must fully conform to the given prototypes.

PART 2: A binary two-dimensional, Neumann-neighborhood (r = 1) cellular automaton will be implemented.
Neumann binary rules are represented as a string of digits. The digits define the transition table - the state a cell will have in every possible configuration. For enumerating all possible neighborhood configurations the "(C)enter,(N)orth,(E)ast,(S)outh,(W)est" order is used.

Example:
A rule has the following definition: 01101001100101101001011001101001
The first digit, '0', tells a cell in a configuration C=0,N=0,E=0,S=0,W=0 will get the state 0.
The second digit, '1', tells a cell in a configuration C=0,N=0,E=0,S=0,W=1 will get the state 1.
The third digit, '1', tells a cell in a configuration C=0,N=0,E=0,S=1,W=0 will get the state 1.
The fourth digit, '0', tells a cell in a configuration C=0,N=0,E=0,S=1,W=1 will get the state 0.
. . .

In this part, you will be given a text file containing a binary matrix of arbitrary size, and another text file containing 32-digit rule sequence. Your C code should take the file name from the command line, should read the file, and store the matrix into the computer memory. Your code should get the rule as a 32-digit binary sequence from the second file. Number of generations will not be given for this part. Yuor code should continue to generate the matrices as long as the user wants. For illustration purposes, it is important that you clean the output screen before outputting the next matrix, and keep the width of the output window larger than the matrix width.

You should write an assembly subroutine, to generate the next sequence for the 2D automaton, from the input matrix, and the rule sequence. The next generated matrix should then be illustrated by outputting it to the standard output. For black and white cells, you can choose any appropriate ASCII character.

It is your responsibility to choose the best parameter passing method between the main function and the subroutine. The function(s) are not required to perform error checking on the parameters, but must follow C calling conventions and must fully conform to the given prototypes.

References:
1.http://mathworld.wolfram.com/CellularAutomaton.html
2. http://mathworld.wolfram.com/vonNeumannNeighborhood.html

Submission Details

You are required to implement the automata generation algorithms in Intel assembly. The function implementations must fully conform to the provided prototypes since they are expected to be linked to the main program implemented in C.

You are required to submit the C and assembly language source code file(s) through the Ninova system as a zip file. Each member of the group must make a submission, even though the submitted files may be the same for all group members.

Group members will be graded individually based on their performance in the lab session and the submitted group project. The students who are not present during the lab session will not receive a grade for the project, even though they may have made a submission through the Ninova system. Any form of cheating or plagiarism will not be tolerated. The submitted work should be the product of the group itself; collaboration or code sharing between different groups will be accepted as plagiarism. This also includes actions such as, but not limited to, submitting the work of others as one's own (even if in part and even with modifications) and copy/pasting from other resources (even when attributed). Serious offenses will be reported to the administration for disciplinary measures.