


Programming Lab 7C

Autonomous Sudoku

 Click to download Lab7C-Main.c

Topics: Shift, bitwise and bitfield instructions; address calculation with pointers.

Prerequisite Reading: Chapters 1-7

Revised: June 22, 2021

Background¹: Sudoku (originally called Number Place) is a logic-based, combinatorial number-placement puzzle. The objective is to fill a 9×9 grid with digits so that each column, each row, and each of the nine 3×3 subgrids that compose the grid (also called "boxes", "blocks", or "regions") contains all of the digits from 1 to 9. The puzzle setter provides a partially completed grid, which for a well-posed puzzle has a single solution. Completed games are always a type of Latin square with an additional constraint on the contents of individual regions. For example, the same single integer may not appear twice in the same row, column, or any of the nine 3×3 subregions of the 9x9 playing board.

Assignment: The main program keeps track of the status of cells in an array of 4-bit "nibbles" (two per byte) – one nibble per cell. It will compile and run without writing any assembly. However, your task is to create equivalent replacements in assembly language for the following two functions found in the C main program to store and retrieve individual nibbles within the array. The original C versions have been defined as “weak” so that the linker will automatically replace them in the executable image by those you create in assembly; you do not need to remove the C versions. This allows you to create and test your assembly language functions one at a time. Your solutions should execute as fast as possible, so using loops is not acceptable.

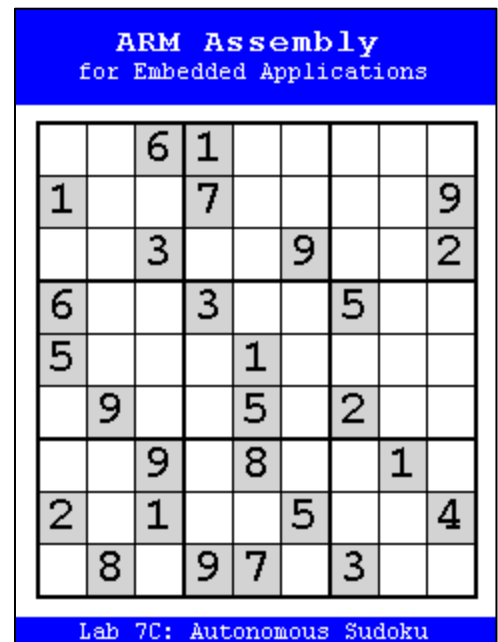
```
void PutNibble(void *nibbles, uint32_t which, uint32_t value) ;
uint32_t GetNibble(void *nibbles, uint32_t which) ;
```

Parameter Description

nibbles	Starting address where nibbles are stored in memory.
which	A nibble position (0 to 80) within the array of nibbles.
value	A four-bit number (0 to 9); 0 represents an empty cell.

Test your functions using the C main program. The program displays a solvable random placement of black digits. You may edit the cells using the touch screen; touching a cell selects the next entry for that cell that is not in conflict with any other cell.

Pressing the blue button starts a recursive back-tracking algorithm to solve the puzzle. Pressing the button before the solution is complete aborts the algorithm. If solved, the program displays the solution and waits for the user to press the blue button, which then displays some statistics. If aborted or if no solution can be found, the program goes directly to the statistics screen. Another button push restarts the program. Note that the time required to solve the puzzle will vary according to the initial placement.



¹ <https://en.wikipedia.org/wiki/Sudoku>