CDA3100
ASSIGNMENT 6 SUDOKU

Objectives:

Learn how to write an assembly program that consists of functions, learn the MIPS calling conventions, and learn how to access a two-dimensional array in assembly language. This will also be an exercise in gaming and game strategy solutions.

The Classic Sudoku is a number placing puzzle based on a 9x9 grid with several given numbers. The object is to place the numbers 1 to 9 in the empty squares so that each row, each column and each 3x3 box contains the same number only once. The following is an example of a valid Sudoku puzzle.

2	7	1	9	5	4	6	8	3
5	9	3	6	2	8	1	4	7
4	6	8	1	3	7	2	5	9
7	3	6	4	1	5	8	9	2
1	5	9	8	6	2	3	7	4
8	4	2	3	7	9	5	6	1
9	8	5	2	4	1	7	3	6
6	1	7	5	9	3	4	2	8
3	2	4	7	8	6	9	1	5

Specifications:

1. There will be 4 test files:
    a. good.dat # Valid Puzzle.
    b. bad1.dat # Invalid Puzzle
    c. bad2.dat # Invalid Puzzle
    d. bad3.dat # Invalid Puzzle
2. You will read in the input file from standard input and store the numbers in an array. The code to do this has been provided.
3. The file will consist of 81 consecutive integer numbers separated by one space.
    a. The first set of 9 numbers will be row 1st, the second set of 9 numbers will be the 2nd row, the third set of 9 numbers will be the 3rd Row, etc.
    b. You can assume that the file will be correct and have exactly 81 numbers with no invalid integers (i.e. less than 0 or greater than 9).

4. Your Program must incorporate the following:
    a. parameter passing and return values
    b. preserving register values, as presented in class
    c. In addition, you should not have any conditional branch instructions that cross function boundaries.
    d. A comment beside each instruction is required in the assembly program (or at a minimum a comment for a small group of related instructions) to help make your code more understandable. You should also have comments at the top of the file indicating your name, this course, and the assignment.
5. Your program must indicate whether the file that was submitted represents a valid solution, or whether there is an error in the row, column, and/or 3x3 quadrant.

Grading

In order to be graded the program must assemble and allow the grader to start the program.

- 10% - Proper declaration of the 2d array and references the array properly
- 10% - Program properly documented
- Contains comments that identifies the programmer
- Contains comments that describe how the program works.
- 25% - Program is modular.
    - Has functions with at least one instance of passing parameters using standard calling conventions and preserving registers.
    - Has functions with at least one instance of returning a value using standard calling conventions and preserving registers
- 55% - Correct implementation.
    - 10% Identifies valid files and continues to ask for a valid false
    - 15% Identifies valid Sudoku solutions
    - 30% Identifies errors with individual errors in Rows, Columns, or 3x3 Quadrants

.

Sample Output:

puzzle1.dat file

```
|1|2|3|4|5|6|7|8|9|
|-|-|-|-|-|-|-|-|-|
|4|5|6|7|8|9|1|2|3|
|-|-|-|-|-|-|-|-|-|
|7|8|9|1|2|3|4|5|6|
|-|-|-|-|-|-|-|-|-|
|2|3|1|5|6|4|9|7|8|
|-|-|-|-|-|-|-|-|-|
|5|6|4|8|9|7|3|1|2|
|-|-|-|-|-|-|-|-|-|
|8|9|7|2|3|1|6|4|5|
|-|-|-|-|-|-|-|-|-|
|3|1|2|6|4|5|8|9|7|
|-|-|-|-|-|-|-|-|-|
|6|4|5|9|7|8|2|3|1|
|-|-|-|-|-|-|-|-|-|
|9|7|8|3|1|2|5|6|4|

Valid Sudoku Puzzle
```

```
# ################################################################
# # File: sudoku.s                                               #
# # Description:   Query the user for a file name. We assume     #
# #  it must be a complete path name since we cannot anticipate#
# #  where QTSPIM will expect a local file. The routine (OFile)#
# #  will read in the filename and try to open it.  If not      #
# #  it will continue to ask the user for a valid filename.     #
# #  Once successful, the file pointer will be passed back to  #
# #  the calling routine.                                       #
# # Author:  Dr. David A. Gaitros                               #
# # Date:  June 25th, 2018                                      #
# # Copyright:  This code is property of FSU Department of      #
# #     Computer Science, written by Dr. David A. Gaitros for  #
# #     the sole use of students taking courses at the          #
# #     Distrubution is not authorized.                         #
##################################################################
      .data
Hello:  .asciiz "Hello,Enter a file name: \n"
      .align 2
NoFile: .asciiz "Error encountered, file did not open\n"
OkFile: .asciiz "File was opened\n"
Filen:  .space  256              # Set aside 256 Characters
EOL:    .byte '\n'
      .align 2
sudoku:    .space 164
NullCh:  .word  0          # Just in case we need a null character
                           #     After the puzzle.
      .align  2            # Align on full word boundary
      .text
      .globl main
main:

# #####################################################################
#  Save the return address ($ra) and call the Open File function(Ofile).
#######################################################################
      addiu $sp,$sp,-4     # Get space from the stack
      sw    $ra,0($sp)     # Store return address on stack
      la    $a0,Filen      # Load $a0 (parameter) with address to
                           #     store filename.
      jal   OFile          # Call Open File

# #######################################################################
# # Save the file pointer in saved register $s1.  Restore the return  #
# #  address and stack pointer.                                        #
#########################################################################
      move $s1,$v0         # Save Filename pointer to $s1
      lw    $ra,0($sp)     # Restore return address
      addiu $sp,$sp,4      # Restore stack pointer
      la    $a0,OkFile     # Get ready to print success message
      li    $v0,4          # Tell syscall to print a string
      syscall
```

```
     li   $v0,14
     move $a0,$s1         # pass file pointer
     la   $a1,sudoku      # Where to store the data
     li   $a2,164         # Size of buffer to read in
     syscall
     li   $v0,4
     la   $a0,sudoku
     syscall
     jr   $ra             # Stop Program

# ######################################################################
# Function to read from standard input a filename and open a file.
#
# Send the address of where to store the filename in $a0.
#
# Return the file pointer in $v0.
#
########################################################################
OFile:
     move $t1,$a0         # Move address of where we want the
                          # file name to go to
Again:                    #    $t1
     li   $v0,4           # Tell syscall to print a string
     la   $a0,Hello       # Load address of string to print
     syscall              # Print string
     move $a0,$t1         # Load $a0 with the address of where we want
                          # the
                          #    Filename to go.
     li   $a1,264         # Load max size of string
     li   $v0,8           # Tell syscall to read a string
     syscall              # Read a string
# #######################################################################
# # Ok, we have read in a string.. Now we want to scan the string and  #
# # find a linefeed (the number 10 or hex A) and replace it with binary#
# # zeros which is a null character.                                   #
#
########################################################################
     la   $t2,EOL         # EOL is the character after the filename
                          #    declaration.
     sub  $t3,$t2,$t1     # Subtract the address of the EOL from
                          #    the address of the Filen to get the
length
     move $t4,$t1         # Put address of filename in $t4
GetB:lb   $t5,0($t4)      # load byte into $t5
     li   $s5,10          # Load line feed in $s1
     beq  $t5,$s5,Done    # Go to Done when line feed found
     addiu$t4,$t4,1       # Get next byte
     j    GetB
Done:
     li   $s5,0           # Load zero (null character) into $s1
     sb   $s5,0($t4)      # Replace the line feed with null character

# ######################################################################
# #  Try to open the file,  If it works move the file pointer to $v0   #
# #  and return.                                                       #
# #  If not, go back and read in another filename.
########################################################################

     li   $v0,13                  # tell syscall to open a file
     move $a0,$t1                 # Move address of file in $a0
     li   $a1,0                   # Open for reading
     li   $a2,0                   # No purpose
     syscall                      # Open file
     move   $s6,$v0
     ble  $s6,$zero,Again         # Bad file, try it again.
     move $v0,$s6
     jr   $ra
```