Lab 3: Sudoku Validator

CMPUT 229

University of Alberta

What is a Sudoku

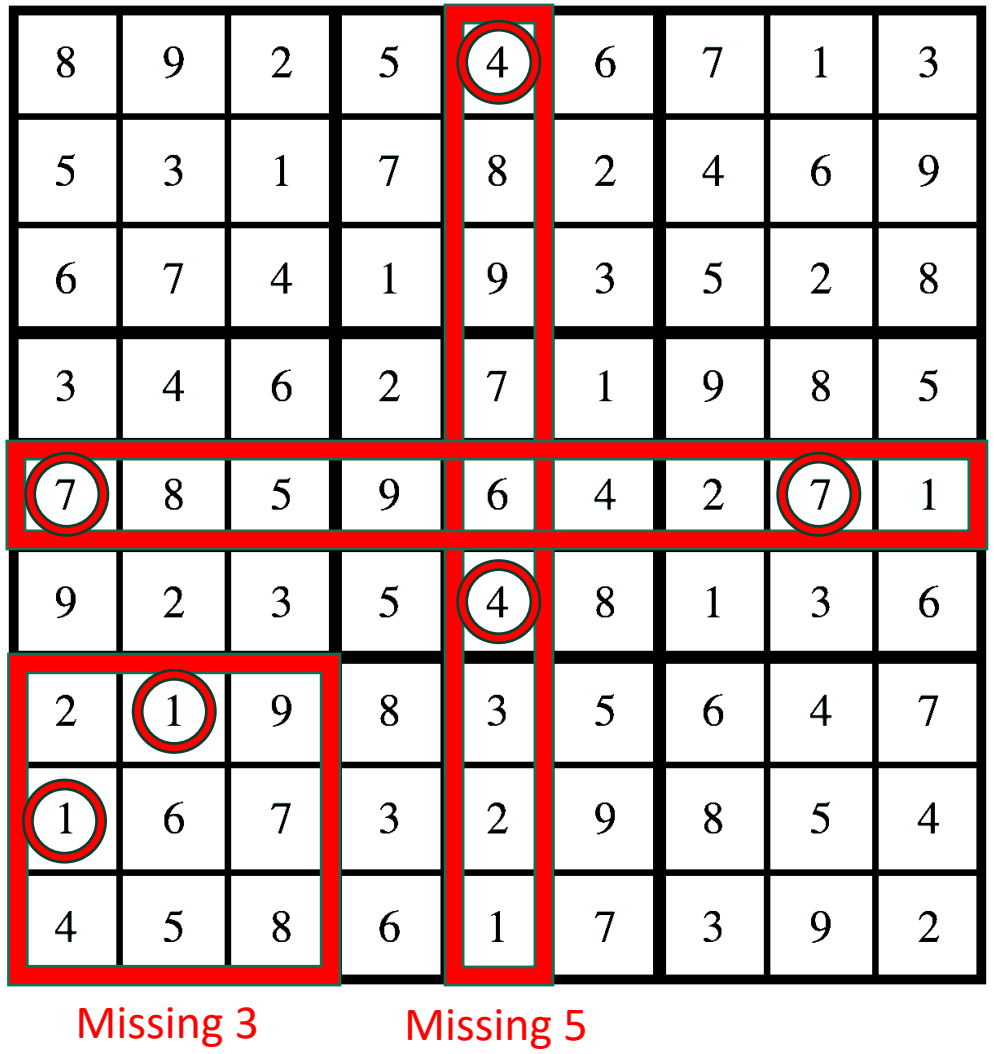
9x9 grid where every row, column, and region contains numbers 1 to 9

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

The Assignment

Given the string of a complete Sudoku, determine if it is valid or invalid. It is valid if every row, column, and region contains all digits 1 to 9 (or no digit is repeated).

The red boxes show invalid examples of each:



Subroutines

sudokuValidator:

This function checks the validity of a given Sudoku.

Input:

- a0: pointer to a 2D 9x9 array of the input Sudoku where each item is a one byte integer

Output:

- a0: Sudoku validity
  - returns 1 if valid, 0 if not
- a1: valid rows (see slide 7)
- a2: valid columns (see slide 7)
- a3: valid regions (see slide 7)

Subroutines

validateArea:

This function checks if a given row, column, or region is valid.

Input:

- a0: pointer to a 2D 9x9 array of the input Sudoku where each item is a one byte integer
- a1: the type of area to check
  - 0 for a row
  - 1 for a column
  - 2 for a region
- a2: integer index of the row, column, or region to validate

Output:

- a0: row, column, or region validity
  - return 1 if valid, 0 if not

sudokuValidator Bit Patterns

Working with bits increases the efficiency of programs.

Return registers a1-a3 in sudokuValidator contain the validity of each row, column, and region.

All the information can be stored directly in the register, rather than needing to point to an array.

- Each row, column, and region is given an index number
- The first 9 bits in each register will correspond to an index number
- A bit will be 1 if its area is valid, otherwise 0
- All other bits must be 0

Mapping with Example

	0	1	2	3	4	5	6	7	8
0									
1		0			1			2	
2									
3									
4		3			4			5	
5									
6									
7		6			7			8	
8									

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

Sudoku is not valid.

Valid rows are: 00000000000000000000000111001101

Valid cols are: 00000000000000000000000101101111

Valid regs are: 00000000000000000000000111001011

Mapping with Example

	0	1	2	3	4	5	6	7	8
0									
1		0			1			2	
2									
3									
4		3			4			5	
5									
6									
7		6			7			8	
8									

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

Sudoku is not valid.

Valid rows are: 00000000000000000000000111001101

Valid cols are: 00000000000000000000000101101111

Valid regs are: 00000000000000000000000111001011

Mapping with Example

	0	1	2	3	4	5	6	7	8
0									
1		0			1			2	
2									
3									
4		3			4			5	
5									
6									
7		6			7			8	
8									

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

Sudoku is not valid.

Valid rows are: 00000000000000000000000111001101

Valid cols are: 00000000000000000000000101101111

Valid regs are: 00000000000000000000000111001011

Mapping with Example

	0	1	2	3	4	5	6	7	8
0									
1		0			1			2	
2									
3									
4		3			4			5	
5									
6									
7		6			7			8	
8									

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

Sudoku is not valid.

Valid rows are: 00000000000000000000000111001101

Valid cols are: 00000000000000000000000101101111

Valid regs are: 00000000000000000000000111001011

Mapping with Example

	0	1	2	3	4	5	6	7	8
0									
1		0			1			2	
2									
3									
4		3			4			5	
5									
6									
7		6			7			8	
8									

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

Sudoku is not valid.

Valid rows are: 00000000000000000000000111001101

Valid cols are: 00000000000000000000001011101111

Valid regs are: 00000000000000000000000111001011

Mapping with Example

	0	1	2	3	4	5	6	7	8
0									
1		0			1			2	
2									
3									
4		3			4			5	
5									
6									
7		6			7			8	
8									

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

Sudoku is not valid.

Valid rows are: 00000000000000000000000111001101

Valid cols are: 00000000000000000000000101101111

Valid regs are: 00000000000000000000000111001011

Mapping with Example

	0	1	2	3	4	5	6	7	8
0									
1		0			1			2	
2									
3									
4		3			4			5	
5									
6									
7		6			7			8	
8									

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

Sudoku is not valid.

Valid rows are: 00000000000000000000000111001101

Valid cols are: 00000000000000000000000101101111

Valid regs are: 00000000000000000000000111001011

Mapping with Example

	0	1	2	3	4	5	6	7	8
0									
1		0			1			2	
2									
3									
4		3			4			5	
5									
6									
7		6			7			8	
8									

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

Sudoku is not valid.

Valid rows are: 00000000000000000000000111001101

Valid cols are: 00000000000000000000000101101111

Valid regs are: 00000000000000000000000111001011

Bit Manipulation

Make things easy by using one bit at a time:

Load bit: li t1, 1

Use sll (shift left logical) to move the bit

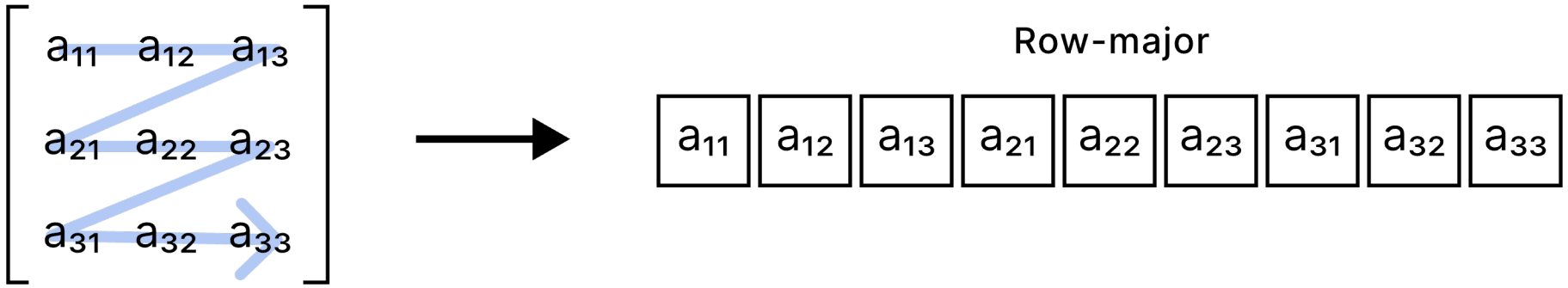
Combining registers:

- and
- or
- xor

2D Arrays

You are given a pointer to a 2D 9x9 array.

This array will be stored in memory the same as an array of length 81 and will be done in row-major order.



Index = (row*9) + col

Only integers 1 to 9 are stored in the array, so each integer is 1 byte rather than the standard 4 bytes.

Functions

Function arguments are set in a0-a7 before a function call.

Function call: jal ra, myFunction

ra stores the return address.

Return values are set in a0-a7 before a function is returned.

Function return: ret

ret returns to the address in ra.

Register Conventions

Register conventions are required for this lab.

Only t and a registers can be different when returning from a function.

S registers, ra, and sp should be unchanged.

Register	Name	Use	Saver
x0	zero	The constant value 0	N.A.
x1	ra	Return address	Caller
x2	sp	Stack pointer	Callee
x3	gp	Global pointer	--
x4	tp	Thread pointer	--
x5-x7	t0-t2	Temporaries	Caller
x8	s0/fp	Saved register/frame pointer	Callee
x9	s1	Saved register	Callee
x10-x11	a0-a1	Function arguments/return values	Caller
x12-x17	a2-a7	Function arguments	Caller
x18-x27	s2-s11	Saved registers	Callee
x28-x31	t3-t6	Temporaries	Caller

Stack

To use s registers, or call another function, the s or ra registers need to be saved. This is done with the stack.

After the registers are saved to the stack, they can be used freely.

Before returning from the function the registers are restored from the stack

Example

```
myFunction:
        addi    sp, sp, -12
        sw      ra, 0(sp)
        sw      s1, 4(sp)
        sw      s2, 8(sp)

        # function code here

        lw      ra, 0(sp)
        lw      s1, 4(sp)
        lw      s2, 8(sp)
        addi    sp, sp, 12
        ret
```

Tips and Notes

- Loop examples (and more) found here:
  - https://cmput229.github.io/229-labs-RISCV/RISC-V-Examples_Public/example.html
- The Sudokus used for grading will only use numbers 1 to 9 and will always be a complete 9x9 Sudoku
- Make sure to return at the end of your sudokuValidator function
- Use the provided example .txt files to test your lab and compare results to the .out files.
  - The tests provided to you are not extensive.
- Do not edit any part of common.s
- Do not use any labels used in common.s
- You are encouraged to make your own helper functions.