Mips Assembler

This project uses MIPS Assembly Language in order to assemble MIPS instructions into their proper binary format so that it can be read by the MIPS processor/simulator and execute the instructions. Open up the "assembler.asm" file and point your file to assemble by changing the `file_location:` variable. Pass in a file to test with a ".asm" extension. Make sure your test file ends with a ".end" in a new line, this means the end of the file and that's what we use to know the file ends. Open up "printFile.asm" and change the file path for the output location, this will create a file for you too. Now go ahead and run the program. Be particular about extra spacing and lack of spacing. Try to format your files something like this:

```
add $t0,$t4,$t3
add $t0,$t4,$t1
.end
```

We have put together a mini-MIPS assembler that can process programmed MIPS code into binary. Currently, our implementation only supports that of Type-R. Here is how we have done it:

1. Open and Read File
2. We begin processing each line, and look at it using characters
3. We then get each unique parameter on a given line that is separated by either spaces or commas. We make this simpler by converting all of the commas to spaces for easier parsing
4. We then process one by one these operations and operands against a lookup table.
5. Once we lookup the right operand and get its type, we call the processOperands method and begin to order operands in the correct format, we store these values in memory.
6. Once we have get the last of our operands, we assemble our code and put out the corresponding machine code.

Supported Commands by Our MIPS Program

Instruction	Type
add	R
and	R
nor	R
or	R
slt	R
sltu	R
sub	R
subu	R

General Helper Functions

Public Methods	args	Description	return
StrEq	$a0, $a1	This method will compare strings the two strings in the args, returns 1 if same	$v0
StrCpy	$a0, $a1	This method will copy the contents of a string in $a1 to $a1	void
isSpace	$a0	This method compares a character and checks if it is a space or no, returns 1 if true	$v0
isEOF	$a0	This method compares a character and checks if it is the end of the file	$v0
handleRType	$a0,$a1,$a2	This method takes in the parameters for building a function. It is called by both processOperand and processOperation files. $a0 is the argument number, $a1 holds the op/operand/shamt codes, and $a2 is the function code	void
processOperation	$a0	This method takes in the parameters after parsing anything. We perform our lookup of the value on $a0	void
processOperand	$a0,$a1	If ProcessOperation could not find the right value, this method is called because it must be a operand. We then handle the type.	void

Registers to Make Note Of:

Global Registers	Description
$S7	Used by the HandleOperand types to know the current number of operands
$S5	Used as a general register to keep track of every new line, hence as a PC counter