Problem PA-1 (2 parts) Function and Stack

Part A Complete this MIPS function that takes three inputs: an integer in `$3`, a running sum in `$5`, and a count in `$7`. If the integer in `$3` is odd, it adds it to the running sum in `$5` and increments the count in `$7`. You may use additional registers.

Label	Instruction	Comment
CountOdd:	**andi $8, $3, 1**	**# is integer in $3 even?**
	beq $8, $0, Return	**# then return to caller**
	add $5, $5, $3	**# else add $3 to running sum $5**
	addi $7, $7, 1	**# increment count in $7**
Return:	**jr $31**	**# return to caller**

Part B The following MIPS function computes the average of all the odd numbers in an integer array starting at location labeled `Array`. Registers are used as follows:

`$1`	Array index	`$5`	Running sum of odd numbers; output average
`$2`	Size of the array in bytes	`$7`	Running count of odd numbers
`$3`	Current Array element	`$8`	Predicate register

Complete the routine by adding MIPS code to preserve registers before the `jal` by pushing them on the stack and to restore them after the function call. Assume `CountOdd` can modify any registers, not just the ones your version modified in Part A.

OddAvg:	addi $1, $0, 0	# init Array index
	addi $2, $0, 400	# init Array size
	addi $5, $0, 0	# init Array index
	addi $7, $0, 0	# init Array index
Loop:	lw $3, Array($1)	# load in current element
	addi $29, $29, -12	**# adjust SP: make room for 3 words**
	sw $1, 0($29)	**# push $1 (preserve on stack)**
	sw $2, 4($29)	**# push $2**
	sw $31, 8($29)	**# push return address**
	jal CountOdd	# in: $3, $5, $7; out: $5, $7
	lw $1, 0($29)	**# pop $1 (restore from stack)**
	lw $2, 4($29)	**# pop $2**
	lw $31, 8($29)	**# pop return address**
	addi $29, $29, 12	**# adjust SP: deallocate 3 words**
	addi $1, $1, 4	# inc Array index
	bne $1, $2, Loop	# if end not reached, loop
	div $5, $7	# odd running sum / odd count
	mflo $5	# put odd number avg in $5
	jr $31	# return to caller