Question 2 (30 points): Now that MIPS-48 has been out on the market for awhile, you have been tasked with analyzing how frequently these new 16 registers are being used as register `rs` in branch instructions. Write the two MIPS procedures specified below. Do not use pseudoinstructions in your code. Your procedures must follow calling conventions for register usage. You may assume that all the instructions you are analyzing are MIPS-48 branch instructions.

Part A (15 points): Write a MIPS procedure `is_new` that takes in the address of a MIPS-48 instruction and determines if register `rs` in the instruction is one of the new registers (numbered 32 to 47). The address of the instruction will be in `$a0`, and the procedure should return `1` in `$v0` if register `rs` is new, and `0` otherwise.

"General" solution (works for cutoffs other than 32)

```
is_new:  lw   $t0, 0($a0)   # load the instruction
         srl  $t0, $t0, 20
         andi $t0, $t0, 0x3F  # isolate rs
         slti $t1, $t0, 32
         beq  $t1, $zero, new # new if ≥ 32
         add  $v0, $zero, $zero  # return 0 if < 32
         j end
new:     addi $v0, $zero, 1   # return 1 if ≥ 32
end:     jr $ra
```

Since this is a leaf procedure, we can use $t-registers and then are not required to save to the stack.

Alternative solution: If rs is an "old" register (0-31), then its leftmost bit is 0. If rs is a "new" register (32-47) then its leftmost bit is 1. This bit corresponds to what we need to return.

```
is_new:  lw   $t0, 0($a0)
         srl  $t0, $t0, 25
         andi $v0, $t0, 0x0001
         jr $ra
```

Part B (15 points): Write a MIPS procedure **count_new** that counts the number of instructions in an array of branch instructions that use the new registers for register **rs**. Your procedure should call **is_new** from Part A. The address of the base of the instruction array will be in **$a0** and the number of instructions in the array will be in **$a1**. The procedure should return the number of instructions using the new registers for register **rs** in **$v0**.

Since this is a non-leaf procedure, if we use the $t registers we will have to save/restore them before/after each call to is_new. If we use the $s registers, then they only need to be saved/restored at the beginning and end of the procedure. We must save/restore $ra too.

```
        addi $sp, $sp, -16
        sw   $s0, 12($sp)
        sw   $s1, 8($sp)      # save registers
        sw   $s2, 4($sp)
        sw   $ra, 0($sp)
        add  $s0, $a0, $zero  # initialize address of first instructio
        add  $s1, $a1, $zero  # initialize counter of instructions remainin
        add  $s2, $zero, $zero # initialize count of new ~~or~~ registers
loop:   beq  $s1, $zero, end  # end if all instructions checked
        add  $a0, $s0, $zero  # put next address in $a0
        jal  is_new
        add  $s2, $s2, $v0    # $v0 is 1 if new, 0 otherwise
        addi $s0, $s0, 4      # increment to next address
        addi $s1, $s1, -1     # decrement instructions remaining
        j    loop
end:    lw   $s0, 12($sp)
        lw   $s1, 8($sp)      # restore registers
        lw   $s2, 4($sp)
        lw   $ra, 0($sp)
        addi $sp, $sp, 16
        jr   $ra
```