1. Review the instructions we have learned:

addi, and, andi, or, ori, xor, xori, sll, srl, beq, bne, j

mainly explain the constraints and concepts involved:

addi: the number is hard coded in the instruction, represented in 16 bits as a signed number

and, andi: anything and with 0 is 0, and with 1 is itself, most commonly used to extract a bit or a set of bits in a register

or, ori: anything or with 0 is itself, or with 1 is 1

xor, xori: output is 1 if the two input bits are different

srl, sll: moving bits in a register to a direction, moving L locations is to divide/multiply by 2 to the power of L

beq (bne): conditional jump, what does a label mean, if not jump, go to the next instruction

j: unconditional jump

1. Write MIPS assembly code that implements the following: if $t0 is storing an odd number, increment $t0 by 6; otherwise, divide it by 4. Note that we only keep the quotient in $t0 after the division.

 andi $t1, $t0, 1

beq $t1, $0, P6\_L1

addi $t0, $t0, 6

j P6\_L2:

P6\_L1:sll $t0, $t0, 2

 P6\_L2: