  2.20  Note: Register bits are numbered from left to right 31 to 0.

            2.25 This is a 'repeat' instruction that tests a counter value,

                     subtracts one from the counter value, and branches to the label 'loop'.

            2.26  Note: there is no 'subi' instruction. The textbook has a 'typo'.

**2.20** [5] <§2.6> Find the shortest sequence of MIPS instructions that extracts bits

16 down to 11 from register $t0 and uses the value of this field to replace bits 31

down to 26 in register $t1 without changing the other 26 bits of register $t1.

srl $t2, $t0, 10 # shift $t0 right 10  
sll $t2, $t2, 26 # bit 0-25 = 0, bits 26-31 = bits 11-16 ($t0)

sll $t1, $t1, 6

srl $t1, $t1, 6 # zero out bits 26-31  
add $t1, $t2, $t1 # place bits 11-16 ($t0) in bits 26-31 of $t1)

**2.25** The following instruction is not included in the MIPS instruction set:

rpt $t2, loop # if(R[rs]>0) R[rs]=R[rs]−1, PC=PC+4+BranchAddr

**2.25.1** [5] <§2.7> If this instruction were to be implemented in the MIPS

instruction set, what is the most appropriate instruction format?

I-Type

**2.25.2** [5] <§2.7> What is the shortest sequence of MIPS instructions that

performs the same operation?

LOOP: slt $t3, $zero, $t2 # if($t2 < 0) {$t3 = 1} else {$t3 = 0}

beq $t3, $zero, END # close loop if $t3 == 0  
 sub $t2, $t2, 1 # $t2 -= 1  
 j LOOP # go back to beginning of loop  
END:

**2.26** Consider the following MIPS loop:

LOOP: slt $t2, $0, $t1

beq $t2, $0, DONE

subi $t1, $t1, 1

addi $s2, $s2, 2

j LOOP

DONE:

**2.26.1** [5] <§2.7> Assume that the register $t1 is initialized to the value 10. What

is the value in register $s2 assuming $s2 is initially zero?

b = 10 \* $t1

At the end of the loop the value in register $t1 is 20

**2.26.2** [5] <§2.7> For each of the loops above, write the equivalent C code

routine. Assume that the registers $s1, $s2, $t1, and $t2 are integers A, B, i, and

temp, respectively.

while(i > 0) {  
 i -= 1 // lower i by 1

b += 2 // increment b by 2 for each iteration

}

**2.26.3** [5] <§2.7> For the loops written in MIPS assembly above, assume that

the register $t1 is initialized to the value N. How many MIPS instructions are

executed?

for N = 0 -> 2 instructions executed  
for N = 1 -> 7 instructions executed

for N = 2 -> 12 instructions executed

…

# instructions = 5N +2 (assuming N is positive; 2 instructions run if N is negative)

For each of these pseudo-instructions, write the MIPS hardware instructions that the Assembler would use to implement them.  Do not destroy the values in the source registers.  ( Use only the minimum number of instructions needed.)

     abs  $t1, $t0     #  absolute value;  page A-51

blt, $t0, $zero, END # check if negative  
nor $t2, $t0, $zero # nor with $zero will invert all the bits

addi, $t2, $t2, 1 # adding 1 to the inverted 2’s compliment number will invert the sign

POS: add, $t1, $t2, $zero # move value from register $t2 to $t1

     rol  $t7, $t6, 8   #  rotate left;  page A-56

sll, $t0, $t6, 8 # store last 24 bits in $t0  
 srl $t1, $t6, 24 # store the first 8 bits in $t1  
 or $t7, $t0, $t1 # combine with or to get result using OR

     ld   $t2, 0($t8)   # load double;  page A-67

lw $t2, 0($t8) # load the first 32 bits of $t8 into $t2  
 lw $t3, 5($t8) # load the second 32 bits of $t8 into $t3 (2^5 = 32 shift)