```
1.      1.   Generate MIPS instructions that compute: a = 200.
addi $s0, $zero, 200     #store 200 into $s0, store in s0

2.      2.   Generate MIPS instructions that compute: a = a - 1.
subi $s0, $s0, 1                #subtract 1 from s0, store in s0

3.      3.   Generate MIPS instructions that compute: a = a - b - c.
        sub $s0, $s0, $s1       #subtract b from a, store in s0
        sub $s0, $s0, $s2       #subtract c from s0

4.      4.   Generate MIPS instructions that compute: a = b - (c + d).
        add $t0, $s2, $s3               #add c and d
        sub $s0, $s2, $t0               #subtract t0 from b, store in a

5.      5.   Generate MIPS instructions that compute: a = 3 * (b + c). (We haven
’t learned an instruction for multiplication yet, but you don’t need one and sho
uldn’t use one.)
add $t0, $s1, $s2               #add b and c
add $s0, $s0, $t0               #add once
add $s0, $s0, $t0               #add twice
add $s0, $s0, $t0               #add three times



6.      6.   Assume that $s0 holds the base address for an array of integers and
 $s1 holds a value used as an index. Write assembly code that doubles the intege
r at the specified position and stores the result back into the appropriate loca
tion in memory. Do not use branches or loops.
.data
        array:   .word 1,2,3,4,5,6 #example array

        la $s0, array           #load base address into s0
        lw $t0, $s1($s0)                #load index, store in t0
        add $t0, $t0, $t0               #double t0, store in t0
        sw $t0, $s1($s0)                #put t0 back into array
```