1. For both of these problems, we have to first convert the hexadecimal numbers to binary and then add them. In MIPS speak, add the values in register $s0 and $s1, and store the sum in register $t0.
   1. CFFFFFFF

1100 1111 1111 1111 1111 1111 1111 1111

C0000000

1100 0000 0000 0000 0000 0000 0000 0000

1100 1111 1111 1111 1111 1111 1111 1111

+ 1100 0000 0000 0000 0000 0000 0000 0000

1100 0111 1111 1111 1111 1111 1111 11111

The number does not overflow

* 1. CFFFFFFF

1100 1111 1111 1111 1111 1111 1111 1111

A0000000

1010 0000 0000 0000 0000 0000 0000 0000

1100 1111 1111 1111 1111 1111 1111 1111

+ 1010 0000 0000 0000 0000 0000 0000 0000

-------------------------------------------------------------------------------------------------

1011 0111 1111 1111 1111 1111 1111 11111

The number overflows out of a 32 bit integer

1. Mul is simply a multiplication instruction in mips. “mul $t0, $s0, $s1” can be done in real MIPS:

mult $s0, $s1 # multiplies $s0 and $s1 and stores result in LO

mflo $t0 # puts LO value into $t0 register

1. 70000030 🡪 0111 0000 0000 0000 0000 0000 0011 0000

00050000 🡪 0000 0000 0000 0101 0000 0000 0000 0000

0111 0000 0000 0000 0000 0000 0011 0000

\*0000 0000 0000 0101 0000 0000 0000 0000

0000 0000 0000 0000 0000 0000 0000 0000

0000 0000 0000 0000 0000 0000 0000 0000

0000 0000 0000 0000 0000 0000 0000 0000//same until 1 appears

when 1 appears ex: 0111 0000 0000 0000 0000 0000 0011…

Final answer is:

0000 0000 0000 0010 0011 0000 0000 0000 0000 0000 1111 0000 0000 0000 0000 0000

HI(first 32 bits): 0000 0000 0000 0010 0011 0000 0000 0000

LO(last 32 bits): 0000 0000 1111 0000 0000 0000 0000 0000

1. 00000050 🡪 0101 0000

00000030 🡪 0011 0000

1

110000 1010000

110000

---------------

100000

LO: 1

HI: 100000

1. The way I thought of it was to keep bitwise-anding $t0 with 1. If the result is not 0, increment the count(in our case $t1). Then keep shifting $t0 right one bit, and do the same thing as before.

Code:

li $t1,0 #setting count equal to 0

count #function declaration

andi $t2, $t0,1 #bitwise and

bne $t2, $0, added #increment

srl $t0, $t0,1 #shift bits

bne $t0, $0, count #if there are more ones, call the function again

j nextFunction

count #other function declaration(basically incrementing and then shifting at the end)

addi $t1, $t1,1

srl $t0,$t0,1

bne $t0, $0,count

6. Code computes the volume of a cylinder.

In C code:

float f0 = 10; //just choosing arbitrary value of height

float f1 = 5; //just choosing arbitrary value of radius

float f2 = 3.14; //value of pie

f2 = f2 \* f0;

f2 = f2 \* f1;

f2 = f2 \* f1;