Q:1. A Benchmark has 50% Integer and 50% Floating Point instructions. A Processor has CPI of 1.3 on Integer instructions and 4.8 on Floating Point instructions. Consider a design options where we can improve the CPI of floating point instructions. What should be the new CPI of FP instructions to make the processor at least twice as fast? How can we make it three times faster? Assume other design parameter like clock speed remain the same.

Let n be the total instructions in the program and c be the clock speed. Then

\[
CPU_{old} = \sum (IC \times CPI \times CC) = 0.5n \times 1.3 \times c + 0.5n \times 4.8 \times c = (0.65 + 2.4) nc = 3.05 nc
\]

**Case 1:**
Let p be the new CPI of floating point instructions. Then

\[
CPU_{new} = \sum (IC \times CPI \times CC) = 0.5n \times 1.3 \times c + 0.5n \times p \times c = (0.65 + 0.5p) nc
\]

To make the computer twice as fast we must have

\[
\frac{1}{2} = \frac{CPU_{new}}{CPU_{old}}
\]

Solving this equation we get \( p = 1.75 \).

**Case 2:**
Let p be the new CPI of floating point instructions. Then

\[
CPU_{new} = \sum (IC \times CPI \times CC) = 0.5n \times 1.3 \times c + 0.5n \times p \times c = (0.65 + 0.5p) nc
\]

To make the computer thrice as fast we must have

\[
\frac{1}{3} = \frac{CPU_{new}}{CPU_{old}}
\]

Solving this equation we get \( p = 0.73 \).
Q:2. Convert the following C Code to MIPS Assembly:

for(count = 0; count<100; count++)
{
}

Assume count is mapped to R1 and x is mapped to R2.

la R3,A     #R3 has the base address of A
la R4,B     #R4 has the base address of B
addi R1,R0,0    #R1 has initialized to initial value of count.
addi R5,R0,100    #R5 has initialized to final value of count.

for:
    slt R11,R1,R5   #check if R1<R11
    be R11,R0,exit   #if so jump to end
    mul R6,R1,4    #R6 = count*4
    add R7,R6,R3    #R7 = address of A[count]
    add R8,R5,R4    #R8 = address of B[count]
    lw R15,0(R7)    #R15 = A[count]
    lw R16,0(R8)    #R16 = B[count]
    add R17,R15,R16   #R17 = R15 + R16
    sub R17,R17,R2   #R17 = R17 – R2 (where R2 = x)
    sw R17,0(R7)   #sets A[counts] = R17
    addi R1,R1,1    #increments count
    b for        #the unconditional jump to top.

exit: