1. (15 points) Suppose that a given optimization results in an OVERALL  
   speedup of 1.5 over the original design. If the optimization speeds up loads and stores, which collectively accounted for 75% of the execution time BEFORE the optimization, by what factor were loads and stores sped up by the optimization?

To find out by what factor ALU (Arithmetic Logic Unit) operations were sped up by the optimization, we can use the concept of weighted speedup. The overall speedup is given as 1.5, and ALU operations account for 75% of the execution time before the optimization. We want to find the speedup factor for ALU operations.

Let S\_total be the overall speedup, S\_ALU be the speedup of ALU operations, and W\_ALU be the weight of ALU operations in the original execution time.

We have: S\_total = 1.5 (given) W\_ALU = 0.75 (75% of the execution time is accounted for by ALU operations)

The weighted speedup formula is given by:

S\_total = (W\_ALU / S\_ALU) + ((1 - W\_ALU) / 1)

Now, we can plug in the values and solve for S\_ALU:

1.5 = (0.75 / S\_ALU) + (0.25/ 1)

To isolate S\_ALU, we subtract 0.60 from both sides:

1.5 - 0.25 = 0.75 / S\_ALU

Now, calculate the left side:

1.25 = 0.75 / S\_ALU

To solve for S\_ALU, take the reciprocal of both sides:

S\_ALU = 0.75 / 1.25

S\_ALU = 0.60

So, ALU operations were sped up by a factor 3.5 when expressed as a decimal or 60% when expressed as a percentage.

**Explanation:**

let's go through the explanation step by step:

**Given Information**: We are given that the overall speedup of a system after optimization is 1.5 times the speed of the original design. Additionally, we know that ALU operations account for 75% of the execution time before the optimization.

**Define Variables**: To find the speedup factor for ALU operations (S\_ALU), we need to set up an equation. Let:

* S\_total be the overall speedup (given as 1.5).
* S\_ALU be the speedup of ALU operations (what we want to find).
* W\_ALU be the weight of ALU operations in the original execution time (given as 75% or 0.75).

**Use Weighted Speedup Formula**: The concept of weighted speedup is used because different parts of a program may have different execution times and speedup factors. The weighted speedup formula is as follows:

S\_total = (W\_ALU / S\_ALU) + ((1 - W\_ALU) / 1)

* The first term (W\_ALU / S\_ALU) represents the contribution of ALU operations to the overall speedup.
* The second term ((1 - W\_ALU) / 1) represents the contribution of the rest of the program (non-ALU operations) to the overall speedup.

**Plug in Values**: Substitute the known values into the formula:

1.5 = (0.75 / S\_ALU) + (0.25 / 1)

* S\_total is 1.5 because it's given.
* W\_ALU is 0.75 because ALU operations account for 75% of the execution time, so 1 - W\_ALU is 0.25 (the remaining portion).

**Isolate S\_ALU**: Rearrange the equation to solve for S\_ALU. First, subtract 0.60 from both sides:

1.5 - 0.25 = 0.75 / S\_ALU

This step isolates the ALU portion on the left side.

**Simplify**: Calculate the left side of the equation:

1.25 = 0.75 / S\_ALU

**Solve for S\_ALU**: To find S\_ALU, take the reciprocal of both sides of the equation:

S\_ALU = 0.75/1.25

S\_ALU = 3.5

**Result**: The speedup factor for ALU operations (S\_ALU) is approximately 3/5. it's approximately 60%.

So, the optimization improved the speed of ALU operations by a factor of approximately 3/5 or 60%

1. (15 points) If, in problem one, the described optimization could make loads and stores take no time at all (not realistic, just for the sake of argument), what would the overall speedup to the execution time be?

Percentage of execution time by ALU operations = 75%

Assuming all operation other than ALU are Loads and stores.

25% of execution time is taken by LOAD and STORE instruction before optimization.

After optimization execution time for Loads and stores instructions will be 0.

speedup = 1/(Percentage of execution time by ALU operations/100)

speedup = 1/(75/100) = (1/0.75) = 4/3 = 1.33

Therefore the overall speedup to the execution time is 1.33

1. (30 points) Suppose that in problem one, the fraction of time taken for loads  
   and stores AFTER optimization is 25%, and overall speedup is 1.5, what  
   would the fraction of time taken by loads and stores BEFORE optimization?  
   What would the speedup to loads and stores be?

1.5/(1-0.25)+(0.25/1.5) = 1.63

1/(1.63/100) = 1/0.0163 = 61.34969325 = 61. 34% speedup

4. (30 points) For a PDP-8, generate assembly code and binary to multiply the  
number in hex address 0x200 by 4, and store the result in address 0x201.  
The program should start in address 0x100. You can assume the number in  
0x200 is positive and less than 0x100. You will need to consult the Internet  
for the PDP-8 mnemonics and instruction formats. Note in particular that the  
PDP-8 has no multiply instruction. Be sure to give the address of each  
instruction.

The PDP-8 doesn't have a multiply instruction, but it can be done using repeated additions. The following code will multiply the number stored at address 0x200 by 4 and store the result at address 0x201:

0x100: AND 0x200 # load the number from address 0x200

0x101: TAD 0x200 # add the number to itself

0x102: TAD 0x200 # add the number to itself again

0x103: DCA 0x201 # store the result at address

0x201 0x104: HLT # stop the program

Here's what happens in each step:

AND 0x200 loads the number from address 0x200 into the accumulator

TAD 0x200 adds the number to itself, doubling it

TAD 0x200 adds the number to itself again, doubling it to four times the original value DCA 0x201 stores the result at address 0x201 HLT stops the program

5. (10 points) If a given processor has a dynamic instruction count that is 20%  
ALU instructions, 40% loads and stores, and 20% jumps, and if ALU  
Instructions take 1 cycle to execute, load or store take 5 cycles, jumps take 3  
cycles, and on average all the other instructions not listed take 2.3 cycles,  
what is the average CPI for the processor?

(0.20 \* 1) + (0.40 \* 5) + (0.20 \* 3) + (0.20 \*2.3) = 3.26