Computer Sciences Department
University of Wisconsin-Madison
CS/ECE 552 – Introduction to Computer Architecture
In-Class Exercise (01/23) SOLUTION

Answers to all questions should be uploaded on Canvas

1. [3 points] From the textbook

(a) [1 point] (Twist on Check Yourself 1.5) A key factor in determining the cost of an integrated circuit is volume. Which of the following are reasons why a chip made in high volume should cost less?

(i) With high volumes, the manufacturing process can be tuned to a particular design, increasing yield.
(ii) It is less work to design a high-volume part than a low-volume part.
(iii) The masks used to make the chip are expensive, so the cost per chip is lower for higher volumes.
(iv) Engineering development costs are high and largely independent of volume, thus, the development cost per die is lower with high-volume parts.
(v) High volume parts usually have smaller die sizes than low-volume parts and therefore have higher yield per wafer.

Given that (i), (iii), and (iv) are correct, why are (ii) and (v) incorrect?

Solution:

NOTE: although (i) is true, in my opinion, it is a secondary effect.

(ii) is definitely not true – designing any sized part is hard, regardless of volume.

(v) is true in some cases, although not generally true. As the book states, higher volume parts could/often do make extra investments to make die size reductions worthwhile economically, but this is not required nor is it always done.

(b) [1 point] (Section 1.6) A program *P* has 2500 instructions, each of which takes 2 cycles to complete. If we run *P* on a processor that has a clock speed of 100 MHz, how long does this program take to execute?

Hint: see page 38 of the textbook.

Solution:

$$\text{Time} = 2500 * 2 * \left(\frac{1}{100\text{E}6}\right) = 0.00005\ \text{s} = 0.05\ \text{ms}$$

(c) [1 point] (Section 1.7) Power is becoming an increasingly important issue to consider when designing processors. Suppose you are the chief architect at a small startup designing an IOT device. Given the tight deadline to ship your product, you only have enough time to make one optimization – reduce frequency by half or decrease the voltage by 25%. Which optimization will be more beneficial?

Solution:

See page 41 in your textbook. Voltage affects power by the square – so the new power is $(3/4)^2$*P = (9/16)*P. Frequency affects power linearly, so normally reducing voltage is better, but reducing frequency by half will lead to a new power of P/2. Thus, frequency reduction is a better choice here.

2. [1 point] Consider a program running on your computer. You apply a technique that accelerates non-memory instructions in your program by 4x. This speeds up the entire program execution by 3x. Now would it be better for performance to (i) eliminate half of the memory instructions in your program or (ii) improve the technique such that it accelerates non-memory instructions by 6x?

Solution:

First we need to find the fraction, *f*, of instructions that are non-memory instructions, since these are the instructions our optimization helps with. To find *f*, we can use Amdahl's Law, since we know that only the portion of the program that is non-memory instructions is sped up by the optimization:

$$\frac{1}{1-f+\frac{f}{4}} = 3$$

Solving for f, we get f = 8/9. This means that 8/9ths of the instructions are non-memory instructions.

Now, we can analyze whether (i) or (ii) is better. To do so, we can again use Amdahl's Law, but now we adjust the terms according to what (i) or (ii) changes.

(i) Eliminating half of the memory instructions is like making half of the memory instructions (which make up 1/9*1/2 = 1/18th of the instructions in the program) infinitely fast. Thus:

$$\frac{1}{\left(\frac{1-f_{nonmem}}{2}\right)+\frac{\left(\frac{1-f_{nonmem}}{2}\right)}{\infty}+\frac{(f_{nonmem})}{4}}$$

$$\frac{1}{\left(\frac{1}{18}\right)+\frac{\left(\frac{1}{18}\right)}{\infty}+\frac{\left(\frac{8}{9}\right)}{4}} = 3.6X$$

(ii) Here we simply change the speedup for the fraction of non-memory instructions to 6 instead of 4:

$$\frac{1}{\left(\frac{1}{9}\right)+\frac{\left(\frac{8}{9}\right)}{6}} = 3.86X$$

Thus (ii) is a better choice.

3. [1 point] Consider a processor architecture with an average CPI of 1.8. You propose a change to the architecture that is able to eliminate half of all memory read instructions in the program but incurs a single-cycle overhead on all memory write instructions. Given the typical instruction breakdowns that you expect in your programs (table below), is this change a worthy trade-off for performance?

% of Program	Instruction Type
10%	Addition/subtraction
5%	Multiplication
40%	Memory read
10%	Memory write
15%	Logic
20%	Branch

Solution:

After removing half of memory read instructions, the % of memory writes = 0.1 / (1 – 0.4/2) = 0.125.

(Another way to think about this: if we had 100 instructions originally, now we have 80 instructions after removing the 20 memory reads. 10 memory writers / 80 total instructions = 0.125)

Now we can apply the speedup equation, with our new values for the number of accesses, to see when this tradeoff is worthwhile:

$$Speedup = \frac{execution\ time_{old}}{execution\ time_{new}}$$

$$Speedup = \frac{instructions_{old} * CPI_{old} * f}{instructions_{new} * CPI_{new} * f}$$

$$Speedup = \frac{\#\ instructions_{old} * CPI_{old}}{\left(\left(1 - \frac{0.4}{2}\right) * \#instructions_{old}\right) * \left(CPI_{old} + 1 * \left(\frac{0.1}{1 - \frac{0.4}{2}}\right)\right)}$$

$$Speedup = \left(\frac{1}{1 - \frac{0.4}{2}}\right) * \left(\frac{1.8}{1.8 + \frac{0.1}{1 - \frac{0.4}{2}}}\right) = 1.17X$$

Thus it would be worthwhile for a 1.17X speedup (i.e., the change helps).

Note: when we apply the speedup equation here, we are using the Iron Law of Performance to compare the relative speedup of the two variants, where the frequency remains constant and thus cancels. The trick though is the dynamic instruction count and CPI are different for the new version (since we remove 20% of reads and the writes take an extra cycle), so we need to determine how to express this information. However, we can express the new version's execution time as a ratio of the old version's execution time – all of the remaining instructions take at least as long as they did before (the left part of the denominator), and the additional cycle for the memory writes must be added to this (the right part of the denominator).

4. [2 points] You build a computer that achieves a SPECratio of 9.5 for the benchmark astar when running at frequency f = 4GHz with a CPI of 1.79. Say that the number of instructions in astar were to increase by 10% and the CPI were to increase by 5%.

(a) [1 point] By how much would astar's execution time increase?

Solution:

$$execution\ time_{new} = \#\ instructions_{new} * CPI_{new} * \left(\frac{1}{f}\right)$$

$$execution\ time_{new} = (1.1 * instructions_{old}) * (1.05 * CPI_{old}) * \left(\frac{1}{f}\right)$$

$$execution\ time_{new} = 1.155 * execution\ time_{old}$$

Thus execution time would increase by 15.5%.

(b) [1 point] What is the new SPECratio?

Solution:

$$SPECratio_{new} = \frac{SPECratio_{old}}{1.155} = \frac{9.5}{1.155} = 8.23$$

5. [3 points]

(a) [1 point] Suppose we propose making a tradeoff in our architecture. Specifically, we propose making a change that reduces the performance of 20% of a program by a factor of 4 (call this "enhancement" *E3*) and improves the performance of another 12% of the program by a factor of 6 (call this enhancement *E4*). Assume the two enhancements are independent and affect different parts of the program (there is no overlap between the 20% and 12% of the program). What is the overall effect on the performance of the program? Is this tradeoff worthwhile?

Solution:

We build on Amdahl's Law here. Somewhat similar to problem 3, since we don't know the old latency (aka, execution time), to solve this problem we can calculate the new latency in terms of the old latency (Note: here one of the enhancements actually hurts performance):

$$new\ latency\ =\ (1-f3-f4) * old\ latency\ +\ \frac{f3}{S3} * old\ latency + \frac{f4}{S4}\ *\ old\_latency$$

(Let f3 = 20%, S3 = 1/4; f4 = 12%, S4 = 6)

$$New\ latency\ =\ (1-0.2-0.12) * old\ latency\ +\ \left(\frac{0.2}{\frac{1}{4}}\right) * old\ latency\ +\ \left(\frac{0.12}{6}\right) * old\ latency$$

$$new\ latency\ =\ old\ latency\ *\ 1.5$$

Plugging in to the Amdahl's Law equation (note: here we are again comparing old vs. new, similar to problem 3), we get:

$$Final\ speedup\ = \frac{old\ latency}{old\ latency\ *\ 1.5}\ =\ 0.67$$

Since the final speedup is < 1, this means that the tradeoff is not worthwhile.

(b) [2 points] Your colleague claims we can overcome *E3* by increasing the clock frequency by 10%. Is this sufficient? Why or why not? Assume the architecture stays the same and the change in frequency does not affect dynamic instruction count or CPI.

Solution:

First, we need to isolate what the effect of *E3* is from part A:

$$new\ latency\ =\ (1-f3)*old\ latency\ +\ \frac{f3}{S3}*old\ latency$$

(Let f3 = 20%, S3 = 1/4)

$$New\ latency\ =\ (1-0.2)*old\ latency\ +\ \left(\frac{0.2}{\frac{1}{4}}\right)*old\ latency$$

$$new\ latency\ =\ old\ latency\ *\ 1.6$$

Plugging in to the Amdahl's Law equation, we get:

$$Final\ speedup\ = \frac{old\ latency}{old\ latency\ *\ 1.6}\ =\ 0.625$$

So, the change from increasing the clock frequency needs to be at least 37.5% to counteract the effect of *E3*. Now, we need to figure out how the change in clock will affect performance. From the lecture slides and Section 1.11 in the textbook, we know that:

$$Latency = \frac{Seconds}{program} = \left(\frac{instructions}{program}\right) * \left(\frac{cycles}{instruction}\right) * \left(\frac{seconds}{cycle}\right)$$

The prompt tells us that dynamic instruction count and CPI are unaffected by the change, so the only thing that does change is the frequency component. Based on this, we know that the difference in latency is:

$$1.1 *\ New\ Latency\ =\ Old\ Latency$$

Now we can plug this into the Amdahl's Law equation:

$$Speedup\ = \frac{old\ latency}{new\ latency} = \frac{1.1 * new\ latency}{new\ latency} = 1.1$$

This means that increasing the clock frequency leads to a speedup of 10%. Thus, this solution helps compared to Part A (where the slowdown was 37.5% for *E3*), but does not completely address the shortcomings of our proposed tradeoff.

NOTE: If you included f4 and S4 in this calculation, that's fine. In this case, you should get the same answer, although the numbers will be slightly different.