ECE-6913 Computer System Architecture
Homework 1

New York University, Fall 2024

Due on 09/19, 11:59 PM

Name: Raman Kumar Jha
NYU ID: N13866145

1. **Assume a program requires the execution of 60 × 106 FP instructions, 120 × 106 INT instructions, 60 × 106 L/S instructions, and 16 × 106 branch instructions. The CPI for each type of instruction is 1, 1, 4, and 2, respectively. Assume that the processor has a 2 GHz clock rate.**

   Given: A program requires execution of:

   - FP instructions: 60×10^6
   - INT instructions: 120×10^6
   - L/S instructions: 60×10^6
   - Branch instructions: 16×10^6

   with CPIs of 1, 1, 4, and 2, respectively. The processor has a clock rate of 2 GHz.

   Let's first compute the original execution time:

$$\text{Total clock cycles} = \sum(\text{Num. of instructions} \times \text{CPI})$$
$$= (60 \times 10^6 \times 1) + (120 \times 10^6 \times 1) + (60 \times 10^6 \times 4) + (16 \times 10^6 \times 2)$$
$$= 60 + 120 + 240 + 32 = 452 \times 10^6$$

$$\text{Execution time (original)} = \frac{452 \times 10^6}{2 \times 10^9} = 0.226\,\text{s}$$

   (a) By how much must we improve the CPI of FP instructions if we want the program to run two times faster?

   Let CPI'_{FP} be the new FP CPI. Then, the new execution time must be $0.226/2 = 0.113\,\text{s}$. The total clocks with improved FP CPI is:

$$\text{New clocks} = (60 \times 10^6)CPI'_{FP} + (120 \times 10^6 \times 1) + (60 \times 10^6 \times 4) + (16 \times 10^6 \times 2)$$

   But Clocks for speedup = 2 is $(0.113)(2 \times 10^9) = 226 \times 10^6$. Thus,

$$60CPI'_{FP} + 120 + 240 + 32 = 226$$
$$60CPI'_{FP} = 226 - 392 = -166$$

   This equation has no solution: It's impossible to reduce the CPI of FP instructions alone enough to speed up the program by a factor of 2, because the FP instructions already have the best possible CPI (1) and are a minority of the workload. Even setting their CPI to 0 is not enough.

   (b) By how much must we improve the CPI of L/S instructions if we want the program to run two times faster?

Now, let $CPI'_{L/S}$ be the new L/S CPI. As above, the sum of the clocks must be 226×10^6. Set up the equation:

$$60 + 120 + (60 \times CPI'_{L/S}) + 32 = 226$$
$$212 + 60CPI'_{L/S} = 226$$
$$60CPI'_{L/S} = 14 \implies CPI'_{L/S} \approx 0.233$$

Improvement factor: $4/0.233 \approx 17.17$ (i.e., CPI for L/S must be reduced over 17 times).

(c) By how much is the execution time of the program improved if the CPI of INT and FP instructions is reduced by 50% and the CPI of L/S and Branch is reduced by 25%?

Calculate new CPIs:

- FP: $1/2 = 0.5$
- INT: $1/2 = 0.5$
- L/S: $0.75 \times 4 = 3$
- Branch: $0.75 \times 2 = 1.5$

$$\text{New cycles} = (60 \times 0.5) + (120 \times 0.5) + (60 \times 3) + (16 \times 1.5)$$
$$= 30 + 60 + 180 + 24 = 294 \times 10^6$$

$$\text{New time} = \frac{294 \times 10^6}{2 \times 10^9} = 0.147\,\text{s}$$

$$\text{Improvement} = \frac{0.226}{0.147} \approx 1.54$$

The program runs about 1.54 times faster.

2. **When a program is adapted to run on multiple processors in a multiprocessor system, the execution time on each processor is comprised of computing time and the overhead time required for locked critical sections and/or to send data from one processor to another. Assume a program requires t = 200 s of execution time on one processor. When running p processors, each processor requires t/p s, as well as an additional 10 s of overhead, irrespective of the number of processors. Compute the per-processor execution time for 2, 4, 8, 16, 32, 64 processors. For each case, list the corresponding speedup relative to a single processor and the ratio between actual speedup versus ideal speedup (speedup if there was no overhead).**

Given: A program requires $t = 200\,\text{s}$ on one processor. On p processors, each processor requires $t/p + 10\,\text{s}$ ($10\,\text{s}$ overhead per processor, regardless of p).

For each $p \in \{2, 4, 8, 16, 32, 64\}$:

- Per-processor execution time: $T_p = 200/p + 10$
- System speedup: $S(p) = 200/T_p$
- Ideal speedup: $S_{\text{ideal}}(p) = p$
- Overhead ratio: $S(p)/S_{\text{ideal}}(p)$

p	T_p (s)	$S(p)$	S/S_{ideal}
2	110	1.82	0.91
4	60	3.33	0.83
8	35	5.71	0.71
16	22.5	8.89	0.56
32	16.25	12.3	0.38
64	13.125	15.2	0.24

3. **Server farms such as Google and Yahoo! provide enough compute capacity for the highest request rate of the day. Imagine that most of the time these servers operate at only 60% capacity. Assume further that the power does not scale linearly with the load; that is, when the servers are operating at 60% capacity, they consume 90% of maximum power. The servers could be turned off, but they would take too long to restart in response to more load. A new system has been proposed that allows for a quick restart but requires 20% of the maximum power while in this "barely alive" state.**

   Given: Server farm operates at 60% capacity most of the time, at 90% of max power. "Barely alive" mode consumes 20% of max power. Power does not scale linearly with load.

   Let P_{max} be the max power consumed by all servers at 100% capacity.

   (a) How much power savings would be achieved by turning off 60% of the servers?

   If 60% are off, only 40% are running at 100% capacity, consuming $0.4P_{max}$.

   Savings: $1 - 0.4 = 0.6$ (60% power saved).

   (b) How much power savings would be achieved by placing 60% of the servers in the "barely alive" state?

   60% barely alive: $0.6 \times 0.2P_{max}$ 40% running: $0.4 \times 0.9P_{max}$

   Total: $0.12 + 0.36 = 0.48P_{max}$

   Savings: $1 - 0.48 = 0.52$ (52% power saved).

   (c) How much power savings would be achieved by reducing the voltage by 20% and frequency by 40%?

   Power scales as $V^2 f$. If $V = 0.8V_{max}, f = 0.6f_{max}$, then:

   $$P = (0.8)^2 \times 0.6 = 0.64 \times 0.6 = 0.384P_{max}$$

   Savings: $1 - 0.384 = 0.616$ (61.6% power saved).

   (d) How much power savings would be achieved by placing 30% of the servers in the "barely alive" state and 30% off?

   30% off, 30% barely alive, 40% running.

   $$0.3 \times 0 + 0.3 \times 0.2 + 0.4 \times 0.9 = 0.06 + 0.36 = 0.42P_{max}$$

   Savings: $1 - 0.42 = 0.58$ (58% power saved).

4. **In a server farm such as that used by Amazon or eBay, a single failure does not cause the entire system to crash. Instead, it will reduce the number of requests that can be satisfied at any one time.**

   Given: 10,000 computers, each MTTF = 35 days. Catastrophic failure if 1/3 of computers fail.

   (a) MTTF for system (allowing 1/3 to fail):

   Assuming independent failures, expected time until $n = 3340$ (rounded up from 1/3 of 10,000) failures is MTBF for the system. For large N, $MTTF_{\text{system}} \approx MTTF_{\text{computer}}/(N \cdot p)$, where p is the fraction that must fail for system failure.

   For 1/3, $p = 1/3$, but in reality, the exact answer comes from reliability engineering, not just division. The correct formula is more complex, but for a rough estimate, $MTTF_{\text{system}} \gg MTTF_{\text{component}}$. In practice, for parallel systems, the system reliability increases exponentially with N, so the MTTF for the system is *much* higher than for a single computer.

   A more formal approach uses the CDF of the binomial distribution, but for a rough calculation, you can use the approximation:

   $$MTTF_{\text{system}} \approx MTTF_{\text{computer}} \times \frac{1}{N \cdot P(\text{failure})}$$

   But this is not accurate for large-scale redundancy. A better simplified answer is: *The system MTTF is much greater than that of a single computer, but a precise value requires more advanced statistical analysis.*

(b) Would it be a good business decision to pay $1000 per computer to double MTTF?

Let $MTTF_{\text{new}} = 70\,\text{days}$. Cost increase: $10,000 \times \$1,000 = \$10,000,000$.

Impact on failures: - Failures per day (current): $10,000/35 \approx 286$ - Failures per day (new): $10,000/70 = 143$

Reduction: $286 - 143 = 143$ fewer failures per day.

If the cost of one failure (e.g., downtime, data loss, replacement, etc.) is $\$C$, then daily savings $= 143C$.

The break-even point (ignoring time value of money) is:

$$10,000,000 = 365 \times 143C \implies C = \frac{10,000,000}{365 \times 143} \approx \$196$$

If the cost of a single computer failure exceeds $196, this is a good investment. Otherwise, it is not.

5. **In this exercise, assume that we are considering enhancing a machine by adding vector hardware to it. When a computation is run in vector mode on the vector hardware, it is 15 times faster than the normal mode of execution. We call the percentage of time that could be spent using vector mode the percentage of vectorization. Vectors are discussed in Chapter 4, but you don't need to know anything about how they work to answer this question!**

Given: Vector hardware makes computation 15 times faster than normal mode. We analyze speedup vs percentage of vectorization using Amdahl's Law: $S = \frac{1}{(1-p)+\frac{p}{15}}$ where p is the fraction vectorized.

(a) Graph of speedup vs percentage vectorization:

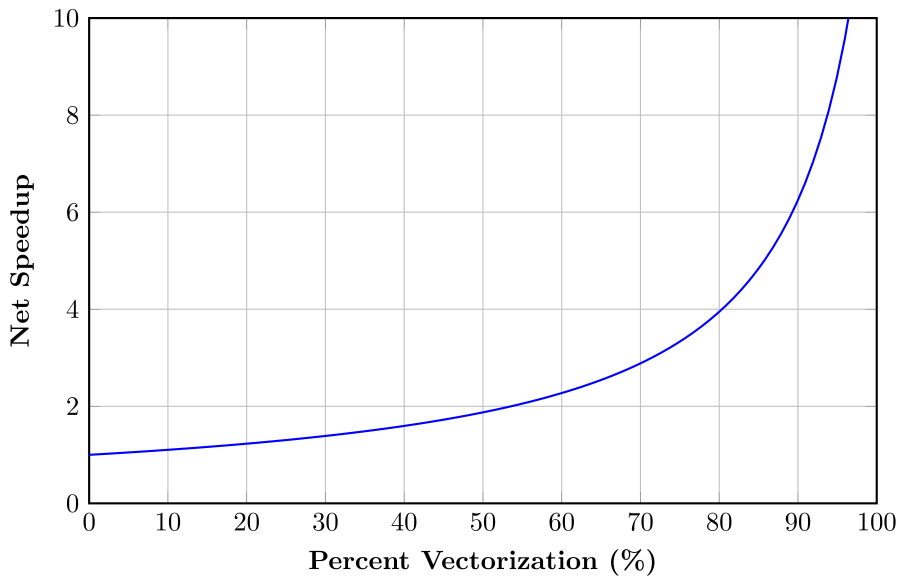
Given:

$$S = 15$$

Let f represent the fraction of computation that can be vectorized. Using Amdahl's Law, the net speedup is given by:

$$\text{Speedup} = \frac{1}{(1-f) + \frac{f}{S}}$$

where $f = \frac{\text{Percent Vectorization}}{100}$.



As the percentage of vectorization increases, the net speedup approaches the maximum limit of 15.

(b) What percentage of vectorization is needed to achieve a speedup of 3?

Using Amdahl's Law: $3 = \frac{1}{(1-p)+\frac{p}{15}}$

Solving: $3[(1-p)+\frac{p}{15}]=1$

$$3-3p+\frac{p}{5}=1$$

$$2=3p-\frac{p}{5}=p(\frac{14}{5})$$

$$p=\frac{10}{14}=\frac{5}{7}=71.43\%$$

(c) What percentage of computation run time is spent in vector mode if speedup of 2 is achieved?

First, find vectorization percentage for speedup of 2:

$$2=\frac{1}{(1-p)+\frac{p}{15}}\implies p=\frac{15}{28}=53.57\%$$

Time spent in vector mode as fraction of total execution time:

$$\frac{\text{Vector time}}{\text{Total time}}=\frac{\frac{p\cdot T}{15}}{\frac{T}{2}}=\frac{2p}{15}=\frac{2\times 53.57\%}{15}=7.14\%$$

(d) What percentage of vectorization is needed to achieve one-half the maximum speedup?

Maximum speedup = 15, so half = 7.5:

$$7.5=\frac{1}{(1-p)+\frac{p}{15}}\implies p=\frac{6.5}{7}=92.86\%$$

(e) Current vectorization is 70%. What vectorization is needed with a 2× faster vector unit (30× total) to achieve the same speedup?

Current speedup with 70% vectorization: $S=\frac{1}{0.3+\frac{0.7}{15}}=2.885$

For same speedup with 30× vector unit:

$$2.885=\frac{1}{(1-p)+\frac{p}{30}}\implies p=67.59\%$$

6. **Assume for arithmetic, load/store, and branch instructions, a processor has CPIs of 2, 10, and 5, respectively. Also assume that on a single processor, a program requires the execution of 2.56E9 arithmetic instructions, 1.28E9 load/store instructions, and 128 million branch instructions. Assume that each processor has a 2GHz clock frequency. Assume that, as the program is parallelized to run over multiple cores, the number of arithmetic and load/store instructions per processor is divided by 0.7 × p (where p is the number of processors) but the number of branch instructions per processor remains the same**

Given: CPIs: Arithmetic = 2, Load/Store = 10, Branch = 5. Clock = 2 GHz. Instructions: 2.56×10^9 arithmetic, 1.28×10^9 load/store, 128×10^6 branch. For p processors: arithmetic and load/store divided by $0.7p$, branch unchanged per processor.

(a) Execution times and speedups:

Processors	Execution Time (s)	Speedup
1	13.12	1.000
2	6.72	1.952
4	3.52	3.727
8	1.92	6.833

(b) Required CPI reduction for load/store instructions:

To match 4-processor performance (3.52 s) on single processor:

Target cycles = $3.52\times2\times10^9=7.04\times10^9$

Solving: New $\text{CPI}_{L/S}=\frac{7.04\times10^9-2.56\times10^9\times2-128\times10^6\times5}{1.28\times10^9}=1.0$

The CPI must be reduced from 10 to 1, a **10× improvement**.

7. **Suppose we developed a simpler processor that has 75complex processor. Further, assume that it can adjust voltage so that it can reduce voltage by 20compared to the complex processor, but this results in a 25% increase in frequency.**

   Given: Simpler processor has 75% capacitive load, 20% voltage reduction ($0.8\times$), 25% frequency increase ($1.25\times$).

   (a) Energy savings:

   Energy per operation $\propto C \times V^2$

   $$\text{Energy ratio} = 0.75 \times (0.8)^2 = 0.75 \times 0.64 = 0.48$$

   $$\text{Energy savings} = 1 - 0.48 = 52\%$$

   (b) Dynamic power impact:

   Dynamic power $\propto C \times V^2 \times f$

   $$\text{Power ratio} = 0.75 \times (0.8)^2 \times 1.25 = 0.75 \times 0.64 \times 1.25 = 0.60$$

   The dynamic power **decreases by 40%**.

8. **One challenge for architects is that the design created today will require several years of implementation, verification, and testing before appearing on the market. This means that the architect must project what the technology will be like several years in advance. Sometimes, this is difficult to do.**

   Technology trends and architectural implications:

   (a) Transistor count growth (Moore's Law):

   From 2015 to 2025 = 10 years = 5 doubling periods (every 2 years)

   $$\text{Multiplier} = 2^5 = 32\times$$

   Transistors in 2025 should be **32 times** the number in 2015.

   (b) Performance growth at 1990s rate:

   With 52% annual performance increase:

   $$\text{Performance multiplier} = (1.52)^{10} \approx 66\times$$

   Performance would be **66 times** higher than 2015 levels.

   (c) Clock rate limitations and responses:

   What has limited clock rate growth:

   - Power consumption (dynamic power $\propto f \times V^2$, leakage power)
   - Heat dissipation challenges
   - Physical limits of silicon technology
   - Voltage scaling limitations
   - Wire delays becoming dominant

   How architects use extra transistors:

   - Multiple cores (parallel processing)
   - Larger caches (reduce memory latency)
   - Specialized accelerators (GPUs, AI chips)
   - Advanced branch prediction
   - Out-of-order execution improvements
   - SIMD/vector processing units
   - Hardware security features

- Power management circuits

9. **General-purpose processes are optimized for general-purpose computing. That is, they are optimized for behavior that is generally found across a large number of applications. However, once the domain is restricted somewhat, the behavior that is found across a large number of the target applications may be different from general-purpose applications. One such application is deep learning or neural networks. Deep learning can be applied to many different applications, but the fundamental building block of inference—using the learned information to make decisions— is the same across them all. Inference operations are largely parallel, so they are currently performed on graphics processing units, which are specialized more toward this type of computation, and not to inference in particular. In a quest for more performance per watt, Google has created a custom chip using tensor processing units to accelerate inference operations in deep learning.1 This approach can be used for speech recognition and image recognition, for example. This problem explores the trade-offs between this process, a general-purpose processor (Haswell E5-2699 v3) and a GPU (NVIDIA K80), in terms of performance and cooling. If heat is not removed from the computer efficiently, the fans will blow hot air back onto the computer, not cold air. Note: The differences are more than processor—on-chip memory and DRAM also come into play. Therefore statistics are at a system level, not a chip level.**

Given: Three systems for deep learning workloads:

- General-purpose: Haswell E5-2699 v3 (TDP: 504W, Idle: 159W, Busy: 455W)
- Graphics processor: NVIDIA K80 (TDP: 1838W, Idle: 357W, Busy: 991W)
- Custom ASIC: TPU (TDP: 861W, Idle: 290W, Busy: 384W)

Performance data:

System	**Throughput**			**% Max IPS**		
	A	B	C	A	B	C
General-purpose	5,482	13,194	12,000	42%	100%	90%
GPU	13,461	36,465	15,000	37%	100%	40%
TPU	225,000	280,000	2,000	80%	100%	1%

(a) TPU speedup over GPU system (70% A, 30% B workload):

Weighted throughput calculations:

$$\text{GPU weighted} = 0.70 \times 13,461 + 0.30 \times 36,465 = 20,362 \text{ ops} \quad (1)$$
$$\text{TPU weighted} = 0.70 \times 225,000 + 0.30 \times 280,000 = 241,500 \text{ ops} \quad (2)$$

$$\text{Speedup} = \frac{241,500}{20,362} = \mathbf{11.86}$$

(b) Percentage of Max IPS for each system (70% A, 30% B):

- General-purpose: $0.70 \times 42\% + 0.30 \times 100\% = 59.4\%$
- Graphics processor: $0.70 \times 37\% + 0.30 \times 100\% = 55.9\%$
- Custom ASIC (TPU): $0.70 \times 80\% + 0.30 \times 100\% = 86.0\%$

(c) Performance per watt of TPU over GPU:

Power consumption (linear scaling from idle to busy):

$$\text{Power} = \text{Idle} + (\text{Busy} - \text{Idle}) \times \frac{\text{IPS\%}}{100}$$

$$\text{GPU power} = 357 + (991 - 357) \times \frac{55.9}{100} = 711.4 \text{ W} \quad (3)$$
$$\text{TPU power} = 290 + (384 - 290) \times \frac{86.0}{100} = 370.8 \text{ W} \quad (4)$$

Performance per watt:

$$\text{GPU:} \quad \frac{20,362}{711.4} = 28.62 \text{ ops/W} \tag{5}$$

$$\text{TPU:} \quad \frac{241,500}{370.8} = 651.22 \text{ ops/W} \tag{6}$$

$$\text{TPU advantage} = \frac{651.22}{28.62} = \mathbf{22.75}$$

(d) Speedups with different workload (40% A, 10% B, 50% C):

Weighted throughput:

$$\text{General-purpose:} \quad 0.40 \times 5,482 + 0.10 \times 13,194 + 0.50 \times 12,000 = 9,512 \text{ ops} \tag{7}$$

$$\text{GPU:} \quad 0.40 \times 13,461 + 0.10 \times 36,465 + 0.50 \times 15,000 = 16,531 \text{ ops} \tag{8}$$

$$\text{TPU:} \quad 0.40 \times 225,000 + 0.10 \times 280,000 + 0.50 \times 2,000 = 119,000 \text{ ops} \tag{9}$$

Speedups over general-purpose:

- GPU speedup: $\frac{16{,}531}{9{,}512} = \mathbf{1.74}$
- TPU speedup: $\frac{119{,}000}{9{,}512} = \mathbf{12.51}$

(e) Server cooling capacity:

Cooling door capacity: 14 kW = 14,000 W

Servers per cooling door:

- Haswell servers: $\lfloor \frac{14{,}000}{504} \rfloor = \mathbf{27}$ servers
- NVIDIA servers: $\lfloor \frac{14{,}000}{1{,}838} \rfloor = \mathbf{7}$ servers
- TPU servers: $\lfloor \frac{14{,}000}{861} \rfloor = \mathbf{16}$ servers

(f) Rack density limitations:

Rack area: 11 square feet, Power limit: 200 W/sq ft Maximum rack power: $11 \times 200 = 2,200$ W

Servers per rack (power limited):

- Haswell: $\lfloor \frac{2{,}200}{504} \rfloor = \mathbf{4}$ servers, $\lceil \frac{4}{27} \rceil = \mathbf{1}$ cooling door
- NVIDIA: $\lfloor \frac{2{,}200}{1{,}838} \rfloor = \mathbf{1}$ server, $\lceil \frac{1}{7} \rceil = \mathbf{1}$ cooling door
- TPU: $\lfloor \frac{2{,}200}{861} \rfloor = \mathbf{2}$ servers, $\lceil \frac{2}{16} \rceil = \mathbf{1}$ cooling door

The rack power density constraint is the limiting factor, requiring 1 cooling door per rack for all server types.

10. **Consider the following two processors. P1 has a clock rate of 4GHz, average CPI of 0.9, and requires the execution of 5.0E9 instructions. P2 has a clock rate of 3GHz, an average CPI of 0.75, and requires the execution of 1.0E9 instructions.**

Given: Two processors with different specifications:

- P1: Clock rate = 4 GHz, CPI = 0.9, Instructions = 5.0×10^9
- P2: Clock rate = 3 GHz, CPI = 0.75, Instructions = 1.0×10^9

(a) Check if the processor with the largest clock rate has the highest performance:

Calculate execution times using $T = \frac{\text{Instructions} \times \text{CPI}}{\text{Clock Rate}}$:

For P1:

$$T_1 = \frac{5.0 \times 10^9 \times 0.9}{4.0 \times 10^9} = \frac{4.5 \times 10^9}{4.0 \times 10^9} = 1.125 \text{ seconds}$$

For P2:

$$T_2 = \frac{1.0 \times 10^9 \times 0.75}{3.0 \times 10^9} = \frac{7.5 \times 10^8}{3.0 \times 10^9} = 0.250 \text{ seconds}$$

Performance is inversely related to execution time:

- P1 performance: $1/1.125 = 0.889$ programs/second
- P2 performance: $1/0.250 = 4.000$ programs/second

Conclusion: P2 is 4.5 times faster than P1, despite having a lower clock rate. **FALLACY CONFIRMED:** Higher clock rate does not always mean higher performance.

(b) Determine instructions P2 can execute in the time P1 needs for 1.0×10^9 instructions:

P1 execution time for 1.0×10^9 instructions:

$$T_1 = \frac{1.0 \times 10^9 \times 0.9}{4.0 \times 10^9} = 0.225 \text{ seconds}$$

Instructions P2 can execute in 0.225 seconds:

$$\text{Instructions}_{P2} = \frac{T_1 \times \text{Clock Rate}_{P2}}{\text{CPI}_{P2}} = \frac{0.225 \times 3.0 \times 10^9}{0.75} = 9.0 \times 10^8$$

Conclusion: P2 executes 9.0×10^8 instructions in the same time P1 executes 1.0×10^9. **FALLACY CONFIRMED:** The processor executing more instructions (P1) actually takes longer.

(c) Check if the processor with the largest MIPS has the highest performance:

MIPS (Millions of Instructions Per Second) $= \frac{\text{Clock Rate}}{\text{CPI} \times 10^6}$:

$$MIPS_{P1} = \frac{4.0 \times 10^9}{0.9 \times 10^6} = 4444.4 \text{ MIPS}$$

$$MIPS_{P2} = \frac{3.0 \times 10^9}{0.75 \times 10^6} = 4000.0 \text{ MIPS}$$

P1 has higher MIPS (4444.4 vs 4000.0), but from part (a), P2 has higher actual performance.

Conclusion: FALLACY CONFIRMED: Higher MIPS does not indicate higher performance. The MIPS/Performance ratios are different (5000.0 for P1 vs 1000.0 for P2), confirming MIPS is unreliable for performance comparison.

11. **A program runs in 100 seconds on a single-core processor. A new processor improves the performance of a specific task within the program by a factor of 5, but this task only accounts for 30Amdahl's Law.**

Given: A program runs in 100 seconds. A new processor improves a specific task by $5\times$ speedup, but this task accounts for only 30% of total execution time.

Using Amdahl's Law to calculate overall speedup:

$$\text{Overall Speedup} = \frac{1}{(1-p) + \frac{p}{s}}$$

where $p = 0.30$ (fraction improved) and $s = 5$ (speedup of improved part).

$$\text{Overall Speedup} = \frac{1}{(1-0.30) + \frac{0.30}{5}} = \frac{1}{0.70 + 0.06} = \frac{1}{0.76} = 1.316$$

Time breakdown:

- Original improved task time: $100 \times 0.30 = 30$ seconds
- Original unimproved time: $100 \times 0.70 = 70$ seconds
- New improved task time: $305 = 6$ seconds
- New unimproved time: 70 seconds (unchanged)
- Total new execution time: $6 + 70 = 76$ seconds

Result: The overall speedup is **1.316**$\times$, reducing execution time from 100 to 76 seconds.

This demonstrates Amdahl's Law: even a $5\times$ improvement in 30% of the program yields only a $1.32\times$ overall speedup, showing the law of diminishing returns when only part of a system is improved.