## Q1

Assume a program requires the execution of 60 × 10^6 FP instructions, 120 × 10^6 INT instructions, 60 × 10^6 L/S instructions, and 16 × 10^6 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.
a. By how much must we improve the CPI of FP instructions if we want the program to run two times faster?
b. By how much must we improve the CPI of L/S instructions if we want the program to run two times faster?
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%?

Assume a program requires the execution of 50 x 10^6 FP instructions, 110 x 10^6 INT instructions, 80 x 10^6 Load/Store (L/S) instructions and 16 x 10^6 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.By how much must we improve the CPI of FP instructions if we want the program to run two times faster?

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

## Part A

We can not improve CPI of FP instructions when we run the program two times faster because it would be negative.
Explanation:

Processor clock rate = 2 GHz
Execution Time = ∑ (clockcycles/clockrate)
Clock cycles can be determined using following formula

Clockcycles = ($CPI_{FP}$ x No. FP instructions )+ ( $CPI_{INT}$ x No. INT instructions) + ( $CPI_{L/S}$ x No. L/S instructions ) + ( $CPI_{BRANCH}$ x No. branch instructions)

Clock cycles = ( 60 x $10^6$ x 1) + ( 120 x $10^6$ x 1) + ( 60 x $10^6$ x 4) + ( 16 x $10^6$ x 2)
Clock cycles = 452 x $10^6$

So,Initial Execution time for FP instructions is,
= (452*10^6)/2*10^9

Initial execution Time = 226 x $10^{-3}$

For 16 processors ,
clock cycle = 452 x $10^6$
Execution Time = 226 x $10^{-3}$
To run the program two times faster, half the number of clock cycles
Clockcycles= (CPI$_{FP}$ x No. FP instructions )+ ( CPI$_{INT}$ x No. INT instructions) + ( CPI$_{L/S}$ x No. L/S instructions ) + ( CPI$_{BRANCH}$ x No. branch instructions)
CPI$_{FPimproved}$ x 60 x $10^6$ = ( (452 x $10^6$)/2 ) - [ ( 120 x $10^6$ x 1) + ( 60 x $10^6$ x 4) + ( 16 x $10^6$ x 2)]
CPI$_{FPimproved}$ x 60 x $10^6$ = ( (226 x $10^6$)/2 ) - [ ( 120 x $10^6$ x 1) + ( 240 x $10^6$ ) + ( 32 x $10^6$)]
CPI$_{FPimproved}$ x 60 x $10^6$ = - 166 x $10^6$
CPI$_{FPimproved}$ = - 166 x $10^6$ / 60 x $10^6$
CPI$_{FPimproved}$ = - 2.76 < 0

## Part B:

Clock cycles = (CPI$_{FP}$ x No. FP instructions )+ ( CPI$_{INT}$ x No. INT instructions) + ( CPI$_{L/S}$ x No. L/S instructions ) + ( CPI$_{BRANCH}$ x No. branch instructions)
(452 x $10^6$)) = ( [ 1 x 60 x $10^6$+( 120 x $10^6$ x 1) + ( 60 x $10^6$ x CPI$_{L/S}$) + ( 16 x $10^6$ x 2)]
226 x =1 x 60 +( 120 x 1) + ( 60 x CPI$_{L/S}$ ) + ( 16 x 2)
-( 60 x CPI$_{L/S}$ )= -226 + 60+120 +( 32 )
-CPI$_{L/S}$ =(-226 + 60 +120 + 32 )/( 60 )
-CPI$_{L/S}$ = (-226+60+120+32)/60
CPI$_{L/S}$ = 0.2333
CPI reduced by 1-0.2333/4 = 0.9416 = 94.16%

## Part C:

New CPI1 = 0.5*1 = 0.5
New CPI2 = 0.5*1 = 0.5
New CPI3 = 0.75*4 = 3
New CPI4 = 0.75*2 = 1.5

New Execution Time $= \frac{\sum_{i=1}^{4} \text{Number of Instruction}^* \, CPI_i}{\text{Clock Rate}}$
New Execution Time= ( 60 x $10^6$ x 0.5) + ( 120 x $10^6$ x 0.5) + ( 60 x $10^6$ x 3) + ( 16 x $10^6$ x 1.5)/2*10^9
= 30*$10^6$ + 60*$10^6$ + 180* $10^6$ + 24*$10^6$)/2*$110^6$
=294*10^6/2*10^9
Increase in speed =1-147*10^-3/226 x $10^{-3}$
=1-0.6504424779 = 0.3495 = 34.95%

## Q2

*When a program is adapted to run on multiple processors in a multiprocessor system, the* ***execution time*** *on each processor is* ***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).*

Answer:
Per-processor execution time for different cases:

For 2 processors: (200/2) + 10 = 110 seconds
For 4 processors: (200/4) + 10 = 60 seconds
For 8 processors: (200/8) + 10 = 35 seconds
For 16 processors: (200/16) + 10 = 22.5 seconds
For 32 processors: (200/32) + 10 = 16.25 seconds
For 64 processors: (200/64) + 10 = 13.125 seconds

Speedup for each case relative to a single processor:

Speedup = Execution time on a single processor / Execution time on p processors
For 2 processors: 200 / 110 ≈ 1.818x
For 4 processors: 200 / 60 ≈ 3.333x
For 8 processors: 200 / 35 ≈ 5.714x
For 16 processors: 200 / 22.5 ≈ 8.889x
For 32 processors: 200 / 16.25 ≈ 12.308x
For 64 processors: 200 / 13.125 ≈ 15.238x

Ratio between actual speedup and ideal speedup:
Ideal Speedup = Execution time on a single processor / Execution time on p processors
For 2 processors: 200 / 100 ≈ 2x
For 4 processors: 200 / 50 ≈ 4x
For 8 processors: 200 / 25 ≈ 8x
For 16 processors: 200 / 12.5 ≈ 16x
For 32 processors: 200 / 6.25 ≈ 32x
For 64 processors: 200 / 3.125 ≈ 64x

Ratio between actual speedup and ideal speedup:
For 2 processors: 1.818 / 2 ≈ 0.909x
For 4 processors: 3.333 / 4 ≈ 0.833x
For 8 processors: 5.714 / 8 ≈ 0.714x
For 16 processors: 8.889 / 16 ≈ 0.556x
For 32 processors: 12.308 / 32≈ 0.384x
For 64 processors: 15.238 / 64 ≈ 0.238x

## Q3

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.
a. How much power savings would be achieved by turning off 60% of the servers?
b. How much power savings would be achieved by placing 60% of the servers in the “barely alive” state?
c. How much power savings would be achieved by reducing the voltage by 20% and frequency by 40%?
d. How much power savings would be achieved by placing 30% of the servers in the “barely alive” state and 30% off?

### Part A

a. We need to find How much power savings would be achieved **by turning off 60% of the servers**. We don’t know at what capacity other servers are operating at. There are two cases here

Case 1: If other servers are working at 60% capacity, they will be consuming 90 percent of power.

Hence power consumed in this case will be = 40% * 90% = 36%

Power consumed originally when most of the servers operated at only 60% capacity
= 100% * 90% = 90%

Power saved = 90% - 36% = 54%

Case 2: In this case I am assuming that all the servers are working at 100 percent capacity.

Hence power consumed in this case will be = 40% * 100% = 40%

Power consumed originally when all of the servers operated at only 100% capacity
= 100%

Power saved = 100% - 40% = 54%

## Part B

We need to find How much power savings would be achieved by placing 60% of the servers in the “barely alive” state. We don’t know at what capacity other servers are operating at. There are two cases here

Case 1: In this case I am assuming that the rest of the servers are working at 60 percent capacity consuming 90 percent power. If 60% of the servers were placed in the "barely alive" state, they would consume 20% of maximum power, compared to 90% of maximum power when running at 60% capacity.

Hence power consumed in this case will be = 60% * 20% + 40% * 90% = 48%

Power consumed originally when most of the servers operated at only 60% capacity
= 100% * 90% = 90%

Power saved = 90% - 48% = 42%

Case 2: In this case I am assuming that the rest of the servers are working at 100 percent capacity consuming 100 percent power.
If 60% of the servers were placed in the "barely alive" state, they would consume 20% of maximum power, compared to 100% of maximum power when running at 100% capacity.

Hence power consumed in this case will be = 60% * 20% + 40% * 100% = 52%

Power consumed originally when most of the servers operated at only 60% capacity
= 100%
Power saved = 100% - 52% = 48%

## Part C

If the voltage is reduced by 20% and the frequency is reduced by 40%, then the power consumption would be reduced by:

$P \propto V^2 f$

Where P = Power, V = voltage, f = frequency
$P_{new} / P_{old} = V_{new}^2 f_{new} / V_{old}^2 F_{old} = (0.8)^2 * 0.6 / 1^2 * 1 = 0.384$
Power saving = 1 - 0.384 = 0.616 = **61.6%**

## Part D

If 30% of the servers were placed in the "barely alive" state and 30% were turned off.

Case 1: Assuming the remaining servers would be running at 100% capacity, consuming 100% of maximum power.

Hence power consumed in this case will be  = 30% * 0.2 + 40*1= 46%
Power Savings = 100%-46% = 54%

Case 2: Assuming the remaining servers would be running at 60% capacity, consuming 90% of maximum power.

Hence power consumed in this case will be  = 30% * 0.2 + 40*0.9= 42%
Power Savings = 100%-42% = 58%

## Q4

*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.*
*a. If a company has 10,000 computers, each with a MTTF of 35 days, and it experiences catastrophic failure only if 1/3 of the computers fail, what is the MTTF for the system?*
*b. If it costs an extra $1000, per computer, to double the MTTF, would this be a good business decision? Show your work.*

### Part A

Mean time to failure is given by

MTTF=(Total working hours of running)/(Total number of units)

Let's say N is the average amount of time system runs until 1/3 computer fail

MTTF of each computer = 35
35 = (N * 10,000) / (10,000 * ⅓)
35 = N*3
N= 35 / 3 = 11.667

### Part B

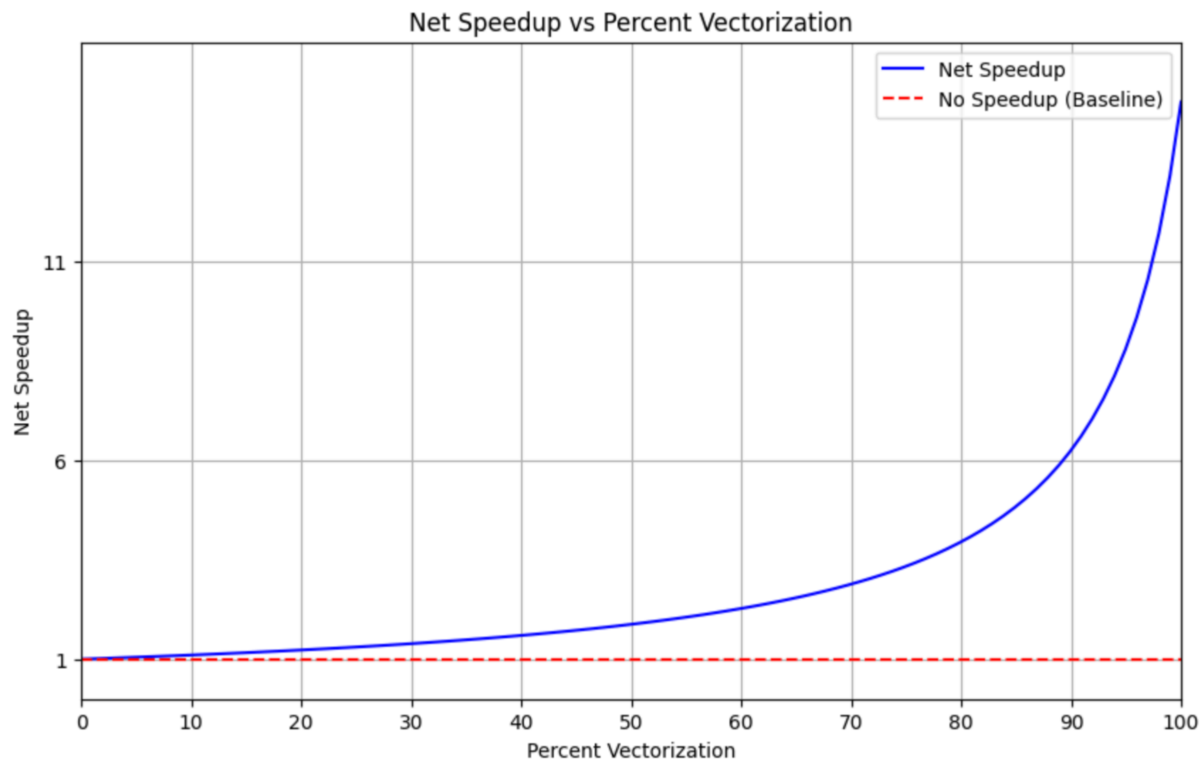
To reach a decision, the company needs to assess the cost of downtime—considering factors like lost revenue and reputational harm—and compare it to the $10 million investment. If the cumulative cost of downtime over the servers' lifetime is greater than $10 million, investing in the servers would be a wise choice. For instance, if downtime results in over $10 million in lost revenue, the investment would be well worth it.

## Q5

*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!*

a. *Draw a graph that plots the speedup as a percentage of the computation performed in vector mode. Label the y-axis "Net speedup" and label the x-axis "Percent vectorization." (Later)*
b. *What percentage of vectorization is needed to achieve a speedup of 3?*
c. *What percentage of the computation run time is spent in vector mode if a speedup of 2 is achieved?*
d. *What percentage of vectorization is needed to achieve one-half the maximum speedup attainable from using vector mode?*
e. *Suppose you have measured the percentage of vectorization of the program to be 70%. The hardware design group estimates it can speed up the vector hardware even more with significant additional investment. You wonder whether the compiler crew could increase the percentage of vectorization, instead. What percentage of vectorization would the compiler team need to achieve in order to equal an addition 2× speedup in the vector unit (beyond the initial 15×)?*

### Part A:



## Part B:

What percentage of vectorization is needed to achieve a speedup of 3?
The formula for Amdahl's law is:

S = 1 / (1 – P + (P / N))

Where:

S is the speedup of the system
P is the proportion of the system that can be improved
N is the number of processors in the system

3 = 1/(1-P + (P / 15))
3 = 1/(1-P + (P / 15))

3 = 1 / (1 - P + (P / 15))

3 * (1 - P + (P / 15)) = 1

3 * 1 - 3 * P + 3 * (P / 15) = 1

3 - 3P + (P / 5) = 1

3 - 3P + (P / 5) - 3 = 1 - 3

-3P + (P / 5) = -2

5 * (-3P + (P / 5)) = 5 * (-2)

-15P + P = -10

-14P = -10

P = -10 / -14

P = 5 / 7

P = 0.714
So, approximately 71.4% of the computation needs to be vectorized to achieve a speedup of 3.

## Part C

$2 = 1 / (1 - P + (P / 15))$
$2 * (1 - P + (P / 15)) = 1$
$2 * 1 - 2 * P + 2 * (P / 15) = 1$
$2 - 2P + (2P / 15) = 1$
$2 - 2P + (2P / 15) - 2 = 1 - 2$
$-2P + (2P / 15) = -1$
$15 * (-2P + (2P / 15)) = 15 * (-1)$
$-30P + 2P = -15$
$-28P = -15$
$P = -15 / -28$
$P = 15 / 28$
$P = 0.536$
So, approximately 53.6% of the computation needs to be vectorized to achieve a speedup of 2.

## Part D: What percentage of vectorization is needed to achieve one-half the maximum speedup?

The maximum speedup is 15 (since vector mode is 15 times faster).
We need to achieve half of this maximum speedup:
Target speedup = 15 / 2 = 7.5
Use the equation:
$7.5 = 1 / (1 - P + (P / 15))$
$7.5 * ((15 - 15P + P) / 15) = 1$
$((15 - 15P + P) / 2) = 1$
$((15 - 14P) = 2$
$P = 13 / 14$
$P \approx 0.9286$

So, approximately 92.86% of the computation needs to be vectorized to achieve a one-half the maximum speedup.

## Part E:

To achieve 30 ( 2x on top of initial 15x)
Target speedup = 30
Using the equation:
Speedup(S) = 1 / (1 - 0.7 + (0.7 / 15))

Solving this we get Speed up as 2.884

Now, suppose the vector hardware becomes 17x faster. Then the speedup is:
Speedup(S) = 1 / (1 - 0.7 + (0.7 / 17)) = 1/ 0.3411 = 2.931

Now, we want to find what percentage of vectorization P′ would give the same speedup (around 2.931) using the original 15× vector hardware

2.931 = 1 / ( (1 - percentage of vectorization) + (percentage of vectorization / 15))
Solving this we get,
percentage of vectorization = 0.7058

Therefore, the compiler team would need to increase the percentage of vectorization to approximately **70.58%** to achieve the same speedup as doubling the vector hardware speed

## Q6

*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.*
*a. Find the total execution time for this program on 1, 2, 4, and 8 processors, and show the relative speedup of the 2, 4, and 8 processors result relative to the single processor result.*
*b. To what should the CPI of load/store instructions be reduced in order for a single processor to match the performance of four processors using the original CPI values?*

Answer

- **CPIs:**
  - Arithmetic instructions: 2
  - Load/store instructions: 10
  - Branch instructions: 5
- **Instruction counts:**
  - Arithmetic instructions: 2.56×10^9
  - Load/store instructions: 1.28×10^9
  - Branch instructions: 128×10^6 =$0.128*10^9$
- **Clock frequency:** 2 GHz

### Part A

Clockcycles = ( $CPI_{Arith}$ x No. Arithmetic instructions) + ( $CPI_{L/S}$ x No. L/S instructions ) + ( $CPI_{BRANCH}$ x No. branch instructions)

**Calculations for 1 core:**

- Arithmetic instructions: 2.56×10^9
- Load/store instructions: 1.28×10^9
- Branch instructions: 128×10^6 =$0.128*10^9$

Number of Clockcycles = $2.56*10^9 * 2 + 1.28 * 10^9 * 10 + 0.128*10^9 * 5 = 18.56*10^9$

Execution time = $18.56*10^9$ cycles / $2.0*10^9$ cycles/second

= 9.28 seconds

**Calculations for 2 core:**

- Arithmetic instructions: $2.56*10^9/0.7*2=1.829*10^9$
- Load/store instructions: $1.28*10^9/0.7*2=0.914*10^9$
- Branch instructions: 128×10^6 $=0.128*10^9$

Number of Clockcycles $= 1.829*10^9 * 2 + 0.91429*10^9 *10 + 0.128*10^9 * 5$
$= 13.4409*10^9$

Execution time = $13.4409*10^9$ cycles / $2.0*10^9$ cycles/second

= 6.72 seconds

Speedup relative to a single processor:

Speedup= 9.28 seconds / 6.72 seconds =1.38

**Calculations for 4 core:**

- Arithmetic instructions: $2.56*10^9/0.7*4=0.914*10^9$
- Load/store instructions: $1.28*10^9/0.7*4=0.457*10^9$
- Branch instructions: 128×10^6 $=0.128*10^9$

Number of Clockcycles $= 0.914*10^9 * 2 + 0.457*10^9 *10 + 0.128*10^9 * 5$
$= 7.04*10^9$

Execution time = $7.04*10^9$ cycles / $2.0*10^9$ cycles/second

= 3.52 seconds

**Speedup relative to a single processor:**

Speedup=9.28 seconds / 3.52 seconds =2.64

**Calculations for 8 core:**

- Arithmetic instructions: $2.56*10^9/0.7*8=0.46*10^9$
- Load/store instructions: $1.28*10^9/0.7*8=0.23*10^9$
- Branch instructions: 128×10^6 $=0.128*10^9$

Number of Clockcycles $= 0.46*10^9 * 2 + 0.23*10^9 *10 + 0.128*10^9 * 5$
$=3.86*10^9$

Execution time = $3.86*10^9$ cycles / $2.0*10^9$ cycles/second

= 1.93 seconds

**Speedup relative to a single processor:**

Speedup=9.28 seconds /1.93 seconds =4.81

## Part B

We want single processor’s execution time to match the performance of four processors

Execution Time= Total clock cycles / clock frequency

Execution Time =( $CPI_{Arith}$ x No. Arithmetic instructions) + ( $CPI_{L/S}$ x No. L/S instructions ) + ( $CPI_{BRANCH}$ x No. branch instructions) / clock frequency

We know the target execution time is 3.52 seconds. Hence, Above equation can be written as:

∴ $3.52 = 2.56*10^9 * 2 + 1.28 * 10^9 * 10 + 0.128*10^9 * 5 / 2*10^9$
∴ $3.52 = 2.56*10^9 * 2 + CPI_{L/S} * 1.28 * 10^9 + 0.128*10^9 * 5 / 2*10^9$
∴ $3.52*2*10^9 = 2.56*10^9 * 2 + CPI_{L/S} * 1.28 * 10^9 + 0.128*10^9 * 5$
∴ $3.52*2*10^9 - 2.56*10^9 * 2 - 0.128*10^9 * 5 = CPI_{L/S} * 1.28 * 10^9$
∴ $1.28*10^9 *= CPI_{L/S} * 1.28 * 10^9$
∴ $CPI_{L/S} = 1$

Hence, the CPI of load/store instructions must be reduced to **1** for the single processor to match the performance of 4 processors.

## Q7

Suppose we developed a simpler processor that has 75% of the capacitive load of a more complex processor. Further, assume that it can adjust voltage so that it can reduce voltage by 20% compared to the complex processor, but this results in a 25% increase in frequency.
a. How much energy do we save through this change?
b. What is the impact on the dynamic power?

### Part A:

$\text{Power}_{new}$ / $\text{Power}_{old}$ = (Capacitive load * 0.75* $\text{Voltage}^2$ * $0.80^2$* Frequency switched* 1.25)/ Capacitive load * $\text{Voltage}^2$ * Frequency switched

∴$\text{Power}_{new}$ / $\text{Power}_{old}$ new = (0.75 * $0.80^2$ * 1.25) = 0.6

Hence, Energy saved = 1 - 0.6 = 0.4
∴ 40% energy is saved.

0.4

### Part B:

As calculated above, the dynamic power of the simpler processor is **60%** of the dynamic power of the complex processor, meaning the dynamic power is reduced by **40%**.

## Q8

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.

a. According to the trend in device scaling historically observed by Moore's Law, the number of transistors on a chip in 2025 should be how many times the number in 2015? [assume # Transistors double every 2 years]

b. The increase in performance once mirrored this trend. Had performance continued to climb at the same rate as in the 1990s, approximately what performance would chips have over the VAX11/780 in 2025? [assume performance increases 52% every year]

c. What has limited the rate of growth of the clock rate, and what are architects doing with the extra transistors now to increase performance?

A. The number of transistor double every
   $2^5 = 32$
B. $1.52^{35}$=23,14,864

C. Growth of clock rates has been limited primarily by several factors:

1. Power Consumption: Higher clock speeds lead to increased power usage and heat generation. As power density rises, it becomes challenging to manage heat dissipation, which can damage components and reduce reliability.
2. Diminishing Returns: As clock speeds increase, the performance gains per additional MHz or GHz decrease due to factors like memory latency and the limits of instruction-level parallelism.
3. Physical Limitations: As technology scales down, the physical limitations of materials and the effects of quantum mechanics come into play, making it harder to push clock rates higher.

**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.

**a.** If Google's data center spends 70% of its time on workload A and 30% of its time on workload B when running GPUs, what is the speedup of the TPU system over the GPU system?

**b.** Google's data center spends 70% of its time on workload A and 30% of its time on workload B when running GPUs, what percentage of Max IPS does it achieve for each of the three systems?

**c.** Building on (b), assuming that the power scales linearly from idle to busy power as IPS grows from 0% to 100%, what is the performance per watt of the TPU system over the GPU system?

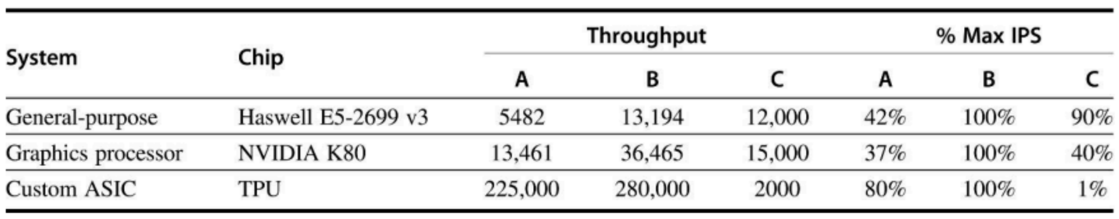
**d.** If another data center spends 40% of its time on workload A, 10% of its time on workload B, and 50% of its time on workload C, what are the speedups of the GPU and TPU systems over the general-purpose system?

**e.** A cooling door for a rack cost $4000 and dissipates 14 kW (into the room; additional cost is required to get it out of the room). How many Haswell-, NVIDIA-, or Tensor-based servers can you cool with one cooling door, assuming TDP in Figures 1.27 and 1.28?

**f.** Typical server farms can dissipate a maximum of 200 W per square foot. Given that a server rack requires 11 square feet (including front and back clearance), how many servers from part (e) can be placed on a single rack, and how many cooling doors are required?

| System | Chip | TDP | Idle power | Busy power |
|---|---|---|---|---|
| General-purpose | Haswell E5-2699 v3 | 504 W | 159 W | 455 W |
| Graphics processor | NVIDIA K80 | 1838 W | 357 W | 991 W |
| Custom ASIC | TPU | 861 W | 290 W | 384 W |

**Figure 1.27** Hardware characteristics for general-purpose processor, graphical processing unit-based or custom ASIC-based system, including measured power



| System | Chip | Throughput | | | % Max IPS | | |
|---|---|---|---|---|---|---|---|
| | | A | B | C | A | B | C |
| General-purpose | Haswell E5-2699 v3 | 5482 | 13,194 | 12,000 | 42% | 100% | 90% |
| Graphics processor | NVIDIA K80 | 13,461 | 36,465 | 15,000 | 37% | 100% | 40% |
| Custom ASIC | TPU | 225,000 | 280,000 | 2000 | 80% | 100% | 1% |

**Figure 1.28** Performance characteristics for general-purpose processor, graphical processing unit-based or custom ASIC-based system on two neural-net workloads

## Part A

Google's data center spends:
70% of its time on workload A = 0.7 x workload (A)
30% of its time on workload B = 0.3 x workload (B)
Hence, TPU performance = 0.7 x TPU performance of A + 0.3 TPU performance of B
= 0.7 x 225000 + 0.3 x 280000 = 157500 + 84000 = **241500**
GPU performance = 0.7 x GPU (A) + 0.3 GPU (B)
= 0.7 x 13461 + 0.3 x 36465 = 9422.7 + 10940 = **20,362.7**
Speedup = TPU performance / GPU performance = 11.86

## Part B

We need to find percentage of Max IPS it achieve for each of the three systems
Google's data centre spends 70% of time on workload A and 30% of its time on workload B
Hence, Haswell Max IPS = 0.7 x Haswell Max IPS of A + 0.3 x Haswell Max IPS of B
= 0.7 x 0.42 + 0.3 x 1 = 0.294 + 0.3 = 0.594 = 59.4%

GPU Max IPS = 0.7 x GPU Max IPS of A + 0.3 x GPU Max IPS of B
= 0.7 x 0.37 + 0.3 x 1 = 0.260 + 0.3 = 0.560 = 56%

TPU Max IPS = 0.7 x TPU Max IPS of A + 0.3 x TPU Max IPS of B
=0.7 x 0.8 + 0.3 x 1 = 0.560 + 0.3 = 0.860 = 86%

## Part C

From part A TPU performance = 241500
From part A GPU performance = 20362.2
assuming that the power scales linearly from idle to busy power as IPS grows from 0% to 100%
Power = Ideal Power + [ (busy - ideal) * (effective IPS / 100) ]
GPU power = 357 + [ (991−357) * (55.9 / 100) ] = 357 + 634 * 0.559 = 711 Watt
TPU power = 290 + [(384−290) I (86/100) = 290 + 94 * 0.86 = 370.84 Watt
GPU performance per watt = 20362.2/ 711.03 = 28.638 IPS per Watt
TPU performance per watt = 241500/ 370.84 = 651.15 IPS per Watt

## Part D:

40% of time spent on workload A = 0.4 x workload (A)
10% of time spent on workload B = 0.1 x workload (B)
50% of time spent on workload C = 0.5 x workload (C)

Haswell performance = 0.4 x Haswell of A + 0.1 Haswell of B + 0.5 Haswell of C
=0.4 x 5482 + 0.1 x 13194 + 0.5 x 12000 = 9512.2

GPU performance = 0.4 x GPU performance of A + 0.1 GPU performance of B + 0.5 GPU performance of C
= 0.4 x 13461 + 0.1 x 36465 + 0.5 x 15000 = 16530.9

TPU performance = 0.4 x TPU(A) + 0.1 TPU(B) + 0.5 TPU(C)
TPU performance = 0.4 x 225000 + 0.1 x 280000 + 0.5 x 2000 = 119000

Speedup of TPU over general purpose = 119000 / 9512.2 = 12.51
Speedup of GPU over general purpose = 16530.9 / 9512.2 = 1.738

## Part E :

Dissipated power = 14kW
Rack Cost = $4000
Haswell = 504 W
GPU = 1838 W
TPU = 861 W

Haswell servers cooled = 14kw / 504 = 14000 / 504 = 27.778 = 28 servers per door
GPU servers cooled = 14kw / 1838 = 14000 / 1838 = 7.617 = 8 servers per door
TPU servers cooled = 14kw / 861 = 14000 / 861 = 16.26 = 17 servers per door

## Part F:

Given, server farms can dissipate a maximum of 200 W per square foot
The server requires 11 foot
Servers are placed on a single rack
Total dissipation per rack = 200 x 11 = 2200 Watt
Haswell servers = 2200 / 504 = 4.365 = 4 servers per rack
GPU servers = 2200 / 1838 = 1.197 = 1 server per rack
TPU servers = 2200 / 861 = 2.555 = 2 servers per rack
Haswell servers cooled = 14kw / ( 4x 504 ) = 14000 / 2016 = 6.944 = 7 doors
GPU servers cooled = 14kw / ( 1x 1838 ) = 14000 / 1838 = 7.617 = 8 doors
TPU servers cooled = 14kw / ( 2 x 861 ) = 14000 / 1722 = 8.13 = 9 doors

## Q10

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.

a. One usual fallacy is to consider the computer with the largest clock rate as having the highest performance. Check if this is true for P1 and P2.

b. Another fallacy is to consider that the processor executing the largest number of instructions will need a larger CPU time. Considering that processor P1 is executing a sequence of 1.0E9 instructions and that the CPI of processors P1 and P2 do not change, determine the number of instructions that P2 can execute in the same time that P1 needs to execute 1.0E9 instructions.

c. A common fallacy is to use MIPS (millions of instructions per second) to compare the performance of two different processors, and consider that the processor with the largest MIPS has the largest performance. Check if this is true for P1 and P2.

### Part A

For P1:

- Clock Rate: 4 GHz = $4*10^9$ cycles/second
- CPI: 0.9
- Instructions: $5.0*10^9$

Execution time of P1 = $0.9 * 5 * 10^9 / 4 * 10^9$= **1.125 seconds**

For P2:

- Clock Rate: 3 GHz = $3*10^9$ cycles/second
- CPI: 0.75
- Instructions: $1.0*10^9$

Execution time of P1 = $0.75 * 1 * 10^9 / 3 * 10^9$ = **0.25 seconds**

Hence, despite P1 having a higher clock rate, P2 has a **shorter execution time** due to executing fewer instructions with a lower CPI.

## Part B

For P1:

- Clock Rate: 4 GHz = $4*10^9$ cycles/second
- CPI: 0.9
- Instructions: $1.0*10^9$

Execution time of P1 = $0.9 * 1 * 10^9 / 4 * 10^9$= **0.225 seconds**

For P2:

- Clock Rate: 3 GHz = $3*10^9$ cycles/second
- CPI: 0.75
- Instructions: N
- Execution time: 0.225 seconds

∴ $0.225 = 0.75 * N / 3 * 10^9$

∴ $N = 0.225 * 3 * 10^9 / 0.75 = 0.9 * 10^9$

In the time it takes P1 to execute 1.0 billion instructions, P2 can execute 900 million instructions.

## Part C

MIPS = Clock rate * $10^{-6}$ /CPI

For P1:

- Clock Rate: 4 GHz = $4*10^9$ cycles/second
- CPI: 0.9

MIPS for P1 = Clock rate * $10^{-6}$ / CPI = $4*10^9 * 10^{-6} / 0.9 = 4.44\times10^3$

For P2:

- Clock Rate: 3 GHz = $3*10^9$ cycles/second
- CPI: 0.75

∴ MIPS for P2 = Clock rate * $10^{-6}$ / CPI = $3*10^9 * 10^{-6} / 0.75 = 4\times10^3$

P1 has a higher MIPS value (4444 MIPS vs. 4000 MIPS). However, as we calculated earlier, P2 completes the task faster than P1 due to executing fewer instructions with a lower CPI. This proves that MIPS alone is not a reliable performance metric, and a higher MIPS value does not necessarily mean higher performance.

## Q11

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 30% of the total execution time. Calculate the speedup achieved on the new processor using Amdahl's Law.

According to Amdahl's Law speedup is:

$S = 1 / (1 – P + (P / N))$
$S = 1 / (1 – 0.3 + (0.3 / 5))$
$S = 1 / (0.7 + 0.06)$
$S = 1 / (0.76) = 1.316$
Hence, the speedup achieved on the new processor is approximately **1.316**.