Karan Vora (N12229954)
Computer Systems Architecture Assignment 1

Solution 1):

CPU P1: Clock = 3 GHz, CPI = 1.5
CPU P2: Clock = 2.5 GHz, CPI = 1.0
CPU P3: Clock = 4.0 GHz, CPI 2.2

A):
Here, we have to calculate instructions executed per second, Thus
CPU Time = 1 Second
Instruction count for CPU = CPU Time/(Clock cycle time x CPI) ---- (1.1)

Clock cycle time for P1 = 1/(3E9) second
Instruction count for P1 = 1/(1/(3E9) x 1.5)
Instruction count for P1 = 2.0E9 Instructions per second

Clock cycle time for P2 = 1/(2.5E9) second
Instruction count for P2 = 1/(1/(2.5E9) x 1.0)
Instruction count for P2 = 2.5E9 Instructions per second

Clock cycle time for P3 = 1/(4.0E9) second
Instruction count for P3 = 1/(1/(4.0E9) x 2.2)
Instruction count for P3 = 1.8E9 Instructions per second

Thus from above calculated results, we conclude that CPU P1 has the highest performance in Instructions per second

B):
Program execution time = 10 seconds
Number of Cycles = Time x Clock Rate
Number of Instructions = Time x Instructions per clock (IPC)

Clock for P1 = 3.0E9 Hz
IPS (Instructions per second) for P1 = 5E8
Number of cycles for P1 = 3.0E9 x 10 = 3.0E10 Cycles
Number of instructions for P1 = 2.0E9 x 10 = 2.0E10 Instructions

Clock for P2 = 2.5E9 Hz
IPS for P2 = 4.0E9
Number of cycles for P2 = 2.5E9 x 10 = 2.5E10 Cycles
Number of instructions for P2 = 2.5E9 x 10 = 2.5E10 Instructions

Clock for P3 = 4.0E9 Hz
IPS for P3 = 1.136E8
Number of cycles for P3 = 4.0E9 x 10 = 4.0E10 Cycles
Number of instructions for P3 = 1.8E8 x 10 = 1.8E10 Instructions

C):
Reduction of execution time = 30%
Increase in CPI = 20%

Lets assume the execution time be ***x,*** therefore new execution time will be ***x* * 0.7**
also assume CPI be ***y,*** therefore the new CPI will be ***y* * 1.2**
Now the formula for CPU time = (Instruction count * CPI) / Clock rate

Putting the two assumed values in the formula we get

x = (Instruction count * ***y***) / Clock rate ---- (1.2)

***x* * 0.7** = (Instruction count * ***y* * 1.2**) / Clock rate$_2$ ---- (1.3)

Now Dividing (1.2) with (1.1) and restructuring the values we get,

Clock rate$_2$ = (1.2/0.7) * Clock rate ---- (1.4)

Now substituting the value of clock rate for each CPU to calculate new clock rate

Clock rate$_2$ for P1 = (1.2/0.7) * 3E9 = 5.1428E9 Hz = 5.1428 GHz
Clock rate$_2$ for P2 = (1.2/0.7) * 2.5E9 = 4.2857E9 Hz = 4.2857 GHz
Clock rate$_2$ for P3 = (1.2/0.7) * 4.0E9 = 6.8571E9 Hz = 6.8571 GHz

==

Problem 2):

Arithmetic CPI = 1
Load/Store CPI = 12
Branch Instruction CPI = 5

Performance of a single processor:
Arithmetic Instruction = 2.56E9
Load/Store Instruction = 1.28E9
Branching Instruction = 2.56E8
Clock speed = 2E9 Hz

Execution time = $\Sigma^{n}_{i=1}(IC_i * CPI_i)$ * Clock cycle time ---- (2.1)

ET for 1 CPU Processor = (2.56 + 1.28*12 + 0.256*5)E9 * (1/2E9)
ET for 1 CPU Processor = 9.6 Second

Performance drop in multi-core scenario for Arithmetic and Load/Store = 0.7 * p (p is number of core)
overall performance = (2.56/(0.7*p) + 1.28*12/(0.7*p) + 0.256*5)E9 * (1/2E9) ---- (2.2)

ET for 2 core = (2.56/(0.7*2) + 15.36/(0.7*2) + 1.28)E9 * (1/2E9) = 7.04 Second
ET for 4 core = (2.56/(0.7*4) + 15.36/(0.7*4) + 1.28)E9 * (1/2E9) = 3.84 Second
ET for 8 core = (2.56/(0.7*8) + 15.36/(0.7*8) + 1.28)E9 * (1/2E9) = 2.24 Second

Relative Performance for 2 cores from 1 = ET for 1 core/ET for 2 core = 9.6/7.04 = 1.36364
Relative Performance for 4 core from 1 core = ET for 1 core/ET for 4 core = 9.6/3.84 = 2.5
Relative Performance for 8 core from 1 core = ET for 1 core/ET for 8 core = 9.6/2.24 = 4.28571

B):
Old Arithmetic Instruction count = 2.56E9
New Arithmetic Instruction count = 2.56E9 * 2 = 5.12E9

Now, Substituting the new value of Arithmetic Instruction Count in formula 2.1, we get
ET for 1 Core = (5.12 + 15.36 + 1.28)E9 * (1/2E9) = 10.88 Second

Substituting the same value in formula 2.2, we get
ET for 2 core = (5.12/(0.7*2) + 15.36/(0.7*2) + 1.28)E9 * (1/2E9) = 7.95429 Second
ET for 4 core = (5.12/(0.7*4) + 15.36/(0.7*4) + 1.28)E9 * (1/2E9) = 4.29714 Second
ET for 8 core = (5.12/(0.7*8) + 15.36/(0.7*8) + 1.28)E9 * (1/2E9) = 2.46857 Second

C):
Time taken to execute the program on 4 core processor = 3.84 Second = Target time for 1 core processor using Load/Store CPI
Now, substituting the target time in formula 2.1 to calculate the value of CPI of Load/Store
Let the new CPI for Load/Store be ***x*.** Therefore,

(2.56 + 1.28****x*** + 1.28)E9 * (1/2E9) = 3.84

x = 3
New Load/Store CPI for 1 Core processor to match 4 core processor is *x* = 3,
Original = 12, New CPI =3. So new CPI = 25% of Original, 75% Reduction vs Original

==

Problem 3):
Clock cycle time for P1 = 0.33E-9 Second, Clock Rate = 1/0.33E-9 = 3.03E9 Hz
Clock cycle time for P2 = 0.40E-9 Second, Clock Rate = 1/0.40E-9 = 2.5E9 Hz
Clock cycle time for P3 = 0.3E-9 Second, Clock Rate = 1/0.3E-9 = 3.33E9 Hz

CPI for P1 = 1.5
CPI for P2 = = 1.0
CPI for P3 = 2.8

A):
Highest Clock Rate = P3 = 3.33E9 Hz = 3.33 GHz

B):
Clock Rate alone doesn’t tell the whole story about the final performance, using the given metric for the processor P1, P2 and P3 we can calculate the Instruction per clock for the each processor. Given that all three processors has same ISA, this calculation becomes easy using the formula

Instruction count for CPU = CPU Time/(Clock cycle time x CPI) --- (3.1)

Lets assume CPU Time = 1 Second
Thus substituting the values for CPU Time in formula 3.1 we can calculate Instruction Count in given time for P1, P2 and P3

Instruction Count for P1 = 1/(0.33E-9 * 1.5) = 2.02E9
Instruction Count for P2 = (1/0.40E-9 * 1) = 2.5E9
Instruction Count for P3 = (1/0.3E-9 * 2.8) = 1.18928E9

From the above calculated figures, P2 has the highest Instruction Count at 2.5E9 instructions executed in 1 second. This is different from the solution of (A) in which P3 had the highest clock rate. This is for the fact that clock rate isn't the only metric to measure the performance of a processor, A processor with lower clocks but higher Instruction per unit time can execute more Instruction in the same time or Takes less time to execute the same number of Instructions.

C):
When it comes to measuring the performance level of a computational device, the general term for the process is called a Benchmark. In a Benchmark of a computational hardware, you run several programs that are computationally compatible to the hardware. The key to execute a benchmark is to have the same piece of program on different hardware without any external influence or change in the source code between the benchmark runs to ensure unbiased results. Now a hardware can be benchmark in multiple ways and one of the way is just looking at the clock speed and comparing the relative performance between the different hardware. While this method can be true for hardware with similar specifications, It is far for true metric of measurement of performance. There are multiple aspects that dictates the overall performance of a computational hardware thus it is required to measure the other aspects and one of the most common aspect is IPS or Instructions per Second. It measures how many instructions a processor executed per clock. Using this metric, we can calculate the total Instructions executed per unit CPU Time, showing us much wider picture of performance. A processor with lower clocks but higher IPS can execute more Instruction in the same time or Takes less time to execute the same number of Instructions. Thus from (A) and (B) we conclude that just clock speed is not the true measure of performance but more meticulous testing with IPS is necessary for benchmark.

==

Problem 4):

A):
It is given that at worst case scenario, the code can run twice as fast, Meaning the at worst it will take half the time to execute a piece of code. Considering the worst case scenario everytime, The system will shut down faster after executing the piece of code thus saving the energy. Here considering the 2x speed up, the energy saving at worst is 50%.

B):
The energy in the CMOS chips is function of voltage. Thus the energy in a CMOS system is directly proportional to the product of capacitive load and square of voltage.

$\text{Energy}_{\text{Dynamic}}$ α capacitive load * voltage^2 ---- (4.1)

Now it is given that voltage for new system is half of previous thus substituting this value in formula (4.1), we get

$\text{Energy}_{\text{New}}/\text{Energy}_{\text{Old}} = (\text{Voltage} * 0.5)^2/\text{Voltage}^2$

$\text{Energy}_{\text{New}} = 0.25 * \text{Energy}_{\text{Old}}$

From the above mentioned relationship, we can conclude that reducing the voltage in half results in 25% of power consumption of old system or in other words, A 75% reduction in total power consumption. Reduction in frequency doesn't affect the total system energy consumption but does affect the power.

==

Problem 5):

Clock rate of P1 = 2 GHz = 2E9 Hz
P1 CPI Class A = 1
P1 CPI Class B = 2
P1 CPI Class C = 2
P1 CPI Class D = 1

Clock rate P2 = 4 GHz = 4E9 Hz
P2 CPI Class A = 2
P2 CPI Class B = 3
P2 CPI Class C = 4
P2 CPI Class D = 4

A):
Dynamic Instruction Count = 1.0E6
Instruction count of Class A = 0.1 * 1.0E6
Instruction count of Class B = 0.2 * 1.0E6
Instruction count of Class C = 0.5 * 1.0E6
Instruction count of Class D = 0.2 * 1.0E6

Now recalling the formula 2.1, and substituting the values, we can calculate the execution time for each processor

ET for P1 = ((0.1 * 1) + (0.2 * 2) + (0.5 * 2) + (0.2 * 1)) * 1.0E6 * (1/2E9)
ET for P1 = 0.85E-3 = 8.5E-4 Second

ET for P2 = ((0.1 * 2) + (0.2 * 3) + (0.5 * 4) + (0.2 * 4)) * 1.0E6 * (1/4E9)
ET for P2 = 0.9E-3 = 9.0E-4 Second

From the above mentioned calculations, P1 is clearly faster than P2

B):
The formula for global CPI is as follows

Global CPI = $\Sigma^{n}_{i=1}$(Percentage Distribution$_i$ * CPI$_i$) --- (5.1)

Substituting values for CPI for P1 and P2 in formula 5.1
Global CPI P1 = ((0.1 * 1) + (0.2 * 2) + (0.5 * 2) + (0.2 * 1)) = 1.7
Global CPI P2 = ((0.1 * 2) + (0.2 * 3) + (0.5 * 4) + (0.2 * 4)) = 3.6

C):
Clock cycle required = Global CPI * Instruction count ---- (5.2)
Substituting values for P1 and P2 in formula 5.2, we can get the values
Clock cycles required for P1 = 1.7 * 1.0E6 = 1.7E6 Cycles
Clock cycles required for P2 = 3.6 * 1.0E6 = 3.6E6 Cycles

D):
Here time is to be calculated for per second therefore CPU time = 1 Second.
From formula 1.1, we can calculate the Instructions per second count

P1 = 1/((1/2E9) * 1.7) = 1.17647E9 Instructions per second
P2 = 1/((1/4E9) * 3.6) = 1.11111E9 Instructions per second

Thus from the values deduced as above, we can conclude that P1 has the highest throughput.

E):
From values deduced as per solutions of (A), (C) and (D), It is clear that P1 is more energy efficient than P2. It is already given that both P1 and P2 has the same Instruction Set Architecture meaning the other unknown variables related to the metrics of performance can be assumed as same for both the processors, Thus the overall deduction of efficiency boils down to the values calculated in this problem.

From (A) we deduce that P1 takes less time vs P2 for executing same number of instructions. Thus in other words, Programs running on P1 will execute faster thus saving much time and increasing the time efficiency.

From (C) we deduce that P1 takes fewer instructions vs P2 for executing same program. Assuming executing each instructions takes the same amount of power for both the processor because both processors has the same Instruction Set Architecture, P1 will consume less power vs P2 for executing the same set of instructions.

From (D) we deduce that P1 has higher overall throughput vs P2 thus more number of tasks can be executed on P1 vs P2 thus increasing the overall productivity performance of the user, therefore increasing the overall performance efficiency of the user.

===

Problem 6):

CISC = Complex Instruction Set Computer
RISC = Reduced Instruction Set Computer

A):
CISC is an term for a family of ISA that was coined after the term RISC. In this family of ISA, A single instruction performs multiple low-level operations simultaneously. For example a single CISC instruction can perform operations like Load, Compute and Store in just one instruction. Unlike the Complex in the name, It doesn’t indicate the complexity of the ISA family or the Instructions itself or the number of Instructions. CISC have variable length instructions where larger Instructions can execute more tasks per instruction. Some of the example of CISC Architecture are Motorola 68K, DEC VAX, Intel x86 etc.

RISC is an acronym for Reduced Instruction Set Computer. It is a family of Architectures designed with more general and simple instruction sets in mind. RISC instructions are smaller, much simple and generalized vs complex specialized instructions of CISC architecture. This allows RISC architecture to execute each instruction with fewer clock cycles thus increasing the overall per instruction efficiency of execution more over all the Instructions in a RISC architecture has a uniform length thus reducing the complexity of decoding pipeline of a processor. Some of the examples of RISC Architecture are ARM, RISC V, IBM 801 etc.

The major advantages of CISC architecture is the reduced overall size of a program. Because CISC requires fewer instructions to finish the same set of task, It is much more Memory and Disc efficient. CISC can perform several task simultaneously in a single instruction thus the access to the main memory is much less reducing the overall latency due to the wait for the main memory to respond and thus inceasing the overall execution speed of the program. It also provides a larger more specialized set of instruction thus a programmer can program a much complex code with relatively ease thus increasing overall productivity.

The major advantages of RISC architecture is the reduced complexity of the instructions thus an engineer can design a far more efficient processor for much lower cost and much faster. Because the simplified instructions and the uniform length Instructions, the RISC processor can be much more efficient than CISC in executing the same set of task. This also allows for much faster debugging of issues because of the reduced complexity of the decoder pipeline and this flexibility allows for much higher scalability of the system.

B):
RISC V is an completely open-source ISA designed by University of California, Berkeley. This ISA was initially designed as a project is now adopted as industry wide standard. The architecture took inspiration from the elements of software design. In the domain of software design, the tools to design and deliver any software is available for free and is completely open to the end user. Following the same principle, the tools to design any hardware using RISC V is freely available for anyone to use.

ARM was designed by Acorn RISC Machines (Now Advanced RISC Machines) as a RISC architecture and x86 was designed by Intel as a CISC architecture. Both of these architecture are proprietary to the companies themselves and only available to other users as either the end products that the companies manufacture or as a license based model if you want to use the architecture. The tools to design and implement these architecture are not available as freely to the user.

The biggest advantage of RISC V is the fact that its open-source. This allows anyone with the necessary skills, design a processor on its own without any licensing fees, this open-source nature also allows anyone with skills to contribute to the continuous research and development of the architecture. For companies and industries designing the product using RISC V, it means they don't have to worry about the heavy licensing fees and added complexity of design due to limited access of the architecture from ARM or Intel. Thus the products designed on RISC V can be delivered faster and at much lower cost.

C):
The biggest issue with RISC V currently is Adaptation, Compatibility and Optimization. RISC V is a fairly new architecture, even though majority of companies working on semiconductors have announce their own RISC V research and development, There are only a few end products available in the market for the end-user user case.

A lot of software is written with other ISA in mind, thus software compatibility is also a major issue. RISC V being very recent vs the other widely adopted ISA that there are only a few software available in the market that are natively compatible to RISC V.

Optimization of software is another major problem. The software written for other widely adopted ISAs are being worked on and optimized for regulary for years. This is not the case with RISC V. Even the natively compatible software for RISC V might not be able to take the full advantage of the hardware’s potential.

==

Problem 7):

A):

The formula of speedup is

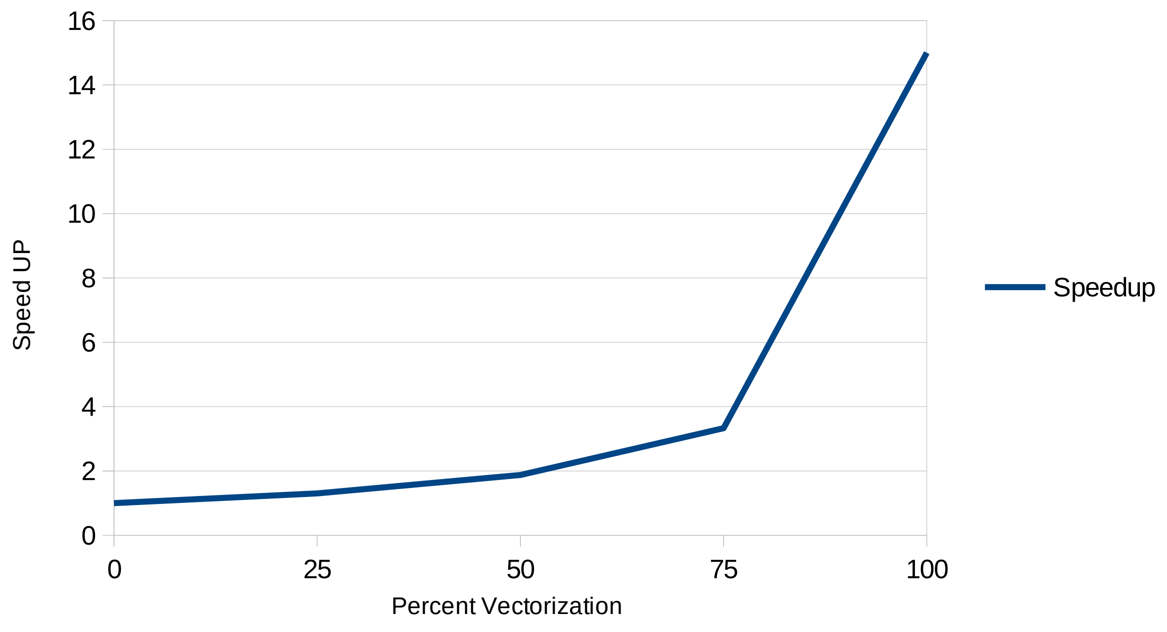
Speedup = $(1/((1 - \text{Fraction}_{\text{Enhance}}) + (\text{Fraction}_{\text{Enhance}} / \text{Speedup}_{\text{Enhance}})))$ --- (7.1)

Let the percent be X and speed up enhance is given as 15

Speedup = $1/((1 – (X/100)) + (X/15*100))$

Speedup = $1 / (1 – (14X/1500))$ ---- (7.2)

Vectorization	Speedup
0	1
25	1.3
50	1.875
75	3.33
100	15



B):
Speedup = 2

Substituting values in formula 7.1

2 = 1 / (1 – (14X/1500))

X = 750/14 = 53.57%
Percentage of vectorization needed is 53.57%

C):
% vectorization when speed up is 2 = 53.57
% non vectorization code when speed up is 2 = 46.43

so if the original time is 100 seconds without speed, vectorization takes 53.37 S and non vectorization takes 46.43 S

When theres speed up of 2, the time will be 50, so vectorization will be 50 – 46.43 = 3.57

so with speed up of 2, 3.57 S are spent in vectorization mode which is 7.14

Percentage Spent in vectorization mode = 7.1401%

D):
Maximum speedup = 1/(1/15) = 15
Achievable speedup = Maximum speedup/2 = 15/2 = 7.5

substituting the values in formula 7.2

7.5 = 1 / (1 – (14X/1500))
X = 92.8571

Percentage of vectorization needed is 92.8571%
E):
Current vectorization = 70% = 0.7

Current speedup = 1(0.3 / 0.7/15) = 2.88462

Target speedup = 5.76923

Substituting the values in Formula 7.2

5.76923 = 1 / (1 – (14X/1500))
X =88.5714

Percentage vectorization needed for new speedup is 88.5714%

===

Problem 8):

A):
Number of computers = 10,000
MTTF of individual computer = 35 days
Catastrophic failure at = 1/3 of total systems

Now,
MTTF of individual system = Number of hours computer worked/Number of computers failed ---- (8.1)

Assume the hours worked by the computer be ***x***, so total hours will be ***x* *** 10000. Substituting the given values in 8.1

35 = (***x*** * 10000)/(10000/3)

***x* =** 11.6667 Days

The MTTF of the system is 11.6667 days

B):
Cost per computer to double MTTF: $1000
Total cost of the to double the MTTF of the system = 1000 * 10000 = 10000000 = 10 Million dollars

Now Doubling the MTTF and making the system more reliable, is a great thing and from the above calculated value, it is clear that spending that amount will double the MTTF from 11.6667 Days to 23.3334 Days deducing from the formula 8.1. Now on surface, 10 Million Dollars sounds a huge sum of money to spend but considering the reliability gains of the overall system thus more requests can be satisfied on timely basis regulary. Thus providing better service and support to the end user. Thus it is a great business decision to spend that amount to increase the reliability and double the MTTF of the computers.

==

Problem 9):

A):
Instructions to execute a program = 1.5E8
Clock rate of processor = 2.7 GHz = 2.7E9 Hz
3 clock instruction = 1.5E8 * 0.7
4 clock instruction = 1.5E8 * 0.2
5 clock instruction = 1.5E8 * 0.1

Time to execute = Total clock cycles / Clock rate ---- (9.1)

Time to execute = ((1.5*0.7*3)+(1.5*0.2*4)+(1.5*0.1*5))E8/2.7E9 = 1.88889E-1 Second
Total time to execute the program = 0.188889 Second

B):
New Clock rate = 1.5 GHz = 1.5E9 Hz
3 clock instruction = 1.5E8 * 0.7
2 clock instruction = 1.5E8 * 0.3

Substituting the new values in formula 9.1,

Time to execute = ((1.5 * 0.7 * 3) + (1.5 * 0.3 * 2))E8 / 1.5E9 = 2.7E-1 Second
Total time to execute the program = 0.27 Second

Performance improvement = $Time_{old}$ / $Time_{New}$

Performance improvement = 0.188889 / 0.27 = 0.699589 ≈ 0.70
The new performance improvement is 0.70 of previous one thus we see an approximately 30% performance reduction

===

Problem 10):

A):
FIT of a single processor = 100

FIT = 10^9 / MTTF ---- (10.1)

Substituting the values in 10.1 we get

MTTF = 10^7 Hours ≈ 416,666.67 Days

B):
Time to repair MTTR = 1 Day

Availability = MTTF / (MTTF + MTTR) ---- (10.2)
Susbstituting the values in 10.2

Availability = 0.999998 ≈ 1

C):
Number of processor = 1000
Number of processor to fail for catastrophic failure = 1

Substituting these values in 8.1, we get

$10^7 = \boldsymbol{x} * 1000 / 1$
$\boldsymbol{x} = 10^4$

The MTTF for the system is 10^4 Hours.

===

Problem 11):
Load on servers = 60%
Power Consumption at 60% = 90% of Maximum power
Power Consumption of new system at barely alive state = 20% of Maximum power

A):
Let the Power be **X** and the number of servers be **Y**
The total system power consumption now is 90% of Maximum power = 0.9 * **X** * **Y**
Now, 60% of systems are shut so Total power = 0.4 * **X** * 0.9 * **Y**

Power savings = Change in power consumption / Old power consumption ---- (11.1)

Substituting values in 11.1,

Power savings = (0.9 – 0.36) / 0.9 = 0.54/0.9 = 60%
New Power savings is 60%

B):
Number of servers in Barely alive state = 60%
New power consumption = 0.4 * **X** * 0.9 * **Y** + 0.6 * **X** * **Y** * 0.2 = 0.48 * **X** * **Y**

Power savings = (0.9 – 0.48) / 0.9 = 0.466667 ≈ 46.6667%
New Power savings is 46.6667%

C):
Voltage reduction = 20%
Frequency reduction = 40%

The Power is directly proportional to the product of capacitive load, square voltage and frequency,
$\text{Power}_{\text{Dynamic}}$ α 0.5 * Capacitive load * Voltage^2 * Frequency ---- (11.2)

Using the formula 11.2, we can deduce the relation between old power and new power

$\text{Power}_{\text{New}}/\text{Power}_{\text{Old}}$ = (0.5*Capacitive Load*$(\text{Voltage}*0.8)^2$*(Frequency*0.6)/(0.5*Capacitive Load*Voltage^2*Frequency) = 0.384

Total power saving = 1 – 0.384 = 0.616 = 61.6%

D):
Number of servers in barely alive state = 0.3
Number of servers in off = 0.3

Total power consumption of the new system = (0.4 * 0.9 + 0.3 * 0.2) * **X** * **Y** = 0.42 * **X** * **Y**

Power saving = (0.9 – 0.42) / 0.9 = 0.53333 ≈ 53.33%

New power savings = 53.33%

==

Problem 12):
CPI Arithmetic Instruction = 1
CPI Load/Store Instruction = 10
CPI Branch Instruction = 3

Program Arithmetic Instruction = 1E8
Program Load/Store Instruction = 2E7
Program Branch Instruction = 2E7

A):
Number of new Arithmetic Instruction = 1E8 * 0.75
Let cycle time be **X**
Increase in cycle time = 10%, New cycle time = 1.1 * **X**

Now, refering the formula 5.1, we can derive a relation

ET_{New}/ET_{Old} = ((0.75*10*1) + (2*10) + (2*3))E7 * 1.1 * **X** / ((10*1) + (2*10) + (2*3)) * **X**

ET_{New} = 1.02361 * ET_{Old}

From the above derived relation, we conclude that the new execution time is higher than the old execution time thus the new design is slower than previous one thus the implementation is not a good choice.

B):
New Arithmetic CPI = 0.5

ET_2/ET_{Old} = ((10*0.5)+*(2*10)*(2*3))E7 * **X** / ((10*1) + (2*10) + (2*3)) * **X**

ET_2 = 0.86 * ET_{Old} = 14% Faster

Thus doubling the Arithmetic Instruction Performance makes the program run 14% Faster than the original Design

New Arithmetic CPI = 0.1

ET_3/ET_{Old} = ((10*0.1) + (2*10) + (2*3))E7 * **X** / ((10*1) + (2*10) + (2*3)) * **X**

ET_3 = 0.75 * ET_{Old} = 25% Faster

Thus increasing the Arithmetic Instruction Performance by Ten times makes the program run 25% Faster than the original Design