NYU Tandon School of Engineering

Spring 2022, ECE 6913

Homework Assignment 1

Instructor Contact : ajb20@nyu.edu

Course Assistants & Graders: Varadraj Kakodkar (vns2008), Kartikay Kaushik (kk4332), Siddhanth Iyer (si2152), Swarnashri Chandrashekar (sc8781), Karan Sheth (kk4332), Haotian Zheng (hz2687), Haoren Zhang (kk4332), Varun Kumar (vs2411)

Homework Assignment 1 [released: Tuesday Sept 6th] [due Friday Sept 16th by 11:59 PM]

You *are allowed* to discuss HW assignments with anyone. You are *not allowed* to share your solutions with other colleagues in the class. Please feel free to reach out to the Course Assistants or the Instructor during office hours or by appointment if you need any help with the HW.
Please enter your responses in this Word document after you download it from NYU Brightspace. *Please use the Brightspace portal to upload your completed HW.*

1. Consider three different processors P1, P2, and P3 executing the same instruction set. P1 has a 3 GHz clock rate and a CPI of 1.5. P2 has a 2.5 GHz clock rate and a CPI of 1.0. P3 has a 4.0 GHz clock rate and has a CPI of 2.2.

a. Which processor has the highest performance expressed in instructions per second?

b. If the processors each execute a program in 10 seconds, find the number of cycles and the number of instructions.

c. We are trying to reduce the execution time by 30%, but this leads to an increase of 20% in the CPI. What clock rate should we have to get this time reduction?

Ans]

a. performance of P1 (instructions/sec) = $3 \times 10^9 / 1.5 = \mathbf{2 \times 10^9}$

performance of P2 (instructions/sec) = $2.5 \times 10^9 / 1.0 = \mathbf{2.5 \times 10^9}$

performance of P3 (instructions/sec) = $4 \times 10^9 / 2.2 = \mathbf{1.8 \times 10^9}$

b. cycles(P1) = $10 \times 3 \times 10^9 = \mathbf{30 \times 10^9}$ **s**

cycles(P2) = $10 \times 2.5 \times 10^9 = \mathbf{25 \times 10^9}$ **s**

cycles(P3) = $10 \times 4 \times 10^9 = \mathbf{40 \times 10^9}$ **s**

c. No. instructions(P1) = $30 \times 10^9 / 1.5 = 20 \times 10^9$

No. instructions(P2) = $25 \times 10^9 /1 = 25 \times 10^9$

No. instructions(P3) = $40 \times 10^9 /2.2 = 18.18 \times 10^9$

CPI new = CPI old × 1.2, then CPI(P1) = 1.8, CPI(P2) = 1.2, CPI(P3) = 2.6

f = No. instr. × CPI/time, then

f (P1) = $20 \times 10^9 \times 1.8/7$ = **5.14 GHz**

f (P2) = $25 \times 10^9 \times 1.2/7$ = **4.28 GHz**

f (P1) = $18.18 \times 10^9 \times 2.6/7$ = **6.75 GHz**

2. Assume for arithmetic, load/store, and branch instructions, a processor has CPIs of 1, 12, 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 256 million **branch** instructions. Assume that each processor has a 2 GHz 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 (ET) 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. If the CPI of the arithmetic instructions were doubled, what would the impact be on the execution time of the program on 1, 2, 4, or 8 processors?

c. 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?

Ans]

a.

Instruction Count [IC]

Arithmetic IC[2] = 2.56B / [0.7 x 2] = 1.83B

Arithmetic IC[4] = 2.56B / [0.7 x 4] = 0.91B

Arithmetic IC[8] = 2.56B / [0.7 x 8] = 0.46B

Load/Store IC[2] = 1.28B / [0.7 x 2] = 0.91B

Load/Store IC[4] = 1.28B / [0.7 x 4] = 0.46B

Load/Store IC[8] = 1.28B / [0.7 x 8] = 0.23B

Branch [1] =Branch [2] = Branch [4] = Branch [8] = 0.256B

Instruction	CPI	IC [1]	IC[2]	IC[4]	IC[8]
Arithmetic	1	2.56B	1.83B	0.91B	0.46B
Load/Store	12	1.28B	0.91B	0.46B	0.23B
Branch	5	0.256B	0.256B	0.256B	0.256B
Total Instruction Count [IC]		**4.096B**	**2.996B**	**1.626B**	**0.946B**

ET (Execution Time) = IC (Instruction Count) x CPI (Cycles per Instruction) x Cycle Time: 1/[2x10^9 Hz]

Instruction	CPI	ET [1]	ET [2]	ET [4]	ET [8]
Arithmetic	1	1x2.56Bx0.5ns=1.28s	1x1.83Bx.5n=0.91s	1x0.91Bx.5n=0.46s	1x0.46Bx0.5n=0.23s
Load/Store	12	12x1.28Bx.5ns=7.68s	12x0.91Bx.5ns=5.46s	12x0.46Bx.5n=2.76s	12x0.23Bx.5n=1.38s
Branch	5	5x 0.256Bx.5ns=0.64s	5x 0.256Bx.5ns=0.64s	5x 0.256Bx.5ns=0.64s	5x 0.256Bx.5ns=0.64s
Total	-	9.6s	7.01s	3.86s	2.17s

b.

ET (Execution Time) = IC (Instruction Count) x CPI (Cycles per Instruction) x Cycle Time: 1/[2x10^9 Hz]

Instruction	CPI	ET [1]	ET [2]	ET [4]	ET [8]
Arithmetic	2	2x2.56Bx0.5ns=2.56s	2x1.83Bx.5n=1.83s	2x0.91Bx.5n=0.91s	2x0.46Bx0.5n=0.46s
Load/Store	12	12x1.28Bx.5ns=7.68s	12x0.91Bx.5ns=5.46s	12x0.46Bx.5n=2.76s	12x0.23Bx.5n=1.38s
Branch	5	5x 0.256Bx.5ns=0.64s	5x 0.256Bx.5ns=0.64s	5x 0.256Bx.5ns=0.64s	5x 0.256Bx.5ns=0.64s
Total	-	10.88s	7.93s	4.31s	2.48s

c.

If a single processor is to execute the same Program in **3.86 seconds** (instead of **9.6 seconds**), then the difference of **5.74 seconds** must come from the improvement in execution time of the *L/S instructions*. This improvement can be accomplished *by reducing L/S instruction CPI from 12 to CPIX*

7.68s-5.74s = 1.94s = *Time now available for single processor to process L/S instructions*

CPIX x 1.28Bx 0.5ns = 1.94s

So, CPIX = 1.94s/[1.28 x 0.5] = 1.94s/0.64 = **3.03**

3. Consider the three different processors P1, P2, and P3 executing the same instruction set. P1 has a clock cycle time of 0.33 ns and CPI of 1.5; P2 has a clock cycle time of 0.40 ns and CPI of 1.0; P3 has a clock cycle time of 0.3 ns and CPI of 2.8.

a. Which has the highest clock rate? What is it?

b. Which is the fastest computer? If the answer is different than above, explain why. Which is slowest?

c. How do the answers for a and b reflect the importance of benchmarks?

Ans]

a. The clock rates are the inverse of the clock cycle time.
P1 = 1/(0.33x10-9 seconds) = 3 GHz;
P2 = 1/(0.40x10-9 seconds) = 2.5 GHz;
P3 = 1/(0.3x10-9 seconds) = 3.33 GHz.

Hence, **P3 has the highest clock rate**.

b. Since all have the same instruction set architecture, all programs have the same instruction count, so we can measure performance as the product of average clock cycles per instruction (CPI) times clock cycle time, which is also the average time of an instruction:
P1 = 1.5 x 0.33 ns = 0.495 ns
P2 = 1.0 x 0.40 ns = 0.400 ns
P3 = 2.8 x 0.3 ns = 0.84 ns

Hence (fastest to slowest)
P2 > P1 > P3

P2 is fastest and P3 is slowest. Despite having the highest clock rate, on average P3 takes so many more clock cycles that it loses the benefit of a higher clock rate.

c. If we go purely by clock rates then clearly P3 is the fastest but in reality this is not true!!! This highlights the importance of the benchmarks.

The CPI calculation was based on running some benchmarks. If they are representative of real workloads, the answers to these questions are correct. If the benchmarks are unrealistic, they may not be. The difference between things that are easy to advertise, like clock rate, and actual performance highlights the importance of developing good benchmarks.

4. You are designing a system for a real-time application in which specific deadlines must be met. Finishing the computation faster gains nothing. You find that your system can execute the necessary code, in the worst case, twice as fast as necessary.

a. How much energy do you save if you execute at the current speed and turn off the system when the computation is complete?
b. How much energy do you save if you set the voltage and frequency to be half as much?

Ans]

a.

Energy is dependent on the number of logic transitions in the system each of which cost $E=CVdd2$. More relevantly, energy consumed is independent of the speed at which these logic transitions complete. So, finishing the computation sooner does not benefit energy reduction.

So, how much energy is saved? **None**

b.

Setting Voltage to half its nominal value lowers energy to ¼ its original value given the quadratic dependence of Energy on operating voltage as we saw in 1a above. Setting frequency to half its nominal value lowers power by half but the energy consumption remains unchanged.

So setting voltage and frequency to half their nominal values lowers Energy to ¼ its nominal value or by 75% and Power to 1/8 its nominal value or by 87.5%

5. Consider two different implementations of the same instruction set architecture. The instructions can be divided into four classes according to their CPI (classes A, B, C, and D). P1 with a clock rate of 2.0 GHz and CPIs of 1, 2, 2, and 1, and P2 with a clock rate of 4 GHz and CPIs of 2, 3, 4, and 4.

a. Given a program with a dynamic instruction count of 1.0E6 instructions divided into classes as follows: 10% class A, 20% class B, 50% class C, and 20% class D. Which is faster: P1 or P2 (in total execution time)?

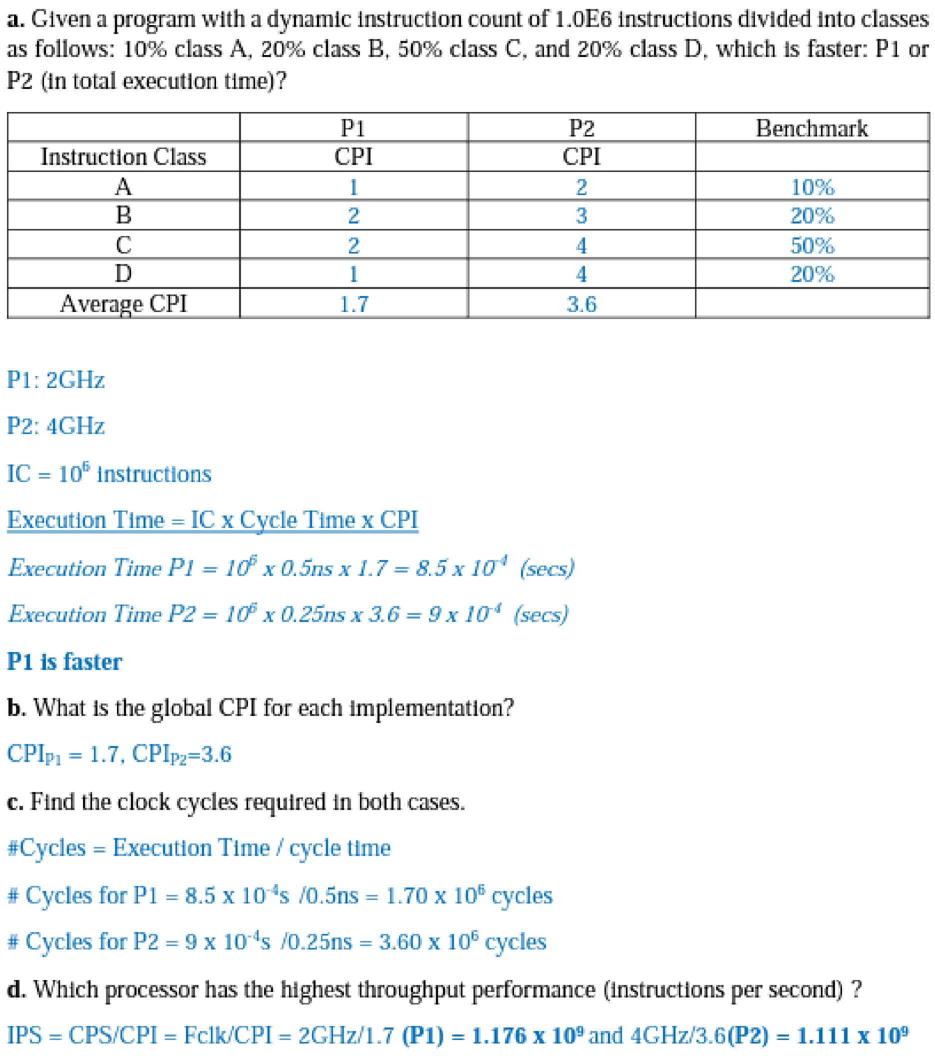
b. What is the global CPI for each implementation?

c. Find the clock cycles required in both cases.

d. Which processor has the highest throughput performance (instructions per second) ?

e. Which processor do you think is more energy efficient? Why?

Ans]



a. Given a program with a dynamic instruction count of 1.0E6 instructions divided into classes as follows: 10% class A, 20% class B, 50% class C, and 20% class D, which is faster: P1 or P2 (in total execution time)?

	P1	P2	Benchmark
Instruction Class	CPI	CPI	
A	1	2	10%
B	2	3	20%
C	2	4	50%
D	1	4	20%
Average CPI	1.7	3.6	

P1: 2GHz

P2: 4GHz

IC = 10^6 instructions

Execution Time = IC x Cycle Time x CPI

Execution Time P1 = 10^6 x 0.5ns x 1.7 = 8.5 x 10^{-4} (secs)

Execution Time P2 = 10^6 x 0.25ns x 3.6 = 9 x 10^{-4} (secs)

P1 is faster

b. What is the global CPI for each implementation?

CPI_{P1} = 1.7, CPI_{P2}=3.6

c. Find the clock cycles required in both cases.

#Cycles = Execution Time / cycle time

Cycles for P1 = 8.5 x 10^{-4}s /0.5ns = 1.70 x 10^6 cycles

Cycles for P2 = 9 x 10^{-4}s /0.25ns = 3.60 x 10^6 cycles

d. Which processor has the highest throughput performance (instructions per second) ?

IPS = CPS/CPI = Fclk/CPI = 2GHz/1.7 **(P1) = 1.176 x 10^9** and 4GHz/3.6**(P2) = 1.111 x 10^9**

e. Which processor do you think is more energy efficient? Why?

P1 completes the same benchmark in less time with fewer clock cycles

6. **a.** What is the difference between CISC and RISC architectures? Give some examples wherein you think CISC architectures are better suited for than RISC architectures and vice versa.

b. Describe in your own words why you think RISC V would be a better alternative compared to ARM or x86 architectures?

c. What do you think are the challenges faced by RISC V architecture going forward?

Ans]

There is no one correct answer for this question! Give your own thoughts and describe in your own words as there is no wrong answer!!!

7. 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."
b. What percentage of vectorization is needed to achieve a speedup of 2?
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 additional 2× speedup in the vector unit (beyond the initial 15×)?

Ans]

a.

Assume TE_{old} is the time taken for the non-vectorized code to run.

Assume 'x' equals the percentage of the code vectorized.

This piece takes 1/15 the time to run as the non-vectorized code. So, [100-x]% of the code is not vectorized.

Total time to run code with x % vectorized = [100 – x]% T0 + [x/10]%T0

if x=0%, then the code takes 100% of TE_{old} time to execute

if x=100%, then code takes 15% of TE_{old} time to execute

Net Speedup $= \boldsymbol{TE_{old}} / \boldsymbol{TE_{new}}$

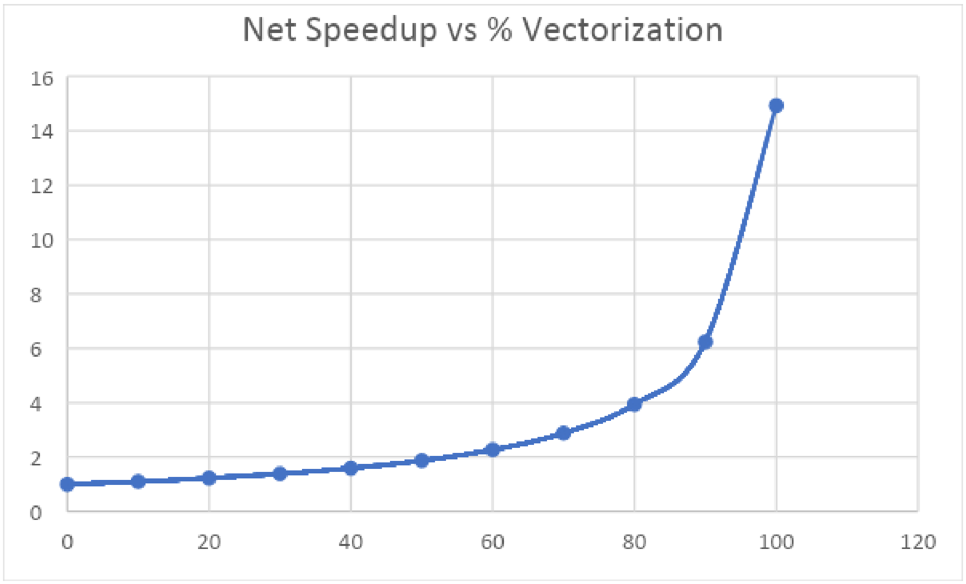
$$= \frac{TEold}{[\%vectorization]\, TEold/15 + [100\% - \%vectorization]\, TEold/1}$$

$$= \frac{1}{0.067[\%vectorization] + [1 - \%vectorization]}$$

$$= \frac{1}{1 - 0.933[\%vectorization]} \qquad \text{-------}(1)$$

Percentage Vectorization	Net Speedup

0	1
10	1.102900629
20	1.229407426
30	1.388696014
40	1.595405233
50	1.874414246
60	2.271694684
70	2.882675123
80	3.943217666
90	6.238303182
100	14.92537313



b.

Net Speedup $= \frac{1}{1-0.933[\%vectorization]}$

2 $= \frac{1}{1-0.933[\%vectorization]}$

0.5 = 1 - 0.933 x

0.933 x = 0.5

x = 0.5359

Hence, for a speedup of 2, we need to have a percentage vectorization of 53.59%.

c. **For 2x speedup, we know that x = 53.59%**

$$\textbf{\% Time spent} = \frac{x/15}{\left(\frac{x}{15}\right)+(100\%-x)}$$

$$= \frac{53.59/15}{\left(\frac{53.59}{15}\right)+(100-53.59)}$$

$$= 7.14\%$$

d.

Maximum speedup from using vector mode = 14.92x

Half of maximum speedup = 7.46x

$$\textbf{Net Speedup} = \frac{1}{1-0.933[\%vectorization]}$$

$$\textbf{7.46} = \frac{1}{1-0.933x}$$

1-0.933x = 0.1340

0.933x = 0.8659

x = 92.80%

i,e 92.80% of code must be vectorized for speedup to be half of maximum:

e.

With a 70% vectorization, speedup would be **100/[30 + 7] = 2.7x**

we have a choice of either

(i) The hardware design group increasing the speedup of the vector hardware to more than 15X with significant investment, or

(ii) The compiler group increasing the speedup with a higher percentage vectorization of the code

This means that,

For (ii) above, if we want to double the speedup accomplished so far of 2.88x with a 70% vectorization, to 5.76x what % vectorization 'y' would be necessary?

$5.76 = \frac{1}{1 - 0.933[\%vectorization]}$

$0.173 = 1 - 0.933[\%vectorization]$

%vectorization = 88.57%

88.57% of the code would need to be vectorized by the compiler group (instead of 70%) to get a 2x improvement over the 2.88x speedup already accomplished with 70% vectorization

8. 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.

Ans]

a.

$MTTF_{computer}$ = 35 days => $FIT_{computer}$ = (1/35) per day

for 1 of 3 computers to fail, FIT_{system} = 3(1/35) failures/day

$MTTF_{system}$ = 1/FIT_{system} = (35/3) days = 11.67 days

Or

The number of failures needed for the system to fail = 10,000 * ⅓ = 3333

FIT(1 computer) = 1/35

FIT(system) = 1/35 * 10,000 = 285

MTTF(system) = number of failures/FIT = 3333/285 = 11.69 days

b.

Cost to double MTTF = \$1000 per computer or \$ 10M for system.

So if a computer costs more than \$1000 its cheaper to simply double the MTTF rather than replace the system.

9. **a.** A program (or a program task) takes 150 million instructions to execute on a processor running at 2.7 GHz. Suppose that 70% of the instructions execute in 3 clock cycles, 20% execute in 4 clock cycles, and 10% execute in 5 clock cycles. What is the execution time for the program or task?

b. Suppose the processor in the previous question part is redesigned so that all instructions that were initially executed in 5 cycles and all instructions executed in 4 cycles now execute in 2 cycles. Due to changes in the circuitry, the clock rate also must be decreased from 2.7 GHz to 1.5 GHz. What is the overall percentage improvement?

Ans]

a.
Inst Count = 150M, FCLK = 2.7GHz, Cycle Time = 1/ [2.7GHz] = 0.37037 ns

Instr Type	% of code	CPI	IC(millions)	No of cycles(millions)	Execution Time(ms)
Instr 1	70	3	105	315	116.67
Instr 2	20	4	30	120	44.44
Instr 3	10	5	15	75	27.78

Total ET = ET1 + ET2 + ET3 = 116.67 + 44.44 + 27.78 = 188.88 ms

b.
Inst Count = 150M, FCLK = 1.5GHz, Cycle Time = 1/ [1.5GHz] = 0.667 ns

Instr Type	% of code	CPI	IC(millions)	No of cycles(millions)	Execution Time(ms)
Instr 1	70	3	105	315	210
Instr 2	20	2	30	60	40
Instr 3	10	2	15	30	20

Total ET = ET1 + ET2 + ET3 = 210 + 40 + 20 = 270 ms

% improvement = 270-188.88 / 270 = 0.43
This is a reduction in performance of **57%**

We see that the new ET is more than the old ET. Therefore, this is not a good design decision as the improvement in CPI is at the cost of reduced clock frequency is not justified.

10. Availability is the most important consideration for designing servers, followed closely by scalability and throughput.

a. We have a single processor with a failure in time (FIT) of 100. What is the mean time to failure (MTTF) for this system?

b. If it takes one day to get the system running again, what is the availability of the system?

c. Imagine that the government, to cut costs, is going to build a supercomputer out of inexpensive computers rather than expensive, reliable computers. What is the MTTF for a system with 1000 processors? Assume that if one fails, they all fail.

Ans]

a.

Given information:

Failure time= 100

MTTF=?

As a metric, MTTF represents how long a product can reasonably be expected to perform in the field based on specific testing.

$$MTTF = \frac{10^9}{100}$$

10^9 represent billion. Generally reported as failures per billion hours of operation or Failure in Time (FIT)

$\boxed{\textbf{MTTF=10}^7}$

b.

Answer: the calculated mean time to failure is 10^7 then we can calculate the availability of the system.

$10^7/10^7 + 24 = 1$ failure

Mean Time to Failure is a reliability measure; the reciprocal of MTTF is a rate of failures. Generally reported as failures per billion hours of operation or Failure In Time (FIT).

c.

MTTF is primarily focused on hardware. If the system contains a 1000 processors then system failure time will take a hit compared to the system which has more nodes installed in the system.

11. 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?

Ans]

a.
Assuming a uniform distribution of load across all servers and assuming the load does not change in each server when turning off any fraction of the servers, 60% of servers turned off, **reduces power by 60%**

b.
40% of the servers would consume power that is unchanged. 60% of the servers would drop their power to 20% of maximum (instead of 90%).
So, total power consumption now is:
= 0.4 x PN + 0.6 x (0.2/0.9) x PN
= [0.4 + 0.133] PN = 0.533 PN or **reduction to 53.3%** of nominal power consumption (PN)

c.
CVV fclk [new] = C (0.8V)(0.8V)(0.6fclk)) = 0.384 x CVV (old)
savings of 61.6% in power achieved

d.
40% of the servers would consume power that is unchanged. 30% are off 30% are in the 'barely alive' state 0.4 x PN + 0.3 x 0 x PN + 0.3 x (0.2/0.9) PN = 0.4666 PN or **reduction to 46.67% of nominal power**

12. Assume for a given processor the CPI of arithmetic instructions is 1, the CPI of load/store instructions is 10, and the CPI of branch instructions is 3. Assume a program has the following instruction breakdowns: 100 million arithmetic instructions, 20 million load/store instructions, 20 million branch instructions.

a. Suppose that new, more powerful arithmetic instructions are added to the instruction set. On average, through the use of these more powerful arithmetic instructions, we can reduce the number of arithmetic instructions needed to execute a program by 25%, while increasing the clock cycle time by only 10%. Is this a good design choice? Why?

b. Suppose that we find a way to double the performance of arithmetic instructions. What is the overall speedup of our machine? What if we find a way to improve the performance of arithmetic instructions by 10 times?

Ans]

a.

Type	CPI	No of Instr (old)	No of Instr (new)
Load/Store	10	20M	20M
Branch	3	20M	20M
Arithmetic	1	100M	75M

Let the cycle period be "T_{cycle}"

ET_{old} = CPI x IC x Cycle Time summed over each instruction
= $(200 + 60 + 100) \times \{T_{cycle}\}$
= $360 \times T_{cycle}$

ET_{new} = $(200 + 60 + 75) \times \{T_{cycle}\} \times 1.1$
= $335 \times \{T_{cycle}\} \times 1.1$
= $368.5 \times x \{T_{cycle}\}$

New execution time is larger, so this is not a good design choice
Speedup = 360/368.5 = 0.9769

This is a 2.30% loss in performance!!!

b.

Type	CPI	No of Instr (old)
Load/Store	10	20M
Branch	3	20M
Arithmetic	0.5	100M

New ET by doubling the performance of arithmetic instructions (half the original CPI)

$= (200 + 60 + 50) \text{ x } T_{cycle}$

$= 310 \text{ x } T_{cycle}$

Speedup = 360M/310M = 1.1612 => **16.12%**

New ET by improving the performance of arithmetic instructions (10x)

$= (200 + 60 + 10) \text{ x } T_{cycle}$

$= 270 \text{ x } T_{cycle}$

Speedup = 360M/270M = 1.3333 => **33.34%**