Problem 1 # Explain instruction-level and machine parallelism and its effect on optimum performance of computer architecture

Instruction level parallelism (ILP) of a program—a measure of the average number of instructions in a program that, in theory, a processor might be able to execute at the same time

• Mostly determined by the number of true (data) dependencies and procedural (control) dependencies in relation to the number of other instructions

• ILP is traditionally "extracting parallelism from a single instruction stream working on a single stream of data"

Machine parallelism of a processor—

Machine parallelism is a measure of the ability of the processor to take advantage of the ILP of the program

• Determined by the number of instructions that can be fetched and executed at the same time

• A perfect machine with infinite machine parallelism can achieve the ILP of a program

Problem 2 a) # Elucidate the law governing the speedup of a program.
b) We are considering an enhancement to the processor of a web server. The new CPU is 20 times faster on search queries than the old processor. The old processor is busy with search queries 70% of the time, what is the speedup gained by integrating the enhanced CPU?

a) Amdahl's law

Amdahl's law sets the limit to which a parallel program can be sped up. Amdahl's Law is a way of showing that unless a program (or part of a program) is 100% efficient at using multiple CPU cores, you will receive less and less of a benefit by adding more cores. At a certain point - which can be mathematically calculated once you know the parallelization efficiency - you will receive better

performance by using fewer cores that run at a higher frequency than using more cores that run at a lower frequency.

Amdahl's Law:

$$S(n) = \frac{T(1)}{T(n)} = \frac{T(1)}{T(1)(B + \frac{1}{n}(1 - B)}$$

$S(n)$ is the theoretical speedup

$T(n)$ is the time an algorithm takes to finish when running ***n*** threads

B is the fraction of the algorithm that is strictly serial (so 1-***B*** is how much of the program can be run in parallel)

Unless you deal with complex equations regularly, this may be a bit daunting of an equation. However, since we are primarily concerned with the maximum speedup that can be achieved by increasing the number of CPU cores, this equation can have simplified a bit into the following:

Parallelization Formula:

$$S(n) = \frac{1}{(1 - P) + \frac{P}{n}}$$

$S(n)$ is the theoretical speedup

P is the fraction of the algorithm that can be made parallel

The way to think about Amdahl's law is that the faster the parallel section of the code run, the more the remaining sequential code looms as the performance bottleneck. In the limit, if the parallel section is responsible for 80 percent of the run time, and that section is sped up infinitely (so that it runs in zero time), the other 20 percent now constitutes the entire run time. It would therefore have been sped up by a factor of 5, but after that no amount of additional parallel hardware will make it go any faster.

b) Speedup is given as

$$S(n) = \frac{1}{(1-P) + \frac{P}{n}}$$

$S(n)$ is the theoretical speedup

P is the fraction of the algorithm that can be made parallel.

Here, P = 70% = 0.7

n = 20

(1-P) = 0.80

S(n) = 1/ 0.335 = 2.985

3) You designed a CPU with a clock that runs at 200MHz. The CPU design is highly efficient in doing FP calculations. After running a benchmark on the CPU you find the following observations -

Instruction Type	Frequency	Cycles
Load and Stores	30%	6
Arithmetic	50%	4
FP instructions	20%	3

i) Calculate the CPI for the benchmark result on your CPU.
ii) What is the processor speed for the benchmark in MIPS?
iii) On doubling the registers in the register file, you find out that the processor cycle time increases by 20%. What would be the new clock frequency?

i) Clock frequency = 200MHz
Frequency of load/store instructions = 30%
cycles to execute load/store instructions = 6
Frequency of arithmetic instructions = 50%
cycles to execute arithmetic instructions = 4
Frequency of FP instructions = 20%
cycles to execute FP instructions = 3

Let's assume the total instruction executed be 100.
30 – load/stores 50 – arithmetic instructions 20 – FP instructions

CPI = (30*6) + (50*4) + (20 * 3)/ (100)
CPI = 4.4

ii) The formula for MIPS = Clock rate/(CPI * 10^6)
Therefore, MIPS = ($200x10^6$)/($4.4x10^6$) = 45.45

iii) Clock cycle = 1/Clock freq = 1/($200*10^6$) = 5ns
Since the cycle time is increased by 20%, hence -
Increased time = 5 * 20/100 ns = 1 ns
Therefore new clock cycle = 5 + 1 ns = 6ns
Clock period = 1/(6ns) = 166.67 MHz

4) Suppose you are designing the floating point unit of a processor. After running a benchmark you discover that the program consists of 50% floating point multiply, 20% floating point divide, and the remaining 30% are from other instructions. The management wants the machine to be 4 times faster and identifies certain bits of improvement in the floating point unit. After initial triage you get to know that it is possible to make the divide run at most 3 times faster and the multiply run at most 8 times faster. But both the improvements cannot occur together in the system as then the area requirements aren't met. Which of the improvement would meet the goal?

Let's assume 100 instructions are executed.
Number of FP multiply = 50
Number of FP divide = 20
Other instructions = 30
Using amdahl's law for finding out the improvement = (Execution time affected by the improvement/(Improvement possible) + Unaffected execution time
Improvement in FP multiply = (50)/(8) + (20 + 30) = 66.67
Improvement in FP divide = (20)/(3) + (50 + 30) = 86.67

Hence the goal is achieved when improvement is done to the multiply unit.

5)The design team of a processor wants to analyze the improvements seen on adding a new Multimedia Co-processor. When a benchmark is run on the Co-pro it is seen that the execution is 10 times quicker. Let the percentage of time for which the Co-pro is active be known as the percentage of co-pro operation.
i) What percentage of co-pro operation is needed to achieve an overall speedup of 2?
ii) If a speedup of 2 is achieved what amount of run-time is spent in Co-pro active mode?
iii) In the Co-pro mode what amount of enhancement time is required to achieve one-half the maximum speedup possible from the co-pro mode?

i) Amdahl's Law - (Execution time affected by the improvement/(Improvement possible) + Unaffected execution time

Let x be the percent of co-pro operation needed for achieving the required speed-up,

(100)/2 = (x)/10 + (100-x)

=> x = 55.55

ii) Since the speed up of 2 is achieved, overall time now will be = 100/2 = 50.
From part a, we know that 55.55 of the original program used co-pro operation.
Now, we can assume x to be the percentage of the updated overall time spent in the co-pro mode with a speedup of 2
=> x = (55.55 *50)/100 = 27.78
Hence, 27.78% of time is spend in co-pro active mode

iii) For the speed up to be maximum, we can assume the entire program runs on the co-processor. Hence, the maximum speedup would be 10. One-half of this is 5. Plugging in 5 instead of 2 in part I, we get:

(100)/5 = (x)/10 + (100-x)

x = 88.89
Hence about 88.89 time the co-pro mode should be active.