1.

MIPS(Millions of Instructions Per Second) is the rate at which instructions are executed. MIPS can be expressed in terms of clock rate and CPI as follows

$$\text{MIPS} = \frac{I_c}{T * 10^6} = \frac{f}{CPI * 10^6}$$

2.

Total time to run the program is given by

$$T = \frac{\sum_{i=1}^{N} CPI_i * I_i}{f}$$

$$T = \frac{(30 + 90 + 24 + 42) * 10^3}{2 * 10^9}$$

$$\mathbf{T = 93 * 10^{-6}\ s}$$

CPI is given by the formula

$$\text{CPI} = \frac{\sum_{i=1}^{N} CPI_i * I_i}{I_c}$$

$$= \frac{(30 + 90 + 24 + 42) * 10^3}{(30 + 45 + 8 + 21) * 10^3}$$

CPI = 1.78

MIPS rate is given by the formula

$$\text{MIPS} = \frac{I_c}{T * 10^6} = \frac{f}{CPI * 10^6}$$

$$= \frac{2 * 10^9}{1.78 * 10^6}$$

MIPS = 1123.595

3.

If the address are staged into two part we can read two bytes of memory in parallel. We can read or write two bytes of memory in one cycle, where earlier we were able to read/write only one byte. This is done to increase the Throughput. Hence the performance of the system is increased.

4.

Ahmdahl's law talks about the speedup of the program resulted by using multiple processors. Consider a program running on a single processor such that a fraction (1 - *f*) of the execution time involves code that is inherently sequential, and a fraction *f* that involves code that is infinitely parallelizable with no scheduling overhead. Let *T* be the total execution time of the program using a single processor. Then the speedup using a parallel processor with *N* processors that fully exploits the parallel portion of the program is as follows:

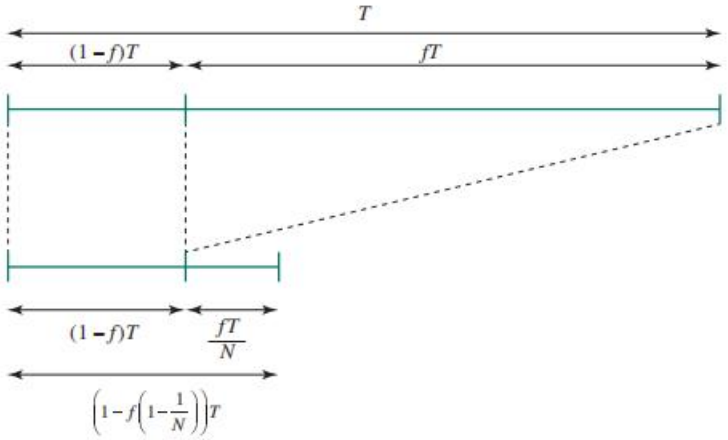
$$\text{Speedup} = \frac{Time\ to\ execute\ on\ single\ processor}{Time\ to\ execute\ on\ N\ paraller\ processor}$$

Time to execute on N parallel processor is given by = T(1-f) + Tf/N

The 1-f is the sequential part and f/N is the parallel part.

Speedup = $\frac{T}{((1-f)+f/N)T}$

Speedup = $\frac{\mathbf{1}}{(\mathbf{1}-\boldsymbol{f})+\frac{\boldsymbol{f}}{\boldsymbol{N}}}$



5.

A benchmark suite is a collection of programs, defined in a high-level language, that together attempt to provide a representative test of a computer in a particular application or system programming area. The best known such collection of benchmark suites is defined and maintained by the Standard Performance Evaluation Corporation (SPEC), an industry consortium. This organization defines several benchmark suites aimed at evaluating computer systems. These performance measures are widely used for comparison and research purpose.

There is no clear definition in the textbook about Components of a CPU Benchmark I am assuming it to be terms in the SPEC Benchmark and discussing it.

Base metric: These are required for all reported results and have strict guide lines for compilation. In essence, the standard compiler with more or less default settings should be used on each system under test to achieve comparable results.

Peak metric: This enables users to attempt to optimize system performance by optimizing the compiler output. For example, different compiler options may be used on each benchmark, and feedback-directed optimization is allowed.

Speed metric: This is simply a measurement of the time it takes to execute a compiled benchmark. The speed metric is used for comparing the ability of a computer to complete single tasks.

We need mean of different metrics to evaluate the performance of an computer system, there are multiple algorithms to calculate this mean value. Some mean calculating algorithms are AM, GM and HM.

AM:

An AM is an appropriate measure if the sum of all the measurements is a meaningful and interesting value. The AM is a good candidate for comparing the execution time performance of several systems. For example, suppose we were interested in using a system for large-scale simulation studies and wanted to evaluate several alternative products.

On each system we could run the simulation multiple times with different input values for each run, and then take the average execution time across all runs. The AM of all the runs is a good measure of the system's performance on simulations, and a good number to use for system comparison.

HM:

For some situations a system execution rate may be viewed as a more useful measure of the value of the system. In this case mean rate is not proportional to the total rate which is some of individual rate. In this case we want the mean to be proportional to the total execution time so we use HM instead of AM.

GM:

GM gives equal weight to all of the values in the data set. GM gives consistent results regardless of which system is used as a reference. Because benchmarking is primarily a comparison analysis, this is an important feature. GM is less biased by outliers than the HM or AM.
GM can be described as the back-transformed average of a lognormal distribution.

We use AM for Speed metric and Peak metric. HM for rate metric. For Base metric If same units AM should be considered and If different units GM should be considered.

6.

Some of the techniques used in the contemporary processors to increase the performance are as follows.

Pipelining: Multiple instructions are executed in parallel. There are different stages in an instruction execution like fetch, decode and execute. When we are executing one instruction another instruction is fetched at the same time. This increase the speed.

Branch prediction: The processor looks ahead in the instruction code fetched from memory and predicts which branches, or groups of instructions, are likely to be processed next. In this way it can pre-fetch the instructions way ahead of time.

Superscalar execution: This is the ability to issue more than one instruction in every processor clock cycle.

Data flow analysis: The processor analyzes which instructions are dependent on each other's results and creates an optimized schedule of instructions. This prevents unnecessary delay.

Speculative execution: Using branch prediction and data flow analysis, some processors speculatively execute instructions ahead of their actual appearance in the program execution. This keeps the processor busy most of the time.

As components on the chip decrease in size, the wire interconnects become thinner, increasing resistance. Also, the wires are closer together, increasing capacitance. The increase in resistance and capacitance increases the RC-Delay and in turn this increases the propagation delay.

7.

Effective Speedup is given by the formula.

$$\text{Speedup} = \frac{\mathbf{1}}{(\mathbf{1}-\boldsymbol{f}) + \frac{\boldsymbol{f}}{\boldsymbol{K}}}$$

Here K =2 and f=0.25

$$\text{Speedup} = \frac{\mathbf{1}}{(\mathbf{1}-\mathbf{0.25}) + \frac{\mathbf{0.25}}{\mathbf{2}}}$$

$$= 1.142$$

Effective Speedup is = **1.142**