Computer Organization 1

Types of Computers

1. Personal Computers (PCs)
   - Intended for a single user at a stationary location
   - Notebooks and workstations
   - Emphasize good performance to single users at low cost
2. Servers
   - Accessed by other computers to provide computation and/or data
   - Typically only accessed via a network
   - Greater computing, storage, and I/O capacity
   - Emphasis on performing well under large workloads with enhanced dependability
3. Embedded Computers
   - **Most Prevalent type of computer/computer class**
   - Computers contained in other devices
   - Usually a small number of predetermined applications
   - Emphasis on cost and low power
4. Personal Mobile Device
   - Battery-powered wireless devices with multimedia user interfaces
   - Smart phones and tablets
   - Reliance on touch screens
   - Emphasis on cost and energy efficiency
5. Large Cluster/Warehouse-Scale-Computers (WSCs)
   - Large collections of servers connected by a network to act as a single powerful computer
   - Scalability and availability handled through the network

Eight Great Architecture Ideas

1. Design for Moore's Law
2. Abstraction
3. Make the common case fast
4. Parallelism
5. Pipelining
6. Prediction
7. Hierachy
8. Improve dependability via redundancy

Steps for executing a program

1. Input device loads the machine code from the executable

2. The machine code is stored in memory
3. Processor fetches an instruction
4. Control decodes the instruction
5. Datapath executes the instruction
6. If application does not complete, then go to step 3

REMEMBER: When executing a program, you first decode the instructions for the control, and then execute the instructions.

REMEMBER: QTSpim is **not** a compiler, it is an **assembler.**

Formulas to remember:

$$\text{Dies per Wafer} \approx \frac{\text{Wafer Area}}{\text{Die Area}}$$

$$\text{Yield} = \frac{1}{(1 + (\text{Defects per area})(\frac{\text{Die Area}}{2}))^2}$$

$$\text{Cost per Die} = \frac{\text{Wafer Cost}}{(\text{Die per Wafer}) * \text{yield}}$$

When comparing performance between Computer_x and Computer_y :

$$\text{Performance} = \frac{1}{\text{Execution Time}}$$

$$\text{Performance}_x > \text{Performance}_y$$

$$\frac{1}{\text{Execution Time}_x} > \frac{1}{\text{Execution Time}_y}$$

$$\text{Execution Time}_y > \text{Execution Time}_x$$

Finding CPU Time:

$$\text{CPU Time} = \text{CPU Clock Cycles} * \text{CPU Clock Cycle Time} = \frac{\text{CPU Clock Cycles}}{\text{CPU clock rate}}$$

$$\text{CPI} = \frac{\text{CPU Clock Cycles}}{\text{Instruction Count}}$$

$$\text{CPU Time} = \text{Instruction Count} * \text{CPI} * \text{CPU Clock Cycles}$$

Relationship between clock rate and clock speed rotation:

$$\text{clock rate} = \frac{1}{\text{clock speed}}$$

$$\text{clock speed} = \frac{1}{\text{clock rate}}$$

Calculate Overall Speedup

$$\text{execution time}_{new} = \text{execution time}_{old} * (1 - \text{fraction}_{enhanced} + \frac{\text{fraction}_{enhanced}}{\text{speedup}_{enhanced}}$$

$$\text{speedup}_{overall} = \frac{\text{execution time}_{old}}{\text{execution time}_{new}} = \frac{1}{(1 - \text{fraction}_{enhanced}) + \frac{\text{fraction}_{enhanced}}{\text{speedup}_{enhanced}}}$$

Terms to know:

1. **Latency Response (execution time)**: The time between the start and completion of an event or task.
2. **Bandwidth/Throughput**: The total amount of work done in a given period of time.
3. **Clock cycles Per Instruction (CPI)**: Average number of clock cycles per instruction for a program or process
4. **Amdahl's Law**: The performance improvement gained from using an enhanced component is limited by the portion improved.

REMEMBER: The memory access bottleneck is caused by fast execution time and slower memory access time. To solve this we use **3 layers of cache**

Conversions:
1 Kib (Kibibyte) $= 2^{10}$ bytes or 1024 bytes
1 Kib $= 1024$ bytes $= 1.024$ KB
For memory: 2^n can store $\{0, 1, ..., 2^{n-1}\}$ bytes.

Power Issues:
Energy is the capacity to change an object's state. Measured in joules.
One joule is equal to one Newton acting through one meter.
Power is the energy amount used over a period of time, units are Watts.
Watts $= \frac{joules}{second}$

$$\text{Power} = \frac{\Delta \text{ Energy}}{\Delta \text{ Time}}$$

1. There is a need for energy efficient Processors becuase of the **major issue of *heat***.
2. Complementary Metal Oxide Semiconductors (CMOSs):
   (a) dominant technology for integrated circuits, energy consumption consists of dynamic and static energy.
   (b) **Dynamic Energy:** The energy consumed by the switching of transistors from 0 to 1 or 1 to 0, this is dependent on the capacitive loading of each transistor and the voltage applied. The **Dynamic Power** is the power required per transistor
      i. Energy $\propto \frac{1}{2} *$ Capacitive load $*$ Voltage^2
      ii. Power $\propto \frac{1}{2} *$ Capacitive load $*$ Voltage^2 $*$ Frequency switched
      iii. $\therefore$ Power $\propto$ Energy $*$ Frequency Switched
   (c) **Static Energy:** energy consumed through current leakage, this is the current that flows through a transistor even when it is off (approximately 40% of power consumption.)
   (d) **Static power:** power lost from static energy, is proportional to the number of

Techniques to Improve Energy Efficiency

1. Turn off the clock for inactive modules (do nothing efficiently)
2. Use a lower clock frequency during periods of low activity (called dynamic frequency scaling or DFS), can often allow lower voltages as well
3. Use a low power mode for memory and storage when not being accessed
4. Completely turn off power to subsets of the chip when not being used

Number Notation The value of a specific numer in a specified base (radix) is calculated by:

$$\sum_{i=-m}^{n-1} \mathrm{d}_i * \mathrm{b}^i = \mathrm{d}_{n-1} * 10^{n-1} + ... + \mathrm{d}_1 * 10^1 + ... + \mathrm{d}_{-m} * 10^{-m}$$

where d is a digit and b is the base.
EXAMPLE

$$425.34_{10} = \sum_{i=-2}^{3} \mathrm{d}_i * 10^i = (4 * 10^2) + (2 * 10^1) + (5 * 10^0) + (3 * 10^{-1}) + (4 * 10^{-2}) = 400 + 20 + 5 + 0.3 + 0.04$$

$$1001.01_2 = \sum_{i=-2}^{4} \mathrm{d}_i * 2^i = (1*2^3)+(0*2^2)+(0*2^1)+(1*2^0)+(0*2^{-1})+(1*2^{-2}) = 8+0+0+1+0+0.25 = 9.25$$

Range of values: To find the range of numbers that can be represented by a certain base with n digits, use the formula M^N where M =base and N =number of digits.
So a base 10 number with 3 decimal number values can represent $M^N = 10^3 = 1000$ numbers, the range would be $[0, 999]$. If it was base 2, then $M^N = 2^3 = 8$ or $[0, 7]$