# Introduction to Computer Architecture

# Principles of Computer Architecture

- Outline:
  - 1. Performance measures
  - 2. High Speed Arithmetic Techniques
    - a) Fast Adder/Subtractor
    - b) Fast Multiplier/Divider
  - 3. Memory Hierarchy, Organization and Design
    - Virtual Memory
    - Cache Memory
    - Interleaved Memory
    - d) Associative Memory
  - 4. Study of Advanced Processor Features
    - a) Uniprocessor (RISC, CISC)
    - b) Stack Machines
    - c) Pipelining and Pipeline Design
    - d) Fine Grain Parallelism
  - 5. Instruction Level Parallelism
  - 6. Study of a multifunctional system
  - 7. How to break RISC barrier/superscalar/VLIW/super pipeline
  - 8. Study of Pentium/Power PC/Multicore architecture

# Principles of Computer Architecture

- Outline
  - Some Performance Measure and Performance Metrics (I am expecting you to be familiar with this subject, if not, please see module1.background)
  - Amdahl's law
  - Green computing
  - CPU Time
  - Formulation of CPU time in terms of Instruction count, clock cycle time, and number of clock cycles per instruction
  - Formulation of CPU time in terms of Instruction count, clock cycle time, number of clock cycles per instruction, and role of different components in a simple computer organization
  - How to improve performance?

# Principles of Computer Architecture

- Introduction
  - Computer Architecture refers to the attributes of a system visible to a programmer — i.e., attributes that have a direct impact on the logical execution of a program.
  - Architectural Attributes include the instruction set, the number of bits used to represent various data types, I/O mechanisms, and techniques for addressing memory.

# Principles of Computer Architecture

- Introduction
  - Computer organization refers to the operational units and their interconnections that realize the architectural specifications.
  - Organizational attributes include those hardware details transparent to the programmer: control signals, interfaces between the computer and peripherals, the memory technology, ...

## Principles of Computer Architecture

- Introduction
  - Whether or not a computer can support a multiplication instruction is an architectural issue.
  - However, whether the multiplication is performed by a special multiply unit or by a mechanism that makes repeated use of the add unit is an organizational issue.

# Principles of Computer Architecture

- Introduction
  - Following IBM, many computer manufacturers offer a family of computer models — all with the same architecture but with differences in organization.
  - An architecture may survive many years but its organization changes with changing technology.
  - In short, as the technology changes, the organization changes while architecture may remain unchanged.

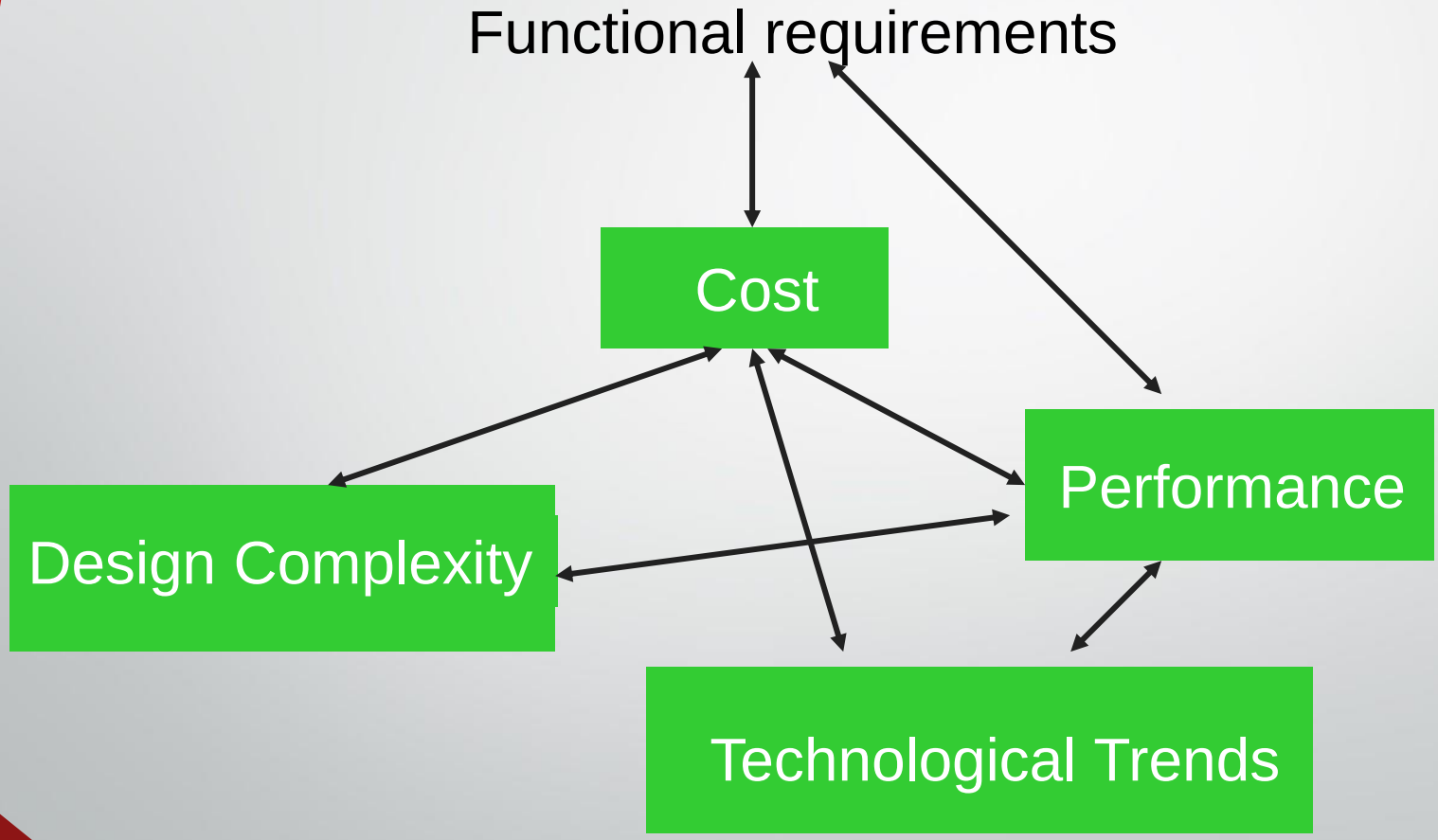
# Principles of Computer Architecture

- Introduction
  - A computer architect is concerned about:
    - The form in which programs are represented to and interpreted by the underlying machine,
    - The methods with which these programs address the data, and
    - The representation of data.

# Principles of Computer Architecture

- Introduction
  - A computer architect should:
    - Analyze the requirements and criteria — Functional requirements
    - Study the previous attempts
    - Design the conceptual system
    - Define the detailed issues of the design
    - Tune the design — Balancing software and hardware
    - Evaluate the design
    - Implement the design — Technological trend

# Principles of Computer Architecture



# Principles of Computer Architecture

- Performance Measures
  - Speed up — How much faster a task will run using the machine with enhancement relative to the original machine.

$$S = \frac{\text{Execution time on Original Machine}}{\text{Execution time on Enhanced Machine}}$$

# Principles of Computer Architecture

- Performance Measures
  - Efficiency — It is the ratio between speed up and number of processors involved in the process:

$$E_p = \frac{S_p}{p}$$

# Principles of Computer Architecture

- Performance Measures
  - Efficiency can be discussed, mainly, within the scope of concurrent system.
  - Efficiency indicates how effectively the hardware capability of a system has been used.
  - Assume we have a system that is a collection of ten similar processors. If a processor can execute a task in 10 seconds then ten processors, collectively, should execute the same task in 1 second. If not, then we can conclude that the system has not been used effectively.

# Principles of Computer Architecture

- Performance Measures
  - Green computing
  - Power consumption and power management

# Principles of Computer Architecture

- Is one number enough?
  - As per our discussion, so far, performance was the major design constraint.  However, the power is becoming a problem.
  - Power consumption became an issue with the growth of wireless technology and mobile devices.  However, it is becoming even more of concern since feeding several Magawatt of power to run a supercomputer is not a trivial task and requires a great amount of supporting infrastructure

# Principles of Computer Architecture

- Challenges for creating Exaflops machine are:
  - Energy and Power,
  - Memory and Storage,
  - Concurrency and locality, and
  - Resiliency
- An Exaflps machine should consume at most 20 Megawatt of power which corresponds to 50 Gflop/W. To reach this goal, power efficiency needs to be increased by a factor of 25 compared to today's most power efficient system (IBM Blue Gere/Q)

# Principles of Computer Architecture

- Performance Measures
  - Amdahl's law — The performance improvement gained by improving some portion of an architecture is limited by the fraction of the time the improved portion is used — a small number of sequential operations can effectively limit the speed up of a parallel algorithm.

# Principles of Computer Architecture

- Performance Measures
  - Amdahl's law allows a quick way to calculate the speed up based on two factors — The fraction of the computation time in the original task that is affected by the enhancement, and, the improvement gained by the enhanced execution mode (speed up of the enhanced portion).

# Principles of Computer Architecture

$$Execution\text{–}time_{new} = Execution\text{–}time_{old} \times \left( \left(1 - Fraction_{enhanced}\right) + \frac{Fraction_{enhanced}}{Speedup_{enhanced}} \right)$$

$$Speedup_{overall} = \frac{Execution\text{–}time_{old}}{Execution\text{–}time_{new}} = \frac{1}{\left( \left(1 - Fraction_{enhanced}\right) + \frac{Fraction_{enhanced}}{Speedup_{enhanced}} \right)}$$

- Performance Measures — Amdahl's law

$$S \leq \frac{\text{Execution time}_{\text{old}}}{\text{Execution time}_{\text{enhanced}}} = \frac{1}{\mathbf{f} + (1 - \mathbf{f}) / \mathbf{p}}$$

Where ***f*** and ***p*** represent the unchanged portion and the speed up of the enhanced portion, respectively.

# Principles of Computer Architecture

- Performance Measures
  - Example — Suppose we are considering an enhancement that runs 10 times faster, but it is only usable 40% of time. What is the overall speed up?

$$S = \frac{1}{\left(.6 + \frac{.4}{10}\right)} = \frac{1}{.64} = 1.56$$

# Principles of Computer Architecture

- Performance Measures
  - Example — If 10% of operations, in a program, must be performed sequentially, then the maximum speed up gained is 10, no matter how many processors a parallel computer has.

# Principles of Computer Architecture

- Performance Measures
  - Example — Assume improving the CPU by a factor of 5 costs 5 times more. Also, assume that the CPU is used 50% of time and the cost of the CPU is 1/3 of the overall cost. Is it cost efficient to improve this CPU?

# Principles of Computer Architecture

$$S = \frac{1}{.5 + \frac{.5}{5}} = \frac{1}{.6} = 1.67$$

- Performance Measures

$$cost - of - the - new - machine = \frac{2}{3} \times 1 + \frac{1}{3} \times 5 = 2.33 times - of - the - original - machine$$

# Principles of Computer Architecture

- Performance Measures
  - Response Time is defined **(**Execution time, Latency**)** as the time elapse between the start and the completion of an event. It is the latency to complete a task that includes disk accesses, memory accesses, I/O activities, operating system overhead, …
  - Is response time a good performance metric?

# Principles of Computer Architecture

- Performance Measures
  - The processor of today's computer is driven by a clock with a constant cycle time ($\tau$).
  - The inverse of the cycle time is the clock rate (f).
  - The size of a program is determined by its instruction count ($I_c$) — number of the machine instructions to be executed.

# Principles of Computer Architecture

- Performance Measures
  - Let us define the average number of clock cycle per instruction (*CPI*) as:

$$CPI = \frac{\text{CPU clock cycles for a program}}{\text{Instruction count}} = \frac{\sum_{i=1}^{n} (CPI_i * I_i)}{I_c}$$

$$= \sum_{i=1}^{n} \left(CPI_i * \frac{I_i}{I_c}\right)$$

Where $I_i$ is the number of time instruction $i$ is executed and $CPI_i$ is the average number of clock cycles for instruction $i$.

# Principles of Computer Architecture

- Performance Measures
  - For a given instruction set, one can calculate the CPI over all instruction types, if the frequencies of the appearance of the instructions in the program is known.
  - *CPI* depends on the organization/architecture and the instruction set of the machine.
  - Clock rate depends on the technology and organization/architecture of the machine.
  - Instruction count depends on the instruction set of the machine and compiler technology.

# Principles of Computer Architecture

- Performance Measures
  - The CPU time (T) is the time needed to execute a given program, excluding the time waiting for I/O or running other programs.
  - CPU time is further divided into:
    - The user CPU time and
    - The system CPU time.

# Principles of Computer Architecture

- Performance Measures

$$\text{CPU}_{\text{Time}} = \text{CPU}_{\text{Clock cycles}} * \text{Clock}_{\text{Cycle time}}$$

$$\text{CPU}_{\text{Time}} = \frac{\text{CPU}_{\text{Clock cycles}}}{\text{Clock}_{\text{Rate}}}$$

- The CPU time is estimated as

$$T = I_c * CPI * \tau = \sum_{i=1}^{n} \left(CPI_i * I_i\right) * \tau$$

# Principles of Computer Architecture

- Performance Measures
  - Example — It takes 10 seconds to run a program on machine *A* that has a 400 MHz clock rate.
  - We are intended to build a faster machine that will run this program in 6 seconds. However, machine *B* requires 1.2 times as many clock cycles as machine *A* for this program. Calculate the clock rate of machine *B*:

# Principles of Computer Architecture

- Performance Measures

$$CpuTime_A = CPUClockCycle_A / ClockRate_A$$

$$10 = CPUClockCycle_A / 400 * 10^6 \, Cycles / Second$$

$$CPUClockCycle_A = 4000 * 10^6$$

$$CPUTime_B = 1.2 * CPUClockCycle_A / ClockRate_B$$

$$ClockRate_B = 1.2 * 4000 * 10^6 / 6 = 800 MHz$$

# Principles of Computer Architecture

- Performance Measures
  - Example — Two machines are assumed: In machine *A* conditional branch is performed by a compare instruction followed by a branch instruction. Machine *B* performs conditional branch as one instruction.
  - On both machines, conditional branch takes two clock cycles and the rest of the instructions take 1 clock cycle. 20% of instructions are conditional branches.
  - Finally, clock cycle time of *A* is 25% faster than *B*'s clock cycle time. Which machine is faster?

# Principles of Computer Architecture

- Performance Measures

$CPI_A = .8*1+.2*2 = 1.2$

$t_B = t_A*1.25$

$CPU_A = I_{CA}*1.2* t_A$

$CPI_B = .25*2+.75*1 = 1.25$

$CPU_B = .8I_{CA}*1.25t_A*1.25 = I_{CA}*1.25*t_A$

  - So *A* is faster.

- Performance Measures
  - Example — Now assume that cycle time of *B* can be made faster and now the difference between the cycle times is 10%. Which machine is faster?

$$CPU_A = I_{CA} * 1.2 * t_A$$

$$CPU_B = .8I_{CA} * 1.1t_A * 1.25 = I_{CA} * 1.1 * t_A$$

  - Now *B* is faster.

# Principles of Computer Architecture

- Performance Measures
  - The execution of an instruction requires going through the instruction cycle. This involves the instruction fetch, decode, operand(s) fetch, execution, and store result(s):

$$T = I_c * CPI * \tau = I_c * (p+m*k) * \tau$$

- Performance Measures
  - The equation

$$T = I_c * CPI * \tau = I_c * (p+m*k) * \tau$$

is the major basis for this course. We will refer to this equation through out the course.

# Principles of Computer Architecture

- Performance Measures
  - *P* is the number of processor cycles needed to decode and execute the instruction, *m* is the number of the memory references needed, and *k* is the ratio between memory cycle time and processor cycle time, memory latency.

$$T = I_c * CPI * \tau = I_c * (p+m*k) * \tau$$

- With respect to the *CPU* time
  in the following sections we will study two major issues:
  - Design and implementation of ALU in an attempt to reduce *P*,
  - Design and implementation of memory hierarchy in an attempt to reduce *m* and *k*.

# Principles of Computer Architecture

- Question
  - With respect to our earlier definition of *CPU* time, discuss how the performance can be improved?

$$T = I_c * CPI * \tau = \sum_{i=1}^{n} \left( CPI_i * I_i \right) * \tau$$

# Principles of Computer Architecture

- In response to this question, the *CPU* time can be reduced by reducing the $I_C$, *CPI*, and/or $\tau$.
- Note the performance improvement with respect to the $\tau$ due to the advances in technology is beyond the scope of this discussion.