Introduction

- Classes of Computers:
  - Personal Computers
  - Server Computers
  - Super Computes
  - Embedded Computers

- Below your program
  - Application software: HLL
  - System software: Compiler and Assembler
  - Hardware

- Levels of Program Code
  - HLL: C, Java
  - Assembly language: embly
  - Hardware representation: 0s and 1s

- Layers of Abstractions
  - Old School Machine Structures
    - Software
      - Application
      - OS
      - Compiler
      - Assembler
    - Hardware
      - Processor, memory, I/O
      - Datapath & Control
      - Digital Design
      - Circuit Design
      - Transistors

    - New School Machine Structures
        - Software
            - Parallel requests
            - Parallel threads
            - Parallel instructions
            - Parallel data
            - Hardware descriptions
        - Hardware
            - Core - cores
                - Instruction Units
                - Functional Units
                    - Logic gates
            - Cache Memory
            - I/O
            - Main Memory
- Eight great ideas:
    - Moore's Law
    - Abstraction
    - Make the common case fast*
    - parallelism
    - pipelining
    - prediction
    - Hierarchy of memory
    - Dependability via redundancy

- Understanding performance:
    - Algorithms: number of operations to execute
    - Programming Language + Compiler + Architecture: number of instructions per operation.
    - Processor and memory: how fast every instruction executes
    - I/O: how fast are I/O operations execute

Chapter 1 - CPU performance and profiling

- CPU performance
  - decrease - response time(execution time): performance = 1 / execution_time
  - increase - throughput: the amount of work done in a given period of time

- Performance factors:
  - CPU execution time: time CPU spends on a task without including I/O wait time
  - CPU exe time = #CPU clock cycles x clock cycle time
  - CPU exe time = #CPU clock cycles / clock rate

  - performance can increased by:
    - reducing the length of the clock cycle
    - reducing the # of clock cycles for a program

  - CPU Clocking:
    - clock period: duration of a clock cycle
    - clock frequency: cycles per second
    - CR = 1 / CC

  - Clock cycles per instruction(CPI)
    - #CPU clock cycles = # instructions x average clock cycles per instruction

  - Effective CPI
    - $\sum_{i=1}^{n} = (CPI_I \times IC_I)$

  - Performance equation
    - CPU time = IC x CPI x CC
    - CPU time = IC x CPI / CR

  - Three key factors that affect performance:
    - Instruction count
    - Clock Cycles per Instruction
    - Clock cycles or Clock rate

  - Performance depends on:
    - Algorithm: affects IC, possibly CPI
    - Programming language: affects IC, CPI
    - Compiler: affects IC. CPI
    - Instruction set architecture(ISA): IC, CPI, T

  - Power trends:

        - In Complementary metal-oxide-semiconductor(CMOS)
            - power = capacitive load x voltage ^2 x frequency

        - Reducing power
            - the power wall:
                - can't reduce voltage any further
                - can’t remove more heat

    - Profiling tools
        - gprof
        - cachegrind, Dtrace
        - perf

Chapter 2 - Memory Hierarchy: what and why?

- The principles of locality:
    - Temporal locality - time
    - Spatial locality - space

- Memory Hierarchy
    - L1-L4
    - Main memory
    - Secondary memory

- Caches
    - cache hit: info is already in the cache
    - cache miss: info is not in the cache and has to fetched from higher cache
    - victors selection:
        - placement policy: uses mod
        - replacement policy: Least Recently Used (LRU)

    - cache misses:
        - Cold: cache is empty
        - Conflict: blocks all map to the same location, mod
        - Capacity: blocks are larger than the cache

    - Cache concepts:
        - Hit Rate: k/n, where k is successful hits, and k is the number of requests
            - Hit time: time to access block + hit/miss determination
        - Miss Rate: 1 - hit rate
            - Miss penalty: time to access the lower level block + time to transmit that block + time to insert block + time to pass the block

        - Average Memory Access Time (AMAT) = Hit rate + miss rate x miss penalty

    - measuring cache performance
        - CPU = IC x CPI_{stall} x CC
        - memory-stall cycles = # accesses / (instructions x miss rate x miss penalty)

    - New AMAT
        - AMAT: L1 Hit time + L1 Miss Rate * L1 Miss penalty
        - L1 Miss penalty = L2 Hit time + L2 Miss Rate * L2 Miss penalty

Chapter 3 - Memory hierarchy: how?

- Types of cache organization
    - Direct-mapped:
    - Fully associative
    - n-way set associative
-