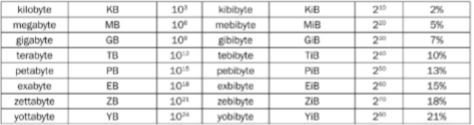
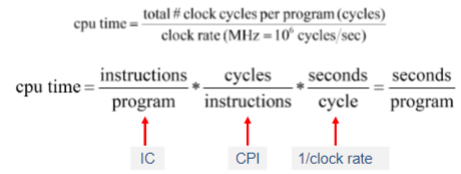
Lecture 1: Intro

* Progress in technology
* Moore’s law
  + Number of transistors on microchip doubles every 1.5, with cost computers halved
* Dennard Scaling
  + As transistors get smaller, power density constant
  + Power use stays in proportion with area
  + Voltage and current scale downward with length
* Types of computers
* Hardware vs software
* HLL vs assembly vs machine language
  + High level language
  + Compiled into machine code by compiler
* Von Neumann Computer Architecture:
  + Processor:
    - Datapath: Performs operations
    - Control: Sequences datapath, memory
  + Memory:
    - Volatile main, loses instructions and data when off
    - Non-volatile secondary: Slower, less expensive, keeps memory when off

Lecture 2: Technology, Performance, Power

* Orders of magnitude (units)
* Response time
  + How long task takes
* Throughput
  + Total work done per unit time
  + eg tasks per hour
* Speedup
  + Timeold / Time new
* Execution Time

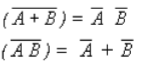
Lecture 3: Technology, Performance, Power II

* Elapsed time
  + Total time including all aspects, determines system performance
* CPU time
  + Time spent processing, discounts I/O, other jobs
  + User CPU + System CPU
  + Clock cycles \* Cycle time
  + = Cycles / Clock rate
* CPU clock cycle
* CPU performance equation
* Amdahl’s Law and overall speedup
  + Law of diminishing returns
* Comparison of performance using clock rate, CPI, IC

Lecture 4: Benchmarking

* Power usage (capacitive load, voltage, frequency)
* Methods
  + real programs : input, output, options (compilers, text processiors)
  + kernels: small, key pieces from real programs
  + toy benchmarks: produce result user already knows, 10-100 lines of code
  + synthetic bencmarks: to match an avg exec profile
  + benchmark suite: collection of benchmarks measuring performance of system with variety of apps, weaknesses lessened by others
* Comparing benchmarks using arithmetic mean
  + Normal average
  + Pro: proportional to overall execution time
  + Cons: can be rigged, cannot be normalized
* normalizaed geometric mean
  + Nth root of product of n factors (average of normalized times)
  + Sae results regardless of re machine
  + Cons: not proportional, large % change in small time contribute to skew

Lecture 5: Combinational Logic

* Logic gates and functions
* Boolean algebra
* Truth tables, logic functions, SOP
* Circuit diagrams

Lecture 6: Adders

* Multiplexers
* Half adder
  + Doesn’t carry-in

Full Adder

* Handles carries
* 3 inputs = x, y, Cin, 2 outputs: S for sum and Cout for carry out

Carry-lookahead adders, using “generate” and “propagate” to calculate carry values

* Generate = A&B > 2, A\*B
* Propagate = Cin > 1 and A+B > 1, A XOR B
* Carries purely a function of A&B, can calculcate all carries without waiting for carry from previos stage

Performance of ripple adder vs carry-lookahead adder

Lecture 7: Sequential Logic

* Arithmetic logic unit
  + +, -, \*, /, %
  + AND, OR, NOT, XOR
* Circuits with a state (“memory”), memory cells, feedback circuits
  + Contains one or more cobinational logic blocks
  + Have state in a feedback loop with logic
  + Use a clock
  + Next state depends on inputs and present state
  + Output depends on present state and perhaps inputs
  + Have memory
* S-R Latch, D Latch, Edge-triggered D Flip-Flop
  + S-R Latch: NOR based, contains basic memory cell, input Set and Reset, outputs Q or Not Q
* Latches vs flip-flops
* Register file

Lecture 8: Counters and finite state machines

* State machine model: Inputs, states, transition, output
* State diagrams and state transition tables
* Circuit design of sate machines (using d flip-flops for state bits)
* Counters

Lecture 8.5: Karnaugh Maps (Extra credit)

* Siplification of sumo of products Boolean function
* Converting truth table to karnaugh map
* Grouping cells with a 1 to form product term
* Simplified function is the sum of product terms for each group