CS2507 Computer Architecture

Dr. Vincent C. Emeakaroha
16-01-2017
vc.emeakaroha@cs.ucc.ie
Room 2.22

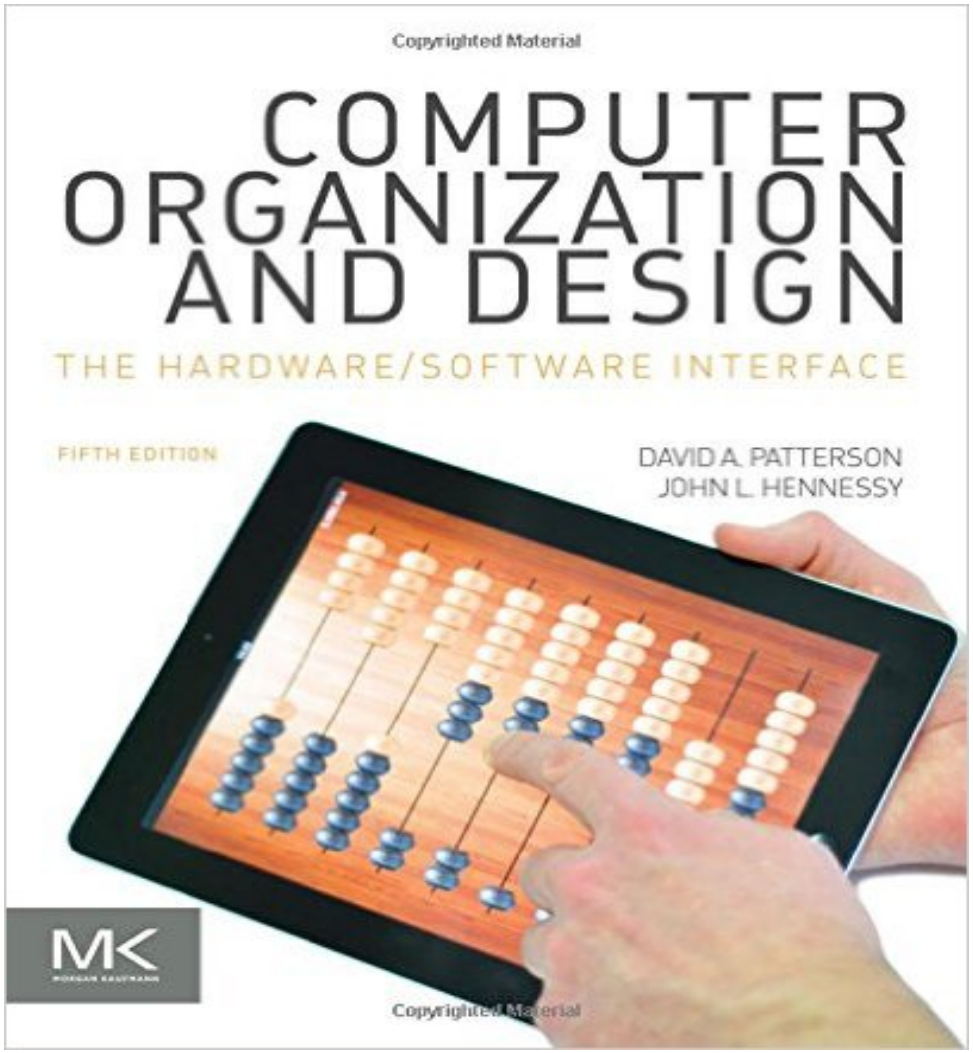
Organisation

- Course webpage
  - Blackboard
- 12 weeks lecturing
  - 24 teaching sessions
- 2 x 1 hr lectures per week
  - Monday 10:00 – 11:00 WGB G02
  - Wednesdays 11:00 -12:00 WGB G03
- 2 hrs lab per week
  - Starting mid February
  - Handled by Thomas Lenihan

Course Evaluation

- Final examination
  - 80 %
- Labs + assignment
  - 20 %
- Pass mark
  - 40 %

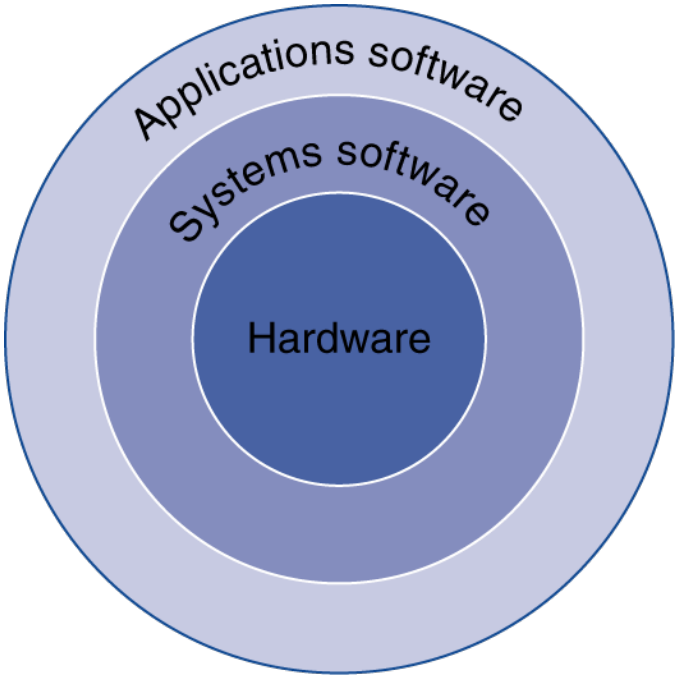
Materials



- Lecture notes
  - Will be posted on Blackboard

Big Picture

Designing for application program performance on modern computers and energy efficiency



- What determines program performance?
- How are programs translated into machine language?
- How do hardware designers improve performance?

Course Overview

- Fundamentals of computer design
- Hardware/software interface
- Program performance and how it can be improved
- Hardware techniques to improve performance
- Hardware techniques to improve energy efficiency
- Instruction set principles
- Pipelining
- Memory hierarchy
- Parallel processors

Computer Revolution

- Progress in computer technology
  - Underpinned by Moore's Law
  - Which states "*integrated circuit resources double every 18-24 months*"
- Makes novel applications feasible
  - Computers in automobiles
  - Cell phones
  - Human genome project
  - World Wide Web
  - Search Engines
- Computers are pervasive

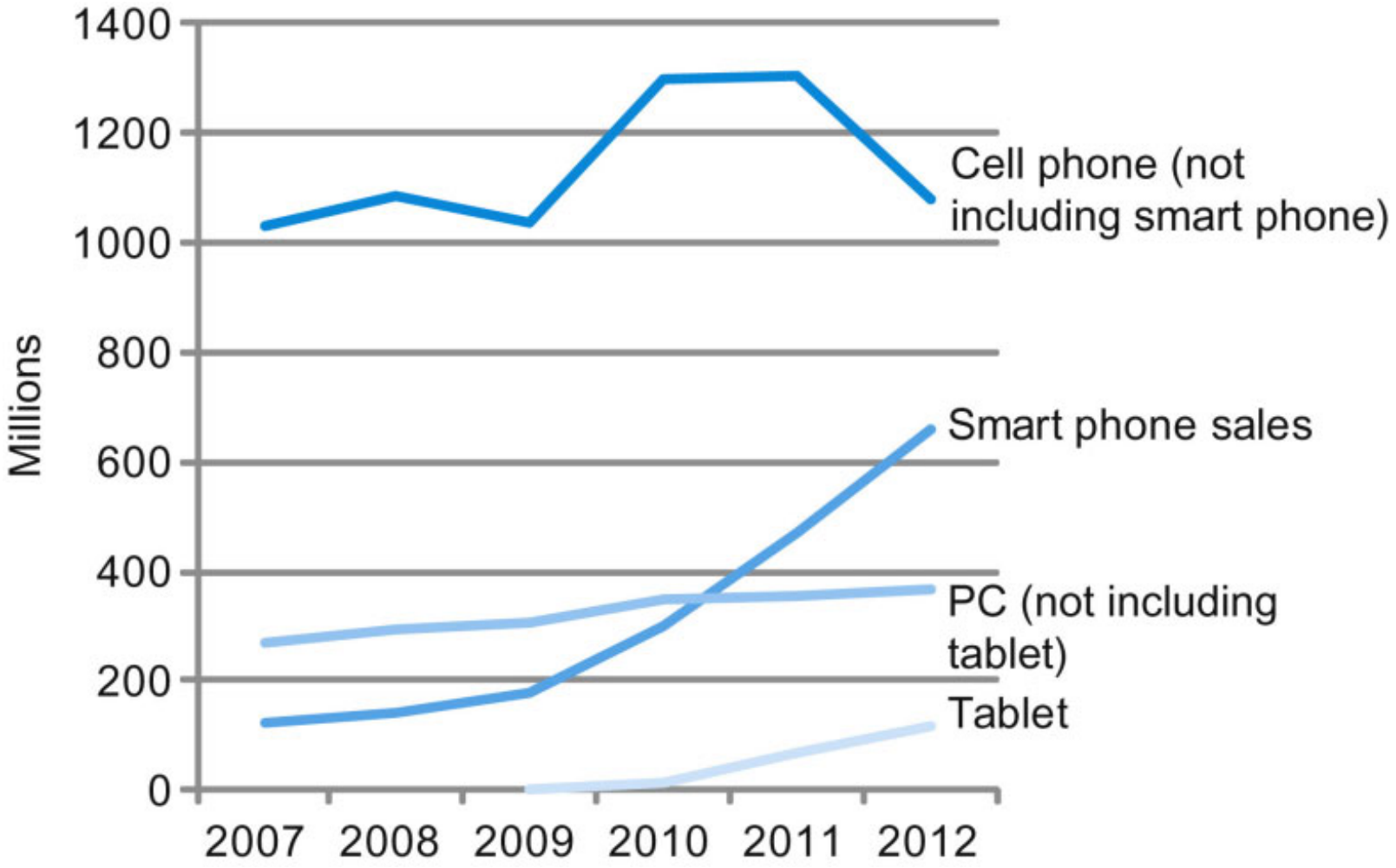
Classes of Computers

- Personal computers (PC)
  - General purpose, variety of software
  - Subject to cost/performance tradeoff
- Server computers
  - Network based
  - High capacity, performance, reliability
  - Range from small servers to building sized

Classes of Computers

- Supercomputers
  - High-end scientific and engineering calculations
  - Highest capability but represent a small fraction of the overall computer market
- Embedded computers
  - Hidden as components of systems
  - Stringent power/performance/cost constraints
  - Most common currently

Device Manufacturing Trend



The PostPC Era

- Personal Mobile Device (PMD)
  - Battery operated
  - Connects to the Internet
  - Hundreds of dollars
  - Smart phones, tablets, electronic glasses
- Cloud computing
  - Warehouse Scale Computers (WSC)
  - Software as a Service (SaaS)
  - Portion of software run on a PMD and a portion run in the Cloud
  - Amazon and Google

Understanding Performance

- Algorithm
  - Determines number of operations executed
- Programming language, compiler, architecture
  - Determine number of machine instructions executed per operation
- I/O system (including OS)
  - Determines how fast I/O operations are executed
- Processor and memory system
  - Determine how fast instructions are executed

Computer Architecture: Eight Great Ideas

1. Design for ***Moore's Law***
   a. Design for rapid change
2. Use ***abstraction*** to simplify design
   a. Representing hardware and software a different levels
3. Make the ***common case fast***
   a. Easier to improve on simple cases than complex ones
4. Performance *via* ***parallelism***
   a. Parallel operations are faster

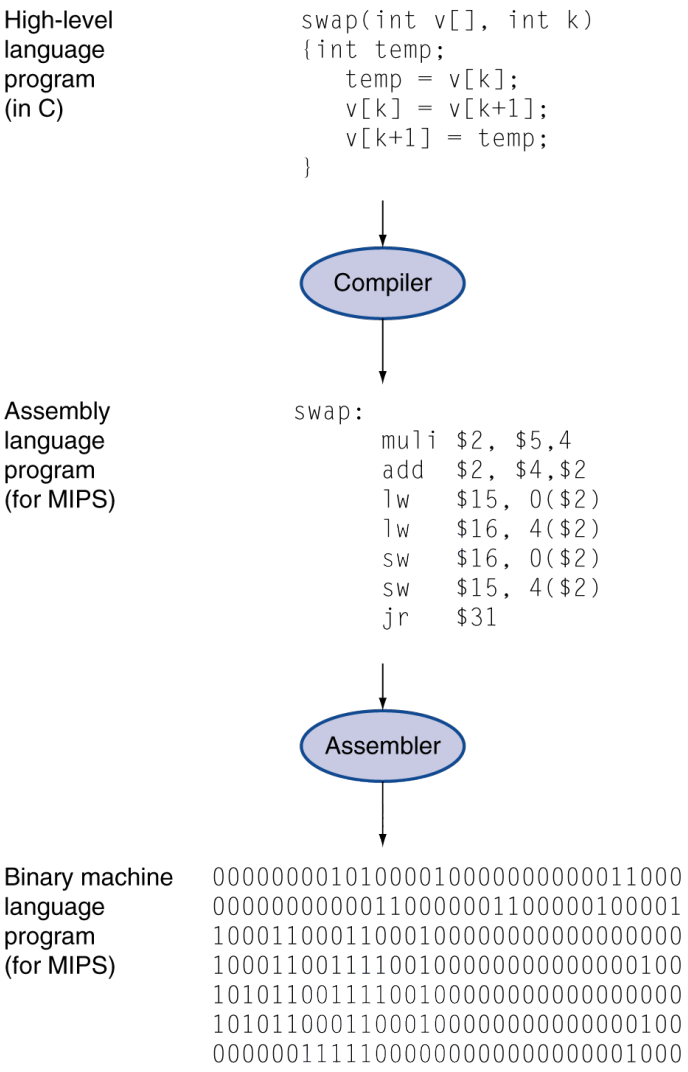
Computer Architecture: Eight Great Ideas

5. Performance *via* ***pipelining***
   a. Sequential pattern of parallelism
6. Performance *via* ***prediction***
   a. Operating based on healthy guess
7. ***Hierarchy*** of memories
   a. Arranging memory according to cost/fastness
8. ***Dependability*** *via* redundancy
   a. Including redundant components for addressing failure

Hardware/Software: Program Translation

- Application software
  - Written in high-level language
- System software
  - Compiler: translates HLL code to machine code (binary)
    - Assembler the first type of translator
  - Operating System: service code
    - Handling input/output
    - Managing memory and storage
    - Scheduling tasks & sharing resources
- Hardware
  - Processor, memory, I/O controllers

Levels of Program Code



- High-level language
  - Level of abstraction closer to problem domain
  - Provides for productivity and portability
- Assembly language
  - Textual representation of instructions
- Hardware representation
  - Binary digits (bits)
  - Encoded instructions and data