CS2507 Computer Architecture

Dr. Ahmed H. Zahran
WGB 182
a.zahran@cs.ucc.ie

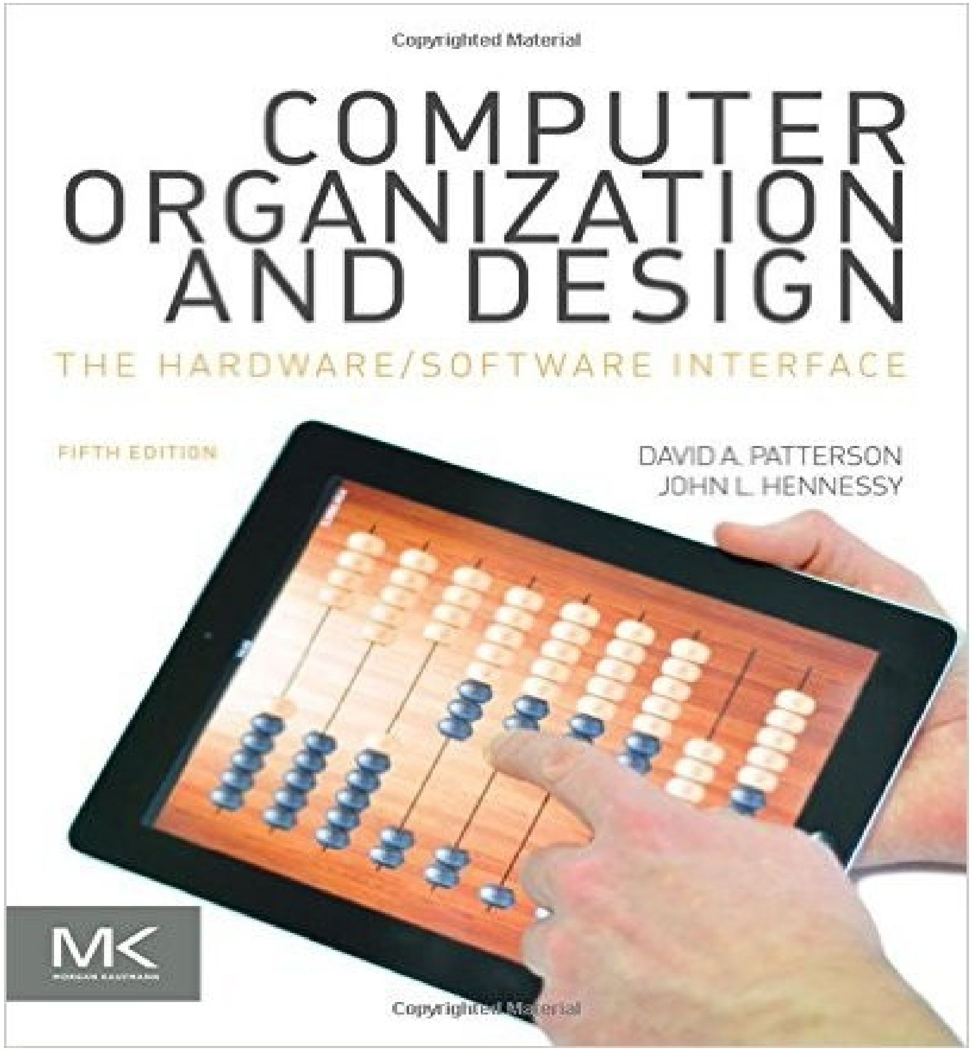
Organization

- Course webpage@Moodle
- Lectures
  - Mondays 10:00 AM -11:00 AM @ WGB G02
  - Wednesdays 11:00 AM – 12:00 PM @ WGB G03
- Office hours
  - 14:30 -15:30 or by appointment
- Labs
  - Wednesdays 9-11 (starting on week 3)
  - 2 hours per week x 5

Course Evaluation

- Final examination
  - 80 %
- Assignments
  - 20 % (Equal weights)
- Pass mark
  - 40 %

Textbook



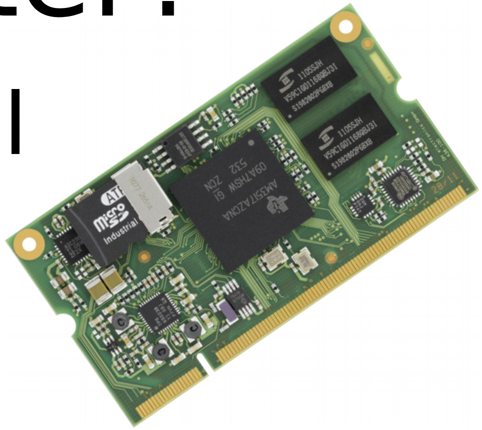
- Lecture slides
  - Will be posted on [Moodle]()
  - Include reading material

Computer Architecture

Objectives
Definitions

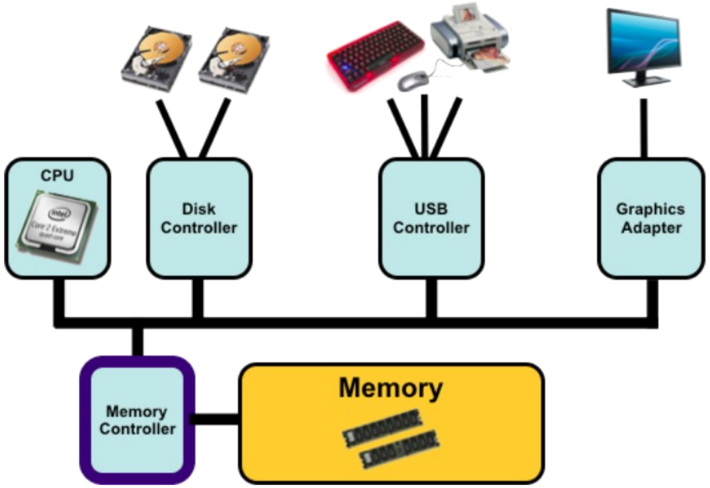
What is a computer?

- Personal computer (general purpose)
- Servers (simple to super computers)
- Embedded computers
- Game consoles
- Personal mobile devices (PMD)
- Cloud (server warehouse)



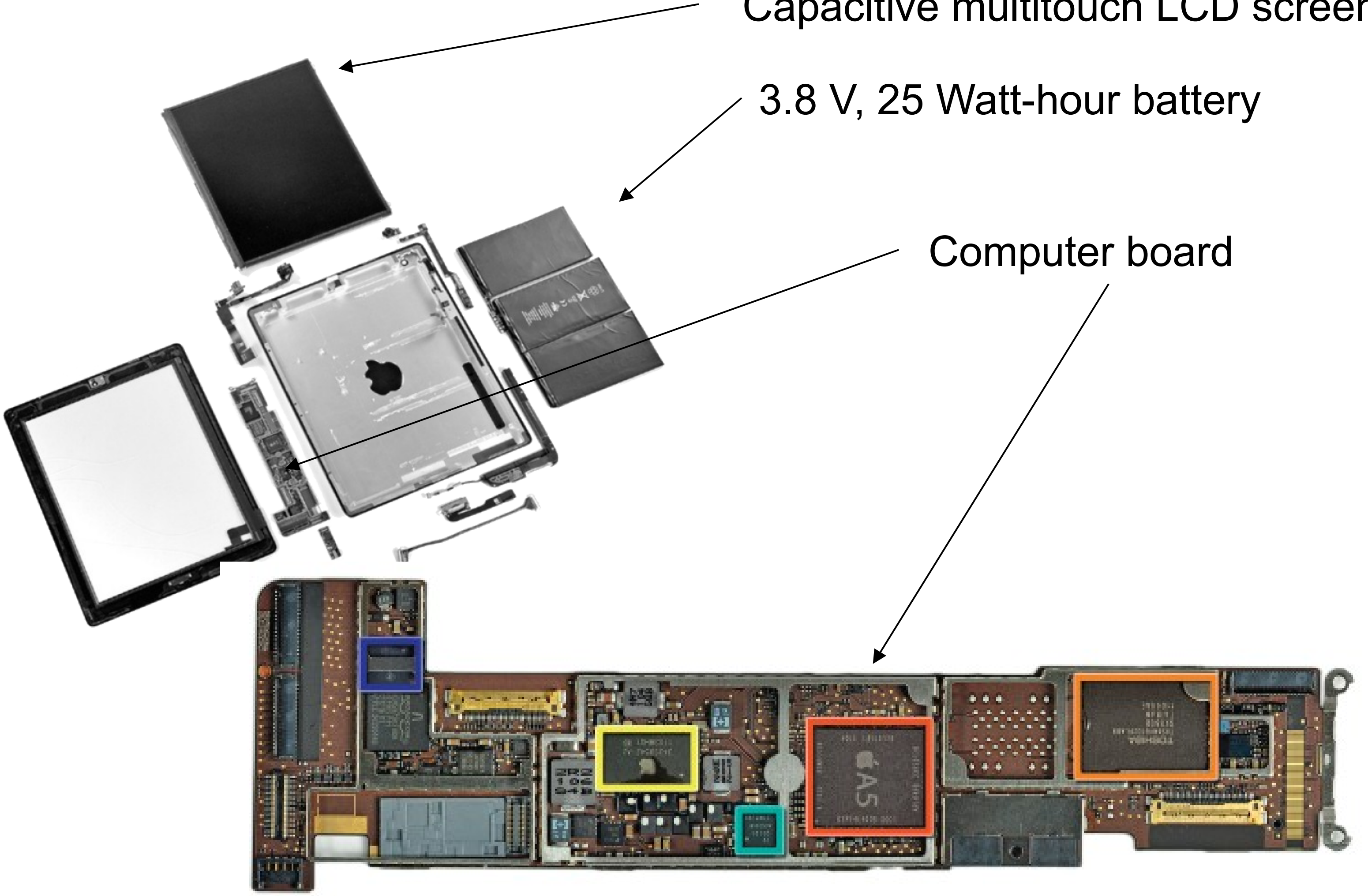
Computer Components

- Same components for all kinds of computer
  - – Inputs/Outputs
  - – Memory/storage
  - – Processor



- These components would have distinct physical and logical implementations

Opening the Box



Capacitive multitouch LCD screen

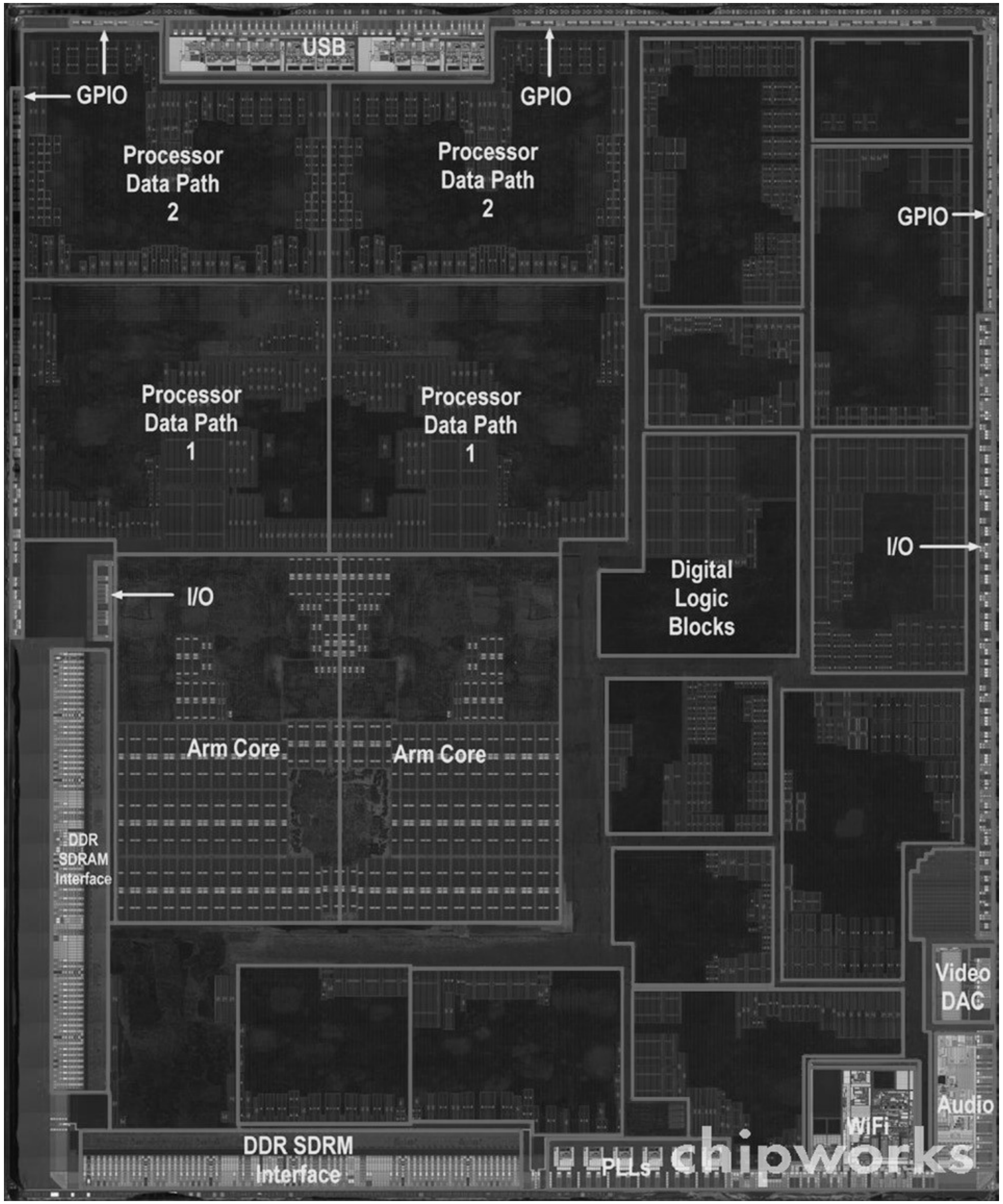
3.8 V, 25 Watt-hour battery

Computer board

Apple iPad 2 tablet

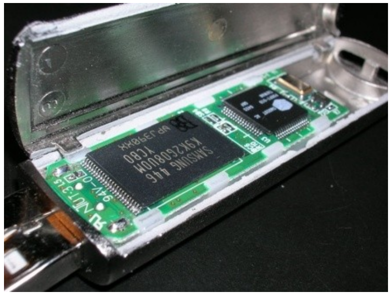
Microprocessor Package

- Apple A5



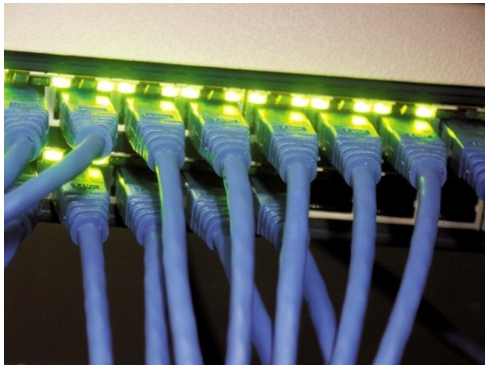
Memory

- Volatile main memory
  - Loses instructions and data when power off
  - RAM and cache
- Non-volatile secondary memory
  - Magnetic disk
  - Flash memory
  - Optical disk (CDROM, DVD)



Networks (I/O example)

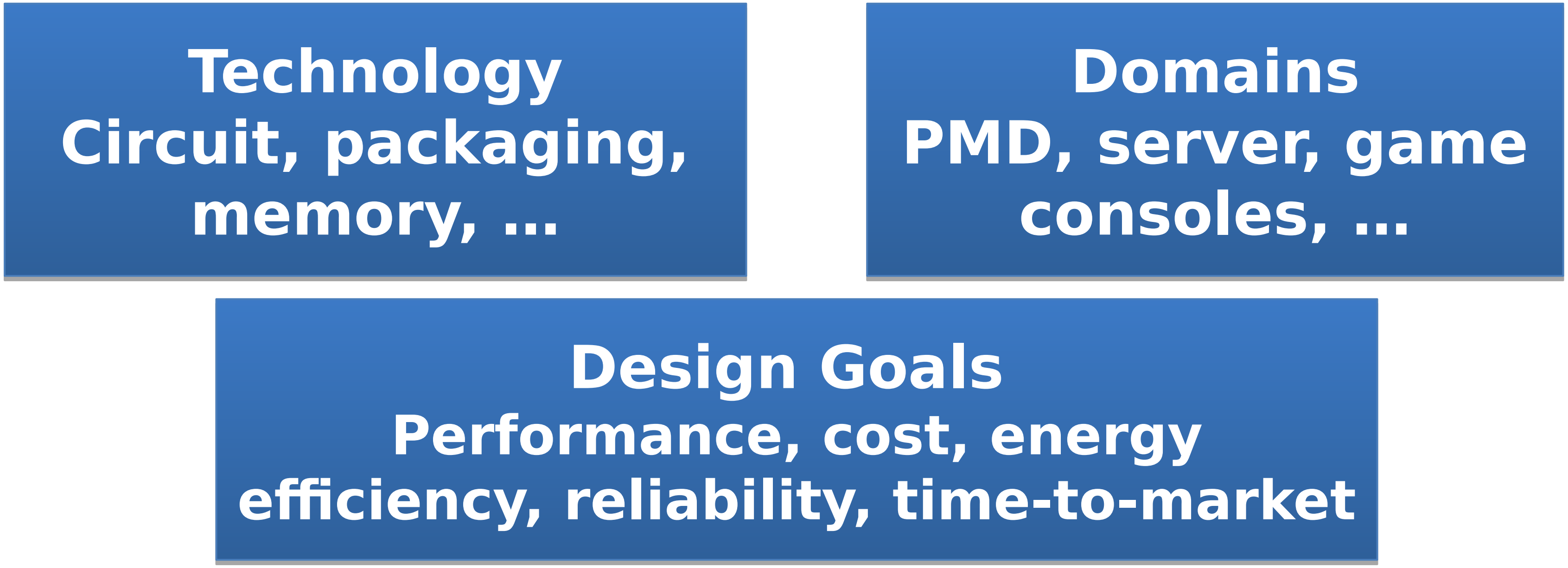
- Communication
  - Ethernet, WiFi, Bluetooth



- Resource sharing (cloud computing, printers, …)
- Nonlocal access (mobile computing)

Computer Architecture

- Computer architecture is the science and art of designing hardware components to create computers that meet functional, performance and cost goals



Technology
Circuit, packaging, memory, …

Domains
PMD, server, game consoles, …

Design Goals
Performance, cost, energy efficiency, reliability, time-to-market

Course Key questions

- How programs, written in a high-level language, such as C or Java, are executed in the computer?
- What determines the performance of a program? How to improve it?
  - Processor and memory design
- How did computer architecture evolve over the years to improve the performance?
- How did that evolution impact software industry?

Eight Great Ideas

Computer Architecture: Eight Great Ideas

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

Computer Architecture: Eight Great Ideas

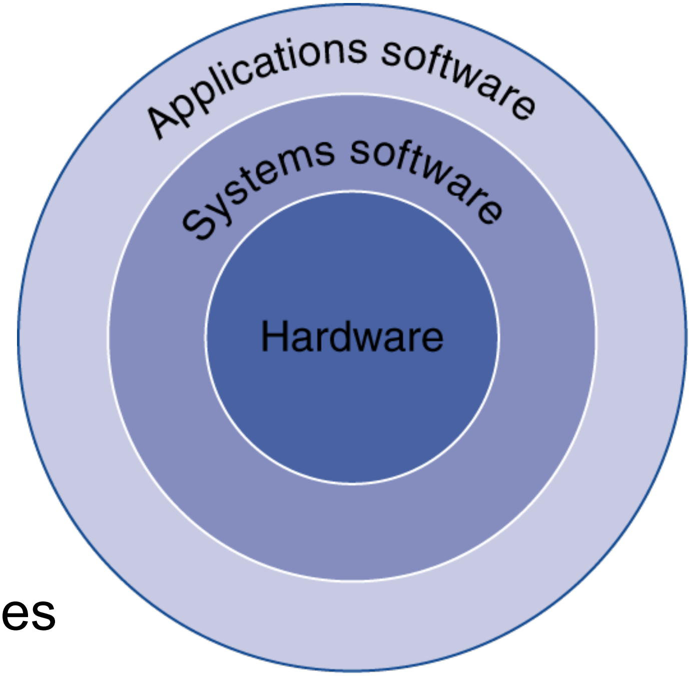
5. *Performance via **parallelism***
   - *Parallel operations are faster*
6. ***Hierarchy*** of memories
   - Arranging memory according to cost/fastness
7. Performance *via* ***prediction***
   - Operating based on healthy guess
8. ***Dependability*** *via* redundancy
   - Including redundant components for addressing failure

Computer Abstraction

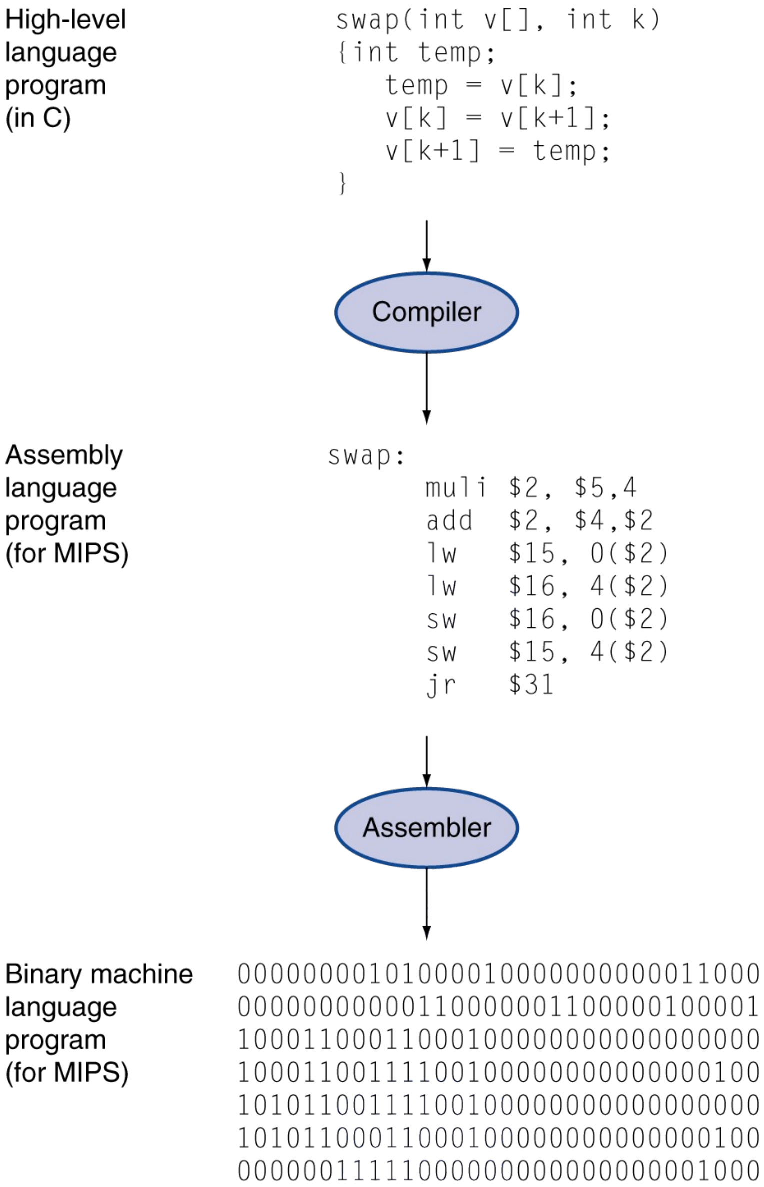
Software
Hardware

Computer Abstraction

- Application software
  - – Written in high-level language
- System software
  - – Operating System: service code
    - Handling input/output
    - Managing memory and storage
    - Scheduling tasks & sharing resources
- Hardware
  - – Processor, memory, I/O controllers



SW abstraction: Levels of Program Code



High-level language program (in C)

```
swap(int v[], int k)
{int temp;
   temp = v[k];
   v[k] = v[k+1];
   v[k+1] = temp;
}
```

Assembly language program (for MIPS)

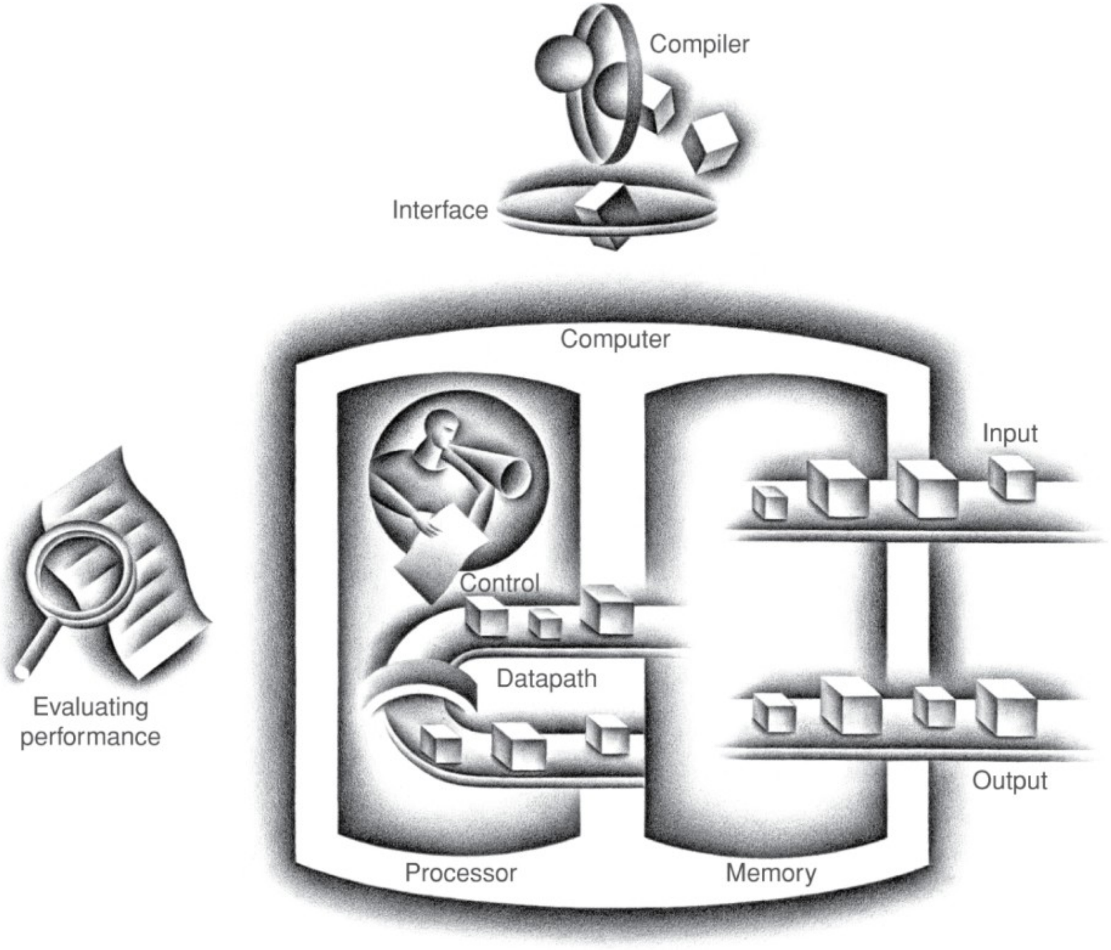
```
swap:
      muli $2, $5,4
      add  $2, $4,$2
      lw   $15, 0($2)
      lw   $16, 4($2)
      sw   $16, 0($2)
      sw   $15, 4($2)
      jr   $31
```

Binary machine language program (for MIPS)

```
00000000101000010000000000011000
00000000000110000001100000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000
```

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

Hardware Operation Overview



Inside the Processor

- Datapath:
  - performs operations on data
- Control:
  - sequences datapath, memory access
- Cache memory
  - Small fast memory for immediate access to data

Instruction Set Architecture

- Both hardware and software consist of hierarchical layers using abstraction
- The instruction set architecture is the key interface between the hardware and low-level software
- This abstract interface enables many implementations of varying cost and performance to run identical soft ware

Summary

- Different computers share a common set of components: processor, memory, and I/O
- Eight design ideas have contributed to the improvement in computer performance over years
- Abstraction is an intrinsic principal in hardware and software design
- The instruction set architecture is the key interface between the hardware and low-level software

Reading

- Section 1.1 – 1.4
- Section 1.5 (optional)