CSCI 4717/5717 Computer Architecture

Topic: Introduction

Reading: Chapter 1

Architecture vs. Organization

- Architecture is the set of attributes visible to the programmer
  - Instruction set, number of bits used for data representation, I/O mechanisms, addressing techniques.
  - Examples:
    - Does this processor have a multiply instr.?
    - How does the compiler create object code?
    - How best is memory handled by the O/S?

Architecture vs. Organization (continued)

- Organization is how features are implemented
  - Control signals, interfaces, memory technology.
  - Examples:
    - Is there a hardware multiply unit or is it done by repeated addition?
    - What type of non-volatile memory is used to store the BIOS?

Architecture vs. Organization (continued)

- All Intel x86 family share the same basic architecture
- The IBM System/370 family share the same basic architecture
- Consistent architecture gives code compatibility, at least backwards, thus protecting user's software investment
- Organization differs between different versions

In-class Exercise

- Assume you are part of a processor manufacturer's marketing group, and you've been asked to generate specifications for a processor that comes in three versions: economy, mid-range, and high-end.
- In groups of three or four, discuss the differences you would have between the three versions of this processor.

Differences in organization but not architecture leads to "families"

- Different cost and performance
- Run same code
- Families may span years of technological advancement

How do CSCI 2150 and CSCI 2160 relate to CSCI 4717?

CSCI 2150/2160	CSCI 4717
•Implementation	•Theoretical
•Bottom-up design	•Top-down design
•Problem solving with:	•Problem solving with:
–bits	–block diagrams
–bytes	–flow diagrams
–code	–performance measures

How do CSCI 2150 and CSCI 2160 relate to CSCI 4717? (continued)

- Understanding digital logic:
  - offers ideas as to how architecture is implemented
  - reveals some of the difficulties encountered when trying to realize an architecture.
- Understanding assembly language:
  - helps explain needs of architecture
  - provides foundation for understanding execution of instructions
  - provides insight to compiler design

In-Class Exercise

In groups of three or four, discuss methods for optimizing a fast food drive thru lane. Be sure to address:

- Menus (both content and presentation)
- Number of steps patrons must go through
- Resources needed for:
  - Ordering
  - Methods of payment
  - Cooking
  - Pickup

Hierarchical Nature of Complex Systems

- Each level of system hierarchy consists of set of components and their interrelationships
  - Operation of components → Function
  - Interrelation of components → Structure
- Each successively higher layer describes simplified/more abstract view of lower levels

Hierarchical Nature of Complex Systems (continued)

- Breaking system into components or modules forces designer to develop a detailed understanding of the data that is passed between them
- Working within the hierarchy, a designer needs to only concern him/herself with the details of his or her module at that specific level
- Working with a well-defined set of inputs, outputs, and function definition, designers can completely design their module without any knowledge of how rest of system is made

Modular System Design

Applying a modular methodology to system design results in:

- a more manageable project
- quicker design time by allowing multiple people with differing expertise to participate (although up-front investment of time feels like a drawback)
- a higher quality system
- a more maintainable system
- increased module reusability

Modular System Design (continued)

There are two methods to use toward a designing a modular system:

- Top down
- Bottom up

Top Down System Design

- Solving a problem by dividing the system into individual functions and building a component to satisfy each function.
- Benefits of Top Down Design
  - Efficient use of components
  - Easier to meet performance goals of the system specification
- Drawbacks of Top Down Design
  - More expensive and time consuming

Bottom Up System Design

- Solving a problem using an existing system (e.g., using DLL's to create a new application)
- Cheaper in small quantities
- Design time is reduced
- Past experiences can be drawn upon

Concept of Black Boxes

- This is the building block of the hierarchical system design.
- If inputs, outputs, and functions are well defined, the designer doesn't need to know about anything above or below in the system

Implementation of components

There are three basic ways to implement a system component

- Hardware (HW)
- Software (SW)
- Firmware (FW)

Hardware

- The permanent, physical implementation of circuits and devices
- Hardware is required for all systems

Software

- The programs contained in read/write memory ranging from machine language to high-level languages
- Requires a processor to run (hardware dependent)

Firmware

- Lies between hardware and software
- Programs (usually machine code) contained in read only memory

Performance Characteristics

- Throughput/speed – HW best; FW average; SW worst
- Development Cost – HW best; FW average; SW worst
- Adaptability – HW worst; FW average; SW best
- Reliability – HW best; FW average; SW average

In-Class Exercise

In groups of three or four, discuss the performance characteristics of hardware, software, and firmware for the following system measures:

- Security
- User interface requirements
- Remote connectivity
- Regulatory standards