CMSC 611

Introduction

Some material adapted from Mohamed Younis, UMBC CMSC 611 Spr 2003 course slides
Some material adapted from David Culler, UC Berkeley CS252, Spr 2002 course slides, © 2002 UC Berkeley
Some material adapted from Hennessy & Patterson / © 2003 Elsevier Science

Overview

- Resources, syllabus, work load
- Grade structure and policy
- Expected background
- An introduction to computer architecture
- Why study computer architecture?
- Organization and anatomy of computers
- Impact of microelectronics technology on computers
- The evolution of the computer industry and generations

Course Resources

- Instructor: Marc Olano / ITE 354
  - Office Hours: Wed 2:00 – 4:00
- TA: Yichuan Gui / ITE 334
  - Office Hours: Tue/Thu 1:00-2:00
- Web Page:
  - **`www.umbc.edu/~olano/611`**
- Book
  - Hennessy and Patterson, *Computer Architecture: A Quantitative Approach*, 3rd or 4th Edition

Syllabus

- Quantitative Design Principles
- Instruction Set Principles
- Pipelining and Instruction Parallelism
- Memory Hierarchy Design
- Storage and I/O
- Multiprocessor Systems
- Interconnection Networks

Workload

- Assignments
  - Approximately 2 hours, every other week
- Exams
  - Midterm in class, Wednesday October 25th
  - Final December 20th, 3:30 – 5:30
- Project

Project

- Teams of three
- You choose application area
  - Best to choose your own research area
- Design architecture for your application
- Final written report / architecture manual

Grades

- Breakdown
  - 30% Homework
  - 25% Midterm
  - 25% Final
  - 20% Project
- Homework late policy
  - Up to 1-week late, -20% of total points
  - >1 week late scores zero

Expected Background

- CMSC 411: Computer Architecture
  - Design of computer systems
    - Information representation
    - Floating point arithmetic
    - Hardwired & micro programmed control
    - Pipelining
    - Cache
    - Bus control & timing
    - I/O mechanisms
    - Parallel processing
- 411 focus on design and implementation (how)
- We focus on design decisions (why)

Introduction & Motivation

- Computer systems are responsible of 5-10% of the gross national product of the US
- WWW, ATM, DNA mapping, … are among the applications that were economically infeasible suddenly became practical
- Even if you don't want to do computer architecture, this class will
  - Help you understand the limits & capabilities of computing
  - Help you understand why
  - Help you understand how to write better code
  - Tools of computer architecture apply everywhere!

Recent Developments

- Multi-core
  - IBM/Sony Cell
  - Intel Core 2, Nehalem
  - Intel Larrabee ☹
- GPUs
- Virtualization
  - vmware: emulate full virtual machine
  - JIT: compile to abstract virtual machine, dynamically compile to host

Slide: David Culler, UCB

More Recent Developments

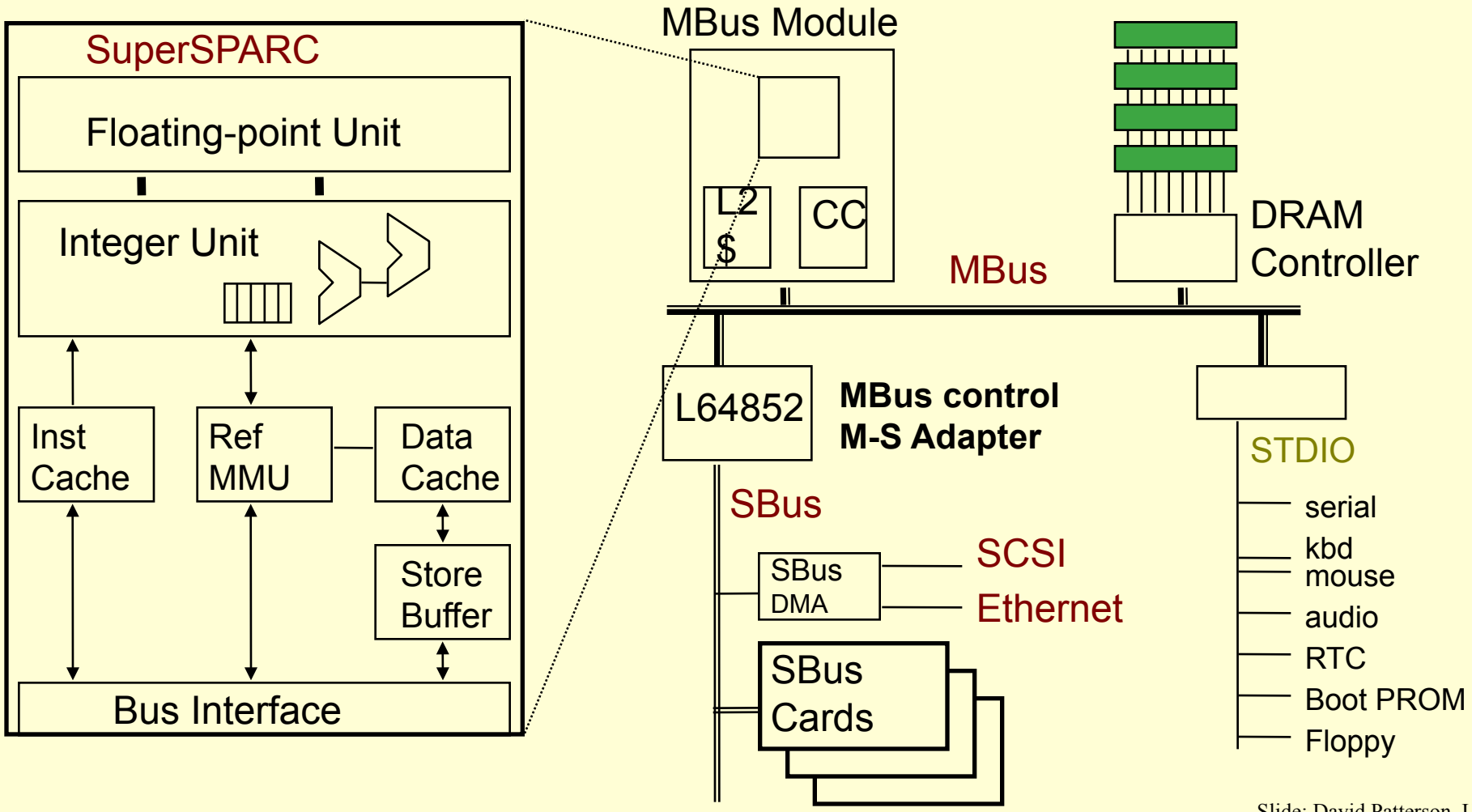
- Parallelism
  - wide issue, dynamic instruction scheduling, EPIC
  - multithreading (SMT)
  - chip multiprocessors
- Communication
  - network processors, network interfaces
- Exotic explorations
  - nanotechnology, quantum computing

What is "Computer Architecture"?

- Instruction set architecture
  - functional behavior of a computer system as viewed by a programmer (like the size of a data type – 32 bits to an integer).
- Computer organization
  - Structural relationships that are not visible to the programmer (like clock frequency or the size of the physical memory).
- The Von Neumann model is the most famous and common computer organization
  - Not the only (e.g. Harvard Architecture)

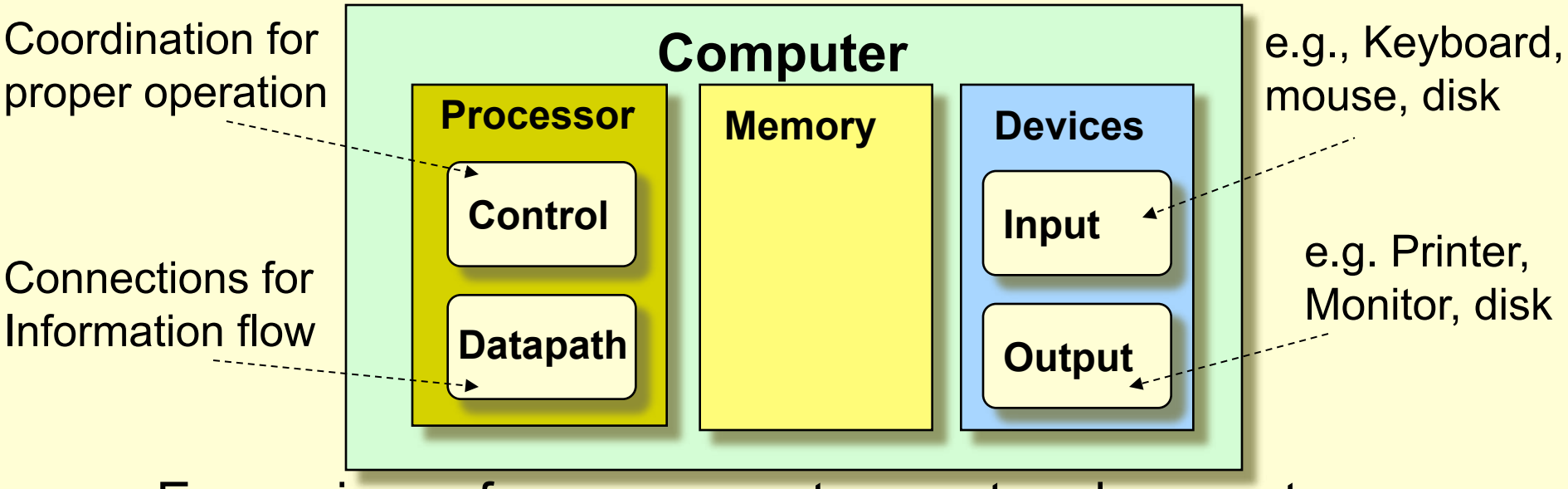
Example Organization

- TI SuperSPARCtm TMS390Z50 in Sun SPARCstation20



Slide: David Patterson, UCB

General Comp Organization



- Every piece of every computer, past and present: input, output, memory, datapath and control
- The design approach is constrained by the cost and size and capabilities required from every component
- An example design target can be 25% of cost on Processor, 25% of cost on minimum memory size, rest on I/O devices, power supplies, and chassis

Levels of Abstraction

- S/W and H/W consists of hierarchical layers of abstraction, each hides details of lower layers from the above layer
- The instruction set arch. abstracts the H/W and S/W interface and allows many implementation of varying cost and performance to run the same S/W

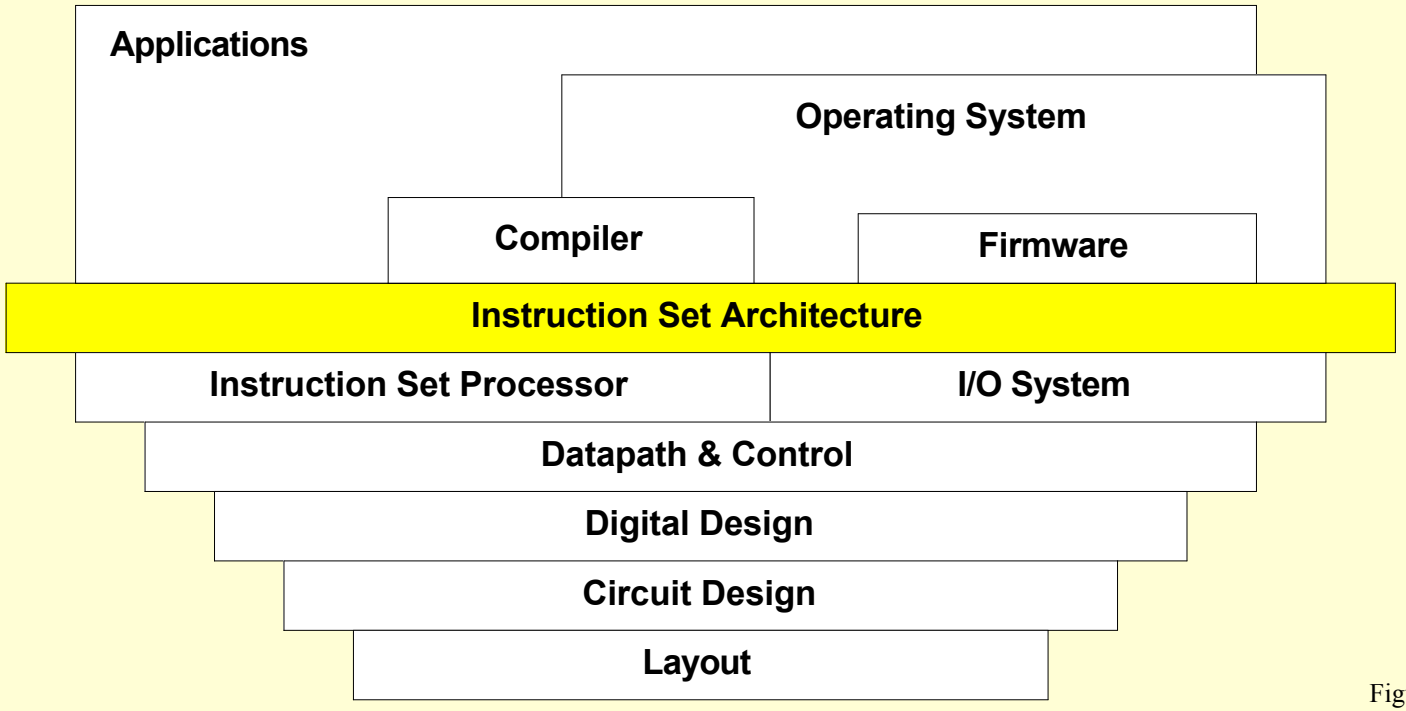


Figure: David Patterson, UCB

Technology – dramatic change

- Processor
  - logic capacity: about 30% increase per year
  - clock rate: about 20% increase per year

Higher logic density gave room for instruction pipeline & cache

- Memory
  - DRAM capacity: about 60% increase per year (4x / 3 years)
  - Memory speed: about 10% increase per year
  - Cost per bit: about 25% improvement per year

Performance optimization no longer implies smaller programs

- Disk
  - Capacity: about 60% increase per year

Computers became lighter and more power efficient