

CS333: Computer Architecture  
University of Virginia Computer Science  
Fall 2006 Michele Co



## Class 10: Big Picture


## Topics/Assignments

- Survey
- Homework 1
- Read "Compilers and Computer Architecture"
  - Principles/factors affecting instruction set design
  - We will discuss this paper next class
    - How it helps us think about M68k and SPARC
- Big Picture

UVa CS333 Fall 2006 - 2

### Where are We? Where are We Going?

Levels of Computer Operation



Application  
High Level Languages  
Assembly language  
Architecture  
Gate-level logic  
Electronics  
Semiconductor  
Real World Physics

UVa CS333 Fall 2006 - 3

### So Far...Mostly Programmer Perspective

- Perspectives: different levels of detail
- Nature of machines and machine languages
- Two real machine examples

UVa CS333 Fall 2006 - 4

### What's Coming Next?

- Architect's View
  - Design Process
  - Functional blocks
    - Single Bus, Multiple Bus
    - Datapath design
    - Control unit
    - Machine reset
    - Exceptions (processor pt of view)
      - Compare to programmer view
- Logic View
  - Implement functional blocks in logic

UVa CS333 Fall 2006 - 5

## Perspectives

## Varying Perspectives

- User
- Assembly language programmer
- Computer architect
- Digital logic designer

## User

- Uses the computer
- No need to understand the internal structure of the machine
  - Just enough knowledge to use

## Assembly Language Programmer

- Model of the machine:
  - Instruction set architecture – ISA
    - Registers, memory, and instructions
    - The stored program **specific to a machine**
    - The fetch-execute cycles
- Programmer uses the ISA to generate programs
  - No knowledge of other architectural features

**Data Movement and transformation**

## Machine data types

- Different than high level language types
  - HLL types: ease of representing objects of interest to the programmer
    - integers, booleans, reals, characters
  - Machine data types: units which can be manipulated by a single machine instruction
    - 8-, 16-, 32-, and/or 64-bit integers and/or reals
    - Compiler writer determines the mapping of HLL type to machine type
  - No type checking
- Compiler writers specify the relationship between HLL and machine language
  - Mapping is *many-to-many*

## Computer Architect

- System design & balance
  - Designs the ISA
    - What is visible to the programmer
  - Designs the hardware for best implementation of the instructions and to meet design goals
  - Balances performance of building blocks such as CPU, memory, I/O devices, and interconnections

## Logic Designer

- Realization of specified function
  - From concept to logic hardware
- Implementation domain
  - Constraints may limit a design suggested by an architect
    - Number and size of registers (chip real estate)

## Communication

- Agreement and understanding needed between programmer, architect, and logic designer
  - To ensure correctness
  - Formal notations: Register transfer languages
    - RTN
    - Used for precision in description

## Machines and Languages

## Machines and Languages

- Machine state
- Machine models
  - Processor-state structure and instruction types
- Instruction sets
  - Addressing modes
- Describing machines and instruction sets
  - Instruction interpretation sequence (fetch-execute)
  - RTN

## States

- Machine state = all register and machine memory
  - programmer-accessible (ISA)
  - non-programmer accessible
- Processor state = all registers internal to CPU
- Memory state = all registers external to the CPU

## Processor State

- Special purpose registers
  - PC, SP, accumulator registers
- General purpose registers

Design spectrum: specialized vs. multi-purpose

## Idealized Machine Models

- Stack (0-address)
  - No register names (implicit top of stack)
- Accumulator (1-address)
  - Limited registers
  - Implied accumulator registers
- General register
  - Range of numbered registers

Models affect the design of the instruction set

## Machine Model and Instruction Set

- Instruction needs to specify needed information
  1. operation
  2. location of operand(s)
  3. location to store result
  4. next instruction
- Implicit specification vs. Explicit specification

Tradeoffs in size of instruction vs. flexibility of operand placement

## Addressing Modes

- Access paths to operands in memory
  - Remember, HLL have complex types, machine language does not
  - Machine needs to provide a way to access machine types in a way that corresponds/maps to HLL types.
    - many-to-many

Addressing modes allow accessing operand in ways that a HLL might structure data or programs

## Register Transfer Languages

- Formal, more precise
- Programmer - understand behavior of machine precisely
  - instructions what
  - addressing modes abstract RTN
  - instruction interpretation
- Logic designer how
  - control signals concrete RTN
  - register transfer implementation details

## Real Machines

## Example Machines

- Performance evaluation
  - metrics: speedup, CPI, MIPS, FLOPS
  - benchmarks: synthetic and real
- Design styles
  - RISC: reduce complexity, increase regularity
  - CISC: more is better
- Examples: M68k, SPARC
  - Still @ programmer perspective...some architect perspective

## M68000

- Programmer should have large range of options available to them
  - 14 addressing modes
    - Some very similar to high level constructs
      - autoincrement, autodecrement
    - Encourage memory traffic
  - Many types of special, irregular instructions

flexibility vs. instruction length

## SPARC

- Programmer can have some basic operations
  - Can compose more complex operations from the basic set
    - Simpler, fixed width insts
    - Load/store
    - Fewer addressing modes

Faster clock frequencies

## What Next?

## What's Coming Next?

- Architect's View
  - Design Process
  - Functional blocks
    - Single Bus, Multiple Bus
    - Datapath design
    - Control unit
    - Machine reset
    - Exceptions (processor pt of view)
      - Compare to programmer view
- Logic View
  - Implement functional blocks in logic