

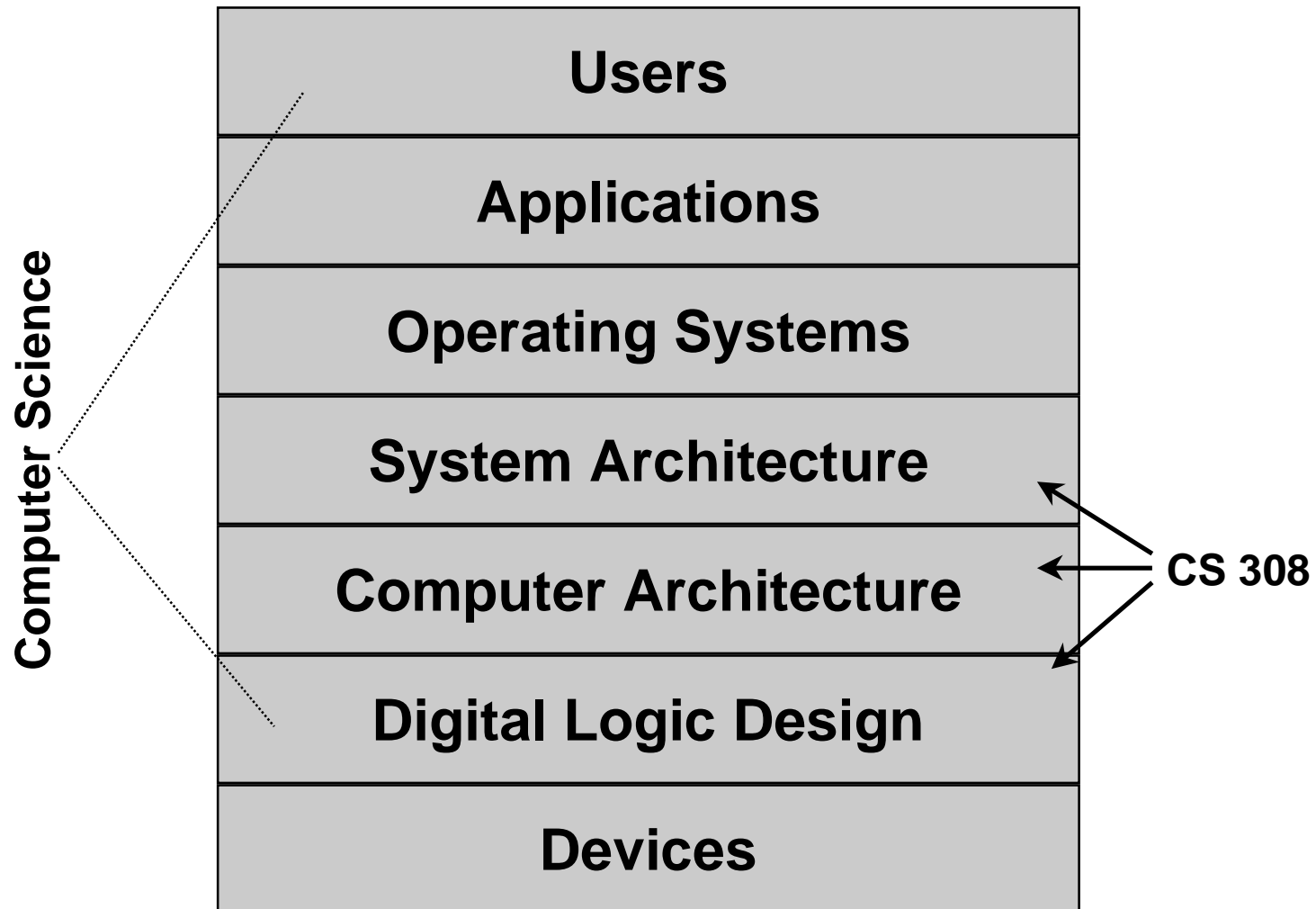
Introduction

- **Alan Turing showed that an abstract computer, a Turing machine, can compute any function that is computable by any means**
- **A general purpose computer with enough memory is equivalent to a Turing machine**
- **Over 50 years, computers have evolved**
 - **from memory size of 1 kiloword (1024 words) clock periods of 1 millisecond (0.001 s)**
 - **to memory size of a terabyte (2^{40} bytes) and clock periods of 1 ns (10^{-9} s)**
- **More speed and capacity is needed for many applications, such as real-time 3D animation**

Historical Generations

- **1st Generation: 1946–59, vacuum tubes, relays, mercury delay lines**
- **2nd generation: 1959–64, discrete transistors and magnetic cores**
- **3rd generation: 1964–75, small- and medium-scale integrated circuits**
- **4th generation: 1975–present, single-chip microcomputer**
- **Integration scale: components per chip**
 - **Small: 10–100**
 - **Medium: 100–1,000**
 - **Large: 1000–10,000**
 - **Very large: greater than 10,000**

Context



Machine/Assembly Language Programmer's View

- **Machine language:**
 - Set of fundamental instructions the machine can execute
 - Expressed as a pattern of 1's and 0's
- **Assembly language:**
 - Alphanumeric equivalent of machine language
 - Mnemonics more human-oriented than 1's and 0's
- **Assembler:**
 - Computer program that transliterates (one-to-one mapping) assembly to machine language
 - Computer's native language is machine/assembly language
 - "Programmer," as used in this course, means machine/assembly language programmer

Machine and Assembly Language

- The assembler converts assembly language to machine language. You must also know how to do this.

MC68000 Assembly Language	Machine Language
MOVE.W D4, D5	0011 101 000 000 100
ADDI.W #9, D2	0000 000 010 111 100 0000 0000 0000 1001

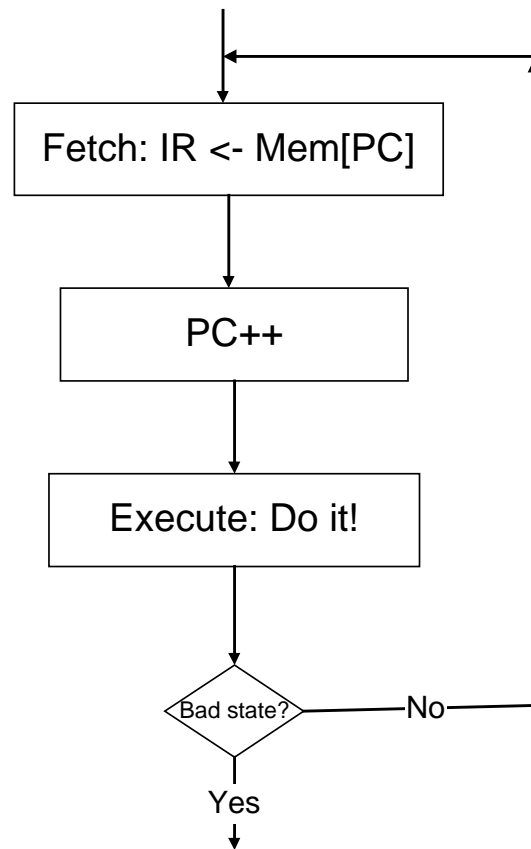
Two Motorola MC68000 Instructions

The Stored Program Concept

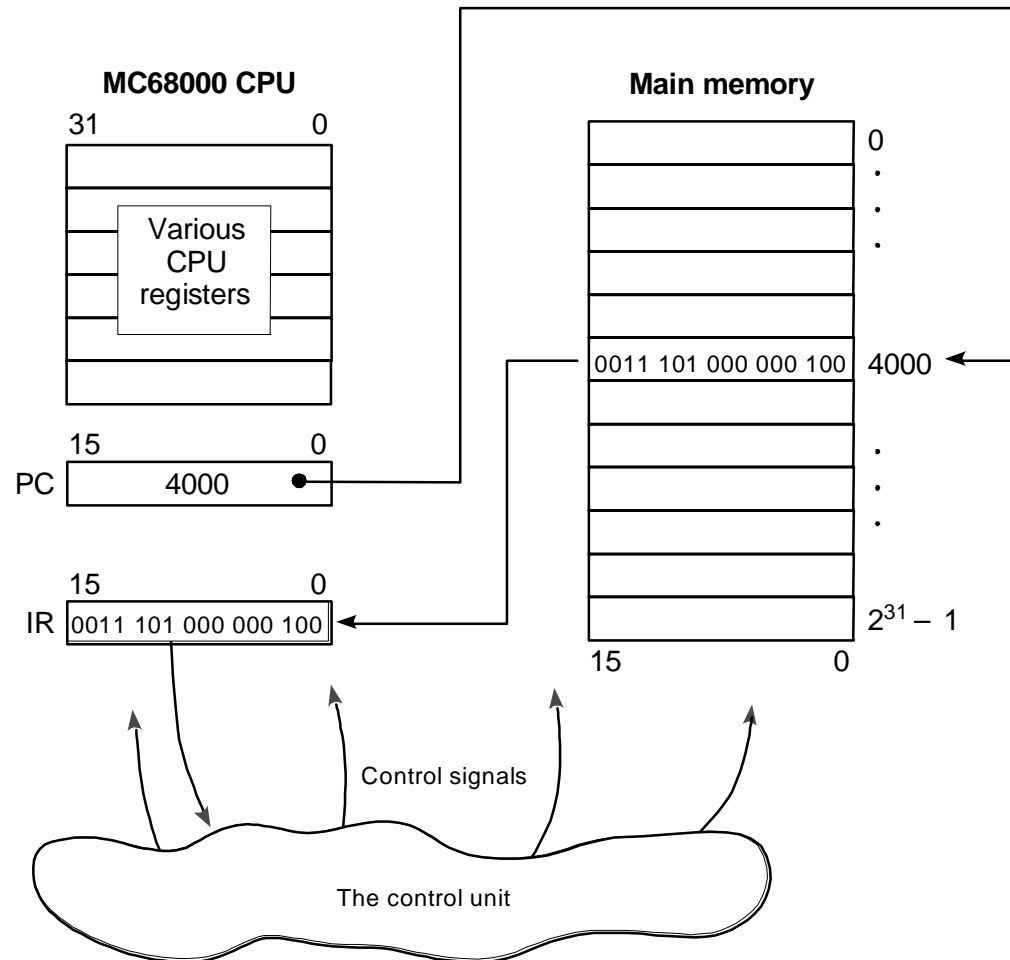
The stored program concept says that the program is stored with data in the computer's memory. The computer is able to manipulate it as data—for example, to load it from disk, move it in memory, and store it back on disk.

- It is the basic operating principle for every computer.
- It is so common that it is taken for granted.
- Without it, every instruction would have to be initiated manually.

The Fetch-Execute Cycle



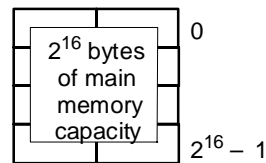
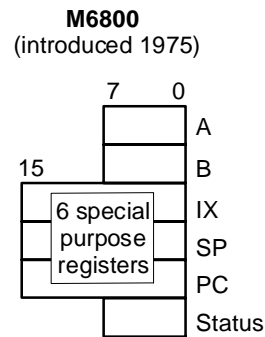
The Fetch-Execute Process



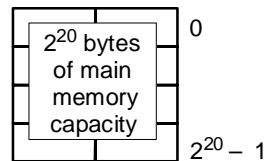
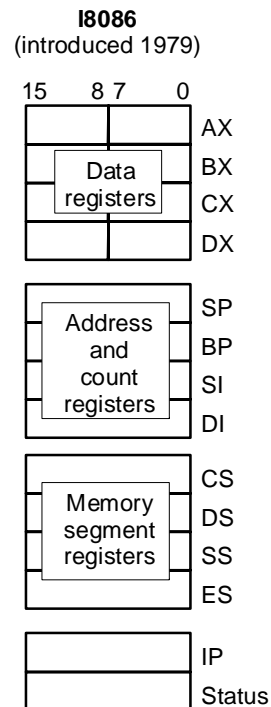
Programmer's Model: Instruction Set Architecture (ISA)

- **Instruction set: the collection of all machine operations.**
- **Programmer sees set of instructions, along with the machine resources manipulated by them.**
- **ISA includes**
 - **Instruction set,**
 - **Memory, and**
 - **Programmer-accessible registers of the system.**
- **There may be temporary or scratch-pad memory used to implement some function is not part of ISA.**
 - **Not Programmer Accessible.**

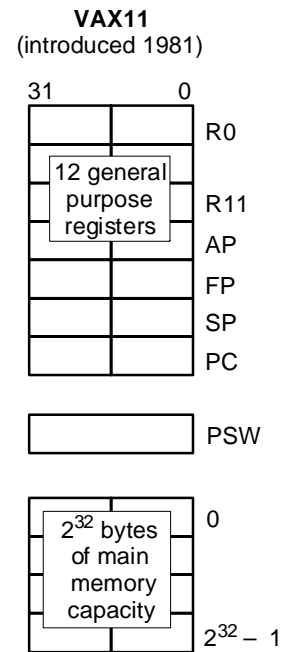
Fig 1.3 Programmer's Models of 4 Commercial Machines



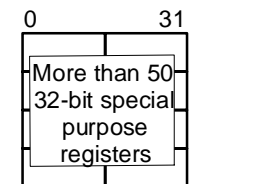
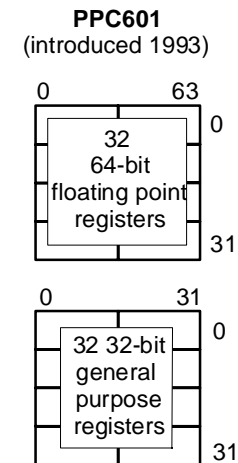
Fewer than 100 instructions



More than 120 instructions



More than 300 instructions



More than 250 instructions

Machine, Processor, and Memory State

- **The Machine State:** contents of all registers in system, accessible to programmer or not
- **The Processor State:** registers internal to the CPU
- **The Memory State:** contents of registers in the memory system
- **“State”** is used in the formal finite state machine sense
- **Maintaining or restoring the machine and processor state is important to many operations, especially procedure calls and interrupts**

Data Type: HLL Versus Machine Language

- **HLLs provide type checking**
 - **Verifies proper use of variables at compile time**
 - **Allows compiler to determine memory requirements**
 - **Helps detect bad programming practices**
- **Most machines have no type checking**
 - **The machine sees only strings of bits**
 - **Instructions interpret the strings as a type: usually limited to signed or unsigned integers and FP numbers**
 - **A given 32-bit word might be an instruction, an integer, a FP number, or 4 ASCII characters**

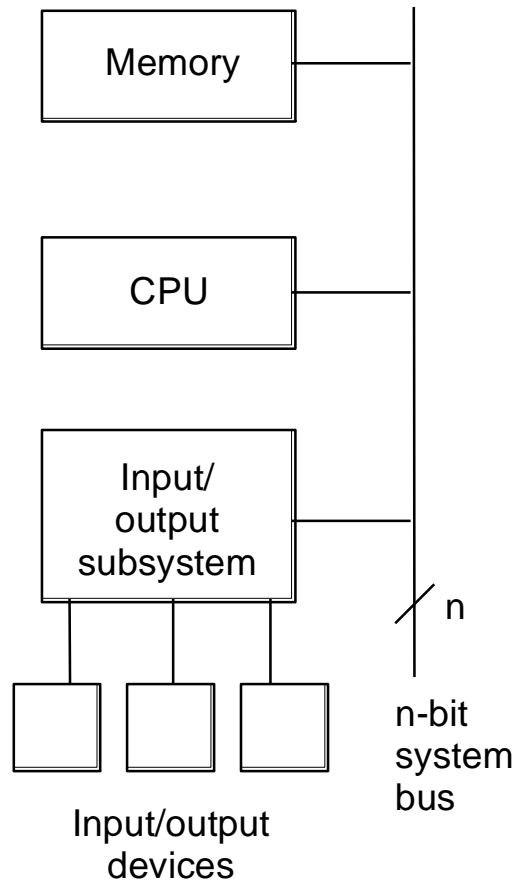
Why Use Assembly Language?

- **The machine designer**
 - **Must implement and trade off instruction functionality**
- **The compiler writer**
 - **Must generate machine language from a HLL**
- **The writer of time or space critical code**
 - **Performance goals may force program-specific optimizations of the assembly language**
- **Special purpose or imbedded processor programmers**
 - **Special functions and heavy dependence on unique I/O devices can make HLLs useless**

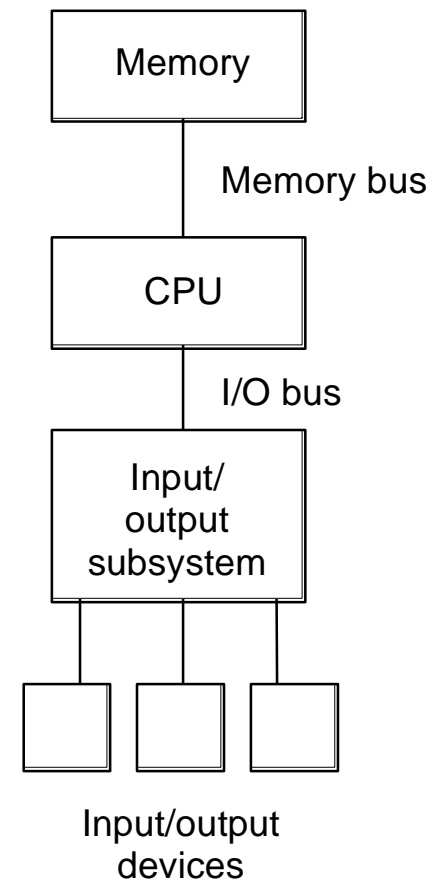
Buses as Multiplexers

- **Interconnections are very important to computer**
- **Most connections are shared**
- **A bus is a time-shared connection or multiplexer**
- **A bus provides a data path and control**
- **Buses may be serial, parallel, or a combination**
 - **Serial buses transmit one bit at a time**
 - **Parallel buses transmit many bits simultaneously on many wires**

Simple One- and Two-Bus Architectures



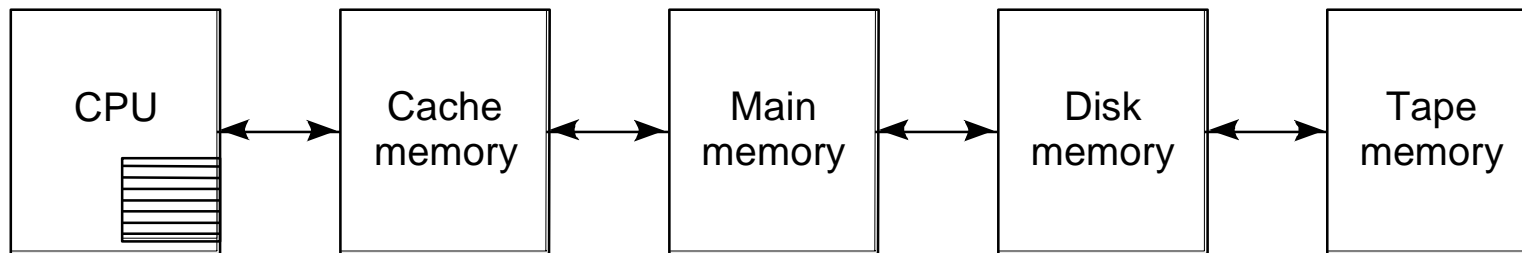
(a) One bus



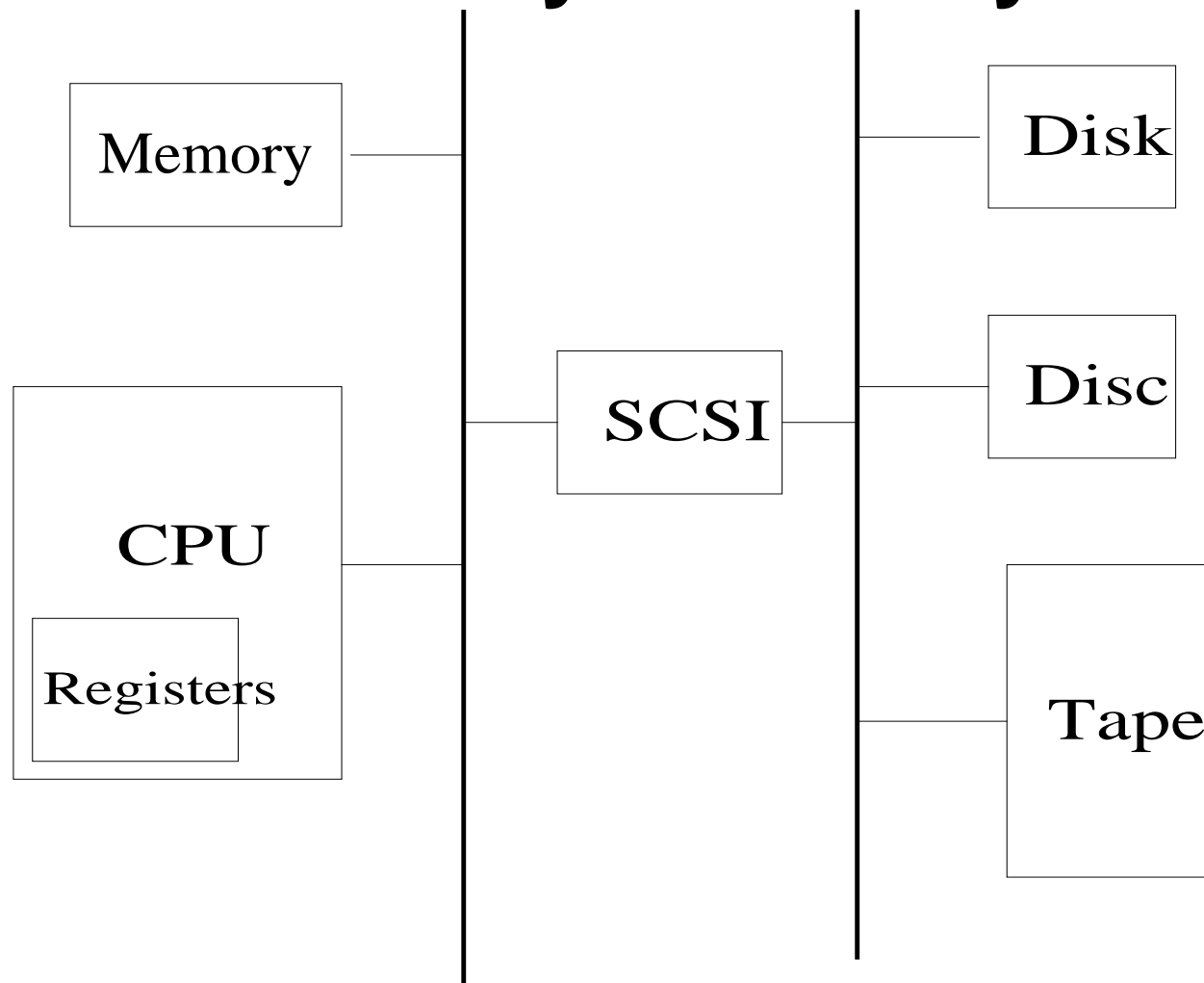
(b) Two buses

The Memory Hierarchy

- **Modern computers have a hierarchy of memories**
 - **Allows tradeoffs of speed/cost/volatility/size, etc.**
- **CPU sees common view of levels of the hierarchy.**



More Accurate Picture of Memory Hierarchy



The Distinction Between Classical Logic Design and Computer Logic Design

- **The entire computer is too complex for traditional FSM design techniques**
 - **FSM techniques can be used “in the small”**
- **There is a natural separation between data and control**
 - **Data path: storage cells, arithmetic, and their connections**
 - **Control path: logic that manages data path information flow**
- **Well defined logic blocks are used repeatedly**
 - **Multiplexers, decoders, adders, etc.**

Summary of Chapter 1

- **Three different views of machine structure and function**
- **Machine/assembly language view: registers, memory cells, instructions**
 - **PC, IR**
 - **Fetch-execute cycle**
 - **Programs can be manipulated as data**
 - **No, or almost no, data typing at machine level**
- **Architect views the entire system**
 - **Concerned with price/performance, system balance**
- **Logic designer sees system as collection of functional logic blocks**
 - **Must consider implementation domain**
 - **Tradeoffs: speed, power, gate fan-in, fan-out**