Microprogramming

CS 333

Fall 2006

Announcements

- Reading Assignment –
  - Read Chapter 5, Section 5.4 – end of chapter
- Lab 2 – Postlab
- Homework clarifications – 4.11, 4.12, 4.19

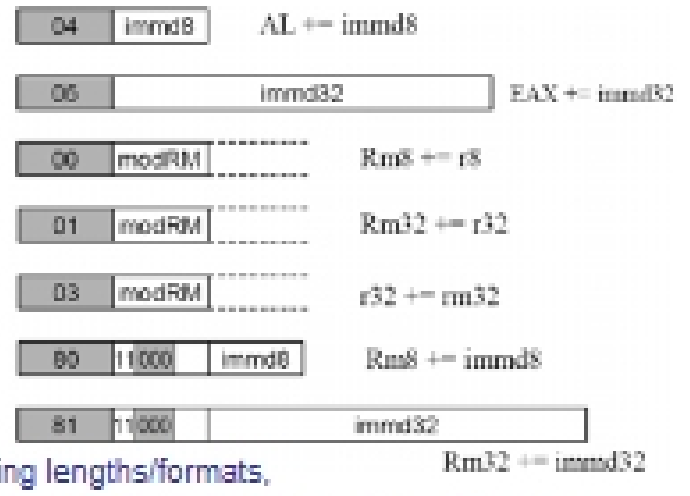
Microprogramming Motivation

- Control functions
  - Complex
    - Manageable for smaller, more regular instruction sets, but consider:
      - IA-32

IA-32

- Intel Architecture 32-bit
  - CISC-like instruction set architecture
    - Several hundred instructions of varying classes
    - Varying length instructions
    - Many addressing modes
  - First implementation 80386
  - Pentium, … Pentium 4, …AMD K7

Example IA-32 add Instruction Formats



Varying lengths/formats, makes control unit design more complicated

Microprogramming!

- Thousands of states
  - Hundreds of different control sequences
- Control signals can be thought of as if they are instructions executed by the datapath
- Simplifies control design

Microprogramming Process

1. Define microinstruction format
2. Create microprogram
3. Implement the microprogram

Defining the Microinstruction Format

Microinstructions

- To avoid confusion with ISA (instruction set architecture)
  - **microinstructions**
    - Defines set of datapath control signals to assert in a given state
    - Executing a microinstruction
      - Asserts control signals specified in microinstruction

Microprogram

- Algorithmic description for how to execute instructions (from ISA) in a program by asserting a sequence of control signals
- Each instruction in ISA has a list of microinstructions
  - Ex., control sequences in concrete RTN for SRC

Sequencing

- The **next microinstruction to be executed** needs to be specified
  - Analogy
    - In programs we use functions/methods to reuse commonly executed sequences of instructions (control flow/subroutines)
    - Microprograms are similar
      - Several instructions in the ISA may have similar control sequences (these can be reused instead of recoded)
      - Example, every instruction at least needs to be fetched before execution

Microprogramming

- Designing the control symbolically
  - Analogy:
    - Assembly language is a symbolic representation of machine instructions
      - Fields: op code, registers, offsets, immediate field, etc.
    - Microprogram is a symbolic representation of microinstructions
      - Also has fields. Many different arrangements
      - Microcoded control units are used to implement complex microprograms

Microinstruction Format

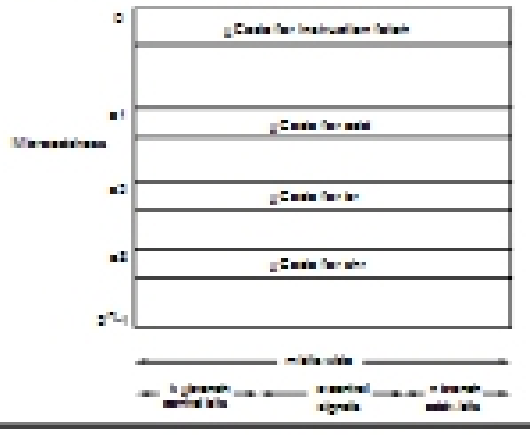
- Need to choose:
  - Number of fields
  - What control signals are specified by each field
- Would like:
  - Readability:
    - Format needs to be simple enough to write and understand the microprogram
  - Difficult to write inconsistent microinstructions
    - Don't want a particular control signal to be specified as two different values in one microinstruction
    - Signals never asserted simultaneously can share the same field

More Parallels with Assembly Programs

- Microinstruction fields may allow combinations that can't be supported by the datapath
  - microassembler – checks/flags these errors

Control Store

- Often a PLA (programmable logic array) or ROM (read-only memory)
  - Each entry has an address



Ways of Sequencing

1. **Increment address** of current microinstruction
   - like sequential execution of instructions
   - often is default
2. Branch to the next microinstruction that executes the next instruction (ISA)
   - Branch to the instruction fetch microinstruction sequence
3. **Dispatch** – Lookup table (PLA)

Creating a Microprogram

Example Microinstruction Fields

- ALU control
  - Add – cause ALU to add
  - Subtract – cause ALU to subtract
- SRC1 – first source register to ALU
  - PC
  - Register A – first ALU input
- SRC2 – second operand
  - Register B, second ALU input
  - 4- for PC + 4
  - Extend (sign extend)