i206: Lecture 8:
HW and Test Review

Tapan Parikh

Spring 2013

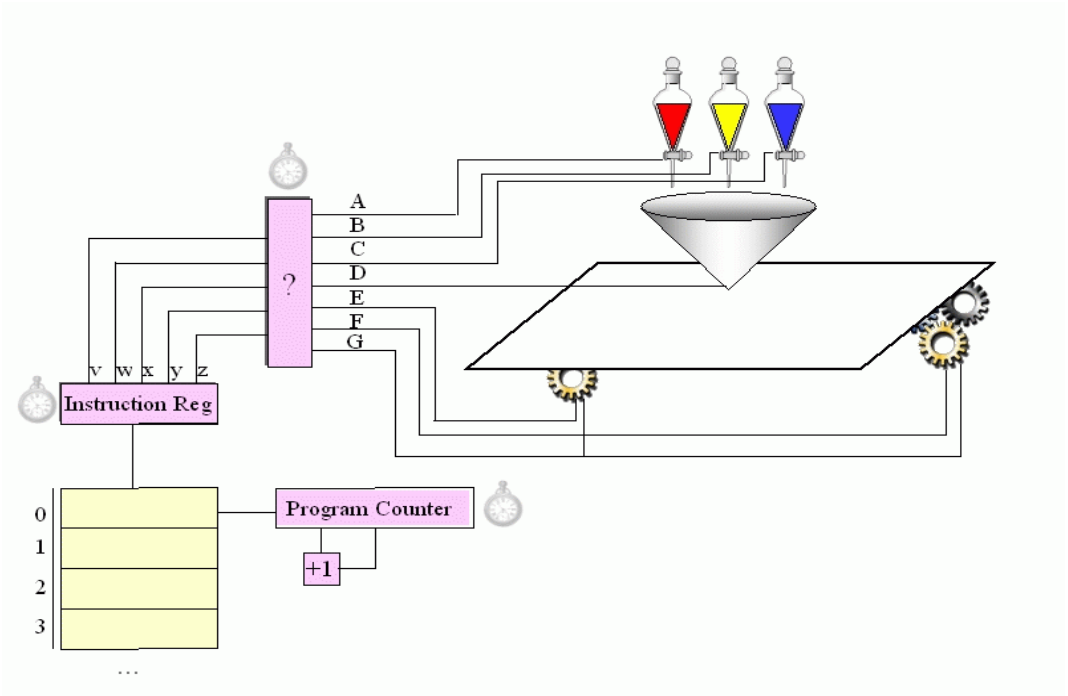
Some slides courtesy Marti Hearst, John Chuang and others

The Paint Machine

Things to learn from this assignment

- Multiple representations for the same thing
  - Truth tables & circuits
  - Boolean logic & controlling machines
  - Opcode & operand is like a method & its arguments
- How to use binary numbers to make codes
  - Important in security and networking
- Designing for modularity
- Applying the principled of abstraction & simplicity

The Paint Machine



- E controls the forwards-backwards movement
  - When E=1, it moves the paper forwards or backwards one pixel
- F controls the left-right movement of the paper.
  - When F=1, it moves the paper one pixel left or right.
- G indicates the direction for the roller.
  - If G=1 and E=1, go forwards. If G=0 and E=1, go backwards.
  - If G=1 and F=1, go left. If G = 0 and F = 1, go right.

Strategy for Designing the Instruction Set

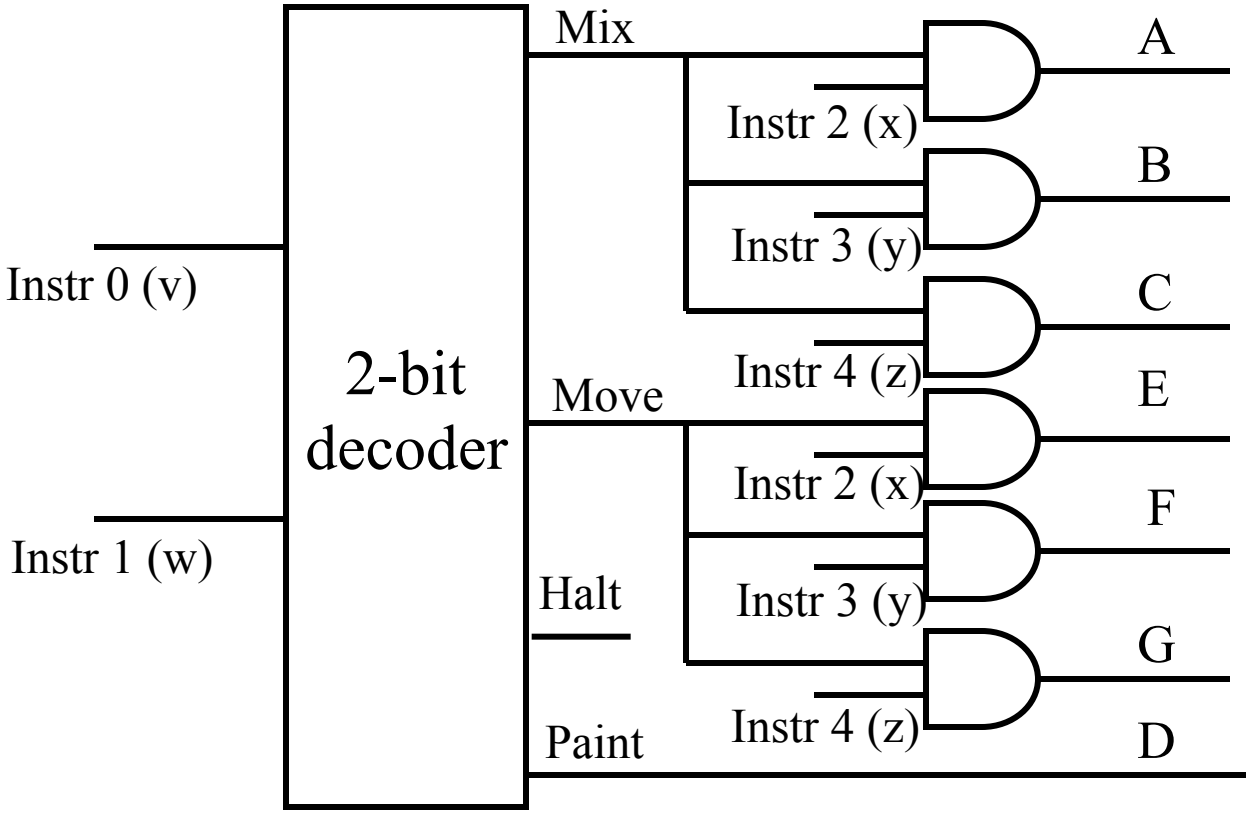
- Use the principle of abstraction to break the problem into subproblems
  - The machine does four operations
  - Each operation has different kinds of arguments
  - Make the instructions reflect this
    - Divide into opcodes and operands
    - Opcodes are the commands saying what to do
    - Operands are the arguments saying how to do it
- Use the principle of simplicity to design the operand
  - Make the bits of the operand mirror the functions of the machine

Designing the Circuit

- Use the principle of abstraction
  - Use the 2-bit decoder to convert the opcode into a control for the machine
- Use the principle of simplicity
  - Make the operands mirror the structure of the machine's controls

			A	B	C	D	E	F	G
mix red	11	100	1	0	0	0	0	0	0
mix yellow	11	010	0	1	0	0	0	0	0
mix blue	11	001	0	0	1	0	0	0	0
mix green	11	011	0	1	1	0	0	0	0
mix purple	11	101	1	0	1	0	0	0	0
mix orange	11	110	1	1	0	0	0	0	0
mix black	11	111	1	1	1	0	0	0	0
mix white	11	000	0	0	0	0	0	0	0
			A	B	C	D	E	F	G
move left	10	011	0	0	0	0	0	1	1
move right	10	010	0	0	0	0	0	1	0
move forw	10	101	0	0	0	0	1	0	1
move back	10	100	0	0	0	0	1	0	0
paint	01	000	0	0	0	1	0	0	0
halt	00	000	0	0	0	0	0	0	0

The Circuit – One Solution



Assumes that the compiler cannot produce an illegal instruction.

For example, 10111 is not legal but if produced would jam the machine.

Bonus Questions

- 1. Unnecessary instruction?
  - Mix white
- 2. Bad program?
  - Many possibilities, including forgetting a “paint”
    - Mix orange
    - Mix purple
    - Paint
- 3. Make paper move >1 step at a time?
  - Make direction part of op-code (needs 6 bits total)
    - Left 3
    - Forward 7
  - Or use the extra bits not used by halt and paint
    - Messy solution
- 4. Reduce instruction time by one clock cycle?
  - Increment PC while the circuit is executing
  - Or don’t disconnect circuit from machine

Exam Review

Boolean

- Be able to convert between Boolean expressions and logic gates and truth tables and Venn diagrams.

- Be able to use deMorgan's laws to convert Boolean expressions.

Binary

- Be able to
  - Do the basics of binary arithmetic
  - Convert among binary, decimal, and hexadecimal

- Understand the relationship between binary numbers and truth tables.

- Understand the relationship between binary numbers and powers of 2.

- Understand how information that isn’t numerical can nonetheless be represented by binary numbers.

Machine Instruction Sets and Assembly Language

- Understand the relationship between computer instructions and binary numbers.
- Understand the relationship between machine instructions and assembly language instructions and be able to convert between them.
- Understand the meaning of a sequence of instructions and describe what that sequence of instructions does.

CPU, Circuits and Memory

- Understand the basic functioning of a CPU.
- Understand the roles and behavior of the major circuits that comprise the CPU, including:
  - Registers
  - ALU
  - RAM
- Understand the difference between operating on a numerical value vs. a memory reference (address).

Math

- Be able express sums (and differences) of sequences of numbers in sigma notation and with python code.

- Understand the graphed shapes of and relative size differences between different kinds of functions
  - E.g., linear, polynomial, exponential, log

Analysis of Algorithms

- Be able to analyze pseudocode or python code to determine the number of steps required to execute that code in the general case (for any value of n)
- Be able to analyze algorithms in terms of their big-O running time.

Sorting Algorithms

- Understand the basics of how the main ones work.
- Know the running times of the main ones.