i206 Spring 2013: Homework 3

The Paint Machine

Goals

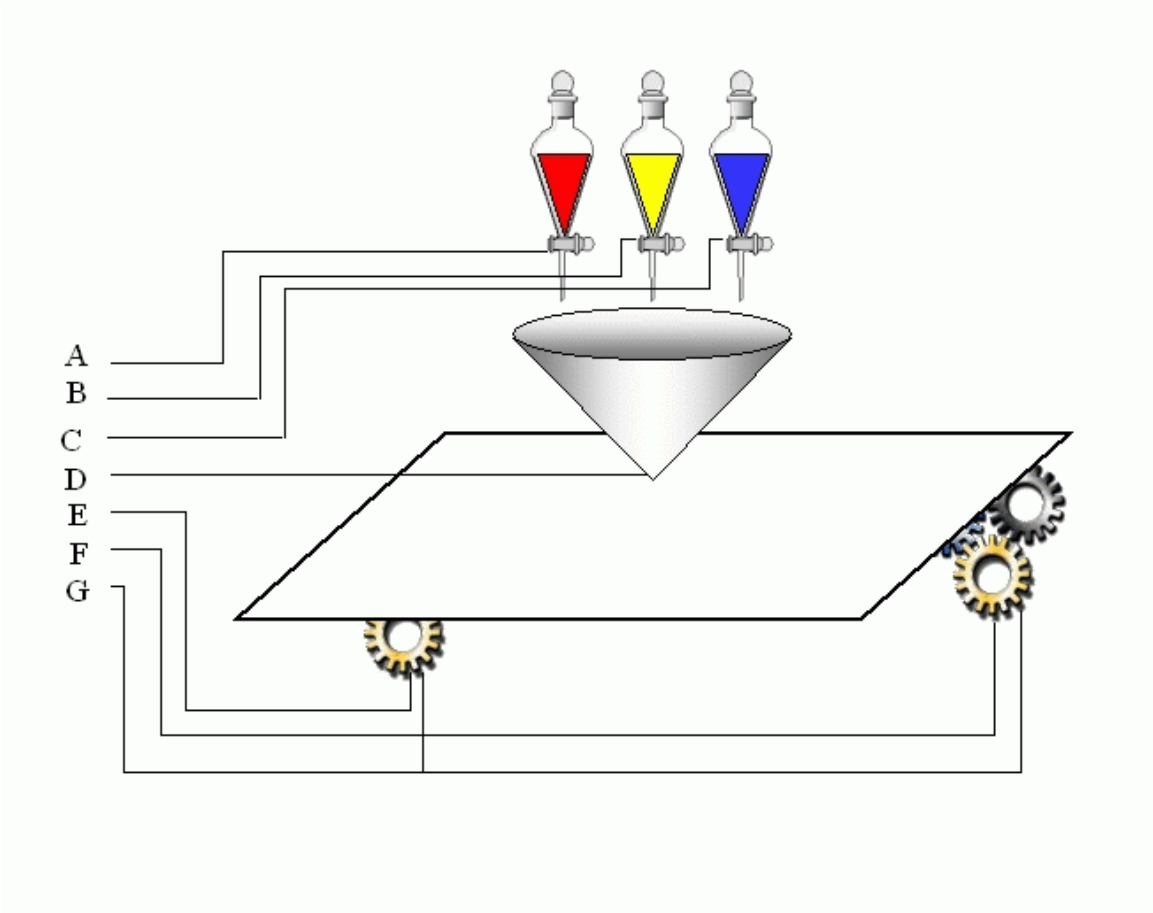
The goal of this question is to give you an understanding of the relationship between assembly language, machine instructions, and how they can be used to control a machine. You will also get more insight about the notion of code and data having the same representation, and some practice with designing a small circuit using logic gates.

The ideas needed here are similar to, but not identical to, what we've seen in class about how gates work, how instruction sets work, how a CPU-like machine works, and how to code things in binary.

Problem Statement

Imagine that some of the other MIMS students developed a great new machine that can produce computer-controlled paintings. The paint is mixed from the three primary colors (red, blue, and yellow), poured through a funnel, and placed, pixel by pixel, onto paper that is positioned underneath the funnel by moving along rollers. (See the figure.) When red and yellow are mixed, they create orange, when all three colors are mixed, they create black, etc.

i206 Spring 2013: Homework 3



These students didn't take i206, so they need your help in making the hardware and instruction set for controlling their painting machine.

They've set things up so there are wires that control the valves on the three paint dispensers (called A, B, and C). When a valve receives a value of 1, it snaps open enough for one blob of paint to flow out of the dispenser and into the funnel, and then snaps shut, all within one clock cycle.

There is another wire (called D) that controls the opening on the funnel. When it receives a 1 it snaps open long enough to paint the spot just underneath it with a perfect pixel of color, and then snaps shut. When the funnel valve opens, it uses up all of the paint in the funnel, leaving the insides of the funnel clean. (This is a special kind of paint that slips off of metal but sticks to paper.)

The hardware is set up so the rollers can either go forwards-backwards or left-right; they can't move in both directions simultaneously. (You can think of these directions as north-south, east-west, if that's easier for you.)

i206 Spring 2013: Homework 3

The wire called E controls the forwards-backwards movement of the paper on the appropriate rollers. When it receives a 1 it moves the paper forwards or backwards one pixel. The wire called F controls the left-right movement of the paper. When it receives a 1 it moves the paper one pixel left or right.

Finally, the wire called G is used to indicate the direction for the roller. When set to 1, if E is also set to 1, it tells the roller to go forwards. If G is set to 0 when E is set to 1, then it tells the roller to go one unit backwards. Similarly, when G is set to 1 and F is set to 1, it means go one unit left, otherwise one unit right.

The assembly language consists of the following 14 operations:

mix red
mix yellow
mix blue
mix purple
mix green
mix orange
mix black
mix white
paint
move left
move right
move forwards
move backwards
halt

Mix means dispense paint into the funnel, where it will mix itself.

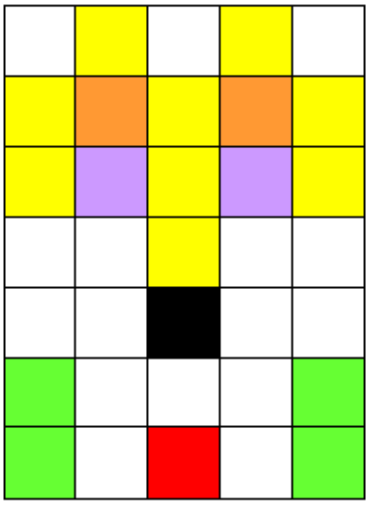
Paint means snap open the valve on the bottom of the funnel, so it will paint the pixel on the piece of paper that is currently under the funnel.

Move means move the paper one pixel either left-right or forward-backward.

Part A: Write a Program for the Machine

To get a feeling for the assembly language, first write a program that paints the three center rows of the painting shown below, using the assembly language instructions shown above. Assume the program starts with the funnel over the center pixel in the center row.

i206 Spring 2013: Homework 3

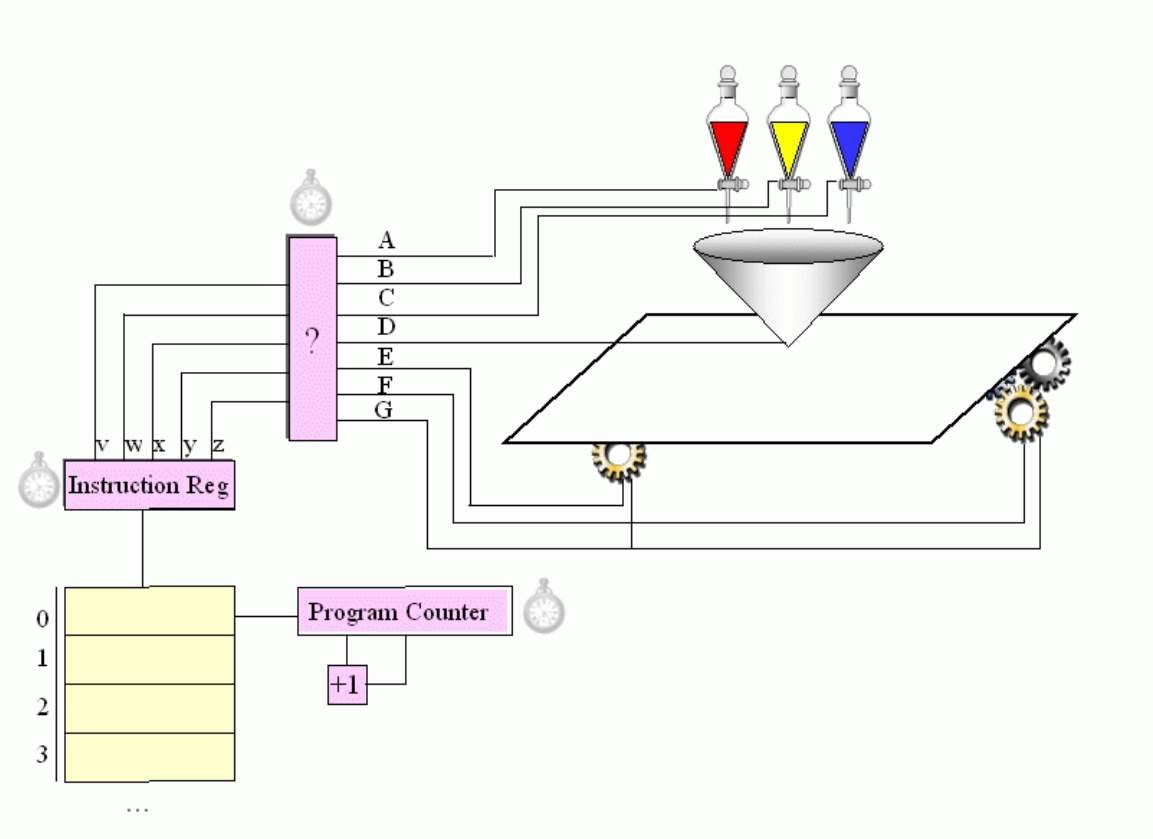


Part B: Design the Instruction Set

To enable the students to be able to run their machine, you need to design an instruction set that can implement the assembly language shown above. To save money, the students need you to restrict your instruction set to consist of instructions of length 5 bits. You also have to design some circuitry that will convert the machine code instructions into actions that run the machine. You have to have one machine code instruction for each and every one of the assembly language instructions shown above.

The system works as follows. (See the expanded figure below.)

i206 Spring 2013: Homework 3



The major components are connected to the clock, as shown in the figure.

The instructions get loaded into RAM, starting at memory address 0. The RAM is quite large, so you don't need to worry about running out of space.

The program counter starts at memory address 0. Each instruction takes four clock cycles to execute. In the first cycle, the contents of the memory location that is pointed to by the Program Counter are read into the Instruction Register. In the second cycle, the contents of the Instruction Register go through the circuit (denoted as ? in the figure) whose output is connected to wires A - G. In the third cycle, all the mechanical parts of the machine are activated and have time to do their work, thus responding to the values on A - G (they are deactivated at the end of the cycle). In the fourth cycle, the Program Counter is incremented.

So first design the binary representation of the instruction set shown above. Remember, instructions can only be 5 bits long. You should make a table that shows each instruction next to its assembly language representation.

i206 Spring 2013: Homework 3

Part C: Design the Circuit

Now design a circuit that connects the outputs of the instruction register to the wires labeled A - G. Use Boolean logic and whatever gates you need.

> Hint 1: build a truth table to help you with this.
>
> Hint 2: consider some of the larger circuits we looked at in class.
>
> Hint 3: Try to use good, modular design when creating this circuit. There should not be a spaghetti of wires all over the place. Using the first two hints will help with this.
>
> Hint 4: The circuit design will be pretty straightforward if you use the previous three hints. If you can't see how to do this, then rethink your instruction set design. You can't just make an arbitrary instruction set or your circuit design will be a mess. Figuring out how to do this is the most important part of this assignment.
>
> OPTIONAL: Optionally, design your circuit so that if an illegal instruction is received (meaning if some combination of inputs is received that does not correspond to a valid instruction), the paint machine does not do anything in response to that invalid instruction (it just sits and does not change state).