1. Announcements

2. Review
   a. All commands
      i. GET          get a number from keyboard into accumulator
      ii. PRINT       print contents of accumulator
      iii. LOAD Val   load accumulator with Val (Val unchanged)
      iv. STORE M     store contents of accumulator into memory location called M
      v. ADD Val      add Val to contents of accumulator
      vi. SUB Val     subtract Val from contents of accumulator
      vii. GOTO L     go to instruction labeled L
      viii. IFPOS L   go to instruction labeled L if accumulator is >= zero
               ix. IFZERO L go to instruction labeled L if accumulator is zero
               x. STOP      stop running
               xi. M Num    before program runs, set this memory location (called M) to numeric value Num
   
3. A more complicated program
   a. Multiplication
      i. Need SUB command
         1. GET
         2. STORE MUL1
         3. GET
         4. STORE MUL2
         5. A IFZERO B
         6. LOAD PROD
         7. ADD MUL1
         8. STORE PROD
         9. LOAD MUL2
        10. SUB ONE
        11. STORE MUL2
        12. GOTO A
        13. B LOAD PROD
        14. PRINT
        15. STOP
        16. ONE 1
        17. MUL1
        18. MUL2
        19. PROD
      ii. Also flow chart
   b. How would we do division
      i. Need IFPOS command
         1. Then what?

4. Back to the internals of the CPU
   a. How does the finite state machine work?
      i. Has to parse commands
         1. Is it GET or STORE or LOAD or ... or a WORD we do not recognize?
ii. Then has to execute commands

b. A simple example of parsing
   i. How to recognize COS109 character by character
      1. Show transitions
         a. Has 8 states
            i. Waiting for C
            ii. Waiting for O
            iii. Waiting for S
            iv. Waiting for 1
            v. Waiting for 0
            vi. Waiting for 9
            vii. ACCEPT
            viii. REJECT
         b. Transitions above
            i. If you saw what you were looking for, move forward
            ii. If not, go to reject
      ii. What if you don’t care if it is COS109 or CS109?
         1. Slight change
            a. Waiting for O becomes waiting for O or S
            b. Transition on S is to waiting for 1
            c. Transition on O is as before
      c. Now imagine we are building a simpler machine to just handle GET, GOTO and STORE
         i. Has states
            1. Waiting for S or G
               a. If you get one of these, transition to state that gets next letter
               b. If not, go to ERROR state
            2. If S, transition in turn to
               a. Waiting for T
               b. Waiting for O
               c. Waiting for R
               d. Waiting for E
               e. Waiting for “ “
               f. Then process label and act state
               g. If you don’t get what you want or label doesn’t exist, go to ERROR state
            3. If G, transition to
               a. Waiting for O or E
               b. If E
                  i. Waiting for T
                  ii. Then process input
               c. If O
                  i. Waiting for T
                  ii. Waiting for O
                  iii. Then process label and execute
iv. If you don’t get what you want or label doesn’t exist, go to ERROR state

4. At end, if all works, go back to START state

5. What we have defined is something called a finite state machine
   a. Finite State machine
      i. Has an alphabet
      ii. A finite number of states
      iii. Transition rules
      iv. And operates by
         1. Read the next symbol of input
         2. Based on your current state and the input
            a. Transition to a new state
            b. Possibly write a symbol of output
   b. What finite state machine can do
      i. Track simple processes
      ii. Example
         1. Combination lock
         2. Count change in e.g. drink machine
   c. What finite state machine cannot do
      i. Remember things
         1. You can only track where you’ve been or what input you’ve seen by recording this in the state name
      ii. Example
         1. I’ll read three symbols and if the second is A, I want to know if the first symbol matched the third; if it is not A, I want to know if they did not match.
            a. Can build machine
               i. After first symbol, you are in a state that records what that symbol was
               ii. After second symbol, you are either in state that says
                  1. Looking for first symbol or
                  2. Looking for something other than first symbol
            b. Cannot build machine where there is a state that says
               i. Do we match first symbol or
               ii. If second symbol was A, do ....

6. Finite state machines can be further abstracted to
   a. Have more power
      i. Some memory to remember where they’ve been
      ii. Input that doesn’t feel finite
      iii. Etc.
   b. Machines with all these properties are called Turing machines
      i. Devised by Alan Turing in grad college at Princeton in the 1930’s
      ii. Shown by Alonzo Church to be able to represent all computations, also at Princeton in 1930’s
iii. Serve as an abstract model of how we think about computing