Quiz 4, 25 points June 5, 2019 duration 30 minutes

Consider the following problem: A series of words are stored in a ROM. Each letter of the word is stored in a contiguous block of address. For example: The word EXAM will be stored as Address -> E, Address+1 -> X, Address+2->A, Address+3 -> M. In the next address there is a unique termination sequence that indicates if the word is complete (you can assume any particular sequence you wish). You can assume that the 26 letters are encoded as BCD i.e. A =1, B=2…Z=26. We want to be able to compare the incoming sequence to a particular word stored in memory. There are three additional inputs: Fetch, Word_Start[xx:0], and Compare. If Fetch is asserted high for one clock cycle, the system will fetch the stored word from memory. Word_Start[xx:0] contains the starting memory address of the stored word. If Compare is asserted high for 1 clock cycle, it begins the comparison sequence between the word stored in memory and the input word. The next clock cycle following Compare, the input string that needs to be compared begins. If the words match up to the termination sequence, the system outputs a signal Match, if not it outputs a signal Fail. Any sequences either aligned with or after the termination sequence are ignored. Construct a FSM that achieves this comparison. You might want to consider using a microcoded FSM. If you are using a microcoded machine draw the logic block in your branch logic and any other logic block you might need.

Bonus: Enhance the machine so that if the signal fails to Match the machine can assert the signal before the entire word sequence is received and the machine no longer accepts the input sequence and asks the sender to send a new sequence.

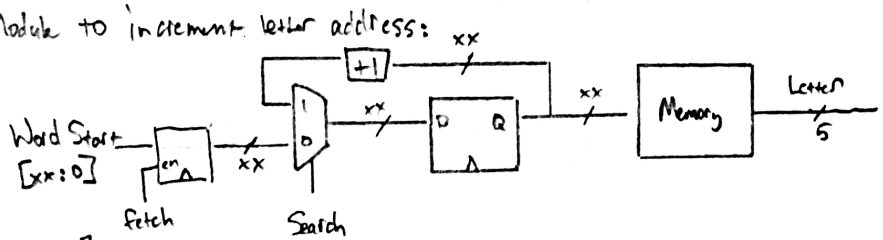
BCD up to 26 => 5 bit words for each letter, with 6 available for additional use

Assume Word Start is available when Fetch asserted

Have System hold letter in Word Start address until Compare asserted for letter-wise Comparison

-> Use enabled FF from previous quiz

Module to increment letter address:



Enable Flop holds address

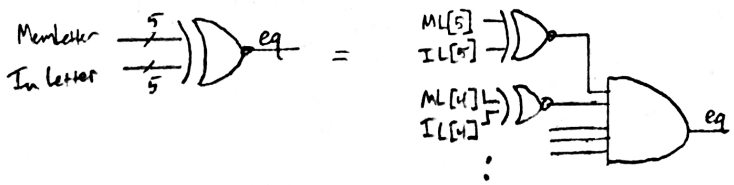
Once Compare asserted, Search asserted high until Match detected or Mismatch detected

Controller needs to initiate comparisons until a mismatch is found, or the termination sequence is detected.

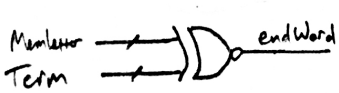
Assume the input doesn't include termination character,

The word in memory is of an unknown length, so fetching all the characters is not a good idea

Instead, use XNOR to compare each letter

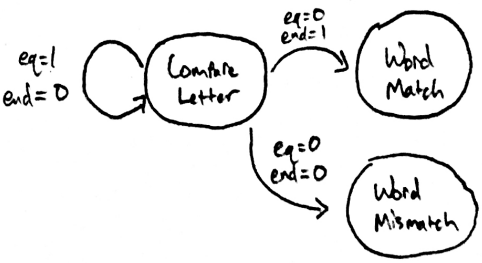


Also, Need to check Memory for termination character



Now, define State transitions based on eq and endWord

Assume for now that Something else monitored Fetch and Compare



(Now that if input doesn't contain term, eq=0 for term character)

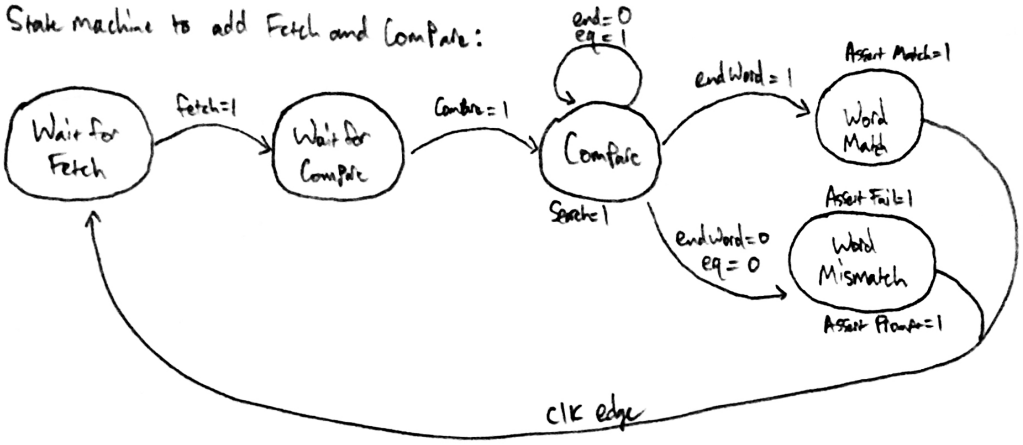
Hence, eq=1 and end=1 cannot happen, So we can reduce (eq=0 and end=1) to end=1

Finally, we can define some simple control logic from the States and inputs, and put it together
Name the variables and define functions

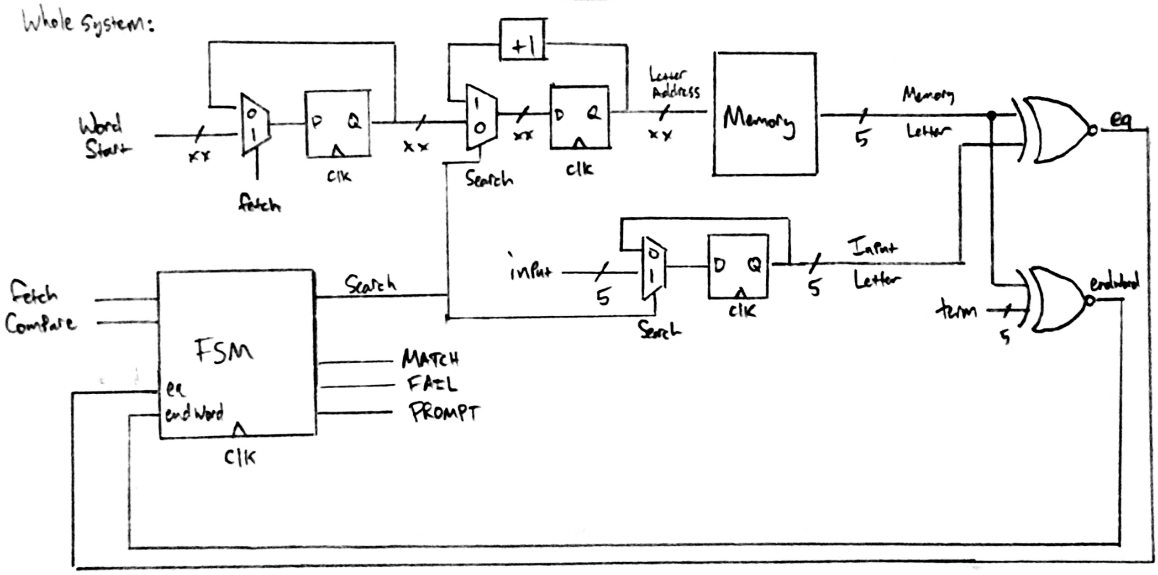
Prompt: If Stop asserted and Fail = true, ask user for new input

Search: If fetch asserted for 1 cycle, and Compare asserted for 1 cycle, increment Memory address until Stop
Also, latch input until Stop asserted and Start Comparing cycle

Now, Expand State machine to add Fetch and Compare:



Whole System:



Assumptions (You should ALWAYS state these)

1) Compare will not be asserted before fetch
2) Fetch will only be asserted if Word Start is settled and correct
   Address will be updated if another fetch comes before Compare
3) The System can detect Match or Fail if they are only
   asserted for 1 cycle
4) The termination character is fixed