Tutorial 3 - Solutions

Q1. Let the I, O1, O2, O3 and O4 be the states as described below.

States	**Description**
I	Initial state
O1	State after s3 is made high
O2	State after s5 is made high
O3	State after s8 is made high
O4	State after s10 is made high

Hence, the register values for each state are,

State	**T**	**A**	**W**	**B**
I	07	02	05	04
O1	07	02	05	02
O2	02	02	05	02
O3	02	02	09	02
O4	02	09	09	02

Q2.

1. 8 lines.
2. 16 lines.
3. 48 lines.
4. Address bus is unidirectional.
   Reason: Address lines are connected to the decoder in the memory (as against any register) so there is no chance of anything going to the address bus from the memory. Address put on the address bus only goes in one direction to memory.

   Data bus is bi-directional.
   Reason: The microprocessor has to fetch (read) the data from memory or input device for processing, and after processing, it has to store (write) the data to memory or output device. Hence, the data bus is bi-directional.

5. Pin numbers 12-19 (AD_0 - AD_7) are used for the data bus.

6. Pin numbers 12-19 (AD_0 - AD_7) and 21-28 (A_8 - A_{15}) are used for the address bus. Pin numbers 12-19 are common to data bus.
7. The process steps are:
    a. Let's assume that 30H is stored in register H and 0AH is stored in register L. The following data transfer in registers can be done to transfer the address to the desired registers:
    Address buffer ← H, Address/data buffer ← L
    Alternatively, we can directly put value 30H into the address buffer and 0AH in the Address/data buffer register.
    b. Since outputs of the buffer registers in (a) are connected to respective lines of the address bus, making OE pins high of both registers will now put a valid address 300AH (or 0x300A) on the address bus pointing to the desired location in memory.
    c. In the next clock cycle now making OE of memory high and WR(A) high (anded with clock cycle) will transfer data from this memory location to A.

Note: Here, the accumulator is referred to as A.

Q3.

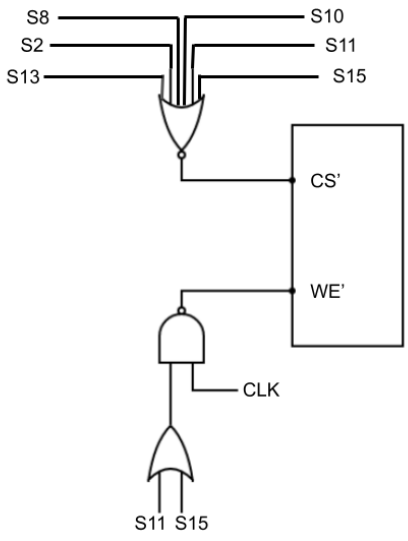
a. For this problem we will use 3 registers A, B and W. The logic behind solving this problem is as follows:
    1. Store addr1 in W.
    2. Store M[addr1] in A.
    3. Store addr2 in B.
    4. Transfer addr2 from B to MA.
    5. Store M[addr2] in B.
    6. Write the value stored in A to M[MA]. (Note: Here, MA still contains addr2, therefore we are essentially writing initial M[addr1] to M[addr2])
    7. Transfer addr1 from W to MA.
    8. Write the value stored in B to M[MA]. (Note: Here, MA contains addr1, therefore we are essentially writing initial M[addr2] to M[addr1])

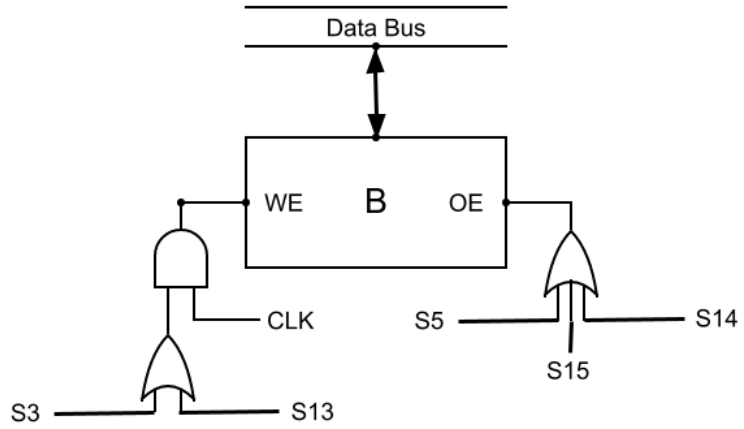
    After following the above steps we would have implemented the SWAP addr1 addr2 command as mentioned in the question.

    The sequence of microinstructions for implementing the SWAP function is as described below.

States	Microinstructions
S1	(MA) ← (PC)
S2	(IR) ← M[MA] ; (PC) = (PC) + 1
S1	(MA) ← (PC)
S8	(W) ← M[MA] ; (PC) ← (PC) + 1
S9	(MA) ← (W)
S10	(A) ← M[MA]
S1	(MA) ← (PC)
S13	(B) ← M[MA] ; (PC) ← (PC) + 1
S14	(MA) ← (B)
S13	(B) ← M[MA]
S11	M[MA] ← (A)
S9	(MA) ← (W)
S15	M[MA] ← (B)

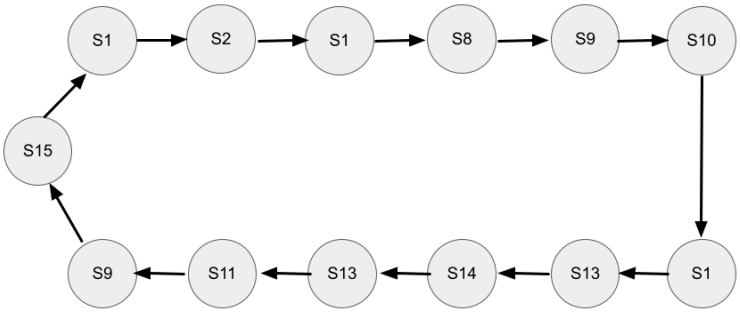
b. The highlighted states in the above sequence are the new states that need to be accommodated. We need to make additional connections according to the microinstructions of these states.





We need to make only these changes in the circuit, rest all remains the same.

c. State diagram (before redefinition of states):

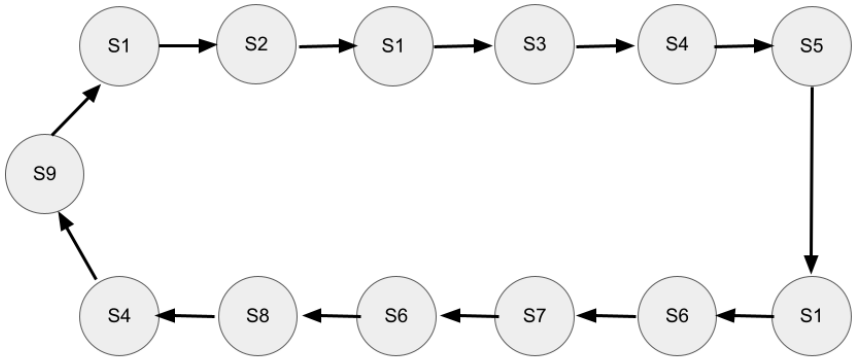


After new state definitions,

States	**Microinstructions**
S1	(MA) ← (PC)
S2	(IR) ← M[MA] ; (PC) = (PC) + 1
S1	(MA) ← (PC)
S3	(W) ← M[MA] ; (PC) ← (PC) + 1
S4	(MA) ← (W)
S5	(A) ← M[MA]
S1	(MA) ← (PC)
S6	(B) ← M[MA] ; (PC) ← (PC) + 1
S7	(MA) ← (B)

S6	(B) ← M[MA]
S8	M[MA] ← (A)
S4	(MA) ← (W)
S9	M[MA] ← (B)

State Diagram:



Circuit Diagram: