ECE 120 Midterm 3 Review

Topic Overview

- Finite State Machine
  - Analysis
  - FSM Design
  - Counter Design
- Memory
  - Flip Flops
  - RAM
  - Tri-State Buffer

FSM

FSM Knowledge

Three main components of FSM's:

- Next State Logic
- Current State
- Output Logic

How many bits are needed to represent N states in a standard FSM?

FSM Knowledge

Three main components of FSM's:

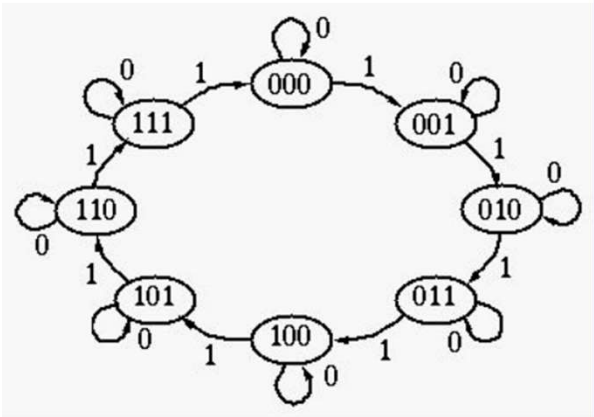
- Next State Logic
- Current State
- Output Logic

How many bits are needed to represent N states in a standard FSM?

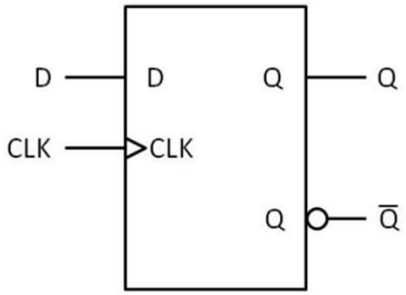
Requires $\lceil \log_2(N) \rceil$ bits to represent each state (ceiling / round up)

FSM (Lecture 19)

- Two types: Mealy and Moore
  - Moore: Outputs are based off of current state
  - Mealy: Outputs are based off of past state and current transition
  - Mealy uses less states, but possibly glitches
- Consist of five elements:
  - Finite number of states
  - Finite number of external inputs
  - Finite number of external outputs
  - Explicit number of state transitions
  - Explicit definition of what determines output values



D Flip Flops



- Made from two D latches. Behavior of D latch
- D latch behavior

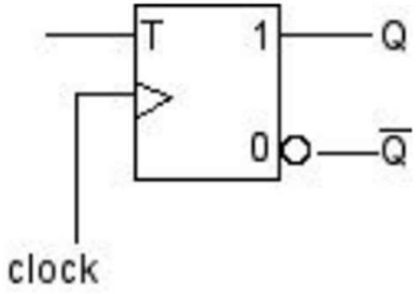
C	D	R	S	Q+	
1	0	1	0	0	Reset
1	1	0	1	1	Set
0	0	0	0	Q	No change
0	1	0	1	1	No Change

- D flip flop stores the input bit D

D	Q+
0	0
1	1

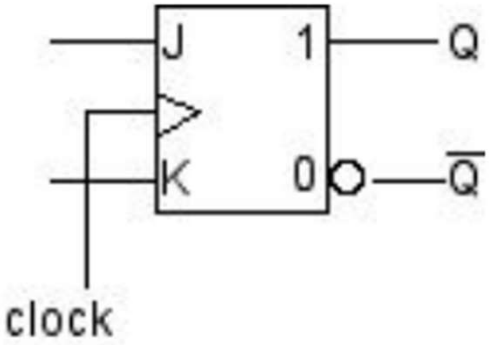
T flip-flop (toggles its output when T=1)

T	Q+
0	**Q**
1	**Q'**



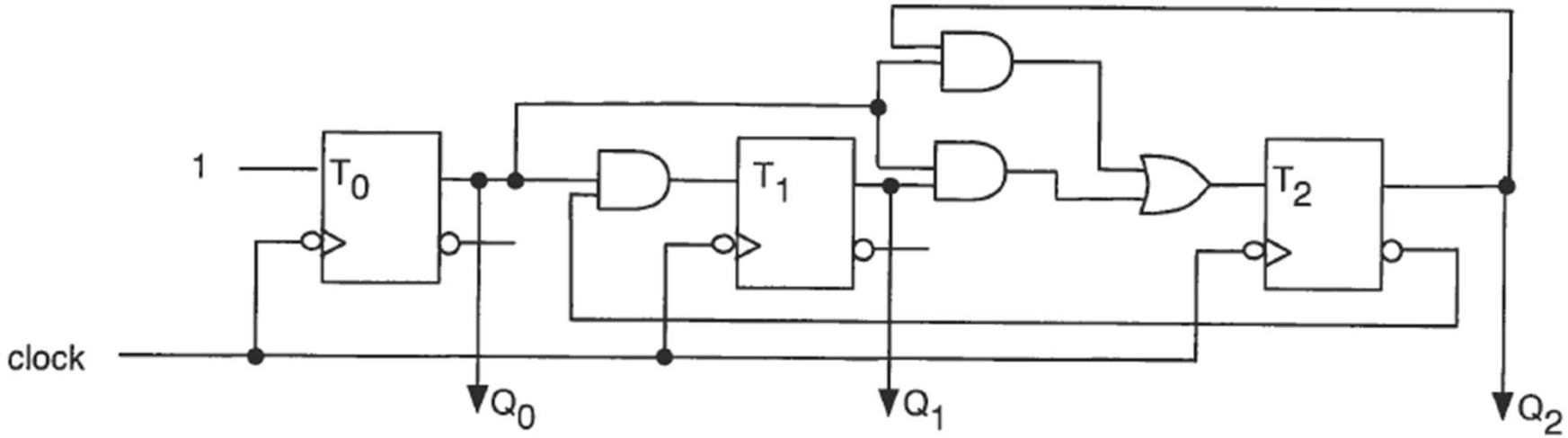
JK flip-flop (T flip-flop when JK=1, SR flip-flop otherwise)

J	K	Q+
0	0	**Q**
0	1	**0**
1	0	**1**
1	1	**Q'**



FSM Analysis

The circuit below shows a 3-bit synchronous counter constructed using 3 negative-edge-triggered **T flip-flops**.



Give Boolean expressions for the T-flip-flop inputs, each as a function of Q{1,2,3} .

FSM Analysis - Continued

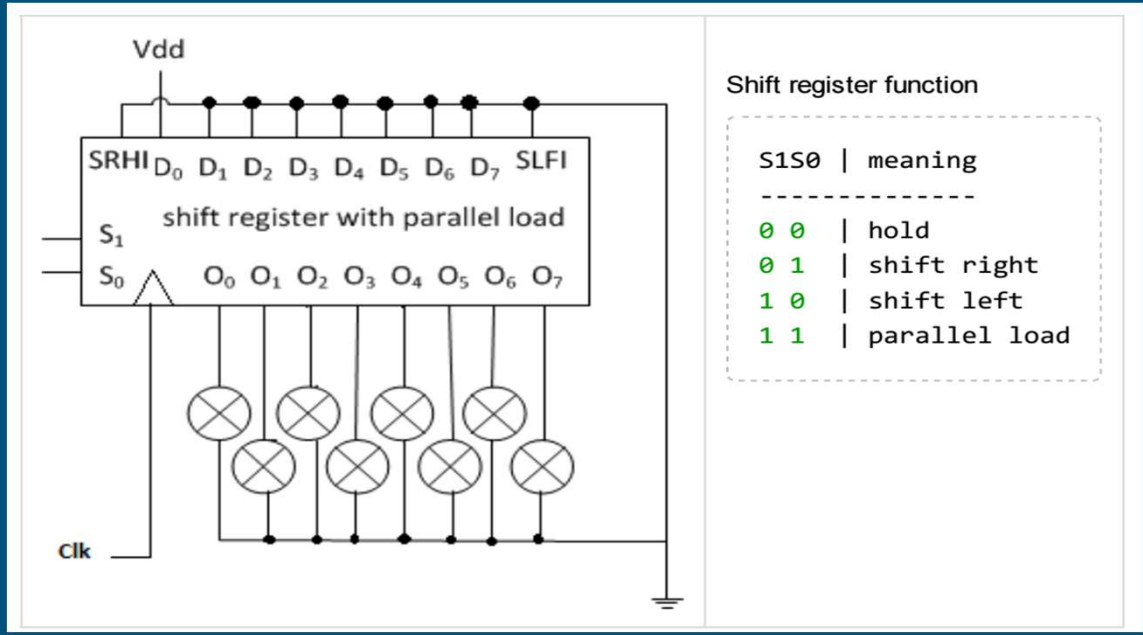
Complete the Transition Table:

Complete the State Transition Diagram:

If the FSM begins at state Q1Q2Q3 = 000, what is the counter sequence ?

Is this FSM self-starting ?

FSM Design



1. List all inputs to your FSM
2. LIst all outputs to your FSM
3. Draw the Moore FSM state diagram to control the shift register. Clearly mark state names, state outputs, and state transitions. Your design should use as few states as possible in order to receive full credit.

Optimized Boolean Expression

When you're trying to find the boolean expression for the outputs for an FSM or output logic,
Consider:

- Both POS and SOP form
- Try to see which of these expression minimizes the total number of gates/area
- Check HW **and the solutions** if you need to review this
    - HW 8 and 9!

Memory

Memory

Memory is an important concept in computing. Memory is the ability to store a value.

Memory has 3 basic parts to describe its structure:

Address: A unique identifier associated with each memory location.

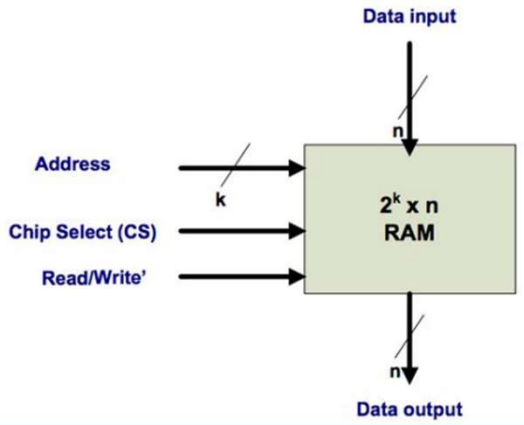
Addressability: Number of bits in each location.

Address Space: The total number of uniquely identifiable memory locations.

Notes: 1MB = 2^20, 1KB = 2^10

RAM

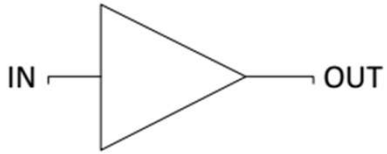
- Inputs:
  - **Addr** - Address to save data to
  - **Data** - Information to be saved
  - **CS** - "Enable bit", turns on RAM Chip
  - **R/W'** - Specifies whether or not you can write (R = 1, W = 0)
  - **Dout** - Output data
- Notes: 1M = 2^20 and 1K = 2^10



CS	R/W'	Operation
0	X	No operation
1	0	write
1	1	read

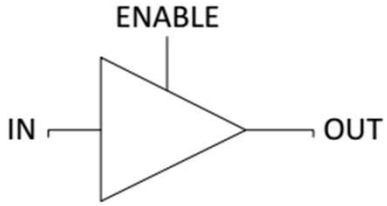
Tri-state buffer

- A *buffer* is a device that passes an input to its output



IN	OUT
0	0
1	1

- A *tri-state buffer* is a device with 3 output states: 0, 1, and 'Z' (high impendence)

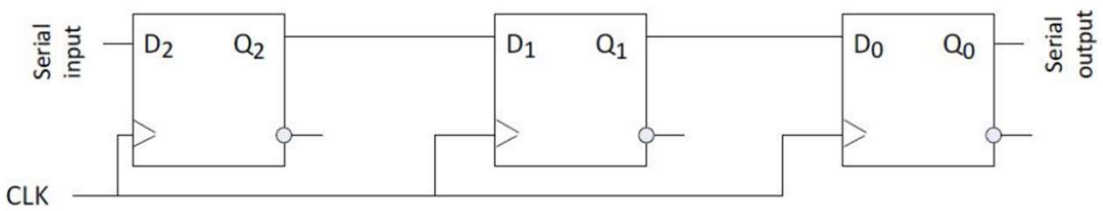


IN	E	OUT
0	1	0
1	1	1
X	0	Z

  - Z, or Hi-Z output means "Disconnected" - no output appears at all

  - We can build a tri-state buffer using just a few MOS FETs

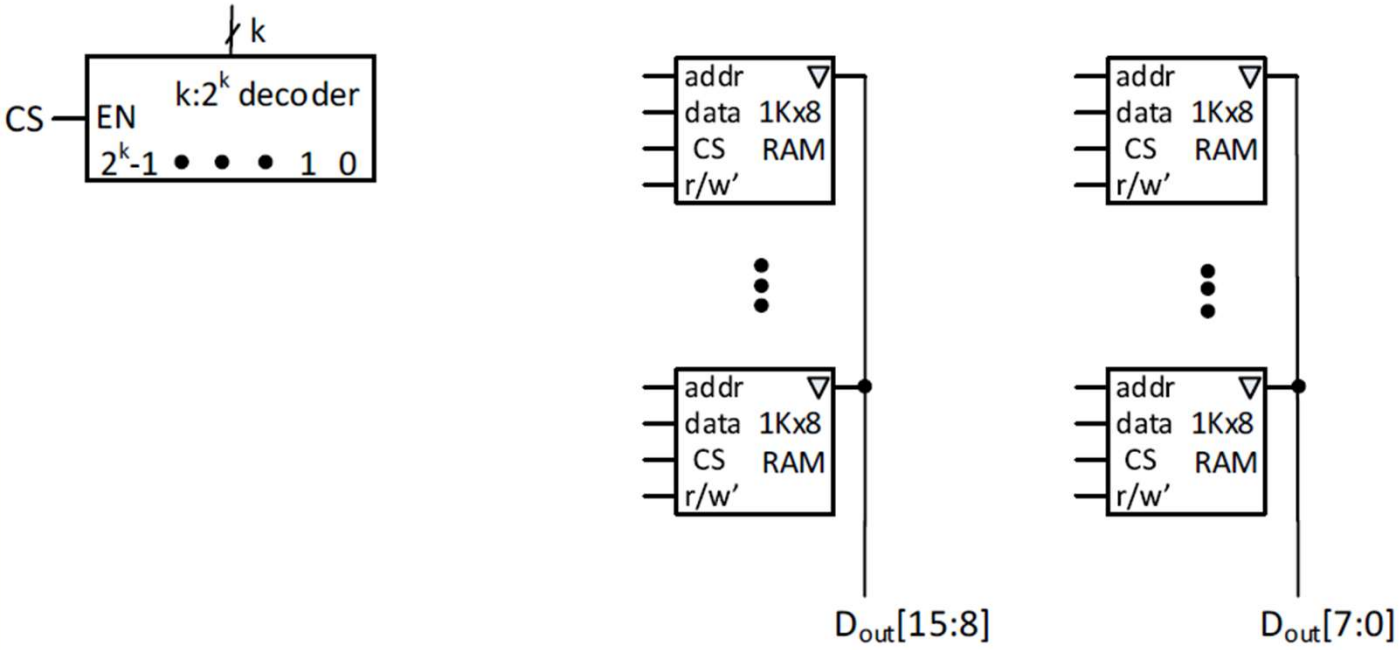
Registers



- Is group of flip flops to store information with inputs Common clock, load, clear
- Shift Registers
  - Data is entered one bit per clock cycle, data in flip flops are shifted in a direction
  - Left shift or right shift
  - Logical shift data is entered in the left or right flip flop
  - Circular shift the output is fed back as the input
  - Arithmetic left shift corresponds to multiply by 2. like logical but input is zero.
  - Arithmetic right shift corresponds to divide by 2
- Parallel Load registers
  - New bits for each flip flop is loaded all at the same time
  - Need load signal, data loaded when clock and load are both high

Fall 2013
Problem 3

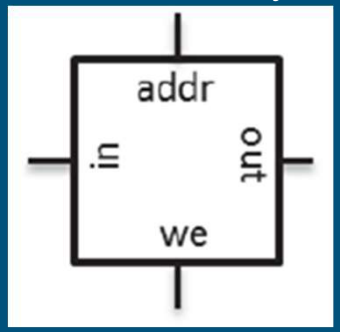
Using 1K x 8 RAM chips, implement a 32K x 16 RAM. Each 1K x 8 RAM chip has inputs **data**, **addr**, **CS**, and **r/w'** and an output gated by a tri-state buffer. Finish the implementation by drawing the missing connections and labeling all newly added wires. (You do not need to draw all the rows, they are shown as " ... ", but be sure the pattern is clear.) The RAM output wires and CS are already drawn for you.



ECE 190: Problem 3 Memory

In this problem you will be working with 2x1-bit memory cells. Each cell stores 2 bits which can only be accessed one at a time. Which bit is accessed is determined by the address input to the memory cell. Given four 2x1-bit memory cells, build a 4x2-bit memory unit using only AND, OR, NOT gates and MUXes.

Individual Memory Cell



LC3

SPRING 2013 PROBLEM 7

Name all components of the Von Neumann architecture and briefly describe their function (101)

SPRING 2013 PROBLEM 7

Name all components of the Von Neumann architecture and briefly describe their function (101)

a. Memory - stores data and program

SPRING 2013 PROBLEM 7

Name all components of the Von Neumann architecture and briefly describe their function (101)

a. Memory - stores data and program

b. Processing Unit - performs the data processing

SPRING 2013 PROBLEM 7

Name all components of the Von Neumann architecture and briefly describe their function (101)

a. Memory - stores data and program
b. Processing Unit - performs the data processing
c. Input - means to enter data and program

SPRING 2013 PROBLEM 7

Name all components of the Von Neumann architecture and briefly describe their function (101)

a. Memory - stores data and program
b. Processing Unit - performs the data processing
c. Input - means to enter data and program
d. Output - means to extract results

SPRING 2013 PROBLEM 7

Name all components of the Von Neumann architecture and briefly describe their function (101)

a. Memory - stores data and program
b. Processing Unit - performs the data processing
c. Input - means to enter data and program
d. Output - means to extract results
e. Control Unit - controls the order of instruction execution

SPRING 2013 PROBLEM 7

List all phases of the Von Neumann Instruction cycle (106)

SPRING 2013 PROBLEM 7

List all phases of the Von Neumann Instruction cycle (106)

a. Fetch - obtain next instruction from memory, store it in IR

SPRING 2013 PROBLEM 7

List all phases of the Von Neumann Instruction cycle (106)

a. Fetch - obtain next instruction from memory, store it in IR

b. Decode - instruction stored in IR is looked at to decide what part of the microarchitecture needs to be used in the execution of the instruction

SPRING 2013 PROBLEM 7

List all phases of the Von Neumann Instruction cycle (106)

a. Fetch - obtain next instruction from memory, store it in IR

b. Decode - instruction stored in IR is looked at to decide what part of the microarchitecture needs to be used in the execution of the instruction

c. Evaluate Address - computes address of the memory location that is needed to process the instruction

SPRING 2013 PROBLEM 7

List all phases of the Von Neumann Instruction cycle (106)

a. Fetch - obtain next instruction from memory, store it in IR

b. Decode - instruction stored in IR is looked at to decide what part of the microarchitecture needs to be used in the execution of the instruction

c. Evaluate Address - computes address of the memory location that is needed to process the instruction

d. Fetch Operands - loading values from register file, or loading operands from memory (depends on instruction)

SPRING 2013 PROBLEM 7

List all phases of the Von Neumann Instruction cycle (106)

a. Fetch - obtain next instruction from memory, store it in IR

b. Decode - instruction stored in IR is looked at to decide what part of the microarchitecture needs to be used in the execution of the instruction

c. Evaluate Address - computes address of the memory location that is needed to process the instruction

d. Fetch Operands - loading values from register file, or loading operands from memory (depends on instruction)

e. Execute - Instruction carried out

SPRING 2013 PROBLEM 7

List all phases of the Von Neumann Instruction cycle (106)

a. Fetch - obtain next instruction from memory, store it in IR

b. Decode - instruction stored in IR is looked at to decide what part of the microarchitecture needs to be used in the execution of the instruction

c. Evaluate Address - computes address of the memory location that is needed to process the instruction

d. Fetch Operands - loading values from register file, or loading operands from memory (depends on instruction)

e. Execute - Instruction carried out

SPRING 2013 PROBLEM 7

Describe the purpose of the following registers in the Von Neumann Architecture:

a. PC - keeps track where the program is in memory (updated in fetch)

SPRING 2013 PROBLEM 7

Describe the purpose of the following registers in the Von Neumann Architecture:

a. PC - keeps track where the program is in memory (updated in fetch)

b. IR - holds the current instruction being executed

SPRING 2013 PROBLEM 7

Describe the purpose of the following registers in the Von Neumann Architecture:

a. PC - keeps track where the program is in memory (updated in fetch)

b. IR - holds the current instruction being executed

c. MDR - acts like a buffer when we read/write from/to memory

SPRING 2013 PROBLEM 7

Describe the purpose of the following registers in the Von Neumann Architecture:

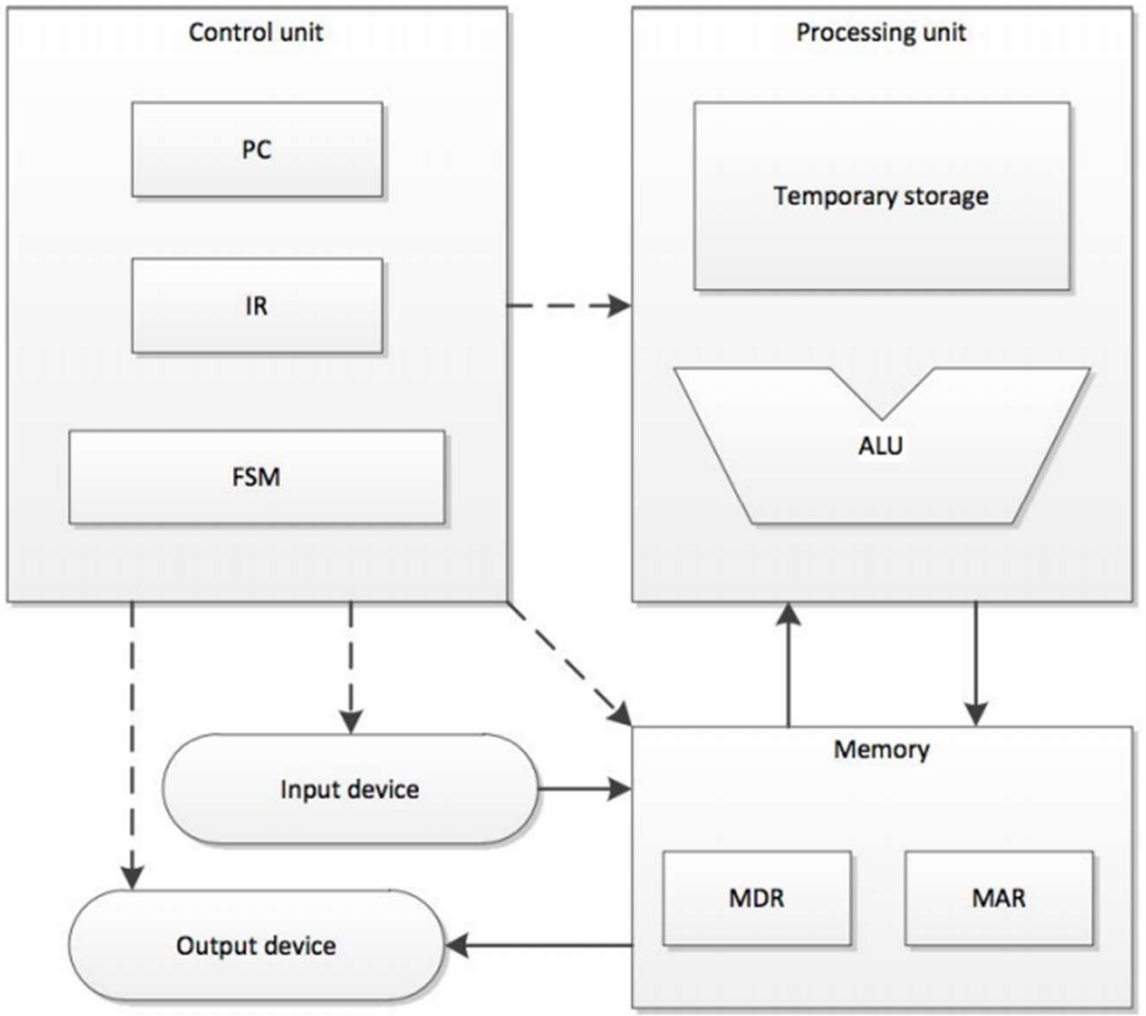
a. PC - keeps track where the program is in memory (updated in fetch)
b. IR - holds the current instruction being executed
c. MDR - acts like a buffer when we read/write from/to memory
d. MAR - holds the address that you are accessing in memory

SPRING 2013 PROBLEM 7

Describe the purpose of the following registers in the Von Neumann Architecture:

a. PC - keeps track where the program is in memory (updated in fetch)
b. IR - holds the current instruction being executed
c. MDR - acts like a buffer when we read/write from/to memory
d. MAR - holds the address that you are accessing in memory
e. General-purpose register file - R0-R7

Von Neumann Architecture



SPRING 2016 PROBLEM 5

The following LC-3 program fragment, represented as four hexadecimal numbers, is stored in memory at the indicated locations and the following values are stored in registers:

Address	Instruction
x3FFF	xAFFE
x4000	x2001
x4001	x743F
x4002	x3002

Register	Value
R0	xF021
R1	xF023
R2	xF025
R3	xF027

Address	Instruction	Binary instruction	RTL (Be specific to this instruction)
x3FFF	xAFFE	1010 111 111111110	R7 ← M[M[PC - 2]] setcc
x4000	x2001		
x4001	x743F		
x4002	x3002		

SPRING 2016 PROBLEM 5

The following LC-3 program fragment, represented as four hexadecimal numbers, is stored in memory at the indicated locations and the following values are stored in registers:

Address	Instruction
x3FFF	xAFFE
x4000	x2001
x4001	x743F
x4002	x3002

Register	Value
R0	xF021
R1	xF023
R2	xF025
R3	xF027

Address	Instruction	Binary instruction	RTL (Be specific to this instruction)
x3FFF	xAFFE	1010 111 111111110	R7 ← M[M[PC - 2]] setcc
x4000	x2001	0010 000 000000001	
x4001	x743F	0111 010 000 111111	
x4002	x3002	0011 000 000000010	

SPRING 2016 PROBLEM 5

Address	Instruction	Binary instruction	RTL (Be specific to this instruction)
x3FFF	xAFFE	1010 111 111111110	R7 ← M[M[PC - 2]] setcc
x4000	x2001	0010 000 000000001	R0 ← M[PC+1] set cc
x4001	x743F	0111 010 000 111111	M[R0-1] ← R2
x4002	x3002	0011 000 000000010	M[PC+2] ← R0

2. (14 points) Assuming PC is initially set to x4000, trace the execution of the given program segment for **two** instruction cycles, filling in the table below. Write down the values stored in the PC, IR, MAR, MDR, R0, N, Z, and P registers **at the end of each instruction cycle**. Values for PC, IR, MAR, MDR, and R0 should be written in ***hexadecimal***. Values for N, Z, and P should be written in ***binary***.

PC	IR	MAR	MDR	R0	N	Z	P

FALL 2016 PROBLEM 5

The registers of an LC-3 processor currently have the values shown in the table to the right.

R0	x0111	R4	bits	PC	x3001
R1	x3002	R5	bits	IR	xE622
R2	x1175	R6	bits	MAR	x3000
R3	x3023	R7	bits	MDR	xE622

The table to the right shows some of the contents of the LC-3 processor's memory.

When the bits represent instructions, an interpretation has been provided for you in RTL.

address	contents	RTL interpretation
x3000	1110 0110 0010 0010	R3 ← PC + x0022
x3001	0110 0010 1100 0001	R1 ← M[R3 + 1]
x3002	1001 0100 0111 1111	R2 ← NOT R1
x3003	0111 0100 1100 0010	M[R3 + 2] ← R2
	...	...
x3023	0000 1111 0000 0101	PC ← PC + xFF05
x3024	0001 0011 0001 0100	(data: x1314)
x3025	1101 1010 1011 1100	(data: xDABC)
x3026	0000 0000 0000 0000	(no operation)

Here's the question:

The LC-3 processor **PROCESSES THREE INSTRUCTIONS**.

1. Write a complete list of the sequence of values taken by the MAR register as the LC-3 processes these instructions.
2. Write the FINAL values (after the 3 instructions are processed) of each register and memory location. If you cannot know a particular value, write "bits".

Extra

Synchronization Problem

Draw a **datapath** (yes, only the datapath, not the FSM) of a circuit that adds two N-bit 2's complement binary numbers, A and B, stored in two N-bit right shift registers, R_A and R_B. Each shift register has serial output SO and serial input SI. Beginning with the least significant pair of bits in R_A and R_B, the circuit should add one pair at a time through a single fulladder (FA) circuit. The sum bit from the FA output should be stored back into shift register R_A.

Hint: You only need two N-bit shift registers, a single fulladder (FA) circuit, and a D flip-flop. Assume that the flip-flop starts in the 0 state.

Synchronization Solution