midterm exam

COSC-120, Computer Hardware Fundamentals, fall 2012
Computer Science Department
Georgetown University

NAME Soln

Open books, open notes (laptops included). Show and explain **all** your work. Answers w/o explanation **will not get credit**. Questions are always perfectly clear to me, and are also maybe clear to you, but sometimes with a different meaning. Therefore, partial credit is important, and I will give as much credit for whatever question you seem to be answering as far as I have evidence in the form of explanation. This applies to questions that are over your head as well.

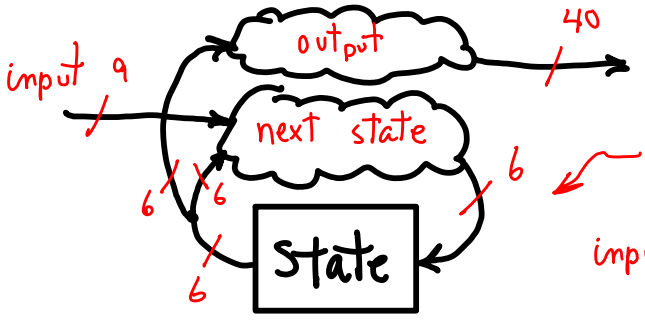
The LC3's micro-coded controller ("uSeq") is nothing more than a finite-state machine (FSM). It has these inputs:

```
BEN         //-- (1 bit) Used for opcode "0000": indicates the branch condition is true or false.
INT         //-- (1 bit) Indicates that some input/output device is requesting attention.
PSR[15]     //-- (1 bit) Indicates the currently executing program has privileged status.
R           //-- (1 bit) Indicates a memory or IO device has completed a read or write operation.
IR[15:12]   //-- (4 bits) The opcode of the currently executing instruction.
IR[11]      //-- (1 bit) Used for opcode "0100", indicates bits IR[10:0] are to be interpreted as data.
```

= 9

uSeq (an instance of the uControl module) has 59 states and 40 output bits. The outputs control the LC3's datapath. This FSM is a Moore machine: Outputs are associated with states, not state transistions: outputs only change when state changes.

Question. Below is a general FSM diagram for uSeq: clouds indicate combinatorial functions, a square indicates state elements, arcs are connections between them. Label the input, the output, the next state, and the current state. For each connection, label the number of bits. You may assume one-hot state encoding, or some other state encoding. Explain what type of state encoding you intend.

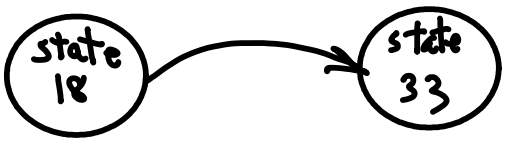


LC3 control signals

next-state bits: 59 states → 64 state codes $= 2^6$ → 6 bits for state

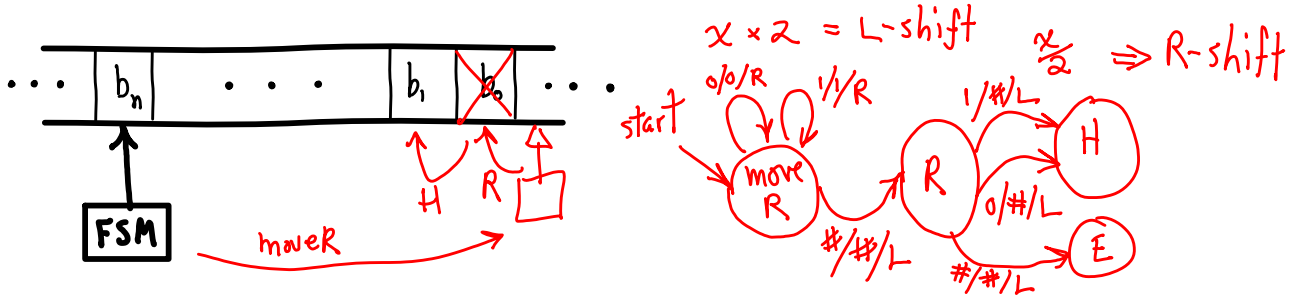
input bits = 9

Question. Consider the portion of the uSeq's state-transistion diagram shown below. Although only 1 arc is shown from state-18 to state-33 for simplicity's sake, how many state transistion arcs must there actually be? (Hint, consider the number of possible different inputs for state-18. Each input bit-pattern is a different symbol.)



uSeq has 9-bit input → $2^9 = 512$ possible inputs for 9-bit symbols ⇒ 512 arcs, one for each symbol

Question. Design a Turing Machine that does integer division by two (remainder is ignored). Input is an (n+1)-bit, binary-encoded, positive integer (b_n ... b_1 b_0). Show the machine's FSM diagram with state transistions. The input tape content and initial R/W head position is shown. The tape is blank, except for the input. The symbol set is {**0**, **1**, **#**} (**#** is a blank cell). The final R/W head position should be on the cell containing the low-order bit of the result. It halts in state **E** if there was an error, and in state **H**, otherwise.These states have no transitions.



Question. For your TM above, show a unary encoding of its FSM. Use the symbols { **<**, **>**, **#**, **1**, **L**, **R** , **E**, **H** }. Rules are coded in this pattern "**<** state **#** input **#** output **#** move **#** next-state **>**". Explain how your state and symbol encodings match your diagram above. Patterns for the rules are given below. No rules are needed for states **E** and **H**.

```
<  1  #  1  #  1  #  R  #  1  >
<  1  #  11 #  11 #  R  #  1  >
<  1  #  111 #  111 #  L  #  11 >
<     #     #     #     #     >
```

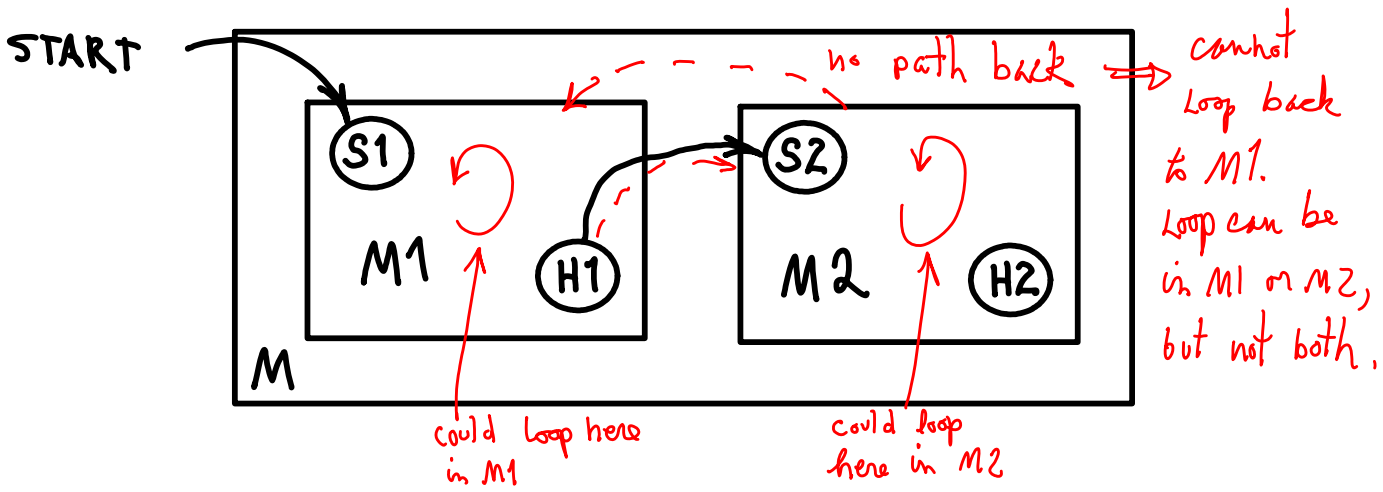
state moveR
moveR = 1
R = 11

0 = 1
1 = 11
# = 111

```
<  11 #  1  #  111 #     #  H  >
<  11 #  11 #  111 #     #  H  >
<  11 #  111 #  111 #     #  E  >
<     #     #     #     #     >
```

state R

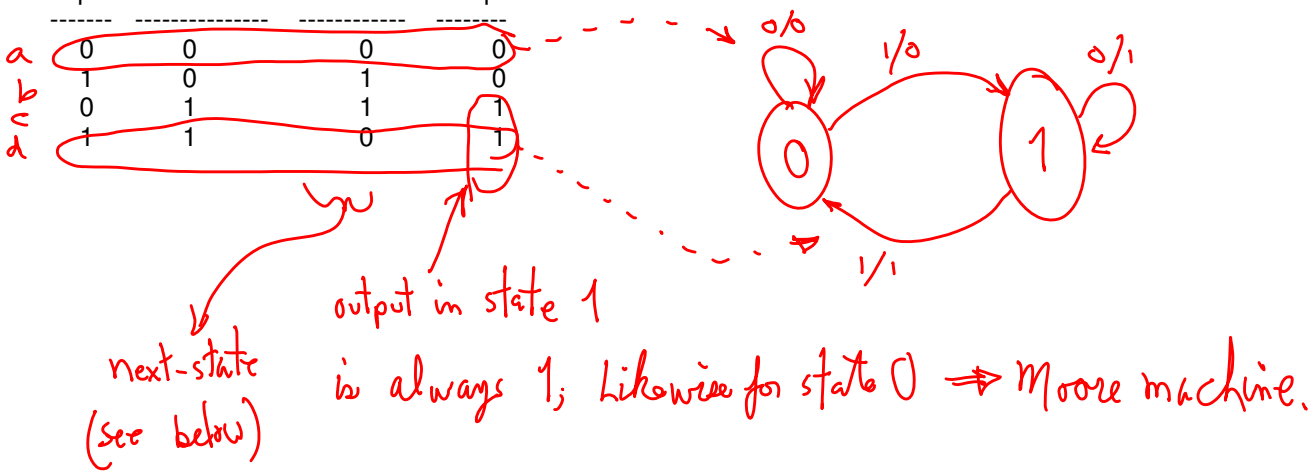
Question. Suppose we build a TM, M, by composing it from two others, M1 and M2. M's start state is M1's start state (S1). M's halt state is M2's halt state (H2). M1's halt state (H1) is the same state as M2's start state (S2); that is, every state transition that enters H1 is actually a transition to S2. We don't know if M will ever halt, given some input. Suppose M gets stuck in a loop. Could the loop include M1 states and M2 states, or only M1 states, or only M2 states? Identify in the diagram where M might get stuck in a loop.



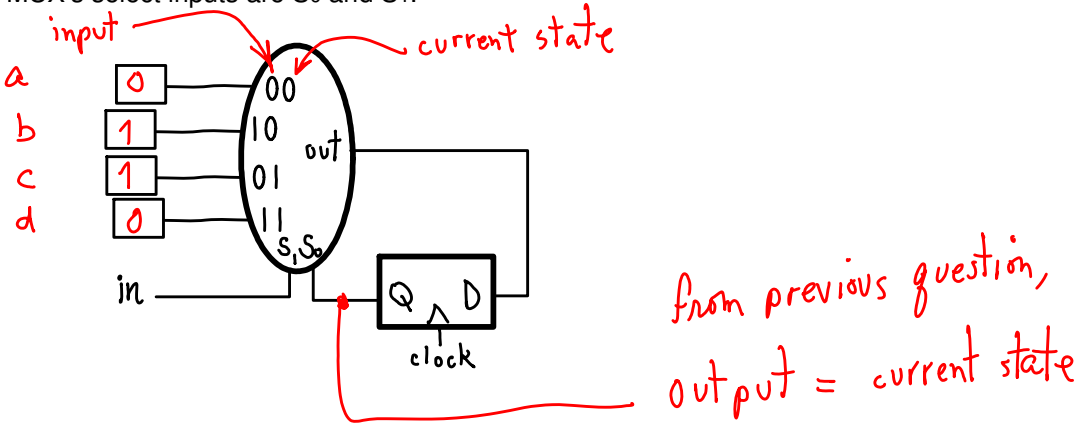
Question. How many states does M have, if M1 as a separate machine has i states, and M2 has j states?

Question. Below is the next-state and output functions for a FSM, F. It has two states, 1-bit input, and 1-bit output. We use 1-bit state encoding. Draw its state-transition diagram. Is F a Moore machine (output only depends on the current state), or a Mealy machine (output depends on both the current state and the current input)?

	input	current-state	next-state	output
a	0	0	0	0
b	1	0	1	0
c	0	1	1	1
d	1	1	0	1



Question. F could be implemented as a 1-bit state element and a ROM implementing the next-state function. This ROM can be implemented as a 1-bit, 4X1 MUX whose inputs are connected to either 0 or 1 as needed. In the diagram below, fill in the missing parts for this implemetation of F. Also, if it is possible, show how to connect a 1-bit output line. The MUX's select inputs are S_0 and S_1.



A Turing-universal language can describe any Turing Machine. There is a Turing-universal language whose only instruction is "Subtract-Then-Branch-If-LessEqualZero" (STBRILEZ):

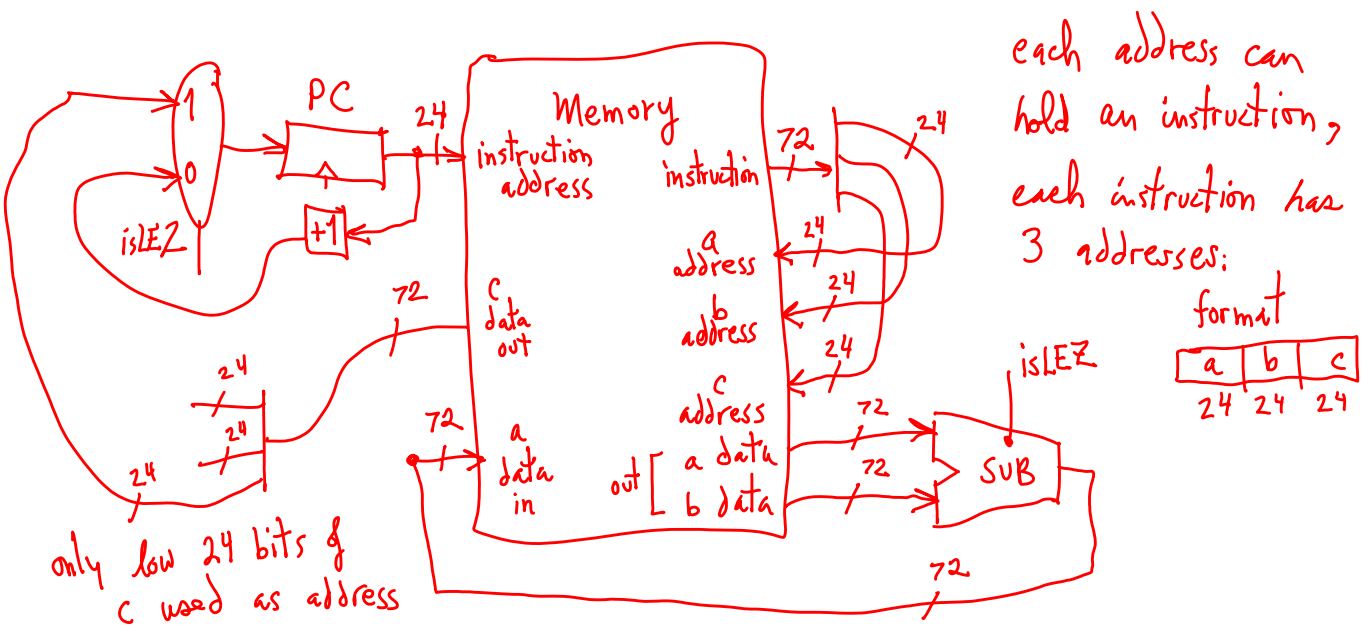
STBRILEZ a, b, c

where "a", "b", and "c" are memory addresses. Execution of this instruction goes as follows,

Mem[a] <=== (Mem[a] - Mem[b])
IF (subtraction result was less or equal 0) PC <=== c
IF (subtraction result was greater than 0) PC <=== PC+1

Unlike the LC3's ISA, this is not a load-store architecture. A 1-cycle STBRILEZ machine could be implemented using a 4-ported memory (1 read port for an instruction, 2 read ports for data, and one write port for data). It would execute an instruction in one cycle without the need for any data registers, nor an instruction register. The only register would be the PC. It would need only muxes and a few logic gates to implement control. Recall that the LC3's register file is essentially a 3-ported memory (it accepts 3 addresses simultaneously, for 1 write port and 2 read ports) with 3-bit addresses and 16-bit words.

Question. Draw a diagram of an STBRILEZ machine. Include a 4-ported memory, an ALU, a PC, and logic gates and MUXes as needed. Note that the instruction format does not include an opcode field because there is only one type of instruction. Assume memory addresses are 24 bits. Label all busses with their width in bits.



Question. Because STBRILEZ is universal, it must be possible to describe any Boolean function using it. Therefore, it must be possible to express Boolean operations using only subtraction. At a minumum, how many different Boolean operators must it be possible to express? Explain very roughly how it could be done, or why it cannot be done.

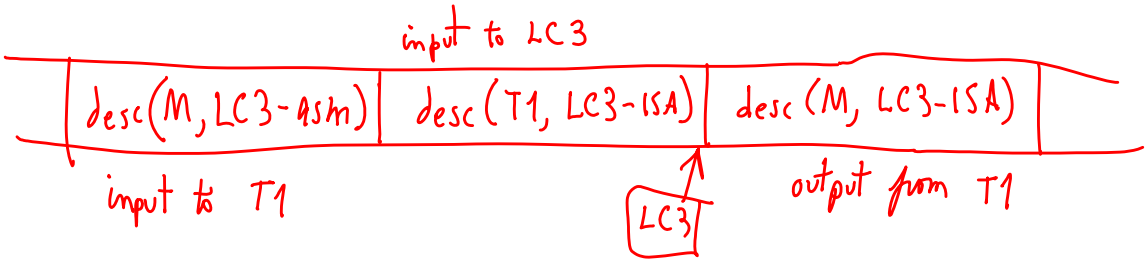
All Boolean logic can be expressed using {NAND} or {NOR} or {AND, OR, NOT}: 1 op. is sufficient to express any Boolean function. If we implement NAND, something like $((x-1)-(y-1))-(-1)$ maybe does it? This all SUB using variables and constant data where 0=F, 1=T.

Universal Turing machines are equivalent to computers, but UTMs and computers use different languages to describe machines. For example, the LC3 uses LC3 machine code, aka LC3-ISA. We have shown a unary encoding language for Turing machines, aka UTM-ISA, and argued that there is a UTM that understands it. The UTM will simulate the machine described to it. Using our terminology from homework, we can say,

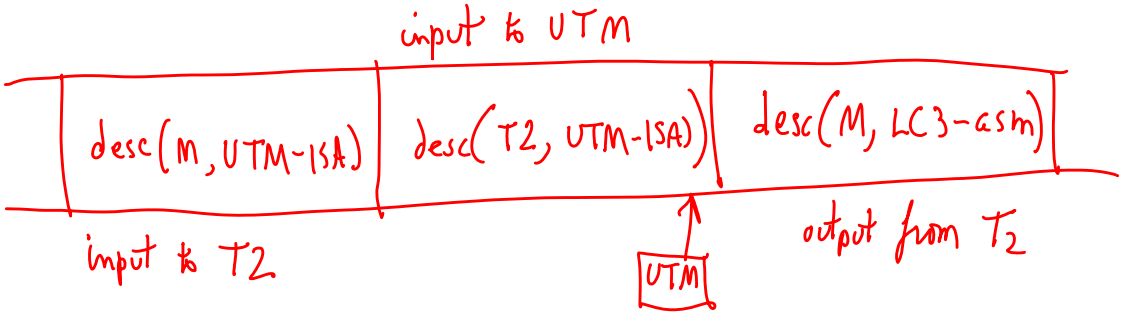
desc(M, LC3-ISA)	=	a program describing M written in the language, LC3-ISA.
desc(M, UTM-ISA)	=	a unary encoding of M written in the langauge, UTM-ISA.

A UTM simulates, and we can say LC3 also simulates. LC3 executes programs, and we can say a UTM also executes programs. We can also say the memory is tape and tape is memory.

Question. A translator, T1, translates programs written in LC3 assembly language, aka LC3-asm, into LC3-ISA machine code. We want a description of M as input for T1 to read. Show the LC3's memory as a TM tape, with appropriate inputs and T1's output. We can assume the LC3 starts by reading T1's description. Use the "desc()" notation shown above.



Question. Suppose we have T2, which translates desc(M, UTM-ISA) to desc(M, LC3-asm). Is it possible to use UTM for this purpose? In which language would we describe T2? Could we also use UTM for T2?



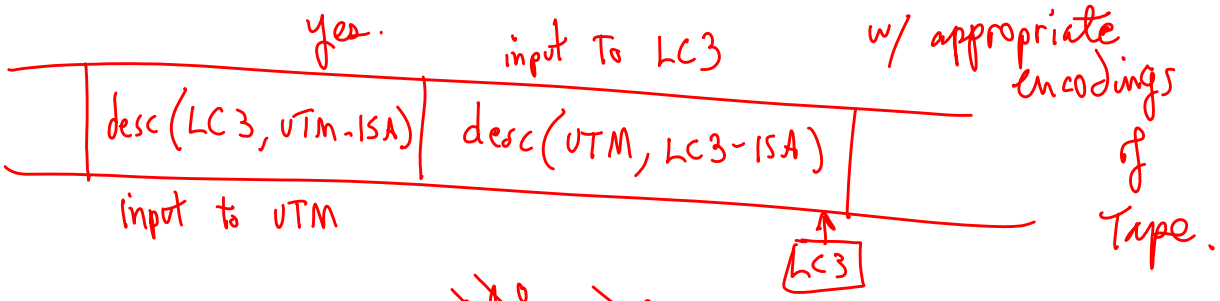
All desc(M, ·) are encoded symbols on UTM's simulated T2 tape.

UTM simulates T2, which translates from LC3-ISA to LC3-asm.
This is a dis-assembler.

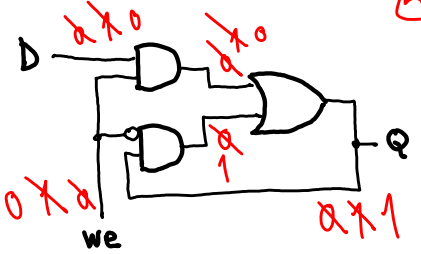
Question. Can we simultate UTM on an LC3? How about vice versa? What languages would we use in each case?

To simulate UTM on LC3, use desc(UTM, LC3-ISA) as input to LC3,
To simulate LC3 on UTM, use desc(LC3, UTM-ISA) as input to UTM.

Question. Can we simultate UTM simulating LC3 on an LC3? Show LC3's memory as a TM tape with appropriate descriptions.



Question. For the circuit right, fill in the missing values for the following sequence of signal changes:



1. D=0 we=0 Q=_0____
2. D=1 we=0 Q=_0____?
3. D=1 we=1 Q=_1____?
4. D=1 we=0 Q=_1____?
5. D=0 we=0 Q=_1____?

Question. Elmo looks at the above circuit and decides to describe its output as a boolean function. He begins by writing, "Q(D, we, Q) = ?". He starts to build a truth table for Q:

D	we	Q	||	Q	
0	0	0	||	0	
0	0	1	||	1	
1	1	0	||	?	(Q = Q?)

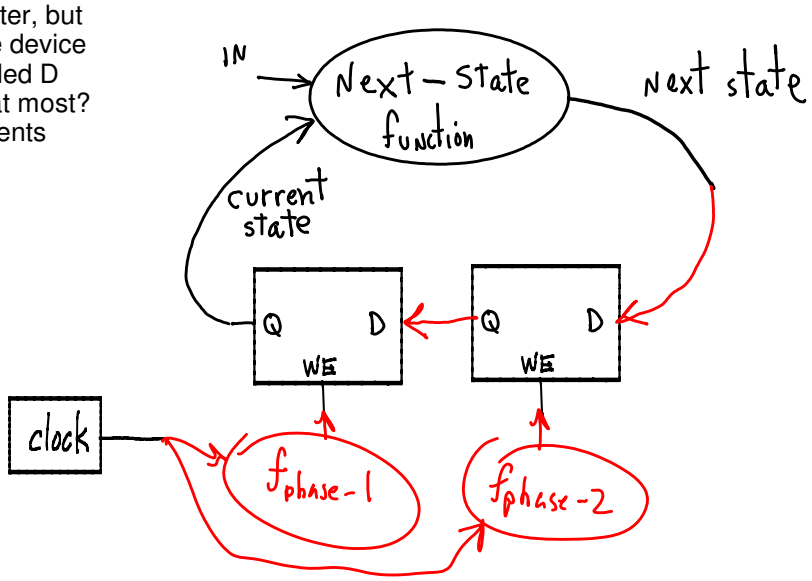
Explain why Elmo is confused. Can you build a correct truth table for this circuit?

output Q depends on itself, but at earlier time. Need to distinguish these: Q is current output, Q^+ is next output.

D	we	Q	Q^+	
0	0	0	0	latched
0	0	1	1	
0	1	0	0	transparent
0	1	1	0	

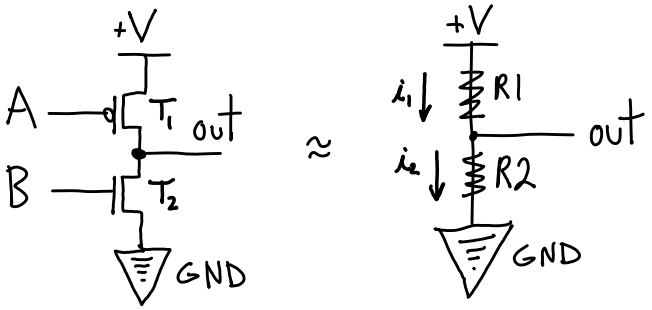
D	we	Q	Q^+	
1	0	0	0	latched
1	0	1	1	
1	1	0	1	transparent
1	1	1	1	

Question. Samantha has layed out part of an FSM circuit at right. She will implement the next-state function later, but asks that you finish up the circuit. She is using the device used in the previous two question as a write-enabled D latch. How many states does her machine have, at most? Fill in the missing connections and any logic elements needed. Why does she use two latches?

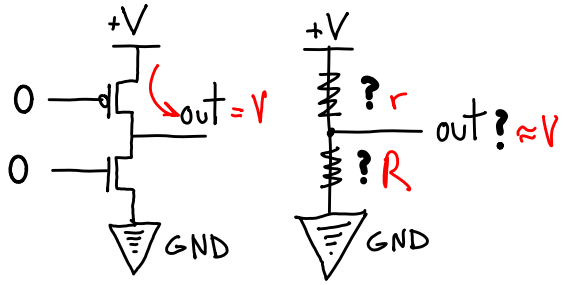


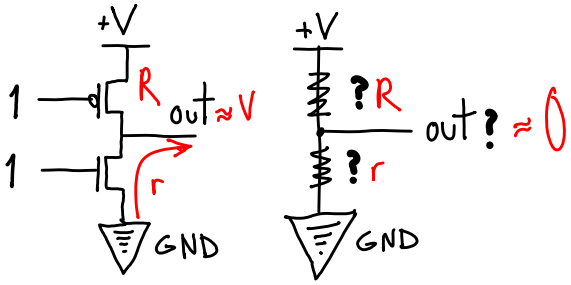
2-phase clocked flip flop is implemented w/ 2 latches. feedback path is controlled, state only changes w/ clock.

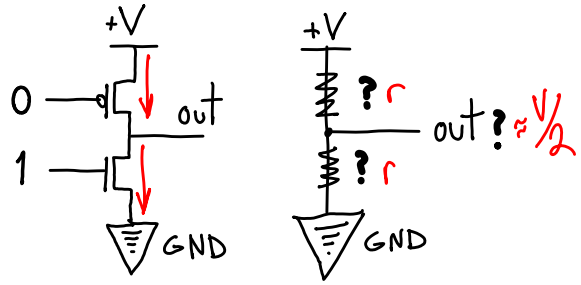
Question. Shown below is a CMOS circuit with two transistors, T1 and T2. For approximate realism, let's assume the transistors have a very small resistance, r, when conducting, and a very large resistance, R, when not conducting. Recall that V1 = i1 * R1 and V2 = i2 * R2 are the voltage differences across the resistances. For each case shown below, show values for each "**?**".

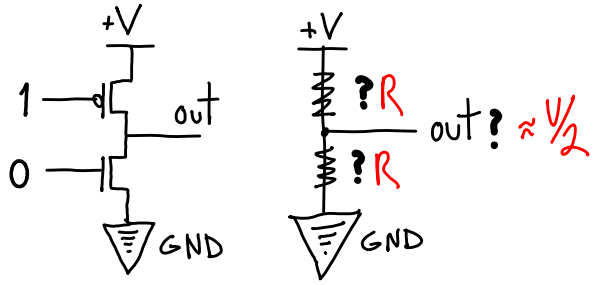


R1 is r or R, depending on A's value. Likewise for R2 and B.



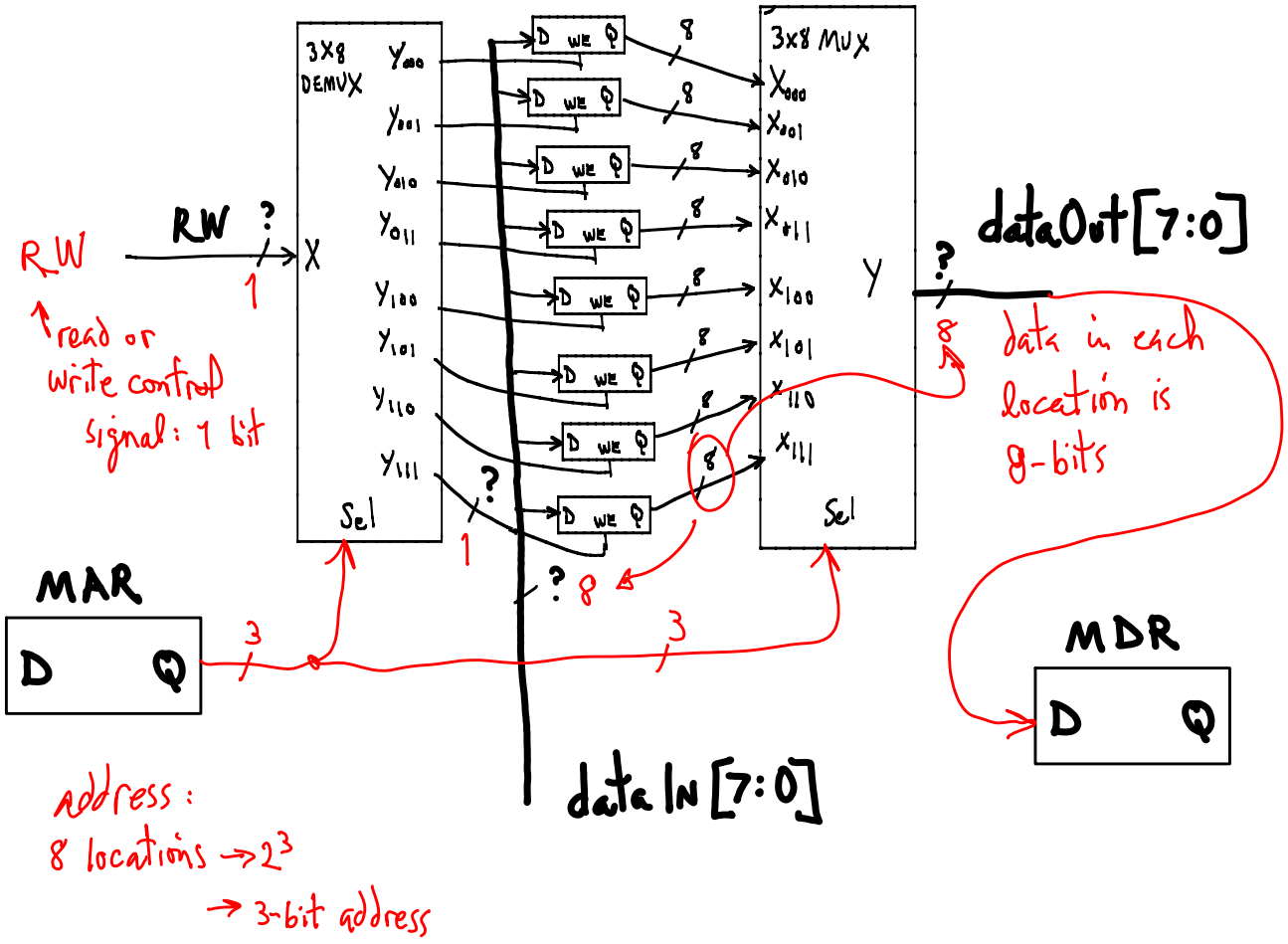






Question. For the circuit below, label signal sizes where a "?" appears, and draw in the missing wire/bus connections.

This circuit is a R/W random access memory (RAM).



Question. Where does the R_W signal come from?

from controller FSM output.

Section II. (write "True" or "False", circle the correct answer, or fill in the blank, as appropriate.)

Q. Which of the following is not a high level language?
Fortran
C++
Assembly language
COBOL

Q. A program can be translated into the ISA of a processor by means of
a compiler
an assembler
an interpreter
all of the above

maybe? depends on meaning of "translate"
in effect yes, directly, no.

Q. A and B are integers expressed in 2's complement. If both are positive, and their sum produces a result that is the representation of a negative number, we know we MUST have caused an overflow (T).

Q. The character "?" does not have an ASCII code. False

Q. The decimal digit 5 can be represented as which of these bit strings:

00000000000101 ← yes
000101 ← yes
0011010I ← no, bigger than 5

Q. Two values A = 11111111111111111111111010101 and B = 1111010101 are 2's complement representations of integers.

A is larger
B is larger
A and B are equal
You can not tell from the information provided.

Q. If a floating point number is represented in normalized form,
it must be a positive number. no
it must be a negative number. no
it must represent a number. no, ∞ NaN
it must be non-zero. no, zero is ok

Q. Using 8 bits, the unsigned integer representation of -13 is:

a. 10001101
b. 11110101
c. not possible to represent.

no sign info ⇒ cannot be < 0.

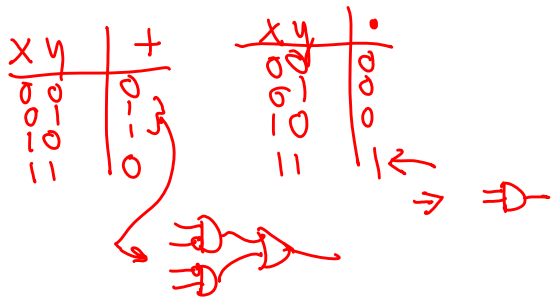
Q. Because we wish to allow each ASCII code to occupy one location in memory, most memories are BYTE addressable.

a. BYTE
b. NIBBLE
c. WORD (16 bits)
d. DOUBLEWORD (32 bits)

Q. Circuit A is a 1-bit adder; circuit B is a 1 bit multiplier.

a. Circuit A has more gates than circuit B
b. Circuit B has more gates than circuit A
c. Circuit A has the same number of gates as circuit B

(Hint: Construct the truth table for the adder and the multiplier)



Q. We say that a set of gates is logically complete if we can build any circuit without using any other kind of gates. Which of the following sets are logically complete

a. set of {AND,OR}
b. set of {EXOR, NOT}
c. set of {AND,OR,NOT}
d. None of the above

no, missing NOT
ok
hmm, not sure

$2^{12} = 2^2 \cdot 2^{10} = 4 \times 1024$

0000000000000
0000000000001
⋮
111111111111
12 13 = $(2^{12}-1)$

F A B C
0 true

14:13 3:2