Computer Science 126 — 11/22/1999

Second Midterm Exam with Answers — 7pm - 9pm

1. Number Representation [4]

Assume 8-bit signed two's complement for the hexadecimal representations in the following questions.

a) Convert 25 from decimal to hexadecimal.

25 / 16 = 1 + 9/16, so 25 = 0x19

b) Convert 0x25 from hexadecimal to decimal.

0x25 = 2*16 + 5 = 37

c) Convert -25 (note the negative sign) from decimal to hexadecimal.

-25 is 2^8 - 25 = 231, now convert 231 to hex as above, so the answer is 0xE7

d) Convert 0xE9 from hexadecimal to decimal.

Since the answer to the last question is 0xE7, which is "more negative" than 0xE9 by 2, the answer here is -23

2. TOY Machine Language Programming [5]

Assume that the following TOY program is loaded at 0x10 and it starts execution from there. Give the contents of the registers R1 and R2 in hexadecimal when the machine halts. (The TOY instruction set definition is provided in the appendix at the back of this exam.)

```
0x10 B101     r1 <- 0x01
0x11 B20A     r2 <- 0x0A
0X12 B301     r3 <- 0x01
0X13 1111     r1 <- r1 + r1
0X14 2223     r2 <- r2 - r3
0X15 7213     r2--; if (r2 != 0) goto 0x13
0X16 0000     halt
```

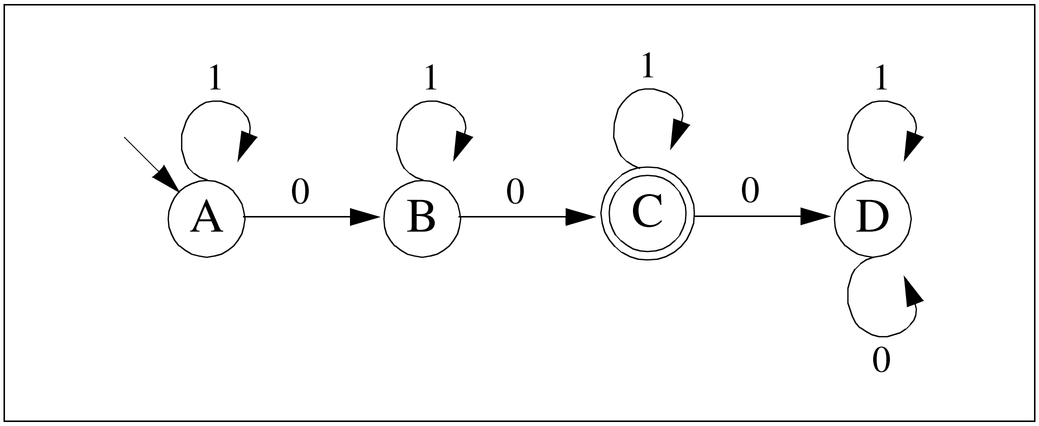
During each iteration, r2 gets decremented by 2, and r1 is doubled. So eventually r1 is doubled 5 times when r2 reaches 0.

R1 = 2^5 = 32 = 0x20, and R2 = 0

This is pretty much the same as the TOY exercise 3 on page 13 of the course packet.

3. Finite State Automata [4]

The machine shown below is started in state A. State C is the accept state.



a) Is the machine deterministic?

Yes. No two edges leaving the same state are labeled with the same alphabet.

b) What state does the machine stop in when given the string 010010?

It goes through the transitions: A, B, B, C, D, D, D. So the machine halts in D.

c) What state does the machine stop in when given the string 110110?

It goes through the transitions: A, A, A, B, B, B, C. So the machine halts in C.

d) Give the regular expression accepted by this machine.

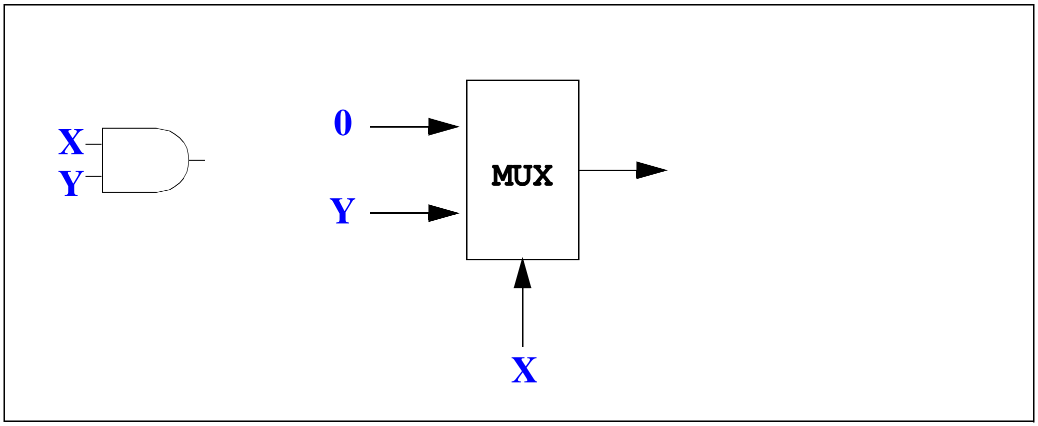
Any binary strings with exactly two 0s. So the answer is 1*01*01*.

This is pretty much the same as the FSA exercises 6-8 on page 29 of the course packet.

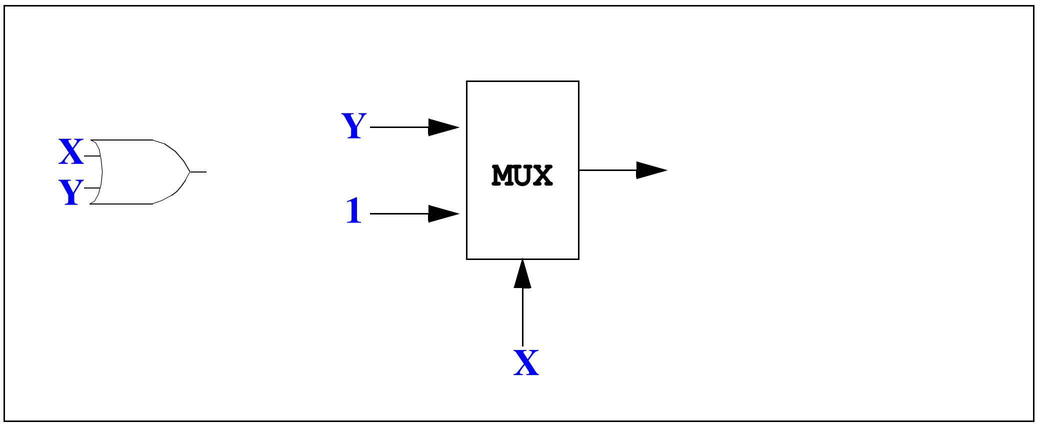
4. Logic Gates and Multiplexers [3]

Assume the availability of constant signals 0 and 1.

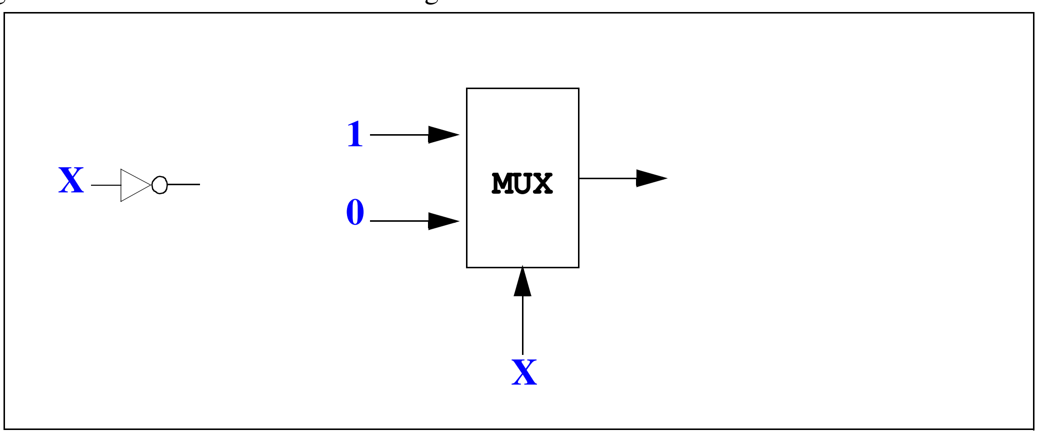
a) Use a single 2-to-1 MUX to construct a 2-input AND-gate:



b) Use a single 2-to-1 MUX to construct a 2-input OR-gate:

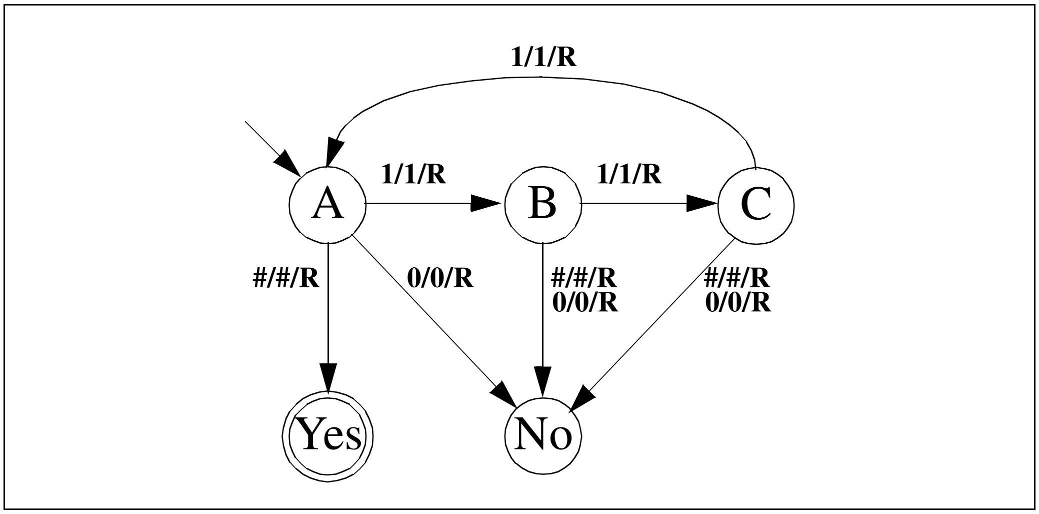


c) Use a single 2-to-1 MUX to construct a NOT-gate:



5. Turing Machines [5]

Consider the Turing machine below. The alphabet consists of three characters: “0”, “1”, and “#”. (The symbol “#” acts as a blank character and marks the two ends of a string.)



This is based on Turing machine exercise 8 on page 33 of course reader.

a) What state does the machine stop in when given the following string and initial read head position?

#	#	#	#	#	#	#	#	#	#

The machine goes through the transitions A and Yes. So the final state is Yes.

b) Answer the same question for the following string.

#	#	1	1	0	1	0	1	#	#

The machine goes through the transitions A, B, No, No,..., No. So the final state is No.

c) Answer the same question for the following string.

#	#	1	1	1	1	1	1	#	#

The machine goes through the transitions A, B, C, A, B, C, Yes. So the final state is Yes.

d) Does there exist a finite state automaton that can recognize the language accepted by this Turing Machine?

Yes.

A quick way of seeing this: this TM never overwrites the tape and only moves to the right. So it’s really an FSA in disguise! (For each label “X/Y/R”, you can “erase” the “/Y/R” to obtain the FSA.)

A slightly longer answer: the machine accepts strings that consist of only 1s and the number of 1s is a multiple of 3. This is the regular expression (111)*. Regular expressions are recognized by FSAs. So the answer is yes.

e) Does there exist a non-deterministic pushdown automaton that can recognize the language accepted by this Turing Machine?

NPDAs are more powerful than FSAs. So the answer is also yes.

6. Combinational Circuits and Two's Complement Arithmetic [6]

Build a combinational circuit that has two bits of input (X_1 and X_0) and three bits of output (Y_1, Y_0, and V).

• The two input bits (X_1X_0) are interpreted as a 2-bit two's complement integer.

• The first two output bits (Y_1Y_0) represent the result of subtracting 1 from the input integer (X_1X_0).

• The last output bit (V) is the overflow/underflow bit so it is 1 iff the addition results in an overflow or underflow condition. (Recall an overflow or underflow occurs when the answer does not fit in the given number of bits.)

a) Derive all the necessary truth tables.

X	Y	Overflow
0	-1	No
1	0	No
-2	-3	Yes
-1	-2	No

X_1X_0	Y_1Y_0	V
00	11	0
01	00	0
10	01	1
11	10	0

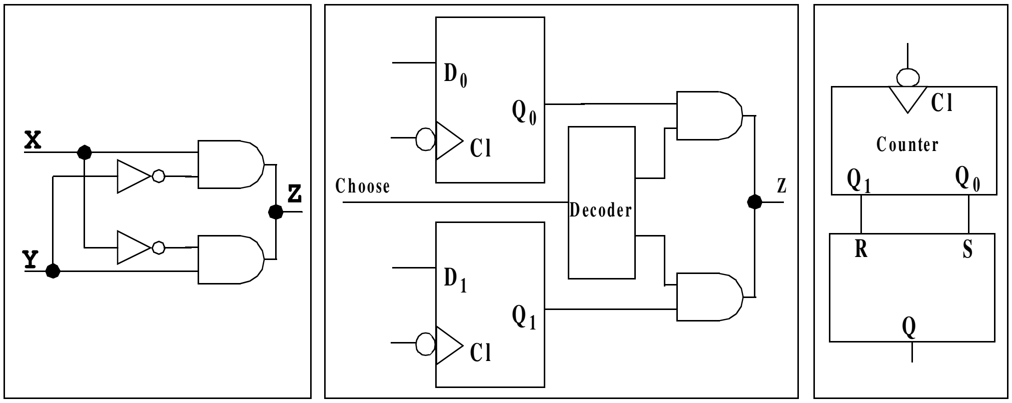
b) Derive all the output boolean expressions.

$Y_1 = X_1'X_0' + X_1X_0$

$Y_0 = X_1'X_0' + X_1X_0' = X_0'$

$V = X_1X_0'$

7. Debugging Circuits [5]



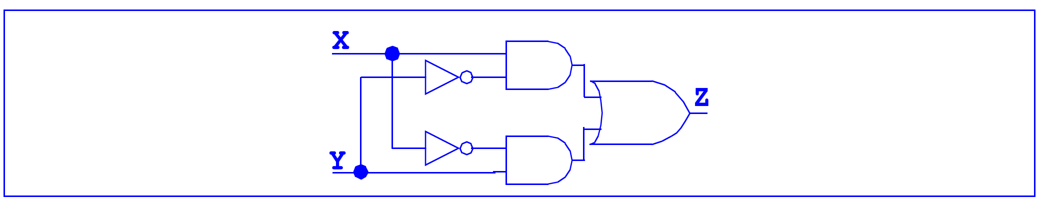
(a) (b) (c)

a) The circuit in Figure (a) is intended to implement a two-input parity function.

a.1) Explain in one sentence what is wrong with this circuit.

Can't connect gate outputs together.

a.2) Sketch how you would correct it.



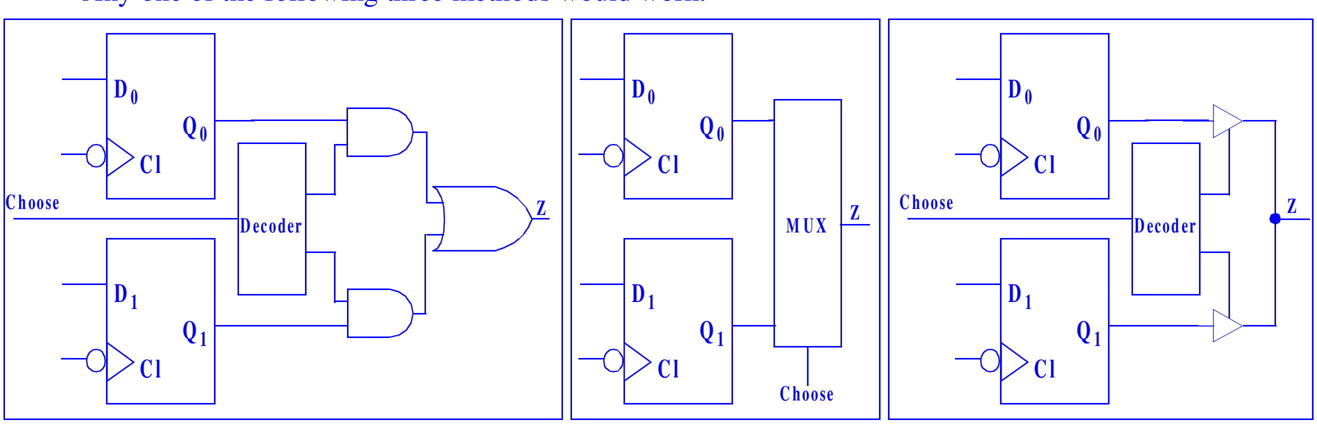
b) In Figure (b), the intended role of the signal labeled with "Choose" is to choose the content of one of the two D flip-flops as the output "Z".

a.1) Explain in a one sentence why the circuit does not work.

Can't connect outputs together, even when all but one is zero.

a.2) Sketch how you would correct it.

Any one of the following three methods would work.

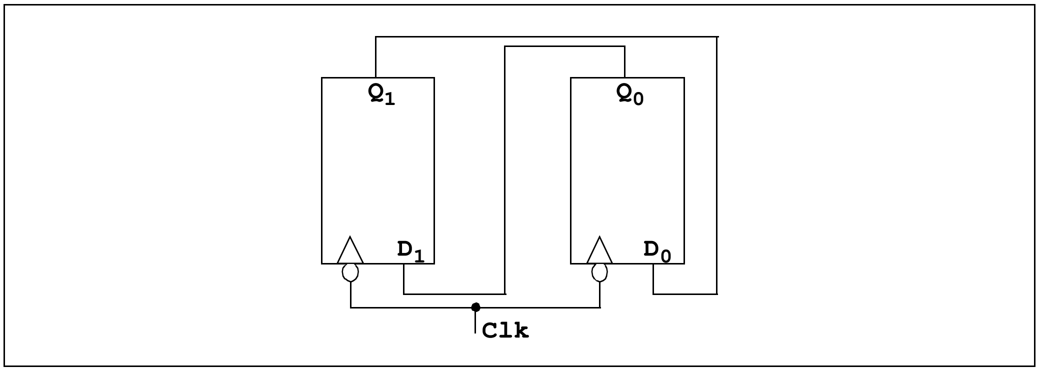


c) In Figure (c), the counter at the top generates a regular 2-bit pattern (Q_1Q_0) which is fed into an RS flip-flop at the bottom to generate a regular 1-bit pattern (Q). Explain in one sentence what problem this circuit has. (You do not need to correct it.)

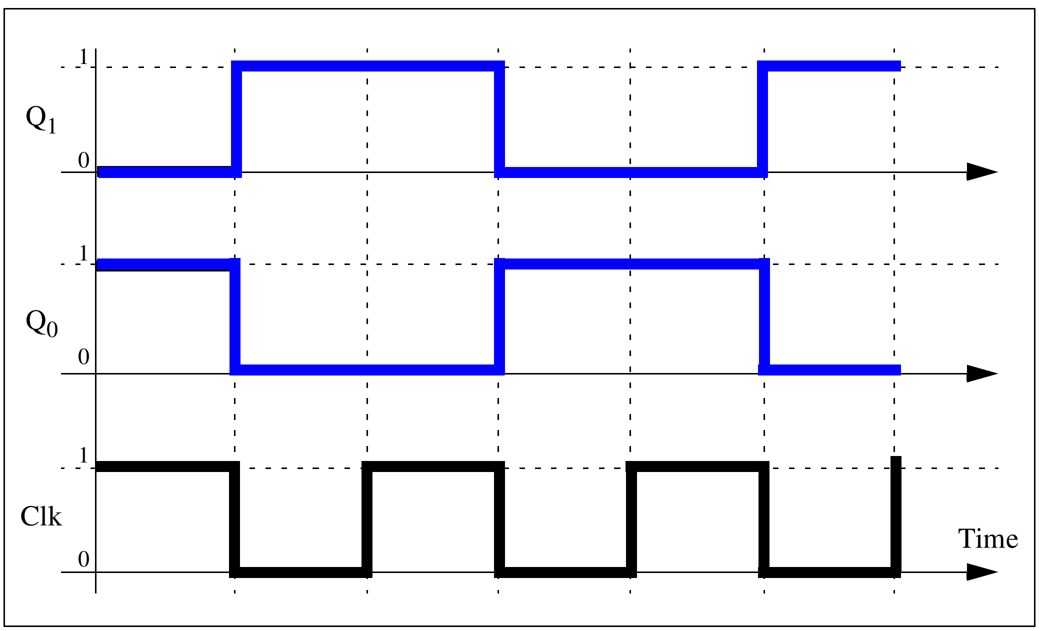
The counter outputs become Q_1=Q_0=R=S=1 at some point, an illegal input combination for the SR flip-flop.

Refer to my lecture slides for "A4: Sequential Circuits" for more details.

8. Sequential Circuits and Timing Diagrams [6]



Two negative edge-triggered D flip-flops are connected as shown in the figure above. Q_1 is initialized to 0 and Q_0 is initialized to 1. Fill in the timing diagram below for these two signals. (You may assume that the delays introduced by the flip-flops and wires are negligible.)



Each of the two flip-flops "remembers" its input (which is the output of the other flip-flop) during the half cycle when the clock value is 1. It then changes its output to this "remembered" value on the negative clock edge.

This is a simplified version of the architecture review exercise 11.

9. TOY Architecture [6]

Assume that each of the following operations in the TOY datapath takes the stated amount of time (in nanoseconds):

	Operation	Time (ns)
1	**fetching an instruction from memory**	**4**
2	**reading from the register file**	**2**
3	**executing any ALU operation**	**2**
4	**loading data from memory**	**4**
5	**storing data into memory**	**5**
6	**writing the result back into the register file**	**2**
7	**all other delays (those of MUXes, controls, wires, etc.)**	**0**

(Each of these operations must finish within a single clock cycle.)

a) How much time (in nanoseconds) does each of the following instructions need to complete?
  a.1) load (from memory) using indexed addressing;

instruction fetch + register file read + add + memory load + register file write
= 4 + 2 + 2 + 4 + 2
= 14 ns

  a.2) store (to memory) using indexed addressing

instruction fetch + register file read + add + memory store
= 4 + 2 + 2 + 5
= 13 ns

b)
  b.1) What is the minimum cycle time for a single cycle design?

The longest instruction delay, namely, that of the load instruction, 14 ns.

  b.2) What is the minimum cycle time for a multicycle design?

The slowest operation in the table, namely, that of the memory store operation, 5 ns.

c) Suppose we speed up the 4th operation in the table (loading data from memory). Instead of the 4 ns, it now takes 2 ns. (All other times remain the same, including the time required to fetch an instruction from memory.)
  c.1) What is the minimum cycle time for a single cycle design?

Now the store instruction takes longer than the load instruction, so the cycle time will have to be at least the delay of the store instruction, which is 13 ns.

c.2) What is the minimum cycle time for a multicycle design?

The store operation is still the slowest. So the minimum cycle time for the multicycle design is unchanged from b.2); it's still 5 ns.

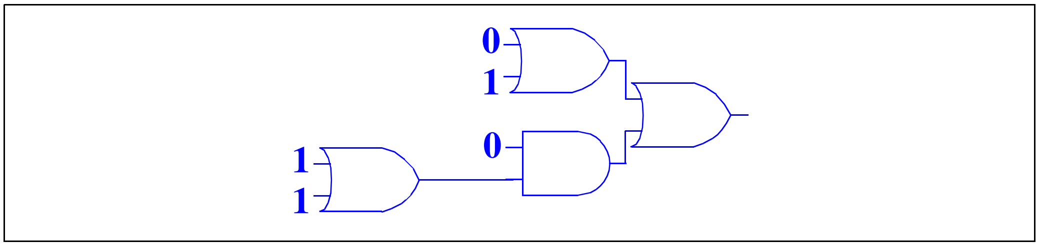
This is a variation of the architecture review exercises 17-19.

10. A Cumulative Question: Circuits, ADT, and Regular Expressions [6]

You have learned postfix notation; we put operators *after* operands using this notation. We can also use postfix notation to write boolean expressions. Suppose the AND operator is represented by "*", and the OR operator is represented by "+".

a) Draw the circuit represented by the postfix expression "`0 1 + 0 1 1 + * +`".

The more familiar infix notation is (0+1) + 0 * (1 + 1).



b) The following program evaluates postfix boolean expressions using a stack.

```
#include <stdio.h>
#include "stack.h"

main (int argc, char *argv[]) {
    char *a = argv[1];
    int i, n = strlen(a);

    stackInit();
    for (i = 0; i < n; i++) {
        if (a[i] == '+') stackPush(stackPop() | stackPop());
        if (a[i] == '*') stackPush(stackPop() & stackPop());
        if (a[i] == '0') stackPush(0);
        if (a[i] == '1') stackPush(1);
    }
    printf ("%d\n", stackPop());
}
```

If we evaluate "`0 1 + 0 1 1 + * +`" using this code, at the moment when the stack is fullest, what is the stack content?

Trace the execution of this code. After the first plus operator is evaluated, a single 1 is on the stack. Right before the second plus operator is evaluated, the stack reaches its tallest height, and the content from bottom to top is “1, 0, 1, 1”.

c) Are boolean expressions regular expressions? Justify your conclusion with one sentence. (Hint: think about expressions like “1 1 1 1 + + +”.)

No. In order to recognize boolean expressions such as the one in the hint, a machine must be able to count the number of 1s and +s, a capability that an FSA does not have.

This is very much like slide T1-13 in the course packet. Few people gave the right reasoning. We give full credit to all those who got the “Yes/No” part right.

Appendix. TOY Instruction Set

INSTRUCTION FORMATS	
Format 1: opcode, r0, r1, and r2 Format 2: opcode, r0, and 8-bit addr Indexed addressing (for format 2): if leading bit of r0 digit is 1, then addr = r1 + r2	
TRANSFER between registers and memory	
9: load A: store B: load address	r0 <- mem[addr] mem[addr] <- r0 r0 <- addr
ARITHMETIC operations	
1: add 2: subtract 3: multiply	r0 <- r1 + r2 r0 <- r1 - r2 r0 <- r1 * r2
LOGICAL operations	
C: xor D: and E: shift right F: shift left	r0 <- r1 ^ r2 r0 <- r1 & r2 r0 <- r0 >> addr r0 <- r0 << addr
CONTROL	

0: halt
4: system call
5: jump
6: jump if positive
7: jump and count
 	
8: jump and link
	halt
	print r0 on tty
	pc <- addr
	if (r0 > 0) pc <- addr
	r0--
	if (r0 != 0) pc <- addr
	r0 <- pc
	pc <- addr

Feel free to tear out this sheet.