ECE 198JL Third Midterm Exam
Spring 2014

Tuesday, April 15th, 2014

Name: ______________________________ NetID: ______________

Discussion Section:

9:00 AM	[ ] JD9	[ ] JDA
10:00 AM	[ ] JD1	
11:00 AM	[ ] JD2	[ ] JDB
12:00 PM	[ ] JD5	
1:00 PM	[ ] JD6	[ ] JD7
2:00 PM	[ ] JD3	
3:00 PM	[ ] JD8	
4:00 PM	[ ] JD4	[ ] JDC

- **Be sure your exam booklet has 14 pages.**
- **Be sure to write your name and lab section on the first page.**
- **Do not tear the exam booklet apart; you can only detach the last page.**
- **We have provided LC-3 instructions set at the back.**
- **Use backs of pages for scratch work if needed.**
- **This is a closed book exam. You may not use a calculator.**
- **You are allowed one handwritten 8.5 x 11" sheet of notes.**
- **Absolutely no interaction between students is allowed.**
- **Be sure to clearly indicate any assumptions that you make.**
- **Don't panic, and good luck!**

Problem 1	10 points:	____________
Problem 2	14 points:	____________
Problem 3	13 points:	____________
Problem 4	14 points:	____________
Problem 5	12 points:	____________
Problem 6	12 points:	____________
Problem 7	10 points:	____________
Problem 8	5 points:	____________
Problem 9	10 points:	____________
Total	100 points:	____________

Problem 1 (10 points): End-of-Transmission detection

ASCII contains several control characters that indicate when and how ASCII data is transmitted and interpreted. The End-Of-Transmission (EOT) character is encoded with the 8-bit hexadecimal string x04. EOT indicates that the transmission of a file is complete. Design a sequence recognizer finite state machine (FSM) that detects whether an incoming bit sequence is the EOT character. Your FSM will receive two inputs (a_i and T) and produce one output f. a_i is a serial transmission of ASCII data. The eight bits of ASCII data are transmitted **beginning with the least significant bit of data**. Input T is 1 for exactly eight clock cycles in a row before returning to 0. When T is 1, the data on a_i should be treated as ASCII data. When T is 0, the data on a_i should be ignored. The output should be 1 only when T is 0 and the most recent ASCII character was the EOT character.

Example inputs and output sequence.

Inputs
```
     ai = 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0 0
     T  = 0 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0
```
Output
```
     f  = 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
```

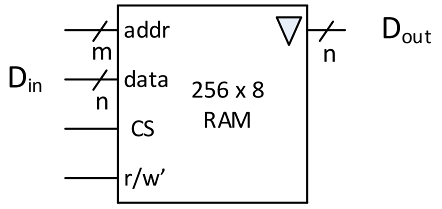
1. Draw the FSM diagram that detects the EOT character. FSMs with more than 8 states will not be graded. Your output should be a function of only the state of your FSM. Hint: How can the T input simplify your FSM? (8 pts)

2. How many flip-flops will you need to implement this sequence recognizer? (2 pts)

Answer: __________________

Problem 2 (14 pts): RAM

Shown to the right is a 256 x 8 RAM.

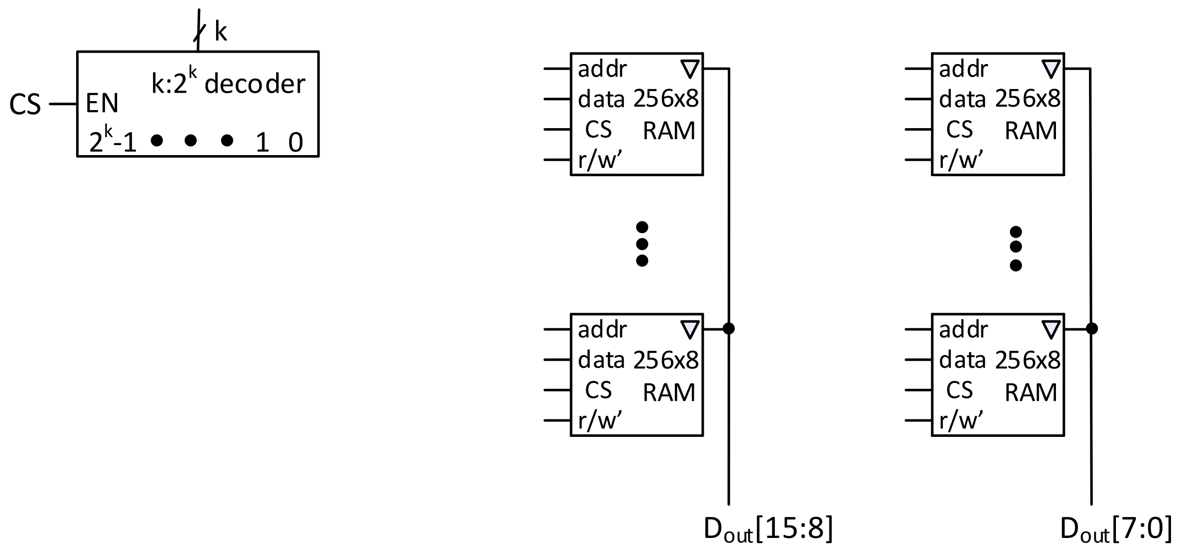


1. How many bits do the **addr**, **data**, **CS**, and **r/w'** ports require? (4 pts)

Answer: addr: __________ CS: _____

data: __________ r/w': _____

2. Using 256 x 8 RAM chips, implement a 4K x 16 RAM (1K = 2^{10}). Each 256 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. (5 pts)



3. How many rows of 256 x 8 RAM chips are needed and what is the value of *k*? (2 pts)

Number of rows: ________________ *k* = ______________

4. Suppose you wish to store the decimal value 96 in memory at the address 1050. In which row (indexing from 0) of 256 x 8 RAM chip(s) will this value be stored? (1 pt)

Answer: _____________________________

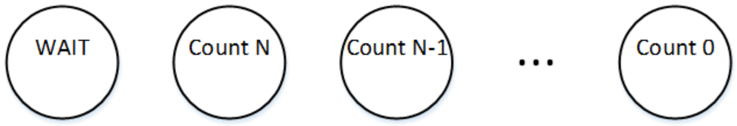
5. What input values must be provided to your 4K x 16 RAM in order to properly store 96 at the address 1050? Write the **addr** and **data** values for your 4K x 16 RAM in **<u>hexadecimal</u>**. (2 pts)

addr: _________________________________ data: _________________________________

Problem 3 (13 pts): FSM design

Mobile computing devices such as smart phones and tablets are increasingly using a single button that performs different functions. In this problem, you will design a hardware device driver that is composed of two finite state machines (FSMs): a control counter and a FSM that distinguishes between long presses and short presses. Both FSMs will be controlled by a system button B which sends a 1 as long as it is pressed.

1. Complete the FSM diagram below by adding labeled state transitions and circuit outputs for every state. The control counter should start counting when the system button is pressed and continue counting until the system button is released or the count reaches 0. Its output (Z) should be 1 only when the count reaches 0 and remain 1 until the system button is released. The output of the counter should be a function of only the current state. (3 pts)



2. Design the FSM that distinguishes between long button presses and short button presses. The FSM has two inputs: Z from the control counter and the system button B. The FSM has two outputs: long (L) and short (S). L should only be one for exactly one clock cycle after the system button is released after a long button press (as counted by the control counter). S should only be one for exactly one clock cycle after the system button is released after a short button press. Solutions with more than 6 states will not be graded. The names of your state must describe the behavior or meaning of the state. The output of the FSM should be a function of only the current state. (10 pts)

Problem 4 (14 pts): Counter Design

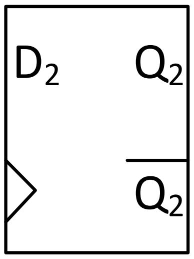
A three-bit binary counter has one control input, C. When C = 0 the counter counts up through the even numbers, i.e. 0 ->2 -> 4 ->6 ->0. When C = 1, the counter counts down through the odd numbers 0->7->5->3->1->0->7. When the control input changes, the counter starts the corresponding count sequence beginning at zero.

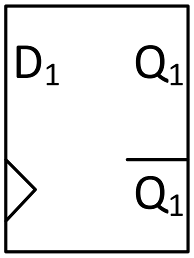
a) Draw the FSM state transition diagram for this counter. The current state's encoding should encode the count value of your counter. (8 pts)

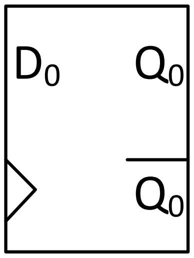
b) Write the next-state table for the FSM (3 pts)

Current State			Input	Next State		
Q2	Q1	Q0	C	Q2+	Q1+	Q0+
0	0	0	0			
0	0	0	1			
0	0	1	0			
0	0	1	1			
0	1	0	0			
0	1	0	1			
0	1	1	0			
0	1	1	1			
1	0	0	0			
1	0	0	1			
1	0	1	0			
1	0	1	1			
1	1	0	0			
1	1	0	1			
1	1	1	0			
1	1	1	1			

c) Using the D flip flops shown in the figure, implement the counter. Please draw short wires with variable names attached rather than drawing long wires. (3 pts)

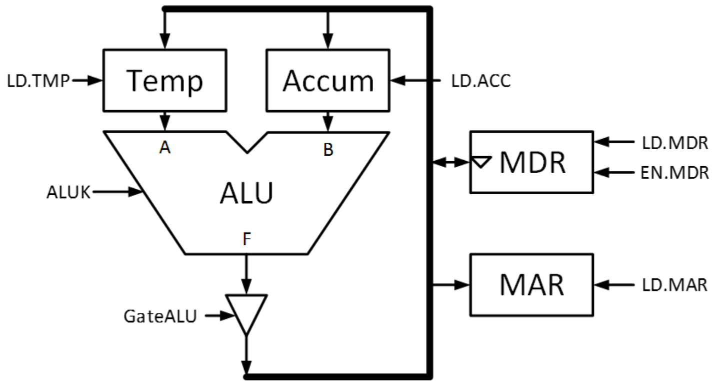






Problem 5 (12 pts): Register transfer

Digital Signal Processing units, like those found in your cell phone, are specialized architectures that use an Accumulator register (Accum) to store the results of all arithmetic operations. The architecture also has a Temporary register (Temp) to store a second operand. The ALU, Memory Data Register (MDR), and Memory Address Register (MAR) are used for the same purpose as in the LC-3. The ALU function table is shown below. The ALU and MDR can both send data to the system bus but only as allowed by the GateALU and EN.MDR signals, respectively. When EN.MDR is 1, MDR's content is output on the bus. When LD.MDR is 1, MDR stores value from the bus. When GateALU is 1, ALU's output is connected to the bus.



ALUK	Operation	Explanation
00	Pass A	F = A
01	Pass B	F = B
10	Multiply-Accumulate	F = A*B + B
11	Clear	F = 0

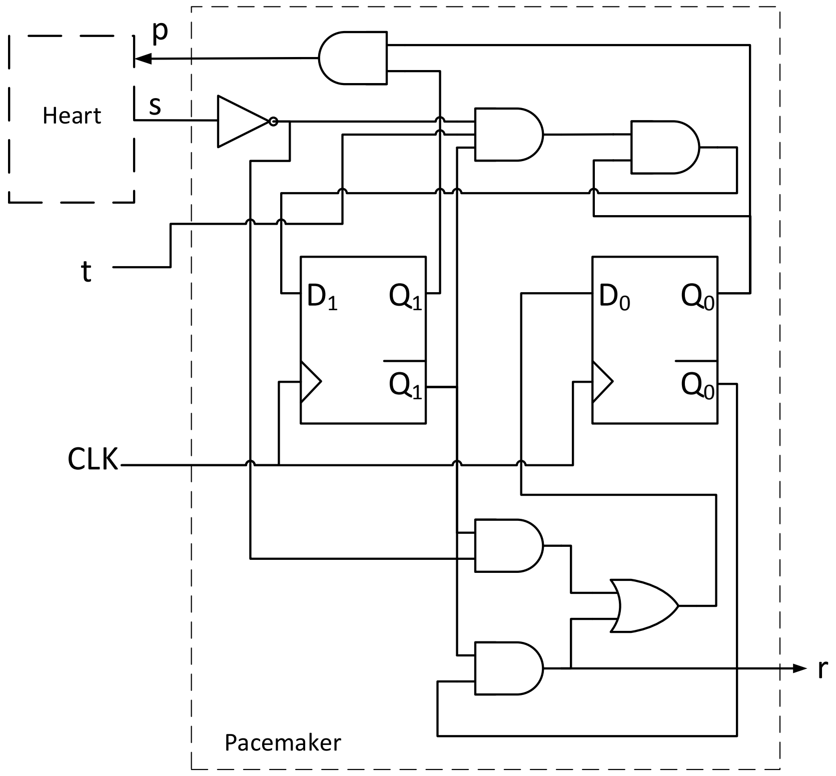
1. The ALU is part of the ____________________ unit of a von Neumann architecture. (2 pts)

2. Assign values to the control signals to execute the following RTL instructions. If the RTL instruction cannot be implemented, write "IMPOSSIBLE." For the last row, determine what RTL instruction is implemented by the selected control signals. (10 pts)

RTL Instruction	CONTROL SIGNALS						
	LD.TMP	LD.ACC	LD.MDR	LD.MAR	GateALU	EN.MDR	ALUK
MDR ← Temp							
Accum ← Temp * Accum + Accum							
MAR ← MDR							
Temp ← MAR							
	1	1	0	0	0	1	11

Problem 6 (12 pts): FSM Reverse Engineering Problem

Below is a simplified block diagram of a heart pacemaker. The output p from the pacemaker pulses high if the heart does not contract within a certain time. The input s indicates whether the heart has contracted (s = 1) or not (s = 0). The input t comes from a timer which counts for the expected time between contractions (approximately 1 second). When t = 1, the timer has counted up to 1 second, and the heart should have contracted. If the heart has not contracted within that time, the pacemaker sends a pulse p = 1. The output r is 1 to reset the timer.



a) Write the Boolean expressions for output p and r and the flip-flop inputs D1 and D0. (4 pts)

p(Q1, Q0) = ___________________________________

r(Q1, Q0) = ___________________________________

D1(Q1, Q0, s, t) = ___________________________________

D0(Q1, Q0, s, t) = ___________________________________

b) Write the next-state table for the pacemaker's sequential circuit. (4 pts)

Current state	External inputs	Next state	Outputs

c) Based on your next-state table, draw the FSM for the pacemaker circuit. (4 pts)

Problem 7 (10 pts): LC-3 Instructions and ISA

a) Listed below are the states of the registers of the LC-3 at two times during the execution of a single instruction. Given the information provided, fill in all 16 bits of the instruction that is being executed in the table below. (8 pts)

The instruction being executed is stored in location x3000 in memory. The first state is just after the instruction has been fetched and stored in the IR. The second state is at the end of the execution of the instruction, before the next instruction is fetched.

Note: Some data has been intentionally left out.

Register	After Fetch	At End
PC	x3001	x3001
MAR	x3000	x4020
MDR	x6???	x0000
R0	x5000	x0000
R1	x4000	x4000
R2	x3000	x3000
R3	x2000	x2000
R4	x1000	x1000
R5	x4040	x4040
R6	x5050	x5050
R7	x6060	x6060

Instruction bits:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

b) Give the RTL operations corresponding to the instruction you identified in (a). (2 pts)

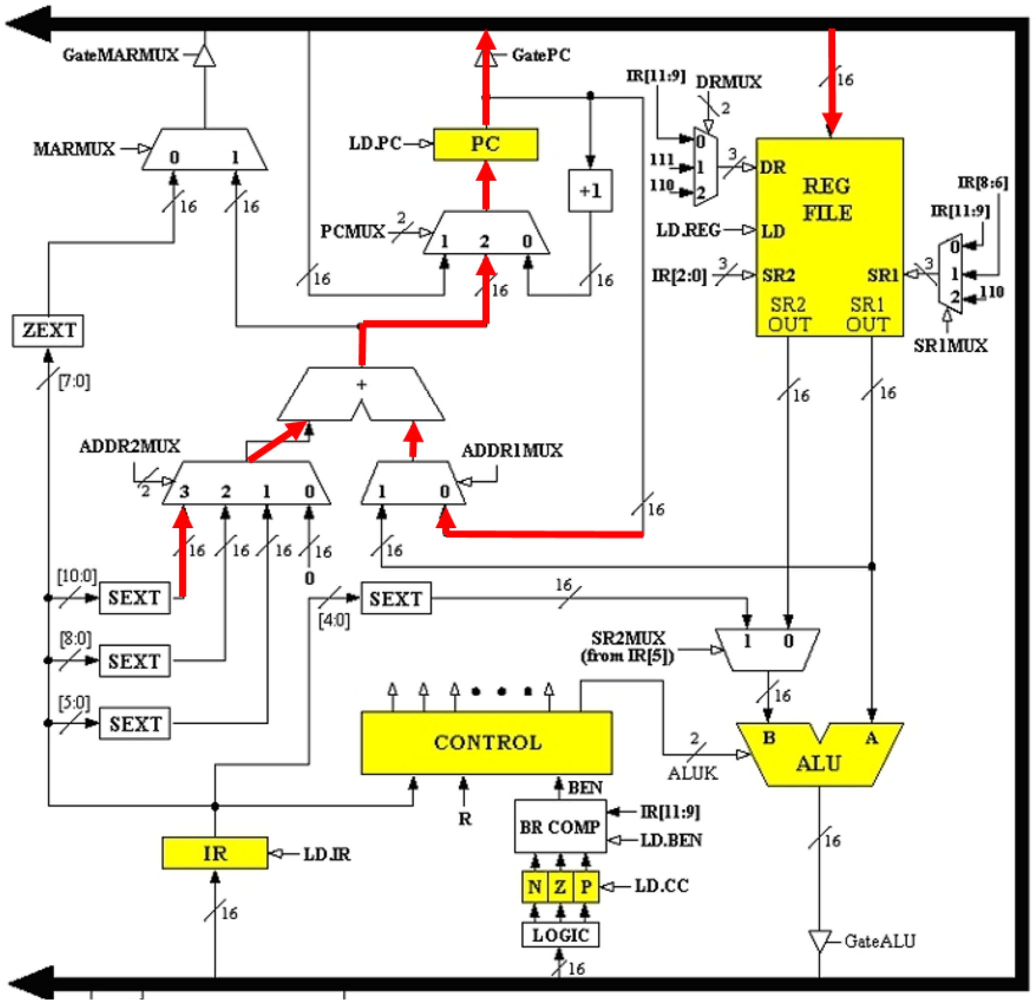
Problem 8 (5 pts): LC-3 Instruction types

a) During one state, a mystery LC-3 instruction sets the values of PCMUX, ADDR1MUX and ADDR2MUX as shown below. The effect of setting the values of these MUXes is also highlighted in the LC3 microarchitecture figure below. (3 pts)

PCMUX: 10
ADDR1MUX: 0
ADDR2MUX: 11

Write the opcode and instruction name of the mystery instruction.

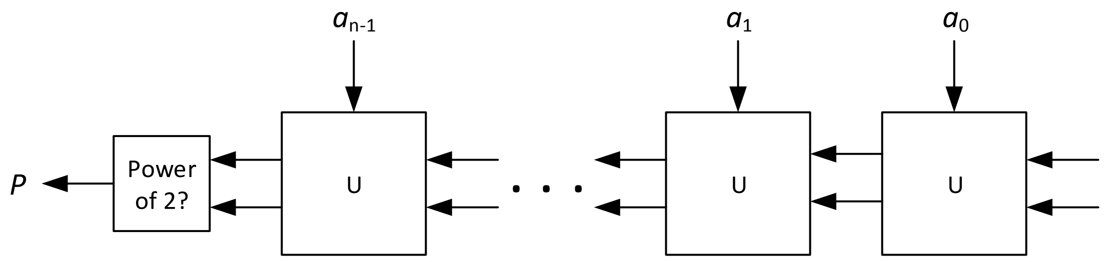
Opcode: ________________________ Instruction name: ____________________



b) What type of instruction is the mystery instruction? (circle one) (2 pts)

Operate	Data Movement	Control

Problem 9 (10 points): Bit-serial design



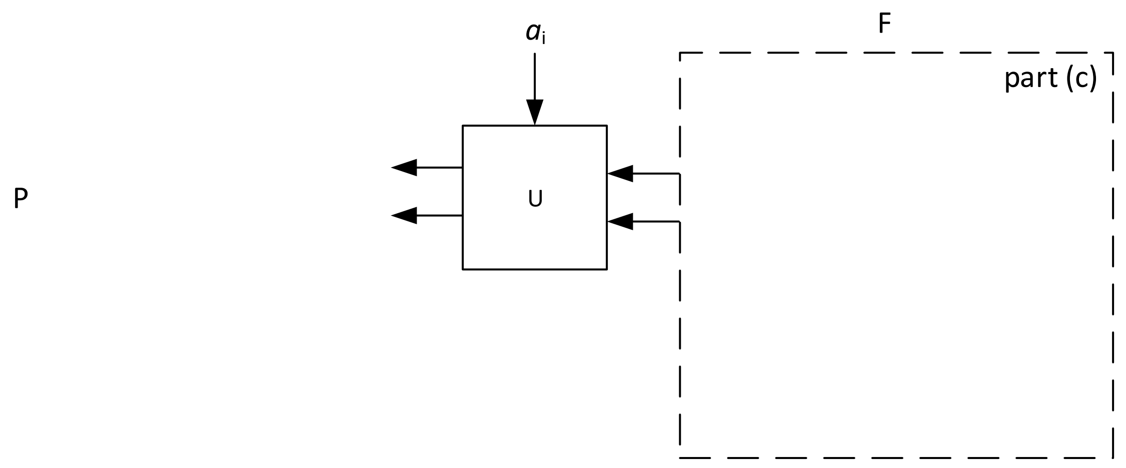
Suppose that we had previously built the *combinational bit-sliced* power of 2 checker above. It outputs *P*=1 if and only if an unsigned integer $A=a_{n-1}a_{n-2}..a_1a_0$ is a power of 2, starting with the least significant bit. Circuit U processes one bit-slice of the input A. **Your task is to design a *sequential bit-serial circuit* that performs an equivalent operation using circuit U as part of your circuit design.**

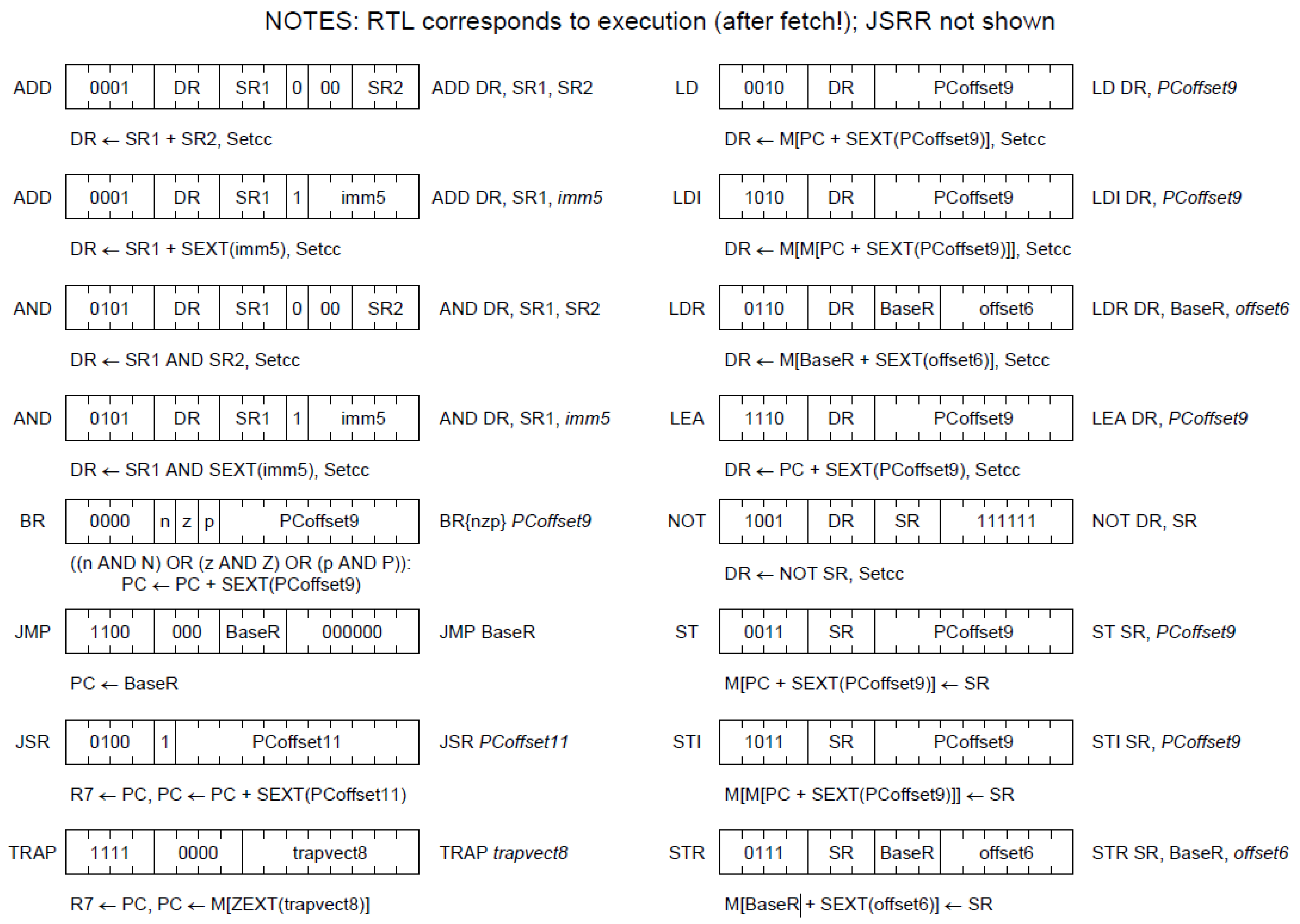
a) Design a finite state machine that implements the behavior of a *sequential bit-serial* power of 2 detection circuit. Your FSM will receive one bit of A (a_i) as an input per clock cycle, starting with the least significant bit. Circuit U will serve as the next-state logic of your FSM in parts b and c. Your states must use names that clearly define their behavior. You must also assign binary state encodings to your states. You must indicate the desired start state for your FSM. Your output *P* must be a function of only the current state. (4 pts)

Implement parts b and c together to implement the bit-serial circuit. Circuit U has been added to the diagram below to help you get started. Place the circuitry for part (b) outside the dotted box and the circuitry for part (c) inside the dotted box.

b) Circuit U has been designed to exactly implement the next-state logic of your FSM from part a. **Add all additional components and logic that you need to implement your entire FSM.** (3 pts)

c) Your bit-serial circuit must also respond to one more input signal *First (F)*. *F*=1 iff your circuit is starting a new power of 2 check. Add additional circuitry in the dotted box to indicate how your FSM will respond to this new input. (3 pts)





NOTES: RTL corresponds to execution (after fetch!); JSRR not shown

ADD	0001 · DR · SR1 · 0 · 00 · SR2	ADD DR, SR1, SR2	DR ← SR1 + SR2, Setcc
ADD	0001 · DR · SR1 · 1 · imm5	ADD DR, SR1, *imm5*	DR ← SR1 + SEXT(imm5), Setcc
AND	0101 · DR · SR1 · 0 · 00 · SR2	AND DR, SR1, SR2	DR ← SR1 AND SR2, Setcc
AND	0101 · DR · SR1 · 1 · imm5	AND DR, SR1, *imm5*	DR ← SR1 AND SEXT(imm5), Setcc
BR	0000 · n · z · p · PCoffset9	BR{nzp} *PCoffset9*	((n AND N) OR (z AND Z) OR (p AND P)): PC ← PC + SEXT(PCoffset9)
JMP	1100 · 000 · BaseR · 000000	JMP BaseR	PC ← BaseR
JSR	0100 · 1 · PCoffset11	JSR *PCoffset11*	R7 ← PC, PC ← PC + SEXT(PCoffset11)
TRAP	1111 · 0000 · trapvect8	TRAP *trapvect8*	R7 ← PC, PC ← M[ZEXT(trapvect8)]
LD	0010 · DR · PCoffset9	LD DR, *PCoffset9*	DR ← M[PC + SEXT(PCoffset9)], Setcc
LDI	1010 · DR · PCoffset9	LDI DR, *PCoffset9*	DR ← M[M[PC + SEXT(PCoffset9)]], Setcc
LDR	0110 · DR · BaseR · offset6	LDR DR, BaseR, *offset6*	DR ← M[BaseR + SEXT(offset6)], Setcc
LEA	1110 · DR · PCoffset9	LEA DR, *PCoffset9*	DR ← PC + SEXT(PCoffset9), Setcc
NOT	1001 · DR · SR · 111111	NOT DR, SR	DR ← NOT SR, Setcc
ST	0011 · SR · PCoffset9	ST SR, *PCoffset9*	M[PC + SEXT(PCoffset9)] ← SR
STI	1011 · SR · PCoffset9	STI SR, *PCoffset9*	M[M[PC + SEXT(PCoffset9)]] ← SR
STR	0111 · SR · BaseR · offset6	STR SR, BaseR, *offset6*	M[BaseR + SEXT(offset6)] ← SR