ECE120 Final Review - Cramming Carnival

Author: Members of HKN

Fall 2024

1 Bitwise Bonanza

(a) **Twos Complement Sign Extension**: Given the binary numbers in two's complement, give the respective value as a 8-bit two's complement representation (If given a 5-bit number, write the same value with 8-bits).

(i) 010100
(ii) 1010100
(iii) 0001010
(iv) 0100011
(v) 111000
(vi) 10
(vii) 00
(viii) 01
(ix) 1

(b) **Converting to Hexadecimal Notation**: Given the binary number below, convert to hexadecimal (Sign extend if necessary).

(i) 1010100111010000
(ii) 0101001111000001
(iii) 1111011110001010
(iv) 0100001110100101

(c) **Converting From Unsigned Representation**: Given the 5-bit unsigned binary string, what is its decimal equivalent?

(i) 10101
(ii) 01010
(iii) 11110
(iv) 11011

(d) **Converting from IEEE 754 32-bit Floating Point**: Given the IEEE 754 32-bit Floating Point, what is its decimal equivalent?

(i) 1 10000000 00010000000000000000000
(ii) 0 01111111 10010000000000000000000
(iii) 1 10000011 11000100000000000000000

(e) **Converting from Hexadecimal Notation**: Given the hexadecimal notation below, what is the binary representation?

(i) 109F

(ii) 74BC

(iii) E235

(iv) 86AD

(f) **Representation Integer Range**: What is the minimum number of bits needed to represent the values (which are in decimal) below?

(i) 343

(ii) 1709

(iii) -123

(iv) -1

(v) 0

(g) **Representing a Number Using Two's Complement Representation**: Give the 6-bit two's complement representation of the following decimal values.

(i) -9

(ii) -4

(iii) 5

(iv) 7

(v) -16

(h) **Converting to IEEE 754 32-bit Floating Point**: Enter the IEEE 754 32-bit floating point representation of each of the following decimal numbers.

(i) $6\frac{15}{16}$

(ii) $4\frac{1}{8}$

(iii) $7\frac{3}{4}$

(i) **Two's Complement Subtraction**: These numbers are in two's complement representation. Subtract the numbers.

(i) 101001 - 111111

(ii) 001101 - 001010

(iii) 110010 - 110110

(iv) 111100 - 000110

(j) **Two's Complement Addition Overflow**: For the following binary values, which of them have an overflow in 2's complement?

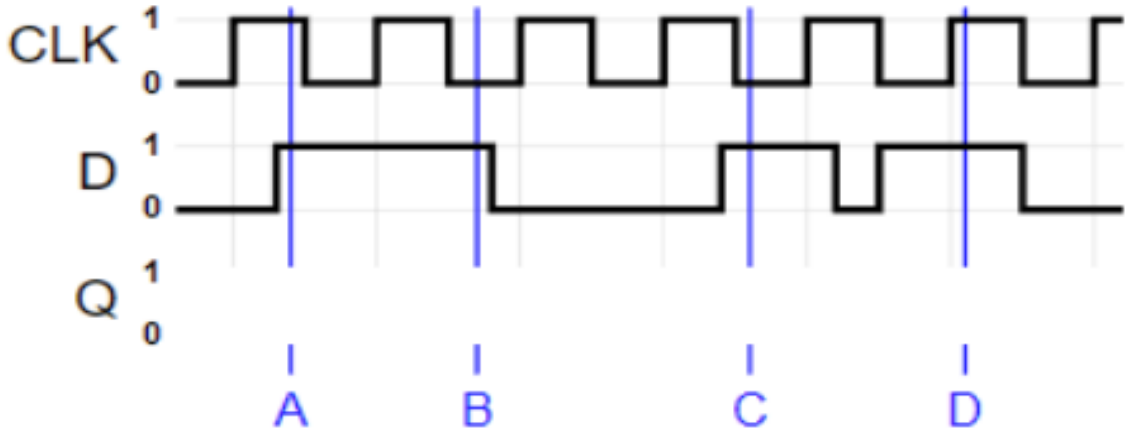
(i) 110001 + 100011

(ii) 100100 + 001010

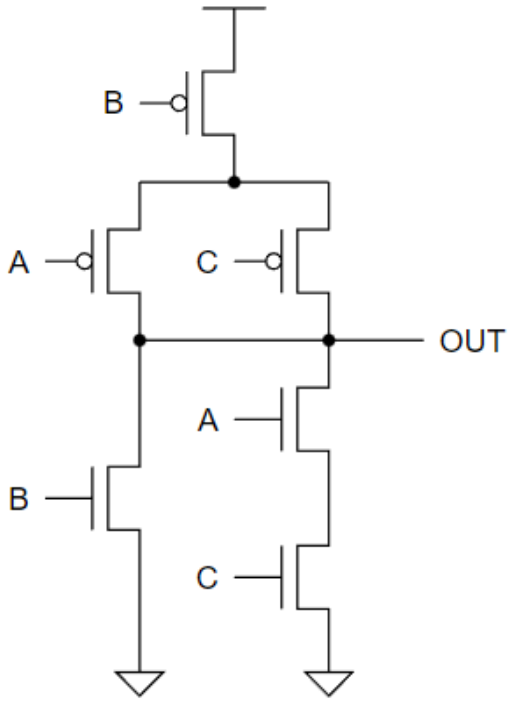
(iii) 110010 + 111111

2 Thrilling Toggles and Terrific Truth Tables

(a) **D Flip-Flop Timing Diagram**: Given this image below of a timing diagram for a positive edge-triggered D flip-flop, sketch the values for Q at each time.



(b) **CMOS Gate Truth Table (3 Variables)**: For the CMOS gate shown below, enter in the truth table output.



A	B	C	Out
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

(c) **Boolean Expression to Truth Table**: Fill in the truth table given by the functions below.

(i) F(A,B,C,D) = (A+B'+C)'D'
(ii) G(A,B,C,D) = (A+B')(CD+A)'
(iii) H(A,B,C,D) = (CB)+A'D'B+B'CD
(iv) I(A,B,C,D) = ACD+AB+(BC)'

Inputs				Outputs			
A	B	C	D	F	G	H	I
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				

(d) **Truth Table to SOP and POS**: Using the truth table below, express outputs F1 and F2 as a minimal sum of products (SOP) and a minimal product of sums (POS).

A	B	C	D	F1	F2
0	0	0	0	1	1
0	0	0	1	1	0
0	0	1	0	0	0
0	0	1	1	x	1
0	1	0	0	0	1
0	1	0	1	0	0
0	1	1	0	1	1
0	1	1	1	x	x
1	0	0	0	1	0
1	0	0	1	0	0
1	0	1	0	0	1
1	0	1	1	1	1
1	1	0	0	0	x
1	1	0	1	1	1
1	1	1	0	0	1
1	1	1	1	x	0

F1 (SOP) =

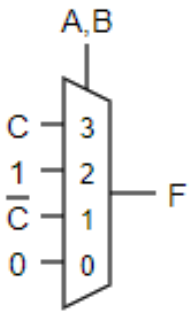
F1 (POS) =

F2 (SOP) =

F2 (POS) =

3 Multiplexer Mania

Multiplexer to Boolean Formula: Give the minimal SOP for F for the multiplexer below (express F in terms of the inputs).



F(A,B,C) =

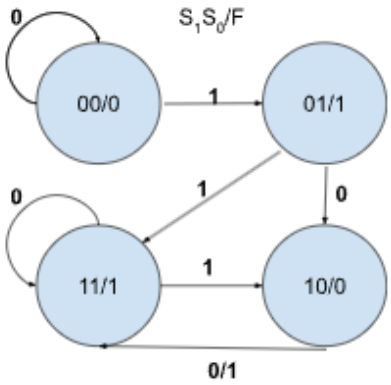
4 Funky FSMs

(a) **FSM Design Application**: Given the FSM below:

(i) Find the K-Maps for S1+,S0+, and F.

(ii) Give their minimal SOP Boolean logic expressions.

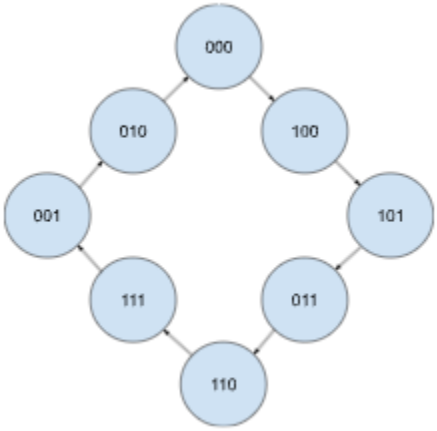
(iii) Implement the FSM as D flip-flops and gates.



(b) **FSM Counter**: Given the FSM below:

(i) Find the K-Maps for S2+,S1+, and S0+.

(ii) Give their minimal SOP Boolean logic expressions.



(c) **FSM Sequence Detector**: Generate an FSM that detects the sequence 001:
Input sequence M (starting from left to right): 100110100101
Output sequence R (Starting from left to right): 00010000010

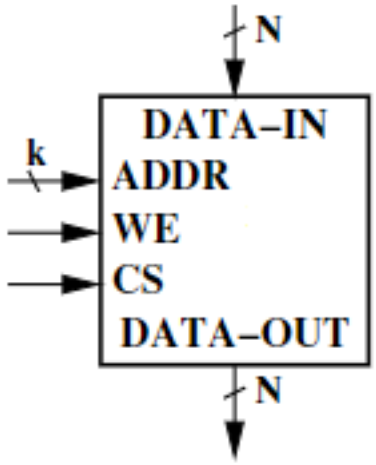
(i) State declaration table

(ii) The K-Maps for S1+,S0+, and F

(iii) Give their minimal SOP Boolean logic expressions

5 Rambunctious RAM

RAM Concepts: Use the RAM Module Below to answer the questions:



(a) If there are k ADDR wires and N DATA-IN wires, what size memory does this chip store? (Hint: AxB)

(b) After entering the chip and before being input to a cell, what do the k wires enter?

(c) If given four 4x4 RAM blocks and any decoder possible, what sizes of RAM could be made?

(d) In a 16x8 RAM do the following tasks

(i) Write 0x12 into address 0x578A

(ii) Write 0x72 into address 0xECEB

(iii) Read from address 0x893F

6 Legendary LC3

(a) **LC-3 bits to RTL**: For each of the following 16-bit sequences, give the RTL equivalent:

(i) 0001 1000 0100 0100

(ii) 0101 0000 0011 1111

(iii) 0000 1101 1111 1100

(iv) 0010 1000 0000 0111

(v) 1001 1010 0011 1111

(b) **LC-3 FSM Concepts (Von Neumann model)**: Answer the conceptual questions below:

(i) What are the first 3 FSM instructions the LC-3 executes with each line of code in memory?

(ii) If the ALUK value is 00 what is the opcode of the line currently being performed?

(iii) How many gates (that go to BUS) can be active at a time?

(iv) Which opcodes set CC?

(v) In the von Neumann model, where is the program stored?

(vi) How many steps are there at minimum in an instruction cycle?

(c) **LC-3 Programming**: Write the following functions in RTL (For each part, assume your first line starts at x3000 and all registers are x3000. Each part is independent of the others.)

(i) Given a value in x3005, set its 2's complement in x3006

(ii) Get an input from the keyboard and store it at x3030 (GETC = x20)

(iii) Given two values in x3004 and x3005 store their sum in x3006

(iv) Given two values in x3004 and x3005 store their product in x3006

(v) Using only register R1, and not accessing any other register, determine if the value stored in R2 is a power of 2 (you may also edit the value in R2)

(d) **String Programming**: After the following code executes, what will be printed to the console? Assume that the data in the table has already been loaded into memory when the code is run.

```
.ORIG x3000
LD R0, STRING
PUTS
STRING
.FILL 0x3047
.END
```

Address	Data	ASCII Representation
x3046	x0021	'!'
x3047	x0048	'H'
x3048	x0069	'i'
x3049	x0000	
x304A	x0074	't'
x304B	x0068	'h'
x304C	x0065	'e'
x304D	x0072	'r'
x304E	x0065	'e'
x304F	x0021	'!'
x3050	x0000	

(i) How many memory locations will the following string use?
.STRINGZ "AMONG US"

(e) **Executing ALU Instructions**: The following sequence of instructions is executed with the initial register values shown below.

x3000: 0001000001000011
x3001: 0101011011000001
x3002: 0101001000100000
R0: xA25B R1: x1B2A R2: x4875 R3: x6227

(i) Give the contents of the following registers after the instruction sequence above finishes executing. Convert your answers to hexadecimal.

(f) **LC3 Missing Instruction**

(i) An LC-3 computer starts with the following register and memory contents. In addition, the word at address x3000, which is not shown, contains an instruction, one of ADD, AND, BR, LD, LDI, LDR, LEA, NOT, ST, STI, or STR. Note: The LEA instruction does not change the CC register (3rd edition behavior).

Registers	**Memory**	
R0: x619F		
R4: x48AA		
R1: x4295	x3001: x4333	
R5: x49FC	x3002: x1BDC	
R2: x259D		
R6: x54C2		
R3: x4000	x4000: x2911	
R7: xF914	x4001: x68A8	
PC: x3000		
CC: b001		

Table 1: Before: Registers and Memory

All other memory locations start with x0000. After the instruction at address x3000 is executed, the contents of registers and memory are:

Registers	**Memory**	
R0: x619F		
R4: x48AA		
R1: x4295	x3001: x4333	
R5: x49FC	x3002: x1BDC	
R2: x259D		
R6: x48B8		
R3: x4000	x4000: x2911	
R7: xF914	x4001: x68A8	
PC: x3001		
CC: b001		

Table 2: After: Registers and Memory

What value could be stored at address x3000? Give your answer in hexadecimal.

x3000 =

(ii) An LC-3 computer starts with the following register and memory contents. In addition, the word at address x3000, which is not shown, contains an instruction, one of ADD, AND, BR, LD, LDI, LDR, LEA, NOT, ST, STI, or STR. Note: The LEA instruction does not change the CC register (3rd edition behavior).

Registers	**Memory**	
R0: x619F		
R4: x48AA		
R1: x4295	x3001: x4333	
R5: x49FC	x3002: x1BDC	
R2: x259D		
R6: x54C2		
R3: x4000	x4000: x2911	
R7: xF914	x4001: x619F	
PC: x3000		
CC: b001		

Table 3: Before: Registers and Memory

All other memory locations start with x0000. After the instruction at address x3000 is executed, the contents of registers and memory are:

Registers	**Memory**	
R0: x619F		
R4: x48AA		
R1: x4295	x3001: x4333	
R5: x49FC	x3002: x1BDC	
R2: x259D		
R6: x48B8		
R3: x4000	x4000: x2911	
R7: xF914	x4001: x68A8	
PC: x3001		
CC: b001		

Table 4: After: Registers and Memory

What value could be stored at address x3000? Give your answer in hexadecimal.

x3000 =

7 Fortuitous FSMs

Note: You may get different answers for this section as these problems are design heavy and are based on your understanding of the problem

(a) **Self Looping FSMs** In our daily lives, FSMs are used everywhere. Sometimes we do not realize it but almost everything around us is a FSM. One such example is a Sewage lift station that helps to manage wastewater generated by us. The lift station works by starting a pump that pumps out wastewater from the sludge tank when wastewater reaches the upper limit of the tank. The lift station will continue to pump out the wastewater till the water level is lower than the minimum water level in the tank. Given 2 signals **U for the upper limit (U=1 when water reaches upper limit)** and **L for the lower limit (L=0 when water level is lower than the lower limit)**, design the FSM for the lift station

(i) Draw the state transition diagram for this FSM. Your output should be the signal P, where P=1 indicates the pump should be running

(ii) Create K-maps for next state and output of the FSM

(iii) Write the next state expressions for the FSM

(iv) Implement this FSM using gates, Combinational Logic Elements, Flip-Flops, RAMs etc

8 Liberating Loading

(a) A memory address, pointing to the start of an array is stored at x30AD. Which load instruction type would you use to access the first array value? Write the single LC-3 assembly instruction to accomplish this when the PC is at x3090.

(b) What state numbers of the LC-3 FSM are used for a LDR instruction? Which of these states are unique only to LDR?

(c) Fill in the following table with the correct control signals on the LC-3 data path for the LDR Instruction. Note: This table doesn't have all the states for LDR.

States	**18**	**25**	**27**
LD.BEN			
LD.MAR			
LD.MDR			
LD.IR			
LD.PC			
LD.REG			
LD.CC			
GateMARMux			
GateMDR			
GateALU			
GatePC			
MARMUX			
PCMUX			
ADDR1MUX			
ADDR2MUX			
DRMUX			
SR1MUX			
ALUK			
MIO.EN			
R.W			

9 Manic Multiplying

You are going to complete a program that multiplies two numbers from memory together and saves the result back into memory. Below are the memory locations for the inputs and output. While you don't have to specify a .ORIG instruction for this problem, it is relevant that the program begins at x3000.

Meaning	**Address**
Multiplicand 1	x3FFE
Multiplicand 2	x3FFF
Product	x4000

(a) Load Multiplicand 1 and Multiplicand 2 into R1 and R2, respectively. This will require 2 LC-3 instruction lines and 2 pre-processor instruction lines (ie .STRINGZ, .FILL, etc).

(b) Create a loop to multiply the multiplicands together and place the result in R3. You are allowed to modify R1 and R2 as needed.

(c) Store R3 into the product memory location. this will require 1 LC-3 instruction line and 1 pre-processor instruction line.

10 Disorienting Decoding

(a) In the LC-3, if the IRD control signal is set to 1, what happens to the contents of the instruction register (IR) during the decoding process? Specifically, how are the bits IR[15:12] utilized?

(b) Given a situation where COND is set to 001, which represents checking for a positive condition code (P), and the instruction's execution previously set the P flag to 1, what would be the value of BEN (branch enable)?

(c) Suppose the current instruction has an opcode of 1101 in IR[15:12], corresponding to a LEA instruction. If the J bits in the microinstruction are set to 011111, explain how the next state of the microsequencer is determined.

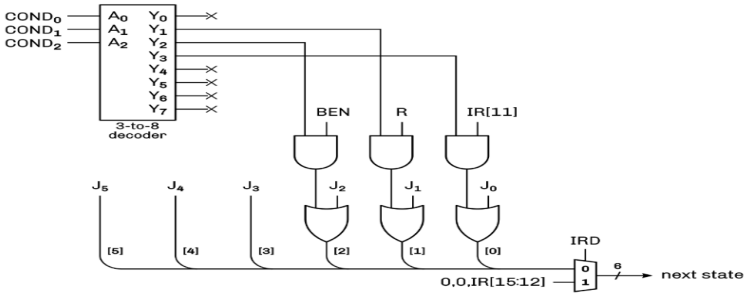


Figure 1: LC-3 Microsequencer

11 Conspicuous Control Signals

(a) Complete the following table by entering values (0, 1, or X) for the LC-3 microinstructions at ROM addresses 4 and 7

ROM Address	IRD	COND(3)	J(6)	LD.BEN	LD.MAR	LD.MDR	LD.IR	LD.PC	LD.REG	LD.CC	GateMARMUX	GateMDR	GateALU	GatePC	MARMUX	PCMUX(2)	ADDR1MUX	ADDR2MUX(2)	DRMUX(2)	SR1MUX(2)	ALUK(2)	MIO.EN	R.W
4																							
7																							

(b) If GateALU=1, what should the other Gate signals be?

12 Novel New Instructions

We want to implement the following instruction:
PC ⟵ PC + M[BaseR + PCOffset6] + 1

Note: the +1 is to accommodate state 18.
Note 2: In the very rare case that you actually read the textbook, pretend Appendix C.6.3 did not exist.

(a) We know that LC-3 has one unused opcode, the opcode 1101. When this opcode is decoded, suppose that the LC-3 FSM enters state 48 (110000). How many FSM states will this instruction need? What will each state do? Assign them in the 48-55 range.

(b) How will each instruction transition to each other state? Draw a state diagram.

(c) Using K-maps or another method, write an expression for the next state transition that can be used after entering your first state and before entering state 18. (Hint: $S_5S_4S_3 = 110$, and memory ready is the signal R. you can work with the 4 bits $RS_2S_1S_0$)

(d) Using a similar table as in Problem 8, write down the control signals for each state.

(e) Trace through your instruction with the following instruction:
1101 000 101 000111
Assume PC = x3001, R5 = x2000, and M[x2007] = x8EED.
What is PC after the instruction completes and before the next execution of state 18?

13 Challenging Conundrums (Just For Fun)

Unless otherwise specified, you may use any registers you would like and store results in any register

(a) Using only three lines of code, set R0 = 0, R1 = 4, and R2 = -3

(b) Implement R1 NOR R2 in only 3 lines of code

(c) Implement R3 XOR R4 in only 5 lines of code

(d) Implement R5 OR R6 in only 4 lines of code