ECE 120: Introduction to Computing

Fall 2024 – Final Exam

Friday, December 13, 2024

- Ensure that your exam booklet has 15 pages and additional four pages of formula sheets + scratch paper.
- **Please, write your NAME, NetID, and UIN clearly.**
- Do not tear the exam booklet apart. You can only detach the scratch paper and the formula sheet if needed.
- This is a closed book/notes exam. You may use a calculator.
- You are allowed **one handwritten sheet** of notes (both sides). Write your name on the cheat sheet. The cheat sheet will be collected at the end of your exam.
- Absolutely no interaction between students is allowed.
- Clearly indicate any assumptions that you make.
- **The questions are not weighted equally**. Budget your time accordingly.
- Show your work and write legibly. Solutions in i**llegible handwriting will be graded as incorrect.**
- Write your **UIN (9-digit #) on each page in the provided space.**

NAME **ANSWER KEY**

NetID	UIN

Problem 1	10 points	_____
Problem 2	15 points	_____
Problem 3	24 points	_____
Problem 4	10 points	_____
Problem 5	24 points	_____
Problem 6	7 points	_____
Problem 7	10 points	_____
Total	100 points	_____

Problem 1 (10 points): Binary Representation and Operations

1. **(2 points)** The current version of Prof. Lumetta's ECE 120 lecture notes is 170 pages long. If Prof. Lumetta decided to refer to each page using fixed-length binary words, what is the **minimum number of bits** needed per page?

   Minimum number of bits: _____ (decimal number)

   How many additional pages can Prof. Lumetta add to his lecture notes while keeping the same number of bits?

   Additional number of pages: _____ (decimal number)

2. **(2 points)** Perform the following **bitwise** logical operations.

   a. 0010 NAND 0011 = _____ (binary)

   b. 1001 XOR (NOT(0011)) = _____ (binary)

3. **(3 points)** Perform the following operation in **4-bit 2's complement** representation.

   1010 – 1110 = _____ (binary)

   Circle one: Carry out? **YES** **NO**

   Circle one: Overflow? **YES** **NO**

4. **(1 point)** Let **x88** be an 8-bit 2's complement number. What is its decimal value?

   x88 = _____ (decimal number)

5. **(2 points)** What is the largest positive number that a 7-bit unsigned number can represent?

   Your answer in decimal: _____

   What is the largest positive number that a 7-bit 2's complement number can represent?

   Your answer in decimal: _____

UIN_____

Problem 1 (10 points): Binary Representation and Operations

1. **(2 points)** The current version of Prof. Lumetta's ECE 120 lecture notes is 170 pages long. If Prof. Lumetta decided to refer to each page using fixed-length binary words, what is the **minimum number of bits** needed per page?

   Minimum number of bits: 8 (decimal number) $\lceil \log_2(170) \rceil$

   How many additional pages can Prof. Lumetta add to his lecture notes while keeping the same number of bits?

   Additional number of pages: 86 (decimal number) $256 - 170$

2. **(2 points)** Perform the following **bitwise** logical operations.

   a. 0010 NAND 0011 = 1101 (binary)

   b. 1001 XOR (NOT(0011)) = 0101 (binary)

3. **(3 points)** Perform the following operation in **4-bit 2's complement** representation.

   1010 – 1110 = 1100 (binary)

   Circle one: Carry out? **YES** **NO**

   Circle one: Overflow? **YES** **NO**

4. **(1 point)** Let **x88** be an 8-bit 2's complement number. What is its decimal value?

   x88 = −120 (decimal number)

5. **(2 points)** What is the largest positive number that a 7-bit unsigned number can represent?

   Your answer in decimal: 127

   What is the largest positive number that a 7-bit 2's complement number can represent?

   Your answer in decimal: 63

Problem 2 (15 points): Bit-Sliced Comparator

In this problem, you will design circuit M (as shown in Figure 1) for a bit-sliced comparator whose bit-slice design is shown in Figure 2. It compares two n-bit unsigned binary numbers A = $a_{n-1}\ldots a_1a_0$ and B = $b_{n-1}\ldots b_1b_0$. It outputs the following.

L_{out} = 1 if and only if A < B
E_{out} = 1 if and only if A = B
G_{out} = 1 if and only if A > B

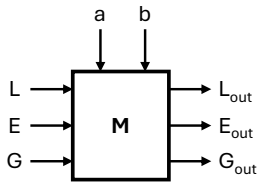


Figure 1

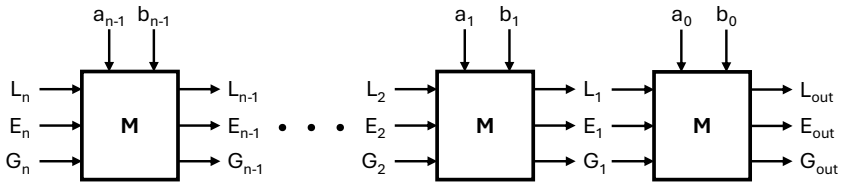


Figure 2.

1. (1 point) Specify the initial inputs L_n, E_n, and G_n.

L_n, E_n, G_n = _____

2. (4 points) In this and next parts, you will design the circuit M. Give the **minimal SOP** expression for $E_{out}(a,b,L,E,G)$. In order to avoid drawing a 5-variable K-map, realize that the expression should not depend on L and G. Therefore, E_{out} only depends on the three remaining variables. Use the following 3-variable K-map if needed.

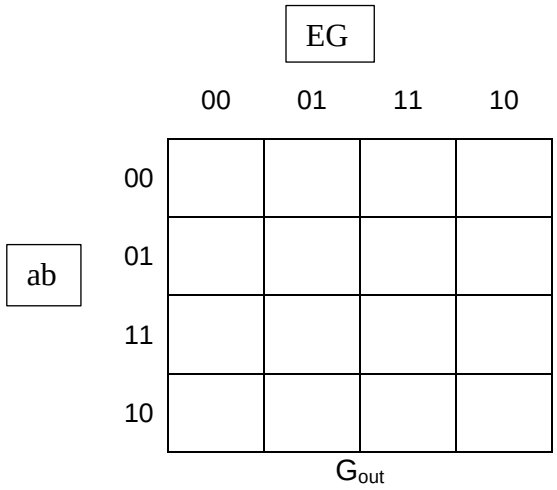
$E_{out}(a,b,L,E,G)$ = _____

	00	01	11	10
0				
1				

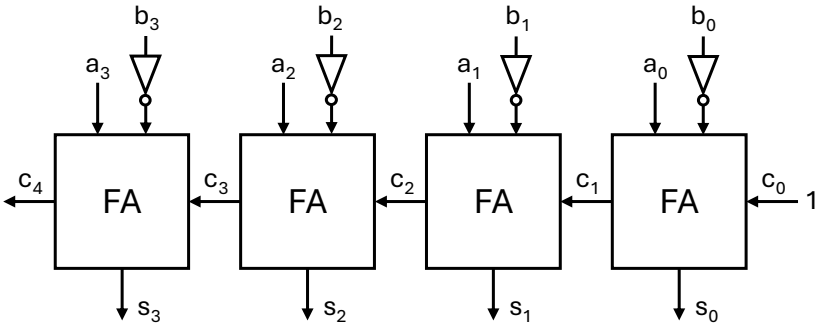
E_{out}

3. (4 points) Now give the **minimal SOP** expression for $G_{out}(a,b,L,E,G)$. Again, we can avoid a 5-variable K-map by realizing that the expression does not depend on L. Use the following 4-variable K-map if needed. You may use don't-cares in the K-map.

$G_{out}(a,b,L,E,G)$ = _____



Parts 4, 5, and 6 below are unrelated to Parts 1, 2, and 3 on the previous page. We are now building a 4-bit comparator using the following adder-based circuit.



4. (2 points) Although the inputs A and B are unsigned, the above circuit effectively calculates A – B, and the result S can be interpreted as a two's complement number. Based on this knowledge, write down the **Boolean expression for E_{out}** which indicates A = B. Write it as a function of adder output signals s_3, s_2, s_1, and s_0.

E_{out} = _____

5. (2 points) Now write down the Boolean expression for L_{out} which indicates A < B. L_{out} can be a function of the adder outputs c_4, c_3, c_2, c_1, s_3, s_2, s_1, and s_0.

L_{out} = _____

6. (2 points) Finally, write down the Boolean expression for G_{out} which indicates A > B. It can be a function of any adder output signals, or it can be a function of the output signals implemented above, which are E_{out} and L_{out}.

G_{out} = _____

UIN_____

1. (1 point) Specify the initial inputs L_n, E_n, and G_n.

$L_n, E_n, G_n =$ 0 1 0

2. (4 points) In this and next parts, you will design the circuit M. Give the **minimal SOP** expression for $E_{out}(a,b,L,E,G)$. In order to avoid drawing a 5-variable K-map, realize that the expression should not depend on L and G. Therefore, E_{out} only depends on the three remaining variables. Use the following 3-variable K-map if needed.

$E_{out}(a,b,L,E,G) =$ $\bar{a}\bar{b}E + abE$

Expression should be unique

a b

E \	00	01	11	10
0	0	0	0	0
1	1	0	1	0

E_{out}

$E = \bar{a}\bar{b}E + abE$

bE

a \	00	01	11	10
0		1		
1			1	

$= \bar{a}\bar{b}E + abE$

	a	b	E	E_{out}
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1

3. **(4 points)** Now give the **minimal SOP** expression for $G_{out}(a,b,L,E,G)$. Again, we can avoid a 5-variable K-map by realizing that the expression does not depend on L. Use the following 4-variable K-map if needed. You may use don't-cares in the K-map.

$G_{out}(a,b,L,E,G) =$ G + ab̄E

ab \ EG	00	01	11	10
00	0	1	X	0
01	0	1	X	0
11	0	1	X	0
10	0	1	X	1

G_{out}

4. **(2 points)** Although the inputs A and B are unsigned, the above circuit effectively calculates A – B, and the result S can be interpreted as a two's complement number. Based on this knowledge, write down the **Boolean expression for E_{out}** which indicates A = B. Write it as a function of adder output signals s_3, s_2, s_1, and s_0.

$E_{out} =$ $\bar{s}_3 \bar{s}_2 \bar{s}_1 \bar{s}_0$

5. **(2 points)** Now write down the Boolean expression for L_{out} which indicates A < B. L_{out} can be a function of the adder outputs c_4, c_3, c_2, c_1, s_3, s_2, s_1, and s_0.

$L_{out} =$ s_3

6. **(2 points)** Finally, write down the Boolean expression for G_{out} which indicates A > B. It can be a function of any adder output signals, or it can be a function of the output signals implemented above, which are E_{out} and L_{out}.

$G_{out} =$ $\bar{s}_3 (s_2 + s_1 + s_0)$

Problem 3 (24 points): FSM Design Problem

In this problem, you will implement the FSM of a dynamic cruise controller (DCC) used in any modern vehicle. The proposed DCC should automatically adjust your vehicle's speed to maintain a safe distance between your car and the car in front of you. The car uses cameras and other range sensors to measure the distance between the cars on the same lane.

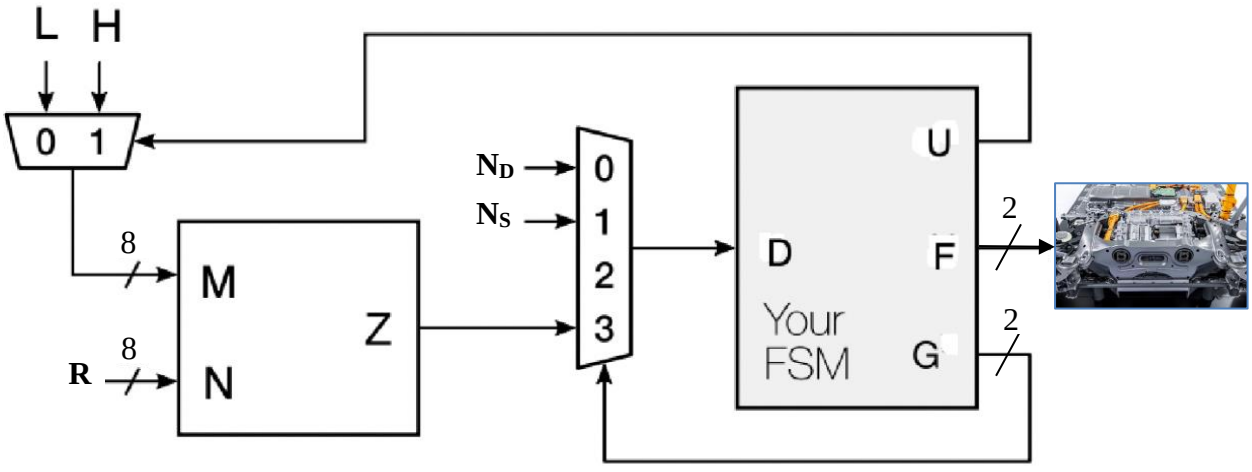
Note: A distance measuring system is already installed in the car and its output **R** is available to you. R is represented as an 8-bit unsigned value. The DCC is intended to keep your car within a safe distance specified as desired values of L and H, i.e., L ≤ R < H, and L < H.

If R is between L and H, the controller stops accelerating or decelerating the car and instead maintains a constant speed; this is to prevent it from cycling the car decelerating and accelerating too often. For example, to maintain an average distance of 100 feet, L might be 75 feet, and H might be 125 feet.

Note that there is a maximum speed limit. When the car reaches the maximum speed, the car engine would generate a high signal Ns=1. The DCC controller will use the signal to maintain a constant maximum speed until the sensing distance R becomes smaller than L (i.e. R<L). Otherwise, Ns = 0.

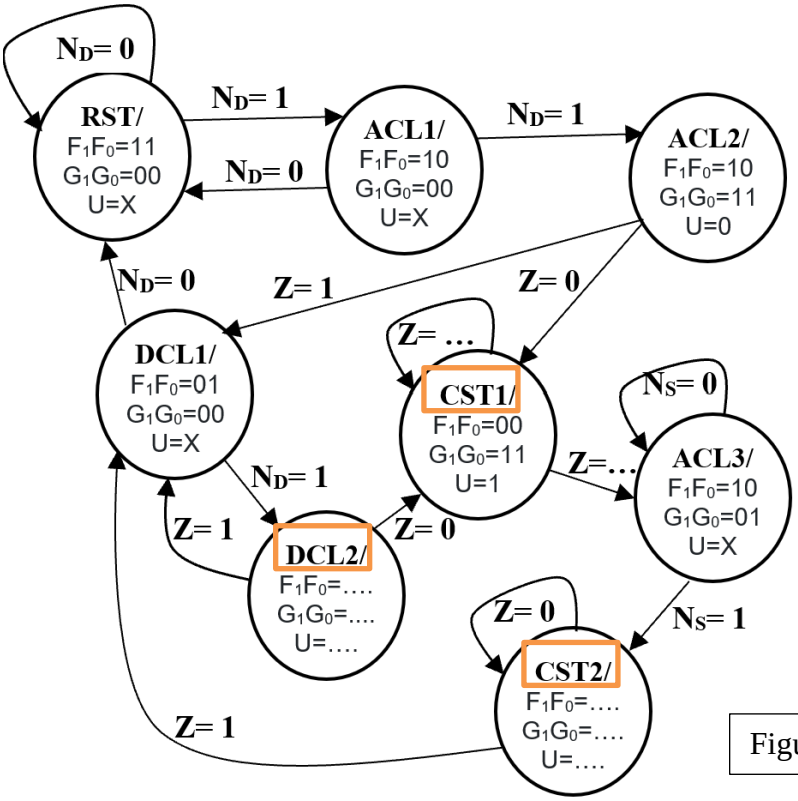
Consider another signal N_D, which is 1 so long as DCC is active, and otherwise 0. Note: for the safety of the system, whenever the driver presses the brake-pedal, N_D becomes 0.

For some design requirements, your FSM is required to have only a single 1-bit input signal, denoted D (as shown in the following figure). To determine the status of the signals (Ns and N_D), they are connected to the multiplexer (whose output is an input to your FSM) that allows you to access the state of the signals as well as the output of the comparator. The circuit is shown in the figure below.



Note that the signal Z is the output of a Greater-Than comparator. Note: Z = **1,** if **M > N,** and **0** otherwise.

The following state diagram of the Moore FSM of the desired DCC controller is given to you to get started with the implementation of Your FSM. Note: The outputs (U, F, and G) in each state are partially filled with desired bits. The states are described on the table below. The current state of the FSM is denoted as ABC. The input to your FSM is 1-bit signal D. The output of your FSM are three signals F, U and G. Signal U controls the multiplexer that selects the M input to the comparator. Signal F is a 2-bit signal (F_1F_0) that controls the engine. We choose the representation as **F_1F_0 = 00** (keeps the car at a constant speed), **F_1F_0=01** (decelerates the car), **F_1F_0 = 10** (accelerates the car), and **F_1F_0 = 11** (manual mode). G is 2-bit signal (G_1G_0) that controls the multiplexer for the input D. U can be don't care (X) if not used in a state.



Abstract names represent DCC's operations:
RST-DCC is not activated
ACL1, ACL2 and ACL3 -Acceleration phases
DCL1 and DCL2 – Deceleration phases
CST1 and CST2 – Constant Speed

States are represented using 3 bits ABC.

Output (**F_1F_0**) is a 2-bit signal, which controls the engine.
Output (G_1G_0) is a 2-bit signal, which selects control inputs to the FSM.
Output U is a 1-bit signal, which selects L or H to be compared with R in the grater than comparator.

Figure 3. State Diagram

States/ ABC	Description
RST/000	Reset state. It keeps the car in manual mode. It checks **N_D** and makes a transition if **N_D =1**
ACL1/001	Acceleration state 1. It accelerates the car. It checks **N_D** and makes necessary transitions.
ACL2/010	Acceleration state 2. It keeps accelerating the car. It checks **L**, compares it with **R** and makes necessary transitions.
DCL1/011	Deceleration state 1. It decelerates the car. It checks **N_D** and makes necessary transitions.
CST1/100	Constant speed state 1. It keeps the car at a constant speed. It checks **H**, compares it with R and makes necessary transitions.
DCL2/101	Decelerate state 2. It keeps decelerating the car. It checks **L**, compares it with R and makes necessary transitions.
ACL3/110	Accelerate state 3. It keeps accelerating the car. It checks **N_S** and makes necessary transitions.
CST2/111	Constant speed state 2. It keeps the car at a constant speed. It checks **L** and makes necessary transitions.

1. [8 points] Fill out the missing information of states DCL2, CST1 and CST2 on the state diagram of figure 3.

2. [6 points] Draw the next state transition table and derive the next state transition expression in minimal SOP. Let A^+,B^+,C^+ denote the next state of the FSM. Hint: The FSM has only one input D. Thus, you need 4-variable k-map.

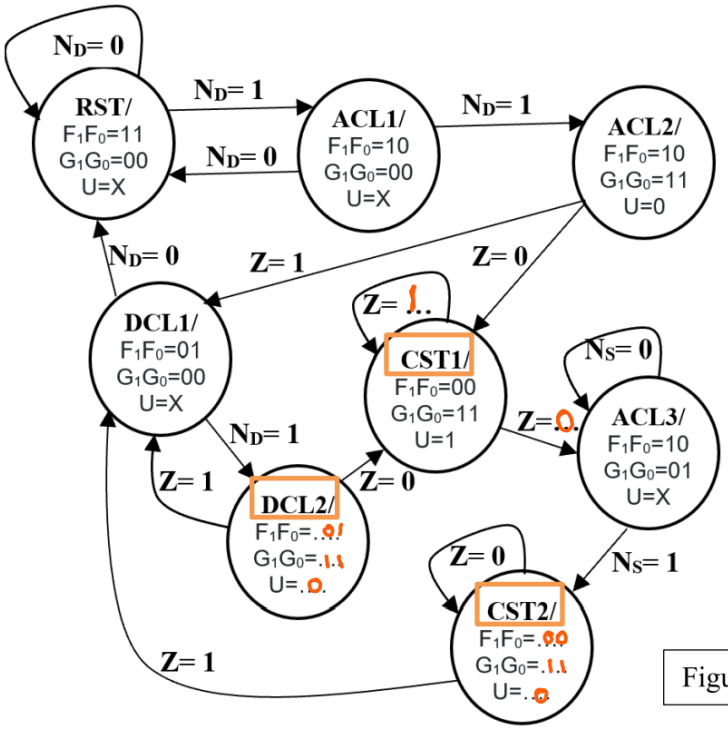
3. [2 points] Derive the expression output expression $\mathbf{F_0}$ in terms of A, B, C in minimal SOP

4. [2 points] Derive the expression output expression $\mathbf{F_1}$ in terms of A, B, C in minimal SOP

5. [2 points] Derive the expression U in terms of A, B, C in minimal SOP

6. [2 point] Derive the expression G_0 in terms of A, B, C in minimal SOP

7. [2 point] Derive the expression G_1 in terms of A, B, C in minimal SOP



Abstract names represent DCC's operations:
RST-DCC is not activated
ACL1, ACL2 and ACL3 -Acceleration phases
DCL1 and DCL2 – Deceleration phases
CST1 and CST2 – Constant Speed

States are represented using 3 bits ABC.

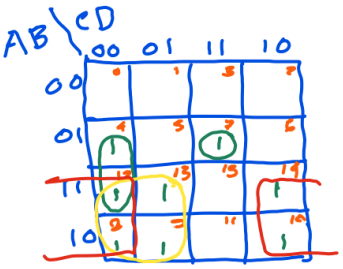
Output (F_1F_0) is a 2-bit signal, which controls the engine.
Output (G_1G_0) is a 2-bit signal, which selects control inputs to the FSM.
Output U is a 1-bit signal, which selects L or H to be compared with R in the grater than comparator.

Figure 3. State Diagram

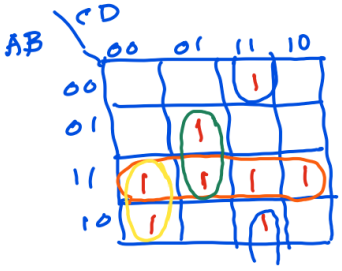
States/ ABC	Description
RST/000	Reset state. It keeps the car in manual mode. It checks N_D and makes a transition if N_D =1
ACL1/001	Acceleration state 1. It accelerates the car. It checks N_D and makes necessary transitions.
ACL2/010	Acceleration state 2. It keeps accelerating the car. It checks **L**, compares it with **R** and makes necessary transitions.
DCL1/011	Deceleration state 1. It decelerates the car. It checks N_D and makes necessary transitions.
CST1/100	Constant speed state 1. It keeps the car at a constant speed. It checks **H**, compares it with R and makes necessary transitions.
DCL2/101	Decelerate state 2. It keeps decelerating the car. It checks **L**, compares it with R and makes necessary transitions.
ACL3/110	Accelerate state 3. It keeps accelerating the car. It checks N_S and makes necessary transitions.
CST2/111	Constant speed state 2. It keeps the car at a constant speed. It checks **L** and makes necessary transitions.

2. [6 points] Draw the next state transition table and derive the next state transition expression in minimal SOP. Let A^+,B^+,C^+ denote the next state of the FSM. Hint: The FSM has only one input D. Thus, you need 4-variable k-map.

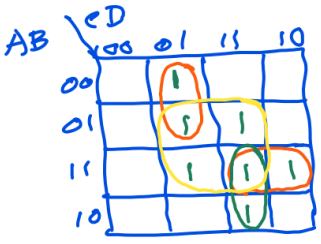
	A B C D	A^+ B^+ C^+
RST	0 0 0 0	0 0 0
	0 0 0 1	0 0 1
ACL1	0 0 1 0	0 0 0
	0 0 1 1	0 1 0
ACL2	0 1 0 0	0 0 0
	0 1 0 1	0 1 1
DCL1	0 1 1 0	0 0 0
	0 1 1 1	1 0 1
CST1	1 0 0 0	1 1 0
	1 0 0 1	1 0 0
DCL2	1 0 1 0	1 0 0
	1 0 1 1	0 1 1
ACL3	1 1 0 0	1 1 0
	1 1 0 1	1 1 1
CST2	1 1 1 0	1 1 1
	1 1 1 1	0 1 1



$A^+ = \bar{A}BCD + B\bar{C}\bar{D} + A\bar{C} + A\bar{D}$



$B^+ = \bar{B}CD + B\bar{C}D + A\bar{C}\bar{D} + AB$



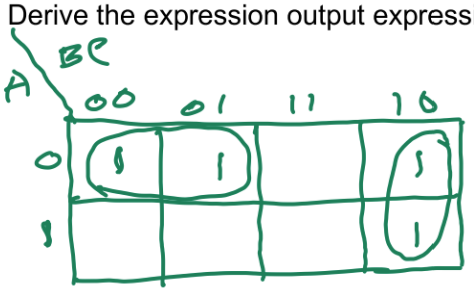
$C^+ = \bar{A}\bar{C}D + BD + ABC + ACD$

3. [2 points] Derive the expression output expression F_0 in terms of A, B, C in minimal SOP

ABC	F_1	F_0	G_1	G_0	U
0 0 0	1	1	0	0	x
0 0 1	1	0	0	0	x
0 1 0	1	0	1	1	0
0 1 1	0	1	0	0	x
1 0 0	0	0	1	1	1
1 0 1	0	1	1	1	0
1 1 0	1	0	0	1	x
1 1 1	0	0	1	1	0

$F_0 = \bar{A}\bar{B}\bar{C} + \bar{A}BC + A\bar{B}C$

4. [2 points] Derive the expression output expression F_1 in terms of A, B, C in minimal SOP



$F_1 = \bar{A}\bar{B} + B\bar{C}$

5. [2 points] Derive the expression U in terms of A, B, C in minimal SOP

$U = \bar{B}\bar{C}$ OR $U = A\bar{C}$

6. [2 point] Derive the expression G_0 in terms of A, B, C in minimal SOP

$G_0 = A + B\bar{C}$

7. [2 point] Derive the expression G_1 in terms of A, B, C in minimal SOP

$G_1 = \bar{A}B\bar{C} + A\bar{B} + AC$

Problem 4 (10 points): LC-3 Interpretation and Assembly

1. The program below consists of LC-3 instructions and data.

```
x3000 0010 0110 0000 1011  LD R3, #11
x3001 0101 1011 0110 0000  _____________________________
x3002 0110 1100 1100 0000  _____________________________
x3003 0000 0110 0000 0011  BRzp #3
x3004 1001 1001 1011 1111  _____________________________
x3005 0001 1101 0010 0001  _____________________________
x3006 0111 1100 1100 0000  _____________________________
x3007 0001 0110 1110 0001  _____________________________
x3008 0001 1011 0110 0001  ADD R5, R5, #1
x3009 0001 1101 0111 1000  _____________________________
x300A 0000 1001 1111 0111  _____________________________
x300B 1111 0000 0010 0101  HALT
x300C 1111 0000 0000 0000  FILL .xF000
```

a. **(8 points)** Decode the remaining instructions in the program above into LC-3 assembly language using the format shown in those already done for you. Note that all blanks correspond to instructions, not data.

b. **(2 points)** In **twenty words or less**, explain what the program does.

UIN_____

1. The program below consists of LC-3 instructions and data.

```
x3000 0010 0110 0000 1011  LD R3, #11
x3001 0101 1011 0110 0000  AND R5, R5, #0
x3002 0110 1100 1100 0000  LDR R6, R3, #0
x3003 0000 0110 0000 0011  BRzp #3
x3004 1001 1001 1011 1111  NOT R4, R6
x3005 0001 1101 0010 0001  ADD R6, R4, #1
x3006 0111 1100 1100 0000  STR R6, R3, #0
x3007 0001 0110 1110 0001  ADD R3, R3, #1
x3008 0001 1011 0110 0001  ADD R5, R5, #1
x3009 0001 1101 0111 1000  ADD R6, R5, #-8
x300A 0000 1001 1111 0111  BRn #-9
x300B 1111 0000 0010 0101  HALT
x300C 1111 0000 0000 0000  FILL .xF000
```

b. **(2 points)** In **twenty words or less**, explain what the program does.

Read consecutive 8 memory location starting from x F000. If the value read is negative, invert the value (i.e. make it positive) and put it back into its location.

Problem 5 (24 points)

In this problem we introduce a new instruction to the LC3 instruction set, called STLDR.

STLDR BaseR1, BaseR2, OP2* **Note: OP2 may be SR2 or SEXT[imm5]**
IR[5] bit =1 if OP2 is SEXT[imm5], else 0.

This instruction copies a value from one memory location whose address is stored in the register specified by IR[11:9] bits to another memory location whose address is specified as a Base-relative address, **BaseR2+OP2**. STLDR has opcode 1101 and the RTL is:

M[BaseR2 + OP2] ← M[BaseR1]

1. (4 points) Give the binary encoding of the instruction **STLDR R3, R4, R5** by filling in the missing bits.

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

2. (2 points) Identify the bug in the instruction **STLDR R2, R1, #-32**.

_____

3. (6 points) Fill the missing microinstructions that implement the STLDR instruction **after the decode state**.

Answer: 1. MAR ← BaseR1

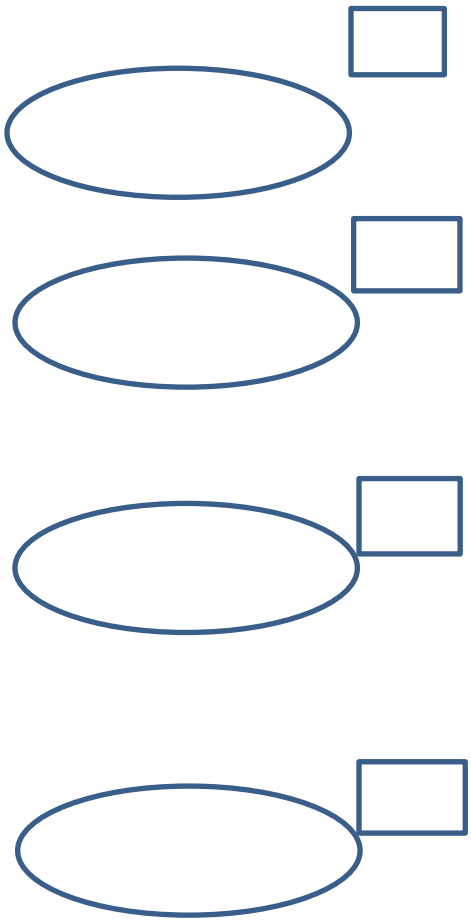
2. _____

3. _____

4. _____

4. (4 points) Draw the state diagram for **STLDR after the decode state with the arc labels as needed, the RTLs mentioned in the states,** and include the state numbers in the boxes provided**.** State numbers 51, 52, 53, and 54 are available as additional states for your use. Hint: your answer should be consistent with the simplified Patt and Patel microsequencer circuit attached to this booklet.

   Note: $51_{10} = 110011_2$, $52_{10} = 110100_2$, $53_{10} = 110101_2$, $54_{10} = 110110_2$



5. **(8 points)** Determine the control ROM microinstructions that implement the RTL statements from part (3). Complete the table below by filling in **0**, **1**, or **x** as appropriate. Specify ROM addresses in **decimal**. Note that your first state number is implied by the decode strategy used in the LC-3 microarchitecture. State numbers 51, 52, 53, and 54 are available as additional states for your use. The order needs to match your solution in part (3).

ROM address in decimal	IRD	COND(3) , COND0 is LSB	J(6), J0 is the LSB	LD.BEN	LD.MAR	LD.MDR	LD.IR	LD.PC	LD.REG	LC.CC	GateMARMUX	GateMDR	GateALU	GatePC	MARMUX	PCMUX(2)	ADDR1MUX	ADDR2MUX(2)	DRMUX(2)	SR1MUX(2)	ALUK(2)	MIO.EN	R.W
				Do not fill in datapath control word fields for these two microinstructions. But be sure to fill in the remaining two.																			

1. (4 points) Give the binary encoding of the instruction **STLDR R3, R4, R5** by filling in the missing bits.

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

2. (2 points) Identify the bug in the instruction **STLDR R2, R1, #-32**

Using 5-bit offset we can not represent #-32 (i.e. wrong Operand)

3. (6 points) Fill the missing microinstructions that implement the STLDR instruction **after the decode state**.

Answer: 1. MAR ← BaseR1

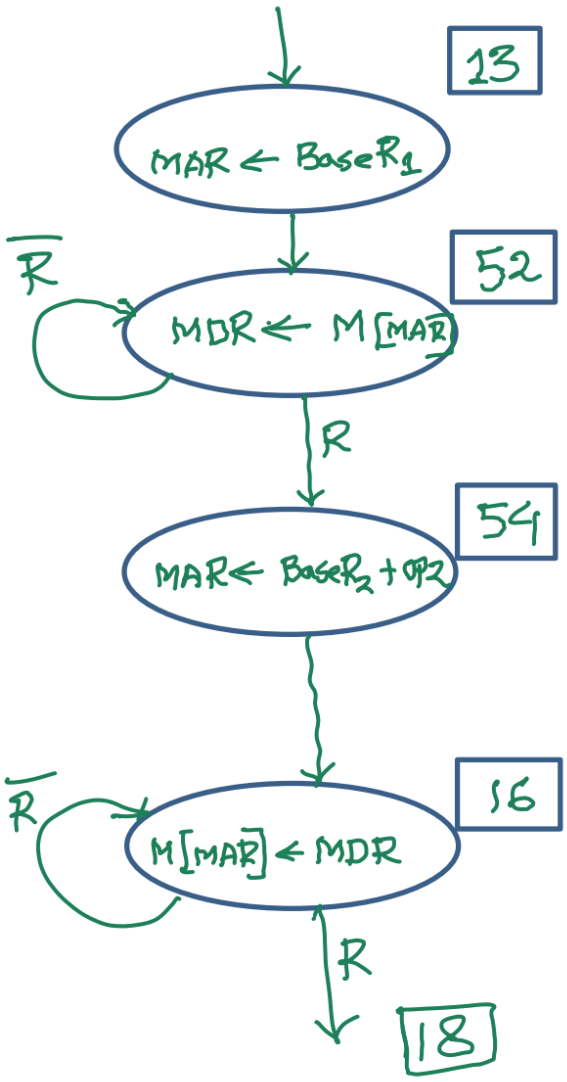
2. MDR ← M[MAR]

3. MAR ← [BaseR2 + OP2]

4. M[MAR] ← MDR

4. (4 points) Draw the state diagram for **STLDR after the decode state with the arc labels as needed, the RTLs mentioned in the states,** and include the state numbers in the boxes provided**.** State numbers 51, 52, 53, and 54 are available as additional states for your use. Hint: your answer should be consistent with the simplified Patt and Patel microsequencer circuit attached to this booklet.

Note: $51_{10} = 110011_2$, $52_{10} = 110100_2$, $53_{10} = 110101_2$, $54_{10} = 110110_2$



5. **(8 points)** Determine the control ROM microinstructions that implement the RTL statements from part (3). Complete the table below by filling in **0**, **1**, or **x** as appropriate. Specify ROM addresses in **decimal**. Note that your first state number is implied by the decode strategy used in the LC-3 microarchitecture. State numbers 51, 52, 53, and 54 are available as additional states for your use. The order needs to match your solution in part (3).

ROM address in decimal	IRD	COND(3) , COND0 is LSB	J(6), J0 is the LSB	LD.BEN	LD.MAR	LD.MDR	LD.IR	LD.PC	LD.REG	LC.CC	GateMARMUX	GateMDR	GateALU	GatePC	MARMUX	PCMUX(2)	ADDR1MUX	ADDR2MUX(2)	DRMUX(2)	SR1MUX(2)	ALUK(2)	MIO.EN	R.W
13	0	000	110100	Do not fill in datapath control word fields for these two microinstructions. But be sure to fill in the remaining two.																			
52	0	001	110100																				
54	0	000	010000	0	1	0	0	0	0	0	0	0	1	0	x	xx	x	xx	xx	01	00	0	x
16	0	001	010000	0	0	0	0	0	0	0	0	0	0	0	x	xx	x	xx	xx	xx	xx	1	1

UIN_____

Problem 6 (7 points): LC3 Basics

1. (1+4 points) Shifting Time:

a. Suppose a register has a value x0002. What is the value of the register **in hex** after shifting the bits to the **left by 2 bits?**

Answer: x_____

b. Complete the following assembly program to shift the value stored in R3 to the left by four bits. Assume there is some initial value in R3.

```
        .ORIG x3000
        AND R2,R2,#0
        ADD R2,R2,#4
LOOP    BRz DONE
        _________________

        _________________

        BRnzp LOOP
DONE    HALT
        .END
```

2. **[2 points]** One of our resident ECE programmers intended to write an LC-3 program to output a **%** to the monitor and then halt. Unfortunately, the programmer got confused about the semantics of each of the opcodes (i.e., exactly what function is carried out by the LC-3 in response to each opcode). Replace exactly **one** opcode in this program with the correct one to make the program work as intended, mentioning the line number and the corrected instruction as your answer.

```
      .ORIG x3000      ;line 0
      STI R0, LABL     ;line 1
      OUT              ;line 2
      HALT             ;line 3
LABL  .STRINGZ "%"     ;line 4
      .END             ;line 5
```

Answer: Line number:_____Correct Instruction:_____

(5 points) Shifting Time:

a. Suppose a register has a value x0002. What is the value of the register **in hex** after shifting the bits to the **left by 2 bits?**

Answer: x 0008

b. Complete the following assembly program to shift the value stored in R3 to the left by four bits. Assume there is some initial value in R3.

```
        .ORIG x3000
        AND R2,R2,#0
        ADD R2,R2,#4
LOOP    BRz DONE
        ADD R3, R3, R3
        ADD R2, R2, #-1

        BRnzp LOOP
DONE    HALT
        .END
```

2. **[2 points]** One of our resident ECE programmers intended to write an LC-3 program to output a character **%** to the monitor and then halt. Unfortunately, the programmer got confused about the functions of the opcodes. Replace exactly **one** instruction in this program with the correct one to make the program work as intended. Mention the line number and the corrected instruction as your answer.

```
        .ORIG x3000      ;line 0
        STI R0, LABL     ;line 1
        OUT              ;line 2
        HALT             ;line 3
LABL    .STRINGZ "%"     ;line 4
        .END             ;line 5
```

Answer: Line number: 1 Correct Instruction: LD R0, LABL

Problem 7 (10 points): LC-3 Assembly Language Interpretation

1. [3 points] In each of the three small programs below, assume that memory location x3004 has been initialized with some bits before the programs are run. We know that the 16 bits in that location may be interpreted as a 2's complement integer, an address, an ASCII value, or an instruction. For each program indicate what the value in x3004 is interpreted as.

A)

```
        .ORIG x3000
        LDI R0, H
        ADD R0, R0, #1
        HALT
H       .FILL x3004
        .END
```

Answer:

B)

```
        .ORIG x3000
        LDI R0, J
        LDR R1, R0, #0
        HALT
J       .FILL x3004
        .END
```

Answer:

C)

```
        .ORIG x3000
        LDI R0, K
        OUT
        HALT
K       .FILL x3004
        .END
```

Answer:

2. [4 points] Consider the following LC-3 assembly language program:

```
        .ORIG x3000
L1      LEA R1, L1
        AND     R2, R2, x0
        ADD     R2, R2, x2
        LD      R3, P1
L2      LDR  R0, R1, xC
        OUT
        ADD     R3, R3, #-1
        BRz     DONE
        ADD     R1, R1, R2
        BR      L2
```

```
DONE HALT
P1    .FILL  x7
      .STRINGZ “NNeaummaasntne!?”
      .END
```

A) After this program is assembled and loaded, what binary pattern is stored in memory location x3005?

Answer (Convert the binary pattern into Hex):

B) What is the output of this program?

Answer:

3. [3 points] The following LC-3 program is assembled and then executed. There are no assemble time or run-time errors. **Assume all registers are initialized to 0 before the program executes.**

```
      .ORIG    x3000
      LEA R0, LABL
      STR R1, R0, #3
      TRAP     x22            ;PUTS
      TRAP     x25            ;HALT
LABL  .STRINGZ "ECE120"
LABL2 .STRINGZ "AMAZING"
      .END
```

A) What is the address of the memory location labeled LABL?

Answer: _____

B) What is the address of the memory location labeled LABL2?

Answer: _____

C) What is the output of this program?

Answer: _____

Problem 7 (10 points): LC-3 Assembly Language Interpretation

1. [3 points] In each of the three small programs below, assume that memory location x3004 has been initialized with some bits before the programs are run. We know that the 16 bits in that location may be interpreted as a 2's complement integer, an address, an ASCII value, or an instruction. For each program indicate what the value in x3004 is interpreted as.

A)

```
        .ORIG x3000
        LDI R0, H
        ADD R0, R0, #1
        HALT
H       .FILL x3004
        .END
```

Answer: 2's Complement Integer

B)

```
        .ORIG x3000
        LDI R0, J
        LDR R1, R0, #0
        HALT
J       .FILL x3004
        .END
```

Answer: An Address

C)

```
        .ORIG x3000
        LDI R0, K
        OUT
        HALT
K       .FILL x3004
        .END
```

Answer: An Asell Value

2. [4 points] Consider the following LC-3 assembly language program:

```
        .ORIG x3000
L1      LEA R1, L1
        AND     R2, R2, x0
        ADD     R2, R2, x2
        LD      R3, P1
L2      LDR  R0, R1, xC
        OUT
        ADD     R3, R3, #-1
        BRz     DONE
        ADD     R1, R1, R2
        BR      L2
DONE    HALT
P1      .FILL  x7
        .STRINGZ "NNeaummaasntne!?"
        .END
```

A) After this program is assembled and loaded, what binary pattern is stored in memory location x3005?

Answer (Convert the binary pattern into Hex): 1111 0000 0010 0001 = xF021

B) What is the output of this program?

Answer: Neumann

UIN_____

3. [3 points] The following LC-3 program is assembled and then executed. There are no assemble time or run-time errors. **Assume all registers are initialized to 0 before the program executes.**

```
        .ORIG    x3000
        LEA R0, LABL
        STR R1, R0, #3
        TRAP     x22          ;PUTS
        TRAP     x25          ;HALT
LABL   .STRINGZ "ECE120"
LABL2  .STRINGZ "AMAZING"
        .END
```


UIN_____

A) What is the address of the memory location labeled LABL?

Answer: x3004

B) What is the address of the memory location labeled LABL2?

Answer: x300B

C) What is the output of this program?

Answer: ECE