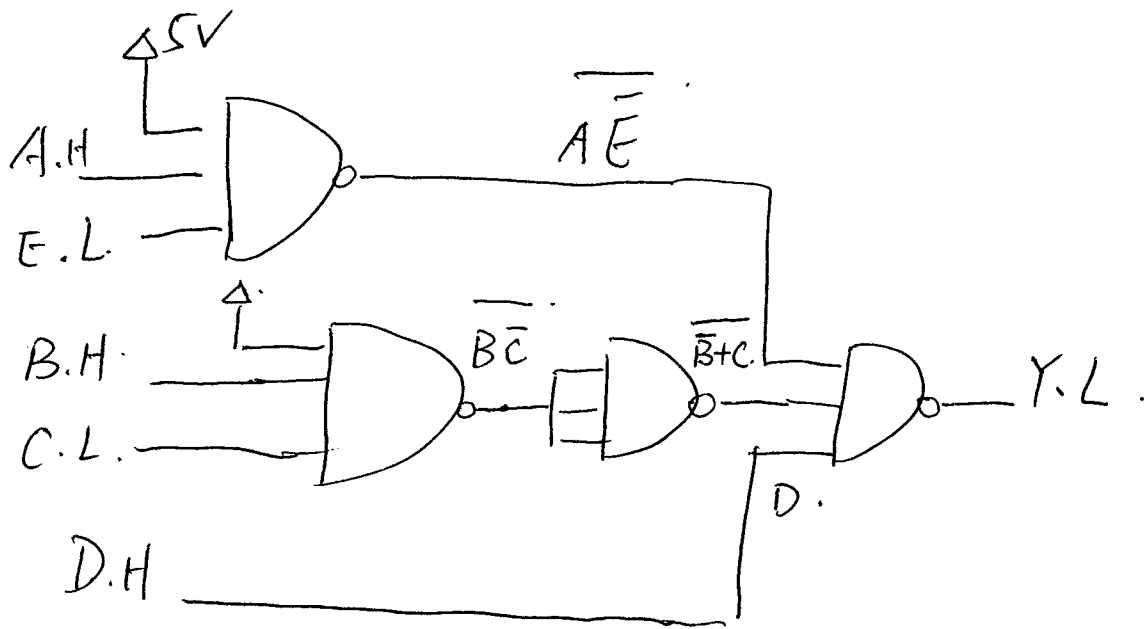
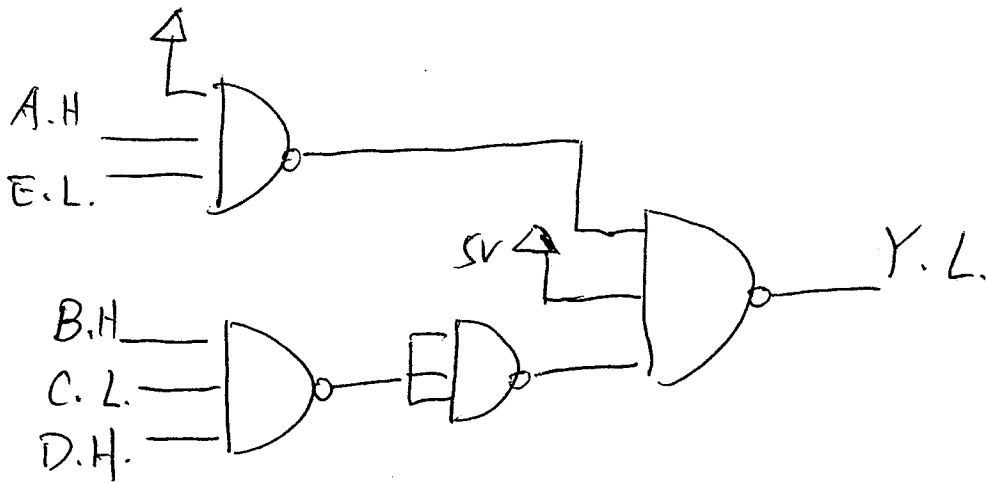


Problem 1: Solution .



OR .



2. Answer the following for the given next-state **truth (logic) table**. Y and Z are outputs and X is the only input.

(14 pts.)

	X	Q ₂	Q ₁	Q ₀	Y	Z	Q ₂ ⁺	Q ₁ ⁺	Q ₀ ⁺
00	0	0	0	0	0	0	1	1	1
01	0	0	0	1	0	0	0	0	0
02	0	0	1	0	0	0	0	0	1
03	0	0	1	1	1	1	0	1	0
04	0	1	0	0	X ₀	X ₀	X ₀	X ₀	X ₀
05	0	1	0	1	X ₀	X ₀	X ₀	X ₀	X ₀
06	0	1	1	0	1	0	0	1	1
07	0	1	1	1	1	1	1	1	0
08	1	0	0	0	0	0	1	1	1
09	1	0	0	1	0	0	1	1	1
0A	1	0	1	0	0	0	1	1	1
0B	1	0	1	1	1	0	1	1	1
0C	1	1	0	0	X ₀	X ₀	1	1	1
0D	1	1	0	1	X ₀	X ₀	1	1	1
0E	1	1	1	0	1	0	1	1	1
0F	1	1	1	1	1	0	1	1	1

- a) What kind of output is Y; and what type is Z? Be specific.

Y: (1 pt.) *Unconditional*

Z: (1 pt.) *Conditional*

OR Y: H Z: L Q₂⁺ Q₁⁺ Q₀⁺

For Q₂⁺ Q₁⁺ Q₀⁺ Y: H Z: L

00	07
01	00
02	01
03	1A
04	00
05	00
06	13
07	1E
08	07
09	07
0A	07
0B	17
0C	07
0D	07
0E	17

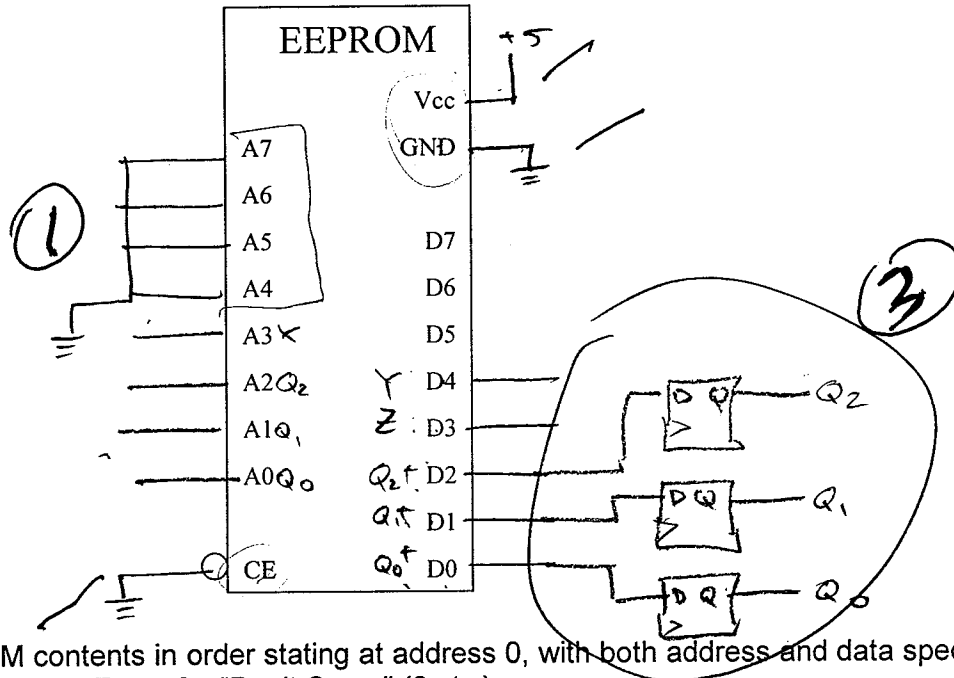
00	1D
01	01
02	05
03	0A
04	00/01
05	00/01
06	0F
07	1A
08	1D
09	1D
0A	1D
0B	1F
0C	1C/1D
0D	1C/1D
0E	1F
0F	1F

2. (continue)

Design a circuit to implement the truth table from the previous page.

- Use only a ROM and D-FF's. Use no other SSI, MSI or LSI elements.
- The signals X and Y are active-high and Z is active-low.
- Use as little of the EEPROM as possible. List the ROM contents in order stating at address 0

(b) Complete the following circuit by drawing in the appropriate number of D-FF's and show connections to all EEPROM signals. (4 pts.)



(c) List the ROM contents in order stating at address 0, with both address and data specified in HEX. Program Zeros for "Don't Cares" (8 pts.)

00	0F
01	08
02	09
03	12
04	08 / 00
05	08 / 00
06	18
07	16
08	0F

09	0F
0A	0F
0B	1F
0C	0F / 07
0D	0F / 07
0E	1F
0F	1F

-1 for activation level

3. Hand Assembly & Program Analysis (Use the G-CPU information in the appendix)

Assume that you are given an 8Kx8 EPROM that has been erased completely. i.e. all memory contents are now \$FF.

(14 pts.) Hand Assemble the following G-CPU program and show the contents of EPROM memory (Address & Data) that has been modified after the EPROM is programmed:

0001000
11000101
1001101
C D

ORG \$0080
T1 dc.b \$45
T2 dc.b \$88
T3 dc.b %10101101 AD
ORG \$0
LDAA \$81
LDAB T1
OR BA
LDX #\$0180
TOP STAA \$12,X
BN TOP

Branch if -ve?

A	B	X
88	45	

Not lower byte first
Only 8bit Address.

(a) EPROM Memory Programmed due to the above Data & G-CPU Code:

9.4
4
first 4 bytes

Addr (Hex)	Data (Hex)
0000	04
0001	81
0002	00
0003	05
0004	80
0005	00
0006	18

Addr (Hex)	Data (Hex)
0007	08
0008	80
0009	01
000A	10
000B	12
000C	22
000D	0A

Addr (Hex)	Data (Hex)
T1 0080	45
T2 0081	88
T3 0082	AD

(b) In the above program what is the effective address associated with the "STX T1" LDAB T1 instruction?

\$0080 Hex

(c) What is the effective address associated with the "LDX #\$0180" instruction?

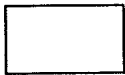
\$0008 Hex

(d) What is the effective address associated with the "STX \$12,X" instruction?

\$0192 Hex 0180 + 0012

(e) What is the value of the A register the first time "STX \$12,X" is executed? Will the branch occur?

A Register Contents = CD Hex Branch will occur (circle one)? T or F

4. Program Execution (Use the G-CPU information in the appendix)

Given the following program in EPROM memory, fill out the cycle table that follows:

(18 pts.)

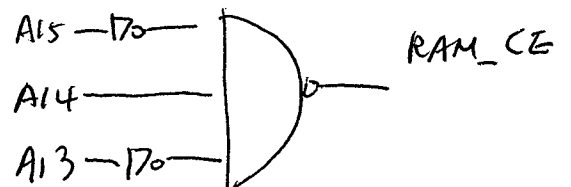
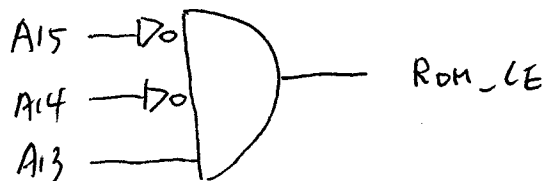
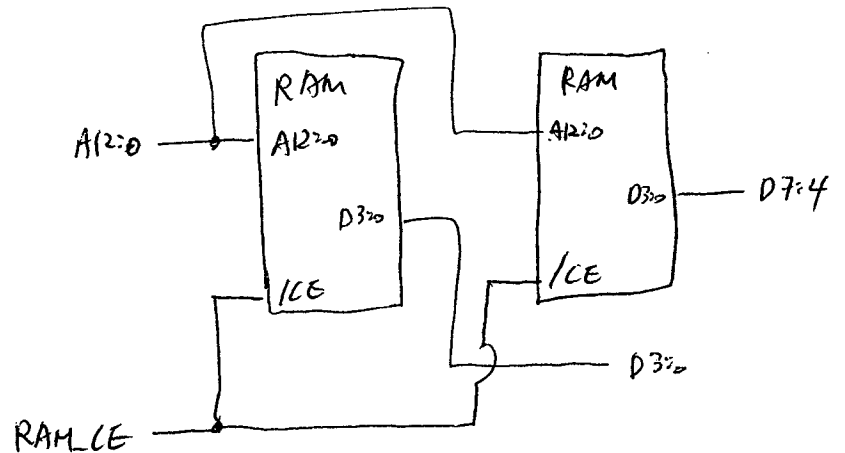
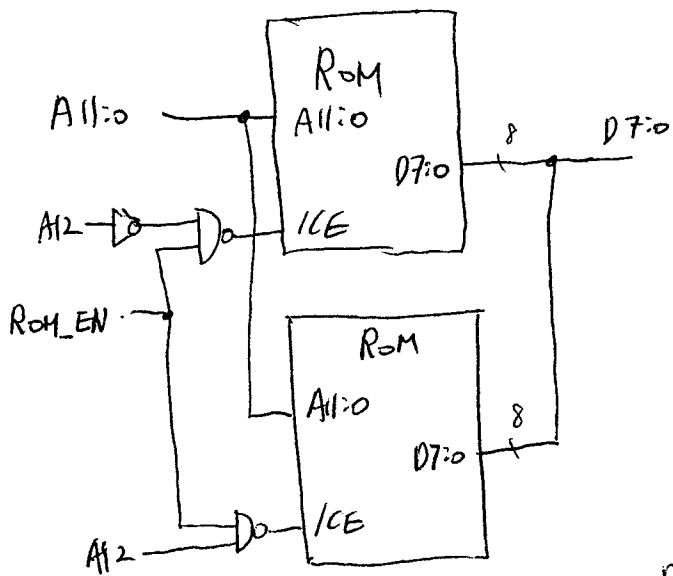
Addr	Data
0	02
1	35
2	06
3	0A
4	00

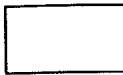
LDAA #35
STAA \$000A

Using the the G-CPU Controller ASM & Block Diagram (Appendix A handouts), complete the cycle table below:

Cycle#	R/- W	PC (Hex)	MAR	A15:0 (Hex)	Data (Hex)	IR (Hex)	A (Hex)	AddrSel1:0
1	1	0000	XXXX	0000	02	XX	XX	00
2	1	0000	XXXX	0000	02	02	XX	00
3	1	0001	XXXX	0001	35	02	XX	00
4	1	0002	XXXX	0002	06	02	35	00
5	1	0002	XXXX	0002	06	06	35	00
6	1	0003	XXXX	0003	0A	06	35	00
7	1	0004	XX0A	0004	00	06	35	00
8	0	0005	000A	000A	35	06	35	01
9								
10								

prob. 5 (solution)



6. GCPU Assembly Programming (Use the G-CPU instruction set in Appendix A)

(12 pts.)

Given the following constants in EPROM and SRAM Memory, write a program to fill the first 128 memory locations in SRAM with zeros.

Assumptions:

1. EPROM exists in the memory map from 0-FFF Hex.
2. SRAM exists in the memory map from 1000-1FFF Hex.
3. Your program should begin at address 0 in EPROM.
4. Write a program to zero out (clear) the first 128 locations in SRAM.

ORG \$0

```
LDX    #$1000    ;ptr to SRAM
LDAB   #128      ;loop counter
```

```
Top LDA    #0
    STAA   0,X    ;clear location
    INX    ;inc ptr
```

```
    LDA    #$FF    ;decrement by
    SUM_BA    ;adding -1
    BEQ    Done    ;test loop counter
    TAB
    BNE    Top
    Done
```

Problem 7 : Solution:

Note: This is **NOT** a proper ASM. Only true signals should be in the rectangles and ovals.

