EEL 3701 – Digital Logic & Computer Systems

Final Exam – Fall Semester 2006 Name:________________________________

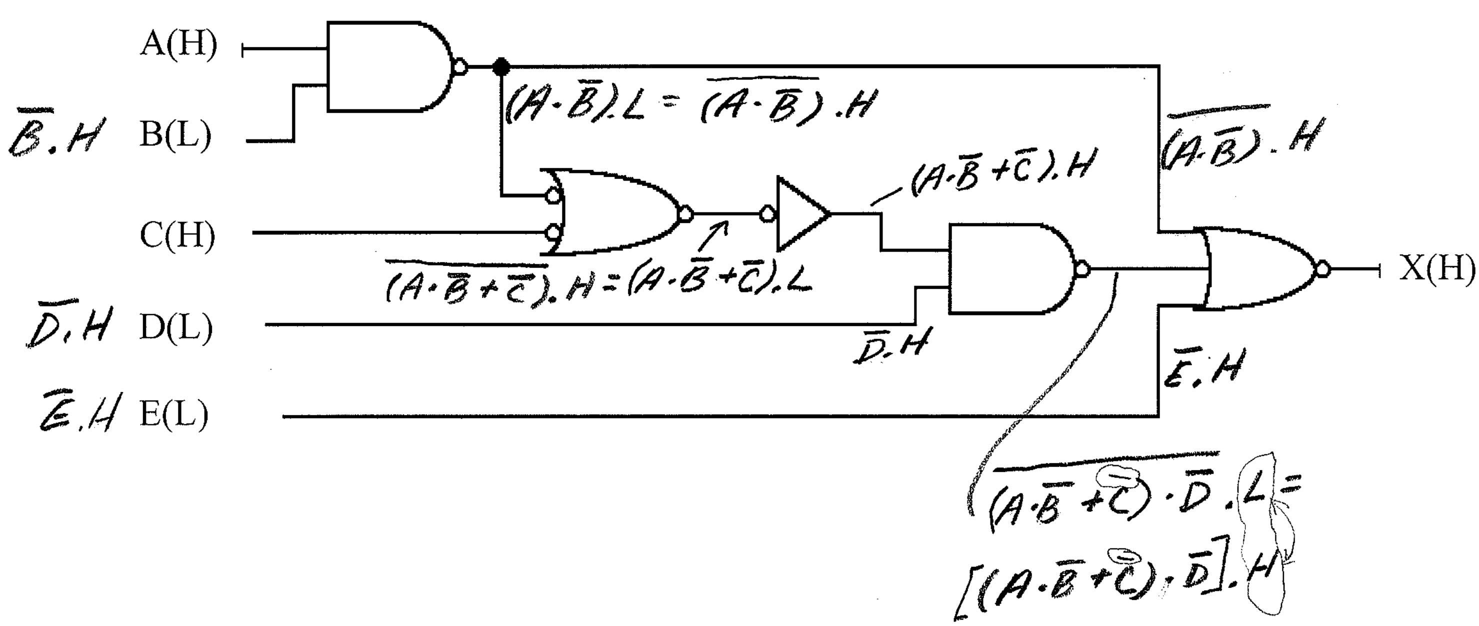
Remember to show <u>ALL</u> work here and in <u>EVERY</u> problem on this exam.

[10%] **1. <u>Circuit Analysis</u>**

What is the logic equation for X in the given circuit? Do **<u>not</u>** simplify or transform it into an SOP or POS form. Leave the logic expression as it is after analysis. Also, draw the **<u>intermediate</u>** expression at the **<u>input</u>** to **<u>each</u>** gate.

- *Notation reminder: A(H) is the same as A.H*
- Boolean expression answers must be in **lexical order**, i.e., /A before A, A before B, etc.

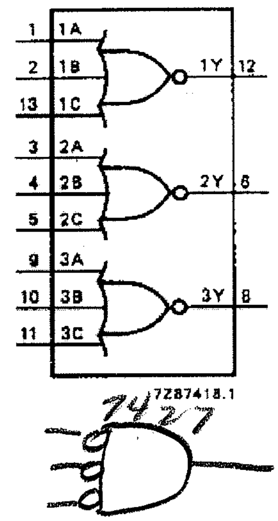
EQUATION: X = $\overline{\overline{(A\cdot\bar{B})} + \overline{(A\cdot\bar{B}+\bar{C})\cdot\bar{D}} + \bar{E}}$



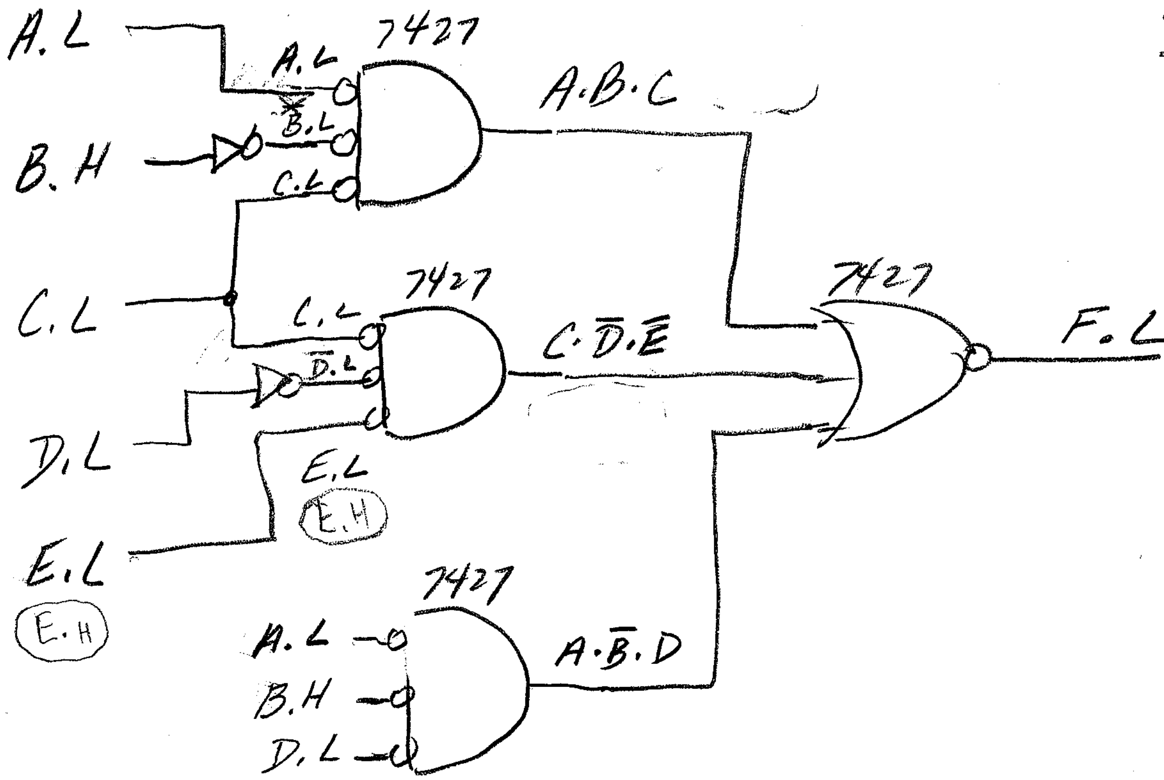
$X = \overline{\overline{(A\cdot B)} + \overline{(A\cdot B+\bar{C})\cdot\bar{D}} + \bar{E}}$

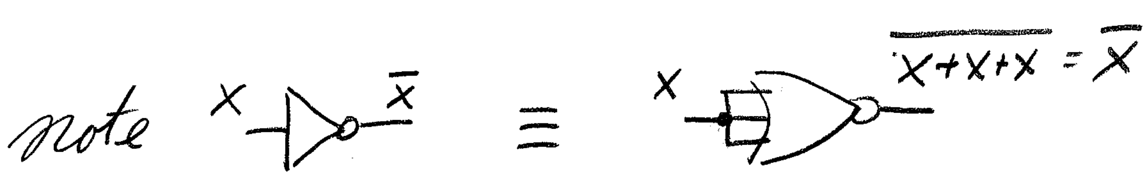
[10%] **2. Circuit Synthesis**

Draw a mixed-logic circuit diagram (with the minimum number of gates) to **directly implement** the below equation. All inputs and the output can be of any activation-level desired. Be sure to **specify the desired activation levels**. Do **not** simplify this equation. You may **only use** gates available on 74HC27 chips (shown). Use as many 74HC27 chips as you need, but use the minimum number required to solve this problem.

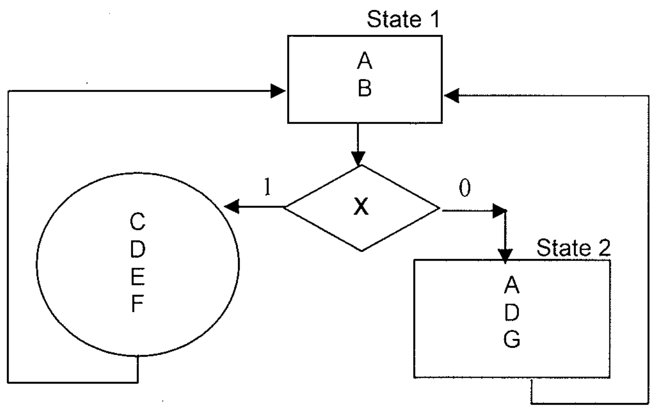


$$F=A*B*C + C*/D*/E + A*/B*D$$





[12%] 3. Implementation of an ASM chart using J-K flip-flops

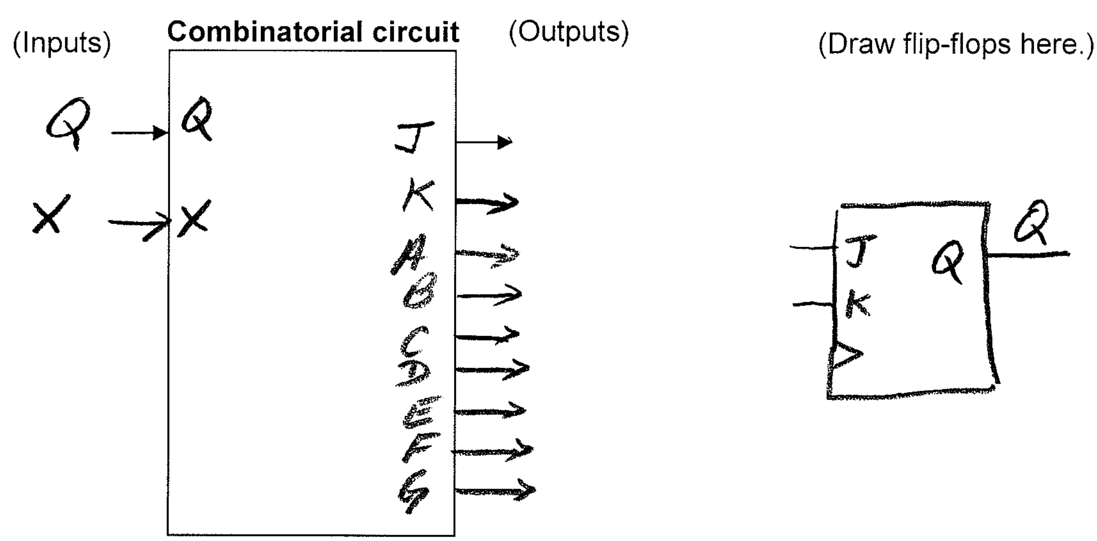


J-K characteristic table:

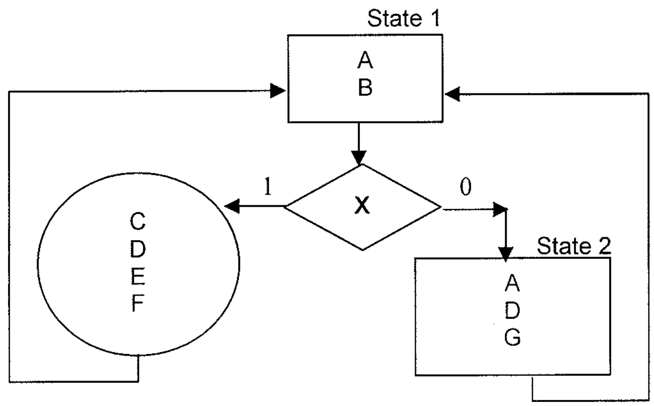
J	K	Q	Q+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

(a) Given the above ASM chart, complete the following block diagram of its implementation using the minimum number of J-K flip-flops: (2%)

- Determine how many J-K flip-flops that are needed. 1
- Draw in all the inputs and outputs of the combinatorial circuit.
- Make all necessary connections to complete the block diagram (between the combinatorial circuit and the flip-flops).



3. (continued) (ASM chart is repeated here for your convenience/)



J-K characteristic table:

J	K	Q	Q+	
0	0	0	0	hold
0	0	1	1	
0	1	0	0	clear
0	1	1	0	
1	0	0	1	set
1	0	1	1	
1	1	0	1	toggle
1	1	1	0	

(b) Finish the implementation of the ASM by determining the **minimum sum-of-products (MSOP)** logic expressions for the following signals: (10%)

J = $Q + \overline{X} \rightarrow \overline{X}$

K = 1

A = 1

B = $\overline{Q}$

C = $\overline{Q} \cdot X = \overline{Q + \overline{X}}$

D = $Q + X$

E = $\overline{Q} \cdot X$

F = $\overline{Q} \cdot X$

G = Q

Q	Q+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

(If necessary, use the bottom/back of the previous page to do your work.)

Q=0: State 1
Q=1: State 2

Q	X	Q+	J	K	A	B	C	D	E	F	G
0	0	1	1	X	1	1	0	0	0	0	0
0	1	0	0	X	1	1	1	1	1	1	0
1	0	0	X	1	1	0	0	1	0	0	1
1	1	0	X	1	1	0	0	1	0	0	1

[12%] **4. Given the following program segment, EQU statements, and contents of memory locations:**

```
X=000C    ORG      0
A=$96   1 LDX      #$000C          Data0  EQU  $0026
B=$3C   2 LDAA     #150;           Data1  EQU  $0032
A=$D2   3 LDAB     #60;            Data2  EQU  $0037
        4 SUM_BA                   Data3  EQU  $0038
        5 "next instruction        Data4  EQU  $0050
           (~~a~~,~~b~~,c,d,e, or f)"      Data5  EQU  $00EF
           a b                     Data6  EQU  $5000
                                   Data7  EQU  $73AB
                                   Data8  EQU  $73AC
```

Memory Locations

Address	Content
$0026	$66
$0032	$BB
$0037	$74
$0038	$EC
$0050	$C3
$00EF	$AB
$5000	$3E
$73AB	$AA
$73AC	$E2

a) Assume the 4 instructions in the above program segment have already been executed and the "next instruction" (i.e., 5th. instruction) is **LDAA 50**. Hand-assemble the LDAA instruction and fill in the blanks (including EA and register A) using <u>**HEX**</u>.

Note: EA is the "effective address", the actual address in memory where the data is loaded from in a "load" instruction (like LDAA, LDX, etc), or where the data is stored in a "store" instruction (like STAA). If no memory is accessed, write "none" for EA.

ADDRESS	INSTRUCTION	HEX ADDRESS	HEX VALUE
$0008	LDAA 50	$0008	04
EA = $0032		$0009	32 (decimal 50)
A = $BB		$000A	00
		$000B	

Repeat the same problem <u>if</u> the "next instruction" is the instruction in "b", "c", "d", "e", **or** "f", again assuming instructions 1 through 4 have been executed.

b) $0008 LDX Data7 LDX $73AB

EA = $73AB

X = $E2AA (little endian)

HEX ADDRESS	HEX VALUE
$0008	0A
$0009	AB
$000A	73
$000B	

c) $0008 STAA 38,X

EA = $0032

Value stored = $D2

000C + 26 = 0032 (12 + 38 = 50)

HEX ADDRESS	HEX VALUE
$0008	10
$0009	26 (38 decimal)
$000A	
$000B	

d) $0008 LDY #Data7 LDY #$73AB

EA = ~~$[illegible]~~ $0009

Y = ~~$[illegible]~~ $73AB

HEX ADDRESS	HEX VALUE
$0008	09
$0009	AB
$000A	73
$000B	

e) $0008 BN $EF

EA = none

PC after this instruction = $00EF (D2 is negative)

HEX ADDRESS	HEX VALUE
$0008	22
$0009	EF
$000A	
$000B	

f) $0008 BEQ $EF

EA = none

PC after this instruction = $000A

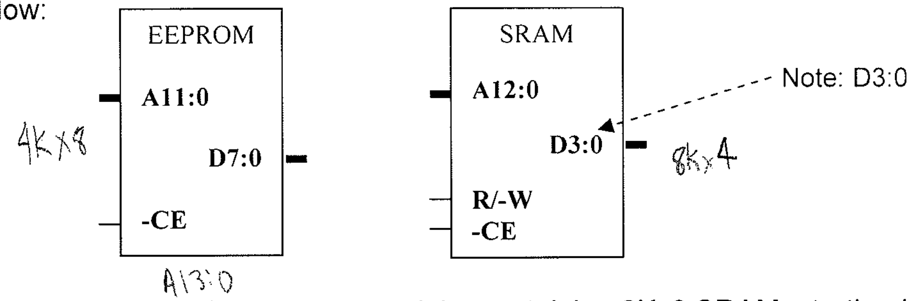
(D2 is not equal to 0)

HEX ADDRESS	HEX VALUE
$0008	20
$0009	EF
$000A	
$000B	

Name:______________________________

[10%] **5. EEPROM and SRAM**

You are given as many of the following EEPROM and SRAM devices that you need for the problem below:

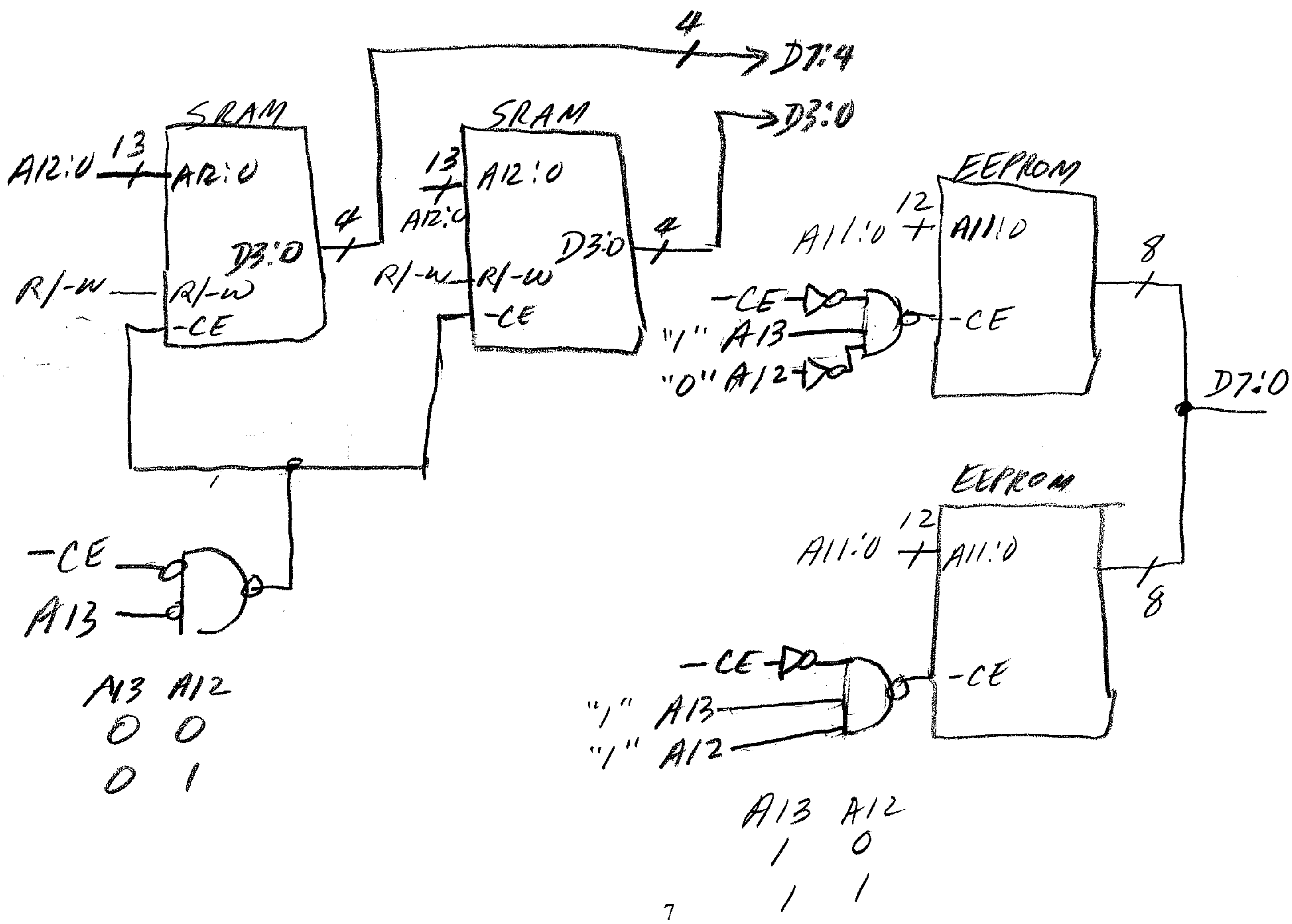


You are to implement a 16Kx8 memory module, containing 8Kx8 SRAM, starting in location 0, followed immediately by 8Kx8 EEPROM.

The 16K X 8 memory module should have an "R/-W" input and a "–CE" input.

For ease of grading, draw the. 8Kx8 SRAM circuit here (i.e., show the devices & connections):

Draw the 8Kx8 EEPROM circuit here (i.e.,show the devices & connections):



EEL 3701 – Digital Logic & Computer Systems

Final Exam – Fall Semester 2006 Name:________________________________

[15%] **6. Program Execution** (Use the G-CPU information in the back of the test.)

Given the following program in EPROM memory, fill out the following cycle table that illustrates its execution:

Addr	Data	
0	01	TBA
1	07	STAB $0701
2	01	
3	07	

Using the the G-CPU Controller ASM & Block Diagram (at the end of this test), complete the cycle table below. **Use as many rows as you need.**

	Cycle#	R/-W	Addr Sel1:0	PC (Hex)	MAR	A15:0 (Hex)	Data (Hex)	IR (Hex)	A (Hex)	B (Hex)	
state 00	1	1	00	0000	0400	0000	01	06	32	16	TBA
state 01	2	1	00	0000	↓	0000	01	01	32	16	
state 00	3	1	00	0001		0001	07	01	16	16	STAB $0701
state 01	4	1	00	0001		0001	07	07	16	16	
state 02	5	1	00	0002		0002	01	07	16	16	
state 0E	6	1	00	0003	04 01	0003	07	07	16	16	
state 0F	7	0	01	0004	0701	0701	16	07	16	16	
	8										
	9										
	10										

MAR connected to addr

1 = memory read

0 = memory write

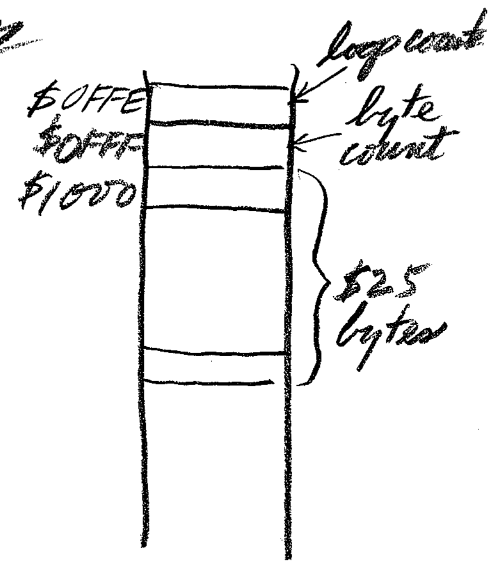
[15%] **7. GCPU Assembly Programming** (Use the G-CPU instruction set in back of this test.)

Write an assembly language program using **only** the G-CPU instructions

- There are $25 bytes already stored in memory starting in Location $1000
- Your program should go through the $25 bytes and count the number of "non-zero" bytes (i.e., bytes that are not equal to $00).
- At the end of the program, Memory Location $0FFF should contain the count of non-zero bytes.
- To increase your chances of partial credit, comment your program.

~~**Important restriction:** You have to use REGA as the "loop counter".~~

```
      LDAA #$25
      STAA $0FFE ; init loop count
      LDX #$1000 ; init index
      LDAA #$00   } init byte
      STAA $0FFF  } count
AGAIN: LDAA 0,X ; load next byte
      BZ NEXT
      LDAA $0FFF  }
      LDAB #$01   } increment
      SUM_BA      } byte count
      STAA $0FFF  }
NEXT: INX
      LDAA $0FFE  }
      LDAB #$FF   } decrement
      SUM_BA      } loop count
      STAA $0FFE  }
      BNE AGAIN
FIN: BRA FIN
```



[16%] **8. Controller (ASM) design for the G-CPU.** (Use the G-CPU block diagram and ASM charts in the back of the test.)

Using the signal names shown in controller G-GPU block diagram in the back of the test, complete the ASM chart (on the next page) to implement the following 3 instructions (**AND_BA, LDAA $addr,** and **BEQ $addrL**), including the completion of State A and State B.

Notes:

(1) In each state and conditional output, specify **only the signals that should be true.**

(2) However, you should use the notation: **MSA**=01, **MSB**=10, **MSC**=000.

(3) The default actions for each state should be specified to **"hold" REGA and REGB** and **OUT = REGA.**

(4) For your convenience, the required values for MSA, MSB, and MSC are given below:

Table 1: Input source MUXs for Registers A and B.

MSA1/ MSB1	MSA0/ MSB0	Bus Selected as Input to REGA/REGB
0	0	INPUT Bus
0	1	REGA Bus
1	0	REGB Bus
1	1	OUTPUT Bus

Table 2: ALU function selection MUX. (MUX C)

MSC2:0	Action
000	REGA Bus to OUTPUT Bus
001	REGB Bus to OUTPUT Bus
010	complement of REGA Bus to OUTPUT Bus
011	bit wise AND REGA/REGB Bus to OUTPUT Bus
100	bit wise OR REGA/REGB Bus to OUTPUT Bus
101	sum of REGA Bus & REGB Bus to OUTPUT Bus
110	shift REGA Bus left one bit to OUTPUT Bus
111	shift REGA Bus right one bit to OUTPUT Bus **without** sign extension

Put the solution on the next page.

8 (continued): Put solution here:

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