EEL 3701 – Digital Logic & Computer Systems

Final Exam – Fall Semester 2006 Name:_____

COVER SHEET:

Prob.	Points:	
1		(10)
2		(10)
3		(12)
4		(12)
5		(10)
6		(15)
7		(15)
8		(16)

Total: (100)

Re-Grade Information:

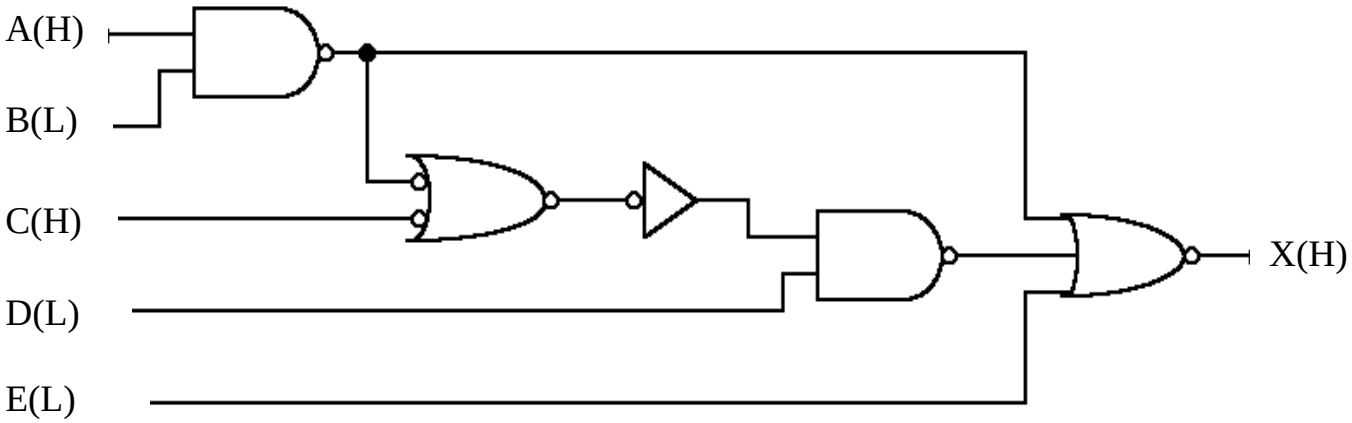
Remember to show ALL work here and in EVERY problem on this exam.

[10%] **1. Circuit Analysis**

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

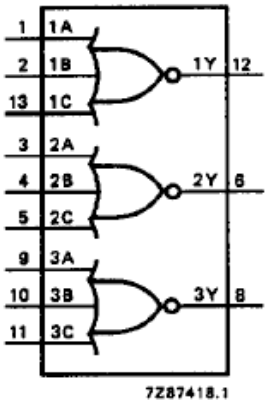
- *Notation reminder: A(H) is the same as A.H*
- Boolean expression answers must be in **lexical order**, i.e., /A before A, A before B, etc.

EQUATION: X = _____



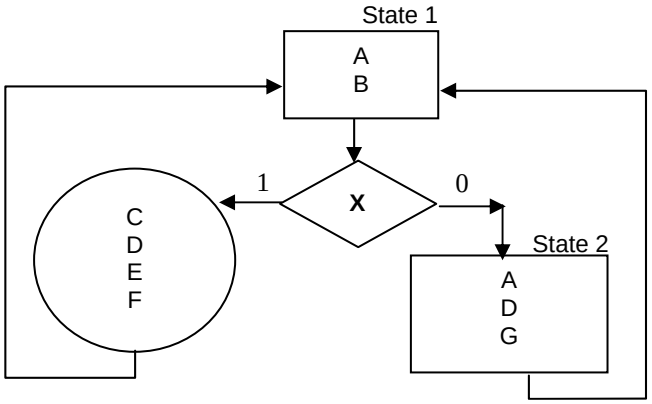
[10%] **2. Circuit Synthesis**

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. <u>Implementation of an ASM chart using J-K flip-flops</u>**

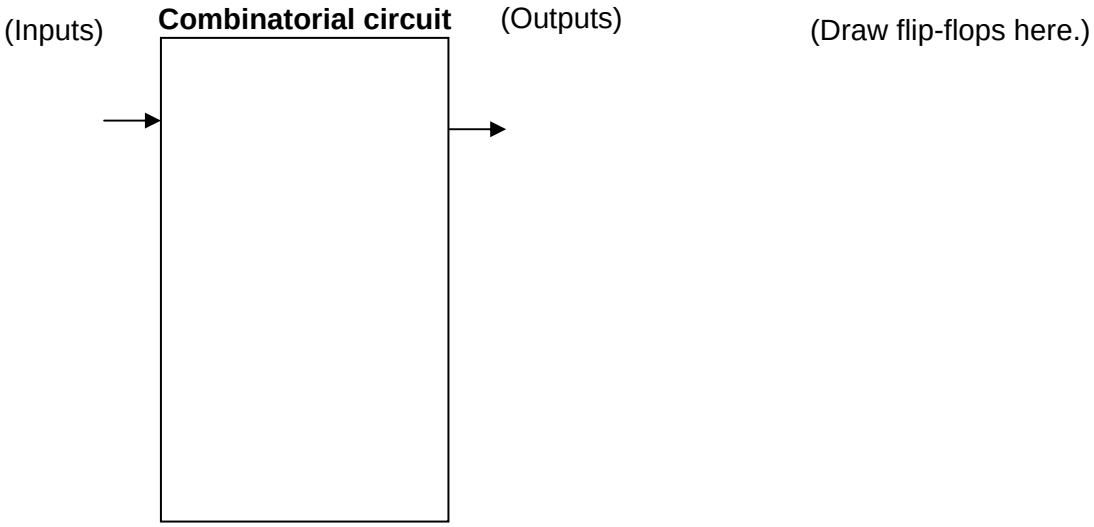


J-K characteristic table:

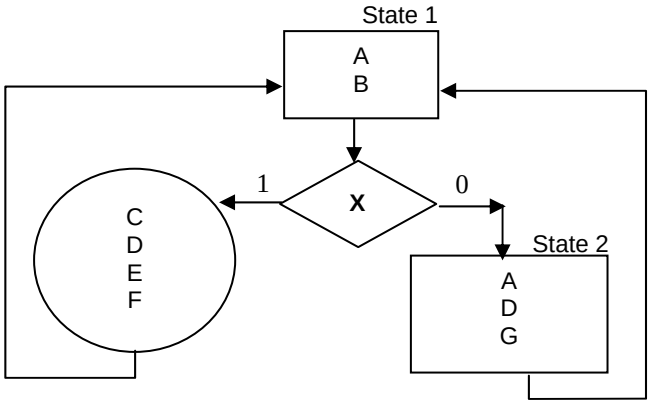
J	K	Q	Q+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

(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.
- 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
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

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

J =

K =

A =

B =

C =

D =

E =

F =

G =

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

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

```
    ORG     0
1   LDX     #$000C
2   LDAA    #150;
3   LDAB    #60;
4   SUM_BA
5   "next instruction
    (a,b,c,d,e, or f)"
```

```
Data0  EQU  $0026
Data1  EQU  $0032
Data2  EQU  $0037
Data3  EQU  $0038
Data4  EQU  $0050
Data5  EQU  $00EF
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 **HEX**.

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	_____
EA = _____		$0009	_____
A = _____		$000A	_____
		$000B	_____

Repeat the same problem if 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
EA = _____
X = _____

HEX ADDRESS	**HEX** VALUE
$0008	_____
$0009	_____
$000A	_____
$000B	_____

c) $0008 STAA 38,X
EA = _____
Value stored = _____

HEX ADDRESS	**HEX** VALUE
$0008	_____
$0009	_____
$000A	_____
$000B	_____

d) $0008 LDY #Data7
EA = _____
Y = _____

HEX ADDRESS	**HEX** VALUE
$0008	_____
$0009	_____
$000A	_____
$000B	_____

e) $0008 BN $EF
EA = _____
PC after this instruction = _____

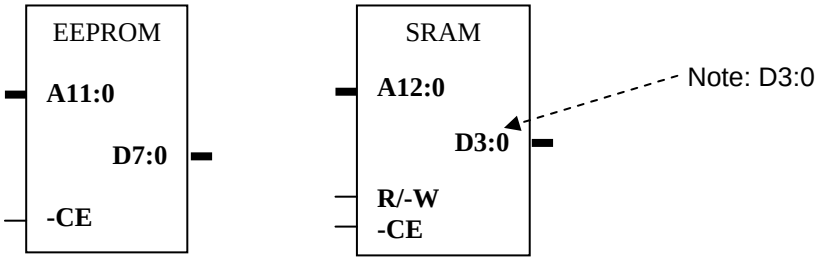
HEX ADDRESS	**HEX** VALUE
$0008	_____
$0009	_____
$000A	_____
$000B	_____

f) $0008 BEQ $EF
EA = _____
PC after this instruction = _____

HEX ADDRESS	**HEX** VALUE
$0008	_____
$0009	_____
$000A	_____
$000B	_____

[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):

[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
1	07
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)
1	1	00	0000	0400			06	32	16
2									
3									
4									
5									
6									
7									
8									
9									
10									

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

Write an assembly language program using **<u>only</u>** 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 **<u>have to use REGA</u>** as the "loop counter"**.**

[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: