EEL 3701 – Introduction to Digital Design

Final Exam – Fall Semester 2005 Name:_____

COVER SHEET:

Prob.	Points:	
1		(10)
2		(14)
3		(14)
4		(18)
5		(15)
6		(12)
7		(17)

Total: (100)

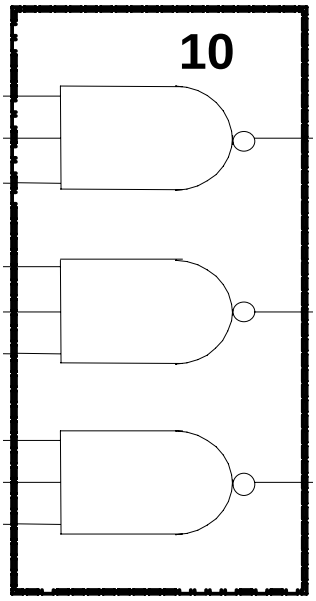
Re-Grade Information:

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

(10 pts.)

1. Draw a mixed-logic circuit diagram (with the minimum number of gates) to **<u>directly implement</u>** the below equation. All inputs and the output can be of any activation-level desired. Be sure to **specify the desired activation levels**. Do **<u>not</u>** simplify this equation. You may **only use** gates available on 74HC10 chips (shown). Use as many 74HC10 chips as you need, but use the minimum number required to solve this problem.

$$\overline{A\overline{E}\ \overline{(\overline{B}+C)}\ D} = Y$$



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^+
0	**0**	**0**	**0**	0	0	1	1	1
0	**0**	**0**	**1**	0	0	0	0	0
0	**0**	**1**	**0**	0	0	0	0	1
0	**0**	**1**	**1**	1	1	0	1	0
0	**1**	**0**	**0**	X	X	X	X	X
0	**1**	**0**	**1**	X	X	X	X	X
0	**1**	**1**	**0**	1	0	0	1	1
0	**1**	**1**	**1**	1	1	1	1	0
1	**0**	**0**	**0**	0	0	1	1	1
1	**0**	**0**	**1**	0	0	1	1	1
1	**0**	**1**	**0**	0	0	1	1	1
1	**0**	**1**	**1**	1	0	1	1	1
1	**1**	**0**	**0**	X	X	1	1	1
1	**1**	**0**	**1**	X	X	1	1	1
1	**1**	**1**	**0**	1	0	1	1	1
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.)

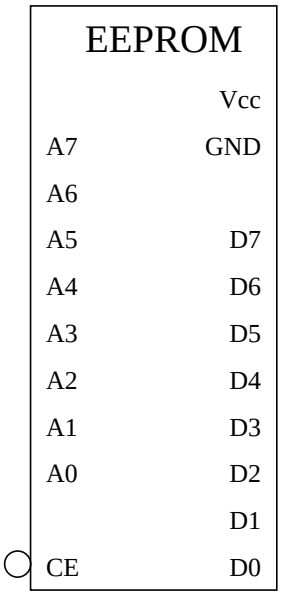
Z: (1 pt.)

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.)

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

(14 pts.)

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

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:

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

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

Addr (Hex)	Data (Hex)	Addr (Hex)	Data (Hex)	Addr (Hex)	Data (Hex)

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

_____ Hex

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

_____ Hex

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

_____ Hex

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

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

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

(18 pts.)

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

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

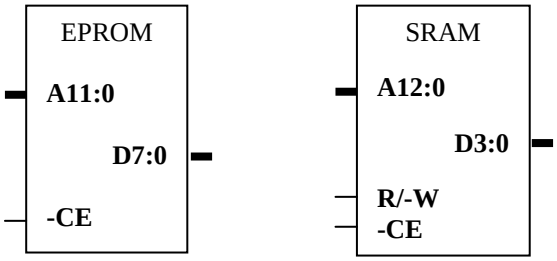
Using the the G-CPU Controller ASM & Block Diagram (Appendix A handouts), complete the cycle table below:

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								
3								
4								
5								
6								
7								
8								
9								
10								

5. Memory Maps, Memory Devices & Decoding

(15 pts.)

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



If the processor is a GCPU with a 16 bit address bus (A15:0) and 8 bit data bus (D7:0), show the required **devices & decode circuitry** below to place an 8Kx8 EPROM starting **at 2000 Hex** and an 8Kx8 SRAM immediately following the EPROM in the system memory map.

Implement the 8Kx8 EPROM. Show devices & connections (below) =>

Implement the 8Kx8 SRAM. Show devices & connections (below) =>

Show EPROM Memory Decode Circuit (below) =>

Show SRAM Decode Circuit (below) =>

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.

(17 pts.)

7. **Controller (ASM) design for the G-CPU. (Use the G-CPU block diagram and ASM charts in the appendix.)**

Using the signal names shown in controller G-GPU block diagram in the appendix, complete the ASM chart (on the next page) to implement the following 3 instructions (**SUM_BA, LDAB #data,** and **STAA $addr**), 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) When not specified, the default actions for each state is to **"hold" REGA and REGB** and **OUT = REGA.**

Important Hint: You should use the ASM charts provided in the appendix. Then, all you have to do is to determine what signals are supposed to be TRUE in each state.

MSA1/MSB1	MSA0/MSB0	
0	0	INPUT Bus
0	1	REGA Output Bus
1	0	REGB Output Bus
1	1	OUTPUT Bus

MSC2:0 (Most Significant Bit is on the left)

000 = > REGA Bus to OUTPUT Bus
001 = > REGB Bus to OUTPUT Bus
010 = > bit wise AND REGA/REGB Bus to OUTPUT Bus
011 = > bit wise OR REGA/REGB Bus to OUTPUT Bus
100 = > complement of REGA Bus to OUTPUT Bus
101 = > REGA Bus Plus REGB Bus Plus Cin to OUTPUT Bus & Cout (There is an external Cin and an external Cout for the ALU)
110 = > shift REGA Bus left one bit to OUTPUT Bus (0 is shifted into OUTPUT Bus[0].)
111 = > shift REGA Bus right one bit to OUTPUT Bus (0 is shifted into OUTPUT Bus[3].)

Put the solution on the next page.

8 (continued): Put solution here: