EEL 3701 – Digital Logic & Computer Systems

Final Exam – Fall Semester 2008 Name:_____

COVER SHEET:

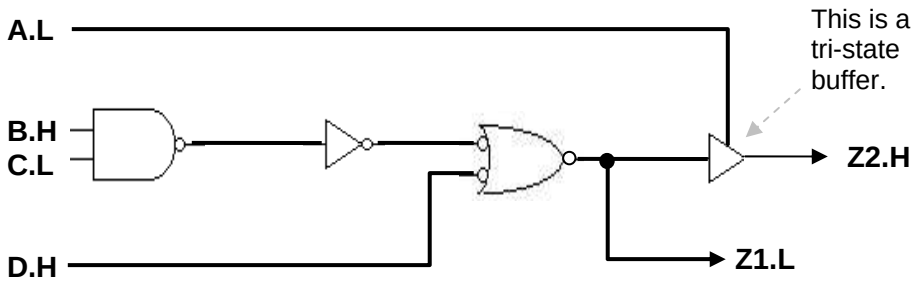
Prob.	Points:	
1		(8)
2		(10)
3		(12)
4		(10)
5		(12)
6		(12)
7		(12)
8		(12)
9		(12)

Total: (100)

Re-Grade Information:

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

[8%] **1. Circuit Analysis**



NOTE: To obtain **partial credit**, label intermediate signals.

(a) Analyze the above circuit and derive the ***logic equation*** for Z1. (4 pts.)

Do not reduce or transform the logic expression (except for the "double inversions"). Put the ***logic*** equation for the Z1 "as it is".

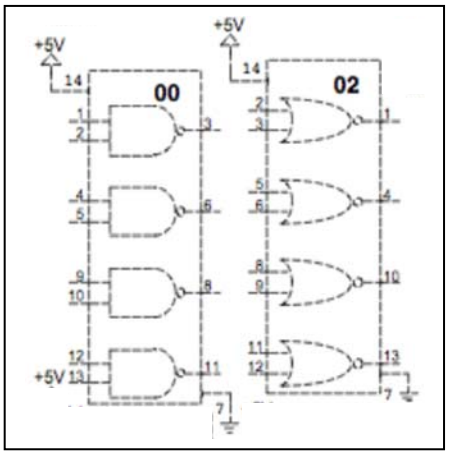
Z1 =

(b) Complete the following VOLTAGE table for Z2. Use "L" for low voltage, "H" for high voltage, "(Z)" for high impedance. Also use "X" for don't cares (and wild cards) to reduce the table size. (4 pts.)

A	B	C	D	Z2

[10%] **2.** You have five **total** gates available from these two chips, a single 74'00 and a single 74'02 (i.e., 5 total gates from any combination of 74'00 and 74'02 gates). **Draw a mixed-logic circuit diagram** for the below equation using **only the available gates**. Do **not** simplify this equation. Specify the desired activation levels to accomplish the required design and **add appropriate pin numbers**. (Check carefully that you see all the horizontal lines (NOT operators) in the equation.)

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



A( )

B( )

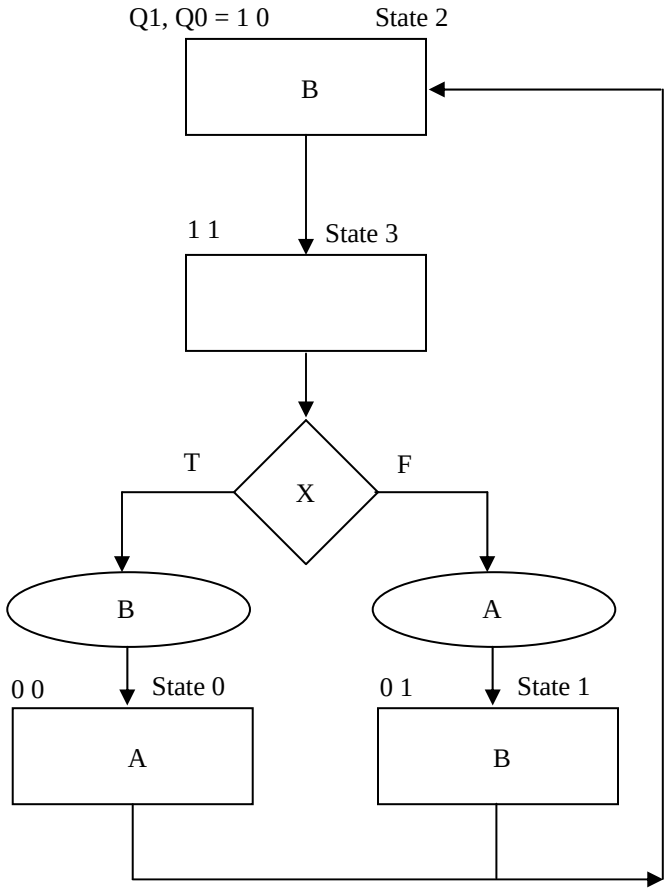
C( )

D( )

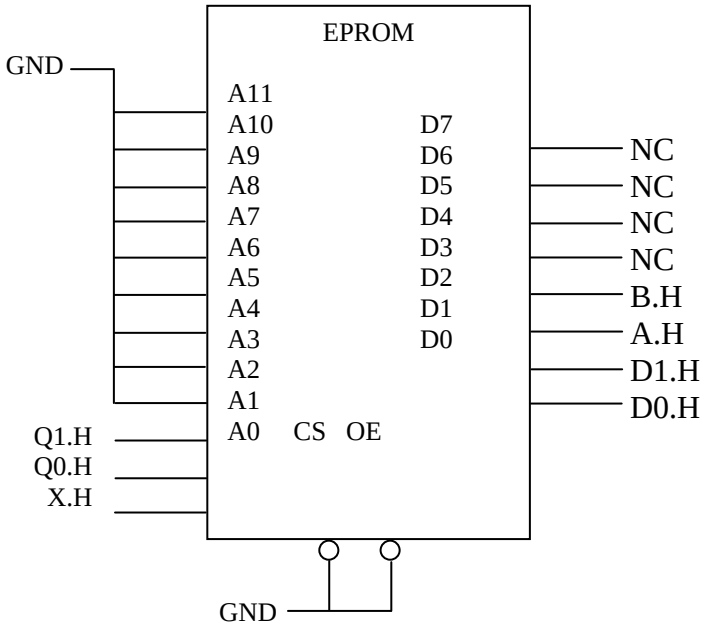
Y( )

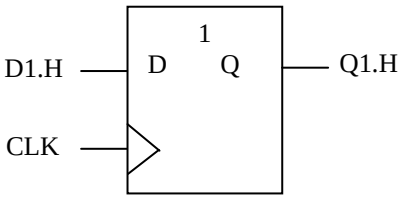
[12%] **3. ASM Implementation**

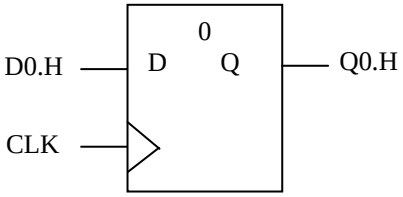
The ASM Flow Chart below is implemented in the EPROM based system below:



NC = No Connect







[3%] **3A.** Create a **Next State (logic) Table** for the design to answer questions on the next page:

[3%] 3B. Show the EPROM Addresses and corresponding Data that must be programmed for this design: **Use '0' for any don't cares or wild card values.**

Address (Hex)	Data (binary)	Data (Hex)

[3%] 3C. If X.H is changed to X.L, show the EPROM Address and corresponding Data that must be programmed for this new design:

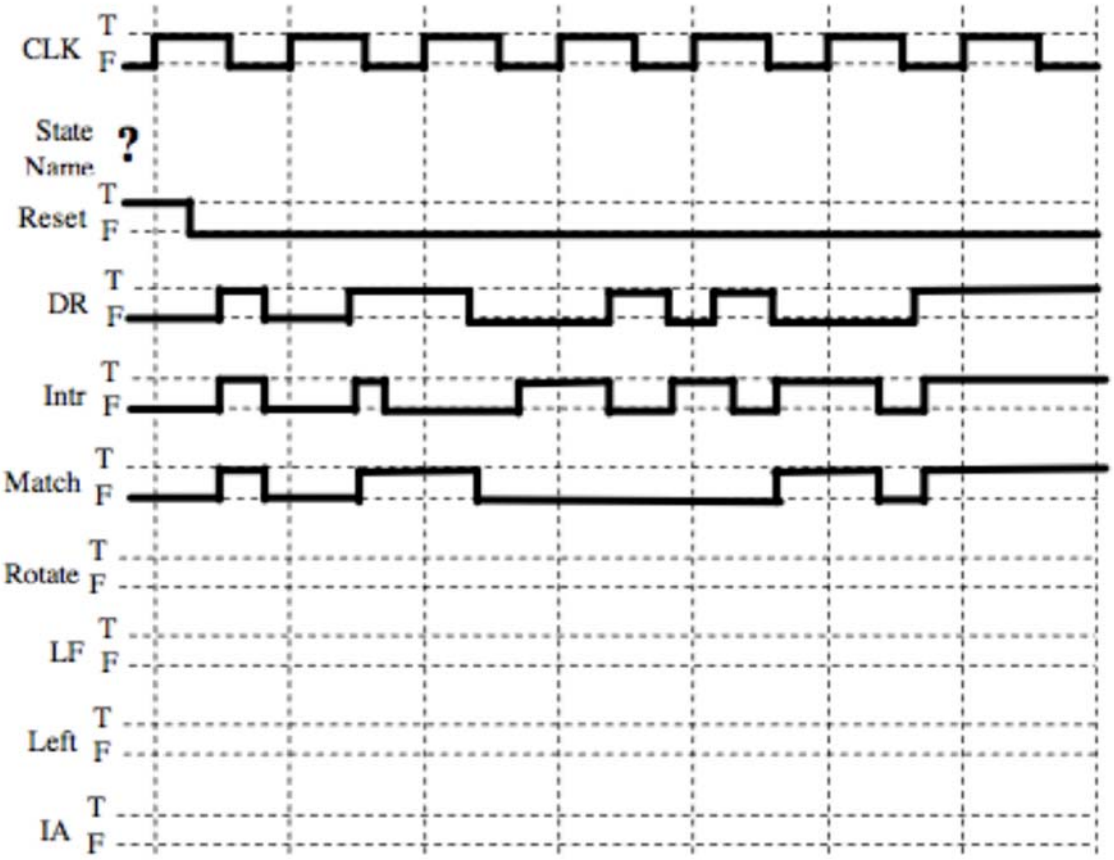
Address (Hex)	Data (binary)	Data (Hex)

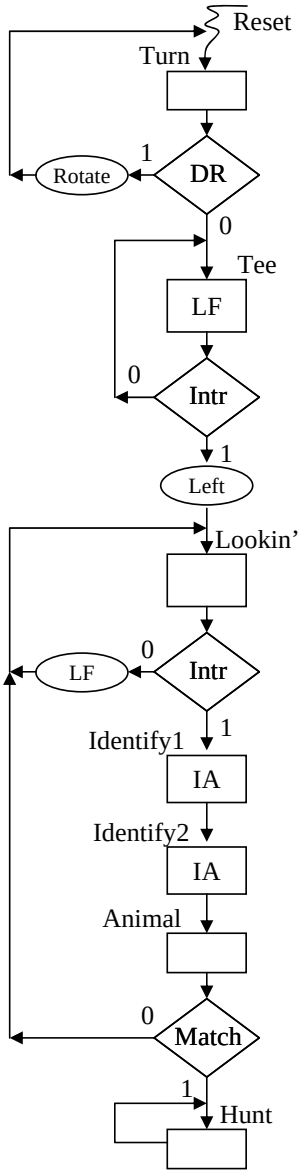
[3%] 3D. If A.H & B.H are changed to A.L & B.L (and X.H), show the EPROM Address and corresponding Data that must be programmed for this new design:

Address (Hex)	Data (binary)	Data (Hex)

[10%] **4. ASM Analysis**

Complete the timing diagram for **this** ASM. Show small propagation delays. Assume that each of the flip-flops used are rising-edge triggered. This system has an asynchronous reset (**Reset)** to the state "Turn"**.** What is the **next state** to the after the below timing diagram below ends? [Abbreviations: DR=Done Rotation, LF=Line Follow, Intr=Intersection, IA=Identify Animal]





[12%] **5. GCPU Assembly Programming**

Write an assembly language program using **only** the G-CPU instructions (in back of this test)

- There is a non-zero byte of data (INBYTE) already stored in Location $1000.
- Also there are an unknown number of bytes already stored in memory starting in Location $1001 (equated to TABLE). All bytes in the table have non-zero value except for the last byte (i.e., last byte is $00).
- Complete the following program to do the following to each byte in the table:
  - add INBYTE to the byte in the table if the byte in the table is positive.
  - do nothing to the byte if the byte in the table is negative, but add 1 to COUNT.
- At the end of the program, COUNT should contain number of negative numbers in the table. Perform an "infinite loop" to stop the program.
- To increase your chances of partial credit, comment your program.

```
COUNT EQU $0FFF
INBYTE EQU $1000
TABLE EQU $1001
ORG _____  ; You should ORG your program at an appropriate memory location
```

[12%] **6. Given the following program segment and ORG, EQU and DC.B pseudo-ops:**

```
        ORG     0
BACK:   LDX     #$1000
        LDAA    #$41
        TAB
        LDAA    0,X
        "next instruction
         (a,b,c,d,e, or f)"
```

```
L1  EQU  $1000
L2  EQU  $1006
L3  EQU  $03
    ORG  $4000
    DC.B $54
    ORG  $4008
L4   DC.B $92
```

```
    ORG  $1005
    DC.B $8A
    DC.B $FF
    DC.B $04
    ORG  $1000
    DC.B $37
    DC.B $56
L5  DC.B $84
    DC.B $22
```

a) Assume the 4 instructions in the above program segment have already been executed and the "next instruction" (i.e., 5th. instruction) is **LDAA #L3**. 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 #L3	$0008	_____
EA = _____ (All answers in hex)		$0009	_____
A = _____ (hex)		$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	LDAA 7,X	$0008	_____
EA = _____		$0009	_____
A = _____		$000A	_____
		$000B	_____
c) $0008	STAA L5	$0008	_____
EA = _____		$0009	_____
Value stored = _____		$000A	_____
		$000B	_____
d) $0008	LDY #L2	$0008	_____
EA = _____		$0009	_____
Y = _____		$000A	_____
		$000B	_____
e) $0008	BNE L3	$0008	_____
EA = _____		$0009	_____
PC after this		$000A	_____
instruction = _____		$000B	_____
f) $0008	BEQ BACK	$0008	_____
EA = _____		$0009	_____
PC after this		$000A	_____
instruction = _____		$000B	_____

[12%] **7. <u>Controller (ASM) design for the G-CPU.</u>** (Use the G-CPU block diagram and ASM charts in the back of the test.)

Using the <u>signal names shown in controller</u> G-GPU block diagram in the back of the test, complete the ASM chart (on the next page) to implement the following 3 instructions (**LDX #data, STAA $addr,** and **SUM_A $addr**), including the <u>completion of State A and State B</u>.

Notes:

(1) SUM_A $addr is a new instruction (opcode = 33). The function of this instruction is to add the content of memory location at $addr to the content of REGA. (You can use REGB if necessary).

(2) The ASM chart is specified <u>as logic</u>, not voltage. For example, since the R/-W signal is active low, specifying R/-W in a state box indicate it is "true" (which means a "low" voltage"

(3) You should specify the default actions for each state to **"hold" REGA and REGB** and **OUT = REGA.**

(4) For your convenience, the required values for MSA, MSB, and MSC are given below. For convenience of grading, you should use the notation: **MSA**=01, **MSB**=10, **MSC**=000.

(5) For all other signals, use the standard ASM notation of specifying only the signals that are true in each state and conditional output.

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

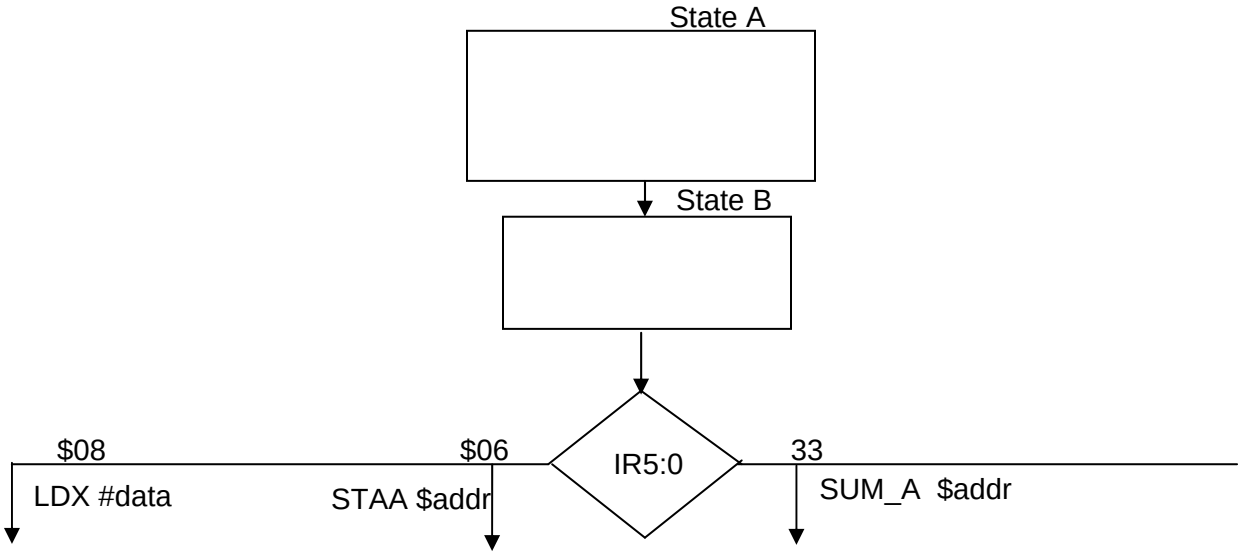
MSA1/ MSB1	MSA0/ MSB0	Bus Selected as Input to REGA/REGB
0	0	INPUT Bus
0	1	REGA Bus
1	0	REGB Bus
1	1	OUTPUT Bus

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.

7 (continued): Complete the ASM chart, including State A and State B.



[12%] **8. Program Execution** (Use the G-CPU information in the back of the test, including the controller flowchart.)

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

Addr	Data (Both Addr and Data are in hex.)
0000	02
0001	07
0002	05
0003	07
0004	20

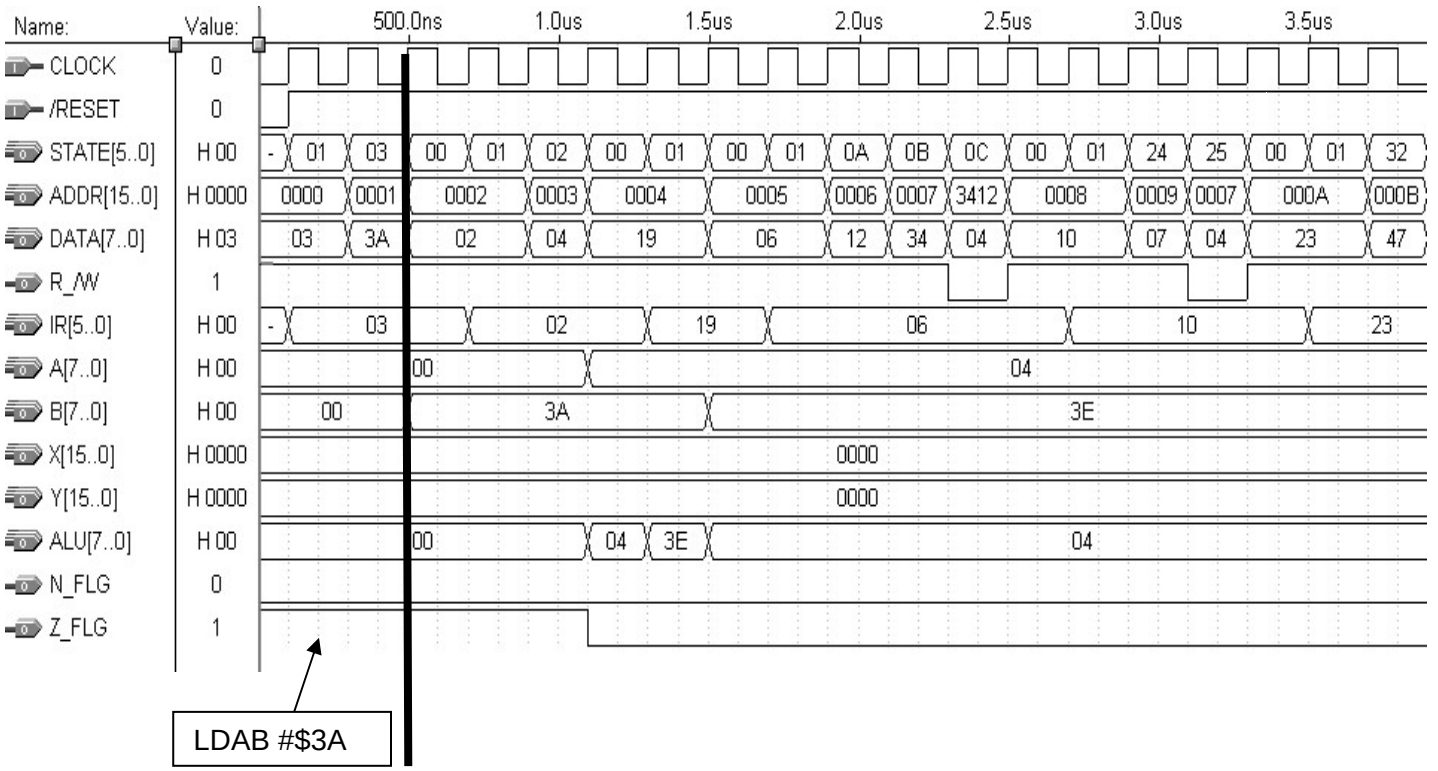
Using the the G-CPU Controller "ASM" chart (Flowchart) & Block Diagram (at the end of this test)**,** complete the cycle table below**. Use as many rows as you need and go as far as you can.**

Assume **R/-W** is set to "read from memory". Also assume location **$2007** contains **$99**.

Cycle #	State code	Addr Sel1:0	PC (Hex)	MAR	A15:0 (Hex)	Data (Hex)	IR (Hex)	A (Hex)	B (Hex)
1	00	00	0000	0400			06	32	16
2	01								
3									
4									
5									
6									
7									
8									
9									
10									

(From GPCU controller ASM (or flow) chart)

[12%] **9. Analysis of G-CPU simulation**



Finish the answer for Part (a) here.

(a) Analyze the above timing diagram and identify all the instructions being simulation. As illustrated above (in the partial answer LDAB #$3A), draw lines between each instruction and identify the instruction.

(b) Analyze the above timing diagram and determine the contents of the memory locations (as in a .mif file). Fill in the table on the right as much as needed.

Address (hex)	Content (hex)

University of Florida
Department of Electrical & Computer Engineering

EEL 3701
Revision 0

Drs. Gugel and Schwartz
13-Nov-06

Page 1/2

G-CPU Instruction Set

Data Movement Instructions:

Opcode	Instruction	Operand	Description	# of States
0	TAB	none	Transfer A to B (inherent addressing)	2
1	TBA	none	Transfer B to A (inherent addressing)	2
2	LDAA #data	8 bit data	Load A with immediate data (immediate addr.)	3
3	LDAB #data	8 bit data	Load B with immediate data (immediate addr.)	3
4	LDAA addr	16 bit address	Load A with data from memory location addr (extended addressing)	5
5	LDAB addr	16 bit address	Load B with data from memory location addr (extended addressing)	5
6	STAA addr	16 bit address	Store data in A to memory location addr (extended addressing)	5
7	STAB addr	16 bit address	Store data in B to memory location addr (extended addressing)	5
8	LDX #data	16 bit data	Load X with immediate data (immediate addr.)	4
9	LDY #data	16 bit data	Load Y with immediate data (immediate addr.)	4
A	LDX addr	16 bit addr	Load X with data from memory location addr. (extended addressing)	6
B	LDY addr	16 bit addr	Load Y with data from memory location addr. (extended addressing)	6
C	LDAA dd,X	8 bit displacement	Load A with data from memory location pointed to by X + dd (indexed addressing)	4
D	LDAA dd,Y	8 bit displacement	Load A with data from memory location pointed to by Y + dd (indexed addressing)	4
E	LDAB dd,X	8 bit displacement	Load B with data from memory location pointed to by X + dd (indexed addressing)	4
F	LDAB dd,Y	8 bit displacement	Load B with data from memory location pointed to by Y + dd (indexed addressing)	4
10	STAA dd,X	8 bit displacement	Store data in A to memory location pointed to by X + dd (indexed addressing)	4
11	STAA dd,Y	8 bit displacement	Store data in A to memory location pointed to by Y + dd (indexed addressing)	4
12	STAB dd,X	8 bit displacement	Store data in B to memory location pointed to by X + dd (indexed addressing)	4
13	STAB dd,Y	8 bit displacement	Store data in B to memory location pointed to by Y + dd (indexed addressing)	4

G-CPU Instruction Set

ALU Related Instructions:

Opcode	Instruction	Operand	Description	# of States
14	SUM_BA	none	Sum A, B and place in A (inherent addressing)	2
15	SUM_AB	none	Sum A, B and place in B (inherent addressing)	2
16	AND_BA	none	AND A, B and place in A (inherent addressing)	2
17	AND_AB	none	AND A, B and place in B (inherent addressing)	2
18	OR_BA	none	OR A, B and place in A (inherent addressing)	2
19	OR_AB	none	OR A, B and place in B (inherent addressing)	2
1A	COMA	none	Complement contents in A (inherent addressing)	2
1B	COMB	none	Complement contents in B (inherent addressing)	2
1C	SHFA_L	none	Shift A left by one bit (inherent addressing)	2
1D	SHFA_R	none	Shift A right by one bit (inherent addressing)	2
1E	SHFB_L	none	Shift B left by one bit (inherent addressing)	2
1F	SHFB_R	none	Shift B right by one bit (inherent addressing)	2
30	INX	none	Increment X (inherent addressing)	2
31	INY	none	Increment Y (inherent addressing)	2

Branch Instructions:

Opcode	Instruction	Operand	Description	# of States
20	BEQ	addrL	Branch if A = 0, i.e., Z Flag = 1 (absolute addressing)	3
21	BNE	addrL	Branch if A ≠ 0, i.e., Z Flag = 0 (absolute addressing)	3
22	BN	addrL	Branch if A is negative, i.e., N Flag = 1 (absolute addressing)	3
23	BP	addrL	Branch if A is positive (or zero), i.e., N Flag = 0 (absolute addressing)	3

Special Notes

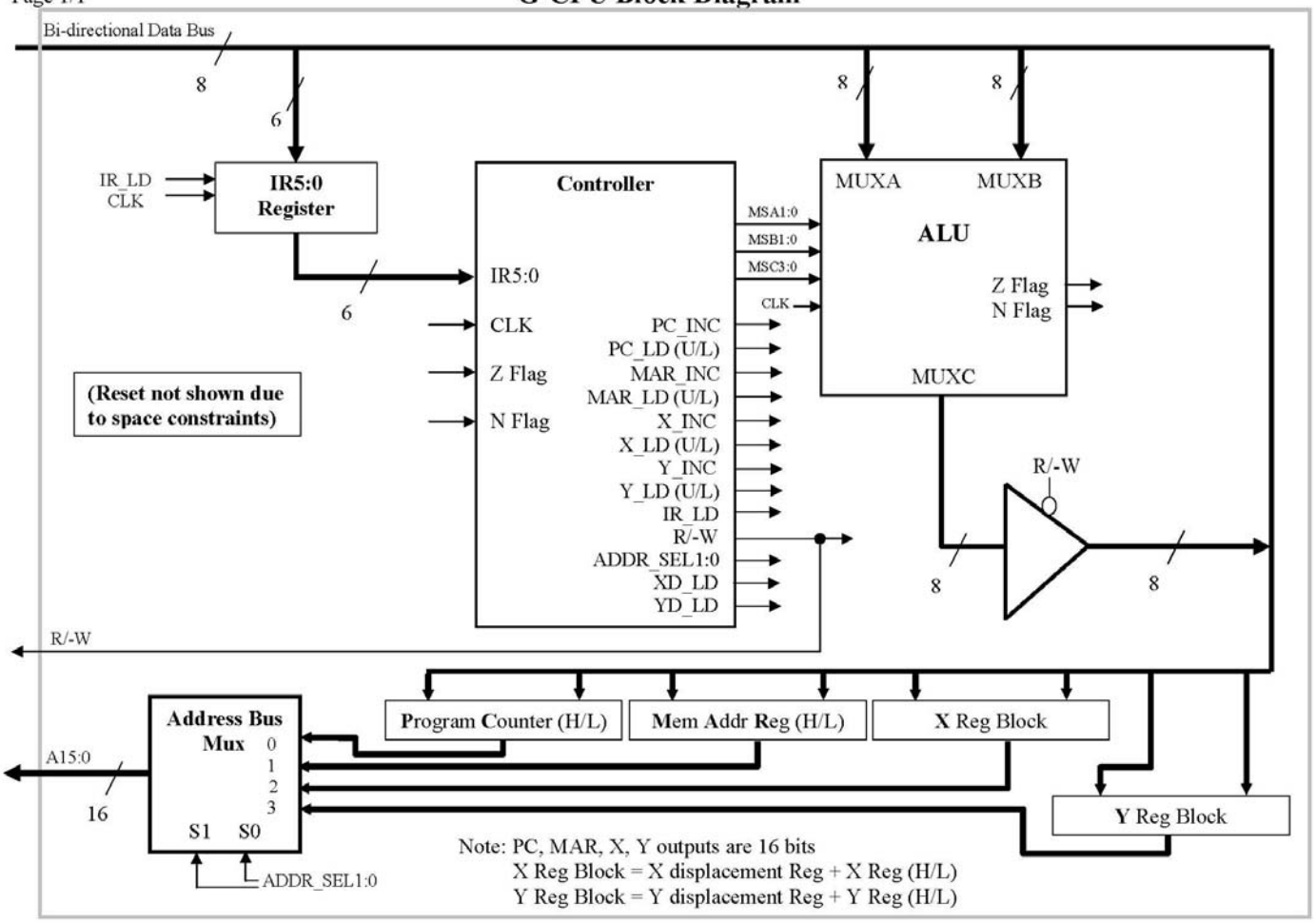
1. Z flag and N flag are only set and cleared by the contents in register A.
2. A branch is accomplished by moving the operand address "addr" to the lower byte of the PC. The upper byte of the PC remains unchanged after a branch.
3. The Branch Instructions use absolute addressing where only the low byte of the address is used as an operand. If the branch condition is met, the high byte of the PC is unchanged and the low byte takes the value of the operand (addrL).

University of Florida
Department of Electrical & Computer Engineering
Page 1/1

EEL 3701
Revision 0

Drs. Gugel and Schwartz
18-Nov-08

G-CPU Block Diagram

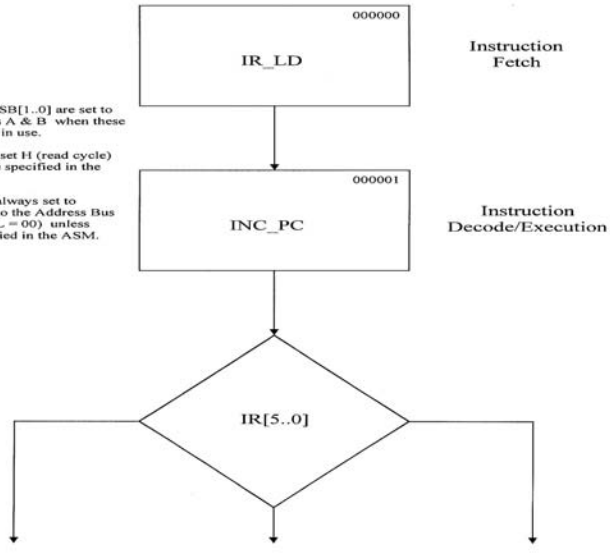


Page 1/6

G-CPU Controller Flow Charts

Special Notes:

1. MSA[1..0] & MSB[1..0] are set to protect registers A & B when these registers are not in use.
2. R_/W is always set H (read cycle) unless otherwise specified in the ASM chart.
3. ADDR_SEL is always set to connect the PC to the Address Bus (i.e. ADDR_SEL = 00) unless otherwise specified in the ASM.



Instructions Listed on Next Pages…

University of Florida
Department of Electrical & Computer Engineering
Page 2/6

EEL 3701
Revision **0**

Drs. Gugel and Schwartz
18-Nov-08

G-CPU Controller Flow Charts