ECE 270 Lab Verification / Evaluation Form

Experiment 13

Evaluation:

IMPORTANT! You must complete this experiment during your scheduled lab period. All work for this experiment must be demonstrated to and verified by your lab instructor *before the end* of your scheduled lab period.

STEP	DESCRIPTION	MAX	SCORE
Pre-lab 1	Augment printed copy of skeleton file	5	
Pre-lab 2	Hand-assemble program and determine test data/results	4	
Step 1	Demonstrate `INA`, `OUT`, `HLT` instruction sequence	3	
Step 2	Demonstrate `LDA`, {`ADD`/`SUB`/`AND`}, `STA` sequence	6	
Step 3	Demonstrate program execution	4	
Step 4	Thought questions	3	
	TOTAL	25	

Signature of Evaluator: _____

Academic Honesty Statement:

"In signing this statement, I hereby certify that the work on this experiment is my own and that I have not copied the work of any other student (past or present) while completing this experiment. I understand that if I fail to honor this agreement, I will receive a score of ZERO for this experiment and be subject to possible disciplinary action."

Printed Name:_____ Class No. __ __ __ __ - __

Signature:_____ Date: _____

The Raulmatic 716 Simple Computer

Instructional Objectives:

- To practice creating a simple computer

Prelab Preparation:

- Read this document in its entirety
- Review the referenced Module 4 lecture material
- Complete the pre-lab exercises

Experiment Description:

In lecture you have been introduced to the architecture and instruction set of a very simple computer. The internal organization and the basic functional units of this device have been discussed in detail. Several "extensions" to the basic machine have also been presented in class (e.g., input/output instructions, transfer of control instructions, a stack mechanism, and a subroutine linkage mechanism). The purpose of this experiment is to reinforce your understanding of basic computer organization and provide you with additional experience designing and testing CPLD-based digital systems.

The version of the "simple computer" realized in this experiment will utilize **7**-bit instructions and sport **16** memory (SRAM) locations – hence, the "**716**" designation. The 7-bit instructions will consist of a 3-bit opcode field and a 4-bit address. The ALU will be 4-bits wide, and utilize radix (2's complement) arithmetic. The ALU will function according to the description in Table I and generate the standard condition codes described in the Module 4 lecture notes.

Table I. ALU Function Table.

ALE	ALX	ALY	Mnemonic	Function Performed	CF	NF	ZF	VF
0	d	d	-	*stay in the same state*	-	-	-	-
1	0	0	ADD	(A) ← (A)+(*data bus*)	↕	↕	↕	↕
1	0	1	SUB	(A) ← (A)-(*data bus*)	↕	↕	↕	↕
1	1	0	AND	(A) ← (A)∩(*data bus*)	-	↕	↕	-
1	1	1	LDA	(A) ← (*data bus*)	-	↕	↕	-

Unlike the simple computer described in the class notes, no buses *per se* will be use (since there are no buses *external* to the CPLD) – instead, multiplexers will be used to route address, data, and control information to various parts of the machine. Also, instead of using latches to realize SRAM and the output port, edge-triggered D flip-flops will be used (since that is the "native" memory element available on the CPLD).

Also unlike the machine described in class, the Raulmatic 716 will feature a "memory edit" mode: when selected, the edit mode will allow the user to enter their program and data into SRAM using the DIP switch. A significant part of this experiment will therefore involve entering different programs into memory, executing them, and noting the results obtained.

The instruction set supported by the Raulmatic 716 is listed in Table II. Note that, unlike the instruction set described in the course notes, the input and output instructions (`INA` and `OUT`) do not have an address field. Also note that when data is loaded from or stored to memory, only the lower four bits are utilized (for the `STA` instruction, the upper three bits stored are always zero).

Table II. Raulmatic 716 Instruction Set.

OPCODE	Mnemonic	Function Performed
000	HLT	*halt execution (retain state)*
001	LDA *addr*	(A)←(*addr*)
010	ADD *addr*	(A)←(A)+(*addr*)
011	SUB *addr*	(A)←(A)-(*addr*)
100	AND *addr*	(A)←(A)∩(*addr*)
101	STA *addr*	(*addr*)←(A)
110	INA	(A)←[DIP3..DIP0]
111	OUT	[LED27..LED24]←(A)

Pre-lab Step (1):

Print out a copy of the skeleton file provided on the course website. Carefully study this file and fill in the lines of code that have been left "blank" (note – *it should not be necessary to add or modify any declarations*). **Be prepared to turn in the "hand-annotated" printed copy of your skeleton file at the beginning of your lab period.**

Pre-lab Step (2):

Hand-assemble the following program into Raulmatic 716 machine code. Write the machine code in both binary and hexadecimal format. Then, calculate the results stored at locations `1110` and **`1111`** for different DIP switch and "data" values entered into memory.

Address		*Instruction Mnemonic*	*Machine Code*	
Binary	*Hex*		*Binary*	*Hex*
0000	0	INA		
0001	1	OUT		
0010	2	ADD 1011		
0011	3	STA 1110		
0100	4	OUT		
0101	5	SUB 1100		
0110	6	OUT		
0111	7	AND 1101		
1000	8	OUT		
1001	9	STA 1111		
1010	A	HLT		
1011	B	*data*		
1100	C	*data*		
1101	D	*data*		
1110	E	*result*		
1111	F	*result*		

DIP[3:0] = _____

(1011) = _____

(1100) = _____

(1101) = _____

(1110) = _____

(1111) = _____

DIP[3:0] = _____

(1011) = _____

(1100) = _____

(1101) = _____

(1110) = _____

(1111) = _____

Step (1):

After your Lab Instructor checks the work you submitted for Pre-lab Step 1, augment the skeleton file accordingly. Verify that you can load a program into memory using the "edit" mode and can execute that program in "run" mode. For this initial verification, enter a program consisting of the instructions `INA`, `OUT`, `HLT`. Demonstrate entry and execution of this program to your Lab Instructor.

IMPORTANT: To make the design fit, open up the Optimization Constraint Editor and change the Nodes_collapsing_mode from Fmax to Area

Step (2):

Similar to the example worked in the class notes, load locations `0100` and `0101` with two different (4-bit) data values, and enter a series of instructions that exercise the `LDA`, {`ADD`/`SUB`/`AND`}, and `STA` instructions. Have each program store its result at location `0110`, and use the memory editor to monitor the contents of this location as each program executes. Note how the condition codes are affected by each program and record the results. Demonstrate entry and execution of these programs to your Lab Instructor.

Address		*Instruction Mnemonic*	*Machine Code*	
Binary	*Hex*		*Binary*	*Hex*
0000	0	LDA 0100		
0001	1	ADD 0101		
0010	2	STA 0110		
0011	3	HLT		
0100	4	*data*		
0101	5	*data*		
0110	6	*result*		

Result stored at location `0110` for:

(0100) = _____

(0101) = _____

(0110) = _____

CF = ___ NF = ___

ZF = VF = ___

Address		*Instruction Mnemonic*	*Machine Code*	
Binary	*Hex*		*Binary*	*Hex*
0000	0	LDA 0100		
0001	1	SUB 0101		
0010	2	STA 0110		
0011	3	HLT		
0100	4	*data*		
0101	5	*data*		
0110	6	*result*		

Result stored at location `0110` for:

(0100) = _____

(0101) = _____

(0110) = _____

CF = ___ NF = ___

ZF = VF = ___

Address		*Instruction Mnemonic*	*Machine Code*	
Binary	*Hex*		*Binary*	*Hex*
0000	0	LDA 0100		
0001	1	AND 0101		
0010	2	STA 0110		
0011	3	HLT		
0100	4	*data*		
0101	5	*data*		
0110	6	*result*		

Result stored at location `0110` for:

(0100) = _____

(0101) = _____

(0110) = _____

CF = ___ NF = ___

ZF = VF = ___

Step (3):

Once you are convinced that all eight instructions as well as the condition codes function correctly, enter the program you hand-assembled for Pre-lab Step 2. Verify that the results you predicted are produced. Demonstrate execution of this program to your Lab Instructor.

Step (4): Write your answers to the following Thought Questions in the space provided**.**

1. Examine the fitter report generated by ispLever and determine the total number of flip-flops utilized by your design as well as the total number of P-terms and macrocells.

   Number of flip-flops: _____

   Number of P-terms: _____

   Number of macrocells: _____

2. Use the ispLever timing analyzer to determine the *maximum clock frequency* (f_{max}) at which the Raulmatic 716 can run.

   _____

   _____

