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	Hand-assemble program and determine test data/results	4	
Step 1	Verify operation of memory editor provided	1	
Step 2	Complete PC module	2	
Step 3	Complete IR module	2	
Step 4	Import CLA4 module and complete ALU module	6	
Step 5	Complete IDMS module and demonstrate `INA`, `OUT`, `HLT`	3	
Step 6	Demonstrate `LDA`, {`ADD/SUB/AND`}, `STA` code execution	3	
Step 7	Demonstrate execution of program assembled for pre-lab	2	
Step 8	Thought questions	2	
	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: _____

Introducing the Personal Simple Computer: PSC 716

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 exercise

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 are also being 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.

In the interest of *Making Computers Simple Again* (rumored to be a provision of the New Green Deal), 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. **PSC 716** 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	LDA	(A) ← (*data bus*)	–	↕	↕	–
1	1	1	AND	(A) ← (A) ∩ (*data bus*)	–	↕	↕	–

Unlike the simple computer described in the class notes, no *tri-state* buses will be used (since there are no tri-state buses *internal* 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 CPLDs).

Also unlike the machine described in class, the **PSC 716** will feature a nifty "memory edit" mode: when selected, the edit mode will allow the user to enter their program and data into SRAM using the DIP switches. 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 **PSC 716** is listed in Table II. Note that, unlike the instruction set described in the course notes, the input and output instructions (`INA` and `OUA`) 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. **PSC 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) ←DIP[3:0]
111	OUA	DIS4 ← (A)

Pre-lab:

Hand-assemble the following program into **PSC 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	OUA		
0010	2	ADD 1011		
0011	3	STA 1110		
0100	4	OUA		
0101	5	SUB 1100		
0110	6	OUA		
0111	7	AND 1101		
1000	8	OUA		
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):
Carefully examine the `lab13_top_template.v` file provided for this experiment on the course website. Rename this file *myuserid*`_lab13.v` (where *myuserid* is your 8-character login name) and insert your identifying information where indicated. Next, open ispLever and create a new Verilog project named `Lab13`, select `LC4256ZE-5TN144C` as the device to use, and import your *myuserid*`_lab13.v` source file. Compile the program and load the JEDEC file produced by ispLever into the CPLD. Noting that memory edit mode is selected when DIP[7]=1, verify that the memory editor provided works as expected. The (4-bit) memory address is displayed as a single hexadecimal digit on DIS3, while the (7-bit) memory location contents is displayed on DIS2:DIS1 as two hexadecimal digits. Pressing the right pushbutton (S1BC) should increment the memory address by one, and pressing the left pushbutton (S2BC) should cause the 7-bit value entered on DIP[6:0] to be loaded into the memory location indicated. Comment out all verification code for this step once its functionality has been verified.

Step (2):
Complete the program counter (PC) module provided, then instantiate and test it following the instructions in the `lab13_top_template.v` file. Comment out all verification code for this step once its functionality has been verified.

Step (3):
Complete the instruction register (IR) module provided, then instantiate and test it following the instructions in the `lab13_top_template.v` file. Comment out all verification code for this step once its functionality has been verified.

Step (4):
Import the CLA4 module (4-bit adder/subtractor with condition codes) you wrote for Lab 12 (*no changes to it should be necessary*). Next, complete the arithmetic logic unit (ALU) module provided, which should include an instantiation of your CLA4 module. Instantiate and test your completed ALU by following the instructions in the `lab13_top_template.v` file. Comment out all verification code for this step once its functionality has been verified.

Step (5):
Complete the instruction decoder and micro-sequencer (IDMS) module provided, based on the system control equations you derived on Problem 1 of Homework 13. You are now ready to instantiate and test the **PSC 716** following the instructions in the `lab13_top_template.v` file. 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`, `OUA`, `HLT`. The **PSC 716** switch and LED mapping is shown in Figure 1.

Trouble-Shooting Tip: If you get an error while ispLever is attempting fit your design, open up the Optimization Constraint Editor and change the Nodes_collapsing_mode from Fmax to Area.

Checkpoint: Demonstrate Step 5 to your Lab Instructor.

Step (6):

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 one of these programs to your Lab Instructor.

Address		*Instruction*	*Machine Code*	
Binary	*Hex*	*Mnemonic*	*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*	*Machine Code*	
Binary	*Hex*	*Mnemonic*	*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*	*Machine Code*	
Binary	*Hex*	*Mnemonic*	*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 = ____

Checkpoint: Demonstrate Step 6 to your Lab Instructor.

Step (7):

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. Verify that the results you predicted are produced. Demonstrate execution of this program to your Lab Instructor.

Checkpoint: Demonstrate Step 7 to your Lab Instructor.

Step (8): 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 **PSC 716** can run.

   _____

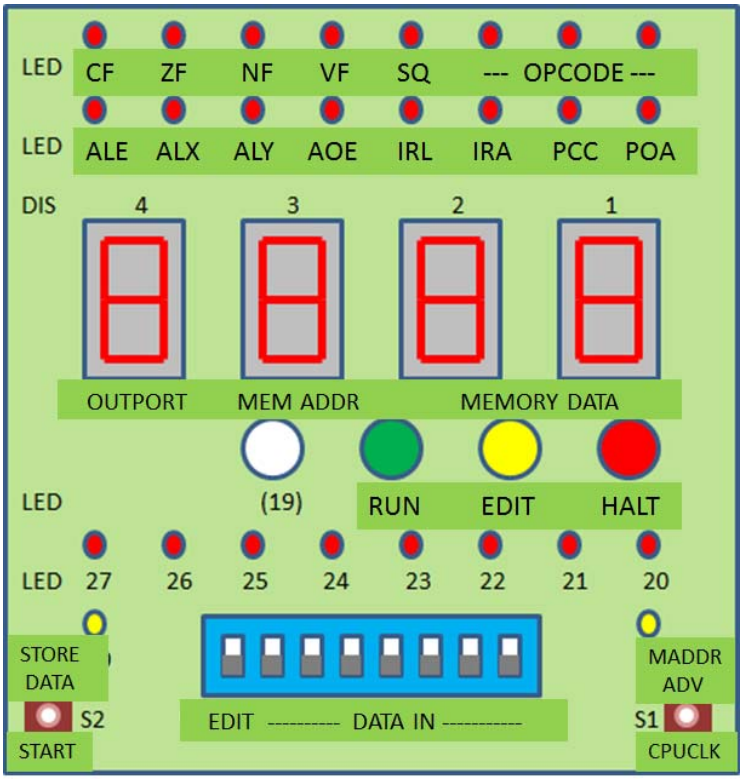

Figure 1. PSC 716 Switch and LED Utilization.

