Project Extra Credit Description
ECEN2350: Digital Logic
Spring 2016

Note: As of 3/13/2016 This is assignment is a Work-In-Progress and the final version will be posted within a few days.

Project Background

An extra project is available for any student wishing to get a "redo" on a previous project or exam score. The grade of this project will therefore replace a low score. However, you will have to demo the project, answer difficult demo questions from the professor and be submit a report on your design. Any submissions suspected as cheating or not their own work will be discarded immediately. This will not be a group assignment. The extra credit will apply to you and you only. Largely incomplete assignments will not be considered as gradeable extra credit effort.

This will not substitute for the final exam. Project 3 Reaction timer will double count as quiz grade 4 and project 3. This extra credit will either replace any low quiz score you have or any low project score you have. Given that, projects will encompass 20% and each quiz is 10% of your grade, this extra opportunity for credit will be able to recoop for a large % of your final grade between 15-20%

If you wish to do this project you have to commit to taking the grade that is assigned as the extra credit project to substitute out one of the other options.

Here is a relatively simple instruction set architecture that you can read about which gives assembly instructions examples like what is listed below:
http://www.atmel.com/Images/Atmel-0856-AVR-Instruction-Set-Manual.pdf

Reading this Wikipedia article will help with the background of an instruction and how it relates to the operations.
https://en.wikipedia.org/wiki/Instruction_set

A more general overview of the CPU and how it operates on data with various operations can be see here:
https://www.youtube.com/watch?v=cNN_tTXABUA

Project Requirements

You need to implement a low-level 8-bit CPU that uses an ALU that has the following specifications

- 16-bit Instruction Size
  - 4-Bit Opcode Identifiers
  - 2x 3-4 bit register identifiers
  - 8 bit immediate data values (0-255)
- 8x 8-bit registers to operate on
- 16 operations (listed in the table below)

The operations that you need to support are:

Operation	**Opcode**	**Description**	**Example**
Add	0 - 0000	Adds two values of a register together	Add r1, r4
Subtract	1 - 0001	Subtracts two registers from one another	Sub r3, r2
Invert (1's Complement)	2 - 0010	Does a bitwise inversion of a register's value	Inv r1
Logical AND	3 - 0011	Bitwise AND of two registers	And r3, r4
Logical OR	4 - 0100	Bitwise OR of two registers	Or r7, r7
Logical EXOR	5 - 0101	Bitwise EXOR of two registers	Exor r2, r0
Increment	6 - 0110	Increments by 1	Dec r3
Logical Shift Right	7 - 0111	Shifts all bits right one position of a register	Lsr r5
Logical Shift Left	8 - 1000	Shifts all bits left one position of a register	Rsr R2
Compare	9 - 1001	Compares to registers	cmp r0 r4
Load	10 - 1010	Loads an immediate value into a register	Ld r4, 127
Display	11 - 1011	Displays a register value on a seven segment display	Dsp r0, s0
Move	12 - 1100	Moves the value of register to another register	Move r2, r3
Branch if Equal	13 - 1101	Move forward/backward in your instruction memory to based on the previous compare operation	Breq 4
Branch if greater than	14 - 1110	Move forward/backward in your instruction memory to based on the previous compare operation if the compare resulted in a non-negative number.	Brgt -4
TBD	15 - 1111		

Registers will have the following encoding:

Register	Binary Value
R0	000
R1	001
R2	010

R3	011
R4	100
R5	101
R6	110
R7	111

It is suggested that you support a "manual instruction mode" where you can enter in commands from the DE0 board to process using the switches and the buttons. Example: Set the op code, set the registers, set the immediate value etc. Then press a button to execute.

The CPU will need to utilize a state machine (Finite State Machine) to process each of these commands in a way that the instruction is decoded so it can identify its operand registers, those operand registers need to be moved into the register holding positions of the ALU, then you need to perform the operation on those registers, then that output needs to be put into a result register, and finally that result will need to move back into the registers. You will need to store a sequence of these instructions in a piece of memory and iterate through them as you run through the each instructions state machine. An example diagram of this system can be seen below:

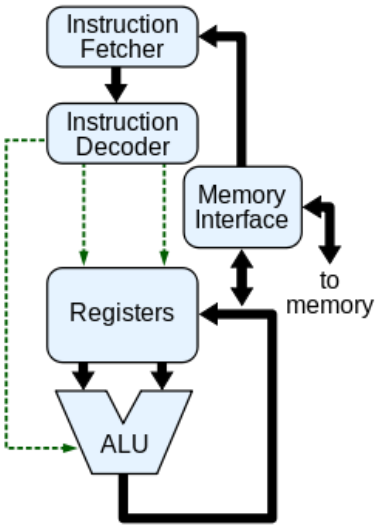

Figure: Simple CPU Design[1]

Procedure and Project Questions

1. Implement the design as a Finite State machine so that each instruction goes through a processing state machine.
2. Use a clock to help synchronize your processing
3. You need to be able to store a set number of instructions to perform. Your code should be able to increment through these instructions performing the operations.
4. Store the following sequence of assembly binary encoded data to multiply the following numbers

        1. 12 * 35
        2. -25 * 8
        3. -25 * -8
    5. Report the time in gate delays each instruction takes to execute
    6. Report the time it takes to multiply these numbers together using your assembly code.
    7. Overflow, carry, negative and zero flags will need to be tracked. Each full operation should have some effect on these flags. The compare operation should set these directly.
    8. You are welcome to increase the instruction operations supported by a bit and support another 16 operations if you find some more useful commands you wish to support.
    9. Conclusion statements.
        1. What was the most difficult part of this project?
        2. If something did not work, what is the reason and how would you correct it?
        3. What would you do different next time?

Due date will be no later than the last day of final exam week with which you will need to demo the project before that date.

Lab Report Requirements

Figures, tables, and diagrams can only give you a better score. Please provide them when applicable. Any videos of the project working are great and can be provided in the report. Format your report as the following.

1. **Cover Page**: Team Members names, class, project
2. **Introduction**: Short introduction on what the project is
3. **Lab procedure/Results**: Answer the questions in the procedure section. Explain how your circuit function
4. **Conclusion statements from above**:
    a. What was the most difficult part of this project? If something did not work, what is the reason and how would you correct it? What would you do different next time?

Demonstrations to the Professor

Every submission must be demonstrated to the professor to receive credit. Demonstrations must be given by the end of final exam week.

References

[1] Image provided by Wikimedia. 12 March 2016.
2005-10-06 02:40 R. S. Shaw 300×418× (6952 bytes) *Title: CPU_block_diagram.png Author: R. S. Shaw Description: Block diagram of a hypothetical simple CPU, showing instruction fetch, decode, data registers, ALU, and memory interface, and major relationships.*