Scott Nidell

Final Project Notes

1000921465

CSE 2441-001

**Introduction:** The last entry of the lab manual details the final project design and assembly. The final project is broken into three parts. Part ‘A’ is to integrate lab 9 into the rest of the TRISC system assembly. Part ‘B’ is to design a FSM for the additional functions Load, Store, Add, and Jump. Part ‘C’ is to integrate the part ‘B’ controller into the FSM system.

**Theory:**  Using the notes from class the load state sequence was first designed. Load must execute the normal fetch and decode but also redirect the memory address register to the next address and load the value of that address to memory. This function happened to be the longest of all added functions.

The system call for load looks something like this: LDA h5. Meaning to load whatever is in memory address h5 to the accumulator. I broke down the action into single step sequences

1. *Have the MAR select MDO3-0 control line C3=1*
2. *Read from memory C4=1*
3. *Transfer data from memory C42=1*
4. *Accumulator selects MDO3-0 C11=1*

Each step involved creates new states: H, I ,J ,K ,L.

Next was to add the store function. A store function looks like this: STA h10. This statement says to store whatever is in the accumulator to address h10. Again the actions were broken down into single sequences

1. *Have the MAR select MDO3-0 control line C3=1*
2. Transfer data write with wren=1. C4=1 and C5=1

The next function to be added was the ADD function. This function will add whatever is in memory to the current value of the accumulator. Again the actions were broken down into single sequences

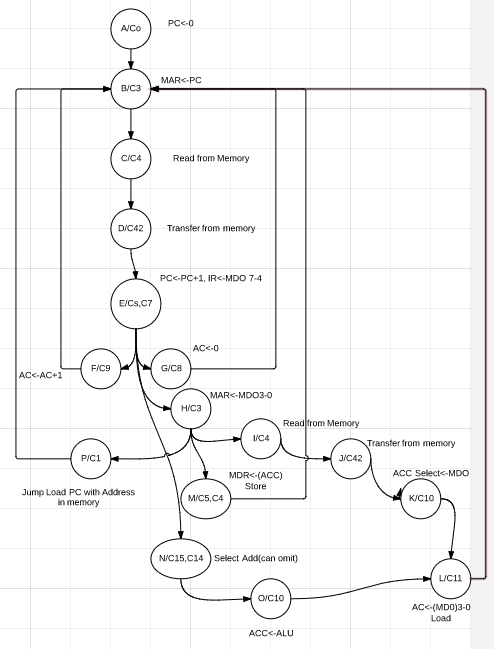
1. *Have the MAR select MDO3-0 control line C3=1*
2. *Have the accumulator select the ALU input. C10=1*
3. *Load what was selected in the Accumulator to the output C11=1*

The last instruction that needed to be implemented is the jump function. This function will jump to the address in an operand that is defined in the current memory address. Again the actions were broken down into single sequences

1. *Have the MAR select MDO3-0 control line C3=1*
2. Load program counter with MD03-0 with the address destination

All of these extra states were implemented into a state diagram (Figure 1).

**Figure 1:** *State Diagram for Project C*

**

The next step is to test the system. To understand what needed to happen I created a separate project to read what data was in the memory given to us. It revealed the following:

|  |  |  |
| --- | --- | --- |
| **Memory Address** | **Value** | **Description** |
| 0 | 70 | Clear ACC |
| 1 | 60 | Increment ACC |
| 2 | 60 | Increment ACC |
| 3 | 88 | Jump to address 8 |
| 4 | 0F | Load what is in ACC to address F |
| 5 | 2F | Add F to accumulator |
| 6 | 1E | Store 1110 to memory |
| 7 | 8B | Jump to address 1011 |
| 8 | 2E | Add 1110 to Accumulator |
| 9 | 1F | Store what is in acc to address F |
| 10 | 84 | Jump to address 4 |
| 11 | 70 | Clear ACC |
| 12 | 60 | Increment accumulator |
| 13 | 80 | Jump to address zero |
| 14 | 03 | Load what’s in ACC to memory at address 3 |
| 15 | 00 | none |

After these functions are designed these are tested first by waveform. Then the design was loaded onto the DE1 and verified by the lab instructor.

**Conclusion:** The project went smooth for the most part. The Add function was creating an error that had to be changed and tested directly on the DE1. Along with finishing the project a project lab report was a supplement to project completion.