ECE 406

Project 2: MIPS CPU Redesign

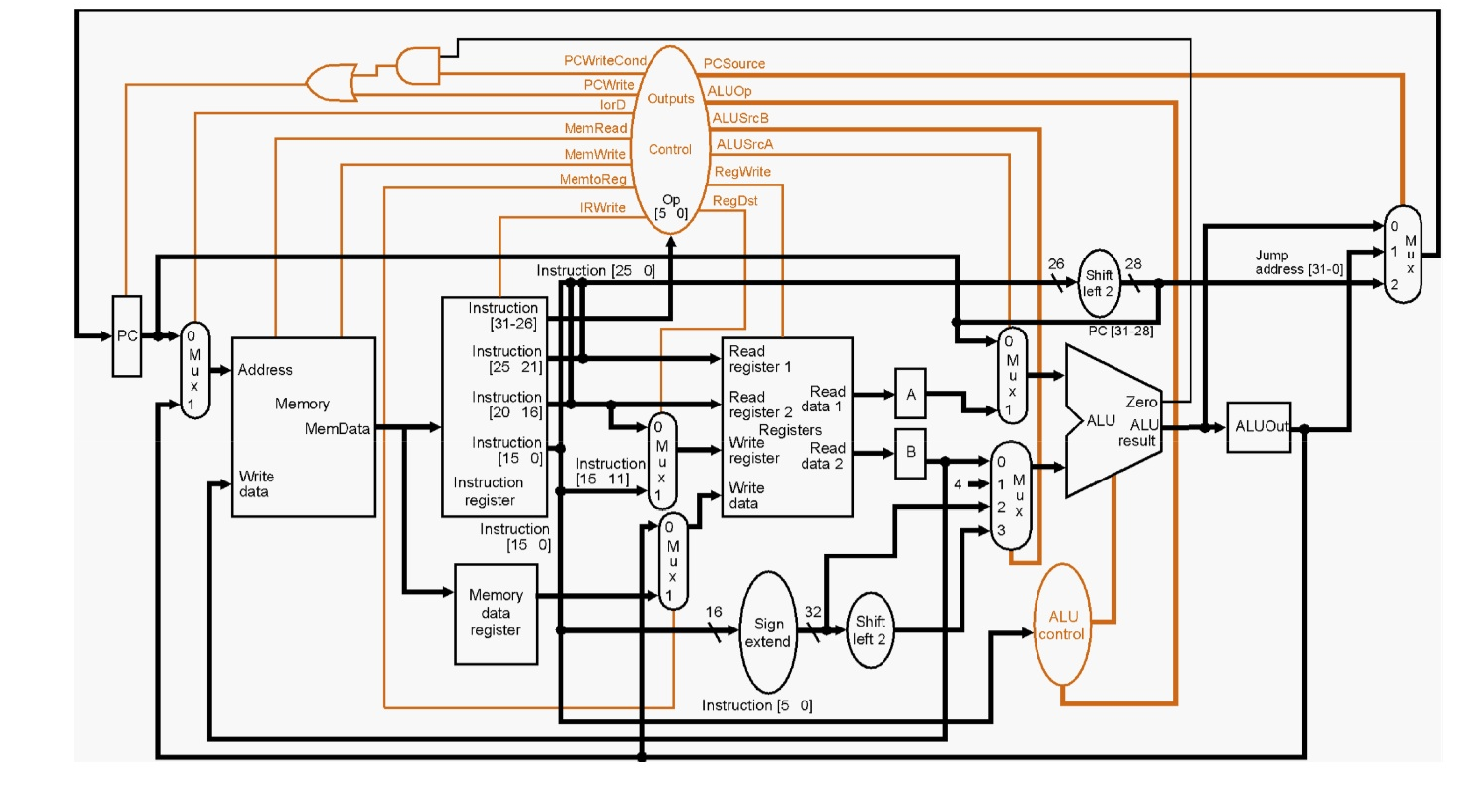
John Sizemore

December 6, 2011

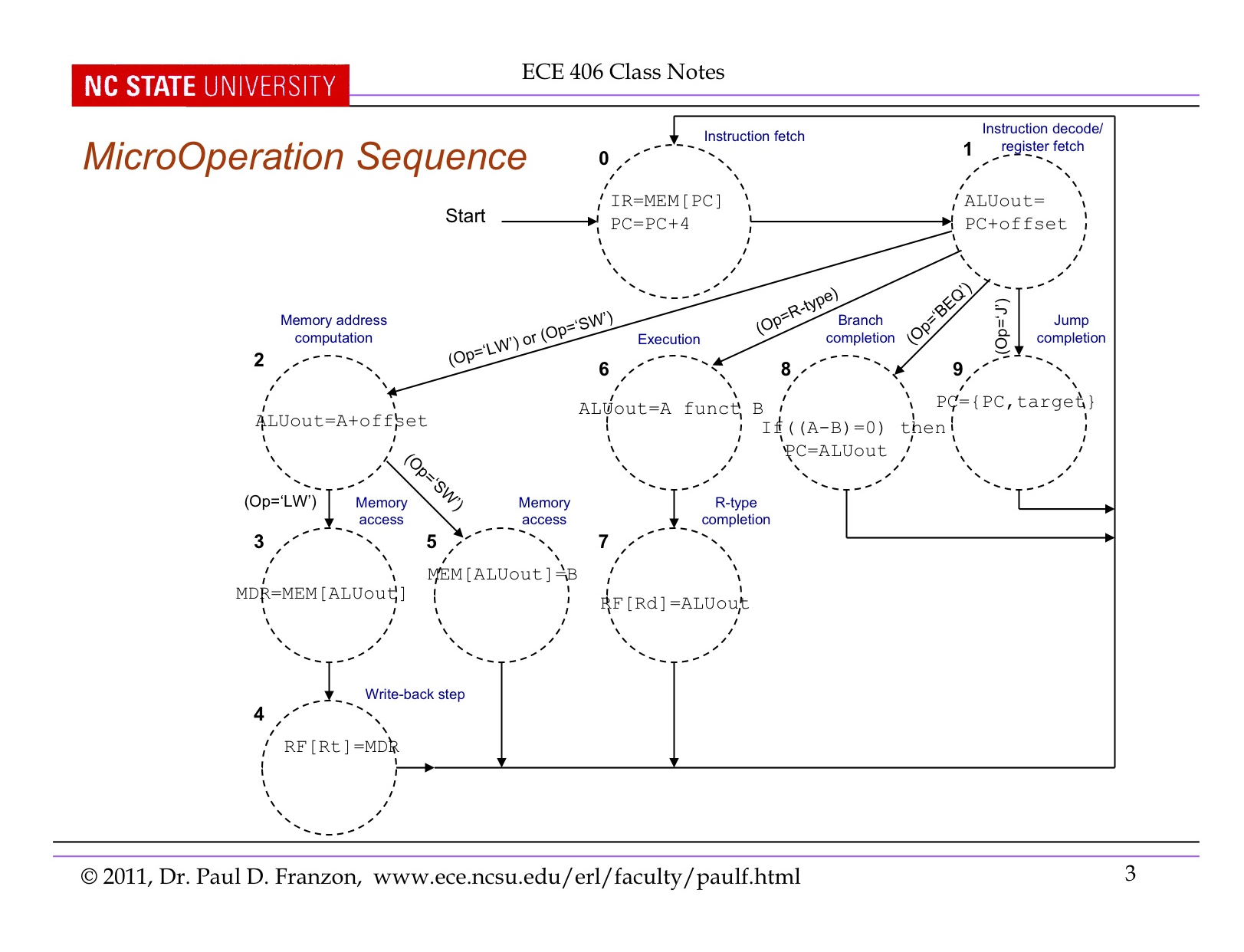
Redesign Background/Implementation

The instruction to be implemented for this project was to enable the MIPS CPU to load a byte into a specified register. This byte would be extracted out of a word stored in the memory data register (MDR) and sign-extended to 32 bits to match the bit length of each register in the register file. The byte to be loaded would fill the lower 8 bits of the register, while the remaining 24 bits would be the sign extended values.

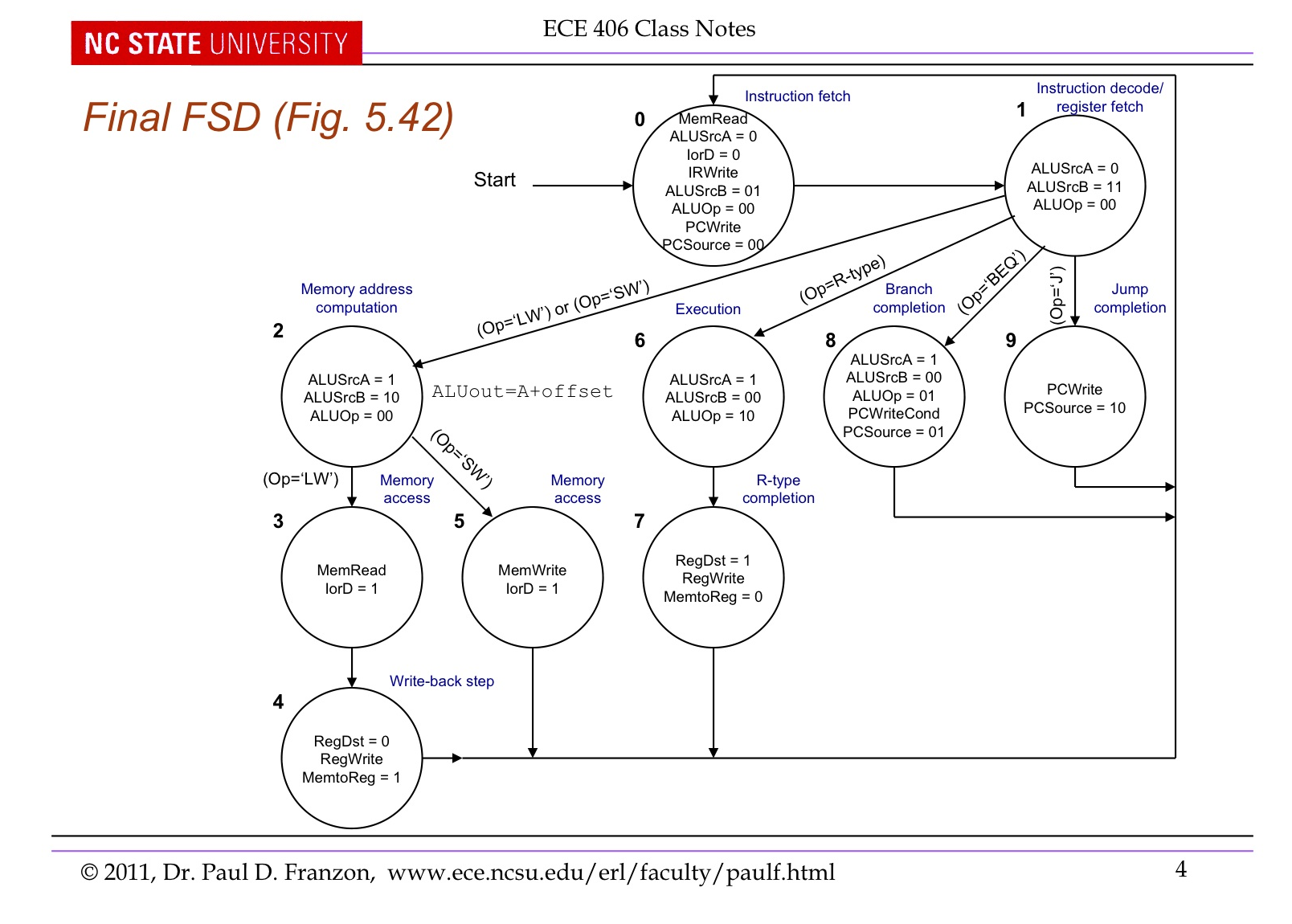
The following three figures are the hardware configuration, micro-operation sequence, and the finite state diagram of the MIPS CPU before implementation of the Load Byte instruction:



**Figure 1: Original MIPS CPU Hardware Configuration**

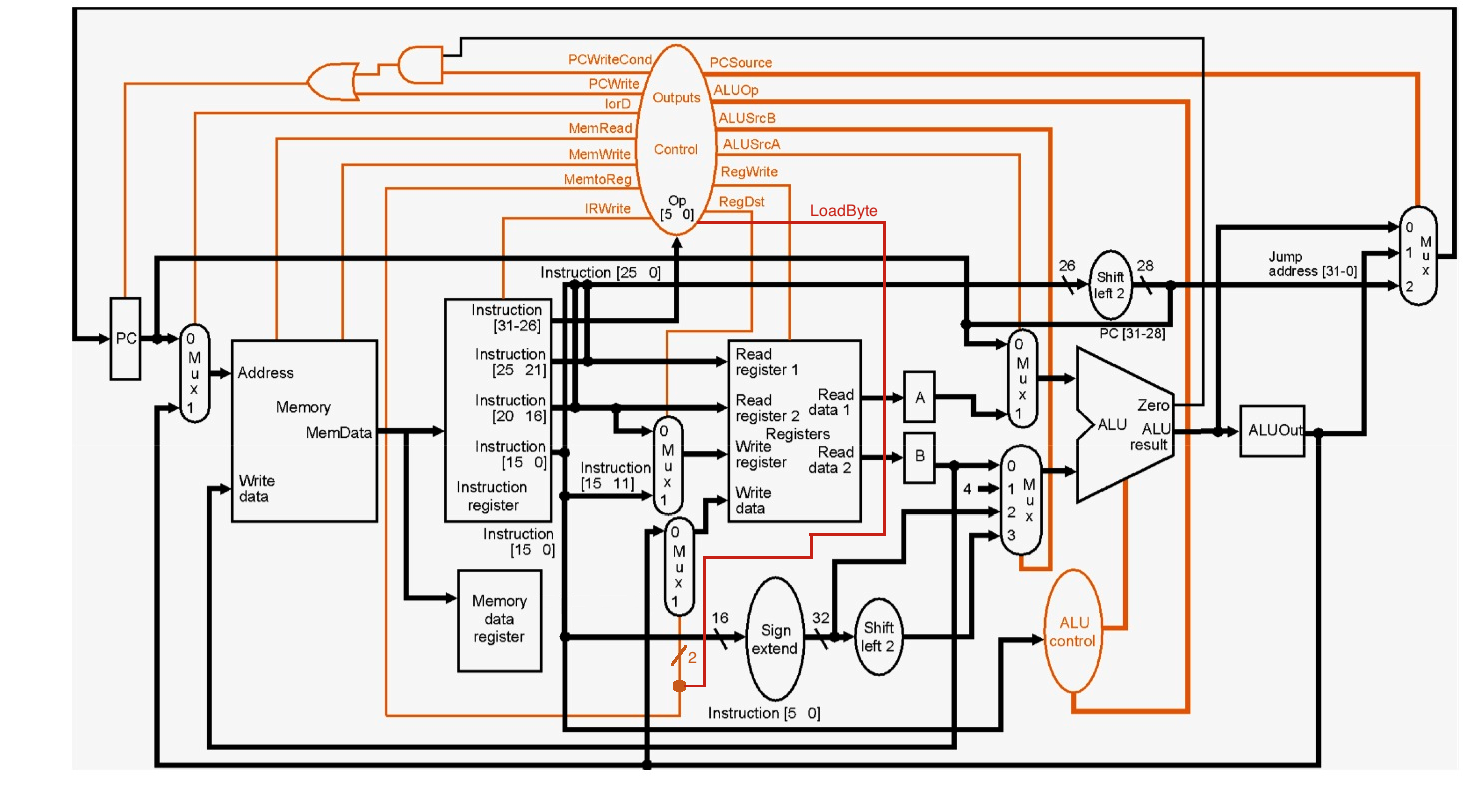
****

**Figure 2: Original MIPS CPU Micro-operation Sequence**

****

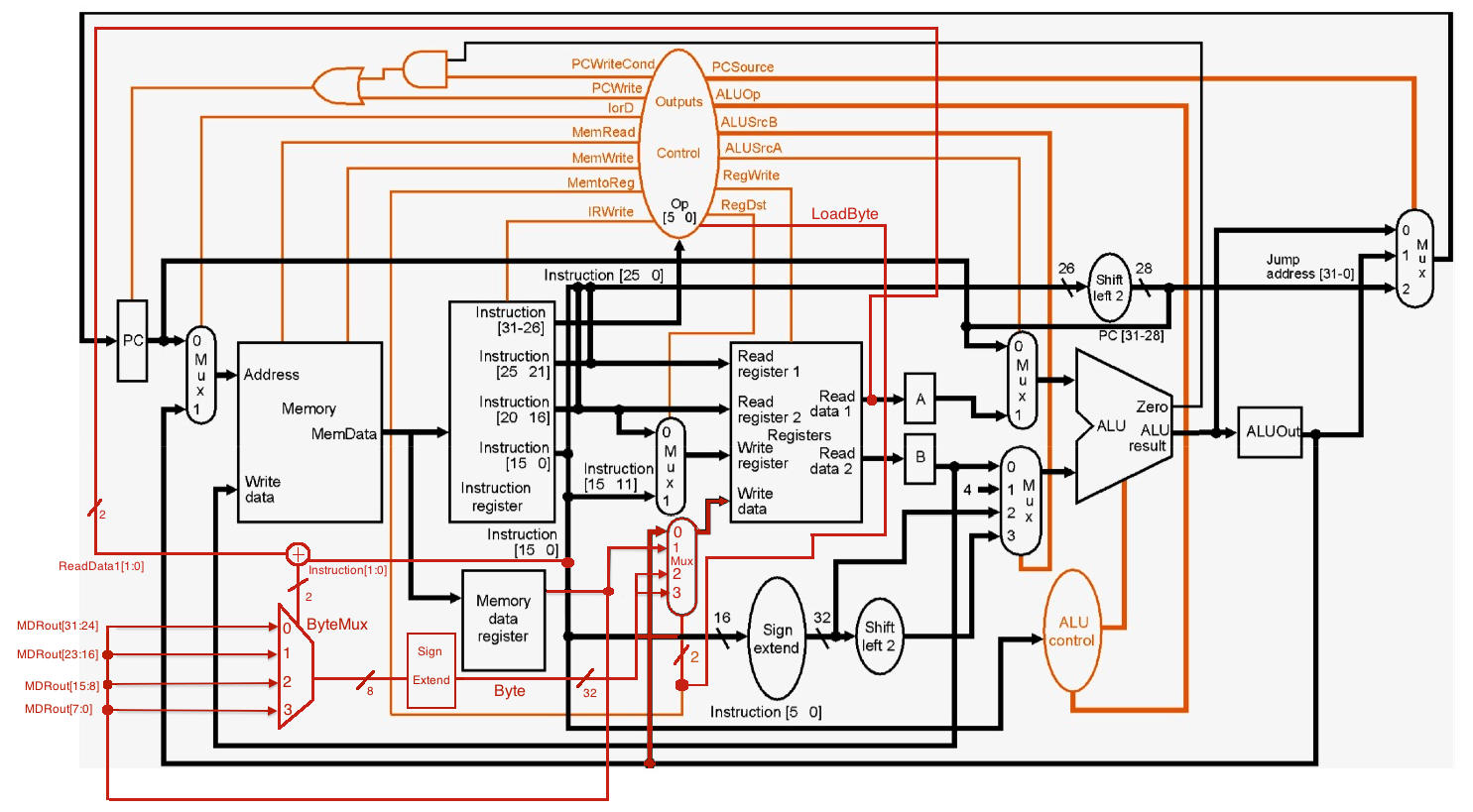
**Figure 3: Original MIPS CPU Finite State Diagram**

The implementation of this instruction was heavily based on the Load Word instruction that was already present within the architecture. Both the Load Word and the Load Byte instructions would go through the same states in the controller in regards to calculating the new program counter (PC), calculating the address of the desired word in memory, and accessing the memory at the computed address to place the desired word into the MDR. The difference between the two commands is after the memory is accessed with the computed address and its value is stored in the MDR. An extra control line was added to the controller that would be set high in the event that the hardware was executing a Load Byte command. This control line also coincided with the creation of a new state in the finite state diagram (FSD) of the MIPS CPU. The addition of this control line to the hardware configuration is as follows (all changes are in red):

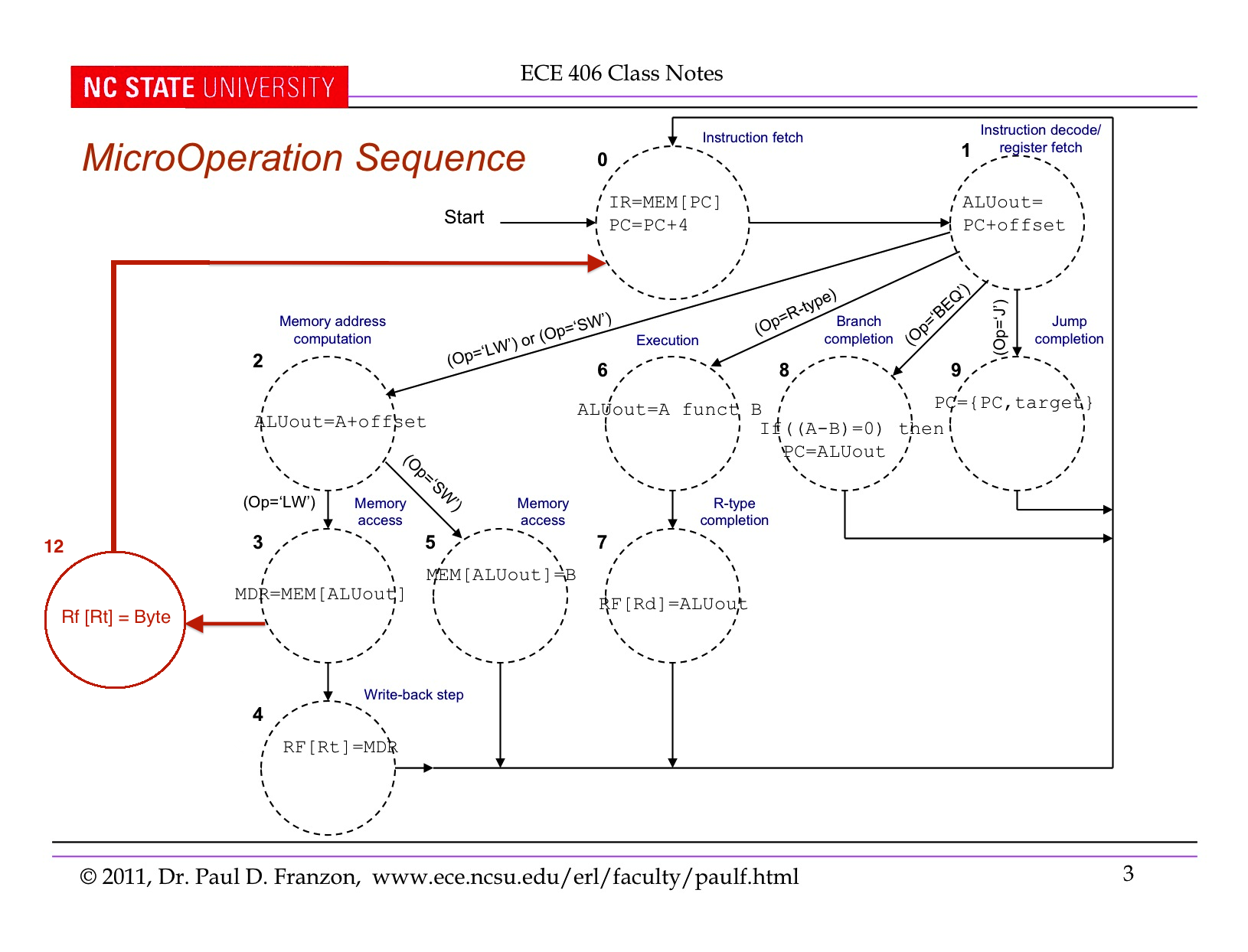


**Figure 4: Load Byte Control Line Addition**

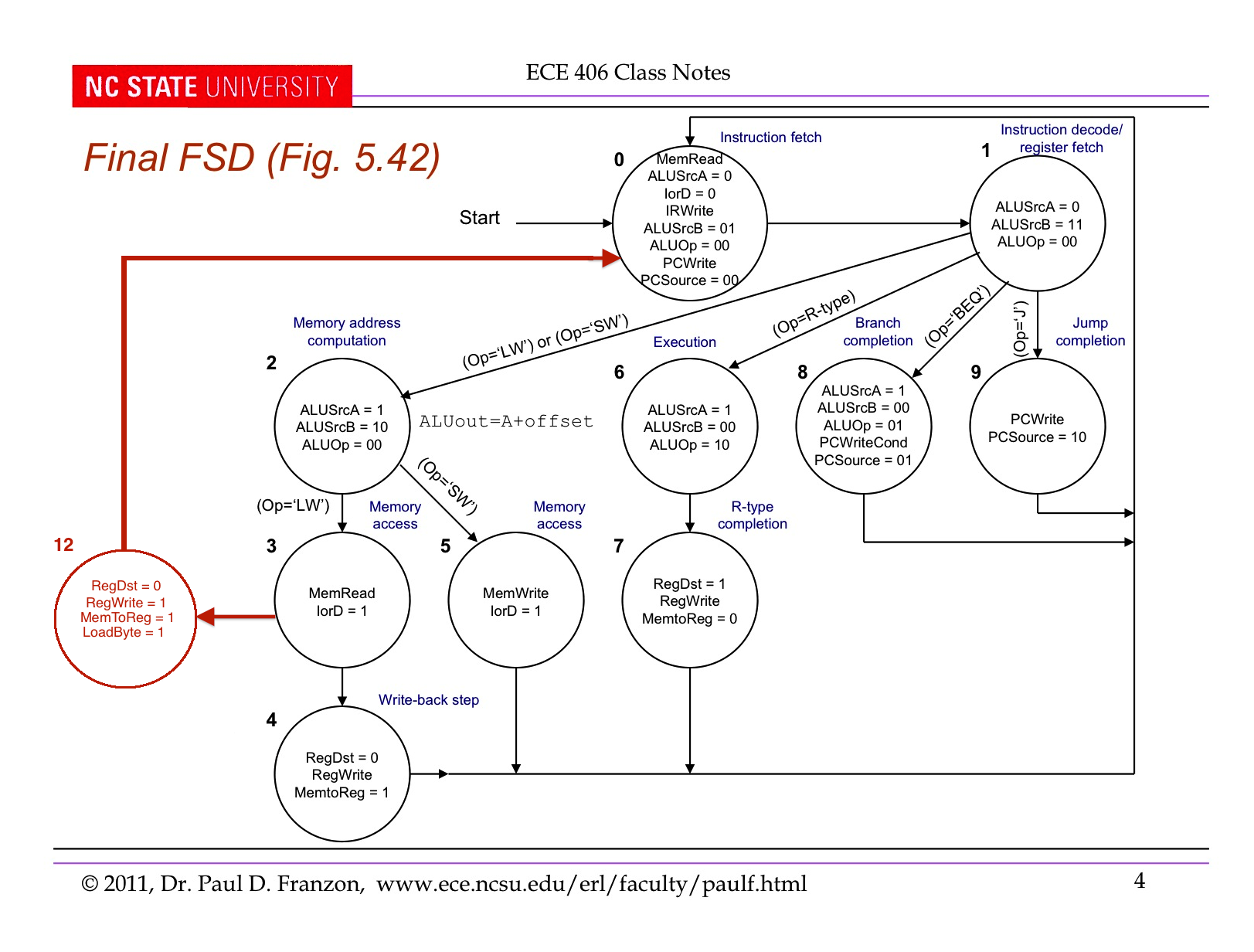
This control line was set as an additional select line to the multiplexer governing the write data port of the register file. The additional select line mandated four inputs, where the first two remained unchanged from the original hardware configuration and the last two were connected to the output of a new multiplexer. This new multiplexer, which was not part of the original hardware configuration, was used to choose the desired byte out of the word stored in the MDR. This multiplexer had two select lines, which were the sum of the bottom two bits of the value in the instruction register (IR) and the bottom two bits of the source register operand used to compute the memory address in the Load Byte MIPS instruction. The bottom two bits of the source register contained the bottom two bits of the base address in memory, while the bottom two bits of the IR value were the bottom two bits of the memory offset in the MIPS Load Byte instruction. Since there are only four possibilities for bytes in a given word, the resulting sum of the above computation was all the information needed to determine the correct byte to load. The resulting diagrams of the hardware configuration, micro-operation sequence, and the finite state diagram reflect the complete addition of the Load Byte instruction into the computer architecture (all changes in red):



**Figure 5: Complete Load Byte Instruction Hardware Changes**

****

**Figure 6: Complete Load Byte Instruction Micro-Operation Sequence Changes**



**Figure 7: Complete Load Byte Instruction Finite State Diagram Changes**

The sequence of executing the Load Byte instruction is as follows: First, the instruction is fetched from memory at the address specified by the PC and stored in the IR. Next, the PC is incremented to point to the next instruction. The address of the desired byte in memory is then computed in the arithmetic and logical unit (ALU) using the source register specified in the MIPS Load Byte instruction as one operand, and the offset in the MIPS Load Byte instruction as the second operand. Next, the memory is accessed at that computed address and its specified word is stored in the MDR. The LoadByte control flag is set high, and the bottom two bits of the value in the IR and the bottom two bits of the source register specified in the MIPS Load Byte instruction are added to select the appropriate byte out of the word in the MDR. The chosen byte is then sign-extended to 32 bits. Since the LoadByte flag is high, the multiplexer connected to the write port of the register file will select the sign extended byte which will then be written to the specified destination register in the MIPS Load Byte instruction.