Learning Activity 5

By Alex Clunan

ECE 4435 Computer Architecture & Design

March 16, 2025

1 Problem Statement

Design and synthesize a controller for a multi-cycle RISC-V CPU. Modules including a branch unit, ALU decoder, immediate source decoder, load-store decoder, and multicycle control finite state machine are used to create the controller. A RISC-V assembly program was also created to test the branching portion of the CPU.

2 Analytical Design

The branch unit supports branching for this CPU. It is implemented using one assign statement that sets the output value based on the type of branch (beq, bne, blt, bge, bltu, bgeu), the flags, and the Branch input.

The ALU Decoder generates an ALUControl output that tells the ALU what operation to complete. It works by checking what ALU operation is being used and then checking all other inputs to generate the ALUControl output. When the ALU Operation is ‘10’, the 5th bit of the opcode, function 3, and bit 5 of function 7 in the instruction are concatenated together into one wire which is used as the input to a case statement. This case statement sets the ALUControl output.

The immediate source decoder works by using the opcode as an input to a case statement. The output is based on the value of the opcode.

The load-store decoder determines the type of load (word, extended half-word, extended byte) and type of store (word, half-word, byte). This works by using a case statement to set the values of LoadType and StoreType based on the function 3 value.

The multicycle control finite state machine (FSM) is used to control the state of the CPU. It is separated into 3 sections: current state, next state logic, and output state logic. The current state portion checks for a reset signal and the rising edge of the clock. If the reset is triggered, the FSM goes to the FETCH stage, otherwise, during the positive edge of the clock, the state is set to the next state.

The next state logic determines which state is next based on the current state and some other factors. This is implemented in a case statement which checks what state the FSM is currently in. If the FSM is in the decode stage, the next state is chosen based on the opcode in a case statement. If in the MEMADR stage, the next state is based on if the instruction is a load or store instruction. A state machine diagram is shown below in Fig 1.

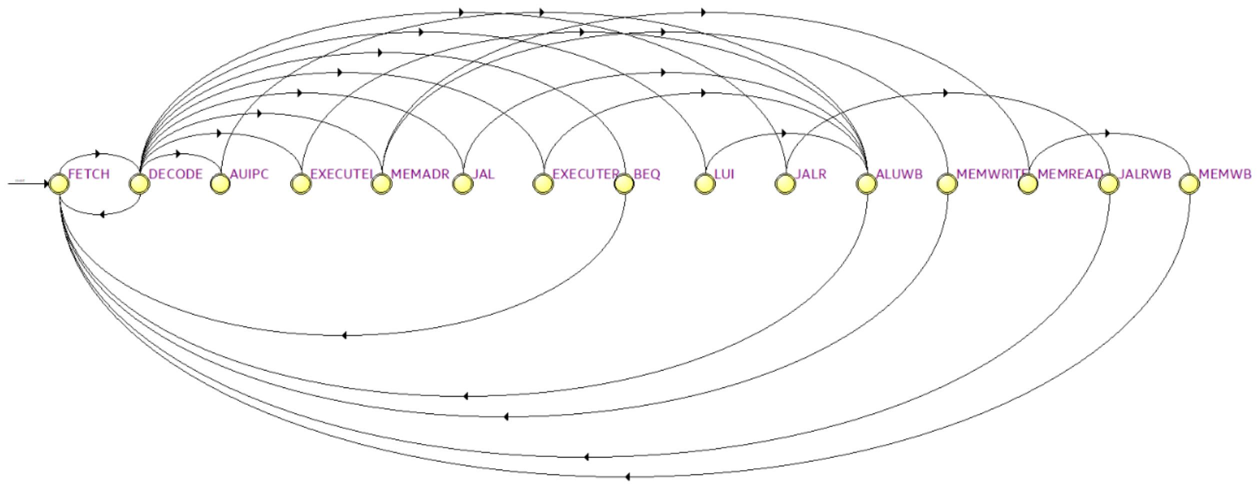


Fig 1. Finite State Machine for Controller

Finally, the output logic section of the FSM determines the output values of the FSM. These values include: ALUSrcA, ALUSrcB, ResultSrc, AdrSrc, IRWrite, PCUpdate, RegWrite, MemWrite, ALUOp, and Branch. Using a case statement based on the current state, these values are set on a per state basis.

The final schematic of the controller is shown below in Fig 2.

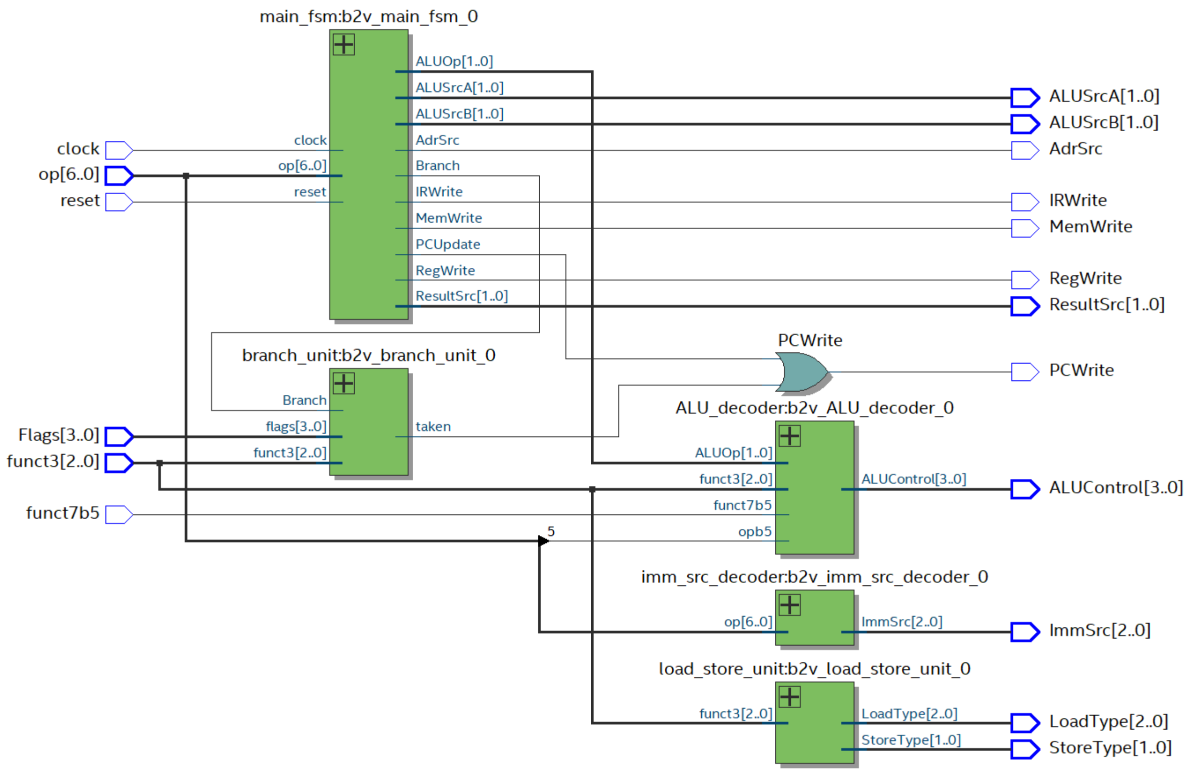
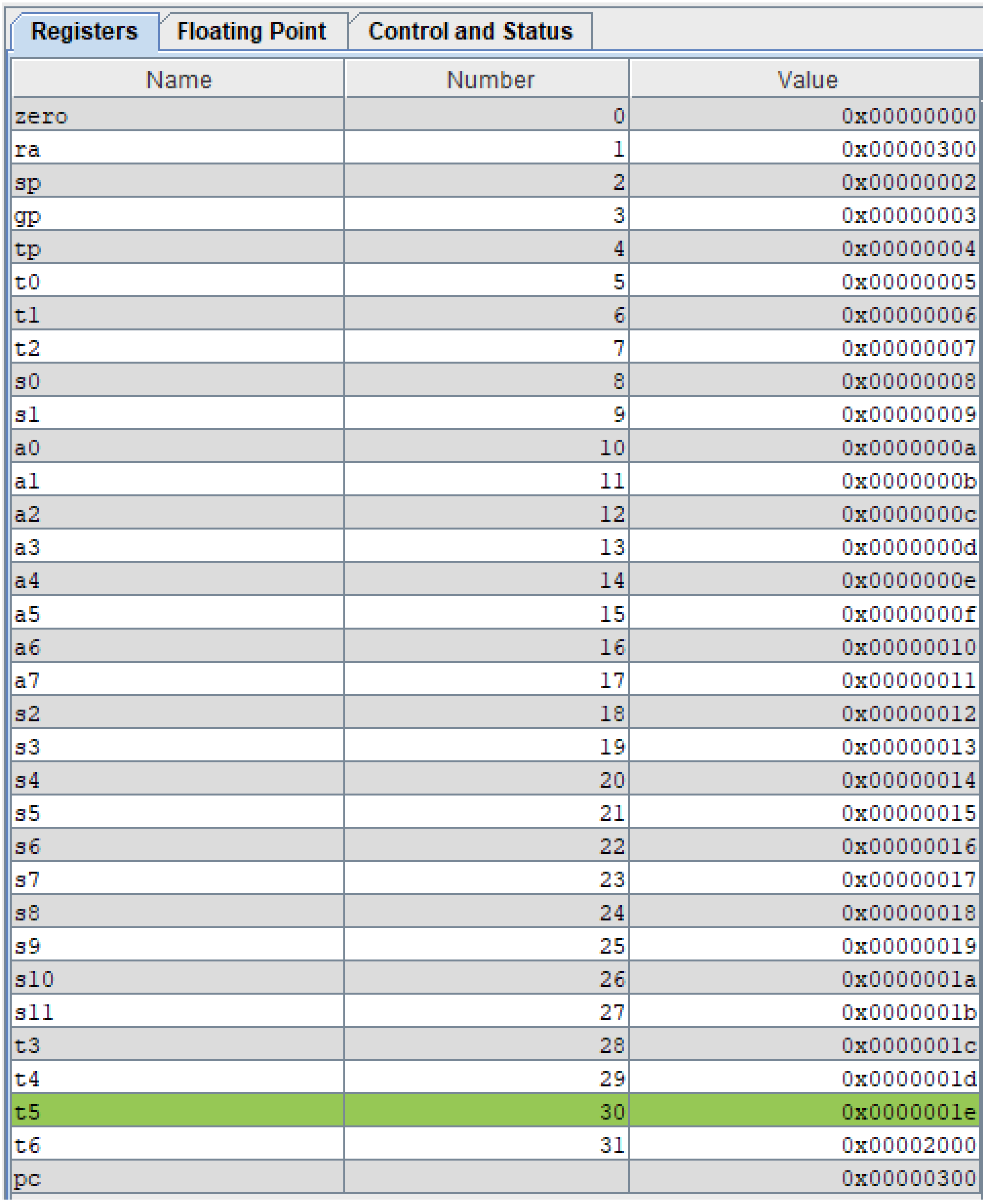


Fig 2. RTL Schematic of Controller for RISC-V CPU

The test branch portion of the RISC-V assembly program was also written to complete the requirements. It works by jumping around to different points while loading the register number into the register.

3 Numerical Verification

The final register states of the assembly program is shown in Fig 3.



Name	Number	Value
zero	0	0x00000000
ra	1	0x00000300
sp	2	0x00000002
gp	3	0x00000003
tp	4	0x00000004
t0	5	0x00000005
t1	6	0x00000006
t2	7	0x00000007
s0	8	0x00000008
s1	9	0x00000009
a0	10	0x0000000a
a1	11	0x0000000b
a2	12	0x0000000c
a3	13	0x0000000d
a4	14	0x0000000e
a5	15	0x0000000f
a6	16	0x00000010
a7	17	0x00000011
s2	18	0x00000012
s3	19	0x00000013
s4	20	0x00000014
s5	21	0x00000015
s6	22	0x00000016
s7	23	0x00000017
s8	24	0x00000018
s9	25	0x00000019
s10	26	0x0000001a
s11	27	0x0000001b
t3	28	0x0000001c
t4	29	0x0000001d
t5	30	0x0000001e
t6	31	0x00002000
pc		0x00000300

Fig 3. Register Outputs of Assembly Program

The program chose the correct branches when moving through it one instruction at a time. It also loaded all the correct values into registers as shown in Fig 3.

4 Summary

The controller portion of the RISC-V CPU was created using a branch unit, ALU decoder, immediate source decoder, load-store decoder, and multicycle control finite state machine. Now that both the datapath and controller portions of the CPU are created, the CPU can now be tested in its entirety using the verified assembly code.