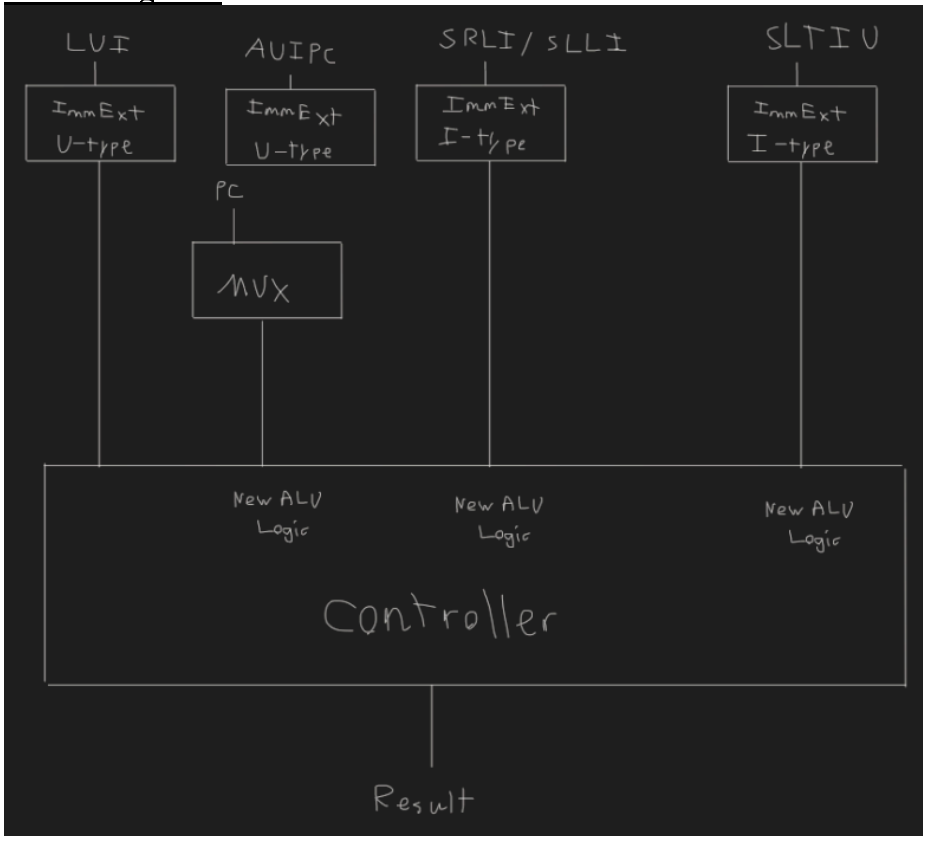
Class:	**CPE300L - Digital System Architecture and Design Lab**	Semester:	**Fall 2025**
Points		Document author:	**Darryll Mckoy and Joshua Sturtevant**
		Author's email:	**Mckoyd1@unlv.nevada.edu**
		Document topic:	**Final Project Report**
Instructor's comments:			

Project Overview:

For this project we expanded the RISC-V multicycle processor to support 5 additional instructions.

- Load upper immediate (LUI)
- Shift left logic immediate (SLLI)
- Shift right logic immediate (SRLI)
- Set less than immediate unsigned (SLTIU)
- Add upper immediate to PC (AUIPC)

Block Diagram:



Necessary Adjustments:

- **LUI**
  - Add 11th state to FSM (S11_ExecuteU). (calls srcA/srcB mux)
  - Add U-type op code to FSM decode state (S1_Decode). (tells FSM which state to call).
  - Update Extend to support LUI (sets immediate bits and rest to zero).
  - Change ALUsrcA mux from 3 to 1 to 4 to 1 mux and add 32'b 0 option.
  - Add LUI to instruction decoder.

- **SLLI/SRLI**
  - Update ALU decoder to include SLLI/SRLI (Tells ALU what instruction to call).

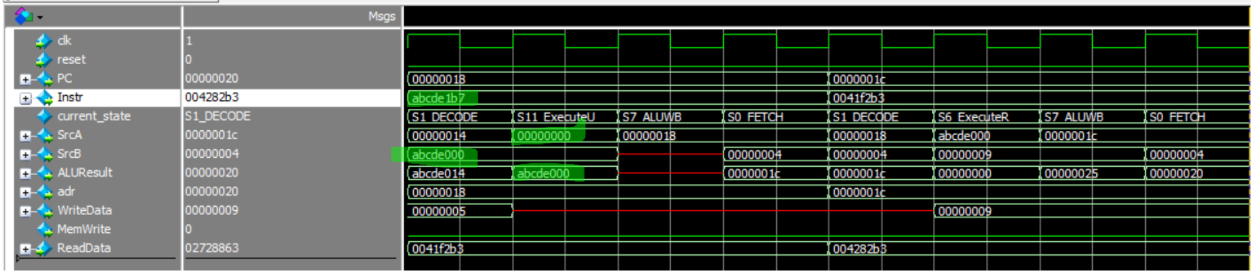
- **SLTIU**
  - Increase ALUdecoder ALUcontrol from 3-bits to 4-bits. (with added instruction it now needs to support more than 8 functions).
  - Add SLTIU to ALUdecoder (Tells ALU what instruction to call).
  - Add SLTIU to ALU (ALU performs operation and returns value).

- **AUIPC**
  - Add AUIPC to instruction decoder (tells immediate extender the datatype)
  - Add 12th state to FSM for AUIPC (S12_Execute_AUIPC).

Simulation Waveforms:

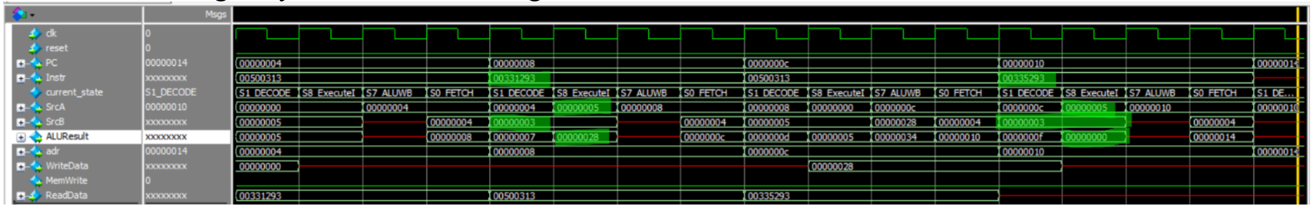
LUI waveform

- When executeU is called upper immediate of machine code is loaded from alusrcB and remainder of zeros are loaded from alusrcA then result is 0xabcde000.



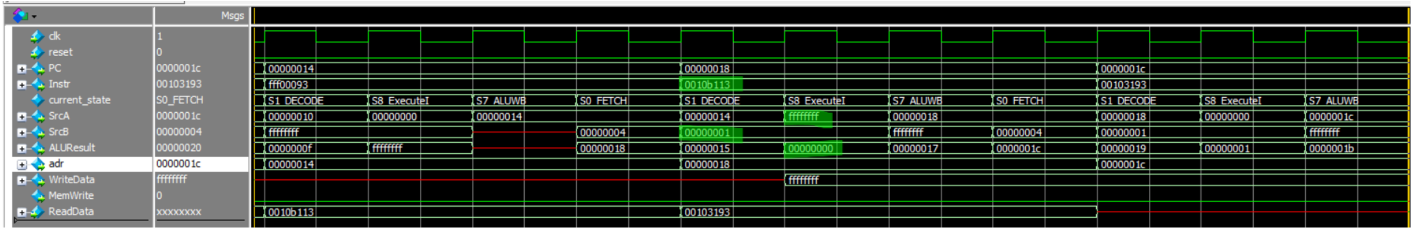
SLLI/SRLI

- (For SLLI, machine code 00331293), When executeI is called alusrcA containing 0x5 is shifted left by alusrcB containing 0x3. The ALU result is 0x28.
- (For SRLI, machine code 00335293), When executeI is called alusrcA containing 0x5 is shifted right by alusrcB containing 0x3. The ALU result is 0x00.



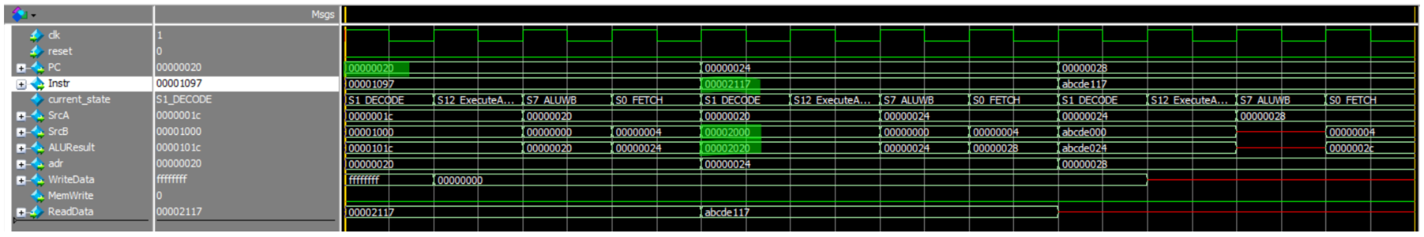
SLTIU (Machine Code 0010b113)

- Compares alusrcA to alusrcB and checks if A is less than B treated as unsigned values. Aluresult is zero which tells us that alusrcA is not less than alusrcB.



AUIPC (Machine Code 00002117)

- Adds upper immediate value stored in alusrcB to PC value 0x20 giving the alu result 0x2020.



<u>Conclusion:</u>

Upon completion of this project, we were able to successfully implement five additional instructions to the multicycle processor and demonstrate simulation of each additional instruction as shown in the images above. This project has given us a much deeper understanding of the inner workings of the multicycle processor with clearer knowledge of how to add more functionalities as needed. For future steps we would like to add additional instruction and display the current state of the FSM and output result onto the DE2115 LCD.

<u>Link to system verilog code and test program files:</u>

Appendix:

Table of Instructions

Step	PC	Instr	State	Result	Added Instruction
1	00	00000000	S0: Fetch	00000004	
2	04	abcde1b7	S1: Decode	abcde000	LUI
2	04	“ “	S11: ExecuteU	abcde000	
2	04	“ “	S7: ALUWB	X	
2	04	“ “	S0: Fetch	00000008	
3	08	00500313	S1: Decode	00000009	addi
3	08	“ “	S8: ExecuteI	00000005	
3	08	“ “	S7: ALUWB	X	
3	08	“ “	S0: Fetch	0000000c	
4	0c	00331293	S1: Decode	0000000b	slli
4	0c	“ “	S8: ExecuteI	00000028	
4	0c	“ “	S7: ALUWB	Abcde00c	
4	0c	“ “	S0: Fetch	00000010	
5	10	00500313	S1: Decode	00000011	addi
5	10	“ “	S8: ExecuteI	00000005	
5	10	“ “	S7: ALUWB	00000038	
5	10	“ “	S0: Fetch	00000014	
6	14	00335293	S1: Decode	00000013	srli
6	14	“ “	S8: ExecuteI	00000000	

6	14	“ “	S7: ALUWB	Abcde014	
6	14	“ “	S0: Fetch	00000018	
7	18	Fff00093	S1: Decode	00000013	addi
7	18	“ “	S8: ExecuteI	ffffffff	
7	18	“ “	S7: ALUWB	x	
7	18	“ “	S0: Fetch	0000001c	
8	1c	0010b113	S1: Decode	00000019	sltiu
8	1c	“ “	S8: ExecuteI	00000000	
8	1c	“ “	S7: ALUWB	0000001b	
8	1c	“ “	S0: Fetch	00000020	
9	20	00103193	S1: Decode	0000001d	sltiu
9	20	“ “	S8: ExecuteI	00000001	
9	20	“ “	S7: ALUWB	0000001f	
9	20	“ “	S0: Fetch	00000024	
10	24	00001097	S1: Decode	00001020	auipc
10	24	“ “	S12: ExecuteAUIPC	00001020	
10	24	“ “	S7: ALUWB	00000024	
10	24	“ “	S0: Fetch	00000028	
11	28	00002117	S1: Decode	00002024	auipc
11	28	“ “	S12: ExecuteAUIPC	00002024	
11	28	“ “	S7: ALUWB	00000028	
11	28	“ “	S0: Fetch	0000002c	
12	28	Abcde117	S1: Decode	Abcde028	auipc
12	28	“ “	S12: ExecuteAUIPC	Abcde028	
12	28	“ “	S7: ALUWB	x	
12	28	“ “	S0: Fetch	00000030	