CS230 Project

Vedang Asgaonkar (200050154)
Prathamesh Pilkhane (200050109)
Shashwat Garg (200050130)
Arpon Basu (200050013)

April 2022

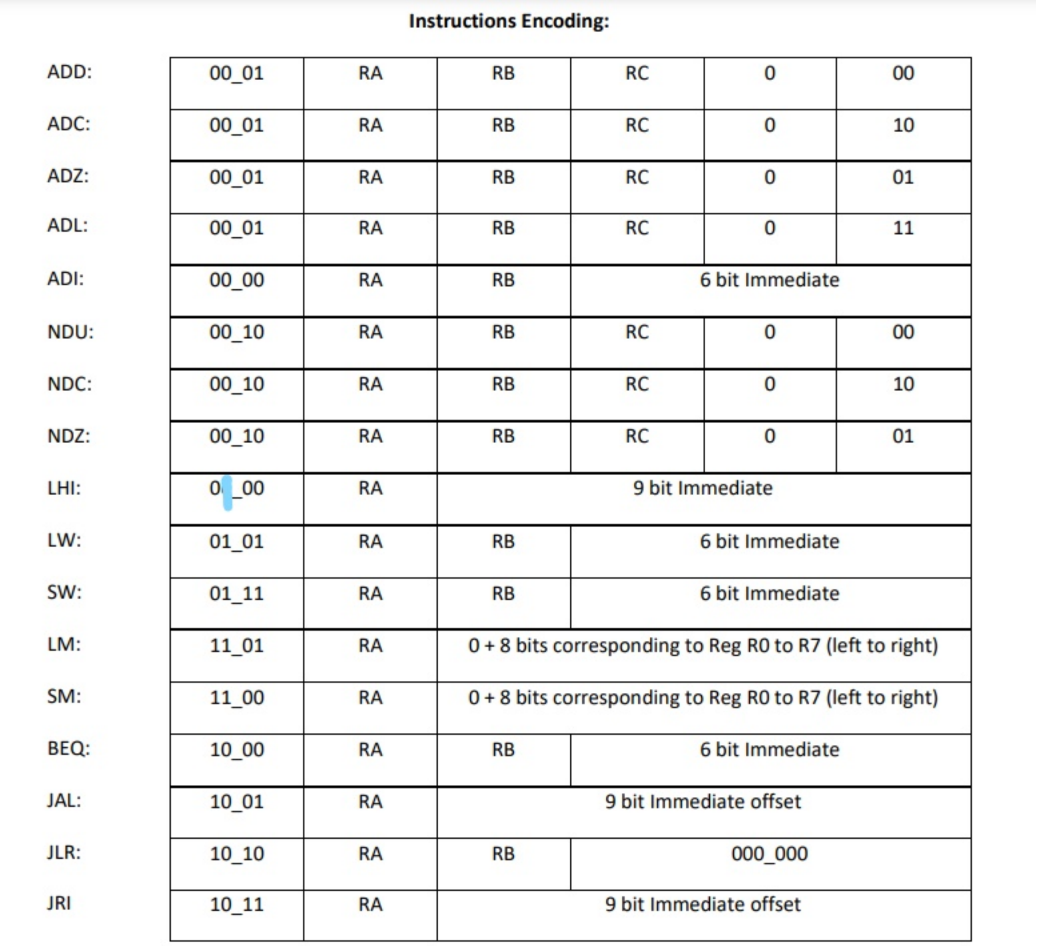
1 General Overview

We simulated the multi-cycle RISC as a state machine of **16** states, beginning from S_1 to S_{16}. Now, each instruction passes through a series of states (beginning from S_1) to execute, and then returns back to S_1, in time for the next instruction to execute. Each instruction takes a variable number of cycles to execute, varying between 4 to 8.
Each state executes on an average of 3-4 μ-ops, involving the components **ALU, Register File, Priority Encoder** and **main memory**. Our registers are of 16 bits each, and we have 8 registers `R0` to `R7`, with the 8$^{\text{th}}$ register `R7` storing the value of the program counter.
Coming to our memory, it is of 128 bytes, and it's where we fetch the instructions from, with the op-code of each instruction being 4 bits long.
The ALU implements elementary arithmetic and logical operations such as addition and `NAND`.
The priority encoder is used for the `load multiple` and `store multiple` instructions, and it returns the position of the least significant '1' in a number.

2 General Architecture

The initial state S_1 carries out a few housekeeping instructions such as the incrementing of the program counter. Also, all of the states are neatly partitioned such that each state either writes stuff, or reads stuff from memory, but never both.

3 Instruction Implementation



Instructions Encoding:

ADD:	00_01	RA	RB	RC	0	00
ADC:	00_01	RA	RB	RC	0	10
ADZ:	00_01	RA	RB	RC	0	01
ADL:	00_01	RA	RB	RC	0	11
ADI:	00_00	RA	RB	6 bit Immediate		
NDU:	00_10	RA	RB	RC	0	00
NDC:	00_10	RA	RB	RC	0	10
NDZ:	00_10	RA	RB	RC	0	01
LHI:	0[illegible]_00	RA	9 bit Immediate			
LW:	01_01	RA	RB	6 bit Immediate		
SW:	01_11	RA	RB	6 bit Immediate		
LM:	11_01	RA	0 + 8 bits corresponding to Reg R0 to R7 (left to right)			
SM:	11_00	RA	0 + 8 bits corresponding to Reg R0 to R7 (left to right)			
BEQ:	10_00	RA	RB	6 bit Immediate		
JAL:	10_01	RA	9 bit Immediate offset			
JLR:	10_10	RA	RB	000_000		
JRI	10_11	RA	9 bit Immediate offset			

Figure 1: Table of instructions, their encodings and their op-codes

4 Actual transitions of our Finite State Machine

Given below are the different FSMs required for the instructions. As one may see each FSM is shared by many instructions, whose names are written in the captions.

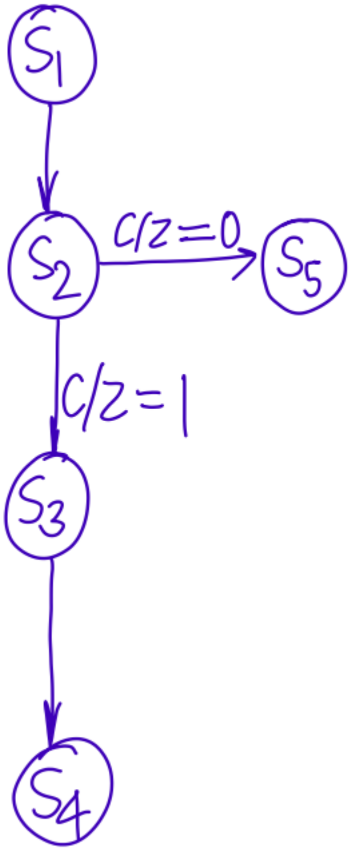


Figure 2: `ADD`, `ADC`, `ADZ`, `NDU`, `NDC`, `NDZ`

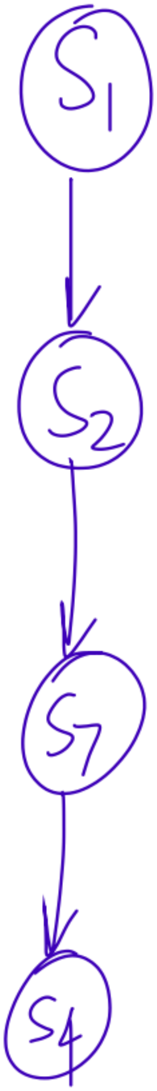


Figure 3: `ADI`

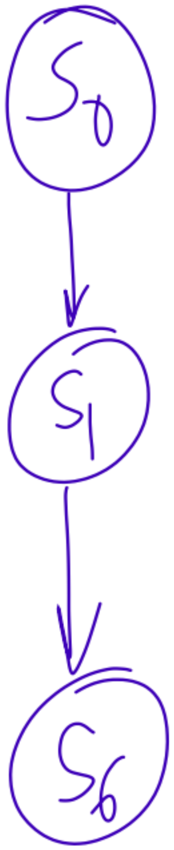


Figure 4: `LHI`

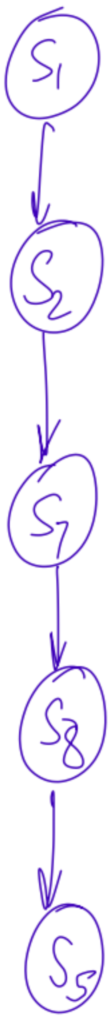


Figure 5: `LW`

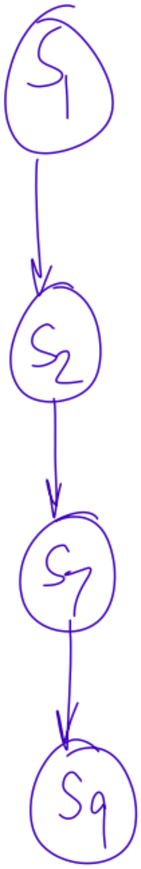


Figure 6: `SW`

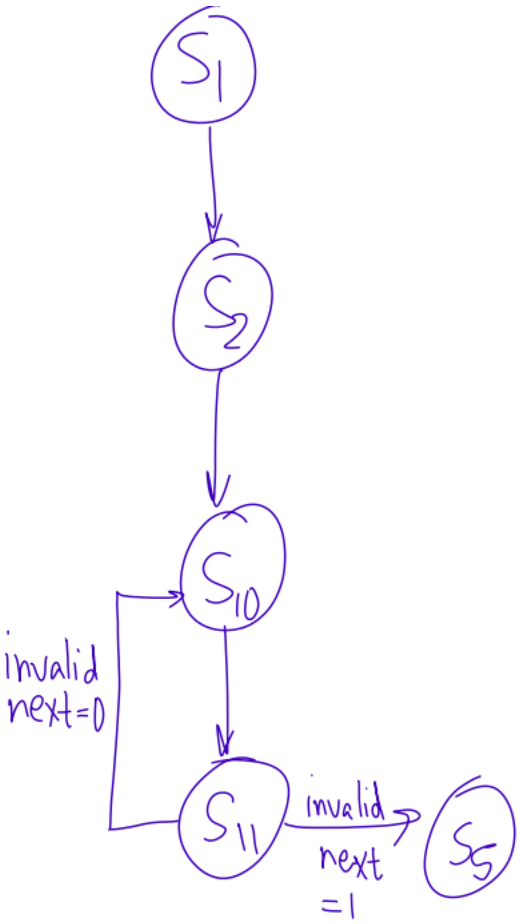


Figure 7: `LM`

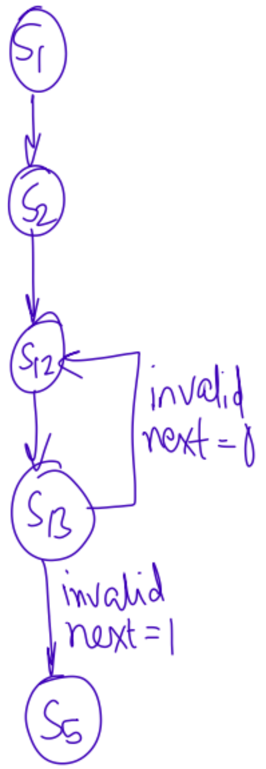


Figure 8: `SM`

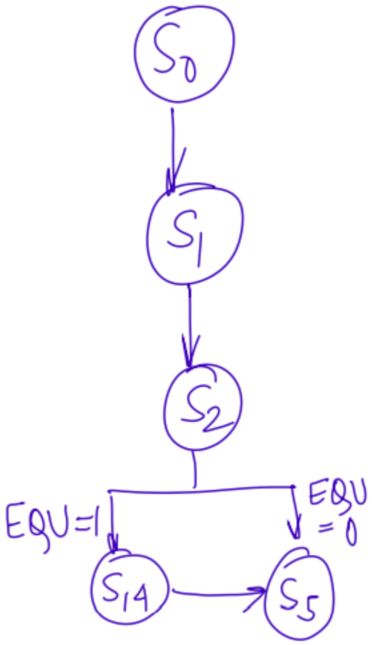


Figure 9: `BEQ`

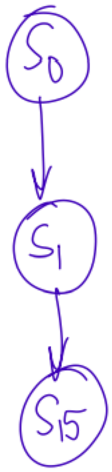


Figure 10: `JAL`, `JRI`

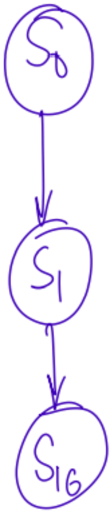


Figure 11: `JLR`