Chibuzo Nwakama, Anjela Bubiy

Part 1

Requirements for the first phase of the project are to finish the implementations of a finite state machine and instantiate each of the six modules to operate as shown in the Figure 1.

Connections of controls and their data signals are found in the “*sisc.v*” file. The *sisc* module has input signals *clk* and  *rst\_f* , and *ir*. Internal wires are declared to be used in the component instantiation. The components *mux4, rf, ctrl, statreg, mux32*, and  *alu* are connected with internal wires so as to follow the modules and data path depicted below. Monitor statements are used to monitor the following signals: *IR, R1* through *R6, RD\_SEL*, *ALU\_OP, WB\_SEL, RF\_WE*, and the 32 bit line connected to the write data input of the rf module.

The control file, “*ctrl.v,*” has the *clk*, and *rst\_f, mm , stat* as input signals that control the outputs *rf\_we, wb\_sel*, and *alu\_op*. States, opcodes, and addressing modes are initialized along with a present state and next stage register. The initial state is *start0*, which only appears at initial start. A sequential procedure progresses the FSM to the next state on the positive edge of the clock or resets the state to *start1* on the negative edge of the *rst\_f*. The combinational procedure which determines the next state of the FSM is set up as a case statement. If *rst\_f* is equal to zero, the next state is start1. Otherwise, the combinational procedure of the present states follows the state diagram in Figure 2.

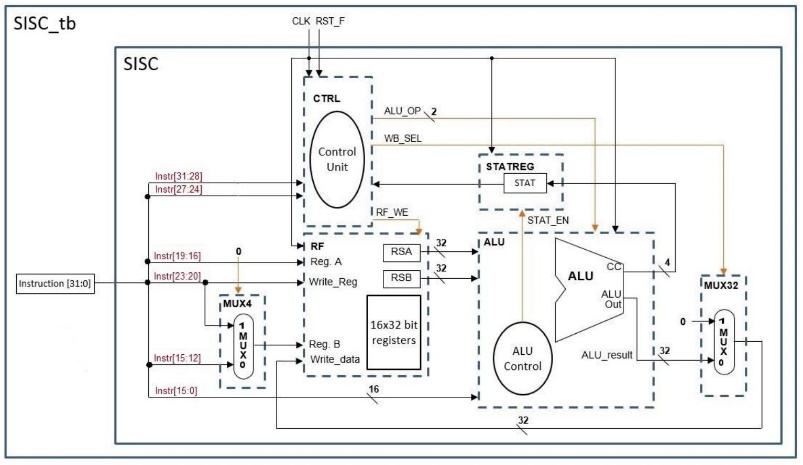


Figure 1 Simple Instruction Set Computer modules and data paths

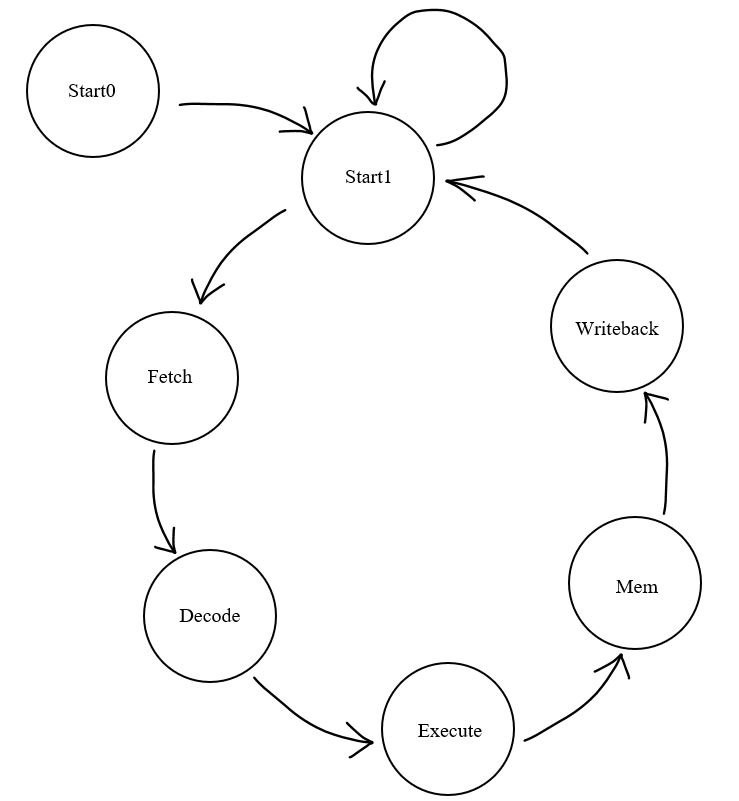


Figure 2 State diagram

Part 2

Using our solution from part one, part two of the project required us to connect the program counter control, instruction memory, and branch execution modules as shown in Figure 1. The fetch and decode state control the program counter and branch instruction. In the fetch state, *pc\_sel* is set to zero (not the default), *pc\_write* is set to one, and *ir\_load* is set to one. *Pc\_write* and *ir\_load* are used to update the program counter and load the next instruction. In the decode state, depending on the opcode, values are set to branch or perform an arithmetic operation. Branching occurs when *pc\_write* is set to one. Branch absolute sets *br\_sel* to one and branch not equals sets *br\_sel* to zero.

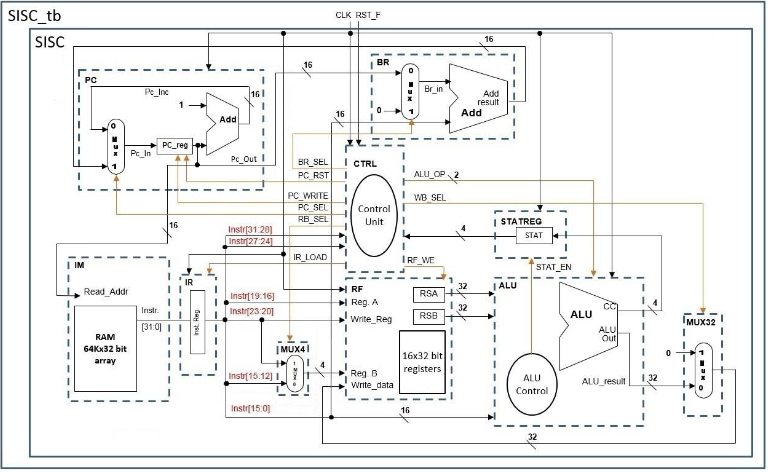


Figure 1 SISC schematic

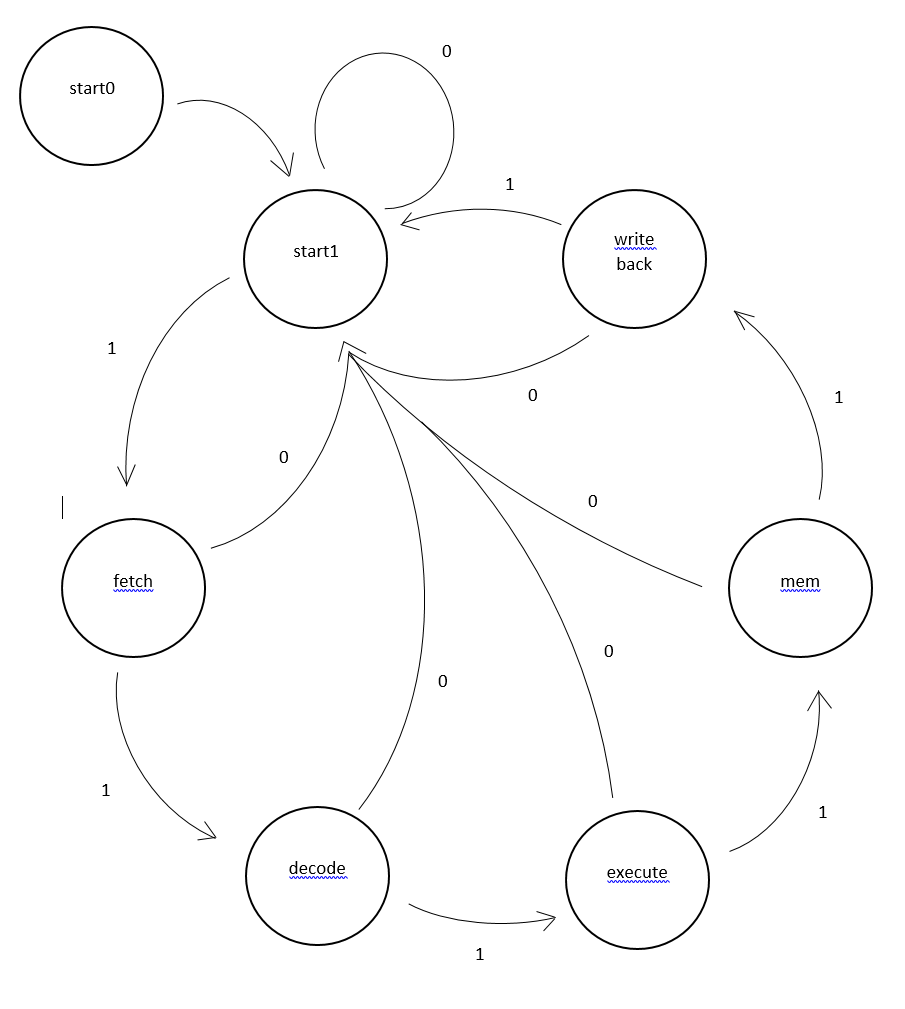


Figure 2 State diagram for SISC

Part 3

The addition that were added to the Simple Instruction Set Computer were the signals of mm\_sel, wb\_sel, dm\_we and rd\_sel on the control module. These signals allowed to use the load and store which were instructions LDA, LDX, STA and STX. These signals were ultized during steps of execute and memory for store instructions and the additions of decode and writeback for load instructions.

Additionally we had create a new instruction called SWP which uses a modified mux4 and rf module. These modifications were adding addition input to mux4 called in\_c. Also multiple control signals called switch and swap\_en\_reg that are used in the mux4 and rf respectively. These signals allows the SISC module to write to a register twice during a call of swap instruction. Also, the signals for mux4, rd\_sel and swap\_en\_reg are two bits in size.

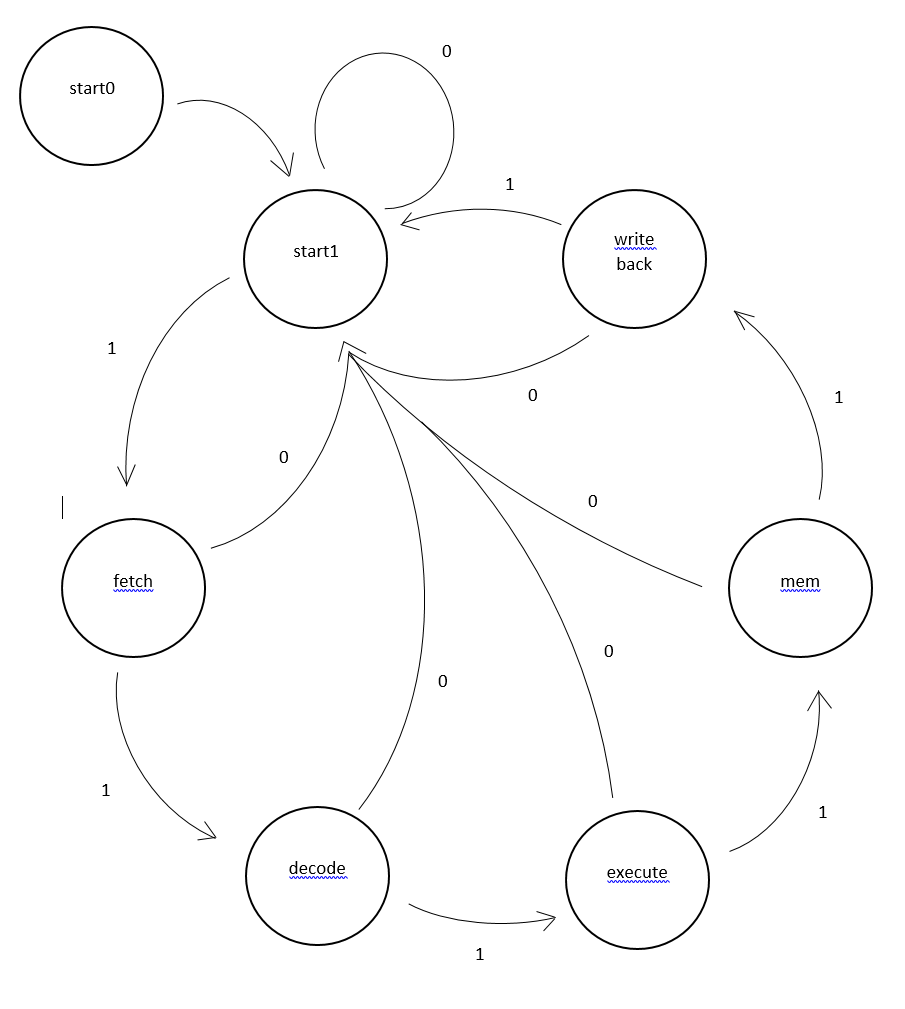
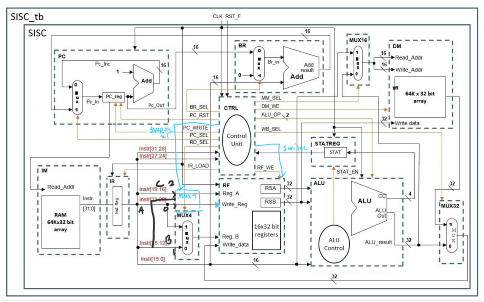


Figure 2 State diagram for SISC



Part 4:

There were no new signals added to the model, but we did add a new input called in\_c which was used in the mux16. The previous signals in the swap instructions were use again in the new instructions which were (Look at part 4). Conditions that were added related to mm which added condtion when the bits were set to 0001 or 1001. The signals that are used for the increment and decrement are wb\_sel, swap\_reg\_sel and rf\_we to write back the values back to RS register.

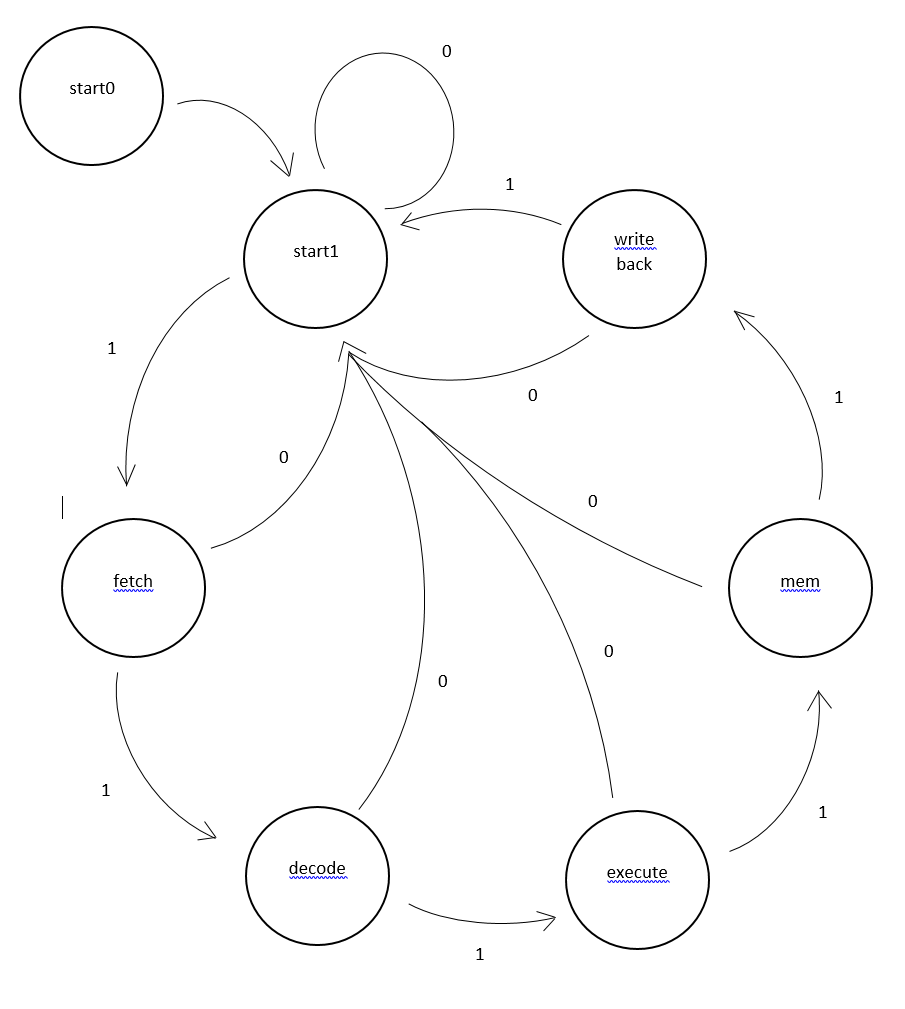


Figure 2 State diagram for SISC

