**ECE 385**

Spring 2021

Experiment #5

**Simple Computer SLC-3.2 in SystemVerilog**

Nathan Moskal and Daniel Marks

ABF – Friday

Abigail Wezelis & Harris Mohamed

**Part 1:** **Introduction:** In this lab, we built the SLC-3, a simple “computer”, in SystemVerilog. The computer operates in an FSM cycle of Fetch 🡪 Decode 🡪 Execute 🡪 Fetch 🡪 … In each cycle, the computer fetches an instruction from memory, decodes it with the Instruction Sequencer/Decoder (ISDU) Unit, then executes the instruction. The computer then recurses through the cycle continuously. The computer contains several modules that make up its structure: the aforementioned ISDU, the Program Counter (PC), Instruction Register (IR), Memory Address Register (MAR), Memory Data Register (MDR), Status Register (nzp), an 8x16-bit general data register file, and a 16-bit Arithmetic Logic Unit (ALU).

**Part 2: Written Description and Diagrams of SLC-3:**

1. **Further Summary of Operation:** The SLC-3 computer can perform several operations: ADD\*, AND\*, NOT, BR (Branch), JMP (Jump), JSR (Jump to Subroutine), LDR (Load Register), STR (Store Register), and PSE (Pause). ADD and AND have asterisks because they also have alternate forms ADDi and ANDi, which use immediate data from the command rather than register data for part of the operation. Which operation the computer performs is determined by the “opcode” of the instruction fetched from memory. In a FETCH cycle, the PC is stored into MAR.Then, the memory at MAR’s held address is stored to MDR, MDR’s value is passed to the IR, and PC is incremented by 1. In DECODE, the data from IR is passed to the ISDU. During the EXECUTE cycle, the ISDU uses the instruction from DECODE to perform an operation, and then writes the result to a register or memory location. Of the listed operations above, ADD, AND, NOT, and LDR set the Status Register. What that means is that after each of these operations, depending on whether the result is Negative, Zero, or Positive, the Status Register stores a 1 in either n, z, or p. When the system tries to execute a BR (Branch) statement, it will only branch if the Status Register’s value matches the statement. Branch commands are formatted as “0000nzpOffset[0:8]”, so 0000111x… would Branch if the result was negative, zero, or positive (so always). 0000101x… branches on nonzero values, etc. Other commands are simpler, as Branch is the only command that uses the Status Register. ADD, AND, and NOT take two register arguments, and output the result to a third register. ADDi and ANDi perform the same operation but with the immediate value from the command as the second operand. JSR stores the current Program Counter value into Register 7 and adds the 11-bit PCoffset to the PC. JMP is simpler: it copies an address from a register to the PC. Unlike Branch, which takes conditions, JSR and JMP will jump regardless of the status registers. JMP is often called on register 7 in order to return from a subroutine but can be used on any register to jump the PC. LDR and STR use the FPGA board’s memory to load data from (and store data to) the 8x16 register file. LDR stores into its destination register (DR) the data from the memory address at (BaseR + SEXT[[1]](#footnote-1)(offset6)), while STR stores the source register’s (SR) data to the address at (BaseR + SEXT(offset6)). In the LC-3 (not SLC) ISA, anytime something uses memory, that step stays until R is true, and R is a control signal that indicates the memory is ready for read/write operation. This control signal is not present in the SLC-3, so instead each operation that uses the memory instead waits at the read/write state for several cycles to hopefully ensure the data is utilized correctly. To answer the post-lab question on the topic, the implication this has for synchronization is that there is a possibility of an invalid read or write, which would have an unknown result. The data taken or stored from/to the memory can have an unpredictable value.

**c.**

Diagram, schematic

Description automatically generated\*ISDU

e. Written Description of all .sv modules

i. A guide on how to do this was shown in the Lab 2.2 report outline.

f. Description of the operation of the ISDU (Instruction Sequence Decoder Unit)

i. Named ISDU.sv, this is the control unit for the SLC-3. Describe in words how the ISDU controls the various components of the SLC-3 based on the current instruction.

ii. If you prefer to, you can lump this section into the module description section under ISDU.sv.

g. State Diagram of ISDU

Diagram

Description automatically generated

**Part 3: Simulations of SLC-3 Instructions:**

a. Simulate the completion of all 6 test programs, I/O Test 1, I/O Test 2, Self-Modifying Code, XOR, Multiplier and Sort.

b. Annotations for the above simulations, should include at a minimum, start of the test program, any user input (for example, entering the numbers in the multiplier test), and reading the expected result.

**Part 4: Post-Lab Questions:**

**a.**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| LUT | DSP | BRAM | Flip-F | Freq | StatP | DynaP | TotP |
| 1656 | 0 | 958464 | 1973 | 96.36mHz | 89.98mW | 8.21mW | 107.02mW |

**b.** MEM2IO

**Part 5: Conclusion:**

a. Discuss functionality of your design. If parts of your design did not work, discuss what could be done to fix it.

b. Was there anything ambiguous, incorrect, or unnecessarily difficult in the lab manual or given materials which can be improved for next semester? You can also specify what we did right, so it doesn’t get changed.

1. Sign Extended [↑](#footnote-ref-1)