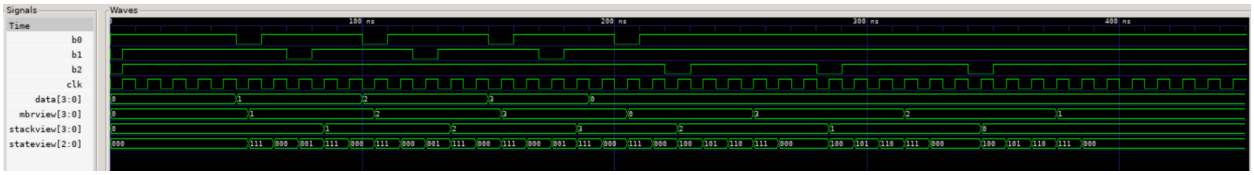
CS232 Project 6: Stack-Based Calculator

October 28th, 2024

This project extends the fundamentals established from the previous project, but instead of only having operations, the ROM now takes into account conditional and unconditional branches, extending the duration of simulations without actually adding more bits.

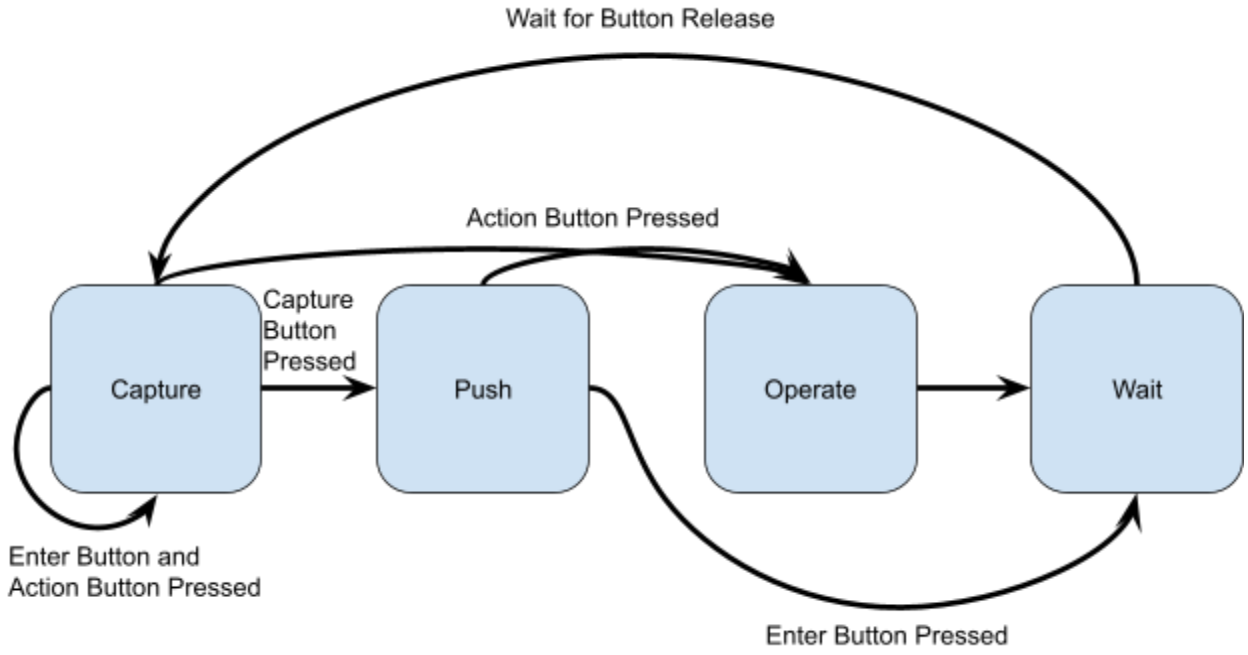
Task 1: Lab

This is to show that the lab program works.



Task 2: Calculator

State Machine Diagram:



Top Level Design:

This is a stack-based calculator using a state machine with four states: Capture, Push, Operate, and Wait. Here's a breakdown:

- Capture: This is the initial state, where data from the input switches is loaded into the Memory Buffer Register (MBR) when the capture button (b0) is pressed. The state moves to Push or Operate depending on whether the capture or action button (b2) is pressed.
    - The latter is for when there already is a value in MBR and the stack to have an operation used on, since all states lead back to this one.
    - There is also a reset option, which occurs when both the enter and action buttons are pressed. This resets all values, the stack, MBR, everything.
- Push: In this state, when the enter button is pressed (b1) the MBR content is written to the RAM stack (using ram_data_in), and the stack pointer is incremented. After pushing the data, the machine transitions to Wait to ensure the buttons are released. Alternatively, pressing the action button (b2) will move to Operate if there is another operand (in this case, a value at the top of the stack)
- Operate: This state performs arithmetic operations based on the selected operation, determined by 2 switches (op). It reads data from the top of the RAM stack, performs the operation with the MBR data, updates the MBR with the result, and decrements the stack pointer. Afterward, it transitions to Wait.
- Wait: The system remains here until all buttons are released, resetting the ram_wren to disable RAM writing. Once released, it returns to Capture.

The state transitions and button interactions are managed based on button presses (b0, b1, b2), while arithmetic operations are controlled by the op signal, allowing for addition, subtraction, multiplication, and division. The stack pointer manages the stack depth, and results are shown on the 7-segment displays (digit0 and digit1). This state machine provides a structured flow for capturing inputs, performing stack-based calculations, and displaying results.

Testing:

https://drive.google.com/file/d/1llBJgrhcZ72_EPh1q_BvfnpI8h6suarc/view?usp=sharing

This video tests the following operation: (10 - (7 - (3 + (2 * 1))) = 5 using the following set of operations:

-- operations:

-- mbr = 10

-- push MBR

-- mbr = 7

-- push MBR

-- mbr = 3

-- push MBR

-- mbr = 2

-- push MBR

-- mbr = 1

-- pop 2, and MBR = 2 * MBR = 2 * 1 = 2

-- pop 3, and MBR = 3 + MBR = 3 + 2 = 5

-- pop 7, and MBR = 7 - MBR = 7 - 5 = 2

-- pop 10, and MBR = 10 - MBR = 10 - 2 = 5

The following Extensions take place in 1 project: calculator2.

Extension 1: Operation Expansion

Sacrificing one of the input switches for another operation switch means 4 new operations could be added: MOD, XOR, AND, OR!

Extension 2: Data Expansion

Although the input stays at a measly 7 bits, the maximum data can go up to 16 bits. This means that the other 2 7 Segment Displays on the circuit can now be used.

Extension 3: Stack Overflow/Underflow Checks

This design also maintains conditions to prevent the stack pointer from going below zero or exceeding the RAM capacity: by checking if the stack pointer is already at the max value “1111” when pushing or if the value is already at the minimum value “0000” when operating.

Testing:

https://drive.google.com/file/d/1u5YlpSt4j50m1TPhTJaKRbJ2O2pweNp_/view?usp=sharing

This video tests the minimum stack amount and 4 7 segment displays.

https://drive.google.com/file/d/1WOnQaE7QUYqrga_dc-xmh2OgiuKsin7z/view?usp=sharing

This video tests the maximum stack amount. (16)

https://drive.google.com/file/d/1_pisPCOphFbtTk47tpsIinTajVzfps0j/view?usp=sharing

This video tests the following operations: (11 MOD (13 XOR (10 AND (9 OR (10 / (7 - (3 + (2 * 1)))))))) = 1 using the following set of operations:

-- operations:

-- mbr = 11

-- push MBR

-- mbr = 4

-- push MBR

-- mbr = 10

-- push MBR

-- mbr = 9

-- push MBR

-- mbr = 10

-- push MBR

-- mbr = 7

-- push MBR

-- mbr = 3

-- push MBR

-- mbr = 2

-- push MBR

-- mbr = 1

-- pop 2, and MBR = 2 * MBR = 2 * 1 = 2

-- pop 3, and MBR = 3 + MBR = 3 + 2 = 5

-- pop 7, and MBR = 7 - MBR = 7 - 5 = 2

-- pop 10, and MBR = 10 / MBR = 10 / 2 = 5

-- pop 9, and MBR = 9 OR MBR = 1001 OR 0101 = 1101 = 13

-- pop 10, and MBR = 10 AND MBR = 1010 AND 1101 = 1000 = 8

-- pop 13, and MBR = 13 XOR MBR = 1101 XOR 1000 = 0101 = 5

-- pop 11, and MBR = 11 MOD MBR = 11 MOD 5 = 1

Acknowledgements

As always, the TAs and Professor Li helped troubleshoot the base code, moreso this time because honestly it's very confusing and I made my own accidental errors!