**Problem Definition**

In this lab, our own version of I/O subroutines will be written for the small calculator program that was created for the previous lab. Instead of the keyboard, a keypad will be used as the input device in the calculator mode. The parallel port will be used for input in the calculator mode while the serial port will be used for input in all other modes. The keys on the keypad will be debounced when they are pressed by users, and the keypad should have auto-repeat functionality.

**Program Design**

**Input (Keypad) Subroutine:**

1. Set column pins as outputs(1) and row pins as inputs(0)
2. Drive column pins high(1) and row pins low(0)
3. Set up a delay time (1ms) for signals to become stabilized before getting row input. This can be done by using a delay loop.
4. The program can now proceed to retrieve row input.
5. If a key is pressed, a 30ms delay will be set to check for key rebounce.
6. The row input will then be stored in a data register.
7. Set up a delay time (1ms) for signals to become stabilized before getting column input. This can be done by using a delay loop.
8. The program can now proceed to retrieve column input.
9. If a key is pressed, a 30ms delay will be set to check for key rebounce.
10. The column input will then be stored in a data register.
11. Compare the row and column input and determine which key is pressed.

**Output Subroutine:**

* outcrlf and out\_string subroutine will remain the same as lab2.
* out\_char subroutine will be modified that Trap#15 will be replaced by the following:
  1. Set “Transmitter Enable” bit (b2) in UCR0 to 1.
  2. Keep looping until a key is pressed. This can be done by check if “transmitter ready” bit of USR0 (b1) is set to one.
  3. Write a character to UTB0.

**Debugging and Testing Strategy**

One method to test the subroutine is to press on every key on the key pad and see if the correct key is detected. Key debouncing can be checked by pressing on one of the keys and make sure only one input is taken.