Report For Lab 4

Connor McCullough

EEN 312

27 February 2013

**Objective:**

Implement a simple one digit calculator that can input button presses and ouput pressed numbers using a seven segment display. The calculator must have an add and a clear function. This project will make use of the concepts of conditional jumps, use of data memory, long jumps, and access of data memory using indirect addressing through registers.

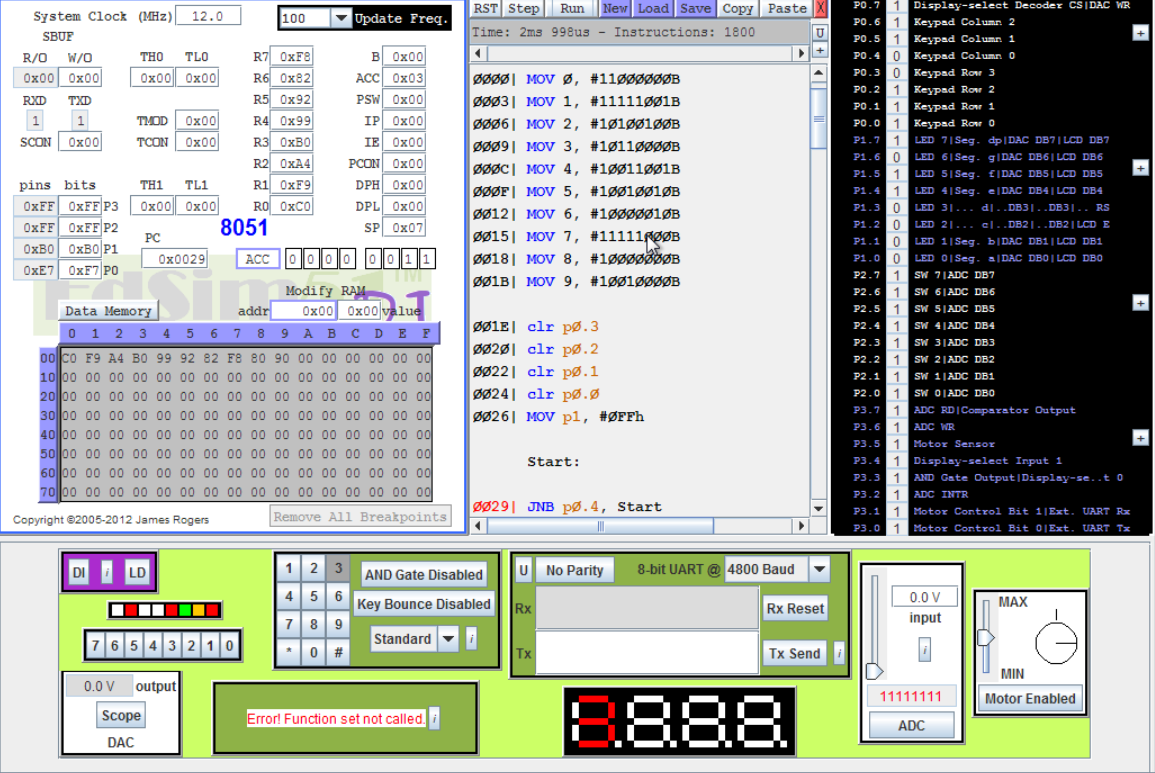
**Equipment Used:**

Software:

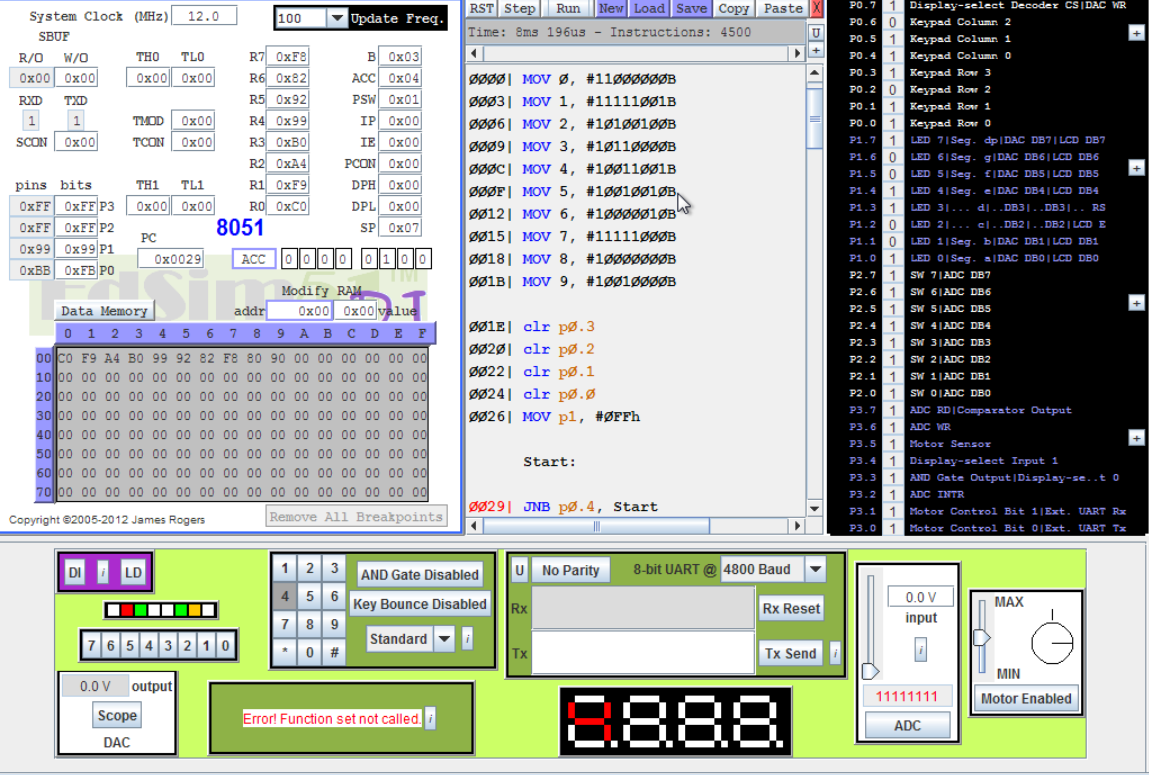
* Text editor and 8051 ASM assembler
* Step debugger to execute program one instruction at a time
* Data memory, accumulator, code memory
* Built in 7 segment display and input buttons

**Results:**

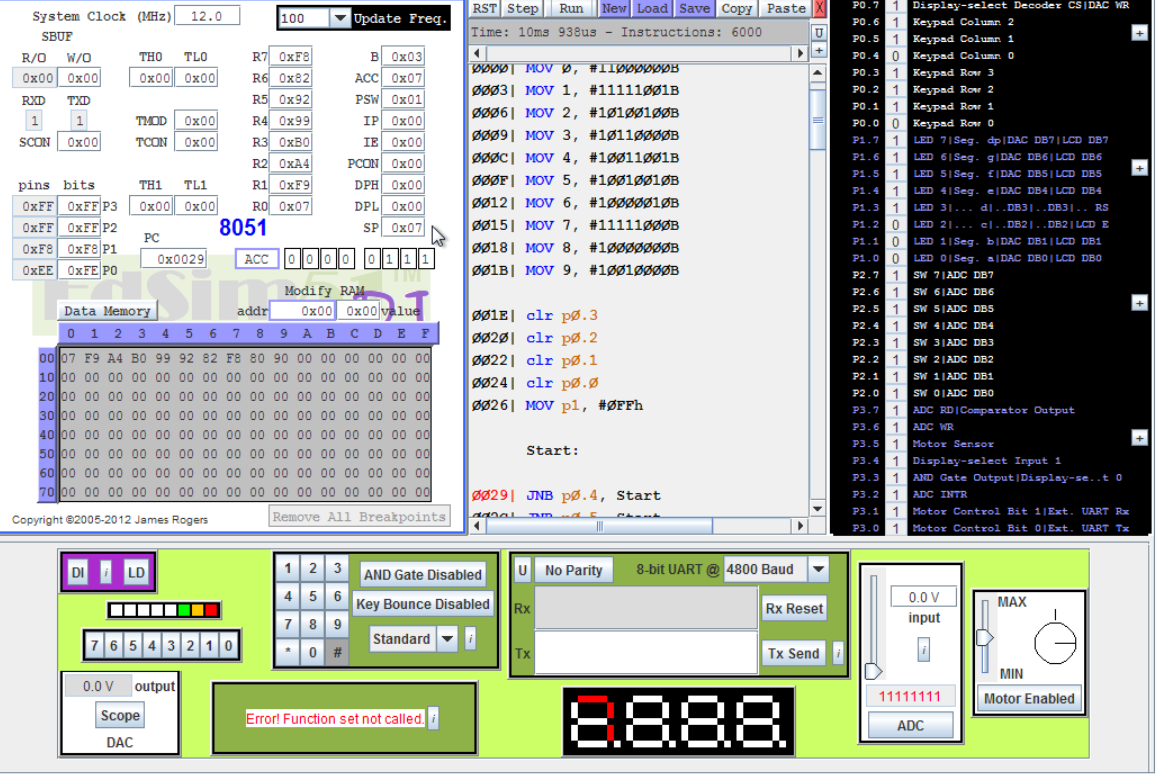
Button 3 is pressed:



Button 4 is pressed:



**Sum of 7 When # is pressed:**

****

**Conclusion:**

This lab taught how to interpret data coming from an input or being written to an output. In order for a key press to be detected, every single key must be checked for a press by grounding individual rows and buttons. The output for the seven segment display will be a binary sequence based on which lights of the display are one or off. This is different from the actual numerical value of the number. In order to account for this, the binary display values must be saved in their respectively numbered data memory addresses. When the accumulator returns a sum, this address must be moved to another register, and then indirect addressing will be used to access the value in the data address through the register and write it to the output pin.

**Code:**

Org 0h

MOV 0, #11000000B  
MOV 1, #11111001B  
MOV 2, #10100100B  
MOV 3, #10110000B  
MOV 4, #10011001B  
MOV 5, #10010010B  
MOV 6, #10000010B  
MOV 7, #11111000B  
MOV 8, #10000000B  
MOV 9, #10010000B ; Move display values to respective data addresses for each digit

clr p0.3  
clr p0.2  
clr p0.1  
clr p0.0 ; Ground all rows  
MOV p1, #0FFh ; Clear output

Start:

JNB p0.4, Start  
JNB p0.5, Start  
JNB p0.6, Start ; Check if all keys are released

setb p0.2  
setb p0.1  
setb p0.0  
clr p0.3 ; Ground row 3 only

THREE:  
JB p0.4, TWO ; Jump to next key if 3 is not presesd  
mov B, A  
mov A, #3  
mov p1, #10110000B ;pattern for 3  
JMP Start

TWO:  
JB p0.5, ONE  
mov B, A  
mov A, #2  
mov p1, #10100100B ; pattern for 2  
JMP Start

ONE:

JB p0.6, GND2  
mov B, A  
mov A, #1  
mov p1, #11111001B; pattern for 1  
JMP Start

GND2:  
setb p0.3  
setb p0.1  
setb p0.0  
clr p0.2 ; Ground row 2 only

SIX:  
JB p0.4, FIVE  
mov B, A  
mov A, #6  
mov p1, #10000010B ;pattern for 6  
JMP Start

FIVE:  
JB p0.5, FOUR   
mov B, A  
mov A, #5  
mov p1, #10010010B ;pattern for 5  
JMP Start

FOUR:  
JB p0.6, GND1  
mov B, A  
mov A, #4  
mov p1, #10011001B ; pattern for 4  
JMP Start

GND1:  
setb p0.2  
setb p0.3  
setb p0.0  
clr p0.1 ; Ground row 1 only

NINE:  
JB p0.4, EIGHT  
mov B, A  
mov A, #9  
mov p1, #10010000B ;pattern for 9  
JMP Start

EIGHT:  
JB p0.5, SEVEN   
mov B, A  
mov A, #8  
mov p1, #10000000B ;pattern for 8  
JMP Start

SEVEN:

JB p0.6, GND0  
mov B, A  
mov A, #7  
mov p1, #11111000B; pattern for 7  
JMP Start

GND0:  
setb p0.3  
setb p0.2  
setb p0.1  
clr p0.0 ; ground row 0

ZER0:  
JB p0.5, POUND  
mov B, A   
mov A, #0  
mov p1, #11000000B ; pattern for 0  
JMP Start

POUND:  
JB p0.4, ASTERISK  
ADD A, B ; Add most recently pressed key and previous key press together  
MOV R0, A ; Move result to R0  
MOV p1, @R0 ; Outputs display pattern of sum by indirectly accessing address given by sum  
JMP Start

ASTERISK:  
JB p0.6, JUMP  
clr A   
mov B, #0h ; Clear both A and B  
JMP Start

JUMP:  
LJMP Start ; Long jump back to start when start is out of range of normal jump

End