CS427 Lab 6 - 10
Due: 2/11, 2/18, 2/25, 3/4 & 3/11 at 4:00 P.M.

CALCULATOR

INTRODUCTION

Over the next 5 weeks you will be designing and building pieces of a single system. While the individual parts of the system are relatively simple, you will need to have a fairly complete design for the WHOLE system before you build anything or your pieces will never fit together.

The device that you are to build is a 4 digit, integer decimal calculator that will add and subtract. The calculator is to be HP like in that it will incorporate a 4 deep stack and will use reverse polish notation. For input the calculator will use the 4x4 key pad that is included in the kit. The figure below shows a possible layout for the key board. Your layout is to have the same keys, but you are free to rearrange them if you like. You are also free to use the spare key for an additional operation if you wish. (Roll stack for example.)

0	1	2	3
4	5	6	7
8	9	CLX	CHS
ENT		+	-

Note that it is possible to (carefully) remove the transparent plastic covers from the keys so that you can label the keys by sticking appropriate size pieces of paper underneath the covers. Do not label the keys with taped on or self sticking labels. This is a sin on a par with oiling your gun and will get you yelled at and/or cost you money.

CLX - clear the number currently being typed in to all 0's

ENT - terminate this number, push stack.

CHS - change the sign of the x entry in the stack.

The rest of the keys are self explanatory.

The 5 digit 7 segment LED display in your kit will be used for output. This will allow you to display 4 digits plus a sign. Note that the display allows

you to drive only one of the digits at a time. The display will normally display the contents of the stack x register. You will have to decide what you would like the display to be during the time that a calculation is taking place.

The calculator will have a 4 deep stack that will work like that in the HP calculators i.e. if more then 4 numbers are input to the stack, the stack will remember the last 4 with the earlier numbers falling off the bottom. The add operation is to add the contents of x to y, putting the result in x and dropping the rest of the contents (see attached handout on HP 10C stack operation). The subtract operation will subtract x from y, putting the result in x and dropping the stack. CHS will change the sign of x, and CLX will clear x to zero.

Each stack entry will have room for 4 digits plus a sign. If more then four digits are keyed in, the LAST digits will be lost. You will have to decide how you wish the display to appear as you key in the numbers and how you wish to store them in the digits of the stack word. Careful, this can have substantial impact on how easy/hard the calculations are to do.

It is ok to ignore overflow on addition or subtraction if you like (i.e. 9999 + 1 => 0000).

In addition to the keyboard for input, the display for output and the stack for storage, you will need an arithmetic unit to do the addition and subtraction. Note that BCD arithmetic is a bit more complex then binary arithmetic. For one thing the binary addition of two BCD digits results in something which will (at least sometimes) require additional manipulation before it is recognizable as a BCD digit and a carry/borrow. Additionally, subtraction will result in the nines compliment of the desired result if you guess wrong about the relative magnitude of the two numbers. You will probably want to build a digit serial arithmetic unit.

Finally you will need a master control state machine, a clock, a master clear button, and some miscellaneous registers to use as temporary storage when moving things about in the machine. It will probably pay to save enough EPROMs to build a stored state machine.

Week 1

Figure out your overall design strategy and prepare a block diagram showing the various modules of the calculator and the data path(s) that interconnect them. Figure out what functions each of the modules will perform and list them. Describe how the modules will cooperate to perform the various operations.

Design and build your system clock and a master clear button. You will want a single step capability like the clock in the design example from last week. For now you may assume that the frequency doesn't matter, but it may pay to build it in a manner that makes changing the frequency latter easy.

Design and build a key board controller that responds to any key being pressed by setting a HIT signal, and by generating an output which is the BCD code of the key pressed if the key is a digit, or a signal named for the key if other then a numeric key is pressed. The signals will need to be latched so that the control can recognize and respond to them in its own good time. Additionally, HIT will have to be resetable when both a line from the control so requests and the key that was pressed has been released.

The keyboard is a simple device which has 4 row wires and 4 column wires. When a key is pushed the row and column wires which coincide under the key are shorted. The pinout is easy to determine by turning the keypad over and looking at the circuit board on the bottom. The brown lines are the the row wires and the metal lines are the column wires. You will have to figure out how to decode the switch. You will also have to figure out if the switch bounces, and if so how to debounce it.

Wednesday checkpoint ### Demonstrate that your clock works in both free run and single step mode. Show your design for the keyboard control to the TA and convince him that it will work.

Friday checkpoint ### Demonstrate that your keyboard control works.

Week 2

Design your stack using one of the 2114 RAM chips. Include whatever additional logic, control lines, data busses etc. that are necessary in order for the master control to be able to load data from the keyboard into the stack, push the stack in response to enter, clear the x register, extract data for the display and for calculations, drop the stack after a calculation, change the sign etc.. Generate a list of the exact steps that will be necessary in order to perform the various operations. Generate timing diagrams that demonstrate that data will be correctly written into and read out of the stack when your clock is free running. If you need to adjust the frequency of the clock to assure that the stack will work correctly, do so.

Wednesday checkpoint ### Demonstrate that it is possible to load data from the keyboard BCD output register into some location in the memory and then read it back out. You may use the clock in single step mode and switches and lights to control and monitor this operation.

Friday Checkpoint ### Demonstrate that a sequence of inputs from the keyboard can be loaded into the x register correctly and that the stack can perform an enter, a change sign, a clear x, and what ever sequence of operations will be used to output data from x for the display. You may use the single step clock and may drive the control lines into the stack with switches and monitor the operation with lights.

Week 3

Design and build the display controller and the arithmetic unit.

Wednesday Checkpoint ### Demonstrate that your display works. You may use the single step clock and control the display inputs from switches.

Friday Checkpoint ### Demonstrate that your arithmetic unit works. You may use the single step clock, control the inputs from switches, and monitor the outputs with lights.

Week 4

Design and build the master control state machine. Integrate the entire project and debug it.

Wednesday Checkpoint ### Demonstrate that your master control state machine works, using the single step clock, switches to control the inputs and lights to monitor the outputs.

Friday Checkpoint ### Demonstrate that the assembled calculator works at least in part using the single step clock.

Week 5

Complete debugging of the assembled calculator, update all documentation and hand in your kit by 5:00 PM on Friday.

Wednesday Checkpoint ### Demonstrate that your calculator works in part using the free running clock.

Friday Checkpoint ### Demonstrate that your calculator works completely using the free running clock.

Note: This weeks check points will be worth 3 points each. Completing the documentation will be worth 4 points. 10 extra credit points will be given for any calculator that works absolutely completely with no bugs of any kind.