ENGI E1112 Departmental Project Report: Computer Science/Computer Engineering

Gabriela Melchior
Alysia Sanchez
Jessica Wang
Dong Yeop Kong

December, 2011

Abstract

We were given an HP 20b calculator that had been stripped of its abilities to act as a calculator should, and received instruction to return it to working order. We accomplished this through a series of labs that had us reprogramming the calculator one step at a time. Each lab assigned a new task, and each relied on its predecessor. As the programmers, we had many decisions to make. Our decisions included determining which route to take in order to accomplish each task, the way the calculator displayed characters, and the limitations of the calculator. This lab work offered us a deeper understanding of the way simple computers (calculators specifically) work. It also gave us experience in reading and programming in C, which is the programming language on which the HP 20b calculator operates.

1 Introduction

In this Computer Science project, we explored programming possibilities on an HP Calculator. The objective was to understand how these devices work so that we can manipulate them. Prof. Edwards configured the calculator so that it would lose most of its usual utilities and also run C language programs. Our group's job was, with each lab, to create script that would add more functions to this gadget.

This Report is divided into sections that analyze different parts of our experiment. We first explain how to use the device and how do our programs work on it. We then consider the social implications of the project and how does it affect the community in general. Later, we take a look at the components of the calculator and how do they work together to achieve the total functionality. We included samples of the code and described problems we had. Finally, we look back at the project as a whole, discuss our experience and offer criticism.

2 User Guide

a) The calculator is programmed to display numbers starting from the left side of the screen to the right side.

b) To input numbers, simply press the keys that the number is comprised of, then press “INPUT”. For example, if the user wishes to input the number “3481”. They would press “3” “4” “8” “1” “INPUT”. To input “7”, press “7” “INPUT”.

c) Users can only input numbers that are twelve characters long or less, and the calculator can only display results of twelve characters or less. If the input or output is longer than twelve characters, the number shown on the display will not correspond to the true value of either the input number or the calculated result.

d) The calculator operates as an RPN calculator, and users should treat it as such. Typing “4” “INPUT” “-” “3” “INPUT” or “4” “-” “3” will not yield accurate results. To subtract three from four, a user should type “3” “INPUT” “4” “-”.

e) Because this is an RPN calculator, it can save several numbers at one time. However, it can only hold ten. If a user attempts to input more than ten consecutive numbers, the calculator will display “ERROR”.

3 Social Implications

In this project we created a calculator that can be very simply used. Calculators such as this would be cheaper and easier to manufacture, therefore countries with low budgets on education could use them in their schooling systems. RPN calculators like ours can also be used faster than regular calculators. Compared to math classes without calculators, with calculators can be more efficient, as students and teachers can perform arithmetic much faster than in their head. Other departments such as physics and chemistry can also benefit from such an improvement.

4 Platform

The platform we used to implement the processor was a jtag4port, which is built into the sam7l. Our development environment consisted of Linux workstations with jtag adapters connected to the USB ports. On the workstations, we used a software package called “OpenOCD”5 that communicates to the sam7l CPU through USB and jtag.

Figure 1 shows how the hardware works. The 20-pin connector is plugged onto the calculator’s jtag connector with the red wire (pin 1) on the left when you are looking at the back of the calculator. The jtag connector extends past the calculator’s 16 connector pins on only the right side. Figure 3: Connecting to the development ports on the back of the HP 20b calculator. Note that the jtag connector extends beyond the header. [2]

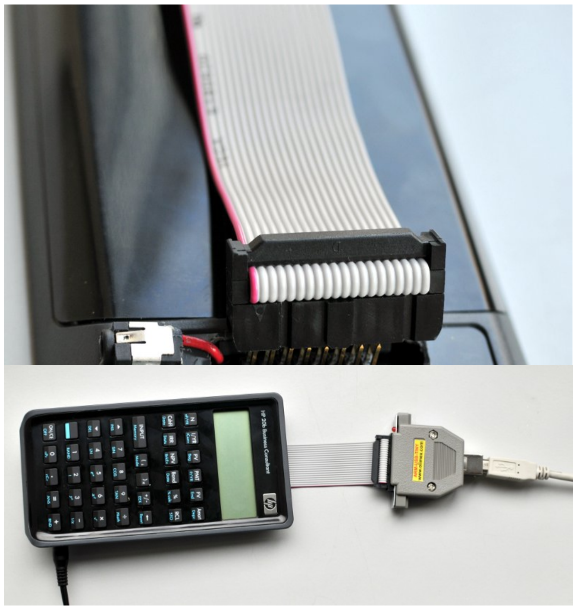

Figure 1: Connecting to the development ports on the back of the HP 20b calculator. Note that the jtag connector extends beyond the header.

4.1 *The Processor*

The processor specifications are:
Processor - Atmel 30MHz, low-power AT91SAM7L128 ARM7 System on chip that runs at a speed of 30MHz. It is PLL controlled. [1]

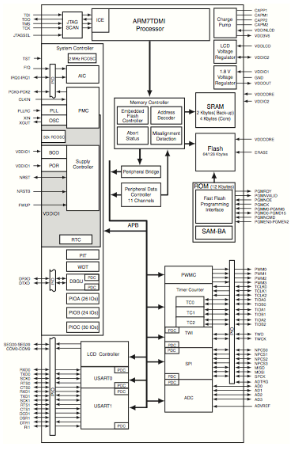

Figure 2: A block diagram of the AT91SAM7L microcontroller that is at the heart of the HP 20b

4.2 *The LCD Display*

LCD display - 1.5 lines x 12 characters + 3 exponent; 7-segment bottom line, 8 character scrolling top line; Display contains two (2) alphanumeric lines in an LCD display. The first (top) line displays eight (8) characters in a scrolling display feature, plus 11 hard-wired indicators. The second (bottom) line displays 12 characters, plus a three (3) digit superscript display (primarily for showing exponents). The calculator provides an adjustable contrast feature. [1]

4.3 The Keyboard

We found that most keyboards use a matrix format, so that the computer knows which key is pressed by knowing which row and column the key is in by conducting electricity through them. The keyboard consisted of plastic sheets with lines drawn on them. These lines formed n X n matrix and allowed the machine to know which key was pressed by conducting electricity through them. The computer keyboards had a rubbery mold over them to keep the keys bouncy. The keyboard of the HP20b is a 6×7 matrix.

5 Software Architecture

The program we created for the calculator comes from four separate labs we completed. The first lab entailed creating a program that would take a string of characters and endlessly scroll it across the calculator’s display. Though this program was not used again, it did provide an introduction into programming in C. This lesson gave the foundation for the labs that followed.

The second lab “listening to the Keyboard”, was about understanding the mechanics behind how keyboards worked, and applying this knowledge to write a program for the keyboard on the calculator. The program we created was capable of pinpointing which calculator key, if any, were pressed.

“Listening to the Keyboard” and “Entering and Displaying Numbers”, the second and third labs, are closely tied. After using the previous program to determine which key was pressed, our new program displayed that key. However, where Lab 2 provides the tools to display one key at a time, Lab 3 built upon this to allow a string of numbers to be displayed. That is to say, with Lab 2 one could press the “3”, “4”, and “5” buttons, and the calculator would first display “3”, then replace that character with a “4”, and then replace “4” with “5”. Lab 3 takes the key presses “3”, “4”, and “5”, and displays “345” as a single number.

The last step in rebuilding the calculator was to turn it into an RPN calculator. Labs 2 and 3 have already provided us with a means of gathering which buttons are pressed and turning that knowledge into a number that can be manipulated with mathematical operations. This final program uses knowledge provided by its predecessors to create a number, store it, and continue storing numbers. It also uses its predecessors to determine when an operation is pressed, then performs that operation - multiplication, division, addition or subtraction - and places a new number back into the array of stored numbers it already has.

6. Software Details

6.1 Lab 1: A Scrolling Display

In the first lab we started exploring the programming possibilities of the calculator. Our first task was to display a phrase and make it slide across the screen. As you can see from our code below (1) we did it by using arrays. We created an array “my string” which contains what we wanted to display plus blank spaces. We used a for loop to have it slide along the screen. We saved the last character to a temporary location and have it start over afterwards, in an infinite loop.

A problem we encountered in this lab was that the calculator would display the word so fast that you could barely read the string. To solve this problem we implemented an arithmetic calculation that would continuously take some of the memory space and therefore slow down the string display.

Code Lab1:

```
#include "AT91SAM7L128.h"
#include "lcd.h"
int main()
{lcd_init();
int i, j, k;
char myString[] = "SEAS ";
char temp;
int length = 14;
int size = strlen(myString);
for( ; ; ){ // Display on Calculator
for(i = 0; i < length; i++){
//slow down
for(k=0; k < 60000; k++)
{
j = k; // dummy
j++;
}
lcd_print7(myString);
// temp stores the last character of myString
// before it gets overwritten
temp = myString[size1];
// Shift everything by 1
for (j = size 1; j > 0; j) {
myString[j] = myString[j 1];}
// Put back the last character of myString
// into the beginning of the myString
myString[0] = temp;
}
}
return 0;
}
```

6.2 Lab 2: Scanning the Keyboard

In the second lab we had to assign different characters to the keyboard and have the calculator display them as a key is pressed. To do so we first created a matrix with all the characters on the calculator. Since the first two rows have more keys then the others, we assigned an extra value of 0 in the end of the shorter rows, so that the matrix would be balanced.

To test if a key is pressed we set the general voltage to high and then set a specific column to low. If a key is being pressed on this row one of the locations in the low column will read high. We then use the number of the column and the row to identify which character in the matrix is to be used. We can then have the calculator execute these instructions on an infinite loop and it will always display which key is being pressed. If no key is being pressed the function will continuously output -1.

6.3 Lab 3: Entering and Displaying Numbers

In the third lab we had to let the user input a number several characters long. To do so we created an empty array and checked for input. As each number is pressed it is added to this array and displayed in the calculator. If the user inputs something that is not a number the array resets and allows for a new number to be entered. The number is converted to an integer and sent to the *struct* in keyboard.h, as well as the operation that reset the array.

A challenge we encountered in this lab was that to send the *number* to the *struct* it had to be in *integer* format, while to display it in the calculator it had to be a character. We solved this by converting the array back and forth: we converted each element of the array to *int* and then multiplied each digit the appropriate number of times by 10, so that it is in the correct decimal place. Finally we added up all of the digits to obtain a final number. To convert back to characters we did the same but instead of multiplying by 10 we divided.

Another problem we had was how to identify if a key is being constantly pushed or if it was pushed and released. To overcome this issue we created nested if statements that would only add a digit to the array if the output of *keyboard_key()* was *-1 char -1*. This way we would be sure that if a key was held down it would only be in the array once, while still allowing the user to input the same key twice in a row.

6.4 Lab 4: An RPN Calculator

In the fourth and last lab we had to make the calculator behave as a calculator. To do so we created an infinite loop that would save the numbers entered as an array. It would continuously add to this "stack" until the user input an operation; instead of simply the key enter. It would then perform the selected operation as a RPN calculator; meaning that if the keys entered were: "2" "enter" "3" "+" the operation would be 2+3.

A problem in this lab was how to compute when an operation was pressed two or more times in a row, without numbers in between. To solve this we had to first understand that when an operation is pressed without a number the number is automatically set to INT_MAX. We then wrote our program as to save the operation to be used before it saved the number. In between those instructions it would check and see if the number was INT_MAX, and if it was it would use the previous number instead to perform the operation.

7 Lessons Learned

First of all, we have learned about C programming language. Some of us already had background information while some did not. Although we did not know every function of C language required to complete the labs in the beginning, with the help of our TA YoonJi, we figured out how to tackle the problems we encountered. As we continued to work on the labs, we were able to understand how those seemingly meaningless letters and symbols come together to form a program, making this machine act as we wanted. In addition, we dissected a keyboard and scrutinized it. It was a good opportunity to not only get familiarized with the software, but also to take a look at the hardware. After completing all the labs, we were surprised to see how this metal junk that displayed only hello on the screen came to function as calculator again. Overall, it was an enjoyable experience. As a group of future engineers, we learned to point out mistakes we made and work together to accomplish our goal.

We believe that future students who consider taking this course should have some sort of background knowledge in computer programming or at least take a computer science class at the same time. Unlike other lab courses like chemical, mechanical engineer, computer science is less exposed to high school students.

8 Criticisms of the Course

One of the issues we consistently had was power connection issue. It was frustrating to see an error when we were about to test the code we had just written. It should be fixed for those who will be working next year on the computer we used.
What could also be helpful would be to offer a few classes in basic programming or at least programming logic, so that new students can get acquainted with the subject

References

[1] Hp-20b repurposing project. Online http://www.wiki4hp.com/doku.
php?id=20b:repurposing_project.
[2] Stephen A. Edwards "Repurposing an HP CalculatorLab 1: Hello WorldComputer Science and Computer Engineering Gateway Project"
http://www.cs.columbia.edu/~sedwards/classes/2011/gateway-fall/hello.pdf