ENGI E1112 Departmental Project Report: Computer Science/Computer Engineering

Ross Basri and Ruchir Khaitan

May, 2012

Abstract

This report details the results of a term project in ENGI E1112 in Computer Science/Engineering. The goal of this project was to repurpose an HP20b calculator by replacing the existing firmware with new software built to given specifications. The final product of the project was a functioning, reverse-polish notation calculator.

The layers of the software implementation, in the order of which they were addressed, are displaying output, scanning the keyboard, entering and displaying numbers and operations, and internalizing mathematical input to produce a result. Students were tasked with designing and implementing algorithms to handle each of these aspects in groups of two or three.

Programming for this project was done in C using a LINUX workstation. A JTAG adaptor was used for communication between the workstation and the HP20b processor. As planned, students were able to produce a fully functioning, reverse-polish notation calculator by the end of the term.

1 Introduction

The design challenge for the ENGI E1112 Departmental Lab Project in Computer Science and Computer Engineering was to program and integrate new firmware for an HP20b calculator (Figure 1). Students worked in groups of two or three to replace the existing firmware with their own original and custom-designed software.

The ultimate goal of the design challenge was to build a functioning, reverse-polish notation calculator. In order to accomplish this, the software architecture students set out to build had to handle input from the keyboard, internalize the input and formulate a mathematical result, and display the result.

Students attacked the design challenge through a series of four labs. Each lab built upon concepts and software architecture formulated in previous labs, and the result was incremental steps toward the final goal of a functioning, reverse-polish notation calculator.

2 User Guide

Reverse-polish notation (RPN) must be used to analyze mathematical functions on the repurposed HP20b calculator. This is often referred to as a âĂIJpost-fixâĂİ notation because the operands are listed before the operator is input and applied. For example, 4+4 in RPN would read 4 4 +. Similarly, 3 * 7 would look like 3 7 * .

When specifically using the repurposed HP20b calculator to analyze RPN functions, the first thing that should be done is to turn the calculator on using the ON/C button. Once the desired software is loaded, check to see that the display reads 0. If anything but 0 is shown on the display, press the = button (this acts as a clear).

Once the display reads 0, the calculator is ready to receive input. To input the first number of any mathematical operation, you must hit the desired number key and then the INPUT key, which looks like an up arrow. Once the INPUT key is entered, the display should still show only the numeral entered. If an operation is entered first, there will be an underflow error, and the display will read UNDERFLOW.

To enter the second operand, simply hit the key of the number desired, and then the operation. The full sequence on the repurposed HP20b calculator to compute 4+4, for example, would be 4 INPUT 4 +. The calculator immediately displays output once and operation is entered, so the display should now read 8.

To compute more complex operations, such as 4+4*3+3, you must first compute each of the terms separately, and then multiply them. For example, the expression above would be entered as 4 INPUT 4 + 3 INPUT 3 + *. Once the first operator + is applied, 8 should be displayed as output. Then, once 3 is input, 8 will be stored in the processor and recalled to be applied as an operand once the * operator is entered. The display should finally read 48.

Users should be aware of the fact that the software implementation used in the repurposed HP20b only allows for five separate terms to be used in any single calculation (see section 6.4 to understand why). If more than five terms are entered, an OVERFLOW message will appear on the display. The same message will be appear if the display value exceeds the maximum allowable spaces on the display. Additionally, users should be alert to the fact that the division operation

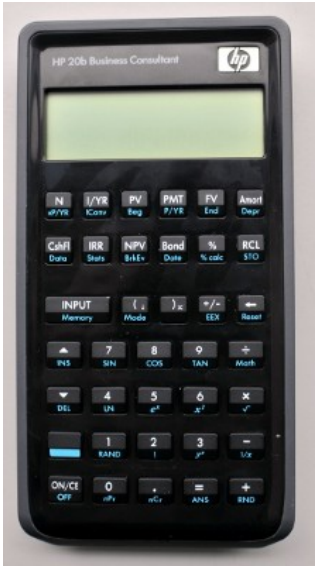

Figure 1: The HP 20b

is not functional on the repurposed HP20b calculator. This is due to a software glitch beyond the scope of the project, and is not a missed goal of the engineers.

3 Social Implications

This platform, with which we practiced embedded systems programming, is far more versatile than merely reprogramming basic arithmetic functions and reverse polish notation. It is not a very large leap from that point to programming all other math functions and infix notation. Beyond that, it is not inconceivable to imagine the HP-20b as a very primitive text editor and 8 bit game platform. Thus, the HP-20b is a perfect example of technology trickle down, what was once a ground breaking personal computer can now be used as a cheap teaching platform in developing countries, where electricity is often unavailable and funds often scarce. The HP-20b can be a valuable tool to introduce kids young and old to programming and application development, with less of a reliance on standard computers, that could lead to new apps for modern phones. Thus, the next Steve Jobs or Bill Gates might spring from places we would never have expected.

4 The Platform

The three main components of the hardware platform of the HP20b calculator are the processor, LCD display, and keyboard. These serve as the three main tools students used to approach the problems of design and implement their solutions.

4.1 The Processor

The standard processor on the HP20b is an Atmel AT91SAM7L128 ARM7TDMI-core low-power microcontroller in CoB packaging (See Figure **??**). It runs at a speed of 30MHz and it is phase-locked loop (PLL) controlled, simply meaning that the output signal is kept in phase with the input signal. The processor includes 128KB of programmable, flash memory and 6KB of RAM, 2KB of which is non-volatile [1].

4.2 The LCD Display

The LCD (Liquid Crystal Display) contains two alphanumeric lines to which output can be written (See Figure 3). At the beginning of the project, students were given a function library, including the function lcdputchar7, which takes as an argument a character in ASCII and an integer indicating the column on the display to which the given character should be placed. This function played an integral role in building other functions facilitating the display of output. One more crucial function was lcdinit, which initialized the LCD to display output and without which display would not have been possible. This function was called at the beginning of all main methods.

4.3 The Keyboard

The last piece of hardware architecture that served a crucial role in approaching the design challenge was the keyboard. The keyboard is scanned as a 6 x 7 matrix ([1]). The exact hardware mechanism that allowed scanning of the keyboard was a circuit running underneath the keyboard and broken by the spaces between the keys. Beneath each individual key lies a conductor that completes the circuit when pressed. To check if a specific key is pressed, a controller compares the voltage in the circuit on both sides of the key. If the key is pressed, the circuit is completed and the voltage will be the same, otherwise, there will be a potential difference, and the controller determines that the key is not pressed (See Figure **??**).

5 Software Architecture

The reprogrammed firmware consists of four functions that build on each other and provide the foundations for each successive module of code. The first is the function for printing integers on the LCD screen, which is an integral part of the calculator function. After that, the second function detects when a key is pressed and using the previous code, prints it on the screen. Extending that functionality, the third function takes meaningful calculator entries by reading a number and an operator from user input and displays them on the keypad. The final code builds on all of these input and output capabilities and uses the processing power of the

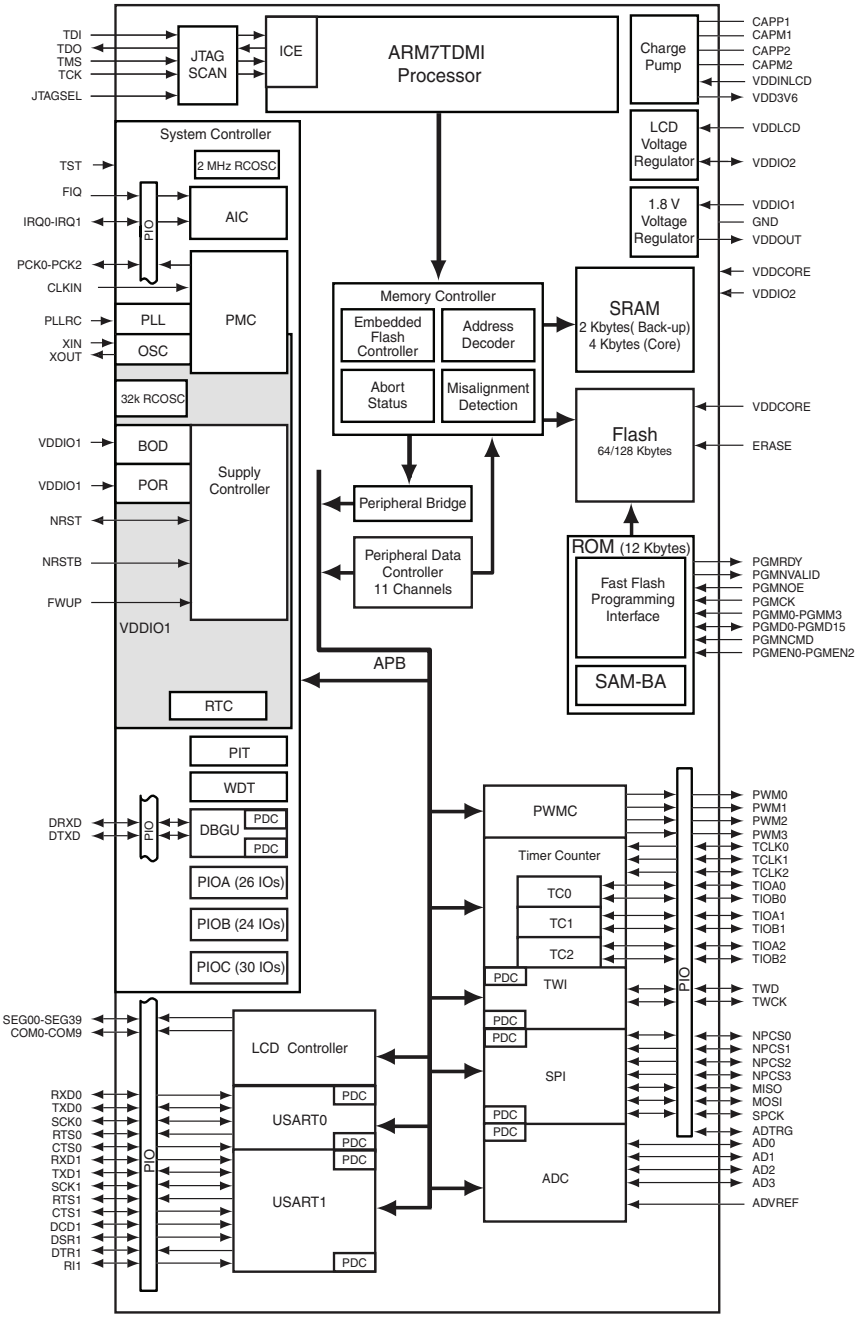

Figure 2: A block diagram of the AT91SAM7L microcontroller that is at the heart of the HP 20b

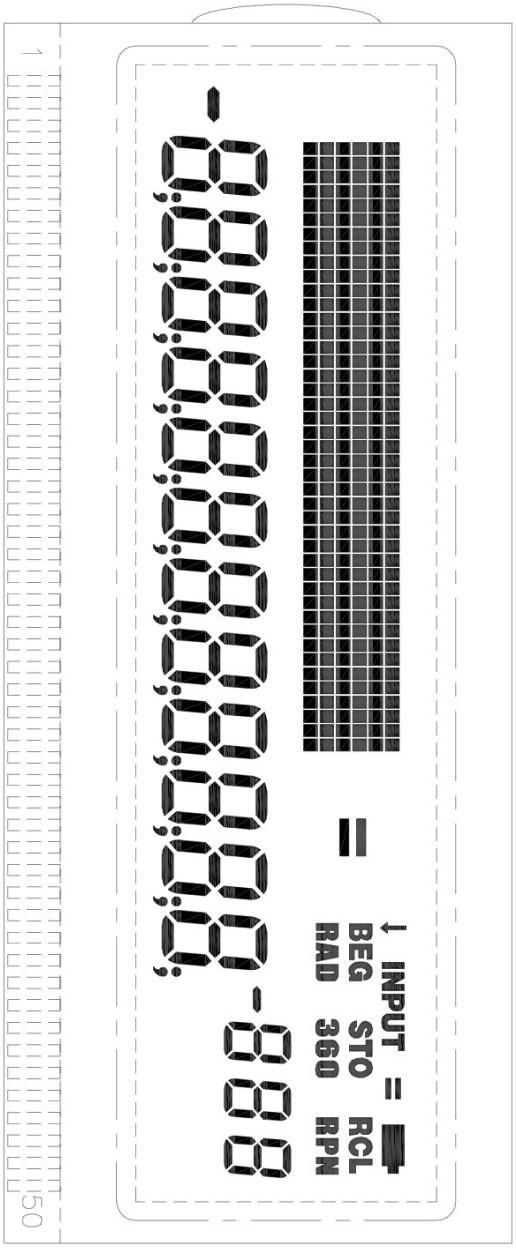

Figure 3: LCD Display of the HP20b calculator

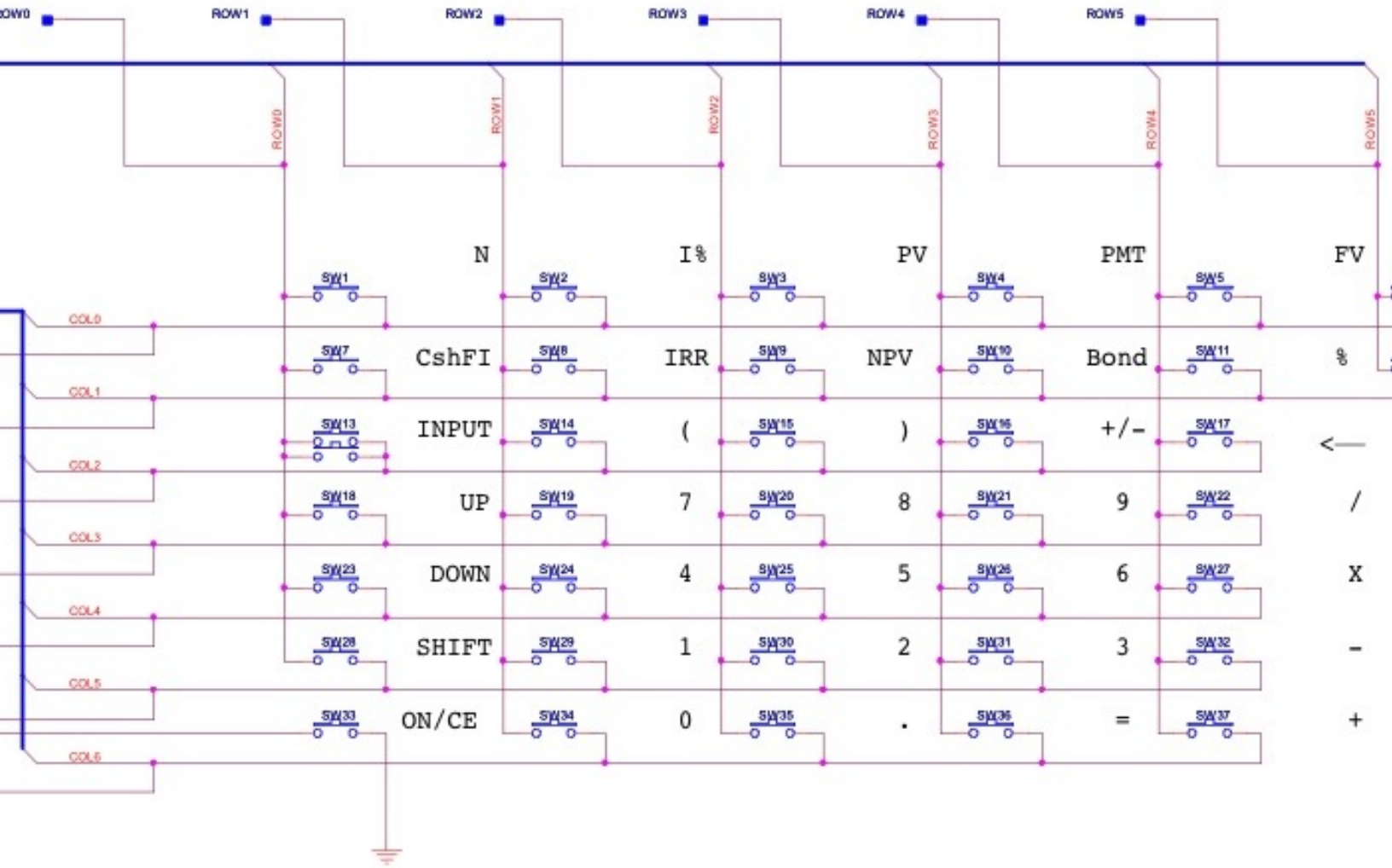

Figure 4: A schematic diagram of the HP20b keyboard

calculator to evaluate the inputs as expressions in a RPN calculator and display the results on screen.

6 Software Details

6.1 Lab 1: A Scrolling Display

The objective of the first lab was to display a scrolling message. Specifically, the objective was to create a function that takes an integer argument and displays it in decimal on the HP20b LCD display. Figure 5 shows one solution to this problem, a function called lcdprint. Firstly, this function checks to see if the desired output value is 0 because this is a case that requires less calculation and is easy to handle apart from the default cases. Then, the function checks if the input is negative. The variable negative is set to true or false accordingly, and if the value is negative, it is made positive to facilitate calculations. Then, a while loop is entered in which the value to be displayed is divided mod 10 until it reaches zero, with each remainder being placed at the next available column on the display (value of post-decremented count). The last job of the function is to check place the character - at the front end of the displayed number if the input value was negative.

6.2 Lab 2: Scanning the Keyboard

For lab 2, the objective was to write software that can read the keyboard of the HP20b and display which key is being pressed. To do so, it was instrumental to use the functions given that set the columns voltages to low, and the function that read the rows to determine whether or not a key was being pressed in a nested for loop, iterating through each column, and within each column, iterating through each row. If no key was pressed, we returned the number of rows as a default case. We also too the column number as an integer, and used a global variable returnColumn to encode the column number as we did not yet know how to pass two variables at the same time. We accessed this variable through an accessor function named getColumn. In main, we encoded the result by noting that numbers are arranged in an orderly matrix, so the column number sets an upper and lower limit on what the values of the keys within the column can be, and exploited that as an alterative to individually defining each row and column combination as a number. We tested this software by pressing numbers and seeing what the calculator displayed on the screen, and thankfully, it worked! Figure 6 shows the code that was used.

6.3 Lab 3: Entering and Displaying Numbers

For lab 3, the objective was to write code that would allow the user to input both a number and an operation, and then display then both on the LCD screen. Fur-

```
#include "AT91SAM7L128.h"
#include "lcd.h"
#define NUM_COLUMNS 11

void lcdprint(int input){
  if(input == 0){
    lcd_put_char7('0', NUM_COLUMNS);
    return;
  }
  int remainder, count = NUM_COLUMNS;
  int negative = input < 0 ? 1 : 0;
  int output = negative ? -1*input : input;
  while(output>0){
    remainder = output%10;
    lcd_put_char7('remainder',count--);
    output = output/10;
  }
  if(negative) lcd_put_char7(45, count); //print minus sign
}

int main()
{
  lcd_init();
  lcdprint(4384);
  return 0;
}
```

Figure 5: My solution for lab 1: the scrolling message

```
#include "AT91SAM7L128.h"
#include "lcd.h"
#include "keyboard.h"
#define KEYBOARD_COLUMNS 0x7f
#define KEYBOARD_ROWS 0x400fc00
#define NUM_COLUMNS 7
#define NUM_ROWS 6

int main()
{
  // Disable the watchdog timer
  *AT91C_WDTC_WDMR = AT91C_WDTC_WDDIS;

  lcd_init();
  keyboard_init();

  lcd_print7("SEE");

  keyboard_column_low(0);


  int row, column;
  for (;;) {
    row = keyboard_key();
    column = getColumn();
    switch(column){
      case 3:
        lcd_put_char7('0'+6+row,5);
        break;
      case 4:
        lcd_put_char7('0'+3+row,5);
        break;
      case 5:
        lcd_put_char7('0'+row,5);
        break;
      default:
        lcd_put_char7('E',5);
        break;
    }
  }
  return 0;
}

const unsigned char keyboard_row_index[] = {11,12,13,14,15,26};
```

thermore, lab 3 was the first time we had to use pointers to structs. Our code, which uses Professor Edwards version of keyboardkey, assigns an integer key, and an integer num. It then waits for a key to be pressed, and if that key, encoded within key, is a number as defined by ASCII, it appends that number to the variable num, after subtracting the ASCII value of zero, to encode it as a pure integer, unless num goes beyond the calculators limit of digits available for display. This continues indefinitely until the user enters an operation. Operations are defined by either characters or combinations of characters, and we used the standard definitions given in the keyboardkey matrix. When the user enters an operation, the pointer is directed to num for the structs number, and key for its operation, and the code closes by printing the num variable. See Figure 7 for more details. If the user enters an operation without entering a number, INTMAX, the largest unsigned integer the processor can handle is passed as an error check.

6.4 Lab 4: An RPN Calculator

The objective of lab 4 was simple: to build upon the functions created in the previous three labs to build an RPN calculator. To implement a post-fix calculator, a stack must be used. A stack is an abstract data type and a last in, first out data structure. The way a stack works is very similar to a stack of dishes, the last dish to be placed on top of the stack is the first one to come off. The stack implemented in this version of the repurposed HP20b is an array of size 5, and the static memory allocation property of an array explains why only 5 terms can be calculated at once. Figure 8 shows the code that implements the RPN calculator. An infinite for loop is used to keep the calculator accepting input and returning output for as long as necessary. The first thing the algorithm does after instantiation/initialization of variables is call the keyboardgetentry function, described in 6.3, then check for underflow/overflow or clear conditions (count value less than 0 or greater than 4 or = button pressed). If these conditions are met, a for loop is used to reset all values in the stack to 0 and reset count to 0. Otherwise, if the input key is the operation passed by the get entry function, the accompanying numeric value is pushed on to the stack. Otherwise, one value is popped from the stack, and if both number and operation are given, the operation is applied and the return value is displayed. If only an operation was given, INTMAX will be passed by keyboardgetentry, and a second value will be popped from the stack and the operation applied instead to these two values. Lastly, the result is pushed back onto the stack.

7 Lessons Learned

We learned many things from this course, not the least of which was the basics of C. This is especially important because C is in certain aspects much more low

```
#include "AT91SAM7L128.h"
#include "lcd.h"
#include "keyboard.h"
#define NUM_COLUMNS 7
#define NUM_ROWS 6
#define KEYBOARD_COLUMNS 0x7f
#define KEYBOARD_ROWS 0x400fc00


int main()
{
  struct entry entry;
  // Disable the watchdog timer
  *AT91C_WDTC_WDMR = AT91C_WDTC_WDDIS;

  lcd_init();
  keyboard_init();

  lcd_print7("PRESS");

  keyboard_get_entry(&entry);

  lcd_put_char7(entry.operation,4);

  for (;;) {
    int c = keyboard_key();
    lcd_put_char7(c > 0 ? c : '␣', 3);
  }

  return 0;
}

const unsigned char keyboard_row_index[] = {11,12,13,14,15,26};

/* Character codes returned by keyboard_key */

const char keyboard_keys[NUM_COLUMNS][NUM_ROWS] = {
  {'N',  'I', 'P', 'M', 'F', 'A'},
  {'C',  'R', 'V', 'B', '%', 'L'},
  {'\r', '(', ')', '~', '\b', 0},
  {'\v', '7', '8', '9', '/',  0},
  {'\n', '4', '5', '6', '*',  0},
  {'S',  '1', '2', '3', '-',  0},
  { 0,   '0', '.', '=', '+',  0}};
```

```
#define STACK_SIZE 5
#include "AT91SAM7L128.h"
#include "lcd.h"
#include "keyboard.h"

int main()
{
  int i;
  struct entry entry;
  // Disable the watchdog timer
  *AT91C_WDTC_WDMR = AT91C_WDTC_WDDIS;

  lcd_init();
  keyboard_init(); //Initialize keyboard

  int stack[STACK_SIZE];
  int current=0;
  int popped, popped2, result;
  for(;;){
    keyboard_get_entry(&entry);
    if(entry.operation == '=' || current > 4 || current < 0){ //clear
      current = 0;
      for(i = 0; i < STACK_SIZE; i++) stack[i] = 0;
      popped = 0;
      popped2 = 0;
      result = 0;
      if(current > STACK_SIZE-1) lcd_print7("OVERFLOW"); //Handle over
      else{
        if(current < 0) lcd_print7("UNDEFLOW"); //Handle underflow
        else lcd_print_int(popped); //Print 0 in case of clear
      }
    }
    else if(entry.operation == '\r') stack[current++]=entry.number;
    else{
      popped = stack[--current];
      if(entry.number == INT_MAX) popped2=stack[--current]; //no numbe
      else popped2=entry.number;
      if(entry.operation == '+') result=popped+popped2;
      if(entry.operation == '-') result=popped-popped2;
      if(entry.operation == '*') result=popped*popped2;
      if(entry.operation == '/') result=popped/popped2;
      stack[current++]=result;
      lcd_print_int(result);
    }
  }
```

level focused than its other beginner programming language counterparts such as Java, Python or Matlab. Also, we learned about the fundamentals of systems programming in a restricted environment where its not possible to call upon libraries or APIs to solve tricky problems. Finally, we learned better communication skills, both between ourselves, and in our presentations to the class.

8 Criticism of the Course

This course was for the most part very effective at teaching systems programming. One area that could have been improved was the instruction of C. We feel that if that topic had been presented earlier in the course, it would have helped make the labs, and the code surrounding them, more comprehensible. As it was, both members of the team had some prior programming experience, but for a team completely devoid of any such experience, which is a probable situation, some of the labs must have been extremely challenging. Other than that though, this course was educational, rewarding, and fun.

References

[1] Hp-20b repurposing project. Online `http://www.wiki4hp.com/doku.php?id=20b:repurposing_project`.