

# How It's Made

## Making a RPN Calculator

Daniel Estrada Taylor

Dev Harrington

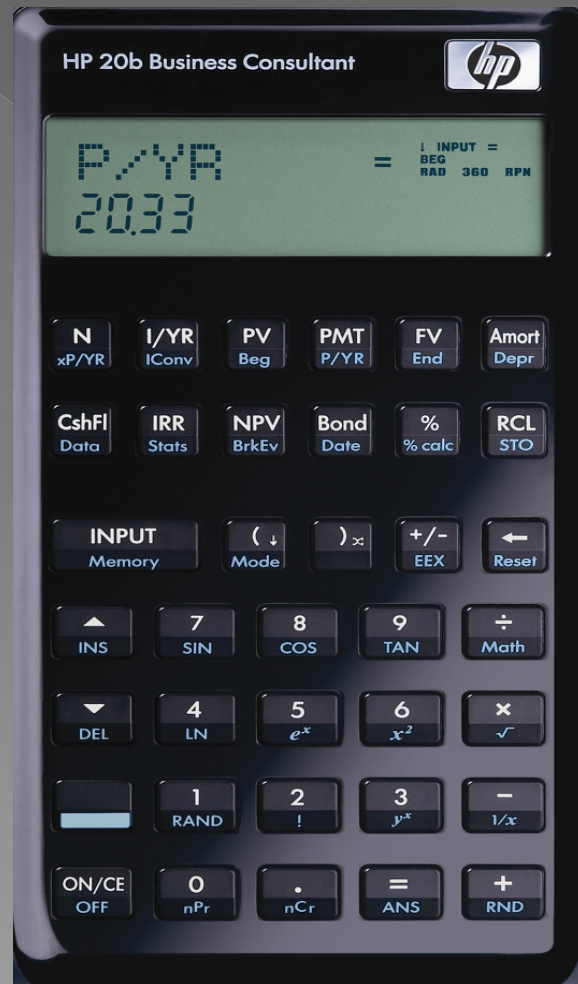
Sekou Harris

# Building A Calculator

In this presentation we will go over exactly what it took for us to attempt to build a calculator from the ground up.

- ◉ Our calculator's user guide
- ◉ The hardware platform
- ◉ The software we implemented

# USER GUIDE



# What to Press



- All numbers
- Basic operators
- Power
- INPUT
- Backspace



# How to Press

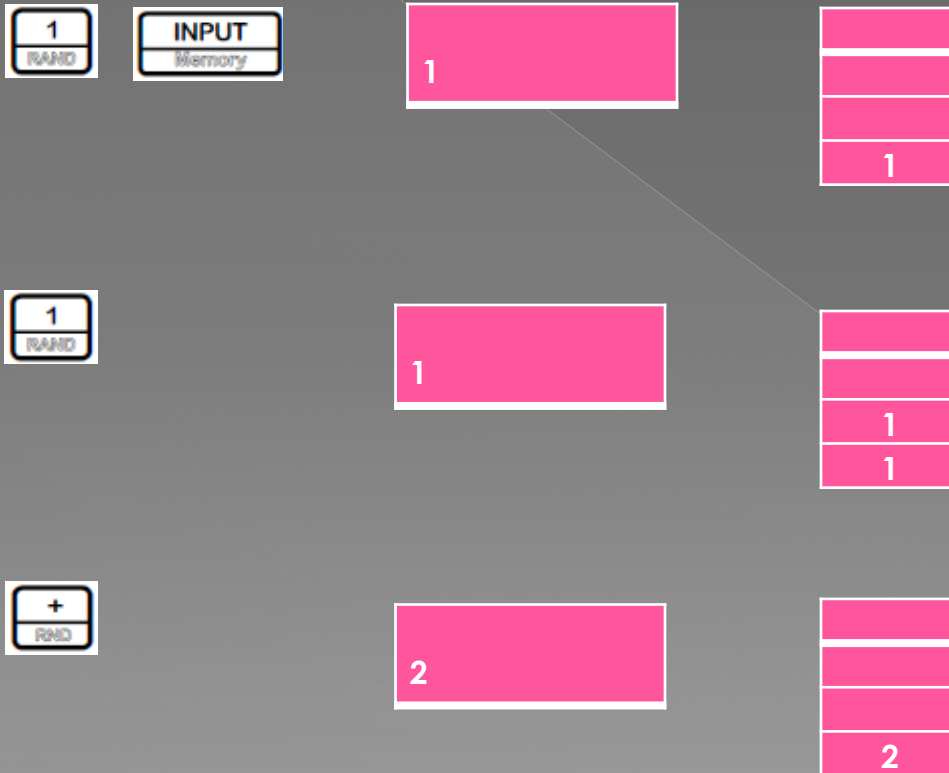
Reverse Polish notation (RPN):

Operators follow operands

- $3+5 \rightarrow 3\ 5\ +$
- $10*8 \rightarrow 10\ 8\ *$

# How RPN is Performed On the Calculator

1 + 1



$$(15/5)^*(7-1)$$

1  
RAND

5  
 $e^x$

INPUT  
Memory

15

15

5  
 $e^x$

5

15  
5

$\div$   
Memory

3

3

$(15/5) * (7-1)$  cont'd

7  
BIN

INPUT  
Memory

7

3  
7

1  
RAND

1

3  
7  
1

-  
1/x

6

3  
3  
6

x  
√

18

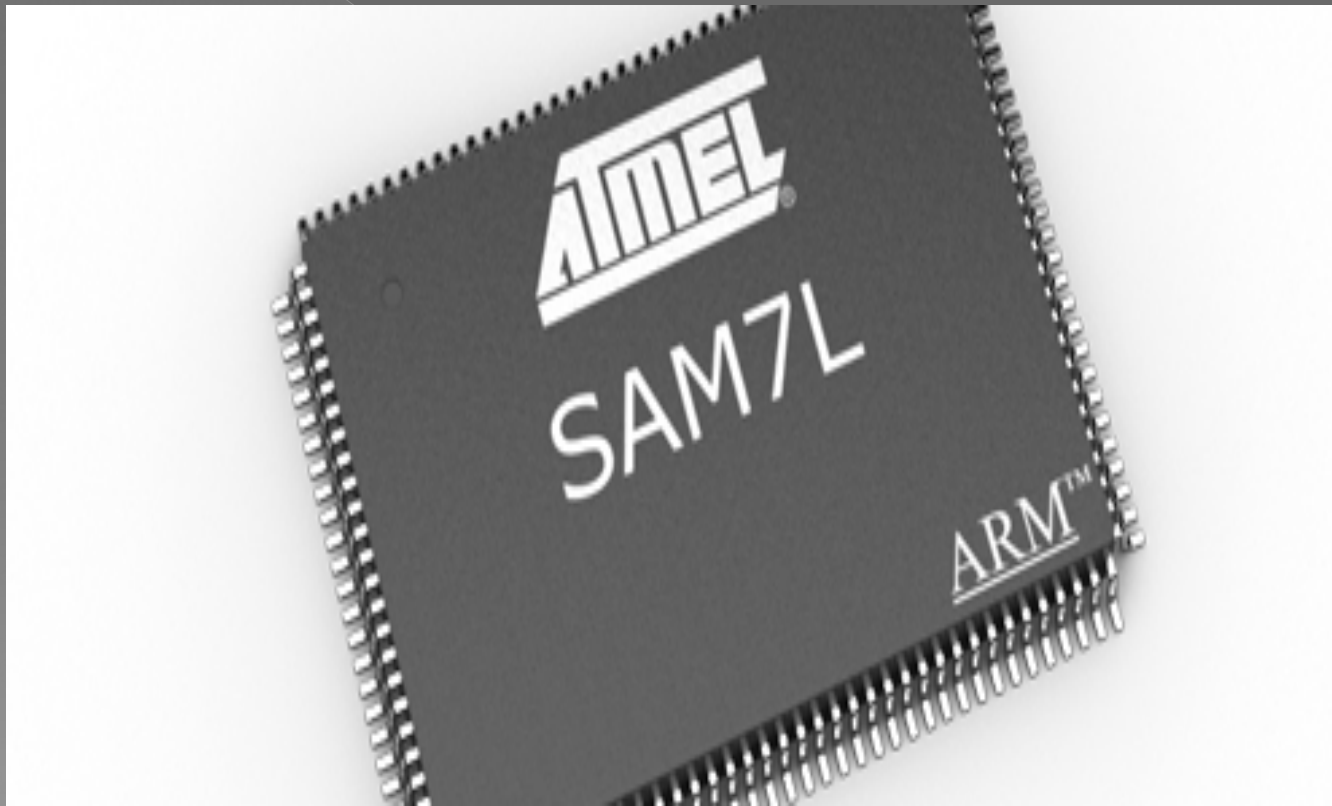
3  
3  
18

# The Platform

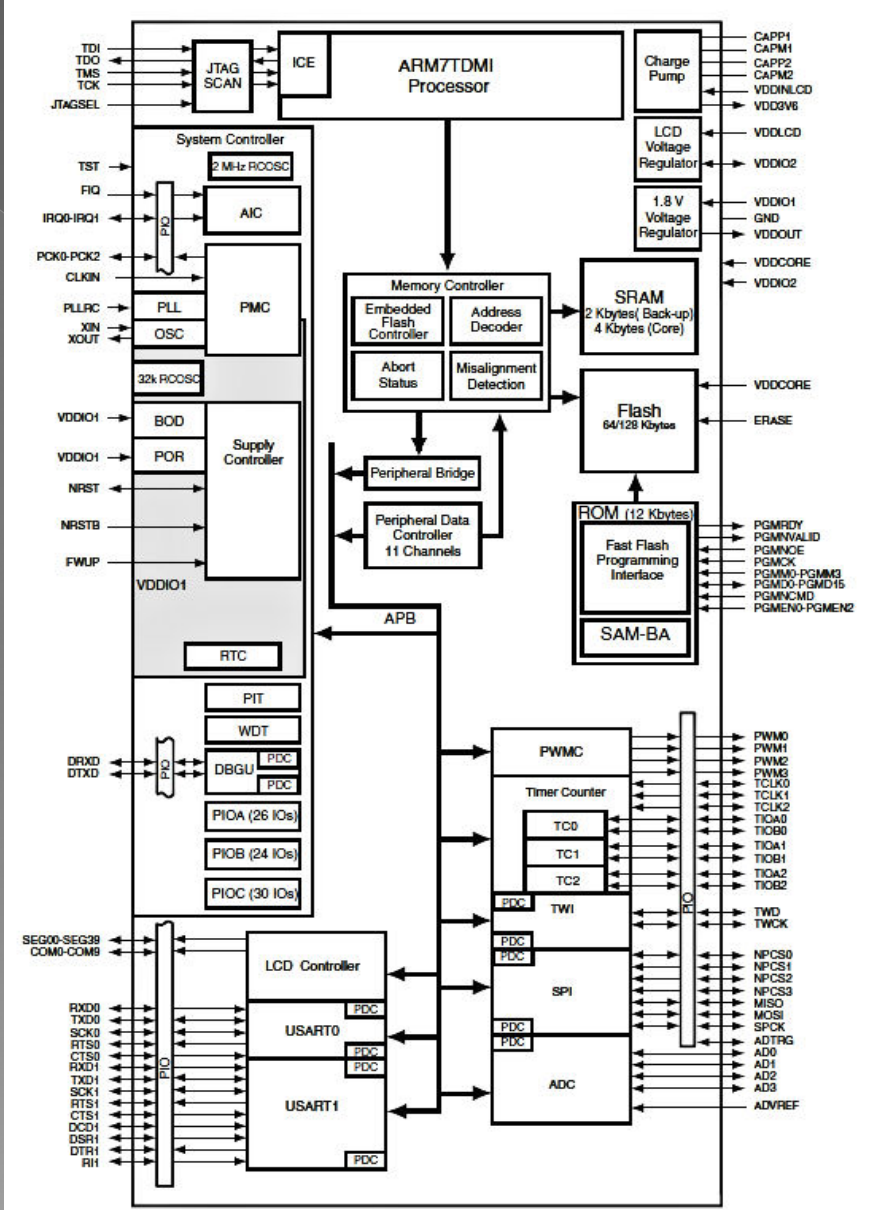




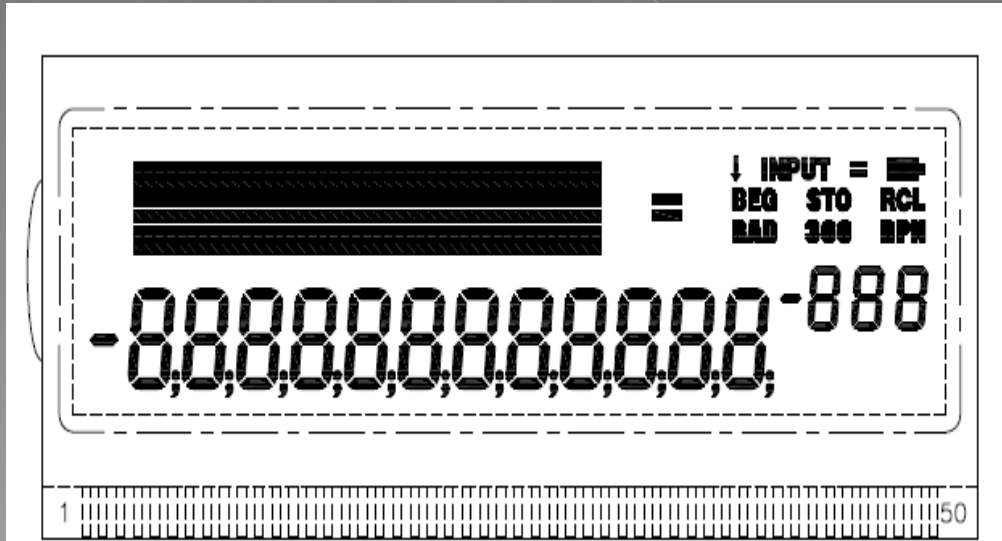
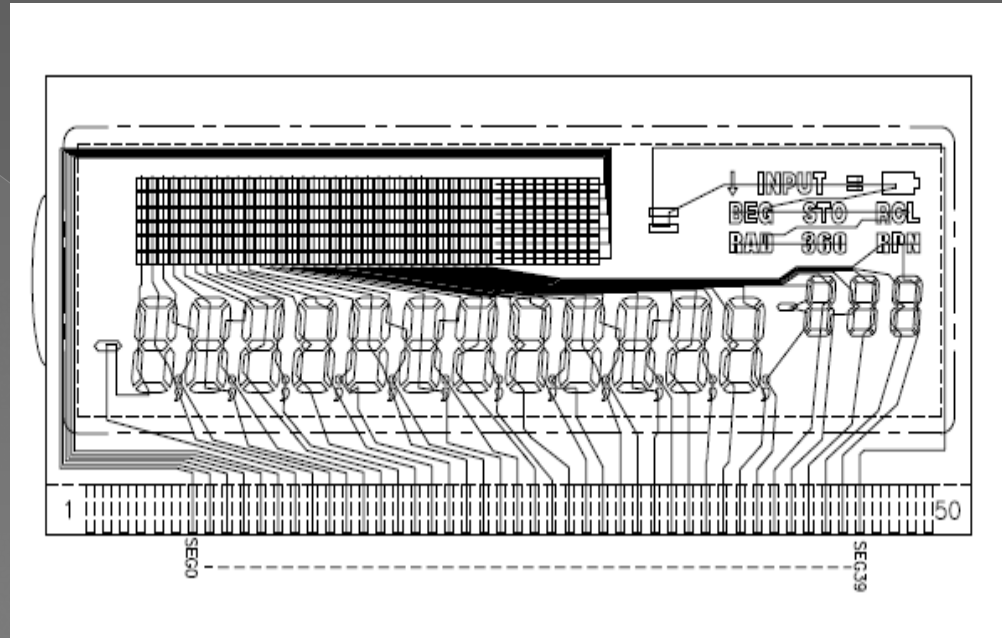
# AT91SAM7L128 Processor



# SAM7L



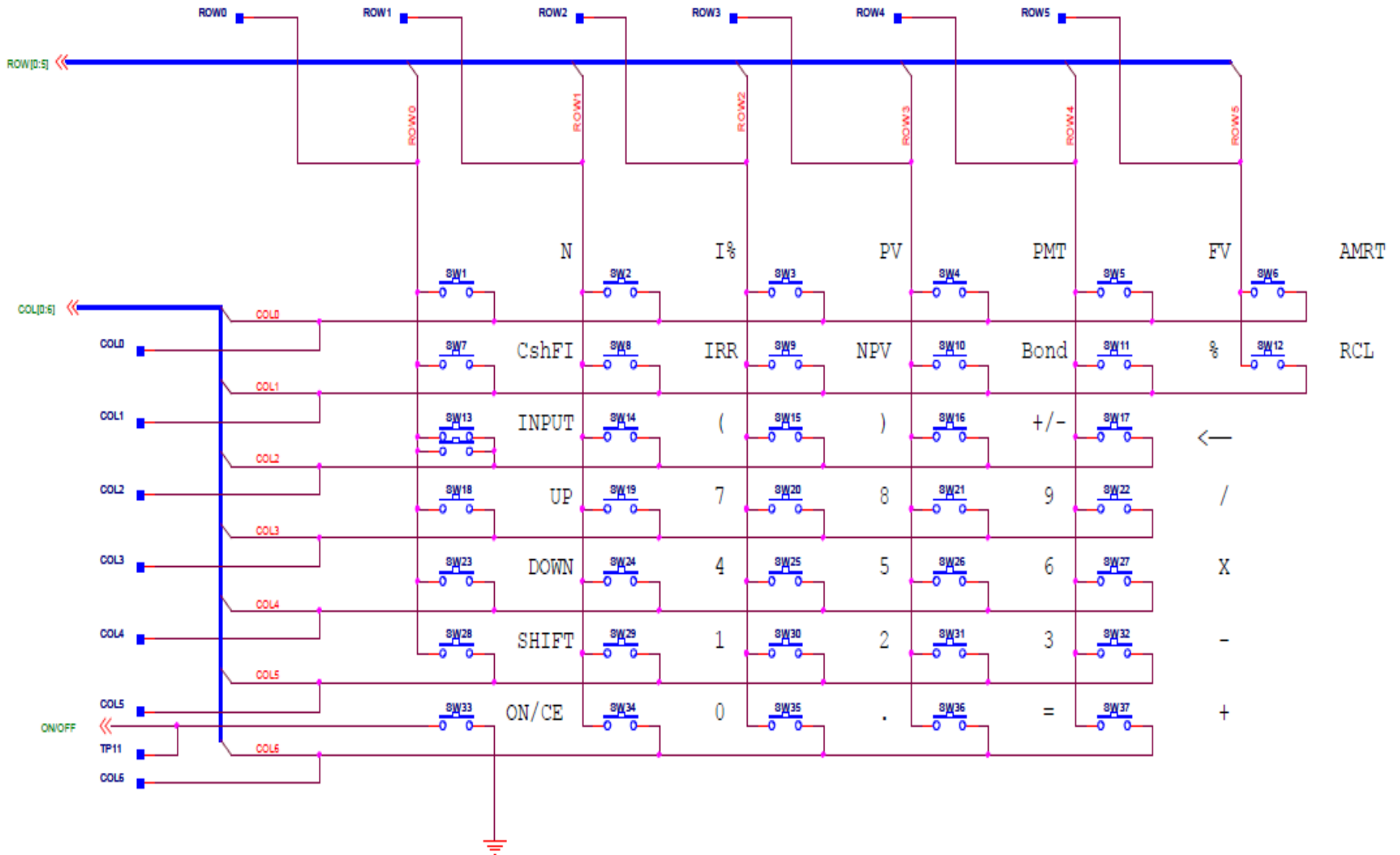
# LIQUID CRYSTAL DISPLAY



# LCD LIBRARY FUNCTIONS

- `lcd_init()`
- `lcd_put_char7(char ch, int col)`
- `lcd_print7(const char *c)`
- `lcd_print_int_neg(int negative, unsigned int n)`
- `lcd_print_int(int n)`

# BOARD OF KEYS





# KEYBOARD LIBRARY FUNCTIONS

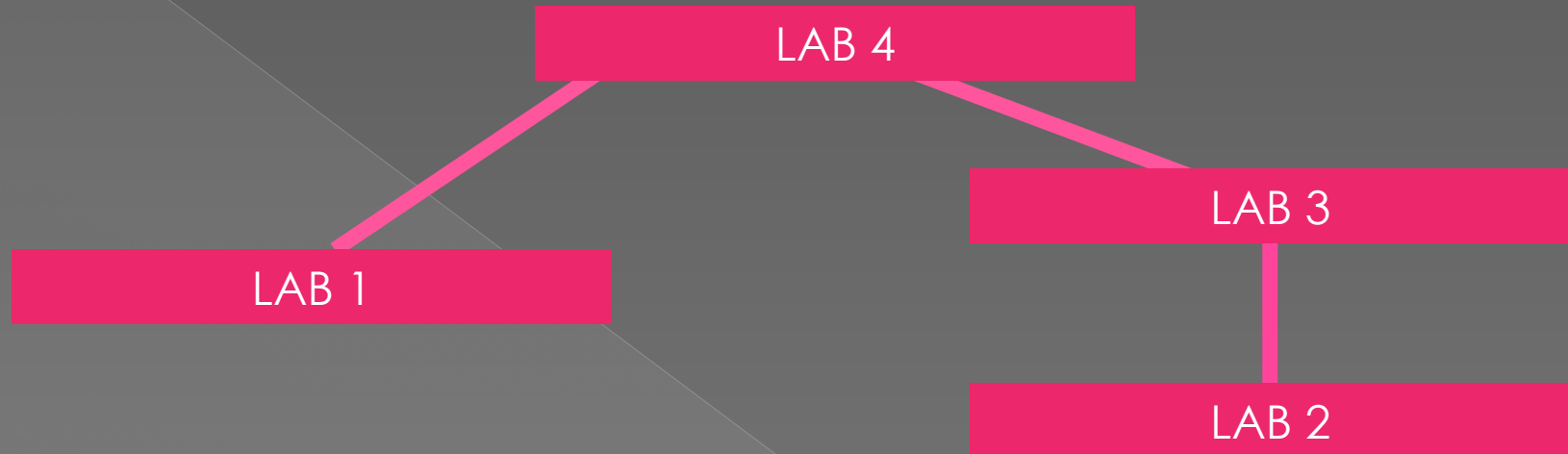
- ◉ `Keyboard_init()`
- ◉ `Keyboard_column_high(int column)`
- ◉ `Keyboard_column_low(int column)`
- ◉ `Keyboard_row_read(int row)`

# SOFTWARE

## Software Jokes



# ARCHITECTURE



1. Lab 4 first makes use of lab 3 so that the user may enter numbers and operations
2. Lab 3 then makes use of lab 2 in order to go through the key board and see what is actually being pressed
3. Lab 4 then gets the results from lab 3 and decodes the results in order to see what to with the number
4. Finally lab 4 makes use of lab 1 in order to output the result

# LAB 1 DETAILS

```
#include "AT91SAM7L128.h"
#include "lcd.h"
#define SCREEN 12
#define INT_MAX 2147483647
void clear() //clears the screen
{
    int i=0;
    while(i < SCREEN) {
        lcd_put_char7(' ',i);
        i++;
    }
}
int main()
{
    lcd_init();

    int num = 2147483646; //integer to be outputted
    int newnum = num;
    int i=0;
    clear();
    if(num == 0)
        lcd_put_char7('0',0); //displays '0' if the integer is 0
    else if (num >= INT_MAX || num*-1 >= INT_MAX) {
        lcd_put_char7('E',0); //displays error message
        lcd_put_char7('R',1);
        lcd_put_char7('R',2);
        lcd_put_char7('o',3);
        lcd_put_char7('R',4);
    }
    else {
        if(num < 0) { //displays negative sign, increments counter
            num = abs(num);
            newnum = abs(newnum);
            lcd_put_char7('-',0);
            i++;
        }
        while(newnum > 0) { //counts number of digits in integer
            newnum = newnum/10;
            i++;
        }
        while(num > 0) { //displays the integer
            char thing = num%10 + '0';
            lcd_put_char7(thing,i-1);
            num=num/10;
            i--;
        }
    }
    return 0;
}
```

# LAB 2 DETAILS

```
#include "AT91SAM7L128.h"
#include "lcd.h"
#include "keyboard.h"

#define KEYBOARD_COLUMNS 0x7f
#define KEYBOARD_ROWS 0x400fc00
#define NOTHING -1
#define COLUMNS 7
#define ROWS 6
#define TENS 10

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

void keyboard_init();
void keyboard_column_high(int column);
void keyboard_column_low(int column);
int keyboard_row_read(int row);

int keyboard_key()
{
    int i, j;

    for (i = 0; i < COLUMNS; i++) //cycles through columns
    {
        keyboard_column_low(i); //sets each test column low
        for(j = 0; j < ROWS; j++) //cycles through rows
        > if(!keyboard_row_read(j)) //checks each row for pressed //button
        {
            keyboard_column_high(i);
            return (i*TENS)+j; //returns an integer with the
            //column as the tens value
            //and row as the ones value
        }

        keyboard_column_high(i); //resets test column to high
    }
    return NOTHING; //returns constant if nothing
}

//is pressed
}

char what_pressed(int col,int row) //takes in the tens place and
//remainder of key_pressed
{
    char layout[COLUMNS][ROWS]={ //layout of the calculator as array
        {'A','B','C','D','E','F'},
        {'G','H','I','J','K','L'},
        {'M','N','O','P','Q','R'},
        {'R','7','8','9','/','*'},
        {'S','4','5','6','*','/'},
        {'T','1','2','3','+','-'},
        {'Z','0','','=' ,'+'};

    return layout[col][row]; //returns character for key pressed
}
}
```



# LAB 3 DETAILS

```
void keyboard_get_entry(struct entry *result)
{
    int counter = -1;
    int num = INT_MAX;
    int pressed = -1;
    int isnegative = 0;

    for(;;)
    {
        while(keyboard_key() == -1);
        pressed = keyboard_key();
        while(keyboard_key() != -1);

        counter++;

        if(counter>11)
            return;

        else if(pressed >= '0' && pressed <= '9')
        {
            lcd_put_char7(pressed,counter);
            if( (counter == 0) || (isnegative == 1) )
                num=0;
            num = num*10 + (pressed-'0');
        }

        else if((pressed == '-') || (pressed == '+') || (pressed == '*') || (pressed == '/') || (pressed == '\r') || (pressed == '='))
        {
            lcd_put_char7(pressed,counter);
            if (isnegative == 1) {
                result->number = num * -1;
                result->operation = pressed;
                return;
            }
            if (counter == 0 && pressed == '-') {
                isnegative = 1;
            }
            else {
                result->number = num;
                result->operation = pressed;
                return;
            }
        }

        else if(pressed == '\b')
        {
            lcd_put_char7(' ',counter-1);
            counter = counter - 2;
            num = num/10;
        }
    }
}
```

# LAB 4 DETAILS



WORK IN  
PROGRESS

Plan

1. Use `keyboard_entry`
2. Have a limit within our while statement that allows for three or four number to be saved within our “stack”
3. As the numbers and operations are entered we will perform those Operations
4. We will re-write a function similar to that in lab 1 which will output the final sum of the users inputs.
5. Also like lab 1 we will have to have tests for “special cases”

Overall

Will mainly just be connecting the pieces of the past three labs and it will most definitely be completed before the deadline.





**FINALE**