# The Project:
# <u>RPN Calculator</u>

Shensi Ding and Joshua Boggs

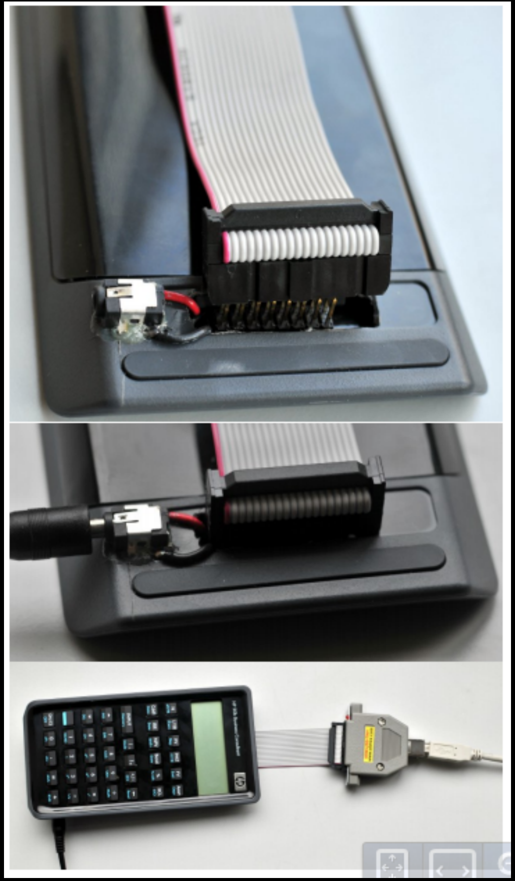
## The Introduction

Over the course of the past semester we have rewritten the firmware for the calculator.

The contents of this document include a user guide, social implications, hardware and software architectures, details of the software, our lessons learned, and criticisms of the course.

# HP 20b Business Calculator

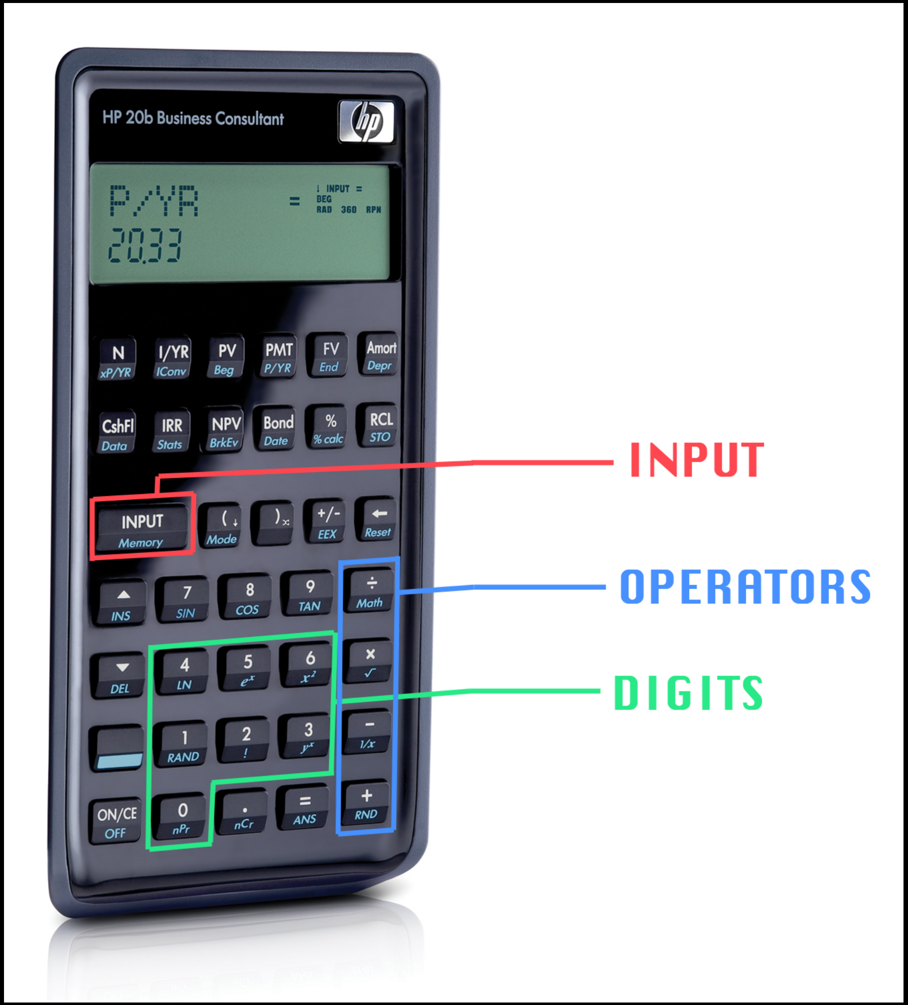
The calculator consists mainly of a LCD connected to an Atmel at91sam7l128 processor. The calculator has been modified to connect to a computer through a JTAG port.



## The User Guide

The calculator we have programmed uses reverse Polish notation. In this notation, the operators come after the operands. No parentheses are used, simply number keys, operators, and the input key.

# User Guide (part 2)



To carry out a simple operation, begin by typing your first operand into the keyboard. Then, press INPUT. The number you entered will appear on the LCD. Next, type out the second operand, and again press INPUT. Finally, press the operator you would like to use. The answer will then appear on the LCD.

## Example 1: A simple operation

| Let's walk through the calculation 9 + 12 | Stack |
|---|---|
| ● Press the 9 digit key | |
| ● Press input to save the first operand | [9] |
| ● Press 1, the first digit of 12 | |
| ● Press 2, the second digit of 12 | |
| ● Press input to save the second operand | [9, 12] |
| ● Press the addition opera + | [21] |

If all is entered correctly, the number 21 will appear on the screen.

# Example 2: A more complex operation

Let's walk through the calculation (3+5)x(7-2)

| | Stack |
|---|---|
| ● Press the 3 digit key | |
| ● Press input to save the first operand | [3] |
| ● Press the 5 digit key | |
| ● Press input to save the second operand | [3, 5] |
| ● Press the addition operator + to add the first two operands | [8] |
| ● Press the 7 digit key | |
| ● Press input to save the third operand | [8, 7] |
| ● Press the 2 digit key | |
| ● Press input to save the fourth operand | [8, 7, 2] |
| ● Press the subtraction operator - to subtract the third and fourth operands | [8, 5] |
| ● Press the multiplication operator x to multiply the resultant operands in the stack | [40] |

If all is entered correctly, the number 40 will appear on the screen.

## The Social Implications

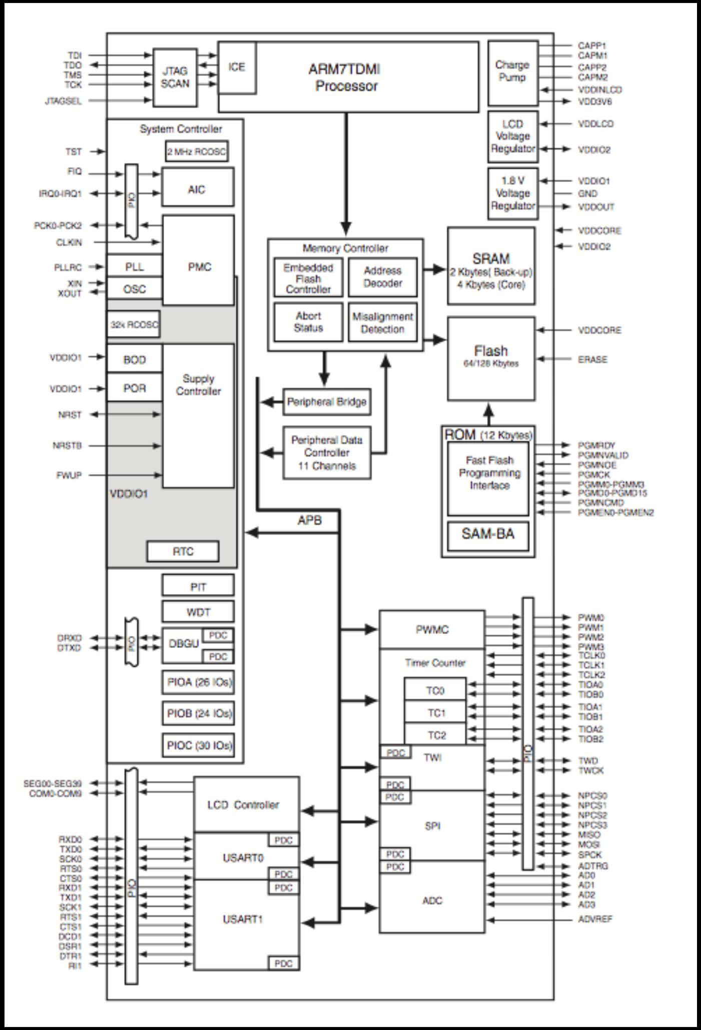
- Efficient handheld calculations
- Simplifies problems and tasks
- Organizes and tracks calculations, making error-prone hand calculations more obsolete

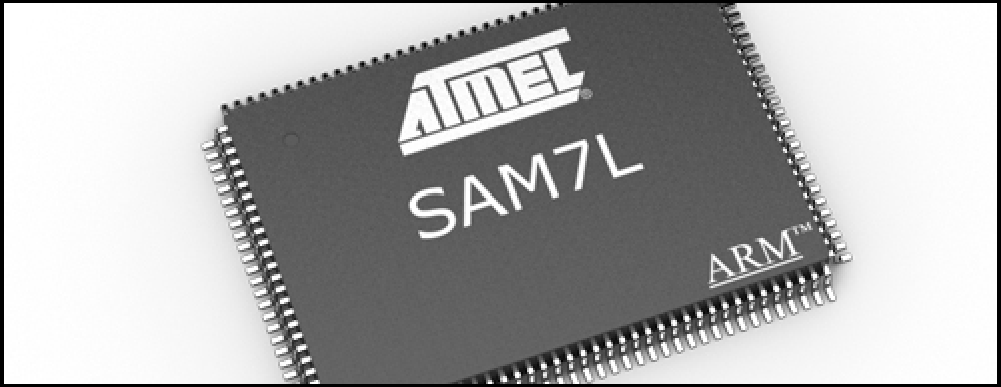
Improvements in:

- Education
- Business
- Scientific research
- Overall study of mathematics

# The Platform - Processor

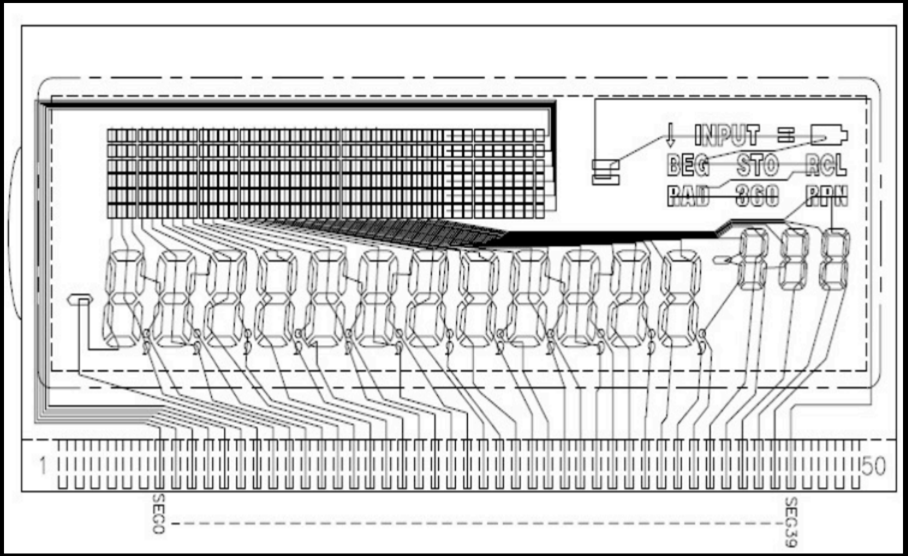
The calculator utilizes an Atmel AT91SAM7L128 processor. To the right is a block diagram, and below is a picture of the processor.

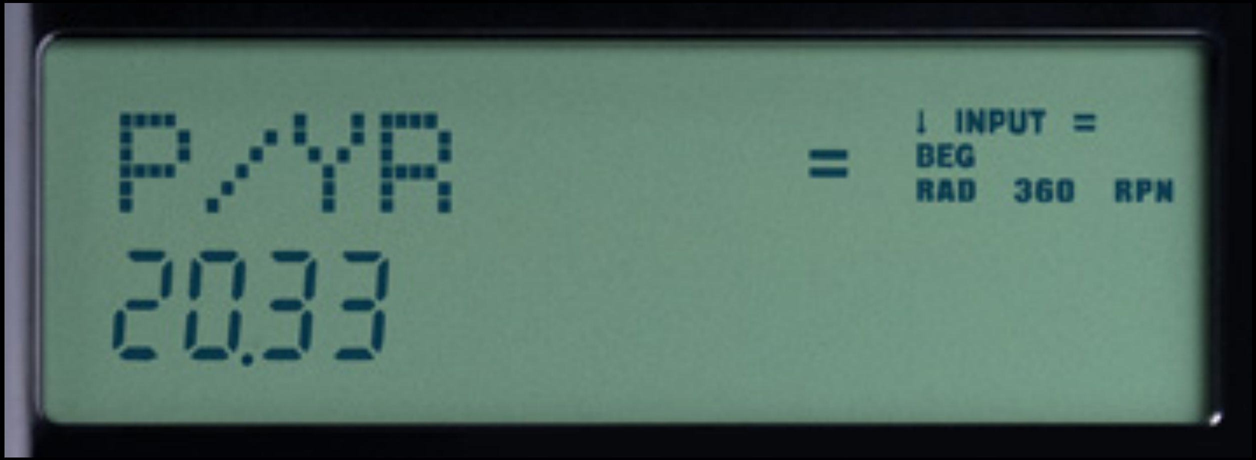


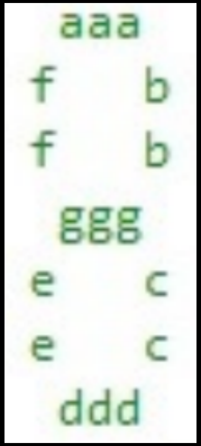


# The Platform - LCD

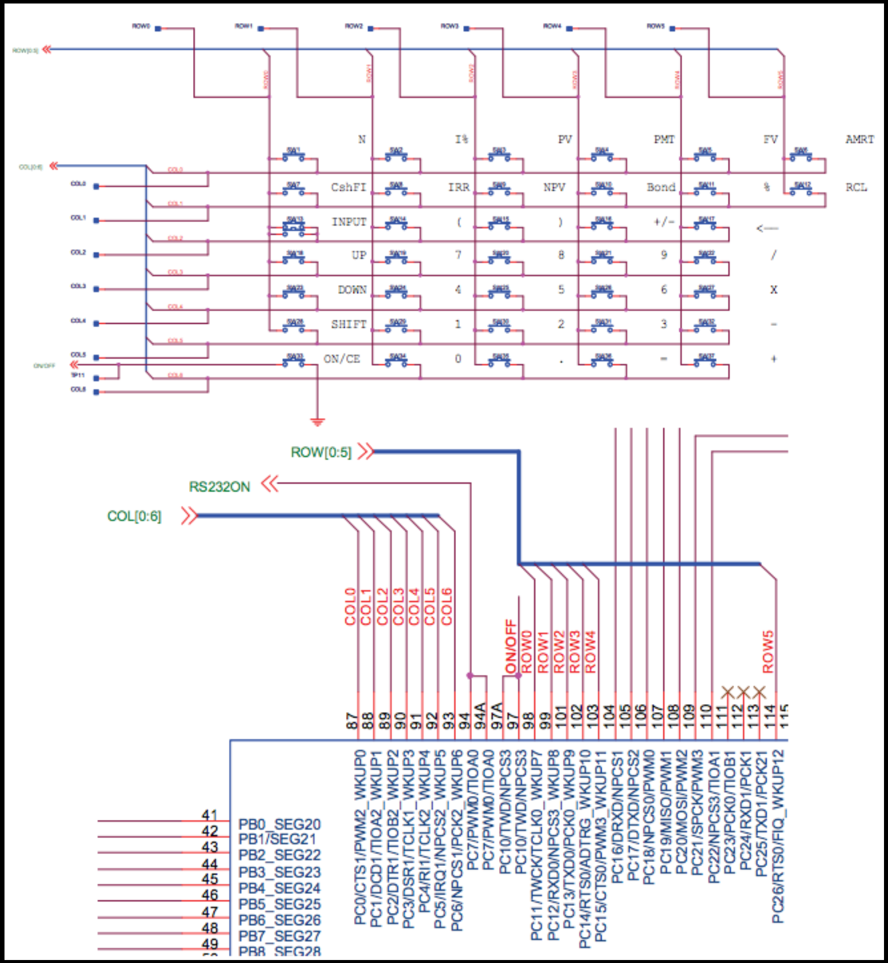
The calculator has a large 2-line display which can show up to 12 numbers at once.



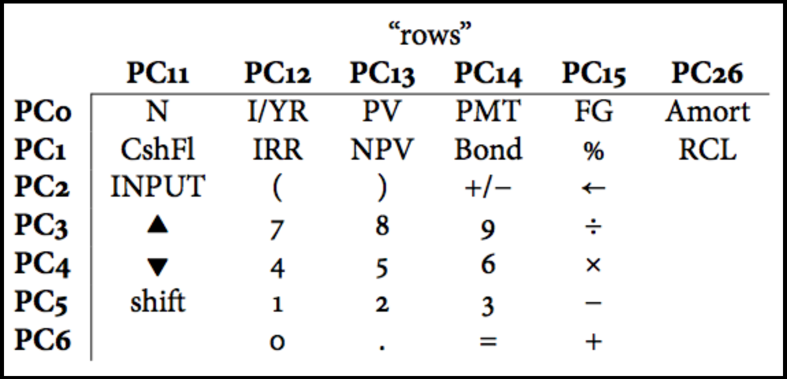




# The Platform - Keyboard



When a key is pressed, one pin is shorted for the column, and another pin is shorted for the row. This is how we are able to read which key is pressed.



| | "rows" | | | | | |
|---|---|---|---|---|---|---|
| | PC11 | PC12 | PC13 | PC14 | PC15 | PC26 |
| PC0 | N | I/YR | PV | PMT | FG | Amort |
| PC1 | CshFl | IRR | NPV | Bond | % | RCL |
| PC2 | INPUT | ( | ) | +/− | ← | |
| PC3 | ▲ | 7 | 8 | 9 | ÷ | |
| PC4 | ▼ | 4 | 5 | 6 | × | |
| PC5 | shift | 1 | 2 | 3 | − | |
| PC6 | | 0 | . | = | + | |

## The Software Architecture

**void lcd_print_int_neg(int n)**

- function prints out the desired number

**int keyboard_key()**

- returns the pressed key

**main.c**

- allows the display for the correct character for keyboard_key()
- holds and organizes the stack for calculations

## <u>The Software Details</u>

## Lab 1 - Goal

The process began with writing the software to display a desired number on the calculator's screen.

# Lab 1 - Code

**In hello.c - int main()**

```
int main()
{
        lcd_init();

        void lcd_print_int_neg(int n)
        {
                // checks to see if the number is negative
                int isNegative = 0;
                if( n < 0 )
                {
                        isNegative = 1;
                        n = n*(-1);
                        // make the negative number positive
                }
// counter keeps track of the column
int columnCounter = 0;
// while then number doesn't equal 0, we keep
// printing out the next digit in a new column
if( n != 0 ) {
        while( n!= 0 && columnCounter < 11 ) {
                int d = n%10;
                lcd_put_char7(48+d, 11-columnCounter);
                n = n/10;
                columnCounter++;
        }
        // if the number was negative, we then add a negative
        // sign in front of the number's absolute value
        if( isNegative == 1 )
                lcd_put_char7('-', 11-columnCounter);
}
else // for if the number equals 0
        lcd_put_char7( 48 , 11 );
}
}
```

## Lab 2 - Goal

We then wrote a code which would respond to a pressed key by displaying the key on the calculator screen.

## Lab 2 - Code

### <u>In keyboard.c - int keyboard_key()</u>

```
int keyboard_key()
{
        keyboard_init(); // sets all columns high
        int row;
        for(row = 0; row <6; row++)
        {
                int col;
                for(col = 0; col<7; col++)
                {
                        keyboard_column_low(col);
                        // goes through each column and sets it low
                        if(!keyboard_row_read(row) )
                        // reads the column, and if row is high
                                return row * 10 + col;
                        keyboard_column_high(col);
                }
        }
        return -1;
}
```

### <u>In main.c - int main</u>

```
char calculator[7][6] = { {'N', 'I', 'P', 'M', 'F', 'A'}, {'C', 'R',
'V', 'B', '%', 'L'}, {'T', '(', ')', '/', '<', ' '}, {'U', '7', '8', '9', '/', ' '},
{'D', '4', '5', '6', '*', ' '}, {'S', '1', '2', '3', '-', ' '}, {' ', '0', '.', '=',
'+', ' '}};

for(;;) // makes the for loop run forever
{
   int key = keyboard_key(); // receives the pressed key
        if(key != -1)
        {
                row = key/10;
                col = key%10;
        char chosen = calculator[col][row];
        // goes through the array to find the pressed key
        lcd_put_char7(chosen, 4);
         // displays the pressed key
        }
 }
```

## Lab 3 - Goal

Next, we worked on a code which would allow users to enter and edit numbers.

# Lab 3 - Code

**<u>In keyboard.c - int keyboard_key()</u>**

```
void keyboard_get_entry(struct entry *result)
{
  // set number to MAX_INT for when the user enters an
operation key without entering a number
  int number = INT_MAX;
  int pressedKey;
  int negative = 0;

  // runs
  for(;;)
  {
    pressedKey = keyboard_key();

    // once a key is pressed, then go through if statement.
    if( pressedKey != -1 )
    {
```

```
switch(pressedKey) {
        // if user enters an operation key, or INPUT
        case '/': case '*': case '+': case '-': case '\r':
        result->operation = pressedKey;
        result->number = negative ? -number : number;
        // if user enters an operation key without entering
        a
        //number
        return;
```

# Lab 3 - Code (continued)

```
// if the user enter a number
case '0': case '1': case '2': case '3': case '4': case '5':
case '6': case '7': case '8': case '9':
// if a number has not been entered already,
// then number = INT_MAX
if( number == INT_MAX )
{
        number = pressedKey - '0';
        lcd_print_int_neg(negative, number);
}
// if a number has been entered already
else
{
        number = number*10 + (pressedKey - '0');
         lcd_print_int_neg(negative, number);
 }
 break;
```

```
// if the user wants to make the number negative
case '~':
negative = !negative;
number = 0;
break;

// if the user wants to erase the last character inputted
case '\b':
number /= 10;
break;
    }
}
// this is in case the user pushes the key for an extended
// period of time so that the if statement doesn't keep
// resetting what number is set to

while( pressedKey != -1)
        pressedKey = keyboard_key();
}
```

## Lab 4 - Goal

Finally, we wrote the code which would allow user to perform calculations using reverse Polish notation.

# Lab 4 - Code

**In main.c - int main() :**

```
lcd_print7("PRESS");
int stack[100]; // create a stack size 100
int pointer = -1;
// runs
for (;;) {
    keyboard_get_entry(&entry);
    lcd_put_char7(entry.operation, 0);
    // prints out the operation key this is in case the user
    // pushes the key for an extended period of time  so that
    // the for loop doesn't  keep resetting what pressedKey
   // is set to
   int pressedKey = keyboard_key();
   while( pressedKey != 0)
           pressedKey = keyboard_key();
```

```
// if the new entry number does not equal the maximum
// number, then we add it to the stack, and move the
// pointer to the left by one place
int newNumber = entry.number;
if(newNumber != INT_MAX)
{
    pointer++;
    stack[pointer] = newNumber;
}
// first check, if the pointer is not at 0, and the operation is
// not input, then "Error" will be printed out.
if(pointer == 0 && entry.operation != '\r')
    lcd_print7("Error");
```

# Lab 4 - Code (continued)

```
// if this is not the case, then move on with the operations if
// the operation is not input
else if(entry.operation != '\r')
{
   pointer--;
   switch(entry.operation){
                  case '+': stack[pointer] = stack[pointer] + stack[pointer+1]; break;
                  case '-': stack[pointer] = stack[pointer] - stack[pointer+1]; break;
                  case '*': stack[pointer] = stack[pointer] * stack[pointer+1]; break;
                  case '/': stack[pointer] = stack[pointer] / stack[pointer+1]; break;
            }
            lcd_print_int(stack[pointer]); // prints out the result
}

}
```

## The Lessons Learned

- how to edit prewritten code to accomplish a given goal
- to fully understand a process before beginning to code
- to be accurate is to be efficient (bugs are not good)
- to consider all possible scenarios while coding

## The Criticisms of the Course

- We would have appreciated an intro to C++ before we began the first lab
- The course assumes that everyone knows computer programming