The Stitch Programming Language

-or-

How I Learned To Stop Worrying and Love P-Threads.
Motivation

Most "modern" programming languages trace their origins back decades to before the advent of cheap, general purpose multicore CPUs. They were designed for a distinctly mono-threaded environment. With Stitch, we aimed to build a language that has the power and flexibility of a fully compiled C style language, while having native threading support for modern multi-threaded applications.
int gcd(int a, int b) {
    while (a != b) {
        if (a > b) {
            a = a - b;
        } else {
            b = b - a;
        }
    }
    return a;
}
Easy(er) Multi-threading

```python
stitch i from 0 to 255 by 1: {
    acc = a[i] * b[i];
}

print(acc)
```
int i = 0;
if (a[i] != b[i]){
    stitch i from 0 to 255 by 1: {
        acc = a[i] * b[i];
    }
    print(acc)
}
Under The Hood

- Wrap everything in pthreads
- Body of the Stitch loop becomes a function
- Pointer to the function is passed into the pthread
- Struct with copies of all variables in the scope of the Stitch loop passed in.
- (Very) limited variety of accumulators can be used in Stitch loops, and are then reconciled automatically.
Not Very Pretty

int main() {
    int a[255] = {1, ..., 1};
    int b[255] = {2, ..., 2};
    int_am acc;
    int i = 0;

    stitch i from 0 to 255 by 1: {
        acc = a[i] * b[i];
    }

    print(acc);
    return 0;
}

(But it works)
**Pros**

- Much simpler to write
- No dirty mutex’s
- Automatic splitting of workload

**Cons**

- Limited in application
- Nested Stitch loops give no benefit, but add overhead
- Pthread code definitely not optimized (but not too bad either)
And now for something completely the same...

Scanner -> Parser

C-AST -> Semantic Analysis

C Code Generation -> Write to File

AST

Singer

GCC
Highpoints of C Code Generation

- Stitch to pthread loop.
- Body of the Stitch loop is turned into a C function.
- Stitch functions and range info are named procedurally.
- print() and error() are dynamically typed into proper printf() call.
Testing

- Automated test suite
- Generate C Code and compile.
- Run compiled C, diff the output
- Positive (should compile and generate good output).
- Negative (shouldn’t compile)
- Modular tests → anyone can add
Demo

• Something parallelizeable
• Something interesting
• Something that we could see the results of

*Image Curves*

• Used all the time in Photoshop
• Map input to output based on predefined curve
Simplest Curve - Invert
More Useful - Increase Contrast
/* Image Inverter */

int main(){

    int curve[256] = { 255, ... , 2, 0 };

    /*File IO*/

    int i = 0;
    //55 = header offset, 98592 = size of file
    stitch i from 55 to 98592 by 1:{
        int tmp = 0;
        tmp = buffer[i];

        if (tmp < 0) {
            tmp = (tmp % 256) * -1;
        }

        buffer[i] = curve[tmp];
    }

    /*More IO*/

}