multafila

“many threads”

Chae Jubb - Project Manager
Zeynep Toraman - Tester and Validator
Alessandra Poblador - Systems Integrator
Bo Yin - Systems Architect
Aiden Yang - Language Guru

Introduction

- a programming language that makes parallel computing more accessible and efficient
- C-like in syntax
- designed to make writing multithreaded programs easier than ever before

Introduction

multafila is:

- simple and easy
  - clean and concise, avoids verbosity
- robust and high-performance
  - eliminates error, compiles to C
- flexible and versatile
  - adapts to personal programming style, complexity of program
- lightweight yet powerful
  - small language with building blocks that combine
- portable

thread

```
thread thread_name;
```

spawn

```
thread t;
spawn ( t ) {

}
```

barrier

```
barrier;
```

pfor

```
thread thread_array[n];
pfor ( thread_array, i, 0 ){

}
```

lock

```
lock ( var1, var2, … ) {

}
```

A threaded “Hello, world!”

spawn and **barrier** in action

```
int main ( ) {
      thread print1;
      thread print2;

      spawn( print1  ) {
            printOut(“Hello, world!”);
      }

      spawn( print2 ) {
            printOut(“Hello again!”);
      }

      barrier;

      return 0;
}
```

Lock it down

pfor and **lock** in action

```
/* int x[10], y[10] */

int result;
int i;
threads threads[10];
pfor ( threads, i, 0 ) {
    lock( result ) {
        result = x[i] + y[i];
    }
}
```

Multi-lock

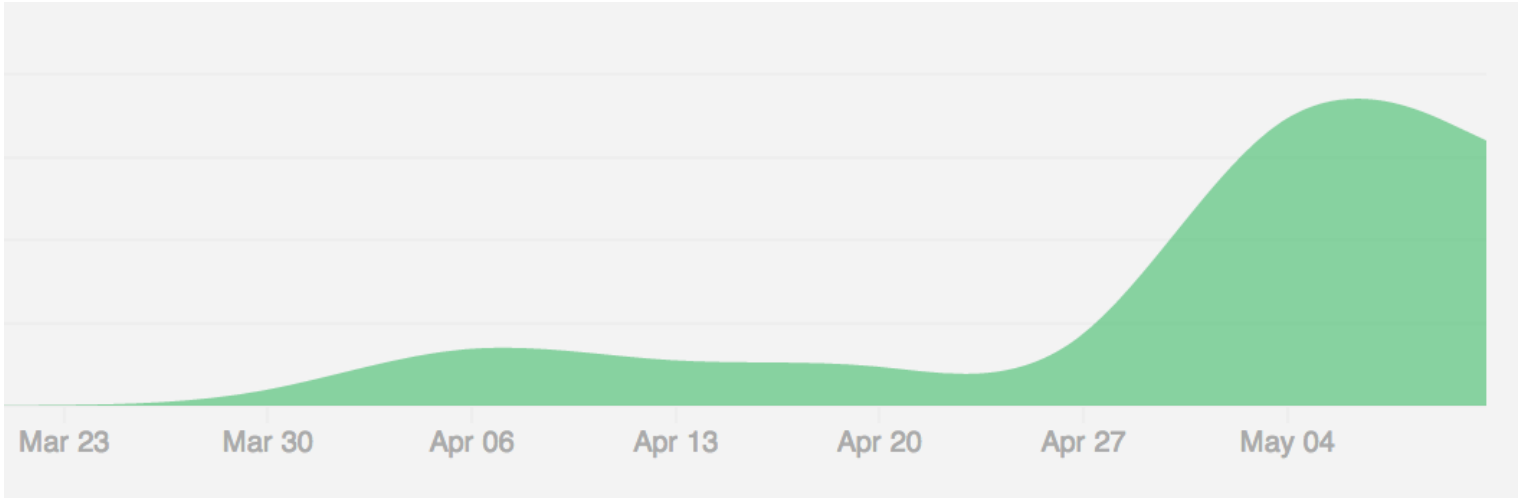
never forget a variable!

never miss a deadlock!

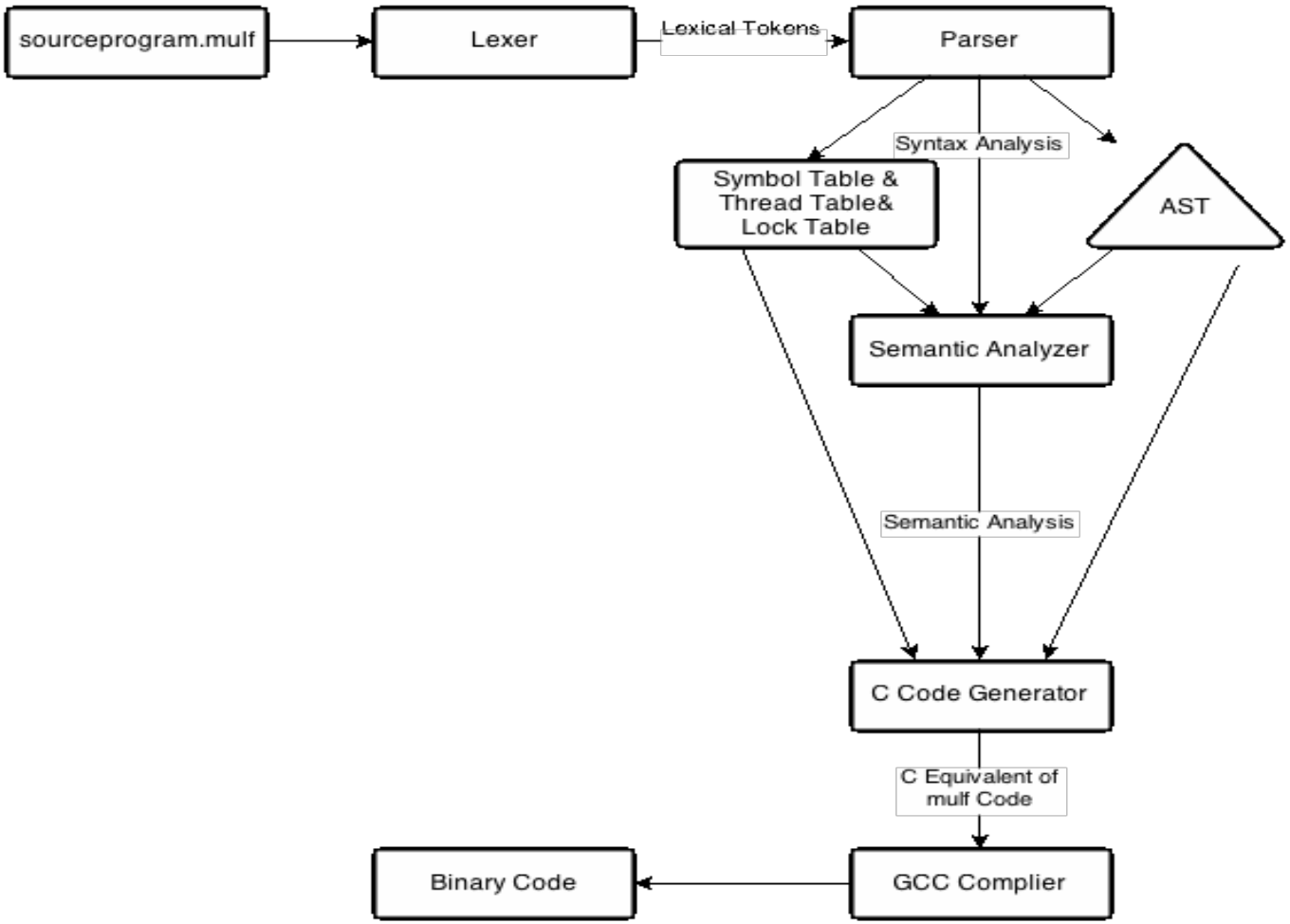
```
pfor ( threads, i, 0 ) {
    lock( a, b ) {
        b = b + 1;
        a = a + b;
    }

    lock( a, b ) {
        b = b + 1;
        a = a + b;
    }
}
```

Project Management

- GitHub
- Trello
- Email



Translator Architecture



Software environments

- vim, Sublime, TextWrangler
- valgrind, gdb
- GitHub

- Linux/OS X/Windows platform
- GCC version 4.2.1
- POSIX library
- flex version 2.5.35
- GNU Bison version 2.3
- GNU bash 3.2.48

Compiler generator tools

- flex for lexer
- bison for parser
- very easy to learn and essential to iterating quickly on the language and adapting grammar
- POSIX implementation for multithreaded features

Testing

- testall.sh:

Shell script for running all test programs

- Test cases:

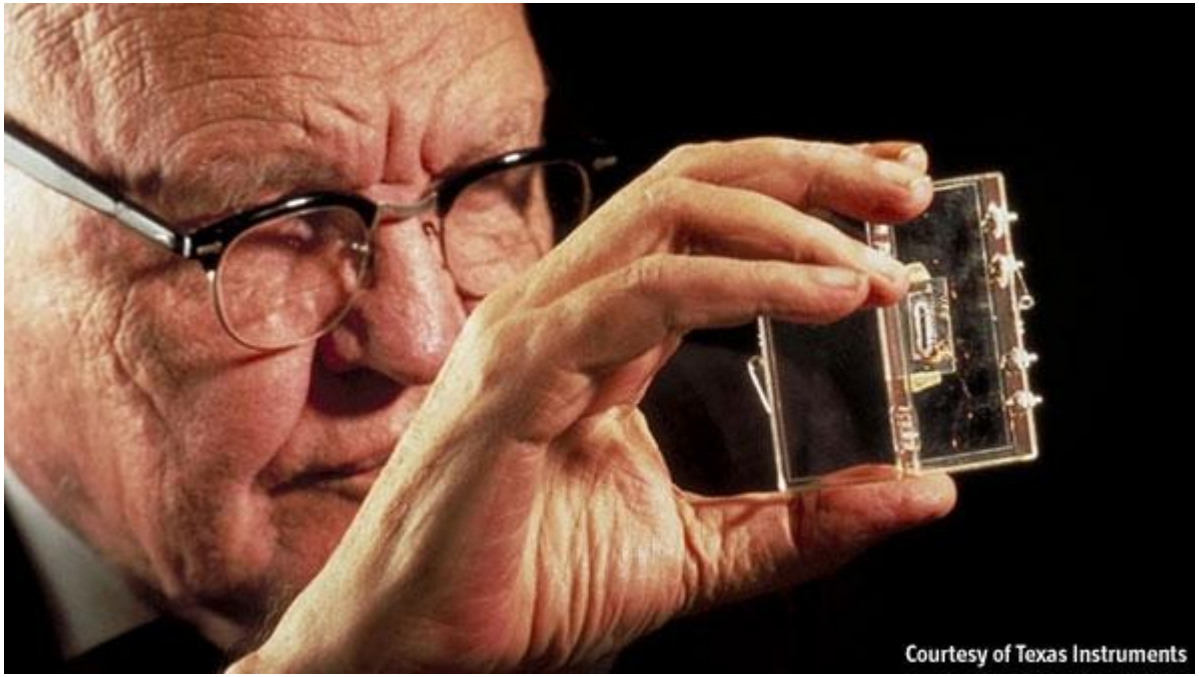
selected to cover trivial aspects of the language

Lessons learned

- Chae
- Alessandra
- Zeynep
- Aiden
- Bo

multafila is the future

- Parallel
- C-style Syntax
- Intuitive
- Easy



DEMO

How to Compile & Run

- **To Compile**

use the multafila BASH script:

./multafila <source_file> [<output_file>]

Sends the output of the translator directly into gcc

- **To Run**

Run the a.out (or <output_file>.out if specified)