

CSC 255/455 Project Introduction

Bin Bao

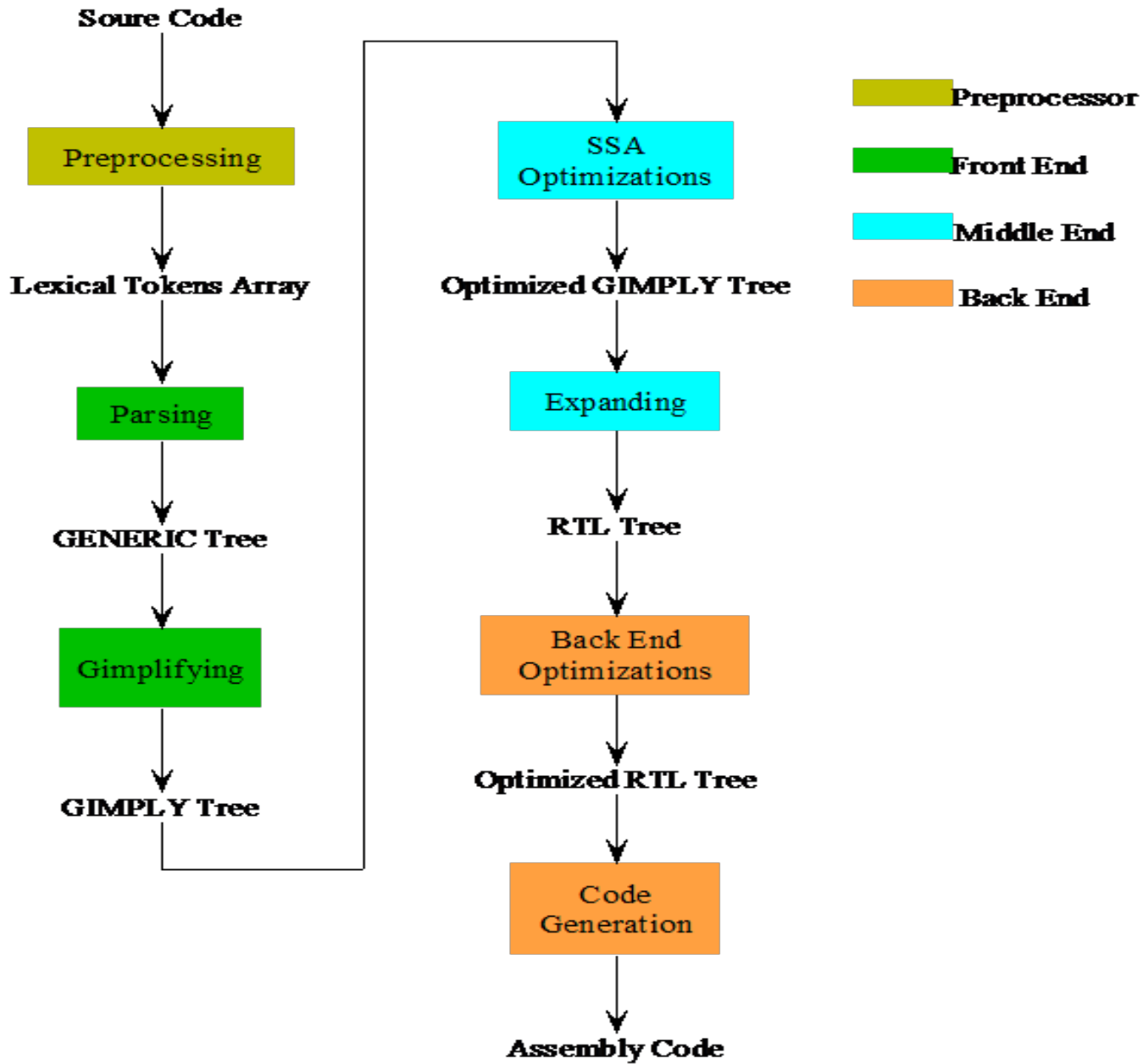
Platform

- GCC 4.2.2
- GNU C Compiler / GNU Compiler Collection

GCC Structure

- Front-end
 - C/C++, Fortran, Ada, Java, ...
- Middle-end
 - dead code elimination, global value numbering, ...
- Back-end
 - x86, POWER, MIPS, ...

http://en.wikipedia.org/wiki/GNU_Compiler_Collection



Goal

- Implement your own optimizations to optimize programs written in a subset of C language
 - No pointer dereference
 - No dynamic data allocation
 - Everything in a single file
 - Be careful about correctness

Project Content

- Warm up
 - 1. Trivial: count the number of executed gimple statements (going to be the metric of final competition)
 - 2. Weird control flow
- 3. Value Numbering
 - Handling of array is only required for CSC455

Project Content (cont.)

- 4. Dataflow
 - Build Def-Use graph
 - Dead code elimination
 - Constant propagation (CSC455 only)
- 5. Group Competition
 - Open benchmarks
 - Hidden benchmarks

Where to put your optimizations

- Operate at gimple level
- In an individual file, *gcc-4.2.2/gcc/cs255.c*
- Create a separate pass, and flag “*-O2 -fcs255*” will invoke your pass
- All other gcc optimizations will be turned off

Alternative choice

- Use Ruby to write a separate optimizer module
- Read-in gimple tree, output C back to GCC
- We will provide some infrastructure
- Pros
 - Data structures are easy to use, e.g. hash table
 - Friendly to team work
- Cons
 - Study a new language
 - New from this semester

Example: Hash Table

- In GCC, the interface of *htab_create*
 - Define *hash* function
 - Define *equal* function
 - Define *delete* function
 - Create a hash table *htab_create*
 - Define *lookup* function
- Look at existing code in GCC for reference, *hashtab.c*, etc.

```
typedef struct string_int_map
{
    char* expr;
    unsigned int number;
} string_int_map_t;
```

```
/* Hash function */
static hashval_t string_int_map_hash (const void *v)
{
    return htab_hash_string (((const string_int_map_t *)v)->expr);
}
```

```
/* Equal function. */
static int string_int_map_eq (const void *va, const void *vb)
{
    const string_int_map_t *a = va, *b = vb;
    return strcmp(a->expr, b->expr)==0;
}
```

```
/* Delete function*/
static void string_int_map_delete (void *v)
{
    free( ((string_int_map_t *)v)->expr );
    free(v);
}
```

```
htab_t myTable = htab_create(500, string_int_map_hash, string_int_map_eq,  
    string_int_map_delete);
```

```
//To look up an element with a key “myString” on the hash table
```

```
string_int_map_t element;
```

```
element.expr = myString;
```

```
string_int_map_t *r = (string_int_map_t *) htab_find( myTable, &element );
```

```
if (r == NULL)
```

```
    printf(“there is no entry with key %s in myTable\n”, myString);
```

```
//To insert an new element
```

```
string_int_map_t **slot = (string_int_map_t **) htab_find_slot ( myTable, &element,  
    INSERT );
```

```
if (*slot == NULL) {
```

```
    string_int_map_t *newEntry = XCNEW( string_int_map_t );
```

```
    newEntry->expr = myString;
```

```
    newEntry->number = 0;
```

```
    *slot = (void *) newEntry;
```

```
}
```

Example: Hash Table (cont.)

- In Ruby

```
vn_table = Hash.new
```

```
vn_table["a"] = 4
```

```
vn_table["a"]
```

Rules about discussion

- You are encouraged to discuss the project and work out the algorithms on paper, board or chat programs with others.
- You are encouraged to share with each other how to use GCC internal data structures.
- You should NOT copy other people's code without explicit acknowledgement.
 - From GCC, document and explain it
 - From other students, no credit

Demo

Compilation and Installation

```
$ cd /localdisk/bao/gcc
```

```
$ svn co file:///p/compiler/repos/gcc-4.2.2/trunk  
gcc-4.2.2
```

```
$ mkdir obj
```

```
$ cd obj
```

```
$ ../gcc-4.2.2/configure --prefix=/localdisk/bao/bin
```

```
$ make
```

```
$ make install
```

Run

```
$ /localdisk/bao/bin/bin/gcc -O2 -fcs255 begin.c -o  
begin
```

```
$ /localdisk/bao/bin/bin/gcc -O2 -fcs255 begin.c -o  
begin -v
```

```
- /localdisk/bao/bin/libexec/gcc/i686-pc-linux-gnu/  
4.2.2/cc1 -quiet -v begin.c -quiet -dumpbase begin.c -  
mtune=generic -auxbase begin -O2 -version -fcs255 -  
o /tmp/cczZ6ce4.s
```

Code Browsing --- Eclipse (CDT)

- 'File'->'New'->'C Project...!', then in the next frame
 - type in the project name
 - uncheck the box '*Use default location*'
 - give the full path of the top directory containing source files
 - click '*finish*' to create an empty C project

Debugging

- *gcc* is just a driver, use *-v* to show details
- *gdb*
 - *file, run, break, print, bt, ...*
- Run *gdb* in *Emacs*
- Dump gimple tree node
 - *p print_generic_stmt(stdout, node, 2)*
 - *p dump_node(node, 2, stdout)*
- *-fdump-gimple-tree, -fdump-cs255-tree-gimple*

The First Project Assignment

Counting Number of Statements

- Due: Wednesday, Feb. 4th
- Insert code into a program to count and output the number of executed statements
- The code for counting the number of basic blocks and statements statically is in *cs255.c*
- The code for getting the dynamic count of the basic blocks is also there

The End

Special thanks to last year's TA, Xiaoming Gu.