The Gnu Compiler Collection (GCC)

COMP 3002

Carleton University

Canada's Capital University

Outline

- History
- Capabilities
- Design
  - Parsing
  - Intermediate representations
  - Code generation
- Additional tools

Carleton UNIVERSITY
Canada's Capital University

Canada's Capital University

Sometimes a few of the users try to hold total power over all the rest. For example, in 1984, a few users at the MIT AI lab decided to seize power by changing the operator password on the Twenex system and keeping it secret from everyone else. (I was able to thwart this coup and give power back to the users by patching the kernel, but I wouldn't know how to do that in Unix.)

However, occasionally the rulers do tell someone. Under the usual `su' mechanism, once someone learns the root password who sympathizes with the ordinary users, he or she can tell the rest. The "wheel group" feature would make this impossible, and thus cement the power of the rulers.

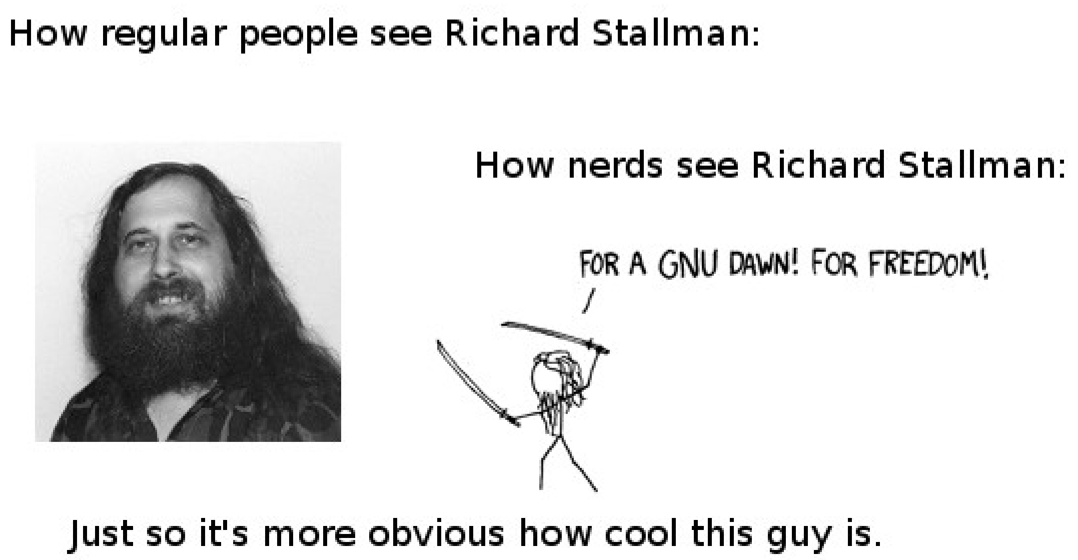
I'm on the side of the masses, not that of the rulers. If you are used to supporting the bosses and sysadmins in whatever they do, you might find this idea strange at first.

History

- The Free Software Foundation
  - Non-profit corporation
  - Founded in 1985 by Richard Stallman (RMS)
  - Initially founded to support the GNU Project
- GNU Project
  - Goal: to develop “a sufficient body of free software [...] to get along without any software that is not free.”
  - GNU Operating System
    - First released in 1992 with a Linux kernel
    - Debian GNU/Hurd (available in *unstable*)
  - Recent developments: GNOME and Gnash
  - Needed development tools to get started

Carleton UNIVERSITY
Canada’s Capital University

RMS


How regular people see Richard Stallman:
How nerds see Richard Stallman:
FOR A GNU DAWN! FOR FREEDOM!
Just so it's more obvious how cool this guy is.

xkcd.com

UNIVERSITY
Canada's Capital University

Ancient History

- Richard Stallman started GCC (1985)
  - Extended an existing Pastel compiler
- Rewritten in C by Stallman and Len Tower (1987)
  - Became the compiler for the GNU Project
- Development supervised by the Free Software Foundation (FSF)
- First stable release in 1991 GCC 1.x

The EGCS Project

- From 1992-1997, official GCC code was carefully controlled by FSF
  - Getting changes submitted was frustrating for many
- In 1997, EGCS merged several experiment forks of GCC
  - Included g77 (Fortran), PGCC (Pentium-optimized GCC), C++ improvements, new architectures and operating systems
- In 1999, EGCS became the official GCC 2.95 compiler

Carleton UNIVERSITY
Canada's Capital University

GCC Today (GCC 4.4)

- Architectures
  - Alpha, ARM, Atmel, AVR, Blackfin, HC12, H8/300, IA-32, (x86), x86-64, IA-64, Motorola, 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU, System/390/zSeries, SuperH, SPARC, VAX, A29K, ARC, ETRAX CRIS, D30V, DSP16xx, FR-30, FR-V, Intel i960, IP2000, M32R, 68HC11, MCORE, MMIX, MN10200, MN10300, Motorola 88000, NS32K, ROMP, and more
- Non FSF
  - Cortus APS3, D10V, LatticeMico32, MeP, Motorola 6809, MicroBlaze, MSP430, Nios II and Nios, OpenRISC 1200, PDP-10, TIGCC (m68k variant), Z8000, PIC24/dsPIC, NEC SX architecture[18]

Carleton UNIVERSITY
Canada's Capital University

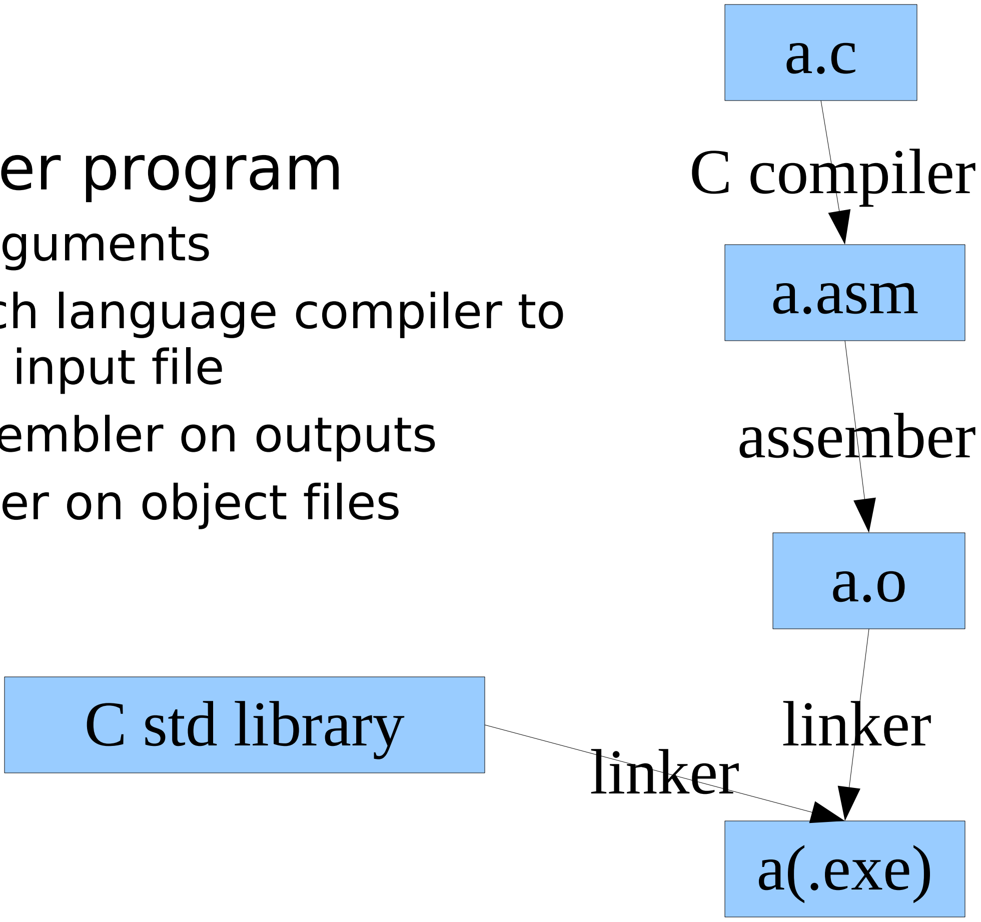
GCC Today

- Languages (standard)
  - C (gcc), C++ (g++), Java (gcj), Ada (GNAT), Objective-C (gobjc), Objective-C++ (gobjc++), Fortran (gfortran)
- Non-Standard
  - Modula-2, Modula-3, Pascal (gpc), PL/I, D (gdc), Mercury, VHDL (ghdl).[15] A popular parallel language extension, OpenMP

Carleton UNIVERSITY
Canada's Capital University

Structure

- gcc is a driver program
  - interprets arguments
  - decides which language compiler to use for each input file
  - runs the assembler on outputs
  - runs the linker on object files



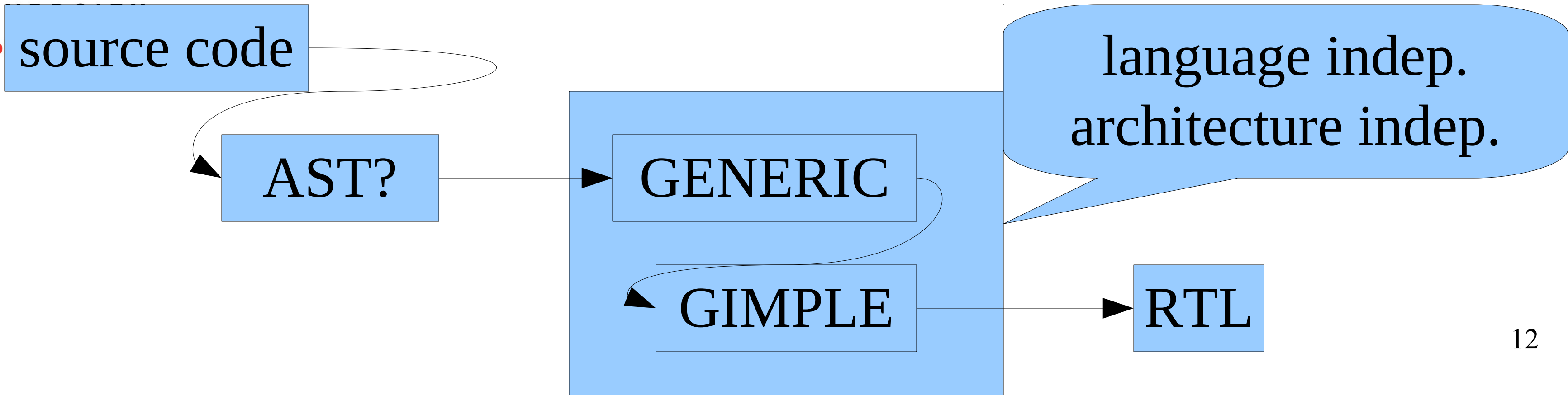
GCC Language Compilers

- All language compilers
  - Read source code
  - Output assembly code
- Language compilers have different *front ends*
  - Each front end parses input and produces an abstract syntax tree
- AST is converted to a common middle-end format
  - GENERIC
  - GIMPLE

Canada's Capital University

GCC Front Ends

- All GCC front ends are currently hand-coded recursive descent parsers
  - (Version 2 of the C compiler was based on a bison grammar)
- C, C++, and Java front ends produce GIMPLE directly
- Other front ends convert AST into GENERIC



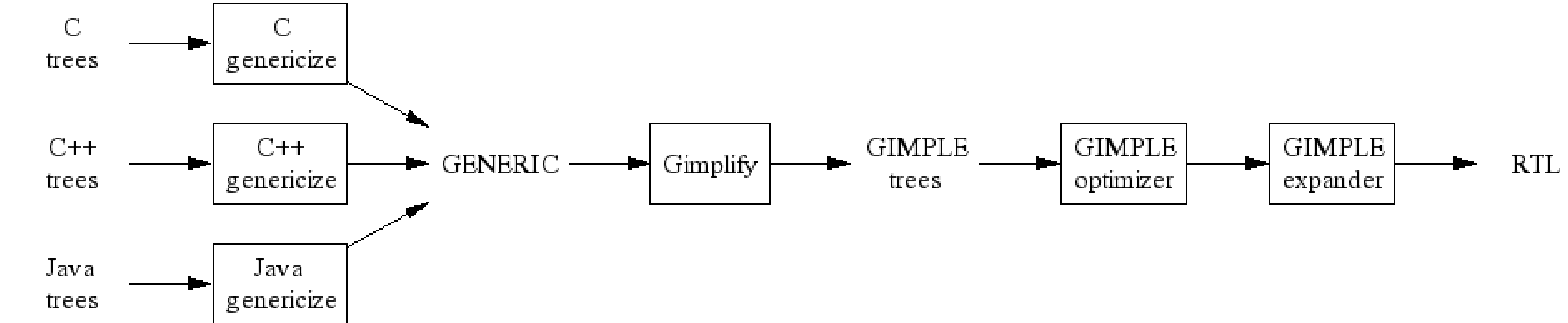
GENERIC

- A standardized form of abstract syntax tree
  - **Types:** offset_type, enumeral_type, boolean_type, char_type, integer_type, real_type, reference_type, ...
  - **Language constructs:** identifier_node, block,
  - **Constants:** integer_cst, real_cst, vector_cst, ...
  - **Declarations:** function_decl, label_decl, field_decl,..
  - **References:** component_ref, indirect_ref, array_ref,...
  - **Expressions:** compound_expr, modify_expr, cond_expr, plus_expr, mul_expr, convert_expr, ...

Carleton UNIVERSITY
Canada's Capital University

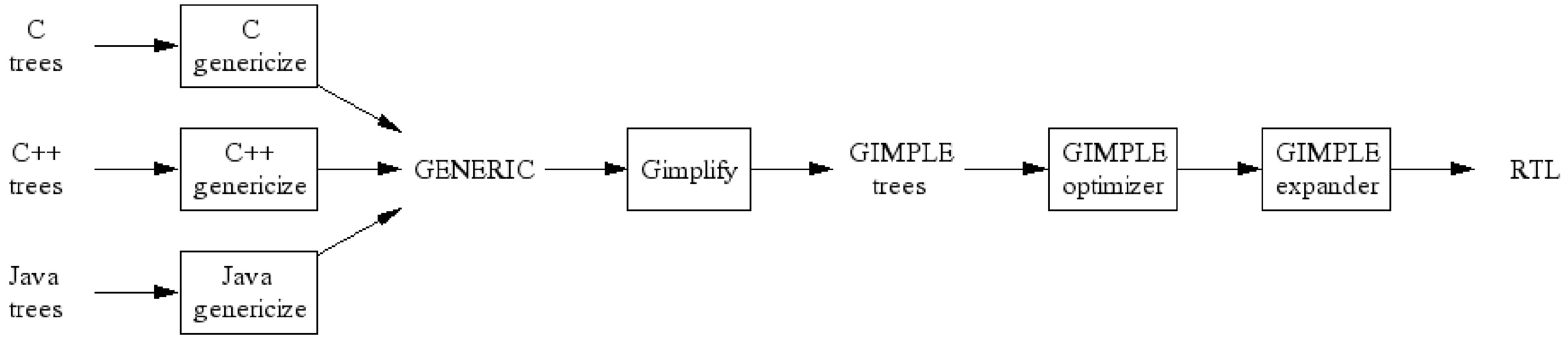
GENERIC

- GENERIC is a useful standard intermediate representation
- Expressions can be too complicated for easy optimization
- GENERIC trees are gimplified into GIMPLE
  - GIMPLE expressions are three address codes
  - More complicated expressions generate temporary variables



GIMPLE

- Grammar and Example
  - gimple.g
  - gimple.eg



Middle-end Optimizations

- GIMPLE code gets optimized
  - dead code elimination
  - partial redundancy elimination
  - global value numbering
  - sparse-conditional constant propagation
  - Loop optimization
  - Jump threading (control-flow optimization)
  - Common subexpression elimination
  - Instruction scheduling (reordering)

Carleton UNIVERSITY
Canada's Capital University

RTL

- Register transfer language (RTL)
  - A Scheme-like language based on virtual registers
- Initial RTL is generated with hints about the target machine
- RTL is refined through many (58) passes
- Final passes use target machine registers and instructions
- From there, conversion to machine-specific assembly language is easy

Canada's Capital University

RTL

- gcc -fdump-tree-all -fdump-rtl-all -S test.c
- less `ls -c test.c.???t*`

Canada's Capital University

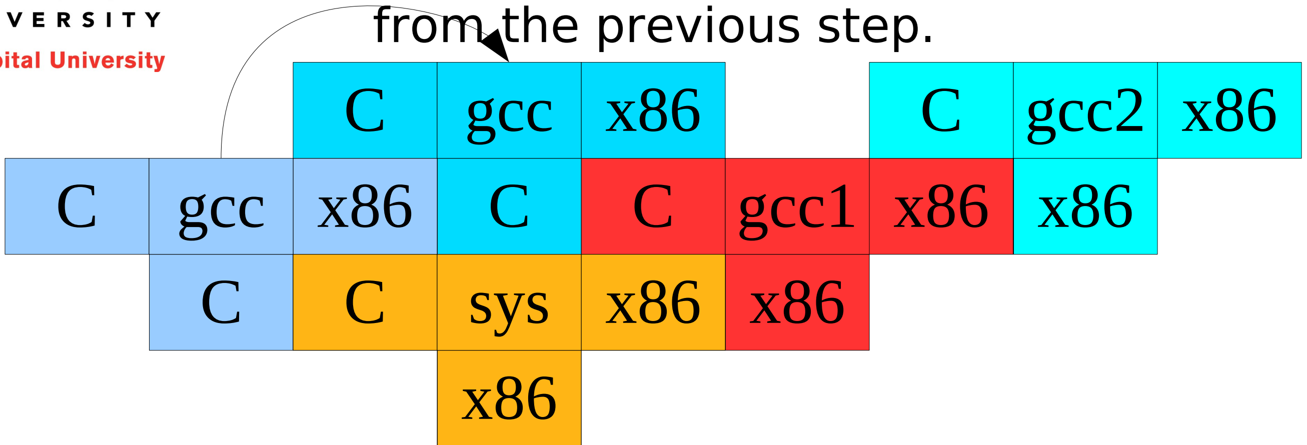
Finishing Up

- Finally, assembly-language output is assembled into an object
- gcc runs the linker on all object files and C libraries to produce an executable file

Carleton UNIVERSITY
Canada's Capital University

Building GCC

- The bootstrapping process:
  - Build tools necessary to build the compiler.
  - Perform a 3-stage bootstrap of the compiler.
    - Includes building three times the target tools for use by the compiler such as binutils (bfd, binutils, gas, gprof, ld, and opcodes)
  - Perform a comparison test of the stage2 and stage3 compilers.
  - Build runtime libraries using the stage3 compiler from the previous step.



Summary

- GCC
  - a real open-source success story
  - *libre* and *gratis*
  - the world's most versatile compiler
  - production strength
  - gcc can make or break a hardware platform
- Compiling to GIMPLE is not much work
  - Offers a quick way to make an optimizing compiler

Carleton UNIVERSITY
Canada's Capital University