A Look at some Compilers

MATERIALS FROM THE DRAGON BOOK AND WIKIPEDIA

MICHAEL WOLLOWSKI

EQN

- Takes inputs like "`E sub 1`" and produces commands for text formatter TROFF to produce "E_1"

EQN

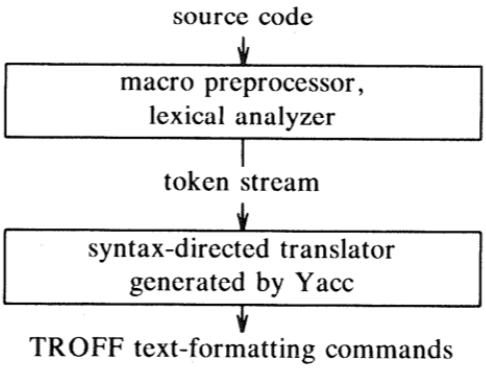


Fig. 12.1. EQN implementation.

EQN

- Treating EQN as a language and applying compiler technologies has several benefits:
  - Ease of implementation.
  - Language evolution. In response to user needs

Pascal

Developed by Nicolas Wirth.

Generated machine code for the CDC 6000 series machines

To increase portability, the Pascal-P compiler generates P-code for an abstract stack machine.

One pass recursive-descent compiler

Storage is organized into 4 areas:

- Code for procedures
- Constants
- Stack for activation records
- Heap for data allocated by the **new** operator.

Procedures may be nested, hence, activation record for a procedure contains both access and control links.

CDC 6000 series

The first member of the CDC 6000 series was the supercomputer CDC 6600,

Designed by Seymour Cray and James E. Thornton

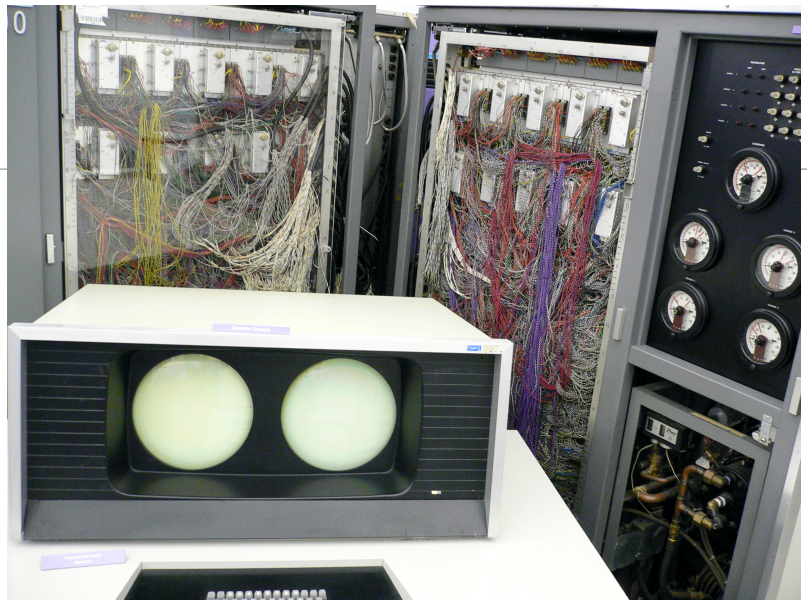
Introduced in September 1964

Performed up to three million instructions per second, three times faster than the IBM Stretch, the speed champion for the previous couple of years.

It remained the fastest machine for five years until the CDC 7600 was launched.

The machine was Freon refrigerant cooled.

Control Data manufactured about 100 machines of this type, selling for $6 to $10 million each.



CDC 6000 series By Steve Jurvetson from Menlo Park, USA - Flickr, CC BY 2.0, https://commons.wikimedia.org/w/index.php?curid=1114605

CDC 205

CDC 205 DKRZ

CDC 205

CDC 205 wiring, davdata.nl

Pascal

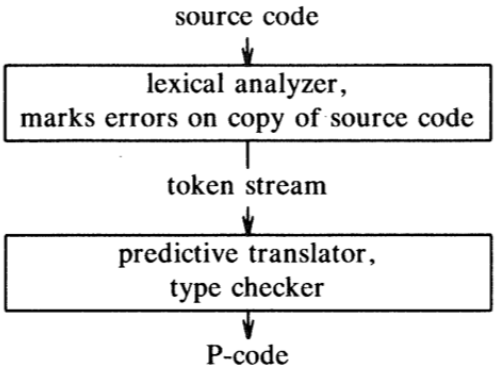


Fig. 12.2. Pascal-P compiler.

Pascal

One of the compiler writers states about the use of a one-pass compiler:

- Easy to implement
- Imposes severe restrictions on the quality of the generated code and suffers from relatively high storage requirements.

C

Compiler for the PDP-11:

- Two-passes
- Optional third pass, to perform optimization (removes redundant and inaccessible statements)

C

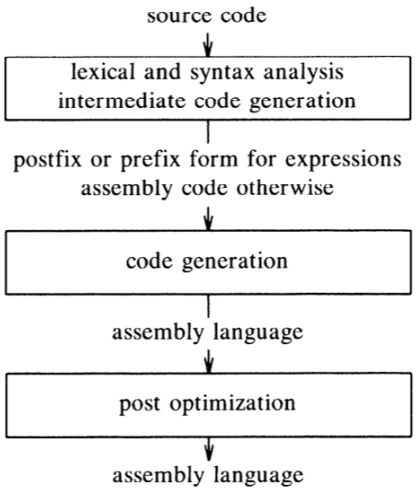


Fig. 12.3. Pass structure of C compilers.

C

Recursive descent parser

Parse everything except expressions, for which operator precedence is used

Intermediate code consists of postfix notation for expressions and assembly code for control-flow statements.

Storage allocation for local names is done during the first pass, so names can be referred to using offsets into an activation record.

Within back-end, expressions are represented by syntax trees.

In the PDP-11 compiler, code generation is implemented by a tree walk.

PDP-11

The **PDP-11** is a series of 16-bit minicomputers sold by Digital Equipment Corporation (DEC) from 1970 into the 1990s,

In total, around 600,000 PDP-11s of all models were sold, making it one of DEC's most successful product lines.

The PDP-11 is considered by some experts to be the most popular minicomputer ever.

PDP-11

Bliss/11

For PDP-11

Optimizing compiler from a world that has ceased to exist, a world where memory space was at a premium to the extent that it made sense to optimize for space rather than time.

Pioneered the syntax-directed approach to optimization

It had no goto statements

As such, it was possible to perform data-flow analysis on the parse tree directly rather than the flow graph.

Bliss/11

BLISS is a system programming language developed at Carnegie Mellon University by W. A. Wulf, D. B. Russell, and A. N. Habermann around 1970.

It was perhaps the best known systems programming language right up until C made its debut a few years later.

Since then, C took off and BLISS faded into obscurity.

BLISS is a typeless block-structured language based on expressions rather than statements, and includes constructs for exception handling, coroutines, and macros. It does not include a goto statement.

Bliss/11

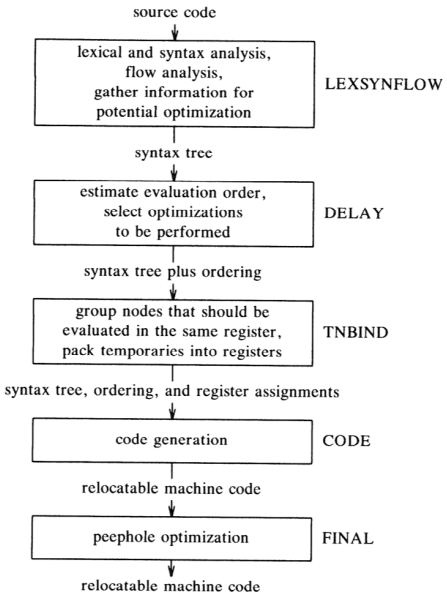


Fig. 12.5. The BLISS/11 compiler.

Bliss/11 - LEXSYNFLO

Lexical analysis and parsing

Recursive-descent parser

Syntax of the language enables us to build flow-graph and determine loops an loop entries as we parse.

Determines common subexpressions

Detect groups of similar expressions

They are candidates for replacement by a single subroutine

This replacement makes a program run more slowly but can save space

Bliss/11 - DELAY

Examines syntax tree to perform optimizations:

- Invariant code motion
- Elimination of common subexpressions
- Order of expression evaluation
- Algebraic laws to determine whether reordering should be performed
- Decide whether it is cheaper to use numerical or control flow evaluation of conditional expressions.

Bliss/11 - TNBIND

- Considers which temporary names should be bound to registers
- First group nodes of the syntax tree that should be assigned the same register
- Advantage gained of keeping a temporary in a register is estimated
- Registers are assigned until used up, packing the most advantageous nodes into registers first

Bliss/11 - CODE

- Converts tree with its ordering and register allocation to relocatable machine code.

Bliss/11 - FINAL

- Repeatedly performs peephole optimization until no further improvements.
- Improvements:
  - Elimination of conditional and unconditional jumps to jumps
  - Complementation of conditionals.
  - Redundant and unreachable code is eliminated
  - Local propagation of constants
  - Machine dependent replacements.
    - Example: Replace jump instruction with PDP-11 branch instructions

Modula-2

Designed to produce good code.

Using optimizations that provide high payoff for little effort

Parser was generated using Yacc

Produces syntax trees in two passes

Intermediate code is P-code for compatibility with many Pascal compilers

Procedure call format agrees with that of the Pascal and C compilers running under Berkeley UNIX

Procedures in the three languages can be integrated easily

Optimizations are similar to those described for Bliss/11

Modula-2

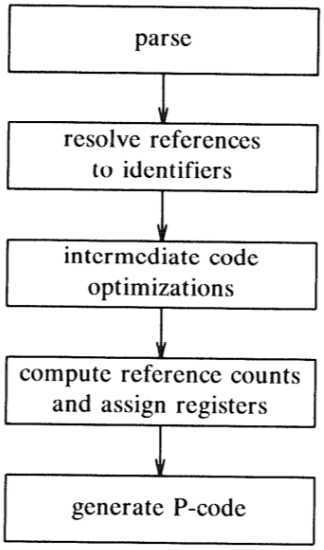


Fig. 12.6. Modula-2 compiler passes.