# A history of compilers

Peter Sestoft

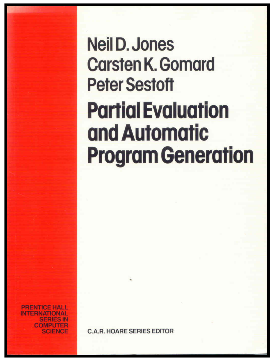
sestoft@itu.dk

Dansk Datahistorisk Forening

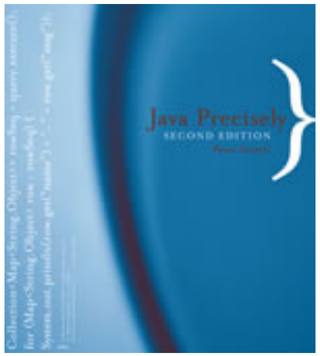
2014-01-23 v 1.0

# The speaker

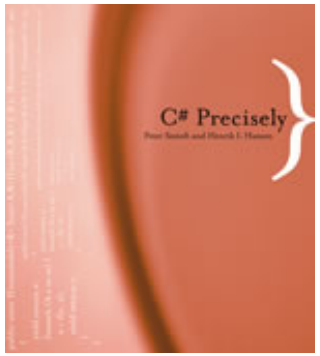
- MSc 1988 computer science and mathematics and PhD 1991, DIKU, Copenhagen University
- KU, DTU, KVL and ITU; and AT&T Bell Labs, Microsoft Research UK, Harvard University
- Programming languages, software development, ...
- Open source software
  - Moscow ML implementation, 1994...
  - C5 Generic Collection Library, with Niels Kokholm, 2006...




1993

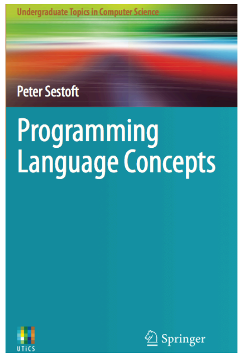



2002 & 2005




2004 & 2012


2007




2012


2014

# Outline

- What is a compiler?
- A history of the history of ...
- Early autoprogramming systems
- The FORTRAN I compiler
- Some Algol compilers
- Lexing and parsing
- Code generation
- Intermediate languages
- Optimization
- Flow analysis
- Type systems

# What is a compiler (today)?

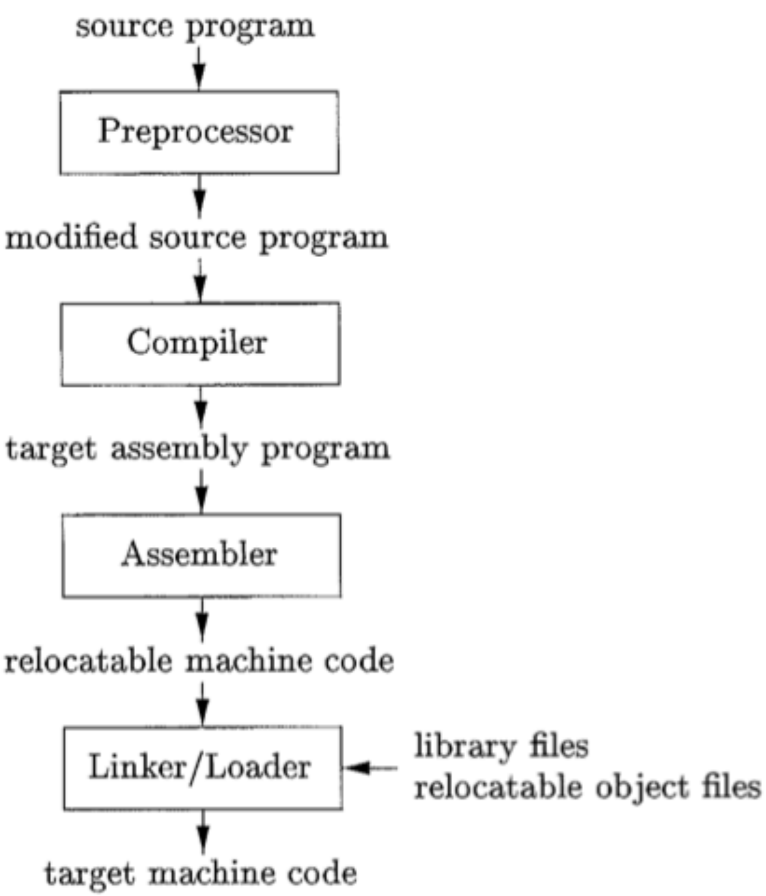
```c
for (int i=0; i<n; i++)
  sum += sqrt(arr[i]);
```

C language source program

**clang** →

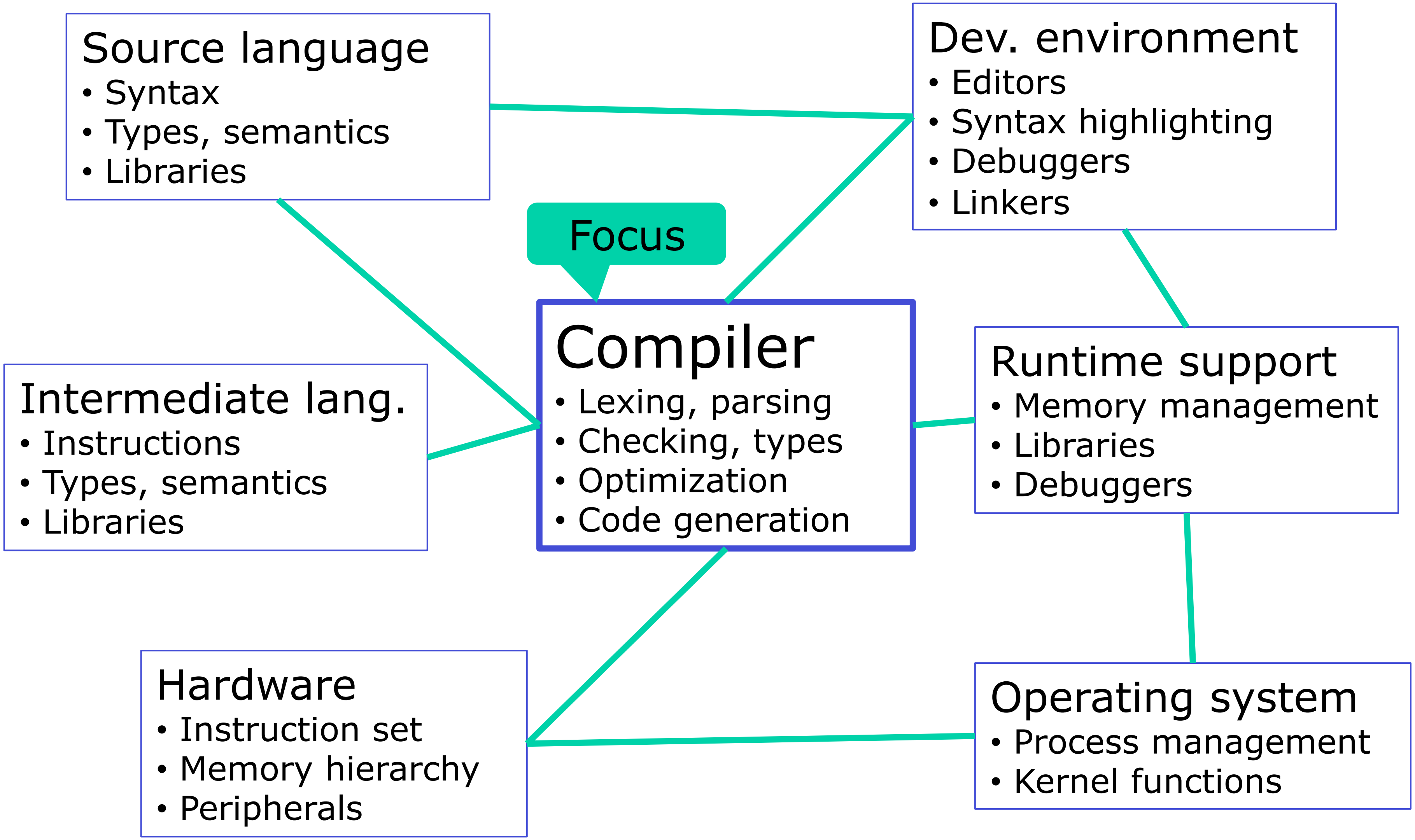
```
LBB0_1:
movl     -28(%rbp), %eax          // i
movl     -4(%rbp), %ecx           // n
cmpl     %ecx, %eax
jge      LBB0_4                   // if i >= n, return
movslq   -28(%rbp), %rax          // i
movq     -16(%rbp), %rcx          // address of arr[0]
movsd    (%rcx,%rax,8), %xmm0     // arr[i]
callq    _sqrt                    // sqrt
movsd    -24(%rbp), %xmm1         // sum
addsd    %xmm0, %xmm1             // sum + ...
movsd    %xmm1, -24(%rbp)         // sum = ...
movl     -28(%rbp), %eax          // i
addl     $1, %eax                 // i + 1
movl     %eax, -28(%rbp)          // i = ...
jmp      LBB0_1                   // loop again
```

x86 machine code

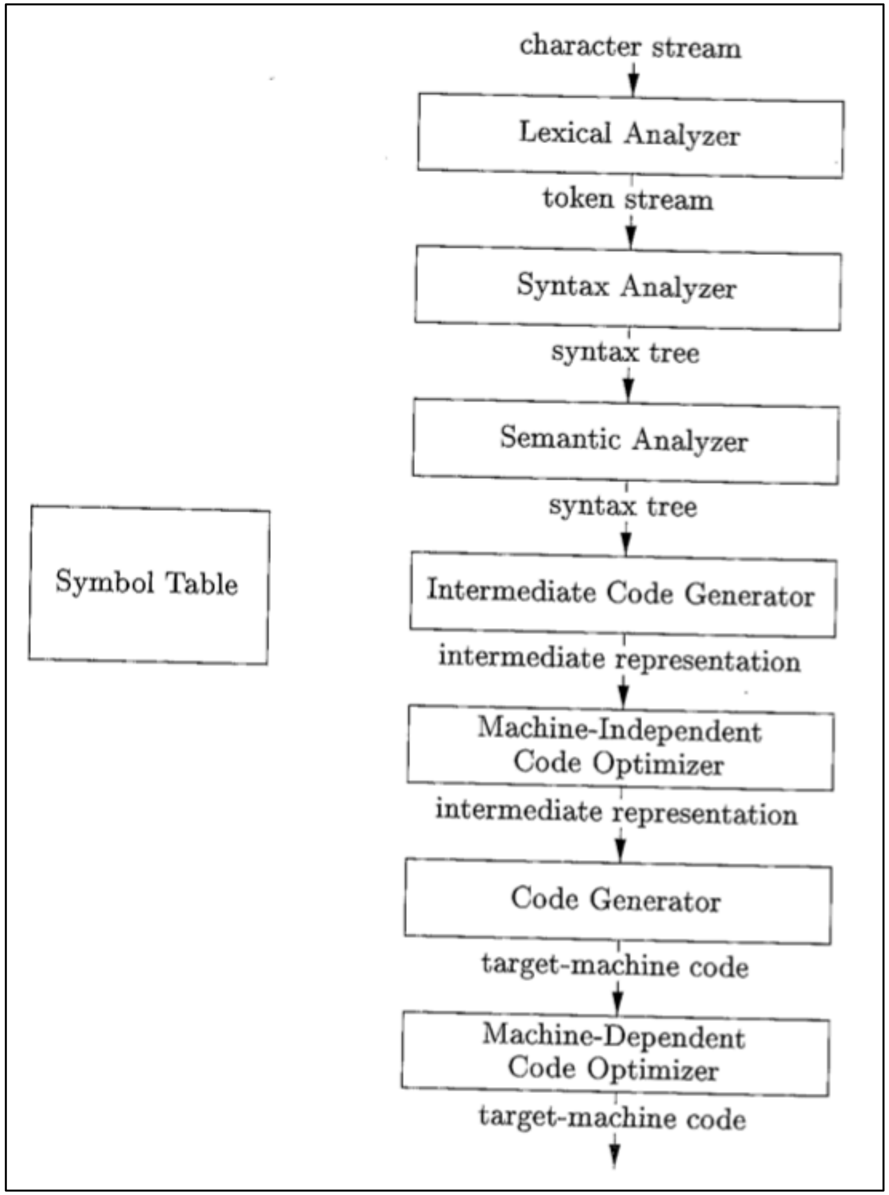


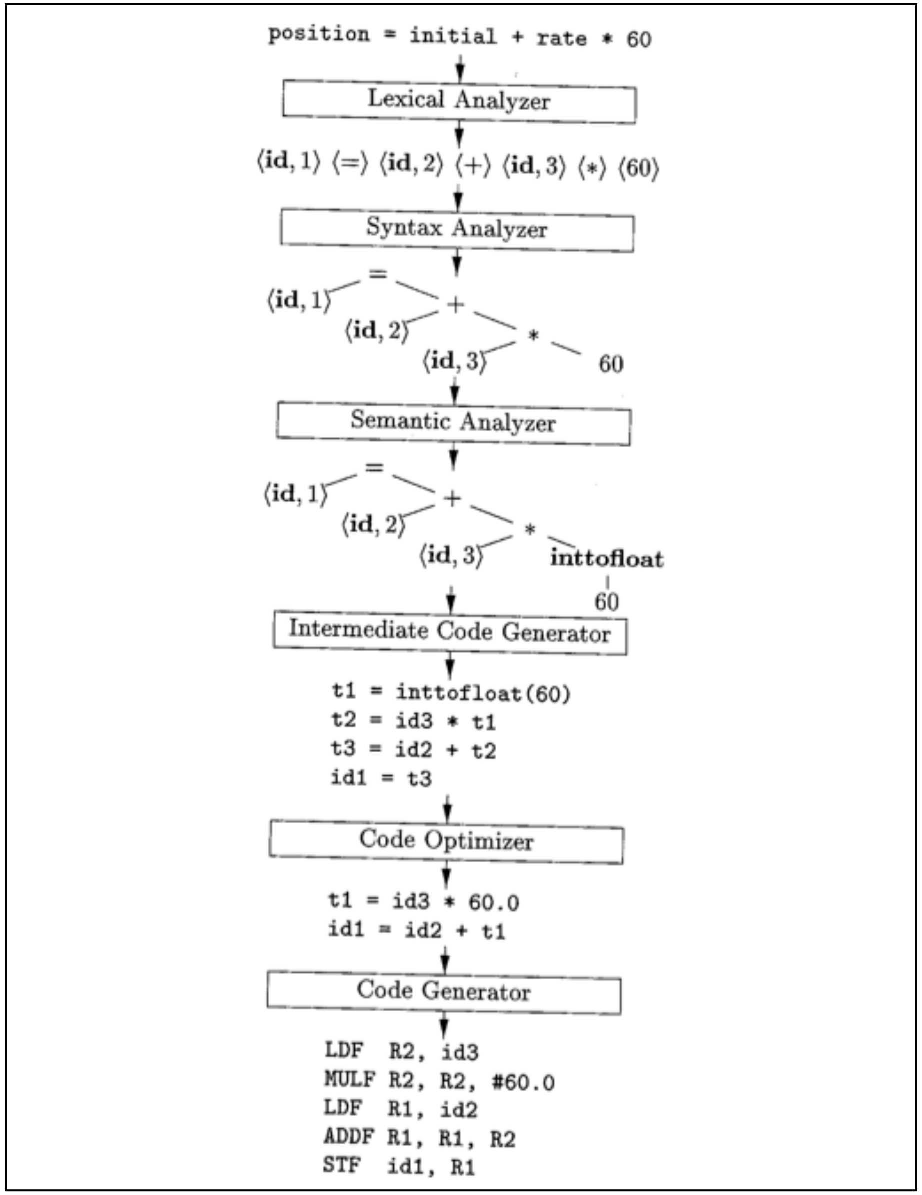

From Aho et al

# A compiler in context



# Conceptual phases of a compiler




From Aho et al

# Why "compiler" not "translator"?

- Hopper: *A Programmer's Glossary*, 1 May 1954:

> Compile (verb) - The process of producing from pseudo-code a specific routine for a particular problem by:
>
> 1) decoding elements of information expressed in pseudo-code and segmenting the problem;
> 2) selecting or generating the required sub-routines;
> 3) transforming the subroutines into specific coding and entering them as elements in the problem routine;
> 4) maintaining a record of the subroutines used and their position in the problem routine.
>
> Compilation decodes the pseudo-code and processes static and dynamic subroutines and generators to produce a specific routine for a problem before computation.

Jones:1954:ASurvey

- Hopper's A-2 compiler collected & inlined subroutines
  - In modern terminology: *macro expansion*
- Later use: "algebraic compiler" to mean *translator*

# A history of the history of …

- Bauer: *Historical remarks on compiler construction* (1974) Bauer:1974:HistoricalRemarks
  - Many important references to early papers
  - USSR addendum by Ershov in 2nd printing (1976) Ershov:1976:Addendum
  - Opening quote:

> D. E. KNUTH [81] has observed (in 1962!) that the early history of cimpiler construction is difficult to assess. Maybe this, or maybe the general unhistorical attitude of our century is responsible for the widespread ignorance about the origins of compiler construction. In addition, the overwhelming lead of the USA in the general de-

# Some older histories of ...

Knuth:1962:AHistory

- Knuth: *A history of writing compilers* (1962)
  - Few references, names and dates, mostly US:

> A complete bibliography of the compiler literature is hard to give; you may, in fact, find it quite distressing to try to read many of the articles.

> A true history gives dates of events and names of people, but I will not do that. In our field there has been an unusual amount of parallel discovery of the same technique by people working independently. Perhaps

  - Knuth later regretted this attitude Knuth:1977:TheEarly
- Jones: *A survey of automatic coding techniques for digital computers*, MIT 1954 ! Jones:1954:ASurvey
- Randell and Russell 1964, par. 1.2 and 1.3 Randell:1964:Algol60Implementation
- Rosen: *Programming Systems and Languages*, 1964 and 1972 Rosen:1964:ProgrammingSystems Rosen:1972:ProgrammingSystems

# Knuth 1977:
# *The early development ...*

Knuth:1977:TheEarly

| Language | Principal author(s) | Year | Arithmetic | Implementation | Readability | Control structures | Data structures | Machine independence | Impact | First |
|---|---|---|---|---|---|---|---|---|---|---|
| Plankalkül | Zuse | 1945 | X, S, F | F | D | B | A | B | C | Programming language, hierarchic data |
| Flow diagrams | Goldstine & von Neumann | 1946 | X, S | F | A | D | C | B | A | Accepted programming methodology |
| Composition | Curry | 1948 | X | F | D | C | D | C | F | Code generation algorithm |
| Short Code | Mauchly | 1950 | F | C | C | F | F | B | D | High-level language implemented |
| Intermediate PL | Burks | 1950 | ? | F | A | D | C | A | F | Common subexpression notation |
| Klammerausdrücke | Rutishauser | 1951 | F | F | B | F | C | B | B | Simple code generation, loop expansion |
| Formules | Böhm | 1951 | X | F | B | D | C | B | D | Compiler in own language |
| AUTOCODE | Glennie | 1952 | X | C | C | C | C | D | D | Useful compiler |
| A-2 | Hopper | 1953 | F | C | D | F | F | C | B | Macroexpander |
| Whirlwind translator | Laning & Zierler | 1953 | F | B | A | D | C | A | B | Constants in formulas, manual for novices |
| AUTOCODE | Brooker | 1954 | X, F | A | B | D | C | A | C | Clean two-level storage |
| ПП-2 | Kamynin & Lîûbimskiĭ | 1954 | F | B | C | D | C | B | D | Code optimization |
| ПП | Ershov | 1955 | F | B | B | C | C | B | C | Book about a compiler |
| BACAIC | Grems & Porter | 1955 | F | A | A | D | F | A | D | Use on two machines |
| Kompiler 2 | Elsworth & Kuhn | 1955 | S | C | C | D | C | C | F | Scaling aids |
| PACT I | Working Committee | 1955 | X, S | A | C | D | C | A | C | Cooperative effort |
| ADES | Blum | 1956 | X, F | D | D | B | C | A | F | Declarative language |
| IT | Perlis | 1956 | X, F | A | B | C | C | A | A | Successful compiler |
| FORTRAN I | Backus | 1956 | X, F | A | A | C | C | A | A | I/O formats, comments, global optimization |
| MATH-MATIC | Katz | 1956 | F | B | A | C | C | A | D | Heavy use of English |
| Patent 3,047,228 | Bauer & Samelson | 1957 | F | D | B | D | C | B | C | Formula-controlled computer |

X=int, F=float, S=scaled          A ... F = much ... little

# Other substantial secondary sources

- Naur: *The replies to the AB14 questionnaire*
  - Survey of Algol compiler construction June 1962

TABLE OF ACTIVITY REPORTS AND TRANSLATOR SPEEDS.

| Brief name of group | No. of mem- bers | 5: Man years | Teaching | Progr. | Impl. | 12: Frac- tion per- cent | 13, 14: Pages written | publ. | Name of machine or system, with times for 100, 1000, and 10 000 instructions in minutes | 100 | 1000 | 10 000 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | - percent - | | | | | | | | | |
| NPL, England | 1 | 0.1 | 100 | 0 | 0 | 0 | 0 | 7 | | | | |
| Zeiss, Germany | 1 | 3 | 70 | 20 | 10 | - | 10 | - | | | | |
| SMIL, Sweden | 2 | 0.2 | 85 | 10 | 5 | - | - | 1 | SMIL | 0.4 | 4 | - |
| Syst.Dev.Corp., USA | 3 | (75) | 10 | 60 | 30 | Note 1 | | 0 | JOVIAL | 0.2 | 2 | 20 |

Naur:1962:Questionnaire

Naur:1962:TheReplies

- Bromberg: *Survey of programming languages and processors* (March 1963)

Bromberg:1963:SurveyOf

| LANGUAGE | MANUAL: IDENTIFICATION | DATE | PP | PUBLISHER | TRANSLATOR: CONSTRUCTOR | MACHINE | 1 | RUN | 2 | 3 | 4 | MINIMUM CONFIGURATION | SOURCE OF INFORMATION AND VERIFICATION | NOTES |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| COBOL NARRATOR | 95-05-000 | DEC 60 | 161 | • | RCA | RCA 501 | 65 | SEP 60 | G | A | B | 16K CHAR, 6 TAPES, RDR, OFFLINE PRNTR | BROMBERG, H | BASED ON COBOL 60. PRELIM. MANUAL MAY 60. SEE ALSO 95-05-002(96PP), 95-05-003(32PP) |
| COBOL NARRATOR | 93-05-002 | FEB 62 | 208 | • | RCA | RCA 301 | 45 | OCT 62 | G | A | B | 20K CHAR, 6 TAPES, RDR, ONLINE PRNTR | BROMBERG, H | ALL OF REQ. COBOL 61 + SOME ELECTIVE. |
| COBOL NARRATOR | | APR 62 | 250 | ENGLISH ELECT | R.C.A. | ENGLISH ELECT. KDP10 | | OCT 61 | G | A | B | 66K, 6 TAPES, P TAPE, PRNTR | DUNCAN, FG | BASED ON COBOL 60 |
| COBOL | | | | | SPERRY-RAND | UNIVAC II | 480 | OCT 60 | G | A | B | 2K, 12 TAPES | | |
| COBOL 60 | | | | | SPERRY-RAND | USS 80 | 15 | APR 61 | G | A | B | 2 TAPES, DRUM, RDR, PCH, PRNTR | | |
| COBOL | 3166/1 | OCT 60 | 143 | • | I.C.T. | ICT 1301 | | | | | B | MARK I, 2K, 12K DRUM | ELLIS, PV | BASED ON COBOL 60 (RAPIDWRITE VERSION IN MANUAL P155, MAY 61) |
| | | | | | I.C.T. | ICT 1301 | | | | | B | MARK II, 2K, 4 TAPES | | |
| COBOL | | | | PHILCO | COMP. SCIENCES | PHILCO 2000 | | | | | B | 8K, 6 TAPES | GUERNACCINI, J | BASED ON COBOL 61 |
| COBOL | 5000-21002-P | SEP 61 | 45 | • | BURROUGHS | B-5000 | | | | | B | | SPEIERMAN, KH | BASED ON COBOL 61, DUE DEC 62 |
| COBOL | | | | | ARMOUR RESEARCH | UNIVAC 1103A | 3.5 | | A | A | B | 8K, 16K DRUM, 8 TAPES | MITMAN, B | BASED ON COBOL 61, PLUS ELECTIVES. DUE DEC 62 |
| COBOL 61 | F-7411 | JUN 61 | 122 | NCR | NCR/GE | NCR(304)GE | 50 | DEC 61 | A | A | AB | 4.8 K, 6 TAPES, RDR, PRNTR | KEATING, WP | |

# Early interpreters and compilers

| Language | Machine | Operatio | Developer | Comp. size | Comp spee | Citation 1 |
|---|---|---|---|---|---|---|
| | EDSAC | Sep-1950 | Wilkes, Wheeler, Gill | | | Wilkes:1951:ThePreparation |
| Speedcode | IBM 701 | Sep-1953 | Backus | | | Backus:1954:TheIbm |
| A-2 | Univac I | Nov-1953 | Hopper | | | Knuth:1977:TheEarly |
| Autocode | Mark I | Dec-1955 | Brooker | | | Knuth:1977:TheEarly |
| IT | IBM 650 | Oct-1956 | Perlis | | | Bromberg:1963:SurveyOf |
| Flow-Matic | Univac I | Dec-1956 | Hopper | | | Bromberg:1963:SurveyOf |
| Fortran I | IBM 704 | Jun-1957 | Backus et al | 24000 ins | 8 cards/min | Backus:1957:TheFortran |
| Alfakod | BESK | Nov-1957 | Riesel, Jonason, von Sydow | | | Lundin:2006:TidigProgramme |
| MAC | Ferranti Mercu | Dec-1957 | Dahl | | | AMS:1958:MathematicalTable |
| Fortran II+III | IBM 704 | May-1958 | Backus et al | | | Backus:1959:AutomaticProgr |
| Runcible | IBM 650 | Dec-1958 | Knuth | | | Knuth:1959:Runcible |
| Algol 58 | Zuse Z22 | Dec-1958 | Bauer, Samelson | | | Samelson:1960:SequentialFo |
| GAT | IBM 650 | Feb-1959 | Arden, Graham | | | Arden:1959:OnGat |
| Neliac | Univac M-460 | Mar-1959 | Halstead | | | Huskey:1959:Neliac |
| Algol 58 | B 220 | Dec-1959 | Barton | 3500 ins | 500 instr/min | Barton:1961:AnotherNameles |
| Lisp 1 | IBM 704 | Jan-1960 | McCarthy | | | McCarthy:1960:RecursiveFun |
| Algol 60 | X-1 | Jun-1960 | Dijkstra | 2500 words | | Dijkstra:1960:RecursiveProgr |
| COBOL | RCA 501 | Sep-1960 | Bromberg | | | Bromberg:1963:SurveyOf |
| Algol 58 | B 205 | Sep-1960 | Knuth | 4000 words | 45 cards/min | Knuth:1960:TheInternals |
| COBOL | Univac II | Oct-1960 | | | | Bromberg:1963:SurveyOf |
| Algol 60 | CDC 1604 | Jun-1961 | Irons | 800 ins/10000 tbl | 300 instr/s | Irons:1961:ASyntax |
| Algol 60 | Zuse Z22 | Jul-1961 | Bauer | | | Bromberg:1963:SurveyOf |
| Algol 60 | DASK | Aug-1961 | Jensen, Naur | | | Jensen:1961:AnImplementati |
| Algol 60 | Facit EDB | Oct-1961 | Dahlstrand | | | Dahlstrand:2009:Minnen |
| Algol 60 | Elliott 503 | Feb-1962 | Hoare | 8000 ins | 1000 char/s | Hoare:1962:ReportOn |
| Algol 60 | Gier | Sep-1962 | Jensen, Naur | 4000 words | 30 instr/s | Naur:1963:TheDesign1 |
| Algol 60 Whet | KDF9 | Sep-1962 | Randell, Russe | 3000 words | input limited | Randell:1964:Algol60Implem |
| Algol 60 Kidsg | KDF9 | Dec-1962 | Hawkins, Huxt | 20000 words | | Randell:1964:Algol60Implem |
| Algol 60 | M-20 | Jan-1964 | Ershov | 45000 words | | Ershov:1966:Alpha |
| Simula I | Univac 1107 | Jan-1965 | Dahl, Nygaard | | | Dahl:1966:Simula |

# Knuth's B 205 Algol 58 compiler

- Developed June-Sep 1960, at age 22
- Idiosyncratic documentation:

Knuth:1960:TheInternals

Waychoff:1979:StoriesAbout

Knuth:2007:OralHistory

How does this compiler work? Frankly, it's a miracle if it does.

The style of presentation is adapted from the practice of computer publications in the U.S.S.R.: the flowchart boxes

call scanners (routine 0) which operate from A1 and A2. The Rube Goldberg procedure call scanner actually begins by operation

........ and that's all there is to this compiler.

- No description of runtime state or its invariants

# Problems and solutions

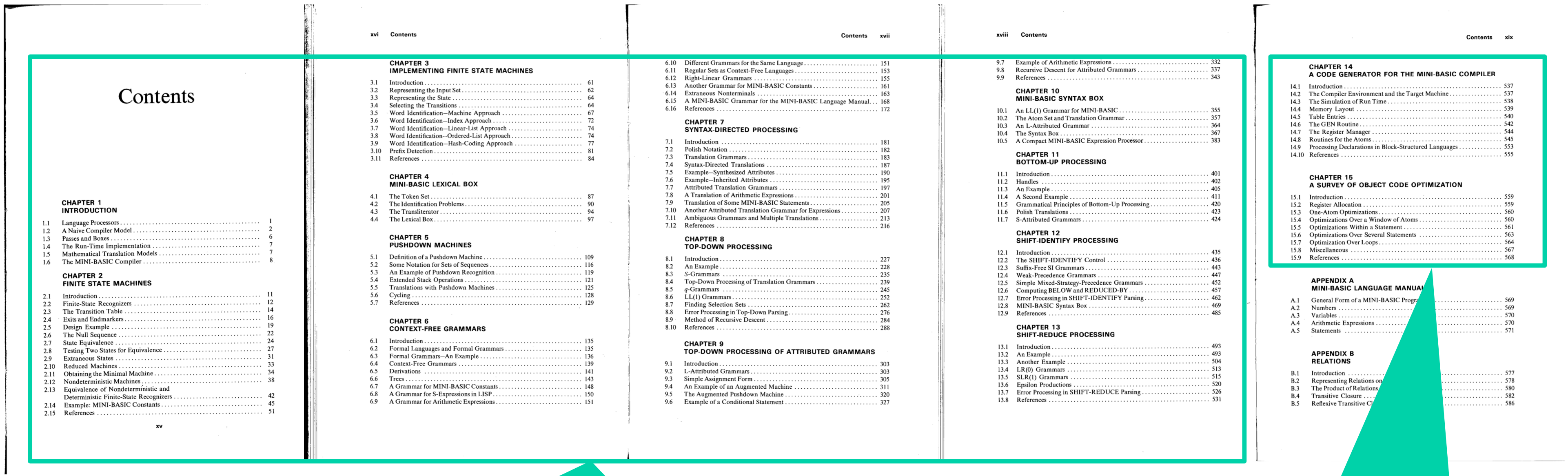
- Lexing and parsing: from text to internal representation
  - Arithmetic operators, precedence, parentheses
- Compilation of expressions
  - To postfix, reverse Polish form
- Storage allocation
  - Runtime state invariants, code to maintain them
- Optimization
  - How to generate efficient code
- Flow analysis
  - How discover program structure and invariants

# History: lexing and parsing

- Initially ad hoc
- Table-driven/automata methods

  Samelson:1960:SequentialFormula
  Irons:1961:ASyntax
  Naur:1963:TheDesign1
- Regular expressions, context-free grammars
- Finite state automata and pushdown automata
- Knuth LR parsing 1965 Knuth:1965:OnThe
- Gries operator grammars 1968 Gries:1968:UseOf
- Lexer and parser generator tools
  - Lex (Lesk 1975) and Yacc (Johnson 1975)
  - LR dominated for a while
  - LL back in fashion: Antlr, Coco/R, parser combinators, packrat parsers

# Lewis, Rosenkrantz, Stearns: *Compiler design theory*, 1976



Contents





















500 pages about lexing and parsing

30 pages **not** about lexing and parsing

- Historically, too much emphasis on parsing?
  - Because it was formalizable and respectable?
  - But also beautiful relations to complexity and computability …

# History: compilation of expressions

- Rutishauser 1952 (unimpl.) Rutishauser:1952:AutomatischeRechenplanfertigung
  - Translating arithmetic expressions to 3-addr code
  - Infix operators, precedence, parentheses
  - Repeated scanning and simplification
- Böhm 1952 (not impl.) Knuth:1977:TheEarly
  - Similar – also at ETH Zürich
- Fortran I, 1957 Sheridan:1959:TheArithmetic
  - Baroque but simple treatment of precedence (Böhm &)
  - Complex, multiple scans, both left-right and right-left
- Samelson and Bauer 1960 Samelson:1960:SequentialFormula
  - One scan, using a stack ("cellar") at translation time
- Floyd 1961 Floyd:1961:AnAlgorithm
  - One left scan, one right scan, optimized code

# History: Compilation techniques

- Single-pass table-driven with stacks
  - Bauer and Samelson for Alcor
  - Dijkstra 1960, Algol for X-1 — Dijkstra:1961:Algol60Translation
  - Randell 1962, Whetstone Algol — Randell:1964:WhetstoneAlgol
- Single-pass recursive descent — Hoare:1962:ReportOn
  - Hoare 1962, one procedure per language construct
- Multi-pass ad hoc
  - Fortran I, 6 passes — Backus:1957:TheFortran
- Multi-pass table-driven with stacks
  - Naur 1962 GIER Algol, 9 passes — Naur:1963:TheDesign2
  - Hawkins 1962 Kidsgrove Algol
- General syntax-directed table-driven
  - Irons 1961 Algol for CDC 1604 — Irons:1961:ASyntax

# Multi-pass compilation is still used

- Good for small-memory systems (eg GIER)
- Still used, now for separation of concerns:

21 internal passes
of the Scala
compiler (2013)

namerFactory
typerFactory
superAccessors
pickler
refchecks
liftcode
uncurry
tailCalls
explicitOuter
erasure
lambdaLift
constructors
flatten
mixer
cleanup
genicode
inliner
inlineExceptionHandlers
closureElimination
deadCode
genJVM

# History: Run-time organization

- Early papers focus on *translation*
  - Runtime data managent is trivial, eg. Fortran I
- Algol: *runtime storage allocation* is essential
- Dijkstra: Algol for X-1 (1960) Dijkstra:1960:RecursiveProgramming
  - Stack of procedure activation records
  - Display, to access variables of enclosing scopes
- Also focus of Naur's Gier Algol papers Naur:1963:TheDesign1 Naur:1963:TheDesign2

- Design a runtime state structure (invariant)
- Compiler should generate code that
  - Can rely on the runtime state invariant
  - Must preserve the runtime state invariant

Diskussion om ALGOL vid symposiet i Rom i mars 1962. Sittande fr.v. P. Naur, Danmark, S. Moriguti, Japan, och A. van Wijngaarden, Nederländerna. Stående E. W. Dijkstra, Nederländerna.

Ekman and Fröberg 1972, p. 127

# History: Target architectures

- EDSAC, IAS machine, BESK
  - No index register, so self-modifying code for arrays
  - Subtle, and makes manual code relocation painful
- Index registers, IBM 704, DASK, ...
  - Simpler and faster code, position-independent
  - IBM 704 at-least-once loops impacts Fortran DO
- Stack machines

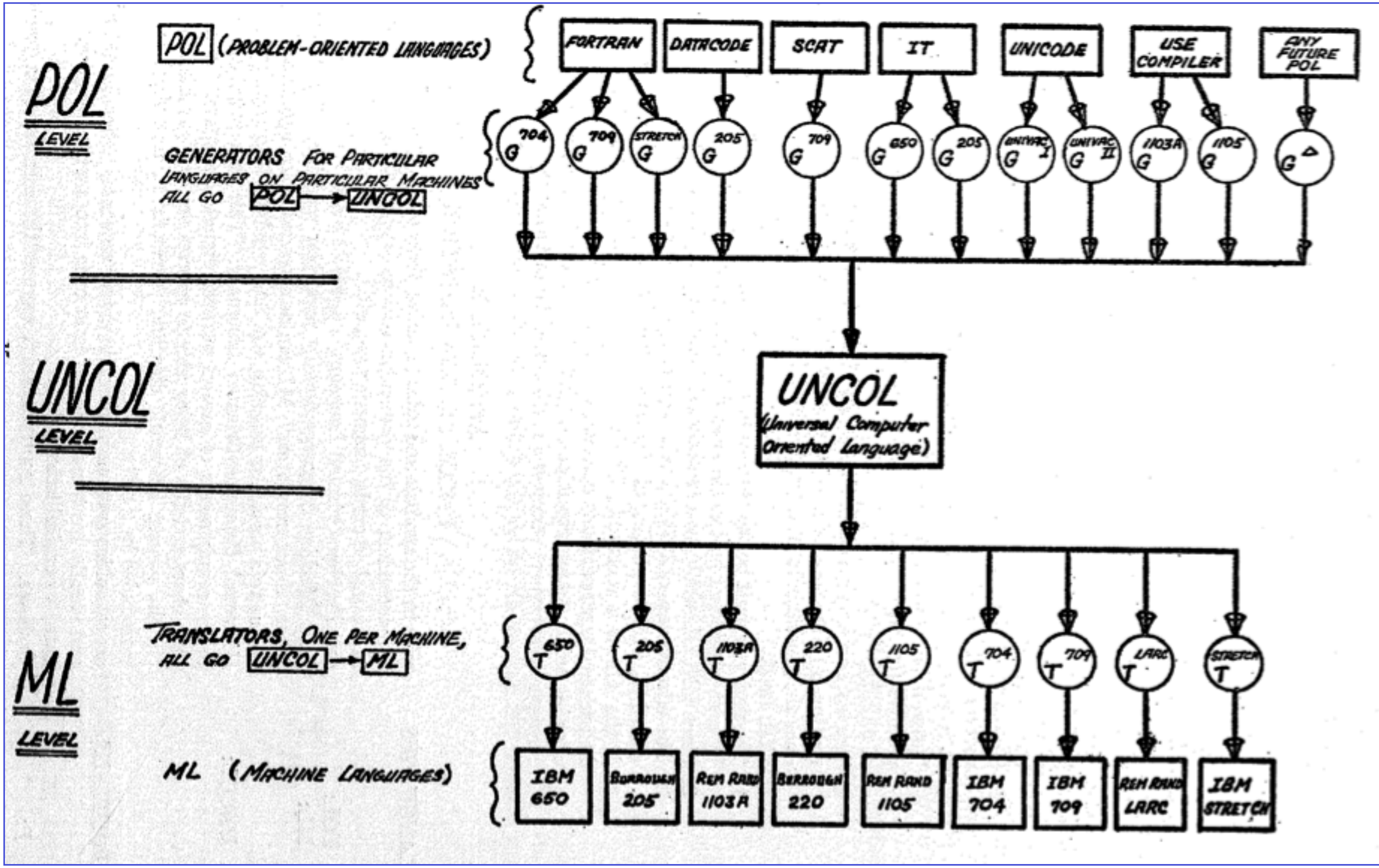
  Hamblin:1962:TranslationTo

  Dijkstra:1960:RecursiveProgramming Barton:1961:ANew
  - Conceptual: Samelson, Hamblin, Dijkstra, Barton
  - Hardware: Burroughs B5000, KDF9 (arith, return)
  - Compile to reverse Polish by post-order traversal
  - Java and .NET/CLI intermediate languages

# History: Intermediate languages

Strong:1958:TheProblem

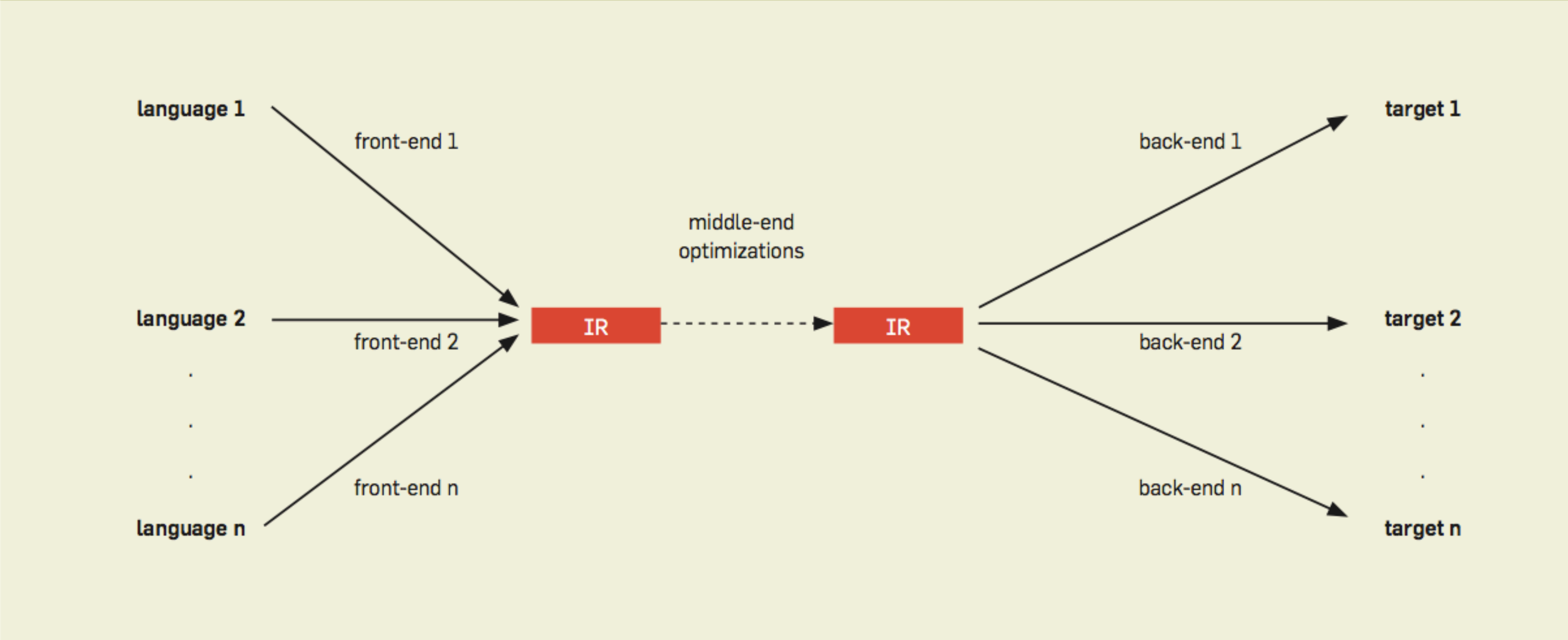
- Strong: *The problem of ...*, February 1958



# Everything old is new again

- Chow: *Intermediate representation*, CACM December 2013 Chow:2013:IntermediateRepresentation

Figure 2. A compiler system supporting multiple languages and multiple targets.



# UNCOL is finally coming true

- Java Virtual Machine, 1994
  - Developed as bytecode for Java only
  - Yet used for Scala, Clojure, Jython, Ceylon, JRuby...
  - Runtime system, libraries, on many CPUs and OSs
- .NET Common Language Infrastructure, 1999
  - C#, VB.NET, F#, JScript, Eiffel, COBOL, IronPython
- JavaScript/EcmaScript, 1994
  - For browser scripting, thus on all user devices
  - Ceylon, Dart, Typescript, F#, ...
- LLVM = Low-Level Virtual Machine, 2003
  - Static single assignment compiler-internal form
  - Clang C/C++/Objective-C, Nvidia CUDA, Mono, ...

# What does a C# compiler do

```
for (int i=0; i<n; i++)
  sum += sqrt(arr[i]);
```

C# language source program

csc /o

```
0b stloc.1              // i = 0
0c br.s 1d              // goto 1d
0e ldloc.0              // sum
0f ldarg.1              // arr
10 ldloc.1              // i
11 ldelem.r8            // arr[i]
12 call Math::Sqrt      // sqrt
17 add                  // sum + ...
18 stloc.0              // sum
19 ldloc.1              // i
1a ldc.i4.1             // 1
1b add                  // i + 1
1c stloc.1              // i = ...
1d ldloc.1              // i
1e ldarg.0              // n
1f blt.s 0e             // if i<n loop
```

.NET/CLI bytecode

runtime system's
just-in-time compiler

```
19 xorl %ebx,%ebx                  // i = 0
1b jmp 3a                          // goto 3a
1d leal 0x00(%ebp),%ebp
20 fldl 0xec(%ebp)                 // sum
23 cmpl %ebx,0x0c(%edi)            // array index check
26 jbe 49                          // if outofbounds, throw
2c leal 0x10(%edi,%ebx,8),%eax     //
30 fldl (%eax)                     // arr[i]
32 fsqrt                           // sqrt
34 faddp %st,%st(1)                // sum + ...
36 fstpl 0xec(%ebp)                // sum = ...
39 incl %ebx                       // i++
3a cmpl %esi,%ebx
3c jl 20                           // if i<n, loop
```

x86 machine code

.NET/CLI bytecode:
- Stack-based
- Loadtime checks, types, local stack
- Runtime checks, array bounds, null references, ...
- Dynamic compilation to real machine code

# History: Type checking

- Naur GIER Algol 1965 Naur:1965:CheckingOf
  - "*pseudoevaluation of the expressions*"
  - "*like a run-time evaluation but works with descriptions of the types and kinds of operands instead of with values*"
- Damas and Milner, ML 1982 Damas:1982:PrincipalTypeschemes
  - Inference of polymorphic type schemes
  - Generalization and specialization
  - Unification (Robinson) to solve type equations
  - Has influenced Haskell, C#, F#, Scala, ...

# Compiler quality attributes

- Produces fast target code
- Produces target code fast
- Checks source code syntax
- Checks source code types and consistency
- Provides precise and clear error messages
- Reports as many errors as possible
- Is itself free of errors
- Does not report spurious errors
- Provides frugal compilation or recompilation or separate compilation

# History: Diagnostics

- EDSAC (Wilkes 1951)
  - Tracing and post-mortem dumps
- GIER Algol
  - Comprehensive compiler error messages
  - Not a focus in eg Hoare's Elliot Algol compiler
  - "*GIER ... had an excellent compiler-cum-operating system*" (Sanders in HiNC 2003)
  - "*the very successful Algol compiler ... for the GIER computer*" (Randell & Russell 1964)

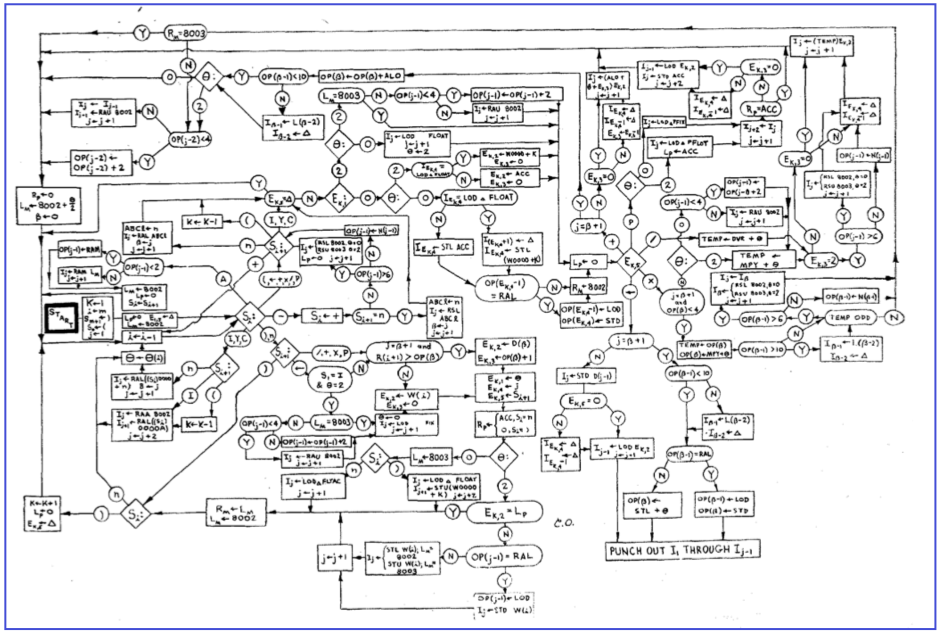
  - (Presumably also important that it was thoroughly tested, Naur:1963:TheDesign2 page 163)

# History: Bootstrapping

- Writing a compiler in the language it compiles
- Runcible 1959

Knuth:1959:Runcible

  - Runcible expression compilation chart, in Runcible



Huskey:1959:Neliac

Masterson:1960:CompilationFor

- Neliac compiler written in Neliac 1959
  - and self-compilation as correctness check

Thanks to Robert Glück for references

# C compiler optimizations

```
for (int i=0; i<n; i++)
  sum += sqrt(arr[i]);
```

C language

clang

```
LBB0_1:
movl    -28(%rbp), %eax         // i
movl    -4(%rbp), %ecx          // n
cmpl    %ecx, %eax
jge     LBB0_4                  // if i >= n, return
movslq  -28(%rbp), %rax         // i
movq    -16(%rbp), %rcx         // address of arr[0]
movsd   (%rcx,%rax,8), %xmm0    // arr[i]
callq   _sqrt                   // sqrt
movsd   -24(%rbp), %xmm1        // sum
addsd   %xmm0, %xmm1            // sum + ...
movsd   %xmm1, -24(%rbp)        // sum = ...
movl    -28(%rbp), %eax         // i
addl    $1, %eax                // i + 1
movl    %eax, -28(%rbp)         // i = ...
jmp     LBB0_1                  // loop again
```

x86 machine code

clang -O3

```
LBB0_2:
movsd   (%rsi), %xmm1           // *p same as arr[n-i]
sqrtsd  %xmm1, %xmm1            // sqrt
addsd   %xmm1, %xmm0            // sum += ...
addq    $8, %rsi                // p++
decq    %rax                    // i--
jne     LBB0_2                  // if i!=0 loop again
```

- Register allocation (sum, i, n, arr)
- Inlining of functions
- Index calculations
- Iteration variable elimination (i)

# History: Optimization

- Fortran I in 1957 has — Backus:1957:TheFortran
  - common subexpression elimination
  - fast index computations: reduction in strength
  - clever allocation of index registers
  - constant folding
- Samelson & Bauer — Samelson:1960:SequentialFormula
  - algorithm for fast index computations (red.stren.)
- Allen 1969, algorithms for: — Allen:1969:ProgramOptimization
  - basic blocks, flow graph of strongly conn. comps.
  - constant folding
  - common subexpression elimination
  - invariant code moving
  - reduction in strength
  - test replacement (for loops after red. strength)

# History: Flow analysis

- Fortran I was amazingly ambitious <u>Backus:1957:TheFortran</u>
  - Control flow analysis graph with edge frequencies
  - Monte Carlo simulation at compiletime based on FREQUENCY statements
- Allen, Cocke 1970 <u>Allen:1970:ControlFlow</u>
  - Control flow graph, dominator, interval, reducible graph, ...
  - Left out of Bauer:1974:HistoricalRemarks
- Cousot and Cousot 1977
  - Abstract interpretation
  - Based on formal semantics
  - Systematic approximation via Galois connections
  - Termination via widening operators

# Some topics not covered at all

- Compiler generation, compiler-compilers
- Adaptive compilation
  - just-in-time compilers for Java, .NET/CLI
- Compilation techniques for
  - lazy functional languages: Haskell
  - object-oriented languages: Java, C#
  - dynamic languages: Smalltalk, Javascript (v8)
  - extensible languages: Lua
- Compilation for parallel architectures
- Continuation-based compilation
- …

# A thought on hardware and language developments 2010-2020

- First languages were made to resemble machines
- Then Algol made the opposite happen (stacks)
- Today: multicore, manycore, massive parallelism
  - Multiple levels memory and caches
  - Complex memory consistency models
  - Computation is cheap, moving data is expensive
  - Still too much focus on scientific computing? (exascale)
- Are future machines designed with enough input from programming language design?
- What are the language features suitable for parallelism and concurrency?

# Interesting reading

- Secondary sources
  - Knuth:1977:TheEarly
  - Bauer:1974:HistoricalRemarks
  - Ershov:1976:Addendum
- Primary sources
  - Backus:1957:TheFortran
  - Samelson:1960:SequentialFormula
  - Dijkstra:1960:RecursiveProgramming
  - Hoare:1962:ReportOn
  - Naur:1963:TheDesign1
  - Naur:1965:CheckingOf
  - Randell:1964:Algol60Implementation

# A collection of sources

| | Bibtex and link | Author | R | Title | Notes, message | Category | Source |
|---|---|---|---|---|---|---|---|
| 3 | | | | | | | |
| 4 | **Bibtex and link** | **Author** | **R** | **Title** | **Notes, message** | **Category** | **Source** |
| 5 | AlgolBulletin | | | Algol Bulletin | | compilers | At http://archive.comp |
| 6 | Allen:1969:ProgramOptimi | Allen | | Program optimization | | static analysis | Annual review in auton |
| 7 | Allen:1970:ControlFlow | Allen | | Control flow analysis | | static analysis | ACM Sigplan Notices 5, |
| 8 | AMS:1958:MathematicalTa | American Mathematica | | Review of Dahl:1957:AutocodingFor | | | Mathematical Tables ar |
| 9 | Arden:1959:OnGat | Arden, Graham | | On GAT [Generalized Algebraic Translator] and the construction of translators | | compilers | CACM 2, 7 (July 1959) |
| 10 | Asker:2005:AlgolGenius | Asker | | Algol-Genius: An early success for high-level languages | | languages | Bubenko:2005:History |
| 11 | Backus:1954:TheIbm | Backus | | The IBM 701 Speedcoding syst | Apparently no compilation process, input is three-adde | interpreters | JACM 1, 1 (1954) 4-6 |
| 12 | Backus:1957:TheFortran | Backus et al | | The FORTRAN automatic codin | About Fortran I, early 1957; only simple arithmetic fun | languages | Western computer proc |
| 13 | Backus:1959:AutomaticPro | Backus | | Automatic programming: Prope | About Fortran II; adds general procedures and functior | compilers | In NPL:1959:Mechanis |
| 14 | Barton:1961:ANew | Barton | | A new approach to the functior | Algol 60 inspired machine design with stack S and regi | hardware | AFIPS Conference Proc |
| 15 | Barton:1961:AnotherName | Barton | | Another (nameless) compiler fc | Some statistics on the Burroughs 220 Algol-58/60 com | compilers | CACM 4, 1 (January 19 |
| 16 | Bauer:1957:VerfahrenZur | Bauer, Samelson | | Verfahren zur automatischen V | Machine (hardware) patent on formula evaluation usin | compilers | depatisnet.dpma.de |
| 17 | Bauer:1959:SequentielleFo | Bauer, Samelson | | Sequentielle formelübersetzung | | compilers | Elektronische Rechenar |
| 18 | Bauer:1974:HistoricalRema | Bauer | rere | Historical remarks on compiler | With bibliography (which misses Allen, Cocke) | compilers | Bauer et al: Compiler c |
| 19 | Bemer:1969:PoliticoSocial | Bemer | | A politico-social history of Algol | | | Annual Review of Autor |
| 20 | Bergin:2007:AHistory | Bergin | | A history of the history pf programming languages | | languages | CACM |
| 21 | Bratman:1961:AnAlternate | Bratman | | An alternate form of the UNCO | First appearance of T diagrams | compilers | CACM 4, 3 (March 196 |
| 22 | Bromberg:1963:SurveyOf | Bromberg | | Survey of programming languages and processors | | compilers | CACM 6, 3 (March 196 |
| 23 | Bubenko:2005:HistoryOf | Bubenko, Impagliazzo, | | History of Nordic computing 2003 | | systems | Springer 2005 |
| 24 | Carr:1959:AVisit | Carr et al | | A visit to the computation cent | Mentions an automatic programming system in Moscow AS developed | | CACM 2, 6 (1959) 8-20 |
| 25 | Chow:2013:IntermediateRe | Chow | | Intermediate representation | | compilers | CACM 56, 12 (Decemb |
| 26 | Conway:1958:ProposalFor | Conway | | Proposal for an UNCOL | A concrete proposal for the UNCOL concept suggested | intermediate | CACM 1, 10 (1958) |
| 27 | Dahl::APlea | Dahl | | A plea for multiprogramming | | concurrency | Algol Bulletin 24 |
| 28 | Dahl:1957:AutocodingFor | Dahl | | Autocoding for the Ferranti Mercury Computer (MAC) | | compilers | NDRE Report 24, 1957 |
| 29 | Dahl:1966:Simula | Dahl, Nygaard | | Simula, an Algol-based simulation language | | compilers | CACM 9, 9 (September |
| 30 | Dahl:1970:CommonBase | Dahl, Myhrhaug, Nyga | | Common Base Language | | languages | Simula Information S-2 |
| 31 | Dahl:2002:TheBirth | Dahl | | The birth of object orientation: the Simula languages | | languages | At http://www.olejohar |
| 32 | Dahlstrand:2009:Minnen | Dahlstrand | | Ingemar Dahlstrands Minnen | Wrote Algol 60 compiler for BESK and Facit; see pages | compilers | At http://www.tekniska |
| 33 | Damas:1982:PrincipalType | Damas, Milner | | Principal type-schemes for functional programs | | types | POPL 1982, 207-212 |
| 34 | Dijkstra:1960:RecursivePro | Dijkstra | | Recursive programming | Joint (software) stack for expression evaluation and re | runtime? | Numerische Mathemati |
| 35 | Dijkstra:1961:Algol60Trans | Dijkstra | | Algol 60 translation. An Algol 6 | Compiler and runtime organization, basically stack of a | compilers | Algol Bulletin, suppl 10 |
| 36 | Dijkstra:1963:MakingA | Dijkstra | | Making a translator for Algol 60 | Slightly generalized presentation of Dijkstra:1961:AnAlgol60 | | Annual Review of Autor |
| 37 | Duncan:1962:Implementat | Duncan | | Implementation of Algol for the English Electric KDF9 | | compilers | Computer Journal 5, 2 |
| 38 | Endres:2013:EarlyLanguag | Endres | | Early language and compiler developments at IBM Europe: a personal retrospection | | history | AHC 2013 |
| 39 | Ershov:1958:OnProgramm | Ershov | | On programming of arithmetic operations | | compilers | CACM 1, 8 (1958) 3-6 |
| 40 | Ershov:1959:AutomaticPro | Ershov | | Automatic programming in the Soviet Union | | | Datamation (Jul-Aug 1 |
| 41 | Ershov:1959:Programming | Ershov | | Programming programme for the BESM computer | | | Pergamon Press 1959 |
| 42 | Ershov:1959:TheWork | Ershov | | The work of the computing centre of the Academy of Sciences of the USSR in the field of automatic programming | | | |
| 43 | Ershov:1966:Alpha | Ershov | | Alpha, an automatic programming system of high efficiency | | compilers | JACM 13, 1 (1966) 17- |
| 44 | Ershov:1976:Addendum | Ershov | | Addendum | | | In 2nd ed of Bauer:197 |

# Jeg har ikke kunnet finde ...

- Information om en Algol-oversætter til SMIL af Ekman og Robertson, Lund
- Information om en Algol-oversætter til Ferranti Mercury fra NDRE = Forsvarets Forskningsinstitut, Oslo, formentlig O.-J. Dahl