A history of (Nordic) compilers and autocodes

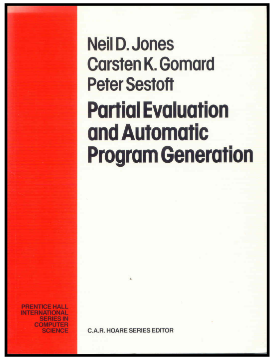
Peter Sestoft
sestoft@itu.dk

2014-10-13

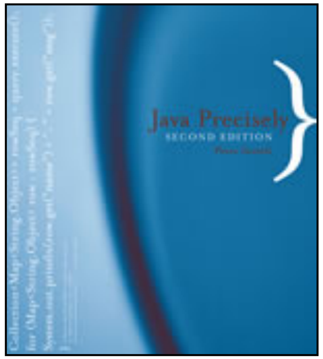
Copenhagen Tech Polyglot Meetup

The speaker

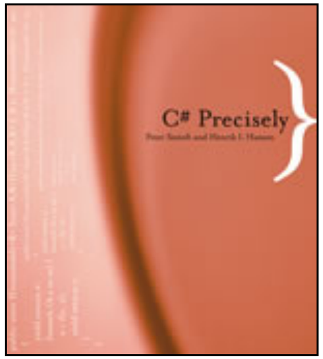
- MSc 1988 computer science and mathematics and PhD 1991, DIKU, Copenhagen University
- KU, DTU, KVL and ITU; and AT&T Bell Labs, Microsoft Research UK, Harvard University
- Programming languages, software development, ...
- Open source software
  - Moscow ML implementation, 1994…
  - C5 Generic Collection Library, with Niels Kokholm, 2006…
  - Funcalc spreadsheet implementation, 2014



1993

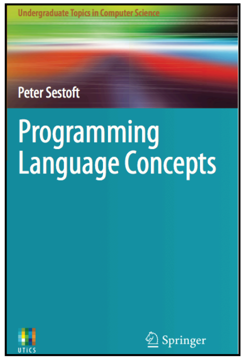


2002, 2005, 2015?

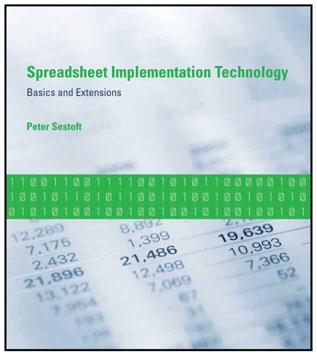


2004 & 2012

2007



2012



2014

Current obsession: a new ITU course

Practical Concurrent and Parallel Programming (PCPP) (PRCPP)

- This MSc course is about how to write correct and efficient concurrent and parallel software, primarily using Java, on standard shared-memory multicore hardware. It covers basic mechanisms such as threads, locks and shared memory as well as more advanced mechanisms such as transactional memory, message passing, and compare-and-swap. It covers concepts such as atomicity, safety, liveness and deadlock. It covers how to measure and understand performance and scalability of parallel programs. It covers tools and methods find bugs in concurrent programs.
- For exercises, quizzes, and much more information, see the course LearnIT site (restricted access).
- For formal rules, see the official course description.

Lecture plan

Course week	ISO week	Date	Who	Subject	Materials	Exercises
1	35	29 Aug	PS	Concurrent and parallel programming, why, what is so hard. Threads and locks in Java, shared mutable memory, mutual exclusion, Java inner classes.	Goetz chapters 1, 2; Sutter paper; McKenney chapter 2; Bloch item 66; Slides week 1; Exercises week 1; Example code: pcpp-week01.zip	Exercises week 1
2	36	5 Sep	PS	Threads and Locks: Threads for performance, sharing objects, visibility, volatile fields, atomic operations, avoiding sharing (thread confinement, stack confinement), immutability, final, safe publication	Goetz chapters 2, 3; Bloch item 15; Slides week 2; Mandatory exercises week 2; Example code: pcpp-week02.zip	Mandatory handin 1
3	37	12 Sep	PS	Threads and Locks: Designing thread-safe classes. Monitor pattern. Concurrent collections. Documenting thread-safety.	Goetz chapters 4, 5; Slides week 3; Exercises week 3; Example code: pcpp-week03.zip	Exercises week 3
4	38	19 Sep	PS	Performance measurements.	Sestoft: Microbenchmarks; Slides week 4; Exercises week 4; Example code: pcpp-week04.zip; Optional: McKenney chapter 3	Mandatory handin 2
5	39	26 Sep	PS	Threads and Locks: Tasks and the Java executor framework. Concurrent pipelines, wait() and notifyAll().	Goetz chapters 6, 8; Bloch items 68, 69; Slides week 5; Exercises week 5; Example code: pcpp-week05.zip	Exercises week 5
6	40	3 Oct	PS	Threads and Locks: Safety and liveness, absence of deadlock and livelock. The ThreadSafe tool.	Goetz chapter 10, 13.1; Bloch item 67; Slides week 6; Exercises week 6; Example code: pcpp-week06.zip	Mandatory handin 3
7	41	10 Oct	PS	Threads and Locks: Performance and scalability	Goetz chapter 11, 13.5; Slides week 7; Exercises week 7; Example code: pcpp-week07.zip	Exercises week 7
	42	17 Oct	Fall break			

http://www.itu.dk/people/sestoft/itu/PCPP/E2014/

The future is parallel – and functional

- Classic imperative for-loop to count primes:

```
int count = 0;
for (int i=0; i<range; i++)
  if (isPrime(i))
    count++;
```

i7: 9.9 ms
AMD: 40.5 ms

- Sequential functional Java 8 stream:

```
IntStream.range(0, range)
.filter(i -> isPrime(i))
.count()
```

i7: 9.9 ms
AMD: 40.8 ms

- Parallel functional stream:

```
IntStream.range(0, range)
.parallel()
.filter(i -> isPrime(i))
.count()
```

i7: 2.8 ms
AMD: 1.7 ms

i7: 3.6 x speedup
AMD: 24.2 x speedup

for free

Outline

- What is a compiler?
- Genealogies of languages and of early computers
- Knuth's survey of early autoprogramming systems
- Lexing and parsing
- Compilation of expressions
- FORTRAN I in the USA
- Algol 60 in Europe
- Early Nordic autocodes and compilers
- (Intermediate languages)
- (Optimization)
- (Flow analysis)
- (Type systems)
- (Compiler generators)
- The nuclear roots of object-oriented programming

What is a compiler? and autocode?

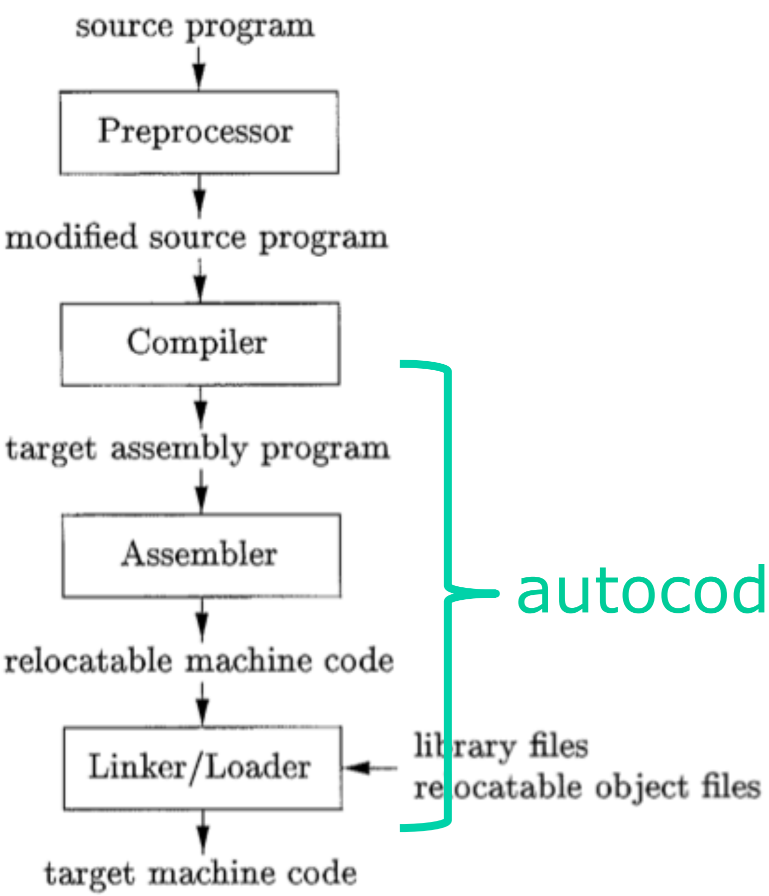
```c
for (int i=0; i<n; i++)
  sum += sqrt(arr[i]);
```

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
```

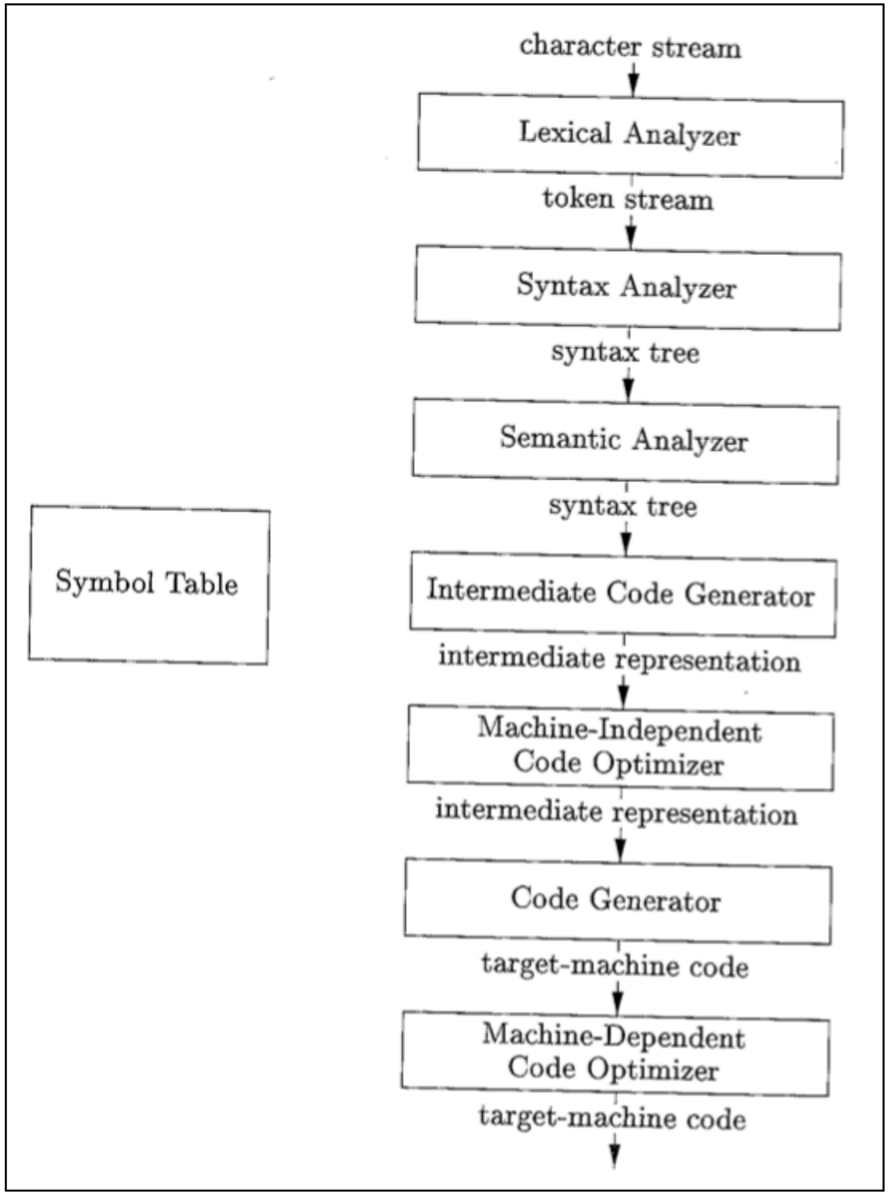
x86 machine code

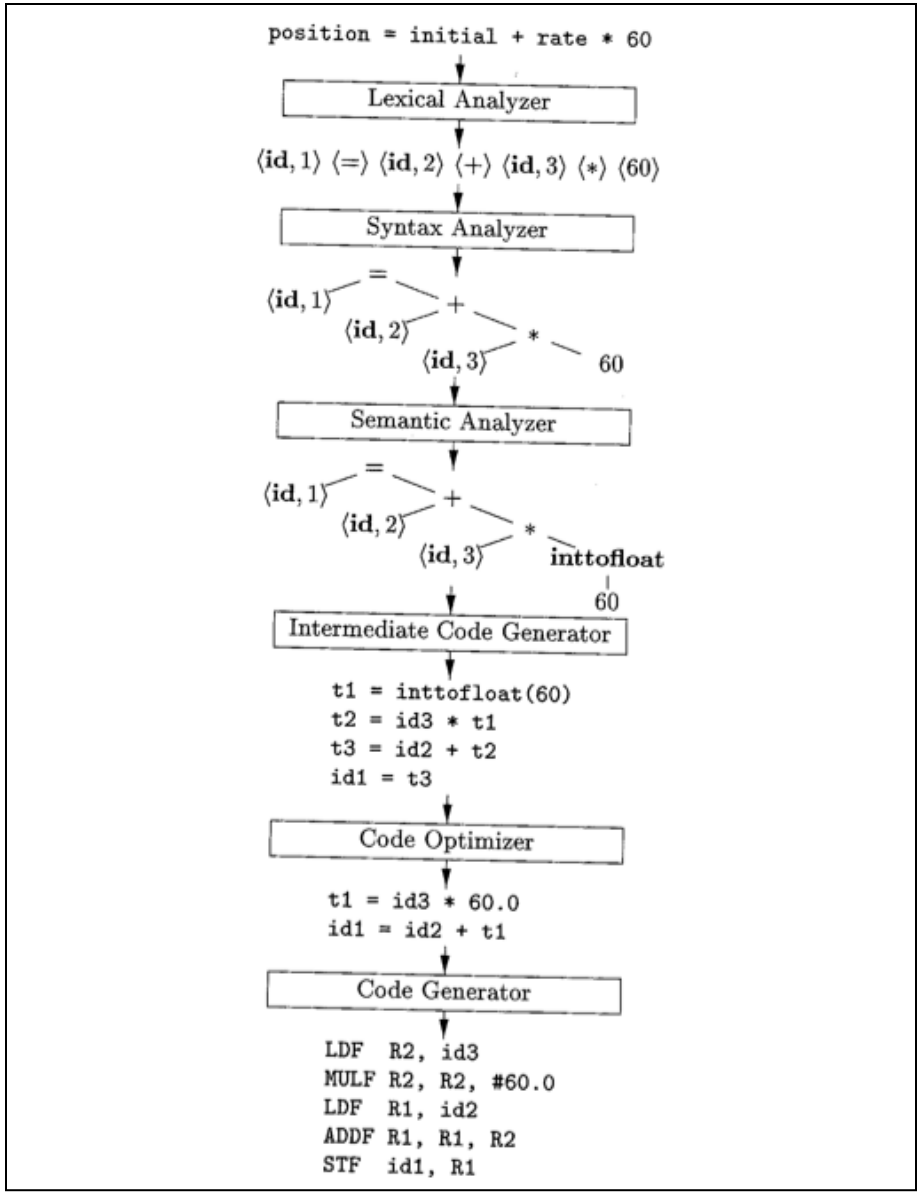


autocode (early compilers)

From Aho et al

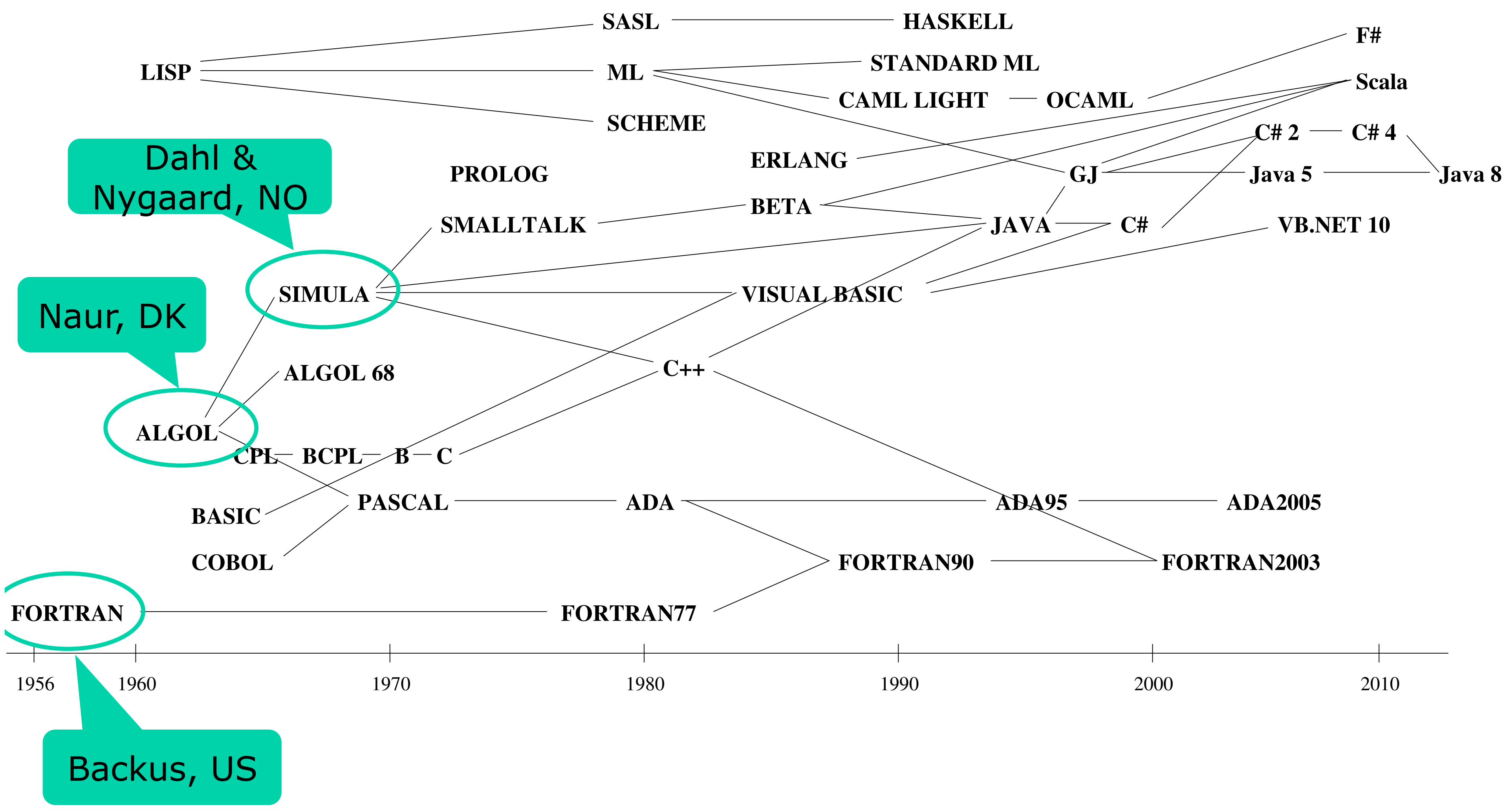
Conceptual phases of a compiler



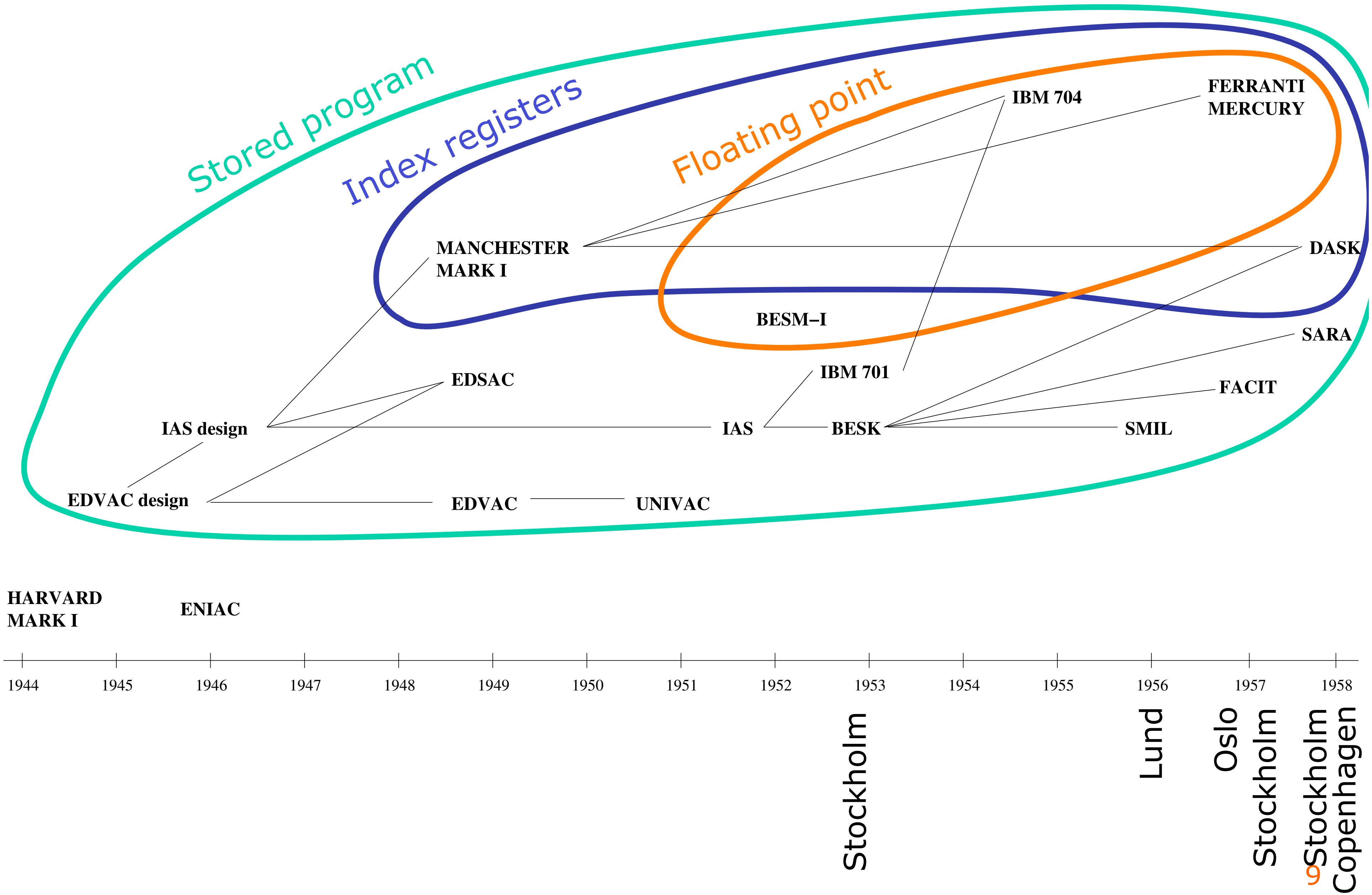


From Aho et al

Genealogy of programming languages



Genealogy of Nordic computers



Stored program computers

- Programs and data stored in the same way
  - EDVAC and IAS designs ("von Neumann") 1945
- So: program = data
- So a program can process another program
  - This is what a compiler or assembler does

- Also, a program can modify itself at runtime
  - Used for array indexing in IAS, EDSAC, BESK, ...
  - Used for subroutine return, EDSAC, the "Wheeler jump"

- Modern machines use *index registers*
  - For both array indexing and return jumps
  - Invented in Manchester Mark I, 1949
  - Adopted in the Copenhagen DASK 1958

A history of the history of …

- Fritz Bauer, Munich: *Historical remarks on compiler construction* (1974) Bauer:1974:HistoricalRemarks
  - Many references to important 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-

Bauer:1974:HistoricalRemarks

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.

- Jones: *A survey of automatic coding techniques for digital computers*, MIT 1954 Jones:1954:ASurvey
  - Also lists people interested in automatic coding
  - Only US and UK: Cambridge and Manchester

Knuth 1977:
The early development ...

Knuth:1977:TheEarly

Language	Principal author(s)	Year	Arith-metic	Imple-men-tation	Read-abil-ity	Control struc-tures	Data struc-tures	Machine indepen-dence	Im-pact	First
Plankalkül	Zuse	1945	X, S, F	F	D	B	A	B	C	Programming language, hierarchic d[illegible]
Flow diagrams	Goldstine & von Neumann	1946	X, S	F	A	D	C	B	A	Accept[illegible]ng
Composition	Curry	1948	X	F	D	C	D	C	[illegible]	[illegible]ithm
Short Code	Mauchly	1950	F	C	C	F	F	[illegible]	[illegible]	[illegible]
Intermediate PL	Burks	1950	?	F	A	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]sion
Klammer-ausdrücke	Rutishauser	1951	F	F	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible] code generation, loop expansion
Formules	Böhm	1951	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	D	Compiler in own language
AUTOCODE	Glennie	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	D	D	Useful compiler
A-2	Hopper	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	C	B	Macroexpander
Whirlwind translator	Laning [illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	C	A	B	Constants in formulas, manual for novices
AUTOCODE	[illegible]	[illegible]	[illegible]	[illegible]	B	D	C	A	C	Clean two-level storage
ПП-2	[illegible]	[illegible]	[illegible]	B	C	D	C	B	D	Code optimization
ПП	[illegible]	1955	F	B	B	C	C	B	C	Book about a compiler
BACAIC	[illegible]ter	1955	F	A	A	D	F	A	D	Use on two machines
Kompiler 2	[illegible]th & 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

Ignores the Nordic countries

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

Adding the Nordics and Algol, Simula

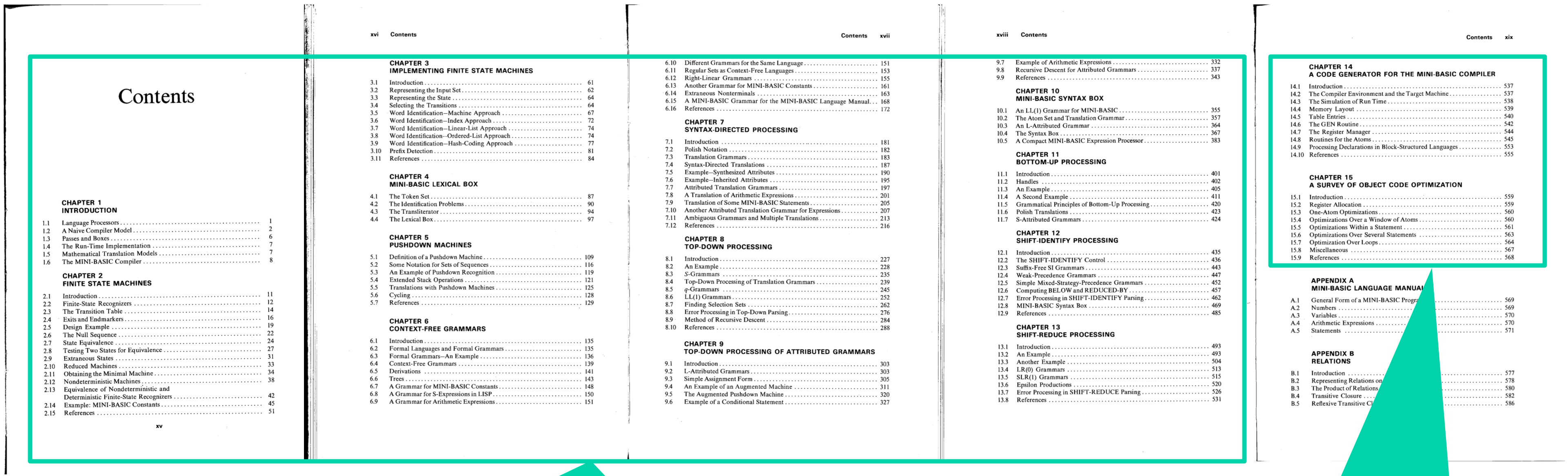
Language	Machine	Operatic	Developer	Comp. size	Comp spe	Citation 1
	EDSAC	Sep-1950	Wilkes, Wheeler, Gill			Wilkes:1951:ThePreparation
Speedcode	IBM 701	Sep-1953	Backus			Backus:1954:TheIbm
A-2	Univac I	Nov-1953	Hopper			Knuth:1977:TheEarly
Autocode	Mark I	Dec-1955	Brooker			Knuth:1977:TheEarly
IT	IBM 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

History: lexing and parsing

- Initially ad hoc
- Table-driven/automata methods
- Regular expressions, context-free grammars
- Finite state automata and pushdown automata
- Knuth LR parsing 1965
- Gries operator grammars 1968
- Lexer and parser generator tools
  - Lex (Lesk 1975) and Yacc (Johnson 1975)
  - LR dominated for a while
  - LL back in fashion today: Antlr, Coco/R, parser combinators, packrat parsers

Samelson:1960:SequentialFormula
Irons:1961:ASyntax
Naur:1963:TheDesign1
Knuth:1965:OnThe
Gries:1968:UseOf

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



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 ...

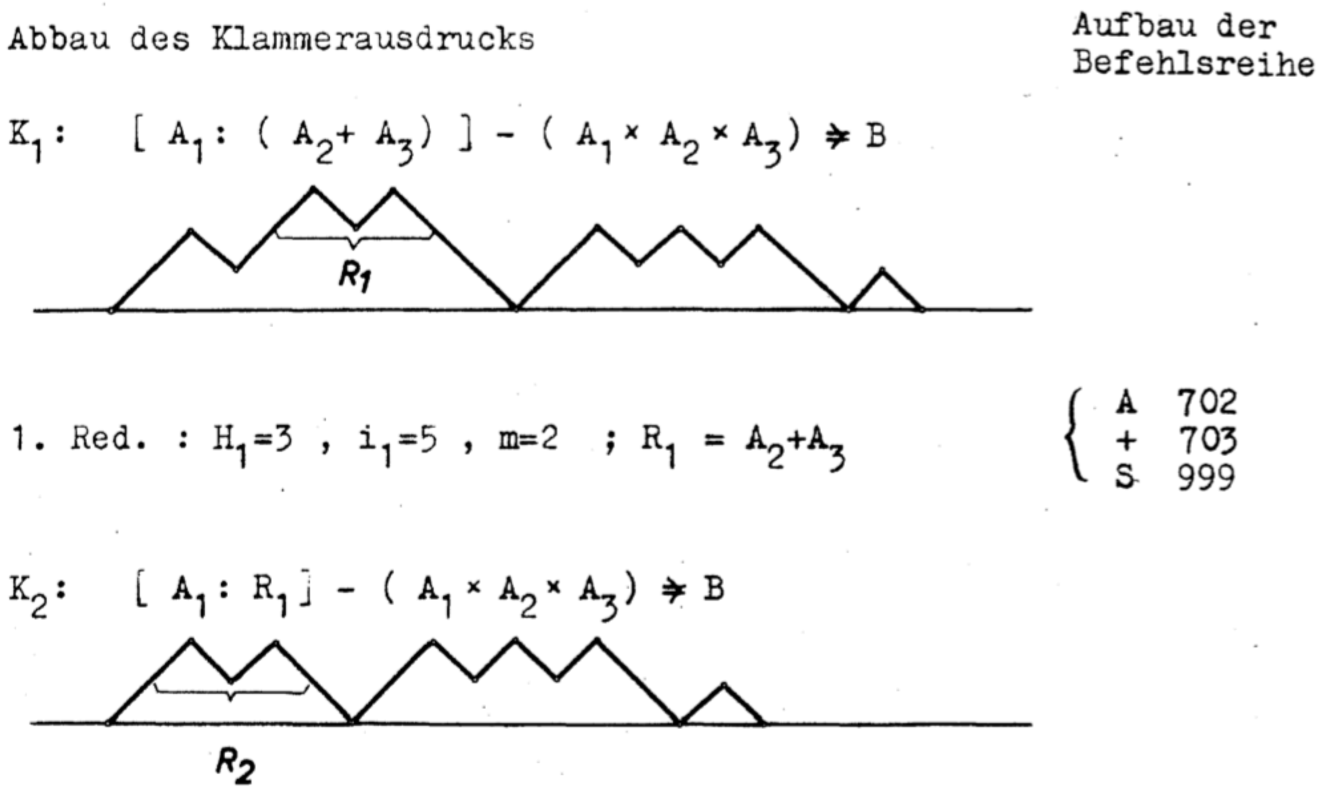
History: compilation of expressions

- Rutishauser 1952 (not impl.) Rutishauser:1952:AutomatischeRechenplanfertigung
  - Translating arithmetic expressions to 3-addr code
  - Infix operators, precedence, parentheses
  - Repeated scanning and simplification
- Böhm 1952 (not impl.) Boehm:1954:CalculatricesDigitales Knuth:1977:TheEarly
  - Single scan expression compilation – also at ETHZ
- Fortran I, 1957 Sheridan:1959:TheArithmetic
  - Baroque but simple treatment of precedence (Böhm &)
  - Complex, multiple scans
- Samelson and Bauer 1960 Samelson:1960:SequentialFormula
  - One scan, using a stack ("cellar") at compile-time
- Floyd 1961 Floyd:1961:AnAlgorithm
  - One left scan, one right scan, optimized code

Rutishauser, ETH Zürich 1952

Rutishauser:1952:AutomatischeRechenplanfertigung

- Multi-pass gradual compilation of expression



- Seems used also by
  - First BESM-I Programming Programme, Ershov 1958

Corrado Böhm, ETH Zürich 1951

Boehm:1954:CalculatricesDigitales

- An abstract machine, a language, a compiler
  - Three-address code with indirect addressing
  - Machine is realizable in hardware but not built
  - Only assignments $m + 1 \rightarrow m$; goto *C* is: $C \rightarrow \pi$
  - Compiler written in the compiled language
  - Single-pass compilation of fully paren. expressions

Nature r du dernier symbole

Nature s de l'avant-dernier symbole	)	(	→	var.	opér.
)	$G \rightarrow \pi$	$\Omega \rightarrow \pi$	$J \rightarrow \pi$	$\Omega \rightarrow \pi$	$T \rightarrow \pi$
(	$\Omega \rightarrow \pi$	$C \rightarrow \pi$	$\Omega \rightarrow \pi$	$P \rightarrow \pi$	$\Omega \rightarrow \pi$
→	$\Omega \rightarrow \pi$	$\Omega \rightarrow \pi$	$\Omega \rightarrow \pi$	$Q \rightarrow \pi$	$\Omega \rightarrow \pi$
var.	$H \rightarrow \pi$	$\Omega \rightarrow \pi$	$\Omega \rightarrow \pi$	$\Omega \rightarrow \pi$	$W \rightarrow \pi$
opér.	$\Omega \rightarrow \pi$	$I \rightarrow \pi$	$\Omega \rightarrow \pi$	$R \rightarrow \pi$	$\Omega \rightarrow \pi$

Expression compiler transition table

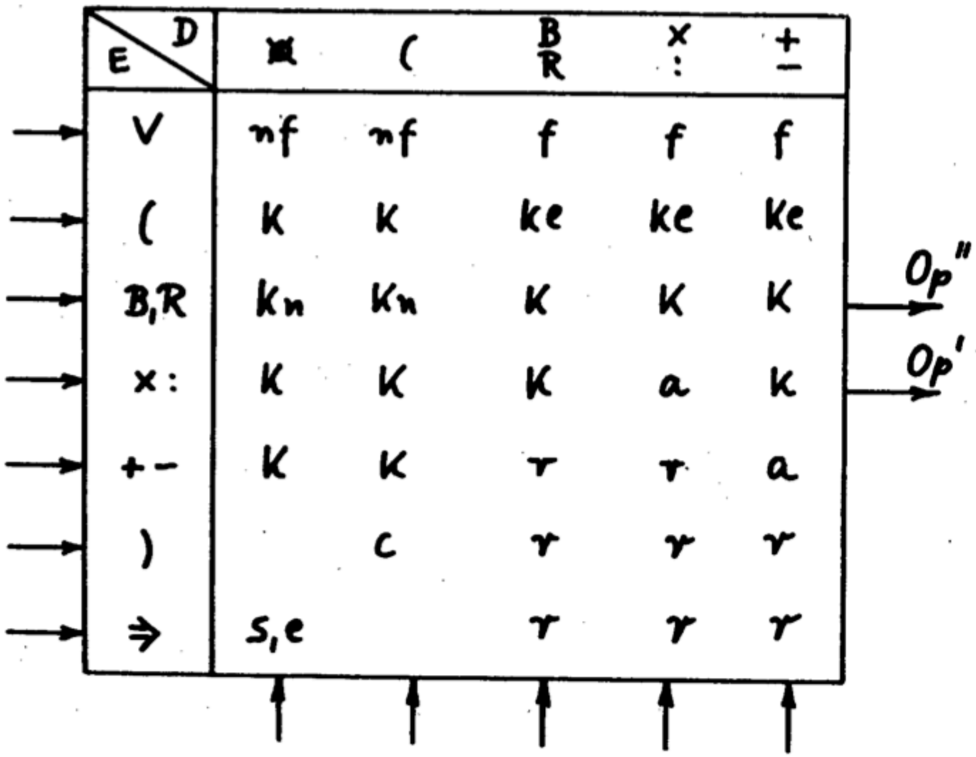
Implementation of transitions, goto

$$\pi' \rightarrow D$$
$$5 \cdot r + s + t \rightarrow \pi$$

Bauer and Samelson, Munich 1957: Sequential formula translation

- Using two stacks for single-pass translation
- Takes operator precedence into account
  - so unlike Böhm does not need full parenthetization

Bauer:1957:VerfahrenZur



E \ D	✖	(	B R	× :	+ −
V	nf	nf	f	f	f
(	K	K	ke	ke	Ke
B,R	Kn	Kn	K	K	K
× :	K	K	K	a	K
+ −	K	K	r	r	a
)		c	r	r	r
⇒	s,e		r	r	r

Bauer and Samelson's patent

Bauer:1957:VerfahrenZur

DEUTSCHES PATENTAMT

PATENTSCHRIFT 1 094 019

ANMELDETAG: 30. MÄRZ 1957

BEKANNTMACHUNG DER ANMELDUNG UND AUSGABE DER AUSLEGESCHRIFT: 1. DEZEMBER 1960

AUSGABE DER PATENTSCHRIFT: 12. AUGUST 1971

WEICHT AB VON AUSLEGESCHRIFT

1 094 019
P 10 94 019.9-53 (B 44122)

1

Die bekannten Rechenautomaten und Datenverarbeitungsanlagen erfordern im Einzelfall Anweisungen über die Art und den Ablauf der numerischen oder sonstigen informationsverarbeitenden Prozesse. Die Schreibweise, in der diese Anweisungen fixiert werden, wurde zu Beginn der Entwicklung so gewählt, daß sie gewisse als elementar erachtete technische Funktionen der Anlage beschrieb. Die so geschriebenen Anweisungen werden üblicherweise »Programm« genannt. Das Programm für einen Rechenprozeß etwa und die mathematische Formel, mit der der Mathematiker diesen Prozeß gewöhnlich be-

Rechenmaschine zur automatischen Verarbeitung von kodierten Daten

Patentiert für:

Siemens AG,
1000 Berlin und 8000 München

Dr. Friedrich Ludwig Bauer
und Dr. Klaus Samelson, 8000 München,
sind als Erfinder genannt worden

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
  - Lucas 1961, using explicit stack Lucas:1961:TheStructure
  - Hoare 1962, one procedure per language construct Hoare:1962:ReportOn
- 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

History: Run-time organization

- Early papers focus on *translation*
  - Runtime data management was trivial, eg. Fortran I
- Algol: *runtime storage allocation* is essential
- Dijkstra: Algol for X-1 (1960) Dijkstra:1960:RecursiveProgramming
  - Runtime *stack* of procedure activation records
  - *Display*, to access variables in enclosing scopes
- Also focus of Naur's Gier Algol papers, and Ekman's thesis on SMIL Algol Naur:1963:TheDesign1 Naur:1963:TheDesign2 Ekman:1962:KonstructionOch

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

Fortran I, 1957

- John Backus and others at IBM USA
- Infix arithmetics, mathematical formulas
- Structurally very primitive language
  - Simple function definitions, no recursion
  - No procedures
  - No scopes, no block structure
- Extremely ambitious compiler optimizations
  - common subexpression elimination
  - constant folding
  - fast index computations: reduction in strength
  - clever allocation of index registers
  - Monte Carlo simulation of execution frequencies (!)
- Large and slow compiler, 8 cards/minute

Algol 60, chiefly Europe

- Dijkstra NL, Bauer DE, Naur DK, Hoare UK, Randell UK, ... but also US, 1958-1962
- Beatiful "modern" programming language
  - Procedures, functions and recursion
  - Procedures as parameters to procedures
  - Block structure, nested scopes
- Compilers generated relatively slow code
  - Few optimizations

Early Nordic hardware and autocodes

- BESK, Sweden 1953, government research
  - By Stemme and others, based on IAS machine design
  - 4 bit binary-only code (Dahlquist, Dahlstrand)
  - FA-4 and FA-5 autocode, Hellström 1956, loader
  - Alfakod, symbolic no infixes, Riesel et al 1958
- Ferranti Mercury, Norway 1957, defense research
  - Commercial, first machine delivered, 1m NOK
  - MAC, Mercury Autocode by O-J Dahl, arrays, indexing, infix
    - Not used elsewhere
    - Independent of Brooker, Manchester Autocode, 1956-1958
- DASK, Denmark 1958, government research
  - By Scharøe and others, based on BESK + index registers
  - Naur EDSAC-inspired symbolic loader, 5 bit, 1957
  - No need for a more complex autocode
  - Instead an Algol 60 compiler (though without recursion)

http://www.itu.dk/people/sestoft/papers/sestoft-hinc-2014.pdf

Example problem

- Compute the polynomial
  $$f(x) = a_0x^8 + a_1x^7 + \ldots + a_7x + a_8$$
  using Horner's rule

- In Java or C or C++ or C# anno 2014:

```
res := 0.0;
for (i = 0; i <= 8; i++)
  res = a[i] + x * res;
```

BESK FA-5 and Alfakod, Stockholm

- Input on 4-bit paper tape (hexadecimal) only
- Hellström & Dahlquist, FA-5 1956 Dahlstrand:2009:Minnen
- Riesel et al, Alfakod 1957

BESK hex. code 1953
Dahlquist:1956:KodningFor

Self-modifying

200	18050	10000 till AR
201	20407	100 till hac204:s adresspos.
202	00648	0 till MR, d.v.s. $y_{-1} = 0$
203	00863	$y_{n-1} \cdot x$ till AR
204	FFF28	$y_{n-1} \cdot x + a_n$ till MR ($y_n \rightarrow y_{n-1}$)
205	20446	$W(a_{n+1}) \rightarrow W(a_n)$
206	1820B	$W(a_n)$ - 113 till AR
207	2034E	Hoppa om $W(a_n) \leq 112$
208	00001	mr till AR
209	18431	$y_9 = f(x)$ till 184
20A	3000C	Uthopp

Self-modifying

AAAOO

FA-5 1956
Hellstroem:1958:KodningMed

```
      18050
      20407
      00648
A203  00863
A204  FFF28
      20446
      1820B
      2034E
      00001
      18431
      3000C
```

Alfakod 1958
Riesel:1958:AlfakodningFor

```
    NOL  AR
    FIX  0,I
1   MUL  X
    ADD  Y/I
    ADX  1,I
    VMX  1,I,8
```

Ferranti Mercury, NDRE Oslo

- Dahl, MAC=Mercury Autocode 1957 Dahl:1957:AutocodingFor Dahl:1957:MultipleIndex
- Note
  - Infix arithmetic, logical expressions
  - Symbolic labels such as `(1`
  - Real and complex numbers, arrays (1D, 2D, 3D)
  - Array index expressions `Un1` with optimization

```
0 -> A
0 -> n1
Un1 + X A -> A      (1
n1 + 1 -> n1
n1 < 9 ? JUMP1
```

DASK loader, Copenhagen

- Naur, EDSAC-inspired external code, 1957
- Naur was unimpressed with the BESK code:

har aldrig forstået fordelen ved dette system. På Besk er det en pinlig nødvendighed, på grund af det rudimentære indlæsesystem

Naur:1957:DaskOrdrekode

DASK code 1958

Andersen:1958:LaereboqI

```
200 2030 A 35 ; IRB := -18
201 2042 A 44 ; MR := 0
202    2 B 35 ; IRB := IRB + 2
203    8 A 0A ; AR := x * MR
204  118 B 00 ; AR:=AR+[118+IRB]
205  202 A 33 ; if IRB<>0 goto 202
```

Index register,
not self-modifying

DASK Algol 1961

Naur:1964:RevisedReport

```
res := 0;
for i := 0 until 8 do
  res := a[i] + x * res;
```

Early Nordic (Algol) compilers

- Naur, Jensen, Mondrup, in Copenhagen
  - DASK Algol 1961, no recursion
  - GIER Algol 1962
- Dahlstrand and Laryd, in Gothenburg (Facit)
  - FACIT Algol 1961, no recursion, based on Naur ...
  - SAAB Algol 1963
- Ekman, in Lund
  - SMIL Algol 1962, no recursion
- Dahl and Nygaard, in Oslo
  - Simula 1965, based on Univac 1107 Algol from US
  - First object-oriented language
  - Extremely influential: Smalltalk, C++, Java, C#...

The nuclear origins of OO

- Garwick, Nygaard, Dahl at NDRE, the Norwegian Deference Research Establishment
- Norway 6th country to have a nuclear reactor
  - in November 1951
  - six years before Risø in Denmark

Randers:1946:RapportTil
Forlan:1987:PaaLeiting
Forlan:1997:NorwaysNuclear

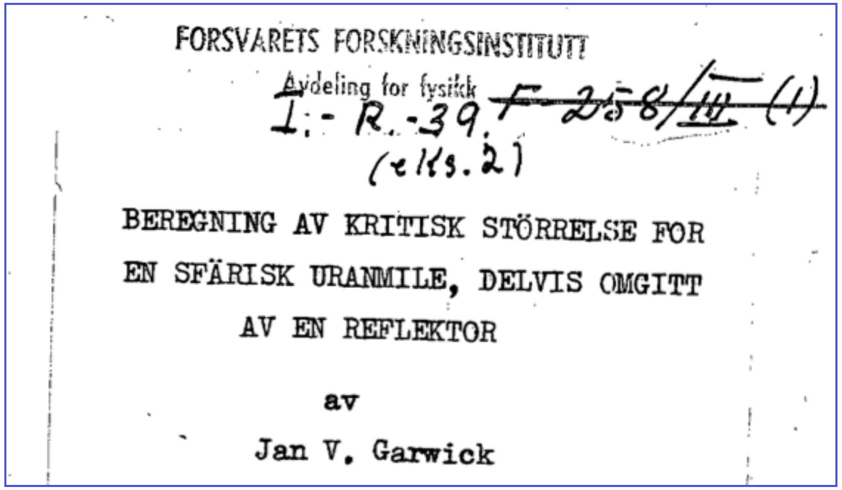
- Garwick and Nygaard computed parts of the reactor design 1947-1951
  - w Monte Carlo methods to simulate neutron flow
  - chiefly hand calculators

Holmevik:1994:CompilingSimula
Holmevik:2005:InsideInnovation

Garwick:1947:Kritisk
Garwick:1951:BeregningAv
Nygaard:1952:OnThe

- Ole-Johan Dahl hired 1952
  - developed "programs" for modified Bull mech. calc.
  - from 1957 developed MAC autocode for Ferranti

Norway: Nuclear and computing 1946-1962



FORSVARETS FORSKNINGSINSTITUTT
Avdeling for fysikk
I.- R.-39. ~~F-258/III (1)~~
(eks. 2)

BEREGNING AV KRITISK STÖRRELSE FOR
EN SFÄRISK URANMILE, DELVIS OMGITT
AV EN REFLEKTOR

av

Jan V. Garwick

Garwick:1947:Kritisk

Holmevik:2005:InsideInnovation

Forlan:1997:NorwaysNuclear

Forlan:1987:PaaLeiting

Randers:1946:RapportTil

Norway's nuclear and computing pioneers						2014-02-06																				
	1939	**1940**	**1941**	**1942**	**1943**	**1944**	**1945**	**1946**	**1947**	**1948**	**1949**	**1950**	**1951**	**1952**	**1953**	**1954**	**1955**	**1956**	**1957**	**1958**	**1959**	**1960**	**1961**	**1962**	**1963**	**...**
Inst. Astrophysic:	GRan																									
	JGar	JGar	JGar	JGar			JGar	JGar																		
		KNyg																								
NDRE/IFA								GRan	GRan	GRan	GRan	GRan	GRan	GRan	GRan	GRan	GRan	GRan	GRan	GRan	GRan	GRan	GRan	GRan	GRan	
								ODah	ODah	ODah	ODah	ODah	ODah	ODah	ODah											
									JGar	JGar	JGar	JGar	JGar	JGar	JGar	JGar	JGar	JGar					JGar	JGar	JGar	
										KNyg	KNyg	KNyg	KNyg	KNyg	KNyg	KNyg	KNyg	KNyg	KNyg	KNyg	KNyg					
														OJDa	OJDa	OJDa	OJDa	OJDa	OJDa	OJDa	OJDa	OJDa	OJDa			
NCC																						KNyg	KNyg	KNyg	KNyg	
																								OJDa	OJDa	
USA		GRan	GRan	GRan				GRan																		
								ODah																		
UK					GRan	GRan	GRan																			
France									JGar																	
							rep		nuc	nuc	nuc	nuc	crit						mac	mac	mac			sim 1	sim 1	
Gunnar Randers	GRan																									
Odd Dahl	ODah																									
Jan Garwick	JGar																									
Kristen Nygaard	KNyg	b																								
Ole-Johan Dahl	OJDa	g																								

Recommended reading

- Secondary sources
  - Knuth:1977:TheEarly
  - Bauer:1974:HistoricalRemarks
  - Ershov:1976:Addendum
  - Randell:1964:Algol60Implementation sec 1.2, 1.3
- Primary sources
  - Backus:1957:TheFortran
  - Samelson:1960:SequentialFormula
  - Dijkstra:1960:RecursiveProgramming
  - Hoare:1962:ReportOn
  - Naur:1963:TheDesign1
  - Naur:1965:CheckingOf
  - Randell:1964:Algol60Implementation

Thanks to

- Christian Gram, Dansk Datahistorisk Forening
- Robert Glück, DIKU, Copenhagen University
- Birger Møller-Petersen, Oslo University
- Knut Hegna, Oslo University Library
- Bjørg Asphaug, NDRE Library, Oslo
- Ingemar Dahlstrand, Lund University
- Torgil Ekman, Lund University
- Mikhail Bulyonkov, Russian Ac. Sci. Novosibirsk
- Christine di Bella, IAS Archives, Princeton
- George Dyson, Bellingham WA, USA
- Peter du Rietz, Tekniska Museet, Stockholm
- Hans Riesel, Uppsala University
- Dag Belsnes, Oslo
- Peter Naur, Copenhagen University
- Norman Sanders, UK