

July 7, 1958

Bill Heising "Inside Fortran"

Prog. Research Dept - Bureau - Fortran
Applied Prog. - 709 versions

- Primarily Numerical Purposes

- (a) - original source language
- (b) - Translating program in 704, — an extension of assembly

1. Translation of formulas + ~~the~~ with + I/O ~ 4000 instructions
2. Index reg. of 704 (Index assembly) — the most difficult 12-15000 instructions
3. Address assembly 2000 instructions

tried to get high efficiency near that of ordinary code,

Types of statements:

I. arith statements $A = B + C * (D + E)$
 $I = I + 1$

only single valued
ops. are handled,

Floating pt : all other letters.
 fixed point I J K L M at start.

+ add
 - sub
 * multiply
 / div

Dimension statements: size of array

DIMENSION A(10, 5, 3)

means 1st element A(1,1,1)
 2nd A(2,1,1)

Most general form allowed:

$A(5*I+3, 4J+2, 6K+7)$

where I, J, K are defined earlier

in column order,
backwards in memory

A(1,1,1) A
 A(2,1,1) A-1
 A(10,1,1) A-9

13 A(I) := 3 * Q / (4.3 * B)

branches

GO TO 13

IF (2 * I - 5) 4, 5, 6
 ↑ ↑ ↑
 4- 40 4+

GO TO (13, 14, 17, 31, 6), I

 ↑ ↑ ↑ ↑
 4 I=1 2 3 + 5

variable bit

STOP

"Do Statements"

DO 6 J=1, K, 2

J from 1 to K
in steps of 2

= A(J) ...

6

equivalent to:

J=1

{

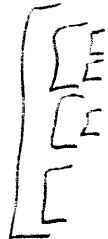
 J=J+2

 IF (J-K) 3, 4.

}

but program is not as efficient as DO.

Do's may be overlapped
nested but not interlocked,



one can jump out of a DO
but not into a DO.

So DO has only one entry

cannot write A(I * J)

must write | K = I * J
 | A(K)

Sections of FORTRAN

(1) Classifies statements by type

(a) compiles arithmetic statements

(b) " I/O "

(c) " IF "

{ TP+
TRO
TR

(d) builds up DO tables
& Subscript tables.

{ FORVAL
(READ defined)

note { Index quantities are not as complicated

So tables are backwards in mem.

SXD active vars → I

TXI , 4, 1

SXD
TXL , 7, (I)

So fixed vars are Dec. field only.

2¹²...2¹⁵

(2) Handle DO's, loading & storing index values etc. compiled in assuming any no. of index registers.

$A(5 \times I, J+3)$ means $S(I-1) + (J+3-1) \times 5 + 1 = 5I + J - 2$

compiles: $114 - 2$ ~~variable part~~ variable & fixed part split.
address in index reg. in table

DO I 3, 7, 1 { $S+J \rightarrow 114-2$
LXD e, $114-2$

carried symbolically.

TXI 1, $114-2, 5$

other ones

I = $\left[\begin{array}{l} J = \bigcirc \\ I = \bigcirc \end{array} \right.$

computed at point of definition:

done: pure subscript defined by DO
mixed subscript (partly by FORVAL
all by FORVAL

3. Odd jobs,

(4) } reduce no of index regs to 3

(5) }

(automatic storage allocation needed)

(6). Conventional assembly program.

4 & 5

Break into Basic Blocks of coding i.e. draw flow charts



Monte Carlo test on frequency - take one or another branch (or using freq. statements)

- get ~~out~~ tables: (a) block nos. + list of successors.
- (b) " " " " predecessors.

- look at most freq parts of program

then let it grow block by block - inserting LX & SX orders,

Take block & trace forward & backward until

- (1) get a loop
- (2) "opaque" all 3 are being used
- (3) "transparent" not all 3 are being used - trace further. may have to permute 2 registers

- as regions grow they become "opaque"

(4) or string is too long - terminate arbitrarily.

take index to be displaced - trace ahead until used again. Insert SXD, LXD.

"Activity" - remembers that index is used & SXD before displacing,

