Algorithm for the Optimization of Arithmetic Expressions in 1401 FORTRAN

Two inefficiencies which exist currently in the strings generated for Surety-
-•^
metric expressions by the 1401 FORTRAN compiler are discussed.
c-^
1. Redundant parenthesis generate redundant object time processing.
2. Generalization of the treatment of functions has led to inefficient output strings in specific cases.

The expression
A - <B) + (C)
generates
GT1 - B
GT2 - C
- A - GT1 + GT2
46
^BF The expression
A = B - C * D
generates
A - C * D + B
which is correct, but because
A - SINE (B) + C * D
would generate
A - SINE (C ^ D ^ B)
a rule was established which states that all functions force a generated temporary. Consequently, the above expression is generated as
GT1 = SINE (B)
I® A - C * D 4- GT1

-2-
which is correct.
^
^^ But this rule leads to an inefficiency in the case of
A - SINE (B)
which produces
GT1 "SINE (B)
A » GT1
To correct this situation, the following algorithm is recommended.
Algorithm
R"^ 1' K the operand immediately following the equal sign of a string is a generated temporary, the computation erf the GT can be substituted for the operand and the GT string can be deleted.
J& Rule 2. When a GT occurs to the right of the equal sign, but is not the first operand, and all preceding operators have the same hierarchy, then by the rule of commutivity, the GT can become the first operand and procedure 1 will apply.
Exception 1. K the operator preceding the operand is "-", the negate function must be used*
A - B - GT1
is equivalent to
A --= NEGATE (GT1) + B
Exception 2. If the operator preceding the operand is "/", optimization should not take place. Tlie use of the invert function is unacceptable

^.   in fixed point computation and processing time might be increased In the case of floating point.

\ - ^ ^ -
n
SNAPSHOT PHASE   ^g
Through.
ARITH PACKAGE
Snapshot Phase
Essentially a copy of the debugging aid. this ^hase prints out storage starting at the beginning of variable storage when requested and if there are no input errors.
Condensed Deck Phase One
Punches the clear storage and bootstrap cards.
Condensed Deck Phase Two
Punches the cards which
1. Initialize sense lights and index registers   ^
2. The parameter card constants required at object time.
\0-^/
3. The debugging hoiil and arith package.
4. Four cards which initialize the arith package.
Condensed Deck Phase Three
Punches out storage from the first executable statement to the end of storage, .bypassing unused storage.
Geanx Phase One
Prints end of job messages.
Geaux Phase Two and Arith Package
Reads in the Arith Package and initializes four operands of this routine. The printer carriage is then restored displaying the end of job messages. ||

" >£> 'J -
REPLACE PHASE TWO
^W   This phase sqffa^ the procedure and format sections for " T" operation codes and for operands with 11-5-8 or 11-6-8 characters in their hundredths positions.
T-op codes
These instructions are those which branch to the relocatable routines. Their A/I operands reference the table generated by the Function/Sub-routine Loader Phase. When encountered, the T op code is changed to B and the address of the relocated routine is substituted in the operand.
Special operands
The operands with 11-5-8 and 11-6-8 combinations represent, those which
^^   reference the two work areas generated in Constants Phase Two equal to
^| the size of fixed word (k) and float word (^+ Z\. The units position of the operand represents character adjustment of these work areas relative to their units position. This phase substitutes the appropriate address in the operands which reference them.
•)

-JT7-
OD   DO PHASE
Each Do statement generates two output statements; the Do statement itself and an unconditional branch which follows the statement that terminates the Do. The Do string generated is
BXXXBYYYAAABBBCCCDDDEEE

where XXX is the address of the relocatable routine which initializes "I^A
index; YYY is the address of the relocatable routine which initializes
the test for satisfaction of the Do; AAA is the address of Mj; BBB is M-?'
CCC is M3; DDD is the index variable; and EEE is the exit address when
the Do is satisfied.
The Do Phase processes these statements backwards, i. e. the last Do
^Q first, and analyses the relationship of the Do being processed to the other
•^J^/ '^3' ^^s^9_)
Do statements. If the Do being processed isa«innermosi\D<<^tneun-
conditional branch generated after the last statement of the Do is a branch
i ^ /
to the relocatable routine which tests whether ihesDrfis satisfied. Other-
wise, the branch generated is to the second unconditional branch of the *t»^
Do string (indicated above).
If more than one Do terminates at the same statement, the exit address
EEE for the inner onel reference^the second unconditional branch instruc-
tion in the string of the next outermost Do. Otherwise the exit address is
to the next executable statement following the range of the Do.
«

- G. ^ -
REPLACE PHASE ONE ^
This phase scans the generated source program twice. The first time
it scans for subscript strings. WhenVncountered, it seta word marka
in the hundredth position of each parameter and unzones the tens posi-
tion of all parameters except the first.
The second scan looks at the procedure section for the following chara-
cters which have word marks: X. T, ^ and for operands of instructions
which have AB zoning in the tens position.
^T word mark
If followed by a character with a word mark (PSKIP) the statement is arith
and is bypassed (SKIP)^
T word mark ' ^
•"ki,
If the statement is a Do, the tens position of all parameters is unzoned
and the exit address is generated.
X word mark
These are the instructions which terminate the range of Do statements
which are not the innermost Do. The proper branch address is generated.
Operands with AB zoning
^^
These are the "y" nnn" type operands whose correct address is relative
pro^r" ^
to itself. The pa?eg<A&6 address is substituted.
•
'-.i-'
te

-^y-
^
I? INPUT/OUTPUT PHASE TWO
Through
CONTINUE PHASE
These phases generate the required in-line instructioz^or proper exe-

cution of the various statement types involved^
^"-^- Input/Output Phase two
Rewind:
U UnR
<.
Backspace:
U^UnB
«c.
End File: »
-^-\
U7o UnM
"' ^
where n is the tape unit number specified. If the tape designated is symbolic, the above instructions are preceded by an instxuction which is symbolically represented as
^ t/^r^ ^^^
MN/IH, X + 4
Computed Go to Phase
^^yi&'roj-^^^
This phase pperatej^e instructions
BCE XXX. III. A
BCE XXX, III, B
•

-J>y-
•
H
^ »
B y- 8 rf
where XXX and YYY are the exits and III is the fixed point non-subscripted variable. If the value of the variable exceeds the number of exits, the machine enters the halt loop at object time.
Go To Phase
This phase generates
B XXX
for Go To statements
Stop/Pause Phase ^ji
this phase generates
1. NOP nnn
2. H
\^V
3. B V- 8 ^
\7
for stop statements, where nnn is the halt number specified or 000 i^B no number is specified. Pause statements are identical except that instruction 3 is not generated.
Sense Light Phase
a&e^se lights are represented by work marks in locations 081-084. The presence of a word mark indicates the light is off.
Sense Light 0
. 082084_

ARITH PHASE SIX ^1
\^r../
This phase plan^arith statements for function codes and sets the switches

^y
to call Wthe required function in the Function/Subroutine Loader Phase.
«
'^
•*

<»
STATEMENT NUMBER PHASE TWO
Same as Variables Phase Two.
•
•

— -y.0-
ARITH PHASE THREE ^|
Key work areas and subroutines
TRAP - value of the temporary store substring
STOR - init,a.Llly TRAP; bumped by one to create additional
temporary store strings.
NORTH - If this location contains a word mark. there are no
arith statements.
TP1SW- If no wordmark then exponentiation encountered.
TP2SW -If no wordmark then multiplication/division encountered.
TP3SW -If no wordmark then addition/subtraction encountered.
STAR 2 -Starting address of mult/div string ^g
STAly 3 -Starting address of add/sub string
fc
PREV-- Previous operator to mult/div string (used to force negate
or invert function)
Subroutines
—— ^'y
INRMO -^C-wt subroutine constitutes the main portion of this phase.
It breaks down the hierarchy of execution.
GETNX -Gets next operand and operator. This is the most used .
subroutine within INRMO. It places the operator preceding
N?/
the operand being analyzed in OP and the succeeding operator
in OP2.
GTEMP -Where additional substrings must be created due to hierarchy. l(|
^
this subroutine generates another temporary work area and

- y^ -
^P ARITH PHASE TWO
This phase scans arithmetic and ^statements for function names.
Where they exist, the name is deleted and a one character code is
substituted.
«
^
•

~^J-
INPUT/OUTPUT PHASE ONE 1B
All I/O statements except Rewind, Backspace and End File are reduced
•^r/
to thxWobject time procedure string.

BW72XAAAB_BB
Where "BW72" is executed by the machine as branch to 1672, the location of the format package; X appears as the I/O type
^ ^ - Print
+ - Read
- - Punch
or the numeric portion represents the tape unit number and the zone indicates the I/O type.
Q
NZ - Read tape 'y
A -Write tape
B - Read Input tape
AB - Write Output tape
If the tape unit numbers is symbolic, the above string is preceded by an instruction which is symbolically represented as
^r^< ^c/. ^
MN III, X + 5
^
where III is the address of the fixed point non-subscripted variables.
^
• -^

Aft STATEMENT NUMBERS PHASE FIVE
^^ ' k
This phase checks for Undefined Statement Number.. Thi. occur. when an entry in the Statement Number Tabl. was unreferenced by the previous phase.
Note that Dimension. Equivalence and Format .tatement. have been elimin-ated prior to the statement number phase,. A. , con.eou.nc., all reference. to these statements will produce an error message.
•
•

-^0-
STATEMENT NUMBERS PHASE FOUR
The external statement numbers are matched against the Table of Statement Numbers that were present in the body of the statements. If there is an entry in the ta.le, this entry is replaced by the Internal * sequence number of the statement which it references.
To illustrate the progress of statement numbers, consider
»
these two statements:
1. GO TO 20
2. 20 STOP 123
Prior to Statement Numbers Phase One, these two statements have been reduced to:
1. T 02 £ G 012 ^
^^ 2. i 32l£ 025016 I:
In Statement Numbers Phase One, the number "20" (appears as "02" above) is converted to a three-character unique representation:
1. ^ XYZ f G012 ^
2. .£ 321 'i XYZS0165
In Statement Numbers Phase Three, "XYZ" is placed in a
I^
statement numbers table by virtue of the GO TO expression. The table

location is substituted for XYZ in the GO TO statement
•

' 71 "
1. ^ G4Z S G012 I
2. £ 321 I XY2S016 5 6
3. XYZ
'—— location G42
In Statement Numbers Phase Four, the table entry XY2 Is
located when processing the STOP statement. The Internal sequence
number of the STOP is substituted in the table
1. I G42 I G012 |
2. ^ 321 I SG4Z X
3. 016
The compiler has now established for future phases that the
GO TO statement will transfer control to internal sequence number 016.
Do statements receive special treatment in the phase. The fl
compiler requires a.at each Do statement have an entry in the statement '"
number table.
If a Do statement has no external statement number, or if this
number is unreferenced, an entry is placed in the table (PSUDO).
This phase also detects unreferenced and multiply defined
statement numbers. Unreferenced statement numbers are those which
have no table entry. Upon referencing a table entry in Table n
the three character representation Is placed in Table 1. if another
statement occurs with the same representation, it is detected as
multiply defined.
••*

- „•? /-
€9
LISTS PHASE THREE
,T: ""• •"•"" - •""— —"•"»./o .«.„....,
°°""h•°tt••"'"•o"l•'"-«-.<.^.^,..,
""(r""' •"""" - ""> - .H. .... „,. _, .^..
applicable). { ere
•
•
t

STATEMENT NUMBERS PHASE ONE
fR
Processing in this phase is straightfor.ard. All statement numbers are
convened to unique three character representations. A table of 50
charters (TABLE) is used. The literal 50 is subtracted from the
-cond and third, and fourth and fifth positions of each statement
n-ber. If the result is positive for the latter, one (1) is added to
the first character. If the former is positive, two (2) is added to the
Hrst character. The characters .n the table replace the three sections
of the original statement number.
Each statement type is processed separately due to the different locations
ror statement numbers. The unique r^,^^^,^ ^ ^
• banning (r^htmost) part of the statement and terminated ^ a comma.
Adequate error chec.ing is provided to check syntax and to insure that
all required statement numbers are present.

1

SUBSCRIPTS PHASE ^j
n
This phase cleans up subscripts, eliminating the commas and asterisks ncccsaary for proper processing of subscript constants.
The end result is the subscript parameters as they will exiat at object {.xn-.e.
«>
\.

--^y-
VARIABLES PHASE FIVE 'll
This phase scans for unreferenced variables. Each element in Variables Table II has the following format:
[x x^x f iY Y Y Y Y!

Address of Variable	Variable Name

The word mark is cleared fron the group mark in Variables Phase Four (SWCHC) when a variable is referenced.
This phase scans Table II for entries where the word mark still exists.
-,-g

•VARIABLES PHASE THREE
A housekeeping phase. The heading line "Storage Assignment - Simple Variables" is printed. NXTOP is converted to five characters and is
^
^^- • stored in VfORKf,
•
I
•

STATEMENT NUMBERS PHASE ONE
«
Processing in this phase is straightforward. All statement numbers are converted to unique three character representations. A table of 50 characters (TABLE) is used. The literal 50 is subtracted from the second and third, and fourth and fifth positions of each statement number. If the result is positive for the latter, one (1) is added to the first character. If the former is positive, two (2) is added to the first character. The characters in the table replace the three sections of the original statement number.
Each statement type is processed separately due to the different locations
<• , , \^-^^ ---ki-^vc^ ^;^l^', ^.y.-../
lor statement numbers. The unique re^Te-se n tat FSH^p laced at the
99 beginning (rightmost) part of the statement and terminated with a comma.
Adequate error checking is provided to check syntax and to insure that all required statement numbers are present.
t
^D

SUBSCRIPTS PHASE A
^1
Th.s phase c.ear.o up subscripts, eliminating the commas and asterisks

nccc^bary for proper processing of subscript constants.
The end rc&ult is the subscript parameters as they will exist at object
Lime.
<
\
•*

/ /
•INS2RT GRO'UP MARK PHASE
All c-5 characters within the range of the source program are
converted LO group mark/word mark. Tnis character appears between
me body of the statement and its appendage.
This. phase is terminated by reaching the character blank (BTEST).
This test Is NOPed when the statement is FORMAT (ISFMT).
•
\.
«

~ ^ —
^P Lipon comi-letlon of cacn ;-hase of the compiler, control Is
-——:.::rrcci to ^ ^L.:r. monitor (^ONTER). The monitor clears
- - prw.o^ p^je (ACLSAR) and L.en reads the next phase from
^-rds or ^pe (:<IONTOR). Tne location MONTOR Is NOPed by the
...^c^ ^r.asc, If system input Is from tape.
^rior to transferring control to the monitor, each phase
-t-^lzes certain operands of ±e monitor. This Initialization Is
-ceomphjned by the FENDX macro.
'.. CL^R - The highest address to be cleared by tne monitor.
PCLZ^R - The lowest address 10 be cleared by the monitor.
^J^ HTT^/d - The address Into which the next phase Is to be
:^-d w:ie^ opc rating as a tape system.
——XT/3 - The address to which the monitor branches after
rc.-.Jl::g ^ next phase. This Is handled by the XFR card In the card
sy^trim.
I; a.'.y o: Aese operands are tne same as the previous phase,
they are ".ot rc-.--.iualiz;ed.
«)

•< l-.C^t-ortr.ir. ''-/' "' • '• .
. /
Deta;i<ja'Explanation ol' Processor
——s do^.;;,.-:^ ,:. dcs^n.-a 10:- persons requiring a thorough knowledge
<- 1;..- ..Ci r-^-ir.,. processor. E^h phase ol the processor is described
- - ^A.-:-.:U .eclion. Infonn.a.on enclosed in parentheses and capitalized
r''^-;- L<. syn-ibolic labels present in the Wsting.
Ge;.••.-.. I
The re..der should be acquainted wKh certain key work areas addressed
tn.-ou^ho.iL ii^- processor.
^^ I . P-K..XL.V - The hundreds position of the machine size specified
or. ti.<- o,;ii'-Qi c.ird when located in storage.
i'.'-.KAMA^Z - The sj zc of the machine
^. P/.^AMA-l - The niodul'^s
c. PAKAMArO - The mantissa and later in the processor,
Ll-ie n-;;tntissa plus two.
^. I-\-JL.S,V - A ^-ord ;nark is set .it Lhis location when the processor

d^tect^ ..n erro" wnich would rnake obj.'cfc tin-;e unrew a rdu^.
NXTOP - Tne next available l^^on ..r ^njoct tirm. fron. the top
^e:Ln.u^L) ^arL o^- storage It is located at Oa6 .
XX^TM - Tne next available location at ..bject time ircm the
^^ ••••-•- • .-^.•;,r":-T-.Obt) pa;-t 01 storage. It ib lo.. .led at 083.

n
Snapshot Routine (00) ^i
This routine exists in storage through the entire compilation and
object time. It produces snapshots of storage whenever control is
transferred to it. The snapshot displays each century of storage on a
separate line with appropriate trimmings to assist in checking programs.
If sense switch F is on, the dump is not performed and only the-message "Phase
name" EXECUTED is printed. If sense switch G is up the routine halts after
execution. The print area (locations 200-332) is neither printed nor
saved.
This is a closed subroutine and control is transferred after execution, to
—— ﬂ
the next sequential instruction after the branch to the as&ssfrot routines ^
^sa- j^.
Snapshot uses index registers 1 and 3 for execution. These registers are
saved at the beginning of execution and restored at the end.
If the index registers have zoning over the tens position, the display of
the registers will be incorrect due to the logic which moved the values to
the print area (HLDXT,HLD32, HLD3 1).
System Monitor and Parameter Card (01)
Up to 19 positions have been allowed for the control card (PRMCD).
\y
Currently, only \\f positions are used.

Exception 3. ff any function computation precedes the GT, optimization
cannot occur.
Example of Rule 1
A -= (B » C) •* D
currently produces
GT1 " B i» C
A » EXPF (LOGF (GT1) * D)
or in string notation
GT1 » BBB * CCC + AAA - GTl^ •r Dg f
By applying rule 1 we get
A = EXPF (LOGF (B * C) * D)
or in string notation
AAA « BBB • CCC ^ * DE t
Example of Rule 2
A - B » (C ^ D)
currently produces
GT1 » C + D
A - B * GT1
but by rule 2 It can be written as
GT1 - C ^ D
A - GT1 • B
and by rule 1 can be reduced to
A - C 4 D * B

where each operation is done serially (hierarchy does not apply).

The algorithm can be refined if it la conveniently programmable when more
flB
^W than one GT occurs in the string. Consider the following expression
A ° (B (- C) * (D * E) * (F * G)
This currently produces
GT1 - B + C
GT2 » D * E
GTS » F * G
A » GT1 « GT8 t GT3
by applying rule 1, it can be reduced to
GT2 » D * E
GT8 - F * G
A - B + C * GT2 * GT3
No further optimization can occur. However, if the original expression had
been written as
A - (D * E) « (F * G) • (B . C)
the string could have been reduced to
GT3 ° B + C
A=*D*Ei»F«G* GT3
It appears, therefore, that a third rule should be established which states
that when an expression contains more than one GT, analysis of the expressions
represented by the GT's should occur before optimization takes place. This
rule wUl be harder to implement than the first two rules.
HJff Gary Mokotoff
GM:meb Gp Advanced Programming Envelopment