The research reported in this paper was sponsored by the Advanced Research Projects Agency Information Processing Techniques Office and was monitored by the Electronic Systems Division, Air Force Systems Command under contract F1962867C0004, Information Processing Techniques, with the System Development Corporation.

TECH MEMO *a working paper*

System Development Corporation / 2500 Colorado Avenue / Santa Monica, California 90406
Information International Inc. / 11161 Pico Boulevard / Los Angeles, California 90064

TM-3417/380/00

AUTHOR [illegible]
D. Anschultz

TECHNICAL Jeff Barnett
J. Barnett

RELEASE [illegible]
C. Weissman, S.D.C.
[illegible]
D. Anschultz, I.I.I.
for J. I. Schwartz

DATE 4/26/67 PAGE 1 OF 4 PAGES

(Page 2 is blank)

LISP 2 Compiler Machine Link Specifications

ABSTRACT

This document describes the functions performed during the Machine Link pass of the LISP 2 compiler proposed for the IBM S/360 computer. The Machine Link (pass IV of the LISP 2 compiler) determines what will be accomplished by in-line code generation as opposed to function calls, and solves machine-related precision problems.

1. INTRODUCTION

The Machine Link is the first pass of the LISP 2 compiler that considers the machine for which code is to be generated. The following tasks are accomplished during this pass:

1. Basic information required for optimal register allocation is obtained.
2. Certain forms previously treated as function calls are detected and changed to the appropriate in-line form.
3. CONVERT's are changed to one of the following:
   (a) function-call on convert routine
   (b) in-line form
   (c) no action at all
4. Optimization to account for the distributive property of MINUS.

2. LEXREG PROCESSING

As bindings for lexical variables are encountered, their location in the expression is added to a list kept in a fluid variable location. Whenever a reference to a variable is encountered and the variable is on this list, one of two things is done: if this reference appears as the argument of an AFIELD or LFIELD form, or as a variable referred to by a CODE form, the locator is removed from the list; otherwise a one is added to a reference count that is associated with this variable. This reference count is included in the expression for all those variables still remaining on the list after all the subexpressions in the scope of a variable have been examined. (If a block variable has a reference count of zero, a warning diagnostic message is issued and the variable is discarded from the binding list.)

3. NET PROCESSING

To insure that all values for EVALGO's to a given confluence point in a net can be found in the same place, the Machine Link performs the same type of logic to the assignment of registers as was used to insure that all values reaching a given confluence point were in a given format. To accomplish this, symbolic register assignments corresponding to the confluence point in the embedding net are placed in EVALGO's so that their values can be top-driven later to this register. When processing the net forms, as decisions are made to use function calls to perform conversions, the confluence points have the appropriate argument registers included with the confluence point. If an in-line sequence which may use arbitrary registers is to be used, a generated register name is included with the confluence points.

4. IN-LINE SEQUENCE DETECTION

Certain forms have been treated as though they would be calls on functions, because they require strong driving to the proper format of their arguments. This pass looks at the function name inside all FNCALL forms, and if it is included in this special list, removes the FNCALL and uses the function name as the form name.

The things discovered in this may include:

WORDOR, WORDAND, WORDXOR, SHIFT, SCAE, CYCLE,

IQUOTIENT, RECIP

5. MINUS OPTIMIZATION

When a minus form is encountered, the embedded expression is examined.

If the embedded expression is a minus form, both minuses are removed.

If the embedded expression is a plus form, the outer minus is removed and applied to each argument of the plus.

If the embedded expression is a times form, the minus is applied only to the first argument.

If the embedded expression is a datum, the minus is removed and the datum is changed to its negative.

If the embedded expression is a recip, the minus is moved inside to apply to the argument of the recip.

When processing times forms, if minus forms exist within (at the top level or within recip's at the top level), the minus forms are removed. If an even number were removed, processing is complete. If a datum can be found to which a minus form can be applied, the form is applied; if a plus can be found, a minus form is applied to that; otherwise a -1 datum is added to the argument list.