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- 3417/300/00

AUTHOR Jeff Barnett
J. Barnett

TECHNICAL Jeff Barnett
J. Barnett

RELEASE Clark Weissman
C. Weissman, S.D.C.
[illegible]
D. Anschultz, I.I.I.

for J. I. Schwartz

DATE 4/26/67 **PAGE 1 OF** 8 **PAGES**

(Page 2 Blank)

LISP 2 Compiler Specifications

ABSTRACT

This document presents an overview of the LISP 2 compiler proposed for the IBM S/360 computer. It includes a brief description of the various passes of the compiler and their functions. Other documents in the series are referenced.

1. INTRODUCTION

The LISP 2 compiler design is for a large multipass optimizer. User-level inputs can be in Source Language, Intermediate Language, or LAP. Source Language is of the ALGOL-like, infix variety; Intermediate Language is a Polish prefix, list-structure resembling LISP 1.5; LAP is the symbolic assembly language used by the system.

2. DESIGN PRINCIPLES

The compiler design is based on several principles: It is expected that the LISP 2 compiler will operate in a time-sharing environment, therefore compilation is incremental. That is, one function definition at a time may be compiled, assuming appropriate declaration information for referenced variables is made available. The multipass structuring makes possible the easy determination of overlays and allows each pass to use its own list spaces. At the end of each pass, these special list spaces may be abandoned instead of being garbage collected.

The passes are independent in that during pass N, the function being compiled and all embedded definitions are handled by pass N. It is possible also for the supervisory function to batch several compilations through the passes, i.e., do all pass ones, do all pass twos, etc.

Each pass of the compiler has a stated purpose. The first five passes resolve a particular question and do a language transformation that makes the answer apparent. The sixth pass is a more conventional compiler pass that generates LAP code. The final pass is the assembler, LAP. The passes and their functions are listed below:

Pass	Name	Function
I	Syntax Translator	Translates SL to IL and resolves questions concerning hierarchy and precedence. (The Syntax Translator is produced by a meta-compiler and will not be described in this document series.)
II	Context Resolver	Translates IL to CRIL and resolves questions concerning denotation of named objects in the IL program. Also macro expansions and label checking are done.
III	Type Resolver	Translates CRIL to TRIL and resolves questions about type conversions and confluence-point branching. All format and type conversions, as well as branches of returns, are made explicit.

Pass	Name	Function
IV	Machine Link	Translates from TRIL to MSIL and resolves questions as to what is to be accomplished by function calls and what is to be generated in-line. Further machine-dependent precision problems are solved. The other function of this pass is determining eligibility of each lexical variable for holding in a register rather than being placed on the stack. The number of references to each lexical variable are counted.
V	Register Counter	Translates from MSIL to RCIL and resolves questions about register needs of various subexpressions. The number of registers, general-purpose and floating, is determined and the information is put in the RCIL output.
VI	Code Generator	Produces the LAP code equivalent of the RCIL input. The peepholer, a part of the code generator, does certain optimizations on the produced LAP code that are difficult to do elsewhere.
VII	LAP	Produces an octal core image and puts it in core or secondary storage by an appropriate plant routine.

3. COMPILER FUNCTIONS

An overview of the various LISP 2 compiler functions and the languages involved is presented in Tables 1 and 2 and Figure 1.

Declarations are absorbed by a declaration scanner which is operated by the supervisor before input to the context resolver. Declarations for all free references from a form to be compiled are included in the form with its output from the context resolver. Hence, it is not necessary for subsequent compiler passes to interact with the external system declaration mechanisms.

Table 1. Description of Compilation Functions

Name	Input	Output	Function
Supervisor	S-file	-	Coordinates the various sub-functions of the compilation process
Token	Characters	Token	Parses characters into tokens
Read	Token	S-Expression	Transforms a token stream into an S-expression
ILREAD	S-Expression	S-File	Transforms an IL entity into an S-file
EDREAD	S-Expression	S-File	Probably a NOP
Syntax Translator	Token	S-File	Transforms a sequence of tokens (Source Language) into an S-file
SREAD	-	S-File	A switch used for reading variously formatted inputs
Declaration Scan	S-File	Declaration List	Scans an S-file and absorbs all global declarations made
Context Resolver	IL	CRIL List	Macro-expands an IL input into a list of CRIL function definitions
Type Resolver	CRIL	TRIL	Transforms a CRIL input into TRIL by making type conversions and confluence-point branches explicit
Machine Link	TRIL	MSIL	Determines what will be accomplished by in-line code generation as opposed to function calls; solves machine-related precision problems
Register Counter	MSIL	RCIL	Counts and remembers register needs of subexpressions
Code Generator	RCIL	LAP	Compiles RCIL into LAP assembly language
Peepholer	LAP	LAP	Optimizes symbolic assembly language
LAP	LAP	CIM	Assembles LAP into an octal core image
Plant	CIM	-	Plants octals either in core or on core image secondary storage

Table 2. Language Description

Mnemonic	Name	Description
SL	Source Language	ALGOL-like language made of a string of tokens
IL	Intermediate Language	Polish prefix, list structure
S-FILE	Software File	A list of IL entities with a name; this is editor format
CRIL	Context-Resolved Interlude Language	A prefix, list-structured language with denotation of named objects explicit
TRIL	Type-Resolved Interlude Language	A prefix, list-structured language with type conversions and terminal branches explicit
MSIL	Machine-Specific Interlude Language	A prefix, list-structured language with decisions made on in-line code generation
RCIL	Register-Counted Interlude Language	Like MSIL with notation to express register needs of subexpressions
LAP	LISP Assembly Program Language	The LISP 2 assembly language
CIM	Core Image	A list of octals ultimately to be included in a LISP 2 system as a binary program

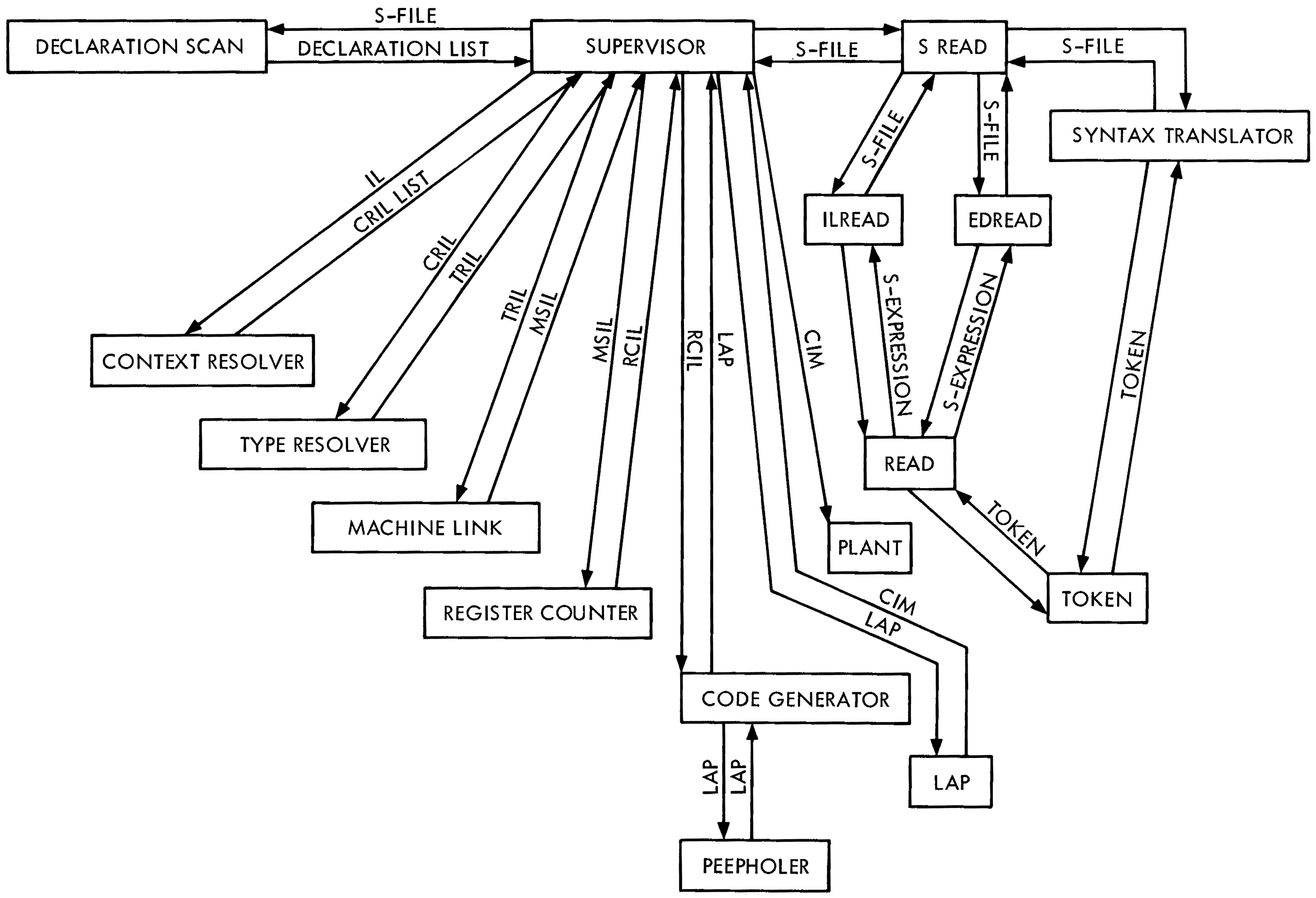

Figure 1. LISP 2 Compiler Flow Diagram

The context resolver introduces some machine-dependent precision information as part of its job. However, the process from syntax translator through the conclusion of the type resolution pass is essentially machine-independent. Other syntax translators and context resolvers may be used to produce CRIL representation of languages other than LISP 2. The optimization process begins with the type resolver pass.

The supervisory program exercises a great deal of control on the compilation process. By means of PUBLIC variables, the output of any or all passes may be printed. Further, the compilation may be halted at any pass, the interlude language stored, and the compilation restarted at another time. An error in any pass will halt the compilation after that pass. Variables involved with the garbage collection and swapping heuristics will be bound by the supervisor to reflect pertinent facts about allocation to be used while compiling. The system action during compilation may be radically different than for other uses of LISP 2.

4. RELATED DOCUMENTS

Other documents having volume numbers 300 to 399 in this series describe the various passes of the LISP 2 compiler and the interlude languages produced by them. Refer to TM-3417/200/00 for descriptions of SL and IL, and TM-3417/400/00 for a description of LAP.