

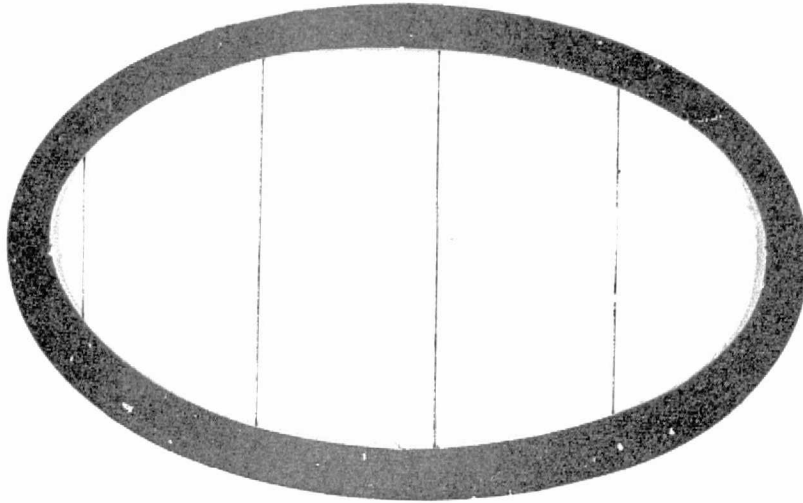
General Disclaimer

One or more of the Following Statements may affect this Document

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.
- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.
- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.
- This document is paginated as submitted by the original source.
- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.

NASA CR.

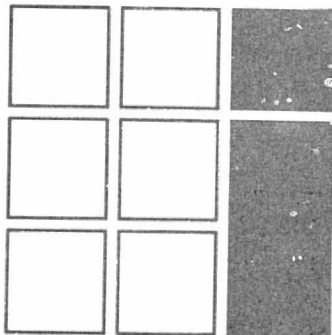
144652



(NASA-CR-144652) ON THE PERFORMANCE OF THE
HAL/S-FC COMPILER (Intermetrics, Inc.) 80 p
HC \$5.00 CSDL 09B

N76-15796

Unclas
G3/60 08471

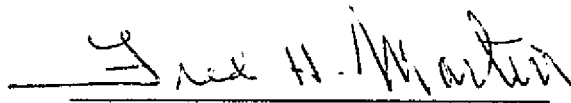


INTERMETRICS

On the Performance
of the
HAL/S-FC Compiler*
IR-162
October 22, 1975

*As reported at the HAL/S Configuration Inspection
held at Intermetrics on July 16, 1975.

Submitted by:


Dr. Fred H. Martin

Preface

This report was prepared for the National Aeronautics and Space Administration, Johnson Space Center under contract NAS9-13864.

Acknowledgement

The author wishes to acknowledge the many contributors to the HAL/S Acceptance Test activity. Within Intermetrics, Ron Kole assumed principal responsibility for the definition, organization and conduct of the test effort. Don Sakahara performed most of the tests and collected and organized the data. Dan Lickly and Arra Avakian helped to analyze the data and suggest iterations and new design approaches.

The whole exercise could not have been a success without the cooperation of Al Mandelin and Gary Pew of IBM/Houston, the dedicated efforts of Jim Marple and his associates from IBM/OWEGO, and the coordination provided by Jack Garman and Josephine Jue of NASA/SSD.

TABLE OF CONTENTS

	<u>PAGE</u>
TABLE OF CONTENTS	iii
LIST OF FIGURES	v
1.0 INTRODUCTION AND BACKGROUND	1
1.1 Compiler Status	1
1.2 Development Experience	6
1.3 Performance Objectives	9
2.0 ACCEPTANCE TEST OBJECTIVES AND PROCEDURES	11
2.1 Acceptance Test Groundrules	12
2.2 Test Objectives	15
2.2.1 Primary Objectives	15
2.2.2 Secondary Objectives	15
2.2.3 Data Collection	16
2.2.3.1 End-to-End vs. No Libraries	17
3.0 TEST ROUTINES AND RESULTS	19
3.1 Functional Description of Each Routine	19
3.1.1 The "GNC" Subset	19
3.1.2 The "UI" Subset	20
3.1.3 The "SM" Subset	20
3.2 Summary of Programming Features	20
3.3 Test Results	22
4.0 DATA ANALYSIS	41
4.1 Measure Definitions (Formulas), in %-Inefficiency	42
4.2 Performance Weighting Factors Based on HAL/S-FC Shuttle Applicability	43
4.3 HAL/S-FC Size Performance	43
4.4 HAL/S-FC Speed Performance	45
4.4.1 End-to-End	45
4.4.2 Alternate Speed Criteria	45
4.4.3 Another Look at No-Libraries	45
4.5 Effect of Compiler Optimization	47
4.6 Effect of Programmer Experience	48

TABLE OF CONTENTS (CONTINUED)

5.0	CONCLUSIONS AND OBSERVATIONS	49
5.1	Summary of HAL/S Compiler Performance	49
5.1.1	Additional Test Routine Results and Comments	50
5.1.2	Comments on Programmer Experience	51
5.2	Some Observations on Test Methods and Criteria	51
5.3	Areas Identified for HAL/S Improvement	54
	REFERENCES	55
	APPENDIX	57

LIST OF FIGURES

<u>FIGURE</u>		<u>PAGE</u>
1-1	HAL/S Development Background	2
1-2	360 Compiler Release Summary	3
1-3	FC Compiler Release Summary	3
1-4	HAL/S-360 Configuration Index	4
1-5	HAL/S-FC Configuration Index	5
1-6	LCR's vs. Compiler Releases	7
1-7	Latency Function for HAL/S-360	7
1-8	Discrepancies Introduced by Release for those Found in Releases 360-8, 9, 10, 11, 12	8
1-9	Turning to HAL/S-FC	8
2-1	The Testing Process	14
3-1	Language Features Exercised within Test Procedures	21
3-2	Test Description - SECOND_ORDER_FILTER	24
3-3	Test Description - MEASINCORP	25
3-4	Test Description - G_FILTER	26
3-5	Test Description - GNC_TACAN_AZ	27
3-6	Test Description - ELCOM	28
3-7	Test Description - GCB_YR_CE	29
3-8	Test Description - GRI_RGA_FDIR	30
3-9	Test Description - GRE_3ELEM	31
3-10	Test Description - GGJ_AL_FDCMD	32
3-11	Test Description - DMC_DISPLAY	33
3-12	Test Description - DMC_NEW_DISPLAY	34
3-13	Test Description - DMC_FILL_BACK_GROUND_FCWS	35
3-14	Test Description - SAS_ANALOG_SCALE	36
3-15	Test Description - SAS_POLYSOL	37
3-16	HAL/S-FC Acceptance Test Raw Data	38
3-17	Size and Speed of Original and Improved HAL/S Source using FC-8	39
4-1	HAL/S-FC Size Performance	44
4-2	HAL/S Speed Performance (End-to-End)	44
4-3	An Alternate Speed Criterion	46
4-4	HAL/S-FC Speed Performance (All Cases)	46
4-5	HAL/S-FC Speed Performance (Cases with NO Libraries Only)	47
5-1	Time/Space Trade-off	53
5-2	Improvement Curve (Test No. 3 - G_FILTER)	53

1.0 INTRODUCTION AND BACKGROUND

1.1 Compiler Status

On July 16, 1975 NASA/SSD conducted the HAL/S Configuration Inspection (CI) at Intermetrics in Cambridge, MA. As a result of this meeting the HAL/S compilers were formally accepted by NASA for use on the Space Shuttle Program.

In a sense the CI was the culmination of 5 years of development in bringing HAL from an RTOP concept to an operational language and compiler. Figure 1-1 traces this development showing the contracting party as well as the general objective of the work. Figures 1-2 and 1-3 list all the HAL/S compiler deliveries up to the date of CI. Of course the earlier versions were developmental milestone releases.

As of the CI, the current status of the compilers may be summarized as follows:

- The HAL/S-360 compiler is operational and in daily use at several installations.

This compiler accepts HAL/S source and emits IBM 360 machine code for either: (1) any compatible IBM 360/370 computer; (2) the Software Development Laboratory (SDL) at NASA/JSC.

- The HAL/S-FC compiler is operational and in daily use at IBM/Houston and Rockwell/Downey.

This compiler accepts HAL/S source and emits IBM AP-101 machine code. This code may be executed on: (1) the AP-101 itself; (2) the AP-101 interpretive computer simulator within the SDL or on any compatible IBM 360/370 computer.

- All User and System Documentation has been published and is up to date.

- A maintenance document is under development.

In order to properly baseline the compilers, a Configuration Index was prepared and presented at the CI. This Index, in terms of 360/OS entities is shown in Figures 1-4 and 1-5.

HAL/S DEVELOPMENT BACKGROUND

- 1970 RTOP (JSC) DEVELOPMENT OF HAL AND HAL/FORTRAN COMPILERS
- 1971 JSC CONTINUED DEVELOPMENT TO OPERATIONAL STATUS
- 1972 ROCKWELL DEFINITION OF HAL/S AND DEVELOPMENT OF HAL/S-360 COMPILER
- 1973 ROCKWELL START OF HAL/S-FC AND CONTINUED HAL/S-360
- 1974 JSC DELIVERY OF OPERATIONAL HAL/S-360 AND HAL/S-FC COMPILERS
- 1975 IBM MAINTENANCE AND OPTIMIZATION OF COMPILERS

Figure 1-1

360 COMPILER RELEASE SUMMARY

<u>DEVELOPMENT</u>		<u>MAINTENANCE</u>	
360-1	APRIL 13, 1973	360-12	APRIL 16, 1975
360-2	JUNE 15, 1973	360-12.1	APRIL 23, 1975
360-3	AUG. 1, 1973	360-12.2	APRIL 30, 1975
360-3.1	AUG. 30, 1973	360-12.3	MAY 7, 1975
360-4	OCT. 1, 1973	360-12.4	JUNE 4, 1975
360-5	NOV. 19, 1973	360-12.5	JUNE 25, 1975
360-6	FEB. 25, 1974		
360-7	MAR. 29, 1974		
360-8	JULY 8, 1974		
360-8.1	JULY 13, 1974		
360-9	AUG. 9, 1974		
360-9.1	AUG. 30, 1974		
360-9.2	SEPT. 5, 1974		
360-10	OCT. 29, 1974		
360-10.1	NOV. 29, 1974		
360-11	FEB. 3, 1975		
360-11.1	MARCH 23, 1975		

Figure 1-2

FC COMPILER RELEASE SUMMARY

<u>DEVELOPMENT</u>		<u>MAINTENANCE</u>	
FC-1	JAN. 22, 1974	FC-8	APRIL 15, 1975
FC-2	APRIL 29, 1974	FC-8.1	APRIL 23, 1975
FC-3	JULY 19, 1974	FC-8.2	APRIL 30, 1975
FC-4	SEPT. 27, 1974	FC-8.3	MAY 7, 1975
FC-4.1	NOV. 6, 1974	FC-8.4	JUNE 4, 1975
FC-4.2	NOV. 15, 1974	FC-8.5	JUNE 21, 1975
FC-5	NOV. 29, 1974		
FC-6	JAN. 6, 1974		
FC-6.1	FEB. 13, 1975		
FC-6.2	MARCH 2, 1975		
FC-7	MARCH 31, 1975		

Figure 1-3

HAL/S-360 CONFIGURATION INDEX

DSN	DESCRIPTION	APPROX. SIZE
HALS360.MONITOR	OS Load Module - Contain several programs: <ul style="list-style-type: none"> • MONITOR - Control Segment of Compiler • HALLINK - HAL/S-360 Link Editor • RUNMON - Control Segment of Stand Alone Diagnostic System • SUBMON - Control Segment for Diagnostic Request Language Processor • SDFPKG - Simulation Data File Access Package 	15K bytes 8K bytes 12K bytes 5K bytes 8K bytes
HALS360.COMPIER	XPL Object Module - The executable HAL/S-360 Compiler	535K bytes
HALS360.ERRORLIB	OS Partitioned Source Module - Text for individual error messages issued by compiler	1200 cards
HALS360.RUNLIB	OS Load Module - Runtime Library for HAL/S-360	45K bytes
HALS360.DIAGPROC	XPL Object Module - Diagnostic Request Language Processor	25K bytes
HALS360.STPLIB	OS Load Module - Alternate FSIM Statement Processor Library	25K bytes
HALS.RUNASM	OS Partitioned Source Module - Source of runtime library (assembler)	12,400 cards
HALS.RUNMAC	OS Partitioned Source Module - Macro library for HAL/S-360 system (assembler)	300 cards
HALS.SDFPKG.ASM	OS Partitioned Source Module - Source of Simulation Data File Access Package (assembler)	1,600 cards
HALS.DIAG.STC.MACLIB	OS Partitioned Source Module - Macro library for Stand Alone Diagnostic System (assembler)	400 cards

HAL/S-360 CONFIGURATION INDEX (Con't)

DSN	DESCRIPTION	APPROX. SIZE
HALS.MONASM	OS Partitioned Source Module - Source for Compiler Monitor and HALLINK (assembler)	6000 cards
HALS.STPGEN.ASM	OS Source Module - Conditional assembly statements to generate alternate statement processors	40 cards
HALPASS1.REL12V5.SOURCE	XPL Source for Compiler: Phase I	15K cards
HALPASS2.REL12V5.SOURCE	XPL Source for Compiler: Phase II	13K cards
HALPASS3.REL12V5.SOURCE	XPL Source for Compiler: Phase III	4K Cards
XCOM.LMON29	OS Load Module - Control Segment for XPL Compiler	7K bytes
XCOMLINK.CMP68	XPL Object Module - XPL Compiler	134K bytes
XPL.LINKLIB5	XPL Source - Library for XPL compilations	150 cards
XPLZAP	XPL Object Module - Program for performing field patches to compiler modules	17K bytes

ORIGINAL PAGE IS
OF POOR QUALITY

Figure 1-4

HAL/S-FC CONFIGURATION INDEX

DSN	DESCRIPTION	APPROX. SIZE
HALS101.SUPPORT	OS Load Module - Contains support software for GPC <ul style="list-style-type: none"> • ASM101 - Assembler (As delivered by Owego) • LNK101 - Link Editor (As delivered by Owego) • SIM101 - Simulator (As delivered by Owego) • HALUCP - User control program for simulator to trap simulated I/O and provide diagnostic support 	31K bytes
HALS101.COMPILER	XPL Object Module - The executable HAL/S-FC Compiler	542K bytes
HALS101.RUNLIB	AP-101 Load Module - Runtime library for HAL/S-FC	6K HW
HALS101.NUCLEUS	Source - Link Editor control cards to include SVC and I/O handler	1 card
HALS101.MONITOR	OS Load Module containing control segment for Compiler	15K bytes
HALS101.ERRORLIB	OS Partitioned Source Module - Text for individual error messages issued by compiler	1200 cards
HALS101.ZCONLIB	AP-101 Load Module - Contains long, indirect branch addresses (ZCONS) for all library members and entry points	600 HW
HALSFC.RUNASM	OS Partitioned Source Module - Source for runtime library (AP-101 assembler)	8500 cards
HALSFC.RUNMAC	OS Partitioned Source Module - Macro library for HAL/S-FC system (AP-101 assembler)	200 cards
HALSFC.ZCONASM	OS Partitioned Source Module - Source for ZCON library (AP-101 assembler)	1200 cards

HAL/S-FC CONFIGURATION INDEX (CON'T)

DSN	DESCRIPTION	APPROX. SIZE
HALS.MONASM	OS Partitioned Source Module - Source for compiler monitor (360 assembler)	3000 cards
HALS.RUNMAC	OS Partitioned Source Module - Macro library for compiler monitor (360 assembler)	200 cards
HALPASS1.REL12V5.SOURCE	XPL Source for compile: Phase I	15K cards
HALPASS2.REL18V5.SOURCE	XPL Source for compile: Phase II (FC code generator)	14K cards
HALPASS3.REL12V5.SOURCE	XPL Source for compile: Phase III	4K cards

ORIGINAL PAGE IS
OF POOR QUALITY

Figure 1-5

1.2 Development Experience

The HAL/S compilers have been developed in a changing environment but under configuration control since December, 1972. All changes and discrepancies have been formally reported through Change Reports and Discrepancy Report mechanisms, i.e.

LCR's for Language Change Requests
PCR's for Program Change Requests (non-language)
ICR's for Interface Change Requests (to SDL, FCOC, etc.)
DR's for Discrepancy Reports

A composite summary of this activity shows that as of June 24, 1975:

132 LCR's were submitted, 77 were implemented;
54 PCR's were submitted, 29 were implemented;
327 DR's were recorded, 260 were applicable.

It is interesting to note the trends concerning these data. As indicated in Figure 1-6, LCR activity has been steadily decreasing from a mid-development high of 14, implemented in 360-8 (July 1974), to only 2 new language features in the CI compiler releases of June, 1975. PCR activity, on the other hand, is still fairly active, reflecting desired changes in system interfaces, diagnostics and code optimization improvements. This activity may be expected to continue through the fall of 1975 based on recent decisions to go ahead with AP-101 micro code changes and further HAL/S-FC code generation optimization.

The patterns of verified discrepancies, up to CI, have been analyzed and are plotted in Figures 1-7 and 1-8 in terms of "latency" value. Latency is defined as the number of releases that a bug remains in the system between introduction and discovery. Figure 1-7 seems to indicate that for HAL/S-360 the bugs found are not particularly correlated with the most recent compiler change activity (i.e. the most recent compiler releases). This is better shown in Figure 1-9 where, for example, discrepancies found in 360-12 (dotted line) could have been introduced in almost any previous release. The data suggests the following tentative conclusions:

LCR'S vs COMPILER RELEASES

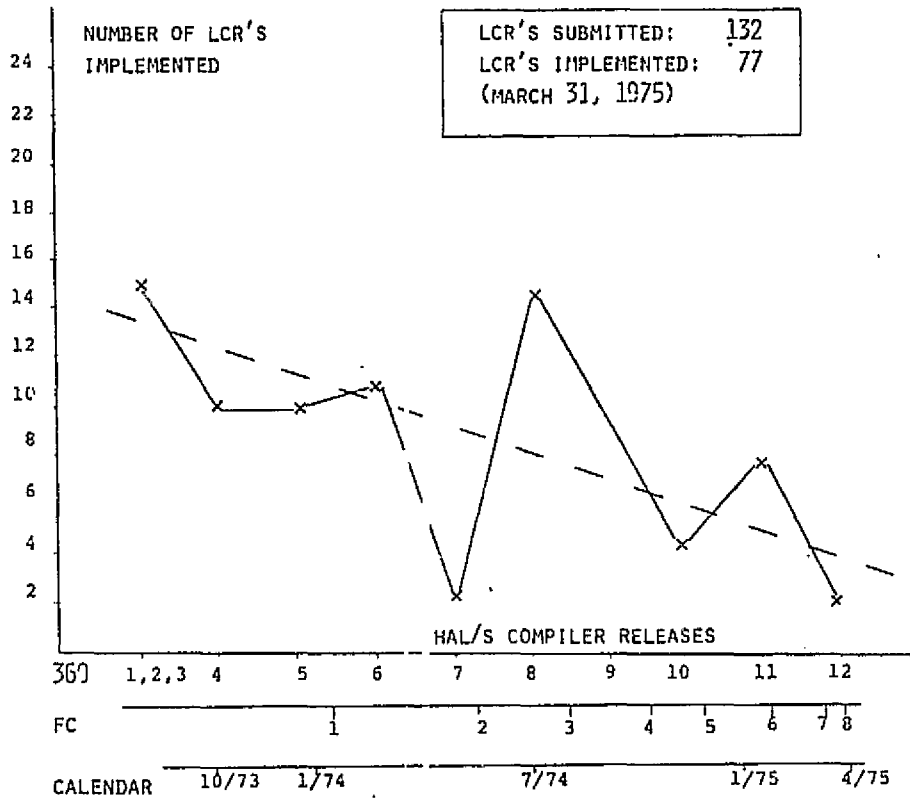


Figure 1-6

LATENCY FUNCTION FOR HAL/S-360
(AGGREGATE OF 360-8, 9, 10, 11, 12)

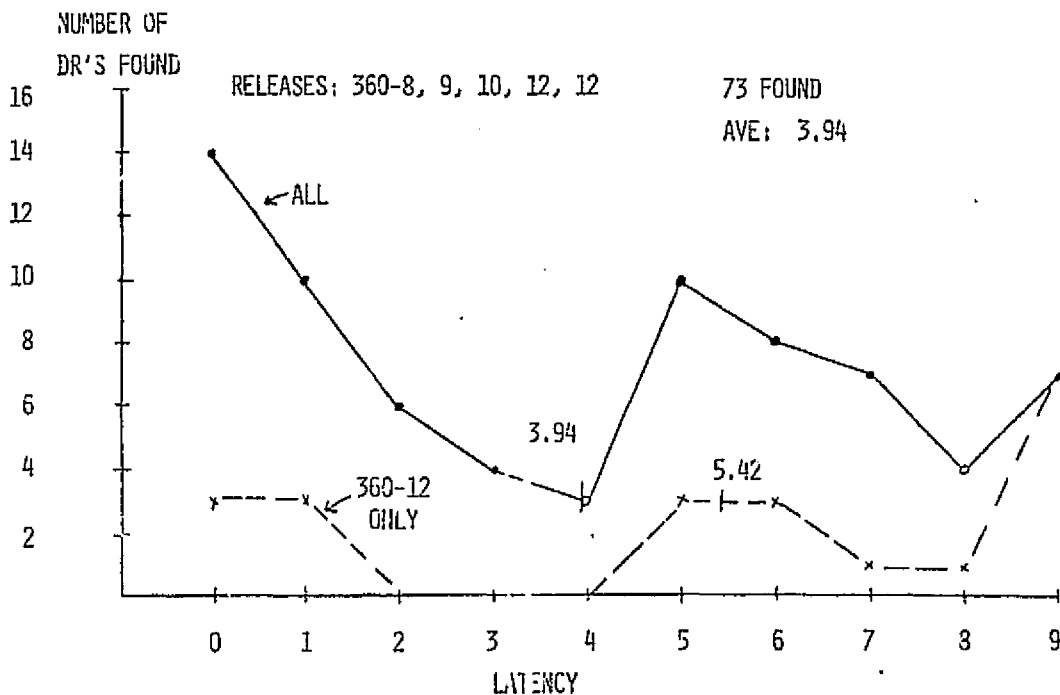


Figure 1-7

DISCREPANCIES INTRODUCED BY RELEASE
 FOR THOSE FOUND IN RELEASES
 360-8, 9, 10, 11, 12

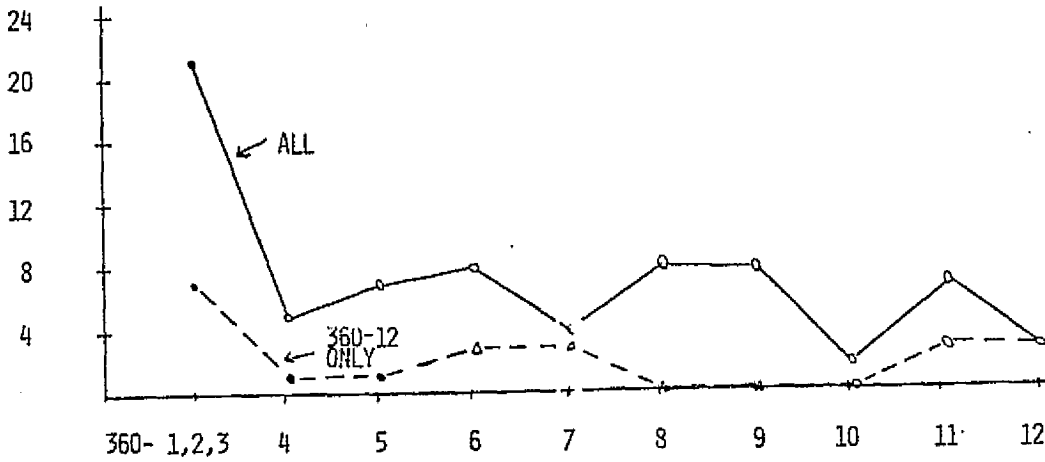


Figure 1-8

TURNING TO HAL/S-FC

- LATENCY FUNCTION IS MORE HIGHLY CORRELATED
- RECENT CODE ACTIVITY AFFECTS BUG DISCOVERY TO A GREATER EXTENT

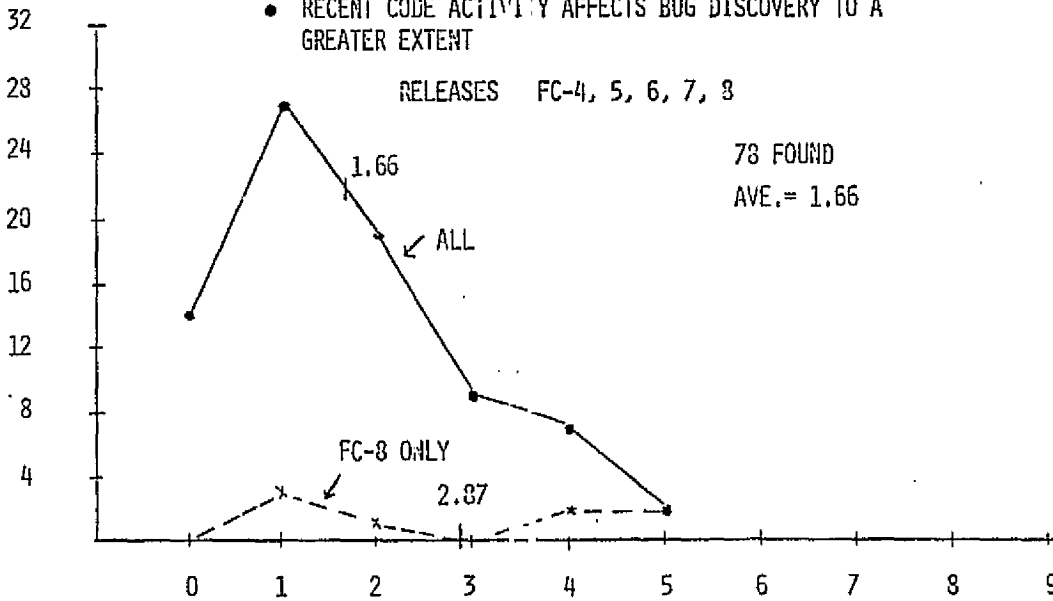


Figure 1-9

- For HAL/S-360
 - 1) Newly discovered bugs appear "equally likely" to have been introduced in any of the previous releases.
 - 2) LCR, PCR and DR activity does not of itself introduce a significant number of secondary bugs.
 - 3) The HAL/S-360 compiler is a stable, mature, operational program.
- For HAL/S-FC
 - 1) Newly discovered discrepancies are more correlated to recent change activity. This is probably due to the fact that FC development has been characterized by new optimization features in addition to maintenance.
 - 2) HAL/S-FC is less stable than HAL/S-360.
 - 3) However, the HAL/S-FC compiler is operational and its absolute DR performance record and frequency are tolerable.

It should be noted that the number of DR's reported is a function of both compiler integrity and the frequency of use. The more users, the greater the chance of finding something. The non-uniformity of use over compiler development will tend to distort statistical inferences and any conclusions should be tempered by this fact.

1.3 Performance Objectives

The performance objectives for the HAL/S compilers were established at the Preliminary Design Reviews and documented in the Compiler System Functional Specification documents^{1,2}. Of particular interest here is the stated HAL/S-FC requirement on generated object code:

"The object code produced will not exceed that produced via hand-coded assembler language methods by more than 15% in either memory requirements or execution time."²

The corresponding HAL/S-360 performance was set at 20%.

A major portion of the CI was devoted to the presentation of data demonstrating the achieved performance of the HAL/S compilers. This data was collected during the first half of 1975 as the result of a coordinated "acceptance test" activity among Intermetrics, NASA, IBM/Houston, and IBM/OWEGO. The emphasis was on the HAL/S-FC compiler.* As a result of the CI acceptance tests and procedures, Intermetrics was able to derive the following:

HAL/S-FC size inefficiency: 11-13% over assembler language
HAL/S-FC speed inefficiency: 9-11% over assembler language

Based on examination of the data, methods and conclusions, NASA accepted the HAL/S-FC compiler.

In the sections that follow the acceptance test objectives and procedures are first described, the raw results are then presented and analyzed and finally, conclusions and observations are drawn. An appendix is also included containing an illustrative set of compiler listings and results for one of the test cases.

* Intermetrics did present some interesting but inconclusive data concerning the HAL/S-360.
-10-

2.0 ACCEPTANCE TEST OBJECTIVES AND PROCEDURES

The effort to define and conduct compiler acceptance testing was formulated under NASA direction in January, 1975. The overall plan called for establishing a representative set of benchmarks, coding them in both HAL/S and AP-101 assembler language and then comparing performance. It was recognized very early that the higher order language/assembler language comparisons are influenced by many variables and may be subject to varying interpretations. In order for the exercise to have acknowledged validity it would have to be carefully designed and controlled, with all parties participating and cognizant of its progress.

Intermetrics, the compiler developer, and IBM/Houston, the compiler user, approached the task from different points of view. Intermetrics first suggested benchmarks that represented theoretical HAL/S usage. That is, the selected routines contained a wide sample of HAL/S constructs designed to correspond to expected Shuttle applications. IBM was more interested in HAL/S performance as it related to "real" Shuttle code. Many of their sizing and timing estimates for the Shuttle Approach and Landing Test (ALT) and Operational Flight Program (OFP) were based on a HAL/S performance of 15% in size and speed and they desired a true reading of the delivered product. The IBM approach was adopted and 14 test routines were established as the acceptance set. The routines were selected from HAL/S coding done by IBM/Houston, Intermetrics and Draper Laboratory. Each routine was approved by both Intermetrics and IBM before being incorporated into the set.

Because of time and resource limitations, it was decided to use the routines themselves as the test specifications instead of generating abstract word representations. In other words, once establishing that originally coded routines would properly execute, they formed the test baseline from which the HAL/S vs. assembler language exercise could begin.

Intermetrics was immediately aware of the fact that the "performance" of these routines would not compare favorably with assembler language and in fact would be significantly improved in a straightforward manner by simply changing the HAL/S source code. That is, a good assembler language coder could be expected to do better than the compiler on these baseline routines, but so should a good HAL/S programmer. The separation of effects on performance, of compiler design and human source code ingenuity, proved to be a continuing problem. In the final analysis it was partially accounted for by numerous iterations and cross-fertilizations of the resulting HAL/S and assembler language solutions.

In order to insure an objective comparison, and also inject an element of competition, IBM/OWEGO was selected to program the routines in AP-101 assembler language while Intermetrics would produce the HAL/S code. It was very important to establish a set of groundrules, at the outset, that would result in valid comparisons and stand up under scrutiny. There would be plenty of opportunity for misinterpretations due to run time environments, different conventions, presumptions about data, timing algorithms, etc.

2.1 Acceptance Test Groundrules

1. The original set of 14 routines coded in HAL/S would serve as the test baseline.
2. Execution results would be based on initialization data for each routine. These data areas would be established and held constant throughout the testing exercise.
3. Assembler language routines would be coded as total substitutes for corresponding HAL/S modules, i.e.
 - a) FCOS interfaces would be maintained.
 - b) the data base established for the HAL/S routines would be frozen and accessed by assembler language code.
 - c) where assembler language routines needed and called library functions already existing in the HAL/S system, these same library routines were to be used with the same conventions as HAL/S.
4. READ and WRITE statements were to be placed in routines calling the tests so that the data base before and after execution could be observed and the results easily compared.
5. All timing information was to be gathered using the AP-101 interpretive computer simulator running in HI-FI mode.
6. While meeting the above requirements, the programmers, using either language, were free to improve the size and/or speed performance of their routines by such

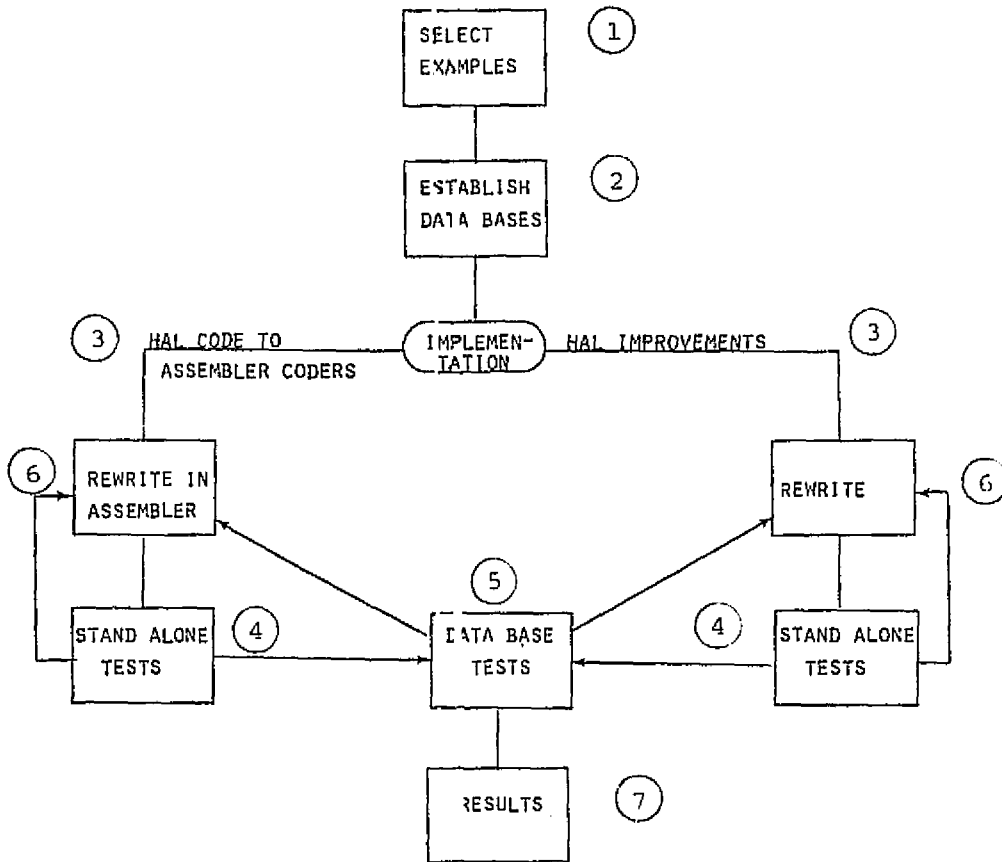
devices as:

- in-line loops vs. subroutines
- straight line code vs. loops
- common sub-expression elimination
- redesign of execution order
- elimination of redundant decision points
- etc.

With respect to this last point (6), there was no doubt that both groups would be striving to compare "best" HAL/S against "best" assembler language. This was felt justified because of the iterative aspect of improvement. That is, given a routine already coded, it seems always (or almost always) possible to improve it. This occurred time and again during this acceptance effort. On many occasions Intermetrics would receive from IBM/OWEGO an assembler code version which out-performed, by far, the latest attempt using HAL/S. More often than not the HAL/S code could be further improved, sometimes exceeding assembler language performance. The reverse was also true. By the CI* both groups were convinced that each had benefited from the designs and "clever" ideas of the other, and that for the most part the results reflected compiler performance and not human ingenuity.

Figure 2-1 depicts the process described above. After selecting the examples (1), and establishing the data bases (2), each group attempted to run and improve their respective routines (3). At first execution was stand-alone (4) in order to check out operation, interfaces, use of simulator, etc. Eventually the official data base was used (5), results were compared and the routines rewritten (6) to improve performance. Best against best, and the pressure of schedule, produced the final results (7).

*In fact, coincident with CI and for a few weeks thereafter IBM/Houston saw further areas for improvement in the Assembler language code. Using similar design approaches Intermetrics re-worked one of the corresponding HAL/S routines and also experienced better performance. The raw data related to this post-CI exercise is included in the discussions in Section 5.



THE TESTING PROCESS

Figure 2-1

ORIGINAL PAGE IS
OF POOR QUALITY

2.2 Test Objectives

2.2.1 Primary Objectives

Simply stated, the primary objective of the acceptance test exercise was to establish the performance of the HAL/S-FC compiler with respect to the Shuttle applications programming task. The performance was to be stated in terms of size and speed inefficiencies over comparable assembler language coding.

IBM and NASA had previously decided to code the Flight Computer Operating System (FCOS) in assembler language, thus performance demonstration was confined to the application areas of: guidance, navigation and control (GN&C); user interface (UI); and systems management (SM).

2.2.2 Secondary Objectives

The collection of a large amount of compiler performance data afforded the opportunity of reporting on several other interesting characteristics of HAL/S development and usage. Since the baseline set of routines was compiled using FC-5, an unoptimized compiler, and new data was to be collected using FC-8, the demonstrated benefit of the FC-8 optimization features could be measured directly. This would be achieved simply by compiling and executing the identical source using both compilers.

In an effort to demonstrate "best" HAL/S performance, Intermetrics intended to improve the original sources. The differences in resulting performance between "old" and "new" sources, using the same compiler (FC-8), would then be a measure of the influence of programmer experience on size and speed.

Thus two measures became secondary objectives of the study:

- (1) degree of improvement due to optimization;
- (2) effect of programmer experience.

2.2.3 Data Collection

The data collected is best explained by discussing the "results template" used for each test routine. The templates for size and speed are shown below.

SIZE PERFORMANCE DATA:

(1)	(2)	(3)	(4)	(5)	(6)	(7)
Original HAL/S Code Size (HW)		% Improvement $\frac{(1)-(2)}{(1)}$	Improved HAL/S Code Size (HW)	% Improvement $\frac{(2)-(4)}{(2)}$	Independent Assembler Language Size (HW)	% HAL/S Inefficiency $\frac{(4)-(6)}{(6)}$
FC-5	FC-8		FC-8			

SPEED PERFORMANCE DATA:

(1)	(2)	(3)	(4)	(5)	(6)	(7)
Original HAL/S Code (μ sec)		% Improvement $\frac{(1)-(2)}{(1)}$	Improved HAL/S Code (μ sec)	% Improvement $\frac{(2)-(4)}{(2)}$	Independent Assembler Language (μ sec)	% HAL/S Inefficiency $\frac{(4)-(6)}{(6)}$
FC-5	FC-8		FC-8			
N.A.		N.A.				

The % measures are designed to answer the performance questions most often asked; i.e. what % improvement was achieved by a new compiler, or through programmer experience; and what % inefficiency is HAL/S over assembler code.

The results achieved for each routine are presented in terms of these templates in Section 3 of this report.

With respect to Size Performance, columns (1) and (2) indicate the code size in AP-101 half-words (HW) using FC-5 and FC-8 respectively. Column (3) contains the % improvement. Note that only code size will be measured, since both compiler and assembler language code utilize identical data areas. After the HAL/S source is improved, this "best" size is entered in column (4). Column (5) then reflects the improvement of "new" source over "old" source using FC-8. The assembler language size appears in column (6) and the HAL/S size inefficiency (over assembler language) is entered in the last column (7).

The speed data is similarly presented. Timing for original source using FC-5 was "not available"; therefore columns (1) and (2) will not contribute to the study.

2.2.3.1 End-to-End vs. No Libraries. The question of how to measure the execution time for a routine, and what is a proper comparison between HAL/S and assembler language caused a certain amount of difficulties in gathering and interpreting data. Two points of view were expressed:

- (1) the HAL/S compiler represented an integrated design of code generator and libraries and therefore timing data should be computed "end-to-end". That is, performance should be measured based on the total time it takes to execute a routine from entry to exit.
- (2) since the HAL/S libraries are themselves written in assembler language, the time spent in the libraries is not a true measure of compiler efficiency. Therefore, by subtracting the time in libraries from the total, the performance of the code generator can be deduced.

This second point of view was, in some cases, easier to state than to measure. No particular problem existed where both HAL/S and assembler language routines used the same libraries. But when the assembler language design elected to compute a function in-line and the HAL/S routine accessed a library function then direct comparison became difficult. In such cases it was decided to subtract the library time from the HAL/S total, and the "functional library" time from the assembler language total. The functional library being the in-line code performing the library function. On occasion it became difficult to decide what belonged to the function and what didn't.

For the most part Intermetrics subscribed to the first point of view, considering the HAL/S compiler product as an entity. IBM was inclined toward the second. In the final analysis, both types of data were collected and integrated before arriving at the final performance figures.

3.0 TEST ROUTINES AND RESULTS

Fourteen routines were selected to form the baseline set for acceptance testing. This set consisted of examples identified by IBM/Houston from their implementation effort, or contributions by Draper Laboratory and Intermetrics which closely resembled expected use. All routines were reviewed and approved by IBM and Intermetrics before being incorporated into the baseline.

3.1 Functional Description of Each Routine

3.1.1 The "GNC" Subset

Test No. 1: SECOND_ORDER_FILTER

This routine by the Draper Laboratory implements a second order discrete linear recursive filter.

Test No. 2: MEASINCORP

This routine by Intermetrics implements an optimal filtering algorithm which incorporates external measurements into position and time components of the state vector. 1 compool is involved.

Test No. 3: G_FILTER

This routine by Intermetrics implements a recursive linear least squares filter which estimates gyro drift on the basis of differencing platform altitudes. 1 compool is involved.

Test No. 4: This routine by IBM/Houston calculates the TACAN azimuth from the state vector, the measurement residual, and the vector of azimuth partials; the routine also computes the variance of the azimuth measurement error.

Test No. 5: ELCOM

This routine by IBM/Houston computes several Shuttle elevator commands.

Test No. 6: GCB_YR_CE

This routine by IBM/Houston represents the yaw/roll control element and performs aileron, rudder and nose-wheel processing for yaw/roll flight control modes.

Test No. 7: GRI_RGA_FDIR

This routine by IBM/Houston performs fault detection indication and recovery (FDIR) for the rate gyro assemblies.

PRECEDING PAGE BLANK NOT FILMED

Test No. 8: GRE_3ELEM

This routine by IBM/Houston performs the necessary 3-element processing as determined by the FDIR status.

Test No. 9: GGJ_AL_FDCMD

This routine by IBM/Houston performs functions for the Approach and Landing (A/L) Flight Director Command Processor for roll and pitch.

3.1.2 The "UI" Subset

Test No. 10: DMC_DISPLAY

This routine by IBM/Houston determines the starting location of one of four display format buffers. 6 compools are involved.

Test No. 11: DMC_NEW_DISPLAY

This routine by IBM/Houston performs control data maintenance and logic in order to output the background format control words (FCW's) to the display electronics units (DEU's).

Test No. 12: DMC_FILL_BACK_GROUND_FCWS

This routine by IBM/Houston locates the background FCW's and issues appropriate SVC's to send the FCW's to the DEU's.

3.1.3 The "SM" Subset

Test No. 13: SAS_ANALOG_SCALE

This routine by IBM/Houston converts parameters from pulse code modulation (PCM) units to engineering units, and vice versa. 5 compools are involved.

Test No. 14: SAS_POLYSOL

This routine by IBM/Houston performs several polynomial solutions as determined by SAS_ANALOG_SCALE.

3.2 Summary of Programming Features

Taken together the 14 routines exercise most of the programming features available in HAL/S. These are summarized in matrix form in Figure 3-1.

Language Features Exercised within Test Routines

Language Feature	Acceptance Test Number									10	11	12	13	14
	1	2	3	4	5	6	7	8	9					
Integer			✓			✓	✓	✓		✓	✓	✓	✓	✓
Scalar	✓	✓		✓	✓			✓	✓				✓	
Vector		✓	✓	✓										
Matrix		✓	✓	✓										
Bit Strings							✓	✓		✓	✓	✓		✓
Booleans	✓				✓				✓				✓	
Array	✓	✓	✓				✓	✓					✓	✓
Structure	✓				✓					✓	✓	✓	✓	✓
Subscript	✓	✓	✓	✓			✓	✓		✓	✓	✓	✓	✓
BUILT-IN FUNCTION				✓				✓	✓	✓		✓	✓	
IF_Then_ELSE	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	
DO CASE					✓		✓	✓				✓		
DO FOR		✓	✓				✓	✓		✓		✓		✓
DO WHILE										✓	✓	✓		
REPEAT/EXIT/RETURN										✓				
Procedure		✓		✓	✓	✓	✓			✓	✓		✓	
Function			✓		✓	✓			✓					
Compool		✓	✓							✓	✓	✓	✓	✓
REPLACE										✓	✓			
REENTRANT/AUTOMATIC													✓	
NAME Variables										✓		✓	✓	
TEMPORARY			✓	✓				✓		✓				

Figure 3-1

3.3 Test Results

The results of the acceptance tests are presented in Figures 3-2 to 3-15 in terms of a standard template for each routine. The specific numbers shown were achieved, in general, just before CI and reflect numerous iterations in trying to reach "Best" HAL/S and "best" assembler language. In the cases of Test Nos. 13 and 14, final results were obtained shortly after CI and have been included in this report.

In trying to improve the performance of the original routines by altering the HAL/S source code, Intermetrics was guided by a few general principles. These may be summarized as follows:

- 1) attempt to reduce the number of decision points in any control flow
- 2) use the structured control statements DO WHILE, EXIT and REPEAT instead of artificial data, e.g. flags, first time values, and additional IF-statements for loop control
- 3) use RETURN expressions for multiple FUNCTION return points instead of creating temporary variables for the same purpose
- 4) simplify expressions which are computed iteratively (i.e. within loops)
- 5) attempt to reduce the frequency of variable bit subscripting
- 6) when logic branches (DO CASE, IF) result in different selections of essentially the same code, try to eliminate explicit code by introducing subscripted forms
- 7) consider size/speed tradeoffs; execution time can be reduced, sometimes significantly, by unravelling inner loops
- 8) attempt to reduce frequency of very complicated subscripts through introduction of temporary NAME variables.

NOTE: items (4), (5) and (8) are now subjects of compiler optimization efforts.

A summary of data is presented in Figures 3-16 and 3-17. Figure 3-16 shows performance for "best" HAL/S vs. "best" assembler language, with and without libraries (also see discussion in Section 2.2.3.1). Figure 3-17 lists improvements in HAL/S results, using FC-8, attributable to programmer experience only.

As an example of the data collection process for a single routine, the computer listings associated with Test No. 8, GRE_3ELEM, are included in the Appendix. The set contains the original HAL/S source listing specifying the test, followed by the "best" source as modified by Intermetrics. Next comes the IBM/OWEGO assembler language code. The interpretive computer simulator trace shows execution and basic timing data. The final listing is that of the data base before and after the test. The data base transformation indicates whether or not the test executed successfully.

In preparation for the CI, a great deal of data was collected and recorded for the fourteen routines. It then became a question of meaningful analysis through the application of appropriate measures.

Figure 3-2

TEST DESCRIPTION

TEST NO.: 1 TEST NAME: SECOND_ORDER_FILTER SOURCE: Draper Laboratory

FUNCTIONAL GROUP: GNC

FUNCTIONAL DESCRIPTION: Second order discrete linear recursive filter

PROGRAMMING CHARACTERISTICS: Array subscripting, Scalar arithmetic

SIZE PERFORMANCE DATA:

(1)	(2)	(3)	(4)	(5)	(6)	(7)
Original HAL/S Code Size (HW)		% Improvement $\frac{(1)-(2)}{(1)}$	Improved HAL/S Code Size (HW)	% Improvement $\frac{(2)-(4)}{(2)}$	Independent Assembler Language Size (HW)	% HAL/S Inefficiency $\frac{(4)-(6)}{(6)}$
FC-5	FC-8		FC-8			
93	91	2.2	77	15.4	64	20.3

SPEED PERFORMANCE DATA:

(1)	(2)	(3)	(4)	(5)	(6)	(7)
Original HAL/S Code (usec)		% Improvement $\frac{(1)-(2)}{(1)}$	Improved HAL/S Code (usec)	% Improvement $\frac{(2)-(4)}{(2)}$	Independent Assembler Language (usec)	% HAL/S Inefficiency $\frac{(4)-(6)}{(6)}$
FC-5	FC-8		FC-8			
N.A.	249.7	N.A.	234.5	6.10	213.5	9.8
	356.3		345.3	3.10	326.7	5.7

ORIGINAL PAGE IS OF POOR QUALITY

Figure 3-3

TEST DESCRIPTION

TEST NO.: 2 TEST NAME: MEASINCORP SOURCE: Intermetrics

FUNCTIONAL GROUP: GNC

FUNCTIONAL DESCRIPTION: Optimal filtering algorithm to incorporate external measurements into position and time components of the state vector.

PROGRAMMING CHARACTERISTICS: n-dimensional Vector/Matrix, 1 Compool

SIZE PERFORMANCE DATA:

(1)	(2)	(3)	(4)	(5)	(6)	(7)
Original HAL/S Code Size (HW)		% Improvement $\frac{(1)-(2)}{(1)}$	Improved HAL/S Code Size (HW)	% Improvement $\frac{(2)-(4)}{(2)}$	Independent Assembler Language Size (HW)	% HAL/S Inefficiency $\frac{(4)-(6)}{(6)}$
FC-5	FC-8		FC-8			
550	450	18.2	316	29.8	268	17.9

SPEED PERFORMANCE DATA:

(1)	(2)	(3)	(4)	(5)	(6)	(7)
Original HAL/S Code (μ sec)		% Improvement $\frac{(1)-(2)}{(1)}$	Improved HAL/S Code (μ sec)	% Improvement $\frac{(2)-(4)}{(2)}$	Independent Assembler Language (μ sec)	% HAL/S Inefficiency $\frac{(4)-(6)}{(6)}$
FC-5	FC-8		FC-8			
N.A.	16327.8	N.A.	15507.6	5.0	14674.4	5.7

Figure 3-4

TEST DESCRIPTION

TEST NO.: 3 TEST NAME: G_FILTER SOURCE: Intermetrics

FUNCTIONAL GROUP: GNC

FUNCTIONAL DESCRIPTION: Recursive linear least squares filter which estimates gyro drift on the basis of differing platform attitudes.

PROGRAMMING CHARACTERISTICS: Arrays of 3-vectors, 1 Compool

SIZE PERFORMANCE DATA:

(1)	(2)	(3)	(4)	(5)	(6)	(7)
Original HAL/S Code Size (HW)		% Improvement $\frac{(1)-(2)}{(1)}$	Improved HAL/S Code Size (HW)	% Improvement $\frac{(2)-(4)}{(2)}$	Independent Assembler Language Size (HW)	% HAL/S Inefficiency $\frac{(4)-(6)}{(6)}$
FC-5	FC-8		FC-8			
677	606	10.5	308	49.2	256	20.3

SPEED PERFORMANCE DATA:

(1)	(2)	(3)	(4)	(5)	(6)	(7)
Original HAL/S Code (µsec)		% Improvement $\frac{(1)-(2)}{(1)}$	Improved HAL/S Code (µsec)	% Improvement $\frac{(2)-(4)}{(2)}$	Independent Assembler Language (µsec)	% HAL/S Inefficiency $\frac{(4)-(6)}{(6)}$
FC-5	FC-8		FC-8			
N.A.	2161.7	N.A.	976.3	54.8	874.1	11.7
	4432.9		1817.3	58.1	1857.9	-2.2

ORIGINAL PAGE IS
OF POOR QUALITY

Figure 3-5

TEST DESCRIPTION

TEST NO.: 4 TEST NAME: GNC_TACAN_AZ SOURCE: IBM

FUNCTIONAL GROUP: GNC

FUNCTIONAL DESCRIPTION: Calculates the TACAN azimuth measurement from the state vector, the measurement residual, and the azimuth partials vector and selects the variance of the azimuth measurement error.

PROGRAMMING CHARACTERISTICS: 3 vectors, 3x3 matrices. Some scalars

SIZE PERFORMANCE DATA:

(1)	(2)	(3)	(4)	(5)	(6)	(7)
Original HAL/S Code Size (HW)		% Improvement $\frac{(1)-(2)}{(1)}$	Improved HAL/S Code Size (HW)	% Improvement $\frac{(2)-(4)}{(2)}$	Independent Assembler Language Size (HW)	% HAL/S Inefficiency $\frac{(4)-(6)}{(6)}$
FC-5	FC-8		FC-8			
123	100	13.7	93	7.0	132	-29.5

SPEED PERFORMANCE DATA:

(1)	(2)	(3)	(4)	(5)	(6)	(7)
Original HAL/S Code (usec)		% Improvement $\frac{(1)-(2)}{(1)}$	Improved HAL/S Code (usec)	% Improvement $\frac{(2)-(4)}{(2)}$	Independent Assembler Language (usec)	% HAL/S Inefficiency $\frac{(4)-(6)}{(6)}$
FC-5	FC-8		FC-8			
N.A.	1050.2	N.A.	872.2	16.9	735.4	18.6

ORIGINAL PAGE IS
OF POOR QUALITY

Figure 3-6

TEST DESCRIPTION

TEST NO.: 5 TEST NAME: ELCOM SOURCE: IBM

FUNCTIONAL GROUP: GNC

FUNCTIONAL DESCRIPTION: Computes various elevator
command variables.

PROGRAMMING CHARACTERISTICS: IF-THEN-ELSE, Scalar Arrays
user function calls.

SIZE PERFORMANCE DATA:

(1)	(2)	(3)	(4)	(5)	(6)	(7)
Original HAL/S Code Size (HW)		% Improve- ment $\frac{(1)-(2)}{(1)}$	Improved HAL/S Code Size (HW)	% Improve- ment $\frac{(2)-(4)}{(2)}$	Independent Assembler Language Size (HW)	% HAL/S Inefficiency $\frac{(4)-(6)}{(6)}$
FC-5	FC-8		FC-8			
193	181	6.2	166	8.3	138	20.3

SPEED PERFORMANCE DATA:

(1)	(2)	(3)	(4)	(5)	(6)	(7)
Original HAL/S Code (μ sec)		% Improve- ment $\frac{(1)-(2)}{(1)}$	Improved HAL/S Code (μ sec)	% Improve- ment $\frac{(2)-(4)}{(2)}$	Independent Assembler Language (μ sec)	% HAL/S Inefficiency $\frac{(4)-(6)}{(6)}$
FC-5	FC-8		FC-8			
N.A.	448.8	N.A.	442.2	1.5	418.4	5.7
	459.6		446.0	3.0	422.6	5.5

Figure 3-7

TEST DESCRIPTION

TEST NO.: 6 TEST NAME: GCB_YR_CE SOURCE: IBM

FUNCTIONAL GROUP: GNC

FUNCTIONAL DESCRIPTION: Yaw/Roll (Roll/Yaw) Control Element will perform the aileron, rudder, and nosewheel processing of less than 25 Hz for all Yaw/Roll (Roll/Yaw) flight control modes.

PROGRAMMING CHARACTERISTICS: IF-THEN-ELSE, Integer and Scalar arithmetic.
User proc/func calls.

SIZE PERFORMANCE DATA:

(1)	(2)	(3)	(4)	(5)	(6)	(7)
Original HAL/S Code Size (HW)		% Improvement $\frac{(1)-(2)}{(1)}$	Improved HAL/S Code Size (HW)	% Improvement $\frac{(2)-(4)}{(2)}$	Independent Assembler Language Size (HW)	% HAL/S Inefficiency $\frac{(4)-(6)}{(6)}$
FC-5	FC-8		FC-8			
117	114	2.6	108	5.3	98	10.2

SPEED PERFORMANCE DATA:

(1)	(2)	(3)	(4)	(5)	(6)	(7)
Original HAL/S Code (μsec)		% Improvement $\frac{(1)-(2)}{(1)}$	Improved HAL/S Code (μsec)	% Improvement $\frac{(2)-(4)}{(2)}$	Independent Assembler Language (μsec)	% HAL/S Inefficiency $\frac{(4)-(6)}{(6)}$
FC-5	FC-8		FC-8			
N.A.	70.1	N.A.	70.1	0.0	68.3	2.6
	122.1		116.3	4.8	113.9	2.1

Figure 3-8

TEST DESCRIPTION

TEST NO.: 7 TEST NAME: GRI_RGA_FDIR SOURCE: IBM

FUNCTIONAL GROUP: GNC

FUNCTIONAL DESCRIPTION: Performs FDIR functions for the rate gyro assemblies.

PROGRAMMING CHARACTERISTICS: Bit strings, ANDing, ORing, Bit partitioning

SIZE PERFORMANCE DATA:

(1)	(2)	(3)	(4)	(5)	(6)	(7)
Original HAL/S Code Size (HW)		% Improvement $\frac{(1)-(2)}{(1)}$	Improved HAL/S Code Size (HW)	% Improvement $\frac{(2)-(4)}{(2)}$	Independent Assembler Language Size (HW)	% HAL/S Inefficiency $\frac{(4)-(6)}{(6)}$
FC-5	FC-8		FC-8			
269	267	0.7	165	38.2	137	20.4

SPEED PERFORMANCE DATA:

(1)	(2)	(3)	(4)	(5)	(6)	(7)
Original HAL/S Code (µsec)		% Improvement $\frac{(1)-(2)}{(1)}$	Improved HAL/S Code (µsec)	% Improvement $\frac{(2)-(4)}{(2)}$	Independent Assembler Language (µsec)	% HAL/S Inefficiency $\frac{(4)-(6)}{(6)}$
FC-5	FC-8		FC-8			
N.A.	504.3	N.A.	443.9	12.0	453.3	-2.1

Figure 3-9

TEST DESCRIPTION

TEST NO.: 8 TEST NAME: GRE_3ELEM SOURCE: IBM

FUNCTIONAL GROUP: GNC

FUNCTIONAL DESCRIPTION: Performs the 3 element processing as determined by the FDIR status.

PROGRAMMING CHARACTERISTICS: IF-THEN, DO CASE
Single bit selection, integer arrays.

SIZE PERFORMANCE DATA:

(1)	(2)	(3)	(4)	(5)	(6)	(7)
Original HAL/S Code Size (HW)		% Improvement $\frac{(1)-(2)}{(1)}$	Improved HAL/S Code Size (HW)	% Improvement $\frac{(2)-(4)}{(2)}$	Independent Assembler Language Size (HW)	% HAL/S Inefficiency $\frac{(4)-(6)}{(6)}$
FC-5	FC-8		FC-8			
220	194	11.8	138	28.9	100	38.0

SPEED PERFORMANCE DATA:

(1)	(2)	(3)	(4)	(5)	(6)	(7)
Original HAL/S Code (µsec)		% Improvement $\frac{(1)-(2)}{(1)}$	Improved HAL/S Code (µsec)	% Improvement $\frac{(2)-(4)}{(2)}$	Independent Assembler Language (µsec)	% HAL/S Inefficiency $\frac{(4)-(6)}{(6)}$
FC-5	FC-8		FC-8			
N.A.	578.7	N.A.	592.5	-2.4	418.5	41.6
	1130.9		836.3	24.2	772.3	8.3

ORIGINAL PAGE IS
OF POOR QUALITY

Figure 3-10

TEST DESCRIPTION

TEST NO.: 9 TEST NAME: GGJ_AL_FDCMD SOURCE: IBM

FUNCTIONAL GROUP: GNC

FUNCTIONAL DESCRIPTION: Performs the functions of the A/L Flight Director Command Processor for roll and pitch.

PROGRAMMING CHARACTERISTICS: Scalar arithmetic, IF-THEN-ELSE, user-functions.

SIZE PERFORMANCE DATA:

(1)	(2)	(3)	(4)	(5)	(6)	(7)
Original HAL/S Code Size (HW)		% Improvement $\frac{(1)-(2)}{(1)}$	Improved HAL/S Code Size (HW)	% Improvement $\frac{(2)-(4)}{(2)}$	Independent Assembler Language Size (HW)	% HAL/S Inefficiency $\frac{(4)-(6)}{(6)}$
FC-5	FC-8		FC-8			
140	134	4.3	104	22.4	84	23.8

SPEED PERFORMANCE DATA:

(1)	(2)	(3)	(4)	(5)	(6)	(7)
Original HAL/S Code (µsec)		% Improvement $\frac{(1)-(2)}{(1)}$	Improved HAL/S Code (µsec)	% Improvement $\frac{(2)-(4)}{(2)}$	Independent Assembler Language (µsec)	% HAL/S Inefficiency $\frac{(4)-(6)}{(6)}$
FC-5	FC-8		FC-8			
N.A.	221.9	N.A.	146.3	34.1	131.9	10.9
	1169.7		654.2	44.1	647.8	1.0

Figure 3-11

TEST DESCRIPTION

TEST NO.: 10 TEST NAME: DMC_DISPLAY SOURCE: IBM

FUNCTIONAL GROUP: UI

FUNCTIONAL DESCRIPTION: Determines the starting location of a Display Format Table within one of four Display Format Buffers and if not found in any of the four buffers then cause a Mass Memory read to load the proper display into a buffer.

PROGRAMMING CHARACTERISTICS: 6 Compools, IF-THEN-ELSE, Integer arith., bits with subscripting structures.

SIZE PERFORMANCE DATA:

(1)	(2)	(3)	(4)	(5)	(6)	(7)
Original HAL/S Code Size (HW)		% Improvement $\frac{(1)-(2)}{(1)}$	Improved HAL/S Code Size (HW)	% Improvement $\frac{(2)-(4)}{(2)}$	Independent Assembler Language Size (HW)	% HAL/S Inefficiency $\frac{(4)-(6)}{(6)}$
FC-5	FC-8		FC-8			
380	365	3.9	166	54.5	214	-22.4

SPEED PERFORMANCE DATA:

(1)	(2)	(3)	(4)	(5)	(6)	(7)
Original HAL/S Code (μ sec)		% Improvement $\frac{(1)-(2)}{(1)}$	Improved HAL/S Code (μ sec)	% Improvement $\frac{(2)-(4)}{(2)}$	Independent Assembler Language (μ sec)	% HAL/S Inefficiency $\frac{(4)-(6)}{(6)}$
FC-5	FC-8		FC-8			
N.A.	377.9	N.A.	303.5	19.7	320.9	-5.4

ORIGINAL PAGE IS OF POOR QUALITY

Figure 3-12

TEST DESCRIPTION

TEST NO.: 11 TEST NAME: DMC_NEW_DISPLAY SOURCE: IBM

FUNCTIONAL GROUP: UI

FUNCTIONAL DESCRIPTION: New Display Processing is to perform UI control data maintenance and logic necessary to control output of background FCW's to the DEU's.

PROGRAMMING CHARACTERISTICS: Same as Test #10

SIZE PERFORMANCE DATA:

(1)	(2)	(3)	(4)	(5)	(6)	(7)
Original HAL/S Code Size (HW)		% Improvement $\frac{(1)-(2)}{(1)}$	Improved HAL/S Code Size (HW)	% Improvement $\frac{(2)-(4)}{(2)}$	Independent Assembler Language Size (HW)	% HAL/S Inefficiency $\frac{(4)-(6)}{(6)}$
FC-5	FC-8		FC-8			
271	206	24.0	199	3.4	138	44.2

SPEED PERFORMANCE DATA:

(1)	(2)	(3)	(4)	(5)	(6)	(7)
Original HAL/S Code (µsec)		% Improvement $\frac{(1)-(2)}{(1)}$	Improved HAL/S Code (µsec)	% Improvement $\frac{(2)-(4)}{(2)}$	Independent Assembler Language (µsec)	% HAL/S Inefficiency $\frac{(4)-(6)}{(6)}$
FC-5	FC-8		FC-8			
N.A.	114.3	N.A.	111.1	2.3	101.3	9.7

Figure 3-13

TEST DESCRIPTION

TEST NO.: 12 TEST NAME: DMC FILL BACK- GROUND_FCWS SOURCE: IBM

FUNCTIONAL GROUP: UI

FUNCTIONAL DESCRIPTION: Locate the background FCW's and issue the I/O SVC to send FCW's to the DEU.

PROGRAMMING CHARACTERISTICS: See test #10.

SIZE PERFORMANCE DATA:

(1)	(2)	(3)	(4)	(5)	(6)	(7)
Original HAL/S Code Size (HW)		% Improvement $\frac{(1)-(2)}{(1)}$	Improved HAL/S Code Size (HW)	% Improvement $\frac{(2)-(4)}{(2)}$	Independent Assembler Language Size (HW)	% HAL/S Inefficiency $\frac{(4)-(6)}{(6)}$
FC-5	FC-8		FC-8			
723	689	4.7	256	62.8	292	-12.3

SPEED PERFORMANCE DATA:

(1)	(2)	(3)	(4)	(5)	(6)	(7)
Original HAL/S Code (usec)		% Improvement $\frac{(1)-(2)}{(1)}$	Improved HAL/S Code (usec)	% Improvement $\frac{(2)-(4)}{(2)}$	Independent Assembler Language (usec)	% HAL/S Inefficiency $\frac{(4)-(6)}{(6)}$
FC-5	FC-8		FC-8			
N.A.	907.7	N.A.	884.7	2.5	730.8	21.1

Figure 3-14

TEST DESCRIPTION

TEST NO.: 13 TEST NAME: SAS_ANALOG_SCALE SOURCE: IBMFUNCTIONAL GROUP: SMFUNCTIONAL DESCRIPTION: Performs the logic necessary to convert parameter values from PCM units to engineering units or from engineering units back to PCM units.PROGRAMMING CHARACTERISTICS: 5 Compoos, IF-THEN-ELSE, Array subscripting, Booleans, Structures.

SIZE PERFORMANCE DATA:

(1)	(2)	(3)	(4)	(5)	(6)	(7)
Original HAL/S Code Size (HW)		% Improvement $\frac{(1)-(2)}{(1)}$	Improved HAL/S Code Size (HW)	% Improvement $\frac{(2)-(4)}{(2)}$	Independent Assembler Language Size (HW)	% HAL/S Inefficiency $\frac{(4)-(6)}{(6)}$
FC-5	FC-8		FC-8			
274	212	22.6	233	-12.3	208	14.4

SPEED PERFORMANCE DATA:

(1)	(2)	(3)	(4)	(5)	(6)	(7)
Original HAL/S Code (µsec)		% Improvement $\frac{(1)-(2)}{(1)}$	Improved HAL/S Code (µsec)	% Improvement $\frac{(2)-(4)}{(2)}$	Independent Assembler Language (µsec)	% HAL/S Inefficiency $\frac{(4)-(6)}{(6)}$
FC-5	FC-8		FC-8			
N.A.	111.3	N.A.	111.3	0.0	112.8	-1.3
	268.7		268.7	0.0	206.8	29.9

Figure 3-15

TEST DESCRIPTION

TEST NO.: 14 TEST NAME: SAS_POLYSOL SOURCE: IBM

FUNCTIONAL GROUP: SM

FUNCTIONAL DESCRIPTION: Performs the various order polynomial backwards solutions as determined by SAS_ANALOG_SCALE.

PROGRAMMING CHARACTERISTICS: Integers, bit strings, array subscripts, Structures.

SIZE PERFORMANCE DATA:

(1)	(2)	(3)	(4)	(5)	(6)	(7)
Original HAL/S Code Size (HW)		% Improvement $\frac{(1)-(2)}{(1)}$	Improved HAL/S Code Size (HW)	% Improvement $\frac{(2)-(4)}{(2)}$	Independent Assembler Language Size (HW)	% HAL/S Inefficiency $\frac{(4)-(6)}{(6)}$
FC-5	FC-8		FC-8			
113	110	2.7	87	20.9	66	31.8

SPEED PERFORMANCE DATA:

(1)	(2)	(3)	(4)	(5)	(6)	(7)
Original HAL/S Code (µsec)		% Improvement $\frac{(1)-(2)}{(1)}$	Improved HAL/S Code (µsec)	% Improvement $\frac{(2)-(4)}{(2)}$	Independent Assembler Language (µsec)	% HAL/S Inefficiency $\frac{(4)-(6)}{(6)}$
FC-5	FC-8		FC-8			
N.A.	921.6	N.A.	868.2	5.8	594.3	46.1

ORIGINAL PAGE IS
OF POOR QUALITY.

GROUP	TEST #	HAL SIZE	ASSEMBLY SIZE	%	HAL TIME (No-lib)	ASSEMBLY TIME (No-lib)	%	(No-lib)
C N G	1**	77	64	20.3	234.5 (234.5)	213.5 (213.5)	9.8	(9.8)
	2	316	268	17.9	345.3 (345.3)	326.7 (326.7)	5.7	(5.7)
	3	308	256	20.3	15507.6 (1433.1)	14674.4 (1262.5)	5.7	(13.5)
	4	93	132	-29.5	1817.3 (1722.7)	1857.9 (1857.9)	-2.2	(-7.3)
	5	166	138	20.3	976.3 (931.7)	874.1 (874.1)	11.7	(6.6)
	6**	108	98	10.2	872.2 (92.7)	735.4 (93.5)	18.6	(-0.9)
	7**	165	137	20.4	446.0 (285.7)	422.6 (262.3)	5.5	(8.9)
	8**	138	100	38.0	442.2 (281.9)	418.4 (258.1)	5.7	(9.2)
	9	104	84	23.8	70.1 (70.1)	68.3 (68.3)	2.6	(2.6)
	Sub-total	1475	1277	15.5*	116.3 (116.3)	113.9 (113.9)	2.1	(2.1)
U H	10**	166	214	-22.4	443.9 (443.9)	453.3* (453.3)*	-2.1	(-2.1)
	11**	199	138	44.2	592.5 (592.5)	418.5 (418.5)	41.6	(41.6)
	12**	256	292	-12.3	836.3 (836.3)	772.3 (772.3)	8.3	(8.3)
	Sub-total	621	644	- 3.6	654.2 (221.9)	647.8 (215.5)	1.0	(3.0)
S M	13**	238	208	14.4	146.3 (146.3)	131.9 (131.9)	10.9	(10.9)
	14**	87	66	31.8	23501.0 (7754.9)	22129.0 (7322.3)	6.2*	(5.9)*
	Sub-total	325	274	18.6*	303.5 (303.5)	320.9 (320.9)	-5.4	(-5.4)
	TOTAL	2421	2195	10.3	111.1 (111.1)	101.3 (101.3)	9.7	(9.7)

* Aggregate & Improvement

** Routines using no HAL/S libraries at all.

Figure 3-16 HAL/S-FC ACCEPTANCE TEST RAW DATA

("Best" HAL/S vs. "Best" Assembler Language)

Figure 3-17. Size and Speed of Original and Improved HAL/S Source using FC-8

GROUP	TEST #	SIZE			TIME		
		VERSION		% IMPROVED	VERSION		% IMPROVED
		ORIGINAL	IMPROVED		ORIGINAL	IMPROVED	
G N C	1	91	77	15.4 %	249.7 356.3	234.5 345.3	6.1 % 3.1 %
	2	450	316	29.8 %	16327.8	15507.6	5.0 %
	3	606	308	49.2 %	4432.9 2161.7	1817.3 976.3	58.1 % 54.8 %
	4	100	93	7.0 %	1050.2	872.2	16.9 %
	5	181	166	8.3 %	459.6 448.8	446.0 442.2	3.0 % 1.5 %
	6	114	108	5.3 %	70.1 122.1	70.1 116.3	0.0 % 4.8 %
	7	267	165	38.2 %	504.3	443.9	12.0 %
	8	194	138	28.9 %	578.7 1103.9	592.5 836.3	-2.4 % 24.2 %
	9	134	104	22.4 %	1169.7 221.9	654.2 146.3	44.1 % 34.1 %
	Sub-Total	2137	1475	31.0 %*	29157.7	23501.0	19.4 %*
U I	10	365	145	60.3 %	377.9	166.9	55.8 %
	11	206	199	3.4 %	114.3	111.1	2.8 %
	12	689	248	64.0 %	907.7	802.5	11.6 %
		Sub-Total	1260	592	53.0 %*	1399.9	1080.5
S M	13	238	238	0.0 %	268.7 111.3	268.7 111.3	0.0 % 0.0 %
	14	110	87	20.9 %	921.6	858.2	5.8 %
		Sub-Total	348	325	6.6 %*	1301.6	1248.2
		3745	2392	36.1 %*	31859.2	25829.7	18.9 %*

* Aggregate %-improvement

ORIGINAL PAGE IS
OF POOR QUALITY

4.0 DATA ANALYSIS

In order to arrive at a performance characteristic for the HAL/S-FC compiler a number of candidate measures were considered. Fundamentally, each measure involved the averaging of the collected data by a particular algorithm. The basic approaches were as follows:

- The performance can be based on the set of routines taken as an aggregate.

In this approach all the routines are viewed as members of a single program. For example, the program size is the sum of the sizes of all the routines. Size comparison is then based on total HAL/S size vs. total assembler language size. The effect of this measure is to give more weight to the larger and more time consuming routines.

- The performance can be defined as the average performance of the routines taken individually.

In this approach, the % inefficiency of each routine is first computed and then all "%-values" are averaged to arrive at compiler performance. The effect of this measure is to equalize the weighting of each routine.

- The performance of the HAL/S-FC compiler can be based on the expected applicability of HAL/S to Shuttle application programming.

In this approach, the routines are grouped into the applications categories designated by IBM/Houston: guidance, navigation and control (GNC), systems management (SM), user interface (UI). The performance of each group is first assessed (by either of the two methods above) and the groups are combined using appropriate weighting factors. The effect here is to tie more closely the compiler performance to the intended applications. However, the performance values become a strong function of the selected applications weighting factors.

- The speed performance can be evaluated with and without libraries.

The intention here is to gain specific insight into the code generation facility of the compiler.

- Minor variations of all of the above can be postulated.

After giving careful consideration to these approaches it became apparent that the "correct one" was a matter of subjective judgement. In order to enhance objectivity and promote the credibility of the results, it was decided not to confine the analysis to a single approach, but instead to try them all.

PRECEDING PAGE BLANK NOT FILMED

4.1 Measure Definitions (Formulae), in %-Inefficiency

Let H_i = size/speed of each HAL/S routine;

A_i = size/speed of corresponding assembler language routine

Measure T: An Aggregate of all code

$$T = \left(\frac{\sum_1^N H_i}{\sum_1^N A_i} - 1 \right) \times 100, \text{ IN \%}$$

Measure A: An Average of %'s

$$A = \frac{1}{N} \sum_1^N \left(\frac{H_i}{A_i} - 1 \right) \times 100$$

Measure WT*: Aggregate by Group then Weighted

$$WT = W_{GNC} \left(\frac{\sum_j H_j}{\sum_j A_j} - 1 \right) \times 100 + W_{SM} \left(\frac{\sum_k H_k}{\sum_k A_k} - 1 \right) \times 100 + W_{UI} \left(\frac{\sum_l H_l}{\sum_l A_l} - 1 \right) \times 100$$

Measure WA*: An Average of %'s by Group then Weighted

$$WA = W_{GNC} \frac{1}{N_{GNC}} \sum \left(\frac{H_j}{A_j} - 1 \right) \times 100 + W_{SM} \frac{1}{N_{SM}} \sum \left(\frac{H_k}{A_k} - 1 \right) \times 100 + W_{UI} \frac{1}{N_{UI}} \sum \left(\frac{H_l}{A_l} - 1 \right) \times 100$$

* W_{GNC} , W_{SM} , W_{UI} are size/speed weighting factors.

4.2 Performance Weighting Factors Based on HAL/S-FC Shuttle Applicability

The weighting factors reflect the relative dominance of GNC, SM and UI routines in the Shuttle applications programming. The numerical values shown below were computed based on the best available estimates of relative sizes and CPU utilizations at the time of CI.

SIZE FACTORS

(EXCLUDING DATA AREAS)

TYPE	SOURCE	SIZE	WEIGHT
GNC	ALT MATED DROP		
	IBM SIZING & TIMING ESTIMATES (6/17/75)	18,559	0.672
	IBM FDS's (2/17/75)		
UI	LESS BUFFERS AND COMPOOL	6,912	0.250
SM	LESS TABLES	2,133	0.077
		27,604	1.000

SPEED FACTORS

TYPE	SOURCE	CPU %	WEIGHT
GNC	IBM SIZING & TIMING ESTIMATES (6/16/75)	117.3	0.732
UI		30.5	0.190
SM		12.3	0.077
		160.1	1.000

4.3 HAL/S-FC Size Performance

The size results for the 14 test routines summarized in Figure 3-16 were analyzed using the four measures of Sec. 4.1. The compiler size performance, expressed as % inefficiency over assembler language, is shown in Figure 4-1.

Measure	HAL/S Inefficiency
1) T	10.3%
2) A	14.1%
3) WT	10.9%
4) WA	13.2%

Figure 4-1. HAL/S-FC Size Performance

Measure	HAL/S Inefficiency
1) T	7.7%
2) A	10.7%
3) WT	9.8%
4) WA	8.6%

Figure 4-2. HAL/S Speed Performance
(End-to-End)

4.4 HAL/S-FC Speed Performance

4.4.1 End-to-End

As listed in Figure 3-16, the 14 test routines contributed 21 execution time results. This occurred because more than one executive path was included for several of the routines. The performance was then calculated using the measures of Section 4.1, and is shown in Figure 4-2.

4.4.2 Alternate Speed Criteria

The issue of "end-to-end" vs. "no libraries" performance was discussed in Section 2.2.3.1 of this report. The "no-lib" data has been included in Figure 3-16 and was subjected to the four standard measures. Figure 4-3 summarizes the groundrules for this approach and the resulting performance figures are shown in Figure 4-4, superimposed on the end-to-end data. Note that measures 1 through 4 are end-to-end, and measures 5-8 are "no-lib".

Additional speed measures were created by the observation that the end-to-end aggregate timing data was subject to significant influence by the test routine #2. Taken as an aggregate this single routine consumed approximately 60% of the total execution time. Therefore in Figure 4-3, measures 9, 10 represent compiler speed performance excluding the results of routine #2. (Only the aggregate speed measures apply.)

4.4.3 Another Look at No-Libraries

Because of the inherent difficulties in extracting the influence of libraries on the HAL/S routines, the consensus at CI was to examine the effect of restricting performance evaluation to the set of HAL/S routines without any libraries at all. (These routines are marked with ** in Figure 3-16.) Using this reduced subset, and the standard measures, the computed compiler performance is shown in Figure 4-5.

AN ALTERNATE SPEED CRITERION

THUS FAR, PERFORMANCE MEASURES HAVE BEEN BASED ON END-TO-END DATA.

HOWEVER, CODE GENERATOR EFFICIENCY LESS
LIBRARIES MAY ALSO BE OF INTEREST.

THE ANALYSIS IS AS FOLLOWS:

FOR HAL/S ROUTINES

DISCOUNT TIME FOR:

- (1) LIBRARIES
- (2) LIBRARY SET-UPS

FOR A/L ROUTINES

DISCOUNT TIME FOR:

- (1) LIBRARIES
- (2) LIBRARY SET-UPS
- (3) FUNCTIONAL LIBRARIES AND SET-UPS
(WHERE IN-LINE CODE HAS BEEN
SUBSTITUTED FOR HAL/S LIBRARIES)

Figure 4-3

Measure	HAL/S Inefficiency
1) T	7.7%
2) A	10.7%
3) WT	9.8%
4) WA	8.6%
5) T	9.7%
6) A	10.1%
7) WT	9.5%
8) WA	9.0%
9) T	10.7%
10) A	10.5%

Figure 4-4. HAL/S-FC Speed Performance
(All Cases)

Measure	HAL/S Inefficiency
1) T	17.0%
2) A	12.9%
3) WT	13.6%
4) WA	10.6%

Figure 4-5. HAL/S-FC Speed Performance
(Cases with NO Libraries Only)

4.5 Effect of Compiler Optimization

The HAL/S-FC compiler has been undergoing an optimization program in order to improve performance. The acceptance test activities permitted at least a tentative assessment of its progress. The data recorded in columns (1), (2) and (3) of each test template (Figures 3-2 to 3-15) indicate the improvement of the modestly optimized FC-8 over FC-5. The results for all routines were analyzed using the standard measures of Section 4.1; a summary is shown below.

MEASURE	DESCRIPTION	SIZE REDUCTION FROM FC-5 TO FC-8
T	STRAIGHT AGGREGATE	10.23%
A	AVERAGE OF %'S	9.51%
WT	WEIGHTED GROUP AGGREGATES	10.28%
WA	WEIGHTED AVERAGE OF %'S	9.03%

4.6 Effect of Programmer Experience

The test routines comprising the HAL/S-FC acceptance set were selected from existing procedures, coded primarily by IBM/Houston and Intermetrics, with no particular attention paid to demonstrated performance. As discussed in Section 2.1, numerous iterations of both HAL/S and assembler language code followed before the "best" results were achieved. The difference between original and final speed and size, using compiler FC-8, may be attributed to a programmer experience factor. Columns (2), (4) and (5) of the templates record this data base and a summary of improvements for each defined measure is shown below.

MEASURE	DESCRIPTION	SIZE REDUCTION USING FC-8	SPEED REDUCTION USING FC-8
T	STRAIGHT AGGREGATE	36.10%	18.9%
A	AVERAGE OF %'s	25.20%	16.3%
WT	WEIGHTED GROUP AGGREGAT :	36.90%	18.8%
WA	WEIGHTED AVERAGE OF %'s	26.10%	17.6%

5.0 CONCLUSIONS AND OBSERVATIONS

The L/S Acceptance Test exercise was characterized by the sincere efforts of all parties to derive compiler performance through controlled procedures and objective criteria. The approach was highly successful. The resulting performance data was scrutinized thoroughly, discussed and finally accepted by consensus, at the Configuration Inspection (CI) meeting.

The compiler performance figures as well as some observations on method and directions are summarized below.

5.1 Summary of HAL/S Compiler Performance

Based on the CI acceptance testing the following figures may be stated for HAL/S-FC compiler performance:

HAL/S-FC Size Inefficiency:	approximately 11-13% over Assembler language
HAL/S-FC Speed Inefficiency:	approximately 9-11% over Assembler language

These specific ranges of values were selected as essentially spanning the significant performance figures as derived via the several measures.

The final values should be qualified as follows: only a sample of Shuttle routines were utilized in the testing, the Shuttle category weighting factors were based on preliminary design data, the programming was accomplished by experienced individuals (at Intermetrics and IBM/OWEGO) participating in a competition -- not in the line production of flight code.

As a by-product of the testing, data was collected on the secondary objectives of the exercise: compiler optimization and programmer experience. The following characteristics were noted:

- FC-8 size optimization
Improvement over FC-5 ~10%
- Influence of programmer experience
Reduced* the size of HAL/S routines by ~25%
- Influence of programmer experience
Reduced* the execution time of HAL/S
routines by ~17%

* Using FC-8

5.1.1 Additional Test Routine Results and Comments

At the time of the CI meeting and for a few weeks thereafter IBM/Houston saw opportunities for further improvement of the assembler language results for the User Interface (UI) subset, viz. Tests Nos. 10, 11, 12. In general by utilizing global knowledge of the computations, significant improvements were gained primarily through better register-savings policies. The specific results* for these three cases were reported as:

Test #	Assembly Size	Assembly Time
10	130	143.3
11	132	98.7
12	236	695.5

As a checkpoint comparison, Intermetrics re-worked Test Routine No. 10 using some of the same design approaches suggested by the IBM/Houston assembler language routines. The new results for this routine became:

Test #	HAL Size	HAL Time
10	144	161.5

Time and available resources did not permit a full-blown post-CI activity; however there is no doubt that further improvements could have been achieved on both sides, through refined design techniques, the taking advantage of special circumstances and the application of "trickier" code.

* Also see Figure 3-16.

The performance values stated at the beginning of Section 5.1 are not precise. They can only reflect the sample chosen and the effort spent. Perhaps an additional tolerance of 2-3% should be applied to account for further refinements. New data on these routines should not, however, significantly affect the qualitative statement of HAL/S performance.

5.1.2 Comments on Programmer Experience

The significant improvements attributable to programmer experience signify that the HAL/S compiler cannot improve HAL/S source code on its own, but simply translates an original design into machine code. The improvement comes from two sources: (1) source level modifications that are implementation independent, (2) knowledge of alternatives which take advantage of specific code generator/machine characteristics. Both approaches found application in modifying the test routines.

In many ways (1) reflects good programming design, exploitation of logical structure, elimination of redundancy, minimization of decision points, and full expression of the language. (2), to some extent, is required because of the mismatch between the instruction set of the machine and the particular higher order language (HOL) at hand. Unless the AP-101 were designed as a HAL/S machine, i.e. executing HAL/S statements (or their equivalent) in microcode, the opportunity for varying efficiency from HAL/S statements to machine code would always exist. Specific knowledge of how a particular HAL/S construct is implemented can then become important when high performance is to be achieved.

The HAL/S compiler design effort is attempting to reduce this gap through code optimization, i.e. by substituting compiler capabilities (and complexity) for required programmer knowledge. Perhaps in future programs the higher order language environment (i.e. the machine) will be more amenable, resulting in greater standard efficiencies at less software impact.

5.2 Some Observations on Test Methods and Criteria

The HAL/S acceptance test activity was carefully planned and controlled in order to insure the credibility of the results. Every effort was made to conduct the tests with a maximum of openness and communication. A number of important elements contributed to its success:

1) selection of independent programming teams for HAL/S and assembler language

2) establishment of comprehensive mutually acceptable groundrules

3) definition of representative benchmarks and common data bases

4) requirement for successful execution, and full interchangeability, of HAL/S and assembler language routines

5) the sharing of design approaches between teams and the maintenance of a friendly spirit of competition

6) the reporting of all data without prejudice.

In spite of the above, "good" and "better" results for HAL/S and/or assembler language were achieved as a result of differing emphasis. Figure 5-1 illustrates the point by plotting the tradeoff between size and speed for Test Routine No. 3. IBM/OWEGO selected a smaller, slower solution than Intermetrics. Figure 5-2 shows the march toward size improvement by successful revisions of HAL/S source. The requirement for "size" instead of "speed" or vice versa, was not clearly stated in the acceptance test groundrules and, for the most part, each team tried for both. Occasionally the results became quite sensitive to the particular emphasis. In retrospect, some criterion should have been established accompanied by a weighting formula, so that the size/speed tradeoffs would be more explicit and better appreciated.

An interesting by-product in the struggle and competition to achieve better performance was the obvious benefits of a compiler in terms of increased productivity. Many new designs and source modifications were attempted by the HAL/S team because the compiler automatically attended to the details of addressing and instruction selection, and provided comprehensive diagnostics. The assembler language group was more cautious and less inclined to change a working routine because of the potential for programming error. The result was more tries* at the problem for HAL/S and a better appreciation for design tradeoffs. When dealing with assembler language the prospect of re-doing a substantial program became overwhelming.

* In Figure 3-16 some of the HAL/S results were, in fact, "better" than the corresponding assembler language. In these cases the assembler language team had stopped at some point, presuming that their routines had reached satisfactory performance. HAL/S work continued, sometimes achieving surprising results.

TIME/SPACE TRADE-OFF

(TEST No. 3 G_FILTER)

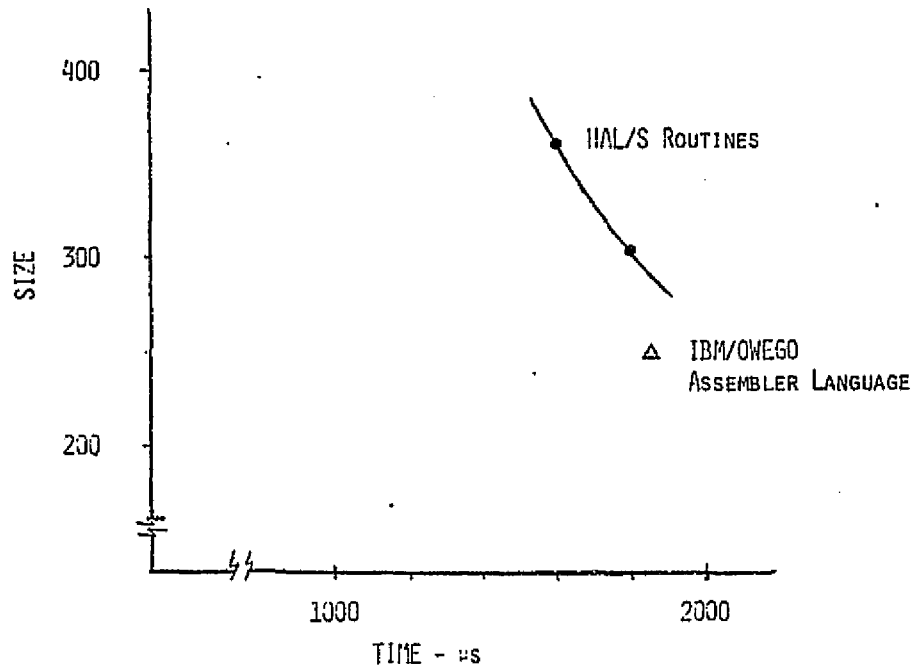


Figure 5-1

IMPROVEMENT CURVE

(TEST No. 3 G_FILTER)

SIZE vs REVISION

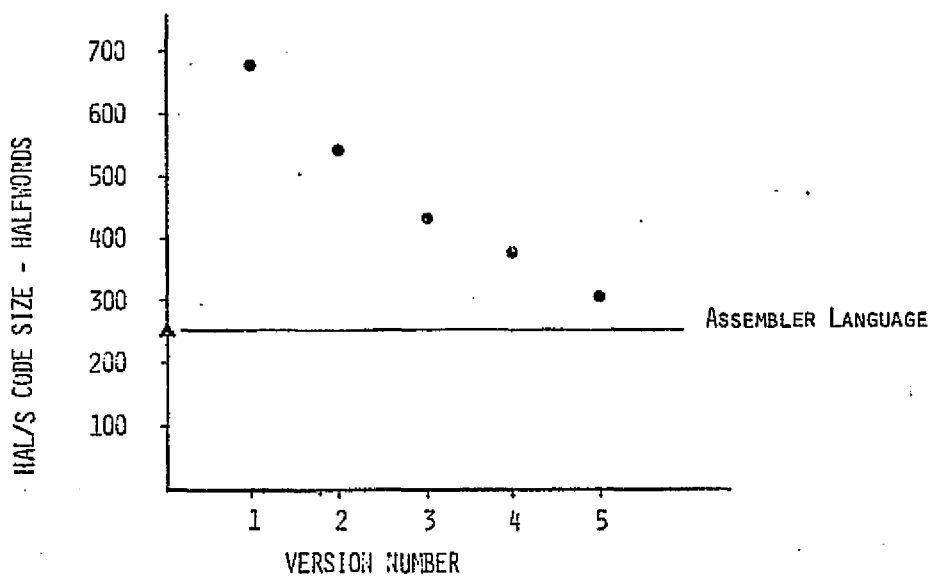


Figure 5-2

5.3 Areas Identified for HAL/S Improvement

As a result of the test exercise and with the benefit of discussions among HAL/S and assembler language team members, a number of areas were identified for improvement in HAL/S code generator and compiler design. Most will contribute to time efficiency by substituting in-line code for the current library calls. The features to be improved are:

- STRUCTURE MOVES
- VECTOR/MATRIX MOVES
- ZEROING OF VECTOR/MATRICES
- ⊗ MAINTAIN VECTOR OF SIZE 3 IN A VECTOR ACCUMULATOR AND PERFORM SIMPLE COMPUTATIONS IN-LINE
- VECTOR (SIZE N) AND MATRIX ADDITION AND SUBTRACTION DONE IN-LINE
- LOOP STREAMLINING
- COMMON SUBEXPRESSION

All items have become part of the HAL/S optimization program.

References

1. HAL/S-360 Compiler System Functional Specification, Intermetrics, Inc. PDR2 #IM004, July 13, 1973.
2. HAL/S-FC Compiler System Functional Specification, Intermetrics, Inc., IR-59-4, July 24, 1974.

APPENDIX

Results Associated with Test Routine #8 GRE_3ELEM

CONTENTS

- A.1: Listing of Original HAL/S Source
- A.2: Listing of "Best" HAL/S Source
- A.3: Listing of "Best" Assembler Language Code
- A.4: Execution Trace (from simulator)
(HAL/S statement numbers correspond to listing in
Appendix A.2.)
- A.5: Data Bases, Before and After Test

PRECEDING PAGE BLANK NOT FILMED

STMT	SOURCE	CURRENT SCOPE
112 E M S	BIT4 = GREB_FAULTPAIRS 3 AT BITPTR ;	GRE_3ELEM
113 E M	GREV_I_FAULTPAIRS = INTEGER(BIT4);	GRE_3ELEM
114 M	DO CASE GREV_I_FAULTPAIRS - 2;	GRE_3ELEM
115 M	ELSE	GRE_3ELEM
115 M	;	GRE_3ELEM
116 M	DO;	GRE_3ELEM CASE 1
117 E M S	GREB_FDIPSTATUS = OFF; BITPTR+2	GRE_3ELEM
118 E M S	GREB_SFFAULT = OFF; 3	GRE_3ELEM
119 M S	GREV_FAILCNTR , GREV_FAILCNTR = 0; 2 3	GRE_3ELEM
120 M	END;	GRE_3ELEM
121 M	;	GRE_3ELEM CASE 2
122 M	DO;	GRE_3ELEM CASE 3
123 E M S	GREB_FDIPSTATUS = OFF; BITPTR	GRE_3ELEM
124 E M S	GREB_SFFAULT = OFF; 1	GRE_3ELEM
125 M S	GREV_FAILCNTR , GREV_FAILCNTR = 0; 1 3	GRE_3ELEM
126 M	END;	GRE_3ELEM
127 M	DO;	GRE_3ELEM CASE 4
128 E M S	GREB_FDIPSTATUS = OFF; BITPTR+1	GRE_3ELEM
129 E M S	GREB_SFFAULT = OFF; 2	GRE_3ELEM

STMT	SOURCE	CURRENT SCOPE
130 M S	GREV_FAILCNTR , GREV_FAILCNTR = 0; 1 2	GPF_3RLPM
131 M	END;	GPF_3TLEM
132 M	END;	GPF_3PLEM DO CASE END
133 M	END;	GPF_3SELM
134 M	CLCSE;	GPF_3ELPM

**** B L O C K S U M M A R Y ****

OUTER VARIABLES USED
 GREV_DATA, GREV_J*, GREV_J, GREV_TOLERANCE, GREV_FAILCNTR*, GREV_FAILCNTR, WORD*, BITPTR, GREV_FAILCNT, WORD, GREB_FAULTPAIRS*
 BIT4*, GPF_FAULTPAIRS, BIT4, GREB_FDIRSTATUS*, GREB_SFFAULT*

ORIGINAL PAGE IS
OF POOR QUALITY

SIXT	SOURCE	CURRENT SCOPE
62 M	GRE_3ELEM:	GRE_3ELEM
62 M	PROCEDURE:	GRE_3ELEM
63 M	DECLARE GREV_INDEX ARRAY(4) INTEGER INITIAL(1, 2, 3, 1);	GRE_3ELEM
64 M	DECLARE PICK ARRAY(8) INTEGER INITIAL(10, 10, 2, 10, 0, 1, 10, 10);	GRE_3ELEM
65 M	EC FOR GREV_J = 1 TO 3;	GRE_3ELEM
66 M	IF ABS(GREV_DATA	GRE_3ELEM
SI	GREV_INDEX - GREV_DATA	
SI	GREV_INDEX GREV_INDEX	
	GREV_J GREV_J+1	
67 M	GREV_FAILCNT = GREV_FAILCNT + 1;	GRE_3ELEM
SI	GREV_J GREV_J	
68 M	ELSE	GRE_3ELEM
68 M	GREV_FAILCNT = 0;	GRE_3ELEM
SI	GREV_J	
69 M	END;	GRE_3ELEM
70 M	IF [GREV_FAILCNT] /= 0 THEN	GRE_3ELEM
71 M	DO;	GRE_3ELEM
72 M	TEMPORARY ACC INTEGER;	GRE_3ELEM
73 M	ACC = 0;	GRE_3ELEM
74 M	IF GREV_FAILCNT ₁ >= GREV_FAILCNT THEN	GRE_3ELEM
SI		
75 M	ACC = 4;	GRE_3ELEM
76 M	IF GREV_FAILCNT ₂ >= GREV_FAILCNT THEN	GRE_3ELEM
SI		
77 M	ACC = ACC + 2;	GRE_3ELEM
78 M	IF GREV_FAILCNT ₃ >= GREV_FAILCNT THEN	GRE_3ELEM
SI		
79 M	ACC = ACC + 1;	GRE_3ELEM
80 M	GREV_FAILPAIRS = SUBBIT(ACC);	GRE_3ELEM
SI	3 AT BITPTR	
81 M	DO CASE ACC - 2;	GRE_3ELEM
82 M	ELSE	GRE_3ELEM

Appendix A.2

"Best" HAL/S Source

STMT	SOURCE	CURRENT SCOPE
82 M	;	GRE_3FLEM
83 M	DO;	GRE_3FLEM CASE 1
E		
84 M	GREB_SFPAULT = OFF;	GRE_3ELEM
S	3	
85 M	GREV_FAILCNTR , GREV_FAILCNTR = 0;	GPF_3ELEM
S	2 3	
86 M	END;	GPF_3ELEM
87 M	;	GRE_3ELEM CASE 2
88 M	DC;	GPF_3ELEM CASE 3
E		
89 M	GREB_SFPAULT = OFF;	GRE_3ELEM
S	1	
90 M	GREV_FAILCNTR , GREV_FAILCNTR = 0;	GPF_3ELEM
S	1 3	
91 M	END;	GPF_3ELEM
92 M	DO;	GRE_3ELEM CASE 4
E		
93 M	GREB_SFPAULT = OFF;	GPF_3FLEM
S	2	
94 M	GREV_FAILCNTR , GREV_FAILCNTR = 0;	GPF_3ELEM
S	1 2	
95 M	END;	GRE_3ELEM
96 M	END;	GPF_3FLEM DO CASE END
E		
97 M	GREB_FDIRSTATUS = OFF;	GPF_3FLEM
S	BITPTR+PICK	
S	ACC	
98 M	END;	GPF_3FLEM
99 M	CLCSF;	GPF_3FLEM

ORIGINAL PAGE IS
OF POOR QUALITY

**** B L C C K S U M M A R Y ****

GLOBAL VARIABLES USED

GREV_J*, GREV_J, GREV_DATA, GREV_TOLERANCE, GREV_FAILCNTR*, GREV_FAILCNTR, GREV_FAILCNT, BITPTR, GREB_PAULTPAIRS*, GREB_SFPAULT*, GREB_FDIRSTATUS*

LCC OBJECT CCDE ADR1 ADR2 SOURCE STATEMENT

GPC VER9.1 14.52 07/01/75

Appendix A.3
 "Best" Assembler Language Code

CC000			3	ABGRESCH	AMAIN		0C003000
CC00000			4+	ABGRESCH	CSFCT		01-AMAIN
CC00001			5+	R0	EQU	0	01-AMAIN
CC00002			6+	R1	EQU	1	01-AMAIN
CC00003			7+	P2	EQU	2	01-AMAIN
CC00004			8+	R3	EQU	3	01-AMAIN
CC00005			9+	F4	EQU	4	01-AMAIN
CC00006			10+	R5	EQU	5	01-AMAIN
CC00007			11+	R6	EQU	6	01-AMAIN
CC00000			12+	R7	EQU	7	01-AMAIN
CC00001			13+	F0	EQU	0	01-AMAIN
CC00002			14+	F1	EQU	1	01-AMAIN
CC00003			15+	F2	EQU	2	01-AMAIN
CC00004			16+	F3	EQU	3	01-AMAIN
CC00005			17+	F4	EQU	4	01-AMAIN
CC00006			18+	F5	EQU	5	01-AMAIN
CC00007			19+	F6	EQU	6	01-AMAIN
CC00000			20+	F7	EQU	7	01-AMAIN
CC00000			21+		USING	STACK,3	01-AMAIN
CC00000	E8C0	CC00	22+		LA	0,0(0)	01-AMAIN
CC00001	C8FC	1003	23+		STM0	NEXTSTK	01-AMAIN
CC00003	980C	CC03	24+		LH	0,NEXTSTK	01-AMAIN
CC00006			26	ACC	EQU	6	0C005000
CC0000E			27	S5	EQU	61	0C006000
CC0003F			28	S7	EQU	63	0C007000
CC00000			30		USING	#DGRESCH,R1	0C009000
CC00000			31		USING	#LOCAL,2	0C010000

ADDRESS STACK AREA
 CLEAR LOWER HALF AS NULL LOCAL DATA PTR
 SAVE REGS AT CALL IN NEW STACK AREA
 UPDATE STACK PTR

LCC	OBJECT	CCDE	ADR1	ADR2
00004	9F16		CC05	
CC005	9FF5	E029		C029
00007	78F5	C032		C032
CC009	1DE7			
CC00A	8516		CC05	
CC00B	5FF5	A029		C029
CC00D	58F5	C032		C032
CC00E	ED04		C011	0001
00010	78E8			
00011	48F9	0016		CC16
00013	EE10		CC18	0004
00014	9EF5	E01A		CC1A
00016	8E16		CC05	
00017	DF04		CC19	0001
00018	CEE6			
00019	EEF5	E01A		CC1A
0001B	E716		CC05	
0001C	971A		CC06	
0001D	IA66		CC05	0019
0001F	9E6D		CC1B	
0001F	8E71		CC1C	
00020	8675		CC1D	
00021	CA77	C03E		CC61 003E
00023	9D2D		CC0B	
00024	9DF6	A00E		CC0B
00026	9C16		0005	
00027	F4F4			003D
00028	9E41		0010	
00029	9F16		CC05	
0002A	9E51		0014	
0002B	96F5	E01A		001A
0002D	E908		CC30	0002
0002E	2BE4			
0002F	EF10		CC34	0004
00030	1DE4			
00031	E4E5	FFFF		FFFF
00033	23E5			
00034	F406		0001	
00035	0716		CC05	
00036	971A		CC06	

SOURCE STATEMENT
33 * ----- HAL STATEMENT NUMBER 44
34 * ----- HAL STATEMENT NUMBER 45
35 * ----- HAL STATEMENT NUMBER 46
36 HAL46 LH 7,KH1
37 * ----- HAL STATEMENT NUMBER 47
38 HAL47 LH 6,GREVINDX (7)
39 LH 0,GREVDATA (6)
40 LR 5,7
41 AH 5,KH1
42 LH 6,GREVINDX (5)
43 SE 0,GREVDATA (6)
44 BNM HAL47A
45 LPCR 0,0
46 HAL47A CE 0,TOLERANC
47 BNH HAL49
48 * ----- HAL STATEMENT NUMBER 48
49 LH ACC,FAILCNTR (7)
50 AH ACC,KH1
51 E HAL49A
52 * ----- HAL STATEMENT NUMBER 49
53 HAL49 SR ACC,ACC
54 HAL49A STH ACC,FAILCNTR (7)
55 * ----- HAL STATEMENT NUMBER 50
56 AH 7,KH1
57 CH 7,KH4
58 BL HAL47
59 * ----- HAL STATEMENT NUMBER 51
60 LH ACC,FAILCNTR+1
61 AH ACC,FAILCNTR+2
62 AH ACC,FAILCNTR+3
63 BZ HAL79
64 * ----- HAL STATEMENT NUMBER 52
65 * ----- HAL STATEMENT NUMBER 53
66 LH 5,BIIPTR
67 LH 5,FLAGS (5)
68 LH 4,KH1
69 SLL 4,(S5)
70 LH 3,FAULTPPS
71 * ----- HAL STATEMENT NUMBER 54
72 LH 7,KH1
73 LH ACC,FAILCNT
74 * ----- HAL STATEMENT NUMBER 55
75 HAL55 CH ACC,FAILCNTR (7)
76 BH HAL57
77 * ----- HAL STATEMENT NUMBER 56
78 OR 3,4
79 B HAL58
80 * ----- HAL STATEMENT NUMBER 57
81 HAL57 LR 5,4
82 XHI 5,'FFFF'
83 NE 3,5
84 * ----- HAL STATEMENT NUMBER 58
85 HAL58 SFL 4,1
86 AH 7,KH1
87 CH 7,KH4

NOTE:

HAL/S Statement numbers refer to an intermediate HAL/S source and do not correspond to listing in Appendix A.1 or A.2.

00012000
 00013000
 00014000
 00015000
 00016000
 00017000
 00018000
 00019000
 00020000
 00021000
 00022000
 00023000
 00024000
 00025000
 00026000
 00027000
 00028000
 00029000
 00030000
 00031000
 00032000
 00033000
 00034000
 00035000
 00036000
 00037000
 00038000
 00039000
 00040000
 00041000
 00042000
 00043000
 00044000
 00045000
 00046000
 00047000
 00048000
 00049000
 00050000
 00051000
 00052000
 00053000
 00054000
 00055000
 00056000
 00057000
 00058000
 00059000
 00060000
 00061000
 00062000
 00063000
 00064000
 00065000
 00066000

ORIGINAL PAGE IS OF POOR QUALITY

LCC	OBJECT	CCDF	ADR1	ADR2	SOURCE STATEMENT			GPC	VEF9.1	14.52	07/01/75
00037	IA36		002B	000D	88	BL	HAL55				00067000
00038	FB41		0010		89	STH	3, FAULTPFS				00068000
					90	*	-----	HAL STATEMENT NUMBER 59			00069000
00039	9T2D		CCCC		91	LH	7, BITPTR				00070000
0003A	9CF6	E001	C001		92	LH	4, HASKS (7)				00071000
0003C	9DF6	E007	C007		93	LH	5, SHIFTS (7)				00072000
0003E	9FF6	E00B	C00B		94	LH	7, FLAGS (7)				00073000
00040	23E4				95	NR	3, 4				00074000
00041	F3F6			003E	96	SRL	3, (S5)				00075000
00042	8B26		C0C9		97	SH	3, KH3				00076000
00043	1C1C		C04B	0007	98	BZ	CASE1				00077000
00044	8B16		C0C5		99	SH	3, KH1				00078000
					100	*	-----	HAL STATEMENT NUMBER 66			00079000
00045	1C6C		C061	001B	101	CASE2	BZ	HAL79			00080000
00046	8B16		C0C5		102	SH	3, KH1				00081000
00047	1C1C		C04F	0007	103	BZ	CASE3				00082000
00048	8E16		C0C5		104	SH	3, KH1				00083000
00049	1C24		C053	0009	105	BZ	CASE4				00084000
0004A	EF58		C0E1	0016	106	B	HAL79				00085000
					107	*	-----	HAL STATEMENT NUMBER 60			00086000
					108	*	-----	HAL STATEMENT NUMBER 61			00087000
0004B	5F22		C0C8		109	CASE1	LH	6, KH6			00088000
0004C	8F3A		C0CE		110	SH	7, KH2				00089000
					111	*	-----	HAL STATEMENT NUMBER 64			00090000
0004E	EE71		001C		112	STH	3, FAILCNTR+2				00091000
0004E	1FC8		C051	0002	113	B	HAL70A				00092000
					114	*	-----	HAL STATEMENT NUMBER 65			00093000
					115	*	-----	HAL STATEMENT NUMBER 67			00094000
0004F	5E26		C0C9		116	CASE3	LH	6, KH3			00095000
					117	*	-----	HAL STATEMENT NUMBER 70			00096000
00050	PE6D		C01B		118	STH	3, FAILCNTR+1				00097000
00051	EE75		C01D		119	HAL70A	STH	3, FAILCNTR+3			00098000
00052	1F10		C057	0004	120	B	SUB				00099000
					121	*	-----	HAL STATEMENT NUMBER 71			00100000
					122	*	-----	HAL STATEMENT NUMBER 72			00101000
00053	9E36		C00D		123	CASE4	LH	6, KH5			00102000
00054	8F16		C0C5		124	SH	7, KH1				00103000
					125	*	-----	HAL STATEMENT NUMBER 75			00104000
00055	EE71		001C		126	STH	3, FAILCNTR+2				00105000
00056	8E6D		C01E		127	STH	3, FAILCNTR+1				00106000
					128	*	-----	HAL STATEMENT NUMBER 76			00107000
					129	*	-----	HAL STATEMENT NUMBER 63			00108000
					130	*	-----	HAL STATEMENT NUMBER 69			00109000
					131	*	-----	HAL STATEMENT NUMBER 74			00110000
00057	9D49		C012		132	SUB	LH	5, SFFAULT			00111000
00058	25E6				133	NR	5, 6				00112000
00059	8E49		C012		134	STH	5, SFFAGIT				00113000
					135	*	-----	HAL STATEMENT NUMBER 62			00114000
					136	*	-----	HAL STATEMENT NUMBER 68			00115000
					137	*	-----	HAL STATEMENT NUMBER 73			00116000
0005A	9T16		C0C5		138	LH	5, KH1				00117000
0005B	F5FC			003F	139	SLL	5, (S7)				00118000
0005C	B4E5	FFFF		FFFF	140	XHI	5, X'FFFF'				00119000
0005E	9E45		0011		141	LH	3, FDSSTATS				00120000
0005F	23E5				142	NR	3, 5				00121000

LCC	CEJECT	CCDE	ADF1	ADR2	SOURCE STATEMENT	GPC VER 9.1 14.52 07/01/75
CC060	EB45		0011		143 STH 3,FDSTATUS	00122000
					144 * ----- HAL STATEMENT NUMBER 77	00123000
					145 * ----- HAL STATEMENT NUMBER 78	00124000
					146 * ----- HAL STATEMENT NUMBER 79	00125000
					147 HAL79 AEXIT	00126000
					148+*****PRTURN TO CALLER*****	
CC061	CCF8	00C0	C000		149+HAL79 LM OLDSTACK RSTORE REGS AT ENTRY	01-AEXIT
CC063	C7EC				150+ BCPE 7,R4 RPTURN TO CALLER	01-AEXIT
					151+*****	

ORIGINAL PAGE IS
OF POOR QUALITY

```

REGISTER SET 0 07000000 08E20000 096E0000 08E20000 025F4A00 00000000 00010000 00040000
REGISTER SET 1 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
FLOATING PT REGS 41319999 00000000 00000000 00000000 00000000 00000000 00000000 00000000
UPDATED PSW 06794ACC 00010000

```

Appendix A.4

Execution Trace

ELPS TIME (SECS)	INSDCT	ICC	INSTRUCT	L.H.SYM.	MNEM	R.H.SYM.	EPAD	OPERAND	EVEN REG	ODD REG	STAT
0.008436500	003247	00678	990C		LH		007AF 07C0		*R0=07C00000	R1=08E20000	4
0.008437700	003248	00679	1P40		LA		007D0		R2=096E0000	*R3=07E20000	4
0.008440500	003249	0067A	E20C		STH		007C3 07D0		R2=096E0000	*R3=07E20000	4
0.008442300	003250	0067B	2BF10058		LA		0093A		R2=096E0000	*R3=093A0000	4
0.008444700	003251	0067D	3B04		STH		007C1 093A		R2=096E0000	*R3=093A0000	4
0.008446500	003252	0067E	7D29		LH		008EC 0000		R4=025F4A00	*R5=00000000	0
0.008448900	003253	0067F	BD41		STH	GFEB_FAU	008E2 0000		R4=025F4A00	*R5=00000000	0
0.008450700	003254	00680	7E7D		LH	CZ1B_G_P	008E5 01FF		*R6=01FF0000	R7=00040000	4
0.008453100	003255	00681	3E45		STH	GRFB_FDI	008E3 01FF		*R6=01FF0000	R7=00040000	4
0.008454900	003256	00682	9F09		LH	CZ1B_G_F	008E4 0007		R6=01FF0000	*R7=00070000	4
0.008457300	003257	00683	BF49		STH	GRFB_SFF	008E4 0007		R6=01FF0000	*R7=00070000	4
0.008459100	003258	00684	9C11		LH	CG1B_S_F	008E6 0007		*R4=00070000	R5=00000000	4
0.008461500	003259	00685	9C3D		STH	GRFB_BIT	008E1 0007		*R4=00070000	R5=00000000	4
0.008463300	003260	00686	9D7D		LH	CZ1B_G_R	008E5 01FF		R4=00070000	*R5=01FF0000	4
0.008464700	003261	00687	9719		LH	CG1B_C_F	008E8 01FF		*R6=01FF0000	R7=00070000	4
0.008466900	003262	00688	25F6		NR				R4=00070000	*R5=01FF0000	C
0.008467300	003263	00689	7F1D		LH	CG1B_C_F	008E9 0000		R6=01FF0000	*R7=00000000	0
0.008469100	003264	0068A	84E701FF		XHI		01FF		R6=01FF0000	*R7=01FF0000	C
0.008470300	003265	0068C	25E7		NR				R4=00070000	*R5=01FF0000	C
0.008472700	003266	0068D	BD0B		STH		0093C 01FF		R4=00070000	*R5=01FF0000	C
0.008474900	003267	0068E	F51A		SEL		00006		R4=00070000	*R5=00070000	C
0.008476500	003268	0068F	36F50007		NHI		0007		R4=00070000	*R5=00070000	C
0.008479300	003269	00691	3D19		STH		0090C 0007		R4=00070000	*R5=00070000	C
0.008481100	003270	00692	9F0B		LH		0093C 01FF		R6=01FF0000	*R7=01FF0000	4
0.008482900	003271	00693	7707		SFL		00003		R6=01FF0000	*R7=003F0000	4
0.008484700	003272	00694	36E70007		NHI		0007		R6=01FF0000	*R7=00070000	C
0.008487500	003273	00696	BFAD		STH		0090D 0007		R6=01FF0000	*R7=00070000	C
0.008488900	003274	00697	9C0B		LH		0093C 01FF		*R4=01FF0000	R5=00070000	4
0.008490700	003275	00698	36E40007		NHI		0007		*R4=00070000	R5=00070000	C
0.008493500	003276	0069A	3C81		STH		0090E 0007		*R4=00070000	R5=00070000	C
0.008495300	003277	0069B	77F30001		LA		00001		*R6=00010000	R7=00070000	C
0.008497700	003278	0069D	3E31		STH		008EE 0001		*R6=00010000	R7=00070000	C
0.008499900	003279	0069E	85760003		CHI		0003		*R6=00010000	R7=00070000	C
0.008502100	003280	006A0	C1F70055		BC		006F7				C
0.008504300	003281	006A2	9DF50026		LH	PTRS	00909 0001		R4=00070000	*R5=00010000	4
0.008507100	003282	006A4	972D		STH		008ED 0001		R4=00070000	*R5=00010000	4
0.008509500	003283	006A5	9FF50029		LH		0090C 0007		R6=00010000	*R7=00070000	4
0.008510900	003284	006A7	9C15		LH	CZ1B_T_S	008F7 0007		*R4=00070000	R5=00010000	4
0.008512100	003285	006A8	2774		NF				R6=00010000	*R7=00070000	C
0.008514500	003286	006A9	3F4D		STH	GRFB_STA	008F5 0007		R6=00010000	*R7=00070000	C
0.008516700	003287	006AA	9DF50017		LH		008FA 0003		R4=00070000	*R5=00030000	4
0.008519500	003288	006AC	3D51		STH		008F6 0003		R4=00070000	*R5=00030000	4
0.008521900	003289	006AD	78F50038		LF		0091C 40666666		*R0=40666666	R1=00000000	4
0.008524300	003290	006AE	312D		STP		008F8 40666666		*R0=40666666	R1=00000000	4
0.008526100	003291	006B0	7C2D		LH		008ED 0001		*R4=00010000	R5=00030000	4
0.008528500	003292	006B1	3FF5901D		LH		00900 0000		R6=00010000	*R7=00000000	C

REGISTER SET 0 C7CCCC00 0PE20000 C96F0000 C93A0000 00010000 00030000 00010000 00000000
 REGISTER SET 1 CCCC0000 C0000000 00000000 00000000 00000000 00000000 00000000 00000000
 FLICATING ET PEGS 40666666 C0000000 00000000 00000000 00000000 00000000 00000000 00000000
 UPDATED PSW 06B4CACC 00010000

EIPS	TIME(SICS)	INSTCT	LOC	INSTRUCT	L.H.SYM.	MNEM	R.H.SYM.	FPAD	OPFRAND	EVEN REG	OLD REG	STAT
0.008530900	003293	006B3	B56D			STH	GFV_FAI	008FD 0000		R6=00010000	*R7=00000000	0
0.008533100	003294	006B4	9DF5801E			LH		00901 0000		R4=00010000	*R5=00000000	0
0.008535900	003295	006B6	BD71			SIH		008FE 0000		R4=00010000	*R5=00000000	0
0.008539300	003296	006B7	9EF5801F			LH		00902 0000		*R6=00000000	R7=00000000	0
0.008540700	003297	006B9	BE75			STH		008FF 0000		*R6=00000000	R7=00000000	0
0.008542500	003298	006BA	9F31			LH		008FE 0001		R6=00000000	*R7=00010000	4
0.008544900	003299	006EB	7AF5E044			LE		00928 41133333		*F2=41133333	F3=00000000	4
0.008547300	003300	006BD	3A81			STE		00922 41133333		*F2=41133333	F3=00000000	4
0.008549500	003301	006BF	7CF5E04A			LE		0092E 41133333		*F4=41133333	F5=00000000	4
0.008552300	003302	006C0	3C85			STE		00924 41133333		*F4=41133333	F5=00000000	4
0.008554700	003303	006C1	7EF5E050			LE		00934 41119999		*F6=41119999	F7=00000000	4
0.008557100	003304	006C3	3F89			STP		00926 41119999		*F6=41119999	F7=00000000	4
0.008558900	003305	006C4	9D4D			LH	GREB_STA	008F5 0007		R4=00010000	*R5=00070000	4
0.008560900	003306	006C5	B0F5FFFF			AHI		FFFF		R4=00010000	*R5=00050000	6
0.008562500	003307	006C7	DE1C			BC		006CF				6
0.008564100	003308	006C8	EAF10062			LA		00944		*R2=C9440000	R3=093A0000	6
0.008566300	003309	006CA	9502			CH		00944 0005		R4=00010000	*R5=00050000	2
0.008567900	003310	006CB	D90C			BC		006CF				2
0.008570100	003311	006CC	9CF6A000			LH		00949 06B6		*R4=C6B60000	R5=00050000	6
0.008571900	003312	006CE	C7F4			BCR		006E6				6
0.008573500	003313	006E6	E4F3071C			BAL	A3GRESCH	0071C		*R4=06B96A00	R5=00050000	6
0.008576600	003314	0071C	C8FC1003	A3GRESCH		SIM		007D0 07C00000				6
0.008588400	003315	0071E	980C			LH		007C3 07D0		*R0=C7D00000	F1=08F20000	6
0.008589600	003316	0071F	EB50			LA		007E4		R2=C9440000	*R3=C7F40000	6
0.008592400	003317	00720	FB0C			STH		007D3 07E4		R2=C9440000	*R3=C7F40000	6
0.008594200	003318	00721	2BF1005C			LA		0093E		R2=C9440000	*R3=C93F0000	6
0.008596600	003319	00723	FB04			STH		007D1 093E		R2=C9440000	*R3=C93F0000	6
0.008598200	003320	00724	7FF30001			LA		00001		R6=C0000000	*R7=00010000	6
0.008601000	003321	00726	BF35			STH		008EF 0001		R6=C0000000	*R7=00010000	6
0.008603400	003322	00727	B5F70003			CHI		0003		R6=C0000000	*R7=00010000	6
0.008605000	003323	00729	D96C			PC		00745				6
0.008607200	003324	0072A	9EF5E02C			LH		0090F 0001		*R6=00010000	R7=00010000	6
0.008609400	003325	0072C	9DF5E02D			LH		00910 0002		R4=C6B86A00	*R5=00020000	6
0.008611600	003326	0072F	7BF5C03E			LE		00922 41133333		*F0=41133333	F1=00000000	6
0.008616000	003327	00730	58F5A03F			SE		00924 41133333		*F0=C0000000	F1=00000000	2
0.008617400	003328	00732	78D8			LFCR				*F0=R0000000	F1=00000000	2
0.008619000	003329	00733	DA0A			BC		00732				2
0.008623200	003330	00734	48F90016			CF		003F8 40666666		*F0=R0000000	F1=00000000	6
0.008625200	003331	00736	DE1C			PC		0073E				6
0.008627000	003332	0073E	9F35			LH		009FF 0001		R6=C0010000	*R7=00010000	6
0.008630800	003333	0073F	A1F5E01A			ZH	GFV_FAI	008FD 0000				6
0.008632600	003334	00741	EFF30001			LA		00001		R6=C0010000	*R7=00010000	6
0.008634000	003335	00743	8735			AH		0092F 0001		R6=C0010000	*R7=00020000	4
0.008636000	003336	00744	DE7F			PC		00726				4
0.008638800	003337	00726	BF35			STH		008EF 0002		R6=C0010000	*R7=00020000	4
0.008641200	003338	00727	B5F70003			CHI		0003		R6=C0010000	*R7=00020000	4

HAL/S Statement Numbers Correspond to

ORIGINAL PAGE IS OF POOR QUALITY

Execution Segment



```

REGISTER SET 0  C7DCCCC0 08E20000 09440000 C53E0000 06E86A00 00020000 00010000 00020000
REGISTER SET 1  C0CCCC00 00000000 00000000 00000000 00000000 00000000 00000000 00000000
PLICATING PT FIGS 8CCCC0C0 00000000 41133333 00000000 41133333 00000000 41119999 00000000
UPDATED ESW      072ACACC 00010000
    
```

FLPS TIME(SECS)	INSTRCT	LOC	INSTRUCT	L.H.SYM.	MNEM	F.H.SYM.	FPAD	OPFPAND	PVTN REG	OPD REG	STAT
0.008642800	003339	00729	D96C		BC		00745				C
0.008645000	003340	0072A	9EF5E02C		LH		00910 0002		*R6=00020000	F7=00020000	4
0.008647200	003341	0072C	9D75F02D		LH		00911 0003		R4=06E86A00	*R5=00030000	4
0.008649400	003342	0072E	78F5C03E		LE		00924 41133333		*F0=41133333	F1=00000000	4
0.008653800	003343	00730	58F5A03E		SE		00926 41119999		*F0=401999A0	F1=00000000	4
0.008655200	003344	00732	78F8	66	LECR				*F0=001999A0	F1=00000000	C
0.008656800	003345	00733	DA0A		BC		00732				C
0.008658200	003346	00732	78F8		LECR				*F0=401999A0	F1=00000000	4
0.008659800	003347	00733	DA0A		BC		00732				4
0.008664000	003348	00734	48F90016		CE		008F8 40666666		*F0=401999A0	F1=00000000	C
0.008666000	003349	00736	DE1C		BC		0073E				C
0.008667800	003350	0073E	9F35	68	LH		008EF 0002		F6=00020000	*F7=00020000	4
0.008671600	003351	0073F	A1F5E01A		ZH		008FF 0000				4
0.008673400	003352	00741	EFF30001		LA		00001		R6=00020000	*R7=00010000	4
0.008674900	003353	00743	8735	69	AH		008EF 0002		R6=00020000	*R7=00030000	4
0.008676800	003354	00744	DE7E		BC		00726				4
0.008679600	003355	00726	BF35		STH		008EF 0003		F6=00020000	*F7=00030000	4
0.008682000	003356	00727	B5E70003	65	CHI		0003		R6=00020000	*R7=00030000	C
0.008683600	003357	00729	D96C		BC		00745				4
0.008685800	003358	0072A	9EF5E02C		LH		00911 0003		*R6=00030000	R7=00030000	4
0.008688000	003359	0072C	9DF5E02D		LH		00912 0001		R4=06E86A00	*R5=00010000	4
0.008690200	003360	0072E	78F5C03E		LE		00926 41119999		*F0=41119999	F1=00000000	4
0.008694600	003361	00730	58F5A03E	66	SE		00922 41133333		*F0=001999A0	F1=00000000	C
0.008696000	003362	00732	78F8		LECR				*F0=401999A0	F1=00000000	4
0.008697600	003363	00733	DA0A		BC		00732				4
0.008701800	003364	00734	48F90016		CE		008F8 40666666		*F0=401999A0	F1=00000000	C
0.008703800	003365	00736	DE1C		BC		0073E				C
0.008705600	003366	0073E	9F35	68	LH		008EF 0003		F6=00030000	*F7=00030000	4
0.008709400	003367	0073F	A1F5E01A		ZH		008FF 0000				4
0.008711200	003368	00741	EFF30001		LA		00001		R6=00030000	*R7=00010000	4
0.008712600	003369	00743	8735	69	AH		008EF 0003		R6=00030000	*R7=00040000	4
0.008714600	003370	00744	DE7E		BC		00726				4
0.008717400	003371	00726	BF35		STH		008EF 0004		F6=00030000	*F7=00040000	4
0.008719800	003372	00727	B5E70003	65	CHI		0003		R6=00030000	*R7=00040000	4
0.008721400	003373	00729	D96C		BC		00745				4
0.008724000	003374	00745	EFF30001		LA		00001		R6=00030000	*R7=00010000	4
0.008726400	003375	00747	9DF5E01A	70	LH	GREV_FAI	008FD 0000		R4=06E86A00	*R5=00000000	C
0.008728000	003376	00749	DB1C		BC		00751				4
0.008729800	003377	0074A	B0E70001		AHI		0001		R6=00030000	*R7=00020000	4
0.008732000	003378	0074C	B5E70003		CHI		0003		R6=00030000	*R7=00020000	C
0.008734000	003379	0074E	DE22		BC		00747				C
0.008737200	003380	00747	9DF5E01A		LH		008FE 0000		R4=06E86A00	*R5=00000000	4
0.008738800	003381	00749	DB1C	74	BC		00751				4
0.008740600	003382	0074A	B0E70001		AHI		0001		R6=00030000	*R7=00030000	4
0.008742800	003383	0074C	B5E70003		CHI		0003		R6=00030000	*R7=00030000	4
0.008744600	003384	0074E	DE22		BC		00747				4

Execution
Segment
(Continued)

REGISTER SET 0 C7D00000 08F20000 09440000 093E0000 06F86A00 00000000 00030000 00030000
 REGISTER SET 1 C0000000 C0000000 00000000 00000000 00000000 00000000 00000000 00000000
 FLICATING PT REGS 401999AC 00000000 41133333 00000000 41133333 00000000 41119999 00000000
 UPDIATED PSW C749CACC 00010000

ELPS TIME(SECS)	INSTCT	ICC	INSTFUCT	L.H.SYM.	MNEM	R.H.SYM.	EPAD	OPPRAND	EVEN REG	ODD REG	STAT
0.008748000	C03385	00747	9DF5E01A		LH		008FF 0000		R4=06E86A00	*F5=00000000	0
0.008749600	C03386	00749	DE1C		BC		00751				0
0.008751400	C03387	0074A	B0E70001		AHI		0001		P6=00030000	*R7=00040000	4
0.008753600	C03388	0074C	B5E70003		CHI		0003		R6=00030000	*R7=00040000	4
0.008755600	C03389	0074F	DE22		BC		00747				4
0.008758000	C03390	0074F	C7F70052		BC		007A3				4
0.008769600	003391	007A3	CCF80000		LM		007D0 07C0093E				4
				99							
REGISTER SET 0	C7C0C93E	08E207E4	09440000	093A	C00	06F86A00	00050000	00000000	00010000		
0.008771000	C03392	007A5	C7E4		BCR		006F8				4
0.008772800	C03393	006E8	9F2D		LH		008FD 0001		R6=00000000	*R7=00010000	4
0.008774200	C03394	006E9	9D6D		LH	GFEV_FAI	008FD 0000		R4=06E86A00	*R5=00000000	0
0.008777400	C03395	006EA	BDF5E01D		STH		00900 0000		P4=06E86A00	*R5=00000000	0
0.008779200	C03396	006EC	9E71		LH		008FF 0000		*R6=00000000	R7=00010000	0
0.008782600	C03397	006ED	BFF5F01E		STH		00901 0000		*R6=00000000	R7=00010000	0
0.008784000	C03398	006EF	9D75		LH		008FF 0000		R4=06E86A00	*F5=00000000	0
0.008787200	C03399	006F0	BDF5E01F		STH		00902 0000		R4=06E86A00	*R5=00000000	0
0.008788800	C03400	006F2	EEF30001		LA		000C1		*R6=00010000	R7=00010000	0
0.008790600	C03401	006F4	8631		AH		008FE 0001		*R6=00020000	R7=00010000	4
0.008793800	C03402	006F5	C7F7085A		BC		0069D				4
0.008796600	C03403	0069D	BE31		STH		008FE 0002		*R6=00020000	R7=00010000	4
0.008798800	C03404	0069E	B5E60C03		CHI		0003		*R6=00020000	R7=00010000	0
0.008801000	C03405	006A0	C1F70055		BC		006F7				0
0.008803200	C03406	006A2	9DF5C026		LH		0090A 0004		R4=06E86A00	*R5=00040000	4
0.008806000	C03407	006A4	BD2F		STH		008ED 0004		R4=06E86A00	*R5=00040000	4
0.008808400	C03408	006A5	9FF5C029		LH		0090D 0007		R6=00020000	*R7=00070000	4
0.008809800	C03409	006A7	9C15		LH	CZ1B_T_S	008E7 0007		*R4=00070000	R5=00040000	4
0.008811000	C03410	006A8	2774		NE				R6=00020000	*R7=00070000	0
0.008813400	C03411	006A9	BF4D		STH	GREB_STA	008F5 0007		R6=00020000	*R7=00070000	0
0.008815600	C03412	006AA	9DF5C017		LH		008FB 0003		R4=00070000	*R5=00030000	4
0.008818400	C03413	006AC	BD51		STH		008F6 0003		R4=00070000	*R5=00030000	4
0.008820900	C03414	006AD	78F5C038		LH		0091E 40666666		*F0=40666666	F1=00000000	4
0.008823200	C03415	006AF	3A2D		STE		008F8 40666666		*F0=40666666	F1=00000000	4
0.008825000	C03416	006B0	9C2D		LH		008FD 0004		*F4=00040000	F5=00000000	4
0.008827400	C03417	006B1	9FF5801D		LH		00903 0000		R6=00020000	*R7=00000000	0
0.008829400	C03418	006B3	BF6D		STH	GREV_FAI	008FD 0000		R6=00020000	*R7=00000000	0
0.008832000	C03419	006B4	9DF5801E		LH		00904 0000		R4=00040000	*F5=00000000	0
0.008834800	C03420	006B6	BD71		STH		008FE 0000		R4=00040000	*R5=00000000	0
0.008837200	C03421	006B7	9FF5801F		LH		00905 0000		*F6=00000000	F7=00000000	0
0.008839600	C03422	006B9	BE75		STH		008FF 0000		*R6=00000000	R7=00000000	0
0.008841400	C03423	006BA	9F31		LH		008FF 0002		R6=00000000	*R7=00020000	4
0.008843800	C03424	006BH	7AF5E044		LE		0092A 41233333		*F2=41233333	F3=00000000	4
0.008846200	C03425	006BD	3A81		STE		00922 41233333		*F2=41233333	F3=00000000	4
0.008848400	C03426	006BE	7CT5E04A		LE		00930 41233333		*F4=41233333	F5=00000000	4
0.008851200	003427	006C0	3C85		STE		00924 41233333		*F4=41233333	F5=00000000	4

↑ Execution Segment

ORIGINAL PAGE IS OF POOR QUALITY

BEPCBE GRI RGA IDIB AND GRE BELEM:

GFEB FAULTPAIRS=
00000000
CZ1B G RGA FAULT=
00000000
GFEB FDIRSTATUS=
00000000
CZ1B G RGA FDIR STAT=
11111111
GREB SFFAULT=
000
GREB EITE=
000
CZ1B G RGA SF FAULT=
111
CG1B S RGA EITE=
111
GREV I=
0
RIGFTR=
0
GREV J=
0
GFEF STATUS=
000
CG1E C RGA VAL FLG=
11111111
CG1B C RGA TST FLG=
00000000
GREV FAILCNT=
0
CZ1B T SMRD PRES RGA=
111
CG1V R RGA N=
3
3
3
GREV ICLEFFANCE=
C.0
CG1V R RGA TCI=
E.9999997E-01
E.9999997E-01
E.9999997E-01
GREV FAILCNTB=
0
0
0
CG1V R RGA FAILCNTB=
0
0
0
0

Appendix A.5

Data Bases, Before and After Test

0
0
0
0
0

GFEV DATA=

C.O
C.O
C.O

CG1V C RGA DATA=

1.1999998E+00
2.1999998E+00
3.1999998E+00
1.1999998E+00
2.1999998E+00
3.1999998E+00
1.0999994E+00
2.0999994E+00
3.0999994E+00

CG1E F RGA SF STAT=

000

CG1E F RGA DATA GOOD=

0

AFTER GHI RGA FDIB AND GRE 3ELEM:

GFEF FAULTPAIRS=

00000000

C21E G RGA FAULT=

00000000

GREB FIRSTATUS=

11111111

C21E G RGA FDIB STAT=

11111111

GFEF SFFAULT=

111

GREB EITE=

111

C21E G RGA SF FAULT=

111

CG1B S RGA EITE=

111

GFEV I=

4

EIPIR=

7

GREV J=

4

GFEF STATUS=

111

CG1B C RGA VAL FLG=

11111111

CG1E C RGA TST FLG=

ORIGINAL PAGE IS
OF POOR QUALITY