ICOPS Application Development Manual

J. L. Michel
G. M. Stabler
R. J. Harrington

March 1, 1975

printed - December 6, 1976

This document contains more details about the various stages involved in the implementation of an application under ICOPS.

This research is being supported by the National Science Foundation Grant GJ-41539, the Office of Naval Research Contract N00014-75-A-0427, and the Brown University Division of Applied Mathematics. Principal Investigator - Andries van Dam.

TABLE OF CONTENTS

1 INTRODUCTION

The following sections contain information on assembling, building and debugging an ICOPS application system. A knowlege of A System for Interconnected Processing by G. M. Stabler (particularly Chapter 3) and the Brown University Graphics System is assumed.

Someday when the Language for Systems Development compiler is operational and supports code generation for both the 360 and BUGS, ICOPS application procedures can be written in this higher level language. Until then applications must be written in an assembly language similar to M4A assembly language and assembled for both machines by using either a 360 or a BUGS macro library to generate code for the appropriate machine.

2 THE ICP ASSEMBLY LANGUAGE

The instruction set is close to that of a 360 and includes a complete set of logical, arithmetic, character handling, and testing and branching instructions.

Note that instructions are based on 16 bit halfwords (i.e., a Load instruction assembled for the 360 generates a Load Halfword). Computations on both machines are done in halfword arithmetic (i.e., an Add instruction assembled for the 360 generates an Add Halfword). Unfortunately addresses are not the same length on both machines; an address is a fullword on the 360 and a halfword on BUGS. For this reason, Real Address instructions were defined which generate the appropriate address manipulation instructions on both machines. Also there is a special macro, SELECT, for generating a branch instruction from an index into a table of addresses.

2.1 INSTRUCTION SET

Below are given the instructions, mnemonics, and formats comprising this instruction set. For more information on the

various instructions, refer to the M4A Principles of Operation manual.

ADD	A	R1,D2(X2,B2)
ADD HALFWORDS	AH	D1(B1),D2(B2)
ADD HEX DIGIT	AX	R1,I2
ADD IMMEDIATE	AI	R1S,R1F,IH2
ADD LOGICAL	AL	R1,D2(X2,B2)
ADD LOGICAL IMMEDIATE	ALI	R1S,R1F,IH2
ADD LOGICAL REGISTER	ALR	R1,R2
ADD REGISTER	AR	R1,R2
ADD TO HALFWORD IMMEDIATE	AHI	D1(B1),I2
AND	N	R1,D2(X2,B2)
AND CHARACTERS	NC	D1(L,B1),D2(B2)
AND CHARACTERS LONG	NCL	D1(LR,B1),D2(B2)
AND IMMEDIATE	NI	R1S,R1F,IH2
AND REGISTER	NR	R1,R2
AND TO BYTE IMMEDIATE	NBI	D1(B1),I2
BRANCH AND LINK	BAL	R1,D2(X2,B2)
BRANCH AND LINK REGISTER	BALR	R1,R2
BRANCH ON COUNT	BCT	R1,D2(X2)
BRANCH ON COUNT REGISTER	BCTR	R1,R2
BRANCH ON INDEX HIGH	BXH	R1,R3,D2(X2)
BRANCH ON INDEX LOW OR EQUAL	BXLE	R1,R3,D2(X2)
COMPARE	C	R1,D2(X2,B2)
COMPARE HALFWORD TO IMMEDIATE	CHI	D1(B1),I2
COMPARE HALFWORDS	CH	D1(B1),D2(B2)
COMPARE IMMEDIATE	CI	R1F,IH2
COMPARE LOGICAL	CL	R1,D2(X2,B2)
COMPARE LOGICAL CHARACTERS	CLC	D1(L,B1),D2(B2)
COMPARE LOGICAL CHARACTERS LONG	CLCL	D1(LR,B1),D2(B2)
COMPARE LOGICAL HALFWORDS	CLH	D1(B1),D2(B2)
COMPARE LOGICAL IMMEDIATE	CLI	R1F,IH2
COMPARE LOGICAL REGISTER	CLR	R1,R2
COMPARE LOGICAL TO BYTE	CLB	R1,D2(X2,B2)
COMPARE LOGICAL TO BYTE IMMEDIATE	CLBI	D1(B1),I2
* COMPARE REAL ADDRESSES	* CRA	R1,D2(X2,B2)
COMPARE REGISTER	CR	R1,R2
CONVERT BINARY TO DECIMAL	*CBD	R1,D2(X2,B2)
* CONVERT BINARY TO HEX	*CBX	R1,D2(X2,B2)
CONVERT DECIMAL TO BINARY	*CDB	R1,D2(X2,B2)
CONVERT HEX TO BINARY	*CXB	R1,D2(X2,B2)
DIVIDE **	D	R1,D2(X2,B2)
DIVIDE HALFWORDS **	DH	D1(B1),D2(B2)
DIVIDE IMMEDIATE **	DI	R1S,R1F,IH2
DIVIDE REGISTER **	DR	R1,R2
EXCLUSIVE OR	X	R1,D2(X2,B2)
EXCLUSIVE OR CHARACTERS	XC	D1(L,B1),D2(B2)

	EXCLUSIVE OR CHARACTERS LONG	M XCL	D1(LR,B1),D2(B2)
	EXCLUSIVE OR IMMEDIATE	XI	R1S,R1F,IH2
	EXCLUSIVE OR REGISTER	XR	R1,R2
	EXCLUSIVE OR TO BYTE IMMEDIATE	M XBI	D1(B1),I2
	EXECUTE	EX	R1,D2(X2,B2)
	FILL	M FILL	D1(L,B1),D2(B2)
	FILL LONG	M FILLL	D1(LR,B1),D2(B2)
	FREE CONTROLLED STORAGE	M FREE	R1,D2(X2,B2)
	FREE CONTROLLED STORAGE REGISTER	M FREER	R1,R2
	GET CONTROLLED STORAGE	M GET	R1,D2(X2,B2)
	GET CONTROLLED STORAGE REGISTER	M GETR	R1,R2
	INSERT BYTE	M IB	R1,D2(X2,B2)
	LOAD	L	R1,D2(X2,B2)
	LOAD ADDRESS	LA	R1,D2(X2,B2)
	LOAD AND ZERO	M LZ	R1,D2(X2,B2)
	LOAD BYTE	M LB	R1,D2(X2,B2)
	LOAD COMPLEMENT	M LC	R1,D2(X2,B2)
	LOAD COMPLEMENT REGISTER	LCR	R1,R2
	LOAD DEFERRED	LD	R1,D2(X2,B2)
	LOAD DEFERRED IMMEDIATE	M LDI	R1S,IA2
	LOAD DEFERRED REGISTER	LDR	R1,R2
	LOAD HEX DIGIT	M LX	R1,I2
	LOAD IMMEDIATE	M LI	R1S,IH2
*	LOAD MULTIPLE PARAMETER ADDRESS *	LMPA	R1,R3,N2,B2
*	LOAD MULTIPLE REAL ADDRESSES	LMRA	R1,R3,D2(B2)
*	LOAD MULTIPLE REAL ADDRESSES DEFERRED	LMRAD	R1,R3,D2(B2)
	LOAD NEGATIVE	M LN	R1,D2(X2,B2)
	LOAD NEGATIVE REGISTER	LNR	R1,R2
*	LOAD PARAMETER ADDRESS *	LPA	R1,N2,B2
	LOAD POSITIVE	M LP	R1,D2(X2,B2)
	LOAD POSITIVE REGISTER	LPR	R1,R2
*	LOAD REAL ADDRESS	LRA	R1,D2(X2,B2)
	LOAD REGISTER	LR	R1,R2
	LOAD SIGNED BYTE	M LSB	R1,D2(X2,B2)
	MOVE CHARACTERS	MVC	D1(L,B1),D2(B2)
	MOVE CHARACTERS LONG	M MVCL	D1(LR,B1),D2(B2)
	MOVE CHARACTERS NON-PROPOGATING	M MVCN	D1(L,B1),D2(B2)
	MOVE CHARACTERS NON-PROPOGATING LONG	M MVCNL	D1(LR,B1),D2(B2)
	MOVE HALFWORDS	M MVH	D1(B1),D2(B2)
*	MOVE REAL ADDRESS	MVRA	D1(B1),D2(B2)
	MOVE TO BYTE IMMEDIATE	M MVBI	D1(B1),I2
	MOVE TO HALFWORD IMMEDIATE	M MVHI	D1(B1),I2
	MULTIPLY **	M	R1,D2(X2,B2)
	MULTIPLY HALFWORDS **	M MH	D1(B1),D2(B2)
	MULTIPLY IMMEDIATE **	M MI	R1S,R1F,IH2
	MULTIPLY REGISTER **	MR	R1,R2
	OR	O	R1,D2(X2,B2)
	OR CHARACTERS	OC	D1(L,B1),D2(B2)
	OR CHARACTERS LONG	M OCL	D1(LR,B1),D2(B2)
	OR IMMEDIATE	OI	R1S,R1F,IH2
	OR REGISTER	OR	R1,R2

OR TO BYTE IMMEDIATE	OBI	D1(B1),I2
SCAN LEFT EQUAL	SLE	D1(L,B1),D2(B2)
SCAN LEFT EQUAL LONG	SLEL	D1(LR,B1),D2(B2)
SCAN LEFT NOT EQUAL	SLNE	D1(L,B1),D2(B2)
SCAN LEFT NOT EQUAL LONG	SLNEL	D1(LR,B1),D2(B2)
SCAN RIGHT EQUAL	SRE	D1(L,B1),D2(B2)
SCAN RIGHT EQUAL LONG	SREL	D1(LR,B1),D2(B2)
SCAN RIGHT NOT EQUAL	SRNE	D1(L,B1),D2(B2)
SCAN RIGHT NOT EQUAL LONG	SRNEL	D1(LR,B1),D2(B2)
SCAN USING TABLE LEFT	STL	D1(L,B1),D2(B2)
SCAN USING TABLE LEFT LONG	STLL	D1(LR,B1),D2(B2)
SCAN USING TABLE RIGHT	STR	D1(L,B1),D2(B2)
SCAN USING TABLE RIGHT LONG	STRL	D1(LR,B1),D2(B2)
SHIFT LEFT ALGEBRAIC	SLA	R1,D2(B2)
SHIFT LEFT ALGEBRAIC IMMEDIATE	SLAI	R1,I2
SHIFT LEFT DOUBLE ALGEBRAIC	SLDA	R1,D2(B2)
SHIFT LEFT DOUBLE ALGEBRAIC IMMEDIATE	SLDAI	R1,I2
SHIFT LEFT DOUBLE LOGICAL	SLDL	R1,D2(B2)
SHIFT LEFT DOUBLE LOGICAL IMMEDIATE	SLDLI	R1,I2
SHIFT LEFT LOGICAL	SLL	R1,D2(B2)
SHIFT LEFT LOGICAL IMMEDIATE	SLLI	R1,I2
SHIFT RIGHT ALGEBRAIC	SRA	R1,D2(B2)
SHIFT RIGHT ALGEBRAIC IMMEDIATE	SRAI	R1,I2
SHIFT RIGHT DOUBLE ALGEBRAIC	SRDA	R1,D2(B2)
SHIFT RIGHT DOUBLE ALGEBRAIC IMMEDIAT	SRDAI	R1,I2
SHIFT RIGHT DOUBLE LOGICAL	SRDL	R1,D2(B2)
SHIFT RIGHT DOUBLE LOGICAL IMMEDIATE	SRDLI	R1,I2
SHIFT RIGHT loGICAL	SRL	R1,D2(B2)
SHIFT RIGHT LOGICAL IMMEDIATE	SRLI	R1,I2
STORE	ST	R1,D2(X2,B2)
STORE BYTE	STB	R1,D2(X2,B2)
STORE DEFERRED IMMEDIATE	STDI	R1F,IA2
STORE DEFERRED REGISTER	STDR	R1,R2
STORE DEFERRED	STD	R1,D2(X2,B2)
STORE LEAST SIGNIFICANT PRODUCT	STLP	R1,D2(X2,B2)
STORE MOST SIGNIFICANT PRODUCT	STMP	R1,D2(X2,B2)
STORE MULTIPLE REAL ADDRESSES	STMRA	R1,R3,D2(B2)
STORE MULT. REAL ADDRESSES DEFERRED	STMRAD	R1,R3,D2(B2)
STORE REAL ADDRESS	STRA	R1,D2(X2,B2)
SUBTRACT	S	R1,D2(X2,B2)
SUBTRACT ADDRESS	SA	R1,D2(X2,B2)
SUBTRACT FROM HALFWORD IMMEDIATE	SHI	D1(B1),I2
SUBTRACT HALFWORDS	SH	D1(B1),D2(B2)
SUBTRACT HEX DIGIT	SX	R1,I2
SUBTRACT IMMEDIATE	SI	R1S,R1F,IH2
SUBTRACT LOGICAL	SL	R1,D2(X2,B2)
SUBTRACT LOGICAL IMMEDIATE	SLI	R1S,R1F,IH2
SUBTRACT LOGICAL REGISTER	SLR	R1,R2
SUBTRACT REGISTER	SR	R1,R2
SWAP	SWP	R1,D2(X2,B2)
SWAP REGISTER	SWPR	R1,R2

TEST AND BRANCH MINUS	TBM	R1,D2(X2)
TEST AND BRANCH MINUS REGISTER	TBMR	R1,R2
TEST AND BRANCH NOT MINUS	TBNM	R1,D2(X2)
TEST AND BRANCH NOT MINUS REGISTER	TBNMR	R1,R2
TEST AND BRANCH NOT PLUS	TBNP	R1,D2(X2)
TEST AND BRANCH NOT PLUS REGISTER	TBNPR	R1,R2
TEST AND BRANCH NOT ZERO	TBNZ	R1,D2(X2)
TEST AND BRANCH NOT ZERO REGISTER	TBNZR	R1,R2
TEST AND BRANCH PLUS	TBP	R1,D2(X2)
TEST AND BRANCH PLUS REGISTER	TBPR	R1,R2
TEST AND BRANCH ZERO	TBZ	R1,D2(X2)
TEST AND BRANCH ZERO REGISTER	TBZR	R1,R2
TEST UNDER MASK BYTE IMMEDIATE	TMBI	D1(B1),I2
TEST UNDER MASK IMMEDIATE	TMI	R1F,IH2
TEST UNDER MASK REGISTER	TMR	R1,R2
TRANSLATE	TR	D1(L,B1),D2(B2)
TRANSLATE LONG	TRL	D1(LR,B1),D2(B2)

* LPA R1,N2,B2 loads the address of a parameter from the parameter list. N2 is the number beginning with zero of the parameter to be loaded, and B2 is the register that points to the parameter list. N2 defaults to the first parameter and B2 defaults to the standard parameter register set up by the CALL macro. LMPA R1,R3,N2,B2 may be used to load multiple parameter addresses. Starting with parameter N2, addresses are loaded into registers R1 through R3.

** Multiply and divide are not implemented as described in the M4A Principles of Operations. Instead they were implemented using 360 conventions. For MULTIPLY

(op1+1) * (op2) the result is in (op1,op1+1)

and for DIVIDE

(op1,op1+1)/(op2) the quotient is in (op1+1) and the remainder is in (op1).

2.2 RESTRICTIONS

An ICP application program must follow certain conventions in register usage. The ENTER macro generates the following register equates.

MNEMONIC	360 REG	BUGS REG	USE
RW	0	13	work register
RP	1	2	parameter register
R2	2	2	general
...	...	...	
R12	12	12	general
unnamed	14	14	link register
unnamed	15	14	entry register
unnamed	13		base register
unnamed		0	machine status register
unnamed		1	PC and base register
unnamed		15	stack frame pointer

RP and unnamed registers can not be referenced.

R12 can not be referenced in a re-entrant routine.

RW can not be used as an index or base register or as a length register in a "long" instruction.

CLC(L) sets R2 if the compare is unsuccessful.

CVB uses R2.

CVD uses R2 and R3.

SLE, SLEL, SLNE, SLNEL, SRE, SREL, SRNE, SRNEL use R2.

SRCH uses R2, R3, R4, and R5.

STL, STLL, STR, STRL use R2 and R3.

SVCD uses R2 and R3.

SVCS uses R2.

The following BUGS instructions are not supported on the 360: DEQ, ENQ, EXCC, LXB, LXBR, MVA, POPH, POPHL, POPM, PSHH, PSHHL, PSHM, RST, SD, SIO, SIOR, SRCH, SS, SVC, SVCD, SVCS,

TRB, TSL, WST. Fixed BUGS procedures can use any BUGS instructions and fixed 360 procedures can use any 360 instruction that does not conflict with any ICP insruction (i.e., LD is Load Deferred not floating point Load Long).

2.3 PROCEDURE FORMAT

ICP application procedures should have the following format:

```
          COPY  OPS364A
NAME      ENTER
          <automatic variables>
          ENDAUTO
          ...
          RETURN
          ...
          PARMS (...)
          END
```

The ENTER, ENDAUTO, RETURN, and PARMS macros are described in the next section.

2.4 SOME USEFUL MACROS

Procedures should be called using the CALL macro to generate the proper linkage between procedures.

```
NAME      CALL PROCEDURE,<,(ADDRESS_PARAMETERS)<,VL>>
                  <,MF=(E,PLIST)>
```

PROCEDURE is the name of the procedure to be called or a register specification, (R1) where R1 contains the address of the procedure. ADDRESS_PARAMETERS is a list of one or more addresses of parameters separated by commas. The address of a parameter can be given in a register. VL causes the high order bit of the last parameter address in the list to be set. PLIST is the address of the parameter list.

The parameter list can be generated by the list form of the CALL macro.

```
NAME      CALL ,(ADDRESS PARAMTERS)<,VL>,MF=L
```

where NAME is the address of the parameter list.

The ENTER macro defines a CSECT specified by name, saves registers, and establishes a base register for addressing within the procedure. ENDAUTO indicates the end of any automatic storage. RETURN restores registers and returns to the calling procedure. Re-entrant subroutines should use RENTER, RENDAUTO, and RRETURN instead of ENTER, ENDAUTO, and RETURN.

The PARMS macro must be included to give ICOPS necessary information about parameters. PARMS takes as operands a series of "declarations" describing the type, length, and access (read-only, read-write, or write-only) of each parameter being passed to the procedure. As an example, the statement

```
PARMS (CL8,:H,.H)
```

indicates that a procedure expects three parameters -- a character string of length eight which is read, but not modified; a halfword which is both read and modified; and a halfword which is only modified.

The SELECT macro will generate a branch to an address selected by an index into a table of addresses. SELECT has the following format:

```
NAME      SELECT I,ALIST,<REG=SCRATCH,ORG=N0,INC=N1,
                  HIGH=ERROR1,LOW=ERROR2>
```

I is either the address of a halfword index or a register specification, (R1), where R1 is a register containing the index. If REG=SCRATCH, the index is computed in R1. ALIST is either the address of a list of addresses or a list of one or more addresses separated by commas and enclosed in parentheses. The index will begin at N0 and be incremented by N1. N1 must be 1, 2, 4, or 8. N0 defaults to 0, and N1 defaults to 1. If the index given is too small a branch is taken to ERROR1; if the index is too large a branch is taken to ERROR2. If ERROR1 or ERROR2 is omitted, the index is not checked for that illegal value.

3 GENERATION OF AN APPLICATION

The following describes the major CMS commands which should be used to generate the application system and maintain the various system libraries. Figure 1 illustrates the relationship between the commands and the files being manipulated. Section 4 contains a complete list of these commands.

3.1 OVERVIEW

Starting with the SYSIN files, each procedure is assembled (using SREP) for either the local, remote, or both processors. After assembling the procedure, the SREP command adds the TEXT output of the mainframe or satellite assembler to the appropriate library -- LCLLIB TXTLIB, RMTLIB TXTLIB, or both. The filetype of the TEXT file is then altered to TEXTL or TEXTR in order to distinguish between the two object decks for icpable procedures.

Once the two TXTLIBs have been created, new load modules are produced by running the SYSBLD command. SYSBLD first runs a link edit pre-processor, LEPRE, on the new libraries. Load modules for the local and/or remote processors are then built. For the 360, this is done with the CMS LOAD and GENMOD

commands; for BUGS, with the GMSLINK command. Note that icpable procedures on BUGS are loaded in separate segments so that they can be dynamically loaded and deleted from BUGS memory. The final step in the generation process is to ship the system MODU, the BUGS load module, to the satellite system. This is done with CHARON, a command which supports the transfer of files between the 360 and BUGS.

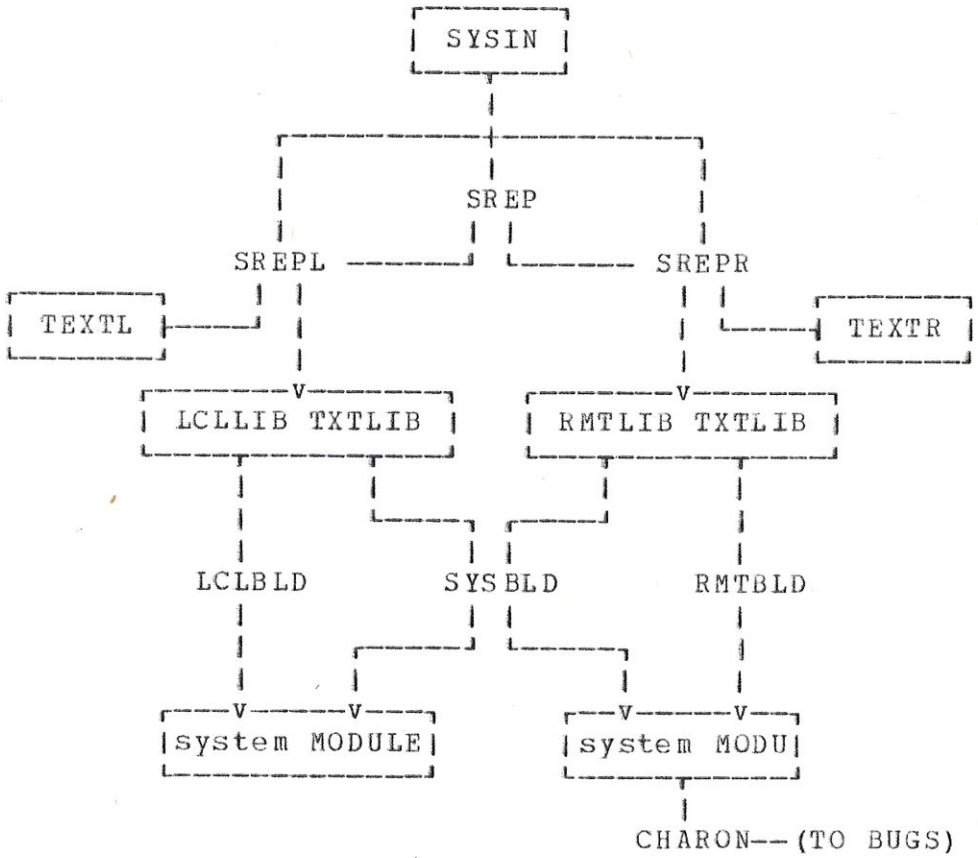

Figure 1. The system Generation Process

3.2 LEPRE

LEPRE requires as input the two text libraries, RMTLIB and LCLLIB, and a data file, LEPRE DATA, containing information about procedures in the application system.

Each command in the LEPRE DATA file is of the form:

keyword (machname) name1 name2 • • • nameN

'keyword' identifies the command type and can take on the following values and meanings:

LIBRARY - 'name1' through 'nameN' specify CMS text libraries in which are to be found the object modules to be located on 'machname'.

FIXED - 'name1' through 'nameN' are the names of modules to be permanently fixed on 'machname'.

ICPABLE - The following names are of modules which may be moved from one processor to the other. They are to be initially located on 'machname'.

RICPABLE - This has the same meaning as ICPABLE, except the modules named are specified to be "reusable".[1]

DEFAULT - This command specifies that any modules not explicitly located by the user are to default to be fixed on 'machname'.

'machname' for this implementation is either '360' or 'BUGS' and takes on the meanings noted above.

---

[1]By reusable, we mean that the static environment of the procedure is completely reinitialized whenever the procedure is entered. For a non-reusable procedure, the static environment on entry is the same as it was left following the last call of the procedure.

As an example of the above, the following commands would produce the allocation - A FIXED on the 360, B ICPABLE and initially on the 360, and C and D FIXED on BUGS:

```
DEFAULT (360)
LIBRARY (360) LCLLIB
LIBRARY (BUGS) RMTLIB
FIXED (360) A
ICPABLE (360) B
FIXED (BUGS) C D
```

After reading and processing the command file, LEPRE builds two text libraries, LWCB and RWCB, which contain procedure tables and control blocks needed by ICOPS.

Note that LEPRE reverses the names of the text decks of all procedures in the text libraries (i.e., a text deck named GENERATE would become ETARENEG).

3.3 DEBUGGING AND REBUILDING

Standard debugging techniques can be used on an ICOPS application system. Load modules can be patched and procedures can be patched or reassembled.

When debugging an ICOPS application, remember that the names of all procedures have been changed. The reverse name must be given when using NEWBUG, FUDD, or MODZAP.

Procedures are reassembled using the appropriate SREP. (See SREP exec in Section 4) Note that the reverse name is included as a SREP parameter since that text deck must be deleted from the text library.

A text deck can be split from either library, and corrected by adding REP cards or by ZAPping the text file. It can then be replaced in the text library. Remember to split and replace the reverse named text file in both libraries.

If REP cards are added to local procedures, just do a local build, LCLBLD. If REP cards are added to remote procedures, do a remote build, RMTBLD, and then CHARON the new MODU to BUGS. If both remote and local procedures are changed, then rebuild the entire system with SYSBLD.

When rebuilding the application system, it is necessary to rerun LEPRE and PT if a change is made to the parameters expected by a procedure or if LEPRE DATA has been changed. If a procedure has been reassembled and no parameters are changed, run LEPRE without PT.

4 GENERATION EXECS

The following is a list of the various EXECs used to build an application system. Optional parameters are indicated by < ... >.

SYSBLD SYSTEM <NOLEPRE> <NOPT>

This exec builds the entire application system. SYSTEM is the entry point on the 360 of the system and will be the name of the module file. LEPRE will be run and Procedure Tables will be assembled unless otherwise specified.

LCLBLD SYSTEM <PT>

This exec builds the 360 module. SYSTEM is the entry point on the 360 of the system and will be the name of the module file. A local Procedure Table will be assembled only if specified.

RMTBLD SYSTEM <PT>

This exec builds the BUGS module. SYSTEM will be the name of the BUGS modu file. A remote Procedure Table will be assembled only if specified.

SREP FILENAME EMANELIF [L or R or I] MACLIB

This exec assembles the procedure specified by FILENAME for either the 360 or BUGS or both. MACLIB is the name of the application system macro library.

ASML FILENAME MACLIB <assembly parameters>

This exec assembles a 360 procedure specified by FILENAME. MACLIB is the name of the system macro library.

ASMR FILENAME MACLIB <assembly parameters>

This exec assembles a BUGS procedure specified by FILENAME. MACLIB is the name of the system macro library.

TXTL FILENAME

This exec replaces the text deck specified by FILENAME in the LCLLIB TXTLIB and alters the file type of the text deck to TEXTL.

TXTR FILENAME

This exec replaces the text deck specified by FILENAME in the RMTLIB TXTLIB and alters the file type of the text deck to TEXTR.

5 THE ICOPS MONITOR, HERMES

One of the major design goals of ICOPS was to allow the user to dynamically examine, evaluate, and modify the manner in which he has divided the tasks in his application between the mainframe and satellite processors. To this end, designed into ICOPS is a terminal-oriented command language with which the user can perform these functions. The following paragraphs describe how the command language monitor is entered and what facilities are available to the user.

5.1 ENTERING THE MONITOR

The design of the command language monitor is such as to make it "invisible" to user programs, that is, user programs do not have to be written with cognizance of the monitor. User programs may, if they wish, invoke the monitor directly by a simple CALL to the appropriate entry point. More often it is expected that the user will wish to enter the monitor at times which are not necessarily synchronized with his program's execution. Therefore, asynchronous entry conditions are defined which cause the monitor to be entered.

On the 360, the monitor is entered when the user's virtual CMS machine receives an external interrupt. Normally, when CMS receives an external interrupt, it enters the DEBUG environment. The ICOPS initialization routine, however, usurps this function and enters instead the ICOPS command language environment.

On BUGS, the satellite system, the relationship of the monitor to the operating system almost exactly parallels the 360 situation. The action causing entry to the monitor is a panel interrupt, an interrupt analogous to the 360's external interrupt. Normally, this interrupt is undefined (i.e., there is no system routine to handle it). The BUGS operating system therefore enters the satellite debugging environment. When ICOPS is run, the initialization routine defines the command language monitor as the routine to handle panel interrupts, and the monitor is thereafter entered on any panel interrupt.

5.2 THE ICOPS COMMAND LANGUAGE

The following lists all commands currently supported by the monitor. Some commands are, as noted, specific to one system or the other. In the command descriptions, underlining indicates the shortest abbreviation recognized by the monitor.

An OR bar (|) between parameters indicates a choice is to be made between the parameters.

BUGS

```
| BUGS cmndline |
```

The BUGS command provides an interface between the satellite monitor and the BUGS operating system. The 'cmndline' as typed by the user is passed on to the operating system just as if it had been entered in the normal BUGS command environment. This allows the user to temporarily suspend execution of his application and execute operating system functions (e.g., file erasure, file listing, etc.).

CMS

```
| CMS cmndline |
```

The CMS command is the 360 equivalent to the BUGS command -- the 'cmndline' is passed on to the CMS command handler.

CP

```
| CP cmndline |
```

The CP command functions as the BUGS and CMS commands, except that the 'cmndline' is passed to CP.

DEBUG

```
| DEBUG |
```

The DEBUG command provides an interface between the ICOPS command language monitor and the debugging environment on the local machine (i.e., the machine on which the DEBUG command was entered).

DISK

```
| DISK filename |
```

It was anticipated that the user of ICOPS would fairly frequently want to enter a prespecified set of commands each time he ran his application. The DISK command allows him to create a file whose name is 'filename' (on either machine) and then request ICOPS execution of each command in the file.

EXTERNAL

```
| EXTERNAL |
```

This command "gives back" to the user the use of the external and panel interrupts which were usurped by the ICOPS monitor. After the monitor is entered, execution of the EXTERNAL command

will cause the local debugging environment to be entered just as if the monitor were not in control.

IOLOG

```
| IOLOG ON|OFF|PRINT |
```

The IOLOG command is used to turn on, turn off, or print the current statistics on I/O activity across the mainframe/satellite link. The statistics kept on the link are the total number of READs and WRITEs issued and the total amount of data transfer in each direction.

ITRACE

```
| ITRACE ON|OFF |
```

ITRACE (for Internal trace) is a command allowing the ICOPS designers to trace the execution of a preselected set of ICOPS system modules. Included in the trace are the names of the called and calling programs together with a selected set of pertinent parameter addresses.

LOCAL PROCS

```
| LCL |
```

The LOCAL command requests a display of the names of all procedures which are currently on the local processor. The monitor responds with such a list of names.

LOCATION

```
| LOCATION procname |
```

The LOCATION command is used tc determine the current location of the procedure named 'procname'. The monitor responds with either "LOCAL" or "REMOTE" depending on the location of the procedure.

MOVE

```
| MOVE procname |
```

The MOVE command is used to move a procedure from one processor to the other. The direction of tranfer is implicit in the current location of the procedure. Note that the actual procedure movement does not occur until the first time the procedure is called after the move request is issued. When the procedure is moved, a message is printed on the user's terminal giving the new location of the procedure.

PRINT STATISTICS TABLE

```
| PST procname |
```

This command is used to display on the user's terminal the current statistics for 'procname'. The monitor responds with a display of the current statistics which the user previously specified for collection.

PRINT PROCEDURE TABLE ENTRY

```
| PTE procname |
```

The PTE command displays the current status of the PTE for the procedure 'procname'. The command is mainly for use by the ICOPS system designers.

REMOTE

```
| RMT |
```

The REMOTE command is completely analogous to the LOCAL command. The result is a list of all ICPable procedures currently on the remote processor.

STATISTICS

```
| STAT procname|ALL ON|OFF OPT= o STATS= s FILE= f |
```

The STATISTICS command controls the taking of statistics for a particular procedure ('procname') or all ICPable procedures (ALL). The three possible functions are to initiate or terminate (ON or OFF) statistics collection. When statistics collection is initiated, the OPT and STATS keywords control what statistics are taken and where they are displayed. The OPT keyword is a string composed of any or all of the following letters:

S Gather statistics for this procedure. (If 'S' is not specified, only procedure calls and returns will be traced.)

M Gather statistics on procedure movement.

D Write statistics records to a disk file.

T Write statistics records to the terminal.

The STATS keyword specifies what statistics are to be kept and is a string composed of any or all of the following:

C Trace the name of the calling procedure.

D Trace dedicated I/O operations (disk and tape) performed by the procedure.

L Trace I/O activity across the mainframe/satellite link caused by the procedure.

P Trace the number of page reads incurred by the procedure (360 side only).

T Trace the elapsed real ("clock") time spent while in the procedure.

U Trace the number of unit-record I/O operations (card reader, card punch, printer) performed by the procedure.

V Trace the elapsed virtual CPU time spent in the procedure. (The virtual CPU time is equal to the real time minus time spent waiting for paging operations, CP overhead, etc.)

$ Trace the cost of each call of the procedure (according to the accounting system of the processor on which the procedure is running).

The 'file=' option specifies the name of the CMS disk file where the statistics will be written. The filetype is always TRACE.

Default values are OPT= ST STATS= CDLPTUV$ FILE= ICPTRACE.

```
| STAT LOG FILE= fname |
```

The LOG command writes a statistics record to disk. The 'file=' option specifies the name of the CMS disk file. The filetype is TRACE and the default filename is ICPTRACE.

WART CONTROL BLOCK

```
| WCB procname |
```

The WCB command displays the current status of the procedure 'procname'. This command is for use by the ICOPS system designers.